NYU
New York University

Graphics Processing Units (GPUs): Architecture and Programming

Projects

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

The project part of the course

- 50% of the total grade
- groups of 1 to 4 students
  - One submission per group
- Presentations in the last two lectures of the course
- You have 2 choices:
  1. pick one of the suggested projects
  2. suggest a modifications to a suggested project
  3. Suggest your own project
- For #2 or #3 you need to discuss it with me first.
  - You will have a week from the time we announce the project to do so.

Milestones

Milestone 1: Sep 24th (Midnight)

Between Now and Sep 24th

- You can discuss with me any suggested ideas you may have about the project.

By midnight of Sep 24th (at the latest)

1. Go to the course Brightspace page.
2. Go to “More Tools”
3. At the bottom of the list click on “All Course Tools”
4. Seach for “Surveys”
5. Click on the survey and submit your selection.

OR

Content → Projects → Project Selection

Warning!!

- If you don’t make the choice for the final group/project by end of Sep 24th

  (-2%) of the total project grade for each late day.
- If you decide for a modified version of the published projects without discussing it with me first, the project topic may be rejected because it is too hard/simple, and you start getting late penalty.
  - So, don’t wait till the last day of the selection period to discuss your suggested idea with me.

Milestone 2: Nov 19th

- **Final submission (report, documentation, code, etc)** → 40% of the total course grade
- No specific length for the report.
- Submit through **Brightspace**: one zip-file that contains:
  - report
  - All source code
  - A small text file, called **readme.txt**, telling us how to compile/execute your work (i.e. sort of quick user manual)
  - VERY IMPORTANT: All projects must work on our CIMS machines and does not use any software/library that does not work or is not present on CIMS. Telling us that it works on your laptop does not count.
- **Report Content:**
  - Abstract
  - Introduction
  - Literature Survey
  - Proposed Idea
  - Experimental Setup
  - Experiments & Analysis
  - Conclusions
  - References

Milestone 3: Dec 3rd and Dec 10th

- **Presentations** → 10% of the course grade
- 10 mins per group
  - 8 mins presentation
  - 2 mins Q&A
- You present only the main points and conclusions. Details are in the report.
- **8 slides only**: (Figures are better than list of bullets whenever possible)
  - Title Slide (including project title and name of all members)
  - 1 slide problem definition & why it is important
  - 1 slide survey
  - 1 slide proposed idea(s)
  - 1 slide experimental setup
  - 2 slides results and analysis
  - 1 slide conclusions (up to three take aways from the project)

Milestone 3: Dec 3rd and Dec 10th

- The presentations will be graded based on:
  - How well you presented
  - Whether you followed the rules (timing, slides, etc)
  - How well you answered the question (s)

Regarding the Final Report

Regarding Final Report: Abstract

- The first thing in your report, but the last thing to write.
- 1 paragraph summarizing your problem and why it is important.
- 1-2 lines hinting on your technique
- 1-2 lines summarizing your finding(s)

Regarding Final Report: Introduction

- Expand the abstract to include why the problem you are solving is important.
- Then, give a general idea about your way of approaching the problem.

Regarding Final Report: Literature Review

- Feel free to organize this section in any way you want:
  - Include any subsections you think you need.
  - Draw any Figures you like.
- But you need to discuss *at least* the following items:
  - What did others do regarding the topic at hand? Note: the worst survey ever is when you say "x has done y; z has done k; …". You must put the work of others in taxonomy. That is, "The solution to this falls in x categories: …. . In the first category x has done …".
  - What are the pros and cons of what they are doing?
  - Why their work is not enough, and your proposed project is needed?
  - Important: Literature must not be about papers solving exactly your same problem. But can include papers about different problems but have similar challenges.

Regarding Final Report: Proposed Idea

- This is a major part of this report.
- Describe in great details the solution to the problem at hand.
  - For example, if you are parallelizing something, you must explain how you did that and justify your choices.
- Don't just explain what you did, but also "why" you did it that way.

Regarding Final Report: Experimental Setup

- State:
  - Tools you used
  - Machines you ran your experiments on (do not say cuda1 for example, but the specs).
  - The benchmarks (if any),
  - etc …
- How do you know you have written a good experimental setup? If another researcher reads this section, the researcher must be able to replicate your experiments without asking you any questions.

Regarding Final Report: Results and Analysis

- **Formatting tip**: using tables, bullets, figures, diagrams, … are better than words in presenting results.
- What is/are the measures of success in your work? That is, what do you need to measure and what are you expecting the results to be in order to say that you succeeded?
- What are the experiments that you did in order to get the measurements you mentioned in the above bullet?
- **Analysis:**
  - **Bad**: "As we can see x is increasing with y" ← You won't get any credits for that!
  - **Much better**: "x increases with y because 1, 2, 3, …. And under the conditions …. x may not increase with y. We learn from that …."

