GPU Delegation: Toward a Generic Approach For Developing MABS using GPU Programming GPU delegation for MABS

Emmanuel Hermellin, Fabien Michel

GPU Delegation: Toward a Generic Approach For Developing MABS using GPU Programming

Emmanuel Hermellin and Fabien Michel {hermellin, fmichel}@lirmm.fr

Laboratory of Informatics, Robotics and Microelectronics (LIRMM) - University of Montpellier

Objective

Considering the use of GPGPU for developing MABS, the conclusion of Perumalla and Aaby's work (P&A) [1] in 2008 was twofold:

- **Data parallel execution capabilities of GPU** can afford **excellent speedup on MABS** models;
- But this comes at the expense of **modularity**, **accessibility** and **reusability**.

Objective: Check if the conclusions outlined by Perumalla and Aaby are still true despite the evolution of GPGPU and MABS.
Method: Applying *GPU delegation* on four models to compare CPU and GPU implementations and analyze the results from both a conceptual and a performance point of view.

GPU delegation for MABS

Motivation: Implementing MABS using GPGPU is very challenging.

- GPU programming relies on specialized hardware.
- It is very difficult to deport the entire MABS model on GPU.

Approach: GPU delegation [2] relies on an *hybrid approach* (Figure 1) and consists in making a clear separation between:

- The agent behaviors, managed by the CPU;
- Environmental dynamics, handled by the GPU.

GPU delegation principle: *Any agent computation that does not modify the agent's states could be modeled as an endogenous dynamics of the environment, and thus considered as a potential GPU environment module.*

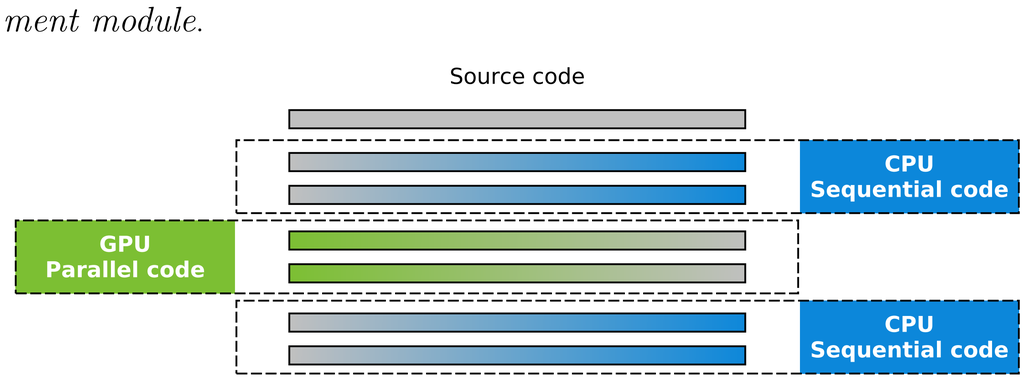
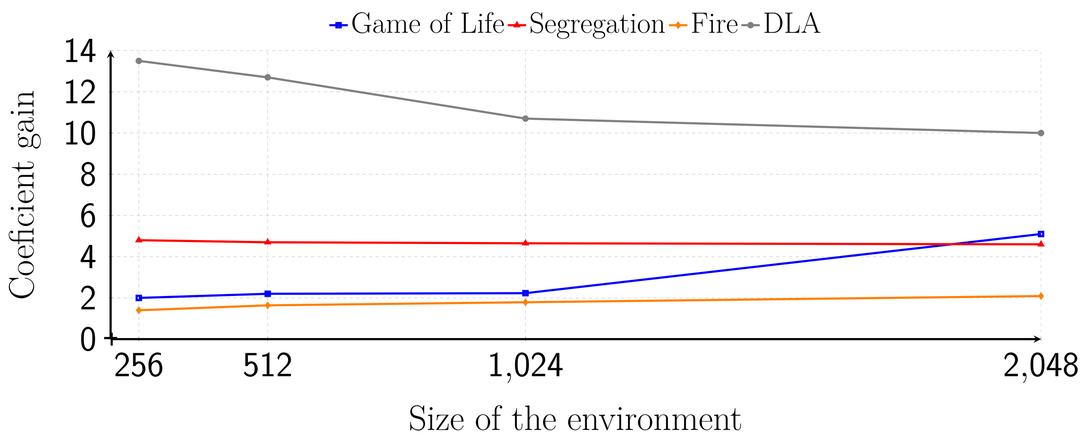


