15-740 Project Proposal

Michael Ashley-Rollman, Michelle Goodstein, Paul Zagieboylo

October 23, 2007

1 Proposed Project

We propose to parallelize the Open Dynamics Engine (ODE).

2 ODE

The Open Dynamics Engine (ODE)[1] is an open source physics engine for simulating rigid body dynamics. It is used in many games and 3D simulation tools to provide physics simulation support. As games strive to be more realistic, physics simulations become more and more important for creating believable, accurate simulations of complex things like troops under mortar attack as well as simple things like dropping items and letting them fall to the ground in the appropriate manner.

At present ODE is single threaded. This is currently a problem for 3D simulation tools as it prevents them from gaining significant benefit from running on a supercomputer, a cluster, or even just a multi-processor machine. This can be extremely inconvenient and frustrating for people running these simulations as they frequently take a very long time to execute. The dprsim simulator for DPR in particular is known to suffer from this problem. Also, while less urgent, a multi-processor capable version of ODE would greatly benefit games in the long run. While most games are presently single-threaded, the move towards multi-core processors rather than faster processors will require games to become multithreaded in the future to benefit from newer hardware. As this happens it will become necessary to have a multi-processor capable physics simulator.

3 Potentially Required Resources

We have identified several potentially required resources.

- ODE source

  We need access to the original serial code so that we can modify it to compile with OpenMP libraries, and insert parallelization. We also need access to the original version of ODE so that we can test how our parallel modifications either help or hurt the performance of the physics lesson.

- Parallel computers

  Access to different types of parallel architectures may allow us to examine how different architectures impose different types of overhead that may make our parallelization achieve a speedup or not.

  - Cobalt/Rachel

    We already have access to these machines. They will allow us to test our parallelization on traditional supercomputers.

[1] `http://www.ode.org/`

  - Intel Condor Cluster

    We have spoken with Dave O'Halloran, who seems interested in the project and has the ability to provide access to the cluster. This would provide a second architecture for us to test our modifications on.
  - Todd's 8-ways

    Using an 8-way processor with all the processors on one machine (negating the need to send data over the network between machines) may provide a way of measuring how much of our overhead is due to latency in sending data.

- Interesting test cases

  We will need to obtain some useful test cases that we can use to determine whether our efforts achieve a speedup. The dprsim simulator, running only physics simulations is a potential source of test cases.

4 Proposed Timeline

1 week–October 30

Analyze the ODE codebase, and identify sections that look like they could be parallelized effectively.

2 weeks–November 6

Look into research done into parallelizing similar systems, and extract several methods that look promising for parallelizing ODE. Order these by those that appear the most promising.

3 weeks–November 13

Have at least one attempt to parallelize ODE implemented and ready to submit in batch mode to the parallel architectures that are available. All interesting test cases should have been obtained by now, so that they can be submitted with jobs.

4 weeks–November 20–project milestone

Obtain preliminary results from at least one parallelization method. Also, another method of parallelization should be at least one week into being coded.

5 weeks–November 27

Code is final, and ready to submit to parallel architectures available to us. At this point, all methods of parallelization that we will be testing should be coded.

6 weeks–December 4

Writeup is due. All results have been obtained and analyzed.

5 Success Criteria

We propose three different levels of success criteria.

75% goal

We propose to consider 75% success as applying several standard methods of parallelization to ODE, but not necessarily succeeding in achieving a measurable speedup. In this case, success would be the ability to pinpoint several methods which work on other problems that fail to work here, potentially highlighting areas which are inherently serial or harder to parallelize.

100% goal

We propose to consider 100% success if we can find critical sections of ODE that are run frequently and achieve speedup via parallelization using at least one standard method.

125% goal

We propose to consider 125% success if we can find critical sections of ODE that are run frequently and achieve speedup via parallelization using more than one standard method. This would allow us to analyze which method worked best across the architectures that were available to us.