CSS 534
Final Project: Coding a Parallel Application with MASS (Multi-Agent Spatial Simulation) Library

Professor: Munehiro Fukuda
Due date: see the syllabus

1. Purpose

The final project is an open-topic programming work to write a parallel application with the MASS (multi-agent spatial simulation) library. You will choose an application for parallelization, code and run it with MASS (either Java or C++), present a summary of your work in the class, and submit your PPT file, source code, and report.

2. Choice of Project Topic

Follow the guideline below to choose a parallelizable application.

Guidance:

(1) Don't choose any of the following applications: Wave2D, Heat2D, Molecular Dynamics, and Mandelbrot. Many of former students have already parallelized these applications with MASS. Please choose a different application.

(2) Practical application: Your application program must be well known to many scientists regardless of the existence of its sequential code. Don’t make up your own application or meaningless dummy program, (e.g. a repetition or a combination of dummy arithmetic and/or logical operations onto an array or agents).

(1) Reasonable size: Your program should include 150+ lines or more with comments and spaces. Some applications may be too large or too complicated to be implemented within 3+ weeks. Therefore, you may simplify your application so as to complete your parallelization work by the due date.

Domains:

For your information, we can consider the following three application domains: (1) agent-based models (or ABMs), (2) spatial simulations, and (3) big data analyses.

(1) **Agent-based models:** simulate an emergent collective behavior of many autonomous objects, (called agents), which may be difficult to be modeled with mathematical formulae. This type of applications includes:

a. **MatSim:** http://www.matsim.org/
This is a traffic simulation that generates many vehicle agents on two-dimensional road map and observes their congestions.

b. **Wa-Tor:** http://en.wikipedia.org/wiki/Wa-Tor
Observes the population increase/decrease of predator and prey agents in the ocean.

c. **AntFarm:** http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.8353
Generates different types of ant agents, observers their food-searching behavior, and finds which agent agents have superior genes.

d. **SugarScape:** http://en.wikipedia.org/wiki/Sugarscape
Allocates inhabits over a 2D space that includes two mountains of sugar, and observes their survival along a time line.

e. **FluTE:** http://www.cs.unm.edu/~dlchao/flute/
Simulates an influenza epidemic among many communities.

(2) **Spatial simulations:** creates a 2D or 3D simulation space and simulates air, fluid, or substance dissemination or interaction of substances in the space. This type of applications includes:

   a. **Fluid Dynamics:** http://en.wikipedia.org/wiki/Computational_fluid_dynamics
      Simulates the flow of fluid substances (such as the air) in a given space.

   b. **Conway's Game of Life:** http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
      Is a cellular automaton where each cell of a given 2D space interacts with its neighboring cells to decide to be alive, terminated, or reproduced for a next simulation time.

(3) **Big data analysis:** finds a given data pattern from a 1D, 2D, or 3D dataset. This type of applications includes:

   a. **Distributed grep:** CSS534's MapReduce programming assignment
      Searches a collection files for a given keyword. Don't mimic MapReduce. Rather, you are encouraged to use agents.

   b. **Maximum/minimum temperature search:** https://github.com/cloudwicklabs/hadoop
      This is a well-known MapReduce example that searches NCDC (National Climatic Data Center) files for the maximum or minimum temperature. Don't mimic MapReduce. Rather, you are encouraged to use agents.

3. Manual and Sample Programs

The MASS library, the manual, and sample programs are found:
http://depts.washington.edu/dslab/MASS/index.html
Additional materials will be made available later at:
~css534/MASS
You may choose either the C++ or Java version of the MASS library.

4. Presentation

Plan on a 3-minute presentation of your project work in the last week. Your presentation should include:

Item	Contents	Pages in PPT
1	Application overview	1 page
2	The most important code snippet	1 page
3	An execution snapshot or a demonstration	1 page or a 1-minute demo
4	Execution performance in graphs or tables	1 page
5	Programmability consideration	1 page
		5 pages in total

Presentations will be scheduled in the alphabetical order of students' names. The audience will receive an evaluation sheet with which they evaluate each student's presentation and choice of an MASS application.

5. What to Turn in

This programming assignment is due at the beginning of class on the due date. Please turn in the following three materials: your (1) PPT file, (2) source code, and (3) report, all to be submitted to CollectIt at http://courses.washington.edu/css534/

The report must include (1) a summary of your application and parallelization techniques in 1 page, (2) your evaluation of execution performance in 1 page, and (3) your MASS programmability analysis in 1 page, total in 3 pages.

6. Bonus Points

You may hack the MASS library to improve its execution performance or add some useful features to it. Outstanding performance and/or functional improvements will give you 5 bonus points. Your report must explicitly show which portion of the MASS library you have hacked and evidences of your additional work in performance or programmability analysis.

7. Evaluation

Criteria	**Grade**
Presentation: evaluated by all the audience 15pts = very good 14pts = good 13pts = fair 12pts = poor 11pts = very poor	15pts
Application: evaluated by all the audience (except you choose one of the following appls) 15pts = very interesting 14pts = interesting 13pts = fair 12pts = not interesting 11pts = guidance not followed, (a small program or Wave2D, Heat2D, MD, Mandelbrot)	15pts
Documentation A summary of your application and your parallelization strategies including explanations and illustration in one page.	5pts
Coding Evidence 15pts = correct and clean code with an appropriate amount of comments 13pts = messy code or minor errors 10pts = incorrect or incomplete	15pts
Execution Snapshots or Demo with Graphical Results 15pts = correct execution snapshots 13pts = wrong or incomplete execution snapshots 10pts = no execution snapshot	15pts
Performance Analysis in Graphs or Tables 15pts = performance improvement with more processors 13pts = little performance improvement 11pts = incomplete performance analysis 10pts = no execution performance	15pts
MASS Programmability Analysis 15pts = good programmability analysis and/or useful bug reports presented 13pts = poor analysis (including just complaints about how much effort you made) 10pts = no programmability analysis	15pts
Bonus Point Explicitly show your performance/functional improvements added to the MASS library.	5pts
Lab4 5pts = submitted 0pts = not submitted	5pts
Total Note that this final project takes 20% of your final grade.	100pts