COE 379L: Homework 7

Ashton Cole

March 19, 2024

1 Exercise 8: Speedup Tests

Table 1: Parallelization speedup tests for various problem sizes on Lonestar 6.

Vector Size	100	100,000	1,000,000	10,000,000	100,000,000
Threads	Elapsed Time (s)				
1	0.002	0.275	2.812	31.864	330.914
2	0.006	0.140	1.367	19.537	173.943
4	0.034	0.074	0.655	12.544	114.224
8	0.010	0.066	0.361	10.520	113.571
12	0.049	0.035	0.255	8.639	114.516
16	0.068	0.082	0.246	8.621	115.107

2 Exercise 9: Performance Improvements

Table 2: The impact of performance improvements on a large problem size.

Case	A	B	C	D	E	F
Threads	Elapsed Time (s)					
1	31.864	34.719	32.699	32.576	32.519	33.272
2	19.537	20.881	16.389	19.830	19.830	20.088
4	12.544	10.045	12.763	12.747	12.741	13.017
8	10.520	10.826	10.681	10.773	10.753	11.005
12	8.639	12.012	10.463	8.586	8.692	8.865
16	8.621	9.021	8.904	8.884	10.090	9.131

- A: Original program on Lonestar 6 at $n = 10,000,000$
- 9 (a) No Wait
  - B: Adding a `nowait` clause between the calculation and assignment loops
- 9 (b) Thread affinity
  - C: Setting the affinity to `sockets`
  - D: Setting the affinity to `cores`
  - E: Setting the affinity to `threads`
- 9 (c) First-Touch

– F: Taking advantage of "first-touch" memory location by allocating in parallel, affinity set to `cores`

Interestingly, none of the changes offer significant improvements over the original wallclock time.

1. For the `nowait` clause, it could be that, since the matrix is tridiagonal, each row takes about the same time to calculate a Jacobi iteration. Thus, none of the threads have to wait long before the others finish.

2. For the affinity, the issue might be that the program is run on a virtual machine, `vm-small`, limited to 16 cores. The actual physical compute nodes may have more than 16 cores per socket, so restricting threads to a single socket makes no difference. However, it is unclear why the program takes the same time when all threads are bound to a single hyperthread.

3. Again, since the compute nodes of the virtual machine could fit on a single socket, the first-touch technique of allocating memory in parallel may not make a difference for latency, because it is already lower in the first place.

3 Exercise 10: Scheduling Improvements

Table 3: The impact of performance improvements on a large problem size.

Case	n = 10,000,000	Static Scheduling	Dynamic Scheduling[1]
Threads	Elapsed Time (s)		
1	31.864	29.942	28.090
2	19.537	15.238	27.450
4	12.544	12.425	15.728
8	10.520	8.400	12.460
12	8.639	8.907	11.836
16	8.621	8.623	11.916

Interestingly, static scheduling shows improvements for lower thread counts, although it is supposed to be the default.

4 Exercise 11: Full Parallelization

An attempt was made to parallelize fully each iteration of the outer loop, using `single` regions for serial operations. However, the compiler throws an error because the outer loop uses a break statement to terminate. Even though the parallel region is kept within a single iteration of the loop, the break statement is not something that OpenMP can handle in parallel. Note that the outer loop is not parallelizable like the ineer for loops, because iterations are sequentially dependedn on each other, and the loop is not static.