ATLAS

ATLAS HLT/DAQ Software

Plans and Actions for Future Large Scale Tests

Document Version:	0.7
Document Date:	30 October 2004
Document Status:	Draft
Document ID:	ATL-D-TN-0002

Abstract

This document presents the plans and actions required to prepare for the next HLT/DAQ large scale tests in 2005.

Institutes and Authors:

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

University of Bern: H.P. Beck

CERN: A. Bogaerts, D. Burckhart-Chromek, P. Werner

Table 1 Document Change Record

Title:	ATLAS HLT/DAQ Software Plans and Actions for Future Large Scale Tests		
ID:	ATL-XXX-XXX		
Version	**Issue**	**Date**	**Comment**
0	1	31.8.2004	Creation and outline
0	2	4.10.2004	Additional text by Sarah and inclusion of Andre's text on system and farm management
0	3	10.10.2004	Additional text by Doris on chapter 3s, 5 documentation and 6 schedule and planning
0	4	25.10.2004	Additional text by Per in chapter 2, 3 and 4. Added chapter 8.
0	5	27.10.2004	Additional text in chapter 6 by Doris, Sarah small edits throughout
0	6	29.10.2004	Move chapter 6 to become chapter 2, mods by Doris
0	7	30.10.2004	more corrections by Doris

1 Introduction

In March 2004 the latest in a series of large scale HLT/DAQ software tests took place. Sub-systems involved were the Online Software, the LVL2 ROI collection software, the Event Filter dataflow software and the associated High Level Trigger (HLT) supervision software. In addition to the sub-system tests the scope was extended for the first time to include an integrated HLT/DAQ large scale system test. A detailed report of the March 2004 tests may be found in [1]. To summarise, the Online Software sub-system tests were generally a success, benefiting from experience gained during earlier tests in the series. The standalone HLT tests confirmed that small configurations worked, establishing that the same implementation of the software could and has been used successfully at the 2004 combined testbeam [2]. However, due to the instabilities observed in the standalone HLT supervision tests and large scale problems with the Event Builder software, no attempt was made to run a fully integrated system. Although this final goal was not achieved, useful experience was gained during the preparation of the integration test and it also demonstrated the importance of scheduling a similar large scale test in the future.

This document presents the plans and actions required to prepare for such a large scale test. In particular, the lessons learnt from the instabilities observed when running the HLT but also from the success of the Online Software sub-system standalone tests are taken into account.This document aims to expand chapter 7 Future Steps of the test report [1].

1.1 Aims and Scope

This document summarises the feedback and discussions which have taken place since the unsuccessful attempt to run a large scale HLT/DAQ integration during the March 2004 Atlas HLT/DAQ Software

Scalability tests. The document presents a list of hardware and software requirements which must be fulfilled before a new large scale test should be attempted. Secondly a detailed planning schedule is presented to indicate how preparation should proceed in relation to the currently planned software release dates and assuming the next large scale tests will take place some time in the first half of 2005. Finally, an outline of the actual content of the planned large scale tests is given. The requirements and planning presented here will be used as input for decisions to be made on the future development of the HLT/DAQ, on timing and changes to software release dates and on aims and layout of future tests. Large-scale testing should be considered an iterative process which continues during the period of rapid software development over the next year or so, in order to identify scaling problems in and/or verify the scalability of the components of each major software release. It is hoped that the majority of effort involved will be in setting up the framework for the first of these tests see section 2.2 . Once this framework is in place, the effort required for each subsequent iteration should be less.

1.2 Glossary, acronyms and abbreviations

The terms and abbreviations used in this document are the ones used by TDAQ and readers should refer to the ATLAS TDAQ glossary [3].

1.3 References

1 H.P. Beck et al., ATLAS HLT/DAQ Software Large Scale Tests March 2004, ATL-D-TR-0001 (2004), EDMS 499884, http://edms.cern.ch/document/499884

2 Testbeam Paper presented by Marc Dobson at CHEP 2004

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

4 Western Canada Research Grid, http://www.westgrid.ca/

5 M.Zurek, ATLAS TDAQ Testbed Management, January 2004, ATL-DQ-ON-0003, EDMS 433779, V1.0, http://edms.cern.document/433779/1.00

6 http://nagios.org

7 D. Kouzis-Loukas, Information Service Data Logger (2003) http://dkouzisl.home.cern.ch/dkouzisl/islogger/documentation.pdf

