The Second Level Trigger of the ATLAS experiment at CERN's LHC

M. Abolins, S. Armstrong, J.T. Baines, M. Barisonzi, H.P. Beck, C.P. Bee, M. Beretta, M. Biglietti, R. Blair, A. Bogaerts, V. Boisvert, M. Bosman, H. Boterenbrood, D. Botterill, S. Brandt, B. Caron, P. Casado, G. Cataldi, D. Cavalli, M. Cervetto, M. Ciobotaru, G. Comune, A. Corso-Radu, E. Palencia Cortezon, R. Cranfield, G. Crone, J. Dawson, B. Di Girolamo, A. Di Mattia, M. Diaz Gomez, R. Dobinson, A. dos Anjos, J. Drohan, N. Ellis, M. Elsing, B. Epp, Y. Ermoline, F. Etienne, S. Falciano, A. Farilla, M.L. Ferrer, D. Francis, S. Gadomski, S. Gameiro, S. George, V. Ghete, P. Golonka, S. González, B. Gorini, B. Green, M. Grothe, M. Gruwe, S. Haas, C. Haeberli, Y. Hasegawa, R. Hauser, C. Hinkelbein, R. Hughes-Jones, P. Jansweijer, M. Joos, A. Kaczmarska, K. Karr, A. Khomich, G. Kieft, E. Knezo, N. Konstantinidis, K. Korcyl, W. Krasny, A. Kugel, A. Lankford, G. Lehmann, M. LeVine, W. Li, W. Liu, A. Lowe, L. Luminari, T. Maeno, M. Losada Maia, L. Mapelli, B. Martin, R. McLaren, C. Meessen, C. Meirosu, A.G. Mello, G. Merino, A. Misiejuk, R. Mommsen, P. Morettini, G. Mornacchi, E. Moyse, M. Müller, Y. Nagasaka, A. Nairz, K. Nakayoshi, A. Negri, N. Nikitin, A. Nisati, C. Padilla, I. Papadopoulos, F. Parodi, V. Perez-Reale, J. Petersen, J.L. Pinfold, P. Pinto, G. Polesello, B. Pope, D. Prigent, Z. Qian, S. Resconi, S. Rosati, D.A. Scannicchio, C. Schiavi, J. Schlereth, T. Schoerner-Sadenius, E. Segura, J.M. Seixas, T. Shears, M. Shimojima, S. Sivoklokov, M. Smizanska, R. Soluk, R. Spiwoks, S. Stancu, C. Stanescu, J. Strong, S. Tapprogge, L. Tremblet, F. Touchard, V. Vercesi, V. Vermeulen, A. Watson, T. Wengler, P. Werner, S. Wheeler, F.J. Wickens, W. Wiedenmann, M. Wielers, Y. Yasu, M. Yu, H. Zobernig, M. Zurek[*]

Presented by

André dos Anjos, Federal University of Rio de Janeiro, RJ, Brazil[†]

Abstract--The ATLAS trigger reduces the rate of interesting events to be recorded for off-line analysis in 3 successive levels from 40 MHz to ~100 kHz, ~2 kHz and ~200 Hz. The High Level Triggers and Data Acquisition System are designed to profit from commodity computing and networking components to achieve the required performance. In this paper, we discuss Data Flow aspects of the design of the Second Level Trigger (LVL2) and present results of performance measurements.

I. INTRODUCTION

The ATLAS experiment [1] will use very high energy proton-proton collisions provided by CERN's LHC for a wide research program including a search for Higgs bosons. The detector is composed of specialized sub-detectors to register the properties of the particles produced: an inner detector inside a magnetic field of 2 Tesla measuring tracks, a Calorimeter to measure energy and finally a muon detector.

