PIYALI DAS

5441 Lab1 Project Report

INTRODUCTION

I have tried using the dissipation model suggested in the project document. This program successfully takes command line arguments of Affect_Rate and Epsilon and uses the input from stdin to iterate over the data and converge in a finite time in a sequential fashion. By tweaking the affect rate and the epsilon number of iterations were modified and the program could be made to converge early. At every iteration the DSV value are computed and stored locally and updated later. Some points worth mentioning:

- I have provision of printing the individual box specifications and parameters. So if the user wants to see detailed description of individual DSV they can activate the print section.
- I have used the clock to print the time from the start to end of the program including data handling and processing. Also I have noted standalone time for the main iterating loop. It helped me to understand exactly what amount of time was utilized for data processing.

TIMING METHODS

I used 3 different ways to time the program that were also suggested in the project description.

- time from library <*ctime*>
  This time is reported in the seconds format. The program converged quickly for some set of inputs and for that the time was always zero. So it was difficult to make sense from this timing method. But this method made more sense where the program iterated for sufficiently long duration.

- Clock from library <*ctime*>
  This clock function returns the processor time consumed by the program. The value returned is expressed in clock ticks, which are units of time of a constant but system-specific length (with a relation of CLOCKS_PER_SEC clock ticks per second). It is in a milliseconds resolution. According to me this is the best time measurement for convergence loop among the methods chosen for timing the serial execution.

- Chrono
  This clock time is provided by the library <*chrono*>. It represents the system-wide real time wall clock. It can be easily used for a millisecond resolution. It represents the duration of time since the epoch, with the clock's resolution.

For making this execution parallel we can use the time provided bu the unix system as it can calculate the wall clock based running time of a parallel program. We can also use libraries designed specially to handle parallel programs like in OpenMP, there is a method called omp_get_wtime which can be used to measure the time taken by the parallel section of the code.

PARALLELISM

The following are the segments of the current code that can easily be made parallel to improve the performance:
1. Each of the box DSV can be calculated in parallel which can significantly boost the performance of the program.
2. The Maximum and Minimum DSV can also be calculated in parallel while updating the box DSV.

PERFORMANCE DATA

Run with the following specifications:

```
% time ./amr_csr_serial 0.1 0.1 < testgrid_1
************************************
dissipation converged in 39 iterations,
with max DSV = 177.643  and  min DSV = 161.22
        affect rate = 0.1;              epsilon = 0.1
elapsed convergence loop time (clock): 0
elapsed convergence loop time (time): 0.003
elapsed convergence loop time (chrono): 0.257

real 0m0.003s
user 0m0.003s
sys 0m0.01s
```

```
% time ./amr_csr_serial 0.1 0.1 < testgrid_2
*************************************
dissipation converged in 204 iterations,
with max DSV = 54.6506  and  min DSV = 49.2002
        affect rate = 0.1;              epsilon = 0.1
elapsed convergence loop time (clock): 0.01
elapsed convergence loop time (time): 0.14
elapsed convergence loop time (chrono): 9.047

real 0m0.14s
user 0m0.003s
sys 0m0.02s
```

```
% time ./amr_csr_serial 0.1 0.1 < testgrid_50_78
***************************************
dissipation converged in 2454 iterations,
with max DSV =  45.2878  and  min DSV = 40.7623
```

affect rate = 0.1; epsilon = 0.1
elapsed convergence loop time (clock): 0.16
elapsed convergence loop time (time): 0.129
elapsed convergence loop time (chrono): 127.047

real 0m0.129s
user 0m.003s
sys 0m0.13s

% time ./amr_csr_serial 0.1 0.1 < testgrid_50_201
****
dissipation converged in 4029 iterations,
with max DSV = 3.45479 and min DSV = 3.10944
affect rate = 0.1; epsilon = 0.1
elapsed convergence loop time (clock): 0.81
elapsed convergence loop time (time): 0.825
elapsed convergence loop time (chrono): 837.254

real 0m0.825s
user 0m0.010s
sys 0m0.85s

% time ./amr_csr_serial 0.1 0.1 < testgrid_200_1166
****
dissipation converged in 19429 iterations,
with max DSV = 1.00068 and min DSV = 0.900614
affect rate = 0.1; epsilon = 0.1
elapsed convergence loop time (clock): 20.77
elapsed convergence loop time (time): 23.772
elapsed convergence loop time (chrono): 24.283

real 0m23.772s
user 0m0.121s
sys 0m24.31s

% time ./amr_csr_serial 0.1 0.1 < testgrid_400_1636
****
dissipation converged in 32894 iterations,
with max DSV = 0.676886 and min DSV = 0.609199
affect rate = 0.1; epsilon = 0.1
elapsed convergence loop time (clock): 46.07

elapsed convergence loop time (time): 58.465
elapsed convergence loop time (chrono): 59956.8

real 0m58.465s
user 0m0.123s
sys 0m1.01s

% time ./amr_csr_serial 0.1 0.1 < testgrid_400_12206
****
dissipation converged in 105840 iterations,
with max DSV = 0.114532 and min DSV = 0.103079
affect rate = 0.1; epsilon = 0.1
elapsed convergence loop time (clock): 1940.49
elapsed convergence loop time (time): 1950.692
elapsed convergence loop time (chrono): 2185170

real 0m1950.692s
user 0m6.485s
sys 36m25s

% time ./amr_csr_serial 0.5 0.5 < testgrid_400_12206
****
dissipation converged in 10425 iterations,
with max DSV = 0.143344 and min DSV = 0.071677
affect rate = 0.5; epsilon = 0.5
elapsed convergence loop time (clock): 198.45
elapsed convergence loop time (time): 198.228
elapsed convergence loop time (chrono): 205873

real 0m198.228s
user 0m0.422s
sys 0m3.26s

PERFORMANCE ANALYSIS

It is seen that the number of iterations and the time to converge greatly depends on the value of Affect rate and Epsilon. For the testgrid_400_12206 it is seen that with Affect rate of 0.1 and Epsilon as 0.1 the dissipation converges after 105840 iterations. But when the Affect rate and Epsilon are both chosen as 0.5 the dissipation converges after 10425 iterations. Hence it can be seen that even for a sequential execution, there are lots of nice techniques which can be employed to improve the performance of the program.