Using the Grid to Test the ATLAS Trigger and Data Acquisition System at Large Scale

Alessandra Forti, Hegoi Garitaonandia, Jiri Masik, Sarah Wheeler, and Thorsten Wengler

***Abstract*—The ATLAS Trigger and Data Acquisition System has been designed to use more than 2000 CPUs. During the current development stage it is crucial to test the system on a number of CPUs of similar scale. A dedicated farm of this size is difficult to find, and can only be made available for short periods. On the other hand many large farms have become available recently as part of computing grids, leading to the idea of using them to test the TDAQ system of ATLAS. However the task of adapting the TDAQ system to run on the Grid is not trivial, as it requires full access to the computing resources it runs on and real-time interaction. Moreover the Grid virtualises the resources to present a common interface to the user. We will describe the implementation and first tests of a scheme that resolves these issues using a pilot job mechanism. The Tier2 cluster in Manchester was successfully used to run a full TDAQ system on 400 nodes using this implementation. Apart from the tests described above, this scheme also has great potential for other applications, like running Grid remote farms to perform detector calibration and monitoring in real-time, and automatic nightly testing of the TDAQ system.**

***Index Terms*—ATLAS, data acquisition, grid computing, nuclear physics, real time systems, scaling, software testing, trigger.**

I. Introduction

ATLAS is a general-purpose high-energy physics experiment for recording proton-proton collisions, nearing completion at the Large Hadron Collider (LHC) at CERN near Geneva in Switzerland. ATLAS has been designed to study the largest possible range of physics at the LHC. Most prominent within this program is the potential to discover many of the missing and hypothesized pieces of the evolving theory of the structure of matter.

ATLAS is designed to handle extremely high data rates, due both to the high frequency of proton-proton collisions per second (40 MHz) at the LHC and the large amount of data produced by the ATLAS detector itself (1–2 Mbytes per collision). However, only a few of these proton collisions or events will contain interesting characteristics that might lead to new discoveries. To reduce this unprecedented amount of data ATLAS uses a specialized multi-level Trigger and Data Acquisition (TDAQ) system to select these events and transfer them to mass storage for later physics analysis at a rate of about 200 per second. Apart from the first selection stage, which uses custom built electronics, the TDAQ system will be implemented as many thousands of software applications running on more than two thousand multi-core CPUs.

As part of the overall design and development program for the TDAQ it has been essential to implement in parallel a program of software verification and testing. Many different types of test are performed such as functionality, performance and stability testing. These take place at every level of the system, for example unit testing of individual software applications, integration tests which verify the compatibility and correct functionality of software applications within a complete sub-system [1], [2] and large-scale testing where the system is tested at a scale approaching the final size of ATLAS. Generally the unit and integration testing has been carried out on small (of the order 50 machines) testbeds dedicated to TDAQ activities located at CERN or collaborating laboratories.

Due to the unprecedented size of the final ATLAS system large scale testing is imperative in order to uncover problems that may not be seen or only have a minimal effect on smaller systems. Initial development and testing of the TDAQ emphasized the critical data flow aspects that had to be demonstrated for this new scale of system. However, as data-taking approaches emphasis has shifted to matching the dataflow performance with an efficient control system to ensure high uptime and fast reconfiguration. Scaling aspects here are only now being fully investigated and understood. For example, the run control software required to synchronise software applications running in TDAQ by its very nature couples the activities of all the separate sub-systems and therefore has to scale correctly.

Unfortunately, large scale testing is the most difficult type of test to carry out due to the lack of suitable testbeds. Since the relation between computing power and price improves with time, ATLAS has postponed the purchase of computing hardware until as near to the start of data taking as possible (taking into account the time needed to install and commission the machines). The consequence is that only about 5–10% of the final system is currently installed and available for software testing. This number of machines is not significantly larger than that in testbeds already available to TDAQ. In the past when large-scale testing has been carried out it has been necessary to find a dedicated cluster of several hundred of machines with exclusive access [3]. This is not an easy task. It usually requires lengthy negotiations with the managers of the cluster to find a mutually acceptable date for the tests, usually several months in advance.

