UMONS
Université de Mons

LIÈGE université

Faculté Polytechnique

Faculté de Gembloux Agro-Bio Tech

Architecture cloud distribuée orientée services, dédiée à l'Internet des objets

THÈSE

en cotutelle UMONS - ULiège

Présentée en vue de l'obtention des grades de Docteur en Science de l'Ingénieur et Technologie et de Docteur en Sciences Agronomiques et Ingénierie Biologique

Par

Olivier DEBAUCHE

Jury:

Pr. Mohammed BENJELLOUN	*Université de Mons*	**Président**
Pr. Saïd MAHMOUDI	*Université de Mons*	**Promoteur**
Pr. Pierre MANNEBACK	*Université de Mons*	**Co-promoteur**
Pr. Frédéric LEBEAU	*Université de Liège*	**Promoteur**
Pr. Sébastien BETTE	*Université de Mons*	**Secrétaire**
Pr. Jérôme BINDELLE	*Université de Liège*	**Membre**
Pr. Pierre TOCQUIN	*Université de Liège*	**Membre**
Pr. Nicolas GENGLER	*Université de Liège*	**Membre**
Dr. François PINET	*INRAE (France)*	**Membre**
Pr. Pascal BOUVRY	*Université du Luxembourg*	**Membre**

POLYTECH MONS

LIÈGE université Gembloux Agro-Bio Tech

Thèse soutenue publiquement le 15 juin 2022

pôle hainuyer

Pôle de académique Namur

À Dimitri XANTHOULIS, mon mentor qui n'a pas pu assister à la présentation de ma thèse, emporté trop tot par la maladie ce 20 Août 2021.

Résumé

Cette thèse de doctorat, fruit de la collaboration entre les universités de Mons et de Liège (Belgique), traite de la conceptualisation et le développement d'une architecture cloud distribuée polyvalente destinée à la gestion des données dans le domaine du Smart Farming. Cette architecture est suffisamment générique pour être utilisée dans d'autres domaines.

Les chercheurs sont poussés par les bailleurs de fonds à conserver et échanger leurs bases de données expérimentales et de tests en vue de les valoriser dans le cadre d'autres projets. La réutilisation des données est motivée par la possibilité d'investir dans une plus vaste gamme de projets en évitant les redondances liées notamment aux données. Notre approche s'inscrit dans le cadre de l'Open Data et de l'Open Science où des architectures distribuées sont utilisées pour le stockage massif de données.

De nombreuses architectures et plateformes IoT génériques existent sur le marché répondant à des besoins variés. Mais, il y a un manque d'outils spécialisés pour la recherche et leur valorisation d'une part et adressant les besoins spécifiques de communautés de chercheurs d'autre part. De plus, les plateformes existantes restent tributaires de la maintenance et de la volonté de pérennisation de la société et/ou de la communauté qui les développent. En matière de recherche scientifique, des plateformes existent sous forme d'écosystèmes cloisonnés la plupart du temps ce qui ne permet pas une valorisation industrielle praticable des recherches menées.

Fort de ces constats, nous proposons dans cette thèse de doctorat, de concevoir une architecture Cloud spécifique au Smart Farming, durable, améliorable et adaptable en fonction des cas d'utilisation sans remettre en cause l'ensemble de l'architecture. Nous proposons également la mise en place d'une chaîne de valeur partant de l'acquisition des données, leur traitement et stockage, l'hébergement d'applicatifs permettant leur exploitation jusqu'à la valorisation et leur exploitation par l'utilisateur final.

Nos recherches s'appuient sur un cas d'utilisation concret permettant de mettre en relief les limites d'utilisation que l'architecture cloud doit pouvoir adresser. Ce cas d'utilisation est l'analyse comportementale des animaux de ferme au pâturage.

Les chercheurs sont incités de plus en plus à conserver et échanger leurs données ce qui se traduit par des besoins en matière de pérennisation de leur infrastructure, de traçabilité et de documentation de leurs données, et de standardisation de leurs outils. Ils sont également amenés à développer des chaînes de traitement en temps réels ou par lots pour traiter des données provenant de sources multiples et dans des formats variés.

Notre architecture se veut innovante, modulable, adaptable à un large panel de cas d'utilisation sans avoir à remettre en question sa structure ni ses briques logicielles constitutives. L'utilisation de composants logiciels interchangeables, rendent durable l'architecture et l'immunise vis-à-vis de la disparition de l'un de ses composants logiciels. A contrario, une brique logicielle peut être remplacée par une autre plus adaptée ou plus performante. De plus, elle offre la possibilité d'héberger et par la suite de monétiser les applicatifs développés par les chercheurs. Sa composante Edge Computing (capacité de traitement située en périphérie du réseau) permet de déployer grâce aux techniques de conteneurisation des micro-services et des algorithmes d'Intelligence Artificielle (IA) adaptés au plus près des capteurs.

Mots clés : Architecture Cloud, Architecture Lambda, Architecture Edge AI-IoT, Edge Computing.

Abstract

This PhD thesis, a collaboration between the Universities of Mons and Liege (Belgium), deals with the conceptualization and development of a versatile distributed cloud architecture for data management in the Smart Farming domain. This architecture is generic enough to be used in other domains.

Researchers are pressured by funders to maintain and share their experimental and test databases for use in other projects. Data reuse is motivated by the possibility of investing in a wider range of projects while avoiding data-related redundancies. Our approach is in line with the Open Data and Open Science framework where distributed architectures are used for massive data storage.

Many generic IoT architectures and platforms exist on the market to meet various needs. However, there is a lack of specialized tools for research and their valorization on the one hand and addressing the specific needs of communities of researchers on the other hand. Moreover, the existing platforms remain dependent on the maintenance and the will of the company and/or the community that develops them. In terms of scientific research, platforms exist in the form of ecosystems that are mostly compartmentalized, which does not allow for a practical industrial valorization of the research conducted.

Based on these findings, we propose in this PhD thesis to design a Cloud architecture specific to Smart Farming, sustainable, improvable and adaptable according to the use cases without calling into question the whole architecture. We also propose the implementation of a value chain starting from the acquisition of data, their processing and storage, the hosting of applications allowing their exploitation until the valorization and their exploitation by the final user.

Our research is based on a concrete use case that highlights the limitations that the cloud architecture must be able to address. This use case is the behavioral analysis of farm animals at pasture.

Researchers are increasingly encouraged to preserve and exchange their data, which

translates into needs for the durability of their infrastructure, traceability and documentation of their data, and standardization of their tools. They also need to develop real-time or batch processing chains to handle data from multiple sources and in various formats.

Our architecture is innovative, modular and adaptable to a wide range of use cases without having to question its structure or its constituent software bricks. The use of interchangeable software components makes the architecture durable and makes it immune to the disappearance of one of its software components. On the other hand, a software brick can be replaced by another one that is more adapted or more efficient. In addition, it offers the possibility of hosting and subsequently monetizing the applications developed by researchers. Its Edge Computing component (processing capacity located at the edge of the network) enables the deployment of micro-services and Artificial Intelligence (AI) algorithms adapted as close as possible to the sensors, using containerization techniques.

Keywords : Cloud Architecture, Lambda Architecture, Edge AI-IoT Architecture, Edge Computing.

Remerciements

Cette thèse de doctorat, fruit de la collaboration entre l'université de Mons (Belgique), instituts InforTech et Numédiart et l'université de Liège, Gembloux Agro-Bio Tech, TERRA/BioDynE, met en exergue la conceptualisation et le développement d'une architecture cloud distribuée polyvalente destinée à la gestion des données dans le domaine du Smart Farming et pouvant également être utilisée dans d'autres domaines.

Je tiens à remercier les universités et les structures qui m'ont accueilli durant la durée de mes recherches ainsi que toutes les personnes qui ont contribué à faire de moi le chercheur que je suis devenu aujourd'hui.

J'aimerais tout d'abord adresser mes remerciements à mes promoteurs de thèse, les professeurs Saïd Mahmoudi (UMONS), Frédéric Lebeau (Gembloux Agro-Bio Tech - ULiège) et mon co-promoteur le professeur Pierre Manneback (UMONS), pour m'avoir soutenu, guidé, encouragé et conseillé durant toute la durée de mes recherches. Je désire leur témoigner toute ma gratitude pour le temps qu'ils m'ont consacré et toutes les opportunités tant en matière d'enseignement, de recherche, ainsi que les compétences transversales qu'ils m'ont permis de développer durant de cette thèse.

Je tiens également à remercier le président le professeur Mohammed Benjelloun et les membres de mon comité d'accompagnement : les professeurs Sébastien Bette (UMONS) et Jérôme Bindelle (Gembloux Agro-Bio Tech - ULiège) pour l'évaluation et le suivi annuel de mes recherches. Leurs critiques, suggestions, encouragements et conseils m'ont permis d'avancer efficacement dans mes travaux et de reconsidérer mon approche pour la rendre plus claire et pertinente.

Je remercie également les membres de mon jury : le professeur Nicolas Gengler (Gembloux Agro-Bio Tech - ULiège) et le professeur Pierre Tocquin (ULiège) ainsi que les membres extérieurs : le professeur Pascal Bouvry (Université du Luxembourg) et le docteur François Pinet (INRAE) d'avoir accepté de faire partie de mon jury de thèse, consacré du temps à la lecture et l'analyse minutieuse de ce document de thèse.

Je remercie le professeur Gaëtan Libert ex-chef service qui m'a permis d'intégrer le service en qualité d'assistant à mandat et permis de prendre une revanche sur la vie en réalisant ma thèse de doctorat dans son service.

Je remercie mes collègues : Adriano Guttadauria, Mohammed El Adoui, Sidi Ahmed Mahmoudi, Michel Bagein, Rim Doukha, Mohammed Amine Larhman, Mohammed Amin Belarbi, Yassine Amkrane, Fabrice Bellarmin Nolack Fote, Amine Roukh, Sébastien Frémal, François Roland, et Justine Plum pour ces cinq années de partage, bonne humeur et complicité. Ils m'ont permis de progresser énormément sur les aspects techniques.

Je remercie le professeur Ghalem Belalem (Université d'Oran 1, Ahmed Ben Bella – Algérie) qui m'a beaucoup aidé par la définition des contours de ma primo architecture cloud.

Je remercie mes mentors qui m'ont laissé quelques citations qui ont jalonné ma vie et font encore écho aujourd'hui : Prof Charles Debouche (« *Jamais rancunier plus d'un quart d'heure* »), Prof Sylvia Dautrebande (« *Quand on n'est pas en avance, on est en retard* »), Prof Dimitri Xanthoulis («*Sois critique avec toi-même avant que les autres ne le soient avec ton travail* »).

Je pense ensuite à mes parents qui ont financé mes études universitaires et m'ont permis d'être là où je suis aujourd'hui.

Je remercie mes amis les professeurs Abderrahmane Bouamri et Mustapha Fagroud (ENA – Meknès - Maroc) qui m'ont encouragé à faire cette thèse et pour leur soutien indéfectible.

A ma muse et complice qui a été à mes côtés chaque jour pour me soutenir, m'encourager, m'aimer inconditionnellement, effacer mes doutes et mes peurs.

À Nora, ma fille, merci pour toute la tendresse et l'amour que tu m'as donnés et qui m'ont permis de dépasser les moments difficiles.

Finalement, à toutes les personnes qui ont croisé mon chemin durant ma thèse et qui au détour d'une conversation ont partagé leurs expériences, conseils et suggestions.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

AES Advanced Encryption Standard
AWS Amazon Web Services
CC Cloud Computing
CDN Content Delivery Network
CSV Comma Separated Values
DCE Distributed Computing Environment
DSV Delimiter-Separated Values
EPC Electronic Product Code
ETL Extract Transform Load
GCM Galois/Counter Mode
GRU Gated Recursive Unit
GUID Global Unique Identifier
IIoT Industrial Internet of Things
IMU Inertial Measurement Unit
IoT Internet of Things
JSON JavaScript Object Notation
JSON-LD JSON for Linked Data
M2M Machine to Machine
MAC Message Authentication Code
MDC Micro Data Centers
MEC Mobile Edge Computing
Nessie New European Schemes for Signatures Integrity and Encryption
NSA National Security Agency
OS Operating System
OSF Open Source Foundation
OWL Web Ontology Language

PA Precision Agriculture
RDF Resource Description Framework
RDFS RDF Schema
REST REpresentational State Transfer
RFID Radio-Frequency Identification
RGPD Règlement général sur la protection des données
RSS Received Signal Strength
SDR Software Defined Radio
SHA Secure Hash Algorithm
SLA Service-Level Agreement
SWT Semantic Web Technologies
TSV Tab-separated values
UAV Unmanned aerial vehicle
uCODES Ubiquity Codes
UGV Unmanned ground vehicle
UUI Universally Unique Identifier
VRF Variable Rate Fertilizer
VS Variable Spraying
W3C World Wide Web Consortium
WEKA Waikato Environment
WSAN Wireless Sensor and Actuator Network
WSN Wireless Sensor Network
XML eXtensible Markup Language

1 Introduction

Ce chapitre introductif présente les différentes facettes de la problématique qui est à l'origine de cette thèse de doctorat et justifie sa réalisation. Les principaux objectifs de la thèse sont ensuite énoncés ainsi que les principales contributions scientifiques. L'organisation du document est ensuite présentée. Chaque chapitre est accompagné d'une description succincte pour guider le lecteur dans sa découverte du document et susciter son intérêt.

1.1 Contexte

L'Internet des objets est aujourd'hui présent dans tous les domaines de notre vie et suit une croissance exponentielle. Le nombre d'appareils connectés est estimé à l'horizon 2022 à 42,5 milliards et 2025 à 75,5 milliards [1] et le trafic IP mondial estimé à 333 ZB par mois en 2022 [2] avec la nécessité de stocker et de traiter ces données [1]. La Commission européenne a prédit que 18 milliards des 29 milliards d'appareils connectés seront liés à l'Internet des objets en 2022 [2]. Cisco dans un livre blanc a annoncé que les appareils connectés à Internet généreront 850 ZB/an en 2021 [3]. Il est difficile de déterminer précisément le nombre d'objets connectés à l'échelle mondiale, mais leur nombre est de plusieurs milliards. En outre, McKinsey Global Institute prévoit un impact économique total des appareils IoT et Edge Computing qui atteindra 11 000 milliards de dollars d'ici 2025 [4]. Dans le secteur de l'agriculture, près de 12 millions de capteurs agricoles se-

1. https ://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
2. https ://www.statista.com/statistics/499431/global-ip-data-traffic-forecast/

ront installés dans le monde d'ici 2023 avec une augmentation de 20% par an, selon le Business Insider Intelligence Service [5]. De plus, le secteur de l'agriculture intelligente était estimé à 13,8 milliards USD en 2020 et devrait atteindre 22 milliards USD d'ici 2025, avec un taux de croissance annuel composé (TCAC) de 9,8% [5].

Le contexte dans lequel s'inscrit cette thèse de doctorat est la problématique d'ingestion des grandes quantités de données produites par les capteurs IoT en Smart Farming ainsi que les données externes provenant d'autres sources pour ensuite les exploiter au travers de services. Outre ce contexte général lié au Smart Farming, s'ajoute les contraintes particulières liées à la recherche scientifique et au besoin de conserver l'ensemble des données sur le long terme, de pérenniser et de standardiser leurs outils de recherche.

Les recherches et projets utilisant l'Internet des objets sont confrontés aux défis d'ingestion et de stockage de masses importantes de données tout en assurant une qualité de service constante. Ces quantités colossales de données nécessitent l'utilisation d'architectures spécifiques pour leur stockage et leur traitement. Les quantités de données actuellement produites par les capteurs intelligents sont telles qu'il n'est plus possible de transférer toutes ces données dans les réseaux télématiques sans générer d'importantes congestions.

Bien que de nombreuses communautés de scientifiques ont développés des solutions propres à leur domaine d'expertise comme par exemple Galaxy [3] dans le domaine la bioinformatique ou le projet européen Fiware [4] devenu entretemps une fondation, ces plateformes sont des écosystèmes fermés ou semi-fermées où les utilisateurs sont soit bloqués par le type de licence de la plateforme ou un droit d'entrée (par exemple : sous forme de licence, payement d'une cotisation, etc.) pour pouvoir contribuer. Ces barrières sont autant de freins pour une valorisation transparente de la recherche qui implique soit le payement de licences pour sortir de l'écosystème soit une réécriture des développements sur un nouvelle base pour éliminer les contraintes liées aux licences afférents aux briques logicielles.

Les développements d'outils et de plateformes s'appuient sur des briques logicielles dont les licences impliquent un niveau de restriction plus ou moins contraignant sur l'utilisation et la valorisation, éventuellement additionné d'un effet contaminant sur les développements les implémentant. De plus, la vitesse de développement de ces briques logicielles est fonction du degré d'investissement et de R&D de la société qui les commercialise ou du dynamisme de la communauté qui les supporte dans le cas de l'open source. Par ailleurs, la nature des licences peut évoluer dans le temps avec des implications notables sur les développements qui ont été réalisés sur leur base. Par exemple : Nous mesurons aisément les implications du changement de la licence de Java (Oracle) sur les briques logicielles l'ayant utilisé pour leur développement.

3. https ://galaxyproject.org/
4. https ://www.fiware.org/

La pérennisation des développements peut être assurée par l'utilisation de briques logicielles payantes mais cela a un coût non négligeable et rend le développeur dépendant d'une ou plusieurs sociétés commerciales. A contrario, l'utilisation de logiciels libres est moins onéreuse mais les développements peuvent s'interrompre subitement et impliquer une refactorisation du code pour changer de brique logicielle. Néanmoins, l'utilisation de briques logicielles soutenues par des fondations sont généralement d'une meilleure qualité et assure une durabilité. Ces briques logicielles sont généralement sponsorisées par une société commerciale et développées par une communauté de développeurs dynamiques.

La relation avec les plateformes cloud pour l'utilisateur final reste toujours quelque chose de nébuleux. On comprend aisément les difficultés que peuvent éprouver ces utilisateurs à confier leur données personnelles et/ou de production à des systèmes sur lesquels ils n'ont pas la main, ne savent pas toujours où ils sont localisés et ce qui est réellement fait de leurs données.

De plus, la question du respect des droits des utilisateurs reste posée notamment la récupérabilité des données dans des délais raisonnables et leur utilisation à d'autres fins que celles consenties. Apporter des garanties à l'utilisateur sur les traitements qui sont réalisés sur ces données afin de créer la confiance est crucial pour la réussite d'un projet de recherche ou d'un produit commercial.

De manière plus pragmatique quel est l'avantage pour les parties prenantes à transmettre et/ou céder ses données et quels bénéfices en retirent-elles ? Quel business model adopter pour qu'elles bénéficient d'un avantage à alimenter les sources de données utilisées par les scientifiques pour mener leurs recherches ?

L'analyse des constats et contraintes montre qu'il existe un manque en matière de plateforme pérenne, dédié à la recherche impliquant les utilisateurs finaux et facilitant un processus de valorisation post recherche fluide et la création d'une communauté d'utilisateurs.

1.2 Motivation de la thèse

L'ingestion de données en grandes quantités produites notamment par les objets connectés issus d'écosystèmes fermés ou semi-fermés dans des formats de données propriétaires ou non, complique drastiquement leur interopérabilité et les possibilités d'exploitation. A cela s'ajoute le manque d'outils spécialisés pour adresser les problématiques spécifiques de communautés de chercheurs. Ces constats ont amené les chercheurs à traiter ces problématiques de pérennisation des outils, la standardisation des données et des outils, le retraitement, la valorisation des anciennes données stockées dans des formats variés. Certaines communautés notamment dans le domaine de la génomique et de l'imagerie médicale ont rapidement développé leurs propres solutions qui sont généralement des écosystèmes fermés ou semi-fermés.

La question la plus fondamentale que les chercheurs se posent est : "Faut-il participer au développement des communautés existantes ou développer une nouvelle communauté répondant aux spécificités de leur domaine de recherche ou encore utiliser des outils existants dans un contexte pour lequel ils n'ont pas été prévus ?" La participation à une communauté existante offre la possibilité d'utiliser rapidement des outils éprouvés mais ne garantit nullement le devenir de la communauté ni la direction que prendront les futurs développements ou encore les évolutions du type de la licence.

Le développement de ses propres outils ou d'une nouvelle communauté nécessite des efforts considérables en matière de ressources mais, permet d'avoir une complète maitrise du devenir des outils. Toutefois, la création d'une nouvelle communauté implique un démarchage important d'autres équipes de recherches afin d'obtenir leur adhésion et une taille critique de la communauté qui assure sa survie. Tandis que détourner des outils non prévus initialement pour un nouveau domaine de recherche peut fonctionner avec un intensité variable selon le degré de spécificité technique et normative, et des possibilités de développement et/ou adaptation des outils existants. Mais cela implique de continuellement actualiser les développements ou adaptations réalisées en fonction des outils d'origine pour garantir la compatibilité et l'interopérabilité.

Le choix du type de licence logicielle pour les développements a des implications directes sur les possibilités de valorisation et la mise à disposition éventuelle du code source. Par ailleurs, certaines licences ont un effet de contagion c'est-à-dire que les licences de certaines briques logicielles sont automatiquement transmises aux développements qui les incorporent ; c'est notamment le cas des licences GPL [5].

Les choix en matière d'architecture de traitement sont cruciaux et ont des implications directes sur les capacités et la qualité des traitements possibles, la vitesse de prise de décision et de réaction ainsi que sur les garanties offertes pour la protection de la vie privée.

Les cas d'applications en Smart Farming de par leur diversité impliquent de devoir supporter des variations importantes en matière de débits de données et de fréquences de transmission de données provenant de nombreux capteurs. Le besoin de prendre des décisions le plus rapidement possible lorsqu'un événement survient nécessite une ingestion rapide et une requêtabilité quasi-temps réelle des données, c'est-à-dire inférieure à une ou deux secondes après leur ingestion. Par exemple, on imagine aisément qu'une défaillance dans un système aéroponique puisse rapidement être dommageable si une réaction rapide n'a pas lieu.

1.3 Objectif de la thèse

L'objectif principal de cette thèse est de proposer une architecture cloud modulaire capable de s'adapter à un grand nombre de cas d'utilisation sans avoir à remettre en

5. https ://en.wikipedia.org/wiki/Viral_license

cause ses fondements.

Pour atteindre cet objectif l'architecture cloud devra garantir l'anonymisation des données, leur traçabilité, le contrôle fin des accès, la récupération des données hébergées et maintenir les performances avec l'augmentation de la quantité de données hébergées. Par ailleurs, les services seront proposés sous forme de micro-services capables à la fois de gérer la sécurité, la traçabilité et l'anonymisation des données à partir des objets au niveau de la passerelle qui centralise les données.

Notre architecture s'inscrit également dans un contexte de recherche, elle devra permettre de gérer et stocker de manière pérenne de grandes masses de données. En effet, l'ensemble des données brutes et traitées issues des expérimentations doivent être conservées afin de pouvoir être réutilisées, valorisées et mises à disposition d'autres projets de recherche.

1.4 Contributions scientifiques de la thèse

Dans le cadre de cette thèse de doctorat, trois contributions principales ont été réalisées ainsi qu'une contribution secondaire qui vient en appui des contributions principales et les exploitent.

Nos principales contributions sont :

1. Le développement d'une **architecture cloud modulaire** destinée à la recherche scientifique et adaptable à un grand nombre de cas d'utilisation principalement dans le domaine du Smart Farming. Cette architecture est composée de composants de collecte, stockage et traitement des données et d'une architecture d'hébergement d'applicatifs.
2. L'**automatisation du traitement et de l'ingestion des données collectées** à l'aide des métadonnées associées aux informations (données brutes accompagnées de métadonnées descriptives). Les métadonnées sont encodées par les chercheurs préalablement à l'envoi des données. Des services adaptent l'architecture pour préparer la chaîne de traitement des données.
3. Le développement d'une **architecture Edge AI-IoT** pour le déploiement de micro-services et d'algorithmes d'intelligence artificielle adaptés. Cette architecture s'appuie sur l'architecture cloud développée.

Notre contribution secondaire est :

4. Le développement d'un **service en ligne d'analyse du comportement des animaux de ferme** à partir des données acquises par un collier doté d'une centrale inertielle sur le cou des animaux. Ce service est également hébergé sur la partie applicative de l'architecture cloud.

1.5 Organisation du document

Dans ce document, nous adoptons les conventions de couleur suivantes : (1) Les liens vers les tableaux et figures sont en rouge ; (2) Les liens vers références bibliographiques sont en vert ; (3) Les liens vers les concepts généraux sont encadrés en rouge ; (4)

Ce document se compose de neuf chapitres organisés comme suit :

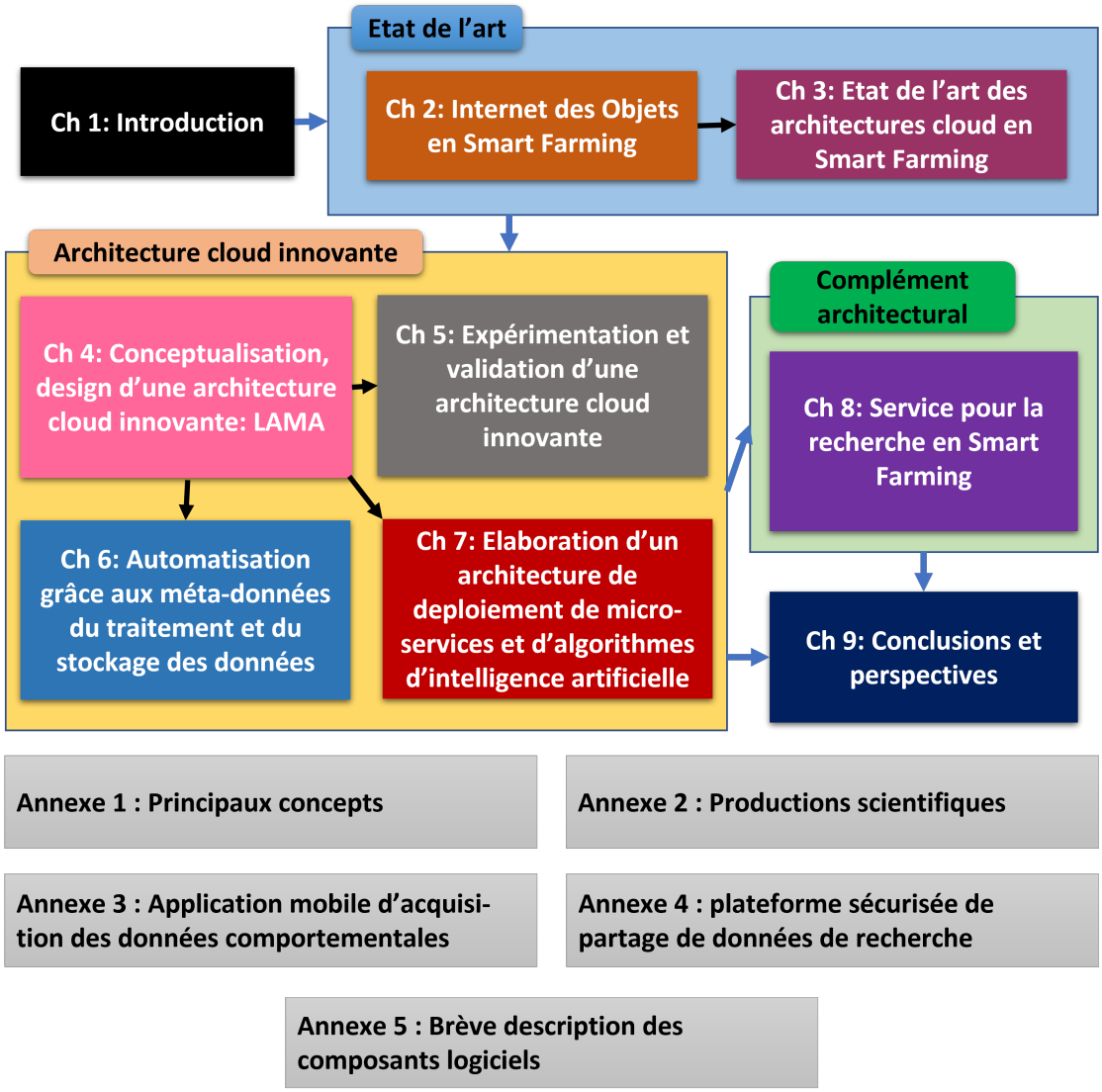


FIGURE 1.1 – Organisation et liens entre les chapitres de la thèse

Chapitre 1 : Introduction

Ce premier chapitre dresse les constats de difficultés de différentes parties prenantes (chercheurs, développeurs, utilisateurs) et les contraintes liés à l'Internet des objets et plus spécifiquement au Smart Farming. Ces constats permettent de décrire le besoin et justifie la réalisation de cette thèse de doctorat. Nos objectifs de recherches sont ensuite décrits ainsi que les principales contributions scientifiques.

PARTIE 1 : ETAT DE L'ART

Cette première partie s'articule autour de deux chapitres consacrés respectivement à l'état de l'art de l'Internet des objets qui aborde d'une manière générale les problématiques rencontrées et des architectures cloud en Smart Farming qui lève voile sur les spécificités de l'IoT en Agriculture.

Chapitre 2 : L'Internet des Objets en Smart Farming

Ce second chapitre pose le cadre et décrit les concepts généraux de l'Internet des objets, ses défis, et ses challenges. Nous décrivons ensuite l'Internet des objets appliqués à deux cas d'utilisation issus de la recherche en Smart Farming pour en dégager les particularités, les défis et les problèmes.

Chapitre 3 : Etat de l'art des architectures cloud en Smart Farming

Nous poursuivons ensuite par l'état de l'art des architectures centralisées, distribuées et les nouvelles tendances pour la collecte, le traitement et le stockage des données produites par les objets connectés en Smart Farming. Le chapitre se conclut par une analyse critique des différentes architectures décrites dans l'état de l'art.

PARTIE 2 : ARCHITECTURE CLOUD INNOVANTE : CONCEPTUALISATION ET TEST

La deuxième partie est organisée en 4 chapitres qui décrivent successivement la conceptualisation, le design, la mise en œuvre et l'évaluation d'un architecture innovante appelée LAMA.

Chapitre 4 : Conceptualisation et design d'une l'architecture cloud innovante

Les conclusions tirées de l'analyse de l'état de l'art, nous amènent à conceptualiser progressivement, une architecture cloud distribuée spécifique à la recherche en agriculture numérique permettant d'adresser les deux cas d'utilisation évoqués dans le premier chapitre. La conceptualisation aboutit à la description détaillée d'une architecture en deux parties dédicacées à la collecte, au traitement et au stockage des données d'une part et à l'hébergement des applications développées par les scientifiques d'autre part.

Chapitre 5 : Expérimentation et validation d'une architecture cloud innovante

Ce chapitre décrit la mise en œuvre de notre proposition d'architecture innovante. Les composantes logicielles de la partie traitement de données sont comparées et testées à l'aide de données réelles produites par le cas d'utilisation décrit au chapitre 1. Le stockage des données est quant à lui été testé à l'aide d'un benchmark spécifique de référence. Les performances de l'architecture de traitement sont ensuite comparées à celles d'une architecture Kappa).

Chapitre 6 : Automatisation grâce aux métadonnées du traitement et du stockage des données

Nous montrons l'apport des métadonnées pour l'adaptation et l'automatisation des processus de traitement et de stockage des données dans une architecture cloud dédicacé, à l'Internet des objets. En outre, ces informations sont également utilisées pour l'exploitation des données et l'extraction de connaissances.

Chapitre 7 : Élaboration d'une architecture pour le déploiement de micro-services et d'algorithmes d'intelligence artificielle

Le septième chapitre couvre l'élaboration d'une architecture Edge AI-IoT qui vient en appui de l'architecture cloud développée au chapitre 4 pour permettre le déploiement de micro-services d'exploitation et de traitement des données avant leur envoi sur le cloud et d'algorithmes d'intelligence artificielle spécifiquement adaptés.

PARTIE 3 : COMPLEMENT ARCHITECTURAL

La troisième partie contient une contribution secondaire développée pour tester la plateforme applicative.

Chapitre 8 : Service pour la recherche en Smart Farming

Ce chapitre décrit le service d'analyse du comportement des animaux de ferme hébergé sur la partie applicative de l'architecture LAMA conceptualisée dans le chapitre 4 et testée dans le chapitre 5.

Chapitre 9 : Conclusion et perspectives

Finalement, le dernier chapitre conclut ce travail de recherche en rappelant les quatre contributions : (1) Une architecture modulaire ; (2) L'utilisation de la sémantique pour l'adaptation d'une architecture Lambda ; (3) l'utilisation de micro-services pour l'exploitation des données stockées ; (4) La mise à disposition d'un service d'analyse du comportement. Nous traçons ensuite les perspectives de notre travail.

Annexes

L'Annexe 1 : Définit les principaux concepts.

L'Annexe 2 : Reprend la liste exhaustive des productions scientifiques, accompagnées des résumés des articles publiés dans des revues.

L'Annexe 3 : Décrit les prémisses du développement de l'application mobile d'acquisition des données comportementales.

L'Annexe 4 : Contient la conceptualisation de la plateforme sécurisée de partage de données de recherche.

L'Annexe 5 : Propose une brève description des composants logiciels.

Première partie

Etat de l'art

2

L'Internet des Objets en Smart Farming

Plan du chapitre

Résumé

L'Internet des Objets (IoT) est un secteur en pleine expansion qui touche l'ensemble de la société et concerne tous les secteurs d'activités. Ce nouvel Internet est bien différent de celui des machines que nous connaissons et appréhendons au quotidien, étant donné le nombre de périphériques (nœuds) émettant généralement des données à faible débit et à une fréquence régulière. On estime, en 2020 qu'environ 50 à 100 milliards de périphériques produisent des données dans le cadre de l'IoT. Toutes les données produites par ces myriades d'objets doivent ensuite être ingérées par des architectures de traitement et de stockage adaptées en vue d'être finalement exploitées. Ce nouvel Internet doit encore relever de nombreux défis inhérents notamment aux nombres d'objets hétérogènes interagissant, à la diversité de protocoles de communication utilisés et à la standardisation des données.

Parmi les nombreux domaines couverts par l'IoT, le Smart Farming a des spécificités intrinsèques telles que la taille importante des réseaux de capteurs, la présence de nombreux standards de données qui ne permettent peu ou pas d'interopérabilité. Ces capteurs sont responsables de variations importantes de charge au niveau des réseaux et des architectures cloud qui collectent, traitent et stockent ces données. Dans le contexte particulier de la recherche l'entièreté des données doivent être conservées suivant les modalités prévues de plan de gestion des données (Data Management Plan) qui prévoit une période pendant laquelle l'ensemble des données sont conservées, en son terme uniquement les résultats de traitement sont conservés pour une seconde période au-delà de celle-ci uniquement les résultats de traitement sont conservés. Ce plan de gestion prévoit une première période durant laquelle les données peuvent être réutilisées. Au terme de cette période, l'évolution technologique fait que les données perdent leur valeur et seul les résultats de traitements restent utilisables. A plus long terme, les résultats finaux sont stockés à froid comme valeur de comparaison pour mémoire. Les bailleurs de fonds (européen, nationaux ou régionaux) imposent la conservation des données de recherche voire leur diffusion à postériori ce qui implique des contraintes sur la conception des architectures qui doivent garantir leur performances malgré l'augmentation du volume de données stockées.

Publication liée à ce chapitre

O. Debauche, S. Mahmoudi, P. Manneback, J. Bindelle, S. A. Mahmoudi, A. Guttadauria, F. Lebeau, "Data Management and Internet of Things : a Methodological Review in Smart Farming". *Internet of Things*, 2021, **100378**. doi : 10.1016/j.iot.2021.100378.

L'IoT également appelé Internet ubiquitaire est un concept décrit pour la première fois par Kevin Ashton en 1999 [6]. Ce nouvel Internet est fondamentalement différent de l'Internet des machines que nous utilisons tous. Le réseau de l'Internet des Objets (Internet of Things (IoT)) est dynamique, distribué, caractérisé par un faible débit. Il est composé de plusieurs dizaines de milliards d'objets, en 2020, hétérogènes du point de vue fonctionnel et technique [7, 8] par opposition à l'Internet des machines qui est un réseau dans lequel transite de grandes quantités de données à grande vitesse. L'Internet des Objets fait référence à une infrastructure réseau globale, dynamique, pourvue de capacités d'auto-configuration basées sur des protocoles de communication standards et interopérables dans lequel des objets connectés utilisent des interfaces intelligentes et sont parfaitement intégrés dans le réseau d'information [9]. Selon le Cisco Internet Business Solutions Group [10], le nombre de nœuds Internet des objets devraient atteindre 50 à 100 milliards d'objets en 2020. Dans un avenir proche, l'impact économique des applications de l'Internet des objets a été évalué à 11% de l'économie mondiale. L'impact global annuel sur l'économie de l'Internet des objets est estimée entre 2,7 à 6,2 trillions de USD à l'horizon 2025 tandis que les communications entre machines (Machine to Machine (M2M)) devraient représenter 45% du trafic réseau mondial [11].

2.1 Les défis de l'Internet des objets et du cloud IoT

Outre les problèmes intrinsèques à l'Internet des objets, plusieurs défis nés de l'association du cloud et de l'Internet des objets ont vu le jour. Les défis les plus importants auxquels nous sommes confrontés dans le cadre de cette thèse sont évoqués brièvement dans les paragraphes suivants.

2.1.1 Le nombre d'objet connectés

Le nombre d'objets hétérogènes connectés interagissant sur le réseau est tellement important qu'il nécessite des infrastructures suffisamment extensibles (*scalable*) pour pouvoir tous les prendre en charge. Les volumes de données générés pas ces objets doivent pouvoir être transmis à travers le réseau. Toutefois, la transmission de ces grandes quantités de données diversifiées et non structurées ne va pas sans poser de problèmes en matière de collecte, de gestion, d'analyse et stockage [12]. Une solution pour le traitement et la gestion de ces quantités importantes de données est de le réaliser dans le cloud. Néanmoins, si toutes les données sont transmises vers le cloud au travers des réseaux, les énormes volumes de données transférés seraient la cause de congestion et de latences importantes dans les transmissions. Ces latences sont intolérables pour certaines applications critiques qui requièrent des traitements et/ou une mise à disposition des données des données ingérées en temps réel (<1s) ou quasi-temps réel (maximum quelques secondes). Ces applications critiques concernent notamment le domaine de la santé ou le pronostic vital peut être engagé, les chaînes de production ou encore

le Smart Farming. Une stratégie alternative à l'utilisation du cloud est de réaliser des traitements légers sur les capteurs quand leur capacités le permettent et de les compléter par des traitements plus conséquents à des niveaux intermédiaires situés entre les capteurs et le cloud afin de décongestionner les réseaux. Dans les cas où la disponibilité des données est cruciale, les traitements peuvent être réalisés en périphérie du réseau (*Edge Network*), au plus près des capteurs avec une latence réduite, sur des passerelles intelligentes (*Smart Gateway*) mais avec des possibilités de traitement et de stockage limitées. Les traitements peuvent également être réalisés à un niveau intermédiaire entre les passerelles et les clouds avec des latences plus importantes mais des capacités de traitement et de stockage plus conséquentes (*Fog computing*) dans des cloud locaux (*Local Cloud*) ou des routeurs qui servent de médiateurs entre les objets et le cloud.

Les défis sont d'une part de déterminer quelles données traiter, quand et à quel niveau et d'autre part de déterminer à quel moment transférer les traitements d'un niveau à un autre en fonction de la charge.

2.1.2 Les métadonnées contextuelles

Outre la génération de données provenant des capteurs, des données contextuelles (méta données) sont également générées pour compléter l'information comme par exemple : les propriétés de la mesure, la position du senseur, son état de fonctionnement, la précision de la donnée acquise. Ces données sont produites en même temps que les données brutes par les capteurs pour une récupération et un traitement ultérieur et sont également utilisés pour déterminer la réaction à une modification de leur environnement. L'ajout d'informations contextuelles apporte une valeur additionnelle et une sémantique à la donnée brute mais également une hétérogénéité supplémentaire en termes de formats de données utilisés.

La manière dont les quantités massives de données produites par la myriade d'objets connectés sont extraites et accédées mérite d'être revue pour permettre un accès facilité et plus aisée aux données pour les applications et les utilisateurs finaux. L'extraction de connaissances est réalisée par différentes machines pour fournir les services demandés. Cette extraction de connaissances inclut la découverte, l'utilisation des ressources ainsi que la modélisation des informations, la reconnaissance et l'analyse des données pour leur donner un sens. Elle permet ensuite de prendre de bonnes décisions en vue de fournir un service de bonne qualité [11]. La sémantique joue donc un rôle capital car c'est elle qui permet d'envoyer les demandes aux bonnes ressources à l'aide de technologies du web sémantique (Semantic Web Technologies,SWT).

2.1.3 La qualité des données

La qualité des données transmises est affectée par des incertitudes, de la redondance d'ambigüités et d'inconsistances. Les incertitudes peuvent notamment être mises en re-

lation avec des lecteurs d'étiquettes Radio-Frequency Identification (RFID) défectueux ou encore par la précision et la répétabilité de mesures effectués par des réseaux de capteurs sans fils (Wireless Sensor Network (WSN)). Les redondances sont soit dues à des lectures multiples au niveau des lecteurs RFID soit à la présence de capteurs sans fils similaires dans les environs. Les ambiguïtés sont dues à l'importance des données en fonction du contexte d'analyse. Les inconsistances sont dues en particulier à la défaillance de lecteurs d'étiquettes RFID dans la chaîne de lecture. Ces inconsistances peuvent également être dues à la perte de paquets de données durant leur transmission ou encore la précision intrinsèque des capteurs observant un même phénomène. La présence de données incomplètes est observée durant la coopération entre appareils mobiles et objets distribués. La sémantique des données produites par les objets connectés est primordiale pour la compréhension entre objets et machines [13].

2.1.4 Confidentialité des données

La confidentialité des données peut être difficile à assurer durant la phase d'intégration des données provenant des nœuds. Les réseaux de capteurs hétérogènes sont utilisés pour collecter des données sur les individus, les produits et leur environnement. L'agrégation dans un deuxième temps de ces données permet de réaliser des recoupements qui peuvent donner lieu à des problèmes de respect de la vie privée. En effet, durant cette phase, les attributs des nœuds et les données sont intégrées et révèlent par conséquent des informations quant à leur localisation et leur activité. Il est alors possible de réaliser des profilages des utilisateurs des objets connectés, de les localiser ou encore de les suivre [14]. Dans certains domaines d'application de l'Internet des objets, la protection de la vie privée est particulièrement sensible : maison intelligente (*Smart Home*), villes intelligentes (*Smart Cities*), Santé (*Healthcare*), données médicales (*Medical Data*) [14]. De manière plus large, les secrets de fabrication ou de production sont aussi concernés pour l'agriculture connectée (*Smart Farming*), l'élevage connecté (*Livestock Farming*) et l'Internet des objets industriel [Industrial Internet of Things (IIoT))].

Malgré un nombre croissant d'algorithmes d'anonymisation des flux et des processus de préservation de la vie privée proposés, ces algorithmes ne tiennent pas compte de la réidentification des données anonymisées. Les utilisateurs ne sont pas du tout conscients des risques en matière de vie privées liés à leurs ressources [14].

2.1.5 Standardisation des données et des services

Les données produites par les objets ne suivent actuellement pas de standard unique. Il n'y a, pour l'heure, aucun moyen de décrire les objets et leurs capacités d'une manière uniforme ce qui pose des difficultés aux agents logiciels (Software Agents) pour découvrir les objets de manière automatique ou orchestrer les nœuds, les données

ou encore les services [12]. Pour remédier à ce problème, l'OASIS [1] (*Organization for the Advancement of Structured Information Standards*) a proposé le profil de périphérique pour les services web (*DPWS*). Il revêt la forme d'un ensemble de spécifications indépendantes du langage et de lignes directrices pour décrire les nœuds et leurs capacités en tant que service en vue de favoriser leur interopérabilité. Cette approche est utilisée dans l'intergiciel OpenIoT [15] qui tend à devenir une référence [12]. L'intergiciel (*framework*) OpenIoT étend la sémantique des réseaux de capteurs (*SSN*) du W3C pour fournir un modèle commun avec l'objectif de pouvoir décrire de manière sémantique les capteurs physiques et virtuels et compléter le vocabulaire avec des concepts venant de l'intégration de l'IoT et du Cloud Computing. OpenIoT permet de combiner des flux provenant et des services issus d'applications diverses qui comportent des sémantiques incompatibles. OpenIoT facilite la découverte et le monitoring de capteurs et n'est pas limité aux capteurs physiques mais peut représenter tout dispositif capable de calculer et/ou d'acquérir des mesures d'un phénomène.

2.1.6 L'hétérogénéité des objets et des environnements cloud

Les objets connectés et le cloud sont tous les deux caractérisés par une grande hétérogénéité. Il faut gérer les différents types d'objets connectés pourvus de capacités et fonctionnalités hétérogènes ainsi que de différents protocoles ce qui ne va pas sans poser de problèmes d'interopérabilité. L'hétérogénéité des objets et le manque de standardisation dans le cloud empêche l'usage de services provenant de différents fournisseurs et affecte directement le développement le déploiement des applications. En effet, les applications sont par conséquent développées autour d'un fournisseur cloud et de ses services.

Le développement d'un intergiciel intercalaire (*Middleware*) capable de gérer l'hétérogénéité des objets connectés et des environnements cloud tout en permettant le développement et le déploiement sur de multiples services cloud offerts par différents prestataires est nécessaire. De plus, il est important de fournir aux utilisateurs et aux développeurs des mécanismes flexibles permettant de facilement basculer d'un service d'une prestataire donné vers un autre [12]. En Europe, la nouvelle règlementation RGPD, impose aux fournisseurs de services d'offrir la possibilité aux propriétaires de données de pouvoir les transférer vers un autre fournisseur de services.

D'autres problèmes liés à l'Internet des objets sont aussi des défis restant à relever mais ne sont pas adressés dans le cadre de cette thèse doctorat. Ils sont évoqués brièvement dans les paragraphes qui suivent pour communiquer une information complète au lecteur.

1. L'OASIS est un consortium mondial qui travaille pour la standardisation de formats de fichiers ouverts fondés notamment sur XML.

2.1.7 L'élasticité

Dans le paradigme cloud, les ressources sont allouées dynamiquement en vue de s'adapter au mieux aux besoins des utilisateurs et par conséquent de réduire les coûts de fonctionnement. Le principe d'élasticité n'a pas été pris en compte lors de l'intégration du cloud avec les objets connectés. Les nouveaux types de ressources comme les flux de données où l'approvisionnement de périphériques par les infrastructures Internet des objets ne sont pas livrées de manière élastique.

De nouvelles approches devront être développées pour intégrer la notion d'élasticité au cloud IoT.

2.1.8 La sûreté de fonctionnement

L'Internet des objets et les environnements cloud sont hautement dynamiques. Les objets connectés peuvent être indisponibles pour diverses raisons comme des défaillances, l'épuisement de la batterie, la perte de connectivité réseau, la mobilité des objets, etc. Le cloud peut également connaitre des indisponibilités, ou des dégradations de la qualité des services utilisés par les applications.

Il est par conséquent nécessaire d'adapter les applications pour gérer ces événements et assurer une réponse adéquate pour satisfaire les exigences non fonctionnelles telles que la disponibilité et la qualité. La sûreté de fonctionnement est critique pour les applications où la sécurité, dans la situation où des vies humaines, des pertes économiques ou des dommages physiques sont en jeu suite à une défaillance ou une perte de qualité de service.

2.1.9 Le support au développement d'applications

Il n'y a pas modèle de programmation standardisé pour élaborer des applications basées sur les objets connectés. Les éléments de l'Internet des objets : capteurs (*sensors*), actionneurs (*actuators*), passerelle (*gateway*), applications sont souvent programmés et déployés séparément sur base de services cloud.

Il est encore nécessaire de fournir des environnements pour supporter le développement d'applications basées sur les flux de données (data streams) générés par les nœuds et disponibles au travers du cloud.

2.1.10 Les modèles de virtualisation pour les nœuds

La première étape pour intégrer l'Internet des objets au cloud est d'avoir un modèle de virtualisation des capteurs, actionneurs et autres objets physiques afin d'exposer leurs fonctionnalités comme des services qui pourront être appelés par les applications.

La virtualisation de nœuds permet d'étendre les fonctionnalités des plateformes cloud et d'englober les objets physiques. Le modèle traditionnel d'approvisionnement (*provisioning*) doit être étendu en créant un modèle capteur et actionneur comme un service pour englober également la fourniture de ce type de service. Toutefois, ces approches de virtualisation doivent tenir compte des limites en matière de traitement et de mémoire des objets connectés.

Les standards web ouverts (*open Web standards*) sont préférables pour exposer les fonctionnalités des nœuds et réseaux aux tierces parties. Ces interfaces permettent d'utiliser parallèlement les outils cloud (*Cloud Tools*) et les plateformes cloud ouvertes (*Open Cloud Platforms*) et assure l'interopérabilité avec les systèmes existants et les applications dédiées à l'Internet des Objets.

2.1.11 L'identification des objets

Pour pouvoir identifier de manière unique les objets connectés, plusieurs standards ont été proposés. La fondation pour le logiciel ouvert [Open Source Foundation (OSF)] a développé l'identifiant universel unique [Universally Unique Identifier (UUI)] en tant que partie intégrante de l'environnement distribué de calcul [Distributed Computing Environment (DCE)] qui peut fonctionner sans coordination centralisée. L'OSF a également proposé l'identifiant unique global [Global Unique Identifier (GUID)]. D'autres méthodes comme le code de produit électronique [Electronic Product Code (EPC)] ou les codes omniprésents [Ubiquity Codes (uCODES)] permettent également d'identifier les objets[11]. Un besoin croissant d'identification multiple de chaque objet se fait sentir, pour permettre à chaque objet de s'adapter aux différents modes d'identification.

2.1.12 L'interopérabilité

L'interopérabilité est un autre défi lié au besoin de gérer de très grandes quantités d'objets hétérogènes qui appartiennent à des plateformes différentes. L'interopérabilité doit être gérée à la fois au niveau des développeurs d'applications, du matériel, des protocoles de communication, mais également au niveau sémantique. Les développeurs doivent donc concevoir leurs applications de manière à pouvoir permettre l'ajout de nouvelles fonctionnalités sans perturber ou perdre des fonctionnalités tout en maintenant l'intégration de nouveaux protocoles de communication. Cela a bien entendu des conséquences sur le design des applications[10].

L'utilisation de radio pilotée par logiciel (Software Defined Radio (SDR)) permet de gérer un large éventail de protocoles sur une large bande fréquences et de proposer un point d'accès unifié[16].

2.2 L'Internet des objets appliqué au Smart Farming

L'agriculture a connu deux vagues de révolution. La première était la mécanisation et la seconde appelée révolution verte avec les modifications génétiques [17]. Depuis la fin des années 1990, la transformation numérique de l'agriculture en Agriculture 3.0 aussi appelée Agriculture de Précision a commencé avec l'intégration du Système d'Information Géographique (SIG), des Systèmes de Positionnement Global (GPS) et l'utilisation de capteurs a envahi l'agriculture. Ils ont permis l'émergence du traitement d'images, des techniques utilisant le Deep Learning et le Machine Learning dans le domaine de la vision par ordinateur. La vision par ordinateur est mise en œuvre pour discriminer les mauvaises herbes, identifier les cultures, détecter les maladies,... La production d'une grande quantité de données par l'agriculture 3.0 a nécessité le développement de technologies big data pour les traiter, reflétant des changements importants dans divers domaines de recherche. Les données collectées doivent être enregistrées dans un format spécifique afin de permettre l'élaboration des modèles, de corriger les erreurs, d'éliminer les données dupliquées ou incohérentes ou de filtrer les problèmes de bruit dans les données [18].

Le Smart Farming aussi appelée Smart Agriculture ou Agriculture 4.0 est un domaine de l'IoT en pleine croissance qui apporte des pistes innovantes pour améliorer l'adaptabilité, l'efficacité et la résilience de l'agriculture des systèmes de production [19] booster la compétitivité et le profit [18], allouer les ressources de manière raisonnable et éviter le gaspillage de nourriture [20, 21] grâce à la contribution de la conscience autonome du contexte fournie par les capteurs et la capacité d'exécuter des actions autonomes ou à distance [21]. L'agriculture intelligente étend les application auparavant limitées à la ferme à des domaines connexes tels que la prise de décision par les agriculteurs, la biodiversité, la gestion des chaînes d'approvisionnement, la disponibilité et la qualité des aliments, l'assurance et la recherche en sciences de l'environnement et de la terre, etc.

Le Smart Farming se distingue des autres domaines de l'IoT par l'observation et l'action sur des objets biologiques (animaux ou végétaux). Il diffère de l'IoT médical par le fait qu'il n'y a pas de problèmes liés à la vie privée mais à la confidentialité des données liées aux processus de production. Comme dans la plupart des domaines de l'IoT, un réseau de capteurs et d'actionneurs sans fil (Wireless Sensor and Actuator Network (WSAN)) utilise un réseau à faible consommation et avec perte de données organisé en routage hiérarchique pour collecter des données et actionner des périphériques. Des protocoles de routage multi-chemins peuvent également être mis en œuvre pour équilibrer la charge liée au transfert des données et conserver l'énergie des nœuds généralement équipée d'une batterie à durée de vie limitée, de capacités de calcul de base, et d'un identifiant de communication unique. En raison de la durée de vie limitée de la batterie, qui est difficile voire impossible à recharger ou à remplacer [22], des techniques d'économie d'énergie et de récupération d'énergie doivent être appliquées

pour gérer le temps de fonctionnement actif et inactif et programmer la transmission des informations [18]. A ces objets peuvent s'ajouter les véhicules agricoles connectés, les robots de traite, les véhicules aériens sans pilote (Unmanned aerial vehicle (UAV)) communément appelés drones, les véhicules terrestres sans pilote (Unmanned ground vehicle (UGV)) également appelés robots, les appareils mobiles tels que les tablettes servant à encoder les observations ponctuelles [22] et des sources externes telles que les géo-services publics [18], des données collectées par des organisations gouvernementales et tierces distribuées à l'aide de dépôts ou de services en lignes, les données recueillies par les drones ou acquises par télédétection, données web en temps réel d'entreprises privées accessibles via des services web en ligne, les données acquises par crowdsourcing à l'aide de smartphone et les flux des médias sociaux [23].

Le niveau d'utilisation de l'IoT dans l'agriculture 4.0 varie de l'agriculture familiale comme par exemple en Inde, à très petite échelle avec quelques capteurs et actionneurs à faible coût à de très grandes échelles avec des milliers de capteurs commerciaux coûteux et de nombreuses machines agricoles connectées comme dans le Midwest américain. L'agriculture intelligente se caractérise comme mentionné ci-dessus par une grande variété d'objets qui peuvent produire une quantité de données très contrastée de quelques octets/s à plusieurs Gb/s. Dans le Smart Farming, les objets opèrent dans des environnements complexes (bâtiments agricoles, serres, dans le sol ou exposés aux intempéries) qui impactent d'une part la conception de ces objets mais également les transmissions de données qu'ils doivent opérer. Les besoins en termes de transmission de données sont très variables selon les cas d'utilisation variant de quelques bits pour la transmission ponctuelles de données environnementales ou physiologiques à plusieurs Mbits pour la transmission de flux vidéo provenant par exemple de drones ou de robots[24]. La disponibilité de protocoles réseau dans les zones rurales pour transmettre ces données impacte également le type d'architecture à mettre en œuvre. Les applications en Smart Farming nécessitent la collecte, le stockage, le prétraitement, la modélisation et l'analyse de grands volumes de données hétérogènes [23] produites par de nombreux capteurs provenant d'un grand nombre d'activités variées, de sources et éventuellement des mesures et/ou observations encodées manuellement [25]. Selon Kalimaris *et al.* [23], l'analyse des applications au travers de la littérature montre que certaines applications comme le calcul des indices d'assurance et amélioration de la productivité des agriculteurs requièrent de très grande quantité de données variées provenant de sources hétérogènes. Pour d'autres applications, c'est la fiabilité des données qui est cruciale.

Les applications en Smart Farming nécessitent des traitements en temps réel et/ou en temps différé. Les exigences « temps réel » sont également très variables selon les cas d'utilisation. Par exemple, le contrôle à distance des drones nécessite des temps de réaction d'au plus quelques millisecondes tandis que l'application d'engrais à taux variable (Variable Rate Fertilizer (VRF)) ou de pulvérisation variable (Variable Spraying (VS)) pour optimiser respectivement l'application des nutriments et des herbicides nécessite un temps de réaction compris entre quelques millisecondes et quelques secondes.

Le traitement en temps réel pour le suivi d'un troupeau de bovins est de l'ordre de quelques minutes à quelques heures. La durée de conservation des données est également très variable et dépend fortement du cas d'utilisation. Par exemple, les drones produisent d'énormes quantités d'images à transférer vers le cloud en temps réel où elles doivent être rapidement traitées et stockées. Ils peuvent également être post-traités pour extraire des données supplémentaires. Tandis que les images de robots perdent de leur valeur après traitement et une éventuelle action. Néanmoins, si les données sont de nature particulière, nouvelle ou exceptionnelle, elles peuvent être stockées en vue, par exemple, d'améliorer les algorithmes d'intelligence artificielle. Certains capteurs ne transmettent des données que lorsque des anomalies sont détectées tandis que d'autres transmettent à intervalles réguliers une petite quantité de données.

L'adoption de l'agriculture intelligente est entravée par le manque de modèles pour guider les parties prenantes sur la façon de mettre en œuvre et de déployer des systèmes de surveillance denses et hétérogènes basés sur l'IoT et de gérer leur interopérabilité [18]. De plus, les capteurs commerciaux sont très chers, ce qui rend impossible pour les petites fermes de les mettre en œuvre [26]. Selon Bahlo *et al.*, [27] l'agriculture adopte lentement l'interopérabilité dans les technologies de l'information, des efforts sont encore nécessaires pour intégrer l'open data, les standards et l'interopérabilité à plus large échelle. Le développement d'applications en Smart Farming nécessite d'intégrer des capteurs commerciaux en fonction des besoins en matière de précision et des budgets. Néanmoins, les vendeurs de solutions spécifiques ne fournissent pas de possibilité d'interagir avec d'autres capteurs ou dans le meilleur des cas une interopérabilité limitée [25]. L'intégration de capteurs spécifiques de différents fournisseurs reste problématique tant au niveau de l'accès aux données brutes que des algorithmes de traitement de ces données. De plus, deux tendances s'opposent actuellement. Celle des constructeurs de machines agricoles qui ont développé leurs écosystèmes et qui souhaitent étendre les services offerts aux agriculteurs en les attirant dans les écosystèmes captifs dans lesquels ils sont enfermés. D'autre part, une autre tendance est le développement d'écosystèmes ouverts dans lesquels les agriculteurs peuvent préserver la propriété de leurs données et garder le contrôle des traitements effectués sur ces données et de leur utilisation. Les agriculteurs sont donc confrontés à un dilemme où ils sont de toute façon contraints d'utiliser des équipements agricoles qui collectent leurs données contre leur gré et souhaitent d'autre part garder le contrôle de leurs données collectées via des capteurs IoT. Actuellement, il est difficile de prédire laquelle de ces deux tendances prendra le pas sur l'autre ou si les deux coexisteront [21]. Dans ce contexte, les chercheurs privés comme publics peuvent soit utiliser des plateformes commerciales génériques proposées par les acteurs du cloud sur lesquelles ils ont des possibilités d'adaptation limitées, soit développer leur propre architecture sur la base de briques commerciales ou gratuites mais avec des possibilités d'adaptation beaucoup plus grandes. Dans ce cas le choix est également délicat, et une mauvaise évaluation des contraintes peut remettre en cause le projet de recherche.

Pour les applications de recherche scientifique, la difficulté réside dans l'intégration des données produites par les capteurs avec les données historiques produites lors des précédentes études qui sont dans des formats de fichiers comme Comma Separated Values (CSV) ou d'archives qui sont difficiles à utiliser, analyser et exploiter et nécessitent par conséquent un processus de traitement et de mise en forme préalable [Extract Transform Load (ETL)] [25].

Le choix des capteurs est crucial car les nœuds peuvent se déplacer dans le temps, ce qui provoque des interférences considérables dans la communication. En outre, l'influence de la température, de l'humidité, des précipitations, de tempêtes de sable et du rayonnement solaire affecte considérablement les liens et la qualité de la communication entre les nœuds, en particulier dans des conditions environnementales difficiles. La température élevée rencontrée dans les environnements semi-arides affecte considérablement l'intensité du signal reçu (Received Signal Strength (RSS)). L'humidité liée au système d'irrigation et les fortes précipitations ont une incidence considérable sur la propagation des ondes radio. Le choix de l'émetteur-récepteur sans fil et du protocole de transmission doit tenir compte du nombre de nœuds, des distances entre eux, de la hauteur des antennes et de la fréquence de fonctionnement en fonction de la taille souhaitée de la charge utile [28].

Finalement, l'absence de cadre légal et de régulation sur la collecte, le partage et l'utilisation des données agricoles. Toutefois, plusieurs législations non spécifiques au secteur agricole influencent potentiellement la propriété, le contrôle et l'accès aux données[29].

2.3 Cas d'utilisation

Parmi les nombreux cas d'utilisations pratiques nous en avons choisi un pour définir les contraintes aux limites et définir les contours de cette thèse de doctorat.

Le cas d'utilisation intitulé « Vaches connectées » a pour objectif de déterminer le comportement des animaux de ferme à l'aide de capteurs GPS et inertiel (IMU) fixés sur le coup des animaux.

L'éthologie des animaux de ferme

L'analyse du comportement des animaux de ferme les étudie dans leur milieu naturel ou dans des environnements expérimentaux à l'aide de méthodes quantitatives et d'observations des mécanismes psychologiques ou physiologiques. L'éthologie permet de comprendre les comportements innés et appris et leurs causes. Elle aide à développer le bien-être animal, optimiser la production, améliore la gestion des espèces en voie d'extinction et de la faune en favorisant leur survie.

Les applications de l'analyse comportementale des animaux de ferme sont nombreuses, nous n'en citons que quelques-unes à titre d'illustration : état de santé des

animaux dans des pâturages distants de plusieurs kilomètres du siège d'exploitation, détection des chevauchements indiquant les périodes de chaleur chez les bovidés, détermination des préférences alimentaires des bovins. Les paramètres issus de la centrale inertielle (IMU) : accéléromètre 3D, gyroscope 3D et magnétomètre 3D sont échantillonnés à haute fréquence (100 Hz) et les données de positionnement par GPS à basse fréquence (0.5 Hz). La quantité de données générée par animal est de 1,2 Go par jour et par animal. Ces données sont soit stockées et déchargées une fois par jour sous forme d'un log revêtant la forme d'un fichier CSV, soit sont compressées et ensuite transmises régulièrement par lots à l'aide du protocole de transmission LoRaWan. Dans ce deuxième cas, le traitement sur les nœuds se fait au détriment de leur autonomie. Néanmoins, la compression des données permet de réduire drastiquement le volume des données à transmettre. Pour remédier au problème d'autonomie, la capacité des batteries a été augmentée. Au niveau du cloud, le traitement des données collectées consiste à calculer des paramètres statistiques sur des intervalles de temps d'une seconde (100 données). Les données collectées brutes doivent également être archivées.

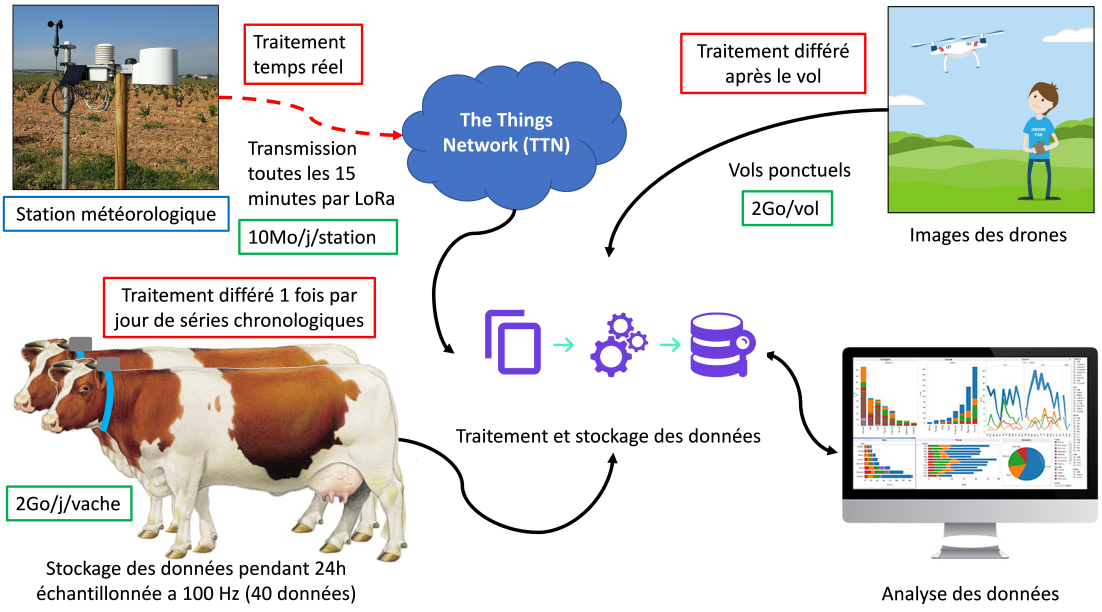


FIGURE 2.1 – Cas d'utilisation "vaches connectées"

2.4 Conclusion

L'un des principaux enjeu pour le Smart Farming est de pouvoir collecter des données issues de plusieurs milliers de capteurs arrivant à des vitesses très variables et donc impliquant périodiquement des montées en charges importantes. A cela s'ajoute la variété de formats de données et les difficultés de les faire interopérer.

Le cas d'utilisation sélectionné permet de tester l'architecture cloud dans des modes d'utilisations variés temps réel et en temps différé. Il permet de valider l'architecture sur la gestion des faibles débits de données environnementales et les grands volumes de données. Il permet également de tester l'architecture sur le traitement de données arrivant à haute fréquence, avec un traitement en quasi-temps réel (quelques secondes). Ce cas d'utilisation permet de couvrir la majorité des spécificités liées aux Smart Farming.

Dans le cadre de cette thèse de doctorat, nous allons adresser par l'entremise de ce cas d'utilisation, les problématiques liées au nombre de capteurs, leur hétérogénéité et à la qualité variable des données collectées dans différents formats accompagnés de métadonnées contextuelles.

3

État de l'art des architectures cloud en Smart Farming

Plan du chapitre

Résumé

L'Agriculture 4.0 aussi appelée Smart Agriculture ou Smart Farming est à l'origine de la production d'une énorme quantité de données qui doivent être collectées, stockées et traitées dans des délais très courts. Le traitement de cette énorme quantité de données nécessite d'utiliser des infrastructures utilisant des architectures IoT adaptées. Notre revue propose un panorama comparatif du Cloud Central, des architectures Cloud Distribuées , des stratégies collaboratives de traitement et nouvelles tendances utilisées dans le contexte de l'agriculture 4.0. Dans cette revue, nous essayons de répondre à 4 questions de recherche : (1) Quelles architectures de stockage et de traitement sont les mieux adaptées aux applications en Agriculture 4.0 et répondre à ses particularités ? (2) Les architectures génériques peuvent-elles répondre aux besoins des cas applicatifs Agriculture 4.0 ? (3) Quels sont les possibilités de développement horizontal permettant le passage de la recherche à l'industrialisation ? (4) Quelles sont les valorisations verticales c'est-à-dire les possibilités de déplacer des algorithmes entrainés dans le cloud à des produits embarqués ou autonomes ? Pour cela, nous comparons les architectures avec 8 critères (proximité utilisateur, latence et gigue (jitter), stabilité du réseau, haut débit, fiabilité, évolutivité, rentabilité, maintenabilité) et nous analysons également les avantages et les inconvénients de chacune d'elles.

