Lab5

Anton Pupkov Richard Li

March 9, 2018

Question 1: Miss ratio and cache size

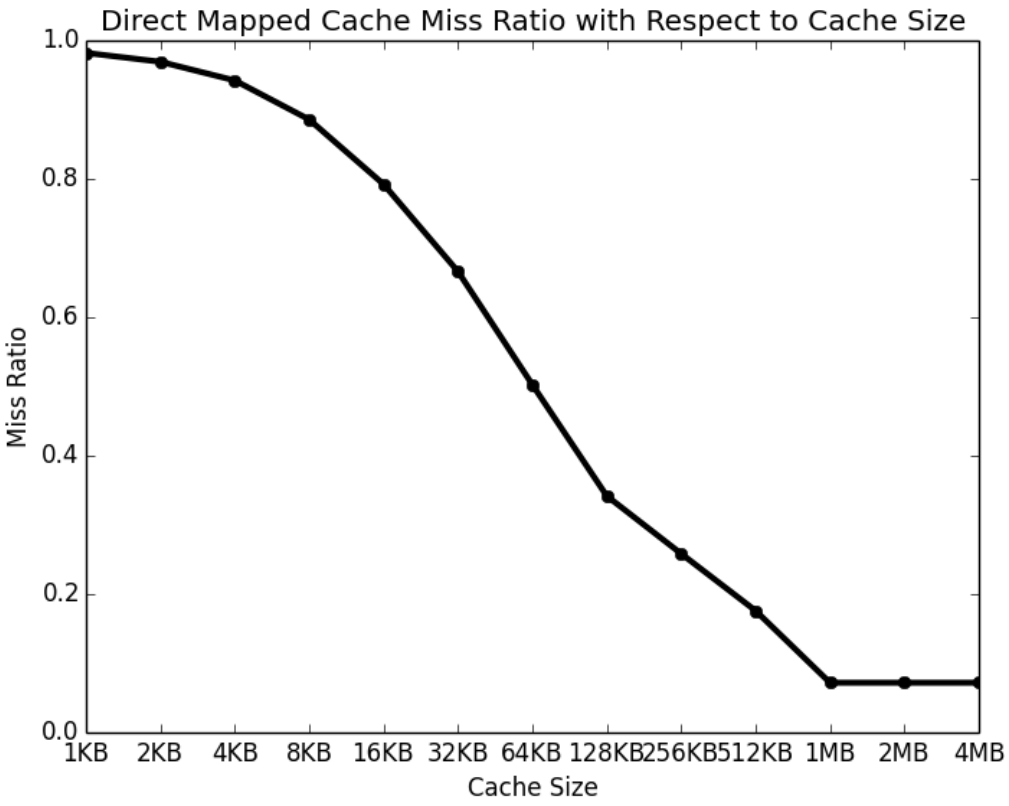

Figure 1: Direct Mapped Miss Ratio

1a

The miss ratio behaves like function $\frac{1}{1+\exp^x}$. As the cache size increases, it starts off decreasing slowly, followed by a period of fast decrementing, then reduces the rate of decreasing and ends with a slope of 0. The miss ratio decreases at first because as the cache size increases, more data can be put into the cache; thus less misses occurs. It eventually stops decreasing because the cache at 1MB could contain all data from all cache accesses.

1b

The minimum miss ratio is $\frac{1}{120000}$. It is not 0 because in the best scenario the first address will miss because the cache is empty and all 119999 subsequent addresses will hit.

