CPSC6770 Assignment 4: Monte Carlo Estimation of Pi using Parallel Computing

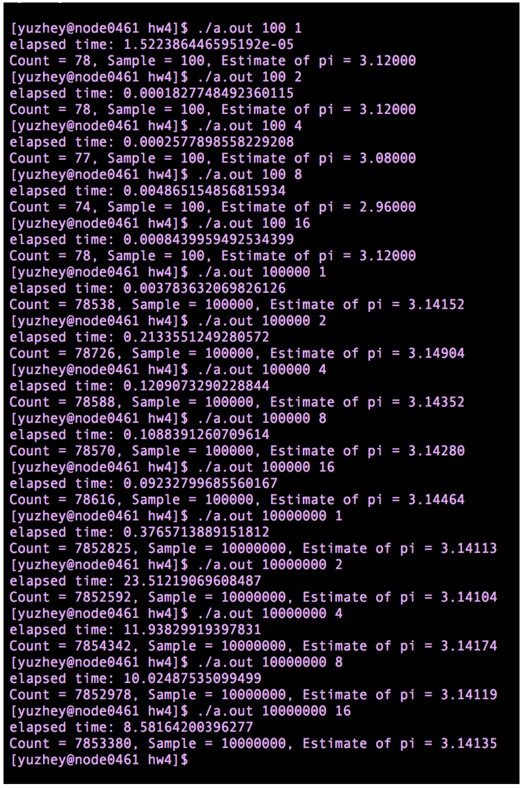
Yuzhe Yang

Task: write the parallelized OpenMP code in either C or Fortran; try to maximize the performance with what you learned in class.

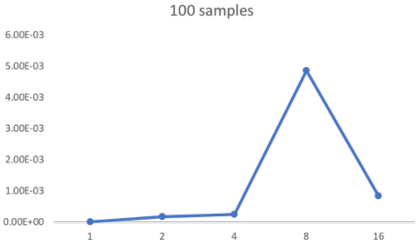
I wrote parallelized OpenMP codes in C and implemented several different methods to test the performance. I requested 16 CPUs in Palmetto working node. The codes can be compiled by command: gcc -fopenmp *codename*; and can be executed by command: ./a.out *sample# thread#*

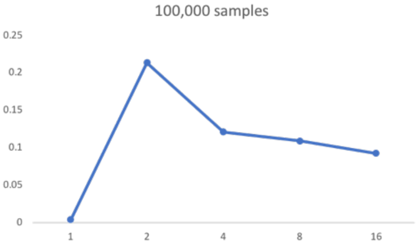
1. OpenMP for loop parallelization in */code/openmp_pi.c*

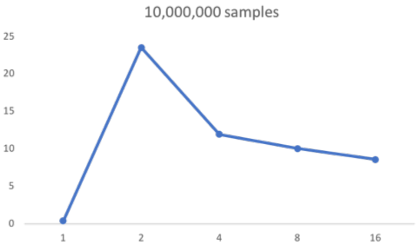
This approach had a significant communication overhead which compromised parallelization performance in both small sample and large sample tasks.

