Parallel and Large Scale Learning with scikit-learn

Data Science London Meetup - Mar. 2013

Olivier Grisel

@ogrisel

Datageek, contributor to scikit-learn, works with Python / Java / Clojure / Pig, interested in Machine Learning, NLP, {Big|Linked|Open} Data and braaains!
Paris, France · http://github.com/ogrisel
About me

• Regular contributor to scikit-learn
• Interested in NLP, Computer Vision, Predictive Modeling & ML in general
• Interested in Cloud Tech and Scaling Stuff
• Starting my own ML consulting business: http://ogrisel.com
Outline

• The Problem and the Ecosystem
• Scaling Text Classification
• Scaling Forest Models
• Introduction to IPython.parallel & StarCluster
• Scaling Model Selection & Evaluation
Parts of the Ecosystem

♥♥♥♥ Single Machine with Multiple Cores

python™ multiprocessing

Joblib

♥♥♥♥ Multiple Machines with Multiple Cores

IP[y]: IPython Interactive Computing
disco massive data - minimal code

StarCluster hadoop
The Problem

**Big CPU** (Supercomputers - **MPI**)
Simulating stuff from models

**Big Data** (Google scale - **MapReduce**)
Counting stuff in logs / Indexing the Web

**Machine Learning?**
often somewhere in the middle
<table>
<thead>
<tr>
<th>Input Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels to Predict</td>
</tr>
</tbody>
</table>
Cross Validation

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cross Validation

Subset of the data used to train the model

Held-out test set for evaluation
Cross Validation

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>
Model Selection

the Hyperparameters hell

param_1 in [1, 10, 100]
param_2 in [1e3, 1e4, 1e5]

Find the best combination of parameters that maximizes the Cross Validated Score
Grid Search

<table>
<thead>
<tr>
<th>param_1</th>
<th>param_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1e3)</td>
<td>(1, 1e4)</td>
</tr>
<tr>
<td>(10, 1e3)</td>
<td>(10, 1e4)</td>
</tr>
<tr>
<td>(100, 1e3)</td>
<td>(100, 1e4)</td>
</tr>
<tr>
<td>(1, 1e5)</td>
<td>(1, 1e5)</td>
</tr>
<tr>
<td>(10, 1e5)</td>
<td>(10, 1e5)</td>
</tr>
<tr>
<td>(100, 1e5)</td>
<td>(100, 1e5)</td>
</tr>
<tr>
<td></td>
<td>(1, 1e3)</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>A B C</td>
</tr>
<tr>
<td></td>
<td>A B</td>
</tr>
<tr>
<td></td>
<td>A C C</td>
</tr>
<tr>
<td></td>
<td>A C</td>
</tr>
<tr>
<td></td>
<td>B C A</td>
</tr>
<tr>
<td></td>
<td>B C</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1, 1e4)</td>
</tr>
<tr>
<td></td>
<td>A B C</td>
</tr>
<tr>
<td></td>
<td>A B</td>
</tr>
<tr>
<td></td>
<td>A C C</td>
</tr>
<tr>
<td></td>
<td>A C</td>
</tr>
<tr>
<td></td>
<td>B C A</td>
</tr>
<tr>
<td></td>
<td>B C</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1, 1e5)</td>
</tr>
<tr>
<td></td>
<td>A B C</td>
</tr>
<tr>
<td></td>
<td>A B</td>
</tr>
<tr>
<td></td>
<td>A C C</td>
</tr>
<tr>
<td></td>
<td>A C</td>
</tr>
<tr>
<td></td>
<td>B C A</td>
</tr>
<tr>
<td></td>
<td>B C</td>
</tr>
</tbody>
</table>
Grid Search: Qualitative Results

gamma $10^{-1}$, C $10^0$
gamma $10^0$, C $10^0$
gamma $10^1$, C $10^0$
gamma $10^{-1}$, C $10^2$
gamma $10^0$, C $10^2$
gamma $10^1$, C $10^2$
gamma $10^{-1}$, C $10^4$
gamma $10^0$, C $10^4$
gamma $10^1$, C $10^4$
Grid Search: Cross Validated Scores
Parallel ML Use Cases

