Building a scalable Python distribution for HEP Analysis

David Lange
August 22, 2017
Concept: Develop and distribute data science oriented Python stack for HEP

• **Reduce startup burden for analysis**: Software environment easily available

• **Provide an(other) option for a standard software environment**: Make it easy to use common software within an analysis group

• **Ease use of distributed computing systems**: Simplify distribution of analysis on the GRID
**Initial motivations for Python in CMS and CMSSW**

---

**Job configuration and control of CMSSW**

```python
import FWCore.ParameterSet.Config as cms

from Configuration.StandardSequences.Eras import eras

# import of standard configurations
process.load('Configuration.StandardSequences.Services_cff')
process.load('SimGeneral.HepPDTESSource.pythiapdt_cfi')

process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(10)
)

# Input source
process.source = cms.Source("PoolSource",
    fileNames = cms.untracked.vstring('file:step2.root'),
    secondaryFileNames = cms.untracked.vstring()
)
```

---

**Analysis: FWLite+PyROOT**

```python
import ROOT
from DataFormats.FWLite import Events, Handle

events = Events('ZmumuPatTuple.root')

# loop over events
for event in events:
    ......
```

~2008
New motivations: Python has evolved to a standard data science platform [inside and outside of HEP]

- Massive increase in interest in applications of data analytics and machine learning techniques to HEP
- Easier for everyone in the experiment to have a pre-built software stack to use as one option to use for analysis
  - And to be able to modify this software stack to suit individual needs
- Anticipate the transition from development to production use of these tools

- Our focus
  - Tool interoperability. E.g., conversion into or out of ROOT into other data formats for analysis (e.g., for Spark, etc)
  - Data science tools developed in python ecosystem
Conflicting goals of production software distributions and user analysis developments?

Experiment Software Distributions

Analysis R&D
Our approach

- We adopted **PIP and PYPI** distributions of packages
  - Simplify, standardize and modernize python package distribution in CMSSW
  - Use source distributions when available (which is almost always) so that tool builds and distributions are consistent with the rest of CMSSW

- Priorities for adding specific packages driven by suggestions and interests from users (and what packages we see the community using)
Adopting PIP simplifies the implementation of a new python package in our build system

---

### RPM external py2-numba 0.33.0

```
## INITENV +PATH PYTHONPATH %\{i\}/\{_PACKAGES\}
```

Requires: py2-funcsigs py2-enum34 py2-six
Requires: py2-singledispatch py2-llvmlite py2-numpy

### IMPORT build-with-pip

---

- Explicitly indicate dependent packages. These dependencies can sometimes be derived from PIP (e.g., they can be automatically determined).
  - However complex packages often require understanding of setup options.
What is inside build-with-pip?

File: with-with-pip

%if "{%{pip_name:set}" != "set"
%define pip_name %{echo %n \ cut -f2-5
%endif

%if "%{?PipDownloadOptions:set}" != "set"
%define PipDownloadOptions --no-deps%=20--no-binary%=3D:all:
%endif

%if "{%{PipBuildOptions:set}" != "set"
%define PipBuildOptions --no-deps
%endif

Source:
pip://{%pip_name}/%{realversion}?pip_options=%{PipDownloadOptions}&output=/source.tar.gz

Requires: python
BuildRequires: py2-pip

%prep
%build
mkdir -p %{
mkdir -p %{
 tar xzf %{_sourcedir}/source.tar.gz

%{?PipPreBuild:%PipPreBuild}
export PIPFILE=`cat files.list`
export PYTHONUSERBASE=%{

pip install --user -v %{PipBuildOptions} $PIPF

%install
%{?PipPostBuild:%PipPostBuild}
Users can now use PIP to explore CMSSW python stack

