Lab-Report

Submitted by: - Parth Parashar

I had a very pleasant experience working with this lab-assignment.

The file I/O operations are an integral part of every programming language and the way MPI handles I/O operations is quite efficient.

I was able to compile and run all the programs for the first question. The way MPI handles the input output operations is quite commendable.

MPI is very quick as well. The use of message passing is quite efficient as compared to other thread handling techniques.

MPI is also very compact as well and the in-built functions are quite quick as well.

I had to struggle a little with the second program though. I had a hard time getting the right functionality. I was able to get the input and division of P was also performed properly along with termination message.

Array was set properly as well.

For computing the total sum, I made use of the sum-mpi.c program.

For doubling the value of each value and writing it to the output file was convenient as well.

This was a very good introduction to the functionality of MPI.

Program-1: -

a) For program -1, I was able to run it properly using the provided code.

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 8 file-in in16
No protocol specified
rank=0: buf=[7653,1338,3008,6847]
rank=1: buf=[7653,1338,3008,6847]
rank=3: buf=[7653,1338,3008,6847]
rank=5: buf=[7653,1338,3008,6847]
rank=6: buf=[7653,1338,3008,6847]
rank=7: buf=[7653,1338,3008,6847]
rank=2: buf=[7653,1338,3008,6847]
rank=4: buf=[7653,1338,3008,6847]
```

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-in in16
No protocol specified
rank=0: buf=[7653,1338,3008,6847]
rank=1: buf=[7653,1338,3008,6847]
rank=2: buf=[7653,1338,3008,6847]
rank=3: buf=[7653,1338,3008,6847]
```

The modified program is given: -

OpenSSH SSH client

```
//-------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-------------------------------------------------------------------------

// A simple demo program of MPI file input.
//
// Usage:
//    linux> mpirun -n <#procs> file-in <filename>
//
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, buf[4];
  MPI_File fh;
  MPI_Status st;

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

  if (argc < 2) {
    if (rank == 0)
      printf("Usage: mpirun -n <#procs> file-in <filename>\n");
    MPI_Finalize();
    return 0;
  }

  // all processes open and read from the same file
  MPI_File_open(MPI_COMM_WORLD, argv[1], MPI_MODE_RDONLY, MPI_INFO_NULL, &fh);
  // read two integers
  MPI_File_read(fh, buf, 4, MPI_INT, &st);
  MPI_File_close(&fh);

  printf("rank=%d: buf=[%d,%d,%d,%d]\n", rank, buf[0], buf[1],buf[2],buf[3]);
  MPI_Finalize();
}
~
```

b) When the given program is compiled and run, there are 4 output files which are created. This number 4 came from the value of n provided with the mpirun statement.

OpenSSH SSH client

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpicc -o file-out file-out.c
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-out output
No protocol specified
parth2@proton:~/Desktop/parallelProgramming/lab8$ ls -l
total 99
-rw------- 1 parth2 them   742 May 17 02:33 datagen.c
-rwx------ 1 parth2 them 17144 May 17 03:06 file-in
-rw------- 1 parth2 them  1009 May 17 03:06 file-in.c
-rwx------ 1 parth2 them 17184 May 20 22:53 file-out
-rw------- 1 parth2 them  1085 May 17 02:33 file-out.c
-rw------- 1 parth2 them  1651 May 17 02:33 file-view.c
-rw------- 1 parth2 them    64 May 17 02:33 in16
-rw------- 1 parth2 them   128 May 17 02:33 in32
-rw------- 1 parth2 them   256 May 17 02:33 in64
-rw------- 1 parth2 them    32 May 17 02:33 in8
-rw------- 1 parth2 them 36716 May 17 02:33 lab8.pdf
-rw------- 1 parth2 them    16 May 20 22:54 output.0
-rw------- 1 parth2 them    16 May 20 22:54 output.1
-rw------- 1 parth2 them    16 May 20 22:54 output.2
-rw------- 1 parth2 them    16 May 20 22:54 output.3
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

The output files are highlighted in the screenshot provided above.

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i output.0
0000000           0           1           2           3
0000020
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i output.1
0000000         100         101         102         103
0000020
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i output.2
0000000         200         201         202         203
0000020
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

As can be seen from the screenshot and the code, each file contains 4 integers.