Question 2: Set-associative caches

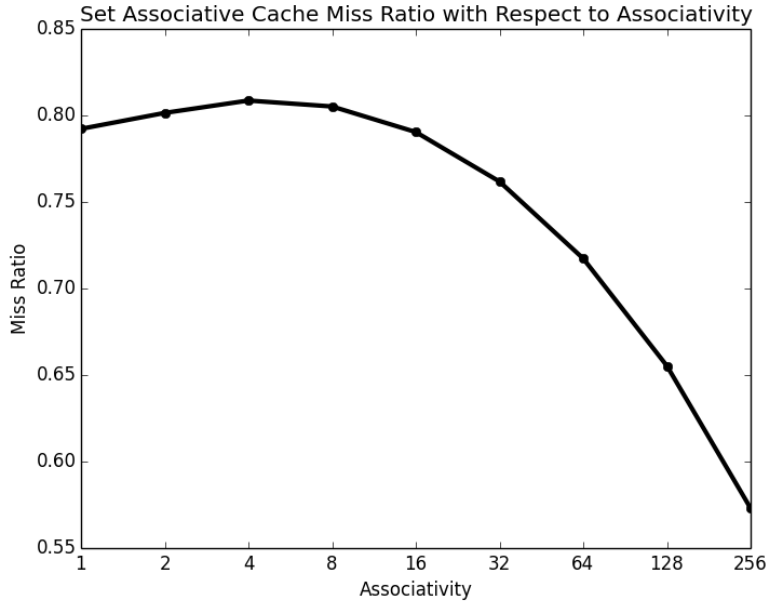

Figure 2: Set Associative Cache Miss Ratio

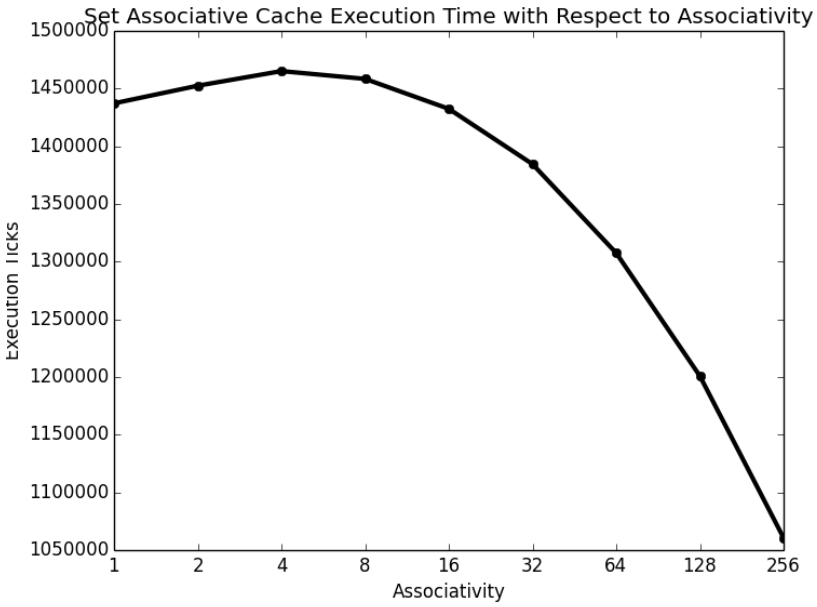

Figure 3: Set Associative Execution Time

2a

Execution time and miss ratio are proportional for the SA cache. For both SA and DM caches, since every request is done sequentially, the execution time is strongly related to the miss ratio. Thus, the execution time and miss ratio are proportional for DM caches too.

2b

It is not always worth it to increase the associativity. As seen in the figures, from 1 to 4 ways, the miss ratio actually increases as the associativity increases. However, when associativity is greater than 8, the execution time starts to decrease by roughly $base + s \times 2000$, for each step s, as associativity increases. Since the decrease function is linear, one can argue that in order to maximize productivity and performance, there is a point to stop increasing associativity. However, the point where to stop could vary in different scenarios.

