CS4290: Project 1

gabriel.spranger

February 2023

1 Methodology

I based my search space on the one provided by the Head TA. For each configuration, I invoked the simulation with those parameters and saved the results in a file. Then, for each trace, I sorted the results by AAT. This way, I could easily identify the best config for a given trace and how the AAT increases as certain parameters change. The choices for the configurations were chosen for the follow reasons:

- **C:** having small C could have high miss rates for the simple fact that it holds a small number of blocks. While not necessarilly true, this greatly reduces the search space.
- **B:** arbitrary choice to reduce the search space.
- **S:** from the minimum for VIPT up until $C - B - 5$, because if the upper limit was $C-B-S$ (the number of sets) that would mean a fully associative cache which in turn would imply that the TLB will have 2^S entries which is too costly in terms of power consumption, physical size on the chip and money.
- **P:** page sizes of computers nowadays is not common to be less than 4 kB.
- **T:** TLBs usually have some number small number of entries, the arbitrary number that was chosen was 2^5 up until the maximum 2^{C-B-S}.
- **M:** from its minimum to its maximum.

2 Results

We ran each trace with the 159 different configurations. The results can be found in the following CSV files. Each file contains the parameters used (C, B, S, P, T, M) and the AAT: matmul_naive, mcf, linpack, leela, matmul_tiled, gcc.

The following section summarizes the obtained results, provides the best configs and makes some observations based on the results:

```
for C in {12..15}
do
  for B in {5..7}
  do
    for P in $(seq 12 $(($C>14 ? 14 : $C)))
    do
      for S in $(seq $(($C-$P)) $(($C-$B-5)))
      do
        for T in $(seq 5 $(($C-$B-$S)))
        do
          M=$(($((32-$P))>20 ? 20 : $((32-$P))))
          echo "run #$nConfig"
          ./cachesim "-v" "-c $C" "-b $B" "-p $P" "-s $S" "-t $T" "-m $M" < $TRACE > "${OUTPUTDIR}/out_${nConfig}.out
          nConfig=$(($nConfig+1))
        done
      done
    done
  done
done
```

Figure 1: Space of used configurations.

- **matmul_naive**
  - **Best AAT**: 2.981
  - **Config**: $C = 15$, $B = 7$, $S = 3$, $P = 14$, $T = 5$, $M = 18$
  - **L1 size:** $2^{15} + \frac{2^{15-7}(32-(15-3)+2)}{8} = 33,472$ bytes
  - **HWIVPT size:** $\frac{2^{M} \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytes
  - **TLB size:** $\frac{2^{T} \times ((64-P)+(32-P)+1)}{8} = 276$ bytes
  - **Memory size:** $2^{M+P} = 4,294,967,296$ bytes
- **mcf**
  - **Best AAT**: 4.145
  - **Config**: $C = 15$, $B = 7$, $S = 1$, $P = 14$, $T = 7$, $M = 18$
  - **L1 size:** $2^{C} + \frac{2^{C-B} \times (PA-(C-S)+2)}{8} = 33,408$ bytes
  - **HWIVPT size:** $\frac{2^{M} \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytes
  - **TLB size:** $\frac{2^{T} \times ((64-P)+(32-P)+1)}{8} = 1,104$ bytes
  - **Memory size:** $2^{M+P} = 4,294,967,296$ bytes
- **linpack**
  - **Best AAT**: 5.257
  - **Config**: $C = 14$, $B = 7$, $S = 2$, $P = 14$, $T = 5$, $M = 18$
  - **L1 size:** $2^{C} + \frac{2^{C-B} \times (PA-(C-S)+2)}{8} = 33,408$ bytes
  - **HWIVPT size:** $\frac{2^{M} \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytes
  - **TLB size:** $\frac{2^{T} \times ((64-P)+(32-P)+1)}{8} = 276$ bytes

