HPC\_HW\_Chap4\_Part2

Chen Fang

**Exercise 4.10**

A barrier is implemented using semaphores in the trapezoidal-rule program by following the example on the book. The max elapsed time is updated by each thread with semaphores in the way that global sum is handled. My result is as follows:

|  |  |
| --- | --- |
| n=100000 | |
| pthread = 4 | |
|  | Time |
| 0 | 5.79E-05 |
| 1 | 5.79E-05 |
| 2 | 5.58E-05 |
| 3 | 5.82E-05 |
| max | 5.82E-05 |

**Exercise 4.16**

Assuming thread 0 compute from y[0] to y[1999], thread 2 computes from y[4000] to y[5999], false sharing will not occur if thread 0 and 2 are assigned to different processors because these y elements do not appear in the same cache line. So updating in one thread will not invalidate the cache line associated with another thread. The same applies to the case when thread 0 and 3 are assigned to difference processors.

**Programming 4.3**

Implementations of the trapezoidal rule with busy-waiting, mutexes and semaphores are attached in a separate file. The advantage of the busy-waiting method is it ensures ordered summation, but it also wastes CPU cycles while it keeps waiting. The mutex and semaphores are almost identical in this specific problem.

**Programming 4.4**

Implementation of timing the creation and destruction of threads is attached in a separate file. The result is shown as follows:

|  |  |  |
| --- | --- | --- |
| thread\_num | Total Time | Avg Time |
| 1 | 7.30E-05 | 7.30E-05 |
| 2 | 1.33E-04 | 6.64E-05 |
| 4 | 1.50E-04 | 3.75E-05 |
| 8 | 2.95E-04 | 3.69E-05 |
| 10 | 2.84E-04 | 2.84E-05 |
| 100 | 2.51E-03 | 2.51E-05 |
| 200 | 5.19E-03 | 2.59E-05 |
| 500 | 1.38E-02 | 2.75E-05 |

When the number of threads is small, the average time per operation is large possibly due to the overhead of the for-loop. However, it can be observed that the average time drops to around 2.6E-05 seconds and becomes stable regardless of the total number of threads.