Chapter 6

Trigger, online and offline computing

6.1 Trigger system

6.1.1 Design considerations

The ALICE Central Trigger Processor (CTP) [17, 196–198] is designed to select events having a variety of different features at rates which can be scaled down to suit physics requirements and the restrictions imposed by the bandwidth of the Data Acquisition (DAQ) system, see section 6.2, and the High-Level Trigger (HLT), see section 6.3. The challenge for the ALICE trigger is to make optimum use of the component detectors, which are busy for widely different periods following a valid trigger, and to perform trigger selections in a way which is optimised for several different running modes: ion (Pb-Pb and several lighter species), pA,and pp, varying by almost two orders of magnitude in counting rate.

The first response from the trigger system has to be fast to suit the detector requirements. The principal design requirement for the tracking detectors is to be able to cope with the large multiplicities in Pb-Pb collisions (interaction rate 8 kHz at $\mathscr{L} = 10^{30}$ cm^{-2}s^{-1}). In some cases this has led to the use of non-pipelined 'track and hold' electronics (e.g. for those detectors using the GASSIPLEX [124] front-end chip) and these can require a strobe at 1.2 μs. To achieve this, the 'fast' part of the trigger is split into two levels: a Level 0 (L0) signal, which reaches detectors at 1.2 μs, but which is too fast to receive all the trigger inputs, and a Level 1 (L1) signal arriving at 6.5 μs, which picks up all remaining fast inputs. Note that the CTP decisions are made in 100 ns, with the rest of the L0 latency coming from the generation time for the trigger input signals and from the cable delays.

Another feature of the ALICE environment is that the high multiplicities of Pb-Pb collisions make events containing more than one central collision unreconstructable. For this reason, 'past-future protection' (see below) is an important part of the ALICE trigger. A final level of the trigger (Level 2, L2) waits for the end of the past-future protection interval (88 μs) to verify that the event can be taken. This interval can also be used for running trigger algorithms.

The CTP consists of seven different types of 6U VME boards housed in a single VME crate. The signals are distributed to the detectors using the Local Trigger Unit (LTU). Transmission of trigger signals to each detector is mediated by one of these boards. In addition, the boards can be decoupled from the CTP and used to emulate CTP signals for testing purposes. Emulation of

trigger sequences using the LTU is an important part of the testing and commissioning of detector electronics.

6.1.2 Trigger logic

The number of trigger inputs and classes required for ALICE (24 L0 inputs, 24 L1 inputs, 12 L2 inputs, 50 trigger classes — see below) means that the trigger logic requires some restrictions, since a simple enumeration of all outcomes (look-up table approach) is not feasible. An investigation of the trigger conditions actually required for ALICE shows that these can be accommodated by logic involving three states of the inputs (asserted, negated, not relevant), coupled by ANDs. Negated inputs are rarely used, so they are available for only six out of the fifty trigger classes, the rest allowing the two requirements asserted and not relevant only.

The definition of an interaction, the basis of the minimum bias trigger, needs an OR of different detector inputs, owing to lower trigger efficiencies in pp interactions because of lower multiplicities. In order to do this, four specific L0 inputs are selected for use in a look-up table for which any arbitrary logic can be applied. Two independent functions can be defined. These derived functions can be used as additional conditions with any other L0 input in defining the overall L0 conditions. Note that these same four inputs also define two further independent functions which are used as the 'interaction' signal in the past-future protection circuits.

We define a trigger class in terms of the logical condition demanded for the inputs, the set of detectors required for readout (a maximum of six combinations, called detector clusters, can be defined at any one time), the past-future protection requirements (which are in fact determined by the detector set) and in addition scaling factors, handling of Region-of-Interest (RoI) data and handling of rare triggers. These features are described in more detail below.

Table 6.1 gives a list of trigger inputs for L0 and table 6.2 those for L1. The suffices 'mb', 'sc' and 'ce' refer to the centrality conditions for minimum bias, semi-central and central events respectively. In table 6.1 the 'ZDC diss. single arm' trigger input is one which gives a rough indication of an electromagnetic dissociation signal, based on a single arm of the ZDC. It is confirmed by the 'ZDC diss' signal in table 6.2, which is much more selective. TOF provides two cosmic ray trigger inputs, one for a single vertical muon, and one for several muons simultaneously. Inputs relevant for a specific data taking period must therefore be selected using a patch panel. Table 6.3 gives the list of trigger classes for pp interactions at LHC startup.

The trigger inputs for V0, TRD and SPD are not given in detail. For V0 there are four basic signals, corresponding to hits on either side of the interaction region in the time windows for incoming beam-gas and outgoing colliding-beam interactions, the latter classified according to multiplicity. The V0 electronics offers all these signals and certain Boolean combinations of them, making sixteen signals in all. Any five of these can be sent to the CTP using programmable logic.

Note that the 'TRD pre-trigger' input fulfils a different function from the other trigger inputs. It is formed by copies of the T0, V0 and TOF inputs sent in parallel to the TRD, where they serve to 'wake-up' the TRD electronics, which otherwise is in a standby state. Only if the TRD pre-trigger is sent, the TRD is ready to accept triggers, or to be read out. Therefore, the confirmatory TRD pre-trigger signal (interaction detected AND TRD in ready state) is a prerequisite for any class for which TRD triggers or readout are required. The other TRD L0 inputs in fact reflect

Table 6.1: List of L0 trigger inputs for Pb-Pb and pp collisions. The $*$ indicates when the input is implemented.

No.	Det.	Description	Pb-Pb	pp	No.	Det.	Description	Pb-Pb	pp
1	MTR	DM like high p_t	$*$	$*$	21	TOF	High Multiplicity		$*$
2		DM unlike high p_t	$*$	$*$	22	V0	V01	$*$	$*$
3		DM like low p_t	$*$	$*$	23		V02	$*$	$*$
4		DM unlike low p_t	$*$	$*$	24		V03	$*$	$*$
5		DM single	$*$	$*$	25		V04	$*$	$*$
6	T0	T0 vertex	$*$	$*$	26		V05	$*$	$*$
7		T0C		$*$	27	TRD	Pretrigger	$*$	$*$
8		T0A		$*$	28		TRD2		
9		T0 sc	$*$		29		TRD3		
10		T0 ce	$*$		30		TRD4		
11	PHO	PHOS signal	$*$	$*$	31		TRD5		
12	ZDC	ZDC diss. single arm	$*$		32	SPD	SPD MB		$*$
13	EMC	EMCAL single shower	$*$	$*$	33		SPD2		
14	ACO	ACORDE single muon	$*$	$*$	34		SPD3		
15		multi-muon	$*$	$*$	35		SPD4		
16	TOF	cosmic vert. muon	$*$	$*$	36		SPD5		
17		cosmic multi-muon	$*$	$*$	37		SPD6		
18		MB		$*$	38		SPD7		
19		ultraperipheral (ρ)		$*$	39		SPD8		
20		ultraperipheral (J/Ψ)		$*$	40		SPD9		
					41		SPD10		

different requirements on the pre-trigger signals. In particular, different combinations of TOF signals (indicating activity at central rapidity) in combination with no activity in V0 can be used to select double diffractive events in pp interactions, while hits in T0 and V0 can be used to signal 'normal' events. The TRD also offers trigger signals at L1, which are 'true' TRD inputs in that they derive from the signals in the TRD detector. The exact allocation of trigger logic will be based on charged-track p_t cuts and electron identification in combination with multiplicity requirements (e.g. like sign high-p_t dielectron pair, four or more high-p_t tracks, identified electron, etc.).

The pixel detector (SPD) also provides trigger signals at L0 level, though its latency does not allow its use in the TRD pre-trigger. Several inputs are reserved for SPD triggers, where, in addition to simple multiplicity requirements, other signals, such as triggers with tracks directed towards the HMPID. These can be set up in a programmable way to run in parallel with the main multiplicity-based trigger selections.

Table 6.3 gives an example of a set of possible trigger class conditions (called trigger descriptors). The signal names 'V0or', 'BEAMGASor' and 'BEAMGASc' correspond to possible settings of the five V0 signals; 'V0or' means a hit in the appropriate time window on either side

Table 6.2: List of L1 trigger inputs for Pb-Pb and pp collisions.The $*$ indicates when the input is implemented.

No.	Det.	Description	Pb-Pb	pp
1	PHO	PHOS low E_t	$*$	$*$
2		PHOS med E_t	$*$	$*$
3		PHOS high E_t	$*$	$*$
4	TOF	jet trigger		$*$
5	TRD	TRD1	$*$	$*$
6		TRD2	$*$	$*$
7		TRD3	$*$	$*$
8		TRD4	$*$	$*$
9		TRD5	$*$	$*$
10		TRD6	$*$	$*$
11	ZDC	ZDC mb	$*$	$*$
12		ZDC sc	$*$	$*$
13		ZDC ce	$*$	$*$
14		ZDC diss.	$*$	
15	EMC	EMCAL single shower low	$*$	$*$
16		EMCAL single shower med	$*$	$*$
17		EMCAL single shower high	$*$	$*$
18		EMCAL jet	$*$	$*$

of V0, and 'BEAMGASor', 'BEAMGASc' to signals in the time windows for beam gas on either side (suffix 'or') or coming from the C-side (suffix 'c').

Table 6.3: List of trigger classes with trigger conditions. The symbols $\oplus$ and $\odot$ indicate OR and AND respectively.

Number	Description	Condition
1	MB1	V0or $\oplus$ PIXELor $\odot$ $\overline{\text{BEAMGASor}}$
2	MB2	V0or $\odot$ PIXELor $\odot$ $\overline{\text{BEAMGASor}}$
3	MB3	V0and $\odot$ PIXELor $\odot$ $\overline{\text{BEAMGASor}}$
4	BG	BEAMGASor
5	BGDM	BEAMGASc
6	DMsingle	DMsingle
7	DMBeamgas	BGc $\odot$ DMsingle
8	DML	DMLikeLow
9	DMU	DMUnlikeLow
10	DMMBSingle	MB1 $\odot$ DMSingle
11	DMMBLike	MB1 $\odot$ DMLikeLow
12	DMMBUnlike	MB1 $\odot$ DMUnlikeLow
13	Photon1	MB1 $\odot$ L0PHOSMB
14	Photon2	MB1 $\odot$ L0PHOSLE
15	Photon3	MB1 $\odot$ L0PHOSHE
16	Photon4	MB1 $\odot$ L1PHOSIP

6.1.3 Trigger inputs and classes

The trigger inputs and classes were introduced in the previous section. Trigger inputs are pulses provided by the trigger detectors and synchronised to the LHC clock cycle, as distributed by the RD12 Trigger Timing and Control (TTC) system [199, 200].

Trigger inputs are sent as LVDS signals using twisted pair cables, and are put in time by delays in the CTP input circuits. The trigger outputs are returned to the detectors using either similar twisted pair cables or signals sent on the TTC system for L0, and the TTC system only (using optical fibre) for the other levels. Trigger data are transmitted with each successful L1 and L2 trigger, as described in the next section.

The purpose of the past-future protection circuit is to ensure that the events selected for readout are not spoilt by pile-up. As pile-up will be frequent, or, in some cases, inevitable at expected LHC luminosities (e.g. for $\mathscr{L} = 10^{30}$ cm^{-2}s^{-1} in pp collisions), the pile-up protection must do more than simply reject events if any other event occurs within a specified time window. In Pb-Pb collisions, the circuit makes use of a classification of events into two groups, peripheral and semi-central, and specifies separate limits for each. The time windows are set to be equal to the sensitive windows for the detectors, i.e. the (nominal) maximum time over which two signals from two different events could overlap. Thus, for instance, the sensitive window for the TPC is $\pm 88\,\mu$s, so for clusters in which the TPC is included a possible past-future protection specification, for example, to demand no more than 2 additional peripheral events and no additional semi-central events to take place in an interval of twice $88\,\mu$s centred on the event under consideration.

A slightly different approach is required for proton-proton interactions, where the increased luminosity makes pile-up a certainty, but where the multiplicities are much lower than in ion-ion interactions, and therefore some degree of pile-up is tolerable. Here, what is important is to ensure that all detectors in a cluster read out in a satisfactory manner. For example, pile-up in the ITS is more serious for reconstruction than pile-up in the TPC. For this reason, in this case the past-future protection does not select on multiplicity, but offers checks in two different time windows (e.g. $\pm 10\,\mu$s for the ITS and $\pm 88\,\mu$s for the TPC) so as to ensure that pile-up is not excessive in either.

In general, the input data handled by the ALICE CTP is global, in the sense that the CTP does not correlate specific geometrical regions in different detectors. However, for certain applications it might not be necessary to record data from all regions of the detector, but only from certain azimuthal sectors, with an obvious saving in the overall data volume. The boundaries between these different azimuthal sectors, which define 'Regions-of-Interest' (RoI), line up in the larger central detectors TPC, TRD and TOF, and equivalent boundaries could be imposed in software in the ITS. If a given trigger detector can identify an azimuthal sector as being the one carrying the information giving rise to the trigger (e.g. the presence of a high-p_{t} electron), it is foreseen that a flag can be set identifying the sectors of interest. These can then be treated in a special way, e.g. by selecting only those sectors for readout to DAQ, or by treating them differently in the HLT.

6.1.4 Trigger data

The CTP must be able to process triggers for all trigger clusters concurrently, and therefore has a tight time budget for collecting and distributing data. For this reason these data are kept to a minimum; for example, all information concerning how a trigger signal was generated must be

obtained from the detector which produces it. Several types of data are recorded concerning the operation of the trigger, as listed below.

Event data. For each accepted event, information is sent to each detector specifying the event identifier (orbit number and bunch crossing number), trigger type (physics trigger, software trigger, calibration trigger), detector set to be read out (given as a cluster list for physics data) and the list of active trigger classes. These data are sent in the L2 accept (L2a) message, which goes to every TTCrx (TTC receiver chip) on each detector.

Interaction record. All interactions that occur in a given orbit are recorded (giving bunch crossing number, and, for Pb-Pb events, a classification into centrality categories: peripheral and semi-central). The principal purpose of this record is to aid pile-up correction. Out-of-time events recorded in the TPC will have different apparent origins from triggered events owing to their different drift times. Their apparent origin can be computed using the interaction record, thus aiding their removal from the event data. The data are sent as special records in the data stream, and are collected in the trigger Local Data Concentrator (LDC).

