A Distributed Lattice Boltzmann-based Flow Simulator

Gérard Dethier

EECS Department
University of Liège (ULg)

February 22th 2009 / EDi'09

Outline

Université de Liège

Outline

Université de Liège

Context

- Flow simulator (LaBoGrid) used by Laboratory of Chemical Engineering of ULg (LGC).
- X-ray tomography gives access to complex geometries (packed beds, metallic foams, . . .).
- More accurate description of phenomena occurring in such geometries $\longrightarrow$ Better design, efficiency of related processes.
- Computational Fluid Dynamics (CFD) tools help to reach this goal.

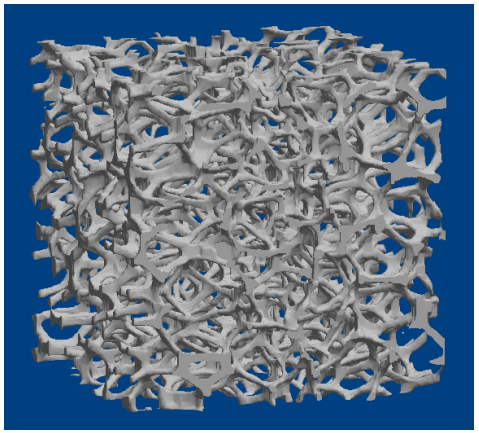
Université de Liège

Outline

Université de Liège

Why Lattice Boltzmann?

- Tomography produces large matrices of voxels (solid in which fluid flows).
- Lattice Boltzmann (LB) methods:
  - can directly use voxel matrices,
  - are easily parallelized,
  - . . . but require large computing power and memory.

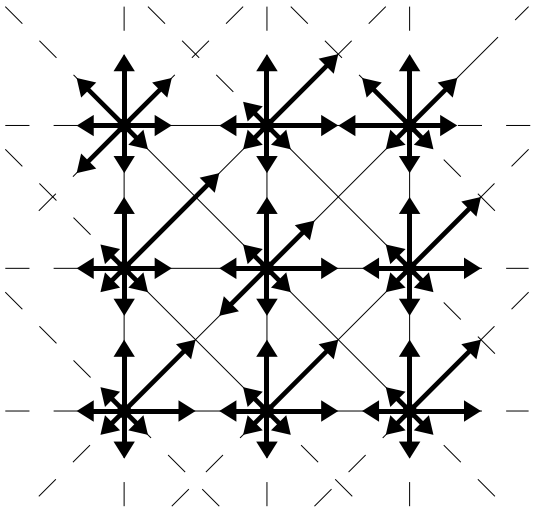


Solid of metallic foam

Université de Liège

Lattice Boltzmann methods

- Fluid = fictive particles moving and colliding on a lattice (= regular mesh, space representation).
- State of lattice point = amount of particles moving in fixed directions.
- State at time $t+1 = f$(state at time t,state of neighbors at time t).

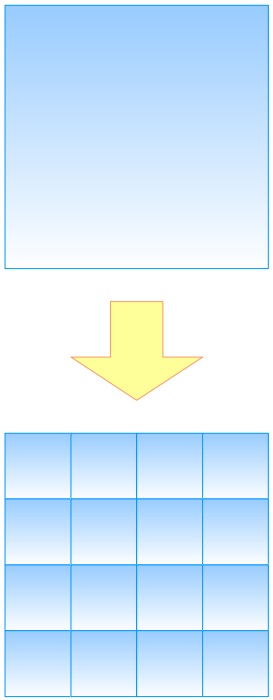


Arrow size = amount of particles moving in a direction.

Université de Liège

Distributed Lattice Boltzmann Code I

- Lattice (and solid) is decomposed into sub-lattices (sub-solids) of same sizes.



2D Lattice decomposed into 16 sub-lattices

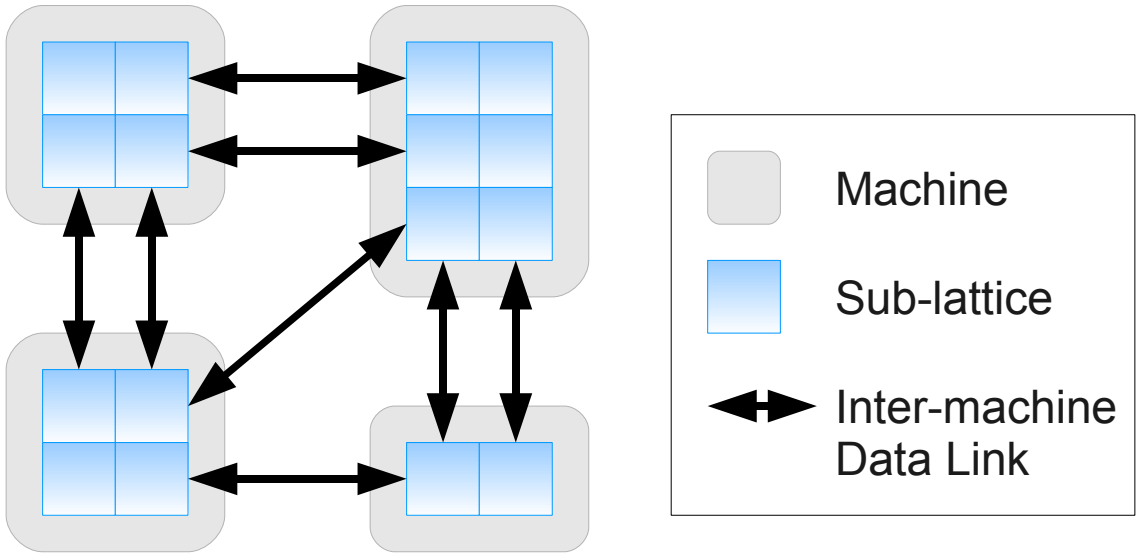
```
i := 0;
do    i < iterations →
      "Send outgoing data";
      "Receive incoming data";
      "Update sub-lattice points";
      i := i + 1
od
```

