

ALICE O²

System Requirements Specifications

alice-o2-cwg1@cern.ch

v1.0 - 24/03/2014

1 Introduction

1.1 Purpose

This document describes the ALICE O² system to be designed for the upgrade of the ALICE experiment during the LHC Long Shutdown 2. It also serves as base of the Technical Design Report, to be submitted in Autumn 2014.

This document is targeted at the members of the ALICE collaboration, in particular those participating in the O² Computing Working Groups.

1.2 Scope

The ALICE experiment will be extensively upgraded during the Long Shutdown 2 (LS2) of the LHC, currently scheduled for 2018. The physics goals of the ALICE upgrade program, the expected increased LHC performance and the chosen detector technologies for the upgrade require fundamental changes in the ALICE detector readout architecture as well as in the ALICE data acquisition, trigger and computing models. These changes imply a tighter integration of these functions, including an increased sharing of software and hardware components.

1.3 Definitions, Acronyms, and Abbreviations

Terminology considerations are handled in a separate document:

<https://twiki.cern.ch/twiki/bin/viewauth/ALICE/Terminology>

1.4 References

References for this document can be found in Appendix A.

1.5 Overview

This document follows the usual Software Requirements Specifications structure as recommended in [3], but is not limited to software. We rather deal with *system* requirements, as the data flow will be implemented by closely coupled hardware and software components. Section 1 provides introductory information and references used in this document. Section 2 presents the ALICE online-offline system. Section 3 details specific requirements of the different

modules.

1.6 History

This section will keep track of the released document versions.

Version	Date	Comment
0.1	14/05/2013	Initial Document Structure
0.2	17/06/2013	Document Version for Discussion at O2 Plenary June Mini-Week
0.3	26/08/2013	Draft to be presented to the collaboration
0.4	08/10/2013	Initial version to be circulated
1.0	24/03/2014	Inclusion of the comments received. New version tagged for reference, before continuing work on TDR.

Note: further updates on the requirements (numbers or architecture changes) will be reflected directly in the corresponding ALICE O² TDR sections. A new revision of the SRS document might be done after the TDR is finalized.

2 Overall description

2.1 System perspective

The O² system will perform the data acquisition and processing of the upgraded ALICE experiment in 2018. It collects the data from the detectors, transports and reduces the data volume before recording, and provides the calibration, the reconstruction, the quality assurance and analysis facilities. It also provides the necessary software infrastructure and code.

A preliminary design study was done by a panel of experts of the DAQ, HLT and Offline systems, who published a report about ALICE Computing software framework for LS2 Upgrade [2]. From this report, we can see in Figure 1 “the outline of the possible future data flow in the ALICE computing systems. The arrangement of Data Sample into Time Frames can be done on the detector level or later in the Local Processing. The Local Processing layer carries out the necessary data compression to ship the data to the next level within the available bandwidth. All other processing is preferably done at the Global Processing stage. Tagging might be used in all levels whenever relevant information for the separation of interactions can be extracted. The handling of calibration data is of importance for the data flow, its generation is probably an iterative process. The exchange and archival is managed by an appropriate database”.

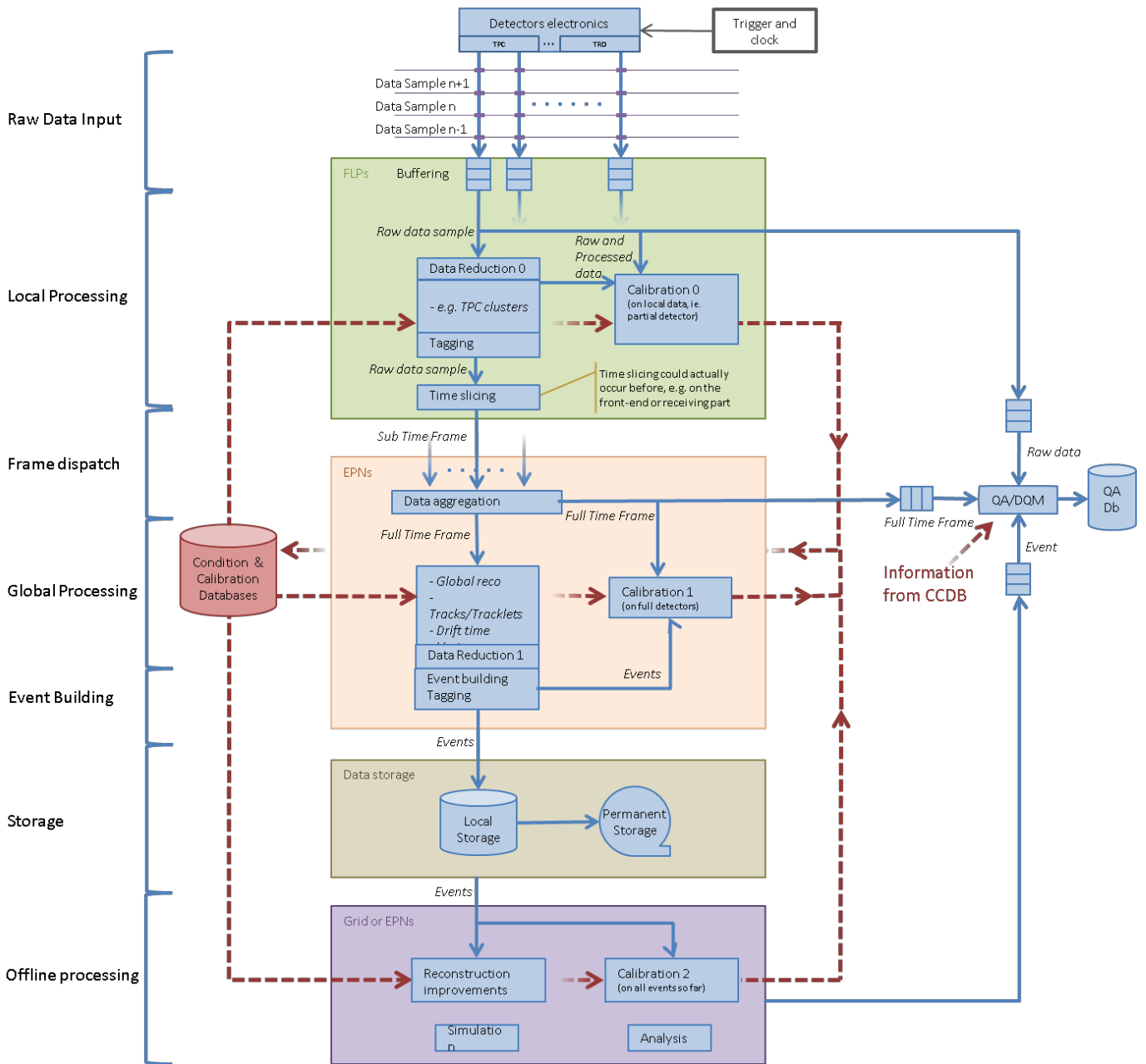


Figure 1 - "Outline of the possible future data flow in the ALICE computing systems.[2].

2.2 Interfaces

The O² system interacts with the following external components:

- the ALICE detector
- the permanent data storage system on the WLCG grid
- the GRID
- the LHC (e.g. clock, DIP)
- ALICE Central Trigger Processor (CTP)
- ALICE Detector Control System (DCS)

In addition to the interfaces to the external systems listed above, the following O2 interfaces have been grouped by category:

- System interfaces
- User interfaces
 - Run control
 - DQM
 - Logbook
 - Event display
- Hardware interfaces
- Software interfaces
- Communication interfaces

2.3 System functions

Based on the design considerations and the dataflow outlined above, the whole system can be split into major functional blocks. They can be roughly divided into two groups: those dealing with specific tasks at a specific point in the dataflow, and those interfacing with some or all of the functional blocks from the first group.

In the following sections, each of the functional blocks will be described in detail:

1. Data Input
2. Data Reduction
3. Storage
4. Reconstruction
5. Simulation
6. Analysis

7. Networking – Data Transport
8. Calibration
9. Condition and Calibration Database
10. Data Monitoring (QA/DQM/etc)
11. Software Framework

2.3.1 Data Input

CWG1 contacts: Sylvain, Tivadar

Concerned CWGs: 3

The data will be transmitted from the detectors electronics to the ALICE online system through data links. Different generations of Detector Data Links (DDL) may coexist, because the legacy electronics of some detectors may not be upgraded (DDL1, DDL2). The detectors readout which will be upgraded should use the same unified data link (DDL3, new design). The currently expected detector connectivity (type and number of links) is shown in the table below (for a 10Gb DDL3).

Detector	DDL1	DDL2	DDL3
ACO	1		
EMCal		20	
FIT		2	
HMP	14		
ITS			60
MCH			250
MID			1
PHOS		16	
TPC			1836
TOF		72	
TRD			36
ZDC			1

Table 1: Possible DDL usage by the ALICE detectors (as shown in Electronics TDR [7] page 20 Table 2.2)

These data links should be able to cope with the data rate of each detector. Each detector may split its data over several links if necessary, to accommodate their geometry and readout system (typically, detectors are split into modules). There could be many links, e.g. for large data providers like TPC. Outside of data taking periods (i.e. when detector data is not already flowing through the link) they should also allow the configuration of the electronics, i.e. upload data to the

detector or send control command. During the data taking period, the links may also be used to monitor the readout electronics (e.g. get temperature and voltage measurements). This traffic might be mixed with the physics data.

