# Project I

Neil P. Parmar
Dept. of Electrical Engineering and Computer Science
Oregon State University, Corvallis, USA
parmarn@oregonstate.edu

## I. Introduction

This project uses Bézier surface in order to measure the performance of the parallel system for computing its volume. This particular report has been categorized into various sections which analyses the system configurations & load, implementation specifications, and then primarily focuses upon the various performance evaluation & analysis.

## II. System Configurations & Load

This project was implemented using the CentOS operating System Release 6.7, at OSU Computer labs. This system has 8 CPU cores, and has an available space of 1.4G. The particular system configuration details have been shown in fig 1 and 2.

As according to the Fig 2, the CPU load average for the last 1, 5 and 15 minutes was 0.50, 0.46 and 0.31 respectively under 8 CPU cores (from Fig 1).

```
dear119-16 ~/workspace 112% grep 'processor. *:' /proc/cpuinfo | wc -l
8
dear119-16 ~/workspace 113% df -h 'pwd'
df: `pwd': No such file or directory
df: no file systems processed
dear119-16 ~/workspace 114% df -h `pwd`
Filesystem            Size  Used Avail Use% Mounted on
stak.engr.oregonstate.edu:/stu
                      3.4G  2.0G  1.4G  59% /nfs/stak/students
dear119-16 ~/workspace 115%
d
```

Fig 1. System configurations

```
dear119-16 ~ 42% uptime
 20:45:44 up 4 days,  1:40,  3 users,  load average: 0.50, 0.46, 0.31
dear119-16 ~ 43%
```

Fig 2. System configurations – System Load

## III. Implementation Specifications

The main code for the particular project is contained in "Project_main.cpp". The particular file computes the volume, and measures the performance, time in microseconds and seconds. The certain snapshot of the code output has been stored in the files Computation1.txt and Computation2.txt.

## IV. Performance Evaluation & Analysis

TABLE I. VOLUME

| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Volume | 4 | 13.83 | 13.83 | 13.83 | 13.83 | 13.83 | 13.83 | 13.83 |
| | 200 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 |
| | 500 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 |
| | 700 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 |
| | 1000 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 |
| | 1500 | 14.07 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 | 14.06 |

Table I represents the volume computed in relation with the number of threads and its respective Number of Subsections (NUMS). As shown in the Table I, the number of threads range from 1 to 7 and NUMS of 4, 200, 500, 700, 1000, 1500 are specifically used, this consistency has been maintained throughout the project and report.

Also the measured volume, as can be observed from Table I and Fig 3, is almost similar to the ranging number of threads to NUMS. Thus the number of threads does not affect the computation but the NUMS (Number of subsections) curves the volume. That is, a lower NUMS would provide a different output as the area for computing the volume is considerably reduced and therefore it does not cover the complete range. Though, as the NUMS increases, the volume curves to be more constant and accurate.

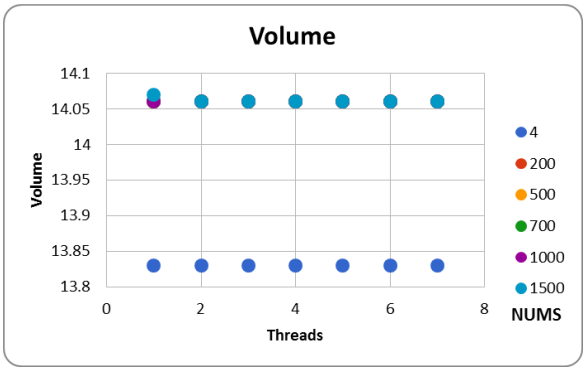


Fig 3. Volume Computation on a graph

TABLE II. TIME (SEC)

| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Time (sec) | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 500 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 | 0 | 0 |
| | 700 | 0.02 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| | 1000 | 0.04 | 0.03 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| | 1500 | 0.1 | 0.06 | 0.05 | 0.05 | 0.05 | 0.04 | 0.03 |

Table II, represents the time evaluated in seconds. As can be seen from the particular, the lower values of NUMS, the output is considerably closer to zero. This is true for all threads, as the computation time for a smaller number of subsections can be considerably low. However, upon increasing the number of subsections, the output ranges closer but away from zero seconds.

Moreover, for the purpose of visual representation, Table III measures the similar time in microseconds. This has been computed for the purpose of analysis only.

