Imperial College
London

Experience with distributed analysis in LHC*b*

Ulrik Egede
on behalf of the LHC*b* collaboration

CHEP 2006
Mumbai

Introduction

The analysis model of LHC*b*

The framework for distributed analysis

Use in the collaboration

Performance of our system

Future developments

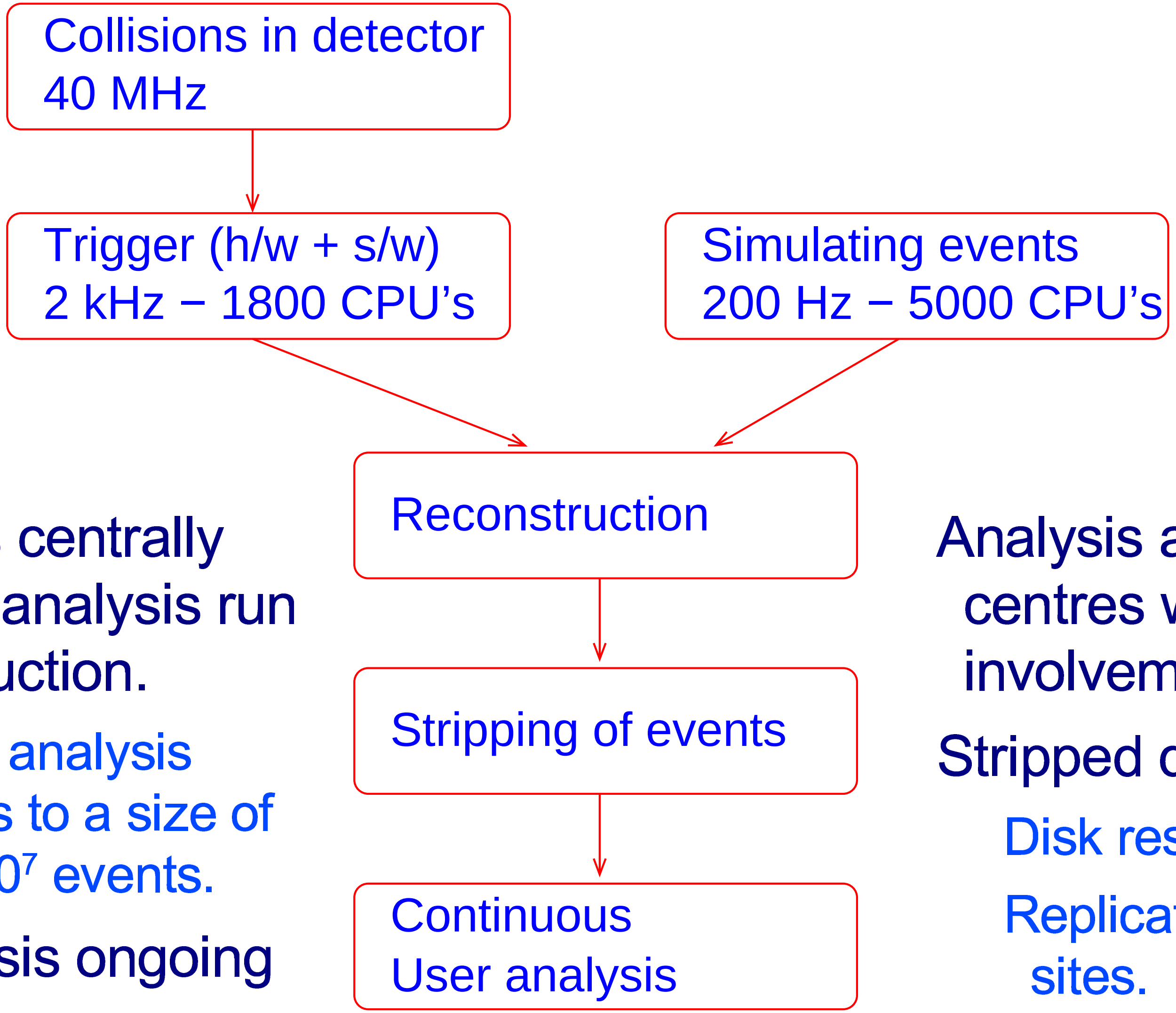
Summary

LHC*b* analysis model (1/2)

Stripping is centrally managed analysis run as a production.

- Reduces analysis datasets to a size of 10^6 to 10^7 events.

User analysis ongoing during the year.



