Exploring a New Paradigm for Accelerators and Large Experimental Apparatus Control Systems

L. Catani, R. Ammendola, F. Zani, C. Bisegni, S. Calabrò, P. Ciuffetti, G. Di Pirro, G. Mazzitelli, A. Stecchi, L.G. Foggetta

EXPLORING A NEW PARADIGM FOR ACCELERATORS AND LARGE EXPERIMENTAL APPARATUS CONTROL SYSTEMS

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

Abstract

The integration of web technologies and web services has been, in the recent years, one of the major trends in upgrading and developing control systems for accelerators and large experimental apparatuses. Usually, web technologies have been introduced to complement the control systems with smart add-ons and user friendly services or, for instance, to safely allow access to the control system to users from remote sites.

Despite this still narrow spectrum of employment, some software technologies developed for high performance web services, although originally intended and optimized for these particular applications, deserve some features that would allow their deeper integration in a control system and, eventually, using them to develop some of the control system's core components. In this paper we present the conclusion of the preliminary investigations of a new design for an accelerator control system and associated machine data acquisition system (DAQ), based on a synergic combination of network distributed object caching (DOC) and a non-relational key/value database (KVDB). We investigated these technologies with particular interest on performances, namely speed of data storage and retrieve for the distributed caching, data throughput and queries execution time for the database and, especially, how much this performances can benefit from their inherent scalability.

INTRODUCTION

Two main motivations support the decision to start investigating a new approach in the design and development of CS for particle accelerators.

New developments in this field, similarly to what has happened in recent years, will be basically directed towards improving their functionalities by introducing new services, or improving existing ones, to complement the basic features that are essential for the remote control of the accelerator's components.

These new capabilities rather than being accessorial will be, in many cases, fundamental for the optimal operation of new accelerators that will require careful tuning to achieve the desired performance. An example may be the data acquisition system that is intended to provide machine physicists, but also the experimental groups, with all the information needed to recreate the operational state of the accelerator (set-point of components, information from the beam diagnostic etc.) at any instant during the operations of the machine.

The analysis of recent developments on high-performance software technologies suggests that the design of new accelerator CS may profit from solutions borrowed from cutting-edge Internet services. To fully profit from this new technologies the CS model has to be reconsidered, thus leading to the definition of a new paradigm.

The second strong motivation for this development follow the recent approval, by the Italian Ministry for Education, University and Research (MIUR) of 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 very high-luminosity particles collider named SuperB [1] and experimental apparatuses, built by an international collaboration of major scientific institutions under the supervision of Istituto Nazionale di Fisica Nucleare. 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 !CHAOS FRAMEWORK

A typical example of software technology emerging from developments of Internet services is the class of non-relational databases known as key/value database. They offer an alternative to relational databases (RDMS) that is having a growing success and interest among developers of web services because of their high throughput, scalability and flexibility.

Another example is the object caching, distributed systems that are used to store, in the servers' RAM, frequently requested sets of information in order to both respond faster to requests and distribute the load of the main server to a scalable cluster of cache servers.

These two software technologies, clearly cited on purpose, represent the core components in the design of this new control system we named !CHAOS[2].

In particular, the KVDB is used by DAQ for storing what we call *history data*, while the DOC implements the service for distributing *live data* from the front-end controllers to clients replacing the client/server communication.

Compared to the typical structure of CS, usually represented by the so-called *standard model* [3] of control systems, in the !CHAOS data flow the client (top) and front-end (bottom) layers are not directly connected and, especially, data is not sent by controllers when triggered by

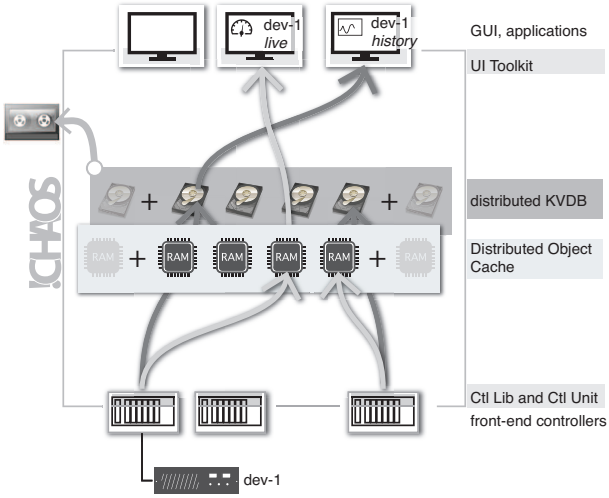

Figure 1: Data flow in the !CHAOS framework.

client request. Instead, alternatively to the typical point to point communication of network distributed systems, in !CHAOS *live data* flow from front-end controllers to the DOC servers, according to the independently adjustable refresh rate, from which datasets can be asynchronously read from any client.

