

# Performance Pandas

Jeff Reback

*@jreback*

StrataNYC 2015

September 29, 2015

<https://github.com/jreback/StrataNYC2015/performance>

# Jeff Reback

*@jreback*

- former quant
- currently working on projects at Continuum
- core committer to pandas for last 3 years
- manage pandas since 2013

# What do we care about when writing code?

## Objectives

- feature set
- readability counts
- maintenance is a virtue
- tests & docs

# What do we care about when writing code?

## Objectives

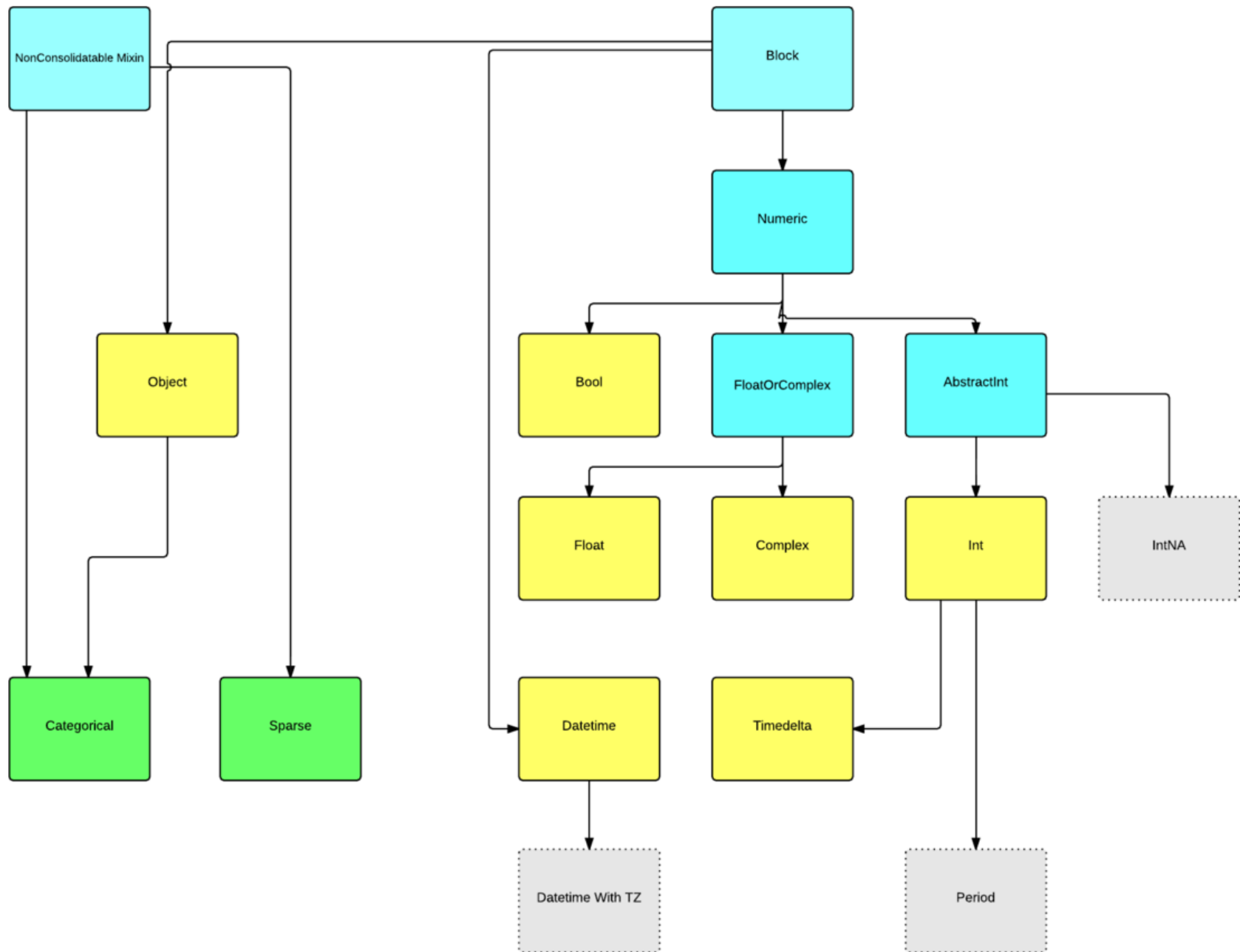
- feature set
- readability counts
- maintenance is a virtue
- tests & docs

