Netherlands eScience Center

... and the physics community

Daniela Remenska
eScience engineer

netherlands eScience center

by SURF & NWO

HSF workshop
Orsay, May 2016

Netherlands eScience Center = digitally enhanced Science

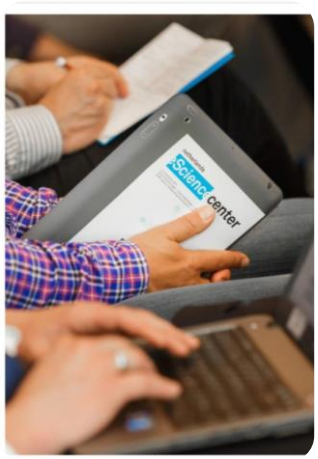
SURF

ICT infrastructure

NWO

Scientific research

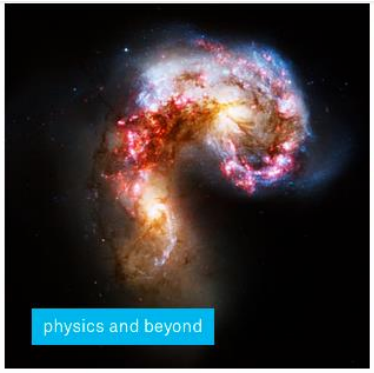
NLeSC in summary



- **Our core technical expertise areas:**
  - **Optimized Data Handling, Big Data Analytics, Efficient computing**
- **We cover “all” of research:**
  - **Environment & Sustainability, Life Sciences, Humanities, Social Sciences, Physics & Beyond**
- **We are all about open-source & open access!**
  - **Re-use community solutions, enforce good SE practices**
- **We parallel the HEP Software Foundation in a way**

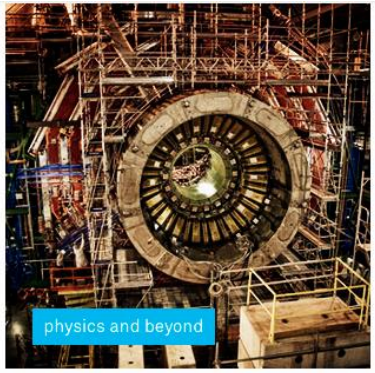
iDark

The intelligent Dark Matter Survey



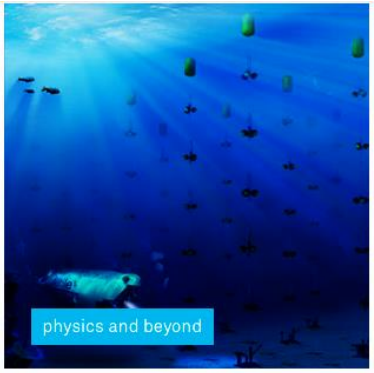
AA-ALERT

Access and Acceleration of the Apertif Legacy Exploration of the Radio Transient Sky



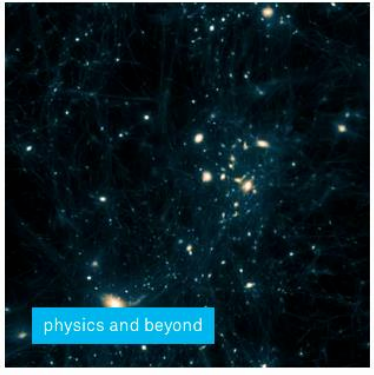
Automated Parallel Calculation of Collaborative Statistical Models

Large scale statistical data analysis in particle physics



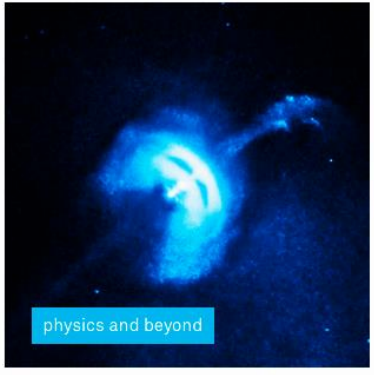
Real-time detection of neutrinos from the distant Universe

