ATL-DAQ-CONF-2007-002
11 January 2007

ATLAS HIGH LEVEL TRIGGER INFRSTRUCTURE, ROI COLLECTION AND EVENT BUILDING

K. Kordas, INFN Frascati, Italy
J.Schlereth, Argonne National Laboratory, IL, USA
J. Masik, ASCR Prague, Czech Republic
H.P. Beck, S. Gadomski, C. Haeberli, Bern University, Switzerland
S. Armstrong, Brookhaven National Laboratory, NY, USA
A. Bogaerts, N. Ellis, S. Gameiro, B. Gorini, M. Joos, J. Haller, T. Maeno, C. Padilla, T. Pauly, J. Petersen, M. Portes de Albuquerque, R. Spiwoks, L. Tremblet, G. Unel, P. Werner, CERN, Switzerland
E. Ertorer, CERN, Switzerland & University of Ankara, Turkey
C. Bee, C. Meessen, F. Touchard, CPPM Marseille, France
Y. Nagasaka, Hiroshima Institute of Technology, Japan
M.L. Ferrer, W. Liu, INFN Frascati, Italy
M. Bosman, E. Garitaonandia, H. Segura, S. Sushkov, IFAE Barcelona, Spain
Y. Yasu, KEK, Japan
G. Comune, R. Hauser, Michigan State University, MI, USA
G. Kieft, S. Klous, J. Vermeulen, NIKHEF Amsterdam, The Netherlands
J. Baines, V. J. O Perera, F. Wickens, Rutherford Appleton Laboratory, U.K
S. George, B. Green, A. Misiejuk, J. Strong, P. Teixeira-Dias, Royal Holloway, University of London, U.K.
A. Gesualdi Mello, M. Seixas, R. Torres, UFRJ - Rio de Janeiro, Brasil
T. Bold, A. Lankford, A. Negri, S. Wheeler, University of California Irvine, CA, USA
R. Cranfield, G. Crone, University College London, U.K.
X. Wu, Université de Genève, Switzerland
P. Morettini, C. Schiavi, University of Genova & INFN, Italy
R. Stamen, S. Tapprogge, J. Van Wasen, Universität Mainz, Germany
A. Kugel, M. Mueller, M. Yu, Universität Mannheim, Germany
R.Ferrari, W. Vandelli, Università di Pavia & INFN, Italy
A. Di Mattia, S. Falciano, E. Pasqualucci, Università di Roma I "La Sapienza" & INFN, Italy
A. Dos Anjos, H. Zobernig, W. Wiedenmann, University of Wisconsin, Madison, WI, USA

(Presented by Kostas Kordas on behalf of the Data Collection, HLT-Integration & HLT Core s/w and Steering Groups of ATLAS TDAQ)

Abstract

We describe the base-line design and implementation of the Data Flow and High Level Trigger (HLT) part of the ATLAS Trigger and Data Acquisition (TDAQ) system. We then discuss improvements and generalization of the system design to allow the handling of events in parallel data streams and we present the possibility for event duplication, partial Event Building and data stripping. We then present tests on the deployment and integration of the TDAQ infrastructure and algorithms at the TDAQ "pre-series" cluster (~10% of full ATLAS TDAQ). Finally, we tackle two HLT performance issues.

INTRODUCTION

The ATLAS experiment is designed to observe the outcome of collisions between protons of 7 TeV (i.e., at the highest energy to-date), provided by the Large Hadron Collider (LHC), which will start operating in 2007 at CERN. Bunches of 10^{11} protons will cross each other at 40 MHz, providing about 25 proton-proton interactions per bunch crossing at the centre of ATLAS. Nevertheless, only a small fraction of this ~1 GHz event rate results in interesting physics processes. The TDAQ system of ATLAS has to select a manageable rate of such events for permanent storage and further analysis. ATLAS records about 1.5 MB of information for each event and we aim in keeping about *O*(200) events per second.

