Parallelization in Python —Dask

Arnav Sood

NYU

April 21, 2016

Dask Hierarchy

Data Structures: Jobs fit easily into operations on common data structures.

Dask Hierarchy

Data Structures: Jobs fit easily into operations on common data structures.
Imperatives: Job is more complex, but doesn't warrant an entire custom graph.

Dask Hierarchy

Data Structures: Jobs fit easily into operations on common data structures.
Imperatives: Job is more complex, but doesn't warrant an entire custom graph.
Custom Graphs: Intricate job.

Dask Hierarchy

Data Structures: Jobs fit easily into operations on common data structures.
Imperatives: Job is more complex, but doesn't warrant an entire custom graph.
Custom Graphs: Intricate job.

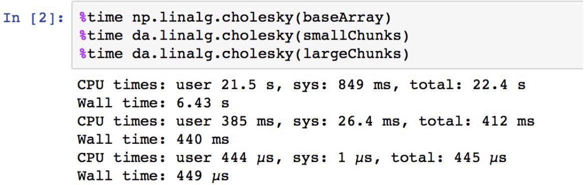
Schedulers: Take the above specs, and run as efficiently as possible.

Chunked Algorithms

```
# Try chunked matrix operations
import dask.array as da
import numpy as np
baseArray = np.random.rand(10000,10000)
# Make positive-definite
baseArray = np.dot(baseArray,baseArray.transpose())
smallChunks = da.from_array(baseArray,chunks=(100))
largeChunks = da.from_array(baseArray,chunks=(1000))
```

Chunked Algorithms

```
# Try chunked matrix operations
import dask.array as da
import numpy as np
baseArray = np.random.rand(10000,10000)
# Make positive-definite
baseArray = np.dot(baseArray,baseArray.transpose())
smallChunks = da.from_array(baseArray,chunks=(100))
largeChunks = da.from_array(baseArray,chunks=(1000))
```



```
In [2]: %time np.linalg.cholesky(baseArray)
        %time da.linalg.cholesky(smallChunks)
        %time da.linalg.cholesky(largeChunks)

CPU times: user 21.5 s, sys: 849 ms, total: 22.4 s
Wall time: 6.43 s
CPU times: user 385 ms, sys: 26.4 ms, total: 412 ms
Wall time: 440 ms
CPU times: user 444 µs, sys: 1 µs, total: 445 µs
Wall time: 449 µs
```

Lazy Evaluation

```
npOnes = np.ones((10000,10000))
%time np.exp(npOnes)[1:100,1:100]

daskOnes = da.ones((10000,10000), chunks=(100))
%time da.exp(daskOnes)[1:100,1:100]
```

Lazy Evaluation

```
npOnes = np.ones((10000,10000))
%time np.exp(npOnes)[1:100,1:100]

daskOnes = da.ones((10000,10000), chunks=(100))
%time da.exp(daskOnes)[1:100,1:100]
```

Result: 10× speedup. (970ms → 90ms)

Ghosting

Internals: Dictionary

```
import itertools slice =
itertools.islice(smallChunks.dask.items(), 0, 1) for
key, value in slice:  print(key,value)
```

Internals: Dictionary

```
import itertools slice =
itertools.islice(smallChunks.dask.items(), 0, 1) for
key, value in slice:  print(key,value)
```

Result:

Key: from-array-8c463a3b962efd2e81b47e1832acfb81’, 83, 94)
Value: (<function getarray at 0x1083260d0>, ‘from-array-8c463a3b962efd2e81b47e1832acfb81’, (slice(8300, 8400, None), slice(9400, 9500, None))

API

Subset of NumPy API.

API

Subset of NumPy API.
Good things:

API

Subset of NumPy API.
Good things:
Bad things:

API

Subset of NumPy API.
Good things:
Bad things:
Reference: dir(dask.array)

API

Subset of NumPy API.
Good things:
Bad things:
Reference: dir(dask.array)