TEXAS STATE UNIVERSITY

The rising STAR *of Texas*

CS 7389D: HPC@SCALE
Assignment 1

Objective

The goal of this programming assignment is to help students (i) gain hands-on experience in writing OpenMP programs, (ii) understand the impact of common OpenMP runtime parameters on application performance, (iii) get an idea about the systematic approach for conducting performance study of a parallel application, (iv) learn about tools that will be useful beyond this course.

Submission Instructions

Please create a directory called "`assgn1`" in your git.txstate.edu repository (from lab1). It is important that you follow the naming convention exactly specified since automated scripts will retrieve your repositories. Submit a 1-3 page report that includes your experimental results by uploading your report under `assgn1`. Please use an IEEE 2 column template for your report.

Tasks

For all of the following tasks, please use `omp_get_wtime()` to time regions of interest. Please do not include the time for initialization or printing results into the effective computation time. You should try to get the best performance you can for compute_pi.c. Your assignment will be evaluated based on your discussions about the performance of your application, completion of all tasks, and quality of your report. Be creative in designing your plots, use bullet points to list your observations, mark plot axes clearly, and think publication quality.

1. Understanding underlying machine architecture:

   Use the `hwloc` tool on Stampede2 to gather information about the compute node. Please add a description of the node's architecture (number of sockets, number of cores per socket, different levels of cache, their sizes, and total amount of memory available on a node).

2. Parallelizing kernel using OpenMP:

   Parallelize the `compute_pi.c` file using OpenMP constructs. The parallel program should be compiled with "`-fopenmp`" and run with "`./compute_pi N`" where N defines the number of iterations in the integration. You can set the number of threads using OMP_NUM_THREADS environment variable. Please make sure to verify that the computed result matches with the result of the sequential application. In your report explain the following:

   a. Thread placement: For N=10000000, compare the performance of your parallel application across the four execution policies: OMP_PLACES=[core|socket] and OMP_PROC_BIND=[close|spread].

   b. OpenMP constructs: Name the OpenMP constructs you used to parallelize and why. Also, experiment with the following work sharing constructs – private, and shared. Which variables did you include for private and shared; why?

   c. Run the following experiments with N=10000000 and collect execution times: (i) varying number of threads from 1 to 16 in power of 2s (so, 1, 2, 4, …16), (ii) for number of threads = 16, compare static and dynamic work scheduling with 3 different chunk sizes [10 | 100 | 1000].

DEPARTMENT NAME
601 University Drive | Building and Suite | San Marcos, Texas 78666-XXXX
phone: 512.245.XXXX | *fax:* 512.245.XXXX | WWW.TXSTATE.EDU

d. Run these experiments using the batch_submission.sh script provided in Lab2. Use either Maverick2 or Stampede2 cluster for running your jobs. Please use the same system that you describe in question 1. For queue names and submit scripts, please refer to [2] and [3].

e. Plot execution times and explain your observations about how these two factors (number of threads, and scheduling policies) impact performance.

References that might be useful:

1. http://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-affinity.html

2. Maverick2 user guide: https://portal.tacc.utexas.edu/user-guides/maverick2

3. Stampede2: https://portal.tacc.utexas.edu/user-guides/stampede2