Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Data management and database framework for the MICE experiment

To cite this article: J Martyniak *et al* 2017 *J. Phys.: Conf. Ser.* **898** 062030

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 062030 doi:10.1088/1742-6596/898/6/062030

Data management and database framework for the MICE experiment

J Martyniak[1], J J Nebrensky[2] and D Rajaram[3] on behalf of the MICE collaboration

[1] Imperial College London, Physics Dept. London SW7 2BW, UK
[2] Brunel University, Uxbridge, UB8 3PH, UK
[3] Illinois Institute of Technology, Chicago, IL 60616, USA

E-mail: janusz.martyniak@imperial.ac.uk, Henry.Nebrensky@physics.org, durga@fnal.gov

Abstract. The international Muon Ionization Cooling Experiment (MICE) currently operating at the Rutherford Appleton Laboratory in the UK, is designed to demonstrate the principle of muon ionization cooling for application to a future Neutrino Factory or Muon Collider. We present the status of the framework for the movement and curation of both raw and reconstructed data. A raw data-mover has been designed to safely upload data files onto permanent tape storage as soon as they have been written out. The process has been automated, and checks have been built in to ensure the integrity of data at every stage of the transfer. The data processing framework has been recently redesigned in order to provide fast turnaround of reconstructed data for analysis. The automated reconstruction is performed on a dedicated machine in the MICE control room and any reprocessing is done at Tier-2 Grid sites. In conjunction with this redesign, a new reconstructed-data-mover has been designed and implemented. We also review the implementation of a robust database system that has been designed for MICE. The processing of data, whether raw or Monte Carlo, requires accurate knowledge of the experimental conditions. MICE has several complex elements ranging from beamline magnets to particle identification detectors to superconducting magnets. A Configuration Database, which contains information about the experimental conditions (magnet currents, absorber material, detector calibrations, etc.) at any given time has been developed to ensure accurate and reproducible simulation and reconstruction. A fully replicated, hot-standby database system has been implemented with a firewall-protected read-write master running in the control room, and a read-only slave running at a different location. The actual database is hidden from end users by a Web Service layer, which provides platform and programming language-independent access to the data.

1. Introduction

The International Muon Ionization Cooling Experiment (MICE) [1] located at the Rutherford Appleton Laboratory, UK, (RAL) is designed to demonstrate the principle of muon ionization cooling for the first time. In Section 2 we briefly describe the principle of ionization cooling and provide an overview of the layout of the experiment and its detectors. In Section 3 we describe the curation of the raw data recorded by the experiment (Section 3.1) as well as the curation of the data reconstructed by the MICE reconstruction software (Section 3.2). The goals of the data curation frameworks (both the raw and reconstructed data) are to ensure that data are safely and quickly moved to permanent storage and that the entire process is fully automated

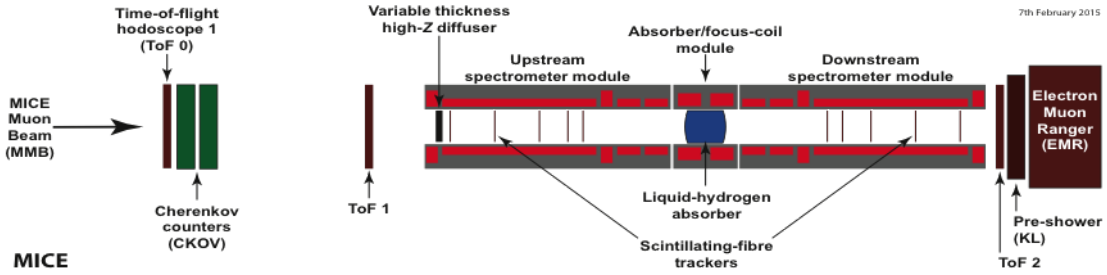

Figure 1. Current MICE configuration.

with sufficient checks to ensure the integrity of the data-transfer. Since MICE is an experiment that is designed to operate under a variety of configurations, it is essential to keep records of the conditions under which data are recorded and reconstructed. In Section 4 we describe the design and implementation of a database framework for this purpose.

2. The MICE Experiment

Ionization cooling is a technique which allows reducing the emittance of a beam of charged particles. The momentum of the particles is reduced by the mechanism of energy loss as they pass through an absorber material, and then subsequent reacceleration through rf cavities restores the lost longitudinal momentum thereby "cooling" the beam. The main component of MICE is a cooling channel which consists of a series of absorbers and high precision scintillating fibre trackers placed in strong 4T solenoidal fields created by a set of superconducting magnets. To discriminate between the muons of interest and the background from other particles, MICE is equipped with a set of particle identification detectors (time-of-flight detectors, Cherenkov counters, and calorimeters)[2]. Figure 1 shows the current setup of MICE with absorbers, particle identification detectors and high precision scintillating-fibre trackers [3].

