CPSC 524: Parallel Programming Techniques

Assignment 2

1. Information on building and running

1.1 Modules

a) Loaded Module List
1) StdEnv (S) 2) Langs/Intel/2015_update2

b) Intel C/C++ Compiler (ICC Version 15.0.2 20150121)
/gpfs/apps/hpc/Langs/Intel/2015_update2/composer_xe_2015.2.164/bin/intel64/icc

1.2 Commands

On the Grace cluster, run the ***make && sbatch run.sh*** script to compile and execute all the programs. You can also run ***make*** to compile them manually. And then run scripts in ***/scripts*** folder to execute each task separately.

1.3 Outputs

The output is as follows:

```
[hs746@grace2 Homework1]$ make && sbatch run.sh
icc -c task1.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task1.o
icc -c drand.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o drand.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task1 task1.o drand.o timing.o
icc -c task2_2.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_2.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_2 task2_2.o drand.o timing.o
icc -c task2_3.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_3.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_3 task2_3.o drand.o timing.o
icc -c task2_4.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_4.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task2_4 task2_4.o drand.o timing.o
icc -c task3_1.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_1.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_1 task3_1.o drand.o timing.o
icc -c task3_2.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_2.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_2 task3_2.o drand.o timing.o
icc -c task3_3.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_3.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task3_3 task3_3.o drand.o timing.o
icc -c task4.c -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task4.o
icc -g -O3 -Wall -xHost -fno-alias -std=c99 -openmp -o task4 task4.o drand.o timing.o
Submitted batch job XXXXXXXX

……Wait for job to be finished……
```

[hs746@grace2 Homework1]$ cat slurm- XXXXXXXX.out

========== **Task 1** ==========
---- *Run Serial Program* ----
area = 1.506678, runtime = 75.852736

========== **Task 2** ==========
---- *Run Program for Threads 1-10* ----
threads = 1
area = 1.506678, runtime = 75.807728
threads = 2
area = 1.506710, runtime = 64.830542
threads = 4
area = 1.506698, runtime = 34.570547
threads = 8
area = 1.506680, runtime = 21.195430
threads = 10
area = 1.506770, runtime = 19.383992
---- *Use Schedule Option* ----
schedule(static,1)
threads = 2
area = 1.506766, runtime = 37.966742
threads = 4
area = 1.506760, runtime = 18.985621
threads = 8
area = 1.506654, runtime = 9.491963
threads = 10
area = 1.506846, runtime = 7.595751
schedule(static,10)
threads = 2
area = 1.506798, runtime = 37.939753
threads = 4
area = 1.506810, runtime = 19.045229
threads = 8
area = 1.506732, runtime = 9.606546
threads = 10
area = 1.506748, runtime = 7.692195
schedule(dynamic)
threads = 2
area = 1.506828, runtime = 37.945300
threads = 4
area = 1.506636, runtime = 18.977053
threads = 8
area = 1.506678, runtime = 9.489214

threads = 10
area = 1.506688, runtime = 7.593668
schedule(dynamic,10)
threads = 2
area = 1.506686, runtime = 37.954031
threads = 4
area = 1.506704, runtime = 18.981671
threads = 8
area = 1.506776, runtime = 9.572211
threads = 10
area = 1.506866, runtime = 7.615146
schedule(guided)
threads = 2
area = 1.506918, runtime = 38.056773
threads = 4
area = 1.506638, runtime = 19.270372
threads = 8
area = 1.506710, runtime = 9.555384
threads = 10
area = 1.506638, runtime = 7.608242
---- *Add Collapse Clause* ----
schedule(static,1), collapse(2)
threads = 2
area = 1.506564, runtime = 38.004552
threads = 4
area = 1.506802, runtime = 19.012299
threads = 8
area = 1.506758, runtime = 9.505091
threads = 10
area = 1.506628, runtime = 7.666156
schedule(dynamic), collapse(2)
threads = 2
area = 1.506782, runtime = 38.021708
threads = 4
area = 1.506768, runtime = 19.028524
threads = 8
area = 1.506566, runtime = 9.529945
threads = 10
area = 1.506686, runtime = 7.631677
schedule(guided), collapse(2)
threads = 2
area = 1.506912, runtime = 38.095900
threads = 4
area = 1.506814, runtime = 19.404521

threads = 8
area = 1.506942, runtime = 9.595585
threads = 10
area = 1.506658, runtime = 7.695481

