CSCI-UA.0480-003

Parallel Computing

Homework Assignment 2 Solutions

(Total of 20 points)

1.

[1]**Write through cache**: Any block updated in the cache is also updated in the lower level cache or the memory (if this cache is the last level one).

+: Always the lower level memory has the latest data, which makes coherence and some parallel constructs easier and faster.

-: requires much higher bandwidth, which is a very scarce resource because of the contention that will occur on buses, ports, and chip pins (if memory is updated).

[1]**Write back cache**: the block is written back (if dirty) to the lower level cache or memory only if this block is replaced from the cache.

+: saves a lot of bandwidth

-: Lower level memory does not always have the latest version of the data.

Note: If we have three levels of caches for example (L1 is connected to the core, L2 is lower, and L3 is the last level cache before going to memory), then if L2 is write through and L3 is write back: L2 will always update the blocks of L3 but L3 will not update the memory until the block at L3 is replaced (i.e. kicked out of L3 by a replacement policy).

2. [1] Use banked memory

3. [2] The closer we are from the processor the more the access latency becomes more important (needs to respond to the processor as fast as possible). The closer we are from main memory, the more important the hit rate is (because main memory access is expensive). So for L1: access latency is more important; for L3: hit rate is more important. This is why L1 is smaller in size than L2, which is smaller than L3. This also explains why L1 has smaller associativity than L2, which is smaller than L3.

4. [3 (1pt for using Amdahl's law, 1 for 3-fold, and 1 for 5 fold)] (α below is the F we saw in class)

This requires the application of Amdahl's law. The part that can be parallelized is $\alpha = 1 - 20\% = 80\%$. The speedup predicted by Amdahl's law is $speedup = \frac{1}{1-\alpha+\frac{\alpha}{N}}$.

Achieving a three-fold speedup requires that:

$$\frac{1}{1-\alpha+\frac{\alpha}{N}} = 3 \Rightarrow \frac{1}{0.2+\frac{0.8}{N}} = 3 \Rightarrow \frac{0.8}{N} = \frac{1}{3} - 0.2 \Rightarrow N = \frac{0.8}{\frac{1}{3}-0.2} = 6 \quad (1.1)$$

Achieving a 5-fold speedup requires that:

$$\frac{1}{0.2+\frac{0.8}{N}} = 5 \Rightarrow N = \frac{0.8}{\frac{1}{5}-0.2} = \frac{0.8}{0} = \infty \quad (1.2)$$

So, it is impossible to achieve a 5-fold speedup, according to Amdahl's law.

5. [3] Coherence protocols affect performance negatively, for several reasons:

- Any write to the memory is delayed till the protocol ensures exclusive ownership of the block.
- Coherence protocol causes extra cache misses due to the invalidation.
- Coherence protocol results in more traffic due to the messages sent to invalidate/update/acknowledge, etc.

6.

a. [2] Because "double size" means the problem size is double the original problem. This means more work is needed and hence more opportunities for parallelism and speedup.

b. [2] If we keep increasing the number of processes (i.e. the x-axis) the speedup will saturate then decreases. This is because if you have more cores than needed, many of them will be idle or do very little work but keep consuming some resources.

7. [2] In slide# 12 of the performance analysis lecture (lecture 6), efficiency decreases as the number of processes increases because the more cores you add, while the problem size is fixed, the less the work those extra cores will do (if any), hence we are not using the available resources efficiently.

8. [1 point] Load imbalance becomes more severe as more synchronization points exist in the program. Because, at each synchronization point, each thread/process must wait for all others to reach that synchronization point. So, if a thread/process has very little work, it will wait at the synchronization point for the slowest thread/process to reach that point, leading to performance loss.

9. [1 point] The parallelism as T_1/T_∞ is an approximation of the maximum possible speedup that we can obtain by any number of processors. So, even though we have 8 parallel paths, but we will never get 8x speedup even if we use 8 processors because there is a sequential part at the beginning and at the end of the graph shown.

10. [1 point] No, we cannot make this assumption. Because on thread can get a cache miss when accessing a data and the other thread gets a cache hit. Also, memory response time is not uniform (Non-Uniform Memory Access or NUMA as we saw in class).