Assignment 4

Lab Overview

This lab consists of two parts. In the first part of the lab, you're tasked with profiling and analysing the runtime behaviour of few provided applications using `Intel VTune Profiler`. In the second part of the lab, you're tasked with running the same applications on a simulator, known as `ChampSim`, to understand the performance impact caused by different configurations of caches in a system.

NOTE:

1. You'll need to do the second part of the lab inside a `Docker` container, that has everything set up. Although this is not mandatory and you can decide to not do it inside the container, setting up the environment and everything else will be up to you however
2. Some tips for this lab are provided in the final section of this document
3. The submission format is in the second-last section of this document

A report that summarizes the results of your analyses and the simulations will need to be submitted. The template for the report is provided in the ZIP file for the assignment.

Part 0: Getting Things Ready (10 points)

(6 points)

Install `Intel VTune Profiler` by following the instructions from this link

(You have to figure out on your own how to install it and get it up and running. Feel free to discuss among yourselves)

(2 points)

How easy was it for you to install `Vtune` and get it up and running ? What were the challenges that you faced during the installation process and how much time did it take in total?

(2 points)

Install `Docker` for your operating system, in case you haven't already installed it.

- Windows Installation Guide
- Linux Installation Guide
- MacOS installation Guide

Pull the following Docker image for the second part of the lab:

`0xd3ba/champsim-lab`

Part 1: Profiling with VTune (30 points)

The `src/` directory in the provided ZIP file has 4 programs

- `bfs.cpp`
- `matrix_multi.cpp`
- `matrix_multi_2.cpp`
- `quicksort.cpp`

- Compile each of the files with the following compiler flags:
  - `-g` (Enable debugging)
  - `-O2` (Enable high compiler optimization)
- Try running the programs for a test run. If it takes less than 1-2 seconds, open the files and configure the programs (described in the comments at the top) so that it takes at least 3-10 seconds to finish running. The parameters will be different for different system configurations.

(6+24 points)
Open VTunes and create a new project for each program. Run the following analyses and answer the following questions for each program.

Performance Snapshot **(2+2+2 points)**

1. What is the IPC that you have observed ?
2. What logical core utilization and physical core utilization have you observed ?
3. What % of the pipeline slots are memory bound ?

Hotspots **(4+20 points)**

1. List down the top hotspots that were identified along with their CPU time.
2. Identify and list down the statements in the program's source code (ignore the ones that are from source code of the libraries) that were responsible for consuming most of the CPU time (in descending order of the % of CPU time consumed. Make sure to note down the % as well)

Summarize all the results that you have obtained, along with screenshots of the results in the report.

Part 2: Simulating with ChampSim (60 points)

Pull the `Docker` image (`0xd3ba/champsim-lab`) and run a container from it.
Alternatively, if you don't want to use `Docker`, fork the `Champsim` repository and get appropriate version of Intel's PIN tool (watch out for nasty issues later on ;))

```
champsim/
    |-- Champsim/            # Forked Champsim repository
```

```
    |-- pin_v3.17/           # Intel's PIN Tool
    |-- programs/            # Source code for the programs
    |-- traces/              # Traces MUST be placed here
```

(Structure of the champsim directory inside the container)

(6 points)

Prepare the *traces* for each program by following the instructions in the Champsim repository (using the same configuration that led to 3-10 second execution time previously. You might want to edit the source files again, if you're doing the lab inside the container)

Move them to the `/champsim/traces/` directory inside the container.

(6 points)

Build `Champsim` and run the simulation on **each** of the traces you prepared previously. These will be the baseline results.

Make sure to store them separately in a directory named as `baseline/`, as they will be required later on. The following are the configurations for building and running the simulations

For building,

Parameter	**Value**
Branch Predictor	bimodal
L1I Prefetcher	no
L1D Prefetcher	no
L2C Prefetcher	no
LLC Prefetcher	no
Replacement Policy	lru
# of CPU cores	1

For running,

Parameter	**Value**
# of warm-up instructions	10
# of simulation instructions	10

You'll get outputs in `results_10M/` directory as `.txt` files for each trace.

IMPORTANT: Before starting to attempt the following questions, make sure you note down the default values of whatever you'll be changing later on.

