Journal of Physics: Conference Series **219** (2010) 022004 doi:10.1088/1742-6596/219/2/022004

Online processing in the ALICE DAQ
The Detector Algorithms

S Chapeland[1], V Altini[2], F Carena[1], W Carena[1], V Chibante Barroso[1], F Costa[1], R Divià[1], U Fuchs[1], I Makhlyueva[1,3], F Roukoutakis[1,4] , K Schossmaier[1], C Soós[1], P Vande Vyvre[1] and B von Haller[1], for the ALICE collaboration

[1] CERN, Physics Department, CH-1211 Geneva 23, Switzerland
[2] INFN, Diparimento di Fisica dell'Università and Sezione INFN Bari, Italy

E-mail: sylvain.chapeland@cern.ch

Abstract. ALICE (A Large Ion Collider Experiment) is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN Large Hadron Collider (LHC). Some specific calibration tasks are performed regularly for each of the 18 ALICE sub-detectors in order to achieve most accurate physics measurements. These procedures involve events analysis in a wide range of experimental conditions, implicating various trigger types, data throughputs, electronics settings, and algorithms, both during short sub-detector standalone runs and long global physics runs. A framework was designed to collect statistics and compute some of the calibration parameters directly online, using resources of the Data Acquisition System (DAQ), and benefiting from its inherent parallel architecture to process events. This system has been used at the experimental area for one year, and includes more than 30 calibration routines in production. This paper describes the framework architecture and the synchronization mechanisms involved at the level of the Experiment Control System (ECS) of ALICE. The software libraries interfacing detector algorithms (DA) to the online data flow, configuration database, experiment logbook, and offline system are reviewed. The test protocols followed to integrate and validate each sub-detector component are also discussed, including the automatic build system and validation procedures used to ensure a smooth deployment. The offline post-processing and archiving of the DA results is covered in a separate paper.

1. Introduction

1.1. The ALICE experiment

ALICE (A Large Ion Collider Experiment) [1][2][3][4], is the heavy-ion detector designed to study the physics of strongly interacting matter and the quark-gluon plasma at the CERN Large Hadron Collider (LHC). As depicted in Figure 1, the detector includes high resolution tracking (silicon detectors, large time-projection chamber), particle identification, and triggering elements. It features two large magnets, a main solenoid and a dipole on the Muon arm. ALICE consists of 18 sub-detectors, being able to take data independently (standalone operation) or in global partitions (set of sub-detectors

[3] Now at ITEP Institute for Theoretical and Experimental Physics, Moscow, Russia
[4] Now at University of Athens, Physics Department, Athens, Greece

running together). All 18 sub-detectors can run standalone in parallel, and up to 6 global partitions can run in parallel.

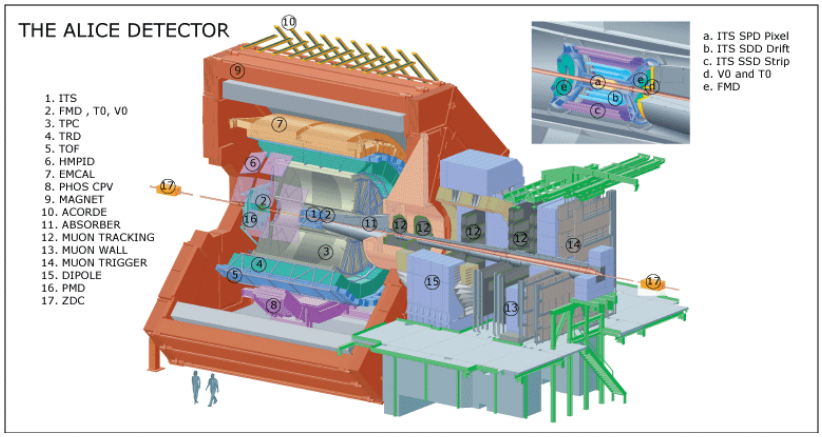


Figure 1 - ALICE experiment layout

1.2. Calibration procedures

The 18 sub-detectors require specific calibration tasks to be performed regularly in order to achieve the most accurate physics measurements. These systems are indeed sensitive to configuration settings, mechanical geometry, environmental conditions changes, components aging and sensors defects. The corresponding set of procedures to calibrate the sub-detectors involves events analysis in a wide range of experimental conditions. These calibration tasks may be done either in dedicated runs, or in parallel to physics data taking. Typical examples of calibrations include pedestal and gain computation, dead and noisy channels mapping, etc. Depending on the sub-detector and the calibration task, one has to define in particular:

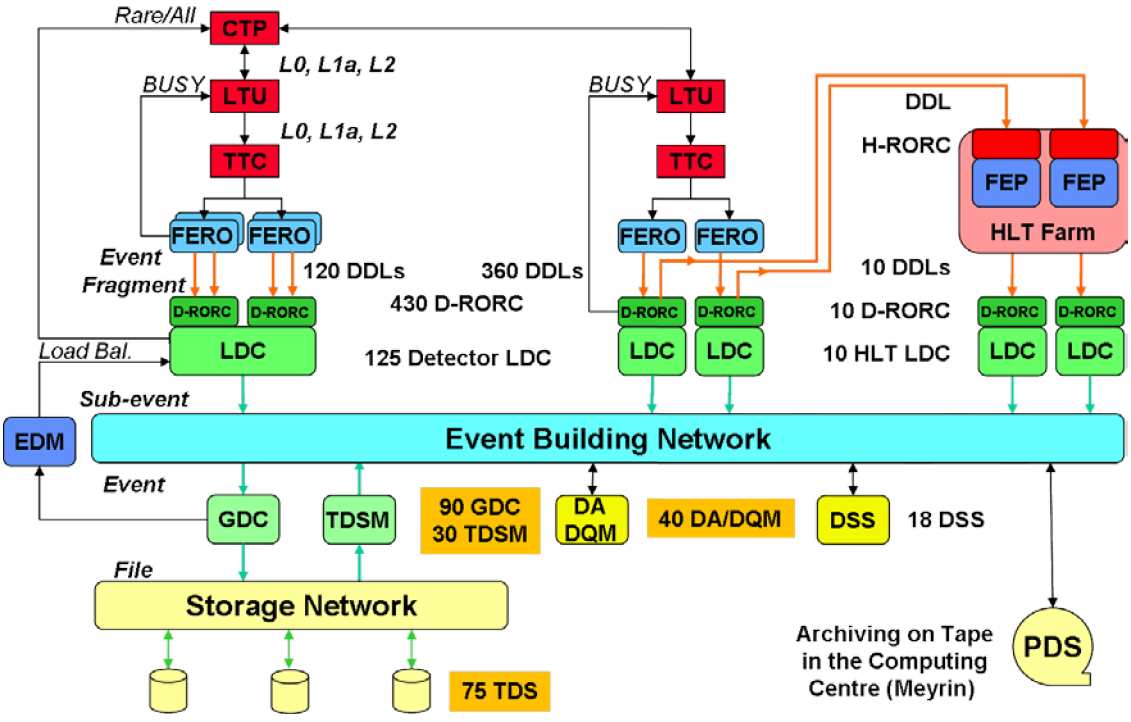
- The trigger type, which can be a normal physics trigger, or some specific events related to dedicated hardware device (e.g. laser, LED, pulser).
- The number of events to collect, from a hundred events for pedestal runs to millions of events for dead channel mapping.
- The event formatting, zero-suppressed or not, which impacts on the required throughput. Event size ranges from sub-events of few kilobytes to 20 MB.
- The detector electronics settings, specific to the sub-detector operation mode.
- The calibration algorithm, i.e. the actual code to interpret the data and produce results.
- The type of run, standalone or global, depending if the task can be performed during normal data taking or requires a specific run. It has an impact on the operation mode and detector dead-time for physics.
- The frequency at which the calibration is required, from few times per day to once a year.

