

IPython

A New Architecture for Interactive and Parallel Computing

Fernando Pérez¹, Brian Granger², Min Ragan-Kelley¹,
Thomas Kluyver³, Evan Patterson⁴

¹UC Berkeley, ²Cal Poly San Luis Obispo, ³U. Sheffield, ⁴Caltech

Contact: `Fernando.Perez@berkeley.edu`

<http://ipython.org>

SciPy'2011

Austin, July 13, 2011

What is IPython?

A toolkit for manipulating namespaces

IPython: the soundbite edition

Getting all the power from interactive computing in Python

- 1 A better Python shell
- 2 A flexible, embeddable interpreter
- 3 Data visualization and GUIs
- 4 A rich toolkit: terminal, Qt console, HTTP client.
- 5 High level (and interactive!) parallel computing interfaces.

The last two years

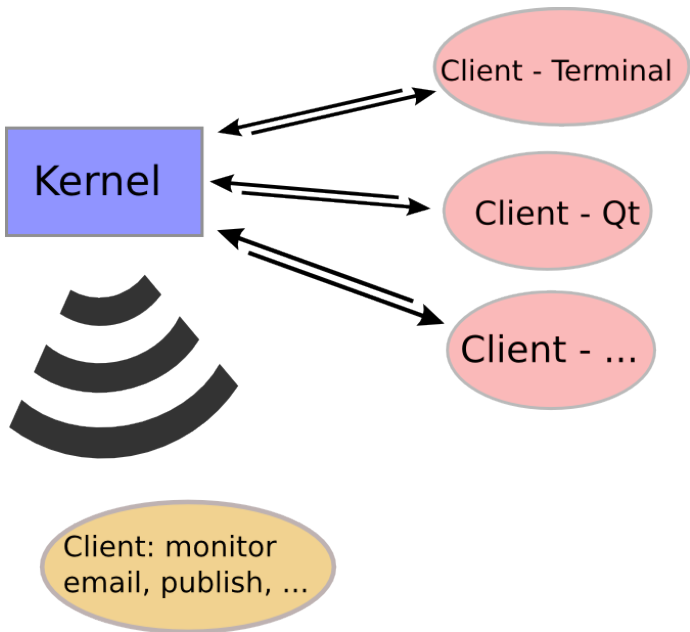
- **Completely reorganized** codebase, much cleaner internal structure
 - A much **more welcoming** project to contribute to!
- Moved to **git/github**: bzt/launchpad was a nightmare, g/gh rock!
 - Between Oct'10 and July'11: over **200 pull requests** merged.
- We touched **everything**. Since the merge-base of 0.10.2 and HEAD:
 - 2074 commits
 - 284713 line diff
- But we didn't break **anything**! (well, almost)
- Great **new contributors** in the core team
 - **Thomas Kluyver**: Python3 port, now everywhere
 - **Evan Patterson**: Qt console

Does plain old IPython still work?

It better!

- Better **configuration** system
- Cleaner support for GUIs without brittle threading tricks
- **SQLite** backend for all input/output **history**.
- **Hundreds** of small **improvements** everywhere
 - ... but the same **comfortable feel** of old, well-worn shoes.

More complex interactive uses?



A messaging protocol

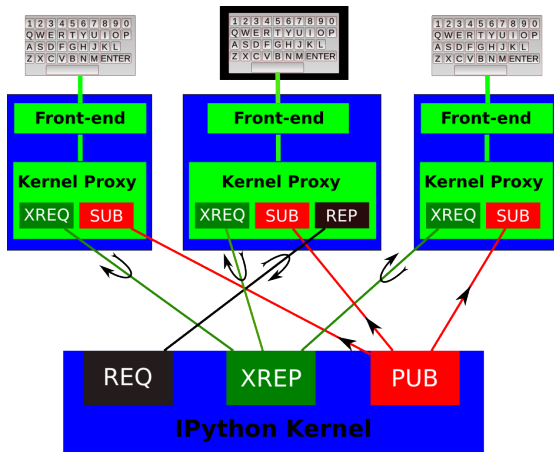
Direct communication

- Execute code ('eval')
- Object information
- Complete
- History
- Connect

Broadcasting

- Functional execution:
 - Python inputs
 - Python outputs
 - Python errors
- Side effects:
 - Streams (stdout, stderr, etc)
 - Display data: plots, other payloads

