@GridPP @twhyntie

GridPP and DIRAC: an example with CERN@school

T. Whyntie*, †

* *Queen Mary University of London*; † *Langton Star Centre*

GridPP
UK Computing for Particle Physics

Queen Mary
University of London

Overview of the talk

- Introduction
- Setup
- Uploading datasets
- Processing datasets
- Basic analysis
- Observations and further work

Introduction

- DIRAC – Distributed Infrastructure with Remote Agent Control:
  - *http://diracgrid.org*
  - *http://github.com/DIRACGrid/DIRAC*
  - *Imperial instance – see GridPP wiki page for details.*
- CERN@school – bringing CERN into the classroom:
  - *http://cernatschool.web.cern.ch*
  - *Flagship small Virtual Organisation (VO) for GridPP engagement activity;*
  - *Currently supported by QMUL, Glasgow, Liverpool, Birmingham – thanks!*
  - *Technology demonstration with CERN@school data and software – already done with CVMFS*

Introduction

- The goal of this work – demonstrate capabilities of DIRAC:
  - *Job management for small VOs:*
    - Command line, web portal, and Python API;
    - Integrates with Ganga (not covered here, working with Mark Slater on this);
    - Replacement for LCG WMS?
  - *Data management for small VOs:*
    - Command line, web portal and Python API for file management;
    - The DIRAC File Catalog (DFC) – replacement for LFC? (Compatible with LFC);
    - Replica management functionality (not covered here);
  - *Metadata management for small VOs:*
    - KEY FUNCTIONALITY – missing from out-of-the-box LCG toolkit;
    - An alternative to AMGA etc. rolled into job and data management;
    - The main focus of the work presented here.

Using DIRAC - overview

- Command line interface:
  - *Pretty comprehensive;*
  - *Useful for manual work.*
- Web portal:
  - *Nicest feature IMHO – easy to track jobs;*
  - *Can even submit jobs once proxy generated. Browser-loaded certificates.*
- Python API:
  - *For heavy lifting/production work.*
  - *Not well documented (yet) but:*
  - *http://github.com/DIRACGrid/DIRAC*

The CERN@school example workflow

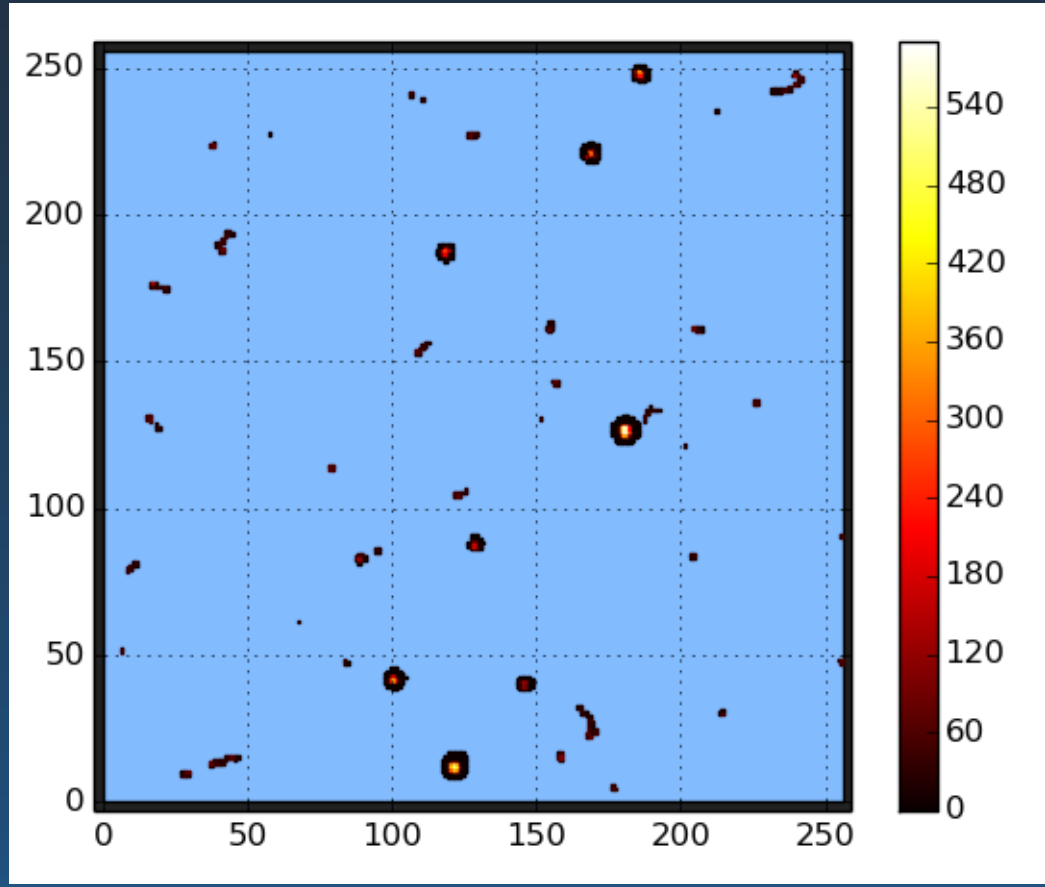
- Upload a dataset
  - *Raw data from the CERN@school detectors;*
  - *Add metadata to the dataset.*
- Process that dataset:
  - *Select data files of interest using metadata query;*
  - *Run CERN@school software via CVMFS on selected data;*
  - *Write the output to a selected storage element;*
  - *Add metadata to the generated data.*
- Run an analysis on the processed data:
  - *Select data of interest using a metadata query;*
  - *Retrieve output from the grid based on the selection.*

