ATLAS

ATLAS DAQ/HLT Software

Large Scale Functionality and Performance Tests July 2005

Document Version:	1.0
Document Date:	28 November 2005
Document Status:	Final
Document ID:	ATL-D-TR-0003

Abstract

This document provides a detailed report on the first HLT/DAQ integrated Large Scale and Performance Tests which were performed in June/July 2005 on the CERN IT LXBATCH cluster. DAQ/HLT System Software tests as well as extended component tests were run on a PC farm size of up to 700 nodes. The integrated system under test included the Online Software, the DataFlow software, Level 2 trigger software and Event Filter software without algorithms. Tests including algorithms were successfully run independently for the Level 2 trigger and the Event Filter. Results are presented, compared to previous tests where applicable and suggestions for the further development and future tests are discussed.

The tests were performed by the community of testers and developers in ATLAS DAQ/HLT

Table 1 Document Change Record

Title:	ATLAS DAQ/HLT Software Large Scale Performance Tests - July 2005		
ID:			
Version	**Issue**	**Date**	**Comment**
0	1	25 July 2005	Framework outline generated - Doris
0	9	2 Nov 2005	Edited contributions from all testers into coherent doucment - HP
0	91	21 Nov 2005	Conclusions and Summary chapter added -Doris
1	0	28 Nov 2005	Final edits - HP

References

1 *Large Scale and Performance Tests of the HLT/DAQ Software* http://atlas-tdaq-large-scale-tests.web.cern.ch/atlas-tdaq-large-scale-tests/

2 HP. Beck, A. Bogaerts, D. Burckhart-Chromek, P. Werner, S. Wheeler; *Plans and Actions for Large Scale Tests*; ATL-D-TN-0002; 1 November 2004;

3 *Report about the ATLAS TDAQ Data Challenge Support by IT-FIO/FS*; CERN-IT-Note-2005-001; http://cdsweb.cern.ch/

4 I. Alexandrov et. al., *Large Scale and Performance Tests of the Atlas Online Software* 9-008, CHEP2001 Beijing; http://www.ihep.ac.cn/~chep01/paper/9-008.pdf

5 D. Burckhart-Chromek et al., *Atlas TDAQ Online - June 2002 Large Scale Tests of the Online Software*, ATL-DQ-TR-0005, EDMS 388889, V1.0 https://edms.cern.ch/document/388889/1.0

6 D. Burckhart-Chromek et al., *ATLAS TDAQ Online Software Large Scale Performance Tests January 2003*, EDMS 392827, ATL-DQ-TR-0012 v.1 , https://edms.cern.ch/file/392827/1/ScalTestsReportJan2003_V1.pdf

7 C. Bee et al., *ATLAS Event Filter Tests on the ASGARD cluster at ETH Zurich*, September 2001, ATL-DAQ-2001-007

8 C. Bee et al., A*TLAS Event Filter: Test Results for the Supervision Scalability at ETH Zurich*, November 2001, ATL-DAQ-2002-006

9 S. Wheeler et al., *Test results for the EF Supervision*, ATL-DH-TR-0001, EDMS 374118, V1.0 https://edms.cern.ch/document/374118/1

10 *ATLAS HLT/DAQ Software Large Scale Tests March 2004;* ATL-D-TR-0001; EDMS 499884

11 *EF large scale tests on the Westgrid Testbed in Canada in 2 phases*; http://atlas-tdaq-large-scale-tests.web.cern.ch/atlas-tdaq-large-scale-tests/Westgrid/default.htm

12 ATLAS TDAQ Glossary, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/glossary.html

13 http://www.eecis.udel.edu/~ntp/

14 http://atlas-onlsw.web.cern.ch/Atlas-onlsw/development/Testing/ScalTests/ScalTests.htm

15 2*005 DAQ/HLT Software Large Scale Tests on CERN LXBATCH testbed*; http://atlas-tdaq-large-scale-tests.web.cern.ch/atlas-tdaq-large-scale-tests/June2005Tests/June2005test.htm

16 S. Kolos, I. Scholtes; *Event Monitoring Design*; ATL-DQ-EN-0023; March 16, 2005

List of Figures

List of Tables

1 Introduction

This document presents and discusses the Large Scale and Performance tests of the DAQ/HLT Software performed in June/July 200 [1], [2]. The integrated system under test included the Online Software, the DataFlow software, Event Building, Level 2 system and Event Filter software without Algorithms. The basic functionality of this integrated system could be verified on the given size of the testbed. Tests including algorithms were successfully run independently for the Level 2 trigger and the Event Filter. The tests verified the functionality on a system whose size is much closer to the final Atlas TDAQ system size than the ones available for general studies. The inter-operability of the system components, the scalability of control and communication functionality and the performance of selected aspects of the integrated system could be studied. The tests were performed on a testbed size evolving in steps from 100 to 700 nodes. The tests were operated at the LXBATCH cluster of CERN IT with special support from the CERN IT-FIO/FS group [3].

For the online software a first series of large scale system integration tests had been performed throughout the year 2001 on 100 nodes and results are presented in [4]. A second series has been performed in the year 2002 on a cluster of 230 nodes and reported in [5], and a third series in 2003 [6]. Large scale functionality tests were performed in the year 2002 to test scaling aspects of the DataCollection software. Event Filter infrastructure control tests were carried out in July 2001 [7] and November 2001 [8] using the Marseille prototype version of the dataflow and supervision software and these tests were repeated with a revised software version in January 2003 [9]. The Large Scale Tests performed in March 2004 [10] aimed for the first time to run integrated tests of Online software, DataFlow, Event Building, Level 2 system and Event Filter software. The recent Event Filter tests which were performed on the WestGrid test facility in April/May 2005 [11] on a farm size of 250 nodes with a peak of 840 nodes provided a valuable binary integration test between the Event Filter software (without algorithms) and the Online software.

1.1 Purpose of the document

The document represents the detailed tests report of the July 2005 Atlas DAQ/HLT 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 used by TDAQ and we refer to the respective systems glossary [12] .

2 Large Scale and Performance Tests

2.1 Scope and Aims

The objective of the large scale and performance tests was the verification of the operational scalability and the study of the limits of the DAQ/HLT system with a configuration containing a large number of nodes. The functional integration of the Online software, Data Flow, Event Building, Level 2 and Event Filter on a large scale with and without trigger selection algorithms was the primary goal. Variations of configurations concerning the partition composition and farm and sub-farm sizes should be studied. Then the performance of selected aspects of the system like communication, databases and state transitions, was of interest. The tests were an important step towards the up-coming deployment of the software system on the installation in the Atlas experimental area and meant to complement tests performed on small test beds.

For the online system, the tests were aimed to confirm the large scale functionality while enlarging the scale and the scope as compared to the tests in 2004. The plan 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 2005 also aimed to verify the correct functionality of new implementations and enhancements of the online software. More specific tests studying the scalability of selected components in the area of Controls, Monitoring and Databases were also performed.

The HLT test were grouped into several groups of tests: The Level2 tests and the EF tests without trigger selection algorithms, the installation tests of the DAQ/HLT software combined with the offline software and their timing measurements, and the tests to run partitions including Level2 and Event Filter algorithms with the installed software.

The Level2 tests on the ROI Collection (L2SVs + L2PUs + ROSs) were performed to test the scalability aspect of High Level trigger LVL2 part of the dataflow without algorithms. It was aimed to study the control aspects, robustness and stability on a large scale, to take measurements for state transitions for varying number of sub-farms and number of nodes per sub-farm as a test for farm control, verify the inter-operability of components using TCP and UDP, study 2tier and 3tier hierarchies, vary the number of sub-farms and number of applications per sub-farm and to verify the functionality for up to 8 L2PU applications per node (few algorithms run multi-threaded; multi-processor nodes) while keeping the data rates on Data Collection (DC) network low (high burn time) not to spoil the measurements.

The EF tests aimed to verify the robustness of the EF control and operational monitoring, study the load balancing on the PTs and the EF communication for SFI, EFD and SFO, verify that the performance is in the expected range and to take measurements for state transitions for varying number of sub-farms and number of nodes per sub-farm as a test for farm control.

The HLT tests including algorithms were first run with 'HelloWorld' and 'HelloWorldLVL1' algorithms. Both algorithms test much of the Athena infrastructure as used in HLT.Then tests including the muFast algorithm were run for the Level2 and the TrigMoore algorithm for EF. Memory consumption, disk access patterns, shared library use and the behaviour at the state transitions were studied.

The DAQ/HLT-I integration tests were planned to follow a series of steps to allow for the testing of sub-system and component groups before combining them to the complete system:

Step 1: Online Software System

- Test the integrated functionality for startup/shutdown and state transitions
- Verify that the performance is in the expected range

Step 2: ONLSW + L2 ROI Collection (L2SVs + L2PUs + ROSs)

- Test scalability aspect of High Level trigger LVL2 part of the dataflow without algorithm. 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
- Inter-operability of components, use of IS and MRS while ROSes use dummy data and UDP (TCP as fall-back solution)
- Performance and scalability aspects of data-movements, to be studied relative to the available network topology
- No aim to measure performance or scalability of the DF system
- ONLSW + L2 ROI Collection + Event Builder
- Components involved: as in step 2 a) + Event Builder (DFM, SFIs, ROSs)

Step 3: ONLSW + EF

- Verify robustness of the eventfilter 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; perform equivalent tests for LVL2 control

Step 4: ONLSW + L2 ROI Collection + Event Builder + EF (step 2+ step 3 together)

- Test integrated functionality
- Take Performance Measurements for state transitions for increasing number of elements
- Verify robustness with the help of operational monitoring while in running state;

2.2 The testbed

The testbed was similar to the one used for the March 2004 tests.Up to 700 PCs of the CERN IT LXBATCH testbed [3] were used. All nodes were time synchronized via ntp [13]. They were equipped with 1 GHz up to 2.8 GHz Dual Pentium III processors, 512 MBytes to 1 GByte of memory running the Cern SLC3 operating system installation with selected system parameters adjusted to the needs of the tests. 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 1

The DAQ/HLT-I Software release in general use was tdaq-01-02-00. Some dedicated tests were performed with tdaq-01-03-00 which includes the new implementation of the control software. For the tests including algorithms the Offline release 10.0.4, HLT-02-00-02 and tdaq-01-02-00 and tdaqc00 The number of dual pc nodes available were increased throughout the testing period as listed below:

- Phase 1: June 06 - 12: installation prior to the tests on 50 (39) nodes
- Phase 2: June 20- 21: 125 nodes
- Phase 3: June 22 - 28: 300 nodes
- Phase 4: June 29-July05: 400 nodes
- Phase 5: July 06 - 12: 500 nodes
- Phase 6: July 13 - 21: 700 nodes

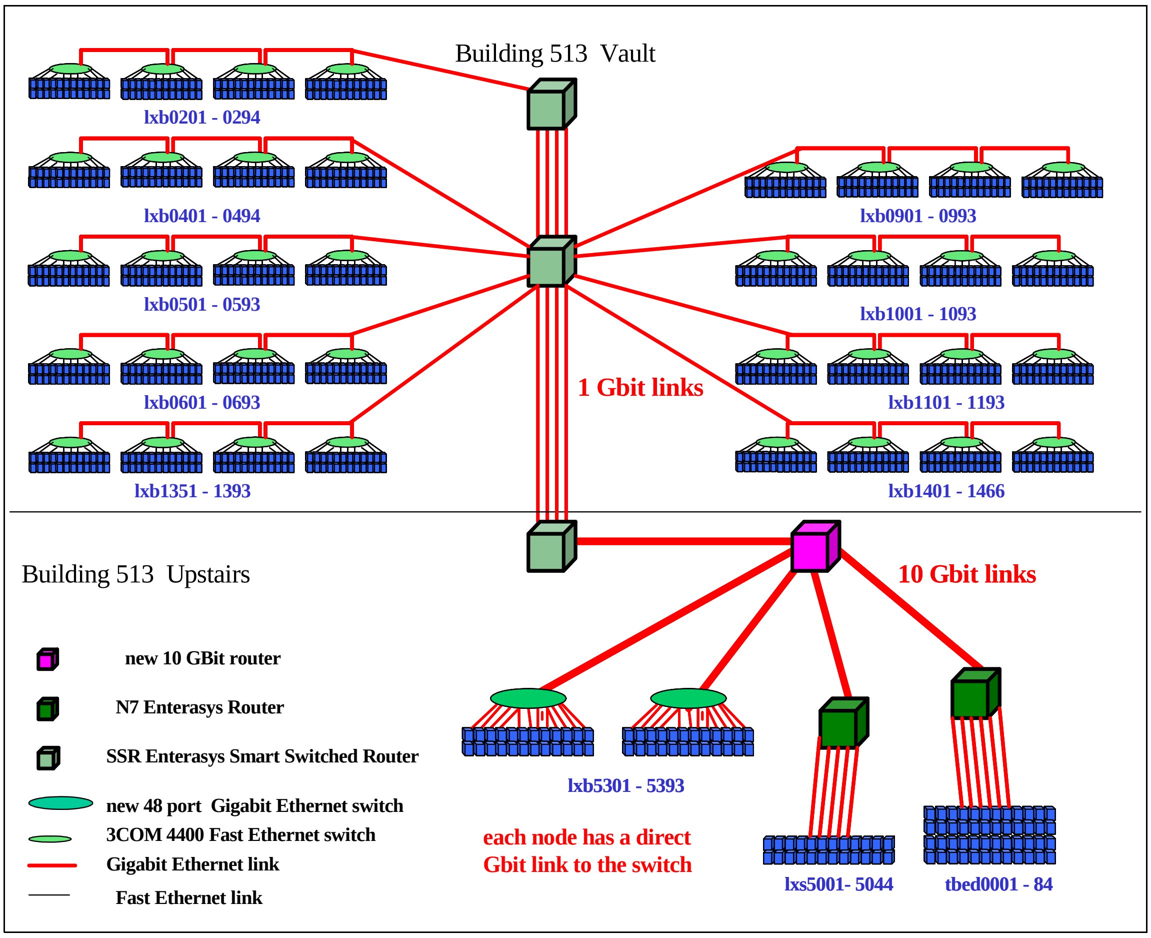
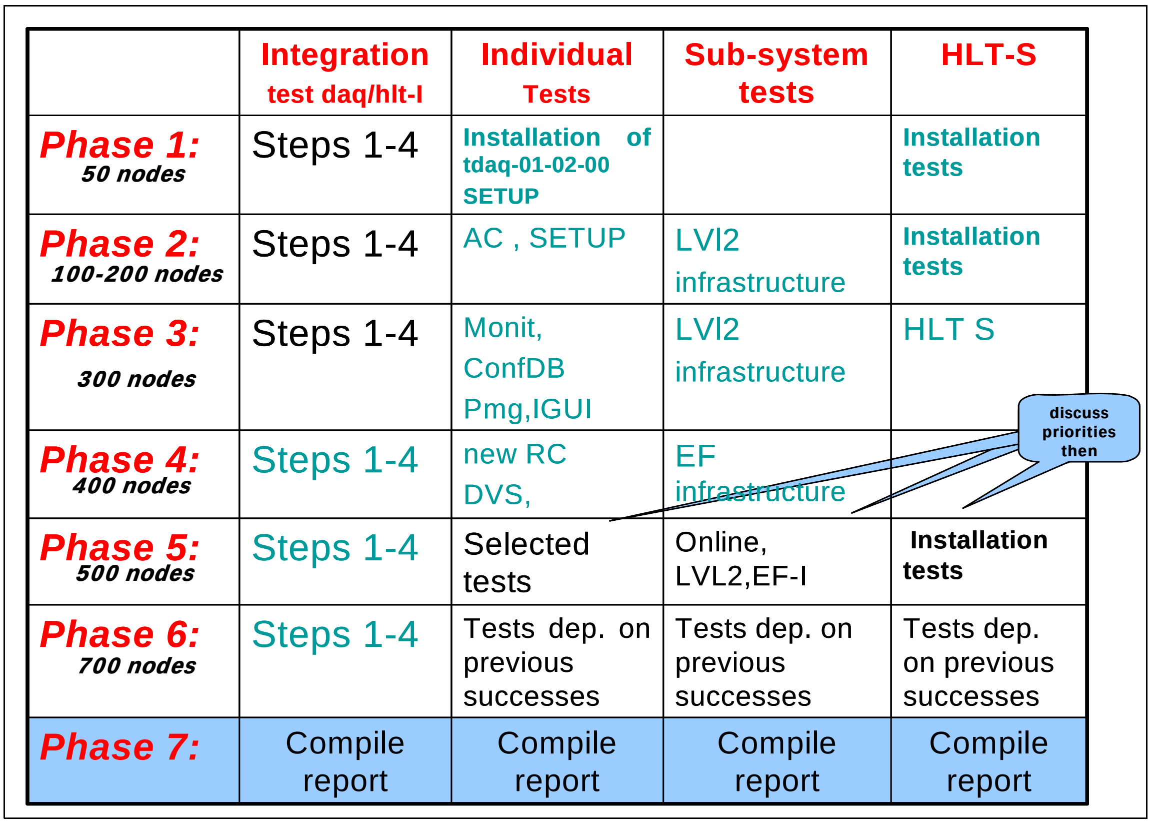


Figure 1 Topology of the LXBATCH testbed

2.3 Test Organisation and Approach

A team of HLT/DAQ testers and developers established the planning for the tests which was presented and discussed in the HLT/DAQ community. A working web page [14] was established containing organisational information, details on testing tools and parameter settings, a description of the testbed and its layout, testbed testing time scheduling, meeting information, and the entry to the online test log book. During the tests, more testers/developers joined the testing team for certain times. Prompt information exchange was encouraged and information was 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. Sub-system tests and integration tests were performed by a team of experts.



	Integration test daq/hlt-I	Individual Tests	Sub-system tests	HLT-S
Phase 1: *50 nodes*	Steps 1-4	Installation of tdaq-01-02-00 SETUP		Installation tests
Phase 2: *100-200 nodes*	Steps 1-4	AC , SETUP	LVl2 infrastructure	Installation tests
Phase 3: *300 nodes*	Steps 1-4	Monit, ConfDB Pmg,IGUI	LVl2 infrastructure	HLT S
Phase 4: *400 nodes*	Steps 1-4	new RC DVS,	EF infrastructure	
Phase 5: *500 nodes*	Steps 1-4	Selected tests	Online, LVL2,EF-I	Installation tests
Phase 6: *700 nodes*	Steps 1-4	Tests dep. on previous successes	Tests dep. on previous successes	Tests dep. on previous successes
Phase 7:	Compile report	Compile report	Compile report	Compile report

Table 2 Schematic test schedule

The schematic work plan is presented in Table 2.

2.4 Tools and Utilities

Many individual tools and utilities were used for running the tests. They comprised automatic execution scripts, analysis tools, farm management tools, command line execution on multiple hosts, and checks on the correct functioning of the running DAQ/HLT system. The tools which were used widely are described in the LST Web pages [15]. Those which were not part of the release already were installed on a general user area. A number of more small test utilities were developed during the tests by the participants and then shared.

Special system settings were also listed in the LST Web pages [15] and updated as feed back experience as the tests were advancing.

3 Online Software System integrated tests

3.1 Aims

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. 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 [4]. A second series has been performed in the year 2002 on a cluster of 230 nodes and reported in [5], a third series in 2003 [6] and the fourth series in 2004 [10].

The concept of distributed RDB servers was introduced and used the first time for these tests. It was needed to allow the use of the RDB on a large scale, a concept needed in preparation for the future implementation of the configuration database access to ORACLE.

3.2 Test Approach for the Online Software System

For the Online System tests, the test approach as explained here is similar to the tests performed in the previous years, and in general described in the Section 2.3, "Test Organisation and Approach". The online software tested here is the infrastructure which is provided by the DAQ to the other parts of the TDAQ system and their processes including the control, information exchange and configuration database access. The work plan was based on the structure of the work plan of the previous years, with details contained in the LST05 web pages [15]. Functional online System tests were performed throughout the testing period because a working infrastructure is needed as a basis for the other tests. Performance test series were run regularly to identify potential scalability problems and to correct them in time so that the next test could confirm the improvement. Performance test series with different controller structures measured on the final farm size of 700 nodes are reported here.

3.3 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. They are documented in the Large Scale Test web pages [15].

A number of scripts had been developed for previous tests and could be re-used: Test cluster handling, test system preparation, the automatic execution of the tests and finally test analysis scripts. These scripts are part of the onl_integ package of the DAQ/HLT-I release. Hints and instructions for other systems on the Usage of the Online Software in this environment were also provided in the Large Scale Test web pages under [15]. For the automatic production of the large database series, the DBGeneration scripts were used. These scripts had to considerably be enhanced and verified to accomodate to the needed functionality throughout the tests.

3.4 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. The scripts were enhanced and adapted to the tests to be performed. Viewing these graphs helped searching for expected but also for unexpected effects. All the processed data and a sub-set of the raw test data is available on the test account atlonl and is also archived. The Excel graphs including the corresponding source data are accessible via the LST05 web page [15] and the LST elog to give easy access for further investigation by the testers.

3.5 Operational sequence for the Online System tests

The operational sequence as explained in this paragraph is mostly the same as the one for the tests in previous years. Different sets of tests varying the number of crate controllers per detector controller were tested this year.

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 a 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

Here the construction of the partition set A as listed in 3.6.1 with results presented in 3.7.2 is described in detail. A base configuration was defined consisting of one detector controller segment and 100 crate controller segments with Example Applications started at boot time. The partitions were generated with the DBGeneration tool. A set of n configurations was prepared with incrementing size, each including an infrastructure segment. The configuration size was incremented by adding one of the base configurations in steps of 100 leaf controllers to achieve 10 partitions containing up to 1000 leaf controllers, plus 2 partitions of 1500 and of 2000 controllers. It was constructed by distributing the large partition on the PC farm in order to reach a partition size of up to 2000 controllers and running on 700 PCs. Each leaf controller had one controlled item attached (ExampleApplication). Additional intermediate controllers and RDB servers were equally distributed as well and the infrastructure processes fixed to a set of dedicated nodes. The infrastructure segment was fixed and distributed over 10 pc’s.

.

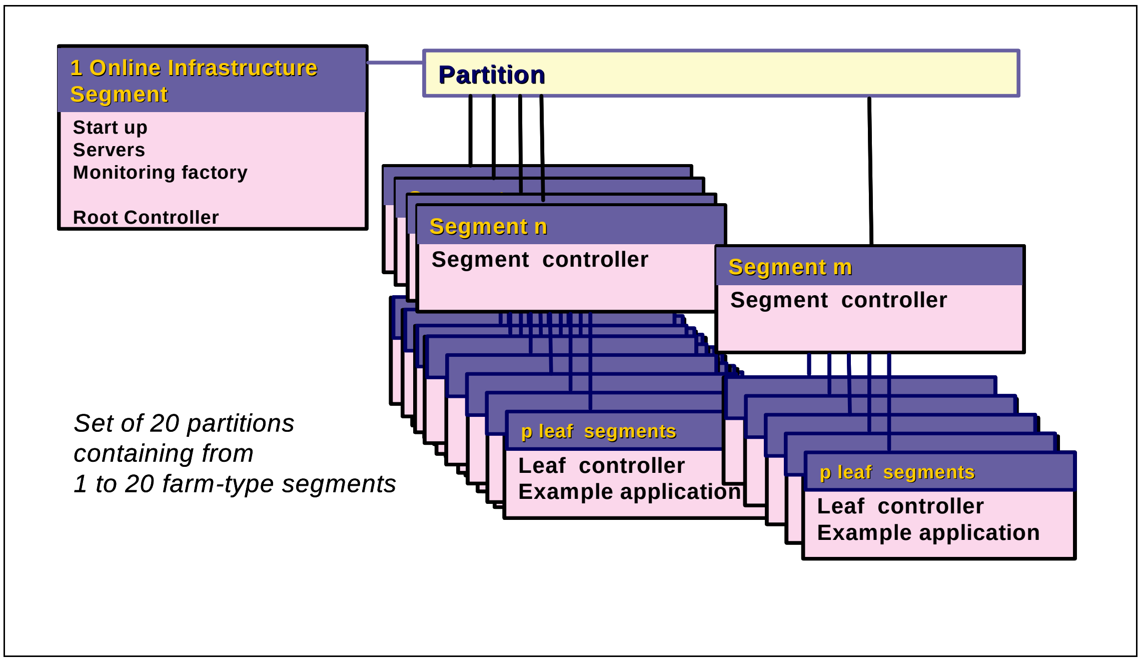


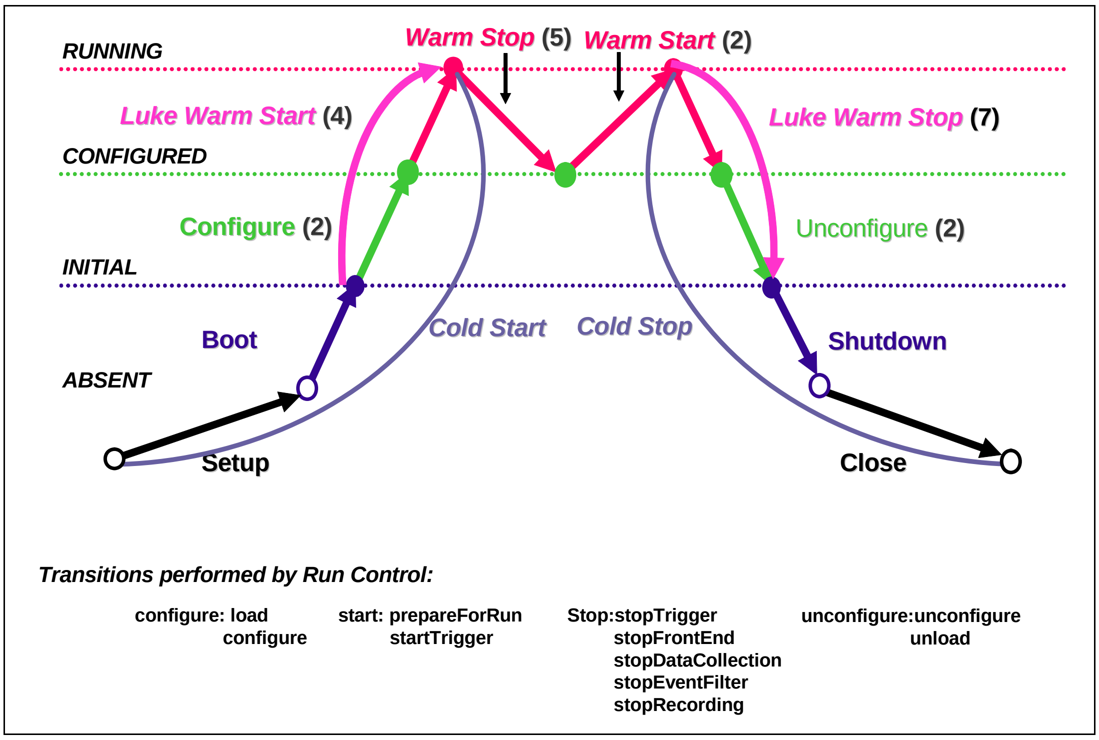
Figure 2 Partition set A as used for the Online Software Integration tests

3.6 Description of the Timing Measurements for the Online System Tests

Two internal methods of measuring the state transitions were in use. Timing measurements were performed for the transitions as shown in Figure 3 and Figure 4 and defined as follows:

Setup: 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.

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.



Transitions performed by Run Control:

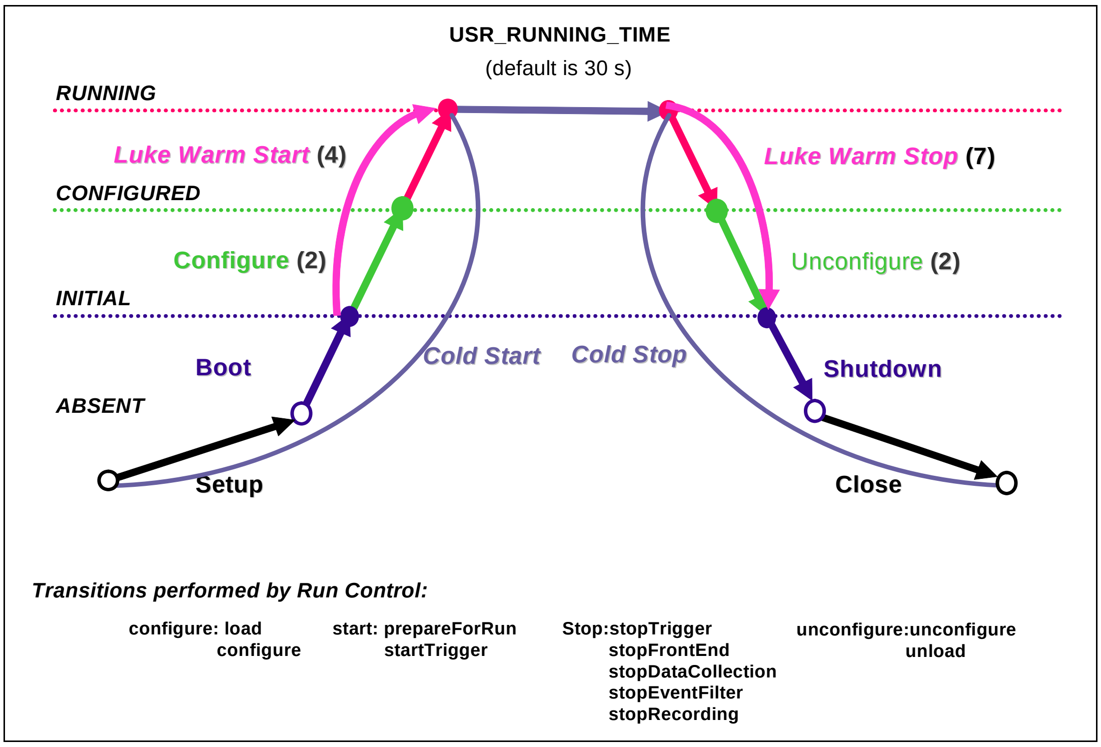
configure: load
configure

start: prepareForRun
startTrigger

Stop:stopTrigger
stopFrontEnd
stopDataCollection
stopEventFilter
stopRecording

unconfigure:unconfigure
unload

Figure 3 Graphical view of data acquisition states and transitions



Transitions performed by Run Control:

configure: load
configure

start: prepareForRun
startTrigger

Stop:stopTrigger
stopFrontEnd
stopDataCollection
stopEventFilter
stopRecording

unconfigure:unconfigure
unload

Figure 4 Graphical view of data acquisition states and transitions including optionally a defined running time.

3.6.1 Test Series

A number of variants of the test partition sets described in 3.5 were run with the aim of investigating the behaviour of the Online System under different conditions. The most important ones are:

- **Partition Set A**: *Configurations with increasing size in a flat controller structure, the size increasing from 100 to 1000 in steps of 100, then 1500 and 2000 controllers; each leaf controller controls one example application*
- **Partition set B**: *Configurations with increasing size, set up with a balanced control structure (n intermediate controllers, each controlling n leaf controller, of which each is controlling n example applications; the largest partition size (n=14) gives 2744 controlled applications for 14*14 +1 controllers;*
- **Partition set C:** *Configurations with different control structures each controlling the constant number of 1000 exampleApplications. The number of intermediate controllers, the number of leaf controllers and the number of example applications per leaf controller varies. The detailed list follows in table 1*

Table 3 Control structure of partition set C

Case	**intermediate controllers**	**leaf ctrl per intermediate controller**	**total leaf controller**	**applications per leaf controller**	**total number of controllers**
1	10	100	1000	1	1011
2	40	25	1000	1	1041
3	40	5	200	5	241
4	40	1	40	25	81
5	0	20	20	50	21
6	0	10	10	100	11

During the tests, the partitions were distributed over the available cluster for each phase. The tests followed the approach as described above.

3.7 Results of the integrated Online Software system tests

The integration tests of the online software system confirmed its functionality on a large scale for a configuration consisting of more than 2000 controllers and more than 4000 processes. 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. Intermediate test results of the configure transition suggested to perform improvements in the use of RDB. System parameters could be tuned and the performance of the system improved. As a consequence of intermediate testing results, a number of improvements were performed during the testing phase and the following tests showed better performance. Testing time was also devoted to investigating, debugging and understanding important system features. Variants of the controller component were investigated. A large number of performance measurements were done under a variety of conditions.

.

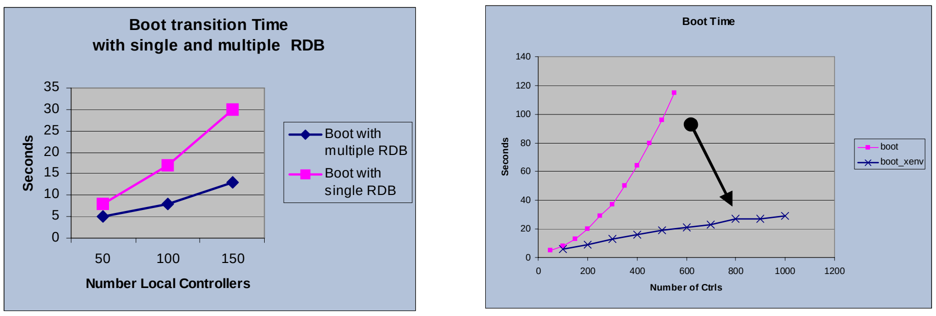


Figure 5 Optimization of the configuration database access

The graphs in Figure 5 show two cases of inefficient database access and the improvement obtained by applying the correct measure. The measures have been applied during the tests in sequence.

- The use of distributed remote database servers (RDBs) is necessary for a system which involves more than 50 nodes.
- Database access is high for infrastructure processes at boot transition. For optimizing the access mode one needs to define the OKS DAL's algorithm option DAL_USE_PATH_QUERY to replace a full scan of the database on the client side by the more efficient query executed by the database server. (This setting was used for debug purposes during the tests and it is default now in the subsequent releases)

3.7.1 Global Online Software Test Results

3.7.2 Partition set A

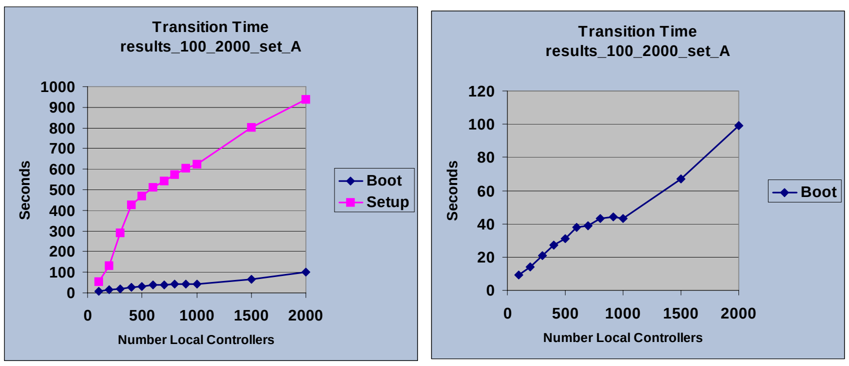


Figure 6 Boot and Setup for partition set A.

In the graphs for the partition set A, see Figure 6, the transitions time per local controller is presented which is equal to the number of controlled applications.

The partition set A shows a graph with steep increase for the setup time which gives a value of 16 s for the number of 2000 controllers plus 2000 controlled applications on 700 nodes. Amongst other operations, setup contacts all the hosts which take part in the run. When the number of tests on the pmg_agent of each node reaches the number of available hosts, a bend in the graph is observed (at about 400 ctrls). The observation of the steep increase of the setup operations with scale has been observed in all the tests. It has been investigated and a solutions is being implemented meanwhile to provide a better scalability behaviour. This unscalable behaviour of the setup component had showed up on all system and sub-system tests and slowed the very large scale testing down considerably.

The steep increase in the boot transition times with scale has been investigated and a solution for better scaling has been implemented in the next release of the controller. There the processes below a tree are now booted by the intermediate and leaf controllers.

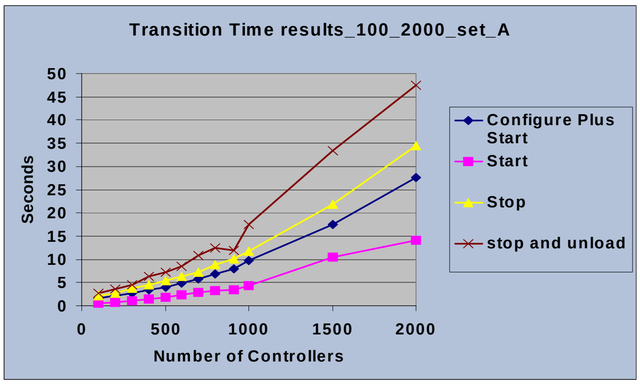


Figure 7 state transitions for partition set A

The state transitions of the test set A show a regular increase with scale, see Figure 7. The various transitions are composed of different numbers of internal state transitions as indicated in the legend. The value for one internal state transition can be calculated and the graphs compare well. The plots are not linear because the processes were overlaid on the available number of nodes (700) and therefore the CPU usage and the bandwidth usage increased with increasing number of controllers and applications.

3.7.3 Partition set B

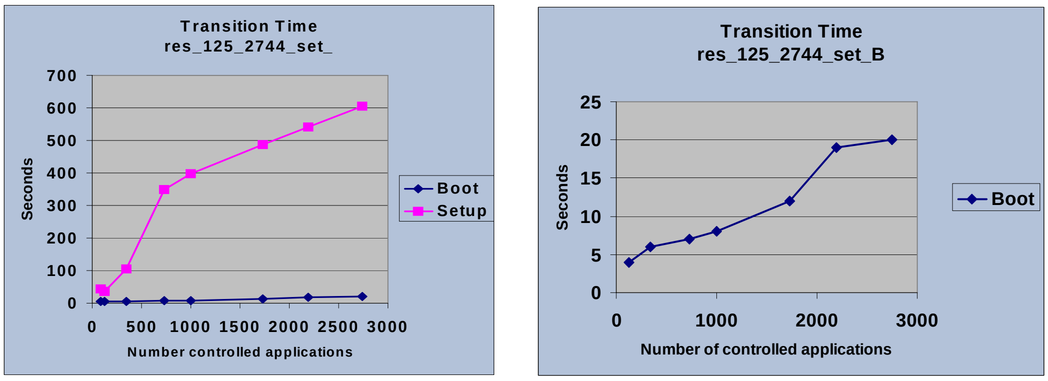


Figure 8 State transitions for partition set B

In the graphs for the partition set B, see Figure 8, the transition time per number of controlled applications is presented, the number of controllers is small compared to the number of controlled items. It has to be remarked that some of the values suffer from low statistics because of problems running the system at very large scale. The boot times compare well with the graph for set A, taking into account that the total number of processes is different for the two cases.

