Chapter 9

Data Acquisition

The architecture of the CMS Data Acquisition (DAQ) system is shown schematically in figure 9.1. The CMS Trigger and DAQ system is designed to collect and analyse the detector information at the LHC bunch crossing frequency of 40 MHz. The rate of events to be recorded for offline processing and analysis is on the order of a few 10^2 Hz. At the design luminosity of $10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, the LHC rate of proton collisions will be around 20 per bunch crossing, producing approximately 1 MByte of zero-suppressed data in the CMS read-out systems. The first level trigger is designed to reduce the incoming average data rate to a maximum of 100 kHz, by processing fast trigger information coming from the calorimeters and the muon chambers, and selecting events with interesting signatures. Therefore, the DAQ system must sustain a maximum input rate of 100 kHz, for a data flow of $\approx$ 100 GByte/s coming from approximately 650 data sources, and must provide enough computing power for a software filter system, the High Level Trigger (HLT), to reduce the rate of stored events by a factor of 1000. In CMS all events that pass the Level-1 (L1) trigger are sent to a computer farm (Event Filter) that performs physics selections, using faster versions of the offline reconstruction software, to filter events and achieve the required output rate. The design of the CMS Data Acquisition System and of the High Level Trigger is described in detail in the respective Technical Design Report [188].

The read-out parameters of all sub-detectors are summarized in table 9.1. Each data source to the DAQ system is expected to deliver an average event fragment size of $\approx$2 kByte (for pp

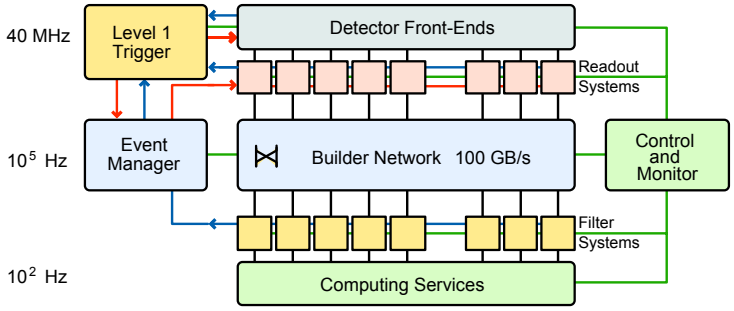

Figure 9.1: Architecture of the CMS DAQ system.

Table 9.1: Sub-detector read-out parameters.

sub-detector	number of channels	number of FE chips	number of detector data links	number of data sources (FEDs)	number of DAQ links (FRLs)
Tracker pixel	≈ 66 M	15840	≈ 1500	40	40
Tracker strips	≈ 9.3 M	≈ 72 k	≈ 36 k	440	250 (merged)
Preshower	144384	4512	1128	56	56
ECAL	75848	≈ 21 k	≈ 9 k	54	54
HCAL	9072	9072	3072	32	32
Muons CSC	≈ 500 k	≈ 76 k	540	8	8
Muons RPC	192 k	≈ 8.6 k	732	3	3
Muons DT	195 k	48820	60	10	10
Global Trigger	n/a	n/a	n/a	3	3
CSC, DT Track Finder	n/a	n/a	n/a	2	2
Total	≈ 55 M			626	458

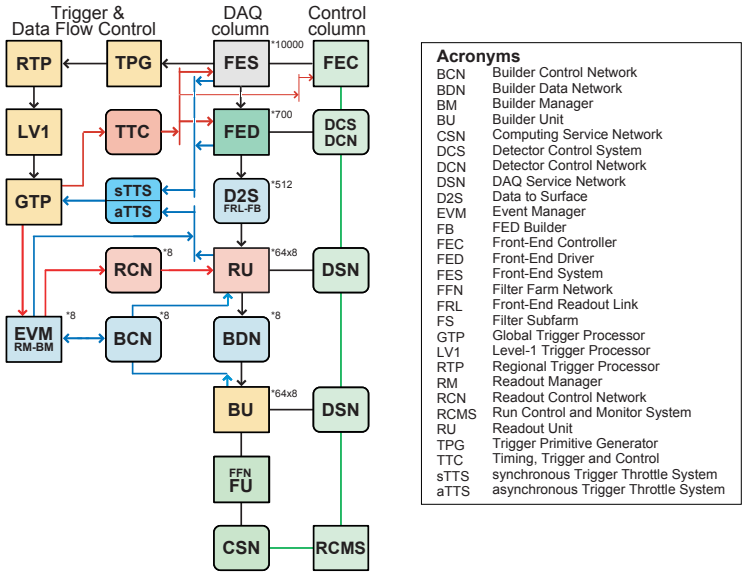

Figure 9.2: Schematic of the functional decomposition of the DAQ. The multiplicity of each entity is not shown for clarity.

collisions at design luminosity). In some case two data sources are merged in order to reach this nominal size.

A schematic view of the components of the CMS DAQ system is shown in figure 9.2. The various sub-detector front-end systems (FES) store data continuously in 40-MHz pipelined buffers. Upon arrival of a synchronous L1 trigger (3.2 μs latency) via the Timing, Trigger and Control (TTC) system [204, 207], the corresponding data are extracted from the front-end buffers and pushed into the DAQ system by the Front-End Drivers (FEDs). The data from the FEDs are read

2008 JINST 3 S08004

into the Front-end Read-out Links (FRLs) that are able to merge data from two FEDs. The number of FRLs corresponding to the CMS read-out parameters of table 9.1 is 458. In the "baseline" configuration, there are 512 FRLs. These additional inputs are used for combined operation with the TOTEM experiment [2], and for inputs from local trigger units, among others. The sub-detector read-out and FRL electronics are located in the underground electronics room (USC).

The event builder assembles the event fragments belonging to the same L1 from all FEDs into a complete event and transmits it to one Filter Unit (FU) in the Event Filter for further processing. The event builder is implemented in two stages (referred to as FED-builder and RU-builder) and comprises a number of components, which are described below (section 9.4). The DAQ system can be deployed in up to 8 *slices*, each of which is a nearly autonomous system, capable of handling a 12.5 kHz event rate. The event builder is also in charge of transporting the data from the underground to the surface building (SCX), where the filter farm is located.

The DAQ system includes back-pressure all the way from the filter farm through the event builder to the FEDs. Back-pressure from the down-stream event-processing, or variations in the size and rate of events, may give rise to buffer overflows in the sub-detector's front-end electronics, which would result in data corruption and loss of synchronization. The Trigger-Throttling System (TTS) protects against these buffer overflows. It provides fast feedback from any of the sub-detector front-ends to the Global Trigger Processor (GTP) so that the trigger can be throttled before buffers overflow. During operation, trigger thresholds and pre-scales will be optimized in order to fully utilize the available DAQ and HLT throughput capacity. However, instantaneous fluctuations might lead to L1 trigger throttling and introduce dead-time. CMS has defined a *luminosity section* as a fixed period of time (set to 2^{20} LHC orbits, corresponding to 93 s), during which trigger thresholds and pre-scales are not changed. The GTP counts the live-time (in numbers of bunch crossings) for each luminosity section and records them in the Conditions Database for later analysis.

The required computing power of the filter farm to allow the HLT algorithms to achieve the design rejection factor of 1000 is substantial and corresponds to O(1000) processing nodes. At the LHC luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ it is foreseen to operate at a maximum L1 rate of 50 kHz, corresponding to 4 installed DAQ slices. It has been estimated [189] that under these conditions the HLT algorithms will demand a mean processing time of around 50 ms on a 3-GHz Xeon CPU-core. This implies for the 50-kHz DAQ system that an equivalent of about 2500 CPU-cores must be deployed for the HLT. After optimising the trigger selection for the LHC design luminosity, the estimated required computing power is expected to be roughly twice as much for 8 DAQ slices operating at a 100 kHz L1 rate.