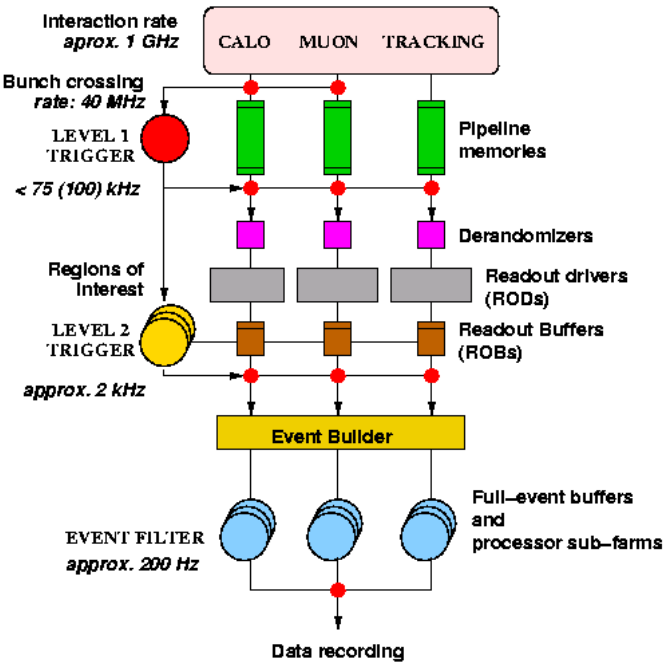

Figure 1: Principal components of the Data Flow and HLT systems.

Due to the high event rate and noisy background conditions, the experiment will be equipped with a powerful trigger system subdivided into 3 levels [2] as shown in Figure 1.

[*] The ATLAS Dataflow and High-Level Triggers Groups : http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/HLT/AUTHORLISTS/nss2003_HLT_DF.pdf

[†] Contact address: UFRJ/COPPE, Laboratório de Processamento de Sinais (LPS), CP 68504, 21945-970, Rio de Janeiro/RJ, Brazil – e-mail: Andre.dos.Anjos@cern.ch

The First Level Trigger (LVL1) is directly connected to the detector front-end electronics of the calorimeter and muon detectors. Data of accepted events are stored in pipeline memories connected to read-out drivers (RODs) and made available to the High-Level Triggers (HLT) through ~1,600 read-out buffers (ROBs). The LVL1 will be built using custom hardware components in order to cope with the high input rate of 40 MHz and will deliver a maximum output rate of 75 kHz, upgradeable to 100 kHz.

The output of the LVL1 trigger defines regions in the detector where the signal exceeds programmable thresholds. These so called Regions of Interest (RoI) are used as seeds for the Second Level Trigger (LVL2). By only looking at data in the LVL1 RoIs, it is also possible to reduce the amount of data transferred into LVL2 processors to less than 2% of the total event data (1.5 MB) while still keeping efficient classification algorithms. LVL2 selection algorithms request data from variable number of RoIs, typically 1 or 2. The average spans ~18 ROBs.

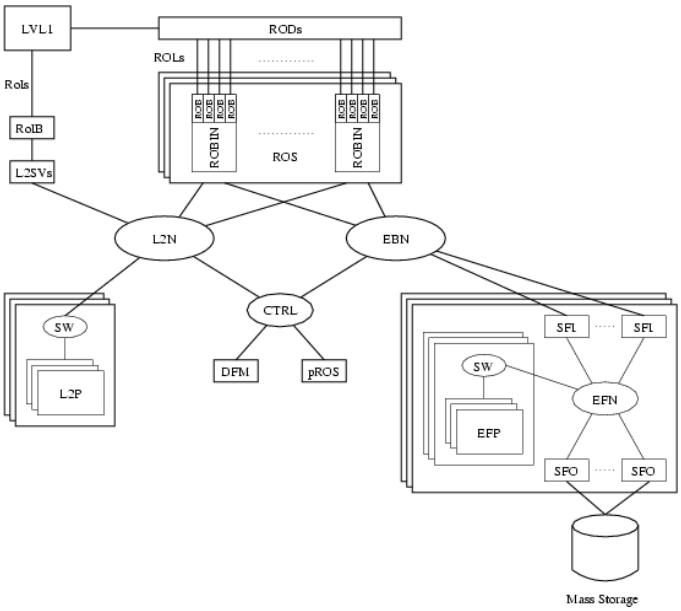

Figure 2: Baseline implementation of the Data Flow and HLT systems.

