A Brief Review of "Shared Memory Consistency Models: A Tutorial"

Yi Luo

1 Summary

Parallel systems that support shared memory need a formal specification about how to manage the memory for read and write operations from multiple processors. A most straightforward memory consistency model is named sequential consistency model which strictly constraints that 1) program order should be maintained from individual processors and 2) a global sequential order should be maintained from all processors. Even thought this model is intuitive and simple, lots of hardware and compiler optimizations are restricted. Then there arises some relaxed consistency models for more flexible hardware and compiler optimizations. This tutorial paper discusses a series of such relaxed models and compared their implementations. In fact, each of them corresponds to different particular scenarios and none of them can be used as a general model. The drawback of these relaxed models is that they lead to high level of complexity to programmers. Then this tutorial introduces the programmer-centric model which is in the view of programmers instead of systems. This kind of model alleviates the complexity to programmers yet requires mature mechanisms to distinguish memory operations and pass information to the hardware. [1]

2 Merits

To improve the performance, researchers propose the relaxed consistency model to replace the sequential consistency model for practical usage. The intuitive sequential consistency model guarantees the correctness of parallel programs in a very general level and is easy to understand by common programmers. However, to avoid any violations of its strict principles, it has to refuse many hardware optimizations and sacrifice the performance. Many hardware optimizations such as reordering mechanisms of writes and reads on an uniprocessor system cannot be implemented in the sequential consistency model. Therefore, a number of relaxed consistency models have been proposed. This tutorial paper compares some popular relaxed consistency models and summaries them in an uniformed way. In a nutshell, different relaxed consistency models and their implementations correspond to different pragmatic situations. If the programmer can gracefully organize various relaxed consistency models for real problems, the system is able to exploit hardware optimizations to boost the performance obviously. For example, suppose there are two processors P1 (A=1; B=flag2;) and P2 (A=2; C=flag1;) with the initial state (A=flag1=flag2=0). If the programmer just care about the value of B and C, then the order of read and write can be relaxed and the relaxed consistency model can be utilized for optimization.

But sometimes the program order is important since the model which relaxes write and read orders may generate surprising results. Under this circumstance, a mechanism to override program order relaxations becomes necessary. Many relaxed consistency models provide the safety net for such purpose. Programmers can make use of these mechanisms to keep certain sequential consistency.

3 Drawbacks

However, the relaxed consistency model leads to high level complexity on programming. This requires the programmer has very deep understanding on both the model and the real task. It is indeed a disadvantage of the relaxed consistency model. To reduce the burden of the programmer, some researchers proposes the programmer-centric models. However, such kind of model requires mature mechanisms to distinguish memory operations as well as translate information to the hardware.

4 Personal Thoughts

I think the programmer-centric model is a good idea to alleviate programmers' workload. However, due to different real applications and situations, it is difficult to create a generalized model for all problems. As programmers for parallel computing, we should deeply understand how the memory, hardware and compiler work together in multiprocessor systems and what is the correct order to solve a particular problem rather than simply rely on provided programmer-centric models or their implementations.

References

[1] Kourosh Gharachorloo Sarita V.Adve. Shared Memory Consistency Models: A Tutorial. *Rice University ECE Technical Report 9512*, 1995.