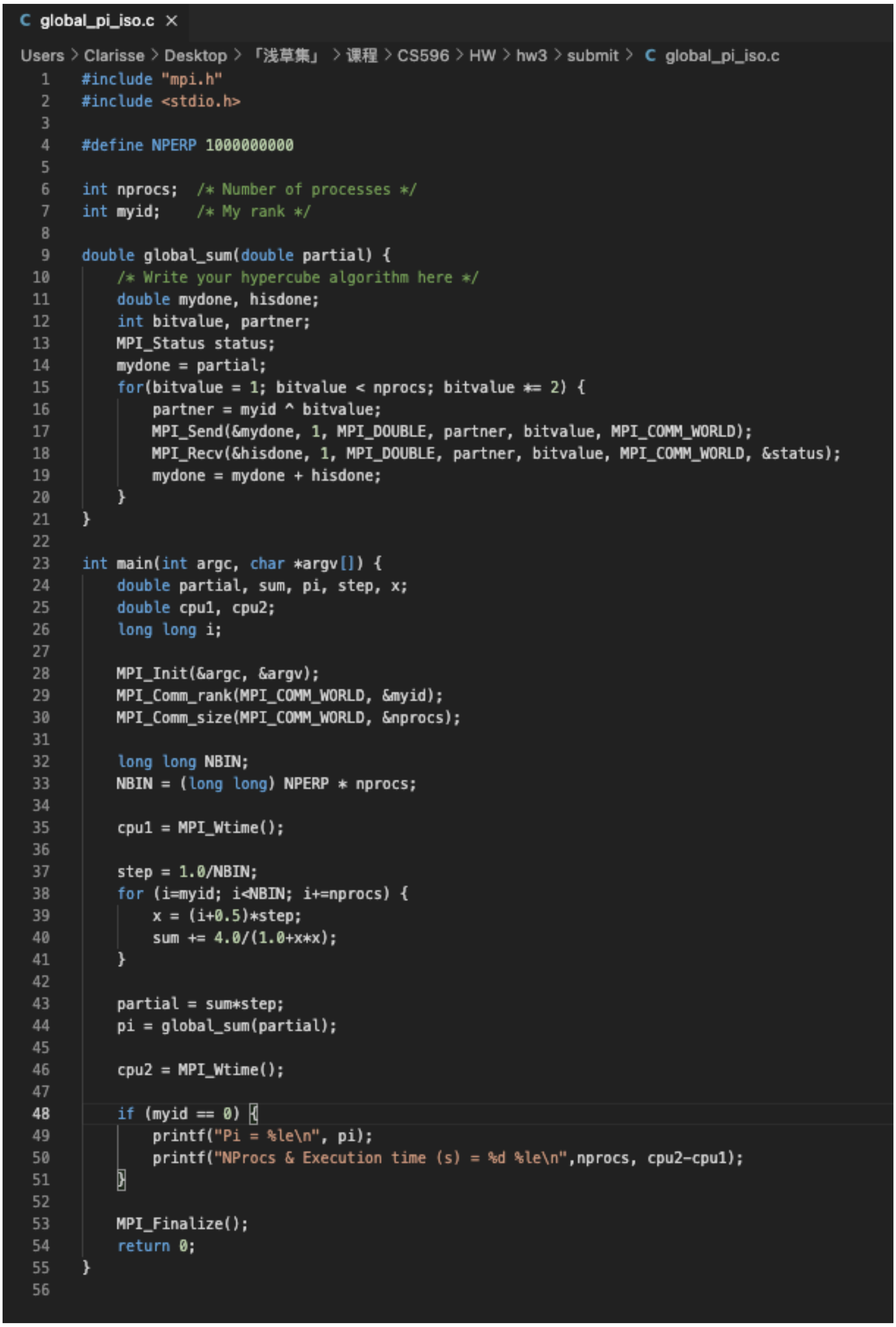
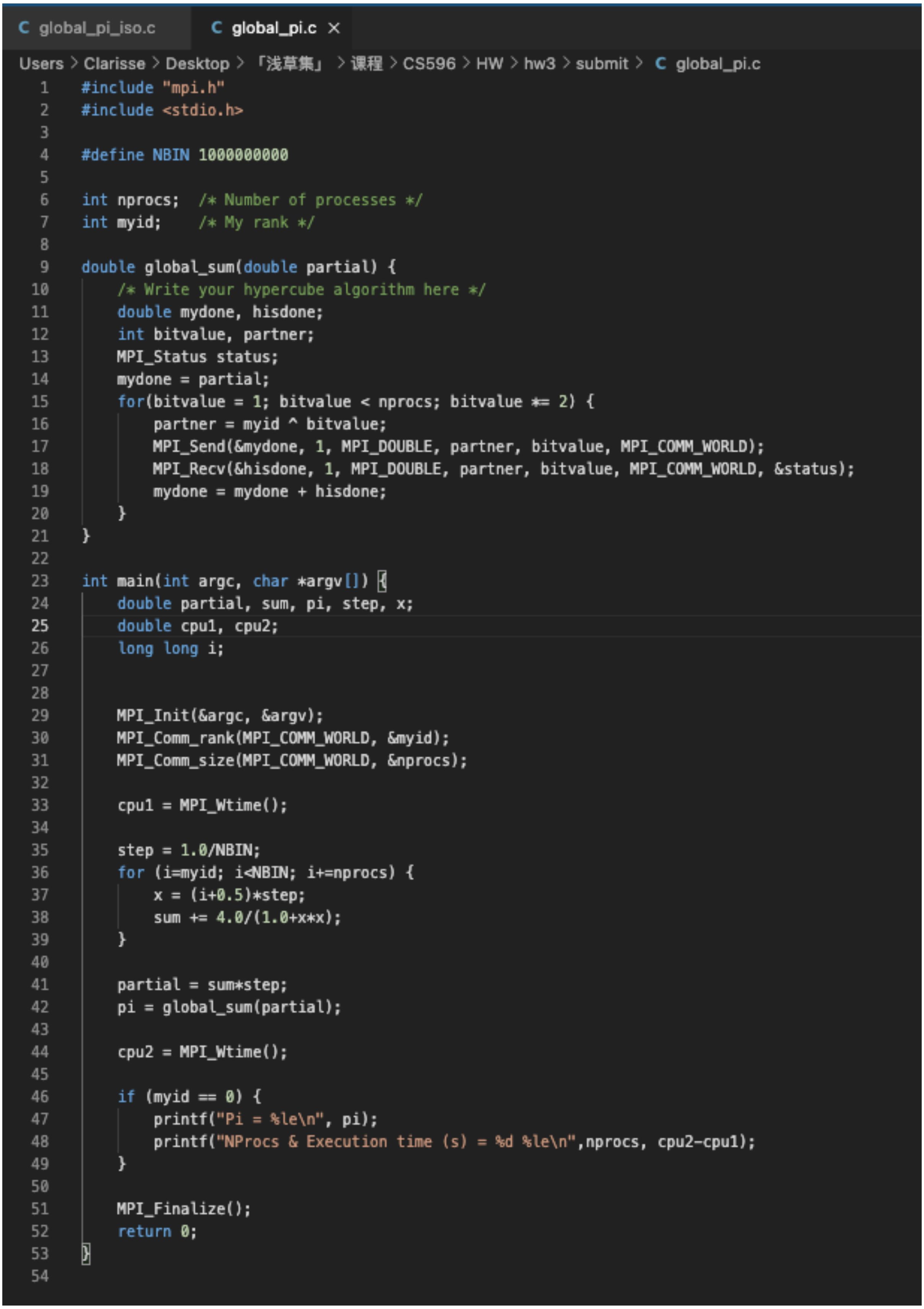
CSCI 596_HW3
Student Name: Cancan Hua
USC ID: 4612363893

