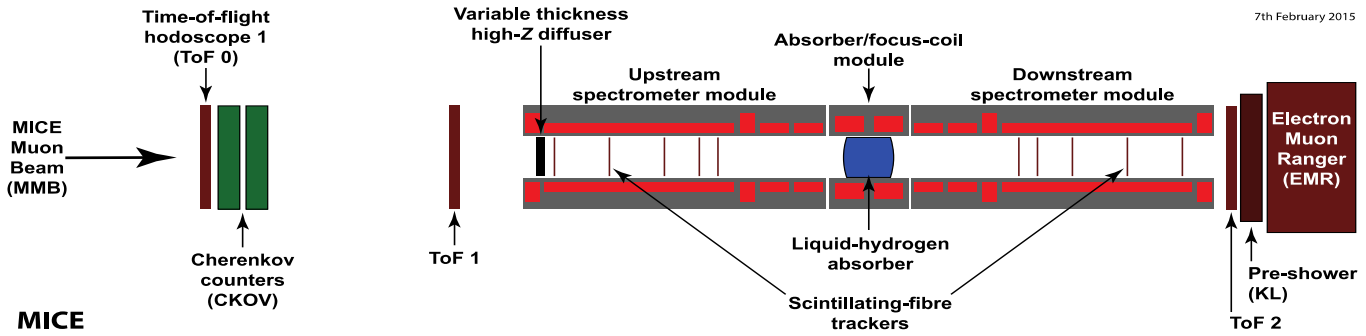
Data Management and Database Framework for the MICE Experiment

Janusz Martyniak, Imperial College London
Henry Nebrensky, Brunel University
Durga Rajaram, IIT Chicago

Muon Ionization Cooling Experiment

Ionization cooling is a technique which allows reducing emittance of charged particle beam. The particle total momentum is reduced by absorbers and then re-accelerated which increases a longitudinal momentum component only, thus "cooling" the beam.

The picture below shows MICE Step4 setup with high precision scintillating-fibre trackers in a 4T field.



MICE Data Handling

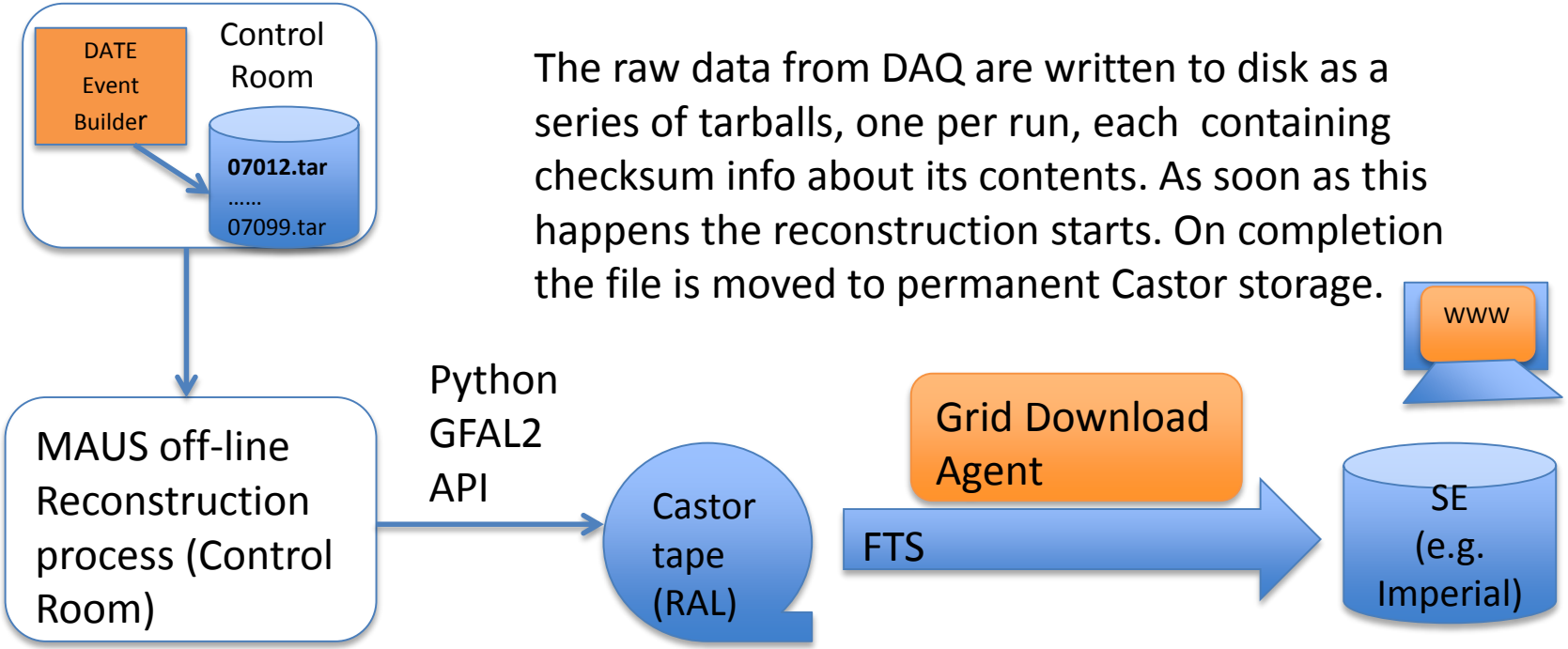
- The original Raw Data Mover has been described in a paper:

MICE Data handling on The Grid
J. Phys.: Conf. Ser. Proceedings of CHEP2013. **513** (2014)
http://iopscience.iop.org/article/10.1088/1742-6596/513/3/032063

Briefly:

1. The data files (tarballs) are verified for integrity and copied to permanent tape storage at RAL
2. The system is written in Python and has been upgraded to use gfal2 API (rather than lcg tools), which is more 'pythonic' and robust.
3. We use a hardware token to store certificates
4. Semaphores indicate the status of the mover. Nogios is used for monitoring of backlog.
5. Files are being replicated to Imperial and Brunel.

MICE Data Handling - Reconstruction



The raw data from DAQ are written to disk as a series of tarballs, one per run, each containing checksum info about its contents. As soon as this happens the reconstruction starts. On completion the file is moved to permanent Castor storage.

MICE Configuration Database

- Contains information about the experimental conditions, like geometry description, magnet currents, absorber materials, cabling etc.
- This is vital to assure accurate simulation and reconstruction.
- We have set up a fully replicated, hot-standby database system. It contains a firewall-protected read-write master situated in the MICE Control Room and read-only slave(s) running at RAL.
- We use PostgreSQL as a DBMS.
- The access to the DB is provided a JAX-WS Web Service layer which provides platform and programming language independent access to data.

MICE CDB, (cont.)

- We provide following CDB APIs:
  - C, to allow the Run Control (built on EPICS) system to read detectors parameters (like magnet current) from the hardware and store them in the DB. Use *gSOAP* based clients. Provide a selection of read-write operations used on run by run basis, e.g magnet currents, absorber settings.
  - C++, to be natively used by the MAUS reconstruction system, use *gSOAP* C++ bindings. Provide a selection of read only operations to access detector cabling and calibration details,
  - Python – to store certain predefined condition from the Control Room (like geometries), also widely used by users for analysis, use *suds*. This API provides most complete set of operations, both for writing and reading.

MICE CDB, (cont.)

  - Java, a selection of read-only operations predominantly used to contact the CDB server from a Google Window Toolkit based viewer. Only slave(s) can be contacted this way.

- For security reasons database records are never deleted. They could only be updated (like adding end-run information to an existing run) or a new version of a record can be added (e.g. a new magnet setting template)
- The CDB API is periodically released and the API is bundled with MAUS as a third-party client, but can also be installed stand-alone.

CDB Replication and Recovery

Running PostgreSQL in hot-standby mode enables promoting a slave to the master in an event of a master being unusable. This is a build-in procedure in PostgreSQL servers. We use a following procedure at MICE

1. Reconfigure (promote) a slave to become a master.
2. Start a (firewalled!) read-write WS, so the Control Room can now write to the new master.
3. The new master still maintains a public read-only interface as before.
4. Sync remaining slaves if applicable.
5. When an old master is fixed, swap back (an stop a read-write access started in 2.)

CDB Viewer

http://cdb.mice.rl.ac.uk/cdbviewer/

8370	2016-10-06 23:06:33.98142	2016-10-07 00:08:23.071679
8371	2016-10-07 00:15:48.517844	2016-10-07 00:49:55.424345
8372	2016-10-07 00:55:18.820916	2016-10-07 02:58:02.474007
8373	2016-10-07 03:00:40.786904	2016-10-07 05:05:56.401879
8374	2016-10-07 05:08:23.77035	2016-10-07 08:01:41.08397
8375	2016-10-07 08:03:29.113524	2016-10-07 09:03:37.390514
8376	2016-10-07 09:07:32.605886	2016-10-07 10:18:01.43574
8377	2016-10-07 10:21:28.190678	2016-10-07 11:03:38.349399
8378	2016-10-07 11:05:56.551121	null

Page 142 of 142 Displaying 2116 - 2124 of 2124

Run Number: 8377

Download Run Summary Download Geometry

Beamline Details Coolingchannel Details Geometry Details

Run Number:	8377
Run Type:	Special Run
Start Date:	2016-10-07 10:21:28.190678
End Date:	2016-10-07 11:03:38.349399
Start Pulse:	2764043
End Pulse:	2766019
Target Depth (mm):	0.0
Target Delay:	0.0
Total Beam Loss (mV):	
Daq Version:	DATE_v7.66 EqList_v1.0.0-1-gf902e9e
Daq Trigger:	TOF1

Summary

- We presented an updated raw and reconstructed data file movement systems:
  - written in Python, using EMI gfal2 API for secure transfers,
  - use a proxy created from a certificate stored on a hardware token,
  - use FTS based download agent to distribute data to other Tier-2 Grid sites,
- We described MICE Configuration Database:
  - PostgreSQL DBMS – Master in the Control Room and fully replicated hot-standby slaves elsewhere,
  - Web Service layer provided by JAX-WS deployed on Tomcat,
  - Java clients, python clients (*suds*) and C/C++ clients (*gSOAP*) provided,
  - Google Window Toolkit based CDB viewer.