Scalers. A large number of scalers are recorded. In particular, all inputs are counted, and for each trigger class the number of events passing each stage of the trigger (L0, L1, L2, after physics selections, if any, and again after the corresponding BUSY and past-future protection constraints have been applied.) These data are essential both for on-line monitoring and for subsequent off-fline luminosity and cross-section calculations. Scalers are read at regular intervals (currently every minute) and their values are broadcast using a Distributed Information Management system (DIM) server [201]. The values are recorded in the 'ProzeßVisualisierungs und SteuerungsSystem' (PVSS) [202], where ratios can be monitored to check the correct operation of the trigger. At the end of a run, a file containing both the individual scaler readings and their sum for the run just ended is transmitted from PVSS to DAQ. Notice that the use of a DIM server to transmit from the CTP to PVSS opens the possibility of defining other DIM clients, should this be required.

Snapshots. For completeness, we note that it will also be possible to run 'snapshots' in which all steps in the trigger (input and output patterns, busy status, etc.) are recorded on a bunch-crossing by bunch-crossing basis for a period of about 30 ms. The purpose originally foreseen for this facility is for diagnostic tests of the CTP itself. Snapshots can also be used for checks of correlations between CTP inputs, since an unbiased record of all input data received during the 30 ms snapshot period can be analysed. For example, time correlations between a pulse on one input and pulses in other input channels in adjacent bunch crossings can be studied; trigger efficiencies for different inputs can also be studied.

6.1.5 Event rates and rare events

When several trigger classes are running concurrently, it becomes necessary to adjust the rates at which they are read out to reflect the physics requirements and the overall DAQ bandwidth. These factors may dictate rates quite different from the natural interaction rates.

There are two main mechanisms for controlling readout rates. Downscaling factors may be applied to each trigger class, specifying, for example, that only every *n*th event should be read out. This mechanism allows gross adjustments to be made, but is inflexible because the downscaling factors remain in place throughout a physics run. This means that the change in interaction rate as the luminosity decays following each fill cannot be followed. For this reason, in addition to downscaling factors, a second method is used. Studies of data flow through the front end and DAQ systems show that with the current choices for numbers of front end buffers, saturation of front end data storage can be avoided, but without further action temporary data storage in the DAQ can become saturated, with a relaxation time of the order of seconds. This phenomenon would particularly affect rare processes, as these would find the available bandwidth for data recording utilized by more common processes. To avoid this problem, all trigger classes are classified into two groups: those corresponding to rare processes and those corresponding to common processes. Initially all activated trigger classes can generate triggers. On a signal from the DAQ, sent when the occupied temporary storage exceeds some preset 'high water mark', the common classes are temporarily disabled, thus ensuring continued available bandwidth for rare processes. When the available temporary storage has gone below some corresponding 'low water mark' the common classes are again enabled. Owing to the long relaxation time, timing is not critical, and software signals are adequate for toggling the suppression of common classes. Details of the model used to study the system response, and a description of a simple example, are given in section 6.2.4).

6.2 Data AcQuisition (DAQ) System

6.2.1 Design considerations

ALICE will study a variety of (physics) observables, using different beam conditions. A large number of trigger classes will be used to select and characterise the events. These trigger classes belong to two broad categories depending on whether they are frequent or rare.

Several triggers (central, semi-central and minimum bias) are so frequent that the limiting factor is the performance of the data acquisition system. These triggers will use a very large fraction of the total data acquisition bandwidth. On the other hand, rare triggers such as dimuon or dielectron events, use less bandwidth and are limited by the detector livetime and the luminosity. The experiment will use the data-taking periods in the most efficient way by acquiring data for several observables concurrently following different scenarios. The task of the ALICE Trigger, Data AcQuisition (DAQ) and High-Level Trigger (HLT) systems is therefore to select interesting physics events, to provide an efficient access to these events for the execution of high-level trigger algorithms and finally to archive the data to permanent data storage for later analysis.

The trigger and DAQ systems were designed to give different observables a fair share of the trigger and DAQ resources with respect to DAQ bandwidth for frequent triggers and detector livetime for rare triggers. The trigger and DAQ systems must balance the capacity to record central collisions which generate large events with the ability to acquire the largest possible fraction of rare events.

In the ALICE Technical Proposal [4–7], it was estimated that a bandwidth of 1.25 GB/s to mass storage would provide adequate physics statistics. While the current estimate of event sizes,

trigger rates and, in particular, the number of physics observables (and therefore trigger classes) has increased considerably, it is still possible to satisfy the physics requirements with the same total bandwidth by using a combination of increased trigger selectivity, data compression and partial readout [203]. Detailed scenarios of use of this bandwidth were elaborated in the ALICE Technical Design Report on Trigger, Data Acquisition, High-Level Trigger and Control System [17]. This bandwidth is consistent with constraints imposed by technology, cost and storage capacity, including the cost of the media needed to store the data and the computing needed to reconstruct and analyse these data. The Tier-0 of the LHC Computing Grid project, that is being assembled at CERN, will provide a bandwidth to mass storage higher than the needs of ALICE for data acquisition. However, efforts will be continued to reduce the bandwidth without jeopardising the physics performance.

6.2.2 System architecture

System architecture

The architecture of the data acquisition is shown in figure 6.1. The detectors receive the trigger signals and the associated information from the Central Trigger Processor (CTP), through a dedicated Local Trigger Unit (LTU) interfaced to a Timing, Trigger and Control (TTC) system. The Front-End Read-Out (FERO) electronics of the detectors is interfaced to the ALICE-standard Detector Data Links (DDL). The data produced by the detectors (event fragments) are injected on the DDLs using the same standard protocol. The fact that all the detectors use the DDL is one of the major architectural features of the ALICE DAQ.

At the receiving side of the DDLs there are PCI-X based electronic modules, called 'DAQ Readout Receiver Card' (D-RORC). The D-RORCs are hosted by the front-end machines (commodity PCs), called Local Data Concentrators (LDCs). Each LDC can handle one or more D-RORCs. The D-RORCs perform concurrent and autonomous Direct Memory Access (DMA) transfers into the LDCs' memory, with minimal software intervention. The event fragments originated by the various D-RORCs are logically assembled into sub-events in the LDCs.

The CTP receives a busy signal from each detector. This signal can be generated either in the detector FERO's or from all the D-RORCs of a detector. These two options are respectively shown on the left and right detectors sketched in figure 6.1. The CTP also receives a signal from the DAQ enabling or disabling the most common triggers. It is used to increase the acceptance of rare triggers by reducing the detector dead-time. This signal is generated from the buffer occupancy in all the LDCs.

The role of the LDCs is to ship the sub-events to a farm of machines (also commodity PCs) called Global Data Collectors (GDCs), where the whole events are built (from all the sub-events pertaining to the same trigger). Another major architectural feature of the ALICE DAQ is the event builder, which is based upon an event-building network. The sub-event distribution is driven by the LDCs, which decide the destination of each sub-event. This decision is taken by each LDC independently from the others (no communication between the LDCs is necessary), but it is synchronized among them by a data-driven algorithm, designed to share fairly the load on the GDCs. The Event-Destination Manager (EDM) broadcasts information about the availability of the GDCs to all LDCs. The event-building network does not take part in the decision; it is a standard commu-

2008 JINST 3 S08002

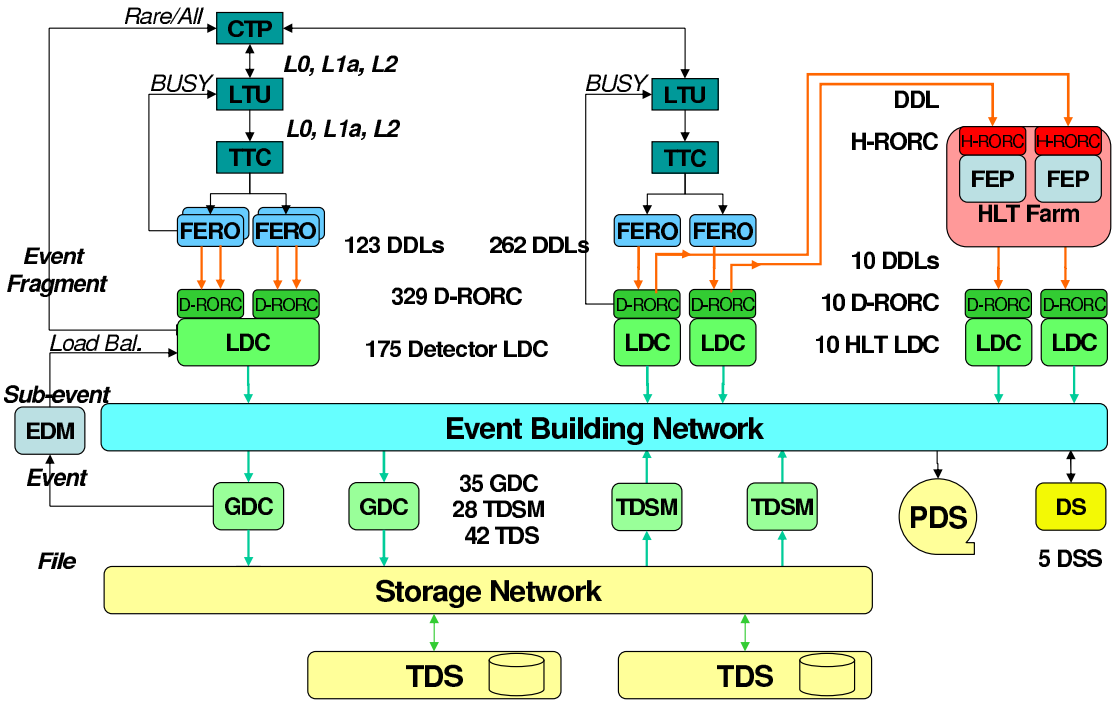

Figure 6.1: The overall architecture of the ALICE DAQ and the interface to the HLT system.

nication network (commodity equipment) supporting the well-established TCP/IP protocol. The role of the GDCs is to collect the sub-events and assemble them into whole events. The GDCs also feed the recording system with the events that eventually end up in Permanent Data Storage (PDS).

The HLT system receives a copy of all relevant raw data via DDLs and the 'HLT Readout Receiver Card' (H-RORC) into the Front-End Processors (FEP). The generated data and decisions are transferred to dedicated LDCs.

Data transfer

The Detector Data Link (DDL) is the common hardware and protocol interface between the front-end electronics and the DAQ system. The transmission medium is a pair of optical fibres linking sites several hundred metres apart. The DDL is able to transmit data in both directions with a sustained bandwidth of 200 MB/s. It can be used to send commands and bulky information (typically pedestals or other calibration constants) to the detectors [204, 205].

The D-RORC is the readout board of the DDL link. It interfaces the DDL to the PCI bus of the host LDC computer. In addition, the D-RORC grants to the LDC full control of all the features of the DDL. The D-RORC technology is based on a FPGA. The card is able to transfer the event fragments from the link directly into the PC memory at 200 MB/s, without on-board buffering; this bandwidth fulfils (and far exceeds) the original requirement of the ALICE data-acquisition system. The Direct Memory Access (DMA) transfer is carried out by the firmware in co-operation with the readout software running on the LDC. In this closely coupled operation, the role of the software is limited to the bookkeeping of the data-block descriptors. This approach

allows sustained autonomous DMA with little CPU assistance and minimal software overhead. The data formatting introduced by the DDL allows the D-RORC to separate the event fragments in memory, without interrupting the host CPU. Performance measurements and long-term tests show that the system fully exploits the PCI bandwidth and that the transfer rate is largely independent of the block size.

The D-RORC includes two DDL interfaces. For the detectors whose data are analysed by the HLT system, one of the DDL interfaces is used to transfer a copy of the raw data to the HLT computers. RORC prototypes were built based on the PCI and PCI-X standards [206, 207]. These prototypes were used to develop the DAQ system to integrate the DAQ with the detector electronics.

Event building

The sub-events prepared by the LDCs are transferred to one GDC where the full event can be assembled. The event building is managed by the Event Building and Distribution System (EBDS). The EBDS distributed protocol runs on all the machines, LDCs and GDCs, participating as data sources or destinations. The goals of the EBDS are to synchronise all the LDCs concerning the GDC used as destination and to balance the loads on the different GDCs. The aim is to keep up with the data-flow while keeping the EBDS protocol overhead as low as possible.

The de-facto standard TCP/IP is used as the transport mechanism for the EBDS protocol and for the data transfer. This protocol is the standard to which the networking industry has converged and it runs on a wide range of hardware layers. Ethernet is the dominant, ubiquitous Local Area Network (LAN) technology. It has therefore become the baseline choice for the event-building network. However, as long as we can rely on a standard protocol such as TCP/IP, the hardware layer used in the event-building network could be modified if a better technology would become available during the experiment lifetime.

Data storage

The GDCs archive the data over the storage network as data files of a fixed size to the Transient Data Storage (TDS). During a run period, each GDC produces a sequence of such files and registers them in the ALICE Grid software (AliEn) [208]. The data files are then read by the TDS movers (TDSM) over the storage network and exported to the computing centre where they are recorded to the Permanent Data Storage (PDS). The TDS is made of storage arrays connected to the storage network. The storage network itself is based on Fibre Channel 4 gigabit/s (FC 4G). The shared access from all the nodes (GDCs and TDSMs) to the TDS is performed through a commerical cluster file system (Quantum StorNext).

DATE: the DAQ software framework

The Data Acquisition and Test Environment (DATE) [209, 210] is the DAQ software framework. It is a distributed, process-oriented system. It is designed to run on Unix platforms connected by an IP-capable network. It uses the standard Unix system tools available for process synchronisation and data transmission. The system configuration is realised with MySQL.

2008 JINST 3 S08002

The dataflow is realized in two steps. The LDC collects event fragments and reassembles them into sub-events. The LDC is also capable of local data recording, if used in stand-alone mode, and online sub-event monitoring. The GDC puts together all the sub-events pertaining to the same physics event, builds the full events and sends them to the mass storage system. The GDC is also capable of online event monitoring.

The DATE controls and synchronises the processes running in the LDCs and the GDCs. It can run on an LDC, a GDC or another computer. The monitoring programs receive data from the LDC or GDC streams. They can be executed on any LDC, GDC or any other machine accessible via the network.

AFFAIR: the DAQ performance monitoring software

The perfomance of a system as large as the ALICE Data Acquisition, including several processes distributed on many processors, needs to be continuously and closely monitored. The fundamental requirement for a detailed, real-time assessment of the DAQ machines (LDCs and GDCs), for the usage of the systems resources, and for the DATE performance is addressed by the AFFAIR package (see p. 217 in [17]). AFFAIR gathers performance metrics from the LDCs and GDCs and performs the centralised handling of them. Statistics and trends are stored in HTML format and can be viewed using any Web browser.

MOOD: the DAQ framework for the Monitoring Of Online Data

To monitor the quality of the data stream created by any of the ALICE detectors, the MOOD [211, 212] toolkit has been developed. MOOD is a data visualisation and data quality monitoring tool. It includes a generic part which implements the interface with DATE and a detector-specific part that can be tailored to detector-specific requirements and setups. MOOD is fully integrated with the ROOT development toolkit, the AliROOT environment, and uses the ALICE common event data format. MOOD can handle online and offline data streams, available on the LDCs and on the GDCs.

