ASSIGNMENT 5

RAM SIZE ESTIMATION

To estimate the RAM size for a Machine, Memory in an increment of 256 MB is added repeatedly and read from the beginning of the Memory. Increment of 256 MB is taken because it is a big enough increment so that while reading the memory it is not fetched from the cache. As we keep on incrementing the memory, at some point it will increase more than the RAM size and the any read from the memory will lead to a huge performance loss. Hence, by timing various read to different chunks of memory, we can estimate the size of RAM.

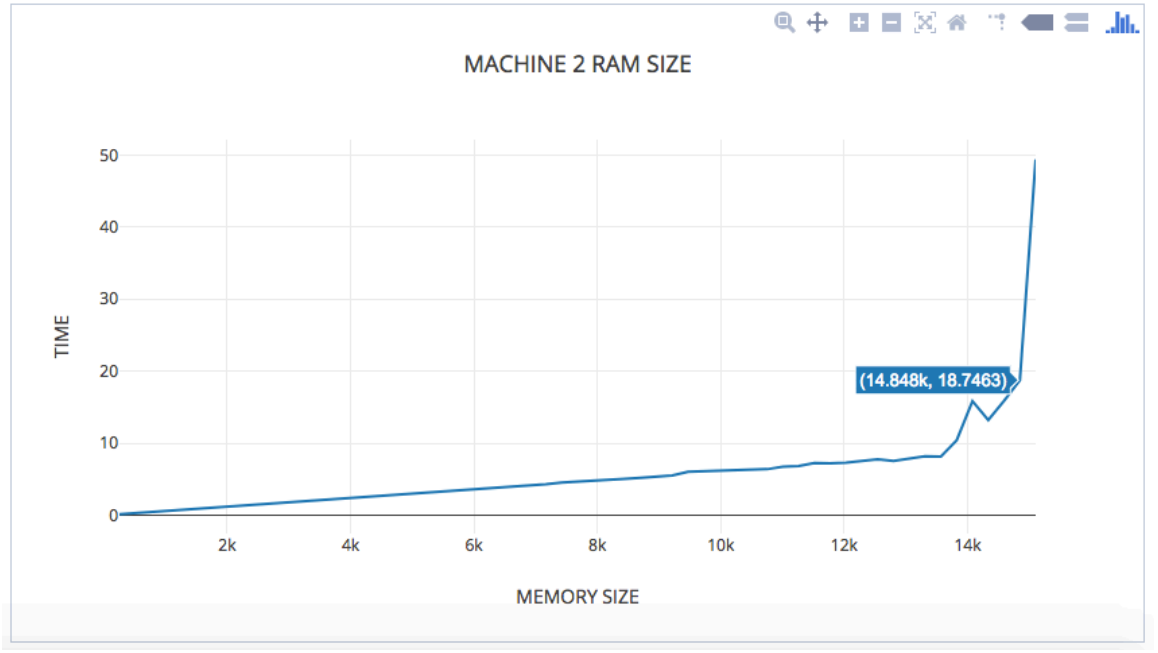

Figure 1

As we can see from the figure above, there is a huge performance loss after 14.84 GB. Hence, we can estimate the RAM size to be around 15-16 GB.

CACHE LINE ESTIMATION

Modern CPUs fetch memory in chunks called cache lines. When you read a particular memory location, the entire cache line is fetched from the RAM into the cache. And, accessing other values from the same cache line is cheap.

