# ARCONS: software to convert photons to science

Chris Stoughton

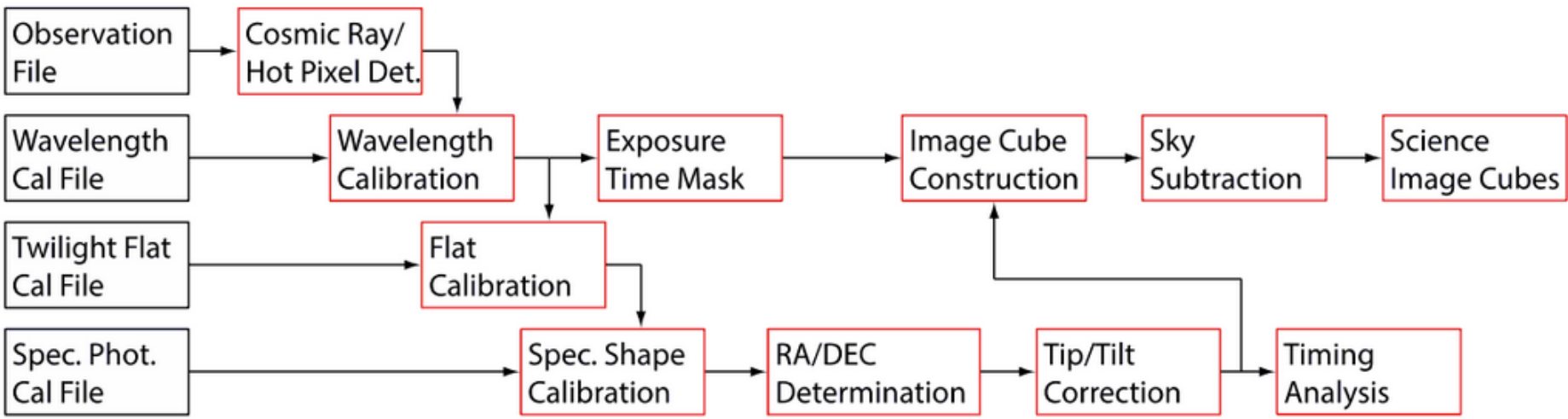
Fermi National Accelerator Laboratory
Center for Particle Astrophysics

*stoughto@fnal.gov*

March 26, 2031

# Outline

- Photon Packet $\rightarrow$ Data on Disk
- ARCONS-pipeline
- I just want to see a picture!
- Conclusion



# Photon Packet → Data on Disk (1)

## photon packet

A photon packet is one 64-bit word, allocated as:

- 8 bits – channel
- 12 bits – Parabola Fit Peak Height
- 12 bits – Sampled Peak Height
- 12 bits – low pass filter baseline
- 20 bits – microsecond timestamp

For each pixel, a variable-length list of photon packets is written each second.
A 2-d array of Roach/channel IDs defines where the pixels are on the array.

# Photon Packet → Data on Disk (2)

This is all written to an hdf5 file:

```
$ h5dump -n obs_20121211-134003.h5
HDF5 "obs_20121211-134003.h5" {
FILE_CONTENTS {
 group      /
 group      /beammap
 dataset    /beammap/atten
 dataset    /beammap/beamimage
 dataset    /beammap/resfreq
 group      /header
 dataset    /header/header
 group      /r0
 group      /r0/p0
 dataset    /r0/p0/t1355233205
 group      /r0/p1
 dataset    /r0/p1/t1355233205
...
```

# Photon Packet → Data on Disk (3)

Identify a file with:

- run – such as PAL2012
- date – such as 20121210
- flavor – such as obs, cal, calsol, timeMask, ...
- tstamp – such as 20121211-134003

Files written during observing are under $MKID_DATA_DIR and generated files are under $INTERM_DIR.

There are also some log files written by the data acquisition system which are used to fix known issues. Matt Strader knows all about these!

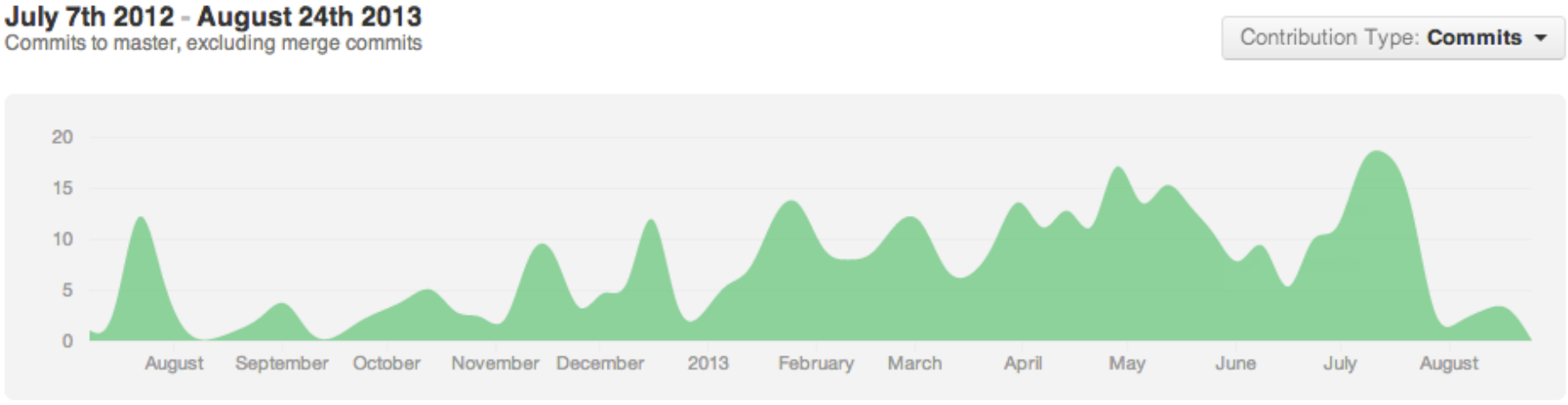
For the PAL2012 data run, 400 GByte raw data written.