For the first LHC run, the Event Filter is foreseen to comprise 720 PC-nodes (with two quad-core processors) for 50 kHz operation. Based on initial experience and evaluation of candidate hardware, the additional filter farm nodes for 100 kHz operation at design luminosity will be procured. The design of the DAQ system allows for this gradual expansion in event building rate and processing.

9.1 Sub-detector read-out interface

The design of the FED is sub-detector specific, however, a common interface from the FED to the central DAQ system has been implemented. The hardware of this interface is based on S-

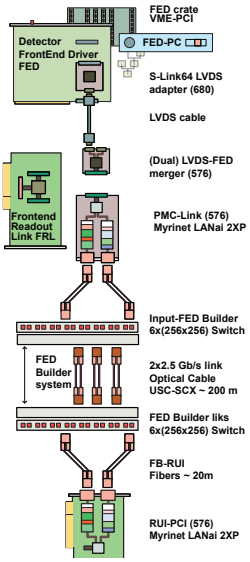

Figure 9.3: Diagram of a generic sub-detector read-out into the FED builder.

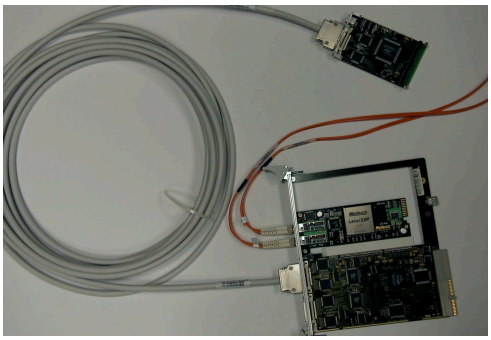

Figure 9.4: Photograph of S-link64 sender card, LVDS cable and compact-PCI FRL card with embedded LANai2XP NIC. Connected to the NIC are fibres that go to the FED Builder input switch.

Link64 [208]. The FED encapsulates the data received from the front-end electronics in a common data structure by adding a header and a trailer that mark the beginning and the end of an event fragment. The header and trailer partially consists of event information used in the event building process, such as the event number derived from the TTC L1A signals. The trailer includes a CRC (Cyclic Redundancy Check) of the data record. The payload of the event fragments is only inspected in the Event Filter.

The physical implementation of one element of the sub-detector read-out interface and its FED builder port is shown in figures 9.3 and 9.4.

The S-Link64 Sender card

The S-Link64 Sender card is a custom developed Common Mezzanine Card (CMC), directly plugged into the sub-detector FED. It receives data from the FED via an S-Link64 port and checks the CRC in order to check transmission errors over the S-Link. The card is able to buffer up to 1.6 kByte of data before generating back-pressure to the FED. The CMC has an LVDS converter to interface with a copper cable which can have a maximum length of 11 m. This cable (Multi-conductor round cable v98 manufactured by 3M) comprises 14 twisted pairs, which are individually shielded. The link provides feedback lines in order to signal back-pressure and to initiate an automatic self test. The nominal data transfer rate over the LVDS cable is 400 MByte/s (50 MHz clock, 64 bits), which is twice the maximum sustained design throughput of the DAQ system.

2008 JINST 3 S08004

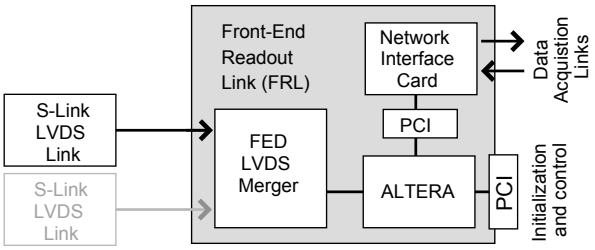

Figure 9.5: FRL layout.

The Front-end Read-out Link

The FRL is a custom 6U Compact-PCI card (figure 9.5). It has three interfaces: an input interface which handles up to two LVDS cables; an output interface to the FED Builder implemented as a 64-bit/66-MHz PCI connector for a Network Interface Card (NIC); and a configuration and control interface which is a PCI bus interface connected to the Compact-PCI backplane. The function of the FRL board is performed by two FPGAs (Altera EP20K100EFC324-1 for the PCI bridge and EP1S10F672C6 for the main logic).

The FRL receives event fragments and checks the CRC in order to check transmission errors over the LVDS cable. In the case where the FRL receives data from two FEDs, the two data records are merged. Data are buffered in memories of 64 kByte size and pushed into the NIC in fixed size blocks via the onboard PCI bus.

The FRL card also provides monitoring features, such as the ability to spy on a fraction of events via the Compact-PCI bus, and to accumulate histograms of quantities such as fragment size.

Up to 16 FRL cards are placed in a crate with a Compact-PCI backplane. Each crate is connected to a PC via a compact PCI bridge (StarFabric CPCI/PCI system expansion board from Hartmann Elektronik), which is used for configuration, control and monitoring. There are 50 FRL crates in total.

9.2 The Trigger Throttling System and sub-detector fast-control interface

The TTS provides the feedback from all FEDs and their associated front-end systems to the GTP. It is a hardwired system, acting on the dataflow with a reaction time of less than 1 μs. Each FED provides fast signals indicating the state of the read-out. The states *Ready, Warning, Busy, Out-Of-Sync* and *Error* are defined (listed in order of increasing priority). Ready, Warning and Busy are generated according to the amount of internal data buffering available and are used to indicate if more triggers can be accepted. Given the trigger rules (section 8.4) and a TTS latency of 1 μs, a FED has to be able to accept 2 more triggers after asserting Busy state. Out-Of-Sync and Error indicate that synchronization was lost or an error occurred in the front-end electronics. The GTP attempts to recover automatically from these states by issuing a L1-Resync or L1-Reset command

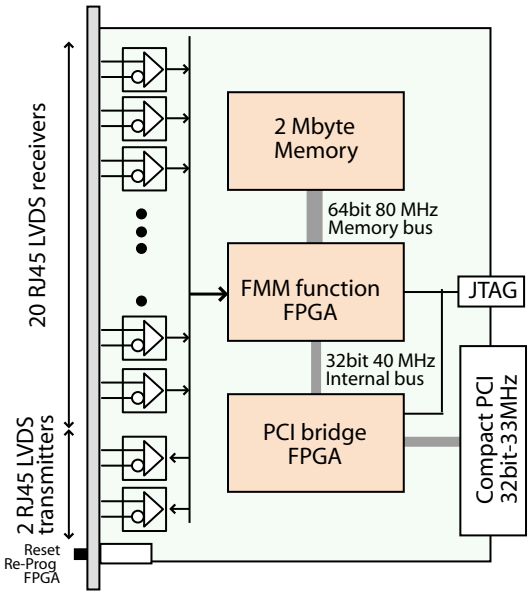

Figure 9.6: Block diagram of the FMM.

Figure 9.7: Photograph of a FMM module.

via the TTC system. These fast TTS signals are transmitted over shielded RJ-45 cables with four twisted pairs using the LVDS signaling standard.

The Fast Merging Module

For flexibility, FEDs are grouped into 32 TTC partitions which may be operated independently of each other. The Level-1 Trigger Control System separately distributes triggers to these 32 TTC partitions and separately receives trigger throttling signals for each TTC partition. TTS signals from all FEDs in a TTC partition thus need to be merged with low latency. Dedicated Fast Merging Modules (FMMs), have been designed for this task. These modules can merge and monitor up to 20 inputs and have quad outputs. Optionally, FMMs can be configured to merge two independent groups of 10 inputs with two independent twin outputs. For partitions with more than 20 FEDs, FMMs are cascaded in two layers.

