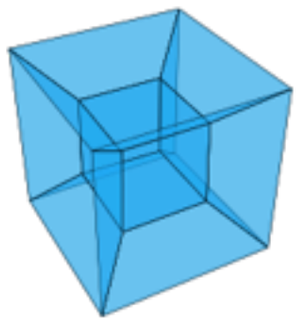
CONTINUUM ANALYTICS

Python in Hadoop Ecosystem
Blaze and Bokeh



Presented by: Andy R. Terrel

About Continuum Analytics

CONTINUUM ANALYTICS

http://continuum.io/

We build technologies that enable analysts and data scientist to answer questions from the data all around us.

Areas of Focus

- Software solutions
- Consulting
- Training

Committed to Open Source

- Anaconda: Free Python distribution
- Numba, Conda, Blaze, Bokeh, dynd
- Sponsor

PyData NumFOCUS Open Code, Better Science

About Andy

Andy R. Terrel
@aterrel
Chief Scientist,
Continuum Analytics

President,
NumFOCUS

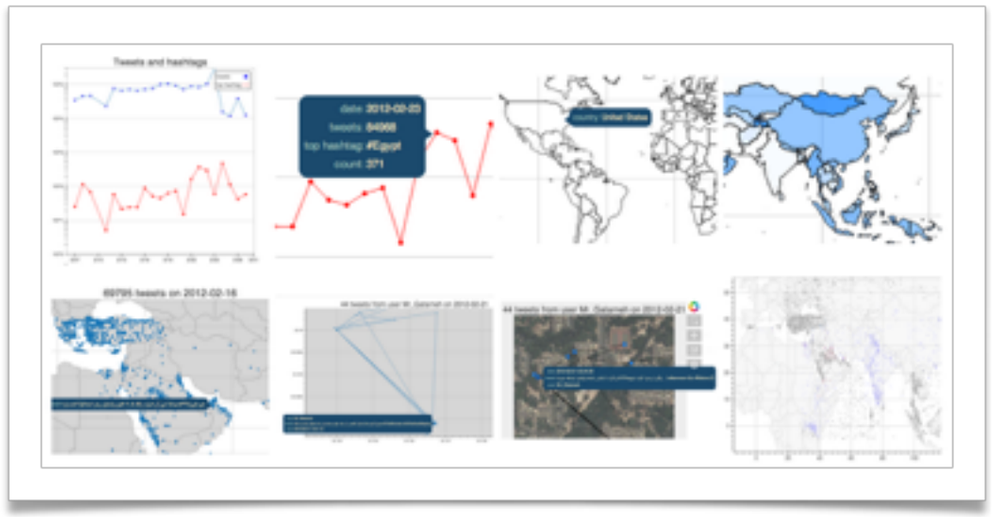
Background:

- High Performance Computing
- Computational Mathematics
- President, NumFOCUS foundation

Experience analyzing diverse datasets:

