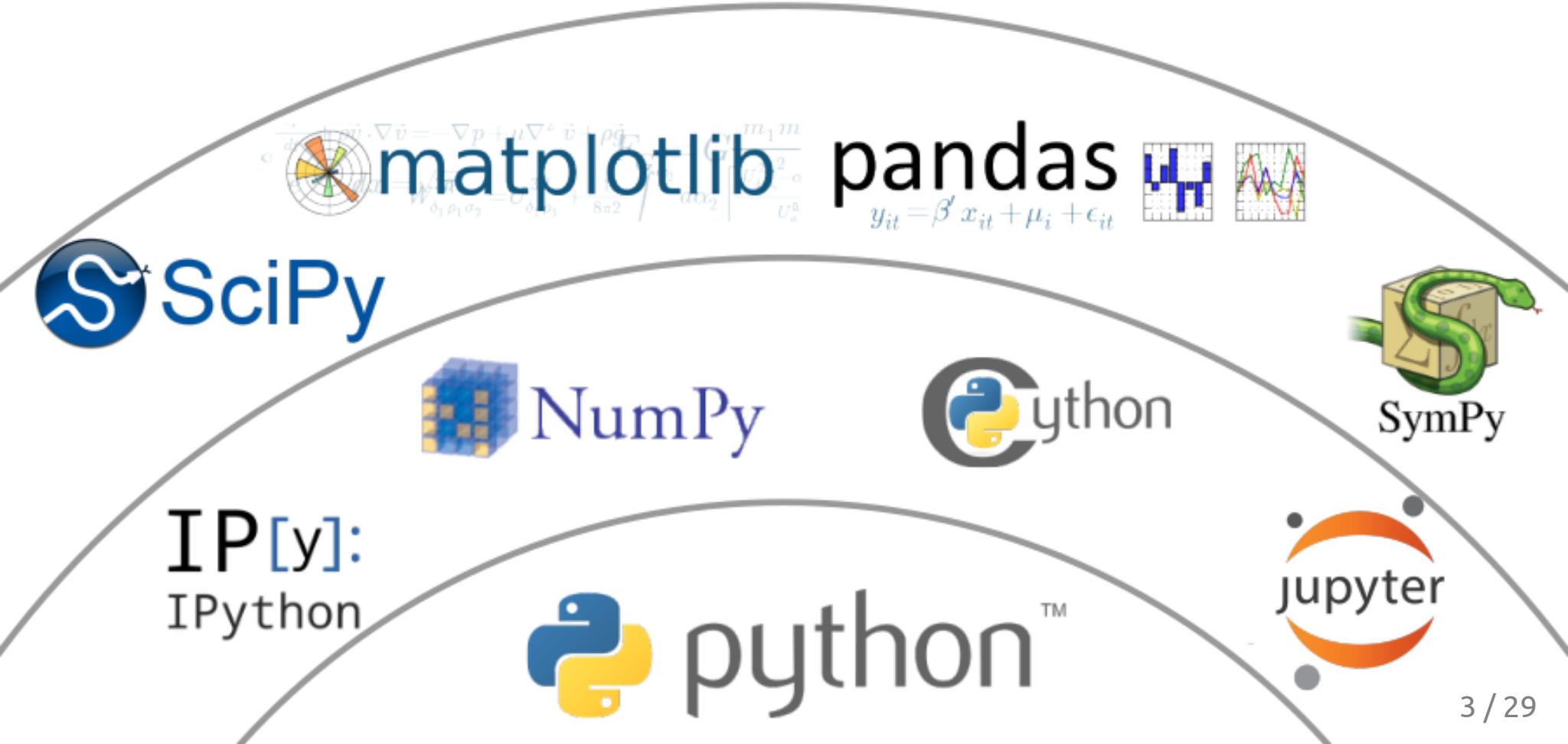
Dask

extending Python data tools for parallel and distributed computing

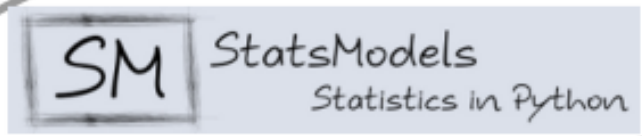
Joris Van den Bossche - FOSDEM 2017

Python's scientific/data tools ecosystem

Thanks to Jake VanderPlas for the figure


matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

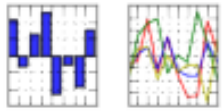
(and many, many more)



scikits-image
image processing in python

pandas

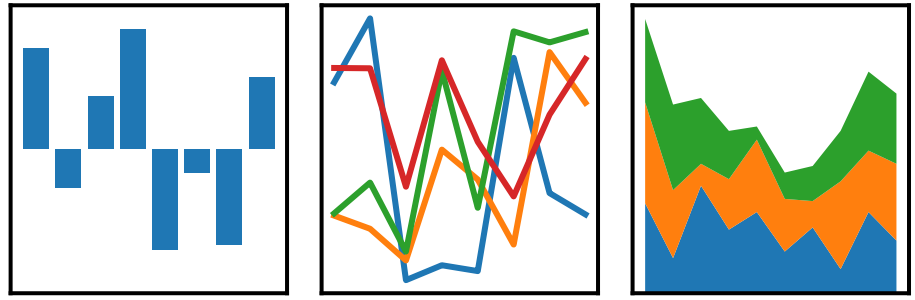
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



IP[y]:
IPython

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Provides high-performance, easy-to-use data structures and tools
- Widely used for doing practical data analysis in Python
- Suited for tabular data (e.g. column data, spread-sheets, databases)

```
import pandas as pd

df = pd.read_csv("myfile.csv")

subset = df[df['value'] > 0]

subset.groupby('key').mean()
```

