CSE 560M Computer Systems Architecture I

Assignment 3, due Fri., Nov. 22, 2019

In this lab assignment we will explore using the `gem5` Out of Order (OoO) CPU model to look at factors within a single core processor and see how their variation can effect performance. You will be asked to sweep parameters that will change the sizes of a few microarchitecture parameters and explain how and why they affect performance.

1. To vary the parameters in the simulation we will be using simple options class to pass parameter values into the setup options of our CPU. The SimpleOpts class makes it easy to set parameters on your simulation items from the command line. To make things straightforward we have provided a file to add this functionality to your simulation. To start, copy over the files into your own working directory (e.g. `hw3`).

   ```
   cp -r $GEM5/hw3/* .
   ```

   Take a moment to review the files. The `hw3config.py` file is similar to the previous setup files you've created, however here we have added the `hw3opts` to the options parser (`SimpleOpts`) that was used in the previous lab to set parameters for the cache. In this assignment, the caches have already been set up for you. They can be set by using the switches

   `--caches` and `--l2cache`

   The input binary is specified with the `--c` switch and specifying the binary path. For example,

   ```
   --c /absolute/path/to/binary
   ```

   We will also be using an Out of Order (OoO) processor, and this parameter is set by the following switch:

   ```
   --cpu-type="DerivO3CPU"
   ```

   For this lab the options that you will be changing are directly related to the CPU. The `hw3opts.py` file contains a skeleton of the opts parser that will be used to set these parameters without having to change the source code every run.

   You'll be sweeping three parameters in this lab: (1) the number of reorder buffer entries, (2) the number of instruction queue entries, and (3) the number of physical floating point registers. In the `addHW3Opts` function the parser options will be added, which is an instance of the `SimpleOpts` class from the last lab.

   When declaring an option for the parser we will use the function `parser.add_option()` as we did in Assignment 2. Use this to add configurable options for the three parameters above. The table below gives the default options for these parameters:

Parameter	Value
num-rob-entries	192
num-iq-entries	64
num-phys-float-regs	256

The other function declared in `hw3opts.py` is the `set_config()` function which receives a list of CPUs and the options which are used to set values. Under the scope of the for loop already declared, set the values of the parameters that you are changing for the given CPU like so:

```
cpu.parameter = option.parameter_name
```

The three parameters that you have to set in the CPU class are `numROBEntries`, `numIQEntries`, and `numPhysFloatRegs`.

2. Now that the options are set properly we can sweep across parameters and take measurements! Similar to Homework 2 we are going to run `daxpy_arm_big` program (which is `daxpy` with a larger input size) using the `ARM` ISA and save the simulator output files as you vary the parameters of the options you've created. Using a **squared** step size (as opposed to a linear one) to perform a measurement of each permutation of configurations as you sweep: (1) the number of physical floating point registers from 256 to 4096, (2) the number of instruction queue entries from 4 to 256, and (3) the number of reorder buffer entries from 4 to 256. This should result in a total of 245 runs that take much longer than previous simulator runs so it is HIGHLY suggested that you create a script to automate the process. The simulation can be started by running the following command:

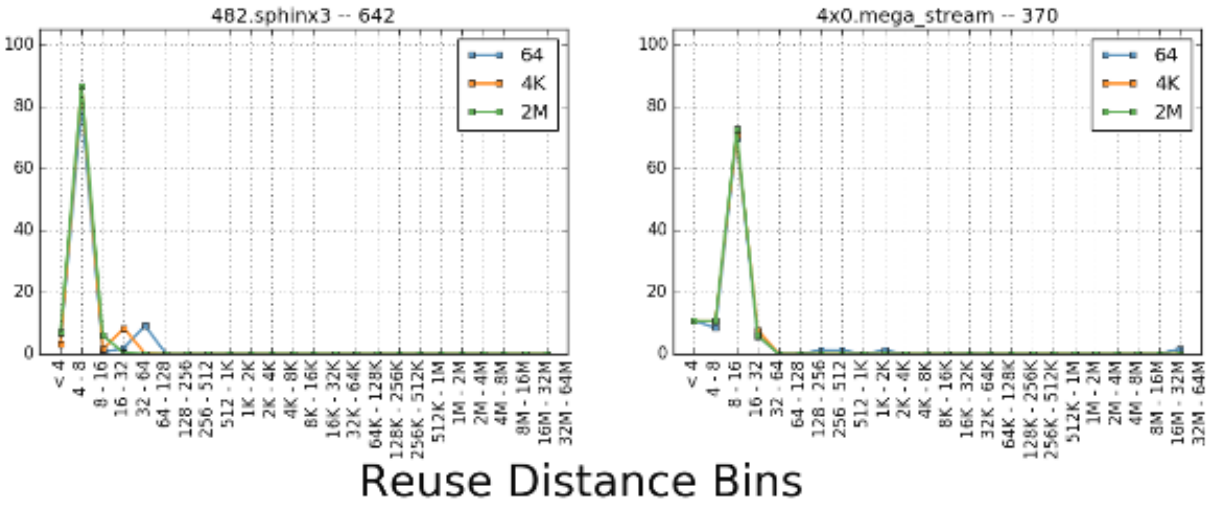
```
$GEM5/build/ARM/gem5.opt hw3config.py -c $GEM5/../test_progs/daxpy/daxpy_arm_big
    --cpu-type="DerivO3CPU" --caches --l2cache
    --num-phys-float-regs=$INT_VALUE
    --num-rob-entries=$INT_VALUE
    --num-iq-entries=$INT_VALUE
```

Please note the above is a one line command and you may have to change the daxpy path depending on your location. Also, make sure you don't overwrite your data each time you run as the default `m5out` folder is replaced each time. The following table shows the values that you should record for each run.

Parameters to record.
Number of seconds simulated
Number of floating rename lookups
Instruction Issue Rate
Idle cycles from register renaming
Number of times rename has blocked due to ROB full
Cache miss rates

Same as last lab, pick two parameters from above and plot the results from them based on the data from the various configurations of our CPU to show some kind of meaningful conclusion. Give a brief summary of what you think is happening and what the data is telling you.

Note: this is a 3-dimensional space as you are varying the parameters so a typical line/bar graph won't necessarily be the *best* way to represent the data. Try to think of something creative! Below are some examples:



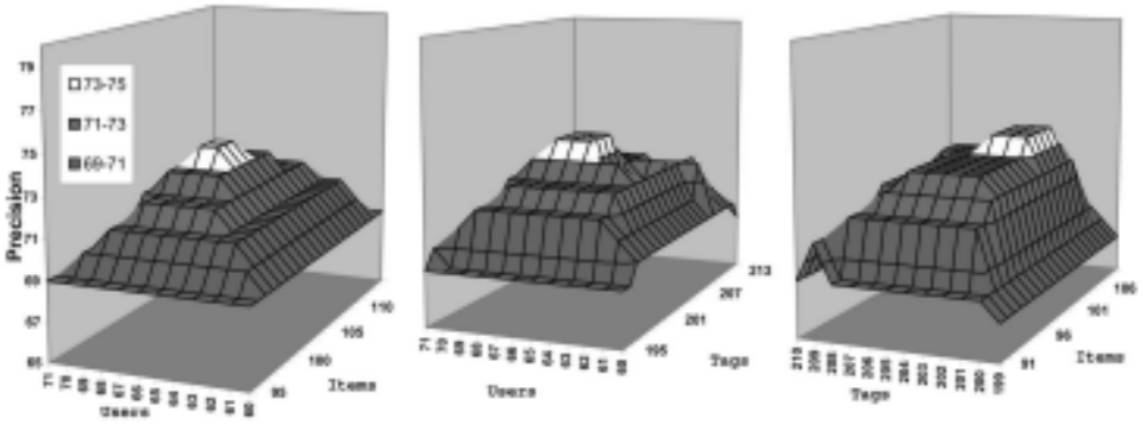
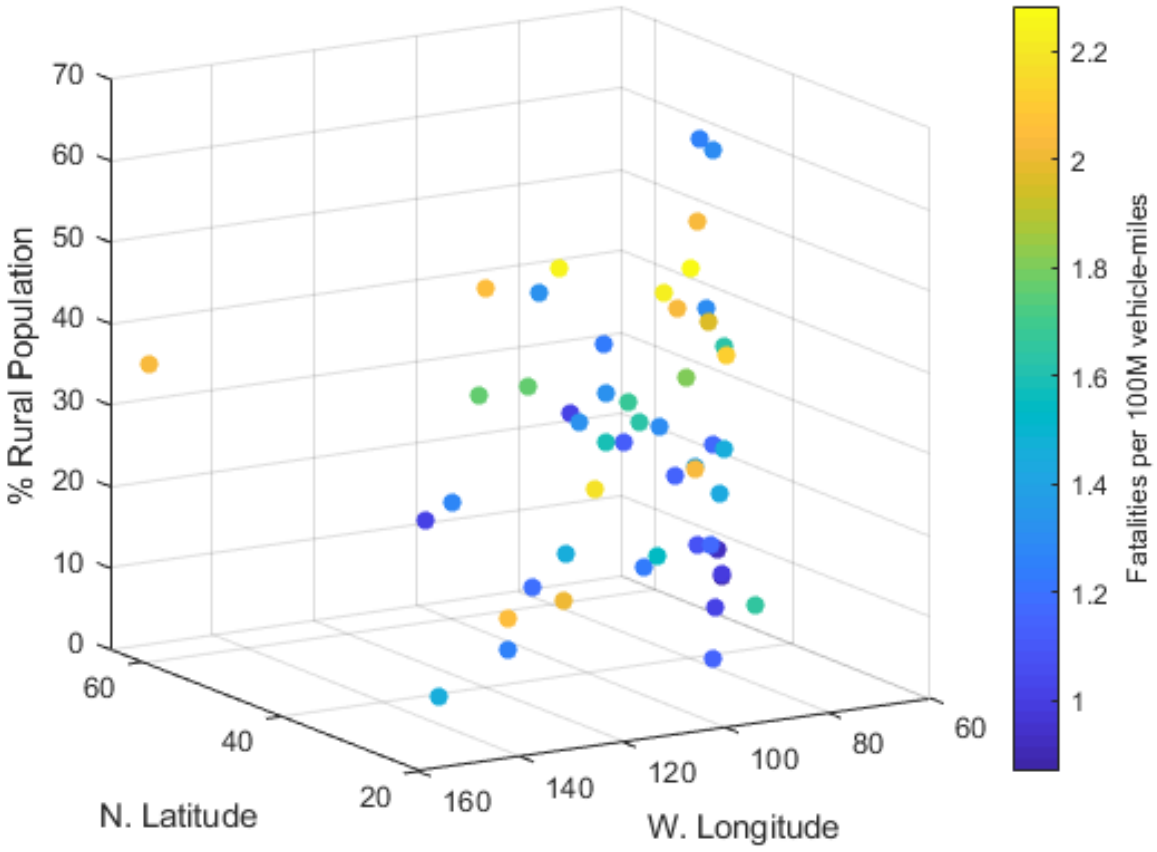


Fig. 15. Precision of Tensor Reduction as dimensions of core tensor vary for BibSonomy data set.


% Rural Population
70
60
50
40
30
20
10
0
60
40
20
N. Latitude
160
140
120
100
80
60
W. Longitude
2.2
2
1.8
1.6
1.4
1.2
1
Fatalities per 100M vehicle-miles