Setup and installation

- DIRAC:
  - *See the GridPP wiki for getting started with DIRAC;*
  - *Setup up environment: . bashrc*
  - *Generate a DIRAC proxy: dirac-proxy-init –g cernatschool_user –M*
- GridPP demonstration code:
  - *git clone https://github.com/GridPP/dirac-getting-started.git*
  - *All the code is there – fully working example & test dataset.*
- Huge thanks to Janusz (IC), CJW (QMUL), Sam S (GLA) for help getting this working!

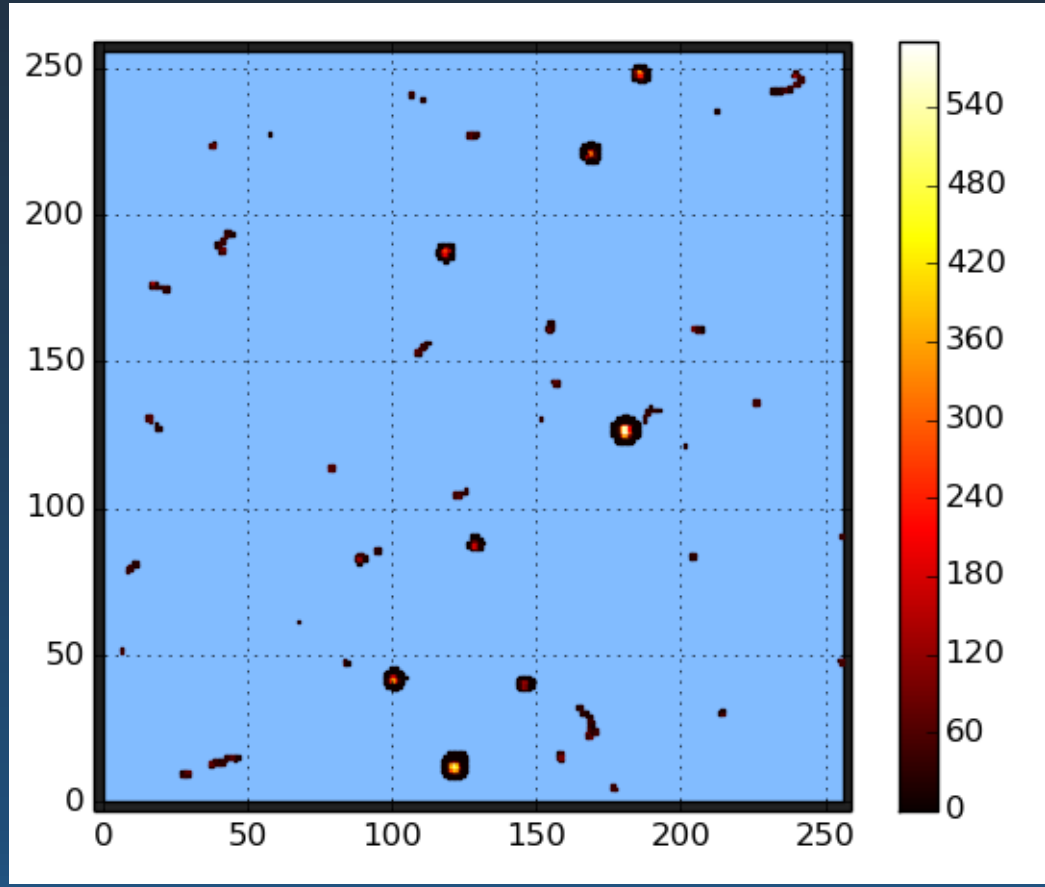
Uploading a dataset

- The data – CERN@school frame:
  - *256 x 256 grid of pixels from the detector;*
  - *Pixels visualise ionising radiation.*
- All done with Python API grid job:
  - *python upload_frames.py*
- Need to specify:
  - *Folder of the input dataset;*
  - *A local output folder;*
  - *Job details;*
  - *Location on DFC for the data.*



