High Performance Computing

Homework #7

Due: Tuesday May 3 2016 by 11:59 PM

Email-based help Cutoff: 5:00 PM on Mon, May 2 2016

Maximum Points: 25

Submission Instructions

This homework assignment must be turned-in electronically via Canvas. Type in your responses to each question (right after the question in the space provided) in this document. You may use as much space as you need to respond to a given question. Once you have completed the assignment upload:

1. The MS-Word document (duly filled) and saved as a PDF file named with the convention `MUid.pdf` (example: **`raodm.pdf`**)

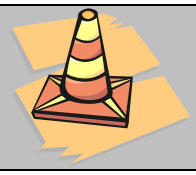
Note that copy-pasting from electronic resources, including lecture slides, is plagiarism. Consequently, you must suitably paraphrase the material in your own words when answering the following questions.

Name: Henry Ni

Objective

The objective of this homework is to review the necessary background information about collective communication operations in MPI in preparation for Final exam.

Read Section 6.6 from E-book "Introduction to Parallel Computing" (all students have free access to the electronic book). Links available off Syllabus page on Canvas.



Although the Safari E-books are available to all students there are only a limited number of concurrent licenses to access the books. Consequently, do not procrastinate working on this homework or you may not be able to access the E-books due to other users accessing books.

1. What is a virtual synchronization point? Explain with a suitable MPI call. How is it different from a barrier? [**1 point**]
   a. 2 Advantages

Collective operations in MPI act as virtual synchronization steps in a parallel program. The synchronization happens because all processes (in a communicator) must call the same function with same parameters. Furthermore, parallel programs typically should call these operations at nearly the same time. However, synchronization is only virtual because processes don't block.

An example of a virtual synchronization point is a call to `MPI_Bcast` function as shown below:

```
int i = 10;
MPI_Bcast(&i,1, MPI_INT, 0, MPI_COMM_WORLD);
```

In real synchronization (called barriers) processes block to achieve global (with a communicator) synchronization.

2. Briefly describe 2 significant differences between conceptual broadcast versus scatter operations [**1 points**]

Broadcast	***Scatter***
In a broadcast operation the same value is sent to all the processes.	In a scatter operations different values are sent to different processes.
In broadcast the amount of data received by each process is the same.	In scatter operations, the amount of data received by each process can vary.

3. Given the following MPI code fragment from process with rank 0, complete the complementary collective operation on other processes [**2 points**]

```
void doManagerTasks(const std::string& data) {
    int strSize = data.size() + 1;
    MPI_Bcast(&strSize, 1, MPI_INT,  0, MPI_COMM_WORLD);
    MPI_Bcast(&data[0], strSize, MPI_CHAR, 0, MPI_COMM_WORLD);
}
```

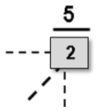
```
std::string recvData() {
    int strSize;
    MPI_Bcast(&strSize, 1, MPI_INT,  0, MPI_COMM_WORLD);

    std::string data(strSize, 0);

    MPI_Bcast(&data[0], strSize, MPI_CHAR, 0, MPI_COMM_WORLD);

    return data;
}
```

4. Illustrate the output from the following program (compiled to a file called `gather`) when it is executed using the command line(s) shown further below. [**3 points**]

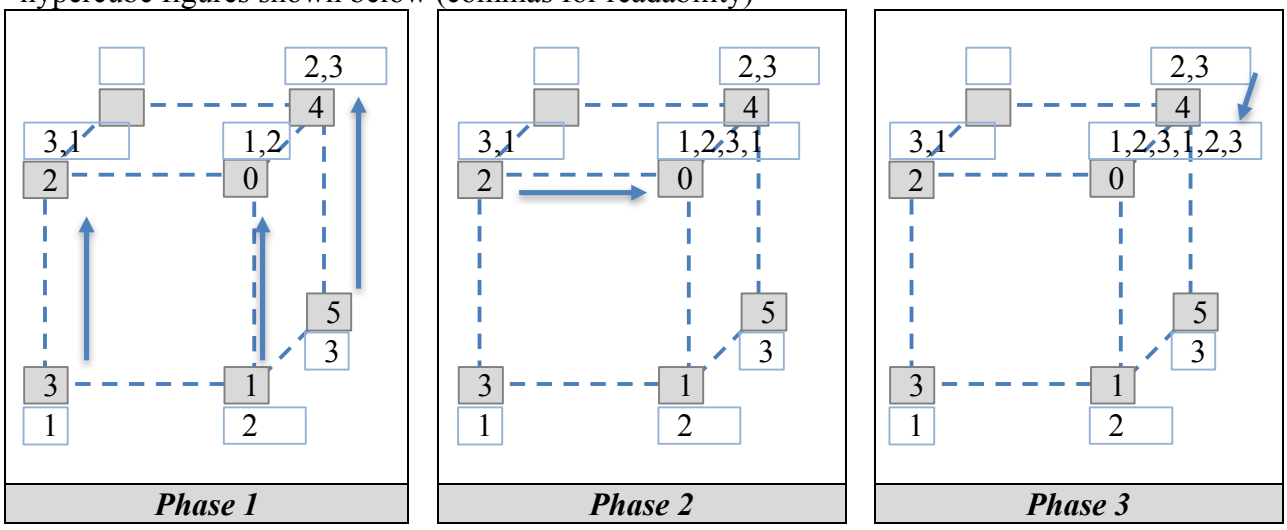
```
#include <mpi.h>
#include <iostream>
int main(int argc, char *argv[]) {
   int rank;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   rank %= 3;  // Ensure rank is valid
   std::string src  = "123456";
   std::string dest = src;
   MPI_Gather(&src[rank], 1, MPI_CHAR, &dest[0], 1, MPI_CHAR,
              0, MPI_COMM_WORLD);
   if (rank == 0) {
       std::cout << "Result = " << src << ", " <<  dest << std::endl;
   }
   return MPI_Finalize();
}
```

```
$ mpiexec -n 6 ./gather 1 2 3 4 5 6
```

