Software Validation Infrastructure for the ATLAS High-Level Trigger

Enoque Ferreira de Lima D.[*][t], Adorisio C.[f], Ask S.[x], Beauchemin P.[n], Bell P.[x], Biglietti M.[h], Coccaro A.[g], Ehrenfeld W.[e], Faulkner P. J. W.[a], George S.[p], Goncalo R.[p], Hamilton A.[v], Igonkina O.[m], Kirk J. H.[q], Kwee R.[i] Masik J.[x], Mincer A.[l], Monticelli F.[s], Oliveira Damazio D.[b], Omachi C.[j], Panikashvili N.[y], Perez Reale V.[d], Potter C.[k], Quinonez Granados F.[o], Reinsch A.[z], Rodriguez Patarroyo D. J.[r], de Seixas J. M.[t], Sinev N.[z], Strom D.[z], Sutton M.[w], Ventura A.[u], Winklmeier F.[c], Zhao L.[l]

[a]*Birmingham Particle Physics Group;*[b]*Brookhaven National Laboratory;*[c]*CERN;*[d]*Columbia University;*[e]*DESY;*[f]*Dipartimento di Fisica dell' Università della Calabria and INFN, Cosenza;* [g]*Dipartimento di Fisica della Università di Genova and INFN, Genova;*[h]*Dipartimento di Scienze Fisiche, Università di Napoli 'Federico II' e INFN, Napoli;* [i]*Humboldt Universitaet, Berlin;* [j]*Kobe University;*[k]*McGill University;* [l]*New York University;*[m]*Nikhef;*[n]*Oxford University;* [o]*Pontificia Univ. Catolica de Chile;*[p]*Royal Holloway College, University of London;*[q]*Rutherford Appleton Laboratory;*[r]*Univ. Antonio Narino, Theor. Phys.;*[s]*Universidad Nacional de La Plata;* [t]*Universidade Federal do Rio de Janeiro;*[u]*Università del Salento and INFN, Lecce;*[v]*Universite de Geneve;*[w]*The University of Sheffield;*[x]*University of Manchester;*[y]*University of Michigan;* [z]*University of Oregon, Eugene, Oregon;*

E-mail: `dferreir@mail.cern.ch`

The ATLAS trigger will need to achieve a 10^{-7} rejection factor against proton-proton collisions, and still be able to efficiently select interesting events. After a first hardware-implemented processing level, the final event selection is done by the high-level trigger (HLT), implemented on software. With more than 100 contributors and around 250 different packages, a thorough validation of the HLT software is essential. This paper describes the existing infrastructure used for validating the HLT software.

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

[*]Speaker.

1. Introduction

CERN has its current main focus at the Large Hadron Collider (LHC), which is being commissioned and expected to produce proton-proton collisions soon. ATLAS [1] (A Toroidal LHC ApparatuS) is a general-purpose particle detector at the LHC. The nominal bunch crossing rate at ATLAS is 40 MHz, with on avarage 23 inelastic proton-proton collisions per bunch crossing at the design luminosity of $1 \times 10^{34} cm^{-2} s^{-1}$. Due to the large amount of data generated, the high background rate and the limited bandwidth to mass storage, the Trigger System was developed to reject uninteresting events, as soon as possible.

The Trigger System is implemented as a three-level filter. The first level is implemented in hardware and uses only the calorimeters and muon detectors with coarse granularity to select Regions of Interest (RoI) to be fully examined by the next levels. The High-Level Trigger (HLT) comprises the Level 2 and Event Filter, which are implemented in software and analyse only the RoIs selected by the previous level[1].

As the LHC plans to have its first collisions soon, it is necessary to validate all sub-systems and ensure that there is no flaw in hardware, software or in the physics analysis. In the case of the HLT, the authors have developed common tools aiming at identifying problems early in the development process to avoid propagating them to the software running online.