The calibration results produced may be needed to configure the detector electronics for data taking, for example to produce zero-suppressed data or to mask noisy channels, in order to reduce the data volume. Therefore, these results should be available right after the calibration data-taking procedure, in order to reconfigure the detector accordingly for the next physics run. In addition, the results are also used offline for the events reconstruction. Both usages of the results involve a drastic timing constraint on the way they are produced. It would be too heavy to make the full calibration analysis offline (a first pass over the data would be needed to produce calibration results), and sometimes too late (for calibrations required very frequently, or for which results are needed for the detector configuration). Only the most complex calibration data analysis should be done offline.

Therefore, a dedicated framework has been designed and implemented to achieve as much as possible the detector calibration directly online, and to address the heterogeneous requirements specific to each calibration task.

1.3. The ALICE DAQ

The ALICE Data-Acquisition system (DAQ) [5][6] handles the data flow from the sub-detector electronics to the archiving on tape, as seen on Figure 2. A first layer of computers, the Local Data Concentrators (LDCs), reads out the event fragments from the optical Detector Data Links (DDLs). Up to 12 DDLs can be connected to the same LDC, and several LDCs may be needed to collect the data from a single sub-detector. The event fragments are then transferred to a second layer of computers, the Global Data Collectors (GDCs), in charge of performing the event building. The same GDC receives all the fragments of a given event, and assembles them in a full event, which is then recorded to a transient storage (TDS) before being migrated to tape (PDS). Each uninterrupted data taking period is called a run, ranging from few minutes to many hours, with the same hardware and software configuration. There can be several runs in parallel, with different sub-detectors running together or standalone.



CTP: Central Trigger Processor
DA: Detector Algorithm
DDL: Detector Data Link
DQM: Data Quality Monitoring
D-RORC: Detector Read Out Receiver Card
DSS: DAQ Service Server
EDM: Event Distribution Manager
FERO: Front End Read Out electronics
FEP: Front End Processor
GDC: Global Data Collector
HLT: High-Level Trigger
H-RORC: HLT Read Out Receiver Card
LDC: Local Data Concentrator
LTU: Local Trigger Unit
PDS: Permanent Data Storage
TDSM: Transient Data Storage Manager
TTC: Timing, Trigger and Control

Figure 2 - The ALICE DAQ architecture

This architecture provides several features relevant for the implementation of an online calibration framework:

- The distributed architecture provides inherent parallelism possibilities to process the event fragments data on many nodes.
- The DDLs are full duplex links to the sub-detectors front-end electronics (FEE). This link can be used to upload settings directly to the detector after calibration results have been computed.
- The data-acquisition software, DATE, includes a monitoring API, allowing grabbing data and retrieving copies of the events on-the-fly without disrupting the main data flow. DATE also has a configuration setting to record data locally on the machines.
- The control software allows the operation in standalone and global runs independently for each sub-detector.

In addition, the Experiment Control System (ECS) is the state-machine software in charge of synchronizing and controlling all the online components: the DAQ, the Detector Control System (DCS), the Central Trigger Processor (CTP) and the High-Level Trigger (HLT). Having such central control point is crucial to define the various calibration tasks and their sequence of operation.

We describe in the next section how these characteristics are used to implement the ALICE online calibration framework.

2. The ALICE online calibration framework

2.1. The detector algorithms

The ALICE online calibration framework is used to implement and run a set of detector algorithms (DAs), which are calibration tasks running online. DAs are provided by the sub-detector teams, using the global framework to develop detector-specific calibration procedures.

Each DA grabs detector data and produces results online. These results can be reused directly online, e.g. to configure the detector, or shipped offline to be post-processed (if necessary) and used in event reconstruction.