Collisions in detector
40 MHz

Trigger (h/w + s/w)
2 kHz − 1800 CPU's

Simulating events
200 Hz − 5000 CPU's

Reconstruction

Stripping of events

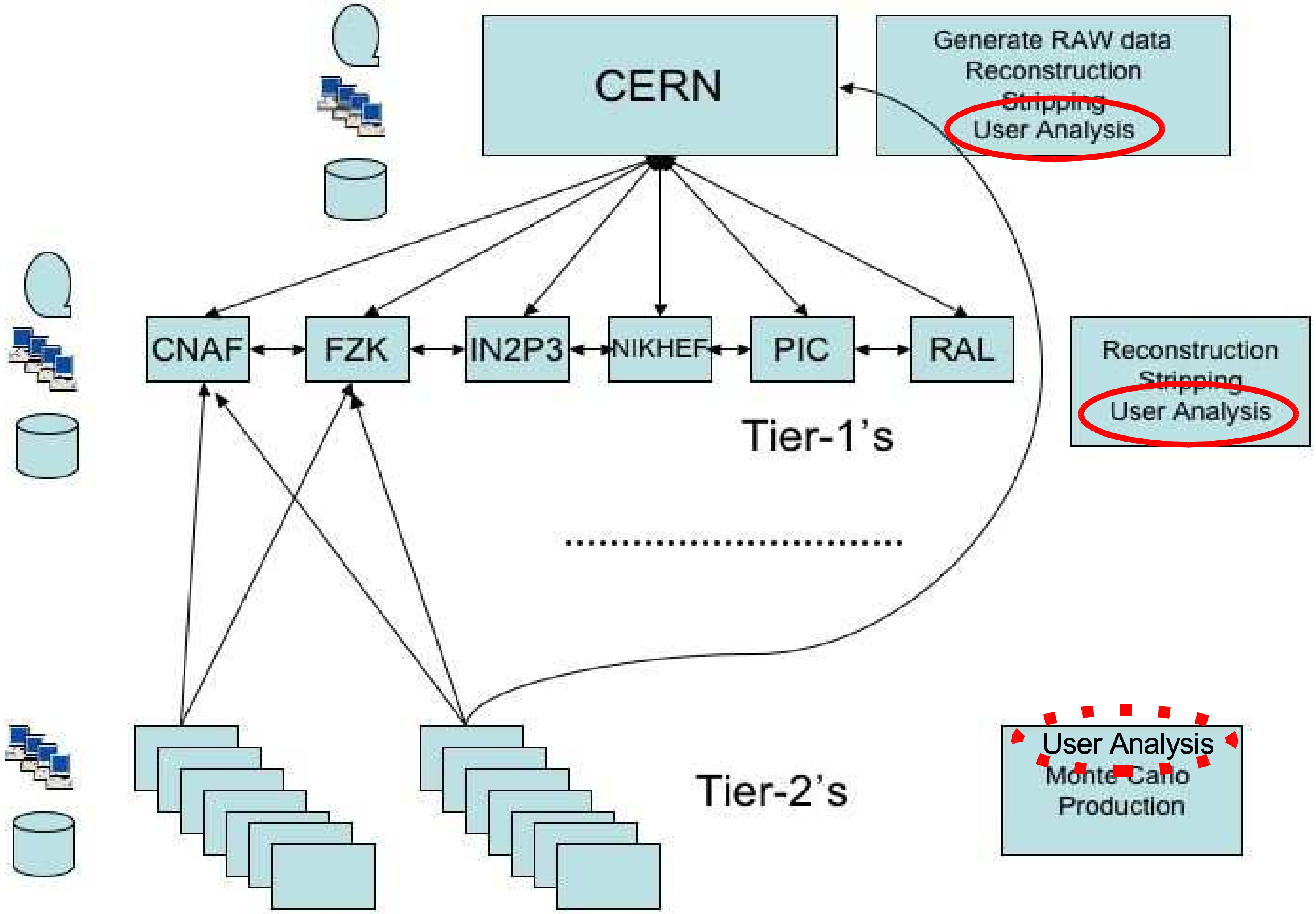
Continuous
User analysis

Analysis at all Tier 1 centres with LHCb involvement.

Stripped data will be:

- Disk resident
- Replicated to all sites.

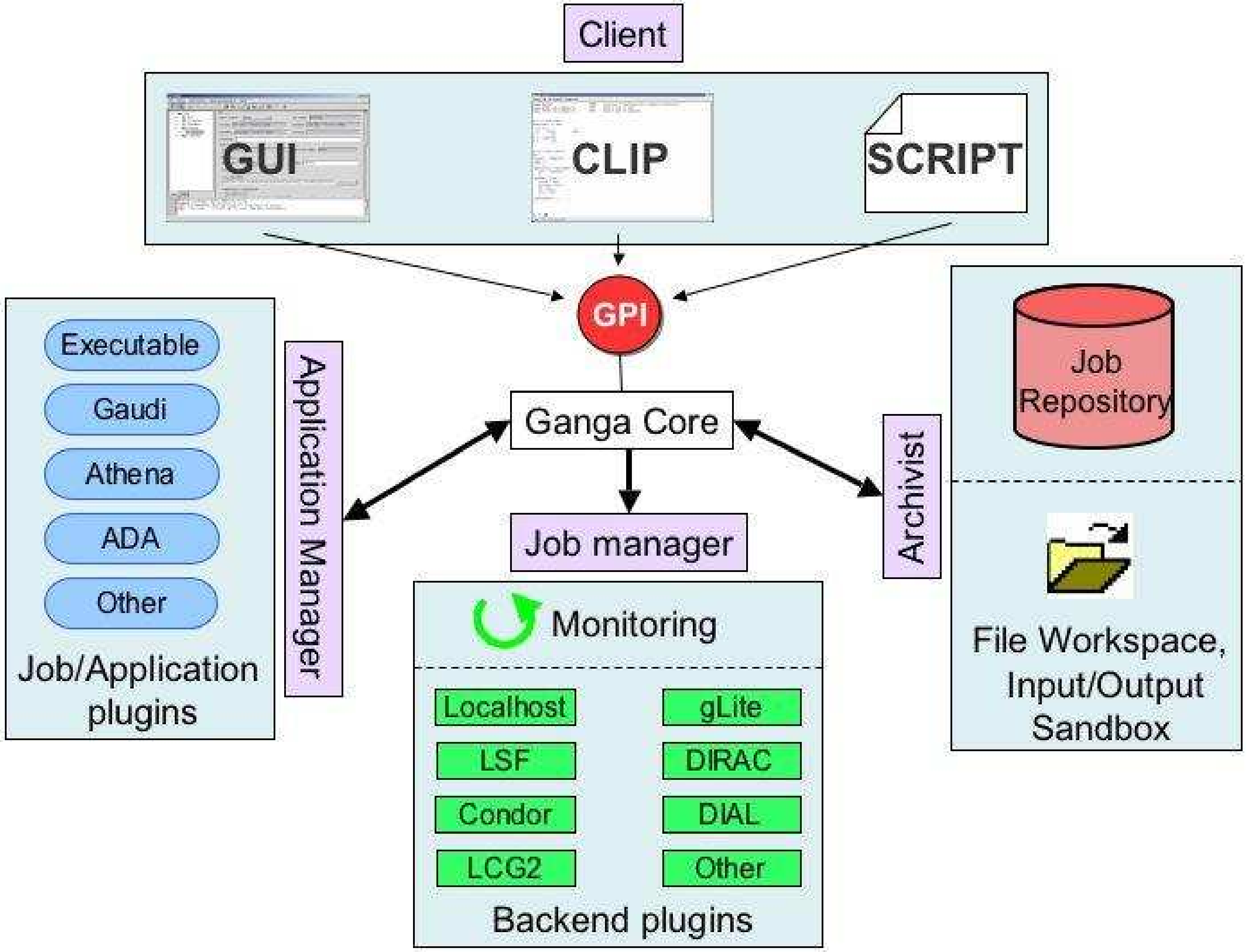
LHC*b* analysis model (2/2)