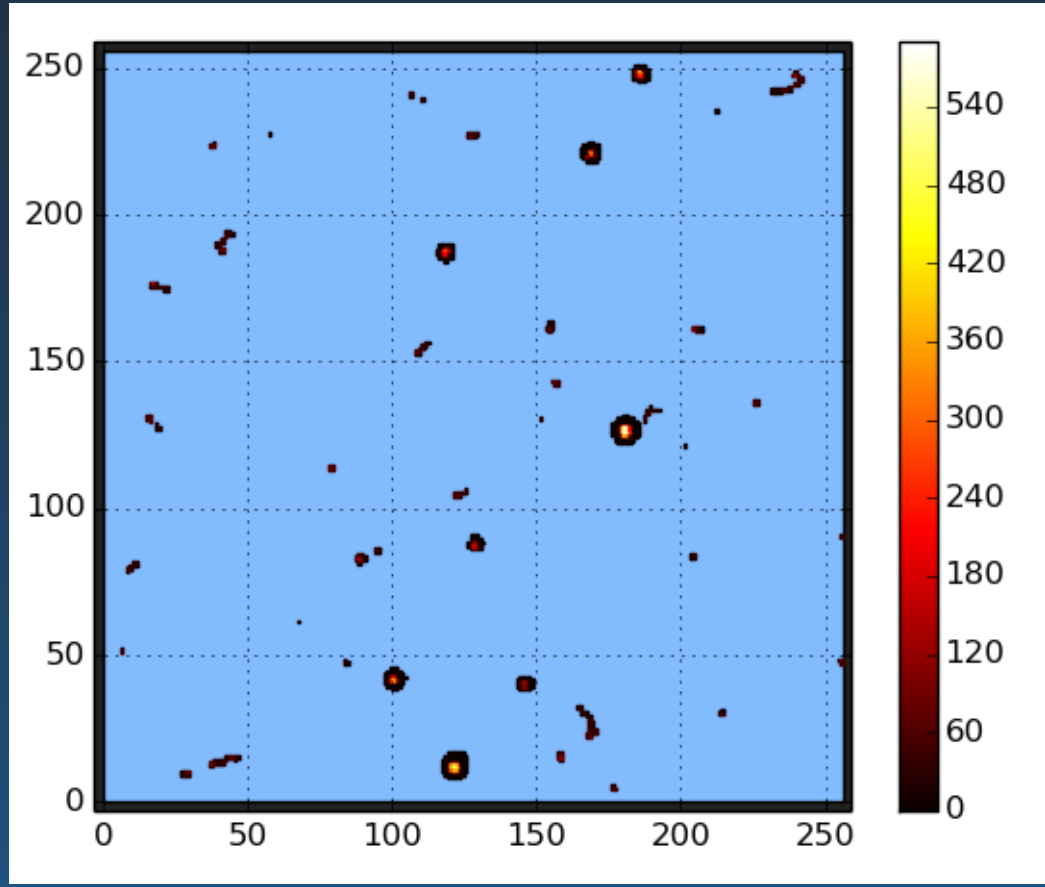
Adding metadata to the dataset

- Metadata fields added via the DFC:
  - *dirac-dms-filecatalog-cli*
  - *DFC:>meta index –f start_time int*
- Metadata is added *after* uploading:
  - *Data registered in DFC after job finishes…*
  - *Dataset metadata stored in local JSON;*
  - *Added via Python script post-job:*
  - *python add_frame_metadata.py*
  - *Does not require a separate job; all done via the Python FileCatalogClient.*