We use a three-level trigger system to achieve this goal. An overview of the system is seen in Fig. 1. The first level trigger (LVL1) uses coarse calorimeter and muon detector information to reach a decision within 2.5 μs. The accept event rate is 75 kHz (upgradeable to 100 kHz). Upon an event accept from LVL1, the front-end electronics of the various sub-detectors pass the corresponding data to the Readout Buffers (ROBs), hosted in the Readout Subsystem (ROS) PCs. Event data remain there and are pulled by the second level trigger (LVL2) and then by the event builder nodes on demand.

LVL2 receives a list of $\eta-\phi$ Regions of Interest (RoIs) identified by LVL1, from the Region of Interest Builder (RoIB). It then accesses the appropriate ROSs to pull and analyse data from the ROBs corresponding to the Regions of Interest; thus, the LVL2 uses only ~2% of the full

event information to take a decision. At this level the event rate is reduced to ~3 kHz with an average latency of *O*(10) ms. Subsequently, the Event Builder (EB) nodes, also known as the "sub-farm input" (SFI) nodes, collect all data from the ROSs and, upon request, provide fully assembled events to the Event Filter (EF) farm, which is the third level trigger. The EF analyzes the entirety of each event data to achieve a further rate reduction to ~200Hz, with a latency of *O*(1) second per event. Finally, accepted events are sent for local TDAQ storage to the "sub-farm output" (SFO) nodes.

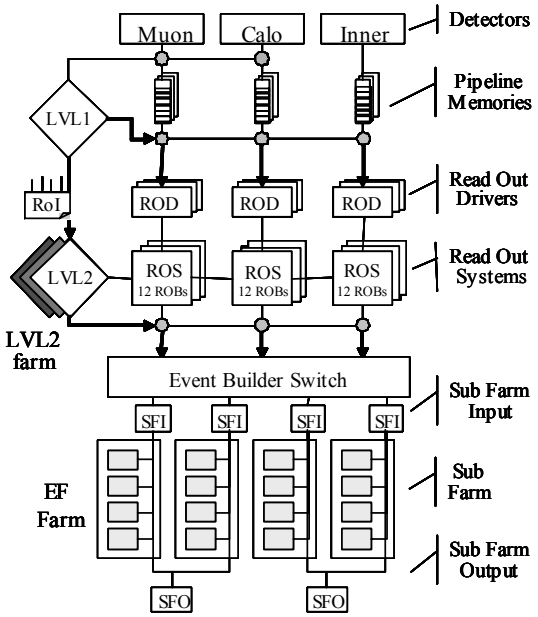

Figure 1. Basic components of the ATLAS TDAQ system. The data flow from the pipeline memories on the detector's front-end electronics (top), towards local storage in the TDAQ system (SFOs, at the bottom), where they wait to be pulled to permanent mass storage.

While LVL1 is based on custom hardware, the LVL2, Event Builder and EF are based on computer farms of *O*(3000) PCs which are interconnected via Gigabit Ethernet and run Linux.

The LVL2 and EF are collectively called High Level Trigger (HLT). We call "Data Flow" the part of the TDAQ system which is responsible for buffering the event data at the ROSs, feeding the HLT with the data it needs in order to reach a decision, and, eventually store the accepted events at the SFOs.

Detailed description of the system can be found in [1], while extensive tests are described in [2] and [3]. This paper discusses functionality improvements on the Data Flow system and results obtained by running algorithms at LVL2 and the EF. We also discuss two tests which demonstrate the proper scaling and the matching of the performance requirements for the LVL2 system.

DATAFLOW

The dataflow components and their interaction with the HLT components which run the algorithms are shown in Fig. 2. The only custom-built hardware in this figure are the LVL1 trigger, the RoI Builder and the custom-built cards inside the ROS PCs [1]; all other components run on standard PCs as C++ applications developed in a common software framework which imposes a suite of common functionalities, e.g., application control, message passing, etc. The dataflow applications are multi-threaded, since they must react to asynchronous 'events', such as run control commands, request for monitoring data and interaction with other components.