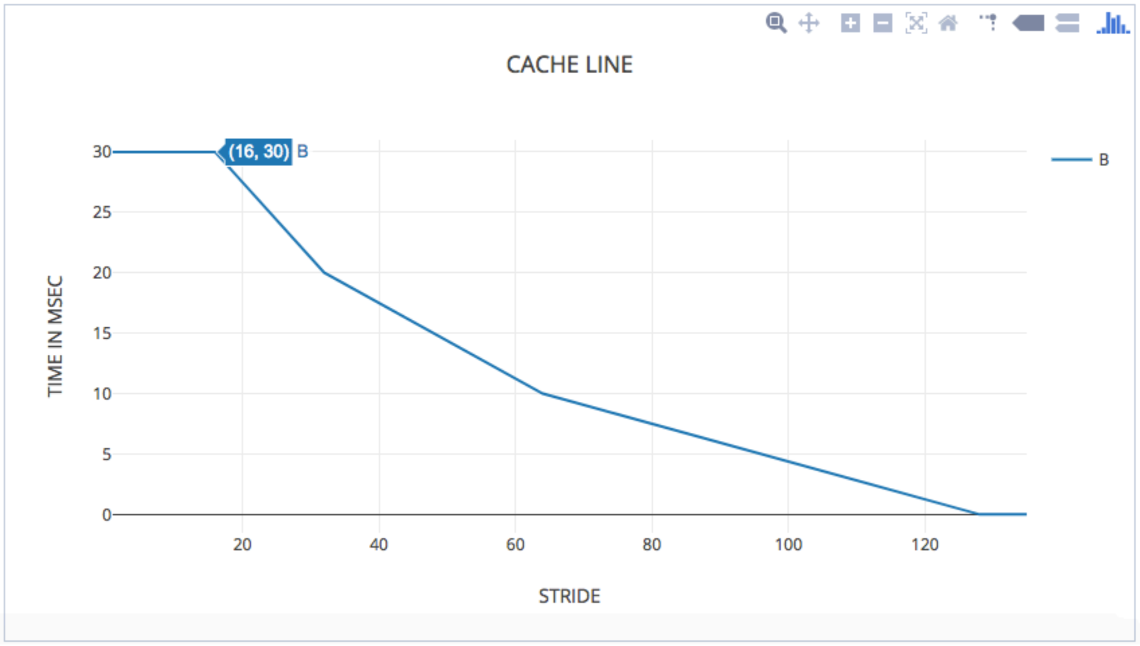

Figure 2

As we can see from the above figure, Since 16 ints take up 64 bytes (one cache line), for-loops with a step between 1 and 16 have to touch the same number of cache lines i.e all of the cache lines in the array. But once the step is 32, we'll only touch roughly every other cache line, and once it is 64, only every fourth. Hence, the graph drops drastically. Hence, we can estimate the cache line size is 64 bytes.

CACHE SIZE ESTIMATION

While estimating the Cache size, we'll jump over various sizes of arrays. By incrementing every 16th integer or the cache line we can modify every cache line, which we determined from the previous step. When we reach the last value, we loop back to the beginning. We repeat the same experiment with different array sizes, and once the array size reaches the cache size, we will notice a performance drop. Hence, by looking at the performance drops at various array sizes we can estimate the cache size. So we measure the time required by the entire array against time.

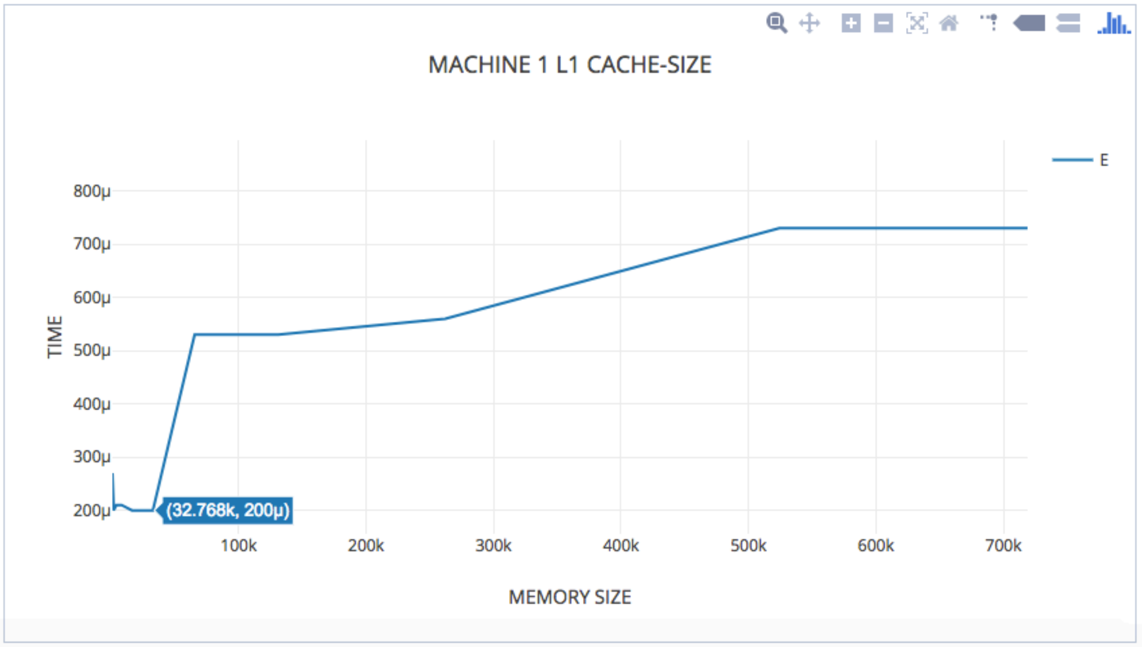

Figure 3

From Figure 3 we can determine that the L1 cache size is around 32 KB. Since, there is a huge performance loss at 32 KB. Similarly we can determine the size of L2 and L3 cache by looking at the spikes of the graph (performance loss).