To cover all the needs, we have defined two types of DAs, running either in exclusive mode (a dedicated run is required), or in the background (the task can be performed in a physics run):

- In the first case, called ‘LDC DA’, the data is recorded locally and in parallel on the LDCs, during a dedicated standalone run (single detector running), usually of short duration. At end of run, a DA process is launched on each LDC to analyze the data. In this mode, parallelization is optimal, and results are readily available for further export to FEE. Typical example is the pedestal run, with few hundreds of big events. The temporary data files stored on local disk are useful to re-play, tune or debug the DA behavior.
- In the second case, called ‘MON DA’, a single DA process of a given type is active during the run, on a dedicated monitoring machine. Data samples are picked up from the normal data flow in a non-intrusive way, and processed directly on the fly. The DA gets only what it can process, events may be dropped in case the DA is busy. The DA selects the type of events it needs (calibration, physics) and the source to monitor (typically, a detector or a set of detectors). At the end of run, the DA goes in a post-processing phase to finalize the results. A generic example is to populate an histogram event by event, and at the end of run compute a fit and extract some key values. Another example of MON DA usage is for the dead channel mapping, where millions of events may be needed to cover the full detector. Many runs may be needed to collect such statistics, in which case intermediate results are saved at end of run, and re-loaded at the next start of run.

2.2. DA framework architecture

The overall DA framework architecture, and in particular the interaction of the DA processes with the online components, can be seen in Figure 3. The DA process consists of detector code, written in C++ using the AliROOT framework [7], which is the ALICE offline code repository (therefore providing

the same calibration algorithm implementation for both online and offline environments). This code uses the DAQ DA interface library in order to communicate with the other systems. The ECS is in charge of launching the DA where needed for the corresponding run type, depending on the experiment running mode selected by the operator. The run type is propagated to the other online systems and to the detector, in order to make sure corresponding settings are applied. This information is also stored in the experiment logbook [8] for bookkeeping and further reference.

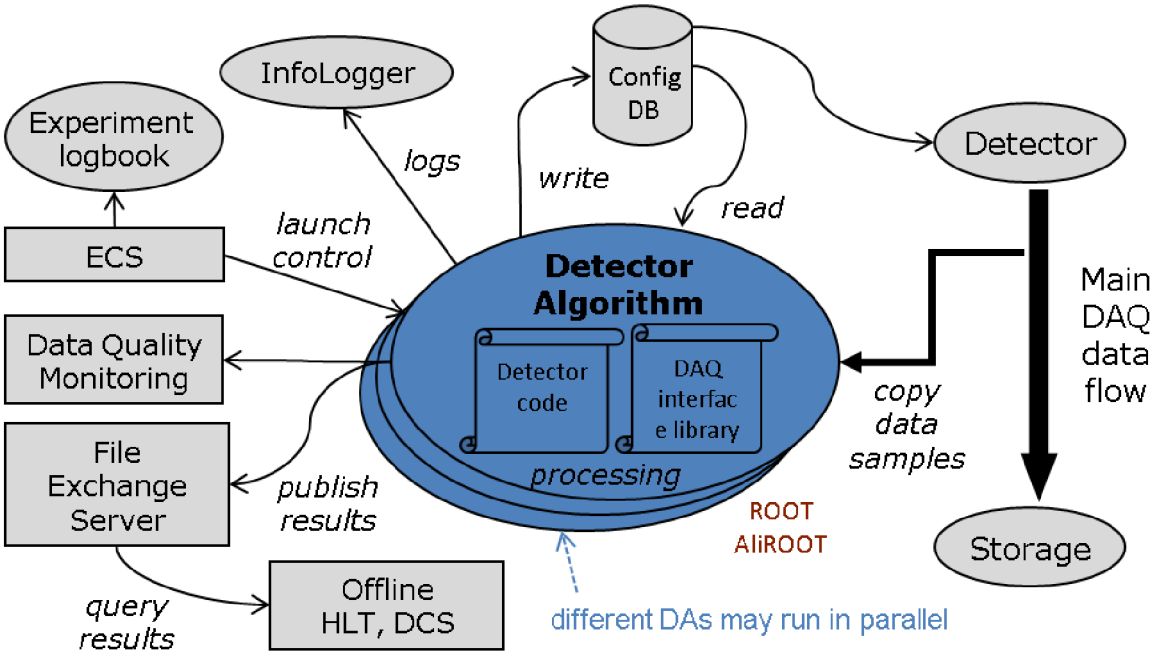


Figure 3 - The DA framework architecture

Upon startup, the DA connects to the main DAQ data flow (being local files for LDC DA, or remote data sources for MON DA) in order to collect events.

