The ATLAS Event Builder

W. Vandelli, M. Abolins, A. Battaglia, H. P. Beck, R. Blair, A. Bogaerts, M. Bosman, M. Ciobotaru, R. Cranfield, G. Crone, J. Dawson, R. Dobinson†, M. Dobson, A. Dos Anjos, G. Drake, Y. Ermoline, R. Ferrari, M. L. Ferrer, D. Francis, S. Gadomski, S. Gameiro, B. Gorini, B. Green, W. Haberichter, C. Häberli, R. Hauser, C. Hinkelbein, R. Hughes-Jones, M. Joos, G. Kieft, S. Klous, K. Korcyl, K. Kordas, A. Kugel, L. Leahu, G. Lehmann, B. Martin, L. Mapelli, C. Meessen, C. Meirosu, A. Misiejuk, G. Mornacchi, M. Müller, Y. Nagasaka, A. Negri, E. Pasqualucci, T. Pauly, J. Petersen, B. Pope, J. Schlereth, R. Spiwoks, S. Stancu, J. Strong†, S. Sushkov, T. Szymocha, L. Tremblet, G. Unel, J. Vermeulen, P. Werner, S. Wheeler-Ellis, F. Wickens, W. Wiedenmann, M. Yu, Y. Yasu, J. Zhang and H. Zobernig

***Abstract*—Event data from proton-proton collisions at the LHC will be selected by the ATLAS experiment in a three-level trigger system, which, at its first two trigger levels (LVL1+LVL2), reduces the initial bunch crossing rate of 40 MHz to ∼3 kHz. At this rate, the Event Builder collects the data from the readout system PCs (ROSs) and provides fully assembled events to the Event Filter (EF). The EF is the third trigger level and its aim is to achieve a further rate reduction to ∼200 Hz on the permanent storage. The Event Builder is based on a farm of O(100) PCs, interconnected via a Gigabit Ethernet to O(150) ROSs. These PCs run Linux and multi-threaded software applications implemented in C++. All the ROSs, and substantial fractions of the Event Builder and Event Filter PCs have been installed and commissioned. We report on performance tests on this initial system, which is capable of going beyond the required data rates and bandwidths for Event Building for the ATLAS experiment.**

Manuscript received November 4, 2007. This research project has been supported in part by a Marie Curie Early Stage Research Training Fellowship of the European Community's Sixth Framework Programme under contract number (MRTN-CT-2006-035606).

W. Vandelli is corresponding author. Email: Wainer.Vandelli@cern.ch

H.P. Beck, A. Battaglia, C. Häberli and K. Kordas are with the Laboratory of High Energy Physics Department, University of Bern, Switzerland

M. Abolins, Y. Ermoline, R. Hauser and B. Pope are with the Michigan State University, Ann Arbor, MI, USA

R. Blair, J. Dawson, G. Drake, W. Haberichter, J. Schlereth and J. Zhang are with the Argonne National Laboratory, Argonne, IL, USA

A. Bogaerts, R. Dobinson, M. Dobson, D. Francis, S. Gameiro, B. Gorini, M. Joos, G. Lehmann, B. Martin, L. Mapelli, G. Mornacchi, T. Pauly, J. Petersen, R. Spiwoks, L. Tremblet, G. Unel, W. Vandelli and P. Werner are with CERN, Geneva, Switzerland

M. Bosman and S. Sushkov are with the Institut de Fisica de Altas Energias (IFAE), Universidad Autonoma de Barcelona, Spain

M. Ciobotaru, A. Negri, S. Stancu and S. Wheeler-Ellis are with the University of California Irvine, CA, USA

R. Cranfield and G. Crone are with the University College London, UK

A. Dos Anjos, W. Wiedenmann and H. Zoberning are with the University of Wisconsin, Madison, WI, USA

R. Ferrari is with the INFN Sezione di Pavia, Italy

M.L. Ferrer is with INFN Frascati, Italy

S. Gadomski is with Université de Geneve, Geneva, Switzerland

B. Green and A. Misiejuk are with the Physics Department, Royal Holloway College, University of London, UK

C. Hinkelbein, A. Kugel, M. Müller and M. Yu are with the Unversität Mannheim, Germany

R. Hughes-Jones is with Manchester University, UK

G. Kieft, S. Klous and J. Vermeulen are with NIKHEF, Amsterdam, The Netherlands

L. Leahu and C. Meirosu are with the National Institute for Physics and Nuclear Engineering "Horia Hulubei", Bucarest, Romania

