

Ganga LHCb Summary

Alexander Richards

Imperial College Sci., Tech. & Med. UK
(IC)

8th – 9th February 2012



Ganga Developer Days, Birmingham

- Introduction
- Recently added/current features
 - Prepared State
 - Tasks for LHCb
 - Splitting GangaLHCb/Gaudi/DIRAC $\sim 80\%$ 1st pass, $\sim 20\%$ 2nd pass
 - LHCb Re-submission Strategy - Discussion later
 - Miscellaneous
- Upcoming features
 - DIRAC Bulk Submit
 - Gaudi XMLSummary
 - Ganga Checkers
- Savannah items to discuss

Under GangaLHCb currently we have 10 applications, 1 backend, 4 splitters, 1 task & 1 transform:

Applications:

Based on Gaudi

- Gauss
- Boole
- Brunel
- DaVinci
- Moore
- Vetra
- Panoptes
- Erasmus

Based on common base class

- GaudiPython
- Bender

Backend:

- Dirac ~ WMS/DMS

Splitters:

- SplitByFiles
- OptionsFileSplitter
- GaussSplitter
- DiracSplitter (Smart splitting of datasets)

Tasks

- LHCbAnalysisTask

Transforms

- LHCbAnalysisTransform

Prepared State

Prepared state needs no introduction. It has already transitioned into Core and most of us are aware of it and making preparable apps.

From LHCb side:

- Implementation of the prepared state for all LHCb applications complete.
- None of our apps define attributes that are modifiable post-prepare.
- Use the prepare to do many of the steps that used to be in master_configure which are data independent.
 - Taring up environment
 - Reading and pickling application options files.
 - Copying and local application configurations.
 - Attaching inputdata from the options files - **Thorny issue!**
- All items stored in share dir (with appropriate directory structure) to share between jobs.

Motivation for LHCb Tasks

- Users have access to experiment specific tools that provide a list of LFN(s) they are interested in
- Accommodating changes to the dataset is currently not easy
 - Corrupted or lost data is sometimes removed from the Bookkeeping
 - New data can also appear in real time as the detector takes data and it is processed
- The Grid is an unstable environment to run jobs and failures will occur, the process of recovering from job failures and achieving 100% success requires manual effort

Tasks framework for LHCb provides users:

- Ability to have many different analyses set up which monitor and keep up to date with different datasets
- Two levels of job resubmission (job level and partition level) to help ensure all user jobs are submitted.

Tasks for LHCb

Tasks in LHCb maps the ideas of *Task*, *Transform* and *partition* objects as:

- LHCbTask → container object holding and responsible for the running of one or more transforms
- LHCbTransform → essentially equivalent to a Ganga JobTemplate with addition of a dataset query attribute. It manages the setting up and re-running of *actual* jobs as the dataset catalog changes.
- Partition → equivalent to a Ganga master job. New one submitted when data catalog is updated.

Since LHCb *probably* not using Tasks for the purpose they were originally invented (ATLAS), the syntax and nomenclature may be somewhat laboured however this is what was inherited. In addition, since the user will mainly interact with the LHCbAnalysisTask and LHCbAnalysisTransform object this shouldn't be too much of a problem (esp. given the similarity of the transform to the jobs that they are used to).

By way of example...

```
In[0]: tr = LHCbAnalysisTransform()
In[1]: tr.application = DaVinci()
In[2]: tr.backend = Dirac()
In[3]: tr.splitter = DiracSplitter()
In[4]: tr.splitter.filesPerJob = 50
In[5]: tr.query = BKQuery(favourite dataset) - Only difference from
setting up a job
In[6]: t = LHCbAnalysisTask(float=10)
In[7]: t.appendTransform(tr)
In[8]: t.run()
```

From the user's point of view, very little overhead from setting up a regular job but with all the benefits.

Tasks for LHCb

- Given the slightly tweaked idea of the LHCb tasks, the overview printout was modified as below.

```
In [3]:t.overview()
Partition Colours:      , ready, running, completed, attempted, failed, bad, unknown
Job Colours: new, submitting, submitted, running, completing, completed, failed, killed, incomplete, unknown
Lists the transforms, their partitions and partition subjobs, as well as the number of failures.
Format: (partition/subjob number)[:(number of failures)]

Transform 0: LHCbAnalysisTransform 'task testing'
-----
Partition 0 (attached to job# 206, containing 3 datafiles):1
0:0 1:1 2:1
Partition 1 (attached to job# 210, containing 3 datafiles):0
0:0 1:0 2:0
```

- Note: All LHCb applications are now 'taskified'

Splitting GangaLHCb/Gaudi/DIRAC

In previous meeting in Munich Mike W. raised issue of possible refactoring of Dirac into core.

http://indico.cern.ch/conferenceOtherViews.py?view=egee_meeting&confId=94195

This is desirable because:

- DIRAC is not LHCb specific but a general WMS/DMS so in some sense belongs in Core.
- Makes us (as a tool) more attractive/scaleable to new collaborations who wish to use DIRAC without the LHCb overhead.

This however can also be extended to *Gaudi*, the analysis application framework on which most of LHCb applications are based (and also ATLAS' *Athena* application).

To this end...