Observing processes that are inaccessible to optical telescopes



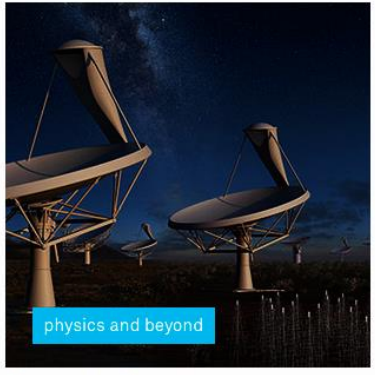
Giving Pandas a ROOT to Chew on

Modern Big Data front and backends in the hunt for Dark Matter



Compressing the sky into a large collection of statistical models

Optimized data handling for observations in astronomy



Beyond the Data Explosion

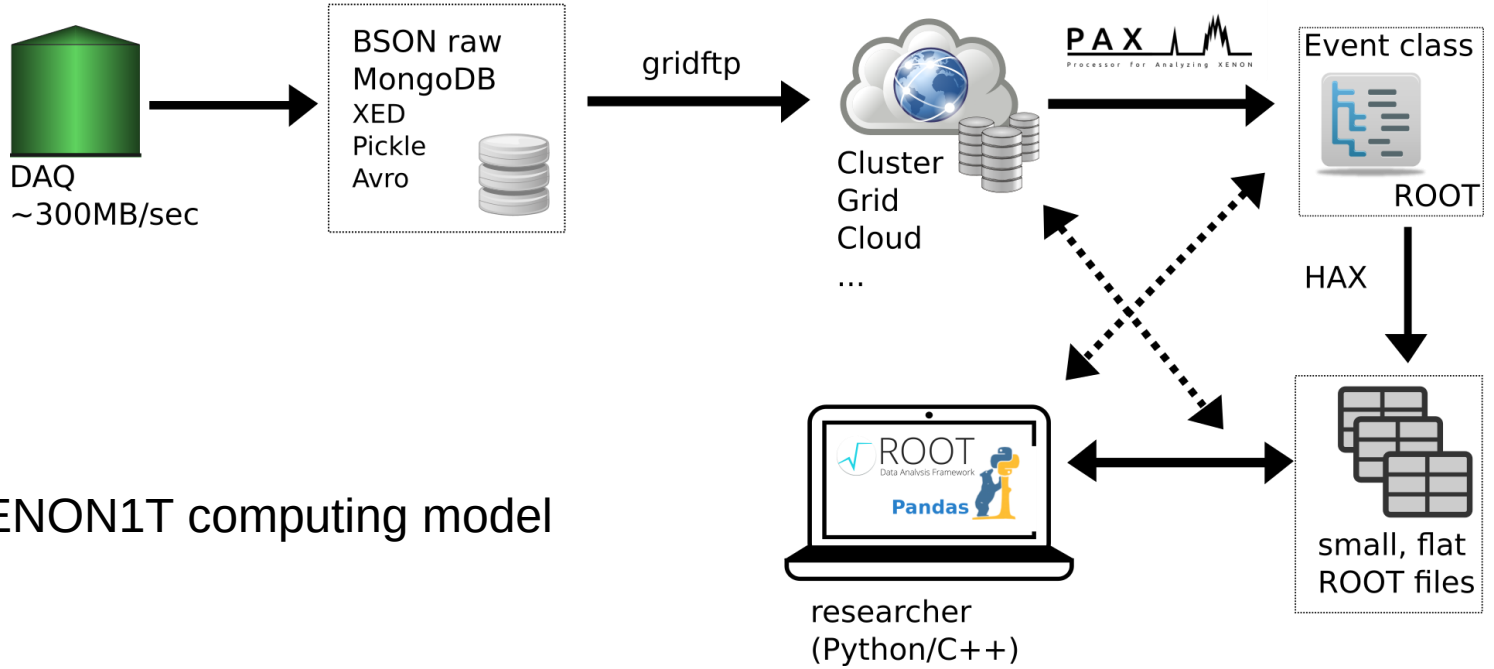
An eScience infrastructure for huge interferometric datasets

