Analysis on the WLCG

Brian Bockelman
OSG Storage

According to Wikipedia

- “Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it”

This talk

- In this talk, I aim to explain "how analysis works" in the LHC.
- Being a CMS employee, my information will be most accurate for CMS.
- Having attended the latest ATLAS data management developer's meetings, I know a decent amount about their models too!

What is Analysis?

- A physics job is generally called analysis if it is not centrally planned. This may imply:
  - Is done using special end-user tools.
  - Implies that it is done to a subset of the data which is interesting to a user/group.
  - May be small scale (each task is not measured in CPU millenium)
    - Usually I/O bound - not CPU-heavy transforms, at least not on the grid..
  - Chaotic! Unplanned!
  - Users can produce *weird, bad code*!
    - Ever seen code leaking 3GB RAM in 15 min?

It might be simpler to say everything it isn't!

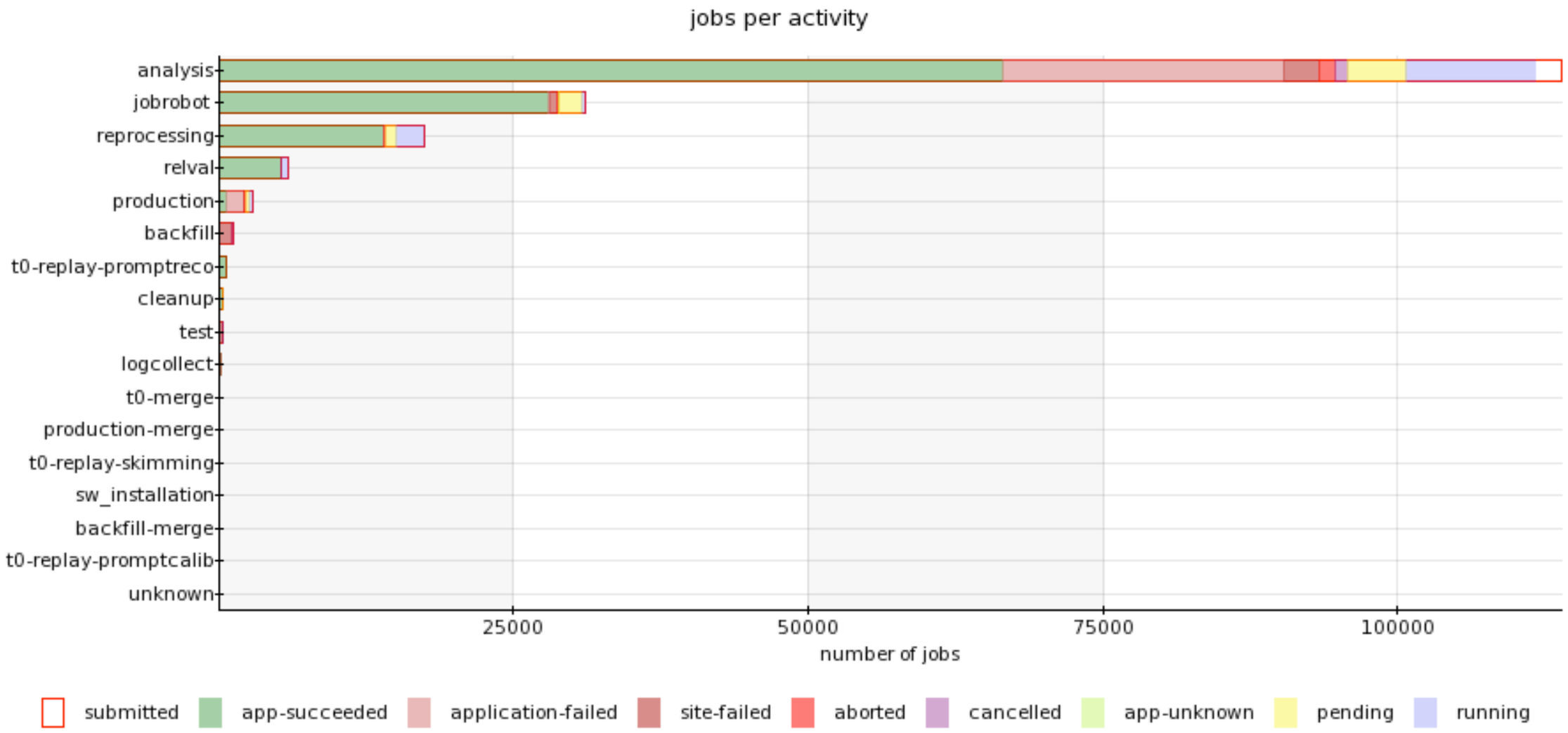
What is Analysis?

