Course: CS72 - High Performance Computing
Dept. of CSE, Ramaiah Institute of Technology, Bengaluru
July-Dec 2020
www.tiny.cc/bhh

HPC- Assignment-1 (A-1)

A-1: POSIX Threads and OpenMp

Instructions:

1. Submission(archive) should contain report, code, screenshots and other files and mail 'usns.zip' to cs72.hpc @gmail.com
2. This is a team assignment. At most three students per team. One submission per team.
3. Bonus marks for creative problem solving. The due date is strictly applied. Late submissions will be penalized by 10% to 50% of the total marks.

A-1: POSIX Threads

1. ***Know Your System:***
   Answer the following by investigating your computing environment/system(use Multi-core system)
   (a) What are the L1, L2, L3 cache, and memory sizes?
   (b) Clock speed, the MFLOPS rating for your CPU?
   (c) Disk seek, latency, and transfer times (your local hard disk)?
   (d) How to know if your processor is 64b or 32b? Is your system 32 or 64 bit?
   (e) Available VM size for user processes. Can it be changed?
   (f) what is the Limits on the stack space, heap space and static area.
   (g) What is a memory leak and how can you detect one?
   (h) How can one tell how many page faults a process had?
2. **Hello World Program**:
   Create 5 threads with the pthread create() routine. Each thread prints a "Hello World!" message with thread id, and then terminates with a call to pthread exit()
3. **Using Condition Variables** :
   Build a pthread implementation to demonstrate using Condition Variables for :

   ```
   pthread_cond_wait and pthread_cond_signal
   ```

4. **DAXPY Loop:**
   The daxpy loop is the core of the benchmark. This loop is used to measure the performance. By using this loop we can observe how fast a certain machine can execute. Daxpy loop multiplies a vector by a scalar and adds it to another vector.
   D stands for Double precision, A is a scalar value, X and Y are one-dimensional vectors of size 216 each, P stands for Plus. The operation to be completed in one iteration i.e `X[i] = a*X[i] + Y[i]`.

```
void daxpy(double y[], double a, double x[], int n)
    {
    int i;
    for (i = 0; i &lt; n; i++)
    y[i] = a*x[i] + y[i];
    }
```

Answer the following :
a) Compare the speedup (in execution time) gained by increasing the number of threads for daxpy loop. Start from a 2 thread implementation.
(b) How many threads give the max speedup? *Note: speed up = (Execution time of an n-threaded program) / (Execution time of the uniprocessor implementation)*

A-1: OpenMp

5. **Hello World Program**:
   Create a default number of OpenMP threads. For each thread OpenMP can create, pass the thread id to a function. The function prints out a Hello World message and the thread id.

6. ***DAXPY Loop**:*
   Repeat the experiment from the previous question 4 for OpenMp implementation.