2. ATLAS Software Development

The ATLAS HLT development profits from the fact that the HLT algorithms can be run in the offline environment, allowing tests to be done without the full online infrastructure. Therefore, it is possible to reutilize the ATLAS Offline Software Validation Tools to do important tests, simplifying the validation task. As a consequence, a complete description of the ATLAS HLT Validation is not possible without describing the Offline development model. The ATLAS software is composed of a set of interdependent projects, to ease code management and formalize package dependencies. Each night, the ATLAS software is built for different platforms and different branches. A *validation* build provides a buffer to test new code before adding it to the more stable *development* build. The *nightly* builds are controlled by the NIghtly COntrol System (NICOS) [2], which includes a web interface to display all the build results.

After the compilation and the unit tests of the packages performed by NICOS, additional tests are run to determine if the releases have severe software problems. These tests can be run using the ATLAS Testing Nightly (ATN) infra-structure, which runs small reconstruction jobs. Larger tests can be run using the Run Time Tester (RTT) infra-structure, generating useful plots for spotting physics inconsistencies. The ATN tests certify a release, by checking the *error* codes on the test jobs, comparing the log files with a reference and checking the exit code of the software. As the tests run on a very verbose mode[2], looking at the differences between the log files is a very useful technique to detect changes and understand whether they were intended or were in fact a mistake. ATN tests also compare the ROOT [3] histograms generated during the tests. RTT tests,

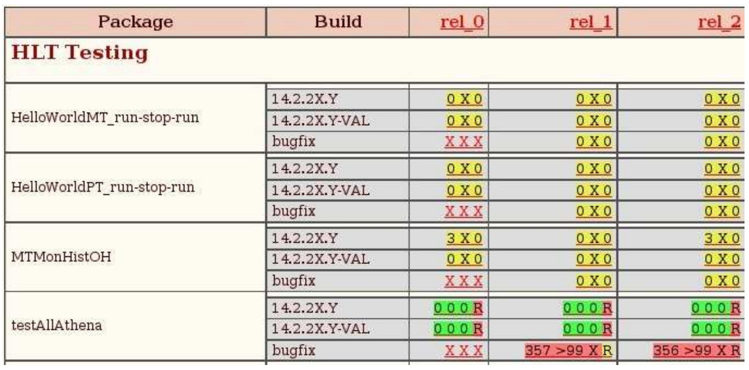
[1]The Event Filter can also perform a full scan of the detector at a lower rate.

[2]In verbose mode, the software prints many steps of its processing, and a thorough view of the processing is available in the log files.

on the other hand, run on much larger datasets and can also estimate the physics performance of the ATLAS software. The RTT infrastructure shows the results on a web page and sends e-mails reporting the status of the tests. When an error is detected, a new bug [4] report is created by the Validation Group.

2.1 ATLAS HLT Validation

With the purpose of providing the trigger developers with a simple interface dedicated to the HLT validation, a separate web interface [5] summarises the trigger ATN tests results. It is updated automatically every hour, showing all relevant information on a single web page, with a self-explanatory colour system indicating error, warning or success status. The ROOT files, log files, error messages and other information are available from a simple menu system, allowing developers to track problems faster. A screenshot of this system is included in Figure 1.



Package	Build	rel_0	rel_1	rel_2
HLT Testing				
HelloWorldMT_run-stop-run	14.2.2X.Y	0 X 0	0 X 0	0 X 0
	14.2.2X.Y-VAL	0 X 0	0 X 0	0 X 0
	bugfix	X X X	0 X 0	0 X 0
HelloWorldPT_run-stop-run	14.2.2X.Y	0 X 0	0 X 0	0 X 0
	14.2.2X.Y-VAL	0 X 0	0 X 0	0 X 0
	bugfix	X X X	0 X 0	0 X 0
MTMonHistOH	14.2.2X.Y	3 X 0	0 X 0	3 X 0
	14.2.2X.Y-VAL	0 X 0	0 X 0	0 X 0
	bugfix	X X X	0 X 0	0 X 0
testAllAthena	14.2.2X.Y	0 0 0 R	0 0 0 R	0 0 0 R
	14.2.2X.Y-VAL	0 0 0 R	0 0 0 R	0 0 0 R
	bugfix	X X X	357 >99 X R	356 >99 X R