- In CMS and ATLAS, the T1 sites are primarily foreseen as centrally controlled.
  - In CMS, analysis is *banned* at the T1.
  - In ATLAS, BNL is often the most used analysis site, but T1s are being used heavily for data-related activities.
- The majority of analysis is meant to be done at T2 sites, against data on disk.

What is Analysis?

- In CMS, analysis is performed using CRAB - CMS Remote Analysis Builder on batch systems - or done by using ROOT/CMSSW on laptops.
- Today, we'll talk about batch systems / analysis on the grid.

Setting the scale



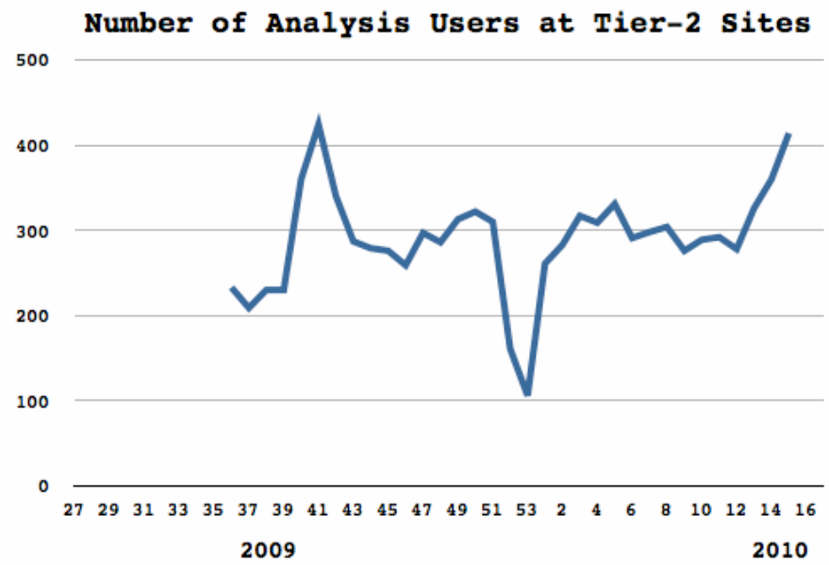
From the last 24 hours of activity.

Note: computing experts away at conference.

