Abstract
This document provides a detailed report on the HLT/DAQ Large Scale Tests which were performed in March 2004 on the CERN IT LXSHARE clusters. HLT/DAQ System Software tests as well as extended component tests were run. The final aim was to run a large scale fully integrated system. Due to instabilities in the control aspects of a large scale integrated system, this aim was not achieved. The Online Software sub-system tests show that a large scale system in principle is capable of running stably. The problems leading to these instabilities lay in the interplay between Online Software and HLT components and reveal themselves only at large scales. Results are presented, compared to previous tests where applicable and a list of issues has been identified leading to improvements to be implemented before the next large scale integration are attempted.

Testers: the community of testers and developers in HLT/DAQ

Editors of the test report:

University of Alberta and University of California at Irvine: S. Wheeler

University of Bern: H.P. Beck

1 Introduction

This document presents and discusses the Large Scale Tests of the HLT/DAQ (High Level Triggers/Data Acquisition System) Software performed in March 2004. For the Online Software a first series of large scale system integration tests were performed throughout the year 2001 and results are presented in [1]. A second series has been performed in the year 2002 on a cluster of 230 nodes and reported in [2], and a third series in 2003 [3]. The Online Software tests verified the functionality on a system whose size is closer to the final Atlas TDAQ system size than the one presently used for general studies and for test beam operation. The inter-operability of the Online Software system components, the scalability of control and communication functionality and the performance of the operation of the integrated system could be studied.

Large scale functional tests were performed in the year 2002 to test scaling aspects of the DataCollection software [4]. The scaling behaviour of the DataCollection framework and its applications were studied with the primary aim of gaining experience in the configuration and running of larger testbeds using the facilities provided by the online system. Due to the fact that the performance of data movement was limited mainly by the network topology available on the LXSHARE testbed, performance tests of the DataCollection software were not foreseen for these tests and neither are they foreseen for similar tests in the future. In addition there are routers in between the available test nodes, thus limiting the possibility of multicasts used for the communication between the DFM (Data Flow Manager) and the ROSs (Read-Out Subsystems).
The first large scale tests of the Event Filter (EF) software were carried out in July 2001[5] using the Marseille prototype version of the dataflow and supervision software and these tests were repeated in November 2001 [6]. Following the prototype studies, a major review of the EF dataflow and supervision requirements was carried out in 2001 as preparation for the design and implementation of the EF software for the first ATLAS run. Initial large scale tests with the new version of the software were carried out in January 2003 [7] on a cluster of 230 nodes. One major difference between the Marseille prototype and the new software is that the supervision system is now implemented using components of the Online Software rather than with Java mobile agent technology. Development of the software has continued and it was hoped that the large scale tests of March 2004 would first verify the latest version of the software by repeating and reproducing the results of the tests made in January 2003 and subsequently extend them to a larger number of machines (~350). Following the individual sub-system tests, the ultimate goal was to verify the functionality of a very large integrated system.

1.1 Purpose of the document

The document represents the detailed tests report of the March 2004 Atlas HLT/DAQ Software Scalability tests. The results and the feedback presented here will be used as input for decisions to be made on the future development of the HLT/DAQ and on aims and layout of future tests. Reference to the location of testware and test result data is given.

1.2 Glossary, acronyms and abbreviations

The terms and abbreviations used in this document are the ones currently used by TDAQ and readers should refer to the ATLAS TDAQ glossary [8].

1.3 References

1. I. Alexandrov et. al., Large Scale and Performance Tests of the Atlas Online Software 9-008, CHEP2001 Beijing
5. C. Bee et al., ATLAS Event Filter Tests on the ASGARD cluster at ETH Zurich, September 2001, ATL-DAQ-2001-007
2 Large Scale Tests

The March 2004 large scale tests aimed to test the functionality and scalability of the integrated HLT/DAQ System and operational performance aspects (i.e. duration of state transition commands) of selected components and sub-systems.

2.1 Scope and Aims

The objective of the large scale tests was the verification of the operational scalability and the study of the limits of the HLT/DAQ system with a configuration containing a large number of nodes. The functional integration of the Event Filter, Level-2 trigger system (LVL2), DataFlow and the Online
Software System on a large scale was the ultimate goal. Integrated tests were foreseen to study the effects on times to start and stop all applications and to perform state transitions as a function of the partition composition (e.g. altering the number and size of the HLT sub-farms in the partition). It was also intended to study other performance aspects of the system such as communications, interaction with databases and operational monitoring.

In preparation for the integrated test, it was planned that each sub-system would verify its functionality on a large scale with a series of their own standalone large scale tests. A major aim of the work was that the standalone tests and preparations for the integrated test should complement activities in the integrated test beam and work on small test beds.

For the online system, it was hoped that the tests would confirm the large scale functionality while enlarging the scope. The aim was to study the interaction between the components, to identify critical areas and to investigate the variation and optimization of online system parameters. The timing values of the steps which lead through the various data acquisition phases were recorded and analysed. The tests performed in 2004 aimed to verify that the new implementations and enhancements of the Communication software, the Run Control software and the remote database server. More specific tests studying the scalability of selected components were also performed.

The High Level Triggers (HLT) tests were split into separate LVL2 and EF standalone tests. The aim for the LVL2 sub-system was to investigate a system of a size as close as possible to final ATLAS (in this case approaching ~30%), tests were also carried out with a system consisting of only LVL2 and ROS and ROS emulator subfarms. Since the LVL2 selection algorithms were not sufficiently stable for large scale deployment, only the control and functional aspects of the RoI (Region of Interest) Collection were tested using a “dummy” LVL2 algorithm. The latter could request dummy RoI data; a CPU burner mimicked the algorithm’s execution time. The aims included: test scalability of LVL2 farm control; investigate many small farms vs. few large farms and multiple L2PU (Level-2 Processing Unit) applications per LVL2 node vs. a single multi-threaded application.

The original aim for the EF standalone tests was to repeat the scalability tests made in January 2003 using the latest implementation of the software. Specifically the tests were intended to investigate state transition times as a function of sub-farm size, number of nodes per sub-farm and number of Event Filter applications. The plan was to ramp the tests up to a similar number of nodes as the 2003 tests (230) and then further, up to the maximum number of nodes available (~330). Due to inexperience in the installation of the Offline software on a large cluster and in its use in the online environment it was decided that no attempt would be made to run offline algorithms in the Event Filter either in the standalone or integrated tests. Therefore the tests focused on the scalability aspects of the supervision and dataflow software only.

