Monitoring the CMS Data Acquisition System

G Bauer, U Behrens, K Biery, J Branson, E Cano, H Cheung, M Ciganek, S Cittolin, J A Coarasa, C Del dicque, E Dusinberre, S Erhan, F Fortes Rodrigues, D Gigi, F Glege, R Gomez-Reino, J Gutleber, D Hatton, J F Laurens, J A Lopez Perez, F Meijers, E Meschi, A Meyer, R Mommansen, R Moser, V O’Dell, A Oh, L B Orsini, V Patras, C Paus, A Petrucci, M Pieri, A Racz, H Sakulin, M Sani, P Schieferdecker, C Schwick, D Shpakov, S Simon, K Sumorok and M. Zanetti

1MIT, Cambridge, USA; 2DESY, Hamburg, Germany; 3FNAL, Chicago, USA; 4UCSD, San Diego, USA; 5CERN, Geneva, Switzerland; 6UCLA, Los Angeles, USA; 7CEFET/RJ, Brazil; 8also at Technical University, Vienna, Austria

E-mail: Luciano.Orsini@cern.ch

Abstract. The CMS data acquisition system comprises O(20000) interdependent services that need to be monitored in near real-time. The ability to monitor a large number of distributed applications accurately and effectively is of paramount importance for robust operations. Application monitoring entails the collection of a large number of simple and composed values made available by the software components and hardware devices. A key aspect is that detection of deviations from a specified behaviour is supported in a timely manner, which is a prerequisite in order to take corrective actions efficiently. Given the size and time constraints of the CMS data acquisition system, efficient application monitoring is an interesting research problem. We propose an approach that uses the emerging paradigm of Web-service based eventing systems in combination with hierarchical data collection and load balancing. Scalability and efficiency are achieved by a decentralized architecture, splitting up data collections into regions of collections. An implementation following this scheme is deployed as the monitoring infrastructure of the CMS experiment at the Large Hadron Collider. All services in this distributed data acquisition system are providing standard web service interfaces via XML, SOAP and HTTP [15,22]. Continuing on this path we adopted WS-* standards implementing a monitoring system layered on top of the W3C standards stack. We designed a load-balanced publisher/subscriber system with the ability to include high-speed protocols [10,12] for efficient data transmission [11,13,14] and serving data in multiple data formats.
1. Introduction
Monitoring the CMS [1] data acquisition system spans all tasks needed to retrieve, collect and display information used to track the status and operation as well as the processing of errors and alarms in a uniform manner. The system is characterized by a large number of hosts and applications [18,19,20,21]. In addition to all traditional requirements that specify the monitoring tasks, scalability requirements are a key concern that pervades all aspects of the system design. Scaling requirements [1,3,5,8] along several dimensions are imposed on the on-line monitoring infrastructure:

Numerical scalability refers to the ability to seamlessly perform operations with an increased number of users, resources, and services.

Geographical scalability refers to the ability to perform the same identical function regardless of the physical resource location.

Administrative scalability is achieved if the system is managed in the same way even if it encompasses multiple administrative domains. This includes network boundaries, physical computers and mapping of applications to resources.

Functional scalability refers to the ability to accommodate additional functionality.

The proposed infrastructure fits these needs by providing a set of expandable and reusable solutions allowing use of the monitoring and alarming system for development, test and operation scenarios.

2. Architecture and design
The infrastructure is based on service-oriented architecture [2,3,4], in which a 3-tier structured collection of communicating components cooperates to perform the monitoring task. The universal

Figure 1. DaqMon (Labview). Layout of the running system with all nodes and their states, history and current status of data flow elements.
application connectivity, that makes every monitoring and application service inter-communicating is based on the XDAQ [5,6,7,8] middleware. As shown in Figure 2, the system builds upon a scalable publisher-subscriber service consisting of a pool of eventing applications orchestrated by a load balancer called a broker.