Publication liée à ce chapitre

O. Debauche, S. Mahmoudi, P. Manneback, "Cloud and Distributed Architectures for Data Management in Agriculture 4.0 : Review and Future Trends". *Journal of King Saud University - Computer and Information Sciences*, 2021. doi : 10.1016/j.jksuci.2021.09.015.

3.1 Introduction

À l'ère de l'Internet des objets, le type de clients devient des appareils IoT de plus en plus légers et l'environnement réseau passe progressivement des réseaux filaires à haut débit à une communication sans fil instable. Dans le même temps, la demande des utilisateurs pour les applications IoT se déplace également vers la fourniture de services en temps réel et sensible au contexte, faisant passer progressivement le focus du cloud vers le Fog [30].

Le cloud est situé au sein d'Internet et est géographiquement centralisé, constitué de quelques nœuds serveurs contenant toutes les ressources et la distance avec les clients nécessite le passage par des relais multiples (multi hops) [31]. Le Cloud Computing (CC) est un paradigme largement répandu qui offre des avantages tels qu'un effort de gestion minimal, une facilité d'utilisation, une élasticité rapide, un paiement à l'utilisation, l'ubiquité [32], une maintenance facile, une gestion centralisée, une utilisation élevée du serveur [33]. Cependant, la centralisation des ressources implique une augmentation de la latence moyenne du réseau, une utilisation importante de la bande passante et un délai de traitement élevé. En effet, l'énorme quantité de données gérées en un point unique peut créer une congestion dans les serveurs cloud et les liens de connexion [34].

Par ailleurs, le développement rapide en parallèle des dispositifs intelligents omniprésents, du réseau omniprésent, de la popularité croissante de la réalité virtuelle et augmentée, des véhicules autonomes, des drones, des réseaux sociaux, des applications et services de réseaux ne sont pas sans conséquences. En effet, la bande passante et la vitesse du réseau limitent les performances et l'efficacité du Cloud Computing, en particulier pour les applications en temps réel et critiques. De plus, le Cloud Computing peut difficilement être adapté ou appliqué à divers types de technologies et de scénarios d'applications [35]. Pour résoudre ces problèmes, diverses extensions du Cloud Computing centralisé ont été proposées par des industriels et des universitaires pour déplacer l'informatique et le stockage à la périphérie du réseau, à proximité des utilisateurs. Le Fog Computing utilise des éléments du réseau situés entre le cloud central, la périphérie du réseau et des éléments aux extrémités de la périphérie tels que des microcontrôleurs proches des capteurs pour traiter et stocker les données de manière distribuée à proximité des nœuds.

D'autre part, avec le développement des appareils mobiles, de nouveaux paradigmes proches des utilisateurs mobiles ont été proposés. Par exemple, des cloudlets ou des micro-centres de données implantés géographiquement et accessibles au moyen de protocoles Wi-Fi, mais cette approche ne garantit pas toujours une qualité de réseau suffisante. Les fabricants d'équipements de réseau cellulaire ont proposé le paradigme Mobile Edge Computing (MEC) qui associe des serveurs Fog à des stations de base pour fournir des services aux appareils mobiles. Le MEC associé à la 5G permet de combiner un réseau à latence ultra faible avec une bande passante élevée, et des ressources de

traitement accessibles à proximité. Le concept original de MEC a ensuite été étendu aux réseaux sans fil et par conséquent renommé en "Multi-access Edge Computing" [36].

En raison des récentes avancées en matière de big data, nous présentons dans ce chapitre, une revue qui donne un aperçu de l'état de l'art concernant l'agriculture intelligente. Cette étude vise à synthétiser les paramètres qui conditionnent le choix de l'architecture pour collecter, traiter et stocker les données agricoles. Comme il existe une grande variété de cas d'utilisation, il est important de faire un choix éclairé en matière d'architecture. De cette façon, nous comblons le manque actuel de la littérature avec une revue de l'architecture cloud utilisée dans Agriculture 4.0 pour collecter, traiter et stocker des données afin d'éclairer le lecteur sur les choix possibles et les nouvelles tendances qui émergent.

3.2 Analyse de la littérature

Nous commençons notre revue en identifiant les précédents travaux réalisés dans le domaine de l'Internet des Objets appliquées à l'Agriculture Intelligente pour faire le point sur l'état de l'art et mettre en évidence des aspects qui n'ont pas complètement été explorés ou ne l'ont pas été à l'heure actuelle. Dans cette section, nous poursuivons deux objectifs. La première vise à positionner notre travail par rapport à la littérature existante. La seconde vise à identifier les architectures couramment utilisées dans le cas d'applications en Agriculture 4.0.

3.2.1 Travaux similaires

Les articles examinés présentés dans le tableau 3.1 ont été sélectionnés dans la période allant de janvier 2017 à décembre 2021. La contribution majeure de chaque article a été extraite et mise en évidence.

TABLE 3.1 – Résumé de la précédente revue réalisée sur la gestion du big data dans un contexte de Smart Farming

Contribution majeure	Réf.	Année
Solutions agro-industrielles & environnementales de suivi, le contrôle, la logistique et la prévision.	[37]	2017
Diagnostic et analyse des protocoles de communication utilisés dans les déploiements IoT.	[38]	2017
Enquête sur les technologies IoT dans l'agriculture et a mis en évidence les défis à venir.	[28]	2017
Identification des défis de l'IoT, son application dans l'agriculture intelligente, et présentation des tendances et de l'innovation technologique.	[39]	2018
Examen des applications IoT dans l'agriculture de précision, évaluation des contributions antérieures des chercheurs et voies vers l'innovation future.	[40]	2019
Bilan du déploiement de l'IoT dans l'agriculture protégée, identification de ses enjeux et prospection de nouveaux domaines de recherche.	[41]	2019
Examen des solutions existantes d'agriculture de précision basées sur l'IoT pour de nouveaux projets.	[42]	2019
Revue, comparaison, prospection et défis des applications des technologies de communication sans fil en agriculture de précision.	[43]	2019
Examen, étude de cas et défis du WSN dans le comportement environnemental.	[44]	2019
Examen, identification, défis des tendances actuelles et futures de l'IoT en agriculture.	[45]	2019
Enquête sur l'agriculture basée sur l'IoT, présentation de la connexion entre l'IoT, le big data et le Cloud Computing, la réglementation et les politiques de l'IoT, et son application en agriculture.	[46]	2019
Enquête sur l'utilisation des drones, aperçu de l'AP et enquête sur 20 applications de drones.	[47]	2020
Défis de l'architecture agricole basée sur l'IoT, résumé des enquêtes existantes sur l'agriculture intelligente. et classification des modèles de menaces, étude, analyse des défis et futurs travaux de sécurité et de confidentialité de l'agriculture verte basée sur l'IoT.	[48]	2020
Discute du rôle de l'IoT et de l'analyse des méga données dans l'agriculture en mettant l'accent sur le statut commercial des applications et les résultats de la recherche translationnelle.	[49]	2020
Recense différentes solutions pour aborder l'IoT dans les défis de l'agriculture arable.	[50]	2020
Revue systématique présentant l'utilisation de l'IoT avec l'agriculture intelligente.	[51]	
Revue méthodologique et analyse des composants IoT et de leurs applications en agriculture 4.0.	[22]	2021
Examen des technologies émergentes pour les agriculteurs et l'agriculture 4.0.	[52]	2020
Examen, classification, présentation, comparaison et défis des technologies émergentes pour l'agriculture basée sur l'IoT.	[53]	2021

Dans les paragraphes suivants, nous effectuons une synthèse des travaux scientifiques publiés au cours des quatre dernières années (2017-2021). En 2017, Ray [38] a passé en revue les applications IoT et les défis rencontrés lors du déploiement de l'IoT dans différents domaines dont l'agriculture. Talavera. *et al.* [37] ont examiné les applications agro-industrielles et environnementales qui utilisent l'Internet des objets (IoT) pour la surveillance, le contrôle, la logistique et la prévision. Tzounis *et al.* ont mené une enquête sur les technologies IoT dans l'agriculture et les défis auxquels les agriculteurs seront confrontés [28]. Élie *et al.* ont identifié les défis les plus rencontrés dans le domaine des applications IoT dans l'agriculture intelligente et ont présenté des tendances communes pour des idées innovantes [39]. En 2019, Ayaz *et al.* ont fourni un état de l'art des architectures basées sur l'IoT appliquées à l'agriculture et identifié les tendances présentes et futures dans le même domaine d'étude [45]. Farooq *et al.* ont présenté les ingrédients de l'agriculture intelligente basée sur l'IoT avec des technologies utilisées qui appliquent l'utilisation de l'architecture et des protocoles de réseau ; de plus, ils ont fourni un aperçu des réglementations et des politiques d'utilisation de l'IoT dans l'agriculture en matière de sécurité et de confidentialité. Ils ont conclu leur étude en résumant les principaux défis rencontrés dans cette discipline [46]. Feng *et al.* ont donné un aperçu des technologies de communication sans fil dans le domaine de l'agriculture de précision. Ils ont comparé la progression et les défis des technologies existantes avec le temps de communication régulier utilisé [43]. Shafi *et al.* ont mené une revue de la littérature sur l'automatisation de l'agriculture basée sur l'IoT avec le réseau de capteurs sans fil (WSN). Ces auteurs ont présenté une étude de cas basée sur deux modèles : 1- un WSN pour surveiller en temps réel la récolte des conditions de santé, 2- une imagerie de télédétection basée sur le système pour la classification entre les rendements sains et malsains [44]. En termes de protection de l'agriculture, Shi *et al.* ont réalisé une revue systématique de la littérature publiée au cours de la dernière décennie pour relever les défis et les travaux futurs pour faire avancer la recherche dans le domaine de l'agriculture protégée c'est-à-dire la production de légumes à haute valeur ajoutée et autres plantes horticoles dans des milieux contrôlés (serre, phytotrons) [41]. Khanna et Kaur ont fait appel à un scénario évolutif pour mettre en évidence l'impact le plus important de l'IoT dans l'agriculture de précision (Precision Agriculture (PA)). Ils ont évalué la contribution de leurs prédécesseurs et amélioré les défis pour ouvrir une nouvelle direction d'inspiration et d'innovation dans l'IoT appliqué à l'agriculture de précision [40]. Ruan *et al.* ont analysé les travaux de la littérature de 2009 à 2018 et suggéré de nouvelles pistes de recherche à explorer dans le domaine de l'IoT agricole, des infrastructures, de la sécurité des données et du partage des données [42]. En 2020, deux études ont été menées pour une vingtaine d'applications de drones consacrées soit à des processus de surveillance aérienne des cultures, soit à des tâches de pulvérisation [47] et aux dilemmes que les chercheurs doivent surmonter lors du déploiement de l'IoT dans le domaine de l'agriculture verte [48]. Villa-Henriksen *et al.* ont identifié différents défis rencontrés lors de la mise en œuvre de l'IoT dans diverses applications et ont proposé différentes solutions pour les relever [50]. Misra *et al.* ont discuté le role

de l'IoT et du big data en Smart Farming [49]. En 2021, une étude plus récente menée par Friha *et al.* formule des hypothèses sur l'utilisation, l'application, la classification et la comparaison des technologies émergentes les plus développées telles que l'Internet des objets (IoT), les véhicules aériens sans pilote (UAV), les technologies sans fil, les plates-formes IoT open source, les réseaux définis par logiciel (SDN), la fonction réseau de technologies de virtualisation (NFV), Cloud/Fog Computing et plateformes middlewares [53].

3.2.2 Plateformes mises en œuvre dans les cas d'utilisation

Nous avons regroupé les applications en 4 catégories : (1) **Gestion de l'eau** dans laquelle nous avons agrégé tous les types d'utilisation de l'eau tels que l'irrigation et l'abreuvement des animaux. (2) **Maladies et Ravageurs Végétaux** regroupe tous les cas d'usages dans la détection des pathologies végétales et le traitement des pathologies végétales (pulvérisation de fongicides, pesticides, etc.). (3) **Crop Management** regroupe tous les cas d'usage relatifs aux opérations culturales : gestion des sols (labour, apport d'engrais), semis, désherbage et récolte. (4) **Elevage** comprend tout ce qui concerne l'élevage des animaux de ferme (nutrition, comportement, maladies, traitements). Le tableau 3.2 résume les plateformes utilisées pour mettre en œuvre ces cas d'utilisation dans l'agriculture intelligente classés selon nos quatre catégories.

Garcia *et al.* donnent un aperçu des tendances dans le domaine de l'irrigation intelligente. Ils ont montré que les données sont stockées dans une base de données ou dans le cloud. Sur 151 articles examinés, un utilise le Raspberry Pi, 18 mentionnent l'utilisation de bases de données, 53 utilisent divers clouds et 79 des architectures originales ou non mentionnées [26]. Navarro *et al.* ont identifié 21 plateformes utilisées dans 50 cas d'utilisation différents classés en 5 catégories : Intelligence artificielle, Big Data, Machine Learning, Computer Vision et Autre/Non identifié [51]. Jayaraman *et al.* présentent SmartFarmNet, une plateforme IoT offrant une intégration sans effort de capteurs, prenant en charge une analyse de données évolutive et proposant des outils facile d'utilisation pour analyser et visualiser les données [25]. Codeluppi *et al.* décrivent LoRaFarM une architecture générale modulée en fonction des caractéristiques et des exigences de la ferme [54].

TABLE 3.2 – Résumé des plateformes cloud, bases de données mentionnées dans les revues Smart Farming

	Gestion de l'eau	Maladie des plantes & Ravageurs	Gestion des cultures	Bétail	Réf.	Année
IoT platform						
Thingspeak	x				[55]	2011
FIWARE	x				[56]	2018
NETPIE	x		x		[57]	2020
Ubidots	x				[58]	2021
SmartFarmNET	x		x		[25]	2016
Thinger.io	x		x		[59]	2019
Kaa IoT Platform	x			x	[60]	2021
IBM Watson IoT Platform	x	x	x	x	[61]	2015
Microsoft Azure IoT Platform	x		x	x	[62]	2021
AT&T M2X Cloud			x		[63]	2021
Blynk			x		[64]	2021
MACQU			x		[65]	2002
ERMES			x		[66]	2017
Agrocloud		x	x	x	[67]	
CropInfra			x		[68]	2014
SensorCloud			x		[69]	2020
LoRaFarM	x		x	x	[54]	2020

TABLE 3.3 – Résumé des plateformes cloud, bases de données mentionnées dans les revues Smart Farming

	Gestion de l'eau	Maladie des plantes & Ravageurs	Gestion des cultures	Bétail	Réf.	Année
Cloud platform						
Amazon Web Service	x	x	x	x	[70]	2021
IBM Cloud	x	x	x	x	[71]	2021
Microsoft Azure	x	x	x	x	[72]	2021
Integra	x		x		[73]	2021
Base de données dans le cloud						
DynamoDB	x		x	x	[74]	2021
MongoDB Atlas	x		x	x	[75]	2021
Firebase	x		x	x	[76]	2021
InfluxDB Cloud	x		x	x	[77]	2021
Base de données locale						
MySQL	x				[78]	2021
SQLite	x				[79]	2021
PostgreSQL / PostGIS	x	x			[80]	2021
Apache Cassandara				x	[81]	2021
Apache Druid			x	x	[82]	2021

3.2.3 Analyse de la littérature

L'analyse des revues littéraires existantes sur l'agriculture intelligente montre que les applications utilisent soient une architecture cloud open source ou commerciale, une architecture spécifique répondant à leurs objectifs est développée ou ne décrivent pas leur système de stockage et de traitement. Ces dernières représentent plus de la moitié des articles et font que certaines architectures de traitement restent inconnues car elles n'ont jamais été spécifiquement décrites et étudiées. De plus, le fait que des architectures soient développées davantage peut être dû au fait que les plates-formes commerciales ne répondent pas pleinement aux besoins de l'Agriculture 4.0. Cela nous amène à nos questions de recherche et à leurs motivations respectives :

1. *Quelles architectures de stockage et de traitement sont les mieux adaptées aux applications Agriculture 4.0 et répondent à ses particularités ?* D'une part les architectures génériques dédiées ou non à l'IoT sont capables d'adresser un grand nombre de cas d'usage mais pas spécifiquement les besoins d'Agriculture 4.0 et d'autre part, les chercheurs développent des architectures pour répondre à des problèmes spécifiques ou aux besoins de cas d'utilisation spécifiques. La sélection d'une architecture adaptée est cruciale pour la bonne mise en œuvre des cas d'utilisation identifiés.
2. *Les architectures génériques peuvent-elles répondre aux besoins des cas d'application Agriculture 4.0 ?* L'agriculture 4.0 a des exigences spécifiques décrites dans la section d'introduction qui ne peuvent pas toutes être traitées par une seule architecture générique classique. Une comparaison entre les avantages et les inconvénients des grandes architectures génériques dans le contexte de l'agriculture 4.0 est importante pour mettre en évidence le choix lors de l'étape de conceptualisation.
3. *Quelles sont les possibilités de valorisation horizontale qui permettent le passage de la recherche à l'industrialisation ?* Motivation : L'utilisation de solution architecturale qui peut être par exemple gratuite pendant la phase de recherche mais nécessite une réimplémentation causée par des limitations de licence, le coût de licence dans le budget des cas d'utilisation, etc. L'utilisation de produits dans des écosystèmes fermés ou semi-fermés sont des freins à la valorisation de la recherche.
4. *Quelles sont les possibilités de valorisation verticale pour passer d'algorithmes entraînés dans le cloud à des produits embarqués ou autonomes ?* Motivation : La collecte massive de données dans le cloud permet de développer des algorithmes complexes qui nécessitent une grande quantité de ressources de calcul à élaborer. Par la suite, ils peuvent être compressés, réduits, optimisés afin d'être déployés dans des appareils embarqués ou divisés et établir une collaboration entre les appareils et les ressources informatiques telles que le Cloud, le Fog, etc.

Afin de répondre à ces questions, une revue de la littérature permettra de synthétiser les différentes approches actuellement utilisées, d'identifier de nouvelles tendances et d'envisager de nouvelles pistes de recherche à explorer.

3.2.4 Critères de comparaison des architectures

Afin de comparer les architectures sélectionnées nous avons choisi de sélectionner les 8 critères suivants :

1. **Proximité avec l'utilisateur final** exprime la nécessité d'être proche de l'utilisateur. Ce critère est important pour les applications où la confidentialité et le temps de réponse aux requêtes sont critiques. Nous attribuons une valeur d'un * lorsque la confidentialité n'est pas cruciale ; ** lorsque la proximité avec l'utilisateur est souhaitable mais pas cruciale pour le développement du cas d'utilisation ; *** lorsque la proximité de l'utilisateur est la pierre angulaire de l'application. Le critère
2. **Latence & Gigue (Jitter)** décrit l'importance pour l'architecture d'avoir une latence et une gigue minimales. Ce critère est particulièrement important pour les cas d'utilisation où le temps de réponse à la requête en quasi (temps réel) est requis et/ou le temps entre la production des données et l'ingestion par l'architecture de traitement et de stockage est essentiel.
3. **Stabilité du réseau** traduit la nécessité d'avoir un réseau stable ou si son interruption peut être tolérée par exemple lors des déplacements de certains objets. Nous utilisons une valeur de * si le cas d'utilisation mis en œuvre peut tolérer l'absence de réseau pendant quelques heures ; ** si quelques minutes d'interruption sont tolérables ; *** La stabilité du réseau est un élément essentiel du cas d'utilisation.
4. **Haut débit** exprime la capacité de l'architecture à traiter rapidement une grande quantité de données arrivant à haute fréquence ; Utilisez une valeur de * si les données arrivent principalement à intervalles réguliers ; une valeur de ** si les données arrivent en rafales, et *** si les données arrivent en continu à haute fréquence ($>$10 Hz).
5. La **Fiabilité** est un critère qui identifie si l'infrastructure est critique en d'autres termes si une interruption de l'infrastructure pourrait causer des pertes de vie ou non. Nous attribuons un poids de * si les données ne sont pas critiques et que les dommages potentiels causés par une interruption de l'architecture sont mineurs ou nuls ; ** si potentiellement dommageables mais tolérables s'ils surviennent plus d'une fois par an ; *** si l'application ne peut tolérer aucune interruption qui causerait des dommages irréversibles ou des pertes en vies humaines.
6. **La mise à l'échelle** est un critère qui mesure la régularité de l'évolution en termes de traitement et de stockage sur une période d'un an. Si l'évolutivité doit être atteinte au maximum une fois par an. Nous utilisons un poids de * ; si

l'évolutivité est atteinte au maximum deux fois par an utiliser **; si l'évolutivité doit être réalisée plus de deux fois par an, on utilise un poids de ***.

7. Le critère **Coût-Efficacité** reflète la nécessité de maîtriser les coûts d'infrastructure. Ce critère est d'autant plus important que l'infrastructure est amenée à évoluer à la fois en termes d'échelle et de complexité. Utilisez un poids de * si le projet conservera une taille relativement constante et n'a pas besoin d'être mis à l'échelle ou modifié de façon spectaculaire ; Utiliser **, si le projet évolue raisonnablement, c'est-à-dire qu'il ne doit pas subir de modification significative plus d'une fois par an. Nous utilisons un poids de *** si la taille du projet et/ou sa complexité nécessite une étude fine des coûts.
8. Le critère **Maintenabilité** est directement lié à la pérennité du projet. Si la pérennité du projet ne dépasse pas deux ans, nous assignons un poids de * ; si la durée de vie du projet est comprise entre 2 et 5 ans, nous affectons un poids de ** au-delà de 5 ans, nous attribuons ***.

3.3 Architectures

Les nombreuses publications traitant des architectures cloud liées à Agriculture 4.0, résumées dans le tableau 3.2, montrent que des efforts importants ont été consacrés à la résolution de toute une série de problèmes liés à de multiple cas d'utilisation. En effet, une architecture universelle et unique qui assure tous les besoins de tous les cas d'utilisation, n'existe pas pour les applications IoT dans l'agriculture intelligente. C'est la raison pour laquelle plusieurs chercheurs ont proposé différentes architectures qui répondent à des problématiques spécifiques des architectures génériques.

La figure 3.1 présente un aperçu global de l'organisation de l'Agriculture 4.0.

Nous présentons dans la suite de cette section les différentes architectures cloud proposées dans la littérature.

3.3.1 Architecture Cloud centrale

Les architectures cloud centrales reposent sur deux architectures de base qui s'associent ou se combinent pour former des architectures modernes. Ces deux architectures sont :

Batch Architecture vise à traiter un ensemble de données entier en mode hors ligne. Pour ce type d'architectures, tant que le traitement du jeu de données n'est pas terminé, il se poursuit et ne produit des résultats que lorsqu'il est arrivé à son terme. Généralement, les données sont sélectionnées et distribuées à différents nœuds afin d'être traitées plus rapidement. Lorsque tous les traitements sont réalisés sur tous les nœuds, les

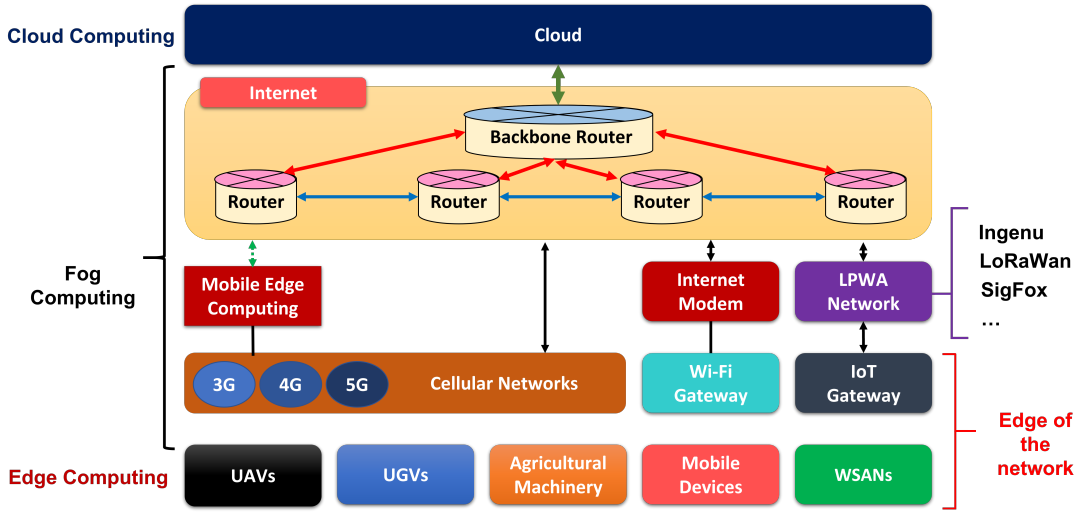


FIGURE 3.1 – Structure globale de l'IoT dans l'agriculture 4.0

résultats sont triés et agrégés pour obtenir une sortie globale. Cette architecture est facile à mettre en œuvre, et l'agrégation se fait par un framework, mais les temps de traitement peuvent être longs, et les données extraites au cours du traitement ne peuvent être traitées avant la fin du traitement en cours. De plus, il est possible d'incrémenter les résultats du lot précédent et de produire un résultat intégrant les données traitées en cours.

TABLE 3.4 – Avantages et inconvénients de l'architecture par lots

Avantages	Inconvénients
- Facile à mettre en œuvre et à entretenir.	- Ne traiter que les données précédemment stockées sous une autre forme (fichier, base de données, etc).
- Capable de réaliser des traitements de longue durée (plusieurs heures ou jours).	- Le traitement ne peut être modifié avant la fin.
- Retraitement des anciennes données facile à réaliser.	- Résultats disponibles uniquement à la fin du traitement.

Sallah *et al.* ont utilisé une architecture par lots pour mettre à jour les données dans le modèle AquaCrop (FAO) intégré dans l'environnement R afin de faciliter l'étalonnage et la validation du modèle, et exécuter et évaluer tous les champs en une seule exécution [83]. Nolack Foté *et al.* ont présenté une architecture pour extraire des connaissances sur le long terme à partir de données en élevage de précision (PLF) [84].

Architecture Temps Réel également nommée Architecture Streaming traite les données au fur et à mesure de leur arrivée et les résultats sont progressivement disponibles par opposition à l'architecture Batch où il n'est pas nécessaire d'attendre la fin de l'ingestion de toutes les données d'entrée pour obtenir un résultat. La notion de temps réel est fortement dépendante du contexte d'analyse avec des temps de traitement de quelques millisecondes à quelques minutes. L'architecture temps réel peut être mise en œuvre de deux manières différentes. D'une part avec le Micro-Batch dans lequel une infime quantité de données est traitée toutes les n secondes et un résultat est obtenu à la fin du traitement ou d'autre part avec une approche en streaming dans laquelle chaque nouvelle donnée est immédiatement traitées et une sortie est rapidement produite. Cette architecture est limitée au traitement des flux de données [85].

TABLE 3.5 – Avantages et inconvénients de l'architecture en temps réel

Avantages	Inconvénients
- Permettre un traitement rapide des données nouvellement arrivées.	- Impossible d'obtenir un traitement sur des lots de grande taille.
- Le traitement par lots peut être émulé à l'aide de micro-lots, mais tous les algorithmes ne peuvent pas être implémentés.	- Retraitement des anciennes données difficile à mettre en œuvre.
- Facile à mettre en œuvre et à entretenir.	- La nécessité d'un traitement en temps réel ne permet pas toujours de réaliser un calcul exact qui serait trop long. On a alors recours à un estimateur pour estimer la valeur.

Diverses données sont produites par différents capteurs en champs ou sur les animaux, les véhicules et les robots en Agriculture 4.0. Ces données doivent ensuite être soient stockées à l'état brut et traitées de manière hors ligne où des traitements longs et complexes peuvent être réalisés. Soient, les données peuvent être traitées avant leur stockage avec un traitement par lots (Batch) ou un traitement en flux (streaming) ou une combinaison de ceux-ci. La durée de stockage est extrêmement variable suivant la nature des données et leur perte de valeur dans le temps. Le **Traitement hors ligne** est classiquement utilisé pour traiter des images provenant de drones, d'UGV ou de satellites, par exemple, pour déterminer l'activité de photosynthèse, évaluer le développement de la canopée ou les stocks d'espèces palatables disponibles dans un pâturage, etc. Tandis que **Traitement en streaming** permet de détecter des anomalies dans les comportements des animaux en temps réel, ou lors d'opérations agricoles telles que la récolte, la détection des maladies et ravageurs, l'élimination des mauvaises herbes. Dans ce dernier cas, les données ne sont pas stockées car elles perdent rapidement toute valeur après leur ingestion. Enfin, une combinaison des deux méthodes précédentes, c'est-à-dire **Traitement hors ligne et en streaming** est utilisée pour estimer les

métriques en temps réel et réaliser des traitements complexes de manière hors ligne en même temps. Cette approche est utilisée par les robots de traite qui détectent les anomalies dans la production en temps réel tandis que le traitement hors ligne estime la production future de chaque vache sur la base de la traite précédente [22].

Les architectures Lambda sont utilisées dans les systèmes qui doivent traiter et exposer rapidement des quantités massives de données en streaming. Cette architecture cloud a été proposée par Nathan Marz et James Warren [86] pour gérer d'énormes quantités de données et résoudre des problèmes complexes combinant le traitement de gros volumes de données (Batch) tout en incorporant les données les plus récentes traitées par des processus en temps réel [87]. Cette architecture est générique, évolutive et tolérante aux pannes contre les pannes matérielles et les erreurs humaines. L'architecture est composée de trois couches : (1) la couche de traitement par lots (Batch Layer) transforme de très grandes quantités de données par Batch ; (2) la couche de traitement des flux (Speed Layer) qui traite les données en temps réel et fournit des vues basées sur les données les plus récentes et (3) couche de service répondant aux requêtes. Les données proviennent soit d'une source de données, soit d'une file d'attente de messages.

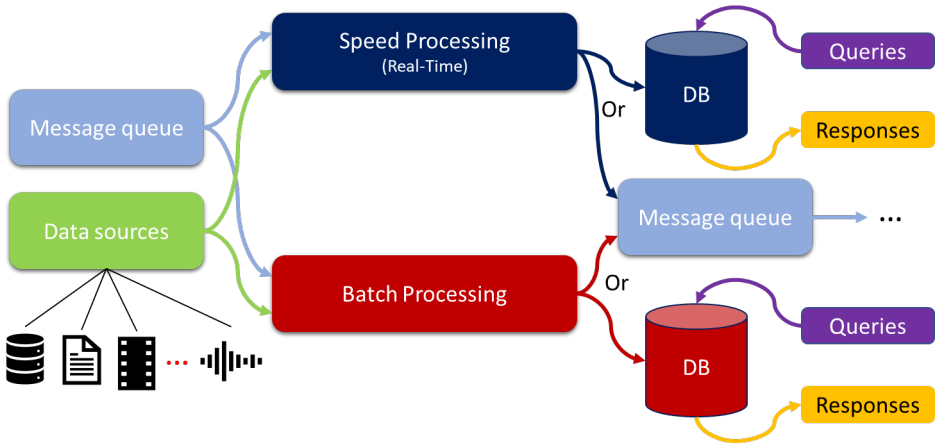


FIGURE 3.2 – Schéma général de l'architecture lambda

Ce paradigme permet d'exécuter des requêtes arbitraires sur n'importe quelle donnée en temps réel et est particulièrement adapté aux infrastructures critiques et aux systèmes de santé [88]. Plusieurs implémentations de l'architecture Lambda dans la gestion de l'environnement intelligent, le stockage de données volumineuses et l'analyse peuvent être trouvées dans [89]. Parmi les critiques qui ont été formulées à l'encontre de l'architecture Lambda, il y a la nécessité de faire deux fois plus de développements pour la branche temps réel et la branche Batch. Il est possible d'effectuer un traitement par lots et en temps réel avec un traitement de flux, c'est ce que fait l'architecture Kappa décrite ci-dessous [90].

TABLE 3.6 – Avantages et inconvénients de l'architecture Lambda

Avantages	Inconvénients
- Traiter les données en temps réel ou en traitement par lots de manière séparée.	- La fiabilité de deux modes de traitement est plus coûteuse que les autres architectures si les deux exécutent le même traitement.

Parmi les cas d'utilisation en agriculture 4.0 utilisant une architecture Lambda, nous aimerions souligner les travaux Roukh *et al.* qui ont proposé WALLeSMART, une plateforme cloud basée sur une architecture Lambda et spécifiquement développée pour le Smart Farming. Cette plateforme implémente Apache Kafka pour stocker les données temporaires avant leur traitement. Apache Hadoop et le modèle de programmation Mapreduce sont utilisés pour le traitement par lots tandis qu'Apache Storm traite les données en temps réel. L'originalité de cette architecture est le couplage d'une base de données NoSQL Apache Cassandra et d'une base de données SQL, PostgreSQL où les données sont stockées en fonction de leur nature. Le langage de requête GraphQL permet d'interroger des bases de données [91, 92]. Debauche *et al.* ont décrit une architecture lambda pour le phénotypage numérique [93] et les comportements des animaux de la ferme couplée à une architecture d'hébergement d'applications basée sur Apache Mesos et la conteneurisation Docker pour faciliter le déploiement de diverses applications. Une API interconnecte et contrôle les accès entre l'architecture Lambda et l'architecture de l'application d'hébergement. L'architecture Lambda est basée sur Apache Beam pour changer facilement de brique logicielle d'exécution en fonction de l'évolution de la technologie et ainsi améliorer sa pérennité. Apache Druid est utilisé pour stocker des données de séries temporelles [94] et des métadonnées de données stockées dans le Datalake basé sur Apache Hadoop [95]. Une variante de cette architecture, nommée architecture Unified Lambda combine des pipelines Batch et Stream qui s'exécutent simultanément, puis les résultats sont fusionnés automatiquement [96].

L'**architecture Kappa**, proposée par Jay Kreps de LinkedIn [90], simplifie l'architecture Lambda en combinant des couches temps réel et Batch. Cette architecture cloud diffère de l'architecture Lambda en utilisant un système de stockage non permanent des données dans un fichier journal non modifiable tel qu'un système comme Apache Spark ou Apache Kafka et par conséquent n'autorise qu'un stockage pendant une durée limitée afin de permettre un éventuel retraitement de données. Les couches Batch et Speed sont également remplacées par un moteur de traitement de flux. Ainsi, l'architecture Kappa est composée de deux couches : les couches de streaming et de service et peut être implémentée avec une messagerie de publication-abonnement (publish / subscribe) comme Apache Kafka qui permet d'ingérer des données.

Le principal avantage de cette architecture est sa simplicité. Cela évite d'avoir à maintenir deux bases de code distinctes pour les couches Batch et Speed. Lorsque

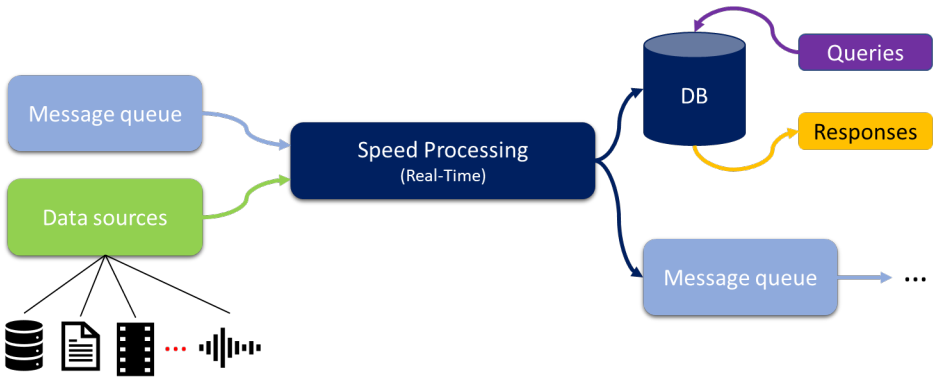


FIGURE 3.3 – Schéma général de l'architecture Kappa

le traitement des données en temps réel et historiques est le même, une architecture Kappa doit être utilisée. La Fast Data Architecture est une variante de l'architecture Kappa dans laquelle les données ne sont plus lues à partir de fichiers mais à partir d'une brique logicielle placée en amont comme Kafka qui capture plusieurs flux et les combine avant qu'ils ne soient traités par la couche de vitesse [97]. Persico *et al.* ont réalisé un benchmark des architectures Lambda et Kappa et montrent que l'architecture Lambda surpasse Kappa pour le traitement des données des réseaux sociaux (YFCC100M) [98].

TABLE 3.7 – Avantages et inconvénients de l'architecture Kappa

Avantages	Inconvénients
- Très efficace pour le traitement en temps réel grâce au traitement en mémoire.	- Le traitement par lots est émulé grâce au découpage des données Micro Batch traité en temps réel par la couche streaming.
- Coût optimisé car permet un traitement en temps réel et par lots.	- Impossible de traiter des lots de grande taille. - Doit être affiné à partir des données pour obtenir les meilleures performances [99].

D'autres architectures dérivées ou inspirées des architecture précédentes (traitements par lots, traitements temps réel, Lambda et Kappa) ont été développées pour résoudre des problèmes spécifiques tels que (1) **SMACK** [100] qui tente de proposer une architecture optimale avec des composants fixes ; (2) **Liquid** [101] est une architecture qui fournit une faible latence, un traitement incrémentiel, une haute disponibilité avec des ressources isolées, et capable de stocker des données à haut débit avec une architecture à faible coût opérationnel ; (3) **Butterfly** [97] propose d'unifier les couches Batch,

Speed et servant dans une plateforme unique dans laquelle les données sont organisées comme une collection de trois types d'abstractions ; (4) **Zeta** [102] qui intègre une architecture lambda avec les données commerciales de l'entreprise ; (5) **BRAID** [103] est une architecture de traitement hybride où toutes les données à venir et le fichier de configuration du traitement et éventuellement les résultats de traitement réécrits sont stockés dans un espace partagé ; (6) **IoT-a** [104] est composé de trois blocs : des requêtes ad-hoc, une base de données et un système de fichiers distribué ; (7) **Polystore** [105] implémente un système de bases de données multiples PostgreSQL, SciDB et Accumulo car une base de données seule ne peut pas stocker efficacement tous les types de données.

TABLE 3.8 – Évaluation qualitative de l'architecture centrée sur le cloud

Critère	Lot	Flux	Lambda	Kappa
Proximité utilisateur	*	*	*	*
Latence & Gigue	*	*	*	*
Stabilité du réseau	*	*	*	*
Haut débit	***	***	***	***
Fiabilité	***	***	***	***
Évolutivité	***	***	***	***
Rentabilité	***	***	*	**
Maintenabilité	***	**	*	**

L'analyse de la littérature montre que deux architectures génériques majeures : Kappa et Lambda permettent d'adresser différents cas d'utilisation et sont largement implémentées et éprouvées dans d'autres domaines de l'Internet des Objets. L'architecture Lambda est plus coûteuse à mettre en œuvre que la Kappa en raison de la nécessité de maintenir deux branches de traitement parallèles et distinctes pour le traitement des flux et le traitement par lots. Il est intéressant que des traitements différents soient effectués sur les deux branches de traitement. Sinon, une architecture Kappa avec une seule branche de traitement qui traite à la fois les flux et les données par lots est plus appropriée dans la plupart des cas car elle est moins chère et plus facile à maintenir avec un seul code, elle effectue les deux types de traitement (flux et Batch). En examinant nos deux premières questions de recherche, nous observons que les architectures cloud Lambda et Kappa sont efficaces mais que ces architectures fonctionnant seulement dans le cloud et ne peuvent pas répondre, par exemple, aux cas d'utilisation où des latences très faibles sont requises. Elles devront être hybridés et complétés pour traiter ces cas particuliers. Deux possibilités s'offrent à nous. La première voie consiste à associer plusieurs plateformes cloud spécialisées pour permettre d'obtenir une plus grande généricité ou du moins de mieux couvrir un domaine. La seconde consiste à compléter les architectures cloud que nous venons d'évoquer par d'autres éléments architecturaux afin de mieux répondre aux besoins spécifiques de l'Agriculture 4.0.

3.3.2 Extension du paradigme du cloud

Avec l'augmentation de la quantité de données produites par la myriade de capteurs implantés dans les champs, les giga de données produites par les drones et les robots ainsi que les véhicules connectés ma manière de stocker et traiter ces données a dû évoluer. En effet, la quantité de données à traiter, à transférer par réseau et à traiter dans le Cloud Computing ont remis en cause l'architecture de stockage et de traitement des données. Pour résoudre le problème, deux voies ont été proposées, la première est le Multi Cloud Computing, dont l'objectif est d'assurer la redondance pour améliorer la latence. Le second est le Cloud fédéré avec pour objectif de mutualiser les ressources pour une meilleure utilisation.

Multi Cloud Computing (MCC) [4] est une extension du paradigme du Cloud Computing où les services sont distribués sur plusieurs clouds. Dans cette architecture, le workflow est entièrement distribué dans le cloud, la redondance des données est également vérifiée. L'un des avantages du MCC est le taux de récupération élevé, mais il présente les mêmes inconvénients que le Cloud Computing c'est-à-dire une latence importante qui peut fluctuer, un risque de congestion, ainsi que des problèmes de complexité et de portabilité.

Kazim *et al.* ont proposé un cadre pour fournir des services IoT et établir une coopération entre plusieurs clouds. Une authentification permet aux cloud de s'authentifier les uns avec les autres cloud dynamiquement. Alors qu'un service sélectionne le meilleur service IoT correspondant aux besoins des utilisateurs parmi plusieurs clouds et en tenant compte des paramètres Service-Level Agreement (SLA) convenus entre l'utilisateur et le fournisseur. Le SLA est généralement convenu sur le débit minimum, la latence et un délai de rétablissement du service après incident [106].

Federated Cloud (FC) agrège les ressources de plusieurs fournisseurs de cloud pour améliorer la liberté de l'utilisateur et permet aux utilisateurs de choisir où il souhaite déployer ses applications. Le cloud fédéré peut être défini comme une collaboration volontaire entre des fournisseurs de cloud hétérogènes collaborant pour partager leurs propres ressources inutilisées. L'utilisation d'une fédération de cloud permet de garantir les performances du service lors des chargements avec des ressources empruntées à d'autres clouds. De plus, la dispersion géographique des installations permet de migrer vers une autre installation et de garantir le service en cas de panne. Une interface unifiée permet une consultation aisée des services proposés. Enfin, grâce à la répartition dynamique de la charge, il est possible d'effectuer le traitement au plus près de l'utilisateur et par conséquent d'améliorer la Qualité de Service [107]. Les fédérations de cloud les plus connues sont : European Federated Cloud [108], Massachusetts Open Cloud, Mosaic [109], IEEE P2302 et Open stack Keystone.

Drakos *et al.* ont décrit agINFA, une infrastructure de données de recherche commune pour l'agriculture, l'alimentation et l'environnement utilisant EGI Federated Cloud [1] pour fédérer les ressources cloud académiques privées et virtualiser les ressources mise en commun. Cette infrastructure permet aux universités de s'associer pour partager des composants d'infrastructure de recherche, des API, un registre de services d'information Web et un ensemble de données sur la thématique de l'agriculture [110].

3.3.3 Architectures distribuées

Les approches dites "post-cloud" ont suivi le paradigme du Cloud Computing qui visait à uniquement réaliser le traitement au niveau du cloud central. Ces nouvelles approches permettent d'améliorer la latence et la gigue pour les entités immobiles mais n'apportent pas de réponse adaptée aux appareils mobiles et à la connaissance du milieu locale (context awareness). La grande quantité de données générées à la périphérie du réseau a nécessité d'augmenter la vitesse de transport des données et est finalement devenu un goulot d'étranglement pour les paradigmes informatiques basés sur le cloud [33]. De plus, le traitement des données dans le cloud n'offre aucune garantie sur la confidentialité, sur le temps de réponse et l'actionnement en temps réel car le grand nombre d'appareils augmente la latence et la fluctuation de celle-ci (gigue). De plus, la mobilité des appareils et les contraintes de puissance rendent difficile la communication avec le cloud en permanence [111, 35]. L'objectif des architectures distribuées est de rapprocher le stockage et le traitement, le filtrage et l'analyse des données au plus près des objets qui les produisent, pour limiter la consommation de bande passante et soulager le cloud. Trois paradigmes majeurs sont proposés dans la littérature (Edge et Fog Computing et Mobile Edge Computing) pour résoudre ces problèmes et apporter des capacités de type Cloud Computing à la périphérie du réseau. Toutes ces infrastructures gèrent des mécanismes de migration de Machines Virtuelles (VM) ou de conteneurs et ajustent, si besoin, le provisionnement des capacités là où se trouvent les utilisateurs. De plus, les trois paradigmes permettent la création d'infrastructures fédérées dans lesquelles peuvent coexister plusieurs infrastructures privées qui peuvent échanger des informations et des services [112].

3.3.4 Eléments constitutifs des architectures distribuées

Afin de toujours rapprocher les capacités de calcul, des traitements intermédiaires peuvent être appliqués entre les objets connectés et le cloud au niveau du réseau (Fog Computing) et au niveau des fournisseurs de téléphonie (Mobile Edge Computing). Dans la suite de cette section nous présentons successivement le Fog Computing, les cloudlets, les micro-centres de données (micro datacenters) et le Mobile Edge Compu-

1. https ://www.egi.eu/federation/egi-federated-cloud/

ting.

Le **Fog Computing** est un concept créé par Cisco Systems et est une extension du paradigme du Cloud Computing [31] dans lequel les services de calcul, de stockage et de réseau sont fournis entre les terminaux et le cloud / classifie et analyse les données IoT brutes au niveau du réseau proche du Edge (gateways) et Edge (microcontrolleurs) [3]. Les nœuds Fog sont soit des composants physiques tels que des passerelles, des commutateurs, des routeurs, des serveurs, etc., soit des composants virtuels tels que des commutateurs virtualisés, des machines virtuelles, des cloudlets, etc. déployés suivant un mode privé, communautaire, public ou hybride. Les nœuds privés sont réservés à une seule organisation, les nœuds communautaires sont utilisés par une communauté, les nœuds publics sont dédiés au grand public et les hybrides mélangent les trois modalités précédentes [113]. Ce paradigme permet de réduire les transferts de données sur le cloud, de réduire la latence [114] et la fluctuation de la latence (jitter) grâce à une architecture à trois niveaux [112]. Dans cette architecture hiérarchique, l'analyse des informations locales est réalisée au niveau inférieur et la coordination et l'analyse globale sont effectuées au niveau supérieur. Le paradigme Fog Computing prend en charge les appareils mobiles [114], le temps de réponse en temps réel ou la latence prévisible [115], l'économie de bande passante, l'amélioration de la sécurité et de la résilience, l'évolutivité, la multi localisation, l'analyse avancée et l'automatisation [116], des services rentables [117]. Le Fog Computing permet également la fédération d'infrastructures Fog afin de permettre la coopération entre les services proposés par différents fournisseurs de services [112]. Cependant, l'architecture est généralement optimisée pour un cas d'utilisation spécifique et les applications qui doivent s'exécuter sur cette infrastructure [116]. Le Fog Computing se différencie du Cloud Computing principalement par la proximité avec les utilisateurs finaux situés à la périphérie de réseaux, localisés ou répartis géographiquement et constitués d'éléments disposant de beaucoup moins de ressources que le cloud [31]. En plus des équipements de réseau, le Fog Computing peut également être réalisé dans des cloudlets et des micro-centres de données (Micro Data Centers (MDC)).

Les cloudlets [118] sont des micro-centres de données déployés géographiquement à proximité des utilisateurs finaux. Ces centres de données cloud à petite échelle sont composés d'ordinateurs dotés d'une puissance de calcul élevée qui fournissent à la fois des ressources de calcul et de stockage. La mobilité des utilisateurs implique l'utilisation d'une machine virtuelle pour instancier rapidement des applications gourmandes en calcul et en latence et migrer les services déchargés entre différents cloudlets en fonction des déplacements des utilisateurs. Cloudlet doit d'abord être découvert, sélectionné avant de commencer le provisionnement. A la fin de la session, l'instance est détruite [32]. Ils sont accessibles par Wi-Fi ce qui implique une latence élevée causée par le réseau et la commutation entre le réseau mobile et le Wi-Fi [118, 4]. Ils couvrent généralement une petite région et n'offrent aucune garantie en matière de disponibilité des ressources et sur l'évolutivité du service [4].

Les Micro Data Centres (MDC) proposés par Microsoft Research sont conçus pour étendre les centres de données cloud en tant que cloudlets. Les MDC sont des boîtiers contenant tous les équipements (informatique, stockage, réseau) nécessaires pour fournir un environnement informatique sécurisé afin d'exécuter des applications douanières nécessitant une faible latence. Les MDC sont également bien adaptés pour fournir des ressources de traitement aux terminaux sur batterie ou avec des capacités de calcul limitées. Les MDC peuvent être adaptés en fonction de la bande passante du réseau et des besoins des utilisateurs grâce à une certaine flexibilité en termes de latence et d'évolutivité de la capacité [36].

TABLE 3.9 – Avantages et inconvénients du Fog Computing

Avantages	Inconvénients
- Temps de réponse rapide en évitant la transmission de données vers le cloud [119]. - Les capacités de stockage et de traitement local empêchent la perte de données et les pannes lorsque la connectivité Internet est limitée [119]. - Les données sensibles peuvent être filtrées localement. Dans ce cas, seul le modèle de données est déplacé dans le cloud [119]. La validation, la compression et le cryptage des données sont également effectués au niveau du Fog. - La passerelle au niveau du Fog assure la compatibilité entre les appareils anciens et modernes [119] et divers protocoles de communication. - Améliore la résilience grâce à la décentralisation du traitement sur les périphériques réseau [119].	- Une panne ou une défaillance de la passerelle peut impacter le fonctionnement des milliers de périphériques. - Les capacités de traitement et de mémoire limitées ne permettent pas le déploiement d'algorithmes nécessitant des ressources importantes ou la réalisation de traitements de longue durée.

Guardo *et al.* ont proposé un framework composé de deux couches Fog filtrant et agrégeant respectivement les données, la clusterisation des données, la gestion de l'actionnement et des alertes. Le framework vise à améliorer l'équilibrage de la charge de calcul entre le Fog et le cloud afin de réduire la quantité de données à transmettre au cloud, de réduire le temps d'attente pour l'utilisateur [120]. Taneja *et al.* ont proposé un SmartHerd, une plateforme IoT dédiée à l'élevage laitier intelligent basée sur des microservices et assistée par le Fog. La passerelle IoT reçoit des données, réalise l'agré-

gation des données, le prétraitement, la classification, l'extraction des caractéristiques, l'envoi d'alertes critiques à l'agriculteur. Les données sont ensuite transmises à la plateforme IBM Watson IoT par l'intermédiaire du protocole MQTT. Dans la plateforme IBM Watson IoT, un broker récupère les données et les stocke dans une base de données Cloudant NoSQL JSON. Python Virtual Machine et Java Virtual Machine ont été utilisés comme conteneurs équivalents pour le déploiement de microservices au niveau du Fog [121]. Sharofidinov *et al.* ont décrit une architecture à 4 couches (couches de capteurs, couche de Fog, couche réseau/cloud et couche d'application) basée sur LoRa pour surveiller et prédire l'état d'une serre à partir d'un algorithme de forêt aléatoire. Dans la couche de capteurs, les capteurs acquièrent la température, l'humidité du sol et de l'air, le taux de CO_2 et l'éclairage connectés au TTGO LoRa32 (ESP32 avec puce LoRa Sx1276) qui sont transmis à la passerelle par LoRa. Au niveau de la couche Fog, une analyse préliminaire avec un algorithme d'apprentissage automatique, un diagnostic de l'état du capteur et une compression des données sont réalisés. Dans la couche réseau/cloud, les données compressées sont transmises afin d'être analysées et stockées de manière permanente. Enfin, dans la couche applicative, les données analysées sont converties sous une forme lisible pour permettre le suivi et le contrôle de la serre [119].

Le **Mobile Edge Computing** (MEC) a été proposé par l'ETSI [2] et est généralement déployé par les entreprises de télécommunications à la périphérie du réseau de communication qui se caractérisent par une latence ultra-faible et une bande passante élevée. [112, 35]. Au tout début, le Mobile Edge Computing (MEC) visait à offrir un accès en temps réel, à bande passante élevée et à faible latence aux applications dépendantes connues sous le nom de capacités de Cloud Computing ; en plus des fonctionnalités de technologie de l'information (IT), des caractéristiques du Cloud Computing. MEC est distribué à la périphérie du réseau et permet à de nouvelles applications cloud natives d'être facilement accessible en raison de la proximité les utilisateurs finaux. Le MEC permet aux opérateurs de réseaux d'ouvrir leur environnement à un nouvel écosystème Où les applications peuvent être déployées dans les macro stations de base du réseau 4G LTE (eNB), sur les contrôleurs de réseau radio 3G (RNC), les points d'accès Wi-Fi, les routeurs de réseau et les serveurs de périphérie d'entreprise. La plateforme MEC contient deux infrastructures d'hébergement principales : des ressources matérielles et un gestionnaire de virtualisation et des services de plateforme [35]. Un défi important pour le MEC est la migration de VM qui doit être un compromis entre le gain de migration et le coût de migration d'une part et la sélection de l'emplacement optimal d'autre part [32].

Tran *et al.* ont étudié le Mobile Edge Computing collaboratif dans les réseaux 5G. Le MEC étend les ressources de traitement et de stockage à la périphérie du réseau d'accès radio (RAN) tandis que la C-RAM repose sur une centralisation de la station de base au moyen de la virtualisation. Les auteurs soutiennent que les deux technologies sont

2. https ://www.etsi.org/

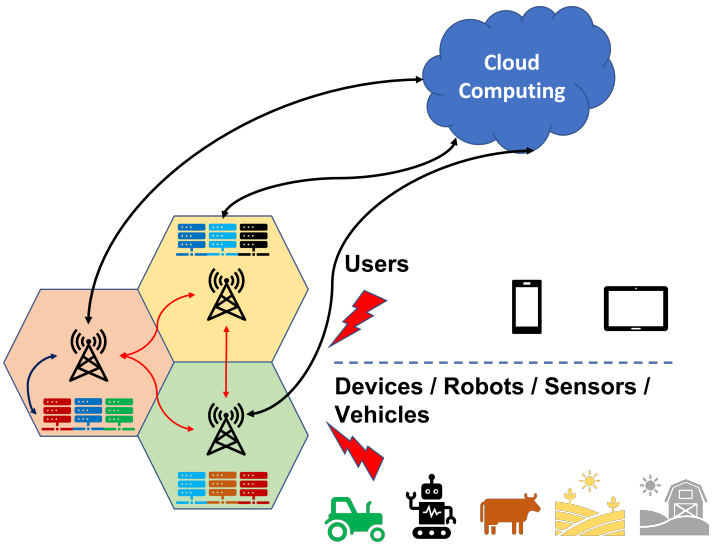


FIGURE 3.4 – Schéma général du MEC

complémentaires dans l'écosystème 5G [122].

TABLE 3.10 – Avantages et inconvénients de MEC

Avantages	Inconvénients
- Réduit les besoins en connexion, les délais de réponse, l'encombrement des autres parties du réseau [123]. - Utilise la transmission de message de bas niveau du Wi-Fi pour déterminer l'emplacement de chaque appareil (conscience de l'emplacement) [123]. - MEC Server peut être utilisé comme des ressources de calcules ouvertes pour les applications et services [123].	- Utilisable uniquement pour les appareils connectés en Wi-Fi ou 3GPP.

Fan *et al.* ont combiné le MEC avec la gestion des liaisons de données et les caractéristiques du bus CAN industriel pour la surveillance de l'eau. Le FPGA (Field Programmable Gate Arrays) Altera implémentant le bus AVALON a été utilisé pour implémenter le système. Ils ont également proposé un protocole pour modéliser les perturbations réseau aléatoires et un algorithme de déchargement des tâches en ligne, basé sur le suivi de l'exécution des tâches [124]. Valecce *et al.* ont proposé une architecture de référence robotique-5G pour l'agriculture intelligente composée de la surveillance basée sur les drones (UAV), de l'automatisation des machines et d'un serveur d'applications MEC. Les drones/satellites capturent des images haute résolution lors des patrouilles qui,

couplées aux données des capteurs, déclenchent une gestion précise des cultures. Les drones peuvent également collecter des données ou servir de station mobile 5G. Sur le terrain, le traitement d'images couplé aux données des capteurs peut être utilisé pour la prise de décision. Le MEC permet de traiter les giga octets/s de données produites par des véhicules et robots autonomes [123].

TABLE 3.11 – Évaluation de l'architecture distribuée avec nos critères

Critère	Fog	MEC
Proximité utilisateur	**(*)	***
Latence & Gigue	*	*
Stabilité du réseau	***	**
Haut débit	**	**(*)
Fiabilité	***	**
Évolutivité	*	*
Rentabilité	**	**
Maintenabilité	**	**

Le développement du Fog Computing et son homologue pour les réseaux sans fil MEC permettent de mettre à disposition des capacités de traitement plus proches des utilisateurs pour améliorer le temps de réponse néanmoins avec des capacités de calcul inférieures à celles proposées par le cloud. Deux questions se posent naturellement : Quelles stratégies d'association utiliser entre le cloud et les autres niveaux de traitement du réseau ? Comment répartir la charge entre ces différents niveaux : traitement local (Edge), réseau (Fog) et Cloud ?

3.3.5 Stratégies informatiques collaboratives

Afin de répondre à notre quatrième question de recherche : "Quelles sont les possibilités de valorisations verticales pour passer d'algorithmes entraînés dans le cloud vers des produits embarqués ou indépendants.". Nous essayons d'identifier différentes possibilités de composer des éléments architecturaux. En effet, différentes stratégies de collaboration entre les différents niveaux de traitement des données (Cloud, Fog, Edge) peuvent être envisagées en fonction des particularités des cas d'utilisation. Dans les paragraphes suivants, nous décrivons les possibilités de collaboration entre les différentes entrées de traitement et nous illustrons chacune avec quelques exemples.

Edge-Cloud vise à connecter les appareils directement au cloud qui effectue le traitement des données. Cette stratégie est souvent utilisée par les drones et les UGV qui prétraitent les données avant leur transfert vers le cloud car le traitement des images nécessite une puissance de traitement et des capacités de stockage. Le défaut de cette approche est que le délai de l'ensemble du processus depuis le transfert de données via

un protocole sans fil ou cellulaire à haut débit jusqu'à la transmission du résultat du traitement ne peut pas être garanti en raison de la fluctuation des débits de données liés aux réseaux sans fil [36]. Le traitement des données peut être réalisé en mode en ligne avec une transmission et un traitement des données en temps réel par un flux, Lambda, Kappa ou une architecture dérivée de celles-ci. Une stratégie hors ligne avec un transfert de données au moyen d'un ordinateur et d'une connexion Internet sur le cloud après le vol du drone et un traitement avec une architecture Batch, Lambda, ou Kappa ou une architecture dérivée de celles-ci est également possible. Cette dernière possibilité évite des transmissions de données coûteuses et est adapté au suivi des cultures ou du bétail qui ne nécessite pas d'action directe.

Selon Tang *et al.*, l'approche edge-cloud est principalement utilisée avec les robots intelligents pour réduire la complexité [125]. En effet, les images de drones pour être utilisables doivent être orthorectifiées et assemblées. Ces opérations nécessitent des ressources importantes en termes de puissance de calcul et de mémoire. Debauche *et al.* ont présenté une architecture Edge-Cloud pour l'analyse du comportement des bovins à partir de données IMU 9-DOF échantillonnées à 100 Hz et d'une localisation GPS échantillonnée à 0,5 Hz qui est traitée avec l'algorithme proposé par [126] en traitement par lots [94, 127]. Popescu *et al.* ont proposé un système intégré UAV-WSN-IoT où les données WSN sont collectées par les drones (UAV) avant leur transmission à la station de contrôle au sol et ensuite vers le cloud. [128]. Debauche *et al.* ont proposé une architecture pour les recherches scientifiques dédiées au trouble d'effondrement des colonies d'abeilles. Dans cette architecture, les données sont compressées sur le LoPy situé au niveau de la périphérie du réseau avant leur collecte par une passerelle LoRaWan et leur transmission à l'architecture Lambda dans le cloud où elles sont traitées [129].

Edge-Fog vise à connecter les appareils directement aux composants du réseau tels que les passerelles, les routeurs qui effectuent le traitement des données. Les avantages majeurs de cette approche sont une optimisation de la bande passante, une réduction du trafic et de la latence, une meilleure confidentialité et un niveau de sécurité amélioré [130]. Les nœuds Fog collectent, agrègent, filtrent, chiffrent, compressent et traitent les données IoT [131]. Cette voie est utilisée par exemple par les robots de traite où les données sont traitées par un ordinateur proche du robot et peuvent être visualisées à distance par l'éleveur.

Debauche *et al.* ont présenté une architecture AI-IoT pour le déploiement d'algorithmes d'intelligence artificielle et de services Internet des objets au niveau du Fog à l'aide de la conteneurisation Docker et de l'orchestration Kubernetes. Cette architecture a été développée pour déployer automatiquement des algorithmes d'IA après reconversion lorsque les performances (performance, rappel, précision) sont améliorées [132]. Debauche *et al.* ont proposé un système multi-agents (SMA) déployé au niveau de la périphérie permettant de contrôler les données anormales présentes dans les données

détectées et éventuellement de corriger ces données lorsque cela est possible. Les SMA gèrent simultanément l'irrigation par pivot, la détection des maladies et ravageurs des plantes et leur traitement. Les données sont partiellement transmises au cloud pour améliorer la détection des maladies et des ravageurs et recycler les algorithmes d'IA avant leur redéploiement au niveau du bord [133]. Debauche *et al.* ont décrit une architecture Fog dans laquelle un algorithme Gated Recursive Unit (GRU) est déployé sur NVIDIA Jetson Nano pour la surveillance des volailles en temps réel. GRU est un algorithme plus simple que LSTM. GRU est conçu pour éviter les problèmes de gradient de disparition (vanish gradient problem). Périodiquement, les données sont transmises à l'interface utilisateur implémentée dans NodeJS et hébergée dans le cloud [134].

Edge-Fog-Cloud est un paradigme dans lequel les données sont partiellement traitées dans le Fog et des traitements plus complexes sont réalisés dans le cloud. Cette voie est utilisée par le réseau de capteurs et d'actionneurs sans fil (WSAN) qui passe par une passerelle qui assure l'interconnexion entre les appareils et le réseau de transmission et les liens entre le cœur de réseau (backhaul) qui transfert ensuite les données vers le cloud. Cependant, un juste équilibre entre le cloud et le Edge/Fog Computing est requis [130] en fonction des ressources disponibles et du caractère sensible ou non de la tâche.

Taneja *et al.* ont utilisé une stratégie Edge-Fog-Cloud pour développer un système de détection de boiterie pour les bovins. Les données du podomètre sont transmises au nœud Fog au moyen d'un protocole propriétaire longue portée à 433 MHz sur une distance de 2 km. Le nœud Fog les stocke dans une base de données locale, les prétraite et les agrège. Le nœud Fog communique avec la plateforme IBM Watson IoT Platform par l'intermédiaire du protocole MQTT. Les données entrantes sont récupérées et stockées dans la base de données Cloudant [3] NoSQL JSON dans le cloud IBM. Une application mobile synchronise les données avec PouchDB, sa base de données locale via l'API REST de la base de données IBM Cloudant lorsqu'une connexion Internet est disponible [135]. Alonso *et al.* ont présenté Global Edge Computing Architecture (GECA), une architecture modulaire à plusieurs niveaux (IoT Layer, Edge Layer, Business Solution Layer) pour surveiller l'état des produits laitiers et des céréales fourragères en temps réel. Dans cette architecture, une technologie de registre distribué (Distributed Ledger) assure la sécurité de la couche IoT à la couche de solution métier. Dans la couche IoT, un ensemble d'agents appelés des oracles vérifient les données entrantes et calculent ensuite le hachage des données avec l'algorithme SHA-256 qui est stocké dans la blockchain pour vérifier la non-altération des données. En parallèle, les données sont chiffrées avec l'algorithme RSA puis envoyées à la couche Edge. La couche Edge est responsable du prétraitement des données et filtre les données transmises au cloud. Elle permet également diverses analyses de données. Dans la couche de solution métier, le stockage final, l'authentification et l'analyse pour la prise de décision sont

3. https ://www.ibm.com/cloud/cloudant

réalisés. Cette couche fournit également une base de connaissances et des API [136].

Edge-Edge est un paradigme dans lequel les appareils interagissent pour collaborer, échanger et traiter des données. Le déploiement du réseau 5G permet l'interconnexion entre drones et UGV/engins agricoles [125]. Ce réseau à haut débit permettra de développer de nouvelles collaborations entre drones (UAV) / robots (UGV) et engins agricoles par exemple un drone fournira des informations à une moissonneuse pour éviter une zone non souhaitable du champ ou éviter des obstacles. Une flotte de drones peut également collaborer pour coordonner leurs opérations sur le terrain bien sûr sous réserve de disponibilité en milieu rural, d'un réseau de transmission avec une bande passante suffisante et une latence courte ou des capacités pour communiquer entre eux en connexion directe ou dans un réseau maillé. [125].

Quatre stratégies de coopération ont été identifiées dans la littérature, dont deux utilisent le cloud, à savoir Fog-Cloud et Edge-Cloud et deux n'impliquent pas le cloud à savoir Fog-Edge cloud et Edge-Edge. Les deux premières stratégies complètent le cloud pour lui permettre de résoudre les problèmes liés aux données de production et aux secrets commerciaux, à la congestion du réseau et aux temps de réponse. Les deux autres stratégies se passent du cloud et supposent donc que les appareils/véhicules ont une capacité suffisante pour effectuer le traitement. Malgré ces stratégies de coopération entre différents niveaux de traitement, certaines questions restent en suspens : Comment stocker toutes les données brutes alors que les données sont si importantes qu'il faudrait des moyens colossaux pour les traiter ? Qu'en est-il de la sécurité ? Comment organiser la répartition des tâches entre le edge, le Fog et le cloud ? Comment assurer le fonctionnement et/ou le traitement lorsque les connexions réseau sont intermittentes ou défaillantes ? Comment améliorer la maintenabilité de ces architectures ? Telles sont les questions auxquelles les nouvelles tendances que nous décrivons dans le paragraphe suivant tentent de répondre.

3.4 Nouvelles tendances

Dans cette section, nous présentons deux architectures émergentes non basées sur les architectures Batch ou/et temps réel ou leurs dérivés. Les nouvelles tendances sont des éléments supplémentaires qui permettent d'enrichir l'analyse de la section 4 afin d'aborder la troisième question de recherche.