Python has a fast and pragmatic data science ecosystem

Python has a fast and pragmatic data science ecosystem

... restricted to in-memory and a single core

a flexible library for parallelism

Dask is

A parallel computing framework

Lets you work on larger-than-memory datasets

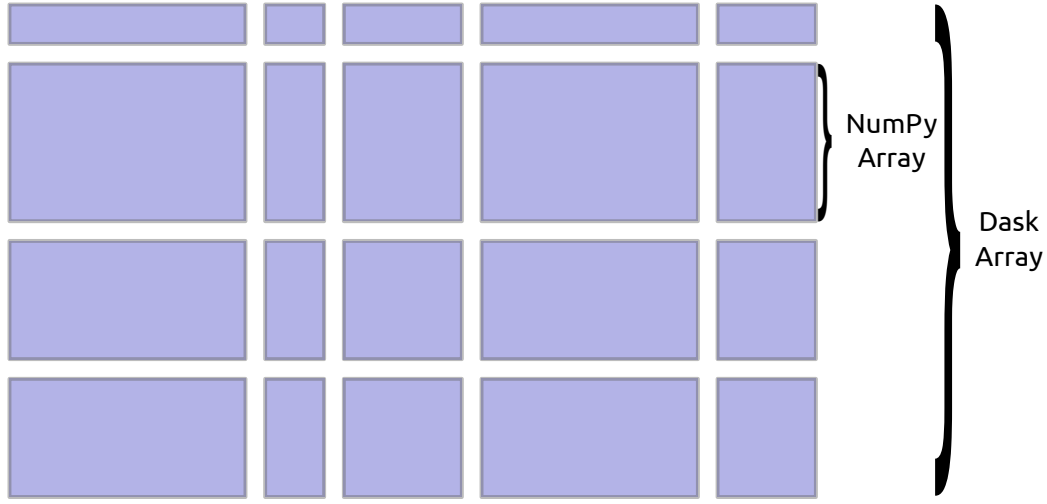
Written in pure Python

That leverages the excellent Python ecosystem

Using blocked algorithms and task scheduling

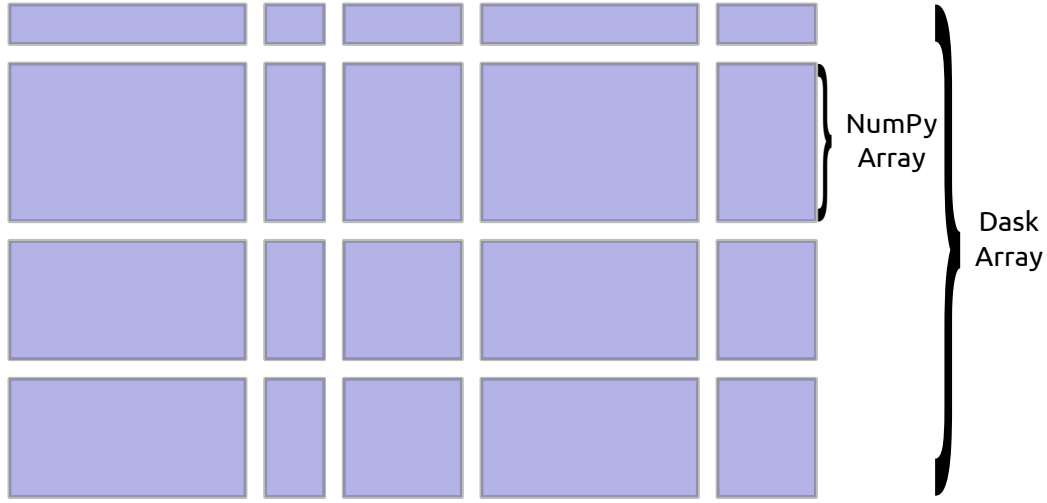
Dask.array

- Parallel and out-of-core array library
- Mirrors NumPy interface
- Coordinate many NumPy arrays into single logical Dask array



Dask.array

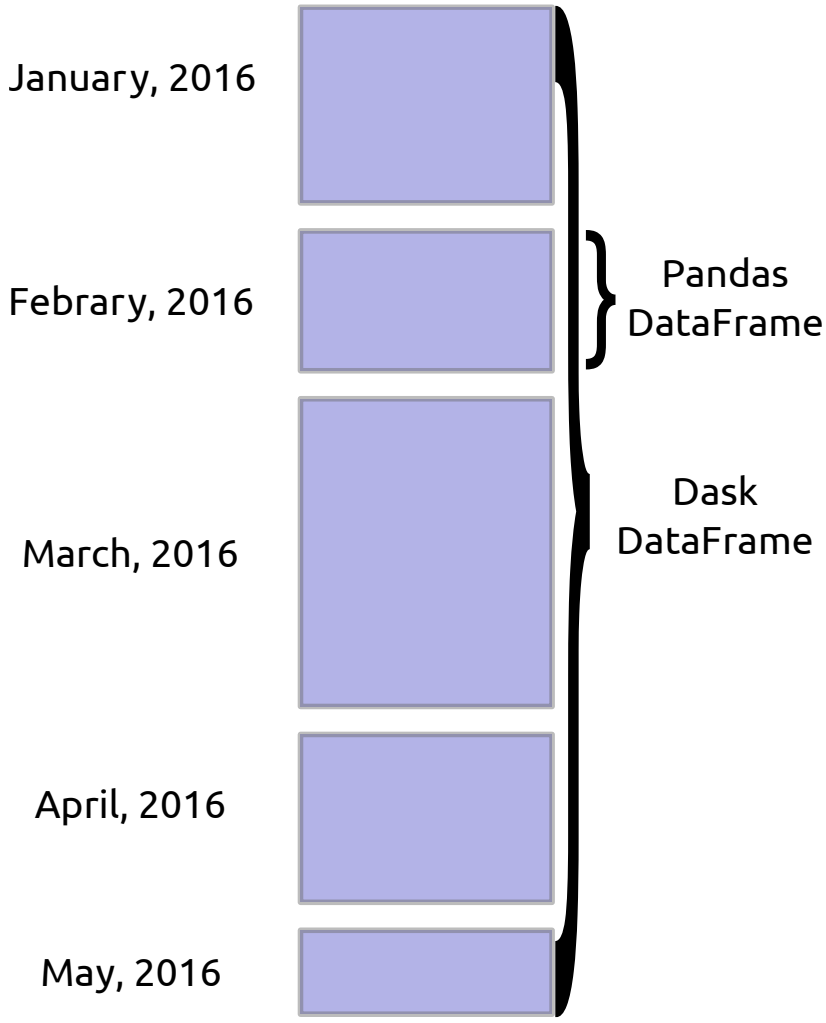
- Parallel and out-of-core array library
- Mirrors NumPy interface
- Coordinate many NumPy arrays into single logical Dask array




```
import numpy as np
x = np.random.random(...)

u, s, v = np.linalg.svd(x.dot(x.T))
```