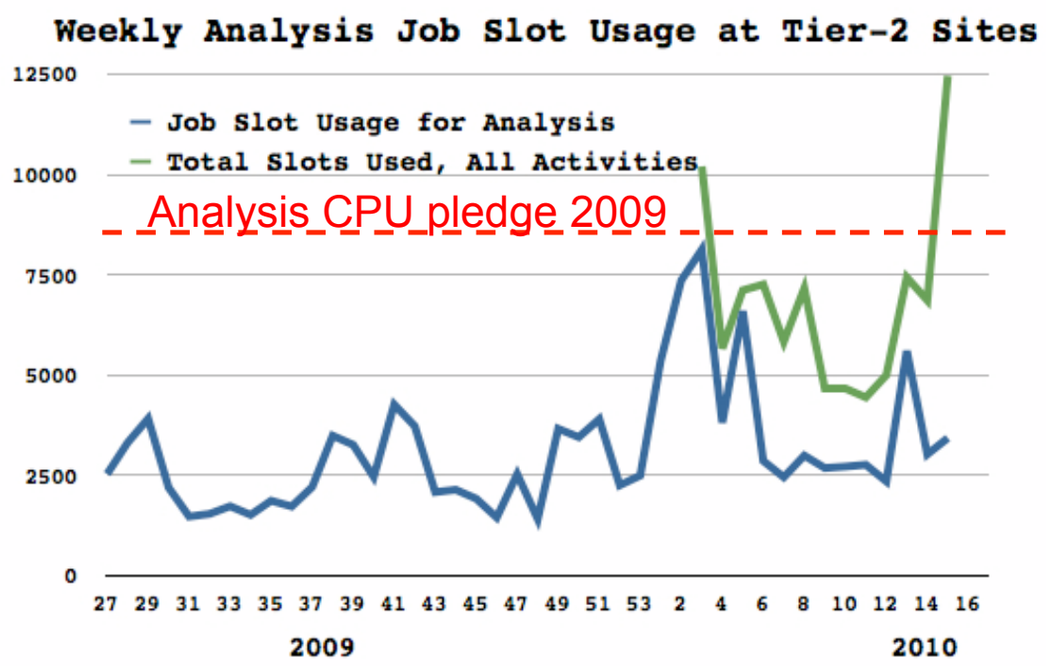
Stolen slide

CPU Utilization

While we have 300-400 active users per week …



