Lab Report 2

Parallelism with MPI

CSC 452-001

Andrew Almond

10/08/19

Introduction

For the majority of the time of computers, sequential programming dominated most of all computational programs. However, today, distributed systems have been making a dramatic climb in use. Because of this, parallel programs are in high demand now. One example of this is MPI that this lab covered. Its use allows developers to write code for one, ten or hundreds of programs to be executed on various grand amounts of systems at the same time. This is done through an approach covered before SPMD (single data, multiple data). Using MPI through this lab, a greater understanding of communication between threads, reduction operations, SPMD programs and more can be made.

Methods & Algorithms

Question 1:

Sequential C Code: In the given C program, a vector is made of NP = 10 million numbers valued (random# % 100)+1. The program then starts a timer while the total sum is added up. The total time is then made by the difference in the end time and start time divided by the # of clocks per second.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NP 10000000
int vect1[NP];

int main(int argc, char *argv[]){
        unsigned long int i, n=(unsigned long int)NP;
        clock_t start, end;
        double totsum = 0.0;
        srand(time(NULL));

        for (i=0; i<n; i++) vect1[i] = (rand() % 100 + 1);
        start = clock();

        for (i=0; i<n; i++){
                totsum += (double) vect1[i];
        }
        end = clock();
        double t = (double) (end-start)/CLOCKS_PER_SEC;
        printf("Time elasped is %f secs.\n",t);
        return 0;
```

Question 2:

MPI Hello World: In this MPI program, we include MPI routines to initialize the parallel threads; have each thread find its own rank and the size of the pool, then print its message. The "MPI_Barrier" command acts as a wall to prevent the first done threads from ending until the last ones finish so that a completion time can be collected. This then prints it at command line and terminates the program.

```c
#include <stdio.h>
#include <time.h>
#include "mpi.h"
int main(int argc, char *argv[]) {
    int rank, size;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    double start = MPI_Wtime();
    printf("Hello, Drew from #%d of %d!\n",rank,size);
    MPI_Barrier(MPI_COMM_WORLD);
    double end = MPI_Wtime();
    if (rank == 0)
        printf("Time on rank #%d is %f!\n",rank,end-start);
    MPI_Finalize();
    return 0;
}
```

Question 3:

MPI Distributed Summation: In this MPI program, like program 1, an array of 10 million integers is initialized across each parallel thread given through a striped manner. The local sum (locsum) is computed with a "for" loop for each thread where it iterates by the pool size. Using the "MPI_Reduce" (reduction) function, each thread will combine its result with the rest as the "MPI_SUM" function. All of the local sums, the total sum and the completion time are printed before exiting the loop.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "mpi.h"
#define NP 10000000
int vect1[NP];

int main(int argc, char *argv[]){
        unsigned long int i, n=(unsigned long int)NP;
        double start, end;
        double totsum,locsum = 0.0;
        int rank,size;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);

        srand(time(NULL)+rank);

        for (i=rank; i<n; i+=size) vect1[i] = (rand() % 100 + 1);
        start = MPI_Wtime();

        for (i=rank; i<n; i+=size){
                locsum += (double) vect1[i];
        }
        MPI_Reduce(&locsum,&totsum,1,MPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD);
        end = MPI_Wtime();

        printf("Local Sum on proc#%2d is %f.\n",rank,locsum);

        if (rank==0){
                double t = (end-start);
                printf("The total sum is %f.\n",totsum);
                printf("Time elasped is %f secs.\n",t);
        }
        MPI_Finalize();
        return 0;
```

Question 4:

Distributed Broadcast: Unlike the previous programs, this one doesn't have each thread print off their own ranks. In this case, the thread with rank 0 will broadcast its message to the remaining threads. With this, the other threads will follow and print the broadcasted message before the pool ends.

```c
#include <stdio.h>
#include <time.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
    int rank, size;
    char message[22];

    MPI_INIT(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);



    double start = MPI_Wtime();

    if (rank == 0)
        strcpy(message,"Hello, Drew from #0!\n");

    MPI_Bcast(message, 22, MPI_CHAR, 0,MPI_COMM_WORLD);
    printf("Message at proc#%2d: %.21s\n",rank,message);

    double end = MPI_Wtime();


    if (rank == 0)
    printf("Time elapsed is %f seconds. \n",end-start);


    MPI_Finalize();
    return 0;
```