TABLE III. TIME (MICROSECONDS)

| | Threads | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Time (microsec) | 4 | 76.88 | 70.85 | 146.24 | 173.35 | 178.6 | 193.35 | 214.75 |
| | 200 | 4141.3 | 2241.52 | 1836.59 | 1902.62 | 1821.52 | 1649.97 | 1470.47 |
| | 500 | 14772.91 | 13275.2 | 7608.18 | 7013.91 | 7655.35 | 5006.22 | 5993.25 |
| | 700 | 25060.98 | 20246.62 | 11665 | 10612.59 | 12185.42 | 9577.24 | 9459.68 |
| | 1000 | 44629.02 | 26034.52 | 20666.85 | 20460.52 | 20280.65 | 17724.04 | 16265.43 |
| | 1500 | 97979.9 | 56889.24 | 47096.16 | 43821.31 | 43434.66 | 38150.22 | 32013.87 |

TABLE IV. SPEEDUP

| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Speedup | 4 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| | 200 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| | 500 | | 2 | 2 | 2 | 2 | #DIV/0! | #DIV/0! |
| | 700 | | 1 | 2 | 2 | 2 | 2 | 2 |
| | 1000 | | 1.333333333 | 2 | 2 | 2 | 2 | 2 |
| | 1500 | | 1.666666667 | 2 | 2 | 2 | 2.5 | 3.333333333 |

TABLE V. PERFORMANCE (OUTPUT I)

| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Performance(MegaHeights per second) | 4 | 0.27 | 0.11 | 0.11 | 0.09 | 0.08 | 0.14 | 0.08 |
| | 200 | 21.34 | 17.58 | 20.33 | 21.32 | 20.6 | 25.86 | 27.53 |
| | 500 | 16.4 | 24.91 | 32.94 | 36.67 | 40.69 | 58.51 | 69.16 |
| | 700 | 23.23 | 31.22 | 46.51 | 41.46 | 39.34 | 40.87 | 57.41 |
| | 1000 | 23.04 | 35.79 | 45.82 | 45.62 | 54.24 | 53.01 | 68.73 |
| | 1500 | 22.76 | 36.16 | 49.72 | 49.42 | 47.27 | 60.24 | 73.11 |

Table II, III, IV, and V observe the time, speedup and performance respectively. Note, in Table IV, the "#DIV/0!" is an error generated by Excel, where the computed speedup is divided by zero. This glitch is observed for lower values of NUMS as the computational time is very small.

Fig 4.1 below shows the performance with respect to Table V. From Fig 4.1, the performance ranges to be considerably low for lower values of NUMS. Also, as can be observed from Table V and Fig 4.1, as the number of threads increases for a lower value of NUMS (e.g. NUMS = 4), the performance degrades with a very small margin. This is because the available thread resources are not completely used for parallel computing. However, the performance then considerably improves with an increased number of threads and NUMS.

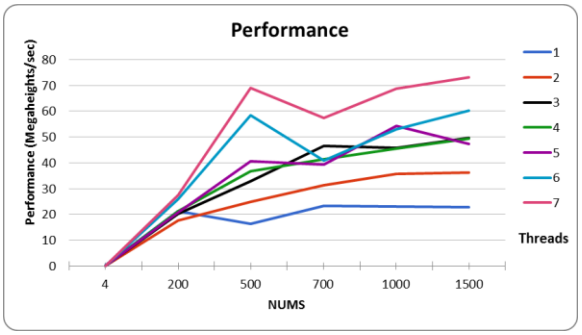


Fig 4.1. Performance Computation for various thread values (Output I)

The transpose graph of Fig 4.1 has been shown below. Here the performance is plotted with respect to the threads and NUMS. Where the performance drops sharply for initial number of threads, but then smooths out (almost equally) as the number of threads increase. However, as can be observed from Fig 4.2, the performance is almost constant for any number of threads where the number of subsections (NUMS) is 4.

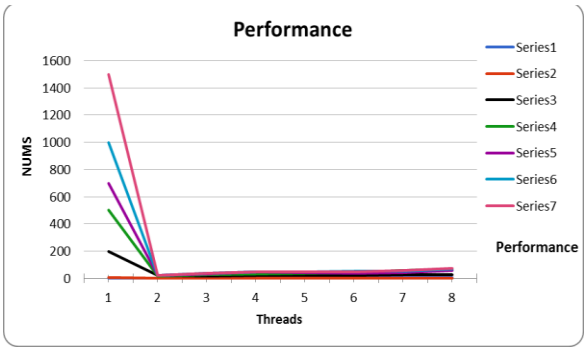


Fig 4.2. Performance Computation for various thread values- Transpose Graph (Output I)