The buffering on the detector side is unspecified. Some detectors may buffer data, whereas others may not have an onboard event buffer in the electronics. The detector link should be able to cope with both strategies, and allow adequate transfer performance. As some detectors will be read out continuously, there may not be a backpressure mechanism. The link will not have to handle trigger or busy. This is part of the Common Readout Unit (CRU).

Resilience is a key aspect of the data input system. The links should be able to ensure the data integrity. The transfer should be safe and robust (radiation tolerant on the sending side, depending on where it is).

At the global level, the proper balance should be found between many slower links and fewer high-speed links. In the first case (many 'slow' links), resilience is maximized: in the event of a link failing, less data is lost. In the second case (fewer 'high-speed' links), the price is likely to be reduced (less optical links, less receiver cards), but the radiation tolerant implementation is more difficult and the system resilience is lower (too much data lost in case the link goes down, preventing reconstruction). It should also be ensured that the readout computers are able to cope (data taking and processing) with the full speed links data throughput (otherwise link bandwidth is unnecessary), and at the same time that the number of readout computers is reasonable (too many links means too many computers as well).

A fair guess would be that the data link speed will be between 10 and 100 gigabits per second.

2.3.2 Data Reduction

CWG1 contacts: *Thorsten*

Concerned CWGs: *3, 4, Reconstruction (8?), 12*

After the data has been collected from the ALICE detectors, a first processing step follows on the O² computing facility. Its goal is to reduce the data sent by the detectors to a size which can be handled by the data storage systems. The experience of Run 1 has shown that such a reduction can not be reached by applying lossless-compression techniques but rather requires a partial online reconstruction for some detectors.

Data reduction can be performed at various stages of the data aggregation process. The stages are naturally defined by the locality of the data: the first processing possibility on the Read Out Receiver Cards (RORC) has access to the data of one readout fibre/link. The next processing step is on the nodes (First Level Processors; FLPs) hosting the receiver cards, where all detector data received by the attached receiver cards is available. The last possible processing

step before transferring the data to the storage system is on the Event Processing Nodes (EPNs), where the full event data is available.

2.3.3 Storage

CWG1 contacts: Latchezar

Concerned CWGs: 4, 6, 12

Storage systems will be separated by design in two distinct groups – the O² system internal storage and the Grid disk and tape external storage. Depending on the final design of the conditions data store, a third type, tailored specifically for large databases support, may be needed. The groups and basic requirements to each are briefly summarized below. Section 3.3 discusses the sizes of the various storages, based on the data rates shown in Table 4.

1. The O² internal storage:

a) HDDs internal to the FLP and EPN nodes: standard technology with sufficient throughput and capacity to serve the buffering and event building functions of these front-end system. The data will be volatile, thus no redundancy is required. The storage should allow for fast I/O and will sustain large number of file operations (FOPs). Depending on the industry development of storage media for servers, Solid State Drives (SSDs) may become sufficiently robust and financially attractive to be considered for this function;

b) Large capacity dedicated storage system for reconstructed and possibly conditions data store (see paragraph c) as depicted in section ‘Storage’ in Fig.1. The capacity and throughput depend on the data volume, which in itself is a function of the achieved data compression at the upstream readout levels and the reconstruction output. Thus several scenarios must be considered for the design and size of the final storage system. We assume that the structure of the storage and management software consists of independent large capacity data servers for the former and either a shared-disk file system (type GPFS/Lustre) or disk-pool system (xrootd/EOS) for the latter. Each management option has distinct, non-overlapping positive aspects, which must be considered depending on the choice of processing farm and software. The management option choice will also have considerable impact on the price of the storage system, thus the type must be taken sufficiently in advance.

The primary function of the storage will be to hold the partially compressed and calibrated RAW data until it is migrated off the O² system toward external storage. Thus the data may reside on the O² system storage for considerable period of time and a strong requirement is a sufficient redundancy (safety) factor;

c) The final design of the conditions data store may rely on Database Management System (DBMS). In this case a storage specifically tailored to support the chosen DB technology may have to be deployed as a part of the O² system. The decision on the type and capacity of the storage can only be taken after the basic design is completed and the DBMS is chosen.

2. The external Grid storage

Although not directly part of the O^2 system, we present here a possible use of the Grid storage, and how it could fit in the global data flow. The following is one possible example implementation.

The Grid historically provides two types of storage: disk- and tape-based

a) The disk-based storage is typically a set of independent, geographically distributed over tens of computing sites, disk-pool systems with capacities ranging from few tens of TB to few PB. These expose various communication protocols, with ALICE using exclusively the xrootd protocol. The disk-based storage offers medium degree of data security, usually through hardware RAID on the individual disk servers. Some data loss is expected to occur and safety of the data is achieved by replication on two or more independent storages, located at different sites. Recently developed storage management solutions like EOS at CERN offer extremely high degree of data security through emulation of RAID across separate JBOD disk servers. EOS may be very attractive also for the O^2 system storage as it is based on cheap hardware, but offers a data security and small file loss frequency almost equal to a tape system.

The capacities of the Grid disk-based storages are determined by two factors – the size of the hosting computing centre (storage is proportional to the amount of deployed CPU) and the resource pledges of the national funding agencies. Although the pledges are roughly proportional to the requirements of the experiment, there are certain limits to what can be requested. Thus the capacity and yearly growth of the storage is deterministic over long time period, of the order of 3-4 years. It is estimated that the Grid disk-based storages will have sufficient capacity to store the Analysis Object Data (AOD) from the RAW data reconstruction and MC simulations, which correspond to approximately 15% of the uncompressed and unprocessed RAW data size. The storage will be mostly accessed by tasks running on the Grid itself. The AODs will be written by the reconstruction tasks running on the O^2 system, thus the system must have sufficient network throughput and WAN connectivity to be able to write to the sites with disk-based storage.

b) The tape-based, or custodial, storage is offered by few large computing centres, including CERN and provides high security storage medium. Due to the large initial investment in the tape systems, these will likely remain in operation during the ALICE Run 3. The access to tape-based storage is strictly controlled. The efficient operation of the tape systems depends critically on minimal and maximal file size as well as a sequential write and read pattern. Thus the tapes will be mostly used to stream compressed RAW data after the initial filtering is completed on the O^2 system and will be recalled back for subsequent passes. Another function of the tape-based storage is to assure high security of the RAW data through replication from CERN to other external centres. The access to RAW can be done from all centres which have a copy of the data, CERN and remote, since the network infrastructure is already in place to support up to 10Gb/sec data transfer between CERN and each of the remote centres with tape-based storage. It is likely that the network throughput will be significantly higher in the future.

2.3.4 Reconstruction

CWG1 contacts: Peter

Concerned CWGs: 6, 7, 13

Reconstruction is the processing of the raw data contained in Time Frames (TF) producing Event Summary Data (ESD), suitable for physics analysis. The following terms are used (see [4]):

Digit: This is a digitized signal (i.e. ADC count) obtained by a sensitive pad of a detector at a certain time.

Cluster: This is a set of adjacent (in space and/or in time) digits that were presumably generated by the same particle crossing the sensitive element of a detector. In case of calorimeters the estimated energy of the cluster is calculated as well.

Reconstructed space point: This is the estimation of the position where a particle crossed the sensitive element of a detector (often, this is done by calculating the centre of gravity of the 'cluster').

Reconstructed track: This is a set of five parameters (such as the curvature and the angles with respect to the coordinate axes) of the particle's trajectory together with the corresponding covariance matrix estimated at a given point in space. An extended representation of a reconstructed track includes the indexes of the associated clusters and can be used for calibration purposes. The particle identification information (PID) is associated to the reconstructed track.

Reconstructed tracklet: This is a part of the track that belongs to subset of sensitive elements. The full reconstructed track can be seen as association of all tracklets created by the same particle,

Primary vertex: The primary vertex is the estimated position of the interaction between the beam particles.

V^0 decay: Decay of long-lived neutral particle such as Λ^0 , K_S to two charged particles.

Secondary vertex: Point at which a particle decayed to two or more charged particles and zero or more neutral particles.

Kink: Point at which a charged particle decayed to one charged and one or more neutral particles.

The ESD typically contain the primary vertex, the tracks, the clusters in the calorimeters and other non-tracking detectors, the V^0 candidates and the kinks. It also contains information about the interaction time, the trigger(s), etc.

An extended ESD format may include the clusters associated to each track in order to allow re-reconstruction of ESD without need to go back to the original raw data.

