CSCI596 Assignment 2—Parallel Computation of p

Youzhi Qu

CSCI-596 assignment2

Prof. Aiichiro Nakano

**Part I: Programming**

1. **global\_pi.c:**

#include "mpi.h"

#include <stdio.h>

#define NBIN 10000000

int nprocs; /\* Number of processors \*/

int myid; /\* My rank \*/

double global\_sum(double partial) {

/\* Implement your own global summation here \*/

double mydone, hisdone;

int bitvalue, partner;

MPI\_Status status;

mydone = partial;

for (bitvalue = 1; bitvalue < nprocs; bitvalue \*= 2) {

partner = myid ^ bitvalue;

MPI\_Send(&mydone, 1, MPI\_DOUBLE, partner, bitvalue, MPI\_COMM\_WORLD);

MPI\_Recv(&hisdone, 1, MPI\_DOUBLE, partner, bitvalue, MPI\_COMM\_WORLD, &status);

mydone += hisdone;

}

return mydone;

}

int main(int argc, char \*argv[]) {

double partial, sum= 0.0, avg, step, x, pi, cpu1, cpu2;

int i;

step = 1.0 / NBIN;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &nprocs);

cpu1 = MPI\_Wtime();

for (i = myid; i < NBIN; i += nprocs)

{

x = (i + 0.5)\*step;

sum += 4.0 / (1.0 + x \* x);

}

partial = sum \* step;

pi = global\_sum(partial);

cpu2 = MPI\_Wtime();

if (myid == 0) {

printf("Value of pi = %le\n", pi);

printf("Execution time (s) = %le\n", cpu2-cpu1);

}

MPI\_Finalize();

return 0;

}

1. **global\_pi\_iso.c:**

#include "mpi.h"

#include <stdio.h>

#define NPERP 10000000

int nprocs; /\* Number of processors \*/

int myid; /\* My rank \*/

double global\_sum(double partial) {

/\* Implement your own global summation here \*/

double mydone, hisdone;

int bitvalue, partner;

MPI\_Status status;

mydone = partial;

for (bitvalue = 1; bitvalue < nprocs; bitvalue \*= 2) {

partner = myid ^ bitvalue;

MPI\_Send(&mydone, 1, MPI\_DOUBLE, partner, bitvalue, MPI\_COMM\_WORLD);

MPI\_Recv(&hisdone, 1, MPI\_DOUBLE, partner, bitvalue, MPI\_COMM\_WORLD, &status);

mydone += hisdone;

}

return mydone;

}

int main(int argc, char \*argv[]) {

double partial, sum= 0.0, avg, step, x, pi, cpu1, cpu2;

int i;

int NBIN;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &nprocs);

NBIN = NPERP \* nprocs;

step = 1.0 / NBIN;

cpu1 = MPI\_Wtime();

for (i = myid; i < NBIN; i += nprocs)

{

x = (i + 0.5)\*step;

sum += 4.0 / (1.0 + x \* x);

}

partial = sum \* step;

pi = global\_sum(partial);

cpu2 = MPI\_Wtime();

if (myid == 0) {

printf("Value of pi = %le\n", pi);

printf("Execution time (s) = %le\n", cpu2-cpu1);

}

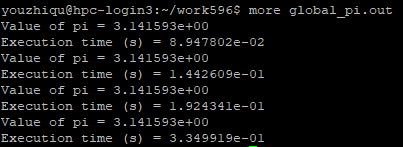
MPI\_Finalize();

return 0;

}

**Part II: Scalability**

**global\_pi.out:**



**global\_pi\_iso.out:**