Question: Non-blocking cache performance

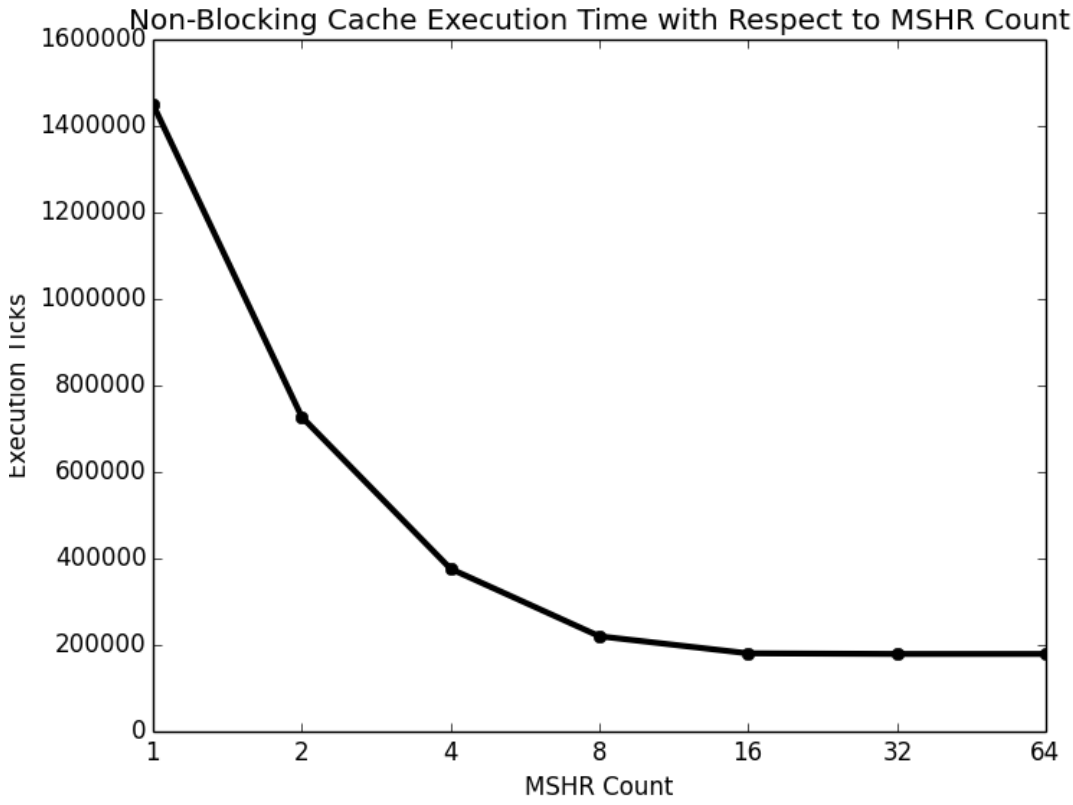

Figure 4: Non-blocking Cache Execution Time

3a

No, it is not linear because memory access takes time. If memory access is instant, increasing MSHR would linearly increase the performance to a degree.

3b (Extra)

Suppose the miss rate for a Non-blocking cache is n. The system requires 0.667 *cycles/request* and 15 *cycle/respond*. Then

$$\frac{2\ cycle}{3\ request} \times \frac{1\ request}{n\ miss} = \frac{2\ cycle}{3n\ miss}$$

Little's Law states,

$$L = \lambda \mathcal{W} \tag{1}$$

where L is the number of MSHR, λ is the number of miss per cycle, and $\mathcal{W}$ is the time required per miss. Therefore, the least number of MSHR required to minimize stalling is

$$\begin{aligned} L &= \frac{3n\ miss}{2\ cycle} \times \frac{15\ cycle}{1\ miss} \\ &= 22.5n \end{aligned}$$

If the miss ratio is 0.6, the least number of MSHR required to minimize stalling would be $22.5 \times 0.6 \approx 14$.

Question 4: Real systems

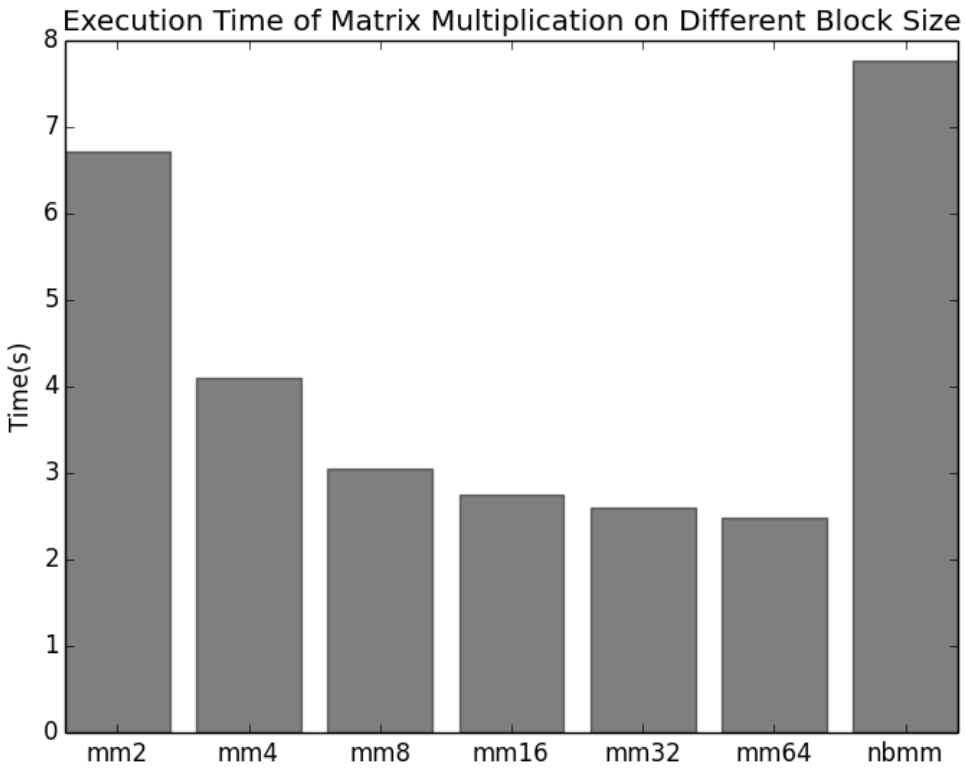

Figure 5: Performance of Macbook Pro on 1024 × 1024 Matrix Multiplication

4a

mm64 performs the best on our testing computer. Smaller block sizes are slower because less data is brought into the cache on each iteration of the for loop, which mean less spatial locality.

4b

We think a smaller block size would perform better on a smart phone. Since a smart phone has a smaller cache size, the data needed for an iteration might not fit into the top level cache, meaning fetching from the second level is constantly required. Thus, a smaller block size would ensure maximum benefit from spatial locality because data is brought into the top level cache.