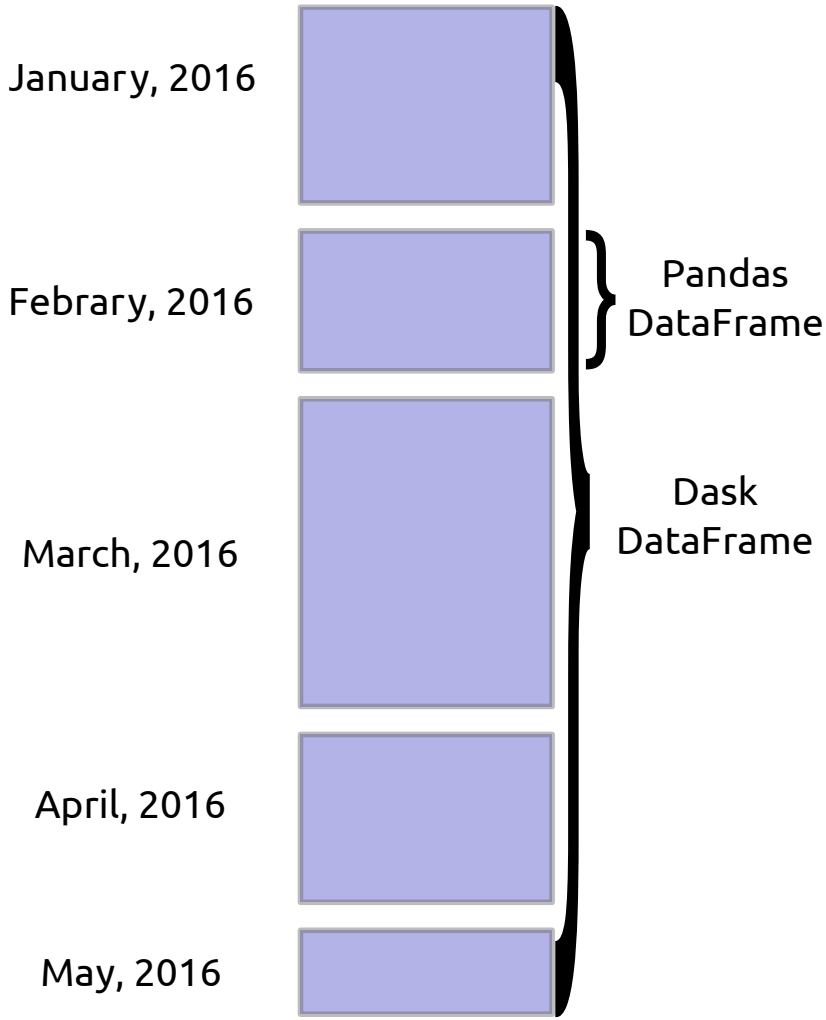
```
import dask.array as da
x = da.random.random(...,
        chunks=(1000, 1000))
u, s, v = da.linalg.svd(x.dot(x.T))
```

Dask.dataframe



- Parallel and out-of-core dataframe library
- Mirrors the Pandas interface
- Coordinates many Pandas DataFrames into single logical Dask DataFrame
- Index is (optionally) sorted, allowing for optimizations