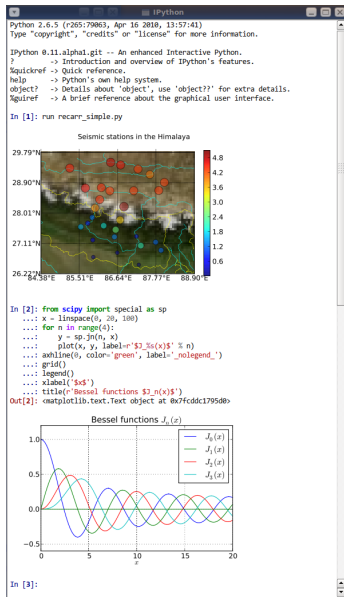
Interactive IPython on ØMQ



- - Kernel raw_input
- - Requests to kernel
- - Kernel output broadcast
- ↻ - Request/Reply direction

Back to the clients: a rich Qt Console

Enthought: sponsorship, Evan Patterson, Robert Kern.



Feels like a console, runs like a GUI

- Inline and floating images
- Syntax highlighting, full multiline editing
- Session saving
 - HTML (with PNG or SVG)
 - PDF/printing
- Help viewer
- %magics, !system access, IPython...
- Detach/reattach support

IPython Notebook

Cell Kernel Help

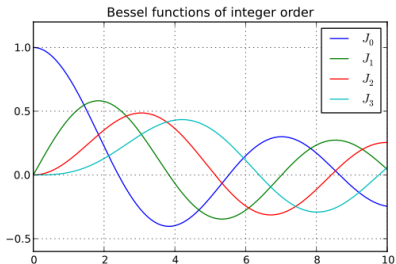
↑ ↓ Before After × Code Text Sort Collapse Expand

Bessel functions

Simple plots of the $J_n(x)$ Bessel functions for x in $[0, 10]$.

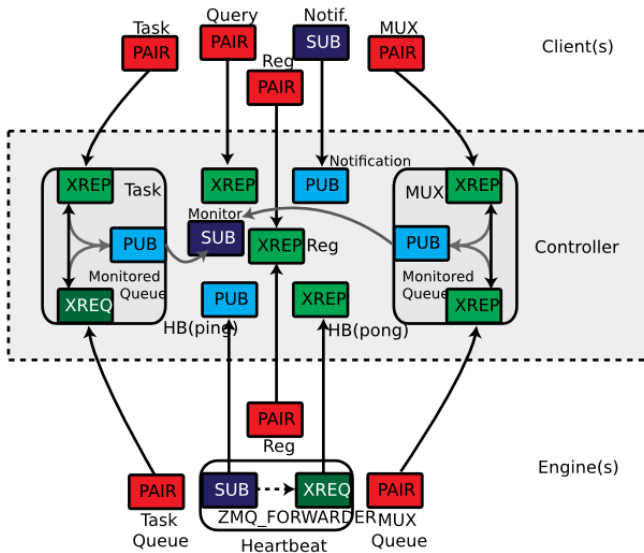
```
In [2]: import scipy.special as sp
x = linspace(0, 10, 200)
for n in range(4):
    plot(x, sp.jn(n, x), label='%J_%s$' % n)
grid()
legend()
title('Bessel functions of integer order');
```

Out[2]:



IPython for parallel computing

With Brian Granger (Cal Poly San Luis Obispo), Min Ragan-Kelley (Berkeley)