Ganga – the user analysis framework

- For analysis we want a system that makes it easy for a user to access the Grid.
  - System should not be general - we know the main use cases
    - Use prior knowledge
    - Identified use pattern
  - Aid users in
    - Bookkeeping aspects
    - Keeping track of many individual jobs.
- The Ganga framework developed in cooperation between LHC*b* and ATLAS is developed to satisfy these criteria:
  - See talk "*Ganga – a Grid user interface*" presented by Karl Harrison for a detailed presentation on the design and architecture of Ganga.
    - Today, 2.40pm, Distributed event production and processing track

Ganga Architecture

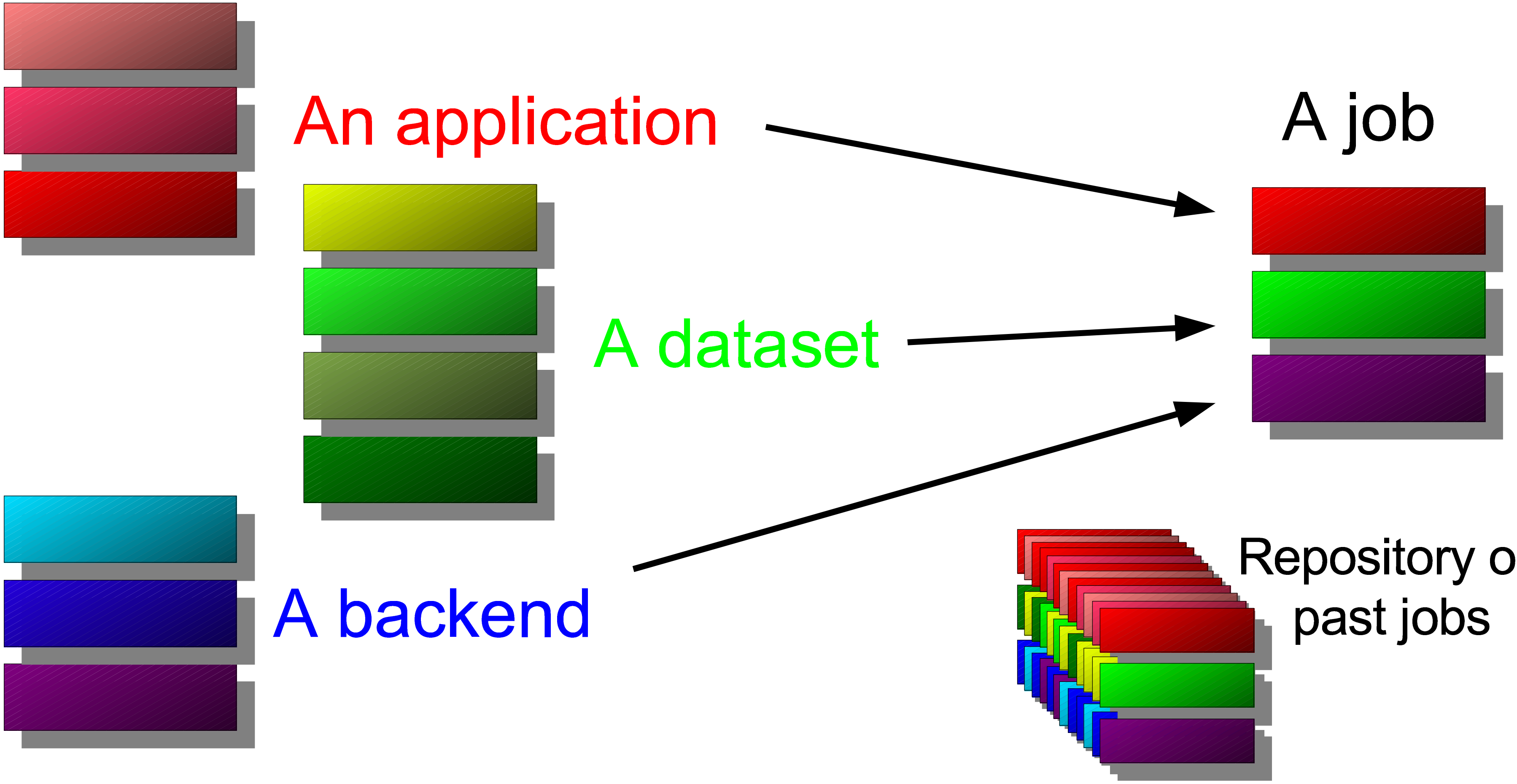


Implementation:

- Pure python
- ~20k lines of code

User view of analysis using Ganga

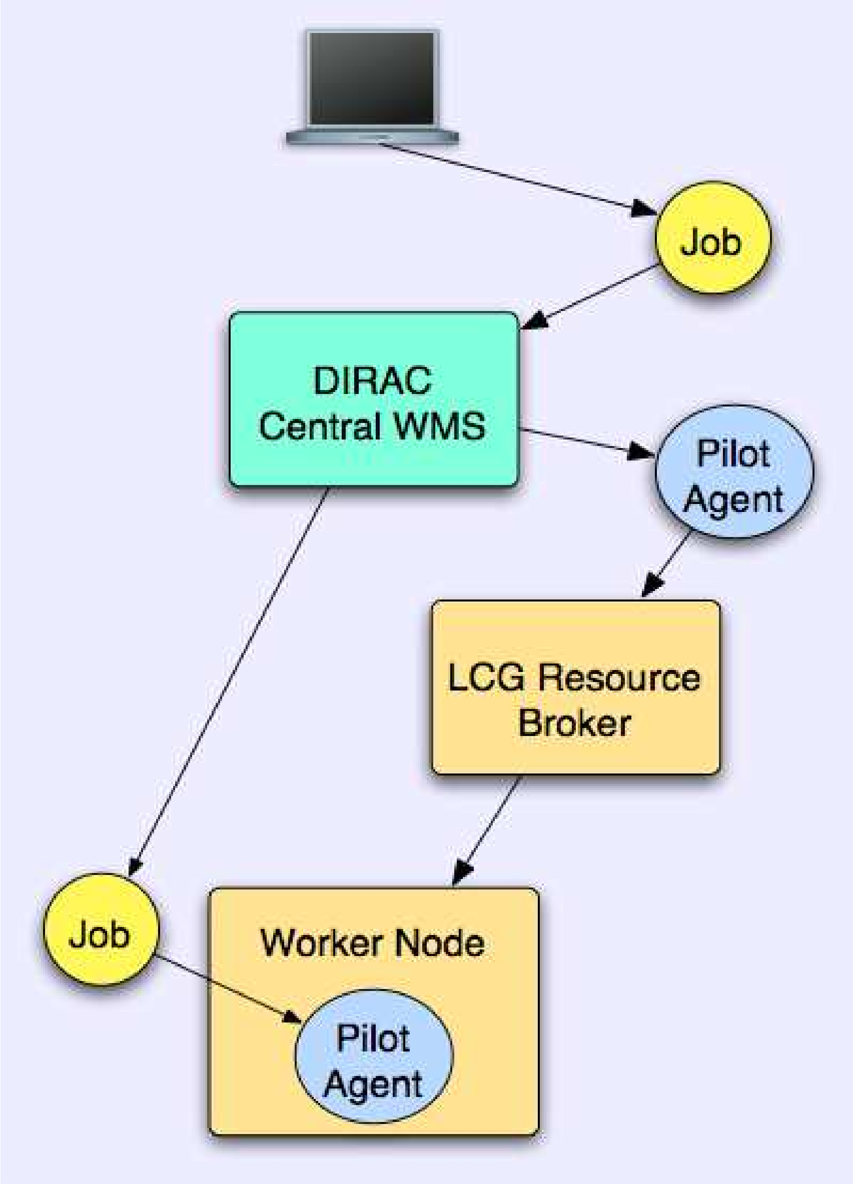
To define a job we combine different parts to create a job