The FMM is a custom-built 6U compact-PCI card (figures 9.6 and 9.7). It has three main components: a PCI Interface FPGA, a main logic FPGA and an on-board memory block. The 80 MHz internal clock of the FMM is not synchronized to the LHC clock. Input signals are synchronized to the internal clock by requiring two successive samples in the same state. The input signals are then merged by selecting the highest priority input signal from the enabled inputs according to the signal priorities listed above. Optionally, Out-of-Sync input signals are only taken into account if the number of inputs in Out-of-Sync state exceeds a programmable threshold.

The FMM also provides extensive monitoring capabilities in order to diagnose the causes for dead-times. Each state transition at the inputs is detected and stored with a time-stamp (25 ns resolution) in a circular buffer memory that can hold up to 128 k transitions. The times spent in the states Warning and Busy are counted with 25 ns resolution for each input channel and for the output(s).

2008 JINST 3 S08004

FMM cards are configured, controlled and monitored by a PC via a compact-PCI interface (StarFabric CPCI/PCI system expansion board from Hartmann Elektronik). The total system comprises 8 FMM crates with up to 9 FMMs in each crate. A total of 60 FMM modules are needed in order to merge the TTS signals of all TTC partitions of CMS.

9.3 Testing

In order to test and commission the central DAQ system independently of the GTP and of the subdetector DAQ systems, a number of additional components have been developed. These are not used in standard data taking.

The Global Trigger Processor emulator (GTPe) [205] emulates the functionality of the GTP (see figure 9.2). It reproduces the LHC beam structure, generates random or clocked triggers up to 4 MHz, respects the trigger rules, applies partitioning, transmits the GTPe data fragment over S-Link64 and receives and handles sTTS and aTTS backpressure signals. The hardware implementation is based on the Generic-III PCI card [206] and an interface module GTPe-IO.

In normal data taking mode, triggers from the global trigger are distributed to the FEDs via the TTC. When using the GTPe, special test triggers are sent directly to the FRL crates via a lemo cable to a trigger distributer card which distributes the trigger over the backplane to all the FRLs in the crate. Because the FEDs are not being used in this mode, busy signals from the FRLs are collected by the trigger distributer card and sent to a dedicated set of FMM modules for fast merging of this subset of sTTS signals. A dedicated mode of the FRL firmware handles the GTPe test triggers and instead of reading out the FEDs, the FRL generates data fragments with sizes according to a predefined table. In this way, the full central DAQ system can be tested.

9.4 The Event Builder

A schematic view of the Event Builder system is shown in figure 9.8. The event builder is composed of two stages: the FED-builder and the RU-builder. Each of the ≈512 FRLs generate event fragments with an average size of ≈2 kByte and the FED-builder is in charge of transporting these fragments to the surface building (SCX) and assembling them into 72 super-fragments with an average size of ≈16 kByte. The super-fragments are then stored in large buffers in Read-out Units (RU), waiting for the second stage of event building (RU-builder), which is implemented with multiple 72×72 networks. There can be up to 8 RU-builders, or DAQ slices, connected to the FED-builder layer. Each FED-builder is in charge of distributing the super-fragments, on an event by event basis, to the RU-builders and ensures that all super-fragments corresponding to one event go to one and only one DAQ slice, and are read by one Builder Unit (BU) of the RU-builder network. The complete event is then transferred to a single unit of the Event Filter. By utilising an architecture with two-stage event building, the full size system can be progressively deployed slice by slice while the traffic load to the second stage is optimized. The event builder is implemented with commodity equipment, including processing nodes, network switches and network interfaces. The processing nodes are server PC's with PCI busses to host the NICs. They run the Linux operating system.

2008 JINST 3 S08004

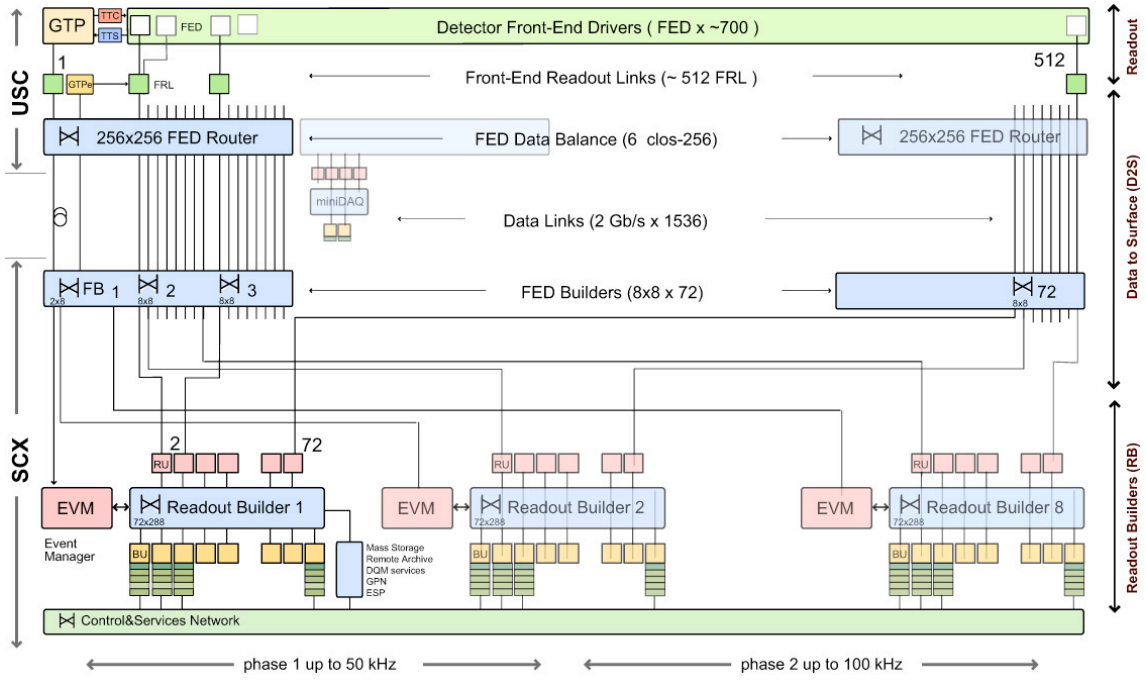

Figure 9.8: Schematic view of the Event Builder.

The event builder is lossless and when necessary back-pressure is propagated up to the FRL and subsequently to the FED, which can throttle the trigger via the TTS.

The FED-Builder stage

The FED Builder is based on Myrinet [209], an interconnect technology for clusters, composed of crossbar switches and NICs, connected by point to point bi-directional links. It employs wormhole routing and flow control at the link level. The LANai ASIC on the NIC contains an embedded RISC processor.

The FED-builder stage is implemented using multiple $N \times M$ FED-builders. In the baseline configuration, $N \leq 8$ and M equals the number of DAQ slices, which is 8 for the full system. In general, one $N \times M$ FED-builder comprises N input NICs hosted by the FRL, M output NICs hosted by the RU PCs, and two independent $N \times M$ switches to form 2 rails (figure 9.9). The NIC (M3F2-PCIXE-2 based on the LANai2XP) has two bi-directional fibre optic ports, with 2 Gbit/s data rate each. Each rail of one NIC is connected to an independent switch. In practice, instead of a large amount of small physical switches, a single large switching fabric is used per rail (see below). This 2-rail configuration doubles the bandwidth to 4 Gbit/s per FRL and provides redundancy.