This is not optimal in that it is important to match the large-scale testing to the release date of major TDAQ software releases as part of the release validation. Release schedules

Manuscript received May 3, 2007; revised June 27, 2007.

A. Forti, J. Masik, and T. Wengler are with the University of Manchester, Manchester M13 9PL, U.K.

H. Garitaonandia is with Instituto de Fisica d'Altes Energies, Bellaterra, E-08193 Barcelona, Spain (e-mail: Hegoi.Garitaonandia@cern.ch).

S. Wheeler is with the University of California, Irvine, CA 92664 USA.

Digital Object Identifier 10.1109/TNS.2007.905169

often change on a rather short-term basis and the reserved testing period often does not synchronize well. The time period allocated for using the cluster is typically shorter than expected. This period includes the time needed for preparation studies and to make any necessary adaptions to the cluster and/or TDAQ software, for instance due to the use of a different version of the operating system or reconfiguration of the network for the cluster. Therefore the time left for actual testing can be very limited. As a result only about one large scale test every year has been carried out, whereas major new releases of the TDAQ software have been much more frequent.

On the other hand we have Grid computing, which offers a model for solving massive computational problems by making use of the resources of large numbers of disparate computers, sometimes desktop computers, treated as a virtual cluster embedded in a distributed telecommunications infrastructure [4].

The Grid usually tries to abstract the computing resources, offering a standard interface. However it is also possible to access a finer selection of resources. Moreover, under certain configurations of parts of the Grid, these resources can be assigned exclusively to a small group of people.

Though there are many Grid implementations, ATLAS uses resources related to the EGEE program. Enabling Grids for E-sciencE is a project funded by the European Commission's Sixth Framework Programme that connects more than 180 institutions in 46 countries to construct a multi-science Grid infrastructure for the European Research Area. As ATLAS is one of the users that this infrastructure will give service to with the capability of running standard analysis jobs, it is not difficult to arrange a reconfiguration of certain parts of the Grid to adapt it to the requirements of the TDAQ system.

II. Grid Testing Project Aims

Given the difficulties in the past of finding and reserving in an exclusive fashion large scale testbeds, a new initiative was begun in 2006 to investigate the use of the UK-based Manchester Tier2 Grid cluster as a possible candidate for Grid-based testing of the ATLAS TDAQ software. The aims of the project are to provide a medium to large-scale testbed facility of up to 900 nodes that would be available at rather short notice (of the order of about one week). Such flexibility meets the TDAQ requirements for large-scale validation of releases since software release dates are often delayed at short notice. The uses foreseen include large-scale verification of software releases, large-scale testing of various sub-components of the TDAQ system and multi-node overnight testing of software currently under development ("nightlies"). Clearly the network architecture of the Manchester Tier2 Grid cluster has not been optimized for TDAQ data collection, and performance tests of the data collection system cannot be performed. However, this should not be a problem for other types of testing especially given the fact that the cluster consists of recent high performance CPUs connected with a high-performance network. More details of the cluster are given in Section V.

III. The Technical Problem

The task of porting the TDAQ system to run on the Grid is complicated by the fact that the facilities provided by the current EGEE implementation for running parallel applications are not enough to satisfy in a direct way all the realtime requirements of the TDAQ system.

A. TDAQ Application

The TDAQ system consists of applications running in parallel on a dedicated farms of linux nodes. It can be described as one head node and many worker nodes. In normal operation the head node checks if the *process manager* applications (or PMGs) are running on the worker nodes. If they are not these applications are started via SSH. Using its proprietary communication protocols the head node then specifies to the different PMGs which processes should be started on that node. Thus processes are created on the fly [5]. Another difficult issue for running on the Grid is that the local log files on each machine are created with names that depend on real time parameters, such as time and process identifier.