Illustrate the **optimal** sequence of operations that MPI would perform on a hypercube by tracing the sequence of communication operations along each dimension in the appropriate order. At each phase: ➔ Draw directed arrows between nodes to indicate flow of data. ➔ Fill in the intermediate value(s) that would be present at each of the processes as the operations proceed. For example, if a process with rank 2 has an value of 5, then this would be represented as shown in the adjacent figure.



Trace the three phases of ***optimal*** communication operations along each dimension in the hypercube figures shown below (commas for readability)



The output from the program is: 123456,123123 **[1 point]**

5. As a HPC specialist you have been assigned the task to evaluate the effectiveness of two different approaches to parallelizing a serial program to eventually run on a cluster with a large number of processors. The two parallel approaches have been code named Impl-α and Impl-β. Your assistant has collated the following execution timings of for the different approaches (where **n** is number of inputs and **p** is number of processors used for parallel execution). Which one of the parallel implementations would you choose to adopt and why? [**3 points**]

n	Serial Timing (sec)	Parallel Timings (sec)			
		Impl-α		Impl-β	
		p=10	p=20	p=5	p=15
32	640	98	54	183	59
64	1280	150	85	284	95
128	2560	336	154	732	234

Table 1: Raw execution timings.

a. Given the raw execution timings, complete the following table for Impl-α:

Calculations for Impl-α				
n	p=10		p=20	
	Speedup	Efficiency	Speedup	Efficiency
32	6.53	.653	11.85	.593
64	8.53	.853	15.06	.753
128	7.61	.761	16.62	.831

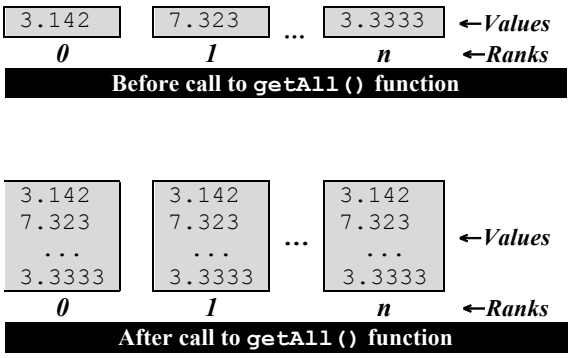
b. Given the raw execution timings, complete the following table for Impl-β:

Calculations for Impl-β				
n	p=5		p=15	
	Speedup	Efficiency	Speedup	Efficiency
32	3.49	.699	10.84	.723
64	4.50	.901	13.473	.898
128	3.49	.699	10.94	.729

c. Using all of the above data clearly indicate which implementation you would choose and briefly (1-2 sentences) discuss the motivation for your choice

Considering the data above, I would choose the first implementation due to its scalability. With each increase in data set size, the efficiency of the program remains near the same or increases, unlike Impl-β.

6. Given n ($0 < n < \infty$) MPI processes, assume each process performs a complex operation to compute a `double` value. Now complete the `getAll()` function that must send/receive the given `value` such that all processes obtain the values from each-and-every processes (as shown in the example in the adjacent figure). Note that the `getAll()` function must collate the values (in the same order on all processes) and return it as a single vector [**3 points**]




```
// myRank is rank of the MPI process which calls this method
std::vector<double> getAll(double value, int n, int myRank) {
   std::vector ret(n);
   MPI_Gather(&value, 1, MPI_DOUBLE, &ret[myRank], 1, MPI_DOUBLE,
0, MPI_COMM_WORLD);

   MPI_Bcast(&ret, n, MPI_DOUBLE, 0, MPI_COMM_WORLD);

   return ret;

}
```

