Project II

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

I. Introduction

The intent of the particular project is to understand the functionality of coarse-grained and fine-grained parallelism while also applying the static and dynamic scheduling.

The problem for this project is focused towards solving the "N-Body Problem", where a group of planetary masses are swarming around while being attracted with each other.

The project has been divided into sections where the next section discusses about the System Configuration & Load, then any Implementation Specifications and then performance evaluation & analysis. The final section provides information about the performance evaluation using rabbit and Xeon Phi.

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. These system configuration 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 particular implementation code is known as the Project2_main.cpp, where the fine-grained and coarse-grained techniques have been implemented using different types of scheduling with the default chunksize as1.

Also, additional computations with rabbit and Xeon Phi has been performed with the project example. The configurations, computations and the output could be found in the files "rabbitconnectionsettings.txt" and "XeonPhiContentsOutputs.txt" respectively.

IV. Performance Evaluation & Analysis

TABLE I. PERFORMANCE

Performance (MegaBodies Compared Per Second)	Thread				
	1	2	3	4	5
coarse+static	30.85	51.56	76.64	78.76	86.03
coarse+dynamic	32.4	61.28	85.53	99.98	107.06
fine+static	29.44	31.88	36.06	44.5	45.69
fine+dynamic	18.94	21.49	24.81	25.66	25.66

Table I represents the performance evaluated with respect to the number of threads, which in this project the threads considered are 1,2,3,4 and 5, and the various parameters of coarse & fine-grained and static & dynamic techniques.

The graph in Fig 3 shows scheduling plotted with respect to the performance.

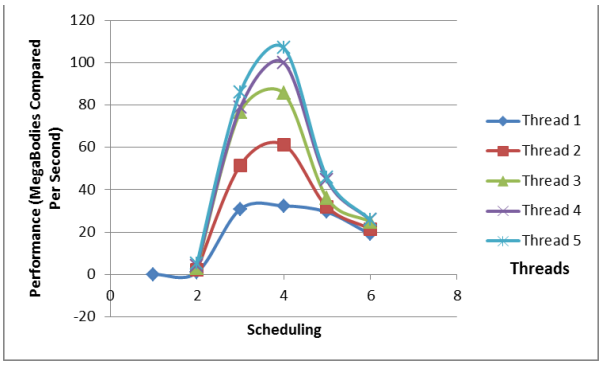

Fig 3. Performance

Here the performance is evaluated based upon the number of threads. As can be seen from Fig 3 and Table 1, the performance for all the coarse grained techniques is considerably increased with dynamic scheduling, however for fine-grained techniques observes a considerably lower performance for dynamic technique. This is because using the fine-grained techniques, the *loop* every time calls the "*#pragma omp parallel for*", which creates the assigned number of threads. Whereas, in coarse-grained technique, the "*#pragma omp parallel for*", is called only once and therefore reducing the overhead of creating more threads while looping.

Fig 4 below shows the evaluated performance with respect to the number of threads. Here, the lower numbers of threads show a lower performance, but as the number of threads increase, the performance almost levels out while using fine-grained & dynamic/static techniques, the performance of coarse & dynamic and coarse & static continues to considerably increase with the increase in the number of assigned threads.

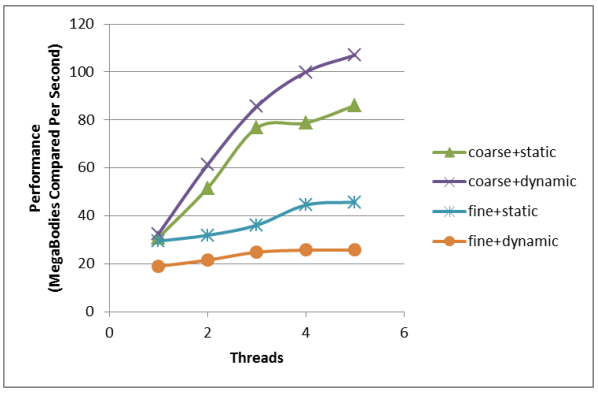

Fig 4. Transpose for Performance

V. Extra Performance evaluation

A. Performance using rabbit

The configuration details of the connection which has been established with the rabbit server can be found in the file rabbitconnectionsettings.txt, where all performance calculations can also be found.

