The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

24 November 2011 (v3, 28 November 2011)

Measuring CMS Software Performance in the first years of LHC collisions

Gabriele Benelli, Balazs Bozsogi, Andreas Pfeiffer, Danilo Piparo and Vidmantas Zemleris on behalf of the CMS Collaboration

Abstract

The CMSSW software framework is a complex project enabling the CMS collaboration to investigate the fast growing LHC collision data sample. A software performance suite of tools has been developed and integrated in CMSSW to keep track of cpu time, memory footprint and event size on disk. These three metrics are key constraints in software development in order to meet the computing requirements used in the planning and management of the CMS computing infrastructure. The performance suite allows the measurement and tracking of the performance across the framework, publishing the results in a dedicated database. A web application makes the results easily accessible to software release managers allowing for automatic integration in CMSSW release cycle quality assurance. The performance suite is also available to individual developers for dedicated code optimization and the web application allows historic regression and comparisons across releases. The performance suite tools and the performance of the CMSSW framework during the first LHC collisions years will be presented in this talk.

Presented at *IEEE-nss-mic-rtsd2011: IEEE2011: Nuclear Science Symposium and Medical Imaging Conference*

Measuring CMS Software Performance in the first years of LHC collisions

Gabriele Benelli, *Member, IEEE*, Balazs Bozsogi, Andreas Pfeiffer, Danilo Piparo, and Vidmantas Zemleris *on behalf of the CMS Collaboration*

Abstract—The CMS software framework (CMSSW) is a modular object-oriented data analysis framework enabling the CMS collaboration to process and analyze the fast growing LHC collision data set. A software performance suite of tools has been developed and integrated in CMSSW itself to keep track of CPU time, memory footprint and event size on disk. These three metrics are key constraints in software development in order to meet the requirements considered in the planning and management of the CMS computing infrastructure. The performance suite allows the measurement and tracking of the performance across the framework, storing the results in a dedicated database. A web application is deployed to publish the results, making them easily accessible to software release managers and allowing for automatic integration in CMSSW release cycle quality assurance. The performance suite is also available to individual developers for dedicated code optimization and the web application allows historic regression and comparisons across releases. The performance suite tools and the performance of the CMSSW framework during the first LHC collisions years are described in this paper.

I. INTRODUCTION

THE Large Hadron Collider (LHC) [1] at CERN has successfully delivered a large number of collisions at the highest center of mass energies ever achieved by an accelerator. The Compact Muon Solenoid (CMS) experiment [2] is one of the four large detectors designed to investigate the LHC collision data set, exploring a rich physics program. The CMS SoftWare framework (CMSSW) is used to simulate Monte Carlo (MC) events, reconstruct LHC collision events and analyze them. In this paper, we will present the tools developed to measure and track the performance of CMSSW in terms of CPU performance, memory footprint and event size on disk. These key metrics affect directly the ability of CMS to fulfill its research goals while meeting its computing requirements [3]–[5].

Manuscript received November 22, 2011. This work was supported in part by the Eotvos Lorand University, Budapest (Hungary) and by the Erasmus student mobility program between Vilnius University, Lithuania and CERN.

G. Benelli is with the Department of Physics and Astronomy, University of Kansas, Lawrence, KS USA (telephone: +41 22 767 3956, e-mail: Gabriele.Benelli@cern.ch).

B. Bozsogi, is with the Faculty of Informatics, Eotvos Lorand University, Budapest, Hungary (telephone: +36 30 437 4131, e-mail: Balazs.Bozsogi@cern.ch).

A. Pfeiffer is with the CERN PH Department, 1211 Geneva 23, Switzerland (telephone: +41 22 767 5029, e-mail: Andreas.Pfeiffer@cern.ch).

D. Piparo is with the CERN PH Department, 1211 Geneva 23, Switzerland (telephone: +41 22 76 71 557, e-mail: danilo.piparo@cern.ch).

Vidmantas Zemleris is with the CERN PH Department, 1211 Geneva 23, Switzerland (telephone: +41 22 767 1691, e-mail: Vidmantas.Zemleris@cern.ch).

The performance measurements are an integral part of the CMSSW development cycle. During the first LHC collision years a large amount of CMSSW performance data was collected, and these measurements have steered the development process, guiding optimization. The CMSSW performance data are made readily available so that software release managers, developers, offline software and computing managers in CMS can exploit this information, each in their respective task. The application used to access the performance data will be described, together with a summary of the results of such measurements, from the first LHC collisions of 2009 to the latest high luminosity collisions of 2011.

