DIAMON2 - IMPROVED MONITORING OF CERN'S ACCELERATOR CONTROLS INFRASTRUCTURE

W. Buczak, M. Buttner, F. Ehm, P. Jurcso, M. Mitev, CERN, Geneva, Switzerland

Abstract

Monitoring of heterogeneous systems in large organizations like CERN is always challenging. CERN's accelerators infrastructure includes large number of equipment (servers, consoles, FECs, PLCs), some still running legacy software like LynxOS 4 or Red Hat Enterprise Linux 4 on older hardware with very limited resources. DIAMON2 is based on CERN Common Monitoring platform. Using Java industry standards, notably Spring, Ehcache and the Java Message Service, together with a small footprint C++ -based monitoring agent for real time systems and wide variety of additional data acquisition components (SNMP, JMS, JMX etc.), DIAMON2 targets CERN's environment, providing easily extensible, dynamically reconfigurable, reliable and scalable monitoring solution. This article explains the evolution of the CERN diagnostics and monitoring environment until DIAMON2, describes the overall system's architecture, main components and their functionality as well as the first operational experiences with the new system, observed under the very demanding infrastructure of CERN's accelerator complex.

INTRODUCTION

The controls infrastructure of CERN's particle accelerator complex covers large surfaces and integrates significant number and wide variety of equipment, including legacy hardware and software. Since the successful introduction of DIAMON [1] in 2007, the operational experience and the additional requirements surfaced, justified further development, which finally led to DIAMON2.

The main requirements identified were: improved scalability, simplified maintenance including system wide dynamic re-configuration at runtime, higher level business logic, improved analysis capabilities, possibility to replay historical events and to create synoptic panels. Last but not least DIAMON1 was getting obsolete, with its technological choices justified few years ago, but not anymore, notably oc4j EJB container and SonicMQ.

The new improved DIAMON2 should have been realized by better use of the available resources with emphasis on applying industry standards wherever possible and reusing existing components. Since two departments at CERN, the Beams (BE) and General Services (GS) deal with similar topics, a decision was taken to set-up a collaboration [2] which resulted in the creation of the CERN Control and Monitoring (C^2MON) platform – a generic framework for building inter-operational monitoring solutions.

DIAMON2, similarly to the Technical Infrastructure Monitoring (TIM2) [3] is based entirely on the C^2MON framework. The implementation time was shortened by delivering ready-to-use solutions for the majority of required functionalities, allowing us to focus on specific DIAMON2 features and customizations, especially on the design of the server-side business logic, system's configuration tools, development of the required Data Acquisition (DAQ) components and finally on the improvement of the GUI applications. At the same time, the active contribution of the DIAMON team to the C^2MON project resulted in the extension of the framework's functionality and significantly improved its robustness.

THE DIAMON2 ARCHITECTURE

DIAMON2 is following the classic multi-layer system approach allowing high level of flexibility. As illustrated in Figure 1, specific DAQ modules, based on the C^2MON DAQ Core ensure the communication between a variety of data providers and the DIAMON2 server. The communication can be uni- or bi-directional (support for settings and commands).

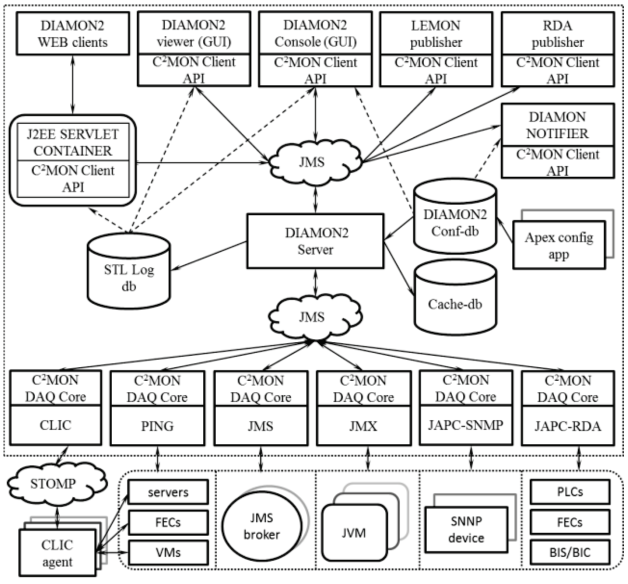

Figure 1: DIAMON2 architecture.