Intermediate formats are also possible: for example we can keep only the sets of clusters associated to tracks, the calorimeter clusters, the primary vertex, etc. The cluster parameters are compressed taking into account the precision of the local reconstruction. The covariance matrix of each cluster is stored in compressed way as well. In such intermediate format the track parameters are recalculated using the up-to-date alignment and calibration. Another example is coming from the current HLT: we can store the track parameters and the deviations of the clusters to the track in a compressed way. Such intermediate formats may be considered as replacement for both ESD and raw data.

The reconstruction uses the calibration and the alignment of the detectors, the global run information, the magnetic field map, and the specific reconstruction parameters of the detectors as input information together with the raw data.

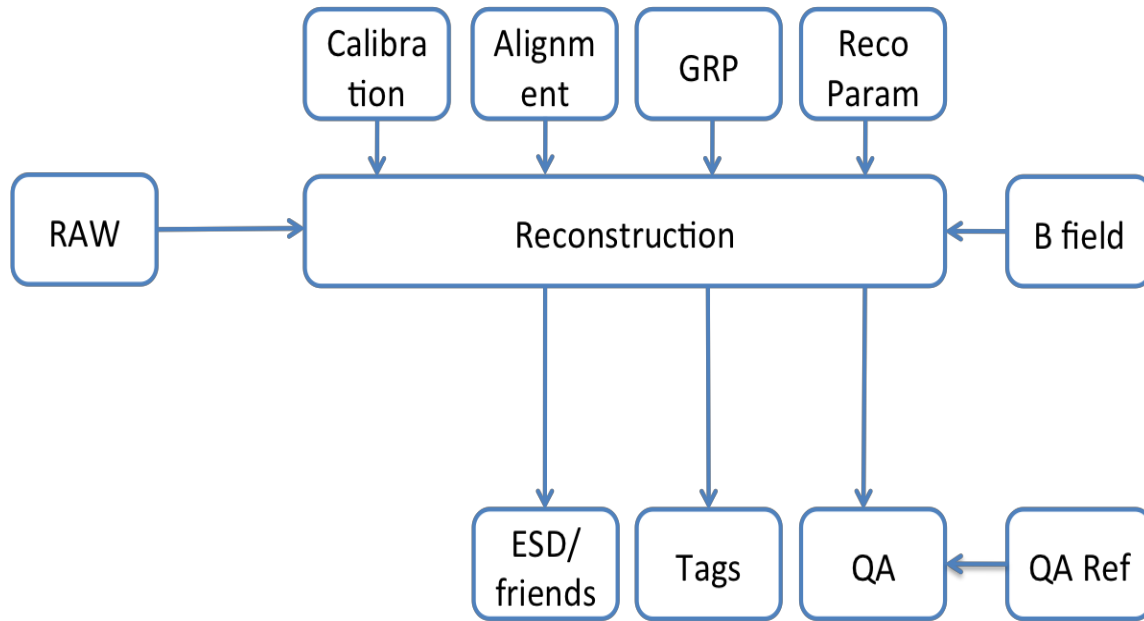
Typically the reconstruction is done in two main steps:

Local reconstruction when the detector response or derivative thereof like cluster or tracklet are found and associated to a certain interval in time, i.e. bunch crossing (BX). The local reconstruction is an essential part of the data reduction algorithms and for many detectors will run on FLPs. It may depend on the calibration, see Fig. 2. In some specific cases the local reconstruction may run also on EPNs.

Global reconstruction: the clusters, tracks and other reconstructed objects are associated with a primary vertex. At this stage the interactions in a TF are disentangled and stored in the ESD. Such direct association of the reconstructed information to event interactions is done using the trigger and time information, the matching between the tracks in the different detectors and to the primary vertex. The global reconstruction will run on EPNs.

Since the calibration in some cases depends on the quality of the reconstruction (i.e. TOF), the reconstruction and calibration may need several iterations

The data flow is represented in Figure 2.



The tags permit fast selection of events based on certain criteria such as number of tracks, etc. They are present in the current ALICE reconstruction and can be optionally considered for the ALICE upgrade.

The automatic quality assurance (QA) is used to verify if the reconstruction is acceptable for physics analysis. It also monitors the behavior of the detectors and issues alarms in case of problems. The QA compares the actual distributions with some reference ones that are prepared in advance.

2.3.5 Simulation

CWG1 contacts: Peter
Concerned CWGs: 8, 13

The simulation consists of the following consecutive steps:

Event generation: usually this is an external package that takes as input the type of the beam particles and their momentum. It produces the kinematics tree of the particles created in the collision (primary particles).

Detector description: it includes the geometry shapes or volumes, their size and position, and the materials of the detectors. Important part is the description of the sensitive elements in each detector and their response function. The information about the alignment is used at this step.

Particle transport: this is an external package such as Geant3, Geant4 or Fluka. It takes as input the detector description and the kinematics tree from the event generator, and transports all primary particles through the detector material. Each particle is moving in the material of the detectors by making steps, and the magnetic field is taken into account in the movement of charged particles. At each step the physics processes (energy loss, scattering, interactions, decay, etc.) are simulated. The resulting secondary particles are added to the kinematics tree and transported as well. The results of the particles transport are typically energy depositions in the sensitive elements of the detectors, known as *hits*.

Digitization: the hits created during the particle transport are collected in each sensitive element and typically converted to charge. The analog response of each detector is taken into account. The resulting amplitudes are known as *summable digits*. They allow the combination of two simulated events into one, using a special procedure called *event merging*. They also enable the addition of simulated events or parts of simulated events to a real event by a procedure called *embedding*. Summable digits are converted to *digits* using the actual thresholds and analog-to-digit conversion algorithm in each detector. The information about the calibration of detectors is used during the digitization.

During the particle transport, the creation of summable digits and digits the Monte-Carlo (MC) truth (the indexes of the particles in the kinematics tree) is propagated from hits to summable digits and then to digits. This permits reliable estimation of efficiencies, fake rates and resolutions later in the reconstruction of MC data.

The MC digits can be optionally converted to the real raw data format (including different possibilities for compression) and the resulting time frames stored for future reconstruction. The MC truth usually is lost during the conversion from digits to raw format, but a separate mapping can be foreseen in case of request.

The output of the simulation consists in kinematics tree, MC digits and/or simulated raw data. The MC digits or simulated raw data are used as input for the reconstruction, while the kinematics is used in the analysis.

2.3.6 Analysis

CWG1 contacts: *Peter*
Concerned CWGs: 13

The analysis of experimental data is the final stage of event processing and it is usually repeated many times. Analysis is a very diverse activity, where the goals of each particular analysis pass may differ significantly [4]. It is coordinated by the Physics board (PB) and the Physics Working Groups (PWG). Typically one analysis task includes some or all of the following steps:

- Selection of events according to the trigger and beam conditions (physics selection);

- Selection of charged tracks following quality, geometrical, kinematical and particle identification criteria;
- Selection of clusters according to quality, energy, etc. criteria;
- Creation of combinations between tracks and/or clusters that are candidates for particle decay of interest;
- Selection of combinations according to geometrical, kinematical, etc. criteria;
- Fitting with or without constraints to extract desired values and/or to improve their resolution;
- Creation of distributions representing important features of the physics process;
- Estimation and subtraction of background(s);
- Estimation of statistical and systematic errors;
- Comparison with theoretical or model predictions.

The optimization of the selection criteria is usually an iterative procedure. The goal is to maximize the signal-to-background ratio in case of search tasks, or the significance of the signal in case of property measurement.

There are three different ways of running analysis:

- Local analysis running on local desktop or laptop;
- Proof-based analysis using one of the analysis facilities where Proof is installed and from where access to the reconstructed data (ESD or AOD) is provided;
- GRID analysis using the GRID resources allocated to the ALICE experiment.

The analysis itself can be run centrally or individually. When it is run in a centralized (coordinated, scheduled) way, several analysis tasks belonging to a PWG are run together (analysis train) by the person responsible in each PWG. Such analysis trains are run according to a schedule agreed within the PWG. An individual analysis is run by the physicist working on it; usually, no synchronization with the other members of the collaboration is required.

The input for the analysis can be in form of reconstructed data (ESD or AOD), filtered ESD or AOD, or data in specific formats, for example the ones developed for data preservation or outreach. The output of the analysis can contain one- or multi- dimensional histograms, trees, ntuples, filtered ESD or AOD, etc.

2.3.7 Networking - Data Transport

CWG1 contacts: *Timo*

Concerned CWGs:

The current online data-flow design foresees a networking layer which enables the transfer of event/time frame fragments from FLPs to EPNs, such that for each timeframe all associated detector fragments are sent to a single EPN for further processing based on global features.

It is therefore similar to the classical "event building network", however there are also significant differences. Especially the continuous, i.e. timeframe based, read-out mode poses new challenges to the network design: the data is transferred in rather coarse-grained units spanning multiple events in contrast to the classical triggered read-out with a fine-grained, event-based transfer unit. The actual length of the time frames depends heavily on the data reconstruction scheme and is thus subject to R&D (CWG?), although we can assume it to be of the order of multiples of TPC drift times ($\approx 100\mu\text{s}$). Likewise, the total amount of data to be transferred can only be estimated, since it is a function of the compression ratio applied on the FLPs which is also under investigation. Early compression implies a trade-off between network bandwidth against CPU resources on the receiving side (for decompression). The final scheme will be a compromise, the driving force for optimization will most likely be overall cost.