II. THE CMSSW PERFORMANCE SUITE

The CMSSW framework has built-in services designed to provide simple and non-intrusive profiling of CPU time and of memory footprint as the application is run. It also provides a detailed analysis of the event contents stored in the Event Data Model (EDM) output files. Examples of these are the CPU time and memory framework services, which are unsophisticated and provide essential tools for monitoring performance. In order to pursue detailed investigations, more comprehensive tools, such as IgProf [6]–[7] and Valgrind [8]–[11] are used. These profiling tools provide more information than the built-in services, at the expense of a penalty in terms of execution time. A detailed description of all profilers is available [12].

A. Implementation

Both the built-in and the external profilers are integrated into a performance suite of tools, called PerfSuite, which facilitates their use, allowing automated procedures, simple customization and straightforward access to the measurement results. The goal of the CMSSW PerfSuite is to offer a clean and user-friendly interface that allows great flexibility in the profiling execution while granting easy access to the results.

In assessing the performance of a High Energy Physics (HEP) experiment software framework, it is important to note how such performance varies dramatically with the physics process and the processing step under consideration. For this reason the PerfSuite can profile all processing steps both individually and combined in such a way to mimic the experiment data operations workflows. A set of standard, representative physics processes, referred to as samples in the following, is chosen among the Release Validation [13] (RelVal) set of samples, to exercise several aspects of the code, from Event Simulation to Reconstruction and Trigger.

The RelVal samples are used to check the physics performance of the framework and they are the basis for release comparison. The PerfSuite design allows addition and modification of extra samples including the extension to actual collision and cosmic ray data.

The measurement process is fully configurable. The user can select samples, profilers and number of events to be run. This number is generally chosen to reduce statistical fluctuations while keeping the processing time within the constraints of release cycle testing.

B. Use cases

The three main PerfSuite use cases are:

- CMSSW Release Validation,
- CMSSW Release Integration Building (IB),
- Individual developer tests.

Both Release Validation and Integration Building are part of the CMSSW software development and validation cycle. While RelVal has a weekly cycle designed to test only main CMSSW releases, both for physics and software performance, the IB cycle is shorter, typically 12hrs long, and it is the process used to continuously integrate new code in the framework. For each of them, there are several release series being developed in parallel with various differences (architecture, open release for new features, production releases for reconstruction, simulation and analysis, etc.). Currently we have 5 such release series running in parallel. For each of them several performance tests are automatically run.

In RelVal profiling, two MC samples are used. The first, minimum bias (MinBias) is a less complex event to reconstruct, but the most common topology in LHC collisions. The second, top-antitop (TTbar) events is a complex event that exercises all aspects of event reconstruction. To reflect the complexity of real LHC collision data, simulated event pile-up collisions were added to the main TTbar process (TTbar PU), matching the low luminosity conditions with an average of 7 pile-up events.

The RelVal workflows mimic the data operations workflow for Monte Carlo simulation (Step1), event reconstruction (Step2) and fast simulation (FastSIM). It also includes a separate High Level Trigger (HLT) workflow, since this part of the framework, by design, has to be the most performing and profiling it together with other processing steps would make it difficult to monitor.

TABLE I. RELEASE VALIDATION PERFORMANCE PROFILING WORKFLOW FOR STEP1 (GEN-SIM-DIGI-L1-DIGI2RAW-HLT)

Profiling Test	MinBias ^(*)	TTbar ^(*)	TTbar PU ^(*)
TimeSize	100	100	100
IgProf	50	50	50
Memcheck	N/A	5	5

^(*) Number of events per sample per test

Table I and Table II show the tests and the number of events that are executed in RelVal profiling. The number of

events used in the Step1 workflow, reported in Table I, is much lower in all other steps, reported in Table II, since Monte Carlo simulation is by far the most cpu-intensive task for the framework.

Finally, for the individual developer use case, tests are usually run to evaluate the performance implications of a newly developed piece of CMSSW code or to investigate further a performance effect observed in the RelVal or IB performance results. In the latter case users can easily re-run RelVal tests varying number of events, steps or conditions.

TABLE II. RELEASE VALIDATION PERFORMANCE PROFILING WORKFLOW FOR STEP2 (RAW2DIGI-RECO), FASTSIM AND HLT

Profiling Test	MinBias ^(*)	TTbar ^(*)	TTbar PU ^(*)
TimeSize	8000	8000	8000
IgProf	500	500	500
Memcheck	N/A	5	5

^(*) Number of events per sample per test

III. PERFORMANCE RESULTS PUBLICATION

A great challenge in the software performance profiling of a large software framework is to integrate it in the development and validation cycle, while making the resulting huge amount of performance data easily accessible and usable.