Question 5:

Distributed MPI Ring: This MPI program utilizes ring communications. However, deadlock should be avoided. To prevent deadlock, thread 0 will send its message to thread 1 upon starting before it tries to receive. All the others will receive their message before they receive. The messages will be received from thread #(n-1), printed off and sent to thread #(n+1)mod0. Once all this is over, the final execution time is displayed.

```c
#include <stdio.h>
#include <time.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
    int rank, size, type = 99;

    char m1[23], m2[23];
    MPI_Status status;

    MPI_INIT(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);



    double start = MPI_Wtime();

    sprintf(m2, "Hello, Droz  from #%2d!\n",rank);


    if (rank == 0) {
    MPI_Send(m2, 23, MPI_CHAR, 1, type, MPI_COMM_WORLD);
    MPI_Recv(m1, 23, MPI_CHAR, size-1, type, MPI_COMM_WORLD, &status);

        printf("Message at proc #%2d: %.22s\n",rank,m1);

    } else {

    MPI_Recv(m1, 23, MPI_CHAR, rank-1, type, MPI_COMM_WORLD, &status);
    printf("Message at proc #%2d: %.22s\n",rank,m1);
    MPI_Send(m2, 23, MPI_CHAR, (rank+1)%size, type, MPI_COMM_WORLD);

    }

    double end = MPI_Wtime();


    if (rank == 0)
    printf("Time elapsed is %f seconds. \n",end-start);


    MPI_Finalize();
    return 0;
}
```

Experiments

All of the following tests were run on the LONI cluster with 1 node, 20 physical cores and 64 GB of RAM. All lab programs but program 1 required changes in one variable, which was the number of threads to utilize.

File Name	Variables	Job Script	Job Output
lab2_1a.c	N_{points} = 1E7	submit_lab2_1a	myjob1a.out
lab2_2.c	$N_{threads}$ = 4, 8	submit_lab2_2	myjob2.out
lab2_3.c	$N_{threads}$ = 4, 8	submit_lab2_3	myjob3.out
lab2_4.c	$N_{threads}$ = 4, 8	submit_lab2_4	myjob4.out
lab2_5.c	$N_{threads}$ = 4, 8	submit_lab2_5	myjob5.out

Results

Before running every program besides the one from question 1 through the LONI cluster using "qsub script_name", the code should be compiled using "mpicc -o lab2_x lab2_x.c".

Question 1:

```
----------------------------------------
Running PBS prologue script
----------------------------------------
User and Job Data:
----------------------------------------
Job ID:    703903.qb3
Username:  m3jpuv
Group:     latechusers
Date:      07-Oct-2019 22:08
Node:      qb105 (23757)
----------------------------------------
PBS has allocated the following nodes:

qb105

A total of 20 processors on 1 nodes allocated
----------------------------------------------
Check nodes and clean them of stray processes
----------------------------------------------
Checking node qb105 22:08:28
Done clearing all the allocated nodes
------------------------------------------------------
Concluding PBS prologue script - 07-Oct-2019 22:08:28
------------------------------------------------------
Time elasped is 0.020000 secs.
------------------------------------------------------
Running PBS epilogue script    - 07-Oct-2019 22:08:29
------------------------------------------------------
Checking node qb105 (MS)
Checking node qb105 ok
------------------------------------------------------
Concluding PBS epilogue script - 07-Oct-2019 22:08:30
------------------------------------------------------
Exit Status:     0
Job ID:          703903.qb3
Username:        m3jpuv
Group:           latechusers
Job Name:        myjob0
Session Id:      23756
Resource Limits: ncpus=1,neednodes=1:ppn=20,nodes=1:ppn=20,walltime=01:00:00
Resources Used:  cput=00:00:00,mem=0kb,vmem=0kb,walltime=00:00:03
Queue Used:      workq
Account String:  loni_cyen405
Node:            qb105
Process id:      24366
------------------------------------------------------
```

