SugarScape on Steroid: Simulating Over a Million Agents at Interactive Rates

Presenters: Roshan M. D’Souza, Mikola Lysenko

Agents 2007
Chicago, IL
Outline

• Introduction
  – Agent-based modeling (ABM)
  – Desktop and cluster ABM systems
• Why use GPU?
• General purpose computation on the GPU (GPGPU)
  – Programming GPUs
• ABMs on GPUs
• Implementation of SugarScape ABM
• Conclusion and future research
Introduction

• **Agent-based modeling (ABM)**
  – Evolution of a system is simulated by its individuals
  – Individuals are heterogeneous and have behavioral rules
  – Intelligent individuals

• **Advantages of ABM**
  – Capturing emergent phenomena
  – Natural description of complex systems
  – They can incorporate empirical and historical data into the model
  – Highly non-linear systems

• **Applications**
  – Biological systems
  – Stock markets
  – Social sciences
Desktop ABM platforms

- **SWARM (heat bugs)**
- **MASON (ant colony)**
- **NETLOGO (AIDS)**

**Objective C**

**Java**

**Its own Language interpreter**

**Advantages:**
- Quick prototyping
- Based on popular programming languages and concepts such as OOP

**Disadvantages:**
- Limited performance
Clusters

- Expensive
- Difficult to program
- Limited by the communication times between CPUs
- Cannot efficiently simulate more than 100,000 agents
- Adding more CPUs does not linearly increase the speed
- Difficult visualization

Example: SIMPDEL
Agent population limited to 20000 on a static environment
Communication Issues

- Divide world and agents into chunks, distribute over processors
- Agents may interact or move across boundaries
- Large communication costs are unavoidable and unpredictable

Inevitable inter-processor communication in cluster computation [SIMPDEL]
Why GPU?
- High memory bandwidth = fast communication
- Massively parallel
- Strong floating point capabilities
- Cheap
- Rendering and visualization features

Radically different architecture
- Data parallel

Memory bandwidth comparison of CPUs and GPUs
The GPU devotes more transistors to data processing [NVIDIA CUDA]
Floating-Point Operations per Second for the CPU and GPU [NVIDIA]
Two modes of operation: fixed and programmable

Programmable processors are used in general purpose GPU computation

Several vertex and fragment processors are available on the GPU to aid parallelism
• Use the GPU to execute a program

• Textures = Data
• Shaders = Program
• Rendering = For loop
Data Update in GPGPU

- Textures can’t be read and written at the same time
- Two textures are used with a processor to update data (ping pong)
- We must “draw” a geometry to write data into a texture
  - Draw a quad

Ping pong operation
Environments on the GPU

- Environments (Static Agents) can be packed directly into a texture using RGBA values for each state
- Use the ping-pong technique to update each cell in parallel
- Common functions include
  - Cellular automata [James 2001]
  - Coupled map lattice [Harris et al. 2002]
  - Convolution filters [Yang and Welch 2003]
  - Lattice Boltzmann methods [Li et al. 2005]
Mobile Agents

• Much harder than environments
  – Not stuck to a fixed point
  – Dynamic neighborhood

• 3 Problems:
  – Storing and updating agent states
  – Calculating interactions
  – Reproduction
Managing agent states

- Store the agent states within a stack of textures

- Updating mobile agent can be done using ping-pong

- Referencing the environment is done using a double look up on the agent’s position
  - Can be used for vision

```c
struct AGENT {
  float x;
  float y;
  float health;
  ...
};
```
Agent Interactions

- Use a scatter operation [Buck 2005]
- The index of the agent is used as an ID
- Agent scatter stores each ID at the corresponding position in a collision map
Reproduction

- Memory has to be allocated for new agents

- We cannot do regular serial allocation
  - GPU memory model not efficient for serial allocation
  - Transferring memory allocation to CPU is very slow

- Stochastic parallel allocation on the GPU
  - Randomized algorithm with O(1) time cost.
  - Asymptotically faster than serial CPU allocation
Stochastic Parallel Allocation

Initial state

Random search with r=2

Final state
Putting it all together

Loop three steps:
- Update environment
- Update agents
- Scatter agents on environment
Over One million SugarScape Agents

Environment size: 2560x1024
Agent State Texture: 2048x2048
Frame rate: >50 FPS
Scalability

![Bar Chart]

- Updates per second:
  - 2.62E+05
  - 5.24E+05
  - 1.05E+06
  - 2.10E+06

- Number of agents

Manufacturing Computations Lab
Dept. of Mechanical Engineering-Engineering Mechanics
Michigan Tech. University
Conclusion

• First implementation of ABM simulation on the GPU
• GPUs are viable and efficient platforms for simulating ecology models
• Simulating relatively complex ABMs with more than one million agents on a high resolution land map is possible with GPUs
  – Dynamic environment
  – Real-time user-interactivity
  – Real-time visualization
  – Stochastic allocation scheme for reproduction
Future Research

• Need new visualization strategies to deal with massive amount of data
• Simplify programming
• Real-time statistics gathering on the GPU
  – Tracking population sizes
  – Averages of various agent properties
Questions?