8 ftp://linux-rep.fnal.gov/pub/rgang

9 A. Kazarov et al., ATLAS TDAQ Controller Requirements, ATL-DQ-ES-0054

10 Diagnostics and Verification Framework, http://atddoc.cern.ch/Atlas/DaqSoft/components/diagnostics/Welcome.html

11

2 Schedule and Planning

2.1 Schedule

The schedule for the next large scale HLT/DAQ integration tests and the releases should be well synchronized. Ideally we would like to schedule a large scale test to verify each major release (upgrade) of the software and in doing so maximize input to the next release. In reality the timing of the tests has to be a compromise because we do not have complete flexibility over the availability of large-scale testbeds. These have to be reserved well in advance and the timing of the allocation cannot be changed to accommodate delays in the software release schedule. A realistic time scale has to be developed which does not put undue pressure on software developers to be ready for large-scale testing. By arranging a series of iterative tests it may be possible to alleviate the problem (manpower permitting). If a particular component is not ready for one large-scale test the system is tested with emphasis on other components and the delayed component is tested the next time around. Testing time should also be reserved for prototype testing of crucial components (like the controller) where needed. As much use as possible of our own testbeds (where we have much more flexibility) should be made in the meantime. Note that when algorithms and the offline software are included in the testing this will introduce further scheduling problems over which we have even less control.

It is foreseen that the first integrated large scale tests should be performed in the **second quarter of 2005**. This will allow sufficient time for feedback to and correction of the major release of the HLT/DAQ software which is planned for summer 2005 and which is planned to be used for commissioning. Well tested and stable releases must be available then. This test would preferably be done on the LXSHARE testbed with about 1000 nodes. If important components are not mature to be included in the large scale tests then at least one other large scale antedated test should be foreseen somewhere towards the end of 2005 to verity the scalability of the new implementations.

In parallel it is planned to run partial tests (EF stand alone) on another large testbed (i.e. about 500 nodes, WestGrid) **in the 1st quarter of 2005.** Again this is foreseen as an iterative activity, manpower permitting.

To prepare for the large scale tests, binary and integrated tests using the latest available versions of the software should be run on **medium scale testbeds** we have at our disposal (building 32 and building 513, about 100 machines in total) **on a continuous basis**. All tools, scripts etc. that are intended to be used for the large-scale tests should be tested (see following section).

It is necessary to **fix the dates for the releases of the HLT/DAQ software** up to summer 2005

2.2 Steps for preparation

The following steps are not listed in order of priority. Most of them will have to start now and will be on-going in parallel at least up to the large scale tests.

1. **Farm management:** TDAQ needs to agree on and provide a common tool and its maintenance for general farm management, cluster monitoring tasks. These are general system management tasks and should best be maintained by a software support team.
    1. Action: HLT/DAQ to determine one or more person(s) (system manager) to investigate, install and maintain cluster management tools; examples are nagios for cluster monitoring and rgang for file distribution
2. **Cluster monitoring specific to the TDAQ system:** There are aspects of task management and monitoring which are specific to the TDAQ system. Integration or specific interaction between selected TDAQ system components and those tools needs to be done. This could concerns the process management and its monitoring in the graphical user interface.
    1. Action: Testers to exploit existing options and present a list of required features to be presented to concerned developers
    2. Action: once agreed upon, developers to implement it
3. **Evolution of existing TDAQ components:** Two aspects should be considered there, **Fault tolerance aspects which are specific to the testing environment** and **HLT/DAQ system debugging facilities**.
   Generally the system is tested in a state where it is not yet functioning correctly, and testing means also to take it to its limits in areas where it is important in view of the final usage. When executing these tests, a number of faults may occur interleaved. The system containing thousands of processes distributed over hundreds of nodes may get into a state which is very complex to analyse and understand. Good facilities to help here are necessary. Features should be added to some of the existing component to help here. Examples are the cleaning up of processes by the pmg_agent in conjunction with ipc when the TDAQ system is in an unrecoverable error state. Use cases should be provided to illustrate those situations. Another example is the organisation and viewing of log files, and the content and understandability of error messages. A number of these problems have also been seen meanwhile in the test beam and its feedback should be equally considered for the steps to be taken.
    1. Action: Testers to provide a list of use cases/features for specific fault tolerance aspects
    2. Action: Testers to provide a list of use cases/features for debugging aspects
    3. Action: once agreed upon, developers to implement it
4. **Database generation scripts:** database generation scripts for large configurations to be used for sub-system tests and for the integrated tests need to be updated as part of each release.
    1. Action: Testers to review the common list of required features
    2. Action: A responsible to implement updates and to support as part of each release
5. **Writing and/or updating and testing of the execution and analysis tools**: scripts exist for sub-system usage. They need to be extended, some of the underlying software has changed and the scripts need to be adapted accordingly. In other areas, they need to be newly developed or adapted to the new tests.
    1. Action: Testers to review necessary features for automatic execution and analysis scripts
    2. Action: Testers to write and/or update automatic execution and analysis scripts