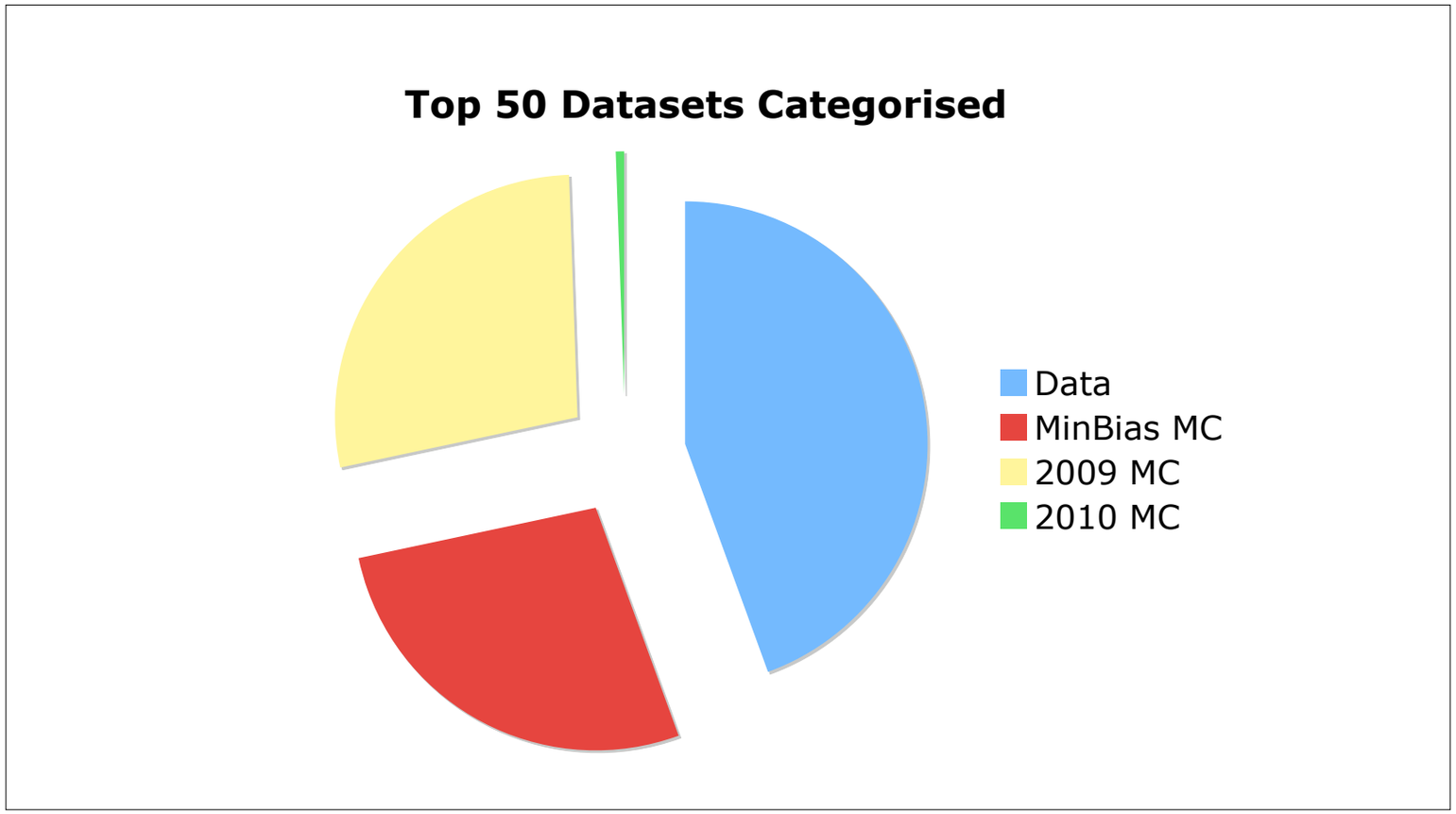
… we use only 1/4 - 1/3 of the analysis CPU pledge for 2009.