It is expected that the average processing time per event to be in average 10 ms [2]. This will require 500 dual-CPU processing nodes. At this configuration each node should deliver a trigger decision at a rate of ~200 Hz, requiring an input bandwidth of ~3.2 MB/s.

The last trigger level is the Event Filter (EF). After a LVL2 accept, the full event data is assembled and redirected into specialized processing farms, where elaborated filtering algorithms are applied. This level will still reduce the trigger rate to no more than ~200 Hz. If the event is accepted, it is recorded to permanent storage for later off-line analysis.

II. The Second Level Trigger

The LVL2 system is part of the ATLAS Data Flow [3] as illustrated in Figure 2. The main components are the Read-out System (ROS), the LVL2 system, the Event Builder (EB) and the Event Filter.

Data for events accepted by the LVL1 trigger are sent to the ROSs and, in parallel, information on the location of RoIs identified by LVL1 is sent to the LVL2 Supervisor. The LVL2 Supervisor sends the LVL1 Result to a LVL2 Processor (L2P), where the event selection is performed. Using the LVL1 Result as guidance, specialized LVL2 algorithms request a sub-set of the event data from the ROSs to perform the event selection. For events accepted by LVL2, details are sent to a "pseudo" ROS (pROS) to be included in the event. The L2P sends the LVL2 Decision back to the LVL2 Supervisor, which forwards it to the Event Builder.

A.The LVL2 Subsystem

The LVL2 trigger provides the next stage of event selection after the hardware-based LVL1 trigger. It uses RoI guidance received from LVL1 to seed the validation and enhancement of the LVL1 using selected full granularity event data. Components involved in the LVL2 process are the RoI Builder (RoIB), the LVL2 Supervisor (L2SV), the LVL2 Processors, the ROS and the pROS. The RoIB assembles the fragments of information from the different parts of the LVL1 trigger and transmits the combined record to a L2SV. The L2SV selects a L2P for the event and sends the LVL1 information to that processor waiting for the LVL2 Decision to be returned. The L2P runs the LVL2 event selection software (ESS), requesting event data as required from the ROSs and returns the LVL2 Decision to the L2SV. For events accepted by LVL2, output data to be included in the event are sent to a pROS. This LVL2 Result is later gathered as part of the event data with the raw detector data and can be used by the EF as a seed or by off-line reconstruction algorithms.

B. The Event Processing framework

The LVL2 components are applications based on the ATLAS Data Flow framework that provides a set of tools that enable these applications to exchange event data, to be remotely controlled, to report messages and to be monitored. Applications in this framework exchange data through a set of predefined messages that may run over standard TCP/IP, UDP or raw Ethernet. Both the framework and the applications are designed using object-oriented techniques [4] and fully implemented in C++.

Figure 3 shows the abstract design of the software running in an L2P. It is composed of an input task that receives data from the L2SV and a set of concurrent Worker tasks each processing one event. The synchronization mechanism between the input and the Workers is an event queue, which is protected against concurrent access. Each Worker, after extracting one LVL1 result out of the input queue, will start the event selection chain.

The L2P tasks, including Workers, have been implemented as threads. This approach is interesting because Symmetric Multi-Processor (SMP) architectures can be more efficiently

exploited to achieve the required LVL2 trigger rate at a reduced cost with a smaller farm as opposed to a process-based approach. In addition, the application running at the L2P becomes insensitive to latencies as the ESS may run in one thread whilst waiting for data in another thread.

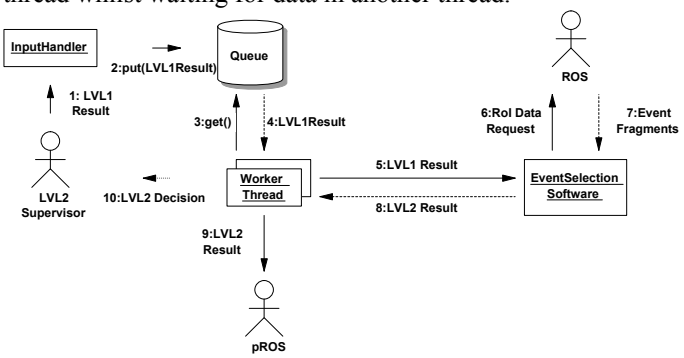

Figure 3: UML collaboration diagram of the application running on the L2P.

