USING MEMCAHED AS REAL-TIME DATABASE IN THE SPARC CONTROL SYSTEM

G. Di Pirro, E.Pace, INFN LNF, Italy

Abstract

The first implementation of the SPARC control system was based on a distributed TCP/IP data server: each front-end CPU had its own server to distribute data to the console. We decided to move the system to a NoSQL key value database. We decided to use an open source database Memcached. This is a database that is high performance key-value cache optimized for speed only. For this reason we could use memcached not for storing data, but as a channel of communication between front-end processors and consoles. The first object that we have installed is the camera system. We chose this class of elements because the amount of data is high; cameras are at least 640x480 with 8 bit. In this first installation we made some speed test: we increased the speed transfer and the data transfer is now independent from the number of high level CPUs that are using the same image. The success of this installation convinced us to bring the entire data transfer of SPARC control system to use Memcahed as data server.

SPARC

The SPARC*[1] (Sorgente Pulsata e Amplificata di Radiazione Coerente, Self-Amplified Pulsed Coherent Radiation Source) (Fig.1) project is to promote an R&D activity oriented to the development of a high brightness photo injector to drive SASE-FEL experiments at 500 nm and higher harmonics generation. Proposed by the research institutions ENEA, INFN, CNR with collaboration of Universita` di Roma Tor Vergata and INFM-ST, it has been funded in 2003 by the Italian Government. The machine is installed at Laboratori Nazionali di Frascati (LNF-INFN). It is composed by an RF gun driven by a Ti:Sa laser to produce 10-ps flat top pulses on the photocathode, injecting into three SLAC accelerating and 6 undulator sections.

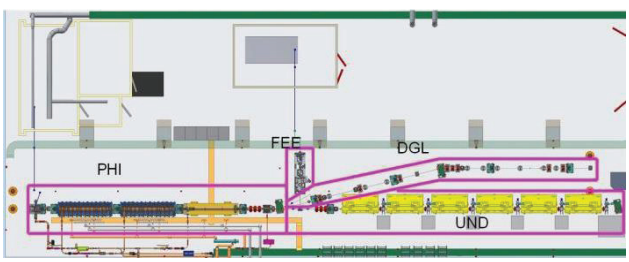

Figure 1: SPARC.

* The SPARC project is financially supported by the EU Commission in the 6th FP, contract no. 011935 EUROFEL and contract no. RII3-CT-2003-506395 CARE

CONTROL SYSTEM DESCRIPTION

The control system should guarantee and simplify machine operation. In general the main operations in an accelerator control system are: data taking, display of information, analysis, command execution and storage.
The simplest and functional control system has distributed processors on a classic three levels architecture (Fig. 2).

- *First level*: At this level we find the console with its human interface to allow the operator to control the machine, a logbook to share information within the collaboration, a database to store all information coming from the machine and a web tools to help the management of the control system and to share some information outside the collaboration;
- *Second level:* At this level we find the front-end CPU that executes commands and handle all the information about the status of the machine available at the first level. Meanwhile it automatically saves data from its various elements in two ways: on value changes and/or at fixed time intervals;
- *Third level*: This is the acquisition hardware where we find an appropriate acquisition board or the secondary field bus to acquire data from the real element.

The interconnection bus between the levels is a Gigabit Ethernet LAN.

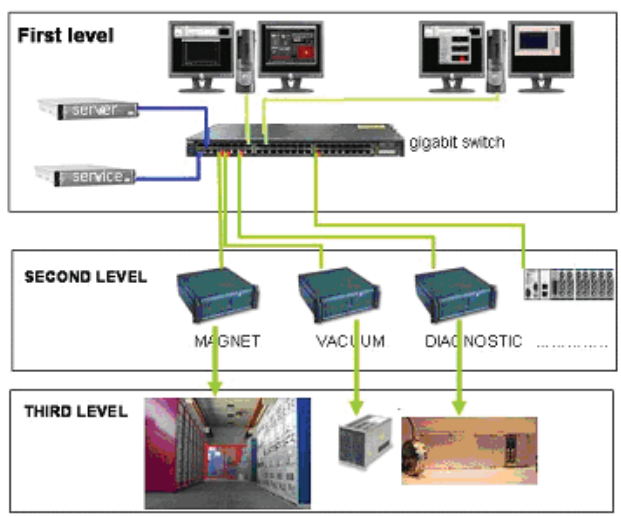

Figure 2: Control System Structure.

ELEMENTS

The control system allows to control all machine elements (from the Laser until undulator) and their diagnostic instruments.

ISBN 978-3-95450-124-3

Table 1 describes the kind of elements under control and their own interface.

Table 1: Element Controlled

Element	Number	Interface
RF Modulator	2	TCP/IP
RF Low Level	1	PXI Digitizer
Vacuum Pump	30	DAC, ADC, IO, Serial
Vacuumetr	12	Serial
MagnetPS	50	Serial, ModBus
Flag	24	Serial Motor
Camera	24	IEE1384, GigaEth
BP M	12	Bergoz ADC
BCM	2	Bergoz DVM
Faraday Cup	1	High speed digitizer
Laser Photodiode	1	High Speed Digitizer
Filter wheel	3	Serial motor

Element Abstraction

To control an element means control its hardware interface. For example to control the magnet mean control its power supply.

In this case we make an abstraction of the power supply to allow include the main characteristic of the element and we define a class.

We have two kind of cluster, (see Fig. 3) one contains all information that is necessary to control the power supply (called static part). It contains the type of interface, the eventual conversion factor and some parameter such as min max value and so on. The second cluster contains dynamic information such as the current, the voltage, the status and the error.