In non debugging mode, the TDAQ system needs to know the names of the nodes it will run on. It also needs network transparency on a range of ports between its nodes. There are also requirements on the network performance, its bandwidth and latency. On a dedicated farm login via SSH is available for developers and testers, to check log files stored locally and investigate problematic applications.

B. Grid Platform

The Grid platform comes with its own requirements and environment. It supports natively several types of jobs. In order to submit a job to the Grid it is necessary to describe it in JDL (Job Description Language). The JDL file associated with a job specifies the type of job, files that are needed to run, outputs, inputs, as well as some services and QoS requirements, like the need of an specific version of some analysis software, CPU speed, ping response time, etcetera. The EGEE implementation of the Grid has a very interesting feature for our task: it allows the user to select manually the cluster where he pretends to submit the jobs, though not specific destination machines.

C. Main Technical Issues to Solve

- The TDAQ system needs to know the names of the nodes it will run on. On the other when jobs are submitted to the Grid it is not known a priori on which machine they will arrive.
- Interactive access: The TDAQ system usually relies on SSH to perform some tasks such as starting the *process managers*, cleaning up files and processes, and collecting logs. There is no such access for Grid nodes. Even if access was possible, one cannot submit a cleanup job for two reasons: it will possibly not arrive at the machine with the problem, and, since the Grid maps a Grid user to an arbitrary local linux user it might not be mapped to the same local linux user.

- The TDAQ system and the Grid may experience their own set of problems like hanging processes and lost jobs, that have to be dealt with.

D. Discarded Alternatives

While looking for a solution several alternatives were discarded:

MPI — Message Passing Interface (MPI) is a library for programming parallel applications. The Grid natively supports MPI jobs. However the TDAQ system has its own protocols for communication, and its own way of starting child processes. This option would imply very intrusive changes to the TDAQ system, that would require a big developement effort. As the objective of this project is to test the TDAQ software, the introduction of modifications in the TDAQ should be avoided. Moreover one of the items to be tested is the TDAQ control system and its replacement would be an error. Hence this option has been discarded.

DAG — Dyrect Acyclic Graph (DAQ) is a facility provided by EGEE to define a job as a combination of sequential and parallel dependent processes. Again the TDAQ is programmed with a very different philosophy, and could not be adapted easily.

IJ — The EGEE middleware supports Interactive Jobs (IJ), which would solve the problem, however, they are considered unstable and represent a heavy load for the machine running the user interface. This solution was therefore also not feasible.

PMG — A more TDAQ friendly option would be to start the PMG agents as plain UNIX services on every machine, without having any user requesting it via the Grid infrastructure. But this alternative has a major drawback. As any user with the usual TDAQ tools could potentially connect to the PMG agents started as services (and not to the ones started by him only) all Grid accounting would be bypassed. This method is therefore not suitable for running the TDAQ system in Grid mode.

As there is not a straightforward solution for running the TDAQ on the Grid without introducing major changes in any of them, we opted to develop a pilot job mechanism that will be described in the next section.

IV. The GridFarm Package

A new software package was developed that is now part of the official ATLAS TDAQ software release (since verion 01-08-00). This package contains the tools that enable the TDAQ system to operate in a Grid environment.

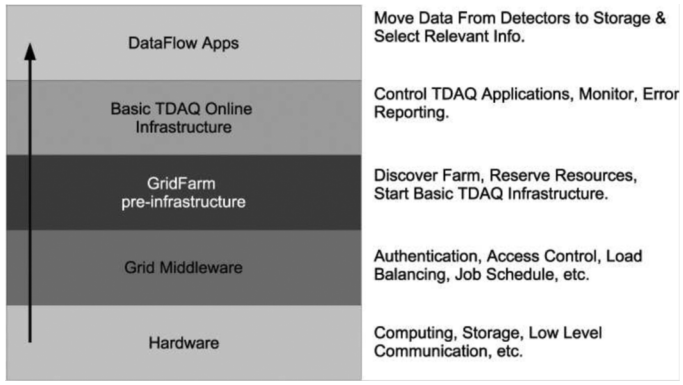

Fig. 1. Layer diagram.