- Begun refactoring GangaLHCb code into three Packages:
 - GangaGaudi
 - GangaDirac
 - GangaLHCb
- This was started \sim 3 months or so ago just as the prepared state was in transition from branch \rightarrow trunk.
- \sim 80% was refactored but since 5.7.X many fixes and changes have had to be put in to trunk to correct bugs introduced which has meant keeping branch up to date has been tricky.
- Since these changes have gone in I have learned a lot more about the LHCb code base. Merging these changes in is tricky to make sure it is done correctly so having to check every *major* file and also with better understanding I have been able to make further re-factorisations.

LHCb Re-submission Strategy

In LHCb we have an issue surrounding re-submission.

- Our LFN \rightarrow PFN Catalog is generated upon job submission (run time handler)
- Standard job re-submission re-sends job straight to backend.
- LFN \rightarrow PFN Catalog out of date if job resubmitted some time later

Might be possible to move the LFN \rightarrow PFN catalog generation into backend. (Haven't looked at possibility of doing this). This however ties the backend into some knowledge of the application that it is submitting. This seems to contradict the splitting and refactoring philosophy from before.

With the advent of the prepared state we can guarantee that the application is in a given pre-prepared state at resubmission so we can simply call `job.submit()`. By default this is stopped however, by the block on submission of jobs not in the *new* state.

Job.resubmit() solution implemented so far...

- revert job to status 'new'
- call `job.submit()`

This is intended to kick-start a more general discussion about whether our re-submission strategy makes sense/needs to change in the context of the new prepared states.

Numerous minor changes & fixes

- Hard coding limit number of DiracSplitter datasets
- ROOT version taken from LHCb setup environment ~ P.Owen (IC)
- Fixed 'A' conflict between monitoring and user thread in the LHCb Tasks framework
- SLC 6 became default supported version as of 1st February 2012, this includes as standard python 2.6.6. I have run a *quick* test of GangaLHCb under python 2.7 and nothing major broke and jumped out at me... ~ more later

A green highway sign with white text and an arrow pointing up and to the right. The sign is mounted on a metal structure against a blue sky and a yellowish ground.

The Future

NEXT EXIT



- Currently large (split) job submission in LHCb is quite slow.
- The main reason for this is the time taken to individually submit each sub job to DIRAC. $\sim 1 \text{ sec/job}$
- This quickly adds up e.g. to submit 300 jobs $\sim 5 \text{ mins}$.
- During this time user may not exit Ganga else causes problems.

Clearly solution is some form of bulk submission. DIRAC now has this in the form of a *Parameterised Job*.

Must consider

- Way to make this either obvious or invisible to the user.
Implementation detail
- What safeguards are necessary in order to stop silly users.
Submitting 1M jobs in same time as 1!

- The *Gaudi* software framework on which LHCb applications are based can return an XML summary object with specific information in it such as run luminosity, number of events processed etc.
- At present there is a mechanism in place for parsing this summary and allowing the user to retrieve the information
 - It is suffering from performance issues as it is old and hasn't been touched in ages.
New faster implementations exist now
 - It is not readily used by many people.
Possibly due to lack of knowledge/ease of use

Ways forward...

- We wish to re-factor this code and integrate it more closely with the existing user experience, eg could imagine:
 - `job.application.luminosity`
 - `job.application.run_events`
- We will as much as possible draw from the newer faster implementations.
- This solution would mean changing the application after a prepare (ATLAS do it).
- Maybe there is a cleverer way. ~ **Would love `job.run_events`**
 - Could consider an `LHCbJob` subclass of `Job` but might catch users offguard and take them a while to switch.
 - Maybe add them into `job.__dict__` ?

In general, basing the determination of a job's success or failure on its exit status alone:

- is inflexible
- can lead to false positives

R.Lambert suggests the idea of checkers.

The idea...

- Define some new category of Ganga object called *checker*
- Checkers could then be defined as predicates that indicate a job's completed status in addition to the exit code.
 - Have some logic like:
`if job.checker and (exit_code is 0): job.updateStatus('complete')`
`else: job.updateStatus('failed')`
- Could envisage some sort of MultiChecker object which could chain multiple checker objects in a logical AND

Savannah



<http://gna.org/projects/savane>

For LHCb we have 10 outstanding tickets:

Item ID	Summary	Submitted On	Assigned To	Submitted By
#90084	Conflict between bookkeeping GUI and job auto-resubmit	2011-12-21 15:35	None	jonrob
#90052	Better handle of large file /outputdata	2011-12-20 14:09	None	jhe
#89657	DiracSplitter allows me more files than Dirac	2011-12-07 08:56	arichard	pkoppenb
#88188	Regenerate LFN->PFN catalog on resubmission	2011-10-27 10:12	arichard	raaij
#88162	Auto job submission and LogicalFile.download(...) conflict	2011-10-26 13:45	None	jonrob
#87644	combine splitters	2011-10-11 08:10	None	arichard
#75807	Make Ganga POSIX compliant	2010-11-29 09:18	jwilliam	None
#74131	Add support for general configuration file packages	2010-10-18 15:14	jwilliam	jonrob
#71139	Possible to 'fail' a sub-job if not all input files were processed	2010-08-06 10:15	jwilliam	ukerzel
#46949	Upload of input sandbox to SE	2009-02-13 11:06	jwilliam	uegede

- Combining splitters
- python 2.6.6 as of 1st Feb in SLC6
- POSIX compliance.
- General configuration
- Conflicts!