Multi Agent Simulation and its optimization over parallel architecture using CUDA™

Abdur Rahman and Bilal Khan NEDUET(Department Of Computer and Information Systems)

INTRODUCTION

In this project we are determined to show that large scale multi-agent systems can benefit from the use of parallel computing and it will show massive enhancement in performance. NVIDIA (GPU) allow general purpose computing on graphics processing units by making use of CUDA technology.

ACHIEVING OUR GOAL

We will implement a distributed application for crowd simulation which is one example of Multi agent modeling. The application will be written in any high level language such as C++

In the next phase, we ported that original application code onto GPU by finding out the parallelizable portions in that code. By running the compute intensive code on multicore GPU will show massive accelerated performance. NVIDIA CUDA architecture will be used for that purpose.

PARALLEL DESIGN MODEL

In this project we are determined to show that large scale multi-agent systems can benefit from the use of parallel computing and it will show massive enhancement in performance. NVIDIA (GPU) allow general purpose computing on graphics processing units by making use of CUDA technology.

MULTIAGENT SYSTEMS

A dynamic environment consisting of many agents (which may represent cells, species, individuals, firms, nations) acting in parallel, constantly acting and reacting to what the other agents are doing.

The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents.

PORTING APPLICATION ONTO GPU

Porting this application onto the GPU requires us to determine the portions that could be ported onto the GPU. While developing the CPU based application using Opensteer we figured out that the architecture of the Opensteer can be modified in two phases.

• Simulation Phase
  In this phase the steering force for an agent is calculated depending on the steering behaviour. There is no modification in position, velocity or alignment

• Modification Phase
  In this phase the previously calculated steering force is used to modify the vehicle properties like position, velocity and alignment

The simulation and modification phase are the portions that we will port onto the GPU.

OPENSTEER

Developed by Craig Reynolds for Sony Cooperation

It's a C++ library to help construct for steering and motion behaviors of autonomous characters in games and animation. It also takes care of all the graphics related work.

INTRODUCING HUMAN BEHAVIOR

Human like behaviors include: Obstacle avoidance, Close-neighbor avoidance, Far-neighbor avoidance. Wandering. These behaviors were integrated with the help of Opensteer to develop CPU based crowd dynamics application

NEIGHBORHOOD SEARCH

In the Opensteer library, the neighborhood dependent behaviors require every agent to review every other agent that is alive. That means if there are n agents then each agent will determine/study the position and alignment of (n-1) agents. This would result in a complexity of O(n²).

There are multiple approaches to improve the inefficient brute-force neighborhood search mentioned above. One of these approaches is called static grid neighbor search: assumes that the simulation takes place in a bounded sphere called World with a radius called World Size. A vehicle trying to leave this sphere is automatically placed on the opposite side of the sphere. The world is divided in equally sized cubes, called cells so that each vehicle is assigned to exactly one cell. A neighbor search can now be reduced to vehicles that are located within cells around the vehicle for which the neighbor search is done. The number of cells that have to be checked depends on the search radius.

CHANGES IN PHASE TRAVERSAL

Simulation: CPU Code

Modification: GPU Code

Creating Real World Scenarios With Satellite Images Using GIS

With the help of this feature we will be able to map real world scenarios and environment developed by the concerned governing authorities into our application.

Hardware used: Nvidi volta 460 gtx

• CUDA Cores : 336
• Memory Bandwidth: 115 GB/s
• Graphics Clocks: 670 MHz
• Processor Clock: 1350 MHz
• Price: $165