Dynamic configuration of the CMS Data Acquisition cluster

G Bauer[1], U Behrens[2], K Biery[3], V Boyer[4,8], J Branson[5], E Cano[4], H Cheung[3], M Ciganek[4], S Cittolin[4], J A Coarasa[4,5], C Deldicque[4], E Dusinberre[5], S Erhan[6], F Fortes Rodrigues[7], D Gigi[4], F Glege[4], R Gomez-Reino[4,] J Gutleber[4], D Hatton[2], J F Laurens[4], J A Lopez Perez[4], F Meijers[4], E Meschi[4], A Meyer[2,4], R K Mommsen[3], R Moser[4,9], V O'Dell[3], A Oh[4,10], L B Orsini[4], V Patras[4], C Paus[1], A Petrucci[5], M Pieri[5], A Racz[4], H Sakulin[4,12], M Sani[5], P Schieferdecker[4,11], C Schwick[4], D Shpakov[3], S Simon[5], K Sumorok[1] and M. Zanetti[4]

[1]MIT, Cambridge, USA; [2]DESY, Hamburg, Germany; [3]FNAL, Chicago, USA; [4]CERN, Geneva, Switzerland; [5]UCSD, San Diego, USA; [6]UCLA, Los Angeles, USA; [7]CEFET/RJ, Brazil; [8]now at Open Bee, Annecy, France; [9]also at Technical University of Vienna, Vienna, Austria; [10]now at University of Manchester, Manchester, United Kingdom; [11]now at University of Karlsruhe, Karlsruhe, Germany

E-mail: Hannes.Sakulin@cern.ch

Abstract. The CMS Data Acquisition cluster, which runs around 10000 applications, is configured dynamically at run time. XML configuration documents determine what applications are executed on each node and over what networks these applications communicate. Through this mechanism the DAQ System may be adapted to the required performance, partitioned in order to perform (test-) runs in parallel, or re-structured in case of hardware faults. This paper presents the configuration procedure and the CMS DAQ Configurator tool, which is used to generate comprehensive configurations of the CMS DAQ system based on a high-level description given by the user. Using a database of configuration templates and a database containing a detailed model of hardware modules, data and control links, nodes and the network topology, the tool automatically determines which applications are needed, on which nodes they should run, and over which networks the event traffic will flow. The tool computes application parameters and generates the XML configuration documents and the configuration of the run-control system. The performance of the configuration procedure and the tool as well as operational experience during CMS commissioning and the first LHC runs are discussed.

1. Introduction

The Compact Muon Solenoid [1] (CMS) Experiment at CERN's new Large Hadron Collider will explore a wide range of physics at the TeV scale. With 55 million readout channels the average event size after zero suppression will be around 1 MB. In CMS, only the first trigger level is implemented using custom electronics. The algorithms of all further trigger levels are implemented as software

[12]To whom any correspondence should be addressed.

running on the DAQ cluster. The DAQ system [1] therefore has to build events at the first-level trigger accept rate of 100 kHz, which translates to a throughput of around 100 GB/s. The DAQ System contains both, custom electronics boards controlled by PCs and a farm of around 1500 PCs connected by commercial networking equipment. The present paper deals with the configuration of the software running on all the PCs in the DAQ cluster. After a short introduction to the DAQ System in section 2, the requirement of the configuration being flexible is discussed in Section 3. Section 4 explains the concept of dynamic configuration of the cluster through the Run Control System. A tool developed to quickly create diverse types of configurations for the central DAQ System – the CMS DAQ Configurator – is presented in section 5.

