CSE 5441
Lab 2 Report

Rajarshi Biswas
biswas.91@osu.edu

Table of Contents

Summary of Timing Results:

All the programs were run on 'testgrid_400_12206' test file with an affect Rate of 0.03 and an epsilon of 0.03 (these values were developed in lab1). All the programs converged in '460838' iteration with a max DSV = 0.0850069 and min DSV = 0.0824567. Below are the summaries of the timing results for sequential and both the parallel versions of the program. All the programs were run on stdlinux.

Sequential Version:

Clock	Time	Chrono	Unix Time utility		
			real	user	sys
182560000	212	211993	3m32.059s	3m2.037s	0m0.588s

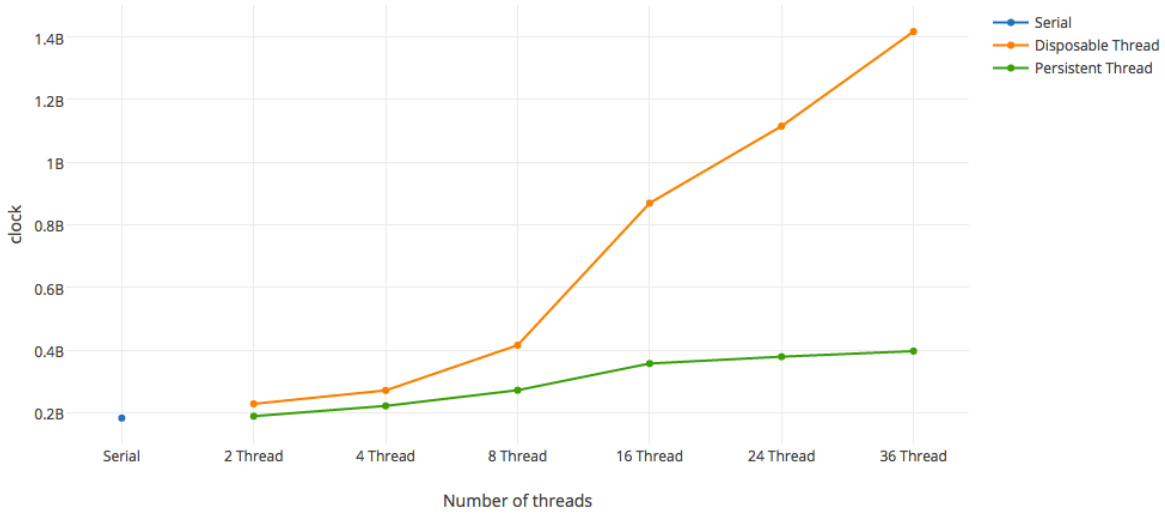
Parallel Version (Disposable Thread Model):

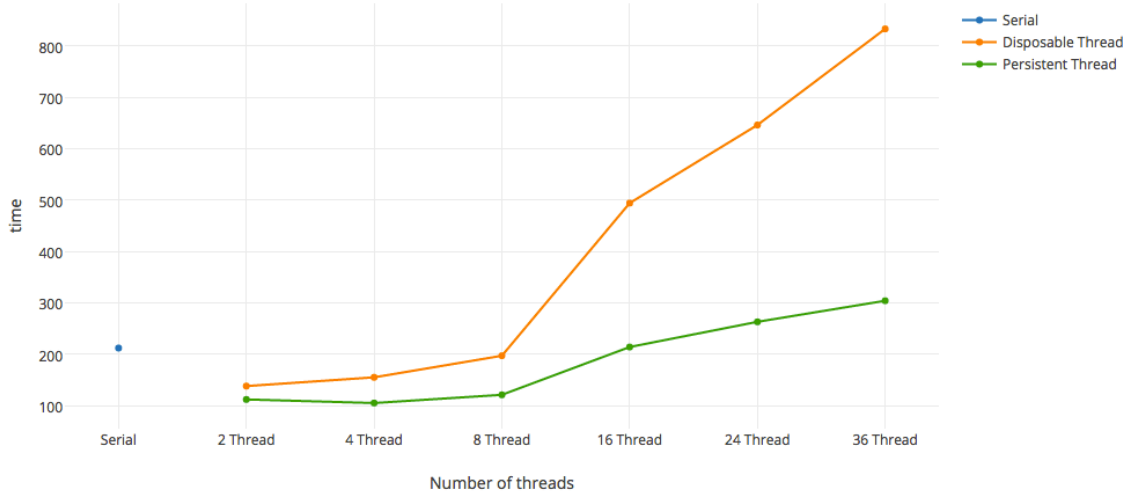
Number of Threads	Clock	Time	Chrono	Unix Time Utility		
				real	user	sys
2	228090000	138	138270	2m18.336s	3m20.646s	0m27.510s
4	270950000	155	154656	2m34.720s	3m33.038s	0m57.973s
8	415400000	197	196765	3m16.829s	4m22.219s	2m33.250s
16	869060000	494	493727	8m13.797s	5m41.668s	8m47.465s
24	1114900000	646	646270	10m46.334s	6m44.873s	11m50.089s
36	1416870000	833	833806	13m53.871s	7m54.331s	15m42.596s