Dask.dataframe

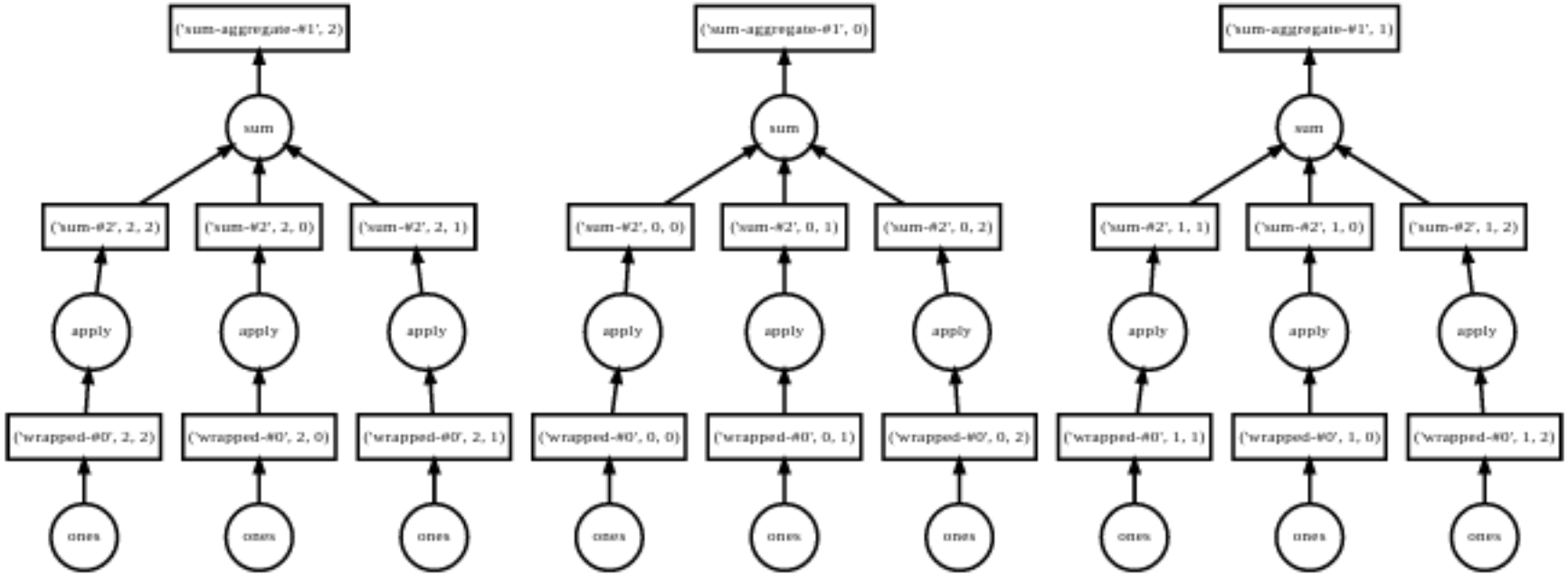



```
import pandas as pd
df = pd.read_csv('2015-01-01.csv')
res = df.groupby('user_id').mean()
```

```
import dask.dataframe as dd
df = dd.read_csv('2015-*-*.csv')
res = df.groupby('user_id').mean()
res.compute()
```