# ARCONS-pipeline (1)

Hosted at (private) github repository
https://github.com/bmazin/ARCONS-pipeline

730 commits, 9 contributors

```
find . -name '*.py' | xargs wc -l
...
39884 total
```



# ARCONS-pipeline (2)

# ARCONS-pipeline (3)

PREREQUISITES (from README.md)

Enthought Python Distribution (EPD) 7.3
(http://www.enthought.com/products/epd.php)

PyEphem (http://rhodesmill.org/pyephem/)

PyGuide
(http://www.astro.washington.edu/users/rowen/PyGuide/Manual.html)

(you can check if you have them with help('modules') within the (i)python interpreter)

If you are having troubles with PyTables (which you shouldn't since it is built into EPD), see http://www.tumblr.com/tagged/pytables and instructions therein for Mac.

Another package you will need is interval. It depends on crlibm

# ARCONS-pipeline (4)

General purpose:

```
/headers contains standard definitions

/params contains files that provide inputs to the pipeline

/utils contains commonly used functions

/examples contains simple examples to show how to use
the software
```

# ARCONS-pipeline (5)

Pipeline components:

```
/cosmic contains a module for cosmic ray cleaning

/wavelengthcal contains a module to do
wavelength calibration

/flatcal contains a module for normalizing the
QE as a function of wavelength between pixels

/fluxcal contains a module to calibrating the system
compared to a standard star

/astrometry contains a module to determine the
RA and DEC of each photon

/skysub contains a module to subtract the sky background
```

# ARCONS-pipeline (6)

`/imagecube contains a module to generate a FITS image cube based on an observations (no timing info)`

`/legacy contains ARCONS analysis code that predates this repository`

`/quicklook contains tools for quickly looking at ARCONS h5 files`

`/beammap contains tools for creating, viewing, and modifying beam maps`

`/hotpix contains tools for finding location and time of hot (and possibly 'cold') pixels`

Each directory contains a /test subdirectory with unittests

# Make a Picture! (1)

```
#
# Look at a .h5 file from the Palomar 2011 run
# Make a FITS image of the photons
# Use the /beammap/beamimage information to gt the list of
#
# Set the environment variable MKID_DATA_DIR to point to th
#
# Example use:
#
# $ export MKID_DATA_DIR=/Volumes/data/Palomar2011/Pal2011(
# python palomar-2011.py obs_20110729-151443.h5

import sys, os
import tables
import pyfits
import numpy as np
```

# Make a Picture! (2)

```
if (len(sys.argv) < 2):
    print "Usage:  ",sys.argv[0]," hdf5FileName"
    sys.exit(1)

# make the full file name by joining the input name to the
hdf5FileName = sys.argv[1]
dataDir = os.getenv('MKID_DATA_DIR','.')
hdf5FullFileName = os.path.join(dataDir,hdf5FileName)
print "full file name is ",hdf5FullFileName
if (not os.path.exists(hdf5FullFileName)):
    print "file does not exist: ",hdf5FullFileName
    sys.exit(1)

# open the actual file.  This might take a while
fid = tables.openFile(hdf5FullFileName, mode='r')
```

# Make a Picture! (3)

```
# get the beam image.  This is a 2d array of roach board/pi
beamImage = fid.getNode("/beammap/beamimage")

# Make a 2d array to hold the image
shape = beamImage.shape
```

# Make a Picture! (4)

```
pixels = np.zeros(shape,dtype=np.uint32)
# count the total number of photons in the file
nPhoton = 0
iRow = -1
for rows in beamImage:
    iRow += 1
    iCol = -1
    for pixel in rows:
        iCol += 1
        sum = 0
        for sec in fid.getNode(pixel):
            for packet in sec:
                # here is the 64-bit number.
                packet = int(packet)
                nPhoton += 1
                sum += 1
        pixels[iRow][iCol] = sum
```

# Make a Picture! (5)

```
print "nPhoton=",nPhoton
hdu = pyfits.PrimaryHDU(pixels)
hdu.writeto('new.fits')
```

# Conclusion

It turns out that this is NOT the best way to access information.

In the util package, there is ObsFile, which does much of this, and adds corrections, such as hot pixel detection, DA timing issues, wavelength calibration, etc.

Julian is working on integrating all calibrations and corrections to yield a "photon list."

Within a week you can be productive in this new environment. Lots of experience to leverage. The group is very responsive to question, bugs, and requests!