Kristy Nguyen
Dr. Ed Jorgensen
CS 219-1002: Computer Organization

CS 219 Assignment #11

Purpose: Become more familiar with concurrency methods and issues
Points: 100 pts → Part A 20 pts
Part B 40 pts

Assignment:

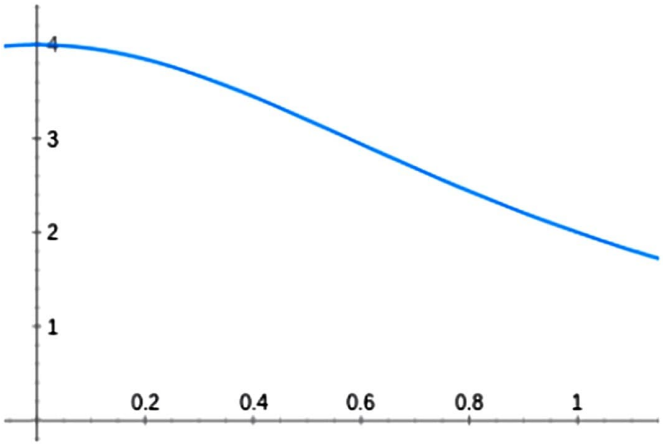
Part A:
Answer the following questions: (20 pts, 10 pts each)

- Given a 100-processors multiprocessor system and an application that has 15% of the code parallelized, what is the approximate effective speed-up?
- Given a 10-processors multiprocessor system and an application that has 85% of the code parallelized, what is the approximate effective speed-up?

Part B:

The value of π can be calculated by numerical integration of the function, $f(x) = \frac{4}{(1+x^2)}$

The value represents the area under the curve of the function for an interval of 0.0 to 1.0. To find an approximation of the area, the interval is divided into some number of n subintervals and each are in the subinterval in calculated. Then all of the areas are added up to give the final result. So, the larger the value of n, the more accurate the value of π.



In order to speed the calculations, it is possible to split the computations into groups and send each group to a different processor or core. Thus, the computations can be performed in parallel. When done, the individual group sums can be combines into a final answer.

We will look at two different ways to accomplish the parallel computations; message passing (MPI) and shared memory (threading). Compile and execute the provided programs; **PImpi.c** and **PIpth.cpp**. Compilation instructions are included in the comments. Review the code and examine the results.

For the **PImpi.c** program. Answer the following questions (no more than 1-2 sentences each):

- include a copy of the results (cut-and-paste)
- summarize basic approach to parallelism and the typical configuration
- list several benefits of the distributed approach
- list several dangers of the distributed approach

For the **PIpth.cpp** program. Answer the following questions (no more than 1-2 sentences each):

- include a copy of the results (cut-and-paste)
- summarize basic approach to parallelism and the typical configuration
- list several benefits of the pthread approach
- list several dangers of the pthread approach

Given a 100-processor system and an application where 15% can be parallelized, the max speed-up is:
1 / [(1-0.015) + 0.015/100] = 1.015
Thus, ~1 times speed-up is possible (at best).

Given a 10-processor system and an application where 85% can be parallelized, the max speeed-up is:
1 / [(1-0.85) + 0.85/10] = 4.255
Thus, ~4 times speed-up is possible (at best).

PImpi.c program:

```
MPI program results:
pi is approximately 3.1415926535897309, Error is 0.0000000000000622
```

PIpth.cpp program:

```
Hardware Cores: 4
Thread Count: 4
P-threads program results with 4 threads:
  pi is approximately 3.1415926535902168
  error is 0.0000000000004237
```

In parallelism, the processors are not getting much faster. In order to speed up the program, we use multiple processors. In parallel programming, there is a shared memory multiprocessor (SMP) with multiple identical processors that share memory and bus and General Purpose computing on Graphical Processing Units (GPGPU's) with NVIDIA graphics cards and a CUDA interface. There is limited scalability, limited float capabilities and may not be general purpose with multiple cores on one chip, but can be very effective for some applications.

For the distributed approach, there are disparete processes, disparete memory, separate CPU/cache/ memory/disk. In order for the distributed approach to work, it uses a Message Passing Interface that uses an MPI library and MPI function calls, which might take some time to set up extra computers. The distributed approach also relies on the networking technologies which can be a problem if the networking speed is slow. However, this is easily solvable with hypercube, a network technlology to connect multiple machines for parallel processing.

For the pthread approach, the threads are light weight processes with shared address spaces and full access to RAM. The benefit includes much faster communication time between threads via shared variables, but uncontrollable changing of shared variables can cause problems, such as incorrect results.