CS 758
Fall 2019
Homework 2
Due: Friday, November 1st, 2019 11:59 PM

You are also encouraged to talk with classmates in person or on Piazza about any issues you may have encountered. I will also post solutions to this assignment on Canvas. You are allowed two "free" late days for the homework assignments across the semester. Beyond that, the standard late assignment policy applies: you may submit up to 48 hours late with a 15% penalty per 24 hours. Additional extensions may be possible in exceptional circumstances, but you must **notify me (via email) at least one week beforehand**.

Total Points: 50

The breakdown for these points is:

- 3 points: turn-in contains appropriate files in appropriate places
- 7 points: your scripts can run CHTC jobs
- 2 points: your copy of GPGPU-Sim compiles (e.g., we can source the setup_environment file, then type make, and everything compiles successfully)
- 18 points: functional design works correctly
- 20 points: report

Overview

In the first homework assignment, the input sizes for the programs you ran were intentionally made small, to make it easy to complete the simulated programs in a short amount of time. However, these input sizes are not realistic for the kinds of tests you'd need to run to submit a conference paper. Unfortunately, running all of your tests sequentially on a single CPU (which you likely did for HW1) is not practical when each simulation takes multiple hours. This is why architects often turn to batch submission software, such as CHTC, to run many jobs in parallel.

Accordingly, the goal of this assignment is to take your TLB solution from HW1 (optionally optimized further, if you want to try and improve its performance further), and measure how it performs for much larger input sizes and multiple TLB configurations on CHTC. By carrying out this assignment in GPGPU-Sim + CHTC, you will be able to (a) study how your application performance varies as TLB configuration varies and (b) demonstrate how to get your simulator runs to run in the CHTC job submission software. As before, you should be using the **dev** branch of GPGPU-Sim for this assignment – if you use the master branch, it will fail immediately because CUDA 9.1 is not supported on the master branch.

Note that while it is technically possible to complete most of this assignment without CHTC (e.g., by running them all sequentially on a CPU), I've attempted to size the inputs and configurations such that this would take such a long time that this approach would not be practical.

For all experiments, you should again use the provided gpgpusim.config configuration file as the baseline, as well as the provided config_volta_islip.icnt interconnect configuration file (Note: this corresponds to the baseline, correlated, tested files for the Titan V GPU: https://github.com/gpgpu-sim/gpgpu-sim_distribution/tree/dev/configs/tested-cfgs/SM7_TITANV). If you implemented your TLB as directed in HW1, you should be able to each change the configuration option you added to vary the TLB associativity, size, miss latency, etc. If you didn't, I recommend adding this functionality before running your tests for this assignment. As before, any updates you make to the gpgpusim.config will need to be copied to the folder for EACH application you are running (alternatively, I recommend you write your CHTC scripts to do this automatically).

To get your experiments running in CHTC, you should be able to take the in class tutorial (which showed you how to get backprop running), and extend it for the other benchmarks, as well as extending it to run different TLB configurations for all benchmarks including backprop. You may also find these references from the CHTC website useful when writing your scripts:

- http://chtc.cs.wisc.edu/multiple-jobs.shtml
- http://chtc.cs.wisc.edu/multiple-job-dirs.shtml

Deliverables

Your goal is to explore any interactions with the surrounding architecture and report your results quantitatively (using data tables and bar charts) and qualitatively (by explaining the data) as the TLB configuration varies.

Write a report of 3-4 pages including figures. IMPORTANT: in your report, interpret the data you collect — i.e., what does the data tell you? Do NOT just cut and paste the output of the simulator into your report. Submit your report through Canvas. To get ideas for how to interpret data, refer to the results section of the paper readings for the course. You should comment on quantitative results and provide as much insight as you can relevant to getting better performance using stream buffers (what design choices did you need to make? why do you obtain the results you get? what do the results that say about the hardware being modeled and or the software being run? what can be done to improve the benefit of this translation mechanism, etc.).

Report results for Backprop, BC, BFS, HOTSP, NN, NW, and PageRank. Your results should minimally compare the performance without the TLB you added and the various configurations below (See "**Experiments**") – and you should provide results file for each experiment. Make sure you appropriately summarizing the data for all benchmarks across the different hardware configurations you explore (i.e., if you examine multiple different TLB approaches, you should include them in your graphs/tables; similarly if you explored different associativity's, sizes, etc., you should also include that in your graphs/tables). The benchmarks are stored here, although I have also uploaded them to a common folder on the CHTC cluster at `/home/groups/CS758/cs758-fall2019-hw2-benchmarks.tgz` (I recommend you

copy them from this common folder instead of downloading them from the course webpage). Note that I've provided pre-compiled binaries for each benchmark, so you shouldn't have to worry about compiling them. See the README in each file for the command to run. I also included the "baseline" Titan V Volta gpgpusim.config file in each benchmark folder too – you will need to replace this with your config file that includes your TLB information.