The software running on the NICs has been custom developed in the C language. The FED-builder input NICs are programmed to read fragments from the FRL and to send them to the switch fabric, with a destination assigned on the basis of the fragment event number and a predefined look-up table. The FED-builder output NICs are hosted by the RU-builder PCs. They are programmed to concatenate fragments with the same event number from all the connected input cards,

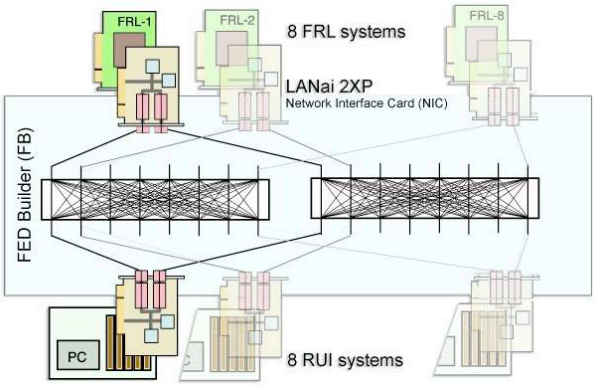

Figure 9.9: 8×8 FED builder with a two-rail network.

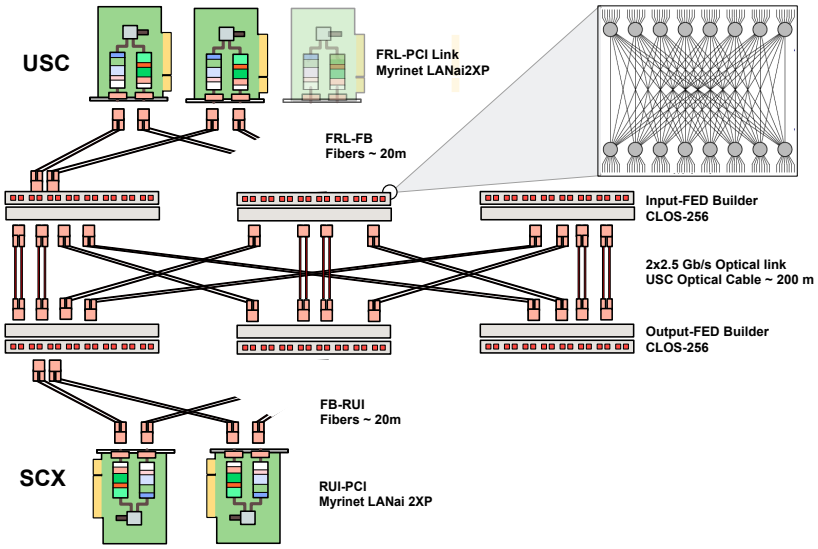

Figure 9.10: The FED builder stage switching fabric (only 1 rail is shown).

in order to build the super-fragments. The FED-builder is lossless due to a basic flow control and retransmission protocol, implemented on the RISC processor on the NIC.

The physical switching fabric is composed of six Clos-256 enclosures per rail (figure 9.10). A Clos-256 enclosure is internally composed of two layers of 16×16 (Xbar32) cross-bars. Three Clos-256 enclosures are located in the underground electronics room (USC), while the other three are in the surface DAQ building (SCX). They are connected by 768 200 m optical fiber pairs per rail, bundled in 12 cables. The Clos-256 enclosures are partly populated with linecards. Currently, the system has ports to accommodate a total of 576 FRLs and 576 RU PCs. In the baseline it is configured as 72 times 8×8 FED-builders. The use of a large switching fabric allows for a high

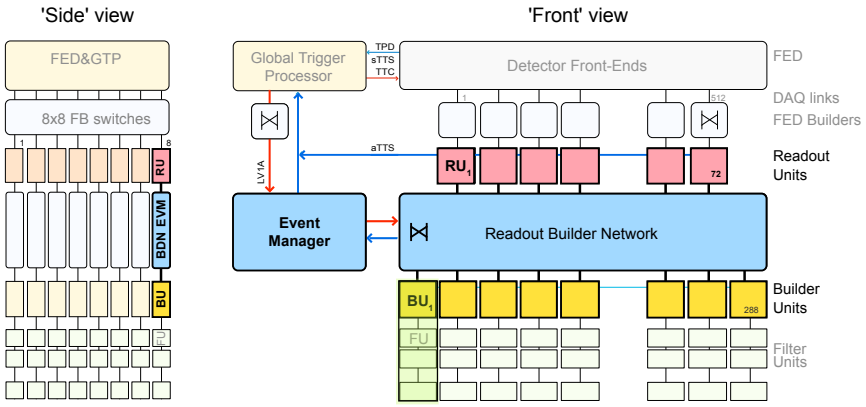

Figure 9.11: A Read-out Builder slice.

re-configurability of the FED-builders in software, which enables traffic balancing by redefining the super-fragment composition and traffic rerouting in case of a hardware failure.

The Myrinet NIC can transfer 4.0 Gbit/s data rate over the two optical rails. For random traffic the network efficiency is approximately 60%, due to head-of-line blocking. An event building throughput of about 300 MBbyte/s per node is achieved for variable sized fragments with a nominal average of 2 kBytes by using two rails [210]. Hence, the sustained aggregate throughput through the FED-builder stage is ≈1.4 Tbit/s, satisfying the CMS DAQ requirements. A maximum peak throughput of ≈2 Tbit/s is possible, if fully exploiting the FRL and Myrinet bandwidth, and using traffic shaping in the FED-builders.

The RU-builder stage

The RU-builder stage assembles super-fragments into complete events. Each RU-builder must collect event superfragments of average size ≈16 kByte from 72 data sources and build them into complete events at a rate of 12.5 kHz. A diagram of one slice of the RU-builder, indicating its various components is shown in figure 9.11. A RU-builder is made up of a number of Read-out Units (RU), Builder Units (BU) and a single Event Manager (EVM) connected together by a switching network. The EVM supervises the data flow in the RU-builder and receives a data record from the GTP via a dedicated FED-builder. The EVM allocates events on request to a BU, which subsequently collects the super-fragments from all RUs. From the BUs the complete events are sent to the Filter Units (FU).

In the baseline configuration the number of RUs per RU-builder is 72 and there is a factor of 4 more FU nodes. The RU, BU and EVM software components can be distributed in various ways on the PC nodes. In the baseline, there are two layers of PCs: RUs and BU-FUs. Here, the events are built by the BU software component in the same PC that runs the filter unit (FU) software, referred to as BU-FU. The RU nodes are server PCs with two 2 GHz dual-core Xeon processors (e5120) and 4 GByte of main memory (Dell PowerEdge 2950). They host a Myrinet NIC (M3F2-PCIXE-2) for the input from the FED-builder and a 4-port GbE NIC (PEG4I-ROHS Quad port copper Gigabit

Ethernet PCI Express Server Adapter from Silicom Ltd.). In the baseline, two links of the 4-port NIC are cabled to the network switch, which is sufficient to satisfy the throughput requirement of at least 200 MByte/s per RU node. A single ethernet link for each BU-FU node is sufficient, as the required throughput is 50 MByte/s. The EVB switching network is implemented with eight E-1200 switches (Terascale E-1200 router from Force10), one per DAQ-slice.

The RU-builder is based on TCP/IP over Gigabit Ethernet. The design choice of TCP/IP over Gigabit Ethernet has the advantage of using standard hardware and software. When operating an event builder over Ethernet close to wire speed, typically packet loss occurs because hardware flow control is not propagated from end-point to end-point through the switches. TCP/IP provides a reliable transport service that removes the need for the event building application to detect and deal with lost packets. It also provides flow control and congestion control. TCP/IP uses a substantial amount of host resources. Roughly 20% of a 2 GHz Xeon CPU core is required to transmit 1 Gbit/s, when using jumbo-frames (MTU=9000 Bytes).

