John Do
CSC 452-001
Dr. Droz
October 6, 2019

Lab Assignment: Lab 2

1.

	Lab2_1		
	10^7	10^5	10^3
Run Times	0.01	0	0

a. ***The link is provided at the end of this report. ***

b. The Unix shells commands that were used:
   i. "mpicc -o lab2_1a lab2_1a.c"
      1. This is to compile the code
   ii. "qsub lab2_1asubmit"
      1. This will load the submit script for lab 2_1a
   iii. "cat myjob.out"
      1. This will show the result of the experiment

```
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
```

c.

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
```

```
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
```

John Do
CSC 452-001
Dr. Droz
October 6, 2019

	Lab2_2	
Threads	4	8
Run Times	0.000013	0.0000184

2.

a. * The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_2 lab2_2.c"

1. This is to compile the code

ii. "qsub lab2_2submit"

1. This will load the submit script for lab 2_2

iii. "cat myjob.out"

1. This will show the result of the experiment

c.

d.

```
--------------------------------------------------
MPI run with 4 processes.
Hello, John from #0 of 4!
Time elaspsed is 0.000014 secs.
Hello, John from #1 of 4!
Time elaspsed is 0.000015 secs.
Hello, John from #2 of 4!
Time elaspsed is 0.000014 secs.
Hello, John from #3 of 4!
Time elaspsed is 0.000014 secs.
MPI run with 8 processes.
Hello, John from #0 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #1 of 8!
Time elaspsed is 0.000022 secs.
Hello, John from #2 of 8!
Time elaspsed is 0.000024 secs.
Hello, John from #3 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #4 of 8!
Time elaspsed is 0.000022 secs.
Hello, John from #5 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #6 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #7 of 8!
Time elaspsed is 0.000021 secs.
```

John Do
CSC 452-001
Dr. Droz
October 6, 2019

3. The advice that would be giving to someone running this code would be use MPI. MPI stands for message passing interface. It is meant for communicating between nodes. You can use OpenMP and MPI together because OpenMP is an API that supports multi-platform shared-memory parallel programming in C/C++. When the user is runs this code, they would compile it first by using mpicc. MPICC is a compiling function that would be best for running with scripts. After the compile is done, the user would run the submit script for the C program file. After that is done, check to see if there is any error in the "myjob.err" file. If there is not, then the submit script did the job correctly. Check the output in the file, "myjob.out", to see the computed runtimes with different vector lengths. With the computed runtimes, 10^7 with 4 threads would be 1,570,004,363 seconds whereas with 8 threads would be 1,569,984,364 seconds. For 10^5, 4 threads would be 1,570,038,274 seconds and 8 threads are 1,570,018,274 seconds. Lastly, 10^3 with 4 threads is 1,570,028,535 seconds and with 8 threads would be 1,570,018,536 seconds. The optimal threads would be 8 because the times that are computed are fairly close with each other. The reason it would be 4 because the sockets on the loni cluster is 2. If we take the optimal threads of 4 and multiply that with 2 then it would be 8. This would use less computational power compared to 16 and it computes the same run times as each other. Vector that has 10^7 is when you can see the speedup of the threads. This has a huge jump compared to 10^5 vector length. This has a differentiation of 13,911 seconds comparing from 10^7 with 4 processes and 10^5 with 4 processes.

4.

	Lab2_4	
Threads	4	8
Run Times	0.000088	0.000116

   a. *** The link is provided at the end of this report. ***
   b. The Unix shells commands that were used:
      i. "mpicc -o lab2_4 lab2_4.c"
         1. This is to compile the code
      ii. "qsub lab2_4submit"
         1. This will load the submit script for lab 2_4
      iii. "cat myjob.out"
         1. This will show the result of the experiment

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
MPI run with 4 processes.
Message at proc# 0:rHello, john from #0!

Time elapsed is 0.000088 seconds.
Message at proc# 1:rHello, john from #0!

Message at proc# 2:rHello, john from #0!

Message at proc# 3:rHello, john from #0!

MPI run with 8 processes.
Message at proc# 0:rHello, john from #0!

Time elapsed is 0.000116 seconds.
Message at proc# 1:rHello, john from #0!

Message at proc# 2:rHello, john from #0!

Message at proc# 3:rHello, john from #0!

Message at proc# 4:rHello, john from #0!

Message at proc# 5:rHello, john from #0!

Message at proc# 6:rHello, john from #0!

Message at proc# 7:rHello, john from #0!
```

c.

