PROCEEDINGS OF SCIENCE

The DAQ/HLT system of the ATLAS experiment

André dos Anjos*
University of Wisconsin, Department of Physics, Madison, USA
E-mail: andre.dos.anjos@cern.ch
on behalf of the ATLAS TDAQ Community (see [1])

The DAQ/HLT system of the ATLAS experiment at CERN, Switzerland, is being commissioned for first collisions in 2009. Presently, the system is composed of an already very large farm of computers that accounts for about one-third of its final event processing capacity. Event selection is conducted in two steps after the hardware-based Level-1 Trigger: a Level-2 Trigger processes detector data based on regions of interest (RoI) and an Event Filter operates on the full event data assembled by the Event Building system. The detector read out is fully commissioned and can be operated at its full design capacity. This places on the High-Level Triggers system the responsibility to select only events of highest physics interest that will finally reach the offline reconstruction farms.

This paper brings an overview of the current ATLAS DAQ/HLT implementation and performance based on studies originated from its operation with simulated, cosmic particles and first-beam data. Its built-in event processing parallelism is presented and discussed.

ATL-DAQ-PROC-2009-006
22 April 2009

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7, 2008
Erice, Italy

*Speaker.

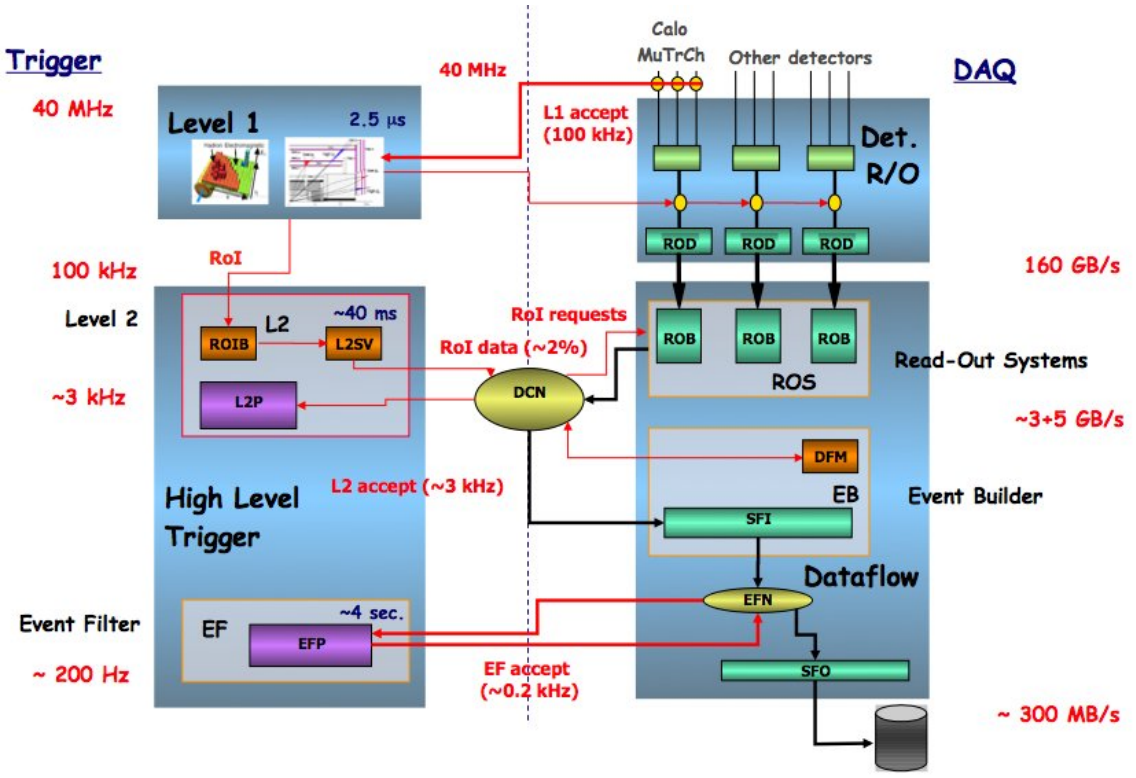


Figure 1: Overview of ATLAS Trigger and DAQ systems.

