Homework 2

Cong Yu

March 13, 2020

github:

```
https://github.com/heliumyc/hpc_homework
```

1 Finding Memory bugs

For val_test01, there are two errors:

1. delete is used which mismatches malloc and it should be free
2. it allocates int array of size n but tris to read and write to position n+1 (index n).

For val_test02, the program tries to use uninitialized data in a newed array.

2 Optimizing matrix-matrix multiplication

Machine information:

```
Architecture:  x86_64
Processor Name: 6-Core Intel Core i7
Processor Speed: 2.2 GHz
Number of processors: 1
Total number of cores: 6
L2 cache (per core): 256 KB
L3 cache: 9 MB
Hyper-Threading Technology: Enabled
Memory: 16 GB
```

In MMult1, the best order is supposed to be j-p-i. Because the CPU fetches data via cache line, the memory access time is reduced if we read array data sequentially.

The best BLOCK SIZE should be 48 or 64 (both reach the roughly same optimal time)

FLOP-percentage is at peak $44.8801/(2.2 * 6 * 4) = 85\%$

Time for blocking

```
Dimension       Time    Gflop/s       GB/s        Error
   64   2.798738   0.714665  11.434634 0.000000e+00
  128   3.079668   0.649642  10.394280 0.000000e+00
  192   2.949185   0.681585  10.905358 0.000000e+00
  256   2.973606   0.677045  10.832723 0.000000e+00
  320   2.950382   0.688594  11.017506 0.000000e+00
  384   3.166501   0.643749  10.299982 0.000000e+00
  448   3.071128   0.702664  11.242616 0.000000e+00
  512   3.168179   0.677829  10.845265 0.000000e+00
  576   3.264838   0.702404  11.238467 0.000000e+00
  640   3.256561   0.643977  10.303640 0.000000e+00
  704   2.966741   0.705651  11.290408 0.000000e+00
  768   4.082646   0.665722  10.651559 0.000000e+00
  832   3.571540   0.645022  10.320349 0.000000e+00
  896   4.580854   0.628113  10.049804 0.000000e+00
  960   5.058059   0.699664  11.194631 0.000000e+00
 1024   3.236734   0.663472  10.615557 0.000000e+00
 1088   3.696069   0.696910  11.150558 0.000000e+00
 1152   4.421231   0.691583  11.065326 0.000000e+00
 1216   5.223907   0.688391  11.014259 0.000000e+00
 1280   6.153863   0.681573  10.905161 0.000000e+00
 1344   7.446900   0.652007  10.432112 0.000000e+00
 1408   8.611343   0.648287  10.372586 0.000000e+00
 1472   9.252391   0.689445  11.031116 0.000000e+00
```

```
1536  10.814957   0.670161  10.722569 0.000000e+00
1600  12.027636   0.681098  10.897570 0.000000e+00
1664  13.908687   0.662527  10.600438 0.000000e+00
1728  15.072107   0.684679  10.954870 0.000000e+00
1792  17.318526   0.664558  10.632933 0.000000e+00
1856  18.964697   0.674245  10.787926 0.000000e+00
1920  20.801535   0.680516  10.888255 0.000000e+00
```

Time for OMP

```
 Dimension       Time    Gflop/s       GB/s        Error
  64   4.705494   0.425069   6.801101 0.000000e+00
 128   2.725564   0.734044  11.744699 0.000000e+00
 192   1.913754   1.050355  16.805676 0.000000e+00
 256   1.404700   1.433235  22.931761 0.000000e+00
 320   1.132417   1.794053  28.704847 0.000000e+00
 384   0.999306   2.039848  32.637565 0.000000e+00
 448   0.924568   2.334031  37.344498 0.000000e+00
 512   0.783661   2.740323  43.845162 0.000000e+00
 576   1.284188   1.785747  28.571955 0.000000e+00
 640   1.191514   1.760073  28.161168 0.000000e+00
 704   1.010659   2.071402  33.142438 0.000000e+00
 768   1.224232   2.220092  35.521475 0.000000e+00
 832   1.020047   2.258447  36.135150 0.000000e+00
 896   1.098557   2.619156  41.906489 0.000000e+00
 960   1.398154   2.531155  40.498478 0.000000e+00
1024   0.842990   2.547461  40.759376 0.000000e+00
1088   1.155860   2.228494  35.655902 0.000000e+00
1152   1.310346   2.333466  37.335459 0.000000e+00
1216   1.423165   2.526827  40.429229 0.000000e+00
1280   1.709503   2.453522  39.256358 0.000000e+00
1344   1.786536   2.717791  43.484655 0.000000e+00
1408   2.111969   2.643324  42.293188 0.000000e+00
1472   2.274150   2.805009  44.880141 0.000000e+00
1536   2.978150   2.433644  38.938307 0.000000e+00
1600   3.611971   2.268014  36.288219 0.000000e+00
1664   3.563329   2.586033  41.376530 0.000000e+00
1728   3.969106   2.599971  41.599533 0.000000e+00
1792   4.370709   2.633250  42.131999 0.000000e+00
1856   4.683591   2.730140  43.682239 0.000000e+00
1920   5.275719   2.683194  42.931099 0.000000e+00
```

3 OMP bug

2. add critical region and to prevent race condition

3. limit the thread to 2

4. use 'ulimit -s unlimited' before run, or else stac overflow

5. dead lock, adjust the lock&unlock statement

6. the variable sum in the function overwrites the shared variable sum. making sum global variable and deleting the local sum solves this.

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

```
thread= 8
Jacobi:
N   timing
```

```
10 0.001525128 s
50 0.425376837 s
100 6.301234761 s
200 N/A

Gauss:
N   timing
10  0.001785497 s
50  0.200445369 s
100 3.216943429 s
200 N/A
```