Preject 3 Report: Cache Replacement Policies on AI Workloads

Tai Duc Nguyen
ECEC 412: Modern Processor Design

November 10, 2019

Abstract

Cache is a solution to the problem of slow memory access in modern computer architecture. From the dawn of the 21st century, many CPU manufacturer, especially Intel and IBM has implemented combinations of advanced dynamic caching techniques, which can be divided into two categories: hardware and software. Some famous hardware techniques, which tackle the issues of maintaning efficient power management while providing fast cache access, are: multi-tiered cache levels and set-associative caching. However, software techniques provide the ability to increase cache hit rate. Some examples include way-predictions methods and cache replacement policies. In this project, different cache replacement policies: Least Recently Used (LRU) and Least Frequently Used (LFU) are evaluated performance wise with different hardware configurations: block size, cache size and number of associative sets.

1 Introduction

In order to achieve the objective of evaluating the performance of different cache replacement policies, a simulation was written in C by Shihao Song and Adarsha Balaji (available here: `https://github.com/Shihao-Song/Computer-Architecture-Teaching/tree/master/C621/Cache_Policy`). Using the framework established, a mechanism to test all pre-defined hardware configurations at once is written on top. After automating the testing of different configs, the LFU replacement policy is implemented in a similar fashion to the already existed code for the LRU replacement policy.

2 Simulation and Results

From the procedures in Section 1, a sample output from the output is shown in Figure 1 below:

The columns in the sample are respectively:

- Type of replacement policy
- Block size (in bytes)
- Cache size (in KB)
- Number of associative sets
- Number of cache blocks
- Cache hit rate in percentage

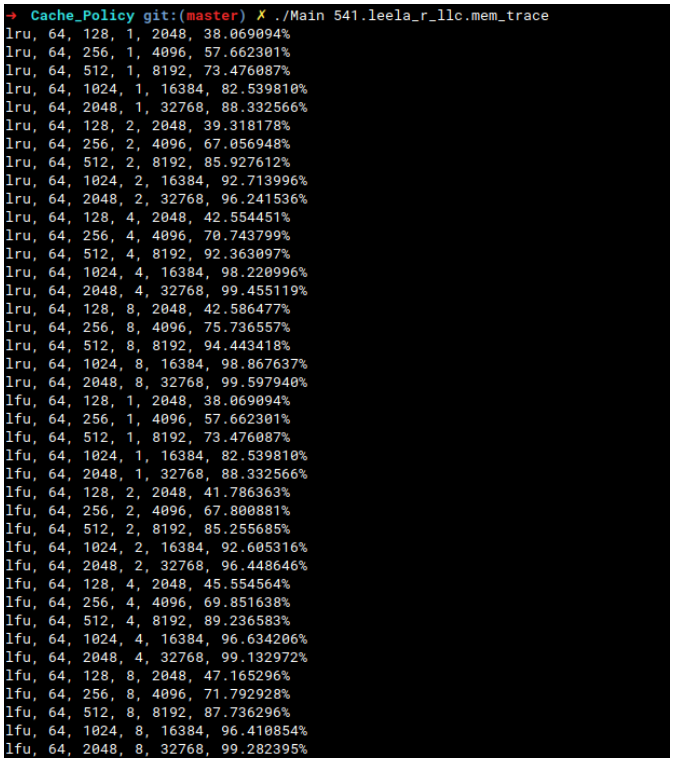
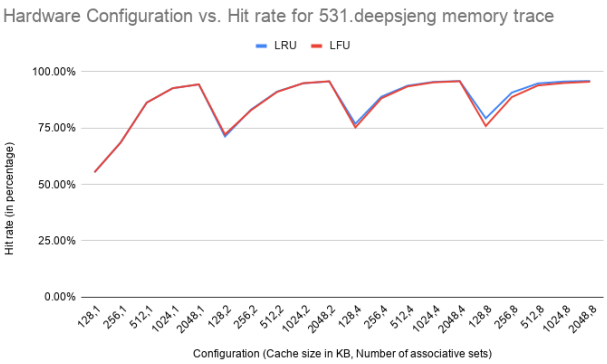
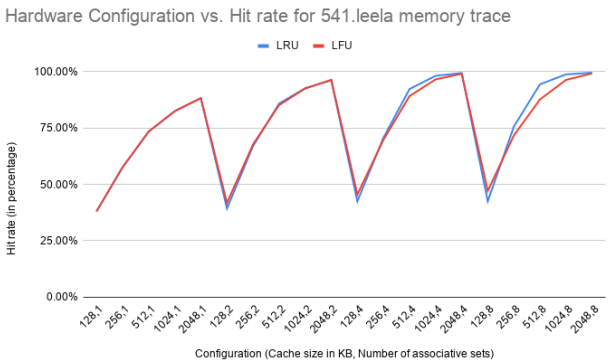


Figure 1: Sample output of the simulation

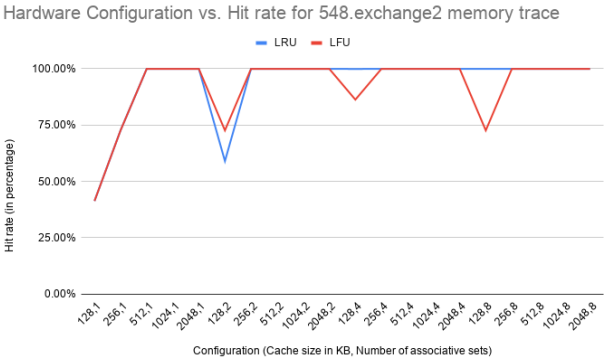
Performing this simulation over all 3 memory traces (each trace includes many Load and Store instructions), the results are shown in graphs (Figure 2 2a, 2b, 2c, and 2d).



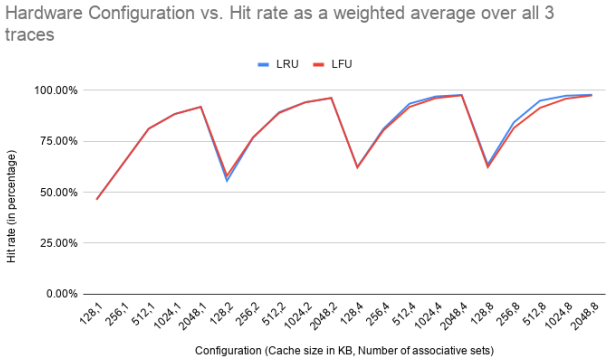
(a) Results for 531.deepsjeng memory trace



(b) Results for 541.leela memory trace



(c) Results for 548.exchange2 memory trace



(d) Overall results as weighted averages of 3 traces

Figure 2: Simulation results. The x-axis have different hardware configurations with the first number being the cache size in KB and the second number being the number of associative sets. The y-axis is the cache hit rate in percentage.

From all the figures above, the general trend is that the hit rate increase when:

- The cache size increases
- The number of associative cache stores increases

Regarding the 2 different replacement policies, LFU does better than LRU only when the number of associative cache stores is small (1 or 2). LRU has higher hit rate than LFU with larger number of associative cache stores (4 or 8). This behavior is seen most clear from the results for the 548.exchange2 memory trace. It is noted that different memory traces have different number of instructions, hence, the overall performance of each policy is decided as a weighted average and shown in Figure 2d.

3 Discussion

From the results above, the fact that cache hit rate is directly proportional to cache size and number of associative cache stores is consistent with other literature on this subject. Also, from the data, it can also be shown that the increase in performance is not linear, but rather curving out, which means that there will be diminishing returns for large cache size and high number of associative cache due to limitation in hardware area and power consumption.