Adapting a Multi-Agent Soil Simulation on GPU

Journée RGE - Belfort

Guillaume Laville

FEMTO-ST/DISC - Université de Franche-Comté

Feb 2013

Overview

1. Context
   - Multi-agent Systems (MAS)
   - Sworm
   - MIOR
   - GPGPU
2. Implementation
3. Results
4. Future works

Multi-Agent Systems

Used to simulate complex models

Natural processes, social behaviours...

... based on entities

- Agents (passive ou active actors)
- Environment (store the global properties of the model)
- Behaviours and interactions (often message-based)

Multi-Agent Systems

Generic platforms for MAS simulation

Repast, Madkit, JADE, Netlogo...

... most of these use traditional CPUs

- Sequential execution
- Shared memory parallelism, using IPC or threads
- Distributed unto multiple host, using native sockets or libraries such as MPI

Multi-Agent Systems

Problematic

Could this type of simulation be executed on one or multiple GPU, using generalised rules of mapping, to exploit the inherent parallelism of this new architecture ?

Logic

Would allow faster execution and/or better scaling on readily available GPU, available at a low cost for any researcher, as well as usage of a new type of hardware platform emerging on supercomputers ?

About the Sworm model

Overview

- Simulation of soil evolution
- Written in Java using the Madkit platform
- Developped for the IRD

Soil considered as a multi-scale agent-based simulation

- Macroscopic ones (earth worms)
- Microscopic ones (aerobic bacterias) ⇒ **Mior**

Sworm multi-scale model

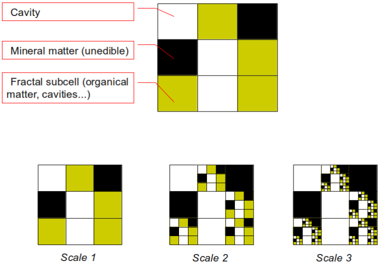

Cavity
Mineral matter (unedible)
Fractal subcell (organical matter, cavities...)
Scale 1
Scale 2
Scale 3

FIGURE: Sworm multi-scale representation

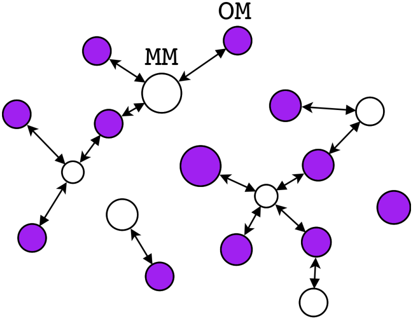


FIGURE: MIOR representation

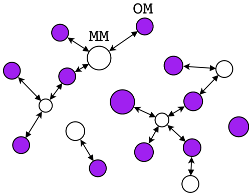
Microscopic scale : MIOR simulation

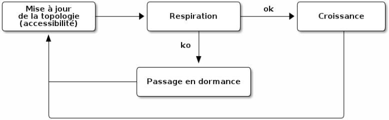
Based on two kind of agents

- Microbial colonies (called `MM`)
- Carbon deposits (called `OM`)

Two main evolution processes linked to the `MM`

- Breathing
- Growth




Mise à jour de la topologie (accessibilité)

Respiration

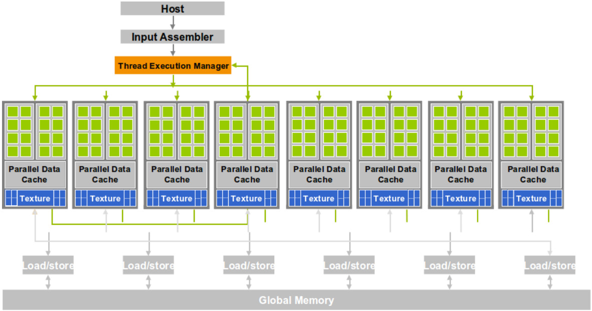
ok

Croissance

ko

Passage en dormance

Overview of a graphic card



What is GPGPU ?

Principe

Use modern, programmable graphic hardware as a generic execution platform for parallelism.

Two main languages available

- CUDA (NVIDIA)
- OpenCL (AMD, NVIDIA, Intel, Khronos Group)

Multipe Java bridges

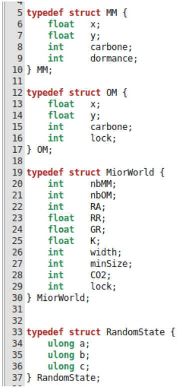
- JCUDA (www.jcuda.org)
- **JOCL (www.jocl.org)** ⇒ Retained solution
- JavaCL, LWJGL, JOCL (jogamp.org)...