- Finance
- Simulations
- Web data
- Social media

About this talk



Visualizing Data
with Blaze and Bokeh

Objective

Introduction to large-scale data analytics and interactive visualization

Structure

1. Discussion of Hadoop
2. Large scale data analytics - Blaze
3. Interactive data visualization - Bokeh

Large scale data analytics - An Overview

Distributed Systems

hadoop

Spark

HIVE

mahout

RHadoop

Scientific Computing

The HDF Group

Numba

PyTables

NumPy

bcolz

scikit learn

pandas

SM

KNIME

WEKA The University of Waikato

R

SQLAlchemy

SQLite

mongoDB

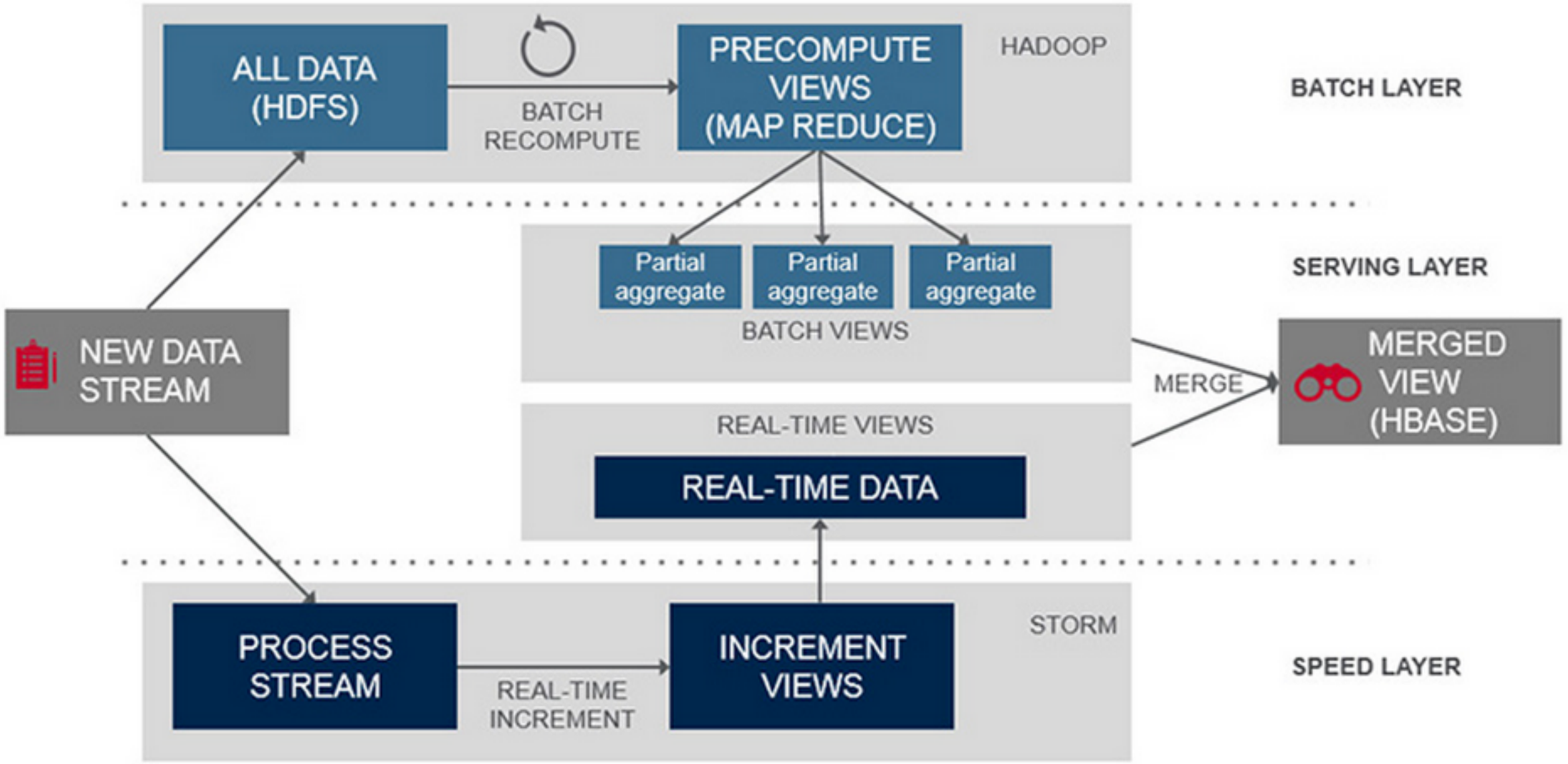
PostgreSQL

MySQL

BI - DB

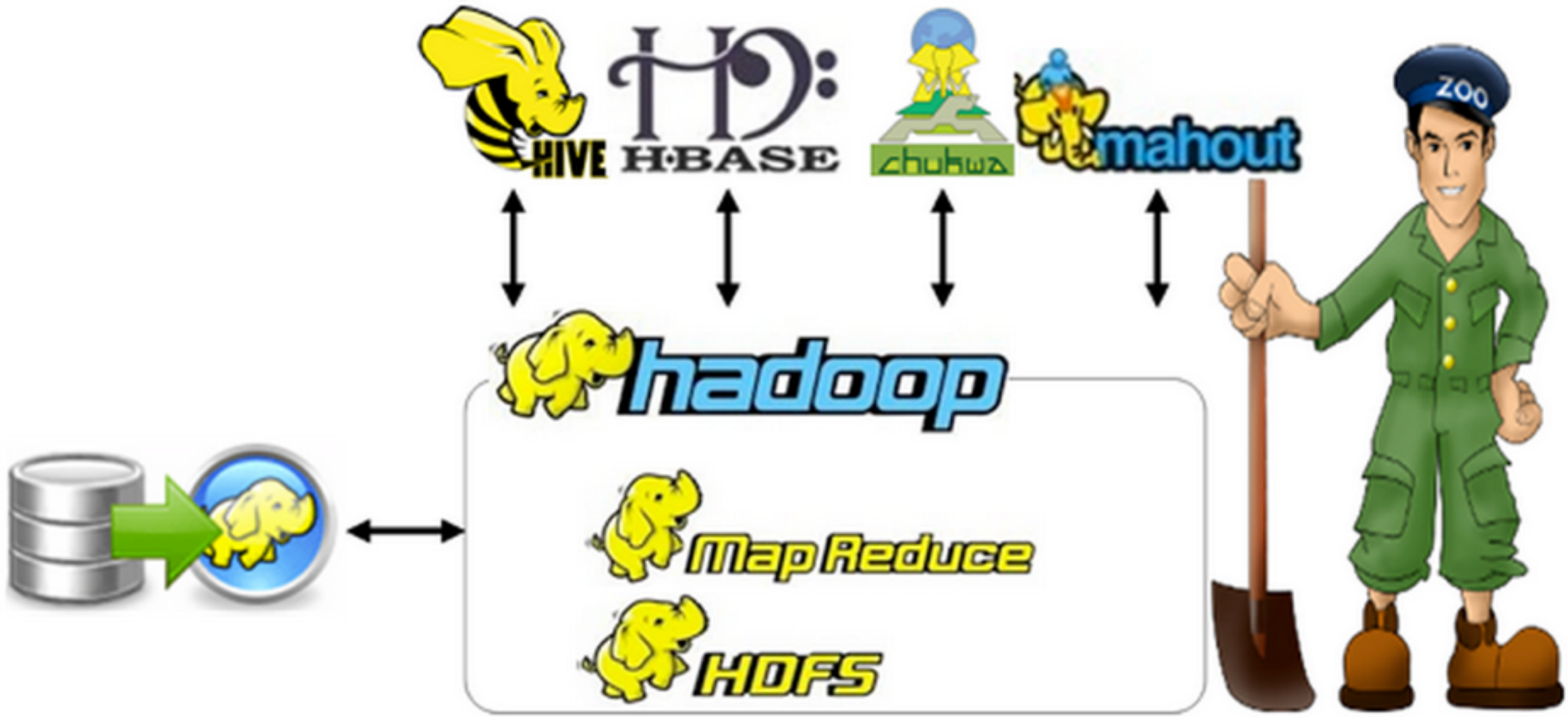
DM/Stats/ML

Lambda Architecture

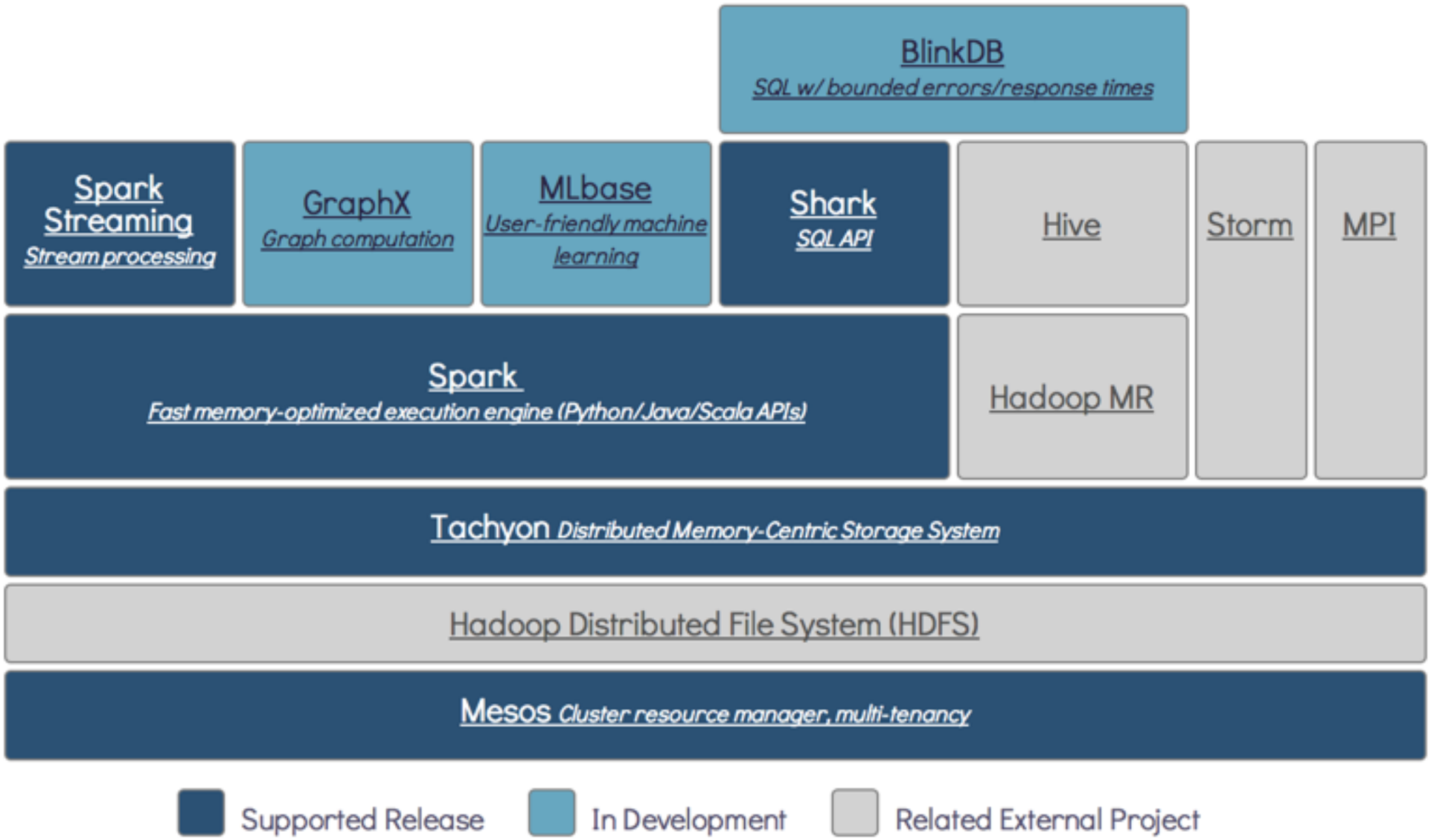


Overview of the Lambda Architecture

Base Hadoop Stack



Berkeley Data Science Stack



Where is Python?

PySpark

Hadoop Streaming

Pig

Anaconda Cluster

Bringing the Python ecosystem to Hadoop and Spark

“ At my company X, we have peta/terabytes of data, just lying around, waiting for someone to explore it”

- someone at PyTexas

Let’s make it easier for users to explore and extract useful insights out of data.

Wakari	Share and deploy
Bokeh	Interactive data visualizations
Blaze	Scale
Numba	Power to speed up
Conda	Package manager
Anaconda	Free enterprise-ready Python distribution