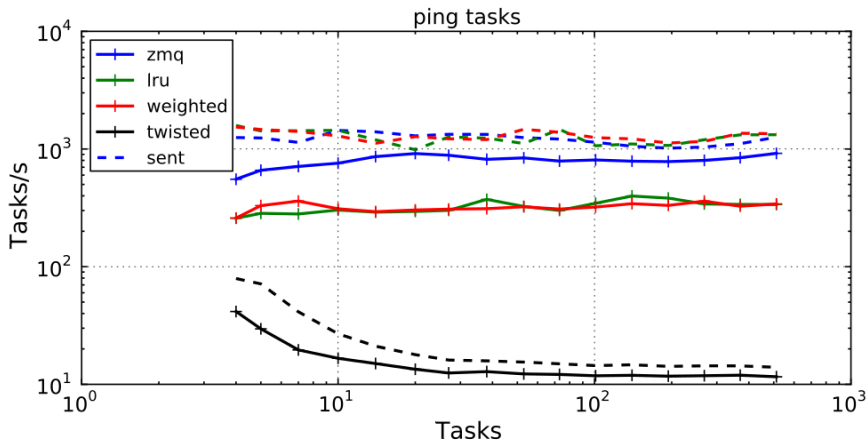
A few simple concepts

- The **client**: lightweight handle on all engines of a cluster
- The **views**: “slice” the client with specific execution semantics
 - **DirectView**: direct execution on *all* engines (blocking or not)
 - **LoadBalancedView**: run on *any one* engine.
- **Apply**: highly functional API
- **AsyncResult**: similar to the one in **multiprocessing**.
- The **hub**: group control of all activity in a cluster (accessed via *clients*)

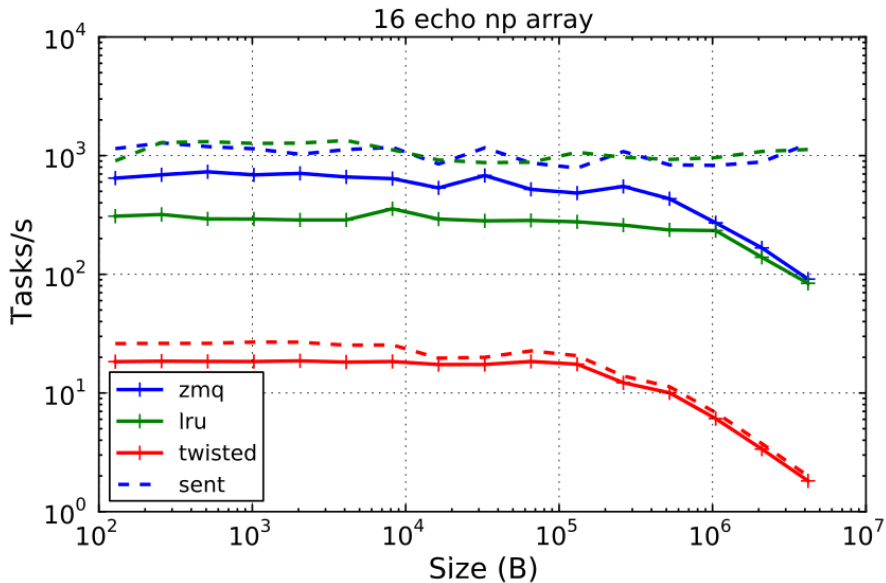
Multiple usage patterns

- **Direct** interface: explicit (and flexible) control of where things run.
 - Choice of blocking behavior up to the user.
- **Task** interface: load-balanced (with flexible scheduling policies)
- **Data** push/pull, scatter/gather.
- **Decorators** that encapsulate many common patterns
- Informative **exception** propagation
- Explicit **node-to-node** communication:
 - MPI-style tasks
 - ... without all the pain of MPI.

Phenomenal task latency



...and throughput



A real-world example (stripped down)

Distributed LASSO - using RegReg by J. Taylor and B. Klingenberg

```
# Create IPython objects to control parallel cluster
from IPython.parallel import Client
view = Client[:]
view.block = True

...
# Define a function that runs remotely, is called locally
@view.remote()
def update_lasso_nodes(pseudo_response, tol):
    node.response = node.fitted + pseudo_response
    node.solver.fit(max_its=1000, min_its=10, tol=tol)
    beta[:] = node.beta
    return node.fitted

...
# Fitting loop, calling remote functions
for i in range(max_iter):
    # Perform remote computations
    fits = update_lasso_nodes(mu-Xbeta-u, tol)
    Xbeta, mu, u = update_global_variables(fits, Y, u, rho)
    # Gather results for local operation
    beta = view.gather('beta')
    new_obj = objective(beta, X, Y, lagrange)
    # Check convergence, break, etc.
```