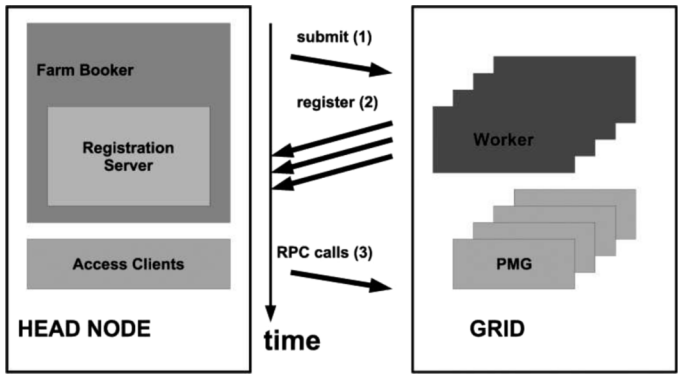

Fig. 2. Architecture.

A. Architecture

The elements needed to operate a GridFarm run are:

UI — The User Interface (UI) is a machine with the EGEE software installed that allows a user with a valid Grid certificate to login and submit jobs to the Grid.

CE — The computing element (CE) is a cluster of nodes in the Grid. As the TDAQ system has high real time requirements, the runs will be restricted to a single CE, therefore relying on the internal network connections of the CE, which are usually faster than those between CEs.

FB — The Farm Booker (FB) is responsible for submitting reservation jobs from the user interface, retransmitting upon error, etc.

WW — The Worker Wrapper (WW) is the job that the user submits to the Grid. Its main functions are to register the node in the Registration Server and to direct the TDAQ run to the appropriate nodes.