From: "Current Status of Operations", Dave Evans, CMS Offline and Computing Week

Stolen Slide

Datasets used within last month

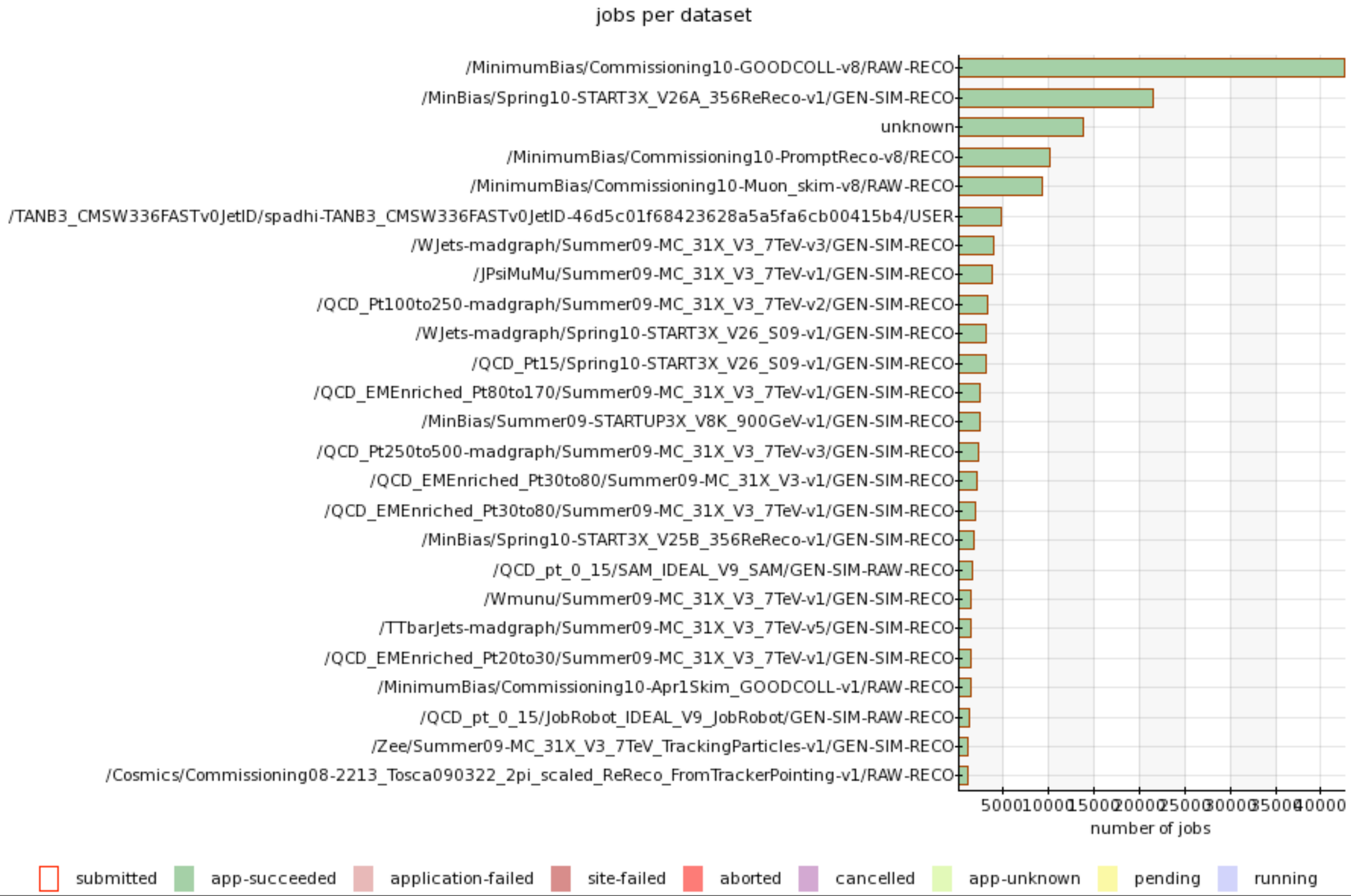


- Total of 1715 datasets used within last month.
- 50 most used datasets account for 2/3 of all jobs.
- Among those, 3/4 are data and MinBias MC.

From same presentation

Tier-2 in review

Most popular datasets, April 16-now:



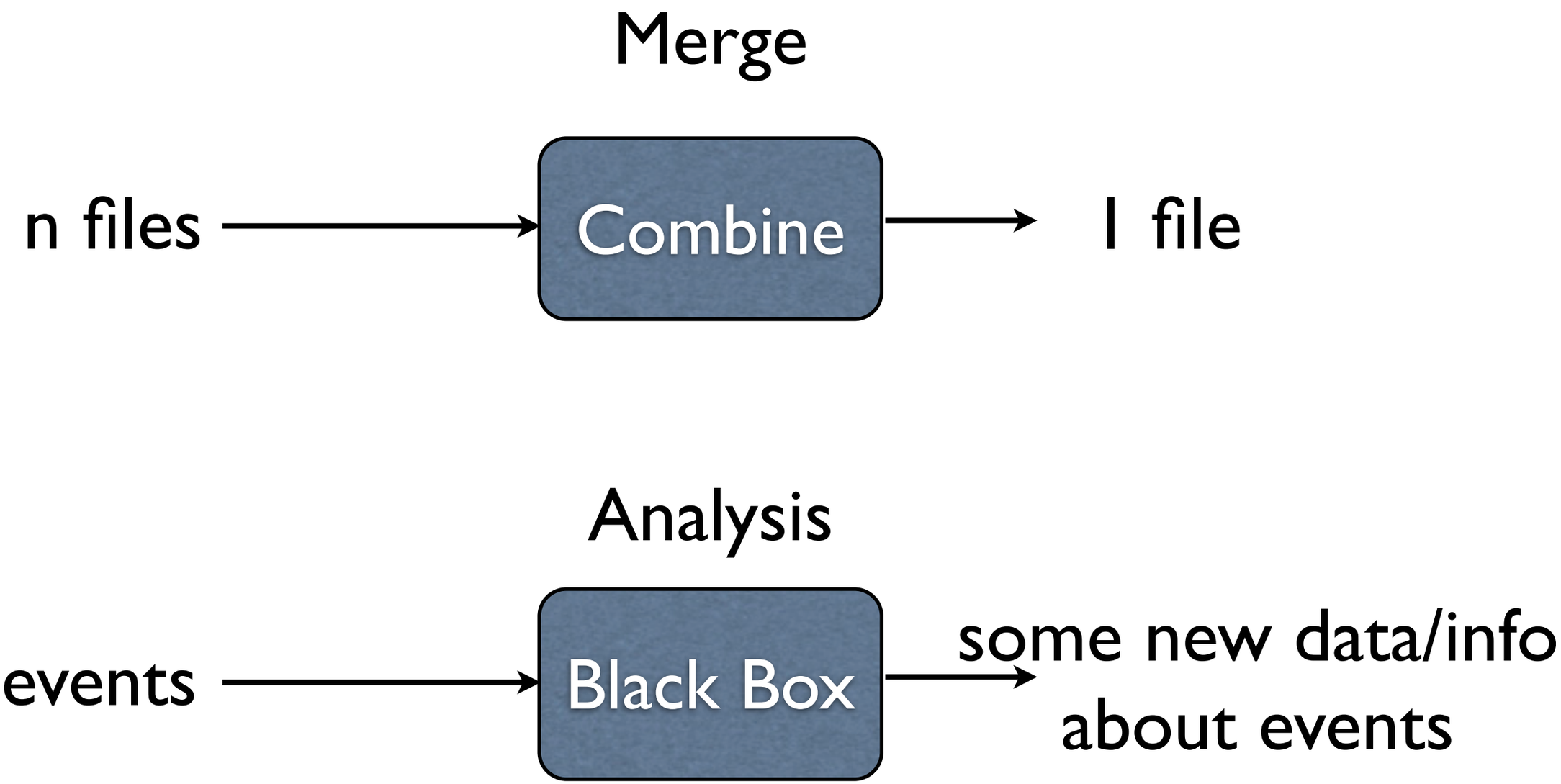
"Typical" Analysis Jobs

*Any number of the below may be combined into 1 job

Tuple creation

Skimming

"Typical" Analysis Jobs



"Typical" Grid Analysis

- Surprisingly, deep and careful analysis - the kind that a dictionary would define - is rare on the grid.
- In my observation, it's all about continuously refining the input data to select possibly interesting events and remove uninteresting parts of the data.
  - Reduce data size to a "laptop friendly" subset, where you can interactively play with things.

Analysis Example

- "Give me all events with more than 2 jets over 10 GeV".
- "Give me events with more than 1 GeV of missing energy."
- "Give me these events, but only save the charged particle tracks."

Life is a database

- To make an analogy to SQL -
  - When we skim, we have a very strict WHERE clause.
  - When we create output, we have a small subset of all the possible columns.

ROOT Data

- To think about I/O and storage of analysis without mentioning ROOT would be a sin.
- For a given data format, each event will have a set of objects describing it.
  - In CMS, these are grouped into about 300 objects called “branches”.
  - Each branch is stored separately on disk.
    - It’s just a column store!