2.3.8 Calibration

CWG1 contacts: *Frank*
Concerned CWGs: *CWG-6*

The main purpose of calibration is to prepare the information needed by the reconstruction in order to correct the raw data for detector effects (e.g. electronics, response to the signals...), environmental and data taking conditions that could affect the quality of the data and introduce biases.

The calibration algorithms may run at different stages of the data taking. They might be procedures executed on the FLPs, or on the EPNs (both at the intermediate or at the final stage of the processing chain). Eventually, second order calibration corrections can be applied at analysis level in some specific cases (e.g. for calorimeters, as it was done during Run1).

Calibrations can be carried out during non-physics or physics data taking. This data is typically taken in standalone mode for the different detectors, using specific triggers. The output of these calibrations has to be made available for the reconstruction of subsequent PHYSICS data, aimed at collecting the events for physics analysis. In the scheme of a (quasi) online reconstruction, this imposes constraints on the availability in time of such calibrations. Examples of calibrations using non-physics data can be seen in Table 2.

Table 2: Examples of calibrations using non-physics data

Detector	Calibration	N. Of Events	Where	Input	Comment
EMCAL	Time independent energy calibration		Offline	QA from CPass	
	Absolute calibration		Offline (?)		
	Dead maps + pedestal		Pedestal run? Done offline		
TOF	Status		Noise runs	DAQ	
FMD	Gain		Standalone run	DAQ	
	Pedestal		Standalone run	DAQ	
	Secondary particles maps		Offline	MC	
	Acceptance		Offline	Gain + Pedestal	
TRD	Pedestal + noise		Pedestal runs	DAQ	

PHYSICS data taking can be used to produce calibration, especially for the cases when such calibrations are strongly time dependent, and when they require tracking, (see Table 3 below). They can both be the outcome of the processing of the Physics data themselves, or of dedicated triggers interleaved with them. Information on detector configuration, detector specific parameters (e.g. HV, currents, temperatures...) can also be a source of information for calibration, and should be made accessible by the calibration algorithms.

Table 3: Examples of calibration using physics data

Detector	Calibration	N. Of Events	Update frequency	Input/When	Depends on...	Comment
EMCAL	Tower gain	O(100)M	1/year	Reconstructed π_0		
TOF	TOFill	1k events (pp)	1/5 mins	During CPass0/1	TPC calibration	BPTX at Run3?
	Channel Offset	10M events (pp)	1/period	From CPass1 output	TPC calibration	Hopefully done once for all at Run3
	Time Slewing	100M events (pp)	1/year	From reconstructed events	TPC calibration	
	Problematic channels	10M events (pp)	1/period	From CPass1 output	TPC calibration	
	Readout Efficiency		1/run	DAQ		
	Status		1/run	DAQ		

These were used in Run1 but will form the basis of requirements for Run3. According to detector experts, it is too early to be precise about the calibration requirements for Run3. A full table will be put in annexe and will be updated as the information is sent in by the detector experts.

Global Run Parameters (GRP), including beam type, beam energy, run duration) should be collected as they are of fundamental importance for the reconstruction.

Calibration on one FLP will have to rely on the data collected by the part of the detector will be read out in the FLP. Here, only procedures like cluster finding in the case of the TPC will be done, due to the fact that only a limited fraction of a detector will be read out, which limits the types of calibration possible. Nevertheless, merging procedures could be foreseen over the output from the various FLPs. Moreover, calibrations on the FLPs could profit of the fact that all data will be seen there. The output of the procedure could be either stored in a database, or embedded in the data.

Each EPN will receive data for a single time frame, collected from all FLPs. In this sense, the calibrations done on the EPNs will have the possibility of being based on the data collected by all the detectors, with the limitation coming from the asynchronous reconstruction of the different time frames in the different EPNs. At this stage, calibration procedure could rely on reconstructed data already corrected for the calibrations coming from calibration dedicated runs and/or for those obtained at the previous step in the FLPs. The final goal of the processing in the EPN will be to store on disk the data with the highest reduction factor possible. The level of data reduction time will depend on the quality of the calibration that could be applied during the reconstruction and should be defined in order to limit as much as possible the risk of loss of useful data. For example, it could be decided that only clusters unambiguously identified as background are dropped, instead of removing them when they are not associated to tracks; the latter approach could cause irreversible loss of data if a non-perfect calibration, resulting in

inefficient tracking, were obtained. In other words, the data reduction factor should reflect the level of calibration, and as such could be different depending on the data it is applied to (for example, a higher reduction could be possible at the outer region of the TPC, where lower occupancy and less space-charge effects are expected).

The final reconstruction step before storing the data on tape should be performed with the best calibration possible to allow the best data reduction. The output of this step will become our new raw data and will be used, together with possibly further improved calibrations, as input for later re-reconstruction passes to finally produce AOD objects aimed at physics analysis.

It is important to guarantee the homogeneous quality of the AODs, and to prepare the calibrations so that Monte Carlo simulations, anchored to the data, are able to reproduce the same performance as was achieved for the real data.

The calibration algorithms have to be written in such a form that allows to re-run them at any time (possibly with improved implementations) to regenerate AODs starting from the stored data.

Calibration inputs

- **Raw:** DDL;.root; memory
- **DCS points:** DCS File Exchange Server (FXS): temperature, pressure, HV, LV, FEC etc
- **ECS :** FXS
- **DAQ:** FXS; electronic logbook
- **Alignment survey data**
- **ESD:** Calibration events; laser events; cosmic and physics triggers

All the above become objects in the **OCDB** to provide the calibration **Output: AODs**.

The calibration objects could be eventually stored in a database as time dependent objects.

2.3.9 Condition and Calibration Database

CWG1 contacts: *Sylvain, Barth*
Concerned CWGs: *CWG-3 & CWG-6*

The condition and calibration database provides the processing algorithms (reduction, reconstruction, calibration, monitoring) with the necessary data to process the events. It consists of calibration results, environmental measurements, etc.

The database may be populated at different stages of the data taking and processing. It is both readable and writable. It may be accessed by a large number of distributed processes. It should allow replication (i.e. to make copies of the database on other sites, or for load balancing and scalability purposes), either by snapshots, or by continuous streaming. The access latency should be kept low (a fraction of the processing time).

The data may be time series (e.g. drift velocity, temperature, ...), or values valid for a given period of time (e.g. gains, detector configuration, ...). The database should be populated in real time when possible, so that its client may get latest data quickly.

A large number of “objects” (to be defined - typically calibration results or processed values, as opposed to ‘raw’ environmental measurements in time series) may also be stored. They are accessed by a key, and associated with a binary content. The following metadata may also be necessary for each object: timestamp, validity range, version, source, access statistics, ... In some cases the objects may be overwritten (versioning feature needed).

The data access rates and storage size to be defined concern read, write, update and delete operations (number of objects or data size, per second or per run). The distribution of the (changing) calibration data to the working nodes at runtime should carefully be investigated, as it may represent a substantial bandwidth. The transport mechanism should be scalable with the number of nodes and amount of data.

It should be possible to delete content (e.g. to manually clear wrong values, although it could simply be overwritten), but the corresponding usage is minimal.

The database will grow over time, and the design should be scalable with the expected rate.

A publish/subscribe feature would be helpful to notify real-time clients of new incoming values (changes in conditions). Such a feature would avoid polling, and ensure a minimal time between when a value is measured or a result produced, and when it is used.

The concept of validity should be clarified. Is it needed at all? Can all queries simply be timestamp-oriented (give me the closest good value to this timestamp)? This would cover the cases of objects being updated on short intervals (a much finer granularity than a run is needed).

2.3.10 Data Monitoring (QA/DQM/etc)

CWG1 contacts: Barth

Concerned CWGs: CWG-9 (& CWG-6)

Definition

Both Data Quality Monitoring (DQM) and Quality Assurance (QA) use analysis software to check the quality of the data itself. In general, DQM refers to online execution of such algorithms to avoid the recording of low quality data. QA, on the other hand, commonly refers to the evaluation of the data after it has been recorded and processed in an attempt to sort out good and bad runs for physics analysis. In view of continuous readout, online reconstruction and calibration requirements for RUN3, it will be difficult to differentiate between DQM and QA. For this reason, a single monitoring system will be described keeping in mind that the source of the data and the analysis might be different.

Users

At each stage of data processing, data quality is controlled and monitored by users with different levels of expertise: shifters, run coordinators, detector experts (including on-call experts), calibration experts and finally analysts in Physics Working Groups (PWGs). Each kind of user has to define the format of the DQM/QA output and requirements on the graphical interface in order to have a comprehensive picture of the quality of the data sample.

When taking data, DQM/QA shifters are the first local users of the DQM/QA output. As they are not usually detector experts, shifters get direct support from the detector on-call experts and the run coordinators, who need to access information on the data being collected as early as possible and remotely. In this way, immediate action can be taken in case of any problem with the data being taken.