## Constraints

- implementation time
- runtime
- resource utilization

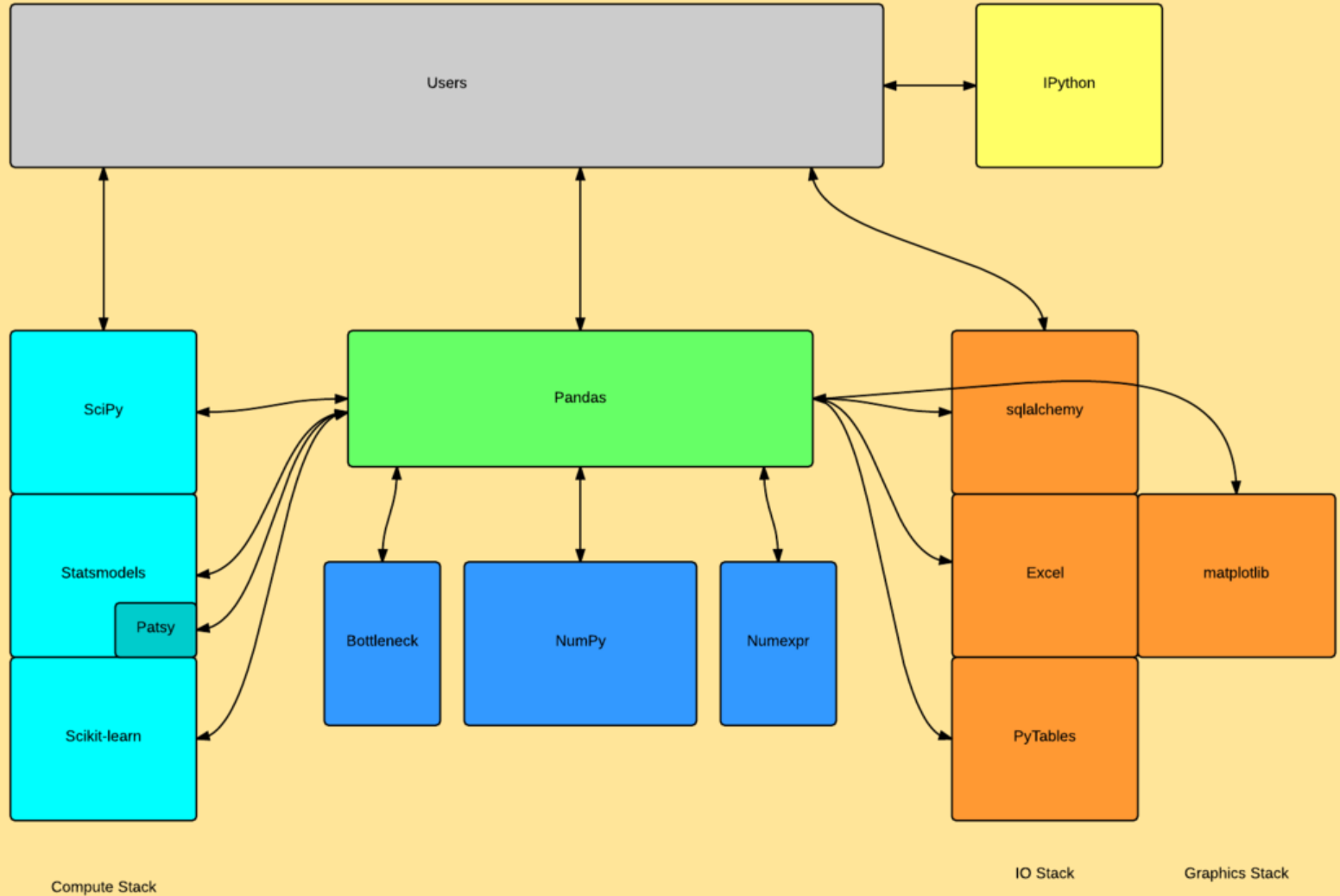
# What drives pandas?

