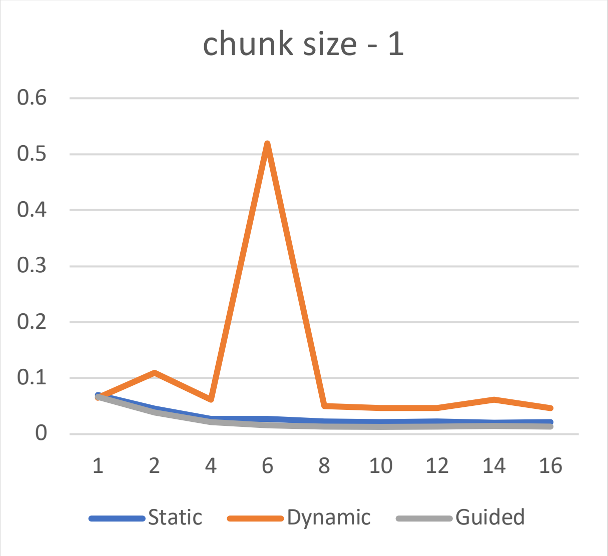
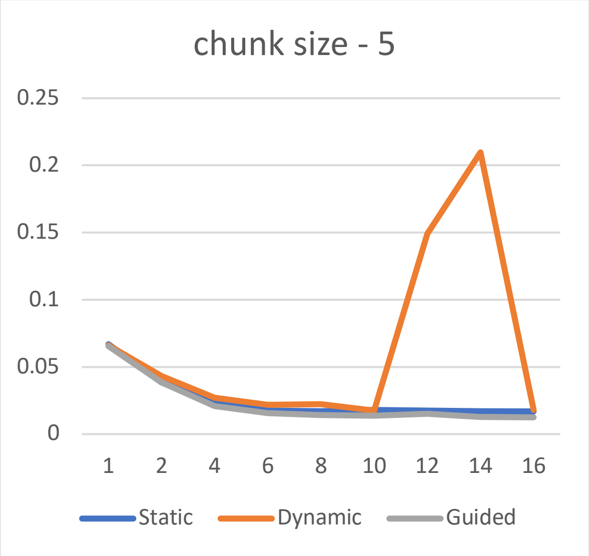
Multicore computing
Project #3 – prob2

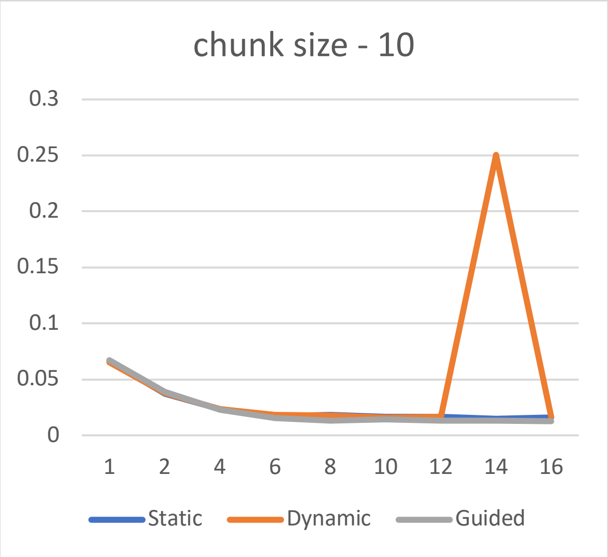
Department : Software
Name : Jeong Eui Chan
Student ID : 20195914