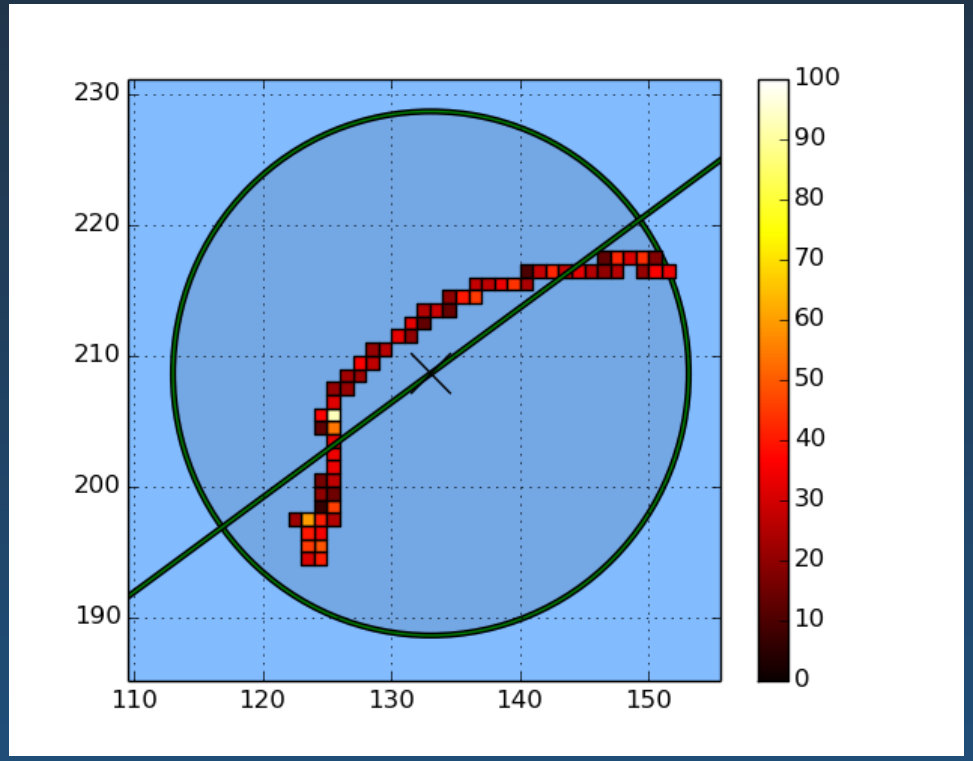
Querying the metadata

- Performed with the FileCatalogClient:
  - *Return a list of frames with search criteria defined in a JSON file;*
  - *python perform_frame_query.py*
  - *Again, instant feedback without a job;*
- Results can be used as:
  - *Input to another job (via API)*
  - *Input to analysis in its own right.*



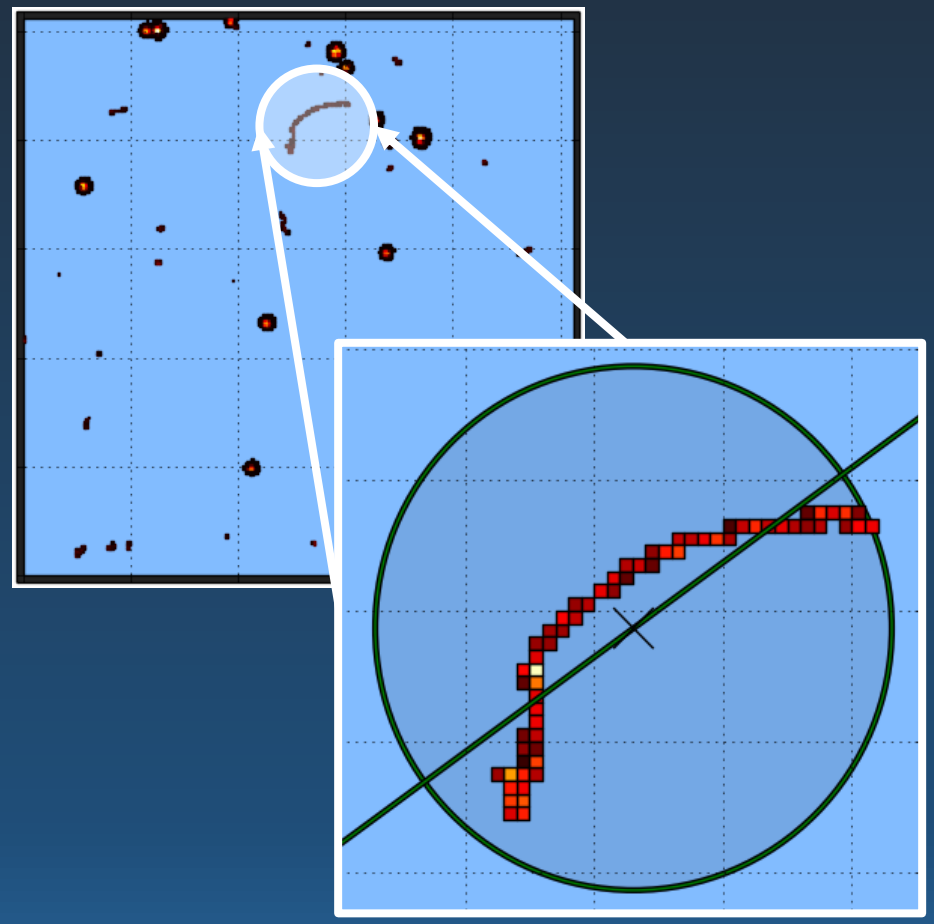
Processing the dataset

- We want to extract individual particle signatures in the detector – *clusters*:
  - *Groups of adjacent pixels;*
  - *Shape dependent on particle type, energy, direction, etc.*
- CERN@school software for this:
  - *Deployed via CVMFS;*
  - *Requires Python pre-built libraries with $PYTHONPATH pointing to CVMFS location;*
  - *Runs anywhere on the grid.*