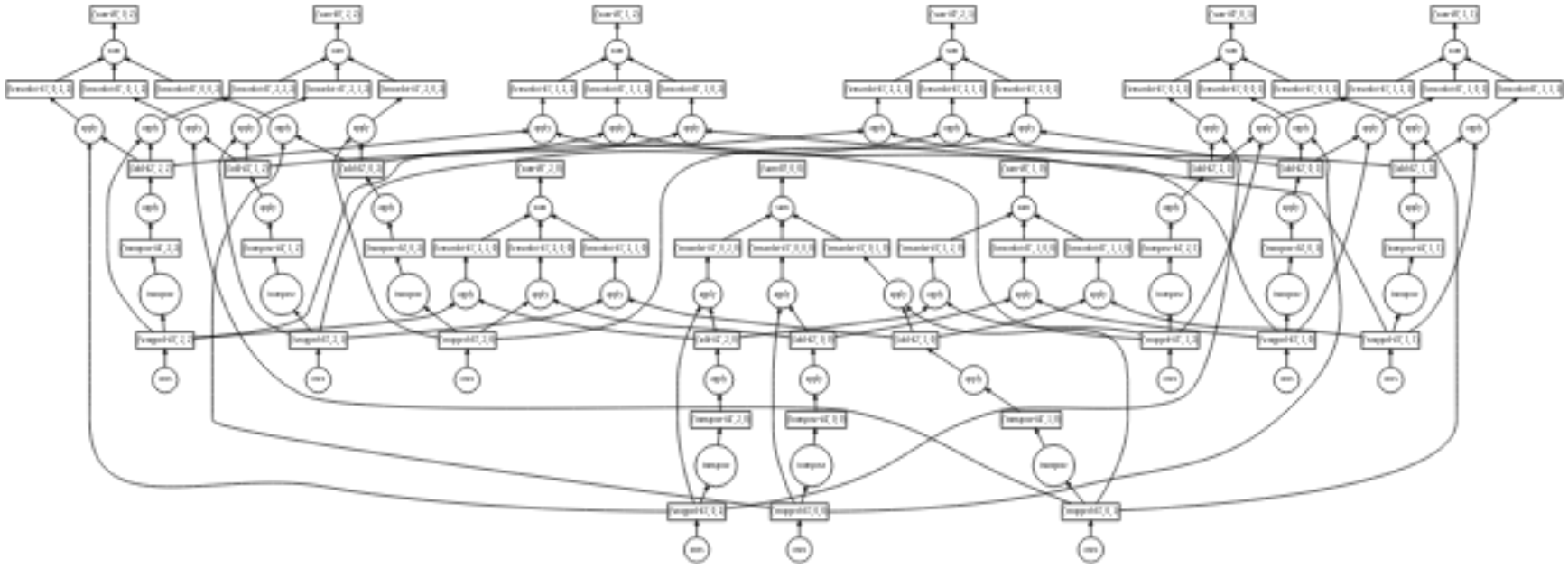
Complex graphs

ND-Array - sum




```
x = da.ones((15, 15), (5, 5))
x.sum(axis=0)
```

ND-Array - matrix multiply

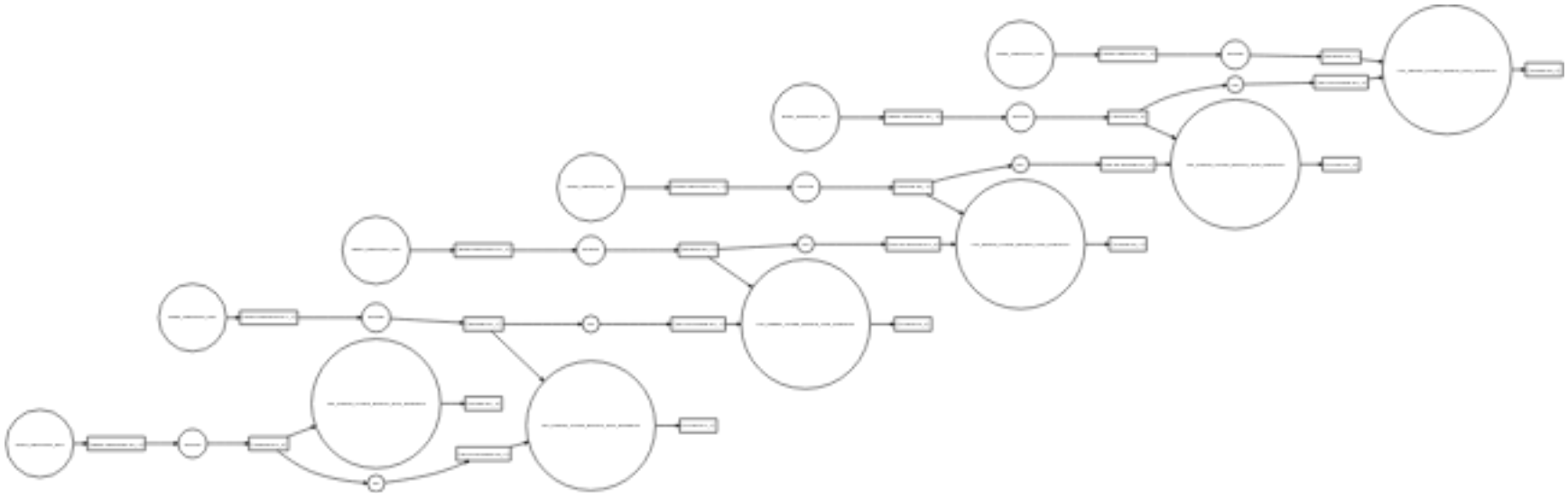


```
x = da.ones((15, 15), (5, 5))
x.dot(x.T + 1)
```

Efficient timeseries - resample

```
df.value.resample('1w').mean()
```

Efficient rolling



```
df.value.rolling(100).mean()
```

Some problems don't fit well into collections

Dask Delayed

- Tool for creating arbitrary task graphs
- Dead simple interface (one function)

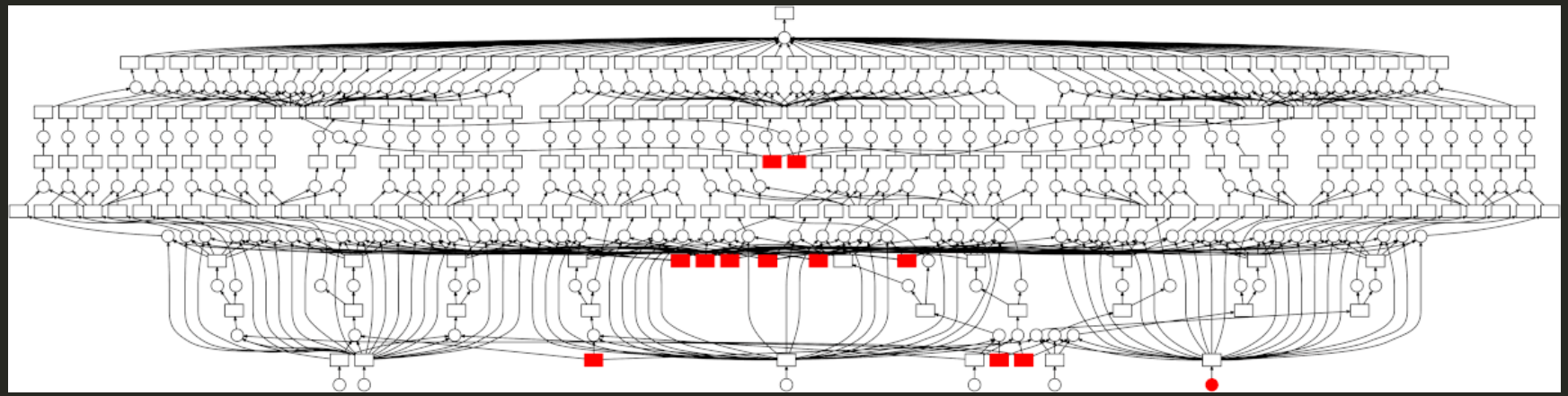
```
_

results = {}

for a in A:
    for b in B:
        results[a, b] = fit(a, b)

best = score(results)
_
```