The DA may use some configuration information stored centrally in a database, which allows the operator to define the DA operation settings or algorithm parameters.

At run time, all output messages from the DA process are exported to the central DAQ/ECS log system for online operator display and archival. Health of the process is also monitored constantly by the ECS, and return error code checked upon exit.

While running, the DA may publish its results to the experiment Data Quality Monitoring (DQM) framework [9] for feedback to the operator, graphical display, consistency and quality checks, or reuse for monitored data reduction in the DQM agents.

The DA is notified the end of the run, and then proceeds to final post-processing and results saving. The control system checks that the DA completes its tasks within the required time, and aborts the process if necessary. Allowed DA end-of-run duration is kept short for the global runs (typically less than one minute), to minimize data taking dead-time. Whenever possible, demanding computation tasks exceeding this threshold are performed offline where there is no such constraint.

The DA can save persistent files to a configuration database (local or central) in case results are to be reused in the DAQ (e.g. for a further DA run, for other online processes, for FEE configuration, etc). Calibration results are also exported to the File Exchange Server, which is the system used to publish data from DAQ to the other components. The DA result files may as well be reused in HLTor DCS. Most importantly, the results are collected offline by the Shuttle framework [10], where the output data is post-processed and archived in the Offline Condition Database.

2.3. Online databases implementation

The configuration database, used to store DA parameters (input) and selected results (output), is implemented using MySQL. All data files are stored directly in a MySQL table given their relatively small size (few hundred megabytes in total for the experiment). Command line tools, C/C++ interface, and Tcl/Tk file explorer have been implemented for the database content access. Selective permission to a subset of the database is given to users depending on the sub-detector they belong to.

A local storage also exists on the machines where the DA runs. It consists of a working directory storing temporary files, and a persistent directory, used to store files saved for a later local usage. This minimizes unnecessary remote file access and improves performance.

The File Exchange Server (FXS) is a transient storage where DAs write their output files. They are deleted once read by target consumer (e.g. the Shuttle). In practice, the files are not deleted straight after being transferred, but rather archived in a FIFO manner to keep an history as far as storage allows. The FXS consists of a MySQL table for the file indexing and bookkeeping, and of a locally attached RAID disk array for storage, as the data files are usually too large to be efficiently stored directly in MySQL tables. File content is accessed by SCP and SFTP. Interactive login is not allowed, the RSSH restricted shell ensures that only SCP/SFTP can be used by the FXS users in order to prevent the machine to be used for other means. This is particularly important as the node is visible on the general purpose network (GPN), whereas all other DAQ components work on an isolated network. It avoids machine misuse (e.g. unwanted gateway to DAQ network) and machine resources are kept stable (CPU used only for file transfert). A dedicated port forwarding mechanism has been setup to access the database index from the GPN. Ad-hoc Linux IPtables configuration on a gateway machine routes IP traffic on the MySQL port from GPN to the FXS database, hosted on a DAQ machine hidden otherwise.

3. Deployment

The DA framework has been in production since December 2007, and the integration of numerous DAs took place for the 2008 ALICE cosmic runs. At the moment, more than 30 DAs are used to calibrate the detector. Each DA is assigned to a dedicated CPU core in the pool of DAQ machines to guarantee computing resources for each task. In total, more than 11000 runs took place with DAs running, producing more than 33000 files, for a total of 80 GB of calibration data exported.

