IMPROVEMENT OF EPICS BASED ACCELERATOR CONTROL SYSTEM RELIABILITY: REDUNDANT IOC FOR ATCA AND EPICS TEST AUTOMATION

A. Kazakov, K. Furukawa, M. Satoh, T. Suwada, S. Michizono, KEK, Tsukuba, Japan
M. Clausen, G. Liu, DESY, Germany
A. Johnson, APS, Argonne NL, USA

Abstract

Modern accelerators require reliable control systems. With constantly growing size and complexity of control systems the reliability and availability concerns become essential. Requirements for the modern systems are to provide 99-99.9% availability (which means several days of down time yearly). In order that some subsystems like power supply, cryogenic plants should virtually never fail (99.(9)% availability). Therefore, redundancy is a must in such places. This work describes implementation of redundant EPICS IOC (RIOC) with support for ATCA. ATCA is a modern and reliable platform, which is suggested for ILC control system. RIOC, in conjunction with ATCA hardware, promise to provide very highly reliable systems. Another important question is software quality. Modern control systems are complex software systems, consisting of many simultaneously running software. An automated system test package for EPICS was developed to allow transparent testing process over different operating systems and configurations.

APPROACHES TO HIGH RELIABILITY

Accelerator Control System is a complex organization of hardware, software, humans and "procedures". Each of these components can be targeted separately or together in order to improve reliability. I.e. for humans we can provide better training, create user-friendly and foolproof software, good and up-to date documentation, logging and monitoring facility can influence human efficiency as well. Hardware: monitoring and logging, good design allowing easy maintenance and service in case of a failure, availability of spare parts. For the most critical parts of the system (such as cryogenic plants) redundancy must be implemented. Software is also critical component. Due to its nature it has bugs from the very beginning and replacement of the program does not fix the problem, in contrast to hardware. Therefore thorough testing of the software is essential to build reliable system.

REDUNDANT IOC

An EPICS redundant IOC was originally developed at DESY. Two major fields of application were defined:

1. Redundancy for cryogenic plants. In this case, the failure may be caused by malfunctioning hardware such as power supplies or fans. An automatic fail-over mechanism should guarantee system stability. However, over the years it was sometimes necessary to manually switch between the main and backup processors due to maintenance work during the runtime period (which is usually one year or more). It might be useful for a software update. Current EPICS does not allow addition or deletion of records and databases during operation.

2.Redundancy for controllers in the XFEL tunnel. Although the main origin of switchover in the first case would be a manual action, it is expected to occur automatically in the XFEL tunnel. Due to high radiation, damage to the CPU and memory is highly possible. The software update is not very important because of more frequent maintenance days when this operation may be performed.

By the design draft one major goal was set: **Any redundant implementation must make the system more reliable than the non-redundant one**. Precaution must be taken especially for the detection of errors that shall initiate the fail-over. This operation should only be activated if there is no doubt that maintaining the actual mastership definitely causes more damage to the controlled system than an automatic fail-over. The fail-over time in any case was defined to be more than several seconds and less than 15 s. The system implemented shows failover times less then 2s.

Hardware Architecture

The hardware architecture consists of two redundant IOCs controlling a remote I/O via shared media such as the Ethernet. The redundant pair shares two network connections for monitoring the state of health of their counterpart, where the private network connection is used to synchronize the backup to the primary and the global network is used to communicate data from the primary to any other network clients requiring the data.

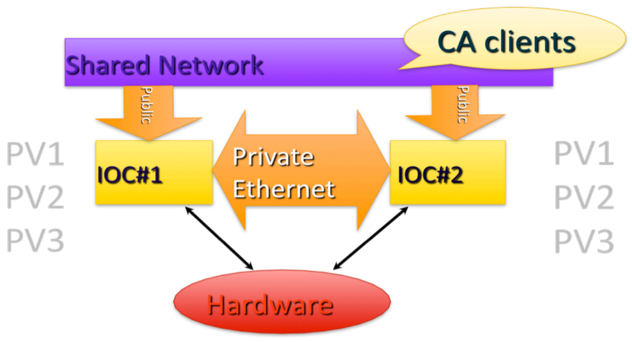

Figure 1: Redundant IOC hardware architecture

Software Components

An EPICS redundant IOC consists from four major parts Redundancy Monitoring Task (RMT), Continuous Control Executive (CCE), State Notation Language Executive (SNLE) and IOC-part (which is same as non-redundant IOC). RMT – is a key component of that system. RMT is responsible for monitoring all other parts of the system, checking connectivity and making decisions regarding fail-over. Other parts of the system (CCE, SNLE, IOC) are controlled by RMT and are called "RMT drivers" accordingly. Any other software which is we want to make redundant has to implement RMT-driver API interface. Both standalone and IOC-related software such as device drivers can be made redundant using RMT. One of the functions of RMT is to check status of its drivers and exchange this information with its partner. Each RMT driver implements its own logic for checking whether it is "ok" or "not ok" depending on the particular driver and implementation. Exactly this approach was used to make ATCA-aware extension for RIOC.

