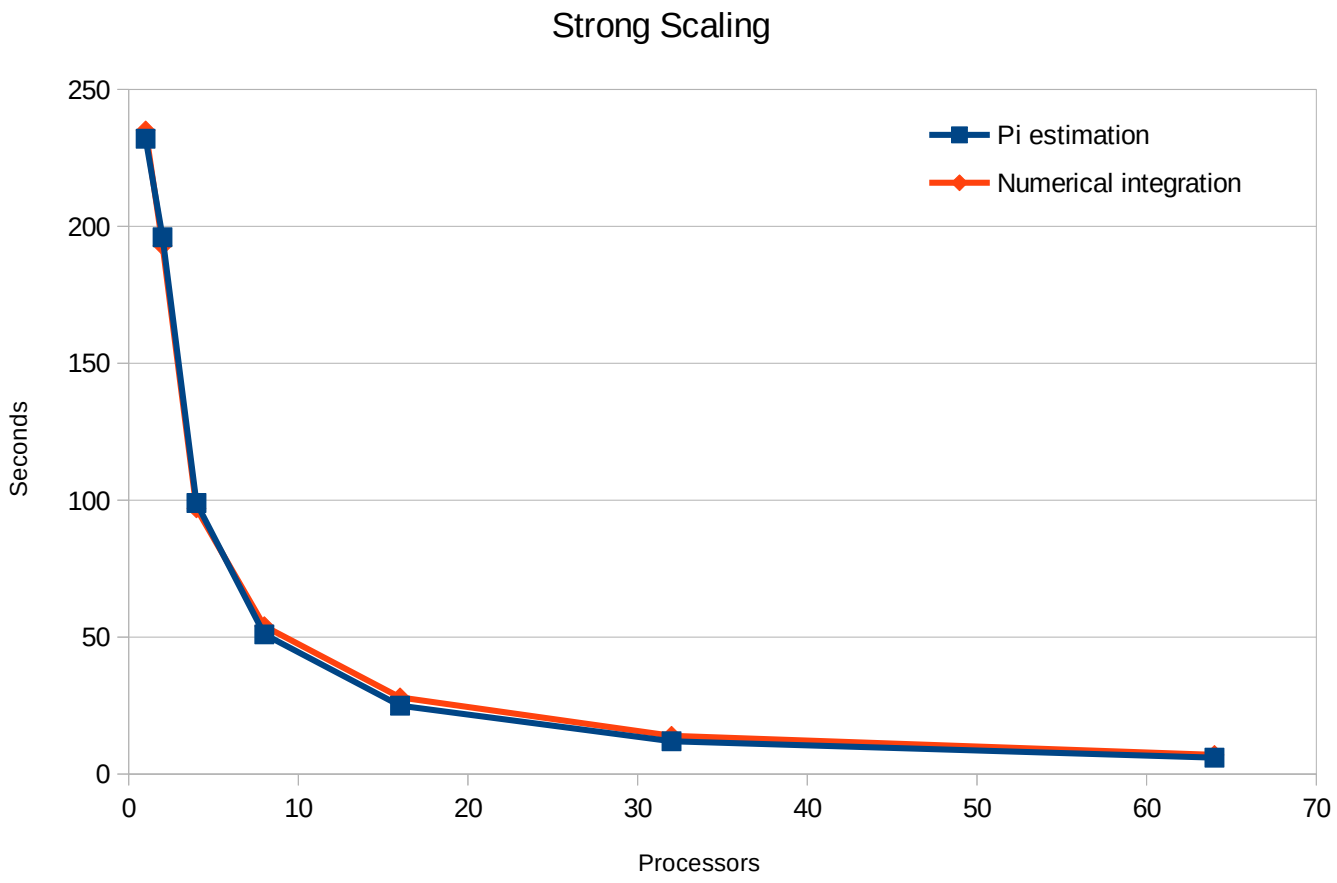
Assignment 4
Intro to HPC

Wesley Liao



Program Output:

```
Working directory is /gpfs/home/w/e/wesliao/BigRed2/assignment4
CALCULATION OF PI STRONG SCALING TEST
computing pi with 1 processors
the total is 3.141592653588989
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 232.000000 s
Application 18950005 resources: utime ~231s, stime ~0s, Rss ~4952, inblocks ~6267, outblocks ~15618
computing pi with 2 processors
the total is 3.141592653589852
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 196.000000 s
Application 18950012 resources: utime ~392s, stime ~0s, Rss ~4960, inblocks ~6345, outblocks ~15618
computing pi with 4 processors
the total is 3.141592653589775
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 99.000000 s
Application 18950015 resources: utime ~393s, stime ~1s, Rss ~4952, inblocks ~6502, outblocks ~15618
computing pi with 8 processors
the total is 3.141592653589803
```

```
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 51.000000 s
Application 18950016 resources: utime ~405s, stime ~1s, Rss ~4960, inblocks ~6814, outblocks ~15618
computing pi with 16 processors
the total is 3.141592653589797
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 25.000000 s
Application 18950017 resources: utime ~405s, stime ~1s, Rss ~4952, inblocks ~7439, outblocks ~15619
computing pi with 32 processors
the total is 3.141592653589790
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 12.000000 s
Application 18950018 resources: utime ~407s, stime ~3s, Rss ~4960, inblocks ~8688, outblocks ~15621
computing pi with 64 processors
the total is 3.141592653589792
true value:  3.141592653589793
req. accuracy:           ^
execution time (wall clock): 6.000000 s
Application 18950019 resources: utime ~423s, stime ~13s, Rss ~9800, inblocks ~18192, outblocks ~31241
NUMERICAL INTEGRATION STRONG SCALING TEST
computing integral with 1 processors
the total is 0.666666666662443
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 235.000000 s
Application 18950020 resources: utime ~236s, stime ~0s, Rss ~4952, inblocks ~6954, outblocks ~17392
computing integral with 2 processors
the total is 0.666666666663556
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 193.000000 s
Application 18950023 resources: utime ~386s, stime ~0s, Rss ~4960, inblocks ~7032, outblocks ~17392
computing integral with 4 processors
the total is 0.666666666666656
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 97.000000 s
Application 18950028 resources: utime ~389s, stime ~1s, Rss ~4952, inblocks ~7189, outblocks ~17392
computing integral with 8 processors
the total is 0.666666666666648
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 54.000000 s
Application 18950029 resources: utime ~433s, stime ~1s, Rss ~4960, inblocks ~7501, outblocks ~17392