========== **Task 3** ==========
---- *Each Cell Constitutes A Task* ----
threads = 1
area = 1.506678, runtime = 78.593695
threads = 2
area = 1.506912, runtime = 39.678024
threads = 4
area = 1.506832, runtime = 20.169052
threads = 8
area = 1.506868, runtime = 10.601872
threads = 10
area = 1.506618, runtime = 8.619913
---- *Each Row Constitutes A Task* ----
threads = 1
area = 1.506678, runtime = 78.401246
threads = 2
area = 1.506652, runtime = 39.323353
threads = 4
area = 1.506736, runtime = 19.733650
threads = 8
area = 1.506694, runtime = 9.960598
threads = 10
area = 1.506728, runtime = 8.000352
---- *Task Creation Shared by All Threads* ----
threads = 1
area = 1.506678, runtime = 78.552400
threads = 2
area = 1.506872, runtime = 39.440324
threads = 4
area = 1.506780, runtime = 19.730228
threads = 8
area = 1.506776, runtime = 9.888495
threads = 10
area = 1.506720, runtime = 7.930795

========== **Task 4** ==========
---- Parallel Random Number Generation ----
schedule(dynamic)
area = 1.506968, runtime = 7.599449

2. Task 1: Serial Program

The average result of serial program is 1.506678. The runtime is 75.8214704.

	Runtime	Area
1#	75.807611	1.506678
2#	75.821507	1.506678
3#	75.815351	1.506678
4#	75.810147	1.506678
5#	75.852736	1.506678
Avg.	**75.8214704**	**1.506678**

3. Task 2: OpenMP Program (Loop Directives)

1. Run code for different threads

Thread	**1**	**2**	**4**	**8**	**10**
1# Runtime	75.868256	64.740778	34.586473	21.203252	19.406334
2# Runtime	75.873475	64.762417	34.570941	21.196061	19.384336
3# Runtime	75.858427	64.784809	34.573074	21.193788	19.391196
4# Runtime	75.878767	64.79326	34.579694	21.192667	19.384436
5# Runtime	75.880293	64.736586	34.574722	21.193138	19.388773
Avg. Runtime	**75.8718436**	**64.76357**	**34.5769808**	**21.1957812**	**19.391015**
1# Area	1.506678	1.50671	1.506698	1.50668	1.50677
2# Area	1.506678	1.50671	1.506698	1.50668	1.50677
3# Area	1.506678	1.50671	1.506698	1.50668	1.50677
4# Area	1.506678	1.50671	1.506698	1.50668	1.50677
5# Area	1.506678	1.50671	1.506698	1.50668	1.50677
Avg. Area	**1.506678**	**1.50671**	**1.506698**	**1.50668**	**1.50677**

2. Use schedule option

a). schedule(static, 1)

Thread	**2**	**4**	**8**	**10**
1# Runtime	38.046780	18.992714	9.496117	7.594300
2# Runtime	38.025757	18.989164	9.498191	7.605892
3# Runtime	37.999203	18.976825	9.493757	7.599185
4# Runtime	37.976222	18.976478	9.487984	7.593928
5# Runtime	37.947175	18.969194	9.488119	7.594220
Avg. Runtime	**37.999027**	**18.980875**	**9.492834**	**7.597505**
1# Area	1.506766	1.50676	1.506654	1.506846
2# Area	1.506766	1.50676	1.506654	1.506846

3# Area	1.506766	1.50676	1.506654	1.506846
4# Area	1.506766	1.50676	1.506654	1.506846
5# Area	1.506766	1.50676	1.506654	1.506846
Avg. Area	**1.506766**	**1.50676**	**1.506654**	**1.506846**

b). schedule(static, 10)