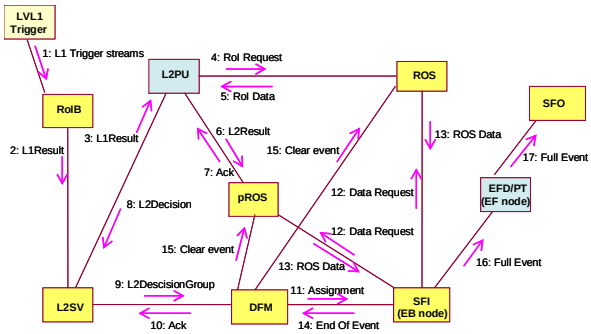

Figure 2: Flow of data and messages relevant for the HLT. The LVL2 selection software runs in the L2PUs requesting selected data from the ROSs according to the RoI information provided by LVL1. The EF deals with full events, received from the SFIs.

The trigger decisions are taken by algorithms running on the LVL2 Processing Units (L2PUs) and on the Event Filter's Processing Tasks (PTs). The rest of the system deals with the flow of data; it serves them to the HLT, and eventually to storage, and deals with the event according to the HLT decision (e.g., it releases the data buffers upon a rejection by the HLT).

The flow of data complies to a pull scenario: data are requested according to needs from subsequent clients. All applications involved in the LVL2 and Event Building communicate via message passing; data is one of the message types flowing from the requested ROSs to the appropriate client. Each EF node (EFD/PT in Fig.2) gets an already assembled event from an SFI and, if it accepts the event, it sends it to an SFO for local storage.

Data Flow for LVL2

After a LVL1 accept, and while the data are buffered into the ROSs, the RoIB collects RoI information from the LVL1 calorimeter and muon triggers and from the LVL1 central trigger processor. This information is put in the "L1Result" and is forwarded to one of the LVL2 Supervisors (L2SVs) in a round-robin fashion. Each L2SV serves a sub-farm of L2PUs and assigns one of them to process the event.

The L2PU figures out the ROBs corresponding to the given RoI and requests data only from the involved

ROSs. It also produces an accept/reject decision which is passed back to the L2SV which in turn forwards it to the Data Flow Manager (DFM). For accepted events, the L2PU puts the decision details (the "L2Result") in the pseudo-ROS (pROS).

If the decision is to reject the event, the DFM sends clear messages to the ROSs to free the involved buffer space. If the event is to be kept, the DFM assigns an SFI to assemble the event by requesting data from all participating ROSs and the pROS. Events are buffered in the SFI and made available to the EF upon request.

Data Flow for the Event Filter

The ATLAS EF system is organized as a set of independent processor farms (sub-farms), each connected to Sub-Farm Input (SFI) and Sub-Farm Output (SFO) elements. Unlike the LVL2 system which involves many components to deal with the dataflow and the trigger aspects of the work, each EF node hosts both functionalities. Dataflow functionalities are provided by the Event Filter Dataflow process (EFD), while the processing tasks (PTs) are in charge of data processing and event selection.

The EFD manages the communication with the SFI and SFO elements and makes the events available to the PTs via a memory mapped file, called the SharedHeap, which is used for local event storage and provides a safe event recovery mechanism in case of EFD crash. The PT cannot corrupt the event because it access the SharedHeap in read-only mode. PT problems are handled by the EFD which can identify PT crashes and dead locks. In both cases, the EFD, which owns the event, can assign it to another PT or send it directly to the SFO.

Inside the PT, the event selection algorithms produce a filtering decision and a selection object (used to classify the event and guide the off-line analysis steps) which are communicated back to the EFD. The filtering decision steers the EFD dataflow and thus decides the event fate.

Partial Event Building and Data Streams

A full event, assembled by an SFI which has gathered all the fragments from the ROSs, is about 1.5 MB in ATLAS. There are cases though, where we may only need "partial" events which contain information from some of the ATLAS sub-detectors only.

The TDAQ system resources can already be partitioned in mutually exclusive configurations. Thus, we can have parallel calibration runs, each dealing with part of the ATLAS detector and giving out partial events.

