Numerical Scientific Computing – Mini Project Part 2

Optimization of data types

Size: 10000

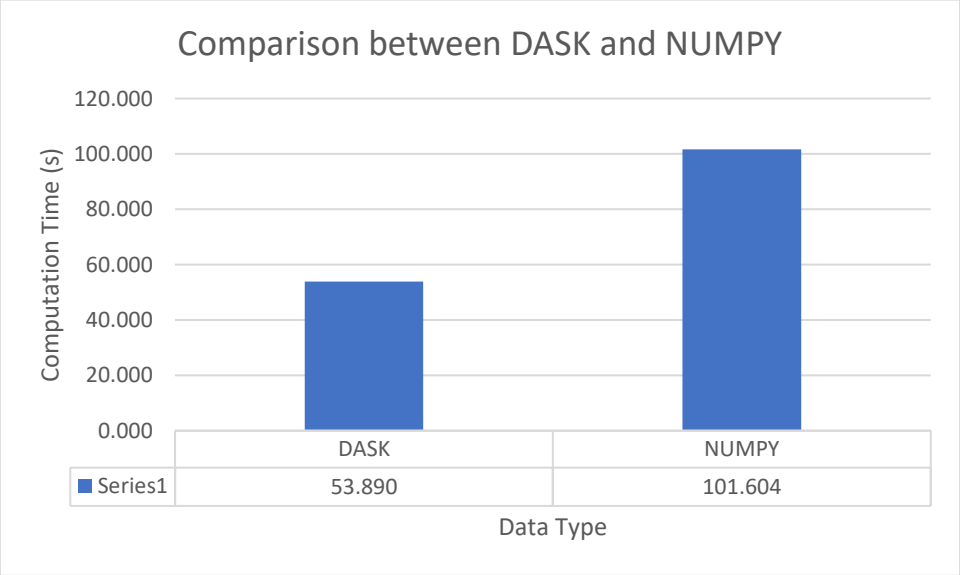
File: mandelbrot_datatypes.py

	Complex64	Complex128
Float16	155.1	193.18
Float32	157.76	199.54
Float64	199.08	223.37