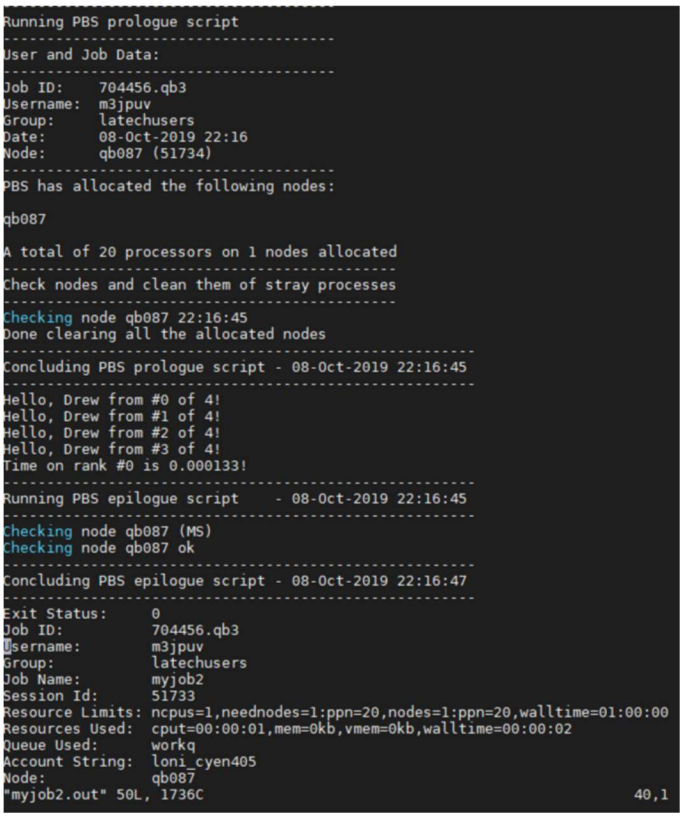
Submit Script:

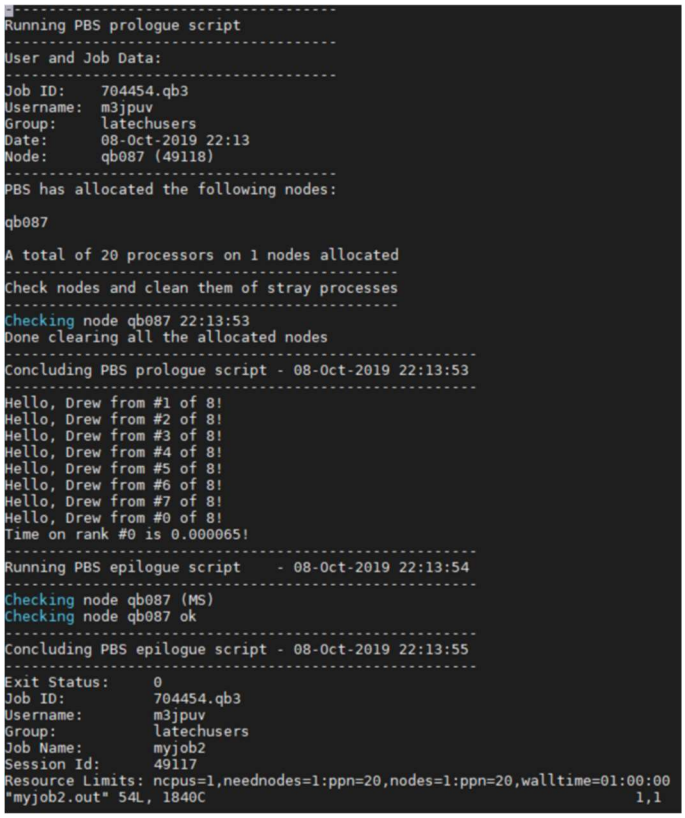
```
1  #!/bin/bash
2  #PBS -l nodes=1:ppn=20
3  #PBS -l walltime=01:00:00
4  #PBS -N myjob1
5  #PBS -o myjob1.out
6  #PBS -e myjob1.err
7  #PBS -q workq
8  #PBS -A loni_cyen405
9  #PBS -m e
10 #PBS -M ama067@latech.edu
11 work="/home/m3jpuv/lab_2"
12 CFILE="lab2_1a"
13 cd $work
14 ./$CFILE
```

t	0.02

Question 2:

For the solutions to questions 2 through 5, the submit script should be altered on the value of np on the bottom like between 4 threads or 8 threads.





