HW06 - Parallel Mandelbrot

CS5500

Chelsi Gupta

October 4, 2018

1 Steps for Implementation

- I decided to parallelize my mandelbrot by dividing the whole image(i.e. a numpy array) into parts and computing each part on a different process.
- The image has been divided according to rows(i.e. according to the real axis) and then all the rows are gathered from all processes and stacked together using 'numpy.vstack' on process 0.
- I compiled and ran the program using 'mpirun –oversubscribe -np 10 python parallel_mandelbrot.py > mandel.ppm'. This saved the output to a ppm format which has been converted to png format so as to be supported by latex.
- It took 18.35 seconds when I ran the program on 10 nodes in my cluster.

2 Code

```
import time
start_time = time.time()
from mpi4py import MPI
comm = MPI.COMM_WORLD    # Defines the default communicator
import numpy
import matplotlib.pyplot as plt


def mandelbrot(Re, Im, max_iter):
    c = complex(Re, Im)
    z = 0.0j
    count = 0

    while((z.real*z.real + z.imag*z.imag) <= 4 and count < max_iter):
        z = z*z + c
        count += 1

    return count
```

```
rows = 512
columns = 512

size = comm.Get_size()  # Stores the number of processes in size.
rank = comm.Get_rank()  # Stores the rank (pid) of the current process

process_rows = rows//size  #number of rows that each process calculates
if(rank==0):
    process_rows+=(rows%size)

width= 3/size
p1=-2 + (width*rank)     #point where the process starts iterating on real line
p2=p1+width              #point where the process ends iterating on real line

result = numpy.zeros([process_rows,columns])
for row_index, Re in enumerate(numpy.linspace(p1,p2, num=process_rows)):
    for column_index, Im in enumerate(numpy.linspace(-1, 1, num=columns)):
        result[row_index, column_index] = mandelbrot(Re, Im, 1024)

result = comm.gather(result, root=0)

if rank==0:
    result=numpy.vstack(result)
    plt.imshow(result.T, cmap = 'RdBu', interpolation = 'bilinear', extent = [-2, 1, -1, 1])
    plt.xlabel('Re')
    plt.ylabel('Im')
    plt.savefig('mandel'+'.png',dpi=100)
    print("--- %s seconds ---" % (time.time() - start_time))
    plt.show()
```

3 Output

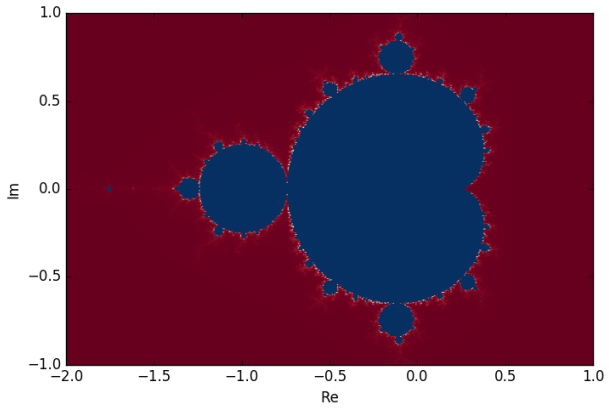

Figure 1: Mandelbrot

References

- https://www.bu.edu/pasi/files/2011/01/Lisandro-Dalcin-mpi4py.pdf