# HIGH PERFORMANCE WEB APPLICATIONS FOR PARTICLE ACCELERATOR CONTROL SYSTEMS

G. Mazzitelli, C. Bisegni, P. Ciuffetti, G. Di Pirro, A. Stecchi - INFN/LNF, Frascati (RM), Italy;
S. Calabrò, L. Foggetta - LAL-CNRS, Orsay, France & INFN/LNF, Frascati (RM), Italy;
L. Catani, F. Zani - INFN-Roma Tor Vergata, Roma, Italy

*Abstract*

The integration of web technologies and applications has been one of the major trends for the development of new services for Control Systems (CS) of particle accelerators and large experimental apparatuses. Nowadays, high performance web technologies exhibit some features that would allow their deeper integration in a CS and their employment in developing CS' core components. In this paper we discuss the results of preliminary investigations of a new paradigm for a particle accelerators CS and the associated machine data acquisition system based on a synergic combination of network distributed cache memory and a non-relational key/value database. Storage speed, network memory data retrieve throughput and database queries execution, as well as scalability and redundancy of the systems, are presented and critically reviewed.

## INTRODUCTION

The Italian Ministry for Education, University and Research (MIUR) recently approved the construction of a new international research centre for fundamental and applied physics to be built in the campus of the University of Rome "Tor Vergata". It will consist of an innovative high-luminosity particle collider named **SuperB** [1] and experimental apparatuses, built by an international collaboration of many important scientific institutions under the supervision of Istituto Nazionale di Fisica Nucleare (INFN) and the skills and support of the neighbouring INFN Frascati Laboratory (LNF). Clearly, it will offer great opportunities not only for new discovering in particle and applied physics, but also for breakthrough innovation in particle accelerators technologies.

The Frascati and Tor Vergata control groups have a long experience in design, development and implementation of innovative CS. In the '90s, the first PC and LabVIEW®[2] based CS has been successfully developed and operated for DAΦNE accelerator at LNF, breaking through the common concept of controls [3]. This experience and know-how is available today for a new challenging project.

The idea is to design a new controls system based on the present software trends, dominated by web technologies and services, where large databases and the most robust available data bus, ETHERNET, are used to match very high throughput. The large community of developers and users involved guarantees a good support and may give hints on the longevity of the product.

The new CS, must be designed in such a way to accommodate any kind of devices to reduce the hardware dependence and the development time by exploiting the availability of many devices with embedded programmable CPU. Furthermore, the CS has to be able to control and - where needed - to **acquire** data with performance limited only by the hardware capability.

These requirements suggest inverting the typical CS device-client data flow from **polling** (the client polls) to **pushing** (the device push) information.

## !CHAOS

The aim of the design (Fig. 1) for this new CS we named !CHAOS (**C**ontrol System based on **H**ighly **A**bstracted **O**perating **S**tructure – but not a mess!) is to provide a solution that naturally allows: redundancy of all its parts, intrinsic scalability, minimization of points of failure, hardware hot-integration and auto configuration.

To achieve such requirements !CHAOS, basically developed in C++, employs distributed object caching for real-time data access (Live Database) and a key-value database for data archiving (History Database).

A Control Library (CL) completely manages data and commands flow, the control processes and the devices configuration. The device's programmer is only asked to develop the driver for the specific controlled hardware.

The CS component hosting the control software for a device (or a family thereof) is called Control Unit (CU) and is the only part that instances the !CHAOS abstraction. The CL also provides, at the CU INIT phase, the **syntax and semantics** for dataset and command to the Metadata Server (MS) that allows the correct information retrievement.

The CL takes also care of the data serialization, the communication with databases, handling of system's and client's commands and standard services of CS.

The data serialization strategy adopted for !CHAOS is BSON, a binary-encoded JSON (JavaScript Object Notation) [4] documents, optimized for fast storage performance. The conversion from BSON and JSON is fully supported.

### *Front-end*

A fundamental requirement for the !CHAOS design is the possibility to accommodate any kind of Acquisition Hardware (AH) in such a way to be free as much as possible from the hardware choice. AH can be grouped into three classes:

- CPU controlled devices: instruments connected to a controller that performs data acquisition and processing;
- Embedded devices: instruments providing an embedded CPU and a set of API allowing the control of their functionalities by software on board; examples are digital oscilloscopes, intelligent power supplies, and so on;
- Complex IO controllers or sub-systems: PLC systems, distributed I/O buses (such as VME, cPCI, xTCA and so on).