![Figure 2. Architecture.](image)

The DAQ applications act as data producers through sensor services to publish monitoring data. Similarly sentinel services are used to report errors and alarms. Other services for processing, storing, filtering and transforming the information express their interest by selectively subscribing to eventing services. Presentation components can either subscribe or directly retrieve monitoring data from the required provider services (An example of presentation is shown in Figure 1). All services are relocatable and run independently of each other without a need for external control. Communication among services is established through a rendezvous mechanism with the help of discovery services facilities [16]. The heartbeat service keeps track of all running services and DAQ applications.

3. Data Collection
This is the method by which data tuples defined for the data acquisition system are retrieved from the distributed applications, merged and made available in various standard formats. All metrics are treated, through the whole processing chain, using a uniform table based data format, as shown in Figure 3.

Table definitions enumerating all data items, called flashlists, are specified in XML. Flashlist specifications reliably identify the content for merging, tracking and analysis with additional information, including timestamps, version and application identification (URI, URL, UUID, IP and others) fields. The framework inserts these data fields transparently into the application software.
Data collection is initiated in either of two ways at the sources: push from the application or pull according to a configurable time period. Merging of distributed tables is performed in one or more steps by a service called a collector. A load-balanced pool of data collectors copes with the data traffic. The data so collected is served to user interface applications on request in JSON, XML, CSV and SunRPC binary format by the live access services over HTTP protocol.

4. Errors and Alarms

DAQ applications have the capability to asynchronously notify exceptional conditions using the same data format as the monitoring infrastructure. Two different scenarios can be identified: applications that detect persistent deviations from the normal system behavior can report errors or a deviation may also be transient, meaning an alarm is fired and eventually revoked when the asserted condition is resolved.
As shown in Figure 4, reporting errors and alarms is performed through sentinel plug-in services that take care of routing notifications, guaranteeing delivery and preventing the system from flooding. All reports are recorded by a persistency service called spotlight that keeps the history of all events, allowing a playback of all reports. Errors and alarms are visualized by a graphic web application [17] the hotspot facility (Figure 6) that maps them to the graphics according user defined models of the system.

Figure 5. Hotspot viewer.
The system model as shown in Figure 5 is an abstract view of the system. Several views can be defined and organized in hierarchical structures. The model so defined is used to categorize errors originating from the running system elements. In order to match errors reported by the running system with the abstract model a number of filters are defined. Filters are associated with views by means of special nodes called guards. The guards define regular expressions matching the group and tag attributes from the error report.

Exceptions raised by XDAQ applications contain information about the system configuration. Therefore, in addition to the specific exception information, exceptions contain an indication of the XDAQ application group and zones to which they belong.

![Figure 6](image)

**Figure 6.** Hotspot example of errors and alarms report according to two different perspectives of the system. Errors and alarms are associated to elements of the system model and displayed according to their severity levels. The tool offers different views of the model such as tree navigation, heat maps, tables and scrolling terminals.

5. **Benchmarks**

The two plots shown in Figure 7 below give scalability measurements for different system sizes in terms of total message rate and throughput. Increasing the size means adding slices starting from ~800 applications on ~150 computers to 5500 applications on 1000 computers. The standard deviation of the rate and throughput measurements grows with the system size.

The achieved performance allows running the system at the required update rate of 1 Hz for all data sources.

The current system collects about 20 different flashlists and all updated values can be synchronized within 1 second. The latency for each report depends on the number of collection steps. It has been measured to be within one second.
6. Summary
The CMS online monitoring system has been implemented and is currently used in an operational environment. This software product line [9] is the result of several years of development and has proven its fitness for operation with the acquisition of the first beam events on September 10, 2008.

This paper summarized key requirements and outlined the resulting architecture of the CMS online monitoring software infrastructure.