5.

	Lab2_5	
Threads	4	8
Run Times	0.000151	0.000224

a. * The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_5 lab2_5.c"
   1. This is to compile the code

ii. "qsub lab2_5submit"
   1. This will load the submit script for lab 2_5

iii. "cat myjob.out"
   1. This will show the result of the experiment

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
MPI run with 4 processes.
Message at proc # 1: Hello, John from # 0!

Message at proc # 2: Hello, John from # 1!

Message at proc # 3: Hello, John from # 2!

Message at proc # 0: Hello, John from # 3!

Time elapsed is 0.000151 seconds.
MPI run with 8 processes.
Message at proc # 1: Hello, John from # 0!

Message at proc # 2: Hello, John from # 1!

Message at proc # 3: Hello, John from # 2!

Message at proc # 4: Hello, John from # 3!

Message at proc # 5: Hello, John from # 4!

Message at proc # 6: Hello, John from # 5!

Message at proc # 7: Hello, John from # 6!

Message at proc # 0: Hello, John from # 7!
```

C. `Time elapsed is 0.000224 seconds.`

John Do
CSC 452-001
Dr. Droz
October 6, 2019

Lab Report: Lab 2

I. Introduction:

The purpose of this lab is to get the basic fundamentals of C program. As a class, we are using basic C program functions to create algorithms, parallelisms, and ring communications between workers. We will be doing experiments on the local machine and on the loni cluster to generate data to help us answer lab questions and plot run times from vectors with increasing length. This will help us get an understanding how data is being generated by algorithms and transmitted between worker nodes. The structure of this report is to implement different methods and algorithms in C program within our local machine or loni cluster. In this lab, we will be using C functions such as MPI and time function. MPI is a message passing interface for communicating between nodes. MPI can be used with OpenMP, which is an API that supports multi-platform shared-memory parallel programming in C/C++. The problems that MPI is meant to address is that it requires little hardware support, other than network and is usually used with distributed memory platforms. It is developed by a group of industrial partners to foster widespread use and portability. Also, the solution that MPI gives is it defines a communication domain, which is a set of processes that are allowed to communicate between themselves. After implementing different methods and algorithms, experiments will be conducted and reported on different values being inputted and get an understanding on how the outcome is generated. Results will be stated the report after conducting experiments and demonstrating how the outcome was gotten. At the end of the report will be the conclusions and future work, summarizing the whole report and determining how this lab will using the future conducting similar experiments.

II. Methods & Algorithms

The metrics of importance to our distributed or cloud systems are time, availability, computational cores. During this lab, we are timing the generated output from different algorithms and functions to determine different factors of the lab. For example, we are timing number of threads output to see if it is faster to generate output with 4 threads and 8 threads or if they would cause too much overhead. Another thing that we are using is the loni cluster, which should be available to us at all times to let worker nodes communicate with each other and pass information to the user. Computational cores and threads are a key factor to the user because it tells them how many workers, they are to generate the output to compute the run times. In this lab, we wrote a sequential C code program that sums a vector of vectors. The other experiments, we ran a simple program, “Hello World”, on the loni cluster using MPI. In lab 3, our vector should be split among the available processes, and each process should add its own components. We use an MPI reduction operator to sum the results on the master process. This should work in on any number of processes. In lab 4, there was a repeat of using lab 2 but using a MPI broadcast command in place of the for loop to broadcast the message to all the processes from the master node. Also, this lab used 4 and 8 threads. The last lab, we

John Do
CSC 452-001
