Parallel Programming and Computing: Group Project

Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180-3590

March 11, 2024

HARD DEADLINE: 11:59 p.m., Tuesday, April 23rd, 2024

1 Description

The central key requirement of your group project is that it **must involve** the design, development in software and experimentation of a massively parallel system. **A key factor is that your parallel system must involved a high degree of interaction and not be considered "embarrassingly parallel" including parallel I/O**. This means that projects involving "wide-area", "campus-area" or other asynchronous, master-slave approaches are not a viable project for this course.

Note, that MPI parallel I/O MUST be included in any project to make-up for not having an Assignment 5. The project assignment will count 40% of the total course grade.

The end goal is that you are making progress towards a result that is of high enough quality to be published. Beyond that, you are free to explore any particular "non-embarrassingly" parallel application that is relevant to your research. More specifically, use the ACM Master Conference Format which can be found at the following url: `https://www.acm.org/publications/proceedings-template`. Both MS Word and LaTeX example formats are available.

Examples of a project might be:

- Re-implement a current serial algorithm/research code using MPI and CUDA effectively on the AiMOS supercomputer with parallel MPI I/O for input/output and/or checkpointing of data.
- Parallelization of components in an existing HPC code using CUDA threads, MPI and MPI I/O for input/output and/or checkpointing of data.
- Design a performance benchmark that tests how CUDA with different MPI ranks with MPI parallel I/O perform together on the AiMOS supercomputer system. What are the overheads as you change the mix of MPI ranks and threads and how are reading and writing data rates to the parallel I/O system effected ? Here, you'll want to consider a number of point-to-point communications and collective calls.
- **A More Specific Example:** Construct a new PARALLEL discrete-event model using a parallel simulator (e.g., ROSS) or re-purpose your Game-of-Life code. For example, create a

human "agent" model of the world (8 billion people) being infected by a pandemic virus (e.g., COVID-19). *See the ROSS GitHub Repo URL shown in the PROJECT directory of Course Materials.*

- Extend a parallel CODES network model and run / benchmark in parallel. *See the CODES GitHub Repo URL shown in the PROJECT directory of Course Materials.*
- Convert a simple event-driven network model to run on a single GPU or multiple GPUs using a cycle-time/fixed time-stepped approach and examine the event rate. Compare and contract event-driven vs. fixed time-stepped approach.
- Extend ideas or concepts discussed in class papers. For example, redo parts of the CPU vs. GPU performance (see Intel paper: `https://dl.acm.org/doi/10.1145/1815961.1816021`) using AiMOS POWER9 processors vs. the NVIDIA V100 GPUs.
- See the proceedings of the 2023 Supercomputing Conference for additional ideas, `https://dl.acm.org/doi/proceedings/10.1145/3581784`.
- Final note: if your project does not involve MPI or CUDA at some level you are probably on the wrong track for this project and should speak with the Prof. Carothers. **Also, you should be careful to design a project that your team can complete by the deadline!!**.

The key components of your project submission paper:

1. First and foremost, your project MUST be about High-Performance Computing with CUDA, MPI AND MPI I/O. So, this excludes any sorts of Cloud Computing projects or embarrassingly parallel "over the web" projects. While those are valid types of computation they are not what this course is about. We are concerned about high-performance computing on supercomputing class systems.
2. You need to have some sort of parallel I/O component to your project where you include parallel I/O performance as part of your experimental/performance study (e.g., how does I/O performance change as you increase the number of MPI ranks).
3. You can do a project using only MPI and MPI I/O provided the algorithm is not very amenable to CUDA. Talk with Prof. Carothers first to confirm this is the case.
4. **It is critical you pick something you and your team can complete.**
5. List your team members. Teams can have up to 4 members. For each team member describe their contribution to the overall project. That is, what did you do besides attend meetings?.
6. Create a 150+ word *Abstract* for your project report.
7. Provide an overview/introduction section in your project report.
8. Describe your parallel implementation both code and algorithms used in the project report. In particular, how is CUDA, MPI and MPI I/O used in your project's algorithm/code.

9. We have read (and will continue to read) a number of papers this semester. So, provide a review of related published articles. Do a **Google Scholar** search. In terms of related work, pick key older papers provided they have a very high citation count otherwise pick the newest results possible from key journals and conferences - e.g., look for IEEE, ACM and SIAM conference and journal publications.

10. **Compute Performance Results:** This will include your sequential and parallel results and indicate your overall speedup. *You ***must*** perform a "strong scaling" study where the problem size remains the same and CPU/GPU/node count increases as well as a "weak scaling" study where the problem size grows with the increase CPU/GPU/node count.* Also, understand the performance improvement of just using CPUs via MPI vs. hybrid CPU/GPU with both MPI and CUDA. Also, how much time (see `clock_now` function below) did you spend reading and writing data using MPI I/O. Here, consider using the NVMe drives available on the CCI systems - see: `https://docs.cci.rpi.edu/clusters/ DCS_Supercomputer/#using-nvme-storage`.

11. **Analysis of performance results:** Provide additional information on why your performance turned out they way it did. In particular, you should understand how much communication overhead your program incurred versus doing real computational work. This should be measured and quantified. Use the cycle counters available on the POWER9 and place in `clockcycle.h`. The inline machine code for getting the cycle count (with a resolution of 512 MHz) is below and posted in the group project course materials:

```
// ONLY WORKS ON POWER9/AiMOS

#ifndef CLOCKCYCLE_H
#define CLOCKCYCLE_H

#include <stdint.h>

uint64_t clock_now(void)
{
  unsigned int tbl, tbu0, tbu1;

  do {
    __asm__ __volatile__ ("mftbu %0" : "=r"(tbu0));
    __asm__ __volatile__ ("mftb %0" : "=r"(tbl));
    __asm__ __volatile__ ("mftbu %0" : "=r"(tbu1));
  } while (tbu0 != tbu1);

  return (((uint64_t)tbu0) << 32) | tbl;
}

#endif // CLOCKCYCLE_H
```

12. Summary and future work. Provide a summary of what you did and directions of where you think this project could go in the future. That is what problems did you not have time to solve?

13. **You can expect your paper/project write-up to be a similar length regular conference paper with the following provisos: For undergraduate teams your report should be at least 6 pages double column, single spacing, 10 point font with performance graphs and references. For graduate teams, your report should be at least 9 pages double column, single spacing, 10 point font with more performance graphs and more references. As paper format examples, use the papers we discussed in class as well as Supercomputing Proceedings.**

14. **The minimum number of references is 10**.

15. **Submit a final copy of your report in PDF format as well as all source code on Submitty.cs.rpi.edu.**

16. EMAIL Prof. Carothers early if have questions or want any sort of guidance about your project. Waiting until a few days before the deadline to ask how or what you should is not acceptable.

17. The deadline (11:59 p.m., Tuesday, April 23th, 2024) is hard because it is the last day of class. So, please plan your use of the supercomputer accordingly, make use of the new "interactive" queue for debug jobs and expect the time for multi-node jobs to take potentially hours to get through the system.