La **Microservices Architecture** (MA) est un nouveau modèle de conception de logiciel système qui divise une application monolithique complexe en microservices dédiés à une seule fonction. Les microservices corrigent les défauts des applications monolithiques dans lesquelles l'amélioration des performances du service nécessite un

déploiement multiple ; un changement dans une fonction peut affecter tout un programme monolithique en raison des fortes dépendances entre les composants ; tout le monolithe utilise une seule pile technologique et des normes de développement qui limitent les possibilités de résoudre les problèmes d'hétérogénéité physique.

Les avantages de cette architecture sont l'utilisation d'un mécanisme de communication léger pour interagir entre les services avec une surcharge minimale [137]. La conception proposée par [137] est composée de 8 microservices (Geo, Security, Tentant, Devices, Big Data, Automation, AI et Application) et d'un service de coordination de base. Ces services fournissent respectivement : (1) Geo, une couche SIG pour restituer les données ; (2) Sécurité, gestion des utilisateurs/groupes/rôles, contrôle d'accès, mécanisme d'administration et d'authentification ; (3) prise en charge de plusieurs applications IoT avec un seul cœur ; (4) plug-ins d'appareils et protocoles de communication pour la détection et l'activation ; (5) persistance évolutive pour stocker les données ; (6) traiter, analyser les événements et informer le participant approprié ; (7) Outils d'intelligence artificielle pour le big data IoT ; (8) composants pour interagir avec les interfaces client ; (9) prise en charge de l'échange de données par message avec les appareils. Les auteurs soutiennent que leur approche est plus flexible, évolutive et indépendante de la plateforme.

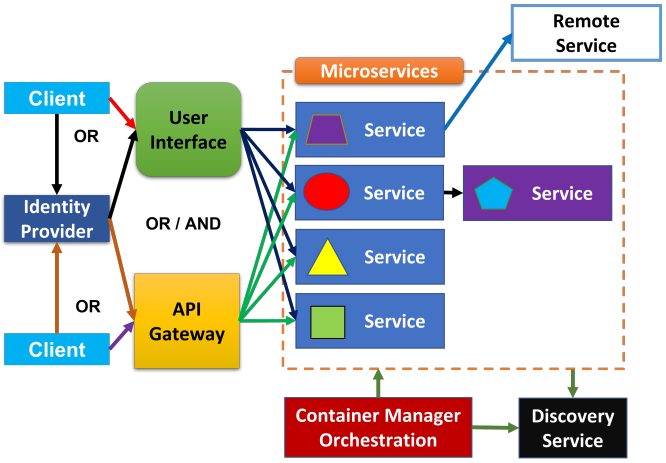


FIGURE 3.5 – Schéma général de l'architecture des microservices

Bixio *et al.* ont proposé une architecture de traitement de flux événementielle basée sur des micro-services de proxy, d'adaptateur et de traitement de données. Cette architecture étend la plateforme IoT Senseioty et utilise le framework Java OSGi [138].

L'**Architecture Data Lake** (DLA) [139, 85] permet le stockage de gros volumes de données de tous types : données brutes dans son format natif, structurées, semi-structurées, de manière économique. Dans cette architecture, les données sont stockées dans leur format natif jusqu'à ce qu'elles doivent être traitées par des moteurs [85] qui

permettent une transformation et un raffinement rapides des données stockées quelle que soit la quantité de données stockées. Ce type d'architecture permet de consommer tous types de données (logs, web services, base de données, fichiers, etc.); différents systèmes d'ingestion consomment les données, puis les stockent dans un référentiel de données. Une fois les données stockées, les systèmes de requête peuvent interroger le data lake. Cette architecture est considérée dans le monde de l'entreprise comme une évolution des architectures existantes. L'avantage de l'architecture Data Lake est qu'elle peut stocker facilement et à moindre coût de grandes quantités de données. Le Data Lake est particulièrement bien adapté au stockage de données dans un format atypique. En entreprise, les Data Lakes sont utilisés en plus des entrepôts de données. Les Data Lakes ne sont cependant pas adaptés à l'évaluation de la qualité des données, les données peuvent être placées dans des Data Lakes sans contrôle de contenu et les performances sont également moins bonnes que sur des infrastructures spécialement conçues et optimisées. Le Lakehouse est une variante du Data Lake où le stockage des données généralement réalisé avec Hadoop dans le data lake est remplacé par un stockage distribué tel qu'Amazon S3, Azure Blob Storage, Google Cloud Storage, et l'analyse est réalisée directement par une infrastructure gérée par des fournisseur de services cloud tels qu'Amazon Athena, EMR ou Databricks, Google Data proc, Azure HDInsight. La figure 3.6 fournit une comparaison entre la structure de data lake et de lakehouse.

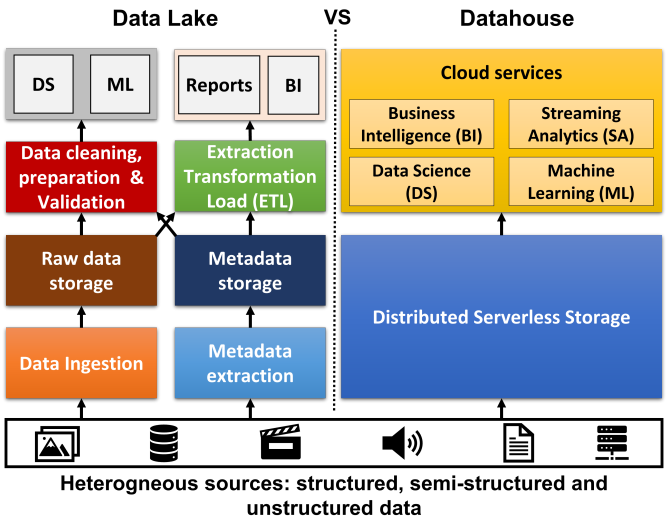


FIGURE 3.6 – Schéma général du Data Lake et du Lakehouse

Il est crucial en agriculture d'explorer des ensembles de données provenant de différentes sources. L'architecture Datalake est indiqué pour gérer la complexité des écosystèmes agricoles, et permet de centraliser toutes les sources de données pour trouver de nouvelles corrélations. [140]. Un Data Lake fournit des vues basées sur des métadonnées. Il est néanmoins nécessaire de disposer d'outils d'analyse avancés pour

la modélisation prédictive et l'analyse statistique. Lopez *et al.*ont utilisé un data lake pour réaliser la fusion de données de différents domaines dans le contexte de l'agriculture intelligente [141]. Gallinucci *et al.* [142, 143] présentent une architecture innovante 3 tiers, appelée Mo.Re.Farming (MOnitoring and REmote system for a more durable FARMING) basée sur un datalake utilisant Apache Hadoop et stockant des données brutes non structurées, des données structurées et semi-structurées et dans laquelle les activités de traitement et d'enrichissement ultérieures sont séparées. Un magasin de données opérationnelles (ODS) utilisant PosgreSQL avec PostGIS est utilisé pour stocker des données structurées et détaillées et répondre aux limites des solutions de Big Data en gérant correctement les données géographiques de champ continu. Enfin, un cube spatial permet le traitement analytique spatial en ligne (SOLAP). Neves *et al.* ont décrit une architecture dans laquelle les données brutes sont stockées dans un datalake. Ensuite, les ETL transforment les données pour qu'elles soient stockées dans une base de données. Les données sont enrichies grâce à une base de connaissances et à leur exploration est réalisée par des algorithmes de data minning (machine learning). Le résultat du traitement est filtré pour améliorer la qualité des données structurées [144].

L'analyse des nouvelles tendances montre que : (1) La **Micro service architecture** permet de décomposer les monolithes en micro services faiblement couplés ce qui facilite leur maintenance tout en permettant aux autres services de continuer à fonctionner. De plus, ce type d'architecture est plus résilient car si l'un des services est en panne, les autres services du fait du couplage faible peuvent continuer à fonctionner au moins en mode dégradé ; (2) Le **Data Lake/DataHouse** propose une nouvelle approche Load Transform Extract (LTE) où les données sont d'abord stockées dans leur format d'origine qui sont ensuite transformées afin d'extraire des informations utiles. Ce paradigme est particulièrement bien adapté lorsque la quantité de données est si importante que le traitement de toutes les données est trop coûteux. Dans ce cas, les données peuvent être échantillonnées afin d'obtenir des informations. Ce paradigme est également bien adapté si l'on veut conserver également des données brutes ou compléter une architecture générique, par exemple, pour stocker des données qui seront traitées en mode Batch.

3.5 Applications en agriculture 4.0

L'agriculture 4.0 utilise notamment des véhicules aériens sans pilote (UAV) équipés de divers capteurs afin d'améliorer le temps de collecte des données, en réduisant le coût d'acquisition par rapport aux technologies traditionnelles de phénotypage de terrain. Toutes ces données collectées doivent être traitées, analysées et visualisées rapidement. **Agroview** [145] est une plateforme qui a développé une application basée sur le cloud et l'IA pour étudier et évaluer le domaine agricole, déployée sur Amazon

Web Services (AWS). Un site Web permet le téléchargement d'images ou d'orthomosaïques existantes, la consultation pour chaque champ d'arbres, le nombre d'arbres, le nombre d'arbres manquants, la superficie du champ, la hauteur moyenne des arbres, la superficie de la canopée, etc. Le site Web permet également l'assemblage d'un orthomosaïque et la génération d'un modèle numérique de surface (MSN). Un algorithme de détection d'arbres développé en C permet la détection d'arbres individuels et des arbres manquants, et d'estimer les paramètres des arbres tels que la hauteur, la superficie de la canopée, l'estimation de la santé/du stress. Le pipeline de traitement utilise un Faster R-CNN pour détecter la région d'intérêt (ROI) et le réseau ResNet101 permet de détecter les arbres et l'orientation des rangées. Ensuite, le classificateur Yolo utilisant Darknet19 est appliqué le long de chaque rangée d'arbres pour obtenir une détection plus précise.

Le suivi des cultures particulièrement les plus sensibles d'entre elles comme le safran est crucial. Le **DIAS Architecture** [18] utilise différents capteurs au sol et foliaires pour surveiller en temps réel le processus de culture du safran 24/24h. Ces données sont transmises par LoRaWAN avec le protocole IPv6 et le protocole MQTT-SN au broker de FIWARE. Le broker gère tous les périphériques réseau au moyen de seize types de messages échangés selon le modèle de publication-abonnement. L'API FIWARE NGSI est en charge de la consommation, de la souscription et du traitement de l'ensemble des informations collectées et de leur publication. Ensuite, les données sont stockées et analysées avec un algorithme de forêt aléatoire qui permet d'extraire des informations sur la croissance et la santé des cultures. Les indices de végétation : l'indice de différence normalisée (NDI), l'indice de verdure excessive (ExG) sont calculés avec les outils de traitement d'image PiX4D [4]. La détection d'objets basée sur l'analyse d'images (OBIA) est utilisée pour reconnaître les mauvaises herbes ou discriminer les espèces. Enfin, les données collectées sont classées et évaluées en conséquence avec les valeurs de l'indice de végétation, le niveau d'humidité et l'état de développement des plantes en utilisant le framework Apache Spark pour l'analyse du Big Data et Waikato Environment (WEKA) un cadre spécialisé dans l'exploration de données pour produire des rapports et des prévisions.

La prise de décision est une tâche très importante dans les activités des agriculteurs mais avec la quantité de données toujours croissante, ils rencontrent des difficultés d'une part pour prendre les bonnes décisions concernant la gestion agricole et d'autre part pour traduire ces données en connaissances pratiques [20]. Par conséquent, il existe un besoin de plates-formes de système d'aide à la décision agricole (ADSS) pour aider les agriculteurs à prendre des décisions précises fondées sur des preuves. Par exemple, **Watson Decision Platform for Agriculture** combine la plateforme IBM Watson qui est une IA qui permet de formuler les requêtes en language naturel avec les drones et les ressources de calcul du Cloud d'IBM pour détecter les maladies des cultures à partir d'images de drones. Il est également possible d'optimiser le temps des opérations de ré-

4. https ://www.pix4d.com/

colte pour obtenir un meilleur prix sur le marché commercial. Le deuxième exemple est **Digital Farming System** [5] tire parti de la vision par ordinateur, du Cloud Computing et de l'IA pour proposer un meilleur moment pour les opérations de l'entreprise, notifier lorsqu'une culture est infectée par n'importe quel maladie. **Smart Irrigation Decision Support System** (SIDSS) est composé d'une part d'un ensemble de capteurs et d'une station météo et d'autre part un système d'aide à la décision basé sur deux algorithmes d'apprentissage automatique. La régression des moindres carrés partiels (PLSR) pour déduire les variables inutiles et les systèmes d'inférence neuro floue adaptative (ANFIS) sont utilisés pour minimiser les erreurs estimées sous un seuil cible [146]. Les SIDSS génèrent une planification de la quantité d'eau et du temps pour l'irrigation. Le **Système multi-robot sense-act** [147] est un planificateur de véhicules aériens et terrestres qui assignent des tâches aux unités de travail les plus appropriées. un algorithme de recherche d'harmonie est utilisé pour optimiser le plan pour les drones tandis que la métaheuristique est exécutée pour les véhicules terrestres.

3.6 Conclusion

Notre revue est motivée par 4 questions de recherche : (1) Quelles architectures de stockage et de traitement sont les mieux adaptées aux applications Agriculture 4.0 et répondent à ses particularités ? (2) Les architectures génériques peuvent-elles répondre aux besoins des cas applicatifs Agriculture 4.0 ? (3) Quelles sont les possibilités de valorisation horizontale qui permettent le passage de la recherche à l'industrialisation ? (4) Quelles sont les possibilités de valorisation verticale pour passer d'algorithmes formés dans le cloud à des produits embarqués ou autonomes ?

L'analyse de la littérature montre qu'un grand nombre d'architectures coexistent. Néanmoins, les architectures Lambda et Kappa semblent émerger comme des architectures génériques. Celles-ci doivent généralement s'accompagner de composants architecturaux complémentaires pour répondre à des besoins spécifiques et s'inscrire dans une stratégie de stockage et de traitement dans laquelle l'architecture cloud est un composant de la chaîne ou peut aussi et plus rarement être absente lorsque des stratégies Fog-Edge ou Edge-Edge sont implémentées.

Le Cloud Computing centralisé traditionnel continuera à rester une partie importante des systèmes informatiques [32], pour les sciences même si d'autres paradigmes apparaissent. En effet, le Cloud, le Fog et le Edge Computing interagissent les uns avec les autres pour former un continuum de services mutuellement bénéfiques et interdépendantes. Certaines fonctions sont naturellement plus adaptées ou avantageuses à un niveau plus qu'à un autre en fonction d'exigences en temps de réponse, de calcul ou de tolérance de latence. Cependant, le cloud ne peut pas être complètement remplacé par le Fog et le Edge Computing, car certaines tâches gourmandes en calcul ne peuvent être

5. http ://prospera.ag/

traitées qu'au niveau du cloud qui a la puissance de calcul et les capacités de stockage [36]. En Agriculture 4.0, c'est notamment le cas pour le traitement d'images satellites, la formation d'algorithmes d'intelligence artificielle comme le Deep Convolutional Neural Network (DCNN).

Les nouvelles tendances permettent de répondre à divers problèmes : (1) Le Data Lake/Data House offre une alternative plus économique au stockage cloud massif en base de données. Dans ce paradigme, toutes les données sont stockées en état et ne se transforment que lorsqu'elles doivent être exploitées. Cette approche est particulièrement intéressante d'une part lorsque toutes les données ne sont pas exploitées et d'autre part lorsqu'une décision ou une action n'est pas attendue dans l'immédiat. Le data lake permet également la fusion de données agricoles d'origines diverses dans différents formats et granularités ; (2) L'architecture microservice offre la possibilité de découpler les architectures monolithiques en microservices faiblement couplés. Ces services peuvent être plus facilement associés, maintenus ou évolués indépendamment. La combinaison de ces microservices permet de développer plus facilement de nouveaux services pour l'utilisateur final qui sont également plus faciles et plus rapides à évoluer en fonction des évolutions technologiques et des besoins.

Deux tendances d'utilisation des architectures de traitement coexistent, d'une part les utilisateurs d'une plateforme IoT payante ou open source, et d'autre part, les utilisateurs qui développent des architectures spécifiques pour mettre en œuvre des cas d'usage particuliers. Du point de vue de la transférabilité, on comprend qu'il est plus facile pour des infrastructures payantes toutes faites et qu'elle peut être limitée pour des infrastructures open source clé en main où le type de licence adopté peut poser des problèmes. Or, la pérennité des infrastructures payantes est conditionnée au développement accordé par l'entreprise qui les gère et à sa santé financière. Le développement d'architectures basées sur des briques logicielles payantes est facilité mais sa pérennité est conditionnée par la disponibilité et la maintenance de ces briques logicielles. Quant à la transférabilité, elle est liée à l'acquisition de licences ad hoc. Le développement d'architecture utilisant des briques logicielles open source issues de fondations comme Apache Foundation permet de ne pas être limité par des licences mais est dépendant des développements et de la maintenance effectués par la communauté des développeurs. Ces briques logicielles peuvent être abandonnées par la communauté, l'entreprise qui les parraine ou la fondation qui les héberge. Le développement d'une architecture pérenne passerait par une indépendance vis à vis des briques logicielles qui permettrait de les changer facilement d'une part lorsque l'une d'entre elles disparaît ou si une nouvelle brique logicielle plus performante apparaît. L'architecture devra par ailleurs être complétée par une extension au niveau du Fog pour adresser plus largement les cas d'utilisation du Smart Farming. Finalement, le contexte de la recherche impose de document les données en vue de pouvoir les réutiliser ultérieurement. Une attention particulière devra donc être données aux métadonnées compagnons qui permettent décrire le contexte expérimental et jeter les bases de développement sémantiques ultérieurement.

Deuxième partie

Architecture cloud innovante

4

Conceptualisation et design d'une architecture cloud innovante

Plan du chapitre

Résumé

Sur base des analyses et des conclusions tirées de l'état de l'art présenté au chapitre 3, une architecture cloud distribuée a été conçue de manière générique et modulaire. Une proposition de choix de briques logicielles est ensuite formulée. Dans ce chapitre, nous décrivons les étapes de conception et nous justifions les choix qui ont été posés pour la création d'une solution architecturale dans le cloud comprenant deux volets. Le premier volet est une architecture lambda unifiée dédiée à la collecte, au stockage, au traitement des données issues de l'Internet des Objets et qui sera ensuite étendue aux données de recherche. Le deuxième volet de la solution architecturale proposée est une plateforme d'hébergement d'applicatifs élaborés dans différents langages et s'appuyant sur les frameworks les plus couramment utilisés.

Publications liées à ce chapitre

O. Debauche, S. Mahmoudi, P. Manneback, M. Massinon, N. Tadrist, F. Lebeau, S.A. Mahmoudi, "Cloud architecture for digital phenotyping and automation," In : 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), Rabat, 2017, pp. 1-9. doi : 10.1109/CloudTech.2017.8284718.

O. Debauche, M. El Moulat, S. Mahmoudi, P. Manneback and F. Lebeau, "Irrigation pivot-center connected at low cost for the reduction of crop water requirements," In : 2018 International Conference on Advanced Communication Technologies and Networking (CommNet), Marrakech, 2018, pp. 1-9. doi : 10.1109/COMMNET.2018.8360259.

M. El Moulat, **O. Debauche**, L. Ait Brahim, "Monitoring System Using Internet of Things for Potential Landslides". *Procedia Computer Science*, 2018, **132** : 26-34. doi : 10.1016/j.procs.2018.07.140.

O. Debauche, S.A. Mahmoudi, S. Mahmoudi, P. Manneback, "Cloud Platform using Big Data and HPC Technologies for Distributed and Parallels Treatments". *Procedia Computer Science*, 2018 : **141** : 112-118. doi : 10.1016/j.procs.2018.10.156.

O. Debauche, M. El Moulat, S. Mahmoudi, S. Boukraa, P. Manneback, F. Lebeau, "Web Monitoring of Bee Health for Researchers and Beekeepers Based on the Internet of Things". *Procedia Computer Science*, 2018, **130** : 991-998. doi : 10.1016/j.procs.2018.04.103.

F. Nolack Fote, A. Roukh, S. Mahmoudi, S.A. Mahmoudi, **O. Debauche**, "Toward a Big Data Knowledge-Base Management System for Precision Livestock Farming". *Procedia Computer Science*, 2020 : **177** : 136-142. doi : 10.1016/j.procs.2020.10.021.

R. Ait Abdelouahid, **O. Debauche**, A. Marzak, "Internet of Things : a new Interoperable IoT Platform. Application to a Smart Building". *Procedia Computer Science*, 2021 : **191** : 511-517. doi : 10.1016/j.procs.2021.07.066.

4.1 Introduction

A la lumière des spécificités des cas d'utilisation de recherche en Smart Farming décrits au chapitre 2, l'architecture cloud que nous proposerons dans cette thèse devra :

— Traiter des flux de données ou les retraiter a posteriori. Mais également traiter des lots de données importés manuellement et provenant de manière non exhaustive de fichiers, bases de données, files d'attente, capteurs, etc.
— Ingérer de grandes quantités de données de nature variées, rapidement et d'en assurer la disponibilité en quasi-temps réel [1].
— S'adapter à des variations importantes de débits de données à ingérer et à traiter.
— Résister aux pannes et continuer à assurer le service même en cas de défaillance d'une partie des nœuds du cluster.
— Être capable de traiter des données de nature très variées : données temporelles, sons, images, vidéos, signaux, ...

L'analyse de l'état de l'art a montré que les architectures génériques Kappa et Lambda sont des choix adaptés pour la collecte, le traitement et le stockage de données de l'Internet des Objets.

Par ailleurs, ces données devront pouvoir être mise à disposition en vue d'être exploitées de manière polyvalente. L'architecture cloud devra également permettre l'hébergement d'applicatifs hétérogènes développées dans différents langages de programmation et utilisant une large diversité d'intergiciels (*Framework*).

4.2 Analyse conceptuelle

Dans le cadre de notre cas d'application de recherche en Smart Farming nous sommes amenés à devoir collecter des données provenant de grandes quantités de capteurs hétérogènes du point de vue matériel, fonctionnement et protocole de communication. Cette diversité dans les capteurs implique d'avoir de la souplesse pour pouvoir s'adapter à des formats de données brutes variés. Les métadonnées qui accompagnent les données brutes ajoute une hétérogénéité supplémentaire qu'il faudra également gérer. Les données sont stockées sous forme de base de données SQL [2] et NoSQL [3], de

1. Nous entendons dans le contexte de cette thèse, par quasi-temps réel, une durée de traitement d'au plus quelques secondes

2. Bases de données relationnelles où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables et respectant les propriétés ACID (atomicité, cohérence, isolation et durabilité).

3. (*Not only SQL*) désigne une famille de systèmes de gestion de base de données qui s'écarte du paradigme classique des bases relationnelles par un non-respect strict des propriétés ACID. Dans ce type

fichiers texte ou dans des formats structurés tels que CSV, TSV, XML, JSON ou provenir de flux de données en temps réel ou à la demande obtenus à partir d'API ou de services web. Les chaînes de traitement à appliquer a ces différentes sources de données vont bien entendu différer en fonction de leur structure et de leur format de stockage.

L'architecture cloud devra pouvoir retraiter des données antérieures de manière à les transformer sous une forme adaptée pour leur stockage et leur exploitation. Elle devra également pouvoir traiter les données qui arrivent en temps réel sous forme de flux de données périodiquement. Ces flux de données peuvent être aussi bien des données temporelles ou multimédia et peuvent arriver sous forme de rafales de données.

4.3 L'importation des données archivées

De nombreuses données sont archivées sous forme de fichiers structurés (CSV, TSV, XML, JSON, etc.) ou de bases de données (Figure 4.1) que nous dénommerons par la suite lots. Ces données proviennent d'archives ou d'exportation d'autres systèmes et doivent au préalable être transformées par traitements successifs pour atteindre une forme plus adaptée (**1**). Les données transformées peuvent alors être consommées et stockées à long terme sous une forme qui permet de les retrouver rapidement et de les consulter aisément (**2**). Elles sont ensuite archivées pour permettre le stockage à long terme (**3**).

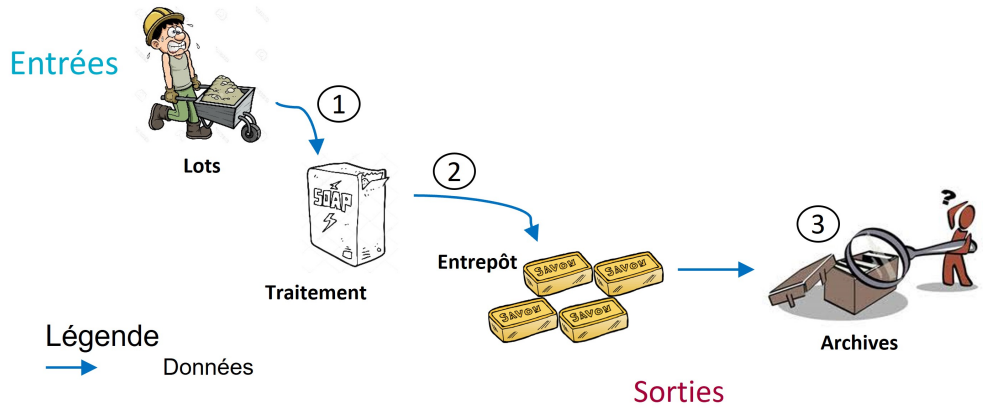


FIGURE 4.1 – Traitement des données en lots

La Figure 4.2 montre la traduction sous forme d'architecture de la Figure 4.1. La correspondance de la numérotation est respectée entre les deux figures.

de base de données, les données sont organisées sous forme de documents, de graphes, de colonnes, ou d'associations clé-valeur.

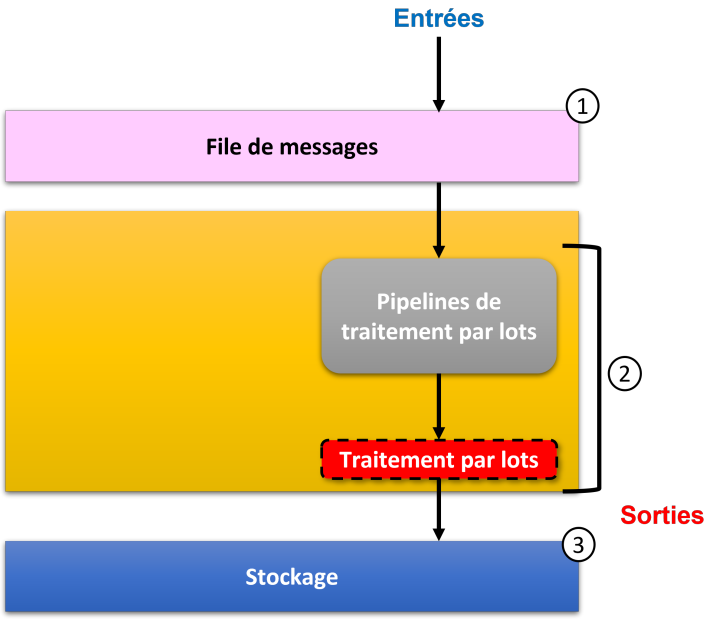


FIGURE 4.2 – Traduction sous forme d'architecture du traitement des données en lots

4.4 Traitement des flux de données temporelles

L'objectif du traitement des flux de données temporelles est de stocker les données provenant de flux sous une forme pérenne qui soit facilement récupérable et utilisable par la suite.

Comme le montre la Figure 4.3, sous la forme d'une métaphore conceptuelle, les entrées revêtent la forme de flux de données provenant de sources diverses et qui doivent être mises en file d'attente pour pouvoir être incorporées de manière plus aisée par la suite (**1**). Toutes les données ne sont pas forcément dans une forme correcte. De plus, lors de la transmission, des données peuvent être corrompues, tronquées ou des paquets des données mal formés peuvent être envoyées par des senseurs défectueux. Il est par conséquent nécessaire de procéder à un traitement (**2**) pour séparer les données non conformes en vue d'une part d'accélérer l'ingestion des données correctes (**3**) et corriger dans la mesure du possible les données incomplètes ou corrompues. Les données sont ensuite consolidées en paquets (**4**) en vue d'être stockées à long terme (**5**). Les sorties sont les flux de données mis sous format de tableaux de données horodatées en vue de leur sauvegarde (**6**).

La Figure 4.4 montre la traduction sous forme d'architecture de la Figure 4.3. La correspondance de la numérotation est respectée entre les deux figures.

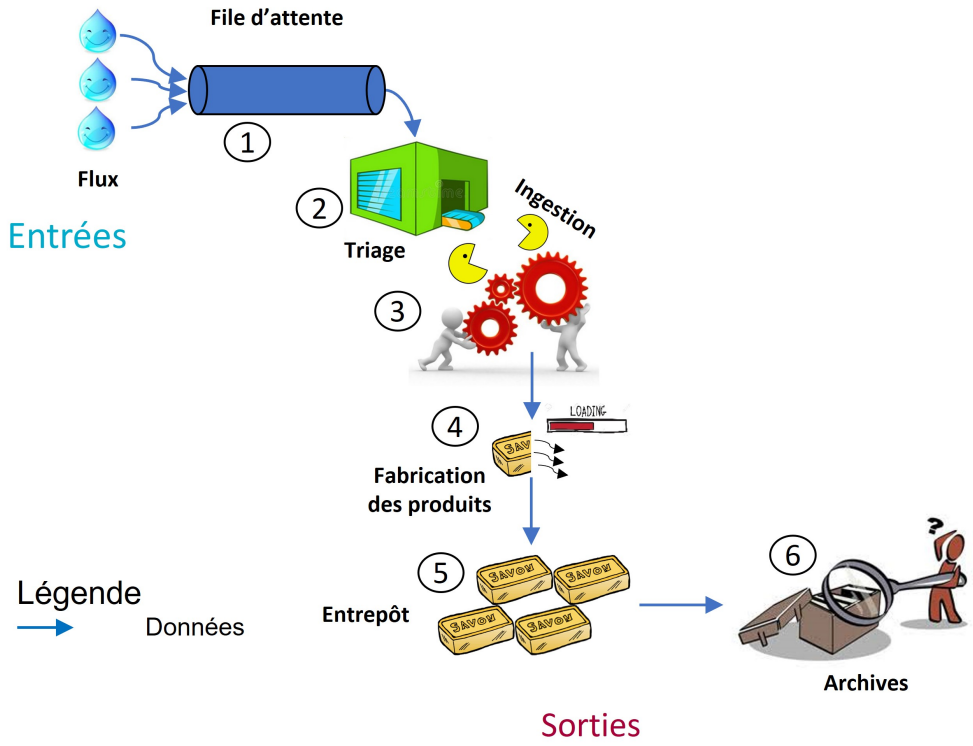


FIGURE 4.3 – Le traitement des flux de données : Métaphore conceptuelle

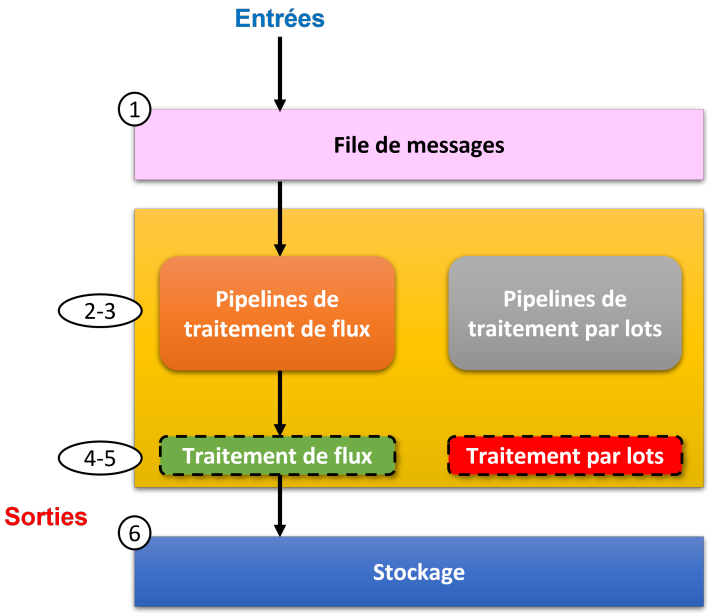


FIGURE 4.4 – Traduction sous forme d'architecture du traitement des flux de données

4.5 Stockage et accès aux données temporelles

Les deux types de données (flux et lots) doivent être stockées dans un entrepôt de données commun.

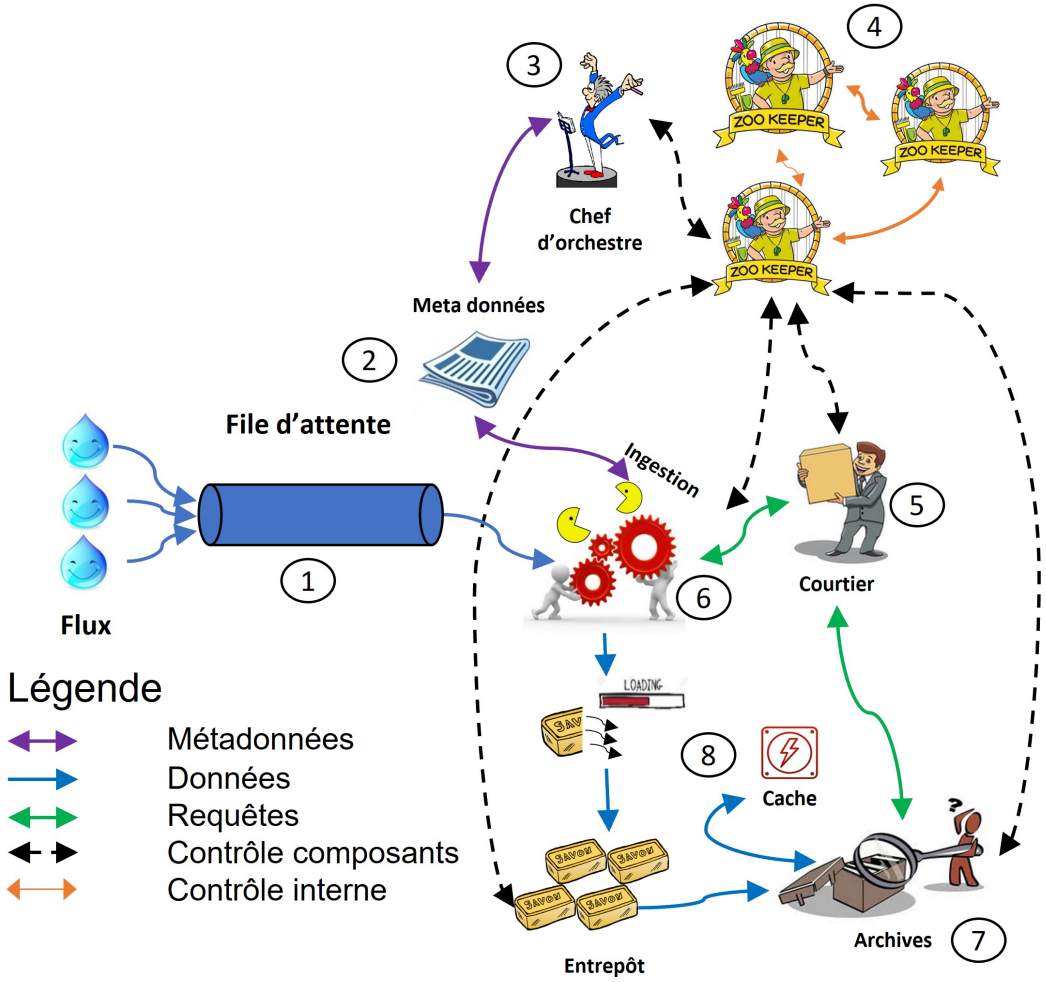


FIGURE 4.5 – Stockage et accès aux données

Comme le montre la Figure 4.5 sous la forme d'une métaphore conceptuelle plus détaillée, les données sont mises en attente dans une file de message (**1**) avant d'être ingérées (**6**). Parallèlement à l'ingestion des données, des métadonnées identifiant le container dans lequel sont entreposées les données (**2**) sont créées et stockées dans une base de données. Lors de l'ingestion des données, un chef d'orchestre (**3**) coordonne le fonctionnement de l'ensemble des acteurs du système de stockage et d'accès aux données. Une équipe de gardiens de zoo (**4**) veille à la bonne santé de l'ensemble des

acteurs et les remplace, dans la mesure du possible, en cas de défaillance. Les données ingérées, les plus récentes sont stockées temporairement en mémoire afin de pouvoir y accéder rapidement avant leur stockage dans l'entrepôt. Les données en mémoire sont régulièrement stockées dans l'entrepôt afin de libérer la mémoire pour les nouvelles données. Un courtier (**5**) est dans l'attente des requêtes provenant des clients (applications, systèmes, etc.). Quand il reçoit une requête, il interroge les gardiens de zoo qui lui indiquent vers qui rediriger la requête : ingesteurs (**6**) et/ou les archives (**7**) pour obtenir les données nécessaires pour répondre à la demande du client. Les archives récupèrent les containers abritant les données dans l'entrepôt. Un système d'accès rapide (**8**) en relation avec les archives garde en mémoire les données les plus souvent demandées pour y accéder plus rapidement. Le courtier agrège les données reçues des différentes parties et adresse le résultat au requérant.

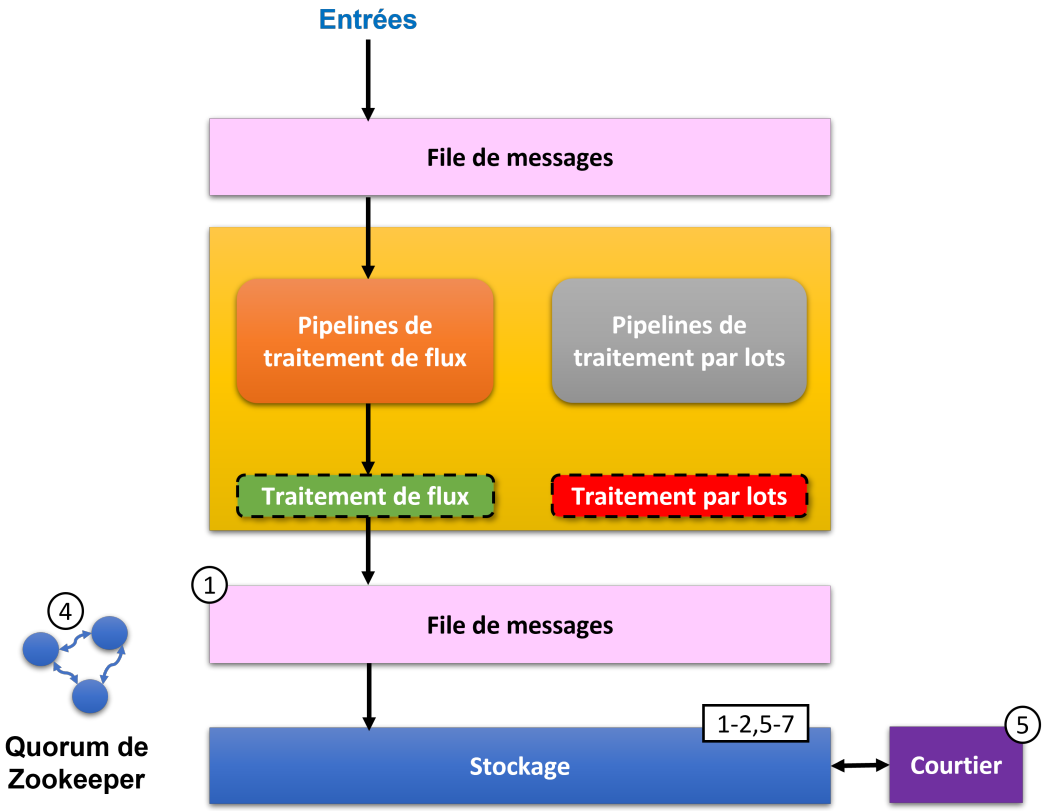


FIGURE 4.6 – Traduction sous forme d'architecture du stockage et de l'accès aux données

La Figure 4.6 montre la traduction sous forme d'architecture de la Figure 4.5. La correspondance de la numérotation est respectée entre les deux figures.

4.6 Hébergements des applicatifs

Les applicatifs développés par les chercheurs utilisent de nombreux cadriciels (*Frameworks*) qui doivent être hébergés sur une ou plusieurs machines pour pouvoir fonctionner et effectuer les traitements de données. De plus, il n'est pas possible d'héberger tous les cadriciels et donc le recours à une technologie d'isolation par conteneurisation peut être nécessaire pour héberger les applicatifs hétérogènes (Figure 4.7). De plus, le développement de nouveaux modèles / applications doivent pouvoir être versionnés. Il faudra également utiliser de bonnes pratiques de développement comme l'évaluation de la qualité logicielle, l'intégration continue et le déploiement continu.

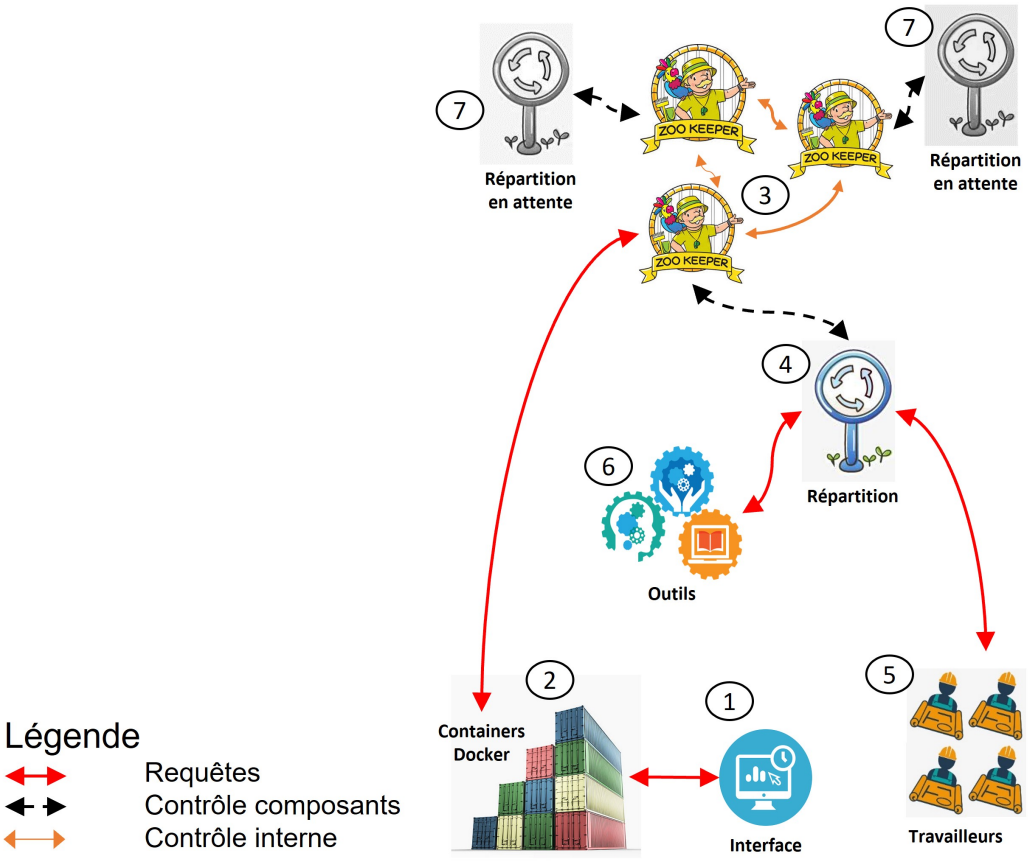


FIGURE 4.7 – Hébergement des applicatifs et utilisation des cadriciels

Une interface web (**1**) fournira aux utilisateurs tous les outils nécessaires pour notamment déployer les applications et gérer les droits d'accès aux applicatifs et aux ensembles de données. En outre, les propriétaires des applications pourront les déployer sur la plateforme dans des environnements virtualisés ou conteneurisés pour garantir

leur isolation de l'infrastructure (**2**). Une demande est adressée aux gardiens de zoo (**3**) pour déterminer quel est le répartiteur actif. Le répartiteur de tâches actif (**4**) reçoit les demandes de tâches et les données provenant des applications hébergées dans les conteneurs (**2**). Il vérifie la disponibilité des nœuds de travail (**5**), installe, le cas échéant, le ou les frameworks nécessaires (**6**) à l'exécution des tâches avant de rediriger les tâches vers les nœuds de travailleurs (**5**) qui hébergent les exécuteurs des cadriciels (**6**) qui effectuent les tâches demandées. Les résultats sont renvoyés au coordinateur qui les redirige ensuite à l'application demanderesse. Une équipe de gardiens de zoo (**3**) veille à la bonne santé du répartiteur de tâches et le remplace par un nouveau (**7**) en cas de défaillance.

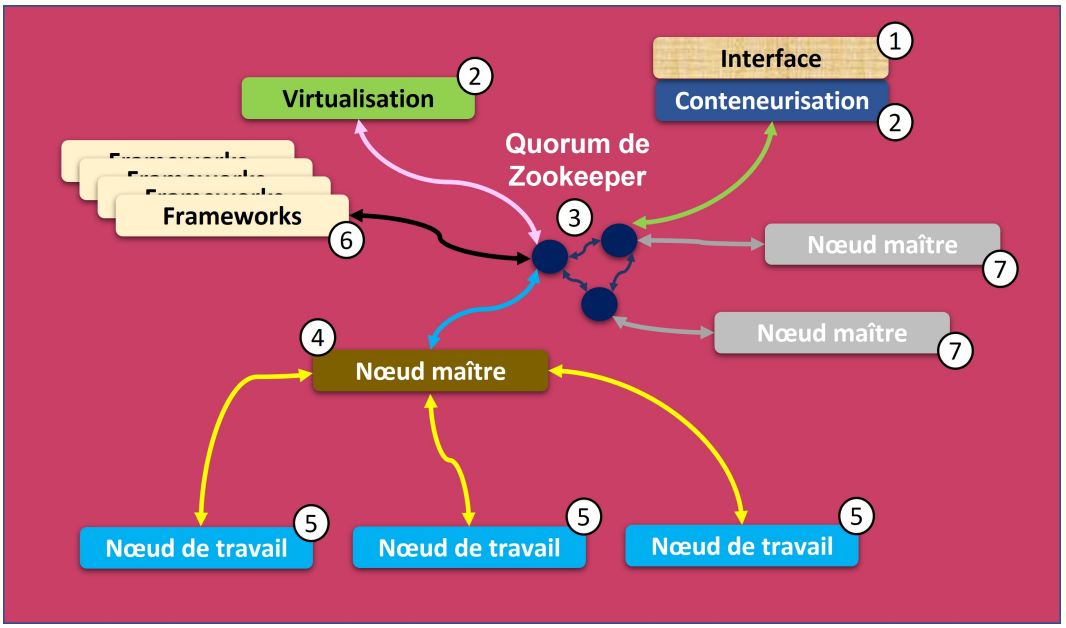


FIGURE 4.8 – Traduction sous forme d'architecture de l'hébergement d'applicatifs

La Figure 4.8 montre la traduction sous forme d'architecture de la Figure 4.7. La correspondance de la numérotation est respectée entre les deux figures.

4.7 Interaction entre les deux architectures

Comme le montre la Figure 4.9, les applicatifs hébergés dans les conteneurs (**1**) envoient les requêtes au douanier (**2**) pour obtenir les données nécessaires à leur exécution. Le douanier vérifie les autorisation d'accès, récupère les données de types binaires (images, vidéos, sons, etc.) directement dans le magasin local (**3**) et envoie les requêtes relatives aux données temporelles au courtier (**4**). Les applications peuvent ensuite interroger le quorum de gardien de zoo (**5**) pour connaitre le répartiteur actif

et lui transmettre les requêtes de traitement. Celui-ci interroge les travailleurs (**7**) pour connaitre leur disponibilité, déployer les composants logiciels manquants (**8**) et transférer les tâches vers les travailleurs (**7**).

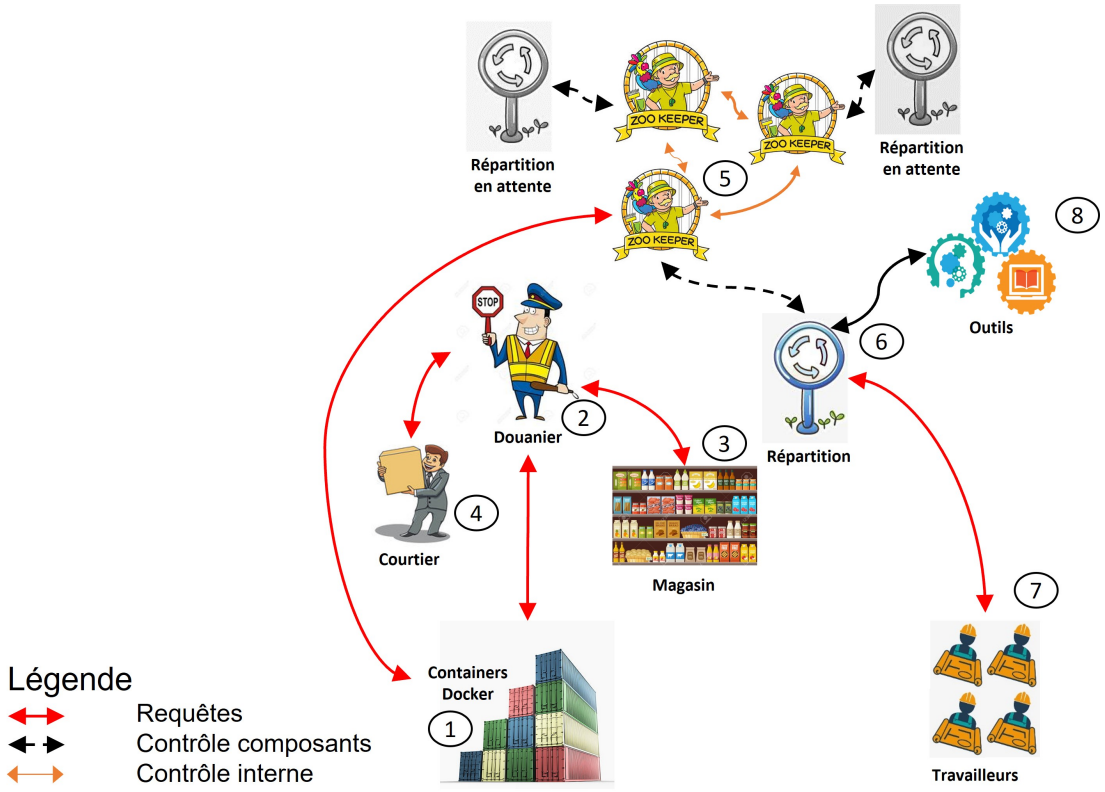


FIGURE 4.9 – Communication entre le stockage des données et les applications

Le douanier offre un service qui assure le contrôle d'accès aux données de la plateforme de stockage et vérifie que l'application qui effectue une demande (requête) est d'une part autorisée à effectuer des requêtes sur la plateforme de données et qu'elle accède uniquement aux ensembles de données pour lesquels elle a préalablement été autorisée. Le douanier tient à jour un journal de toutes les demandes et vérifications effectuées lors de la communication entre les deux plateformes cloud. Les données résultantes des requêtes sont anonymisées afin de garantir la protection de la vie privée des parties prenantes qui ont consenti au partage de leurs données à des fins de recherche et d'exploitation industrielle.

La Figure 4.10 montre la traduction sous forme d'architecture de la Figure 4.9. La correspondance de la numérotation est respectée entre les deux figures.

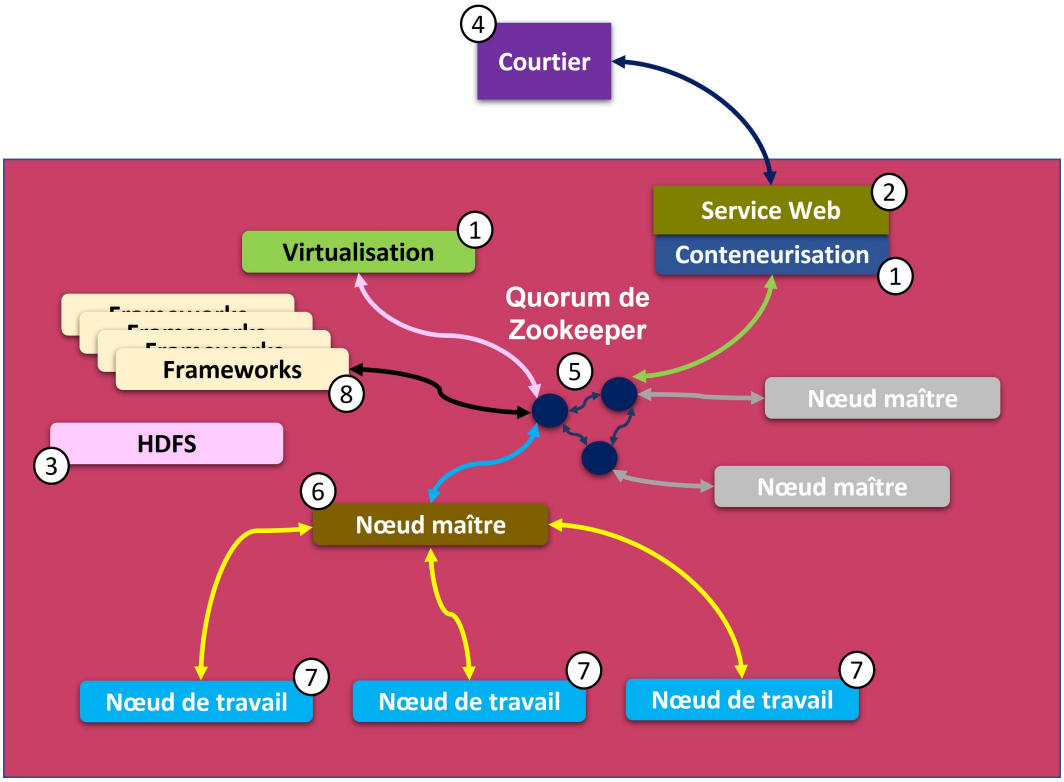


FIGURE 4.10 – Traduction sous forme d'architecture de la communication entre les deux architectures

La Figure 4.8 montre la traduction sous forme d'architecture de la Figure 4.7. La correspondance de la numérotation est respectée entre les deux figures.

Notre proposition architecturale se nomme LAMA (*Lambda Architecture Multi Activities*) et se base sur une architecture Lambda générique. Selon Díaz *et al.* [88] l'architecture Lambda est la plus souple et la mieux adaptée pour l'Internet des Objets (IoT) car il est nécessaire de pouvoir faire des requêtes sur les données en temps réel[4] ou quasi-temps réel mais également de pouvoir extraire des connaissances, des prévisions des données historiques. Les traitements qui doivent être effectués sur les données en temps réel et sur les données historiques sont parfois fondamentalement différents. En effet, Villari *et al.* [89] ont montré la capacité de gestion de l'architecture Lambda à grande échelle pour des environnements intelligents (Smart Environnent), de stockage et d'analyse des données massives (*Big Data*). Néanmoins, il est souvent reproché aux architectures Lambda d'être coûteuses car elles nécessitent le développement et la maintenance de deux branches de traitement distinctes.

4. En moins d'une seconde

Dans le cas des données scientifiques et plus particulièrement dans le cadre de la (re)valorisation des données historiques, il est nécessaire de pouvoir retraiter ces données pour notamment en adapter le format. Un traitement par lots (*Batch*) est donc nécessaire et est par essence fondamentalement différent de celui qui est effectué en temps réel. Cette contrainte justifie l'utilisation d'une architecture à deux branches de traitement. Notre architecture LAMA peut également fonctionner comme une architecture Kappa ou ne faire que des traitements par lots. Elle est modulable au niveau de ses composantes logicielles c'est-à-dire que ses briques logicielles peuvent être changées sans avoir à remanier l'architecture. Nous utilisons Apache Mesos pour gérer les ressources du cluster car il a été éprouvé sur des clusters jusque 10 000 nœuds, gère finement les ressources, permet d'exécuter d'anciennes applications et des applications cloud sur le même cluster.

L'architecture LAMA est spécifiquement développée pour le stockage de données expérimentales issues du Smart Farming et répondre aux contraintes particulières y afférent. Les spécificités reposent sur le fait que les données ne se déprécient pas avec le temps avec la même vitesse que dans d'autres applications de l'Internet des objets.

L'ensemble des données doit toujours pouvoir être réutilisables pour le développement de nouveaux modèles ou applications. Dans le futur, les programmes de recherche imposeront la mise à dispositions des données brutes et résultats expérimentaux issus des projets de recherche financés par les pouvoirs publics (Europe – projets H2020, SPW, ...) ce qui implique que l'architecture devra permettre l'accès à de grands volumes de données brutes et traitées sur des durées définies par le plan de gestion des données de chaque projet de recherche. Les temps de réponse aux requêtes effectuées sur les données stockées doivent être effectuées en quasi-temps réel pour être compatible avec la définition de temps réel des applications les plus exigeantes du Smart Farming. De plus, l'agrégation d'un nombre important de capteurs conduit à devoir gérer des débits de données importants au niveau de l'architecture cloud qui peuvent également fluctuer significativement dans des laps de temps courts. L'utilisation de mécanisme de file d'attente permet d'absorber les données lors des montées en charge brutale. D'un point de vue expérimental, les conditions expérimentales peuvent varier de manière importante entre les campagnes de mesures. Il est en effet courant qu'un chercheur change l'objet biologique sur lequel les expérimentations sont réalisées, de périphérique (*Device*) ou les senseurs attachés à ce périphérique ou encore la fréquence à laquelle les mesures sont effectuées. Ces variations de modalités expérimentales ont des implications sur l'adaptation des pipelines de traitement utilisés pour la transformation des données expérimentales avant leur stockage.

Dans les paragraphes qui suivent, nous allons présenter la pérennisation de l'architecture et l'exploitation des données. La pérennisation de l'architecture et des données qu'elle héberge est un aspect crucial dans les cas d'utilisation de recherches en Smart Farming que nous implémentons durant de cette thèse. La pérennisation couvre à la fois les aspects liés aux logiciels et leur cycle de vie mais aussi les aspects valorisation

de l'architecture post recherche. C'est ce dernier aspect, qui est développé dans le paragraphe qui suit. Le paragraphe suivant est consacré à la collecte des données et à la manière dont elle influe sur l'utilisation et l'utilisabilité de l'architecture cloud qui effectue les traitements centralisés.

Comme le montre la Figure 4.11, l'originalité de notre architecture cloud réside dans le couplage d'un cluster à haute performance (délimités par les pointillés rouges) à l'architecture Lambda (délimités par les pointillés bleus). Ces deux parties sont interconnectés par un webservice en Restful qui permet l'interopérabilité entre les deux composantes. Ce couplage consistant à partager le même système de fichiers, nous permet d'exploiter facilement les données stockées dans le système de stockage sans devoir les copier d'une infrastructure à l'autre ce qui procure un gain de temps considérable. Notre architecture se distingue d'une architecture Lambda classique par le fait de son auto-adaptabilité en fonction de la sémantique des données et par son association avec un cluster de calcul à haute performance. Cette association permet de traiter et d'héberger des applicatifs de natures variées. Dans les architectures classiques, tout changement dans la structure des données implique une modification des pipelines qui nécessitent une intervention humaine, des coûts de maintenance et un temps d'adaptation pendant lequel les nouvelles données ne peuvent être traitées. L'adaptabilité en fonction de la sémantique des données et la modularité à partir des composants logiciels sont les points forts de notre architecture. Un autre aspect original de l'architecture est sa modularité, c'est-à-dire la possibilité de changer les briques logicielles constitutives de l'architecture sans devoir revoir en profondeur le code. Finalement, la dernière originalité de notre architecture est l'utilisation des métadonnées associées aux données brutes provenant des capteurs pour adapter dynamiquement les chaînes de traitement des données et leur stockage.

La première partie (en pointillés bleu sur la Figure 4.11) est dédiée à la collecte, au traitement et au stockage des données issues de l'Internet des objets. Les données produites par les objets connectés qui utilisent généralement les radio fréquences pour communiquer, sont envoyées à une passerelle locale (*Local Gateway*) (**1**). Cette passerelle assure l'interopérabilité entre les protocoles de communication, centralise les données avant de les transmettre au bus de messages (**2**) c'est-à-dire une file d'attente distribuée ; afin d'y être stockées temporairement sous forme d'association clé-valeur avant leur traitement. La file de messages centralise les flux provenant de différentes sources en mode publication-souscription c'est-à-dire que les données sont stockées par sujet (topic) et les consommateurs de ces données s'abonnent aux sujets. Lorsque de nouvelles données sont disponibles, les abonnés sont automatiquement notifiés. La file de message doit être capable de supporter des débits très importants pour la publication et la lecture de données. Les messages qui y sont stockés peuvent être consommés en temps réel ou par lots par les pipelines de traitement (**3**). Un modèle unifié open source est utilisé pour définir les pipelines de traitement parallèles des données par lots (*Batchs*), par flux (*Streams*) et traiter différents types de sources de données (fichiers, base de données, file de messages). Les données traitées sont publiées sur un sujet (topic) de la

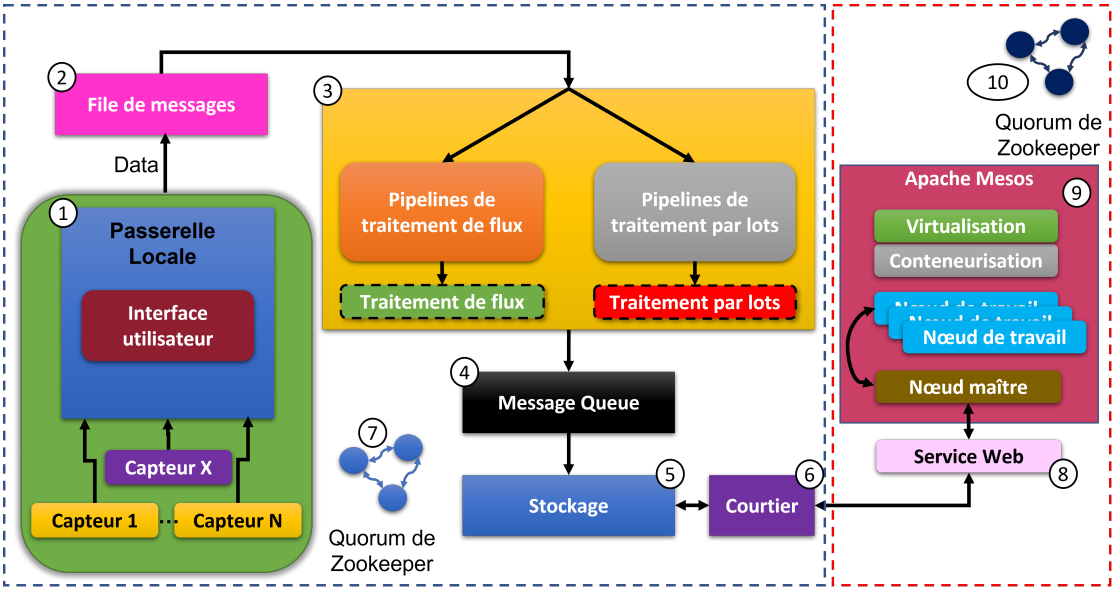


FIGURE 4.11 – Architecture cloud proposée

file de messages (**4**) et sont ensuite consommées pour être stockées en base de données (**5**). Une base de données distribuée open source stocke rapidement l'énorme quantité de données temporelles et répond aux requêtes avec un temps de latence faible. Un courtier (*broker*) reçoit les requêtes les distribues vers les nœuds de la base de données et agrège les résultats des requêtes avant de renvoyer le résultat consolidé (**6**). Un quorum de Zookeeper s'occupe de l'échange d'informations entre les différents éléments du cluster (**7**). Les liens entre le quorum de Zookeeper et les éléments constitutifs du cluster n'ont pas été représentés dans un soucis de clarté de la Figure 4.11).

La seconde partie (en pointillés rouge sur la Figure 4.11) est consacrée à l'hébergement des applications développées (**9**) par les chercheurs qui sont codées dans de multiples langages et utilisent des librairies spécifiques. De ce fait, notre infrastructure d'hébergement d'applications doit pouvoir héberger et utiliser les frameworks les plus courants (MPI [5], CUDA [6], Tensorflow [7], etc) tout en rationalisant les ressources disponibles. De plus, ces applications éclectiques sont élaborées pour fonctionner sur des systèmes d'exploitation différents. Ces prérequis impliquent de recourir concomitamment à la conteneurisation et à la virtualisation pour adresser ces exigences. De plus, le cluster doit également pouvoir gérer finement l'attribution des ressources pour optimiser l'utilisation des ressources. Les applicatifs peuvent interroger la base de données

5. Message Passing Interface (MPI) est une norme de transmission de messages standardisée et portable conçue pour fonctionner sur une grande variété d'architectures informatiques parallèles.

6. CUDA est une plateforme informatique parallèle et un modèle de programmation développé par Nvidia pour l'informatique générale sur ses propres GPU (unités de traitement graphique).

7. TensorFlow est une bibliothèque logicielle gratuite et open source utilisée pour l'apprentissage automatisé (*Machine Learning*).

de la partie collecte de données au travers d'une API en RESTful (**8**) qui transmet les requête au broker (**6**). Un quorum de Zookeeper s'occupe de l'échange d'informations entre les différents éléments du cluster applicatif (**10**). Les liens entre le quorum de Zookeeper et les éléments constitutifs du cluster n'ont pas été représentés dans un soucis de clarté de la Figure 4.11).

4.8 Conclusion

Dans ce chapitre nous avons décrit la conceptualisation d'une architecture cloud pour la recherche en Smart Farming basée sur les exigences de deux cas d'utilisations représentatif des exigences liées d'une part par la recherche et d'autre part par le domaine d'activité. Notre proposition architecturale s'appuie sur un framework qui nous permet de configurer notre architecture aussi bien en Lambda pour le traitement des flux temps réels et des lots de données ou en Kappa pour un traitement en temps réels des flux et un traitement par lot émulés sous forme de micro-lots de données.

L'architecture LAMA sera implémentée et les différents composants constitutifs seront testés individuellement dans le chapitre suivant. Finalement, une évaluation globale sera réalisée pour comparaison aux performances obtenues avec une architecture Kappa.

5

Expérimentation et validation d'une architecture cloud innovante

Plan du chapitre

Résumé

La conceptualisation de l'architecture LAMA a permis de mettre en exergue les rôles clés des principaux composants et les caractéristiques techniques attendues. L'étape suivante nous amène naturellement à son implémentation et plus spécifiquement aux choix de ses composants logiciels constitutifs. Ces composants logiciels identifiés comme candidats potentiels seront comparés entre eux du point de vue des performances. Les meilleurs seront ensuite sélectionnés et mis en œuvre pour constituer notre architecture LAMA (*Lambda Architecture Multi Activities*). Cette architecture a ensuite été étalonnée sur un ensemble de données de référence en vue d'en mesurer les performances. Elle est ensuite comparée aux performances d'une architecture du même type afin d'en démontrer l'efficience et finalement ses performances seront évaluées en conditions réelles en l'appliquant au cas d'utilisation des comportements des troupeaux de bovins.

Publications liées à ce chapitre

O. Debauche, S.A. Mahmoudi, N. De Cock, S. Mahmoudi, P. Manneback, F. Lebeau, "Cloud Architecture For Plant Phenotyping Research," *Concurrency and Computation : Practice and Experience*, 2020 : **32(17)**, e5661. doi : 10.1002/cpe.5661.