The setup curve is slightly shifted as compared to the one presented above because the number of processes per partition is smaller than for partition set A, when compared to the number of controlled applications.

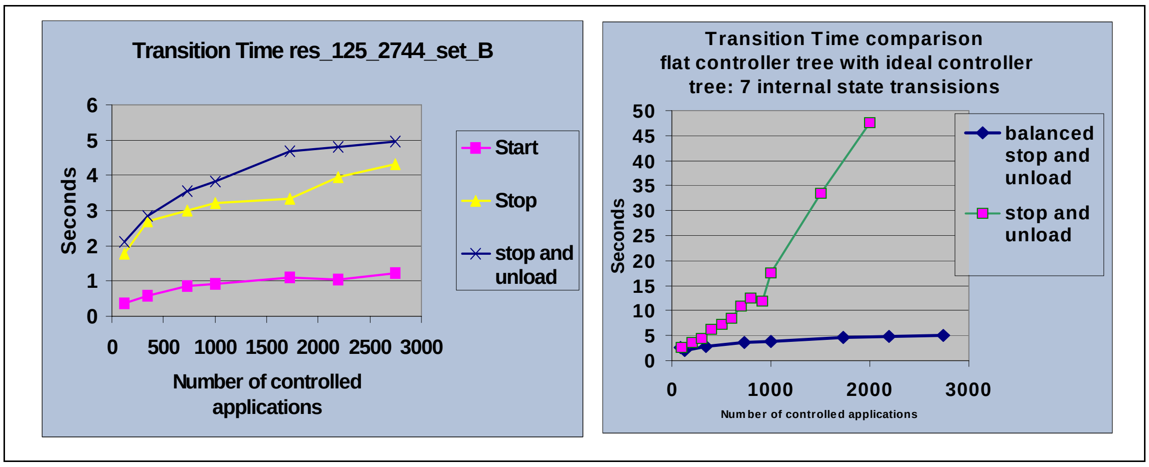


Figure 9 Transition times for balanced controller tree and comparison with flat controller tree transition times

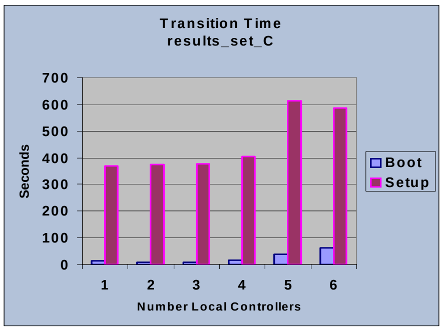
The comparison of the stop and unload transition from set A in Figure 9 with the same transition from set B shows clearly the advantage in choosing a balanced controller tree in a configuration. The results show that the state transition curve for an optimal balanced controller tree (150 controllers for 2000 applications) after an initial small increase is close to flat with 0.7 s per state transition.

3.7.4 Partition set C

The partition set C has been deployed to study the timing behaviour on a large scale with various controller hierarchies. The setup transition shown in Figure 10 is again very high, in particular with a high number of processes. The boot transition shows as expected that the configuration with the lowest number of processes to be booted is the best performant.

The state transition times show low values for the relatively balanced configurations, cases 1,2,3 in a three level hierarchy and then increase with less hierarchy levels and unbalanced tree.

Comparing the graphs C in Figure 11 to the numbers in construction table Table 3, it seems that not the total number of processes is significant for the boot, nor that the intermediate control structure is most determining. The values seems to be high at places where the local contoller has many applications to control. This would suggest that the efficiency of the local controler was not so good compared to the intermediate controller. No confirmation was found in the code for his behavior. As in the new implementation of the controller the local controller implementation is identical to the controller, further investigations were discontinued.



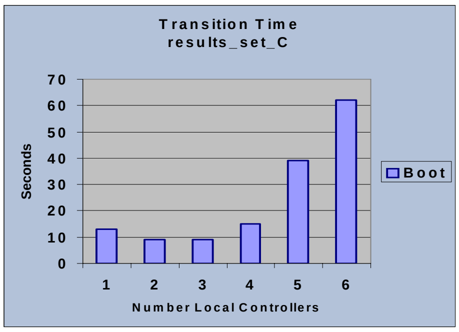


Figure 10 Setup and boot transision for partition set C

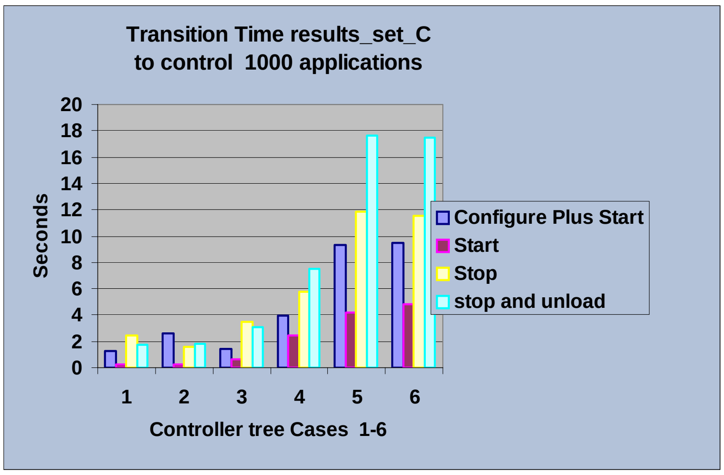


Figure 11 State transitions of partition set C

3.7.5 Comparison to results from Large Scale Performance tests in 2004

The measurements in the previous years had shown for one state transition 0.6 s in 2004 on 330 nodes. These value were obtained for a partition containing 1000 controllers which have no items to be controlled. The configuration in 2005 contained one example application per leaf controller. Each example application is a controlled item, receiving the state transition requests from its leaf controller and therefore involving the corresponding communication overhead. Therefore the values cannot be directly compared. The boot transition shows similar results as in 2004 when running with dedicated servers. The shutdown time has decreased significantly presenting a good improvement.

3.7.6 Experience

It has been the 5th time that tests for the online software were run since the first large scale tests in 2001 and the experience from organising and running large scale tests in the past has helped significantly.

The same strategy could be used for setting up the test series as 5 years ago, obviously running the latest revision of the software on largely increased scale as compared to the farm size of 100 nodes of the first large scale test at the time. Running the online infrastructure tests was less straight forward than in previous years for the first weeks for reasons explained in the following, but finally they were nicely successful.

Problematic was the database generation. In previous years an 'online only' approach had been used consisting of a script set which produced symmetric partition sets. This time, the DBGeneration scripts were used which in fact had been deployed and in use successfully for other parts of the system. They contained too much knowledge of a realistic TDAQ system to be used for the pure online system not allowing a number of constructs. Ad-hoc changes were required and implemented and testing of the revised version was time consuming. The requirements, implementation and testing of these type of helper tools should be prepared better in advance next time.

While the running of the test series was in principle well prepared and straight forward with the automatic running scripts and the analysis tools, there was a lack of tools which would diagnose the state of the farms (disk usage, permissions, host key problem) on one hand and the state of the DAQ/HLT system (hanging processes, zombies) on the other hand in case of problems. Many ad hoc scripts were prepared and also shared between the various sub-systems. Many hours were spent in some cases to go through individual steps of diagnosing the system status and cleaning it up in order to be ready for a new run. This was especially inconvenient as the setup time for the very large partitions was up to 16 minutes The the time out values had been increased accordingly and the pmg-agents were running already. Getting back to a usable state of both the farm and the DAQ system could take many hours. Improvements in these areas are strongly encouraged. Faulty situations came up too often and most fault tolerance features were not available for the automatic running mode.

A specific problem experienced concerned the local controller. It crashed about every 1/10000 operations and affected the stable running of tests during the first weeks. This problem could not be solved but a work around was found which enabled the continuation of the tests. The new version of the controller is generic and does not include the local controllers and therefore not this problem.This phenomenon is an excellent example for the usefulness of the tests on a large scale. The problem also occurs on a very small scale but extremely rarely and it is difficult to capture it in order to debug it, not being reproducible. On the large scale involving a very high number of transitions, it occurs at nearly each run at one of the first transitions.

The other limitation on a number of threads is somewhere in kernel, which depends on kernel and thread library versions - to be investigated.

Exercising the infrastructure software without real applications provides us with the basic measurements for the state transitions. It is important having confirmed that these are well scalable, the curve being even mostly flat for an ideal structure. The values are very low if not neglectable compared to the time consumed by the real applications for those transitions.

The Setup component was used the first time seriously. It supports the newly introduced concept of multiple RDBs servers, which was vital for the tests, and therefore play_daq was not a fall back solution. Being a new component, a number of problems were found in the first phases and 4 times patches were provided. A special kernel setting was necessary so that it could function properly. The test pmg_agent failed frequently, automatic retry included. Even after the last patch, serious scalability problem (not solved during the LST): 16 minutes to start a large partitions on 700 nodes. Large partitions were very difficult to start, as the setup component exhibited a non-linear increase in the time it took to start and test pmg_agents. Sometimes we just could not get a partition started, even with drastically increased timeout values, or had to resort to killing all processes belonging to the partition. Often it seemed a matter of good or bad luck whether a large partition would start. E.g. the 16x32 L2PU partition worked fine the first time we ran it, but failed to start an hour later when no change was made anywhere. It was however important to demonstrate the unscalable behaviour so that steps for improvements can be taken.

The boot transition is being improved as well, contained already in the next release

Operating the online software tests as part of the overall testing team was a pleasure and the good collaboration was highly appreciated.

3.7.7 Conclusions

Online infrastructure tests were carried out to verify the functionality and to determine the basic behaviour of the DAQ/HLT system without the overhead introduced by real applications. The concept of distributed RDB servers was introduced and used the first time for these tests. It was very successful allowing the use of the RDB on a large scale, a concept needed in preparation for the future implementation of the configuration database access to ORACLE. The setup component was used the first time in a large scale and a number of findings were reported. Several partition sets with increasing number of controllers controlling dummy applications were run. The partition sets were build following different control structures. The results show that the state transition curve for an optimally balanced controller tree is basically flat with 0.7 s per state transition. A new implementation of the controller was tested in a separate test series which is reported in detail in Section 11.3. With this new controller, similar tests were performed as reported here and the results on the state transition times and controller tree structur were confirmed with this new implementation. The boot transition was considerably improved and the problem with the local controller did not exist any more because in the new implementation only one generic controller is implemented and used for all controllers in the hierarchy tree.

3.7.8 Future tests

In the infrastructure domain new implementations need to be verified on a large scale. This concerns in particular modifications in the operations of the boot/shutdown and the stop procedure, the revision of the configuration DB schema, the usage of the new error reporting system and the resource manager, and the new implementation of the process manager and the corrections for the Setup. Sufficient time must be available for re-stabilizing the software and deploying it on a smaller scale to ensure its fitness for testing and use in large scale.

4 Event Building Tests

4.1 Aims

The full event building (EB) system in ATLAS is foreseen to involve 100 Sub Farm Input PCs (the Event Building nodes) pulling data from 140 PCs hosting the Read-Out-System (ROS). Given the availability of about 700 PCs for this year's Large Scale Tests (LST05), we had a chance to construct partitions with *EB systems up to the full ATLAS scale.*

The main focus was on the functionality, performance and scalability of the control aspects of EB systems up to the full ATLAS scale, because it was understood that the heterogeneous network topology and PCs available to us would not be suitable to optimize and test the event building performance of the system. In particular we wanted to test the times it takes for the event building part of the ATLAS TDAQ system to complete the various transitions of the Finite State Machine of the Run Control sequence.

It was also understood that even some part of the testbed had nodes which were connected via GigaBit ethernet, the specifics of the network topology affects the event building performance of the system greatly. Thus, no attempt for optimization was made and only control aspects were checked in this case too, for comparison with the random assignment of nodes over the whole mixed network.

4.2 Test description and Configurations

In our tests we tried to use a realistic ratio for the number of ROS machines vs. the number of SFI nodes. As in the full ATLAS system we will have around 140 ROS's and 100 SFI's, we have kept the ratio of ROS-to-SFI machines to the value 3:2 where possible

We built configurations were we randomly assigned the following tasks to the available PCs:

i. 6 to 150 ROS nodes, each hosting 12 Read Out Buffers (ROBs) of 1 kByte of "data".
ii. 1 Data Flow Manager (DFM), running in auto triggering mode
iii. 4 to 100 SFI nodes building complete events out of the 12 kByte event fragments pulled from the ROS nodes.
iv. 10 nodes where reserved to host online services. e.g., Information Service (IS), Inter Process Communication (IPC), etc.
v. a varying number of nodes hosting controllers (one per farm of ROS and Event Building (SFI) nodes) according to the structure shown in Figure 1. In order to a serve large partitions we have each farm hosting one replica Remote Database (RDB) Server

We build our partitions using a) scripts to construct the hardware files describing the PCs in our testbed, and b) the DBGeneration scripts to assign tasks (e.g, ROS, DFM, SFI) to these nodes. The DBgeneration scripts were updated regularly during the tests to follow our needs for a unified control structure for all applications and bug fixes related to the increased sizes of our configurations

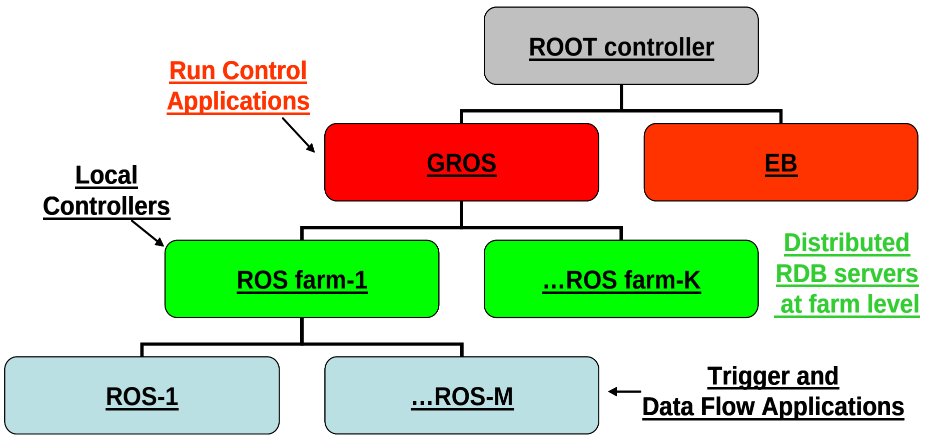


Figure 12 The control structure of the configurations used. The ROS sector is shown, but the Event Building sector is made in an analogous fashion to control the SFI applications.

4.3 Test Approach and Execution of the Tests

We used the tdaq-01-02-00 release of the online software as it was installed locally, with periodic patches to solve problems found during these large scale tests, or elsewhere. We used shell scripts to cycle automatically all applications in a partition through the run control states, shown in Figure 2. Each complete cycling is called a run, and we typically took 10 runs with each partition.

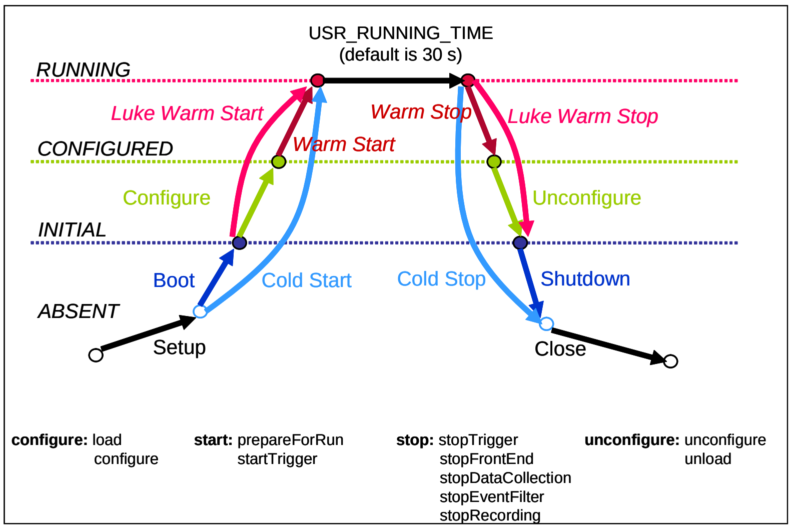


Figure 13 Transitions through the run control states. Note that there is a different number of steps which have to be taken to complete each transition.

4.4 Results

In Table 4 we show the configuration of the partitions we used for the measurements. In particular we show the total number of hosts, the number of ROS and SFI nodes, together with the number of ROS and Event Building subfarms controlling them

Table 4 Characteristics of the partitions used for the timing measurements presented in Section 4.4

# Hosts in partition	**# ROSs**	**# SFIs**	**# ROS farms**	**# EB farms**	**# measurements taken**
25	6	4	1	1	10
37	12	8	2	2	10
61	24	16	4	4	10
109	48	32	8	8	8
205	96	64	16	16	9
328	150	100	15	50	8

In Figures 3 and 4, we see the average transition times in the startup (Setup-Boot-Configure-Warm Start) and the shutdown (Warm Stop - Unconfigure - Shutdown - Close) phase of a run, respectively. The transitions are defined in Figure 1, and the characteristics of the partitions are defined in Table 3-1

In general the transition times are quite acceptable, with the exception of the time needed for setting up the partition, as discussed in the Integration DAQ/HTL-I section (Section 6.4). With the tdaq-01-02-00 implementation of the setup procedure we need about 0.2s per host for partition is organized with 6 ROSs per farm and 4 SFIs per EB farm. In such partitions we see that we need about 0.03s per host.The rest of the transition times are constant with the partition size and quite acceptable.

When we use machines which are connected with a Gbit switch, we see exactly the same picture, except that the setup time is about 50% faster.

Even if we have built and operated partitions with nodes connected via Gbit (up to 80 ROSs and 50 SFIs), the network topology is not permitting large event building rates, so performance tests are of no significance in this environment.

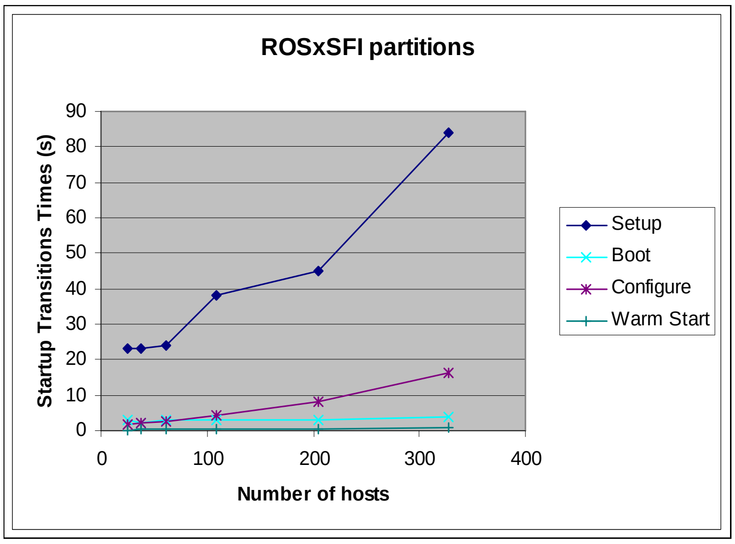


Figure 14 Average times (in seconds) for completing the startup transitions for partitions described in the table above.

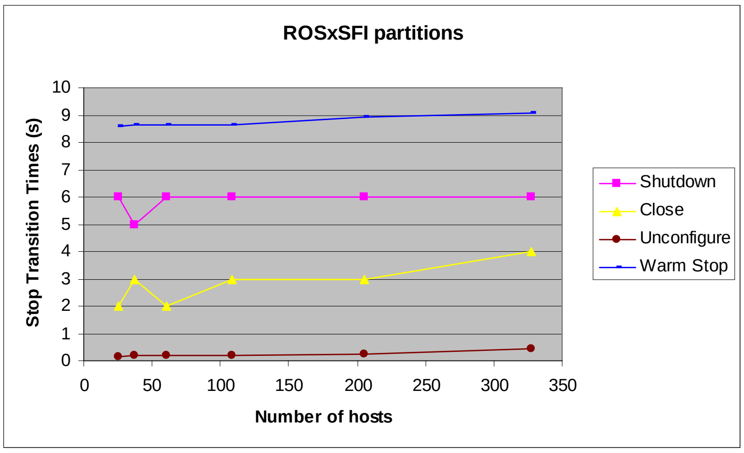


Figure 15 Average times (in seconds) for completing the stop transitions for partitions described in the table above

4.5 Summary and Conclusions

We have tested the event building system up to the full ATLAS scale. We see reasonable transition times and we conclude that we are happy with the control aspects of the event building system.

For the performance measurement, we confirm that the real tests have to be done in an environment with the realistic ALTAS network and realistic PC configurations. So, such tests are not suited for a generic system like the one we used in these Large Scale Tests, but they are to be performed at the dedicated testbeds and the pre-series and then extrapolate the observations to the full ATLAS size.

4.6 Future Steps

Since we are happy with the performance of the control aspects of the event building system, and since we tested the full ATLAS scale, we do not really need any further Large Scale Tests. Any future tests should put emphasis identifying problems concerning the robustness (fault tolerance) of the system, since problems show up faster at large scale systems.

5 LVL2 DataFlow Tests

5.1 Aims

The main aim of these tests was to assess the scaling, stability and operability of a large LVL2 system using the latest stable release of the TDAQ software. The latter included the dataflow applications, but also the software for preparing the ‘partitions’ (generation scripts) and software to set up, configure, run and monitor the applications.

It was also hoped that the large scale tests would help optimising some technical aspects of the LVL2 system: the concept of grouping LVL2 Processors into farms, the depth of the control hierarchy and the scaling behaviour of multiple L2PU applications running on each LVL2 processing node vs. running a single multi-threaded application.

It was also understood that the network capacity of the test system was insufficient to support a reasonable flow of data into the LVL2 Processors. The amount of ‘event’ data flowing through the system was therefore severely limited to a rate of a few Hz if necessary.

5.2 Test description and Configurations

The configurations used for testing the LVL2 system contained:

- ROSs or ROS emulators, typically a total of ~ 100 ROS(E)s to supply data
- LVL2 Processors grouped into farms, typically 1-16 farms containing 8-64 nodes each running 1-8 L2PU applications
- LVL2 Supervisors, one per LVL2 farm
- A tree of controllers, at least one per farm
- ~ 10 nodes for the standard online services such as OKS, IS and monitoring

5.3 Test Approach and Execution of the Tests

The standard TDAQ software distribution, release 1.2, was used for all the tests. The software was installed on the local discs as the global file system, AFS in this case, was not sufficiently performant and robust for ‘online’ applications. The aim was to use tested software but various problems occurred. During the LVL2 test period over ~ 20 patches were installed in ~ 30 days.

Since a large number of different partitions have to be run, testing has been automated using scripts both for generating the required partitions (with a few parameters being varied) and a top-level script for automatic execution and collection of results.

Since configurations with > ~700 applications/node mostly failed, for comparison, some runs were made with ‘test applications’. These are simple dummy applications which obey the standard run control commands but do not require specific configuration data nor even set up network connections for transfer of ‘event’ data.

Many OKS RDB server was used for the configuration. Logs, monitoring, status and error information was written to local discs. Summary information was collected at the end of each run.

A run consisted of cycling all applications in the partition under test through all the run control states. Information gathered was the execution time for each state transistion, or, for failing configurations, at which state transistion the sytem failed.

5.4 Results

The plots in Figure 16 and Figure 17 show the state transistion times for various configurations running with L2PUs or using test applications. Configurations are labeled as **r**ROSE**c_s**L2SV_**f**S_**l**L2PU_**p**P where

- **r** = number of ROSEs
- **c** = number of LVL2 farms per controller
- **s** = number of LVL2 Supervisors
- **f** = number of LVL2 sub farms
- **l** = number of L2PUs per subfarm
- **p** = number of L2PU applications per node

The number of nodes running L2PU applications is hence **s*****l**/**p** for each of the partitions.

The state transistion timing, when a partition runs successfully, as shown in Figure 16 and Figure 17 is quite acceptable. The setup time obviously takes much longer (and is more variable) then all other transistions but this phase should normally not be executed under more realistic conditions. The transistions (un)configure, start & stop are typically application dependent. More realistic (longer times) are obtained when running with algorithms. .

The plots in Figure 18 to Figure 21 show the success/failure when running L2PU or test applications. The vertical axis shows at which point in the configuration cycle (setup, boot, config, start, stop, unconfig) the partition fails. Figure 18 shows this for an increasing number of number of nodes running L2PU applications (up to 698 plus ~ 10 nodes for online services). For comparison, Figure 19 shows this for test applications instead of L2PUs.

The same data is shown in Figure 20 and Figure 21, but this time showing the number of L2PU or test applications (up to 1238).

There have been numerous occasions when partitions failed that are not shown in these plots. This includes such scenarios as too short time-out values (tuned for smaller and more predictable laborarory test beds), un-clean machines with applications from previous tests that did not get aborted (often occupying communication ports needed in the next run), machines in an undefined state that could be reactivated by rebooting and other problems that could be solved and corrected during the test period (by changing configuration details or applying patches to the code).

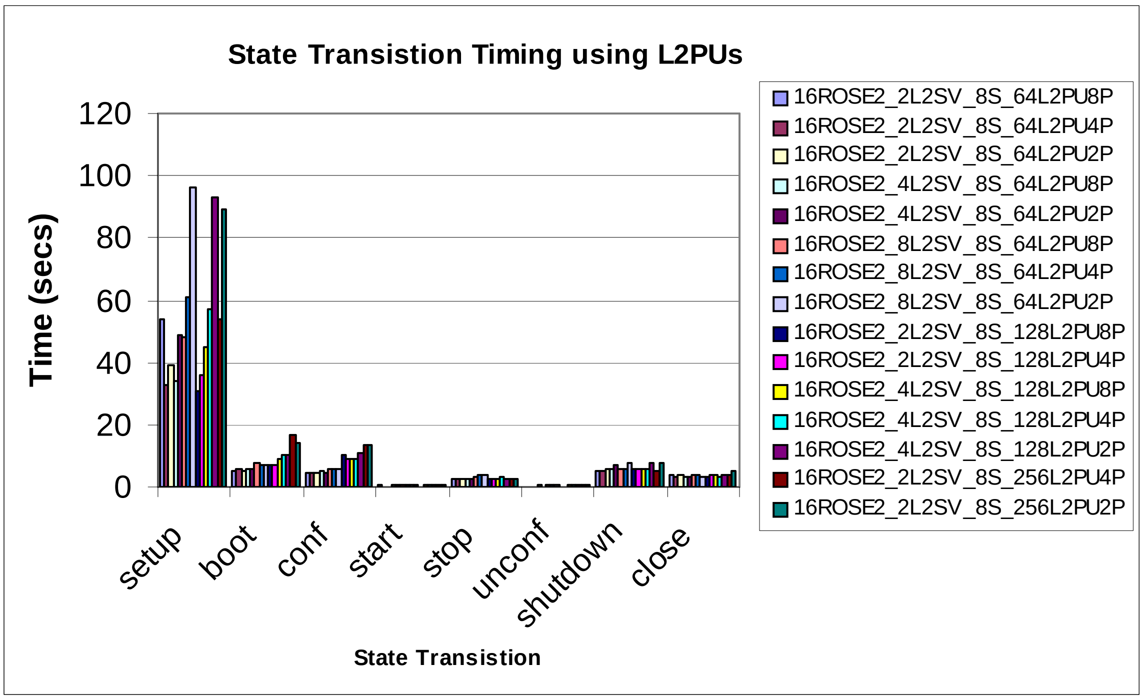


Figure 16 State transistion times obtained with various LVL2 partitions running L2PU applications

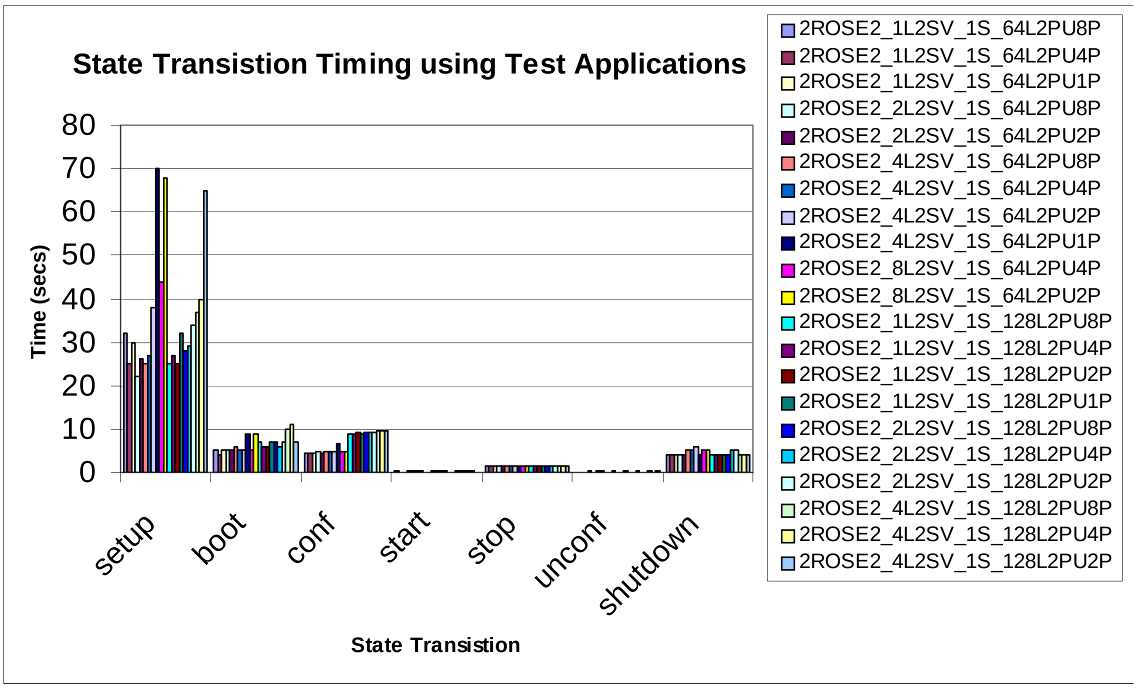


Figure 17 State transistion times obtained with various LVL2 partitions running test applications.

The plots show that most problems occur in the setup and boot phase. These problems increase with the size of a system. Partitions exceeding ~ 700 nodes failed systemtically.

Looking into more detail, tests using L2PUs were successful in 4/10 cases; using test applications the this number is 17/24, resulting in a total of 13 unsuccessful runs out of 34. The causes for failure are distributed as follows:

- 7/13 errors during SETUP (CORBA system exception)
- 2/13 errors during BOOT (Run Control waiting to be in idle state)
- 3/13 errors during CONFIG (Run Control timeout)
- 0 errors during START and STOP
- 1 error during UNCONFIG (Run Control timeout)

. .

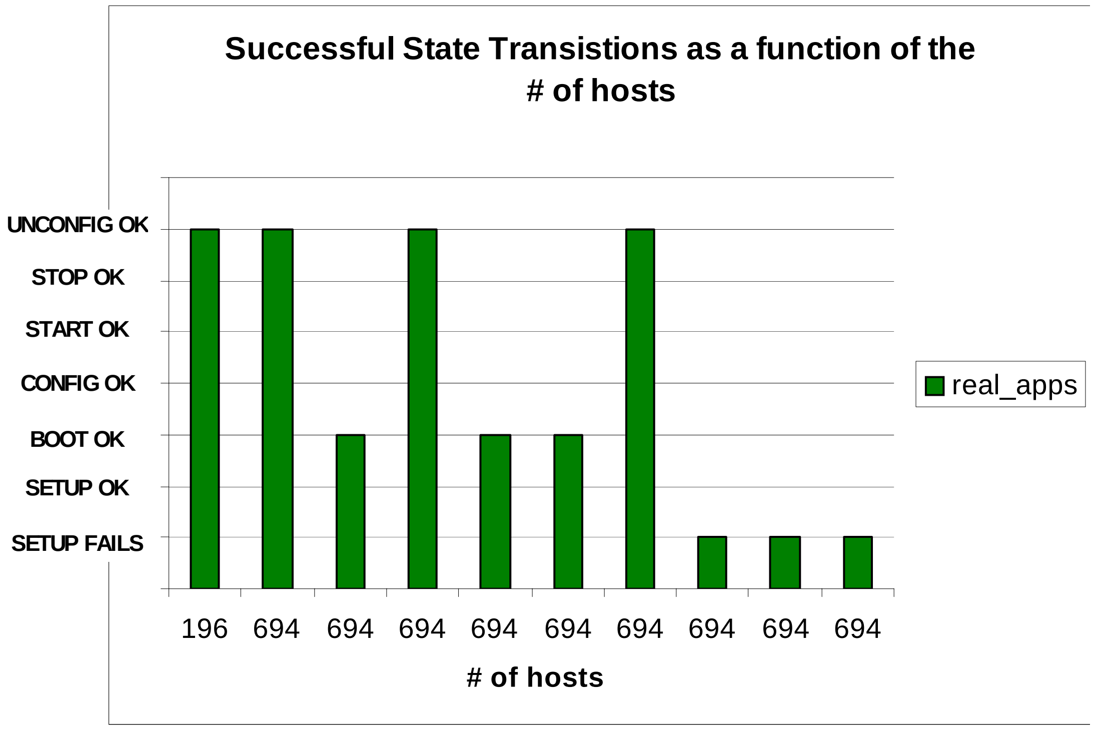


Figure 18 Sucess/Failure of state transistions vs. # nodes using L2PU applications

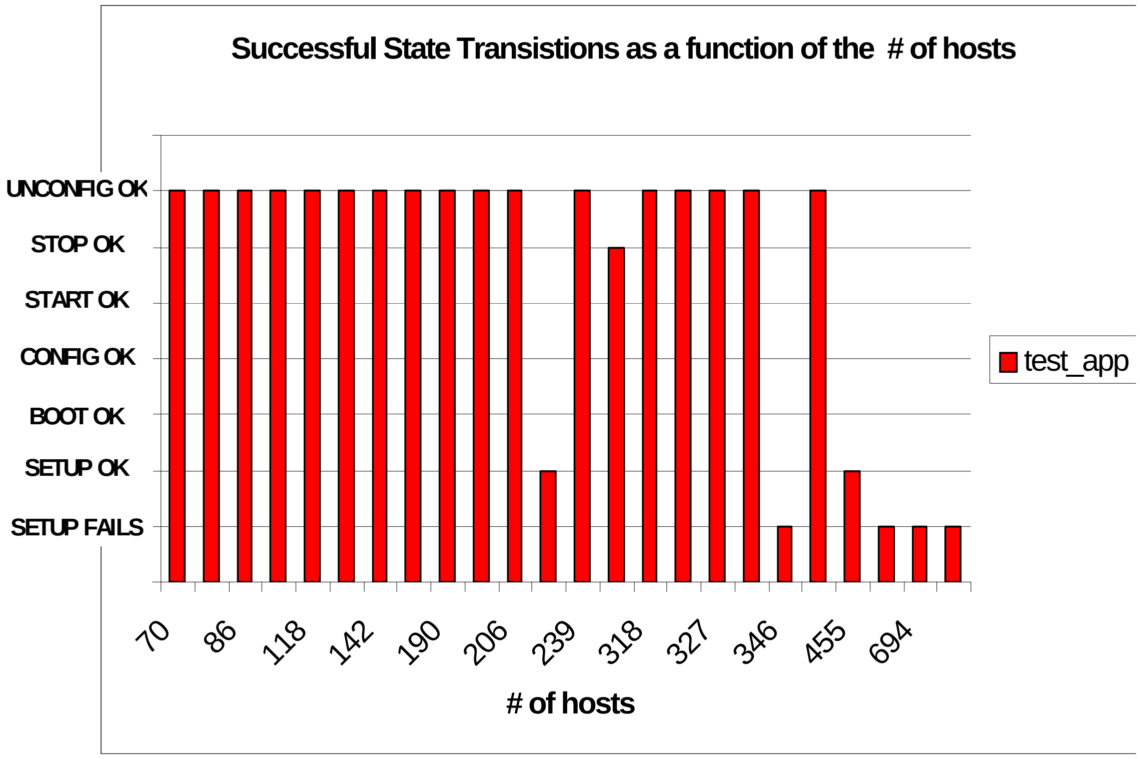


Figure 19 Sucess/Failure of state transistions vs. # nodes using test applications.