Dr. Droz
October 6, 2019

implement the ring communications from the MPI lecture, where each thread sends its rank number to the process with rank one higher with 4 and 8 threads. All of these labs, we used time function from the C library to compute the time in each iteration or the time of 4 and 8 threads computing the output. As a class, we generated new scripts within the editor to run programs that will generate the times to run the compiled C program located in the home directory. This is will output the results and the time of each labs.

A. Lab2_1A:
    a. Line 1-4: C program libraries
    b. Line 6: define vector length
    c. Line 9: The vector length variable
    d. Line 11: The main function
    e. Line 14: Long integer type I, and longer integer vector
    f. Line 17: The clock variable
    g. Line 19: Floating point variable totsum
    h. Line 21: Seeds the random number generator with the time function with NULL
    i. Line 23: for loop where the vector, I, equals to the random integer modded by 100 plus 1
    j. Line 25: start the clock
    k. Line 26: for loops where the totsum + the floating variable vector and return value back to totsum
    l. Line 32: End the clock time
    m. Line 34: elapsed time calculation set to t
    n. Line 35: print out the results of the totsum and the time
    o. Line 40: quit after the program is done
B. Lab2_2:
    a. Line 1: C libraries
    b. Line 7: The main functions
    c. Line 12: initialize the MPI execution
    d. Line 14: determines the rank of the calling process in the communicator
    e. Line 16: Determines the size of the group associated with a communicator
    f. Line 18: Floating point variable start time
    g. Line 19: print out the results with the rank and size
    h. Line 22: stop the timer
    i. Line 24: print out the final time elapsed
    j. Line 26: terminates the calling MPI process’s execution environment
    k. Line 28: quit the program
C. Lab2_3:
    a. Line 1: C libraries
    b. Line 7: Define the vector length
    c. Line 9: vector variable
    d. Line 12: main function
    e. Line 15: long integer type I and long integer vector

f. Line 17: clock variable
g. Line 19: floating point variable totsum
h. Line 21: rank and size variable
i. Line 23: initialize the MPI execution
j. Line 25: determines the rank of the calling process in the communicator
k. Line 27: Determines the size of the group associated with a communicator
l. Line 29: Seeds the random number generator with the time function with NULL
m. Line 31: for loop where the vector, I, equals to the random integer modded by 100 plus 1
n. Line 33: start the clock
o. Line 35: for loops where the totsum + the floating variable vector and return value back to totsum
p. Line 40: Reduces values on all processes to a single value
q. Line 42: end the time of elapsed time on the calling processor
r. Line 44: print the final results of the rank and locsum
s. Line 46: if the rank is 0, stop the timer and print the total sum and time elapsed
t. Line 53: terminates the calling MPI process's execution environment
u. Line 55: quit the program

D. Lab2_4:
   a. Line 1: C libraries
   b. Line 6: main function
   c. Line 9: rank and size variables
   d. Line 12: message with 22 characters
   e. Line 14: initialize the MPI execution
   f. Line 16: determines the rank of the calling process in the communicator
   g. Line 18: Determines the size of the group associated with a communicator
   h. Line 20: floating variable with the elapsed time on the calling processor
   i. Line 23-24: if the rank is 0, then copies the string pointed to the destination
   j. Line 26: Broadcasts a message from the process with rank "root" to all other processes of the communicator
   k. Line 28: Prints the message with rank and message
   l. Line 30: floating variable with the elapsed time on the calling processor
   m. Line 32-33: if the rank is 0, stop the timer and print the total sum and time elapsed
   n. Line 35: terminates the calling MPI process's execution environment
   o. Line 37: quit the program

E. Lab2_5:
   a. Line 1: C libraries
   b. Line 6: main function
   c. Line 10: rank and size variables with type 99
   d. Line 12: message 1 and 2 with both 23 characters
   e. Line 14: Structure that represents the status of the received message