O. Debauche, S. Mahmoudi, P. Manneback, A. Assila, "Fog IoT for Health : A new Architecture for Patients and Elderly Monitoring," In : The 10th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2019). *Procedia Computer Science*, 2019, **160** : 289-297. doi : 10.1016/j.procs.2019.11.087.

J.B. Nkamla Penka, S. Mahmoudi, **O. Debauche**, "A new Kappa Architecture for IoT Data Management in Smart Farming". *Procedia Computer Science*, 2021 : **191** : 17-24. doi : 10.1016/j.procs.2021.07.006.

J.B. Nkamla Penka, S. Mahmoudi, A. Guttadauria, **O. Debauche**, "An Optimized Kappa Architecture for IoT Data Management in Smart Farming", *Journal of Ubiquitous Systems & Pervasive Networks*, 2022 : *17(2)* : 59-65. doi : 10.5383/JUSPN.17.02.002.

5.1 Introduction

L'architecture LAMA conceptualisée au chapitre 4 doit maintenant être mise en œuvre et testée. Nous débuterons nos évaluation par la sélection des briques logicielles les plus performantes. Nous allons évaluer les performances de certaines briques logicielles supportées par Apache Beam. Ce framework intercalaire apporte la généricité à l'architecture LAMA en permettant le changement aisé de logiciel d'exécution par recompilation. Le choix de la meilleure brique d'exécution est crucial pour obtenir une architecture rapide. Le second point important pour l'architecture est la vitesse d'ingestion et la vitesse de réponse aux requêtes. En effet, si l'on veut obtenir une réponse en quasi-temps réel, nous devons avoir outre la rapidité de traitement une base de données adaptée aux grands volumes de données et extensible (scalable). Il est donc important de tester les bases de données pour effectuer le meilleur choix. Nous allons ensuite évaluer l'influence des paramètres de Kafka sur la vitesse d'ingestion des données. Finalement, nous réaliserons une évaluation globale des performances de l'architecture LAMA en la comparant à une architecture Kappa afin de démontrer son efficacité.

5.2 Généricité de l'architecture et évaluation des performances des composants logiciels

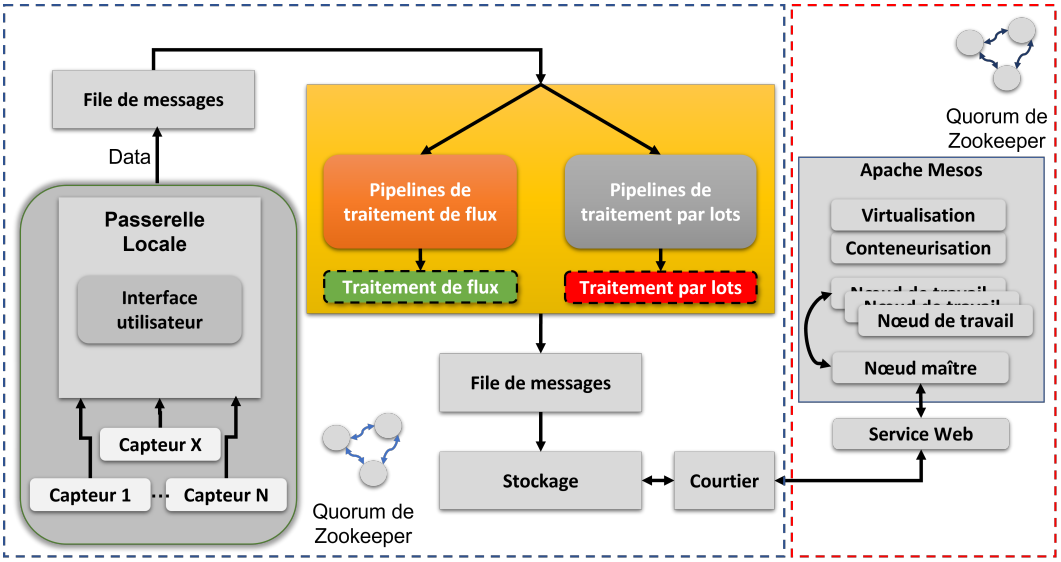


FIGURE 5.1 – Evaluation des briques logicielles de traitement

Apache Beam est un modèle unifié open source permettant de définir des pipelines de traitement parallèles des données par lots (*Batchs*) et par flux (*Streams*) et de traiter

différents types de sources de données. Trois types de sources de données peuvent être consommées par Apache Beam :

1. A base de fichiers : HDFS, Amazon S3, Google Cloud Storage, système de fichiers local. Les formats de fichiers supportés sont : Avro, Texte (CSV, TSV, ...), TFRecord, Xml, Tika, Parquet, RabbitMq, Sqs ;
2. Service de messagerie : Amazon Kinesis, AMQP, Apache Kafka, Google Cloud Pub/Sub, JMS, MQTT ;
3. Bases de données : Apache Cassandra, Apache Hadoop Input/Output Format, Apache HBase, Apache Hive, Apache Kudu, Apache Solr, Elastic Search, Google BigQuery, Google Cloud Bigtable, Google Cloud Datastore, Google Cloud Spanner, JDBC, MongoDB, Redis.

Apache Beam peut également sauvegarder le résultat du traitement du pipeline dans tous les formats mentionnés.

Le pipeline est défini dans un programme créé à l'aide de l'une des SDK Beam Open Source (Java, Python, Go). Le pipeline est ensuite exécuté par l'un des systèmes de traitement distribués pris en charge par Apache Beam à l'heure actuelle : Google Cloud Dataflow, Apache Apex, Apache Flink, Apache Gearpump, Apache Nemo, Apache Samza, Apache Spark et Hazelcast Jet. Le pipeline est composé d'un ensemble de transformations successives qui permettent d'adapter une collection initiale de données pour aboutir à une collection finale dont la structure peut différer de celle de départ (Voir Figure 5.2).

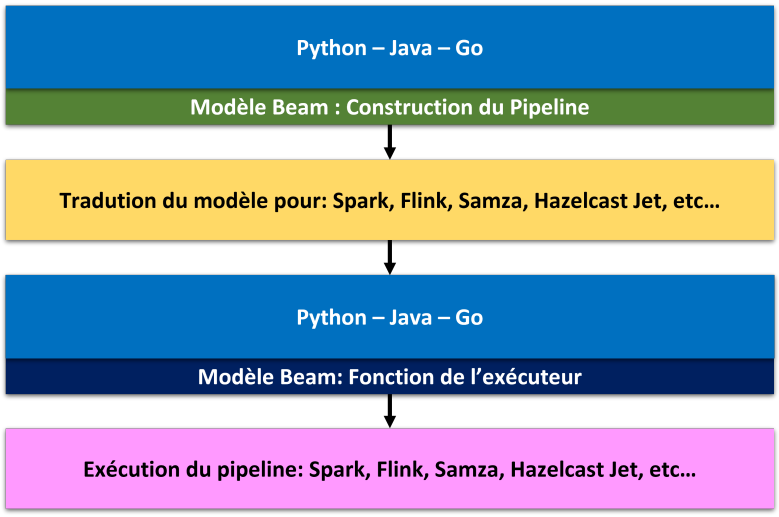


FIGURE 5.2 – Fonctionnement d'Apache Beam

Apache Beam est particulièrement bien adapté pour les tâches de traitement de données dite « parfaitement parallèles » c'est-à-dire dans lesquelles le problème peut

être décomposé en de nombreux ensembles de données plus petits pouvant être traités indépendamment et en parallèle. Apache Beam peut également être utilisé pour les tâches d'extraction, de transformation et de chargement (**E**xtraction **T**ransformation and, **L**oad) et l'intégration des données pure. Ces tâches sont utiles pour déplacer des données entre différents supports de stockage et sources de données, transformer ces données dans un format plus adapté ou les charger pour être compatible avec un nouveau système.

Apache Apex et Apache Flink se distinguent des autres briques logicielles par l'utilisation d'un modèle unifié de traitement de flux qui permet également le traitement par lots. L'ensemble des briques logicielles actuellement supportées par Beam permettent de traiter des gros débits de données avec une faible latence et sont pourvues de mécanismes de tolérances de pannes de type « exactly-one » à l'exception d'Apache Spark qui utilise les objets RDD [1] et les Dstreams [2] ; et d'Apache Samza qui utilise un système de points de contrôles incrémental (*Incremental Checkpoint*). Apache Gearpump et Apache Samza permettent le redéploiement à chaud c'est-à-dire de changer la configuration du cluster sans perte de données. Apache Apex, Apache Flink, Apache Samza, Apache Nemo et Apache Spark sont par ailleurs compatibles avec YARN [3] et l'écosystème Hadoop [4].

Nos tests ont été réalisés d'une part en comparant les performances de traitement de l'architecture Lama pour les principales briques logicielles (Flink, Hazelcast Jet, Samza et Spark). Les temps d'exécution totaux ont été mesurés et couvre toute la chaine de traitement de la compilation du pipeline jusqu'à l'exportation des données sur Apache Kafka.

Les tests ont été réalisés sur 2 fichiers représentant respectivement 1h de log à 100 Hz et 24h de log à 100Hz de 40 données différentes. L'algorithme de traitement utilisé est l'arbre décisionnel pour la détection des comportements de rumination et de pâturage proposé par Andriamandroso *et al.* [126].

Quatre chaines de traitement (pipelines) ont été codées en Java 1.8 avec le framework Beam pour traiter les fichiers csv d'entrées brutes et de les transformer pour en sortir une série chronologique de l'état de l'animal (timestamp, comportement) ou de coordonnées géographiques (latitude, longitude, comportement). Nous avons pris le parti de lire directement les fichiers csv pour ces tests pour nous abstraire de l'influence de la paramétrisation de Kafka sur la vitesse de traitement des briques logiciels et ainsi

1. RDD : Resilient Distributed Datasets est la structure de données fondamentale d'Apache Spark qui est une collection immuable d'objets distribués. Chaque ensemble de données dans RDD est divisés en partitions logiques qui peuvent être calculées sur différents nœuds du cluster.

2. Dstream : Discretized Stream est la structure de base d'Apache Spark Stream. Un Dstream est un flux continu d'objets RDD qui contiennent les données sur un intervalle de temps.

3. Yarn : Yet Another Resource Negotiator est le gestionnaire de ressources et de planification de tâches d'Apache Hadoop.

4. Apache Hadoop est un framework qui permet le traitement distribué de grands ensembles de données sur des clusters d'ordinateurs à l'aide de modèles de programmation simples.

éviter les résultats biaisés. L'optimisation des paramètres de Kafka fera l'objet du paragraphe suivant.

— Le **pipeline 1** traite le fichier log de 1h avec l'algorithme de détermination du comportement.
— Le **pipeline 2** traite le fichier log de 24h avec le même algorithme.
— Le **pipeline 3** traite le fichier log de 1h avec l'algorithme de détermination du comportement et la localisation.
— Finalement, le **pipeline 4** traite le fichier log de 24h avec le même algorithme que le pipeline 3.

Le modèle de traitement par lots (pipeline 1 à 4) a été conçu pour transformer les deux fichiers CSV en une série temporelle de catégories de comportement. Ce modèle réalisé avec Apache Beam transforme successivement les lignes du fichier en une classe objet contenant l'ensemble des valeurs des variables contenues dans une ligne du log. Une *map* est ensuite créée ou la clé est le timestamp et la valeur l'instance de la classe correspondante. Cette map est ensuite transformée en une *multimap* dans laquelle la clé est un temps arrondi en seconde et les clés sont les 100 instances de la classe objet contenues dans un intervalle de temps d'une seconde. Cette *multimap* permet ensuite de calculer les paramètres statistiques nécessaires à la détermination des comportements. La dernière étape consiste à appliquer l'arbre décisionnel décrit dans Andriamandroso *et al.* [126] qui utilise les valeurs des paramètres statistiques calculés à l'étape précédente et à produire soit une série temporelle de comportement (pipeline 1 et 2) ou un des données géographiques mettant en relation la position avec le comportement (pipeline 3 et 4). La Figure 5.3 illustre les différentes étapes suivies pour la production des données des pipelines 1 et 3. Les pipeline 2 et 4 suivent un modèle identique à la différence que le temps (time) est remplacé dans les étapes 3 à 5 par la position.

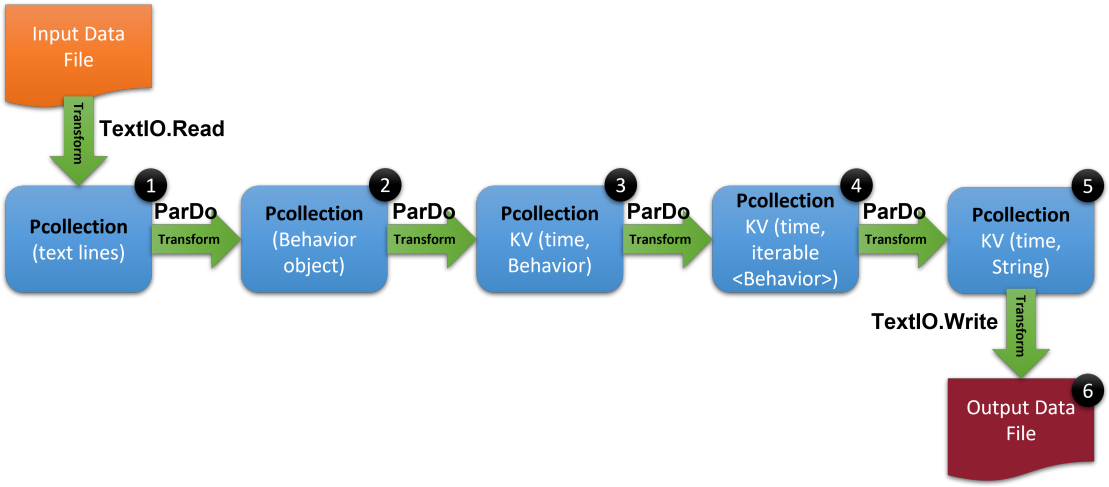
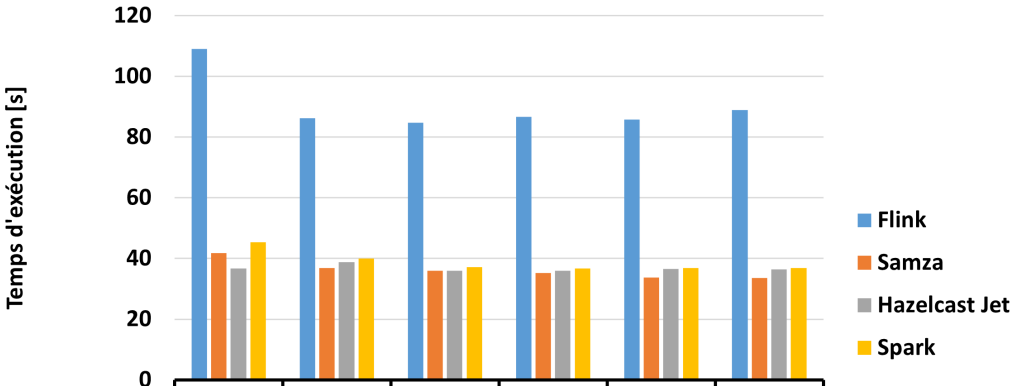


FIGURE 5.3 – Diagramme de flux du traitement par lots

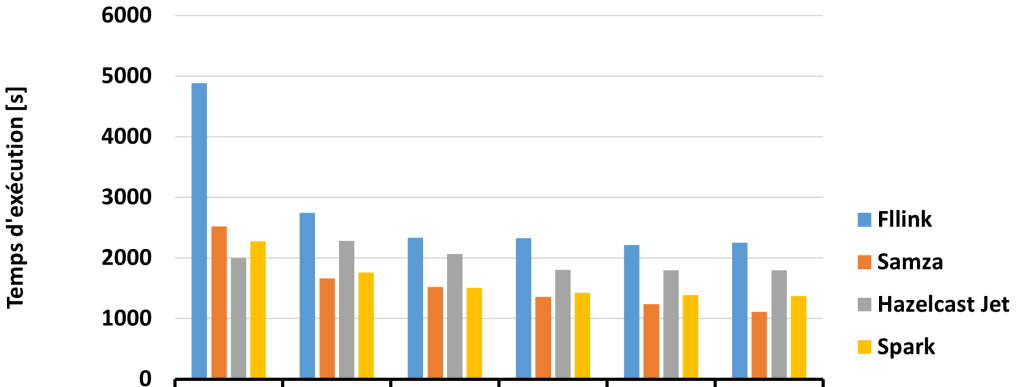
Les temps d'exécution pour les 4 pipelines sur (Flink, Hazelcast Jet, Samza et Spark) sont repris dans les Figures 5.4 à 5.7.

L'analyse des résultats obtenus pour les pipelines 1 et 2 qui correspondent respectivement à l'analyse comportementale appliquée aux fichiers de log de 1h et 24h montent qu'Apache Flink systématiquement moins performante par rapport à Samza, Hazelcast Jet et Spark. Hazelcast Jet est le plus performant quand on effectue le traitement sur 1 core par contre dès que le nombre de core augmente Samza obtient de meilleur performances ce qui s'explique par le fait que Samza est optimisé pour la parallélisation. Les résultats pour le traitement du fichier d'une 1h sont assez similaires pour Samza, Hazelcast Jet et Spark où il a peu de données à traiter par contre les différences se marquent davantage sur un traitement plus long. Par contre les temps d'exécution ne diminuent pas significativement avec l'évolution du nombre de cœurs ; cela peut s'expliquer par le fait que nous utilisons des VPS ou la disponibilité de la mémoire est garantie mais pas celle des vCore. Pour compenser le problème disponibilité des vCore, nous avons répété chacune des expériences 100 fois afin d'obtenir des temps d'exécution moyens.



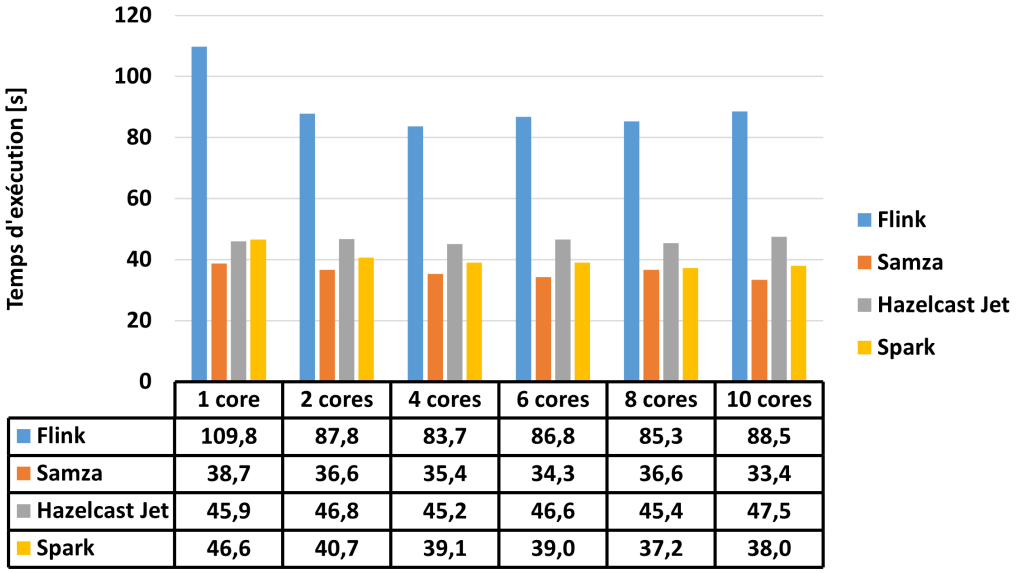
	1 core	2 cores	4 cores	6 cores	8 cores	10 cores
Flink	109,1	86,2	84,7	86,7	85,8	88,9
Samza	41,8	36,9	36,0	35,2	33,8	33,5
Hazelcast Jet	36,7	38,8	36,0	36,0	36,6	36,4
Spark	45,3	39,9	37,2	36,8	36,8	36,9

FIGURE 5.4 – Temps d'exécution de l'analyse du comportement sur le fichier de log de 1h



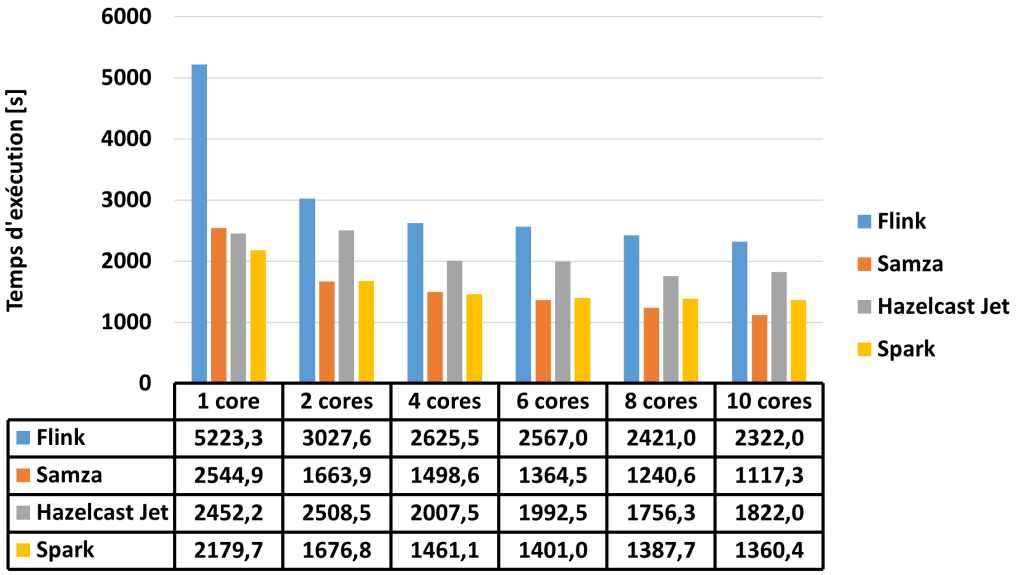
	1 core	2 cores	4 cores	6 cores	8 cores	10 cores
Fllink	4884,6	2741,6	2331,9	2326,9	2216,4	2248,4
Samza	2523,6	1658,9	1520,8	1357,2	1235,9	1113,9
Hazelcast Jet	1998,3	2279,6	2067,8	1804,8	1797,3	1795,3
Spark	2271,1	1757,7	1503,8	1423,5	1384,2	1369,6

FIGURE 5.5 – Temps d'exécution de l'analyse du comportement sur le fichier de log de 24h



	1 core	2 cores	4 cores	6 cores	8 cores	10 cores
Flink	109,8	87,8	83,7	86,8	85,3	88,5
Samza	38,7	36,6	35,4	34,3	36,6	33,4
Hazelcast Jet	45,9	46,8	45,2	46,6	45,4	47,5
Spark	46,6	40,7	39,1	39,0	37,2	38,0

FIGURE 5.6 – Temps d'exécution pour la production des données SIG à partir du fichier de log de 1h



	1 core	2 cores	4 cores	6 cores	8 cores	10 cores
Flink	5223,3	3027,6	2625,5	2567,0	2421,0	2322,0
Samza	2544,9	1663,9	1498,6	1364,5	1240,6	1117,3
Hazelcast Jet	2452,2	2508,5	2007,5	1992,5	1756,3	1822,0
Spark	2179,7	1676,8	1461,1	1401,0	1387,7	1360,4

FIGURE 5.7 – Temps d'exécution pour la production des données SIG à partir du fichier de log de 24h

L'analyse des résultats obtenus pour les pipelines 3 et 4 qui correspondent au traitement du fichier de 1h et de 24h pour la production de données cartographiques (posi-

tion, comportement) montre que les meilleurs performances sont également obtenues par Samza à l'exception du traitement du fichier 24h avec un 1 cœur. Samza est la brique logicielle de traitement performante pour ce cas d'application suivie de près par Spark qui pourrait être une alternative au cas où Samza devrait être remplacée.

En conclusion, Apache Samza a donné les meilleurs résultats et se distingue de ses concurrents par :

— Ses capacités de déploiement à très grande échelle ;
— Sa capacité à traité des flux de données à haute fréquence ;
— Son système de tolérance de pannes incrémental ;
— Ses capacités de redéploiement à grande échelle avec un temps d'indisponibilité très court ;
— La possibilité de mettre à jour les traitements sans perte de données ;
— Les possibilités de retour en arrière (*roll back*).

Dans notre cas d'utilisation lié au comportement animaux, les données sont transmises à des fréquences allant jusque 100 Hz. Par conséquent, Apache Samza est la brique logicielle la mieux adaptée pour ce cas d'application. De plus ses capacités, de mises à jour à chaud sans pertes de données et la possibilité de revenir en arrière sont bien adaptées pour l'application que nous développons.

5.3 Evaluation des bases de données

Nous allons maintenant évaluer les performances de différentes bases de données et choisir la base de données la plus appropriée pour notre architecture. Voir Figure 5.8.

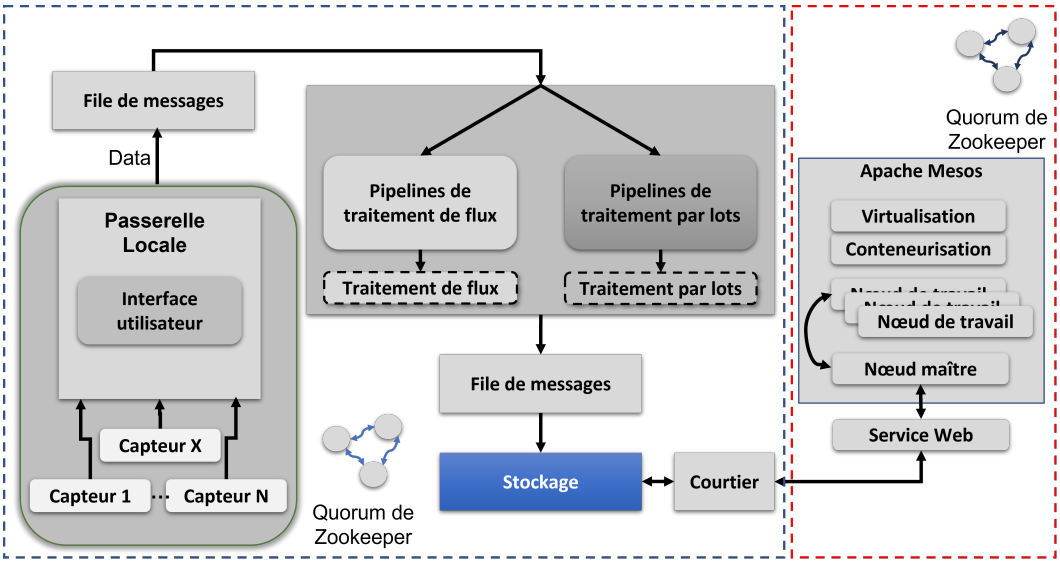


FIGURE 5.8 – Evaluation des bases de données

La partie stockage de notre architecture cloud nécessite une base de données. Afin de réaliser ce choix, nous avons éprouvé les principales bases de données Open source afin de déterminer laquelle est la plus adaptée. Rappelons-nous que les cas d'utilisations en Smart Farming exigent bien souvent des temps d'interrogation et de réponse de la base de données de quelques secondes au plus. Le critère temps de réponse est par conséquent critique dans type d'applications. Nos tests ont été réalisées en se basant sur le critère de temps de réponse, sur base de requêtes typiques (comptage, agrégation et filtrage), largement utilisées dans nos cas d'applications en Smart Farming. Ces requêtes ont été effectuées sur un ensemble de données standardisées issues d'un Benchmark de référence. Nos résultats sont ensuite comparés à ceux obtenus sur MySQL une base de données open source largement utilisée.

Les systèmes de gestion de bases de données testées sont :

1. MySQL 5.7.24 (version de la distribution Ubuntu 18.04 LTS) ;
2. MySQL 8.0.14 avec les moteurs MyISAM (ancien moteur) et InnoDB (moteur par défaut) ;
3. PostgreSQL 10.6 (version fournie avec la distribution Ubuntu 18.04 LTS) ;
4. MongoDB 3.6.3 (version fournie avec la distribution Ubuntu 18.04 LTS) ;
5. Cassandra 3.11.4 (dernière version au moment des tests) ;
6. Druid 0.14-incubating (dernière version au moment des tests).

La version 5.7.24 de MySQL a été évaluée pour permettre la comparaison avec les essais effectués en 2014 par Xavier Léauté[148] avec une version antérieure (Druid 0.6.62) et MySQL 5.6. La dernière version de MySQL 8 a également été évaluée car elle constitue une avancée majeure par rapport à l'ancienne branche 5 avec des performances améliorées jusqu'à 2,5x selon Oracle. Elle apporte également le support du NoSQL via le support JSON, et le basculement du moteur d'exécution MyISAM vers le moteur InnoDB.

PostgreSQL 10.6 est un système de gestion de bases de données orienté objet SGDBRO pourvu de capacités NoSQL.

MongoDB 3.6.3 est un système de gestion de base de données NoSQL orienté document utilisant BJSON . Ce type de base de données permet l'ajout, la suppression, la modification des champs ou encore de les renommer à tout moment car il n'y a pas de structure prédéfinie.

Cassandra 3.11.4 est également un système de gestion de base de données orienté document qui utilise le CQL, un langage de requête proche de SQL utilisé pour l'élaboration des requêtes.

Finalement, Apache Druid est un projet qui a rejoint la fondation Apache. Il connait depuis lors un développement rapide grâce une communauté active. Apache Druid

a été élaboré pour combiner les principes des bases de données temporelles, les systèmes de recherche et les traitements analytiques en ligne (OLAP) sur plusieurs axes. Apache Druid stock et compresse chaque colonne individuellement et est optimisé pour prendre en charge les analyses rapides, les classements et les regroupements (GROUP BY).

Nous avons évalué le temps de réponse de la demande sur l'infrastructure avec le benchmark TPC-H [149] qui évalue les performances de l'architecture de deux manières : (**1**) Le Power Test mesure la puissance d'exécution des requêtes de l'architecture lorsque connecté avec un seul utilisateur. Les demandes sont exécutées une fois à la fois et le temps écoulé est mesuré séparément pour chaque requête ; (**2**) Le test de débit mesure la capacité de l'architecture à traiter des requêtes simultanées dans un environnement multi-utilisateurs. Ce benchmark a été sélectionné pour permettre une comparaison avec le benchmark de Léauté en 2014 [148] et est généralement utilisé pour évaluer les performances d'une architecture Lambda [150]. Les tests de performances ont été réalisés sur des VPS XL SSD loués chez Contabo (`http://contabo.com`). Ces VPS équipés de Xeon E5-2630v4 (10 cœurs), 60 Gb de Ram garantie, 1600 GB d'espace disque en SSD en NVMe, connexion réseau 1000 Mbit/s sous Ubuntu 18.04 LTS, Kernel 4.15. La base de données 100 Gb correspond à une taille classique obtenue à partir d'une expérimentation de phénotypage. Nous avons comparé les temps de réponse à 3 types de requêtes classiques (nombre, somme et top) sur MySQL 5.7.24 (MyISAM), MySQL 8.0.14 (MyISAM), MySQL 8.0.14 (InnoDB), Druid 0.14-incubating-iap8 avec TPC-H bases de données de 1, 10 et 100 Go installées sur un seul nœud. Les trois tailles de base de données représentent respectivement 6 001 215 lignes, 60 003 588 lignes et 600 037 092 lignes. La table *lineitem* utilisée pour l'expérimentation est composée de données d'événements quotidiens couvrant plusieurs années et d'un ensemble varié de dimensions et de mesures, y compris à la fois une cardinalité très élevée et des dimensions de cardinalité faible telles que le champ *l_partkey* avec 20 272 236 uniques valeurs et champ *l_commitdate* avec 2466 dates distinctes dans l'ensemble de données de 100 Go.

```
-- count_star_interval
SELECT COUNT(*) FROM LINEITEM WHERE L_SHIPDATE BETWEEN
  '1992-01-03' AND '1998-11-30';

-- sum_price
SELECT SUM(L_EXTENDEDPRICE) FROM LINEITEM;

-- sum_all
SELECT SUM(L_EXTENDEDPRICE), SUM(L_DISCOUNT), SUM(L_TAX),
  SUM(L_QUANTITY) FROM LINEITEM;

-- sum_all_year
SELECT YEAR(L_SHIPDATE), SUM(L_EXTENDEDPRICE), SUM(L_DISCOUNT),
```

```
  SUM(L_TAX), SUM(L_QUANTITY)
  FROM LINEITEM GROUP BY YEAR(L_SHIPDATE);

-- sum_all_filter
SELECT SUM(L_EXTENDEDPRICE), SUM(L_DISCOUNT), SUM(L_TAX),
  SUM(L_QUANTITY)
  FROM LINEITEM WHERE L_SHIPMODE LIKE '%AIR%';

-- top_100_parts
SELECT L_PARTKEY, SUM(L_QUANTITY) FROM LINEITEM
  GROUP BY L_PARTKEY ORDER BY SUM(L_QUANTITY) DESC LIMIT 100;

-- top_100_parts_details
SELECT L_PARTKEY, SUM(L_QUANTITY), SUM(L_EXTENDEDPRICE),
  MIN(L_DISCOUNT), MAX(L_DISCOUNT)
  FROM LINEITEM GROUP BY L_PARTKEY ORDER BY SUM(L_QUANTITY)
        DESC LIMIT 100;

-- top_100_parts_filter
SELECT L_PARTKEY, SUM(L_QUANTITY), SUM(L_EXTENDEDPRICE),
  MIN(L_DISCOUNT), MAX(L_DISCOUNT)
  FROM LINEITEM WHERE L_SHIPDATE BETWEEN '1996-01-15'
        AND '1998-03-15'
  GROUP BY L_PARTKEY ORDER BY SUM(L_QUANTITY) DESC LIMIT 100;

-- top_100_commitdate
SELECT L_COMMITDATE, SUM(L_QUANTITY) FROM LINEITEM
GROUP BY L_COMMITDATE ORDER BY SUM(L_QUANTITY) DESC LIMIT 100;
```

Listing 5.1 – Request used for experimentation - copied from Léauté in 2014[148]

Chacune des requêtes a été effectuée 100 fois. Les temps moyens et d'écart type obtenus sont exprimés en secondes sur la Figure 5.9. Les temps de réponse pour les requêtes *Count* et *Sum* sont illustrés à gauche sur la Figure 5.9, respectivement pour les bases de données de 1, 10 et 100 Go. Cependant, les temps de réponse pour les requêtes *topN* sont isolés à droite de la Figure. 5.9 pour la même taille de base de données pour des raisons de temps d'échelle très différentes. MySQL 5.7.24 a été choisi pour permettre une comparaison avec le précédent benchmark effectué par exemple par Léauté en 2014 [148]. De plus, MySQL 8.0.14 a été testé avec le moteur MyISAM pour être comparable à MySQL 5.7.24 et à InnoDB, le moteur par défaut de cette version. La version Druid 0.14-incubating-iap8 était la dernière version publiée.

L'analyse montre que MySQL 5.7.24 se comporte mieux que MySQL 8.0.14 sur une petite base de données (1 Go) sur tous les types de requêtes. MySQL 8.0.14 avec le

moteur InnoDB est plus efficace sur les opérations d'agrégation sur une taille de base de données de 1 à 100 Go que MySQL 8.0.14 MyISAM. MySQL 8.0.14 avec le moteur MyISAM est plus efficace que le moteur InnoDB sur les requêtes topN. Enfin, l'incubation de Druid 0.14 est incontestablement plus efficace que n'importe quelle version de MySQL avec un rapport jusqu'à 180 fois meilleur. De plus, le temps de réponse à la demande évolue linéairement avec la taille de la base de données. Les requêtes TopN avec MySQL 8.0.14 prennent plus de 250 000 secondes. Elles ne sont pas représentées sur la Fig. 5.9.

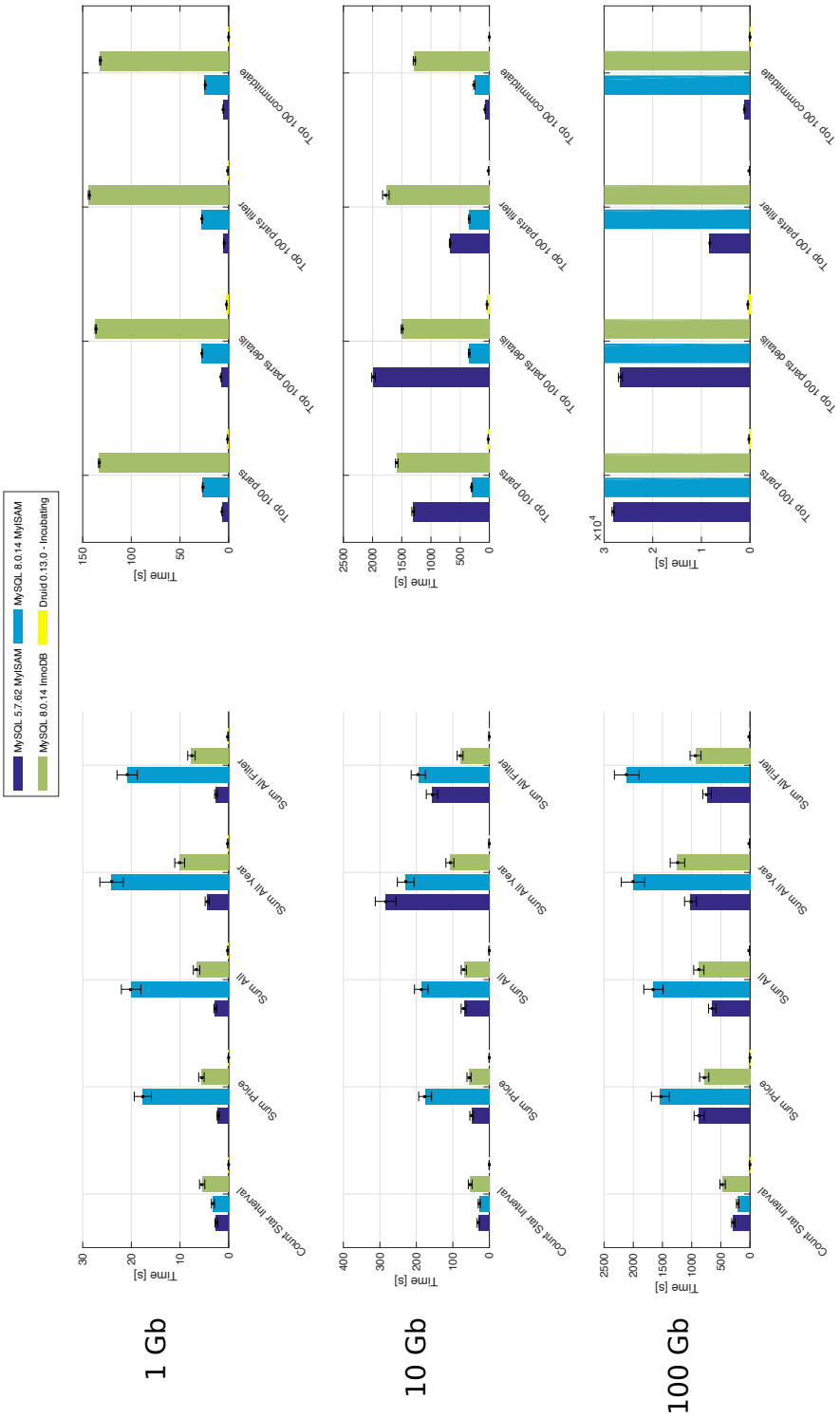


FIGURE 5.9 – Temps de réponse aux requêtes - BDD TPCH - 1 à 100 GB

La mise à l'échelle de Druid sur la base de données TPC-H de 100Gb a été étudiée respectivement sur 1 nœud (10 cœurs), 3 nœuds (30 cœurs) et 6 nœuds (60 cœurs). Une diminution presque proportionnelle au nombre de cœurs est observée sur le nombre de requêtes et les requêtes d'agrégation (Figure 5.10). Une autre diminution entre 2 et 3 est obtenue pour 3 nœuds et entre 4 et 5 pour 6 nœuds sur les requêtes supérieures (Figure 5.11). Cette croissance plus faible a également été observée par Léauté en 2014[148]. Cette baisse importante observée peut s'expliquer d'une part par l'importance des fusions et d'autre part par la faible cardinalité des données. Apache Druid utilise un stockage orienté colonnes, il fonctionne mieux avec des requêtes utilisant moins de colonnes. Cependant, pour les requêtes avec un plus grand nombre de colonnes, les avantages des moteurs de stockage orientés lignes deviennent moins importants.

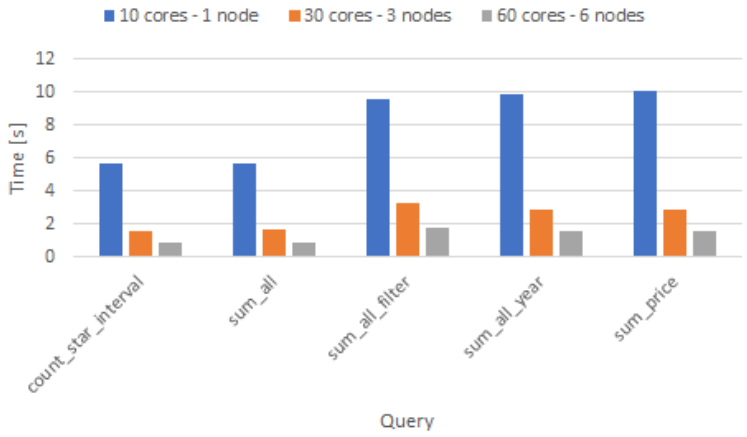


FIGURE 5.10 – Mise à l'échelle de Druid - DB TPC-H 100GB - partie 1

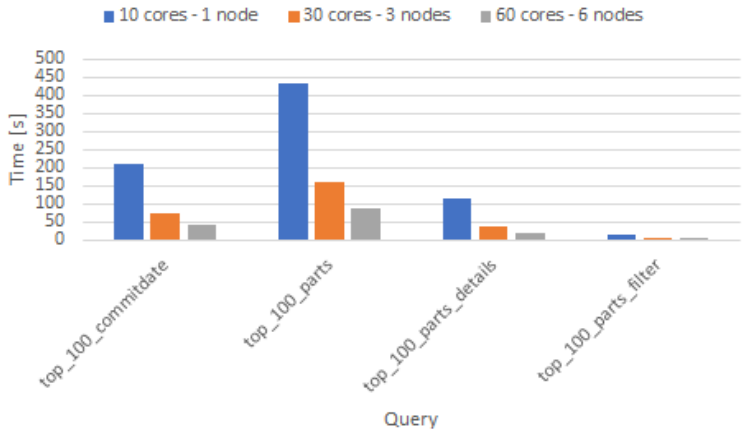


FIGURE 5.11 – Mise à l'échelle de Druid - DB TPC-H 100GB - partie 2

L'analyse des résultats montre qu'Apache Druid est plus performantes que MySQL 5.7 et 8.0, PostgreSQL 10.6, MongoDB 3.6.3, et Cassandra 3.11.4. Le test de mise à l'échelle d'Apache Druid montre que les performances se maintiennent avec l'augmentation du nombre de cœur. Néanmoins, le teste de mise à l'échelle n'a été réalisé que sur nombre restreint de nœuds et donne, à ce stade, un résultat indicatif.

5.4 Évaluation de l'influence des paramètres de Kafka sur l'ingestion des données

Nous allons nous intéresser au paramétrage Kafka et son influence sur la vitesse d'ingestion des données. Voir Figure 5.12.

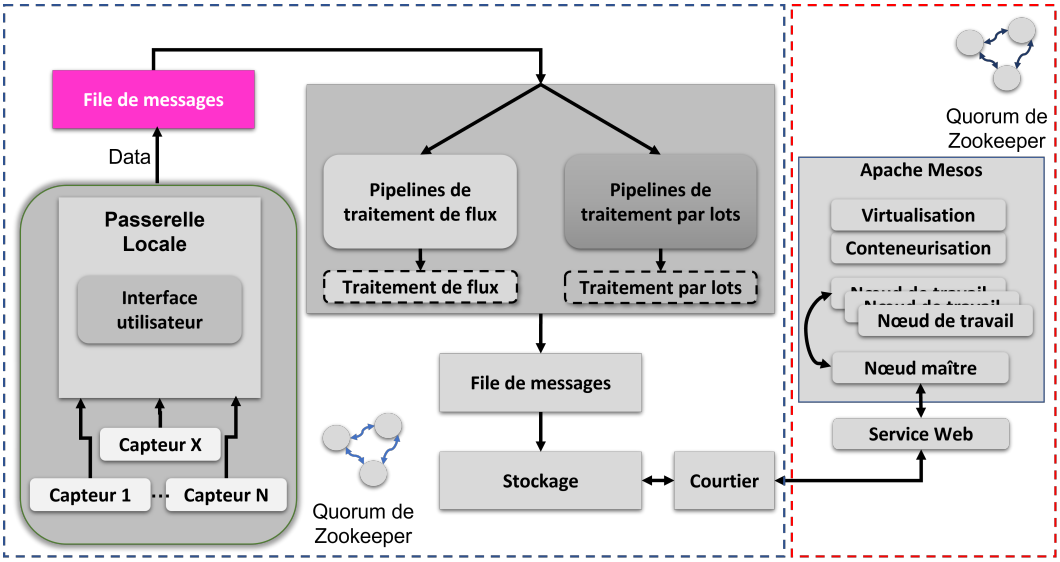


FIGURE 5.12 – Paramétrisation de Kafka

Les analyses relatives à l'influence des paramètres de Kafka sur l'ingestion des données ont été réalisés dans le cadre du mémoire de fin d'études de M. Jean Bertin NKAMLA PENKA supervisé par nos soins. Les résultats obtenus ont par ailleurs fait l'objet de deux publications. Pour la réalisation de nos tests, nous avons opté pour l'utilisation d'une architecture Kappa pour d'une part confirmer les performances de l'association Samza - Druid d'une part et d'avoir un point de comparaison pour l'évaluation globale des performances de l'architecture d'autre part.

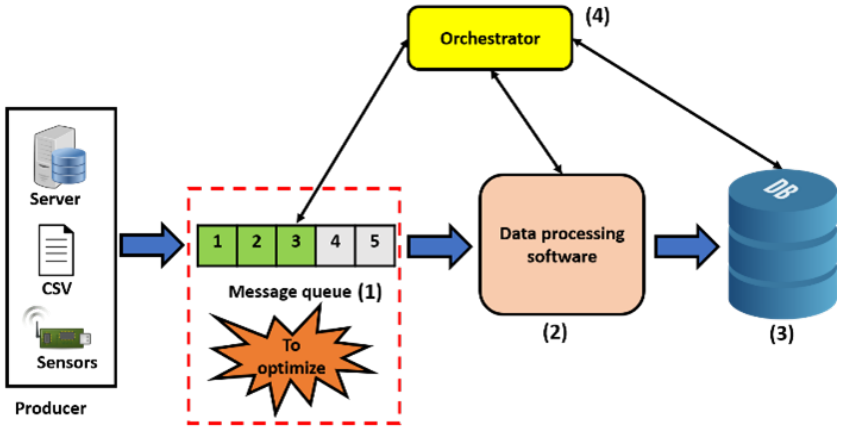


FIGURE 5.13 – Schéma général d'une architecture Kappa

La Figure 5.13, montre les 4 principaux composants d'une architecture Kappa. (**1**) une file de message qui stocke temporairement les données avant leur traitement par la brique de traitement de données (**2**) qui produit un résultat qui est stocké dans une base de données (**3**). Un orchestrateur (**4**) assure la coordination des opérations effectuées par les différents composants logiciels et surveille également leur état.

Nos expérimentations ont porté sur la combinaison de deux briques logicielles de traitement des données avec deux bases de données. La Figure 5.14 illustre les 4 combinaisons évaluées : (**1A**) Apache Storm - Apache HBase ; (**1B**) Apache Storm - Apache Druid ; (**2A**) Apache Samza - Apache HBase et (**2B**) Apache Samza - Apache Druid. Il est à noter que les combinaisons (**1B**) et (**2B**) utilisant la base de données Druid recourt à une file d'attente Kafka pour stocker les données à ingérer. Le service "Druid-Kafka-Indexing" ingère les données stockées temporairement dans Kafka au sein d'un topic et les insère dans la base de données Druid. Nos expérimentations ont pour objectif de montrer l'impact de la mémoire et de la période de décalage de commit (offset commit period).

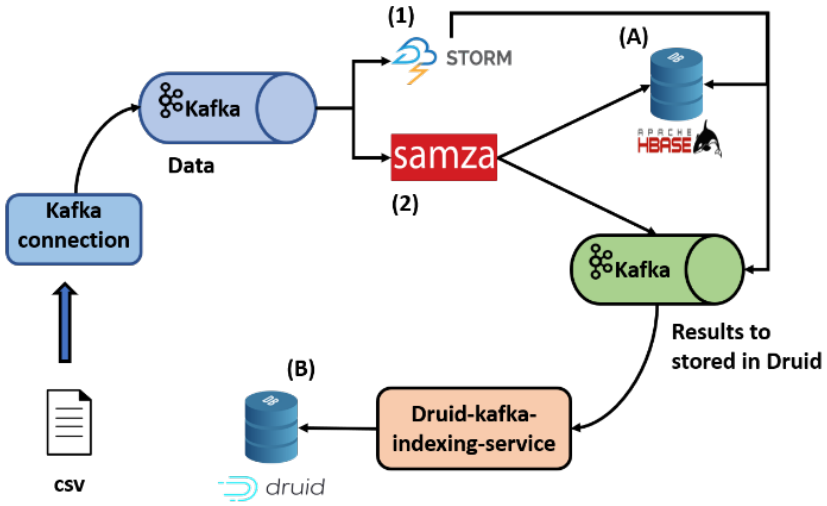
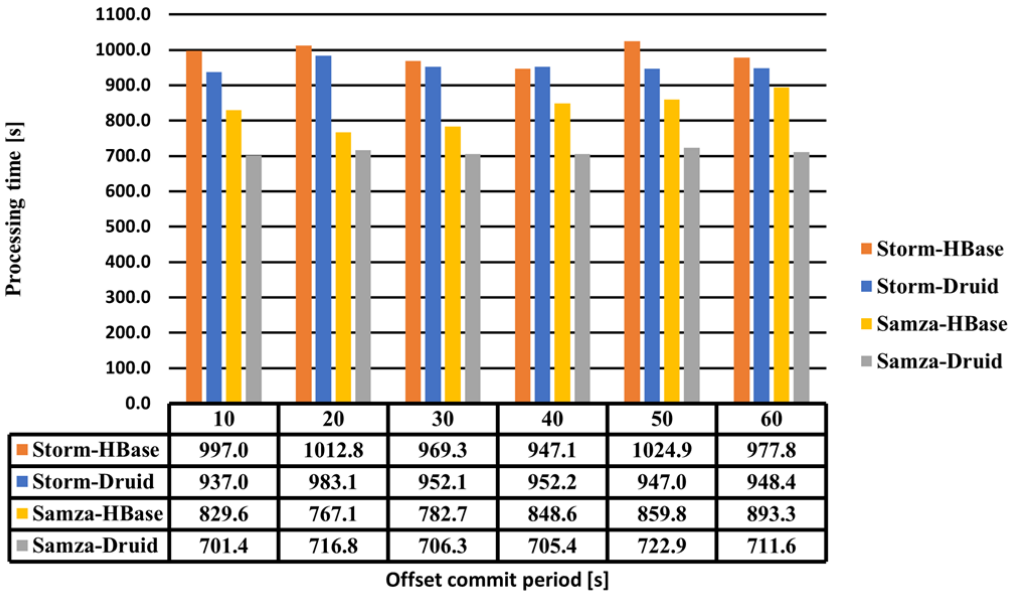


FIGURE 5.14 – Combinaisons de briques de traitement - base de données testées

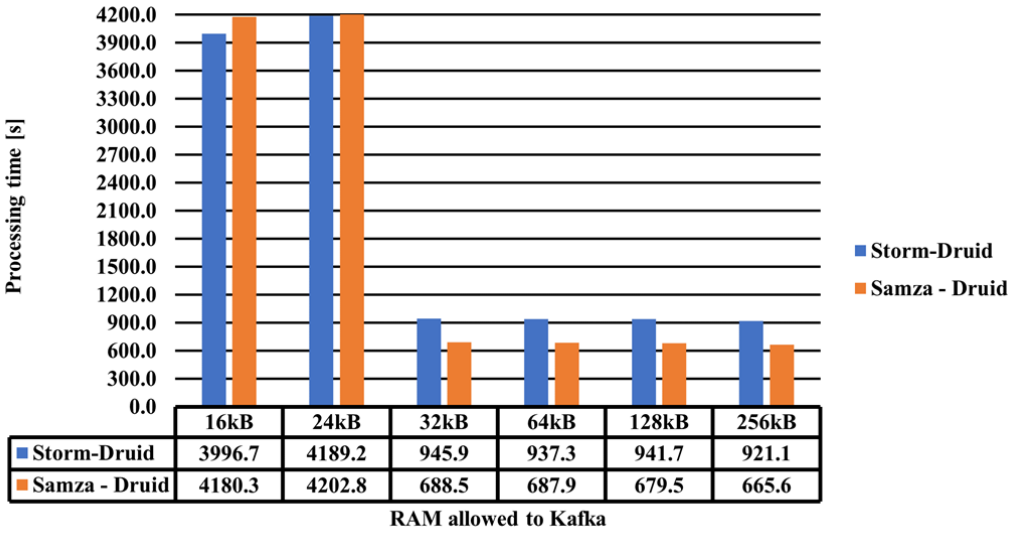
Apache Kafka est une plateforme open source de diffusion d'événements distribués qui est principalement utilisée pour le stockage temporaire des journaux. Apache Storm et Apache Samza sont deux systèmes open source de calcul distribué en temps réel. Apache Storm est également évolutif et tolérant aux pannes, ce qui garantit que les données seront traitées en cas d'incident. Une topologie Apache Storm ingère des flux de données et traite ces flux de manière arbitrairement complexe. Apache Samza est un moteur de traitement de données évolutif qui permet de traiter et d'analyser des données en temps réel. Apache HBase est une base de données non relationnelle open source, distribuée, versionnée, construite pour fournir un accès aléatoire en lecture/écriture en temps réel aux Big Data avec de grandes tables composées de milliards de lignes et de millions de colonnes. Apache Druid est un magasin de données distribué open source doté de capacités d'analyse en temps réel très performantes et combinant les concepts d'entrepôts de données, de bases de données de séries chronologiques et de systèmes de recherche. Nous avons choisi Apache Storm car il s'agit d'un outil bien connu et utilisé pour le traitement des données, tandis qu'Apache Samza est un logiciel prometteur de traitement des flux.

Les expérimentations ont été réalisées sur les données produites par l'IMU 9-DOF et l'affichage GPS enregistré au moyen d'un iPhone 5s placé au-dessus du cou d'une vache. L'iPhone 5s grâce au Data Sensor v1.26 permettant de collecter 41 paramètres à une fréquence allant jusqu'à 100Hz durant 24h.



	10	20	30	40	50	60
Storm-HBase	997.0	1012.8	969.3	947.1	1024.9	977.8
Storm-Druid	937.0	983.1	952.1	952.2	947.0	948.4
Samza-HBase	829.6	767.1	782.7	848.6	859.8	893.3
Samza-Druid	701.4	716.8	706.3	705.4	722.9	711.6

FIGURE 5.15 – Impact du temps de décalage entre les commit sur les temps de traitement



	16kB	24kB	32kB	64kB	128kB	256kB
Storm-Druid	3996.7	4189.2	945.9	937.3	941.7	921.1
Samza - Druid	4180.3	4202.8	688.5	687.9	679.5	665.6

FIGURE 5.16 – Influence de la quantité de mémoire allouée à Kafka sur le temps de traitement

La Figure 5.15 montre l'impact de la période de décalage entre les commit sur les temps de traitement exprimé en secondes c'est-à-dire le temps écoulé entre la lecture des données du fichier csv leur traitement par Storm ou Samza et le stockage en base de données. Les meilleurs performances sont obtenues indépendamment de la durée de décalage entre les commit par l'association de Samza avec la base de données Druid. Par ailleurs, Samza obtient de meilleurs résultats que Storm peu importe la base données associée.

La Figure 5.16 présente l'impact de la taille de la mémoire allouée à Kafka sur le temps de traitement moyen calculé sur dix répétitions. On observe une évolution significative du temps de traitement quand on fait passer la taille de la mémoire de sa valeur par défaut : 8Ko à 32Ko.

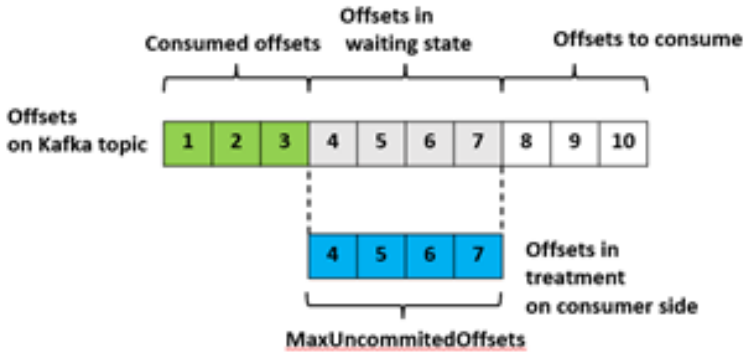
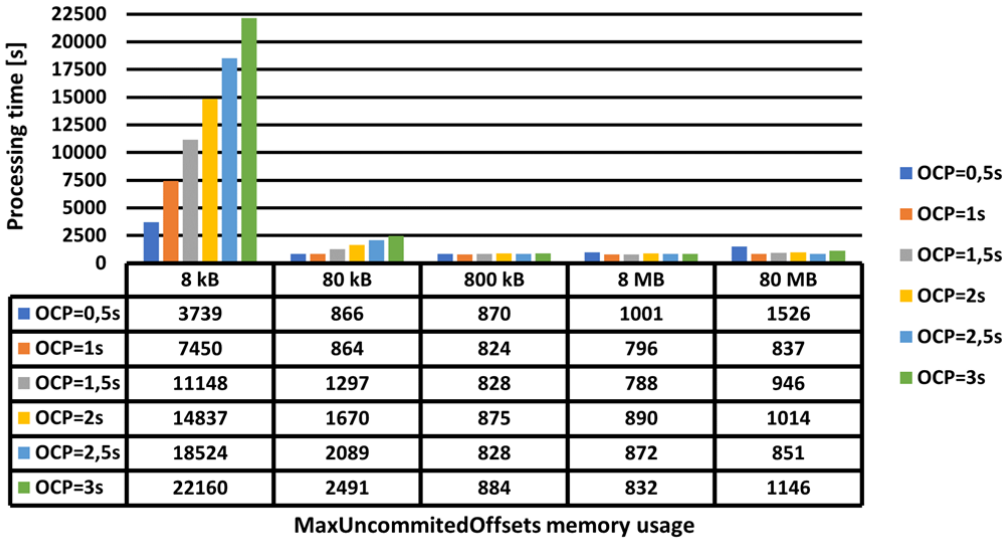


FIGURE 5.17

De plus, Apache Storm implémente un paramètre particulier appelé *MaxUncommitedOffsets* qui définit la période maximale pendant laquelle les données peuvent être stockées temporairement entre ces traitements. En d'autres termes, des micro-lots sont simulés, et leur taille est intrinsèquement contrôlée par le paramètre *MaxUncommitedOffsets* qui permet de régulariser la taille des ensembles de données à traiter.

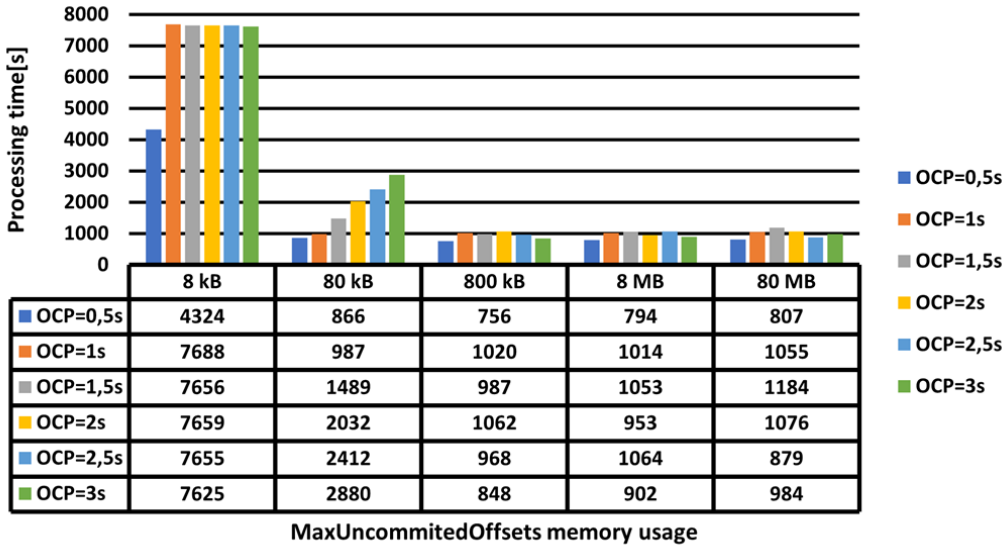
La Figure 5.17 est une représentation simple du mécanisme de traitement et de mise à jour des offsets sur les sujets Kafka par un consommateur. Sur cette figure, les offsets sur les sujets Kafka sont les numéros 1 à 10. Les numéros de 1 à 3 en vert sont déjà consommés. Les numéros de 4 à 7 en gris sont en état d'attente car ils sont en traitement du côté du consommateur, où les mêmes numéros sont représentés en bleu. Lors de l'utilisation d'Apache Storm pour le traitement des données, les offsets en bleu sont également connus sous le nom de *UncommitedOffsets* car il incombe au consommateur d'engager un offset tel que consommé sur le topic Kafka. Ce mécanisme permet au sujet Kafka d'avoir plusieurs consommateurs en même temps. Apache Storm a défini un paramètre interne appelé *MaxUncommitedOffsets* qui est le maximum de *UncommitedOffsets* qu'un consommateur peut engager sur le sujet Kafka. Le commit se produit pendant une période de temps appelée *OffsetCommitPeriodMs* (OCP) qui est également

un paramètre interne défini par Apache Storm pour le traitement des données d'un sujet Apache Kafka. Les offsets numéro 8 à 10 en blanc seront consommés plus tard.



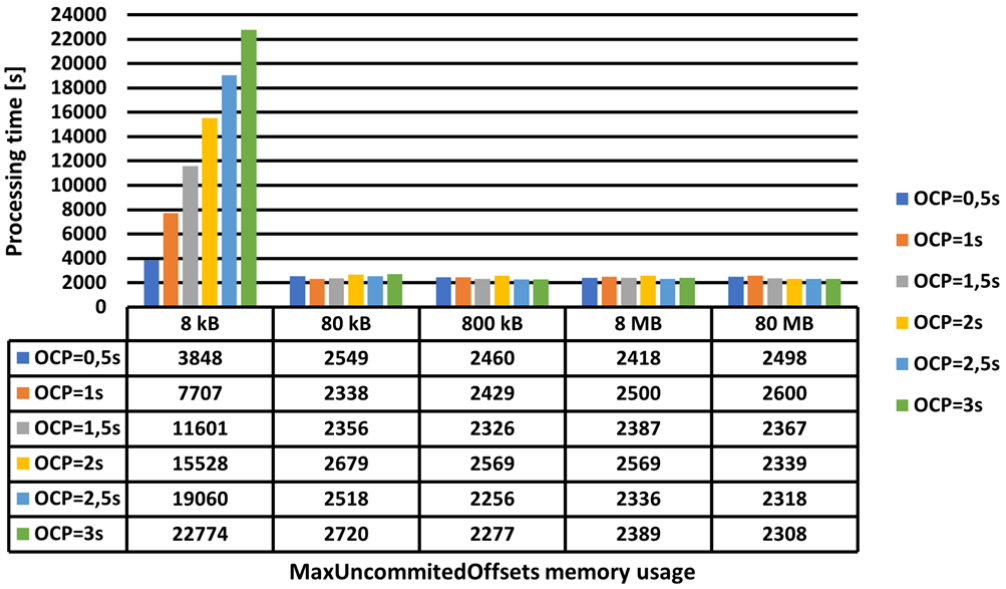
	8 kB	80 kB	800 kB	8 MB	80 MB
OCP=0,5s	3739	866	870	1001	1526
OCP=1s	7450	864	824	796	837
OCP=1,5s	11148	1297	828	788	946
OCP=2s	14837	1670	875	890	1014
OCP=2,5s	18524	2089	828	872	851
OCP=3s	22160	2491	884	832	1146

FIGURE 5.18 – Storm-HBase : Impact du paramètre MaxUncommittedOffsets sur le temps de traitement par l'utilisation de la mémoire



	8 kB	80 kB	800 kB	8 MB	80 MB
OCP=0,5s	4324	866	756	794	807
OCP=1s	7688	987	1020	1014	1055
OCP=1,5s	7656	1489	987	1053	1184
OCP=2s	7659	2032	1062	953	1076
OCP=2,5s	7655	2412	968	1064	879
OCP=3s	7625	2880	848	902	984

FIGURE 5.19 – Storm-Cassandra - Impact du paramètre MaxUncommittedOffsets sur le temps de traitement par l'utilisation de la mémoire



	8 kB	80 kB	800 kB	8 MB	80 MB
OCP=0,5s	3848	2549	2460	2418	2498
OCP=1s	7707	2338	2429	2500	2600
OCP=1,5s	11601	2356	2326	2387	2367
OCP=2s	15528	2679	2569	2569	2339
OCP=2,5s	19060	2518	2256	2336	2318
OCP=3s	22774	2720	2277	2389	2308

FIGURE 5.20 – Storm-HBase : Impact du paramètre MaxUncommittedOffsets sur le temps de traitement par l'utilisation de la mémoire

L'impact du paramètre *MaxUncommitedOffsets* sur le temps de traitement de notre architecture Kappa a été analysé pour les configurations suivantes : (**1**) Apache Kafka - Apache Storm - Apache HBase ; (**2**) Apache Kafka - Apache Storm - Apache Druid ; (**3**) Apache Kafka - Apache Storm - Apache Cassandra. Dont les résultats sont illustrés respectivement aux Figure 5.18 à 5.20.

Le but de ces expérimentations est d'étudier le lien entre le *MaxUncommitedOffsets* et le temps de traitement pour des valeurs d'OCP comprises entre 0,5s et 3s avec un pas de 0,5s. Ces expérimentations ont également permis de confirmer l'importance de la paramétrisation de Kafka et son impact sur la vitesse de traitement de l'architecture.

Ces expérimentations ont permis de montrer que la combinaison Apache Kafka comme file de message, Apache Samza associé à la base de données Druid était optimale pour une valeur de mémoire alloué à Kafka supérieure à 32 Ko et une valeur de 10s pour le paramètre *OffsetCommitPeriod* pour ce cas d'application en particulier.

5.5 Architecture applicative

Comme le montre la Figure 5.21, nous allons à présent nous intéresser à l'architure applicative.