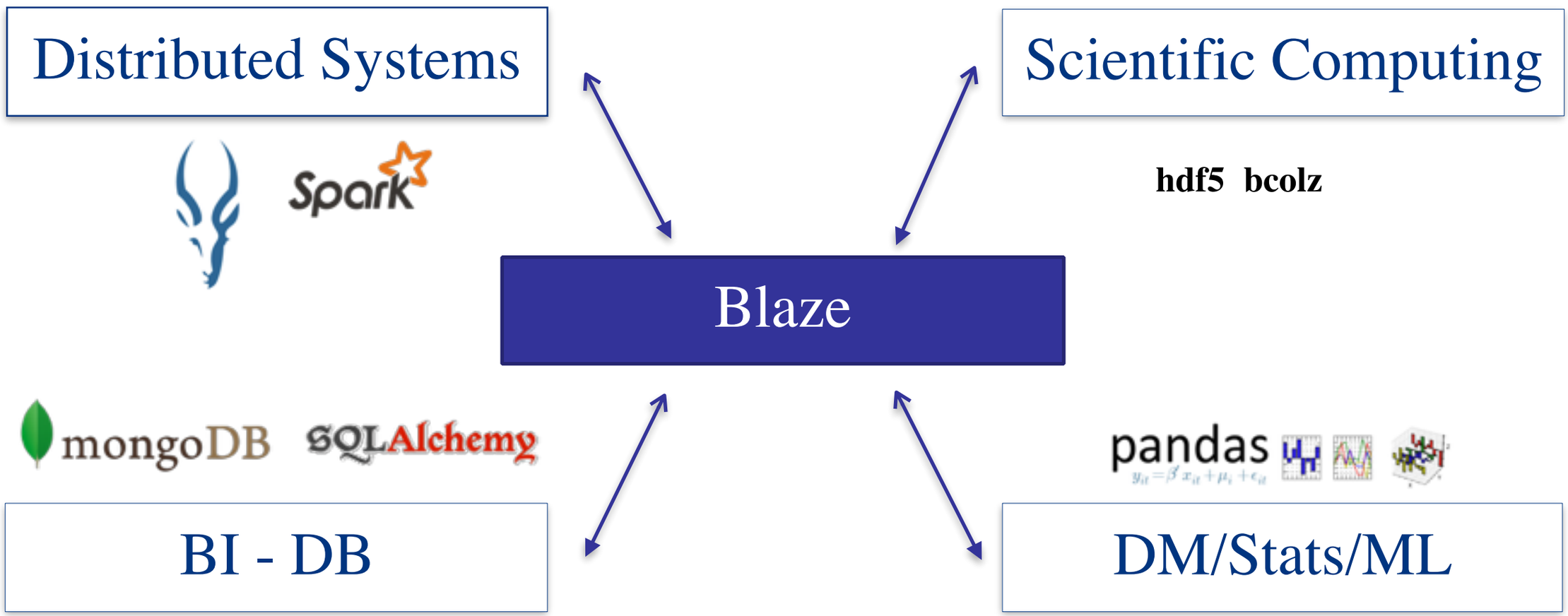
Blaze

Data Pain

- Dealing with data applications has numerous pain points
  - Hundreds of data formats
  - Basic programs expect all data to fit in memory
  - Data analysis pipelines constantly changing from one form to another
  - Sharing analysis contains significant overhead to configure systems
  - Parallelizing analysis requires expert in particular distributed computing stack

Blaze

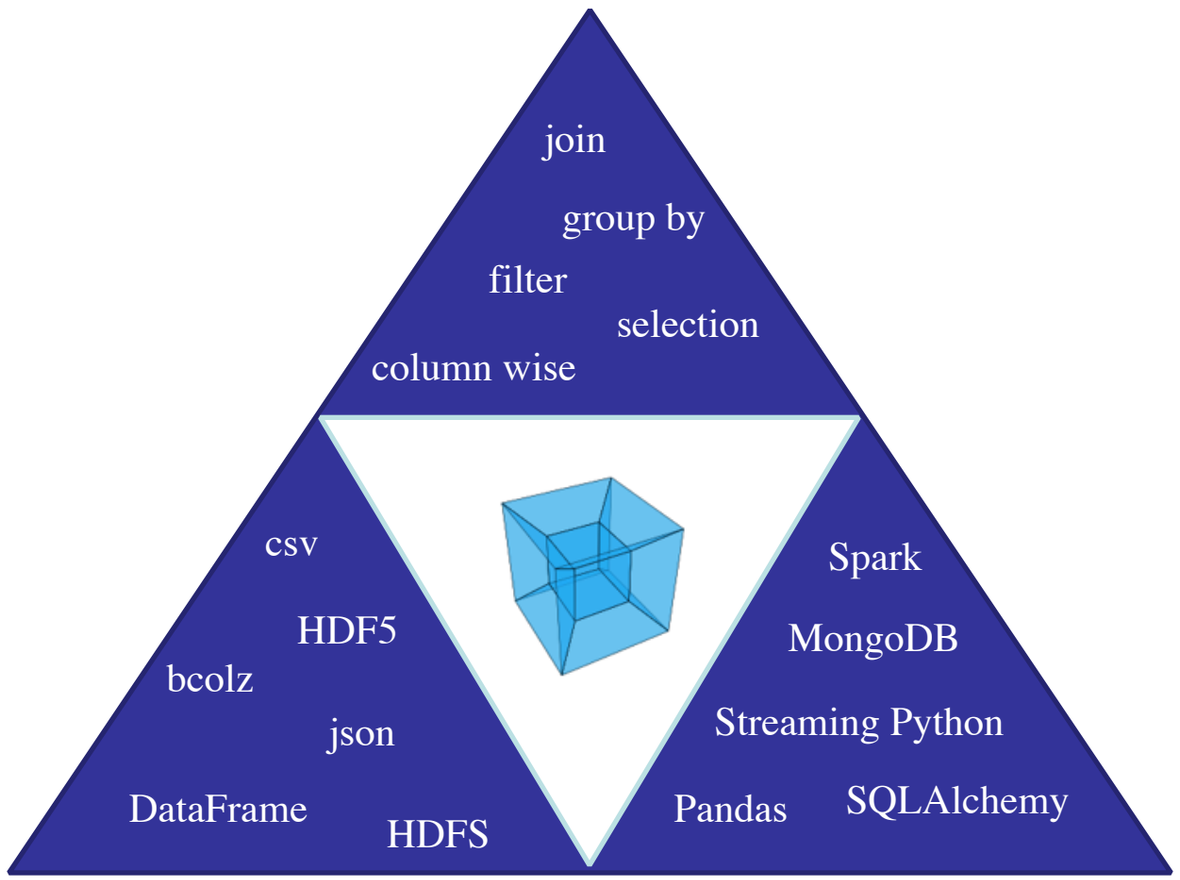
Blaze



Connecting technologies to users
Connecting technologies to each other

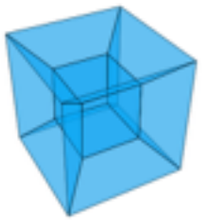
Blaze

Abstract expressions



Data Storage

Computational backend



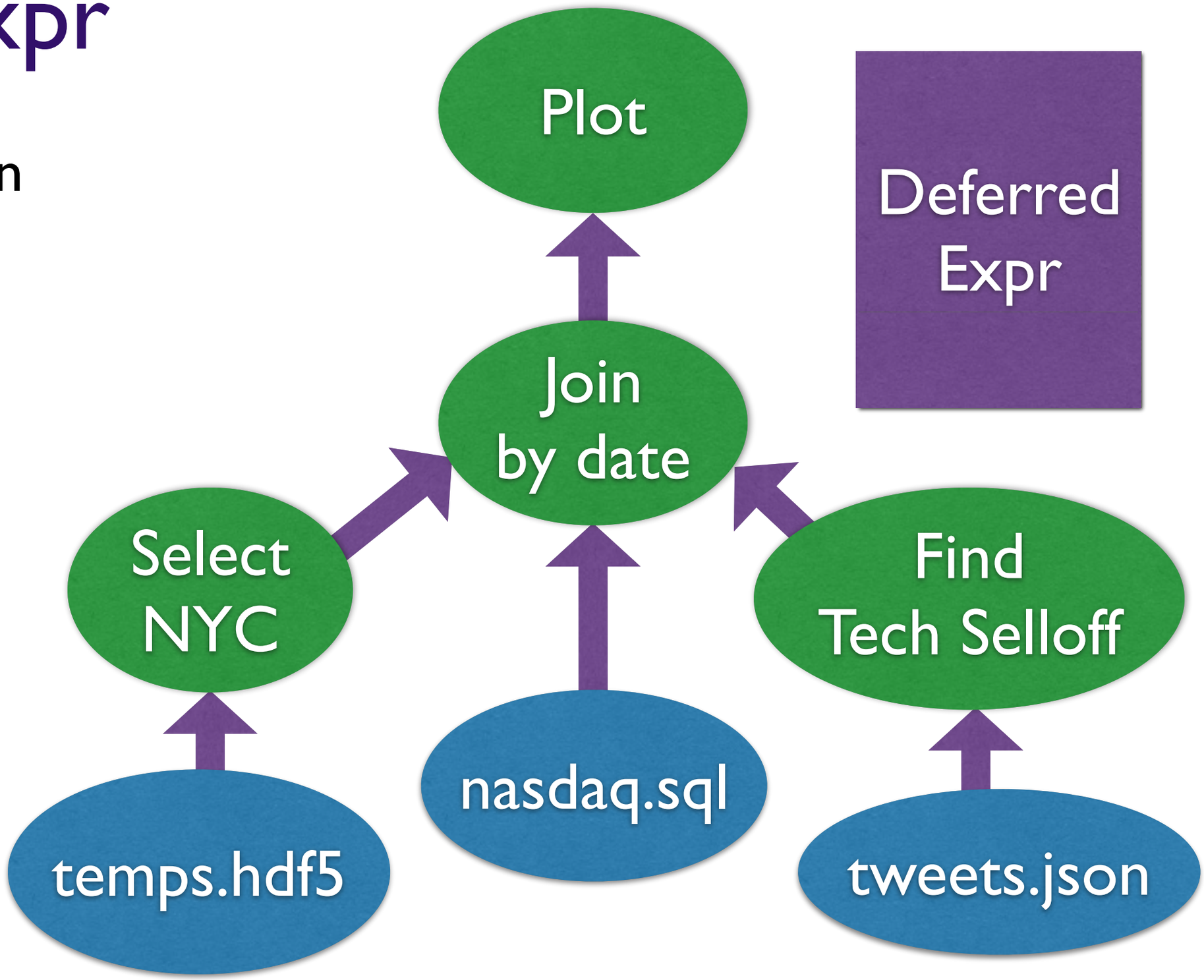
Blaze Architecture

- Flexible architecture to accommodate exploration
- Use compilation of deferred expressions to optimize data interactions

