6550 Parallel Programming HW5

Riley Densley

October 2019

1 Mandelbrot on Single Process

For this program I used to process he described in class. I picked the bounds and found the number of iterations it took to get the magnitude of the complex number above 4. One complication I ran into was ensuring that I had my ups and downs correct as I stepped through the mandelbrot. On my machine this code took 5.63 seconds to run. On Watso it took 12.094 seconds.

2 Code

```
#include <iostream>
#include <mpi.h>
#include <random>
#include <unistd.h>
#include <cmath>
#include <fstream>
#include <chrono>
#define MCW MPI_COMM_WORLD

using namespace std;
//using namespace std::chrono;

struct Complex
{
  double r;
  double i;
};

Complex operator+(Complex s, Complex t)
{
  Complex v;
  v.r = s.r + t.r;
  v.i = s.i + t.i;
  return v;
```

```
}

Complex operator*(Complex s, Complex t)
{
  Complex v;
  v.r = s.r * t.r - s.i * t.i;
  v.i = s.r * t.i + s.i * t.r;
  return v;
}

int mandelbrot(Complex c, int maxIters)
{
  int i = 0;
  Complex z;
  z = c;
  while (i < maxIters && z.r * z.r + z.i * z.i < 4)
  {
    z = z * z + c;
    i++;
  }
  return i;
}

string COLORS[] = {
    "66 30 15 ",    // brown 3
    "25 7 26 ",     // dark violett
    "9 1 47 ",      // darkest blue
    "4 4 73 ",      // blue 5
    "0 7 100 ",     // blue 4
    "12 44 138 ",   // blue 3
    "24 82 177 ",   // blue 2
    "57 125 209 ",  // blue 1
    "134 181 229 ", // blue 0
    "211 236 248 ", // lightest blue
    "241 233 191 ", // lightest yellow
    "248 201 95 ",  // light yellow
    "255 170 0 ",   // dirty yellow
    "204 128 0 ",   // brown 0
    "153 87 0 ",    // brown 1
    "106 52 3 "     // brown 2
};

int main(int argc, char **argv)
{
  int rank, size;
  int data;
```

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MCW, &rank);
MPI_Comm_size(MCW, &size);
MPI_Status mystatus;
srand(time(NULL));

int maxIters = 500;
const int rows = 516;
const int cols = 516;
double re[2] = {stod(argv[1]), stod(argv[3])};
double im[2] = {stod(argv[2]), stod(argv[2]) + (stod(argv[3]) - stod(argv[1]))};
// For WATSO
// double re[2] = {-.72043, -.72019};
// double im[2] = {.2024, .2024 + (-.72019 + .72043)};

double stepC = double((re[1] - re[0]) / cols);
double stepR = double((im[1] - im[0]) / rows);

//diagonal -.722 .2003 -.718
// twist -.72043 .2024 -.72019

if (rank == 0)
{
  auto begin = chrono::high_resolution_clock::now();

  ofstream fout;
  fout.open("mandelout.ppm");
  fout << "P3\n"
       << rows << " " << cols << endl
       << "255\n";

  //initialize Array of random numbers
  // int subSize = rand() % 10 + 5;

  int **mand = new int *[rows];
  for (int r = 0; r < rows; r++)
    mand[r] = new int[cols];

  Complex current;

  for (int row = rows - 1; row > -1; row--)
  {
    for (int col = 0; col < cols; col++)
    {
      current.i = double(row) * stepR - im[1];
      current.r = double(col) * stepC + re[0];
```

```
    //current.r = col;
    mand[row][col] = mandelbrot(current, maxIters);
  }
  // fout << endl;
  // cout << row << endl;
}
// Write to file
for (int row = rows - 1; row >= 0; row--)
{
  for (int col = 0; col < cols; col++)
  {
    if (mand[row][col] >= maxIters)
    {
      fout << "0 0 0 ";
    }
    else
    {
      int colorStep = maxIters / 16;
      int j = 0;
      bool set = false;
      int current = mand[row][col];
      for (int i = 0; i < 16; i++)
      {
        if ((i + 1) * colorStep > current)
        {
          fout << COLORS[i];
          set = true;
          break;
        }
      }
      if (!(set))
        fout << COLORS[15];
    }
  }
  fout << endl;
}
fout.close();

auto end = chrono::high_resolution_clock::now();
auto dur = end - begin;
auto ms = std::chrono::duration_cast<std::chrono::milliseconds>(dur).count();
cout << "Total time taken is " << ms / 1000.0 << " seconds\n";

for (int i = 0; i < rows; i++)
  delete[] mand[i];
delete[] mand;
```

```
  }

  MPI_Finalize();

  return 0;
}
```

3 Output

3.1 Commands

```
mpic++ assign5.cpp
mpirun -np 4 ./a.out -.72043 .20240 -.72019
```

3.2 Output

```
Total time taken is 5.63 seconds
```