The RelVal and IB use cases produce a wealth of performance measurements, given the large number of automatic tests and the multiple release cycles that characterize the rapid evolution of CMSSW. The performance information is gathered and analyzed, and the relevant information is then formatted in XML format and transferred to a dedicated relational database, for easy retrieval and comparison. A sophisticated web interface to the database has been set up, called PerfSuiteGUI [14], which is the main portal to all the software performance information related to CMSSW. This solution makes it easy for coordinators and developers to monitor and track the software performance in one place, both for general overview and for investigation of specific details.

An exhaustive presentation of all PerfSuite functionalities and results (more than 2000 profiling jobs, spanning over 150 CMSSW releases in two years) goes beyond the scope of these proceedings, so only a few examples of the type of information available will be given, with an emphasis on performance results.

Fig. 1 shows an example of the CPU time profiling information for reconstructing TTbar events in CMSSW version 3.3.6, the release used for the very first LHC collision data taken in late 2009.

Fig. 1. CPU time per event for the reconstruction of a TTbar event sample in CMSSW 3.3.6 release. The interface reports the test information at the top: CMSSW release, sample, processing step, conditions, event contents, event pile-up conditions and number of events. It also reports average, RMS, min and max CPU time and the event number of the maximum. The plot can be interactively zoomed-in by the user and the statistical information is recalculated on the fly. The CPU time versus event number graph is reported at the top and the CPU time per event distribution is reported at the bottom.

While the plots of Fig. 1 are already quite informative on the overall CPU time per event performance, more detailed information is provided. An interactive table shows the CPU time information in terms of CMSSW modules, i.e. linking the CPU time information to specific parts of the framework. This table allows sorting by average, minimum, maximum, RMS, first event and by fraction of the total event CPU time. For each framework module in the table, it is possible, by clicking on its name, to link to the information on the average amount of disk space per event produced by that module in the EDM output file. The performance suite also aggregates, for the reconstruction processing step, the information into "sequences", allowing developers to monitor the performance of sets of modules designed to perform a particular task, such as charged particle tracking or electron reconstruction. Finally, the CPU time results from the IgProf profiling are linked from the PerfSuiteGUI for in-depth analysis. These features enable developers and managers to spot the source of any CPU time performance deterioration or improvement, helping in the development and release cycle of the software.

Since the CPU time performance depends on the hardware on which CMSSW is profiled, the use of CPU time seconds is meaningless when comparing results obtained on different machines. The performance suite allows the user to toggle between CPU time seconds and HS06 seconds, using the HS06 [15] benchmarking score of the machine used for the profiling. HS06 seconds are calculated as:

$$\text{HS06 seconds} = \frac{(\text{CPU time seconds}) * (\text{Machine HS06 score})}{\text{Machine's number of cores}}$$

This feature is particularly important not only when looking at the performance regression across releases, but also in the

estimation of the collaboration CPU resources needs, since HEP computing CPU infrastructure pledges are nowadays based on the HS06 benchmarking metric. The perfSuite was actually used to establish and validate the HS06 benchmark in the HEPiX CPU benchmarking group [16].

Plots for virtual memory size (VSIZE) and for resident set size (RSS) per event, equivalent to the CPU time ones, are dynamically produced by the web application. Fig. 2 shows an example of VSIZE per event for the same TTbar events in CMSSW 3.3.6 shown in Fig. 1.

Fig. 2. VSIZE per event for the reconstruction of a TTbar event sample in CMSSW 3.3.6 release. The test information is reported at the top: CMSSW release, sample, processing step, conditions, event contents, event pile-up conditions and number of events. Also average, RMS, min and max allocated VSIZE and the event number of the maximum. The plot can be interactively zoomed-in by the user and the statistical information is recalculated on the fly. The VSIZE versus event number graph is reported at the top and the VSIZE distribution is reported at the bottom.

The plot in Fig. 2 gives a quick idea of how memory allocation is proceeding: one expects VSIZE to initially increase with event number until all containers have been initialized and filled, after that it should reach a plateau. Memory leaks are usually quite apparent in such a plot, with either a continuous leak (slow but constant increase in the top plot of Fig. 2), or a large jump upwards not followed by a jump downwards from de-allocation. Besides this type of plot, the other two memory-related types of results, IgProf Mem and Valgrind Memcheck, are also included in the PerfSuiteGUI application. A table with the collected IgProf Mem results allows the user to glance over the main features of the tests, and then access detailed information to track any memory increase to the actual C++ method in the CMSSW framework. For Valgrind Memcheck, the total number of memory errors is reported, together with a link to the full ASCII log, where memory errors can be analyzed.

Fig. 3. Compressed file size distribution by event content type for TTbar reconstruction output in CMSSW 3.3.6.