C. Messens is with the CPPM Marseille, France

Y. Nagasaka is with the Hiroshima Institute of Technology, Japan

E. Pasqualucci is with the Universita di Roma "La Sapienza" and with the INFN Roma, Rome, Italy

K. Korcyl and T. Szymocha are with the Henryk Niewodniczanski Institute for nuclear Physics, PAS, Cracow, Poland

F. Wickens is with the CCLRC Rutherford Appleton Laboratory, Chilton, Didcot, UK

Y. Yasu is with the High Energy Accelerator Research Organization (KEK), Tsukuba, Japan

†deceased

I. Introduction

ATLAS [1] is one of the four experiments being installed at the Large Hadron Collider (LHC). The experiment includes several challenging detection technologies and is supported by a large, distributed trigger and data acquisition system (TDAQ). LHC will provide collisions at a center-of-mass energy of 14 TeV with a frequency of 40 MHz. The output of the ATLAS first level trigger will be in the order of 75 kHz (up-gradable to 100 kHz). This frequency will be further reduced by the higher trigger levels and finally some hundreds of events will be selected and stored every second. The whole ATLAS detector consists of about 140 million electronic channels and the expected average event size is about 1-2 MB.

The ATLAS TDAQ system is based on in-house designed software mostly written using C++ and Java programming languages and running on Linux. Software releases are produced on a regular basis and deployed through ad-hoc shell script via AFS as well as exploiting RPM packages. The final system will consist of a few thousand processing nodes interconnected by a multi-layer Gigabit Ethernet network.

II. ATLAS Data Acquisition

As shown in Fig. 1, the ATLAS TDAQ [2] system exploits three levels of on-line event selection to reduce the initial bunch crossing frequency of 40 MHz to a rate of stored events of ∼200 Hz. The first-level trigger, implemented in dedicated custom hardware, decides, for each bunch crossing, whether the data from calorimeters and/or muon detectors satisfy the trigger criteria. Upon the reception of the accept signal, the front-end electronics transmit the data to the Read-Out Drivers (RODs). RODs are subdetector-specific custom module that collect data and send them as fast as possible to the Read-Out System (ROS) via optical Read-Out Links (ROLs). The ROS is composed by about 150 PCs equipped with up to four

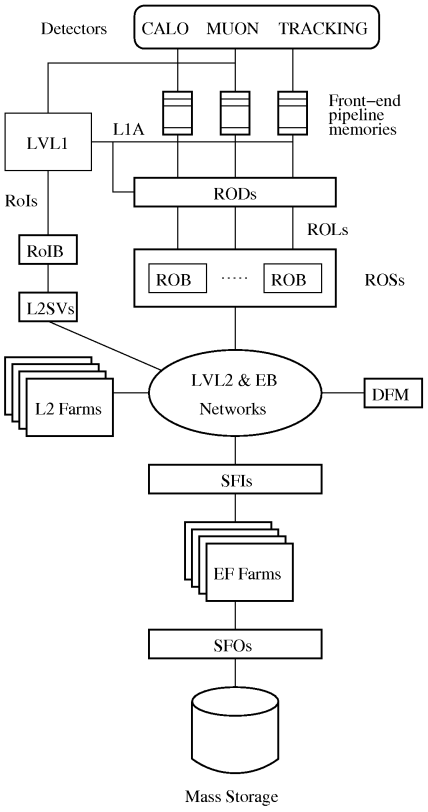

Fig. 1. Layout of the ATLAS TDAQ. The component acronyms are explained in Sec. II.

custom-made PCI cards, called ROBINs, able to handle three ROLs each.

In parallel to the data path, on the trigger path, the Region of Interest Builder (RoIB) assembles the trigger information and transmits them to one supervisor (L2SV) of the second-level trigger (LVL2). The L2SV assigns then the event to a processing application (L2PU) running in the LVL2 farm, which collects partial event data from the ROSes in order to complete its filtering algorithm. After the analysis completion, the L2PU sends the decision message back to its L2SV, which in turn forwards the information to the DataFlow Manager (DFM). The event rate expected after LVL2 selection is in the order of 3 kHz. The DFM is responsible for starting the building process assigning the event to one Event Builder node, referred as SFI, and to send data release messages to the ROSes once the events are built or in case of LVL2 rejection. The SFI requests the data fragments to all the ROSes, building a complete event. The event is then made available to an Event Filter (EF) processor, the second component, together with the LVL2, of the High-Level Trigger (HLT), where the third and final event selection is performed. Events accepted by the EF are sent to one of the Sub-Farm Output (SFO) for the transmission to the mass storage.