The high portability of the CL to the widest selection of operating systems and CPU boards leads to a multi task process that provides to handle input (commands) and output (readouts) data, initialize and configure data flow (type, execution frequency, etc).

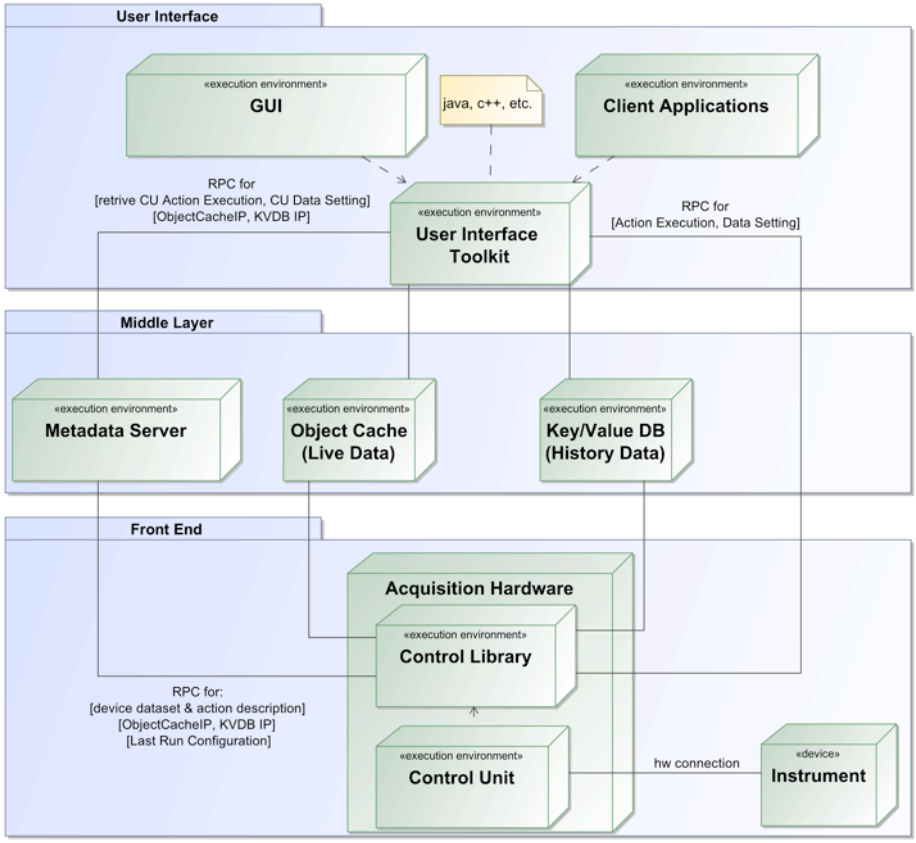


Figure 1: !CHAOS Control System operating structure and data flow: low level hardware and controller (front end), database (middle layer), high level (user interface).

The Metadata Server (MS) collects configurations, syntax and data semantics. Its main function is to maintain the knowledge of all components and devices under control of the CS and their operating condition. MS acts also as supervisor of commands flowing between CL and User Interface (UI).

Each *front-end* sets and gets its own device configurations in the MS. It also auto-configures all data semantics and syntax needed to client applications for data retrieving and to optimize access, load, etc.

## *Middle Layer - Databases*

Data acquired by the CU, formatted according their dataset and then serialized, are both updated in the **data object caching** and stored into the **key-value DBs** with independent and user adjustable push rates.

The data object caching is optimized to cache data in real time, typically for displaying on the User Interface, while the key-value DB is optimized to store large amount of data with high throughput.

The possibility to set their push rates independently can be used for optimizing performance and load on the two services.

Two open-source software are currently under tests as candidates for the live-data (memcached [5], a data object caching) and the history-data (mongoDB [6], a key-value DB). Both of them allow for scalability and redundancy.

## *User Interface*

Symmetric to the CL, at the higher level in the !CHAOS structure, is the **User Interface Toolkit.** This library provides the client applications (display panels, measurement applications, etc.) with the interface to the CS framework for retrieving configuration information from MS, access live data (from cache db) and archived data (from history db), send commands to devices, etc.

By using the datasets information stored into MS, the UI Toolkit will be able to auto-generate Graphical User Interfaces for controls and data presentation.