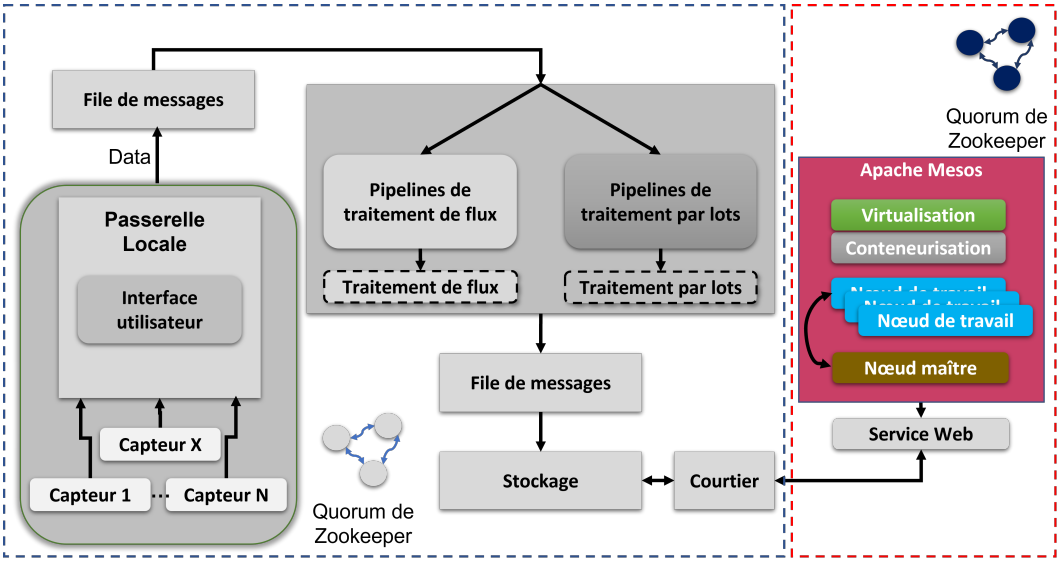


FIGURE 5.21 – Architecture applicative

L'architecture cloud d'hébergement d'applicatifs est construite au tour d'Apache Mesos. Comme le montre le Tableau 5.1, différentes solutions d'orchestration de container existent sur le marché. Kubernetes est une solution d'orchestration open source qui permet de déployer et de gérer de grandes quantités de containers Docker. Docker permet d'orchestrer des clusters containers Docker à l'aide de Swarm mais les développements de Docker bien qu'open source sont toujours dirigés par Docker Inc. Apache Mesos est open source permet l'orchestration de container Docker mais également d'autres systèmes de conteneurisation via Marathon. En Outre Mesos est spécifiquement développé pour le déploiement de cluster à haute performance. Mesos est n'est pas lié à une société commerciale contrairement à Singularity qui propose des spécifications similaires. Singularity est spécifiquement développé pour le domaine scientifique tandis que Mesos est généraliste. Singularity supporte outre les images le format de container OCI. Nomad propose une approche plus simple et plus flexible que ses concurrents en matière d'orchestration de containers, n'est pas dévolue au déploiement de cluster à haute performance et est maintenue par une société privée.

TABLE 5.1 – Comparaison des principales solutions d'orchestration de containers

Caractéristique	Kubernetes	Docker	Mesos	Singularity	Nomad
Compatibilité Docker	Oui	Oui	Oui	Oui	Oui
Autre format de conteneurisation	Non	Non	Oui	OCI	Oui
Open source	Oui	Oui	Oui	Oui	Oui
Support CUDA	Oui	Oui	Oui	Oui	Oui
Mainteneur	CNFC [5]	Docker Inc	Apache Foundation	Sylabs Inc	Hashi Corp
HPC	Non	Non	Oui	Oui	Non
Architecture modulaire	Non	Non	Oui	Non	Oui
Orchestrateur	Oui	Swarm	Marathon	Oui	Oui

Notre choix s'est porté sur Apache Mesos car il est maintenu par une fondation solide avec une communauté actives de développeurs. Outre l'hébergement et la gestion de containeurs, de gérer des environnements virtualisés et de répartir finement les ressources du cluster entre différents frameworks et services du cluster HPC. Il répond à toutes les exigences de notre cas d'utilisation en matière de sécurité, souplesses, évolutivité. Mesos est un système qui a été éprouvé avec succès sur un cluster de 2000 machines en production pendant 7 ans.

5.6 Architecture LAMA

Nos expérimentations, nous ont permis de déterminer que la brique de traitement est Samza. Ensuite, nous avons évalué les bases de données et montré que Druid était la plus performante.

Le recours à Apache Beam permet d'améliorer la souplesse de l'architecture cloud en réalisant un modèle de traitement qui peut ensuite être exécuté sur différentes briques logicielles. Apache Beam permet à la fois de répondre aux inconvénients qui sont formulés à l'encontre des architectures Lambda c'est-à-dire la nécessité d'un double développement pour la branche traitement de flux et traitement par lot et une maintenabilité couteuse. Le modèle et les SDK d'Apache Beam permettent de mettre en œuvre une architecture Lambda dite « unifiée » où les deux branches de traitements sont réalisées avec le même cadriciel (*Framework*) et donc de réduire les coûts de maintenance. De plus, la portabilité des codes développés sur différents types de nœuds ajoute de la souplesse par rapport à une Architecture Lambda classique. L'architecture de traitement peut donc évoluer en fonction du temps en changeant les briques logicielles sans toucher au code par simple recompilation du code initial.

Le recours à des composants logiciels issus de la fondation Apache assure la pérennité et la viabilité de l'architecture. De plus l'utilisation d'Apache Beam et la possibilité de changer de brique logicielle par recompilation du code facilitera la migration vers d'autres briques logicielles en cas de disparition de l'une d'entre elles. De plus, la licence Apache 2.0 permet de refermer le code et d'en faire, éventuellement, une valorisation commerciale.

La Figure 5.22 présente une vue globale de l'architecture LAMA, de ses différents composants.

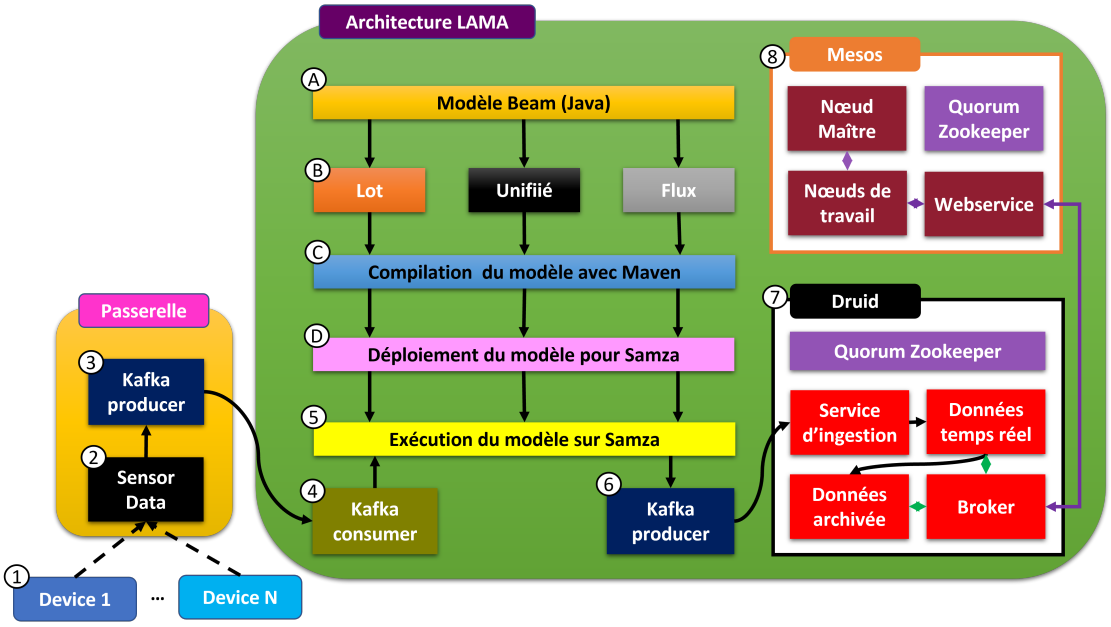


FIGURE 5.22 – Architecture LAMA

Le pipeline de traitement des données est créé préalablement au traitement des données à l'aide du modèle Apache Beam (**A**). Ce framework permet de développer et de maintenir un seul code qui peut ensuite être compilé pour être exécuté sur différentes briques logicielles d'exécution (runners). Ce framework permet de développer aussi bien des pipelines pour le traitement par lot, de flux de données ou unifiés c'est-à-dire pouvant traiter des données par lots et /ou en flux (**B**). Le modèle est ensuite compilé à l'aide d'Apache Maven (**C**) pour une brique logicielle supportée par Beam et ensuite déployé (**D**).

Les données produites par les capteurs (**1**) sont collectées et prétraitées (**2**) au niveau de la passerelle qui assure la commutation entre les protocoles radio fréquences (RF) et Internet. Les données sont régulièrement publiées (**3**) par la passerelle locale via le protocole TCP/IP dans un sujet (Topic) d'Apache Kafka préalablement créé. Ces données sont consommées (**4**) par le modèle exécuté sur la brique d'exécution du modèle Beam.

Les données produites à la fin de l'exécution des pipelines modélisés à l'aide d'Apache Beam sont envoyés dans une ou plusieurs partitions (**6**). Le nombre de partitions est fonction du nombre de services d'indexation qui vont devoir consommer simultanément l'information produite. En effet une partition ne peut être lue que par un et un seul Consommateur Kafka (*Kafka Consumer*) à la fois. Cependant, une tâche du service d'ingestion peut consommer les données sur plusieurs partitions simultanément. Les donnes sont ensuite ingérées par le Kafka Indexing Service (**7**) et rapidement ajoutées dans la base de données Apache Druid.

La Figure 5.23 détaille le fonctionnement internet de Druid et de son service d'ingestion de données.

Le service d'indexation de données Kafka permet de créer des superviseurs au niveau du nœud **Druid Overlord** qui supervise lest tâches d'ingestion des données au niveau des partition de Kafka. Le noeud **Druid Overloard** coordonne simultanément l'ingestion et la production des segments au niveau des **Druid Middle Manager**. Le traitement des données et la production des données est effectuée de manière parallèle au niveau de JVM [6] appelées « *Peons* ». L'ingestion des données permet de produire des "segment" qui sont les paquets de 2 à 4 millions de données. Les segments sont ensuite archivés à long terme (*Deep storage*) au niveau du système de fichiers HDFS [7].

Le nœud **Druid Coordinator** node est responsable de la répartition des segments entre les nœuds **Druid Historical**, du chargement de nouveaux segments, de l'abandon de segments (suppression), de la réplication des segments entre les nœuds **Druid Historical**. Le nœud Druid Coordinator est également connecté avec le quorum de Zookeeper et la base de données qui maintient les informations relative aux segments.

Le nœud broker permet à partir des métadonnées publiées par les noeuds **Druid Historical** de rediriger les requêtes vers les nœuds détenant les segments nécessaires à l'accomplissement de la requête.

6. Java Virtual Machine
7. Hadoop Distributed File System

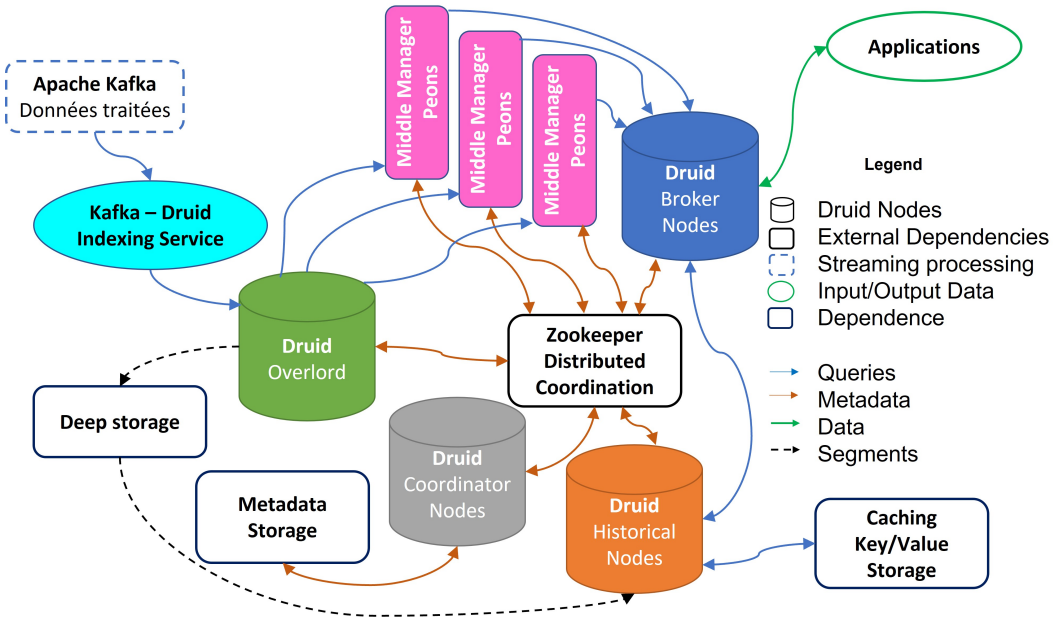


FIGURE 5.23 – Architecture interne d'Apache Druid

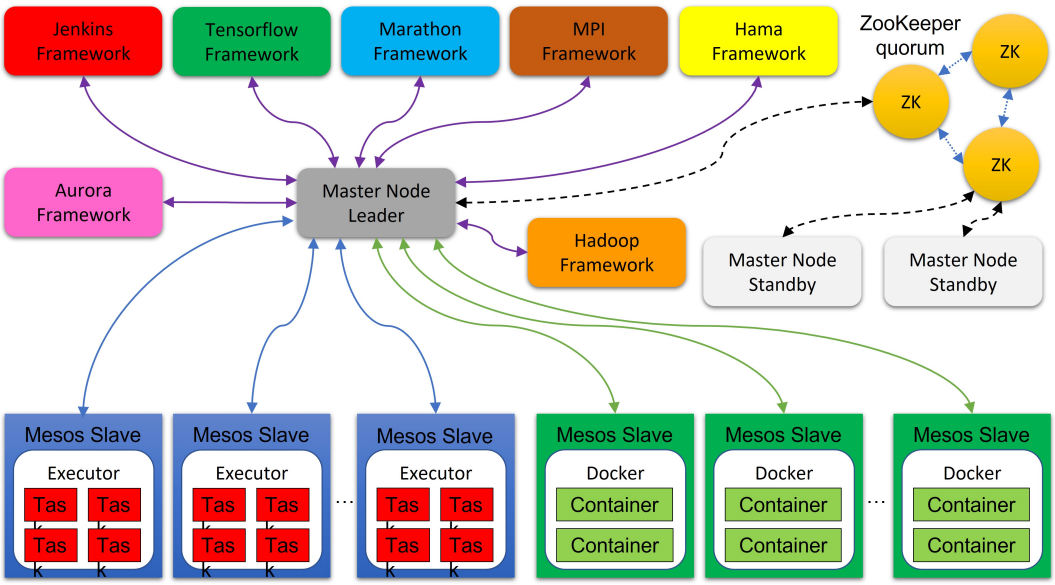


FIGURE 5.24 – Architecture interne d'Apache Mesos

Le système de cache permet d'améliorer la vitesse de requête pour les segments les plus demandés et lors du démarrage d'un nœud **Druid Historical** de rendre dis-

ponibles les nœuds données en caches plus rapidement en prenant connaissance des données présente dans le cache.

La Figure 5.24 présente les différentes composantes d'Apache Mesos. Un quorum de Zookeeper veille au bon fonctionnement des différents types de nœuds. Plusieurs nœuds maître sont présents mais un seul est élu et actif. Au moins deux autres nœuds maitre sont en attente et prêt à prendre la place du nœud maître en cas de défaillance. Lorsque qu'une demande de tâche est reçue, le quorum de Zookeeper est interrogé pour savoir quel est le nœud maître actif et rediriger la demande vers le nœud maître actif. Le nœud maître interroge chacun des nœuds de travail pour connaitre leur disponibilité et s'ils sont en mesure d'exécuter la tâche. Le premier en mesure d'exécuter la tâche se la voit attribuée. Si besoin le ou les framework nécessaire à l'exécution de la tâche sont déployés le temps de son exécution. La plupart des framework les plus usités sont disponibles pour Mesos. Mesos est également capable de déployer des conteneurs et de les orchestrer grâce à Marathon.

Les applications déployées dans les containers peuvent interroger la base de données Druid en passant par le service Web qui contrôle les accès et transmet à son tour la requête au nœud **Druid Broker**.

L'architecture LAMA est une architecture intégrant la collecte des données et l'hébergement d'applicatifs permettant au chercheur d'une part d'archiver leurs données expérimentales et de déployer leurs données modèles facilement grâce à la conteneurisation sur la partie applicative. L'ensemble est développé sur base de briques logicielles open source.

Un entrepôt de données ou Data Warehouse est un concept qui vient du monde de l'entreprise, qui consiste en une base de données dédiée au stockage de l'ensemble de données utilisées dans le cadre de la prise de décision et de l'analyse décisionnelle. Elle est alimentée par les base de données de production hétérogènes et variées à grâce à des ETL (Extract Transform Load) et organisée par thème. Cette base de données est également horodatée et non volatile c'est-à-dire que l'ensemble des données ne sont ni transformées ni altérées au fil du temps et des traitements.

Un parallèle peut être fait entre la structure d'un entrepôt de données et notre architecture. En effet, nous collectons des données provenant de capteurs et de sources de données hétérogènes qui sont ensuite transformées à l'aide d'ETL qui permettent ensuite de les stocker par expérimentation au sein de la base de données sous une forme horodatée. La base de données sert dans notre cas à l'analyse expérimentale.

5.7 Test de l'architecture

Les tests menés sur les composants clés de l'architecture ont permis de mettre en évidence les paramètres optimaux pour la configuration d'Apache Kafka, de choisir Apache Samza comme brique d'exécution et finalement Apache Druid comme base

de données. Notre architecture Lambda développée spécifiquement pour la recherche scientifique a été comparée avec une architecture Kappa optimisée spécifiquement pour la vitesse de traitement de données d'analyse comportementale de bovins.

Notre architecture se distingue fondamentalement d'une architecture Lambda classique par la nécessité de conserver aussi bien les données brutes que les données traitées ; ainsi que les métadonnées y afférent en vue de leur réutilisation éventuelle. Notre architecture est construite autour d'Apache Beam pour lui apporter la polyvalence en matière d'adaptabilité au niveau des briques logicielles de traitement. La base de données utilisée est Apache Druid.

L'architecture Kappa optimisée réalisée dans le cadre du mémoire de Jean Bertin NKAMLA PENKA utilise dans la configuration testée, une file d'attente Kafka, une brique de traitement Samza et une base de données Druid.

Les temps de traitement ont été comparés pour un fichier contenant 24h de données (2Gb de données brutes). Les temps de traitement total obtenus pour les deux architectures pour le traitement du même fichier d'entrées sont respectivement de 611,35s pour l'architecture Kappa et de 630,75s l'architecture Lama.

Il est également important de noter que le temps de traitement des données par Apache Samza Kappa et que la durée de traitement pour la Lambda intègre également les temps de lecture du fichier de données et d'écriture des résultats en base de données ainsi que le temps de compilation du pipeline de traitement. De plus, les traitements effectués sont de nature différente. L'architecture Kappa utilise un traitement par flux tandis que l'architecture Lambda utilise le traitement par lot pour traiter les données.

Dans l'absolu, le temps de traitement obtenu pour l'architecture Kappa est moindre (3,15%) que celui de l'architecture Lama. Cela s'explique notamment par le fait qu'une couche supplémentaire est utilisée au niveau de l'architecture Lambda (Apache Beam) pour lui apporter de la polyvalence et de la flexibilité. Cela se traduit par une baisse légère des performances. D'autre part, l'architecture Lambda modifiée est optimisée pour le stockage massif de très grandes quantités de données et le traitement par lots et n'est pas spécifiquement développée pour les performances mais pour le traitement de grande quantités de données et les traitements à long terme.

5.8 Évaluation de la durabilité de l'architecture

La pérennisation de l'architecture est liée aux briques logicielles qui y sont implémentées et à la société et / ou la communauté qui les maintient et les développent. En outre, les licences d'utilisation adossées à chacune des briques logicielles et à leurs interactions impactent directement la valorisation qui peuvent être faite des développements logiciels s'appuyant sur elles.

5.8.1 Analyse des licences logicielles

Les licences Open source peuvent être réparties suivant 4 catégories : (1) permissive, (2) à permission faible, (3) à forte protection et finalement (4) à protection en réseau.

1. Les licences **permissives** autorisent la commercialisation, la distribution, la modification et l'usage privé des logiciels et codes sources sous conditions que les copyrights et notices de licences soient incluses dans le logiciel. La fourniture du code source n'est pas obligatoire.
2. Les licences à **protection faibles** permettent un usage commercial, la distribution, la modification, garantisse un droit de licence aux contributeur et l'usage privé sous réserve de fournir le code source, mentionner les copyright et les notices de licence, d'adopter la même licence et de documenter tous les changements effectués dans le code. En cas de contribution importante par rapport au code source d'origine, le logiciel peut être distribué sous d'autres termes que ceux de la licence d'origine.
3. Les licences **protection forte** ont les mêmes conditions que les licences à protection faibles sans admettre la clause d'exemption en cas de contribution importante.
4. Les **licences en réseau** ajoutent aux licences à protection forte le droit à tout utilisateur interagissant avec l'application au travers d'un réseau de recevoir une copie du code source.

Comme le montre la Figure 5.25, la licence Apache 2.0 peut évoluer successivement vers les licences LGPL v3(+), GPL v3(+) et finalement AGPL v3. Une définition succincte des principaux types de licence est donnée dans le glossaire en fin de document.

Les briques logicielles utilisées pour la mise en place de l'architecture sont des projets de la fondation Apache sous licence Apache 2.0. Ce type de licence est très permissif et permettra soit d'évoluer vers d'autres types de licences plus contraignantes c'est-à-dire assurant un plus grand niveau de protection au niveau des droits d'auteur ou l'opposé de refermer le code en vue d'une éventuelle valorisation future et / ou dépôt de brevet. Par conséquent, le choix du type de licence adopté pour les développements fait autour des briques logicielles est aussi crucial et doit être compatible avec les contraintes des licences de briques logicielle.

Le Tableau 5.2 reprend la liste des composants logiciels actuellement utilisés pour le développement des architectures de stockage et de traitement et de la plateforme d'hébergement des applicatifs. L'ensemble des composants logiciels à l'exception de MySQL sont couverts par des licences permissives. Seul MySQL est couvert par une licence de type protection forte (GNU GPL v2). Comme nous n'envisageons pas de développements sur MySQL mais une simple utilisation, cela ne devra pas poser de problème. Toutefois, son type de licence implique qu'il ne soit pas distribué indépendamment du reste du projet.

TABLE 5.2 – Liste des composants logiciels utilisés et de leur licence

Logiciels utilisés	**Type de licence**	**Localisation dans l'architecture Lama**
Apache Beam	Apache 2.0	Architecture Lambda
Apache Samza	Apache 2.0	Architecture Lambda
Apache Kafka	Apache 2.0	Architecture Lambda
Apache Druid	Apache 2.0	Stockage
Redis	BSD	Stockage
PostgreSQL	PostgreSQL	Stockage
MySQL / MariaDB	GNU GPLv2	Passerelle, Plateforme applicative
Apache Jena	Apache 2.0	Passerelle, Plateforme applicative
Apache http Server	Apache 2.0	Passerelle, Plateforme applicative
PHP	PHP License v3.01	Passerelle, Plateforme applicative
NodeJS	MIT	Webservice
Node Express Gateway	Apache 2.0	Webservice
Apache Mesos	Apache 2.0	Plateforme applicative
Apache Marathon	Apache 2.0	Plateforme applicative
Jenkins	MIT	Intégration continue / tests unitaires
Java Open JDK 8	GNU GPL v2	Partout
Apache Maven	Apache 2.0	Partout
Apache Zookeeper	Apache 2.0	Architecture Lambda et applicative
Exhibitor	Apache 2.0	Quorum Zookeeper

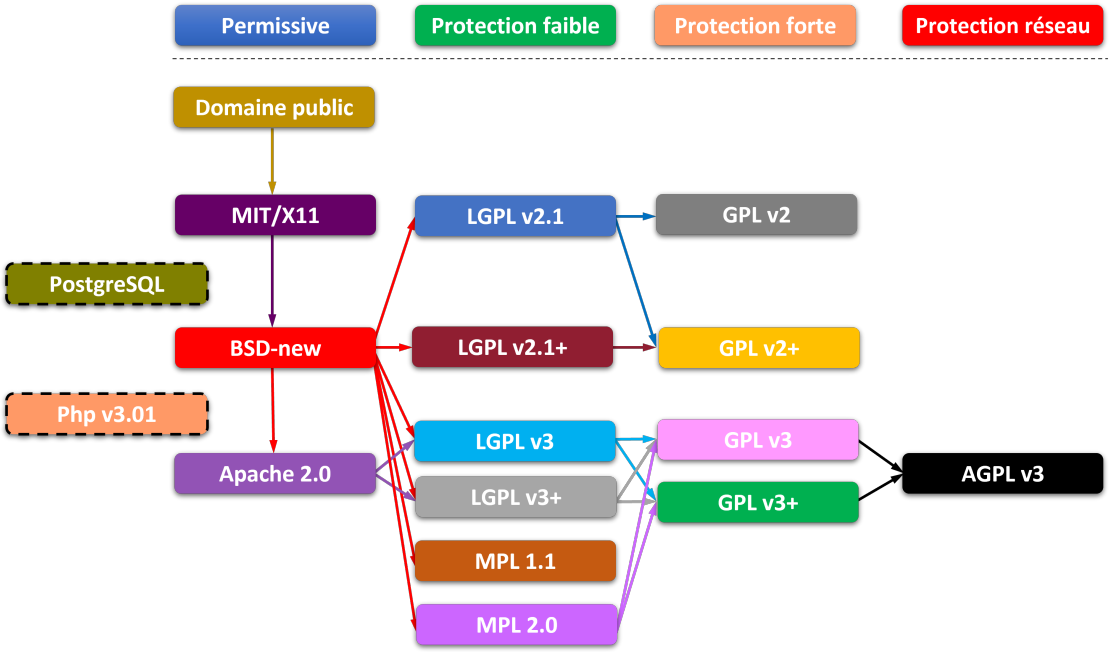


FIGURE 5.25 – Relation entre les principaux types de licence open source

Le Tableau 5.2 contient les licences des briques logicielles d'exécution alternatives à Apache Samza. A l'exception de Google Cloud Dataflow qui est une technologie propriétaire de Google, toutes les autres relèvent de la licence Apache 2.0.

Outre l'architecture cloud en elle-même, un point important qui conditionne son bon fonctionnement est la manière dont les données sont collectées et la manière dont les données sont transmises à cette architecture cloud. Les flux de données peuvent être transmis de manière continue, périodiquement à intervalles réguliers, ou par rafale quand un évènement spécifique déclencheur se produit.

5.8.2 Analyse de la qualité logicielle

La durabilité de l'architecture dépend outre le type de licence des composants logiciels constitutifs, par la qualité intrinsèque du code de ces composants logiciels. Les composants logiciels ont été clonés à partir de leur dépôt GitHub respectifs avant d'être analysés en ligne à l'aide de SonarCloud [8]. Les métriques utilisées pour l'analyse sont les SonarLint qui sont fournies pour l'ensemble des langages utilisés pour coder les composants candidats. Au terme de chacune des analyses, un tableau récapitulatif reprenant les évaluations sous forme de notations alphabétiques et numériques des principales métriques est généré.

8. https ://sonarcloud.io

TABLE 5.3 – Evolution de la qualité logicielle des briques logicielles utilisables par notre architecture

Runner	Date	Reliability		Security			Maintainability			Duplications	
		Bugs	**Class**	**Vulnera-bilities**	**Class**	**Security Hotspots**	**Debt**	**Class**	**Code Smells**	**Rate**	**Blocks**
Apex-core	9/7/19	2	C	0	A	0	15min	A	4	0,00%	0
Apex-malhar	9/7/19	66	D	0	A	0	7h	A	95	0,80%	10
Beam	9/7/19	256	C	0	A	156	57d	A	4,7K	2,80%	162
	22/11/20	162	E	0	A	179	62d	A	4,5K	2,60%	309
Flink	9/7/19	159	D	0	A	17	49d	A	3k	10,00%	2,8k
	22/11/20	764	D	0	A	20	61d	A	3,5k	9,80%	2,9k
Gearpump	9/7/19	5	E	1	E	0	2d	A	158	0,90%	19
	22/11/20	6	E	0	A	0	2d	A	158	0,19%	19
Hazelcast-jet	9/7/19	1	B	0	A	0	15min	A	3	0,00%	0
	22/11/20	0	A	0	A	1	1h	A	14	0,00%	0
Druid	9/7/19	52	C	13	B	110	6d	A	434	1,00%	33
	22/11/20	2	C	0	A	7	2d	A	154	1,60%	33
Nemo	9/7/19	0	A	0	A	5	2d	A	151	0,00%	0
	22/11/20	5	B	0	A	0	2d	A	222	0,00%	0
Samza	9/7/19	60	D	0	A	13	3d	A	234	1,70%	38
	22/11/20	69	D	0	A	16	3d	A	248	2,10%	46
Spark	9/7/19	217	D	13	E	222	80d	A	6,4k	3,70%	1,5k
	22/11/20	144	D	0	A	99	94d	A	7,5k	3,60%	1,7k
Twister 2	22/11/20	85	E	0	A	44	608d	D	58k	54,10%	51k

L'analyse du code source proposée par Sonarqube se focalise sur (**1**) La fiabilité est caractérisée par la présence de bogues dans le code et leur niveau de criticité, d'une note globale allant de A à E (A étant la meilleure note et E la plus mauvaise) qui synthétise la fiabilité du code. (**2**) La sécurité est évaluée par la détection des vulnérabilités présentent dans le code, une note globale qui permet d'apprécier globalement le niveau de sécurité, et des problèmes critiques de sécurité. (**3**) La maintenabilité au travers de la dette technique exprimée sous forme d'une durée de travail pour y remédier, d'une note globale allant de A à E, ainsi que la détection d'un nombre de mauvaises pratiques présentes dans le code source.

L'analyse de l'évolution de la qualité logicielle des différentes briques montrent que les indicateurs d'apache Druid se sont améliorés significativement. Le nombre de bugs dans Apache Beam a été significativement réduit mais la dette technique est toujours présente. La qualité d'Apache Samza n'a quant à elle pas évolué et reste toujours moyenne comme celle d'Apache Flink. Le code de Gearpump reste quant à lui de mauvaise qualité. La qualité logicielle d'Apache Nemo n'a pas évolué car il n'y a pas eu de développement significatif entre temps. Apache Spark souffre toujours d'une dette technique importante et d'une niveau de qualité de logiciel au regard des métriques de Sonarqube assez mauvaise. Finalement, le Twister 2 est en plein développement est à une qualité logicielle comparable celle d'Apache Spark et mais un taux de duplication de code important, probablement dû à un développement très actif.

Par ailleurs, entre juillet 2019 et novembre 2020 l'exécuteur (*runner*) Apex a été retiré et Twister 2 est apparu ce qui montre tout l'intérêt de notre démarche en matière de flexibilité et de durabilité de l'architecture.

5.9 Conclusion

Dans ce chapitre nous avons décrit la mise en œuvre d'une architecture cloud LAMA pour la recherche en Smart Farming basée sur les exigences de notre cas d'utilisation représentatif des exigences liées d'une part par la recherche et d'autre part par le domaine d'activité. Notre proposition architecturale s'appuie sur un framework qui nous permet de configurer notre architecture aussi bien en Lambda pour le traitement des flux temps réels et des lots de données ou en Kappa pour un traitement en temps réels des flux et un traitement par lot émulés sous forme de micro-lots de données.

Le recours à Apache Beam permet d'améliorer la souplesse de l'architecture cloud en réalisant un modèle de traitement qui peut ensuite être exécuté sur différentes briques logicielles. Apache Beam permet à la fois de répondre aux inconvénients qui sont formulés à l'encontre des architectures Lambda c'est-à-dire la nécessité d'un double développement pour la branche traitement de flux et traitement par lots et une maintenabilité coûteuse. Le modèle et les SDK d'Apache Beam permettent de mettre en œuvre notre architecture Lambda « unifiée » où les deux branches de traitements

sont réalisées avec le même framework et donc de réduire les coûts de maintenance. De plus, la portabilité des codes développés sur différents types de nœuds ajoute de la souplesse par rapport à une Architecture Lambda classique. L'architecture de traitement peut donc évoluer en fonction du temps en changeant les briques logicielles sans toucher au code par simple recompilation du code initial.

Le recours à des briques logicielles issues de la fondation Apache assure la pérennité et la viabilité de l'architecture. De plus l'utilisation d'Apache Beam et la possibilité de changer de brique logicielle par recompilation du code facilitera la migration vers d'autres briques logicielles en cas de disparition de l'une d'entre elles. De plus, la licence Apache 2.0 permet de refermer le code et d'en faire une valorisation commerciale (par ex. spin off).

L'analyse à l'aide de SonarQube a permis de montrer une évolution positive de la qualité d'Apache Druid et d'Apache Beam ce qui est aussi un gage de durabilité. Entre les deux analyses successives séparées de 17 mois, nous avons pu observer la disparition d'Apache Apex et l'apparition de Twister2 une nouvelle brique logicielle démontrant ainsi la dynamique qui opère au niveau des briques logicielles d'exécution et la nécessité de pouvoir évoluer.

La comparaison des temps de traitement avec une architecture Kappa optimisée avec notre architecture configurée en Lambda a montré un temps d'exécution plus long 3,15% dû au framework intercalaire qu'est Beam. La souplesse se fait au détriment d'une légère perte de vitesse de traitement.

6

Automatisation grâce aux métadonnées du traitement et du stockage des données

Plan du chapitre

Résumé

Les architectures cloud sont déployées pour traiter un ou au plus quelques cas d'utilisation. Toute modification dans l'application a des conséquences directes sur l'architecture au niveau du prétraitement (ETL) et du schéma d'ingestion des données au niveau de la base de données. Dans un contexte de recherche, la documentation des données, des traitements et des conditions expérimentales est cruciale. Les conditions expérimentales, les capteurs utilisés pour l'acquisition des données varient très fréquemment et implique une adaptation des traitements. De plus, les chercheurs n'ont pas vocation à adapter les chaînes de traitement et l'ingestion des données. La sémantique permet d'accompagner les données de métadonnées qui décrivent notamment leur origine, leur structure, les différents traitements qui leur ont été appliqués. Nous proposons dans ce chapitre d'utiliser la sémantique pour automatiser l'adaptation des pipelines de traitement des données ingérées. Notre contribution utilise la sémantique expérimentale préalablement encodée par les chercheurs durant l'expérimentation pour automatiser la mise en forme des données collectées et leur ingestion au niveau de l'architecture LAMA.

Publications liées à ce chapitre

R. Ait Abdelouahid, **O. Debauche**, S. Mahmoudi, A. Marzak, P. Manneback, F. Lebeau, "Open Phytotron : A New IoT Device for Home Gardening,". In : The 5th International Conference on Cloud Computing and Artificial Intelligence : Technologies and Applications (CloudTech), Marrakech, Morocco, 2020. doi : 10.1109/CloudTech49835.2020.9365892.

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, F. Lebeau, "Edge Computing and Artificial Intelligence Semantically Driven. Application to a Climatic Enclosure". In : The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC). *Procedia Computer Science*, 2020, **175** : 542–547. doi : 10.1016/j.procs.2020.07.077.

R. Ait Abdelouahid, **O. Debauche**, S. Mahmoudi, A. Marzak, P. Manneback, F. Lebeau, "Smart Nest Box : IoT based Nest Monitoring In Artificial Cavities,". In : The 3rd International Conference on Advanced Communication Technologies and Networking (CommNet 2020), Marrakech, Morocoo, 2020. doi : 10.1109/CommNet49926.2020.9199624.

O. Debauche, R. Ait Abdelouahid, S. Mahmoudi, Y. Moussaoui, A. Marzak, P. Manneback, "ℜevoCampus : a distributed open source and low-cost smart campus,". In : The 3rd International Conference on Advanced Communication Technologies and Networking (CommNet 2020), Marrakech, Morocoo, 2020. doi : 10.1109/CommNet49926.2020.9199640.

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, J. Bindelle, F. Lebeau, "Edge Computing for Cattle Behavior Analysis,". In : Second international conference on Embedded & Distributed Systems (EDiS'2020), 2020. doi : 10.1109/EDiS49545.2020.9296471.

6.1 Introduction

L'énorme quantité de données produites dans des formats variés par les objets connectés n'est pas sans poser de problèmes pour les exploiter. L'une des approches pour y remédier consiste à associer à ces données de la sémantique descriptive propre au contexte. Ces données sémantiques permettent au final, aux systèmes informatiques de posséder des connaissances et de faciliter la prise de décisions.

Toutefois, pour être efficace, la sémantique doit être générique pour permettre l'interopérabilité entre objets, la mise en place de mécanismes de diagnostic et de supervision automatiques, et l'exploitation des données. Cet enrichissement des données par la sémantique peut se faire avant la collecte et/ou avant ou après le stockage des données. Avec l'ajout de ces métadonnées, les données ne sont plus spécifiques à l'application pour laquelle elles ont été collectées, ce qui élargit leur champ d'exploitation.

Les technologies sémantiques se basent sur des représentations interprétables par les machines qui facilitent le partage, l'intégration des données, l'interopérabilité des systèmes, la découverte de ressources, la modélisation et l'interrogation de l'information, ainsi que l'inférence de nouvelles connaissances. Cependant les représentations sémantiques et les techniques de raisonnement nécessitent une quantité considérable de ressources. Les volumes conséquents de données à traiter et les ressources limitées des nœuds implique la nécessité de nouvelles architectures utilisant conjointement le Edge et le Cloud pour fournir des services performants et de qualité.

6.2 Technologies sémantiques

Le consortium World Wide Web Consortium (W3C) a développé une famille de standards Web sémantique (*Semantic Web*) parmi ces technologies on peut citer Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL).

Le RDF [151] utilise un modèle de données basé sur les graphes qui lui permet de représenter une structure arbitraire sans un schéma a priori. Le graphe est constitué de déclarations structurées comme suit : (sujet, prédicat, objet), ce qui peut se traduire par un objet o a une relation p avec un sujet s). Le RDF peut être sérialisé dans différents format tels que : RDF/XML [152], JSON for Linked Data (JSON-LD) [153], N-Triples [154], N-Quads [155], Turtle[156], Notation 3 (N3) [157], Entity Notation (EN) [158]. Le RDF fournit un vocabulaire de modélisation des données pour des concepts comme : classe, sous-classe, domaine, intervalle. Il peut être utilisé pour créer de simples ontologies sur RDF. Le OWL [159] étend RDFS avec un vocabulaire plus complet pour modéliser des ontologies complexes (Figure 6.1).

Un raisonneur sémantique infère des conséquences logiques à partir d'un ensemble explicite de faits affirmés ou d'axiomes. L'inférence sémantique découvre de nouvelles

FIGURE 6.1 – Stack RDF

relations basées sur les données et informations additionnelles sous la forme de vocabulaire (ontologie et un ensemble de règles). Les raisonneurs sémantiques les plus connus sont HermiT [160], Owlgres [161], Pellet [162] et Apache Jena [163]. Jena est un cadriciel (*framework*) utilisé pour construire des applications sémantiques à partir d'un moteur d'inférences basé sur des règles et capable d'effectuer des raisonnements au départ de RDFS et d'ontologies OWL.

6.3 Utilisation de la sémantique dans l'architecture LAMA

L'intégration de la sémantique au niveau de l'architecture LAMA se fait d'une part au niveau de la passerelle réseau où la sémantique est encodée et d'autre part au niveau de l'architecture cloud ou les pipelines de traitement des données ingérées sont adaptés. La Figure 6.2 présente la chaîne complète de l'encodage préalable des conditions expérimentales à l'adaptation des pipelines et finalement l'ingestion des données.

La chaîne de traitement consiste en deux étapes qui se matérialise par des numéros cerclés de rouges pour la sémantique et de noir pour la transmission des données. Le traitement sémantique débute par l'encodage des métadonnées expérimentales au niveau de l'interface web hébergée sur la passerelle réseau (1) qui s'appuie sur un service web (*Web Service*) (2) qui joue le rôle d'intermédiaire entre l'interface utilisateur et la base de données MySQL (3). La sémantique est générée à partir des données encodées par les chercheurs au niveau de l'interface Web (4). La sémantique produite est publiée sur le bus de messages Kafka (5) avant d'être consommées par l'architecture cloud (6) pour adapter les pipelines de traitement des données qui vont être transmises.

Les données des capteurs (*Sensors*) sont envoyées via le périphérique (*Device*) (**1**) à la passerelle locale (**2**) (*Local Gateway*), généralement, à intervalles réguliers à l'aide de différents protocoles radio fréquences ; matérialisés par les flèches en pointillés sur la Figure 6.2. Dans le cadre de notre cas d'utilisation « vaches connectées » ce sont les protocoles de communication Wi-Fi et le LoRa qui sont envisagés. Ces deux protocoles ont des débits de transmission de données très différents. Le Wi-Fi permet de décharger

rapidement un volume de données important tandis que le LoRa envoie des quantités de données très faibles de manière régulière avec une propagation pouvant atteindre plusieurs km. Les données sont publiées dans un bus de message Apache Kafka (**3**) et sont ensuite consommées par l'architecture cloud (**4**).

L'architecture cloud a également la possibilité d'interroger l'API du webservice pour récupérer les sémantiques en vue de les synchroniser avec sa base de données locale en cas de panne par exemple. Cette possibilité est matérialisée par la flèche verte sur la Figure 6.2.

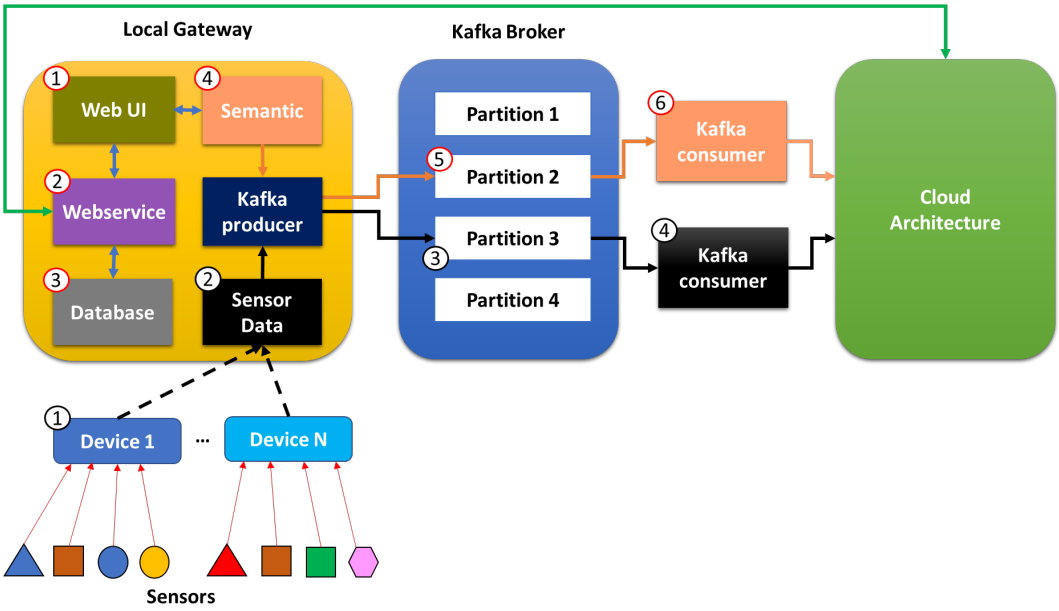


FIGURE 6.2 – Collecte et transmission des données vers l'architecture cloud

Les paragraphes qui suivent décrivent l'encodage des métadonnées expérimentales, leur transformation en RDF et l'adaptation des pipelines de traitement à partir des méta données.

6.3.1 Encodage des métadonnées expérimentales

L'encodage des métadonnées expérimentales réalisé par le chercheur sur la passerelle réseau, permet de définir des profils associant des capteurs contenu dans différents périphériques et le sujet faisant l'objet de l'expérimentation.

La Figure 6.3 illustre d'une manière schématique l'association des capteurs, périphériques,sujet formant le profil expérimental.

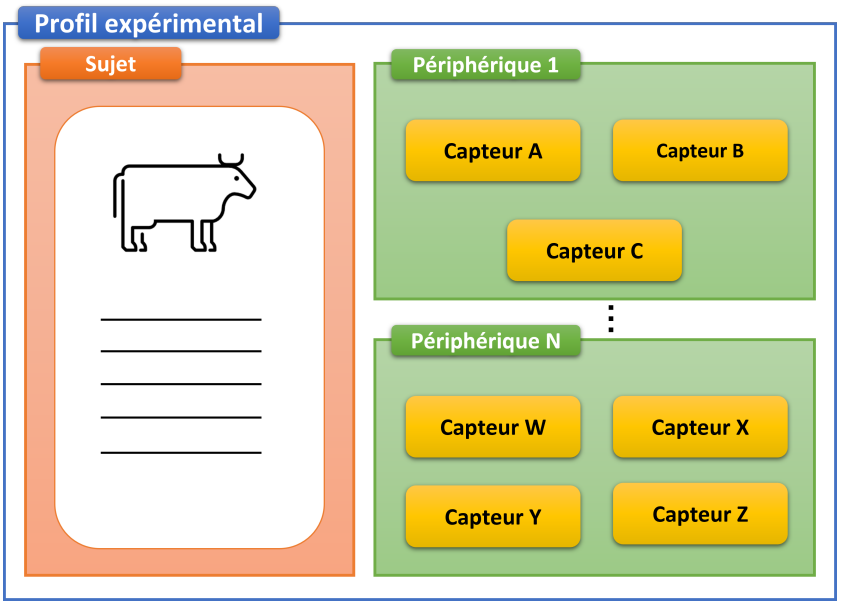


FIGURE 6.3 – Description à haut niveau de l'encodage des métadonnées

Afin d'illustrer l'utilisation des métadonnées expérimentales, nous décrivons un cas d'utilisation d'acquisition des données d'un iPhone 4S installé sur le cou d'une vache expérimentale nommée Natacha en vue de réaliser l'analyse de son comportement. Le périphérique est dans ce cas un iPhone 4S et les capteurs utilisés sont l'accéléromètre et le gyroscope contenus dans la centrale inertielle et le GPS de l'iPhone . Le sujet de l'expérimentation est la vache expérimentale Natacha. L'association de l'iphone 4S sur le cou de Natacha constitue le profil expérimental.

6.3.2 Transformation des métadonnées en RDF

L'étape suivante consiste à transformer les métadonnées sous forme de triplet RDF (sujet, prédicat, objet) associant la description d'une ressource (*sujet*), une propriété (*prédicat*) et un *objet* qui peut être une autre ressource ou une valeur.

Dans l'exemple proposé au paragraphe précédent, nous avons défini le profil expérimental comme l'association d'un sujet faisant l'objet de l'expérimentation avec un ou plusieurs périphériques contenant eux-mêmes un ou plusieurs capteurs. Si nous traduisons ces concepts sous forme de triplets RDF, le sujet est Natacha, la propriété est le périphériques et l'objet est l'iPhone 4S. Si plusieurs périphériques sont associés au sujet un triplet sera créé par périphérique. Les associations suivantes concerne le périphérique. Celui-ci devient à son tour le sujet, le prédicat sera le type de capteur, et la valeur sera le modèle de capteur. Dans notre exemple, le sujet est l'iPhone 4S à trois prédicats : (1) accéléromètre dont la valeur est L1S331DLH ; (2) gyroscope dont la valeur

TABLE 6.1 – Triplets de description de l'accéléromètre

Sujet	Predicat	Objet
L1S331DLH	vendor	STMicroelectronics
L1S331DLH	description	Accéléromètre
L1S331DLH	première valeur renvoyée	GravAcc_X
L1S331DLH	deuxième valeur renvoyée	GravAcc_Y
L1S331DLH	troisième valeur renvoyée	GravAcc_Z
L1S331DLH	gamme de mesure	$\pm$2g
L1S331DLH	facteur d'échelle	1 mg/digit
L1S331DLH	erreur de facteur d'échelle	$\pm$10%
L1S331DLH	variation du facteur d'échelle avec la température	$\pm$0.01 %/C
L1S331DLH	erreur de biais	$\pm$20 mg
L1S331DLH	erreur de biais avec de la température	$\pm$0.1 mg
L1S331DLH	densité du bruit	218 ug/$\sqrt{Hz}$

est L3G4200D ; (3) GPS dont la valeur est BCM4750. Chaque senseur fait ensuite l'objet d'une description de ses caractéristiques. A titre d'exemple, nous ne décrivons que l'accéléromètre L1S331DLH qui est maintenant le sujet et les prédicats sont les différentes caractéristiques de ce capteur et les objets sont les valeurs associées à ces caractéristiques. Le tableau 6.1 donne les triplets associés à la description de l'accéléromètre.

6.3.3 Exploitation de la sémantique pour adapter les pipelines de traitement

Les données et métadonnées sont publiées par la passerelle locale via le protocole TCP/IP dans deux sujets (*Topics*) d'Apache Kafka préalablement créés. Les métadonnées sont consommées par le service (**1**) qui identifie si un pipeline de traitement associé au profil sémantique transmis existe. Si aucun pipeline de traitement n'est associé à la sémantique transmise, un nouveau pipeline est généré à l'aide du modèle Beam (**2**). Les pipelines générés à l'aide du framework Apache Beam (**3**) peuvent traiter les données par lots (*Batch Processing*), par flux (*Stream Processing*) ou être unifiés c'est-à-dire traiter indifféremment les données en lots et en flux (*Batch and Stream Processing*). Les pipelines sont ensuite compilés (**4**) avant d'être déployé le cloud (**5**) et finalement exécuté à l'aide d'une des briques d'exécution supporté par Apache Beam (Runner)(**6**). Les données présentes dans le topic Kafka sont alors consommées et le produit du traitement est la production d'une ou plusieurs topics Kafka.

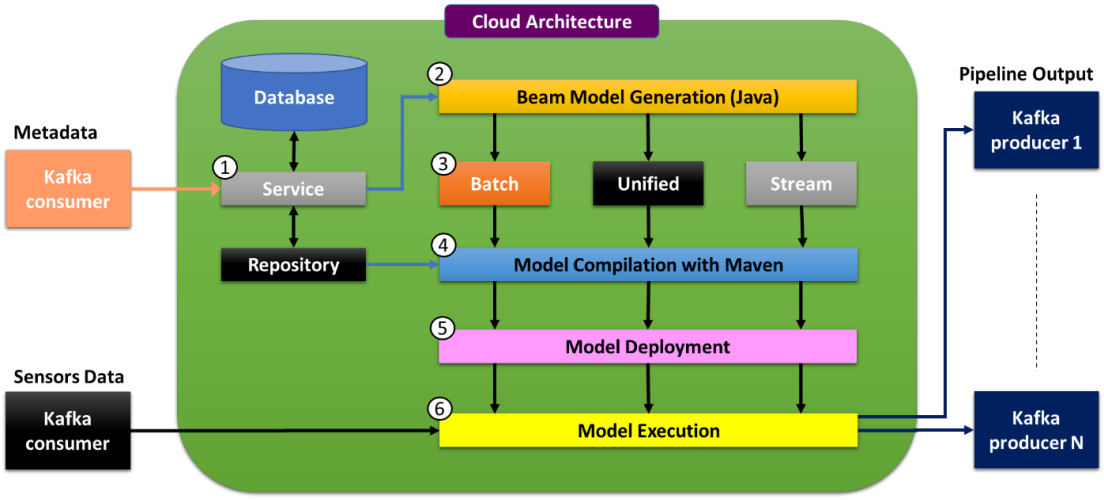


FIGURE 6.4 – Architecture cloud basée sur Apache Beam

6.4 Implémentation

Notre architecture LAMA utilise une passerelle réseau local (située en périphérie du réseau) pourvue de capacité de stockage pour la centralisation des données. Dans de nombreux cas les passerelles de capteurs se limitent à assurer du transfert de données en assurant une interopérabilité entre les protocoles réseau. L'utilisation d'une passerelle réseau (Figure 6.5) est un élément essentiel qui permet de soulager le réseau et les infrastructures cloud en procédant au traitement local des données. Cependant dans le cas des données à caractères scientifiques, toutes les données sont importantes et doivent nécessairement être transmises. Afin de limiter au mieux les flux de données transférées, la passerelle transfert au préalable la sémantique et ensuite les données par micro-batchs. Les données en Smart Farming peuvent relever de la confidentialité, du secret industriel ou de fabrication et doivent dès lors être cryptées avant d'être transmises.

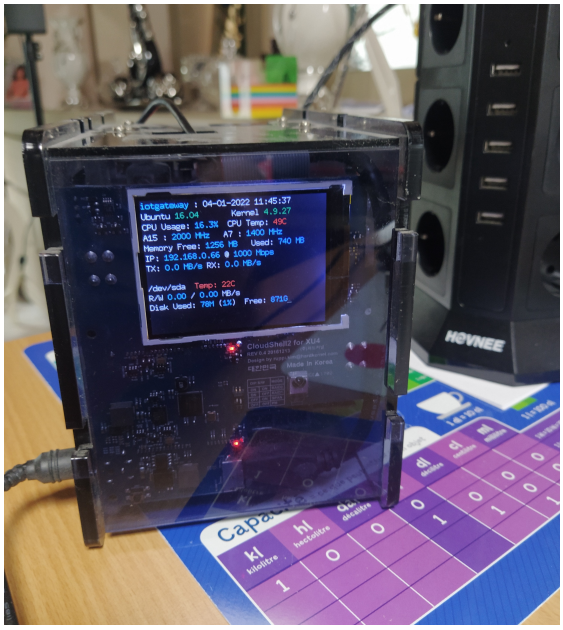


FIGURE 6.5 – Passerelle réseau installée à proximité des capteurs

La passerelle locale (Figure 6.5) collecte les données en provenance des périphériques (*Devices*) et permet l'encodage des données sémantiques préalables à l'envoi de données. La passerelle, un CloudShell 2 équipé de deux disques durs de 1 To montés en RAID1 et propulsée par une Odroid XU4Q fonctionnant sous Ubuntu 18.04 LTS. k3s est utilisés comme technologie conteneurisation qui héberge l'interface web d'encodage des métadonnées, l'application de conversion des métadonnée au format RDF, le serveur de base de données et le web service.

L'interface web d'encodage des métadonnées développée en PHP 8, HTML 5, Bootstrap 5.1 est illustrée à la Figure 6.6. Les données sont stockées localement dans une base de données MySQL 8.0.14 via le we-b service développé en NodeJS avec le framework Express Gateway qui permet de gérer les accès de manière sécurisée grâce à l'utilisation de tokens.

FIGURE 6.6 – Menu de l'interface de gestion des données sémantiques des capteurs

Les boutons « *Targets* », « *Devices* », « *Sensors* » permettent de définir les éléments constitutifs des profils expérimentaux. Les « Profils expérimentaux » décrivent l'association d'un ou plusieurs périphériques muni de ces capteurs implantés sur un sujet d'expérimentation biologique (animal ou végétal) ou inerte.

Chaque capteur est décrit au niveau de l'onglet « Sensor » à l'aide des champs suivants : (**1**) Nom de la variable qui contiendra la donnée ; (**2**) Modèle du capteur ; (**3**) Marque du capteur ; (**4**) Description du capteur ; (**5**) Valeurs minimales et maximales que peut prendre la mesure ; (**6**) Unité de la mesure ; (**7**) Précision de la mesure ; (**8**) Expression de la précision (pourcentage, absolue). Ces données sont en partie reprises dans le tableau qui liste les senseurs (Voir Figure 6.7). Dans cet exemple, le senseur mesure l'accélération gravitaire selon l'axe de X. La donnée sera transmise dans la variable *GravAcc_X*. Elle peut varier entre -19.62 et 19,62 m/s^2. La précision du capteur est de 0,01 m/s^2. D'autres champs peuvent être ajoutés pour décrire des particularités ou répondre à des besoins spécifiques.

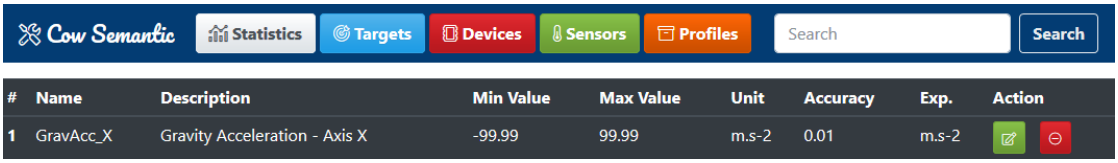


FIGURE 6.7 – Onglet de gestion des capteurs

Les senseurs sont associés à un ou plusieurs périphériques qu'il convient également de décrire précisément. Les métadonnées expérimentales reprennent l'identifiant du périphérique, sa description ainsi que la liste des capteurs qui y sont attachés. Tous les capteurs attachés à un périphérique ne sont pas forcément actifs. La position du périphérique est également décrite le plus précisément possible. Finalement, le dernier champ de l'interface reprend à titre indicatif, la dernière date / heure où des données ont été transmises par ce périphérique à la passerelle.

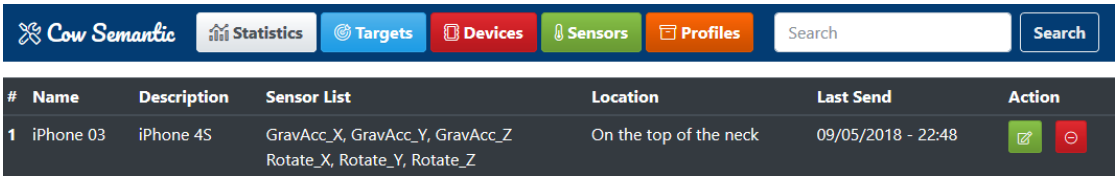


FIGURE 6.8 – Onglet de gestion des périphériques

L'onglet **Targets** reprend le listing des cibles sur lesquelles des périphériques peuvent être implanté. Dans l'exemple illustré à la Figure 6.9, la cible est une vache expérimentale nommée Natacha et reprise sous l'identifiant Cow 1.

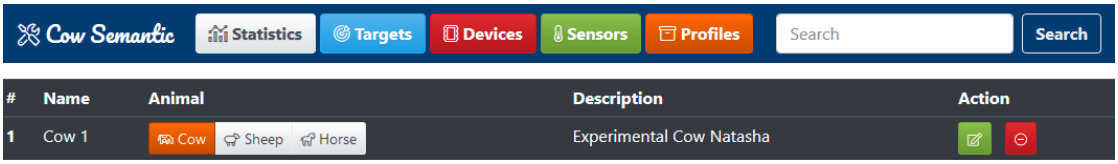


FIGURE 6.9 – Onglet de gestion des cibles

Finalement, l'onglet **Profils** reprend l'association d'une cible avec un périphérique et la liste des senseurs actifs parmi ceux connectés au même périphérique. Le mode de traitement des données qui seront transmises est également indiqué (lots ou flux).

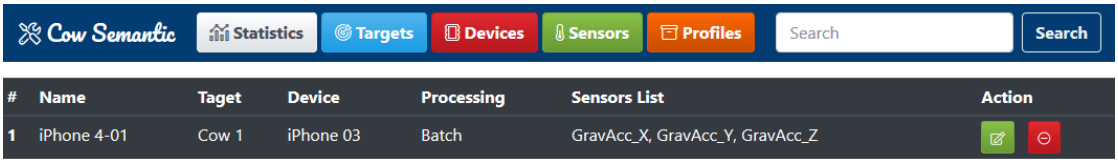


FIGURE 6.10 – Onglet de gestion de profils

Apache Jena est utilisé pour générer les fichiers, en suivant le modèle de graphe RDF (Resource Description Framework). Chaque fichier reprend les informations sémantiques contenues dans un profil expérimental associant capteur, périphériques et l'objet biologique étudié. Le format RDF/JSON reprend les informations structurées sous forme de triplets. Dans un graphe RDF, le motif sujet-prédicat-objet dans lequel le nœud d'origine est le sujet, l'arc est le prédicat et le nœud cible est l'objet.

6.5 Expérimentation

Notre expérimentation a consisté a décrire complètement la centrale inertielle et le GPS d'un iPhone 5S sous forme de métadonnées expérimentales pour ensuite générer le pipeline de traitement associé pour déterminer les comportements à partir d'un arbre de décision. Nous avons ensuite réutilisé le même iPhone 5S installé sur nouveau sujet d'étude afin de montrer la réutilisation automatique du pipeline précédemment généré.

La Figure 6.12 montre la traduction des concepts de sujet d'études (*Target*), Périphérique (*Device*), Capteur (*Sensor*) regroupé au sein d'un Profil (*Profile*) traduit sous forme d'un diagramme de classes simplifié.

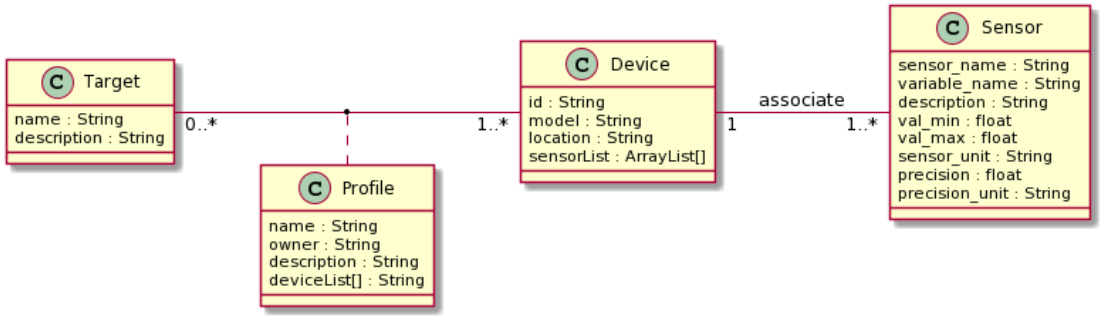


FIGURE 6.11 – Diagramme de classes simplifié

Les cardinalités des relations ne pouvant pas être représentées en RDF, ils ont dû être remplacés par des "rôles". L'adaptation du diagramme de classes simplifié est présentée à la Figure 6.12.

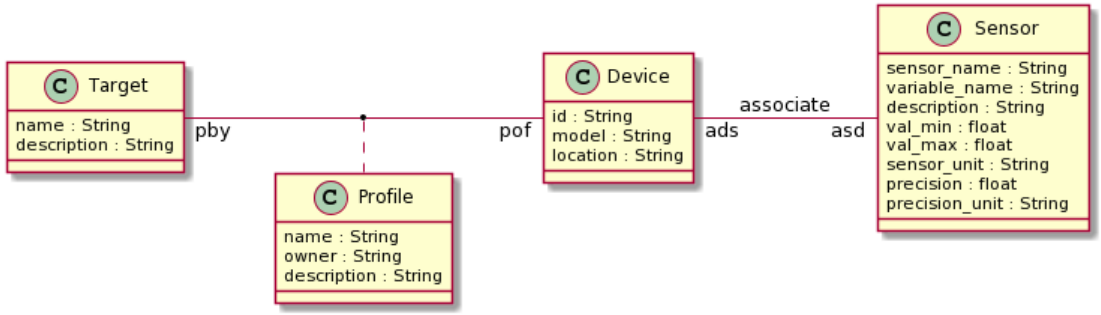


FIGURE 6.12 – Diagramme de classes adapté

Le diagramme de classes adapté est ensuite traduit sous forme d'une représentation RDF en appliquant les règles proposées par Tong et al. [164].

```
{
<rdfs:Class rdf:ID = "Target"/>
<rdf:Property rdf:ID = "target_name">
      <rdfs:domain rdf:resource = "#Target"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "target_description">
      <rdfs:domain rdf:resource = "#Target"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdfs:Class rdf:ID = "Device"/>
```

```
<rdf:Property rdf:ID = "device_id">
      <rdfs:domain rdf:resource = "#Device"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "device_model">
          <rdfs:domain rdf:resource = "#Device"/>
          <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property
<rdf:Property rdf:ID = "device_location">
          <rdfs:domain rdf:resource = "#Device"/>
      <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property>
<rdfs:Class rdf:ID = "Profile"/>
<rdf:Property rdf:ID = "profile_name">
      <rdfs:domain rdf:resource = "#Profile"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "profile_owner">
      <rdfs:domain rdf:resource = "#Profile"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "profile_description">
      <rdfs:domain rdf:resource = "#Profile"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "pby">
          <rdfs:domain rdf:resource = "#Profile"/>
          <rdfs:range rdf:resource = "#Target"/>
</rdf:Property>
<rdf:Property rdf:ID = "pof">
          <rdfs:domaine rdf:resource = "#Device"/>
          <rdfs:range rdf:resource = "#Device"/>
</rdf:Property>
<rdfs:Class rdf:ID = "Sensor"/>
<rdf:Property rdf:ID = "sensor_name">
      <rdfs:domain rdf:resource = "#Sensor"/>
          <rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "sensor_variale">
          <rdfs:domain rdf:resource = "#Sensor"/>
          <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property
```

```
<rdf:Property rdf:ID = "sensor_description">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property>
<rdf:Property rdf:ID = "sensor_val_min">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:float" />
</rdf:Property>
<rdf:Property rdf:ID = "sensor_val_max">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:float" />
</rdf:Property>
<rdf:Property rdf:ID = "sensor_unit">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property>
<rdf:Property rdf:ID = "sensor_accuracy">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:float" />
</rdf:Property>
<rdf:Property rdf:ID = "sensor_accuracy_unit">
          <rdfs:domain rdf:resource = "#Sensor"/>
      <rdfs:range rdf:resource = "#xsd:string" />
</rdf:Property>
<rdfs:Class rdf:ID = "associate"/>
<rdf:Property rdf:ID = "ads">
          <rdfs:domain rdf:resource = "#associate"/>
          <rdfs:range rdf:resource = "#Devicer"/>
</rdf:Property>
<rdf:Property rdf:ID = "asd">
          <rdfs:domaine rdf:resource = "#associate"/>
          <rdfs:range rdf:resource = "#Sensor"/>
</rdf:Property>
}
```

Cette traduction de notre diagramme de classes sous forme d'ontologie simple (RDFS) permet de générer la description RDF des profils expérimentaux et jette les bases des futurs développements sémantiques. De plus cette structure contenant également les intervalles de variation des mesures de chacun des senseurs permet de vérifier que les donnes transmises sont cohérentes avec les possibilités techniques du capteur, estimer l'erreur globale des valeurs calculées par le modèle.

6.6 Conclusion

Les métadonnées descriptives sont à ce stade uniquement utilisées pour la description des conditions expérimentales liées à une expérience donnée. Le format RDF permet d'associer les capteurs à un périphérique (*Device*), un périphérique à une cible (*Target*) et de décrire notamment la localisation du périphérique sur la cible. Ces informations sont nécessaires si l'on veut pouvoir tirer des conclusions valides à partir des mesures effectuées par les capteurs et notamment apprécier la qualité des données en vérifiant que les données transmises sont conformes à l'intervalle de validité des mesures.

Dans le futur, ces données sémantiques pourront servir à l'élaboration d'ontologies, c'est-à-dire la création d'un modèle de données dans une ontologie pour l'extraction de connaissances afin de réaliser des prédictions sur bases des données collectées.

Actuellement, les données sémantiques sont envoyées sous forme d'un topic séparé dans le bus de messages d'Apache Kafka. Lorsqu'une variation dans la sémantique des données est détectée, un nouveau pipeline est généré au niveau d'apache Beam ainsi qu'un nouveau schéma d'ingestion de données au niveau d'Apache Druid.