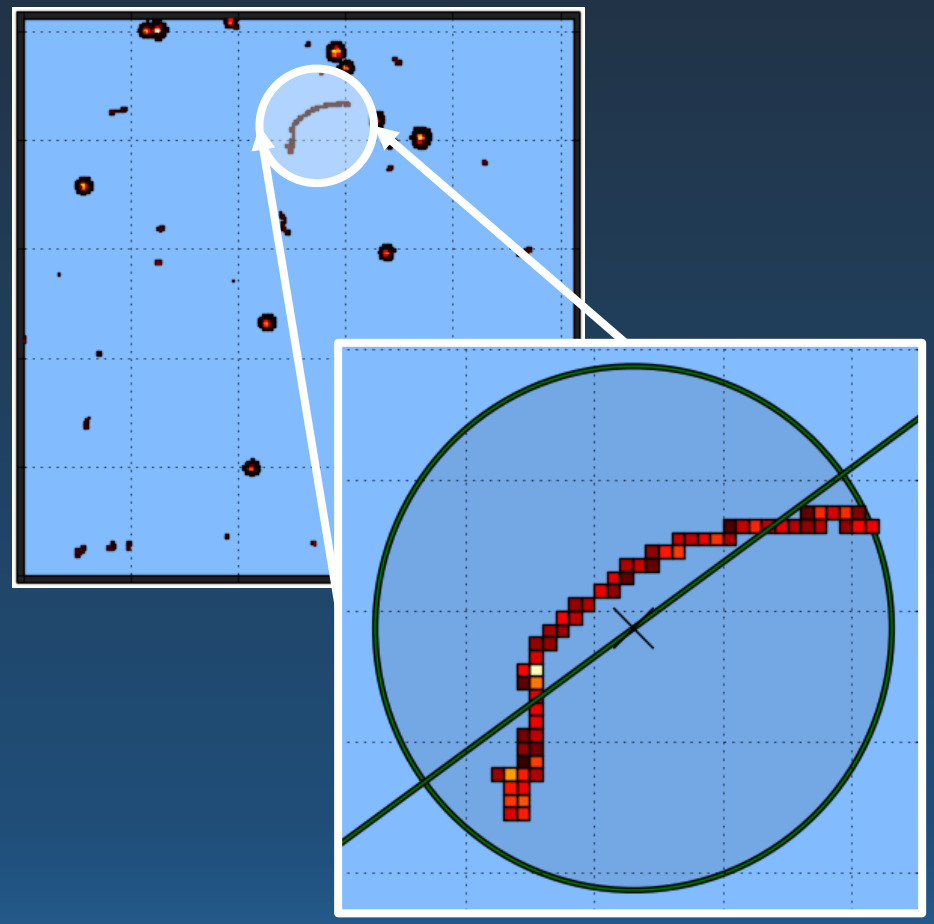
Processing the dataset

- Use a metadata query to select desired frames as the job input and run on them:
  - *python process_frames.py*
  - *Need to specify: query JSON, local output location, job details, DFC output folder.*
- Cluster processing and analysis performed on the grid:
  - *Individual clusters visualised as .png files;*
  - *Includes cluster metadata – the cluster properties are calculated…*
  - *…and returned in the job output via a JSON file.*



Processing the dataset

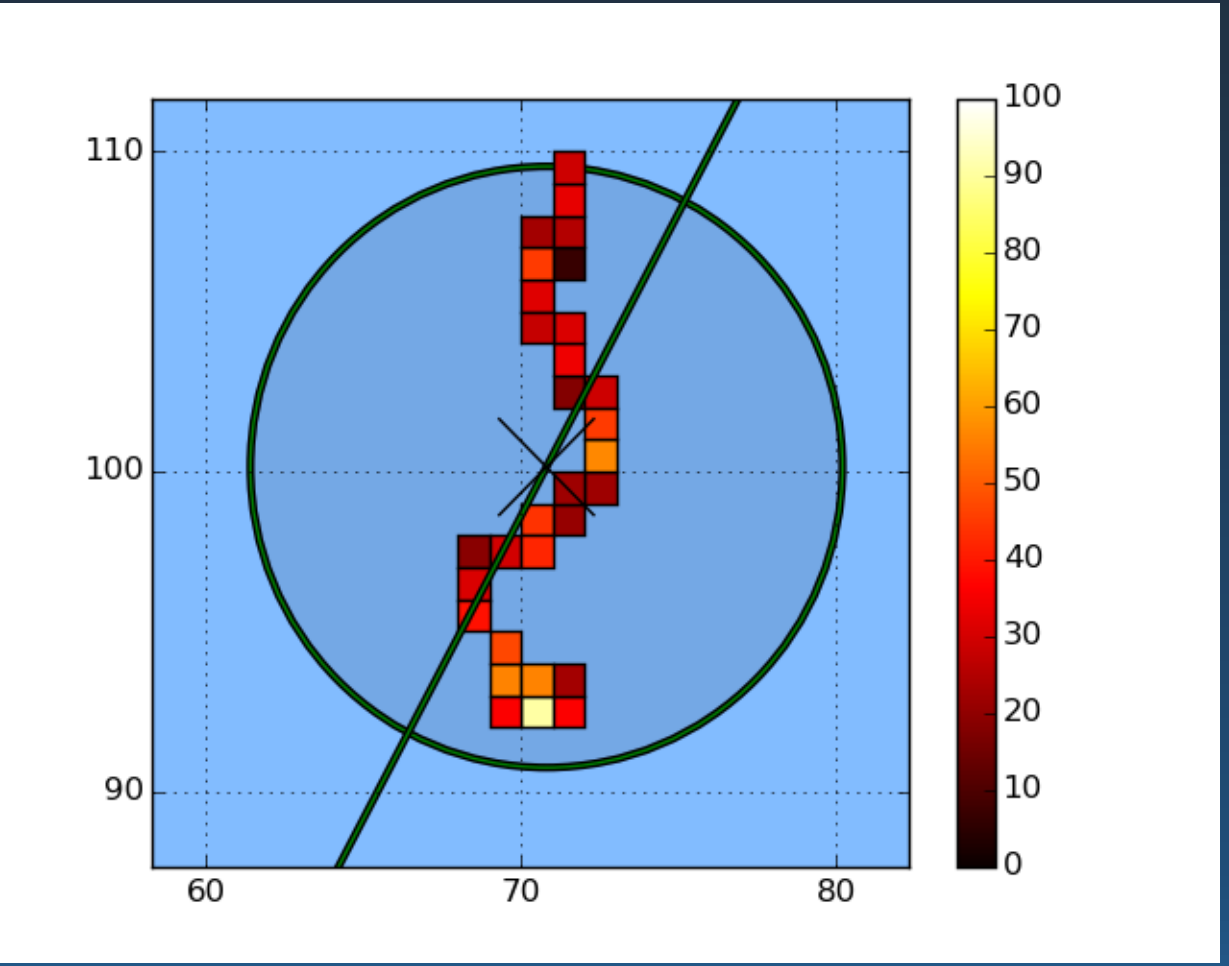
- Again, the cluster metadata is assigned with a separate script using the JSON returned by the grid job:
  - *Not ideal – DIRAC needs to think about a way of assigning metadata on the fly…*
  - *python add_cluster_metadata.py*
- Clusters can now be searched via the metadata – the cluster properties.
- It is also possible to define the parent/child relationships – TODO…