- dtype segregation
- block memory layout



# What drives pandas?

- dtype segregation
- block memory layout
- **computation backends**





# What drives pandas?

- dtype segregation
- block memory layout
- computation backends
- **cython for critical parts**
- **hashtable for indexing**

# how to make pandas perform

1. Have Correct Code
2. Profile / Compare

# how to make pandas perform

1. Have Correct Code
2. Profile / Compare
3. **Refer to Rules #1 and #2**

**I DON'T ALWAYS COMPARE  
THINGS**



**BUT WHEN I DO,  
IT'S APPLES TO  
ORANGES**

*// Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered.*

*// premature optimization is the root of all evil (or at least most of it) in programming.*

# How to make pandas *fast*

- **algo**
- idioms
- built-in / vectorization
  - pandas/numpy
  - bottleneck/numexpr
  - cython
- ad-hoc cython/numba

# How to make pandas *fast*

- algo
- **idioms**
- built-in / vectorization
  - pandas/numpy
  - bottleneck/numexpr
  - cython
- ad-hoc cython/numba

# How to make pandas *fast*

- algo
- idioms
- **built-in / vectorization**
  - pandas/numpy
  - bottleneck/numexpr
  - cython
- **ad-hoc cython/numba**



# How to make pandas ~~fast~~

**slow**

- **apply across the rows**

# dealing with apply if you're not a pandas expert

*“ look for a way to vectorize it*

*“ even if you are, look for another way*

# How to make pandas ~~fast~~

**slow**

- apply across the rows
- **itertuples/iterrows**

# How to make pandas ~~fast~~

**slow**

- apply across the rows
- itertuples/iterrows
- **iterative updating**

**.values, a double edged sword**



# Do's

- have the correct dtypes
- ***pd.concat***
- ***Categoricals***
- Use idioms & builtin
- ***.apply*** across columns

# Don'ts

- repeated insertions
- micro optimize
- use loops / re-invent the wheel
- ***.apply*** across rows
- ***.applymap***
- nest ***groupby.apply()***
- ***inplace=True***

# Memory Considerations

- conversions



# Memory Considerations

- conversions
- **categoricals**

# Memory Considerations

- conversions
- categoricals
- **iterators**

# I/O & Serialization

- HDF5
- bcolz
- CSV
- SQL
- JSON
- pickle
- msgpack

<http://matthewrocklin.com/blog/work/2015/03/16/Fast-Serialization/>

<http://odo.readthedocs.org/en/latest/>

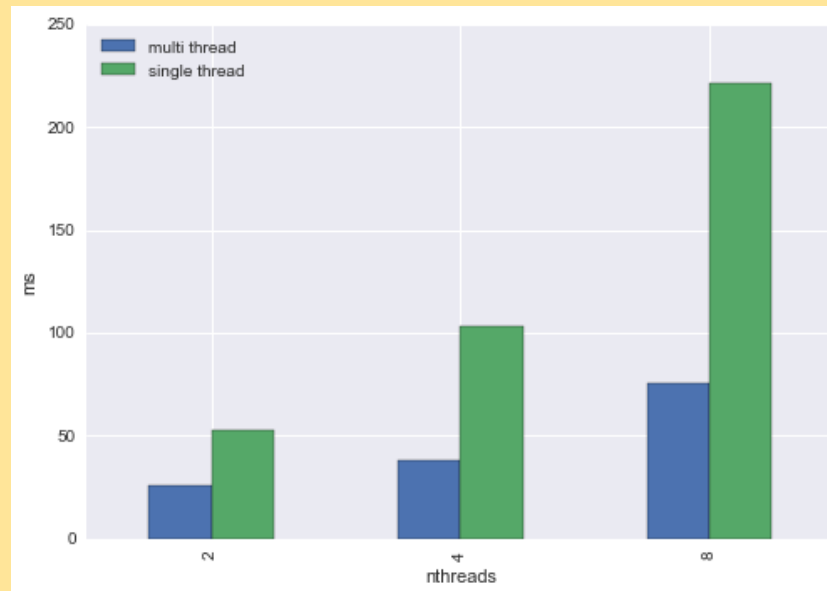
# I need even more!

- Global-Interpreter-Lock (GIL)
  - <http://continuum.io/blog/pandas-releasing-the-gil>

# I need even more!

- Global-Interpreter-Lock (GIL)

■ <http://continuum.io/blog/pandas-releasing-the-gil>



# I need even more!

- Global-Interpreter-Lock (GIL)
  - <http://continuum.io/blog/pandas-releasing-the-gil>
- **out-of-core**

# I need even more!

- Global-Interpreter-Lock (GIL)
  - <http://continuum.io/blog/pandas-releasing-the-gil>
- out-of-core
- **dask**
  - threading
  - multi-process
  - distributed

<https://dask.readthedocs.org/en/latest/>

# How to contribute

<https://github.com/pydata/pandas/issues>

# This Talk

<https://github.com/jreback/StrataNYC2015/performance>

*@jreback*