7. The following `stats` method is called on **n** processes of an MPI program. It must be coded such that process with rank 0 (zero) prints the average and variance of the parameter `value` passed to each process (each process will have a different `value`). **Note that your implementation must use only collective communication and computation operations** (and cannot use `MPI_Probe`, `MPI_Send`, `MPI_Isend`, `MPI_Recv`, or `MPI_Irecv` function). [**3 points**]

$$mean\ (average)\colon \mu \qquad \mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$variance\colon \sigma^2 \qquad \sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

Sample Output

```
avg: 35.2, var: 1.5
```

```
// myRank is rank of the MPI process which calls this method
void stats(double value, int n, int myRank) {
      double avg;
      MPI_Reduce(&value, &avg, 1, MPI_DOUBLE, MPI_SUM, 0,
      MPI_COMM_WORLD);
      if (myRank == 0)
            avg /= n;

      MPI_Bcast(&avg, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

      double newVal = std::pow((value - avg), 2);
      double var;
      MPI_Reduce(&newVal, &var, 1, MPI_DOUBLE, MPI_SUM, 0,
      MPI_COMM_WORLD);

      if (myRank == 0)
            std::cout << "avg: " << avg << ", var: " << var/n;
}
```

1. Carefully read the following article related to teaching energy efficiency and then answer the following questions using the information from the paper (on-campus link): http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7163239

   a. What is the primary source of greenhouse gases? Is this information the authors' opinion or is it a fact? [**1 point**]

   Generating electricity accounts for 27% of global emissions and 31% in the US, as reported by the EPA (so it is fairly factual).

   b. On Thihanhe-2 (a large supercomputer), what does a mere 1% reduction in a program's runtime translate to in terms of energy saving for other purposes? [**1 point**]

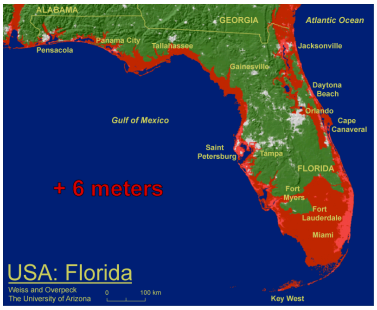
   1% of 24 megawatts is 240 kilowatts, enough to power thousands of homes.

   c. How does the author propose to improve efficiency using programming languages? Indicate if you have any experimental evidence (possibly collected in some homework in this course) to confirm/refute the authors' claim on efficiency differences in programming languages? [**1 point**]

   The more efficient the programs are, the less resources they will use. For example, the assignment in which we compared Python, Java and C++ showed that C++ ran faster (to save electricity as seen above).

d. With society becoming compute centric, you as a potential future programmer are in a unique position to have some influence. C++ strives for efficiency, resulting in high performance and energy efficient programs (but allegedly makes it hard for programmers to develop programs). This is in contrast to other languages like Java and Python. Choosing one of the topics below, discuss why programming in languages like Java/Python (**and not in C++**) would make the world a far better place after all. [**3 points**]

Topic #1: Let's make it Un-Bear-able!
Polar bears, walruses and other animals are evil (just look at the little guy) and must be made extinct. Coding in C++ will give these terrible mammals an unfair fighting chance while coding in other languages will help melt their house down, thereby making the world a better place.



Topic #2: Let's get Flo-Rid-Ya!
Costal states like Florida, California, Maine, Massachusetts are very annoying. Coding in C++ will slow down rise of sea levels prolonging the annoyance of these states while coding in other languages will help rapidly sink these states making the world a better place.

Topic #3: Let's not face it!
With increasing pollution kids and soon adults will require masks to breath. Finally, we will no longer have to tolerate look at faces of children. Coding in C++ will give future generations the unacceptable chance of not wearing breathing masks or not using bottled air and therefore we should code only in other languages.

Topic #4: Let's Pass the Buck$
C/C++ programs are hard to outsource/offshore. Other languages are easy to develop and can be offshored to other countries. That way we will no longer have to worry about energy or jobs and finally we will have time to catch up on all them parties and movies we missed due to annoying C++ homework.

Pick any one of the 4 topics and discuss/debate (need at least 6-8 convincing and cohesive sentences to earn full credit) how not programming in C++ will increase greenhouse gases/pollution thereby making our world a much better place and why all Miami computer science graduates should do so ☺

Have you ever had a time where you couldn't keep your kids from eating everything? Mom made a nice dinner, but those two kids just had to ruin their appetite with snacks just an hour before. You were too busy debugging your students' poorly written code and couldn't keep track of them to stop your fool kids from eating you out of your own house.

Just think, if the air pollution got to a point where everyone had to wear masks, when you go out to a nice family dinner they won't be able to gobble up all the snacks from street vendors on the way there. You'll finally be able to take the whole house to a sit-down meal where it's not just you and your wife eating at the end. No, with these masks the would-be full kids arrive hungry and ready to eat, while you can enjoy a tall domestic or if you're feeling fancy, the new hipster import that everyone's too cool to admit it's mainstream.