Submit Scripts:

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob2
#PBS -o myjob2.out
#PBS -e myjob2.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_2"
cd $work
mpirun -np 4 -machinefile $PBS_NODEFILE ./$CFILE
```

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob2
#PBS -o myjob2.out
#PBS -e myjob2.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_2"
cd $work
mpirun -np 8 -machinefile $PBS_NODEFILE ./$CFILE
```

	4 Threads	8 Threads
t	0.000133	0.000065

Question 3:

```
Running PBS prologue script
----------------------------------------
User and Job Data:
----------------------------------------
Job ID:    704460.qb3
Username:  m3jpuv
Group:     latechusers
Date:      08-Oct-2019 22:19
Node:      qb113 (50754)
----------------------------------------
PBS has allocated the following nodes:

qb113

A total of 20 processors on 1 nodes allocated
----------------------------------------
Check nodes and clean them of stray processes
----------------------------------------
Checking node qb113 22:19:11
Done clearing all the allocated nodes
------------------------------------------------------
Concluding PBS prologue script - 08-Oct-2019 22:19:11
------------------------------------------------------
Local Sum on proc# 1 is 126216578.000000.
Local Sum on proc# 3 is 126240643.000000.
Local Sum on proc# 2 is 126211001.000000.
Local Sum on proc# 0 is 126310093.000000.
The total sum is 504978315.000000.
Time elasped is 0.004615 secs.
------------------------------------------------------
Running PBS epilogue script    - 08-Oct-2019 22:19:12
------------------------------------------------------
Checking node qb113 (MS)
Checking node qb113 ok
------------------------------------------------------
Concluding PBS epilogue script - 08-Oct-2019 22:19:13
------------------------------------------------------
Exit Status:     0
Job ID:          704460.qb3
Username:        m3jpuv
Group:           latechusers
Job Name:        myjob3
Session Id:      50753
Resource Limits: ncpus=1,neednodes=1:ppn=20,nodes=1:ppn=20,walltime=01:00:00
Resources Used:  cput=00:00:01,mem=0kb,vmem=0kb,walltime=00:00:03
Queue Used:      workq
Account String:  loni_cyen405
"myjob3.out" 51L, 1837C                                       1,1
```

```
Running PBS prologue script
----------------------------------------
User and Job Data:
----------------------------------------
Job ID:    703907.qb3
Username:  m3jpuv
Group:     latechusers
Date:      07-Oct-2019 22:15
Node:      qb105 (29147)
----------------------------------------
PBS has allocated the following nodes:

qb105

A total of 20 processors on 1 nodes allocated
----------------------------------------
Check nodes and clean them of stray processes
----------------------------------------
Checking node qb105 22:15:36
Done clearing all the allocated nodes
------------------------------------------------------
Concluding PBS prologue script - 07-Oct-2019 22:15:36
------------------------------------------------------
Local Sum on proc# 1 is 63137008.000000.
Local Sum on proc# 5 is 63122906.000000.
Local Sum on proc# 3 is 63092624.000000.
Local Sum on proc# 6 is 63154205.000000.
Local Sum on proc# 7 is 63208820.000000.
Local Sum on proc# 4 is 63136294.000000.
Local Sum on proc# 2 is 63143352.000000.
Local Sum on proc# 0 is 63190199.000000.
The total sum is 505185408.000000.
Time elasped is 0.007412 secs.
------------------------------------------------------
Running PBS epilogue script    - 07-Oct-2019 22:15:37
------------------------------------------------------
Checking node qb105 (MS)
Checking node qb105 ok
------------------------------------------------------
Concluding PBS epilogue script - 07-Oct-2019 22:15:39
------------------------------------------------------
Exit Status:     0
Job ID:          703907.qb3
Username:        m3jpuv
Group:           latechusers
Job Name:        myjob0
Session Id:      29146
Resource Limits: ncpus=1,neednodes=1:ppn=20,nodes=1:ppn=20,walltime=01:00:00
Resources Used:  cput=00:00:02,mem=0kb,vmem=0kb,walltime=00:00:02
Queue Used:      workq
Account String:  loni_cyen405
Node:            qb105
Process id:      29772
------------------------------------------------------
```