3.1. Procedures

In order to ease the interactions with the 18 detector teams and optimize the code release time, a single deployment procedure has been defined. First, the appropriate run types are discussed with the detector team and control operations implemented in the ECS. The detector teams then provide the DAs, as code delivered in the AliRoot framework. Each DA is then validated on a test platform, on the same hardware and environment as used at runtime. An automated tool checks out the code of the corresponding version, builds the executable, launches a set of unit tests to check the DA interfaces and documentation compliance with the framework, and generates a report. The DA testing is done by providing as input a reference data file, either real data taken at the experimental area in the corresponding conditions, or issued from a simulation process. Using realistic runtime environment and source data is crucial in this test phase to evaluate the behaviour and stability of the DA. Many of the problems can be spotted and corrected before deployment, therefore saving a lot of time. It also provides an excellent feedback on the DA performance, so that computing resources required can be evaluated, matched with existing hardware, and if necessary code optimized. Once validated, the DA is deployed at the experimental area, and the calibration procedure tested in real life before being put in production.

Concerning the code packaging, all DAs are retrieved and built in a standard way from AliROOT using a common makefile target. The DAs are compiled statically to avoid runtime dependencies. This was needed so that various software versions can cohabit together in the runtime environment. This constraint was especially strong during the phase of experiment development and commissioning, with many code releases and heterogeneous software. This requirement is although not so important in a production runtime environment with more stable packages and versions. The executables are distributed by RPM packages, which also include self-documenting tags, such as: contact person, link to documentation, reference run numbers, DA type, number of events needed, input and output files, trigger type, run types, etc. This minimal information is vital for the long term maintenance and operation of the system.

3.2. Performance considerations

The calibration tasks are often CPU demanding. Better DA computing performance allows more statistics to be collected and analysed in a shorter time. It reduces the time needed for a calibration, improves calibration quality and shortens physics data-taking dead time. Therefore, it is crucial that DAs executables are optimized as much as possible.

The first mean is to tune the algorithm itself, which can be done only at the level of the detector code. This approach is therefore outside of the reach of the global framework. However, it is still possible to provide feedback to the code author, with performance evaluation and non-regression tests on reference data sets. In particular, one can check that the time needed to process the number of events required for calibration is acceptable. This is done during the automatic validation procedure.

The second area of code optimization deals with the choice of a compiler, and this can be done globally at the level of the framework. Our investigations showed that, in most cases, the DA code produced with Intel ICC was 10-20% faster than with GCC. Given the good compatibility and straight forward compilation of existing code, and the central build-server architecture in this project, a major improvement can therefore be achieved at a minimal setup cost. It may imply however some runtime debugging issues. Now that DA versions have stabilized, we are investigating this possibility for production.

