Spring 2022: Advanced Topics in Numerical Analysis: High Performance Computing
Homework 2
https://github.com/anmolsinghal/HPCHW2

Submitted by: Anmol Singhal
as15151@nyu.edu

1. **Finding Memory bugs.**
   Code attached for val_test1_solved.cpp and val_test2_solved.cpp. Make and run using

```
make val_test01
./val_test01_solved
```

   or

```
make val_test02
./val_test02_solved
```

2. **Optimizing Matrix Matrix multiplication** Code attached in MMult1.cpp

Figure 1: MMult0 base line

Dimension	Time	Gflop/s	GB/s	Error
4	0.427137	23.411700	74.917441	0.000000e+00
52	0.221956	45.054143	144.173259	0.000000e+00
100	0.248092	40.347989	129.113564	0.000000e+00
148	0.256109	39.112766	125.160851	0.000000e+00
196	0.283168	35.365140	113.168447	0.000000e+00
244	0.287803	34.827521	111.448067	0.000000e+00
292	0.285660	35.734156	114.349298	0.000000e+00
340	0.288224	35.455199	113.456637	0.000000e+00
388	0.286808	36.658617	117.307574	0.000000e+00
436	0.318150	33.866546	108.372946	0.000000e+00
484	0.288392	35.383046	113.225746	0.000000e+00

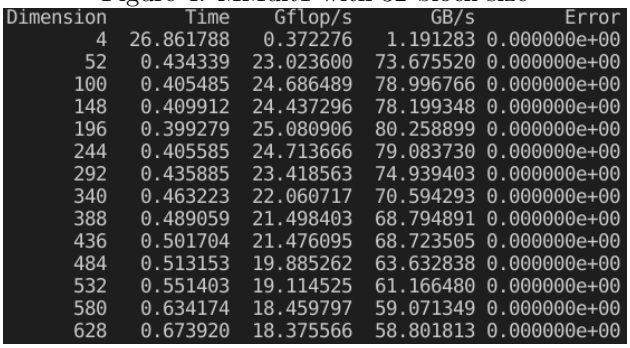
Figure 2: MMult1 with 4 block size

Dimension	Time	Gflop/s	GB/s	Error
4	0.968811	10.321934	33.030189	0.000000e+00
52	0.987843	10.123103	32.393931	0.000000e+00
100	1.165737	8.586843	27.477899	0.000000e+00
148	1.267056	7.905834	25.298669	0.000000e+00
196	1.339789	7.474521	23.918468	0.000000e+00
244	1.360477	7.367622	23.576390	0.000000e+00
292	1.408357	7.248025	23.193680	0.000000e+00
340	1.421886	7.186959	22.998270	0.000000e+00
388	1.482537	7.091894	22.694061	0.000000e+00
436	1.542090	6.987038	22.358523	0.000000e+00
484	1.492956	6.834892	21.871656	0.000000e+00
532	1.564087	6.738636	21.563635	0.000000e+00

Figure 3: MMult1 with 16 block size

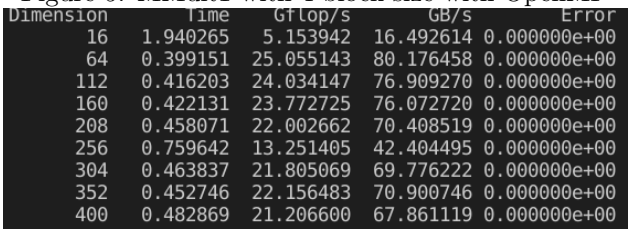
Dimension	Time	Gflop/s	GB/s	Error
16	0.843765	11.851663	37.925323	0.000000e+00
64	1.106781	9.035929	28.914974	0.000000e+00
112	1.298001	7.706535	24.660911	0.000000e+00
160	1.367512	7.338291	23.482533	0.000000e+00
208	1.405590	7.170498	22.945594	0.000000e+00
256	1.886625	5.335628	17.074009	0.000000e+00
304	1.437548	7.035594	22.513900	0.000000e+00
352	1.446722	6.933789	22.188124	0.000000e+00
400	1.470626	6.963021	22.281666	0.000000e+00
448	1.593184	6.772505	21.672016	0.000000e+00

Figure 4: MMult1 with 32 block size



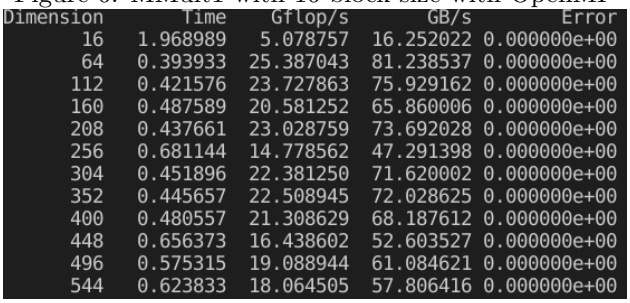
Dimension	Time	Gflop/s	GB/s	Error
4	26.861788	0.372276	1.191283	0.000000e+00
52	0.434339	23.023600	73.675520	0.000000e+00
100	0.405485	24.686489	78.996766	0.000000e+00
148	0.409912	24.437296	78.199348	0.000000e+00
196	0.399279	25.080906	80.258899	0.000000e+00
244	0.405585	24.713666	79.083730	0.000000e+00
292	0.435885	23.418563	74.939403	0.000000e+00
340	0.463223	22.060717	70.594293	0.000000e+00
388	0.489059	21.498403	68.794891	0.000000e+00
436	0.501704	21.476095	68.723505	0.000000e+00
484	0.513153	19.885262	63.632838	0.000000e+00
532	0.551403	19.114525	61.166480	0.000000e+00
580	0.634174	18.459797	59.071349	0.000000e+00
628	0.673920	18.375566	58.801813	0.000000e+00