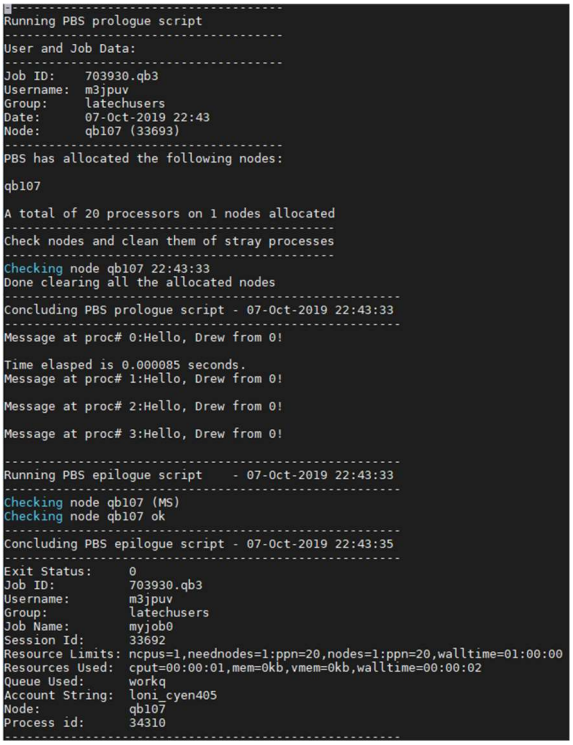
Submit Scripts:

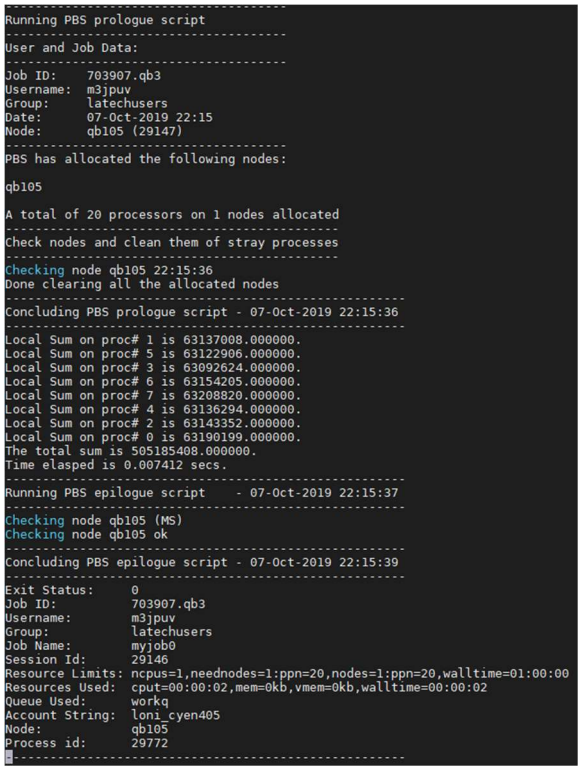
```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob3
#PBS -o myjob3.out
#PBS -e myjob3.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_3"
cd $work
mpirun -np 4 -machinefile $PBS_NODEFILE ./$CFILE
```

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob3
#PBS -o myjob3.out
#PBS -e myjob3.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_3"
cd $work
mpirun -np 8 -machinefile $PBS_NODEFILE ./$CFILE
```

	4 Threads	8 Threads
t	0.004615	0.007412

Question 4:





Submit Scripts:

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob4
#PBS -o myjob4.out
#PBS -e myjob4.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_4"
cd $work
mpirun -np 4 -machinefile $PBS_NODEFILE ./$CFILE
```

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob4
#PBS -o myjob4.out
#PBS -e myjob4.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_4"
cd $work
mpirun -np 8 -machinefile $PBS_NODEFILE ./$CFILE
```

	4 Threads	8 Threads
t	0.000085	0.007412

Question 5:

```
Running PBS prologue script
------------------------------------------
User and Job Data:
------------------------------------------
Job ID:    703942.qb3
Username:  m3jpuv
Group:     latechusers
Date:      07-Oct-2019 22:55
Node:      qb117 (24316)
------------------------------------------
PBS has allocated the following nodes:

qb117

A total of 20 processors on 1 nodes allocated
------------------------------------------------
Check nodes and clean them of stray processes
------------------------------------------------
Checking node qb117 22:55:38
Done clearing all the allocated nodes
------------------------------------------------------
Concluding PBS prologue script - 07-Oct-2019 22:55:38
------------------------------------------------------
Message at proc # 1: Hello, Drew from # 0!