Calibration experts will need to be able to monitor data quality remotely both during and after the calibration stages. The last, but far from the least, to use the data quality mechanisms are the PWGs and the physics analysts who may not be detector experts. They do not need to access the details of the entire quality control process but need a clear statement on the usability of the data samples for given analysis and a summary of the potential issues. Analysts (any Collaboration member, in principle) will need to be able to access the information on data quality from a distance.

Input and Output

The DQM/QA system receives data at different stages of the online data flow or from the offline data storage. Information about the conditions in which this data were collected, and the related calibration, must also be made available to this system. The DQM/QA provides objects (histograms, graphs, values, ...) and a quality assessment for all or most of them. The quality assessment is provided by the setting of quality flags, e.g. Undefined, Ignored, Good, Bad, Warning. Different types of monitoring can be carried out, depending on the time and information available, in order to assess or refine the quality of an object. In addition, the DQM/QA output objects will require metadata allowing indexing the monitoring information and reducing the need to collect such information from additional systems.

Data collection

The online collection of sufficient random samples is done in a way that allows a statistical analysis without slowing down involuntary the data taking. Moreover, the QA can run on data samples from different stages such as FLPs (raw data, sub time frame) or EPNs (full time frame, reconstructed events). The data collection accepts different types of "requests": e.g. from specific detectors or partitions and for certain types of events. This data collection system might as well be used by the calibration, the reconstruction and other components of the O2 framework. Since other systems apart from DQM/QA, are involved in data collection it is not clear, at the time of writing, which will be the provider.

Automation

In online mode, the QA results cannot be analyzed manually. Results, in the form of histograms or other objects, should be evaluated by algorithms that produce a clear and discrete value representing its quality. Such value could allow the data flow procedure to take unambiguous decisions and actions in an automatic way based on discrete quality values. This is especially important for critical online processing stages (calibration, reconstruction, ...). The same (or similar) evaluation could be adopted offline for simple checks or analysis.

Access to results

The results of the DQM and QA must be available worldwide to the Collaboration members (only). A public access shall be granted to a selected set of interesting data and results for Public Relations (PR). The results of both the online and the offline QA need to be available in the same place, possibly separately.

Event display

The Event Display refers to a program able to display an event in 3D and able to manipulate it. The event must have been partially or fully reconstructed beforehand and can therefore be in an intermediate or final format. The event usually comes from EPN or the Grid although the source is irrelevant to the Event Display. It can also consist of simulated data. The user can customize the view by coloring or hiding tracks and retrieve details about them. Manual or automatic navigation between events from different sources will be possible.

2.3.11 Software Framework

CWG1 contacts: *Predrag*

Concerned CWGs:

A software framework is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing desired functionality. It provides reusable abstractions of code wrapped in a well-defined Application Programming Interface (API). The software frameworks control the overall program's flow but they are

designed to be extensible. In a typical example, the framework calls user functions that extend default implementation provided by the framework itself.

Users are normally not allowed to modify the framework code and alter program flow but they are offered possibility to selectively override or specialize its well defined parts.

AliRoot, current reconstruction, simulation and analysis software developed and used by ALICE Collaboration is an example of such software framework. It is written in C++ and uses standard OO techniques such as inheritance and subclassing. Built directly on top of ROOT, AliRoot in a way extends the ROOT framework with ALICE specific classes and uses CINT (C++ interpreter provided by ROOT) as an ultimate framework that steers program execution.

While this approach has proven to be very flexible and allowed us to quickly prototype and develop very complex code containing several millions of lines of code, it also exposed a number of problems such as difficulties in finding and isolating memory leaks, continuously growing large memory footprint, memory fragmentation issues, direct dependency and coupling to ROOT development lifecycle, difficulty to use threads...

In the past, huge advances in technology and frequency scaling allowed the majority of computer applications to increase in performance without requiring structural changes or custom hardware acceleration. While these advances continue, their effect on modern applications is not as dramatic and the primary method of gaining extra performance out of computing systems is to introduce additional specialized resources, thus making a computing system heterogeneous.

As; the modern high performance computing platforms are evolving towards a hybrid, heterogeneous model, using both CPU and GPU cores used for general purpose computation, several languages or (CUDA, OpenCL) are being developed to fully harness the potential of these processors. This typically means running the control code on the CPU and running the performance-critical, data-parallel kernel code on the GPUs.

This allows the use of multiple types of processing elements; each one able to perform the tasks for which it is best suited for. With the addition of extra, independent computing resources most heterogeneous systems can be considered to be parallel computing, or multi-core (computing) systems requiring a concurrent programming approach.

Concurrent computing is a form of computing in which programs are designed as collections of interacting computational processes that may be executed in parallel. Concurrent programs (processes or threads) can be executed on a single processor by interleaving the execution steps of each in a time-slicing way, or can be executed in parallel by assigning each computational process to one of a set of processors that may be close to each other or distributed across a network.

The main challenges in designing concurrent programs are ensuring the correct sequencing of

the interactions or communications between different computational executions, and coordinating access to resources that are shared among executions. A number of different methods can be used to implement concurrent programs, such as implementing each computational execution as an operating system process, or implementing the computational processes as a set of threads within a single operating system process.

As we do expect that such alternatives to pure CPU will be used on large scale to speed up specific parts of ALICE event processing, the next major version of AliRoot framework (6.x) has to support such heterogeneous computing systems and concurrent computing models spanning across the nodes in the O² farm.

At present, ALICE HLT is already supporting such heterogeneous systems (CPUs, GPUs and FPGAs) using a custom built framework providing and implementing a publish-subscribe pattern. While this has been shown to work, we should also investigate other solutions that could provide off-the-shelf Open Source implementations of various patterns like: N-to-N via fanout; pubsub; pipeline; request-reply. These implementations can ensure ultimate flexibility, high performance messaging and asynchronous I/O on top of various wire protocols while supporting the major operating systems (including Linux, Windows, OS X) and binding to all major programming languages.

While an approach driven by the messaging framework would allow us to run our software on all hardware platforms (from a laptop with one or many cores to an entire O2 facility with many thousands of cores and specialized hardware such as GPUs, FPGAs or MIC accelerators), it will also mark a significant departure from traditional software framework as implemented in current version of AliRoot. What is today a single threaded application will be transformed into many small components running concurrently as independent processes (executing on the same node or distributed over the network) with some of them utilizing capabilities of specialized hardware (where available) and communicating by asynchronous messaging.

2.4 User characteristics

The users of the system are to various degrees all members of the ALICE collaboration, more than 1000 physicists from more than 100 institutes from all over the world. It has to cope with at least the user profiles of end users, developers and administrators.

End user

An end user is expected to be internet literate, to get access to all results and the corresponding meta-data. He should be able to use analysis macros in his local environment.

Developers

The system has to support different groups of developers:

- physicists who develop analysis macros for their own or their Physics Working Groups use;

- detector specialists who implement the algorithms for digitization, calibration, alignment, clustering, reconstruction, quality assurance;
- core developers for the different subsystems and the overall architecture.

Even after the start of run 3 the O2 framework will need further development. While the core framework and the detector specific algorithms should be relatively stable, the analysis will be highly in flux. A developer should be able to deploy the system in his local environment, understand the software engineering tools used, as well as the programming environment and the APIs needed for his task. For simple task basic OO knowledge should be enough but the system should be coherent and well architecture, so that it is possible to learn on the job how to advance to more complex tasks. For parts of the core advanced IT specialists with in depth knowledge of real time techniques, parallelism and advanced architecture are needed, but the system should be structured to allow the contribution of non-IT-experts to the physics code.

Administrators

For the system administration of the central O2 farm, expert linux system knowledge is needed. The deployment and maintenance of the parts of the software needed on outside farms or Grid/Cloud systems should be as simple as possible. Further administrative roles like production manager/operator, release coordinator, ... are needed; the corresponding tasks should not need in-depth system management knowledge.

A close collaboration between the core developers and the core administrators is needed to maintain a continuous adaption to the changing requirements of physics. Developers have the responsibility for building subsystems that are easy to test, to deploy, to scale and to monitor.

Operators

The online system is operated at the experimental area by trained operators at run time. The high-level interfaces (control, monitoring, error reporting, documentation) should be easy to learn and use.

On-call experts

The first-level support is usually done by on-call experts. Remote (but restricted) connectivity is necessary to help the operators in case of problems. Web applications have the advantage of facilitating this remote access, in comparison with platform specific human interfaces. This should be taken into account when developing GUIs.

2.5 Constraints

The data volume and bandwidth at the input of the O2 online system are fixed by the detector electronics and characteristics. Table ? summarizes the expected event sizes, subject to change when more details about the upgraded detectors will become available.

Detector	TPC	TRD	ITS	Others	Sum
Event Size (MByte)	20	1.6	0.8	0.5	22.9

Table 1: Estimated event sizes for MB Pb-Pb events (from [1]).