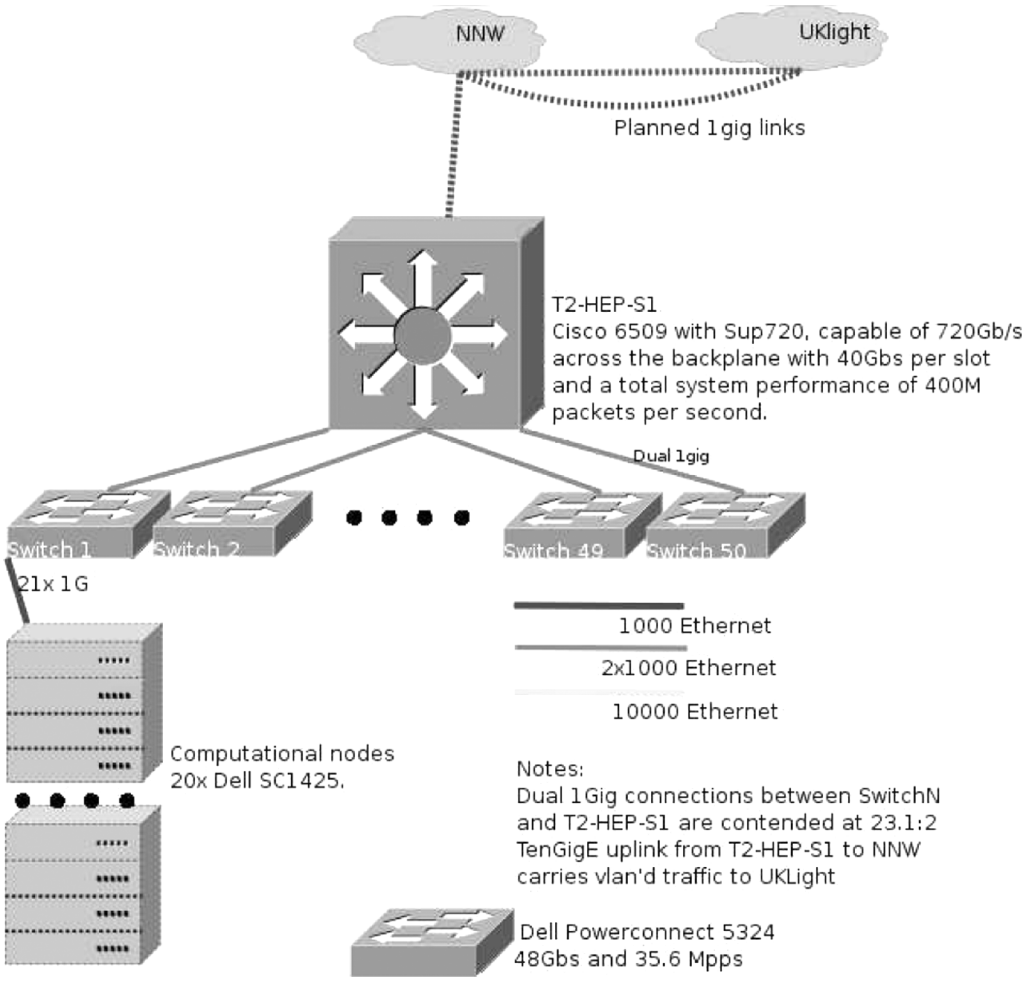

Fig. 3. Network Topology of the Manchester Tier-2 Grid Cluster.

RS The Registration Server (RS) is an element that will run in the user interface machine during the first stage. It is where the worker wrappers publish the names of the nodes they are running in.

AC The Access Clients (AC) are light-weight RPC (Remote Procedure Call) clients that will connect to the worker wrappers to gather, monitor, start or stop a TDAQ element. They will be the link of the user with the cluster for running the TDAQ.

The components described above provide the GridFarm pre-infrastructure layer (see Fig. 1) that accounts for the functions listed on its right: farm discovery, reservation of resources, etcetera. This layer interfaces directly two systems: the Grid middleware, that abstracts resources, provides global authentication, etcetera; and it also interfaces the basic TDAQ online infrastructure, composed of the PMG processes.

The order of creation of these layers is from bottom to top as will be detailed in the next section.

B. Start, Run, Monitor, Stop

After the user has registered to the Grid in a User Interface, several pilot jobs are submitted to a single Computing Element where the TDAQ software is installed (first step in Fig 2). At the same time the Registration Server is started in the User Interface. Once the pilot jobs are assigned to a node in the Computing Element, their first task is to register in the Registration Server the name of the node they have been assigned to (second step in Fig. 2). This way the user working on the User Interface gets a full list of nodes that it controls.

After the pilot jobs have registered themselves, they start an RPC server that answers to a small set of functions:

START_PMG_AGENTS
LIST_RUNNING_PROCESSES
LIST_FILES_IN_LOG_AREA
SHOW_A_LOGFILE
GET_ALL_LOGS
END_WRAPPER
CLEANUP_LOGS
CLEANUP_TDAQ_PROCESSES

The user can now generate the configuration for the TDAQ run, taking as input the list of nodes that was created during the reservation. The user then connects via a light-weigh client to the RPC servers in the reserved nodes to start the PMG agents.[1] Finally the user initiates a TDAQ session that will connect to

[1]As these PMG agents are started via the Grid infrastructure they do not bypass accounting.

these PMG agents and run the TDAQ transparently, as if running on a dedicated cluster.

The usual TDAQ monitoring tools are available to the user through the usual TDAQ monitoring panels. However, if an action is to be performed that would imply an SSH connection to the worker nodes in normal operation, it must be substituted with an Access Client, a script that connects to the Worker Wrapper to perform one of the tasks mentioned above.

C. Additional Issues

There is nothing in the system a priori that prevents several reservation jobs to arrive on the same nodes. To some extend this issue is resolved by introducing locks in the Worker Wrapper, that prevent more the one to run on the same node. There is, however, another issue related to this. Since a job arriving on a node that is already occupied is canceled, the number of registered machines may be smaller than the number of submitted jobs (even if no job is lost). The code must take this possibility into account. The Farm Booker manages replicated reservations, resubmission of jobs, and clearing of surplus reservations.

