CPEN 411: Assignment 4
Branch Predictors

Prashant J. Nair

November 25, 2024

Contents

Chapter 1

Introduction and Setup [0 Points]

This assignment aims to implement branch prediction policies for the processor cores. It uses ChampSim, an industry and academic simulator. This document enables you to execute your code on the ECE server. For the ECE server, please ensure that you have at least **300 MB of free space** in the home directory. Please read this document very carefully and in detail.

You may choose to run this on your personal machine. We do not provide any support for that, *but it is allowed and valid*. However, be warned that some machines may get 'cooked' off as they can't handle these simulations. We are not responsible for such events.

1.1. Preparatory Steps: Setup

1. Kindly delete any previous versions of the **Champsim** folder and **assignment3.tar.gz** on **<username>@ssh.ece.ubc.ca** server. If you want to save this data, you can copy your old **Champsim** folder into your local machine before you delete them.

2. Copy the **assignment4.tar.gz** to the **<username>@ssh.ece.ubc.ca** server.

```
prashantnair@Prashants-Mac-mini Downloads % rsync -avzh --progress --stats assignment4.tar.gz prashantnair@ssh.ece.ubc.ca:~/.
prashantnair@ssh.ece.ubc.ca's password:
building file list ...
1 file to consider
assignment4.tar.gz
     107.50M 100%    2.78MB/s    0:00:36 (xfer#1, to-check=0/1)

Number of files: 1
Number of files transferred: 1
Total file size: 107.50M bytes
Total transferred file size: 107.50M bytes
Literal data: 107.50M bytes
Matched data: 0 bytes
File list size: 80
File list generation time: 0.003 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 107.53M
Total bytes received: 42

sent 107.53M bytes  received 42 bytes  2.53M bytes/sec
total size is 107.50M  speedup is 1.00
prashantnair@Prashants-Mac-mini Downloads %
```

If you do not have rsync installed. You could try installing it or use the scp command. For instance, Mac users can install these tools using Homebrew [1]. Windows and Mac users can use GUI tools like VSCode [2], CyberDuck [3], and MobaXterm [4] (**Windows Only**) to transfer files to remote servers. Linux users can use their *package manager* and install *rsync*, *scp*, or *sftp*.

3. Switch to a bash script by typing the command bash. Then untar **assignment4.tar.gz** in the ECE server and change the directory into the **ChampSim** folder.

```
ssh-linux3:~> bash
prashantnair@ssh-linux3:~$ tar -xzf assignment4.tar.gz
prashantnair@ssh-linux3:~$ cd ChampSim
prashantnair@ssh-linux3:~/ChampSim$ ls
bin                                 champsim_config_taskone.json    docs                            run_champsim.sh
branch                              champsim_config_taskthree.json  inc                             softlinks.sh
btb                                 champsim_config_tasktwo.json    LICENSE                         src
champsim_config_bimodal.json        CITATION.cff                    Makefile                        test
champsim_config_gshare.json         compile_champsim.sh             output                          tracer
champsim_config_hashed_perceptron.json config                       prefetcher                      vcpkg
champsim_config_pathfinder.json     config.sh                       PUBLICATIONS_USING_CHAMPSIM.bib vcpkg.json
champsim_config_perceptron.json     CONTRIBUTING.md                 README.md                       workloads
champsim_config_taskfour.json       create_submission.sh            replacement
prashantnair@ssh-linux3:~/ChampSim$
```

4. ChampSim uses vcpkg to manage its dependencies. In this repository, vcpkg is included as a submodule. You should bootstrap it with the following command (required only once).

```
prashantnair@ssh-linux4:~/ChampSim$ vcpkg/bootstrap-vcpkg.sh
Downloading vcpkg-glibc...
vcpkg package management program version 2023-08-09-9990a4c9026811a312cb2af78bf77f3d9d288416

See LICENSE.txt for license information.
Telemetry
---------
vcpkg collects usage data in order to help us improve your experience.
The data collected by Microsoft is anonymous.
You can opt-out of telemetry by re-running the bootstrap-vcpkg script with -disableMetrics,
passing --disable-metrics to vcpkg on the command line,
or by setting the VCPKG_DISABLE_METRICS environment variable.

Read more about vcpkg telemetry at docs/about/privacy.md
```