1. Introduction

The ATLAS experiment [2] at CERN, Switzerland counts on its Trigger and Data Acquisition (TDAQ) systems to select and store interesting physics reactions out of the Large Hadron Collider's (LHC) 14 TeV proton-proton bunch crossings, happening once every 25 ns (40 MHz). Each complete event read out in ATLAS is composed of roughly 1.6 Megabytes of data. Out of the large input rate only ∼200 events per second can be recorded to permanent media, given the experiment's read out infrastructure (ROS) and offline computing capacity. An overview of the ATLAS TDAQ is depicted in Figure 1.

To achieve this design goal, the ATLAS Trigger is divided in three sub-levels. The First Level Trigger (L1) is built out of custom hardware, being capable of a rate reduction of three orders of magnitude. It uses signals from the ATLAS Calorimeters and Muon Detectors to reach up to 75 kHz (upgradeable to 100 kHz) output event rate. It does that by using simple scanning algorithms to detect high-p_T phenomena and to discard less interesting reactions. Such high-p_T objects have their region of interaction with the detector, or simply *Region of Interest* (RoI) annotated. In case of an accept, a summary of the L1 search known as the L1 Result (L1R) is passed to the High-Level Triggers system (HLT) and selection continues from this point. The L1 also triggers the read out system to buffer detector data while HLT refines its decision.

The HLT is composed of a Second Level Trigger (L2) and a third level or Event Filter (EF). By design, L2 delivers a maximum of 3 kHz output rate to EF of which only about 200 Hz survives

to tape. The L1 is connected to L2 through the means of a input gate known as the RoI Builder (RoIB). The RoIB has the task of fanning out the L1R to up to 20 L2 supervisors (L2SV) which, in turn, have access to up to 500 L2 Processors (L2Ps). Each L2P is equipped with the ATLAS Trigger Software and can perform selection, starting from the L1R and requesting RoI-centered data to the ROS for achieving a decision. Because only data around RoIs ($\sim$2% of the total event data) is retrieved from the ROS, L2 is able to sustain such a high-input rate.

If the event is accepted by L2, a signal is sent to the Event Building system (EB) and data is collected from all ROS buffers and assembled before EF selection is performed. In case of a rejection, these aforementioned buffers are cleared and no event assembly happens. After event building, each EB processor known as Subfarm Input (SFI) waits for a transfer request from one of the 1800 EF processors (EFP).

Each EFP, like in L2, is equipped with the HLT selection software. Based on the refined search by L2, it is able to reduce the rate of events to be recorded by a factor of 10, writing about 300 Megabytes per second to permanent storage. Each EFP is connected to a few Subfarm Output (SFO) processors, which take events that have survived the selection process and puts them into files separated by stream types [3]. ATLAS stream types are set by the HLT processors at L2 or EF, additively.

2. System performance

Except for the connection between detector front-end and the L1/RoIB to the ATLAS DAQ/-HLT, detector data processing and messaging are implemented by means of commodity computing and network. Currently, this system is composed of Gigabit Ethernet switches and x86 multi-core computers running Linux.

The DAQ/HLT software was written to benefit from the multi-core architectures of today [4]. Where possible, DAQ applications make use of multiple threads of control for increased efficiency. L2 and EF HLT applications though, process events in multiple independent processes inside L2Ps and EFPs respectively. L2 Processing Unit (L2PU) is the name given to each instance of the L2 HLT application running on a L2P. EF Processing Task (EFPT) is the equivalent for EFPs. The number of L2PUs run by L2P is set to match the number of cores available in each machine; the same is valid for EFPTs with respect to the EFPs.

As far as concerns connectivity, detector data traffic routing is implemented using 3 separated logical networks: L2, EB and EF. Communication between nodes is configurable and can be run using either standard UDP or TCP over IP. The computing nodes are only connected to the relevant networks. For example L2Ps connect to the ROS through the L2 Network; SFIs build events through the EB Network, but are also connected to the EF Network in order to serve events to EFPs. Because of the unknown optimum balance factor between L2 and EF processing capacity that minimizes dead-time, a set of HLT processors is connected to both L2 and EF networks and can be setup to act either as L2Ps or EFPs. These are called Exchangeable Processing Units (XPUs). At the present state, there about 850 XPUs, 63 SFIs, 5 SFOs and all the 154 ROS computers required to read the $\sim$1600 buffers which are connected directly to the detector read out. XPUs available are dual quad-core architectures and have 16 Gigabytes of RAM available to accommodate the HLT selection software. The remaining machines are either dual double or single-core.

