ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA179

MANAGEMENT SOFTWARE AND DATA EXCHANGE PROTOCOL FOR THE INFN-LNS ACCELERATORS BEAMLINES

G. Vecchio, S. Aurnia, S. Cavallaro, L. Cosentino, B. Diana, E. Furia, S. Pulvirenti
INFN Laboratori Nazionali del Sud, via S. Sofia 62, 95135 Catania

Abstract

This paper describes the design and the development of an innovative management software for the accelerators beamlines at INFN-LNS. The Graphical User Interface, the data exchange protocol, the software functionality and the hardware will be illustrated. Compared to traditional platforms for the accelerators console, at INFN-LNS we have developed a new concept of control system and data acquisition framework, based on a data structures server which so far has never been used for supervisory control. We have chosen Redis as a highly scalable data store, shared by multiple and different processes. With such system it is possible to communicate cross-platform, cross-server or cross-application in a very simple way, using very lightweight libraries. A complex and highly ergonomic Graphic User Interface allows to control all the parameters with a user-friendly interactive approach, ensuring high functionality so that the beam operator can visually work in a realistic environment. All the information related to the beamline elements involved in the beam transport, can be stored in a centralized database, with suitable criteria to have a historical database.

INTRODUCTION

In a complex and heterogeneous environment, like the LNS beamlines, is very important to have a clear and powerful synoptic which helps the operators in their work of beam transportation.

For this purpose, we developed a complex and highly ergonomic Graphical User Interface, that allows to control all the parameters adopting a user-friendly interactive approach, so that the beam operator can visually navigate through a pseudo-realistic beamline reproduction. The modular approach of this platform allows to build and modify all the beamlines, just adding/removing all the elements, that can be managed individually.

The users that transport the beam can therefore set and read all the setting parameters, thanks to a continuous communication with the field level. The data communication has been realized following a new concept of control system and data acquisition framework, based on a data structures server which so far had never been used for supervisory control.

The beamline consists of different devices from different vendors and for this reason a standard protocol for data exchange has been developed.

We have chosen Redis [1] as a highly scalable data store, shared by multiple and different processes and applications. This system easily allows cross-platform, cross-server and cross-application communication, using extremely lightweight libraries.

We decided to call our system E.T.N.A., acronym of Enhanced Transport Network for Accelerators.

SYSTEM ARCHITECTURE

The system architecture is composed by three level, as shown on Figure 1. The field level is composed by all sensors, PLCs and field machines of the beamlines. The user interfaces aim to provide control and monitoring of the beamline to operators. In the middle of this architecture we have the core of our system, the REDIS online DB, used for data exchange from both field level and the GUIs.

The middle tier also hosts a MySQL server for static data storage and we have a certain degree of fault-taulerance by using Linux-HA and DRBD. Linux-HA is a clustering solution that provides reliability, availability and serviceability. In our architecture we have two identical servers, a master and a slave, which ensure that the services running on one of them can be automatically moved and restarted in the backup's node.

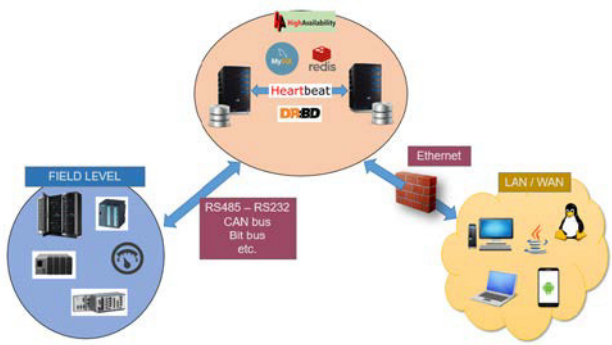

Figure 1: System architecture.

Cluster infrastructure services are implemented via Heartbeat. This daemon allows clients to know about the presence or disappearance of peer processes on other machines and to easily exchange messages with them. The Heartbeat daemon must be combined with a cluster resource manager, which has the task of starting and stopping the services that cluster will make highly available. We used Pacemaker that is the preferred cluster resource manager for clusters based on Heartbeat.

THE CONSOLE GUI

The local console GUI, queries the in-memory database through a polling procedure and gets the data coming

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA179

from the field. The communication between Redis and the local console is done by a TCP connection through a proprietary protocol called RESP.

The software architecture adopted on the local console is three-tier with the use of the architectural pattern Model-View-Controller (**MVC**), which separates the presentation layer from business logic and data storage.

The **view** shows the data provided by the **model** that is directly connected to the database, whereas the **controller** accepts inputs and converts them to commands for the model or view, as shown on Figure 2.

Moreover, in the model layer, we have used Java Persistence API (JPA), a Java specification for the access, the persistence and the management of data between Java objects and relational databases.