Figure 1: Screenshot of the Trigger summary page with test results

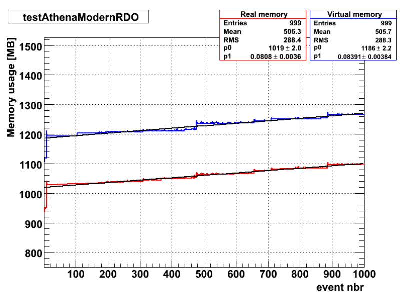


Figure 2: Plot of memory usage generated by PerfMon

Due to the limitation on each of the HLT levels' running time, it is also important to measure the execution time of the algorithms and ensure that it is, on average, less than 40 ms for the Level 2 and less than 4 s for the Event Filter. A monitoring of the time required by the algorithms can be done with the PerfMon [6] tool, which also has its results available on all RTT jobs.

Another important feature of the HLT software is that it should run continuously for O(10 hours) without crashes on around 2000 CPUs. One of the biggest challenges is the controlling of memory leaks. PerfMon also monitors the memory consumption per event, keeping track of the amount of memory that was allocated to each event. If a memory leak is detected, the software must be corrected immeadiately in order to gain stable online operation. Figure 2 shows a plot generated by PerfMon containing the memory usage per event, in which a clear memory leak is identified. There is also work being done on a tool to monitor long-term memory usage of the algorithms in RTT, which could be used to analyse the evolution of the software performance.

Another tool is DCube[7], an infra-structure developed to automatically compare histograms. It is configured via an XML file and the results are displayed on a web page. Many RTT tests use this infra-structure in order to automatically verify their latest results and notify developers faster.

3. TrigEgammaValidation package

One example of a validation method using the RTT infrastructure is the TrigEgammaValidation package. The purpose of this software is to validate the HLT electron and photon selection algorithms. This is accomplished by plotting variables relevant to the trigger decision at each step of the filtering software and comparing the results with those from a previous release. This tool is run in RTT on different simulated datasets and calculates also the HLT selection efficiency for each step involved in such filtering algorithms.

As part of each test, several statistical metrics are used to compare the histograms against a set of references. These metrics translate the histograms differences into a quantitative information. TrigEgammaValidation uses four methods to measure the similarity between histograms. The Kolmogorov-Smirnov test [8] measures the maximum distance between the histograms of the cumulative distribution functions of the variables. The Symmetrized Kullback-Leibler [9] [10] and the Jensen-Shannon [11] divergences use the concept of entropy from Information Theory to measure the similarity between histograms. The Quadratic Negentropy [12] measures the distance between the histograms of the probability density functions. Both the comparison results and the generated plots are available on the web for collaboration developers.

To compare the metrics, the bins of the TrigEgammaValidation-generated histograms on a stable release were changed by adding or multiplying their contents with a Gaussian random variable with a standard deviation equals to *Bin content* $\times$ *Noise level*, so that the amplitude of the *Noise level* can be changed and its effect can be analysed. Using this methodology, Figures 3 and 4 were generated on a stable release taking the mean of all histograms obtained from the HLT algorithms, using TrigEgammaValidation. We can observe that the Kolmogorov-Smirnov[3] plots are very sensitive to changes in the histograms[4], while the Kullback-Leibler and Jensen-Shannon metrics both increase slowly. It is also clear that after high noise levels the Kullback-Leibler, Jensen-Shannon and Kolmogorov-Smirnov methods can not detect big differences on histograms. The Quadratic Negentropy has a very poor sensitivity to noise, particularly to multiplicative noise.

4. Conclusion

Due to the importance of the High-Level Trigger validation in ATLAS, such validation is being performed systematically, controlling problems that range from compilation issues on the code up to physics performance. These tests have been handled in a common infra-structure with a flexible framework, which allows specific tests to be run in ATN or RTT for every new release. Systematic control of the trigger system is also done by comparing histograms using semi-automatic tests, through statistical metrics.

Acknowledgments

The authors would like to thank the ATLAS Software Infrastructure Team for providing many

[3]The Kolmogorov-Smirnov test has been adjusted to show a zero output in case of complete similarity and 1 for the biggest dissimilarity.