Dask Delayed

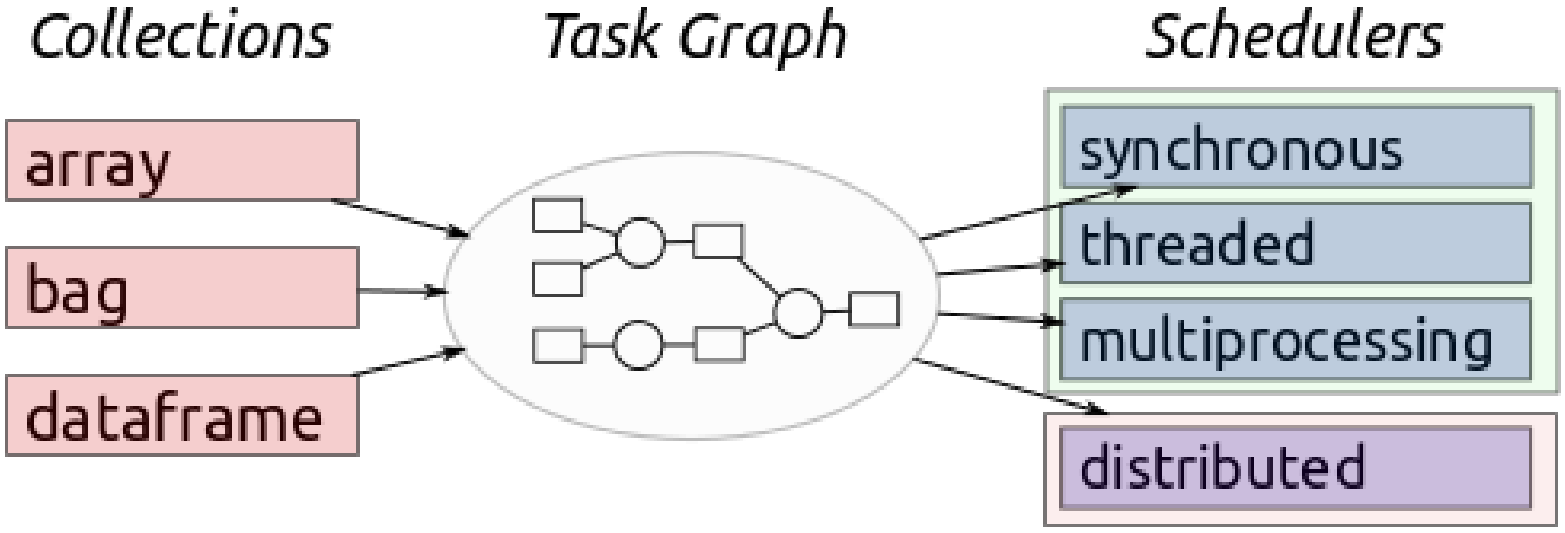
- Tool for creating arbitrary task graphs
- Dead simple interface (one function)

```
from dask import delayed

results = {}

for a in A:
    for b in B:
        results[a, b] = delayed(fit)(a, b)

best = delayed(score)(results)
result = best.compute()
```