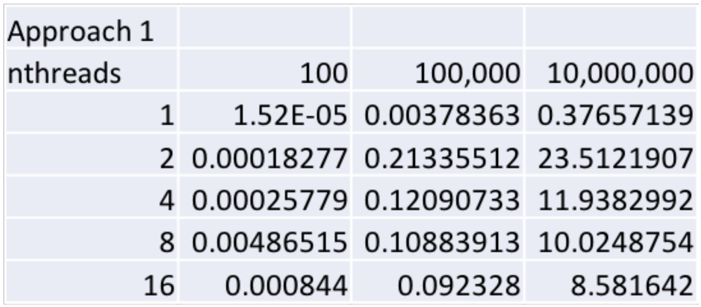


```
[yuzhey@node0461 hw4]$ ./a.out 100 1
elapsed time: 1.522386446595192e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node0461 hw4]$ ./a.out 100 2
elapsed time: 0.0001827748492360115
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node0461 hw4]$ ./a.out 100 4
elapsed time: 0.0002577898558229208
Count = 77, Sample = 100, Estimate of pi = 3.08000
[yuzhey@node0461 hw4]$ ./a.out 100 8
elapsed time: 0.004865154856815934
Count = 74, Sample = 100, Estimate of pi = 2.96000
[yuzhey@node0461 hw4]$ ./a.out 100 16
elapsed time: 0.0008439959492534399
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node0461 hw4]$ ./a.out 100000 1
elapsed time: 0.003783632069826126
Count = 78538, Sample = 100000, Estimate of pi = 3.14152
[yuzhey@node0461 hw4]$ ./a.out 100000 2
elapsed time: 0.2133551249280572
Count = 78726, Sample = 100000, Estimate of pi = 3.14904
[yuzhey@node0461 hw4]$ ./a.out 100000 4
elapsed time: 0.1209073290228844
Count = 78588, Sample = 100000, Estimate of pi = 3.14352
[yuzhey@node0461 hw4]$ ./a.out 100000 8
elapsed time: 0.1088391260709614
Count = 78570, Sample = 100000, Estimate of pi = 3.14280
[yuzhey@node0461 hw4]$ ./a.out 100000 16
elapsed time: 0.09232799685560167
Count = 78616, Sample = 100000, Estimate of pi = 3.14464
[yuzhey@node0461 hw4]$ ./a.out 10000000 1
elapsed time: 0.3765713889151812
Count = 7852825, Sample = 10000000, Estimate of pi = 3.14113
[yuzhey@node0461 hw4]$ ./a.out 10000000 2
elapsed time: 23.51219069608487
Count = 7852592, Sample = 10000000, Estimate of pi = 3.14104
[yuzhey@node0461 hw4]$ ./a.out 10000000 4
elapsed time: 11.93829919397831
Count = 7854342, Sample = 10000000, Estimate of pi = 3.14174
[yuzhey@node0461 hw4]$ ./a.out 10000000 8
elapsed time: 10.02487535099499
Count = 7852978, Sample = 10000000, Estimate of pi = 3.14119
[yuzhey@node0461 hw4]$ ./a.out 10000000 16
elapsed time: 8.58164200396277
Count = 7853380, Sample = 10000000, Estimate of pi = 3.14135
[yuzhey@node0461 hw4]$
```





