CSCI 455: Lab #3 — Pi Calculation

Darwin Jacob Groskleg

Winter 2020

Contents

Pi Calculation

Q1. Any speedup or unusual changes?

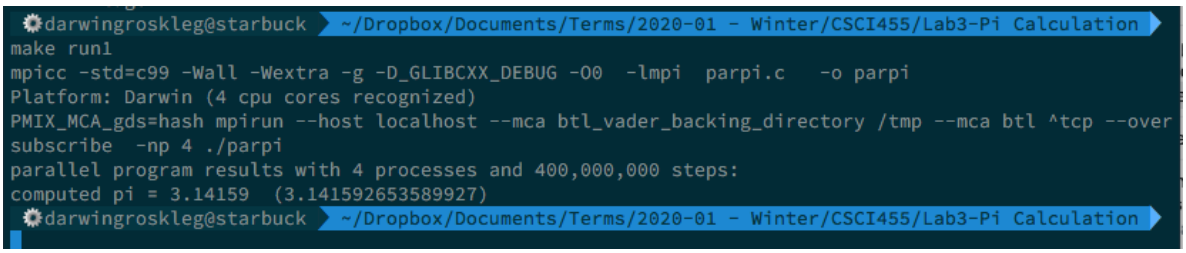

Figure 1: Code with 5, 10, 20, 40, 60 processors

Q2. Observations after doubling `NUM_STEPS`

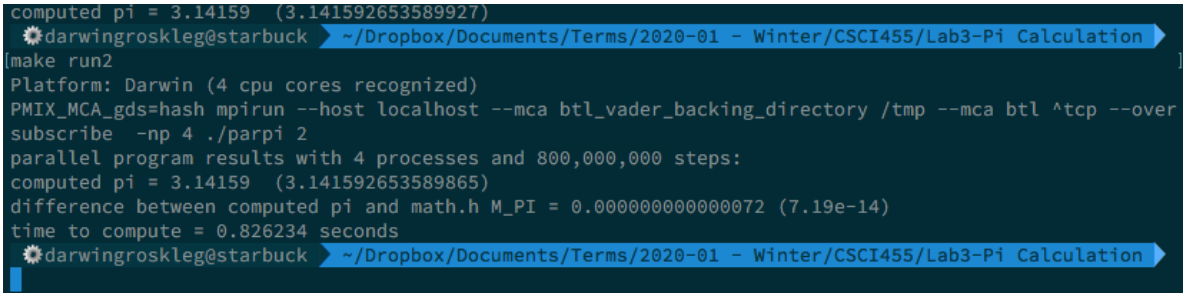

Figure 2: Same code as Q1 but `NUM_STEPS` is doubled

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
77      /* print results */
78      if (myid == 0) {
79          setlocale(LC_NUMERIC, ""); /* thousand separator for easy reading  */
80          printf("parallel program results with %'d processes and %'d steps:\n",
81                  cluster_size, NUM_STEPS);
82          printf("computed pi = %g  (%'17.15f)\n",pi, pi);
83          long double delta_err = fabs(pi - M_PI);
84          printf("difference between computed pi and math.h M_PI = ");
85          printf("%'17.15Lf (%1.2Le)\n", delta_err, delta_err);
86          printf("time to compute = %g seconds\n", end_time - start_time);
87      }
88
89      /* clean up for MPI */
90      MPI_Finalize();
91
92      return 0;
93  }
```