ANACONDA

Numba and conda
Tools for fast reliable data analysis

Jonathan Helmus, Software Engineer

Who am I?

Jonathan Helmus

- software engineer at Anaconda
- member of the distribution team
- conda developer
- conda-forge core team

Numba

- A type-specializing JIT compiler for Python
- Uses LLVM codegen
  - clang backend
- Focus in scientific/numeric use cases
  - numpy.array, loops
- Aim to bridge high-level programming and high-performance computations

What Numba is not

- Not a general-purpose JIT for Python
  - Numba cannot speedup your favorite web framework
- Not a replacement for the CPython interpreter
  - Numba is a Python library
  - Operates a function at a time

If you are looking for something like this investigate PyPy

The Basic API

@numba.jit

```
@numba.jit
def mvmul(A, x):
    return A * x
```

```
In [22]: %timeit mvmul(A, x)
28.2 µs ± 556 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)

In [23]: %timeit py_mvmul(A, x)
42.1 µs ± 1.01 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

x1.5 speedup

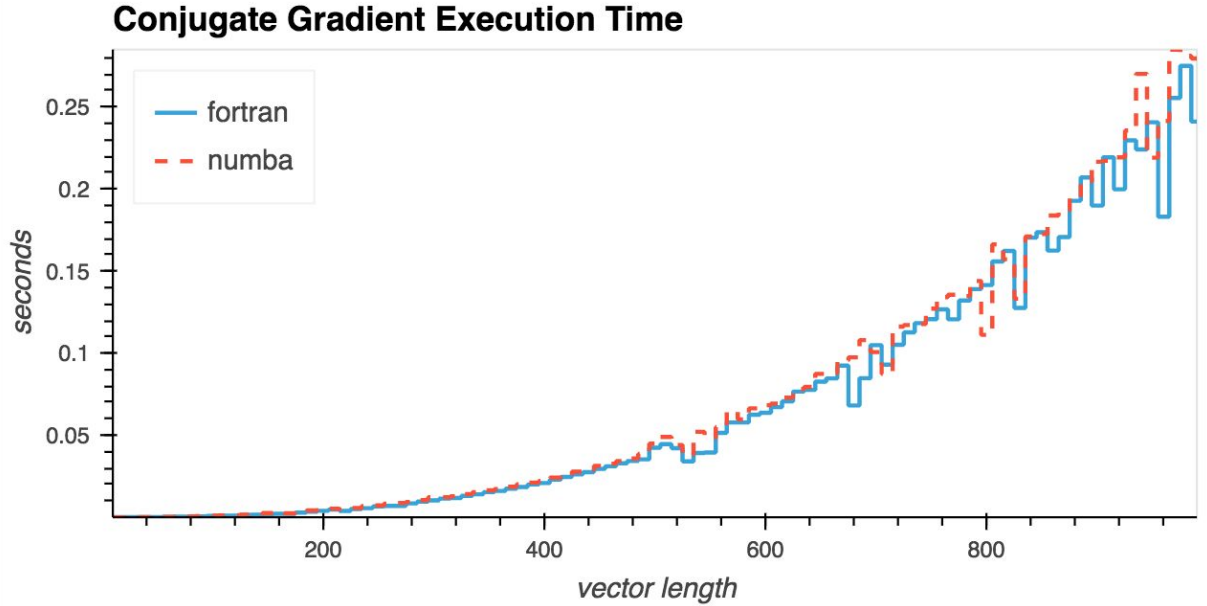
Compiling

Source → Analyze Bytecode and Type Inference → Optimize and Emit Machine Code →

```
@numba.jit
def mvmul(A, x):
    return A * x
```

```
@numba.jit

# --- LINE 6 ---

def mvmul(A, x):

    # --- LINE 7 ---
    #   A = arg(0, name=A)  :: array(float64, 2d, C)
    #   x = arg(1, name=x)  :: array(float64, 1d, C)
    #   $0.3 = arrayexpr(expr=('*', [Var(A, <ipython-input-7-cbc32991a40
0> (7)), Var(x, <ipython-input-7-cbc32991a400> (7))]), ty=array(float64,
2d, C))  :: array(float64, 2d, C)
    #   $0.4 = cast(value=$0.3)  :: array(float64, 2d, C)
    #   return $0.4