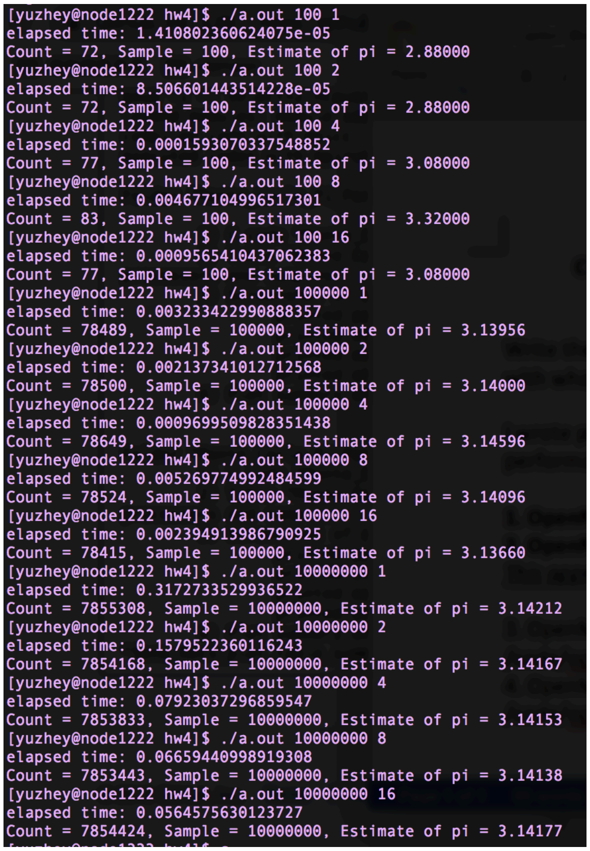




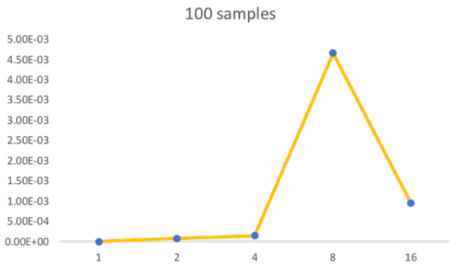
Approach 1			
nthreads	100	100,000	10,000,000
1	1.52E-05	0.00378363	0.37657139
2	0.00018277	0.21335512	23.5121907
4	0.00025779	0.12090733	11.9382992
8	0.00486515	0.10883913	10.0248754
16	0.000844	0.092328	8.581642

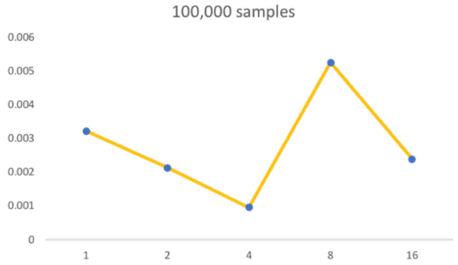
2. OpenMP for loop parallelization with rand_r(&seed) */code/openmp_piseed.c*

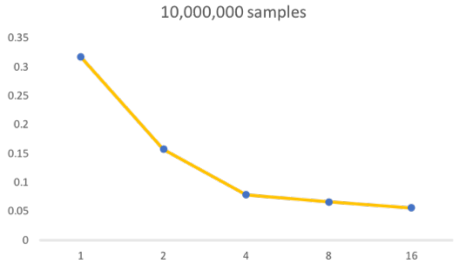
This approach resulted in the best parallel performance in large sample tasks (10,000,000 samples).

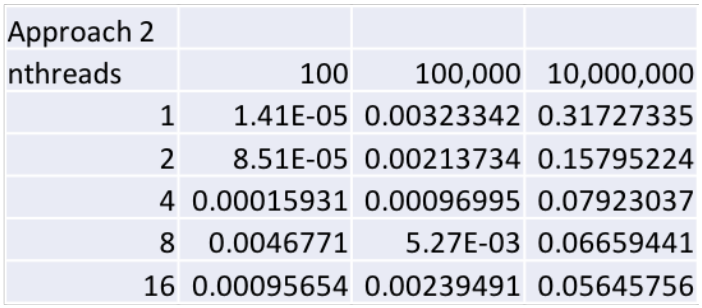


```
[yuzhey@node1222 hw4]$ ./a.out 100 1
elapsed time: 1.410802360624075e-05
Count = 72, Sample = 100, Estimate of pi = 2.88000
[yuzhey@node1222 hw4]$ ./a.out 100 2
elapsed time: 8.506601443514228e-05
Count = 72, Sample = 100, Estimate of pi = 2.88000
[yuzhey@node1222 hw4]$ ./a.out 100 4
elapsed time: 0.0001593070337548852
Count = 77, Sample = 100, Estimate of pi = 3.08000
[yuzhey@node1222 hw4]$ ./a.out 100 8
elapsed time: 0.004677104996517301
Count = 83, Sample = 100, Estimate of pi = 3.32000
[yuzhey@node1222 hw4]$ ./a.out 100 16
elapsed time: 0.0009565410437062383
Count = 77, Sample = 100, Estimate of pi = 3.08000
[yuzhey@node1222 hw4]$ ./a.out 100000 1
elapsed time: 0.003233422990888357
Count = 78489, Sample = 100000, Estimate of pi = 3.13956
[yuzhey@node1222 hw4]$ ./a.out 100000 2
elapsed time: 0.002137341012712568
Count = 78500, Sample = 100000, Estimate of pi = 3.14000
[yuzhey@node1222 hw4]$ ./a.out 100000 4
elapsed time: 0.0009699509828351438
Count = 78649, Sample = 100000, Estimate of pi = 3.14596
[yuzhey@node1222 hw4]$ ./a.out 100000 8
elapsed time: 0.005269774992484599
Count = 78524, Sample = 100000, Estimate of pi = 3.14096
[yuzhey@node1222 hw4]$ ./a.out 100000 16
elapsed time: 0.002394913986790925
Count = 78415, Sample = 100000, Estimate of pi = 3.13660
[yuzhey@node1222 hw4]$ ./a.out 10000000 1
elapsed time: 0.3172733529936522
Count = 7855308, Sample = 10000000, Estimate of pi = 3.14212
[yuzhey@node1222 hw4]$ ./a.out 10000000 2
elapsed time: 0.1579522360116243
Count = 7854168, Sample = 10000000, Estimate of pi = 3.14167
[yuzhey@node1222 hw4]$ ./a.out 10000000 4
elapsed time: 0.07923037296859547
Count = 7853833, Sample = 10000000, Estimate of pi = 3.14153
[yuzhey@node1222 hw4]$ ./a.out 10000000 8
elapsed time: 0.06659440998919308
Count = 7853443, Sample = 10000000, Estimate of pi = 3.14138
[yuzhey@node1222 hw4]$ ./a.out 10000000 16
elapsed time: 0.0564575630123727
Count = 7854424, Sample = 10000000, Estimate of pi = 3.14177
```





