Claude Shyaka
Cameron Whipple
CSE 539S: Concepts in Multicore Computing
Instructor: Angelina Lee
February 8, 2018

Project 1: Parallelizing Merge Sort

The goal of this project was to implement a parallel merge sort algorithm using Cilk Plus and Pthreads in C. The Cilk Plus merge sort algorithm implemented in this project was similar to the algorithm described in lectures. The same algorithm was used to implement the pthreads merge sort.

The parallelism strategy used for the pthreaded implementation was based on keeping truck of the number of running threads by creating a thread pool. The maximum number of the threads was passed in as an argument, and as long as that many threads were not running, a parent thread would be able to spawn another thread. When the maximum number was reached the execution of the current thread would simply continue without any other thread being spawned, otherwise, a deadlock situation would take place if a forced spawn of a thread was initiated. Mutexes were used to prevent data races, specifically, when updating the number of threads running.

As expected and shown in figure 1, the runtime of the parallel merge sort algorithm decrease as the number of processors increases. However, after eight threads the slope of the graph in figure 1 becomes more or less constant, which means that adding more cores would significantly increase the parallelism. With one core, the Cilk Plus and Pthreads implementation reached an average runtime of 1.13078 and 1.09575 respectively, whereas, at sixteen cores the average runtime was 0.11349 and 0.19946 respectively, approximately an order of magnitude less than the original runtime. Table 1 shows the standard deviation percentage for both the Cilk Plus and Pthreads implementation and it is worth noting that they all fall below 5%.

Results complied by running the serial elision merge sort, and computing the speed up are shown in figure 2 below. The Cilk Plus and Pthreads implementations attained a speed up of 12.67908 and 7.21415 respectively with sixteen cores, and based on these results, a sublinear speed was obtained. In addition, the difference in speedups noted in the two implementations was due to the fact that the Cilk plus scheduler had a better thread scheduling strategy compared to the parallelism strategy used in the Pthread merge sort implementation.

Cameron worked on the implementation of the algorithm using Cilk Plus, and we collectively worked Pthreads portion. The report was written by Claude worked on the report and edited by Cameron.

Table 1: Data collected using and input array of 10^7.

processor_count	cilk_avg_time	cilk_std_dev_percentage	pthread_avg_time	pthread_std_dev_percentage
1	1.130789	0.00%	1.095758	-0.03%
2	0.576448	-0.02%	0.571226	0.00%
4	0.324241	-0.02%	0.342821	-0.05%
8	0.20044	-0.02%	0.245134	-0.03%
16	0.113494	-0.56%	0.199469	-0.05%