5. Install vcpkg with the following command (required only once).

```
prashantnair@ssh-linux4:~/ChampSim$ vcpkg/vcpkg install
A suitable version of cmake was not found (required v3.27.1) Downloading portable cmake 3.27.1...
Downloading cmake...
https://github.com/Kitware/CMake/releases/download/v3.27.1/cmake-3.27.1-linux-x86_64.tar.gz->/ubc/ece/home/pnj/faculty/prashantnair/ChampS
im/vcpkg/downloads/cmake-3.27.1-linux-x86_64.tar.gz
Extracting cmake...
Detecting compiler hash for triplet x64-linux...
The following packages will be built and installed:
    bzip2[core,tool]:x64-linux -> 1.0.8#5
    catch2:x64-linux -> 3.4.0
    cli11:x64-linux -> 2.3.2
    fmt:x64-linux -> 10.1.0
    liblzma:x64-linux -> 5.4.3#1
    nlohmann-json:x64-linux -> 3.11.2
```

6. Create softlinks for your workloads using the following command (required only once).

```
prashantnair@ssh-linux4:~/ChampSim$ ./softlinks.sh
```

1.2. Preparatory Steps: `tmux`

1. Start a **`tmux`** session. This is needed to keep your simulations running even if you get disconnected. **Note that you MUST NOT generate more than one session**.

```
prashantnair@ssh-linux4:~/ChampSim$ tmux new -s champsim-session
```

2. Once inside the session, please type the bash command.

```
ssh-linux4:~/ChampSim> bash
prashantnair@ssh-linux4:~/ChampSim$
```

3. You must now detach this **`tmux`** session by pressing: **`Ctrl+b, then d`**. We will use **`tmux`** to execute our simulations later.

```
prashantnair@ssh-linux4:~/ChampSim$ tmux new -s champsim-session
[detached (from session champsim-session)]
prashantnair@ssh-linux4:~/ChampSim$
```

1.3. Preparatory Steps: Compilation

1. You can compile your code with the **`compile_champsim.sh`** command. For instance, if you have to compile for the **`bimodal`** branch prediction as a policy, you should type

```
prashantnair@ssh-linux3:~/ChampSim$ ./compile_champsim.sh bimodal
```

2. The following policies are valid policy names that can be supplied to the compilation script: **bimodal**, **gshare**, **hashed_perceptron** [5], **perceptron** [6], **taskone**, **tasktwo**, **taskthree**, **taskfour**, **taskfive**, or **pathfinder**. The policies in red would not compile without you first implementing these policies.

3. These policies must be implemented at **branch/<policyname>/<policyname>.cc**

4. You can ONLY edit the files for the policies in red. For these, you can edit ONLY **branch/<policyname>/<policyname>.cc**. **You MUST NOT edit any other folders or files in ChampSim. You will directly get ZERO points if you do**.

5. If your compilation is successful. You will see a statement at the end of the compilation. This would look as follows for the **bimodal** branch prediction policy.

```
Configuration build complete for bimodal. The binary is now named champsim-bimodal.
```

1.4. Preparatory Steps: Execution

1. Before you execute your code. You should switch to the **tmux** session that you created. You may do so using the following command.

```
prashantnair@ssh-linux4:~/ChampSim$ tmux attach -t champsim-session
```

2. You will be taken to the same **tmux** session screen where you left off before detaching from **tmux**.

```
ssh-linux4:~/ChampSim> bash
prashantnair@ssh-linux4:~/ChampSim$
```

3. You can execute your **bimodal** branch prediction policy code with the **run_champsim.sh** command by typing **./run_champsim.sh bimodal**

4. You may detach from the **tmux** screen once the simulations start. Your assignment has nine workloads. The simulation time, per workload, will vary. You may take nearly *one hour per simulation* to run across workloads. DO NOT EDIT any of the simulation scripts.