Sub-lattice update Code

Université de Liège

Distributed Lattice Boltzmann Code II

- Sub-lattices can be deployed on different machines.
- Data link between sub-lattice A and sub-lattice B = outgoing data of A are incoming data of B.



Université de Liège

Distributed LB Code Representation

- Distributed LB code can be represented by a graph.
- Node = Sub-lattice (and associated sub-solid).
- Edge = Data link between sub-lattices.
- This graph is called Model Graph.

Outline

Université de Liège

Environement : Dynamic Heterogeneous Cluster

- Heterogeneous software environements
  $\longrightarrow$ Java
- Heterogeneous computing power
  $\longrightarrow$ Load Balancing
- Limited reliability
  $\longrightarrow$ Fault Tolerance
- Available computing power fluctuations (background load)
  $\longrightarrow$ Dynamic Load Balancing (DLB)

Université de Liège

Outline

Université de Liège

Definitions

A Machine Graph

- Represents a cluster
- Node = Machine
- Edge = Network link
- Nodes are labelled with computing power

Mapping graph $G_1 = (N, E)$ onto graph $G_2 = (M, F)$

- Partition $P(G_1)$ with $|P(G_1)| = |M|$
- $i \in [1..|M|] : P_i(G_1) \longrightarrow M_i$

Université de Liège

Load Balancing

- Optimization problem:
  - Map Model Graph M onto Machine Graph R
  - Mapping minimizes execution time
    - $|P_i(M)| \propto$ power of machine i.
    - Edge cut is minimized $\longrightarrow$ network data transfers minimized
- Static load balancing: load balancing is done ones for all before execution.
- Existing tools: JOSTLE, SCOTCH, METIS, ...

Université de Liège

Dynamic Load Balancing Requirements

- Load balancing can occur several times during execution.
- Additional requirements, mapping must be:
  - Calculated fast ($\longrightarrow$ parallel/distributed algorithm).
  - Incremental.
  - Calculated in parallel of simulation execution.
- Another additional requirement: no centralized architecture.

Université de Liège

Existing Methods

- Iterative optimization algorithms.
- Optimal workload estimation for each machine (size of Model Graph partition) based on local information (workload estimation of subset of available machines).
- After convergence, migration of work (Model Graph nodes) from overloaded to underloaded machines.

+	−
• Easy to implement • Decentralized	• Slow convergence

Université de Liège

Proposed Approach

Enhanced Tree Walking Algorithm (Shu'96):

1. Machine Graph is organized into a tree.
2. Bottom-up (leaves $\longrightarrow$ root) message propagation to compute optimal workload.
3. Top-down (root $\longrightarrow$ leaves) message propagation to schedule work migration.
4. Local partition refinements (modified KL (Kernighan-Lin'90) refinement with neighboring partitions).

Université de Liège

Remarks

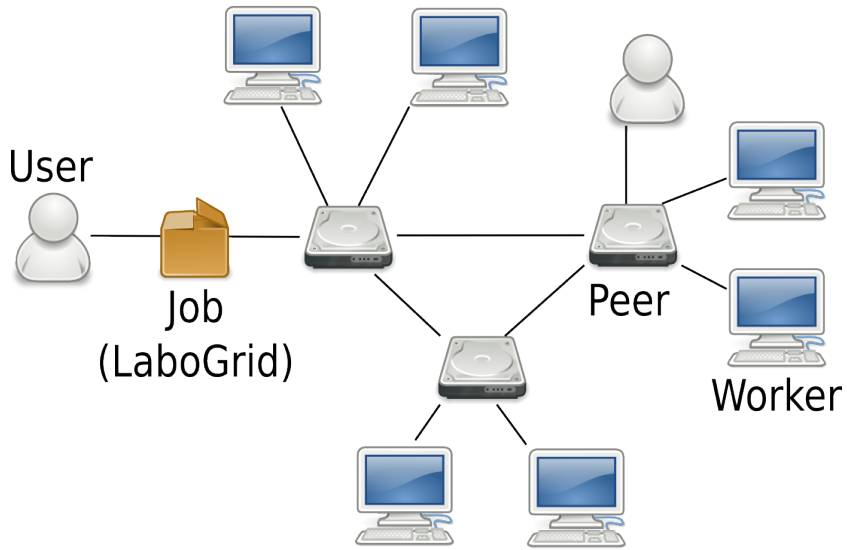
- No iterative optimization $\longrightarrow O(10^3)$ less messages, no slow convergence.
- Distributed and scalable.
- Incremental.