(8 points)
Effect of using *Direct-Mapped Cache* at all levels
Explore your way out through the source code and change all the caches to *Direct-Mapped caches* while ensuring that the cache sizes remain constant. Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `direct-mapped/`

(8 points)
Effect of using *Fully-Associative Cache* at all levels
Reset the changed parameters to their original values and change all the caches to *Fully-Associative caches* while ensuring that the cache sizes remain constant.
Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `fully-associative/`

(8 points)
Effect of *halving* the size of the caches at all levels
Reset the changed parameters to their original values and decrease the number of sets in all the caches by half. Make sure to change the latency (in cycles) appropriately (and mention the values in the report)
Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `reduced-size/`

(8 points)
Effect of *doubling* the size of the caches at all levels
Reset the changed parameters to their original values and double the number of sets in all the caches. Make sure to change the latency (in cycles) appropriately (and mention the values in the report)

Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `doubled-size/`

(8 points)
Effect of *doubling* the number of the MSHRs at all levels
Reset the changed parameters to their original values and double the number of MSHRs in all the caches.
Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `doubled-mshr/`

(8 points)
Effect of *halving* the number of MSHRs at all levels
Reset the changed parameters to their original values and decrease the number of MSHRs in all the caches by half.
Build Champsim again and run the simulations using each of the traces, similar to the previous section.

Comment out your observations after comparing with the results obtained from the baseline configuration (of the appropriate trace). Did the IPC improve ? What about MPKI ? What about the average miss latency ? Why do you think it happened ?
Store the results separately in a separate directory named as `reduced-mshr/`

Summarizing the results
In the report, you need to plot graphs for each trace (for example plots, see the `example_plots/` directory in the lab file) and each of the six configurations, when compared against the baseline configuration:

- Attained IPC speedup (normalized)
- MPKI improvement/degradation

NOTE:

- You'll also need to submit **all** the `.txt` files of the results that you saved earlier (0 marks for entire part-2 of lab otherwise)

- Comment your observations (for each config.) in the report and provide the graphs (or else even if your submission is 100% correct, only **half** of the marks will be given for that part of the question)

Submission Instructions

```
<your_roll_number>/
    |-- report.pdf
    |-- baseline/                   # .txt files of results
    |-- direct-mapped/              # .txt files of results
    |-- fully-associative/          # .txt files of results
    |-- reduced-size/               # .txt files of results
    |-- doubled-size/               # .txt files of results
    |-- reduced-mshr/               # .txt files of results
    |-- doubled-mshr/               # .txt files of results
```

(Directory structure of your submission)

Compress the directory (for example, `123456789.tar.gz`) and submit it on Moodle.

Tips for the Lab (in case you get stuck) :)

- Don't know how to prepare the traces ? You might want to check out the *"How to create Traces"* section in the ChampSim's repository.
    - No need to change the PIN tool's location. It has been already set for you
    - You will need to compress whatever PIN outputs using **`xz`** utility. This is important or else ChampSim will complain. Worst case scenarios - will segfault or will be stuck in an infinite loop without telling you (thank you ChampSim).
    - One of the things that is required to prepare the trace with PIN tool will be located inside `obj-intel64/` sub-directory inside Champsim as a `*.so` file (Don't bother searching for it as it only gets created when you execute the appropriate steps ;))

- What should be the number of instructions to trace with `pin` tool ? Set it to sum of the number of warmup and simulation instructions.

- Champsim's `run_champsim.sh` script takes the number of instructions in units of millions. So if you are asked to simulate for 20 million instructions, give it a value of 20 and it will interpret it as 20 million.
  Don't make the mistake of giving it a value of 20,000,000 instead!

- If Champsim takes too long to finish simulation (15+ minutes for 10M simulation instructions), interrupt the execution and check the (partially written) `.txt` file in `results_10M/` directory to ensure everything's alright.

- How to convert the set-associative cache to direct-mapped / fully-associative ? Try to think - *What are the number of "sets" and "number of lines per set" in a direct-mapped cache ? What are they for fully-associative ?*

- When you increase the size of the cache, does the hit time (lookup latency) increase or decrease ?

- Where to find IPC and other information ? Check the `.txt` file for the corresponding trace, after the simulation has been completed. You'll find the statistics there.

- *"I forgot to note down the default values and I don't know what to set now "* - Fear not, just delete the container and create a new container from the docker image. Everything will be reset to defaults (although you'll have to prepare the traces again).