This solution offers a number of advantages.

Firstly, we can use the same strategy, and topology, for both distributing *live data* and storing *history data* as shown in Fig.1. Datasets that need to be updated are identically pushed, by front-end controllers, to both DOC and KVDB servers by issuing *set* commands. It means that data collection mechanism for DAQ is inherently included in the !CHAOS communication framework because both *live* and *history* data are pushed by the data source (the front-end controllers) to similarly distributed caching and storage systems. Moreover, since both DOC and KVDB use key/value data storage, formatting and serialization of datasets can be identical.

Secondly, both the client applications and the front-end controllers are simple clients of the distributed object caching and DAQ. In particular, provided the DOC has an object container for each dataset of the CS, defined by its unique key, a GUI client simply send to the !CHAOS DOC service a *get* request for the object identified by that particular key, i.e. the dataset describing the associated device. On the other side the controller responsible for that device update its dataset, according to the push rate defined for it, by issuing *set* commands to the DOC.

Data refresh rates, as well as other meta-data and global parameters, will be managed by the meta-data server (MDS) that will be described later.

It's worth to underline that in !CHAOS the front-end controllers don't need to run servers for providing data to clients since they themselves are clients of the data distribution and storage services. That improves their robustness, portability and prevents front-end controllers from overload originated by multiple clients' requests.

A fundamental property of both DOCs and KVDBs is their intrinsic scalability that allows distributing a single service over several computers. Moreover, dynamical keys re-distribution allows automatic failover by redirecting to other servers the load of failed one. By taking advantage of this feature !CHAOS can be easily scaled according to both different size of the accelerator infrastructure and the performance required thus avoiding any potential bottleneck that may be expected as the weak link of the star-like communication topology.

In conclusion !CHAOS is a scalable control system infrastructure providing all the services needed for communication, data archiving, timing, etc. to which both front-end controllers and GUI applications plug-in to access and, to some extend, expand its functionalities.

Control Units: The !CHAOS Front-end

Fig.2 shows the logical structure of the software running in a front-end controller. The Control Library (CL) and the Control Unit (CU) are components of the !CHAOS infrastructure while the device management plug-ins (DMPs) are software drivers complementing the !CHAOS framework functionalities by providing the interface to the device. The development of these components is expected either as contribution, or under responsibility, of device experts.

One or more instances of CU can run simultaneously, though completely independent, in a front-end controller. Each of them will be dedicated to a particular device or a family thereof, specialized for that particular components by means of the device management plug-in. The latter are a set of routines implementing five main functionalities: initialization, de-initialization, control loop, dataset update and commands execution.

The Control Library, by means of its *managers*, will provide both the environment for the execution of the Control

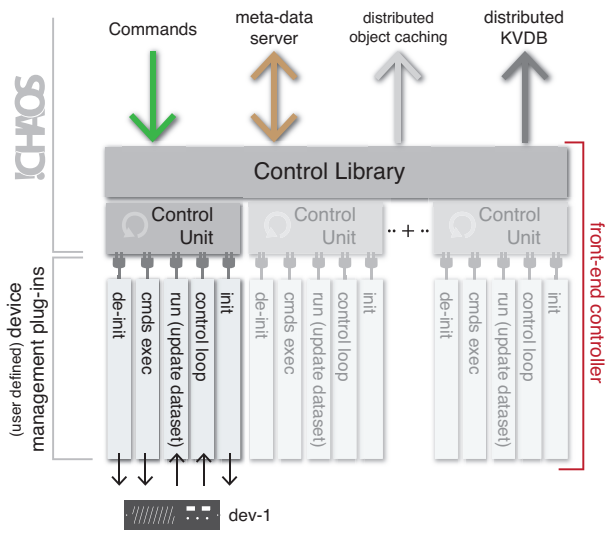

Figure 2: !CHAOS components for the front-end controllers.