CCE and SNLE are RMT-drivers that synchronize IOC-database and SLE programs between peers. Those parts are not relevant to the topic of this paper, so we will not go into the details of their functionality and implementation.

ADVANCED TELECOM COMPUTING ARCHITECTURE – ATCA

ATCA standard is defined by PCI Industrial Computer Manufactures Group with 100+ companies participating. It is relatively new standard, but it has been already widely supported by major vendors. It is primarily targeted to requirements of carrier grade communications equipment and incorporates the latest trends in high speed interconnect technologies, next generation processors and improved reliability, manageability and serviceability [1]. Availability of ATCA crate is designed to be 99.999%. That and other features of ATCA standard made it the platform of choice for the ILC control system.

Smart Hardware

ATCA defines extensive monitoring and controlling capabilities. All the components of an ATCA system are interconnected via (usually) redundant Intelligent Platform Management Bus (IPMB).

One particular board – Shelf Manager (SM) - plays a centre role in the management of the system. SM is responsible for polling and controlling other hardware via IPMB. Usually SM implements some simple logic for monitoring overall system health and some fail-over procedures (which are vendor and hardware specific).

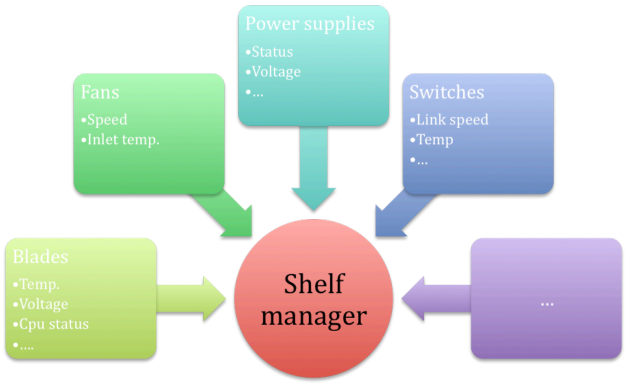

Figure 2: Shelf manager

SM provides access to the hardware for third-party via Hardware Platform Independent library (HPI) or SNMP. HPI covers all the differences in actual implementation of the ATCA standard and provides hierarchal representation of all available hardware.

RIOC AND ATCA

It is possible to run "vanilla" RIOC on ATCA hardware, but it will be no different from running it on two separate "normal" PCs, though we may get some indirect benefit from using reliable ATCA platform. But we may get much more if we make RIOC aware of the hardware its running on.

That ability to monitor the hardware of the system allows us to improve reliability of RIOC. By implementing ATCA-driver for RMT, we can include the hardware status of a particular board or the whole system into the fail-over decision procedure. For example if a CPU temperature starts to rise, there is some limited time before it will crash. And if properly monitored, we can initiate the fail-over process before the actual hardware failure happens. For the RIOC applications it gives us two major benefits:

1. Failover happens while the system is still working, so actual transition happens in a stable and controlled environment.
2. Channel Access connections can be gracefully closed. That will reduced reconnect time for Channel Access Clients (CAC) drastically. Normally in case of "hard" failure of a master RIOC it takes up to 30 seconds for CAC to reconnect to the slave RIOC (30 seconds is a default EPICS connection time-out, actual time-out may be changed by user). Even though slave RIOC notices the problem instantly and takes over within 2 seconds.

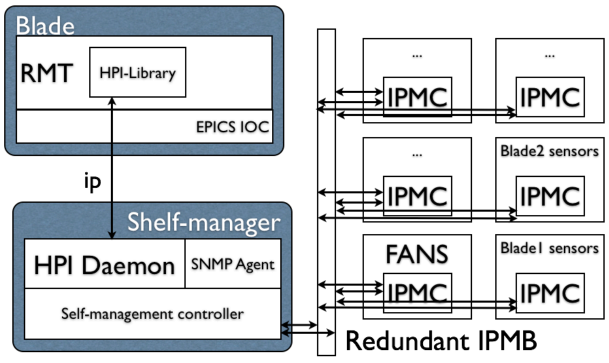

Figure 3: ATCA aware RIOC

Conclusion

We used a software solution (RIOC) aimed for highly available and critical tasks and run it on the hardware specifically designed to be very reliable (ATCA). RIOC functionality was extended to use hardware sensors information to make better failover decisions. The choice of HPI library resulted in ATCA independent implementation. Actual RMT-driver can be used on any platform, which provides HPI library: for instance modern server systems running Linux (HPI on top of sysfs) or IBM blade-servers (HPI on top of SNMP).

