Indian Institute of Information Technology, Vadodara

Parallel Programming(CS403)

Prof. Reshmi Mitra

Name - Dilip Puri
ID - 201351014
Collaborator - Hemant Kumar(201352026)

Lab 03

Submission Date - September 1, 2016 **Deadline** - Aug 26, 4:00 PM

---

1. Familiarize yourself with the OpenMP code given in this handout.

2. Using the command in the section on "How to measure time for a block of C-code", find the time required in the thread function call in the previous lab assignment.

3. Write a C-code using OpenMP threads to create an unbalanced load using sleep command and omp_hello.c. The sample sleep times are given below:
   (a) thread-1: 10 sec, thread-2: 5 sec, thread-3: 20 sec, thread-4: 30 sec.
   (b) Measure the total time taken for the complete execution of code with and without the additional sleep command.

4. Write a threaded code using OpenMP on matrix multiplication:
   (a) Take overall execution time measurement using time command for different application size and thread count for the serial and parallel code.

	p=1	p=2	p=4	p=8
Vector Length = 100,000	0.0095	0.0124	0.0121	0.0102
Vector Length = 200,000	0.014	0.0166	0.0121	0.0163

$$Speedup = \frac{ExecutionTime(p)}{ExecutionTime(serialcode)}$$

	p=2	p=4	p=8
Vector Length = 100,000	1.31	1.27	1.07
Vector Length = 200,000	1.19	1.26	1.16

   (b) Observe `gnome-system-monitor` output as your fire up different thread counts.
   (c) Use the overall execution time measurements to plot and comment upon the speed-up. Verify that:
      i. Parallel execution time is more compared to the serial time when the application size is small.

ii. For large application size, initially the speedup increases as you increase the number of threads. However, for very large number of threads (e.g. 64 threads) your performance becomes much worse w.r.t. serial code and lower thread count.

5. **Multi-access threaded queue - using OpenMP** Implement a multi-access threaded queue with multiple threads inserting and multiple threads extracting from the queue. Use mutex-locks to synchronize access to this queue. Document the time for 1000 insertion and 1000 extractions each with 4 insertion threads (producers) and 4 extraction threads (consumers).(Buffer size = 250)

3 Multi-access threaded queue

1. Implement a multi-access threaded queue with multiple threads inserting and multiple threads extracting from the queue. Use mutex-locks to synchronize access to this queue. Document the time for 1000 insertion and 1000 extractions each with 4 insertion threads (producers) and 4 extraction threads (consumers).
2. Repeat above problem with condition variables (in addition to mutex locks). Document the time for the same test case as above. Comment on the difference in the times.