Units and the functions needed to access the centralized services (DOC, KVDB, MDS).

Since the CU will control the execution of DMPs, the former will be responsible of invoking the *run* module, according to the refresh rate defined for that device, for reading the device status. The data returned will be used by the CU, via the Control Library, to feed the KVDB and to refresh the value of the correspondent key/value pair in the distributed object caching service.

On the other side, when a command issued by a client application will be received by the CU, the *command* module will be invoked for executing it. Parameters passed to the command's execution plug-in module will specify the action to be taken according to the instructions provided. The use of separated threads assures that requested periodicity of dataset refreshing is preserved even during commands' execution while serialization of parameters passed to the command plug-in allow a common interface for all the commands to be implemented.

Live Data Caching, DAQ and Central Services

Caching of live data, by means of distributed object caching service, and continuous archiving of accelerator data, by using a distributed key/value database, are the main innovations introduced by the !CHAOS paradigm.

DOC service is distributed over many nodes working together to provide clients with a single virtual pool of solid-state memory by sharing a portion of the RAM of each node. Objects are stored in memory as *key/value* pairs and a given object is always stored and always retrieved from the same node in the cluster, unless the number of node changes for any reason.

In !CHAOS a *key* identifies a unique dataset of the control system that is the set of information used to fully describe a real, or virtual, accelerator device.

Each dataset is periodically refreshed by the Control Unit in charge for the corresponding device. In the DOC service, dataset refreshing means overwriting the old data with newer describing the actual state of the device. Refresh rate is set and adjusted independently for each device and typical values span from milliseconds to few seconds.

Similarly, for the DAQ, key/value pairs are pushed to a node of the distributed KVDB to be stored on disks. In this case the *key* contains encoded both the unique dataset indicator and the timestamp. By querying the DAQ for all the datasets corresponding to a given timestamp the status of the accelerator at that particular time can be recovered.

The software opted for implementing the DOC and the KVDB are memcached[4] and mongodb[5] respectively. They demonstrated to offer the needed features and performances and are supported by a large and growing community of users. Nevertheless the abstraction of services provided by the !CHAOS components would allow their replacement, with other implementation of DOC and KVDB, without any modification of both its functionalities and API.

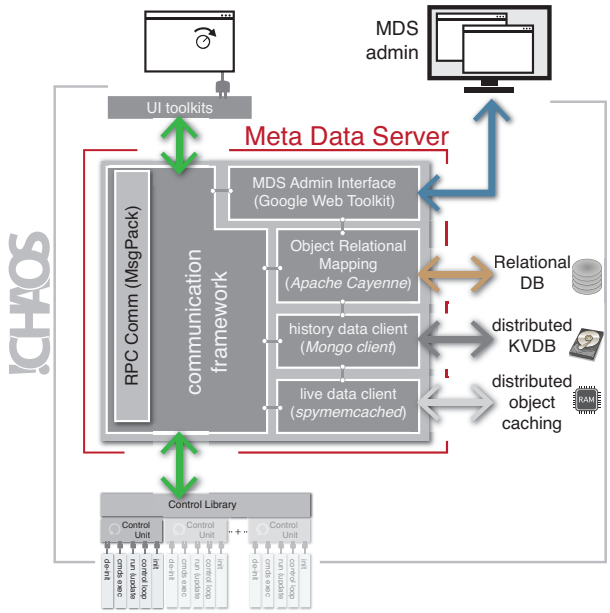

Figure 3: The !CHAOS meta-data server.

A key aspect in the !CHAOS development is the solution used to format data for both storing it, either into DOC or KVDB, and passing it between the different CS components.

Binary serialization is a convenient solution for flattening even complex data structure into a one-dimensional stream of bits suited for transmission through network. It is well suited especially for large binary arrays that are frequently included in datasets of accelerator's components.