During a physics run, accepted events will be stored in four different data streams at the SFO level: physics, express physics (for fast reconstruction of very interesting events), calibration and debug. Further refinement is envisaged outside the TDAQ system. In order to optimize bandwidth, processing and storage resources in the TDAQ, we are adding partial event building functionality for events which are only good for calibration and/or debugging.

For routing and (partial or full) event building purposes, we define a stream-type message to be used up to the event builder. Since the EF is handed already assembled events, the SFI will incorporate this information in the event. The trigger part of the system (LVL1 and HLT) is responsible for deciding on the stream tag(s) characterizing each event, whereas, as usual, the dataflow components have to act according to these tags.

The LVL1 central trigger processor generates triggers for testing (debugging) and calibration purposes of various sub-detectors, mixed with the physics triggers. An event accepted by LVL1 with a non-physics trigger will not be processed by a L2PU and it will be built partially: the assigned SFI will pull data only from the ROSs corresponding to the sub-detector's needs for calibration and/or testing. Physics-triggered events will be always analyzed by the LVL2. In case they satisfy a physics trigger, they will be always built fully, no matter if they are also good for calibration and/or debugging purposes. On the other hand, an event which is rejected for physics and is only found useful for calibration and/or debugging will be built partially, according to the info passed to the event builder by the stream-type message.

An equivalent strategy is envisaged at the EF level. E.g., an event can be routed by the EFD directly to the SFO, without making it available to a processing task. In case an event is sent to a PT and it is accepted for physics, the full event will be sent to the SFO which will check the event's stream tag(s) and store it locally to the appropriate data stream(s). Note that an event which is tagged as express physics as well, will be written to both the standard physics and the express physics streams. If an event is only good for calibration/debugging though, the EFD will strip the unnecessary information and send the partial event to the SFO for writing to the appropriate stream. All use cases and the details of the implementation are currently under investigation.

Event Filter I/O protocol

The EFD manages the EF host's connection with the data sources (SFIs) and destinations (SFOs) and implements the client part of the communication protocol, EFIO [4]. As part of the configuration information, at start-up each EFD receives the host and port numbers of the SFI(s) and SFO(s) to which it must connect. The EFD then establishes one or more TCP connections to these SFI and SFO and in case of breaking connections, the EFD reestablishes them to ensure the proper flow of data.

It is always the EFD which initiates the transactions: it requests events from the SFIs and after it receives all Ethernet fragments successfully, it acknowledges the event reception and is ready to initiate a new event request. The EFIO was designed with full events (~1.5 MB) in mind. However, this hand-shaking mechanism is inefficient in case of very small event sizes (which could occur in the case of partially built events), because a large fraction of the event transaction time consists of the event request and acknowledgement messages from the EFD towards the SFI. For this reason, the latency overhead is

compensated by having multiple connections between each EFD and the serving SFI(s). Thus, an SFI can wait for an acknowledgment message in one channel, while fully utilizing the offered network bandwidth to serve an event to the same EFD via a parallel channel.

The connection towards an SFO works essentially the same way, but the EFD initiates the event transaction by sending a space request to the SFO when it has an event to send. If space is available, the SFO requests an event from the EFD and, after receiving the event, it acknowledges its' receipt. Here too, the latency overhead is compensated by having multiple connections between each EFD and SFO(s).

HIGH LEVEL TRIGGER

Both LVL2 and EF use online software for the control and data collection aspects, but the event processing and trigger algorithms are developed and tested in the ATLAS offline software environment (Athena). A common approach of the LVL2 Processing Unit and the EF Processing Task for the steering, seeding and sequential processing of the selection algorithms has been developed [5]. Through a sequence of "feature extraction" and "hypothesis testing" algorithms, the HLT tries to reject the event as soon as possible in order to minimise required CPU resources. By dealing only with RoI data at LVL2, we also minimise the required network resources.

A thin layer of software, common to LVL2 and the EF, acts as an interface between the dataflow and selection software. This maps the online state machine used by the run control on the event loop manager of Athena; it provides access to the data in the format the algorithms need it (e.g., RoI data for LVL2); and, it maps online services on equivalent Athena services (e.g., initialization, data access, error reporting). The idea is to be able to "plug" easily the event selection algorithms developed offline into the online HLT framework.