6. **Documentation and problem knowledge base**: Under complex system test conditions, there is the regular need to get quick help on the usage of components and on error codes is lacking. A 'quick reference' to all the components in the system like an "online -h", 'HLT-h', 'DF -h' etc. Problems are encountered which cannot always be fixed very quickly but intermediate solutions or work around may have been found. A simple knowledge base should be established where these situations are kept and can be searched for in an easy way.
    1. Action: Sub-systems to establish a 'quick reference' for all the components of the system as part of each release
    2. Action: establish a light knowledge base for the description of unsolved problems with their work around and fill it when cases come up
7. **Pre-testing on a medium scale:** integration testing of the sub-systems with emphasis on the features which are relevant for the large scale tests on available medium scale test beds; this should be performed on a continuous basis for each stable and well tested release. The developed testware and tools like the database generation, automatic testing and analysis scripts should be used on a regular basis to ensure their correct functionality.
    1. Action: HLT/DAQ to define all release dates up to summer 2005
    2. Action: HLT/DAQ to ensure for standard release testing
8. **Getting ready with specific HLT/DAQ components**: verifying the test specific fault tolerance in Local Controller, verify/investigate the possibility of using the ROS (not the ROS emulator as last time) and the HLT Gatherer
    1. Action: Testers to test on medium scale testbed
9. **Include algorithms:** This requires reliable a distribution kit for "online version" of offline software. Experience over the summer shows that the current Pacman distribution kit was not satisfactory and will be replaced by a new one.
    1. Action: HLT to make the necessary modifications to enable running on remote testbeds on a large scale
10. **Identification of a test bed:** A test bed which is available during the requested testing time (which is to be defined) and which matches the necessary technical needs should be found (i.e. 4 weeks in May 2004).
    1. Action: HLT/DAQ to investigate the possibility for the use of the LXSHARE cluster at Cern, the Liverpool farm, the Westgrid facility in Canada (and others if there are any).

2.3 Manpower needs

Manpower is required in setting up the framework, testing tools and testware on medium-scale TDAQ testbeds at CERN. Once this testing framework is in place and a coherent set of tools is available then subsequent large scale testing may become straight forward. We may be able to benefit from experience gained in remote farm project if/when facilities outside of CERN are used for testing. We should foresee automated integrated tests running continuously on building 32 and 513 clusters. The figures given below assume that standard release testing will have been done by the respective sub-systems prior to the medium scale tests.

DAQ/HLT-I:

1. Preparatory work as explained above for and on medium size cluster, including initial testing: 50% of 3 people (distributed over 3-6 people) from December 2004 to March 2005.
2. Testing maintenance: testing for new release on medium scale; For each new release 50% of 3 people (distributed over 3-6 people) for 2 weeks
3. For each large scale test: 2 weeks before and 3 weeks after testing time: 50% of 3 people (distributed over 3-6 people); during the tests (4 weeks) 80% of 4 people (distributed over n people)

in addition for HLT-P:

For the HLT software, it is assumed that the preparation, testing and distribution of releases as well as setting up and maintaining configuration data bases will have to be done irrespective of the Large Scale Tests. Additional manpower is then needed for testing, installing and ensuring access to databases specifically for the Large Scale Tests. It is difficult to assess at this moment because streamlined procedures are not in place yet. A rough initial estimation is 50% of 2 people for 4 weeks.