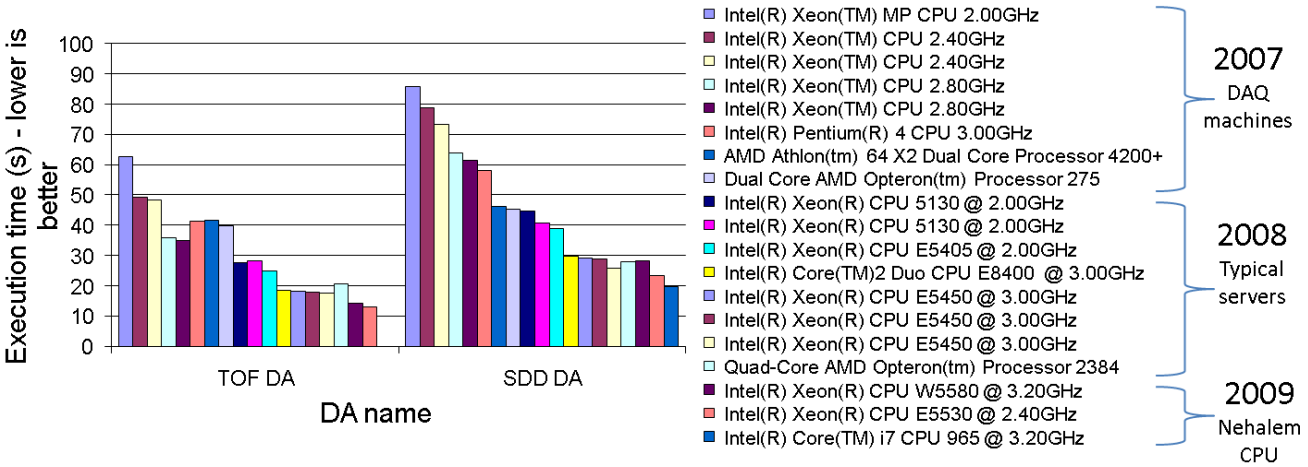


Figure 4 - DA benchmark with different CPUs

Finally, the last degree of freedom on DA performance is the hardware on which DAs run. According to our benchmarks, the latest CPU generation (Intel Nehalem) provides a big performance gain compared to existing hardware installed at the experimental area in 2007, as can be seen from a comparison of several platforms for the execution time of two DAs in Figure 4. We are in the process

of migrating to this type of hardware. The figure shows processing time using a single core of a given CPU type, as DAs are not multithreaded at the moment.

Despite no performance issue has been met yet, we are also investigating the possibility to have a way to distribute events across platforms to improve parallelism, i.e. having several instances of the same MON DA running on different nodes, each receiving different events from the same set of detectors. This would be a major step forward in scalability, but this approach is not compatible with all algorithms, as independent event processing and global combination of partial results are not always feasible.

4. Conclusion

A framework for the implementation of calibration procedures in the ALICE online environment has been designed and put in production with success for the experiment commissioning in 2008. The work required to set up a common interface was worth the effort, as all calibration tasks requested so far fit in the architecture, providing a smooth and efficient single release process. The system is mature, fine performance tuning is being addressed, and operation is ready to scale up with the higher calibration demand arising with the LHC start-up.

References

[1] ALICE Collaboration 1995 Technical Proposal, *CERN-LHCC-1995-71.*
[2] ALICE Collaboration 2008 The ALICE experiment at the CERN LHC, *JINST* **3** S08002
[3] ALICE Collaboration 2004 ALICE: Physics Performance Report Vol I, *J. of Phys. G: Nucl. Part. Phys* **30**, 1517-1763
[4] ALICE Collaboration 2006 ALICE: Physics Performance Report Vol II, *J. of Phys. G: Nucl. Part. Phys* **32**, 1295-2040
[5] ALICE Collaboration 2004 The Technical Design Report of the Trigger, Data-Acquisition, High Level Trigger, and Control System, *CERN-LHCC-2003-062*
[6] Altini V, Carena F, Carena W, Chapeland S, Chibante Barroso V, Costa F, Divià R, Fuchs U, Makhlyueva I, Roukoutakis F, Schossmaier K, Soós C, Vande Vyvre P and von Haller B 2009 Commissioning the ALICE Experiment, *Proc. CHEP 2009* (Prague, CZ: JPCS)
[7] AliROOT web site, http://aliceinfo.cern.ch/Offline/AliRoot/Manual.html
[8] Altini V, Carena F, Carena W, Chapeland S, Chibante Barroso V, Costa F, Divià R, Fuchs U, Makhlyueva I, Roukoutakis F, Schossmaier K, Soós C, Vande Vyvre P and von Haller B 2009 ALICE Electronic Logbook, *Proc. CHEP 2009* (Prague, CZ: JPCS)
[9] Altini V, Carena F, Carena W, Chapeland S, Chibante Barroso V, Costa F, Divià R, Fuchs U, Makhlyueva I, Roukoutakis F, Schossmaier K, Soós C, Vande Vyvre P and von Haller B 2009 The ALICE Online Monitoring, *Proc. CHEP 2009* (Prague, CZ: JPCS)
[10] Zampolli C, Grosse-Oetringhaus JF et al. 2009 The ALICE Online-Offline Framework for the Extraction of Conditions Data, *Proc. CHEP 2009* (Prague, CZ: JPCS)