For the EF this task is easier, because the less stringent latency requirements (~1s, compared to ~10ms per event at LVL2), allow the straight-forward reuse of offline algorithms as configurable plug-ins. For LVL2 algorithms, the relatively long idle time between requests and arrival of RoI data (~10 to 20% of total, see next section) allow resource sharing in multiple-CPU nodes. An L2PU could then deal with multiple "worker threads", each handling one event. Thus, algorithms which run at LVL2 should be designed to be thread-safe and efficient. For most algorithms this is not yet the case. The alternative (and our baseline solution) is to run multiple applications on each node at the cost of using more resources.

Because online and offline software releases are independently managed using different tools and conventions, we install all the software needed by the HLT, together with setup scripts and data files, into an 'image' for deployment on multi-node test beds [6].

Tests with algorithms at LVL2

Recently we have tested the 'e/γ' and 'μ selection slices' on the "pre-series" cluster, which is about 10% of the full TDAQ system installed at the ATLAS site [2,7]. A 'selection slice' involves running a series of feature extraction and hypothesis algorithms on RoIs identified by LVL1 in the muon spectrometer or the calorimeter. The algorithm sequence proceeds "outside-in": E.g., if the RoI seeds in the muon spectrometer contain a good muon candidate, algorithms requesting a matching track in the inner tracking detectors (Si pixel and strip detectors) are executed next.

We preloaded eight ROSs with three different data files containing simulated muon, dijet and single electron data; the corresponding LVL1 RoI information was preloaded to one L2SV which triggered the system. The algorithms ran on a single L2PU which put the L2Result for selected events into a pROS. We note that the dijet sample is the most representative of LHC conditions, because in p-p collisions jets occur much more frequently than electroweak events. Each dijet event had, on average, 23 minimum bias events superimposed.

The dataflow application running on the L2PU was instrumented to provide timing and data throughput information. Though neither the trigger menu nor the event mix were representative of the final ATLAS (such tests are planned for later in 2006), the runs give us already useful estimates of the required CPU power and bandwidth. Most importantly, valuable lessons were learned about the complex software integration procedure.

The results of the measurements are summarised in Table 1, where we see the average values (per event) for the LVL2 decision latency, the processing time (i.e., the actual execution time for the chain of algorithms), the RoI data collection time, the size of RoI data requested, and the number of requests for RoI data.

Table 1: Results of algorithm testing at LVL2 with simulated events preloaded on ROSs.

Data File	LVL2 Latency	Process Time	RoI Coll Time	RoI Coll Size	#Reqs /Event
	(ms)	(ms)	(ms)	(bytes)	
μ	3.4	2.8	0.6	287	1.3
jet	3.6	3.3	0.3	2785	1.2
e	17.2	15.5	1.7	15820	7.4

The electron sample used was pre-selected and the complete chain of algorithms had to be ran, so results from this sample are not representative. From the muon and dijet samples though, we observe that i) 80-90% of the LVL2 latency is due to the algorithm processing, with the time spent to collect the RoI data from the ROSs being a small fraction of the total, and ii) the size of RoI data requested per event is indeed very small (recall that the full ATLAS event size is about 1.5 MB). By plotting the fraction of events processed as a function of the LVL2

latency, we also observe that the majority of events is processed early (e.g., within the first 2ms more than 80% of the dijet events in this sample are rejected).

Tests with algorithms at EF

A similar test was performed for the EF at the pre-series machines. Simulated muon events were preloaded in an emulated SFI node which served a number of EF nodes, each running two Processing Tasks. Fig. 3 shows the data throughput achieved as a function of the number of EF processing nodes, for two different algorithms.