With a set of DAQ core plugins we establish communication to all required types of devices. The DIAMON2 server handles the messages received from the DAQ layer, computes the state of the rules (business logic) and provides the results to the client applications through C^2MON Client API. The communication between components of the system layers is based on JMS messages and configured through a set of web applications operating on DIAMON2 configuration database [4]. All configured points (further called *metrics*), alarms and rules (so-called *limits*) are handled inside the server's internal cache (*Ehcache*) and the states

ISBN 978-3-95450-139-7

are periodically persisted into the server's cache-persistency database. Updated metrics and limits are also periodically written into the Short-Term-Log (STL) database, which holds thirty days snapshots and provides the data on demand to the client applications. This allows our users to replay the chain of events and precisely analyse the problems retroactively. For all GUI clients and web-based client applications an instance of the Tomcat servlet container has been set-up, hosting a Spring Java module which renders a set of web pages with the data received at runtime from the server as well as the offline data from the STL database.

DATA ACQUISITION LAYER

DIAMON2 uses C^{2}MON DAQ core component. It handles automatic data-type conversions (from source value types to the types expected by the server), value range-checks, value and time-based dead-band filtering which are individually configurable. The core optimizes the traffic by buffering and grouping metric updates of the same priority before putting them to the output channels. The number and type of the output channels is also configurable (see Figure 2).

Equipment message handlers implement specific protocols, dedicated to the type of equipment, systems or processes they communicate with. C^{2}MON framework offers a wide variety of handlers from which DIAMON2 uses: JMX, JMS, PING, JAPC [5] (RDA, SNMP) and CLIC.

The DAQ modules are configured by the DIAMON2 server at start-up, and can be dynamically reconfigured anytime without the need of a process restart. This feature is extremely important and was one of the fundamental requirements for DIAMON2, dictated by frequently changing environment of the monitored infrastructure.

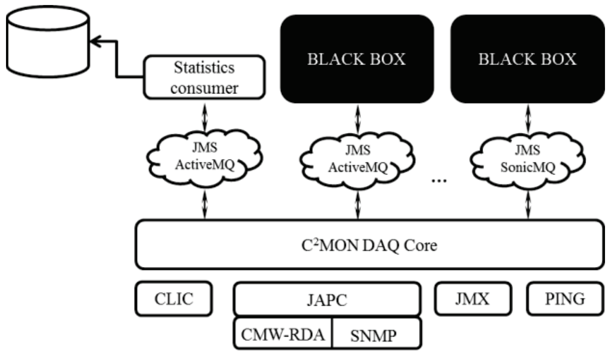

Figure 2: DIAMON2 DAQ components.

DIAMON2 CLIC Agent

The CLIC agent is a small footprint, non-invasive, modular data acquisition agent for monitoring operating system's parameters and other specific metrics. It is written purely in C++ and is available on multiple platforms (supports PowerPC and Intel processors, different operating systems both in 32 and 64 bit versions). The agent is modular (Figure 3) and easily extendable. The agent provides simple acquisition and does not have any business-logic, since this part has been moved entirely to the server in DIAMON2 system. In addition, the CLIC agent integrates acquisition functionality for selected system elements like hardware timing and CMX [6].

An instance of a CLIC agent is deployed on every computer monitored by DIAMON2 and started automatically when the machine boots. It communicates with CLIC DAQs through the STOMP protocol, as presented in Figure 1. Subscriptions and command-reply mechanisms are ensured via the *diamon2-agentlib* module, available for C++ and Java. Putting the STOMP broker in between the DAQs and the CLIC agents significantly improved the system's stability. Since handling of multiple subscriptions remains on the broker side, stopping (or rebooting) any agent (or CLIC DAQ) has no impact on the behaviour of other agents and data acquisition components.

At start-up, only the agent's core system module is loaded while additional modules are loaded only if requested. Currently network, disk, network timing, hardware timing, WorldFip and CMX modules are available, some of them being limited to specific operating system versions. Implementation and integration of additional modules are possible.

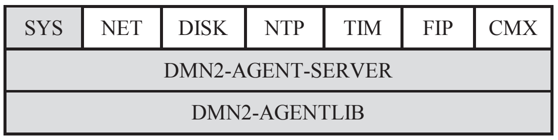

Figure 3: Modular structure of the CLIC agent.

The CLIC agents are deployed currently on about 2000 machines covering Linux, LynxOS and Windows. The agent acquires and publishes the data in regular intervals or as response to a specific acquisition command received from the DAQs. The CLIC starts with a predefined list of metrics to be collected and that list can be extended at run-time if remote instruction is received from the DAQ. The CLIC DAQ pushes reconfiguration request to the CLIC agent if it detects that some expected metrics are missing. By having this mechanism in place we keep the CLIC configuration centrally assuring the agents always monitor required metrics.

JMS and JMX

DIAMON2 not only monitors system properties, but also metrics of selected Java processes hosted on those systems. For that reason a dedicated JMX DAQ has been implemented with the aim to monitor configured metrics through periodic polling or by receiving attribute notification events. Since the introduction of DIAMON2 we observe growing interest in our JMX interface as this provides a powerful tool for debugging problems and even for preventing problems from happening thanks to the system's business logic and the notification service.