- Stateless Feature Extraction
- Model Assessment with Cross Validation
- Model Selection with Grid Search
- Bagging Models: Random Forests
- In-Loop Averaged Models
Embarrassingly Parallel
ML Use Cases

• Stateless Feature Extraction
• Model Assessment with Cross Validation
• Model Selection with Grid Search
• Bagging Models: Random Forests
• In-Loop Averaged Models
Inter-Process Comm.

Use Cases

• Stateless Feature Extraction
• Model Assessment with Cross Validation
• Model Selection with Grid Search
• Bagging Models: Random Forests
• In-Loop Averaged Models
Scaling Text Feature Extraction

The Hashing Trick
(Count|TfIdf)Vectorizer
Scalability Issues

• Builds an In-Memory Vocabulary from text tokens to integer feature indices
  • A Big Python dict: slow to (un)pickle
  • Large Corpus: ~10^6 tokens
• Vocabulary == Statefulness == Sync barrier
• No easy way to run in parallel
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> vec = TfidfVectorizer()
>>> vec.fit(["The cat sat on the mat."])
>>> vec.vocabulary_
{u'cat': 0,
 u'mat': 1,
 u'on': 2,
 u'sat': 3,
 u'the': 4}
The Hashing Trick

• Replace the Python dict by a hash function:

```python
>>> from sklearn.utils.murmurhash import *
>>> murmurhash3_bytes_u32('cat', 0) % 10
9L
>>> murmurhash3_bytes_u32('sat', 0) % 10
0L
```

• Does not need any memory storage

• Hashing is stateless: can run in parallel!
```python
>>> from sklearn.feature_extraction.text import HashingVectorizer
>>> vec = HashingVectorizer()
>>> out = vec.transform(["The cat sat on the mat.""])  
>>> out.shape
(1, 1048576)
>>> out.nnz  # number of non-zero elements
5
```
Some Numbers
Loading 20 newsgroups dataset for all categories
11314 documents - 22.055MB (training set)
7532 documents - 13.801MB (testing set)

Extracting features from the training dataset using a sparse vectorizer
done in 12.881007s at 1.712MB/s
n_samples: 11314, n_features: 129792

Extracting features from the test dataset using the same vectorizer
done in 4.043470s at 3.413MB/s
n_samples: 7532, n_features: 129792
HashingVectorizer

Loading 20 newsgroups dataset for all categories
11314 documents - 22.055MB (training set)
7532 documents - 13.801MB (testing set)

Extracting features from the training dataset using a sparse vectorizer
done in 5.281561s at 4.176MB/s
n_samples: 11314, n_features: 65536

Extracting features from the test dataset using the same vectorizer
done in 3.413027s at 4.044MB/s
n_samples: 7532, n_features: 65536
HashingVectorizer on Amazon Reviews

- Music reviews: 216MB XML file
  140MB raw text / 174,180 reviews: 53s

- Books reviews: 1.3GB XML file
  900MB raw text / 975,194 reviews: ~6min

- https://gist.github.com/ogrisel/4313514
Parallel Text Classification
HowTo: Parallel Text Classification

- All Text Data
- All Labels to Predict
Partition the Text Data

Text Data 1
Labels 1

Text Data 2
Labels 2

Text Data 3
Labels 3
Vectorizer in Parallel

<table>
<thead>
<tr>
<th>Text Data 1</th>
<th>Text Data 2</th>
<th>Text Data 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels 1</td>
<td>Labels 2</td>
<td>Labels 3</td>
</tr>
<tr>
<td>vec</td>
<td>vec</td>
<td>vec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vec Data 1</th>
<th>Vec Data 2</th>
<th>Text Data 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels 1</td>
<td>Labels 2</td>
<td>Labels 3</td>
</tr>
</tbody>
</table>
Train Linear Models in Parallel