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

References
[1] Gutleber J, Murray S, Orsini L 2003 Towards a homogeneous architecture for high-energy physics data acquisition systems *Elsevier Comp. Phys. Comm.* 153(2) 155-163
[2] Gutleber J, Moser R, Orsini L 2007 *Data Acquisition in High Energy Physics* Proc. Astronomical Data Analysis Software and Systems (ADASS 23-26 Sept 2007) XVII, ASP vol. 394 pp 47
[3] CERN 2002 Data Acquisition & High-Level Trigger, Technical Design Report CMS TDR 6.2, LHCC 2002-26 (ISBN 92-9083-111-4)

[4] McIlroy M D 1968 Mass-produced software components in: Software Engineering Concepts and Techniques, eds. Buxton J M, Nauran P and Randell B, Reprinted proceedings of the 1968 and 1969 NATO Conferences, Petrocelli/Charter (ACM Press, 1076) pp 88-98.

[5] Parnas D L 1979 Designing Software for Ease of Extension and Contraction, IEEE Trans. Softw. Eng SE-5(2) 128-137

[6] Nierstrasz O, Gibbs S and Tsichritzis D 1992 Component-oriented software development Comm. ACM 35(9) 160-164

[7] Bauer G et al 2009 Dynamic configuration of the CMS Data Acquisition cluster J. Phys. Conf. Ser. (same conference, in print)

[8] Grama A Y, Gupta A and Kumar V 1993 Isoefficiency: measuring the scalability of parallel algorithms and architectures, IEEE Par. & Distr. Tech.: Systems & Applications 1(3) 12-21

[9] Clements P, Northrop L 2002 Software Product Lines Addison-Wesley (ISBN 0-201-70332-7)

[10] Gutleber J and Orsini L 2002 Software Architecture for Processing Clusters based on I/O Clust. Comp., J. of Netw., Software and Applications Kluwer Acad. Pub., 5(1) 55-65

[11] Fujimoto R M and Panesar K S 1995 Buffer management in shared-Memory Time Warp Systems ACM SIGSIM Sim. Digest, 25(1) 149-156

[12] Wind River Systems Inc. 1999 Network Protocol Toolkit, User’s Guide V 5.4 ed. 1 Part # DOC-12820-ZD-03 500 Wind River Way, Alameda, CA 94501-1153, USA

[13] Thadani M and Khalidi 1995 An efficient zero-copy I/O framework for UNIX Tech. Rep. SMLI TR95-39 Sun Microsystems Lab Inc. USA

[14] Bershad B N et al 1995 Extensibility, Safety and Performance in the SPIN Operating System in Proc. 15th ACM SOSP pp 267-284

[15] Booth D. et al. 2004 Web Service Architecture http://www.w3.org/TR/ws-arch

[16] Guttman E, Perkins C, Vaizades J and Day M 1999 Service Location Protocol Version 2 Internet RFC http://www.ietf.org/rfc/rfc2608.txt

[17] Adobe Flex 3 Rich Internet Applications http://www.adobe.com/products/flex/

[18] Antchev G et al 2001 The Data Acquisition System for the CMS Experiment at LHC in Proc. 7th Intl. Conf. Adv. Tech. and Particle Phys. Villa Olmo, Como, Italy (Oct. 15-19, 2001) World Scientific Publishers (ISBN 981-238-180-5)

[19] Murray S 2007 RU Builder User Manual CERN EDMS ID 875261, http://edms.cern.ch/document/875261/1.6

[20] Antchev G et al 2001 The CMS Event Builder Demonstrator and Results with Myrinet, Comp. Phys. Comm. 140(1-2) 130-138

[21] Bauer G et al 2008 CMS DAQ Event Builder Based on Gigabit Ethernet IEEE Trans. Nucl. Sci. 55(1) 198-202

[22] Gutleber J, Orsini L et al 2005 HyperDAQ Where Data Acquisition Meets the Web Proc. 10th Intl. Conf. Accel. and L. Exp. Phys. Control Sys. (Geneva, Switzerland, 10-14 October 2005)