For the event builder with 2 Ethernet links per RU node, a throughput of ≈240 MByte/s per RU node has been achieved at the nominal super-fragment size of 16 kBytes [211]. This can be increased, if needed for higher trigger rate or larger event sizes, to ≈360 MByte/s per RU node by installing a third Ethernet link per RU node. At a nominal throughput of 60 MByte/s, corresponding to 1/4 of the RU throughput, the event building tasks consume roughly 10% of the CPU resources on the BU-FU event filter nodes.

9.5 The Event Filter

The primary goal of the Event Filter complex is to reduce the nominal Level-1 Trigger accept rate of 100 kHz to a manageable level for the mass storage and offline processing systems while preserving interesting physics events and not introducing additional experiment dead-time.

The Event Filter complex:

- collects events accepted by the Level-1 Trigger system from the Event Builder and distributes them to worker nodes for further processing;
- performs basic consistency checks on the event data;
- runs offline-quality reconstruction modules and filters to process and select events for storage (High Level Trigger, “HLT”);
- generates, collects, and distributes Data Quality Monitoring (DQM) information resulting from online event processing in the HLT;
- serves a subset of the events to local and remote online consumers (EventServer, ES) for calibration and DQM;
- routes selected events to local storage in several online streams according to their trigger configuration;
- transfers data from local storage at the CMS site to mass storage in the CERN data centre at the Meyrin site.

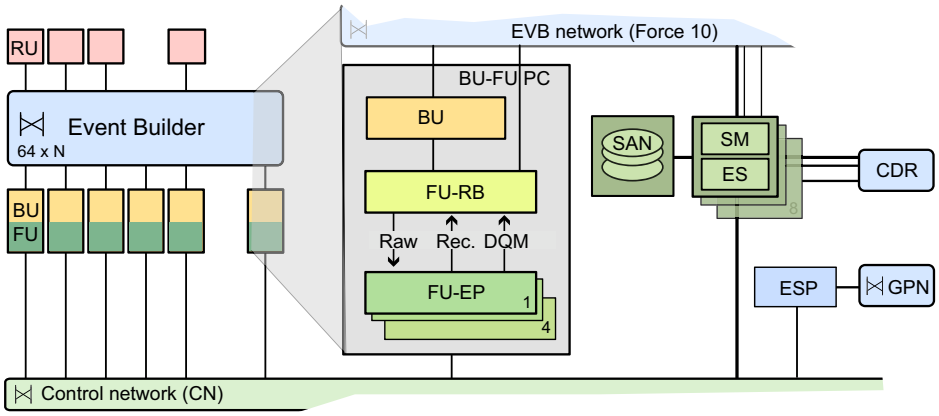

Figure 9.12: Architecture and data flow of the Filter Farm.

Architecture and data flow

The architecture of the CMS Event Filter is schematically illustrated in figure 9.12. The Event Filter hardware consists of a large farm of processors (on the order of 1000), running the HLT selection (Filter Farm), and a data logging system connected to a Storage Area Network (SAN). The Builder Unit (BU), belonging to the event builder, delivers complete events to one of multiple copies of the Filter Unit Event Processors (FU-EP) via the Filter Unit Resource Broker (FU-RB). A logically separate switch fabric provides the connectivity from the Filter Units to the data logging nodes. These data logging nodes are connected to a Fibre-Channel SAN, that is capable of a peak bandwidth of 1 GByte/s and has a capacity of several hundred TBytes.

The filter farm is logically subdivided into groups of processing nodes (Builder/Filter Unit, BU-FU). Each BU-FU hosts an identical set of software modules in a distributed environment based on the XDAQ online framework (section 9.7). As mentioned above, there are three separate applications, the Builder Unit (BU), the Filter Unit "Resource Broker" (RB) and the Event Processor (EP). The RB is responsible for managing memory resources for input and output to/from the HLT processes, and the communication with the Event Builder and the data logger. A complete event is handed by the RB, upon request, to the Event Processor (EP). The EP process uses the CMS reconstruction framework (CMSSW) [212] to steer the execution of reconstruction and selection code forming the HLT selection. Multiple EP processes can coexist in a single processor to provide concurrent execution and thus saturate the processing CPU resources.

Event processing

The EP reconstruction and selection algorithms are configured at the start of each data-taking run, by a configuration management system based on Oracle and working under run control supervision. The reconstruction, selection, and analysis modules specified in the configuration are instructed to obtain calibration constants and other time-dependent information from an Oracle database using standard access methods supported by the reconstruction framework.

Each reconstruction or selection algorithm runs in a predefined sequence, starting from raw data unpacking modules, which deal with sub-detector specific raw-data formats. The binary raw-data is formatted into C++ objects associated with sub-detector channels through a channel map using the FED block identifiers. Reconstruction modules can attach reconstructed objects to the event data structure. A full history of the execution of the HLT is attached to each accepted event. In addition, bookkeeping information is maintained by each EP process, and periodically collected by a supervising system to provide full accounting of events accepted and rejected by each HLT path. When a decision is reached, accepted events, comprising raw data and event reconstruction information produced by the HLT, are handed back to the RB for transfer to the data logging process.

Monitoring

The operation of unpacking and reconstruction modules running in the Event Processors is monitored using the Physics and Data Quality Monitoring infrastructure (DQM). Additional analysis modules may be executed in the process, outside the normal selection sequences, to provide fast feedback about the quality of the data using the same infrastructure. Information collected by the DQM on the individual FU nodes is periodically forwarded to the data logging system via the RB, providing a DQM Collector functionality (DQMC).

Data logging and Event Server

Events accepted for storage by one of the EP processes are transmitted to the RB, which forwards them to the Storage Manager (SM) process running in the data logger nodes via the data logging network. The SM is responsible of managing the various online streams to provide an appropriate granularity of event data for transfer, processing and bookkeeping. The data logger supports both disk streams for physics or calibration data, and network streams for the usage of consumer processes carrying out calibration or monitoring tasks. The network streams are created "on demand" by a consumer process connecting to the EventServer (ES) function of each SM. In normal operation with multiple SMs, the ES is multiplexed across the various sub-farm SMs by a caching Event Server Proxy (ESP). File and network streams can deal transparently with event data or DQM data.

Each data logger node hosting a SM is responsible for direct management of its disk pool in the storage area network. This includes correct interleaving of write transactions of the SM and data transfer, via the Central Data Recording (CDR) network to the offline systems for analysis and distribution to remote sites.

9.6 Networking and Computing Infrastructure

Networking Infrastructure

The general networking infrastructure of the experiment is based on Ethernet and is separated into:

- CMS Technical Network (CMS-TN);
- CERN general purpose network (CERN-GPN);

- Connection to the Central Data Recording (CDR);
- LHC technical network (LHC-TN).

These four networks are interconnected. In addition there are the dedicated DAQ event building networks (Myrinet and Ethernet) and dedicated networks for equipment control which have no connection to any of those four.

The CMS-TN is a general purpose network connecting all machines directly related to the operation of the experiment. It is used for system administration of all computers and for configuration, control and monitoring of all online applications. This network is not accessible directly from outside, but can be reached through dual-homed Application Gateway (AG) machines, which have one connection to CMS-TN and one to CERN-GPN switches. The CMS-TN is implemented as a distributed backbone with two routers located in SCX, and two in USC. Typically all computers in a rack are served by a switch in that rack. Each switch has two Gigabit Ethernet uplinks to the backbone routers. The desktop computers in the control room are also connected to the CMS-TN.