Multi-threading imposes a few restrictions to the software that runs concurrently. Initially it needs to be made thread-safe, which means that concurrent write access to globally visible variables should be protected by process locks. Software that runs in multiple threads of execution also needs to be made thread-efficient which means that the number of locks should be minimized to avoid thread inter-locking. Excessive locking may slow down the processing considerably by enforcing sequential execution of large parts of the threads.

It is by far not evident to make C++ code thread-safe and even less to make it thread-efficient. For example, by using the latest GNU C++ compiler, locks are systematically used in memory allocators for standard container objects. Naive use of lists, maps or even vectors in parallel threads of execution may considerably reduce the performance of applications. The solution here is to change the standard memory allocator by an allocator without locks. However, this method will fail when libraries pre-compiled with standard allocators are used.

The ESS is dynamically loaded. It is developed in an off-line framework, restricted to use only services also available on-line and respecting the programming guidelines implied by the L2P application design. Many of the existing off-line tools available are also implanted into the application running at an L2P [5]. For testing, there is also a possibility to emulate the complex ESS. Measurements presented below were made with the dummy version of the ESS. The dummy ESS implementation emulates data processing, including the multi-step analysis and data requests.

III. Measurements

A. Critical Parameters of the readout system

Some measurements varying parameters that are critical for the performance of the RoI Data Collection have been made. Though the number and average size of the RoIs depend on physics, the ROB concentration factor (the number of ROBs contained within one ROS which may be read in one operation) is a parameter that may be chosen to optimize the readout. The optimum number of parallel worker threads depends on the number of CPUs in one node and the idle time incurred when waiting for ROSs to send their data. The latter depends obviously on the size of the RoI as well as the ROB concentration factor. In the measurements below, the L2P collects only the RoI data, i.e. it does not execute any LVL2 algorithm.

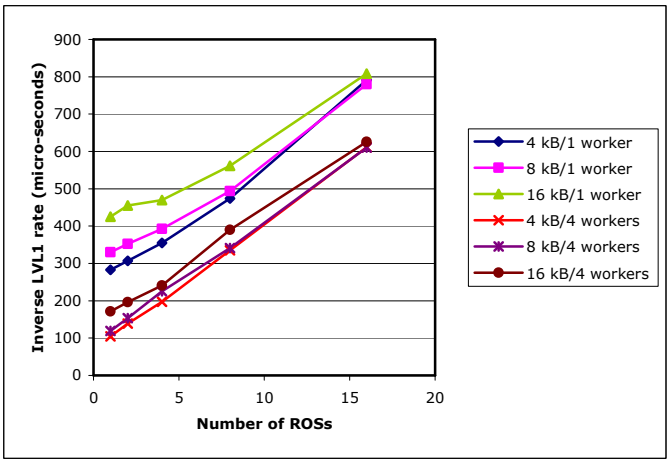

Figure 4: Summary of the performance of the RoI data collection for various combinations of RoI sizes (in bytes) and worker threads. The plot shows the inverse of the input L2SV rate as a function of the number of ROSs that contributes with RoI data.

The contribution of RoI Data Collection to the L2P event processing time as a function of the RoI size and the number of ROSs over which the RoI is distributed is shown in Figure 4. The range of measurements presented in this figure corresponds to the currently expected RoI sizes and their distribution over ROBs, e.g. an e/γ RoI in the Liquid Argon (the electromagnetic ATLAS Calorimeter section) detector is expected to be distributed over 13 to 16 ROBs [2] and have a size of approximately 16 kB. It can be observed from these results that the time taken to collect RoI data contributes, in the worst case, to less than 10% to the average event processing time budget.

Figure 5 shows the performance of a single L2P as the inverse of the LVL1 accept rate sustained versus the number of worker threads. In this measurement, the RoI has a fixed size of 16 kB. The different curves represent the RoI being collected from 2, 4, 8, or 16 ROSs. For example, the upper curve represents the results of collecting a 16 kB RoI from sixteen ROSs, each of which contributes 1 kB to the RoI. The results indicate that the optimum number of worker threads, in this set-up, is approximately three and is independent of the number of ROSs from which the RoI is being collected. In addition, the results show that for the conditions of this measurement and for three worker threads, the collection of RoI data contributes less than 10% to the average event processing time of 10 ms.

