Introduction to IPCluster

Sponsored by AWS

Continuum Analytics

Unofficial Python Slogan

“ CPU time is cheap, ***my*** *time is expensive ”*

Some Truthy Statements

- We prefer an expressive language over raw speed

Some Truthy Statements

- We prefer an expressive language over raw speed
  - 97% of the time, it's fast enough every time
  - Premature optimization is the root of all evil

Some Truthy Statements

- We prefer an expressive language over raw speed
  - 97% of the time, it's fast enough every time
  - Premature optimization is the root of all evil

- But scientific workflows can get big

Some Truthy Statements

- We prefer an expressive language over raw speed
  - 97% of the time, it's fast enough every time
  - Premature optimization is the root of all evil

- But scientific workflows can get big
  - My expensive time spent waiting instead of doing science
  - Let's replace that with more cheap CPU time

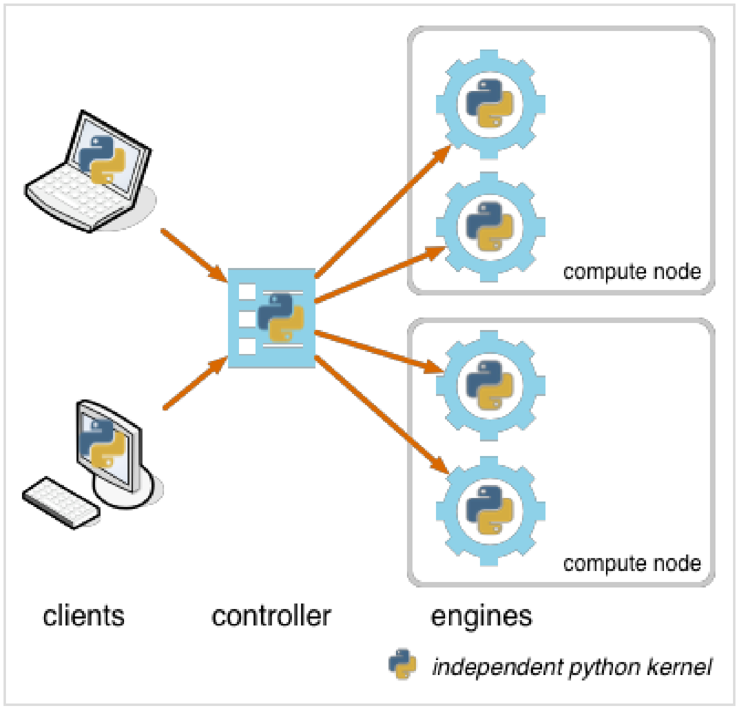
Some Truthy Statements

- We have horsepower on-demand (AWS, Azure, Heroku, University Clusters)

Some Truthy Statements

- We need a nice way to wrangle the data/code between nodes

IPCluster Overview



The Basic Use Case

- An embarassingly (pleasingly) parallel workflow
    - Little/no communication between jobs
    - Parameter scans, multiple BLAST queries, genetic algorithms, etc.

The Basic Use Case

- An embarassingly (pleasingly) parallel workflow
  - Little/no communication between jobs
  - Parameter scans, multiple BLAST queries, genetic algorithms, etc.
- Can utilize the CPU/GPU/RAM of many nodes simultaneously

The Basic Use Case

- An embarassingly (pleasingly) parallel workflow
    - Little/no communication between jobs
    - Parameter scans, multiple BLAST queries, genetic algorithms, etc.
- Can utilize the CPU/GPU/RAM of many nodes simultaneously
- Moves relatively small amounts of data
    - Aggregates/subsets are good enough
    - The data is already on the nodes

Parallel-o World

Create a client object

```
from IPython.parallel import Client

client = Client()
```

Parallel-o World

Create a client object

```
from IPython.parallel import Client

client = Client()
```

Construct a view

```
balanced = client.load_balanced_view()
# or
direct   = client[:]
```