It was planned that the HLT/DAQ integration tests should follow a series of steps to allow for the testing of sub-system and component groups before combining them in the complete system. Details of these steps are set out below:

1. Step: Online Software System
   - Test the integrated functionality for startup/shutdown and state transitions
   - Verify that the performance is in the expected range
2. Step:
   a. **ONLSW + LVL2 ROI Collection (L2SVs + L2PUs + ROSs)**
      - Test scalability aspect of High Level trigger LVL2 part of the dataflow without algorithms. Components involved: as in step 1 + ROI Collection + O(200 ROSs, 200 LPUs, 10 L2SVs)
      - Study of control aspects, robustness and stability on a large scale
      - Take measurements for state transitions for varying number of sub-farms and number of nodes per sub-farm as a test for farm control
      - Inter-operability of components, use of IS and MRS while ROSes use dummy data and UDP (TCP as fall-back solution)
      - Functional aspects of data-movements in a large scale system, to be studied relative to the available network topology
   b. **ONLSW + LVL2 ROI Collection + Event Builder**
      - Components involved: as in step 2 a) + Event Builder (DFM, SFIs, ROSs)

3. **Step 3: ONLSW + EF**
   - Verify robustness of the EF control and operational monitoring
   - Verify that the performance is in the expected range
   - Study load balancing on PTs
   - Study EF communication for SFI, EFD, SFO
   - Take measurements for state transitions for varying number of sub-farms and number of nodes per sub-farm as a test for farm control

4. **Step 4: ONLSW + L2 ROI Collection + Event Builder + EF (step 2+ step 3 together) see Figure 1.**
   - Test integrated functionality
   - Take Measurements for state transitions for increasing number of elements
   - Verify robustness with the help of operational monitoring while in running state

Note that the so-called standalone tests of both the LVL2 and EF software are in fact already binary tests of the LVL2 and EF software respectively with the Online Software. Although some of the control software has been customized specifically for use in the HLT (e.g. the Local Controller implementation of the Run Control) the HLT supervision system as a whole relies heavily on the Online Software infrastructure for process control, inter-process communication, monitoring, etc.

Component tests for the Online Software Information Service were planned to be performed at the beginning of the tests in order to use the resulting information in the optimization of configuration parameters for the subsequent integration tests. Tests for the Configuration Databases, Setup, and for Web services were planned to be performed at convenient times, independently of the integration steps described.
### 2.2 The Testbed

The testbed was similar to the one used for the June 2002 and 2003 tests. Up to 330 PCs of the CERN IT LXSHARE testbed [9] were used. All nodes were time synchronized via ntp[10]. They were equipped with 600–800 MHz up to 2.4 GHz Dual Pentium III processors, 256–512 MBytes of memory running the Linux RedHat 7.3 operating system with selected system parameters adjusted to the needs of the tests. Several series of tests were conducted. As there was no dedicated common file system such as NFS available for the PC’s, the Software was replicated on the local disk of each PC. Afs was used to help installing the software. A few selected processes were run also from afs. The PCs were connected in groups of 22 via Fast Ethernet to local Fast Ethernet switches. Those were connected via Gigabit Ethernet uplinks to Gigabit Ethernet switches themselves being connected to the CERN network switching routers. The exact topology of the system is given in Figure 2 and Figure 3.

![Diagram of the testbed configuration](image-url)
Figure 2 Topology of the March 2004 testbed; week 1 and 2

Figure 3 Topology of the March 2004 testbed, week 3 and 4
2.3 Hardware

During the first two weeks of testing, only the less powerful PCs were available as shown in Figure 2. During the third and forth week, the set of 2.4 GHZ PC’s each connected via Gigabit Ethernet to Gigabit Routers were available in addition, as shown in Figure 3.

- ~220 dual P-III 600-800Mhz on shared Gig-Ethernet (weeks one to four)
- ~100 dual Xenon 2.4GHz on private Gig-Ethernet (weeks three and four)

2.4 Software

- CERN Linux RH7.3
- Software releases: Online Software 21-01 and 21-02 Dataflow 00-07-00 and 00-08-00
- ROS sub-farms with ROS Emulators (real ROS does not support SMP yet).
- LVL2 sub-farms each with multiple L2PUs and one L2SVs

2.5 Test Organisation and Approach

A core group of HLT/DAQ developers and testers established the planning for the tests which was presented and discussed in the HLT/DAQ community [11]. A working web page [12] was established containing organisational information, details on the testbed and its layout, testbed testing time scheduling, meetings, and the entry to the online test log book, which had been introduced for the tests. During the tests, more testers/developers joined the testing team for certain times. Information was also communicated via the web page, the log book and a dedicated mailing list. During the tests, regular short meetings were held every 2nd day to inform the test participants on the latest status, prepare the testing schedule for the next days and discuss problems where necessary. In general, the test cluster was used during the day for system integration, debugging and system tuning while automated tests were run over night. During certain times, mostly during the day, the PC farm was shared among the testers. Timing measurements were run in exclusive mode, then the use of the PC farm was dedicated to a particular test only.

During the first part of the tests, component and sub-system testing was performed by sub-system testers and by the component developers. The EF and the LVL2 system use both the Online Software system and the dataflow system. Therefore the data acquisition transitions were the same for all types of integration tests.

A number of tests dedicated to study scalability and performance aspects of other software components were also performed. Although separate from the main integration activities described in section
2.1 The availability of a large cluster made this a convenient time to carry out the tests. These tests were:

- Specific tests dedicated to the Information Service Component (IS)
- Specific tests dedicated to the Configuration Database Component (CONFDB).
- Specific tests dedicated to the study of the Setup Component (Setup)
- Specific tests dedicated to the study of the Integrates Graphical user Interface (IGUI)
- Specific tests dedicated to the study of Web services

A detailed description of the tests is presented together with the test results, see chapters 4 and 5.

3 HLT Scalability Test

3.1 Aims

The original aim was to extend the scalability tests made in January 2003 of the Event Filter farm to the entire HLT, i.e. to include the LVL2 trigger and most of the DAQ components. In order to test each particular subsystem with the largest number of nodes available, it was envisaged to run tests with DAQ components alone (scalability of the event building), LVL2 alone (with ROS emulators but no EB (Event Builder)), the EF alone (with an emulated EB but no LVL2) and eventually the full system. Due to problems with large configurations, scalability tests of the EB alone as well as the full HLT system were abandoned.

The testing period in March 2004 was the first occasion on which the LocalController implementation of the Run Control and the associated applications in the HLT and dataflow applications had been used on a large scale. The LocalController is based on the customizable Run Control Skeleton provided by the Online Software group. It was originally developed for controlling small numbers of applications in the ROS sub-system. One major difference between the LocalController and other sub-system implementations of the leaf controller (i.e. controller that interacts directly with sub-system applications) is that it implements its own process management rather than relying on the Online Software DSA supervisor application. In autumn 2003, small scale feasibility studies were carried out to investigate whether the LocalController could be adapted for use system-wide in order to provide a unified interface between the Online Software Run Control hierarchy and the controlled applications. The feasibility studies were successful requiring at the time, only a few modifications to the software and the LocalController was subsequently adopted system-wide. The March 2004 tests were the first time this interface was tested on a large scale, in particular in the trigger farms where configurations using a single LocalController for controlling each sub-farm containing ~100s applications were studied.

