U. Egede
Imperial College
London

Status of distributed analysis for LHCb

Presented by Ph.Charpentier

21 October 2004

What is analysis in LHCb?

The aim of LHCb is to extract results on *CP* violation from rare *B*-meson decays.

- Detector will see the full event rate of 40 MHz from LHC collisions.
- When data leaves the detector the rate should be around 200 Hz.
  - This is the job of the trigger and not for discussion today.
- Output data in DST format will be divided into streams each containing maybe 10^7 events per year.
  - These are the events that the end user will access for analysis.
  - Size for DST is about 150 kB per event.
- For analysis we also need access to simulated data.
  - 10 times higher statistics.
  - 2 times larger size.

Status of Data Challenge '04

There are 3 steps in DC '04

Step 1: Distributed production of a large dataset

- Finished, but keep alive at low rate to be able to restart quickly.
- Ran on a mixture of DIRAC (agent based) and LCG resources.
- Status presented in many other places – see CHEP04 talks for a good overview.
- LCG2 overall efficiency: 61% -> too low for analysis!

Step 2: Stripping

- Take the data produced in step1 and select events for individual physics streams.
- Each stream will select below 0.1% of generic *B* data.
- Takes place at the Tier 1 centres (if SRM works, otherwise CERN)
- All output will be distributed to all Tier 1 centres.
- This is where we currently are in DC '04.

Status of Data Challenge '04

Step 3: Distributed analysis

- We want to demonstrate that any physicist within LHCb can run an analysis on distributed resources.
- Work pattern for this will briefly be:
  - Develop GAUDI algorithm for data analysis
    - Debug and test on small dataset on local disk
  - Define dataset to analyse
    - Selection through bookkeeping system
    - Divide dataset into bits that fit individual analysis jobs
  - Submit analysis
    - No specific knowledge of data location or at which Tier 1 site the analysis is running on should be required.
  - Retrieve results
    - Get the output from jobs back to a single point and merge the data.

A user view of an analysis – create algorithm

- An analysis will define a work flow of algorithms to run.
  - The algorithms will be written mainly in C++ and run inside the Gaudi framework.
    - Possibility to write algorithms in Python as well but no widespread use.
- Code for analysis will be a mixture of
  - Standard LHCb algorithms
    - Under version control
  - User modified standard algorithms
    - This is typically for development of the standard tools
  - User specific algorithms
    - These might be different from job to job.
    - No version control.
    - This is a major difference to production style jobs.

A user view of an analysis – configure job

LHCb jobs are configured through a set of options files.

- These files have no state and can be pre-processed in a static way.

For analysis these options might be different from one job to the next.

```
ApplicationMgr.TopAlg += { "PreLoadParticles" };

// Set to use the CombinedParticleMaker
PreLoadParticles.PhysDesktop.ParticleMakerType = "CombinedParticleMaker";

// Default values for particle types to be made
// Note that when exclusive mode is selected the selection is done in
//the order set below
PreLoadParticles.PhysDesktop.CombinedParticleMaker.Particles = {"kaon", "pion" };

// Default value for particles to be selected exclusively or not
PreLoadParticles.PhysDesktop.CombinedParticleMaker.ExclusiveSelection = false;

// Default value for selection of kaon
//  PreLoadParticles.PhysDesktop.CombinedParticleMaker.KaonSelection = {
//   "det='RICH' k-pi='2.0' k-p='-2.0'" };

PreLoadParticles.PhysDesktop.CombinedParticleMaker.KaonSelection = {
  "det='RICH' k-pi='-5.0'" };
PreLoadParticles.PhysDesktop.CombinedParticleMaker.PionSelection = {  };
//  "det='RICH' pi-k='-5.0'" };
```

A user view of an analysis – test and run algorithm

- For testing and running analysis the average physicist will always do what seems the easiest way to get a result tomorrow!
  - Interaction with complicated systems that might bring long term time savings never gain wide acceptance.
  - So the default behaviour for testing analysis is to fork jobs from the command line.
- For running larger scale jobs the incentive will always be to use what worked yesterday.
  - For larger scale analysis jobs this means the PBS batch system at CERN.
- To change the behaviour to be for GRID jobs we can.
  - Make it easier to use than LSF
    - Tough but should be our final goal.
  - Limit the available CPU and data resources at CERN.
    - If (usable) alternatives not in place we will get very unpopular!