computing integral with 16 processors
the total is 0.666666666666515
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 28.000000 s
Application 18950030 resources: utime ~442s, stime ~1s, Rss ~4952, inblocks ~8126, outblocks ~17393
computing integral with 32 processors
the total is 0.666666666666656
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 14.000000 s
Application 18950031 resources: utime ~459s, stime ~4s, Rss ~4960, inblocks ~9375, outblocks ~17395
computing integral with 64 processors
the total is 0.666666666666663
true value   0.666666666666667
req. accuracy:       ^
execution time (wall): 7.000000 s
Application 18950032 resources: utime ~474s, stime ~11s, Rss ~8912, inblocks ~19566, outblocks ~34789
```

Program listings:

p1.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int myid, numprocs;
  MPI_Request request;
  MPI_Status status;

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

  time_t start, end;
  if (myid==0)
    time(&start);

  MPI_Barrier(MPI_COMM_WORLD);

  long divisions = strtol(argv[1], NULL, 10);
  const double radius = 2.0;
  double div_width = radius/divisions;

  double mysum = 0.0;

  for(long division = myid; division < divisions; division += numprocs) {
    double l = (division+0.5)*div_width;
    double h = sqrt((radius*radius)-(l*l));

    mysum += h * div_width;
  }

  double total = 0.0;
  MPI_Reduce( &mysum, &total, 1,
              MPI_DOUBLE, MPI_SUM,
              0, MPI_COMM_WORLD);

  if(myid == 0){
    time(&end);
    double timediff = difftime(end, start);
    printf("the total is %1.15f\n", total);
    printf("true value:  %1.15f\n", M_PI);
    printf("req. accuracy:           ^\n");
    printf("execution time (wall clock): %f s\n", timediff);
  }

  MPI_Finalize();
  return 0;
}
```

p2.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
  int myid, numprocs;
  MPI_Request request;
  MPI_Status status;

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

  time_t start, end;
  if(myid==0)
    time(&start);
  MPI_Barrier(MPI_COMM_WORLD);

  long divisions = strtol(argv[1], NULL, 10);
  double interval = M_PI/2.0;
  double div_width = interval/divisions;

  double mysum = 0.0;

  for(long division = myid; division < divisions; division += numprocs) {
    double x = division*div_width;
    double h = cos(x)*sin(2*x);

    mysum += h * div_width;
  }

  double total = 0.0;
  MPI_Reduce( &mysum, &total, 1,
              MPI_DOUBLE, MPI_SUM,
              0, MPI_COMM_WORLD);

  if(myid == 0){
    time(&end);
    double timediff = difftime(end, start);
    printf("the total is %1.15f\n", total);
    printf("true value   %1.15f\n", 2.0/3.0);
    printf("req. accuracy:       ^\n");
    printf("execution time (wall): %f s\n", timediff);
  }

  MPI_Finalize();
  return 0;
}
```