    return A * x
```

Performance



CONDA®

- Conda is a cross platform **package** and **environment** management system
- Written and maintained by Anaconda, Inc
- Open Source, BSD licensed
- Packages software written in any language
- Many **Python** and **R** data science, machine learning and AI frameworks
- Available by installing the **Anaconda Distribution** or **Miniconda**

conda package management

- Packages are **binaries**, no compiler or libraries are needed
- Does not require administrator privileges
- Uses a SAT solver for dependency resolution

Package management commands:

- **conda install** : install one or more package(s)
- **conda remove** : remove a package
- **conda update** : update a package
- **conda list** : list the installed packages

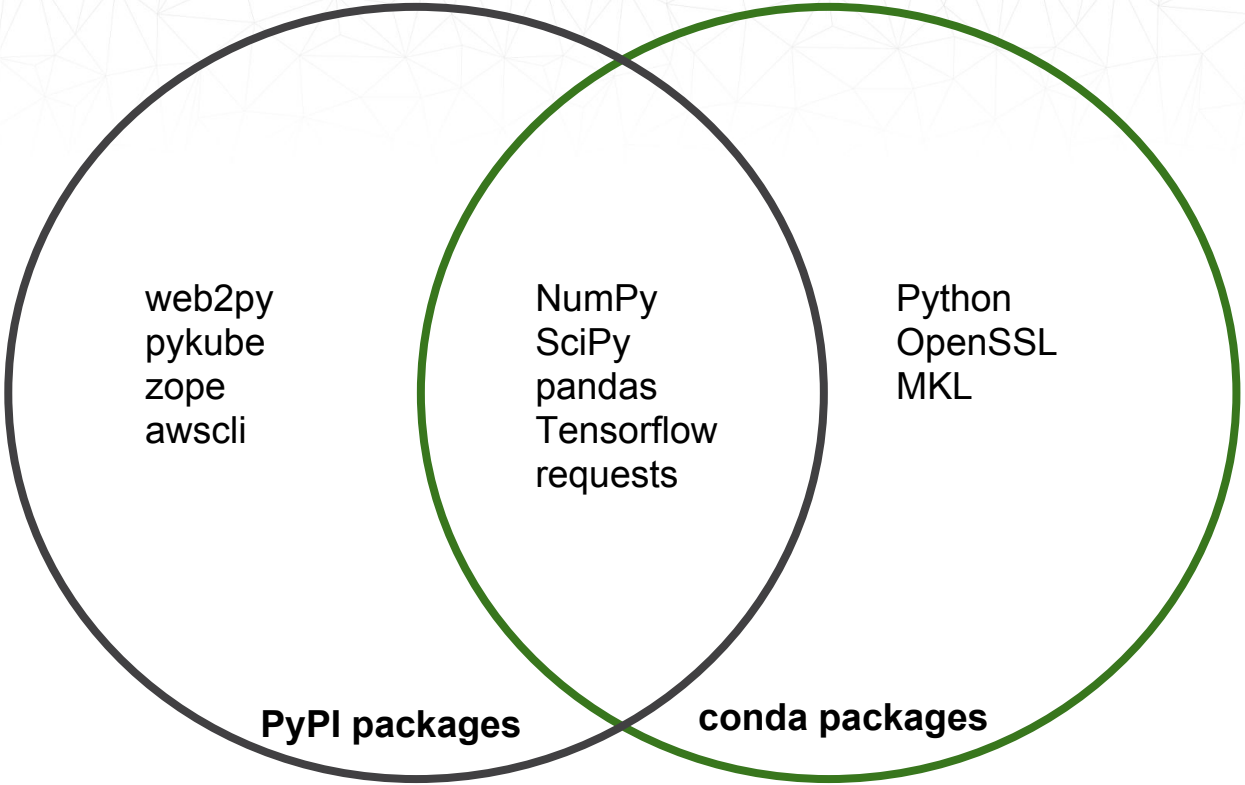
Differences between conda and pip

Pip

- Installs **Python** packages
- **Binaries** (wheels) and source distributions
- virtualenv or venv needed to create isolated environment.
- Resolves dependencies recursively

Conda

- Installs software written in **any language**
- Packages are **binaries**
- Built in support for creating isolated environments
- Uses a SAT solver for dependency resolution


web2py
pykube
zope
awscli
NumPy
SciPy
pandas
Tensorflow
requests
Python
OpenSSL
MKL
PyPI packages
conda packages

conda channels

- Conda packages are provided from different repositories, called **channels**.
- Out of the box, conda installs packages from the "**defaults**" channel.
- Other channels can be enabled to access **additional** collections of packages

Some key channels are:

- **defaults** : packages from Anaconda, Inc
- **intel** : optimized packages from Intel
- **conda-forge** : large community led collection of packages
- **bioconda** : community collection of bioinformatics packages

conda-forge

- Numfocus-affiliated community organization made up of volunteers
- One github repository per recipe
- Fine granularity over permissions
- Heavy use of automation for building, deploying, and updating recipes
- Packages built on public CI services (TravisCI, CircleCI, Appveyor, Azure)
- https://conda-forge.org/

conda environments

Conda can create isolated environments that have their own set of packages.

- **conda create** : create a new conda environment
- **conda activate** : activate a conda environment
- **conda deactivate** : deactivate the current conda environment

Great when you need to work with different versions of a library or application.

Environment specification can be exported to a file and recreated.

Summary

- JIT compiler that translates Python + NumPy into fast machine code
- Can be used for GPU CUDA programming in Python
- http://numba.pydata.org/

- Package and environment manager with a focus on Python/R
- https://conda.io