global_pi_iso.c


C global_pi_iso.c ×
Users > Clarisse > Desktop > 「浅草集」 > 课程 > CS596 > HW > hw3 > submit > C global_pi_iso.c

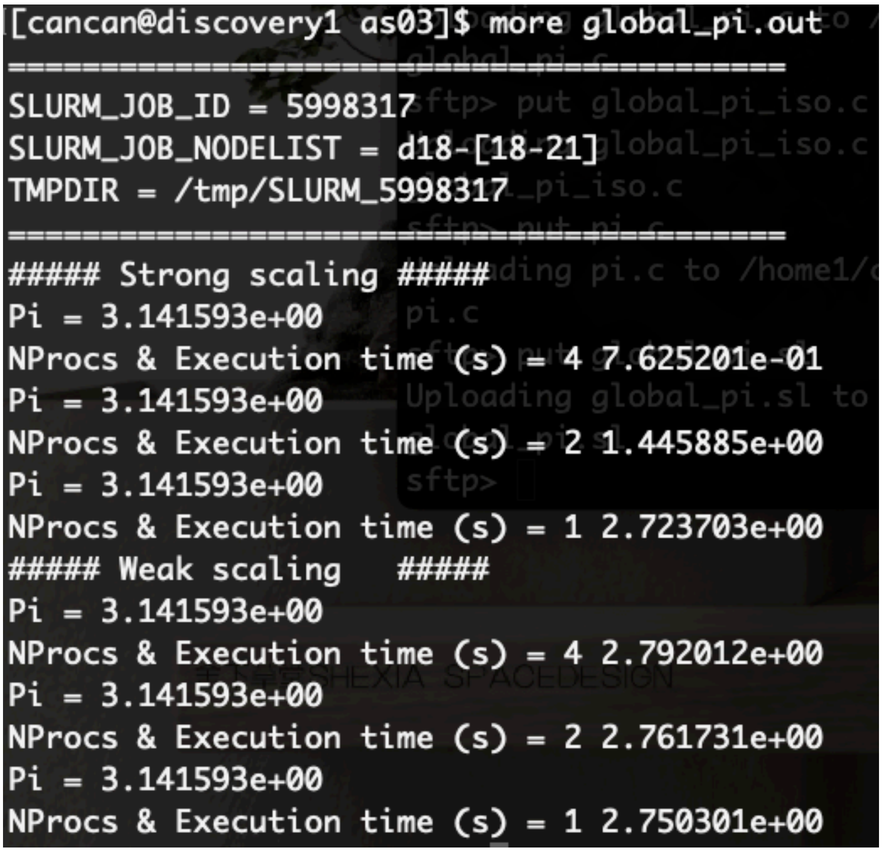
```c
#include "mpi.h"
#include <stdio.h>

#define NPERP 1000000000

int nprocs;  /* Number of processes */
int myid;    /* My rank */

double global_sum(double partial) {
    /* Write your hypercube algorithm 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 = mydone + hisdone;
    }
}

int main(int argc, char *argv[]) {
    double partial, sum, pi, step, x;
    double cpu1, cpu2;
    long long i;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    long long NBIN;
    NBIN = (long long) NPERP * nprocs;

    cpu1 = MPI_Wtime();

    step = 1.0/NBIN;
    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("Pi = %le\n", pi);
        printf("NProcs & Execution time (s) = %d %le\n",nprocs, cpu2-cpu1);
    }

    MPI_Finalize();
    return 0;
}
```

global_pi.c


C global_pi_iso.c
C global_pi.c ×
Users > Clarisse > Desktop > 「浅草集」 > 课程 > CS596 > HW > hw3 > submit > C global_pi.c

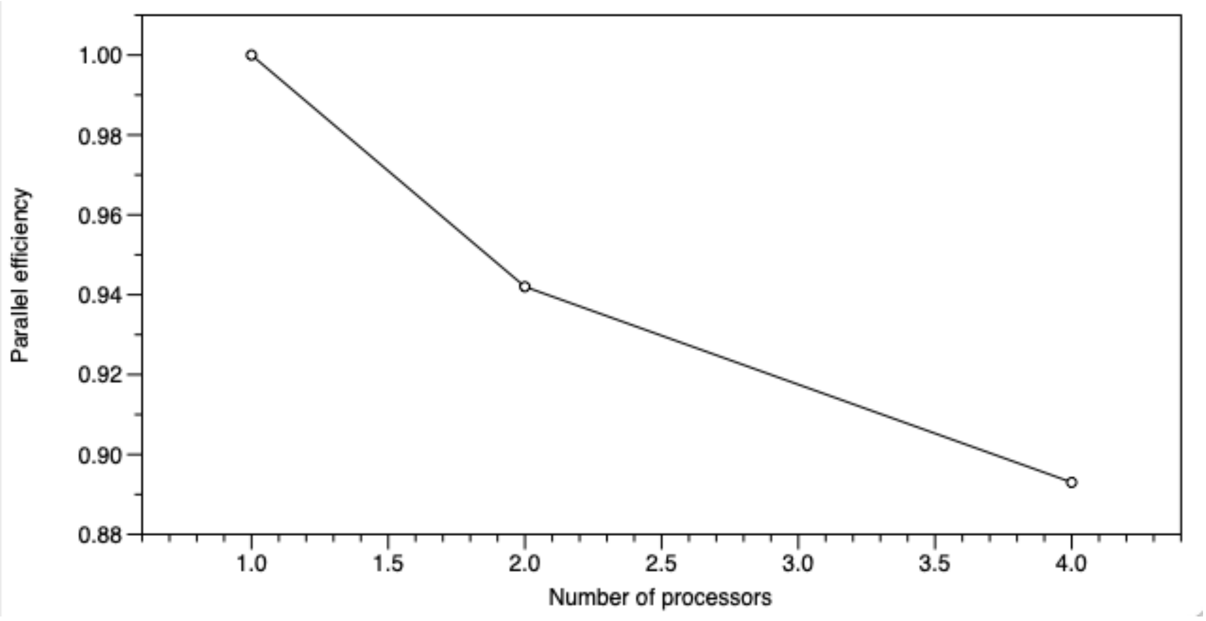
```c
#include "mpi.h"
#include <stdio.h>

#define NBIN 1000000000

int nprocs;  /* Number of processes */
int myid;    /* My rank */

double global_sum(double partial) {
    /* Write your hypercube algorithm 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 = mydone + hisdone;
    }
}

int main(int argc, char *argv[]) {
    double partial, sum, pi, step, x;
    double cpu1, cpu2;
    long long i;


    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    cpu1 = MPI_Wtime();

    step = 1.0/NBIN;
    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("Pi = %le\n", pi);
        printf("NProcs & Execution time (s) = %d %le\n",nprocs, cpu2-cpu1);
    }

    MPI_Finalize();
    return 0;
}