EPICS TEST AUTOMATION

The Problem

EPICS supports multiple Operating Systems since release 3.14 when EPICS libOSI (Operating System Independent Library). Both client and server applications can run on Linux, Windows, Mac OS, Solaris, vxWorks, RTEMS, osf-alpha, FreeBSD. It is very common situation when even within one laboratory more than one OS is used in EPICS environment. Versions of these OS also may differ very widely. And from version to version, or from one distribution to another there may be so many significant and not so much differences. Besides OS differences, EPICS can run on a wide variety of hardware platforms. All this makes it virtually impossible to test all the possible combinations of operating systems and hardware. Of course during the release phase EPICS distribution is tested by the core-team, but this test does not cover all the usage cases of EPICS in the real world.

There is a package called mrkSoftTest, which is system integration test package for EPICS. It includes several tests, which check CA network links, alarm functionality, correctness of conversion functionality between remote nodes, etc. Typical test scenario involves one or more IOCs and several CA clients.

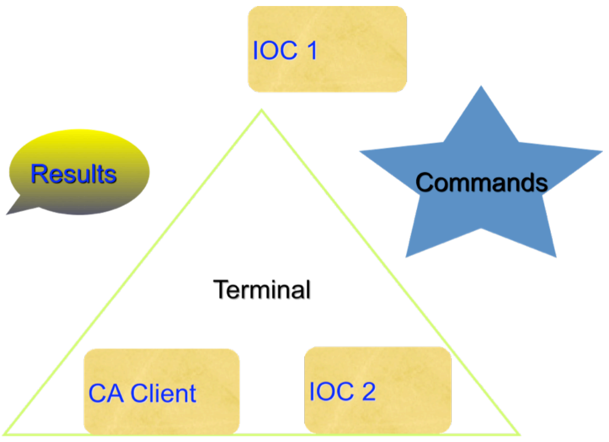

Figure 4: Typical test scenario includes many components

The person who runs the test has to follow these steps:

1. Connect to remote station via SSH/Telnet/ SH
2. Configure & start EPICS IOC
3. Repeat steps 1-2 for require number of times to start other IOCs
4. Start CA client(s), locally or remotely
5. Issue the required sequence of control commands for IOC and/or start additional CA clients.
6. Gather the output from all the programs
7. Compare the output to the reference file.
8. Shutdown all the IOCs and CA clients.

This procedure is quite complex by itself, but becomes more complicated when we introduce differences between OS and Hardware. Commands and output may differ for different architectures and particular configuration.

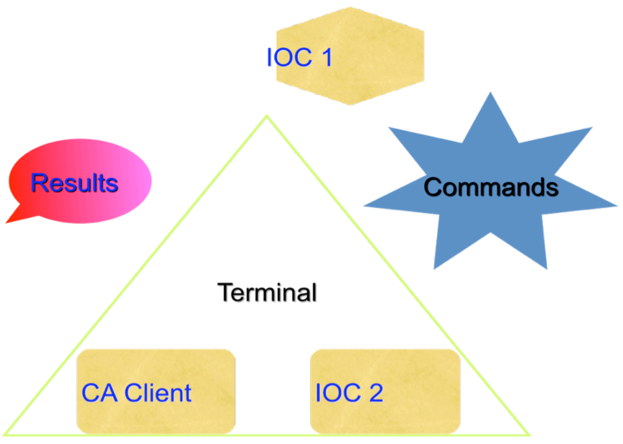

Figure 5: Commands and output may differ according to the hardware and particular configuration

Most people do not remember how to run these test, and have to read the instruction every time they run these test. Overall this becomes very time and effort consuming operation.

Test Automation

To solve this problem EPICS test automation package (EPICS TAP) was developed. EPICS TAP simplifies the process of running system integration tests and hides all the low level details from the person running test. EPICS TAP was developed using high level scripting language Ruby.

Introduction of EPICS TAP made running test as simple as "./runAlltests.rb" and then wait until it finishes. Depending on the configuration some changes are required for the configuration file (such as ip-addresses and OS system types). EPICS TAP ruby classes can be used to write additional test cases as well. EPICS TAP provides fully automated infrastructure for testing EPICS implementations.

REFERENCES

[1] AdvancedTCA, PCIMG 3.0 Short Form Specification