15-418/618, Spring 2021
Final Project

Key Deadlines
(No grace days or late submissions allowed)

Assigned	Mon., March 29	
Proposal Due	Wed., April 7	11:00 pm
Project Checkpoint	Mon., April 26	11:00 pm
Report Due	Thur., May 13	11:00 pm

1 Overview

Your 15-418/15-618 final project gives you the opportunity to dive deeply into a parallel systems problem of your choosing for the final month of the course. What you attempt for your project is completely up to you. There are only two requirements: (1) We want your project to be challenging (you should learn something relevant to the themes of this class), and (2) we want your project to be fun (you should find yourself eager to keep working on it!).

2 Choosing a Project

There are four basic directions that students have typically pursued with their projects:

1. *Application Driven.* Choose some application of interest to you and that presents significant computational challenges. Select an appropriate platform, identify suitable algorithms and benchmark test cases. Implement and experiment with different possible approaches. Example application domains include machine learning, computer games, graphics, computational photography, mathematical problems, computer vision, and scientific computing.

2. *Exploring System Capabilities.* Choose some class of capabilities that computer systems should provide. Find possible approaches, implement them, and measure and compare their performance on a suitable set of benchmarks. Examples capabilities include synchronization, consistency, transactions, cryptography, and data compression.

3. *Explore Platforms.* Identify a new platform and see how well it can perform problems relative to platforms you have already used. Examples include other flavors of GPUs or other resources. The

platform may also include comparing two different programming languages, e.g., Go vs. C++. You could also explore domain-specific languages, such as Halide, or program libraries, such as TensorFlow or Caffe.

4. *Exploring the Architecture.* Take a part of the hardware system that was taught, then study and model it. Explore different implementation choices, compare on suitable benchmarks, and analyze the results. Examples include coherence or network-on-chip simulations, and performance analysis models.

Some projects combine some combination of these basic directions, e.g., mapping an interesting application onto a novel platform.

A few important things you should consider:

- Application-driven projects can be very interesting, but it's important to remember that this is a course on parallel computing. The application should be amenable to parallelism, but not trivially so. You should not have to spend large amounts of effort just understanding the application and the relevant algorithms. Avoid projects where you would need to write large amounts of code just to implement the application. You should have a source of good benchmarks.
- It's important to scope the project so that it is neither too easy nor too hard. In terms of effort, your project is worth 25% of your total grade for the course. Compare this to Assignments 2 and 3, which were each worth 12% of the course. We therefore expect you to expend an effort comparable to the combined effort required for our two most challenging assignments. Since it is difficult to predict how hard or easy a given programming task will be, it helps to formulate a plan that has multiple goals, with different levels of difficulty, and with clear objectives for each.
- Your project does not need to be original research. It is perfectly acceptable to read some research papers on a new way to solve a problem, implement the ideas in the paper, and measure the performance. You do not need to invent new algorithms.
- It's OK to build on work you are doing or have done for a different course or for a research project, as long as the previous focus was not on parallel computing. (We don't want you to double count your efforts in this course with something you would be doing otherwise).
- It's OK to use existing code written by you or by someone else. However, the more you rely on existing code, the more we expect you to do a very thorough job conducting and evaluating experiments, examining alternative approaches, and in general being very thorough in your evaluation.
- Although you will see past projects involving solving combinatorial problems, such as Sudoku and word search, many of these problems suffer from 1) being too simple to be worth the effort of parallelizing, and/or 2) being trivial to parallelize. Try to be more imaginative!
- You will also see past projects that are focused more on distributed systems issues than on performance-oriented parallel computing. These topics have been covered in other versions of the course, but not in ours. For example frameworks designed to support scalable and robust services on a distributed platform, such as Hadoop and Spark, are not appropriate.

Below is a list of project pages from past terms. Some of the links are no longer valid, but even the titles might provide you with ideas:

- Spring 2012
- Spring 2013
- Spring 2014
- Spring 2015
- Spring 2016
- Fall 2016
- Spring 2017
- Fall 2017
- Spring 2018
- Fall 2018
- Spring 2019
- Fall 2019
- Fall 2020

Of course, you cannot simply replicate previous projects. We expect you to challenge yourselves!