Model mapping on OpenCL concepts

Model	OpenCL
Agent properties	C-like Structures
Agent actions	Device-local functions
Simulation step	OpenCL kernel

Four main data structures

- MM array
- OM array
- Topology matrix
- Environment (MiorWorld)



```
typedef struct MM {
    float   x;
    float   y;
    int     carbone;
    int     dormance;
} MM;

typedef struct OM {
    float   x;
    float   y;
    int     carbone;
    int     lock;
} OM;

typedef struct MiorWorld {
    int     nbMM;
    int     nbOM;
    int     RA;
    float   RR;
    float   GR;
    float   K;
    int     width;
    int     minSize;
    int     CO2;
    int     lock;
} MiorWorld;


typedef struct RandomState {
    ulong a;
    ulong b;
    ulong c;
} RandomState;
```

Execution of multiples agents at the same time

Topology

Each GPU thread is associated to an (OM, MM) couple.

Carbon distribution

Each GPU thread is associated to an OM agent.

Breathing/growth

Each GPU thread is associated to an MM agent.

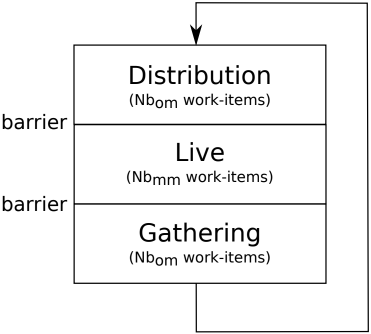


FIGURE: Execution decomposition

Data structures optimizations

Topology

- two-dimension low density matrix
- walkthought-optimized matrix

Carbon parts

- parts in global device memory
- parts cached in local memory

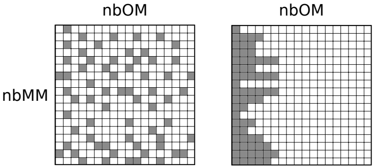


FIGURE: Topology representations

Results for one simulation

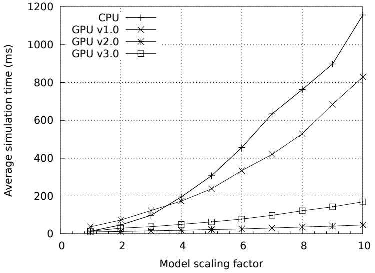


Figure: NVIDIA GeForce 8800GT

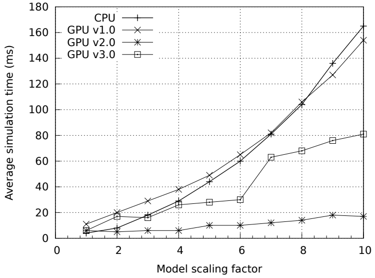


Figure: NVIDIA Tesla 1070

Results for a fixed number of simulations

- fixed number of simulations to execute
- varying the number of execution launched in one GPU execution

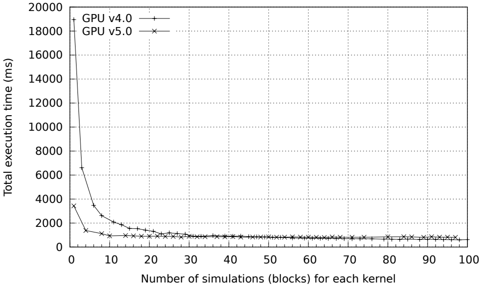


FIGURE: Execution time of 1000 simulations

Future works

- Synchronize MIOR execution on GPU with the existing Sworm CPU computations.
- Allow the distribution of GPU computations on multiple GPUs across multiple nodes.
- Propose a generic execution model for computing-intensive, environment-wide model evolutions in ABMs.

Future works

- Synchronize MIOR execution on GPU with the existing Sworm CPU computations.
- Allow the distribution of GPU computations on multiple GPUs across multiple nodes.
- Propose a generic execution model for computing-intensive, environment-wide model evolutions in ABMs.

Future works

Generic GPU execution platform

- Provide generic implementations of common MAS algorithm (diffusion, computation of distances in 2d and 3d, matrices reduction...)
- Modular : Each set of functions regrouped in modules
- Extensible : Possibility to add modules to treat new problems.

Thanks you

- Questions ?