Assignment #1

You can do this assignment in a Group of at most 2 students. Due: Oct. 4, 2024

1. In this exercise you will evaluate six different versions of the matrix multiplication program. You are required to use the standard $O(n^3)$ matrix multiplication algorithm. You will measure the performance of these different versions on two different matrix sizes of (2048 x 2048) and (8192 x 8192). You will also obtain the counts of various events such as the L1/L2/L3 cache misses, TLB misses, page faults, etc. from the Performance Monitoring Counters (PMCs). Performance counter values for running application can be obtained using performance monitoring tools such as PAPI [1] or perf [2].
   (a) The first variant uses `(i,j,k)` as the loop order. The next two variants use `(j,i,k)` and `(k,i,j)` loop orders. Correlate the execution time of different versions of the program with the memory hierarchy performance (misses at various levels of memory hierarchy).

   (b) Experiment whether using large pages (2MB) can help reduce the execution time of the above variants. For the huge pages, you can directly map using mmap instead of malloc. Use performance counters to reason the measured performance of the loop variants.

   (c) Next, for each of the above variants, create a corresponding blocked/tiled version with a tile size of 64. An example tiled version of the `(i,j,k)` loop is shown below.

```
for(i=0; i<N; i=i+B)
   for(j=0; j<N; j=j+B){
      A[i][j] = 0.0;
      for (k=0; k < N; k=k+B)
         for(ii=i; ii<i+B; ii++)
            for(jj=j; jj < j+B; jj++){
               for(kk=k; kk < k+B; kk++){
                  A[ii][jj] += B[ii][kk] * C[kk][jj];
   }
```

   Compare the performance for the tiled versions with the respective original versions, and reason their performance using the performance counters. For this part you can use 4K pages.

   Report the obtained performance and your reasoning in a short report (~4 pages). Your write-up should include the methodology used for measuring the execution time, and how the performance counter values were obtained.

2. In this exercise In this exercise you are required evaluate different branch predictors using the Champsim simulator [1]. You will evaluate (i) a bimodal predictor, (ii) G-share predictor, (iii) a perceptron predictor, and (iii) any variant of the TAGE predictor. The Champsim code already implements the first three predictors. You are free to use any open-source code available for the TAGE predictor (with 4 tables) and integrate it with the Champsim. To

make the comparison fair, you will consider equal hardware cost, roughly 64KB (or 64KB ± 2KB) for each branch predictor. Each predictor uses a BTB of 2048 entries which is not included in the storage budget.

(a) Evaluate 3 Cloudsuite benchmark programs and report MPKI (Mispredictions per Kilo Instructions) and IPC for the different predictors. Each group will be given a distinct set of 3 benchmark traces which should be used in the evaluation.

(b) Explore a different set of history lengths and different table sizes for the TAGE predictor, but keeping the storage budget for the predictor to 64KB, to see if the MPKI and/or IPC can be improved further.

(c) Explore a hybrid g-share and TAGE predictors with a total storage of ~64KB (for the two predictors). Assume you have an additional storage budget of 1KB for the meta predictor. The relative storage allocated to the individual predictors (g-share and TAGE) can be varied as 25:75, 50:50, or 75:25.

Report your findings in a short report (~4 pages). Your report should include the simulation methodology used and a detailed performance analysis. Make sure you simulate at least 500 Million instructions in the detailed mode after sufficient fast-forwarding and warmup.

[1] PAPI User's Guide. Available at: http://icl.cs.utk.edu/projects/papi/files/documentation/PAPI_USER_GUIDE_23.htm

[2] Linux Kernel Profiling with perf. Available at: https://perf.wiki.kernel.org/index.php/Tutorial

[3] Large Page Support in the Linux kernel: https://linuxgazette.net/155/krishnakumar.html

[4] https://man7.org/linux/man-pages/man2/mmap.2.html

[5] https://man7.org/linux/man-pages/man2/madvise.2.html

[6] N. Gober, et. al., "The Championship Simulator: Architectural Simulation for Education and Competition", https://arxiv.org/pdf/2210.14324.pdf

[7] Champsim: https://github.com/ChampSim/ChampSim

[8] Cloudsuite traces: https://www.dropbox.com/sh/hh09tt8myuz0jbp/AACAS5zMWHL7coVuS-RbpUksa?e=2&from_auth=register&dl=0