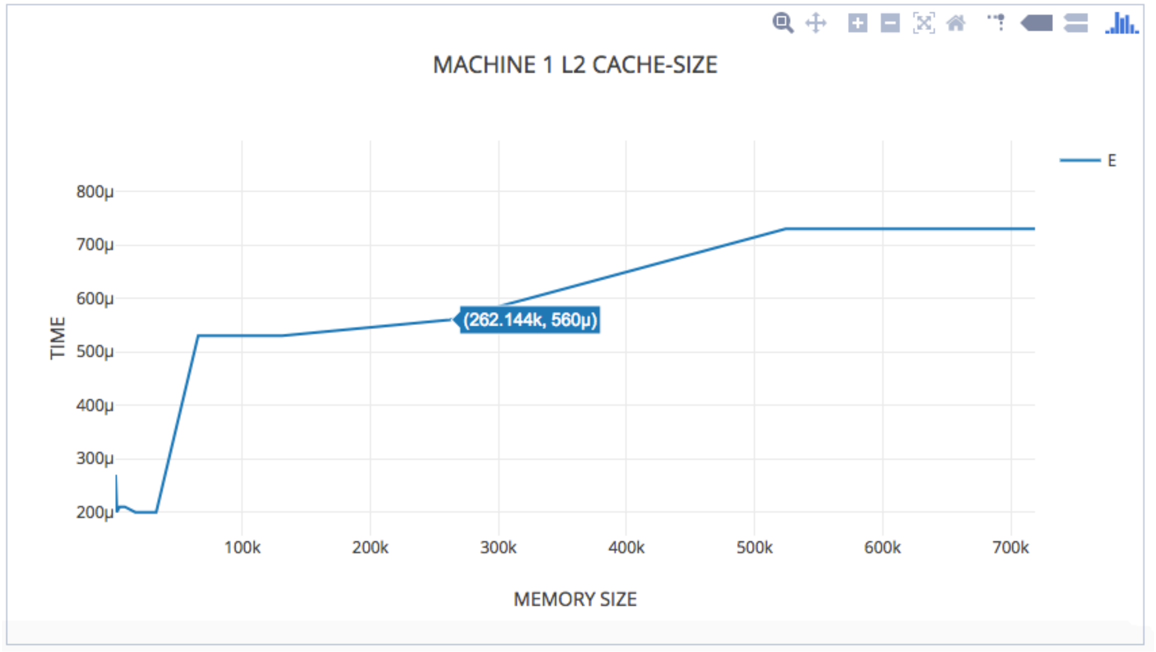

Figure 4

The next performance drop is at 256 KB. Hence, we can estimate that the L2 cache size is around 256KB. Similarly, in the figure 5 we can determine the size of L3 to be around 17 MB.