Blaze Expr

- Lazy computation to minimize data movement

- Simple DAG for compilation to
  - parallel application
  - distributed memory
  - static optimizations

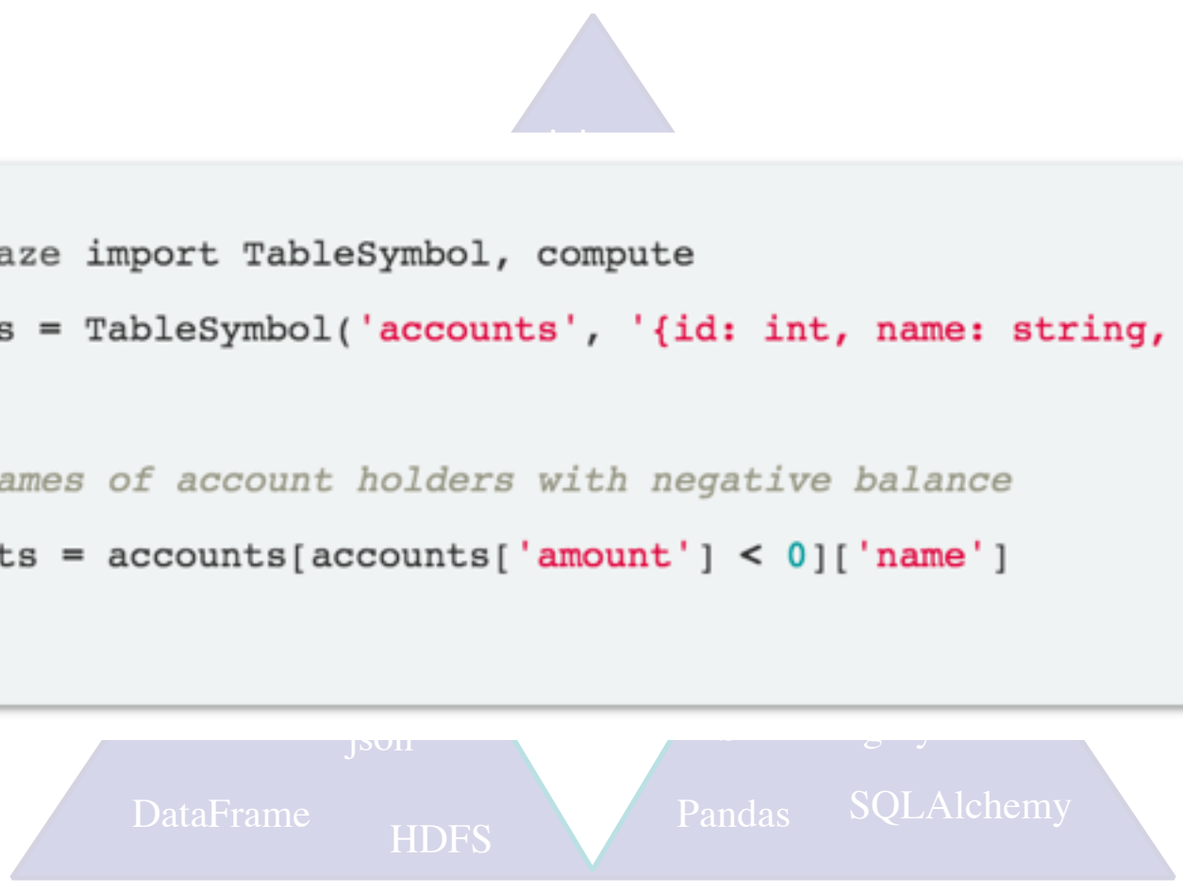


Blaze.expressions

Abstract expressions

```python
from blaze import TableSymbol, compute
accounts = TableSymbol('accounts', '{id: int, name: string, amount: int}')

# The names of account holders with negative balance
deadbeats = accounts[accounts['amount'] < 0]['name']
```

Python

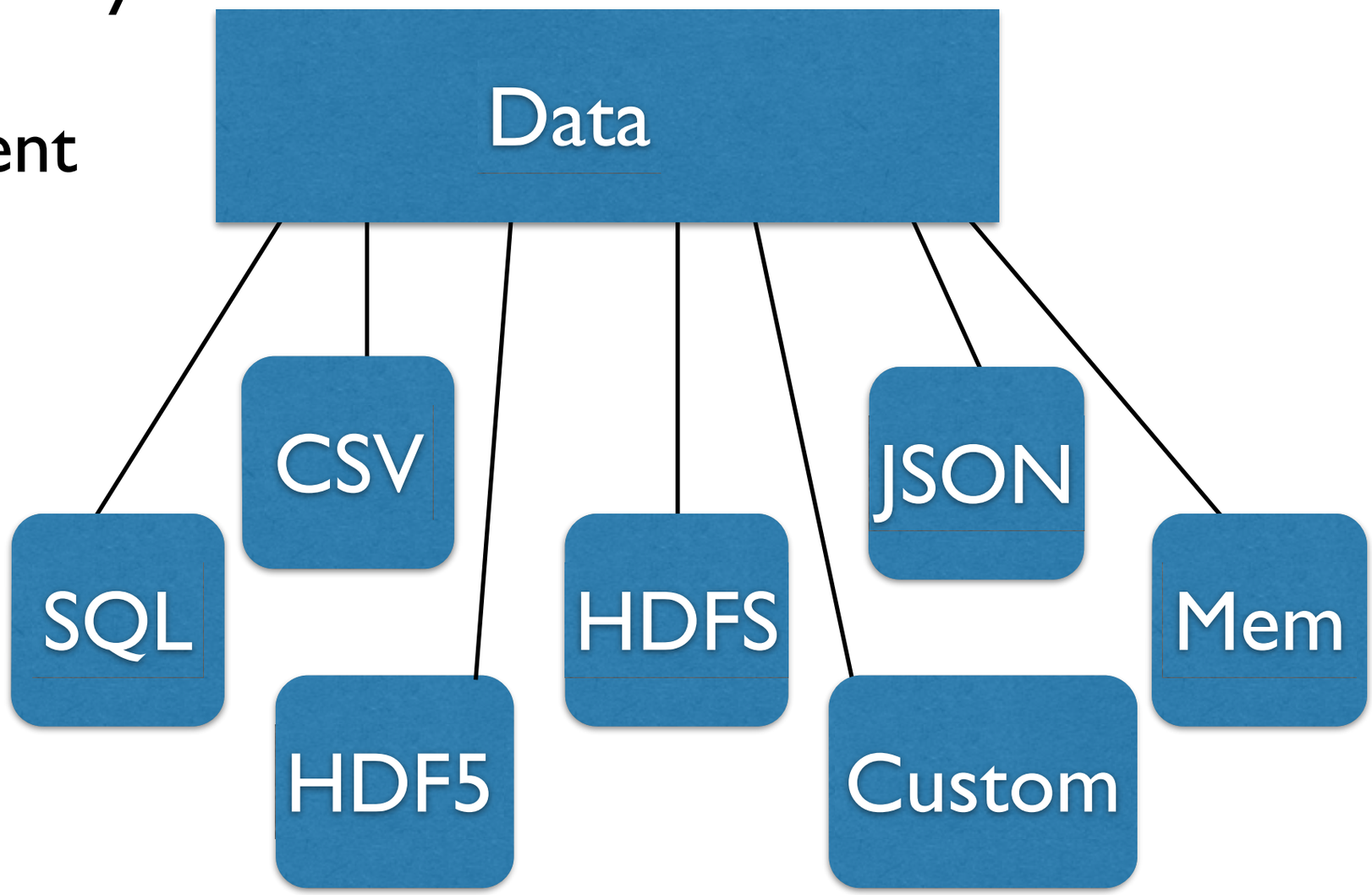


Data Storage

Computational backend

Blaze Data

- Single interface for data layers
- Composition of different formats
- Simple api to add custom data formats