3 System and Farm Management Requirements

3.1 Testbed

3.1.1 Size

A cluster of at least 500 to 1000 nodes will be required to approach the size of the final ATLAS system which will have thousands of nodes. However, where LST04 tests were unsuccessful clusters of 300 nodes would already be useful, in the first instance to re-run the standalone HLT tests. [1]

3.1.2 Operating System version

The kernel must be compatible with the software releases.There should be provisions to customise the OS regarding drivers, libraries, compilers, databases and custom packages. Presumably root privileges will be required for some operations. Large scale tests may be difficult if all nodes are not identically configured. Note 1: The remote farm project may be able to provide help for running on operating

1. The linux facility at WestGrid [4] has already been identified as a possible candidate to run these intermediate tests perhaps as early as January 2005. Feasibility studies are currently under way for standalone Event Filter tests and a request for time on this facility is being prepared. The WestGrid linux facility is a 500 node, 1000 cpu cluster located at the University of British Columbia in Canada.

systems other than RH7.3. Note 2: Once algorithms are to be included in the large-scale tests the offline software may impose strict constraints on the operating systems which may be used.

3.1.3 Availability

It must be possible to reserve the cluster well in advance for the period of time identified by ATLAS HLT/DAQ as the most suitable for the tests. ATLAS HLT/DAQ must have exclusive use during this period. The cluster provider must not cancel or change the reservation at short notice. Equally, we will not be able to change the testing period and will have to be ready in time.

3.1.4 CERN-like environment

Ideally we would like to run on the LXSHARE cluster at CERN if the above conditions can be met. If not it must be possible to configure the cluster to be compatible to the LXSHARE cluster. A related requirement being that the effort involved in achieving this should not be disproportionate to work required for the actual tests themselves. For clusters external to CERN it must be possible to gain access to a few nodes of the cluster as early as possible to run tests with example partitions to determine feasibility before a formal request for time is made.

3.2 Generic tools to manage a large cluster

The tools for management of large clusters should be agreed and maintained TDAQ wide. We should therefore involve the TDAQ community in the preparation of the Large Scale Tests. We should minimise the work invested in interim solutions and re-use existing solutions as far as possible. Use should be made of experience already gathered during testbed administration and troubleshooting in TDAQ [5].

3.2.1 Booting

The local boot procedures for the cluster should be used. It should be possible to reboot individual machines at any time.

3.2.2 Faulty Networking Equipment

As for booting, we expect to be able to follow the standard local procedures.

3.2.3 Faulty Nodes

As in the previous tests, we need a "machine" list with procedures to add new nodes and eliminate faulty ones. Hopefully it can be generate automatically, for example with nagios. We also need to be able to quickly regenerate the OKS database ("generation scripts") to reflect such changes, see 4.1 .

3.2.4 System Monitoring

A "Nagios like" [6] tool, customised for TDAQ, to monitor, maintain and repair the cluster (nodes, network, applications) is essential.

3.2.5 Utilities to deal with local files

The "rgang" [8] is a tool developed, maintained and distributed by Fermilab. It provides *parallel remote execution of commands* via rsh or ssh on a number of nodes. Initial tests of this tool have been made by the authors. Its possible usage should be discussed within TDAQ as it is fast by virtue of its parallelism.

3.2.6 Privileged access

Users should be able to execute certain privileged commands, e.g., reboot, system statistics. This could be done with a tool like *sudo*.

3.3 TDAQ specific tools

The following is a list of tools required to run a large-scale TDAQ configuration. This category of tools relies on TDAQ specific information, e.g., the configuration database. In general the tools have already been developed in TDAQ but should be extended/adapted in order to improve performance at large scale.

It would be an advantage if application names are forced to be unique in the configuration database.

3.3.1 TDAQ Monitoring Tools

The tools for monitoring both at system level (ps -wxyz ...) and application level (using the display tools in the GUI) were insufficient and much too primitive, e.g., the current view of pmg_agents is often not useful for large systems. An optional view by segment hierarchies would be more useful. A merge of the RunControl panel and the pmg panel would be useful, e.g., *view the hierachey of segments - applications* - and optionally on which host they run.

We should have a means to visually compare the configuration defined in the database with the actual configuration running in such a way that it is easy to spot duplicate and/or missing applications.

For operational monitoring, the islogger [7] has proved to be a useful tool at the testbeam for visualizing monitoring information. Its use should be investigated further.

Further development of sub-system specific IGUI panels is required. Information on work done already e.g. on the EF panel should be gathered.

3.3.2 Application Management

Often there were multiple copies of the pmg agent on the same machine; aborting pmg agents did not kill the applications it managed; restarting and checking pmg agents on a cluster is a time consuming procedure..