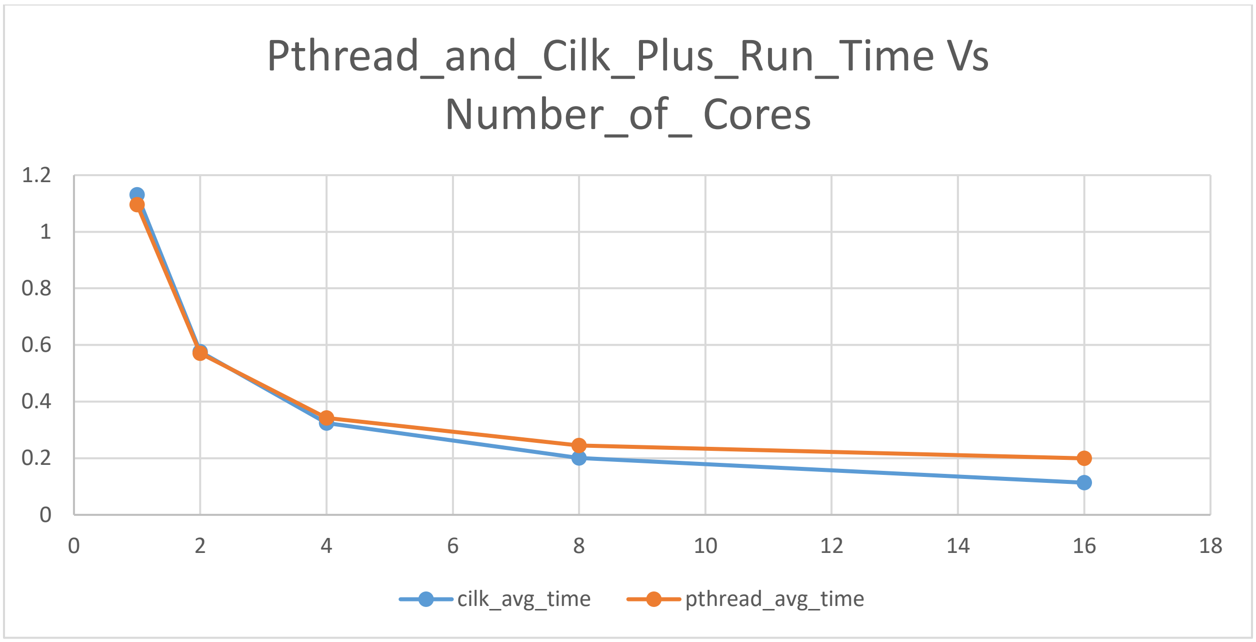

Figure 1: Cilk Plus and Pthreads Runtime vs. Number of Cores.

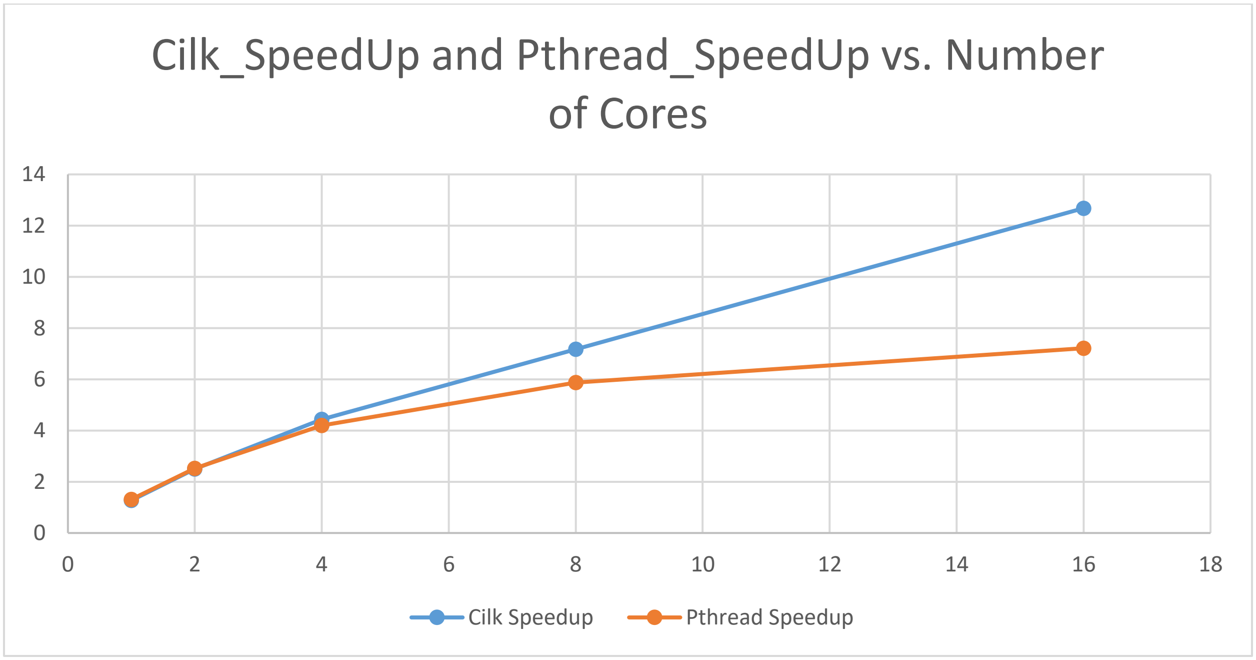

Figure 2: Cilk Plus and Pthreads Speedup vs. Number of Cores.