Travis Robinson
CS475
Spring 2016
Project 1
OpenMP: Numeric Integration with OpenMP

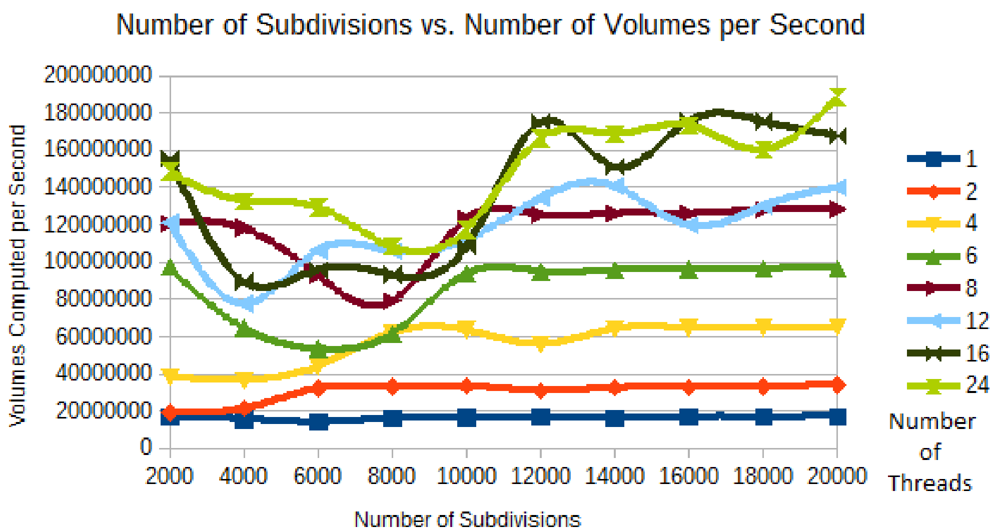
This project was run on the flip servers at OSU

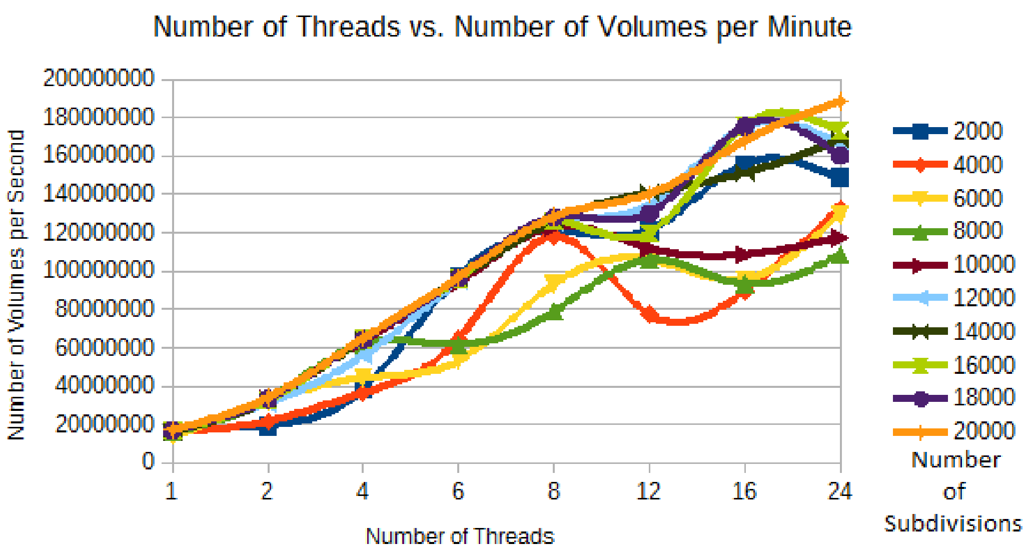
I believe the volume is actually 14. (Or rather a small decimal amount above it)

Charts and Graphs for Project 1

Volumes Computed Per Second

	2000	4000	6000	8000	10000	12000	14000	16000	18000	20000
1	16969675.35	15720587.85	14019824.96	16006844.21	16325932.26	16893591.85	16073567.69	16781753.49	16832042.42	17025980.57
2	19237460.05	21389926	32082689.7	32889951.89	33320606.24	30987264.45	32468719.06	32713002.13	33211500.71	33835606.13
4	38419225.11	36420146.5	44033590.32	61927659.12	63449759.7	55874561.97	64150573.98	64475867.77	64379075.7	64824258.84
6	97268059.06	64569029.38	53280510.88	61431029.2	93620593.77	94876718.72	95647442.64	96161622.52	96432396.21	96745161.3
8	120590014.4	118319789.34	93074512.75	78910413.59	122891333.85	125200122.75	125993236.67	125996817.38	128059227.14	128269646.63
12	121213351.67	77471032.52	106109663.81	105873854.1	111891954.09	134411167.66	140887808.26	119638573.06	130002987.22	140185807.87
16	155821367.2	89510825.08	95743713.71	93129382.75	108373899.49	175002546.09	150984687.35	175514576.93	175521542.94	167838685.19
24	148745284.87	132786516.54	129422557.97	108338357.9	117397362.54	166367820.54	168755358.8	173348811.09	160221016.18	188626594.7





Noticable Patterns and Their Causes

There are two sets of patters in the speeds, depending on whether we are looking from the number of threads or the number of subivisions. When looking at the number of threads, we find that for an increased number of threads, we compute more volumes per second, across all subdivision values. The reason for this is that with multiple threads working there are more threads capable of doing the needed math, so that completing all the computations is able to be done faster.

When looking at the number of subdivisions, we see that the number of volumes calculated remains (relatively) flat across the number of threads. That is, as the number of subdivisions increases, the number of volumes computed remains relatively the same, though more threads do more computing. It was interesting to note that at higher numbers of threads there was more variance in this though. The reason that the number of computations remains flat across the number of threads is that each thread is only capable of doing so much work; increasing the number of subdivisions won't change that. The reason that there is more variance with more cores is that when only using one core, there aren't a whole lot of other uses being made on it. But when using all 24 cores, the computer by necessity has other work to do, other users needing computations done, etc so that not all cores are able to be fully utilized by this program.

<u>Parallel Fraction and Max Speedup</u>

To find the parallel fraction of this application, we first needd to find the speedup. To do this, we we need times for when we use one core and n cores. In this case we'll use n=12. With one thread, we had a time of 245315.69 microseconds. With twelve threads, it was 34524.24 microseconds.

We can then use the formula $\frac{n}{n-1}*(T1-\frac{Tn}{T1})$ to find the parallel portion. Plugging in the times and value for n gives us $\frac{12}{12-1}*(245315.69-\frac{34524.24}{245315.69})=.937381172$

To find the max speedup then, we use the formula $\frac{1}{1-Fparallel}$, which is $\frac{1}{1-.9374}$
This gives us a max speedup of 15.9696