Python for High Performance Computing

Monte Lunacek

Research Computing, University of Colorado Boulder
University of Colorado

- Some tightly-coupled MPI codes
- Many independent tasks
- Diverse computing backgrounds
  - Geography
  - Ecology and Evolutionary Biology
  - Microbial Ecology
  - Astronomy
  - Geology
- Range of computational experience
High Throughout Computing

- Simulations
  - Monte Carlo
  - Parameter scan
  - Uncertainty Quantification
- Parameter Optimization
- Data Analysis (MapReduce)
- Parallel workflows
Supercomputing Without the Pain

• Accessible to anyone with:
  • Simulation or analysis to run
  • Desire to do it faster
• Remove barriers to entry
Success Stories

~500,000 simulations on ~7,000 cores with mpi4py (http://mpi4py.scipy.org/)
Parameter optimization on ~100 cores with Scoop (https://code.google.com/p/scoop/) and DEAP (https://code.google.com/p/deap/)
Improved biological workflow with IPython Parallel (http://ipython.org/ipython-doc/dev/parallel/)
Wrapped an engineering simulation with f2py (http://www.scipy.org/F2py) and IPython Parallel (http://ipython.org/ipython-doc/dev/parallel/)
Outline

- Python (http://python.org)
- Ipython Notebook (http://ipython.org/ipython-doc/dev/interactive/htmlnotebook.html)
- High Throughput Computing
  - IPython Parallel (http://ipython.org/ipython-doc/dev/parallel/)
  - Scoop (https://code.google.com/p/scoop/)
  - mpi4py (http://mpi4py.scipy.org/)
- Data Analysis with pandas (http://discoproject.org/)
- Conclude
What is Python?
Python

- Flexible, powerful programming language
  - Object oriented
  - Runs everywhere
- Easy, clean syntax
- Glue: Cython, F2py
- Large community of support
  - Consistent feel
- Free as in **free beer**
- Free as in **free speech**
Packages for Computational Science

- **python**: the base language
- **numpy**: arrays, fast operations on arrays
- **scipy**: higher level computational routines
- **matplotlib**: plotting
- **ipython**: notebooks, flexible shell, and parallel
- **pandas**: data analysis
What can you do with Python?

- OS support: manage files and directories
- Glue existing applications
- LAPACK and BLAS: access powerful C and Fortran libraries
- Parallel
- Data Analysis
- Visualization
- GUI programming
- Scrape websites
- Build websites
- Anything!
Distributions

- Python(x,y) (http://www.pythonxy.com/)
- Anaconda (https://store.continuum.io/cshop/anaconda)

IPython terminal

ipython --pylab

IPython notebook

ipython notebook --pylab=inline
This webpage is not available

Google Chrome's connection attempt to 127.0.0.1 was rejected. The website may be down, or your network may not be properly configured.

Here are some suggestions:

- [Reload this webpage later.](#)
- Check your Internet connection. Restart any router, modem, or other network devices you may be using.
- Add Google Chrome as a permitted program in your firewall’s or antivirus software’s settings. If it is already a permitted program, try deleting it from the list of permitted programs and adding it again.
- If you use a proxy server, check your proxy settings or contact your network administrator to make sure the proxy server is working. If you don’t believe you should be using a proxy server, adjust your proxy settings: Go to **Applications > System Preferences > Network > Advanced > Proxies** and deselect any proxies that have been selected.

Error 102 (net::ERR_CONNECTION_REFUSED): The server refused the connection.
High Throughput Computing
Bash

```bash
count_base=0
for i in {1..N}
do
  for j in {1..12}
do
    b=($($count_base + $j))
    ./simulator -s 5 -t $b &
  done
wait
  count_base=($($count_base + $np))
done

Limited to a **single node**
```
#!/bin/bash
PATH=$PBS_O_WORKDIR:$PBS_O_PATH
TRIAL=$(($PBS_VNODENUM + $1))
python ./simulator.py -s 5 -t $TRIAL

for i in {1..N}
do
  pbsdsh wrapper.sh $count
  count=$(($count + 12))
done

A little painful
A little inefficient
Objective Function

```python
def simulation(x):
    value = x*x + 10
    return value
```