2. The CMS Data Acquisition System

The central DAQ system is composed of an input stage and two event-building stages (figure 1). Event data arrive at the DAQ System through around 650 Frontend Drivers (FEDs) which provide fragments of around 2 kB size, each. The input stage contains two types of custom electronics boards. The Front-end Readout Links (FRLs) interface the custom detector readout links coming from the FEDs to commercial Myrinet Network Interface Cards. The Fast Merging Modules (FMMs) merge fast feedback signals from the FEDs in order to provide throttling signals to the trigger. FMMs are arranged in a tree structure. Both types of custom electronics are controlled by PCs through a compact PCI interface. The event-building stages are implemented using commercial hardware: Myrinet is used to build super-fragments out of on average eight event fragments. These super-fragments are then distributed to a number of Event Builder/Filter slices (currently eight). These independently build events using multiple rails of 1 Gb/s Ethernet. In each slice, Readout Units buffer the event super-fragments and send them on to Builder Units, which assemble the full events. The Event Manager, which receives the event fragments from the Level-1 Trigger, manages the data flow in the Event Builder. The high-level trigger processes run on the same machines as the Builder Units. In typical configurations, one Builder Unit and seven High-Level Trigger applications are run on an eight-core machine. Selected events are sent to the Storage Manager application, which stores them to a local disk array and controls the transfer to the Tier-0 at CERN.

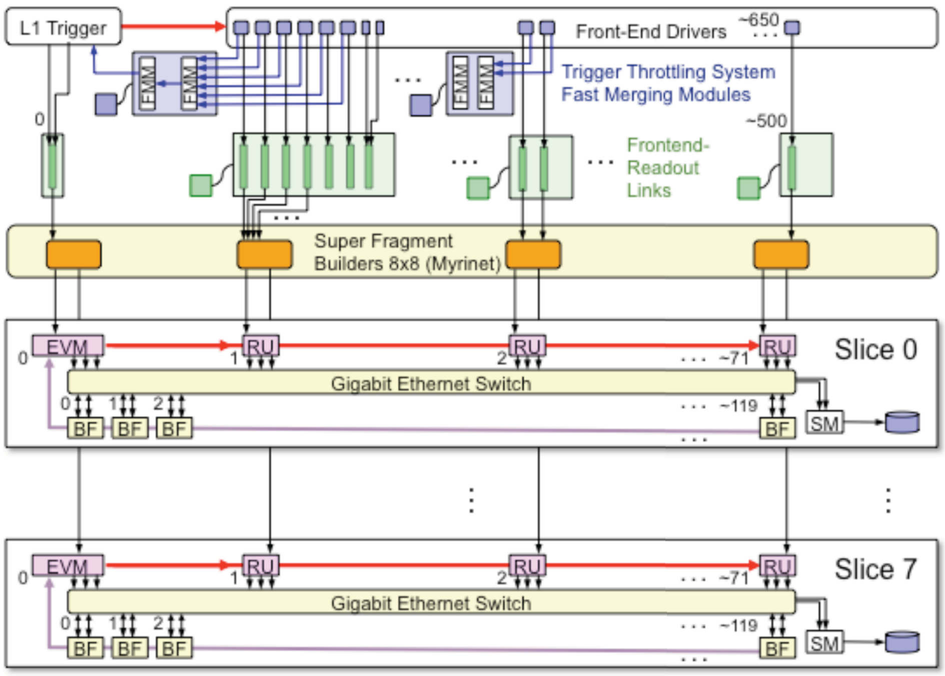

Figure 1. The CMS DAQ System. See text. (FMM: Fast Merging Module, EVM: Event Manager, RU: Readout Unit, BU: Builder Unit, SM: Storage Manager)

All software used to control custom hardware and the event building and filtering is implemented using the XDAQ [2] infrastructure (figure 2). The basic unit provided by XDAQ is an executive (process), which can run one or more XDAQ applications. The XDAQ infrastructure provides data transport protocols used in the event building and hardware access libraries used to control custom hardware. XDAQ executives are highly configurable through an XML description, which determines the libraries to be loaded, the applications to be instantiated, the application parameters and the network connections to collaborating applications. The Job Control service, itself a XDAQ application, runs on all machines in the cluster. Job Control takes care of starting XDAQ executives when instructed to via its SOAP[13] interface.