Outline

Université de Liège

LBG-Square

- Joint work with colleague Cyril Briquet (ULg).
- Combination of Flow Simulator (LaBoGrid) with a P2P Grid Middleware (LBG).
- LaBoGrid = Job of LBG.



Université de Liège

Lightweight Bartering Grid (LBG)

- Peer schedules tasks on its workers ($\longrightarrow$ peer schedules local tasks).
- 1 Task / Worker.
- Tasks can be submitted by peer A to peer B to accelerate job's execution ($\longrightarrow$ peer B schedules remote tasks).
- When peer schedules local tasks, remote tasks' execution can be interrupted (priority to local task)
  $\Rightarrow$ Tasks preemption

Problem:

- Task interruption (=failure) $\longrightarrow$ flow simulation cannot continue!
- A solution: Checkpoint/restart mechanism.

Université de Liège

Checkpoint/restart mechanism

1. Each task saves its state on a regular time basis (checkpoint).
2. In case of failure, all tasks restore their previous saved state (roll-back).
3. The states associated to lost tasks are restored by new or remaining tasks.
4. Tasks are restarted from restored state.

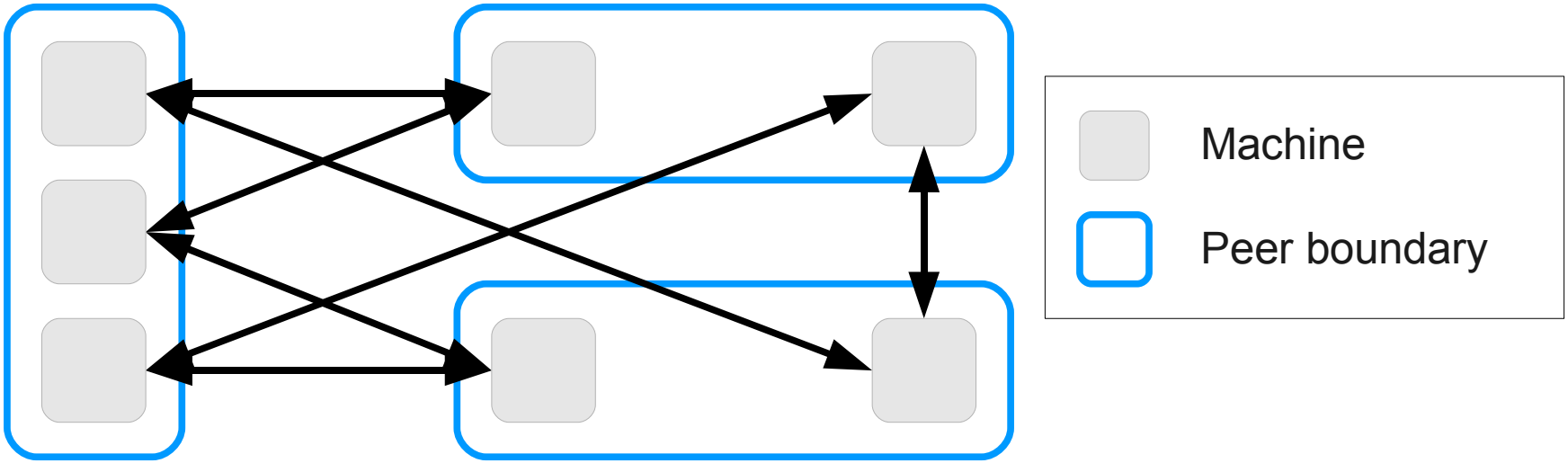
NB: State of LaBoGrid task = values of associated sub-lattices

Proposed Approach : Distributed Checkpointing

- Task state is replicated on a subset of available machines
  $\longrightarrow$ Several state replicas per task.
  - Robust (No single point of failure).
  - Scalable (No central bottleneck).
- State is saved to disk (reduced memory footprint).
- Parameters: number of state replicas and checkpointing frequency.
- Goal : $\geq$ 1 state copy / task available in case of task(s) interruption(s).
  $\Rightarrow$ What state on what machine?

Université de Liège

Checkpointing Graph



Properties:

- Fixed number of outgoing edges per node (number of state replicas).
- Load balancing on the number of incoming edges per node.
- Edges should always cut peer boundaries (tolerance to preemption).

Université de Liège

Outline

Université de Liège

Summary and Future Work

- Distributed implementation of a Lattice Boltzmann fluid flow simulator.
- Fast Dynamic Load Balancing.
- Distributed checkpointing.
- What needs to be done?
  - Integrate described dynamic load balancing method (centralized method currently used).
  - Further decentralization.

Thank you for your attention!

Some figures of these slides use pictures from The Gnome Project.

`http://www.montefiore.ulg.ac.be/~dethier/`

Université de Liège