<table>
<thead>
<tr>
<th>Text Data 1</th>
<th>Text Data 2</th>
<th>Text Data 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels 1</td>
<td>Labels 2</td>
<td>Labels 3</td>
</tr>
<tr>
<td>vec</td>
<td>vec</td>
<td>vec</td>
</tr>
<tr>
<td>Vec Data 1</td>
<td>Vec Data 2</td>
<td>Text Data 3</td>
</tr>
<tr>
<td>Labels 1</td>
<td>Labels 2</td>
<td>Labels 3</td>
</tr>
<tr>
<td>clf_1</td>
<td>clf_2</td>
<td>clf_3</td>
</tr>
</tbody>
</table>
Collect Models and Average

\[ \text{clf} = \left( \text{clf}_1 + \text{clf}_2 + \text{clf}_3 \right) / 3 \]
Averaging Linear Models

```python
>>> clf = clone(clf_1)
>>> clf.coef_ += clf_2.coef_
>>> clf.coef_ += clf_3.coef_
>>> clf.intercept_ += clf_2.intercept_
>>> clf.intercept_ += clf_3.intercept_
>>> clf.coef_ /= 3; clf.intercept_ /= 3
```
Averaging Linear Models

```python
>>> clf = clone(clf_1)
>>> clf.coef_ += clf_2.coef_
>>> clf.coef_ += clf_3.coef_
>>> clf.intercept_ += clf_2.intercept_
>>> clf.intercept_ += clf_3.intercept_
>>> clf.coef_ /= 3; clf.intercept_ /= 3
```
Averaging Linear Models

```python
>>> clf = clone(clf_1)
>>> clf.coef_ += clf_2.coef_
>>> clf.coef_ += clf_3.coef_
>>> clf.intercept_ += clf_2.intercept_
>>> clf.intercept_ += clf_3.intercept_
>>> clf.coef_ /= 3; clf.intercept_ /= 3
```
Averaging Linear Models

>>> clf = clone(clf_1)
>>> clf.coef_ += clf_2.coef_
>>> clf.coef_ += clf_3.coef_
>>> clf.intercept_ += clf_2.intercept_
>>> clf.intercept_ += clf_3.intercept_
>>> clf.coef_ /= 3; clf.intercept_ /= 3
Training Forest Models in Parallel
Tricks

• Try: ExtraTreesClassifier
  instead of: RandomForestClassifier
  • Faster to train
  • Sometimes better generalization too

• Both kind of Forest Models are naturally embarrassingly parallel models.
HowTo: Parallel Forests

All Data

All Labels to Predict
Partition Replicate the Dataset

<table>
<thead>
<tr>
<th>All Data</th>
<th>All Data</th>
<th>All Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Labels</td>
<td>All Labels</td>
<td>All Labels</td>
</tr>
</tbody>
</table>
Train Forest Models in Parallel

Seed each model with a different random_state integer!
Collect Models and Combine

Forest Models naturally do the averaging at prediction time.

```python
clf = (clf_1 + clf_2 + clf_3)
```
What if my data does not fit in memory?
HowTo: Parallel Forests (for large datasets)

<table>
<thead>
<tr>
<th>All Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Labels to Predict</td>
</tr>
</tbody>
</table>
Partition Replicate
Partition the Dataset

Data 1
Labels 1

Data 2
Labels 2

Data 3
Labels 3
Train Forest Models in Parallel

Data 1
Labels 1
clf_1

Data 2
Labels 2
clf_2

Data 3
Labels 3
clf_3

jeudi 7 mars 13
Collect Models and Sum

\[
\text{clf} = (\text{clf}_1 + \text{clf}_2 + \text{clf}_3)
\]

```python
>>> clf = clone(clf_1)
>>> clf.estimators_ += clf_2.estimators_
>>> clf.estimators_ += clf_3.estimators_
```
Warning

• Models trained on the partitioned dataset are not exactly equivalent of models trained on the unpartitioned dataset

• If very much data: does not matter much in practice:

Gilles Louppe & Pierre Geurts

http://www.cs.bris.ac.uk/~flach/
Implementing Parallelization with Python
Single Machine with Multiple Cores
```python
from multiprocessing import Pool
p = Pool(4)

p.map(type, [1, 2., '3'])
[int, float, str]

r = p.map_async(type, [1, 2., '3'])
r.get()
[int, float, str]
```
multiprocessing

- Part of the standard lib
- Simple API
- Cross-Platform support (even Windows!)
- Some support for shared memory
- Support for synchronization (Lock)
multiprocessing: limitations

- No docstrings in the source code!
- Very tricky to use the shared memory values with NumPy
- Bad support for `KeyboardInterrupt`
- `fork` without `exec` on POSIX
Joblib

- transparent **disk-caching** of the output values and lazy re-evaluation (memoization)
- easy simple **parallel computing**
- **logging and tracing** of the execution
>>> from os.path.join
>>> from joblib import Parallel, delayed

>>> Parallel(2)(delayed(join)('/ect', s)
...     for s in 'abc')
['/ect/a', '/ect/b', '/ect/c']
Usage in scikit-learn

- **Cross Validation**
  
  ```python
cross_val(model, X, y, n_jobs=4, cv=3)
  ```

- **Grid Search**
  
  ```python
  GridSearchCV(model, n_jobs=4, cv=3).fit(X, y)
  ```

- **Random Forests**
  
  ```python
  RandomForestClassifier(n_jobs=4).fit(X, y)
  ```
>>> from joblib import Parallel, delayed
>>> import numpy as np

>>> Parallel(2, max_nbytes=1e6)(
...     delayed(type)(np.zeros(int(i)))
...     for i in [1e4, 1e6])
[<type 'numpy.ndarray'>, <class 'numpy.core.memmap.memmap'>]
Only 3 allocated datasets shared by all the concurrent workers performing the grid search.

<table>
<thead>
<tr>
<th></th>
<th>(1, 1e3)</th>
<th>(10, 1e3)</th>
<th>(100, 1e3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1e4)</td>
<td><img src="matrix1_1e4.png" alt="Matrix" /></td>
<td><img src="matrix10_1e4.png" alt="Matrix" /></td>
<td><img src="matrix100_1e4.png" alt="Matrix" /></td>
</tr>
<tr>
<td>(1, 1e5)</td>
<td><img src="matrix1_1e5.png" alt="Matrix" /></td>
<td><img src="matrix10_1e5.png" alt="Matrix" /></td>
<td><img src="matrix100_1e5.png" alt="Matrix" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problems with multiprocessing & joblib

- Current Implementation uses fork without exec under Unix
- Break some optimized runtimes:
  - OpenBlas
  - Grand Central Dispatch under OSX
- Will be fixed in Python 3 at some point...
Multiple Machines with Multiple Cores
IPython.parallel

IP[y]: IPython Interactive Computing

- Parallel Processing Library
- Interactive Exploratory Shell

Multi Core & Distributed
Working in the Cloud

* StarCluster

- Launch a cluster of machines in one cmd:

  
  $ starcluster start mycluster -s 3 \
  -b 0.07 --force-spot-master

  $ starcluster sshmaster mycluster

- Supports Spot Instances provisioning

- Ships blas, atlas, numpy, scipy

- IPython plugin, Hadoop plugin and more
[global]
DEFAULT_TEMPLATE=ip

[key mykey]
KEY_LOCATION=~/.ssh/mykey.rsa

[plugin ipcluster]
SETUP_CLASS = starcluster.plugins.ipcluster.IPCluster
ENABLE_NOTEBOOK = True

[plugin packages]
setup_class = pypackage.PyPackageSetup
packages = msgpack-python, scikit-learn