There may still be a problem in that the reservation may not be as large as the user would like it to be. There are two possible strategies to avoid small reservations. One could configure the Computing Element so that only one job arrives to each node. Though this alternative is the most effective, it is quite invasive on the Computing Element.

The other strategy is to make the Worker Wrapper run a very CPU consuming thread at startup and during a fixed time, so that the Grid sees these nodes as very loaded, and assigns different nodes to the next jobs. This strategy used in combination with submission of extra jobs, re-submission and handling of double reservations has proved to be efficient. However, due to its not 100% predictable output the option has been relegated to debugging purposes of the tool itself.

V. Large Scale Tests in Manchester Tier2

With the GridFarm package presented in the previous section, it was possible to run the TDAQ system on the Manchester Tier2 Grid cluster.

This cluster consists of 1000 nodes, of which 100 are reserved for Grid engineering purposes, leaving a maximum of 900 nodes in principle. Currently the cluster is split in 2 separate clusters. Each of them has Gbps interfaces connected to 1 Gbps rack switches (see Fig. 3); each rack switch groups 20 machines and is connected to the top level 10 Gbps CISCO switch. All nodes have a 2.8 GHz dual cpu Intel, with 4 GB of memory, and 500 GB of disk space. The operating system installed is currently Scientific Linux CERN 3.0.4.

The TDAQ tests of increasing complexity have been carried out on a number of hosts ranging from 10 to 200. The hosts were assigned different tasks representing all components of the Atlas TDAQ system. Read-out system applications were preloaded with Monte-Carlo simulated signals and provided input to the rest of the dataflow system for further processing. Most of the hosts were performing event processing and selection in one of the two levels of the High-Level trigger system. In the largest setup there were 15 hosts in the second-level trigger and 150 hosts in the last trigger level (called Event filter), with a total of 500 applications (Fig. 4 presents part of the TDAQ control tree, which gives a feel of the size of the system). The system operated with a rate of 80 Hz (Fig. 5).

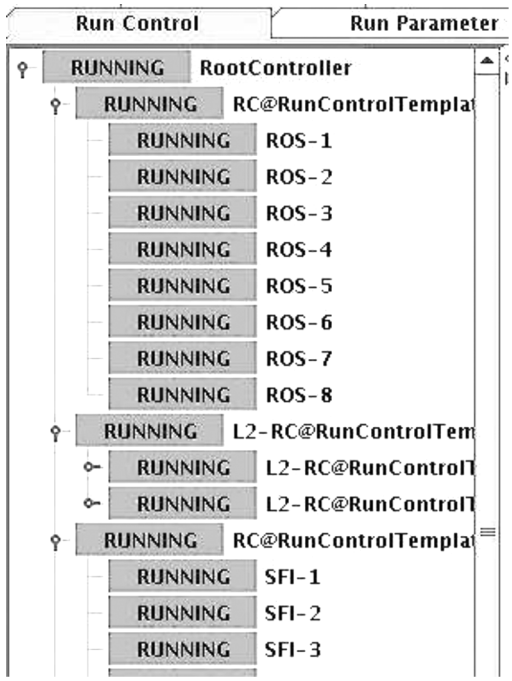

Fig. 4. Part of a Control Tree of a Run with 200 Nodes.