AMORE: the DAQ framework for the Automatic MOnitoRing Environment

The data quality monitoring is automatized. Each detector defines a set of physics plots which have to be continuously filled and checked against reference ones. The AMORE framework [213] includes three components: the client part which collects the data, the server part which accumulates the plots and archives them, and the display program which provides an interactive distributed access to the plots archives. In addition, alarms are raised as soon as collected plots do not conform any more to the expected reference. These alarms are displayed on the operator screens and initiate automatic recovery actions.

6.2.3 System flexibility and scalability

The requirements for the DAQ system evolve with data taking and with new ideas of high-level triggers. Currently, the only hard limit present in the design of the DAQ system is the number of DDLs used to read out the TPC and the TRD. The number of links has been fixed so that these two

detectors can be completely read out at a maximum rate of 200 Hz at the highest anticipated multiplicity for central collisions. The next stages in the data flow chain (computers and event-building network) are completely scalable and their number can be adapted to the needs. No architectural change would be needed in the DAQ system to scale to a total throughput varying by a factor 4 or 5. This flexibility also authorise a staged deployment of the system and the implementation of new scenarios on short notice.

6.2.4 Event rates and rare events

The full data flow behaviour is modelled, using an interaction rate of 8 kHz at nominal luminosity and realistic event size distributions of different trigger classes. The model consists of the Central Trigger Processor (CTP), the detectors, the DAQ, the High-Level Trigger (HLT) and the Permanent Data Storage (PDS) [214–216]. Events are characterised by different trigger classes, peripheral (PE), semi-central (SC) or central (CE), according to their multiplicity. In addition, an event can also be of the electron (EL) or muon (MU) trigger class based on an assigned probability. This model is based on the public-domain tool Ptolemy [217].

A rate control mechanism has been included in the DAQ such that no rare events (electron and muon) are lost due to back-pressure, while at the same time as many frequent events (peripheral, semi-central or central) are accepted as possible with the installed bandwidth.

This rate control is based on a back-pressure from the DAQ that detects the amount of data filled in each computer, and informs the trigger whenever this level reaches a high fraction of the buffer size. When this happens, the frequent events, as the main bandwidth contributors and events of lower priority, would then be blocked by the trigger.

Figure 6.2 gives the L2 rates as a function of time when the LDC high level is set to 0.3, the low level is at 0.2, and the downscaling factor for PE, SC and CE events is set at 0.2. The periodic behaviour of the PE, SC, and CE events is apparent, and so is the constant rate of the EL and MU events. The rate control increases the rates of rare events with 22% of EL events (it was 2%) and 78% of MU events (it was 22%) accepted. This is a large performance improvement and is close to the maximal rates possible, after past-future (P/F) rejection. The PE, SC, and CE events fill up the remaining bandwidth to the PDS.

Figure 6.3 shows the LDC occupancy as a function of time, from which it is clear that the LDCs are never full, and hence there is no loss due to LDC back-pressure. Figure 6.4 shows the fraction of time a detector is busy. The events lost due to busy detectors are only caused by detector readout dead times and by transfer limitations through the detector buffer chain. These are hardware properties of the detector and cannot be influenced by the DAQ or CTP.

In summary, an effective and robust method to maximise the L2 rates of rare and interesting events can be achieved by controlling the rates of peripheral, semi-peripheral and central events with feedback from the LDC occupancy and with downscaling. No difficulty in either messaging the information from the LDCs, or the stability of the system with different interaction rates or envisioned physics is anticipated.

6.2.5 Data challenges

The anticipated performances of the ALICE DAQ were verified one year before data taking.

2008 JINST 3 S08002

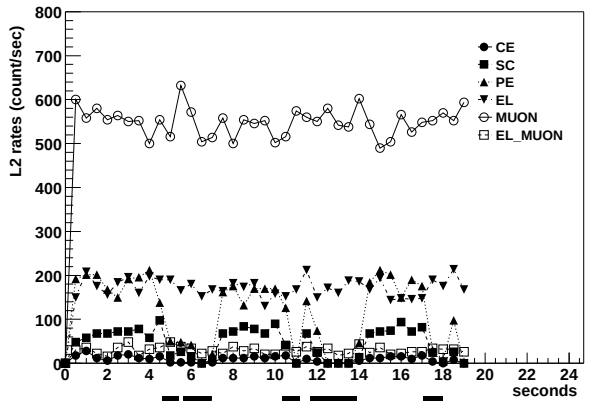

Figure 6.2: L2 rates as a function of time. The bold lines under the plot indicate the periods of activation of rare triggers only.

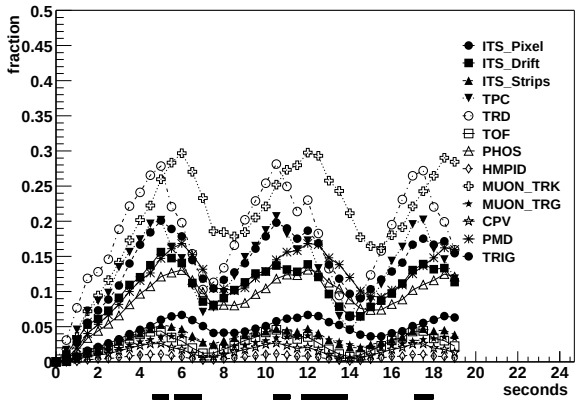

Figure 6.3: Fraction of the LDC buffer filled as a function of time. The bold lines under the plot indicate the periods of activation of rare triggers only.

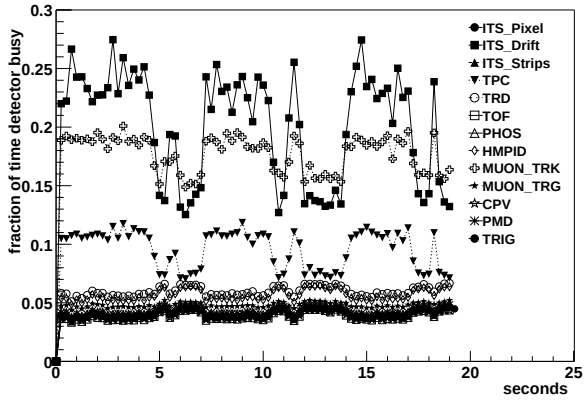

Figure 6.4: Fraction of time a detector is busy as a function of time. The bold lines under the plot indicate the periods of activation of rare triggers only.

The two curves of figure 6.5 show respectively the aggregate bandwidth to and from the event-building, and the bandwidth of the data archived to the MSS in the computing centre.

Owing to the unprecedented quantity of data to be stored in the MSS, the volume of data

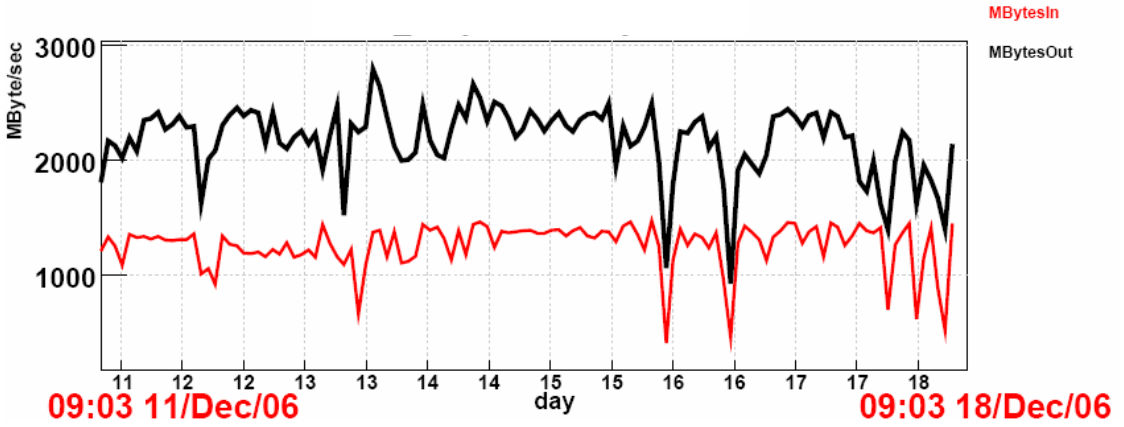

Figure 6.5: ALICE Data Challenge bandwidth results.

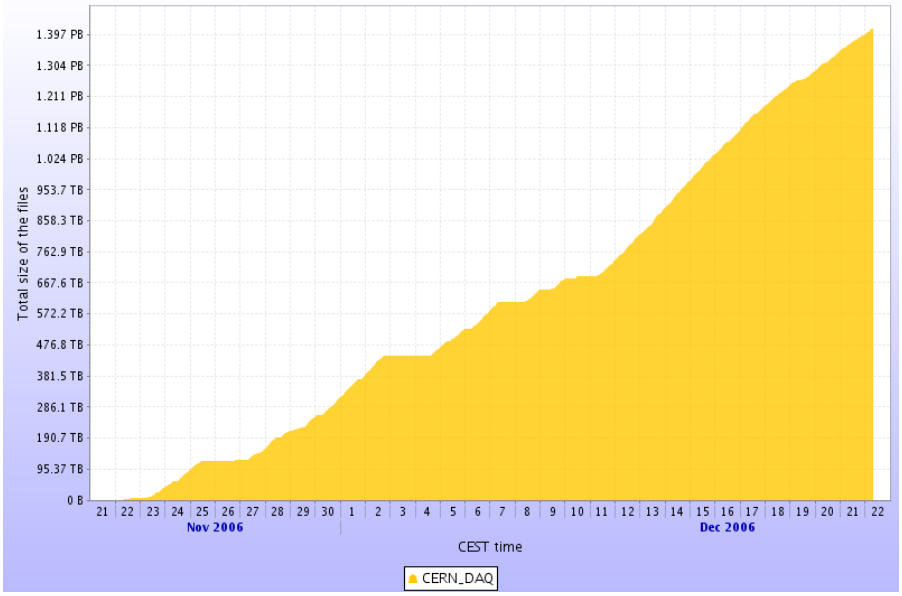

Figure 6.6: ALICE Data Challenges data to Mass Storage planning.

recorded during the Data Challenge have reached a volume equivalent to the total amount of data that the ALICE experiment will archive during a heavy-ion run as shown in figure 6.6.

6.3 High-Level Trigger (HLT)

6.3.1 Introduction

According to the simulation studies [4], the amount of data produced in the TPC alone, in a single central nucleus-nucleus collision, corresponds to about 75 MB assuming $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$ at mid-rapidity. The data rate for all detectors, resulting after a trigger selection, can easily reach 25 GB/s, while the physics content of a large number of events might be small and the DAQ archiving rate is about 1 GB/s. Therefore on-line processing is advised in order to select relevant events or

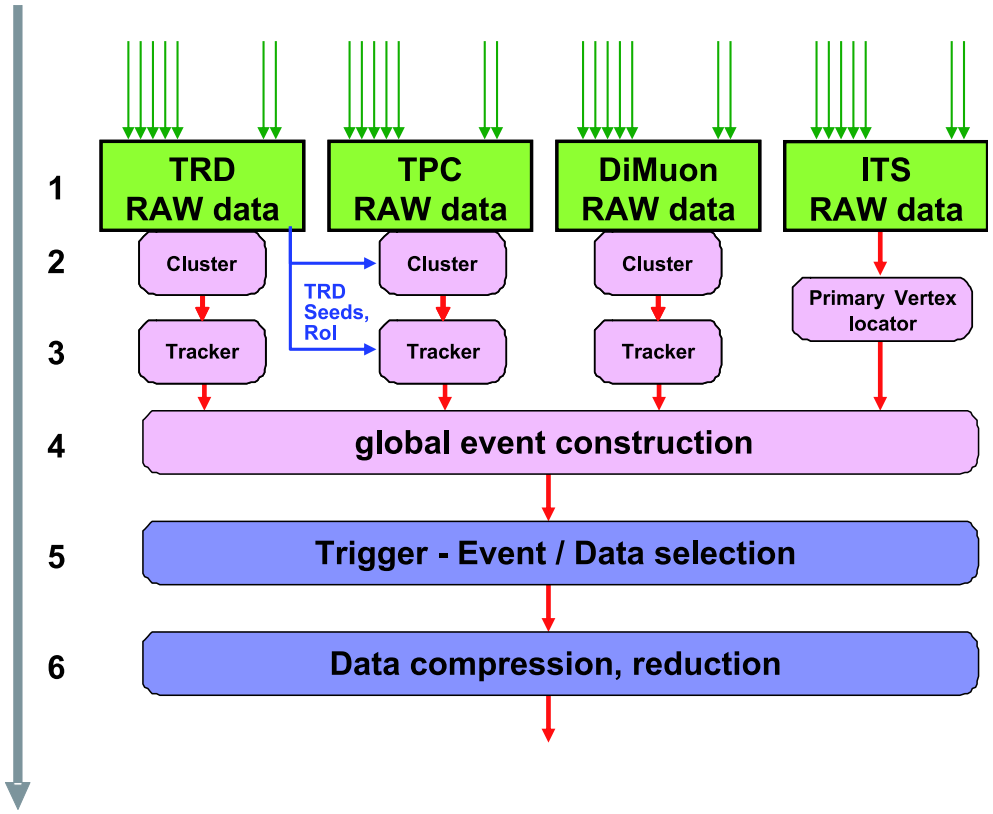

Figure 6.7: The six architectural layers of the HLT.

sub-events and to compress the data without loosing their physics content. The overall physics requirements of the High-Level Trigger [218] are categorised as follows:

Trigger Accept or reject events based on detailed online analysis.

Select Select a physics region of interest within the event by performing only a partial readout.

Compress Reduce the event size without loss of physics information by applying compression algorithms on the accepted and selected data.

6.3.2 Architecture

The HLT implements a processing hierarchy as outlined in figure 6.7. The raw data of all ALICE detectors are received via 454 Detector Data Links (DDLs) at layer 1. The first processing layer performs basic calibration and extracts hits and clusters (layer 2). This is done in part with hardware coprocessors (see section 6.3.2.2) and therefore simultaneously with the receiving of the data. The third layer reconstructs the event for each detector individually. Layer 4 combines the processed and calibrated information of all detectors and reconstructs the whole event. Using the reconstructed physics observables layer 5 performs the selection of events or regions of interest, based on run specific physics selection criteria. The selected data is further subjected to complex data compression algorithms.

