CIS 510 Final Project

Distributed Partial Differential Equation Solver For 3-Dimensional Models

(diffyq)

Project Team:
Adam Martini
Wes Erickson
Ran Tian

April 21, 2014

Executive Summary

We will design a flexible system for solving large partial differential equations in a parallel distributed system. The product of our this project will solve 3-dimensional models of any size for a given number of time steps. Our system will take any size model and process it efficiently in parallel on multiple nodes.

1 Project Description

DIFFYQ is a distributed partial differential equation (PDE) solver for 3-dimensional models. This systems takes advantage of the memory and processing power of multiple nodes to efficiently compute the PDE. Given a 3D dataset, a user defined set of update equations, a time step granularity, and a number of steps; our system automatically marshalls the data and computational code to multiple nodes to initialize processing. The PDE is then computed for the given number of steps and the final result of stored on the main node.

A primary goal of this project to create a generalized application that can used to solve the PDE for any waveform. To accomplish this, we allow the user to provide a set of user defined functions as input for computation. Our system uses these functions to iteratively update the PDE matrix during parallel processing. We carefully outline the format for these user defined functions in Section 2.

Communication between nodes allows the system to calculate the border values of between the divisions of the PDE matrix. This is an example of the stenciling pattern for parallel processing. On each node, our program takes advantage of opportunities for shared-memory parallelism through map-reduce and divide conquer techniques for computation update components.

2 Highlevel Architecture

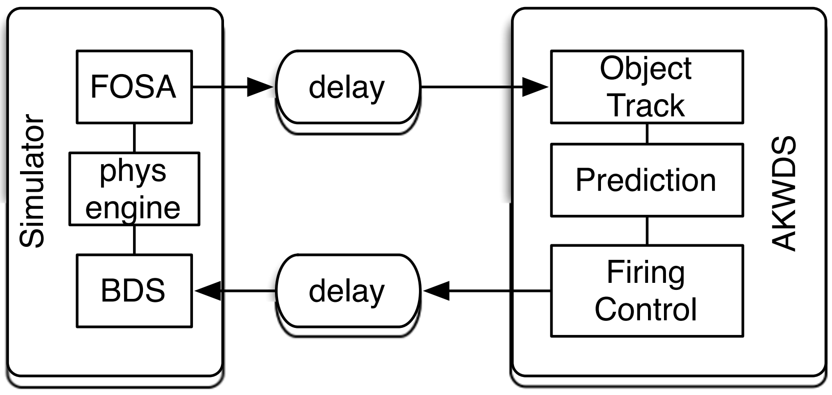

Figure 1: High Level Architecture Diagram

Figure 1 shows a block level diagram of the AKWDS software (without the visualization component). There are two main programs in the solution.

AKWDS This is the control software that OSNAP needs in support of its mission to plan for and execute plans related to real and imagined extraterrestrial encounters. In this case, a hostile invasion scenario.

Simulator The Simulator is responsible for emulating the world that a deployed AKWDS system would be deployed in. This software supports evaluation and tuning of AKWDS without enduring actual extraterrestrial invasion.

AKWDS and the Simulator will be connected over a network. The Simulator will be responsible for handling the effects of delay called out in the project description.

Simulator

The core of the simulator is a basic physics simulation. This N-body simulation will track projectiles and vehicles within the world and apply basic Newtonian physics. The simulator will expose the current location of each simulated object at regular intervals as a list of 3D cartesian coordinates and object radii. This stream will be sent using the FOSA format. The simulator will also take in a stream of BDS actions and add projectiles to the simulation as appropriate. The simulator also enforces some sanity rules regarding object motion and provides some basic flight paths for UFOs.

AKWDS

AKWDS will take the FOSA stream of object locations and attempt to formulate some tracking of the objects (object A with location L last time step is at location L' this time step). This will be used to help estimate the future location of the object so that targets can be lead correctly for firing. AKWDS will then select a set of object to fire at and set of cannons to fire from and send the needed information via the BDS stream.

Visualization

Visualization is not a core component of the project deliverable. However AKWDS debugging and demonstration will be very difficult without providing some animation of the system behavior. A visualizer, online or post-hoc, that works from the FOSA stream will be provided to help validate the software.

3 Parallel Plan

The AKWDS design includes several places to introduce and explore parallelism.

N-body The core N-body physics simulation seems like it will support up to one thread of execution per body. Efficiently handling interactions, such as collisions, can add some complexity to the solution.

AKWDS The main computation can be articulated as a processing pipeline with phases for identifying objects between samples, computing trajectories, and computing firing solutions.

Object Tracking At each sample, objects need to be identified and grouped. It seems like a non-trivial computation that could benefit from running in parallel and may be able to take advantage of stencils due to locality expectations between time steps.

Trajectories Trivially parallel. For N tracked objects, fit the data points to a function.

Firing Solutions Trivially parallel. For M targets (where M is the number of cannons to fire), compute the future state of the target and compute the required BDS heading, azmuth, and velocity to hit the target in the future.

We plan to look at system scaling by varying the number of initial objects in the simulation and the number of cannons available. We intend to look at wall clock time to complete a fixed number of simulation steps. We also intend to look at latency between the FOSA transmission time and BDS receive time to see if pipelining provides a significant benefit.

If we have extra time, we would like to explore using MPI to spread the simulator and AKWDS computations across nodes to increase the size of simulation that can be done in realtime.

4 Project Schedule

Week	Deliverable
5	FOSA and BDS stream protocol, select 3D technology
6	Basic N-body simulation with collisions, gravity, and FOSA/BDS streams
7	Basic N-body simulation with vehicles and BDS cannons
8	Bad firing control (does not lead target) and basic visualization
9	Improved firing control, pipeline vs non-pipeline measurement
10	Poster length presentation including measurements on scaling