- CSE 5441 Lab 1 Report -

Rajarshi Biswas
biswas.91@osu.edu

Table of Contents

Test cases

All the test cases were run on stdlinux.

Test Case # 1

Running the test case testgrid_1 with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_1

********************************************************************

Dissipation converged in 52 iterations.
With max DSV = 118.918 and min DSV = 107.279.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 0
Elapsed covergence loop time (time)   : 0.
Elapsed covergence loop time (chrono) : 0.009.

********************************************************************

real	0m0.004s
user	0m0.000s
sys	0m0.001s
```

Test Case # 2

Running the test case testgrid_2 with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_2

********************************************************************

Dissipation converged in 245 iterations.
With max DSV = 55.8359 and min DSV = 50.2669.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 0
Elapsed covergence loop time (time)   : 0.
Elapsed covergence loop time (chrono) : 0.247.

********************************************************************

real	0m0.004s
user	0m0.001s
sys	0m0.001s
```

Test Case # 3

Running the test case testgrid_50_78 with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_50_78

*******************************************************************

Dissipation converged in 1508 iterations.
With max DSV = 23.3695 and min DSV = 21.0358.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 0
Elapsed covergence loop time (time)   : 0.
Elapsed covergence loop time (chrono) : 2.582.

*******************************************************************

real	0m0.023s
user	0m0.004s
sys	0m0.000s
```

Test Case # 4

Running the test case testgrid_50_201 with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_50_201

*******************************************************************

Dissipation converged in 2286 iterations.
With max DSV = 4.78875 and min DSV = 4.30989.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 10000
Elapsed covergence loop time (time)   : 0.
Elapsed covergence loop time (chrono) : 10.533.

*******************************************************************

real	0m0.025s
user	0m0.012s
sys	0m0.001s
```

Test Case # 5

Running the test case `testgrid_200_1166` with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_200_1166

*******************************************************************

Dissipation converged in 14461 iterations.
With max DSV = 0.812727 and min DSV = 0.731455.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 420000
Elapsed covergence loop time (time)   : 1.
Elapsed covergence loop time (chrono) : 859.088.

*******************************************************************

real	0m0.930s
user	0m0.428s
sys	0m0.000s
```

Test Case # 6

Running the test case `testgrid_400_1636` with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_400_1636

*******************************************************************

Dissipation converged in 22283 iterations.
With max DSV = 1.18174 and min DSV = 1.06357.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 950000
Elapsed covergence loop time (time)   : 1.
Elapsed covergence loop time (chrono) : 1274.66.

*******************************************************************

real	0m1.323s
user	0m0.954s
sys	0m0.007s
```

Test Case # 7

Running the test case `testgrid_400_12206` with affect rate 0.1 and epsilon 0.1.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .1 .1  < testcase/testgrid_400_12206

*******************************************************************

Dissipation converged in 75269 iterations.
With max DSV = 0.0866714 and min DSV = 0.0780043.
Affect rate = 0.1;   Epsilon: 0.1.
Elapsed covergence loop time (clock)  : 29690000
Elapsed covergence loop time (time)   : 34.
Elapsed covergence loop time (chrono) : 33832.3.

*******************************************************************

real	0m33.896s
user	0m29.680s
sys	0m0.076s
```

Run test case longer

Running the test case # 7 with Affect rate = 0.03 and Epsilon = 0.03. The dissipation converges in more number of iterations, and the time takes a little over 3 minutes.

```
[biswas.91@sl4 5441]$ time ./amr_csr_serial .03 .03  <
testcase/testgrid_400_12206

*******************************************************************

Dissipation converged in 460838 iterations.
With max DSV = 0.0850069 and min DSV = 0.0824567.
Affect rate = 0.03;   Epsilon: 0.03.
Elapsed covergence loop time (clock)  : 182560000
Elapsed covergence loop time (time)   : 212.
Elapsed covergence loop time (chrono) : 211993.

*******************************************************************

real	3m32.059s
user	3m2.037s
sys	0m0.588s
```

Observation on using different time methods for serial programs:

- **Clock** ticks returns the number of clock ticks elapsed since the program was launched.
    - The clock timing method might be useful when we just need to measure the processor time consumed by a program.
    - Clock gives output analogous to the ‘user’ time output of ‘time’ UNIX utility.
    - Not a very useful measure in the case of profiling programs that may incur latencies due to the bus or network communication, that is not it is an ideal way to measure the real time spent by the program.

- **Time** returns the number of seconds since 00:00 hours, Jan 1, 1970 UTC.
    - The output of only seconds does not provide enough granularity to profile fast programs.
    - Outputs similar to the ‘real’ time output of ‘time’ UNIX utility, only with less accuracy.

- **Chrono** namespace has the element ‘clocks’ and ‘duration’. In the program I used ‘system_clock’ to get the time stamp, and ‘duration’ to find the duration of two consecutive time stamps.
    - The output is analogous to the ‘real’ time output of ‘time’ UNIX utility.
    - The ‘system_clock’ and ‘duration’ gives sufficient accuracy of elapsed time.

- **UNIX ‘time’** utility reports how long it took to execute a program in terms of user CPU time, system CPU time, and real time.
    - Provide good profiling in terms of real time spent and granularity.

Based on the above observation, in my opinion, using the elements in Chrono or using UNIX ‘time’ utility give the best result for serial programs. In one hand these give the real time consumed by a program, on the other hand these provide enough accuracy to compare program run times.

Hypothesis on best suited timing methods for parallel programs:

In my opinion, for parallel programming Chrono and UNIX time will work best as these provide finer level of granularity and provide way to measure the real time spent in the computation including for example - any communication delay incurred during multi-processor communication.