Blaze.data

Abstract expressions

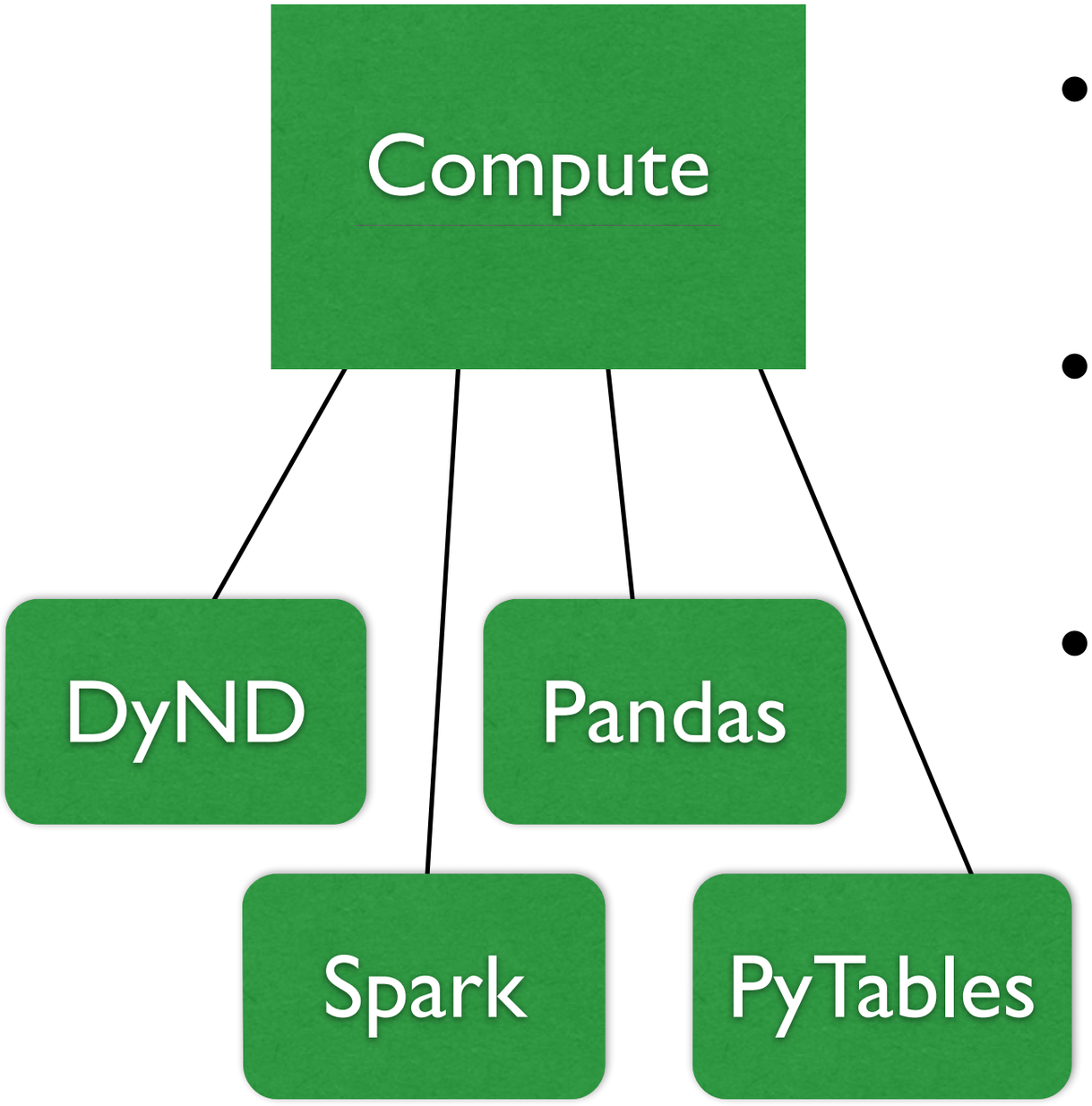
```
import pymongo

db = pymongo.MongoClient().db

db.mycollection.insert([{'id': 1, 'name': 'Alice',   'amount':  100},
                        {'id': 2, 'name': 'Bob',     'amount': -200},
                        {'id': 3, 'name': 'Charlie', 'amount':  300},
                        {'id': 4, 'name': 'Dennis',  'amount':  400},
                        {'id': 5, 'name': 'Edith',   'amount': -500}])
```

DataFrame HDFS Pandas SQLAlchemy

Data Storage

Computational backend

Blaze Compute



- Computation abstraction over numerous data libraries
- Simple multi-dispatched visitors to implement new backends
- Allows plumbing between stacks to be seamless to user

Blaze.compute

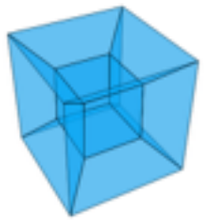
Abstract expressions

```
>>> list(compute(deadbeats, L))          # Python
['Bob', 'Edith']

>>> compute(deadbeats, df)               # Pandas
1       Bob
4     Edith
Name: name, dtype: object

>>> compute(deadbeats, db.mycollection)  # MongoDB
[u'Bob', u'Edith']
```

Python

Data Storage

Computational backend



Blaze Example - Counting Weblinks

Common Blaze Code

```
# Expr
t_idx = TableSymbol('{name: string,
                      node_id: int32}')
t_arc = TableSymbol('{node_out: int32,
                      node_id: int32}')
joined = Join(t_arc, t_idx, "node_id")
t = By(joined, joined['name'],
       joined['node_id'].count())

# Data Load
idx, arc = load_data()

# Computations
ans = compute(t, {t_arc: arc, t_idx: idx})
in_deg = dict(ans)
in_deg[u'blogspot.com']
```

Blaze Example - Counting Weblinks

load_data

Using Spark + HDFS

```
sc = SparkContext("local", "Simple App")
idx = sc.textFile(“hdfs://master.continuum.io/example_index.txt”)
idx = idx.map(lambda x: x.split(‘\t’))\
         .map(lambda x: [x[0], int(x[1])])
arc = sc.textFile("hdfs://master.continuum.io/example_arcs.txt")
arc = arc.map(lambda x: x.split(‘\t’))\
         .map(lambda x: [int(x[0]), int(x[1])])
```

Using Pandas + Local Disc

```
with open("example_index.txt") as f:
    idx = [ ln.strip().split('\t') for ln in f.readlines()]
idx = DataFrame(idx, columns=['name', 'node_id'])

with open("example_arcs.txt") as f:
    arc = [ ln.strip().split('\t') for ln in f.readlines()]
arc = DataFrame(arc, columns=['node_out', 'node_id'])
```

Blaze.API

Table

Using the interactive Table object we can interact with a variety of computational backends with the familiarity of a local DataFrame

```
>>> from blaze import Table
>>> t = Table(db.mycollection)
>>> t
   amount  id     name
0     100   1    Alice
1    -200   2      Bob
2     300   3  Charlie
3     400   4   Dennis
4    -500   5    Edith

>>> t[t.amount < 0]
   amount  id   name
0    -200   2    Bob
1    -500   5  Edith
```

Blaze.API

Table

```
>>> from blaze import *
>>> iris = CSV('examples/data/iris.csv')

>>> t = Table(iris)
```

```
>>> from blaze import *
>>> iris = SQL('sqlite:///examples/data/iris.db', 'iris')

>>> t = Table(iris)
```

```
>>> import pyspark
>>> sc = pyspark.SparkContext("local", "blaze-demo")
>>> rdd = into(sc, csv)  # handle data conversion
>>> t = Table(rdd)
```

Blaze.API

Migrations - into

```
>>> import pymongo
>>> db = pymongo.MongoClient().db
>>> into(db.iris, df)                    # Migrate from Pandas DataFrame to Mong
Collection(Database(MongoClient('localhost', 27017), u'db'), u'iris')
```

Python

Python

Blaze notebooks

Why I like using Blaze?

- Syntax is very similar to Pandas
- Easy to scale
- Easy to find best computational backend to a particular dataset
- Easy to adapt my code if someone handles me a dataset in a different format/ backend

Want to learn more about Blaze?