The pmg_kill_agent utility is very useful but relies on the ipc being in a working state. Some improvements could be

1. the system kill command should do the same as pmg_kill_agent (OK already ?)
2. optionally the pmg_agent should kill all applications and exit if it has been disconnected from the IPC server for more than a configurable time.
3. pmg_rm <reg-exp> where application names, as in confDB, which matches <reg-exp> will be removed (killed)

3.3.3 IPC Server

The ipc server may need to have replicas. (not clear: The ipc server sometimes needed to be aborted and restarted, for unknown reasons and causing long delays.)

3.3.4 Process Cleanup

There should be a tool which rapidly cleans up the system in order to return to a well-defined state. It must kill all sub-system applications plus all the partition-dependent and partition-independent Online Software infrastructure. One possibility would be that the tool checks the configuration database to see which applications might be running and then kills these applications (and duplicates of these applications) when it finds them. Another approach would have the cleanup facility killing all TDAQ applications it finds on a list of machines regardless of the partitions to which they belong. This would require that applications must be recognisable as TDAQ applications without referring to the configuration databases. The second approach would provide the more robust cleanup procedure.

3.3.5 Tools to (un)install application software

Tools will be required to install/maintain all software releases (Online, Dataflow, HLT, Offline) and to create user accounts. The use of shared accounts should be avoided as it is both a security issue and a source of potential operational problems.

4 Requirements on Tools

4.1 Database generation

During the March 2004 tests scripts in the DBGeneration package were used to generate the OKS configuration databases. These scripts are necessary for the production of a suite of large databases which may vary in terms of size or number of sub-farms or which use different system parameters for the tests. These scripts are maintained as part of the DataFlow release. A continued effort must be made to keep the database generation scripts up-to-date with each software release in order that valid configurations can be generated for that release. [1]

4.2 Automatic test execution tools

4.2.1 HLT

On top of the DBGeneration package mentioned above, LVL2 used two levels of scripts:

1. script A which given a list of available machines (?) generates one specific LVL2 farm.
2. script B which runs a set of LVL2 farm configurations, increasing the number of farms for each configuration until resources are exhausted.

These scripts were adapted to EF farms. Script A was used. Script B was not used due to controller related scaling problems in EF running configurations generated using Script A.

Note: These scripts were a solution of necessity and we think that common tools for Online, DF and HLT should be agreed upon in the near future.

4.2.2 Online

The online tests made use of the already existing “time” option in play_daq which automatically records the timing of the individual state transitions. Scripts to automate the execution of long series of such tests with sets of databases were re-used from previous tests. As play_daq is being **replaced by Setup**, an equivalent option should exist there which allows the automatic execution of long test series and consequently **new scripts will have to be written**.

1. A discussion has taken place together with the current script developers and responsibles from the configuration group in order to clarify if the database generation tools could become the responsibility of the configuration group. It was decided that for the time-scale of the next large scale tests, the scripts would be continued to be updated and used. In the longer term, a more general tool aiming for its use at commissioning and final Atlas which would be more closely coupled to other configuration components could be envisaged.

For the component tests, custom scripts had been written and used by the component testers. Re-use of those scripts is assumed in future tests as far as the demand allows it.

4.3 Analysis tools

4.3.1 HLT

Some scripts were written to analyse the results from the LVL2 tests and produce necessary plots. As for the test execution scripts, common TDAQ tools are needed. The Online tools described below could be a good starting point.

4.3.2 Online

The online system re-used a set of scripts running under linux. From the output produced by play-daq and the corresponding test execution scripts, a large set of histograms was produced on the mean value of runs and on comparing those with varying parameters. The output of these scripts was also conveniently used as input to the production of more specific excel graphs. **The analysis scripts will have to be updated according to the output provided by Setup.**

For the component tests, custom scripts had been written and used by the component testers. Re-use of those scripts is assumed in future tests as far as the type of tests allow for it.

5 Diagnostic, Debugging and Fault Tolerance Requirements

5.1 Debugging and Diagnostic tools

Should this refer to tools to gather/display log files? Clearly gathering, collation, archiving and display of the log files written on the local disk of each machine in a large cluster is a major issue. DVS [10] currently provides some tool for browsing logs. This facility should be investigated.

It is hoped that existing tools can be used here for example, DVS [10]. Tests on medium-scale testbeds should be made to understand the use of this component.

5.2 Fault Tolerance in the Testing Environment

Improved fault tolerance is required to enable the system to start/run/terminate correctly despite some applications which fail either while performing a state transition or during a steady run control state.

