Critique of Shared Memory Consistency Models: A Tutorial

Xin Wang

1 Summary

The paper "Shared Memory Consistency Models: A Tutorial" [1] describes and discusses issues about sequential and other more relaxed memory consistency models especially for shared-memory systems based on hardware architecture. The abstraction of shared memory consistency model helps developers have a clear understanding of read/write operations in systems with more than one processors and supplies the specifications for memory system. The model serves as the interface between programmer and system and is a good tool for programmers to check if their programs are right, to provide performance and portability.

The paper first introduces the memory semantics in uniprocessor systems, that is, sequential semantics based on the assumption that all memory operations take place in sequential order. This kind of semantics enables both hardware optimizations and compiler optimizations that enable memory operations' overlapping and reordering. Therefore the sequential semantics in uniprocessor systems make many system simplifications possible and correctness is also guaranteed.

In the following, the concepts of sequential consistency is introduced, that is to maintain sequential program order among operations from all the processors. Sequential consistency provides programmers with a view of a unique gobal memory and a switch for connection between arbitrary processor and memory.

Both architectures with and without caches are described as implementation of abstract sequential consistency model in practical system. The complications of memory operation reordering in latter architecture are caused by conditions such as write buffers with bypassing capability, overlapping write operations and non-blockding read operations. About architecture with caches, cache coherence and sequential consistency, detection of completion of write operations and maintainence of atomicity illusion are key issues. From the discussion we can conclude that sequential consistency constrains a wide range of optimizations. Several relaxed memory models, that are proposed to avoid these constrains for optimizations are introduced next.

One kind of relaxed memory models is to relax the write to read program order, that allows a read operation be reordered to several former write operations from the same processor. This can hide the latency of writes and improve performance at the hardware level. Another kind is to relax the write to read and write to write program orders, that eliminates writes to different locations. The third kind is to relax all program orders so that a read or write operation can be reordered with respect to their following read or write operations to different locations.

Last but not least, an alternate abstraction for relaxed memory model that provides a higher level abstraction for programmers yet allow designers to realize the same kinds of optimizations is introduced. An example programmer-centric framework illustrates the program-level information provided by programmer to enable optimizations similar to the work of weak ordering model. Several mechanisms for conveying information at the programming language level and to the hardware work for this model.

Therefore, compared to sequential consitency model, relaxed memory consitency models enable better hardware optimizations. In addition, the former model play more important role in enabling compiler optimizations. Moreover, the support to relaxed memory models has a good perspect. The disadvantage of complexity in programming has been dealt with by using a higher level abstraction of models. However, choosing the best memory consitency model remains unresolved yet.

2 Positive reactions

This tutorial describes shared memory consistency models clearly and compares different kinds of models in detail with persuasive examples and illustrations. The relaxed consistency models provide many hardware optimizations and have overcome some disadvantages of sequential one. The relaxed memory models provide mechanisms to override program order relaxations, which are good for performance optimization.

3 Negative reactions

There are several limitations of the models introduced in this paper. The performance limitations for hardware optimizations and restrictioins for compiler optimizations related to sequential consistency model are already mentioned in the paper. However, the relaxed memory models which can address some limitations in hardware optimizations also have disadvantages. For example, they are generated at a low level and thus difficult to provide flexibility to higher level language programmers and these relaxed memory models also constrains compiler optimizations.

Also, it is hard to choose a perfect consistency model. For example, a sequential memory consistency model may have the advantage of working efficiently yet has limitations in reordering of memory operations; while a relaxed memory consistency model may allow intuitive reordering of memory operations yet makes its work less efficient because of memory barrier insertion or other behaviors [2].

4 Extension proposal

Since parallel processors become more common and popular than ever before, a better model for shared memory is to ensure both coherence and consistency. So as to provide coherence of memory, the hardware should keep track of the latest verson of particular memory address, do recovery and support communication between processors. Shared memory consistency models have been devised and software synchronization routines have been crafted to provide synchronization. Combining and implementing interaction of coherence, synchronization and consistency can make the job of parallel programming more convenient. [4]

Another extension possibility is to overcome both performance and hardware limitations of both sequential consistency and relaxed memory models. This kind of model can achieve both simple programmability and the implementation flexibility. Several models of this kind have been proposed such as data-race-free, properly labeled models etc. [3]

5 Conclusions

To sum up, the paper provides us with a deep insight into shared memory consistency models. However, since the paper is nearly 20 years from now, some drawbacks such as complexity problem have not been dealt with yet.

References

[1] Sarita V Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. *computer*, 29(12):66–76, 1996.

[2] Colin Blundell, E Christopher Lewis, and Milo MK Martin. Subtleties of transactional memory atomicity semantics. *Computer Architecture Letters*, 5(2):17–17, 2006.

[3] Hans-J Boehm and Sarita V Adve. Foundations of the c++ concurrency memory model. In *ACM SIGPLAN Notices*, volume 43, pages 68–78. ACM, 2008.

[4] Lance Hammond, Vicky Wong, Mike Chen, Brian D Carlstrom, John D Davis, Ben Hertzberg, Manohar K Prabhu, Honggo Wijaya, Christos Kozyrakis, and Kunle Olukotun. Transactional memory coherence and consistency. In *ACM SIGARCH Computer Architecture News*, volume 32, page 102. IEEE Computer Society, 2004.