Collections author task graphs

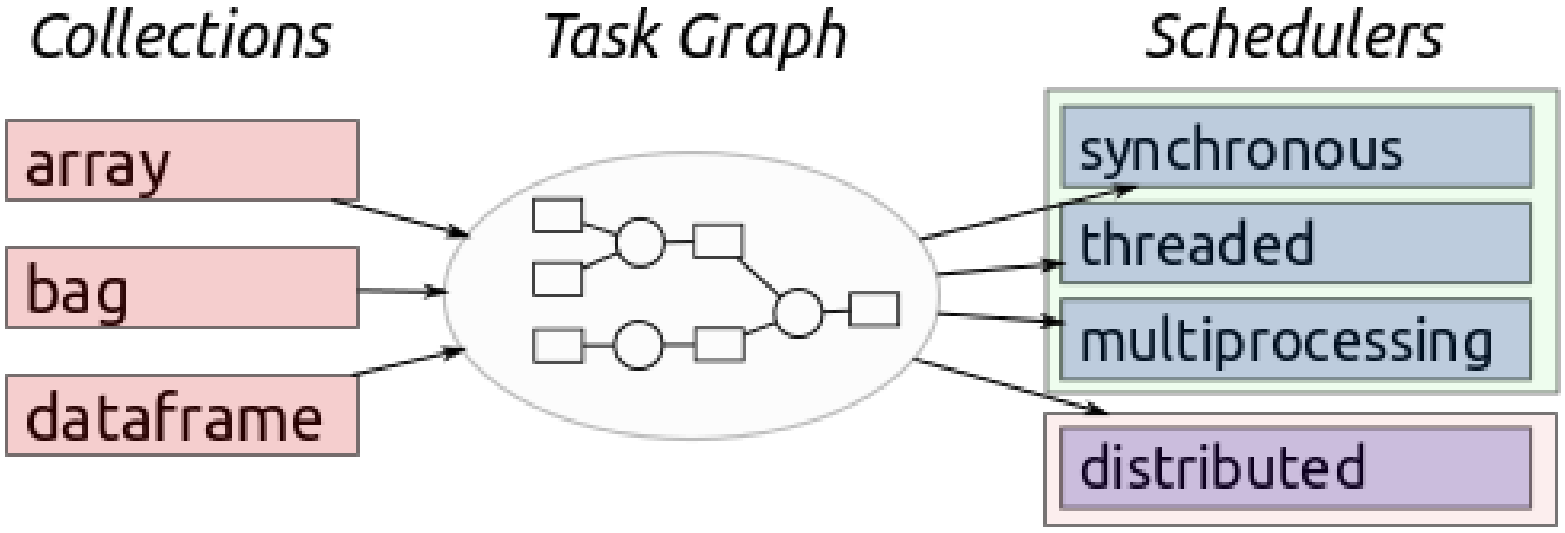


Now we need to run them efficiently



Collections build task graphs

Schedulers execute task graphs



Collections build task graphs

Schedulers execute task graphs

Dask schedulers target different architectures

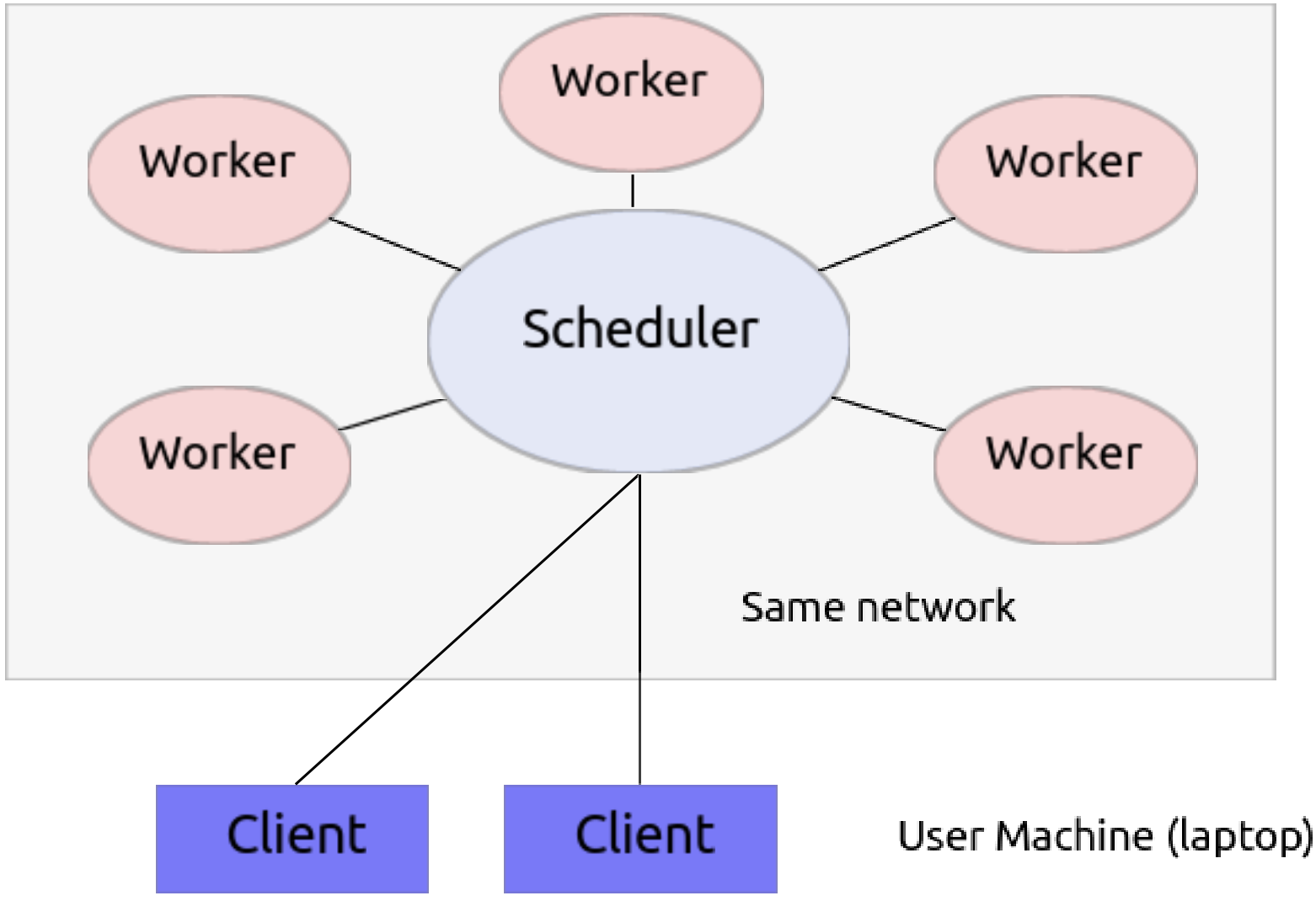
Easy swapping enables scaling up and down

Single Machine Scheduler

Optimized for larger-than-memory use

- **Parallel CPU**: Uses multiple threads or processes
- **Minimizes RAM**: Choose tasks to remove intermediates
- **Low overhead**: ~100us per task
- **Concise**: ~600 LOC, stable