Analysis access to the Grid (1/2)

- No direct submission of jobs to LCG for LHC*b*
  - Analysis jobs are submitted to the Dirac workload management system (WMS) originally developed for LHC*b* Monte Carlo production.
- This gives us the advantage to:
  - Protect users from instabilities of LCG.
  - Reduce the knowledge required of users.
  - Provides transparent access for reading and writing data on SE's
  - Allows LHC*b* to set priorities and or restrictions for analysis jobs.
- See the talk by Stuart Paterson “DIRAC Infrastructure for Distributed Analysis” for technical details on supporting LHCb analysis jobs on the Grid.

Analysis access to the Grid (2/2)



User sends job to the DIRAC WMS

WMS sends a pilot agent as an LCG job

When pilot agent runs safely on a worker node it fetches job from WMS

- Small data files returned to WMS
- Large files registered in LFC file catalogue

User query WMS for status and finally retrieve output from there.

Learning the system

Distributed analysis recently integrated into the overall training for analysis in LHC*b*.

About 50% of new users manage to submit a successful analysis job to the Grid within a 1 hour training session.

An intuitive feeling for the Ganga system is quick

Obtaining a grid certificate and register in VO still problem for many users

Learning very simple Python commands easy

About 30 regular users of system

Creating an analysis (1/3)

Predefined Python classes with specific knowledge about LHCb applications:

- Gauss – for simulating events
- DaVinci – for physics analysis
- 5 other LHCb applications

Objects know how to compile code, extract configuration, place user DLLs in input sandbox, specify files for output sandbox etc.

```
# Define an application object
app = DaVinci(version = 'v12r15',
       cmt_user_path = '~/public/cmt',
       optsfile = 'myopts.opts',
       extraopts = 'ApplicationMgr.EvtMax = 10;')
```

Creating an analysis (2/3)

A backend describes how the job will be executed

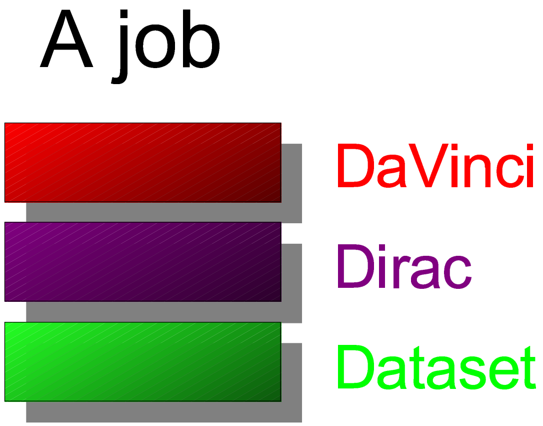
- Local – run in the background on the client
- LSF/PBS/SGE – submit to the batch system
- Dirac – submit to the Grid via Dirac

```
# Define a Dirac backend object
d = Dirac()
print d
Out[34]: Dirac (
 status = None ,
 destination = None ,
 id = None
 )
```

Creating an analysis (3/3)

To put together and submit a job is simply by combining the different parts:

```
# Create an LHCb job and submit
j = Job(name='MyJob',
        application=app,
        backend=d)
print j
Out[38]: Job(
 status = 'new' ,
 name = 'MyJob' ,
 application = DaVinci (...)
 backend = Dirac (...)
 ... )
j.submit()
```