Parallel-o World

Define a function

```
def hello():
    return "Hello World!"
```

Parallel-o World

Define a function

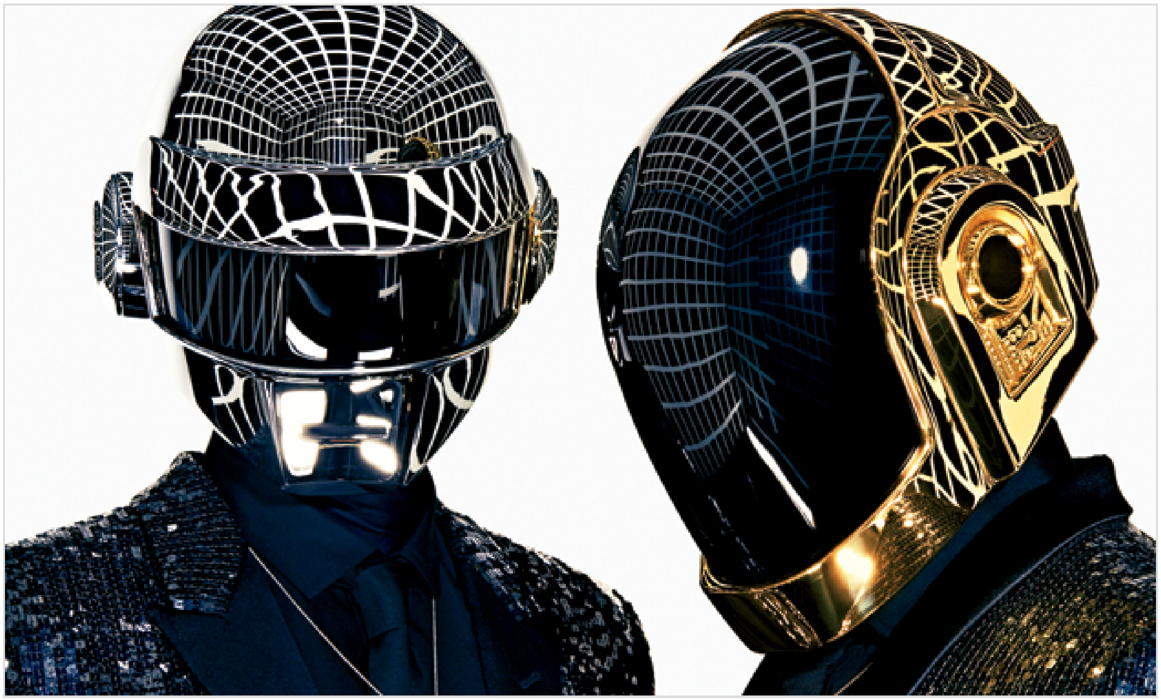
```
def hello():
    return "Hello World!"
```

Run it remotely

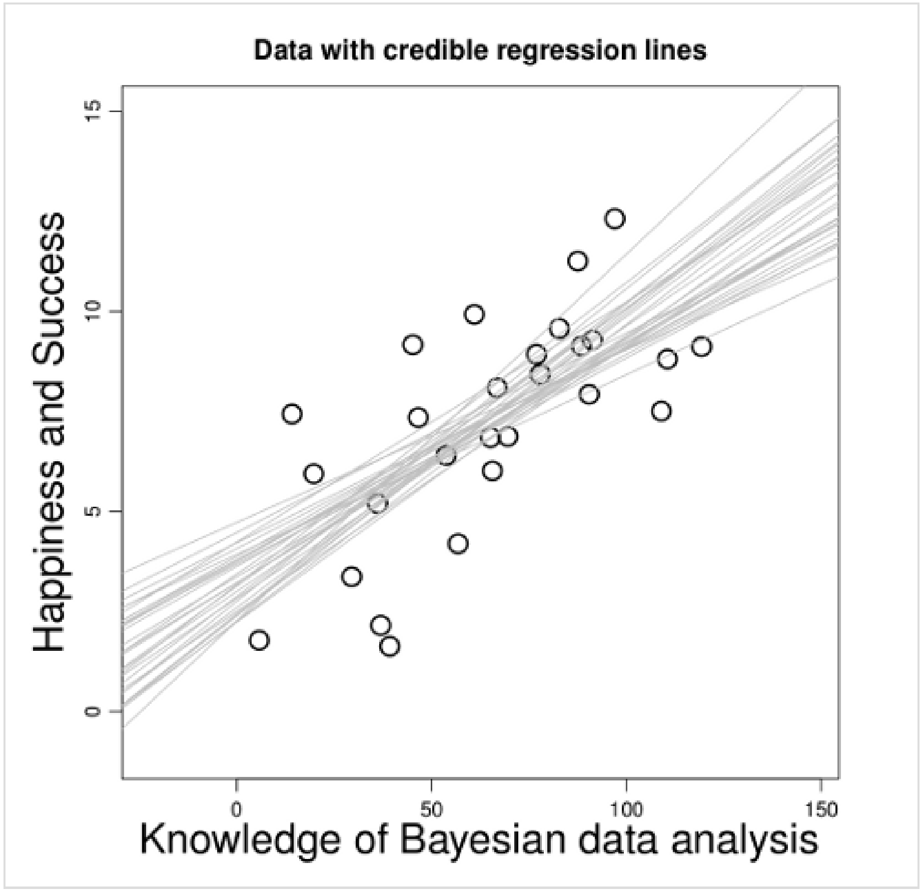
```
resp = direct.apply(hello)
resp.get()
```

```
Out[33]: ['Hello World!', 'Hello World!', 'Hello World!']
```