5. Make sure you kill your simulations before you start new simulations. You can kill a simulation by attaching your **tmux** session and typing **Ctrl+C**.

6. You can monitor the progress of your simulation run by typing the **htop** command.

7. You may iterate Preparatory Steps 1.2, 1.3, and 1.4 for **gshare**, **hashed_perceptron** [5], and **perceptron** [6] policies – these policies are already implemented in ChampSim.

8. You can see the outputs of your runs in the **outputs/<policyname>** folder. The outputs are log files with the names of the SPEC2006 benchmarks.

9. For each benchmark, you will primarily be interested in LLC statistics like Branch Prediction Accuracy, Average ROB Occupancy at Mispredict, and CORE IPC values (you find other statistics interesting as well).

10. **Note that: For this assignment, your baseline system policy employs bimodal branch prediction. You would need to compare the metrics of your implemented policies, Branch Prediction Accuracy, Average ROB Occupancy at Mispredict, and CORE IPC values to these metrics for bimodal branch prediction.**

11. ChampSim also provides **gshare**, **hashed_perceptron** [5], and **perceptron** [6] policies. You may run them too (especially if you are implementing **pathfinder**) if you like.

Chapter 2

Evaluation: Branch Predictor [5 + 1 Points]

These policy simulations take a long time. Once you implement them, you would need **nearly a week to simulate** these different policies. So please start your implementation today itself!

2.1. Policies

1. **[1 Point]** Implement a new bimodal branch predictor that has the same number of 'entries' as the baseline bimodal branch predictor, but all odd entries have a single bit saturating counter per entry. All even entries (including entry 0) have a three bit saturating counter per entry. All entries are initialized to weakly-not-taken (for a single-bit entry, it is initialized to not-taken). This is called **taskone**.

You may ONLY edit the file **branch/taskone/taskone.cc** for this. You can compile your implementation using **./compile_champsim.sh taskone** command. You can then attach your **tmux** session and run this using the **./run_champsim.sh taskone** command.

2. **[1 Point]** Implement a new context based branch predictor that keeps track of the previous branch context. It keeps track of the outcome of the last branch and based on the outcome indexes into one of the two tables. Each table has half the number of entries as the baseline bimodal branch predictor. If the outcome of the previous branch was 'taken', you index into table-1, else you index into table-0.

Table-1 has entries with a single bit saturating counter per entry. Table-0 has entries with three bit saturating counter per entry. All entries are initialized to weakly-not-taken (for a single-bit entry, it is initialized to not-taken). This is called **tasktwo**.

You may ONLY edit the file **branch/tasktwo/tasktwo.cc** for this. You can compile your implementation using **./compile_champsim.sh tasktwo** command. You can then attach your **tmux** session and run this using the **./run_champsim.sh tasktwo** command.

3. **[1 Point]** Implement a tournament branch prediction policy called **taskthree**. It follows the following rules:

- It implements the gshare branch predictor (as provided).
- It implements the bimodal branch predictor (as provided).
- It implements the hashed_perceptron branch predictor (as provided).
- It takes a vote of these three predictors and then chooses the majority vote as its prediction.
- It does this only if the branch prediction has been accurate for the last branch outcome.
- If the branch predictor has been inaccurate for the last branch outcome, it only uses the hashed_perceptron branch predictor.

You may ONLY edit the file **branch/taskthree/taskthree.cc** for this. You can compile your implementation using **./compile_champsim.sh taskthree** command. You can then attach your **tmux** session and run this using the **./run_champsim.sh taskthree** command.