A job

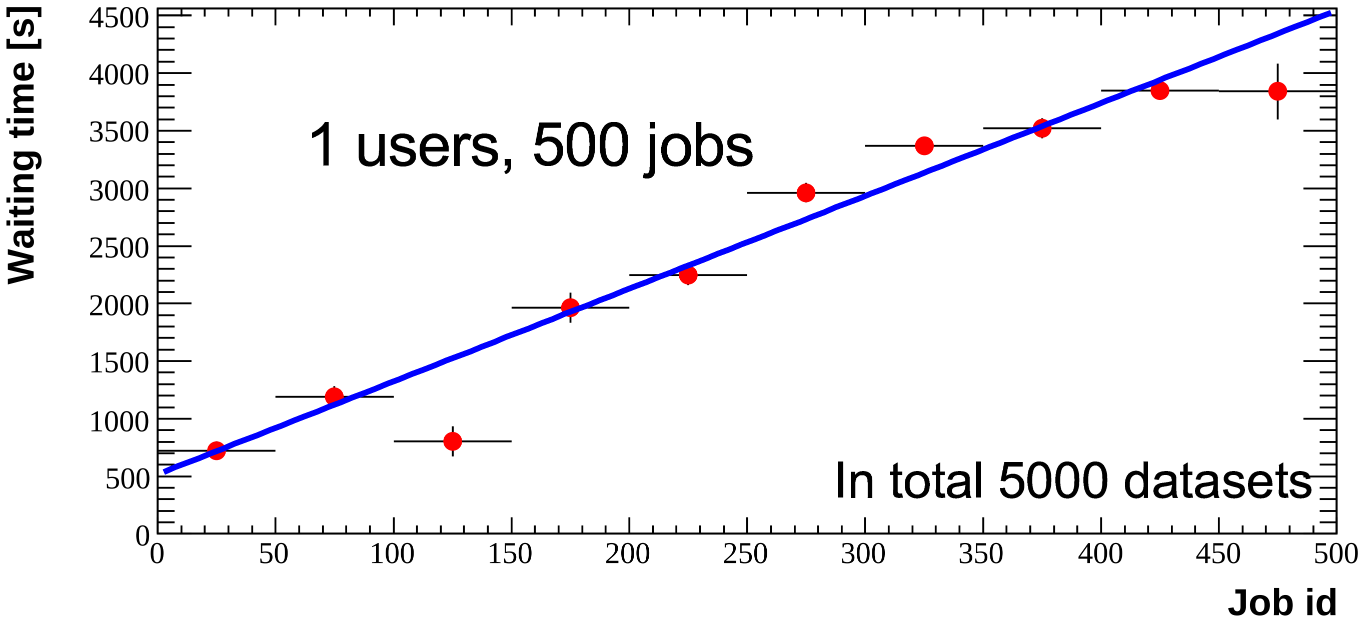
DaVinci

Dirac

Dataset

Performance: Throughput

An analysis of 5M events entered into the system at the same time.



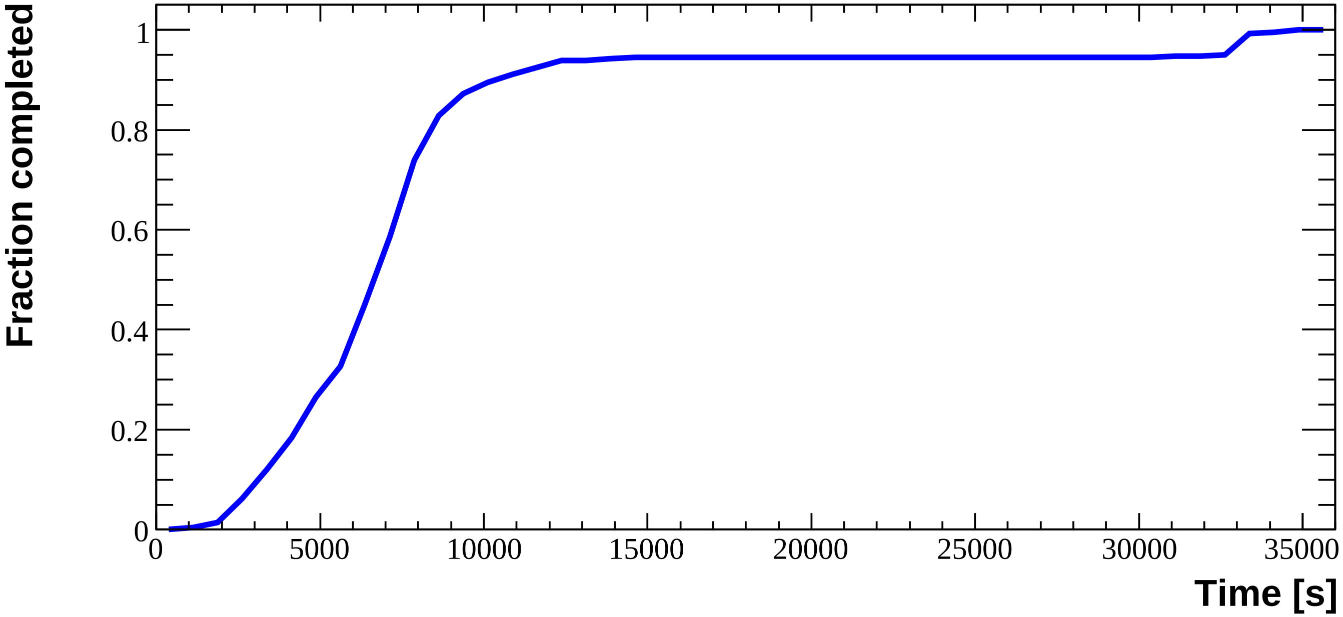
$$\text{wait [s]} = 517 + 8.06 \times \text{job id}$$

517 s (8 min) before first job starts is dominated by latency in jobs for LCG and software installation.

Queue time afterwards is dominated by the time it takes for WMS system to submit agents to LCG.

Performance: Throughput

Look at the time it takes before the results are back



90% of results are back within less than 3 hours

95% in 4 hours

100% in 10 hours

- Last 5% caused by Tier 1 site with data access problems

Performance: Scalability

Comparisons have been made of performance using the same analysis submitted simultaneously by different users.

We observe that waiting time is consistent with queue behaviour.

- Performance scales linearly with number of users

Total execution time does not change much with more users

- We have not in the analysis testing managed to saturate the available LCG resources.

Have tried up to 30 simultaneous users, so far no hard limit found.

- The corresponding number of intermittent users will be much higher.