1. It should be noted that the required granularity of the Run Control hierarchy in the HLT for the final ATLAS system is still an open question and must be investigated further.
3.1.1 Specific aims for the EF

The aim was to repeat the Event Filter farm control scalability tests made in January 2003 [7], using the latest implementation of the software.

The major changes since the previous tests were:

a. a new implementation of the Event Filter sub-farm Run Controller by the LocalController
b. implementation of the Run Control interface by the Processing Tasks (PTs)
c. use of the configuration database by the Event Filter Dataflow application (EFD)

In January 2003 only the EFDs responded to Run Control commands, the PTs did not implement a Run Control interface and ignored Run Control commands. A large part of the tests focused simply on the process management aspects of the DSA supervisor for starting and stopping the processes in different sub-farm configurations. By March 2004 the PT Run Control interface had been implemented implying that for a sub-farm consisting of the same number of nodes, the sub-farm Controller now had to control five times as many applications compared with the January 2003 tests (i.e. 1 EFD and 4 PTs per node as opposed to 1 EFD per node).

The March 2004 tests were intended to investigate Run Control transition times as a function of sub-farm size, number of nodes per sub-farm and number of Event Filter applications. The plan was to ramp the tests up to a similar number of nodes as the 2003 tests (~200) and then further, up to the maximum number of nodes available (~330)

It was planned that on successful completion of the stand-alone EF scalability tests the EF would be included as part of a large-scale test of a full TDAQ system (ROS/LVL2/EB/EF).

3.1.2 Specific aims for LVL2

In order to test the scalability of the LVL2 subsystem of a size as close as possible to final ATLAS (in this case approaching ~30%), tests were also carried out with a system consisting of only LVL2 and ROS(E) subfarms. Since the LVL2 selection algorithms were not sufficiently stable for large scale deployment, only the control and dataflow aspects were tested using a ’dummy’ LVL2 algorithm. The latter could request dummy RoI data; a CPU burner mimicked the algorithm’s execution time. The aims included:

1. test scalability of LVL2 farm control. Issues:
   • Many small farms vs. few large farms.
   • Multiple L2PU applications per LVL2 node vs. a single multi-threaded application.
   • Investigate transition times for Run Control state changes (boot/configure/start/stop/unload/shutdown) as function farms size, number of nodes and number of applications.

2. Verify functionality of DataCollection on a large testbed
   • Configure and run systems with both UDP or TCP connections
   • Verify that event data flows correctly.
   • Verify load-balancing.
• Verify operational monitoring.

Repeat the same tests with a smaller LVL2 subsystem but integrated with Event Builder and EventFilter.

3.2 Event Filter Configuration

Stand-alone configurations were generated with an Sub-Farm Input/Sub-Farm Output emulator pair and one LocalController per sub-farm, 1 EFD and typically 4 PTs per sub-farm node. The EF configuration files were created with database generation scripts. It was planned to test the difference in behaviour between systems consisting of a large number of small sub-farms versus systems with a small number of large sub-farms. It is assumed that ATLAS will have EF sub-farms consisting of approximately 30 nodes. The software used to run the configurations was taken from Online Software release 21-02 and Dataflow release 00-08-01.

3.3 Results for the EF

It was not possible to run a very large scale stand-alone Event Filter configuration. The largest configurations that were run with any success were:

a. 4 sub-farms, 25 nodes per sub-farm, 1 EFD, 4 PTs per sub-farm node
b. 6 sub-farms, 25 nodes per sub-farm, 1 EFD, 4 PTs per sub-farm node

It was possible to get configuration (a) to the Running state but it failed at the Unload transition. Configuration (b) could be brought to the Running state but failed at Stop. The system was far too unstable to make measurements.

It was found that the probability to successfully cycle through the Run Control FSM dropped exponentially with increasing system size. Systems greater than approximately 100 nodes essentially did not work, this conclusion is also supported by the standalone LVL2 tests reported in section 3.4. Before moving to LXSHARE a number of pre-tests were made on the MAGNI cluster. It had been assumed that the major scaling problem would be caused by the total number of applications running per sub-farm, that is under the control of a single LocalController. Therefore, these pre-tests focused exclusively on scalability issues in a single sub-farm. A number of problems were discovered and solved. It was possible to run a single sub-farm successfully consisting of ~20 nodes and ~300 applications. After this success it was assumed that it would be trivial to scale up the system rapidly on LXSHARE by increasing the number of sub-farms. This was not the case. Not only were problems observed as the number of nodes and applications per sub-farm increased but also as the number of sub-farms increased. Two possible causes of the scaling problems as a function of number of sub-farms have been identified:

• After the tests a bug was found in the Online Software PMG package which the LocalController uses for process management. This has now been fixed.
• The version of the LocalController used in the tests lacked fault tolerance. Since a sub-farm LocalController would fail its transition if only one application within the sub-farm failed its transition, the likelihood that a particular transition will fail will depend strongly on the total number of sub-farms. The fault tolerance of the LocalController has subsequently been improved although it is only a first attempt and is still rudimentary.

Other possible sources of problems cannot currently be ruled out since the software has not yet been tested again on a large scale since the above problems were identified and fixed.

No attempt was made to run a combined large-scale test due to the problems observed in the stand-alone EF test.

### 3.4 Results for LVL2

For various reasons, not always completely analysed and understood, a large fraction of the runs failed. Figure 4 is an example showing the fraction of successful runs as a function of the system size: an increasing number of L2PU nodes plus a fixed number of 64 ROSs. It can clearly be seen that the success rate drops exponentially with the increasing number of L2PU nodes, irrespective of the number of L2PU applications running on each node (1 or 2 in the example shown).

![Figure 4: Fraction of successful runs for all UDP test with 64 ROSs.](image)

Figure 5 shows transition times for Run Control state changes with respect to the number of applications while Figure 6 is with respect to the number of hosts in the system.

All transition times are acceptable but we expected a success rate close to (or equal to) one.

A problem was found in the message passing code used by LVL2 which may have been responsible for some of the scaling problems seen. This has subsequently been corrected however, it has not yet been
tested again at large scale therefore it is not possible to say at this stage whether this will have a beneficial effect on the scaling behaviour.

Figure 5 Run Control transition times with respect to the number of applications.

Figure 6 Run Control transition times with respect to the number of hosts.
3.5 HLT Conclusions

The large scale tests provided a major impetus to HLT developers to get software ready for use in the 2004 integrated testbeam. In particular:

- this was the first big test of the unified Run Control based on the LocalController and has resulted in improved scalability of this component; work following the large scale tests has also improved its fault-tolerance (restarting or ignoring where appropriate, failed applications)
- the robustness (thread-safeness) of the EFD application of the EF dataflow was improved significantly
- the EF dataflow software was included for the first time in the standard dataflow release
- the LVL2 ROI collection software has been verified
- a comparison of the use of multiple applications versus multithreading has been made in LVL2
- preliminary tests of the Gatherer application for operational monitoring were successfully carried out
- improved generation scripts for the population of OKS databases for both LVL2 and the EF have been produced

The standalone HLT tests confirmed that small (less than ~100 nodes) configurations work, establishing that the current implementation of the software could be used successfully at the 2004 testbeam.