A user view of an analysis – define dataset

- Dataset for analysis needs specification.
- Selection can happen in many ways
  - Fixed list taken from static web page.
  - Selection in a database
    - "*The B to D_sK background calibration set*".
  - "*Same data as I used yesterday*".
  - Specific event
    - Event 2134539 from run 3473.
  - "*My usual test data*"
- Notice that none of these cases contain anything about LFN, PFN or other file specific information.
- User will often be blind to if they read the data directly or read an event summary file that simply points to the events.

A user view of an analysis – keeping track and merging

- For production of simulated events we know the importance of very good bookkeeping.
- In an analysis situation users love it if it comes for free.
  - The default way of working is to sort out in retrospect which jobs failed, which produced corrupt output etc.
- The last stage of an analysis is to pull all the results together.
  - Merge all histogram files.
  - Chain ROOT output files.
  - Normalise to analysed luminosity.
  - Search stdout for specific patterns.
    - Not nice but seen a lot.

User requirements for distributed analysis system

- For all users:
  - Responsive
  - Robust
  - Reliable
- In addition for new users
  - Intuitive.
  - Clear error reporting.
  - Transparent to changes in middleware.
- For more powerful users
  - Ability to deal with Event data collections.
  - Scripting language for job control.

LHCb requirements

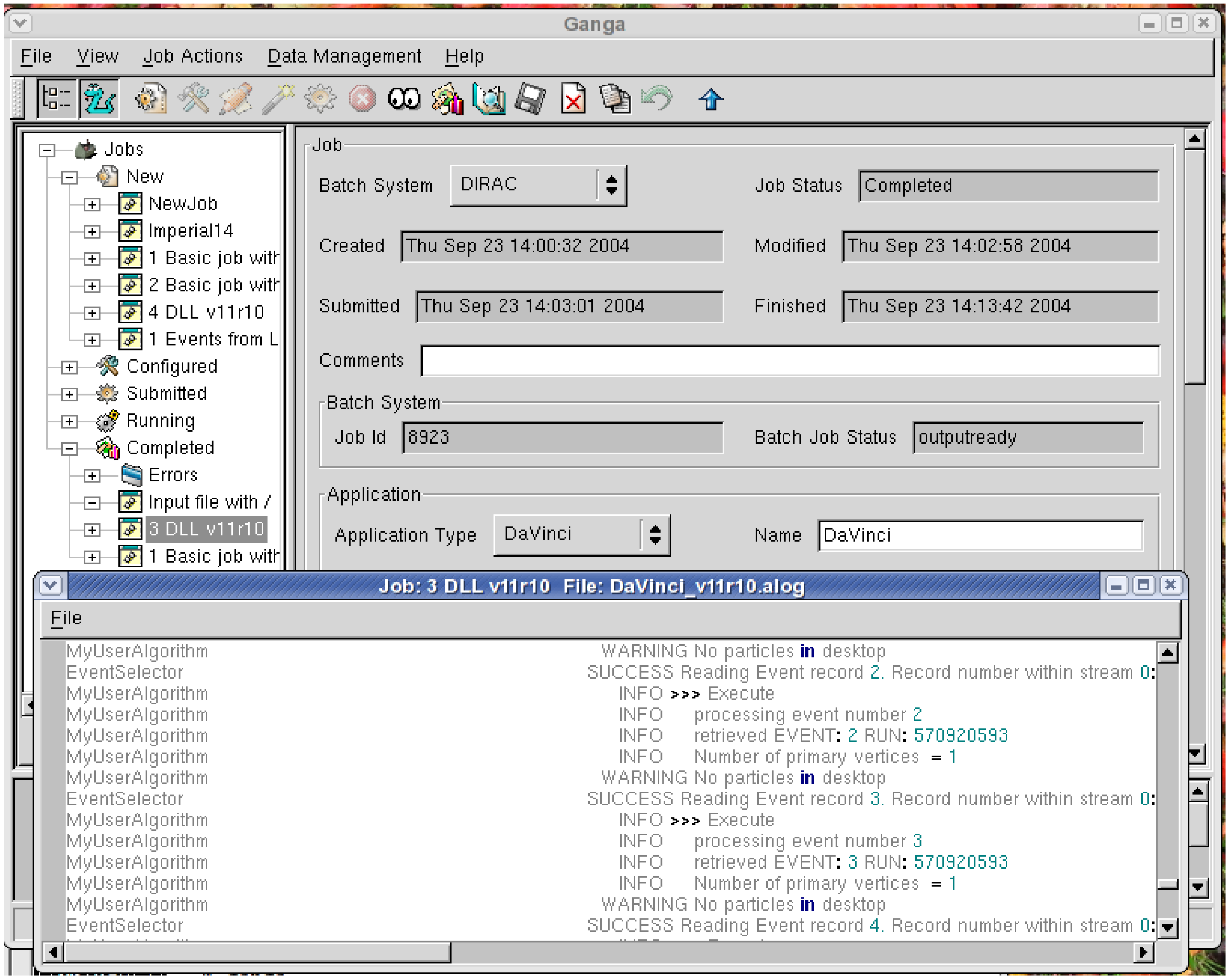
- Ability to set priorities
  - Within LHCb we need the ability to control how resources allocated to LHCb are used.
- Accounting
  - From our current production we have seen the efficiency of monitoring to track down errors and wrong configurations in hardware/LCG/jobs.
  - Monitoring at the job level required of:
    - CPU usage
    - Memory usage
    - Data requests
    - Efficiency
    - Users