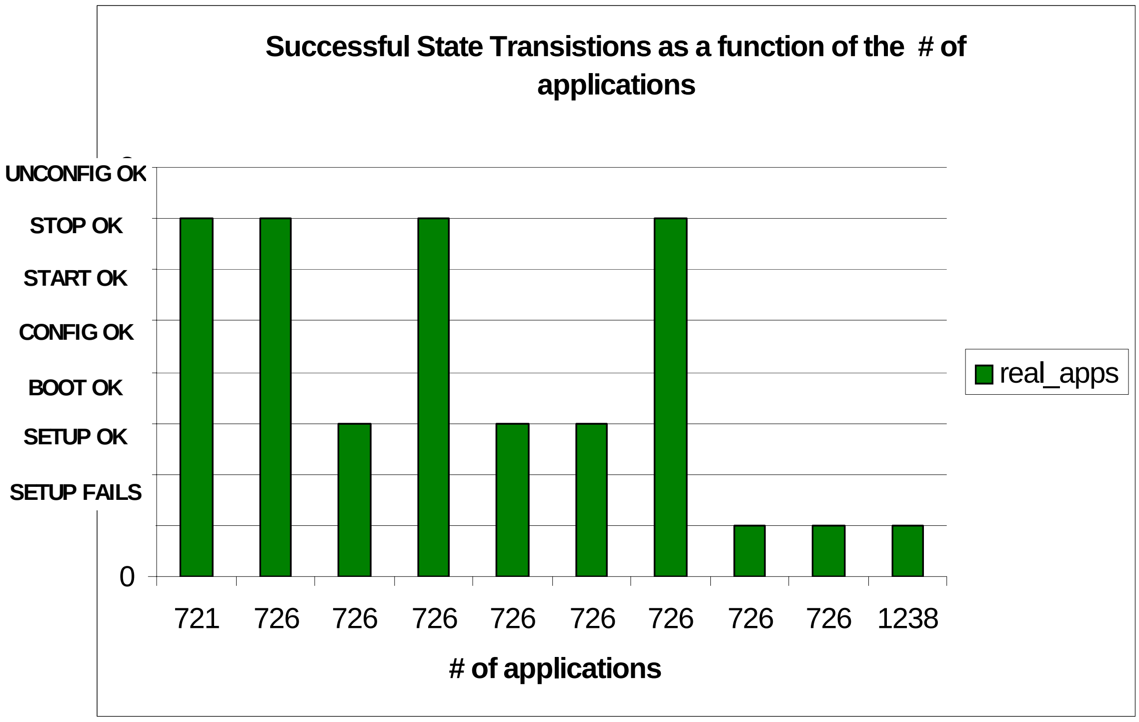


Figure 20 Sucess/Failure of state transistions vs. # applications using L2PU applications.

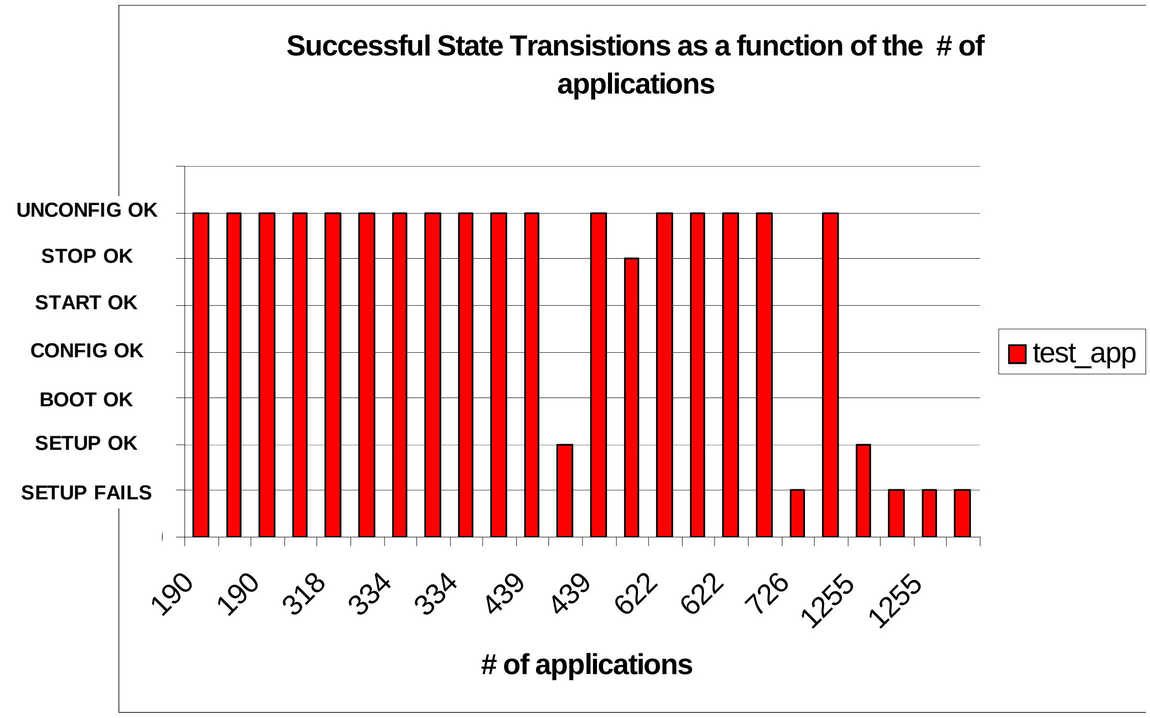


Figure 21 Sucess/Failure of state transistions vs. # applications using test applications.

5.5 Summary and Conclusions

For smaller systems, the timing results for cycling through the run control states, are satisfactory. *Unfortunately, even with the latests patches, large configurations (> 700 nodes) with 'dummy' algorithms and transfering no event data, mostly fail; this is even the case for test applications.*

The tools for generating large configurations for the release used for the tests, are largely insufficient.Tools for error diagnostics are not good enough to handle information generated by these large systems (we had to resort to grepping through error logs on 700 nodes, collecting logs with scripts and reading these with the help of editors); error messages are incomprhensible of not misleading; the sheer volume of messages caused sometimes crashes.

Though the IS provided reliably reasonable monitoring information, a means of 'integrating' results obtained from individual applications and improved presentation (e.g. ordering the information according to the hosts on which applications run) would be helpful. Information provided by the PMG was not helpful, as it did not allow easy identification of applications nor the nodes on which they run. It was frequently necessary to 'clean up' machines (abort old or hanging programs) for which PMG did not provide any facilities.

5.6 Future Steps

Though the result is maybe not what we hoped for, the LST have been very good in assessing where we stand from the point of view of running on a number of machines aproaching final ATLAs (~ 500 L2PUS). The following steps should be considered for the future:

- obviously, make it possible to run reliably large applications
- improved procedures to generate partitions (urgent and useful beyond LST)
- scripts for running these and collecting results/errors/logs
- obviously better and more diagnostics and monitoring
- proven scalability of run control
- a larger test cluster is needed at CERN or elsewhere for additional tests to avoid delaying such work during the deployment

6 Event Filter Tests without Algorithms

6.1 Aims

The Event Filter (EF) as the third level trigger of the Atlas TDAQ has several unique features with respect to other TDAQ sub-systems: access to full event data, relatively less stringent requirement on latency and thus, ability to use high-complexity event reconstruction and analysis algorithms of the Offline Software. These features, together with design and scale of EF Processing Farms, up to 1000 (in final Atlas design) Processing Nodes running some Processing Tasks (PTs) with Offline event selection algorithms each, make the Event Filter in many aspects one of the most resource-consuming part of the Atlas TDAQ System.

Owing that only a single network was available (while in final Atlas design the data network will be separated from the control network), the aims of the tests are not related to throughput performance. The focus is the evaluation of possible scalability problems mainly related to the interaction with the online infrastructure (simultaneous access to configuration DB, transition time scaling, etc) and the definition of benchmarks for the analogous EF tests with real algorithms (discussed in the Section 8.3).

6.2 Approach and execution of the EF Tests without Algorithms

These tests have been done running the EF system in stand alone mode, without using upstream TDAQ components (Level 2 and Event Building). The SFI and SFO are emulated, the former reads events from disk datafile (or generates dummy ones), while the SFO do not perform disk writing operation.

The scalability behavior has been measured using a fixed EF SubFarm size (20 hosts) and changing the number of SubFarms. The chosen size is the biggest one compatible with the rack size and the performance of the LocalController; each EF node runs one EFD and two dummy PTs (unless stated explicitly), thus every LocalController has to manage 60 applications.

6.3 Access to ConfDB: optimization of finding PT configuration objects

When we started testing EF partitions on relatively large scale (>>10 EF nodes), we observed a suspicious non-linear growth of the configuration transition times with partition size. After more detailed investigation we found that the reason for this was non-optimal mechanism of finding a configuration object in ConfDB corresponding to particular instance of PT application. A n-th instance of PT application was reading from DB and scanning the whole list of all existing configuration objects for all PT applications and needed to parse (n-1) objects until it finds n-th one. Thus, the configuration time for PT-200 was much longer than for PT-1 or PT-10. In order to solve this problem, the search mechanism was changed to using SQL query in the ConfDB, when the search was done much faster at the DB side, and PT application received only one object from DB corresponding to particular instance PT-n. This correction was installed as patch to the used TDAQ release tdaq-01-02-00 and is included in all forthcoming TDAQ releases. Similar behavior was observed with all DataCollectin applications

(ROSE, DFM, SFI, SFO, L2PU, L2SV), which used the same base "sysmonapps" library to find objects in ConfDB, and the correction applied to "sysmonapps" improved ConfDB reading for all of them. On the other side, the DataFlow applications, for example EFD, did not use "sysmonapps" and thus, did not have such problem.

Table 5 presents TDAQ transition times measured with EF-standalone partitions before and after solving this problem. The measurements were repeated for each partition from 7 to 10 times for running with non-optimal search and from 3 to 6 times for the corrected search mechanism, and the results were averaged over the same measurements (note that this statistics is obviously very small and possible fluctuations should be kept in mind).

Table 5 Non-linear scaling of configuration transition time with non-optimal mechanism of finding PT configuration objects in ConfDB. Comparison of time before and after optimization fix.

Partition	#EF hosts	# EF proc	setup	boot	shut down	cold start	cold stop	config	unconfig	Total time
Athena HelloWorld before correcting the search in ConfDB										
01x20x1EFD+2PT	34	60	22	4,4	6,0	15,9	27,9	11,1	6,3	78,3
05x20x1EFD+2PT	126	300	35	6,2	6,6	37,8	29,1	30,9	6,7	114,9
10x20x1EFD+2PT	241	600	67	8,1	7,3	66,9	34,8	56,8	8,8	183,1
Athena HelloWorld after correcting the search in ConfDB										
01x20x1EFD+2PT	34	60	22	4	6	14	28	9,6	6,2	77
05x20x1EFD+2PT	126	300	33	6	7	17	29	9,9	6,2	93
10x20x1EFD+2PT	241	600	110	7	8	21	31	10,1	7,4	176

After the mechanism of finding the PT configuration objects in ConfDB was optimized, the configuration transition time scales linearly with the size of EF partition. All the other measurements for Event Filter partition both with and without algorithms described later have been performed with the corrected search mechanism.

6.4 Transition times

Figure 22 illustrates behavior of various transition times as a function of the number of EF applications. The setup transition, not showed in the figure, is heavily influenced by the system size. But it is related to TDAQ infrastructure and is discussed elsewhere in this document.

The transitions, which do not trigger operations that entail access to shared resources, are not depended (in first approximation) to the system size. For example the Start transition entails a single very fast operation (the opening of the EFD input barrier) and therefore the transition delay is only the time needed to propagate the command in the control tree and collecting back the acknowledge. The "Stop" transition implies more complex operations (empty of the EFD buffers, wait for PT processing, etc.) and is slower. The actual constant value (about 15 seconds) is artificial and is related to a internal timeout. This has been removed in the new releases and the transition is now faster, but it should not be

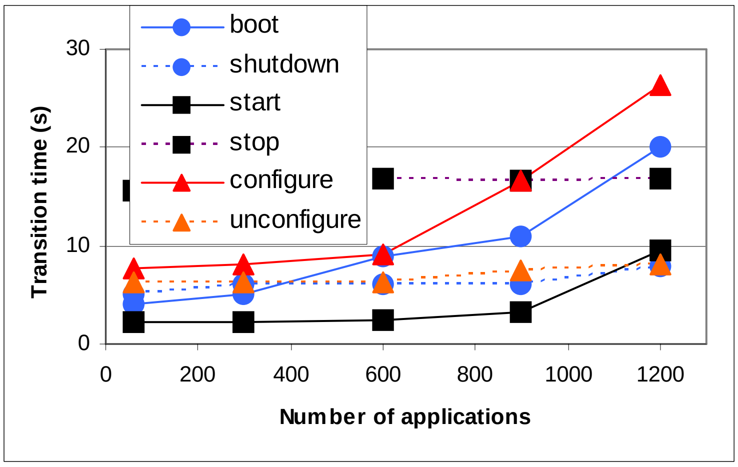


Figure 22 Transition times as a function of the EF partition size.

influenced by the system size. It is heavily dependent by the EFD buffer occupancy at the moment when the transition command is received.

Also the Shutdown transition time seems to be flat, while the Boot transition time is proportional to the number of applications. The dependence is probably due to the access to binaries and libraries in the shared file system.

Access to a common resource, the ConfDB server, occurs also for the Configure transition, and indeed the transition time is system size dependent. The function seems to be not strictly linear but it looks reasonable in the sense of being affected by only one obvious factor: number of EF applications.

Since Event Filter with dummy Steering (without algorithms) was used also for studies of the integrated TDAQ partitions (with ROSes, Level-2 and Event Building), more information about Event Filter behavior at large scales can be found also in corresponding parts of the section X of this document.

6.5 Summary and Conclusions

Since the DataFlow part of the Event Filter, i.e. the version with dummy Steering Controller PTdummy, is well developed over the last several years, the behavior of the Event Filter without Athena algorithms did not show any crucial problems which could block the overall TDAQ running.

However, several tests and investigations on large scales are important to do using Event Filter in dummy Steering mode (i.e. without algorithms), for example:

- improvement of error handling and diagnostics;
- stability tests and improvement of robustness;
- further optimization of HLT configuration schema by introducing templates in OKS;
- development of special tools/scripts for managing TDAQ/HLT partitions and computer clusters on large scale (checking computers, disks, processes left running, etc.)

7 Integrated DAQ/HLT-I Scalability Tests

7.1 Aims

The complete Trigger and DataFlow system in ATLAS is foreseen to involve about 2000 PCs. Thus, the availability of about 700 PCs for this year's Large Scale Tests, allowed to construct partitions with *systems up to about 30% of the full ATLAS scale.*

The main focus was on the functionality, operability, performance and scalability of the control aspects of an integrated system, because it was understood that the heterogeneous network topology and PCs available to us would not be suitable to optimize and test the data-acquisition performance of our system. In particular we wanted to see how well we can control the system in completing the various transitions of the Finite State Machine of the Run Control sequence. In more general perspective, building and operating large partitions was a way to reveal problems which show up rarely in a small scale. Identifying and solving such problems was a primary goal of the integrated tests and a crucial step towards operating the full scale ATLAS system.

In addition, these tests were a great opportunity to improve the tools for generating automatically and operating big partitions and train people for the work awaiting the ATLAS pre-series TDAQ system, installed at Point 1with realistic network topology and PCs.

7.2 Test description and Configurations

The partitions we built used all components of a full TDAQ system, trying to share the available nodes in quasi-realistic proportions among the various TDAQ applications. We created the partitions using the DBgeneration scripts, like the Event Building and Event Filter partitions[1]. The control structure is kept like the one used for the Event Builder tests (Section 3.2), with the addiction of a LVL2 and an Event Filter sector. For simplicity we did not make use of the ability of the DBGeneration scripts to create a sub-farm structure. Our configurations were made by randomly assigning the following applications among the available PCs:

i. 5 to 55 ROSs, each hosting 12 Read Out Buffers (ROBs) of 1 kByte of "data", organised in farms of 5 ROSs.
ii. 1 Data Flow Manager (DFM), triggered by the L2SVs with a very low rate, because we want to test the control aspects of our partitions.
iii. 3 to 35 SFIs nodes, organised in farms of mostly 6 SFIs.
iv. 10 to 110 L2PUs, put in farms of 5 L2PUs each.
v. 2 to 22 L2SVs, one for each L2PU farm.
vi. 30 to 350 EFDs, organized in farms of 10 EFDs each, with a unique SFI serving each farm.

1. The use of all the TDAQ applications in partitions of ever increasing sizes dictated modifications to the DBGeneration scripts. This feedback is reflected to the new TDAQ releases and is a valuable heritage for work done at the pre-series and other test beds.

vii. 1 to 4 emulated SFOs

viii. 10 nodes where reserved to host online services. e.g., Information Service (IS), Inter Process Communication (IPC), etc.

ix. a varying number of nodes hosting controllers (one per farm of ROS, SFI, L2PU and Event Filter nodes)

x. Both the L2PU and the EFDs were running one dummy algorithm each.

7.3 Test Approach and Execution of the Tests

As for all the other tests, we used the tdaq-01-02-00 release of the online software. We report timing measurements obtained with the same method as described briefly in the Event Building chapter (Section 3.3).

7.4 Results

In Table 1 we show the configuration of the partitions we used for the measurements. In particular we show the total number of hosts, the number of ROS, SFI, L2PU, EFD and emulated SFO nodes. In parentheses we see the number of farms comprising the ROS, Event Builder, L2 and Event Filter sections, respectively.

Table 6 Characteristics of the partitions used for the timing measurements of Integrated partitions.

# Hosts in partition	**# ROSs**	**# SFIs**	**#L2PUs**	**#EFDs**	**#SFO emulators**	**# measurement**
68	5 (1)	3 (1)	10 (2)	30 (3)	1	3
135	10 (2)	6 (1)	20 (4)	60 (6)	1	6
255	20 (4	12 (2)	40 (8)	120 (12)	2	5
495	40 (8)	24 (4)	80(16)	240 (24)	4	4
614	50 (10)	30 (5)	100 (20)	300 (30)	4	4
674	55 (11)	33 (6)	110 (22)	330 (33)	4	3
691	55 (11)	35 (6)	110 (22)	350 (35)	4	1

The largest system we built used 701 nodes (691 plus 10 nodes for online services), which is almost all the nodes made available to us. As the system grew in size, it was more and more difficult to even bring a partition up to the INITIAL state, let alone run it. In fact, the number of measurements shown in Table 6 indicates the successful runs out of about 10 attempts in each case. The largest partitions were only run manual and we report their results here to give an idea of the instability of the system.

The failures due to timeouts and dying processes during running revealed problems in the way some applications accessed the database during configuration and to a memory leak(?) in the Local

Controller, respectively. The problems made tests at large scales difficult, but it is also true that they were revealed, only because we were using large systems!

In Figures 3 and 4, we see the average transition times in the startup (Setup-Boot-Configure-Warm Start) and the shutdown (Warm Stop - Unconfigure - Shutdown - Close) sequences, respectively. The transitions are defined in Section 3.3, and the characteristics of the partitions are defined in Table 6-1

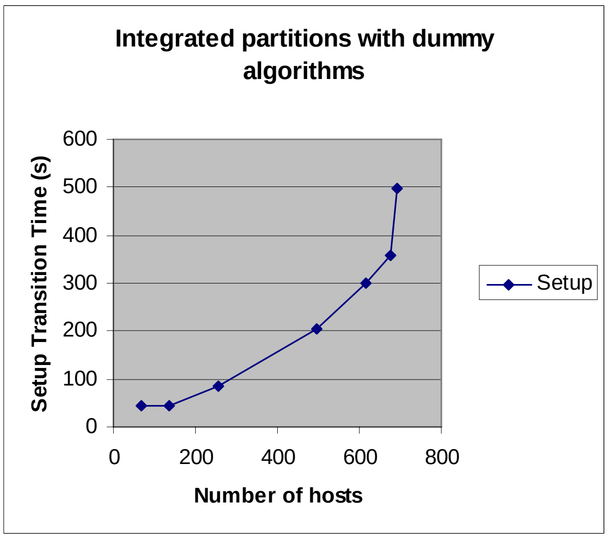


Figure 23 Average times (in seconds) to setup a partition, as a function of the partition size.

Apart from the evident fluctuations of the largest partitions we operated, we conclude that, in general, the transition times are acceptable, with the exception of the time needed for setting up the partition.The steps performed during the setup procedure consisted of the sequential execution of many checks, all launched as different binaries from the same host machine. This resulted in increasing times to complete the setup procedure, as the total number of hosts in a partition increase. This realization lead to changes in the way the setup is done, but not in time for the LST05.

With the tdaq-01-02-00 implementation of the setup procedure we need about 0.5 s per host, which is about 10 times slower than the next bigger contribution, the time to configure the system. The rest of the transition times are small, while the times for the stopping transitions are also constant with the partition size.

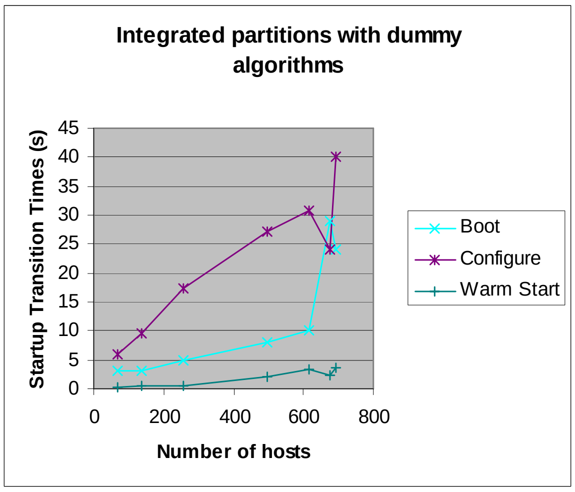


Figure 24 Average Times (in seconds) for completing the startup transitions for partitions described in Table 6.

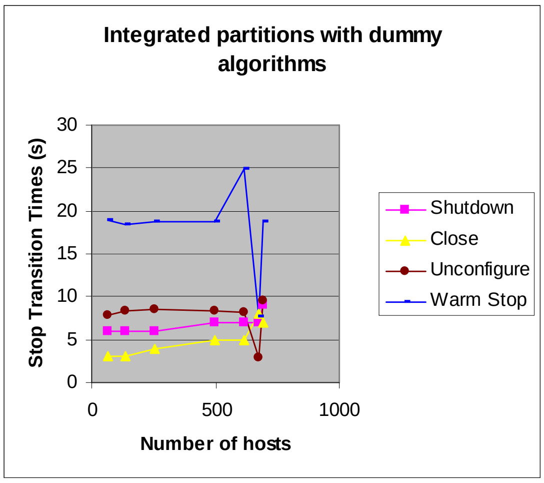


Figure 25 Average times (in seconds) to complete the stopping sequence of transitions for partitions described in Table 6.

7.5 Summary and Conclusions

We have tested and operated integrated partitions up to about 30% of the full ATLAS system. We had increasing difficulties to operate larger and larger systems, due to problems revealing themselves clearly at large scales only. Actions had been taken to fix them and continued. This was in fact a principal goal of such tests. With the improvements expected on the setup procedure, we conclude that the transition times are not beyond control, even though we have to mention that a system of 2000 PCs would need about 90 seconds to get configured if the pattern in Figure 2 holds.

Apart for bug fixing, the work on LST05 provided a valuable test-bed for training for the pre-series and for enhancing and developing tools to build and operate large partitions.

7.6 Future Steps

Future tests with larger testbeds are needed to test the solutions and bug fixes generated because of LST05, and also to find the next problems, and thus be ready for the full ATLAS system.

8 HLT Software Tests with Trigger Algorithms

The July 2005 Large Scale tests were the first opportunity for the ATLAS High Level Trigger to exercise the trigger software with real trigger algorithms on several hundred Linux machines. For this purpose methods needed to be developed to support the deployment and usage of all necessary software components. The following subchapters describe these methods, the tests performed and the results obtained for the Level 2 trigger and the Event Filter.

8.1 HLT software distribution

8.1.1 Components of the HLT trigger software

The ATLAS High Level Trigger is implemented as a pure software trigger. Events that have been accepted by the Level 1 hardware trigger are analyzed by Level 2 Processing Units (L2PU), which use the data contained in the Region of Interest (RoI) to decide whether the events should be kept. If so, these events undergo full event building and are finally analyzed again by the Event Filter Processing Task (EFPT), which has the full event data available to make the final trigger decision.

L2PU and EFPT share a considerable amount of software infrastructure. Both use large parts of the ATLAS Offline Software **Athena** to handle the decoding of raw data (ByteStream) and to support the various trigger algorithms that are to run in the Level 2 Trigger or the Event Filter. The trigger algorithms as well as the many utility packages of the Offline software used by them and by the HLT software have to be initialized, which is currently done via a Python program interface (jobOptions.py files).

All this means, that the set of software components that are required for testing the HLT algorithms in the online environment includes all of the following:

- the TDAQ software (tdaq-01-02-00)
- the ATLAS Offline Software (Athena 10.0.4)
- the High Level Trigger software (HLT 2.0.2)
- all the external software used by the above

It should be noted that all other parts of the Large Scale Tests besides the HLT tests with algorithms need and use just the TDAQ software and the externals needed by it.

In addition to the above software components, the HLT tests with algorithms also need appropriate test data, which have to be provided as well.

8.1.2 The HLT image concept

From previous experience with using the HLT and Offline software in the 2004 ATLAS Test Beam and on various smaller software testbeds, it was realized that there was an urgent need to tightly control the versions of all packages in all the software used. In particular it was found necessary to prevent situations where individual software packages (shared libraries) could be updated concurrently to their use in the tests. In the LXBATCH cluster, the only shared file system available was AFS, and AFS is not foreseen to be used in the actual ATLAS data taking system. It was thus decided to install all the software needed for the HLT algorithm tests on a local disk on each of the cluster machines separately.

To make this task manageable, we considered it undesirable and nearly impossible to separately perform a complete installation of the very complex ATLAS Offline Software on each of up to 700 cluster hosts. It seemed unlikely that we would be able to efficiently detect and correct any potential problem that might come up during the software installation, e.g. space running short on a few cluster machines. Instead, we devised a method that allowed all the software installation, verification and functional testing to be done once, on a separate machine (or set of machines), before distributing the result as a single large file to the test cluster.

This single large file, which we call the “HLT image”, starts as an empty 6 GB container file, into which we install a Linux ext3 file system. Once the ext3 file system is available (mounted) on the build machine, it can be populated by the normal procedures of installing our software. First, we needed to make a convention that declared that all software in the image needed to be installed below a common directory (mount point) called **/mnt/atlas**. Below this mount point, we then installed the **tdaq-01-02-00** release, the **Athena 10.0.4** and the **HLT 2.0.2** releases, including external software used by these. In addition to these software releases, we also added a number of data files to be used on the HLT tests.

After some testing on the build machine and some other testbeds, the image was considered finished. The 6 GB container file was compressed using gzip, resulting in a compressed file size of about 2.1 GB. Of the 6 GB space reserved in the original ext3 file system, ca 5.2 GB was taken up by the software and data files. The rest was kept as spare space, to be used e.g. for emergency updates.

The normal usage of the compressed HLT image are the following steps, taken on each cluster node:

- copy the image to the target machine(s)
- uncompress the image on the target machine(s)
- mount the 6 GB image file as an ext3 file system on mount point **/mnt/atlas** in **readonly** mode.
- set up a run time environment by **source**ing a setup script contained in **/mnt/at/as/setup**.
- start running tests

Note that in the case of the above mentioned emergency update, the image file system has to be re-mounted in read/write mode. We did not have to use this facility during the Large Scale Tests.

We should also emphasize, that due to our requirement of well-controlled versions of all software in the HLT image, we had to accept that there was more than one version of the TDAQ software installed on the cluster nodes. The one installed with the HLT image, and the one used for the tests without HLT algorithms. That version in fact received a number of patches in the course of the LST, while the HLT image was kept constant over longer times. We produced just 2 versions of the HLT image during the LST time, and we only applied patches, if any, to the TDAQ part when making a new HLT image.

On the machines of the LXBATCH cluster, all participants of the LST used a local file system called **/pool**, which also contained the directory where the compressed HLT image was to be put (some subdirectory in **/pool/hlt**). However, on a subset of the cluster, the total size of **/pool** did not provide enough space to also contain the uncompressed image - at least 8.1 GB were needed during uncompressing. These machines, about 100 in number, were not used for the HLT algorithm tests.

8.1.3 HLT software distribution method

The next issue to be solved was that of distributing a 2.1 GB file (the compressed HLT image) to up to 700 cluster nodes in an efficient manner. The IT group in charge of the LXBATCH cluster advocated using the Quattor system for distributing the software. This uses rpm files, which get ditributed in a 2-stage process, first from a main location to a number of file servers, and then from each file server to the cluster nodes. In both steps the transfers procede in parallel. This method was chosen by the TDAQ group for distributing the TDAQ software.

Because of the very large file size of the HLT image, which makes producing rpm files very awkward, and because we already had explored it on smaller clusters, we decided to use publicly available open Peer-to-Peer software. The Peer-to-Peer software chosen was **BitTorrent v3.4.2**. This choice was made for the following reasons:

- ease of availability and use of **BitTorrent**, which is only about 400 kB of Python scripts
- its capability of adaptively optimizing the bandwidth use of arbitrary networks
- pure curiosity, to see just how well the BitTorrent protocol performed on large clusters

The last point was important to us, as the final size of the ATLAS HLT trigger might contain well over 2000 cluster nodes.

Briefly, the **BitTorrent** protocol shares a file in the following way:

- one or more hosts (“seeders”) start serving the complete file
- meta-information about the file is encoded in a **.torrent** file. This is ususally a fairly small binary file.
- when a client (“leecher”) wants to download the file, it first needs to get hold of the **.torrent** file
- the **.torrent** files are usually served from a web page, but any normal file transfer method such as ftp, scp or e-mail attachments works just as well
- files are distributed in **pieces** (in our case of 256 kB size)
- a central place, the **tracker**, keeps track of which **client** has which **pieces**.
- every **client** that receives a **piece** also becomes a **server** for all the **pieces** it already has. This is completely symmetrical, hence the name **peer** is also used for the clients, and gives rise to the name **Peer-to-Peer** software.
- the **peers** offer to upload their **pieces** to other **peers**, but only if those are uploading in turn at a sufficient rate to others
- BitTorrent has a number of **policies** to ensure that the process starts and procedes in a reasonably fair and efficient manner. In particular, the policy of favoring peers that are “good uploaders” over those that are not, ensures that the process will adapt automatically to find the best network paths, provided there are enough peers participating.

- **fault tolerance**, e.g. retransmission of pieces or resumption of file transfer, is built into the **BitTorrent** protocol
- there is also a built-in **throttling** of the transfer rate, each **peer** selects a maximum transfer rate at which it is willing to upload. This rate is selectable at the time the **peer** gets started.

In order to control and monitor the running of hundreds of BitTorrent peers, a set of wrapper scripts was developed which permiitted to start, synchronize, check and stop all the peer processes, the seeder and the tracker. An important function of these was an automatic integrity check using the MD5 checksum of the image file.

In all cases, we used exactly one seeder to start the download. The monitoring allowed to determine automatically the time when the last peer had finished its download. The interval from the start of the download process to this moment is the total transfer time.

8.1.4 HLT software distribution measurements

The goal was to measure the total transfer time as a function of the number of cluster nodes, for a few different sizes of the image file. We used two versions of the HLT image, one with a file size of 1776 MB (during the first half of the LST) and another with a size of 2173 MB. For comparison, some runs with a smaller file of 100 MB were also done.

It has to be emphasized that the network topology connecting the cluster nodes at the various stages of the LST was not very well known to us. Most nodes were connected by 100 Mbit/s Ethernet, a fraction of the machines were connected by 1 Gbit/sec Ethernet. In addition, it was not guaranteed that our tests were the only activities generating network traffic, so that our result cannot be taken as determining a well defined function of cluster size. Rather, the times obtained and the transfer rates derived from them, should be considered as somewhat typical cases. The effect of randomly occurring extraneous network traffic is of course always to slow down our transfers and increase the transfer time. Indeed, for the 2173 MB file size, the longest transfer time was not observed on the largest cluster size, but for one of about half the size, as shown in the following table nr.7-1.

Table 7 Transfer times and rates

nr. of cluster nodes	**total transfer time [sec]**	**effective transfer rate [Mbits/sec]**	**avg. transfer rate / node [Mbits/sec]**
187	3490	916	4.9
239	2334	1749	7.3
312	4980	1072	3.4
500	2618	3268	6.5
597	2650	3814	6.5

Here the **“effective transfer rate”** is computed as **(number of nodes) * (file size in bytes) * 8 / (total transfer time in sec)**. The **“average transfer rate per node”** is defined as **(file size in bytes) * 8 / (total transfer time in sec)**. The two quantities are also plotted as a function of the number of nodes in Figure 26 and Figure 27.

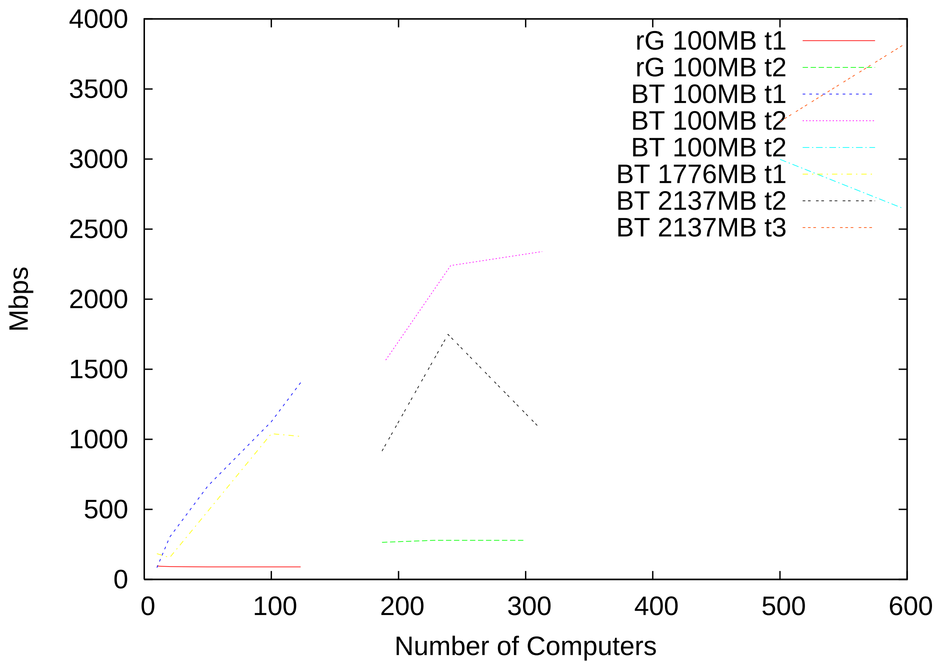


Figure 26 Effective transfer rate (Mbits/sec)

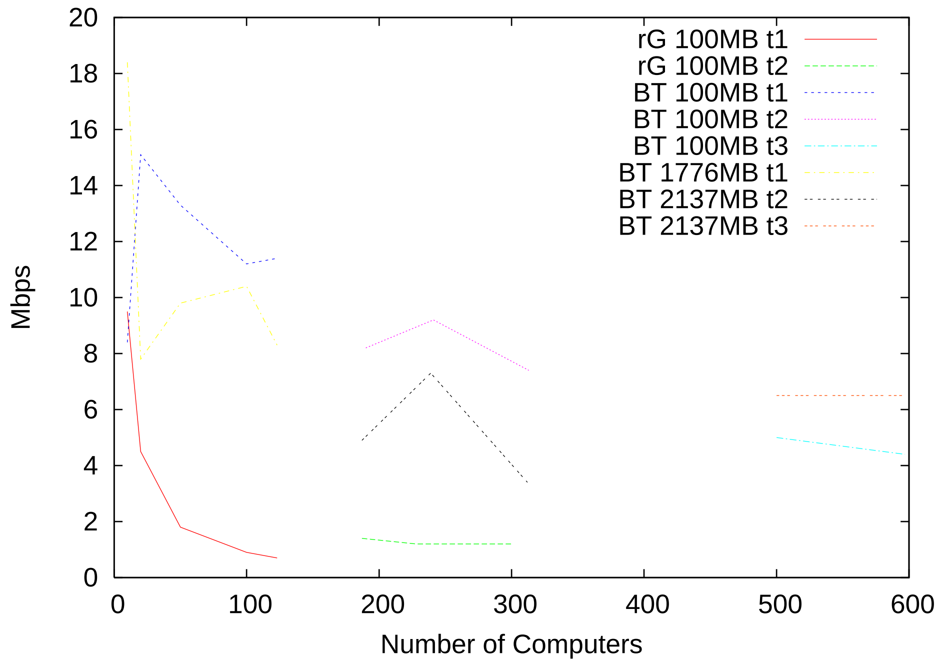


Figure 27 Transfer rate per node (Mbits/sec)

8.1.5 Software Distribution: Summary and Conclusion

During the tests with the **BitTorrent** distribution method, we encountered a number of cluster hosts that could not be used in our tests. Out of the ca. 700 machines that were available at the end of the Large Scale Tests, 107 did not have a large enough local disk area to fit our HLT image and had to be removed from the list of machines we used. A further few developed hardware and/or software problems (ssh access) that made them unusable as well. This explains the sometimes odd numbers of nodes used in various test sizes, such as 597.

The main question we had with regard to **BitTorrent** was wether this method was indeed flexible enough to do the software distribution with acceptable performance and reliability. It had to work on a network of more or less unknown topology and with unknown background traffic. Our results show that this was indeed achieved. It was able to download a 2 GB file to 600 hosts in less than 45 minutes, although we also saw the effective transfer rate vary by about a factor of 2. We should also note that relatively little effort was spent optimising the many parameters that govern **BitTorrent**. Essentially we just removed the limit on the upload rate, and set the **piece** size to a fairly high value (256 kB).

Our conclusion is that the **BitTorrent** method is a viable tool for distributing large image files to a large number of machines. It is obvious that in a situation where the network topology is exactly known, and where the competing traffic can be controlled, one can always devise distribution methods that will perform better. However **BitTorrent** is an easy to use and quick to deploy tool, which can be regarded as a fallback solution. Hence the existence of **BitTorrent** and similar **Peer-to-Peer** mechanisms guarantees that off-the-shelf tools can be used if one wishes to save the effort of developing special purpose tools.

We should like to express our gratitude to the networking group of the CERN IT division for accepting our use of the the BitTorrent protocol within the CERN domain, as the use of such tools is normally prohibited by the Computing Rules to prevent abuse of the CERN bandwidth.

8.2 Level 2 Tests with Algorithms

With the task of distributing the HLT software image solved, the tools were in place for running tests of the HLT with trigger algorithms, both **Level 2** and **Event Filter**. In this chapter we describe the aims, procedures and results of the **Level 2** tests.