It is clear from the results presented in the previous sections that larger configurations work rather inconsistently and any further attempt at system integration would have been unproductive. We believe the main reasons why the large scale integration failed are the following:

- the time between the software releases designated for the tests and the scheduled time for the start of tests on LXSHARE was far to short, releases were only ready at the time the tests were scheduled to start
- the correlation of actual release date (as opposed to planned release date), available manpower (testbeam) and availability of the test cluster did not allow sufficient time/effort for the testing of the combined HLT/DAQ system on a medium scale prior to the tests, in particular for the EF, major software debugging was still going on (on the MAGNI cluster) even after the official test period had started
- insufficient pre-testing of the LocalController for use in large trigger farms; standalone large scale tests of the Online Software Run Control system were successful which implies some of the instabilities observed are due to problems with unforeseen interplay between the Online and HLT software in this interface
- adaptation of the database configuration scripts and scripts to automate measurements took longer than expected
- once tests were moved to LXSHARE we lacked (or were unaware of) the necessary analysis and diagnostic tools to understand why the tests were failing
- we did not have the necessary tools to deal efficiently with the lack of a shared file system on LXSHARE
• the fault tolerance of the system was very poor meaning that if in one process out of many failed a transition, the whole system would fail and become unusable; note that subsequent modifications to the LocalController have improved its fault tolerance in the testbeam (see above).

A general observation made was that when instabilities started to occur a general degradation was usually seen at each subsequent attempt to boot and run the system. It was very difficult to understand the causes of the problem on such a big system without the use of appropriate visualization tools. The only way to improve the situation was to kill all the processes on all the nodes and start over again. This was very inefficient again due to the lack of proper tools and wasted a great deal of the allocated testing time. We could only speculate that the degradation in performance might be due to processes from previous runs not being cleaned up correctly, or resources not getting released properly. Although the reasons were not fully understood, in some cases it could be determined that the degradation was due to hanging processes.

4 Online Software System integrated tests and their results

The aim of the online system tests was to confirm its performance and scalability after a number of internal modification with the enhanced system 14 months after the last large scale tests. Major software changes were the move from ILU to the CORBA implementation OMNIOrb and the new Run Control and Supervision implementation based on CLIPS. For the Online Software a first series of large scale system integration tests had been performed throughout the year 2001 and results are presented in [1]. A second series has been performed in the year 2002 on a cluster of 230 nodes and reported in [2], and a third series in 2003 [3].

4.1 Test Approach for the Online Software System

For the Online System tests, the tests approach as explained here is similar to the one for the June 2002 and the January 2003 tests. The tests were an iteration of the Large Scale and Performance tests which had been performed in 2001 and in June 2002. Therefore the same test plan [16] could serve as a basis. Updates to the test plan for the June 2004 tests were described on the web page [15]. The described items included the aims and scope, the tentative schedule and responsibilities and information on the usage of the testbed. Specific meetings were held for the planning of the tests with the testers and the colleagues from CERN IT who were efficiently supporting the test bed.

4.2 Testware of the Online System tests

Existing test and example programs which are part of the standard Online Software Release for the involved Components were used for the integrated tests. Specific additions were made for specialized tests as described in 4 .
A number of scripts had been developed for previous tests and could be re-used: Test cluster handling, test system preparation, the automatic production of the large database series, the automatic long-term execution of the tests and test analysis scripts. These scripts are part of the Online Software release. They are described in detail of the page on the previous Scalability Tests of the Atlas TDAQ Online Software System Web page [17]. Hints and instructions for other systems on the Usage of the Online Software in this environment had also been provided on this web page.

4.3 Test result data handling for the online system

For the integrated tests of the Online System, the raw test data were automatically recorded by the test execution scripts. Analysis scripts helped to transfer them to Excel sheets for convenient representation of mean value graphs. Other analysis scripts produced complete sets of histograms of scatter plots of individual runs and specific test series. Generally 274 of such plots were produced per test run. Viewing these graphs helped searching for expected but also for unexpected effects. All the raw test data was copied to CD-ROM. All the processed data and a sub-set of the raw test data is available on the used test account. The Excel graphs including the corresponding source data are accessible via the web page [12] to give easy access for further investigation by the testers.

4.4 Partition set and operational sequence for the Online System tests

The operational sequence as explained in this paragraph is mostly the same as the one for the June 2002 and for the January 2003 tests. This time, the standard number of crate controllers per detector controller was 50 as opposed to 10 in the tests of the previous years.

The information on all the processes controlled by the Online System including their relationships, the Run Control hierarchy in the online system as well as start-up and shutdown dependencies is defined in the configuration database data file. A database data file which represents the partition was used to drive the tests involving all the components. Starting with booted but idle machines, the tests simulate the start of data taking activities by creating all the necessary processes in the defined order and then cycling the system through the states prescribed by the Run Control to simulate a data taking activity. At the end of these cycles, the system is shut-down in an orderly manner and all the DAQ related processes are destroyed. Timing values were written to file and subsequently loaded into a spreadsheet.

The configuration database schema allows to define segments to establish a Run Control hierarchy consisting of a number of detectors. The configuration database allows to use variables when defining
the environment and the command line parameters of an application. This provides for the necessary flexibility when including a number of similar segments into a partition.

A base configuration was defined consisting of one detector controller segment with one monitoring sampler (not started when running the partition) and 50 crate controller segments with monitoring tasks started at boot time. The segments were all defined to run on separate PCs. As the configuration database information is represented in XML it was possible to easily clone the detector controller segments of this base partition 20 times with the help of a dedicated script, to run on different machines. A set of 20 configurations was prepared using the segments as building blocks. An infrastructure segment was added to the first configuration to form the first partition. Then the configuration size was incremented step-wise by adding one of the base configurations to achieve 20 partitions containing 1+51*n detectors, where n=[1...20] as shown in Figure 7. A similar set of configurations constructed of segments containing 10 local controller segments and one detector controller each was originally used in the 2001 tests. The size was increased for the 2002, 2003 and 2004 tests up to a configuration of 1000 controllers. it was constructed by distributing the large partition on the PC farm in order to reach a partition size of up to 1000 controllers and running on 330 PCs. One extra process was started on each sub system segment.