III. ATLAS Read-Out

The ROS system is implemented with $\mathcal{O}(150)$ PCs, each of them holding up to four custom PCI modules equipped with three optical links [3]. The ROS CPU, a 3.4 GHz Intel Xeon processor, can access the event data fragments stored in the ROBINs thanks to four independent PCI busses. Indeed, upon the LVL1 decision, the formatted data fragments from the RODs are pushed via ∼1600 links, at an aggregated bandwidth of up to ∼120 GB/s, into the ROS system, where the data will be stored waiting for the LVL2 decision. The ROS system is also responsible for providing partial event information or complete data fragments respectively to the LVL2 and the EB systems, via a multi-layer GE network, and for the deletion of the data belonging to rejected and already built events.

IV. The ATLAS Event Builder

The main commitment of the ATLAS Event Builder is to assemble and format complete events, collecting all the data fragments from the ROS system. The components involved in this process are the DFM and the SFIs. The DFM task is to supervise the Event Building process assigning events to the Event Builder nodes, load-balancing the farm. The SFIs are only responsible for the correct assembling of the events and the data transmission to the Event Filter.

Given the output rate of the second level trigger and the foreseen average event size of ∼1.5 MB, the Event Builder will have to pull from the ROS system ∼4.5 GB/s of data fragments, build complete events and send them, at same data rate, to the Event Filter.

In the next sections (Sec. IV-A and Sec. IV-B), the implementation and the features of the Event Builder components, the DFM and the SFI, are presented and discussed. Fig. 2 can be used as a reference for the network communications between those components.

A. DataFlow Manager

The DataFlow Manager is a multi-threaded application implemented is C++ running on a DFM node. Up to now 12 DFM nodes, sporting a dual 2.6 GHz AMD Opteron 252 CPU [4], have been installed and commissioned.

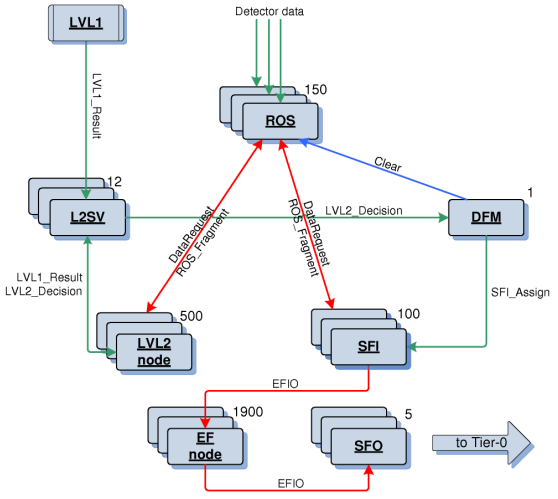

Fig. 2. Communications between the ATLAS TDAQ system components. The number close to each subsystem indicates the expected farm size for that component.