8.2.1 Goals and Non-Goals for the Level 2 Algorithm Tests

The Level 2 group in charge of integrating the trigger algorithms with the pure data flow software framework (which uses a dummy trigger algorithm) joined the Large Scale Test on quite a short notice. It was therefore decided to be very cautious and attempt to test only a small number of trigger algorithms that had already seen substantial testing in other testbeds.

The short list of algorithms was:

- **HelloWorld**, which just demonstrates the **initialization** and **event loop** facility (in its HLT incarnation) of **Athena**.
- **TrigT1Decode** algorithm, which does **ByteStream** decoding and unpacks the **L1Result**.

- **muFast**, which decodes the **ByteStream**, unpacks the **L1Result** to obtain the **ROI** information, then uses this to request the data in the **ROI** from **ROS** via the **RobDataProvider** class. With these data **muFast** then tries to find and fit muon tracks and makes a trigger decision.

We note that of these, only **muFast** currently needs to access the **Conditions DataBase**. It has to obtain the detector geometry for the muon system, in order for **muFast** to be able to recognize and reconstruct muon tracks. All three algorithms are first initialized by means of Python jobOptions.py files, as described earlier in section 7.1.1.

The **main goal** of testing these algorithms in the Large Scale Test cluster was to verify that **no** fundamental **show stopper** had appeared compared to the small scale tests that were done before. In particular, the fact that all L2PUs share the access to the **Conditions DataBase** when running **muFast**, meant there was a potential for contention and worse problems at **Configuration** time. This happens near simultaneously on all L2PUs when the Run Control sends the **Configure** command to the L2PUs. Other relevant Run Control commands of interest to us were the **PrepareForRun**, which happens just before the real start of a run and allows for a few very quick actions to happen (such as applying new **Run Parameters**), and the **Unconfigure** command, which is used after the end of a run.

Essentially, we wanted to see how large we could make the number of L2PUs in one partition, before things would start to fail, during access to the **Conditions DataBase** or elsewhere. This Large Scale Test was the very first time that many L2PUs were run simultaneously in this mode.

It was however explicitely **not** our **goal** to obtain detailed performance measurements. This will need to wait for other Large Scale Tests where we have better control over the network topology and network traffic. Nevertheless, an effort was made to instrument the L2PU code with a good number of timers that allowed us to record the wall time at which various crucial points of the L2PU code were executed. These were used to collect the measurements described below.

8.2.2 Test Description

The **muFast** test described above in fact includes both of the other tests described under **HelloWorld** and **TrigT1Decode -** success of the **muFast** test implies that the other two test succeed as well. We felt that the muFast algorithm had in fact seen sufficient previous testing on small testbeds, so that we decided to start the test program directly with **muFast**. This turned out to be a good, time-saving decision as will be shown below, and we were able to drop the other two tests completely.

8.2.2.1 Partitions for running the muFast algorithm

The test program which we pursued used a number of partitions that included a medium to large number of **L2PUs**. Each **L2PU** was configured to run the **muFast** algorithm in single-threaded mode

To provide data for the **L2PUs**, each partition had the same, fixed number of **ROS** nodes, namely 8, which is in fact the number of **ROS** nodes that the muon system of the final **ATLAS** detector will have. We used the **ROSE** emulator as the **ROS** application running on these nodes.

A data file with **20,000 single muon** simulated events had been prepared, where each muon had a transverse momentum of at least 10 GeV/c. These events were processed through the detector simulation with noise and cavern background, and had the **Level 1** hardware trigger simulation applied to them, so that a **Level 1 decision** and a **L1Result** as well as **ROI** information was available on the file.

This data file was preloaded into each **ROSE** application. Each **ROSE** was configured to hold 1/8 of the readout channels that happened to contain any data. There were in fact exactly 248 such readout channels, so that each **ROSE** was configured to hold 31 of them.

Since we wanted to drive a large number of **L2PUs**, we subdivided the set of L2PUs into a number of subfarms. Each subfarm contained exactly one **Level 2 SuperVisor** (**L2SV**), which handled either 8 or 32 L2PUs. The total number of subfarms was varied in steps of 2 from 8 to 16, and in some cases beyond.

Focusing only on the Level 2 algorithm tests, we did not need to add an **Event Builder** and therefore did not configure a **pROS** either to receive the **L2Result**.

Each subfarm also contained a **LocalController** from the Run Control tree, as well as an **rdb server** which replicated the **TDAQ Configuration DataBase** (not to be confused with the **Conditions DataBase**). In addition, there was a number of intermediate level **Controllers** and **rdb servers**, as well as a specially crafted **Online Segment**, which was a group of 10 machines set aside to run the **Online Infrastructure Software**, such as **IS servers**, etc.

For most partitions, we also made a further simplification: each **L2PU node** ran exactly **1 L2PU application**. For a few of the larger configurations, we also tried to place 2 or 4 L2PUs on a single machine. See Table 8 below.

8.2.2.2 Accessing the Conditions Database

For using the **ATLAS Conditions DataBase**, the ATLAS **Database Group** had arranged a dedicated database server for us. This machine was used exclusively by the HLT algorithm tests for the duration of the Large Scale Test. Level 2 and Event Filter took turns in the use of this machine, but never accessed it simultaneously.

The machine was a **MySQL** server that served the **Detector Geometry** part of the **Conditions Database**. Access to this database server was transparent and without problems, until we exceeded a threshold of 256 **L2PUs**. At this point, apparently the database machine ran out of connections and crashed. The Database Group kindly reconfigured the server to use an increased limit of 4k connections, up from 1k. After this, there were no further problems with the MySQL server.

8.2.3 Test Execution

The sequence of actions undertaken for each partition was the following:

- use the database generation script **create_lvl2.bash** to generate a partition file (.xml file) with parameters suitable for running the muFast algorithm inside each L2PU.
- each partition file needed small number of tweaks, applied by editing the .xml file with a normal text editor. These were needed to take care of details that **create_lvl2.bash** was not designed to provide.
- we tried to start each partition until we achieved one successful run, or had to give up. No attempt was made to collect more statistics by running each partition multiple times.

- during each successful run, every L2PU wrote two files in the /tmp area of its host machine:
    - a normal log file with the usual printout from the muFast algorithm, containing some of the timing information we wanted to collect
    - a special file where the output from dedicated timing macros was collected. This was only produced because the L2PU and muFast code had been compiled with a special compiler flag turned on that enabled the above macros.
- after each run, these files were combined in a compressed tar file on each L2PU host in parallel
- finally, these tar files were collected in a central place for later evaluation

The following table lists the partitions for which we wish to report measurement results here.

Table 8 List of **partitions** to be tested with **muFast**

# of SubFarms	#L2PUs per SubFarm	total # L2PUs	# L2PUs per host	# of L2PU hosts	success?
8	8	64	1	64	yes
8	32	256	1	256	yes
10	32	320	1	320	yes
12	32	384	1	384	yes
14	32	448	1	448	yes
16	32	512	1	512	yes
32	32	1024	2	512	no
64	32	2048	4	512	no

For the partitions where the run succeeded, we give some time measurements in the following section. The two large partition with 1024 and 2048 L2PUs did not succeed due to problems within tdaq-01-02-00 that prevented very large partitions from working well.

8.2.4 Test Results

In this section, we show timing results for the partitions whose runs finished successfully. For each partition, we show two groups of plots:

- pure **DataBase timings**
    - DB **get connection** time **-** the time it takes to establish the connection
    - DB **shutdown connection** time **-** the time it takes to shut the connection down
    - DB **total time to get all records -** just for muFast access to Geom DB
    - DB **average time to get one record**
- times spent inside the **PESA Steering Controller (PSC)**, during different state transitions
    - SC **Configuration** time **-** time spent in PSC during Configuration

- SC **PrepareForRun** time **-** time spent in PSC during PrepareForRun
- SC **Unconfigure** time **-** time spent in PSC during Unconfigure

Note that the histograms do not represent multiple runs. Rather, each histogram shows a single run for one partition.Each L2PU in the partition contributes one entry of the histogram. In a few cases it was not possible to collect all the log files for each L2PU of a partition, so there may be fewer entries in the histogram than there were L2PUs in the partition. This was the case for the 16 x 32 partition, where only 487 instead of 512 entries are seen.

8.2.4.1 Pure Database Timings

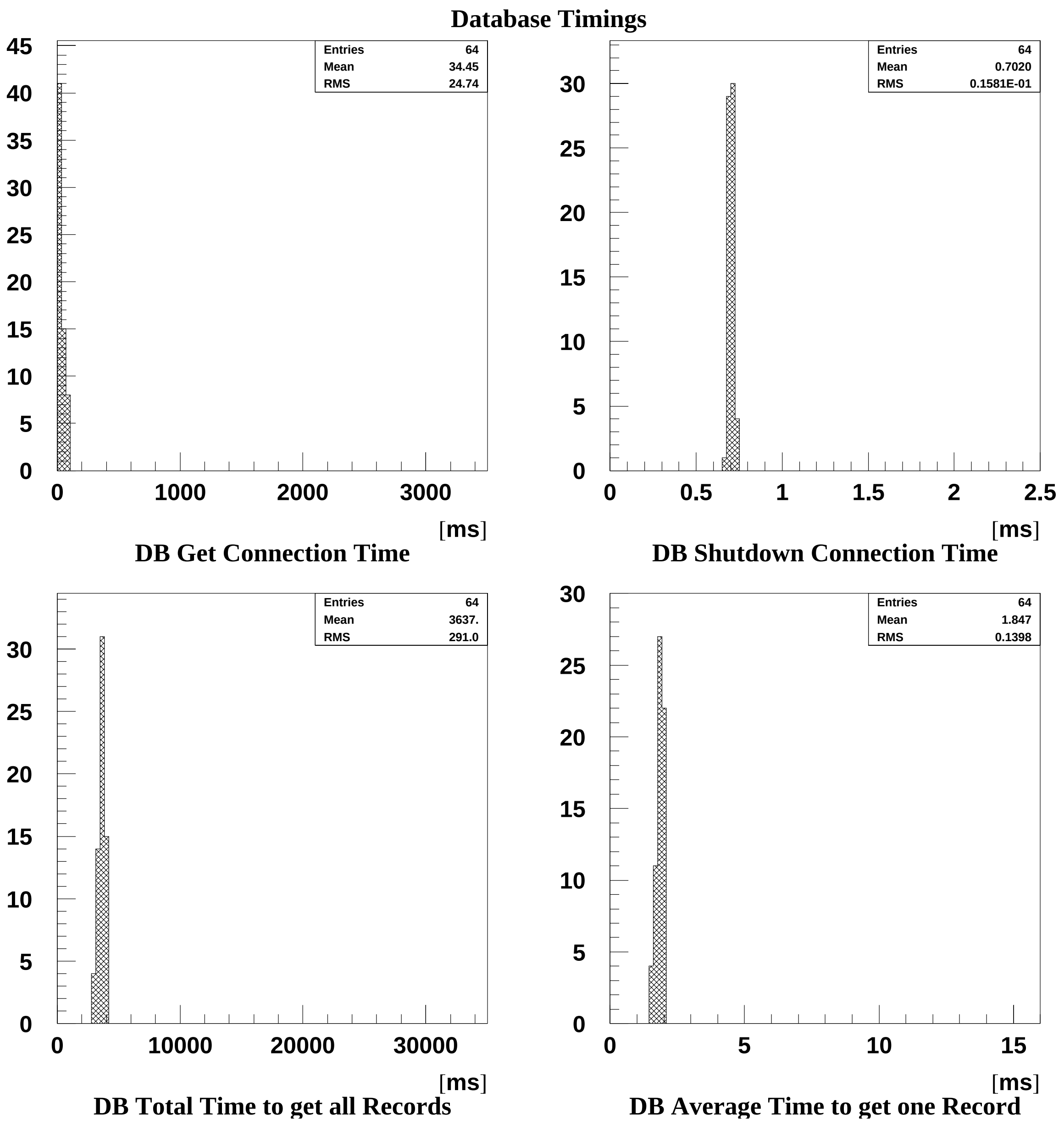


Figure 28 DB timings 8 x 8

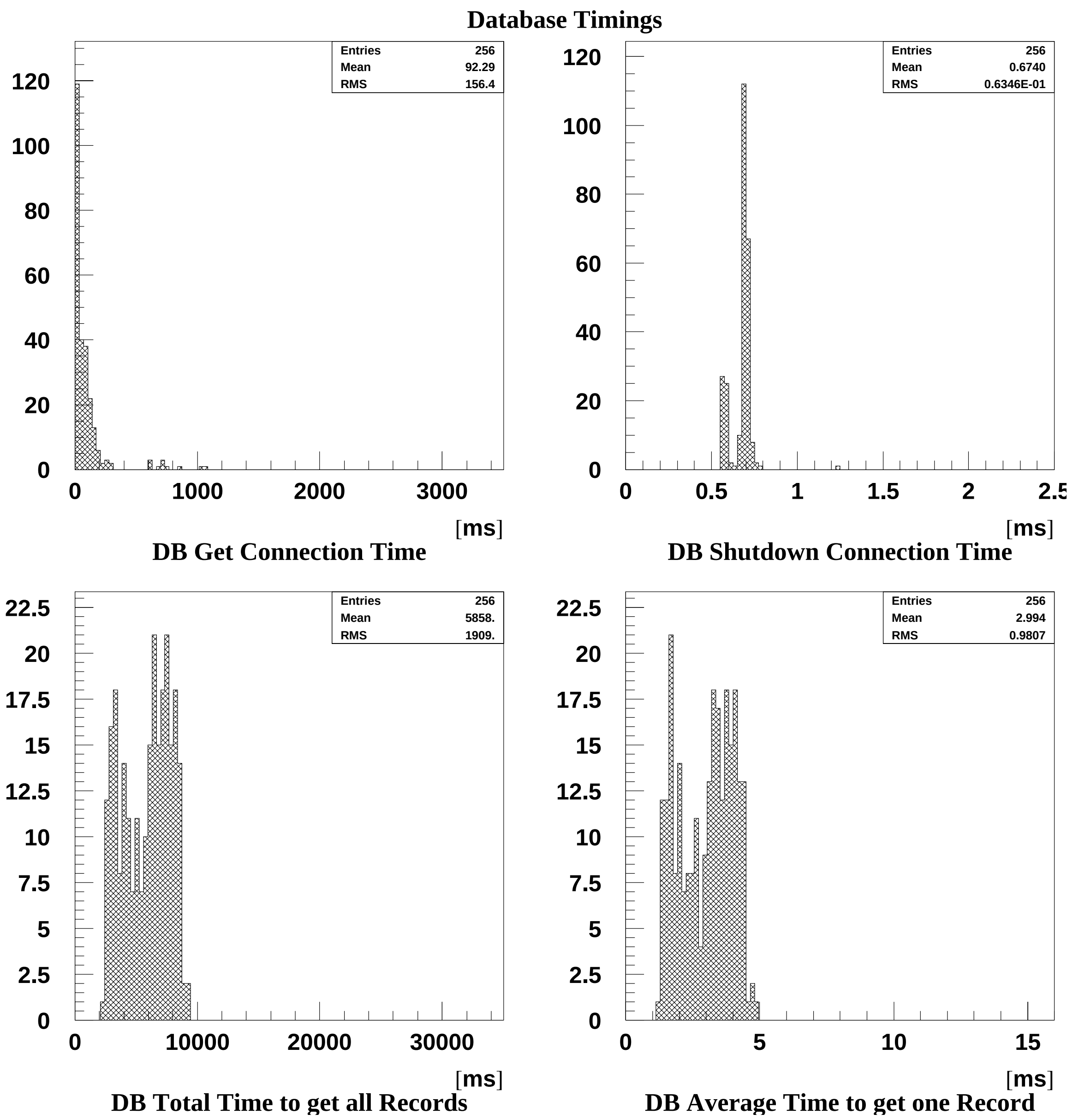


Figure 29 DB timings 8 x 32

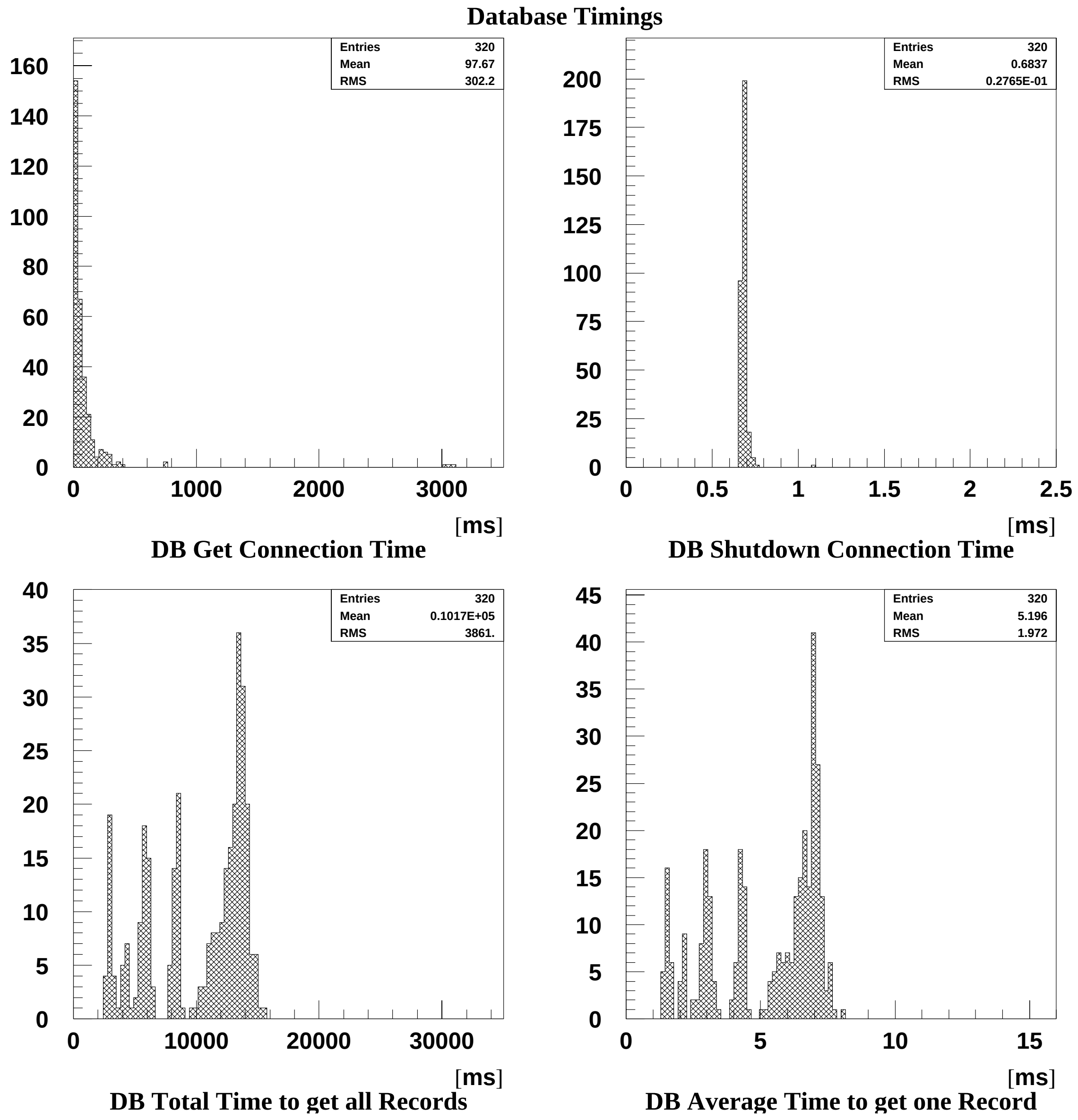


Figure 30 DB timing 10 x 32

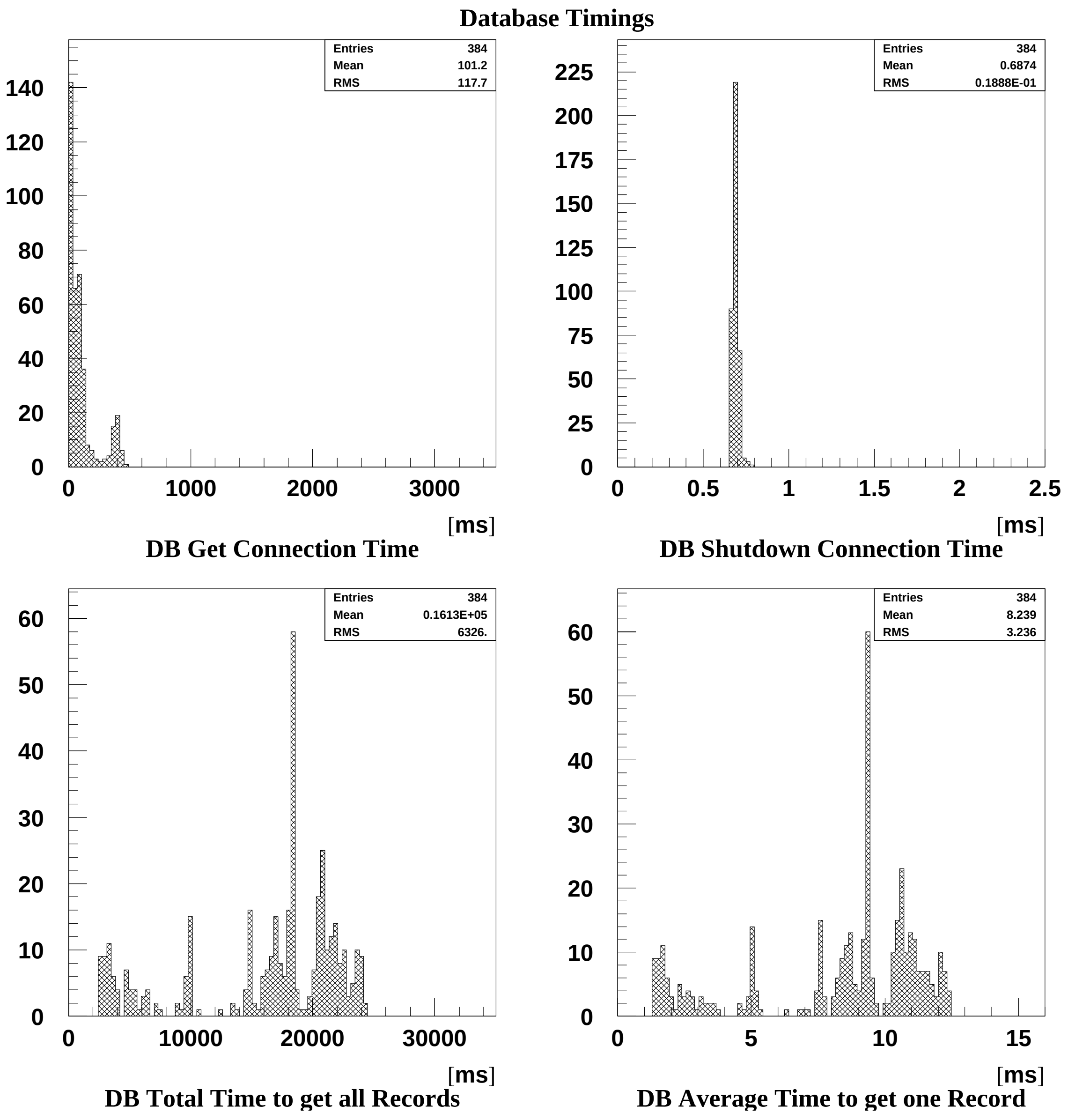


Figure 31 DB timings 12 x 32

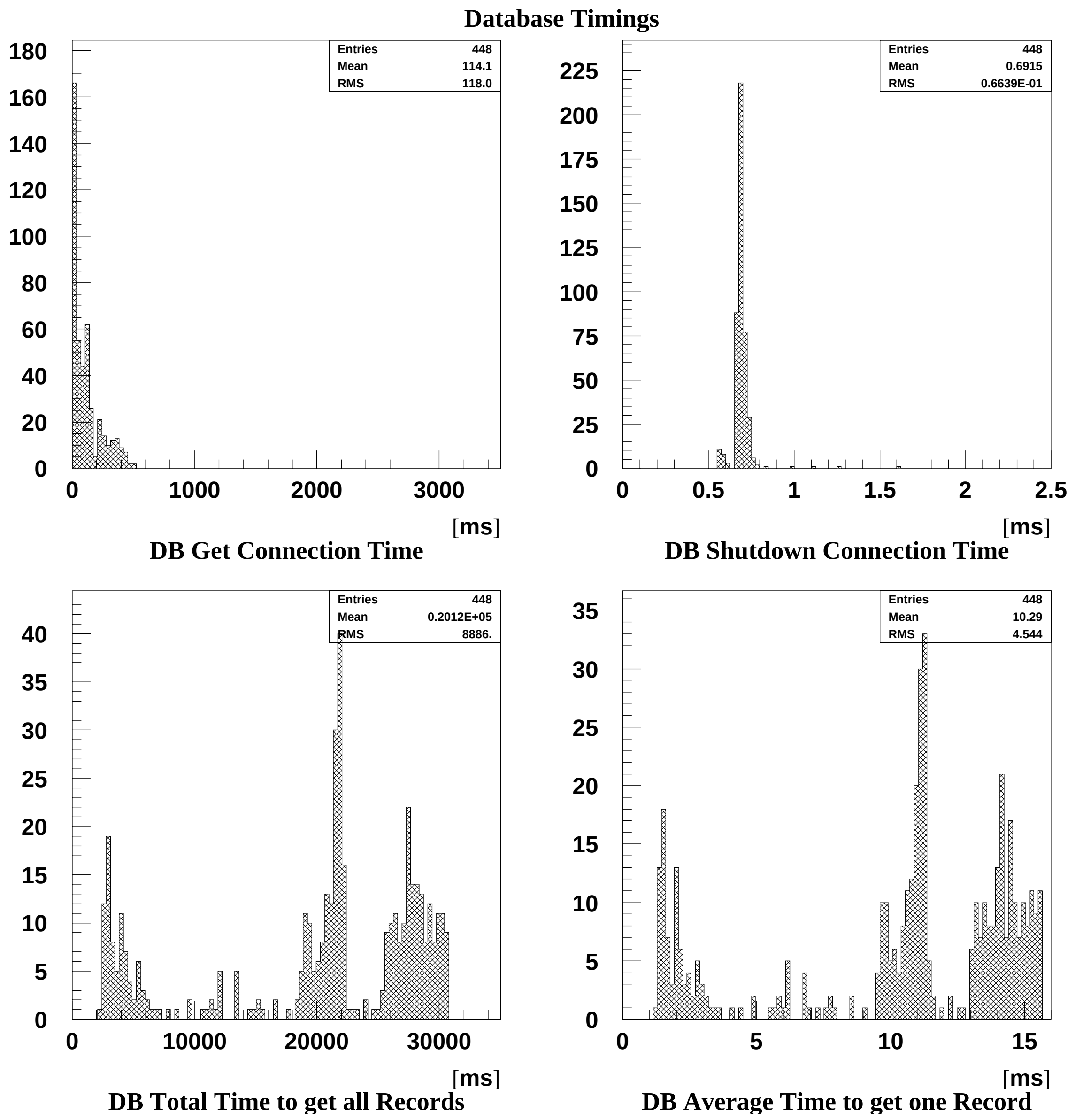


Figure 32 DB timings 14 x 32

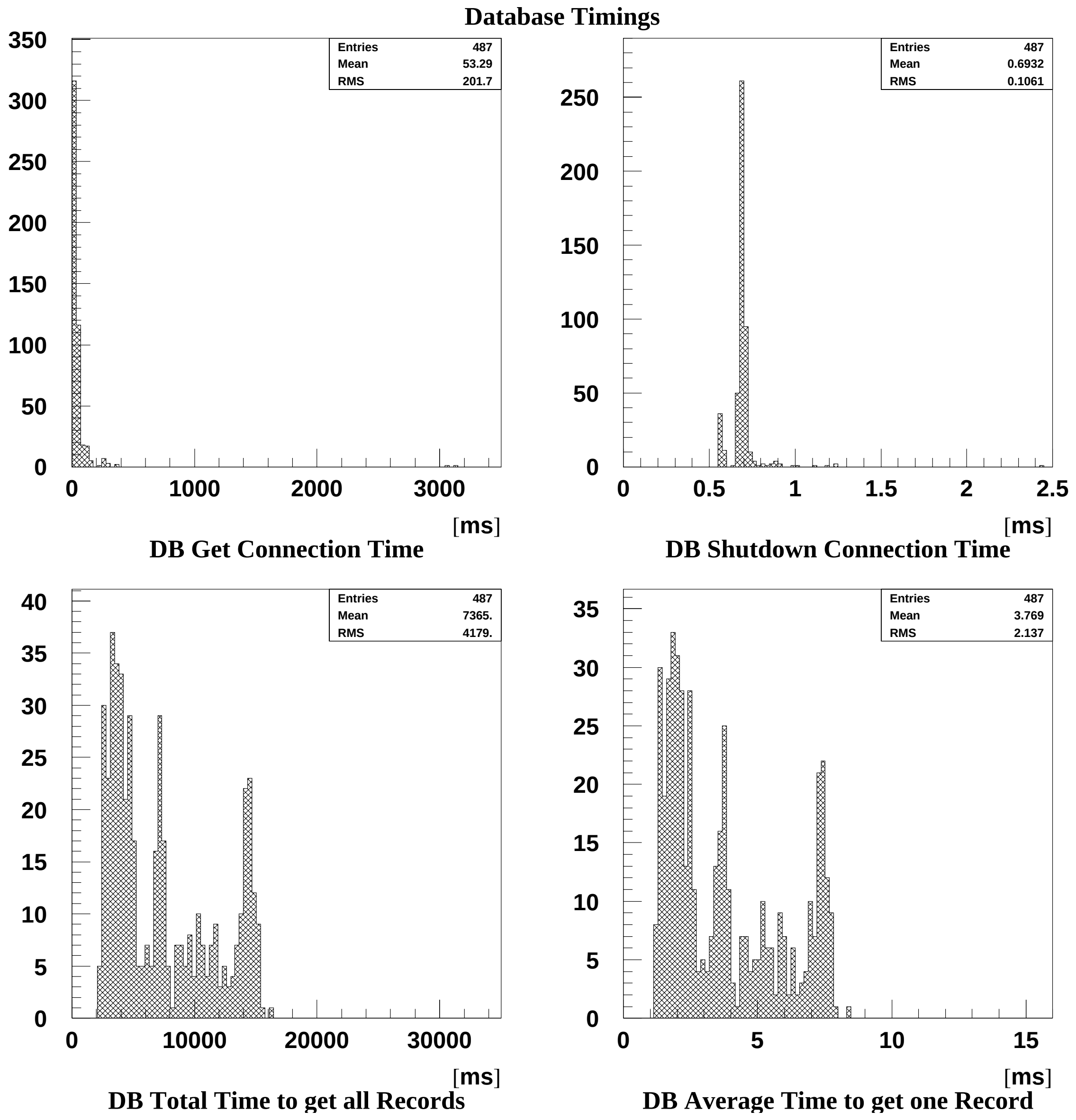


Figure 33 DB timings 16 x 32

8.2.4.2 PESA Steering Controller timings

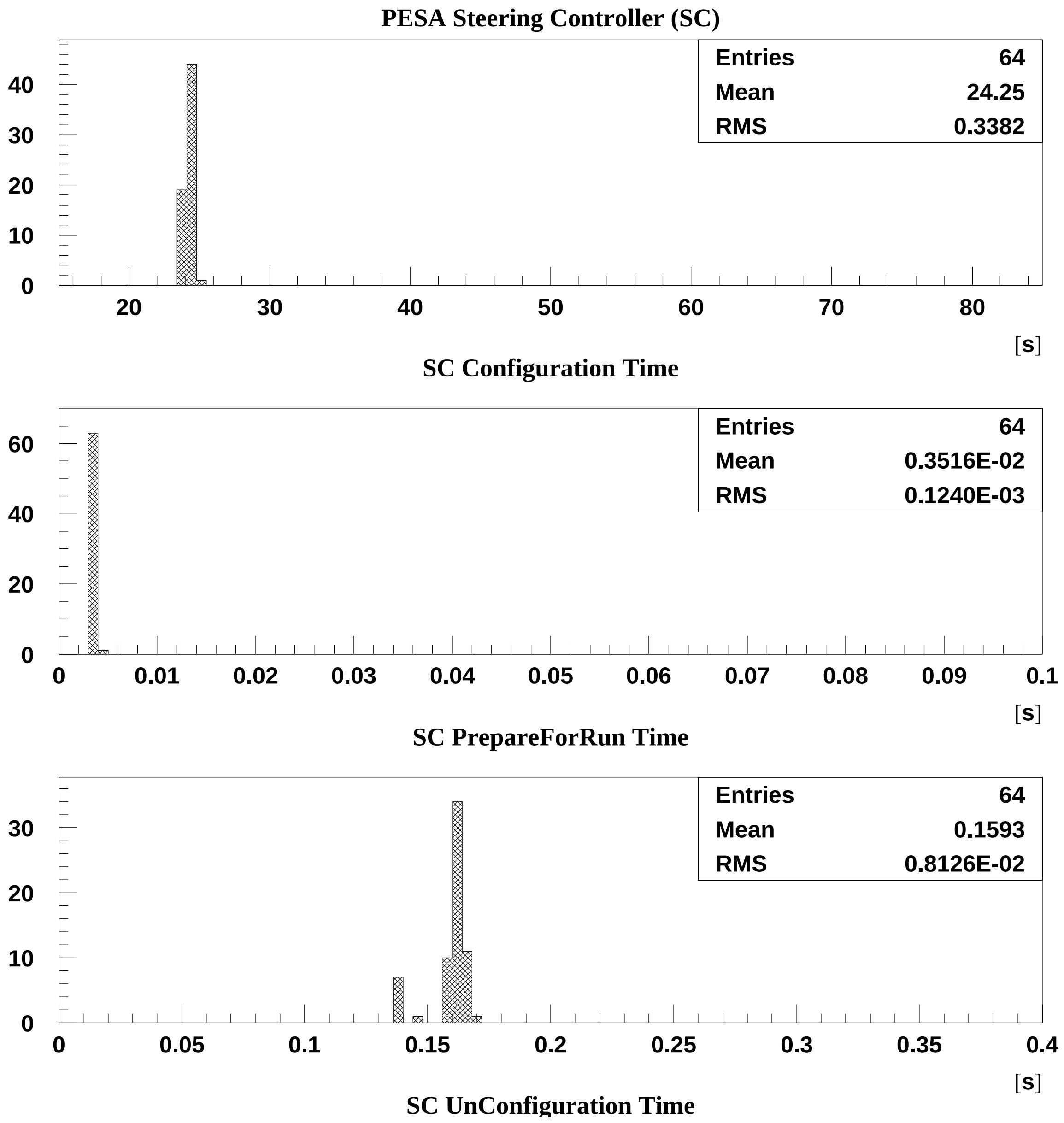


Figure 34 PSC timings 8 x 8

8.2.5 Conclusions and Summary of Experiences

The results obtained in the above measurements can be summarized as follows:

several of the quantities measured increase slowly with the number of L2PUs in the partition:

- the DB get connection time
- the DB time to get all records
- the DB average time to get one record

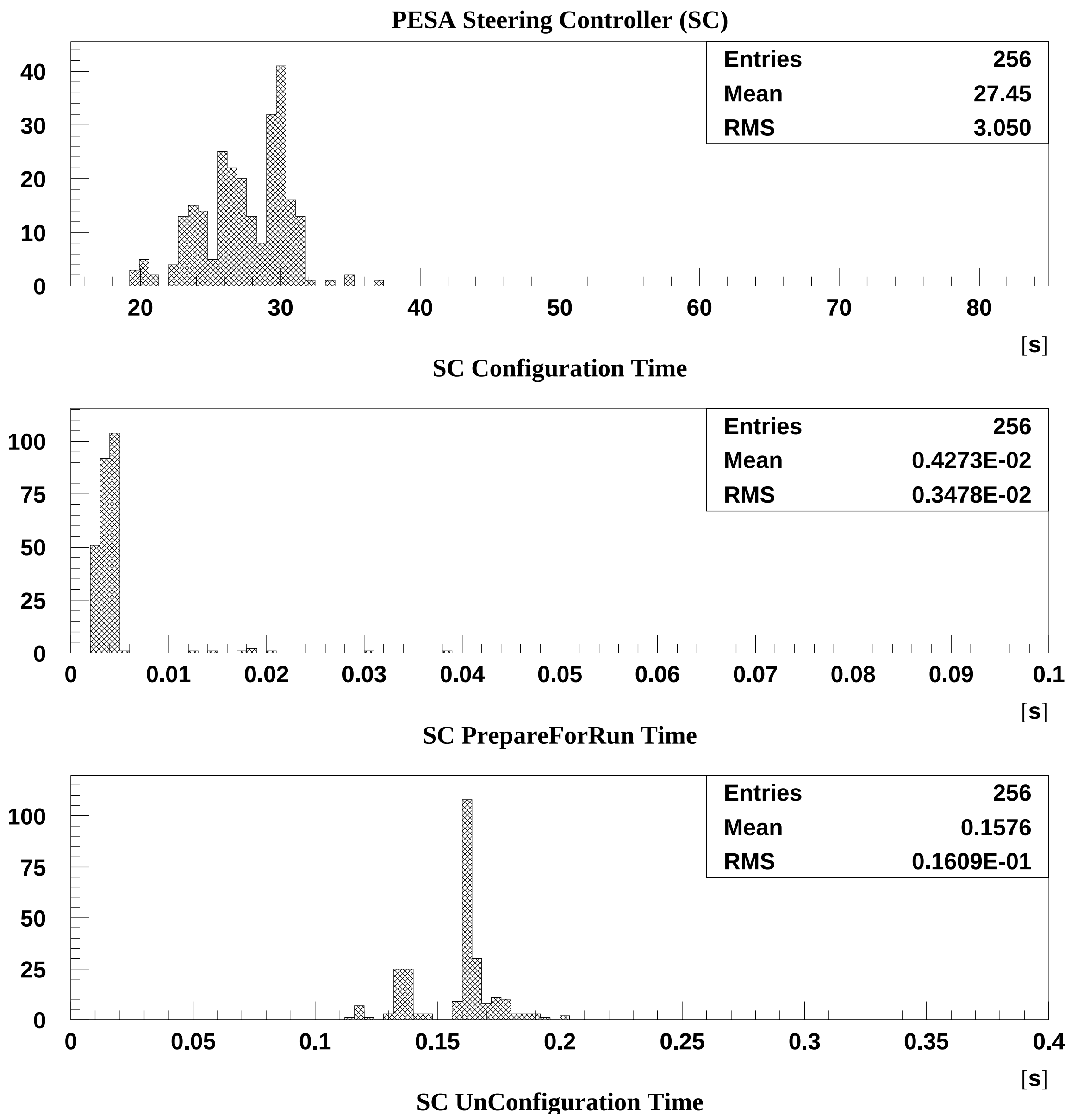


Figure 35 PSC timings 8 x 32

- the PSC Configuration time

the only exception to this trend is visible in the 16 x 32 partition, where 25 out of 512 L2PU measurements were missing. It is possible that those happened to belong to the tail of the distributions. All other quantities measured appear roughly constant with partition size.