7

Elaboration d'une architecture pour le déploiement de micro-services et d'algorithmes d'intelligence artificielle

Plan du chapitre

Résumé

Dans ce chapitre, nous étendons notre architecture cloud originale au "Edge Computing" pour permettre le déploiement de micro-services et d'algorithmes d'intelligence artificielle adaptés. Ces capacités de traitement locales s'appuient sur l'architecture cloud LAMA conceptualisée, développée et testée dans les chapitres précédents. Le Edge Computing nous permet de nettoyer, valider et prétraiter les données avant leur envoi dans le cloud à l'aide des micro-services, d'extraire des caractéristiques des données (features) ou encore réaliser des prédictions. Nous testons ensuite notre architecture Edge sur deux cas d'applications : (1) L'identification des maladies des plantes à partir d'images ; (2) La création de double numérique (Digital Twin) de vaches à partir de vidéos.

Publications liées à ce chapitre

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, F. Lebeau, "A new Edge Architecture for AI-IoT services deployment". In : The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC). *Procedia Computer Science*, 2020, **175** : 10-19. doi : 10.1016/j.procs.2020.07.006.

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, J. Bindelle, F. Lebeau, "Edge Computing and Artificial Intelligence for Real-time Poultry Monitoring". In : The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC). *Procedia Computer Science*, 2020, **175** : 534-541. doi : 10.1016/j.procs.2020.07.076.

M. Elmoulat, **O. Debauche**, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, F. Lebeau, "Edge Computing and Artificial Intelligence for Landslides Monitoring". In : *Procedia Computer Sicence*, 2020, **177** : 40-48. doi : 10.1016/j.procs.2020.10.066.

O. Debauche, S. Mahmoudi, M. Elmoulat, S.A. Mahmoudi, P. Manneback, F. Lebeau, "Edge AI-IoT Pivot Irrigation, Plant Diseases and Pests Identification". In : "The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)". *Procedia Computer Sicence*, 2020, **177** : 480-487. doi : 10.1016/j.procs.2020.10.009.

M. Elmoulat, **O. Debauche**, S. Mahmoudi, S.A. Mahmoudi, A. Guttadauria, P. Manneback, F. Lebeau, "Towards Landslides Early Warning System With Fog - Edge Computing And Artificial Intelligence", *Journal of Ubiquitous Systems & Pervasive Networks*, 2021 : **15(1)**, 11-17. doi : 10.5383/JUSPN.15.02.002.

N. Tadrist, **O. Debauche**, S. Mahmoudi, A. Guttadauria, "Towards Low-Cost IoT and LPWAN-Based Flood Forecast and Monitoring System", *Journal of Ubiquitous Systems & Pervasive Networks*, 2022, **17(1)** : 43-49. doi : 10.5383/JUSPN.17.01.006.

O. Debauche, S. Mahmoudi, A. Guttadauria, "A New Edge Computing Architecture for IoT and Multimedia Data Management", *Information*,2022, **13(2)**, 89. doi : 10.3390/info13020089.

7.1 Introduction

Différentes stratégies ont été proposées pour réduire la charge du cloud et diminuer les besoins en bande passante en conciliant les besoins en matière de traitement de données sur les appareils mobiles ou sur les ordinateurs des utilisateurs. Les approches dites "post-cloud" sont apparues après le traitement centralisé sur le cloud en vue d'améliorer la latence et la fluctuation de celle-ci (*Jitter*) pour les entités immobiles mais ne fournissent pas de réponse adaptée aux appareils mobiles et à la prise de décisions (*Awarness*) locales. Les ressources Edge permettent de fournir des prévisions plus rapides par rapport à l'utilisation des ressources cloud. Un compromis entre l'utilisation rationnelle du cloud pour l'apprentissage des modèles sans transfert de l'intégralité les données et le déploiement en périphérie du réseau permet de trouver un équilibre entre la quantité de données à transférer, l'utilisation de la bande passante, la conservation des performances des modèles.

Dans ce contexte, les techniques de *Edge Computing* peuvent être utilisées pour résoudre les problèmes de latence, de fluctuation (*Jitter*) et diminuer le traitement de la charge sur le cloud en rapprochant le traitement du niveau producteur et/ou consommateur des données. De plus, la combinaison de l'Internet des objets (IoT) et de l'Intelligence Artificielle (IA) en bordure du réseau ouvrent le champs à différentes applications telles que le traitement d'images, la reconnaissance faciale, le traitement vidéo, la segmentation d'objets, le suivi d'objets, etc.

Notre objectif est de compléter notre architecture LAMA en fournissant un système automatisé pour le déploiement dynamique des micro-services ou de réseaux de neurones sur un cluster situé à la périphérie du réseau en vue de décharger l'architecture cloud notamment des opérations de prétraitement des données.

7.2 Design de la partie Edge Computing

L'architecture cloud LAMA que nous proposons outre les fonctions de collecte, de traitement et de stockage en base de données se voit, avec l'ajout de cette nouvelle partie, dotée de nouvelles fonctionnalités nécessaires à la collaboration avec le Edge Computing. Le déploiement et la gestion de conteneurs sont des fonctions clés qui permettront la distribution de micro-services et d'algorithmes d'intelligence artificielle adaptés aux plus près des capteurs. L'entraînement des algorithmes d'intelligence artificielle se fera sur le cloud qui dispose de capacités de calcul importantes ainsi que des GPU pour l'accélérer. Le traitement de la sémantique nécessite de disposer d'une base de connaissances, d'une dépôt d'ontologies contenant des ontologies de domaines et spécialisées. Un moteur de raisonnement sémantique permet de tirer des conséquences et des conclusions en fonctions des connaissances accumulées, des règles ontologiques et des données d'entrée transmises par les capteurs.

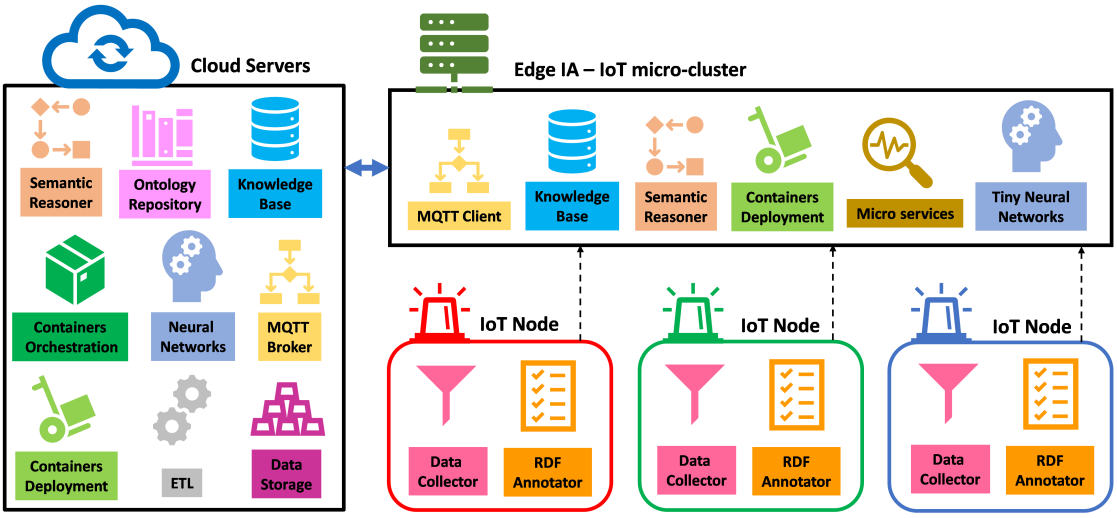


FIGURE 7.1 – Schéma à haut niveau de l'architecture Edge AI-IoT

L'architecture Edge AI-IoT associant les capacités de traitement en périphérie du réseau pour : (**1**) Le traitement des données provenant des nœuds avant leur envoi (éventuel) vers le cloud ; (**2**) Le raisonnement et la prise de décisions rapide et locales à l'aide de la sémantique et/ou de l'intelligence artificielle ; (**3**) L'entraînement en continu d'algorithmes d'intelligence artificielle adaptés au ressources limitées disponibles au niveau du edge ; (**4**) Le déploiement de micro-services liées à l'exploitation des données collectées et aux possibilités d'effectuer des actions au niveau des nœuds.

Comme le montre la Figure 7.1, la communication entre l'architecture cloud et son pendant en bordure du réseau s'effectue à l'aide d'un système de messagerie et d'un courtier (*Broker*). Tandis que les nœuds IoT communiquent généralement en utilisant des protocoles de communication sans fil.

7.3 Choix technologiques

Le déploiement des micro-services peut se faire soit à l'aide de machines virtuelles ou de containers. Les machines virtuelles offrent un meilleure isolation grâce à l'utilisation d'un système d'exploitation propre à la machine virtuelle. Tandis que les conteneurs sont plus légers car il n'héberge que les couches logicielles nécessaires à l'exécution du programme ou du modèle. Les conteneurs étant plus légers que les machines virtuelles, il est possible d'en déployer d'avantages à ressources égales. Les ressources au niveau Edge (périphérie du réseau) étant contraintes, c'est la technologie de conteneurisation qui sera privilégiée.

Parmi les systèmes d'orchestration de containers, Kubernetes fait office de référence, il a été développé spécifiquement pour le déploiement de cluster de plusieurs milliers de nœuds pour former des clouds privés, publics ou hybrides. Micro Kubernetes (*micro k8s*) est une plateforme d'orchestration de conteneurs open source léger qui gère les charge de travail (*worflow*) et les services conteneurisés au niveau du Edge avec la possibilité de gérer des conteneurs GPU Nvidia. Kubernetes et Micro Kubernetes sont conçus pour fonctionner ensemble et être déployés respective au niveau du cloud et du Edge. Tous deux utilisent la notion de "pods" qui sont des unités de déploiement contenant un ou plusieurs conteneurs. k3s est une version allégée de k8s spécifiquement destinée pour l'IoT en tant que projet de la Cloud Native Computing Foundation. Il revêt la forme d'une binaire de 40Mo, ne nécessitant que 512Mo de mémoire et est adapté pour fonctionner sur ARMv7 et ARM64. k3s et micro k8s peuvent fonctionner sur Raspberry Pi, Nvidia Jetson Nx et Nano en théorie. Dans la pratique, micro k8s ayant une emprunte mémoire importante pour des nœuds disposant d'une faible quantité de mémoire, nous lui avons préféré k3s qui offre outre sa légèreté, un niveau de sécurité suffisant pour mettre en œuvre un cluster au niveau Edge sur des périphériques contraints. Fondamentalement, k3s se distingue de k8s par le remplacement le remplacement de etcd par SQLite au niveau du nœud maître, l'ajout d'un proxy tunnel qui sécurise la connexion entre le nœud maitre et chaque nœud de travail et le remplacement de l'interface réseau de conteneurs (CNI) par un composant plus léger appelé "flannel".

En fonction de la nature des tâches à réaliser, des cas d'applications et des contraintes qui en dépendent guident le choix des nœuds de calcul. Par exemple : Pour le traitement d'images ou de flux vidéo on pourra se contenter de nœuds de calcul avec de nombreux cœurs. Si l'on veut utiliser des algorithmes d'IA préentrainés des nœuds à base de GPU seront préférés pour ce type d'applications.

Un cluster de nœuds comprenant différents types de nœuds permet d'adresser la plupart de besoins au niveau du Edge. Par ailleurs, en cas de besoin des calculs fortement parallélisés pourront également être exécutés sur les CPU des nœuds comportant un GPU additionnel et normalement réserver aux algorithmes d'intelligence artificielle. De même, les algorithmes d'intelligence artificielle pourront également s'exécuter sur des nœuds à bases de CPU mais plus lentement. Le défi étant alors d'affecter de préférences les tâches en fonction de leur nature sur le type ressource la mieux adaptée en fonction de la charge de leur disponibilité et à défaut de leur charge (ressources CPU/GPU et mémoire utilisées).

7.4 Intégration avec l'architecture LAMA

L'orchestrateur Rancher est utilisé pour déployer des containers k3s au niveau du Edge. Au niveau du cloud, le serveur Rancher est déployé dans un container docker sur Apache Mesos. Au niveau du Edge, l'agent Rancher déployé au niveau du cluster Edge

assure la communication avec le serveur Rancher déployé dans le cloud. Le cluster Edge est géré par un nœud k3s maître communique d'une part avec l'agent Rancher et l'agent k3s installé sur chaque nœud de travail k3s. La structure et le fonctionnement du cluster Edge sont détaillés dans le paragraphe suivant.

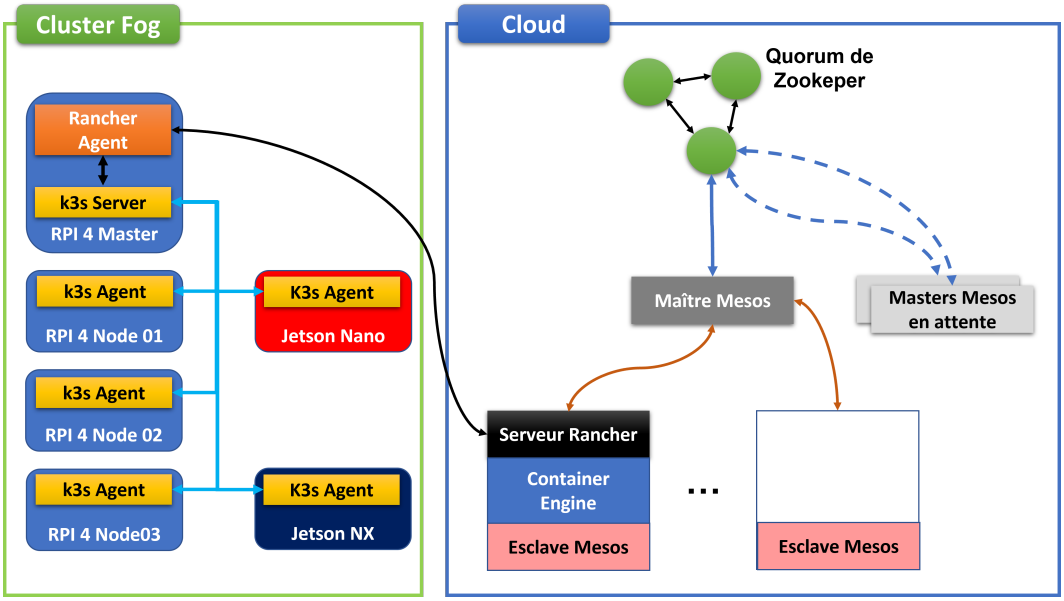


FIGURE 7.2 – Interaction entre l'architecture LAMA et le mini cluster Edge

Comme le montre la Figure 7.2, l'orchestrateur Rancher est utilisé au niveau du cloud avec Rancher Kubernetes Engine (RKE2). Il est aussi utilisé pour l'orchestration des cluster Edge où k3s est utilisé pour le déploiement des conteneurs.

7.5 Mise en œuvre

Pour la mise en œuvre de notre mini cluster expérimental nous avons choisi d'utiliser des cartes à base d'ARM et de GPU Nvidia car elles sont supportées par k3s et offrent le meilleur rapport caractéristiques (puissance, mémoire) / prix.

7.5.1 Matériel

Le couplage de Nvidia Jetson Nano / NX développés spécifiquement pour l'intelligence artificielle embarquée avec des Raspberry Pi 4 8Go. Cette association de nœuds permet de créer un mini cluster hétérogène. Ce cluster est utilisé pour déployer d'une part des micro-services et d'autre part des algorithmes d'intelligence artificielle spécifiquement adaptés pour le Edge Computing, au moyen de Kubernetes.

FIGURE 7.3 – Micro-cluster Edge

L'architecture Edge AI-IoT a été déployé sur un ensemble interconnectés de périphériques regroupés en trois catégories :

1. Un Jetson Nano (472 GFlops FP16) dotés de 128 cœurs CUDA Maxwell, d'un processeur ARM A57 quadruple cœur cadencés à 1,43 GHz et munis de 4GB LPDDR4 64 bits à 25,6 GB/s. Le Jetson Nano utilise JetPack 4.6 installé sur une carte de 32 GB Micro SDHC UHS-3. Le JetPack contient NVIDIA L4T, TensorRT, cuDNN, CUDA, OpenCV. L'objectif du nœud Jetson Nano est d'entraîner des modèles d'intelligence artificielle sur les données IoT collectées à des fins de test.
2. Un Jetson Xavier NX (21 TeraOPS (TOPS) dotés de 384 cœurs CUDA Volta et 48 cœurs Tensor, d'un processeur CPU NVIDIA Carmel ARM® 64-bit version 8.2 à 6 cœurs 6 Mo L2 + 4 Mo L3 et munis de 8 Go 128-bit LPDDR4x à 51,2 Go/s. Le Jetson Nano est équipé d'une disque dur SSD M.2 de 512Go. Le Jetson Xavier NX utilise également le JetPack 4.6 installé sur une carte de 32 GB Micro SDHC UHS-3. Le nœud Jetson Xavier NX est quant à lui utiliser en production.
3. Quatre nœuds Raspberry Pi 4 intégrant un Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz. Un des Raspberry Pi 4 sert de nœud maitre et gère l'ensemble

du mini cluster. Les trois autres Raspberry Pi sont des nœuds esclaves dévolus à l'hébergement de conteneurs. Le nœud maître est équipé d'un disque dur SSD de 1To et les nœuds esclaves de disques SSD de 500 Go. L'ensemble des Raspberry Pi 4 boot sur SSD ce qui permet d'obtenir une vitesse de lecture moyenne de 232 mo/s au lieu de 22 mo/s sur carte SD mesurée avec hdparm.

4. Un switch 8 ports GigaEthernet permet d'interconnectés les différentes nœuds de ce cluster hétérogène (Jetson Nano - Jetson Xavier NX et les Raspberry Pi 4).
5. Un routeur VPN permet d'attribuer de manière statiques à l'ensemble des éléments du cluster et de le rendre accessible de l'extérieur via un tunnel crypté.

7.5.2 Logiciels

Rancher est utilisé pour l'orchestration tandis que k3s est utilisé pour le déploiement et l'organisation des conteneurs en pods. La Figure 7.4 montre l'agencement des différentes composantes de k3s au sein du micro-cluster.

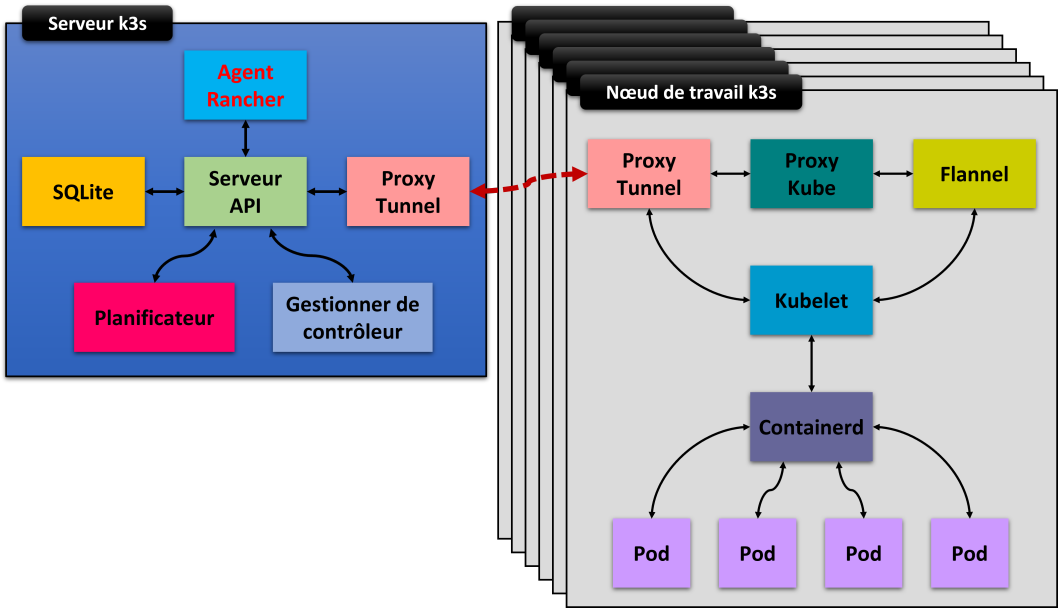


FIGURE 7.4 – Kubernetes dans le micro-cluster

Le **nœud principal** (Master Node), un Raspberry Pi 4, est responsable de la répartition de la charge de travail des applications, de la planification ainsi que de la détection et de la gestion des modifications de l'état des applications déployées. Le nœud maître est également chargé d'affecter l'application à un nœud choisi en fonction de ses besoins. Il existe deux façons de configurer notre nœud : la première, en utilisant le fichier de configuration du pod pour décrire le nœud qui sera chargé de planifier l'application.

La deuxième manière, en utilisant la ligne de commande et en spécifiant le libellé du nœud spécifique.

Les services installés sur le nœud maître sont : (**1**) SQLite, pour la persistance et la maintenance des informations statistiques sur les différents composants du cluster k3s. SQLite est utilisée à la place de etcd qui est habituellement utilisé par k8s, mais est trop gourmand en mémoire pour fonctionner dans un environnement à mémoire contrainte ; (**2**) un serveur API expose les points de terminaison pour toutes les interactions avec et dans le cluster k3s ; (**3**) un planificateur planifie en fonction des besoins en ressources d'application et des contraintes d'affinité spécifiques quel(s) pod(s) d'application doivent s'exécuter sur le ou les nœuds de travail sélectionnés du cluster k3s ; (**4**) un gestionnaire de contrôleur ; (**5**) un proxy de tunnel qui gère et maintient le tunnel de connexion entre le nœud maître et chacun des nœuds de travail ; ((5) Le "Flannel" remplace l'Interface réseau de conteneurs de k8s (CNI) et permet l'interconnexion des nœuds de travail entre eux.

Les **nœuds de travail** (Worker Nodes) mélangeant Raspberry Pi 4, Nvidia Jetson Nano et Nvidia Xavier NX sont composés de : (**1**) Kubelet, un agent qui s'exécute sur chaque travailleur (*Worker*) de k3s. Il crée et démarre un module d'application sur le travailleur et surveille l'état de santé de ce dernier et de tous les modules en cours d'exécution sur le nœud maître via le serveur API ; (**2**) Kube-proxy, un proxy de réseau qui est un point d'entrée permettant un accès à divers points d'extrémité de service d'application et achemine une demande vers les pods appropriés du cluster ; (**3**) Containerd gère le cycle de vie des conteneurs tels que l'obtention des images, le démarrage et l'arrêt des conteneurs, etc., (**4**) Le proxy tunnel maintient la liaison entre nœud maître et le nœud de travail.

Les **Nœuds IoT** réalisent la collecte des données à partir des capteurs qui y sont attachés et effectuée également l'annotation des données avec un annotateur RDF.

7.6 Tests et validation

Nous avons ensuite testé et validé notre complément architectural au niveau du Edge à partir deux cas d'application. Le premier à pour objectif d'identifier les maladies des plantes à partir d'images de feuilles issues de différentes cultures. Le second cas d'application consiste à produire des doubles numériques à partir de vidéo de vaches au champ.

7.6.1 Identification des maladies des plantes

Les premiers tests réalisés pour objectif de mettre en évidence le ratio entre la prédiction dans le cloud avec celui des solutions embarquées (Jetson Nano).

La base de données PlantVillage[165] a été utilisées pour expérimentations. Elle contient 54.303 images de feuilles malades et non malades ventilées en 38 catégories représentant une maladie pour une espèces donnée. La base de données brutes (sans prétraitement) a été entraînée avec les algorithmes MobileNet[166], MobileNetV2[167], DenseNet121[168], NasNetMobile[169], EfficientNetB0[170], EfficientNetB1[170], EfficientNetB2 [170] dans le cloud sur un serveur équipé d'une Tesla P100. Tous les modèles ont été entraînés avec l'optimiseur sgd avec un taux d'apprentissage (learning rate) de 10^{-4}, un arrêt précoce (*Early Stopping*) avec une patience de 5 et un delta minimum de 10^{-3} sur la précision de la validation (validation accuracy - val_acc). Les modèles entraînés ont été convertis en Tensorflow Lite et conteneurisés pour être déployé sur Jetson Nano en vue d'évaluer leur performances sur la prédiction d'une maladie de plante à partir d'une image non traitée.

La Figure 7.5 montre une comparaison entre les tailles des fichiers des modèles standards entraînés (fichier h5) et les fichiers compressés (TFlite) obtenus par pruning.

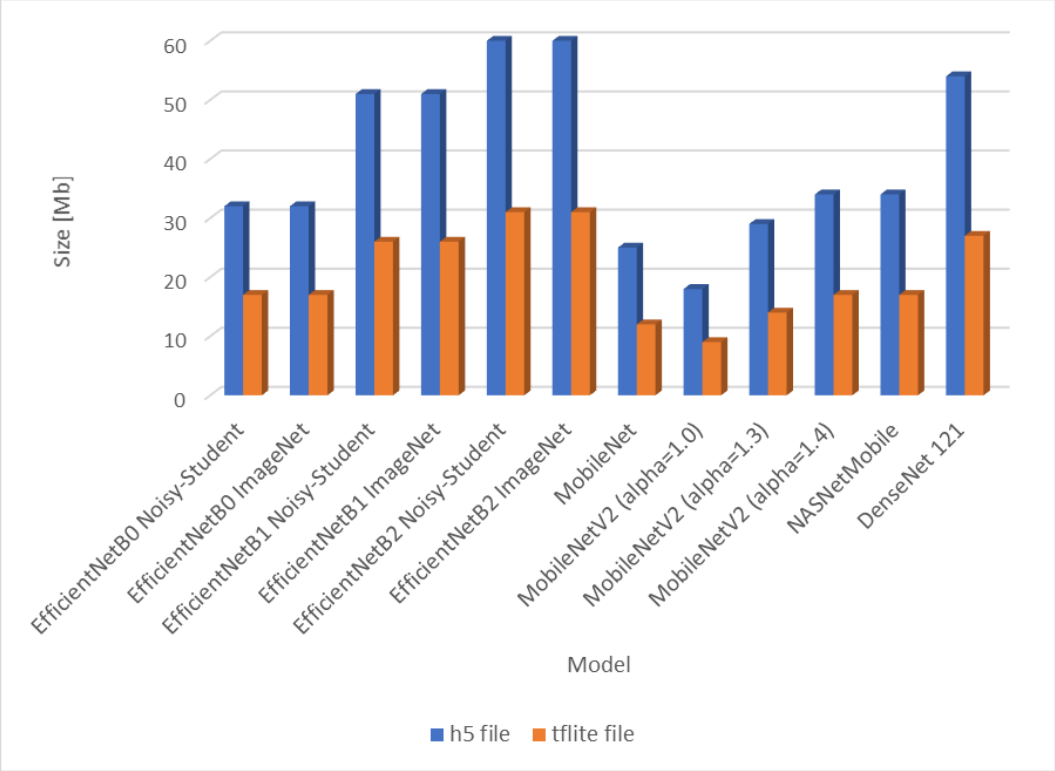


FIGURE 7.5 – Comparaison de taille des fichiers h5 et tflite

La Figure 7.6 présente également le temps de test de validation exprimé en secondes et la précision obtenue avec le modèle h5 sur le cloud et le modèle TFlite sur Jetson Nano. Notre test de prédiction a été réalisé sur 443 images. Finalement, le Jetson Nano

est 28 fois plus lent que le serveur cloud équipé d'une Tesla P100. L'analyse des résultats montrent que les modèles MobileNetV2 et MobileNet s'appuie sur des fichiers .h5 avec les tailles les plus faibles.

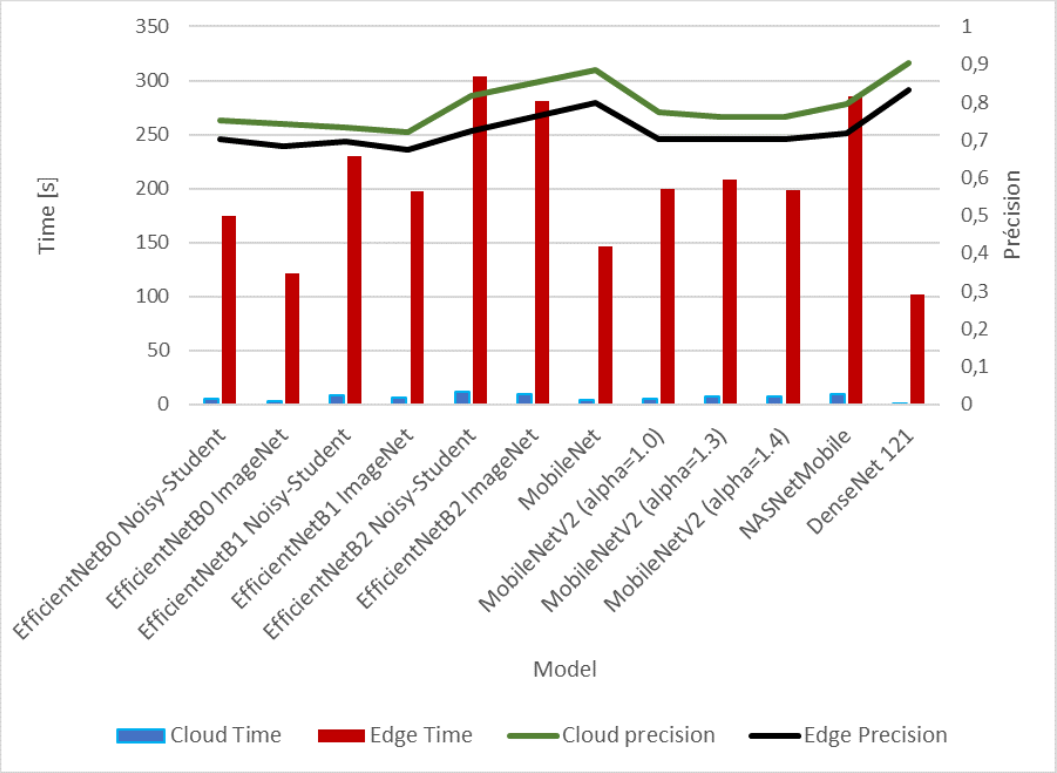


FIGURE 7.6 – Comparaison des temps de prédiction et des précisions

L'analyse de la précision modèles avant et après compression illustré par les courbes noires et vertes sur la Figure 7.6 montre que meilleurs niveaux de précision sont obtenus par MobileNet et DenseNet121 aussi bien sans qu'avec compression par pruning. Les meilleurs temps de prédiction sont obtenus par DenseNet121, EfficientNetB0, et MobileNet. L'analyse croisée de la taille et des performances montre que le modèle MobileNet offre à la fois une faible taille de modèle et un niveau de performances après compression parmi les plus élevés.

7.6.2 Elaboration de doubles numériques de bovins

Nous avons ensuite implémenté un deuxième cas d'application consistant à créer des doubles numériques des bovins en champs. Un modèle top-down a été élaboré pour localiser et ensuite identifier les points clés nécessaires à l'élaboration du squelette du double numérique (Digital Twin). Au départ des vidéos au champ, le modèles

Yolo v3 spécifiquement entrainé avec les vaches, les moutons et les chevaux à partir des images annotées issues de la base de données Animal Pose [171] localise les animaux. La localisation se matérialise sous la forme d'un rectangle vert circonscrivant l'animal sur la Figure 7.7. Les squelettes se matérialisent sous la forme de 20 points caractéristiques : les deux yeux, la gorge, le nez, le garrot, les deux bases d'oreilles, la base de la queue, les quatre coudes, les quatre genoux, les quatre pattes.

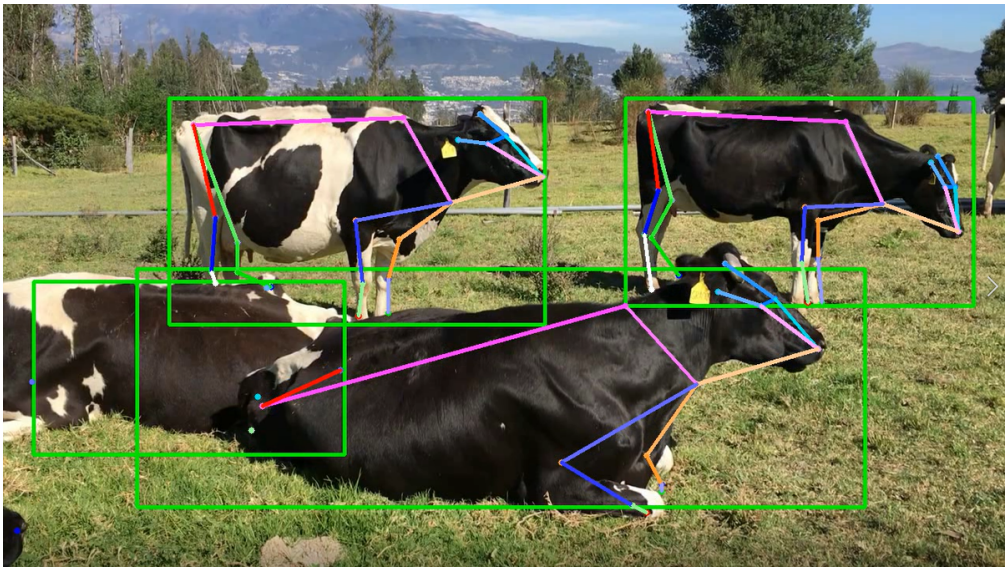


FIGURE 7.7 – Localisation avec superposition des doubles numériques

Le modèle a été testé sur une vidéo avec un seul animal et sur des vidéos de troupeau comme l'illustre la Figure 7.7. Le vitesse de traitement du flux vidéo est de 0,5 frame par seconde.

A ce stade, seul Yolov3 a été testé, des versions plus récentes du modèle ont vu le jour ainsi que des variantes de ces nouvelles versions qu'il serait intéressant de tester afin de sélectionner la version offrant le meilleur rapport précision/vitesse de prédiction. Il en est de même pour le modèle qui est utilisé pour la création des doubles numériques, d'autres modèles alternatifs existent et devront être évaluer pour trouver la bonne association de modèle qui offre le meilleur compris précision / vitesse.

7.7 Conclusion

Dans ce chapitre, nous avons développé un complément architectural au niveau Edge pour l'architecture LAMA. Ce complément d'architecture a pour objectif de déployer des algorithmes de traitement et des algorithmes d'intelligence spécifique-

ment développés et/ou adaptés pour fonctionner sur des périphériques à ressources contraintes en matière de traitement, de mémoire et de capacité de stockage. L'architecture logicielle s'appuie sur k3s un orchestrateur Kubernetes allégé pour obtenir une emprunte mémoire réduite et fonctionner sur les architectures Edge. Rancher, un outil de gestion de cluster Kubernetes est utilisé pour l'administration du cluster Edge à partir du cloud. L'association Rancher avec k3s permet de déployer des conteneurs docker en pods au niveau Edge avec un solution logicielle optimisée pour les architecture ARMv7, ARM64 et x86 de manière aisée au départ d'un fichier yaml.

Troisième partie

Compléments architecturaux

8

Service pour la recherche en Smart Farming

Plan du chapitre

Résumé

Ce chapitre décrit le service pour permettre d'analyser les comportements de leurs animaux de ferme tout en gardant le contrôle de leur données.

L'analyse des comportements des animaux à partir des données inertielles acquises à l'aide de colliers placés sur le cou des bovins permet d'en déduire les comportements après calibration à partir des vidéos annotées de ces bovins. Cette plateforme offre un outil important aussi bien pour les éleveurs qui peuvent faire une analyse du comportement de leurs animaux, que pour les chercheurs qui peuvent analyser des quantités importantes de données.

Publications liées à ce chapitre

O. Debauche, S. Mahmoudi, A.L.H. Andriamandroso, P. Manneback, J. Bindelle, F. Lebeau, "Web-based cattle behavior service for researchers based on the smartphone inertial central," *Procedia Computer Science*, 2017, **110** : 110-116. doi : 10.1016/j.procs.2017.06.127.

O. Debauche, S. Mahmoudi, A.L.H. Andriamandroso, P. Manneback, P. Bindelle, F. Lebeau, "Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors," J Ambient Intell Human Comput, 2019, **10**, 4651–4662. doi : 10.1007/s12652-018-0845-9.

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, J. Bindelle, F. Lebeau, "Edge Computing for Cattle Behavior Analysis". In Second international conference on Embedded & Distributed Systems (EDiS'2020), Oran, Algeria, 2020. doi : 10.1109/EDiS49545.2020.9296471.

O. Debauche, M. Elmoulat, S. Mahmoudi, J. Bindelle, "Farm Animals' Behaviors and Welfare Analysis with AI Algorithms : A Review" *Revue d'Intelligence Artificielle*, 2021, **35(3)** : 243-253. doi : 10.18280/ria.350308.

8.1 Service d'analyse comportementale

Un service a été créé pour donner accès à d'autres équipes de recherche et aux agriculteurs aux outils développés pour analyser le comportement des animaux. Ce service permet différents types d'analyse au départ d'un fichier csv contenant les données de la centrale inertielle d'un iPhone échantillonnées à une fréquence de 100Hz. Au terme de l'analyse, il est possible d'exporter les données en différents formats (xls, csv, odt, shp, gpx) pour en faciliter leur exploitation. Les services proposés sont : (**A**) L'analyse du comportement à partir des données d'une centrale inertielle ; (**B**) L'analyse des classes de déplacements et de vitesse des bovins dans le pâturage ; (**C**) La production de données cartographiques ; (**D**) L'élaboration d'un rapport de synthèse.

8.1.1 L'analyse du comportement à partir des données d'une centrale inertielle

Les macro-comportements des bovins sont déterminés sur base des données produite par une centrale inertielle à haute fréquence (100 Hz) et sauvegardées sous forme csv. Ces données sont traitées de manière statistique pour déterminer les paramètres utiles pour déterminer les comportements à l'aide de l'arbre décisionnel décrit dans Andriamandroso et al. [126].

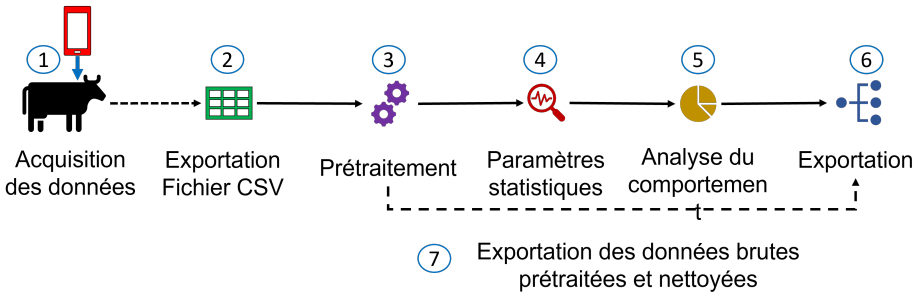


FIGURE 8.1 – Chaîne de traitement pour l'analyse du comportement.

Comme le montre la Figure 8.1, un iPhone est placé sur le cou d'un bovin (**1**) pour une durée variant de quelques heures à quelques jours. L'iPhone est pourvu d'une centrale inertielle à 9 degrés de libertés et d'un GPS. Ces données sont acquises à 100 Hz pour la centrale inertielle, à 0,5 Hz pour le GPS et sauvegardées au format csv. Les données sont ensuite téléchargées de l'iPhone 5S (**2**), ensuite parsées et prétraitées (**3**) pour permettre le calcul des paramètres statistiques, sur un intervalle d'une seconde correspondant en moyenne à 100 mesures, de la moyenne de l'accélération gravitationnelle selon l'axe des X (mGx), de l'écart type de l'accélération gravitationnelle selon l'axe des X (sGx) et de l'écart type du taux de rotation selon l'axe des Y (sRy) (**4**) nécessaires à l'analyse des macro-comportements (**5**) selon l'algorithme proposé par Andriaman-

droso et al. [126]. Le résultat de l'analyse comportementale consiste en la production d'un fichier csv, odt (Open Office) ou xls (Microsoft Excel) horodaté donnant à chaque seconde le comportement déterminé de l'animal (**6**). De manière optionnelle, les données brutes peuvent également être exportée en odt ou xls (**7**).

8.1.2 L'analyse des classes de déplacements et de vitesse

En complément de l'analyse des macro-comportements, une analyse des déplacements et des classes de vitesses de déplacement permet d'identifier les zones de repos, de rumination, de broutage et les couloirs de déplacement des animaux au pâturage (Figure 8.2).

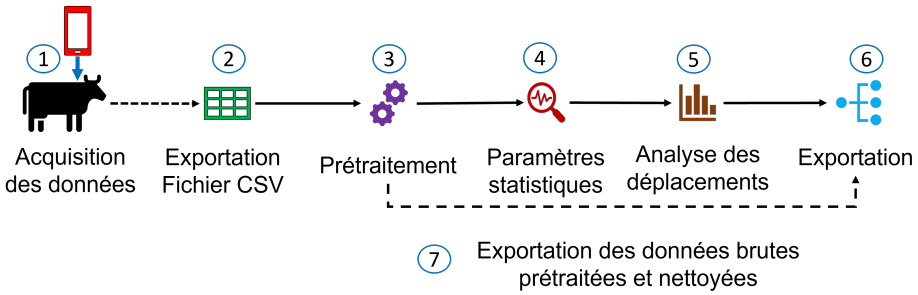


FIGURE 8.2 – Analyse des déplacement et classes de vitesse.

La Figure 8.2 montre les différentes étapes de l'analyse des déplacements et des classes de vitesse des bovins à partir d'un log des positions GPS échantillonnées à 0,5 Hz. Les données GPS d'un iPhone placé sur le cou d'un bovin sont loggées et sauvegardées dans un fichier csv (**1**). Elles sont téléchargées (**2**), ensuite parsées et prétraitées (**3**) pour permettre le calcul du déplacement et de la vitesse entre deux intervalles de temps. La valeur est par défaut est d'une seconde ce qui correspond à des échantillons fluctuant entre 98 à 102 valeurs (**4**). Les limites de classe proposées par défaut et exprimées en m/s sont : [0 ;0,1],]0,1 ;0,2],]0,2 ;0,3],]0,3 ;0,4],]0,4 ;0,5], [0,5 ;→[. Une analyse des différentes classes de déplacement et de vitesse à l'aide de la statistique descriptive est réalisée pour en extraire les valeurs moyennes et écart-type, minimales, maximales, médianes et quartiles (**5**). Le résultat de l'analyse des parcours et des vitesses de déplacement est la production d'un fichier csv, odt (Open Office) ou xls (Microsoft Excel) horodaté donnant à chaque seconde le comportement déterminé de l'animal (**6**). De manière optionnelle, les données brutes peuvent également être exportée en odt ou xls (**7**).

8.1.3 La production de données cartographiques dérivées

Les analyses effectuées sur les macro-comportements (**A**), les déplacements et les vitesses (**B**) permettent de produire des représentations cartographiques telles que la localisation sous forme de coordonnées de chaque classe de comportement, la représentation des classes de déplacement ou de vitesse sous forme de segments, la carte des points chauds. L'ensemble des couches de données cartographiques sont générées aux formats ESRI Shapefile (shp), Google kmz et GPX. Le format shapefile a été choisi car il est communément utilisé et largement supporté par les systèmes d'informations géographiques, et le Google kmz est directement visualisable sur Google Earth. Le format de fichier GPX permet de l'importer dans un GPS de terrain ou de l'exploiter par une exploitation sur Smartphone pour identifier par exemple sur le terrain les zones qui ont été pâturées.

— Les cartes de comportements (*Behavior Maps* revêtent la forme d'une couche de points où chacun d'eux représente un coordonnées géographique (latitude, longitude) associée à un comportement sur un intervalle de temps. Une carte des points d'arrêts correspondant à une position fixe d'au moins 5 s de l'animal. La carte des points de repos ne reprend que les points d'arrêt de plus de 15 minutes (Figure 8.3 - 5).
— La carte des classes de déplacements (*Displacement Map*) représente sous forme de segments les classe de déplacements de l'animal et permet d'apprécier visuellement le nombre et la longueur des déplacements. Un allongement des déplacements montre que l'animal doit chercher d'avantage sa nourriture et donc la raréfaction des espèces palatables à sa disposition (Figure 8.3 - 6).
— La carte des classes de vitesses (*Speed Map*) permet de visualiser les vitesses de déplacement de l'animal dans le pâturage. Une diminution de sa vitesse de déplacement habituelle de l'animal pourrait indiquer une boiterie ou une maladie (Figure 8.3 - 7).
— La carte des points chauds (*Heat Map*) permet de localiser les zones où chaque animal à une activité particulière et son intensité en fonction du temps écouler à cet endroit. Il est donc possible de déterminer les zones ou un animal rumine, se repose et les zones préférentiellement broutées (Figure 8.3 - 8).

La Figure 8.3 montre les différentes étapes suivies pour élaborer les différentes cartes. Un iPhone 5S équipé d'une centrale inertielle (*IMU*) à 9 degrés de liberté et d'un GPS, placé sur le cou d'un animal de ferme, acquière les données à une fréquence de 100 Hz et les stock (**1**). Les données sont téléchargées de l'iPhone 5S sous la forme d'un fichier csv horodaté reprenant la date et l'heure, les données de la centrale inertielle et les positions GPS (**2**). Les données contenu dans le fichier csv sont parsées (**3**) ligne par ligne pour déterminer les macro-comportements, analyser les parcours et les vitesses de déplacement (**4**). L'association des macro-comportements et des positions GPS permet d'élaborer la carte des comportements (**5**). Le regroupement des positions entre deux points d'arrêt permet de former des segments qui sont ensuite classés en

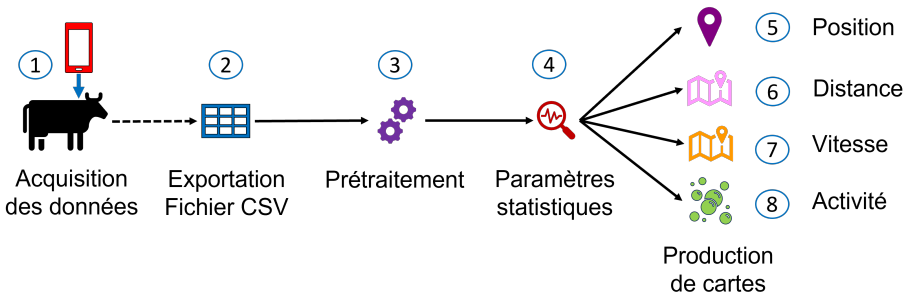


FIGURE 8.3 – Chaîne de traitement pour l'analyse du comportement.

classe de distance parcourues pour former la carte des classes de déplacement. Les classes de déplacement utilisées par défaut exprimées en m/s sont : [0 ;0,1],]0,1 ;0,2],]0,2 ;0,3],]0,3 ;0,4],]0,4 ;0,5],]0,5 ;→[, celles-ci peuvent être personnalisées (**6**). L'analyse des vitesses de déplacement entre deux points permet de classer les vitesses de déplacement en catégories. Les classes proposées par défaut sont les suivantes [0 ;0,3],]0,3 ;0,5],]0,5 ;1],]1 ;2],]2 ;→[et peuvent être personnalisées. Les segments successifs appartenant à une même classe de vitesse permettent d'élaborer la carte des vitesses de déplacement (**7**). Finalement, une carte d'activité est élaborée en fonction des durées de comportement observés. Plus les durées sont longues et plus la pondération des points est importante (**8**). La carte d'activité permet d'identifier les zones où il y a une concentration d'activités comme les points d'eau, les zones de repos, etc.

8.1.4 L'élaboration d'un rapport de synthèse

Un rapport d'analyse individuel pour chaque animal est élaboré sur base de l'analyse comportementale, des parcours et des vitesses de déplacements. Ces indicateurs sont comparés avec les données précédentes du même animal et du troupeau, quand les données sont disponibles.

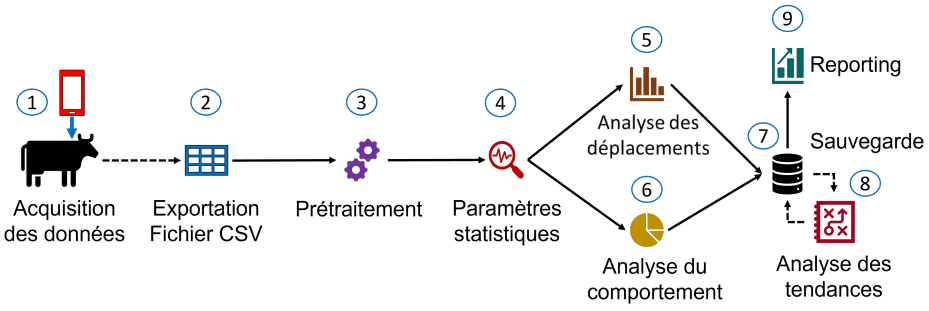
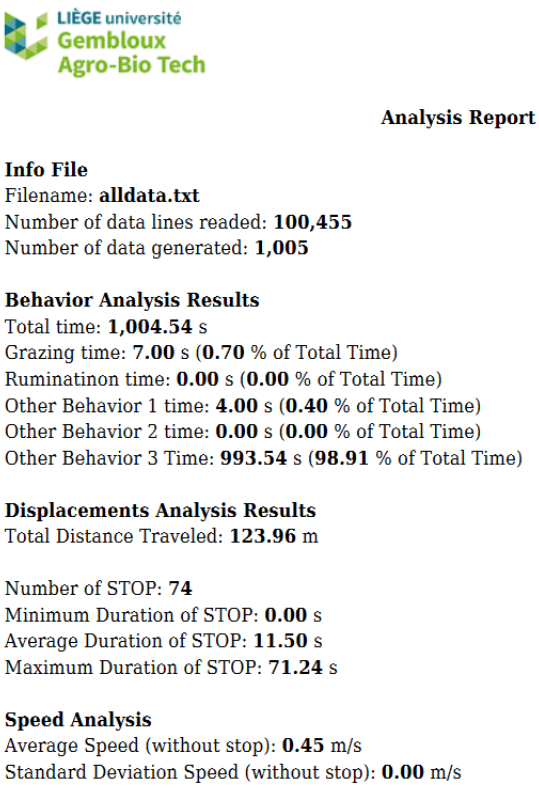


FIGURE 8.4 – Rapport d'analyse comportementale.

Les points 1 à 4 étant commun aux points A et B, ils ne sont pas détaillés, le lecteur qui souhaite davantage de détails est invité à consulter les deux points précédents.

Comme le montre la Figure 8.4, un iPhone acquière les données accélérométriques et GPS à haute fréquences (**1**). Ces données sont téléchargées de l'iPhone 5S sous forme d'une fichier csv (**2**). Les données de ce fichier sont extraites ligne par ligne et stockées pour ensuite permettre la détermination des macro-comportements (**5**) et l'analyse des parcours et vitesse de déplacement (**6**). A l'issue des analyses (**6**) et (**7**), les résultats sont sauvegardés en base de données. Si données du même animal sont disponibles pour les jours précédents et/ou pour le troupeau, les paramètres de l'animal sont comparés avec ceux-ci pour en calculer les tendances (**8**). Finalement un rapport de synthèse est généré reprenant les résultats des différents analyses (**9**).

La Figure 8.5 montre un exemple de rapport obtenu à l'issue de l'analyse d'un fichier log correspondant à une durée de 1004s.


LIÈGE université
Gembloux
Agro-Bio Tech

Analysis Report

Info File
Filename: **alldata.txt**
Number of data lines readed: **100,455**
Number of data generated: **1,005**

Behavior Analysis Results
Total time: **1,004.54** s
Grazing time: **7.00** s (**0.70** % of Total Time)
Ruminatinon time: **0.00** s (**0.00** % of Total Time)
Other Behavior 1 time: **4.00** s (**0.40** % of Total Time)
Other Behavior 2 time: **0.00** s (**0.00** % of Total Time)
Other Behavior 3 Time: **993.54** s (**98.91** % of Total Time)

Displacements Analysis Results
Total Distance Traveled: **123.96** m

Number of STOP: **74**
Minimum Duration of STOP: **0.00** s
Average Duration of STOP: **11.50** s
Maximum Duration of STOP: **71.24** s

Speed Analysis
Average Speed (without stop): **0.45** m/s
Standard Deviation Speed (without stop): **0.00** m/s

FIGURE 8.5 – Extrait de rapport d'analyse comportements, parcours, vitesses

8.1.5 Choix technologiques

La conteneurisation et la virtualisation sont deux moyens de déployer des applications en les rendant indépendant du système d'exploitation sous-jacent. Elles sont utilisées par les DevOps pour déployer rapidement des développements informatiques et/ou les tester sur différentes configurations. La technologie de conteneurisation Docker est supportée par Apache Mesos qui est la brique maîtresse de la plateforme applicative adjointe à l'architecture cloud de collecte, traitement et stockage des données IoT. Afin de permettre la réplicabilité et l'installation du service sur d'autres instances de l'architecture LAMA nous avons développé une image docker. Cette image contient les briques logicielles nécessaires à l'hébergement et au bon fonctionnement du service.

8.1.6 Mise en œuvre

Le service d'analyse comportementale s'appuie sur l'architecture cloud (LAMA) développée au chapitre 4, mise en œuvre et testé au chapitre 5. Le service est développé en PHP 8.0 et est hébergé dans la partie applicative de l'architecture. Il permet à l'utilisateur de téléverser ses fichiers qui sont ensuite traités par la branche traitement en lot de l'architecture (*Batch Processing*). Les données y sont mises en forme, transformées et stockées dans la base de données (ETL). Les données stockées sont ensuite traitées pour réaliser les analyses et/ou transformations de format demandées par l'utilisateur et au final produire les fichiers de sortie (Figure 8.6) Le patron de conception comportemental **Chaîne de responsabilité** (*Chain of Responsability*) a été implémenté pour l'extraction (parsage) et le prétraitement des données brutes en faisant circuler les demandes dans un chaîne de gestionnaires (handlers) traitant un format de données spécifique. L'utilisation de ce patron de conception permettra d'étendre aisément le traitement d'autres formats de fichiers et/ou structuration des données. Le patron de conception comportemental **Stratégie** (*Strategy*) est quant à lui utilisé pour la gestion des algorithmes de traitement. Ce patron de conception les rend interchangeables dans la chaîne de traitement. Finalement, le patron de conception de création **Fabrique** (*Factory Method*) facilite l'ajout de nouveaux format de sorties. Le service d'analyse s'adapte à la fréquence d'échantillonnage et au fluctuation dans le nombre de données en comptant les comptant sur chaque interval d'une seconde.

8.1.7 Validation

Afin de valider le fonctionnement correct de l'identification des comportements et de déterminer les performances en matière de vitesse de calcul. Une expérimentation a été menée sur plusieurs fichiers de différentes tailles correspondant à des durées de collecte de données de 2 minutes (3 Ko) à 36h (3 Go) et nombre de paramètres enregistrés variables (15 et 40) ont été traités pour démontrer la souplesse de fonctionnement. Les résultats des analyses ont été comparés à ceux obtenus par en appliquant l'arbre

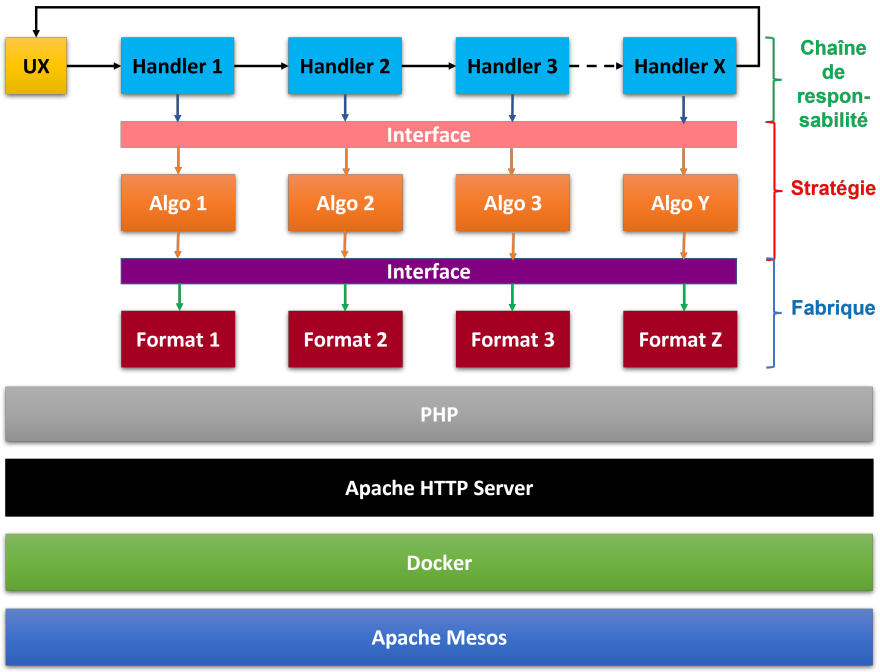


FIGURE 8.6 – Schéma général du service mis en oeuvre

décisionnel décrits dans Andriamandroso et al. [126] sur les mêmes données traitées sous Excel. Les résultats obtenus sont en tous points identiques à ceux calculés manuellement avec Excel. Les classes de vitesse ainsi que les durées des différentes activités : rumination, broutage, autre 1, autre 2, autre 3 ont également été vérifiées sur plusieurs fichiers csv produits par l'application Sensor Data 1.26 installée sur un iPhone 5s.

Les résultats de l'analyse comportementale (csv, ods, shp et excel) produits par des traitements distincts ont ensuite été comparés entre eux et aux mêmes fichiers traités manuellement à l'aide d'Excel. L'analyse des classes de vitesse et de déplacement du fichier PDF a également été validée sous Excel avec les classes par défaut et pour des classes avec des limites personnalisées. Le Tableau 8.1 montre les résultats obtenus pour différentes tailles de fichiers allant de 3Ko à 3Go.

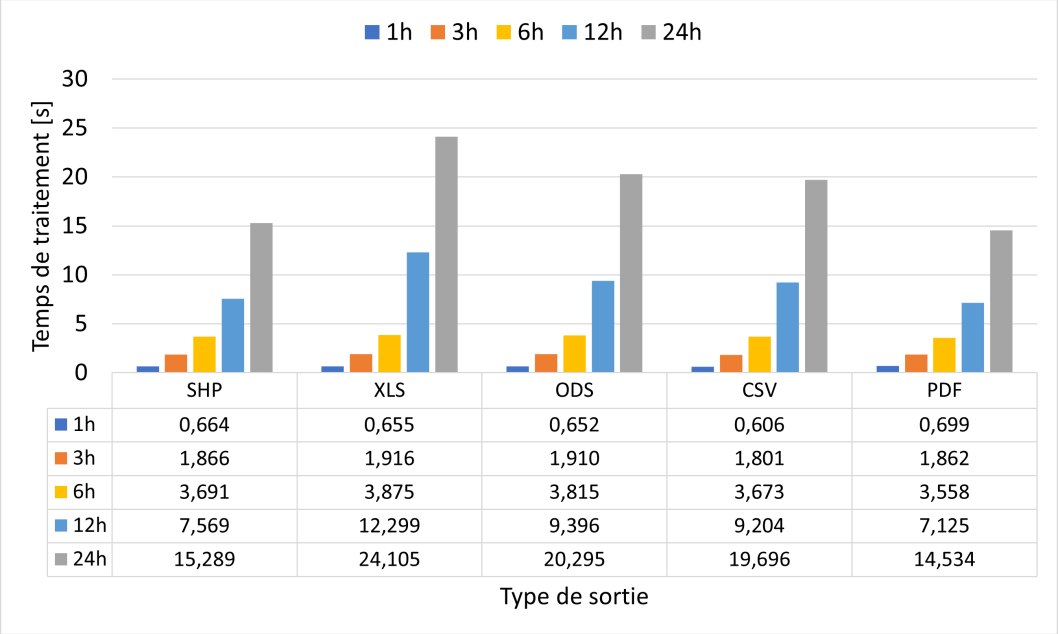
L'analyse des résultats montre de légères différences entre ce qui est calculé sous Excel et le service d'analyse du comportement. Ces différences sont dues au fait que les fichiers brut comportent une série de valeurs nulles qui correspondent au temps d'initialisation des capteurs qui sont classés par Excel comme des comportements autres (OTHER), tandis que le service ne considère pas la durée d'initialisation des capteurs. La non-nullité de l'ensemble des valeurs sauvegardées à un instant donné correspond au début de l'analyse comportementale.

Nous avons ensuite évalué les performances exprimées en temps de traitement du service d'analyse du comportement obtenus à partir de fichiers de log issus de Sensor

TABLE 8.1 – Comparaison des résultats de l'analyse comportementale réalisée sous Excel et par le service

Taille du fichier	Durée	Comportements					
		Brouttage (GRA)		Rumination (RUM)		Autre (OTHER)	
	[s]	Excel	Service	Excel	Service	Excel	Service
3Ko	0,12	0	0	0	0	0	0
30Ko	1,29	0	0	0	0	1	1
300Ko	12,90	0	0	0	0	13	13
3Mo	129,00	25	27	2	0	102	102
30Mo	1209,00	265	268	2	0	942	941
300Mo	12009,00	2722	2736	18	15	9269	9258
3Go	120009,00	28950	28935	28726	28729	62333	62345

Data 1.26 installé sur un iPhone 5S comportant 40 paramètres et correspondant à des durées de 1h, 3h, 6h, 12h et 24h. Les résultats de ces tests sont synthétisés par le graphique de la Figure 8.7. Ils correspondent aux temps de traitement de fichier uploadé manuellement à partir de l'interface web de l'application.



	SHP	XLS	ODS	CSV	PDF
1h	0,664	0,655	0,652	0,606	0,699
3h	1,866	1,916	1,910	1,801	1,862
6h	3,691	3,875	3,815	3,673	3,558
12h	7,569	12,299	9,396	9,204	7,125
24h	15,289	24,105	20,295	19,696	14,534

FIGURE 8.7 – Temps de traitement de fichiers correspondant à différentes durées et exportés dans différents formats

L'analyse des résultats montre que la production de fichiers Excel (XLS) est plus longue en matière de traitement que pour les formats ODT et CSV qui utilisent des

structures plus simples pour stocker les données et sont par conséquent plus rapides à exporter peu importe la taille du fichier à traiter. La taille des fichiers ODS est cependant plus importante que celle du XLS. La production de fichiers SIG reprenant la position et le comportement sont produits plus rapidement que les fichiers XLS, ODS et CSV malgré que les attributs c'est-à-dire les comportements sont sauvegardés dans un fichier DBF. Cette durée de traitement plus courte est attribuée à une librairie plus optimisée que celle qui est utilisée pour la sauvegarde en XLS, ODS et CSV.

Nous avons ensuite testé les performances de traitement de lignes de données stockées dans la base de données de l'architectures LAMA afin d'évaluer les performances globale de l'architecture et plus particulièrement de l'interaction entre les deux sous architectures de collecte et stockage d'une part et d'hébergement d'applicatifs d'autre part. Les temps de traitement intègrent le temps de requêtage de la base de données, de transfert des données au travers du réseau et de traitement à proprement dit des données par le service.

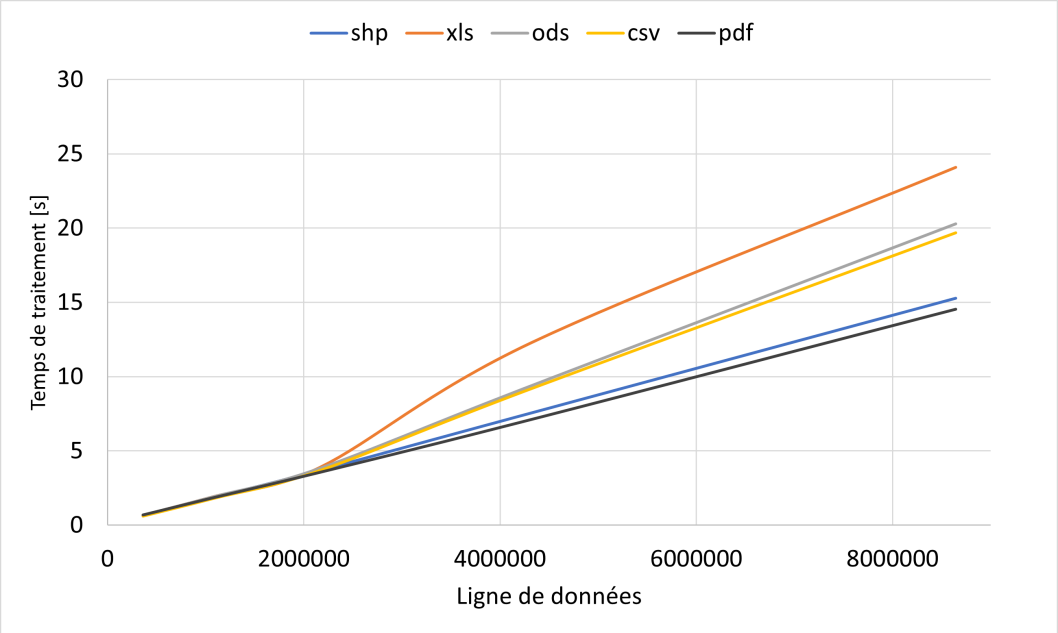


FIGURE 8.8 – Temps de traitement en fonction du nombre de lignes de données

Sur la Figure 8.8, une rupture dans la linéarité des courbes aux alentours de 2,2 millions de lignes peut être observée. Ce point d'inflexion correspond à l'utilisation de la totalité de la mémoire vive disponible au niveau de la VM. Le recours à la SWAP conduit une augmentation du délai de traitement qui conserve néanmoins, une allure linéaire avec l'augmentation du nombre de lignes de données à traiter.

8.1.8 Limitations

Cette plateforme a pour l'instant certaines limitations : (1) Elle ne dispose actuellement d'aucun service d'indexation et de découverte des données. Le partage des datasets se fait manuellement par une gestion des droits entre équipes de recherche ; (2) Elle ne peut traiter que les données issues des logs de l'application Sensor Data uniquement disponible sur iOS ou stockées dans l'architecture cloud ; (3) Un seul algorithme de traitement proposé par Andriamandroso et al [126] est disponible. Néanmoins, l'utilisation du design pattern Stratégie permettra d'implémenter d'autres algorithmes aisément à l'avenir.

8.2 Conclusion

L'outil d'analyse du comportement développé permet d'une part de restituer les informations aux éleveurs après traitement de leurs données et d'autre part aux chercheurs du monde entier de traiter facilement de grandes quantités de données et les mettre en commun pour élaborer des modèles plus robustes et/ou moins spécifiques à certaines races de bovins et/ou conditions environnementales. L'utilisation des patrons de conception permettra de compléter les formats de données supportés, de proposer d'autres algorithmes de traitement ainsi que de format de sortie après traitement.

De plus, l'association des comportements horodatés et des positions permettra une analyse des interactions entre animaux pour la détection notamment, des chevauchements des animaux indiquant de possibles chaleurs lors du chevauchement entre animaux, les agressions entre animaux, les relations de dominance. Le croisement des positions où les bouchées d'herbe ont été effectuées avec la cartographie des espèces végétales présentes dans le pâturage permettra de déterminer les préférences alimentaires individuel, la compétition pour la consommation de chaque espèce, d'effectuer un classement des préférences alimentaires de chaque individu.

9

Conclusions et perspectives

Ce chapitre dresse le bilan des contributions scientifiques engrangées durant cette thèse de doctorat et envisage les perspectives, les futurs développements de l'architecture. Nous évoquons également les thématiques de recherche qui pourront être abordée dans le prolongement de la présente thèse de doctorat.