In order to test this setup, it is possible to preload simulated or pre-acquired data simultaneously at the ROS computers and the L2SVs. In this case, the latter operates in self-triggered mode feeding the available L2Ps as fast as L1Rs can be analyzed. Equally simple is resetting the system configuration to take triggers from L1 and data from the detector instead of preloading. These setups are used in the following tests and explained in more details bellow.

2.1 Testing with simulated data

Testing with simulated data is necessary to stress the system limits, since no beam with the required structure (25 ns bunch-crossings) is available. For the presented results, the existing XPUs were split to make up a complete DAQ/HLT system composed of 360 L2Ps and 310 EFPs. This amounts to about, respectively, 70% and 17% of the L2 and EF final farm sizes. About 60 SFIs were used for EB and 5 SFOs were assuring the flow of events from the EF to disk. The L2 farm was managed by 4 L2SVs, each managing in turn $\sim$90 L2Ps.

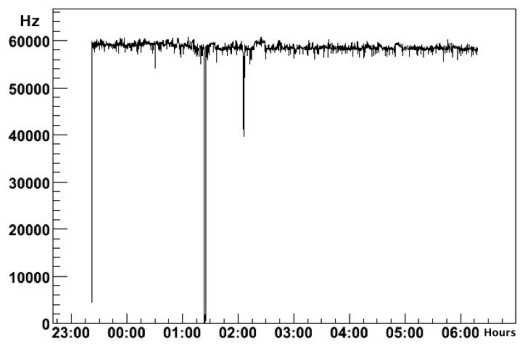
This apparently unbalanced configuration for the HLT is possible taking into account the rejection ratios programmed into the L2 system, to not overflow the available EB bandwidth. L2 and EF are both loaded with a trigger menu that is optimized for a $10^{31} cm^{-2} s^{-1}$ luminosity. The simulated data preloaded contains a mixture of events providing both signal and background events which is compatible with the expected collider production. The average event size in the mixture is 800 Kilobytes, mainly due to the lack of pile-up simulation.

The plots in Figure 2 summarize a few important results for tests done with simulated data. Figure 2a shows the combined L2 farm throughput. This shows a constant throughput 60 kHz during many hours of operation. Figure 2b the equivalent plot for the total event building throughput. Taking into consideration the average size in the preloaded sample, this rate is equivalent to about 3.5 Gigabytes per second bandwidth utilization. The spike in the rates, which is noticeable at around 1:30 p.m. at the plots, was due to disk-intensive cron jobs that were executed in parallel on all machines in the cluster. This problematic behavior has been fixed since these tests.

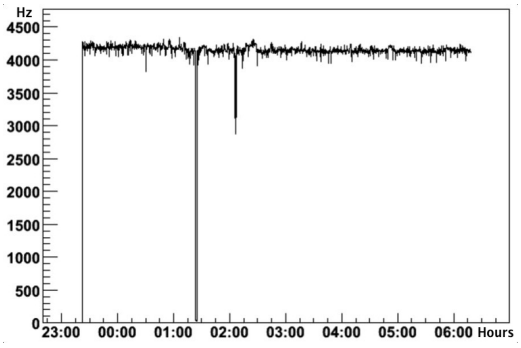
HLT thresholds were set to reject around 93% of the events in L2 and 62% of the events in the EF. The average execution time to reach a L2 reject is $\sim$38 ms and $\sim$125 ms for the EF as can be verified in Figures 2c and 2d. The average time it takes to accept events in L2 and EF are, respectively, $\sim$136 ms and $\sim$260 ms. The overall processing time for L2 and EF can be extrapolated from these numbers and are, respectively, $\sim$45 ms and $\sim$176 ms. These numbers are in agreement with design expectations of the ATLAS TDAQ. The low latency for EF event processing can be explained by the low luminosity condition in which the simulated events were generated. With increasing luminosity and, by consequence, detector activity, it is expected that the latency of event processing in EF grows to match typical reconstruction timings, which are in the order of a few seconds.