It is clear that there are no very surprising features in these measurements. The Level 2 tests with the **muFast** algorithm have clearly shown that with this one algorithm and the partition sizes explored so far, there are no worrisome problems apparent.The number of tests which we were able to perform is however still quite small. Nevertheless, we could not have predicted at the start of the Large Scale Test that we would achieve even this set of tests with this rather good outcome.

A few words on our general experiences during the Large Scale Test:

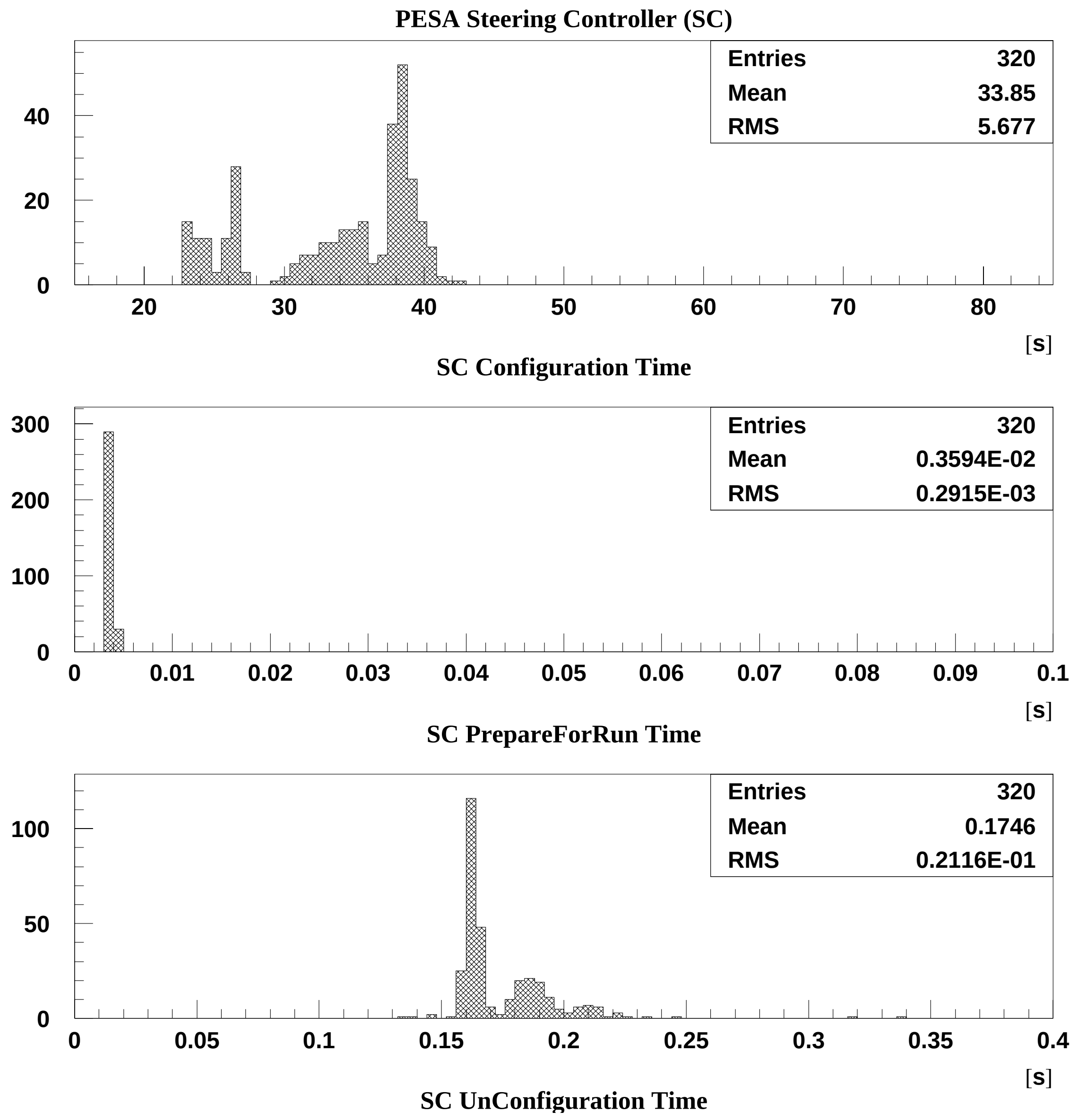


Figure 36 PSC timings 10 x 32

The number of tests we performed successfully is in fact quite small. This is due to the fact that a large fraction of our allotted testing time had to be spent struggling with an array of problems that took time away from the Level 2 algorithm tests. These were the same general problems that also afflicted the other, non-algorithm tests. Without going into great detail, since these problems are described in detail elswhere in this report, the most prominent of these were:

- the Config DB generation scripts started out in a not very usable form. They evolved rapidly during the LST and towards the end had acquired practically all features that were needed for convenient DB generation. Without this, very large partition could not have been tested.
- LocalControllers exhibited a tendency to crash, partly in response to high rates of error message from controlled applications. We had to suppress those.

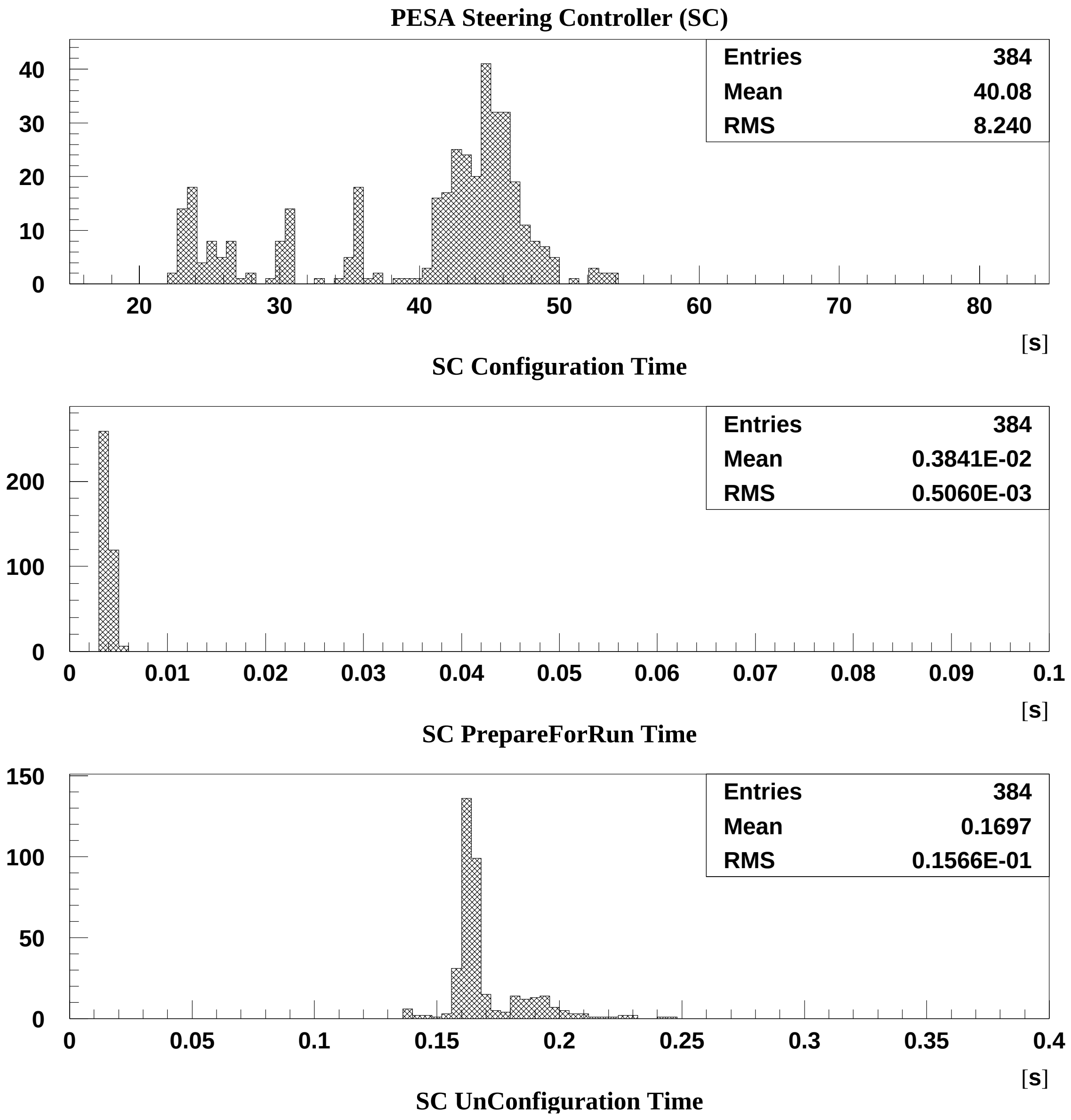


Figure 37 PSC timings 12 x 32

- large partitions were very difficult to start, as the setup component exhibited a non-linear increase in the time it took to start and test pmg_agents. Sometimes we just could not get a partition started, even with drastically increased timeout values, or had to resort to killing all processes belonging to the partition. Often it seemed a matter of good or bad luck whether a large partition would start. E.g. the 16x32 L2PU partition worked fine the first time we ran it, but failed to start an hour later when no change was made anywhere.
- the single most worrying aspect of this situation was that there was no real way to diagnose the situation. Guesswork and trial-and-error had to replace a systematic study of the problems.

Except for the last point, it appears that most of the above problems (and others not listed here) have been identified and corrected or are being corrected. Another class of problems was related with the ssh authentication on the LXBATCH cluster. At certain times the hostkeys for a fraction of the cluster became apparently invalid and caused failures that were at first very hard to diagnose.

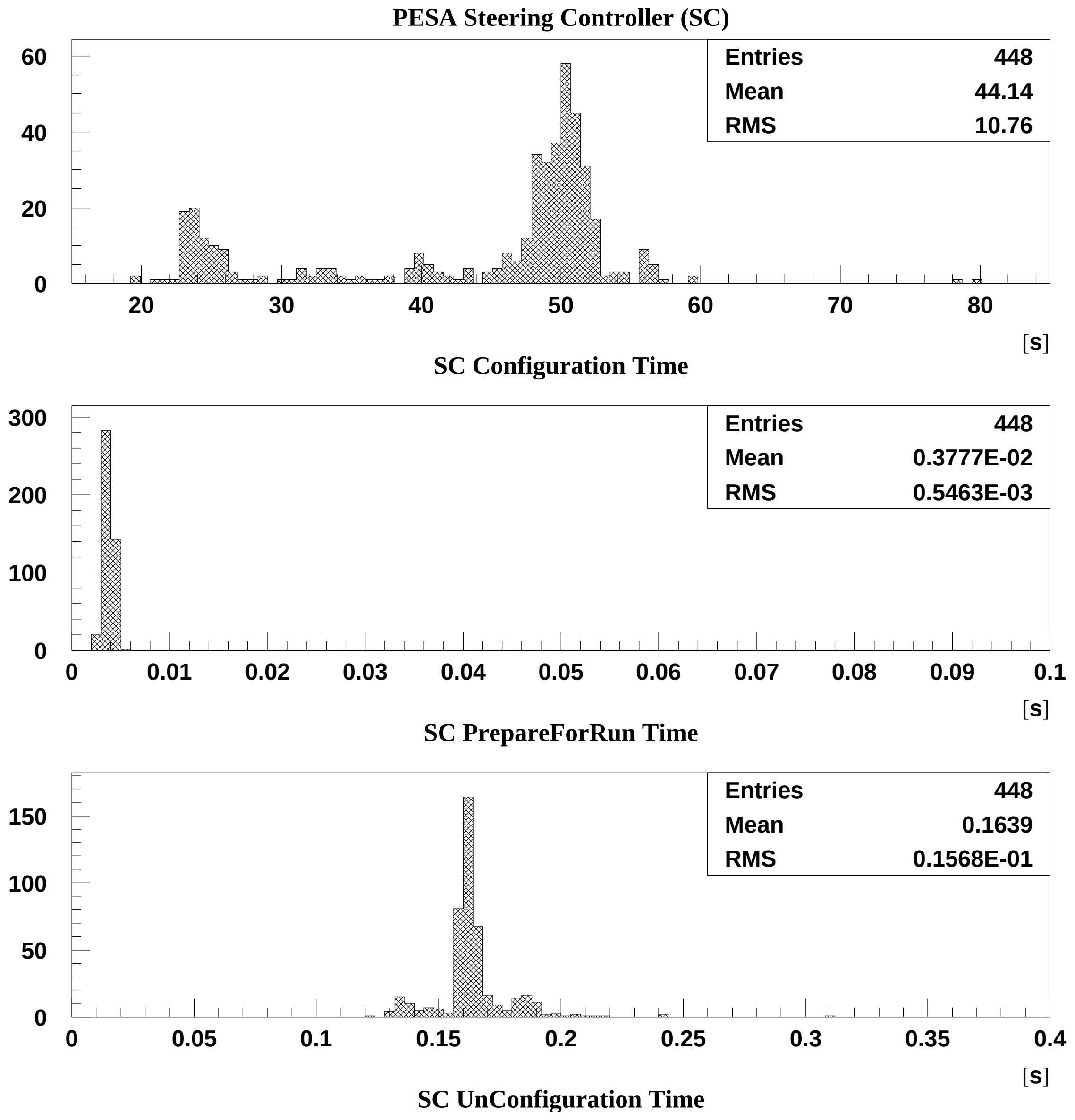


Figure 38 PSC timings 14 x 32

On the positive side, it was a great pleasure to work with the LST group in such a good spirit and cooperation. It's hard to imagine a better cooperation, especially when quite often the different groups are in natural competition for the limited resources of such a Large Scale Test.

8.2.6 Future Steps

The algorithm tests of Level 2 have clearly shown that after this first successful step there are a large number of other tests to be undertaken, when the opportunity for new Large Scale tests arises:

- tests need to be done with multiple runs and in an automated fashion, so that mean values, standard deviations and long tails of time distributions can be studied.

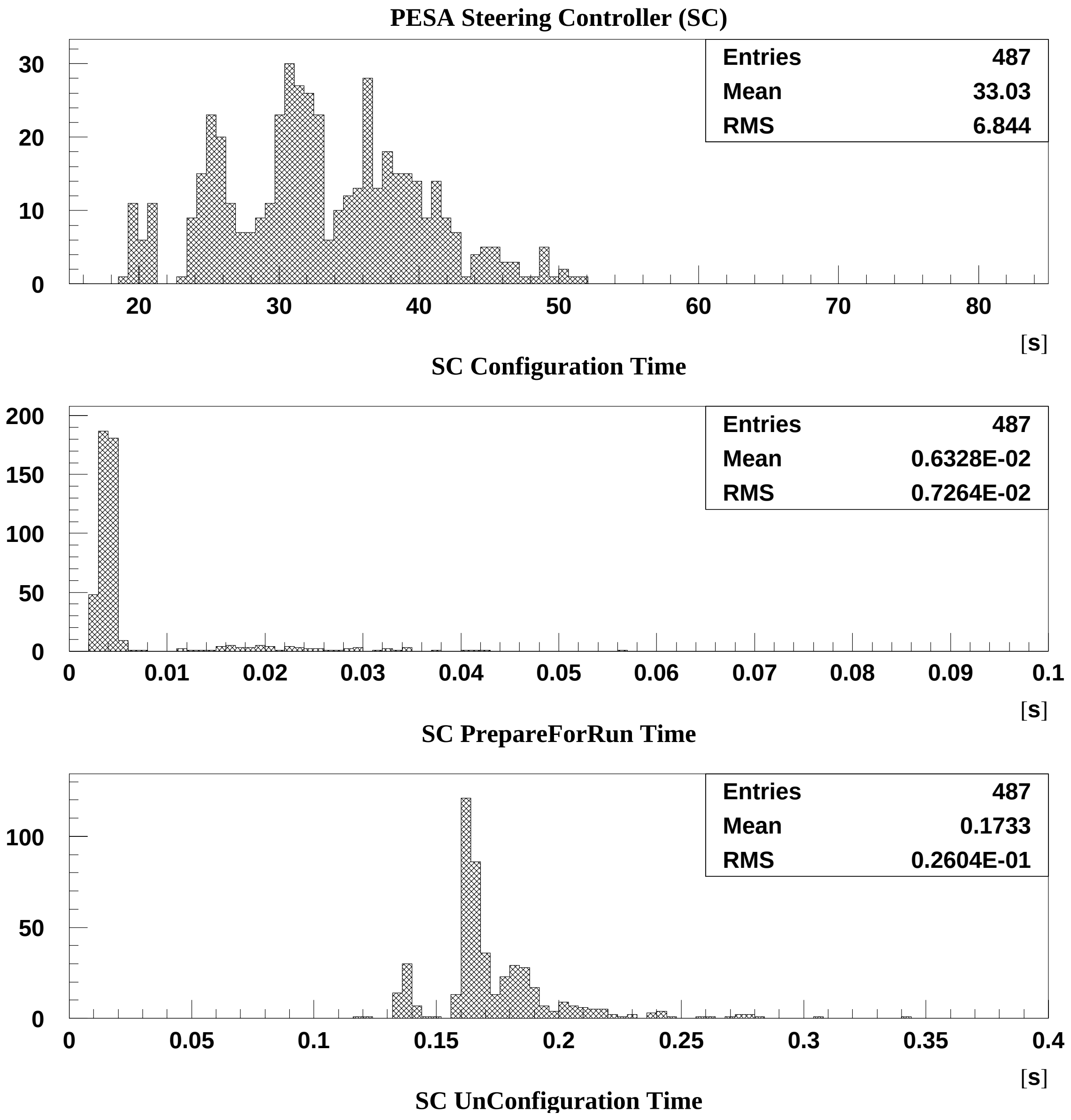


Figure 39 PSC timings 16 x 32

- we did not get to run in a mode where more than one L2PU was running on a single host. This needs to be studied to understand the implications for resource consumption. Also, by packing many L2PUs on a single machine, one can study very large partitions even when the size of the cluster is considerably smaller than the full ATLAS HLT cluster size.
- the behaviour of the Conditions DataBase needs to be studied when a larger number of algorithms is used, and many L2PUs simultaneously access the Conditions DataBase for considerably more data than just the detector geometry.
- need to verify that the problems of starting large partitions have indeed been cured.
- understand the flow of event data with real algorithms and a realistic data network
- the reliability (long-term stability) of the trigger algorithms can be studied with better turnaround time on a Large Scale Testbed, since many L2PUs can run large numbers of events in parallel.

- for a reliable HLT trigger operation it is considered necessary that the trigger algorithms are initialized (configured) not from Python jobOptions.py files, but directly by reading the ConditionsDataBase. Code that will achieve this is being developed and will need to be tested on a Large Scale Testbed.

These and other topics should be foreseen as points to study further. Any possibility for new Large Scale Tests that presents itself should be carefully considered.

8.3 Event Filter Tests with Algorithms

8.3.1 Aims of the Event Filter Tests with Algorithms

The aims of the tests of Event Filter with algorithms were to probe the realistic heavy-loaded running of the High Level Trigger of Atlas TDAQ and to investigate various factors which can be crucial for the performance of the Event Filter in the final Atlas design. Particular topics to investigate in these Tests were:

- simultaneous access to Configuration and Conditions Databases by large number of EF/PT applications;
- TDAQ transition time scaling with size of Event Filter partitions (EF Farm) **-** for dummy PT Steering algorithms and for realistic Offline algorithms;
    - effect of using Offline framework (basic Athena services and libraries) for overall performance;
    - effect of realistic event reconstruction and selection algorithms with respect to "dummy" algorithms used only general Offline framework;
- possible degradation of performance from having configuration or data files on the network-mounted file system (which is important for EF+Algo scenario because a lot of files are used from the Offline software release).

We should remark that the main goal of the Tests was to investigate potential bottlenecks and crucial problems, which may block running the whole TDAQ+EF+Algo system, but not detailed studies of performance of particular instances of EF processes with algorithms, which can be done on small scale computer clusters. However, a deeper understanding of performance of particular EF processes in the future should help us to understand the overall TDAQ+EF performance on large scales.

According to aims of EF+Algo Tests, they were performed in 3 subsequent steps: tests of effect of Offline framework, comparison of dummy HelloWorld and realistic Offline event selection algorithms, and investigations of EF running realistic algorithms. The order of these EF+Algo Test "steps" coincided with the schedule of the increase of the LST cluster size. Thus, smaller cluster was available for testing dummy algorithms, and in the final part of the LST program, when more computers were available, the focus was shifted to behavior of realistic algorithms. Unfortunately, since the LXBATCH cluster time was shared between several TDAQ subsystems, we did not have enough time to perform all tests which were possible for Event Filter technically, which means that more LST tests for Event Filter are needed in the future. We focused on running Event Filter with algorithms at that time, because testing the Event Filter with dummy algorithms on large scale was done before in LST at WestGrid computer facility (May 20005).

8.3.2 EF+Algo Tests Approach and Execution of the Tests

To be able to run Offline algorithms in Event Filter, we needed to provide to the algorithms a set of shell environment variables, which define versions of the used software libraries and the corresponding paths. Because this is needed only when running EF with offline algorithms, such sets of environment variables were not contained in the TDAQ SW release, and we added them to OKS configuration as modified user-local version of "databases/daq/sw/setup-environment.data.xml" file. The values of the variables were defined using the CMT tool for the given release of Offline SW, Athena 10.0.4. We should remark that this implementation was a temporary "trick" to provide the fastest solution during the LST, but after the LST it was implemented in a form of dedicated shell "wrapper-script" for PT application, which derived all the necessary environment automatically from the CMT tool.

The Offline Athena SW used was installed on a local disk of each computer (see chapter on SW installation), thus most of the libraries were used from local disks and not over the network filesystem. However, some user-local files were used (jobOptions and OKS configuration files with environment variables for the Offline algorithms) and the effect of this on the overall TDAQ performance was negligible.

The partitions tested and compared were EF-standalone TDAQ partitions, which do not have lower-level TDAQ components (Level 2 and Event Building). They used emulated SFI, which read events from disk datafile and made them available at the input of the Event Filter, and emulated SFO, which served as output of EF DataFlow, writing only few test events to a disk file. Because tests with varied sizes of EF SubFarms are related to effect of different TDAQ infrastructure (LocalControllers and EFD/PT communications to IS/MRS, etc.) and were investigated as part of "EF without algorithms" program, the EF partitions used for Tests with algorithms were of the same EF SubFarm structure, consisted of one/several equal SubFarms of 20 EF Processing Nodes each. Each Processing Node (EF Node) was running 1 EFD application and 2 PTs. Because of such uniform topology, the total number of EF processes (EFDs and PTs) is equal to number of EF nodes multiplied by 3. The partitions to test were generated by the TDAQ DBGeneration scripts, using version of these scripts adapted to LXBATCH environment (see chapter X for details).

One of the issues for running EF partitions of large size with Athena algorithms was the necessity to increase several timeout values, both of TDAQ infrastructure services and of EF applications. Unfortunately, the whole set of existing timeout parameters and their hierarchy and interdependence are not trivial thing to understand for non-experts of TDAQ infrastructure, that's why we increased values of $TDAQ_IPC_TIMEOUT, $TDAQ_SETUP_START_TIMEOUT and $TDAQ_SETUP_COMMAND _TIMEOUT variables to 4-6 minutes by recommendations of TDAQ experts and then probed increasing InitTimeout and ActionTimeout in EFD and PT configuration objects up to 3,4,5 and 6 minutes until we succeeded to complete the TDAQ boot/configure transitions without crash due to timeout. Finally, we run the largest tested EF partitions with all timeouts set to 6 minutes. However, we should keep in mind that this was only temporary solution and for the final Atlas TDAQ/HLT System the setup/configuration times should be within about 30–60 seconds, i.e. we should find a way to make EF configurable within this time interval and foresee timeout values of about 1 minute.

8.3.3 EF+Algo Tests: effect of using Offline Software Framework

The investigations of the effect of using the Offline framework in the Event Filter were based on comparison of running EF partitions of the same size and structure with PTdummy application as dummy Steering Controller (no Offline framework used) and with simplest Athena algorithm HelloWorld, which has only the skeleton structure of Offline selection algorithm, but does not do any processing actions on events.

The partitions run for these tests were consisted of 1, 5 and 10 EF SubFarm by 20 EF Nodes each. Thus, up to 200 EF Nodes, or up to 600 EF processes were run (which corresponds to number of available computers with installed Athena software at the beginning phase of 2005 LST). In addition, for PTdummy configuration, 2 larger partitions were tested, with 15 and 20 SubFarms by 20 EF nodes each (the total number of available computers was ~ 400, but Athena software was installed only on roughly half of them). Each partition was run from 3 to 6 times (larger partitions were run less times because they take longer time, and also sometimes crashes due to TDAQ infrastructure problems were observed). The measured transition times (see Chapter X for description) were averaged over series of the same runs. The scaling behavior can be compared by number of EF nodes, or equivalently, by number of total EFD+PT processes. The measurements averaged over the same partitions are presented in Table 9.

Table 9 TDAQ transition times measured with EF standalone partitions with PTdummy and Athena HelloWorld (the transition times are in seconds).

Partition	**#EF hosts**	**# EF proc**	**setup**	**boot**	**shut down**	**cold start**	**cold stop**	**config**	**unconfig**	**Total time**
PTdummy (no Offline framework)										
01x20	20	60	23	4	5	14	27	7,7	6,3	77
05x20	100	300	38	5	6	14	28	8,2	6,3	94
10x20	200	600	73	9	6	9	29	9,1	6,2	136
15x20	300	900	173	11	6	31	30	16,7	7,4	249
20x20	400	1200	235	20	8	56	33	26,4	8,2	339
Athena HelloWorld (with Offline framework)										
01x20	20	60	4	22	6	14	28	9,6	6,2	77
05x20	100	300	33	6	7	17	29	9,9	6,2	93
10x20	200	600	110	7	8	21	31	10,1	7,4	176

As seen from the table, the most influenced by partition scale transitions are "setup", "boot" and "config". This is explained by the fact that in these transitions the TDAQ infrastructure is set up (in particular, LocalController applications per EF SubFarm), EF applications (EFDs and PTs) start, read their configuration from the Conf DB server and load necessary shared libraries. The most sensitive to the partition scale TDAQ transition times are represented in the Figure 40.

It is seen that the "TDAQ setup" times are of an order of magnitude larger than the "boot" and "configure" times. The "setup" time is related to TDAQ infrastructure and was not the goal of the investigations of Event Filter tests with algorithms. On the other side, the "boot" and "configure" times are not bottleneck in the overall TDAQ running. The overall scaling of transition times look reasonably linear, growing proportionally to the number of EF processes and no side-effects have been observed. In

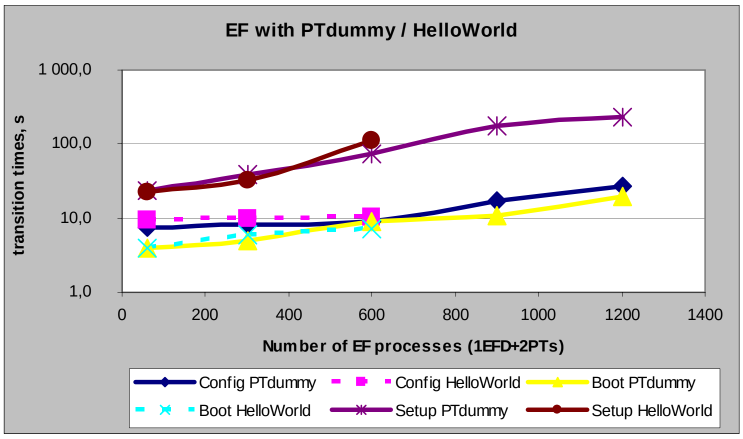


Figure 40 Scaling behavior of the most sensitive TDAQ transition times with number of EF processes (1EFD+2PTs per EF node).

particular, reading configuration from ConfDB simultaneously by many PTs looks correct, without problem of concurrence between many PT application, both on side of PT and on side of ConfDB.

The effect of including the Offline framework with respect to running only dummy Steering Controller (PTdummy) is very small and almost negligible comparing to the dependency on the total partition size. The increase of the "boot" and "configure" transition times with scale of the EF Farm (partition) looks reasonable, as linear dependency on number of EF processes. Comparing behavior of the equivalent EF partitions with PTdummy and HelloWorld on the scale up to 600 EF processes, we can foresee similar linear scaling for partitions with HelloWorld on the range from 600 EF processes up to 1200. The only case which is remaining to be tested is the behavior of such partitions on Very Large Scales, which is the final Atlas design for Event Filter: 3000 EF processes on 1000 EF nodes.

Using the Offline framework means mostly loading by PT application the additional Offline libraries in memory (the PT application with PTdummy Steering consumes ~ 90 MB memory in total, including ~17 MB for shared libraries, while PT with Athena HelloWorld algorithm consumes ~135 MB memory including ~47 MB for shared libraries). Taking this into account, we can conclude that this particular operation does not introduce any slow-down in overall TDAQ/HLT performance having the Offline/HLT/TDAQ SW releases installed on local disks of each EF node and having physical memory of that amount (at least 512 Mb or more). Having only few XML/jobOptions files on network-mounted disks (in local patch or local user area) is also safe enough for the overall TDAQ / HLT performance.

8.3.4 EF+Algo Tests: effect of realistic Offline Selection Algorithms

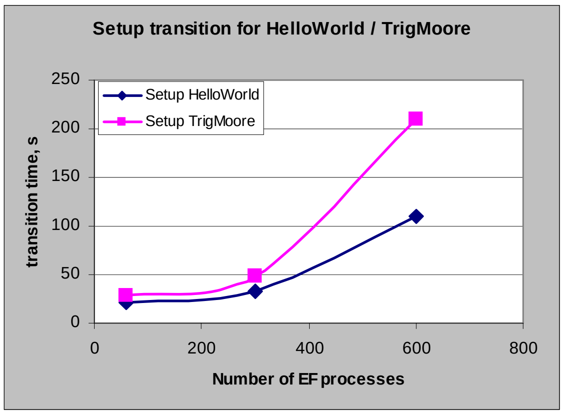
Next step of EF+Algo Test program was to investigate the effect of realistic Offline selection algorithm with respect to dummy Athena HelloWorld algorithm. This means mostly the additional memory and CPU consumption by realistic Athena algorithms (comparing to PTdummy and PT with Athena HelloWorld, the PT application with Athena TrigMoore algorithm consumes ~620 MB of memory including ~145 MB for shared libraries) and additional network and CPU resource-consuming operation of accesing Condition DB and reading large amount of Atlas Detectors geometry data there.

The measurements of TDAQ transition times were performed with EF-standalone partitions of 1, 5 and 10 SubFarms by 20 EF nodes each, which for "1EFD+2PTs" node configuration correspond to 60, 300 and 600 EF processes. The largest partition with TrigMoore algorithm, having 600 EF processes, was of slightly different structure: 15 SubFarms by 20 EF nodes, with "1EFD+1PT" configuration, which in terms of number of EF processes is equivalent to partition of 10 SubFarms by 20 nodes of "1EFD+2PTs". We used this partition because we observed problems to run such partition with "1EFD+2PTs". The reason of failure to run equivalent partition with 2 PTs per EF node is probably related to the Conditions DB access and will be discussed in next sub-section.

The measurements of TDAQ transition times were performed from 3 to 6 times for each configuration (depending on failures due to general TDAQ problems from time to time, like for example processes remained running from previous TDAQ run). The results averaged over the same partitions are presented in Table 10 and are illustrated in Figure 41.a and Figure 41.b for comparison.

Table 10 Measured TDAQ transition times for Athena HelloWorld and TrigMoore algorithms (comparison).

Partition	**#EF hosts**	**# EF proc**	**setup**	**boot**	**shut down**	**cold start**	**cold stop**	**config**	**unconfig**	**Total time**
Athena HelloWorld (Conditions DB not used)										
01x20x1EFD+2PT	34	60	22	4	6	14	28	9,6	6,2	77
05x20x1EFD+2PT	126	300	33	6	7	17	29	9,9	6,2	93
10x20x1EFD+2PT	241	600	110	7	8	21	31	10,1	7,4	176
Athena TrigMoore algorithm (using MySQL Conditions DB for Detector Geometry)										
01x20x1EFD+2PT	32	60	28	4	7	52	30	47,6	6,3	124
05x20x1EFD+2PT	128	300	48	5	9	125	301	116,1	10,9	494
10x20x1EFD+2PT	348	600	210	10	9	150	301	139,7	8,5	676



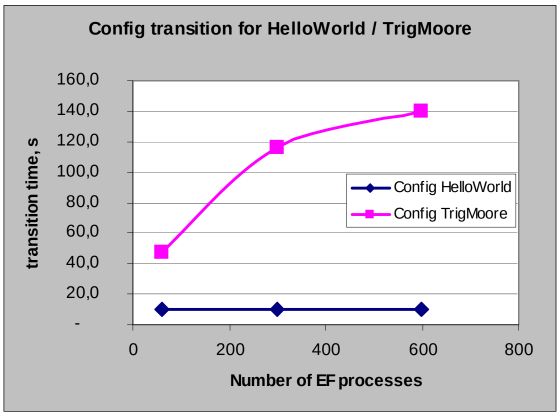


Figure 41 (a, left) TDAQ "configuration" and (b, right) "setup" transitions for dummy Athena HelloWorld and realistic TrigMoore algorithms.

It is seen that TDAQ back-end setup times are very similar for HelloWorld and TrigMoore for 60 and 300 EF applications, while for 600 applications they differ a lot. This difference is most likely related to the different EF Farm structure at this scale, when EF/HelloWorld partition consisted of 10 SubFarm of

20 nodes (with 2 PTs per node) and EF/TrigMoore partition was composed of 15 SubFarms of 20 nodes (with 1 PT per node).

Another effect is that the configuration times of TrigMoore partitions are in general much longer than corresponding times of Athena HelloWorld partitions for all sizes of EF partition. Moreover, the configuration times for TrigMoore increase more significantly with size of EF partition. The configuration times are defined mainly by the total number of the EF applications and do not depend (or, perhaps may depend only very slightly) on structure of the EF Farm, i.e. this effect is different and independent from the effect observed with "setup" transition, so, we can consider them separately.

The behavior of configuration times with TrigMoore can be explained by the fact that PTs with TrigMoore algorithm need to read much more data to configure the realistic algorithm, and especially the detailed Atlas Detectors geometry data from the Conditions DB. Besides the amount of data read to configure the Athena algorithm, another effect can be (and later has proven to be) important here, namely the mechanism of accessing the Conditions DB. It is important also because all PT applications with TrigMoore configure and try to access/read CondDB nearly simultaneously. Due to these reasons, the CondDB access was investigated later in more details. The behavior of the configuration time for TrigMoore algorithm let us suspect that several factors contribute together here.

8.3.5 EF+Algo Tests: Conditions DB performance with TrigMoore

To investigate the effect of accessing the Conditions DB and reading geometry data from it during the configuration transition, we compared EF partitions with TrigMoore of different sizes, using in one case MySQL server as CondDB and Oracle server in another case. The used EF partitions were of the similar EF SubFarm structure as above: 1, 5 and 8 SubFarms by 20 EF nodes, 1EFD+2PT applications per EF node. Unfortunately, we did not succeed to run such EF partitions at larger scales (neither with MySQL nor Oracle servers for CondDB). TDAQ crashed due timeouts from some of the PT applications (randomly), and even trying to increase timeout values for both TDAQ services and EF applications up to 8, 10 and 12 minutes did not allow us to run EF partitions on all of ~400 computers with Athena/HLT software available at that time. Since after some level of timeouts (12 minutes) the crashes occurred in shorter time interval than the timeout we set, we concluded that the reasons of failures were of different nature, most probably originated from accessing the Conditions DB and reading the geometry data.

The measurements were performed from 3 to 5 times for each case and averaged over the same running configuration. For TrigMoore with MySQL server as CondDB, the largest EF partitions which we succeeded to run was the described above partition of 15 SubFarms by 20 EF nodes with 1EFD+1PT application per EF node. Keeping in mind slight different in number of LocalController applications (per SubFarm) we still can investigate the effect of larger number of PT applications accessing the CondDB. The measured TDAQ transition times are presented in Table 11.

Table 11 TDAQ transition times for PT with TrigMoore algorithm, comparison between MySQL and Oracle servers used for Conditions DB.