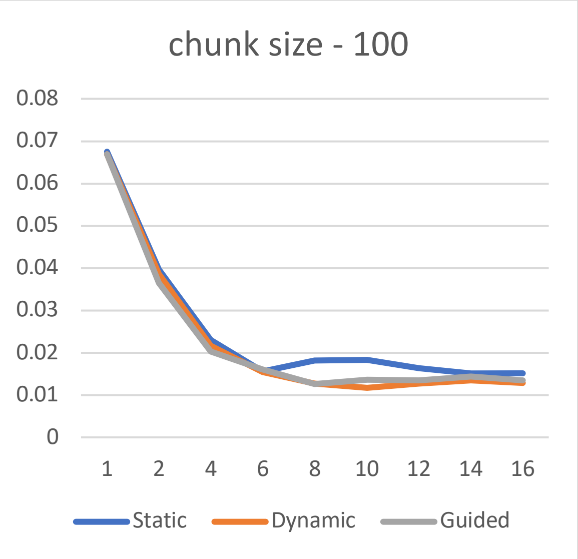
Exec time graph

Execution time	Chunk size	1	2	4	6	8	10	12	14	16
Static	1	0.0699	0.0460	0.0268	0.0277	0.0225	0.0218	0.0225	0.0204	0.0214
Dynamic		0.0652	0.1094	0.0613	0.5194	0.0497	0.0465	0.0468	0.0616	0.0464
Guided		0.0665	0.0383	0.0219	0.0152	0.0136	0.0130	0.0134	0.0147	0.0131
Static	5	0.0669	0.0384	0.0237	0.0173	0.0171	0.0182	0.0173	0.0170	0.0169
Dynamic		0.0662	0.0431	0.0270	0.0216	0.0224	0.0175	0.1490	0.2097	0.0182
Guided		0.0655	0.0386	0.0211	0.0157	0.0141	0.0138	0.0150	0.0129	0.0125
Static	10	0.0666	0.0373	0.0233	0.0173	0.0180	0.0167	0.0164	0.0148	0.0157
Dynamic		0.0654	0.0379	0.0232	0.0180	0.0179	0.0159	0.0166	0.2505	0.0163
Guided		0.0671	0.0386	0.0226	0.0152	0.0131	0.0140	0.0130	0.0129	0.0126
Static	100	0.0675	0.0397	0.0230	0.0156	0.0182	0.0183	0.0164	0.0151	0.0152
Dynamic		0.0669	0.0386	0.0217	0.0155	0.0127	0.0117	0.0127	0.0134	0.0128
Guided		0.0669	0.0365	0.0202	0.0161	0.0126	0.0136	0.0135	0.0144	0.0134



