CSCI 455: Lab #3 — Pi Calculation

Darwin Jacob Groskleg

Winter 2020

Contents

Pi Calculation

Q1. Any speedup or unusual changes?

```
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$ make sample_parpi_extended
Platform: Linux (96 cpu cores recognized)
MPIRUN parpi with 1 node processes:
parallel program results with 1 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589552)
difference between computed pi and math.h M_PI = 0.000000000000241 (2.41e-13)
time to compute = 4.57988e-09 seconds
MPIRUN parpi with 2 node processes:
parallel program results with 2 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589526)
difference between computed pi and math.h M_PI = 0.000000000000267 (2.67e-13)
time to compute = 2.34311e-09 seconds
MPIRUN parpi with 3 node processes:
parallel program results with 3 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589825)
difference between computed pi and math.h M_PI = 0.000000000000032 (3.20e-14)
time to compute = 1.53144e-09 seconds
MPIRUN parpi with 4 node processes:
parallel program results with 4 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589927)
difference between computed pi and math.h M_PI = 0.000000000000134 (1.34e-13)
time to compute = 1.15312e-09 seconds
MPIRUN parpi with 5 node processes:
parallel program results with 5 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589925)
difference between computed pi and math.h M_PI = 0.000000000000131 (1.31e-13)
time to compute = 9.58631e-10 seconds
MPIRUN parpi with 10 node processes:
parallel program results with 10 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589796)
difference between computed pi and math.h M_PI = 0.000000000000003 (2.66e-15)
time to compute = 4.90285e-10 seconds
MPIRUN parpi with 20 node processes:
parallel program results with 20 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589794)
difference between computed pi and math.h M_PI = 0.000000000000001 (1.33e-15)
time to compute = 2.48424e-10 seconds
MPIRUN parpi with 40 node processes:
parallel program results with 40 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589766)
difference between computed pi and math.h M_PI = 0.000000000000027 (2.66e-14)
time to compute = 1.25258e-10 seconds
MPIRUN parpi with 60 node processes:
parallel program results with 60 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589769)
difference between computed pi and math.h M_PI = 0.000000000000024 (2.40e-14)
time to compute = 8.39765e-11 seconds
MPIRUN parpi with 80 node processes:
parallel program results with 80 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589798)
difference between computed pi and math.h M_PI = 0.000000000000004 (4.44e-15)
time to compute = 6.28676e-11 seconds
MPIRUN parpi with 96 node processes:
parallel program results with 96 processes and 400000000 steps:
computed pi = 3.14159  (3.141592653589796)
difference between computed pi and math.h M_PI = 0.000000000000003 (2.66e-15)
time to compute = 7.77471e-11 seconds
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$
```

Figure 1: Console screenshot of multiple executions of `parpi` under MPI with 1, 2, 3, 4, 5, 10, 20, 40, 60, 80 and 96 compute nodes respectively.

Q2. Observations after doubling `NUM_STEPS`

```
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$ make run1
mpicc -std=c99 -Wall -Wextra -g -D_GLIBCXX_DEBUG -O0    parpi.c  -lm -o parpi
Platform: Linux (96 cpu cores recognized)
mpirun --use-hwthread-cpus -np 96 ./parpi
parallel program results with 96 processes and 400000000 steps:
computed pi = 3.14159  (3.1415926535897796)
difference between computed pi and math.h M_PI = 0.000000000000003 (2.66e-15)
time to compute = 5.31957e-11 seconds
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$ make run2
Platform: Linux (96 cpu cores recognized)
mpirun --use-hwthread-cpus -np 96 ./parpi 2
parallel program results with 96 processes and 800000000 steps:
computed pi = 3.14159  (3.1415926535897811)
difference between computed pi and math.h M_PI = 0.000000000000018 (1.78e-14)
time to compute = 1.06613e-10 seconds
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$
```