In order to meet the high computing demands, the HLT consists of a PC farm of up to 1000 multi-processor computers. The data processing is carried out by individual software components running in parallel on the nodes of the computing cluster. In order to keep inter-node network traffic to a minimum and for the means of parallelisation, the HLT data processing follows the natural hierarchical structure. Local data processing of raw data is performed directly on the Front-End Processors (FEPs), hosting the H-RORCs. Global data processing, with already reduced data, is done on the compute nodes. The trigger decision, Event Summary Data (ESD) of reconstructed events and compressed data are transferred back to the DAQ via the HLT output DDLs.

6.3.2.1 Physical cluster layout

The HLT physical cluster layout consists of 51 19" racks located in two counting rooms (CRs) at the ALICE experimental area at Point2 at CERN. The first installation stage includes 81 FEP nodes and 15 infrastructure nodes, each equipped with one CHARM (Computer Health and Remote Management) PCI card for Cluster Monitoring.

All the nodes are off-the-shelf PCs with two AMD Dualcore Opteron 2 GHz CPUs, 8 GB RAM and two Gigabit Ethernet connections. Each FEP contains two H-RORCs with two DDLs receiving the detector data. The infrastructure nodes consist of four server nodes, two gateway nodes and 9 portal nodes. The gateway nodes connect the cluster to the CERN network. Two server nodes are used as monitoring servers for the Cluster Management Framework SysMES (see section 6.3.3) and two as AFS file-servers. Each file-server implements a net storage capacity of 2 TB on a hardware Raid 6. The portal nodes are used for connections to various ALICE subsystems (see section 6.3.5). In addition, three multipurpose portal nodes and two nodes of every type were installed for redundancy reasons. All the nodes are connected via 1-to-1 Gigabit Ethernet switches, while the FEPs use two Gigabit Ethernet ports. The infrastructure nodes and racks, housing the FEPs and compute nodes, are interconnected via an InfiniBand backbone network.

All ALICE detectors are connected to the HLT, following the read-out granularity (number of DDLs) of each sub-detector: TPC 216, TRD 18, PHOS 20, DiMUON 23, ITS (SPD and SSD) 36, EMCAL 24, HMPID 20, FMD 3, and ACORDE, Trigger, ZDC, T0 and V0 1 DDL each. 12 DDLs transfer the results of the HLT to DAQ.

6.3.2.2 HLT-Readout Receiver Card (H-RORC)

The H-RORC (figure 6.8) is a Virtex-4 FPGA based PCI card designed for both (i) receiving and pre-processing of the detector RAW data of all ALICE detectors and (ii) transmitting processed events out of the HLT computing farm to the DAQ. The H-RORC therefore implements the interface between the HLT system and the ALICE data transport links DDL.

In order to receive these data, the H-RORC is able to host two Destination Interface Units (DIUs). Each link has a maximum bandwidth of 160 MB/s giving a maximal total incoming data rate of 320 MB/s. Incoming data can be transferred directly into the main memory of the Front-End Processors via PCI DMA burst transfers with a rate of up to 370 MB/s per H-RORC. The FEPs are capable of handling two H-RORCs with two DDL interfaces each, running at full speed. In order to send processed events out of the HLT, the H-RORC can host up to two Source Interface Units (SIUs) to send data via DDL fibres to the DAQ. All transactions are PCI DMA based.

2008 JINST 3 S08002

Figure 6.8: H-RORC data pump and FPGA coprocessor.

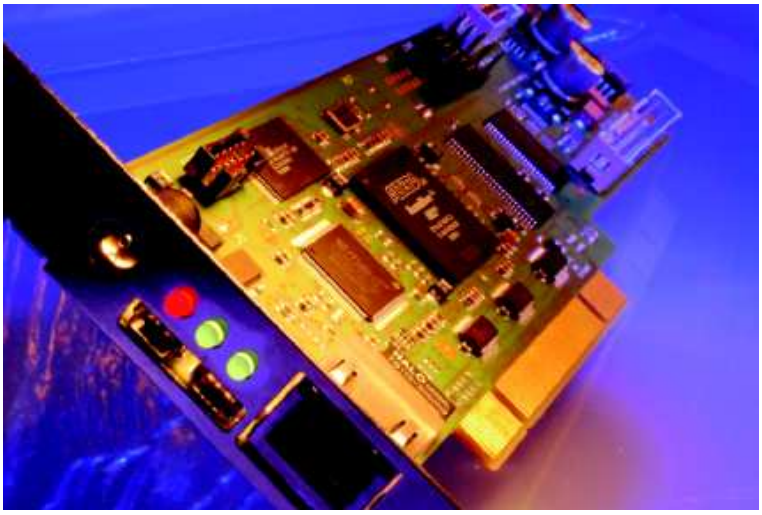

Figure 6.9: CHARM remote management card.

Data can also be pre-processed inside the FPGA [219]. This is supported by four independent Double Data Rate Synchronous Dynamic Random Access Memory (DDR-SDRAM) modules of 32 MB, each allowing data storage with up to 2.3 GB/s, and two fast full-duplex serial links connect each H-RORC to its neighbours. There are two on-line-processing modules being prepared, the TPC and the DiMuon Cluster Finder.

6.3.3 Cluster management

The HLT Cluster is managed using the SysMES Framework (System Management for Networked Embedded Systems and Clusters) and Lemon (LHC Era Monitoring). SysMES is a scalable, decentralised, fault tolerant, dynamic and rule-based tool set for the monitoring of networks of target systems based on industry standards. The resources to be monitored are the applications running on the computer nodes as well as their hardware, the network switches and the RMS (Rack Moni-

toring System). In addition each computer implements a remote administration sensor and actuator (CHARM card figure 6.9), allowing full remote control of the individual node, also implementing autonomous error detection and correction functionality.

The system monitoring data is visualized by Lemon servers and the SysMES servers will react accordingly. The data transportation framework is designed to handle dynamic re-routing of the data flow during run-time. This is utilised to keep the analysis chains running even if severe problems (like hardware failure) occur, such that no data loss or damage of the online analysis will happen. Therefore an intermediate layer between SysMES and the PubSub framework (see section 6.3.4.1) is being developed, consisting of two applications, the Configuration Manager and the Resource Manager.

6.3.4 Software architecture

The HLT software is divided into two functional parts, the data transportation framework and the data analysis. The first part is mainly of technical nature, it contains the communication of the components, steering and data transfer between components/nodes. The second part contains the physics analysis. Development of Analysis Components is independent from data transportation which allows for a high degree of flexibility. The Analysis Components can be run in the offline analysis without changes, making a direct comparison of results possible.

6.3.4.1 Data transportation and Publisher-Subscriber framework

Based on the Publisher-Subscriber (PubSub) principle, a generic data transportation and steering framework was developed, the HLT PubSub framework [220–223]. The PubSub framework carries out the task of communication, data transportation and load distribution within the HLT. Figure 6.10 illustrates key principles and components of the PubSub framework.

Communication takes place in a pipe-lined data flow push architecture. Each component receives data, processes it and forwards new output data to the next component in the chain. The entire communication mechanism is designed for a low processing overhead, primarily through the use of shared memory and the exchange of descriptors. Figure 6.10 also depicts key components of the Publisher Subscriber Framework used to shape the data-flow in the system. An event merger component can merge data streams consisting of data blocks of the same event into one stream with multiple blocks per event. Communication between different nodes is supported by dedicated network bridge components. These connect to other components using the common interface and tunnel data over the network. The scatterer and gatherer pair of components provide capabilities for load balancing and distribution among multiple nodes or one multiprocessor node having multiple physical and/or logical processors. The scatterer splits a single data stream consisting of multiple events into several data streams, each transporting a correspondingly smaller number of events. Each event is forwarded by the scatterer unchanged. On the other end the gatherer collects those streams again into a single stream. An interface that allows access to data at any point in a processing chain, e.g. for monitoring, is described in section 6.3.5.4.

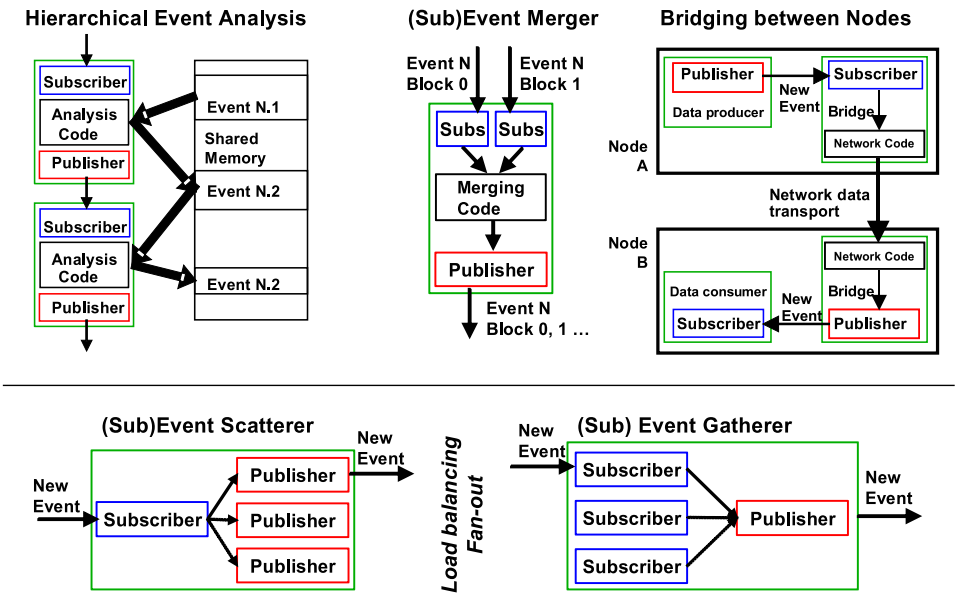

Figure 6.10: Illustration of key principles and components of the HLT Publisher Subscriber framework.

6.3.4.2 HLT analysis framework

The HLT analysis framework [224] encapsulates the data and physics analysis and separates it from the data transportation. A class hierarchy hides the complex interface logic from the derived component classes implementing the actual physics analysis algorithms. The framework integrates the HLT analysis components into the offline AliRoot reconstruction and models a behaviour similar to the PubSub Framework for the components. A C interface layer provides access to the analysis components for external applications, which include a PubSub interface wrapper program as well as a simple standalone run environment for the components.

The online and offline environment are two different running modes of the HLT analysis. The first is the main running mode, while the latter one is frequently used for the evaluation of the HLT algorithms and analysis. Therefore, binary compatibility is considered to be a necessary feature of the system. The same shared library has to run in AliRoot and PubSub. In the online HLT, a special processing component, the AliRootWrapperSubscriber, can load an analysis module and instantiate the actual analysis component via the external C interface and implements all the other online functionality which is necessary to plug the analysis code into a HLT processing chain. As a result, the component libraries compiled in the AliRoot framework can directly be loaded and used from the PubSub system.

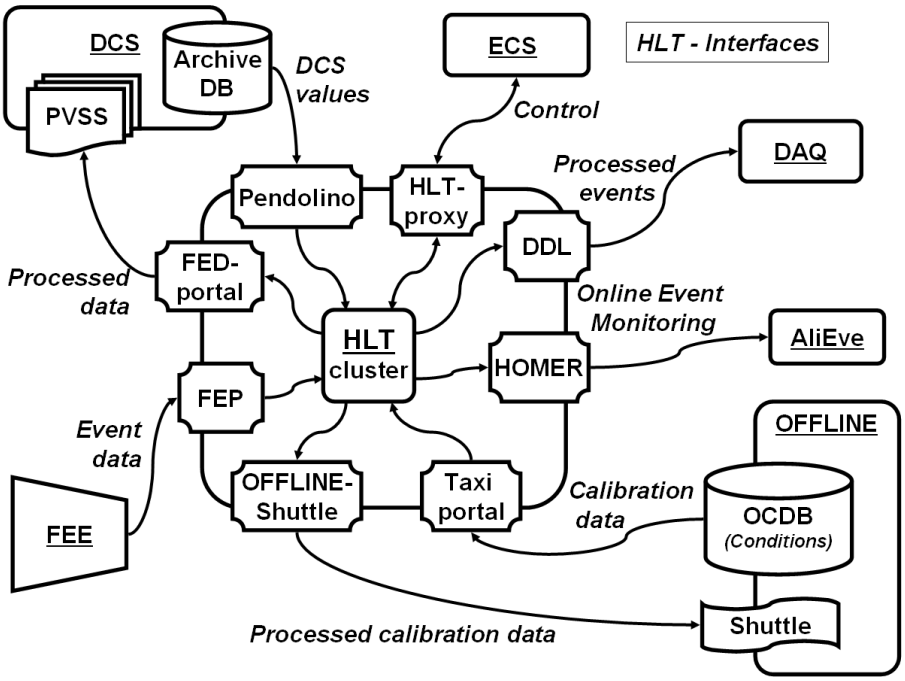

Figure 6.11: HLT interfaces to the other systems of ALICE.

6.3.5 HLT interfaces to other online systems and offline

The HLT communicates with the other ALICE systems through various external interfaces and portal nodes (figure 6.11, [225]). Fail safety and redundancy in the design of these interfaces will avoid single points of failure and reduces the risk of time delays or data loss.

6.3.5.1 Offline interface

The Offline Conditions Data Base (OCDB) stores all produced calibration and condition settings. A special application, the Taxi, requests new calibration data in regular time intervals from the OCDB and synchronises them with the local HLT copy of the database which is available on all processing nodes and allows Detector Algorithms to access the calibration data of previous runs in the same way as in the offline environment. After each run, updated calibration data is shipped back to the OCDB via a portal. A special PubSub data sink component collects the produced calibration data inside the HLT cluster and saves them to the File Exchange Server (FXS). A MySQL DB is used for storing additional meta data. The offline Shuttle will contact both after each run to fetch the updated data.

6.3.5.2 Detector Control System (DCS) interface

Permanently monitored values like temperatures are requested in regular time intervals by the Pendolino software from the DCS Archive DB through the DCS interface during the run. These fetched values are then provided to the Detector Algorithms on the HLT cluster nodes. Vice versa, the HLT writes back data to the DCS via the Front-End-Device Application Programming Interface

(FedAPI), which is common among all detectors. It consists of a DIM-server (Distributed Information Management) on the HLT-side, and a DIM-client, embedded in the PVSS (Process Visualisation and Steering System) of the DCS. The server publishes monitored data and the client includes them into the DCS.

6.3.5.3 Experiment Control System (ECS) interface

All the interactions with the other systems are governed by states and state transitions issued from the ECS. For this purpose a finite state machine runs on the corresponding interface portal. It accepts transition commands and returns the current state of the HLT through an HLT proxy interface software. In addition, ECS hands in running conditions like run number and trigger settings before the start of each run.

6.3.5.4 Monitoring tools

In order to export data from the HLT at any point in the analysis chain, a tool called HOMER (HLT On-line Monitoring Environment including ROOT) was developed. It provides a general access method via two user transparent mechanisms of connecting to PubSub components: via a TCP/IP port or shared memory. A reader library allows access to data blocks and can directly be used from the interactive ROOT prompt.