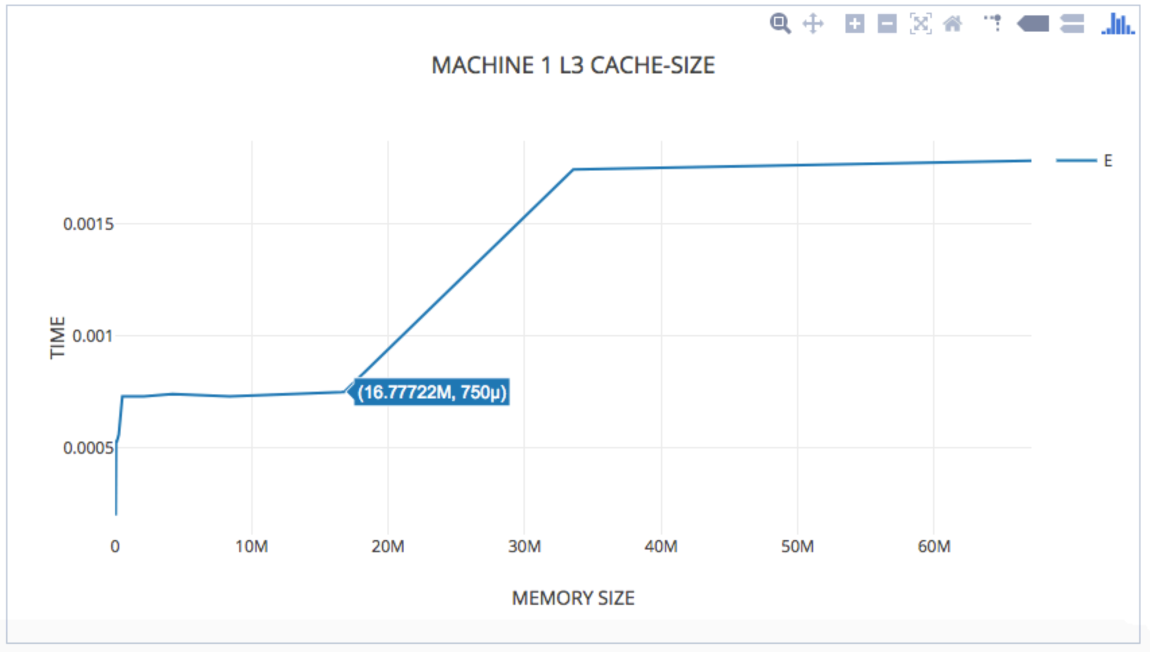

Figure 5

CACHE ASSOCIATIVITY

Cache Associativity is determined by keeping the Array size fixed while changing the stride values by power of 2 till the stride value is equal to the half the size of array length. In the program, the stride value is kept at 64 MB. At some point the effect of cache misses disappears, then the point where this occurs using the formula a = N/s gives us the associativity.

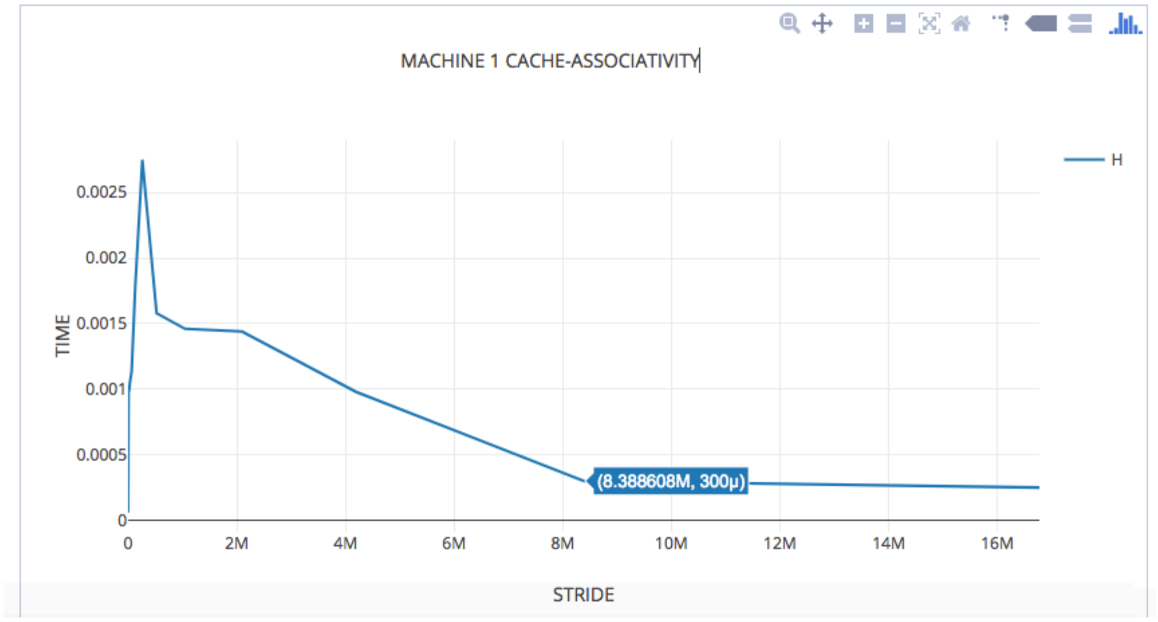

Figure 6

As we can see from figure 6, the effect of cache miss disappears at stride 8 MB and the array size is 64 MB. Hence, the associativity can be determined by dividing 64 by 8, which is equal to 8.

Specifications

CCIS machine:
L1 cache: 32 KB
L2 cache: 256 KB
L3 cache: 20 MB
L1 Associativity: 8
L2 Associativity: 8
L3 Associativity: 20

LINUX machine:
L1 cache: 32 KB
L2 cache: 256 KB
L3 cache: 8 MB
L1 Associativity: 8
L2 Associativity: 8
L3 Associativity: 8

References

[1]http://ieeexplore.ieee.org.ezproxy.neu.edu/stamp/stamp.jsp?arnumber=467697&tag=1

[2]http://igoro.com/archive/gallery-of-processor-cache-effects/