After making the necessary changes, we get the following output: -

OpenSSH SSH client

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ rm -r output.*
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-out output
No protocol specified
parth2@proton:~/Desktop/parallelProgramming/lab8$ ls -l
total 98
-rw------- 1 parth2 them   742 May 17 02:33 datagen.c
-rwx------ 1 parth2 them 17144 May 17 03:06 file-in
-rw------- 1 parth2 them  1009 May 17 03:06 file-in.c
-rwx------ 1 parth2 them 17184 May 20 23:02 file-out
-rw------- 1 parth2 them  1083 May 20 23:02 file-out.c
-rw------- 1 parth2 them  1651 May 17 02:33 file-view.c
-rw------- 1 parth2 them    64 May 17 02:33 in16
-rw------- 1 parth2 them   128 May 17 02:33 in32
-rw------- 1 parth2 them   256 May 17 02:33 in64
-rw------- 1 parth2 them    32 May 17 02:33 in8
-rw------- 1 parth2 them 36716 May 17 02:33 lab8.pdf
-rw------- 1 parth2 them    16 May 20 23:03 output.1
parth2@proton:~/Desktop/parallelProgramming/lab8$ vi file-in.c
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

The modified code is given below: -

OpenSSH SSH client

```c
//-------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-------------------------------------------------------------------------

// A simple demo program of MPI file output.
//
// Usage:
//   linux> mpirun -n <#procs> file-out <filename>
//
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, buf[4], cnt=4, i;
  char fname[20];
  MPI_File fh;
  MPI_Status st;

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

  if (argc < 2) {
    if (rank == 0)
      printf("Usage: mpirun -n <#procs> file-out <filename>\n");
    MPI_Finalize();
    return 0;
  }

  // prepare data
  for (i=0; i<cnt; i++)
    buf[i] = rank*100 + i;
  // each process writes to a separate output file
  sprintf(fname, "%s.%d", "output", 1);
  MPI_File_open(MPI_COMM_SELF, fname, MPI_MODE_CREATE|MPI_MODE_RDWR, MPI_INFO_NULL, &fh);
  // write four integers
  MPI_File_write(fh, buf, cnt, MPI_INT, &st);
  MPI_File_close(&fh);

  MPI_Finalize();
}
```

c) When we compile and run the program, we get the following output: -

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-view  in16 outputfi
No protocol specified
rank=0: buf=[7653,1338]
rank=1: buf=[3008,6847]
rank=2: buf=[5919,3580]
rank=3: buf=[6404,87]
parth2@proton:~/Desktop/parallelProgramming/lab8$ ls -l
total 109
-rw------- 1 parth2 them   742 May 17 02:33 datagen.c
-rwx------ 1 parth2 them 17144 May 17 03:06 file-in
-rw------- 1 parth2 them  1009 May 17 03:06 file-in.c
-rwx------ 1 parth2 them 17184 May 20 23:02 file-out
-rw------- 1 parth2 them  1083 May 20 23:02 file-out.c
-rwx------ 1 parth2 them 17224 May 20 23:06 file-view
-rw------- 1 parth2 them  1651 May 17 02:33 file-view.c
-rw------- 1 parth2 them    64 May 17 02:33 in16
-rw------- 1 parth2 them   128 May 17 02:33 in32
-rw------- 1 parth2 them   256 May 17 02:33 in64
-rw------- 1 parth2 them    32 May 17 02:33 in8
-rw------- 1 parth2 them 36716 May 17 02:33 lab8.pdf
-rw------- 1 parth2 them    16 May 20 23:03 output.1
-rw------- 1 parth2 them    64 May 20 23:08 outputfile
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i outputfile
0000000        7653        1338           0           0
0000020        3008        6847           1           1
0000040        5919        3580           2           2
0000060        6404          87           3           3
0000100
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

In the output, we can see that there are four integers according to their ranks along with the total number.

Now, the offset calculation when changed as given below will give the output as provided in the next two screens.