Thread	**2**	**4**	**8**	**10**
1# Runtime	37.956579	19.05593	9.609695	7.695954
2# Runtime	37.974703	19.043588	9.607437	7.700020
3# Runtime	37.948666	19.056125	9.610510	7.697621
4# Runtime	37.942704	19.050209	9.607668	7.697779
5# Runtime	37.964051	19.055558	9.610277	7.696233
Avg. Runtime	**37.957341**	**19.052282**	**9.609117**	**7.697521**
1# Area	1.506798	1.50681	1.506732	1.506748
2# Area	1.506798	1.50681	1.506732	1.506748
3# Area	1.506798	1.50681	1.506732	1.506748
4# Area	1.506798	1.50681	1.506732	1.506748
5# Area	1.506798	1.50681	1.506732	1.506748
Avg. Area	**1.506798**	**1.50681**	**1.506732**	**1.506748**

c). schedule(dynamic)

Thread	**2**	**4**	**8**	**10**
1# Runtime	37.955135	18.970175	9.492658	7.593728
2# Runtime	37.947522	18.977528	9.489235	7.594256
3# Runtime	37.956382	18.983681	9.494589	7.599565
4# Runtime	37.935027	18.974133	9.489602	7.593402
5# Runtime	37.951165	18.982489	9.492900	7.595261
Avg. Runtime	**37.949046**	**18.977601**	**9.491797**	**7.595242**
1# Area	1.506902	1.506648	1.506770	1.506860
2# Area	1.506864	1.506864	1.506812	1.506856
3# Area	1.506738	1.506868	1.506824	1.506812
4# Area	1.506760	1.506772	1.506700	1.506702
5# Area	1.506694	1.506686	1.506552	1.506644
Avg. Area	**1.506792**	**1.506768**	**1.506732**	**1.506775**

d). schedule(dynamic, 10)

Thread	**2**	**4**	**8**	**10**
1# Runtime	37.945912	18.976414	9.571752	7.617556
2# Runtime	37.945990	18.973250	9.569291	7.604855
3# Runtime	37.950324	18.981225	9.581376	7.612417

4# Runtime	37.925187	18.96789	9.578490	7.603822
5# Runtime	37.952543	18.977249	9.574698	7.611535
Avg. Runtime	**37.943991**	**18.975206**	**9.575121**	**7.610037**
1# Area	1.506944	1.506880	1.506756	1.506780
2# Area	1.506654	1.506768	1.506786	1.506818
3# Area	1.506694	1.506756	1.506812	1.506724
4# Area	1.506664	1.506704	1.506750	1.506656
5# Area	1.506928	1.506766	1.506696	1.506816
Avg. Area	**1.506777**	**1.506775**	**1.506760**	**1.506759**

e). schedule(guided)

Thread	**2**	**4**	**8**	**10**
1# Runtime	38.016159	19.260652	9.560724	7.616599
2# Runtime	38.012091	19.277559	9.559698	7.607061
3# Runtime	38.048616	19.285970	9.564160	7.610609
4# Runtime	38.004803	19.271879	9.559598	7.606690
5# Runtime	38.008488	19.267668	9.565806	7.608576
Avg. Runtime	**38.018031**	**19.272746**	**9.561997**	**7.609907**
1# Area	1.506918	1.506658	1.506642	1.506720
2# Area	1.506918	1.506646	1.506654	1.506672
3# Area	1.506918	1.506712	1.506646	1.506626
4# Area	1.506918	1.506712	1.506656	1.506648
5# Area	1.506918	1.506646	1.506652	1.506568
Avg. Area	**1.506918**	**1.506675**	**1.50665**	**1.506647**

Overall Performance

a. schedule(static,1)	37.9990274	18.980875	9.4928336	7.5975050
b. schedule(static,10)	37.9573406	19.052282	9.6091174	7.6975214
c. schedule(dynamic)	37.9490462	18.977601	**9.4917968**	**7.5952424**
d. schedule(dynamic,10)	**37.9439912**	**18.975206**	9.5751214	7.6100370
e. schedule(guided)	38.0180314	19.272746	9.5619972	7.6099070