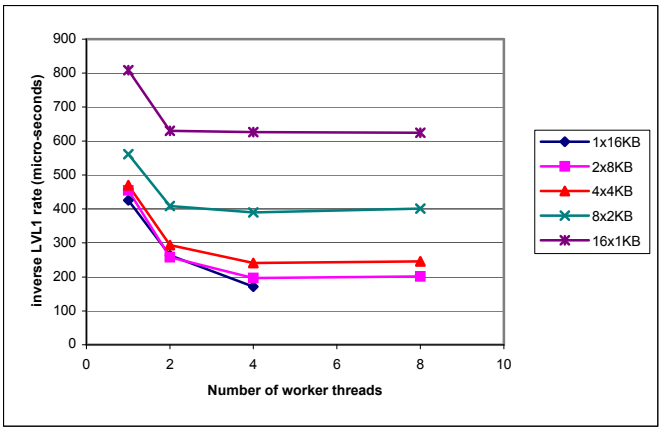

Figure 5: The inverse of the LVL1 rate in a LVL2 system as a function of the number of worker threads for different RoI sizes.

B.Scalability

The measurements so far presented have shown the rate at which a single L2P, with a dummy ESS, can collect RoI data. The achieved performance depends on the RoI size, the ROB concentration factor and the number of worker threads. For the system, however, it is also necessary to demonstrate that when many L2Ps are requesting data at the same time from the same set of ROSs the performance of the RoI collection does not degrade unacceptably.

For this test, L2Ps with dummy ESS were again used, although it should be noted that the request rate for data per L2P is much higher that it would be if the real classification algorithms were included. Thus, in these measurements, each L2P generates more than ten times the normal request rate. Similarly, the requests are sent to a small number of ROSs due to machinery availability, so that again it is important to ensure that the total request rate did not exceed the capability of these components.

The tests were run in a testbed consisting of three L2SVs, four ROSs and up to eleven L2Ps. All the nodes in this testbed were PCs in the same configuration as before, interconnected via a Gigabit Ethernet switch. For this test each ROS was configured to emulate 12 readout links (ROBs), to give a total of 48 ROBs across the four ROSs. For each request the L2P chose one of the 48 ROBs at random, and in the case of 6 ROBs per RoI and 24 ROBs per RoI requested the following 5 and 23 ROBs respectively, wrapping round across the ROSs as necessary. Figure 6 presents the results of these measurements. It shows the rate of events handled by each L2P and the rate of requests to each ROS as a function of the number of L2Ps used in the testbed for an RoI distributed over 1, 6 or 24 ROBs (with 1.5 kB per ROB). The plot shows that the rate per ROB decreases as the number of L2Ps is increased, by ~20% for 1 ROB per RoI and ~40% for 24 ROBs per RoI. However, with 11 L2Ps in the test the request rate per ROS is 17 kHz for 1 ROB per RoI and 8 kHz for 24 ROBs per RoI, both much higher demands on a ROS compared to the final system. In addition, the total RoI request rate in the small test set-up is already 70 kHz and 11 kHz for the two RoI sizes respectively. Given these extreme conditions the fall from scaling can be seen to be relatively modest, and it seems reasonable to assume that with more typical conditions a full-sized system will show acceptable scaling. This interpretation is reinforced by the fact that running similar tests with the ROSs replaced by custom hardware ROS emulators shows scaling within better than 10% over the same range of RoI data traffic.