   f. Line 16: initialize the MPI execution

g. Line 18: determines the rank of the calling process in the communicator
h. Line 20: Determines the size of the group associated with a communicator
i. Line 22: floating variable with the elapsed time on the calling processor
j. Line 24: sends a formatted string output to a string pointed to the rank
k. Line 26-30: if the rank is 0, send the message and receive the message then print out message with the rank and the final message
l. Line 32-36: receive the message, print the message with the rank and final output then send m2
m. Line 38: floating variable with the time elapsed time on the calling processor
n. Line 40-41: if the rank is 0, print the time elapsed
o. Line 43: terminates the calling MPI process's execution environment
p. Line 45: quit the program

III. Experiments:

Test Matrix

Labs	Description	Tools
Lab2_1a	This experiment, we write a sequential C code program that sums a vector of values vect1[i]. Compile and run the code on a compute node on the LONI cluster. Report the run times.	HW: Loni Cluster SW: Program idle, Cyberduck, Vim, XQuartz terminal. Program Language: C
Lab2_2	Run hello world from your username, based on the code from the lecture. This program will be run on the LONI cluster using MPI. This will run on 4 and 8 threads. Report the run times.	HW: Loni Cluster SW: Program idle, Cyberduck, Vim, XQuartz terminal. Program Language: C
Lab2_3	In Lab2_3, we wrote an MPI code to sum a vector of values vect1[i]. The results should be placed in a variable called totsum. The vector should be split among the available processes, and each process should add its own components. Use MPI reduction operator to sum the results on the master process. This should work on any number of process. Compute the run time with different vector lengths. Then	HW: Loni Cluster SW: Program idle, Cyberduck, Vim, XQuartz terminal. Program Language: C

John Do
CSC 452-001
Dr. Droz
October 6, 2019

	we answer questions related to it.	
Lab2_4	In this experiment, we will repeat question 2 using an MPI broadcast command in place of the for loop to broadcast the message to all the processes from the master node. This will also use 4 and 8 threads. Report the run times.	HW: Loni Cluster SW: Program idle, Cyberduck, Vim, XQuartz terminal. Program Language: C
Lab2_5	In lab2_5, we implemented a ring communication from the MPI lecture, where each thread sends its rank number to the process with rank one higher. This will run on 4 and 8 threads. Report the run times.	HW: Loni Cluster SW: Program idle, Cyberduck, Vim, XQuartz terminal. Program Language: C

File Name	Np	Threads	#tests
Lab2_1a	**10^3, 10^5 , 10^7**	**4**	**3**
Lab2_2	**None**	**4 and 8**	**2**
Lab2_3	**10^3, 10^5 , 10^7**	**4 and 8**	**2**
Lab2_4	**None**	**4 and 8**	**5**
Lab2_5	**None**	**4 and 8**	**5**

A. The utility that was provided is called a “loni cluster”. This cluster is located at LSU. We acquire access to it by requiring permissions from our instructor to grant us access for the loni with some allocations. After the permission was granted, we go on our mobaxterm or xquartz (Mac Users) and login with our credential with x11 forwarding in order to display certain things. Once in the loni cluster, we are allowed to run simple experiments with their resources. To run simple experiments on the loni cluster, we created a submit script such as we in did in all the lab experiments so far to run on the compiled C program in order to generate the output. The program was called by the script, it was transferred over to the loni by using a software that can use SFTP, which is a secure file transfer protocol. After the script is done, we will use “cat” on the file “myjob.out” to show the result of our compiled program.

John Do
CSC 452-001
Dr. Droz
October 6, 2019

	Lab2_1		
	10^7	10^5	10^3
Run Times	0.01	0	0

IV. Results:

Lab2_1a:

1.

a. *The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_1a lab2_1a.c"

1. This is to compile the code

ii. "qsub lab2_1asubmit"

1. This will load the submit script for lab 2_1a

iii. "cat myjob.out"

1. This will show the result of the experiment

c.

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
The total sum is 505028671.000000.
Time elapsed is 0.010000 secs.
```