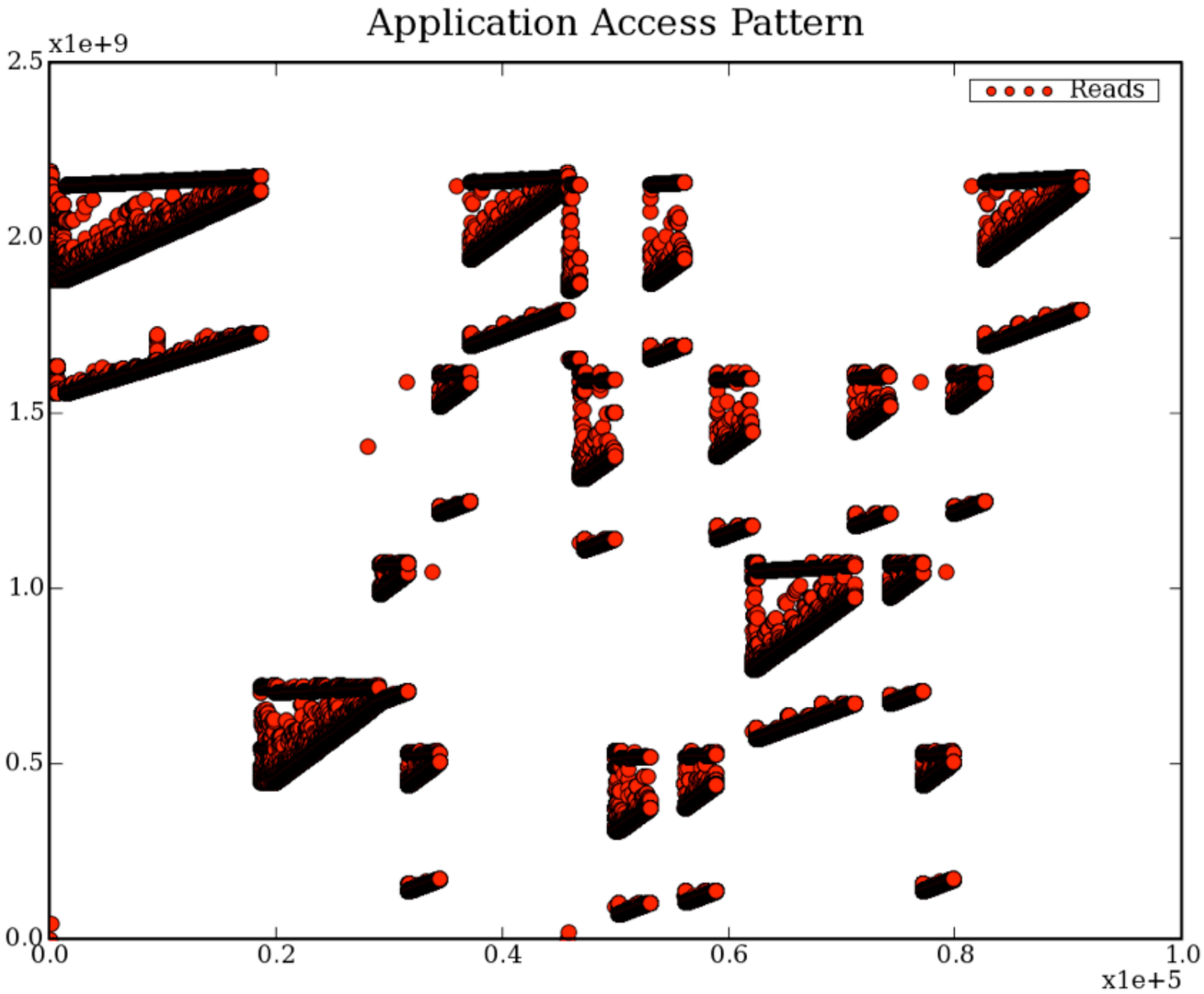
But not that simple

- We compress all data along the column.
  - So each saved buffer on disk contains some number of events, and the serialized version is zlib compressed.
    - Not every buffer contains the same number of events.
- Because of the compression of variable number of events, we might end up very small, very random reads even when the physicist reads “serially” through the file.

ROOT I/O

ROOT I/O Can Be Bad!

Interesting data point:
~75% of network traffic is not used by the application.



ROOT I/O

- In the last year, ROOT started thinking about optimizing I/O:
  - Prefetch data so we don't wait on it.
  - Change file organizations to not have such silly access patterns.
- And experiments got serious about using these optimizations and implementing their own.
- More later on this.