PROMIMOOC

Process mining for multi-objective online control

Giving Pandas ROOT to Chew On

Modern Big Data front and backends in the hunt for Dark Matter



XENON1T computing model

In [9]:
```
# I also load the S2 width, for use in the plot in the appendix
class NewDriftTime(hax.minitrees.TreeMaker):
    __version__ = '0.0.1'
    extra_branches = ['peaks.index_of_maximum', 'peaks.hit_time_std']

    def extract_data(self, event):
        if not len(event.interactions):
            return dict()
        s1 = event.peaks[event.interactions[0].s1]
        s2 = event.peaks[event.interactions[0].s2]
        return dict(drift_time_2=(s2.index_of_maximum - s1.index_of_maximum) * 10,
                    s2_width=s2.hit_time_std)

data = hax.minitrees.load('xe100_111110_1127', ['Basics', NewDriftTime])
```

```
WARNING:hax.paxroot:Root file /mnt/xecluster/archive_lngs/common/PaxReprocessed_9/good/xe100_111110_1127.root does not include pax event class. Normal for pax < 4.5.Falling back to event class for pax 430
WARNING:hax.paxroot:Root file /mnt/xecluster/archive_lngs/common/PaxReprocessed_9/good/xe100_111110_1127.root does not include pax event class. Normal for pax < 4.5.Falling back to event class for pax 430

Created minitree Basics for dataset xe100_111110_1127
Created minitree NewDriftTime for dataset xe100_111110_1127
```

`data` is now a pandas DataFrame containing basic info (see below) for all the events. For more details, and instructions on how to select your own variables, see the hax tutorial. Here is a look inside the data we just loaded:

In [10]:
```
data.head()
```

Out[10]:

	index	cs1	cs2	dataset_number	drift_time	event_number	event_time	largest_coi
0	0	533.132248	117434.248258	1111101127	102700.195312	0	1320920877023216128	0
1	1	NaN	NaN	1111101127	NaN	1	1320920877024649984	0
2	2	916.212603	89846.426385	1111101127	159293.125000	2	1320920877028420096	0
3	3	NaN	NaN	1111101127	NaN	3	1320920877029669120	0
4	4	337.785598	32902.233486	1111101127	160682.093750	4	1320920877032971008	0

5 rows × 22 columns

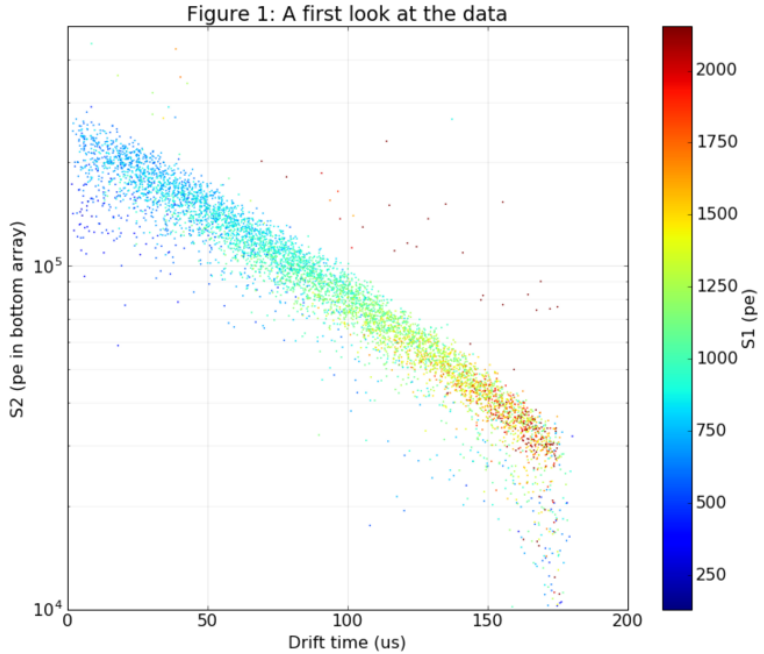
We'd like to have the drift time in us, and the S2 area on the bottom. Let's compute these from the basic variables:

In [12]:
```
drift_time = data['drift_time_2'].values / units.us
s2_bottom = data['s2'].values * (1 - data['s2_area_fraction_top'].values)
```

Exploratory analysis

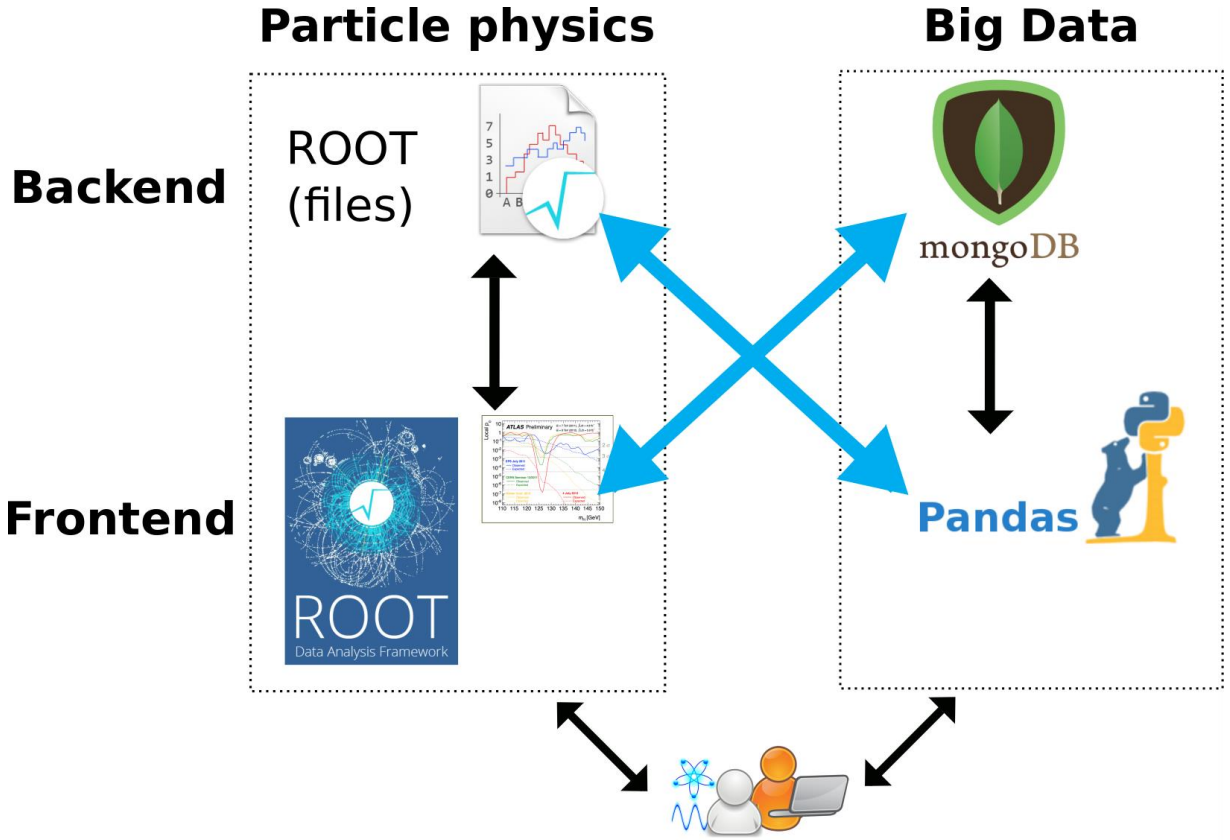
Before fitting, let's have a first look at the data we have:

In [13]:
```
plt.scatter(drift_time, s2_bottom,
            c=(data['s1'].values), vmax=2 * data['s1'].mean(),
            marker='.', edgecolors='none', s=10)
plt.yscale('log')
cb = plt.colorbar(label='S1 (pe)')
plt.grid(which='minor', linestyle='-', alpha=0.08)
plt.grid(which='major', linestyle='-', alpha=0.15)
plt.xlim(0, 200)
plt.ylim(1e4, 5e5)
plt.xlabel('Drift time (us)')
plt.ylabel('S2 (pe in bottom array)')
plt.title('Figure 1: A first look at the data')
plt.show()
```



Giving Pandas ROOT to Chew On

Modern Big Data front and backends in the hunt for Dark Matter



Giving Pandas ROOT to Chew On

Modern Big Data front and backends in the hunt for Dark Matter

XENON Dark Matter Project

? **root_pandas: ROOT I/O for Pandas**	*pandas* should read ROOT files
 **rootpy: Pythonic ROOT**	Truly “Pythonic” ROOT interface
PyROOT: A Python -- ROOT Bridge	Python bindings for ROOT
 ROOT Data Analysis Framework	C++

```python
from rootpy.tree import Tree, TreeModel
from rootpy.tree import FloatCol, IntCol
from rootpy.tree import FloatArrayCol, CharCol
from rootpy.io import root_open
from random import gauss, choice

f = root_open("test.root", "recreate")
# define the model
class Event(TreeModel):
    s = CharCol()
    x = FloatCol(default = 'nan')
    y = IntCol(default = 0)
    f = FloatArrayCol(5)
tree = Tree("test", model=Event)
# fill the tree
for i in range(5):
    tree.s = ord(choice(ascii_letters))
    tree.x = gauss(.5, 1.)
    tree.y = i
    for j in range(5):
        tree.f[j] = gauss(-2, 5)
    tree.fill()
tree.write()
f.close()
```

Listing 1: Creating tree models with rootpy

```python
class ReconstructedPosition(object):
    x = float('nan')
    y = float('nan')
    ....

class Peak(object):
    area = 0.0
    detector = 'ptc'
    ...
    rec_positions = list(ReconstructedPosition)

class Hit(object):
    channel = 0
    center = 0.0
    ...
class Event(object):
    event_number = 0
    dataset_name = 'Unknown'
    ...
    peaks = list(Peak)
    hits = np.array([], dtype=Hit.get_dtype())
```

Listing 2: The pax Event model

ROOT in the Anaconda Cloud

- **goal**: `conda install root={5,6} python={2,3}`
  - no-sudo
  - cross-platform

- **Conda: packaging, dependency & environment management system**
  - Not really python-specific

conda-recipes / python / gensim / meta.yaml

```
package:
  name: gensim
  version: 0.8.8

source:
  fn: gensim-0.8.8.tar.gz [py2k]
  url: https://pypi.python.org/packages/source/g/gensim/gensim-0.8.8.tar.gz
  md5: 39b47095185f05a01b83ebf1a6748953 [py2k]

#  patches:
   # List any patch files here
   # - fix.patch

build:
  number: 1

requirements:
  build:
    - python
    - setuptools
    - scipy

  run:
    - python
    - scipy

test:
  # Python imports
  imports:
    - gensim.similarities
    - gensim.test
    - gensim.corpora
    - gensim.models
about:
  home: http://radimrehurek.com/gensim
  license: GNU Library or Lesser General Public License (LGPL)
```

conda-recipes / python / gensim / build.sh

```
#!/bin/bash

$PYTHON setup.py install
```