- **Memory size:** $2^{M+P} = 4,294,967,296$ bytes

- **leela**
  - **Best AAT**: 2.393 (4 configs achieved it)
  - **Configs**:
    - $C = 15,\ B = 6,\ S = 1,\ P = 14,\ T = 5,\ M = 18$
    - $C = 15,\ B = 6,\ S = 1,\ P = 14,\ T = 6,\ M = 18$
    - $C = 15,\ B = 6,\ S = 1,\ P = 14,\ T = 7,\ M = 18$
    - $C = 15,\ B = 6,\ S = 1,\ P = 14,\ T = 8,\ M = 18$
  - **L1 size:** $2^C + \frac{2^{C-B} \times (PA-(C-S)+2)}{8} = 33,408$ bytes
  - **HWIVPT size:** $\frac{2^M \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytes
  - **TLB size:** $\frac{2^T \times ((64-P)+(32-P)+1)}{8} = 276$ bytes (for $T = 5$)
  - **Memory size:** $2^{M+P} = 4,294,967,296$ bytes
  - **Observations:** we can see that even though the number of entries in the TLB is decreased, the AAT stays the same, which implies that this program has a working set that spans a few set of pages. Since there is no change in the ATT for $T = 6, 7, 8$, the best choice would be $T = 5$ since it would be the cheapest (in terms of size, cost and power usage) while achieving the best AAT.

- **matmul_tiled**
  - **Best AAT**: 2.741
  - **Config**: $C = 14,\ B = 7,\ S = 2,\ P = 14,\ T = 5,\ M = 18$
  - **L1 size:** $2^C + \frac{2^{C-B} \times (PA-(C-S)+2)}{8} = 16,736$ bytes
  - **HWIVPT size:** $\frac{2^M \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytes
  - **TLB size:** $\frac{2^T \times ((64-P)+(32-P)+1)}{8} = 276$ bytes
  - **Memory size:** $2^{M+P} = 4,294,967,296$ bytes

- **gcc**
  - **Best AAT**: 4.704 and 4.705
  - **Configs**:
    - $C = 15,\ B = 7,\ S = 1,\ P = 14,\ T = 7,\ M = 18$
    - $C = 15,\ B = 7,\ S = 2,\ P = 14,\ T = 6,\ M = 18$
  - **L1 size:** $2^C + \frac{2^{C-B} \times (PA-(C-S)+2)}{8} = 33,408$ bytes
  - **HWIVPT size:** $\frac{2^M \times ((64-P)+(32-P)+1)}{8} = 2,260,992$ bytess
  - **TLB size:** $\frac{2^T \times ((64-P)+(32-P)+1)}{8} = 552$ bytes (for $T = 6$)

- **Memory size:** $2^{M+P} = 4,294,967,296$ bytes
- **Observation:** the AAT of the second config is 0.001 higher, but its TLB size is half of the first config, so it would be a great tradeoff to choose the second config because its AAT is barely worse than the first config but it reduces the TLB size by half, which saves processor size, cost of the processor and reduces power usage.

As expected, each trace has its own best config, as the AAT highly depends on not only the architecture itself, but also on the access patterns of the program. So clearly there is no one-size-fits-all.

3 Size of Arch

Given the parameters (C, B, S), P, T and M, the sizes are as follows (for my implementation):

- **L1**
  - 2^C **bytes**: this is the size of the data the L1 can hold
  - $2^{C-B} \times (PA - (C - S) + 2)$ **bits**: there are 2^{C-B} blocks and each block has a tag (with VIPT the size of the tag is $PA - (C - S)$), a valid bit and a dirty bit
- **TLB**
  - $2^T \times ((64 - P) + (32 - P) + 1)$ **bits**: the TLB has 2^T entries and each entry has a VPN, PPN and a valid bit
- **HWIVPT**
  - $2^M \times ((64 - P) + (32 - P) + 1)$ **bits**: the HWIVPT has 2^M entries (one for each possible frame) and each entry has a VPN, PPN and valid bit
- **Memory**
  - Since there are 2^M frames and each of size 2^P, then the total size of memory is 2^{M+P} **bytes**