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

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

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

Time elasped is 0.000144 seconds.
------------------------------------------------------
Running PBS epilogue script    - 07-Oct-2019 22:55:39
------------------------------------------------------
Checking node qb117 (MS)
Checking node qb117 ok
------------------------------------------------------
Concluding PBS epilogue script - 07-Oct-2019 22:55:40
------------------------------------------------------
Exit Status:     0
Job ID:          703942.qb3
Username:        m3jpuv
Group:           latechusers
Job Name:        myjob0
Session Id:      24315
Resource Limits: ncpus=1,neednodes=1:ppn=20,nodes=1:ppn=20,walltime=01:00:00
Resources Used:  cput=00:00:01,mem=0kb,vmem=0kb,walltime=00:00:03
Queue Used:      workq
Account String:  loni_cyen405
Node:            qb117
Process id:      24933
------------------------------------------------------
```

```
Running PBS prologue script
------------------------------------------
User and Job Data:
------------------------------------------
Job ID:    703940.qb3
Username:  m3jpuv
Group:     latechusers
Date:      07-Oct-2019 22:53
Node:      qb104 (78396)
------------------------------------------
PBS has allocated the following nodes:

qb104

A total of 20 processors on 1 nodes allocated
------------------------------------------------
Check nodes and clean them of stray processes
------------------------------------------------
Checking node qb104 22:53:15
Done clearing all the allocated nodes
------------------------------------------------------
Concluding PBS prologue script - 07-Oct-2019 22:53:15
------------------------------------------------------
Message at proc # 1: Hello, Drew from # 0!

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

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

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

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

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

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

Message at proc # 0: Hello, Drew from # 7!

Time elasped is 0.000221 seconds.
------------------------------------------------------
Running PBS epilogue script    - 07-Oct-2019 22:53:16
------------------------------------------------------
Checking node qb104 (MS)
Checking node qb104 ok
------------------------------------------------------
Concluding PBS epilogue script - 07-Oct-2019 22:53:18
------------------------------------------------------
Exit Status:     0
Job ID:          703940.qb3
Username:        m3jpuv
Group:           latechusers
Job Name:        myjob0
Session Id:      78395
Resource Limits: ncpus=1,neednodes=1:ppn=20,nodes=1:ppn=20,walltime=01:00:00
Resources Used:  cput=00:00:02,mem=0kb,vmem=0kb,walltime=00:00:03
Queue Used:      workq
Account String:  loni_cyen405
Node:            qb104
Process id:      79022
------------------------------------------------------
"myjob0.out" 62L, 1989C
```

Submit Scripts:

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob5
#PBS -o myjob5.out
#PBS -e myjob5.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_5"
cd $work
mpirun -np 4 -machinefile $PBS_NODEFILE ./$CFILE
```

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -N myjob5
#PBS -o myjob5.out
#PBS -e myjob5.err
#PBS -q workq
#PBS -A loni_cyen405
#PBS -m e
#PBS -M ama067@latech.edu
work="/home/m3jpuv/lab_2"
CFILE="lab2_5"
cd $work
mpirun -np 8 -machinefile $PBS_NODEFILE ./$CFILE
```

	4 Threads	8 Threads
t	0.000144	0.000221

Conclusion:

Given the results from each use of MPI, an assumption could be made on which iterations seem to be more efficient for the given number of threads. In these cases, there are comparisons between each thread printing their own message, printing a received broadcast or receiving, printing and sending messages. Each has its own advantages and disadvantages, but the solution for lab2_5 happened to be the best for speed due to the passing of messages. Given more time in the future, a further understanding and adaptation of such programming could be discovered, increasing speed and efficiency through parallelism.

Github Link:

https://github.com/M3JPUV/CSC452_Labs.git