Figure 7 Partition set for the Online Software Integration tests
4.5 Description of the Timing Measurements for the Online System Tests

Timing measurements were performed for the transitions shown in Figure 8 and defined as follows:

Set-up: start online system infrastructure.
Close: remove online system infrastructure.
Boot: start all supervised processes.
Shutdown: stop all supervised processes.
Cold start: start the supervised processes and go to the Running state.
Cold stop: reverse of the cold start phase.
Luke warm start: once all processes are started and the controllers are in the Initial state, go to the Running state.
Luke warm stop: reverse of the luke warm start phase.
Warm start: once all processes are alive and all controllers are in the Configured state (this is equivalent to the Pause state for the online system tests), go to the Running state.
Warm stop: Reverse of the warm start phase.

![Graphical view of data acquisition states and transitions](image)

The analysis of the setup, boot, shutdown and close times allows to test the process management and the initialization of Online Software System components (configuration). The state transitions give information on the communication overhead in the system.
4.6 Results of the integrated Online Software system tests

The integration tests of the online system could confirm its functionality on a large scale for a configuration consisting of more than 1000 controllers and more than 2000 processes already on the second day of testing. The remaining time was devoted to investigating, debugging and understanding important system features and to apply successively improvements. A large number of performance measurements were done under a variety of conditions.

The change of the size, configuration and availability of the tests cluster at times introduced extra overhead and difficulties to run complete test series under similar conditions to compare test results.

4.6.1 Test Series

A number of variants of the test partition sets described in 4.4 were run with the aim of investigating the behaviour of the Online System under different conditions. The most important ones are:

- **mon_standard**: all the controller segments were evenly distributed over all the available nodes
- **mon_server**: the processes play_daq, partition IPC server, DSA supervisor, RDB server, the is servers for PMG and for RC were each running on a dedicated 2.4 GHz PC. The controller segments were equally distributed over the remaining segments.
- **mon_server_rdb**: same as **mon_server**, but all processes use RDB.
- **mon_local_infrastructure**: 1 RDB server and 1 Run Control IS server were running per detector controller, distributed over the cluster so that some of them were running on 660-800 MHz nodes, some on 2.4 GHz nodes

During the first two weeks of the tests, the partitions were distributed over the initial cluster of up to 216 nodes with a performance of 800 Mhz. In general, a testing script was run to start play_daq (run the DAQ cycle from booted but idle machines to the running state and then close the DAQ again) in a loop with one entire partition set (20 partition with 52 up to 1021 controllers). 10 cycles of such a series were generally run which resulted in 200 individual play_daq tests. The timing measurements were automatically recorded. Scatter plots were produced on all the transition values for each partition and combined graphs showing the behaviour of a transition with increasing size were build. Generally 274 of those graphs were produced per test run. The quasi immediate availability of those results allowed to study them and react on problems very quickly. In parallel, studies on the performance behaviour of the communication component IS based on the new Corba implementation OMNIOrb were going on. The results of configuration database tests suggested to perform also integration test series using RDB. System parameters could be tuned and the performance of the system increased.

Once the additional 120 2.4 GHz nodes were made available, the test partition was distributed over all the available nodes. These results are presented here in 4.6.2.
4.6.2 Global Online Software Test Results

Configuration comparison

The following graphs present the mean values of test series with 10 iterations each for the partitions sets mon_standard, mon_server and mon_server_rdb and for 5 iterations for mon_local_infrastructure.

Figure 9 presents the Boot transition, Figure 10 presents the Warm start transition (2 internal state transitions), Figure 11 presents the Luke Warm Stop transition (7 internal state transitions) and Figure 12 presents the Shutdown transition for the partition sets mon_standard, mon_server, mon_server_rdb and mon_local_infrastructure. The comparison shows the performance advantage of the configurations which were running the servers and the infrastructure on dedicated and fast PC’s (mon_server, mon_server_rdb). It also demonstrates the good scalability of the RDB in the integrated Online Software System. The values obtained when running with the local infrastructure show also an advantage over the standard configuration. However it has to be taken into account that this infrastructure was, together with the detector controllers, distributed over both powerful (2.4 GHz) and less powerful (600-800 MHz) PCs. As the system has to wait for all the processes to reach the next state before completing the step, it is assumed that the transitions times necessary on the slow PCs dominate the integrated values. Further tests distributing the servers on fast PCs only should be done at a later time.

![Figure 9 Boot transition for the partition sets mon_standard, mon_server, mon_server_rdb and with local_infrastructure](image)
The observed two seconds offset for the luke warm stop transition is due to an implementation artefact, that has been identified and will be eliminated in a future release of the software.

**Figure 10** Warm start transition for the partition sets mon_standard, mon_server, mon_server_rdb and with local_infrastructure consisting of 2 internal state transitions

**Figure 11** Luke Warm stop transition for the partition sets mon_standard, mon_server, mon_server_rdb and with local_infrastructure
A time-out of 30 s had been set and was hit for the shutdown transition for the mon_standard partition at the time of running. Other test series where this time-out had been increased showed that the curve continues up to the maximum number of controllers along the same slope as between 300 and 700 controllers. The peak value of 4.3 at 817 controllers for the mon_server partition in the Luke Warm Stop is due to one single very high value (out of 10) and probably a disturbance coming from applications running simultaneously (and unexpectedly) on the farm.

**Comparison with distribution over clusters with different PC performance**

Another test aimed to look at the impact of the PC performance on the transitions. Two test series were run. In the first, the mon_standard partition was distributed over 120 800 MHz nodes and in the second distributed over 120 2.4 GHz nodes. The impact on Boot (Figure 13) and Shutdown transitions was not very significant whereas the performance increase for the state transitions was over 100% for 1000 controllers (Figure 14).
Comparison to results from Large Scale Performance tests in 2003

Compared with the measurements from 2003, the measurements of 2004 showed a significant improvement per one state transition (0.6 s in 2004, 1.7 s in 2003) for 1000 controllers. For the boot and shutdown transitions similar results as in 2003 were obtained when running with dedicated servers. Since the last performance measurements in 2003, the Run Control implementation was completely revised by using CLIPS. The tests demonstrated the improvement of the implementation for the state transitions. As the current implementation had aimed primarily on functionality aspects, further improvements can be expected in the future.

4.7 Information Service tests

4.7.1 Description of the Timing Measurements

4.7.1.1 Specific tests dedicated to the Information Service component

The Information Service (IS) component is used to share user defined information between applications in the Trigger/DAQ system. It is responsible for a large number and potentially high rate of the monitoring data exchange. A specialized test was set up for the IS with the aim of studying the IS behaviour in a large scale system and understanding potential limits of the IS itself as well as the limits of the underlying CORBA technology.

4.7.1.2 Test applications

There are three applications which have been used for the IS tests:

- is_server - this application implements the IS repository functionality.
- is_test_source - this application is a configurable test for the IS information provider.
- is_test_receiver - this application implements a configurable test for the IS information receiver.

All of them are included to the release of the Online Software. One can run them with the ‘-h’ flag to see their usage.