The CERN-GPN is the CERN-site network. The GPN will be used at all places to connect visitors (wired or wireless), who will use the application gateways to connect to the CMS-TN. This network will typically provide 1 Gbit/s connections to the GPN backbone.

The CDR connects to the Tier0 (chapter 11) system at the CERN Meyrin site using a minimum of 10 Gbit/s. A set of 8 triple-homed servers (one connection on the DAQ, one to CERN-TN, one on the CDR switches) are dedicated to this task. These are the Storage Manager nodes.

The LHC-TN allows data exchange between some of the CMS equipment and the LHC controls. The CMS-TN interconnects with the CERN-GPN and the LHC-TN using a filter implemented in the backbone routers.

Computing infrastructure

As previously discussed, the DAQ system comprises thousands of computing nodes. These are all rack mounted PCs. All PCs are Intel x86 based, running the CERN distribution of the Redhat Linux OS. The PC cluster includes a global file server and other services to be able to operate independently from the CERN-GPN. System installation is done with the Quattor toolkit [213]. In addition around a hundred PCs are used for DCS, running Microsoft Windows.

Database services are provided by a 6-node Oracle Real Application Cluster.

9.7 DAQ software, control and monitor

As stated previously, the CMS DAQ is designed in a way such that its hardware implementation can be staged as the LHC accelerator luminosity increases as well as the experiment's need for higher throughput. Thus the CMS DAQ online software must be highly scalable and must also support a diverse hardware base. The online software encompasses a distributed processing environment, data acquisition components, the run control and the detector control system. All subsystems of the DAQ have adopted the central online software frameworks with the philosophy of using common and standardized software technologies in order to reduce the effort associated with the maintenance and evolution of the detector read-out system over the long lifetime of the experiment.

2008 JINST 3 S08004

XDAQ Framework

The XDAQ (Cross-Platform DAQ Framework) framework [214] is designed for the development of distributed data acquisition systems. XDAQ includes a distributed processing environment called the "executive" that provides applications with the necessary functions for communication, configuration control and monitoring. Written entirely in C++, it provides applications with efficient, asynchronous communication in a platform independent way, including the use of memory pools for fast and predictable buffer allocation, support for zero-copy operation and a dispatching mechanism for an event-driven processing scheme. A copy of the executive process runs on every processing node in the data acquisition network.

XDAQ Applications are modeled according to a software component model [216] and follow a prescribed interface. They are compiled and the object code is loaded and configured dynamically at run-time into a running executive using the XML schema [221]. Multiple application components, even of the same application class, may coexist in a single executive process.

All configuration, control and monitoring can be performed through the SOAP/http [217] protocol, widely used in Web enabled applications. A rich set of data structures, including lists, vectors and histograms are exportable and can be inspected by clients through the executive SOAP services. Additional utility components provide support for hardware and database access.

XDAQ Applications and Libraries

XDAQ components [219] developed for CMS include applications such as the distributed Event Builder (EVB), sub-detector electronics configuration and monitoring components (FEC and FED), and central DAQ applications.

The generic event builder application consists of the main XDAQ components: a read-out unit (RU), a builder unit (BU) and an event manager (EVM). Data that are recorded by custom read-out devices are forwarded to the read-out unit application. A RU buffers data from subsequent single events until it receives a control message to forward a specific event fragment to a BU. A BU collects the event fragments belonging to a single event from all the RUs and combines them into a complete event. The BU provides an interface to the event data processors that apply event-filtering algorithms and provide data persistency (section 9.5). The EVM interfaces to the trigger read-out electronics and so controls the event building process by mediating control messages between RUs and BUs. For efficient transmission of binary (i.e. event) data the I2O specification [218] is followed. The event builder is a generic application that can run on a wide range of underlying hardware and is also used in local data acquisition systems, such as sub-detector test beams [215].

Data transmission in the XDAQ programming environment is carried out by special application components named peer transports. Peer transports register themselves with the XDAQ executive as being capable of resolving addresses as well as transmitting and receiving data. Communication between XDAQ applications is then accomplished by using an executive function that, when invoked, redirects the outgoing message to the proper peer-transport that in turn delivers the data over the associated medium. In this way the framework is independent of any transport protocol or network and can be extended at any time to accommodate newly appearing communication technologies. Baseline peer transports have been implemented for efficient binary data transmission using an asynchronous TCP/IP protocol and for simple message handling using the

2008 JINST 3 S08004

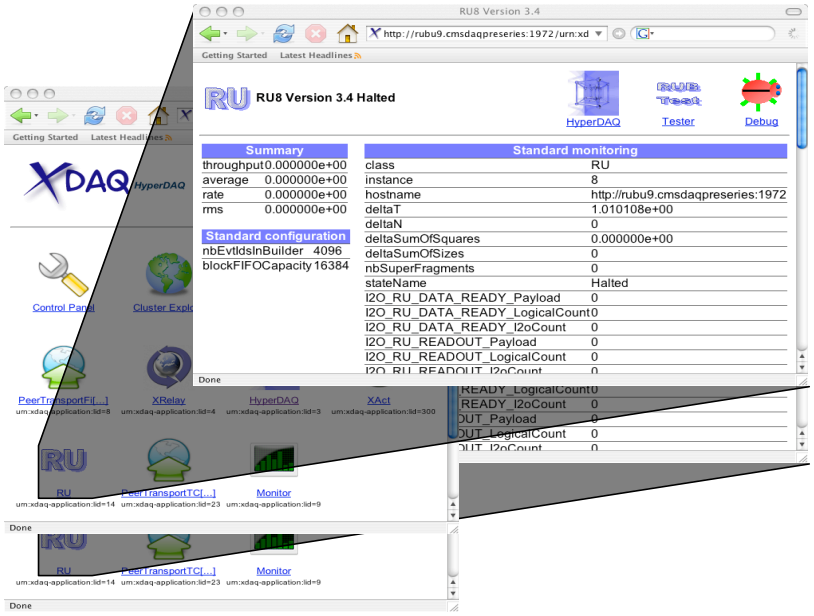

Figure 9.13: Example HyperDAQ page. Clicking on the RU application in the HyperDAQ page brings up monitoring information for the Read-out Unit application.

SOAP XML message format, however the framework is independent of the peer transport used, so optimisation of this layer is transparent to the rest of the XDAQ applications.

Libraries and device drivers have been developed that provide generic user access to VME and PCI modules and support Myrinet. Additional XDAQ components include support for the CMS custom front end devices, persistent monitoring message support and a gateway application to interface the XDAQ SOAP environment to the Detector Control System (section 9.8).

HyperDAQ

An extension to the XDAQ framework, HyperDAQ [220], exploits the http protocol which creates an entry point into XDAQ executives. A combination of HyperDAQ and Peer-to-Peer technology, used to discover new XDAQ applications providing data content, presents to the user links to the data content providers as they become available (figure 9.13). In this way, any node in the distributed online cluster can become an access point from which the entire distributed system can be explored by navigating from one application to another as the links become available. The HyperDAQ system has proved to be invaluable in debugging and monitoring the DAQ during full system integration tests.

Run Control System

The Run Control System configures and controls the online applications of the DAQ components and interfaces to the Detector Control Systems. It is an interactive system furnishing diagnostic

2008 JINST 3 S08004

and status information and providing the entry point for control. There are O(10000) applications to manage, running on O(1000) PCs.

The Run Control structure is organized into eleven different sub-systems, with each sub-system corresponding to a sub-detector or self-contained component, e.g. the Hadron Calorimeter, central DAQ or global trigger. The Run Control and Monitoring System (RCMS) framework, provides a uniform API to common tasks like storage and retrieval of process configuration data, state-machine models for process control and access to the online monitoring system.