Neat trick: DAG dependencies

A simple DAG example

```
In [2]: G = random_dag(32,128)
In [3]: jobs = {}

# in reality, each job would presumably be different
# randomwait is just a function that sleeps for a random interval
In [4]: for node in G:
...:     jobs[node] = randomwait

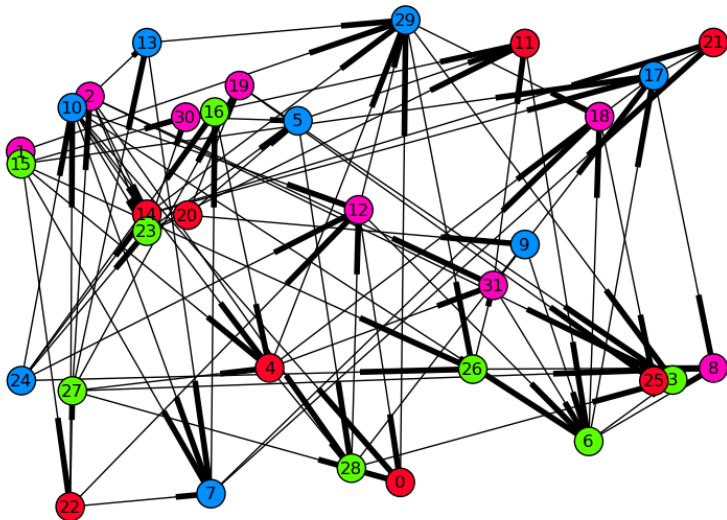
In [5]: c = client.Client()

In [6]: results = {}

In [7]: for node in G.topological_sort():
...:     # get list of AsyncResult objects from nodes
...:     # leading into this one as dependencies
...:     deps = [ results[n] for n in G.predecessors(node) ]
...:     # submit and store AsyncResult object
...:     results[node] = client.apply(jobs[node], after=deps,
        block=False)

In [8]: [ r.get() for r in results.values() ]
```

DAG dependencies - validation



What next?

- **Release 0.11** (next week)
- Complete the clients:
 - **Two-process terminal** client
 - **One-process Qt** console.
 - One and two-process **curses** client (SSH environments)
- Push hard on the **HTML client**
 - Better UI (ExtJS/JQuery?, ...)
 - Document model for re-execution
- Complete, improve **parallel** APIs
 - No-op client/view for local/serial debugging.
 - Improved node-to-node interfaces (yes, MPI, we're talking to you)
- Full **integration** of parallel and interactive client code (almost there).

Please join us!
<http://github.com/ipython>

Support

Thank you!

- **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
- **Google**: summer of **code** 2005, 2010.

(Incomplete) Cast of Characters

- **Brian Granger** - Physics, Cal State San Luis Obispo
- **Min Ragan-Kelley** - UC Berkeley
- **Thomas Kluyver** - U. Sheffield
- **Evan Patterson**- Caltech/Enthought
- **Robert Kern** - Enthought
- **Jörgen Stenarson** - Sweden.
- Ondrej Certik - Physics, U Nevada Reno
- Darren Dale - Cornell
- Laurent Dufrécho - France
- James Gao - UC Berkeley
- Satra Ghosh- MIT Neuroscience
- John Hunter - TradeLink Securities, Chicago.
- Paul Ivanov - UC Berkeley
- Prabhu Ramachandran - Aerospace Engineering, IIT Bombay.
- Justin Riley - MIT
- Thomas Spura - Fedora project
- Ville Vainio - CS, Tampere University of Technology, Finland
- Stefan van der Walt - Applied Math, U. Stellenbosch, South Africa
- Gaël Varoquaux - Neurospin (Orsay, France)
- Mark Voorhies - UC San Francisco
- **Many more!** (~60 commit authors)

Thank you!

Questions?