CS 5341

Oscar Galindo

**Sharing Memory Consistency Models: A Tutorial**

This paper is a summary that explores two memory models (i.e., sequential, and relaxed) and a proposed high-level modeling tool to design memory models with some relaxations to aid performance. The paper begins by studying the importance of memory consistency models. In summary the paper mentions that memory consistency models drive the creation of parallel programs due to the mapping and ordering that takes place from lines of code to machine instructions using compilers. To explore the importance of the mapping and ordering of code lines once translation to machine instruction occurs, the paper first defines what is Sequential Consistency.

As the paper mentions, Sequential Consistency is a memory consistency model that requires “the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.” Furthermore, the paper mentions that two aspects of sequential consistency are “maintaining program order among operations from individual processors and maintaining a single sequential order among operations from all processors.” From there the paper explains how an implementation of Sequential Consistency can occur. Particularly, the paper first mentions three possible optimizations that could be implemented to optimize the memory model for non-cache systems. Then, the paper describes what optimizations are possible on systems that do contain cache memory. In addition, the paper describes some of the necessary conditions so compilers “respect” the optimizations, while at the same time the paper concludes that “that compiler for a shared memory parallel program can not directly apply many common optimizations used in uniprocessor compiler if sequential consistency is to be maintained.”

After providing a summarization for Sequential Consistency models, the paper proceeds to embark on the exploration of Relaxed Memory Models. Quickly, the paper points out that several relaxed consistency models exist, which could provide for several possible reordering (i.e., Write to Read, Write to Write, Read to Read-Write, Read Others’ Write Early, and Read Own Write Early). While for each kind of possible “relaxation” – kind of reordering- the paper offers the advantages and goals, the paper also offers the technique or synchronization constructs that aid at protecting memory consistency when allowing a specific reordering.

Finally, the paper offers some ideas about what to do to construct a specific memory model utilizing abstraction by requiring the user to provide some inputs, like instruction level tags. This last is the main contribution of the paper. In this las section the paper offers as an example the classification of memory operations into synchronizing operations (which provide ordering to the memory model) and data operations (all other operations). Which once completed is provided to the proposed framework to perform the possible relaxations to the execution sequence to aid performance. To successfully perform the classification the paper offers some mechanisms to perform the instruction-level classification and offers some perspective as to what to do when the user is “unsure”.

In my opinion the paper is great at pointing out the different kinds of instructions reordering that exist, it also very thoroughly describes the sequential consistency models and offers some ideas as to how to elevate the level of abstraction to not construct memory models from bare memory operations but rather from the use given to every instruction. However, I think the paper lacks some perspective on what the different consistency models could mean for a user, examples would be of great value to people barely learning about consistency models. In addition, I believe that the paper could benefit by stating common domains where the different consistency models are used.