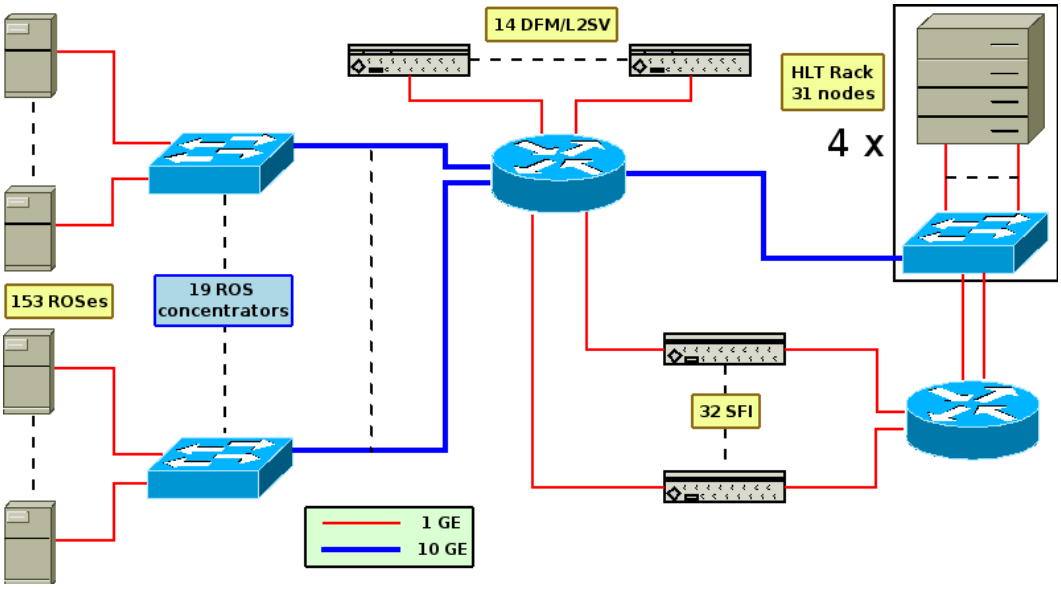

Fig. 3. Currently installed data flow TDAQ infrastructure. Data logging, on-line service and monitoring nodes as well as central and local file servers are not shown.

The DFM application receives network messages containing information about LVL2 rejected and accepted events. Moreover, for commissioning purposes, the DFM can either be directly triggered by the LVL1, effectively by-passing the LVL2 system, or internally generate accepted events, which is useful for development and performance tests.

For each accepted event, the DFM allocates a SFI, according to its load-balancing algorithm, to which it sends a network message to initiate the build process.

Rejected events and already built events have instead to be delete from the ROS memories: the DFM bundles the identifiers of these events in network messages containing $\sim$100 events. Since these messages have to be sent to all the ROSes, the UDP multicast protocol is used. However, as UDP messages are sensible to packet losses, the DFM also keeps track of the oldest LVL1 identifier in the system and ships this information with the clear messages. Using this information, a garbage collection mechanism can be initiated in the ROSes in case of large memory consumption.

Even if a single DFM is sufficient to supervise the building process for the whole ATLAS, up to 12 are available during the commissioning period in order to allow for independent detector slices to run in parallel.

B. Sub-Farm Input

The Sub-Farm Input is a multi-threaded application implemented in C++ running on the event builder nodes (also called SFI nodes). 32 SFI nodes, equipped with two 2.6 GHz AMD Opteron 252 CPUs, are currently installed in respect of $\mathcal{O}(100)$ foreseen in the final system.

Following the event assignment by the DFM, the SFI application request and receive data fragments from the ROSes via network messages. In case where a ROS fragment is missing after a configurable timeout, the SFI application can issue further requests for it, that is the so called "re-ask" mechanism. However, after several consecutive failures, the event is anyhow built and flagged as incomplete event. For efficiency reason, the SFI can build events in parallel, but in order to avoid network congestion, due the simultaneous fragments sent by the ROSes, the SFI also limits the number of outstanding requests. Moreover requests are issued using a set of randomized ROS lists aiming for a better spreading of the load on the network.

The SFI also serves built event to the Event Filter. In particular the events are delete from the internal buffers only after the reception acknowledgement from the EF. Furthermore the SFI is able to write the data on disk, for commissioning purposes, and to provide sampled events to monitoring clients.

C. Network Protocols

Since the ATLAS Event Builder is completely based on the network infrastructure, the network protocols employed by the system deserve to be discussed. In the present implementation it is possible to independently configure the usage of either TCP/IP or UDP/IP for each message type (Fig. 2). The baseline solution foreseen the utilization of UDP/IP for data requests and replies between ROSes and SFIs and of TCP/IP for control messages, like event-assignment messages. For the clear messages, sent by the DFM to the ROSes, the UDP/IP multicast mechanism is used.

TCP/IP is a reliable, connection-based, protocol that insures message delivery and network-congestion handling. However these features comes at the price of possible scaling issues. In fact, into the TCP/IP stack, every connection has its own set of timers, used to handle packet loss and congestion avoidance. Therefore large resource utilization may occur when many connections are open concurrently. Moreover, the timer configuration cannot be set at the application level and the reaction of TCP/IP to packet losses is the reduction of the sending rate. Summarizing, the resources needed by the