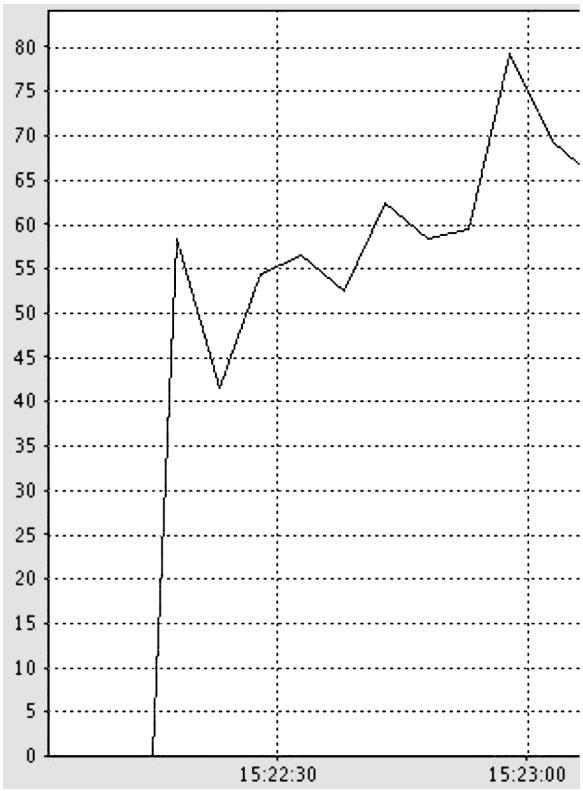

Fig. 5. Event Building Rate (Hz versus time) for a 200 Node Run.

VI. Conclusion

A number of proof of principle tests have been accomplished establishing that the ATLAS TDAQ system may be run on the Manchester Tier2 Grid farm. Not only do these tests show that the TDAQ software can be ported to a Grid platform, but they also establish that the TDAQ system can be run interactively in the same way as on dedicated testbeds at CERN with a minimum of changes. To date it has been possible to run successfully a complete TDAQ system comprising 180 worker nodes. This is the largest configuration that has been been run so far using the most recent TDAQ software releases that are currently installed on the ATLAS prototype (pre-series) at CERN.

We aim to keep the software installation on the Manchester Tier2 Grid synchronized with that on the ATLAS pre-series and (ultimately) ATLAS itself. Use cases will include weekly testing of multi-node configurations and testing of specific components, such as trigger algorithms and database access, at a large scale.

A major issue to be addressed for any system of this type is end user documentation, sufficient to operate the tests without the presence of experts. Ease of use is a main ingredient for the success of the project, which should in the end be measured by the real-world use that is made of the facility. The documentation provided focuses on the operational aspects of the system, including the prerequisites for operating the pilot scheme in a Grid environment, while technical details of implementation are documented separately. Another important aspect is to ensure the evolution of the tools in sync with the system it serves to test. Significant developments are still expected over the next months and years, and a structured development process is needed to keep the Grid tools in line with the developments of the TDAQ system. To this end an official software package as part of the TDAQ release has been created, which will be maintained as part of all future TDAQ software releases.

Future plans include the implementation of the automatic nightly software testing on a multi-node basis and the possible use of the GridFarm package as a bridge to using other Grids for remote farming, to perform calibration and monitoring opertions during ATLAS running [6].

References

[1] I. Riu *et al.*, "Integration of the trigger and data acquisition systems in ATLAS," presented at the IEEE NPSS 15th Real Time Conf., Batavia, IL, 2007.

[2] N. G. Unel *et al.*, "Studies with the ATLAS trigger and data acquisition pre-series setup," presented at the Computing in High Energy and Nuclear Physics Conf., Mumbai, India, Feb. 2006.

[3] D. Burckhart *et al.*, "Testing on a large scale: Running the atlas data acquisition and high level trigger software on 700 pc nodes," presented at the Computing in High Energy and Nuclear Physics Conf., Mumbai, India, Feb. 2006.

[4] R. Buyya, Grid Computing: Making the Global Cyberinfrastructure for eScience a Reality, Jul. 2005, CSI Comunications.

[5] G. Avolio, M. Dobson, G. Lehmann-Miotto, and M. Wiesmann, "The process manager in the ATLAS DAQ system," presented at the IEEE NPSS 15th Real Time Conf., Batavia, IL, 2007.

[6] J. Pieczykolan, L. Dutka, B. Kryza, K. Korcyl, and J. Kitowsk, "Data dispatcher for real time applications in grid environment," presented at the 7th Int. Conf. Computational Science., Beijing,, China, May 2007.