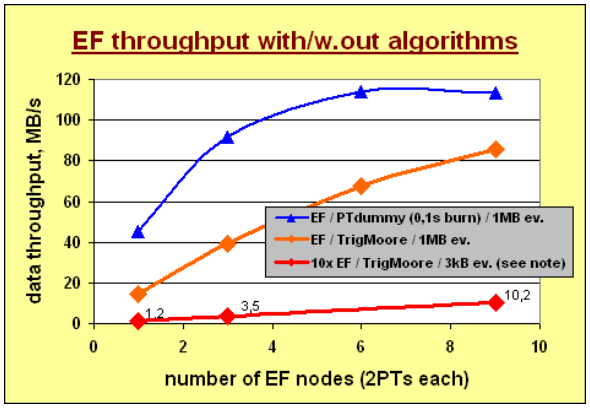

Figure 3: EF data throughput as function EF farm size in EF-only configuration with dummy (top) and realistic algorithms. The top two curves are obtained with an event size of 1 MB. The bottom curve was obtained with a 3 kB event size and shows ten times the measured throughput.

The upper curve corresponds to the case of a "dummy" algorithm which randomly accepts or rejects events of 1MB in size with a fixed latency. The performance scales linearly with the increasing number of EF nodes, till the limit of the Gbit connection between the SFI and the EF sub-farm is reached. The middle curve is obtained under the same conditions, but the muon selection algorithm ("TrigMoore") was running in the PTs. The average processing time is found to be 150 ms per event and the system performance scales again linearly with the EF farm size, but 9 EF nodes were not enough to saturate the Gbit link from the SFI.

The bottom curve is obtained with the same realistic muon selection algorithm, but for events with 3 kB in size and it actually shows ten times the measured throughput. The EFIO protocol used in these tests allowed only one TCP connection between the SFI and each EF node, which clearly limited the data rate for such a small event size. This problem could occur in real ATLAS running for partially built events from the SFI to the EFDs, and for stripped events from the EFDs to the SFOs. Allowing multiple EFD-SFI connections, as discussed in the relevant section above, cures this limitation.

Apart from physics selection algorithms, we have also run monitoring algorithms on the PTs. Actually, for more universality and flexibility, the PT I/O interface is generalized in such a way, that one can easily switch between the various event sources at configuration level: the standard TDAQ dataflow (EFD), the online event monitoring service or a data file on disk. Using the EF for online monitoring is going on as part of the activities of the Atlas TDAQ Monitoring Working Group [8].

Scaling tests at LVL2

The capability of the RoI Builder and LVL2 Supervisors to cope with the maximum LVL1 output rate is of fundamental importance to the TDAQ. The scaling of the LVL2 system has been verified by preloading RoI information into the RoIB, which triggers the system and serves one or two L2SVs. Each L2SV receives the L1Result from the RoIB and distributes it to a sub-farm of one to 8 L2PUs.

In the case of the setup with one L2SV, the sustained rate is ~35 kHz, constant to within 1.5% independent of the number of L2PUs. When two L2SVs are served by the RoIB, the sustained rate is ~70 kHz with an equal sharing of the load between the two L2PU sub-farms, each managed by one L2SV. Since ATLAS will have a handful of L2SVs, we are confident that we can easily manage the max. 100 kHz LVL1 rate.

Performance of processing nodes

In the TDAQ Technical Design Report [9] the computing power for LVL2 was estimated to require the equivalent of ~500 dual processor CPUs at 8 GHz each, resulting in an average latency of 10 ms at the maximum LVL1 design output rate of 100 kHz. This assumption was used to plan the budget but also the infrastructure, such as available space, power and cooling.

The CPU clock speed is unlikely to exceed 4 GHz. However, multi-core CPUs would fulfil the computing power requirement of 16 GHz per box (e.g., 2 CPUs with 2 cores each at 4 GHz in 2007), provided that the processing power of such machines shows scaling with the number of cores they contain. A preliminary measurement shows that this is indeed the case, as illustrated in Fig. 4.

