ECSE 420 – Parallel Computing – Fall 2019

Project Guidelines

Form a group of up to 4 students and choose a topic (some suggestions appear below). Your project should involve developing a parallel program either for a shared-memory platform, a distributed memory platform, or a massively parallel SPMD platform such as a GPU. Feel free to propose your own project; we are available to discuss any ideas you have.
To ensure that the topic is well-matched to the course material and that it is of appropriate difficulty, you can contact the instructor and TAs, and discuss the feedback of your submitted abstract.

1 Deliverables

The three deliverables for the project are listed below.
Project abstract (10%): Submit a brief description of the project you will undertake on MyCourses by the prescribed deadline.
Presentation (40%): Presentation in class the during the last two weeks of the course. You are expected to be present for all days of those lectures to present when called and also to help in completing and evaluating your classmates' presentations.
Code and Writeup (50%): The code and a write-up summarizing the results of your project are due by the midnight of the last day of the classes.

2 Project Abstract

Submit a brief project description. The document should include a summary of your project and justification for why the project is appropriate for four people to complete in the remaining half of the semester. Your project description should include:

Problem statement: What problem will you be solving?

Methodology: What approach will you take to solving this problem? What type of platform will your program require, and what programming framework will you use?

The following two parts are not required in the abstract submission, but it is good for you to discuss them as you finalize the proposal.

Testing and/or evaluation: How will you test your program to demonstrate it is accurate and to gauge its performance?

Workload breakdown: Who will be responsible for the different tasks/aspects of the project?

Be as specific as possible in answering all of these questions.

3 Presentation

The last 3-4 lectures have been allocated for project presentations. The presentation order will be determined randomly, so any group can be called at any time during the presentation time slots.
Presentation Length: Each group will be given few minutes (to be determined, but likely in

the neighborhood of 5-8 minutes). The timing will be very strict since we need to fit in all groups. It is expected that all group members will participate in the presentation.
Presentation Content: Your presentation should introduce the problem you addressed, describe the design and approach you took in developing your solution (highlighting how your program exploits parallelism), and should include some measure/indication of results. You may do a demo of your program but this is not required.
Peer Evaluation: When your team is not presenting, you are encouraged to pose relevant, pointed questions, and we might ask for your input for evaluating all other teams.

4 Final Deliverable: Code and Write-Up

Submit the code (or alternative research/design work, if the program is not the main deliverable) for your project along with a brief writeup. Be sure to include instructions required for compiling and running your program. In the writeup, describe the problem you addressed, outline the structure of your code. Include an analysis of your program, indicating things like the type of decomposition technique(s) used, the maximum possible parallelism, how many processes are needed to achieve this parallelism, the critical path and its length, the speedup achieved over a serial program for the same task. Also include a description of how you tested your program for correctness and for performance, and provide the results of these tests.

5 Project Ideas

The subsections below outline a few ideas for potential projects. Feel free to also propose any project idea that you prefer.

5.1 Engineering Task acceleration - Porting Parallel CAD on GPUs

Advances in GPU technology have made GPU architectures and tools valuable resources to be used in areas different from their traditional roles. A representative example is the EDA algorithms that can benefit from GPU acceleration. For instance, in SPICE, a large fraction of the simulation time is spent for the transistor model parameters to be evaluated. During each time step, the same subroutine is executed for each transistor in the design.
In this project, your team would work to port model evaluations on the GPU. Both the CUDA and OpenCL environments could be used. Specific SPICE netlists (e.g SRAM cells, array multipliers etc) could be used for your implementation to be evaluated and for the overall expected speedup to be verified. Furthermore, the scalability of your code could be tested using the powerful Tesla cards.

5.2 Video Processing – Image Transformation, Motion Estimation and Related Tasks

Various image processing tasks can be considered for parallelization. For instance, motion estimation is a key part of all the modern video compression and processing techniques. A great number of motion vectors could be concurrently determined during the transformation between adjacent frames in a video sequence.
In such a project, a serial motion estimation algorithm would be used as the implementation base.
A first step of optimization would be the usage of common loop transformations, for parallel processing capabilities to be revealed. The next step would be the parallelization of these code segments using either multicore programming tools (e.g OpenMP, MPI etc) or GPGPU tools (e.g
CUDA, OpenCL etc). More than one parallel implementations could be explored and an

extensive comparison between different approaches could be presented.

5.3 Architectural Considerations in Multiprocessors on Chip and Networks on Chip (NoC)

Over the past 5 years, Networks on a Chip have grown in importance. Many applications could benefit from both the Software and Hardware resources that a NoC offers. Different architectural considerations could be explored using efficient open-source simulation tools (e.g NoC simulators).

In this type of a project, the execution of a sample parallel application (e.g Gaussian elimination) on a NoC will be inspected. The performance impact of varying different architectural parameters should be evaluated. For example, the impact of routing variability on performance could be explored. Furthermore, the impact of different data decomposition schemes or interprocess communication patterns (e.g point-to-point, collective) on the memory transactions overhead could be inspected.

5.4 NVIDIA Jetson Board

This platform promises to deliver very power efficient parallel computing in a small package.

In this project, the different programming tools (e.g libraries, APIs etc) should be extensively explored. Representative parallel implementations should be used as simple testing scenarios.

5.5 Mobile Computing – High performance Architectures

Parallel and distributed computing advances have led to complex hardware systems that enable high performance applications on commercial electronics (i.e. smartphones, tables, etc.). For instance, Apple's Metal API provides efficient access to the GPU, maximizing the graphics and compute potential of your iOS applications.

Several hardware acceleration mechanisms have been developed and integrated to recent smartphone motherboards. For example, recent Android APIs enable hardware acceleration at the application, activity, or window level.

In this project, existing mobile computing APIs should be explored. Simple iOS- or Android-based applications should be developed that exploit such mechanisms. Performance increase using mobile GPUs for representative functionalities (i.e. scaling, rotating images) should be reported.

Different mobile OS and architectures could be tested. Comparison between newest and older systems (i.e. current versus previous Android OS editions) could be presented.