[cluster ip]
KEYNAME = mykey
CLUSTER_USER = ipuser
NODE_IMAGE_ID = ami-999d49f0
NODE_INSTANCE_TYPE = c1.xlarge
DISABLE_QUEUE = True
SPOT_BID = 0.10
PLUGINS = packages, ipcluster
$ starcluster start -s 3 --force-spot-master demo_cluster
StarCluster - (http://star.mit.edu/cluster) (v. 0.9999)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Using default cluster template: ip
>>> Validating cluster template settings...
>>> Cluster template settings are valid
>>> Starting cluster...
>>> Launching a 3-node cluster...
>>> Launching master node (ami: ami-999d49f0, type: c1.xlarge)...
>>> Creating security group @sc-demo_cluster...
SpotInstanceRequest:sir-d10e3412
>>> Launching node001 (ami: ami-999d49f0, type: c1.xlarge)
SpotInstanceRequest:sir-3cad4812
>>> Launching node002 (ami: ami-999d49f0, type: c1.xlarge)
SpotInstanceRequest:sir-1a918014
>>> Waiting for cluster to come up... (updating every 5s)
>>> Waiting for open spot requests to become active...
3/3 |---------------------------------------------------------------| 100%
>>> Waiting for all nodes to be in a 'running' state...
3/3 |---------------------------------------------------------------| 100%
>>> Waiting for SSH to come up on all nodes...
3/3 |---------------------------------------------------------------| 100%
>>> Waiting for cluster to come up took 5.087 mins
>>> The master node is ec2-54-243-24-93.compute-1.amazonaws.com

jeudi 7 mars 13
>>> Configuring cluster...
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Configuring hostnames...
3/3 100%
>>> Creating cluster user: ipuser (uid: 1001, gid: 1001)
3/3 100%
>>> Configuring scratch space for user(s): ipuser
3/3 100%
>>> Configuring /etc/hosts on each node
3/3 100%
>>> Starting NFS server on master
>>> Configuring NFS exports path(s):
/home
>>> Mounting all NFS export path(s) on 2 worker node(s)
2/2 100%
>>> Setting up NFS took 0.151 mins
>>> Configuring passwordless ssh for root
>>> Configuring passwordless ssh for ipuser
>>> Running plugin ippackages
>>> Installing Python packages on all nodes:
>>> $ pip install -U msgpack-python
>>> $ pip install -U scikit-learn
>>> Installing 2 python packages took 1.12 mins
Running plugin ipcluster
Writing IPython cluster config files
Starting the IPython controller and 7 engines on master
Waiting for JSON connector file...
/Users/ogrisel/.starcluster/ipcluster/SecurityGroup:@sc-demo_cluster-us-east-1.json 100% ||
Time: 00:00:00 0.00 B/s
Authorizing tcp ports [1000-65535] on 0.0.0.0/0 for: IPython controller
Adding 16 engines on 2 nodes
2/2 100%
Setting up IPython web notebook for user: ipuser
Creating SSL certificate for user ipuser
Authorizing tcp ports [8888-8888] on 0.0.0.0/0 for: notebook
IPython notebook URL: https://ec2-54-243-24-93.compute-1.amazonaws.com:8888
The notebook password is: zYHoMhEA8rTJSCXj
*** WARNING - Please check your local firewall settings if you're having
*** WARNING - issues connecting to the IPython notebook
IPCluster has been started on SecurityGroup:@sc-demo_cluster for user 'ipuser'
with 23 engines on 3 nodes.
To connect to cluster from your local machine use:

from IPython.parallel import Client
client = Client('/Users/ogrisel/.starcluster/ipcluster/SecurityGroup:@sc-demo_cluster-us-
est-1.json', sshkey='/Users/ogrisel/.ssh/mykey.rsa')

See the IPCluster plugin doc for usage details:
http://star.mit.edu/cluster/docs/latest/plugins/ipython.html

IPCluster took 0.679 mins
Configuring cluster took 3.454 mins
Starting cluster took 8.596 mins
Demo!
Perspectives
2012 results by Stanford / Google
The YouTube Neuron
Thanks