Portable ROOT conda binaries

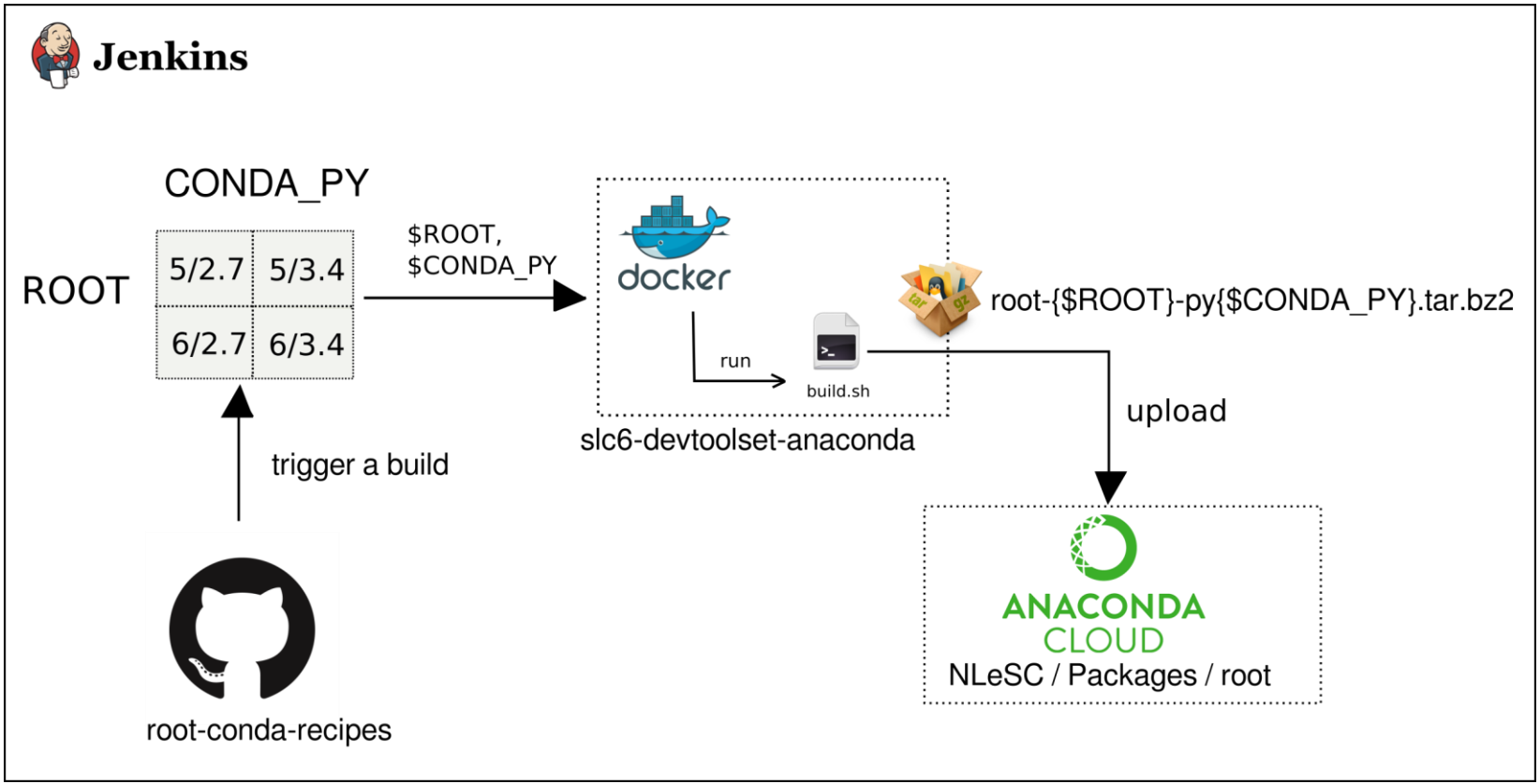
- dynamic dependencies on GCC/glibc
- ROOT 6 needs GCC 4.8 or newer
- should work on older Linux distributions
  - preferably the same binary (one size fits all)
  - SLC 6 + CERN Developer Toolset (v2) makes it possible to have fresh compiler with old glibc (2.12)
- Please give it a try and report problems