The GANGA project

- The purpose of GANGA is to work as a wizard for LHCb users running GAUDI applications.
  - Mainly we have running C++ analysis applications (DaVinci) in mind.
  - Further analysis (ROOT) is expected to happen locally (low number of events, small events), although files will reside on the Grid.
- We need to support the following behaviour:
  - writing new Gaudi algorithms, either in C++ or Python
  - modifying existing algorithms;
  - modifying job options of existing/new algorithms.
- Data will now (and in the future?) be distributed mainly at Tier 1 centres but we should support Tier 2 and local data as well.
  - The input data will be data in POOL format.
  - Output from distributed analysis will be a combination of data in POOL format, ROOT/HBOOK files and stdout/stderr.

A quick reminder of GANGA

- Jobs in Ganga moves between different states.
  - This gives the bookkeeping
- The user can monitor jobs in the states:
  - New
  - Configured
  - Submitted
  - Running
  - Completed
  - Error – if reported!
- Jobs are preserved between invocations.
  - Status updated at regular intervals.



Batch System	DIRAC	Job Status	Completed
Created	Thu Sep 23 14:00:32 2004	Modified	Thu Sep 23 14:02:58 2004
Submitted	Thu Sep 23 14:03:01 2004	Finished	Thu Sep 23 14:13:42 2004
Comments			
Job Id	8923	Batch Job Status	outputready
Application Type	DaVinci	Name	DaVinci