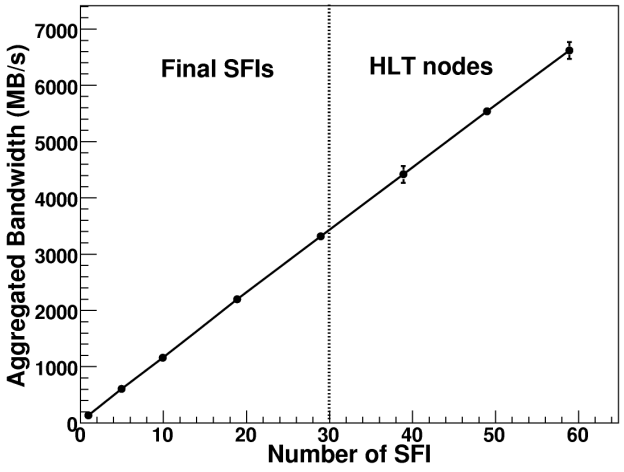

Fig. 4. Scaling of the Event Builder as a function of the number of SFIs employed. The measurement was taken with 124 ROSes and an event size 1.5 MB, without sending built events to the Event Filter. In order to extend our test capabilities, SFI applications were running also on the HLT nodes.

TCP/IP stack and the internal packet loss and congestion avoidance mechanisms can induce performance degradations when a large number of connections with an important data rate are foreseen, like in the communication between ROSes and SFIs. However, for the Event Builder control message, where a small data rate and message rate are expected, the reliability of the TCP/IP is exploited.

On the other hand, UDP/IP is an unreliable, connectionless protocol for which scaling issues will not occur. Hence it will be used for the communications between ROSes and SFIs, justified also by the fact that packet loss recovery and congestion avoidance mechanisms are implemented at the application level: respectively the re-ask and the traffic shaping mechanisms.

During all the measurements discussed in the following sections the baseline Event Builder network protocols were applied.

V. Present TDAQ Infrastructure

Presently the hardware resources needed for the final ATLAS data taking are not yet completely deployed. As reported in Fig. 3, the complete ROS system, including 153 PCs, is installed, while concerning the Event Builder, about one third of the builder nodes and 12 DFMs are deployed. Moreover, also four HLT racks, accounting for 31 nodes (dual-CPU quad-core Xeon "Clovertown" 1.86 GHz [5]) each, are installed. These racks can act both as LVL2 and EF processing farm thanks to their double network connection. In Fig. 3 also a layout of the multi-layer GE network, base of the data acquisition system, is shown. All the presented hardware is already commissioned and tested in several configurations, both in stand-alone mode as well as in fully integrated systems during cosmic-ray data-taking sessions. We should point out that not all the installed resources are included in Fig. 3: in particular central and local file servers, on-line service and monitoring machines as well as data-logging nodes and control network infrastructure are not presented.

VI. Event Builder Performance

Even if the complete TDAQ system is not yet completely deployed, performance measurements are performed for all the subsystems in order to understand scalability issues and disclose possible bottlenecks or needed optimizations as well as to drive the purchase of the remaining hardware. In this contest, we performed different measurements concerning the evaluation of the current Event Builder performance, the exploitation of the now largely diffused multi-core CPU architecture and the integration of the LVL2 trigger and the Event Builder.

A. Scaling

We measured the scaling of the Event Builder system with 124 ROSes, providing a total event size of 1.5 MB, and 1 to 59 SFIs. In fact, in order to extend our test capabilities, HLT nodes have been used to run SFI applications. During the measurement the SFIs deleted the built events immediately, without sending them to the Event Filter.

As shown in Fig. 4 a perfect scaling has been obtained up to 59 SFIs. The Event Builder system was able to build events at a rate of about 4.5 kHz, corresponding to 6.5 GB/s pulled out of the ROS system. Indeed each SFI adds 78 Hz and 114 MB/s respectively to the total event rate and the aggregated bandwidth. This means the SFI application is able to fully exploit a GE link. It has to be noticed that also the HLT nodes, although having a very different architecture in respect of the SFI nodes, perform very well running the SFI application. This result led to the double SFI approach discussed in the next section.

Summarizing, we were able to exceed the foreseen Event Builder bandwidth of 4.5 GB/s with only two third of the building nodes. This is a very encouraging result even taking