During the lifetime of the experiment, LHC is expected to provide on the order of 50-100kHz of Pb-Pb collisions. The detector systems as well as the readout should be able to handle 100kHz, the O2 system in Run 3 is expected to handle 50 kHz (with an option to handle 100 kHz). The event rate in p+p collisions will be limited by the detector read-out and not the accelerator itself (leveled operation). Details on the rates the detectors can handle depend on the actual implementations and will only be available after the relevant TRDs have been submitted. In the LoI a baseline of 2 MHz p+p collisions was used, however several key detectors will only be able to handle O(100 kHz).

Here is a copy of the detector readout rate from the LOI p.86 table 3.1:

Detector	ITS	TPC	TOF	TRD	EMcal	Muon
Maximum readout rate (khz)	continuous	continuous	200-400	27-100	50	5

For some of the calculations in this document, the average uptime of the LHC as well as the luminosity decay in heavy-ion collisions due to IBS have to be taken into account. These numbers are not expected to change after LS2, we use an LHC uptime of 50% (Huebner-factor 0.5) and an average collision rate of 20 kHz over one heavy-ion fill.

As the current online systems, the new O2 farm has to fulfill the relevant requirements on reliability, availability and security. Currently up-times from more than 98% are achieved.

3 Specific requirements

In this section the requirements for each of the functional modules as defined in section 2.3 are listed.

3.0 Detector&Systems

To ease the processing, the O2 system will set some requirements on the detector and other systems.

We list here the identified points:

- TPC data ordered by pad-row/pad+time to allow for local cluster finding
- Trigger to issue special triggers for time-frame definition

3.1 Data Input

3.1.1 Data links

- The online system should be able to input data from different types of DDL depending on the detector.
- The data rate of each detector can be absorbed by one or several DDLs.
- The system allows several (up to hundreds of) links for a given detector.
- The throughput of a DDL can be handled by a single computer.
- The DDL3 link speed is between 10Gb and 100Gb to keep the number of FLP nodes reasonable.
- Several DDLs can be connected to a single computer (link aggregation).
- The transfer through DDL preserves data integrity.
- The DDL accommodates data from buffered and unbuffered detectors.
- The DDL3 can be used to for the following (possibly concurrent) tasks:
 - readout physics data from the detectors;
 - send data to the detectors (e.g. electronics configuration);
 - readout control data from the detectors.

3.1.2 Data readout (FLPs)

- Each FLP should be able to cope with an incoming data rate of 100Gb per second.
- Each FLP should be able to buffer incoming data for a few seconds in order to allow smoothing peaks in incoming/outgoing data traffic and processing latencies.
- FLPs should be able to process incoming data with a latency compatible with the available memory size and incoming data throughput.

3.2 Data reduction

- Reduce input data bandwidth from 1.1 TByte/s to 82.5 GByte/s, see table 4 for details and required compression factors for the detectors.
- If needed, data should be reduced on the detector read-out cards to meet bandwidth limit of FLP computing bus (for example, no more than 16 GByte/s for PCIe Gen3 16 lanes).
- Data reduction on FLPs to stay within available network bandwidth, see section 3.7 for limits.
- Data reduction needs to run synchronously with data taking, i.e. with the same rate.
- Compression factors need to be reached for all running conditions.

Detector	Input Bandwidth	Peak Output to Local Data Storage	Compression Factor	Average Output to Tape
TPC	1000 GByte/s	50 GByte/s	20	8.0 GByte/s
TRD	81.5 GByte/s	10 GByte/s	8	1.6 GByte/s
ITS	40 GByte/s	10 GByte/s	4	1.6 GByte/s
Others	25 GByte/s	12.5 GByte/s	2	2.0 GByte/s
Total	1146.5 GByte/s	82.5 GByte/s	14	13.2 GByte/s

Table 4: Data rates for the major ALICE detectors, estimated compression factors and corresponding peak output to local data storage as given in the LOI. Taking the LHC intensity variations during the fill and the uptime into account, the expected rates to the permanent tape storage can be calculated.

3.3 Data storage

The O2 internal storage, as well as the required Grid storage capacities can be calculated using the rates and compression factors shown in Table 4 and the projected LHC performance during Run3.

Average recorded data output of 12 GByte/s.

The simulated and reconstructed data on disk amounts to 110 PB out of which 70 PB are for simulated data. The raw data of real events on tape for a total of 330 PB.

- Local storage
- Permanent storage

3.4 Reconstruction

3.4.1 Functional requirements

- Handling of time frames.
- Reading/decoding of raw data format from memory or file.
- Reading of MC digits format from file.
- Access to calibration and alignment information, magnetic field map and global run information.
- Local reconstruction: clusterization algorithms.
- Local reconstruction: tracklet reconstruction.
- Local reconstruction: reconstruction of primary interaction vertex (i.e. using only ITS).
- Global reconstruction: barrel tracking (i.e. matching of tracklets and clusters to form tracks)

- Global reconstruction: tracking in the MUON arm.
- Global reconstruction: matching of tracks to the primary interaction vertex to form event.
- Global reconstruction: reconstruction of V0.
- Global reconstruction: reconstruction of kinks.
- Global reconstruction: reconstruction of cascade decays.
- Global reconstruction: final association of reconstructed objects (tracks, clusters, V0s, etc.) with primary interaction vertex and separation of time frames to individual interaction events.
- Global reconstruction: particle identification.
- Creation of event tags.
- Quality assurance.
- Propagation of MC truth from MC digits to clusters, tracklets, and tracks.
- Bookkeeping - possibility to retrieve the version of the software, the reconstruction macro, the versions of the OCDB objects (calibration and alignment), etc. Possibility to repeat the reconstruction in the same conditions as it was done previously.

3.4.2 Physics Performance requirements

The physics performance requirements are extracted from the Physics Performance Report [5,6].

- Reconstruction efficiency for pp, pA and AA as function of momentum, detector occupancy, particle type, etc.
- Fake rate for pp, pA and AA as function of momentum, detector occupancy, particle type, etc.
- Amount of pileup events for pp, pA and AA.
- Resolution of track/cluster parameters for pp, pA and AA as function of momentum, detector occupancy, particle type, etc.
- PID capabilities for pp, pA and AA as function of momentum, detector occupancy, particle type, etc.
- Mass resolution for specific signals (V0s, cascades, D-mesons, J/ψ , Υ , etc.)

3.4.3 Interface requirements

- Input: fixed formats of raw data, MC digits, calibration and alignment information, magnetic field and global run parameters.
- Output: fixed formats for ESD, event tags and QA output.

3.4.4 Operational requirements

- Possibility to configure the reconstruction chain from a scripting language macro.
- Possibility to store intermediate information on demand.
- Possibility to stop the reconstruction, create a restore point and restart the reconstruction from such point.

- Exception handling.

3.4.5 Resource requirements

- The CPU consumption per event must permit the full processing of the data flow.
- The resident memory consumption should be less than 2-4GB per core(?).
- The virtual memory consumption should be less than 4-6GB per core(?).

3.4.6 Validation and testing of the system requirements

- The full reconstruction chain has to be tested on selected raw data and on the corresponding anchored MC data for pp, pA and AA.
- The performance of the reconstruction has to be within the specified range.
- The validation procedure should be able to run within the O2 farm and outside and give similar results

3.4.7 Documentation requirements

We need user manual, documentation of the reconstruction algorithms and their performance.

3.4.8 Portability requirements

- We should support several Linux distributions (at least SLC, Ubuntu and Debian), MacOS and Windows(? depends on support for the external libs we depend on).
- The code should compile with the main available compilers on each platform
- Architectures ?
- Backward compatibility: possibility to process old data with new software versions.
- Forward compatibility: possibility to process new data with old software versions(i.e. reconstruction of new raw data with old version of the reconstruction).

3.5 Simulation

3.5.1 Functional requirements

- Possibility to use different event generators for pp, pA and AA collisions.
- The same detector description for all transport packages.
- Possibility to use different transport packages (Geant3, Geant4 and Fluka).
- Access to calibration and alignment information, magnetic field map and global run information.
- Creation of summable digits for each detector.
- Creation of digits from summable digits from each detector.
- Creation of digits from hist for each detector.
- Creation of summable digits from raw data from each detector.
- Event merging procedure for each detector.

- Embedding procedure for each detector.
- Propagation of MC truth (i.e. labels of particles in the generated kinematics tree) from hits to summable digits and digits.
- Bookkeeping - possibility to retrieve the version of the software, the reconstruction and configuration macros, the versions of the OCDB objects (calibration and alignment), the random seed, etc. Possibility to repeat the simulation in the same conditions as it was done previously.

3.5.2 Performance requirements

- Similar material distributions for MC and real data (i.e. using gamma conversions)
- Similar distributions for the variables used for event and track/cluster selections for MC and real data

3.5.3 Interface requirements

- Input: interfaces to existing event generators.
- Output: fixed formats of raw data, MC digits, calibration and alignment information, magnetic field and global run parameters, metadata (i.e. Root and AliRoot versions, versions of OCDB files).

