
PyPWA

Package Features

- A blend of Functional and Object Oriented Programming
- Plugin Based Architecture
- Supports Python 2.7, and Python 3.4+
- Installable in Anaconda or with Pip
- Continuously tested with Travis-CI
- Latest documentation generated by Readthedocs
- Code coverage tracked by Coveralls

About Python

- Can be used as glue
- Typeless Scripting Language
- Supports both Object Oriented and Functional Programming practices.
- Nice GUIs with PyQt
- Numpy and SciPy for Math
- Graphing with Matplotlib

Worldwide, Jun 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.7 %	-1.3 %
2		Python	15.7 %	+3.5 %
3		PHP	9.3 %	-1.1 %
4		C#	8.3 %	-0.5 %
5		Javascript	7.9 %	+0.5 %
6		C++	6.9 %	-0.2 %
7		C	6.7 %	-0.1 %

Plugin System

- Everything is a plugin approach
- Plugins can have Plugins
- All plugins loaded at runtime by PluginLoader
  - Extendable by other plugins
  - Search by baseclass or by function name

- Current Plugin Types:
  - Kernel Processing
  - Shell Modules
  - Likelihoods
  - Optimizers
  - Data Parsers
  - Data Iterators
  - Data Loaders

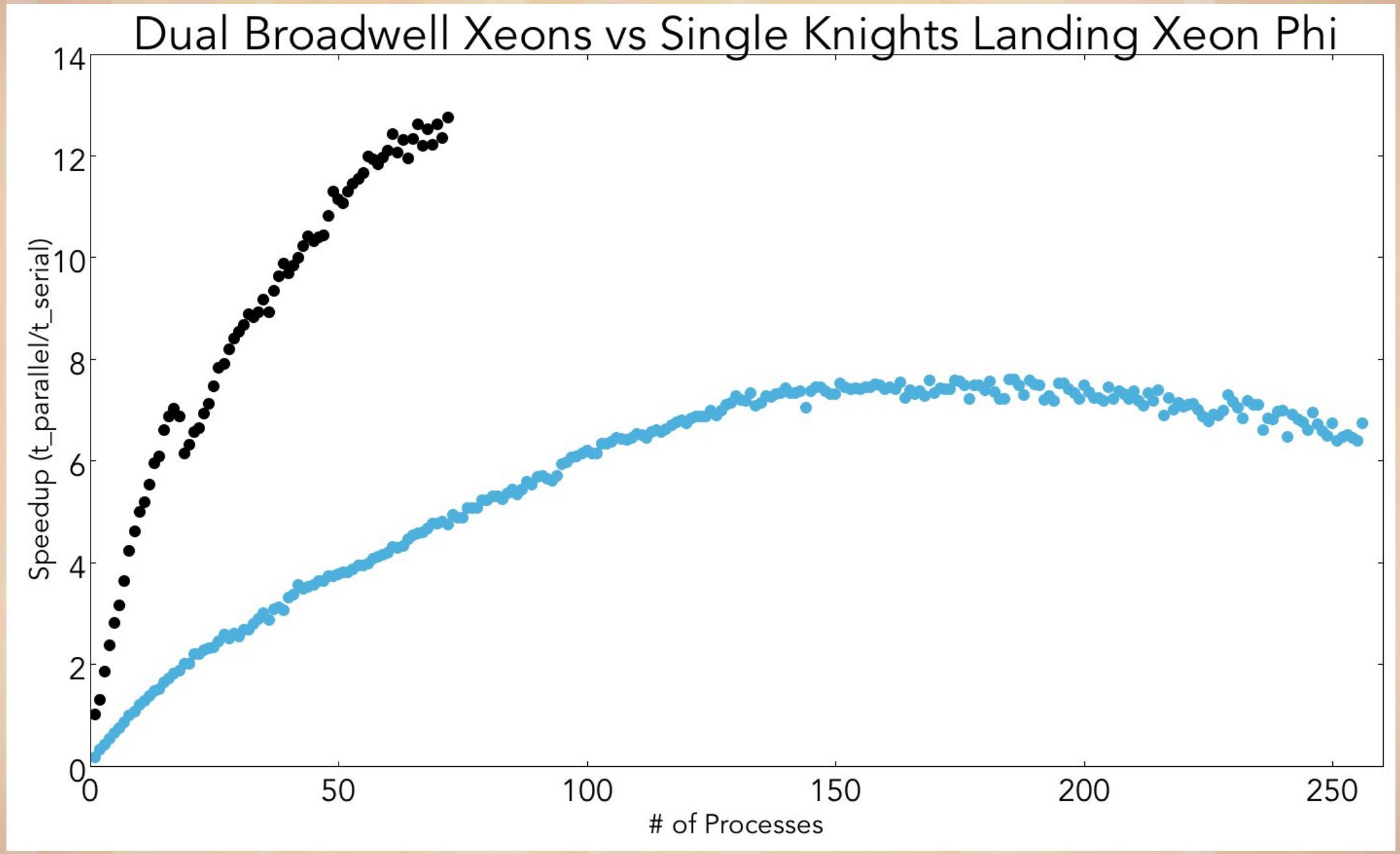
Core Libs

Builtin Processing

- Easily extendable
- Scales efficiently across many cores
- Can use all available physical cores of the machine.
- Has been thoroughly tested with Pure Numpy and with FORTRAN amplitudes
- Supports the XeonPhi Knights Landing

Builtin Data

- Can parse straight to Numpy array or iterate one event at a time
- Supports CSV, TSV, and GAMP data
- Values are read once then cached for future use.
  - If source file changes, cache file is automatically invalidated.


Dual Broadwell Xeons vs Single Knights Landing Xeon Phi
Speedup (t_parallel/t_serial)
# of Processes
0
2
4
6
8
10
12
14
0
50
100
150
200
250

Optimizers

iMinuit

- A Cython wrapper around ROOT's Minuit2
- Will only find nearest local minima
- Quick in execution
- Should operate much like the familiar Minuit2 optimizer

Nestle

- A pythonic rewrite of Multinest
- Finds all minimas in the parameter space
- Takes significantly longer to run
- Samples uniformly or non-uniformly in a user defined parameter space

Configurator

- All configuration for all main programs is through YAML
- Configuration template can be generated with a simple TUI