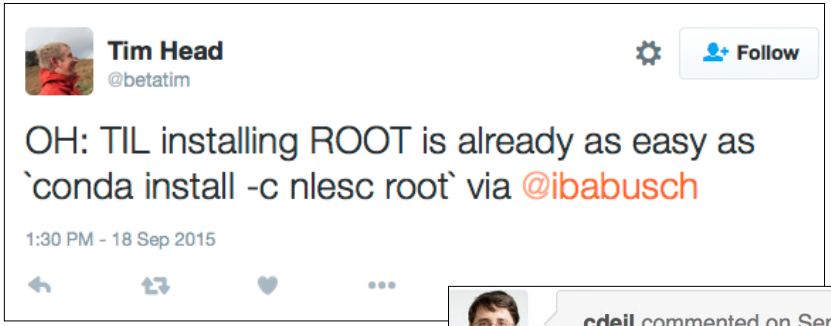
```
conda install -c nlesc root
```

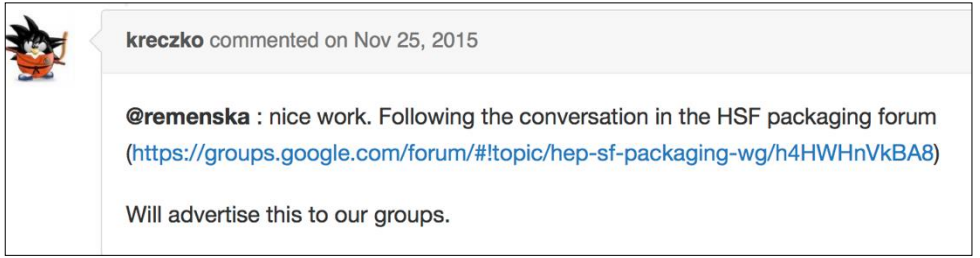
https://github.com/nlesc/root-conda-recipes