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
The total sum is 5052466.000000.
Time elapsed is 0.000000 secs.
```

```
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
The total sum is 51089.000000.
Time elapsed is 0.000000 secs.
```

John Do
CSC 452-001
Dr. Droz
October 6, 2019

Lab2_2:

	Lab2_2	
Threads	4	8
Run Times	0.000013	0.0000184

1.

a. * The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_2 lab2_2.c"

1. This is to compile the code

ii. "qsub lab2_2submit"

1. This will load the submit script for lab 2_2

iii. "cat myjob.out"

1. This will show the result of the experiment

c.

```
--------------------------------------------------------
MPI run with 4 processes.
Hello, John from #0 of 4!
Time elaspsed is 0.000014 secs.
Hello, John from #1 of 4!
Time elaspsed is 0.000015 secs.
Hello, John from #2 of 4!
Time elaspsed is 0.000014 secs.
Hello, John from #3 of 4!
Time elaspsed is 0.000014 secs.
MPI run with 8 processes.
Hello, John from #0 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #1 of 8!
Time elaspsed is 0.000022 secs.
Hello, John from #2 of 8!
Time elaspsed is 0.000024 secs.
Hello, John from #3 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #4 of 8!
Time elaspsed is 0.000022 secs.
Hello, John from #5 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #6 of 8!
Time elaspsed is 0.000021 secs.
Hello, John from #7 of 8!
Time elaspsed is 0.000021 secs.
```

Lab2_3:

1. The advice that would be giving to someone running this code would be use MPI. MPI stands for message passing interface. It is meant for communicating between nodes. You can use OpenMP and MPI together because OpenMP is an API that supports multi-platform shared-memory parallel programming in C/C++. When the user is runs this code, they would compile it first by using mpicc. MPICC is a compiling function that would be best for running with scripts. After the compile is done, the user would run the submit

script for the C program file. After that is done, check to see if there is any error in the "myjob.err" file. If there is not, then the submit script did the job correctly. Check the output in the file, "myjob.out", to see the computed runtimes with different vector lengths. With the computed runtimes, 10^7 with 4 threads would be 1,570,004,363 seconds whereas with 8 threads would be 1,569,984,364 seconds. For 10^5, 4 threads would be 1,570,038,274 seconds and 8 threads are 1,570,018,274 seconds. Lastly, 10^3 with 4 threads is 1,570,028,535 seconds and with 8 threads would be 1,570,018,536 seconds. The optimal threads would be 8 because the times that are computed are fairly close with each other. The reason it would be 4 because the sockets on the loni cluster is 2. If we take the optimal threads of 4 and multiply that with 2 then it would be 8. This would use less computational power compared to 16 and it computes the same run times as each other. Vector that has 10^7 is when you can see the speedup of the threads. This has a huge jump compared to 10^5 vector length. This has a differentiation of 13,911 seconds comparing from 10^7 with 4 processes and 10^5 with 4 processes.

Lab2_4:

1.

	Lab2_4	
Threads	4	8
Run Times	0.000088	0.000116

a. * The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_4 lab2_4.c"
   1. This is to compile the code

ii. "qsub lab2_4submit"
   1. This will load the submit script for lab 2_4

iii. "cat myjob.out"
   1. This will show the result of the experiment

c.

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
MPI run with 4 processes.
Message at proc# 0:rHello, john from #0!

Time elapsed is 0.000088 seconds.
Message at proc# 1:rHello, john from #0!

Message at proc# 2:rHello, john from #0!

Message at proc# 3:rHello, john from #0!

MPI run with 8 processes.
Message at proc# 0:rHello, john from #0!

Time elapsed is 0.000116 seconds.
Message at proc# 1:rHello, john from #0!

Message at proc# 2:rHello, john from #0!

Message at proc# 3:rHello, john from #0!

Message at proc# 4:rHello, john from #0!

Message at proc# 5:rHello, john from #0!

Message at proc# 6:rHello, john from #0!

Message at proc# 7:rHello, john from #0!
```

Lab2_5:

1.

	Lab2_5	
Threads	4	8
Run Times	0.000151	0.000224

a. * The link is provided at the end of this report. *

b. The Unix shells commands that were used:

i. "mpicc -o lab2_5 lab2_5.c"
   1. This is to compile the code

ii. "qsub lab2_5submit"
   1. This will load the submit script for lab 2_5

iii. "cat myjob.out"
   1. This will show the result of the experiment

John Do
CSC 452-001
Dr. Droz
October 6, 2019

```
MPI run with 4 processes.
Message at proc # 1: Hello, John from # 0!

Message at proc # 2: Hello, John from # 1!

Message at proc # 3: Hello, John from # 2!

Message at proc # 0: Hello, John from # 3!

Time elapsed is 0.000151 seconds.
MPI run with 8 processes.
Message at proc # 1: Hello, John from # 0!

Message at proc # 2: Hello, John from # 1!

Message at proc # 3: Hello, John from # 2!

Message at proc # 4: Hello, John from # 3!

Message at proc # 5: Hello, John from # 4!

Message at proc # 6: Hello, John from # 5!

Message at proc # 7: Hello, John from # 6!

Message at proc # 0: Hello, John from # 7!
```

c. `Time elapsed is 0.000224 seconds.`

V. Conclusions & Future Work

From the results being shown, each experiment was conducting MPI communication. MPI is a message passing interface for communicating between nodes. MPI can be used with OpenMP, which is an API that supports multi-platform shared-memory parallel programming in C/C++. The results from this portion indicates that it is getting slightly faster when the vector is increasing in LAB2_3. We use MPI code to sum a vector of values and add timing commands to report the times using 4 and 8 threads. Since we are using the loni cluster, we can use different number of threads and length vectors such as 10^3 , 10^5 , and 10^7 . In the end the results are close with testing different vectors. 4 and 8 threads came close in run time when experimenting. The difference was only 13,000 seconds. LAB2_5 is implementing a ring communication, like the last lab, with MPI. This is where each thread sends it rank number to the process one higher. In future work, this will help us program algorithms using MPI in C program language and learn how input data values are being transmitted and broadcast. This also help us understand the distribution of jobs within the parallel computing. This will be beneficial on working with complex algorithm using MPI that require larger number of data being inputted and visualizing how the data is going to be outputted. This will help us during future tasks of using the loni cluster because of parallelism between number of threads and workers that is provided.

GitHub Link: https://github.com/jonathanmdo/CSC_452_Labs/tree/Lab2