TABLE VI. PERFORMANCE (OUTPUT II)

| | | Threads | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Performance(MegaHeights per second) | 4 | 0.21 | 0.23 | 0.11 | 0.09 | 0.09 | 0.08 | 0.07 |
| | 200 | 9.66 | 17.85 | 21.78 | 21.02 | 21.96 | 24.24 | 27.2 |
| | 500 | 16.92 | 18.83 | 32.86 | 35.64 | 32.66 | 49.94 | 41.71 |
| | 700 | 19.55 | 24.2 | 42.01 | 46.17 | 40.21 | 51.16 | 51.8 |
| | 1000 | 22.41 | 38.41 | 45.93 | 48.87 | 49.31 | 56.42 | 61.48 |
| | 1500 | 22.96 | 39.55 | 47.77 | 51.34 | 51.8 | 58.98 | 70.28 |

Table VI, represents a second set of performance. The timing for the particular hasn't been specifically noted, but is almost similar to that observed with the first test case. Here, as seen from the Table VI and Fig 4.3, the output is comparatively smoother to that observed in Fig 4.2. The system understands a minor glitch depending upon the system load and its performance. The Table VI and Fig 4.3 represent the best output observed by the system. Also, the transpose version of Fig 4.3 graph is shown below in Fig 4.4 where the performance is plotted with respect to the number of threads and NUMS.

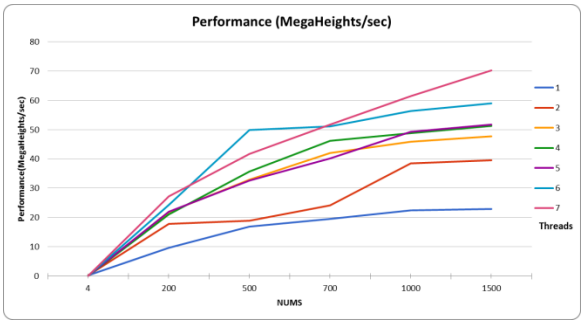


Fig 4.3. Performance Computation for various thread values (Output II)

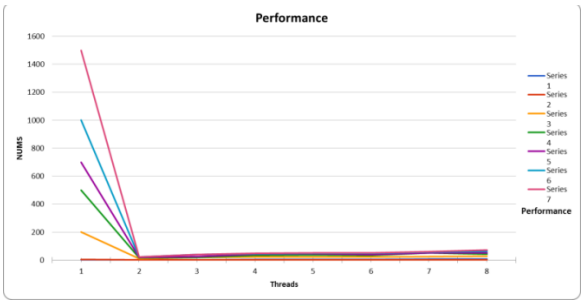


Fig 4.3. Performance Computation for various thread values- Transpose Graph (Output II)

TABLE VII. FPARALLEL

| NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 4 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| 200 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| 500 | | 1 | 0.75 | 0.666666667 | 0.625 | #DIV/0! | #DIV/0! |
| 700 | | 0 | 0.75 | 0.666666667 | 0.625 | 0.6 | 0.583333333 |
| 1000 | | 0.5 | 0.75 | 0.666666667 | 0.625 | 0.6 | 0.583333333 |
| 1500 | | 0.8 | 0.75 | 0.666666667 | 0.625 | 0.72 | 0.816666667 |

Table VII, represents the computation of $F_{parallel}$. The computation for the particular is based upon the values of Speedup observed in Table IV. Here the maximum value of $F_{parallel}$ observed is for seven threads with the number of subsections being 1500. Upon converting the value into percentage, $F_{parallel}$ is about *81.67%*.

TABLE VIII. MAX SPEEDUP

| | | Threads | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | NUMS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Max Speedup | 4 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| | 200 | | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! | #DIV/0! |
| | 500 | | #DIV/0! | 4 | 3 | 2.666666667 | #DIV/0! | #DIV/0! |
| | 700 | | 1 | 4 | 3 | 2.666666667 | 2.5 | 2.4 |
| | 1000 | | 2 | 4 | 3 | 2.666666667 | 2.5 | 2.4 |
| | 1500 | | 5 | 4 | 3 | 2.666666667 | 3.571428571 | 5.454545455 |

Also, the maximum value of the speedup observed is about *5.45* units, as shown in Table VIII, where the number of threads is seven to 1500 NUMS.

## V. Conclusion

Thus, from the particular project, it can be analyzed that as the number of threads considerably increase, an increase in performance is observed with a higher number of subsections. However, if the number of subsections is smaller to the higher thread values, the performance degrades with a very minor margin.

Also, an improved performance is best observed for an increased number of subsections to the highest number of threads, based upon ones system configurations.