Lab 4: Specify & Experiment w/ Command-Line Parameters in a gem5 Script

EEL 5764 – Fall 2022

Introduction:

The objectives of Lab 4 are to:

- Add parameters to a configuration script to allow varying these parameter values in the command line.
- Perform experimentation by varying parameters using the command line.
- View and explore detailed statistics generated by a gem 5 run, in preparation for the project.

Part A – Adding parameters to the two_level.py script and caches.py

In Part 1, you will finish the tutorial from the gem5.com website that you started in Lab 3 ("Adding cache to the configuration script"), to add the capability of specifying the cache sizes from a command line in a configuration script.

https://www.gem5.org/documentation/learning_gem5/part1/cache_config/

Notes with regards to the tutorial: (I will further explain in class)

- **Finish the "Adding cache to the configuration script" tutorial.**

  In Lab 3, you performed the first two sections named "Creating cache objects" and "Adding caches to the simple config file". You will continue and finish the tutorial by performing the last section, "Adding parameters to your script".

- In the "Adding parameters to your script" section (**by following the tutorial**):
  - You will modify the **two_level.py script** to add statements necessary to add the capability of specifying cache sizes from a command line and passing the cache size values to the classes in caches.py.

    Warning: in the **first block of instructions** to be added (starting with "**import argparse**"), there may be some (straight-forward) syntax errors that you need to debug.
  - In **caches.py**, you will add the necessary constructors (member functions) to the cache classes to receive the cache size values from two_level.py.
  - The new capabilities are tested using the hello world code provided by gem5.org.

Deliverable A (Page 1 of .pdf file): screenshot of the last screen from running hello world.

Part B – Perform experiment with matmult.c and L1-data cache in the new architecture

Using your matmult.c program (from Lab 3), perform an experiment by varying the sizes of the L1-data cache, **using a command line** in the two_level.py configuration script: e.g.,

```
build/X86/gem5.opt configs/tutorial/two_level.py --param1=value1
```

Deliverable B (Page 2 of .pdf file): For the L1-data cache, start with cache size of **1kB** and increase the size (**doubling each time**) until increasing the size **no longer has an effect** on the performance. Keep the L1-instruction cache and L2 cache at the default sizes.

- Put the results in a two-column table (L1ICache.size, # of ticks)
- Create a graph plot showing the results.

Part C – Change the two_level.py and caches.py scripts to allow the specifying of the l1i_assoc parameter from a command line

- Modify the two_level.py script to add statements necessary to add the capability of specifying a **new parameter, l1i_assoc** (set association parameter for the L1-instruction cache) from a command line and passing the value to the L1ICache class in caches.py.
- In caches.py, you will add the necessary constructors (member functions) to the appropriate class to receive the l1i_assoc value from two_level.py.

Deliverable C (Page 3 of .pdf file): screenshot of the last screen from running the following. For matmult, demonstrate the new functionality by setting the following two parameters in the command line.

build/X86/gem5.opt configs/…./two_level.py --l1i_assoc=4 --l1i_size=4kB

Part D – Experiment by varying l1i_assoc from command line

Perform an experiment by varying (from the command line) l1i_assoc = 1, 2 , 4, and 8. Keep all the other parameters at the default values.

Deliverable D (Page 4 of .pdf file): Complete Row 1 from running the 4 cases. Then, using the corresponding stat.txt files, complete the columns for the L1iCache for l1i_assoc=1 and l1i_assoc=2:

Row #	l1i_assoc =	1	2	4	8
1	**Exit tick #** (from last screen)				
2	# number of overall hits (Count)				
3	# number of overall misses (Count)				
4	# number of overall miss ticks (Tick)				
5	# average overall miss latency ((Tick/Count))				
6	# number of overall (read+write) accesses (Count)				
7	# miss rate for overall accesses (Ratio)				
8	# number of replacements (Count)				

After the summary table (still on Page 4 of .pdf file), answer the following questions:

Q1: Based on Row 1, give one sentence describing the performance trend.

Q2: What is the common name of the mapping method for l1i_assoc = 1?

Q3: How was Row 5 calculated? (Your answer should be in the form of Row X op Row Y, where op can be +, -, *, or /.

Q4: How was Row 7 calculated?

Q5: You should be able to see why l1i_assoc=2 performed better than l1i_assoc=1. Give me all the row numbers which support that observation.

Lab 4: Specify & Experiment w/ Command-Line Parameters in a gem5 Script

EEL 5764 – Fall 2022

SUBMISSION INSTRUCTIONS: Turn in **one pdf file** and **one zip file**.

The pdf file should have the following naming convention:

LastNameFirstNameLab4.pdf ex: LamHermanLab4.pdf

The pdf file contains the following:

- Page 1: Deliverable A screenshot of the last screen from running hello code in Part A.
- Page 2: Deliverable B, from the experiment of varying the size of L1-data cache:
  - Table with results
  - graph plot of the results
- Page 3: Deliverable C, screenshot of the last screen of the outputs from setting --l1i_assoc=4 and --l1i_size=4kB
- Page 4: Deliverable D
  - Table in Part D
  - Answers to Q1 – Q5

.

The zip file should have the following naming convention:

LastNameFirstNameLab4.zip ex: LamHermLab4.zip

There should be 2 files in the zip file:

1. Part C caches.py
2. Part C two_level.py