dlage> pip list
appdirs (1.4.3)
bleach (2.0.0)
Bottleneck (1.2.1)
certifi (2017.4.17)
chardet (3.0.4)
click (6.7)
climate (0.4.6)
configparser (3.5.0)
cycler (0.10.0)
Cython (0.22)
decorator (4.0.11)
deepdish (0.3.4)
docopt (0.6.2)
downhill (0.4.0)
............
VirtualEnv helps users be more agile than CMSSW releases when they need to be

```
dlang> pip list | grep Theano
Theano (0.8.2)

dlang> python
Python 2.7.11 (default, Apr 28 2017, 13:50:27)
[GCC 6.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import theano
>>> print theano.__version__
0.8.2

dlang> virtualenv updateTheano
New python executable in /build/dlang/CMSSW_9_3_X_2017-08-07-2300/updateTheano/bin/python
Installing setuptools, pip, wheel...done.
dlang> source updateTheano/bin/activate.csh

dlang> setenv PYTHONPATH $PWD/updateTheano/lib/python2.7/site-packages:
```

Example: I want to upgrade Theano

CMS distributes 0.8.2

To get started: Set up and activate virtualenv
VirtualEnv helps users be more agile than CMSSW releases when they need to be

```
[updateTheano] dlange> pip install Theano==0.9.0
Collecting Theano==0.9.0
Requirement already satisfied: scipy>=0.14
Requirement already satisfied: numpy>=1.9.1
Installing collected packages: Theano
  Found existing installation: Theano 0.8.2
  Not uninstalling theano at /cvmfs/cms-ib.cern.ch/......... /lib/python2.7/site-packages, outside environment
  /build/dlage/CMSSW_9_3_X_2017-08-07-2300/updateTheano
Successfully installed Theano-0.9.0

[updateTheano] dlange> pip list | grep Theano
Theano (0.9.0)

[updateTheano] dlange> python
Python 2.7.11 (default, Apr 28 2017, 13:50:27)
>>> import theano
>>> print theano.__version__
0.9.0
```
Aside: Python can interact with CMSSW analysis jobs (and data structures)

Maintain processing control after job configuration is complete (instead of launching an executable)

```python
e = CmsRun(process)
e.run()
```

- Enables new possibilities. E.g., Allocate memory in numpy arrays, fill and/or process data within experiment framework applications

Iterative alignment or calibration techniques

Ease interface to spark (e.g.) analysis

https://github.com/diana-hep/c2numpy/.../commonblock-demo.ipynb
Tools we integrated: HEP developments (Incomplete list)

histogram

rootpy

root_pandas

xrootdpyfs
Tools we integrated: Jupyter stack

• Common use case: Do python analysis in CMSSW software stack from laptop

https://github.com/cms-sw/cmssw/...pcsfVerify.ipynb
Tools we integrated: Data science (incomplete list)

- HDF5 for Python
- matplotlib
- ThriftPy
- Bottleneck
- Numba
- PyTables
- NumPy
- uncertainties
- pandas
- SciPy library
- DOWNHILL
- SymPy
Tools we integrated: Machine learning (incomplete list)

- Keras
- Theano
- TensorFlow
- scikit-learn
But my laptop does not run CentOS7. What can I do?

- We defined a simple wrapper package that provides that versions of Python tools corresponding to current CMSSW distribution
  - Preliminary version of this is available now.
- We plan to update this as we deploy and validate significant changes to the python tools in CMSSW
  - Thus, it won’t the latest and greatest set of tools available on PYPI, but hopefully a self consistent version.

```
dlange> pip install pyCMSSW
```
Conclusion and Outlook

• Attempt to develop and distribute a Python software stack to match wants and needs of data analysis in HEP.
  • We believe this approach can simultaneously fit the needs of production environments and user agility needs.
  • CMS software stack has been used for this work, but nothing is CMS specific (aside from SPEC file implementations, but these are nearly trivial)

• Working to develop performance monitoring unit tests

• Try it out – we would like to hear how to improve what we’ve done
Backup information
Example: Converting ROOT data to the data format you need for ML