Keith Schmitt

Professor Trefftz

Final Paper

OpenMP and Threads in Java and C++

Parallel programming is such a powerful concept in modern computing. What comes with this power is great responsibility to the programmer because they not only have to correctly manage synchronous communication between processes or more lightweight threads, but they also intend to have readable and understandable code.

C++ has access to utilities such as pthreads to solve multiprocessing programming issues. With this comes a lot of baggage, utilizing control structures like mutexes, semaphores and shared memory. Java hides the complexity of threads using data encapsulation. The programming language decides to solve multithreading via interface implementation. They have interfaces like Runnable and Thread. These are useful because this helps the writabliity of Java multithreading immensely over the C++ pthread. This still brings more overhead, but at the same time gives more control to the programmer, which could turn out to be more efficient.

OpenMP is so nice because it utilizes #pragma omp parallel for and guarding critical sections by #pragma omp critical, which is absolutely readable with no strange function calls with semaphores and mutexes. It just says almost in plain English, here I want to parallelize my code, and here I have to protect this section of the code respectively. OpenMP is also very portable compared to pthreads of C++, which usually are limited by the operating system. It almost feels as though if I wanted to parallelize code, this would be as simple as placing pragmas in front of code, rather than having to restructure the entire code structure. I have found that code restructuring more often than not creates more unforeseen bugs and headaches and if OpenMP is able to incrementally parallelize certain parts of my programs, and gives me a portable C++ program, then it really is an amazing tool.

Overall I do believe that these are outstanding tools for a programmer to know about, and they make the process of parallelizing code attainable. With simple threads like pthreads in C++, there comes a lot of overhead and code restructure that weigh it down in my opinion, while OpenMP provides parallelization extremely elegantly and does not require this major code restructure.