3.5.4 Operational requirements

- Possibility to configure the simulation chain from a scripting language macro.
- Possibility to store intermediate information on demand.
- Possibility to stop the simulation, create a restore point and restart the reconstruction from such point.
- Exception handling.

3.5.5 Resource requirements

- The CPU consumption per event must permit the full processing of the data flow.
- The resident memory consumption should be less than 2-4GB per core(?).
- The virtual memory consumption should be less than 4-6GB per core(?).

3.5.6 Verification and testing requirements

- The full simulation chain has to be tested using anchored MC data for pp, pA and AA.
- The comparison with the distributions from the raw data should be within the specified range

3.5.7 Documentation requirements

We need user manual, documentation of the detector response simulation.

3.5.8 Portability requirements

- We should support several Linux distributions (at least SLC, Ubuntu and Debian), MacOS and Windows(?).
- The code should compile with the main available compilers on each platform

3.6 Analysis

3.6.1 Functional requirements

- Access to ESD and AOD in a uniform way without changes in the user's code.
- Support for ESD to AOD conversion (filtering).
- Support for AOD extensions (delta AOD), for selection of (rare) events (event skimming).
- Support for local, proof and GRID modes in uniform way.
- Support for coordinated and individual analysis.
- Tools for physics selection, for cuts, for fits with or without constraints, etc.
- Tools for background estimation, including event mixing.
- Tools for efficiency estimation, efficiency correction and unfolding of distribution.
- Backward compatibility: possibility to run new software on old ESD or AOD.
- Possibility for automatic QA.
- Bookkeeping - possibility to retrieve the version of the software, the analysis macros, the versions of the OADB objects, etc. Possibility to repeat the analysis in the same conditions as it was done previously.

3.6.2 Performance requirements

- Possibility to analyze the data from one month of data taking in 72h.
- Ratio between CPU and wall time above 75%.

3.6.3 Interface requirements

- Input: reconstructed raw or MC data in form of ESD or AOD.
- Output: histograms, trees, ntuples, tables, etc.

3.6.4 Operational requirements

- Possibility to configure the analysis chain from a scripting language macro.
- Possibility to store intermediate information on demand.
- Possibility to stop the analysis, create a restore point and restart it from such point.
- Exception handling.
- Monitoring of the analysis tasks included in a train.
- Possibility to disconnect automatically tasks that consume too much CPU, memory or cause extensive IO load.

3.6.5 Resource requirements

- The resident memory consumption should be less than 2-4GB per core(?).
- The virtual memory consumption should be less than 4-6GB per core(?).

3.6.6 Validation and testing of the system requirements

- Each analysis has to be tested on reconstructed raw and MC data from a reference sample containing pp, pA and AA runs.
- The performance of the analysis task has to be within the specified range.
- The validation procedure should be able to run on different platforms and produce similar results

3.6.7 Documentation requirements

We need user manual, tutorial and examples of the analysis tasks.

3.6.8 Portability requirements

- We should support several Linux distributions (at least SLC, Ubuntu and Debian), MacOS and Windows(? depends on support for the external libs we depend on).
- The code should compile with the main available compilers on each platform
- Architectures ?

3.7 Networking - Data transport

- **Total (aggregate) throughput**

The network has to be able to handle the following aggregate throughput (all FLP -> all EPN)

$$\begin{aligned} \text{tot. throughput} &= \text{event rate} * \text{total event size} * \text{FLP compression factor} \\ &= 50\text{kHz} * 23\text{MB} / O(5) \\ &= O(230\text{GB/s}) \end{aligned}$$

- event rate: Lol
- event size: Lol, detector
- FLP compression factor: R&D -> CWG?

These numbers are valid for the initial hardware deployment. The system architecture should be able to scale to 100kHz to have some headroom for future upgrades.

- **Transfer unit**

The total size of all (FLP-compressed) data fragments associated to a time frame plays an important role in the data-flow design. Although the data can be send in smaller chunks it has to be assembled on the receiving side (EPN) where it occupies buffer space and can only be processed once the transaction has completed. The transaction involves all (most) FLPs and a single receiver (EPN). At the moment we can assume the minimal length to be a multiple of TPC drift times (100 us).

- $\text{transfer unit} = \text{time frame length} * \text{total throughput}$
 $= O(1000) * 100 \text{ us} * O(230\text{GB/s})$
 $= O(23\text{GB})$
- time frame length: R&D -> CWG?
- **FLP output**
The typical output of an FLP ultimately depends on the number of FLPs which is subject to R&D. A lower limit may be given by the needed CPU resources to fulfill the first compression step, which can come close to zero if the compression is done completely in H/W, plus other task like calibration. It also depends on the chosen detector link protocol and H/W, hence bandwidth, and their number per FLP. However, minimizing the number of FLPs (and hence number of links and switches) might not in all cases state the global optimum in terms of cost, since it might require the use of a significantly more expensive high-bandwidth technology. Again, total cost would be the driving force for optimization.
 - $\text{typical FLP output} = \text{total throughput} / \text{number of FLP}$
 $= O(230\text{GB/s}) / O(250)$
 $= O(920\text{MB/s})$
 - number of FLPs: R&D -> CWG (data-flow, calib, compression, RORC, H/W)
- **EPN input**
Similar to the facts above. Depends on total number of EPNs, which is, among other factors, determined by CPU resource usage of the global processing tasks etc.
 - $\text{typical EPN input} = \text{total throughput} / \text{number of EPN}$
 $= O(230\text{GB/s}) / O(1250)$
 $= O(184\text{MB/s})$
 - number of EPNs: R&D -> CWG (data-flow, calib, compression)
- **Networking hardware**
Given the numbers above and current trends in network technology, there should be a wide range of suitable networking hardware available. Typical technologies at the moment are Ethernet and Infiniband, both with foreseen link bandwidths between 10GB/s and 100GB/s, but other might emerge over time. Those also have similar features, e.g. in terms of possible topologies, but also in terms of network protocols: while IP is the typical protocol for ethernet it is also possible to use RDMA (RoCE, iWARP), and Infiniband has an IP layer (IPoIB) along with its "natural" RDMA mode. Hence the decision for a particular technology might be made rather late in the development process, although small-scale lab setups are required to execute needed tests.

3.8 Calibration

- All detectors will be required to carry out Online calibration, reconstruction and data quality monitoring.
- The online calibration should be based on automatic procedures.
- The output will be calibrated ESDs.
- There will no post-processing as is presently the case.
- Only second order corrections will be allowed and the calibrated ESDs should allow for

these if necessary.

- Possibility to repeat the same calibration algorithms offline --> calibration algorithms should be transparent to the fact that they are running on reconstructed data, or on the data format available in the farm
- Possibility to retrieve information from DCS, both for detector-specific information and for global run parameters
 - What is needed? At which step? When? SOR, EOR...
- Possibility to merge at FLP level calibration object: is this needed for any detector?
- Possibility to recover later in time the calibrations that were used for a specific reconstruction
 - Should the information be stored in the data? In a dedicated database?
- Possibility to assess calibration quality before using it
- Possibility to perform QA on the calibration over an extended time
- Possibility to select interesting events for calibration (on trigger classes when possible, on software trigger/selections...)
- Identification of calibration accuracy at each step: FLP, EPN, final reconstruction
- Identification of possible interdependencies between detector: which level of calibration for detector A is needed to calibrate detector B?
- Evaluation of the calibration performance if constrained to run on the data on one EPN
 - Could calibration be possible over one time frame only?
 - Could calibration work on time frames on the same EPN, but separated in time?
 - Evaluation of the time dependence and stability of the calibration
 - Would the calibration need aggregation over time frames?
- Evaluation of the data reduction factor with the smallest risk of data loss possible
- Identification of the calibrations suited to run on FLPs for each detector
- Identification of the calibrations suited to run on EPNs for each detector
- Identification of the calibrations needed only at analysis level (hopefully none, or very few)

3.9 Condition and calibration database

The following specific requirements apply to the calibration database

- Accessibility
 - simple access configuration (e.g. one string)
 - access control (e.g. to limit non-read operations)
- Data access patterns: read, write, update, delete
 - rates to be defined
 - short latency ($< 1s + size * network\ speed$)
 - have a 'validity' oriented query (get 'last' or get 'good' value for this time/run)
 - scalable distribution of the calibration data to the working nodes at runtime
- Content type
 - time series