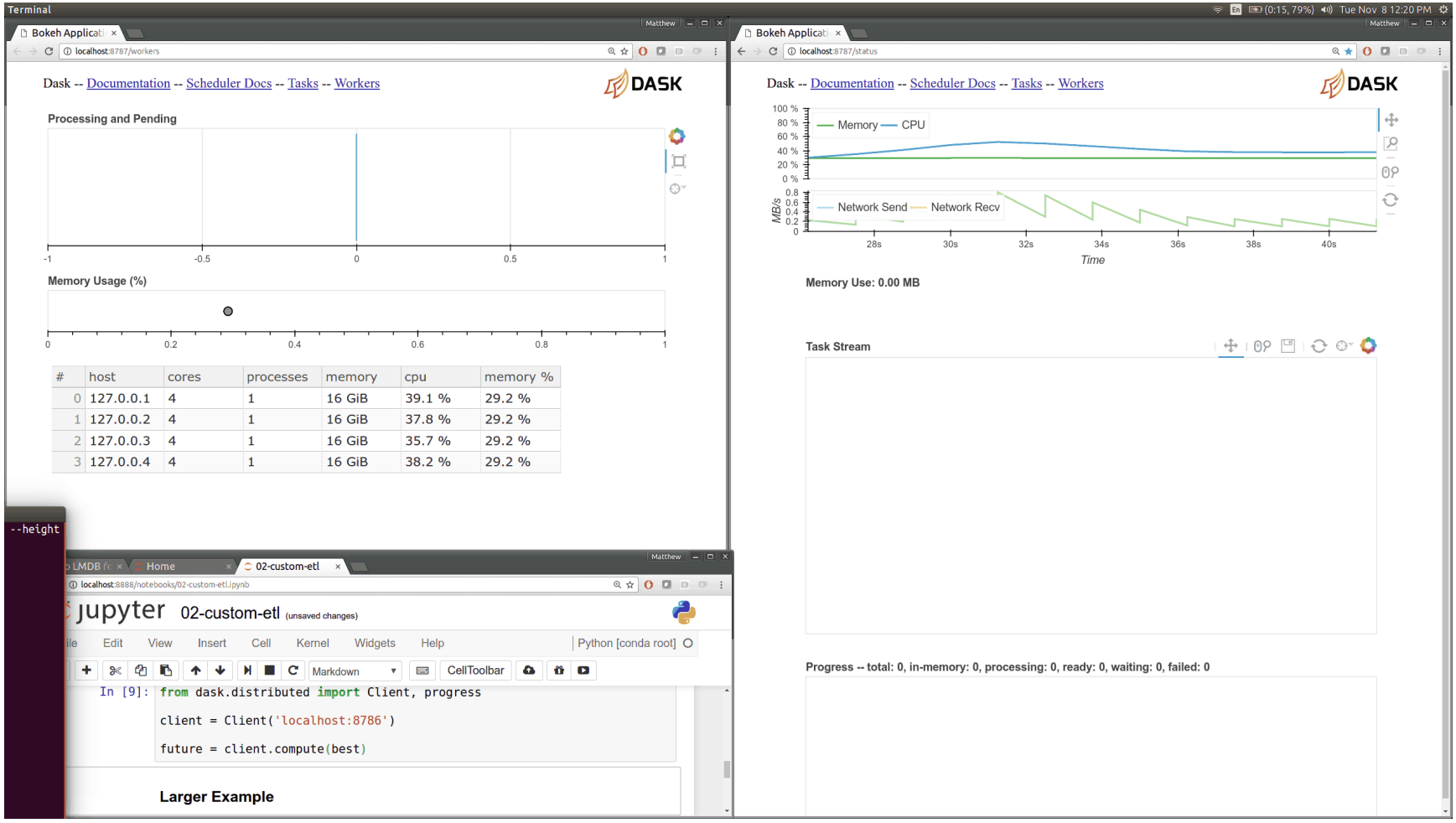
Distributed Scheduler



Distributed Scheduler

- **Distributed**: One scheduler coordinates many workers
- **Data local**: Tries to moves computation to "best" worker
- **Asynchronous**: Continuous non-blocking conversation
- **Multi-user**: Several users can share the same system
- **HDFS Aware**: Works well with HDFS, S3, YARN, etc..
- **Less Concise**: ~3000 LOC Tornado TCP application

Visual dashboards



To summarise: Dask is

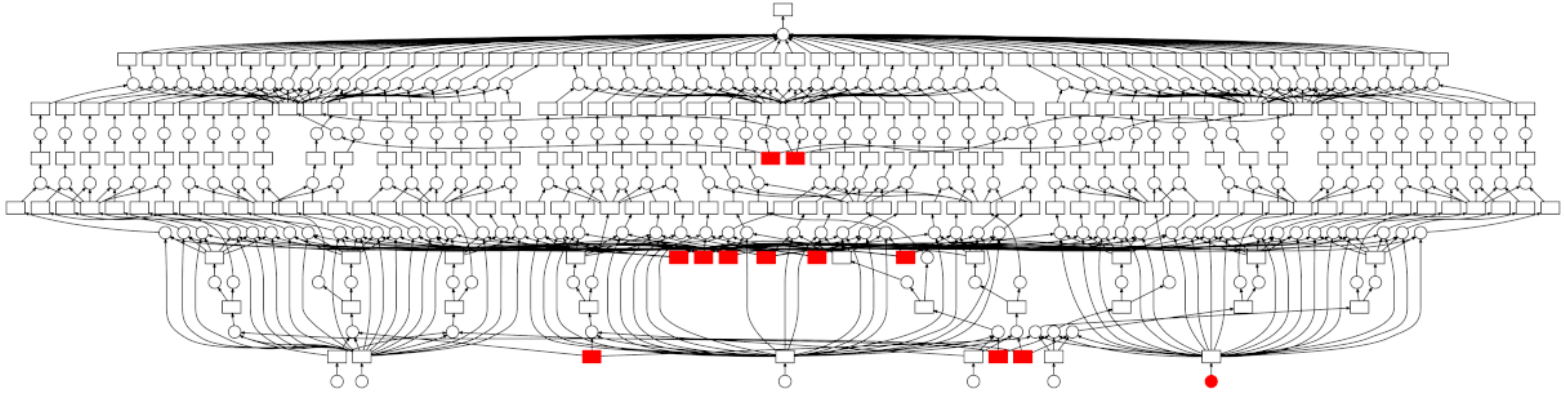
Dynamic task scheduler for arbitrary computations

- **Familiar**: Implements NumPy/Pandas interfaces
- **Flexible**: Handles arbitrary task graphs efficiently (custom workloads, integration with other projects)
- **Fast**: Optimized for demanding applications
- **Scales up**: Runs resiliently on clusters with 1000s of cores
- **Scales down**: Pragmatic on a laptop
- **Responsive**: for interactive computing

Dask ***builds on*** the existing Python ecosystem.

Acknowledgements: slides partly based on material from dask developers Matthew Rocklin and Jim Crist (Continuum Analytics)

http://dask.pydata.org



About me

- Researcher at Vrije Universiteit Brussel (VUB), and contractor for Continuum Analytics
- PhD bio-science engineer, air quality research
- pandas core dev

https://github.com/jorisvandenbossche

@jorisvdbossche