Improving Reliability: Redundant IOC for ATCA and automation of EPICS system tests

Accelerator Reliability Workshop 2009

Artem Kazakov, KEK/SOKENDAI
How to improve Reliability of a Control System:

- Design with high availability in mind
- Use better Software
- Implement redundancy for critical parts
- Etc...
Outline:
Redundant EPICS IOC for ATCA

• Brief introduction of EPICS, IOC, RIOC
• What is ATCA and why do we want it?
• RIOC + ATCA features and benefits
• Conclusion
EPICS: Experimental Physics and Industrial Control System

- EPICS is a set of software tools and applications which provide a software infrastructure for use in building distributed control systems
- It is used to operate Particle Accelerators, Large Experiments and Telescopes
EPICS

- Client/Server model
- Communication protocol: Channel Access (CA)
  - CAC: client
  - IOC/CAS: server
- PV: Process Variable
  - Named Piece of Data
Redundant IOC

• Provides redundancy support for EPICS IOCs
• Developed at DESY, Germany
• Supported in “official” EPICS distribution since v3.14.10 release
  – No need to patch/reconfigure/recompile BASE
  – Just download RIOC libs and link them to your IOC to make it redundant
What is redundant IOC?

Shared Network

PV1
IOC#1
PV2
Private Ethernet
PV3

Hardware

IOC#2

CA clients

PV1
PV2
PV3
Redundancy Monitoring Task (RMT) - Key component of RIOC

- Controls drivers
- Monitors “health” of the drivers
- Checks the partner status
- Decides when to failover (or not to)
RMT – Key component of RIOC

• Independent from EPICS core facilities
  – It uses libCom though

• Defines RMT driver interface API
  – Which is very simple and easy to use

• Can be used to make other software redundant
  – i.e. caGateway
Advanced Telecom Computing Architecture (AdvancedTCA)

- Defined by PCI Industrial Computer Manufacturers Group with 100+ companies participating
- Targeted to requirements for the next generation of carrier grade communications equipment
- Incorporates the latest trends in high speed interconnect technologies, next generation processors and improved reliability, manageability and serviceability
AdvancedTCA chassis and blades
Why run RIOC on ATCA?

• ATCA is a modern industry standard for HA applications
  – Supposed to be very reliable (99.999% design availability)
• ATCA is suggested as a platform for the ILC control system
• ATCA is a hardware designed for critical applications and RIOC is a software designed for critical applications
ATCA Features

ATCA provides monitoring and management controls for many parts of the system: fans, network connection, power supplies, bios images, boot ROMs etc...

The key role in this process is played by Shelf Manager

We want to use this features to make better decisions for fail-over
ATCA Shelf manager

Data is exchanged through redundant Intelligent Platform Management Bus (IPMB).
“plain” Redundant IOC on ATCA
“plain” Redundant IOC on ATCA

• Runs “as-is”

• But does not know anything about the “smart” hardware of ATCA

• Basically is same as running on two normal PCs
Benefits of “ATCA”-aware RIOC

• Failures can be “predicted”
  – i.e. temperature starts to rise and the CPU is still working -> we can initiate fail-over procedure before actual hardware fails -> fail-over occurs in more stable and controlled environment

• Client connections can be gracefully closed
  – Allowing the client to reconnect to back-up IOC within 1 second
  – In case of “real” hardware failure reconnect would occur only after 30 seconds
ATCA/HPI driver for RMT

Shelf Manager
- HPI Daemon

IP

RMT
- HPI Client Library

HPI - Hardware Platform Interface – Generic Platform
Independent specification to monitor and control HA systems

28.01@ARW2009

RIOC on ATCA & EPICS Test Automation
“HPI-aware” RIOC on ATCA

Blade

RMT

HPI-Library

EPICS IOC

ip

Shelf-manager

HPI Daemon

SNMP Agent

Self-management controller

Redundant IPMB

... 

IPMC

Blade2 sensors

... 

IPMC

FANS

Blade1 sensors

... 

IPMC

... 

IPMC

IPMC
Result:

• Reliable Hardware (ATCA) was used in conjunction with Reliable Software (RIOC)
• RIOC was extended to use available hardware sensors to make better failover decisions
• The software can be used on other hardware (i.e. “common” server-type PC): the requirement is HPI library, which can run on top of IPMI, SNMP, Sysfs(linux)…
Now RMT can monitor any available sensor on ATCA shelf and make better fail-over decision

collection via iocSh:
rmtHPIDriverStart
"{RACK,0}{ADVANCEDTCA_CHASSIS,0}{PHYSICAL_SLOT,4}{PICMG_FRON T_BLADE,0}" 1

rmtHPIDriverStart "entityPath" “Sensor ID”
How to improve Reliability of a Control System:

• Design with HA in mind
• Use better Software
• Implement redundancy for critical parts
• Etc...
The Problem

• EPICS can run on many different OS (Linux, Windows, Mac OS, FreeBSD, Solaris, vxWorks, RTEMS, osf-alpha)

• Usually even within one laboratory more than one OS is used

• OS versions are also different

• We need to test all the configurations being used in real control systems
EPICS existing system test package (mrkSoftTest)

- Nobody remembers how to run these tests, so every time people have to read instructions => most people do not run tests
- You have to configure, execute, compare results MANUALLY
- it’s inconvenient and **IT TAKES TIME**
Typical Test Scenario: configure & start components

Terminal

Commands

IOC 1

IOC 2

CA Client

Typical Test Scenario: configure & start components
Typical Test Scenario:
wait for the results

Terminal

IOC 1

CA interaction

CA Client

IOC 2
Typical Test Scenario:
gather results & analyze

Results

IOC 1

IOC 2

CA Client

Terminal
Typical Test Scenario: commands depend on configuration

Terminal

IOC 1

IOC 2

CA Client

Results

Commands
Typical Test Scenario: commands depend on configuration

Terminal

Results

IOC 1

Commands

CA Client

IOC 2

Typical Test Scenario: commands depend on configuration
The Answer: Automation

• Just say “run all tests for me” :
• Actually you say: ./runAllTests.rb
• Currently automated tests can be run on local/remote machine over sh, ssh, rsh, telnet, cu ... any other “shell-like” program
How to create new test:

• Develop the test: ioc, clients, reference results file, etc..

• write the corresponding section in the config file: config.yml

• write “test scenario” using provided ruby classes
Config.yml

:TestGeneric:
  <<: *default
  IocBootDir: iocBoot/iocput
  Cmd: put.main
  reference: testcache26MAR2008.darwinx86
Simple example

class TestGeneric < RubTap::TestCase
  @ioc = common_setup_local
  def test_name
    @ioc.talk("< p2sec", 5)
    cache_response = @ioc.talk("< testcache", 20)
    assert_equal(@testcache_reference,cache_response)
  end
Running tests

• Group tests into a test suite

```ruby
ts = RubTap::TestSuite.new
ts<<TestGeneric.new(TestConfig::AllTestsConfig.new("config.yml"))
ts.run
```
EPICS Test Automation package

- Was used to automate the existing test package, making it much more user-friendly and easy to run
- Can be used to write new automated tests
- on-going work: to make it more “human” readable
- git://github.com/akazakov/epicstest.git