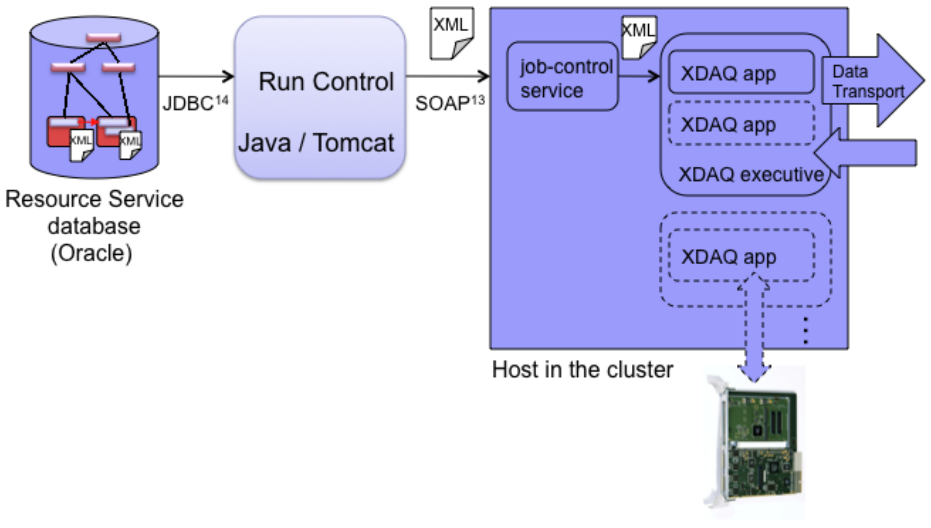

Figure 2. Dynamic configuration of the CMS data acquisition cluster through the Run Control System.

3. Configuration Requirements

The configuration of the DAQ system needs to be flexible in order to support

- Fault tolerance: Event building data flow needs to be re-routable in case of failure of hardware components. Redundancy is foreseen in most parts of the DAQ system but in general requires a change of configuration.
- Scalability: The structure of the DAQ configuration needs to be scalable in terms of number of slices, size of slices and number of front-ends to be read out. This is required in order to allow for staged deployment of hardware, maintenance of part of the hardware, test runs parallel to data taking and partitioned operation.
- Optimization of parameters: DAQ Configurations need to be re-computed in order to optimize parameters of DAQ System components.
- Evolution of software: Evolution of DAQ System software components frequently requires corresponding adjustments of DAQ configurations.

The required flexibility is achieved through two key ingredients:

- dynamic configuration of the cluster via the Run Control System (section 4) and
- a powerful tool to prepare DAQ configurations (section 5).

[13]Simple Object Access Protocol
[14]Java Database Connectivity

4. Dynamic Configuration via the Run Control System

The *Run Control System* [3] handles the configuration and control of the CMS central and sub-detector DAQ systems. It reads the configuration from the *Resource Service* database and starts and configures all processes (XDAQ executives) of the online-software. Through this procedure the DAQ systems may quickly be re-configured or even re-structured just before the start of a run, hence the name dynamic configuration. The dynamic configuration procedure described in this section is uniform across all CMS sub-systems, except for the Level-1 Trigger that uses a more static approach. The remainder of this paper however deals mainly with aspects and tools specific to the central DAQ system.

The Run Control System is a distributed Java application running on Apache Tomcat servlet containers. It consists of framework components and a set of loadable modules (*Function Managers*), which control parts of the DAQ system and can be arranged in a tree structure. At the start of a session or just before a run, the Run Control system loads the configuration of the Function Manager hierarchy and of all XDAQ executives from the Resource Service database. It loads and starts the needed Function Managers, which then start all the XDAQ executives on all hosts through the job-control services. Only at this point, the roles of each host in the system and the data flow topology are decided.

The *Resource Service* database is a relational database storing DAQ configurations. DAQ configurations are organized in a tree structure. Individual configurations are versioned. A configuration in the Resource Service contains all XDAQ executives and XDAQ applications with their parameters and all network connections between XDAQ applications. A Resource Service configuration also contains the configuration of the Run Control system, i.e. the hierarchy of Function Managers to be loaded in order to control the system. The entire configuration is stored in a relational schema. Additionally, the XML configuration files for each XDAQ executive are stored in the database as Character Large Objects. The Java API of the Resource Service loads an entire configuration into memory translating it into a structure of Java objects. In order to minimize the time needed to load a configuration these Java objects are cached in the database. Groups of the Java objects are serialized and stored in the relational schema as Binary Large Objects.