The open concept of HOMER makes detector monitoring independent of the specific monitoring back-end. The offline monitoring AliEve is also used for general detector online monitoring applications. AliEve is part of AliRoot and combines an event display including 3D visualisation with tools for investigation and browsing of ROOT data structures and histograms. Specific monitoring components run on the HLT which accumulate results of the analysis of many events. The data can be in any format, also a complex ROOT data structure, e.g. a histogram. This makes the HLT a versatile monitoring tool which allows not only examination of the quality of the raw data, but also the monitoring of analysed data online.

6.3.6 Online data processing

The studies described in the following subsections were all performed on data simulated with the ALICE offline software AliRoot.

6.3.6.1 Event reconstruction

The Time Projection Chamber (TPC) is read out by 557 568 channels, delivering event sizes up to 75 MB for central Pb-Pb collisions. Fast online reconstruction algorithms consisting of a Cluster Finder and a Track Follower were developed [219, 226–231]. By using the 2D position in the pad-row plane and the drift time in the gas, the Cluster Finder reconstructs 3D space-points using the centre of gravity approach. The Track Follower maps the space-points into conformal space where helical tracks can be fitted by a linear parametrisation. A helix-fit of the tracks provides the kinematic properties of the particles. The tracking is done on the sector level. If tracks cross more than one sector, they are merged [226]. The algorithms were originally developed for the STAR L3 trigger [232]. Beyond multiplicities $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta$ of 4000 the Cluster Finder efficiency deteriorates

2008 JINST 3 S08002

due to a significant number of overlapping clusters. A fast Hough-transform tracker was developed for this scenario, using the raw data directly [231].

The Transition Radiation Detector (TRD) reconstruction on HLT was designed to directly use the algorithms implemented for the offline analysis. The offline HLT-TRD cluster finder and the HLT-TRD tracker are able to handle a full non-zero suppressed TRD event (load of 43.2 MB) in real time. In the case of zero-suppressed data the volume of a raw Pb-Pb $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$ event does not exceed 7.1 MB including a 20% coding overhead. Given the design rate of the HLT-TRD event processing chain as of 200 Hz the HLT cluster allocates 6 processing nodes per TRD super-module to fulfill the timing requirements. In addition to the components responsible of the reconstruction the TRD-HLT processing chain incorporates the offline model to produce the calibration reference data and delivers a collection of relevant histograms to the interface to OCDB.

For the Photon Spectrometer (PHOS) detector the main objective is to measure the exact timing and energy information of an electromagnetic shower in a typical 3×3 crystal matrix of the calorimeter. Therefore the signal will be oversampled in order to measure the pulse shape. Consequently, at a trigger rate of 1 kHz the 5 PHOS modules will deliver a maximum of 3 GB/s for non-zero-suppressed and non-baseline-corrected events. Based on the Peak Finder algorithm, processing components were developed to analyse the pulse shape and reduce the data to the relevant time and energy information. Furthermore, algorithms for cluster and shower reconstruction will be run on the HLT.

In the Muon Spectrometer the primary goal of the dimuon High-Level Trigger (dHLT) is to improve the sharpness of the p_{t} cut compared to what is obtained by the L0 trigger. dHLT performs fast, online, partial event reconstruction [17] on stations 4 and 5, which allows it to improve the p_{t} resolution by more than a factor of two. Thus a better separation between background and interesting signal events is possible reducing the background data accepted by the spectrometer. The dHLT can handle at least a 1 kHz event rate with a data rate of 500 MB/s. Event reconstruction starts with hit reconstruction, which uses a simple 3 pad cluster finding algorithm. The efficiency achieved is above 98%, and a resolution of 100 μm and 1 mm along the y and x directions (perpendicular and horizontal to the ground respectively) is reached respectively. For tracking, a track following algorithm [233] is used, giving a single muon reconstruction efficiency above 97% and is independent of the event size. From the found tracks the p_{t} is estimated using a simple calculation involving the particle's deflection angle [12]. This gives a p_{t} resolution of about 0.13 GeV/c and 0.18 GeV/c around the 1 GeV/c and 2 GeV/c p_{t} cuts respectively. The total processing time for a central event is about 2.5 ms. Further details about the algorithms can be found in the dHLT Project Review [234].

6.3.6.2 Online data compression

For the TPC detector, the data rate is expected to reach up to 15 GB/s after zero suppression. Any means for reducing the data size online is therefore desirable in order to increase the number of events which can be collected and written to mass storage. Lossless and lossy data compression techniques like entropy encoding and vector quantisation were evaluated using simulated TPC data [235]. The results indicate that standard compression algorithms achieve compression factors of about 2. However, HLT online pattern recognition results in track and cluster information which effectively models the TPC data and can be utilised to compress such data more efficiently and

2008 JINST 3 S08002

reduce the data size to about 11%, with a negligible loss of tracking performance for the expected particle multiplicities at LHC [226, 236].

In PHOS, after zero-suppression and skipping of empty channels in the front-end electronics, the data rate into HLT is about 300 MB/s. Online processing of the pulse shapes results in an energy and a time-of-flight information per channel. This information is forwarded to DAQ for achieving at a data rate of about 20 MB/s which corresponds to a data volume reduction factor of 15.

6.3.6.3 Calibration procedure

Detector algorithms are implemented by detector groups in order to create useful calibration data, which is archived in the OCDB. These algorithms are designed to run in both offline and HLT framework. In the HLT, the detector algorithms are embedded in normal HLT processing components. A special base class for those calibration components was added to the HLT analysis framework, as calibration algorithms mainly accumulate data of several events and deliver the result at the end of run. The calibration data can be shipped via 3 parallel output streams namely the OCDB, to TCP ports (HLT online monitoring) and to the DAQ via the data stream. Calibration components publish results at the end of the run via the offline interface to the OCDB (see section 6.3.5).

For the TPC Calibration the general off-line calibration classes for signal and noise were integrated in the HLT framework. They were successfully tested, processing more than one million raw data events recorded during the TPC commissioning phase.

Benchmarks indicate less than 16% of the total time is spent inside the HLT framework.

The calibration parameters for PHOS calibration mainly consist of energy and time-of-flight distribution for each detection channel, for relative calibration as well as π^0 invariant mass histograms for absolute calibration. Using simulated data, the calibration components were successfully tested at the HLT cluster.

6.4 Offline computing

6.4.1 Introduction

The role of the Offline Project is the development and operation of the framework for data processing. This includes tasks such as simulation, reconstruction, calibration, alignment, visualisation and analysis. These are the final steps of the experimental activity, aimed at interpreting the data collected by the experiment and at extracting the physics content. In an experiment of the complexity and dimension of ALICE, this implies the development and operation of a quite diverse set of environments. In particular we can distinguish three main areas:

- Distributed computing: The computing resources required to process the ALICE data are such that they cannot be concentrated in a single computing centre. Therefore data processing is distributed onto several computing centres located worldwide. Grid Middleware allows treatement of this heterogeneous collection of distributed computing resources as an integrated computing centre.

- Offline data processing: This includes the typical offline activities of calibration, alignment, reconstruction, simulation and analysis, both centrally organized and end-user driven. The offline data processing typically uses the framework for distributed computing and only in some rare cases runs on local resources.

- Quasi-online operations: During proton–proton collisions, the data is reconstructed and analysed quasi-online. Nucleus-nucleus events is reconstructed during the accelerator shutdown following the nucleus-nucleus run. This implies a tight synchronisation with the more 'classical' online activities of DAQ and HLT. It is a new area for offline computing, since it dispenses with the old practice to process the data usually many months after it was collected.

The coordinated operation of all these elements, to realise timely and reliably the physics discovery potential of the data collected by the ALICE experiment, is called the ALICE computing model. The realisation of this model, elaborated during the last ten years by the ALICE collaboration under the coordination of the ALICE Offline Computing Project, constitutes the subject of this section.

6.4.2 Computing model

The inputs to the ALICE Computing Model are the computing resources needed to process the data, the amount of data foreseen and the time lapse between the moment data are recorded and the moment results are needed. The constraints are the foreseeable amount of resources available, their location and the offered service level.

The amount of resources required to process the data coming from the ALICE experiment is of such magnitude that it is not feasible to concentrate it in a single place. While Funding Agencies do realise that this investment in computing equipment is necessary to harness the physics potential of the experiment and to realise the investment made in the hardware, they can only afford the related investments in their own countries. Moreover, even if the computing resources could be all brought to CERN, the maintenance and operation of this complex system would still remain a problem due to manpower availability. The solution to this problem is to provide the necessary computing resources in centres distributed around the world. In the case of ALICE the number of such centres is around sixty, spread in more than thirty countries.

The amount of different tasks to be performed, the necessity to perform them in a timely and documented fashion, and the number of users involved require that this heterogeneous distributed system works as a single integrated computing centre. This is by no means a small challenge, for which there is no ready-made solution to be found. This problem is the focus of current computing science research and development that goes under the name of Grid [239].

The distributed computing infrastructure serving the LHC experimental programme is coordinated by the Worldwide LHC Computing Grid (WLCG) project. The WLCG computing infrastructure is, by nature, highly hierarchical. All real data originate from CERN, with a very large computing centre called Tier-0. Large regional computing centres, called Tier-1, share with CERN the roles of a safe storage of the data on high reliably storage media (at present, magnetic tapes) and to perform the bulk of the organised processing of the data. Smaller centres, called Tier-2, are

2008 JINST 3 S08002

logically clustered around the Tier-1's. The main difference between the two kind of centres is the availability of high-reliability mass-storage media at Tier-1's. Tier-2's use the 'closest' Tier-1 to store the data that they produce. The major role of Tier-2's is simulation and end-user (sometimes also called chaotic) analysis.

Within the WLCG structure, a centre, to qualify as a Tier-1 or Tier-2, has to sign and follow up the corresponding, morally binding, Service Level Agreement (SLA), which specifies Quality -Of-Service (QoS) and intervention delays.

Smaller centres, corresponding to a departmental computing centre and sometimes called Tier-3's, contribute to the computing resources but there is no definite role or definition for them.

6.4.2.1 Computing needs

In determining the computing needs of the ALICE experiment, we distinguish two modes of operation, the proton–proton and the nucleus–nucleus data taking. The amount of data collected depends on the running conditions (such as luminosity, triggers, available detectors, settings in the detectors' hardware, DAQ and HLT mode, etc.) and on their evolution during the life of the accelerator. The time to process these data and the amount of secondary data produced depends on the offline programs, which are in constant evolution following the requirements of the physicists. This makes it very difficult to provide a firm estimate of the needs, particularly in the startup phase, as we have witnessed in these years a constant redefinition of the accelerator commissioning scenario.

We present here the estimation of the computing resources needed in a Standard Data Taking Year (SDTY). These data are a reasonable compromise between the figures determined at the time of writing the Physics Performance Report [20, 21] and the current values, which are larger, but still subject to optimisation, both as far as the data size and the computing time are concerned.

In table 6.4 we show the parameters for a SDTY and in table 6.5 we report the computing needs derived using the WLCG [237] standard efficiency factors. The figures for the initial periods, before standard running conditions are reached, are more difficult to calculate as they depend on the exact accelerator schedule.

6.4.2.2 Data processing strategy

The data processing strategy and the Tier computing centres hierarchy derive from the Monarch model [238]. The strategy varies according to the type of collision. During proton–proton collision the data, recorded at an average rate of 100 MB/s, are written by the DAQ on a disk buffer at the CERN (Tier-0) computing centre, where the following four activities proceed in parallel on the RAW data:

- Copy to the CASTOR tapes;
- Export to the Tier-1 centres to have a second distributed copy on highly-reliable storage media and to prepare for the successive reconstruction passes that will be processed in the Tier-1 centres;
- First pass processing at the Tier-0 centre. This includes: reconstruction, production of calibration and alignment constants and scheduled analysis;

Table 6.4: Parameters of the ALICE Computing Model wherefrom the computing needs are determined. The proton–proton RAW event size is an average figure. The real pile-up factor will change during the run as a function of the luminosity. The Pb–Pb data rate is an average figure. The actual event size depends on the trigger and the event multiplicity. What is important to retain is that the 'bandwidth budget' of 1.25 GB/s will be fully exploited, irrespectively of the chosen trigger.

Category	Item	Value	
		pp	Pb–Pb
Real Data	RAW single pp	200 kB	
	pp pile-up	5	
	Rate	100 Hz	100 Hz
	RAW size per event	1.1 MB	13.75 MB
	ESD size per event	40 kB	3 MB
	Tag catalogue per event	11 kB	11 kB
	Yearly running time	10^7 s	10^6s
	Events per year	10^9	10^8
Simulated Data	RAW size per event	400 kB	300 MB
	ESD size per event	90 kB	6 MB
	Tag catalogue per event	11 kB	11 kB
	Events per year	10^9	10^7
	Signals per underlying event		10
Statistics	Reconstruction passes per year	3	
	RAW replication factor	2	
	ESD replication factor	3	
	Scheduled analysis passes per reconstruction	3	
	Chaotic analysis passes per year	20	

- Fast processing of selected sets of data, mainly calibration, alignment, reconstruction and analysis on the CERN Analysis Facility (CAF) that will be described in section 6.4.4.7.

During nucleus–nucleus runs, the model is slightly different, as the rate of data acquisition is so high that an excessive amount of computing resources and network bandwidth would be necessary for quasi-online processing. Therefore the processing of the nucleus–nucleus RAW data proceeds as follows:

- Registration of the RAW data in CASTOR;
- Partial export to the Tier-1 centres to allow remote users to examine the data locally;
- Partial first pass processing at the Tier-0 centre to provide rapid feedback on the offline chain;
- Fast processing, mainly calibration, alignment, reconstruction and analysis on the CAF.

At the end of the period of nucleus–nucleus data taking, there will be a four-months shutdown of the accelerator. During this period the nucleus–nucleus data are read back from tapes and are

Table 6.5: Computing needs during a standard data taking year.

Category	Item	Value	
		pp	Pb–Pb
CPU Resources / year (MSI2k s)	Reconstruction of real and simulated data	4.6×10^7	5.7×10^8
	Generation of of simulated data	4.9×10^7	2.1×10^8
	Scheduled analysis	2.3×10^8	3.4×10^8
	End-user analysis	5.8×10^7	8.7×10^7
	Total	1.6×10^9	
Mass storage / year (TB)	RAW data	1.1×10^3	1.4×10^3
	Reconstructed data	5.8×10^2	3.7×10^3
	Simulated RAW data	4.0×10^2	3.0×10^3
	Reconstructed simulated data	5.8×10^2	3.7×10^3
	Calibration	1.0×10^1	
	Total	1.2×10^4	
Disk (TB)	Output buffer at CERN	2.4×10^2	
	Disk at Tier-1 centres	1.2×10^4	
	Disk at Tier-2 centres	4.3×10^3	
	Total	1.6×10^4	

processed in a way similar to proton–proton data (export to the Tier-1 centres, first pass processing, fast processing of selected data) but at a rate four times lower than the actual nucleus–nucleus data acquisition. This requires a large-scale resource deployment that is still affordable within the budgetary and infra-structural constraints.