3 Add collapse clause

1). schedule(static,1) + collapse(2):

Thread	**2**	**4**	**8**	**10**
1# Runtime	38.030193	19.004396	9.502254	7.665978
2# Runtime	38.03555	19.003586	9.501232	7.663411
3# Runtime	38.000439	19.004179	9.505711	7.665925
4# Runtime	38.05242	19.00955	9.505691	7.666408
5# Runtime	38.05305	19.016706	9.507274	7.663491

Avg. Runtime	**38.0343304**	**19.0076834**	**9.5044324**	**7.6650426**
1# Area	1.506564	1.506802	1.506758	1.506628
2# Area	1.506802	1.506758	1.506628	1.506514
3# Area	1.506564	1.506802	1.506758	1.506628
4# Area	1.506564	1.506802	1.506758	1.506628
5# Area	1.506564	1.506802	1.506758	1.506628
Avg. Area	**1.5066116**	**1.5067932**	**1.506732**	**1.5066052**

2). schedule(dynamic) + collapse(2):

Thread	**2**	**4**	**8**	**10**
1# Runtime	37.987512	19.009146	9.526192	7.633396
2# Runtime	37.986509	19.004836	9.52635	7.634566
3# Runtime	38.000129	19.013409	9.526588	7.628109
4# Runtime	38.005173	19.017042	9.530838	7.63507
5# Runtime	37.993041	19.01869	9.526628	7.635402
Avg. Runtime	**37.9944728**	**19.0126246**	**9.5273192**	**7.6333086**
1# Area	1.506912	1.506732	1.506836	1.50674
2# Area	1.506514	1.506754	1.506834	1.506732
3# Area	1.50688	1.50682	1.506992	1.506688
4# Area	1.506764	1.506852	1.5069	1.506828
5# Area	1.506838	1.506726	1.506818	1.5068
Avg. Area	**1.5067816**	**1.5067768**	**1.506876**	**1.5067576**

3). schedule(guided) + collapse(2):

Thread	**2**	**4**	**8**	**10**
1# Runtime	38.099894	19.392383	9.596122	7.687025
2# Runtime	38.094905	19.40308	9.594289	7.692484
3# Runtime	38.07885	19.389661	9.591017	7.686615
4# Runtime	38.111126	19.403681	9.599155	7.690554
5# Runtime	38.084177	19.401357	9.595852	7.684202
Avg. Runtime	**38.0937904**	**19.3980324**	**9.595287**	**7.688176**
1# Area	1.506912	1.506814	1.506878	1.506578
2# Area	1.506912	1.506818	1.506966	1.50659
3# Area	1.506912	1.506814	1.506942	1.506606
4# Area	1.506912	1.506818	1.506946	1.506568
5# Area	1.506912	1.506802	1.506884	1.506748
Avg. Area	**1.506912**	**1.5068132**	**1.5069232**	**1.506618**

It doesn’t make much of a performance difference when we add collapse(2) clause.

4. Task 3: OpenMP Program (Tasks)

3.1 Each cell constitutes a task

Thread	**1**	**2**	**4**	**8**	**10**
1# Runtime	78.522985	39.635178	20.123727	10.511971	8.655112
2# Runtime	78.557128	39.649584	20.123467	10.578675	8.689065
3# Runtime	78.552725	39.66335	20.172853	10.570294	8.591436
4# Runtime	78.608824	39.640204	20.146664	10.579267	8.622321
5# Runtime	78.629803	39.638696	20.176296	10.502333	8.645302
Avg. Runtime	**78.574293**	**39.6454024**	**20.1486014**	**10.548508**	**8.6406472**
1# Area	1.506678	1.50673	1.506944	1.50678	1.506734
2# Area	1.506678	1.506802	1.50669	1.5068	1.506846
3# Area	1.506678	1.506736	1.506816	1.506642	1.50667
4# Area	1.506678	1.50683	1.506684	1.506896	1.506754
5# Area	1.506678	1.506738	1.506854	1.50668	1.506678
Avg. Area	**1.506678**	**1.5067672**	**1.5067976**	**1.5067596**	**1.5067364**