Figure 5: MMult1 with 4 block size with OpenMP



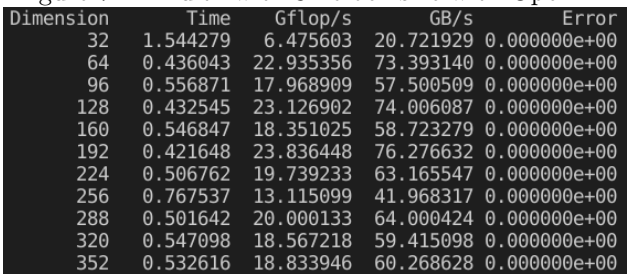
Dimension	Time	Gflop/s	GB/s	Error
16	1.940265	5.153942	16.492614	0.000000e+00
64	0.399151	25.055143	80.176458	0.000000e+00
112	0.416203	24.034147	76.909270	0.000000e+00
160	0.422131	23.772725	76.072720	0.000000e+00
208	0.458071	22.002662	70.408519	0.000000e+00
256	0.759642	13.251405	42.404495	0.000000e+00
304	0.463837	21.805069	69.776222	0.000000e+00
352	0.452746	22.156483	70.900746	0.000000e+00
400	0.482869	21.206600	67.861119	0.000000e+00

Figure 6: MMult1 with 16 block size with OpenMP



Dimension	Time	Gflop/s	GB/s	Error
16	1.968989	5.078757	16.252022	0.000000e+00
64	0.393933	25.387043	81.238537	0.000000e+00
112	0.421576	23.727863	75.929162	0.000000e+00
160	0.487589	20.581252	65.860006	0.000000e+00
208	0.437661	23.028759	73.692028	0.000000e+00
256	0.681144	14.778562	47.291398	0.000000e+00
304	0.451896	22.381250	71.620002	0.000000e+00
352	0.445657	22.508945	72.028625	0.000000e+00
400	0.480557	21.308629	68.187612	0.000000e+00
448	0.656373	16.438602	52.603527	0.000000e+00
496	0.575315	19.088944	61.084621	0.000000e+00
544	0.623833	18.064505	57.806416	0.000000e+00

Figure 7: MMult1 with 32 block size with OpenMP



Dimension	Time	Gflop/s	GB/s	Error
32	1.544279	6.475603	20.721929	0.000000e+00
64	0.436043	22.935356	73.393140	0.000000e+00
96	0.556871	17.968909	57.500509	0.000000e+00
128	0.432545	23.126902	74.006087	0.000000e+00
160	0.546847	18.351025	58.723279	0.000000e+00
192	0.421648	23.836448	76.276632	0.000000e+00
224	0.506762	19.739233	63.165547	0.000000e+00
256	0.767537	13.115099	41.968317	0.000000e+00
288	0.501642	20.000133	64.000424	0.000000e+00
320	0.547098	18.567218	59.415098	0.000000e+00
352	0.532616	18.833946	60.268628	0.000000e+00

3. **Finding OpenMP bugs**

omp_bug2.c - Since total is to be shared, it should be declared outside the parallel block and used as reduction variable in for loop

omp_bug3.c - there are barriers inside the function call. So the threads which execute the section keep waiting for other threads to join. One simple solution is to make sure there are only 2 threads executing this. Otherwise we can remove some uneccessary barriers

omp_bug4.c - tid and a are private to each thread. Hence a needs to be declared inside the thread block in the heap to prevent stack overflow.

omp_bug5.c - to prevent deadlock, the locks must be locked in same order in every critical section. This ensure the processes dont lock locks out of order and starve each other into deadlock.

omp_bug6.c - we remove parallel pragma in main function and add the requisite parallel block in dotprod function. This is because we want to calculate dotprod in parallel and not calculate multiple dot prods in parallel

4. **OpenMP version of 2D Jacobi/Gauss-Seidel smoothing**

    1. **Processor:** Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
    2. **Compiler:** gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1 20.04)
    3. Make and run code using

        ```
        make [gaussSeidel | jacobi]
        ./[gs2d-omp | jacobi2D-omp] <dimension> <iterations> <threads>
        ```

    4. Timing report with 100 iterations on 2d smoothing problem using OpenMP with k = threads, N= dimension

        (a) N:100 K:1
        Total time taken by Jacobi: 0.02803
        Total time taken by Gauss Seidl: 0.01228s

        (b) N:100 K:2
        Total time taken by Jacobi: 0.01647s
        Total time taken by Gauss Seidl: 0.06020s

        (c) N:100 K:3
        Total time taken by Jacobi: 0.00376s
        Total time taken by Gauss Seidl: 0.10701s

        (d) N:100 K:4
        Total time taken by Jacobi: 0.05788s
        Total time taken by Gauss Seidl: 0.13607s

        (e) N:100 K:8
        Total time taken by Jacobi: 0.06660s
        Total time taken by Gauss Seidl: 0.13470s

        (f) N:100 K:16
        Total time taken by Jacobi: 0.07598s
        Total time taken by Gauss Seidl: 0.10641s