Project 0

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

I. Introduction

This project discusses about the implementation details for understanding the use of parallel programming with a variable array size and number of tries (NUMTRIES). This report primarily focuses upon the system configurations, the implementation, and performance evaluations & analysis.

II. System Configurations

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 and load have been shown in fig 1 and 2.

```
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 ~ 32% uptime
 13:58:30 up  3:42,  2 users,  load average: 0.45, 0.48, 0.45
dear119-16 ~ 33% gcc
gcc: no input files
dear119-16 ~ 34% g++ --version
g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-16)
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

dear119-16 ~ 35% uptime
 14:18:56 up  4:03,  2 users,  load average: 0.00, 0.00, 0.11
dear119-16 ~ 36%
```

Fig 2. System configurations – System Load

III. Implementation

In this project, "Project0_t1" evaluates the performance with the use of a single thread and "Project0_t4" evaluates the performance with the use of four threads. The details of the implementation have been saved in the files Imp1.txt, Imp2.txt, Imp3.txt, and Imp4.txt.

IV. Performance Evaluation and Analysis

In this project, the performance of the project is evaluated with respect to the Number of Tries (NUMTRIES) and the Array Size measuring Peak and Average Performance in Mega-Multiples per Second. This particular section explains the implementation and analysis of the various tested results.

A. Performance evaluation with varying Number of tries

TABLE I. PEAK PERFORMANCE MEASUREMENT FOR VARYING NUMTRIES

		Number of Tries			
		10	100	1000	10000
Thread Peak Performance	1	381.23	381.34	381.54	381.49
	4	424.14	1289.4	1291.04	1290.26

Considering a single thread with varying number of tries, the peak performance almost remained constant. However, using 4 threads, the peak performance was considerably low with initial NUMTRIES. However, the peak performance sharply increased with an increase in NUMTRIES, as represented in Fig 3.1. However, after reaching its peak, the performance then remained almost constant with an increase in NUMTRIES.

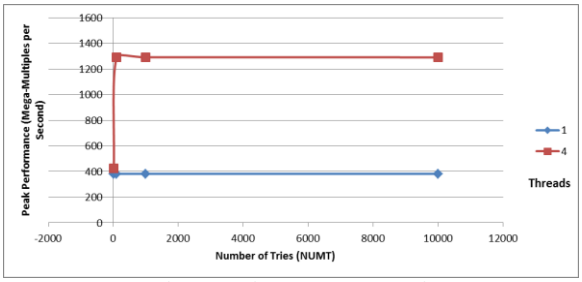

Fig 3.1. PEAK Performance for varying NUMT for threads 1 and 4

Based upon Table I and Fig 3.1, the lower NUMTRIES using a parallel 4 thread environment, would not be utilizing the resources to a considerable extent of its capacity. As the system peak performance remains almost constant for a single thread environment with varying NUMTRIES.

Fig 3.2 below represents the similar results where the number of threads (1 and 2) has been plotted on the horizontal axis with respect to the performance on the vertical axis. Fig 3.2 shows similar graphical representation where the lower NUMTRIES have poor performance for a thread size of 4, however the performance remains almost constant for a higher varying NUMTRIES having a thread size of 1 and 4.

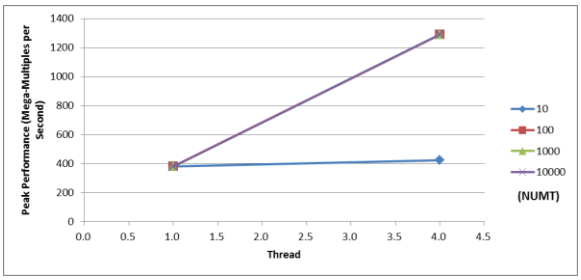

Fig 3.2. PEAK Performance for varying NUMT for threads 1 and 4

Secondly, the Table II represents the Average performance for the similar varying Number of Tries (NUMTRIES).

TABLE II. AVERAGE PERFORMANCE MEASUREMENT FOR VARYING NUMTRIES

		Number of Tries			
		10	100	1000	10000
Thread Average Performance	1	373.97	379.68	380.47	380.59
	4	412.25	1272.1	1282.19	1203.86

As being observed from the Table II, Fig 4.1, and Fig 4.2, the average performance shows a similar performance measurement for the varying NUMTRIES. Here, a lower average performance is observed for lower values of NUMTRIES using thread size of 4. However, with a further increase in the NUMTRIES, the average performance after reaching its peak gradually decreases. The particular behavior can be observed using Fig 4.1.

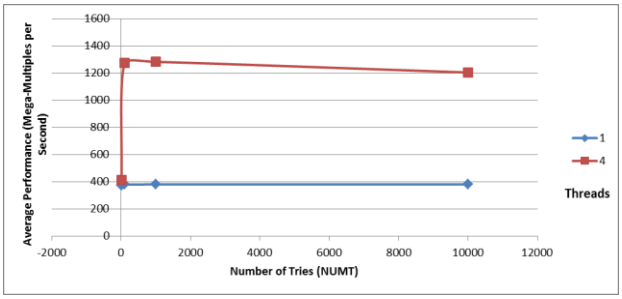

Fig 4.1. AVERAGE Performance for varying NUMT for threads 1 and 4

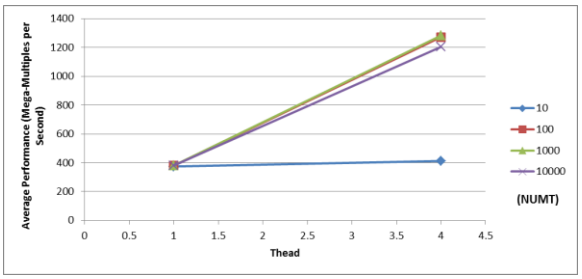

Fig 4.2. AVERAGE Performance for varying NUMT for threads 1 and 4

The gradual decrease in the average performance may not be considerably conclusive as there are certain factors which may affect performance, such as the system load at the moment while evaluating performance.