Job: 3 DLL v11r10 File: DaVinci_v11r10.alog

File

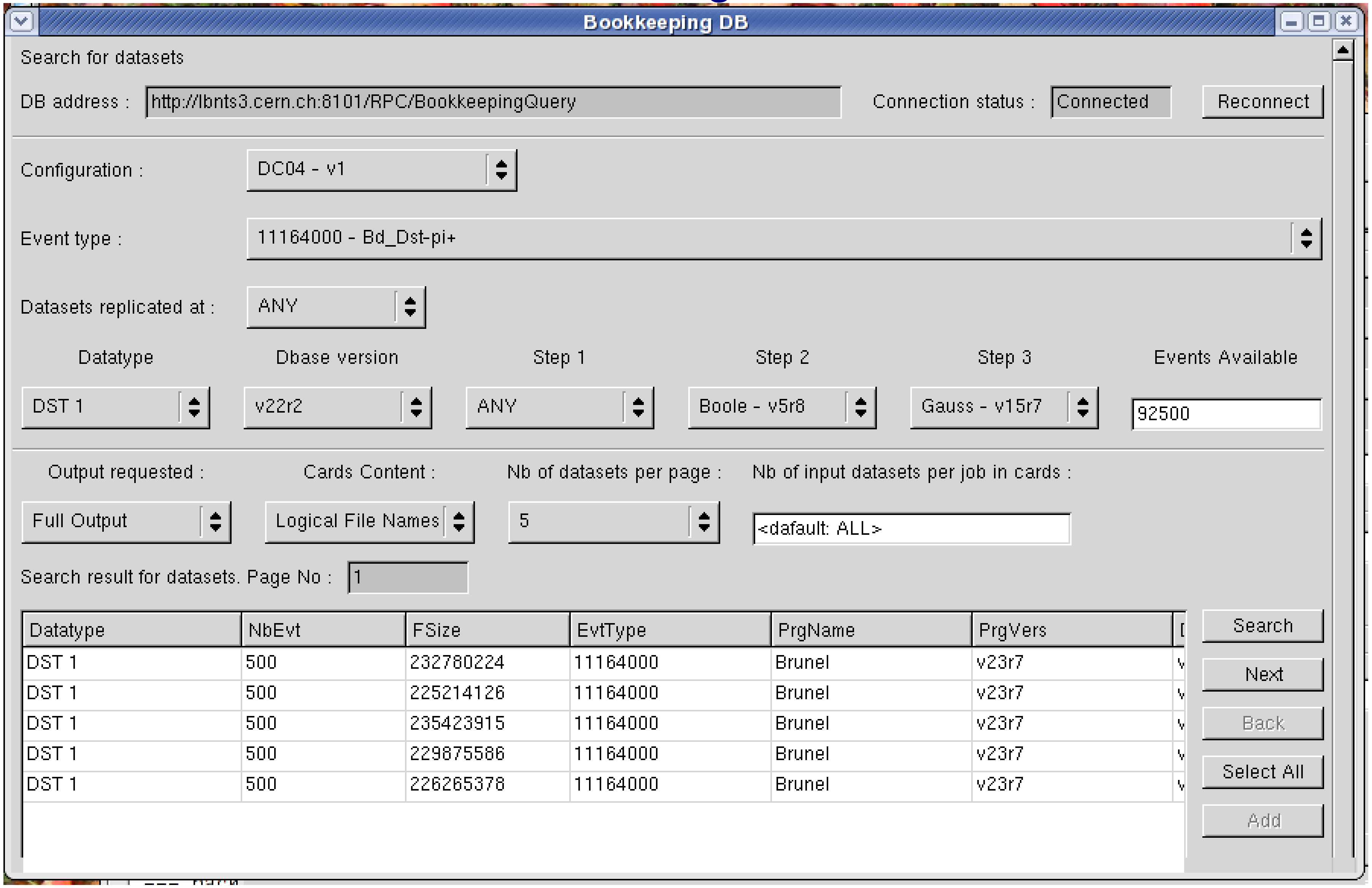
```
MyUserAlgorithm                                  WARNING No particles in desktop
EventSelector                                   SUCCESS Reading Event record 2. Record number within stream 0:
MyUserAlgorithm                                    INFO >>> Execute
MyUserAlgorithm                                    INFO     processing event number 2
MyUserAlgorithm                                    INFO     retrieved EVENT: 2 RUN: 570920593
MyUserAlgorithm                                    INFO     Number of primary vertices  = 1
MyUserAlgorithm                                  WARNING No particles in desktop
EventSelector                                   SUCCESS Reading Event record 3. Record number within stream 0:
MyUserAlgorithm                                    INFO >>> Execute
MyUserAlgorithm                                    INFO     processing event number 3
MyUserAlgorithm                                    INFO     retrieved EVENT: 3 RUN: 570920593
MyUserAlgorithm                                    INFO     Number of primary vertices  = 1
MyUserAlgorithm                                  WARNING No particles in desktop
EventSelector                                   SUCCESS Reading Event record 4. Record number within stream 0:
```

A word on “catalogs”

- Metadata catalog: BKDB
  - Current implementation well tested in DC03 and DC04 (several million entries)
  - Uses specialised views on top of a generic set of tables (based on key-value), depending on the type of usage. Very good scalability
  - Includes for the time being a replica table as well
  - Interested in developments in ARDA (also to input ideas)
    - In contact for a joint meeting
- File Catalog
  - Two brands populated in parallel: BKDB and AliEn
  - Single interface to both (using XML-RPC)
  - Expect to be integrated within POOL through a simplified interface
  - Expect also an easy way to create XML slices for simple usage (no navigation)

Interaction with LHCb bookkeeping database

This allows a user to pick the data for analysis directly from within GANGA, save it in a favorite's catalog.



Search for datasets

DB address : http://lbnts3.cern.ch:8101/RPC/BookkeepingQuery

Connection status : Connected

Configuration : DC04 - v1

Event type : 11164000 - Bd_Dst-pi+

Datasets replicated at : ANY

Datatype	Dbase version	Step 1	Step 2	Step 3	Events Available
DST 1	v22r2	ANY	Boole - v5r8	Gauss - v15r7	92500

Output requested :	Cards Content :	Nb of datasets per page :	Nb of input datasets per job in cards :
Full Output	Logical File Names	5	<dafault: ALL>

Search result for datasets. Page No : 1

Datatype	NbEvt	FSize	EvtType	PrgName	PrgVers	[
DST 1	500	232780224	11164000	Brunel	v23r7	v
DST 1	500	225214126	11164000	Brunel	v23r7	v
DST 1	500	235423915	11164000	Brunel	v23r7	v
DST 1	500	229875586	11164000	Brunel	v23r7	v
DST 1	500	226265378	11164000	Brunel	v23r7	v

