Yazhuo Gao
Matthew Sills

ECE 506 Program3 Report

All samples run with trace file *swaptions_medium_cleaned*

- Discuss trends with respect to change in the configuration of the system as well as across the protocol
- Consider the following issues.

1. Which helps more on miss rate? Doubling the associativity, the block size, or the cache size? Does it depend on which protocol is in use?

For the Dragon protocol, increasing the cache size had the most impact over the total increase. Increasing the block size from 128 KB to 2048 KB resulted in a 93.43% lower miss rate. Increasing the cache associativity from 1 set to 16 sets had the second largest impact resulting in an 85.47% lower miss rate. Increasing the cache block size had the lowest overall impact resulting in a 47.57% lower miss rate.

The single largest decrease in miss rate came from increasing the cache size from 128 KB to 256 KB. This one step resulted in a 74.99% reduction in miss rate. Increasing any of the metrics has diminishing returns. The first doubling provides the largest reduction in miss rate with each subsequent doubling providing a smaller reduction. The final doubling provides a negligible reduction for all metrics. Cache associativity is slightly different as its first two doublings reduce the miss rate by a similar amount. The final two doublings though, provide almost no reduction in miss rate. This signals that a cache associativity of 4 is the best choice for this problem.

For the write-through protocol, increasing the block size helps more on miss rate. Increasing the cache size from 128KB to 2048KB reduces 2.56% total cache miss. Increasing the cache associativity from 1 to 16 reduces 4.05% total cache miss. Increasing the block size from 32B to 512B reduces 13.20% total cache miss.

The largest reduction of total miss rate happens when block size increases from 32B to 64B, which reduces 6.54% total cache miss. For cache size, the largest reduction happens when it increases from 128KB to 256KB, which reduces 2.47% total cache miss; for cache associativity, the largest reduction happens when it increases from 1 to 2, which reduces 3.83% total cache miss.

The Dragon and write-through protocols have an inverse relationship for which metric produced the largest reduction in miss rate. For the Dragon protocol, the order from largest to smallest was cache size, cache associativity, and then cache block size. For the Write-through protocol, the order is swapped; cache block size, cache associativity, and then cache size.

2. Does higher associativity benefit a large cache or a small cache more?

Higher associativity benefits a large cache more than a small cache because a small cache is more likely to encounter capacity misses preventing the associativity increase from having a large effect.

3. Which protocol (Dragon, WT) has the most memory transactions? Which one has the least? Why?

The write-through protocol has the same total number of reads and writes as the Dragon protocol, while the write-through protocol has the most memory transactions and the Dragon protocol has the least. The total memory transaction of WT depends on the sum of read misses and write transactions. Therefore, since the write-through protocol has many more read misses and write misses than the Dragon protocol, it significantly increases the memory transaction for WT.

4. Compare the number of read and write misses in Dragon vs. WT. Explain your observations.

Dragon has a dramatic reduction of read and write misses compared to WT, because WT does not write the dirty block into the cache but writes directly to the memory, resulting in more misses on both read and write. Dragon updates the misses to the cache and writes back the dirty blocks to the memory, leaving updated values in caches and resulting in less misses.

5. Compare the number of bus transactions and writebacks in Dragon vs. WT. Explain your observations.

The Dragon and Write-through protocols have a similar number of bus write transactions. The Dragon protocol has many fewer bus-read transactions than the Write-through protocol. The Write-through protocol does not have any writebacks whereas the Dragon protocol has many writebacks. Overall the Write-through protocol has more bus traffic than the Dragon protocol. The Write-through protocol has a similar number of bus write transactions but has significantly more bus read transactions. Even adding the writeback traffic for the Dragon protocol leaves the Dragon protocol with at least 10,000 fewer bus transactions.

This is due to how the Write-through protocol functions. Every time a cached value has a processor write that value is also written to memory even if no other cache needs the new value. This causes significantly more bus traffic than the Dragon protocol which only creates bus traffic to read a value or when writing a shared value.