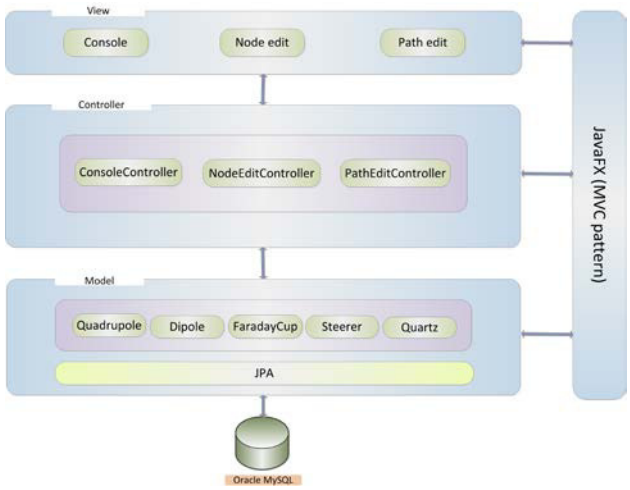

Figure 2: Console software architecture.

The main advantages of JPA compared to a database driver connector is that in JPA data is represented by classes and objects rather than by tables and records.

Interactive User Interface

The synoptic for the control of beamline elements was developed using the JavaFX framework [2]. Figure 3 shows the panel that contains the plant of the building, where the operator can navigate easily whit a mouse and can zoom in and out to see some place in detail.

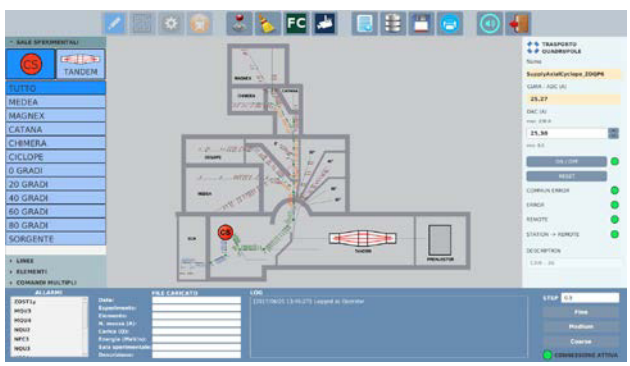

Figure 3: Interactive user interface.

Thanks to the modular panel that shows the status of beamline, the console operator can have a real-time feedback and operate without knowing in detail the nomenclature of the various power supplies in the beamline.

Configurations Save and Reload

A great advantage of this software platform is represented by the availability of a MySQL database to save and recall a specific system configuration. Figure 4 shows the particular ergonomic design of the panel for the configuration retrieval, that can be done by element, mass number (A), state of charge (Q), energy (E), date and name of the experiment.

Moreover, it is possible to print a specific setting simply by choosing a record stored in the database. This feature avoids tedious work to operators, which, until now, had to write by hand every configuration in a huge book.

Figure 4: Save and reload configurations interface.

PERFORMANCE TEST

The tests carried out on the system showed the excellent management of the server resources like CPU and memory. Indeed, as shown in Figure 5, we have seen that the bottleneck is represented by the traffic network, if data size increase, whereas the CPU time decreases in the server where Redis runs [3].

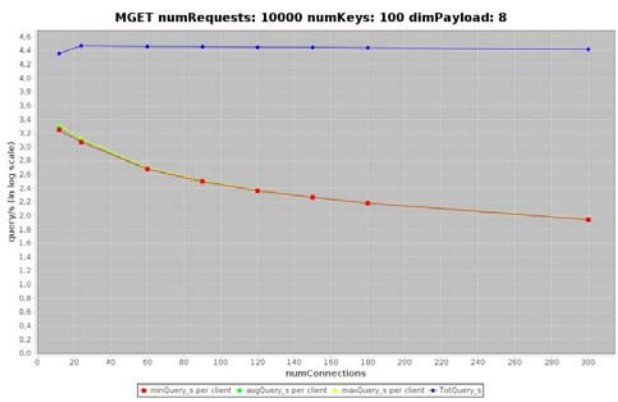

Figure 5: Number of query/s decreases as the number of connections increase.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA179

CONCLUSIONS

In this paper we presented a new system architecture and a new user interface for the management of the LNS accelerators beamline. The system meets the initial requirements and provides to operators a simple and interactive control.

It also shows excellent performance, reliability and fault tolerance.

REFERENCES

[1] Redis documentation, https://redis.io/documentation

[2] Johan Vos *et al.*, "Pro JavaFX 8: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients", 2014, ISBN 978-1-4302-6575-7.

[3] Sabrina Micale, "Redis come back-end di un sistema di controllo distribuito: analisi delle prestazioni e delle criticità", master's thesis.