CS553 Programming Assignment #1

Benchmarking

Instructions:

- ***Due date: 11:59PM on Tuesday, 09/23/14***
- ***Maximum Points: 100%***
- ***Maximum Extra Credit Points: 30%***
- *You should work in teams of 3 for this assignment.*
- *Please post your questions to the Piazza forum.*
- *Only a softcopy submission is required; it must be submitted to "Digital Drop Box" on Blackboard.*
- *For all programming assignments, please submit just the softcopy; please zip all files (report, source code, compilation scripts, and documentation) and submit it to BB.*
- *Name your file as this rule: "PROG#_LASTNAME1_LASTNAME2_LASTNAME3.{zip|tar|pdf}". E.g. "Prog1_Raicu_Li_Wang.tar".*
- *Late submission will be penalized at 10% per day (beyond the 7-day late pass).*

1 Your Assignment

This project aims to teach you how to benchmark different parts of a computer system, from the CPU, GPU, memory, disk, and network.

You can be creative with this project. You are free to use any programming languages (C, C++, Java, etc) and any abstractions such as sockets, threads, events, etc. that might be needed. You are free to use any machines for your development as long as it will work under Linux for your final evaluation. The TAs will compile and test your code in Linux. If your code does not compile, you will get 0 for the code component of your assignment.

In this project, you need to design a benchmarking program that covers the five components listed below:

1. **CPU (2*4 = 8 experiments):**
    a. Measure the processor speed, in terms of floating point operations per second (Giga FLOPS, 10^9 FLOPS) and integer operations per second (Giga IOPS, 10^9 IOPS); ***hint:*** *modern processors can do multiple instructions per cycle, so make sure to give your benchmark good code to allow it to run multiple instructions concurrently*
    b. Measure the processor speed at varying levels of concurrency (1 thread, 2 threads, 4 threads & 8 threads)
    c. Identify the optimal number of threads to get the best performance
    d. Compute the theoretical peak performance of your processor in flops/sec
    e. What efficiency do you achieve compared to the theoretical performance
    f. ***Extra Credit (1%):*** *Repeat each experiment 3 times and report the average and standard deviation for all evaluations.*
    g. *Extra Credit (5%): Run the Linpack benchmark (http://en.wikipedia.org/wiki/LINPACK) and report the best performance achieved; make sure to run Linpack across all cores; what efficiency do you achieve compared to the theoretical performance?*

2. **GPU (1*2+2*3 = 8 experiments):**
    a. Measure the GPU speed, in terms of floating point operations per second (Giga FLOPS, 10^9 FLOPS) and integer operations per second (Giga IOPS, 10^9 IOPS); ***Hint:*** *You will have to use either CUDA or OpenCL to do this assignment; timing might also be tricky, compared to measuring timing on the host system*
    b. Measure the processor speed with full concurrency (mapping 1 thread per core) ➔ 2 experiments
    c. Compare the measured GPU speed with the theoretical compute speed, and explain your results
    d. Measure the read and write memory bandwidth of the GPU memory with different size messages (1B, 1KB, 1MB)
    e. Compare the measured memory bandwidth with the theoretical memory bandwidth, and explain your results
    f. Make sure to make your GPU code generic, and not hard coded to match your specific GPU; the TAs will run your code on a GPU that might be different than your GPU, and your code must still be correct
    g. ***Extra Credit (1%):*** *Repeat each experiment 3 times and report the average and standard deviation for all evaluations.*
    h. ***Extra Credit (5%):*** *Run the Linpack benchmark (http://en.wikipedia.org/wiki/LINPACK) and report the best performance achieved; make sure to run Linpack across all cores; what efficiency do you achieve compared to the theoretical performance?*
3. **Memory (1*2*3*2 = 12 experiments):**
    a. Measure the memory speed of your host; ***hint:*** *you are unlikely going to be able to do this benchmark in Java, while C/C++ is a natural language to implement this benchmark*
    b. Your parameter space should include read+write operations (e.g. memcpy), sequential access, random access, varying block sizes (1B, 1KB, 1MB), and varying the concurrency (1 thread & 2 threads)
    c. The metrics you should be measuring are throughput (Megabytes per second, MB/sec) and latency (milliseconds, ms)
    d. Vary the number of threads and find the optimal number of concurrency to get the best performance
    e. Compute the theoretical memory bandwidth of your memory, based on the type of memory and the speed
    f. ***Extra Credit (1%):*** *Repeat each experiment 3 times and report the average and standard deviation for all evaluations.*
    g. ***Extra Credit (5%):*** *Run the Stream benchmark (http://www.cs.virginia.edu/stream/) and report the best performance achieved; what efficiency do you achieve compared to the theoretical performance?*
4. **Disk (2*2*3*3 = 36 experiments):**
    a. Measure the disk speed; ***Hint:*** *there are multiple ways to read and write to disk, explore the different APIs, and pick the fastest one out of all them*
    b. Your parameter space should include read operations, write operations, sequential access, random access, varying block sizes (1B, 1KB, 1MB), and varying the concurrency (1 thread, 2 threads, 4 threads)
    c. The metrics you should be measuring are throughput (MB/sec) and latency (ms)
    d. Identify the optimal number of concurrency to get the best performance; explain your findings, putting in perspective the hardware you are testing
    e. ***Extra Credit (1%):*** *Repeat each experiment 3 times and report the average and standard deviation for all evaluations.*
    f. ***Extra Credit (5%):*** *Run the IOZone benchmark (http://www.iozone.org/) and report the best performance achieved; what efficiency do you achieve compared to the theoretical performance? Hint: The theoretical performance is generally advertised by the manufacturer.*