B. Performance evaluation with varying Array Size

TABLE III. PEAK PERFORMANCE MEASUREMENT FOR VARYING ARRAY SIZE

		Array Size			
		10000	100000	1000000	10000000
Thread Peak Performance	1	159.68	183.17	383.31	376.54
	4	586.1	799.93	1447.5	1260.52

Firstly, the highlighted columns in Table III represent the array size which observes a fluctuating performance. The particular behavior can be seen in the implementation file Imp4.txt.

While testing the peak performance, for a single thread a lower performance is observed for a lower array size. However, the performance remains almost constant for higher array size.

However, while using 4 threads, the performance gradually increases with the increase in the number of threads until it reaches a peak. After which, a steady

decrease in the performance is observed with an increase in the array size.

The particular result is shown in the fig 5.1 and fig 5.2.

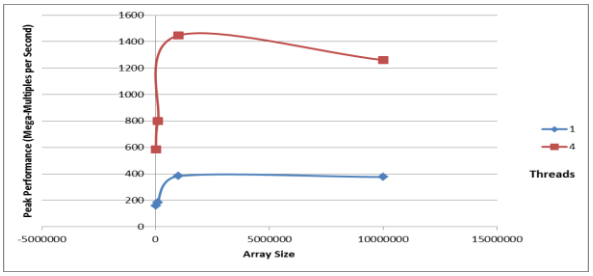

Fig 5.1. PEAK Performance for varying Array Size for threads 1 and 4

Fig 5.2 shows the similar behavior where the performance for thread 1 is almost gradually constant with increase in array size, however the performance degrades with the increase in the array size for 4 threads.

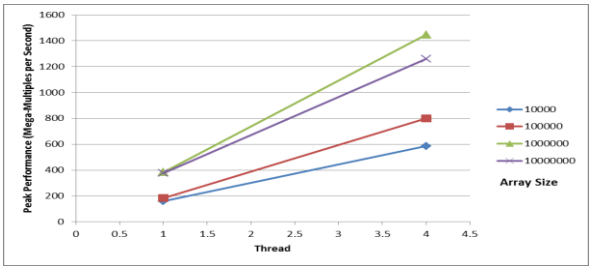

Fig 5.2. PEAK Performance for varying Array Size for threads 1 and 4

Secondly, the table IV below represents the average performance with the similar varying array size. Here, the initial two columns have fluctuating values due to the smaller array size.

TABLE IV. AVERAGE PERFORMANCE MEASUREMENT FOR VARYING ARRAY SIZE

		10000	100000	1000000	10000000
Thread Average Performance	1	102.28	155.77	307.52	356.18
	4	521.53	606.01	983.79	1118.45

The average performance, as shown in fig 6.1 and fig 6.2, gradually increases for threads 1 and 4 with the increase in the Array Size. For a single thread, observing fig 6.1, the performance gradually increases with the increase in the array size. However for 4 threads, the average performance gradually increases with the increase in the number of threads. This result is completely opposite to that observed for the Peak Performance in fig 5.1.

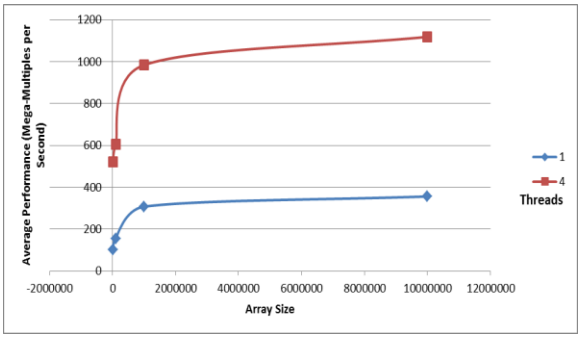

Fig 6.1. AVERAGE Performance for varying Array Size for threads 1 and 4

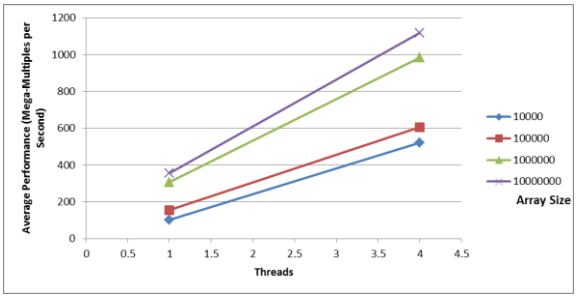

Fig 6.2. AVERAGE Performance for varying Array Size for threads 1 and 4

Fig 6.2 represents a similar increase in performance based upon the threads plotted with respect to the array size.

From the results analyzed using the peak and average performance using varying array size, it can be concluded that the array size should not be considered too large for measurement as the system might not allow much memory usage. Moreover, too lower a value would degrade the performance as the machine would not utilize the assigned capacity.

V. Conclusion

In this particular implementation, an increased system performance was observed with higher NUMTRIES and with standard array size. Moreover, a systems performance may be affected due to various factors such as machine load and its memory capacity.