Architecture of the LHCb Distributed Computing System

Federico Stagni, Philippe Charpentier
On behalf of the LHCb collaboration
LHCbDirac: What and why
The “DIRAC” (Distributed Infrastructure with Remote Agent Control) INTERWARE is a software framework for distributed computing providing a complete solution to one (or more) user community requiring access to distributed resources. DIRAC builds a layer between the users and the resources offering a common interface to a number of heterogeneous providers, integrating them in a seamless manner, providing interoperability, at the same time as an optimized, transparent and reliable usage of the resources.”

(See A. Tsaregorodtsev 2 days ago)

http://diracgrid.org
What's LHCbDirac (a.k.a. BeautyDirac)

- Dirac consists of a set of collaborating agents and services
  - Using its own secure communication protocol (DiSeT)
- DIRAC grew within LHCb, but now:
  - There’s not only LHCb
    - Still, the biggest community
  - Each community can decide to create its own extensions
    - Independent release cycles
  - Beauty/LHCb-Dirac extension: LHCb-specific code
LHCb activities and how to handle them

LHCb distributed computing activities:
(see presentation from C. Haen yesterday)

“Production” (central) activities:
- Reconstruction
- Reprocessing
- Stripping (selection of events)
- Monte-Carlo simulations
- Indexing

Non-“Production” activities:
- User analysis
- Monitoring and testing

How to handle:

- Using a Bookkeeping system (BK) for the data provenance and dataset retrieval (see poster on LHCb Data Management)
- Extending the DIRAC Transformation system
  - Using the BK for retrieving datasets
  - Implementing its own task creation plugins
- Implementing a productions’ requests system
- Automating at most production management
(LHCb)Dirac Components and concepts
A data provenance catalog
- Contains all files ever created in LHCb and information about how they were created
  - Raw data as well as derived or simulated data
- Not specifically a tool for distributed computing
- Main tool for retrieving datasets
  - Users (for analysis) and production system
    - Conditions (data taking, simulation)
    - Processing (applications, detector condition parameters)
    - Event type
    - File type
- Fully integrated in LHCbDirac
  - Based on DIRAC services and DiSeT
- Oracle backend
  - High level of optimization (indices, views)
- **Job description format**
  - Enables running “complex” jobs
    - e.g. multiple applications, linked together via input/output data
    - I/O chaining
  - Description in different formats: XML, JDL, python
    - JDL executable: dirac-jobexec
    - Argument: jobDescription.xml (which is in the Input Sandbox)
- A workflow is composed of steps
  - that are made of modules
  - Workflow modules are instantiated by python modules
    - that do the real job
  - Parameters at any level

---

Job description

Application Step 1
Application Step 2
Finalization Step

(for users AND production jobs)
A system for handling “repetitive” work:

- Handles input datasets (if present)
  - Plugins are grouping input files into tasks according to various criteria
  - Tasks are created

2 main usages:

- Productions: the same job - i.e. the same workflow - is executed
  - Client for the Workload Management System
- Data Handling: replications, removals
  - Client for the Data Management System
  - See Poster #02
Transformation System architecture

1. InputData Query
2. InputData Agent
3. Files
4. Transformation Agent
5. Plugins
6. Tasks
7. Transformation (definitions)
8. Workflow Tasks Agent
9. Request Tasks Agent
10. Database table
11. Agent

WMS

DMS

ISGC2015, F. Stagni and Ph. Charpentier, CERN
LHCb Production Requests system

- A way for specific users to make requests for data processing or MC simulation
  - Includes an authorization mechanism for physics relevance (and technical acceptance)
- Formalization of the data processing activities

A list of application steps define a production request (with I/O chained)

Application steps can be grouped in productions
  - As many productions as convenient
    ✴ Intermediate datasets must be saved (temporarily at least) on storage
    ✴ They can be destroyed by the next production if required
Managing the whole cycle

Operationally:
1. Application managers create steps
2. Production Managers (PMs) or Application Managers create requests
3. PMs launch requests using production templates
4. Productions are followed by shifters, GEOC (Grid Expert On Call) and the same PMs
Managing Productions
Simulation productions go through a testing phase before being submitted

A limited amount of jobs are created and submitted to a site chosen for testing
- each job produces a fixed amount of events

Productions undergoing a testing phase are monitored by a dedicated agent
- when all jobs are finished, an evaluation takes place:
  - If all jobs failed, the production request is rejected
  - If jobs are successful, the following results are evaluated
    - CPU-work (in HS06.s) per event (CPUe) is calculated
    - Job description is modified: CPUe is added, destination is changed...

Simulation productions have to produce at least a requested amount of events
- In case not enough events have been produced, simulation productions are automatically extended
Elastic Simulation jobs

1) We (just) discovered that we can run up to here..

2) And we already used up to here..

3) We'll try and fit a job there, no matters what

- Given the CPU-work available and CPUe, compute how many events may be simulated
  - 30% safety margin to cope with uncertainties in CPU power estimate
Elastic MC jobs in practice

CPU Work vs Number of Events

Entries: 26667
Mean x: 817
Mean y: 5.003e+05
RMS x: 215.6
RMS y: 2.031e+05

WNCPUSH06 vs Job Power

Entries: 28711
Mean x: 7.573
Mean y: 7.476
RMS x: 1.611
RMS y: 1.87

Job/HS06
Closing a production

- We can close a production when consistency checks are successful.
- Successfully completed productions are archived:
  - Jobs can be removed
  - Transformation tasks can be removed
  - Job logs can be archived

Bookkeeping

Replica Catalog

Transformation System

SE
LHCb Pilots
“Pilots to fly in all the skys”
LHCbDirac is a DIRAC extension

- Used for all the distributed computing activities
  - One system to rule them all

- Actively developed
  - 2 or 3 minor releases/year
    - Patch releases as frequently as required (weekly on average)
  - 1 major release every 3 or 4 years

- Spread over ~30 servers:
  - 50 services running (~40 different ones)
  - ~100 agents running (~70 different ones)
  - ~10 Executors
  - ~20 DBs
    - Mostly MySQL
      - ~500 GB of data... with high variance for certain DBs
    - Oracle for the bookkeeping
    - ElasticSearch being experimented for real-time monitoring