Introduction to Parallel Processing

Project Overview

Professor Amanda Bienz

Project Overview

- Oct 27 : Proposal
- Nov 17 : Progress Report
- Dec 11 @ 9am : Final Report Due
- Dec 4-8 : Final Presentations in Class
  - You will be randomly assigned to provide **constructive** feedback on 3 presentations

Project Overview

- Group project : preferably 3-4 people per project
  - Solo projects are okay (particularly if you are already doing thesis research, relate your project to your thesis)
- Must be related to parallel computing
  - Should use CARC supercomputers. If you have a project idea that would not involve developing and running code on a CARC computer, run it by me before proposal.
  - Can focus on MPI, OpenMP, CUDA/HIP/etc

Proposal

- 1-2 paragraphs
  - Your plan for your project, including :
    - Current problem (reason you want to look at this)
    - Current hypothesis
    - What you plan to accomplish
- **It is okay if your project changes direction after the proposal**

Progress Report

- 1-2 pages
  - What you have done so far
  - What remains to be done
  - **If any of the following are true, make sure to include them in your progress report:**
    - If your project has changed, describe why and make sure to describe your updated project in detail
    - If you have run into any significant issues, what are they and how do you plan to overcome them
    - If there is anything you would like my input/advice on

Final Presentation

- Last week of class (distributed across M/W/F class period). Will be randomly assigned.
- Approximately 5 minute presentations per group (may slightly change based on number of groups)
- Slides covering :
  - The problem you looked at (and why)
  - Your original hypothesis
  - Your results
  - Anything else of importance (if you needed to drastically change projects midway through, include a slide describing why)
- You will be randomly assigned to provide **constructive** feedback on 3 presentations

Final Report

- Final reports due last day of class (grace period of the weekend)
- A few pages, don’t make too long
- Make sure to include related work, lessons learned (if you changed anything from original project), what you did, results
- **Include citations!** No need to cite anything from this course, but if you use any outside resources, cite them.

Plagarism

- Make sure to be clear if something is not your work
- If you didn't create a figure, cite it
- If you describe and algorithm that isn't yours, cite it
- Do not present others work as your own

Project Ideas

- Collective algorithms
  - All-to-all(v) algorithms, everyone sends to everyone. Many different ways to do this (send to one at a time, send to all at once, somewhere in between) https://arxiv.org/pdf/2306.16589.pdf
    - Analysis of how many processes you should send to at once (will vary with process count)
    - How does load balance affect these algorithms (have each process sleep varying amounts of time before calling algorithm)

Project Ideas

- Collective algorithms
  - Recursive doubling vs recursive halving
  - Alltoall vs alltoallv
  - Analysis of when you should use different algorithms
    - Accurately model when binomial tree broadcast becomes more expensive than scatter-allgather
    - Analyze how process count affects this crossover point
    - Show performance results for how well your model predicts

Project Ideas

- Communication aggregation (good for smaller messages)
  - Combine messages going in the same direction (e.g. all going from node n to node m) into one single message.
  - https://arxiv.org/pdf/1910.09650.pdf

Project Ideas

- Partitioned communication
  - Many OpenMP threads per process, each sends a portion of a message
  - Can receive part of a message before whole message is ready, and can start computing on that data (asynchrony)

Project Ideas

- Compression
  - Can compress data before communicating. If we communicate data many times (e.g. broadcast, allgather) compression overhead is outweighed by communication improvements
  - Analysis how compressible your data must be for this to pay off, how many processes do you need for this to pay off(e.g. how many times are you sending this same data)

Project Ideas

- Sparse vs Dense Matrix Operations
  - How sparse does a matrix need to be for sparse parallel operations to be cheaper than dense
  - Can provide code that reads in sparse matrices and does standard parallel communication, you can compare this to converting the same matrix into dense format (double* matrix) and performing cannon's algorithm or some related dense operation.

Project Ideas

- Parallel sorting algorithms
  - How do you sort data in parallel? There are some great existing algorithms, you can analyze the tradeoffs between these (when should you use one over another)
    - Likely depends on how sorted your data is at beginning, along with data size, process count, etc

Project Ideas

- Two-sided vs one-sided communication
  - One-sided communication will be covered week after exam, often cheaper than standard Send/Recvs
  - Replace standard communication with one-sided
    - Tradeoffs between different one-sided optimizations
    - Analysis on when you should use one-sided vs standard
    - Sparse matrix operations are a good fit for one-sided communication