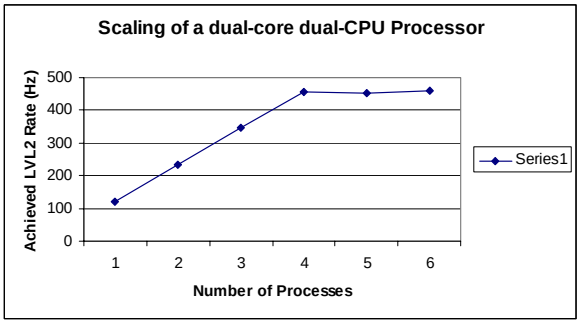

Figure 4: Rate of processed events by LVL2 as a function of the number of identical L2PU applications running on a dual-core dual-CPU LVL2 processing node.

As a LVL2 processor node we used a rack-mounted processor from Super Micro containing two dual core AMD CPUs running at 1.8 GHz with a total of 4 GB of

memory. A similar configuration to the one described above to test the algorithms at LVL2 was deployed, but we preloaded just four ROSs with simulated muon events. We observe that the rate of processed events scales linearly with the number of identical L2PU applications running on the node, till the resources (four cores in this case) are exhausted; additional L2PU applications do not increase the LVL2 rate any further. Therefore, we believe that the multi-core multi-CPU technology should be able to provide the necessary performance per PC, at the cost of higher memory needs and latency. Such problems could be alleviated if we were running applications in a shared memory model .

CONCLUSIONS AND OUTLOOK

We have presented the base-line design and implementation of the Data Flow and High Level Trigger part of the ATLAS Trigger and Data Acquisition system and discussed improvements of the system design to allow the handling of events in parallel data streams and the possibility for event duplication, partial Event Building and data stripping.

The communication protocol between the Event Filter nodes and its' data sources (event builder nodes) and data destinations (local storage nodes) has been improved to handle efficiently small-size events, like partially built events into the EF and stripped events shipped from the EF to local storage nodes.

We have also exercised the integration of algorithms into the DAQ/HLT infrastructure by running algorithms seeded by the result of the previous trigger level. We observe that the RoI collection at LVL2 behaves as expected and that the data throughput scales linearly with EF farm size.

The LVL2 system is shown to be able to cope with the maximum LVL1 rate possible (100 kHZ) and we have also provided evidence that the multi-core multi-CPU technology should be able to deliver the original landmark performance expected from PCs with "dual CPU, 8 GHz each" for the HLT nodes.

Running algorithms at LVL2 and EF in a single chain is currently in progress at the pre-series cluster on the ATLAS site. We are also getting ready to test the system with realistic trigger menu and simulated events, and also to use the HLT on real muon data from the upcoming cosmic run with part of the muon spectrometer and the calorimeter already installed in their final position. We are thus working on delivering an HLT system which shows the proper scaling and matches the performance and robustness requirements of ATLAS towards the proto-proton data-taking starting in 2007.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the help and support of the ATLAS TDAQ group. K.K would also like to thank the local organising committee for the excellent organisation of the conference.

REFERENCES

[1] B. Gorini et al, "The ATLAS Data Acquisition and High-Level Trigger: concept, design and status", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[2] G. Unel et al, "Studies with the ATLAS Trigger and Data Acquisition pre-series setup", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[3] D. Burchart-Chromek et al, "Testing on a large scale: Running the ATLAS Data Acquisition and High Level Trigger software on 700 PC nodes", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[4] H.P Beck et al, " EFIO: Protocol Specification", ATLAS DAQ note: ATL-DQ-ES-oo40 (v.2.0), 2006.

[5] G. Comune et al, " Steering the ATLAS High-Level Trigger ", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[6] H. Garitaonanda-Elejabarrieta et al, "Worm and Peer To Peer Distribution of ATLAS Trigger & DAQ Software to Computer Clusters", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[7] M. Dobson et al, "The architecture & administration of the ATLAS online computing system", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[8] W. Mandelli et al, " Strategies and Tools for ATLAS Online Monitoring", CHEP06 conference, 13-17 Feb. 2006, Mumbai, India.

[9] The ATLAS TDAQ Collaboration, "ATLAS High-Level Trigger Data Acquisition and Controls Technical Design Report", CERN/LHCC/2003-022 2003.