4. **[1 Points]** Implement a tournament branch prediction policy called **taskfour**. It follows the following rules:

- It implements the taskone branch predictor.

- It implements the tasktwo branch predictor.
- It implements the perceptron branch predictor (as provided).
- It takes a vote of these three predictors and then chooses the majority vote as its prediction.
- It does this only if the branch prediction has been accurate for the last branch outcome.
- If the branch predictor has been inaccurate for the last branch outcome, it only uses the perceptron branch predictor.

You may ONLY edit the file **`branch/taskfour/taskfour.cc`** for this. You can compile your implementation using **`./compile_champsim.sh taskfour`** command. You can then attach your **`tmux`** session and run this using the **`./run_champsim.sh taskfour`** command.

5. **[1 Point]** – BONUS (optional) Implement your own branch prediction policy – let us call this policy (**`pathfinder`**). You can ONLY use a maximum of 16KB additional overheads for pathfinder. This can be in the form of a table, or however you like. **If you exceed this limit, you will get a ZERO in the entire assignment**. Note that this needs to be a new policy. It cannot be a replica of published policies.

You may ONLY edit the file **`branch/pathfinder/pathfinder.cc`** for this. You can compile your implementation using **`./compile_champsim.sh pathfinder`** command. You can then attach your **`tmux`** session and run this using the **`./run_champsim.sh pathfinder`** command.

To get this BONUS point, (i) Your policy must be among the **top 5** speedups from the **`pathfinder`** submissions in the class *AND* (ii) you should report your speedup (using geomean) by comparing your **`pathfinder`** implementation to the bimodal baseline branch predictor. *We will start an anonymous tracking sheet for this soon.* You will not be graded for this bonus point if you do not add your name to this tracker and report your performance.

6. **[1 Point]** Write a report in pdf format explaining how you implemented **`taskone`**, **`tasktwo`**, **`taskthree`**, **`taskfour`**, and **`pathfinder`** (optional) policies. This pdf should also have graphs that compare IPC (normalized and actual IPC across policies), branch prediction accuracy, ROB occupancy over time, and other key metrics you deem fit. You should be able to explain your results convincingly, exhaustively, and logically in this document. There is a page limit of 5 pages. **Place your report within the ChampSim folder**.

2.2. A Few Pro Tips

1. If you have <1 week remaining and have not started your simulations, you may struggle to complete this assignment (if you use an ECE server). So start early!

2. Please learn how to track your jobs using the **`top`** and/or **`htop`** commands.

3. You DO NOT need to run your simulations only on **`ssh-linux4`** or whatever server you get during login. You can use **`htop`** and check if this server is busy with other jobs from other students. If so, you can ssh to other servers. This will help balance the load. We have **`ssh-linux1`**, **`ssh-linux2`**, **`ssh-linux3`** and **`ssh-linux4`** servers. For instance, once you login, can ssh into **`ssh-linux1`** from **`ssh-linux4`** by typing the command as follows:

```
jeonghyunwoo@Jeonghyuns-Laptop.local:~/OneDrive - UBC/TA/CPEN 411/2024/Assignemnts/Assignment2 $ ssh jhwoo36@ssh-linux1.ece.ubc.ca
```

4. Please kill your simulations before you restart a run. Otherwise, your simulations will stack up, and that will cause **immense slowdowns** for you.