Applications may fail for various reasons, including bugs in the application software, network problems, failure of the hardware components on which they are running etc. Following the problems observed during the March 2004 large scale tests some rudimentary fault tolerance was added to the LocalController for the combined testbeam; an automatic restart of failed applications is attempted if specified in the database; if the failed application is defined as a Resource the LocalController will ignore it by removing it from its list of controlled applications (after having tried to restart it - if specified). The new fault tolerance behaviour has yet to be tested at large scale. Note that the fault tolerance in the new run control component [9] will be fully customizable by each sub-system.There will also be means to manually recover from faulty conditions.

Fault tolerance also required in starting the Online Software infrastructure. For instance during the March 2004 tests if one pmg agent failed to start correctly on a cluster of 300+ nodes the whole procedure aborted. It must be possible to continue to run even if some pmg agents fail to start on some machines. There must be some defined “quality of service” parameter for the minimum number of running nodes with successfully started pmg agents below which the start up would fail. Note that there must be more flexibility in the configuration; the applications which were assigned to nodes on which pmg agents have failed to start must be reassigned dynamically to other nodes.If the failed node is in a trigger farm it could be ignored (provided a given minimum are up), if the node was to have run (e.g.) a DFM then it must be reassigned.

6 Documentation, flow of Information

Under complex system test conditions, there is the ongoing need for a quick reference option on the usage of components and error codes. Unexpected problems come up and new ways of running the software are being tried out.

A ‘quick reference’ to all the components in the system like an “online -h”, ‘HLT-h’, ‘DF -h’etc. which lists and provides a brief description of all the sub-system Software utilities online or/and in form of Web pages. It should be part of the release to encourage it being updated.

Problems are encountered which cannot always be fixed very quickly but intermediate solutions or work around may have been found. A simple knowledge base should be established where these situations are kept and can be searched for in an easy way. The electronic log book is a first step in this direction.

Effort must be made to keep all documentation, both paper and online, updated.

7 What to test

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

In preparation for the integrated test, it is planned that each sub-system would verify its functionality on a large scale with a series of their own stand alone large scale tests directly prior to the complete integration.

For the online system, the confirmation of the large scale functionality of the system and of selected individual components like the communication software, the run control software, setup and the remote database server will be of importance.

The High Level Trigger tests are split into separate LVL2 and EF stand alone tests. The aim for the LVL2 sub-system is to investigate a system of a size as close as possible to final ATLAS. The EF tests are intended to investigate state transition times as a function of sub-farm size, number of nodes per sub-farm and number of Event Filter applications.

It is aimed to include the ROS this time into the system as opposed to the ROS Emulator which had to be used during the last tests. The Gatherer should be included into the tests and be used on a large scale It also is envisaged to include algorithms if possible. New implementations of existing components as for example the controller, the pmg, or possibly a relational database to replace OKS, and others should be included into the system to be tested if they are mature. Otherwise dedicated tests should allow to test prototype versions. New components like the access manager could be included as well.

7.1 Databases

With the inclusion of HLT algorithms many more databases will be needed there:

- OKS (populated by scripts that generate the required information for many different configurations)
- Initialisation of algorithms (‘cabling’ and the like)
- Conditions Database

This poses clearly scalability questions that we should address in the tests.

7.1.1 Local file servers

The current HLT model is to use subfarms (physically corresponding to a ‘rack’) with a file server and access to databases. Unresolved issues are the need for local replicas of databases, caches, proxies, synchronisation & coherency, subfarm wide shared file system or diskless operation. There are clearly scalability questions to be addressed.

Since it is unlikely that all these issues will be resolved by the time of the test, we should assume the availability of a cluster wide shared file system (afs) in addition to a a (node) local file systems. The latter implies a the need for a cluster wide tool to distribute/collect/delete/create/copy/search these local files.

Keep in mind that HLT relies heavily on dynamically loadable libraries, job option files, ancillary data files and data to be pre-loaded into ROSs and LVL2 Supervisors.

8 Appendix

8.1 Use Cases for pmg_agent

1. The ipc_server dies.
2. The IGUI dies or is killed.
3. The pmg_agent loses the connection to the ipc_server.

8.2 Use Cases for Fault Tolerance

4. An algorithm crashes an L2PU (or a PT).
5. A ROS fails due to node hardware problems.
6. A LocalController dies.