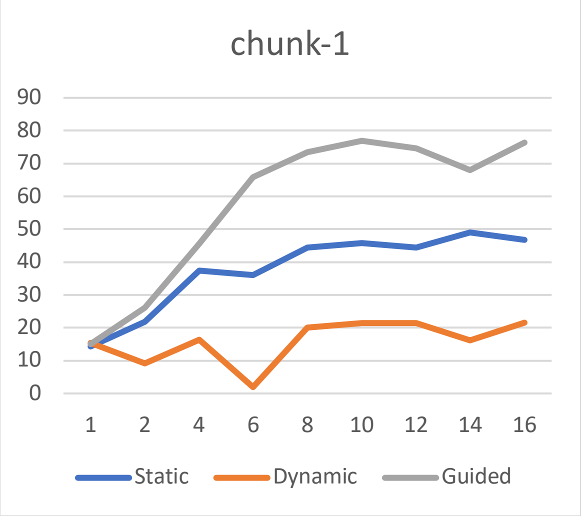


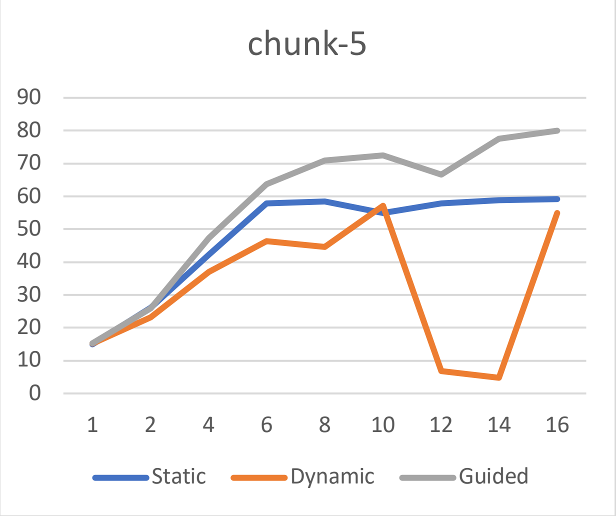


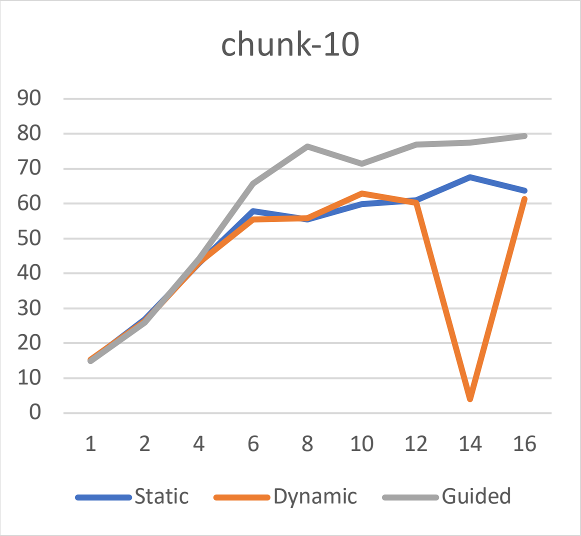


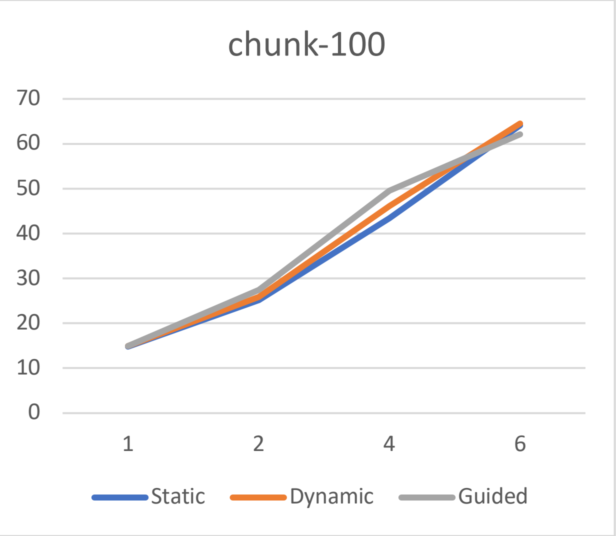
Performance graph

Execution time	Chunk size	1	2	4	6	8	10	12	14	16
Static	1	14.306	21.739	37.313	36.101	44.444	45.872	44.444	49.02	46.729
Dynamic		15.337	9.141	16.313	1.925	20.121	21.505	21.368	16.234	21.552
Guided		15.038	26.11	45.662	65.789	73.529	76.923	74.627	68.027	76.336
Static	5	14.948	26.042	42.194	57.803	58.48	54.945	57.803	58.824	59.172
Dynamic		15.106	23.202	37.037	46.296	44.643	57.143	6.711	4.769	54.945
Guided		15.267	25.907	47.393	63.694	70.922	72.464	66.667	77.519	80
Static	10	15.015	26.81	42.918	57.803	55.556	59.88	60.976	67.568	63.694
Dynamic		15.291	26.385	43.103	55.556	55.866	62.893	60.241	3.992	61.35
Guided		14.903	25.907	44.248	65.789	76.336	71.429	76.923	77.519	79.365
Static	100	14.815	25.189	43.478	64.103	54.945	54.645	60.976	66.225	65.789
Dynamic		14.948	25.907	46.083	64.516	78.74	85.47	78.74	74.627	78.125
Guided		14.948	27.397	49.505	62.112	79.365	73.529	74.074	69.444	74.627









The chunk size is 1 5 10 100. When the chunk size is 1, each thread is allocated one iteration at a time. Potentially higher scheduling overhead due to reduced data reuse between successive iterations. Therefore, it shows the lowest performance among chunk sizes of 1, 5, 10, and 100. For a chunk size of 5, each thread is allocated 5 consecutive iterations at a time. May provide better cache locality than chunk size 1. However, it has lower scheduling overhead than larger chunk sizes. For a chunk size of 10, each thread is allocated 10 consecutive iterations at a time. Scheduling overhead can be further reduced. However, different iterations can lead to more imbalanced workloads. For a chunk size of 100, each thread is allocated 100 consecutive iterations at a time. A large chunk size greatly reduces scheduling overhead by requiring a much smaller number of chunks to be distributed. Likewise, the Warlord's imbalance could become serious.

It can be seen that guided has excellent performance for each chunk size. Guided scheduling is an approach of starting with a large chunk size and gradually reducing the chunk size as the next more iterations are processed. It also uses smaller chunks to improve load balancing when there are fewer iterations left or fewer threads. Therefore, it has improved load balancing with reduced overhead compared to static and dynamic scheduling.

On the other hand, dynamic scheduling shows poor performance. Dynamic scheduling has higher overhead than static scheduling because synchronization and decision making are required during loop execution. You can optimize the overhead by increasing the chunk size, but other issues such as poor load balancing can occur if the number of chunks is much smaller than the number of threads or if the workload per iteration is not uniform. In the written code, dynamic scheduling shows lower performance than static scheduling due to high overhead.