The functions name is `simulation`
Multiprocessing

Import

```python
from multiprocessing import Pool
```

Map the values

```python
if __name__ == '__main__':
    pool = Pool(12)  # workers
    data = range(200)  # tasks
    results = pool.map(simulation, data)
```

Great for single node

Python's `threading` library
Scoop

Import

```python
from scoop import futures
```

Map the values

```python
if __name__ == '__main__':
    data = range(200)  # tasks
    results = futures.map(simulation, data)
```

Launch

```bash
python -m scoop filename.py
```

**Efficient** startup!
from IPython.parallel import Client, require

Map the values

if __name__ == '__main__':

    data = range(200) # tasks
    rc = Client(profile='mpi')
    lview = rc.load_balanced_view()

    results = lview.map(simulation, data)
    results.wait()
Compare

results = pool.map(simulation, data)

results = futures.map(simulation, data)

results = lview.map(simulation, data)

It's the way you create the **object.map()** that separates these methods.
## Compare

### Good

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IPython</td>
<td>Fault-tolerance</td>
</tr>
<tr>
<td>IPython</td>
<td>Schedule</td>
</tr>
<tr>
<td>IPython</td>
<td>Interactive</td>
</tr>
<tr>
<td>Scoop</td>
<td>Efficient launch</td>
</tr>
<tr>
<td>Multiprocessing</td>
<td>Included in the standard Library</td>
</tr>
</tbody>
</table>

### Needs work

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Scaling unknown</td>
</tr>
<tr>
<td>IPython</td>
<td>Launcher (configuration)</td>
</tr>
<tr>
<td>Scoop and MP</td>
<td><strong>main</strong></td>
</tr>
<tr>
<td>Scoop</td>
<td>Schedule</td>
</tr>
<tr>
<td>Multiprocessing</td>
<td>One node (kind of)</td>
</tr>
</tbody>
</table>
Scheduling: Static
Scheduling: Dynamic
Tenacious Robustness Test

```python
@require('time','socket','random',
'IPython.parallel.error.KernelError')
def simulation(x):
    time.sleep(5)
    if random.random() < 0.3:
        raise KernelError
    return {'task':x,
            'host':socket.gethostname()}
```

Launch 10 nodes

Run several tasks

At some point, kill a node
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    data = {'key1': [7, 2.72, 3.2],
            'key2': ('abc', 'xyz')}
else:
    data = None

data = comm.bcast(data, root=0)
mpi4py Scaling

3 second jobs

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2048</td>
<td>92</td>
</tr>
<tr>
<td>4096</td>
<td>87%</td>
</tr>
<tr>
<td>8192</td>
<td>67%</td>
</tr>
</tbody>
</table>
Data Analysis
This webpage is not available

Google Chrome's connection attempt to 127.0.0.1 was rejected. The website may be down, or your network may not be properly configured.

Here are some suggestions:

- **Reload** this webpage later.
- Check your Internet connection. Restart any router, modem, or other network devices you may be using.
- Add Google Chrome as a permitted program in your firewall's or antivirus software's settings. If it is already a permitted program, try deleting it from the list of permitted programs and adding it again.
- If you use a proxy server, check your proxy settings or contact your network administrator to make sure the proxy server is working. If you don't believe you should be using a proxy server, adjust your proxy settings: Go to Applications > System Preferences > Network > Advanced > Proxies and deselect any proxies that have been selected.

Error 102 (net::ERR_CONNECTION_REFUSED): The server refused the connection.
Conclusions

Python makes **supercomputing accessible**

Combine libraries to achieve the task at hand.

- Simulate and analyze
- Share methods in a notebook
- Push your data to a database
- Share it on the web
- In **parallel**
References

- Python Scripting for Computational Science
  (http://www.springer.com/mathematics/computational+science+%26+en&3-540-73915-9)

- Python Snakes Its Way Into HPC
  (http://www.hpcwire.com/hpcwire/2010-11-17/python_snakes_its_way_into_hpc.html)

- Andy Terrel: Getting Started with Python in HPC
  (http://andy.terrel.us/blog/2012/09/27/starting-with-python/)

- Python Tutorial (http://docs.python.org/2/tutorial/)

- Think Python (http://www.greenteapress.com/thinkpython/)

- Data Analysis with Python