The quasi-online data processing scenario is more complicated, and it will be described in somewhat greater detail in section 6.4.4.7.

During the first pass reconstruction, high-precision alignment and calibration data are produced, as well as a first set of Event Summary Data (ESD) and Analysis Object Data (AOD). The feedback derived from the first pass, including analysis, is used to tune the code for the second pass processing. In our computing model, we have foreseen three processing passes. However this has to be interpreted as an estimation of the computing resource budget needed to extract the physics content from the data. We expect the second pass reconstruction to be composed of a large number of jobs processing only a fraction of the data, to tune the chain in order to perform one more complete pass on the data, which constitutes the third pass.

One full copy of the raw data is stored at CERN, and a second one is shared among the Tier-1's outside CERN. Reconstruction is shared by the Tier-1 centres, CERN being in charge of processing the first pass. Subsequent data reduction, analysis and Monte Carlo production is a collective operation where all Tier's participate, with Tier-2's being particularly active for Monte Carlo and end-user analysis.

6.4.3 Distributed computing

6.4.3.1 ALICE Grid

The implementation of the above model relies on a distributed computing infrastructure enabled by Grid Middleware. Since 2001, ALICE developed a set of Middleware services, AliEn [240], which implements the above model. With the development of various large Middleware projects, it became possible to replace some of these services with the services offered by these projects. In the resulting architecture, the user interacts with the Grid via the AliEn User Interface (UI), and the services are offered by a combination of AliEn Middleware, providing high-level or ALICE-specific services, and the Middleware installed on the computing centre, providing basic services.

This architecture allows a seamless distribution of high-level services, offering uniform functionality and UI across different Grid domains, thus implementing one of the most characteristic features of the 'Grid vision'. The system has been in continuous operation since 2001 with periodic large scale exercises called 'data challenges' performed to test the evolution and scalability of the system. During the last year before the data taking, these data challenges were replaced by a continuous 'production mode' in order to verify the operational stability of the system.

The AliEn system is built around Open Source components, uses Web Services model and standard network protocols. AliEn Web Services play the central role in enabling AliEn as a distributed computing environment. The user interacts with them by exchanging SOAP (Simple Object Access Protocol) [241] messages and they constantly exchange messages between themselves behaving like a true Web of collaborating services. AliEn consists of the following key components and services:

Authentication. AliEn supports various authentication schemes, implemented using the extensible Simple Authentication and Security Layer (SASL) protocol, in particular GLOBUS GSI/GSSAPI that makes it compatible with the WLCG security model.

Authorisation. The authorisation in AliEn is similar to any UNIX-like system. Files have read, write and execution permissions, and they can be set for the owner of the file, group, or world.

Auditing services. Every job (task) submitted and executed in AliEn is tracked with a unique job identifier. The AliEn Task Queue database holds complete information on all active jobs: the job owner, the job scripts, input and output files, processing stage, execution site and host. All completed jobs are moved to an archive database. The job auditing is possible at any stage of a job and tools are available to do per-job and summary audit of both the active and archived jobs.

Workload management. The AliEn workload management system is based on a *pull* approach. A central service manages all the tasks, while computing elements are seen as 'remote queues' and can provide access to a single machine dedicated to run a specific task, a cluster of computers, or even an entire foreign Grid. When jobs are submitted, they are sent to the central queue. The workload manager optimises the queue taking into account job requirements based on input files, CPU time, architecture, disk space, etc. The remote queues

advertise their capabilities to the Job Broker. If there are free resources, the Job Broker signals the remote queue to start Job Agents. At this point, the real job is still not assigned to the site. Once the Job Agents start running on the Worker Nodes (WN), they do a set of sanity checks of the environment, and advertise to the Job Broker their capabilities. The Job Broker compares these with the requirements of the jobs in the Task Queue and in case of a successful match, assigns the job to a Job Agent. The Job Agent will then execute the job. If at the end of the job there are still sufficient resources, in particular lifetime, the Job Agent will ask again the Job Broker for tasks. The AliEn user interface uses the Condor ClassAds [242] as a Job Description Language (JDL). The Workload Management service can modify the job's JDL entry by adding or elaborating requirements based on the detailed information it gets from the system like the exact location of the dataset and replicas, client and service capabilities. When a job requires several files, the workload management systems 'splits' the job in several sub-jobs, each of them dealing with files that are co-located at the same Storage Element (SE). The workload management system still handles the job as a single entity, and keeps track of each individual sub-job.

API services. The API (Application Programming Interface) services provide a SOAP interface to the AliEn command set. This interface allows the building of very thin clients, requiring only one library (gsoap) to be completely Grid-enabled. The API services are used to provide an AliEn interface in ROOT.

Data management. Real files are managed by an AliEn SE. Access to the data is obtained via a POSIX interface and mediated by the xrootd program [243]. When a file is requested by an application running on a worker node, the following chain of events is triggered:

1. The File Catalogue is queried with the Logical File Name (LFN) or the Globally Unique IDentifier (GUID) of the file to obtain permission to perform a given operation in the form of a security envelope.
2. The local xrootd redirector is contacted to get access to a copy of the physical file. The Physical File Name (PFN) is obtained from the File Catalogue, but it may also be generated on the fly from the GUID.
3. The local xrootd redirector provides the location of the file. At this point two options are possible:
   (a) The file is opened directly at the location provided by xrootd via a POSIX call;
   (b) The file is copied on the local disk of the WN and then opened there.

The xrootd redirector can also offer access to a file that is not local at the computing centre where the job is running. This possibility is not currently exploited because batch jobs are executed where the files are stored. The xrootd program was interfaced with the major storage systems in use in HEP, and can of course run 'natively' on a set of Unix disk servers. Bulk transfer of data between Tier-0 and Tier-1's is performed via FTS [244], under control of an AliEn element called File Transfer Daemon (FTD), which takes care of the needed interaction with the File Catalogue. Data transfer other than Tier-0 $\Leftrightarrow$ Tier-1's is done via xrootd itself.

2008 JINST 3 S08002

File and meta-data catalogues. Input and output associated with any job is registered in the AliEn File Catalogue, a virtual file system in which a file or a file collection (data set) is identified by a GUID. Meta-data can be assigned to files, and in particular the file's 'logical name', i.e. its Unix-like name in the virtual distributed file system.

The file catalogue does not own the files; it only keeps an association between the LFN and (possibly more than one) PFN on a real file or mass storage system. PFNs describe the physical location of the files and include the name of the AliEn storage element and the path to the local file.

The system supports file replication and caching and uses file location information to schedule jobs for execution. The File Catalogue supports Unix-like permissions for files and directories. These are automatically checked when accessing the File Catalogue through the xrootd interface with the security envelope [245] developed by the ARDA group [246].

The File Catalogue supports regular files as well as DataSets and information about running processes in the system (in analogy with the `/proc` directory on Linux systems). Each job sent to AliEn for execution gets a unique ID and a corresponding `/proc/id` directory where it can register temporary files, standard input and output, as well as all job products. In a typical production scenario, only after a separate process has verified the output, will the job products be renamed and registered in their final destination in the File Catalogue.

The data in the File Catalogue are organized by year, accelerator period and run. The data that are common to several runs are stored in a logical partition of the File Catalogue called Offline Condition Data Base (OCDB). The files contained in this portion are selected via a special combination of meta-data and support versioning.

Information service. The information service (IS) keeps track of the status, type and capabilities of all AliEn services, central and at the sites. For instance, all storage elements register themselves in the IS. They also publish the protocols they support. This way, the clients can figure out the best place to store files according to their needs.

Monitoring service. Monitoring in AliEn is based on the MonALISA framework [247]. A hierarchic architecture with selective aggregation at each level is employed for the monitoring, before sending the information upstream. This is a determinant factor in reducing the volume of raw monitoring information as much as possible while preserving important details. All AliEn services and the job wrapper are instrumented with a monitoring module (ApMon). Extensive information is collected about CPU and memory usage, consumed and wall-clock kSI2k time, open files and network traffic per service and job. A global view of the entire AliEn Grid, as well as long term persistence of the data is assured by the MonALISA repository. This service subscribes to general interest data and stores it into a database. It offers both near real-time and history views, with different levels of detail, down to individual user jobs.

AliEn provides a complete Grid solution for the ALICE experiment. It is, however, in our interest to minimise the amount of ALICE code and to maximise the use of common Middleware and

2008 JINST 3 S08002

services. This is in order to reduce the maintenance load to be performed by ALICE members and to rely as much as possible on common infrastructures.

This is possible only to a certain extent, insofar as the various Grid projects do not provide all the services needed by the ALICE Computing Model. To alleviate this situation, AliEn services are interfaced to the Grid systems present on the computational resources which are part of the ALICE Grid. This represents a good compromise, offering to the user a seamless interface to the heterogeneous components of the ALICE Grid, and minimising the amount of proprietary services and components.

AliEn is interfaced with the WLCG and ARC [248] Grid systems, where it is running in production, while the interface with the OSG [249] is at the moment of writing at the prototype stage.

To minimise the intrusiveness of the ALICE-specific Grid services, these are all installed on a single computer at each computing centre, the so called 'VO-Box'. This machine is the contact point between the Grid implemented via the common Middleware and the ALICE Grid. This setup allows us to realise a sort of 'Grid of Grids', where different Grids can seamlessly inter-operate.

During several 'data challenges', AliEn has shown its capability to implement the ALICE Computing Model as described in the Computing Technical Design Report [19] for simulation and reconstruction.

The AliEn capability to run analysis was tested to a lesser degree. At the moment of writing there are many millions of simulated events that were produced and reconstructed, and on which the analysis algorithms are exercised. This part of the framework is the one evolving most rapidly.

As the first step, the analysis framework extracts a subset of the Datasets from the File Catalogue using meta-data selection. Then the framework negotiates with dedicated Grid services the balancing between local data access and data replication. Once the distribution is decided, the analysis framework creates sub-jobs. The framework collects and merges available results from all terminated sub-jobs on request. An analysis object associated with the analysis task remains persistent in the Grid environment so the user can go offline and reload an analysis task at a later date, check the status, merge current results, or resubmit the same task with a modified analysis code.

6.4.4 AliRoot framework

The ALICE offline framework, AliRoot [250], is shown schematically in figure 6.12. Its implementation is based on Object-Oriented techniques for programming and, as a supporting framework, on the ROOT system [251], complemented by the AliEn system which gives access to the computing Grid. These fundamental technical choices result in one single framework, entirely written in C++, with some external programs (hidden to the users) still in FORTRAN.

The AliRoot framework is used for simulation, alignment, calibration, reconstruction, visualisation and analysis of the experimental data. AliRoot has been in continuous development since 1998. It was used to perform simulation studies for the Technical Design Reports of all ALICE subsystems and to optimise their design. In the preparation phase, before the start of data taking, it was used to evaluate the physics performance of the full ALICE detector and to assess the functionality of the framework towards the final goal of extracting physics from the data.

The role of the AliRoot framework is shown schematically in figure 6.13. The kinematics tree containing, for example, the physics processes at the parton level and the results of the

2008 JINST 3 S08002

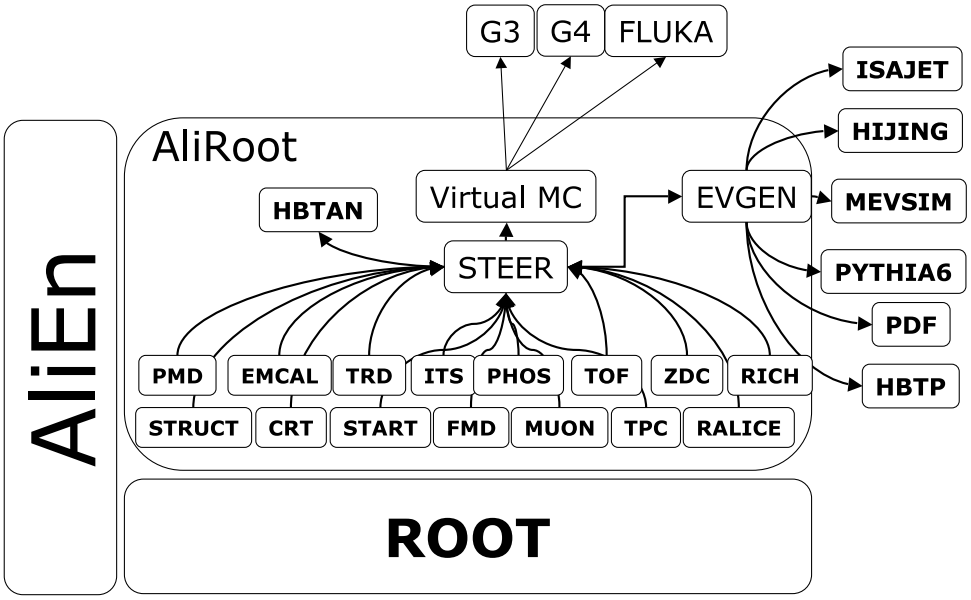

Figure 6.12: Schematic view of the AliRoot framework.

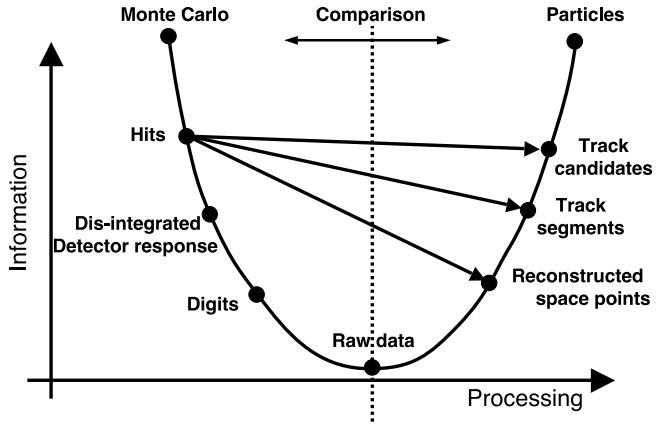

Figure 6.13: Data processing framework.

fragmentation (primary particles) is created by event generators. The data produced by the event generators contain full information about the generated particles: type, momentum, charge, and mother-daughter relationship. During the transport, the response of the detectors to each crossing particle is simulated. The hits (energy deposition at a given point and time) are stored for each detector. The information is complemented by the so called 'track references' corresponding to the location where the particles are crossing user defined reference planes. The hits are converted into digits taking into account the detector and associated electronics response function. Finally, the digits are stored in the specific hardware format of each detector as raw data. At this point the reconstruction chain is activated. Typically the detectors perform local reconstruction such as clusterisation, then a seeding procedure is used to start a Kalman filter [252] tracking. To evaluate

2008 JINST 3 S08002

the software and detector performance, simulated events are processed through the whole cycle and finally the reconstructed particles are compared to the Monte Carlo generated ones.