Partition	#EF hosts	# EF proc	setup	boot	shut down	cold start	cold stop	config	unconfig	Total time
TrigMoore algorithm with MySQL server as Conditions DB										
01x20x1EFD+2PT	32	60	28	4	7	52	30	47,6	6,3	124
05x20x1EFD+2PT	128	300	48	5	9	125	307	116,1	10,9	494
08x20x1EFD+2PT	200	480	65	7	10	178	299	170,3	6,6	555
15x20x1EFD+2PT	348	600	210	10	9	150	301	139,7	8,5	676
TrigMoore algorithm with Oracle server as Conditions DB										
01x20x1EFD+2PT	34	60	21	4	7	104	29	99,2	6.6	168
05x20x1EFD+2PT	128	300	51	6	7	308	298	301,8	8,4	671
08x20x1EFD+2PT	200	480	88	7	10	309	301	301,5	9,5	711

It was clearly observed that the access to the CondDB currently is the bottleneck for the large scale partitions with realistic Athena algorithms. Unfortunately, we did not have enough time to investigate the issues and various factors of accessing CondDB and reading geometry data from it by TrigMoore algorithm, and this is one of the goals for the future Large Scale Tests of HLT with realistic algorithms, which are definitely needed and important.

However, from behavior of the Event Filter partitions with TrigMoore and from our overall experience of running large EF partitions, we can certainly indicate several potential sources of problems, which should be probed in the future:

1. potential problems on CondDB server side:
   a. several times we observed error messages that the limit on the number of DB connections has reached, thus, we can try to increase such limits on the DB server side and check if this will allow us to run EF partitions of larger size or if it will improve the TDAQ configuration times;
   b. we may need to check if the overall performance of DB servers reaches limits and is it possible to improve the performance by tuning parameters of DB server, or if we may need to increase the hardware resources of the DB computer;
   c. unfortunately, we did not observe expected improvement from possible caching of the data by the CondDB server during subsequent requests of the same type, when we booted/configured/run the same EF partition several times within the same TDAQ run; we may need to check if such caching is possible by proper adjusting the DB server parameters;
   d. depending on the two above tests, we may check, if replicated copies of CondDB (i.e. to say, having duplicated DB server per EF SubFarm) may be needed to solve the problems.

2. potential problems on the side of PT applications
   a. we may need to check if the source of the problem is just simultaneous requests to and reading from the CondDB; for this we may test a modified version of PT application with introduced random delay in the configure method before starting accessing the DataBase; such random delays will smear in time the DB access by many PT application and will let us avoid intense simultaneous requests from numerous PTs, and as a result, we expect better overall performance at the configuration transition;
   b. perhaps we should think about and understand, if we may need additional tools to "cache" the large amount of geometry data (or calibration data in other cases) from CondDB somehow on the local disk/memory of EF node.

8.3.6 EF+Algo Tests Summary and Conclusions

The Large Scale Tests of summer 2005 at CERN were the first Tests of running Event Filter with realistic Offline event selection algorithms on large scale. Besides the general studies of scaling of TDAQ/EF performance with size of EF partitions, two important potential bottleneck points were investigated: intense access by numerous PT applications to Configuration DB and to Conditions DB. Both topics originally shown serious problems on large scale, which block the possibility of stable running of the final Atlas design Event Filter Farm. The first problem, related to finding PT configuration objects in ConfDB, was successfully solved during LST 2005 and this is an important step towards the final system. The second topic, intense access and reading the Conditions Database, was also probed and as a result of our tests, several potential sources of problems were indicated. Unfortunately, due to the limited duration of the Tests, we did not have enough time to solve these problems and thus, they are subjects for the future Large Scale Tests.

The experiences accumulated during the 2005 Large Scale Tests of Event Filter with realistic Offline event selection algorithms and the potential problems found lead us to clear conclusion that we definitely need more possibilities and time to continue testing the Event Filter on the large scale in the near future.

8.3.7 EF+Algo Future Steps

From the results of the performed Large Scale Tests of Event Filter with algorithms, we may draw out preliminary list of tasks for the future Large Scale Tests:

1. more detailed investigation and solution of the observed problems with accessing and reading data from CondDB (the particular tests to do are listed in Section 8.3.6);
2. TrigMoore is just one of many other Offline event selection algorithms intended to run in the Event Filter, and other algorithms, like jet reconstruction, can be even more resource consuming; they are subjects for the continuation of investigations (note that for such tests, the algorithms must be provided to us by corresponding developer groups in good working conditions well before the tests);
3. we did not have enough time to test and investigate the stability and fault tolerance issues on the large scales, this is also important topic for the future;

4. with realistic Offline event selection, and later, with additional monitoring and calibration Athena algorithms running in Event Filter, the performance of PT with various algorithms and load-balancing between PT applications (EF nodes) in the overall EF Farm will become a crucial point to investigate and optimize; this is another important thing to do in order to provide robust and performant Event Filter system for the Atlas Experiment;
5. and additional topic found to be important on the large scales is various technical tricks needed to manage such big EF Farms: cleaning disks filled with log files, killing processes left running from previous partitions, diagnostics of OS, network and installed Atlas Software; this is especially important for resource-consuming and thus, relatively slower running of Event Filter partitions.

9 Monitoring Services Scalability Tests

Three monitoring services, namely Event Monitoring, Information Service and Online Histogramming, have been tested in order to understand their performance and scalability. These services form the basis of the ATLAS online monitoring system and the results of these tests can be used to estimate the overall monitoring system scalability and performance.

9.1 Event Monitoring Service Tests

The aim of the Event Monitoring scalability tests was to prove performance and scalability after a re implementation that has been done since summer 2004. A bottleneck of the former implementation, the central distributor has been removed following the Peer-to-Peer paradigm, which should result in improved scalability on the monitoring task side. Another aim of the test was to evaluate the overhead in the conductor application, induced by measures guaranteeing fault tolerance (like sampler pinging) and efficiency (like sampling rate adaptation). Finally, the tests have been used to evaluate the network transfer overhead generated by CORBA in regard of different event sizes.

9.1.1 Tests Description and Configuration

All the following measurands result from tests at machines in building 513 Vault, interconnected by a Fast Ethernet router. For the different test aims, several configurations have been created.

In configuration A, N monitoring tasks have been connected to one sampler, all monitoring tasks requesting events matching the same selection criteria. As described in the Event Monitoring design document [16], in this case monitoring tasks are arranged in k-nary trees, parameter k depending on the conductor configuration. During the tests, different values for k have been used, for configuration A tested parameters include 1, 2, 5 and 10. For this configuration the most interesting value is the number of events per second received by the monitoring tasks.

In configuration B, a single sampling application and different numbers N of monitoring tasks, all requesting events with different selection criteria have been started, resulting in sampling applications running N concurrent event channels, each served by a different thread. For this configuration, the most interesting value is the number of events per second received by the connected monitoring tasks, which reflects the rate at which the event channels could ship events and the CPU time spent by the sampling application, which reflects the CPU load generated by the different event channels.

Every test in configuration A and B has been performed using three different profiles, a ROD, ROS and EventBuilder profile, using event sizes of 512, 2096 and 500000 4-byte words.

9.1.2 Tests Approach and Execution

For the execution of tests, test applications from the DAQ/HLT-I release have been used. Test scripts started a given number of test applications on the testbed machines using the “ssh -x” command. The test applications produced local raw result data files, which were copied to a central directory and parsed by an analysis script, calculating mean values and formatting output data to result files, from where they could be plotted using Excel or GnuPlot. General measurands that have been recorded for all test applications involve CPU time spent, total time spent and events shipped or received per second.

9.1.3 Tests Results

After completion of the tests a bug has been detected concerning CORBA's memory management. This bug was present in the version of the Event Monitoring framework used for the tests and renders all measurand recorded in child monitoring tasks for configuration A unusable, as in the case of child monitors no actual event data but zero-size events have been received. Although this bug has been fixed in current versions, there was no time to repeat tests for configuration A with the patched version. Regarding this bug, one has to be careful with the interpretation of , which shows the event rate in events per second in the ROD profile against the number of concurrent monitoring tasks, all requesting events matching the same selection criteria. Actually the value resulting from the test with a single monitoring task is significant, as only child monitors did receive zero length data, root monitors were not affected by that bug. Using the ROD profile, a single monitor connected to a sampling application reaches a net data transfer rate of about 30 MBit/s, which is less than one would expect from a 100 MBit connection. Comparing this value to the ROD-profile values recorded in configuration D, which is about 16.9 Mbit/s per monitor, accumulating to a total data transfer rate of 84.6 Mbit/s in the sampling application this leads to the conclusion, that monitoring tasks are the limiting factor, regarding how many pushEvent calls per second they can handle at maximum. A reason for this might be locking mechanisms used in order to synchronize concurrent access to event buffers by the user and the pushEvent calls, but this still has to be investigated in more detail. Another reason might be limitations in the CORBA implementation used for the Event Monitoring framework. As one can see in figure 2 a single monitoring task in the EB profile - with much bigger and less frequent pushEvent calls - receives event data at the maximum rate at which the sampling application can ship event data, which is about 85 Mbit/s.

Although the bug prevented actual data from being transferred between parent and child monitors, the almost linear scalability in the number of concurrent monitoring tasks that can be seen in figure 1 is what one would expect from the re implementation. As no central conductor is involved in actual data transfer and all the monitoring tasks are participating in data distribution, there should be no more bottleneck preventing scalability. In the former implementation, even with zero length data, the rate at which nextEvent calls by the monitoring tasks could be answered by the central distributor would have dropped linearly with an increase in the number of monitoring tasks. Figure 42 proves that at least with empty pushEvent calls this problem is solved.

Figure 43 shows the received event rate of monitoring tasks in events per second for the ROD and EB profiles in scenario B. In this configuration, sampling applications have to create a different sampling channel for every connected monitoring task, transferring data to every single connected task, instead of transferring data to a root monitor only, letting it spread event data among its children. This does not scale in the number of event channels, but there does not seem to be a proper solution to this problem. Event data actually have to be transferred several times, as they are different for each channel respectively for each monitoring task. In the re implementation, at least this does not influence other sampling applications connected to the same conductor, so the effect is limited to monitoring tasks connecting to that sampling application, rather than having influence on the Event Monitoring performance of a whole partition.

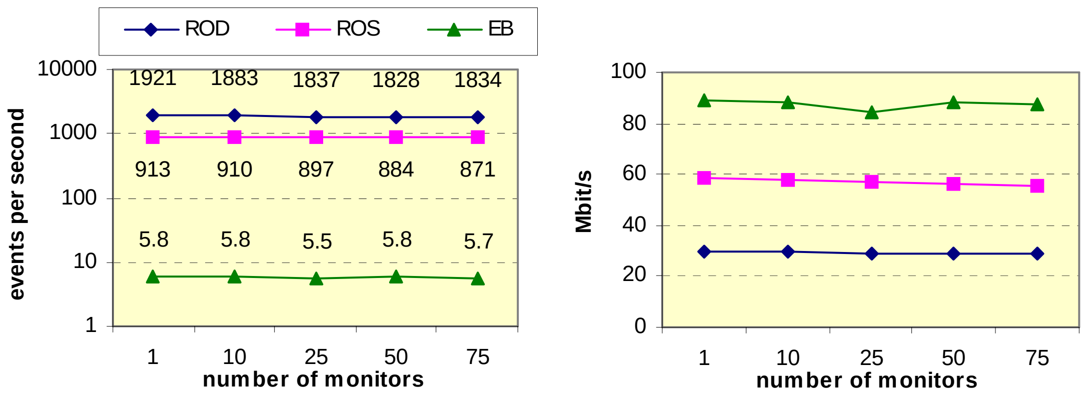


Figure 42 Configuration A results

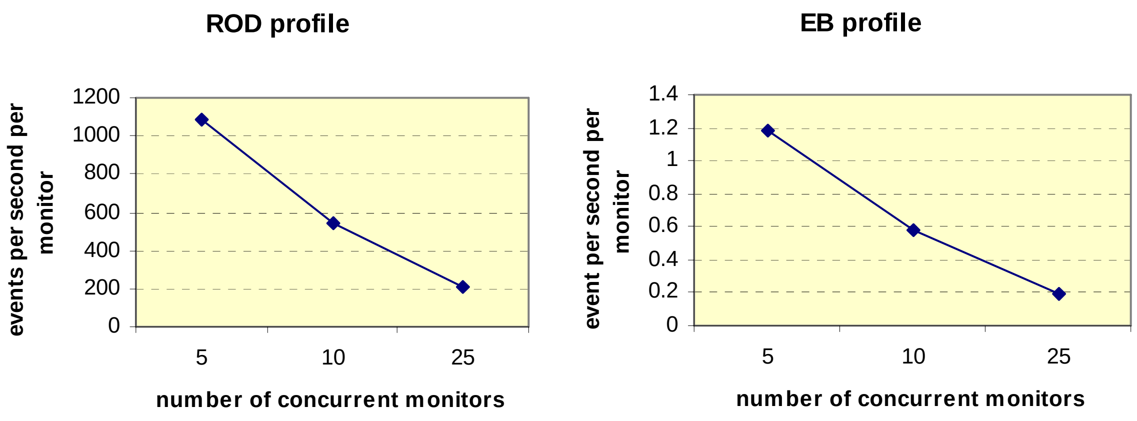


Figure 43 Scenario B

For EB profile tests with 5 concurrent event channels, every connected monitoring task receives 1.17 events per second, so the overall data rate at which the sampling application ships events is roughly 5 * 1.177 * 500,000 * 4 bytes = 11.224 MB/s = 89.8 MBit/s, which roughly matches the value one would expect from a satisfied 100 MBit link, considering the data overhead generated by RPC marshalling.

In order to investigate the bandwidth utilization by the sampling application, Figure 44 shows the net data rate in MBit/s at which the sampling application ships event to all event channel subscribers, using different event profiles. This is actually interesting in order to evaluate the overhead generated by the RPC paradigm used in CORBA. Naturally, the overhead is the bigger, the more remote procedure calls have to be performed, as the ratio between payload and marshalled data becomes worse. This is the reason for the increase in the net data transfer rate, which can be clearly seen in Figure 43. The bigger the event size, the more beneficial it is for the net data transfer rate, resulting in the above mentioned net data transfer rate of 89.8 Mbit/s for the EB profile. Nevertheless with bigger event sizes the net data transfer rate will converge to a value slightly less than 100Mbit/s, because of the data overhead generated by TCP and MAC fragmentation.

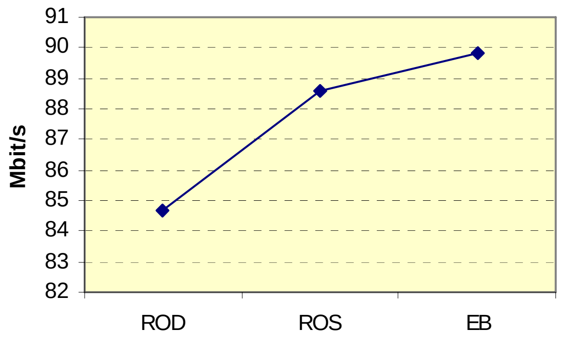


Figure 44 Accumulated throughput over 5 sampling channels for a single event sampler

9.1.4 Summary and Conclusions

As it was already mentioned, the scalability of Event Sampler with regard to the number of monitoring tasks with the same selection criteria was not proved because of the bug in the Emon package. It should be also noted that there is limitation on the monitoring task side, regarding how many pushEvent calls it can handle per time interval. This certainly is an important outcome of the Large Scale Test period and deserves some further investigation as mentioned in next chapter.

Concerning scenario B, there should be a discussion among users how many event channels per sampling application they will need. If there is a need for many channels per application, there should be some considerations on how to get rid of the bottleneck described above. On the other hand, if having only one or at maximum few channels per application is common a solution for this will be required.

9.1.5 Future Step

Regarding the bug, that prevented configuration A from delivering actual meaningful measurands, these tests should be repeated later in order to prove scalability in the number of monitors even with actual event data. Actually some tests have been performed on lxplus to do so - delivering the expected outcome - but regarding the utilisation of the lxplus cluster, receiving significant results is quite difficult.

Concerning the maximum data transfer rate of 30 Mbit/s at which monitoring tasks in the ROD profile seem to be able to handle pushEvent calls, some more investigations on the reason for this shall be made. A dummy monitor implementation receiving ROD profile event data at maximum speed, but not putting them to a buffer, not doing any locking should show whether this is a CORBA specific limitation or caused by locking overhead.

As the re implementation of the Event Monitoring framework has first been included in the tdaq-release in April 2005, there is much room for feedback and user suggested improvements. Recently, this feedback lead to an API extension, giving users using iovec memory fragments an efficient possibility to push data to the Event Monitoring framework, without having to copy data to a contiguous memory chunk. For the future, this might be another configuration being worth testing.

Currently, another focal point is the usage of smart pointers in the Event Monitoring framework, which will hopefully further improve performance and simplify memory management. It will certainly be interesting to compare test results taken with the smart pointer implementation with the results presented in this Large Scale Test report.

9.2 Information Service tests

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. Last year a comprehensive set of tests have been performed for the IS and results of these tests are described in [10].

Recently a number of new functionalities have been added to the IS. The main aim of the IS tests this year was to investigate a possible impact of these new features to the IS performance and scalability. In order to get this information some of the last year tests have been repeated with the new IS implementation.

9.2.1 Tests description and Configurations

9.2.1.1 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 part of the DAQ/HLT-I release. One can run them with the ‘-h’ flag to see their usage.

9.2.1.2 Hardware description

Slightly less then 400 computers have been used for the IS tests. One dedicated computer (dual PIV 2.8 GHz with 2 GB memory) was used to run a single IS server application. Fifteen machines (dual PIV 2.4 GHz with 1 GB memory) have been used to run 1, 5, 10 and 15 IS information receivers with only one receiver per machine. Another 350 machines were used to run from 1 to 10 IS information providers on each of them simultaneously. The slowest section of the network connection between those computers was fast Ethernet.

9.2.1.3 Tests description

Several test series with different number of the test applications have been performed. In each test sequence the following steps have been done:

1. **N** is_test_receiver applications have been started on the 20 dedicated machines, one per machine. Each of them subscribed for all information in the given IS server. Number N was set to 1, 5, 10, 15 and 20 for different test series.
2. **M** is_info_source applications have been executed on each of the 300 computers. Each of those applications created one information item in the given IS server and then updated this information 500 times once per second. **M** number was changed from 1 to 10 with step 1
3. All the is_test_receiver applications have been stopped.

9.2.2 Tests Results

Figure 45 and Figure 46 show the mean time for one information update as function of a number of concurrent information providers and receivers. Figure 45 has a logarithmic scale in order to present the whole results and Figure 46 shows the same plot in the normal scale to present the behaviour of curves in more details.

Figure 45 Mean value for one information update (logarithmic scale)

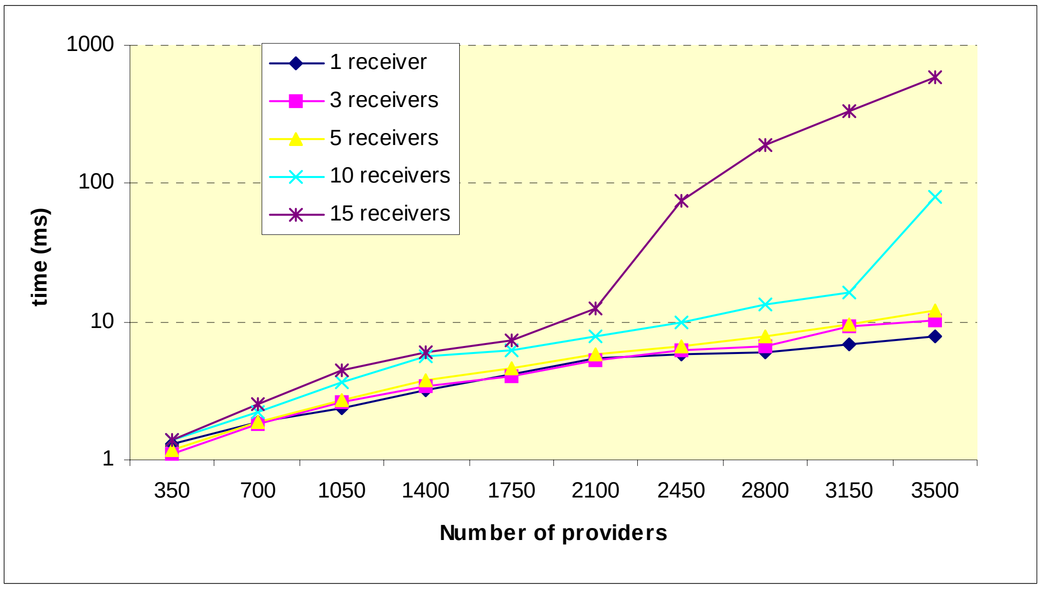
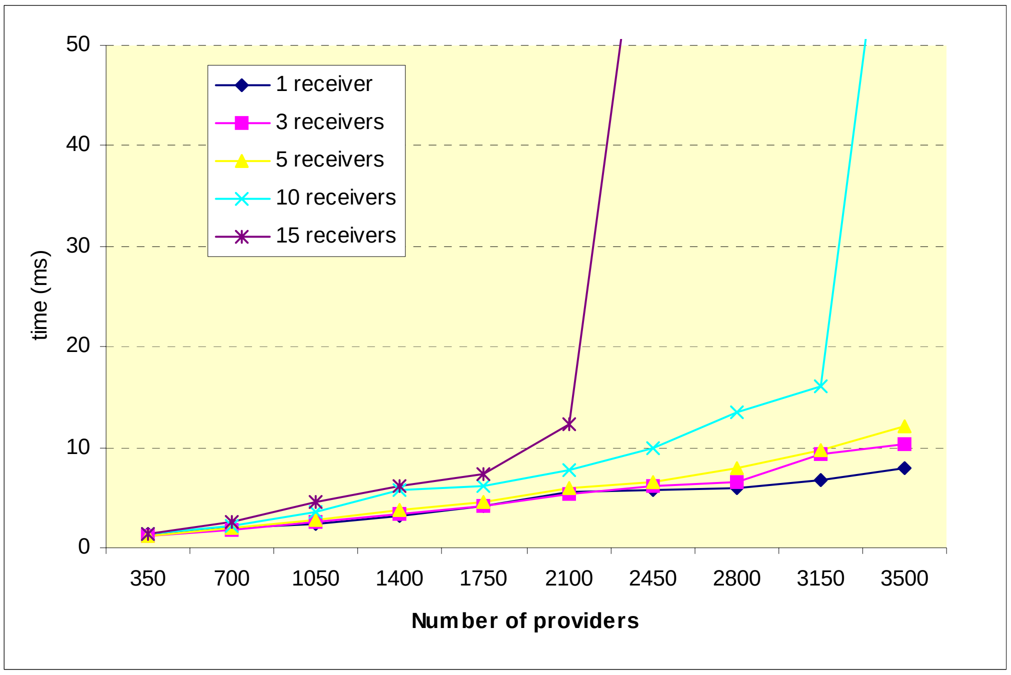


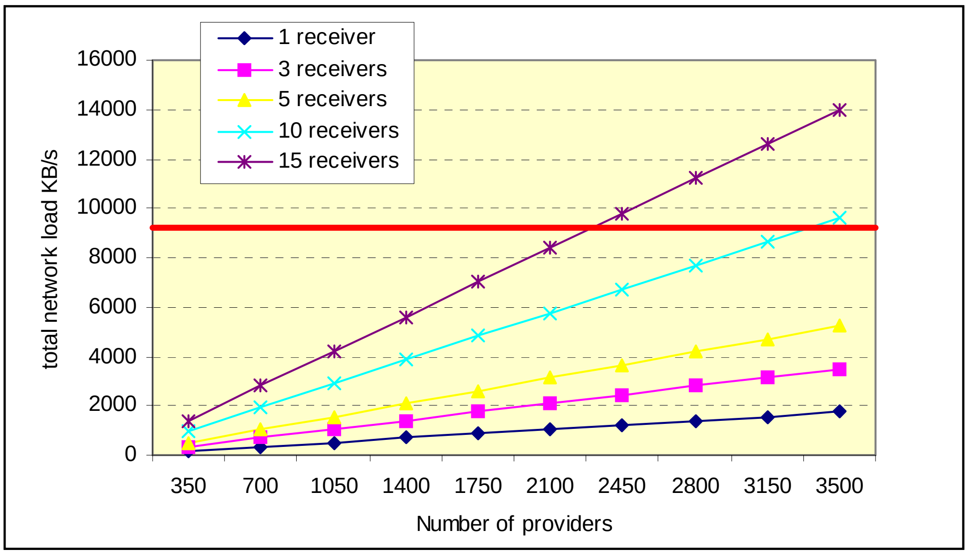
Figure 46 Mean value for one information update



For up to 1000 information providers the behaviour of the curves is liner and the mean update time is in a range of 10 ms. For 1 and 5 receivers the curves are liner up to the 2000 providers. For 10 and 15 information receivers the mean time at some point (3150 and 2100 providers respectively) shows exponential growth. This might be explained by the network throughput limit as explained in the next paragraph.

The IS information objects, which have been used for the tests have complex structure - each of them includes 13 attributes - one per each base C++ type plus one string, two attributes of the types Date and Time. In result for every update operation in the IS about 250 bytes have been transferred between every provider and IS server and also between every receiver and IS server. Using this information one can calculate the total load to the network produced during different test series. Figure 47 shows this infomation.This red line shows the maximum throughput of the fast Ethernet network (9.5 MB/s of user data plus some low level protocols overhead).

Figure 47 Total network load for different tests



It is interesting to compare Figure 47 with Figure 45. One can notice that the mean time for a single information update shows exponential growth for the configurations, which correspond to the area above red line on Figure 47. This shows that the bottleneck in the tests could be the network throughput and may indicate that for a better network (i.e. 1 GBit) one can hope getting better results.

9.2.3 Summary and Conclusions

Presented results are very similar to the ones, which have been obtained in March 2004 (during the last LST) using very similar hardware configuration. This demonstrates that a number of new functionalities, which have been provided for the IS, do not made negative impact to the IS scalability and performance.

9.3 Online Histogramming Service

The Online Histogramming (OH) service is implemented on top of the Information Service (IS). The main component of the OH is the is_server application, which has been tested as part of the IS specific tests. Nevertheless there is one important aspect, which concerns handling of very large information objects (e.g. histograms), which was not covered by the IS tests and which has been investigated during the dedicated OH ones.

9.3.1 Tests description and Configurations

9.3.1.1 Test applications

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

- is_server **-** this application implements the OH repository functionality.
- oh_test_provider **-** this application is a configurable histogram provider.
- oh_test_receiver - this application implements a configurable histogram receiver.

All of them are part of the DAQ/HLT-I release. One can run them with the '-h' flag to see their usage.

9.3.1.2 Hardware description

653 computers have been used for the OH tests. One dedicated computer (dual PIV 2.8 GHz with 2 GB memory) was used to run a single IS server application. Two computers (dual PIV 2.4 GHz with 1 GB memory) have been used to run 2 histogram receivers (with one receiver per machine). Another 650 machines were used to run a single histogram provider on each of them simultaneously. The slowest section of the network connection between those computers was fast Ethernet.

9.3.1.3 Tests description

Two test series have been performed. In the first one **650** oh_test_provider applications have been executed on **650** computers (one per node). Each of those applications created 1 histogram in the OH and then updated this histogram **500** times once for every **10** second.

In the second series **2** oh_test_receiver applications have been started on the 2 dedicated machines, one per machine. Each of them subscribed for all histograms in the OH repository. The same number of oh_test_provider applications has been used, but the time interval between histogram updates was set to **30** seconds. This has been done to make a balance of the network load between the first and second test series. The network load in the OH tests was proportional to the number of receivers because the IS server forwarded all it's incoming traffic to every receiver (OH test receivers subscribed for all the information). So that the configuration with two receivers produced 3 times more load to the network then the one with no receivers. Because of that the update interval for the 2 receivers test configuration has been increased to 30 seconds.

Two types of one-dimensional histograms have been used for different tests:

1. Histograms with short bin values (2 bytes per bin), double precision errors (8 bytes per bin) and variable axis (8 bytes per bin).
2. Histograms with floating point bin values (4 bytes per bin), double precision errors (8 bytes per bin) and variable axis (8 bytes per bin).

Histograms also had different number of bins for different tests. Table 12 summarizes all the histogram sizes, which were used in the tests.

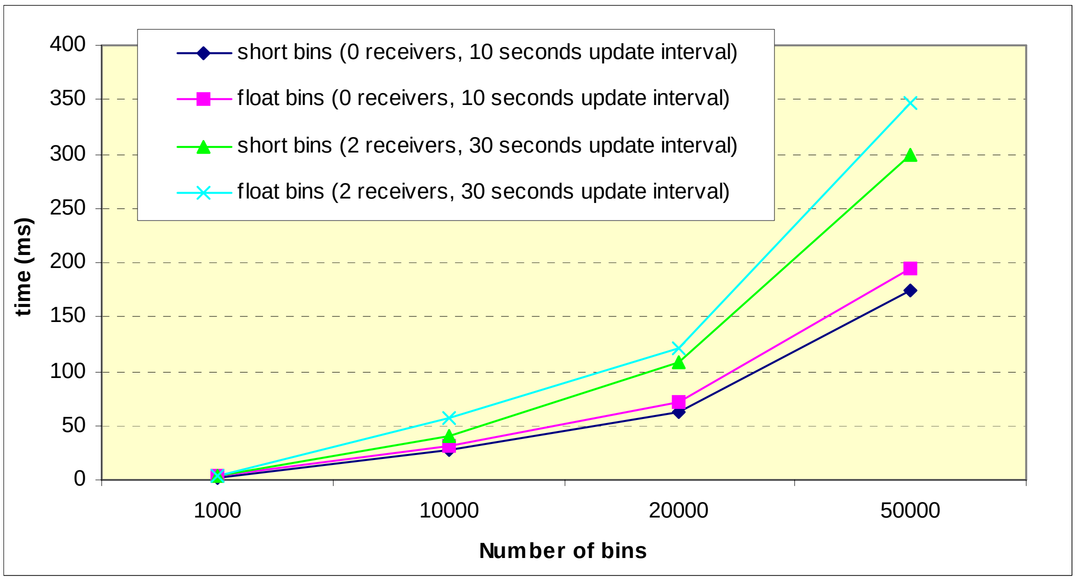
Table 12 Histogram sizes

	Histogram Size (KB)	
Number of bins	**short bins**	**float bins**
1000	18	20
10000	180	200
20000	360	400
50000	900	1000

9.3.2 Tests Results

Figure 48shows mean time of a single histogram update as function of the histogram size for all the OH test series.

Figure 48 Mean time for a single histogram update



9.3.3 Summary and Conclusions

The tests show that OH service works very reliably. No failures were observed, no information was lost event for the tests with very high network and server load. One should also take into account that tests have been done using Fast Ethernet network. The final ATLAS will use 1 GBit network, which will reduce the overhead of large histograms transportation.

Tests show that the current model of the Information Service, in which all the information is transported from providers to receivers via one or several servers, works quite well for small histograms (few thousands bins). For large histograms the time of network transfer becomes significantly higher then the overhead of marshalling and unmarshalling histograms data on the provider and receiver sides. This indicates that in case of large histograms one should try to avoid unnecessary network traffic. OH allows this by providing a special type of subscriptions. A receiver, which made such subscription, will get notified whenever a subscribed histogram is updated, but the histogram itself will be transported only in case of additional explicit request from this receiver.

For efficient use of the Online Histogramming service, it is very important to use proper histogram types. The difference between using histograms with the floating point bin values and short bin values was not so significant in the tests because the variable axis histograms have been used (see Table 12). One-dimensional variable axis histograms contain two values for each bin: the double precision axis value and the bin value itself. The difference in size between such histograms with different bin values types is not therefore very significant, The situation is quite different for the fixed axis histograms, which contain only bin values. For such histograms the difference in their size is the same as the difference between their bin values types sizes. This results in significant difference in the efficiency of the OH with respect to handling of such histograms.

In case of publishing large histograms (with more then few thousands bins) to OH one should take into account that publishing time might be significant due to the large amount of data transfer over network. The best way to cope with that problem is to have a dedicated thread for the OH publishing. Most of the time this thread will be sleeping on network data transfer without consuming CPU, which will allow other threads to do their job in parallel.

10 Databases Group Scalability Tests

The tests have been performed for the Configuration Databases using OKS with relational backend and the Conditions Database Interface using old Lisbon MySQL based and new COOL Oracle based conditions databases.

10.1 Aims

The aim of the ConfDB tests was to obtain performance and scalability results for OKS using relational backend to store configuration data. The results of the tests allow drawing conclusion about applicability of such approach for the final system.

The aim of the CDI tests was to confirm the performance and scalability of the Lisbon API based CDI, which had previously been tested in another Large Scale Tests, and measure the performance of the new COOL based CDI. From the results of these tests it would be possible to have some conclusions about the possible of the use of each system. It was also intended to check the localized performance of the CDI, trying to identify the bottlenecks and which parts should be improved.

10.2 Test description and Configurations

The CERN IT lxfs6101 Oracle server was used in exclusive mode during the tests. It is dual Intel(R) Xeon(TM) 2.40GHz processors PC with 2 Gbytes of memory and 1.2 Tbytes disks.

For the ConfDB tests several OKS data versions were created with different number of OKS objects varying from 1,000 to 256,000. Such configurations have been read simultaneously by several database clients. Each client has been started on a separate node. During each test a client established connections with the Oracle server, read required configuration including OKS schema and data, printed out read configuration and exited. The number of clients was varying from 10 to 170 (170 is maximum number of simultaneous connections for Oracle server, that was set by IT Oracle support team during the tests). For each fixed number of clients and size of configuration the test was repeated several times (10 times for small configurations, 5 times for big configurations). The mean value, the maximum value, the standard deviation and the number of errors (e.g. communication problems, wrong results of operations, etc.) are available for each test.

In addition to the above Oracle server the CDI tests used the ATLAS atlmysql01 MySQL Server in exclusive mode. The configuration of this server is the same as the Oracle Server.

For the tests it was attempted to store 1,000 objects in the database through the CDI using different methods. A small test on distributed storage was also made. In both implementations it was attempted to store several times the same object from a single machine into a single Information Service Server. There were also made tests using multiple different objects and multiple IS Servers. In order to better test the performance of the CDI, localized time measures were made at key points of the code.

As the CDI is a package that does not really depend on the frequency of data submission but on the frequency the data is sent to it from the Information Service, the tests using multiple machines didn’t provide any relevant information other than being able to execute for the frequencies tried. The tests were executed storing data at a specified frequency, which was incremented each time.

10.3 Test Approach and Execution of the Tests

The ConfDB and CDI tests used TDAQ tdaq-01-02-00 release.

To populate ConfDB database the **oks_generate_data**, **oks_put_schema** and **oks_put_data** programs from the release have been used. To read the data during tests the **oks_get_data** program from the release has been used. The tests have started simultaneously on remote nodes via ssh.

For CDI tests a couple of applications have been prepared which have been added to the CDI package. These tests had a couple of arguments that could be tuned by the user, which allowed to use the same application for different tests. The tests that were executed using several machines were started simultaneously via ssh. The objects used during the tests were the RunParams object, which is the default object stored in with the CDI, and a couple of other objects explicitly created for these tests containing all available IS data types.

10.4 Results

10.4.1 Results of the Configuration Databases Tests

Below there are results of ConfDB tests, presented as dependencies of time from the number of clients and sizes of the read information.

The Figure 49 shows dependency of mean and maximum time required to read different OKS configuration from number of database clients. For example, 10 clients can get 1,000 database objects in 1 second, 50 clients can get the same number of objects in 4 seconds and 170 clients require 13 seconds. For 4,000 objects such times are 3, 10 and 33 seconds and for 64,000 objects they are 29, 124 and 402 seconds. Note, the biggest configuration used for LST 2005 contained less than 4,000 objects and a typical one was between 1,000 and 2,000 objects.

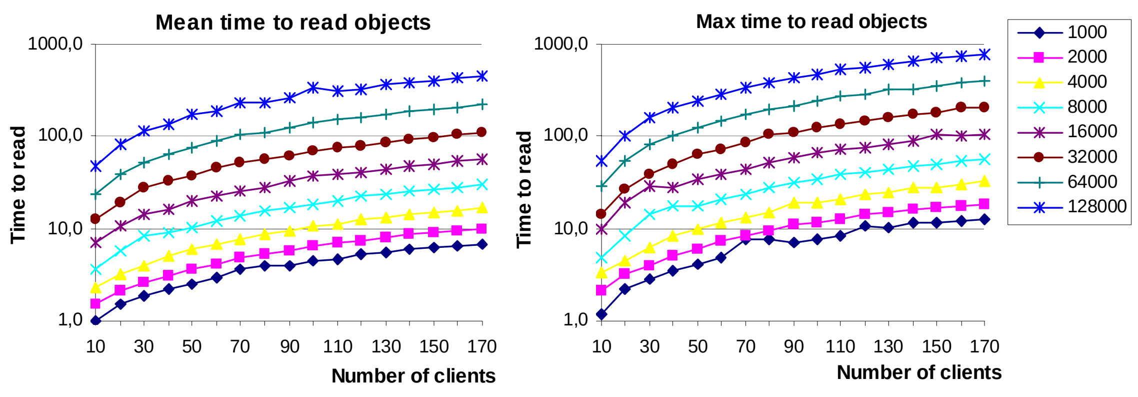


Figure 49 Mean and maximum times to read different configurations depending from number of clients.

The same data, as above are shown in different manner on the Figure 50. It demonstrates, how time to read database configuration depends on the size of the configuration for different number of clients. Note, values on both axes are in logarithmic scale and one can see, that for configurations with more than 8,000 objects there is a practically linear dependency of time to read configuration from the size of the configuration, e.g. 8 thousands objects can be read by 100 clients in 34 seconds, 16 thousands objects can be read in 66 seconds, 32 thousands objects in 123 seconds, 64 thousands in 238 seconds and 128 thousands objects can be read in 470 seconds.