Run Control applications and services are implemented in Java as components of a common web application "RCMS" provided by the framework. The Run Control is designed as a scalable and distributed system to run on multiple machines, thus the system can be easily expanded by adding additional hardware.

In RCMS the basic unit is the Function Manager (section 9.7). The interfaces are specified with the Web Service Description Language (WSDL) using the Axis [222] implementation of Web Services (WS). Various Web Service clients including Java, LabView and Perl have been implemented to provide access to the Run Control services. The publicly available official reference implementation of the Java Servlet technology Tomcat [223], by the Apache Software Foundation, has been chosen as the platform to run the Run Control web-applications. For persistency both Oracle and MySQL are supported by the RCMS framework.

For the baseline DAQ system, ten PCs running Linux are sufficient to control the experiment. A special copy of the XDAQ executive, the job control, is always running on each online node to accept SOAP commands from the run control to start and configure additional XDAQ executives. One common database (Oracle) is shared by all online processes and RCMS installations. Configuration management across sub-systems is achieved using global configuration keys (section 9.7).

The services and tools provided by RCMS comprise:

- Function Manager Framework;
- Resource and Account Management Services;
- Configurator;
- Log Message Collector.

In the following a few key components of Run Control are discussed.

Function Manager

A hierarchical control tree, with a central controller on the top and one or more controllers for each sub-system, structures the flow of commands and state information. The controllers are written using a common design paradigm, the so-called "Function Manager" (FM).

The FM has available a finite state machine, an interface for remote invocation, and a set of services to start, configure and control remote processes and to access configuration information from a DBMS. The FM is the basic element in the control tree. A standardized state machine model has been adopted by the sub-system for the first level of FMs which are directly steered by the central controller in the control tree (figure 9.14).

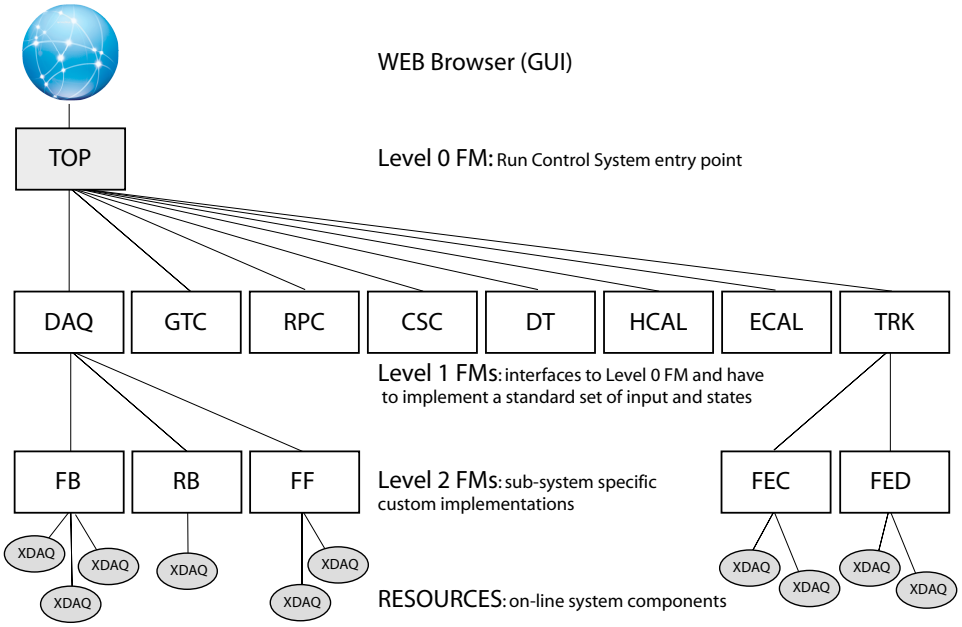

Figure 9.14: The RC hierarchy showing the full DAQ system. The Top Function Manager controls the next layer (Level 1) of Function Managers who in turn control the Level 2 (sub-detector level) Function Managers. The sub-detector Function Managers are responsible for managing the online system component resources.

Resource and Account Management Services

The Resource Service (RS) stores all information necessary to start and configure the online processes of the DAQ and sub-detectors. The data is represented as Java objects which are made persistent in the DBMS both as blobs and optionally as relational tables. RCS views the experiment as a collection of configurations, where a configuration is one or more groups of resources and one or more function manager implementations for control. The configuration is specific to each sub-system. Each sub-system has its own schema instance of the RS in the DBMS. The resource definition of a run is then the set of configurations of all participating sub-systems. The configuration of a given sub-system is resolved via a key mechanism. The sub-systems register a "configuration key" to a given global key identifying the configuration of the global run. All changes to configurations and global keys are versioned and trackable.

Users have to be authenticated to get access to the RCS resources. The resource service manages the configurations based on RCMS user accounts. Multiple configurations by multiple users can be run simultaneously in the same instance of the RCMS web-application.

Configurator

In order to create central DAQ configurations for different data taking scenarios, the CMS DAQ configurator application has been developed. The configurations can be tailored for reading out

2008 JINST 3 S08004

specific sub-sets of FEDs with specific throughput requirements, using different FED-Builder configurations and different sub-sets of the available event builder hardware. The configurations can be adapted to different versions and parameter settings of the online software components. The process of parametrising tens of thousands applications on thousands of hosts is largely simplified by factorizing the structure of the DAQ system and of the software settings. The structural representations of the DAQ configurations are stored in the CMS DAQ hardware and configuration database which also holds a representation of the available hardware components and their connectivity. Templates of software parameters for all software components are stored in a separate software template database. The CMS DAQ configurator application reads from both databases. It automatically calculates application parameters such as those depending on the connectivity information of the underlying hardware and creates the Java objects of the Resource Service. XML configuration files for the XDAQ executive processes are generated from these Java objects and stored in the Resource Service database. The CMS DAQ configurator application can also be used to generate configurations for test-bed hardware.

Log Message Collector

The Log Message Collector (LMC) is a web application to collect logging information from log4j and log4c compliant applications. The LMC has receiver modules for log4c messages used with C++ applications and log4j messages used with Java applications in XML and in binary format. Appender modules are implemented for TCP socket, TCP socket hub and JMS connections. Log messages can be stored on files with a File Appender, or in a DBMS with a DB Appender. Appenders can be active concurrently. Log messages are filtered by severity in the appender modules.

Each subsystem has its own instance of a LMC. A central LMC concentrates the messages of the subsystems and forms the entry point for the visualization client of messages, e.g. the Apache Chainsaw log message viewer.

9.8 Detector Control System

Function

The main purpose of the Detector Control System (DCS) is to ensure the correct operation of the CMS experiment, so that high quality data is taken with the apparatus. The scope of DCS includes all subsystems involved in the control and monitor of the detector, its active elements, the electronics on and off the detector and the overall environment.

The Detector Control Systems of individual sub-detectors are connected to the central DCS Supervisor (figure 9.15) for combined operation. These sub-detector DCS subsystems handle all the individual detector electronics such as the CAEN high-voltage power supplies and other electronics both commercial and custom made. The low-voltage system and the gas system are common for all sub-detectors whereas the cooling systems are built individually by each sub-detector. Additional components such as front-end detector read-out links are also monitored by the DCS.

