CPE 419/466 - Fall 2015 - Combined Project

Parallel PageRank

Due date: Sunday, November 15, midnight.

Overview

In this assignment you will work **with CSC 419 students** to implement two versions of the parallel PageRank ranking algorithm; one to be run on an NVIDIA GPU, and the other to be run on an Intel Xeon Phi. You will test on a number of datasets.

Assignment Preparation

This is a team programming assignment. The instructors will construct teams with at least one person from each course (419/466). The intent is **not** for 419 students to just inherit and parallelize the 466 students' code. Much more learning happens if the 419 students come out of the project understanding PageRank, and the 466 students know how to parallelize code. It is expected that you will help to teach your team members the material from the class you are in, and to learn from the team members of the other class. Every team member is expected to participate in the design, implementation, testing and optimization of the program(s)! Each team submits only one copy of the assignment deliverables.

Teams

Please note: because the classes run on different days, we are unable to schedule mandated joint labs. It is the responsibility of each team to contact their partners from the other class and to set up the implementation schedule.

Data

There are several datasets available for this assignment. They are broken into two categories: the small datasets and the SNAP: Stanford Large Network Dataset Collection, https://snap.stanford.edu/data/

Small Datasets

Your implementation shall run in adequate time on any of the datasets from the ”small datasets” list provided below. The results of running your implementation of PageRank on these datasets must be put in your report.

1. **STATES**. This dataset contains information about the 48 mainland U.S. states plus the District of Columbia and the borders that these states share. (Alaska and Hawaii are excluded from the dataset since they do not share borders with other states).
2. **NCAA-FOOTBALL**. This dataset contains information about ev- ery single game played by Division I teams in the 2009 NCAA regular football season (before bowls and championships started). A total of 1537 games was played, their results are documented in the dataset.
3. **KARATE**. This dataset describes a small social network consisting of members of a university karate

club.

4. **DOLPHINS**. A social network of a group of dolphins as observed by researchers over a period of time.
5. **LES-MISERABLES**. A graph of co-occurrence of different characters in the chapters of Victor Hugo's novel Les Miserables.
6. **POLITICAL-BLOGS**. A graph of political blogs connected by their citations of each other on the eve of the 2004 Presidential election in the US.

Assignment

You have already implemented a program that takes as input a data file, runs the PageRank analysis on the graph extracted from the input file, and outputs the individual items (football teams, states, etc...) ranked in descending order of their computed PageRank together with the PageRank score and the rank. The program also contains timing functions that measure the following:

1. Read time. The time it takes to read in the data from the input file and build the initial graph data structure.
2. Processing time. The time it takes for the PageRank process of compute the PageRank of each node in the graph.
3. Number of iterations. The total number of iterations it takes for PageRank to converge (this is, unless you are running PageRank on a preset number of iterations).

Your Lab 3 implementation of PageRank will be considered a golden standard when it comes down to the accuracy. That is: ***your parallel implementation MUST return the same results as your Lab 3 implementation[1] on each data set given the same input parameters.***

Method

Some of the advice below concerns specific material covered in CSC 419. Please consult with your CSC 419 partners on it.

Refactoring/Recompilation

Refactor your implementation into C/C++ and recompile the PageRank implementation using both the NVIDIA (nvcc) and Intel (icc) compilers. Use any reporting mechanisms the compilers offer to better understand how the code is being optimized. Compare runtimes of the refactored and recompiled implementations and include differences in your report.

Profiling

The first step you should take after recompiling is to profile your code using both NVIDIA's and Intel's profilers (nvvp and vtune, respectively). Identify the hot spots in the existing implementation. Specifically, identify the top three functions where the most CPU time is spent. Include these in your report.

[1] Unless you discover an actual error in the code, in which case you will need to fix your Lab 4 implementation as well.

Memory Layout Optimization

Before parallelizing the code, make every attempt to exploit memory locality in your implementations (for both architectures). Focus on the techniques we discussed in class: arrays vs. lists, row-major vs. column-major ordering, Structures-of-Arrays versus Arrays-of-Structures, memory alignment, etc. Collect timing information after optimizing memory layout and compare with the version of the code you started with. Include this in your report.

Xeon Phi Implementation

For the Xeon Phi, you may implement either an offload model, or a native MIC executable, your choice. For either option, you will need to determine whether the entire input dataset will fit in the device's memory. Also for either option, plan to utilize threading via OpenMP, and vectorization using pragmas.

GPU Implementation

For the GPU, you may implement your solution using either CUDA or OpenACC, your choice. For either option, plan to implement the optimization strategies we discussed in class: tiling, shared memory, minimize branch diversion, etc.

Memory Movement Optimization

As you've seen, for the accelerators to provide maximum performance, data transfer between the device and host needs to be minimized. Make every effort to put the data you need on the device early, and reuse it as much as possible without a transfer back to the host.

Performance Requirement

At the very least, your parallel implementations must have better performance than the original version of the code you start out with. The goal is to **significantly** improve performance. It is not unreasonable to expect an order of magnitude performance gain using the methodology described above.

Report

The assignment will be graded primarily based on the contents of your report. Your report shall be a word-processed document submitted, preferably, in PDF format, which contains the following information:

1. Front matter. Title, names of team members.
2. Implementation Overview. A few paragraphs describing the parallelization details of your implementations, including the specific items mentioned above.
3. Results. For each dataset, include a subsection which contains the following information:
    a. Any specific information about the settings (if any) used to run PageRank on this dataset.
    b. A concise discussion on any differences in output between your parallel implementations and the original version.
4. Overall performance summary. Provide a short overall summary of the observed timing results. Include the timing results from the original version, the Xeon Phi version, and the GPU version.
5. Appendix. README. Attach your README file with instructions on compiling and running your program as an appendix to your report.

Extra Credit

Two 20% bonuses will be given: one for the fastest Xeon Phi implementation, and one for the fastest GPU implementation. Both will be tested on the SNAP LiveJournal dataset. Both bonuses can be given to the same team if appropriate. Implementations must produce correct output! ***We may also test your implementations on a few other SNAP datasets that exceed the size of SNAP,*** *so we encourage testing on them (find the largest directed graph in SNAP and see what you can do for it!)*

Tips for Success

- Start Early!
- Meet your team members right away, exchange contact info (email, phone, etc.).
- Compare calendars to find times to work together. You will need more than your three hours of lab per week to complete this project.
- Have a kickoff meeting with all members to go over the code. 466 members should discuss the flow of the program and answer questions about implementation decisions. 419 members should make every effort to understand the original implementation.
- Delegate tasks for each team member so that nobody is wondering what to do.
- Set intermediate milestones, and meet them.
- Meet Often!
- Use a collaboration tool (e.g. subversion, git, bitbucket, etc.), but please **don't share publicly**.

Handin Instructions

Hand in using the command:

```
handin clupo 419_pagerank <all source files> report.pdf README
```

One submission per team!

PLEASE NOTE: The full submission for this assignment goes to the CSC 419 handin. The submissions will be shared/graded by both instructors and the teams will receive the same grade for each of the classes.