Recent performance measurements show an initialization time of 35 seconds for the complete central DAQ System at LHC start-up consisting of 1500 hosts running 6500 executives and 10000 applications. The Run Control System for the central DAQ system includes 5 Tomcat servers running around 50 Function Managers in total. This initialization time includes the loading of the configuration from the Resource Service, loading and initialization of the Function Managers in the Run Control system and starting of all of the XDAQ executives and applications. Further optimizations of the initialization procedure are underway at the time of writing.

It should be noted that additional steps are required in order to start a run that are not dealt with in this paper. These include opening of network connections, setting of masks, loading of the high-level trigger configuration and distribution of the run number.

5. Creating DAQ Configurations for the central DAQ System

The preparation of the configuration of thousands of applications clearly requires automation. An analysis of how configurations typically differ yields three aspects:

1. The composition of functional units (i.e. the applications that make up a single entity such as a Readout Unit or Builder Unit) typically stays constant or evolves slowly. Most of the application parameters either remain the same or develop slowly. The same is true for the network connections required between applications in different types of functional units. They are replicated using one of a small number of routing algorithms. Also, the control structure stays constant or evolves slowly. Parts of the control structure need to be replicated for each

slice to be controlled. All the slowly changing information is stored in *Configuration Templates*.

2. The location, multiplicity and connectivity of functional units change frequently. The user can define the location and connectivity of most functional units in the *High Level Configuration Layout*. The High Level Configuration Layout is a model of the DAQ System at a high level.
3. The functional units needed to control the custom electronics in the input stage of the DAQ system and their parameters change when the High Level Configuration Layout changes, but can be computed knowing the underlying structure of the DAQ system. Similarly, the configuration of the super-fragment building stage and the network connections used by each application depend on the High Level Configuration Layout, but can be computed algorithmically. The *CMS DAQ Configurator* tool, a standalone Java application, handles all the computations of the algorithms and the generation of the XML configuration documents.

The workflow of the Configurator is shown in figure 3. The key elements are explained in the following sub-sections.

5.1. Configuration Template Database

Configuration Templates are stored in the Configuration Template database. The structure of a Configuration Template follows the structure of a Resource Service configuration explained in section 4 with the difference that each functional unit is present only once. A Configuration Template contains the Template Control Structure, Template Functional Units including XDAQ executives and XDAQ applications with template parameters and template connections. The template control structure specifies which parts of it are to be replicated per slice. Template XDAQ applications contain fixed parameters and placeholder parameters that will be replaced with actual values from the High Level Configuration Layout or by computation. Template connections between template XDAQ applications specify what network(s) and protocol to use and what algorithm to use to replicate the connections globally within the configuration or within a slice of the configuration.

Configuration Templates may be parameterized using variables. Variables may depend on other variables that are declared before. Simple arithmetic expressions of variables with numeric content are evaluated. Variables may be declared in three places: in Site Settings, in Account Settings and in the Template itself (evaluated in this order). Site Settings and Account settings are sets of variables that are stored in the Configuration Template Database. The user may select any combination of a Site Setting and an Account setting and a Configuration Template when creating a configuration. Variables defined in the Template may be “exported” so that the user can modify them though ad-hoc input, just before creating a configuration. Through this mechanism the same Configuration Template may be used to create configurations for different situations, e.g. for the production system and for test systems, for different Run Control installations, or for testing and debugging runs.

Configuration Templates in the Configuration Template database are arranged in a tree structure. Configuration templates are stored in the database either by a standalone Java application (Filler) or by a graphical editor. Currently, the templates are stored in the database as serialized Java objects.