Data Management in CMS

- Files are organized into ~200GB blocks.
  - Blocks are the smallest unit of data we can run grid jobs against. Done to so we don't have to track file-level locations.
  - Blocks form datasets. Each dataset has some physics-related definition, and are defined centrally (heavyweight).
- For experimental data, (tape) archival sites are T1s.

File Movement in CMS

- T0 exports data to all T1s.
- T1s and transfer data with any T1.
- T2 can download data from any T1.
- (New!) T2 can transfer with any T2.
  - This brings us close to a “full mesh” of transfer links -- we rely on the fact that the internet is really a global network.

File Movement in CMS

- Because there are so many different paths a dataset can take, we have to do non-trivial routing to decide on the best source site.
- File movement is subscription-based.
  - Some subscriptions are done centrally.
  - About half are done by decentralized physics groups.

Data Management in ATLAS

- The transfer mesh is more according to the hierarchical model. Differences:
  - T2s only talk to their T1.
    - T2s do not talk to each other.
- Datasets are more “lightweight” objects - it can be output of a user’s task.
- Subscriptions are handled almost 100% centrally.

Analysis Workflow

- First, CMS:
- User puts together a task in a cfg file.
- CRAB splits this into many jobs (or submits the cfg to a server which splits the jobs).
- Jobs are submitted to the grid or glidein.
- Jobs arrive on worker node and startup...

WN Analysis Workflow

- Once on the worker node, the job will read the site's configuration to determine how to connect to the SE. Depending on the job config, it will either:
  - Read directly from the SE.
  - Read the file in 128MB chunks and copy them to local disk (not widely used).
- In ATLAS, each site has a "local file mover" which can copy the file to the local disk completely.

WN Analysis

- Because ROOT I/O has been so bad in past, experiments have been moving away from direct connections.
  - This is OK if you read more than about 1/4 of the file in your analysis
  - For skimming workflows, you might only read out 1/10 or less of the data.
  - So, pulling in 100% of the file is getting expensive. Sites say it's acceptable to do this to limit worst case usage.

Output

- CMS and ATLAS handle the output differently.
- In CMS, the output is copied back with the grid tools as part of the batch system job.
  - I.e., work stops until the copy finishes.
  - The output is not moved by PhEDEx, rather directly by grid tools (lcg-cp).
  - User output can be registered in local group database systems, but not in the experimental-wide ones.

Output

- In ATLAS, the "dataset" creation is more lightweight.
  - User tasks output a dataset in the global experiment, which can be handled by their production file movement.
  - User data is staged out asynchronously - the next job can start before stageout is finished.
  - Stageout stays in the regional "cloud" - unlike CMS, where stageout can go anywhere in the world (and hence be quite slow).

Output

- There are several use cases for using the output of one analysis for the next analysis.
  - However, this is currently relatively rare - for example, in CMS, you need a valid dataset to submit to.
    - Brave users can do this.
- Most folks do one analysis pass, then take it off the grid.
  - If they need to re-iterate, they start off with the base physics dataset again.

Bottlenecks

- ATLAS has their nice asynchronous functionality because they write their own pilots.
  - Why can't this be part of the grid middleware?
- Usually, the storage systems scale fine for grid workflows - the only "pain" is self-inflicted due to ROOT I/O.
- Grid workflows have trivial bandwidth needs - not always true for non-grid workflows.

Future Directions

- Right now, we *only* run on sites where 100% of the data we want is prestaged.
  - Can this be relaxed?
  - Can we run on site where 95% is prestaged? 75%? 25%?
  - Can we run on sites where we don't have any storage at all?
    - I.e., can we use T3 or opportunistic sites. We only need 1MB/s / job.

Future Directions

- For conditions data, both experiments distribute this through a network of HTTP caches.
  - No explicit location tracking.
  - But allows data locality.
- Is there a lesson to learn here?