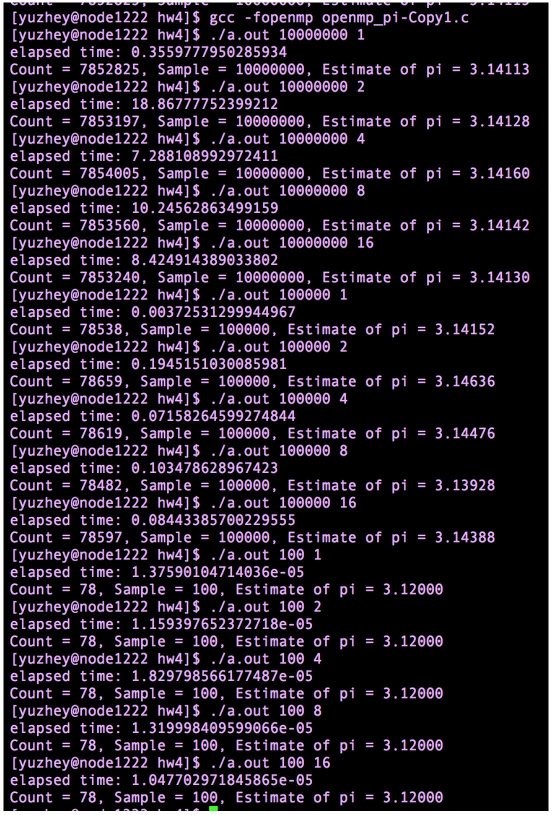




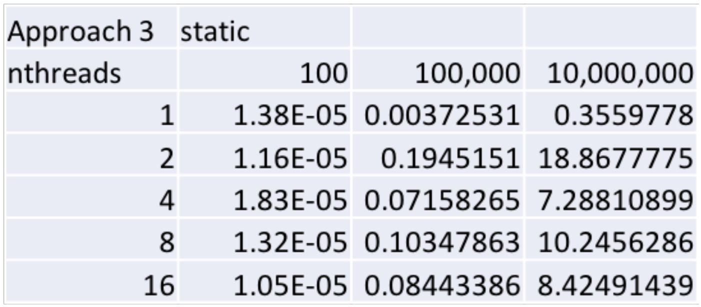
Approach 2			
nthreads	100	100,000	10,000,000
1	1.41E-05	0.00323342	0.31727335
2	8.51E-05	0.00213734	0.15795224
4	0.00015931	0.00096995	0.07923037
8	0.0046771	5.27E-03	0.06659441
16	0.00095654	0.00239491	0.05645756

3. OpenMP for loop parallelization with task threshold and static scheduling /code/openmp_pi_threshold_static.c

In this approach, the code will be executed in the parallel mode only when the number of samples is greater than 100. However, in large sample tasks, the static scheduling strategy still cannot overcome the performance degradation caused by communications between different CPUs.



