ภาคผนวก

ตัวอย่างโปรแกรม BzipC.c ที่ใช้ในงานวิจัย

วิธีการบีบอัดข้อมูลทุกวิธีในงานวิจัยนี้จะถูกเขียนอยู่ในโปรแกรมเดียวนี้ โดยวิธีการบีบอัดข้อมูลจะเรียกใช้แตกต่างกันขึ้นอยู่กับคำสั่งที่ใช้ในการทดลอง ดังกล่าวไว้แล้วในบทที่ 4

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <bzlib.h>
main(int argc, char *argv[])
{
        int start, end;
        int count;
        int length;
        int error;
        char* buffer;
        char* outbuf;
        int nprocs;
        int myrank;
        int outsize,insize,sizeview;
        double starttime, finishtime, starttime1, finishtime1;
        char *filename;
        char *filenamebuf;
        int i, size;
        double weight, Pweight;
        int algorithm;
        MPI_Status    status;
        MPI_File      fh;
        MPI_Offset    filesize;
        MPI_Init(&argc, &argv);
```

```
starttime = MPI_Wtime();
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
algorithm = atoi(argv[2]);
if (algorithm == 1) // MSBzip2
{
        if (myrank == 0)  //Master
        {
                filename = argv[1];
                filenamebuf = (char *)malloc((strlen(filename)+7)*sizeof(char));
                sprintf(filenamebuf,"%s.bz2", filename);
                MPI_File_open(MPI_COMM_SELF, filename,
MPI_MODE_RDONLY | MPI_MODE_UNIQUE_OPEN, MPI_INFO_NULL, &fh);
                MPI_File_get_size(fh, &filesize);
                filesize = filesize/sizeof(char);
                length = filesize / nprocs + 1;
                for (i=1; i<=nprocs-1; i++ )
                {
                        buffer = (char *)malloc(length * sizeof(char));
                        MPI_File_set_view(fh, i*length * sizeof(char),MPI_CHAR,
MPI_CHAR, "native", MPI_INFO_NULL);
                        MPI_File_read(fh, buffer, length, MPI_CHAR, &status);
                        MPI_Get_count(&status, MPI_CHAR, &count);
                        MPI_Send(&count, 1, MPI_INT, i, 1,
MPI_COMM_WORLD );
                        MPI_Send(buffer, count, MPI_CHAR, i, 1,
MPI_COMM_WORLD );
                }
                buffer = (char *)malloc(length * sizeof(char));
                MPI_File_set_view(fh, 0,MPI_CHAR, MPI_CHAR, "native",
MPI_INFO_NULL);
                MPI_File_read(fh, buffer, length, MPI_CHAR, &status);
```

```
                MPI_Get_count(&status, MPI_CHAR, &count);
                MPI_File_close(&fh);
        }
        else  // Slave
        {
                MPI_Recv(&count, 1, MPI_INT, 0, 1, MPI_COMM_WORLD,
&status);
                buffer = (char *)malloc(count* sizeof(char));
                MPI_Recv(buffer, count, MPI_CHAR, 0, 1, MPI_COMM_WORLD,
&status);
        }
        outbuf = (char *)malloc((count + 600 +(count/100)) * sizeof(char));
        BZ2_bzBuffToBuffCompress (outbuf, &outsize, buffer, count, 9, 0, 30 );
        if (myrank == 0)  //Master
        {
                MPI_File_open(MPI_COMM_SELF, filenamebuf,
MPI_MODE_WRONLY | MPI_MODE_CREATE, MPI_INFO_NULL, &fh);
                MPI_File_set_view(fh, 0,MPI_CHAR, MPI_CHAR, "native",
MPI_INFO_NULL);
                MPI_File_write(fh, outbuf, outsize, MPI_CHAR,
MPI_STATUS_IGNORE);
                sizeview=outsize;
                for (i=1; i<=nprocs-1; i++ )
                {
                        MPI_File_set_view(fh, sizeview,MPI_CHAR, MPI_CHAR,
"native", MPI_INFO_NULL);
                        MPI_Recv(&outsize, 1, MPI_INT, i, 1,
MPI_COMM_WORLD, &status);
                        sizeview=sizeview+outsize;
                        outbuf = (char *)malloc(outsize* sizeof(char));
                        MPI_Recv(outbuf, outsize, MPI_CHAR, i, 1,
MPI_COMM_WORLD, &status);
```