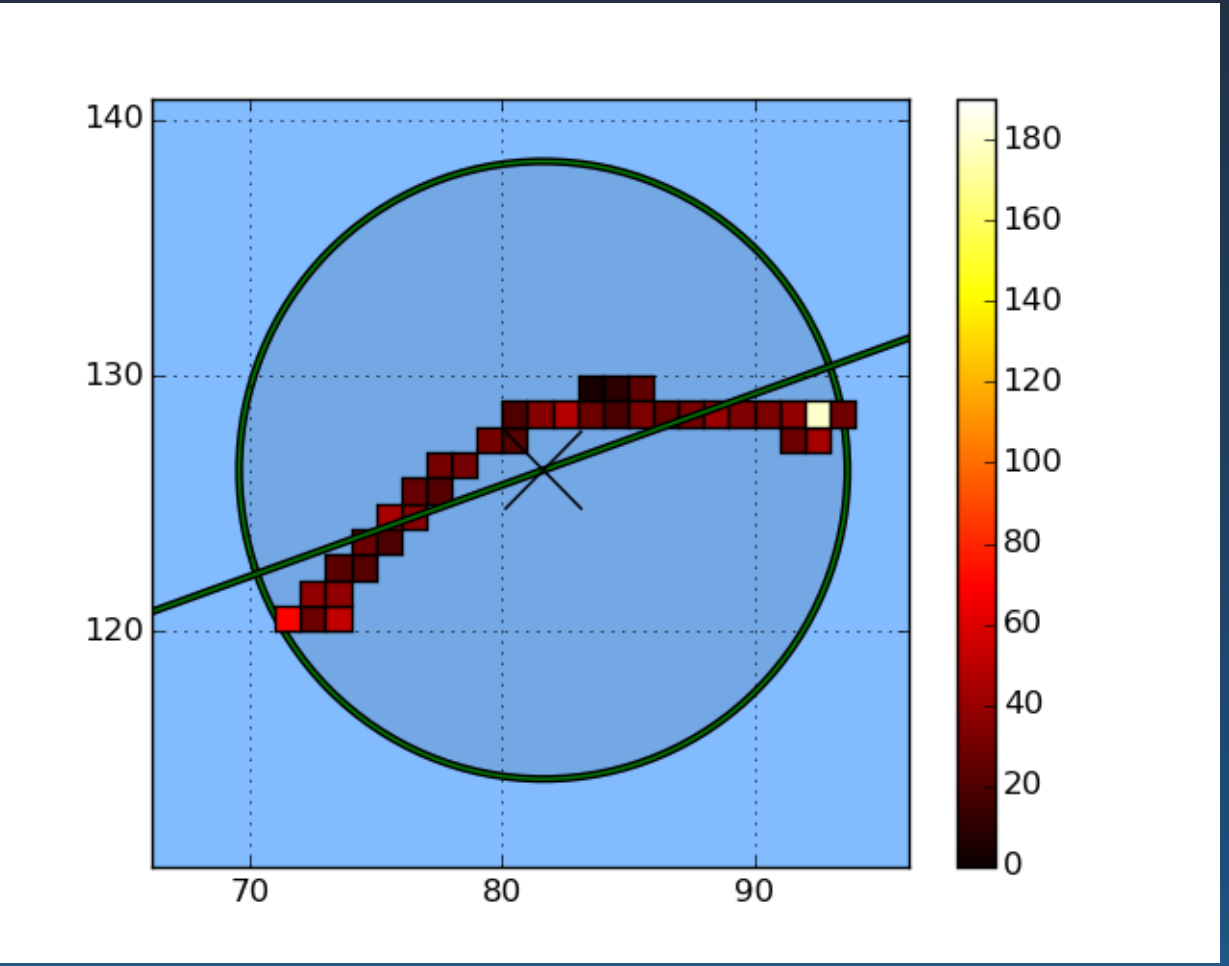
Data analysis with metadata

- Use case: search a huge frame dataset for “interesting” clusters:
  - *Near continuous running over a weekend; acquisition (shutter) time 60s;*
  - *“Interesting”: cluster size > 30 pixels;*
  - *Retrieve cluster images for analysis;*
  - *python get_clusters.py*
- Data uploaded, processed and analysed using DIRAC and the “Getting Started” toolkit.