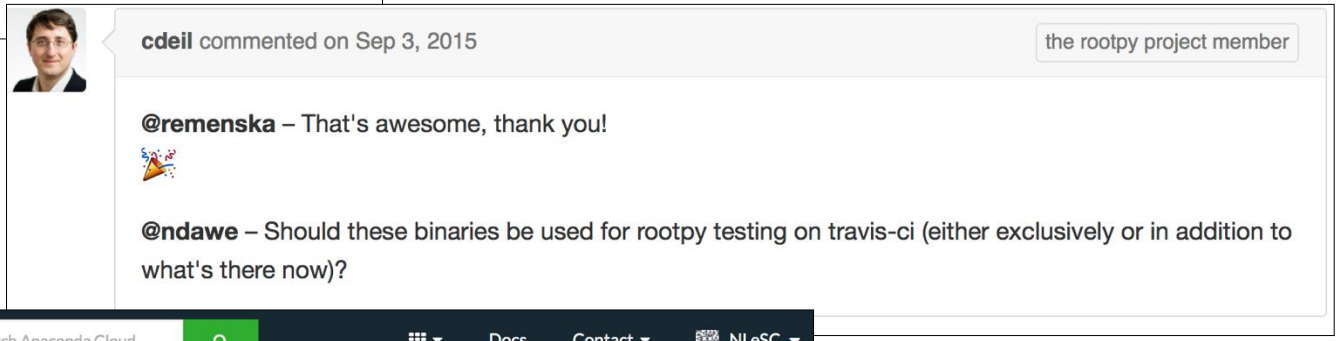
Giving Pandas ROOT to Chew On

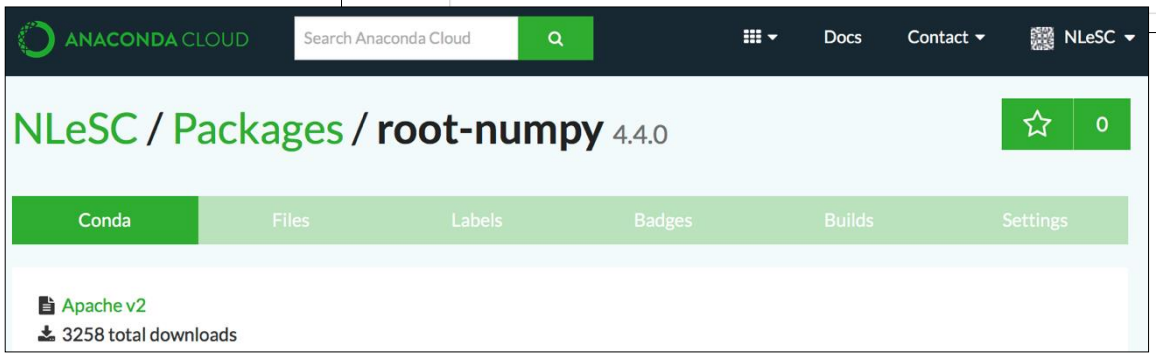
Modern Big Data front and backends in the hunt for Dark Matter




Tim Head
@betatim
Follow
OH: TIL installing ROOT is already as easy as `conda install -c nlesc root` via @ibabusch
1:30 PM - 18 Sep 2015

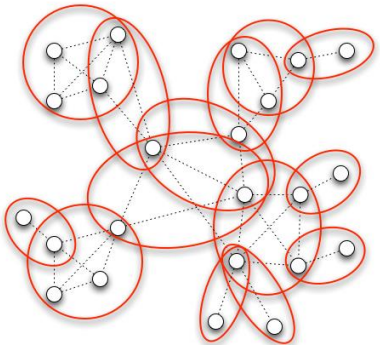

kreczko commented on Nov 25, 2015
@remenska : nice work. Following the conversation in the HSF packaging forum (https://groups.google.com/forum/#!topic/hep-sf-packaging-wg/h4HWHnVkBA8)
Will advertise this to our groups.


cdeil commented on Sep 3, 2015
the rootpy project member
@remenska – That's awesome, thank you!
@ndawe – Should these binaries be used for rootpy testing on travis-ci (either exclusively or in addition to what's there now)?

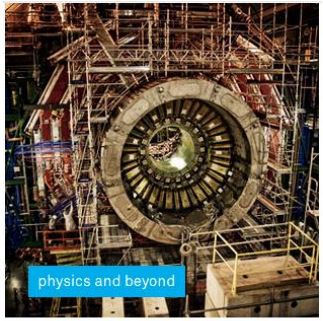

ANACONDA CLOUD
Search Anaconda Cloud
Docs
Contact
NLeSC
NLeSC / Packages / root-numpy 4.4.0
0
Conda
Files
Labels
Badges
Builds
Settings
Apache v2
3258 total downloads

Real-time detection of neutrinos from the distant Universe

- **Trigger based purely on L0 hits**
  - Challenging: amount of data+combinatorics…
  - Can we compare each hit with all other hits?
- **Real-time online event reconstruction**
- **Clique algorithm on GPU: find largest causally related set in a group**



… and more HEP NLeSC projects



Automated Parallel Calculation of Collaborative Statistical Models

Large scale statistical data analysis in particle physics

- **RooFit: statistical models of measurements performed by independent teams combined a posteriori without loss of detail**
- **scaling issues**
- **parallel algorithms / new data structures needed**

- **Combine the worldwide data within the most general models of Dark Matter**
- **algorithms to find (tiny, fragmented) solution areas in large multidimensional parameter spaces**
- **make a (web-accessible) largely automated "DM model" database**

iDark

The intelligent Dark Matter Survey