The DCS provides both bookkeeping of detector parameters (table 9.2) and safety-related functions, including alarm handling and limiting the control of critical components via a software

2008 JINST 3 S08004

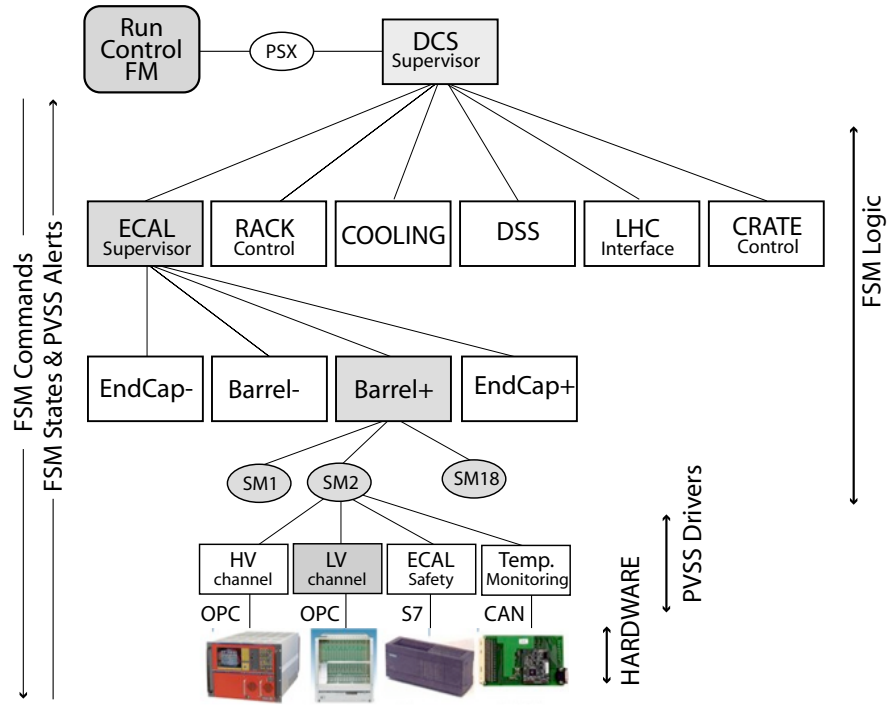

Figure 9.15: Outline of the Detector Control System hierarchy. Shown are all global services and ECAL as an example of a sub-detector control.

access control. The alarm handling and automated actions are designed in a way to anticipate major problems that would otherwise initiate Detector Safety System (DSS) actions, and warn the operator in advance that some action is needed. The alarm handling includes an SMS (mobile "text messaging") system that warns DCS users (for example a sub-detector expert) about abnormal system parameters. These SMS messages may require an acknowledgment by replying to the received alert SMS, and the status of both the alerts and acknowledgment's is displayed in the control room so that the operators in the control room are aware that experts are investigating the alarms. The DCS also collects relevant information from DSS. Monitoring of DCS parameters is possible via the Oracle Portal web pages that allow users to analyse both real time and archived data.

The DCS has to communicate with external entities as well, in particular the DAQ run control, and serves as the interface between the CMS experiment and the LHC accelerator. Many of the features provided by the DCS are needed at all times, and as a result selected parts of the DCS must function continually on a 24-hour basis during the entire year. To ensure this continuity UPS and redundant software and hardware systems are implemented in critical areas, however even non-critical nodes can be recovered in the order of minutes thanks to a CMS specific automated software recovery system.

2008 JINST 3 S08004

Table 9.2: Summary of detector parameters that are specific to each sub-detector.

Sub-detector	Monitored Parameters	Drivers	PCs / PLCs
muon CSCs	HV: 11 k channels, 218 k params LV: 8 k params Peripheral crate controller: 24 k params	Wiener and CAEN OPC custom: HV controller, Peripheral crate ctr CANopen-ELMB OPC	16
muon DTs	HV: 15 k channels - 110 k params LV: 1100 channels - 10 k parameters trigger: ≈12 k params	CAEN OPC CAEN OPC custom	7 5
muon RPCs	HV: 1200 channels LV: 1400 channels sensors: 500 channels Front end: 50 k params	CAEN OPC CAEN OPC same as LV custom	2 2
HCAL	HV: 450 channels, 3500 params LV: 270 channels, ≈1600 params HSS: 220 params Front end: 21 k params	custom CAEN OPC custom	
ECAL	LV: ≈4500 params HV: ≈2600 params Cooling: ≈200 params FE monitoring: ≈80 k params ESS: ≈600 params PTM/HM: ≈1 k params	CAN (Wiener) CAEN OPC Simatic S7, PSX custom: PSX Simatic S7 CANopen-ELMB OPC	3 4 2 / 1 1 1 / 2 2
Strip tracker	HV: 4 k channels LV: 4 k channels + 365 ctrl ch, 160 k params Temperature: 1100 sensors DCUs: 18 k channels, ≈ 100 k params	CAEN OPC, Siemens S7 custom: PSX	10 / 9
Pixel tracker	HV: 192 channels, 384 params LV: 192 channels, 384 params Temperature: 200 sensors	CAEN OPC, Siemens S7 custom: PSX	2 / 1
Alignment	LV: 200 channels 10 k LEDs, 150 lasers ≈2 k sensors	CAEN OPC ELMB, custom	5
Central DCS Services	rack control: 10 k params LHC interface: 1 k params cooling and ventilation DSS	Wiener OPC, CAN, SNMP custom: PSX	15

Implementation

The DCS software is based on the Siemens commercial SCADA package PVSS-II and the CERN Joint Controls Project (JCOP) framework [224]. Industrial controls hardware is interfaced by PVSS-II via various supported drivers OPC (OLE for Process Automation) [225] protocol, Siemens S7, SNMP, or Modbus. The JCOP framework provides common solutions to similar problems across all LHC experiments. This framework includes PVSS-II components to control and monitor the most commonly used commercial hardware (CAEN and Wiener) as well as control for additional hardware devices designed at CERN like the widely used ELMB (Embedded Local

Monitoring Board). It also provides a Data Interchange Protocol (DIP). For hardware not covered by JCOP, PVSS-II offers the possibility of implementing new drivers and components, and CMS has developed sub-detector specific software.

The control application behaviour of all sub-detectors and support services are modeled as Finite State Machine (FSM) nodes, using the FSM toolkit provided by the JCOP framework. The detector controls are organized in a tree-like FSM node hierarchy representing the logical structure of the detector, where commands flow down and states and alarms are propagated up (figure 9.15). The different control systems of the experiment have been integrated into a single control tree, whose top node is referred to as the CMS DCS Supervisor. CMS has put policies into place to ensure a homogeneous and coherent use of the DCS [226].

The DCS is a distributed system and comprises all control applications dedicated to sub-systems, communicating via the PVSS proprietary network protocol. In total there will be around 100 PCs with the majority of them running Microsoft Windows, although Linux is also supported.

PVSS-II includes a proprietary database that is used to store in real time the values of all the parameters defining the current state of the system (e.g. high-voltage settings, alarms, etc.). The configuration of PVSS-II itself is also stored in this database. For static and large amounts of data, an external Oracle database is used to store configuration data, and to archive measured values of parameters from PVSS to Oracle tables. Selected data from DCS is exported to the CMS conditions database, which contains all the data describing the detector environment needed for the offline reconstruction. The DCS access control system uses the LDAP and Oracle identity management tools which has web support for account management.

During normal physics data taking the DCS will act as a slave to run control and will therefore have to receive commands and send back status information. A communication mechanism between DCS and external entities is provided by the CMS specific PVSS SOAP interface (PSX). The PSX is a SOAP server implemented with XDAQ (section 9.7) using the PVSS native interface and JCOP framework, and allows access to the entire PVSS-II system via SOAP.

2008 JINST 3 S08004