hpc_hw2

March 24, 2017

1 Christopher Prince [cmp670@nyu.edu]

1.1 MATH.2012 HW 2

1.1.1 *Preliminaries*

```
In [1]: import pandas as pd
        import numpy as np
        import os

In [2]: import pylab as pl
        %matplotlib inline
```

1.2 Source files and reproducibility

The program has been packaged with a make file and a bash script to run the experiment on a given machine. The process is:

- Clone the github repo cmprince/hpc17 (files are in the hw2 directory).
- Run `make`.
- Run `runlaplace.sh` (if necessary change file permissions). This runs both Gauss-Seidel and Jacobi iterations for $N \in \{50, 100, 1000\}$, repeating 5 times, and writes the results of each run to a file.

1.3 Architectures

The tests were run on 4 different machines: my personal laptop, a server at NYU-CUSP, and two servers at CIMS. The specifications for the CUSP server was obtained from datahub.cusp.nyu.edu, and for the CIMS server from cims.nyu.edu. Additionally, on brawler, versions of the codes without paralellizing the residual calculations were run seperately to see the effect on performance.

Machine	Owner	Processor	No. Processors	No. Cores/Proc.	Freq. (GHz)	Memory (GB)	O
brawler	CIMS	AMD Opteron	2	2	2.6	8	R
compute	CUSP	Xeon E5-4640	4	8	2.40	1024	O
crunchy4	CIMS	AMD Opteron 6136	4	8	2.4	128	R
laptop	Me	Xeon E3-1505M v5	1	4	2.80	64	U

1.4 Results and observations

1.4.1 Timing results

The results files are space-delimited with the following columns:

```
In [3]: names=['N','run','usec']
```

Note timings are reported in μsec. Uncomment the cell below to retrieve the sample data used in this notebook:

```
In [4]: #!wget -r -l1 -np -nd -R index.html?*,html -e robots=off https://www.cims.nyu.edu/~cmp67
```

Read the files in as dataframes and merge the timing columns, starting with Jacobi:

```
In [5]: gb = pd.read_csv("jacobi2D-omp_laptop", header=None, delimiter=" ", names=names).drop("u
```

```
In [6]: for f in sorted(os.listdir('.')):
            if f[:13]=="jacobi2D-omp_":
                g = pd.read_csv(f, header=None, delimiter=" ",names=names[:-1] + [f])
                gb = gb.merge(g, on=names[:-1])
```

```
In [7]: gb
```

Out[7]:

	N	run	jacobi2D-omp_brawler	jacobi2D-omp_brawler_no_l2norm_parallel	\
0	50	1	64098	122950	
1	50	2	62954	324677	
2	50	3	62743	122595	
3	50	4	64429	122490	
4	50	5	62655	122655	
5	100	1	225179	492691	
6	100	2	224954	685548	
7	100	3	230868	1062484	
8	100	4	230655	596687	
9	100	5	224027	579145	
10	1000	1	21845399	47086142	
11	1000	2	22503545	47557294	
12	1000	3	22449011	47027564	
13	1000	4	21808371	46989342	
14	1000	5	21793035	47565443	

	jacobi2D-omp_crunchy4	jacobi2D-omp_laptop	jacobi2D-omp_mauler
0	3869467	79251	64626
1	3903125	25442	62799
2	3926278	24723	64779
3	3864763	21033	64820
4	3871138	20687	63231
5	3942070	73839	225253
6	3914154	72827	225845
7	405486	76498	225949

```
8                3895513                72453                230387
9                3924761                78413                225295
10               6944077              7032157              22376295
11               6916533              6291400              23219363
12               7054551              6769999              22188391
13               5993197              6500057              21984691
14               7056586              6462039              21998788
```

There are a few runs where the timing is much greater than the others; instead of taking a mean value over the runs I calculate the median run time here.

```
In [8]: g = gb.groupby(names[0])
        g.agg(np.median).drop('run', axis=1)
```

```
Out[8]:       jacobi2D-omp_brawler  jacobi2D-omp_brawler_no_l2norm_parallel  \
        N
        50                   62954                                  122655
        100                 225179                                  596687
        1000              21845399                                47086142

              jacobi2D-omp_crunchy4  jacobi2D-omp_laptop  jacobi2D-omp_mauler
        N
        50                  3871138                24723                64626
        100                 3914154                73839               225845
        1000                6944077              6500057             22188391
```

Repeating for Gauss-Seidel:

```
In [9]: gb = pd.read_csv("gs2D-omp_laptop", header=None, delimiter=" ", names=names).drop("usec"
```

```
In [10]: for f in sorted(os.listdir('.')):
             if f[:9]=="gs2D-omp_":
                 g = pd.read_csv(f, header=None, delimiter=" ", names=names[:-1] + [f])
                 gb = gb.merge(g, on=names[:-1])
```

```
In [11]: gb
```

```
Out[11]:         N  run  gs2D-omp_brawler  gs2D-omp_crunchy4  gs2D-omp_laptop  \
          0     50    1             124188            3893753            85588
          1     50    2             125482            3838401            85986
          2     50    3             124588            3890386            80742
          3     50    4             124582            3876093            84353
          4     50    5             125483            3887672            83798
          5    100    1             483850            3859820           309043
          6    100    2             841047            3898532           310122
          7    100    3             483088            2820291           311661
          8    100    4             483126            3886792           314079
          9    100    5             489542            3962736           313914
          10  1000    1           46242520           32425595         25087905
```

```
11  1000    2          47183560           32286412         25479990
12  1000    3          46306250           32326000         25177850
13  1000    4          46119539           32287583         23282460
14  1000    5          46696857           32292971         23516537

    gs2D-omp_mauler  gs2D-omp_mauler_no_l2norm_parallel
0            126195                                134271
1            211472                                126126
2            124792                                179788
3            124500                                125224
4            125350                                124929
5            484326                                487323
6           1353349                                490656
7            497028                                497853
8            483626                                484107
9            580930                                486020
10         46590516                              47566617
11         46820416                              47573191
12         46644781                              46385122
13         46665327                              46865096
14         47380288                              46662648
```

In [12]:
```
g = gb.groupby(names[0])
g.agg(np.median).drop('run', axis=1)
```

Out[12]:
```
      gs2D-omp_brawler  gs2D-omp_crunchy4  gs2D-omp_laptop  gs2D-omp_mauler  \
N
50              124588            3887672            84353           125350
100             483850            3886792           311661           497028
1000          46306250           32292971         25087905         46665327

      gs2D-omp_mauler_no_l2norm_parallel
N
50                                126126
100                               487323
1000                             46865096
```