Execution time between NUMPY and DASK version

Size: 8000

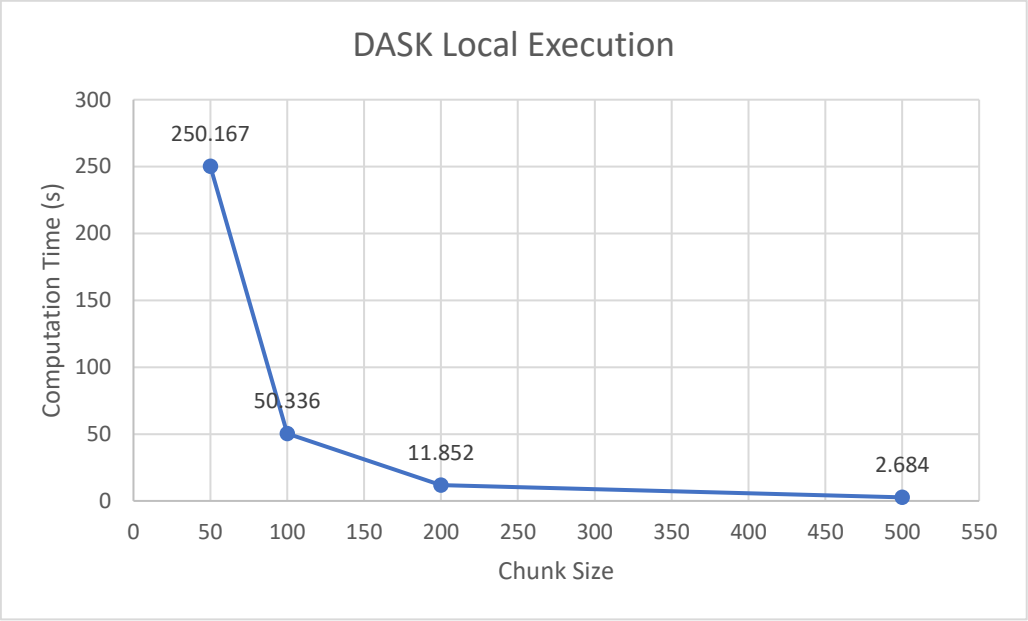
File: mandelbrot_dask.py



Local DASK execution

Size: 1000

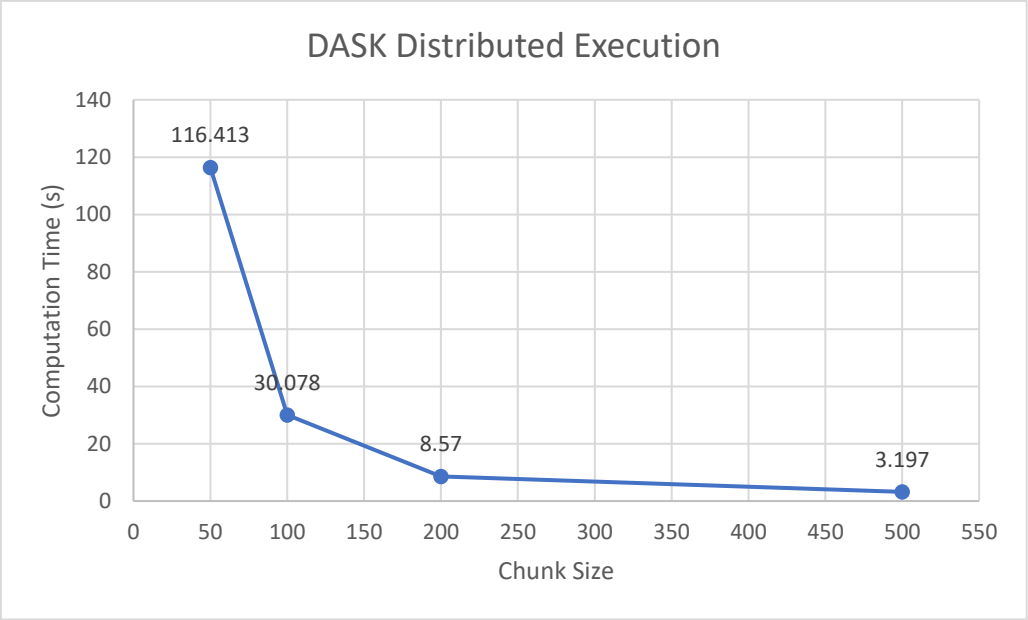
File: mandelbrot_dask.py



Distributed DASK execution

Size: 1000

File: mandelbrot_dask.py



Chunk Size Performance

In the previous two tests, it was found for both local and distributed that a chunk size of 500 was the best performing.

Improvements/optimizations

1. Stops early if a point is already diverged. See the end of the Mandelbrot function in mandelbrot_dask.py, mandelbrot_vectorized.py and mandelbrot_datatypes.py:

```
38
39          # Stops early if the absolute value of z is greater than the threshold (point diverged)
40          mandelbrot_mask[da.abs(z) > threshold] = False
41
```

2. The NumPy and Dask version of the Mandelbrot is optimized to run on the most optimal datatype based on the first computation test. See mandelbrot_vectorized.py and mandelbrot_dask.py:

```
18
19      divergence_time = numpy.zeros(c.shape, dtype=numpy.float16)
20
```

```
40
41      # Generates linear spaces with pRE and pIM elements respectively around the plane of the Mandelbrot set
42      x_space = numpy.linspace(-2.3, 0.8, pRE, dtype=numpy.float16).reshape((1, pRE))
43      y_space = numpy.linspace(-1.2, 1.2, pIM, dtype=numpy.float16).reshape((pIM, 1))
44
```

3. Using dask.abs() is more optimal than using numpy.abs(), since Dask uses lazy evaluation.

```
38
39          # Stops early if the absolute value of z is greater than the threshold (point diverged)
40          mandelbrot_mask[da.abs(z) > threshold] = False
41
```