'Shortcuts' are possible in the interest of saving computing time, for instance using 'hits' directly for physics studies, instead of producing digitised data and then reconstructing them.

The users can intervene in this cycle provided by the framework to implement their own analysis of the data or to replace any part of it with their own code.

The basic design features of the AliRoot framework are re-usability and modularity. Modularity allows replacement of well defined parts of the system with minimal or no impact on the rest. For example, it is possible to change the event generator or the transport Monte Carlo without affecting the user code. Elements of the framework are made modular by defining an abstract interface to them. The codes from the different detectors are independent so that different detector groups can work concurrently on the system while minimising the interference. The adopted development strategy can handle design changes in production code for cases when new elements are introduced. Re-usability is the protection of the investment made by the programming physicists of ALICE. This investment is preserved by designing a modular system and by making sure that the maximum amount of backward compatibility is maintained while the system evolves.

The ROOT framework, upon which AliRoot is developed, provides an environment for the development of software packages for event generation, detector simulation, event reconstruction, and data analysis. It offers, among other features, integrated I/O with class schema evolution, an efficient hierarchical object store with a complete set of object containers, C++ as a scripting language and a C++ interpreter, advanced statistical analysis tools (multidimensional histograms, several commonly used mathematical functions, random number generators, multi-parametric fit, minimisation procedures, cluster finding algorithms etc.), HTML documentation tools and advanced visualisation tools. The ROOT system was extended with ALICE specific classes and libraries grouped in modules. These libraries are loaded dynamically and the contained classes share the same services with the native ROOT classes, including object browsing, I/O, dictionary and so on.

The ROOT system is interfaced with the Grid Middleware in general and, in particular, with the ALICE-developed AliEn system. In conjunction with the PROOF [253] system, which extends ROOT capabilities on parallel computing systems and clusters, this provides a distributed parallel computing platform for large-scale production and analysis.

6.4.4.1 Event simulation

The offline framework was developed to allow for efficient simulations of nucleus–nucleus, proton-nucleus and proton–proton collisions and to provide a predictive and precise simulation of the detector response. It includes the following options:

- Interfaces to several external generators, like for example HIJING [142].
- A simple event generator based on parametrised η and p_{t} distributions can provide a signal-free event with multiplicity as a parameter.
- Rare signals can be generated using the interface to external generators like PYTHIA [254] or simple parameterisations of transverse momentum and rapidity spectra defined in function libraries.

2008 JINST 3 S08002

- Tools to combine underlying events and signal events on the primary particle level (cocktail) and on the digit level (merging).
- Generators for realistic beam-gas and beam-halo event simulations.
- 'After-burners' are used to introduce predefined particle correlations.

To facilitate the usage of different generators, we developed an abstract generator interface that allows the study of full events or signal processes, and a mixture of both, i.e. cocktail events.

Several event generators are available via the generic generator interface class, `TGenerator`. Via this abstract base class we wrap FORTRAN Monte Carlo codes that are thus accessible from AliRoot.

In many cases, the expected transverse-momentum and rapidity distributions of particles are known. In other cases the effect of variations in these distributions must be investigated. In both situations it is appropriate to use generators that produce primary particles and their decays by sampling from parametrised spectra. The corresponding parameterisations are stored in independent function libraries wrapped into classes that can be plugged into the generator.

The modularity of the event-generator framework allows easy integration of objects that are responsible for changing the output of event generators or for assembling new events making use of the input of several events. These processors are generally called 'after-burners'. They are especially needed to introduce a controlled (parameterised) particle correlation into an otherwise uncorrelated particle sample.

6.4.4.2 Detector response simulation

For the detector-response simulation different transport Monte Carlo packages are available, namely GEANT 3 [141], GEANT 4 [255] and FLUKA [139]. They have a very different user interface for signal scoring and geometry definition. To be able to use them to simulate the ALICE detector via the AliRoot framework, they were interfaced to the Virtual Monte Carlo abstract interface in ROOT. Moreover, their native geometry modellers were replaced by `TGeo`, the geometry modeller provided by ROOT. Thanks to this, it is enough to instantiate the appropriate class in the simulation configuration file to simulate ALICE with any of these three codes.

For special studies, usually requiring very large sample sizes, the detector response is simulated via appropriate parameterisations or other techniques that do not require the full particle transport. These application are known as 'fast simulations'. Several were implemented in AliRoot. The systematic error introduced by the parameterisations is in general small compared to the reduction of the statistical error.

The ALICE detector is described in great detail, see figure 6.14, including services and support structures, absorbers, shielding, beam pipe, flanges, and pumps. The `TGeo` package is used for particle transport during simulation, and for alignment and reconstruction. The magnetic field of the solenoid and the dipole magnet is described by a parametrisation of detailed measurements obtained with an accuracy of the order of 1.0 Gauss.

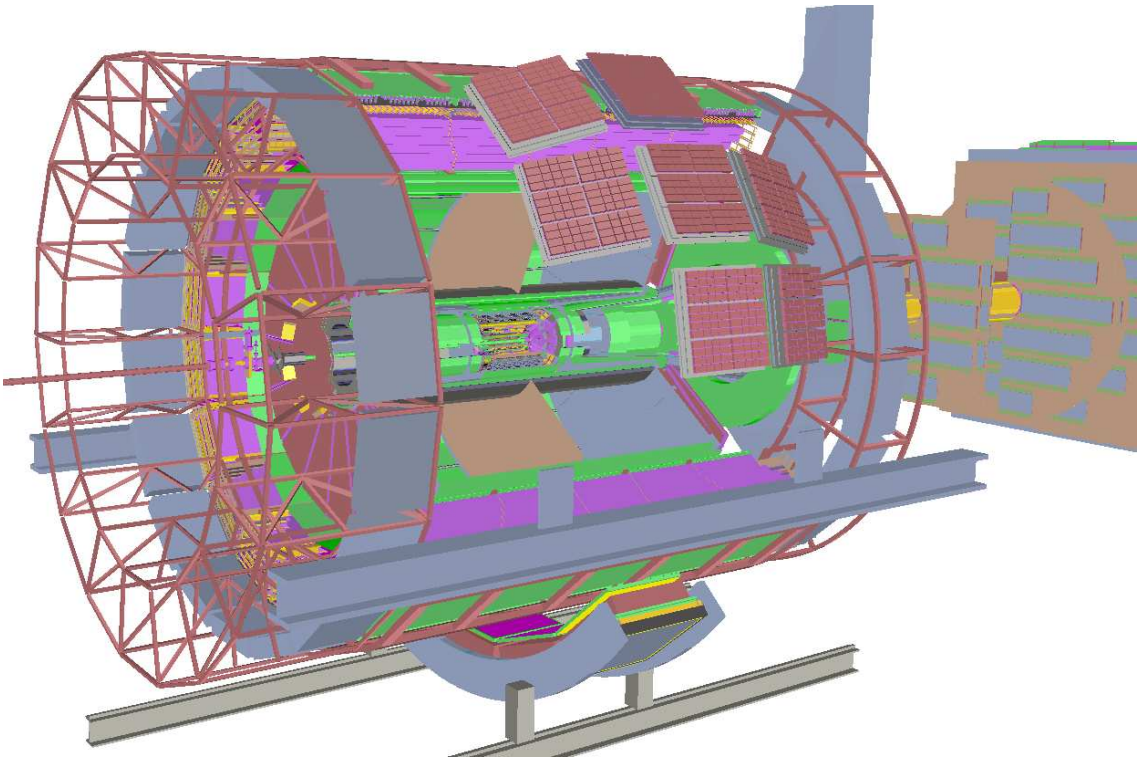

Figure 6.14: Geometry of the ALICE detector in the AliRoot simulation.

6.4.4.3 Alignment framework

Thanks to the use of the `TGeo` package, it is possible to perform detailed alignment studies for the ALICE detector. The parts of the detector that are subject to relative positioning different from the ideal one, are called 'aligneable volumes'.

When the simulation program is started, the ideal geometry is generated via compiled code or read from the OCDB where it was saved in a previous run. Several objects are marked as 'aligneable', that is the geometrical modeller is ready to accept modifications to their position, even if they were obtained by replication. The framework then reads the 'alignment objects' which contain the 'adjustments' in the position of the aligneable objects. The particle transport is then performed in the modified geometry.

The same procedure can be repeated for reconstruction and, therefore, the effect of detector misalignment and the performance of the alignment algorithms can be tested. During reconstruction of real data, the 'best' alignment objects are loaded from the OCDB. For the first pass reconstruction these are the alignment objects produced from survey data. During subsequent reconstruction passes, the alignment objects are produced by the alignment algorithms optimizing the reconstruction quality during processing of the raw data.

Survey data are automatically loaded on the Grid in a standard text format for automatic parsing and conversion to alignment objects. Alignment objects are stored in the OCDB and accessed via meta-data.

2008 JINST 3 S08002

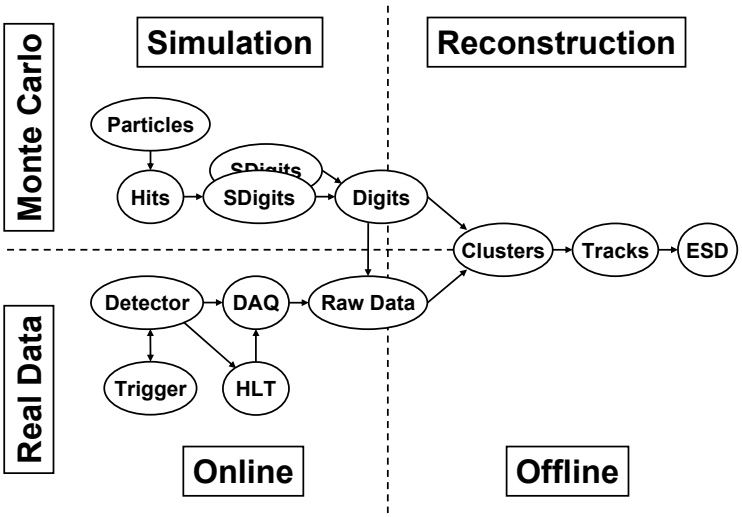

Figure 6.15: Interaction of the reconstruction code with the other parts of AliRoot.

6.4.4.4 Calibration framework

This framework is very similar to the alignment one. The initial calibration constants come either from the detector properties as measured during construction, or from algorithms running online during data-taking aimed at providing a partial calibration sufficient for the first-pass data reconstruction. During the reconstruction itself, better calibration constants can be calculated and stored in the OCDB.

6.4.4.5 Reconstruction framework

The ALICE reconstruction code is part of the AliRoot framework. Its modular design allows its code to be compiled into separate shared libraries and executed independently of the other parts of AliRoot. As an input, the reconstruction uses the digits together with some additional information like module number, readout channel number, time bucket number, etc. The reconstruction can use both digits in a special ROOT format, more convenient for development and debugging purposes, and digits in the form of raw data, as they are output from the real detector or can be generated from the simulated special-format digits above (see figure 6.15). The output of the reconstruction is the ESD containing the reconstructed charged particle tracks (together with the particle identification information), decays with the V^0 (like $\Lambda \rightarrow p\pi$), kink (like charged $K \rightarrow \mu\nu$) and cascade (like $\Xi \rightarrow \Lambda\pi \rightarrow p\pi\pi$) topologies and particles reconstructed in the calorimeters.

The main steering reconstruction class, `AliReconstruction`, provides a simple user interface to the reconstruction. It allows users to configure the reconstruction procedure, include or exclude a detector from the run, and ensure the correct sequence of the reconstruction steps:

- reconstruction steps that are executed for each detector separately (typical example is the cluster finding);
- primary vertex reconstruction;

- track reconstruction and particle identification (PID);
- secondary vertex reconstruction (V^0, cascade and kink-decay topologies).

The `AliReconstruction` class is also responsible for the interaction with the AliRoot I/O sub-system and the main loop over the events to be reconstructed belongs to this class too.

The interface from the steering class `AliReconstruction` to the detector-specific reconstruction code is defined by the base class `AliReconstructor`. For each detector there is a derived reconstructor class. The user can set options for each reconstructor in the form of a string parameter. Detector-specific reconstructor classes are responsible for creating the corresponding specific cluster-, track- and vertex-finder objects and for passing the corresponding pointers to the `AliReconstruction`. This allows one to configure the actual reconstruction process using different versions of the reconstruction classes at the detector level.

The detailed description of the reconstruction and particle identification in all the ALICE detectors can be found in ref. [21] and it is reviewed in the chapter 8. Here we shall only outline briefly the software structure.

The space points are reconstructed by a detector-specific cluster-finding procedure. For each space point we also calculate the uncertainty of the space-point position estimation. All of the central tracker detectors (ITS, TPC, TRD) have their own detailed parametrisation of the space-point position uncertainties. The space-point coordinates together with the position uncertainties are then passed to the track reconstruction. If, in addition to the space-point position, the detector is also able to measure the produced ionization, this information can be used for the particle identification.

Offline track reconstruction in ALICE is based on the Kalman filter approach. The detector specific implementations of the track-reconstruction algorithm use a set of common base classes, which makes it easy to pass tracks from one detector to another and test various parts of the reconstruction chain. For example, we can use smeared positions of the simulated hits instead of the ones reconstructed from the simulated detector response, which is very useful for testing purposes. In addition, each hit structure contains the information about the track that produced it. Although, this implies the storage of extra information, it was proved to be very useful for debugging the track reconstruction code.

Combining (using a Bayesian approach) the particle-identification information coming from the outer detectors (TRD, TOF, HMPID) and $\mathrm{d}E/\mathrm{d}x$ measured by TPC and ITS, ALICE is able to identify charged particles. The neutral particles in the central-rapidity region are identified by the calorimeters (PHOS and EMCAL). There is also the possibility to count photons in the forward region using the PMD.

The muon arm is designed to precisely measure the muon momenta behind the absorber. The muon track-finding algorithm uses the Kalman-filter technique as well.

The algorithms in preparation for the High-Level Trigger (HLT) reconstruction code are implemented as a last step within the AliRoot simulation chain. This allows for studying the HLT track-finding performance and replaying the online trigger decisions during the offline data analysis. The code that reads the data produced by HLT is organised as a 'virtual' detector reconstructor class which derives from the base `AliReconstructor` class and is called by the steering class `AliReconstruction`. The output of the HLT reconstruction is stored in an ESD object using the same format as the offline reconstruction, which facilitates the comparison between offline and HLT reconstruction results.