Figure 2: Console screenshot of *parpi* running over 96 compute nodes regularly then again with a doubling of `NUM_STEPS`.

```
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$ make sample_parpi2_extended
Platform: Linux (96 cpu cores recognized)
MPIRUN parpi with 1 node processes:
parallel program results with 1 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653588957)
difference between computed pi and math.h M_PI = 0.000000000000836 (8.36e-13)
time to compute = 9.16142e-09 seconds
MPIRUN parpi with 2 node processes:
parallel program results with 2 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653590092)
difference between computed pi and math.h M_PI = 0.000000000000298 (2.98e-13)
time to compute = 4.68279e-09 seconds
MPIRUN parpi with 3 node processes:
parallel program results with 3 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653590044)
difference between computed pi and math.h M_PI = 0.000000000000250 (2.50e-13)
time to compute = 3.11371e-09 seconds
MPIRUN parpi with 4 node processes:
parallel program results with 4 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589865)
difference between computed pi and math.h M_PI = 0.000000000000072 (7.19e-14)
time to compute = 2.29137e-09 seconds
MPIRUN parpi with 5 node processes:
parallel program results with 5 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589656)
difference between computed pi and math.h M_PI = 0.000000000000137 (1.37e-13)
time to compute = 1.91926e-09 seconds
MPIRUN parpi with 10 node processes:
parallel program results with 10 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589856)
difference between computed pi and math.h M_PI = 0.000000000000063 (6.26e-14)
time to compute = 9.81938e-10 seconds
MPIRUN parpi with 20 node processes:
parallel program results with 20 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589783)
difference between computed pi and math.h M_PI = 0.000000000000010 (9.77e-15)
time to compute = 4.93007e-10 seconds
MPIRUN parpi with 40 node processes:
parallel program results with 40 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589766)
difference between computed pi and math.h M_PI = 0.000000000000027 (2.71e-14)
time to compute = 2.49083e-10 seconds
MPIRUN parpi with 60 node processes:
parallel program results with 60 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589792)
difference between computed pi and math.h M_PI = 0.000000000000001 (8.88e-16)
time to compute = 1.67042e-10 seconds
MPIRUN parpi with 80 node processes:
parallel program results with 80 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589807)
difference between computed pi and math.h M_PI = 0.000000000000014 (1.42e-14)
time to compute = 1.25419e-10 seconds
MPIRUN parpi with 96 node processes:
parallel program results with 96 processes and 800000000 steps:
computed pi = 3.14159  (3.141592653589811)
difference between computed pi and math.h M_PI = 0.000000000000018 (1.78e-14)
time to compute = 1.62026e-10 seconds
ubuntu@ip-172-31-43-252:~/csci455/Lab3-Pi_Calculation$
```

Figure 3: For the sake of interest, here is the console sample identical to that found in Q1, but with `NUM_STEPS` also doubled.

parpi.c

```c
/* parpi.c
 * -------
 * Authors: Darwin Jacob Groskleg
 * CSCI 455
 * Lab 3 - Calculation of Pi
 *
 * Purpose: to determine the value of pi. The lab uses the method to evaluate
 *          the integral of 4/(1+x*x) between 0 and 1.
 *
 * See 'lecture09-Ch3-Embarassingly Parallel.ppt' for algorithm details.
 *
 * TODO:
 *  - [x] Complete the skeleton code.
 *  - [x] Can you run the code with 5, 10, 20, 40, 60 or more processors,
 *        respectively, to see the speedups? Do you see any unusual changes on
 *        speedup?
 *  - [x] Can you double the NUM_STEPS with the same number of processors as
 *        above, then compare and observe the speed ups?
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <locale.h>

#include <mpi.h>

#ifndef M_PI
/* copied from not-strictly-standard part of math.h */
#define M_PI    3.14159265358979323846
#endif  /* M_PI */

#define BASE_NUM_STEPS 400000000
int NUM_STEPS = BASE_NUM_STEPS;

int main(int argc, char *argv[]) {
    /* initialize for MPI */
    MPI_Init(&argc, &argv);

    if (argc>1)
        NUM_STEPS *= atoi(argv[1]);

    /* get number of processes */
    int cluster_size;
    MPI_Comm_size(MPI_COMM_WORLD, &cluster_size);

    /* get this process's number (ranges from 0 to cluster_size - 1) */
    int myid;
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);

    /* record start time */
    double start_time = MPI_Wtime();

    /* do computation */
    double step = 1.0/(double) NUM_STEPS;
```

```
56      double x;
57      double sum = 0.0;
58      for (int i=myid+1; i<=NUM_STEPS; i+=cluster_size) {
59          x = step * ((double) i - 0.5);
60          sum += 4.0 / (1.0 + x*x);
61      }
62      double piece_of_pi = step * sum;
63      double pi = 0.0;
64      MPI_Reduce(
65              &piece_of_pi,
66              &pi,    // receiver
67              1,      // data count
68              MPI_DOUBLE,
69              MPI_SUM,
70              0,      // root node rank (receiver)
71              MPI_COMM_WORLD
72          );
73
74      /* record end time */
75      double end_time = MPI_Wtime();
76      double seconds_per_tick = MPI_Wtick();
77      double seconds_elapsed = (end_time - start_time) * seconds_per_tick;
78
79      /* print results */
80      if (myid == 0) {
81          setlocale(LC_NUMERIC, ""); /* thousand separator for easy reading  */
82          printf("parallel program results with %'d processes and %'d steps:\n",
83                  cluster_size, NUM_STEPS);
84          printf("computed pi = %g  (%'17.15f)\n",pi, pi);
85          long double delta_err = fabs(pi - M_PI);
86          printf("difference between computed pi and math.h M_PI = ");
87          printf("%'17.15Lf (%1.2Le)\n", delta_err, delta_err);
88          printf("time to compute = %g seconds\n", seconds_elapsed);
89      }
90
91      /* clean up for MPI */
92      MPI_Finalize();
93
94      return 0;
95  }
```