What's more, both DOC and KVDB allows binary serialized data. In !CHAOS BSON[6] serialization is used for encoding dataset to be stored both in the *live data* DOC and in the DAQ . BSON serialization is also used by UI toolkit (see next section) for formatting commands sent to front-end Control Units and for passing parameters between CU and device management plug-ins.

Another fundamental component in the !CHAOS framework is the meta-data server (Fig.3). It is designed to store information such as CU configuration, commands list, commands and data semantic, naming service etc.

Object Relational Mapping (ORM) packages will be used to abstract the relational database, used for storing meta-data, by mapping its tables into Java object.

User Interface Toolkit

Client access to !CHAOS services will uniquely allowed through the API provided by the *User Interface toolkit*.

The set of API aiming to abstract and simplify the access of client applications to the !CHAOS service.

Fig.4 shows the logical structure of the UI toolkit layer with the blocks of API to client application and the substrate of API for the abstraction of the !CHAOS services.

In the figure is also introduced the concept of *UI data*

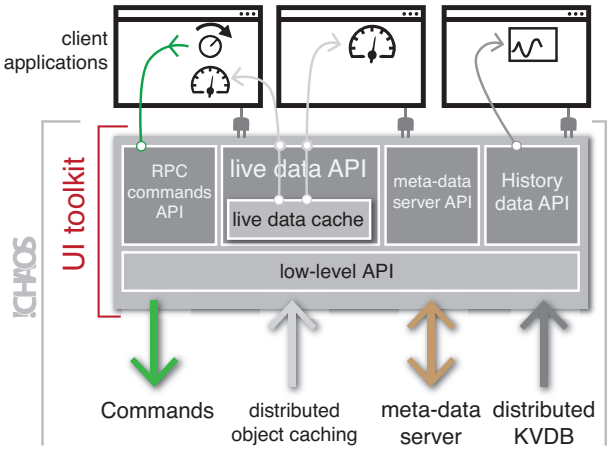

Figure 4: The User Interface toolkit components.

cache we are currently developing to achieve a further improvement of UI toolkit performance.

Practically it consist of a local cache of data and meta-data where UI toolkit APIs may store and share both frequently used meta-data, produced by queries to MDS, and live data read from distributed object cache. Similarly to distributed live data caching, we are considering a solution based on a key/value object caching to store locally this information. Caching of live data will take into account the refresh rate of the particular device dataset for setting its expiration time.

While most of the !CHAOS code is written in C, C++ and Java, development of both client applications and device management plug-ins should include a larger selection of programming languages.

Since at INFN LNF and Roma TV there is a long tradition in developing control and data acquisition systems with National Instruments LabVIEW we already started remodeling of existing front-end software to adapt it to !CHAOS DMP requirements.

On the client side, UI toolkit will provide APIs for most common measurement and analysis software like Matlab and the before mentioned LabVIEW.

CONCLUSION

!CHAOS is a scalable control system infrastructure providing all the services needed for communication, data archiving, timing, etc. in a control system for a particle accelerator or any other large apparatus. Front-end controllers and GUI applications can be seen as plug-ins that access and expand its functionalities.

The innovative communication framework is based on a distributed object caching service while continuous archiving of data is implemented by means of a non-relational distributed key/value database.

The use of the before mentioned software technologies introduces a new paradigm of control system in which the two layers representing the front-end and the client level are complemented by a third intermediate level collecting and distributing the the data produced by the lower front-end controllers.

The control groups at INFN-LNF and INFN-Roma Tor Vergata are committed to finalizing the development of this conceptual design, validating its functionalities and performance, and candidate !CHAOS as the control system for future INFN particle accelerators.

REFERENCES

[1] SuperB-CDR2 INFN-LNF-11/9(P) 15 Jun 2011

[2] G. Mazzitelli *et.al.*, "High Performance Web Applications for Particle Accelerator Control Systems", Proceedings of IPAC2011, San Sebastian, Spain, pp.2322-2324, `http://www.JACoW.org`.

[3] M.E. Thuot, L.R. Dalesio, "Control system architecture: the standard and nonstandard models," Particle Accelerator Conference, 1993., Proceedings of the 1993, pp.1806-1810 vol.3, 17-20 May 1993

[4] `http://memcached.org`.

[5] `http://www.mongodb.org`.

[6] `http://bsonspec.org`.