2.2 Testing with cosmics

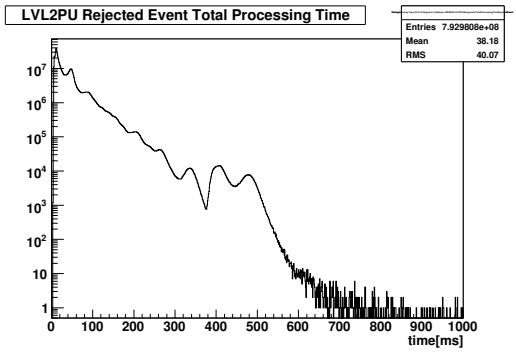
Testing with the whole ATLAS experiment infrastructure is possible using muons originated from cosmic ray interactions. In this mode, the ATLAS detector is configured for normal operations an L1 is adjusted for triggering using either its muon and/or calorimetry infrastructure, opening up a reasonable number of configuration possibilities. Random L1 triggers may be mixed in to test



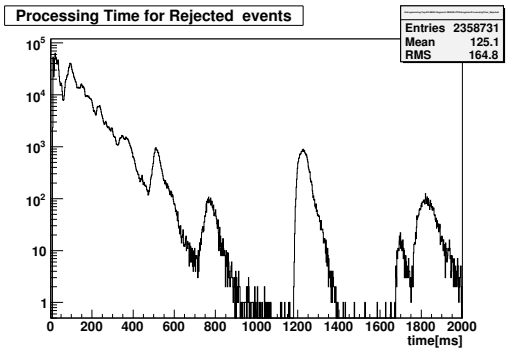
(a) L2SV rate



(b) EB rate



(c) L2 processing time



(d) EF processing time

Figure 2: Results for timing tests using simulated data.

the trigger selectivity and stress the detector readout electronics. In this case, simple cosmic muon detection algorithms are placed into the HLT processors and used to discard random triggers.

The experiment has been operated in this condition for nearly 2 months without major problems. Figure 3 shows the number of events taken as a function of the time, in days. Each stream (trigger mode) is represented by a different color in such a plot. As can be verified, about 213 million events have been taken. The average size of these events is 2.1 Megabytes. The total size of data taken amounts to approximately 450 Terabytes.

3. The ATLAS HLT and multi-core technologies

3.1 Context

Event processing inside the ATLAS HLT is programmed sequentially for both L2 and EF. It is bootstrapped by the arrival of the precedent level's processing result and a set of feature extraction and hypothesis making algorithms is run one after the other to as quickly as possible reach a decision. The strategy is optimized for early rejection.

The development of the HLT infrastructure for ATLAS has seen major changes in the commodity computing industry with the development of multi-CPU (a.k.a. SMP) architectures and, since a few years, introduction of multi-core x86 CPUs. These phenomena have compensated for the so-called end of the *Frequency Scaling Era*, where programs created would just get faster by

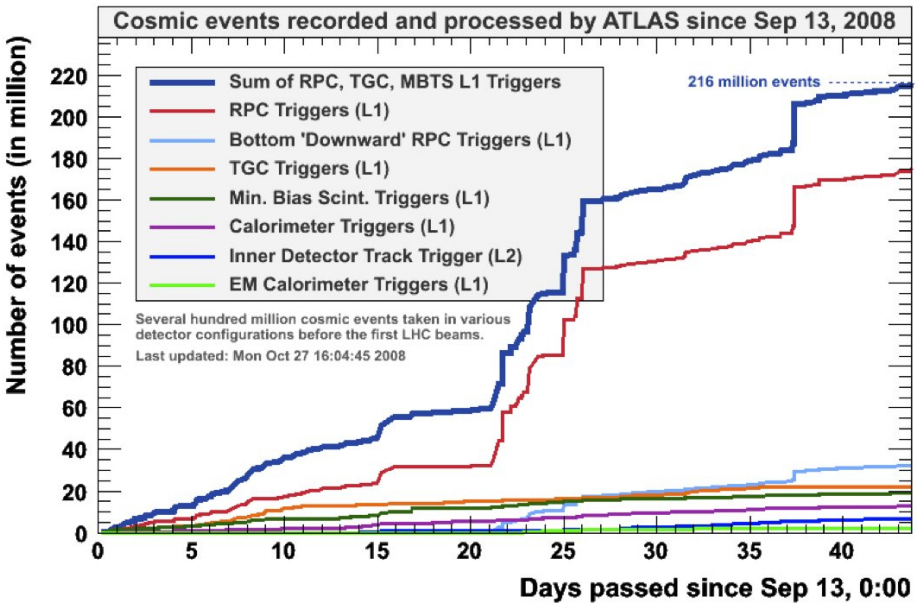


Figure 3: Data streams during cosmics data taking.

waiting for the next generation of computer chips. In these new times, higher throughput should be obtained by simultaneous execution of many tasks. Because of the nature of triggering in High-Energy Physics, specially taking into consideration the sequential selection design by the ATLAS Trigger software, the most effective way to increase the event through by parallelization is by processing each event in a independent computer task.

