A CUSTOMIZABLE PLATFORM FOR HIGH-AVAILABILITY MONITORING, CONTROL AND DATA DISTRIBUTION AT CERN

M. Braeger, M. Brightwell, A. Lang and A. Suwalska, CERN, Geneva, Switzerland*

Abstract

In complex operational environments, monitoring and control systems are asked to satisfy ever more stringent requirements. In addition to reliability, the availability of the system has become crucial to accommodate for tight planning schedules and increased dependencies to other systems. In this context, adapting a monitoring system to changes in its environment and meeting requests for new functionalities are increasingly challenging. Combining maintainability and high-availability within a portable architecture is the focus of this work. To meet these increased requirements, we present a new modular system developed at CERN. Using the experience gained from previous implementations, the new platform uses a multi-server architecture to allow running patches and updates to the application without affecting its availability. The data acquisition can also be reconfigured without any downtime or potential data loss. The modular architecture builds on a core system that aims to be reusable for multiple monitoring scenarios, while keeping each instance as lightweight as possible. Both for cost and future maintenance concerns, open and customizable technologies have been preferred.

INTRODUCTION

The context of this work is the required replacement of an existing monitoring and control system at CERN: the Technical Infrastructure Monitoring system (TIM) takes part in the supervision of infrastructure surrounding the accelerator complex and is mainly used within the CERN Control Centre [1][2]. It has been in place for the last 5 years (TIM) and is now at an advanced stage of maturity. However, due to the use of now obsolete technologies, a new implementation was required, and this was an occasion to reassess the current and future use of the system, and update the design to meet these new challenges.

The services provided by TIM have evolved over the years. In addition to the direct monitoring of hardware, the software also provides real-time control of access systems and transmission of video signals. Such real-time services impose strict time constraints on the transmission of selected data. Moreover, due to the proven reliability of the service, an increasing number of external systems have come to rely on it: data re-distribution over various protocols is now one of the principal tasks of the system. The data is then being used in making critical decisions elsewhere.

With the evolutions described above, the availability of the system has become increasingly important, imposing strict constraints on any maintenance stops. This is accentuated by the variety of the data transiting through the system, since it cannot be linked to any particular phase of accelerator operation. One of the main aims during the redesign was to improve the availability of the old system – although already very high – and facilitate maintenance procedures.

The re-designed TIM system is based on the new CERN Control and Monitoring Platform (C^2MON*). The aim is to provide a generic platform that can be reused in multiple monitoring scenarios. A second service at CERN will also be based on C^2MON in the near future (the Diagnostic and Monitoring service or DIAMON [3]). More generally, a shared platform should help in harmonizing monitoring solutions used at CERN, optimizing costs, maintenance and system compatibility.

CONSTRAINTS ON MODERN MONITORING SYSTEMS

Modern monitoring systems are asked to meet ever more stringent requirements. Reliability has always been a key demand, as for all systems used on a 24/7 basis for operational monitoring: data should pass reliably through the system and be reliably correct! On the other hand, the handling of critical data can result in new availability constraints: external operational decisions may rely on this data. Moreover, these actions may even be triggered in an automatic way, with processes gathering data from the monitoring system through some external protocol. In addition to these availability constraints, performance in terms of data throughput and delivery speed must often be guaranteed. Any transmission of signals used for human interactions impose tight constraints on the time needed for data to pass through the system, including during data avalanches. A good example of this is data used for access control, when operators will expect a rapid response (typically under 1s). Finally, while satisfying these additional constraints, it is expected that the product will provide good maintainability qualities, in particular in terms of applying patches at short notice and the introduction of new features to satisfy evolving demands.

In view of the discussion above, the C^2MON platform aims to increase the availability and maintainability of the systems it is replacing, while maintaining the reliability of the older systems.

SYSTEM ARCHITECTURE: A GENERIC-PURPOSE MONITORING PLATFORM MAXIMISING AVAILABILITY

Having covered the background to this project and discussed current monitoring requirements in general, we now focus on the design of the C^2MON platform, and

* Further details from mark.brightwell@cern.ch

how the design choices have helped achieve the goals outlined above.

Generic: in what sense?

Many monitoring scenarios have similar requirements: acquiring data using a variety of protocols, managing the data (temporary storage, logging, distribution for instance), and displaying it for human visualization. Using a modular architecture, the C^{2}MON platform was designed from the start to fit multiple monitoring scenarios: at CERN, it will be used as the basis for two distinct monitoring systems TIM and DIAMON.