4.7.1.3 Hardware description

For the IS tests two different test beds have been used. For each of them all the available computers have been subdivided into three mutually exclusive sets. The first set contains only one machine on which a single IS server application was running. The second set consists of 25 machines which have been used to run in total 1, 5, 10, 15, 20 and 25 IS information receivers. The last group includes 200
machines. On each of those machines from 1 to 15 IS information providers have been running. Table 2 shows the characteristics of computers and network for different hardware test beds.

Table 2  IS tests hardware configuration

<table>
<thead>
<tr>
<th>Test bed name</th>
<th>IS server and IS receivers HW</th>
<th>IS providers HW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC</td>
<td>Network</td>
</tr>
<tr>
<td>slow set</td>
<td>dual PIII 800 Mhz</td>
<td>100 Mb Ethernet</td>
</tr>
<tr>
<td>fast set</td>
<td>dual PIV 2.4 GHz</td>
<td>1 GE</td>
</tr>
</tbody>
</table>

4.7.1.4 Test description

Two different series of tests have been done for each set of computers. The first one has been used to find out the mean time of a single information update operation for a single IS server in case it is used concurrently by a large number of information providers. In these tests every information provider performed one information update every second.

In the second series of tests each information provider did a certain number of information updates sequentially with no delay between them. This test has been used to find out the maximum number of information updates, which can be handled by a single IS server depending on the number of concurrent information providers and receivers.

In each test sequence the following steps have been performed:

3. $N$ is_test_receiver applications have been started on the 25 dedicated machines, one per machine. Each of them subscribed for all the available information in the given IS server.

4. $M$ is_info_source applications have been executed on each machine from the 200 computers set. Each of those applications created one information item in the given IS server and then updated this information 500 times either once per second for the first series of tests or as fast as possible for the second one. The time of each update was measured and stored to the dedicated file in the local file system.

5. All the is_test_receiver applications have been stopped.

For the “slow” configuration the number $N$ was 0, 1, 5, 10 and 15 and the number $M$ was incremented from 1 to 10 with step 1. For the “fast” configuration the number $N$ was 0, 1, 5, 10, 15, 20 and 25 and the number $M$ was incremented from 1 to 15 with step 1.

4.7.2 IS Test Result

All the test sequences described in the chapter 4.7.1.1 have been successfully completed. There were no errors observed. No messages have been lost.

4.7.2.1 “Slow” test bed results

The results, which are presented in this chapter, has been taken on the “slow” test bed.
Figure 15 and Figure 16 show the mean time for one information update as function of a number of concurrent information providers and receivers. Figure 15 has a logarithmic scale in order to show the whole results and Figure 16 shows the same plot in the linear scale to present the behaviour of curves in more details.

For up to 1000 information providers the behaviour of the curves is linear and the mean update time is in a range of 10 ms. For 1 and 5 receivers the curves are linear up to the 2000 providers. For 10 and 15 information receivers the mean time starts to grow exponentially from 1200 providers onwards, which indicates saturation of the IS server. This can be explained by the existence of a limit for a number of requests the IS server is able to process in one second.

These results are very similar to the ones, which have been obtained in January 2003 using very similar hardware configuration. This is very important because we were using different CORBA implementation in January 2003 and now we can be certain that the new CORBA broker (omniORB) is at least not worse in terms of performance as the previously used one (ILU).
Figure 17 shows the number of update requests a single IS server is able to handle per second as function of a number of information providers and receivers.

![Figure 17](image1.png)

Figure 17 Number of requests per second, which a single IS server can handle.

### 4.7.2.2 “Fast” test bed results

The results, which are presented in this chapter, has been taken on the “fast” test bed.

Figure 18 and Figure 19 show the mean time for one information update as function of a number of concurrent information providers and receivers. Figure 18 has a logarithmic scale in order to show the whole results and Figure 16 shows the same plot in the normal scale to present the behaviour of curves in more details.

![Figure 18](image2.png)

Figure 18 Mean value for one information update
For up to 1400 information providers the behaviour of the curves is linear and the mean update time is in a range of 10 ms. For 1, 5 and 10 receivers the curves are linear up to the 3000 providers. For 15, 20 and 25 information receivers the mean time at some point (2800, 2000 and 1600 providers respectively) starts to grow exponentially in the same way as in case of the “slow” test bed measurements.

Figure 20 shows the number of update requests a single IS server is able to handle per second as function of a number of information providers and receivers.

It has been realised after the measurements have been done already that the values, which have been measured for the 0 receivers configuration are not reliable enough, especially with the small number of
information providers. The reason is that information providers were not very well synchronised for that measurements. While some of them just started to update the information some others may finish already. Therefore they were not really working in parallel and the measured values are probably high then they really should be.

The tests, which contain a large number of information providers (more then 1000), last long enough to ignore the impact of that factor to the measured values. So that the results obtained for the configurations with more then 1000 providers are reliable enough. The same applies to the tests, in which 1 or more information receivers have been used.

### 4.8 Configurations Database tests

The configurations database component is used to keep the set of parameters describing TDAQ and provides access to it.

The database is accessed simultaneously by many TDAQ applications during initialization phase. Each application may read different sets of parameters in accordance with it’s needs (varying from several bytes up to tens of Mbytes) and subscribe on notification in case of their modifications. The total number of applications including high-level trigger may reach in the final system up to several thousands. A single server is not capable to implement the required database service for performance reasons and therefore it will be necessary to run several such servers in the final system. The main goal of the configurations database component tests was to find the performance and scalability parameters of the single database server to be able to calculate the required number of servers during commissioning phase and for the final system.

Since the last large performance and scalability tests [3] there were several important changes in the configuration database component implementation:

- the database access is implemented via automatically generated data access library
- the low-level mechanism for communication between server and client application was changed from ILU to OMNI broker
- optimization of the server interface for passing sequences of primitive types

The goal of the tests also was to prove that the new configurations database component implementation is capable to function correctly in a large scale with the changes listed above.

#### 4.8.1 Test Applications

The test applications are standard applications from Online Software release:

- the database server was the standard *rdb_server* from online release;
- the database client was the standard *dal_dump_rdb* application from online release;
- the standard *ipc_server* from online release was used to provide the naming service.

The database server and IPC server were ran permanently during the tests. The clients were started one per node for each new test. After start up, each client established a connection to the database server, read the required portion of data and closed the connection.
4.8.2 Hardware Description

All available nodes were split on two sets:

- fast nodes: dual Pentium 2.4 GHz on 1 GB Ethernet
- slow nodes: dual Pentium III 600-800 MHz on 100 MB Ethernet

The database server and IPC server were permanently ran on two dedicated fast nodes. All clients were run on slow nodes (one database client per node).

4.8.3 Tests Description

The following tests were performed on all available slow nodes from 10 to 220 with 10 incremental:

- read small single object by object identity
- read small single object by query (search by name, no index)
- read composite object (consists of 2042 small nested objects)
- read big single object containing array of integers (1 MB, 4 MB, 8 MB and 16 MB)

Each single test was repeated 10 times.