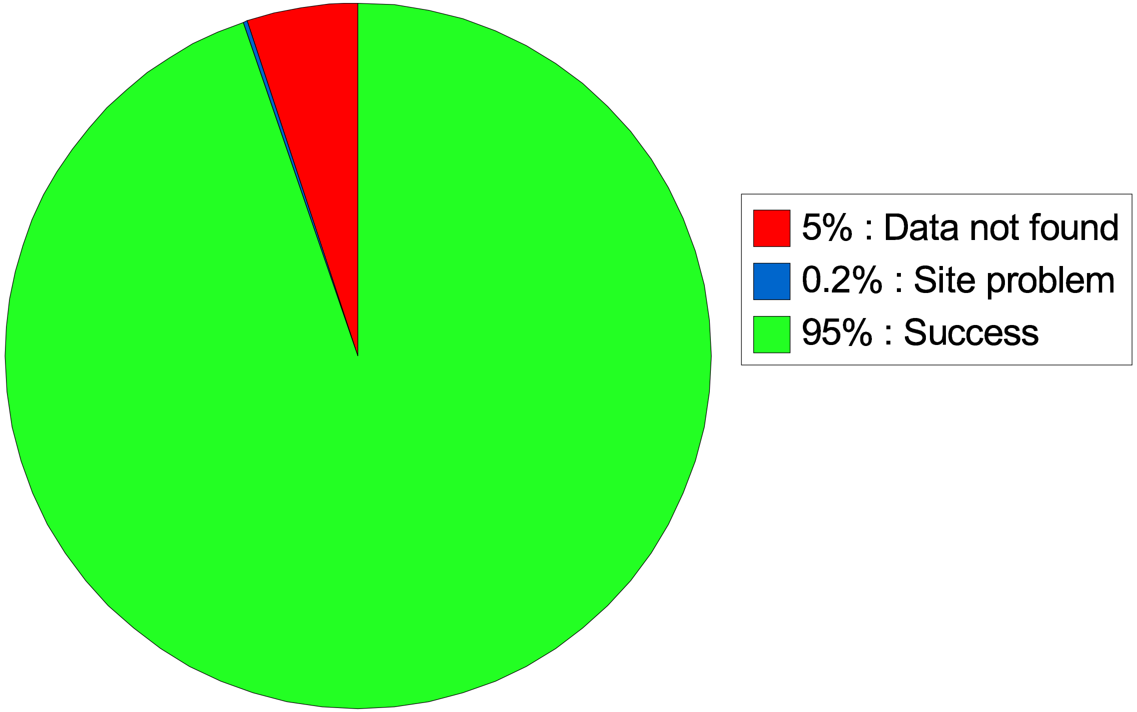
Performance: Reliability

Evaluate the fraction of jobs that succeed.

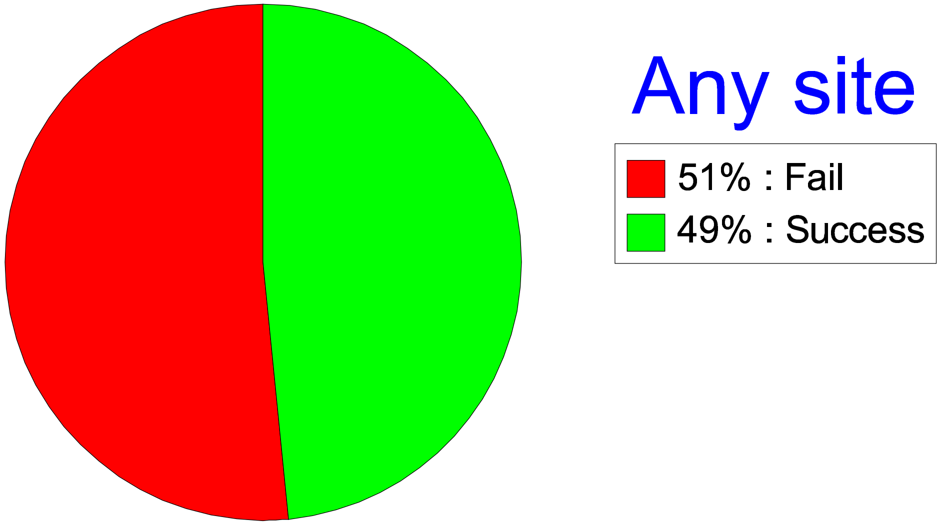
With success is meant that submission, running and retrieval of correct output all work as expected.

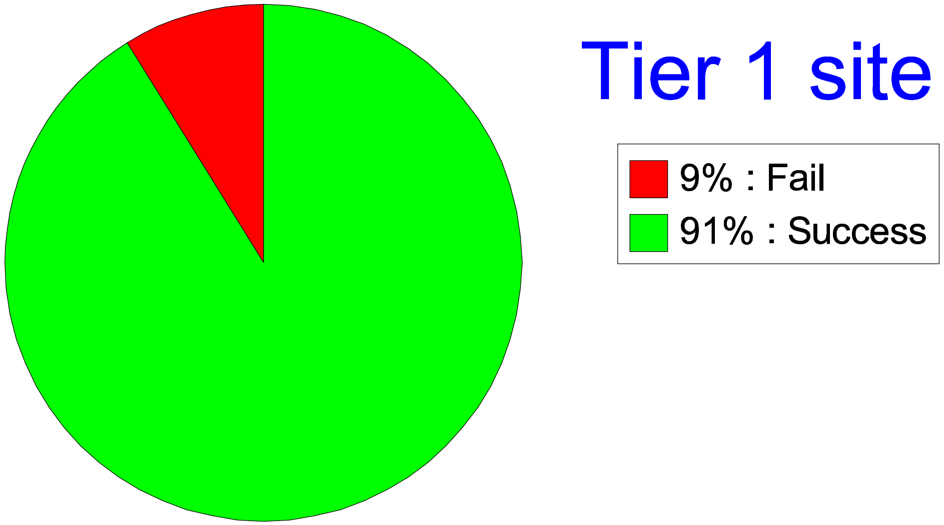
95% of jobs succeed in the first round.

Remainder dominated by occasional inconsistencies in file catalogue.



Comparing different Grid submission strategies





- Direct submission to LCG attempted as well.
  - If no limitation of submission sites the success rate is below 50%
  - If submission limited to well known functional Tier 1 site performance is close to results obtained via DIRAC.
    - Small difference attributed to lack of retries.
- Waiting time and execution time is similar for DIRAC and direct submission.

Future plans

The Ganga framework and its implementation in LHC*b* will continue to develop:

- Better concept of datasets to ease data selection.
- Automatic merging of output data from multiple analysis jobs.
- Provision of experiment wide and analysis group specific templates for analysis.
- Provision of a GUI to give users the choice between using a GUI, interactive work at Python prompt and writing scripts.
- More application types and backends will be supported

Conclusion

- The Ganga framework allows LHC*b* users to submit jobs to the Grid in an easy and transparent way.
- Submission of Grid analysis jobs submitted through the Dirac WMS tested.
  - System scales to the number of users and size of analysis required for LHC*b*.
- The Ganga framework makes it trivial to perform testing on local system and then transfer to the Grid for full scale analysis.
  - Start-up time in Grid system hence not an issue.

LHC*b* has demonstrated a working system for distributed analysis.