3.2 Developments

Two main courses could be chosen taking into consideration the context described above, one which implements event processing in isolated processes (MP) and a second option that uses multi-threading (MT) to achieve parallelism. In the latter mode of operation, a single HLT process would be able to handle many events in parallel. Which to choose depends, as usual, on a number of trade-offs concerning application control limitations, configuration requirements and maintainability, just to mention a few.

Multi-threading allows the re-use of the current processing context (program variables) to implement synchronization and data transfer mechanisms, which makes programming easier and the resulting code typically faster. Unfortunately, the benefits of MT don't go much further. Presently there are very few tools to allow the detection of synchronization problems in MT programs and common knowledge in this area is very limited, which is of fundamental impact in large projects such as the ATLAS software.

Secondly, MT applications require that all code used in their context is *thread-safe*. A piece of code is thread-safe if it functions correctly during simultaneous execution by multiple threads, in particular considering multiple concurrent accesses to shared data. This is another drawback for large projects since all code produced and directly or indirectly used must be checked for conformity. In this context, an important aspect of the ATLAS Trigger design that comes into play

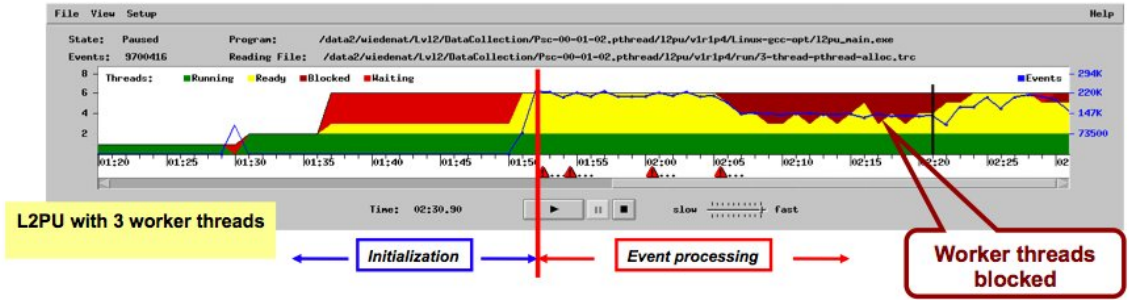


Figure 4: Thread inter-locking during event processing.

is code sharing. To simplify analysis and minimize maintenance [6], trigger and offline analysis share several tens of thousands lines of code, many of which were written during the Frequency-Scaling Era, when typical analysis applications were meant to be executed in a single-threaded way.

Still, thread safety does not satisfy a basic triggering condition which is time efficiency. Despite access is made safe, shared data may still be needed to pursue the processing of an event and its access may be blocked by other threads. This requirement has been called *Thread Efficiency*. Blocking concurrent threads for a resource represents a slow-down in the overall speed-up of the MT application leading to a sub-optimal result [7].

It is a laborious task to detect all points in a program where a potentially inefficient piece of code is being made use. It may be even harder to fix it as it may lie on libraries that are just being used by routines being executed, to which the programmer may have no access to. Figure 4 shows an example of lock between concurrent threads for a L2PU executing HLT selection code in 3 parallel threads - each thread processing an event. The horizontal axis describes the time since the start of the application and the vertical axis the number of threads available. The different colors mark the different state of each thread in the application. The green color on the bottom of this graphic means those threads are running while the yellow section marks how many threads are ready to run, but are not running. Dark red indicates the referred threads are blocked waiting for a shared resource. After the process has been started, only one thread was executing at each time, with the other two waiting for a locked resource. Such a resource was later identified as a global memory manager used by the GNU C++ Standard Template Library (STL) [4]. These types of problems are not exclusive to any STL implementation and do happen, many times, in quite unexpected ways.

Multi-processing can cure problems observed with MT by relaxing on thread safety and efficiency requirements, at the expense of more computing power. Taking in consideration current dual quad-core platforms, going MP means increasing by 8 the number of applications to control and configure. More generally, MP imposes scalability requirements on TDAQ infrastructure which must follow the computer chip industry trends. If data sharing is not implemented between applications executed in the same computing node, memory requirements are also expected to increase.