```
                                MPI_File_write(fh, outbuf, outsize, MPI_CHAR,
MPI_STATUS_IGNORE);
                        }
                        MPI_File_close(&fh);
                }
                else  // Slave
                {
                        MPI_Send(&outsize, 1, MPI_INT, 0, 1, MPI_COMM_WORLD );
                        MPI_Send(outbuf, outsize, MPI_CHAR, 0, 1, MPI_COMM_WORLD );
                }
                finishtime = MPI_Wtime();
                printf ("Process %d Total Time is : %5.8f \n",myrank ,finishtime-starttime);
        }
        if (algorithm == 2) //AWBzip2
        {
                filename = argv[1];
                filenamebuf = (char *)malloc((strlen(filename)+7)*sizeof(char));
                sprintf(filenamebuf,"%s.bz2", filename);
                MPI_File_open(MPI_COMM_SELF, filename, MPI_MODE_RDONLY |
MPI_MODE_UNIQUE_OPEN, MPI_INFO_NULL, &fh);
                MPI_File_get_size(fh, &filesize);
                filesize = filesize/sizeof(char);
                length = filesize / nprocs + 1;
                buffer = (char *)malloc(length * sizeof(char));
                MPI_File_set_view(fh, myrank*length * sizeof(char),MPI_CHAR, MPI_CHAR,
"native", MPI_INFO_NULL);
                MPI_File_read(fh, buffer, length, MPI_CHAR, &status);
                MPI_Get_count(&status, MPI_CHAR, &count);
                MPI_File_close(&fh);
                outbuf = (char *)malloc((count + 600 +(count/100)) * sizeof(char));
                BZ2_bzBuffToBuffCompress (outbuf, &outsize, buffer, count, 9, 0, 30 );
                if (myrank==0)
```

```
                {
                        start=0;
                }
                if (myrank>0)
                {
                        MPI_Recv(&start, 1, MPI_INT, myrank-1, 1, MPI_COMM_WORLD,
&status);
                }
                if (myrank<nprocs-1)
                {
                        size=start+outsize;
                        MPI_Send(&size, 1, MPI_INT, myrank+1, 1, MPI_COMM_WORLD );
                }
                MPI_File_open(MPI_COMM_SELF, filenamebuf, MPI_MODE_WRONLY |
MPI_MODE_CREATE, MPI_INFO_NULL, &fh);
                MPI_File_set_view(fh, start, MPI_CHAR, MPI_CHAR, "native",
MPI_INFO_NULL);
                MPI_File_write(fh, outbuf, outsize, MPI_CHAR, MPI_STATUS_IGNORE);

                MPI_File_close(&fh);
                finishtime = MPI_Wtime();
                printf ("Process %d Total Time is : %5.8f \n",myrank ,finishtime-starttime);
        }
        if (algorithm == 3) //AWWBzip2
        {
                filename = argv[1];
                filenamebuf = (char *)malloc((strlen(filename)+7)*sizeof(char));
                sprintf(filenamebuf,"%s.bz2", filename);
                weight = atoi(argv[3]);
                MPI_File_open(MPI_COMM_SELF, filename, MPI_MODE_RDONLY,
MPI_INFO_NULL, &fh);
                MPI_File_get_size(fh, &filesize);
```

```
filesize = filesize/sizeof(char);
Pweight = filesize * weight / 1000;
length = (filesize - nprocs * (nprocs - 1) * Pweight / 2 ) / nprocs + 1;
start=0;
for (i=0; i<myrank; i++ )
{
        length = length + i * Pweight;
        start = start+length;
}
length = length + myrank * Pweight;
buffer = (char *)malloc(length * sizeof(char));
MPI_File_set_view(fh, start, MPI_CHAR, MPI_CHAR, "native",
MPI_INFO_NULL);
MPI_File_read(fh, buffer, length, MPI_CHAR, &status);
MPI_Get_count(&status, MPI_CHAR, &count);
MPI_File_close(&fh);
outbuf = (char *)malloc((count + 600 +(count/100)) * sizeof(char));
BZ2_bzBuffToBuffCompress (outbuf, &outsize, buffer, count, 9, 0, 30 );
if (myrank==0)
{
        start=0;
}
if (myrank>0)
{
        MPI_Recv(&start, 1, MPI_INT, myrank-1, 1, MPI_COMM_WORLD,
&status);
}
if (myrank<nprocs-1)
{
        size=start+outsize;
        MPI_Send(&size, 1, MPI_INT, myrank+1, 1, MPI_COMM_WORLD );
}
```

```
                MPI_File_open(MPI_COMM_SELF, filenamebuf, MPI_MODE_WRONLY |
MPI_MODE_CREATE, MPI_INFO_NULL, &fh);
                MPI_File_set_view(fh, start, MPI_CHAR, MPI_CHAR, "native",
MPI_INFO_NULL);
                MPI_File_write(fh, outbuf, outsize, MPI_CHAR, MPI_STATUS_IGNORE);
                MPI_File_close(&fh);
                finishtime = MPI_Wtime();
                printf ("Process %d Total Time is : %5.8f \n",myrank ,finishtime-starttime);
        }
        MPI_Finalize();
}
```