[4]Note that the X axis on the additive noise plot for the Kolmogorov-Smirnov metric has a different scale!

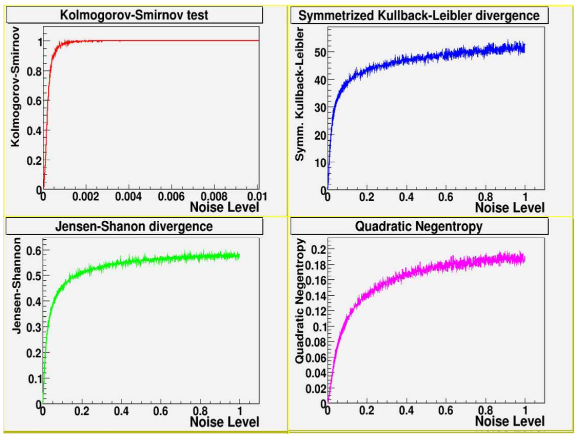


Figure 3: Effect of Gaussian additive noise to the different metrics versus noise level

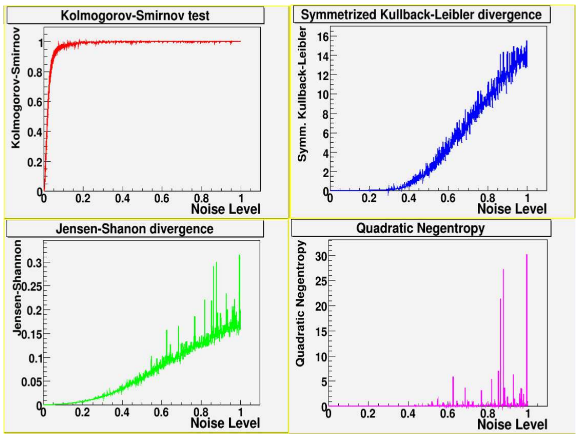


Figure 4: Effect of Gaussian multiplicative noise to the different metrics versus noise level

of the tools used in the trigger validation. The authors from Rio de Janeiro, Brazil, would like to thank the Brazilian agencies CNPq, CAPES, FINEP and FAPERJ for their support.

References

[1] The ATLAS Collaboration, G. Aad et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* (2008) 3 S08003; M. Bosman, *ATLAS Detector Status and Physics Startup Plans*, these proceedings.

[2] A. E. Undrus, *in (CHEP 03), 24-28 Mar 2003, pp TUJT006* [arXiv:hep-ex/0305087].

[3] ROOT System Web Page, `http://root.cern.ch/`, accessed on 2 December, 2008.

[4] Savannah Web Page, `http://savannah.gnu.org/`, accessed on 2 December, 2008.

[5] Trigger Summary Web System, `http://fernando.web.cern.ch/fernando/TriggerSummaryPage/TriggerHLTP1TestPage.html`, accessed on 2 December, 2008.

[6] S. Binet, P. Calafiura, W. Lavrijsen, A. Undrus, Performance Measurement and Monitoring for HENP Applications, poster at *CHEP 07*

[7] DCube Twiki, `https://twiki.cern.ch/twiki/bin/view/Sandbox/DCubeDoc`, accessed on 12 March, 2009.

[8] F. E. James, in *Statistical Methods in Experimental Physics*, 2^{nd} ed., World Scientific Publishing Co. (2006), p. 316-317.

[9] M. Tumminello, F. Lillo, R. N. Mantegna, Phys. Rev. E 76, 031123 (2007), *arXiv:0706.0168v1* [physics.data-an], 1 Jun 2007

[10] D. Johnson, S. Sinanović, *Symmetrizing the Kullback-Leibler Distance*, IEEE Transactions on Information Theory, 2001

[11] J. Lin, *Divergence Measures Based on the Shannon Entropy*, IEEE Transactions on Information Theory, Vol. 37, No. 1, January 1995

[12] J. Lee, T. Kim, S. Lee, *Robust Independent Component Analysis using Quadratic Negentropy*, p. 227-235, Independent Component Analysis and Signal Separation (2007)