Project 4 Report

Introduction

Cache coherence is hard problem to solve for multi-processor systems. Sharing a memory space in which each entity has its own copy cache blocks presents many challenges for performance. The five coherence protocols that are explored in this project are MSI, MESI, MOSI, MESIF, and MOESIF. As the amount of states in the protocol increases, generally the performance increases, but so does the complexity of the implementation. Five memory access patterns between varying amount of processing elements are simulated to explore the performance of each protocol.

Experiment 1

The first experiment is a four processor trace where the first three processors are accessing the same address in a write-write-read pattern. The last processor just reads an independent address three times. The protocols all perform very similar for these small amounts of accesses. However, the two protocols that implement an owner (O) state perform better than the others. The last processor to modify the shared block will receive a recall and go to S on the first read of that address, which will place it in the owner state. This allows it to forward the dirty block to the other processors rather than require the other processors to access the high-latency memory. MOSI and MOESIF have the best performance for this access pattern.

Experiment 2

Experiment 2 is four processor trace with a mixed bag of lots of reads and sparse writes. The same trend of more states provides better performance applies here too. Although, since there are lots of reads rather than writes, the forwarding (F) capability rather than the owner (O) capability is more important. MESIF performs better than MOSI by cutting down on the amount of memory reads, even though cache misses and writebacks increases. MOESIF performs significantly better than all the others with both the forward and owner states implemented.

Experiment 3

Experiment 3 bumps the amount of processors up to 8 and includes similar amounts of reads and writes. Forwarding, similar to experiment 2, performs better than protocols without forwarding because reading from memory remains the most important thing to reduce. MOESIF performs significantly the best by increasing the number of cache-to-cache transfers and reducing the number of memory reads.

Experiment 4

Experiment 4 includes 16 processors where processors 0 – 7 have a similar access patterns, and 8 – 16 have similar access patterns. Forwarding, similar to experiment 2 and 3, performs better than protocols without forwarding because reading from memory remains the most important thing to reduce. MOESIF performs significantly the best by increasing the number of cache-to-cache transfers and reducing the number of memory reads.

Experiment 5

Experiment 5 has 8 processors, each accessing only a total of 4 different blocks. Each processor only writes, at most, once. An interesting result is MESI performs better than MOSI for this experiment. This is because MESI has more of an impact on increasing cache-to-cache transfers due to the fact that a GETS or GETM while the block is in E always results in a cache-to-cache transfer rather than needing to access DRAM. The owner state is less useful when writes are sparse. Again, MOESIF performs the best overall as it optimizes the amount of cache-to-cache transfers and reduces the amount of DRAM accesses.

Conclusion

Depending on the amount of reads and writes the owner/exclusive/forward states can be more important for reducing the amount of memory reads. Overall, MOESIF, which implements all of the states, performs the best.