- Options default to sane choices.

```
Builtin Multiprocessing:
  number of processes: 8
Nestle:
  prior location: processing_function.py
  prior name: prior_function
  ndim: 1
  npoints: 100
  method: single
Builtin Parser:
  enable cache: true
General Fitting:
  likelihood type: likelihood
  generated length: 10000
  function's location: processing_function.py
  processing name: processing_function
  setup name: setup_function
  qfactor location: # Blank defaults to None
  data location: data.csv
  internal data:
    quality factor: Qfactors
  accepted monte carlo location: monte_carlo.tsv
  save name: example_fit
```

PyFit and PySim

- Generate a template configuration with PyFit -wc
- Can take an Amplitude in C or FORTRAN and scale it across multiple physical cores without special work from the user
- Can use the same processing function for PyFit and PySim

Extended Log Likelihood

$$\sum_{i=0}^{n} Q * \ln I(data) + \frac{1}{Number\ of\ generated\ events} * \sum_{i=0}^{n} I(monte\ carlo)$$

Log Likelihood, Optionally Binned

$$\sum_{i=0}^{n} Qf * Bin * \ln I(data)$$

Chi-squared Likelihood

$$\sum_{i=0}^{n} \frac{(I(data) - expected)^2}{error}$$

Chi-squared Binned Likelihood

$$\sum_{i=0}^{n} \frac{(I(data) - Bin)^2}{Bin}$$

Empty

$$\sum_{i=0}^{n} I(data)$$

Testing

Travis-CI

- All tests defined with py.test
- Every push to Github runs the entire test suit against python 2.7, 3.4, 3.5, and 3.6
- Pull Requests can't be merged if tests fail
- We are notified if any of our tests fail.

Coveralls

- Tracks our test coverage over time
- Reports test coverage change on each pull request
- Gives us a precentage of our code that is tested by our test suite

Summary

- Uses a git branch workflow
- A standard coding standard to emphasize clean code
- Techniques borrowed from Robert C. Martin's Clean Code
- Documentation is kept in sync with the project.
- Uses a plugin architecture
- Portable by design
- Can be used with lower level languages
- 92% test coverage
- Supports Multinest and Minuit
- Supports Multiprocessing
- Currently 2.5 developers working on the package

Github
 - https://www.github.com/JeffersonLab/PyPWA
Travis-CI
 - https://travis-ci.org/JeffersonLab/PyPWA
Coveralls
 - https://coveralls.io/github/JeffersonLab/PyPWA