The DAQ design allows modules to be written for new protocols and underlying hardware. The server architecture builds on a core part, to which optional modules can be added for custom behaviour. A client API is provided and can easily be reused in external GUIs for displaying C^{2}MON data (an actual GUI implementation is also available). The only genuine constraint when using the C^{2}MON platform is the data structures involved: for instance, internal data points correspond to single primitive values and need configuring as such in the database. Adding new data structures to the system remains possible, but requires a good understanding of the core design of the platform.

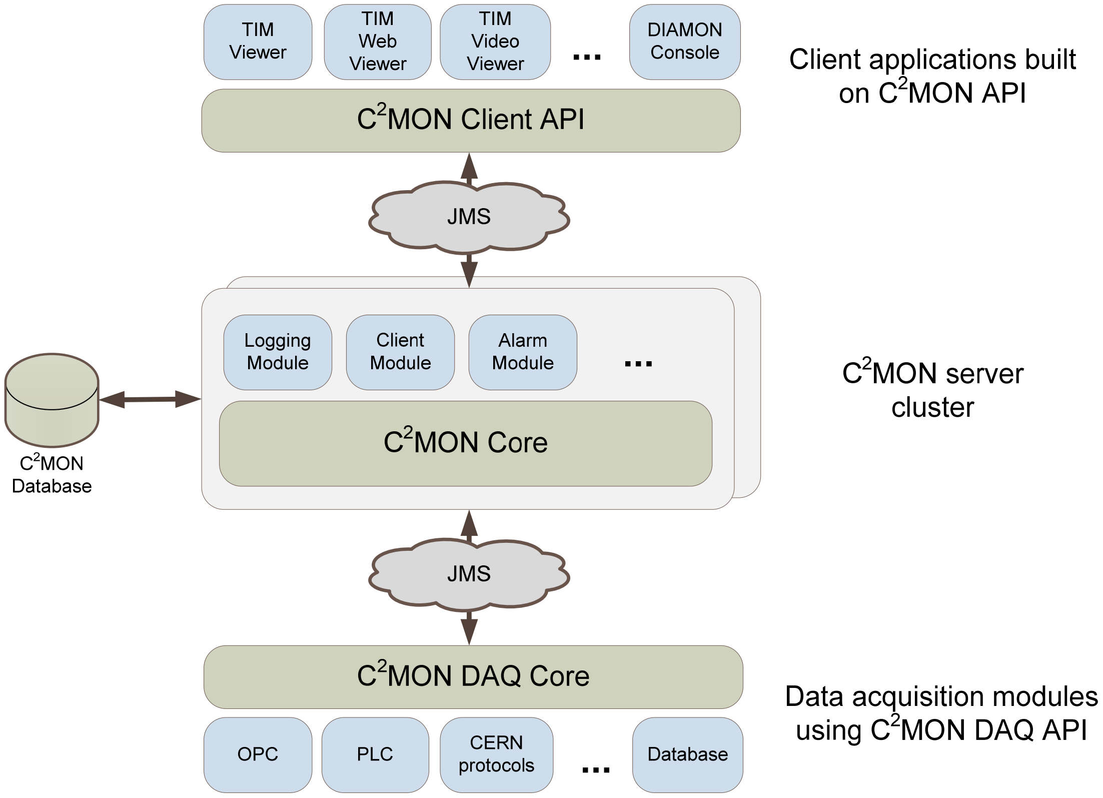

Figure 1: C^{2}MON architecture overview

Architecture Overview

The C^{2}MON platform follows a traditional 3-tier architecture, with data acquisition (DAQ), business logic (server) and client layers. An overview of the architecture is provided in figure 1 above. The DAQ layer is made up of any number of separate processes acquiring data through a variety of protocols. A DAQ implementation makes use of a common provided Core API for communicating with the business layer. The server layer itself consists in a cluster of servers. The server architecture is made up of a core part and a number of optional modules providing additional functionalities. The client layer provides a complete API to communicate effectively with the server. A number of client applications have been built on top of this API (e.g. the TIM Viewer). The various layers are linked with each other through messaging middleware.

The system is designed to run both in a single-server and multi-server configuration. In the multi-server configuration, data is shared between the servers in a distributed cache. Servers can be started on the fly and join an existing cluster. The multi-server configuration improves the availability of the overall system by adding failover support: in case of a single server failure, the remaining cluster is able to take over its workload. It also improves the maintainability of the system, since patches can often be rolled out across the servers without bringing down the whole cluster. The cache is regularly persisted to the DB and can be recovered even after a complete cluster restart.