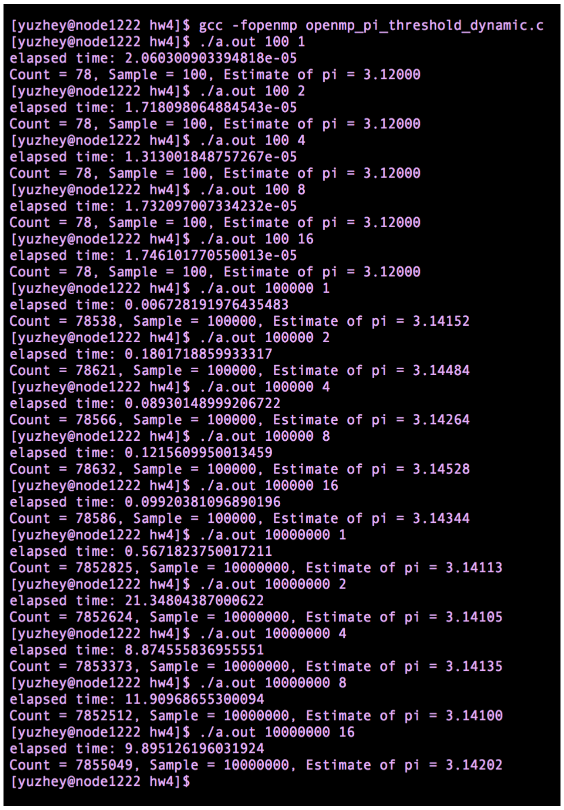
```
[yuzhey@node1222 hw4]$ gcc -fopenmp openmp_pi-Copy1.c
[yuzhey@node1222 hw4]$ ./a.out 10000000 1
elapsed time: 0.3559777950285934
Count = 7852825, Sample = 10000000, Estimate of pi = 3.14113
[yuzhey@node1222 hw4]$ ./a.out 10000000 2
elapsed time: 18.86777752399212
Count = 7853197, Sample = 10000000, Estimate of pi = 3.14128
[yuzhey@node1222 hw4]$ ./a.out 10000000 4
elapsed time: 7.288108992972411
Count = 7854005, Sample = 10000000, Estimate of pi = 3.14160
[yuzhey@node1222 hw4]$ ./a.out 10000000 8
elapsed time: 10.24562863499159
Count = 7853560, Sample = 10000000, Estimate of pi = 3.14142
[yuzhey@node1222 hw4]$ ./a.out 10000000 16
elapsed time: 8.424914389033802
Count = 7853240, Sample = 10000000, Estimate of pi = 3.14130
[yuzhey@node1222 hw4]$ ./a.out 100000 1
elapsed time: 0.00372531299944967
Count = 78538, Sample = 100000, Estimate of pi = 3.14152
[yuzhey@node1222 hw4]$ ./a.out 100000 2
elapsed time: 0.1945151030085981
Count = 78659, Sample = 100000, Estimate of pi = 3.14636
[yuzhey@node1222 hw4]$ ./a.out 100000 4
elapsed time: 0.07158264599274844
Count = 78619, Sample = 100000, Estimate of pi = 3.14476
[yuzhey@node1222 hw4]$ ./a.out 100000 8
elapsed time: 0.103478628967423
Count = 78482, Sample = 100000, Estimate of pi = 3.13928
[yuzhey@node1222 hw4]$ ./a.out 100000 16
elapsed time: 0.08443385700229555
Count = 78597, Sample = 100000, Estimate of pi = 3.14388
[yuzhey@node1222 hw4]$ ./a.out 100 1
elapsed time: 1.37590104714036e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 2
elapsed time: 1.159397652372718e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 4
elapsed time: 1.829798566177487e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 8
elapsed time: 1.319998409599066e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 16
elapsed time: 1.047702971845865e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
```