Experiments

For this homework assignment, you should experiment with the following TLB sizes and miss latencies (SA == set associative, FA == fully associative):

Config Name	Associativity	L1 TLB Size	L1 TLB Miss Latency	Page Size
"noTLB"	N/A	N/A	N/A	N/A
"faTLB-128entry-20lat"	FA	128 entries	20 cycles	4 KB
"faTLB-128entry-40lat"	FA	128 entries	40 cycles	4 KB
"faTLB-64entry-20lat"	FA	64 entries	20 cycles	4 KB
"8wayTLB-128entry-20lat"	8-way	128 entries	20 cycles	4 KB
"8wayTLB-128entry-40lat"	8-way	128 entries	40 cycles	4 KB
"8wayTLB-64entry-20lat"	8-way	64 entries	20 cycles	4 KB

Note that Config Name here is the same config name you should use for your results files (see below).

As in Homework 1, on a TLB miss, you should assume that the page walker takes *L1 TLB Miss Latency* (additional) cycles to find the correct translation and bring it into the cache. On a hit, the TLB takes 1 cycle. Moreover, you may again ignore TLB shootdowns and translations for constant memory and texture memory.

GPGPU-Sim Output

As in Homework 1, in addition to emitting output about the run to the screen (stdout), GPGPU-Sim also generates a number of temp files (e.g., 'bfs.1.sm_70.ptxas and _app_cuda_version_*). You can ignore these temporary files. It will also create a "checkpoint_files" repo – this is a new feature recently added to GPGPU-Sim, but you will not be using this support in this assignment, so you can also ignore that folder. The only GPGPU-Sim output you should need to do this assignment is what gets emitted to the screen. Most importantly, note that in CHTC you will need to redirect this output to a file for each run – which the in-class tutorial will show you how to do.

Where to Run

For this assignment, you will be running your tests on the *learn.chtc.wisc.edu* CHTC cluster, which contains a large number of CPUs for running tests just like these. I have already worked with the CHTC staff to get accounts created for all of you (see Piazza for details). **Please let me know ASAP if you are not able to log in with this account**!

Note that you do not need a "real" NVIDIA GPU or CUDA installed on any machine in this cluster to run GPGPU-Sim. The tutorial we're doing in class will show you how to include the necessary CUDA libraries in the input files and transfer these to each worker nodes as part of the input sandbox.

What to Hand In

To submit this assignment, zip or tar the following files together and submit them as a **single file named <netID>-hw2.tgz or <netID>-hw2.zip** on Canvas, organized in a tree structure:

- $HANDIN_ROOT_DIR/
  - report.pdf – a short report analyzing and comparing the results you see; see "**Deliverables**" for more details on the report
  - gpgpu_sim/ -- your checkout of the GPGPU-Sim repo, including any changes you made for the assignment (e.g., from HW1 or any improvements you have made subsequent to HW1).
  - results/ – folder for the results output for each of the benchmark-configurations combinations you are to run (see **Deliverables** and **Experiments**).
    - noTLB/
      - backprop-results-noTLB.txt
      - bc-results-noTLB.txt
      - bfs-results-noTLB.txt
      - hotspot-results-noTLB.txt
      - nn-results-noTLB.txt
      - nw-results-noTLB.txt
      - pagerank-results-noTLB.txt
    - faTLB-128entry-20lat/
      - backprop-results-faTLB-128entry-20lat.txt
      - bc-results-faTLB-128entry-20lat.txt
      - bfs-results-faTLB-128entry-20lat.txt
      - hotspot-results-faTLB-128entry-20lat.txt
      - nn-results-faTLB-128entry-20lat.txt
      - nw-results-faTLB-128entry-20lat.txt
      - pagerank-results-faTLB-128entry-20lat.txt
    - faTLB-128entry-40lat/
      - backprop-results-faTLB-128entry-40lat.txt
      - bc-results-faTLB-128entry-40lat.txt
      - bfs-results-faTLB-128entry-40lat.txt
      - hotspot-results-faTLB-128entry-40lat.txt
      - nn-results-faTLB-128entry-40lat.txt
      - nw-results-faTLB-128entry-40lat.txt
      - pagerank-results-faTLB-128entry-40lat.txt
    - faTLB-64entry-20lat/
      - backprop-results-faTLB-64entry-20lat.txt