A few suggestions for ideas to explore are presented in Appendix A

3 Resources

You already have experience with two important computing platforms:

- Multi-core servers, such as the GHC and Latedays machines
- GPUs, such as those connected to the GHC machines

Others are readily available:

- The ARM processors running in Apple and Android cellphones and tablets, as well as low-cost platforms such as Raspberry Pi's (these all include GPUs).

4 Important Deadlines

As indicated above, we will impose a series of deadlines to allow us to provide suitable feedback and to help you manage your time. All of the reports described below should be submitted as PDF files to Gradescope.

4.1 Project Proposal

Writing your ideas down forces you to organize your thoughts about your project. It gives the course staff the ability to verify your plans are of the right scope given our expectations, and it also gives us the ability to offer suggestions and help.

Your project proposal should include the following:

Title Please provide the title of your project, followed by the names of all team members. Teams may include up to two students.

Summary Summarize your project in no more than 2–3 sentences. Describe what you plan to do and what parallel systems you will be working with. Example one-liners include (you should add a bit more detail):

- We are going to implement an optimized Smoothed Particle Hydrodynamics fluid solver on the NVIDIA GPUs in the lab.
- We are going compare large-scale sorting algorithms written in Go to optimized ones written in C++.
- We are going to create optimized implementations of sparse-matrix multiplication on both GPU and multi-core CPU platforms, and perform a detailed analysis of both systems' performance characteristics.
- We are going to back-engineer the unpublished machine specifications of the GPU in the tablet my partner just purchased.
- We are going to implement two possible algorithms for a real-time computer vision application on a mobile device and measure their energy consumption in the lab.

Background If your project involves accelerating a compute-intensive application, describe the application or piece of the application you are going to implement in more detail. This description need only be a few paragraphs. It might be helpful to include a block diagram or pseudocode of the basic idea. An important detail is what aspects of the problem might benefit from parallelism? and why?

The Challenge Describe why the problem is challenging. What aspects of the problem might make it difficult to parallelize? In other words, what to you hope to learn by doing the project?

- Describe the workload: what are the dependencies, what are its memory access characteristics? (is there locality? is there a high communication to computation ratio?), is there divergent execution?
- Describe constraints: What are the properties of the system that make mapping the workload to it challenging?

Resources Describe the resources (type of computers, starter code, etc.) you will use. What code base will you start from? Are you starting from scratch or using an existing piece of code? Is there a book or paper that you are using as a reference (if so, provide a citation)? Are there any other resources you need, but haven't figured out how to obtain yet? Could you benefit from access to any special machines?

Goals and Deliverables Describe the deliverables or goals of your project. This is by far the most important section of the proposal:

- Separate your goals into what you *plan to achieve* (what you believe you must get done to have a successful project and get the grade you expect) and an extra goal or two that you *hope to achieve* if the project goes really well and you get ahead of schedule. It may not be possible to state precise performance goals at this time, but we encourage you be as precise as possible. If you do state a goal, give some justification of why you think you can achieve it. (e.g., I hope to speed up my starter code $10\times$, because if I did it would run in real time.)
- If your project relies more on analysis than implementation, what are you hoping to learn about the workload or system being studied? What question(s) do you plan to answer in your analysis?
- Systems project proposals should describe what the system will be capable of and what performance is hoped to be achieved.
- *In General*: Imagine that you weren't given grading script on Assignments 2, 3, or 4. Imagine you did the entire assignment, made it as fast as you could, and then turned it in. You wouldn't have any idea if you'd done a good job!!! That's the situation you are in for the final project. And that's the situation the staff is in when grading your final project. As part of your project plan, and *one of the first things you should do when you get started working* is to implement the test harnesses and/or baseline "reference" implementations for your project. Then, for the rest of your project you always have the ability to run your optimized code and obtain a comparison.
- Describe how you will present your report at the final poster session. Will you mostly present charts and graphs? Will you be able to give a demonstration?