4.8.4 Tests Results

All the tests were successfully completed without an error. No memory leaks were found on the database server. No requests were lost.

4.8.4.1 Read single object

Below there are results of reading single object by object identity and by query (no index):

![Chart showing average and maximum values of reading single object](image)

**Figure 21** Average and maximum values of reading single object

The left chart represents average values and the right one represents maximum measured values. The average value linearly grows for both types of object access. The maximum value in some cases is
significantly higher than the average value because of underlying tcp/ip protocol internal delays. Such 3
seconds delays can appear when many clients try simultaneously establish connection with single
server. However for our tests the probability of such delays was less than 0.1% and practically did not
affected the average values.

4.8.4.2 Read composite object

Below the results are presented of reading one composite objects consisting of 2042 nested objects.
Each single object is relatively small (several hundreds of bytes), but it is important that each object is
read in a separate network operation between client and server. Such results can be used to indicate how
much time is required for sequential access of a given number of objects.

Figure 22 Average and maximum values to read composite object

The maximum and average results are quite close.

For 210 clients the server processed around 430,000 requests in 55 seconds, or about 8,000 requests per
second.

4.8.4.3 Read array

Below there are results of reading big single objects containing arrays of integers. The tests were
performed for objects of 1M, 4M, 8M and MB of size.

Figure 23 Average and maximum values to read integer array
The average and maximum times differ by a factor of 2 from 20 clients onwards. This is explained by the parameters used to configure the database server. The maximum number of processing threads was set to 10. This means the server transfers the data simultaneously to 10 clients. Others clients need to wait until one of the previous transfers has finished. The number of simultaneously processed client requests can be increased to reduce the difference between maximum and average time, but this will decrease the overall server performance because the switching between many parallel threads introduces again overhead. The server configuration should be set according to the actual needs of the DAQ system.

The behaviour of 8 MB graph can only be explained by external “noise”, since the network was not isolated during the tests.

In case of 16 MB object and 210 clients the server passed about 3.36 GBytes in 50 seconds, i.e. the performance was 538 Mbits per second on 1 GBit network.

4.8.5 Conclusions

The tests prove that the current implementation of the database component is reliable and scalable for the tested number of clients.

When an application reads single small objects the response time is negligible. The only problem is that a tcp/ip internal delay appeared, when too many clients try to establish a connection with the server simultaneously. One should keep this in mind and set the connection timeout to a value which is not too short.

For the tested numbers of simultaneously connected clients the server is capable to process about 8000 requests per second. This allows to make an estimation of the time to get data from a single server provided the number of clients and number of objects they read is known. For a big number of clients accessing the same server proper data organisation and access methods are strongly desirable (for example avoid the unnecessary scanning of a large database, access objects directly if possible).

The code of the database component was optimized before the tests to pass big arrays. This allowed to reach 54% of the raw network performance.

4.9 SETUP tests

The SETUP component starts and verifies the functionality of all necessary Online S/W infrastructure applications, defined in the ‘setup’ segment, which is included in every partition. It also starts partition-independent processes like PMG agents. Agents are started via the remote shell facility, and all other applications are started with help of PMG.

Setup also launches tests which are defined in Tests Repository database to verify the functionality of the components (application, PMG agent and computer) in the configuration.
The aim of scalability tests for Setup component was to measure the time to initialize and verify the functionality of the various parts of the infrastructure. The following times were measured:

- **Cold** setup: time from scratch to the “Initial” state, from where DSA can start RC tree. It includes:
  - Test of the h/w components (Computers)
  - Start & test of PMG agents
  - Start & test of infrastructure applications
- **Warm** setup: all agents are running, only set up partition infrastructure
- **Recovery** setup: all components are running, only testing of agents and infrastructure is done. (This is an emulation of “recovery” procedure, where a single problem is to be identified by means of testing.)
- **PMG** setup: Cold minus Warm time. Time to start and test only PMG agents. This time was not measured, it is calculated from other measurements.

Partitions of different sizes, in terms of number of involved nodes and therefore PMG agents, were used in measurements. Maximum number of available agents (and nodes) was 337.

The results of measurements are presented on Figure 24:

![Figure 24](image_url)

**Figure 24** Setup time versus number of PMG agents.

The “PMG setup” curve in Figure 24 shows a good level of linearity with respect to the number of PMG agents. It takes 40 seconds to initiate SSH connections to 337 computers, and to start and test the remote PMG agents. In the current implementation the SETUP is centralized in one process, which is a
performance limiting factor. One should keep in mind that in the future, the setup functionality will be distributed over the whole RC tree.

The behaviour of other curves is less linear due to a number of reasons:

- all tests are executed on the partition default host, which also runs all the infrastructure processes and eventually becomes saturated
- the time, required to initialize and to test some components which load the database (like DSA supervisor and RDB server) depends on the size of the database. This grows with the number of agents. A particular problem with the test of the DSA supervisor, which took more than 1 minute for the biggest partition, was identified and already fixed in the next version of the software.

### 4.10 IGUI tests

The Integrated Graphical User Interface (IGUI) is one of the Online Software component. It is intended to present the status of the data acquisition system and its sub-systems (Dataflow, Event Filter, Online Software) and to allow the user to control its operation.

The aim of the IGUI tests was to verify the functionality of all the graphical panels for a large data acquisition configuration.

In order to test the IGUI, a large test configuration has been prepared. The configuration contained 10 detector segments, each one with 10 crate segments. The crate segment was controlled by a controller publishing operational monitoring information during the RUNNING state, with an information structure similar to the information published by the Data Flow ROS application. This controller behaviour to publish random values every second was obtained at the level of the controller clips file.

Running the partition with the IGUI showed that the information was correctly updated in all the panels. The tree structure used in the “RunControl” and “Segments & Resources” panels is appropriate for large size partitions. Some limitations have been observed in the DataFlow panel. In the actual implementation the history is maintained in the IGUI (as time series) for all information attributes. After some time, the application size grows and the IGUI becomes slower. A possible improvement would be the use of the new “information history” facility of the Information Service and to load in the IGUI only the information needed for the item selected to be displayed as a chart.
Figure 25 shows the DataFlow panel for the test partition.

During the tests, the OptimizeIt tool of the Borland JBuilder have been used to obtain the profiling information.

4.10.1 Conclusions

The IGUI can be used for control and status display for a large configuration. The implementation for the display of the information as charts in the Data Flow panel should be reviewed to improve the performance.
4.11 Conclusions and Recommendations for the Online Software

For the Online Software system, these tests performed in 2004 represented the 4th iteration of large scale and performance tests.

- Partitions with more than 1000 controllers including more than 2000 processes in total could be run successfully from the second day on.
- The new implementation of the Communication Software has proven to work very well on a large scale.
- The new implementation of the Run Control Software has proven to work very well on a large scale. The state transition time could be significantly improved.
- RDB can be used well on a large scale when using efficient data organisation and data access methods.
- The tests for the Setup component showed that it can be deployed on a large scale and will improve the performance of the start-up and shutdown procedures. The standard deployment of the SETUP component will help improving the setup phase time wise and in terms of reliability and fault tolerance.
- The IGU can be used successfully at large scale.