            - bc-results-faTLB-64entry-20lat.txt
            - bfs-results-faTLB-64entry-20lat.txt
            - hotspot-results-faTLB-64entry-20lat.txt
            - nn-results-faTLB-64entry-20lat.txt
            - nw-results-faTLB-64entry-20lat.txt
            - pagerank-results-faTLB-64entry-20lat.txt
        - 8wayTLB-128entry-20lat/
            - backprop-results-8wayTLB-128entry-20lat.txt
            - bc-results-8wayTLB-128entry-20lat.txt
            - bfs-results-8wayTLB-128entry-20lat.txt
            - hotspot-results-8wayTLB-128entry-20lat.txt
            - nn-results-8wayTLB-128entry-20lat.txt
            - nw-results-8wayTLB-128entry-20lat.txt
            - pagerank-results-8wayTLB-128entry-20lat.txt
        - 8wayTLB-128entry-40lat/
            - backprop-results-8wayTLB-128entry-40lat.txt
            - bc-results-8wayTLB-128entry-40lat.txt
            - bfs-results-8wayTLB-128entry-40lat.txt
            - hotspot-results-8wayTLB-128entry-40lat.txt
            - nn-results-8wayTLB-128entry-40lat.txt
            - nw-results-8wayTLB-128entry-40lat.txt
            - pagerank-results-8wayTLB-128entry-40lat.txt
        - 8wayTLB-64entry-20lat/
            - backprop-results-8wayTLB-64entry-20lat.txt
            - bc-results-8wayTLB-64entry-20lat.txt
            - bfs-results-8wayTLB-64entry-20lat.txt
            - hotspot-results-8wayTLB-64entry-20lat.txt
            - nn-results-8wayTLB-64entry-20lat.txt
            - nw-results-8wayTLB-64entry-20lat.txt
            - pagerank-results-8wayTLB-64entry-20lat.txt
    - scripts/ – folder for the CHTC scripts you used to run your experiments.
        - backprop/
            - backprop_test.sh
            - backprop_test.sub
        - bc/
            - bc_test.sh
            - bc_test.sub
        - bfs/
            - bfs_test.sh
            - bfs_test.sub
        - hotspot/
            - hotspot_test.sh
            - hotspot_test.sub
        - nn/

            - nn_test.sh
            - nn_test.sub
        - nw/
            - nw_test.sh
            - nw_test.sub
        - pagerank/
            - pagerank _test.sh
            - pagerank_test.sub
    - config/
        - gpgpusim.config -- your updated gpgpusim.config file that models the "faTLB-128entry-20lat" TLB configuration above in "**Experiments**"). We will be using this to test your design on additional benchmarks, so please make sure that your simulator and config file work for any benchmark and does not require any local changes.

If you don't have experience with tar, I recommend consulting tutorials such as this one.

Note: Please run 'make clean' before creating your final tar/zip and uploading your submission to Canvas. Many of you had problems uploading your Homework 1 submissions to Canvas because of the large size of the compiled GPGPU-Sim library. Doing this will make it significantly easier to upload to Canvas, and will also make it easier to grade your assignments faster.

Verifying Your Handin

We have also created a script to check that your submission correctly follows the format, located at:

```
/u/s/i/sinclair/public/html/courses/cs758/fall2019/handouts/scripts/hw2/verify_submission_format.py
```

You should run this script before submitting in order to ensure that you don't lose points for incorrectly formatting your submission! To run it, simply use the following command from wherever you tarball or zip is located:

```
python3 /u/s/i/sinclair/public/html/courses/cs758/fall2019/handouts/scripts/hw2/verify_submission_format.py <netID>-hw2.tgz
```

Note that the above path will only be accessible on CSL machines – the CHTC machines are actually part of a different set of resources with their own filesystem. Accordingly, I've also placed the script on the CHTC cluster at:

```
/home/groups/CS758/scripts/
```

To run this script on the CHTC cluster use the following command wherever your tarball or zip is located:

```
python3 /home/groups/CS758/scripts/verify_submission_format.py
<netID>-hw2.tgz
```

References

- CHTC User Guides: http://chtc.cs.wisc.edu/guides.shtml
- Bharath Pichai, Lisa Hsu, and Abhishek Bhattacharjee. 2014. Architectural support for address translation on GPUs: designing memory management units for CPU/GPUs with unified address spaces. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems* (ASPLOS '14). ACM, New York, NY, USA, 743-758. DOI: https://doi.org/10.1145/2541940.2541942
- J. Power, M. D. Hill and D. A. Wood, "Supporting x86-64 address translation for 100s of GPU lanes," *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Orlando, FL, 2014, pp. 568-578. doi: 10.1109/HPCA.2014.6835965
- Douglas Thain, Todd Tannenbaum, and Miron Livny. 1995. Distributed Computing in Practice: The Condor Experience. https://research.cs.wisc.edu/htcondor/doc/condor-practice.pdf