TEAM INFINITE TUPLES

http://www.cs.rit.edu/~bdi8241/adhoc/

Jacob Hays

Brad Israel

TEAM INFINITE TUPLES

Agenda

- Project Overview
- Design
- Demo
- Results
- Future Work
- What we learned

TEAM INFINITE TUPLES

Project Overview

- Adhoc collaborative problem solving framework.
- Focus is on fractals.
- Allow users to create fractal problems and distribute workload to other users within the adhoc network

TEAM INFINITE TUPLES

Design

- ProblemTuple – Holds problem name and fractal image paramters. Contains the Class file that generates the fractal and the specified arguments.
- ChunkTuple – Specifies a part of the problem to solve. Contains the data which is initially empty, but updated later with competed chunk.

TEAM INFINITE TUPLES

Design

- ControlPanelLogic – Sets up Tupleboard and manages the current Problems on the Network. User input starts working on problems, or creates new ones.
- CreateProblem – Reads the problem class file, gets user input parameters, and posts a new ProblemTuple with its ChunkTuples.
- SolverLogic – Keeps track of the Chunks and the state changes of them for a problem. Creates a dispay GUI to show fractal and statistics.

TEAM INFINITE TUPLES

Design

- SolverThread – Thread created to solve a chunk of a problem. Runs the class file to solve for the chunk that it is given.
- ChunkSelector – A thread started up for each problem the program is working on. It chooses what chunks to work on and creates solver threads for each one.
- ChunkCollection – A storage object for the chunk tuples, whose methods are synchronized to work with threads.
- FractalImplemetation – An Interface with a single method used to create a fractal. A class implements this interface when creating a new fractal.

TEAM INFINITE TUPLES

Demo

TEAM INFINITE TUPLES

Results: Mandelbrot Fractal Test

Image size : 2500x2500

Center point : (-0.75, 0)

Pixels Per unit (Resolution) : 6000

Number of Iterations: 1000

Break out Value: 4.0

TEAM INFINITE TUPLES

Test 1:

Baseline – A completely sequential version of the program, without Java Reflections or Tupleboard.

Total Time: 105836ms (~105.8sec)

Computational Time: 105339ms (~105.3sec)

TEAM INFINITE TUPLES

Test 2:

Baseline with Java Refections – To see how much slowdown is the result of Java Reflections, a sequential version of Mandelbrot run with Java Reflections.

Total Time: 43988ms (~43.9sec)

Computational Time: 43744ms (~43.7)

TEAM INFINITE TUPLES

Test 3:

Baseline with Java Reflections and Tupleboard – To test the slow down of Tupleboard, run Mandelbrot through our program, but only one process works on it.

Total Time: 90858ms (~90 sec)

Computational Time: 53461ms (~53sec)

Average Time per Chunk: 106ms

TEAM INFINITE TUPLES

Test 3:
Multiple Adhoc Processes

Process Count	Total Time(ms)	Comp Time(ms)	Avg Time(ms)
2	78634	54919	109
3	71393	72177	144
4	70283	81552	163
5	64331	83753	167
6	68096	87353	174

TEAM INFINITE TUPLES

Future Work:

- Give the user control in how to split up the problem.
- Implement various chunking methods, including methods that can split up chunks after the problem has started.
- Allow user to specify how many problems to automatically work on as soon as they are posted.
- Allow the user to set the number of solving threads for each problem that they are working on.
- Implement the middleware for a broader use. E.x. Image processing or use with Matlab.

TEAM INFINITE TUPLES

What we learned

- It is possible to share resources with others in an ad-hoc network.
- Requires less set up and infrastructure.
- Because of the lack of infrastructure, there is a lot more overhead and error checking that must be done.

Questions?

http://www.cs.rit.edu/~bdi8241/adhoc/