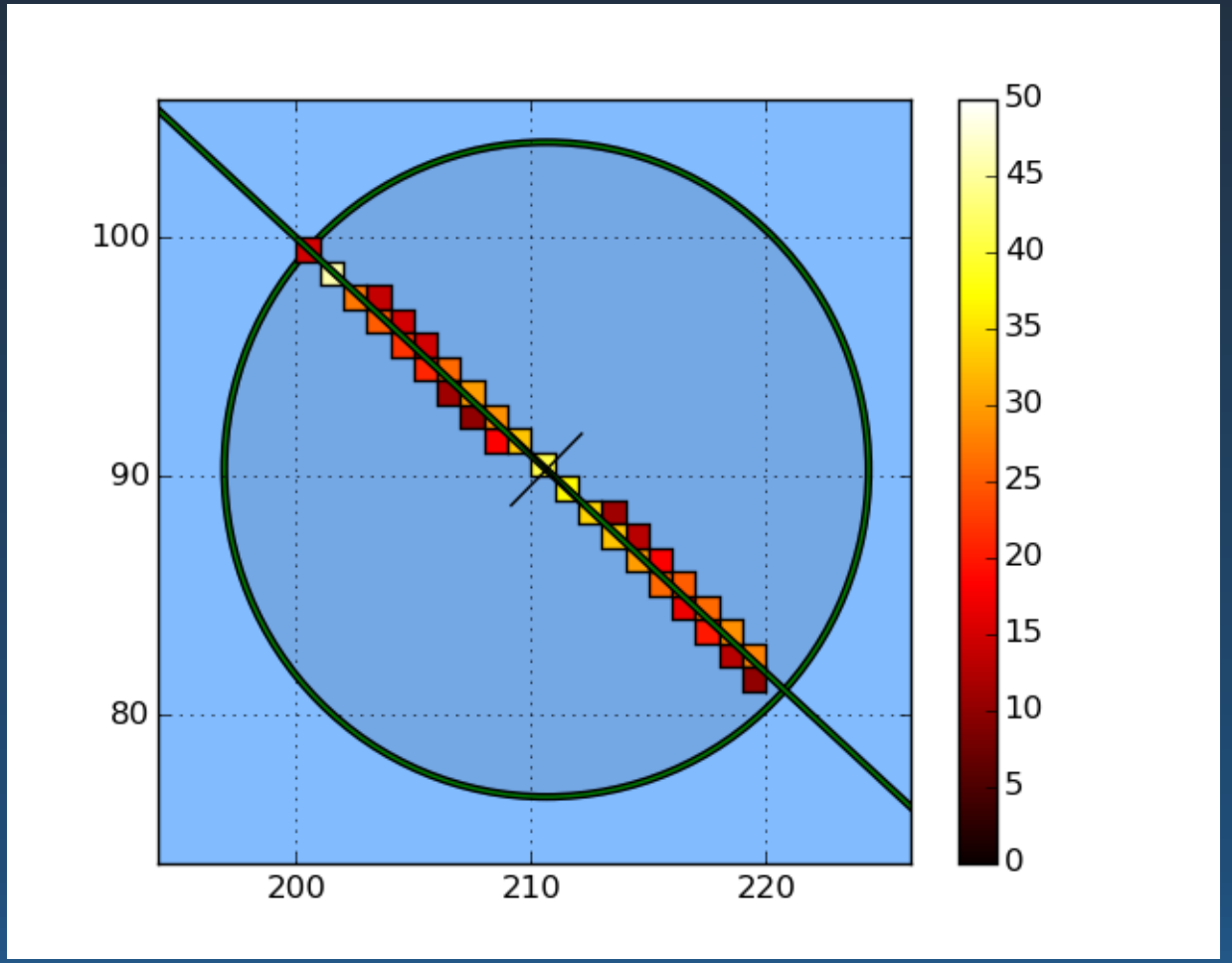
Results



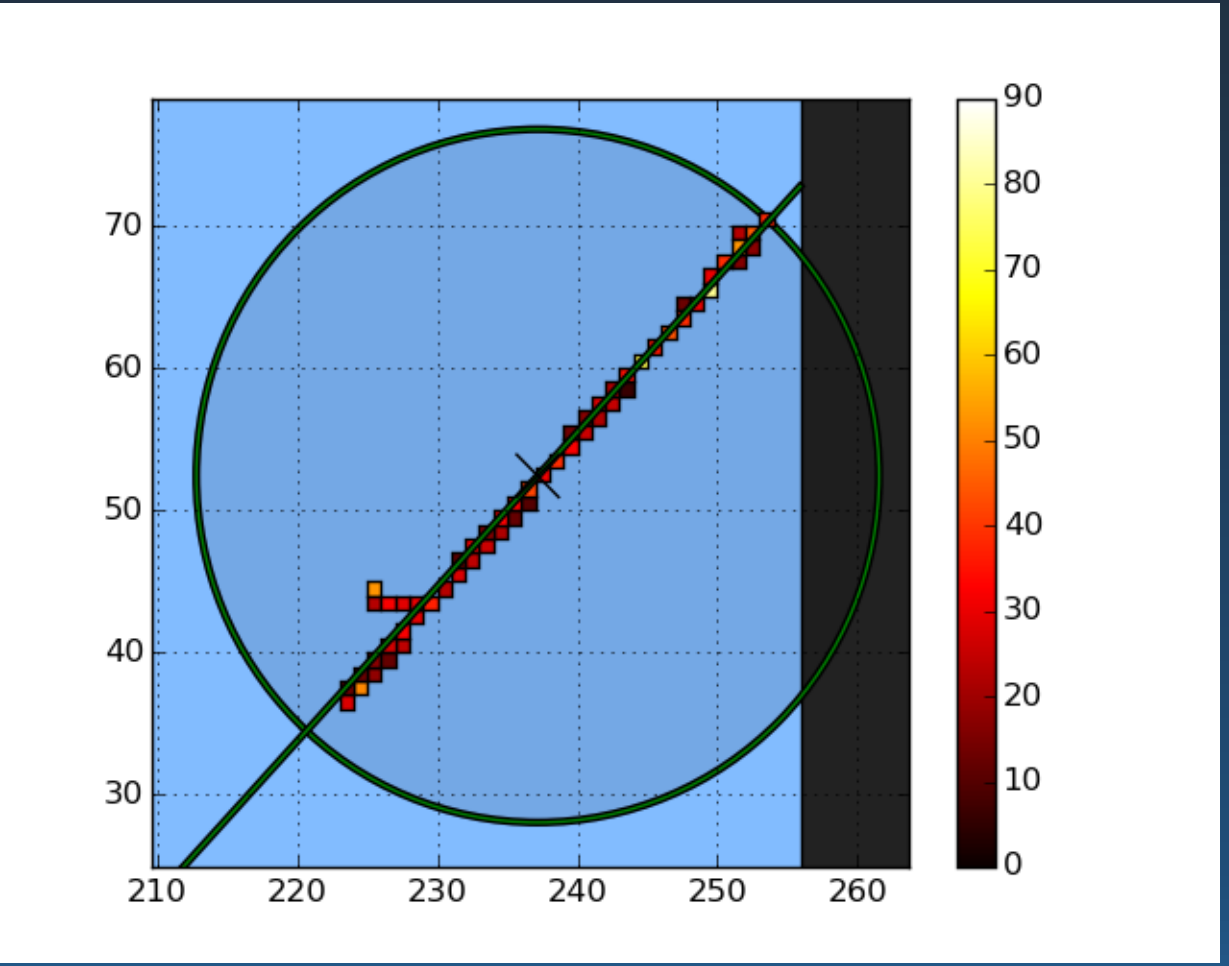
Results



Results



Results



Observations and further work

- It is possible to implement a data management system using DIRAC:
  - *Job management, data upload, secondary processing, etc.*
  - *Metadata can be defined, added and queried within the framework.*
- Some refinements needed from DIRAC:
  - *Assigning metadata "on the fly" during the grid job;*
  - *Documentation for the Python API (work in progress – can contribute now!);*
  - *Metadata keys common to all DIRAC users and VOs…*
- Next steps:
  - *Implement parent/child relationships;*
  - *Integrate with Ganga for job management/clever job splitting;*
  - *Replica management?*

Again – huge thanks to Janusz et al. at Imperial for help and support!

@GridPP @twhyntie

Thank you for listening!
Any questions?

T. Whyntie*, †

* *Queen Mary University of London*; † *Langton Star Centre*

GridPP
UK Computing for Particle Physics

Queen Mary
University of London