Regarding Final Report: Conclusions

- What are the most important findings of your project?
  - A bullets list is the best here.
- No more than three points.

Regarding Final Report: References

- If you only mention the references I provided you, it means you did not do any extra work.
- Websites like wikipedia, stackoverflow, etc are not references, don't use them.
- Don't use urls unless there is no paper about the topic.
- Example of reference format:
  - Chris Rohlfs and Mohamed Zahran, **Optimal Bandwidth Selection for Kernel Regression Using a Fast Grid Search and a GPU**, in 7th IEEE Workshop Parallel / Distributed Computing and Optimization (PDCO 2017), in conjunction with 31st IEEE International Parallel & Distributed Processing Symposium (IPDPS), May 2017.

Suggested Projects

Project 1:
Parallelize an Application
(YOU must pick the application.)

The application you pick must have several characteristics:

- Must be non-trivial
- Must have a major part (in terms of execution time) that can benefit from a GPU.
- Must be written in C/C++ with CUDA under Linux.
- Can be compiled and executed on our CUDA cluster.
- Must not require any proprietary libraries.

You need to compare against state-of-the-art implementation of this application. So, pick an application where you have access to other state-of-the-art implementation or an application that has not been parallelized for the GPU.

When parallelizing an application, your measure of success is:

- The overall performance of your program is better than the other state-of-the-art. The state-of-the-art can be a multicore version, or another GPU version. You will also compare yourself to the sequential program, that you need to build anyway before implementing the GPU version.
- Scalability: If we move your code to a bigger GPU, do we see performance enhancement? You will have access, on NYU cluster, to GPUs of different strengths. Assuming of course the problem size is big enough to justify a bigger GPU. How does your program behave if we keep the same GPU but give a bigger problem size?

Another measure of success is that the performance of your implementation is the same/comparable to the state-of-the-art but uses less resources like less memory.

Project 2:
Predicting the Performance of a Kernel

- You pick one of the following two points for your project:

A. Can we predict the speedup of a kernel, relative to one thread, on a GPU *without* executing it?

B. Knowing the performance of a kernel on a GPU, can we predict its performance on another GPU with different specifications?

- You can build an analytical model, use machine learning, or you can use any other method as long as you have a good reason why you picked it.

Project 3:

Data Structure Library for GPUs

- The main task is to build a library that implements a data structure for GPU applications.

- You need to pick one structures (e.g. stack, queues, linked list, or even invent one). You may want to look at several GPU applications and decide on a data structure that will be useful for several.

- Implement all operations to be done on that data structure in CUDA. For example, a queue can have: enqueue, dequeue, count, etc.

- The library must be usable by others. So, you need to have the programmer in mind when designing this. For example, for queues, what will be a single element in a queue? A number? a structure? Can you make it generic?

Project 4:
The Path to Multi-GPU systems

- In this project you need to parallelize applications using multi-GPU systems and compare the performance with a single GPU.

- Based on this, you need to analyze the results and build an analytical model that can predict whether it is better to use single GPU or multi-GPU systems given: some characteristics of the kernel and specs of the GPU.

- You need to determine:
  - What are the inputs to that model?
  - What is the model itself?
  - What is the accuracy of the model?

- Test the model using a set of benchmark suite you create. The suite must have programs of different characteristics (e.g. embarrassingly parallel, heaving sharing data among threads in the same block, sharing among threads globally through the memory, using a lot/ or little communication among devices and host, etc.).

Project 5:
Cache Management for GPU

- All modern GPUs are equipped with L2 cache that is outside all SMs.
- One of the important aspects of a cache is to decide which block to discard when a cache set is full.
- This project analyzes access patterns for L2 and suggests a modified L2 cache replacement policy that yelds better performance.
- This project will be done using a simulator (AccelSim), or you write your own.
  - The simulator is written in C++ and you will need to hack it to include your new technique. Therefore, you need to be good in C++.
- The measure of success is the total execution time of the different kernels on the GPU relative to the default configuration.

Project 6:
Tool to suggest grid and block sizes

- **Input:** Information about the device, the kernel, the problem size, and anything you see important.
- **Output:** Number of blocks and number of threads per block.
- **How to approach it?**
  - Design an analytical model after analyzing many kernels with different dimensions.
  - Use some machine learning technique.
- **How to test:**
  - Design a set of benchmark programs with different characteristics.
  - For each program, test with your tool as well as other dimensions.
  - Analyze the results obtained.

Have Fun!