9.1 Conclusions

Cette thèse de doctorat fruit de la collaboration entre les universités de Mons et de Liège a permis de mettre au point une architecture cloud modulaire et facilement adaptable pour des cas d'utilisations en Smart Farming. Cette architecture a été élaborée sur base d'un cas d'utilisation principal d'analyse du comportement des animaux de ferme qui a été complètement implémenté.

Notre principale contribution est cette architecture qui vise principalement à adresser la problématique liée à l'ingestion des données provenant des capteurs IoT, des sources de données extérieures et leur exploitation au travers de services. Par ailleurs, les chercheurs sont poussés à l'Open Science et l'Open Data dans lesquels, ils sont amenés à collaborer de plus en plus et à échanger leurs données brutes et traitées en vue mieux les valoriser.

Une architecture cloud modulaire capable de s'adapter à un grand nombre de cas d'utilisation sans avoir à la remanier en profondeur a été conceptualisée, mise en œuvre et testée. De plus, cette architecture cloud garantit l'anonymisation des données, leur traçabilité, le contrôle fin des accès, la récupération des données hébergées et maintient

ses performances avec l'augmentation de la quantité de données. D'autre part, elle permet de proposer des services sous forme de micro-services capables à la fois de gérer la sécurité, la traçabilité et l'anonymisation des données à partir des objets au niveau de la passerelle qui centralise les données.

Par ailleurs, notre architecture s'inscrivant également dans un contexte de recherche, elle permet de gérer et stocker de manière pérenne de grandes masses de données. En effet, l'ensemble des données brutes et traitées issues des expérimentations doivent être conservées afin de pouvoir être réutilisé et/ou valorisé dans d'autres projets de recherche. L'architecture a ensuite été améliorée grâce à l'automatisation du traitement et de l'ingestion des données collectées à l'aide de l'analyse de la sémantique associée aux informations (données brutes accompagnées de métadonnées descriptives). Les métadonnées sont encodées par les chercheurs préalablement à l'envoi des données expérimentales. Des services adaptent l'architecture pour préparer la chaîne de traitement des données. Le développement d'une architecture Edge AI-IoT pour le déploiement de micro-services et d'algorithmes d'intelligence artificielle est venu compléter et appuyer les capacités de traitements de notre architecture cloud.

Par ailleurs, une contribution secondaire qui revêt la forme d'un service en ligne d'analyse du comportement des animaux de ferme a été développé. Ce service s'appuie sur les données acquises par une centrale inertielle (IMU) implanté dans un collier placé sur le cou de l'animal. Ce service est hébergé sur la partie applicative de l'architecture cloud et permet de démontrer son fonctionnement et d'exploiter les capacités de l'ensemble de l'architecture.

Nos travaux de recherche adresse les préoccupations exprimées par les chercheurs en proposant une solution architecturale pérenne, un processus d'automatisation de la création des pipelines à partir des métadonnées expérimentales. L'architecture LAMA permet également de conserver les données à long terme, de traiter les données en temps réels et en temps différés par lots.

L'architecture LAMA et les différentes contributions scientifiques ont fait l'objet **1** chapitre de livre, **9** articles de journaux, **20** papiers de conférence, **2** proceedings de conférence et **3** posters démontrant son fonctionnement et son applicabilité dans de nombreux cas d'utilisations liés au Smart Farming et à l'environnement afin de démontrer la polyvalence de nos travaux. Par ailleurs, nous avons également publié **5** articles de vulgarisation et réalisé **8** papiers de conférence et **1** chapitre connexe à nos travaux. Soit un total de **49** publications sur la durée de nos travaux.

En outre, nos travaux ouvrent le champ vers de nouvelles thèses de doctorat sur la sémantique, le déploiement de micro-services et d'algorithmes d'intelligence artificielle et par extension à une nouvelle tendance qui est le traitement sur périphérique mobile (Mobile Edge Computing).

Finalement, une collaboration scientifique a vu le jour dans le cadre de cette thèse.

La première avec le Dr Meryem Elmoulat de l'université Mohammed V de Rabat, laboratoire GeoRisk grâce à laquelle nous avons pu réaliser le déploiement sur site d'un réseau de capteurs pour le monitoring des mouvements de terrain, tester l'architecture durant 2 années dans des conditions réelles d'utilisation et de réaliser trois publications.

9.2 Perspectives

Les perspectives envisagent d'une part l'évolution et le développement de l'architecture et des contributions complémentaires initiés durant cette thèse de doctorat et d'autre part l'intégration de l'architecture LAMA dans d'autres axes de recherches (Mobile Edge Computing et Edge AI-IoT).

9.2.1 Architecture développée.

1. **Implémentation de la sémantique**

   Apache Beam
   Le développement de la sémantique par l'implémentation pour la description des capteurs des ontologies SSN et SOSA, et l'adaptation d'une ontologie complète et interopérable existante ou le développement d'une nouvelle ontologie en vue de décrire intégralement l'ensemble des données présentes sur la plateforme, de faciliter leur découverte et de les rendre interopérables avec les systèmes qui les exploiteront.

   La plateforme évolue actuellement dans un contexte privatif limité aux équipes de recherches qui en connaissent l'existence. Celle-ci sera rendue plus attractive grâce au développement de services de découverte des données et des services proposés par l'architecture respectivement au niveau de ses parties IoT et applicatives. La possibilité de découvrir les données, les applications et les services de la plateforme pour d'autres équipes de recherches et/ou prestataires de services publics ou privés.

2. **Implémentation de la couche de service**

   Une première évolution consistera à développer une couche d'accès aux services de la plateforme au travers d'un webservice. En effet, l'architecture initiée dans cette thèse de doctorat ne dispose pas de couche d'accès aux services. Celui-ci permettra de gérer facilement le déploiement des applications, des droits d'accès, la gestion des versions et l'archivage sur la plateforme applicative. Ces actions étant actuellement pour la plupart gérée manuellement et pourraient

pour certaines être automatisées comme les sauvegardes, afin d'en faciliter l'utilisation et la gestion.

De plus, Le seul service actuellement proposé par la plateforme est l'analyse comportemental des bovins. Le développement de nouveaux services améliorera l'attractivité et l'utilisabilité de la plateforme.

3. **Optimisation de l'ingestion et du traitement distribués**

Le traitement au niveau du Edge devra être optimisé de manière à permettre l'ingestion, le traitement rapide des données en vue de les compresser avec ou sans perte et/ou de ne transmettre que les données possédant un caractère particulier. Le déport d'une partie du traitement du cloud vers le Edge permet de diminuer substantiellement les besoins de traitement au niveau cloud, les coûts de la bande passante et apporte une réponse aux problèmes de confidentialité des données sensibles. La répartition dynamique des traitements des tâches entre les niveaux Cloud, le Fog et le Edge est une question de recherche qui n'a pas encore trouvée une solution consensuelle.

4. **Développement complet du cas d'utilisation "Phénotypage Digital"**

Ce cas d'utilisation à pour l'heure uniquement fait l'objet d'une conceptualisation mais pas encore d'une implémentation complète dans une situation de fonctionnement réelle. Ce cas d'utilisation utilisera des images issues de caméra 3D utilisées pour la construction du double numérique de chacune des plantes. Parallèlement les paramètres environnementaux font l'objet d'une acquisition basse fréquence.

9.2.2 Amélioration des contributions complémentaires.

1. **Extension du service d'analyse comportemental**

Le service d'analyse du comportement implémente actuellement l'algorithme décrit dans Andrianmandroso et al. 2017 [126] basé sur un arbre décisionnel utilisant des paramètres statistiques. D'autres algorithmes existent et devront être incorporé au service pour permettre à l'utilisateur de choisir. De plus, autoriser la paramétrisation fine des algorithmes pour les utilisateurs avancé permettra d'améliorer la qualité des analyses. Il serait également intéressant d'envisager que les utilisateurs avancés puissent proposer leur propre algorithme. Finalement, le dépôt de plusieurs fichiers simultanément permettra d'automatiser le traitement des fichiers afférents à tout le troupeau plutôt que de charger indivi-

duelle de chaque fichier comme c'est le cas actuellement.

La collecte massive de données provenant d'animaux de races et évoluant des environnements différents pourrait permettre d'élaborer des modèles statistiques plus robuste ou d'élaborer des algorithmes d'intelligence artificielle capable de détecter plus finement les macro et les micro-comportements des animaux de fermes. L'application Sensor Data qui permet la collecte des donnée de l'IMU sur iPhone n'est plus développée et devra donc être remplacée par une autre application commerciale ou une application multi plateforme pour l'acquisition (*logging*) à haute fréquence (100 Hz voir plus) d'un panel de capteurs plus large que l'IMU (9-DOF) + GPS et être accompagnées de métadonnées descriptives devra être développée.

9.2.3 Intégration de l'architecture cloud développée dans de nouveaux axes de recherches.

1. **Mobile Edge Computing**

   Grâce au déploiement de la 5G et aux possibilité d'interconnexion de grandes quantités de périphériques mobiles à haut débit, elle permet l'émergence du paradigme Mobile Edge Computing. L'évolution constante des périphériques mobiles tant en matière de puissance de calcul que de capacité de stockage permet d'envisager leur utilisation pour l'apprentissage fédéré (*Federated Learning*) et une distribution de l'entraînement des algorithmes d'intelligence artificielle au niveau local. Ces aspectes ont partiellement été évoqués dans Elmoulat et al. 2021 [172].

2. **Edge AI-IoT**

   Une architecture Edge AI-IoT a été conceptualisé, implémentée et partiellement testée dans Debauche et al. 2020 [173], appliquée à la détection des maladies et des ravageurs dans Debauche et al. 2020 [127], au suivi des glissements de terrains dans Elmoulat et al. 2020 et 2021 [174, 172]. Celle-ci devra encore être développée pour améliorer sa collaboration avec l'architecture cloud développée et permettre un déploiement aisé des algorithmes élaborés sur le cloud durant la période de recherche, en vue de réaliser leur exploitation sous une forme allégée au niveau du Edge.

3. **IoT médicale**

   Cette perspective s'inscrit dans le prolongement des travaux publiés dans Debauche et al [175] dont l'objectif était la mise en place d'un dispositif Fog pour le suivi des patients alités en convalescence ou les personnes âgées.

   L'application de l'architecture au domaine médical nécessitera l'obtention de certifications spécifiques : (1) Certification et SMSI relatifs à la gestion de la sécurité de l'information pour les services cloud ISO/IEC 27001 :2013 ; (2) Certification relative au contrôle de la sécurité de l'information pour les services cloud ISO/IEC 27017 :2015 ; (3) Code de bonne pratique relatif à la protection des données personnel pour les services cloud ISO/IEC 27018 :2014 ; (4) Certification et PIMS relatifs à la gestion de la sécurité du traitement des données personnelles ISO/IEC 27701 :2019 ; (5) Conformité au Règlement (UE) 2016/679, dit Règlement Général sur la Protection des Données (RGPD). Auxquelles s'ajoute des certifications locales liées au lieu d'utilisation de l'architecture par exemple :(HDS [1] en France, HIPAA & HITECH [2] aux USA).

Les travaux initiés dans le cadre de cette thèse permettront également d'alimenter les recherches de l'institut dans la thématique de l'Intelligence Artificielle. En effet, l'intelligence artificielle en Région Wallonne bénéficie d'une structuration interuniversitaire (en collaboration avec les CRA actifs en IA, AI4Belgium et l'Agence du Numérique) : TRAIL.

1. L'Agence du numérique en santé (ANS) fixe un cadre rigoureux pour les pratiques liées à l'hébergement de données de santé. La certification HDS y est d'ailleurs une condition obligatoire.

2. La loi américaine HIPAA (Health Insurance Portability and Accountability Act) établit depuis 1996 les conditions d'utilisation, de divulgation et de protection des informations de santé. En 2009, la loi HITECH (Health Information Technology for Economic and Clinical Health Act) a été créée pour la compléter.

Bibliographie

[1] L. Carnevale, A. Celesti, A. Galletta, S. Dustdar, and M. Villari, "Osmotic computing as a distributed multi-agent system : The body area network scenario," *Internet of Things*, vol. 5, pp. 130 – 139, 2019. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S2542660518300751 1

[2] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the internet of things," *Cluster of European Research Projects on the Internet of Things, European Commision*, vol. 3, no. 3, pp. 34–36, 2010. 1

[3] C. Cisco, "Cisco global cloud index : Forecast and methodology, 2016–2021," *San Jose : Cisco*, 2018. 1, 47

[4] J. Manyika and M. Chui, "By 2025, internet of things applications could have $11 trillion impact," *Insight Publications*, 2015. [Online]. Available : https://www.mckinsey.com/mgi/overview/in-the-news/by-2025-internet-of-things-applications-could-have-11-trillion-impact 1, 45, 47

[5] A. Meola, "Smart farming in 2020 : How iot sensors are creating a more efficient precision agriculture industry," 2021. [Online]. Available : https://www.businessinsider.com/smart-farming-iot-agriculture?IR=T 2

[6] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009. 15

[7] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things : Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012. 15

[8] B. Billet, "Système de gestion de flux pour l'internet des objets intelligents," Ph.D. dissertation, Université de Versailles, Saint-Quentin-En-Yvelinnes (France), 2015. 15

[9] H.-J. Yim, D. Seo, H. Jung, M.-K. Back, I. Kim, and K.-C. Lee, "Description and classification for facilitating interoperability of heterogeneous data/events/services in the internet of things," *Neurocomputing*, vol. 256, pp. 13–22, 2017. 15

[10] D. Evans, "The internet of things how the next evolution of the internet is changing everything (april 2011)," *White Paper by Cisco Internet Business Solutions Group (IBSG)*, 2012. 15, 20

[11] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things : A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. 15, 16, 20

[12] E. Cavalcante, J. Pereira, M. P. Alves, P. Maia, R. Moura, T. Batista, F. C. Delicato, and P. F. Pires, "On the interplay of internet of things and cloud computing : A systematic mapping study," *Computer Communications*, vol. 89, pp. 17–33, 2016. 15, 18

[13] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter : A survey on data-centric internet of things," *Journal of Network and Computer Applications*, vol. 64, pp. 137–153, 2016. 17

[14] N. Madaan, M. A. Ahad, and S. M. Sastry, "Data integration in iot ecosystem : Information linkage as a privacy threat," *Computer law & security review*, vol. 34, no. 1, pp. 125–133, 2018. 17

[15] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko *et al.*, "Openiot : Open source internet-of-things in the cloud," in *Interoperability and open-source solutions for the internet of things*. Springer, 2015, pp. 13–25. 18

[16] P. P. Ray, "A survey of iot cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, 2016. 20

[17] V. Saiz-Rubio and F. Rovira-Más, "From smart farming towards agriculture 5.0 : A review on crop data management," *Agronomy*, vol. 10, no. 2, p. 207, 2020. 21

[18] A. Triantafyllou, P. Sarigiannidis, and S. Bibi, "Precision agriculture : A remote sensing monitoring system architecture," *Information*, vol. 10, no. 11, p. 348, 2019. 21, 22, 23, 58

[19] J. Iaksch, E. Fernandes, and M. Borsato, "Digitalization and big data in smart farming–a review," *Journal of Management Analytics*, vol. 8, no. 2, pp. 333–349, 2021. 21

[20] Z. Zhai, J. F. Martínez, V. Beltran, and N. L. Martínez, "Decision support systems for agriculture 4.0 : Survey and challenges," *Computers and Electronics in Agriculture*, vol. 170, p. 105256, 2020. 21, 58

[21] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big data in smart farming–a review," *Agricultural systems*, vol. 153, pp. 69–80, 2017. 21, 23

[22] O. Debauche, J.-P. Trani, S. Mahmoudi, P. Manneback, J. Bindelle, S. A. Mahmoudi, A. Guttadauria, and F. Lebeau, "Data management and internet of things : A methodological review in smart farming," *Internet of Things*, vol. 14, p. 100378, 2021. 21, 22, 31, 41

[23] A. Kamilaris, A. Kartakoullis, and F. X. Prenafeta-Boldú, "A review on the practice of big data analysis in agriculture," *Computers and Electronics in Agriculture*, vol. 143, pp. 23–37, 2017. 22

[24] O. Debauche, J.-P. Trani, S. Mahmoudi, P. Manneback, J. Bindelle, S. Mahmoudi, and F. Lebeau, "Data management and internet of things : A methodological review in smart farming," *Internet of Things*, p. 100378, 2021. [Online]. Available : https://www.sciencedirect.com/science/article/pii/S2542660521000226 22

[25] P. Jayaraman, A. Yavari, D. Georgakopoulos, A. Morshed, and A. Zaslavsky, "Internet of things platform for smart farming : Experiences and lessons learnt," *Sensors*, vol. 16, no. 11, p. 1884, 2016. 22, 23, 24, 33, 34

[26] L. García, L. Parra, J. M. Jimenez, J. Lloret, and P. Lorenz, "Iot-based smart irrigation systems : An overview on the recent trends on sensors and iot systems for irrigation in precision agriculture," *Sensors*, vol. 20, no. 4, p. 1042, 2020. 23, 33

[27] C. Bahlo, P. Dahlhaus, H. Thompson, and M. Trotter, "The role of interoperable data standards in precision livestock farming in extensive livestock systems : A review," *Computers and Electronics in Agriculture*, vol. 156, pp. 459 – 466, 2019. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0168169918312699 23

[28] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of things in agriculture, recent advances and future challenges," *Biosystems Engineering*, vol. 164, pp. 31–48, 2017. 24, 31, 32

[29] L. Wiseman, J. Sanderson, A. Zhang, and E. Jakku, "Farmers and their data : An examination of farmers' reluctance to share their data through the lens of the laws impacting smart farming," *NJAS-Wageningen Journal of Life Sciences*, 2019. 24

[30] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge : A scalable iot architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017. 29

[31] A. Munir, P. Kansakar, and S. U. Khan, "Ifciot : Integrated fog cloud iot : A novel architectural paradigm for the future internet of things." *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 74–82, July 2017. 29, 47

[32] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things : a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77 – 86, 2018. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S2352864817301335 29, 47, 49, 59

[33] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing : Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016. 29, 46

[34] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. Lin, "Edge of things : The big picture on the integration of edge, iot and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2018. 29

[35] Y. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms : a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, December 2017. 29, 46, 49

[36] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, *Edge AI : Convergence of Edge Computing and Artificial Intelligence*. Springer Nature, 2020. 30, 48, 52, 60

[37] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, "Review of iot applications in agro-industrial and environmental fields," *Computers and Electronics in Agriculture*, vol. 142, pp. 283–297, 2017. 31, 32

[38] P. P. Ray, "Internet of things for smart agriculture : Technologies, practices and future direction," *Journal of Ambient Intelligence and Smart Environments*, vol. 9, no. 4, pp. 395–420, 2017. 31, 32

[39] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (iot) and data analytics in agriculture : Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758–3773, 2018. 31, 32

[40] A. Khanna and S. Kaur, "Evolution of internet of things (iot) and its significant impact in the field of precision agriculture," *Computers and electronics in agriculture*, vol. 157, pp. 218–231, 2019. 31, 32

[41] X. Shi, X. An, Q. Zhao, H. Liu, L. Xia, X. Sun, and Y. Guo, "State-of-the-art internet of things in protected agriculture," *Sensors*, vol. 19, no. 8, p. 1833, 2019. 31, 32

[42] J. Ruan, H. Jiang, C. Zhu, X. Hu, Y. Shi, T. Liu, W. Rao, and F. T. S. Chan, "Agriculture iot : Emerging trends, cooperation networks, and outlook," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 56–63, 2019. 31, 32

[43] X. Feng, F. Yan, and X. Liu, "Study of wireless communication technologies on internet of things for precision agriculture," *Wireless Personal Communications*, vol. 108, no. 3, pp. 1785–1802, 2019. 31, 32

[44] U. Shafi, R. Mumtaz, J. García-Nieto, S. A. Hassan, S. A. R. Zaidi, and N. Iqbal, "Precision agriculture techniques and practices : From considerations to applications," *Sensors*, vol. 19, no. 17, p. 3796, 2019. 31, 32

[45] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture : Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129 551–129 583, 2019. 31, 32

[46] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of iot in agriculture for the implementation of smart farming," *IEEE Access*, vol. 7, pp. 156 237–156 271, 2019. 31, 32

[47] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of uav applications for precision agriculture," *Computer Networks*, vol. 172, p. 107148, 2020. 31, 32

[48] M. A. Ferrag, L. Shu, X. Yang, A. Derhab, and L. Maglaras, "Security and privacy for green iot-based agriculture : Review, blockchain solutions, and challenges," *IEEE access*, vol. 8, pp. 32 031–32 053, 2020. 31, 32

[49] N. Misra, Y. Dixit, A. Al-Mallahi, M. S. Bhullar, R. Upadhyay, and A. Martynenko, "Iot, big data and artificial intelligence in agriculture and food industry," *IEEE Internet of Things Journal*, 2020. 31, 33

[50] A. Villa-Henriksen, G. T. Edwards, L. A. Pesonen, O. Green, and C. A. G. Sørensen, "Internet of things in arable farming : Implementation, applications, challenges and potential," *Biosystems Engineering*, vol. 191, pp. 60–84, 2020. 31, 32

[51] E. Navarro, N. Costa, and A. Pereira, "A systematic review of iot solutions for smart farming," *Sensors*, vol. 20, no. 15, p. 4231, 2020. 31, 33

[52] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz, "From industry 4.0 to agriculture 4.0 : Current status, enabling technologies, and research challenges," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4322–4334, 2020. 31

[53] O. Friha, M. A. Ferrag, L. Shu, L. A. Maglaras, and X. Wang, "Internet of things for the future of smart agriculture : A comprehensive survey of emerging technologies," *IEEE CAA J. Autom. Sinica*, vol. 8, no. 4, pp. 718–752, 2021. 31, 33

[54] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari, "Lorafarm : A lorawan-based smart farming modular iot architecture," *Sensors*, vol. 20, no. 7, 2020. [Online]. Available : https://www.mdpi.com/1424-8220/20/7/2028 33, 34

[55] M. A. G. Maureira, D. Oldenhof, and L. Teernstra, "Thingspeak–an api and web service for the internet of things," *World Wide Web*, 2011. 34

[56] M. A. Rodriguez, L. Cuenca, and A. Ortiz, "Fiware open source standard platform in smart farming - a review," in *Collaborative Networks of Cognitive Systems*, L. M. Camarinha-Matos, H. Afsarmanesh, and Y. Rezgui, Eds. Cham : Springer International Publishing, 2018, pp. 581–589. 34

[57] NECTEC, "Netpie - network platform for internet of everything," 2020. [Online]. Available : https://netpie.io 34

[58] Ubidots, "Ubidots," 2021. [Online]. Available : https://ubidots.com/ 34

[59] A. Luis Bustamante, M. A. Patricio, and J. M. Molina, "Thinger. io : An open source platform for deploying data fusion applications in iot environments," *Sensors*, vol. 19, no. 5, p. 1044, 2019. 34

[60] KaaIoT, "Ubidots," 2021. [Online]. Available : https://docs.kaaiot.io/KAA/docs/current/Welcome/ 34

[61] IBM, "Ibm watson iot platform," 2015. [Online]. Available : https://internetofthings.ibmcloud.com/ 34

[62] Microsoft, "Azure iot," 2021. [Online]. Available : https://azure.microsoft.com/en-us/overview/iot/ 34

[63] P. AT&T, "At&t continues to fuel growth of the internet of things with launch of new developer-friendly managed service," 2021. [Online]. Available : https://about.att.com/story/m2x_data_service_for_enterprise_developers.html 34

[64] Blynk, "Blynk iot platform : for businesses and developers," 2021. [Online]. Available : https://blynk.io 34

[65] N. Sigrimis, K. Arvanitis, and K. Ferentinos, "Macqu : An open scada system for intelligent management and control of greenhouses," in *2002 ASAE Annual Meeting*. American Society of Agricultural and Biological Engineers, 2002, p. 1. 34

[66] C. Granell, I. Miralles, L. E. Rodríguez-Pupo, A. González-Pérez, S. Casteleyn, L. Busetto, M. Pepe, M. Boschetti, and J. Huerta, "Conceptual architecture and service-oriented implementation of a regional geoportal for rice monitoring," *ISPRS International Journal of Geo-Information*, vol. 6, no. 7, 2017. [Online]. Available : https://www.mdpi.com/2220-9964/6/7/191 34

[67] S. Kodati and S. Jeeva, "Smart agricultural using internet of things, cloud and big data," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 10, pp. 3718–3722, 2019. 34

[68] L. A. Pesonen, F. K.-W. Teye, A. K. Ronkainen, M. O. Koistinen, J. J. Kaivosoja, P. F. Suomi, and R. O. Linkolehto, "Cropinfra–an internet-based service infrastructure to support crop production in future farms," *Biosystems engineering*, vol. 120, pp. 92–101, 2014. 34

[69] P. H. Corp, "Sensorcloud," 2020. [Online]. Available : https://sensorcloud.com/ 34

[70] Amazon, "Amazon web services," 2021. [Online]. Available : https://aws.amazon.com/ 35

[71] IBM, "Ibm cloud," 2021. [Online]. Available : https://www.ibm.com/cloud 35

[72] Microsoft, "Azure," 2021. [Online]. Available : https://azure.microsoft.com 35

[73] I. Integra Souces, "Iot solution development services," 2021. [Online]. Available : https://www.integrasources.com/iot-page/ 35

[74] Amazon, "Amazon dynamodb," 2021. [Online]. Available : https://aws.amazon.com/fr/dynamodb/ 35

[75] Mongo, "Mongodb atlas," 2021. [Online]. Available : https://www.mongodb.com/en-us/cloud/atlas 35

[76] Google, "Firebase," 2021. [Online]. Available : https://firebase.google.com/ 35

[77] Influxdata, "Infludb cloud," 2021. [Online]. Available : https://www.influxdata.com/products/influxdb-cloud/ 35

[78] Oracle, "Mysql," 2021. [Online]. Available : https://www.mysql.com 35

[79] SQLite, "Sqlite," 2021. [Online]. Available : https://www.sqlite.org 35

[80] P. The PostgreSQL Global Development Group, "Postgresql : The world's most advanced open source relational database," 2021. [Online]. Available : https://www.postgresql.org/ 35

[81] Cassandra, "Cassandra," 2021. [Online]. Available : https://cassandra.apache.org 35

[82] Druid, "Druid," 2021. [Online]. Available : https://druid.apache.org 35

[83] A.-H. M. Sallah, B. Tychon, I. Piccard, A. Gobin, R. Van Hoolst, B. Djaby, and J. Wellens, "Batch-processing of aquacrop plug-in for rainfed maize using satellite derived fractional vegetation cover data," *Agricultural Water Management*, vol. 217, pp. 346–355, 2019. 39

[84] F. N. Fote, A. Roukh, S. Mahmoudi, S. A. Mahmoudi, and O. Debauche, "Toward a big data knowledge-base management system for precision livestock farming," *Procedia Computer Science*, vol. 177, pp. 136–142, 2020. 39

[85] N. Miloslavskaya and A. Tolstoy, "Big data, fast data and data lake concepts," *Procedia Computer Science*, vol. 88, pp. 300 – 305, 2016, 7th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2016, held July 16 to July 19, 2016 in New York City, NY, USA. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S1877050916316957 40, 55

[86] N. Marz and J. Warren, *Big Data : Principles and best practices of scalable real-time data systems*. Manning, 2013. 41

[87] K. N. Singh, R. K. Behera, and J. K. Mantri, "Big data ecosystem : Review on architectural evolution," in *Emerging Technologies in Data Mining and Information Security*, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds. Singapore : Springer Singapore, 2019, pp. 335–345. 41

[88] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99 – 117, 2016. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S108480451600028X 41, 75

[89] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "Alljoyn lambda : An architecture for the management of smart environments in iot," in *2014 International Conference on Smart Computing Workshops*, Nov 2014, pp. 9–14. 41, 75

[90] J. Kreps, "Questioning the lambda architecture," *Online article, July*, p. 205, 2014. [Online]. Available : https://www.oreilly.com/radar/questioning-the-lambda-architecture/ 41, 42

[91] A. Roukh, F. N. Fote, S. A. Mahmoudi, and S. Mahmoudi, "Wallesmart : Cloud platform for smart farming," in *32nd International Conference on Scientific and Statistical Database Management*, 2020, pp. 1–4. 42

[92] A. Roukh, F. N. Fote, S. Mahmoudi, and S. A. Mahmoudi, "Big data processing architecture for smart farming," *Procedia Computer Science*, vol. 177, pp. 78–85, 2020. 42

[93] O. Debauche, S. A. Mahmoudi, N. De Cock, S. Mahmoudi, P. Manneback, and F. Lebeau, "Cloud architecture for plant phenotyping research," *Concurrency and Computation : Practice and Experience*, vol. 32, no. 17, p. e5661, 2020. 42

[94] O. Debauche, S. Mahmoudi, A. L. H. Andriamandroso, P. Manneback, J. Bindelle, and F. Lebeau, "Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 12, pp. 4651–4662, 2019. 42, 52

[95] O. Debauche, S. A. Mahmoudi, S. Mahmoudi, and P. Manneback, "Cloud platform using big data and hpc technologies for distributed and parallels treatments," *Procedia Computer Science*, vol. 141, pp. 112–118, 2018. 42

[96] T. Siciliani, "Big data using lambda architecture," 2015. [Online]. Available : https://dzone.com/articles/lambda-architecture-big-data 42

[97] B. Lakhe, *Practical Hadoop migration : how to integrate your RDBMS with the Hadoop ecosystem and re-architect relational applications to NoSQL*. Apress, 2016. 43

[98] V. Persico, A. Pescapé, A. Picariello, and G. Sperlí, "Benchmarking big data architectures for social networks data processing using public cloud platforms," *Future Generation Computer Systems*, vol. 89, pp. 98–109, 2018. 43

[99] J. B. Nkamla Penka, S. Mahmoudi, and O. Debauche, "A new kappa architecture for iot data management in smart farming," *Procedia Computer Science*, 2021, in press. 43

[100] R. Estrada and I. Ruiz, "Big data smack : A guide to apache spark," *Mesos, Akka, Cassandra, and Kafka*, 2016. 43

[101] R. C. Fernandez, P. R. Pietzuch, J. Kreps, N. Narkhede, J. Rao, J. Koshy, D. Lin, C. Riccomini, and G. Wang, "Liquid : Unifying nearline and offline big data integration." in *CIDR*. Citeseer, 2015. 43

[102] J. Scott, "Zeta architecture : Hexagon is the new circle. an enterprise architecture solution for scale and efficiency," 2015. [Online]. Available : https://www.oreilly.com/ideas/zeta-architecture-hexagon-is-the-new-circle 44

[103] C. Giebler, C. Stach, H. Schwarz, and B. Mitschang, "Braid : A hybrid processing architecture for big data," in *Proceedings of the 7th International Conference on Data Science, Technology and Applications*, ser. DATA 2018. Setubal, PRT : SCITEPRESS - Science and Technology Publications, Lda, 2018, p. 294–301. [Online]. Available : https://doi.org/10.5220/0006861802940301 44

[104] M. Hausenblas, "Internet of things architecture (iot-a) home page," 2014. [Online]. Available : https://github.com/mhausenblas/iot-a.info 44

[105] J. Meehan, S. Zdonik, Shaobo Tian, Yulong Tian, N. Tatbul, A. Dziedzic, and A. Elmore, "Integrating real-time and batch processing in a polystore," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2016, pp. 1–7. 44

[106] M. Kazim, L. Liu, and S. Y. Zhu, "A framework for orchestrating secure and dynamic access of iot services in multi-cloud environments," *IEEE Access*, vol. 6, pp. 58 619–58 633, 2018. 45

[107] M. Assis and L. Bittencourt, "A survey on cloud federation architectures : Identifying functional and non-functional properties," *Journal of Network and Computer Applications*, vol. 72, pp. 51 – 71, 2016. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S1084804516301436 45

[108] G. Sipos, M. Turilli, S. Newhouse, and P. Kacsuk, *A European Federated Cloud : Innovative distributed computing solutions by EGI*, ser. EGU General Assembly Conference Abstracts. EGU, Apr. 2013. 45

[109] D. Petcu, B. Di Martino, S. Venticinque, M. Rak, T. Máhr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka *et al.*, "Experiences in building a mosaic of clouds," *Journal of Cloud Computing : Advances, Systems and Applications*, vol. 2, no. 1, p. 12, 2013. 45

[110] A. Drakos, V. Protonotarios, and N. Manouselis, "aginfra : a research data hub for agriculture, food and the environment," *F1000Research*, vol. 4, 2015. 46

[111] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things : A survey," *Future Generation Computer Systems*, vol. 56, pp. 684 – 700, 2016. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0167739X15003015 46

[112] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al. : A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680 – 698, 2018. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0167739X16305635 46, 47, 49

[113] M. Uehara, *Mist Computing : Linking Cloudlet to Fogs*. Cham : Springer International Publishing, 2018, pp. 201–213. [Online]. Available : https://doi.org/10.1007/978-3-319-63618-4_15 47

[114] P. Sethi and S. R. Sarangi, "Internet of things : architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017. 47

[115] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing : Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, p. 37–42, Sep. 2015. [Online]. Available : https://doi.org/10.1145/2831347.2831354 47

[116] C. C. Byers, "Architectural imperatives for fog computing : Use cases, requirements, and architectural techniques for fog-enabled iot networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, Aug 2017. 47

[117] S. Yang, "Iot stream processing and analytics in the fog," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 21–27, Aug 2017. 47

[118] P. Mach and Z. Becvar, "Mobile edge computing : A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, thirdquarter 2017. 47

[119] F. Sharofidinov, M. S. A. Muthanna, V. D. Pham, A. Khakimov, A. Muthanna, and K. Samouylov, "Agriculture management based on lora edge computing system," in *Distributed Computer and Communication Networks*, V. M. Vishnevskiy, K. E. Samouylov, and D. V. Kozyrev, Eds. Cham : Springer International Publishing, 2020, pp. 113–125. 48, 49

[120] E. Guardo, A. Di Stefano, A. La Corte, M. Sapienza, and M. Scatà, "A fog computing-based iot framework for precision agriculture," *Journal of Internet Technology*, vol. 19, no. 5, pp. 1401–1411, 2018. 48

[121] M. Taneja, N. Jalodia, J. Byabazaire, A. Davy, and C. Olariu, "Smartherd management : A microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming," *Software : practice and experience*, vol. 49, no. 7, pp. 1055–1078, 2019. 49

[122] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks : New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, April 2017. 50

[123] G. Valecce, S. Strazzella, and L. A. Grieco, "On the interplay between 5g, mobile edge computing and robotics in smart agriculture scenarios," in *Ad-Hoc, Mobile, and Wireless Networks*, M. R. Palattella, S. Scanzio, and S. Coleri Ergen, Eds. Cham : Springer International Publishing, 2019, pp. 549–559. 50, 51

[124] D. Fan and S. Gao, "The application of mobile edge computing in agricultural water monitoring system," in *IOP Conference Series : Earth and Environmental Science*, vol. 191, no. 1. IOP Publishing, 2018, p. 012015. 50

[125] Y. Tang, S. Dananjayan, C. Hou, Q. Guo, S. Luo, and Y. He, "A survey on the 5g network and its impact on agriculture : Challenges and opportunities," *Computers and Electronics in Agriculture*, vol. 180, p. 105895, 2021. 52, 54

[126] A. L. H. Andriamandroso, F. Lebeau, Y. Beckers, E. Froidmont, I. Dufrasne, B. Heinesch, P. Dumortier, G. Blanchy, Y. Blaise, and J. Bindelle, "Development of an open-source algorithm based on inertial measurement units (imu) of a smartphone to detect cattle grass intake and ruminating behaviors," *Computers and electronics in agriculture*, vol. 139, pp. 126–137, 2017. 52, 85, 86, 151, 152, 157, 160, 164

[127] O. Debauche, S. Mahmoudi, M. Elmoulat, S. A. Mahmoudi, P. Manneback, and F. Lebeau, "Edge ai-iot pivot irrigation, plant diseases, and pests identification," *Procedia Computer Science*, vol. 177, pp. 40–48, 2020. 52, 165

[128] D. Popescu, F. Stoican, G. Stamatescu, L. Ichim, and C. Dragana, "Advanced uav–wsn system for intelligent monitoring in precision agriculture," *Sensors*, vol. 20, no. 3, p. 817, 2020. 52

[129] O. Debauche, M. El Moulat, S. Mahmoudi, S. Boukraa, P. Manneback, and F. Lebeau, "Web monitoring of bee health for researchers and beekeepers based on the internet of things," *Procedia computer science*, vol. 130, pp. 991–998, 2018. 52

[130] E. Badidi, "Qos-aware placement of tasks on a fog cluster in an edge computing environment," *Journal of Ubiquitous Systems & Pervasive Networks*, vol. 13, no. 1, pp. 11–19, 2020. 52, 53

[131] M. Gupta, M. Abdelsalam, S. Khorsandroo, and S. Mittal, "Security and privacy in smart farming : Challenges and opportunities," *IEEE Access*, vol. 8, pp. 34 564–34 584, 2020. 52

[132] O. Debauche, S. Mahmoudi, S. A. Mahmoudi, P. Manneback, and F. Lebeau, "A new edge architecture for ai-iot services deployment," *Procedia Computer Science*, vol. 175, pp. 10–19, 2020. 52

[133] O. Debauche, S. Mahmoudi, M. Elmoulat, S. A. Mahmoudi, P. Manneback, and F. Lebeau, "Edge ai-iot pivot irrigation, plant diseases, and pests identification," *Procedia Computer Science*, vol. 177, pp. 40–48, 2020. 53

[134] O. Debauche, S. Mahmoudi, S. A. Mahmoudi, P. Manneback, J. Bindelle, and F. Lebeau, "Edge computing and artificial intelligence for real-time poultry monitoring," *Procedia Computer Science*, vol. 175, pp. 534–541, 2020. 53

[135] M. Taneja, J. Byabazaire, N. Jalodia, A. Davy, C. Olariu, and P. Malone, "Machine learning based fog computing assisted data-driven approach for early lameness detection in dairy cattle," *Computers and Electronics in Agriculture*, vol. 171, p. 105286, 2020. 53

[136] R. S. Alonso, I. Sittón-Candanedo, Óscar García, J. Prieto, and S. Rodríguez-González, "An intelligent edge-iot platform for monitoring livestock and crops in a dairy farming scenario," *Ad Hoc Networks*, vol. 98, p. 102047, 2020. 54

[137] L. Sun, Y. Li, and R. A. Memon, "An open iot framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, February 2017. 55

[138] L. Bixio, G. Delzanno, S. Rebora, and M. Rulli, "A flexible iot stream processing architecture based on microservices," *Information*, vol. 11, no. 12, p. 565, 2020. 55

[139] H. Fang, "Managing data lakes in big data era : What's a data lake and why has it became popular in data management ecosystem," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, June 2015, pp. 820–824. 55

[140] C. Madera, A. Laurent, T. L. Rouge, and A. Miralles, "How can the data lake concept influence information system design for agriculture ?" in *11th European conference dedicated to the future use of ICT in the agri-food sector, bioresource and biomass sector (EFITA 2017)*, 2017, pp. 181–182. 56

[141] I. D. López, J. F. Grass, A. Figueroa, and J. C. Corrales, "A proposal for a multi-domain data fusion strategy in a climate-smart agriculture context," *International Transactions in Operational Research*, 2020. 57

[142] E. Gallinucci, M. Golfarelli, and S. Rizzi, "A hybrid architecture for tactical and strategic precision agriculture," in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2019, pp. 13–23. 57

[143] E. Gallinucci, S. Rizzi, and M. Golfarelli, "Mo. re. farming : A hybrid architecture for tactical and strategic precision agriculture," *Data & Knowledge Engineering*, vol. 129, p. 101836, 2020. 57

[144] R. A. Neves and P. E. Cruvinel, "Model for semantic base structuring of digital data to support agricultural management," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. IEEE, 2020, pp. 337–340. 57

[145] Y. Ampatzidis, V. Partel, and L. Costa, "Agroview : Cloud-based application to process, analyze and visualize uav-collected data for precision agriculture applications utilizing artificial intelligence," *Computers and Electronics in Agriculture*, vol. 174, p. 105457, 2020. 57

[146] H. Navarro-Hellín, J. Martinez-del Rincon, R. Domingo-Miguel, F. Soto-Valles, and R. Torres-Sánchez, "A decision support system for managing irrigation in agriculture," *Computers and Electronics in Agriculture*, vol. 124, pp. 121–131, 2016. 59

[147] J. Conesa-Muñoz, J. Valente, J. Del Cerro, A. Barrientos, and A. Ribeiro, "A multi-robot sense-act approach to lead to a proper acting in environmental incidents," *Sensors*, vol. 16, no. 8, p. 1269, 2016. 59

[148] X. Léauté, "Benchmarking druid," 2014, http://druid.io/blog/2014/03/17/benchmarking-druid.html Accessed january 8, 2019. 91, 92, 93, 96

[149] T. P. Council, "Tpc-h benchmark," 2013, http://tpc.org/tpc Accessed February 1, 2019. 92

[150] A. Cuzzocrea, R. Moussa, and G. Vercelli, "An innovative lambda-architecture-based data warehouse maintenance framework for effective and efficient near-real-time olap over big data," in *Big Data – BigData 2018*, F. Y. L. Chin, C. L. P. Chen, L. Khan, K. Lee, and L.-J. Zhang, Eds. Cham : Springer International Publishing, 2018, pp. 149–165. 92

[151] D. Brickley, R. V. Guha, and B. McBride, "Rdf schema 1.1," *W3C recommendation*, vol. 25, pp. 2004–2014, 2014. 120

[152] D. Beckett and B. McBride, "Rdf/xml syntax specification (revised)," *W3C recommendation*, vol. 10, no. 2.3, 2004. 120

[153] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, P.-A. Champin, and N. Lindström, "Json-ld 1.1–a json-based serialization for linked data," Ph.D. dissertation, W3C, 2019. 120

[154] D. Beckett, "Rdf 1.1 n-triples," *World Wide Web Consortium*, 2014. 120

[155] G. Carothers, "Rdf 1.1 n-quads : A line-based syntax for rdf datasets," *W3C Recommendation*, vol. 25, 2014. 120

[156] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers, "Rdf 1.1 turtle," *World Wide Web Consortium*, 2014. 120

[157] T. Berners-Lee, "Notation 3 logic," *W3 Design Issues [online]*, pp. 2005–09, 2005. 120

[158] X. Su, J. Riekki, and J. Haverinen, "Entity notation : enabling knowledge representations for resource-constrained sensors," *Personal and Ubiquitous Computing*, vol. 16, no. 7, pp. 819–834, 2012. 120

[159] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, S. Rudolph *et al.*, "Owl 2 web ontology language primer," *W3C recommendation*, vol. 27, no. 1, p. 123, 2009. 120

[160] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "Hermit : an owl 2 reasoner," *Journal of Automated Reasoning*, vol. 53, no. 3, pp. 245–269, 2014. 121

[161] M. Stocker and M. Smith, "Owlgres : A scalable owl reasoner." in *OWLED*, vol. 432, 2008. 121

[162] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet : A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007. 121

[163] B. McBride, "Jena : A semantic web toolkit," *IEEE Internet computing*, vol. 6, no. 6, pp. 55–59, 2002. 121

[164] Q. Tong, F. Zhang, and J. Cheng, "Construction of rdf (s) from uml class diagrams," *Journal of computing and information technology*, vol. 22, no. 4, pp. 237–250, 2014. 129

[165] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016. [Online]. Available : https://www.frontiersin.org/article/10.3389/fpls.2016.01419 143

[166] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets : Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv :1704.04861*, 2017. 143

[167] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2 : Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520. 143

[168] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708. 143

[169] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710. 143

[170] M. Tan and Q. Le, "Efficientnet : Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114. 143

[171] J. Cao, H. Tang, H.-S. Fang, X. Shen, C. Lu, and Y.-W. Tai, "Cross-domain adaptation for animal pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 145

[172] M. Elmoulat, O. Debauche, S. Mahmoudi, S. A. Mahmoudi, A. Guttadauria, P. Manneback, and L. Frédéric, "Towards landslides early warnings system with fog - edge computing and artificial intelligence," *arXiv preprint arXiv :1811.04283*, 2018. 165

[173] O. Debauche, S. Mahmoudi, S. A. Mahmoudi, P. Manneback, and F. Lebeau, "A new edge architecture for ai-iot services deployment," *Procedia Computer Science*, vol. 175, pp. 10–19, 2020. 165

[174] M. Elmoulat, O. Debauche, S. Mahmoudi, S. A. Mahmoudi, P. Manneback, and F. Lebeau, "Edge computing and artificial intelligence for landslides monitoring," *Procedia Computer Science*, vol. 177, pp. 480–487, 2020. 165

[175] O. Debauche, S. Mahmoudi, P. Manneback, and A. Assila, "Fog iot for health : A new architecture for patients and elderly monitoring." *Procedia Computer Science*, vol. 160, pp. 289–297, 2019. 166

[176] Q. H. Dang, "Secure hash standard," NIST, Tech. Rep., 2015. 216

[177] M. J. Dworkin, "Sha-3 standard : Permutation-based hash and extendable-output functions," NIST, Tech. Rep., 2015. 216

[178] I. 10118, "It security techniques — hash-functions — part 3 : Dedicated hash-functions," 2018, https://www.iso.org/standard/67116.html Accessed December 16, 2020. 216, 217

[179] R. Rivest, "Rfc1321 : The md5 message-digest algorithm," 1992. 216

[180] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2 : The memory-hard function for password hashing and other applications. 2015," *Specification available https ://www. cryptolux. org/index. php/Argon2*, 2018. 217

[181] F. Denis, "The xchacha20-poly1305 construction," 2018. 218

Quatrième partie

Annexes

Annexe 1 : Définitions

Dans le cadre de cette thèse de doctorat, il nous est apparu important de préciser certaines notions et concepts afin d'éviter toutes ambigüités.

Nous entendons par **temps réel**, un temps de traitement inférieur à la seconde et par **quasi-temps réel** un temps de traitement n'excédant pas les quelques secondes.

Nous rappelons également la distinction entre SQL et NoSQL. Le **SQL** est un langage informatique normalisé servant à exploiter des bases de données relationnelles où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables et respectant les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité). Le **NoSQL** (*Not only SQL*) désigne une famille de systèmes de gestion de base de données qui s'écarte du paradigme classique des bases relationnelles par un non respect strict des propriétés ACID. Dans ce type de base de données, les données sont organisées sous forme de documents, de graphes, de colonnes, ou d'associations clé-valeur.

Le **Cloud** est un ensemble de ressources matérielles, de raccordements réseau et de logiciels fournissant des services accessibles et utilisable de n'importe où dans le monde.

Le **Cloud Computing** ou informatique en nuage conciste à accéder à des services informatiques (serveurs, stockage, mise en réseau, logiciels) proposés par un fournisseur à l'aide d'Internet.

Le **Fog Computing** ou informatique en brouillard consiste à exploiter des applications et des infrastructures de traitement et de stockage de proximité, servant d'intermédiaire entre des objets connectés et une architecture cloud.

Le **Edge Computing** ou Informatique en périphérie du réseau est une technique qui consiste à traiter les données à la périphérie du réseau, près de la source des données c'est-à-dire sur les microcontrolleurs. L'utilisation du Edge Computing diminue signi-

ficativement les volumes de données à transmettre, le trafic réseau qui en résulte, les coûts de transmission, diminue de la latence et l'amélioration de la qualité du service (QoS).

L'est un métalangage informatique de balisage générique qui permet de décrire n'importe quel domaine grâce à son extensibilité. Le **CSV** est un fichier texte dans lequel les champs sont séparés par des virgules. Le **Delimiter-Separated Values (DSV)** est un format de fichier texte dans lequel les données sont séparées par un plusieurs délimiteurs. Le **Tab-separated values (TSV)** est une format de fichier texte dans lequel les données sont séparés par des tabulations. Le **JavaScript Object Notation (JSON)** est un format de donnée textuelle qui permet une représentation compacte des données basée sur la notation des objets du javascript et de manière structurée comme le permet le eXtensible Markup Language (XML) tout en étant lisible pour un humain.

Les **données massives** également appelées mégadonnées (*Big data*) désignent des données produites par les nouvelles technologies tant personnelles que professionnelles. Elles sont caractérisées par les 5 "V" (L'explosion du **Volume** de données stockées dans une grande Variété de formats nécessite des algorithmes performant pour les traiter avec une **Vitesse** (*Vélocité*) élevée grâce a des logiciels performants et d'une grande puissance de calcul. Les données sont entachées d'erreurs, il est par conséquence nécessaire de prendre des précautions vis à vis de leur **Véracité**. Le traitement de ces masses colossale de données n'a sens que si elle apporte une **Valeur** ajoutée.).

L'architecture logicielle **REpresentational State Transfer (REST)** définit un ensemble de contraintes qui peuvent être utilisées pour mettre en place des services web (*Webservices*). Ces règles sont : (**1**) L'utilisation de l'URI pour identifier les ressources ; (**2**) Les verbes HTTP (GET, POST, PUT, DELETE) permettent d'identifier les opérations ; (**3**) Les réponses HTTP sont utilisées pour représenter les ressources ; (**4**) Les liens permettent de mettre en relations les ressources ; (**5**) Un paramètre est utilisé comme jeton d'authentification. Les services web qui respectent l'architecture logicielle REST sont qualifiés de RESTful.

On entendu par **architecture ou application monolithique**, un programme contenant l'ensemble du code et des fonctionnalités et dans lequel les composants logiciels sont fortement couplés.

La **latence** désigne le temps que mets un paquet pour transiter dans un réseau informatique de la source vers son destinataire. La **gigue** ou jitter en anglais désigne la variation de cette latence au fil du temps. Les effets de la gigue peuvent être atténués pour améliorer la qualité de service (QoS)en plaçant une mémoire tampon au niveau du récepteur mais cela se traduit par une augmentation de la latence.

Les **drones** ou Unmanned Aerial Vehicle (UAV) en anglais sont des aéronefs sans passager pilotés à distance ou pouvant voler de manière autonome. Les **robots** ou Unmanned Ground Vehicule (UGV) sont quant à eux des véhicules terrestres sans passagers qui sont opérés a distance ou évolue de manière indépendante sur le terrain.

La **virtualisation** utilise des "imitations" de serveur physique appelées machine virtuelle. Chaque machine virtuelle contient son propre système d'exploitation et ses applications. L'ensemble des machines virtuelles sont gérée par un hyperviseur qui s'exécute sur un système d'exploitation hôte. La **conteneurisation** permet de partager un système hôte unique et s'affranchir de la nécessité d'installer un système d'exploitation pour chaque instance. Elle ne nécessite pas l'installation d'un Operating System (OS), les conteneurs sont par conséquent plus petits, facile à migrer, sauvegarder, restaurer, ou à télécharger. L'hyperviseur est remplacé par un moteur de conteneurisation assurant l'isolation des conteneurs les uns des autres bien qu'en pratique l'isolation entre machines virtuelles est plus importante qu'entre conteneurs.

Le **Smart Farming** est la combinaison de l'agriculture classique et des solutions numériques et techniques en vue d'en d'améliorer l'efficacité.

L'**agriculture protégé** cultive dans des plantes à haute valeur ajoutées ou horticoles dans des milieux contrôlés comme les serres et les phytotrons pour optimiser le rendement. L'agriculture de précision a pour objectif d'optimiser les rendements de cultures en champs et de minimiser les investissements en tenant compte de la variabilité inter et intra parcellaire.

Un **entrepôt de données** ou EDD, également appelé base de données décisionnelle (BDD) ou encore Data Warehouse (DWH) est une base de données relationnelle hébergée sur un serveur dans un Data Center ou dans le Cloud. Il recueille des données de sources variées et hétérogènes dans le but principal de soutenir l'analyse et faciliter le processus de prise de décision de l'entreprise. Au sein d'un entrepôt de données, les données sont non volatiles, c'est-à-dire qu'elles ne disparaissent pas et ne change pas au fil du temp et des traitement, et sont horodatées et organisées par sujet.

L'ontologie **Semantic Sensor Network (SSN)** est une ontologie permettant de décrire les capteurs et leurs observations, les procédures impliquées, les caractéristiques intéressantes étudiées, les échantillons utilisés et les propriétés observées, ainsi que les actionneurs. Le SSN suit une architecture de modularisation horizontale et verticale en incorporant une ontologie de base légère mais autonome appelée SOSA (Sensor, Observation, Sample and Actuator) pour ses classes et propriétés élémentaires. Avec leur por-

tée différente et leurs divers degrés d'axiomatisation, le SSN et le SOSA sont capables de prendre en charge un large éventail d'applications et de cas d'utilisation, notamment l'imagerie satellitaire, la surveillance scientifique à grande échelle, les infrastructures industrielles et domestiques, la détection sociale, la science citoyenne, l'ingénierie ontologique axée sur l'observation et le Web des objets.

Annexe 2 : Publications

Chapitre de livre

1. Debauche Olivier, Mahmoudi Saïd, Manneback Pierre, "A new collaborative platform for Covid-19 benchmark datasets", in Garg Lalit, Chakraborty Chinmay, Mahmoudi Saïd, Sohmen Victor (eds), "Intelligent Healthcare Informatics for Fighting the COVID-19 and Other Pandemics and Epidemics, EAI/Springer Innovations in Communication and Computing, Springer, 2022. doi : 10.1007/978-3-030-72752-9_12.

Publications dans des journaux

2. Debauche Olivier, Mahmoudi Said, Andriamandroso Andriamasinoro Lalaina Herinaina, Manneback Pierre, Bindelle Jérôme, Lebeau Frédéric, "Cloud Services Integration for Farm Animals' Behavior Studies Based on Smartphones as Activity Sensors" in *Journal of Ambient Intelligence and Humanized Computing*, 2019 : **10(12)**, 4651–4662. doi : 10.1007/s12652-018-0845-9.
3. Debauche Olivier, Mahmoudi Saïd, Manneback Pierre, Lebeau Frédéric, "Cloud Architecture For Plant Phenotyping Research" *Concurrency and Computation : Practice and Experience*, 2020 : **32(17)**, e5661, 1–16. doi : 10.1002/cpe.5661.
4. Debauche Olivier, Trani Jean-Philippe, Mahmoudi Saïd, Manneback Pierre, Bindelle Jérôme, Lebeau Frédéric, "Data Management and Internet of Things : A Methodological Review in Smart Farming", *Internet of Things*, 2021 : **14**, 100378. doi : 10.1016/j.iot.2021.100378.
5. Debauche Olivier, Mahmoudi Saïd, Manneback Pierre, "Cloud and Distributed Architectures for Data Management in Agriculture 4.0 : Review and Future Trends", *Journal of King Saud University - Computer and Information Sciences*, 2021. doi : 10.1016/j.jksuci.2021.09.015.
6. Elmoulat Meryem, Debauche Olivier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Guttadauria Adriano, Manneback Pierre, Lebeau Frédéric, "Towards Landslides Early Warning System With Fog - Edge Computing And Artificial Intelligence", *Journal of Ubiquitous Systems & Pervasive Networks*, 2021 : **15(1)**, 11–17. doi : 10.5383/JUSPN.15.02.002.

7. Debauche Olivier, Elmoulat Meryem, Mahmoudi Saïd, Bindelle Jérôme, "Farm Animals' Behaviors and Welfare Analysis with AI Algorithms : A Review" *Revue d'Intelligence Artificielle*, 2021 : **35(3)**, 243–253. doi : 10.18280/ria.350308.
8. Tadrist Nassima, Debauche Olivier, Mahmoudi Saïd, Guttadauria Adriano, "Towards Low-Cost IoT and LPWAN-Based Flood Forecast and Monitoring System", *Journal of Ubiquitous Systems & Pervasive Networks*, 2022 : **17(1)**, 43–49. doi : 10.5383/JUSPN.17.01.006.
9. Nkamla Penka Jean Bertin, Mahmoudi Saïd, Debauche Olivier, "An Optimized Kappa Architecture for IoT Data Management in Smart Farming", *Journal of Ubiquitous Systems & Pervasive Networks*, 2022 : **17(2)**, 59–65. doi : 10.5383/JUSPN.17.02.002.
10. Debauche Olivier, Mahmoudi Saïd, Guttadauria Adriano, "A New Edge Computing Architecture for IoT and Multimedia Data Management", *Information*, 2022 : **13(2)**, 89. doi : 10.3390/info13020089.

Actes de conférences publiés dans des revues indexées

11. Debauche Olivier, Mahmoudi Said, Manneback Pierre, Massinon Mathieu, Tadrist Nassima, Lebeau Frédéric, Mahmoudi Sidi, "Cloud architecture for digital phenotyping and automation" in "CloudTech 2017 : The 3rd International Conference on Cloud Computing Technologies and Applications", Rabat, Maroc, 2017. doi : 10.1109/CloudTech.2017.8284718.
12. Debauche Olivier, El Moulat Meryem, Mahmoudi Said, Manneback Pierre, Lebeau Frédéric, "Irrigation pivot-center connected at low cost for the reduction of crop water requirements" in "International Conference on Advanced Communication Technologies and Networking (CommNet'18)", Marrakech, Morocco, 2018. doi : 10.1109/COMMNET.2018.8360259.
13. Elmoulat Meryem, Debauche Olivier, Ait Brahim Lahcen, "Monitoring System Using Internet of Things for Potential Landslides" in *Procedia Computer Science*, **132**, 26–34. doi : 10.1016/j.procs.2018.07.140.
14. Debauche Olivier, Mahmoudi Sidi, Mahmoudi Said, Manneback Pierre, "Cloud Platform using Big Data and HPC Technologies for Distributed and Parallels Treatments" in *Procedia Computer Science*, 2018 : **141**, 112–118. doi : 10.1016/j.procs.2018.10.156.
15. Debauche Olivier, Mahmoudi Said, Andriamandroso Andriamasinoro Lalaina Herinaina, Manneback Pierre, Bindelle Jérôme, Lebeau Frédéric, "Web-based cattle behavior service for researchers based on the smartphone inertial central" in *Procedia Computer Science*, 2017 : **110**, 110–116. doi : 10.1016/j.procs.2017.06.127.
16. Debauche Olivier, El Moulat Meryem, Mahmoudi Said, Boukraa Slimane, Manneback Pierre, Lebeau Frédéric, "Web Monitoring of Bee Health for Researchers

and Beekeepers Based on the Internet of Things" in *Procedia Computer Science*, 2018 : **130**, 991—998. doi : 10.1016/j.procs.2018.04.103.

17. Debauhe Olivier, Mahmoudi Saïd, Manneback Pierre, Assila Abdessamad, "Fog-IoT-Cloud for Health : A new Architecture for Patients and Elderly Monitoring" in *Procedia Computer Science*, 2019 : **160**, 289–297. doi : 10.1016/j.procs.2019.11.087.
18. Ait Abdelouahid Rachida, Debauche Olivier, Mahmoudi Saïd, Marzak Abdelaziz, Manneback Pierre, Lebeau Frédéric, "Open Phytotron : A New IoT Device for Home Gardening" in "The 5th IEEE International Conference on Cloud Computing and Artificial Intelligence : Technologies and Applications", Marrakech, Morocco, 2020. doi : 10.1109/CloudTech49835.2020.9365892.
19. Debauche Olivier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Manneback Pierre, Bindelle Jérôme, Lebeau Frédéric, "Edge Computing for Cattle Behavior Analysis", in "Second international conference on Embedded & Distributed Systems (EDiS'2020)", Oran, Algeria, 2020. doi : 10.1109/EDiS49545.2020.9296471.
20. Debauche Olivier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Manneback Pierre, Lebeau Frédéric, "A new Edge Architecture for AI-IoT services deployment", in *Procedia Computer Science*, 2020 : **175**, 10–19. doi : 10.1016/j.procs.2020.07.006.
21. Debauche Olivier, Mahmoudi Said, Mahmoudi Sidi Ahmed, Manneback Pierre, Bindelle Jérôme, Lebeau Frédéric, "Edge Computing and Artificial Intelligence for Real-time Poultry Monitoring", in *Procedia Computer Science*, 2020 : **175**, 534–541 doi : 10.1016/j.procs.2020.07.076.
22. Debauche Olivier, Mahmoudi Said, Mahmoudi Sidi Ahmed, Manneback Pierre, Lebeau Frédéric, "Edge Computing and Artificial Intelligence Semantically Driven. Application to a Climatic Enclosure", in *Procedia Computer Science*, 2020 : **175**, 542–547. doi : 10.1016/j.procs.2020.07.077.
23. Ait Abdelouahid Rachida, Debauche Olivier, Mahmoudi Said, Marzak Abdelaziz, Manneback Pierre, Lebeau Frédéric, "Smart Nest Box : IoT Based Nest Monitoring In Artificial Cavities", in "The 3rd International Conference on Advanced Communication Technologies and Networking (CommNet 2020)", Marrakech, Morocco, 2020. doi : 10.1109/CommNet49926.2020.9199624.
24. Debauche Olivier, Ait Abdelouahid Rachida, Mahmoudi Saïd, Moussaoui Yahya, Marzak Abelaziz, Manneback Pierre, "RevoCampus : a distributed open source and low-cost smart campus", in "The 3rd International Conference on Advanced Communication Technologies and Networking (CommNet 2020)", Marrakech, Morocco, 2020. doi : 10.1109/CommNet49926.2020.9199640.
25. Elmoulat Meryem, Debauche Olivier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Manneback Pierre, Lebeau Frédéric, "Edge Computing and Artificial Intelligence for Landslides Monitoring", in *Procedia Computer Science*, 2020 : **177**, 40–48. doi : 10.1016/j.procs.2020.10.066.

26. Debauche Olivier, Mahmoudi Saïd, Manneback Pierre, "A new Collaborative Platform for Research in Smart Farming", in "The 6th International workshop on Big Data and Networks Technologies (BDNT 2020)", in *Procedia Computer Sicence*, 2020 : **177**, 450–455. doi : 10.1016/j.procs.2020.10.061.
27. Debauche Olivier, Mahmoudi Saïd, Elmoulat Meryem, Mahmoudi Sidi Ahmed, Manneback Pierre, Lebeau Frédéric, "Edge AI-IoT Pivot Irrigation, Plant Diseases and Pests Identification", in "The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)", in *Procedia Computer Science*, 2020 : **177**, 480–487. doi : 10.1016/j.procs.2020.10.009.
28. Nolack Fote Fabrice, Roukh Amine, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Debauche Olivier, "Towards a Big Data Knowledge-Base Management System for Precision Livestock Farming", in *Procedia Computer Science*, 2020 : **177**, 136–142. doi : 10.1016/j.procs.2020.10.021.
29. Nkamla Penka Jean Bertin, Mahmoudi Saïd, Debauche Olivier, "A new Kappa Architecture for IoT Data Management in Smart Farming", in *Procedia Computer Science*, 2021 : **191**, 17–24. doi : 10.1016/j.procs.2021.07.006.
30. Ait Abdelouahid Rachida, Debauche Olivier, Marzak Abdelaziz, "Internet of Things : a new Interoperable IoT Platform. Application to a Smart Building", in *Procedia Computer Science*, 2021 : **191**, 511–517. doi : 10.1016/j.procs.2021.07.066.

Actes de conférences

31. Debauche Olivier, Mahmoudi Said, Manneback Pierre, Tadrist Nassima, Bindelle Jérôme, Lebeau Frédéric, "Improvement of battery life of iPhones Inertial Measurement Unit by using edge computing" in "International Symposium on Computer Sciences and Applications", Errachidia, Morocco, 2017.
32. Debauche Olivier, Ait Abdelouahid Rachida, , Moussaoui Yahya, "An open source and low-cost Smart Auditorium" in "The 2nd international Workshop on innovative Smart city technologies 2020", Casablanca, Morocco, 2020.

Posters

33. Roland François, Jovicic Nikola, Debauche Olivier, Bette Sébastien, Moeyaert Véronique, "Experimental setup to monitor environment impact on LoRa transmissions", UMONS Infortech Day 2019, Mons, Belgique, 2019. doi : 10.13140/RG.2.2.23230.74564.
34. Roland François, Debauche Olivier, Filippi Enrico, Bette Sébastien, "Geolocation of Tools on Construction Sites and LoRa Performance" in "Mardi des Chercheurs 2019 (MdC2019)", Mons, Belgique, 2019. doi : 10.13140/RG.2.2.27259.16167/1.

35. Debauche Olivier, Mahmoudi Saïd, Manneback Pierre, Lebeau Frédéric, "Web-based animal behavior study service for researchers based on the smartphone inertial central" in "Huitième édition montoise du MdC : le Mardi des Chercheurs 2017", UMons, Belgique, 2017. doi : 10.13140/RG.2.2.26856.88328/1.

Publications de vulgarisation

36. Debauche Olivier, Mahmoudi Saïd, "Vaches, pâturage et environnement", Polytech News, **57**, 35, 2019.
37. Debauche Olivier, Mahmoudi Saïd, "La déperdition des abeilles, le sos d'un environnement en détresse", Polytech News, **57**, 36, 2019.
38. Debauche Olivier, Mahmoudi Sidi Ahmed, "Le pivot intelligent surveille et optimise les besoins des cultures", Polytech News, **57**, 7, 2019.
39. Debauche Olivier, Bouazzati Ibtissam, "L'unité de croissance végétale automatisée", Polytech News, **55**, 20, 2017.
40. Mahmoudi Saïd, Debauche Olivier, El Adoui Mohammed, Belarbi Mohammed Amin, "L'Internet des objets met la lumière dans tous ses états". Polytech News, **55**, 18, 2017.

Publications connexes

41. Debauche Olivier, Mahmoudi Saïd, Belarbi Mohammed Amin, El Adoui Mohammed, Mahmoudi Sidi, "Internet of Things : learning and practices", in "International Conference on Advanced Communication Technologies and Networking (CommNet'18)", Marrakech, Morocco, 2018. doi : 10.1109/COMMNET.2018.8360247.
42. Debauche Olivier, Mahmoudi Said, Mahmoudi Sidi, "Internet of Things : learning and practices. Application to Smart City", in "The 4th IEEE International Conference on Cloud Computing Technologies and Applications", Brussel, Belgique, 2019. doi : 10.1109/CloudTech.2018.8713337.
43. Debauche Olivier, Mahmoudi Saïd, Moussaoui Yahya, "Internet of Things Learning : a Practical Case for Smart Building automation", in "The 5th IEEE International Conference on Cloud Computing and Artificial Intelligence : Technologies and Applications", Marrakech, Maroc, 2020. doi : 10.1109/CloudTech49835.2020.9365920.
44. Jemai Bornia, Frihida Ali, Debauche Olivier, Mahmoudi Sidi Ahmed, Manneback, Pierre, "Deep Learning and Tensorflow for Tracking People's Movements in a Video", in "The 5th IEEE International Conference on Cloud Computing and Artificial Intelligence : Technologies and Applications", Marrakech, Maroc, 2020. doi : 10.1109/CloudTech49835.2020.9365886.

45. Lessage Xavier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Laraba Souhaib, Debauche Olivier, Belarbi Mohammed Amin, "Chest X-ray images analysis in Deep Convolutional Neural Networks (CNN) for COVID-19 detection", in Garg Lalit, Chakraborty Chinmay, Mahmoudi Saïd, Sohmen Victor (eds), "Intelligent Healthcare Informatics for Fighting the COVID-19 and Other Pandemics and Epidemics, EAI/Springer Innovations in Communication and Computing, Springer, 2022. doi : 10.1007/978-3-030-72752-9_21.

