Zhenyu Yan

THREADS CANNOT BE IMPLEMENTED AS A LIBRARY

From

Hans-J. Boehm

In this paper, Boehm believe multithreaded code is written in the originally designed language, with no thread support, and then a thread primitive library has been added. He provides some pure library method, where the compiler is designed independently of threading issues and does not guarantee the correctness of the resulting code. He reviewed why this approach is almost feasible, and then researching this may lead to surprising behavior. He further stated that in a very simple case, the pure library-based approach does not seem to express an efficient parallel algorithm. His discussion was done in the C context with Pthreads because it is usually used, reasonably specified, and does not attempt to ensure type safety.

First Boehm talks about the Pthreads approach to con-currency. They use Pthread_mutex lock(semaphore) to synchronize the code other. This approach clearly works most of the time. Unfortunately, we will see that it is too imprecise to allow the programmer to reason convincingly about program correctness. As a result it has two problems. One is the correct program may fail when using a new compiler or hardware version, or starting to fail, the other one is in many large systems, either intentionally, or unintentionally, violate the multithreading rules. Secondly, he talks about the correctness issues: concurrent modification, rewriting of adjacent data and register promotion. At the end The talks about the performance. Fully Synchronizing a program can actually cause it to run slow because The overhead cost of atomic operations and memory barriers are immense.

For result, Some types of security and security-driven issues become less important. In particular, it is not clear at this time. In the context of type-unsafe languages, much of the work on causality is needed. In some cases, the Java memory model can simply exchange performance, which may not be appropriate in this case. In the case of at least C++ bit fields, the compiler must introduce storage and therefore introduce the possibility of races that do not exist in the source. In modern architectures, this seems likely to be limited to adjacent bitfields.

I think this article is very useful although it is sixteen years ago already. Because multiprocessors are predominant today and in now day, I have heard there already have 24 cores in a computers. So the discussion of multi-threading is important and we still need to modify the implementation to improve the synchronization.

The presenter Adrian Popescu and Logan Woodworth were awesome. Their slide were detail and include some tables and pictures. Also they make two programs and shown as a live demo to demonstrate using single thread and a for loop to sum up numbers is faster and more efficiency than using multithreading to add up numbers. Overall I learn a lot form them.