MEMCACHED AS COMMUNICATION INFRASTRUCTURE

In a control system is natural developing a communication channel between the front-end CPUs and the user based on a simple client server mechanism.

This communication system has a weak point in performance. The system can be influenced by the number of customers that require data to the server running on the front-end, and the possibility that the acquisition process is stopped or the computer is switched off. This second case leads to a general slowdown of the console display and in system performance due to time-out.

A method for exchanging information between a system that produces data and one using data is a database. In a relational database is the stiffness in the definition of the data that have lengths required.

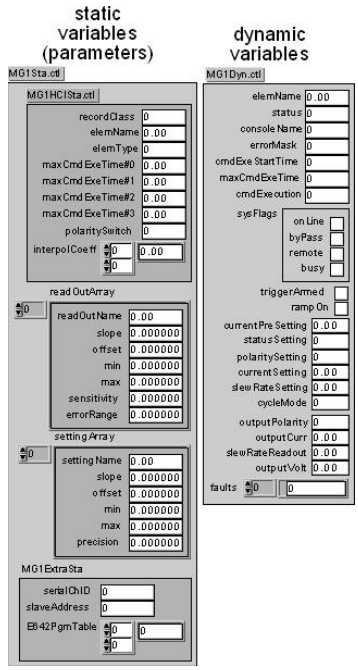

Figure 3: Magnet Clusters.

Development, especially in the field of web, non-relational database type key value NoSQL db, largely simplifies these problems. These systems allow you to associate a value with each key.

A NoSQL database allows you to store information of variable length within the same database as there is a simple association between a key and its associated data.

The non-relational databases have received an impulse in the development of Web applications and cloud where, given the high number of nodes, it is difficult to define a priori the tables. Their simplicity and no need to predefine the tables make the system very flexible in the implementation.

There are several versions of NoSQL databases in particular we want try the system based on RAM for speed reasons We chose an open source implementation of a NoSQL based on ram called Memcahed.

Memcached [2] is a NoSQL key-value database. It is a high performance key-value cache. It is intentionally a dumb cache, optimized for speed only. Applications using memcached should not rely on it for data, a standard database guarantee, but applications can run much faster when cached data is available in memcache for the display.

IMPLEMENTATION

Transferring the information we decide to transfer the whole cluster because the transfer time is similar to transfer a single information (the normal dimension of a dynamic cluster is normally less than 1500 byte). The choice of data is performed by the console software. If we need to transfer a big amount of data, i.e. image or

ISBN 978-3-95450-124-3

waveform, we developed a specific data tape. All the data are transferred in binary code,

The communication protocol with the database is easy to develop we write some subroutines directly in TCP/IP without using Memcahed libraries. This choice has allowed us to use memcahed also for operating systems where the libraries are not developed.

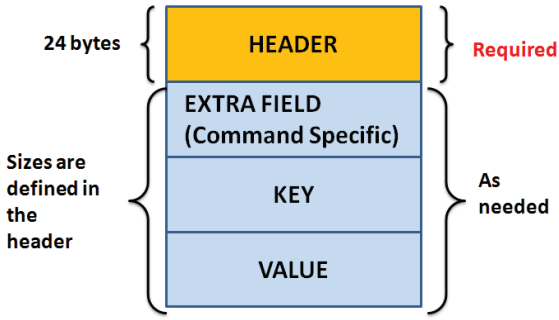

Figure 4: Packet Structure.

The protocol is simple: in Fig. 4 you can see the description of the packet. The transmission standard TCP/IP the only requested part is the header part all other are option. There are two kind of header one for request information (Fig. 5) and we have as answer header data (Fig. 6). The answer has different length in function of the request.

MAGIC (1 byte)	Opcode (1 byte)	Key Length (2 bytes)
Extra Length (1 byte)	Data Type (1 byte)	Reserved (2 bytes)
Total Body Length (4 bytes)		
Opaque (4 bytes)		
CAS (Compare and Swap) (8 bytes)		

Figure 5: Request Header.

We have developed a library in Labview that allow data transfer to the database so that easily integrate in our system of control.

Each element has been associated with a key and the data were written in binary.

MAGIC (1 byte)	Opcode (1 byte)	Key Length (2 bytes)
Extra Length (1 byte)	Data Type (1 byte)	Status (2 bytes)
Total Body Length (4 bytes)		
Opaque (4 bytes)		
CAS (Compare and Swap) (8 bytes)		

Figure 6: Response Header.

The first object that we have tested is the camera system. We chose this class of elements because the amount of data is high (our camera is at least 640x480 with 8 bit). In this first installation we make some speed test: we increased the speed transfer and we have the data transfer independent from the number of consoles taking the image. The success of this installation convinced us to bring the entire data transfer of SPARC control system[3] to use memcache as data server.

The performance of the image transfer is very good we can transfer any kind of real time image with the follow time:

Image 640x480 16 bit 40 ms sample variance 8 ms
Image 640x480 8 bit 23 ms sample variance 8 ms

We do not see variation on performance from 1 to 4 consoles in simultaneous acquisition from the camera.

CONCLUSION

The use of database NoSQL there has demonstrated both in performance and in simplicity of realization. So it is possible to use such database basic data for a control system.

The future development will be in the evaluation into using a database like this but hard as a database for the data history.

ACKNOWLEDGEMENTS

We want to We are also grateful to all the SPARC staff for their continuous suggestions and encouragement for making a good and useful job.

REFERENCES

[1] SPARC Project Team, Sparc Injector TDR http://www.lnf.infn.it/acceleratori/sparc
[2] http://memcached.org/
[3] G. Di Pirro et al., "The SPARC Control System", THP042, http://jacow.org, Proc. of ICALEPCS2009, Kobe, Japan.

ISBN 978-3-95450-124-3