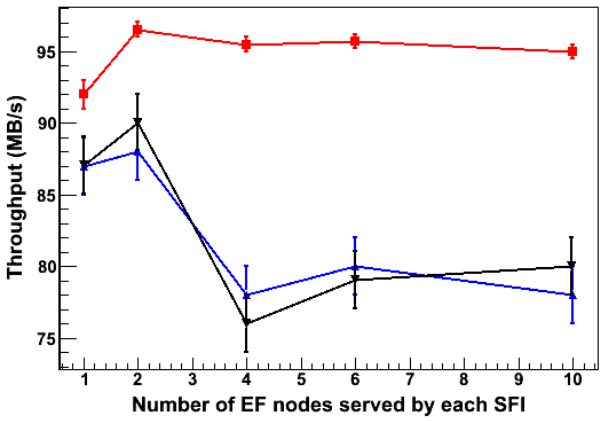

Fig. 5. Throughput of the double SFI approach in respect of a normal SFI as a function of the number of served Event Filter nodes. The line with squared markers corresponds to the throughput of the reference SFI, while the line with triangle markers correspond to the two SFI applications running on the same HLT node equipped with a multi-NIC card. See Sec. VI-B for the details.

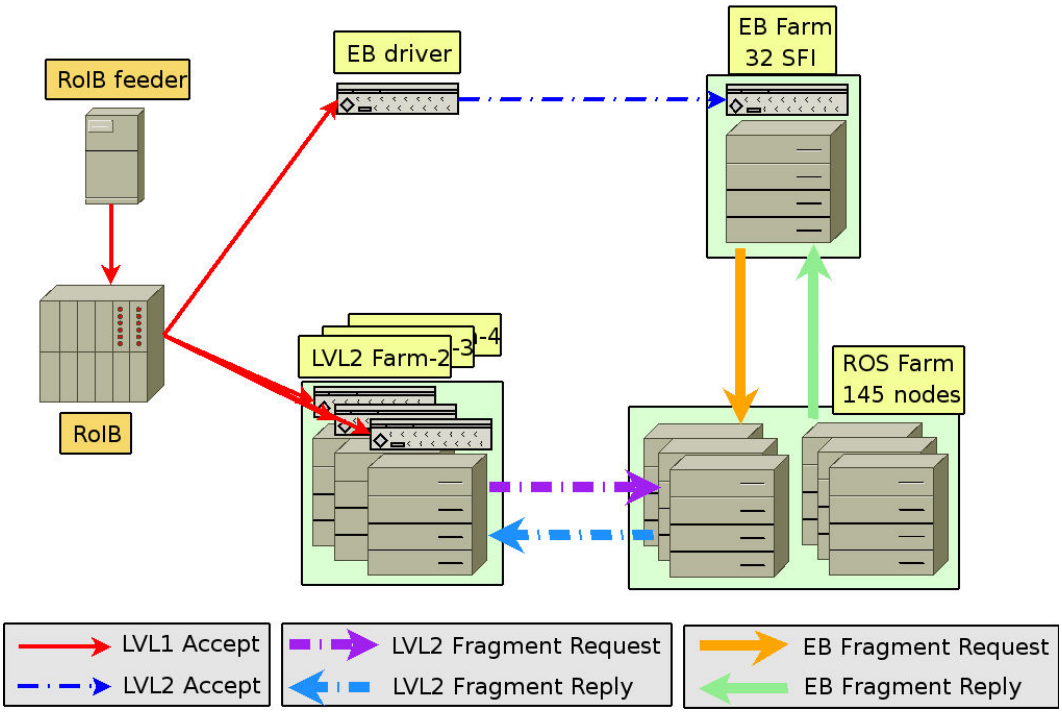

Fig. 6. Layout of the system used to introduce the LVL2 load into the system measuring the effect on the Event Builder. In order to independently tune the subsystem working points, having hence a larger control on test conditions, a custom topology was used. The LVL2 system was configured to reject all the events, while a configurable Event Builder driver provided the accepted events to the Event Builder. The system was initiated by the RoIB feeder application pushing fake RoI information into the RoIB.

into account the 10% performance decrease expected when sending data to the Event Filter [6].

B. Double SFI Approach

As described in the Sec. VI-A, we observed the same performance running the SFI application on the final SFI nodes and on the HLT nodes, even if the machines present different CPU architectures. Given this result we decide to evaluate the possibility of running more than one SFI application on a single machine equipped with a multi-core processor. Clearly such a node would need an extension of its network capabilities beyond the single GE link since a single SFI application is able to completely exploit such a link.

In order to evaluate this setup, we equipped a HLT node with a quad-NIC GE card, connecting two links toward the ROSes and two links toward the Event Filter. At the operative system level, we exploited the bonding mechanism provided by the linux kernel to present, at the application level, the two link pairs as single, improved, network interfaces. The linux bonding mechanism is not perfectly efficient, however it largely simplified the setting up of the measurement giving us the possibility of understanding the potential of this approach.

