Aashutosh Y Taikar
933020005

Project #2 OpenMP: Static vs Dynamic and Small vs. Large Chunksize

1. Tell what machine you ran this on
   Ubuntu running on Intel Core i5 8GB RAM
   And
   Flip server OSU

2. Create a table with your results.

Ubuntu running on Intel Core i5 8GB RAM:

	static-1	static-4096	dynamic-1	dynamic-4096
1	242.54	242.21	237.54	238.31
2	464.18	422.6	463.66	414.83
4	755.62	609.56	808.4	624.55
6	714.84	622.58	807.38	719.66
8	770.95	726.75	801.2	647.74
10	766.75	656.45	799.53	664.58
12	761.18	618.74	788.32	652.25
14	767.87	711.84	791.38	726.8
16	758.46	685.43	789.66	670.19

Flip server:

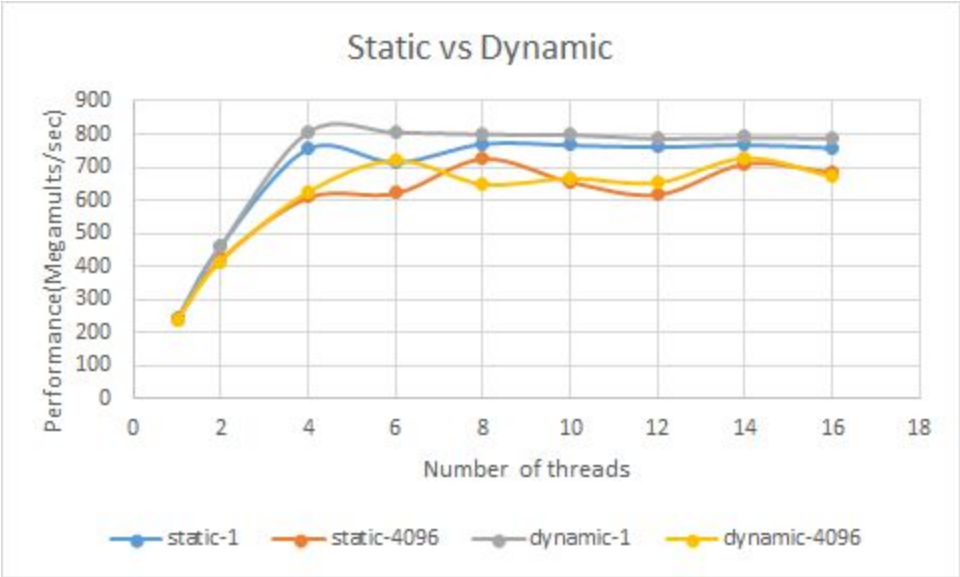
	static-1	static-4096	dynamic-1	dynamic-4096
1	270.66	270.01	270.52	270.1
2	541.26	482.11	539.37	471.4
4	1038.51	770.6	1036.23	765.04
6	1550.19	938.27	1554.68	933
8	2039.77	1130.58	2071.94	1127.33
10	2024.63	1129.46	2392	1120.3
12	2095.8	1120.43	2375.11	1121.25
14	2514.24	1130.61	2210.52	1115.87
16	2571.68	1130.45	3037.68	1032.27

flip1 ~ 178% uptime

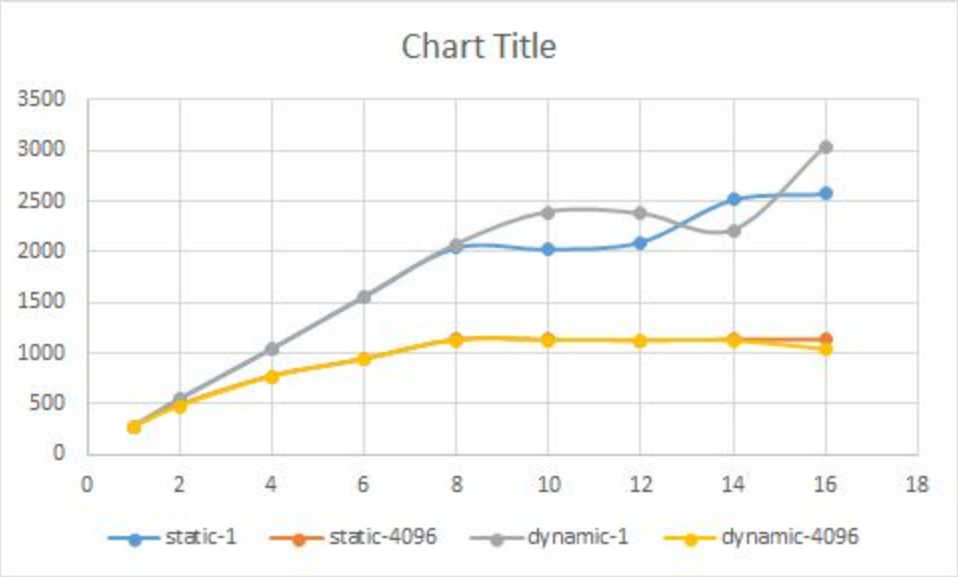
22:17:08 up 2 days, 3 min, 97 users, load average: 2.04, 3.01, 4.90

3. Draw a graph. The X axis will be the number of threads. The Y axis will be the performance in whatever units you sensibly choose. On the same graph, plot 4 curves:
   static,1
   static,4096
   dynamic,1
   Dynamic,4096

Observation from my laptop:



Observation from flip server



4. What patterns are you seeing in the speeds?

The static and dynamic with chunk size 1 has higher speeds as it gets executed faster than the static and dynamic having the chunksize 4096. This is because more computation is required when we have larger chunksize. In this case the dynamic could have been faster while using 4096 chunksize but because of the overhead it falls back as seen from the graphs.

5. Why does chunksize 1 vs. 4096 matter like this?

By using **static** scheduling:

OpenMp divides the loop iterations into chunks in a circular order and distributes them to each thread. If the chunksize is 1 the loop iterations are first divided by chunksize(1) and then divided by number of threads. For the chunksize(4096) the iterations will be divided into 4096. When we use chunksize 1 the performance is more as the compute:communicate ratio is better than that of using the chunksize 4096 where the compute:communicate ratio is too much.

By using **dynamic** scheduling:

In dynamic scheduling using chunksize 1 is better than 4096. By using chunksize 1 only 1 iteration is given to a thread at a time. In this case there are random number getting multiplied from the array. Thus the time required for every iteration can vary as they have different values to compute. By using the chunksize 4096 the (iterations/4096) will be dynamically distributed to each thread whenever the thread is available to compute. By doing this a thread spends more time to compute the iterations. As this scheduling is unpredictable and happens in the runtime it is difficult for the scheduler to get threads which are ready to take the load of computing the next

iterations. Due to this overhead the performance while using dynamic with the chunksize 4096 decreases.

6. Why does static vs. dynamic matter like this?

Static vs. Dynamic depends upon the computations required. If the computations are varying in each loop iteration then **dynamic** scheduling will be better than the **static**. But the chunksize of the dynamic needs to be set carefully. By default the chunksize 1 for dynamic is the best in the case when we do not know the overhead. The chunksize 1 for dynamic will allocate next available iteration to the next available thread. In this case the array has random numbers which are to be multiplied. Thus the data to be computed is random and hence by using dynamic scheduling with chunksize-1 more performance can be obtained.