3. Raw and reconstructed data handling

3.1. Raw data handling

The MICE data acquisition (DAQ) system, developed on the basis of the DATE [4] package from the ALICE experiment is responsible for recording data consistent with experimental triggers. Data are taken in "runs" of varying configurations and a run typically corresponds to 2 hours of operation. The DAQ writes raw data to disk in 250 MB chunks during each run. When the run has finished a File Compactor process collects the recorded chunks along with essential logs and outputs from the online monitoring and online reconstruction programs and collates them into a single tarball. The compactor also adds the list of files and their MD5 checksums to allow integrity checking later in the processing chain. The data-mover process actively monitors the creation of new tarballs and transfers data to long-term tape storage on the CASTOR [5] Storage Element (SE) at the GridPP Tier-1 facility at RAL [6] which maintains 2 tape copies for added security. The process is described in detail in [7]. The raw data movement task has 2 steps: 1) an initial temporary copy with self integrity check and 2) a final copy to the Grid. The first step is intended to avoid bottlenecks on the DAQ disk access as the DAQ writing must take priority during the run. At the end of each run the compacted tarball is placed in a directory actively watched by the data movement process and the initial copy is started only when the tarball is ready. This is indicated by a presence of a run-dependent semaphore. After the initial

copy is made and the file's integrity is verified, the CASTOR upload process is started. The CASTOR upload process makes a check-summed transfer, registers the file with the LCG File Catalog (LFC) and stores basic file attributes in the MICE Metadata Catalogue which stores essential attributes about the files. We have since upgraded the system for robustness and now use GFAL2 Python API [8] to transfer data and perform LFC registration. This has proved to be more robust than the binary LCG tools used in the original data mover.

3.2. Handling of reconstructed data

In our paper [7] we described a distributed Grid MICE data reconstruction system. It involved submitting a series of reconstruction jobs to the Grid and copying data back to the RAL CASTOR storage. The reconstruction process was triggered by raw data uploaded to CASTOR and a relevant run record present in the MICE Metadata Catalogue. This method was prone to delays and unnecessary bottlenecks associated with the processing and submission chain. A lot of effort was made to improve the reconstruction speed of the software which has allowed us to perform offline reconstruction on a dedicated machine in the control room at a speed consistent with the data-taking rate. The reconstruction mover (Figure 2) was then accordingly modified and uses a single stage data movement process, similar to the raw data handling task. The reconstructed data transfer task is now triggered by the same method as the raw data transfer - namely, a semaphore indicating that the reconstructed tarball is ready to be moved. This made the raw and reconstruction movement tasks independent of each other. The Grid-based distributed reconstruction system is still available and is planned to be used for large scale reprocessing.

4. MICE Configuration Database

Since MICE takes data under a variety of configurations - different magnetic fields, triggers, absorber materials, etc - information about experimental conditions is vital for accurate simulation and reconstruction. MICE uses a Configuration Database (CDB) - a fully replicated, hot-standby database system running on a master-slave PostgreSQL DBMS [9] (Figure 3). Access to the database is provided by a Java EE JAX-WS Web Service (WS) [10] layer deployed on Apache Tomcat. We have set up a firewall protected read-write master node located in the experiment control room and a public read-only server hosted within the RAL PPD Tier-2. A JAX-WS Web Service is a SOAP [11] service. SOAP is a lightweight XML-based messaging protocol which allows platform and programming language independent access to data.

4.1. CDB API

We provide following CDB APIs:

- *C*, to allow the MICE Run Control system to read various parameters from the hardware and store them in the CDB. Run Control is based on EPICS [12], which is a toolkit with C APIs, so it was natural to design a WS client interface in the same programming language. We provide a selection of read-write operations used on a run-by-run basis to define magnet currents, absorber and target settings, beam parameters, and any other information that would have to be read from or written to the CDB. We also supply functions to define run condition templates (tags) to be read and written to the CDB. We use a gSOAP [13] based client API to contact the CDB.
- *C++*, to be natively used by the reconstruction software, especially in areas where speed is critical. We have designed a selection of read-only operations to access detector calibrations, electronics and detector channel maps. We utilise gSOAP C++ bindings to access the CDB.
- *Python*, to store certain predefined conditions such as detector geometry definitions, detector calibration, cabling and other non run-based information. This is the most complete set