TABLE II. PERFORMANCE WITH RABBIT

Performance (MegaBodies Compared Per Second)	Thread				
	1	2	3	4	5
coarse+static	28.28	35.57	43.91	49.56	60.43
coarse+dynamic	27.91	31.14	45.24	53.27	54.56
fine+static	23.59	16.3	13.51	10.36	10.04
fine+dynamic	9.1	3.95	3.86	3.07	2.87

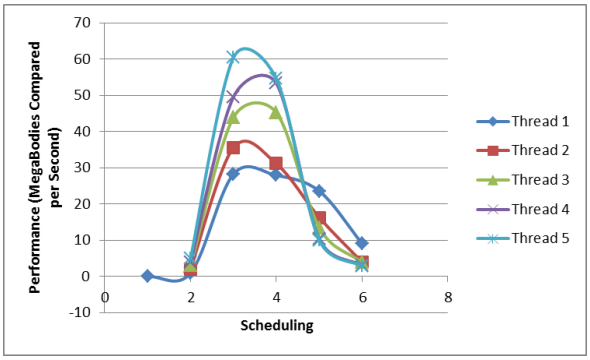

Fig 5. Performance evaluation using rabbit

As shown in the Table II and Fig 5, the performance shown for fine-grained has considerably decreases with the increase in the number of threads. However, for coarse-grained technique, the performance increases with respect to the number of threads. This is due to the overhead caused in the fine-grained technique with the increased number of threads created while looping.

This can also be seen from Fig 6, which is a graph transpose to Fig 5.

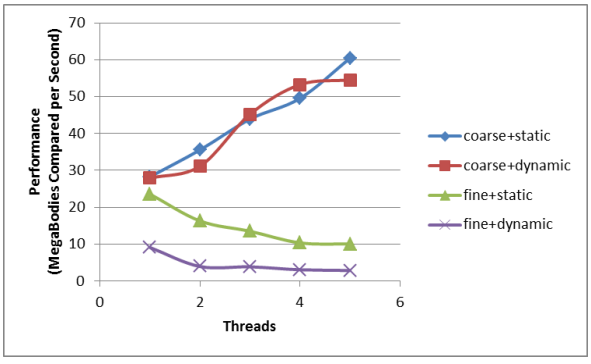

Fig 6. Performance evaluation using rabbit –Transpose

B. Performance with Xeon Phi

III. PERFORMANCE WITH XEON PHI

Performance (MegaBodies Compared Per Second)	Thread				
	1	2	3	4	5
coarse+static	3.63	6.13	7.08	13.61	16.48
coarse+dynamic	3.64	7.01	10.05	13.29	16.25
fine+static	3.24	4.28	4.96	5.25	5.34
fine+dynamic	3.21	2.55	3	2.3	2.13

Table III shows the performance evaluation using Xeon Phi processor. Here, the particular processor shows the similar curves as observed previously using CentOS and rabbit. As shown in Table III and Fig 7, the performance evaluated is particularly higher with the coarse grained scheduling. Fig 8 shows a transpose representation to the scheduling where the fine-grained techniques performance is considerably constant or slowly descending.

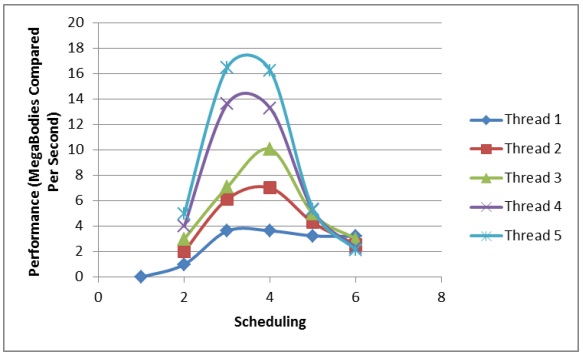

Fig 7. Performance evaluation using Xeon Phi

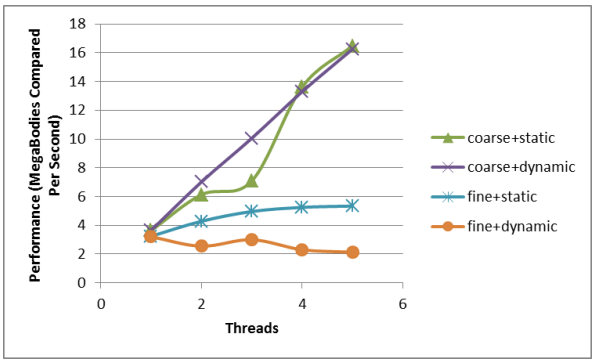

Fig 8. Performance evaluation using Xeon Phi –Transpose

VI. Conclusion

Thus from the particular project it can be concluded that the performance is observed highest with the coarse-grained technique while observing dynamic/static scheduling due to lesser overhead created with respect to the number of threads.