46. Lessage Xavier, Mahmoudi Saïd, Mahmoudi Sidi Ahmed, Laraba Souhaib, Debauche Olivier, Belarbi Mohammed Amin, "Deep Convolutional Neural Networks (CNN) for COVID-19 classification in chest X-ray images with Explainable Artificial Intelligence (XAI)" in *International Journal of Computer Assisted Radiology and Surgery*, 2021 : 16 (Suppl 1) - S1–S119. doi : 10.1007/s11548-021-02375-4.

47. Najii Mohammed Amine, El Filali Sanaa, Aarika Kawtar, Benlahmar El Habib, Ait Abelouhahid Rachida, Debauche Olivier, "Machine Learning Algorithms For Breast Cancer Prediction And Diagnosis", in *Procedia Computer Sicence*, 2021 : **191**, 136–142. doi : 10.1016/j.procs.2021.07.062.

48. Najii Mohammed Amine, El Filali Sanae, Bouhlal Meriem, Benlahmar El Habib, Ait Abdelouhahid Rachida, Debauche Olivier, "Breast Cancer Prediction and Diagnosis through a New Approach based on Majority Voting Ensemble Classifier", in *Procedia Computer Sicence*, 2021 : **191**, 481–486. doi : 10.1016/j.procs.2021.07.061.

49. Oqaidi Mohammed, Ait Abdelouahid Rachida, Debauche Olivier, Marzak Abdelaziz, "An open source and low-cost Smart Auditorium", in *Procedia Computer Science*, 2021 : **191**, 518–523. doi : 10.1016/j.procs.2021.07.076.

A new collaborative platform for Covid-19 benchmark datasets.

Abstract : Fast and efficient collaboration between researcher is a crucial task to advance effectively in Covid-19 research. In this chapter, we present a new collaborative platform allowing to exchange and share both medical benchmark datasets and developed applications rapidly and securely between research teams. The aim of this platform is to facilitate and encourage the exploration of new fields of research. This platform implements proven data security techniques allowing to guarantee confidentiality, mainly Argon2id password hashing algorithm, anonymization, expiration of forms, and datasets double encryption and decryption with AES 256-GCM and XChaCha20Poly1305 algorithms. Our platform has been successfully tested as part of a project aiming to develop artificial intelligence algorithms for imagery based upon detection of Covid-19. Indeed, by using our collaborative platform allowed us to advance more quickly on the development of some artificial intelligence algorithms which mainly achieve both segmentation and classification of CT-Scan and X-Ray images of patients' lungs and chests.

Keywords : Covid-19, medical data, sharing data, exchange data, medical research

Cloud Services Integration for Farm Animals' Behavior Studies Based on Smartphones as Activity Sensors.

Abstract : Smartphones, particularly iPhone, can be relevant instruments for researchers in animal behavior because they are readily available on the planet, contain many sensors and require no hardware development. They are equipped with high performance Inertial Measurement Units (IMU) and absolute positioning systems analyzing users' movements, but they can easily be diverted to analyze likewise the behaviors of domestic animals such as cattle. The study of animal behavior using smartphones requires the storage of many high frequency variables from a large number of individuals and their processing through various relevant variables combinations for modeling and decision-making. Transferring, storing, treating and sharing such an amount of data is a big challenge. In this paper, a lambda cloud architecture innovatively coupled to a scientific sharing platform used to archive, and process high-frequency data are proposed to integrate future developments of the Internet of Things applied to the monitoring of domestic animals. An application to the study of cattle behavior on pasture based on the data recorded with the IMU of iPhone 4s is exemplified. Performances comparison between iPhone 4s and iPhone 5s is also achieved. The package comes also with a web interface to encode the actual behavior observed on videos and to synchronize observations with the sensor signals. Finally, the use of Edge computing on the iPhone reduced by 43.5% on average the size of the raw data by eliminating redundancies. The limitation of the number of digits on individual variable can reduce

data redundancy up to 98.5%.

Keywords : Animals' behavior, Smart agriculture, IMU, iPhone, Lambda architecture, Precision livrestock Farming

Cloud Architecture For Plant Phenotyping Research.

Abstract : Digital phenotyping is an emergent science mainly based on imagery techniques. The tremendous amount of data generated needs important cloud computing for their processing. The coupling of recent advance of distributed databases and cloud computing offers new possibilities of big data management and data sharing for the scientific research. In this paper, we present a solution combining a lambda architecture built around Apache Druid and a hosting platform leaning on Apache Mesos. Lambda architecture has already proved its performance and robustness. However, the capacity of ingesting and requesting of the database is essential and can constitute a bottleneck for the architecture, in particular, for in terms of availability and response time of data. We focused our experimentation on the response time of different databases to choose the most adapted for our phenotyping architecture. Apache Druid has shown its ability to respond to typical queries of phenotyping applications in times generally inferior to the second.

Keywords : digital phenotyping, plant phenotyping, lambda architecture, research application hosting platform, cloud architecture.

Data Management and Internet of Things : A Methodological Review in Smart Farming.

Abstract : Introduction. In the field of research, we are familiar to employ ready-to-use commercial solutions. This bibliographic review highlights the various technological paths that can be used in the context of agriculture digitalization and illuminates the reader on the capacities and limits of each one.

Literature. Based on a literature review that we conducted, we describe the main components of the Internet of Things. Also, we analyzed the different technological pathways used by researchers to develop their projects. Finally, these versatile approaches are summarized in the form of tables and a methodological flowchart of communication protocols choices.

Conclusions. In this article, we propose a methodology and a reflection on the technological choices and their implication on the durability and valorization of research

projects in the field of smart agriculture.

Keywords : Internet of Things, network protocols, technological selection methodology, cloud architecture, security, smart farming.

Cloud and Distributed Architectures for Data Management in Agriculture 4.0 : Review and Future Trends.

Abstract : The Agriculture 4.0 also called Smart Agriculture or Smart Farming is at the origin of the production of a huge amount of data that must be collected, stored and processed in very short times. Processing this massive quantity of data needs to use specific infrastructure that use adapted IoT architectures. Our review offers a comparative panorama of Central Cloud, Distributed Cloud Architectures, Collaborative Computing Strategies, and new trends used in the context of Agriculture 4.0. In this review, we try to answer 4 research questions : (1) Which storage and processing architectures are best suited to Agriculture 4.0 applications and respond to its peculiarities ? (2) Can generic architectures meet the needs of Agriculture 4.0 application cases ? (3) What are the horizontal development possibilities that allow the transition from research to industrialization ? (4) What are the vertical valuations possibilities to move from algorithms trained in the cloud to embedded or stand-alone products ? For this, we compare architectures with 8 criteria (User Proximity, Latency & Jitter, Network stability, high throughput, Reliability, Scalability, Cost Effectiveness, Maintainability) and analyze the advantages and disadvantages of each of them. Our review offers a comparative panorama of Central Cloud, Distributed Cloud Architectures, and new trends used in the context of Agriculture 4.0.

Keywords : Agriculture 4.0, Smart Farming, Smart Agriculture, Lambda Architecture, Kappa Architecture, Micro-Service Architecture, Data Lake.

Towards Landslides Early Warning System With Fog - Edge Computing And Artificial Intelligence.

Abstract : Landslides are phenomena that cause significant human and economic losses. Researchers have investigated the prediction of high landslides susceptibility with various methodologies based upon statistical and mathematical models, in addition to artificial intelligence tools. These methodologies allow to determine the areas that could present a serious risk of landslides. Monitoring these risky areas is particularly important for developing an Early Warning Systems (EWS). As matter of fact, the variety of landslides' types make their monitoring a sophisticated task to accom-

plish. Indeed, each landslide area has its own specificities and potential triggering factors ; therefore, there is no single device that can monitor all types of landslides. Consequently, Wireless Sensor Networks (WSN) combined with Internet of Things (IoT) allow to set up large-scale data acquisition systems. In addition, recent advances in Artificial Intelligence (AI) and Federated Learning (FL) allow to develop performant algorithms to analyze this data and predict early landslides events at edge level (on gateways). These algorithms are trained in this case at fog level on specific hardware. The novelty of the work proposed in this paper is the integration of Federated Learning based on Fog-Edge approaches to continuously improve prediction models.

Keywords : Landslides Susceptibility, IoT, Artificial Intelligence, Early Warning System, Landslides Monitoring, Edge AI, Edge IoT.

Farm Animals' Behaviors and Welfare Analysis with IA Algorithms : A Review.

Abstract : Numerous bibliographic reviews related to the use of AI for the behavioral detection of farm animals exist, but they only focus on a particular type of animal. We believe that some techniques were used for some animals that could also be used for other types of animals. The application and comparison of these techniques between animal species are rarely done. In this paper, we propose a review of machine learning approaches used for the detection of farm animals' behaviors such as lameness, grazing, rumination, and so on. The originality of this paper is matched classification in the midst of sensors and algorithms used for each animal category. First, we highlight the most implemented approaches for different categories of animals (cows, sheep, goats, pigs, horses, and chickens) to inspire researchers interested to conduct investigation and employ the methods we have evaluated and the results we have obtained in this study. Second, we describe the current trends in terms of technological development and new paradigms that will impact the AI research. Finally, we critically analyze what is done and we draw new pathways of research to advance our understanding of animal's behaviors.

Keywords : animal behavior, machine learning, artificial intelligence, livestock, cow, sheep, pig, chicken.

Towards Low-Cost IoT and LPWAN-Based Flood Forecast and Monitoring System.

Abstract : The recent floods have shown that the classic monitoring systems for watercourses are no longer adapted because other phenomena such as the insufficient capacity and/or obstruction of drainage networks, the modification of cultivation practices and rotations, the increase in the size of plots linked to the reparcelling, the urbanization of floodable areas, etc. The combination of all these causes, plus the modification of the water regime, implies an increase in the risk of flooding and an adapted monitoring that is no longer limited to watercourses in order to give early warning of the risk of flooding by runoff. The Internet of Things (IoT) and the availability of microcontrollers and sensors with low data rates and long ranges, as well as low-power wide area networks (LPWANs), allow for much more advanced monitoring systems.

Keywords : Monitoring System, Warning System, Flood, Runoff, Moody Flood, Flash Flood, Runoff Flooding, Sewer, IoT, LPWAN, LoRaWan, NB-IoT.

An Optimized Kappa Architecture for IoT Data Management in Smart Farming.

Abstract : Agriculture 4.0 is a domain of IoT in full growth which produces large amounts of data from machines, robots, and sensors networks. This data must be processed very quickly, especially for the systems that need to make real-time decisions. The Kappa architecture provides a way to process Agriculture 4.0 data at high speed in the cloud, and thus meets processing requirements. This paper presents an optimized version of the Kappa architecture allowing fast and efficient data management in Agriculture. The goal of this optimized version of the classical Kappa architecture is to improve memory management and processing speed. the Kappa architecture parameters are fine tuned in order to process data from a concrete use case. The results of this work have shown the impact of parameters tweaking on the speed of treatment. We have also proven that the combination of Apache Samza with Apache Druid offers the better performances.

Keywords : Agriculture 4.0, IoT, Internet of Things, Kappa Architecture, Smart Farming, Smart Agriculture.

A New Edge Computing Architecture for IoT and Multimedia Data Management.

Abstract : The Internet of Things and multimedia devices generate a tremendous amount of data. The transfer of this data to the cloud is a challenging problem because of the congestion at the network level, and therefore processing time could be too long when we use a pure cloud computing strategy. On the other hand, new applications requiring the processing of large amounts of data in real time have gradually emerged, such as virtual reality and augmented reality. These new applications have gradually won over users and developed a demand for near real-time interaction of their applications, which has completely called into question the way we process and store data. To address these two problems of congestion and computing time, edge architecture has emerged with the goal of processing data as close as possible to users, and to ensure privacy protection and responsiveness in real-time. With the continuous increase in computing power, amounts of memory and data storage at the level of smartphone and connected objects, it is now possible to process data as close as possible to sensors or directly on users devices. The coupling of these two types of processing as close as possible to the data and to the user opens up new perspectives in terms of services. In this paper, we present a new distributed edge architecture aiming to process and store Internet of Things and multimedia data close to the data producer, offering fast response time (closer to real time) in order to meet the demands of modern applications. To do this, the processing at the level of the producers of data collaborate with the processing ready for the users, establishing a new paradigm of short supply circuit for data transmission inspired of short supply chains in agriculture. The removing of unnecessary intermediaries between the producer and the consumer of the data improves efficiency. We named this new paradigm the Short Supply Circuit Internet of Things (SSCIoT).

Keywords : Edge Computing, image analysis, Internet of Things, multimedia management, A2IoT.

Cloud architecture for digital phenotyping and automation.

Abstract : Digital phenotyping presents a very important tool for scientists to measure with high accuracy the effects of external phenomena on plant development. Plant phenotyping is mainly based on imaging techniques. However, the number of images and parameters used to store and treat these parameters are continuously growing. Consequently, the high-throughput of data and the need of specific treatment in real or near real-time requires a large quantity of resources. Moreover, the increasing amount of particular phenotyping case studies needs the development of specific application. Cloud architectures offers means to store a wide range of numerous data and host a

large quantity of specific software to process these data. In this paper, we propose a new approach that shows how logic synthesis works to match digital phenotyping need and cloud possibilities in a lambda cloud architecture in order to store and treat this important amount of data in real time. We also suggest a data platform allowing to host applications and access to the stored data within the lambda architecture. The present application platform allows to use several frameworks with a fine-grained resource use of the cluster. Finally, we develop a case study in a controlled environment system (growth chamber) where we grow basil plants.

Keywords : cloud, lambda architecture, digital phenotyping, 3D plant model, phytotron, application platform

Irrigation pivot-center connected at low cost for the reduction of crop water requirements.

Abstract : Irrigation, particularly pivot-center, is widely used around the world to fill the need of crop watering. This method of irrigation has a low efficiency compared to other methods of irrigation such as drip systems and generally they use water without consider the real need of plants. In this paper we propose an automation system based on the Internet of Things (IoT), Geographic Information System (GIS) and quasi real-time in the cloud of water requirements to improve the efficiency of water use. Indeed, each segment of the pivot-center moves at a different speed compared to others; thus, must be individually controlled to optimize the yield of irrigation. Moreover, it necessary to integrate factors such as stage of crops' development, heterogeneity of soil, runoff, drainage, soil components, nutrients and moisture content. In this paper we develop a complete system integrating sensors, GIS, Internet of Things and cloud computing. This approach allows to automate fine-grained the consumption of water without decreasing the yield. In addition to that, the collect of data and the soil moisture measurement will allow to adapt coefficient of evapotranspiration to local weather without having to resort to lysimetric measures. The proposed architecture allows to store and treat real-time, time series data and low-priority data such as 3D images used in digital phenotyping field which are treated with batch processing.

Keywords : smart farming, smart environment, precision agriculture, connected irrigation, water requirements

Monitoring System Using Internet of Things for Potential Landslides.

Abstract : The North-Western RIF of Morocco is considered as one of the most mountainous zone in the Middle East and North Africa. This area is more serious in the corridor faults region, where the recent reactivation of those tectonic layering may greatly contribute to the triggering of landslides. The consequences of this phenomenon can be enormous property damage and human casualties. Furthermore, this disaster can disrupt progress and destroy developmental efforts of government, and often pushing nations back by many years. In our previous works of Tetouan-Ras-Mazari region, we identified the areas that are prone to landslides by different methods like Weights of Evidence (WofE) and Logistic Regression (LR). In fact, these zones are built and susceptible. Undoubtedly, the challenge to save human lives is vital. For this reason, we develop a robust monitoring model as part of an alert system to evacuate populations in case of imminent danger risks. This model is ground-based remote monitoring system consist of more than just field sensors ; they employ data acquisition units to record sensor measurements, automated data processing, and display of current conditions usually via the Internet of Things (IoT). To sum up, this paper outlines a new approach of monitoring to detect when hillslopes are primed for sliding and can provide early indications of rapid and catastrophic movement. It reports also continuous information from up-to-the-minute or real-time monitoring, provides prompt notification of landslide activities, advances our understanding of landslide behaviors, and enables more effective engineering and planning efforts.

Keywords : Landslides, Internet of Things, Lambda Architecture, Risk Monitoring

Cloud Platform using Big Data and HPC Technologies for Distributed and Parallels Treatments.

Abstract : Smart agriculture is one of the most diverse research. In addition, the quantity of data to be stored and the choice of the most efficient algorithms to process are significant elements in this field. The storage of collecting data from Internet of Things (IoT), existing on distributed, local databases and open data need a particular infrastructure to federate all these data to make complex treatments. The storage of this wide range of data that comes at high frequency and variable throughput is particularly difficult. In this paper, we propose the use of distributed databases and high-performance computing architecture in order to exploit multiple re-configurable computing and application specific processing such as CPUs, GPUs, TPUs and FPGAs efficiently. This exploitation allows an accurate training for an application to machine learning, deep learning and unsupervised modeling algorithms. The last ones are used

for training supervised algorithms on images when it labels a set of images and unsupervised algorithms on IoT data which are unlabeled with variable qualities. The processing of data is based on Hadoop 3.1 MapReduce to achieve parallel processing and use containerization technologies to distribute treatments on Multi GPU, MIC and FPGA. This architecture allows efficient treatments of data coming from several sources with a cloud high-performance heterogeneous architecture. The proposed 4 layers infrastructure can also implement FPGA and MIC which are now natively supported by recent version of Hadoop. Moreover, with the advent of new technologies like Intel® MovidiusTM ; it is now possible to deploy CNN at the Fog level in the IoT network and to make inference with the cloud and therefore limit significantly the network traffic that result in reducing the move of large amounts of data to the cloud.

Keywords : GPU, FPGA, MIC, CPU, TPU, Cloud, Big Data, parallel, distributed processing, heterogeneous cloud architecture

Web-based cattle behavior service for researchers based on the smartphone inertial central.

Abstract : Smartphones, particularly iPhones, can be relevant instruments for researchers in animal behavior because they are readily available on the planet, contain many sensors and require no hardware development. They are equipped with high performance inertial measurement units (IMU) and absolute positioning systems analyzing users' movements, but they can easily be diverted to analyze likewise the behaviors of domestic animals such as cattle. The study of animal behavior using smartphones requires the storage of many high frequency variables from a large number of individuals and their processing through various relevant variables combinations for modeling and decision-making. Transferring, storing, treating and sharing such an amount of data is a big challenge. In this paper, a lambda cloud architecture and a scientific sharing platform used to archive and process high-frequency data are proposed. An application to the study of cattle behavior on pasture on the basis of the data recorded with the IMU of iPhones 4S is exemplified. The package comes also with a web interface to encode the actual behavior observed on videos and to synchronize observations with the sensor signals. Finally, the use of fog computing on the iPhone reduced by 42% on average the size of the raw data by eliminating redundancies.

Keywords : precision livestock farming, smart breeding, smart agriculture, database, intertial unit, webservice, Internet of things, animal behavior, classification algorithms

Web Monitoring of Bee Health for Researchers and Beekeepers Based on the Internet of Things.

Abstract : The Colony Collapse Disorder (CCD) also entitled 'Colony Loss' has a significant impact on the biodiversity, on the pollination of crops and on the profitability. The Internet of Things associated with cloud computing offers possibilities to collect and treat a wide range of data to monitor and follow the health status of the colon. The surveillance of the animals' pollination by collecting data at large scale is an important issue in order to ensure their survival and pollination, which is mandatory for food production. Moreover, new network technologies like Low Power Wide Area (LPWAN) or 3GPP protocols and the appearance on the market easily programmable nodes allow to create, at low-cost, sensors and effectors for the Internet of Things. In this paper, we propose a technical solution easily replicable, based on accurate and affordable sensors and a cloud architecture to monitor and follow bees' behavior. This solution provides a platform for researchers to better understand and measure the impacts factors which lead to the mass extinction of bees. The suggested model is also a digital and useful tool for beekeepers to better follow up with their beehives. It helps regularly inspect their hives to check the health of the colony. The massive collection of data opens new research for a better understanding of factors that influence the life of bees.

Keywords : precision beekeeping, precision agriculture, lambda architecture,colony collapse disorder, Internet of Things, bee health

Fog-IoT-Cloud for Health : A new Architecture for Patients and Elderly Monitoring.

Abstract : The important increase of the elderly population and their desire to conduct an independent life, even when having medical diseases related to their age, requires the development of new technologies to ensure optimal living comfort for this population. In addition, another category of people, those who are patients with life-threatening problems, may benefit from preventive medical monitoring. In this paper, we present a Fog IoT Cloud-Based Health Monitoring System by using physiological and environmental signals allowing to provide contextual information in terms of Daily Living Activities. Our system enables healthcare providers to follow up health state and behavioral changes of elderly or alone people. Moreover, our system provides a monitoring rehabilitation and recovery processes of patients. Our Fog-IoT architecture consists of a wireless sensor network, a local gateway for data stored locally and quickly, and a Lambda cloud architecture for data processing and storage. The originality of our work resides in the graphical monitoring of new and recent patient data at local smart gateway level. This checkup gives the opportunity to the medical staff quick

access to the data, and allows them to validate automatically the observed anomalies. Finally, if a telematic break occurs, the gateway continues to accumulate the data while conducting their analysis. Anonymized data are sent periodically from Smart Gateways to the cloud for archiving and for checkup by medical staff who follow up with patients.

Keywords : Fog IoT, Cloud IoT, Context-Aware, Health at Home, e-Health, Patient Monitoring, Elderly People Monitoring

Open Phytotron : A New IoT Device for Home Gardening.

Abstract : Phytotron also called growth chambers are research installations where environmental parameters such as temperature, humidity, irrigation, conductivity, lighting, and CO_2 are finely controlled. This kind of installation allows on one hand to measure the impact of environmental changes, and on the other hand to optimize the natural grow of plants. With the democratization of the materials, cloud computing and new possibilities offered by Internet of Things (IoT). Therefore, it is possible to develop a low-cost personal phytotron. In this paper, we propose to use connected things to develop a personal growth chamber with the aim to produce fresh vegetable in an urban context.

Keywords : phytotron, growth chamber, smart home, smart agriculture, cloud computing, Internet of Things, Home Assistant, openHAB, Node-Red

Edge Computing for Cattle Behavior Analysis.

Abstract : Smartphones, particularly iPhone, can be relevant instruments for researchers because they are widely used around the world in multiple domains of applications such as animal behavior. iPhone are readily available on the market, contain many sensors and require no hardware development. They are equipped with high performance inertial measurement units (IMU) and absolute positioning systems analyzing user's movements, but they can easily be diverted to analyze likewise the behaviors of domestic animals such as cattle. Using smartphones to study animal behavior requires the improvement of the autonomy to allow the acquisition of many variables at a high frequency over long periods of time on a large number of individuals for their further processing through various models and decision-making tools. Indeed, storing, treating data at the iPhone level with an optimal consumption of energy to maximize battery life was achieved by using edge computing on the iPhone. This processing reduced the size of the raw data by 42% on average by eliminating redundancies. The decrease in sampling frequency, the selection of the most important variables and postponing calculations to the cloud allowed also an increase in battery life by reducing of

amount of data to transmit. In all these use cases, the lambda architectures were used to ingest streaming time series data from the Internet of Things. Cattle, farm animals' behavior consumes relevant data from Inertial Measurement Unit (IMU) transmitted or locally stored on the device. Data are discharged offline and then ingested by batch processing of the Lambda Architecture.

Keywords : cattle behavior analysis, edge computing, farm' animal, iPhone, Flutter

A new Edge Architecture for AI-IoT services deployment.

Abstract : Artificial intelligence (AI) and Internet of things (IoT) have progressively emerged in all domains of our daily lives. Nowadays, both domains are combined in what is called artificial intelligence of thing (AIoT). With the increase of the amount of data produced by the myriad of connected things and the large wide of data needed to train Artificial Intelligence models, data processing and storage became a real challenge.

Indeed, the amount of data to process, to transfer by network and to treat in the cloud have call into question classical data storage and processing architectures. Also, the large amount of data generated at the edge has increased the speed of data transportation that is becoming the bottleneck for the cloud-based computing paradigms.

The post-cloud approaches using Edge computing allow to improve latency and jitter. These infrastructures manage mechanisms of containerization and orchestration in order to provide automatic and fast deployment and migration of services such as reasoning mechanisms, ontology, and specifically adapted artificial intelligence algorithms.

In this paper we propose a new architecture used to deploy at edge level micro services and adapted artificial intelligence algorithms and models.

Keywords : Edge computing, Edge Internet of Things, Edge Artificial Intelligence, Edge A2IoT architecture, Internet of Things, Artificial Intelligence, Edge AI-IoT architecture

Edge Computing and Artificial Intelligence for Real-time Poultry Monitoring.

Abstract : Smart Poultry acquires data from aviaries by means of sensor network at reduced intervals of time (every minute) that generate hundred thousands of data. The conjunction of Internet of Things and Artificial Intelligence open the field of the

real-time monitoring of poultry and ,advance analytics and automation if data is from high quality. In this paper, we propose a scalable monitoring of a poultry achieved with open hardware wireless sensors network and software. We use a Gated Recurrent Unit, an artificial intelligence algorithm to validate and predicate environmental parameters.

Keywords : Edge AIoT, Edge Computing, Edge Artificial Intelligence, Internet of Things, Artificial Intelligence, Poultry, Smart Poultry, Gated Recurrent Unit, GRU

Edge Computing and Artificial Intelligence Semantically Driven. Application to a Climatic Enclosure.

Abstracts : Climatic chamber are enclosures where the ambient conditions, i.e. the temperature and humidity, are finely controlled. This one can play multiple roles such as the cultivation of plants (phytotron), the breeding of insects or habitat for exotic animals. The availability on the market of a wide variety of equipment makes it difficult to share settings and operating recipes. In this paper, we propose a versatile and automated modular climatic enclosure system that can be adapted according to the use cases and available material. In this paper, we use IoT device virtual representation, data validation by means AI algorithm, the semantic description of material, and ontology locally deployed from the cloud allowing to automate the local installation with container technology.

Keywords : phytotron, climatic enclosure, growth chamber, pogona, semantic rules, Internet of Things

Smart Nest Box : IoT Based Nest Monitoring In Artificial Cavities.

Abstract : With climate change, habitat loss, and impoverishment of food sources, several species of bird are are threatened today. It is crucial to conserve the biodiversity in ecosystems but the conservation that requires an improved knowledge of these. In this paper, we propose a low-cost connected nest box that make photos of nestling and measures them weight with a load charge. Air temperature and humidity are also regularly controlled to follow environmental conditions and their impact on the nestling.

Keywords : Bird Nesting, ecosystems conversation, ESP32, conservation of species, nesting monitoring, nest box

RevoCampus : a distributed open source and low-cost smart campus.

Abstract : Smart Campus can be assimilated to small smart cities in which learning experience and living conditions are improved by smart environment and IoT concepts. In this paper, we present (R)evoCampus our Smart Campus solution based diverted smart Home interoperable protocol platforms, micro controllers ESP32, low-cost sensors. This architecture uses at same time the principles of IoT, smart environment technologies, and smart city concepts to develop an effective use of the resources, and to improve the quality of life inside the whole University. In the proposed solution, Wi-Fi protocol is used for communication in indoor while outdoor communications are ensured by LoRaWAN protocol.

Keywords : Smart Campus, Smart City, Smart Environment, Internet of Things.

Edge Computing and Artificial Intelligence for Landslides Monitoring.

Abstract : Landslides are phenomena widely present around the world and responsible each year of numerous life loss and extensive property damage. Researchers have developed various methodologies to identify area of high susceptibility of landslides. However, these methodologies cannot predict 'when' landslides are going to take place. Indeed, Wireless Sensors Network (WSN), Internet of Things (IoT) and Artificial Intelligence (AI) offer the possibility to monitor in real-time parameters causing the triggering factors of rapid landslides. In this paper, we suggest a real-time monitoring of landslides in order to precociously alert population in dangerous situation by means of a warning system. The novelty of this paper is the coupling of wireless sensors network and a multi-agent system deployed on an edge AI-IoT architecture by means of Kubernetes and Docker.

Keywords : Landslides susceptibility, Internet of Things, Artificial Intelligence, early warning system, landslides monitoring.

A new Collaborative Platform for Research in Smart Farming.

Abstract : The sharing of experimental dataset and benchmark between researchers is particularly import for the cross validation of models and algorithms that have been developed. In practice the sharing of data is difficult because certain legislations must be respected such as protecting privacy, copyrights, information confidentiality, etc.

In this paper, we propose a scientific collaborative platform to share, exchange, and transfer data, applications, and models between researchers. This is only possible if we implement a high-level of encryption and security. The choice of encryption algorithms is crucial to ensure a long-term high level of protection against theft, willful alteration, and falsification. Our platform has been experimented within a community of researchers interested in cow behavior analysis based on Inertial Movement Unit and GPS data acquired by iPhone at high frequency (100 Hz).

Keywords : Animal Behavior, Behavior Analysis, Sharing Platform.

Edge AI-IoT Pivot Irrigation, Plant Diseases and Pests Identification.

Abstract : Overcoming population growth dilemma with less resources of soil and water, the irrigated agriculture allows us to increase the yield and the production of several crops in order to meet the high requirements of demands of food and fibers. Efficiently, an irrigation system should correctly evaluate the amount of water and also the timing, when applying certain irrigation doses. Global warming of the planet, to which is added in some regions an irregular regime of precipitation and a scarcity of available water resources, requires precision irrigation systems. The rational use of water and inputs (mainly fertilizers and pesticides) is crucial in some areas of the planet suffering from a deficiency of water. Hence, in these regions where the environmental conditions are harsh to ensure an efficient crop growth. Moreover, plant diseases and pests impact the yields of crops. For these reasons is it why an early detection gives us the opportunity to treat the disease or pest as quickly and effectively as possible, in order, to reduce the impact of these latter. Nowadays, the identification of plant diseases and pest with Artificial Intelligence algorithms on video flow in real conditions with variable exposition are still being a very challenging problem. Researchers classically develop algorithms that are trained on calibrated exposition images, which does not perform well in real conditions. Furthermore, the processing of a video in real time needs specialized computing resources close to the pivot-center irrigation trained with AI algorithms on real images and then analyzes rapidly, detects problem, and then react accordingly. In this paper, we complete our previous proposed IoT system to optimize the water use and we displaced the computing of data at the edge level in order to be able to process videos locally, event the Internet connection is limited. This local computing power also allows us to manage the supply of fertilizers and the treatment of plant diseases, and pests.

Keywords : Center-pivot Irrigation, Connected Irrigation, Smart Irrigation, Water Requirement, Intelligent Irrigation

Toward a Big Data Knowledge-Base Management System for Precision Livestock Farming.

Abstract : Nowadays, we are in the era of advanced technologies where tremendous amount of data is produced by multiple sources such as sensors, devices, social media, user experiences, etc. Furthermore, this raw data has a low value, and major part is not really useful or important for business. One way to give an added value to this stored data is to extract useful knowledge from it, for the ending-system or the end-users by a process commonly called knowledge Discovery in Database (KDD). Smart Farming uses a large amount of connected technologies producing also a huge amount of data in order to maximize productions by reducing : human efforts, environment impact and wasting natural resources. In this paper, We develop a new data analytic architecture dedicated to Precision Livestock Farming (PLF) to improve in particular the livestock animals production, animals' welfare, and farming processes. We present a new data processing architecture for a knowledge-base management system (KBMS) allowing to ease decision support and monitoring operations that can help farmers and stakeholders to better exploit data and have a long-term view of the evolution of the knowledge it contains. Our main contribution in the present paper is a new architecture specifically developed for the precision livestock farming integrating a periodical data reevaluation which address the problematic of data conservation and the decrease in data value over time.

Keywords : Big Data, Databases, Knowledge discovery in databases, knowledge-base management system, Precision livestock farming

A new Kappa Architecture for IoT Data Management in Smart Farming.

Abstract : Agriculture 4.0 is a domain of IoT in full growth which produces large amounts of data from machines, robots and sensors networks. This data must be processed very quickly, especially for the systems that need to make real-time decisions. The Kappa architecture provides a way to process Agriculture 4.0 data at high speed in the cloud, and thus meets processing requirements. This paper presents an optimized version of the Kappa architecture allowing fast and efficient data management in Agriculture. The goal of this optimized version of the classical Kappa architecture is to improve memory management and processing speed. the Kappa architecture parameters are fine tuned in order to process data from a concrete use cases. The results of this work have shown the impact of parameters tweaking on the speed of treatment. We have also proven that the combination of Apache Samza with Apache Druid offers the

better performances.

Keywords : agriculture 4.0, IoT, Internet of Things, kappa architecture, smart farming.

Internet of Things : a new Interoperable IoT Platform. Application to a Smart Building.

Abstract : According to statistics provided by Cisco, the Internet of Things market will grow rapidly and will reach 50 billion devices connected to the Internet by 2020. All these connected devices, in smart Home/Building should make life easier, more sweet and enjoyable. Ideally, each device must work immediately and not require people to try to make them work. However, connected things come from different manufacturers using different protocols of communication. The diversity of protocols and sometimes their incompatibilities stay a major issue. Although, efforts have been made in recent years to improve the compatibility of the protocols but these interactions remain limited. One solution to address this issue is to use an Internet of Things (IoT) platform to facilitate their interaction, control them intelligently using a web interface or a voice assistant. In this paper, we define connectivity requirements to improve interoperability between devices that make up the Internet of Things (IoT). An implementation of a Web application and virtual assistance that ensures interoperability regardless of form factor, operating system, service provider or transport technology, creating a "network of everything".

Keywords : phytotron, growth chamber, smart home, smart agriculture, cloud computing, Internet of Things, Home Assistant, openHAB, Node-Red.

Annexe 3 : Application mobile multi OS d'acquisition des données comportementales

Publications liées à ce chapitre

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, J. Bindelle, F. Lebeau, "Edge Computing for Cattle Behavior Analysis". In Second international conference on Embedded & Distributed Systems (EDiS'2020), Oran, Algeria, 2020. doi : 10.1109/EDiS49545.2020.9296471.

Pour faciliter l'acquisition des données, nous avons développé une application mobile en Flutter [3] codée en Dart [4] pour récupérer les données provenant de la centrale inertielle des smartphones. Ce qui a motivé le développement de cette application est l'abandon du développement du logiciel original qui permettait le log des données de l'Inertial Measurement Unit (IMU) des iPhone 4s et 5s à une fréquence allant jusqu'à 100 Hz. D'autre part, cette application n'existant que sur iOS et donc il n'était pas possible d'aborder le marcher le plus important qui est celui des smartphones sous Android. Nous avons profité de l'occasion pour ajouter deux aspects qui manquait à l'application précédentes à savoir la collecte d'informations provenant d'autres capteurs que ceux de la centrale inertielle et du GPS et l'adjonction de méta-données qui permettent de documenter de décrire les capteurs qui ont été utilisés pour l'acquisition des données.

Cahier des charges de l'application

Le but de l'application est de collecter les données provenant des capteurs d'un nœud placé sur le cou d'un animal de ferme (par ex. : une vache). L'acquisition des signaux doit pouvoir être réalisé à haute fréquence cela implique le développement d'une application native bas niveau qui puisse accéder aux capteurs directement. Il faut également que cette application soit indépendante de la plateforme qui l'héberge et soit aisément évolutive et maintenable afin d'assurer sa durabilité.

3. https ://flutter.dev
4. https ://dart.dev

Les fonctionnalités à mettre en place sont : (**1**) Sélectionner les capteurs dont les données doivent être acquises ; (**2**) Choisir la fréquence (Hz) d'acquisition des données ; (**3**) Collecter les informations relatives au capteurs (méta-données descriptives) et les stocker au format RDF pour documenter les caractéristiques des capteurs sélectionnés ; (**4**) Créer un fichier sémantique au format RDF qui accompagne chacune des acquisitions de données et qui reprendra les capteurs sélectionnés, la fréquence d'échantillonnage, le nombre de données collectées pour chacun des capteur pour la durée de fonctionnement. (**5**) Chaque acquisition doit faire l'objet d'un fichier csv séparé dont le nom contient le nom du périphérique, la date et l'heure du démarrage de l'acquisition de données.

Choix technologiques

Ces dernières années ont vu l'essor de nombreuses technologies multi-plateforme dont l'objectif de réaliser une application pour iOS, Android, le Web à partir d'un code source unique. Une technologie très en vogue actuellement est React Native qui utilise un bridge Javascript. C'est une technologie mature provenant de React. Flutter est une technologie jeune poussée par Google, qui évolue très rapidement. Il s'appuie utilise un Skia écrit en C++. Les applications React Native se programment en Javascript tandis que celle Flutter sont écrites en DART, un langage plus propre et plus robuste que le Javascript. De plus, Flutter permettra à terme de créer une application Windows 10 sans réécrire le code. React Native nécessite quant à lui une réécriture partielle du code pour éditer l'application Windows 10 correspondante. Note choix s'est porté sur Flutter car il permet de développer des applications bas niveau déployables en multi plateforme (Android & iOS). C'est néanmoins encore une technologie jeune qui manque de maturité mais, elle offre un large éventail de modules permettant d'ajouter rapidement de nouvelles fonctionnalités.

Implémentation

L'application comporte 3 vues : La première permet de sélectionner les capteurs. La seconde de consulter les informations d'un capteur. La troisième permet de paramétrer l'application c'est-à-dire choisir la fréquence d'échantillonnage, choisir l'emplacement où les fichiers sont sauvegardés.

La Figure 1 montre l'écran de sélection des capteurs. La liste des capteurs est constituée dynamiquement en fonction de ceux disponibles sur le smartphone. Chaque nom de capteur est précédé par un icône (i) qui permet de consulter les caractéristiques du capteurs (Voir Figure 2). Un sélecteur (toogle) situé à droite de la dénomination de chaque capteur permet de l'activer ou de le désactiver. Le bouton "Start" permet démarrer l'acquisition de données.

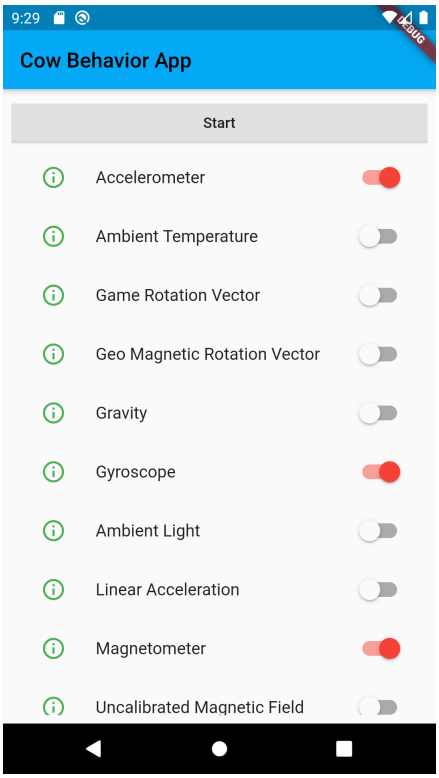


FIGURE 1 – Sélection des capteurs

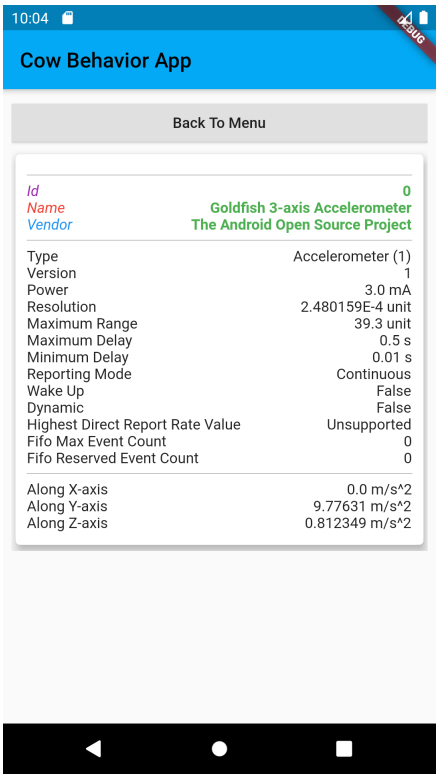


FIGURE 2 – Méta-données des capteurs

Annexe 4 : Plateforme sécurisée de partage de données de recherche

Publications liées à ce chapitre

O. Debauche, S. Mahmoudi, P. Manneback, "A new collaborative platform for Cow Behavior benchmark datasets". In : The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020). Procedia Computer Science, 2020, **177** : 450-455. doi : 10.1016/j.procs.2020.10.061.

O. Debauche, S. Mahmoudi, S.A. Mahmoudi, P. Manneback, "A new collaborative platform for Covid-19 benchmark datasets". In L. Garg , C. Chakraborty, S. Mahmoudi, V. Sohmen (eds), "Intelligent Healthcare Informatics for Fighting the COVID-19 and Other Pandemics and Epidemics, EAI/Springer Innovations in Communication and Computing, Springer, 2022. doi : 10.1007/978-3-030-72752-9_12.

Introduction

Les données de recherche en Smart Farming acquises à l'aide des objets connectés et traitées en local au niveau du Edge Computing et s'appuyant sur le Cloud peuvent être partagées avec d'autres équipes de recherche. Mais cela n'est possible que si la confiance entre chercheurs existe. Un moyen pour y parvenir est de donner un maximum de garanties en matière de sécurité, de respect de la confidentialité des données et des secrets de fabrication.

Le partage des données de recherches pour notre cas d'utilisation d'analyse du comportement des animaux de ferme avec d'autres équipes de recherche permet notamment la validation croisées des modèles et l'enrichissement rapide des bases de données de modélisation avec des données particulières terme de rareté. Néanmoins ce partage et/ou échange de données est toujours délicat, notamment à cause des droits d'auteurs, du caractère confidentiel de certaines données, des secrets de fabrications, droits sui generis pour les données des bases de données, ainsi que les droits voisins et connexes. A

ces droits viennent s'ajouter la protection des secrets de fabrication et industriels ainsi qu'éventuellement les prescriptions du Règlement général sur la protection des données (RGPD) si des données relatives aux personnes sont consignées, sur le territoire européen. Dans cette optique nous avons créé notre plateforme de partage de données qui est déployée dans la partie hébergement d'applications notre architecture LAMA.

Mise en place de la sécurité

La sécurité est implémentée à différents niveaux de la plateforme pour se prémunir des risques liés aux défaillances matérielles et aux différentes possibilités d'attaques (piratage informatique). La redondance des données et des sauvegarde permet de prémunir des défaillances matérielles ou de restaurer les données les données en cas d'altération involontaires par exemple. Le choix des technologies pour la sécurisation de la plateforme est primordial pour garantir la confidentialité, se prémunir des vols, des altérations volontaires ou non, falsification de données et des attaques dirigées mettant en péril l'intégrité de la plateforme.

Algorithmes de hachage

Différents familles d'algorithmes de hachage existent. Les fonctions de hachage Secure Hash Algorithm (SHA) : SHA-0, SHA-1 et SHA-2 ont été développées par la National Security Agency (NSA) et sont décrite dans la publication FIPS-180-4 [176]. La version SHA-3 [177] a quant à elle fait l'objet d'un concours et est décrite dans la publication FIPS-202 [177].

L'algorithme de hachage Whirlpool [178] a été développé dans le cadre du projet européen New European Schemes for Signatures Integrity and Encryption (Nessie) dont le but était de proposer des algorithmes européens pour la protection des données industrielles. Whirlpool est un algorithme de hachage sur 512 bits utilisant le bloc de chiffrage de l'algorithme Advanced Encryption Standard (AES), ce qui garantit sa robustesse et sa fiabilité.

Il est toutefois nécessaire de trouver un compromis entre solidité du hachage et un temps de calcul qui reste raisonnable. Nous avons évalué les performances de différents algorithmes de hachage sur des tailles croissantes de fichiers. Les algorithmes crc32, crc32b, md5 [179] et SHA-1 (160 bits) [178] ne sont plus jugés fiables mais sont néanmoins testés à titre de base de comparaison. Actuellement, les versions 2 et 3 du l'algorithme SHA sont toujours réputées fiables. Néanmoins, la version 2 laisse craindre des vulnérabilités étant donné qu'elle est construite suivant le même schéma que md5 [179] et SHA-1. La version 3 est quant à elle construite sur un principe tout à fait différent. Nous avons testé différentes versions des algorithmes SHA-2 [178] : SHA-224 (224 bits), SHA-256 (256 bits), SHA-384 (384 bits), SHA-512 (512 bits) et les versions

tronquées SHA-512/256 (256 bits) et SHA-512/224 (224 bits) et SHA-3 [178] : SHA3-224 (224 bits), SHA3-256 (256 bits), SHA3-384 (384 bits) et SHA3-512 (512 bits) pour en mesurer les performances sur le hachage de fichiers de différentes tailles. La Figure 3 montre l'évolution du temps de hachage en fonction de la taille du fichier pour différents algorithmes appliqués sur des fichiers log au format csv de tailles croissantes.

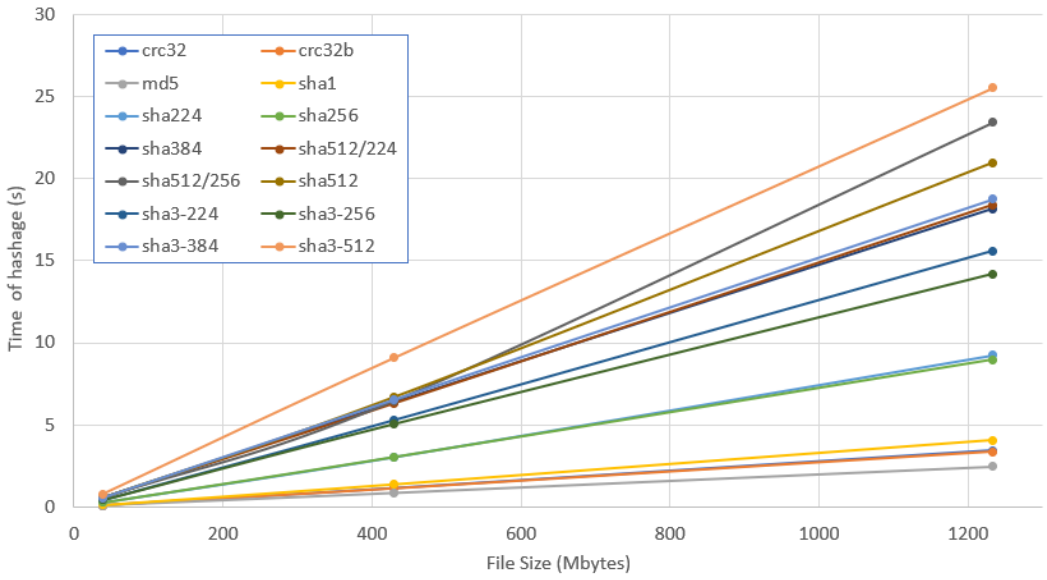


FIGURE 3 – Comparaison des vitesses de hachage de différents algorithmes

Il n'est toutefois pas recommandé d'utiliser les algorithmes de hachage SHA pour le cryptage des mots de passe car ceux-ci sont des chaînes de caractères de longueur courte et générées par des humains, ce qui les rend vulnérables aux attaques de type dictionnaire. La tendance actuelle est d'utiliser des algorithmes spécifiques pour les mots de passe tel qu'Argon 2id [180] qui est une hybride entre Argon2d [180] qui est conçu spécifiquement pour résister au crackage par GPU et Argon2i [180] qui est quant à lui optimisée contre les attaques par canal auxiliaire.

Algorithmes de cryptage

Le cryptage des données est nécessaire pour garantir leur protection contre le vol, les indiscrétions et les altérations volontaire ou non. Différents algorithmes de cryptages existent mais l'un des plus utilisés et réputé actuellement est l'AES.

L'algorithme de chiffrement symétrique AES est basé sur des blocs de 128 bits et est décrit dans la publication FIPS-197. Il est réputé à l'heure actuelle comme l'un des plus sure. Il existe trois variantes d'algorithme AES-128 (clé de 128 bits), AES-192 (clé de

192 bits) et AES-256 (clé de 256 bits). La NSA a par ailleurs recommandé son utilisation pour la protection des données sensibles (top secret) avec la version AES-256 [5]. Pour le cryptage des fichiers, une version d'AES-256 associée avec le mode d'opération de chiffrement par bloc Galois/Counter Mode (GCM) qui est largement répandu, efficace et performant. L'AES-256-GCM est conçu pour garantir l'intégrité, l'authenticité et la confidentialité des données. Il est limité à environ 350 GB par couple (clé, nonce [6]).

Le chiffrement de flux ChaCha20 et l'authentification Poly1305 sont des algorithmes dont le but d'assurer des hauts niveaux de sécurité, tout en atteignant des performances élevées sur une large gamme de plates-formes logicielles. ChaCha20-Poly1305 AEAD est un cryptage authentifié avec un algorithme de cryptage de données supplémentaires utilisant une clé de 256 bits et un vecteur d'initialisation (nonce) de 96 bits et un message à crypter de longueur arbitraire. XChaCha20-Poly1305 [181] est une version améliorée de ChaCha20-Poly1305 utilisant une clé de 256 bits, un nonce de 192 bits et des blocs de 512 bits. Par ailleurs, Chacha20-Poly1305 et Xchacha20-Poly1305 permettent de crypter des fichiers jusque 256 GB.

Algorithme de cryptage des clés

Les clés des algorithmes de cryptage évoqués au paragraphe précédent doivent être au préalable cryptées avant d'être stockées. Le X25519XSalsa20Poly1305 est l'association de trois algorithmes de cryptages : Le X25519 est le Diffie-Hillman utilisant le Curve25519, dispose de 128 bits de sécurité et permet de créer du «secret partagé», entre les deux utilisateurs impliqués. Le XSalsa20 est l'algorithme de chiffrement symétrique, c'est-à-dire qui utilise la clé secrète, qui a été échangée grâce au X25519, et le nonce qui a été généré. Le Poly1305 est le Message Authentication Code (MAC), il garantit que le message n'est pas modifié par un intervenant externe.

Mise en œuvre

Nous avons mis en œuvre une plateforme codée en PHP 8.0.5 compilé avec les librairies sodium 1.0.17 et OpenSSL 1.0.1t. Elle est hébergée sur un hébergement mutualisé performance 4 d'OVH (8 vCores et 8 Gb RAM) répliqués sur trois sites. Cet hébergement offre des performances garanties (VPS), une sauvegarde journalière des fichiers et à J+1, J+2, J+3 et J+7 ainsi qu'une sauvegarde journalière des bases de données des trente derniers jours. La base de données est hébergée sur un serveur MySQL 5.6.50. Un Content Delivery Network (CDN) réplique les fichiers hébergés sur le cluster central sur 19 points répartis dans le monde. De plus, le réseau de DNS répartis

5. `https://web.archive.org/web/20150815072948/https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`

6. Nonce signifie Number used ONCE. Le nonce est un nombre aléatoire ou pseudo-aléatoire destiné à être utilisé une seule fois.

(DNS Anycast) permet d'améliorer la latence lors des requêtes DNS en redirigeant les requêtes vers le nœud le plus proche parmi les 19 disponibles. En cas d'indisponibilité de l'un des points de répartition, le nœud le plus proche acceptera les requêtes.

Authentification

L'authentification des utilisateurs se déroule en 12 étapes (Figure 4). L'utilisateur se connecte à la plateforme à l'aide d'un navigateur web (**1**). Le serveur web prépare la page web de réponse en interrogeant Google Recaptcha V3 pour y intégrer la partie client sous la forme d'un code javascript (**2**). L'ensemble des noms des champs du formulaire d'authentification sont remplacés par des hachage en sha3-224. La correspondance entre les noms réels des champs et les clés de hachage est sauvegardée dans une session sur le serveur (**3**). Une clé d'authentification du formulaire hachage en sha3-384 à usage unique est créé avec une validité de 2 minutes. Elle est ajoutée à la page web et également sauvegardée en base de données. Elle permet de garantir que les données envoyées proviennent bien du formulaire de la plateforme (**4**). La page d'authentification est ensuite retournée par le serveur à l'utilisateur ainsi que le cookie de session (**5**).

L'utilisateur procède ensuite à l'encodage de ses données d'identification : adresse mail et mot de passe. La page est envoyée au serveur contient la clé d'authentification sha3-384 à usage unique du formulaire (**6**). La valeur du captcha est en envoyée à Google Recaptcha qui retourne une score entre 0 et 1. Les scores supérieurs ou égaux à 0.5 sont jugés fiables (**7**). La correspondance Les données transmises par la méthode POST du formulaire sont récupérées et filtrées pour éliminer toute tentative d'attaque par injection (**8**). La correspondance entre l'adresse mail de l'utilisateur et son mot de passe haché avec l'algorithme Argon 2id sont vérifiés avec les données en base de données (**9**). La clé de hachage de validation du formulaire en sha3-384 est vérifiée avec celle qui est stockée en base de données ainsi que sa durée de validité. (**10**). Si la captcha, le couple adresse mail/mot de passe et la clé de validation du formulaire sont vérifiés un session utilisateurs est créée et la clé de validation est supprimée (**11**). Un nouvelle page web est générée et renvoyée par le serveur web à l'utilisateur. Cette page intègre les contenus accessibles à l'utilisateur en fonction de son niveau de pouvoir (**12**).

Accès restreint à la plateforme

L'utilisateur authentifié peut accéder à des contenus réservés à la session utilisateur a été créée sur le serveur et au cookie de session qui a été transmis à l'utilisateur. La Figure 5 montre le fonctionnement de l'accès contrôlé de la plateforme. Il peut alors effectuer une requête (**1**). Le serveur web vérifie que l'utilisateur est légitime c'est-à-dire identifié pour accéder au contenu restreint et possède le niveau de pouvoir nécessaire

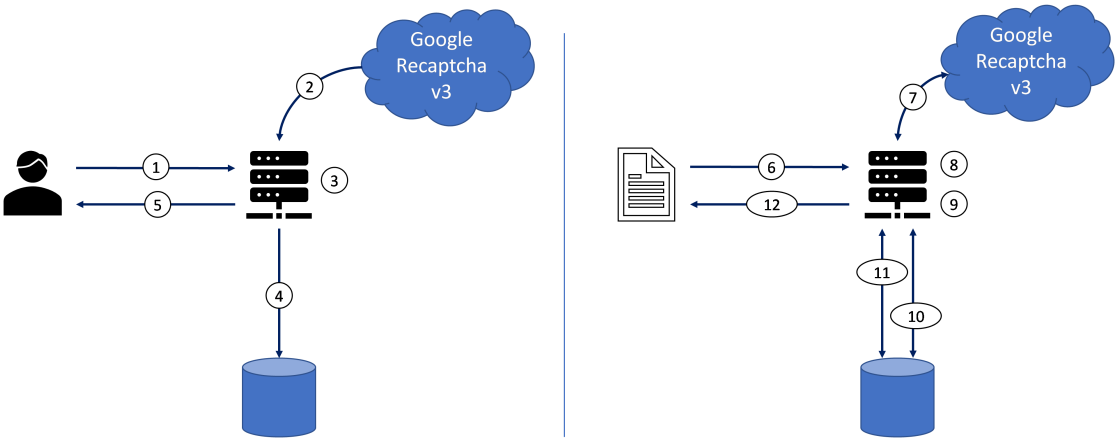


FIGURE 4 – Processus d'authentification

(**2**). Si c'est le cas, il récupère le code javascript de Recaptcha (**3**) et crée un clé d'authentification dans la base de données (**4**) et l'ajoute au formulaire de la page sous forme caché. Comme pour l'authentification l'ensemble des champs sont remplacés par des hachages sha3-224. Si l'utilisateur n'est pas légitime il est renvoyé vers la page d'identification avec un code d'erreur indiquant la raison pour laquelle il a été rejeté.

Si la page contient un formulaire, les données encodées par l'utilisateur sont transmises au serveurs (**6**). Le recatpcha est vérifié au niveau du serveur. Si le score est supérieur ou égal à 0.5 l'utilisateur est réputé fiable. Dans le cas contraire, l'utilisateur est redirigé vers une page spécifique (**7**). L'existence d'une session utilisateur valide est ensuite vérifiée (**8**) et la clé de validation du formulaire est comparée avec celle sauvegardée en base de données (**9**). La correspondance entre les hachages des champs et les vrais noms de champs est effectuées et les données transmises sont récupérées et nettoyées (**10**). Les données complémentaires sont le cas échéant également récupérées au niveau de la base de données (flèche en pointillés rouges) et la page est retournée à l'utilisateur.

Si la page de contient pas de formulaire, un lien vers un autre page est activé (**6**). Le captcha est vérifié (**7**), la présence d'une session utilisateur valide (**8**) et de la possession d'une clé de validation correctes (**10**) sont effectués. Le contenu additionnel éventuel est récupéré de la base de données et la page est retournée à l'utilisateur (**11**).

Dépôt des données

Chaque ensemble de données (*Dataset*) est décrit par son nom, une description, la source (propriétaire légal des données), le type de licence et le statut des données sur la plateforme (publiques, privées, partagées). Dans le mode privé, l'accès est donné individuellement. Les autorisations sont liées à un profil utilisateur et un mot de passe tan-

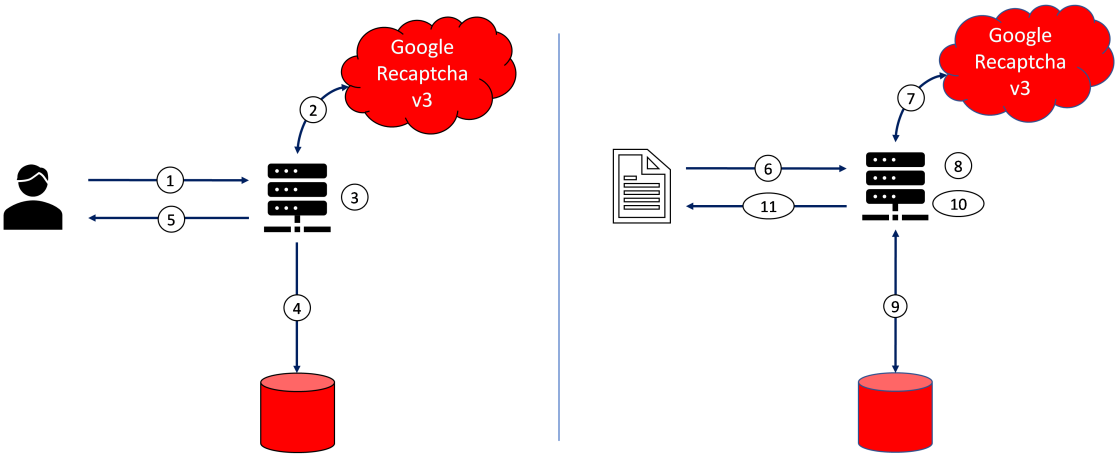


FIGURE 5 – Consultation des pages à accès restreint

dis que dans l'accès en mode partagé, l'accès est donné à une ip fixe et un mot de passe commun à tous les utilisateurs autorisés. Le mode public ne nécessite aucune identification. Une compte sftp spécifique au dataset est créé. Le propriétaire des données peut alors téléverser l'ensemble de ses fichiers. L'étape suivante consiste à identifier pour chaque fichier sa nature (dataset, annotation, licence, image, note), et à préciser son niveau d'accès privé ou public. En effet, un dataset peut être privé dans son ensemble et admettre comme exception un échantillon ou quelques images qui donne un aperçu sur le contenu avec un accès public ou partagé. Les données saisies par l'utilisateur sont sauvegardées en base de données et les fichiers autres que les licences et les images publiques sont cryptés. Au terme du dépôt, une vignette (card) est générée.

Cryptage / Décryptage des données sensibles

Chaque fichier déposé, contenant des données, est crypté à l'aide d'AES-256-GCM. Le fichier crypté est ensuite crypté à l'aide de XChachaPoly1305. Les clés AES-256 et XChachaPoly1305 ainsi que le nonce sont ajoutés au dépôt de clés du propriétaire des données. Le dépôt de clés de chaque utilisateur est crypté à l'aide de l'algorithme X25519XSalsa20Poly1005.

Chaque fichier crypté est ensuite décrypté et le hachage du fichier décrypté est comparé à celui du fichier original pour s'assurer que l'intégrité des données est conservée. Si les hachages sont identiques le fichier décrypté et le fichier original sont supprimés de l'espace de stockage.

Log de l'activité de la plateforme

L'ensemble des transactions effectuées sur la plateforme sont consignés sous forme de logs. Ces logs permettent de détecter les défaillances ou de conserver les traces des tentatives de piratages au sein de la plateforme. Les logs au niveau du serveur web apache du prestataire de service fournissant l'hébergement sont également conservés et permettent un analyse de la fréquentation et des requêtes envoyées à la plateforme sous forme d'url. Le croisement des actions menées sur la plateforme via les logs internes et les urls externes utilisées pour effectuer les requêtes (logs externes) permettent d'avoir une vue détaillée du fonctionnement de la plateforme et du comportement des utilisateurs.

Vérification d'intégrité de la plateforme

Toutes les heures, l'intégrité de la plateforme est vérifiée en recalculant les hash sha3-224 de chacun des fichiers de l'ensemble du code. Ces hachages sont ensuite comparés à ceux stockés en base de données. L'ajout de nouveaux fichiers peut être aisément détectée car non présent dans la base de données. La suppression de code est détectée lorsque le hachage d'un fichier ne correspond pas à celui de la base de données et que sa taille est inférieure à celui stocké en base de données. Il en est de même avec l'ajout de code mais dans ce cas la taille du fichier est supérieure à celle stockée en base de données. En cas de détection de modification la liste des fichiers est envoyée par mail à l'administrateur et la plateforme est mise en mode maintenance.

Annexe 5 : Description des composants logiciels

Apache Accumulo est système de gestion de base de données NoSQL de type clé-valeur de la fondation Apache créé par la NSA. Il utilise le système de fichier Hadoop Distributed File System (HDFS) et est spécialisé dans la gestion de données de masse.

Licence : Apache 2.0 - Site : https://accumulo.apache.org/.

Apache Apex est une plateforme qui unifie le traitement des flux et des batch. Il traite les données big data-in-motion d'une manière qui est évolutive, performante, tolérante aux pannes, dynamique, sécurisée, distribuée, et facilement accessible.

Licence : Apache 2.0 - Site : https://apex.apache.org/.

Apache Beam est un modèle de programmation unifié open source pour définir et exécuter des flux de données, y compris les opérations d'extraction, transformation, centralisation (ETL), le traitement par lot et en flux.

Licence : Apache 2.0 - Site : https://beam.apache.org/.

Apache Cassandra est une base de données distribuée NoSQL opensource évolutive et à haute disponibilité sans compromettre les performances.

Licence : Apache 2.0 - Site : https://cassandra.apache.org.

Docker est une plateforme de conteneurs logiciels conçue pour le développement et le déploiement rapide d'applications et de microservices.

Licence : Apache 2.0 - Site : https://www.docker.com/.

Apache Druid est un base de données analytique en temps réel à haute performance fusionnant les caractéristiques clés des entrepôts de données, des bases de données chronologiques et des systèmes de recherche.

Licence : Apache 2.0 - Site : https://druid.apache.org/.

Apache Hadoop un framework open source écrit en Java destiné à faciliter la création d'applications distribuées et évolutives. Ce framework est conçu pour permettre aux applications qui l'utilisent de travailler avec des milliers de nœuds et des pétaoctets de données.

Licence : Apache 2.0 - Site : https://hadoop.apache.org/.

Apache HBase est une base de données distribuée open source conçue pour le stockage de grande base de données contenant des tables de plusieurs milliards de lignes et millions de colonnes.

Licence : Apache 2.0 - Site : https://hadoop.apache.org/.

Apache Jena est un framework Java open source dédié à la création d'applications de Web sémantique et à données liées.

Licence : Apache 2.0 - Site : https://jena.apache.org/.

Jenkins est serveur d'automatisation open source qui permettent de construire, de déployer et d'automatiser les tests de projets de développement logiciel.

Licence : MIT - Site : https://www.jenkins.io/.

Apache Kafka est un système unifié d'agents de messages, open source, en temps réel et à latence faible pour la manipulation de flux de données ; inspirés au niveau conceptuel des journaux de transactions.

Licence : Apache 2.0 - Site : https://kafka.apache.org/

Kubernetes (k8s) est une plateforme permettant d'automatiser le déploiement la mise en œuvre de conteneurs d'application sur des clusters de serveurs et d'adapter le cluster avec la charge.

Licence : Apache 2.0 - Site : https://kubernetes.io.

Apache Marathon est une plateforme d'orchestration de conteneurs de niveau production pour le système d'exploitation de centre de données de Mesosphere (DC/OS) et Apache Mesos.

Licence : Apache 2.0 - Site : https://mesosphere.github.io/marathon/.

MariaDB est un système de gestion de base de données relationnelles open source parmi les plus populaires issus du fork de MySQL.

Licence : GPL - Site : https://mariadb.org/.

Apache Maven est un outil de gestion et d'automatisation de production des projets logiciels Java.

Licence : Apache 2.0 - Site : https://maven.apache.org/.

Apache Mesos est un gestionnaire de cluster opensource, développé par l'université de Berkeley.

Licence : Apache 2.0 - Site : https://mesos.apache.org/.

MongoDB est un système de gestion de base de données orienté documents, distribuable, ne nécessitant pas de schéma prédéfini des données.

Licence : AGPL 3.0 - Site : https://www.mongodb.com.

MySQL est un système de gestion de bases de données relationnelles.

Licence : GPL / propriétaire - Site : https://www.mysql.com.

Nomad un planificateur et un orchestrateur de charges de travail simples et flexibles pour déployer et gérer des conteneurs.

Licence : MPL 2.0 - Site : https://www.nomadproject.io/.

PostgreSQL est un système de base de données objet-relationnel puissant et open source, fiable, avec des fonctionnalités robustes et performant.

Licence : PostgreSQL - Site : https://www.postgresql.org/.

Rancher est une plateforme de gestion de conteneurs open source développé pour les infrastructures en production.

Licence : Apache 2.0 - Site : https://rancher.com/.

Redis est un système de gestion de base de données clé-valeur extensible, très hautes performances, écrit en C. Il fait partie de la mouvance NoSQL et vise à fournir les performances les plus élevées possible.

Licence : BSD - Site : https://redis.io/.

Apache Samza permet de créer des applications à état qui traitent les données en temps réel à partir de plusieurs sources.

Licence : Apache 2.0 - Site : https://samza.apache.org/.

SciDB est un système de gestion de bases de données (SGBD) orienté colonnes, conçu pour la gestion et l'analyse de données multidimensionnelles communes aux applications scientifiques, géospatiales, financières et industrielles.

Licence : propriétaire - Site : https://www.paradigm4.com/technology/scidb-platform-overview/.

Apache Storm est un framework de calcul de traitement de flux distribués. Il utilise des "spouts" et des "bolts" créés sur mesure pour définir les sources d'informations et les manipulations permettant un traitement par lots et distribué des données en continu.

Licence : Apache 2.0 - Site : https://storm.apache.org/.

Singularity est une plateforme de conteneurisation open source permettant d'exécuter des applications complexes sur des clusters à hautes performances.

Licence : BSD - Site : https://sylabs.io.

Apache Zookeeper est un service centralisé qui maintient les informations de configuration, les noms des services, fournit un synchronisation distribuée et des services de groupe.

Licence : Apache 2.0 - Site : https://zookeeper.apache.org/.