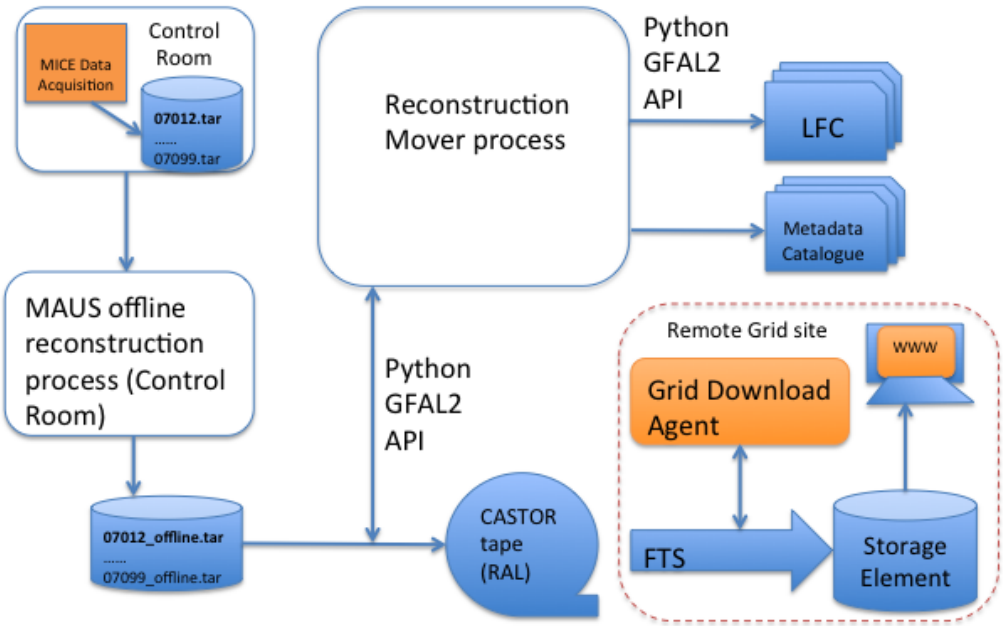

Figure 2. Fully automated system for MICE data reconstruction and archiving. Also shown is an optional, custom built, FTS [14] based Grid Download Agent to facilitate data distribution to other Grid sites.

of operations widely used both by experts and end-users for their analyses. The API is regularly released and is bundled with the reconstruction software distribution but can also be installed stand-alone. It uses Python suds [15] module which is the only external dependency.

- *Java*, used by a Web application to view the CDB, based on the Google Web Toolkit (GWT) [16]. Only a slave database can be contacted this way.

4.2. CDB replication and recovery

Since the CDB is actively used during data taking it is very important to ensure that the master server is available at all times. Running the PostgreSQL DBMS in a hot-standby mode allows a slave server to be promoted to a master in the event of the original master being unusable. This is a built-in feature of PostgreSQL. We use the following procedure at MICE:

- Reconfigure (promote) a slave server to become a new master,
- Start a firewalled read-write WS, so the control room can now write to the new master. We still maintain the original read-only slave WS as before, so the read access is not affected,
- Synchronise remaining slaves, if applicable. We have an operational master-slave system at this point,