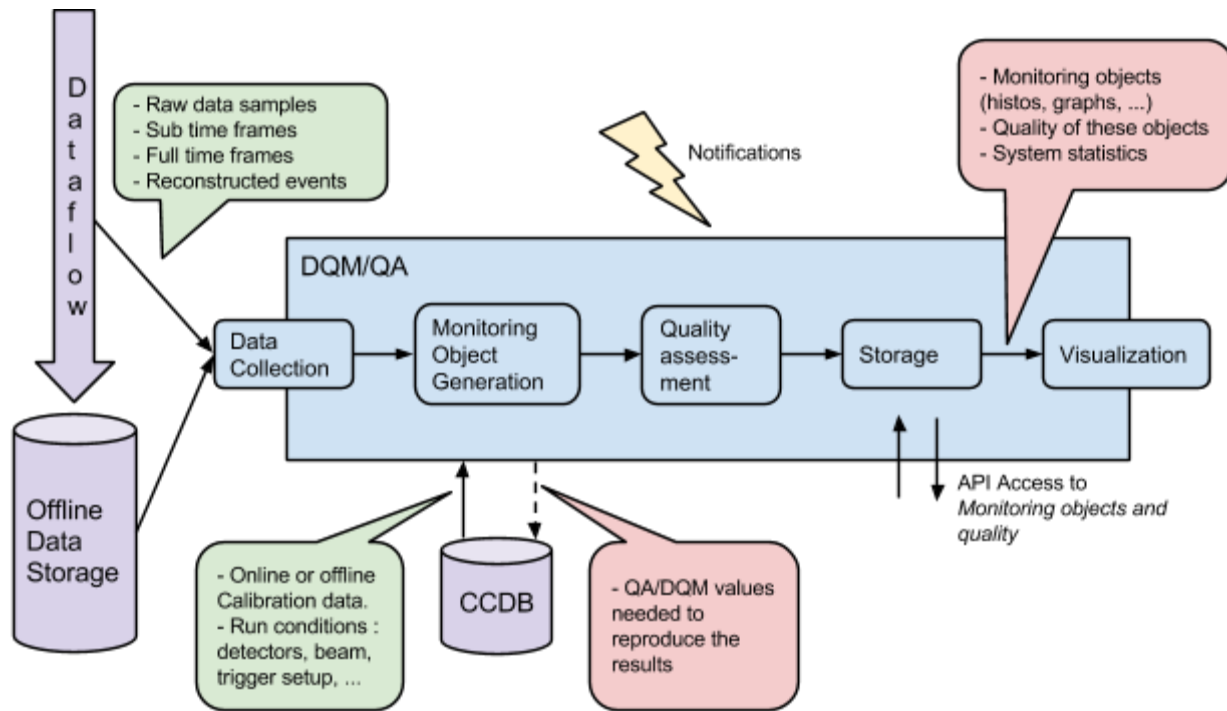
- objects
- metadata (associated to content): timestamp, validity range, run, version, source, access statistics
- Versioning system
- Storage
 - size to be defined, expandable (growth rate to be defined)
 - number of entries/records to be defined
 - data safety / redundancy
- Publish/subscribe mechanism
 - to notify clients of updates
- Scalability
 - db accessible by thousands of distributed processes
 - replication (read-only snapshots and/or continuous streaming)

As the specific rates and sizes for the conditions database are not yet fully defined, the initial reference points are the OCDB constraints of 2012. The OCDB serves roughly 200 CDB objects, accounting for 200 MB per reconstruction job and 150MB per simulation job. The OCDB snapshot size for reconstruction is around 130MB.

- The amount of data per detector is distributed as follows:
 - ITS,TOF,MUON,FMD,T0 ~20 MB
 - TPC ~30 MB
 - TRD ~50 MB
 - PHOS,HMPID,EMCAL,VZERO ~10 MB
 - PMD,ZDC,ACORDE ~1,2 MB
- The biggest object is 50MB, the smallest ones are in the order of a few kB.
- The number of objects per run range from zero (that calibration can rely on the value in the default object) to around ten.
- The overall OCDB size is of order a few tens of GB (e.g. 36GB for the raw OCDB 2012).
- There are typically 1 to 5 versions per object (1 default, 1 created online/offline, and for some objects possibly 3 more for the successive passes). More versions can be created manually when needed.

3.10 Data monitoring (QA/DQM/etc)

Figure 3



Details of the different steps involved in the DQM/QA system and the inputs and outputs the system.
 CCDB: Condition and Calibration DataBase. QA values are stored there rather than in our own storage in order to get the infrastructure for free.

3.10.1 General Requirements

- Data monitoring should not slow down the data taking or have an impact on the data flow, unless intended (e.g. for debugging or during commissioning).
- Online, the system must allow the monitoring of enough data in order to produce meaningful physics results. 10% of the whole data seems more than enough for most of the cases, except for rare triggers. The latter point can be addressed by a smart events selection (see 3.10.2 Input Requirements).
- The same software framework will be used for both offline and online data monitoring with specialized plug-ins for offline/online-specific tasks.
- The system will run at different stages of the data flow (before FLPs, after FLPs, during EPNs, after EPNs) and thus on different types of data (see 3.10.2 below for details).
- Usages
 - *Online*
 During data taking, inside the O² infrastructure, the analyses performed are fast enough to run in a synchronous way. The time available depends on the stage (see bullet directly above). In addition, changes can be made to the percentage of data being analysed.
 - *Offline*
 After data taking inside the O² infrastructure, events that were not treated before can now be analysed. Moreover, extended QA algorithms that couldn't be performed online, should be run at this stage.

- *Test / Debug*
Outside the O² infrastructure, at any moment, and most probably on a standalone machine or small scale infrastructure, developers and users may want to develop, test and debug the system (framework and QA algorithms). This requires the whole system to be installable and executable on a pc.
- Calibration
 - Immediate access to new calibration values.
 - In order to have reproducible QA results certain parameters should be stored. The CCDB (Conditions and Calibration DataBase) seems the right place for such data. It should be possible to update these values later on.
- Architecture - Decoupling
 - Plugin-based: avoid dependency from the framework toward the users' algorithms at the level of the code. Also isolate the code of each detector or user. This would avoid one "faulty" task affecting all the others or the general framework. Different release paces for the framework and for the different algorithms should be defined.
 - Decouple the different tasks at execution time.
 - Exception handling

3.10.2 Input Requirements

- Data samples collection
 - Must be available at all processing stages
 - before FLPs: raw data samples
 - after FLPs: sub time frames
 - during EPNs: full time frames
 - after EPNs: fully reconstructed events (or at least tracks)
 - locally from file (e.g. for testing or replaying): data in one of the above format
 - Should provide enough data to assess the data and processing quality at a given stage without impacting the data taking. 10% seems a reasonable early guess.
 - Should give a way to request a set proportion of data for certain algorithms (with possible impact on the data taking) : all events, a fixed fraction, some events.
 - Should provide a representative/realistic fraction of the whole data. Without any filter, the user gets the same proportion of whatever metrics as compared to the whole data set.

Should be filtered by:

- Trigger classes, clusters, detectors, header bit mask, event size, event type
- Global physical observables: vertex, multiplicity, centrality. These criteria are available at different stages. It imposes an event tagging online as early as possible. Feasibility depends on time available.

3.10.3 Output Requirements

Results of the QA processing are made up of

- An object
 - The object can range from simple plots to fully custom objects and including values and ROOT core objects.
 - Objects shall be mergeable to fully use the framework.
 - Merging is time-consuming and non-trivial, thus limitations on the type might be decided within the framework or with policies. They might differ depending on the stage.
 - Custom objects are useful for detector-specific QA and visualization, especially during commissioning.
- An associated quality
 - It requires automatic evaluation using custom checks and/or predefined checks provided by the framework.
 - The quality should be expressed by a limited set of discrete flags, for example: Undefined, Ignored, Good, Bad, Warning.
 - If possible, a distinction between technical errors and data/content-related issues should be provided.
- Metadata
 - Trivial information should be added to the output in order to record timestamp, run number and task associated to the result.
 - A clear identifier should be added to specify the type of processing (FLP, EPN, offline), the monitored component (calibration, reconstruction, ...) and the data sample ID (BCID/orbit, run, ...).
 - Information to distinguish between technical and content-related issue should be added.

Other requirements about the results :

- There will be a limit on the number of objects created (globally, by detector and by task)
- Results will be *linked* to express a dependency relationship, which could be part of the results' metadata. For example, if result A has a dependency B of a bad quality, one might want to skip A.
- Time-related aspects
 - Keep a history of the objects and their associated quality. For a given run (or fill), one should be able to retrieve all QA objects and their quality as they existed at the end of the run (or fill).
 - Trending of the quality of any object, over run or fill.
 - Trending of the objects will also be possible, at least for those that are not too complex. Trending often implies one or several transformations of the data beforehand (e.g. fit or mean).
 - For long runs it might be useful to have *snapshots* every X minutes or hours to allow inter-run trending. These *snapshots* might not be kept forever as opposed to the history linked to runs and fills.
 - Allow correlation between objects, even from different detectors (a unified data format for trending would allow correlation between different QAs and metrics).

- Trending and correlations can be done later on, or even on demand.
- The format of the result (object+analysis+metadata) must allow an incremental approach: The results of the online monitoring (fraction of all data and limited analysis) will be complemented by the results obtained during the offline monitoring (remaining events and extended analysis).

3.10.4 Storage Requirements

- Centralized results repository for online and offline QA.
- Not used when doing debug and test (inside and outside O² infrastructure)
- Provide a direct API to storage in order to possibly input extra objects or update records, but more importantly in order to retrieve results and correlate them with external data.