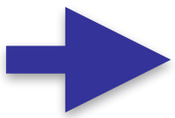
Free Webinar:
http://www.continuum.io/webinars/getting-started-with-blaze

Blogpost:
http://continuum.io/blog/blaze-expressions
http://continuum.io/blog/blaze-migrations
http://continuum.io/blog/blaze-hmda

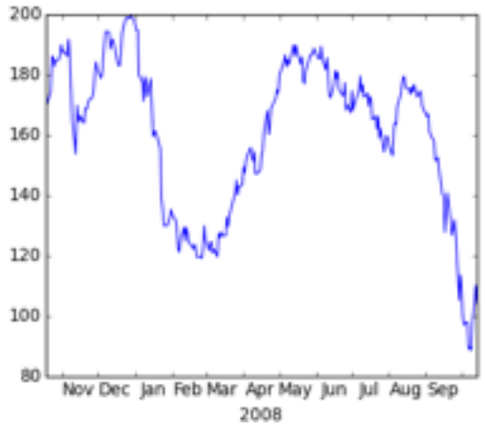
Docs and source code:
http://blaze.pydata.org/
https://github.com/ContinuumIO/blaze

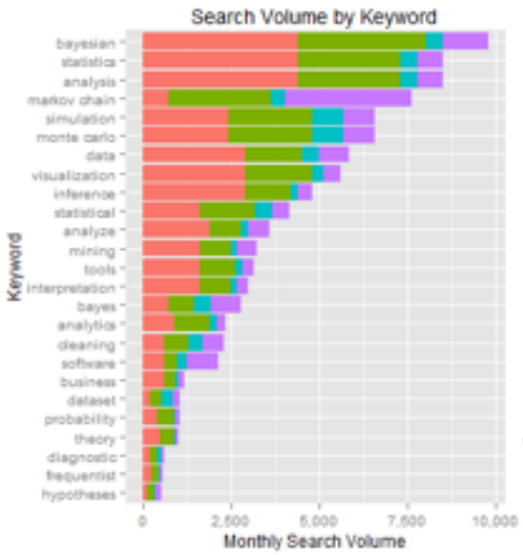
Data visualization - An Overview

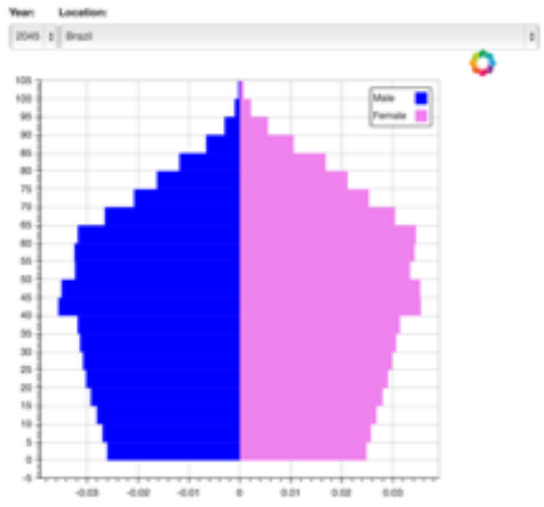
Results presentation
Static
Small datasets
Traditional plots

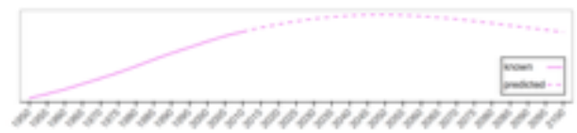


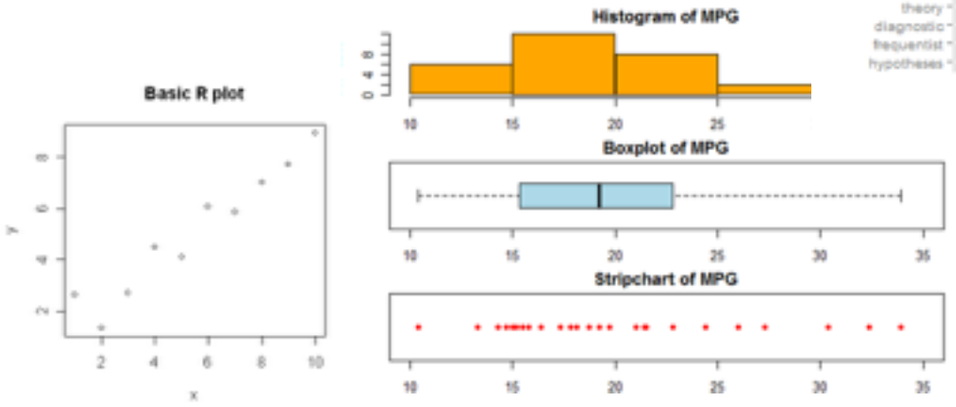
Visual analytics
Interactive
Large datasets
Novel graphics











CONTINUUM ANALYTICS

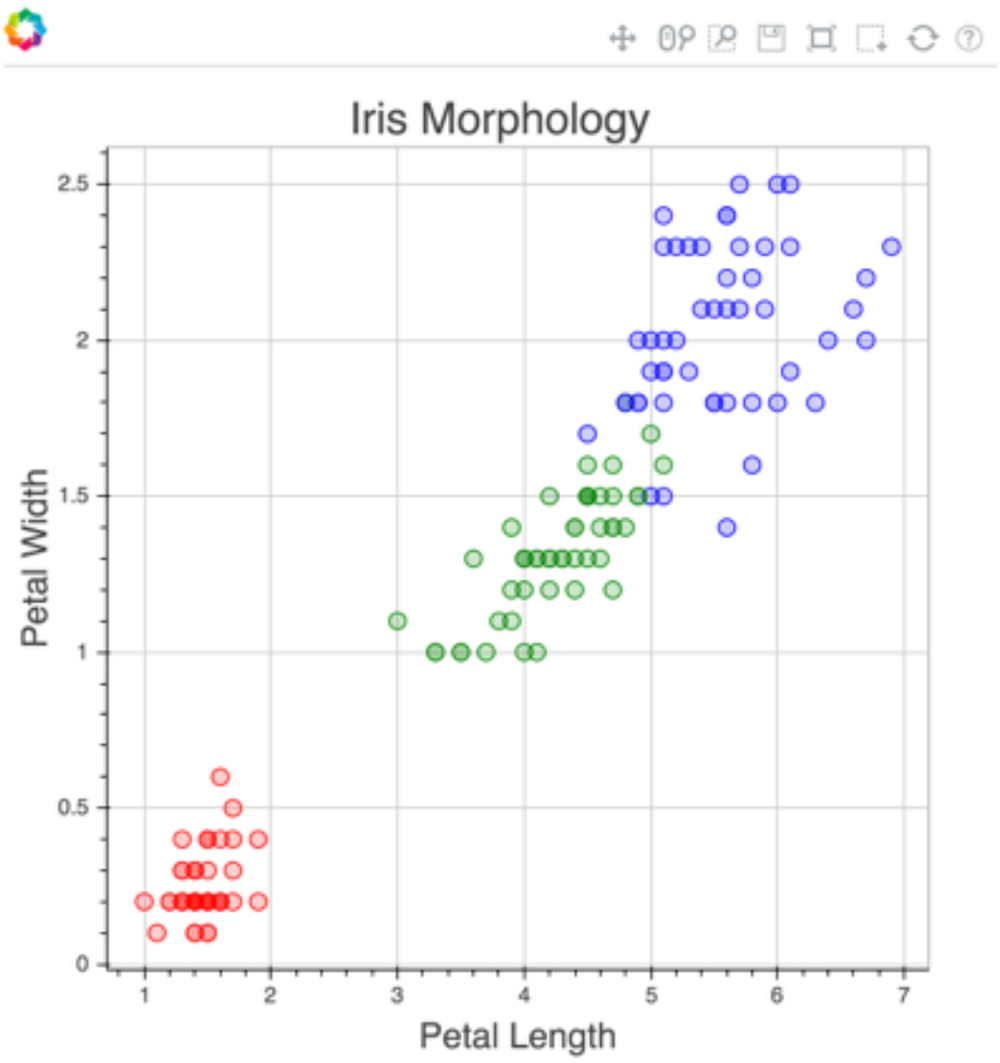
Bokeh

- Interactive visualization
- Novel graphics
- Streaming, dynamic, large data
- For the browser, with or without a server
- Matplotlib compatibility
- No need to write Javascript