OpenSSH SSH client

```c
//------------------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//------------------------------------------------------------------------------

// A demo program of MPI concurrent I/O, with file view settings.
//
// Usage:
//   linux> mpirun -n <#procs> file-view <infile> <outfile>
//
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, offset, buf[4];
  MPI_File fin, fout;
  MPI_Status st;

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

  if (argc < 3) {
    if (rank == 0)
      printf("Usage: mpirun -n <#procs> file-view <infile> <outfile>\n");
    MPI_Finalize();
    return 0;
  }

  // all processes open the same pair of files
  MPI_File_open(MPI_COMM_WORLD, argv[1], MPI_MODE_RDONLY, MPI_INFO_NULL, &fin);
  MPI_File_open(MPI_COMM_WORLD, argv[2],
                MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fout);

  // each reads two integers, from a specific position
  offset = rank * 1 * sizeof(int);  // now skips rank*1 integers
  MPI_File_set_view(fin, offset, MPI_INT, MPI_INT, "native", MPI_INFO_NULL);
  MPI_File_read(fin, buf, 2, MPI_INT, &st);
  printf("rank=%d: buf=[%d,%d]\n", rank, buf[0], buf[1]);

  // each writes four integers, to a specific position
  buf[2] = buf[3] = rank;
  offset = rank * 4 * sizeof(int);  // skip rank*4 integers
  MPI_File_set_view(fout, offset, MPI_INT, MPI_INT, "native", MPI_INFO_NULL);
  MPI_File_write(fout, buf, 4, MPI_INT, &st);

  MPI_File_close(&fin);
  MPI_File_close(&fout);
  MPI_Finalize();
}
```

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ vi file-view.c
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpicc -o file-view file-view.c
parth2@proton:~/Desktop/parallelProgramming/lab8$ rm -r outputfile
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-view  in16 outputfile
No protocol specified
rank=0: buf=[7653,1338]
rank=1: buf=[1338,3008]
rank=2: buf=[3008,6847]
rank=3: buf=[6847,5919]
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i outputfile
0000000        7653        1338           0           0
0000020        1338        3008           1           1
0000040        3008        6847           2           2
0000060        6847        5919           3           3
0000100
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-view  in32 outputfile
No protocol specified
rank=0: buf=[2110,1113]
rank=1: buf=[1113,1161]
rank=2: buf=[1161,5175]
rank=3: buf=[5175,7947]
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i outputfile
0000000        2110        1113           0           0
0000020        1113        1161           1           1
0000040        1161        5175           2           2
0000060        5175        7947           3           3
0000100
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

Now, when offset is changed again, then we get the following results (screenshots for changed value and the result)

OpenSSH SSH client

```
//-----------------------------------------------------------------------
// Program code for CS 415/515 Parallel Programming, Portland State University.
//-----------------------------------------------------------------------

// A demo program of MPI concurrent I/O, with file view settings.
//
// Usage:
//   linux> mpirun -n <#procs> file-view <infile> <outfile>
//
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, offset, buf[4];
  MPI_File fin, fout;
  MPI_Status st;

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

  if (argc < 3) {
    if (rank == 0)
      printf("Usage: mpirun -n <#procs> file-view <infile> <outfile>\n");
    MPI_Finalize();
    return 0;
  }

  // all processes open the same pair of files
  MPI_File_open(MPI_COMM_WORLD, argv[1], MPI_MODE_RDONLY, MPI_INFO_NULL, &fin);
  MPI_File_open(MPI_COMM_WORLD, argv[2],
                MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fout);

  // each reads two integers, from a specific position
  offset = rank * 1 * sizeof(int);  // now skips rank*1 integers
  MPI_File_set_view(fin, 3, MPI_INT, MPI_INT, "native", MPI_INFO_NULL);
  MPI_File_read(fin, buf, 2, MPI_INT, &st);
  printf("rank=%d: buf=[%d,%d]\n", rank, buf[0], buf[1]);

  // each writes four integers, to a specific position
  buf[2] = buf[3] = rank;
  offset = rank * 4 * sizeof(int);  // skip rank*4 integers
  MPI_File_set_view(fout, offset, MPI_INT, MPI_INT, "native", MPI_INFO_NULL);
  MPI_File_write(fout, buf, 4, MPI_INT, &st);

  MPI_File_close(&fin);
  MPI_File_close(&fout);
  MPI_Finalize();
}
```

OpenSSH SSH client

```
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpicc -o file-view file-view.c
parth2@proton:~/Desktop/parallelProgramming/lab8$ mpirun -n 4 file-view  in32 outputfile
No protocol specified
rank=3: buf=[284928,297216]
rank=0: buf=[284928,297216]
rank=1: buf=[284928,297216]
rank=2: buf=[284928,297216]
parth2@proton:~/Desktop/parallelProgramming/lab8$ od -i outputfile
0000000      284928      297216           0           0
0000020      284928      297216           1           1
0000040      284928      297216           2           2
0000060      284928      297216           3           3
0000100
parth2@proton:~/Desktop/parallelProgramming/lab8$
```

Program-2: - Program-2 is attached in this zip file.