

# The Performance and Scalability of the back-end DAQ sub-system

*I. Alexandrov<sup>1</sup>, V. Amoral<sup>2</sup>, A. Amorim<sup>2</sup>, E. Badescu<sup>3</sup>, D. Burckhart<sup>4</sup>, M. Caprini<sup>4,9</sup>, L. Cohen<sup>5</sup>, P-Y. Duval<sup>5</sup>, R. Hart<sup>6</sup>, R. Jones<sup>4</sup>, A. Kazarov<sup>7</sup>, S. Kolos<sup>4,10</sup>, V. Kotov<sup>1</sup>, D. Laugier<sup>5</sup>, L. Mapelli<sup>4,11</sup>, L. Moneta<sup>8</sup>, Z. Qian<sup>5</sup>, C. Ribeiro<sup>2</sup>, V. Roumiantsev<sup>1</sup>, Y. Ryabov<sup>7</sup>, D. Schweiger<sup>4</sup>, I. Soloviev<sup>4,10</sup>*

- <sup>1</sup> Joint Institute for Nuclear Research, Dubna, Russia
- <sup>2</sup> Lisbon Institute of Physics, Lisbon, Portugal
- <sup>3</sup> Institute of Atomic Physics, Bucharest, Romania
- <sup>4</sup> CERN, Geneva, Switzerland
- <sup>5</sup> Centre de Physique des Particules de Marseille, IN2P3, France
- <sup>6</sup> NIKHEF, Amsterdam, Netherlands
- <sup>7</sup> Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia
- <sup>8</sup> Section de Physique, Universite de Geneve, Geneva, Switzerland
- <sup>9</sup> On leave from 3.
- <sup>10</sup> On leave from 8.
- <sup>11</sup> Spokeperson for DAQ-1 project

## Abstract

The DAQ group of the future ATLAS experiment has developed a prototype system based on the Trigger/DAQ architecture described in the ATLAS Technical Proposal [1] to support studies of the full system functionality, architecture as well as available hardware and software technologies. One of its sub-systems is the back-end [2] which encompasses the software needed to configure, control and monitor the DAQ, but excludes the processing and transportation of physics data.

This paper gives an overview of the back-end software, its performance and scalability tests, their current status and future developments.

Keywords: DAQ, back-end, performance, scalability

## 1 Introduction

The goal of the ATLAS data acquisition (DAQ) and Event Filter (EF) prototype “-1” project is to produce a prototype system representing a “full slice” of a DAQ suitable for evaluating candidate technologies and architectures for the final ATLAS DAQ system on the LHC accelerator at CERN. Within the prototype project, the back-end sub-system encompasses the software needed to configure, control and monitor the DAQ, but excludes the processing and transportation of physics data. The back-end software is essentially the “glue” that holds the sub-systems together. It does not contain any elements that are detector specific as it will be used by all possible configurations of the DAQ and detector instrumentation. The DAQ system includes interfaces required by the triggers, processor farm, accelerator, event builder, data-flow and detector control system (DCS).

## 2 Back-end architecture

In this section there is description of the back-end components, their design principles and back-end operational environment.

### 2.1 Back-end components

The user requirements gathered for the back-end sub-system have been divided into groups related to activities providing similar functionality. The groups have been further developed into

components of the back-end with well defined purposes and boundaries. The components have interfaces with other components and external systems, specific functionality and their own architecture. The components have been grouped into two sets: core components and trigger/DAQ and detector integration components.

### 2.1.1 Core components

The core components of the back-end sub-system constitute its essential functionality and they had priority in terms of time-scale for development in order to have a baseline sub-system that can be used for integration with the data-flow sub-system and event filter. The following components are considered to be the core of the back-end sub-system:

- **Configuration databases** are used to describe a large number of parameters of the DAQ system architecture, hardware and software components, running modes and status. One of the major design issues of Atlas DAQ is to be as flexible as possible, parameterized by the contents of the configuration databases.
- **Message reporting system (MRS)** provides a facility which allows all software components to report messages to other components in the distributed environment. The MRS performs transport, filtering and routing of messages.
- **Information service (IS)** provides an information exchange facility for software components. Information (defined by supplier) from many sources can be categorized and made available to requesting applications asynchronously or on demand.
- **Process manager (PMG)** performs basic job control of software components. It is capable of starting, stopping and monitoring the status (e.g. running, exited) of components independent of the underlying operating system.
- **Run control (RC)** is used to control the data taking activities by coordinating the operations of the DAQ sub-systems, back-end software and external systems. It has a user interface for the shift operators to control and supervise the data taking sessions. It has software interfaces with other DAQ sub-systems and back-end components to exchange commands, status and control information.

