COMP605/CS605: Scientific Computing, Spring 2023
Assignment 1 - Due Tuesday 2/28/23

The assignment should be turned in handwritten. Precision and neatness are expected at the graduate level.

1. (20 points) Consider the following code:

```
double A[MAX][MAX], x[MAX], y[MAX];
...
/* Initialize A and x and assign y = 0 */
...
/* First pair of loops */
for (i = 0; i < MAX; i++)
    for (j = 0; j < MAX; j++)
        y[i] += A[i][j]*x[j];
...
/* assign y = 0 */
...
/* Second pair of loops */
for (j = 0; j < MAX; j++)
    for (i = 0; i < MAX; i++)
        y[i] += A[i][j]*x[j];
...
```

   Suppose that $MAX = 8$, the cache can store four cache lines (named 0,1,2,3) and each cache line can store four elements of the array.

   (a) How many misses occur in the reads of A in the first pair of nested loops? Explain in detail why.

   (b) How many misses occur in the reads of A in the second pair of nested loops? Explain in detail why.

2. (10 points) Suppose a program must execute 10^{12} instructions to solve a problem. Suppose a single processor can solve the problem in 10^6 seconds (about 11.6 days). Now, suppose this program has been parallelized for execution on a distributed-memory system. Assume that if the parallel system uses p processors, each processor will execute $10^{12}/p$ instructions and each processor will send $10^9(p-1)$ messages. Suppose, there is no additional overhead in executing the parallel program (the program will complete after each processor has executed all of its instructions, and sent all of its messages, and there will not be any delays due to things such as waiting for messages).

   (a) Suppose it takes 10^{-9} seconds to send a message. How long will it take the program to run with 1000 processors, if each processor is as fast as the single processor on which the serial program was run?

   (b) Suppose it takes 10^{-3} seconds to send a message. How long will it take the program to run with 1000 processors?

3. (10 points) Suppose a shared-memory system.

   (a) Suppose the system uses snooping cache coherence and write-back caches. Also suppose that core 0 has a variable x in its cache, and it executes the statement $x = 5$. Suppose that core 1 does not have x in its cache, and after core 0's update to x, core 1 tries to execute $y = x$. What value will be assigned to y? Why?

   (b) Suppose that the shared-memory system in the previous part uses a directory-based protocol. What value will be assigned to y? Why?

4. (20 points) Suppose the size of a problem is n and the run-time of a serial program is given by $T_{serial} = n^2$ (in microseconds).

   (a) Suppose the parallelization of this program has run-time $T_{parallel} = \frac{n^2}{p} + \log_2(p)$. Compute the speedups and efficiencies of this program for various values of n and p.

      i. Report your results in a table (with n as columns and p as rows) for $n = 10, 20, 40, 80, 160, 320$, and $p = 1, 2, 4, 8, 16, 32, 64, 128$.

      ii. What happens to the speedups and efficiencies as p is increased and n is held fixed?

      iii. What happens when p is fixed and n is increased?

   (b) Suppose that $T_{parallel} = \frac{T_{serial}}{p} + T_{overhead}$. Also suppose p is fixed.

      i. Show that $T_{overhead} < T_{serial}$, the parallel efficiency will increase as we increase n.

      ii. Show that $T_{overhead} > T_{serial}$, the parallel efficiency will decrease as we increase n.

5. (20 points) Suppose the problem size is n. Assume $T_{serial} = n$ and $T_{parallel} = \frac{n}{p} + \log_2(p)$ (in microseconds).

   (a) If we increase p by a factor of k, find the formula for how much we will need to increase n to maintain constant efficiency.

   (b) How much should we increase n by if we double the number of processes from 8 to 16?

   (c) Is the parallel program scalable?

6. (20 points) The Unix shell command *time* reports the user time, the system time, and the "real" time or total elapsed time. Suppose programmer A has defined the following C functions.

   - double utime(void); that reports the user time or time spent in user code and library functions (e.g., *sin, cos*) that do not need OS.
   - double stime(void); that reports the system time or time spent in functions that do need to use OS (e.g. *printf, scanf*).
   - double rtime(void); that reports the real time or total number of seconds.

   (a) What is the mathematical relation among the three function values? Assume the time it takes to call the functions is negligible).

   (b) On programmer A, any time an MPI process spends waiting for messages is not computed by either *utime* or *stime*, but the time is counted by *rtime*. Explain how programmer A can use these facts to determine whether an MPI process is spending too much time waiting for messages.

   (c) Programmer A has given programmer B his timing functions. However, programmer B has discovered that on her system, the time an MPI process spends waiting for messages is counted as user time. Furthermore, sending messages does not use any system time. Can programmer B use programmer A's functions to determine whether an MPI process is spending to much time waiting for messages? Explain your answer.