CS553 Homework #4

Benchmarking Memory

Instructions:

- *Assigned date: Monday March 11th, 2019*
- *Due date: 11:59PM on Monday March 25th, 2019*
- *Maximum Points: 100%*
- *This programming assignment must be done individually*
- *Please post your questions to the Piazza forum*
- *Only a softcopy submission is required; it will automatically be collected through GIT after the deadline; email confirmation will be sent to your HAWK email address*
- *Late submission will be penalized at 10% per day; an email to cs553-s19-group@iit.edu with the subject "CS553: late homework submission" must be sent*

1 Your Assignment

This project aims to teach you how to benchmark memory. You can be creative with this project. You are free to use any of the following programming languages (C, C++) and abstractions (PThreads) that might be needed. Other programming languages will not be allowed due to the increased complexity in grading; do not write code that relies on complex libraries (e.g. boost), as those will simplify certain parts of your assignments, and will increase complexity in grading.

You can use any Linux system for your development, but you must use the Chameleon testbed [https://www.chameleoncloud.org]; more information about the hardware in this testbed can be found at https://www.chameleoncloud.org/about/hardware-description/, under Standard Cloud Units. Even more details can be found at https://www.chameleoncloud.org/user/discovery/, choose "Compute", then Click the "View" button. You have been created accounts on this virtual cluster for this assignment; if you have not received your accounts information, reach out to the TAs for this information.

In this project, you need to design a benchmarking program that measures memory performance. You will perform strong scaling studies, unless otherwise noted. The TAs will compile (with the help of make) and test your code on the same Chameleon testbed. If your code does not compile and the TAs cannot run your project, you will get 0 for the assignment.

1. Implement: MyRAMBench benchmark; ***hint:*** *make sure to measure the performance of your memory and not your processor caches*
2. Workload: 1GB data; operate over it 100X times with various access patterns (RWS, RWR) and various block sizes (1KB, 1MB, 10MB)
    a. RWS: read+write (e.g. memcpy) with sequential access pattern
    b. RWR: read+write (e.g. memcpy) with random access pattern
3. Concurrency: 1 thread, 2 threads, 4 threads, 8 threads
4. Measure:
    a. throughput, in terms of bytes per second; report data in GB/sec, gigabytes (10^9) per second; these experiments should be conducted over 100GB of data
    b. latency (read+write 1 byte of data), in terms of time per access; report data in us, microseconds; limit experiment to 100 million operations

5. Run the Memory benchmark pmbw (https://panthema.net/2013/pmbw/) with 1, 2, 4, and 8 threads, and measure the memory sub-system performance
6. Compute the theoretical bandwidth and latency of your memory. What efficiency do you achieve compared to the theoretical performance? Compare and contrast your performance to that achieved by pmbw, and to the theoretical 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 other courses where you wrote similar benchmarks, you are welcome to start with your codebase as long as you wrote the code in your prior class.
- All of the benchmarks will have to evaluate concurrency performance; concurrency can be achieved using threads. Use strong scaling in all experiments, unless it is not possible, in which case you need to explain why a strong scaling experiment was not done. Be aware of the thread synchronizing issues to avoid inconsistency or deadlock in your system.
- Your benchmarks can be run on a single machine.
- 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.
- For the best reliability in your results, repeat each experiment 3 times and report the average and standard deviation. This will help you get more stable results that are easier to understand and justify.
- No GUIs are required. Simple command line interfaces are expected.

1. Fill in the table 1 below for Memory Throughput:

Work -load	**Con-currency**	**Block Size**	**MyRAMBench Measured Throughput (GB/sec)**	**pmbw Measured Throughput (GB/sec)**	**Theoretical Throughput (GB/sec)**	**MyRAMBench Efficiency (%)**	**pmbw Efficiency (%)**
RWS	1	1KB					
RWS	1	1MB					
RWS	1	10MB					
RWS	2	1KB					
RWS	2	1MB					
RWS	2	10MB					
RWS	4	1KB					
RWS	4	1MB					
RWS	4	10MB					
RWS	8	1KB					
RWS	8	1MB					
RWS	8	10MB					
RWR	1	1KB					
RWR	1	1MB					
RWR	1	10MB					

RWR	2	1KB					
RWR	2	1MB					
RWR	2	10MB					
RWR	4	1KB					
RWR	4	1MB					
RWR	4	10MB					
RWR	8	1KB					
RWR	8	1MB					
RWR	8	10MB					

2. Fill in the table 2 below for Memory Latency:

Work -load	**Con-currency**	**Block Size**	**MyRAMBench Measured Latency (us)**	**pmbw Measured Latency (us)**	**Theoretical Latency (us)**	**MyRAMBench Efficiency (%)**	**pmbw Efficiency (%)**
RWS	1	1B					
RWS	2	1B					
RWS	4	1B					
RWS	8	1B					
RWR	1	1B					
RWR	2	1B					
RWR	4	1B					
RWR	8	1B					

2 Where you will submit

You will have to submit your solution to a private git repository created for you at git@gitlab.com:cs553-2019/<student email id>.git. A directory structure for all the repositories will be created that looks like this: hw1/, hw2/, ... You will have to firstly clone the repository. Then you will have to add or update your source code, documentation and report. Make sure this assignment's source code can be found in your hw4 folder. Your solution will be collected automatically after the deadline. If you want to submit your homework later, you will have to push your final version to your GIT repository and you will have let the TAs know of it through email cs553-s19-group@iit.edu. There is no need to submit anything on BB for this assignment. If you cannot access your repository contact the TAs. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/

3 What you will submit

When you have finished implementing the complete assignment as described above, you should submit your solution to your private git repository. Each program must work correctly and be detailed in-line documented. You should hand in (points will be given for each of the items below):

1. **Source code:** 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.
2. **Readme:** 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. **Report / Performance:** Must have working code that compiles and runs on the specified cluster to receive credit for report/performance; furthermore, the code must match the performance presented in the report. A separate (typed) design document (named hw4-report.pdf) of approximately 1 page long 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). Since this is an assignment aimed at teaching you about benchmarking, this is one of the most important parts; you must evaluate the benchmark 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 1 table to showcase the results.

Submit code/report through GIT.

Grades for late programs will be lowered 10% per day late.