CSS 534
Final Project: Programmability and Execution Performance Analysis of mpiJava, MapReduce, Spark, and MASS

Professor: Munehiro Fukuda

Due date: see the syllabus

1. Purpose

The final project is an open-topic programming work as well as a group work to write a parallel application with each of mpiJava, MapReduce, Spark, and the MASS (multi-agent spatial simulation) library and to analyze their programmability and execution performance. You will form a team of four students and choose an application from graph algorithms, computational geometry, GIS, or any other domains using 2D/3D space, trees, or graphs; parallelize it; and present your work in the class.

2. Choice of Project Topic

Follow the guideline below to choose a parallelizable application.

Guidance:

(1) Don't choose any of the following applications: GA-based TSP, Wave2D, Inverted Indexing, BFS-based Shortest Path Search, Matrix Multiplication, Triangle Counting, Heat2D, Mandelbrot, Game of Life, and Maximum/Minimum/Average Data Search because:
   a. They were given in programs 1-4, (i.e., GA-based TSP, Wave2D, Inverted Indexing, and BFS-based Shortest Path)
   b. Their code was detailed in the lectures, (i.e., Triangle Counting),
   c. They would severely penalize a particular programming tool, (i.e., Wave2D, Heat2D, and Matrix Multiplication kill MapReduce's parallel execution), or
   d. They are too simple, (e.g., Mandelbrot, Game of Life, and Max/Min/Ave).

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

(3) Reasonable size: Your program should include approximately 100+ lines or more with comments and spaces. Some applications may be too large or too complicated to be implemented within 2+ weeks. Therefore, you may simplify your application so as to complete your parallelization work by the due date.

Domains:

The application domains you should consider include (1) graph programs, (2) computational geometry, (3) GIS, and (4) 2D/3D or graph-based data sciences. As far as you follow the topic guidance shown above, you are free to choose any application.

(1) Graph programs:
   a. Graph bridges
   b. Articulation points
   c. Subgraph search, (i.e., much more challenging than Triangle Counting)
   d. Link prediction (This may be a data science application.)
   e. Graph clustering

(2) Computational geometry:
   a. Closest pair of points
   b. Convex hull

c. Range search with a KD tree
d. Point location
e. Largest empty circles

Don't choose Voronoi diagram, Delaunay triangulation, or Euclidean shortest path (too complicated to complete in two weeks!)

(3) GIS

Any GIS applications using 2D/3D space, (quad/octo-)trees, and/or graphs

(4) 2D/3D or graph-based data sciences
   a. Biologically Inspired TSP such as ACO (ant colonial optimization)-based TSP
   b. K means biologically inspired algorithms such as particle colonial optimization (PCO) and grasshopper optimization algorithm (GOA).
   c. K nearest neighbors in 2D or 3D or graph (=KNN graph)

More suggestions:

(1) **MPI** can do anything but its programmability is more demanding than the others as it is based on low-level collective communication.
(2) **MapReduce** can take care of many files but performs slow as it's disk-oriented. It has a big difficulty in handling data structures including graphs.
(3) **Spark** is based on data streaming like MapReduce, but it also enables graph programming as you saw in HW4. It even eases graph programming with GraphX.
(4) **MASS** would work intuitively by dispatching agents into a structured dataset but wouldn't benefit text processing as compared to MapReduce and Spark.

3. Group Work

Please form a team of four students and follow the work procedure given below:

(1) Each group will choose an interesting and challenging application, based on the above guidance and suggestion.
(2) Each student should choose a different parallel-programing tool from mpiJava, MapReduce, Spark, or MASS to parallelize the application with the chosen tool.
(3) The topic, (a.k.a., midpoint) and final project presentations will be given by group. All the group members should participate in the presentations.
(4) The final report may be written and submitted to Canvas jointly as one soft copy. However, each member must submit Lab 5 and a confidential peer evaluation independently to Canvas.

4. MASS 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:
/home/NETID/css534/MASS

5. Team Formation, Choice of Your Application, and Overview of Your Parallelization Strategy

Below is the timeline for forming your final project team, choosing the application for your project, and developing your parallelization strategy in course of your work:

Items	Tasks	Due Date
1	**Team Formation** Please form a team of four students. If it is quite hard to form a team of four, you may form a team of four students, in which case you need to consult the professor in advance. Please sign up the Canvas spreadsheet that the professor will make available soon.	By the end of week 3. (Let's count the first week as week 0)

2	**Application Selection** Please choose the application your team is going to parallelize and add the information of your application to the spreadsheet above.	By the end of week 5.
3	**Parallelization Strategy** Please submit your parallelization strategy to Canvas Collaboration. Include each strategy of mpiJava, MapReduce, Spark, and MASS. This will be the basis for your midpoint presentation in week 9 and final project presentation in week 10	By the end of week 7.