Platform Choice Describe why the platform (computer and/or language) you have chosen is a good one for your needs. Why does it make sense to use this parallel system for the workload you have chosen?

Schedule Produce a schedule for your project. Your schedule should have at least one item to do per week. List what you plan to get done each week from now until the end of the term in order to meet your project goals. Keep in mind that due to other classes, you'll have more time to work some weeks than others (work that into the schedule). You will need to re-evaluate your progress at the end of each week and update this schedule accordingly. In your schedule we encourage you to be as precise as possible. It's often helpful to work backward in time from your deliverables and goals, writing down all the little things you'll need to do (establish the dependencies!). Factor in what you plan to have accomplished for each of the two project checkpoints.

4.2 Project Checkpoints

As the schedule indicates, you have a checkpoint to help you keep on track and to help the course staff be able to track your progress. For the checkpoint, you should submit a brief (at most 5 pages) report.

The checkpoint gives you a deadline at an intermediate point in the project. The following are suggestions for information to include in your checkpoint report. Your goal in the report is to assure the course staff (and yourself) that your project is proceeding as you said it would in your proposal. If it is not, your report should emphasize what has been causing you problems, and provide an adjusted schedule and adjusted goals. As projects differ, not all items in the list below are relevant to all projects.

- Report on how things are progressing relative to the schedule you provided in your project proposal. What work have you completed so far, what adjustments do you need to make to the original schedule and project goals? By the checkpoint, you should have a good understanding of what is required to complete your project. By this point, you should be able to provide a very detailed schedule for the coming weeks. Break time down into half-week increments. Each increment should have at least one task, and for each task put a team member's name on it.
- Include one to two paragraphs, summarizing the work that you have completed so far.
- Describe how you are doing with respect to the goals and deliverables stated in your proposal. Do you still believe you will be able to produce all your deliverables? If not, why? What about the "nice to haves"? In your checkpoint reports, we want a new list of goals that you plan to achieve for the project.
- What do you plan to present at the end of the course? Charts and graphs? Demonstrations?
- Do you have preliminary results at this time? If so, it would be great to include them in your checkpoint write-up.
- List the issues that concern you the most. Are there any remaining unknowns (things you simply don't know how to solve, or resource you don't know how to get) or is it just a matter of coding and doing the work?
- (Optionally) schedule a meeting with the course staff to discuss progress

4.3 Project Report

Your proposal should include the following basic sections. Not all the sub-bullets apply to all projects, but they are given as examples/suggestions of issues to address. You are also encouraged to provide more detail if you wish. Note that some of the information in your final report can be pulled directly from your proposal if it is still accurate.

Summary A short (no more than a paragraph) project summary and some key results achieved. Examples:

- Example: We implemented smooth particle hydrodynamics in CUDA on the GPU and in ISPC on the CPU and compared the performance of the two implementations.
- Example: We parallelized a chess bot. Our 64-core implementation on AWS achieves a 40x speedup and won several games on an internet chess server.

- Example: We accelerated image processing operations using the GPU. Given the speed of our implementation, we demonstrate that a brute-force approach to breaking CAPTCHAS is effective.

Background Describe the algorithm, application, or system you parallelized in computer science terms. Figure(s) would be really useful here.

- What are the key data structures?
- What are the key operations on these data structures?
- What are the algorithm's inputs and outputs?
- What is the part that computationally expensive and could benefit from parallelization?
- Break down the workload. Where are the dependencies in the program? How much parallelism is there? Is it data-parallel? Where is the locality? Is it amenable to SIMD execution?

Approach Tell us how your implementation works. Your description should be sufficiently detailed to provide the course staff a basic understanding of your approach. Again, it might be very useful to include a figure here illustrating components of the system and/or their mapping to parallel hardware.

- Describe the technologies used. What language/APIs? What machines did you target?
- Describe how you mapped the problem to your target parallel machine(s). **Important**: How do the data structures and operations you described map to machine concepts like cores and threads. (or warps, thread blocks, gangs, etc.)
- Did you change the original serial algorithm to enable better mapping to a parallel machine?
- If your project involved many iterations of evaluation and optimization, please describe this process as well. What did you try that did not work? How did you arrive at your solution? The notes you've been writing throughout your project should be helpful here. Convince us you worked hard to arrive at a good solution.
- If you started with an existing piece of code, please mention it (and where it came from) here.