### 2.1.2 Trigger/DAQ and detector integration components

The following components are required to integrate the back-end with other online sub-systems and detectors:

- **Resource manager (RM)** allocates resources (hardware and software resources which can't be freely shared) and allows several groups to work in parallel without interference.
- **Partition manager** extends RM to allow simultaneous work of several partitions.
- **Test manager (TM)** organizes individual tests for hardware and software components (the individual tests themselves are not the responsibility of the TM which simply assures their execution and verifies their return status).
- **Diagnostics package (DS)** uses tests held in the test manager to diagnose problems and verify functioning status of separate components or the entire system (DS verification component) and to control the system, diagnose and recover problems during different phases of system functionality in automatic or operator assistance modes (DS supervision component).
- **Integrated graphical user interface (IGUI)** allows the operator to control and monitor the status of the current data taking run in terms of its main parameters, detector configuration, trigger rate, buffer occupancy and state of the sub-systems.
- **Online bookkeeper** archives information about the data recorded to permanent storage by the DAQ system. It records information on a per-run basis and provides a number of interfaces for retrieving and updating the information.
- **Event dump** samples events from the data-flow to present them to the user in order to verify event integrity and structure.

## 2.2 Operational environment

It is expected that this environment will be a heterogeneous collection of UNIX workstations, PC running Linux or Windows NT and embedded systems running various flavours of UNIX operating systems with real-time features (e.g. Lynx OS) connected via a local area network.

The back-end software has been developed in C++ and ported to several compilers on the Solaris, Linux, HP-UX, Windows NT and Lynx operating systems.

## 2.3 Components design

The various components describing above all require a mixture of facilities for data storage, inter-process communication, graphical user interface, complex logic-handling and general operating system services. Candidate freeware and commercial software packages were evaluated to find the most suitable products for each technology. C++ with a general purpose library, called Tools.h++, is the primary programming language. The Objectivity/DB object-oriented database and custom-made in-memory persistent object manager (OKS) are used for data persistence<sup>1</sup>. CORBA (ILU) and a custom-made package on top of it (IPC) are used for communication. Finite state machines are used for implementing object behaviour (CHSM). The CLIPS expert system tool is used as a basis for diagnostic system. Motif and Java are used to implement graphical user interfaces.

## 3 Test Results

Performance and scalability tests have been made for individual components (including configurations comparable to those required for the final ATLAS system), and, more recently, for the complete back-end sub-system integrated with some components of the data-flow sub-system.

### 3.1 Components test results

A brief overview of individual component test results are presented below (see [3] for more information).

#### 3.1.1 Configuration databases

The configuration databases are used by many components during initialization and their performance is very important for the system start-up time. The performance benchmark [4] has been made for several OKS configurations including a single read-out crate, a typical prototype -1 configuration (10 crates) and the expected final ATLAS configuration (200 crates). By the benchmark's results, the average workstation requires less than one second to initialize the prototype -1 DAQ configuration and about half of one second to perform complete traverse and shutdown. The average single board VME based front-end computer is about three times slower.

#### 3.1.2 Information service and message reporting system

The performance and scalability of information systems are important during all phases of system operation. Both MRS and IS are scalable since they allow multiple servers to split the work with different clients or information domains across them. The benchmark results [5], [6] made with single servers show that on average workstations the response time to report a single message or update a piece of information is a few milliseconds depending on its size and the number of senders and receivers.

---

<sup>1</sup> The back-end DAQ can run without Objectivity/DB by relying solely on OKS

### 3.1.3 Process manager

The performance of PMG is important during system initialization and shutdown. The results [7] show that the time to start new processes slowly increases with the number of managed processes and is about a few hundred milliseconds on an average workstation. The time to kill a process is constant at less than 100 ms.

### 3.1.4 Run control

The performance of RC is important when the system needs to change its state. The RC is scalable by changing the structure of the control tree hierarchy (a controller may control any number of nodes). The results [8] obtained on average workstations show that the time to change a system's state with several tens of nodes varies from several hundreds of milliseconds up to a few seconds depending on the state of the system. The overhead of starting/stopping a succession of runs is less than one second.

