

Chapter 5

Conclusions

Motivated by improving software performance in multiprocessors, this thesis has identified a number of performance measurements that can be made in hardware. These measurements were coalesced into a hardware performance monitor for NUMAchine which implements 18 of the 23 recommended features.

In retrospect, the MIPS R4400 processor used in NUMAchine was an excellent choice from a performance monitoring standpoint. The observable pipeline status and the single in-order pipeline were especially helpful at keeping the design simple yet flexible.

5.1 Contributions

In addition to developing hardware performance monitoring circuitry for NUMAchine, this thesis has made the following contributions:

Comprehensive List of Performance Measurements for Multiprocessors

A thorough examination of factors affecting performance in cache-coherent, shared-memory multiprocessors resulted in a list of 23 important types of performance measurements. Many of the measurements can be implemented directly by the processor, but some of them require specialized off-processor hardware. The cost of the external hardware is modest, so it should be feasible to add to all but the most cost-sensitive computers. In particular, the hardware suggested would add significant value to application developers and many scientific and engineering users.

Informing Memory Operations, TRIGGER Support

The idea of informing memory operations, where a cache miss stimulates the invocation of a software miss handler, was proposed in [Horowitz95]. To enhance performance and

permit greater selectivity, this thesis has proposed that an external TRIGGER pin on the processor enable or disable the handler's invocation.

PhaseID Register

To separate performance statistics at a fine or coarse granularity, this thesis has recommended a PhaseID register. To guarantee proper consistency under all circumstances, the register should be integrated into the processor. A novel aspect of PhaseID use is attaching it to all network and memory transactions so that those subsystems can also be sensitive to changes in program state. Additionally, the problem of maintaining consistency of PhaseID contents to memory operations was briefly described.

Recent independent work by Martonosi [Martonosi96] has proposed a similar *category* register for separating performance data. However, in their use the category does not travel through the network with memory transactions as done with PhaseID.

Memory State Indicator

In addition to transporting the PhaseID value with transactions from the processor, this thesis has suggested attaching a memory state indicator (MSI) to responses. In the NUMAchine implementation, the MSI describes the state of the memory block as found at the home memory module or the network cache. It is used to help identify data sharing patterns and estimate the amount of coherence overhead necessary to construct the response.

To date, the only feature (of which we are aware) that is similar to this is the ability of some processors to count hits to cache lines found in a particular state.

Invalidation and Ownership Misses

Recent work on multiprocessor misses from invalidate protocols [Dubois93] concentrates on the precise identification of invalidation misses and obtaining ownership for writes. In particular, the penalty of obtaining ownership is downplayed because of relaxed consistency models. Instead, invalidation misses are categorized as true and false sharing misses

by simulating updates within a write-invalidate protocol. Unfortunately, this scheme is not practical to implement in hardware because the intrusive updates involved could saturate an interconnection network.

This thesis has described obtaining ownership as an ownership miss because of the penalty associated in sequentially-consistent multiprocessors. Also, practical hardware mechanisms for detecting invalidation and ownership misses were outlined. Additionally, although invalidation misses are not categorized distinctly as true or false sharing misses, it was noted that some false sharing patterns can be detected using the external invalidation hit counts and separating misses based on the MSI and the processor request type.

Other Contributions

A reactive cache conflict detection scheme was developed by separately counting accesses to each cache line. A similar, but more limited, scheme was used in [Singhal94] which involved counting accesses to even and odd lines separately. Also, by raising an interrupt when one line suffers from excessive misses, our approach can be used to help identify the data objects that conflict in the cache.

Using the same base hardware, a method was developed for performing fine-grained timing that is comparable in overhead to basic block counting. This timing information is an alternative to program counter sampling, the current widely-used technique.

Due to current latency-hiding mechanisms found in recent processors, miss cycles per instruction (MCPI) no longer gauges processor performance well. Instead, this thesis proposed a new metric, apparent MCPI, to help quantify the positive effect these mechanisms have on performance by measuring lost processing opportunity. In practise, it may be difficult to precisely quantify this opportunity cost; this could be the subject of future investigation.

5.2 Future Work

As testified by new processors, the realm of hardware-based performance monitoring is just beginning to take shape. Future work in the monitoring area involves better integration of monitoring functions in processors and new software tools to exploit the monitoring features. Also, improvements in computer architecture will demand new measurements be made for effective deployment of resources.

With respect to NUMAchine, future work involves the construction of hardware to monitor queue lengths and memory state transitions. A brief introduction to this work in progress is presented in Appendix A. Additionally, other work will involve providing an off-processor functional equivalent to the informing memory operations, exploring a use for the response to the read that changes the PhaseID register, creating a Configuration Controller circuit to use the SRAM as a small trace buffer, and the development of a bus-transaction tracing board.