Documentation of Ganga system is available at http://cern.ch/ganga.

Backup slides

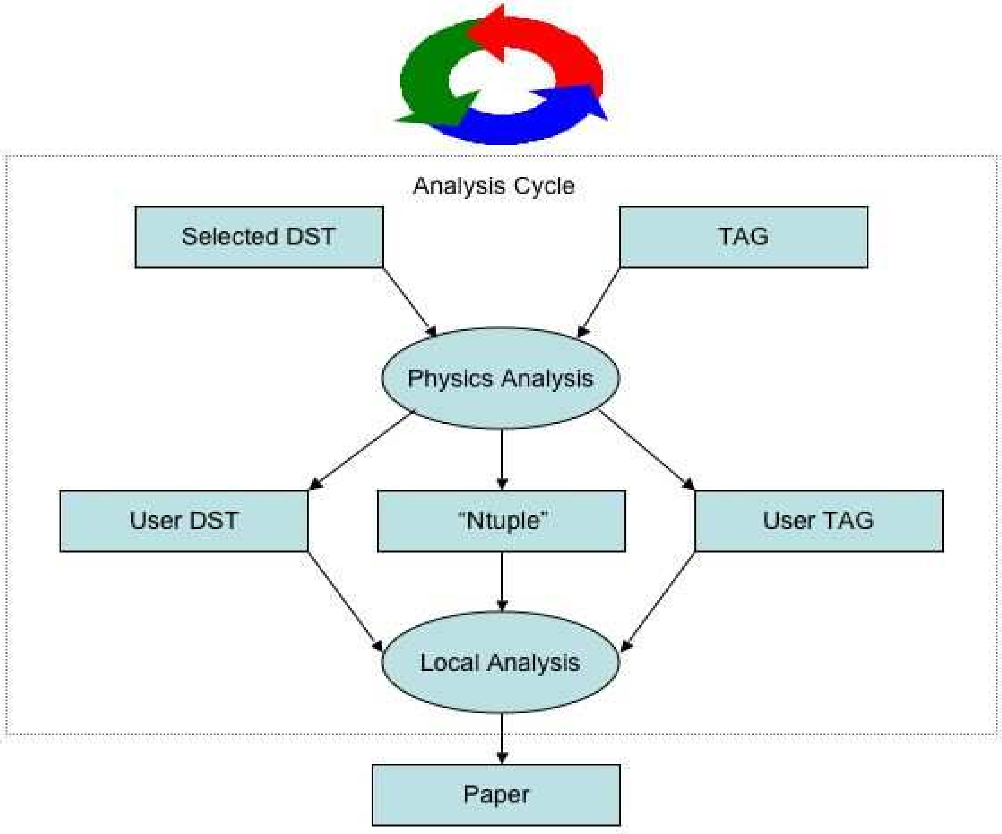
Dataflow - analysis

User physics analysis will be primarily performed on the output of the stripping

Output from stripping is self-contained i.e. no need to navigate between files

Analysis generates quasi-private data e.g. Ntuple and/or personal DSTs

Data publicly accessible - enable remote collaboration

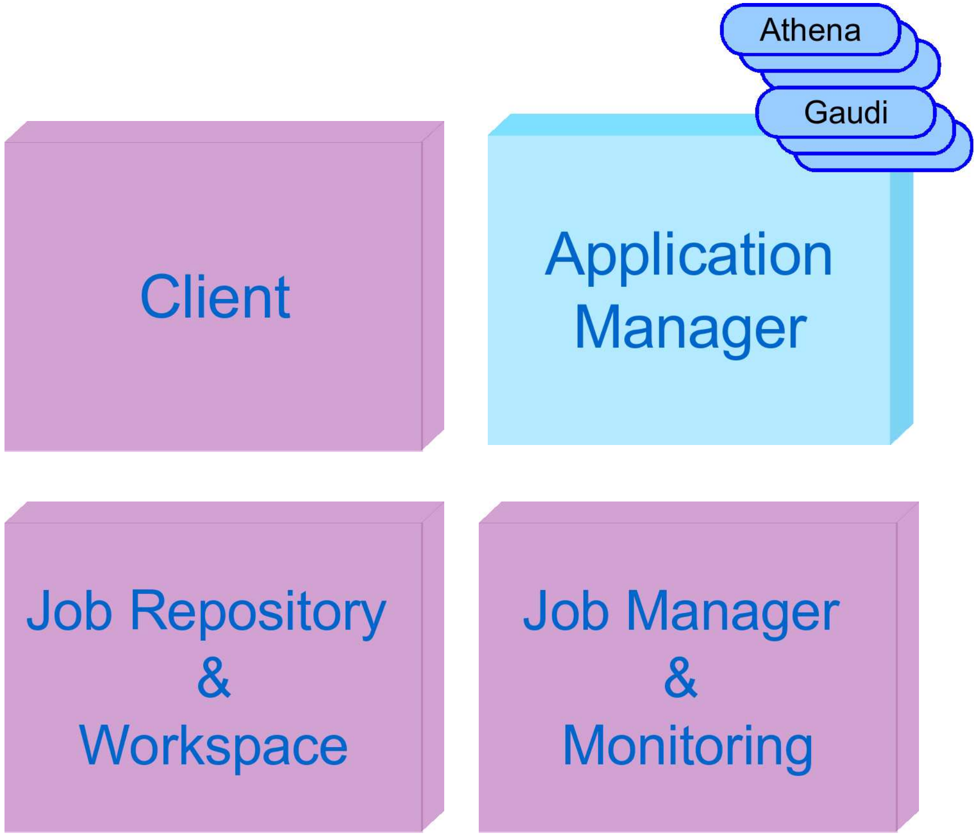


Analysis

- User analysis accounted in model predominantly batch
  - ~30k jobs/year
- Predominantly analysing 10^6 event but assume some analyses 10^7
  - CPU of 0.3 kSI2k.s/evt
  - Analysis needs grow linearly with year in early phase of experiment