In Fig. 5 the SFI throughput as a function of the number of served Event Filter nodes is presented for both a reference SFI, running on a final node, and the two SFI applications running on the HLT machine. The data were provided by 32 ROSes, for a total event size of 1.1 MB. In this configuration, the reference SFI is able to throughput $\sim$95 MB/s, while the two applications running on the double SFI node, for more than a few EF clients, provide $\sim$80 MB/s each. Hence, the multi-core node presents a total throughput of 160 MB/s, limited by the output bonding efficiency.

Even if a performance degradation is introduced running two applications on the same node, they are still able to exceed the foreseen 60% link utilization (70 MB/s) of the final system. Moreover, the degradation is due to bonding driver performance, hence different solutions can be implemented to avoid this bottleneck: for example the links can be configured so that they use different IP-address families.

VII. Accounting for LVL2 system

The LVL2 trigger system, in order to take its decision, collects partial event data, roughly 1-2%, from the ROSes, via the same network used by the Event Builder. Hence, one can suppose scalability issues due to the interference of the two systems.

The LVL2 uses a sequence of "feature extraction" and "hypothesis testing" algorithms with the aim of rejecting events as soon as possible with the minimum CPU utilization [7]. Moreover, by dealing with RoI data only, the LVL2 system also tries to reduce the network requirements. In any case, it is expected the LVL2 will fetch $\mathcal{O}(2)$ GB/s from the ROSes.

Due to the algorithm mechanism cited above, the LVL2 asks to the ROSes RoI information in steps. In each of them it is expected that a few fragments or ROLs per ROS will be requests. From this point of view, an important parameter to evaluate the load induced by the LVL2 on the ROS system is the LVL2 ROS-request rate. The RoI mechanism moreover produces a

very non-uniform request rate on the ROS farm, with the so-called "hot ROSes", belonging to interesting regions of the detector, like the forward electromagnetic calorimeter, experiencing request rates up to ∼12 kHz.

Due to the aim of the LVL2 system of rejecting events as soon as possible, the parameters describing the LVL2 data collection features, like the number of steps per event and the number of ROLs per step, are expected to present long-tailed distribution, with small average values driven by the large number of rejected events.

A. Measuring the LVL2 effect on the Event Builder

Due to the special features of the LVL2 system discussed above, the best way to measure the effect of its full load on the system involves the usage of real selection algorithms. However this requires to deal with their processing latency and, since a sufficient computing power to achieve high rate with algorithms is not yet installed, such a test cannot be performed. Instead we decided to use a simplified LVL2 algorithm that randomly fetch data from the ROSes, without processing time, and for which the fundamental LVL2 data collection parameters (e.g. number of steps per event, number of ROLs per step, ...) are configurable. In this way it is not possible to exactly reproduce the ROS request rate distribution, hence we aimed in reaching the highest rate both on the LVL2 and the Event Builder, looking for possible system limits.

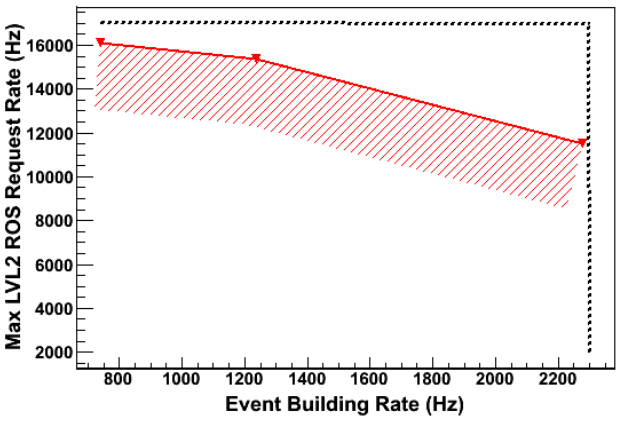

Fig. 7. Explored space in the plane defined by the Event Builder rate and maximum LVL2 request rate on the ROS system. Dashed lines define the limits of presently installed network bandwidth. In the area below the data, the system demonstrated to be able to always sustain the concurrent load of LVL2 and EB systems.

In Fig. 6 a layout of the system used to evaluate the effects of the LVL2 is presented. A part for using a simplified LVL2 algorithm, we also decided to implement a custom configuration, decoupling the LVL2 and the Event Builder. Three LVL2 farms (HLT racks), accounting for a total of ∼750 L2Pus, were rejecting all the events, while the Event Builder was driven by an independent driver for which was possible to configure the accept ratio and hence the EB rate. The system was initiated by the RoIB feeder application pushing fake RoI information into the RoIB.