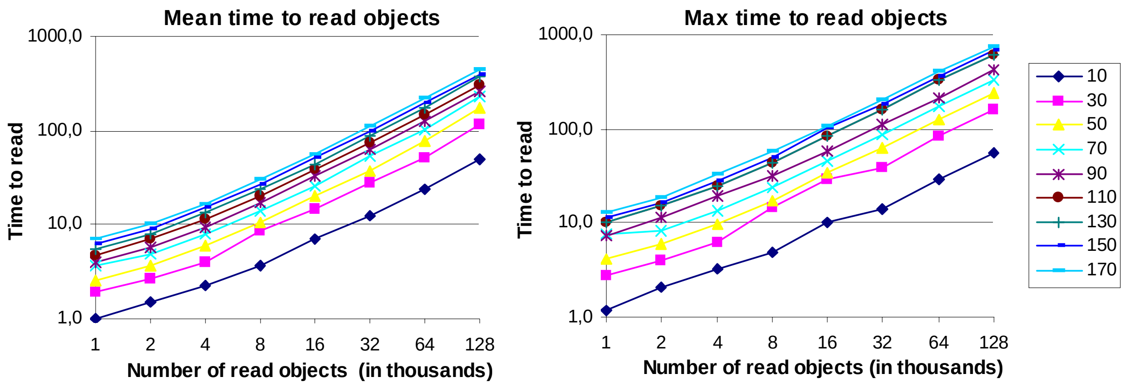


Figure 50 Mean and maximum times to read different configurations depending from size of read data

There were no errors found except for the configuration of biggest and maximum number of objects, when several times “Connect timeout occurred (attaching a server)” and “lost contact (attaching a server)” errors were reported by the Oracle library.

The standard deviation for a big number of clients and big number of configurations is quite high. Some part of database clients are “lucky” and can read configuration in a short time, while the others are waiting until some other operations will not finish. It looks like, the Oracle server allows limited number of queries to be executed in parallel and some queries are going to the queue. For example, when 160 clients are reading configuration with 128,000 objects, the fastest client got configuration in 45 seconds, while the slowest required 12 minutes and 8 seconds during the same test. Inside such time interval the distribution of results is practically linear, i.e. 10% of clients got configuration faster than 2 minutes, 19% of clients got configuration faster then in 3 minutes, 28% of clients required less than 4 minutes, 36% in 5 minutes, etc.

During the tests the Oracle server was overloaded, when big numbers of clients or big configurations were tested. For the heaviest tests the server's average load parameter has reached 25. The loading of the clients was negligible. The server's performance was a clear bottleneck during the test.

10.4.2 Results of the Conditions Database Interface Tests

After running a first instance of the tests, it became clear that the measurements using different IS servers or multiple objects were not conclusive. The CDI receives a callback from the IS server each time an object is published or updated following which the CDI processes the object and stores it into the database. In mean time, if the database structure already exists, the CDI - both the Lisbon based and COOL implementations, take no longer than 10ms to store an object, being this time independent of the fact of using a different IS Server, a different publisher and, in some extent, even a different object. It was verified that the structure of the object didn't pose great influence in the performance, except if the object contained strings. Both the processing time of the object in the CDI and the time of storage showed some fluctuations when the objects contained strings.

Hence, the following results show the measurements of time taken to execute several tasks during the processing of the objects in the CDI.

10.4.2.1 Results of the Conditions Database Interface based in the Lisbon API

This implementation has already been tested in a previous Large Scale Tests in which its performance and scalability have been confirmed. Thus, this new series of tests were basically to test the minor changes applied in the Lisbon API of the Conditions Database and to use these results as a compare with the ones from the COOL based CDI.

All the tests with this implementation worked without any frequency constraint for publishing data into the IS. This implementation of the CDI takes approximately 6ms to store each object, with just a minimum fluctuation in case that the object contains strings.

The results for each task during an object storage as presented in the Table 13:

Table 13 Time results for the Lisbon based CDI

	Creating a new DB	**Using an existing DB**
CondDB Initialisation	42ms	27ms
IS object subscription	28ms	28ms
Folder creation	37ms	-
Loading folder information	-	10ms
Fill the CondDBTable	0.38ms	0.38ms
Storing the CondDBTable	6.58ms	6.58ms
Overall mean storage time	7.57ms	7.57ms

The first two rows are initialisation measures, which obviously differ depending if the database already exists or not. The CDI takes longer to initialise, but this does not impose a problem as at initialisation time it is not supposed to be taking data yet.

The folder creation is only executed either if the object to be stored was never stored before or the database has just been created. This step is not in the initialisation process but is done only once, as the information is kept in memory.

The task of loading the folder information is executed if the folder already exists. Although it is not in the initialisation step, it is executed at the first object storage and the information is then kept in memory for faster access.

From the times measured, it became clear that the bottlenecks in the Lisbon based CDI execution are the tasks which require access to the database. Although the load in the client machines, including the machine running the CDI, was very low (never above 3%), the load in the database server was much higher. During the tests were observed peaks of almost 30% CPU execution.

This implementation of the Conditions Database is being discontinued, so there is no need for major improvements. However, a possible improvement could be the improvement of the algorithm that is responsible to load the information about a folder, as a user may subscribe a folder during the CDI execution, which will impose an additional 10ms penalty for the object storage.

10.4.2.2 Results of the Conditions Database Interface based in COOL

This implementation was first tested during LST, and being this the implementation that will be used for future, these tests are of extreme importance to observe the applicability and future improvements.

Not all tests with this implementation worked as expected. There were some frequency constraints, which are related with a problem in Oracle server, which conditioned the results.

The time measurements result for this implementation are presented in Table 14:

Table 14 Time measurements of the tasks executed by the COOL based CDI.

	Creating a new DB	**Using an existing DB**
COOL Initialisation	1223ms	675ms
Verification of database existence	587.87ms	586.55ms
IS object subscription	28ms	28ms
Creating a folder	4339ms	-
Loading folder information	-	34ms
Generate AttributeListSpec	0.054ms	0.053ms
Fill the AttributeList	0.178ms	0.184ms
Storing the AttributeList	20-100ms[a]	20-100ms
Overall Storage Time	53.57ms	54.85ms

a. It was measured that the first time the object is stored it takes around 100ms to store, while the subsequent objects take a mean of 20ms.

Like in the results from with the Lisbon API, the first two rows represent the time taken by the main task in the initialisation step of the CDI. The results show a much higher time, especially in the verification if the database already exists.

This difference in the verification of the database existence can be explained by a difference in the API specifications. While the Lisbon API has a way to verify if a database already exists in the database, in the case of COOL there's need to use a workaround, which consists in trying to open a database and catch the exception in case it does not exist, returning then the result. A portion of the code is represented below:

```
try {
  cool::IDatabasePtr db = m_dbSvc->openDatabase(dbProfile);
  // this code is necessary to tell the smart pointer to free the
  connectioncool::IDatabasePtr dbNull;
  db = Null;
  return true;
}
catch (seal::Exception &e) {
  // this code is necessary to tell the smart pointer to free the connection
  cool::IDatabasePtr dbNull;
  db = Null;
  return false;
}
```

This procedure takes a long time because the exception handling by the SEAL package, in which COOL is based, is a slow procedure.

Another reason for the difference in time for the initialisation id that, since COOL is based in the SEAL framework, there are a lot of plug-ins that are loaded only as needed, which is also a slow procedure.

Like in the case of the Lisbon API based CDI, the bottlenecks are always related with database activities. In particular the case of the folder creation is quite slow due to an open bug in the Oracle Server, in which the creation of tables requires the execution of the application to "sleep" for some time with the risk of crashing if ignored.

Comparing the time results of the implementation, one can see that in general this implementation is not as fast as the Lisbon API based implementation, however, COOL is still a project under intense development an further improvements have been made meanwhile.

The big difference in time for storing an AttributeList, the specific object used by COOL, for the first time, when compared with the subsequent storages, is, like in the case of the initialisation, slowed down by the loading of plug-ins needed.

This issue is critical as it must be done during data taking and there is no way to displace it to an initialisation step. To address this there is a task to implement bulk object insert, which will allow the objects to be pooled in the CDI and stored in bulks of as many objects as specified.

During the execution of these tests the load of the client machines was quite low (seldom higher than 5%), however the Oracle Server was continuously overloaded, which caused problems when storing objects at a high frequency.

10.5 Summary, Conclusions and Future Steps

It is expected, a proper organization of the configuration databases will require an order of 10 ... 20 thousands of database objects to describe all configurations for final ATLAS system. With the current approach a single Oracle server can only be used via intermediate rdb servers, which are accessed by the user’s processes reading configuration and play role of cache. The total number of required rdb servers varies from 30 to 100 (i.e. about total number of racks). In this case the database can be read by all rdb servers in time interval from 15 seconds (minimal size and number of rdb servers) to 1.5 minutes (maximum size and number of rdb servers).

There is a small room to improve performance of OKS relational backend via optimization of Oracle tables and OKS code using RAL. This can give some performance improvement, but will not change it dramatically. A good performance improvement can be reached with usage of RAC Oracle servers that may give linear performance improvement for read-only database access with growing number of server nodes.

The CDI is expected to store a series of objects, subscribed by the users, which need to be stored into the Conditions Database.

As we have seen from the test results, the current implementation copes with the performance of the packages it based on. The Lisbon implementation tests showed that there is still some improvement needed in the COOL based CDI, as there are some bottlenecks that may impose restrictions at a higher rate of object publishing.

There are a couple of improvements that may be done in the COOL based CDI, in particular concerning the performance of object storage, which may be improved using object bulk insertion. This feature is already available in the current version of COOL and may improve considerably the performance. However this improvement may require some dramatic changes in the structure of the CDI code.

11 Controls SubSystem Scalability tests

This chapter summarizes the results obtained during testing of individual components developed within the Controls WG: the Access Manager, the Log Service, the Integrated Graphical User Interface (IGUI), the Diagnostics and Verification System (DVS), the setup component and the Run Control.

11.1 Access Manager Scalability Tests

The Access Manager (AM) is a component of the TDAQ system responsible for authorizing users based on their rights in the system and to stop or allow actions based on the result of this authorization. The underlying access control model used by the AM is the Role Based Access Control (RBAC) model. The current implementation of the AM is based on a server-client model where the server has access to users and rights information and also decides if an action is allowed or not, while the client is represented by CORBA interceptors which may allow or deny a CORBA call depending on the authorization result provided by the server.

11.1.1 Aims

Given the performance constraints imposed to the TDAQ system in general, the goal of the AM tests is to measure the performance impact, namely, the execution delay when CORBA calls are authorized using the AM component. More specific, given a large partition (might be purely dummy or contain real DF applications) evaluate the performance loss at setup and transitions introduced by switching on the access management.

11.1.2 Test description and Configurations

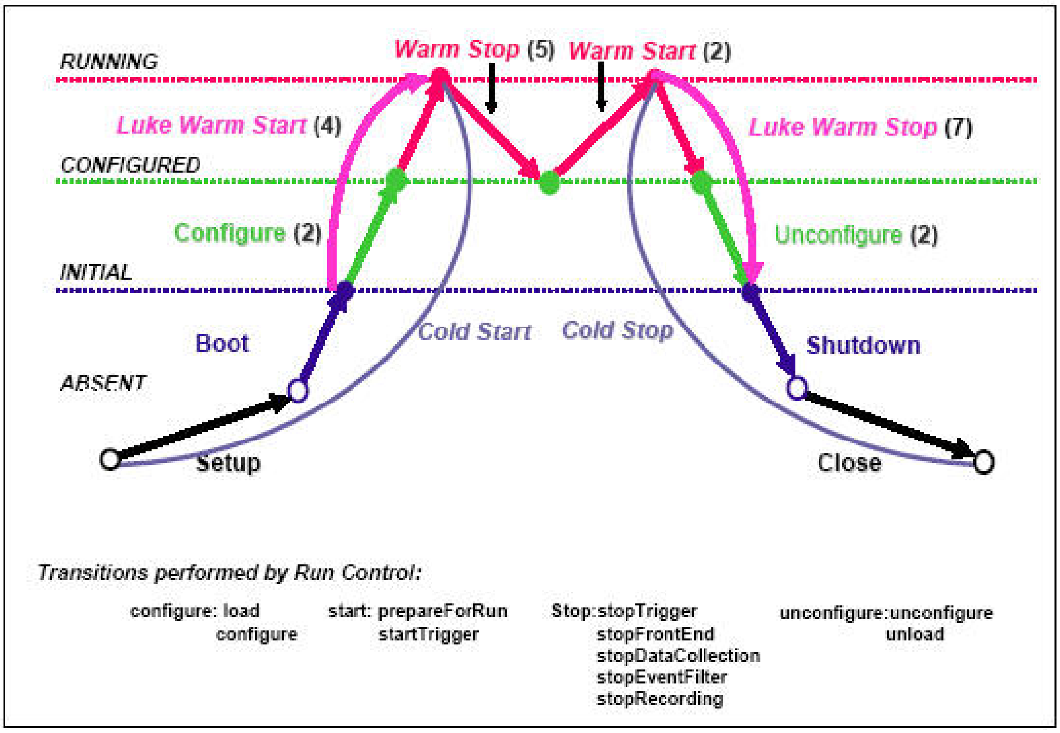
The configurations needed prior to use of AM component consist in setting up a MySQL database server and populate a database with information about users in the system, roles and actions allowed to each role. The users' names were the used AFS accounts, roles can be *observer*, *shifter* and *expert*, and actions are represented by IDL function names. The *amViewer* tool can be used to add, modify and delete information from the database.

Basically, the AM performance test consist in running a partition with AM deactivated and the with AM active. Time differences for booting and transitions shall be recorded. If feasible, the AM should be then always kept active. This test should be repeated with partitions of different sizes to observe how performace loss scales with the partition size.

Since functionality will not be tested, the above tests were run using a user set to *expert* role in AM database.

11.1.3 Test Approach and Execution of the Tests

Partitions of different sizes with control tree only have been used: 100x0x0_tcp, 200x0x0_tcp, 300x0x0_tcp, 400x0x0_tcp, 500x0x0_tcp. Each partition type has been run 10 times with AM component disabled and 10 times with AM component enabled, each time in automatic mode without IGUI and state transitions have been timed. The default timing test have been run (see below the state transition graph).

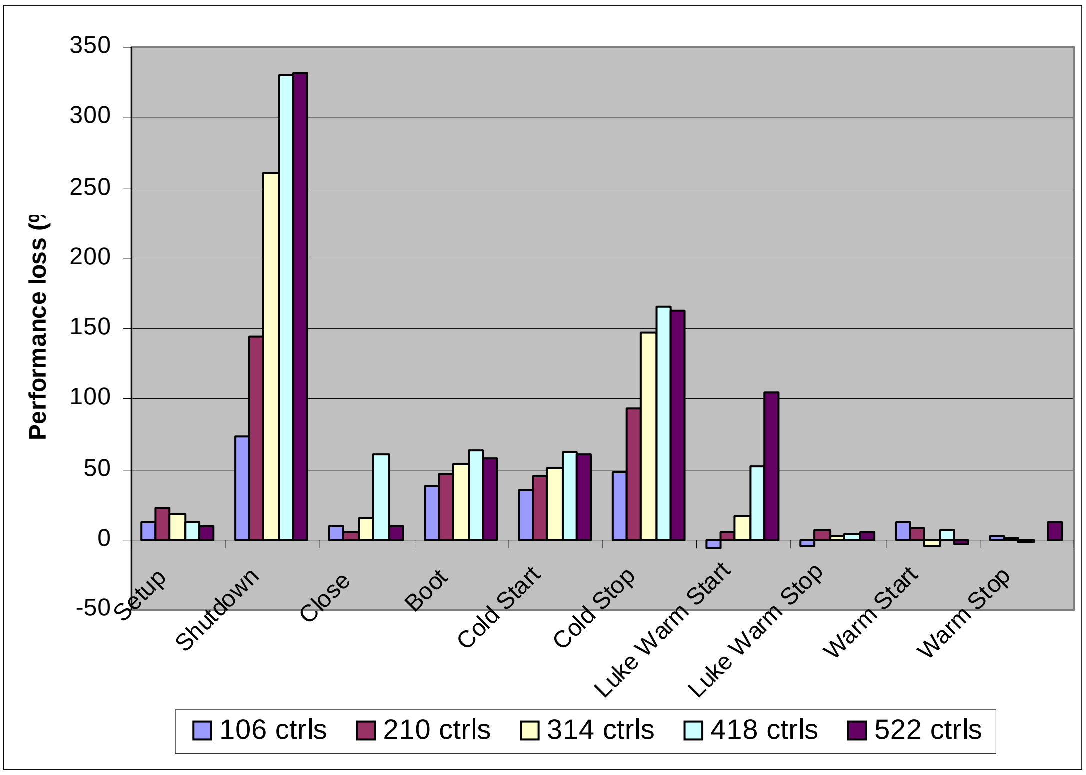

RUNNING
Warm Stop (5)
Warm Start (2)
Luke Warm Start (4)
Luke Warm Stop (7)
CONFIGURED
Configure (2)
Unconfigure (2)
INITIAL
Boot
Cold Start
Cold Stop
Shutdown
ABSENT
Setup
Close
Transitions performed by Run Control:
configure: load configure
start: prepareForRun startTrigger
Stop:stopTrigger stopFrontEnd stopDataCollection stopEventFilter stopRecording
unconfigure:unconfigure unload

11.1.4 Results

The average execution time growth with AM enabled was ~30%. The table below summarizes the performance loss in % for each state transition when AM has been used.

Num. Ctrls	Num. Hosts	Setup	Shutd own	Close	Boot	Cold Start	Cold Stop	Luke Warm Start	Luke Warm Stop	Warm Start	Warm Stop	Total Time
106	115	13	73	9	38	35	48	-6	-4	13	2	21
210	217	23	144	6	46	44	94	5	7	8	1	32
314	321	18	260	15	54	51	147	16	2	-5	-2	32
418	392	12	330	60	64	62	166	52	4	7	0	33
522	392	10	332	10	58	61	163	105	6	-3	13	32

Following chart represents the performance loss on each partition type on each run control transition:



11.1.5 Summary and Conclusions

A performance loss was expected and confirmed by the tests. The big performance loss observed has the explanation in the intensive exchange of CORBA messages during initialization, termination and transition between states. Nevertheless, it should be noted that the AM does not affect the performance of data taking since it only impacts on CORBA based operations. CORBA interceptors, that enforce authorization policy, have an internal caching mechanism for authorization results returned by the server and the performance can be improved increasing the value of the cache timeout. Security aspects areteh other component which has to be taken into account, to tune the best value of the cache interval.

11.1.6 Future Steps

The development of AM component is focused mainly on extension of the functionality (integration of XML technology and usage of LDAP for user information), but performance aspects have to be kept in mind all the way. The test results give some indications about states and transitions were the performance loss is significant and the investigations and future improvements should be focused on that points.

11.2 Log Service Scalability Tests

11.2.1 The Log Service

The Log Service is the component in charge of collecting, saving and archiving all information which needs to be logged in the TDAQ system. It also provides the means to retrieve, display and remove log information based on requests from the user.

Figure 51 depicts the overall picture of the Log Service. Although the Log Producers are conceptually part of the Log Service, their implementation is not dependent on this package.The ERS provides the logging front-end for the TDAQ applications and uses the MRS as the underlying transport mechanism. These services perform the actual qualifying of the Log Message.

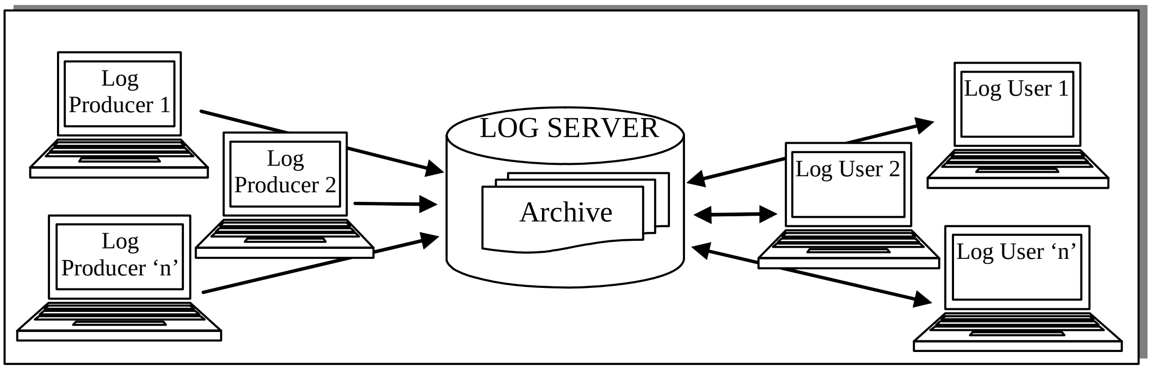


Figure 51 The log Service architecture

The Log Service can be divided into two different elements, namely the Log Server and the Log Manager. The former includes the Log Service C++ API and the Log Receiver, whilst the latter provides a web-based user-friendly interface to the Log Server database. The following bullet-point list summarizes the Log Manager functionality:

- Search Criteria/Filter - qualifier-based criteria applied to the database query.
- Message View - this feature allows the user to view messages that meet a given search criteria.
- Local archive - this feature allows the user to locally store messages that meet a given search criteria. Although this option implies message duplication, it may be useful if logs are to be 'transported' (email, removable disk, etc.).
- Delete - message removal based on a search criteria.

11.2.2 Aims

Two sets of test were planned for the Log Service. The aim of the first test is to verify the correct functionality of the Log Service and its capability to archive and later retrieve (by means of the Log Manager) the Log Messages. The second set of tests is aimed at studying the behaviour and constraints of the Log Servers.

11.2.3 Test Description and Configurations

11.2.3.1 Hardware Description

The nodes lxb0601 and lxb0602 were allocated as Log Servers. Both machines have the same specs; 2 CPU PIII 1GHZ FASTETHERNET.

Figure 52 outlines the data flow. It reflects the idea that the Log Managers can connect to the Web Server and/or the Log Server on either node. Also, the Log Producers can send the Log Messages on either Log Server database depending on the logging policy.

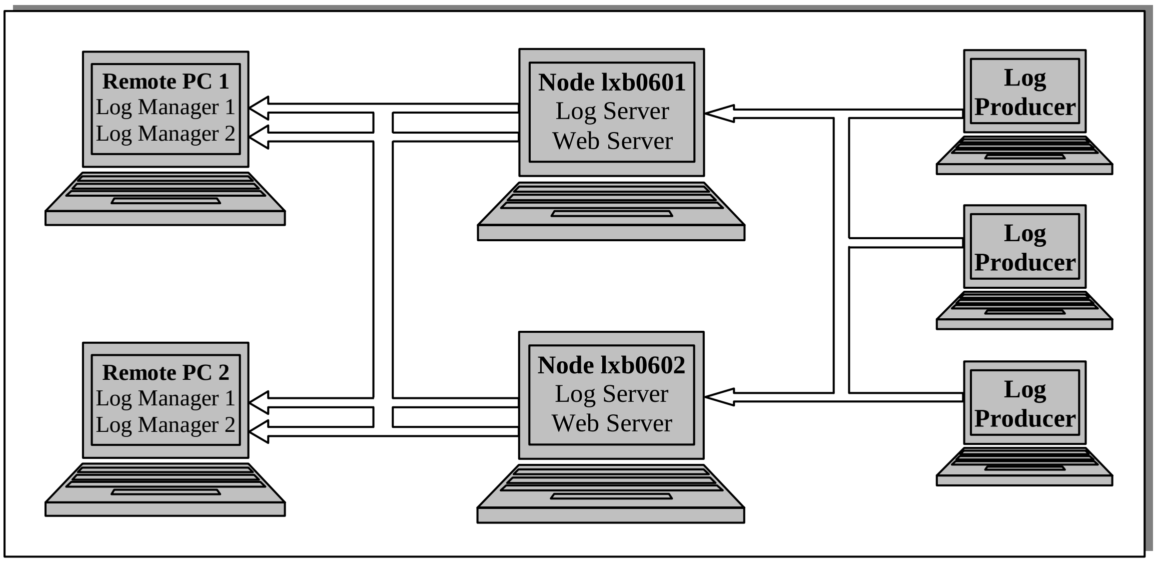


Figure 52 Flow of data within the Log Service.

Metrics of the Log Server nodes and the MRS server have been retrieved from the CERN LEMON monitoring repository. These metrics include the CPU utilization and the network I/O rates. The metrics sample rate was reduced to 15 seconds from the default 5 minutes to obtain a better resolution.

11.2.3.2 Test Application

The Log Service release includes the logReceiver binary, in charge of collecting and archiving the Log Messages. The Log Producers correspond to standard applications from the Online Software release. An instance of the MySQL Server and the Apache Server has been installed in each Log Server.

One of the nodes has been configured to log messages with severity level WARNING, ERROR and FATAL, whilst the second has been configured to log messages with severity level SUCCESS, INFORMATION and DIAGNOSTIC. The commands below show this setup.

```
Node 1:
        logReceiver -p $TDAQ_PARTITION -c 'WARNING|ERROR|FATAL'
```

```
Node 2:
       logReceiver -p$TDAQ_PARTITION -c 'SUCCESS|INFORMATION|DIAGNOSTIC'
```

Where $TDAQ_PARTITION is an environment variable holding the name of the partition associated to the MRS server.

The logging policy has been defiend in the lsServersTable table. Each entry in this table defines the IP, the host name and the logging policy of each Log Server.

11.2.4 Results

The first test has been run in parallel with a full combined partition of the TDAQ created by Kostas Kordas. A total of 390 nodes were employed. The run has lasted for over 37 minutes until a problem outside the scope of the Log Service occurred and the partition needed restarted. The figures below show the nature of the messages received:

```
FATAL                    1
ERROR                    796
WARNING                  2537
DIAGNOSTIC               0
INFORMATION              160744
SUCCESS                  0
TOTAL                    164078

AVERAGE                  lxb0601        1.47 msgs/sec.
AVERAGE                  lxb0602        71.63 msgs/sec.
```

The first test has proved the correct functionality of the Log Service under normal conditions, that is, under a TDAQ partition and a low message transmission rate. After the test, the messages could be browsed and delete with the Log Manager.

For the second set of tests a more controllable environment was needed. This was achieved by modifying the Run Control package to produce a deterministic rate of outgoing messages. To test the influence of the Web Server on the Log Server and its capability of receiving Log Messages, the Log Manager has been employed to consult the database. Table 15 shows the different actions taken during the test run. These actions encompass all the possible combinations of Web Server and Log Server access. Normal Receiving Mode defines the time slots during which message retrieval is null.These time slots measure the behaviour of the Log Service on receiving mode alone. For this test up to four Log Managers were opened on two remote machines to cover all the possible combinations highlighted in Section 11.2.1.

The partition created for this test run includes 1331 application, each issuing a WARNING and an INFORMATION message every second. Each Log Server has therefore recorded 1331 Log Messages per second.

Unfortunately, soon after the Large Scale Tests concluded, the nodes in the cluster were reformated and the databases removed. Although the actual information on the messages was of no importance, the corresponding timestamp of each entry would have been crucial to determine the actual incoming rate.

Figure 53 shows the CPU utilization and Network I/O for the Log Servers (lxb0601 and lxb0602) and the MRS Server.

Table 15 Actions taken during the test run.

Time (hh:mm)	ACCESS TO: Web Server		Log Server (DB)		Description
	lxb0601	lxb0602	lxb0601	lxb0602	
14:58-15:10	No	No	No	No	Normal Receiving Mode
15:10-15:16	Yes	Yes	Yes	Yes	2 remote browsers connect to the web server on lxb0601 and lxb0602. Both access the Log Server 0xb0601.
15:16-15:20	No	No	No	No	Normal Receiving Mode
15:20-15:25	Yes	No	Yes	Yes	2 remote browsers connect to the web server on lxb0601. Each access one Log Server.
15:25-15:30	No	No	No	No	Normal Receiving Mode
15:30-15:35	Yes	No	Yes	No	2 remote browsers connect to the web server on lxb0601. Only the Log Server on lxb0601 is accessed.
15:35-15:40	No	No	No	No	Normal Receiving Mode
15:40-15:45	Yes	No	No	Yes	2 remote browsers connect to the web server on lxb0601. Only the Log Server on lxb0602 is accessed.
15:45-15:50	No	No	No	No	Normal Receiving Mode
15:50-15:55	Yes	Yes	Yes	Yes	Sporadic access to both Web Servers and Log Servers
15:50-15:55	No	No	No	No	Normal Receiving Mode

During the time interval 14:58-15:10 the results for lxb0601 and lxb0602 are fairly similar. Both have recorded a CPU utilization of around 35% and a data reading rate of 350 kB/sec, which accounts for the 1331 Log Messages received every second. The data written on the network corresponds to MRS acknowledgments. As expected, the MRS Server reading rate is over 700 kB/sec, since 2662 messages are issued per second.

The subsequent peaks in Figure 54.a and Figure 54.c (15:10-15:16) depict the use of the Log Manager to consult messages from lxb0601. This involves a connection to the Apache Web Server and access to the MySQL server, which explains the fairly large CPU utilization (nearly 100%). The node lxb0602 presents less prominent CPU utilization peaks since its database is not accessed. This also explains the peaks around 15:12-15:15 in Figure 54.e and Figure 54.f, as a result of data being received and passed on to the Log Manager.

The second cycle of normal receiving mode (15:16-15:20) records practically zero CPU utilization and network access in lxb0601. This sudden drop is unexpected and illustrates the problem found during this test; Log Messages are no longer archived after a period of full CPU utilization.As a consequence, the MRS Server writing rate (Figure 54.i) is reduced by 350 kB/sec. Subsequent CPU utilization and network I/O peaks in lxb0601 are associated to the Log Manager.

Note that whilst lxb0601 fails to archive Log Messages after 15:10, lxb0602 continues to function normally. Only after a nearly full CPU utilization (15:40) does lxb0602 cease to store messages, as observed in Figure 54.d and Figure 54.e. Again, the MRS server transmission rate decreases by 350 kB/sec.

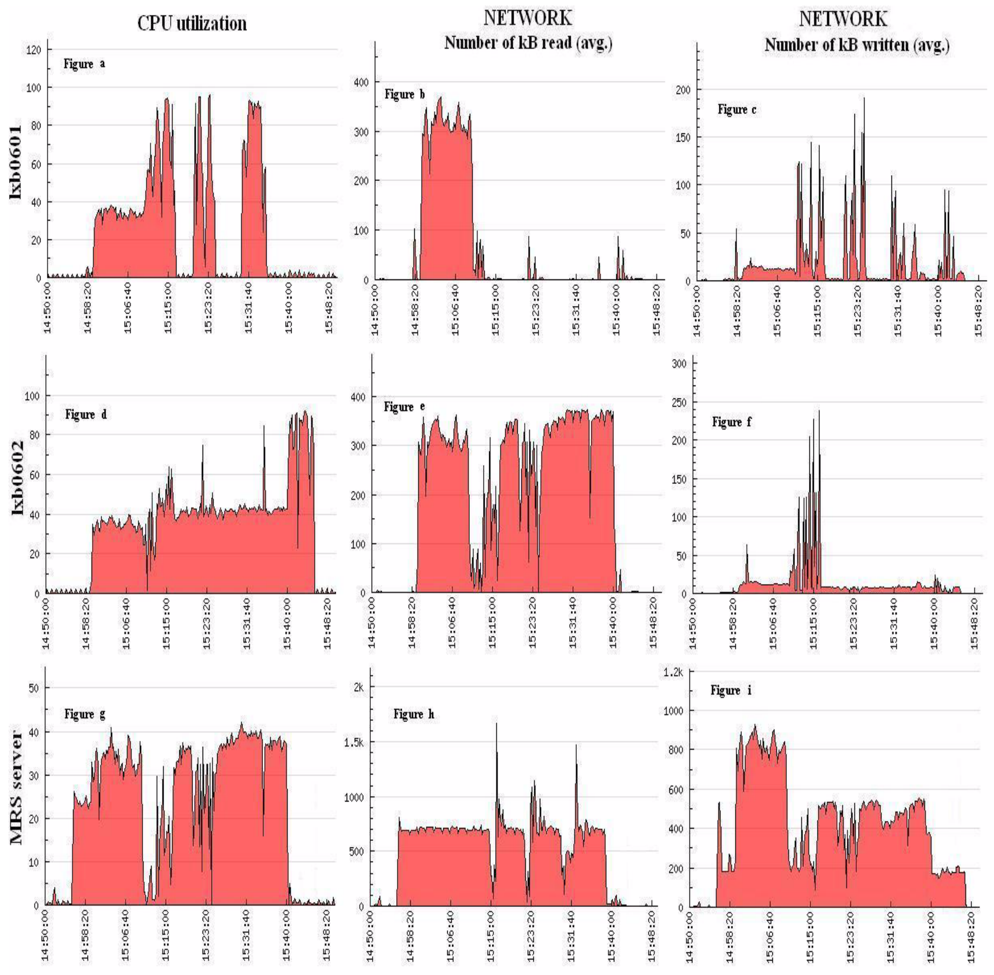


Figure 53 Measurement results as a function of time

The explanation for the problem described above lies in the MRS Service design. Every time the Log Manger webpage is refreshed, the Web Browser establishes a connection to the Apache Server. In turn, each connection creates a thread that consumes CPU bandwidth and prevents the CORBA thread(s) from logging of messages onto the database. The MRS Server attempts to alleviate this situation by using up all its potential connections. Iironically, this worsens the problem since the MRS server is blocked thus preventing new messages from being transmitted. After a timeout and no reception of acknowledgments, the MRS Server automatically unsubscribes the Log Receiver and resumes normal operation. Of crucial importance is the side effect incurred in the Log Producers, which remain blocked and the application halted until the timeout expires. At this point the data stored on the network buffers is transmitted, which explains the peaks in the MRS reading rate. This also explains the sudden drop of the MRS Server metrics values (Figure 54.g and Figure 54.h) when the Log Manager is used.

11.2.5 Summary and Conclusions

Unfortunately, due to problems encountered with the MRS service, the Log Service tests have not accomplished the most important goal, namely to study the performance and limitations of this package. The outcome of this excersie is however not all fruitless since serious issues with the MRS service have been pinpointed. The system can be forced to halt the MRS Server and the Log Producers. This requires a review of the unsubscribe timeout value and of the Log Server configuration. Either the Apache and MySQL server need decoupled, which may in fact not alleviate the problem, or the number of connections to the Apache Server must be limited to prevent an overuse of the CPU.

Once these issues are addressed, further tests shall be carried out to study in more detail the Log Service package.

11.3 IGUI Scalability Tests

11.3.1 Aims

The Integrated Graphical User Interface (IGUI) is intended to present the status of the data acquisition system and its sub-systems 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.

11.3.2 Test description and Configurations

During the large scale tests, the IGUI has been used to verify the new generated partitions, before running the automated performance tests. The tested partitions had different configuration sizes, up to the maximum configuration with 700 computers and 2000 controllers and controlled applications. In this way the IGUI was used for larger configuration than in the 2004 large scale tests, when IGUI was tested only for a dedicated test partition with 110 controllers on 110 computers [10].

11.3.3 Test Approach and Execution of the Tests

For most of the new generated partitions, the IGUI was used to monitor and control the partition. After the infrastructure setup, a complete “data taking” cycle (Boot, Configure, Start, Stop, Unload, Shutdown) has been exercised. IGUI was running on one of the fast computers with large memory, with a remotely display. The functionality for normal operation has been verified, but also the IGUI behaviour in some error conditions (the recovery from some error conditions, the messages display for high rate of error or warning messages).

11.3.4 Results

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. The limitations observed in 2004 large scale tests in the DataFlow panel on the representation of DataFlow information as charts are still present, especially for a remotely display. The DataFlow panel was used mainly with the option “Turn dataflow graph off”.

Even for largest partition, it was possible to control the DAQ system from IGUI, with an acceptable IGUI overhead.

The usage of IGUI by many testers provided several proposals for improvement:

- in the RunControl panel, display also the information on the status of controlled items (applications, resources);

- in the ProcessManager panel, use a lexicographic order for the display of the agent and process lists;

- in the RunControl panel, examine the possibility to avoid the display of an incorrect information in the case of an incorrect order of the IS callbacks;

- review the font size and the display layout for some information where the number of the displayed items depends on the configuration size, for example the list of agents in the Infrastructure panel;

- change font style (from “italics” to “normal”) for the expressions of subscription criteria or filter in the MRS panel.

11.3.5 Summary and Conclusions

The IGUI can be used for control and status display for a large configuration.The feedback received will be considered for the future releases. The implementation for the display of the information as charts in the DataFlow panel should be reviewed.

11.4 DVS Scalability Tests

The DVS component is responsible for the verification and the diagnostics of the whole data acquisition system, including software and hardware components as defined in the configuration database. It also performs tests on specific components of the TDAQ system which are not described int hte databse, e.g. the pmg_agents.

DVS is using the TestManager to launch tests defined in the TestRepository database.

The aim of the tests was to measure the time to initialize the DVS tool and to test different configurations: from simple to complex (in terms of controller tree), from small to large (in terms of number of nodes used). Only pure online configurations were used in these tests.

The following times were measured:

- start_up: time to bring up the DVS GUI (quantity dominated by reading the database configuration)
- test hardware (composed only by Computers in the configurations used)
- test controller tree
- test PMG agents: time to test the pmg_agents (it assumes the pmg_agents are up and running)

Different partition configurations were used:

TO ADD: explanations on the different configuration structures.

- type A partitions (denoted PartA in Figure 2)
- type B partitions (denoted PartB in Figure 2)- the best structure for the controller tree
- type C partitions (denoted PartC in Figure 2)

The maximum number of nodes used was 698.