HLT/DAQ integrated tests gave useful feed-back of the Online Software when being used in a more realistic context. Code improvements and tuning of system parameters could be applied and the software re-tested during this testing phase due to efficient analysis scripts and ‘online’ analysis of results.

The 2004 Online Software tests profited a lot from experience gained in previous large scale tests in setting up the testing sequence and the test series and in general test organisation. The re-use of scripts and testware was very effective.

Recommendation for the use of the Online Software on a large scale

The tests of the various configurations showed that it is of significant performance advantage on a large scale to run servers, SETUP, and the IGU on dedicated and powerful PCs. The IGU is best run locally. If the scale increases even further then additional servers can be added to the system configuration. It should be noted that the performance of the server node is more important than the number of servers.

Future

Large scale and performance tests are necessary for the understanding of the behaviour of the integrated Online Software system and its components and will have to be iterated in the future. It is suggested to run the next tests in early 2005. The functionality on a large scale needs to be studied also in the future in order to optimize the performance and reliability and to achieve understanding for the tuning of system parameters. Partially new or enhanced software implementations will have to be verified.

The need for a suitable test bed setup for fine tuning of the system came up such that regular medium size tests can be performed. More ‘close to real’ tests of the Online Software as part of the integrated TDAQ should be performed to allow for the identification of problems or shortcomings at an earlier stage.

The tests showed that the reliability and timing has been improved. Further optimization is possible. Future development and tests should concentrate on the reliability and fault tolerance aspects.
5 Specific tests dedicated to Web Services through a distributed histogram demonstrator

Web services are a series of evolving standards that are being designed and specified by the Worldwide Web Consortium (W3C) to foster cross-platform program-to-program communications.

A real-time histogramming demonstrator called HiWesD (Histogramming Web Services Demonstrator), based on several Web services packages, especially Axis from Apache. The goal of such an exercise was to understand Web Services standards and to evaluate the possibility to integrate them into a control/monitoring system. In 2003, HiWesD was ported to C (called HiWesD-c) using the gSOAP toolkit, a compiler tool which provides a SOAP/XML-to-C/C++ language binding to ease the development of SOAP/XML Web services and client application in C/C++ [2]. The goal of the exercise was to understand the toolkit. In the context of the March 2004 tests described in this document an one-day farm-size test was performed, which has used HiWesD-c as test application. The test focused mainly on gSOAP toolkit, in terms of scalability, reliability and performance. Functionality aspect of HiWesD-c have also been considered.

The results of the tests were very successful and encourage the use of gSOAP. Fit was found that a gSoap application is easy to implement, its interoperability is very good and the implementation is lightweight, with low resource demand in terms of cpu, memory and bandwidth.

Web Services can be an alternative technology for farm-size application management, especially in the domain of monitoring. A detailed description of the implementation, the tests and their results is available in [15].

6 Experience, Conclusions

In the March 2004 large scale tests, the scope was extended for the first time from component and sub-system tests to testing of an integrated HLT/DAQ system. Although the original aim of achieving a fully integrated large scale system was not realised the tests provided a major impetus to HLT developers to get software ready for use in the 2004 combined testbeam. The standalone HLT tests confirmed that small configurations work, establishing that the current implementation of the software could be used successfully at the testbeam. The HLT goals achieved during the large scale tests are listed in detail in section 3.5. Due to the severe instabilities observed in the standalone EF tests when trying to run at large scale, the mixed success of the corresponding large scale LVL2 tests and large scale problems with the EB, no attempt was made to run a fully integrated system. Informal discussions following the large scale tests have identified the main issues leading to the failure of the integration and these are listed in detail in section 3.5.

In general the Online Software sub-system tests were a success, benefiting as they did from experience gained during the three previous large scale tests outlined in chapter 1. The Online Software components in release 00-21-02 have been shown to work well at a large scale and useful recommendations for the use of the Online Software in large systems have been developed. The performance values recorded for the Online Software system tests, the Information service tests and the configuration database tests show good results. Further discussion of these results can be found in the
conclusion in section 4.11. Database generation scripts, execution scripts and analysis scripts were revised and updated and testware was provided.

Although the final goal of running a fully integrated system was not achieved, useful experience has been gained when preparing the integration test on a large scale. The unsuccessful attempt to run a fully integrated large scale system demonstrates the importance of scheduling a further integrated test at large scale in the future. It is important that lessons are learnt from the instabilities observed when running the HLT but also from the success of the Online Software sub-system stand-alone tests.

It is essential that the timing and schedule of future large scale tests, the plans for HLT/DAQ releases and their testing and the plans for other tasks are well coordinated within the project planning. A need for improvement of the testing environment was identified in the areas of system management, farm management, in-time preparation of scripts and testware, farm monitoring and visualization of process management. Tester and expert debugging and diagnostics tools and fault tolerance options in a testing environment should be improved to allow an HLT/DAQ configuration to run even in case of faults.

7 Future Steps

A number of issues have been identified which must be addressed before another attempt is made to run a large scale integration test, these are:

- large scale testing is important and sufficient manpower from all parts of the HLT/DAQ project must be devoted to this effort; manpower must be continuously available for all stages of the tests: preparation, testing, follow-up. Note: in addition to problems with preparation and testing, the follow-up to the March 2004 tests has been inefficient due to other commitments (combined testbeam)
- a suitable cluster running the appropriate OS on which ATLAS HLT/DAQ has priority to run the large scale tests has to be identified and reserved well in advance for our exclusive use; the IT LXSHARE cluster would be ideal provided these conditions can be met
- availability of mature releases of offline, dataflow and trigger software; these must be stable, well tested and understood
- tests must be scheduled at a mutually convenient time, there must be sufficient time between the release date of the software to be used in the tests and the date on which the tests start
- pre-tests should begin as soon as possible on the HLT testbeds we already have available to understand and correct the problems we have observed e.g. more testing of the LocalController
- testing on medium size tests beds must be performed on a continuous basis
- tools to manage large clusters must be available, e.g. for monitoring processes cluster wide, fast utilities for cleaning up, tools to deal with the lack of a common file system etc.
- availability of tested generation scripts to populate OKS databases
- improved fault tolerance to enable the system
- availability of diagnostic tools
- availability of debugging tools
A proposal for the next tests taking the points mentioned above into account is in preparation [19]. Requirements will include the availability of a larger system approaching the scale of ATLAS on which to run the tests and the inclusion of HLT algorithms with the necessary access to configuration and initialisation databases.

8 Acknowledgement

We thank the CERN IT division for providing the access to the CERN IT LXSHARE test bed and our IT colleague Bernd Panzer-Steindel for their kind support when setting up the PCs and during the scalability tests.