5.2. DAQ Hardware and Configuration Database

The CMS DAQ Hardware and Configuration database holds High Level Configuration Layouts and the underlying structure of the central DAQ System. The database is organized in three levels:

- *Equipment Sets* contain a description of all the equipment in a particular DAQ System. They include all custom electronic boards and their cabling to the Front-end Drivers, to the Myrinet Switches, to the Trigger and among each other. They also include all hosts and their cabling to the Myrinet Switches and to the Gigabit Ethernet data networks. The CMS DAQ Hardware and Configuration database holds equipment sets for the production system and various test systems. Equipment sets need to be updated, only when hardware or cabling are modified. Equipment sets are updated using a standalone administration tool.

- Based on an Equipment Set, a number of *Super-Fragment Builder Sets* can be defined. A Super-Fragment Builder Set defines the grouping of Front-end Drivers for the assembly of super-fragments. Super-Fragment Builder Sets usually remain constant over long periods of time. They may be updated, for example in order to balance super-fragment sizes or to add/remove sub-detectors. Super-Fragment Builder Sets are created using small standalone Java applications (Fillers).
- Based on a Super-Fragment Builder Set, a number of *DAQ Partition Sets* can be defined. A DAQ Partition Set defines the high-level layout of one or more DAQ partitions. For each DAQ Partition, the locations of the functional units in the Event Builder/Filter slices are defined. Readout Units are assigned to super-fragments. DAQ Partition Sets need to be updated more frequently in order to exclude hosts form the configuration or to scale the system size in order to allow for parallel activities on the cluster. DAQ Partition Sets are created using small standalone Java applications (Fillers). These Java fillers algorithmically assign Readout Units to super-fragments, taking into account the availability of readout units and the routing constraints of the Myrinet switch fabric.

All three types of set are stored in a relational schema. The three levels of sets are organized in a common tree structure. Any set in the CMS DAQ Hardware and Configuration Database can be uniquely addressed by its path in the tree, like in a file system. It is possible to insert directories at any level in the tree in order to group sets (for example by run period).

5.3. The CMS DAQ Configurator

The CMS DAQ Configurator is an interactive standalone Java web-start application. It reads Configuration Templates and High Level Configuration Layouts from the respective databases and creates comprehensive central DAQ configurations, which are then written to the Resource Service database (figure 3). The creation of a configuration involves the following steps:

1. Through the graphical user interface, the user selects a Configuration Template, a Site Setting and an Account Setting from the Configuration Template database. The user selects a DAQ Partition Set from the CMS DAQ Hardware and Configuration database and an output location in the Resource Service database. Optionally, exported parameters of the Configuration Template may be adjusted.
2. The CMS DAQ Configurator reads the Configuration Template, the Site Setting and the Account setting from the Configuration Template database, optionally takes into account ad-hoc user input and parameterizes the template.
3. The DAQ Partition Set and the underlying Super-Fragment Builder Set and Equipment Set are loaded from the CMS DAQ Hardware and Configuration database. Based on the connectivity information in the Equipment Set and on the high-level configuration layout in the other two sets, the CMS DAQ Configurator determines the functional units needed to control custom electronics and their configuration such as channel masks. Based on all three sets, the configuration of the Myrinet Super-Fragment Builder is computed. The output of this step is a generic high-level layout consisting of slices, units, sub units and parameters.
4. Template Functional Units from the Configuration Template are replicated for each matching functional unit in the high-level layout. Parameters from the high-level layout are substituted into placeholders in the template. A part of the template control structure is replicated for each slice.
5. The network connections between XDAQ applications are generated using one of several built-in routing algorithms. Currently there is a global routing algorithm for single-rail network connections and a per-slice routing algorithm for $n \times m$ rail connections, using n Gigabit Ethernet rails on the source and m Gigabit Ethernet rails on the destination applications.
6. For each XDAQ executive, an XML configuration document is created containing the applications of the executive and their parameters, all collaborating applications in other executives and all network connections.

7. The configuration may optionally be edited with a graphical editor.
8. The configuration is saved to the Resource Service database.

Creation of the complete configuration for the current central DAQ system as given in section 4 (step 2 to 6 and 8 in the above list) typically takes around 15 seconds when running the DAQ Configurator on an eight-core 2.66 GHz Xeon machine and using an Oracle RAC[15] system for all three databases. The time may be longer if database and/or CPU are under heavy load due to other parallel activities.
It is also possible to start the CMS DAQ Configurator in batch mode from the command line.