Results How successful were you at achieving your goals? We expect results sections to differ from project to project, but we expect your evaluation to be very thorough (your project evaluation is a great way to demonstrate you understood topics from this course). Here are a few ideas:

- If your project was optimizing an algorithm, please define how you measured performance. Is it wall-clock time? Speedup? An application specific rate? (e.g., moves per second, images/sec)
- Please also describe your experimental setup. What were the size of the inputs? How were benchmark data generated?
- Provide graphs of speedup or execution time. Please precisely define the configurations being compared. Is your baseline single-threaded CPU code? It is an optimized parallel implementation for a single CPU?
- Recall the importance of problem size. Is it important to report results for different problem sizes for your project? Do different workloads exhibit different execution behavior?

- **Important**: What limited your speedup? Is it a lack of parallelism? (dependencies) Communication or synchronization overhead? Data transfer (memory-bound or bus transfer bound). Poor SIMD utilization due to divergence? As you try and answer these questions, we strongly prefer that you provide data and measurements to support your conclusions. If you are merely speculating, please state this explicitly. Performing a solid analysis of your implementation is a good way to pick up credit even if your optimization efforts did not yield the performance you were hoping for.
- Deeper analysis: Can you break execution time of your algorithm into a number of distinct components. What percentage of time is spent in each region? Where is there room to improve?
- Was your choice of machine target sound? (If you chose a GPU, would a CPU have been a better choice? Or vice versa.)

References Please provide a list of references used in the project.

Division of Work If your project is a team project, please list which part of the work was performed by each partner. Alternatively, you can simply state: "Equal work was performed by both project members."

5 Grading

The project will have a maximum score of 100 points, allocated as follows:

Points	Item
15	Proposal
10	Checkpoint
25	Project idea
25	Project execution
25	Report quality

Each of the items must be submitted by the deadline to receive any credit. The main part of the grade will be for the project itself. The first two items are to make sure that you are making adequate progress and receiving sufficient support in your work. It will be graded on the basis of three different attributes:

Project idea: Was it of the right nature and scope? Was it interesting or clever (not strictly required, but nice to have)?

Project execution: How far did you get? How deeply did you explore and understand your performance results?

Report quality: How well does it provide a suitable background/context? How well does it document the results? What kinds of insights does it provide?

A Project Ideas

Here are a few ideas for project arising from some of our new equipment and some of the lectures and recitations. Keep returning to this list as more ideas are added.

- **Features found in recent model GPUs.** The GPUs in the current generation of GPUs contain intriguing new features:
  - Tensor cores. According to NVIDIA:

    > "Tensor Cores can accelerate large matrix operations, which are at the heart of AI, and perform mixed-precision matrix multiply and accumulate calculations in a single operation. With hundreds of Tensor Cores operating in parallel in one NVIDIA GPU, this enables massive increases in throughput and efficiency."

    Project: Understand what tensor cores are, how to program them, and what peformance gain they give.
  - Ray tracing support. According to NVIDIA:

    > "RT Cores on GeForce RTX GPUs provide dedicated hardware to accelerate BVH [Bounding Volume Hierarchy] traversal and ray / triangle intersection calculations, dramatically accelerating the ray tracing process."

    How does this hardware acceleration simplify writing ray-tracing code, and how much performance gain does it provide?
- **Exploring the grid solver**. The grid solver was used as an example application for much of the course, including for Recitation 5 on MPI programming. There are many aspects that could be further explored, starting with the code developed for the recitation:
  - How would the evaluation using the red-black coloring described in Lecture 5 compare to the Jacobi iterations in Recitation 5?
  - Would it be possible to overlap computation and communication in MPI, by computing the interior cells for each region while exchanging data about the border cells?
  - Is there any way to exploit multiple nodes in the Latedays cluster to outperform running on a single node?
  - What kind of performance can be obtained for the grid solver by mapping it onto a GPU?