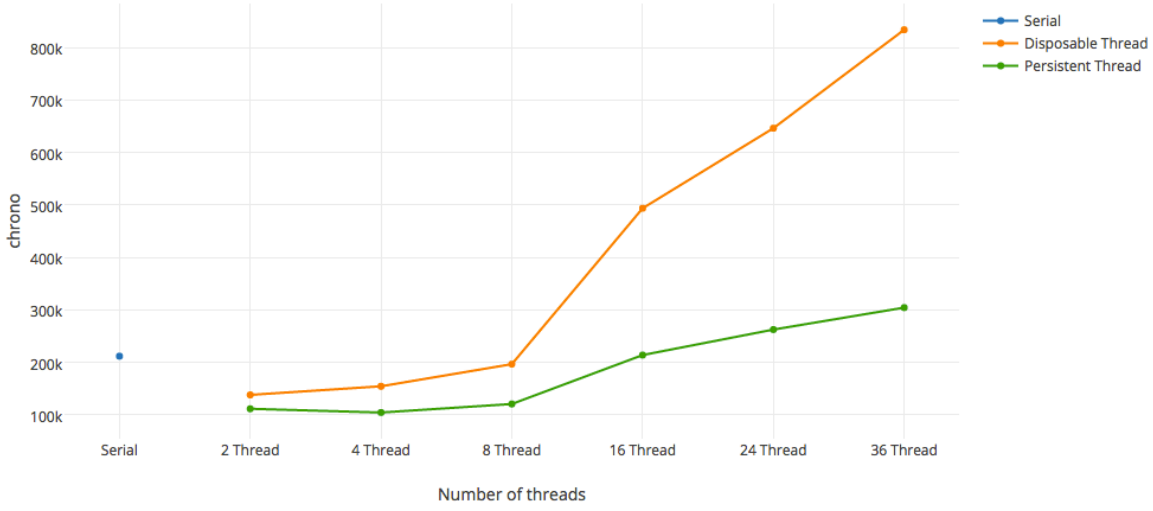
Parallel Version (Persistent Thread Model):

Number of Threads	Clock	Time	Chrono	Unix Time Utility		
				real	user	sys
2	188720000	112	111781	1m51.846s	3m0.389s	0m8.408s
4	221490000	105	104661	1m44.722s	3m17.744s	0m23.801s
8	271640000	121	120907	2m0.969s	3m35.730s	0m55.977s
16	356840000	214	214131	3m34.191s	3m59.867s	1m57.033s
24	378630000	263	262654	4m22.721s	4m11.550s	2m7.147s
36	396480000	304	304505	5m4.567s	4m11.046s	2m25.495s

Comparison of timing results among different versions of the program:







Performance Analysis:

- The program performed better in parallel for a certain number of threads. On the other hand, if I keep on increasing the number of threads, both the parallel versions performed poorly compared to the sequential version.

Effective Thread Number:

- For disposable threading model the effective thread number was 2. However, for 4 and 8 number of threads, the disposable threading model performed better than the sequential one.
- On the other hand, the persistent threading model version performed best with 4 number of threads. This version performed better than the sequential one with 2, 8 number of threads as well. With 16 number of threads it performed almost similar to the sequential version.

Effective Parallel Version:

- From my experiment the persistent thread model parallel version was the most effective compared to the disposable thread model parallel version.

Match with Expectation:

- The persistent thread model performed better than the disposable thread model.
- Both the parallel version performed better than the sequential one for certain number of threads, but performed poorly when we keep on increasing the number of threads. This was because of the increasing overhead of thread creation, join, and synchronization.
- In the Unix Time Utility results, the users time is sometime more than the real time. This is expected as the user time is measured across all the CPUs. More real time than user time indicates more overhead on synchronizing threads.

Conflict with Expectation:

- I expected with an increase in clock ticks the elapsed wall clock time would increase, which was not true. But, as I analyze the timing, it seems, the user time (Unix time utility output) would increase with the clock ticks.

Unexpected Anomalies:

- There were no unexpected anomalies in the timing information collected.

Best Timing Methods for Parallel Programs:

- In my opinion Chrono and Unix (real) time utility seem best for parallel programs.
- I expected the same, as these two provide finer level of granularity and way to measure the real time spent in the computation.