For C++ developers, we offer similar functionality through CMX [6] and the CLIC agent.

BE-CO's JMS broker infrastructure is monitored through JMX (number of topics, number of dynamic producers, message consumers, usage of broker internal storage, etc.) and through a set of tests evaluated by the JMS DAQ (queue and topic performance tests).

Additional Data Acquisition Modules

In addition to CLIC and JMX, DIAMON2 verifies in regular intervals network availability of all monitored computers and virtual machines using the PING DAQ.

Certain amounts of devices are also monitored using CERN's JAPC protocol, with its SNMP and CMW-RDA [7] extensions.

CMW-RDA protocol is used by DIAMON2 mainly to acquire information from FEC computers and PLC controllers around the LHC complex and for the Beam Interlock System (BIS). Finally, we provide a CMW-ADMIN DAQ which monitors the health of the CMW-RDA middleware's infrastructure.

SERVER LAYER

The DIAMON2 server, entirely based on C^{2}MON framework, is also modular. Modules are individual jars, loaded into the server's *Spring* context at start-up. They implement specialized functionality (e.g. preloading the cache from external database, server-to-DAQs communication, etc.) and usually operate on the internal cache instances of the server. C^{2}MON delivers large set of modules, from which some are mandatory, i.e. the server must load them at start-up in order to work properly, providing basic functionality (core server modules), while some are optional (see Figure 4). DIAMON2 loads five optional modules which are responsible for logging updated metrics into STL database (logging), user authentication (auth), server-client communication, rules evaluation (rule) and for propagating alarms to LHC Alarm Service (LASER).

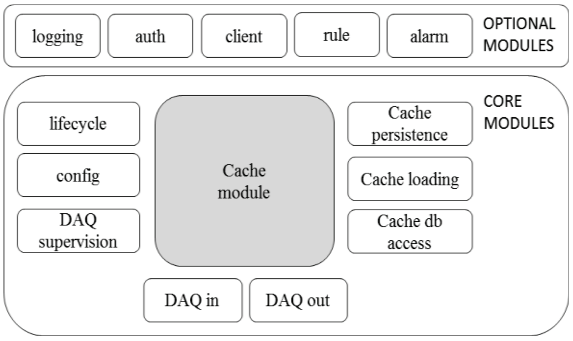

Figure 4: C^{2}MON-DIAMON2 server components.

Business Logic

In DIAMON1 business logic was distributed across components of different layers. In DIAMON2 the entire business-logic has been centralised inside the server. The server's virtual representation of the monitored systems reflects the physical one. We define a number of *equipment*, each monitoring a number of *metrics*. On top of selected metrics *limits* are declared. A limit is a *virtual metric* with simple logic behind (arithmetical, relational, bitwise or logical operators). The values of the limits are calculated by the server's rule module. Values of different limits (eventually based on data arriving from different DAQs) are combined to provide the overall state of the equipment (*equipment rule*) as illustrated in Figure 5. Each state has its unique value (OK, WARNING or ERROR), easily identifiable by associated colour which is used by the client GUI consoles and on the synoptic panels. Client applications subscribe to high level equipment limits, simple limits or individual metrics, depending on the use-case.

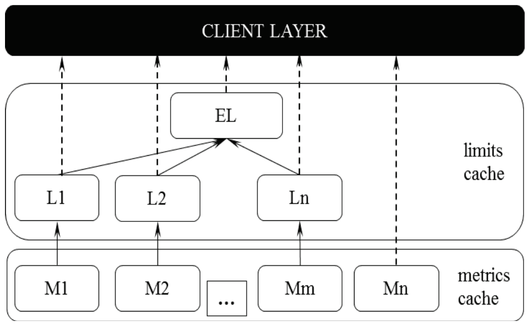

Figure 5: Metrics, Limits and Equipment-Limits in DIAMON2.

CERN's Safe Machine Parameters (SMP) controllers' (SMPC) health checks are also integrated into DIAMON2 through a set of metrics and limits.

CLIENT LAYER APPLICATIONS

DIAMON2 delivers interactive and a few non-GUI applications all build on top of the C^{2}MON Client API (see Figure 1). C^{2}MON Client provides API for subscribing to metrics, limits, authentication mechanisms, functions to query for historical data, connection assurance mechanisms (heartbeat) and commands execution facility.

DIAMON2 Console

The primary DIAMON2 GUI application used by the operators in the CERN Control Center (CCC) and other control rooms is the DIAMON2 Console. The application is also used by equipment experts to monitor and check the health of different equipment. The Console gives high level overview of the state of selected computers which are assigned to several groups. On demand the console can also present a list of services running on a selected host or a list of monitored metrics and limits. Authorised users may perform some operations, such as triggering additional data acquisition, restarting selected processes or rebooting a machine.

