cs6550

Brett Bonar

October 5, 2018

1 Implementation

I separated the problem into processes by assigning each process a set of rows based on its rank and number of processes. This way each process would know what it needs to compute without requiring input or instructions from any master process.

The only MPI communication is an MPI Gather to collect all computed pixels from each process into the "master" process with rank 0 to output the results to a file. The RGB values of each pixel were saved as a single integer to save some space in transferring between processes.

2 Results

For 512 x 512 images, it took about 2.5 seconds to run using a single process. This goes down to about 1.9 seconds with two processes, 1.4 with four, and around 1 second with eight.

Any number of processors above eight would start to slow down the time slightly. This is likely because the Linux VM I used had eight cores and using more than eight processes would just create more communication overhead than necessary.

3 Code

```
1  #include <iostream>
2  #include <unistd.h>
3  #include <cmath>
4  #include <mpi.h>
5  #include <string>
6  #include <vector>
7  #define MCW MPI_COMM_WORLD
8
9  using namespace std;
10
11 struct Complex{
12   double r;
```

```
13    double i;
14  };
15
16  Complex operator * (Complex a, Complex b){
17    Complex c;
18    c.r = a.r*b.r-a.i*b.i;
19    c.i = a.r*b.i+a.i*b.r;
20    return c;
21  }
22
23  Complex operator + (Complex a, Complex b){
24    Complex c;
25    c.r = a.r+b.r;
26    c.i = a.i+b.i;
27    return c;
28  }
29
30  Complex operator - (Complex a, Complex b){
31    Complex c;
32    c.r = a.r-b.r;
33    c.i = a.i-b.i;
34    return c;
35  }
36
37  int mbrot_iters(Complex c){
38    int i=0;
39    Complex z = c;
40    while(z.r*z.r+z.i*z.i<2.0*2.0 && i<1024){
41      z = z*z+c;
42      i++;
43    }
44    return i;
45  }
46
47
48  int main(int argc, char **argv){
49    int rank, size;
50    int data;
51    int PIXELS = 512;
52
53    MPI_Init(&argc, &argv);
54    MPI_Comm_rank(MCW, &rank);
55    MPI_Comm_size(MCW, &size);
56
57    if (argc > 1)
58    {
59      PIXELS = std::stoi(argv[1]);
60    }
61
62    Complex c1,c2,cx,cdiff;
63    double rinc;
64    double iinc;
65    int iters;
66    c1.r = -1.5;
67    c2.r = -0.5;
68    c1.i = 1.0;
69    c2.i = 0;
```

```
70
71    cdiff = c2 - c1;
72    rinc = cdiff.r / PIXELS;
73    iinc = cdiff.i / PIXELS;
74
75    int rowsPerProcess = PIXELS / size;
76    int numLocalColors = PIXELS * PIXELS / size;
77    int numGlobalColors = PIXELS * PIXELS;
78    std::vector<int> localColors(numLocalColors);
79    std::vector<int> globalColors(numGlobalColors);
80
81    for(int i = rank * rowsPerProcess, row = 0; i < rank *
          rowsPerProcess + rowsPerProcess; ++i, ++row)
82    {
83      for(int j = 0; j < PIXELS; ++j)
84      {
85        cx.i = c1.i + j * iinc;
86        cx.r = c1.r + i * rinc;
87        iters = mbrot_iters(cx);
88
89        int r,g,b;
90        if (iters == 1024)
91        {
92          r = 0;
93          g = 0;
94          b = 0;
95        }
96        else
97        {
98          r = (log(iters) / log(1024)) * 255;
99          g = 0;
100         b = 255 - (log(iters) / log(1024)) * 255;
101       }
102
103       localColors[row * PIXELS + j] = (r << 16) + (g << 8) + b;
104     }
105   }
106
107   MPI_Gather(localColors.data(), numLocalColors, MPI_INT,
108     globalColors.data(), numLocalColors, MPI_INT, 0, MCW);
109
110   if (rank == 0)
111   {
112     cout << "P3" <<endl;
113     cout << PIXELS << " " << PIXELS << endl;
114     cout << "255" <<endl;
115
116     for(int i = 0; i < PIXELS; ++i)
117     {
118       for(int j = 0; j < PIXELS; ++j)
119       {
120         int red = (globalColors[i * PIXELS + j] >> 16) & 0xFF;
121         int green = (globalColors[i * PIXELS + j] >> 8) & 0xFF;
122         int blue = globalColors[i * PIXELS + j] & 0xFF;
123         cout << red << " " << green << " " << blue << " ";
124       }
125       cout << endl;
```

```
    }
  }

  MPI_Finalize();

  return 0;
}
```

4 Compile and Run Commands

```
mpic++ Assignment6/Assignment6.cpp -o Assignment6/run.out
time mpirun -np 8 Assignment6/run.out 512 > mbrotp.ppm
```