5. For faster development, learn to use **vi**, **vim**, or **emacs**.

6. **Important**: You can test your code for fewer instructions and only a few benchmarks before instantiating a full run. This can be done via the command line as follows:

```
[prashantnair@ssh-linux3:~/ChampSim$ ./bin/champsim-bimodal --warmup-instructions 2000000 --simulation-instructions 5000000 workloads]
/astar.trace.xz
WARNING: physical memory size is smaller than virtual memory size.

*** ChampSim Multicore Out-of-Order Simulator ***
Warmup Instructions: 2000000
Simulation Instructions: 5000000
Number of CPUs: 1
Page size: 4096

Off-chip DRAM Size: 64 GiB Channels: 2 Width: 64-bit Data Race: 3200 MT/s
Warmup finished CPU 0 instructions: 2000001 cycles: 500046 cumulative IPC: 4 (Simulation time: 00 hr 00 min 03 sec)
Warmup complete CPU 0 instructions: 2000001 cycles: 500046 cumulative IPC: 4 (Simulation time: 00 hr 00 min 03 sec)
```

Here, we are testing the **bimodal** branch prediction policy on the **astar** benchmark with a warm-up of only 2 million instructions and execution of only 5 million instructions. This will be a quick run; you won't have to use **tmux**.

Note that the main submission warms up for 50 million instructions and executes an additional 50 million instructions for each benchmark.

7. If you have time, learn how to use the tracer folder to develop your **own traces** of tailored access patterns. The tracer uses **Pin**, and this strategy will help you debug with your tailored binaries. You can check if your cache policy provides the expected hit rates for your custom trace.

Chapter 3

Submission Instructions

1. Before submitting your assignment, please kill the **tmux** session as follows:

```
prashantnair@ssh-linux4:~$ tmux ls
champsim-session: 1 windows (created Tue Oct  8 19:55:27 2024)
prashantnair@ssh-linux4:~$ tmux kill-session -t champsim-session
prashantnair@ssh-linux4:~$ tmux ls
no server running on /tmp/tmux-18623/default
prashantnair@ssh-linux4:~$
```

2. To submit, please execute the following command within the ChampSim folder.

```
prashantnair@ssh-linux4:~/ChampSim$ bash create_submission.sh
```

3. You can then run **rsync** from your local computer and download the submission file.

```
prashantnair@Prashants-Mac-mini Downloads % rsync -avzh --progress --stats prashantnair@ssh.ece.ubc.ca:~/ChampSim/submission.tar.gz submis
sion.tar.gz
prashantnair@ssh.ece.ubc.ca's password:
receiving file list ...
1 file to consider
submission.tar.gz
      10.22K 100%    9.74MB/s    0:00:00 (xfer#1, to-check=0/1)

Number of files: 1
Number of files transferred: 1
Total file size: 10.22K bytes
Total transferred file size: 10.22K bytes
Literal data: 10.22K bytes
Matched data: 0 bytes
File list size: 81
File list generation time: 0.002 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 38
Total bytes received: 10.34K

sent 38 bytes  received 10.34K bytes  902.78 bytes/sec
total size is 10.22K  speedup is 0.98
```

4. If and only if you have implemented **pathfinder**, write a comment in your submission with the speedup number and your pseudonym in the Excel sheet. You DO NOT need to write a comment if you did not implement **pathfinder**.

We will evaluate your pathfinder implementation ONLY if we see this comment. So do not forget to write this if you did implement a competitive policy.

5. Please upload **ONLY** the **submission.tar.gz** on Canvas. **If we find stale files and execute those, you can get "ZERO" points for this assignment**.

6. Thus, be extremely careful and double-check if your source files and other things are right! We have 80 students and will not re-evaluate your assignment if you submit incorrect or stale work.

Bibliography

[1] Homebrew. Website Access for Homebrew. https://brew.sh/.

[2] Vscode. Website Access for Remote Development using SSH - Visual Studio Code. https://code.visualstudio.com/docs/remote/ssh.

[3] Cyberduck. Website Access for Cyberduck. https://cyberduck.io/.

[4] MobaXterm. Website Access for MobaXterm . https://mobaxterm.mobatek.net/.

[5] D. Tarjan and K. Skadron, "Merging path and gshare indexing in perceptron branch prediction," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 3, p. 280–300, Sep. 2005. [Online]. Available: https://doi.org/10.1145/1089008.1089011

[6] D. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, 2001, pp. 197–206.