

Computer Architecture-I

Assignment 5 Solution

6. The given matrices are of size 256 x 256.
- Each element of the matrix is a double precision i.e. 8 bytes. The available cache of 16KB will thus accommodate 2048 elements. Since there are two matrices required in the loop, one can fit 1024 elements of each matrix in the cache. Thus, a block size of $\sqrt{1024} = 32$ would fill up the entire data cache.
 - The cache block size is 64 bytes i.e. 8 elements of the matrix. For the unblocked version of the code, we have a cache miss for every 8 elements of `input[i][j]` when they are fetched. There is also a miss on every column read of `output[j][i]` as one column stored in cache requires 256 x 64bytes = 16KB. For a total of 256 x 256 elements in `input[][]`, we have 8192 misses. For `output[][]`, we have a total of 256 x 256 misses. Thus, the unblocked code has total misses = 73,728.

Now, let us consider the blocked version with block size of 32 x 32. For the `input[][]`, we will have 8192 total misses again. However, for the `output[][]`, we only have 256 x 256/8 misses i.e. 8192. Thus, total misses for the blocked code = 16384.

Thus, for every 9 misses in the unblocked code, we have 2 misses in the blocked version.

- c. The blocked version of the code is shown below:

```
for (i = 0; i < 256; i = i + B) {
    for (j = 0; j < 256; j = j + B) {
        for (x = i; x < B + i; x++) {
            for (y = j; y < B + j; y++) {
                output[y][x] = input[x][y];
            }
        }
    }
}
```

7. We are considering the ideal hardware prefetcher for the unblocked version of the code. In each iteration of the inner loop, we have total of 32 misses for the `input[][]` and 256 misses for `output[][]`. Upon a miss, the prefetches from L2-cache can happen once every 2 cycles in steady state.

Thus, a total of $(32 + 256) \times 2 = 576$ cycles is required. The inner iteration caters to one row of matrix, thus giving a performance of $576/256 = \underline{2.25}$.

14. Virtual Machines (VM):

- a. It is possible to consolidate a large number of uniprocessor applications onto one high performance CMP, by making each application run on a VM with its required host operating system.
- b. Viruses, worms and spyware attack the security holes in the application or operating system. By using VMs, we can definitely limit the damage to the application/VM where the attack happened. The other VMs will be unaffected thus letting the system function sans the affected VM.
- c. It is to be noted that the VMs are finally running on the same underlying hardware. This has a finite cost in terms of TLB misses when applications have a large memory footprint. Thus, there actually might be a slight degradation in performance of memory intensive applications when run on a VM as compared to their standalone performance.
- d. Dynamic provision for extra capacity can be provided on a demand basis by creating a new VM for that application.
- e. It is possible to create a VM to host an older version of an operating system to support legacy applications.

18. Cache miss analysis for the graph in Fig:5.33

- a. There are two levels of cache indicated by distinct groups of lines for lower and higher sizes of the array.
- b. The L1- cache has a block size of 64 bytes indicated by the steady access times for strides starting from 64B for the smaller array sizes. Cache size is around 32K evident by the sudden drop in access time starting at 32K stride.
- c. The Y-axis (read time) of the graph is logarithmic. L1-cache miss can be estimated to be around 4.6ns.
- d. It can be seen that the read time degrading as the stride increases up to a certain limit. This indicates that the cache is set-associative and NOT fully associative.
- e. Since, the LRU policy used will not provide perfect prediction, there will be a few misses which cause off-chip memory accesses. This degrades the read time when compared to the ideal LRU policy.