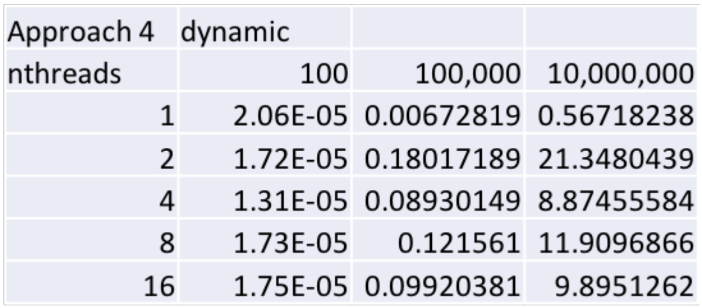
Approach 3	static		
nthreads	100	100,000	10,000,000
1	1.38E-05	0.00372531	0.3559778
2	1.16E-05	0.1945151	18.8677775
4	1.83E-05	0.07158265	7.28810899
8	1.32E-05	0.10347863	10.2456286
16	1.05E-05	0.08443386	8.42491439

4. OpenMP for loop parallelization with task threshold and static scheduling /code/openmp_pi_threshold_dynamic.c

In this approach, the code will be executed in the parallel mode only when the number of samples is greater than 100. However, in large sample tasks, the dynamic scheduling strategy still cannot overcome the performance degradation caused by communications between different CPUs. In my experiments, the dynamic approach had worse performance than the static scheduling approach.



```
[yuzhey@node1222 hw4]$ gcc -fopenmp openmp_pi_threshold_dynamic.c
[yuzhey@node1222 hw4]$ ./a.out 100 1
elapsed time: 2.060300903394818e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 2
elapsed time: 1.718098064884543e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 4
elapsed time: 1.313001848757267e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 8
elapsed time: 1.732097007334232e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100 16
elapsed time: 1.746101770550013e-05
Count = 78, Sample = 100, Estimate of pi = 3.12000
[yuzhey@node1222 hw4]$ ./a.out 100000 1
elapsed time: 0.006728191976435483
Count = 78538, Sample = 100000, Estimate of pi = 3.14152
[yuzhey@node1222 hw4]$ ./a.out 100000 2
elapsed time: 0.1801718859933317
Count = 78621, Sample = 100000, Estimate of pi = 3.14484
[yuzhey@node1222 hw4]$ ./a.out 100000 4
elapsed time: 0.08930148999206722
Count = 78566, Sample = 100000, Estimate of pi = 3.14264
[yuzhey@node1222 hw4]$ ./a.out 100000 8
elapsed time: 0.1215609950013459
Count = 78632, Sample = 100000, Estimate of pi = 3.14528
[yuzhey@node1222 hw4]$ ./a.out 100000 16
elapsed time: 0.09920381096890196
Count = 78586, Sample = 100000, Estimate of pi = 3.14344
[yuzhey@node1222 hw4]$ ./a.out 10000000 1
elapsed time: 0.5671823750017211
Count = 7852825, Sample = 10000000, Estimate of pi = 3.14113
[yuzhey@node1222 hw4]$ ./a.out 10000000 2
elapsed time: 21.34804387000622
Count = 7852624, Sample = 10000000, Estimate of pi = 3.14105
[yuzhey@node1222 hw4]$ ./a.out 10000000 4
elapsed time: 8.874555836955551
Count = 7853373, Sample = 10000000, Estimate of pi = 3.14135
[yuzhey@node1222 hw4]$ ./a.out 10000000 8
elapsed time: 11.90968655300094
Count = 7852512, Sample = 10000000, Estimate of pi = 3.14100
[yuzhey@node1222 hw4]$ ./a.out 10000000 16
elapsed time: 9.895126196031924
Count = 7855049, Sample = 10000000, Estimate of pi = 3.14202
[yuzhey@node1222 hw4]$
```



Approach 4	dynamic		
nthreads	100	100,000	10,000,000
1	2.06E-05	0.00672819	0.56718238
2	1.72E-05	0.18017189	21.3480439
4	1.31E-05	0.08930149	8.87455584
8	1.73E-05	0.121561	11.9096866
16	1.75E-05	0.09920381	9.8951262