11.4.1 Tests Results

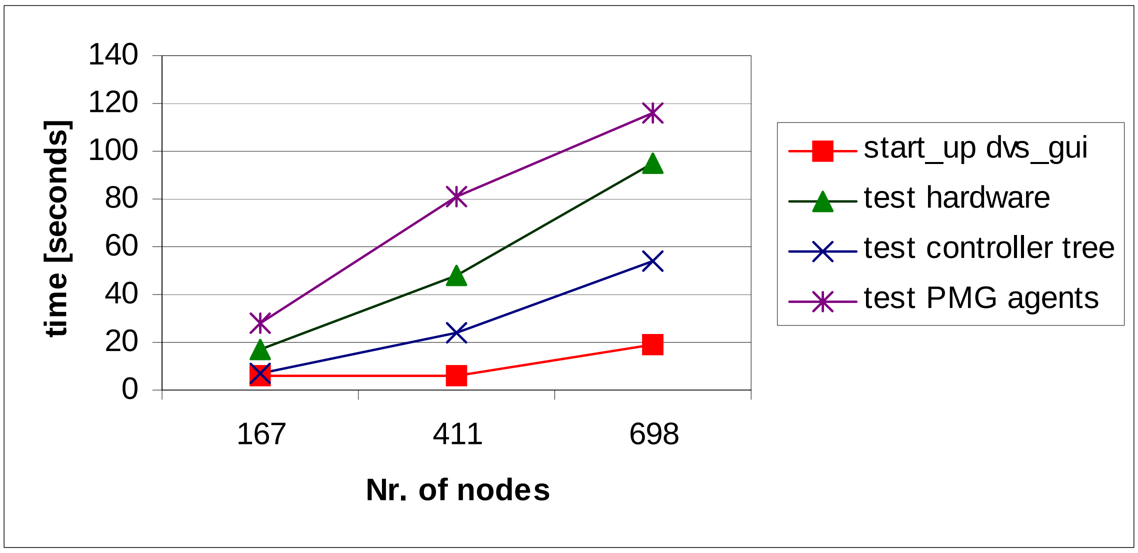


Figure 54 Timing tests for configurations using different number of nodes

As expected, the results in Figure 54 show a good scalability of the system going up to the maximum number of nodes available for the LST measurements.

Figure 55 shows timing measurements for different types of controller tree. On x axis in paranthesis are given the number of applications in each partition used. The same number of nodes is used for all points of measuments (698 - the maximum).

From Figure 55 one can see that configurations of ‘type A’ are to be avoided. The controller structure in this case is too heavy, the time to test such a tree is much larger (about 5 times) than for the other configurations. In general, all quantities measured have larger values for configurations ‘typeA’.

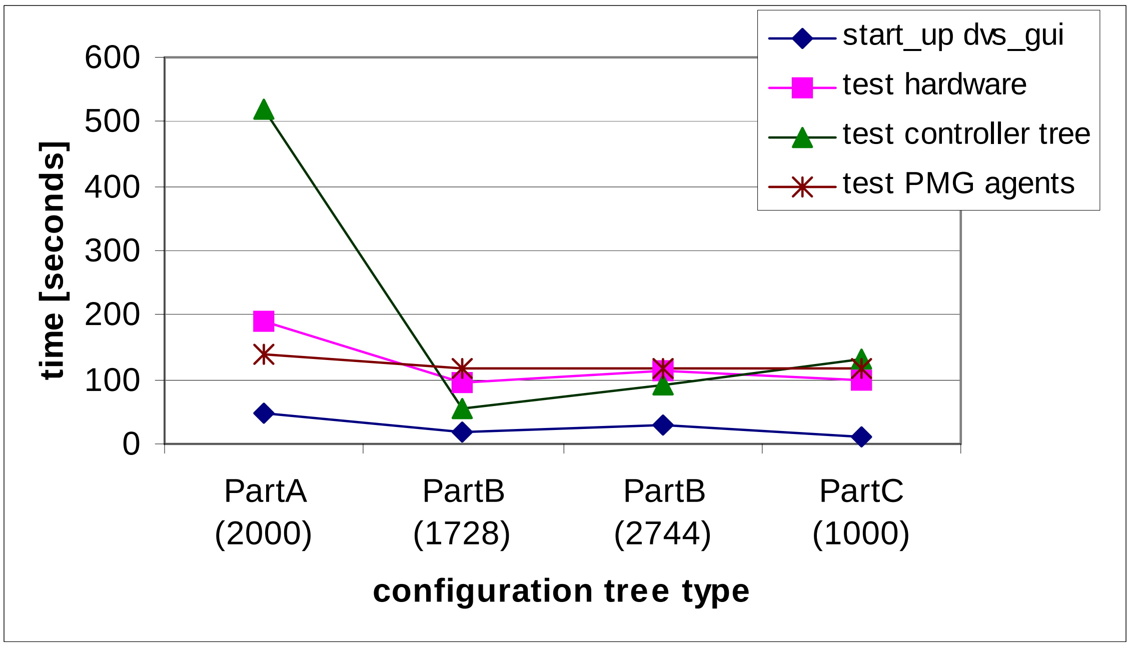


Figure 55 Timing tests for different configuration types

11.5 Setup component Scalability Tests

11.5.1 Aims

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

To verify the presence and functionality of the components (Applications, PMG agents, Computers) Setup launches dedicated tests which are defined in Tests Repository database.

The aim of Setup scalability tests are to estimate performance and scalability of the component (and other components it depends on) on the partitions of a large scale.

11.5.2 Test description and Configurations

Partitions of different sizes, from 50 to 600 controllers, using 50 to 390 nodes were generated. Setup component was used to launch these partitions (without entering into RC cycles) in automated (non-interactive) mode, and different time intervals were measured. Each partition contained Online Infrastructure applications plus a number of RDB servers (one per 20 controllers) which were also launched by the setup component. The Infrastructure segment (16 applications in total) was running on ten dedicated machines, while RDB servers were distributed over the cluster along with Run Controllers. The setup server was also launched on a dedicated machine.

The used command was

```
> setup_daq <partition>.data.xml <partition> -nt
```

where <partition> is one of the generated partitions, substituted in the cycle. Each partition was running 5 times and timestamps of the following setup stages were recorded:

- *Cold* setup: time from scratch to the "Initial"[1] 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 and partition-independent applications are running, only set up of the partition infrastructure is done
- *Testing*: all components are running, only testing of agents and infrastructure is done. (This is an emulation of "recovery" procedure, where a single problem can be identified by means of testing of the whole partition.)
- *Shutdown*: stopping running applications and launching 'shutdown' ones.
- *Agents Boot*: Time to start only PMG agents (without testing). This time was not measured explicitly, it is calculated from other measurements (Cold Setup minus Warm Setup time).
- *Infrastructure Boot* time: Time to launch infrastructure applications, without testing. This time was not measured explicitly, it is calculated from other measurements (Warm Setup minus Testing time).

11.5.3 Test Results

The results of the measurements are presented on Figure 56.

The chart shows dependencies of different times on partition size, i.e. on the number of Run Controllers and PMG Agents. The maximum number of agents used in the configuration was 392 (indicated on the chart by the dash vertical line), while the partition size was growing up to 600 Controllers, i.e. the number of infrastructure applications to start was increasing.

The cold Setup time (blue curve) is a combination of Testing, Agents Boot and Infrastructure Boot times. The time to start agents (brown curve) and to start applications (dark magenta curve) scale quite good, while the Testing time shows non-linear behaviour, contributing to the same behaviour of the Cold and Warm Setup times. Non-scalable behaviour of testing of PMG agents was also seen very clearly on bigger partitions (500-700 nodes).

The shutdown time (orange curve) is almost constant and shows good scalability.

1. Note that this "Initial" state is NOT the run control INITIAL state which occurs after the control tree has been booted by the DSA.

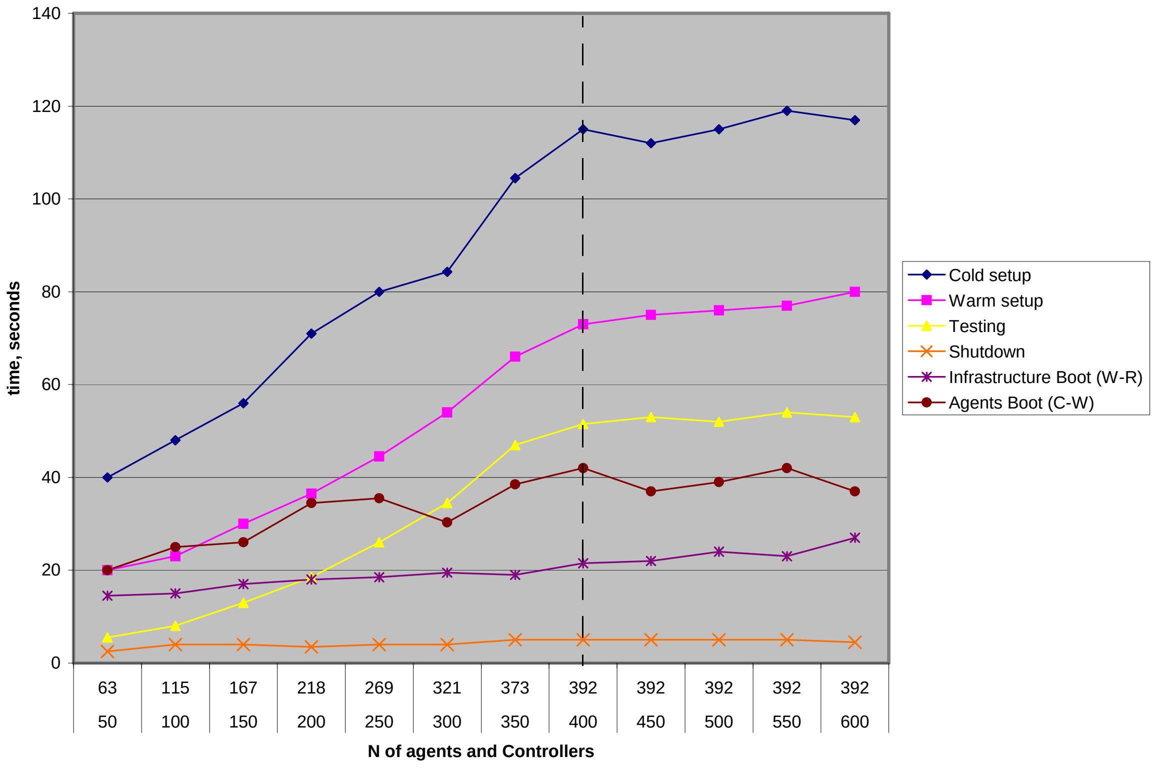


Figure 56 Setup timing versus number of PMG agents and Run Controllers.

11.5.4 Test Summary and Conclusions

The non-scalable behaviour of the Testing time, contributing to the general Boot time was understood and explained by the way how tests queuing was organized. As most of the tests are executed locally, in order not to overload the system a queuing of tests was implemented in the DVS knowledge base (CLIPS engine) by introducing some rules. It appeared that as number of objects in the partition is growing (the total number of objects loaded in CLIPS memory is about 2 times number of Agents plus 2 times number of Controllers in a partition), these rules were fired extensively and slowed down performance of the engine. The proposed solutions should visibly improve the performance. What is to be implemented in the coming releases:

- move tests queuing from DVS rules to the TM code
- load only necessary objects to setup memory
- implement test on PMG Agents as library call (instead of using TM/PMG calls)

In addition to the measurements explained above, partitions with of 300-600 Controllers were run with different settings of DVS, allowing it to launch 10, 15, 20 and 30 tests simultaneously (and to put other tests in a queue). The result were compared and an optimal number of simultaneous tests was determined as 15. This setting is controlled via TDAQ_TMGR_MAX_TESTS environment variable.

11.6 Run Control Scalability Tests

The Controller has recently been re-implemented. Three software components (DSA, rc, LocalController) have been substituted by a single component, the Controller, encompassing the functionalities of run and process control. While most of the integrated tests described within this document have been carried out with the old version of the run control and supervision, this section focuses on measurements performed on the new implementation.

11.6.1 Aims

The aim of these tests is to prove that the new implementation of the Controller shows a satisfactory performance and is suited to control large systems.

11.6.2 Test description and Configurations

These tests have been carried out when the cluster had a size of about 500 PCs. Partitions have been build with the automatic database generation tools, making sure that each Controller had about the same number of children to launch and command.The Control tree had always three levels. A partition labeled “512 apps“ thus means the following configuration and Control tree:

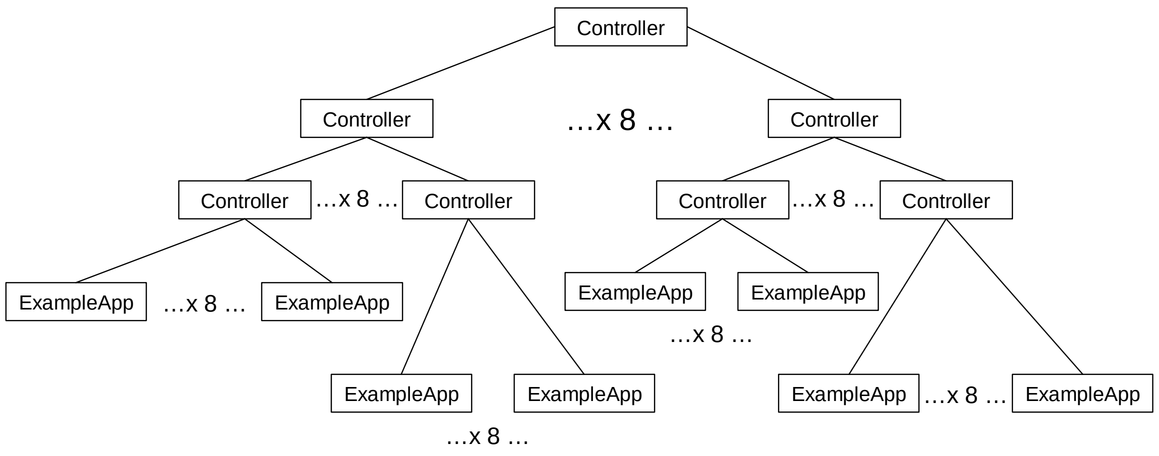


Figure 57 The controlers and applications tree for the partition labeled 512_apps.

11.6.3 Test Approach and Executionof the Tests

Tests were executed sequentially via a script. The usual tools were used for automated timing tests. Each configuration was measured 5 times.

11.6.4 Results

The results are shown in Figure 58. The longest transition is the Boot (contained in the cold start) in which the children (other Controllers and/or leaf applications) of each Controller are launched via the Process Manager. The command distribution over IPC and the treatment of the IS callbacks notifying the result of the transition are very fast: 0.2 - 0.3 seconds/transition and only slightly varying with the size of the system.

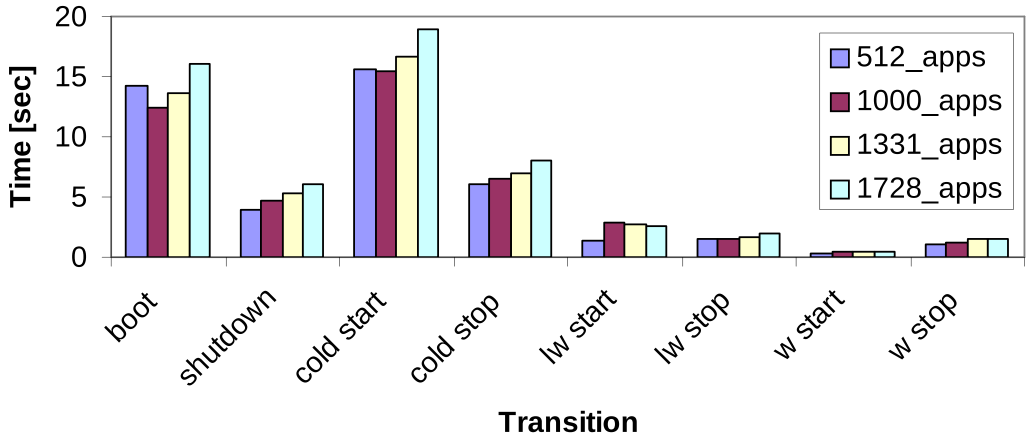


Figure 58 Timing results for the different state transitions for varying size partitions.

11.6.5 Summary and Conclusions

The new implementation of the Controller shows a satisfactory performance and works stably. The fault tolerance features included up to now are working correctly, but they are fairly elementary yet.

11.6.6 Future Steps

The Controller needs now to evolve in the area of fault tolerance and recoveries. Also, its capabilities shall now be compared with the official requirements document, to ensure that every requirement has been satisfied.

11.7 Controls WG Summary and Conclusions

A detailed set of measurements has been carried out to study the behaviour of individual components developed within the Controls WG. A number of limitations and short comings have been revealed, which will serve as a basis for future developents. Also, new components such as the Access Manager and the Log Service have been tested for the first time in a large scale environment and have shown encouraging results, though of course there is room for improvement in all areas.

12 Conclusions: Summary, Experience and Future Large Scale Tests

The large CERN LXBATCH facility provided the opportunity to run in July 2005 online functionality tests over a period of 6 weeks on a stepwise increasing farm size from 100 up to 700 pc dual nodes. The interplay between the control and monitoring software with the event readout, event building and the trigger software has been exercised the first time as an integrated system on this large scale. New was also to run algorithms in the online environment for the trigger selection in Level2 and in the event filter processing tasks on a larger scale. A mechanism has been developed to package the offline software together with the DAQ/HLT software and to distribute it via peer-to-peer software efficiently to this large pc cluster. The findings obtained during the tests lead to many immediate improvements in the software. Trend analysis allowed identifying critical areas. Most of the planned tests could be run, in some areas we went further than originally hoped. Running an online system on a cluster of 700 nodes successfully was found to be especially sensitive to the reliability of the farm as well as the DAQ/HLT system itself and the future development will concentrate on fault tolerance and stability.

12.1 Summary Results

The main results of the integrated tests, the sub-system tests and the component group tests are presented in the following.

Tests were run with the **integrated DAQ/HLT-I software**: the online infrastructure, the readout, the event building, the Level2 trigger without algorithms and the Event filter without algorithm. The integrated partitions were operated and tested up to about 30% of the full ATLAS system size. There were increasing difficulties to operate larger and larger systems, due to problems showing up clearly at large scales only. Actions had been taken to fix them and the tests continued throughout the testing period. Problems with the setup procedure are hoped to be taken up and improvements are expected, while the state transition times are quite acceptable.

Online infrastructure tests were carried out to verify the functionality and to determine the basic behaviour of the DAQ/HLT system. The controllers were controlling a dummy example application which takes part in the control flow but does not perform real other tasks.The concept of distributed RDB servers was introduced and used the first time for these tests. It was very successful allowing the use of the RDB on a large scale, a concept needed in preparation for the future implementation of the configuration database access to ORACLE. Several partition sets with increasing number of controllers controlling dummy applications were run. The partition sets were build following different control structures. The results show that the state transition curve for an optimal balanced controller tree is basically flat with 0.7 s per state transition. Similar tests were performed and these results confirmed with the new implementation of the controller.

The **event building system** was tested up to the full ATLAS scale. The recorded transition times were in an acceptable range and it is concluded that the principle control aspects of the event building system are verified up to the scale necessary for the full Atlas size. For the performance measurement, it can be confirmed that the real tests will have to be performed in an environment with the realistic ALTAS network and realistic PC configurations at dedicated testbeds and the pre-series.

The **Level 2 tests without algorithms** concluded that for smaller systems, the timing results for cycling through the run control states, are satisfactory. For larger partitions a high number of failures were encountered which were mainly due to problems with the infrastructure software.

Event Filter data flow tests with a dummy Steering Controller (PTdummy) were performed with varying sizes of event filter sub-farms and up to a total of 1200 processes.The behaviour of the Event Filter without Athena algorithms show good performance with a scaling behaviour in the state transitions which depends on the farm size.

The main goal of testing **Level2 with Offline selection algorithms** in the Large Scale Test cluster was to verify that no fundamental show stopper had appeared compared to the small scale tests that were done before. In particular, the tests meant to investigate the access sharing of L2PUs of the Conditions DataBase when running muFast, in particular at configuration transition. This Large Scale Test was the very first time that many L2PUs were run simultaneously in this mode. Although timing graphs are presented in this report, it was explicitly not the goal to obtain detailed performance measurements. This will need to wait for other Large Scale Tests with better control over the network topology and network traffic. The test results show that several of the quantities measured for the conditions database access increase slowly with the number of L2PUs in the partition. The Level 2 tests with the muFast algorithm have clearly shown that with this one algorithm and the partition sizes explored so far, there are no worrisome problems apparent.The number of tests which could be performed is however still quite small.

The **Event Filter tests with Offline event selection algorithms** were run the first time on a large scale. Besides the general studies of the scaling behaviour with increasing size of EF partitions, special attention was given to the simultaneous access of a high number of PT applications to the configuration database and to the conditions database. The efficient access to the configuration database was successfully established during the cause of these tests. The simultaneously access and reading of the Conditions Database was also probed and problems were identified. Due to the limited duration of the Tests, there was not enough time to solve these problems and thus, these measurements are subject for future Large Scale Tests.

With the aim of using a coherent set of software releases of the DAQ/HLT-I, HLT and offline software throughout the testing period thus allowing for reasonably easy updates on the test cluster, it was decided to install all the software needed for the HLT algorithm tests on a local disk on each of the cluster machines separately in one large container file. This **HLT image concept** was developed as part of the large scale tests and has proven very useful. As **software distribution method** the peer to peer mechanism software BitTorrent was used successfully and the presented performance values demonstrate very good values. It was able to download a 2 GB file to 600 hosts in less than 45 minutes, which is almost a factor of four faster than what would result from a sequential copy operation.

Three **Monitoring Services**, namely Event Monitoring, Information Service and Online Histogramming, have been tested. These services form the basis of the ATLAS online monitoring system and the results of these tests can be used to estimate the overall monitoring system scalability and performance. The results for the IS service are very similar to the ones which have been obtained in March 2004 LST with similar hardware configuration. This shows that the overhead of additional functionality was met by improved efficiency. The aim of the Event Monitoring scalability tests was to prove performance and scalability after a re-implementation since summer 2004. There is a limitation on the monitoring task side, regarding how many pushEvent calls it can handle per time interval. This certainly is an important outcome of the Large Scale Tests and deserves some further investigation.The tests with different sampling criteria showed a bottleneck for a higher number of monitoring tasks. It must be investigated among the users how many event channels per sampling application are needed in order to obtain a clearer view on where to set priorities for future steps. The Online Histogramming service is implemented on top of the Information Service. It includes the handling of very large information objects (e.g. histograms), an aspect which was not covered by the IS tests. The tests show that the OH service works very reliably. For large histograms the time of network transfer becomes significantly and unnecessary network traffic should be avoided by making use of a special type of subscriptions which is provided by the OH.

Tests have been performed for the **Configuration Databases** using **OKS with relational backend** as an evaluation for its future use in the final system. It was expected, that the configuration databases require an order of 10... 20 thousands of database objects to describe the final ATLAS system. With the current approach the use of a single Oracle server via intermediate rdb servers is under consideration. The total number of required rdb servers varies from 30 to 100 (i.e. about total number of racks). In this case the database can be read by all rdb servers in a time interval from 15 seconds (minimal size and number of rdb servers) to 1.5 minutes (maximum size and number of rdb servers). Some improvements are foreseen to be implemented on the OKS relational back-end side. Significant improvements are however expected from the use of RAC[1] Oracle servers. Once they will be available, corresponding tests should be performed.

The **Conditions Database Interface** using the **new COOL Oracle based conditions database** has been tested and compared to test with the existing Lisbon conditions database interface based on MySQL. The tests showed that there is still some improvement needed in the COOL based CDI, as there are some bottlenecks that may impose restrictions at a higher rate of object publishing, in particular concerning improvements of the performance of object storage by using object bulk operations.

Tests in the area of Control were performed on the new implementation of the controller, the DVS, the IGUI and the Setup component and also for the first time the new components Access Manager and Log Service. A detailed set of measurements has been carried out to study the behaviour of these individual components. A number of limitations and shortcomings have been revealed, which will serve as a basis for future developments. The new components have shown encouraging results. Specific tests on the setup component allowed to identify the problems which had lead to the rather non scalable behaviour observed throughout the large scale tests and several steps for improvements are planned. The new implementation of the Controller shows a satisfactory performance and works stably. It will now evolve in the area of fault tolerance and recoveries.

12.1.1 Experience

Running the DAQ/HLT system on a large scale during a period of six weeks increased significantly the experience in the understanding of the system and in the tuning of system parameters like for example time out values, database access, the optimization of configuration trees.

The tests were run the first time on such a large scope and on such a large scale. A work plan had been established prior to the tests and discussed in DAQ/HLT. Scheduled yet flexible testing periods had been defined and the team members had defined responsibilities. The tests had been prepared before the large scale tests had started in many areas and this turned out to be very useful. Many useful helper tools were also developed during the testing period driven by urged need and they are now taken up to

1. RAC is a scalable farm composed of several Oracle servers, which shares common database. It should be linearly scalable for read-only requests (two nodes are in two times more fast than one, 4 nodes in 4 times than 1, etc.)

be centralised and integrated into the DAQ/HLT-I software. The communication tools, namely elog, email, the web site, and Twiki pages were heavily used and regular meetings were held three times a week, discussing the last tests, priorities, and making plans and a schedule for the next tests to be done. The team was very flexible to accommodate unforeseen situations, cooperative to promptly share information and to provide help. This very good team spirit is an aspect which is important when running tests around the clock for weeks with the aim of making good use of the available testing time. The choice of a stepwise increase of the farm size was a very good one and gave us a longer period of testing and thus allowed to identify and correct upshowing problems continuously throughout the tests. It would be better though next time to have less steps and increase the step size so to reduce the overall disturbance when changing from one step to the next.

In addition, the work on LST05 provided a valuable test-bed for gaining practical experience for the pre-series and for enhancing and developing tools to build and operate large partitions.

During the tests a very high number of failures were experienced when exercising the DAQ/HLT system. Those were due to the instability of the DAQ/HLT system and the vulnerability of a large computing farm.

Fault Tolerance

As a consequence, the fault tolerance behaviour of the DAQ/HLT system was implicitly exercised on numerous occasions. At the current level of implementation, fault tolerance actions are implemented in some areas only, in particular the actions related to controls. This applies to runs started manually via the IGUI. However if run in automatic mode, most actions cannot be recovered on failure by setup or control. Failures in the communication between application processes lead to hang-ups. When working on a large scale, frequent failures must be expected to occur even when the system is mature. Small changes in the software, hardware failures and aging can provoke unexpected problems. A large number of possible failures can be anticipated and the corresponding actions implemented. Unexpected failures due to hardware problems or intermediate bugs in the software will also occur and higher level ‘global’ actions should be part of the DAQ/HLT system to either lead the system into a state from which it can recover or to stop the system gracefully.

For a high multiplicity of processes crashes must be excepted.
For example, one of the problems experienced during the test was the following: A partitions is started on 700 nodes which contains 2000 controllers and 2000 example applications. The boot process does not terminate correctly. One out of the 2000 identical example applications crashed. There was no recovery available. Then cleanup procedure and subsequent restart has cost ~ 1 hour. Running the identical partition again, no failure occurred.

Fault tolerance needs to be seriously taken up in all areas in the HLT/DAQ system.There are also areas in the fault tolerance behaviour of the DAQ/HLT system on which choices have to made and a strategy has to be identified.

Stability of the DAQ/HLT system

While setup, shutdown and state transitions have been exercised numerous times during the tests, no major long runs were performed when staying in the running state. During the tests on the transitions many failures were observed, as described in detail in the individual sub-system chapters.

A system is stable when a disturbance does not produce too disrupting an effect on that system. For the DAQ/HLT system this stability is required over a longer period of time, the data taking period. On one hand, the correct functioning of the software components is a pre-requisite. This implies prior review and testing of the code. Secondly, the definition of the interfaces between software components and the strategy for the communication between processes also in case of errors need to be ***clear.*** The overall stability of a system is largely related to the level of available fault tolerance, as described in the paragraph above.

Farm system monitoring, stability and repair; Diagnostic tools

An overall experience from running tests on a farm with up to 700 nodes is the impression that the complexity of the system and the potential for problems grows like ‘exponentially’ with scale

- Problems were repetitively encountered with disc space and access.
- There were no or only preliminary tools for farm, disc, and processes monitoring.
- Cleanup and repair procedures and tools were missing.
- Another class of problems was related with the ssh authentication on the LXBATCH cluster. At certain times the hostkeys for a fraction of the cluster became apparently invalid and caused failures that were at first very hard to diagnose.
- A difficult aspect of the testing situation was that there was no coherent way to diagnose a problem situation.

A global picture of the state of the farm system should be easily available. Diagnosing a problem should start with the verification of the farm system before DAQ/HLT system specific diagnose tools can be deployed. A way of integrating those two tools sets could be envisaged. It is suggested to explore the pre-series for development and testing in this area.

Database Generation tool

The configuration database contains the description of the hardware and software objects, parameters and their parameterization for relation of the TDAQ partition. A partition to be used on a large scale contains thousands of elements. A tool to generate such partitions is essential. As most of the knowledge of the TDAQ system lies in the database, this is a rather complex tool. The database generation tool used during the large scale tests had been used successfully before the tests for other specific sub-system database generation tasks. The demands to build the other parts of the TDAQ system and also the integrated system lead to numerous changes throughout the testing period. Testing the tool before generating a configuration had to become a necessary step.The homogeneity of the available command line user interface suffered from ad-hoc changes. The tool did not support configurations with algorithms and so the partitions had to be modified by hand to accommodate for this. Flexibility of a future database generation tool is one of the aspects which is important because frequent software change will occur.

The automated database generation is vital for the tests, a re-visit of the generation tool is strongly recommended.

Examples of other problems

Note that infrastructure components setup, controller, configuration database handling and information exchange are particularly eminent when running (in general and) at large scale because they are vital for the starting and controlling the run and needed before any other component can be tested. Problems, if they exist, include nearly all tests. Therefore special emphasis is devoted to those components here.

Setup component

- The Setup component was used the first time seriously. It supports the newly introduced concept of multiple RDBs servers, which was vital for the tests, and therefore play_daq was not a fall back solution.
- Being a new component, a number of problems were found in the first phases and 4 times patches were provided.
- The test pmg_agent failed frequently, automatic retry included.
- Even after the last patch, serious scalability problem (not solved during the LST): 16 minutes to start a large partitions on 700 nodes.
- large partitions were very difficult to start, as the setup component exhibited a non-linear increase in the time it took to start and test pmg_agents. Sometimes we just could not get a partition started, even with drastically increased timeout values, or had to resort to killing all processes belonging to the partition. E.g. the 16x32 L2PU partition worked fine the first time we ran it, but failed to start an hour later when no change was made anywhere.
- play_daq could not be used as a back-up solution because the concept of multiple RDB servers had only been implemented in Setup.
- On the example of the Setup component it can be seen that new components need more thorough testing at existing test beds.

Local controller problem

- The local controller crashed about every 1/10000 operations. This problem could not be solved but a work around was found which enabled the continuation of the tests. The new version of the controller is generic and does not include the local controllers and therefore not this problem.
- This is an excellent example for the usefulness of the tests on a large scale. The problem also occurs on a very small scale but extremely rarely and it is difficult to capture it in order to debug it, not being reproducible. On the large scale involving a very high number of transitions, it occurs at nearly each run at one of the first transitions.

Operations critical to scale

Testing the DAQ/HLT system on a large scale implies in particular the investigation of the operations which are critical to scale. It is recommended to review and test the performance systematically for those parts of the code for which the execution frequency increases with scale. Operations critical to scale include the state transitions. The controller is sensitive to the control structure, the setup function sensitive to the number of hosts which it has to serve, the boot transition to the number of processes (in tdaq-01-02-00), the stop transition is determined by the time to empty the event buffer for EF, and the configure transition to the access to the configurations and conditions database.Thrend analysis at least on a smaller scale should become part of the regular component testing.

Example: Efficiency of code accessing configuration databases

- Database access is sensitive to scale and is done by many processes concurrently. The use of an efficient access method is important to the system.
- The graphs in figure Figure 59 show three cases of inefficient database access and the improvement obtained by applying the correct measure. The measures have been applied during the tests in sequence.
    1. The use of distributed remote database servers (RDBs) is necessary for a system which involves more than 50 nodes.
    2. Database access is high for infrastructure processes at boot transition. For optimizing the access mode one needs to define the OKS DAL's algorithm option DAL_USE_PATH_QUERY to replace a full scan of the database on the client side by the more efficient query executed by the database server. (This setting was used for debug purposes during the tests and it is default now in the subsequent releases)
    3. Inefficient database access method leads to scalability problem. Once replaced by a query, the improved configuration time for 400 EF PTs is reduced from one minute to a few seconds.

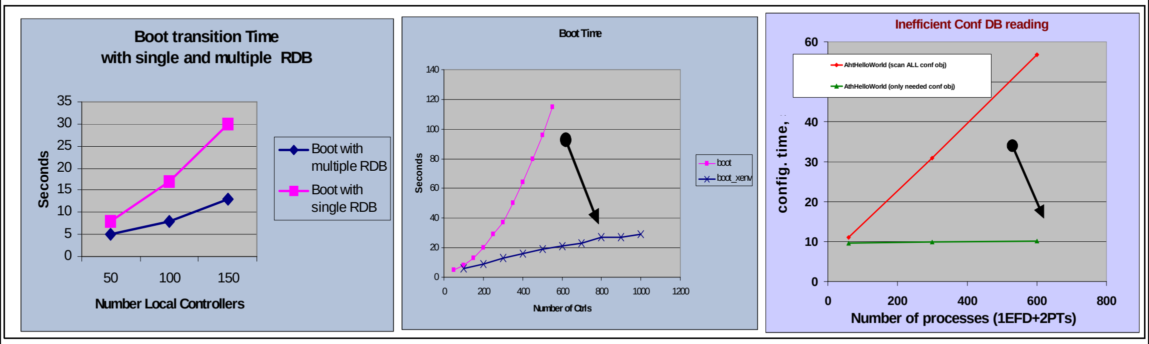


Figure 59 example for the optimization of the configuration database access

Follow-up

The results and the experience of these large scale tests have been presented to the TDAQ community and many of the problems have been or are being solved by the time of finalizing this report. Suggestions are being taken up by component developers, in the context of the working groups and in specialized activities.

12.2 Future Tests

A number of areas have been identified which need further testing and verification on a medium or large scale before the final Atlas run. The following list is meant to give an idea of the most prominent type of tests which still need to be done. Testing on small and medium scale test beds as well as on the pre-series installation will have to precede a future test on a very large scale with ~1000 nodes.

The main areas which need further testing on a large scale are:

- For all areas in DAQ/HLT, the fault tolerance behaviour will have to be tested. Triggered by the results of these tests, improvements in the area of fault tolerance are already on-going in particular in the controls working group and will help for a more satisfactory system behaviour on a large scope.
- Stability tests will need to be emphasized in all areas of DAQ/HLT. This concerns the failure free repeatability of control transitions including all controlled applications but also the operations during the running state and the communication between processes taking part in the control of the dataflow and the dataflow itself. Many of the tests on stability can be prepared and tested already in the pre-series and on small test beds. More attention should be given to problem situations even when occurring rarely on a smaller test bed and they should be analysed correctly, keeping the large scale in mind.
- In the infrastructure domain new implementations need to be verified on a large scale. This concerns in particular modifications in the operations of the boot/shutdown and the stop procedure, the revision of the configuration DB schema, the usage of the new error reporting system and the resource manager, and the new implementation of the process manager and the corrections for the Setup. Sufficient time must be available for re-stabilizing the software and deploying it on a smaller scale to ensure its fitness for testing and use in large scale.
- Alternative Level2 farm structures than studied in this years tests need to be investigated on a medium and large scale.
- For the Event Filter, the following studies need to be performed:
    - Study the subfarm behaviour as a function of the number of EFD/PTs, with real algorithms and their memory performance.
    - Investigate the optimal number of PTs (2,4,6) per EFD with real algorithms.
    - Transition times should be studied as a function of the subfarm dimension, both without and with algorithms.
    - The throughput, oscillations and back pressure mechanisms should be studied with and without algorithms and as a function of the number of PTs per EFD. This needs to be done on a testbed which allows to separate control and dataflow.
    - Stability studies on a small and medium size testbed should study the long term stability, test PT and EFD recovery procedures, and study throughput behaviour (throughput degradation).
- For both LVL2 and EF tests including algorithms, the following studies need to be performed:
    - Conditions database access: Understand timing, server access bottlenecks, the optimization of Oracle and MySQL server configurations and the possibility for server replication for read and write access to
        - geometry database
        - calibration and alignment database
        - algorithm initialization database

- Include as many algorithms as possible. For this years test there was only one algorithm available for Level2 (muFast), and one for the Eventfilter tests (TrigMoore). It is expected that from the next offline release onwards fife or six algorithms can be used for Level 2. Different algorithms need also to read different data from the above listed database information in the conditions database which increases the load.
- Run tests combining LVL2 and with EF algorithms together

Testing on existing small to medium scale test-beds and at the pre-series will be continued. As much preparation work as possible should be done there throughout this and throughout the coming year 2006.

Medium scale tests, in particular for the tests including algorithms, are envisaged on remote clusters. We hope for the possibility for sub-system tests on Westgrid in Canada or the Liverpool cluster in 2nd quarter 2006. Operating system incomparability in particular in view of the athena software, availability of the test clusters and remote farm management are uncertainties.

A larger farm which is permanently available for testing at a medium scale (several hundreds of nodes) would be **highly desirable**. Therefore it is suggested to buy a large number of pc's for the experiment earlier as foreseen (Mid 2006) and use them for the various testing purposes.

Finally it is proposed to run **one more Large Scale Integration Test** at LXBATCH at Cern in **Autumn 2006, on ~1000 nodes** with the aim of verifying the fitness of the DAQ/HLT software on a large scale just in time before final, to have a chance to remove remaining bottlenecks, limits and other scalability problems.

12.3 Acknowledgement

We thank the Cern IT division for providing the access to the CERN IT LXBATCH tests bed, Bernd Panzer-Steindel for his organisational support.We thank in particular out IT colleagues Veronique Lefebure, Thorsten Kleinwort, and German Melio for setting up and managing the nodes, installing our release software and for being kindly available throughout the testing period for special support operations.

.