## 3.2 Back-end sub-system test results

The back-end sub-system integration tests [9] bring together all the core components and several trigger/DAQ/detector integration components to simulate the control and configuration of data taking sessions. The tests have been made using a shell script that goes through different phases as shown on Figure 1:

- start the back-end server processes to initialize communication services and PMG
- launch configuration specific processes via DAQ supervisor as described in the database
- marshal the hierarchy of RC controllers through different states (initialized -> loaded -> configured -> running -> configured -> running -> configured -> loaded -> initialized)
- stop configuration specific processes via the DAQ supervisor
- stop the back-end server processes

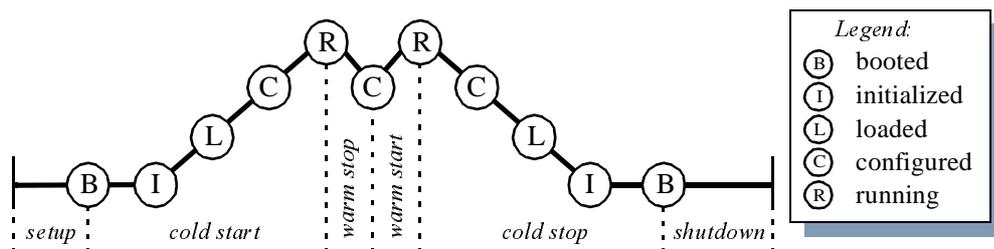


Figure 1: Activity diagram showing the actions performed by the benchmark script

For all these tests the back-end communication server processes, DAQ supervisor and RC root controller were always started on the same workstation (i.e. the host on which the benchmark script was launched). The others processes (PMG agent, local DAQ emulator and corresponding RC controller) were started on separate PCs running linux, or VME based PowerPC CPU boards running Lynx OS. The computers were not dedicated to the benchmark and they were used by other developers at the same time.

The tests were performed several times for each configuration and the average values for each test were calculated. The standard deviation of the tests strongly depends on loading of the computers and may reach a few tens of percents for lengthy operations (e.g. setup and stop).

The PMG agents are started during the setup stage via a remote shell that requires long delays (20 seconds) for synchronization purposes. In test-beam and production use the agents will be started during the boot of the machines.

The results of such tests are presented below for different configurations and types of operations<sup>2</sup>:

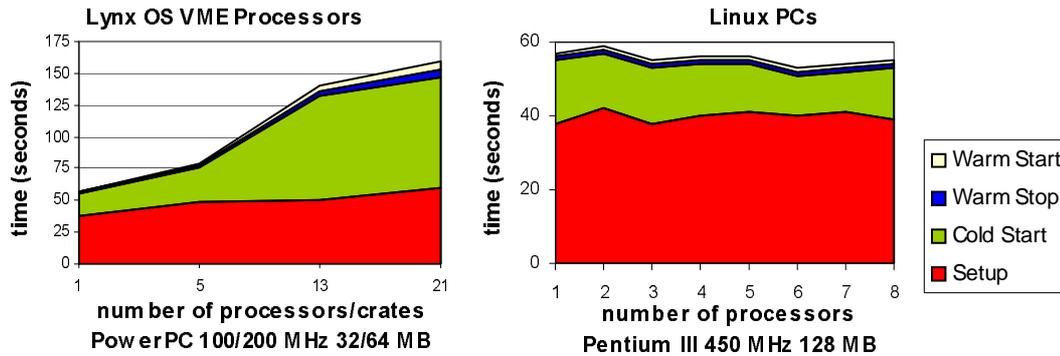


Figure 2: Tests results for start-up and warm stop/start operations

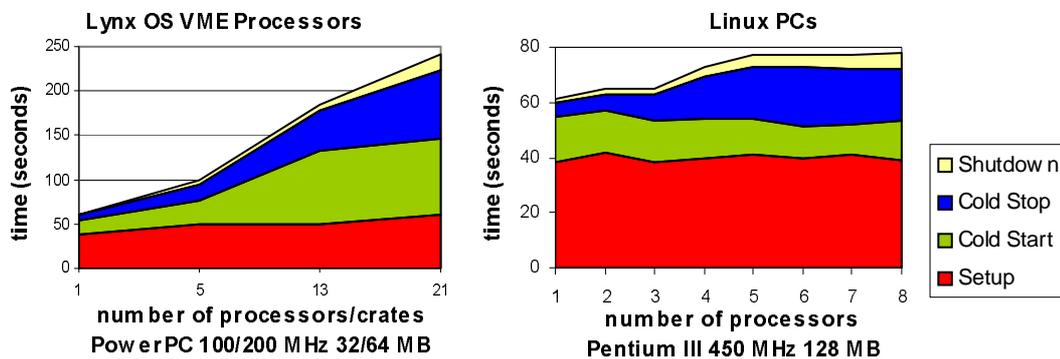


Figure 3: Tests results for start-up and close operations

From the results the following insights can be obtained:

- The time to start/stop processes is dependent on the operating system, computer architecture and configuration.
- Once all the processes have been started, the time taken to change system state remains constant which indicates good scalability of the distributed control provided by the RC.
- The use of IS, MRS and configuration databases has a negligible effect on the performance.
- The test results show that even for the largest configurations the benchmark execution time is in an acceptable range (maximum one minute to start-up a linux based system).

#### 4 Summary and future

The individual performance and scalability unit tests have been made for all back-end core components and show that they are in accordance with the DAQ/EF prototype requirements. The work will continue to perform similar tests for the back-end integration components.

Following individual component unit tests, integrated tests have been performed employing the majority of the components. The goal of such tests is to verify the correct inter-operation of the components, the ability to operate in a distributed, heterogeneous multi-platform environment and gather performance measurements relevant to the operation of the DAQ in a production environment.

<sup>2</sup> Start-up operation means “setup” and “cold start”. Close operation means “cold stop” and “shutdown”.

Future testing is required to gather more statistics for various configurations (especially on larger configurations if access to suitable hardware is possible). It will be necessary to detect more precisely the reasons for the inferior performance of Lynx OS compared to Linux. Future work to further improve the performance of the back-end will concentrate around the scripts running the DAQ system, DAQ supervisor and start-up of PMG agents to make more use of available synchronization techniques and not rely on delay timeouts.

In the future we will replace the commercial Tools.h++ general purpose C++ library with STL. We will also investigate alternative CORBA implementations that provide a better support for the latest version of the standard and more layered services. We are currently investigating using XML as a replacement storage technique inside the OKS in-memory database package.

## 5 Acknowledgments

The authors would like to thank all of their colleagues in the ATLAS prototype DAQ project for providing valuable input during the development of the back-end components. We are indebted to our initial external users, namely Murrough Landon et al., for their valuable feedback on this software and the associated documentation. We also would like to thank our system managers, Corine Costaz and Tony Wildish, for keeping the computers humming.

## 6 References

- 1 ATLAS Technical Proposal, CERN/LHCC/94-43 (ISBN 92-9083-067-0)
- 2 I.Alexandrov et al., "Back-end sub-system of the ATLAS DAQ prototype", CHEP-98, Chicago
- 3 I.Alexandrov et al., "Performance and Scalability of the Back-end sub-system in the ATLAS DAQ/EF Prototype", RT-99, Santa Fe, 1999
- 4 I.Soloviev, "Test Report of the Configuration Databases for the Atlas DAQ Prototype-1", ATD TN #114, 1999, (see <http://atddoc.cern.ch/Atlas/Notes>)
- 5 E. Badescu, M. Caprini, S. Kolos, "Test Report of the Information Service for the Atlas DAQ Prototype -1", ATD TN #118, 1999, (see <http://atddoc.cern.ch/Atlas/Notes>)
- 6 D.Burckhart, M.Caprini, A.Radu, "Unit Test Report of the Message Reporting System for the Atlas DAQ Prototype-1", ATD TN #121, 1999, (see <http://atddoc.cern.ch/Atlas/Notes>)
- 7 J-H. Jureit, P-Y Duval, "Unit Test Report of the Process Manager for the Atlas DAQ Prototype-1", ATD TN #137, 1999, (see <http://atddoc.cern.ch/Atlas/Notes>)
- 8 P-Y. Duval, R. Jones, D. Schweiger, S. Wheeler, "Test Report of the Run Control for the Atlas DAQ Prototype-1", ATD TN #113, 1999, (see <http://atddoc.cern.ch/Atlas/Notes>)
- 9 D. Burckhart et al., "Back-end Summary Document", ATLAS internal note ATL-DAQ-2000-001