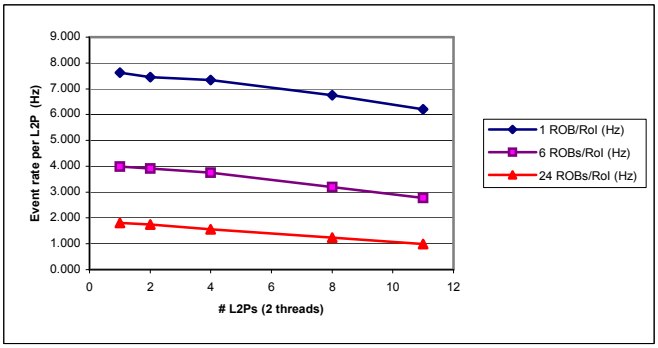

Figure 6: Event rate per L2P as a function of the number of L2Ps in the system for a different number of ROBs per RoI. Each ROB contributes with an equal amount of data for the RoI in every test.

Figure 7 shows the LVL1 rate sustained by the L2SV as a function of the number of L2Ps that it controls. In this measurement the computer running the L2SV is also connected to a gigabit Ethernet switch on which a number of PCs are also connected, each of which were executing the L2P code equipped with a dummy ESS. The L2SV was not connected to the LVL1 prototype; instead it emulated the receiving of events from this subsystem. As can be seen from the figure, the L2SV can sustain a LVL1 accept rate of 32 kHz when distributing RoI information to a single L2P and the dependency on the number of L2Ps is 1%. Thus, based on today's prototype implementation and PCs, ten L2SVs are sufficient to support a LVL1 accept rate of 100 kHz.

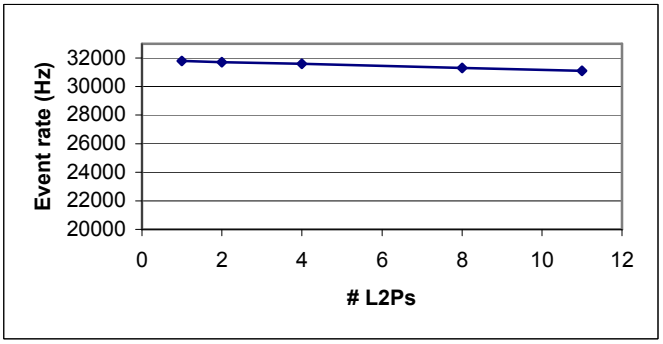

Figure 7: L2SV sustained LVL1 accept rate versus the number of L2Ps.

IV. Conclusions

We have presented a LVL2 system that uses a farm of conventional PCs running ESS guided by results from the LVL1 trigger. Standard Gigabit Ethernet is used to connect the individual processors to the readout system. All measurements were made using a connection-less protocol, UDP or raw Ethernet. Although not explicitly shown in the measurements, the performance differences were minimal.

It has been shown that the I/O capacity of each node is largely sufficient. The CPU resources devoted to RoI data collection use a modest fraction of the allocated time budget for a large range of RoI sizes. Though concentration of multiple ROBs in one ROS (resulting in the transfer of larger data slices if the ROB to detector mapping is optimized) is more efficient, even an architecture with a concentration factor of one would still give acceptable results.

Event parallelism using multiple threads is effective, allowing the use of multi CPU nodes to save on cost, cooling and floor space. Additional threads are necessary to compensate the readout latency.

The scalability has been measured for a system with a small number of nodes. It has been shown that, with unrealistically high data rates into each processor and a small number of ROSs, LVL2 rates as high as 70 kHz can be obtained with a relatively modest fall of scaling.

V. References

[1] ATLAS Collaboration, "ATLAS: Technical Proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN," CERN/LHCC/94-43, 1994.

[2] ATLAS Trigger and Data Acquisition Collaboration, "ATLAS High-Level Triggers, DAQ and DCS Technical Design Report," July 2003.

[3] Jos Vermeulen et al., "The Baseline Dataflow System of the ATLAS Trigger & DAQ," 9th. Workshop on Electronics for LHC Experiments, Amsterdam, Netherlands, October 2003.

[4] Szymon Gadomski et al., "Experience with multi-threaded C++ applications in the ATLAS Dataflow software," Computing in High-Energy Physics 2003, San Diego, California, USA, March 2003.

[5] Werner Wiedenmann et al., "Studies for a common selection software environment in ATLAS: from Level-2 Trigger to offline reconstruction," Nuclear Sciences Symposium 2003, Portland, Oregon, USA, October 2003