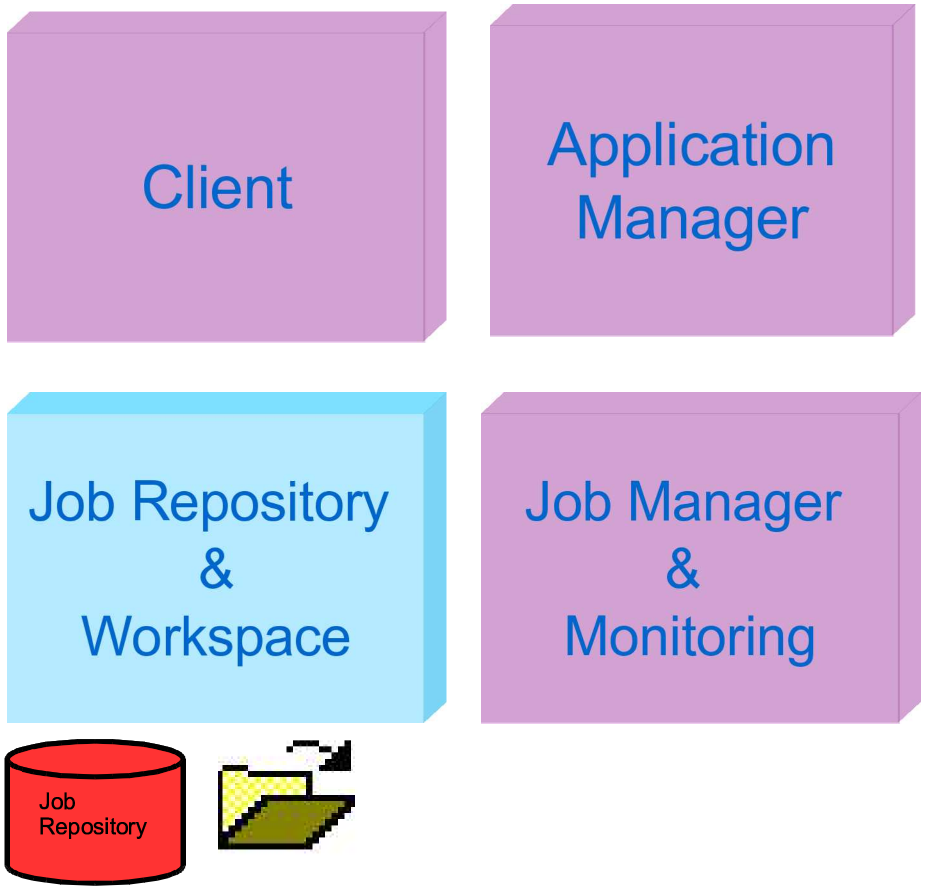
Nos. of physicist performing analysis	14
Nos. of analysis jobs per physicist/week	4
Fraction of jobs analysing 10^6 events	80%
Fraction of jobs analysing 10^7 events	20%
Event size reduction factor after analysis	5
Number of “active” Ntuples	10
2008 CPU needs (MSI2k.years)	0.78
2008 Disk storage (TB)	200

Application Manager



- Prepares and configures the application, e.g.:
  - pre-process options
  - compile user code
  - setup environment
- The work is done by Application Handlers.
  - An application handler is specific to a framework in an experiement – the Gaudi framework for LHC*b*.

Job Repository and Workspace



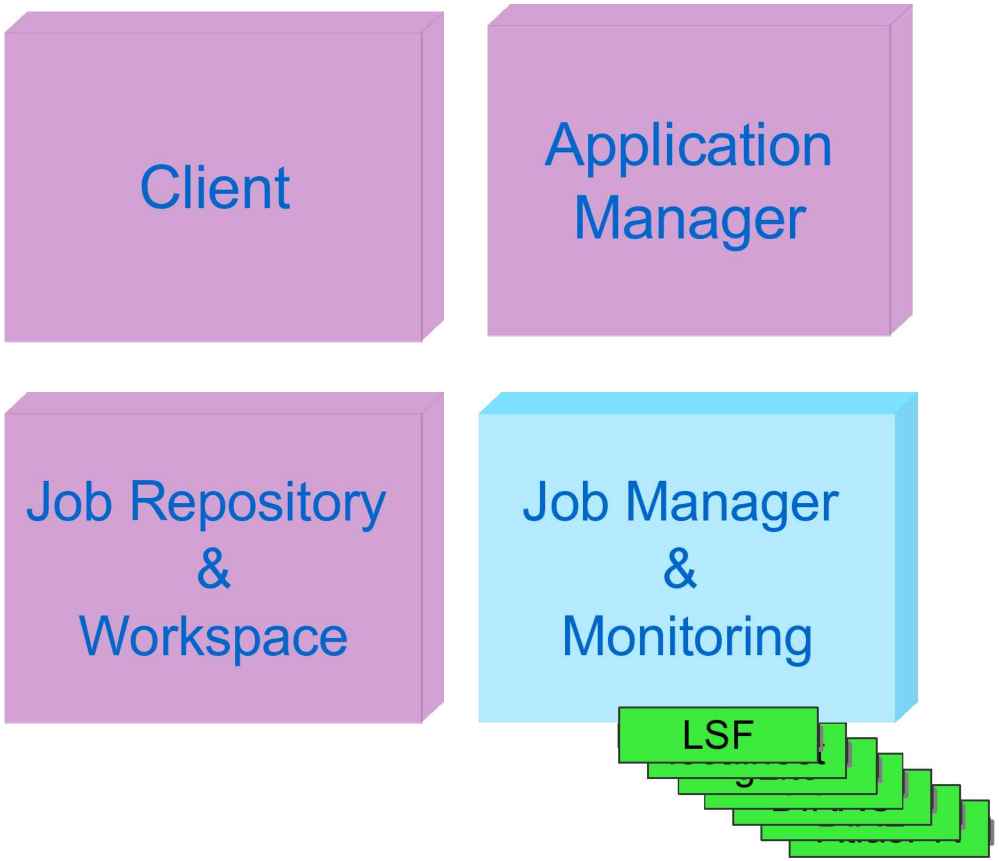
Job Repository:

- keeps track of jobs
- stores job metadata
- “roaming-profile”
- may be Local or Remote

File Workspace

- caches the job output
- allows to share files within Ganga

Job Manager



Submits configured jobs to the submission backends using Backend Handlers, e.g.:

- creates wrappers scripts
- created JDL files
- etc.

Monitors the status of jobs

- by polling the backends
- by monitoring notifications

Notifications allow to see changes in jobs submitted outside of Ganga