IPython & StarCluster: Scalable Cloud Computing for the rest of us (a Qiime illustration)

Fernando Pérez, Min Ragan-Kelley, Brian Granger, Justin Riley
Greg Caporaso, Tony Walters, Rob Knight

http://ipython.org

Fernando.Perez@berkeley.edu

NIH Cloud Workshop, CU Boulder
April 3, 2012
A CU Boulder project

- Started when I was a graduate student in Physics (2001).
- Continued as a postdoc in Applied Mathematics.
- Brian Granger: CU Physics.

In brief

1. A better Python shell
2. Embeddable Kernel and powerful interactive clients
   - Terminal
   - Qt console
   - Web notebook
3. Flexible parallel computing
A CU Boulder project
- Started when I was a graduate student in Physics (2001).
- Continued as a postdoc in Applied Mathematics.
- Brian Granger: CU Physics.

In brief

1. A better Python shell
2. Embeddable Kernel and powerful interactive clients
   - Terminal
   - Qt console
   - Web notebook
3. Flexible parallel computing
IPython: Matlab/IDL-like interactive use

```python
In [1]: import math, numpy
In [2]: from scipy.integrate import quad
In [3]: from scipy.special import j0
In [4]: def j0i(x):
   ...:     """Integral form of J_0(x)""
   ...:     def integrand(phi):
   ...:         return math.cos(x*math.sin(phi))
   ...:     return (1.0/math.pi)*quad(integrand,0,math.pi)[0]
In [5]: x = numpy.linspace(0,20,200) # sample grid: 200 points between 0 and 20
In [6]: y = j0i(x) # sample J0 at all values of x
In [7]: x1 = x[:,::10] # subsample the original grid every 10th point
In [8]: y1 = map(j0i,x1) # evaluate the integral form at all points in x1
In [9]: # Make a plot with these values (the ; suppresses output)
In [10]: plot(x,y,label='J_S0(0)(x)');
In [11]: plot(x1,y1,'ro',label='J_S0^\text{integ}(x)');
In [12]: axhline(0,color='green',label='_nolegend_');
In [13]: title(r'\text{Verify } J_0(x) = \frac{1}{\pi} \int_0^\pi \cos(x \sin \phi) \, d\phi$');
In [14]: xlabel('x');
In [15]: legend();
In [16]: matshow(numpy.random.random((32,32)));
```

Welcome to pylab, a matplotlib-based Python environment.
For more information, type `help(pylab)`.

Qt console: inline plots, html, multiline editing, ...

Evan Patterson (Enthought)

```
In [1]: import scipy.linalg as la
   ....: mineigs = []
   ....: n = 256
   ....: for i in range(10):
   ....:     a = rand(n, n)
   ....:     mineigs.append(la.eigvals(a).min().real)
   ....: mean(mineigs)

Out[1]: -4.569467643237938

In [2]: %run mapping_seismic_stations.py

In [3]: |
```
This example computes PI to certain precision using 4 processors and a Monte Carlo simulation.

```python
import random
from mpi4py import MPI
comm = MPI.COMM_WORLD
import numpy as np

def computePi(nsamples):
    rank, size = comm.Get_rank(), comm.Get_size()
    oldpi, pi, mypi = 0.0, 0.0, 0.0

    done = False
    while not done:
        inside = 0
        for i in xrange(nsamples):
            x = random.random()
            y = random.random()
            if (x**2) + (y**2) < 1:
                inside += 1

        oldpi = pi
        mypi = (inside * 1.0) / nsamples
        pi = (oldpi + mypi) / 2

    return pi
```

In [1]: from IPython.parallel import Client
In [2]: rc = Client()
In [3]: rc.ids
[0, 1, 2, 3]
In [4]: dview = rc[:]
In [5]: serial_result = map(lambda x: x**10, range(32))
In [6]: parallel_result = dview.map_sync(lambda x: x**10, range(32))
In [7]: serial_result==parallel_result
True
In [8]: parallel_result
[0,
1,
1024,
59049,
1048576,
9765625,
60466176,
282475249,
1073741824,
3486784401L,
100000000000L,
25937424601L,
61917364224L,
137858491840L,
289254654976L,
576652982268L]
Simple spectral analysis

An illustration of the Discrete Fourier Transform

\[
X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}kn} \quad k = 0, \ldots, N - 1
\]

using windowing, to reveal the frequency content of a sound signal.

We begin by loading a datafile using SciPy's audio file support:

```python
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

```python
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```
Interactive and high-level parallel APIs
Min Ragan-Kelley, Brian Granger
- **Enthought**, Austin, TX: **Lots!**
- **Tech-X Corporation**, Boulder, CO: Parallel/notebook (previous versions)
- **Microsoft**: WinHPC support, Visual Studio integration
- **NIH**: via NiPy grant
- **NSF**: via Sage compmath grant
- **DoD/HPTi**.
Brian Granger - Cal State San Luis Obispo Physics
Min Ragan-Kelley - UC Berkeley Nuclear engineering.
Thomas Kluyver - U. Sheffield Plant biology
Jörgen Stenarson - SP Technical Research Institute of Sweden
Paul Ivanov - UC Berkeley neuroscience
Robert Kern - Enthought
Evan Patterson - Caltech Physics/Enthought
Stefan van der Walt - UC Berkeley
John Hunter - TradeLink Securities, Chicago.
Prabhu Ramachandran - Aerospace Engineering, IIT Bombay
Satra Ghosh - MIT Neuroscience
Gaël Varoquaux - Neurospin (Orsay, France)
Ville Vainio - CS, Tampere University of Technology, Finland
Ondrej Certik - Physics, U Nevada Reno
Darren Dale - Cornell
Justin Riley - MIT
Mark Voorhies - UC San Francisco
Nicholas Rougier - INRIA Nancy Grand Est
Thomas Spura - Fedora project
Julian Taylor - Debian/Ubuntu
Many more! (~140 commit authors)
Simplify the process of building, configuring, and managing clusters of virtual machines on Amazon’s EC2 cloud.
Resource Links

IPython

http://ipython.org/videos.html

http://ipython.org/documentation.html

StarCluster

http://web.mit.edu/star/cluster

StarCluster/IPython/Qiime public AMI:

ami-2faa7346