5. **Network (1*2*3*2 = 12 experiments):**
    a. Measure the network speed over the loopback interface card (between 2 processes on the same node)
    b. Your parameter space should include the TCP protocol stack, UDP, varying packet/buffer size (1B, 1KB, 64KB), and varying the concurrency (1 thread & 2 threads)
    c. The metrics you should be measuring are throughput (Megabits per second, Mb/sec) and latency (ms)
    d. ***Extra Credit (1%):*** *Repeat each experiment 3 times and report the average and standard deviation for all evaluations.*
    e. ***Extra Credit (5%):*** *Run the IPerf benchmark (http://en.wikipedia.org/wiki/Iperf) and report the best performance achieved; what efficiency do you achieve compared to the theoretical memory performance?*

Other requirements:

- You must write all benchmarks from scratch. You can use well known benchmarking software to verify your results, but you must implement your own benchmarks. Do not use code you find online, as you will get 0 credit for this assignment. If you have taken CS451 which had a simpler variation of this assignment, you are welcome to start with your codebase from CS451. Please carefully read the requirements of this assignment, and the benchmarks that must be done, as many of them have changed.
- All of the benchmarks will have to evaluate concurrency performance; concurrency can be achieved using threads. Be aware of the thread synchronizing issues to avoid inconsistency or deadlock in your system.
- All of these benchmarks could be done on a single machine.
- Experiments should be done in such a way that they take multiple seconds to minutes to run, in order to amortize any startup costs of the experiments; that means that for some of the operations that are really small (e.g. 1B), you might need to do many thousands or even millions of them to run long enough to amortize the costs of the benchmark overheads.
- Not all timing functions have the same accuracy; you must find one that has at least 1ms accuracy or better, assuming you are running the benchmarks for at least seconds at a time.
- Since there are many experiments to run, find ways (e.g. scripts) to automate the performance evaluation.
- Make sure your machine is idle when running the benchmarks as it would help to improve reliability and consistency of the results. For the best reliability in your results, repeat each experiment 3 times and report the average and standard deviation.
- If you do not have a GPU to test your benchmark on, get in touch with the TAs and they will give you access to nodes with GPUs in the Jarvis cluster.
- No GUIs are required. Simple command line interfaces are fine.

2 What you will submit

When you have finished implementing the complete assignment as described above, you should submit your solution to 'digital drop box' on blackboard. Each program must work correctly and be detailed in-line documented. You should hand in:

1. **Design Doc (10%):** A separate (typed) design document (named prog1-report.pdf) of approximately 1-3 pages describing the overall program design, and design tradeoffs considered and made. Also describe possible improvements and extensions to your program (and sketch how they might be made).
2. **Manual (10%):** A detailed manual describing how the program works. The manual should be able to instruct users other than the developer to run the program step by step. The manual should contain example commands to invoke each of the five benchmarks. This should be included as readme.txt in the source code folder.
3. **Source code (30%):** All of the source code; in order to get full credit for the source code, your code must have in-line documents, must compile, and must be able to run the sample benchmarks from #2 above. You must have a makefile for easy compilation. Please specify which student contributed to what code.
4. **Performance (50%):** Since this is an assignment aimed at teaching you about benchmarking, this is one of the most important part; you must evaluate the five benchmarks with the entire parameters space

mentioned in Section 1, and put as a sub-section to your design document mentioned in (1) above. You must produce graphs to showcase the results. Please combine data and plot on the same graph wherever possible, for either compactness reasons, or comparison reasons. Don't forget to plot the average and standard deviation, as opposed to just a simple value (for up to 5% extra credit). Also, you need to explain each graph's results in words. Hint: graphs with no axis labels, legends, well defined units, and lines that all look the same, are likely very hard to read and understand graphs. You will be penalized if your graphs are not clear to understand. Please specify which student contributed to what benchmark experiments.

Please put all of the above into one .zip or .tar file, and upload it to 'digital drop box' on blackboard'. The name of .zip or .tar should follow this format: *PROG#_LastName1_LastName2_LastName3.{zip|tar}.* Please do NOT email your files to the professor and TA!! You do not have to submit any hard copy for this assignment, just a soft copy via Black Board.

Grades for late programs will be lowered 10% per day late beyond the 7-day late pass.