http://bokeh.pydata.org/
https://github.com/ContinuumIO/bokeh

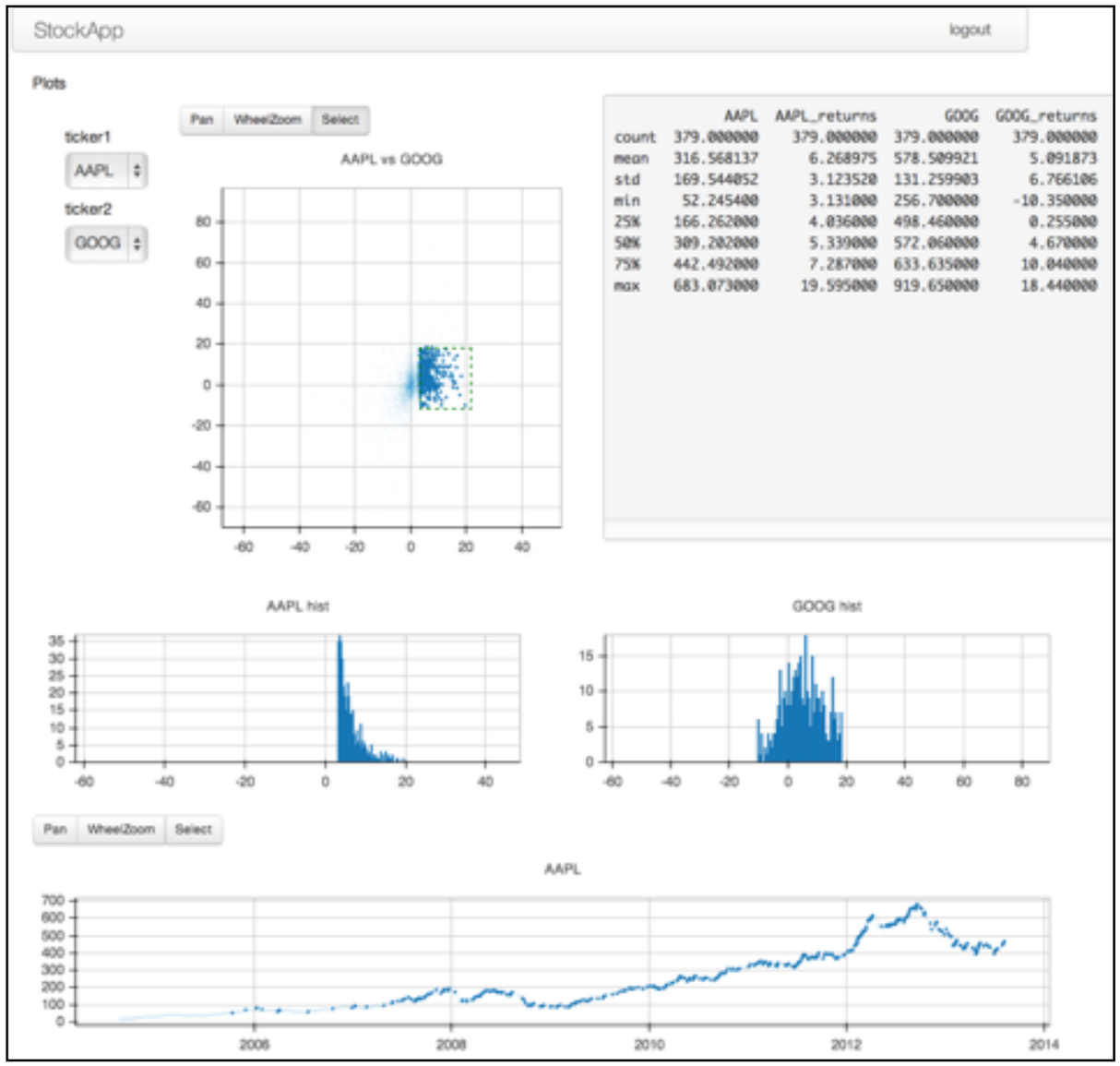
Bokeh - Interactive, Visual analytics

- Tools (e.g. Pan, Wheel Zoom, Save, Resize, Select, Reset View)