• http://scikit-learn.org
• http://packages.python.org/joblib
• http://ipython.org
• http://star.mit.edu/cluster/
• http://speakerdeck.com/ogrisel

@ogrisel
If we had more time...
MapReduce?

\[
[ \langle k_1, v_1 \rangle, \langle k_2, v_2 \rangle, \ldots ]
\]

mapper

mapper

mapper

[ \langle k_3, v_3 \rangle, \langle k_4, v_4 \rangle, \ldots ]

reducer

reducer

[ \langle k_5, v_6 \rangle, \langle k_6, v_6 \rangle, \ldots ]

hadoop
disco

massive data - minimal code

jeudi 7 mars 13
Why MapReduce does not always work

**Write** a lot of stuff to **disk** for failover

*Inefficient for small to medium problems*

\[
\text{[(k, v)] mapper [(k, v)] reducer [(k, v)]}
\]

*Data and model params as (k, v) pairs?*

Complex to leverage for **Iterative** Algorithms
When MapReduce is useful for ML

- Data Preprocessing & Feature Extraction
- Parsing, Filtering, Cleaning
- Computing big JOINs & Aggregates
- Random Sampling
- Computing ensembles on partitions
The AllReduce Pattern

- Compute an aggregate (average) of active node data
- Do not clog a single node with incoming data transfer
- Traditionally implemented in MPI systems
AllReduce 0/3

Initial State

Value: 1.0
Value: 2.0
Value: 0.5
Value: 1.1
Value: 3.2
Value: 0.9

jeudi 7 mars 13
AllReduce 1/3

Spanning Tree

Value: 1.0

Value: 2.0

Value: 0.5

Value: 1.1

Value: 3.2

Value: 0.9

jeudi 7 mars 13
AllReduce 2/3

Upward Averages

Value: 1.0

Value: 2.0

Value: 0.5

Value: 1.1

Value: 3.2

Value: 0.9

(1.1, 1)

(3.1, 1)

(0.9, 1)

jeudi 7 mars 13
AllReduce 2/3
Upward Averages

Value: 1.0

Value: 2.0
(2.1, 3)

Value: 0.5
(0.7, 2)

Value: 1.1
(1.1, 1)

Value: 3.2
(3.1, 1)

Value: 0.9
(0.9, 1)

jeudi 7 mars 13
AllReduce 2/3

Upward Averages

Value: 1.0
(1.38, 6)

Value: 2.0
(2.1, 3)

Value: 0.5
(0.7, 2)

Value: 1.1
(1.1, 1)

Value: 3.2
(3.1, 1)

Value: 0.9
(0.9, 1)

jeudi 7 mars 13
AllReduce 3/3

Downward Updates

Value: 1.38

Value: 2.0
   (2.1, 3)

Value: 0.5
   (0.7, 2)

Value: 1.1
   (1.1, 1)

Value: 3.2
   (3.1, 1)

Value: 0.9
   (0.9, 1)
AllReduce 3/3

Downward Updates

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.1

(1.1, 1)

Value: 3.2

(3.1, 1)

Value: 0.9

(0.9, 1)

jeudi 7 mars 13
AllReduce 3/3

Downward Updates

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38

jeudi 7 mars 13
AllReduce Final State

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38

Value: 1.38
AllReduce Implementations

http://mpi4py.scipy.org

IPC directly w/ IPython.parallel

https://github.com/ipython/ipython/tree/master/docs/examples/parallel/interengine
Killall IPython engines on StarCluster

[plugin ipcluster]
SETUP_CLASS = starcluster.plugins.ipcluster.IPCluster
ENABLE_NOTEBOOK = True
NOTEBOOK_DIRECTORY = notebooks

[plugin ipclusterrestart]
SETUP_CLASS = starcluster.plugins.ipcluster.IPClusterRestartEngines
$ starcluster runplugin ipclusterrestart demo_cluster
StarCluster - (http://star.mit.edu/cluster) (v. 0.9999)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Running plugin ipclusterrestart
>>> Restarting 23 engines on 3 nodes
3/3 100%