Easy Demo

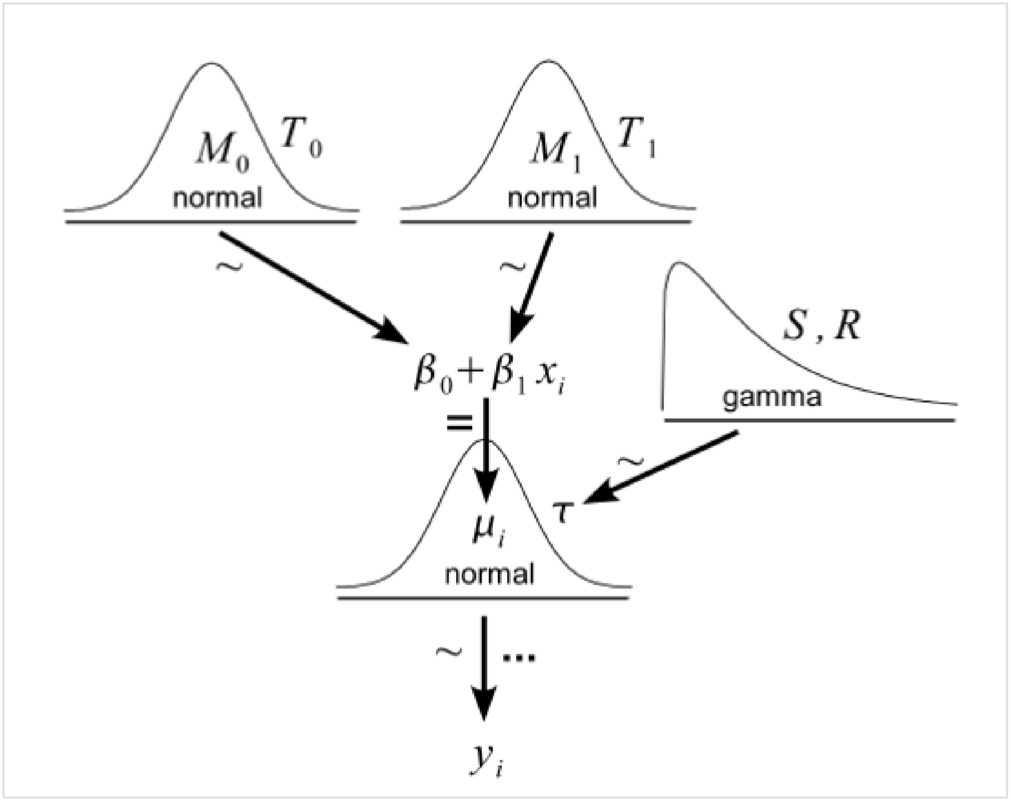


Use Case: Bayesian Estimation

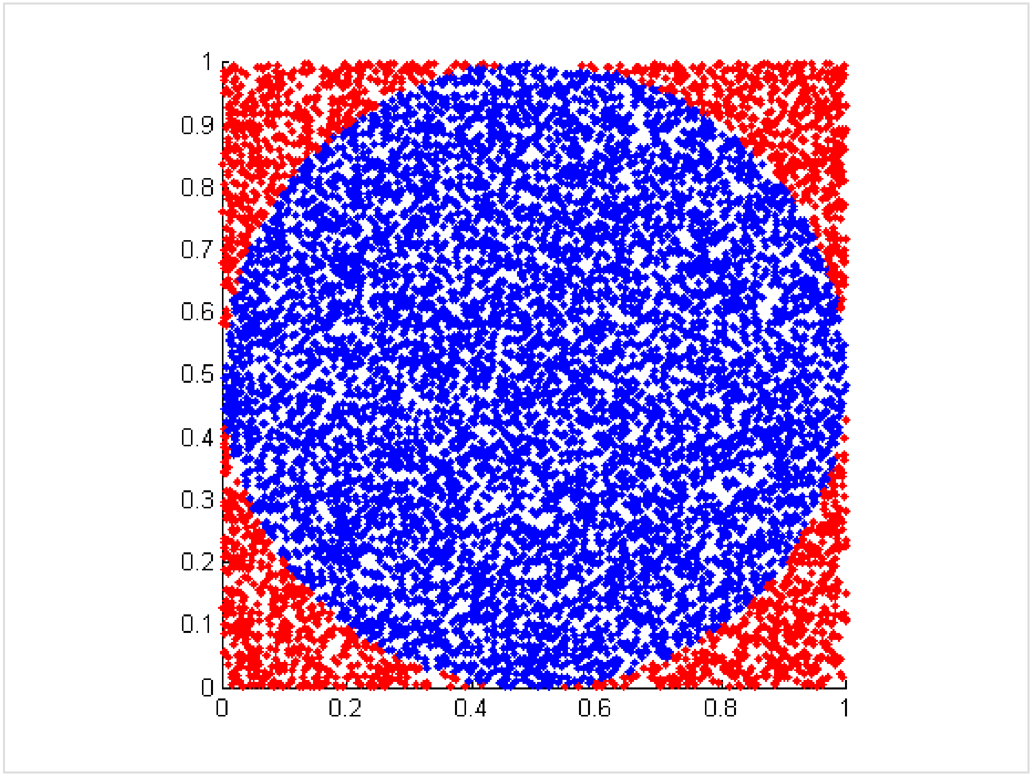


Images: John K Kruschke

Use Case: Bayesian Estimation



Use Case: Bayesian Estimation



Bayesian Demo