6.4.4.6 Analysis

Analysis is the final operation performed on the data and the one finally destined to extract physics information. In the ALICE Computing Model, the analysis starts from the ESD produced during the reconstruction step. Analysis tasks produce AOD with standard content condensed from the ESD as well as AODs for specific analyses. Further analysis passes can start from condensed AODs.

We distinguish two different analysis activities, scheduled (or ordered) and end-user (or chaotic) analysis. Scheduled analysis will be performed in a way that sometimes is indicated as 'freight train'. ALICE generic analysis framework 'attaches' a number of 'official' (at the level of the collaboration) algorithms (the 'coaches') and 'carries' them through the data. The advantage is that each event is read only once and the different algorithms are applied to it. Care has to be taken that one coach does not derail the whole train. Scheduled analysis has a very predictable resource consumption and data access pattern, and this makes it an ideal activity for large batch job runs at Tier-1's centres.

End-user analysis comprises all the activities performed by users in the framework of a specific physics analysis activity. Its peculiarity is the unpredictability of both the data access pattern and the resource consumption. End-user analysis is prototyped on local systems with a limited amount of data and then performed interactively on local clusters using the parallel PROOF system, or it is submitted as a batch job to the AliEn system, which will execute on Tier-2 and Tier-1 systems, according to resource availability.

We developped an analysis framework, `AliAnalysis` that can be used for scheduled and end-user analysis. Based on `TSelector` and `TTask`, it was implemented such that the user code is independent of the used computing scheme (local, PROOF or Grid). It also allows us to include of Monte Carlo truth information into the analysis chain so that it can be used for efficiency and acceptance correction studies. The analysis framework permits the splitting of each analysis into a tree of dependent tasks. Each task is data oriented: it registers the required input data and publishes its output. The optimization of the execution chain is done after the registration of all tasks included in the analysis.

6.4.4.7 CERN Analysis Facility (CAF)

The need for a very fast feedback during data taking lead to the deployment of a large cluster for interactive parallel processing of the data with PROOF as the enabling technology. The CAF allows physicists to perform, very rapidly, various operations on the data acquired with the full offline framework, providing information on the quality of the data itself, and on the used calibration algorithms. CAF is also intended for early discovery of important features of the data, in order to direct and refine the 'freight train' analysis passes.

The general architecture of the CAF is shown in figure 6.16.

Most of the activity is on the raw data and intermediate reconstruction output (e.g. clusters). However, the final reconstruction output (ESD and AOD) is also available, to e.g. exercise AOD production on the CAF. The local disk space of the CAF acts as a cache for files stored in other persistent storage systems. The CAF is enabled to stage files from AliEn SE identifying files via the AliEn File Catalogue. Furthermore files can be retrieved from CASTOR directly.

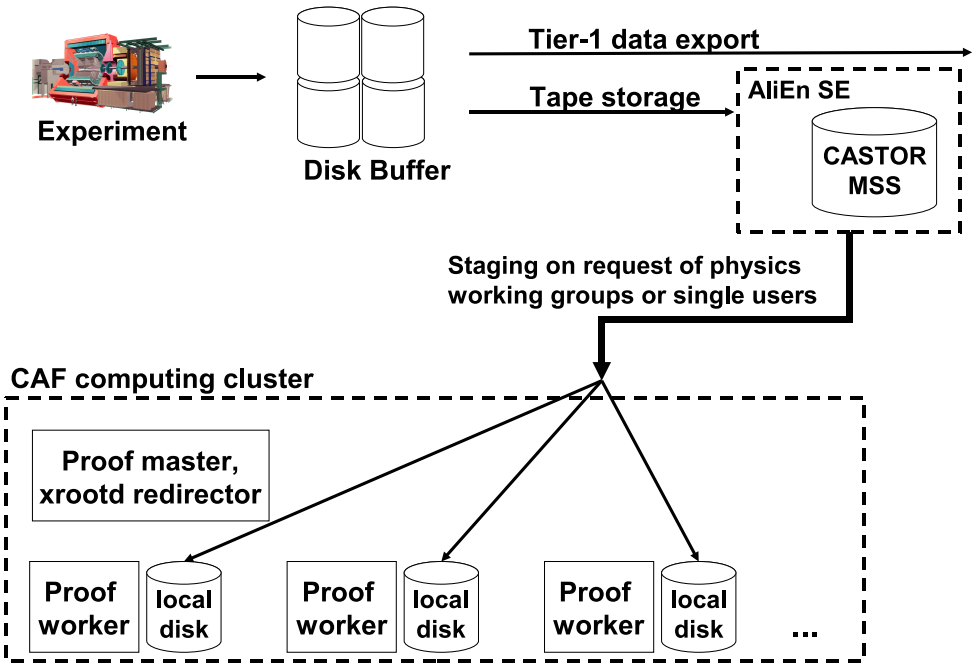

Figure 6.16: Architecture of the CERN Analysis Facility.

Staging is triggered by user's requests to the head node of the system. The system chooses a disk server, and the file is staged to its local disk. As the access is done via the AliEn File Catalogue, the user is able to chose to stage a remote file, located on a SE not necessarily at CERN. All the file staging and access is achieved via xrootd plus a plugin that contains the staging logic from AliEn and CASTOR. The CAF also allows to access files from AliEn and CASTOR without staging them.

The above functionality is part of xrootd and was tested on the CAF. This functionality is however insufficient for the intended use of the CAF. We foresee different groups, subsystems and Physics Working Groups, accessing the system at the same time. It is therefore necessary to introduce a system of CPU and disk quotas in order to regulate the usage of the CAF.

The performed tests indicate that with a reasonable mix of queries of different lengths, each user obtains a performance which is very close, or even larger, than their quota of resources (number of machines per user), because of the necessary pauses between queries in interactive usage. The overall utilisation of the system in this case stays above 75%. These tests were conducted with normal batch nodes which were removed from the CERN lxbatch system. This is very important because the possibility to use standard lxbatch nodes reduces the cost of the CAF.

The CAF is monitored on different levels with the help of MonALISA. Apart from host-monitoring, query statistics is recorded after each query. This includes the amount of data processed, the number of events, the CPU time consumed and the time the user waited for the query to finish. This information is used to visualize the utilization of the cluster as well as for the CPU fairshare mechanism.

6.4.4.8 Software development environment

The ALICE software community is composed of small groups of 2–3 people who are geographically dispersed and who do not depend hierarchically from the Computing Coordinator. This affects not only the organisation of the computing project but also the software development process itself as well as the tools that have to be employed to aid this development.

This is a novel situation for HEP quantitatively if not qualitatively. The experiments' offline programs have to be developed by truly distributed teams. Traditional software engineering methods are not meant to cope with this reality. The development of AliRoot has therefore followed some empirical principles.

Requirements are necessary to develop the code. ALICE users express their requirements continuously with the feedback they provide. The core offline team redirects its priorities according to the user feedback. In this way, we make sure that we are working on the highest priority item at every moment, maximising the efficiency of our work by responding to user's need and expectations.

Design is good for evolving the code. AliRoot code is re-designed continuously since the feedback received from the users is folded into the continuous design activity of the core offline team. At any given time there is a design of the system considered as a short-term guide to the next development phase and not a long-term static objective. This design activity is essential since it rationalises the development and indicates how best to include the features demanded by the users.

Testing is important for quality and robustness. While component testing is the responsibility of the groups providing modules to AliRoot, full integration testing is done nightly to make sure that the code is always functional. The results of the tests are reported on the Web and are publicly accessible [256]. The problems are reported on a Savannah server and assigned to the responsible developer.

Integration is needed to ensure coherence. We have a single code base handled via the SVN [257] version management tool where one or two developers for every module store their code. The different modules are largely independent to allow parallel development. A HTML version of the code is also extracted and created every night thanks to the ROOT documentation tools [258].

Discussions are valuable for users and developers. In ALICE, the discussion list is a valid place to express requirements and discuss design, extending the discussions we have during offline meetings. It is not infrequent that important design decisions are taken on the basis of an email discussion thread where all interested users participate. Redefinition of planning and sharing of work is also very frequent, adding flexibility to the project and allowing it to address both short-term and long-term needs. A specialised Web-based tool is used to track the current planning tasks and the available FTE resources.

2008 JINST 3 S08002

The ALICE offline code is composed of one single framework, Object Oriented in design and implemented in C++ written by multiple authors. This requires a high degree of uniformity in the code as its structure has to be readable and easily understandable. We have therefore adopted a limited set of coding and programming rules [259]. To effectively enforce them, we have developed, together with IRST, a code analysis tool [260] able to check compliance of the code with our coding rules. This now runs nightly and a table of the violations in all modules is published on the Web [261].

This development requires a specific release policy. It is based on the principle of fixed release dates with flexible scope. Given that the priority list is dynamically rearranged, it is difficult to define the scope of a given release in advance. Instead, we decided to schedule the release cycle in advance. The current branch of the SVN repository is 'tagged' approximately once a week with one major release every couple of months.

A release is announced a fortnight before. If a conflict arises, the scope of the release is reduced but the date is not moved. On the date, a branch is created. Usually before the announced date a large amount of code is checked into the head branch, so once the branching is performed, the code tend to be rather unstable. Then the code in the release branch is stabilised and corrected for one more fortnight and finally the release is tagged.

We have tried to make installation as fast and reliable as possible. AliRoot is one single package that links only to ROOT, and of course to the transport Monte Carlo. Thanks to this, we have not needed configuration management tools. We try to be as independent of the specific version of the operating system and compiler as possible. To ensure easy portability to any future platform, one of our coding rules states that the code must compile without warning on all supported platforms. At this time, these are Mac OS X, Linux on IA32 and IA64 machines, and Solaris.

6.4.5 The quasi-online environment

One of the requirements to the ALICE Offline framework is to provide physics results as early and as reliably as possible. The amount of data produced both in proton–proton and in nucleus–nucleus running and the limitation on the amount of resources available (disk spindles, tape drives, CPUs) sets a minimum time for a full reconstruction pass through the data. To obtain meaningful physics results timely, the data have to be processed quasi online for proton–proton and at a very high pace after the nucleus–nucleus run during the shutdown, as explained in the computing model section 6.4.2.2.

To perform such a reconstruction, it is necessary to have 'reasonable' calibration information available immediately after the end of each run. This is achieved via a quasi-online framework developed jointly with the Data Acquisition (DAQ) and High Level Trigger (HLT) project, which is shown in figure 6.17.

The main idea is to provide calibration information that can be used at the end of the run for the first reconstruction pass. This information comes from different sources.

- DAQ: The first source of calibration are the data themselves. The data for calibration are collected on the Local Data Concentrators (LDC), Global Data Concentrators (GDC) and the monitoring farm local disks. The various run conditions also participate to the calibration process, and these are collected from the DAQ logbook or from the Experiment Control System (ECS) which is part of the DAQ.

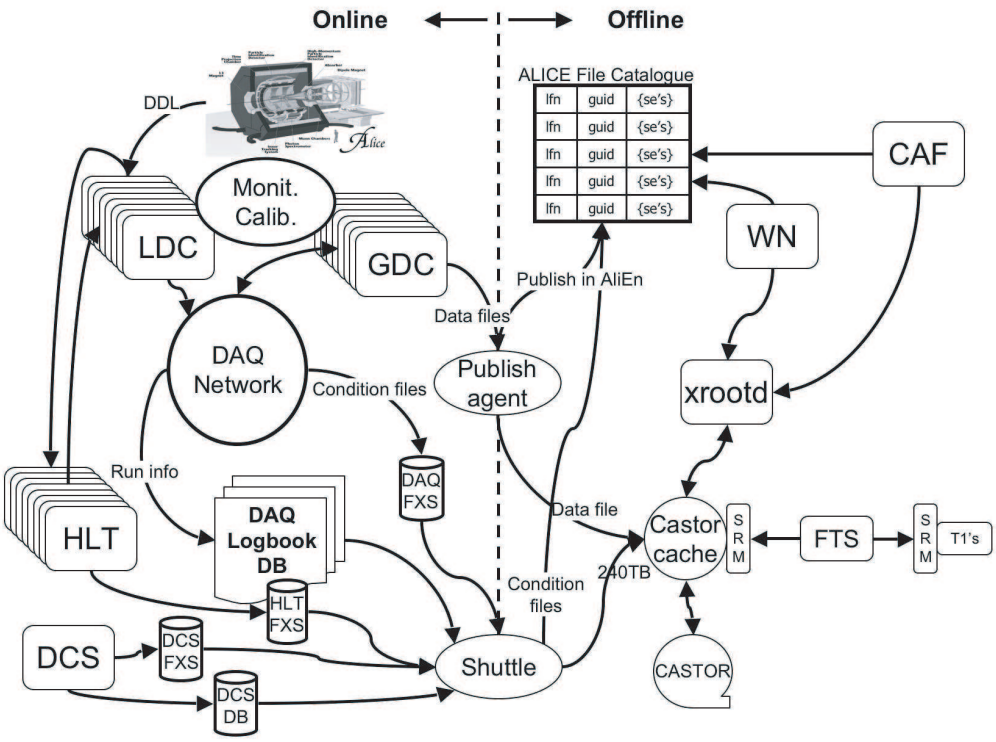

Figure 6.17: The quasi-online framework.

- DCS: A large part of the information necessary for calibration comes from the Detector Control System. A fraction of the DCS data consists of parameters monitored continuously and unasinchronously with respect to data taking, and archived in an Oracle database (DCS archive). A special server called AMANDA provides an API to the DCS archive and allows to recuperate these data. DCS may also produce calibration data which cannot be archived on the Oracle database: these data are shipped to Offline through a file exchange server.
- HLT: The High-Level Trigger is of course a very important source of calibration information. It is also a user of calibration data for its own reconstruction.
- Trigger: The trigger scalers can also provide information needed.
- DCDB: Some calibration information is static and it is measured once and for all in the laboratory and registered in the Detector Construction Data Base (DCDB). The subset of these data which are needed for Offline reconstruction are transferred to the OCDB before the start of the data taking.

These data are collected during the run, but only after they are processed by the detector-specific algorithms that are used to calculate the calibration parameters. The original data from which the parameters are calculated are called 'reference data' and some of them are stored and catalogued as the rest of the data. The processed data are put on shared storage areas, called File eXchange Servers (FXS). A program called Shuttle runs at the end of each data-taking run. Its task is to collect all the data from the FXSs and from the DCS archive, further process them if necessary,

and then catalogue them in the OCDB, a special portion of the AliEn File Catalogue. These data will be used for the first pass reconstruction.