Getting the analysis job to the Grid

- Our aim is to get LHCb analysis to run on LCG resources
  - LCG2 today, gLite/EGEE, LCG3…
- Obvious solution is to submit jobs directly to the LCG2 resource broker.
  - For several reasons we do not pursue this direction.
    - LCG UI impossible to install.
    - No way for LHCb to keep track of analysis… and 60% probability to get it back
  - Solution chosen is to use our infrastructure of production system (DIRAC) as a portal to LCGn
    - We get most monitoring/accounting for free.
    - Only trivial DIRAC UI to install (one click). Shipped with Ganga
    - We can use the experience of running production jobs on LCG.
    - Will enable running on DIRAC sites as well.
  - Looking forward to using gLite (pre-)production system
    - Directly or through DIRAC
    - Catalogs should be accessible as well as data!

Getting the analysis job to the Grid

- General status
  - Submission of jobs to DIRAC works without any problems
  - Long standing problem with upload of user created shared libraries solved.
  - Creation of XML POOL catalog slices to read data defined with Logical filenames needs further testing.
  - Upload of user job options files works as required.
- Running on DIRAC (agent) site
  - Jobs are picked up by the agent and run without problems.
    - Some mismatch in status flags between DIRAC and GANGA concerning run status and if job finished in error or not.

Getting the analysis job to the Grid

Running on LCG site

- On the machine running GANGA client the user needs to obtain a grid proxy
  - We have small (700 kB) tarball that can do grid-proxy-init (and a few other related commands) that can be distributed with GANGA.
  - No need for full GT2 installation.
- If proxy is available if will be uploaded in encrypted form to the DIRAC WMS.
- Job is then submitted to LCG from DIRAC WMS.
  - This scheme will not support any type of proxy renewal
- All pieces tested individually but the full chain has not yet been demonstrated.

Getting the analysis job to the Grid

Running on Glite

- See Andrew Maier's presentation from yesterday.
- For future progress in this area we want to take advantage of the modular structure of EGEE middleware.
  - Want to be able to use the DIRAC WMS instead of the default supplied by gLite (we have all the infrastructure to interact with it)
  - Plan then to look at relative advantages. No “religious” opinion on which WMS we will use.
- Feedback on the EGEE design document.
  - Some initial feedback has already been given through ARDA.
  - We are in the process of evaluating it in more detail.
- We would like the gLite (pre-)production service to run the current middleware that we had hands on for 6 months.
  - No change of gear at this stage

Command Line Interface

- A GUI is good for providing overview and for new users to learn about distributed analysis.
- Many advanced users would like to write scripts that perform repeated operations.
- Solution is to have a python based CLI that interoperates with the GUI.
  - A syntax has been defined and a subset of commands are implemented within GANGA.
  - Many additional benefits
    - Will allow test jobs to be defined and run in new releases in an automatic way.
    - Will allow prototyping of new plugins for applications or submission systems to be tested without initially worrying about GUI.
  - Longer term plan is to implement GUI through the API defined by the CLI.

Command Line Interface

A few examples of what you can do in CLI

Create a job:

```
>>> j = Job()
>>> j.application = DaVinciApplication()
>>> j.backend = LSF()
>>> j.describe()
Job #5  [new] {'name': None}
 backend <LSF>{'queue': '8nm'}
 application <DaVinciApplication>{'outputfiles': [], 'inputfiles': [],
'options': None}
```

List jobs:

```
>>> jobs()
Statistics: 8 jobs
------------
 ID   status      name
#  1 completed
#  2     new
#  6 submitted      myjob
#  8 submitted      myjob
```

Status of GANGA for LHCb analysis

Submission	Local	Working
	LSF	Working
	DIRAC	Working
	LCG proxy transfer	In Progress
Job Options	Selection	Working
	Editing	Working
	Expansion	Working
	Logical view	Deferred
Job handlers	Generic	Working
	DaVinci	Working
	User defined templates	Deferred
Data selection	Connection to bookkeeping database	Working
	Creation of XML slice	Needs testing
Installation	AFS	Working
	Local	Working
Job management	Splitting of input data	In Progress
	Merging of output	Deferred
Roaming	Job profile	Deferred
	Options and DLL packaging	In Progress
CLI	Integration with GANGA core	In Progress