The DIAMON2 Console has been ported from DIAMON1 [8], refactored and equipped with a set of new functionalities (history replay, charts, link to the online web client applications etc.).

DIAMON2 Viewer

DIAMON2 Viewer is the secondary GUI application used to present synoptic diagrams. The viewer can run as standalone application or as an extension to the console. Users can now design their own synoptic panels in order to present selected monitoring data in a customized way, according to their requirements. Based on C^{2}MON framework, the DIAMON2 Console and the Viewer applications reuse a variety of common GUI components, making both interfaces very similar.

Notification Service

The result of server rule evaluation also triggers notifications. The users can set up notifications on selected limits in order to be notified via e-mail or SMS whenever any value they monitor gets out from the normal state. This task is handled by a standalone Java notification service process (see Figure 1).

Publishers

Publisher applications (see Figure 1) provide data to external applications. Clients interested in receiving updates (metrics or limits) from DIAMON2 have currently following options:

1. They can integrate with DIAMON2 using C^{2}MON Client library
2. They can subscribe to the DIAMON2 data published through the DIAMON2 CMW-RDA publisher.
3. Selected metrics can be accessed directly through LEMON [9] interface.

CMW-RDA Publisher

CMW-RDA publisher is a standalone Java process which acts as a gateway between DIAMON2 and CMW-RDA middleware. It allows users to subscribe to the DIAMON2 data, without the use of C^{2}MON Client library. For selected clients at CERN that method is preferred, mainly for none-Java clients and clients which already depend on CMW-RDA middleware. Currently our main client is the EN-ICE group with their PVSS SCADA systems.

LEMON Publisher

LEMON is a computer monitoring solution for Linux machines provided by CERN IT Department. Over a web interface it offers interesting functionalities for system administrators such as pre-generated clustered overview of historical evolution on different metrics. The LEMON publisher has been implemented to allow the use of the LEMON web interface for PowerPC FECs running LynxOS and profiting from the fact that DIAMON2 monitors those computers (through CLIC agents). Like the CMW-RDA publisher it uses the C^{2}MON Client API to subscribe to the selected metrics. On updates, it prepares the data package for LEMON and injects it into the LEMON server, using the LEMON-specific protocol.

OPERATIONAL EXPERIENCE

DIAMON2 was made available to operations in 2012 and completely replaced DIAMON1 in March 2013. Due to LHC Long Shutdown, the user community is currently reduced. However, DIAMON2 has been used for several months for the monitoring of parts of the CERN accelerators controls infrastructure and successfully passed the first LHC dry-run tests.

At the moment of writing this article, the production server maintains 4820 equipment (including 2060 CLIC), with around 67500 metrics and 30500 limits. Due to the dead-band filtering policy, tuned over time, the DAQs generate limited load on the server (around 200 updates per sec, with av. 4-5% CPU consumption), which is far below the server's capacity evaluated during performance tests, done in 2011 (during the tests we generated 30'000 updates/sec and the server computed approximately 40'000 rules/sec, with avg. heap consumption of 1.5Gb and CPU load oscillating around ~18%. We conducted the tests on HP ProLiant BL460c G7 24-core machine).

First operational experience also revealed that the configuration of new elements to be monitored must become simpler for the users. This is important especially for the successful integration of controls software into the monitoring system. The DIAMON team will put the focus on this aspect in the coming months.

REFERENCES

[1] P. Charrue et al. "The DIAMON project – Monitoring and diagnostics for the CERN controls infrastructure" Proceedings of ICALEPCS 2007, p. 588 (2007).

[2] A Suwalska, M. Buttner, M. Braeger, W. Buczak "C^{2}MON CERN control and monitoring platform", CERN 2011.

[3] M. Braeger et al. "A customizable platform for high-availability monitoring, control and data distribution at CERN" Proceedings of ICALEPCS'11, p. 418 (2011).

[4] Z. Makkonen et al. "Challenges to Providing a Successful Central Configuration Service to Support CERN's New Controls Diagnostics and Monitoring System" Proceedings of ICALEPCS 2013 (2013).

[5] V. Baggiolini et al, "JAPC - the Java API for LHC Timing Parameter Control", ICALEPCS'05, Geneva, Switzerland, TH1.5-8O.

[6] F. Ehm et al. "CMX - A Generic In-Process Monitoring Solution for C and C++ Applications" Proceedings of ICALEPCS 2013 (2013).

[7] Controls Middleware Project (CMW-RDA) http://proj-cmw.web.cern.ch/proj-cmw.

[8] M. Buttner et al. "Diagnostics and Monitoring CERN Accelerator controls infrastructure: the DIAMON project first deployment in operation" ICALEPCS'09

[9] LEMON – LHC Era Monitoring http://lemonweb.cern.ch.

ISBN 978-3-95450-139-7