3.2 Each row constitutes a task

Thread	**1**	**2**	**4**	**8**	**10**
1# Runtime	78.362224	39.325157	19.750444	9.958163	8.00409
2# Runtime	78.398539	39.289046	19.727296	9.949502	8.005827
3# Runtime	78.359297	39.268969	19.724556	9.949776	7.998325
4# Runtime	78.360324	39.331486	19.762247	9.957959	8.000045
5# Runtime	78.437284	39.319343	19.717844	9.957242	7.99762
Avg. Runtime	**78.3835336**	**39.3068002**	**19.7364774**	**9.9545284**	**8.0011814**
1# Area	1.506678	1.506542	1.506796	1.506622	1.50674
2# Area	1.506678	1.506508	1.506594	1.506716	1.506816
3# Area	1.506678	1.506702	1.506684	1.506756	1.50665
4# Area	1.506678	1.506534	1.506664	1.506748	1.506652
5# Area	1.506678	1.506678	1.506614	1.506764	1.506846
Avg. Area	**1.506678**	**1.5065928**	**1.5066704**	**1.5067212**	**1.5067408**

3.3 Task creation is shared by all the threads

Thread	**1**	**2**	**4**	**8**	**10**
1# Runtime	78.575856	39.418279	19.720701	9.88056	7.926114
2# Runtime	78.562937	39.40869	19.73277	9.879576	7.928711
3# Runtime	78.541956	39.412228	19.718801	9.882719	7.926758
4# Runtime	78.650306	39.431859	19.724186	9.884766	7.923911
5# Runtime	78.627489	39.40426	19.717239	9.878992	7.91916
Avg. Runtime	**78.5917088**	**39.4150632**	**19.7227394**	**9.8813226**	**7.9249308**

1# Area	1.506678	1.506812	1.506744	1.506598	1.506772
2# Area	1.506678	1.506846	1.506712	1.506638	1.506726
3# Area	1.506678	1.506922	1.50675	1.506644	1.506674
4# Area	1.506678	1.506986	1.50677	1.506696	1.506754
5# Area	1.506678	1.5069	1.506774	1.506564	1.506624
Avg. Area	**1.506678**	**1.5068932**	**1.50675**	**1.506628**	**1.50671**

3.4 Analyze

From the experiment results we can know that all cell having one corresponding task has worst performance, and each row having one corresponding task has better performance and create tasks by all threads has best performance. The reason is that if one single thread creates all threads for all cells, there will be a lot of time wasted on the creation of tasks. By change to create one task for one row, we reduce the number of tasks needed to be created, thus we can have better performance. And if we create tasks using all thread, this will be somehow similar to for loop directives in task 2. Each thread will be responsible for both generating and executing tasks, thus it is equivalent to each thread executes parts of the problem. However, each thread still wastes some time on produce tasks into the task pool, and execute task from that pool, which makes the overall performance worse than what we got in task 2.

5. Task 4: Parallel Random Number Generation

The initial idea of parallel random number generator is that we set the seed of each thread as different value, such as the thread number. Another way I used is that we still use one sequence of random number. However, we set a different offset for each thread on that sequence. In the case, each thread will use its own random number starting from different position at that random sequence. The performance doesn't change much since in task2, when we use thread private seed to make program thread safe, we actually set each thread the same seed. The only difference in this task will be each thread starting from different seed. Thus, there shouldn't be much performance difference.

	New drand	**Original drand**
1# Runtime	7.605032	7.617556
2# Runtime	7.607843	7.604855
3# Runtime	7.597024	7.612417
4# Runtime	7.605769	7.603822
5# Runtime	7.599108	7.611535
Avg. Runtime	**7.6029552**	**7.610037**
1# Area	1.50669	1.506780
2# Area	1.506648	1.506818
3# Area	1.50662	1.506724
4# Area	1.506674	1.506656
5# Area	1.506642	1.506816
Avg. Area	**1.5066548**	**1.506759**