An Evaluation of Cache Coherence Protocols

Linda Bigelow
Veynu Narasiman
Aater Suleman

Introduction

In shared memory systems each processor has its own cache but memory is shared among all processors. Therefore, it is possible for multiple processors to have the same memory block in their cache. If one processor wishes to modify a shared memory block, it is important for all other processors who have that block in their cache to see this modification. Otherwise, a consistent memory image would not be maintained, and processors could read stale data values. To avoid this, there needs to be some mechanism for keeping caches *coherent.* This involves each processor knowing about the memory accesses made by other processors. However, this added complexity can often times negatively affect system performance. The best cache coherency protocol will not only keep caches coherent, but will do so with minimal performance loss.

Existing Cache Coherency Protocols

There are several different *snoopy* based cache coherence protocols that have been proposed [1]. All of these protocols assume a special bus where one processor can issue bus operations that other processors can observe, or *snoop*. Perhaps the simplest of these protocols is the classic MSI protocol, which involves three states (Modified, Shared, and Invalid) that a memory block can be in. If a processor has a memory block in the *Modified* state, that implies that it is the only processor which has this block in its cache, and the copy it has in its cache is different than the copy in memory. The *Shared* state implies that the processor has this block in its cache, but others may also have a copy in their cache. The copy in memory is also valid for the *Shared* state. Lastly, the *Invalid* state implies that the block of memory is not in the processor's cache. State transitions for a memory block may happen when either a processor wishes to read or write to that memory block, or if it *snoops* a read or write to that memory block. For example, if all of the processors have the same block of memory in their cache (all are in the *Shared* state), and one of them wishes to write to that block, it will issue a bus write and transition to the *Modified* state. All of the other processors will *snoop* that write and invalidate their copy of the cache line (transition to *Invalid* state), thereby maintaining cache coherency.

An improvement to the MSI protocol is the MESI protocol which adds a 4th state, the *Exclusive* state. A processor having a memory block in the *Exclusive* state implies that it is the only processor that has the data in its cache, and that the copy in memory is also valid. The advantage of having such a state is that if a processor wants to write to a memory block that is in the *Exclusive* state, it can save a bus operation by not having to broadcast this write to the other processors since it knows that no other processor has the data in its cache.

An extension of the MESI protocol is the MOESI protocol which adds in a 5th state known as the *Owner* state. The goal of the MOESI protocol is to minimize accesses to memory. One way of doing this is by having the *Owner* of a cache block supply the data to another processor instead of having that processor read the data from memory. In addition, the inclusion of the *Owner* state allows for fewer write backs to main memory.

Our Project

The first step in the project is to perform a literature search on current cache coherence protocols and the bus interface units that support these protocols. We will then extend one of the cache coherency protocols (eg. to support multiple owners) and discuss future improvements that can be made to bus interface units to support new coherency protocols. To evaluate our extended protocol against existing protocols, we will simulate the various protocols on a multiprocessor system. We will use the M5 simulator [2] from the University of Michigan for our simulations.

M5

M5 is a full system simulator that currently supports the Alpha ISA. M5 has multiprocessor capabilities and supports a bus-based interconnect with snooping coherence. In fact, four snoopy based coherence protocols (MSI, MOSI, MESI, and MOESI) are already implemented in M5, which makes it an ideal choice for us to use in our simulations. In addition, M5 reports performance numbers that we will need to use in order to evaluate the different protocols. To simulate our own extended cache coherence protocol as well as any other protocols we are interested in studying, we will need to modify the M5 simulator to support those protocols.

Benchmarks

There are several possible benchmarks that can be used to exercise the various coherency protocols. One option is to use benchmark suites that were designed to test multiprocessor systems, such as SPLASH and SPECjbb. Since these benchmark suites are often used in research, it should be possible to verify our simulations by comparing our results against previously published research. Another option we are considering is using multithreaded applications written in more modern parallel programming languages, such as Cilk and UPC.

Metrics

The primary metric we will use to compare the efficiency of the coherency protocols is overall system performance. Additionally, we will characterize each protocol in terms of processor utilization (the time spent waiting for memory), bus utilization, and the number of accesses to physical memory. With this information, we hope to determine the reason for system performance differences between the various protocols.

Conclusion

For this project we plan to study several existing cache coherence protocols and the bus interface units that they use. We will then simulate these protocols using M5 and evaluate them based on overall system performance. This will give us an understanding of what constitutes a good protocol and where exactly there is room for improvement. Using this knowledge, we will extend one of the protocols and compare its performance to the other protocols. Lastly, we will discuss possible improvements to bus interface units to support novel cache coherence protocols.

References

[1] Suh, T., Lee, H. S., and Blough, D. M. 2004. Integrating Cache Coherence Protocols for Heterogeneous Multiprocessor Systems, Part 1. *IEEE Micro* 24, 4 (Jul. 2004), 33-41. DOI= http://dx.doi.org/10.1109/MM.2004.33

[2] N. L. Binkert, E. G. Hallnor, and S. K. Reinhardt. Network-Oriented Full-System Simulation using M5. Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW), pp. 36-43, February 2003.