```

Result:

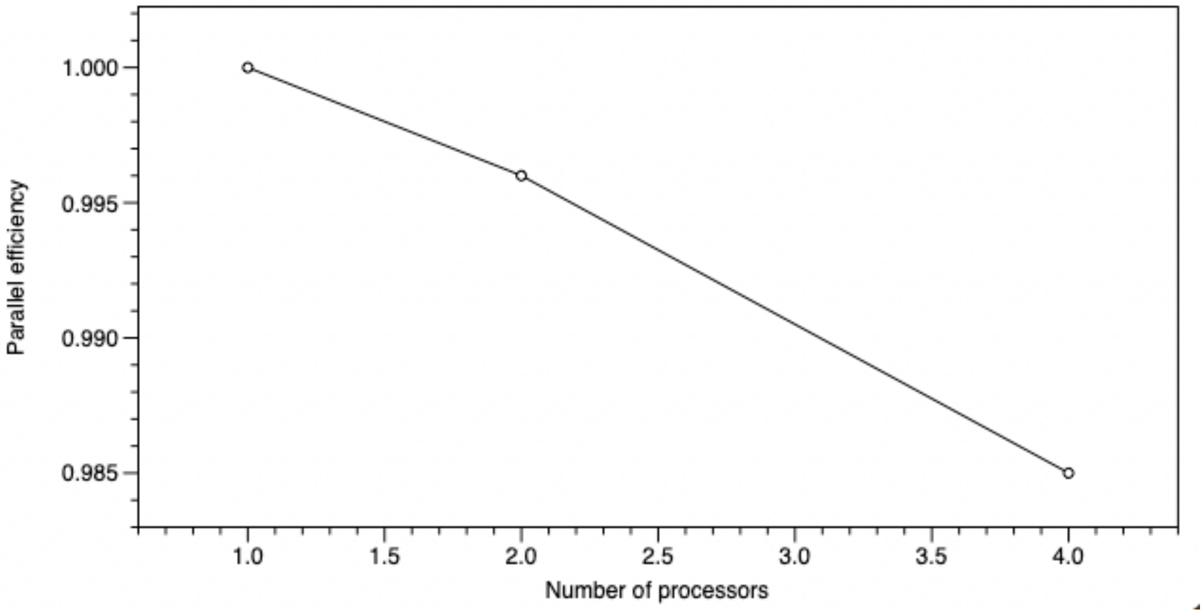


```
[cancan@discovery1 as03]$ more global_pi.out
==========================================
SLURM_JOB_ID = 5998317
SLURM_JOB_NODELIST = d18-[18-21]
TMPDIR = /tmp/SLURM_5998317
==========================================
##### Strong scaling #####
Pi = 3.141593e+00
NProcs & Execution time (s) = 4 7.625201e-01
Pi = 3.141593e+00
NProcs & Execution time (s) = 2 1.445885e+00
Pi = 3.141593e+00
NProcs & Execution time (s) = 1 2.723703e+00
##### Weak scaling   #####
Pi = 3.141593e+00
NProcs & Execution time (s) = 4 2.792012e+00
Pi = 3.141593e+00
NProcs & Execution time (s) = 2 2.761731e+00
Pi = 3.141593e+00
NProcs & Execution time (s) = 1 2.750301e+00
```

Fixed Problem Size Parallel Efficiency Plot:



Isogranular Parallel Efficiency Plot:


Parallel efficiency
Number of processors