Proposal

Joseph Conway's Game of Life via Multithreading

Team members

Devan Patel, Kyle Koceski, and Daniel Fuentes

March 10, 2014

0.1 Team members

1. Devan Patel - devanp92@ufl.edu
2. Daniel Fuentes - danielffuentes@ufl.edu
3. Kyle Koceski - kkoceski93@ufl.edu

0.2 Designated Submitter

Devan Patel

0.3 Project Overview

Design Our version of The Game of Life will be implemented in a server-client architecture, providing a single game instance which can be controlled and played on multiple clients. Using this architecture, the calculation of the next iteration can be computed on multiple clients, as well as multiple threads on these clients.

On the server side, the game board will be maintained and updated according to commands from clients. The board will be constantly be transmitted to clients when changes are made. When the game is played, the server will also distribute the task of calculating the next iteration amongst the clients. These clients can then calculate the next iteration from the information they have, returning this information to the server. The server can then merge the calculations, distribute the board to the clients, and distribute tasks again for the next iteration.

On the client side, clients will be able to update the grid size and initial living cells and be able to play the game. Clients will then receive commands for calculating the next iteration. After these calculations are complete, the information is sent to the server for merging and the client waits for future information.

Implementation The Game of Life will be implemented by managing the board as a list of alive cells. These alive cells can be evenly distributed to clients for calculation and determining whether nearby cells live or die. This calculation can be split further using multiple threads on each client managed by a ThreadPool. By only listing alive cells, we are hoping to take advantage of the sparsity of most boards.

0.4 How This Project Relates To Concurrent Programming

In order to maintain a consistent state with multiple clients potentially creating a board simultaneously, as well as handling the division and merging of multiple sets of data, proper concurrency techniques will need to be handled.

Alongside this, our implementation of the Game of Life will be multi-threaded; in the aforementioned it notes that there will be a selected thread dedicated to a certain number of rows (dependent on the grid size) which will determine which cells will be alive or dead in that iteration. After each iteration the game board will be updated to show how the grid

evolves.

For each iteration, there will be a timed result which calculates the time difference between the start and end of each iteration. The timed results from the concurrent solution will be tested against a sequential version; we will be able to create a graphical time plot with the give results in order to conclude if parallelism has produced a more efficient model than its sequential counterpart.

0.5 Hardware and Software Used

There will be no additional hardware required other than a multi-core laptop since our product is purely software based. The software that we plan on using are as follows:

- Backend - Java 1.7
- Frontend - JSP 2.2, CSS, HTML5, and XML.
- Testing - Apache JMeter