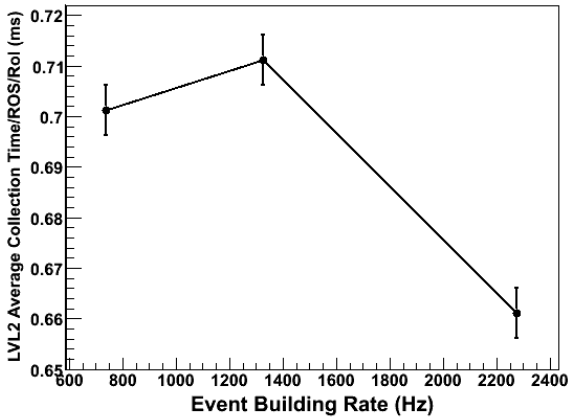

Fig. 8. Average time needed by the LVL2 to fetch data from the ROS system. The collection time is measure of the ROS load.

Fig. 7 presents the explored space in the plane defined by the Event Builder rate and the maximum LVL2 request rate on the ROSes. The last parameter has been chosen since the most loaded ROS drives the building latency. The test did not expose major system limits. Indeed the observed rate decrease in Fig. 7 is actually due to a correlation effect between the LVL2 and the Event Builder introduced by the RoIB, where a non-optimized firmware is still used.

The fact that major limits were not reached can also be pointed out by the collection time measured at the LVL2 (Fig. 8). The LVL2 collection time is the time needed to fetch the data from the ROS system and it is expected to increase with the ROS load. Since all along our test the collection time was almost constant with an average value smaller than 1 ms, we can conclude the ROS system was far from its limits.

In the extreme conditions reached during the measurement the LVL2 system was fetching 2.3 GB/s from the ROSes, while the Event Builder was pulling 3.6 GB/s, limited by the presently installed bandwidth (dashed lines in Fig. 7). In the final system the expected aggregated bandwidths respectively are $\mathcal{O}(2)$ GB/s and 4.5 GB/s.

VIII. Conclusions

One third of the ATLAS Event builder farm is installed and commissioned. The Event Builder system includes two main components: the DFM, managing the building process, and the SFI, the actual building application.

Extended Event Builder scalability tests from 1 to 59 SFIs demonstrated a perfect behaviour: each SFI application is able to almost exploit a full GE link with a a data rate of 114 MB/s. Using only two third of the final EB farm, we measured an aggregated data rate of 6.5 GB/s, exceeding the foreseen rate of 4.5 GB/s in the final ATLAS running conditions.

A configuration with two SFI applications running on the same multi-core node, equipped with extended network capabilities, has been successfully tested. A 15% throughput degradation was observed, in respect of the condition in which two nodes are used, due to the inefficiency of the network bonding. The setup was anyhow able to exceed the per-link

network utilization required in the final system enabling the saving of a processing node.

Initial measurements including both the LVL2 and the Event Builder systems did not expose major limits. We were not able to reach the final network and ROS load, mostly because of the currently limited hardware resources. However, after the forthcoming expansions of the hardware infrastructure, it will be possible to perform deeper and conclusive tests.

REFERENCES

[1] ATLAS Collaboration, *ATLAS Technical Proposal* (in CERN/LHHCC/94-43, LHCC/P2). Geneva: CERN, 1994.

[2] ATLAS Collaboration, *High-Level Trigger, Data Acquisition and Controls TDR*, CERN/LHCC/2003-022

[3] B. Green, G. Kieft, A. Kugel, M. Müller and M. Yu, "ATLAS trigger/DAQ RobIn prototype", *IEEE Trans. Nucl. Sci.*, vol. 51, 465–469, June 2004.

[4] AMD Corporation, [Online]. Available: http://www.amd.com/opteron

[5] Intel Corporation, [Online]. Available: http://www.intel.com/xeon

[6] H. P. Beck et al., "Performance of the final Event Builder for the ATLAS Experiment", in *Proc.* 15^{th} IEEE RTC 2007, Fermilab, Batavia, IL, USA, 2007.

[7] K. Kordas, "ATLAS High Level Trigger Infrastructure, ROI Collection and Event Building", in *Proc.* 15^{th} CHEP, Mumbai, India, 2006.