

Experiments using IBM's Software Transactional Memory Compiler

Barna L. Bihari, ICON Consulting, Inc.

John Gyllenhaal, Scott Futral, and Tom Spelce

Lawrence Livermore National Laboratory

Livermore, California

Scicomp 15 Annual Meeting, May 18-22, 2009, Barcelona, Spain

Abstract

As massively parallel computing evolves well into the era of multi-core shared-memory systems and as the message-passing programming paradigm gets hybridized with threading, there seems to be an explicit need for making thread-safe codes efficient as well. The concept of transactional memory (TM) promises to provide most, if not all, of the attributes of thread-safety but with much reduced barrier and lock times. While pure hardware implementations are sparse or nonexistent, software transactional memory (STM) allows one to experiment with the applicability of TM in a particular code environment.

We present recent results using IBM's software transactional memory C compiler on our benchmark code that intends to mimic the algorithms and code behavior of some very large scientific simulations. In constructing such a code, one needs to make sure that the salient features of the target code are reproduced by the benchmark. Memory conflicts being our main focus here, we generate the conflicts in either a deterministic or random yet still controllable manner. That is, even when the conflicts are random, their probabilities are given by an input parameter. We measure both the number of potential conflicts as well as that of the actual conflicts. The latter will reflect the total number of retries performed by the STM subsystem.

Since it is well-known that all software implementations of TM systems incur a large overhead (see e.g. [1]), pure wall-clock timings are not yet the relevant measure of performance. Instead, our aim is to track the *number of conflicts* as a measure of actual STM usage and compare to more traditional constructs such as locks and barriers. Expecting the performance to be highly problem-dependent, we are especially interested in identifying those algorithms and specific codes for which this novel technology may have a high payoff. This would indeed provide us with the necessary preparation to readily apply it to HTM once it becomes available, at which point the actual run-time would of course become the ultimate bottom line.

Our benchmark C code was compiled on an AIX system using the STM library obtained from IBM's alphaworks website [2]. The results should be systematic enough to be useful to application developers contemplating the use of STM for large, threaded computational physics codes run on IBM systems.

References:

[1] C. Cascaval, C. Blundell, M. Michael, H.W. Cain, P. Wu, S. Chiras, and S. Chatterjee, "Software Transactional Memory: Why is it only a research toy," ACM Queue, pp.46-58, September 2008.

[2] <http://www.alphaworks.ibm.com/tech/xlcstm>