**Student:** Hong Quan Doan

**Student ID:** 986956

**Day 04 – Feb 28**

**Exercise**

**OMP1**

Static scheduling where computation workload is regular (the same for each thread) such as basic image convolution, naive matrix computation. If the workload cannot be evenly (or almost evenly) distributed between threads, like triangular matrix, then the dynamic scheduling would perform better than static one.

**OMP2**

If the workload in the loop is complex computational, like logarithm with pow, comparing, … irregular, (chunks are not precomputed as in static scheduling) hence some overhead might appear, then using schedule (static, 1) will help in this case, like Mandelbrot set (https://en.wikipedia.org/wiki/Mandelbrot\_set).

**OMP3**

Guided scheduling is quite similar to dynamic scheduling but starts with large chunk size and decreases through time.

**OMP4**

Shared: n

Private: i, j

**OMP5**

I think static should be better, because the workload of computation is a bit simple, regular.

**OMP6**

1. Applicable for parallel, because the loop has a regular computation.
2. Could be data loop-carried dependence.
3. Applicable for parallel, but if foo() is complex, then need to review the algorithm.
4. Could be data loop-carried dependence.
5. Condition of exiting the loop can occur anytime, then the result will be different.
6. Applicable for parallel, because the loop has a regular computation.
7. Could be data loop-carried dependence.
8. Could be data loop-carried dependence.

**OMP7**

Outputs:

Schedule(static,1): Thread Schedule(static,1): Thread 2 executing iteration Schedule(static,1): Thread 20 executing iteration 0

Schedule(static,1): Thread 0

Schedule(static,1): Thread 21 executing iteration executing iteration 3

executing iteration 5

Schedule(static,1): Thread 2 executing iteration 8

Schedule(static,1): Thread 2 executing iteration 11

Schedule(static,1): Thread Schedule(static,1): Thread 0 executing iteration 61

Schedule(static,1): Thread 0 executing iteration 9

Schedule(static,1): Thread 0 executing iteration 122

executing iteration 14

Schedule(static,1): Thread 2 executing iteration 17

Schedule(static,1): Thread 0 executing iteration 15

Schedule(static,1): Thread 0 executing iteration 18

Schedule(static,1): Thread 1 executing iteration 4

Schedule(static,1): Thread 1 executing iteration 7

Schedule(static,1): Thread 1 executing iteration 10

Schedule(static,1): Thread 1 executing iteration 13

Schedule(static,1): Thread 1 executing iteration 16

Schedule(static,1): Thread 1 executing iteration 19

Default: Thread 1 executing iteration 7

Default: Thread 1 executing iteration 8

Default: Thread Default: Thread 0 executing iteration 0

Default: Thread 0 executing iteration 1

Default: Thread 0 executing iteration 2

Default: Thread 0 executing iteration 3

Default: Thread 0 executing iteration 4

Default: Thread 0 executing iteration 5

Default: Thread 0 executing iteration 6

1 executing iteration 9

Default: Thread Default: Thread 1 executing iteration 10

Default: Thread 1 executing iteration 11

Default: Thread 1 executing iteration 12

Default: Thread 12 executing iteration 14

Default: Thread 2 executing iteration 15

Default: Thread 2 executing iteration 16

Default: Thread 2 executing iteration 17 executing iteration 13

Default: Thread 2 executing iteration 18

Default: Thread 2 executing iteration 19

The differences in the output are about the sequence of the thread with the iteration.

* In default mode, the “i” variable of the iteration is divided evenly between threads
* In the static mode, each thread calls each “i” individually, not by a chunk.

**OMP8**

1. Yes, flow.
2. Yes, output.
3. No loop-carried dependence.
4. Yes, output.

**OMP9**

First thread writing on x[0] - Second thread writing on x[1]

Time: 0.663715

Time: 0.65062

Time: 0.675795

Time: 0.707577

Time: 0.675479

Time: 0.742284

Time: 0.705417

Time: 0.662121

Time: 0.688205

Time: 0.697868

First thread writing on x[0] - Second thread writing on x[99]

Time: 0.160858

Time: 0.165638

Time: 0.167106

Time: 0.175002

Time: 0.154401

Time: 0.161508

Time: 0.154282

Time: 0.15573

Time: 0.151237

Time: 0.151604