Because of the nature of the processing and the statistical independence of physics events flowing inside trigger systems, a perfect scaling is expected between the throughput achieve while

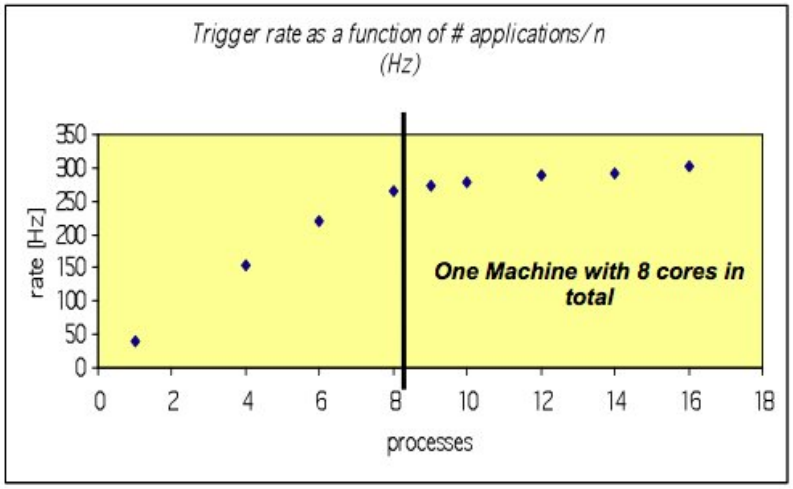


Figure 5: Speed-up of the ATLAS HLT as a function of the number of processes used in a single computing node.

executing code in a single task or multiple tasks excluded, of course, hardware limitations. Figure 5 shows the speed-up obtained by running the HLT code using an MP approach. The machine in question had 8 cores. As can be seen, the processing rate scales quite linearly as one increases the number of HLT processes running in the node. By running 9 or more processes the rate does not increase much more.

4. Conclusions

The ATLAS DAQ/HLT has reached the required maturity to take new physics, to be provided by LHC in mid-2009. It has been shown that this system can operate stably for many hours and fits perfectly under its design specifications. L2 can sustain up to 60 kHz which represents about 80% of its design capacity by counting only on 70% of the specified hardware. EB can assemble events stably at $\sim$4 kHz for the same tests. Because of limiting interconnect conditions between the SFIs and the EF present in the system during these tests, only a fraction of the EB bandwidth could be utilized. Later tests with EF disconnected witnessed much higher EB bandwidth usage, above 6 Gigabytes/s (it was originally designed for 5 GB/s) with 60 SFIs, which represent only 60% of the expected hardware.

Tests with cosmics have been used to fine-tune interfaces between the DAQ/HLT, L1, the detector read out front-end and the offline analysis processing farms. Since the begin of operations in August 2008, nearly 1 Petabyte of data has been taken, 450 Terabytes from September 2008. These data were recorded by the SFO farm in nearly 400000 files. Streaming has been proven to work correctly and without major problems.

In particular for the HLT processing strategy, multi-processing is the current preferred solution and is deployed in both L2 and EF. For pure DAQ applications, like SFIs, L2SVs or SFOs, multi-threading is used to optimize data-acquisition to its maximum. The use of MP for HLT event processing is due to the lack of support tools for MT development, common expertise and support

from software packages used in the project considering both thread safety and efficiency. MP has proven to scale well in current multi-core architectures while tests with the infrastructure have not shown major problems to control and configure the larger number of applications. Techniques for optimizing memory usage inside HLT processing nodes are currently under study.

References

[1] The ATLAS Trigger/DAQ Authorlist, version 1.0, **2008**, CERN Document Server, ATL-DAQ-PUB-2008-004

[2] The ATLAS Collaboration, G. Aad et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, **2008** JINST 3 S08003

[3] A. Battaglia et al, *The Data-Logging System of the Trigger and Data Acquisition for the ATLAS Experiment at CERN*, **2008**, Nuclear Science Symposium and Medical Imaging Conference

[4] S. Gadomski et al, *Experience with multi-threaded C++ applications in the ATLAS dataflow software*, **2003**, Conference for Computing in High-Energy and Nuclear Physics

[5] A. dos Anjos et al., *The Second Level Trigger of the ATLAS Experiment at CERN's LHC*, **2004**, IEEE Transaction on Nuclear Science, volume 51, number 3, pages 909-914

[6] W. Wiedenmann et al., *Studies for a common selection software environment in ATLAS : from the Level-2 Trigger to the offline reconstruction*, **2004**, IEEE Transactions on Nuclear Science, volume 51, number 3, pages 915-920

[7] A.S. Tanenbaum, *Modern Operating Systems*, **1992**, Prentice Hall