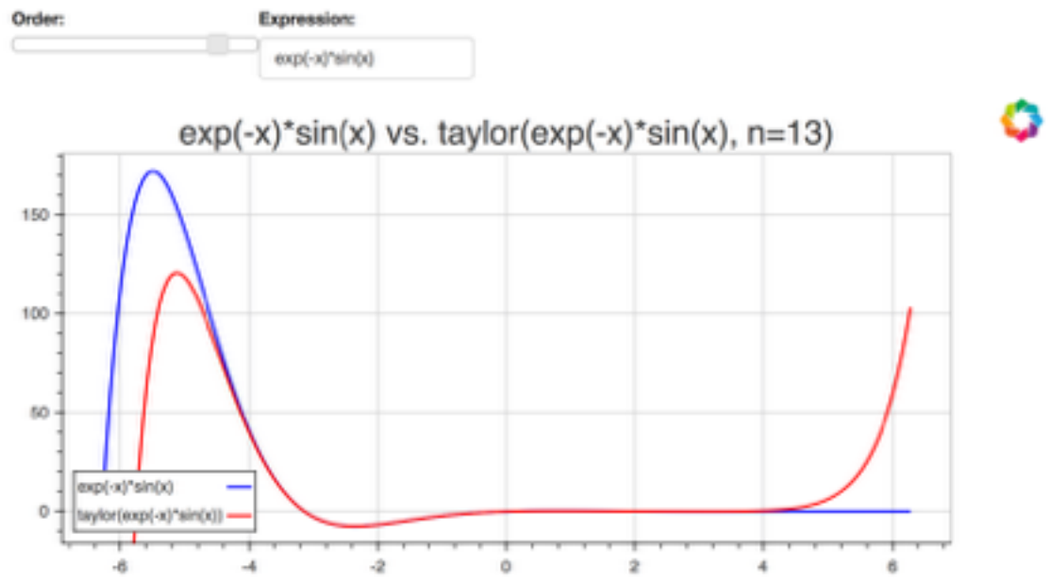


Bokeh - Interactive, Visual analytics

- Widgets and dashboards

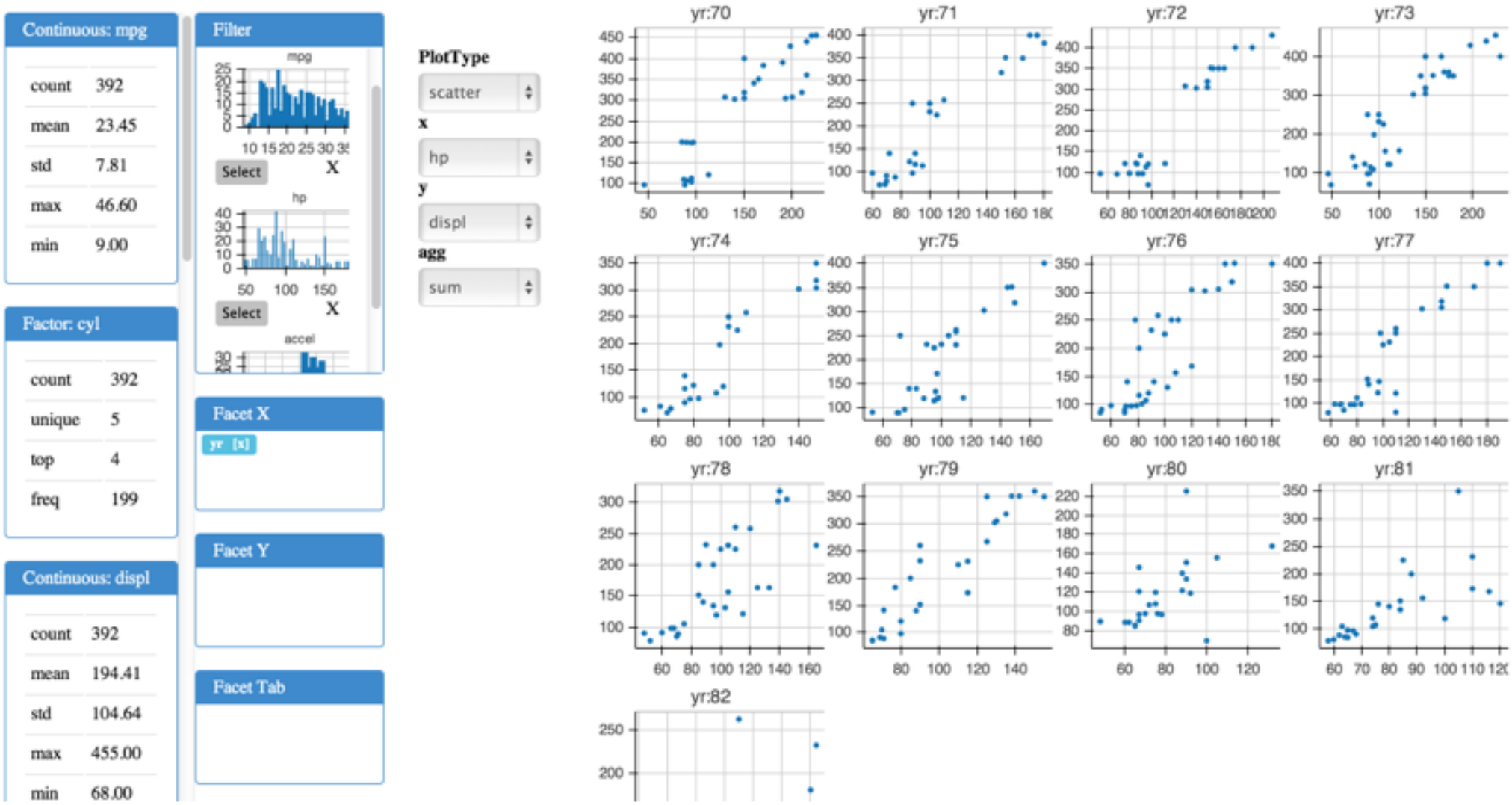


	AAPL	AAPL_returns	GOOG	GOOG_returns
count	379.000000	379.000000	379.000000	379.000000
mean	316.568137	6.268975	578.509921	5.091873
std	169.544052	3.123520	131.259903	6.766106
min	52.245400	3.131000	256.700000	-10.350000
25%	166.262000	4.036000	498.460000	0.255000
50%	309.202000	5.339000	572.060000	4.670000
75%	442.492000	7.287000	633.635000	10.040000
max	683.073000	19.595000	919.650000	18.440000



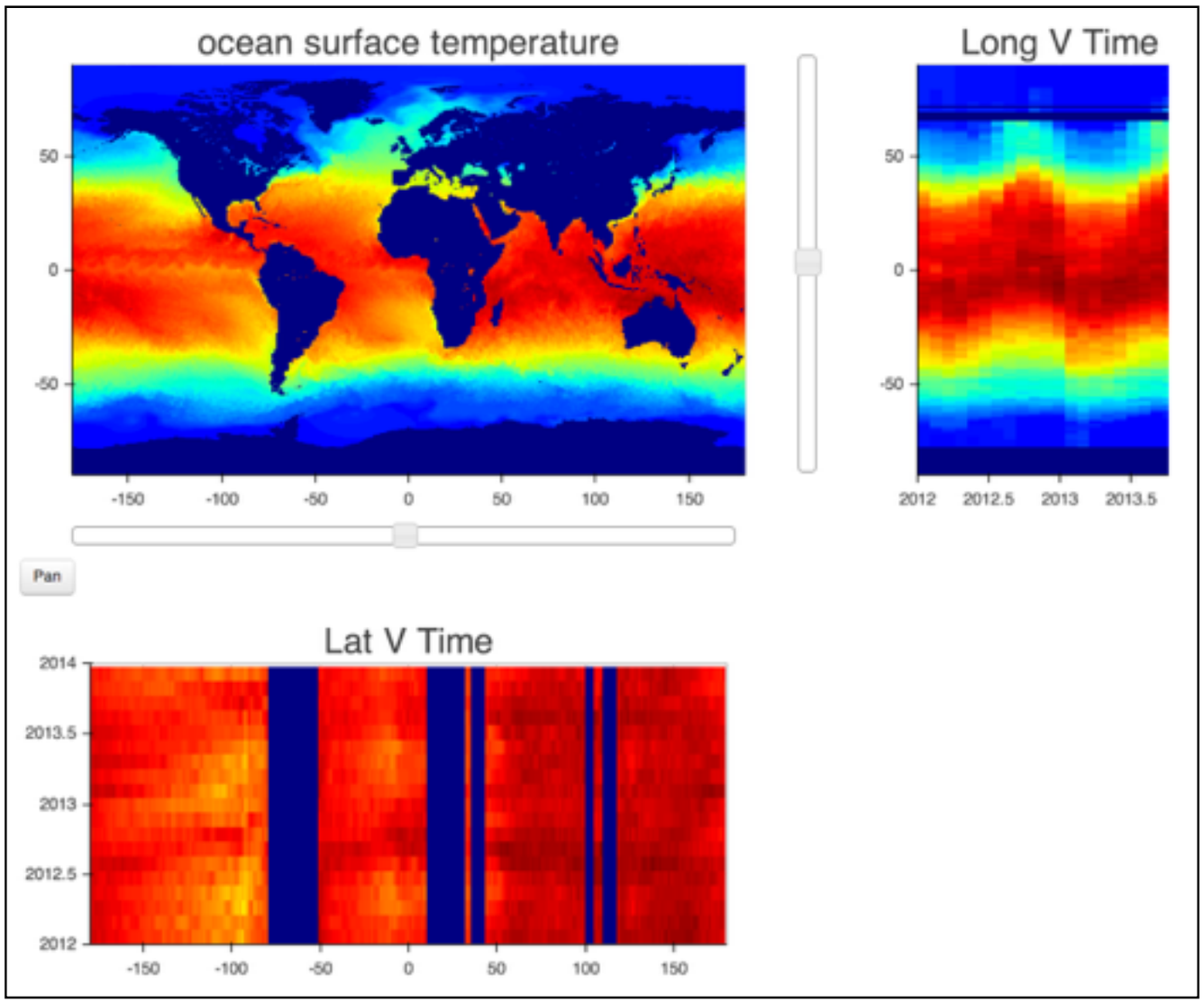
Bokeh - Interactive, Visual analytics

- Crossfilter

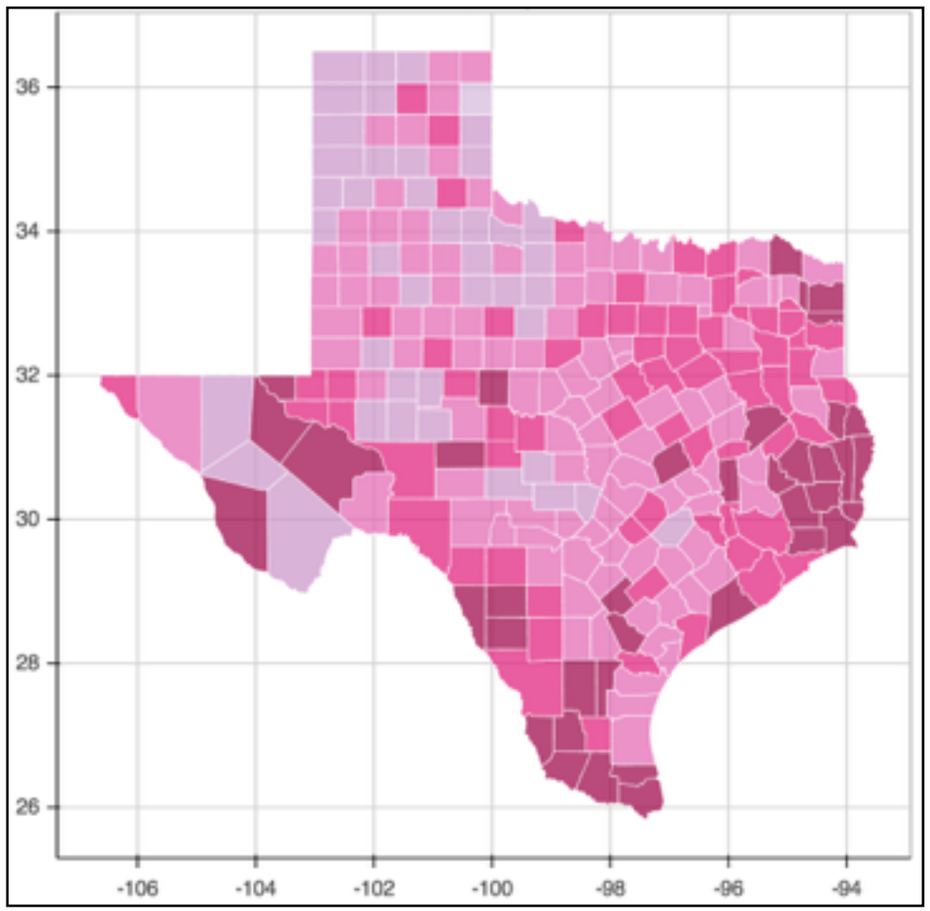


Bokeh - Large datasets

Server-side downsampling and abstract rendering



Bokeh - No JavaScript




```
output_file("texas.html", title="texas.py example")

patches(county_xs, county_ys, fill_color=county_colors, fill_alpha=0.7,
        line_color="white", line_width=0.5, title="Texas Unemployment 2009")

show()
```

Thank you! :)