Alex Esplin
CS 401r
Project 2 Report

Project report for Hot Plate with pthreads.

Goals:

The main goal of this project was to determine if a bit-mask-based thread barrier would be possible, and if it would actually work. While technically still a linear barrier, using an unsigned integer where each thread looks at a single bit determined by it's thread number is in most cases on "normal" computers faster than a log barrier. So the goal was also to have the bit-mask barrier be fast.

Methods:

I didn't go through any extreme effort to write a fast program (no loop-unrolling or other such methods), but I intended to streamline the processing of the hotplate as much as possible by doing the following two things: using a bit-masked unsigned integer as my thread barrier, and not doing any computation that wasn't necessary.

To accomplish the barrier, there is a global variable of type *volatile unsigned int.* Each thread keeps track of its bit of this variable by left-shifting a 1 *iproc* times, where *iproc* is the thread number in the program. The barrier is formed by a busy while loop where the thread spins under the condition *while((GoCheck & myBit) == 0)*. The last thread to reach the barrier sets *nproc* bits of *GoCheck* to 1 by using a left-shift by 1, followed by adding 1, *nproc* times, then assigning that value to *GoCheck*. All threads are then free to go immediately because they are all reading the same integer.

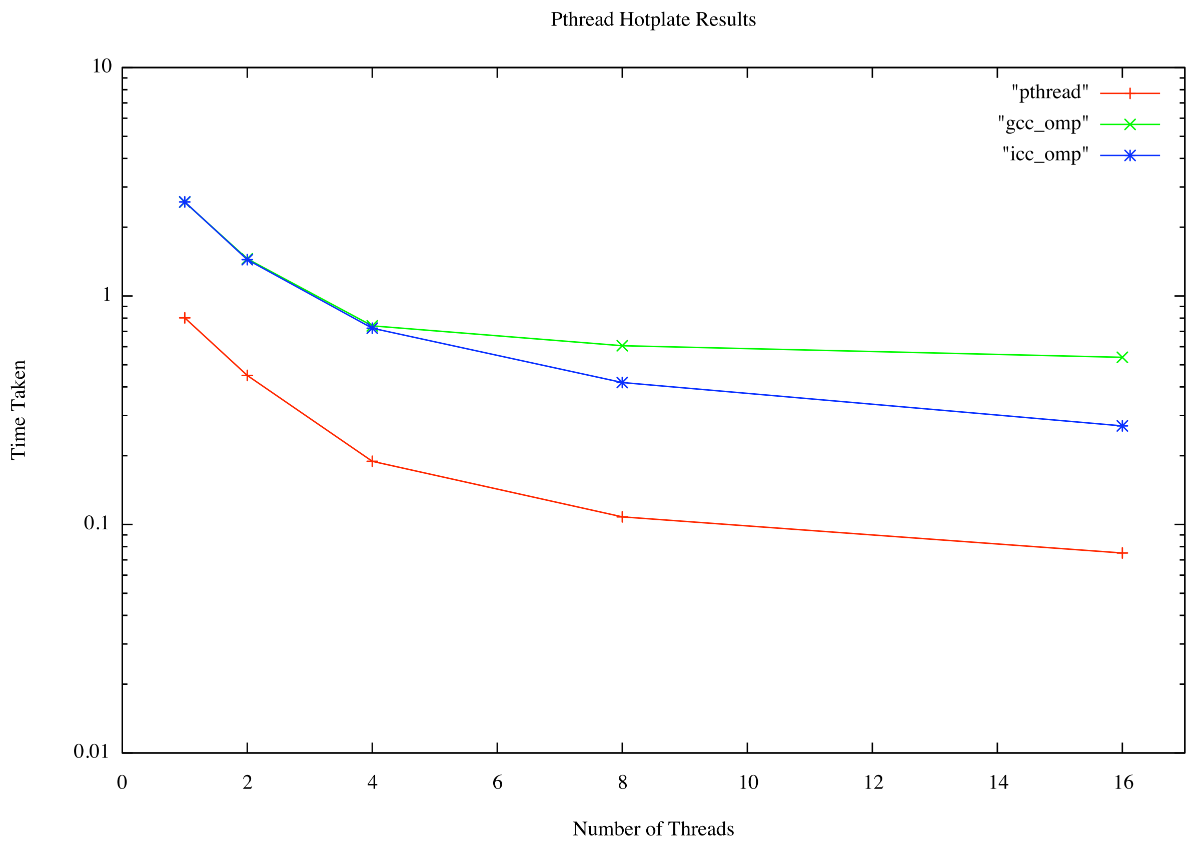
To remove unnecessary computation, each iteration of the checking phase looks to see if the immediate result was unsteady. If it was, the thread breaks out of the checking loop, as there is no further need to continue checking once we know that at least one location is still unsteady.

I considered using a test-and-set mechanism for mutex locking, which would probably have resulted in even greater speedup, but making the bit-mask barrier work took more time than I thought, so I didn't.

Results:

The bit-mask barrier, while tricky to implement, is very fast. In addition, for "normal" shared memory computing (even high-performance workstations have at-most 32 processor cores) the bit-mask would tend to outperform a log barrier due to the fact that if there were enough threads to necessitate using more than one global integer for bit-masks, there would be more threads than processor cores and threads would be sharing cores, resulting in much more of a slow-down than the fact that more than one memory location had to be set.

The pthread implementation of the hotplate was notably faster than either (gcc or icc) of the omp versions. To be entirely objective, the omp versions could have been faster, as the algorithm is noticeably different.



In all, the objective of a very fast barrier using a bit-mask was achieved. There were a few other things that could have been done to streamline and optimize the code, but as the results were completely satisfactory, they were left for a future time.