6. Presentations

(1) Midpoint Presentation in Week 9

Each group will explain what application they have chosen, introduce how it will be parallelized with the four parallel-programming tools, and justify any anticipated challenges in their work. Each group is allocated approximate 12 minutes. Your group presentation should include:

Item	Contents	Pages in PPT
1	Application overview	1 page
2	The parallelization strategies/algorithms in mpiJava, MapReduce, Spark, and MASS	4 pages
3	Anticipated challenges	1 page
	Total	6 pages
	Q & A	4 minutes

Week 9 is intended to make sure that your choice and planned parallelization of application is appropriate. Only your active participation, (i.e., both presenter and audience roles) will be graded (2pt), but the performance or contents of your presentation will not be graded. Your group presentation will receive feedbacks from the audience including the instructor, based on which your group may update the spec of your application and/or parallelization strategies.

(2) Final Project Presentation in Week 10

The final project presentation is planned in week 10.

Each group will present their achievements and demonstrate their program execution in approximately 12 minutes. Your group presentation should include:

Item	Contents	Pages in PPT
1	Application overview	1 page
2	The most important code snippet in mpiJava, MapReduce, Spark, and MASS	4 pages
3	An execution snapshot or a demonstration	4 pages or a 4-minute demo
4	Execution performance in a graph or a table In other words, did more computing nodes contribute to faster execution, (i.e., time reduction) and/or spatial scalability, (i.e., ability to handle big data with distributed memory)?	1 page
5	Programmability consideration (summarized in a table)	1 page
	Total	11 pages (12 pages if title included)

The audience will receive an evaluation sheet with which they evaluate each group's presentation, MASS application, and parallelization techniques.

7. What to Turn in

This programming assignment is due at the beginning of class on the due date. Please turn in the following four materials jointly per group to Canvas except "Lab5" and "confidential peer evaluation".

Item	Materials	Pages
1	PPT file uses in the week-10 final presentation	12 pages including your title page
2	Source code (20pts)	4 difference sources, each in mpiJava, MapReduce, Spark, and MASS
3	A term report (45pts, written jointly in your group)	4-7 pages
4	Execution snapshot or graphical results (10pts)	No page limitation
5	Lab 5 (3pts) – Each student!!	MASS Code and execution snapshots
6	Confidential peer evaluation (10pts) – Each student!!	Within 1 page

The report must include:

Item	Contents	Pages
1	A summary of your application (5pts)	1 page
2	Parallelization techniques in mpiJava, MapReduc, Spark, and MASS (10pts)	Up to 2 pages
3	Performance analysis including graphs, tables, and your narrative. (10pts)	Up to 2 pages
4	Programmability analysis (10pts)	Up to 2 pages
	Total	4-7 pages

Each student must turn in an independent and confidential peer evaluation letter, so that the professor will assess if all group members have worked collaboratively. In particular, please make sure who was in charge of parallelization with which library as well as completing which portion of the group report.

8. Evaluation

Criteria	**Grade**
Documentation A summary of your application (1 page)	5pts
Overview of Parallelization Strategy Amount of information in 2 pages: 2pts mpiJava: 2pts MapReduce: 2pts Spark: 2pts MASS: 2pts	10pts
Final Presentation: evaluated by all the audience 10pts = very good 9.5pts = good 9pts = fair 8.5pts = poor 8pts = very poor	10pts
Application/Parallelization Techniques: evaluated by all the audience 10pts = very interesting and very challenging 9.5pts = somewhat interesting or challenging 9pts = fair 8.5pts = not interesting and easy	10pts

8pts = guidance not followed	
Coding Evidence **mpiJava** 5pts = correct and clean code with an appropriate amount of comments 4pts = messy code or minor errors 3pts = incorrect or incomplete **MapReduce** 5pts = correct and clean code with an appropriate amount of comments 4pts = messy code or minor errors 3pts = incorrect or incomplete **Spark** 5pts = correct and clean code with an appropriate amount of comments 4pts = messy code or minor errors 3pts = incorrect or incomplete **MASS** 5pts = correct and clean code with an appropriate amount of comments 4pts = messy code or minor errors 3pts = incorrect or incomplete	20pts
Execution Snapshots or Demo with Graphical Results 10pts = correct execution snapshots 8pts = wrong or incomplete execution snapshots 5pts = no execution snapshot	10pts
Performance Analysis in Graphs or Tables as well as your group narrative 10pts = performance improvement with more CPUs (in at least two programming tools) 8pts = little performance improvement (in 3+ programing tools) 6pts = incomplete performance analysis 4pts = no execution performance	10pts
Programmability Analysis 10pts = good programmability analysis 8pts = poor analysis (including just complaints about how much effort you made) 4pts = no programmability analysis	10pts
Lab5: Individual • MASS source code submitted (1.5pts) • An execution snapshot submitted (1.5pts)	3pts
Confidential Peer Evaluation: Individual -10pts = no contribution to the group presentation slides and/or final report -10pts = no collaboration found from letter(s) or no submission	10pts
Your Active Participation in Presentations Presenter role 1pt, Audience role 1pt	2pts
Total Note that this final project takes 17% of your final grade.	100pts