### *Triggered DAQ mode and timing*

One of the features we expect by the !CHAOS framework, or rather by the CL, is the possibility to easily trigger the hardware and synchronize operations among distributed components. The integration of the trigger/timing system in the CS will give the CL the possibility to handle synchronously hardware trigger dispatched to the instruments distributed around the accelerator:

- a PRE TRIGGER command configures the devices for a specific task to be executed when the hardware trigger will be detected. The pre-trigger also configures a **timing controller** to broadcast the hardware trigger to the interested devices;
- any PRE TRIGGER is flagged with a specific tag, required to retrieve coherently data in the databases;
- TRIGGER command is sent to timing controller which latches the time stamp and sends hardware trigger to AH;
- data from AH and timing controller are updated with their own duty cycle in the live/history databases.

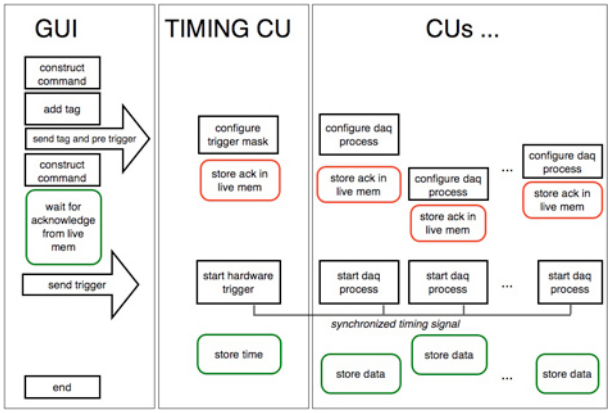


Figure 2: Timing trigger flow.

In such a way it will be possible to synchronize data coming from many different acquisition hardware, being limited only by the latency of the different devices.

This item is strictly correlated to the choice on the accelerator timing system, under study for the SuperB accelerator. A custom controller, white rabbit system, etc. are under study to understand the best choice.

## PRELIMINARY RESULTS

The CS, presently driving the DAΦNE accelerator has been used for testing one of the core components of !CHAOS: the *live-data* caching. At DAΦNE the front-end controllers acquire the data from the devices and continuously update - through fiber optics - a central common VME address space, which is also accessible from the user's applications.

For our tests, the routines used to update the devices' set values have been easily modified in order to write into a memcached server, running on a Linux box with 100 Mbps Ethernet network interface, instead of the VME address space. The class of devices migrated is the ICE (Ion Cleaning Electrodes), which is represented by a data structure of 64 bytes. During the test, the dataset has been fetched with a frequency ~ 100 Hz. The result shows no dependency on the number of fetching consoles (up to 7 in the test) and the *memcached* server CPU load never exceeded 0.7 % and a memory usage ~ 0.1%.

An analogous test has been performed at SPARC by caching a 640x480@8 bit (300 kByte) image from a digital camera of the optical beam diagnostic. The image was transferred from the digital camera with no dependency on the consoles number (up to 4 in the test) accessing the image (Gbit network).

A first release of the CL has been developed on Linux and OS X Operating Systems and it is also under integration with the LabVIEW environment.

## CONCLUSION

SuperB is pushing us to study and implement new ideas in controls, to be up to date in integrating commercial web technologies and to overcome the primary issue coming from previous architectures: the limits due to the usage of specific hardware and software.

The plans are to develop the core software of the CL and to explore its critical issues - if any - by the end of 2011. In the mean time some preliminary test started on the DAΦNE and SPARC accelerators, where is available a natural gym to understand any possible problem and rapidly solve it in a real operative contest.

The preliminary tests undergoing on DAΦNE accelerator, in real environment and on real elements to be controlled, have confirmed that the performance of a no-relational database resident on RAM is practically limited only by Ethernet bandwith. The systems load is very low, while redundancy and scalability allows being confident on the behaviour for a larger accelerator complex such as the SuperB.

## ACKNOWLEDGMENTS

We would like to thank the Padova group of M. Bellato et al, M. Serio and A. Drago who are starting to collaborate on the timing issues related to the SuperB controls and accelerator synchronization, as well as the LNF computing centre for the support provided in system administration issues and networks analysis and implementation. M. Biagini and P. Raimondi for the encouraging start up support.

## REFERENCES

[1] SuperB-CDR2 INFN-LNF-11/9(P) 15 Jun 2011
[2] http://www.ni.com/labview/
[3] G. Di Pirro eta al, NIM A 352, 455-457 (1994)
[4] http://bsonspec.org/; http://www.json.org/
[5] http://memcached.org/
[6] http://www.mongodb.org/