Figure 1: Illustration of an Hybrid approach

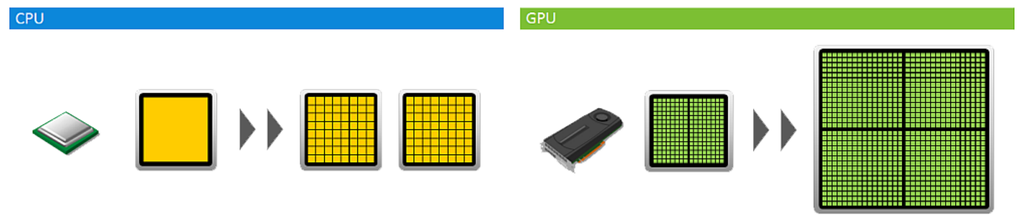
Performance perspective



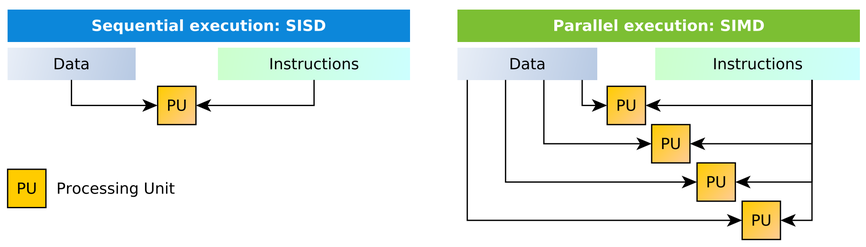
Results: Performance gains vary significantly depending on the simulated model, the size of the environment and the density of agents (gains can reach x14 but is more likely between x2 and x5).

What is GPGPU

General-Purpose computing on Graphics Processing Units (GPGPU) consists in using the **highly parallel architecture** of Graphics Processing Units (GPU) to do **General-Purpose computing** and thus provide **high performance gains**.



Called *Stream Processing*, the programming model associated to GPGPU (*SIMD*) relies on the execution of a series of operations on a dataset simultaneously.



Experiments

Experience: Applying GPU Delegation on four models.

- Conway's Game of Life and Schelling's segregation (also conducted in P&A study).
- Fire and DLA model (taken from the Netlogo models library).

Design process: For each model, the purpose is to identify some computations that can be modeled as environmental dynamics and then translated into GPU modules.

- **1.** Find intensive computations that comply with GPU delegation.
- **2.** Translate sequential computations into computations performed by the GPU.
- **3.** Implement GPU modules.

Example: Fire model.

- **1.** Intensive computation: The heat diffusion.
- **2.** Translate computation: See Figure 2.
- **3.** Figure 3 illustrates how this computation can be integrated into independent GPU module and then used in the TurtleKit platform.

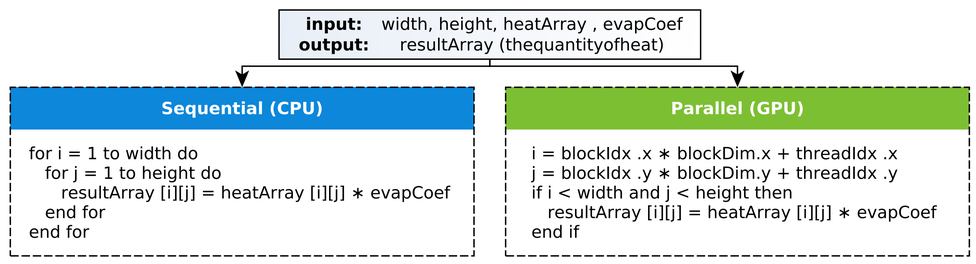


Figure 2: Sequential vs Parallel implementation

One major characteristic of the GPU code: The parallelization of the loop is realized by the hardware architecture.

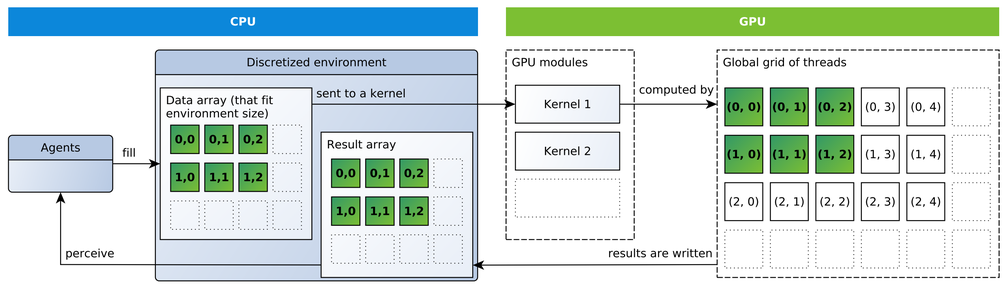


Figure 3: Integration and use of GPU modules

Software engineering perspective

Improvements since P&A's study:

- **Genericness**: May be used on a wide variety of MABS models;
- **Reusability**: Reusing created GPU modules;
- **Accessibility**: Promoting a modular design of the model, thus producing simple kernels (little GPGPU knowledge).

GPU delegation benefits: As a generic approach, GPU delegation allows to take advantage of the computing power of GPGPU without changing the agent model.
Perspective: Formalizing the modeling process related to GPU delegation by defining a methodology based on GPU delegation for developing MABS using GPGPU.

References

[1] Kalyan S. Perumalla and Brandon G. Aaby.
Data parallel execution challenges and runtime performance of agent simulations on GPUs.
Proceedings of the 2008 Spring simulation multiconference, pages 116–123, 2008.

[2] Fabien Michel.
Translating Agent Perception Computations into Environmental Processes in MABS: A means for Integrating Graphics Processing Unit Programming within Usual Agent-Based Simulation Platforms.
Systems Research and Behavioral Science, 30(6):703–715, 2013.

http://www.lirmm.fr/~hermellin/
http://www.lirmm.fr/~fmichel/