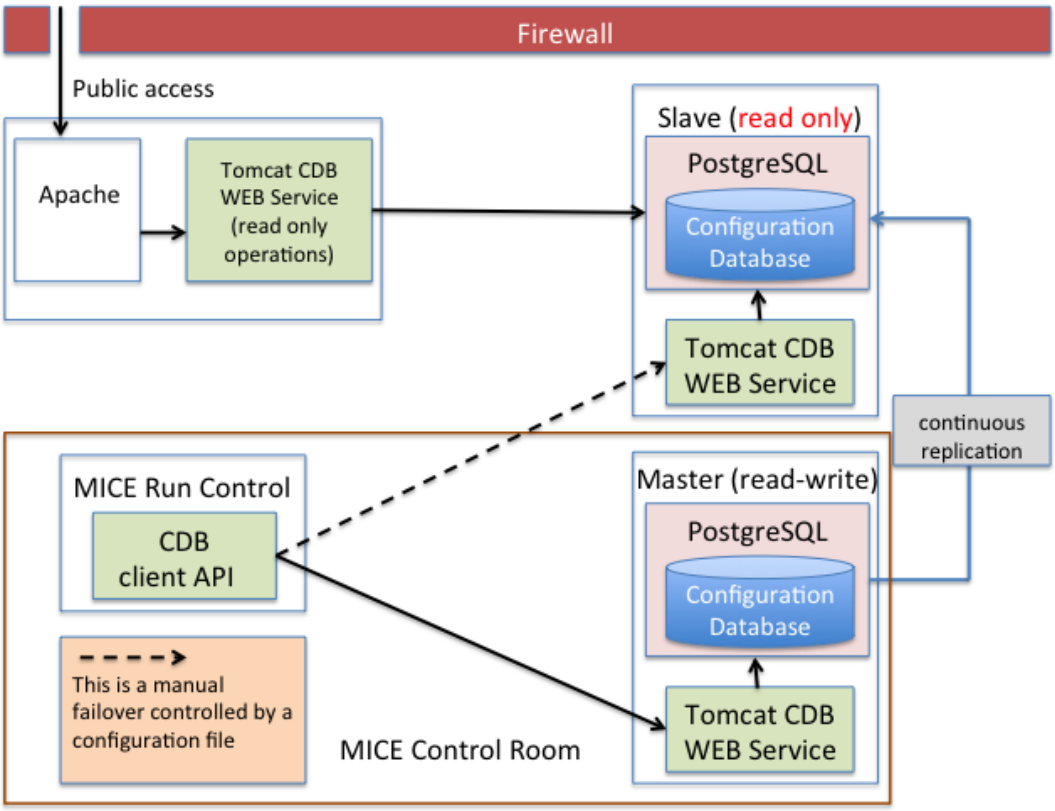

Figure 3. MICE Configuration Database system.

- When the old master is available again, swap the servers back and stop read-write access started above.

In addition to running a master slave(s) system in hot-standby mode we perform regular CDB back-ups for safety.

5. Conclusions

We have presented updated, automated versions of the raw and reconstructed data transfer tasks for MICE which allow files to be reliably and efficiently uploaded to permanent tape storage. Both the raw and reconstruction movers have been performing reliably during data-taking. We have also presented a robust, fully replicated MICE Configuration Database holding vital information about experiment settings. An extensive set of APIs have been developed supporting C, C++, and Python clients thus enabling end-users to easily interact with the database.

References

[1] Bogomilov M *et al.* "Design and expected performance of the MICE demonstration of ionization cooling" arXiv:1701.06403
Bogomilov M *et al.* 2012 "The MICE Muon Beam on ISIS and the beam-line instrumentation of the Muon Ionization Cooling Experiment" *Journal of Instrumentation* **7** P05009

[2] Bertoni R *et al.* 2010 "The design and commissioning of the MICE upstream time-of-flight system" *Nucl. Instrum. Methods* A **615**(1) pp 14-26
[3] Ellis M *et al.* 2011 "The design, construction and performance of the MICE scintillating fibre trackers" *Nucl. Instrum. Methods* A **659**(1) pp 136-53
[4] Carena F *et al.* 2006 "The ALICE Data-Acquisition Software Framework DATE" *Proc. 15th Intl. Conf. Computing in High Energy and Nuclear Physics* pp 25-28
[5] The CERN Advanced Storage manager (CASTOR) `http://castor.web.cern.ch`
[6] Britton D *et al.* 2009 "GridPP: the UK Grid for particle physics " *Phil. Trans. R. Soc.* A **367** pp 2447-57
[7] Martyniak J 2014 "MICE data handling on the Grid" *J. Phys.: Conf. Series* **513** 032063
[8] Data Management Clients GFAL2-Python `https://dmc.web.cern.ch/projects/gfal2-python`
[9] PostgreSQL `https://www.postgresql.org/`
[10] Java API for XML Web Services (JAX-WS) `http://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html`
[11] SOAP: Simple Object Access Protocol `https://www.w3.org/TR/2000/NOTE-SOAP-20000508`
[12] EPICS: Experimental Physics and Industrial Control System `www.aps.anl./epics/index.php`
[13] van Engelen R and Gallivan K 2002 "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks" *Proc. 2nd IEEE Int. Symp. on Cluster Computing and the Grid* (Berlin, Germany) pp 128-135
[14] FTS: File Transfer Service `http://fts3-service.web.cern.ch`
[15] suds: Python SOAP Web Services client library `https://sourceforge.net/projects/python-suds/`
[16] GWT: Google Web Toolkit `www.gwtproject.org`