CIS 510 Final Project

Distributed Partial Differential Equation Solver For 3-Dimensional Models

(diffyq)

Project Team:
Adam Martini
Wes Erickson
Ran Tian

April 22, 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.

The size of the fields we will be simulating can be several gigabytes of data per simulation frame, which can easily exceed the memory of single node. However, since the evolution of a point in the field depends only on the neighbors, the data can be divided among a set of nodes that process independently with a small amount of communication. Additional parallelism can be gained by parallelizing the simulation on each node by splitting the problem into cache sized chunks. Lastly, to reduce overhead due to data output, frames will be saved less frequently than the simulation rate, and will be stored on each individual node.

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 initial conditions. 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. The details of the format for these user defined functions are described 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.

Since data output rates are often a limiting factor for the runtime of, we also will apply data reduction techniques to reduce the overall size of the data that will be written to disk. These include subsampling in both time and space, and compression methods (e.g. discrete cosine transform),

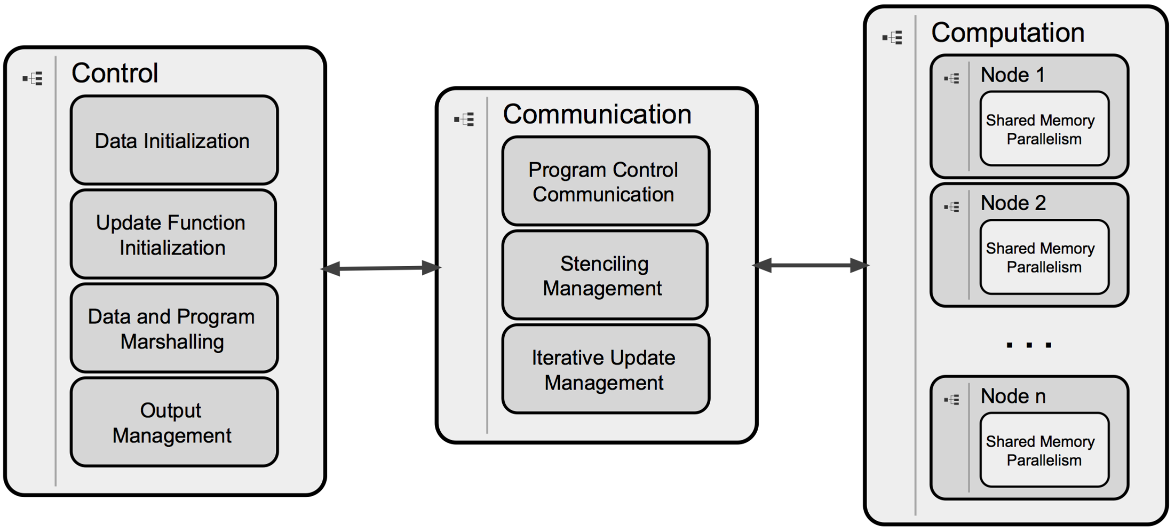

Figure 1: High Level Architecture Diagram

2 Highlevel Architecture

Figure 1 shows a block level diagram of the Diffyq software. There are three main components to the solution.

Control Layer

Communication Layer

Computation Layer

2.1 Control

2.2 Communication

2.3 Computation

2.4 Data Reduction and Storage

We will save the output files onto each node while the simulation is running, and then gather them together on a single head node afterwards. This gives us less of a bottleneck on a single hard drive. During the simulation, the output files will be stored locally on each node, split into files according to the data handled by each node and by the timestep (and perhaps by chunk).

2.5 Data Format

The data will be formatted in the standard Visualization Tool Kit (VTK) format. We will use a regular regtagonal structured grid.

3 Parallel Plan

The Diffyq design includes several places to introduce and explore/optimize parallelism.

Data and Program Marshalling Map Stenciling...

Stenciling Management Communication minimization along borders...

Node Level Parallelism Matrix Multiplication...

Data Reduction Data reduction...

1. Spacial subsampling
2. Temporal subsampling
3. Compression

4 Project Schedule

Week	Deliverable
5	
6	
7	
8	
9	
10	