G22.3033-004: Architecture and Programming of Parallel Computers
Handout #2: Course Project

This handout describes the guidelines for the course project. Project completion is mandatory, contributing 40% to your semester grade. All projects will be done in groups of 2-3 students.

Background

The primary objective of the project is to explore in detail one of the areas or issues touched on by the course. A secondary objective is to become aware of, and develop skills that are required for the execution of any project — building a team, planning, collecting tools, designing experiments, analyzing data, deciding what the message is, and building an argument to communicate the message.

The project can address any area of parallel computing, including applications, system software, resource management algorithms, compilers, tools, and parallel architecture. As described below, I will work with you to ensure that the project topic is germane to parallel computing, and that it is doable in the time allocated for it this semester.

I do expect you to do a significant project. A good group project should involve each participant putting in approximately the same amount of work as required for doing all of the programming assignments combined.

Groups

All projects should be done in groups of 2-3 students. Although forming groups is your responsibility, I would like to suggest that you form groups based upon complementarity of skills and backgrounds, rather than proximity of offices! Except in extraordinary circumstances, all group members will receive the same project grade, so choose your group members carefully.

Proposals

To ensure that you have selected a reasonable topic, are aware of important prior work, and to encourage you to start early, a 2-page project proposal must be submitted by October 10th. This proposal must contain:

• the composition of the group,
• a one page summary that describes the topic and explains why it is both important and relevant,
• a one page outline of the specific project goals (i.e., a list of bullets), describing the approach, the tools, and expected outcomes of the project,
• a list of references, showing that you are familiar with prior work in the area.

I will read these proposals, approve, or modify them, and return them by October 17th.

Reports

To prevent the “wait until the last minute” approach, I will use a two-phase grading policy for the projects.

Part one of the project, constituting 40% of the project grade (16% of the overall grade), will combine the one page summary of the topic and its importance (submitted as part of the proposal) with a description of previous work in the area and an outline of the project’s essential approach to the problem. This part of the report should be no longer than 5 pages, and is due November 28.
Part two of the project will combine part one with a complete description of the approach, a presentation of the results obtained and the conclusions. This part of the report should be no longer than 15 pages, making the complete project report at most 21 pages (1+5+15). This final part of the report is due December 12.

The different components of the report together should take the form of a research paper which communicates in turn, the motivation for the problem being solved, previous work, your approach, experiments, results, and conclusions. The quality of writing does matter! To ensure that project members understand the grading standard, I will accept draft papers and offer comments prior to the due dates.

Topic Suggestions

While the project can be on any relevant topic, here are some samples to provide some guidance about what I consider an appropriate topic. Feel free to talk to me to get further details. Don’t be discouraged if you feel you know little about the topic at this point; that’s the point of the project.

- Parallelize a non-trivial sequential application using any combination of programming models and architectures discussed in the course. Note that we are talking about parallelizing a complete application here, rather than the application kernels that will be discussed in class and you will program in your assignments. Good sources of such applications are various members of the CS and Math faculty: Marsha Berger, Mike Shelley, Leslie Greengard, Bud Mishra, Dennis Zorin, Ken Perlin, and Dan Melamed to name a few. The project would identify the sources of parallelism, orchestrate the parallel tasks, and map these tasks to the parallel machine.
- Recently, the dominant use of scalable parallel systems has shifted from traditional scientific computing to applications such as parallel database systems, decision support, and web serving. Port a system (you’ll need to get the source) for one of these systems to a scalable parallel computer. Ideally, the application would require the efficient support for orchestration present in large-scale parallel machines. Explore the performance: what are the critical aspects, and how well are different architectures suited for such applications?
- Using several regular application kernels, develop a model for how differences in cache organizations (cache line sizes, set associativity, etc.), and coherence protocols is likely to affect overall performance. Verify the model using measurements on small-scale and large-scale shared memory machines. How easy/difficult is it to tune the application to deliver high levels of performance across multiple coherence protocols?
- In the class, you will learn about shared memory, message passing, and a hybrid of the two, remote memory access-based programming models. Using several application kernels, compare and analyze the programmability and performance potential of the underlying primitives in these different models.
- Clusters of commodity workstations have been proposed as an alternative to parallel machines. Using application kernels, compare the performance differences between a large-scale message passing machine, and a workstation cluster. Explore the critical performance differences, and important distinctions that remain.
- Instead of providing a shared name space in hardware, many researchers are advocating the use of software to provide a “shared object name space”. The latter approach has the advantage of allowing application-specific custom protocols that may not be feasible with hardware protocols. Port an existing object shared memory system to a scalable parallel platform, and using application kernels, explore the cost and performance advantages of such an approach. To what extent can these advantages be realized automatically by a compiler?