As already explained, running in a multi-server mode improves the availability of the C^{2}MON platform. On the other hand, it comes at the cost of a more complicated design, and involves the use of more advanced/specialized software for the cache distribution (see the section below on technological choices). For this reason, the option of running in a single-server mode is of interest, particularly when occasional short downtimes are possible. This mode also remains as an emergency fallback scenario, rapidly deployable with minimum configuration settings.

Runtime System Data Configuration

As already mentioned, adding/removing or reconfiguring the monitored data is a frequent use-case for most monitoring systems. To maintain a high availability, the design must allow for these to be applied with minimal disruption to the service. To achieve this, the C^{2}MON platform uses a flexible approach, allowing each data acquisition module to decide whether it is able to apply the changes with or without a restart of the particular acquisition process. Before providing some details, let us briefly review the C2MON re-configuration process.

All the configuration details for the monitored data points are held in an *offline* database. Before submitting this data to the online C^{2}MON system, it must pass a strict data validation process. This process has proven as important as the system design to the long-term success of the service, but will not be discussed further in this paper. Once the data is correctly configured offline, it can be passed to the C^{2}MON system for online configuration. On reception of a configuration request, the C^{2}MON server loads the details from the offline database. The changes are applied internally, persisted in the online DB and made available in the distributed cache. These changes are applied immediately across all servers in the cluster. Moreover, any changes affecting the data acquisition layer will be propagated down to the appropriate DAQ modules. The functionality available at this level will then depend on the particular DAQ module implementation (specific to the protocol or hardware): if any changes cannot be applied at runtime by the module, the server will be informed that a DAQ process restart is necessary for the changes to take effect.

Using Real-time Filtering on the DAQ Layer

Guaranteeing the availability of a monitoring system also involves protecting it from overload during data avalanches. The C^{2}MON system already greatly restricts the amount of data sent through the system by filtering out redundant values at the DAQ layer (these are forwarded to a separate statistics-gathering process for system administration purposes). However, to help protect the system further, the DAQ layer provides a *dynamic-filtering* option, whereby time dead bands are imposed on individual points detected as feeding too much data to the C^{2}MON server cluster (various strategies on measuring the data throughput are provided).

Technological Choices

The platform is implemented in Java. Based on previous experience, the technological choices were guided by the following principals:

- Stick to proven technologies, facilitating project maintenance and stability.
- Where possible, use open-source resources, facilitating the reuse of the software in other projects. If justified by performance, use other 3rd party products, but provide a basic open-source fallback option.

With these principals in mind, the following choices were made:

- Spring is used as the basic framework wiring together the various components.
- Concerning the JMS middleware, implementations of the various C^{2}MON layers are available for ActiveMQ only.
- The system was designed against an Oracle database, but should run with other databases (this has not been tested so far; some transaction support is required).
- Database persistence is done using the iBatis Java library.
- The cache library used is Ehcache, an open-source Java cache.
- To run in a multi-server configuration, the cache is distributed using the Terracotta technology, which involves running a separate Terracotta server (same company as Ehcache). This technology remains free when run in a basic configuration (sufficient for most purposes).

CONCLUSION

The C^{2}MON platform has been designed over the last year with a view of reusing it in multiple monitoring scenarios. With this in mind, care was taken to provide a flexible core, with convenient hooks for adding custom functionalities. Together with the essential reliability of such a system, the design has attempted to maximize its availability, without affecting maintainability. We list the important design choices made in achieving this:

- The option of running a cluster of servers using a distributed cache, providing a failover mechanism and allowing rolling application of software patches.
- Runtime configuration of the monitored points/ hardware (so without server/acquisition process restarts).
- No operational database dependencies (only required for reconfiguration functionality).

All core parts of the system rely solely on open-source & free technologies.

The C^{2}MON platform is currently in the testing phase, with a planned rollout into production in December 2011.

REFERENCES

[1] U. Epting, P. Ninin, R. Martini, P. Sollander, R. B. B. Vercoutter and C. Morodo, “CERN LHC technical infrastructure monitoring”, ICALEPCS'99, Trieste, 1999, https://edms.cern.ch/document/394483/1.

[2] J. Sowisek, A. Suwalska and T. Riesco, “Technical infrastructure monitoring at CERN”, EPAC'06, Luzern, 2006, https://edms.cern.ch/document/750284.

[3] M. Buttner, P. Charrue, J. Lauener and M. Sobczak, “Diagnostic and Monitoring CERN Accelerator Controls Infrastructure - The DIAMON Project - First Deployment in Operation”, ICALEPCS 2009, Kobe, 2009.