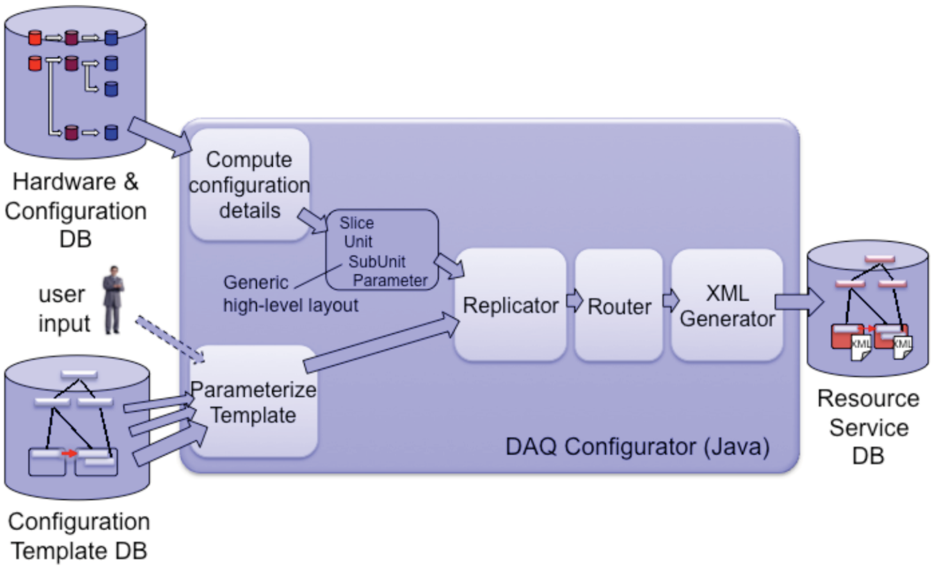

Figure 3. CMS DAQ Configurator workflow.

6. Conclusion

Dynamic configuration of the CMS Data Acquisition cluster through the Run Control System allows the CMS DAQ system to be quickly re-structured in order to respond to hardware faults or in order to allow for runs or maintenance to take place in parallel on part of the cluster. Processes on the nodes in the cluster are started and configured based on a configuration stored in the Resource Service database. The time to load a configuration and to start all processes has been optimized and is currently at around 35 seconds for typical configurations of the central DAQ system utilizing the entire cluster.
The CMS DAQ Configurator tool greatly simplifies the generation of central DAQ configurations, enabling a pool of on-call experts to create new configurations with a fast turn-around time of only several minutes. The user specifies the DAQ System layout providing only a minimal amount of high-level input. The CMS DAQ Configurator then combines the high-level layout with a configuration template. Configuration details such as channel masks, which in the past had to be calculated by the user, are automatically computed using pre-defined algorithms. The typical time to create DAQ configuration from a high-level layout and a configuration template and to store it in the Resource Service database is around 15 seconds. The CMS DAQ Configurator has successfully been in use since 2007 to create all DAQ configurations for DAQ test setups and for the production DAQ system. Its ease of use and the resulting fast turn-around time have been important in keeping dead-time low during cosmic data-taking campaigns and during the first phase of data-taking with the LHC.

[15]Real Application Cluster

Acknowledgements
This work was supported in part by DOE and NSF (USA) and the Marie Curie Program.

References

[1] The CMS Collaboration (Adolphi R *et al.*) 2008 The CMS Experiment at CERN LHC *JINST* **3** S08004 361
[2] Bauer G *et al.* 2009 The CMS Data Acquisition System Software *J. Phys. Conf. Ser.* (CHEP 09, Prague, Czech Republic, 23-27 Mar 2009) in print
[3] Bauer G *et al.* 2008 The run control and monitoring system of the CMS experiment *J. Phys. Conf. Ser.* (CHEP 07, Victoria, BC, Canada, 2-7 Sep 2007) **119** 022010