The file size on disk is reported in several ways: the physical size of the CMSSW EDM output root files, the file size of the EDM products stored in the output files organized by the CMSSW modules that produced them, or the same file size of the EDM products but aggregated by CMSSW event content type, i.e. by the chosen type of event output to write on disk. This information is reported in tabular form that, similarly to the CPU time one, can be interactively sorted, but also as pie charts that show, for example in Fig. 3, the compressed and uncompressed file size on disk per event content type. This graphical representation helps spotting large changes in the event contents, while the tables help find the actual source of the observed file size change, linking it to individual CMSSW modules.

A. Regression Features

The most important feature of result publication is the possibility to show regression plots and tables for the basic software performance metrics. The PerfSuiteGUI application offers an advanced interface to navigate all the available tests in the performance database. Once the user has chosen the releases, steps, samples and conditions of interest, she or he can choose between a one to one comparison between two specific releases, or a summary table with an arbitrarily large number of releases, jobs and tests. Fig. 4 shows, for example, the CPU time comparison between CMSSW version 4.2.8.p7 and CMSSW version 4.4.1 for the reconstruction of TTbar events with low luminosity event pile-up. This comparison of two releases produces also an interactive CPU time table that highlights changes of the order of 10% to 20% in yellow and the ones of the order of 20% or more in red. Besides this direct regression, only available between given tests, it is possible to compare in a table an arbitrarily large number of tests over several CMSSW releases, samples, steps and conditions. This helps users that are looking for a long-term trend to get a quick overview, before delving into the details for one or two specific sets of sample, step and CMSSW version.

Fig. 4. CPU time per event regression plots for the reconstruction of TTbar events with low luminosity event pile-up in CMSSW version 4.2.8.p7 (job1, in blue) and in CMSSW version 4.4.1 (job2 in red), in HS06 [14] seconds. In this plot one can appreciate the CPU time performance improvement between the CMSSW 4.2 release cycle and the CMSSW 4.4 one.

For the same TTbar events and CMSSW releases, Fig. 5. shows the RSS memory regression plots. These plots are of key importance since a serious increase in the RSS peak footprint could imply the systematic failure of CMSSW jobs on the grid, when running on grid sites that only comply with the minimum amount of memory per node specified in the CMS computing requirements.

Fig. 5. RSS memory regression plots for TTbar reconstruction with low luminosity event pile-up in CMSSW version 4.2.8.p7 (job1, blue line) and CMSSW version 4.4.1 (job2, red line). The memory footprint reduction is clearly visible.

For file size on disk the regression simply shows the average file size per event for the jobs being compared, compressed and uncompressed and runs a quick validation against the total output files.

IV. CMSSW PERFORMANCE RESULTS

In order to give an overview, from the first LHC collisions years, the basic metrics results for the event reconstruction processing step are shown in Tables III-IV. Each table shows the evolution of one of the metrics from the first LHC collision runs of 2009 to the high luminosity collision runs of 2011. During this period of time CMSSW evolved from version 3.3.6 to version 4.4.1. For all three tables the same 3 representative samples are used: MinBias, TTbar and TTbar PU, which includes low luminosity event pile-up.

Table III shows the CPU time evolution measured in HS06 seconds, the values are per event and they are averaged over 8000 events per test. Constant improvement over the years is observed, notwithstanding the increased physics performance of the algorithms.

TABLE III. CMSSW CPU TIME PER EVENT PERFORMANCE EVOLUTION

Data-taking Period	CMSSW Release	MinBias RECO ^(*)	TTbar RECO ^(*)	TTbar PU RECO ^(*)
Fall 2009	3.3.6	5.2	40	94
Fall 2010	3.8.6	5.3	45	98
Fall 2011	4.2.8.p7	3.4	36	73
Fall 2011	4.4.1	6.1	28	51

^(*) CPU time expressed in HS06 seconds per event, averaged over 8000 events.

In Table IV, the VSIZE memory peak over 8000 events jobs is reported. Here one can observe the expected deterioration in terms of VSIZE memory footprint between CMSSW release 3.8.6 and 4.2.8.p7, due to the switching from the 32-bit to the 64-bit architecture. Version 4.4.1 already shows encouraging signs of decrease for VSIZE peak.

TABLE IV. CMSSW VSIZE PEAK MEMORY FOOTPRINT EVOLUTION

Data-taking Period	CMSSW Release	MinBias RECO ^(*)	TTbar RECO ^(*)	TTbar PU RECO ^(*)
Fall 2009	3.3.6	897	967	1105
Fall 2010	3.8.6	898	955	1057
Fall 2011	4.2.8.p7	1190	1258	1441
Fall 2011	4.4.1	1167	1228	1299

^(*) VSIZE expressed in MB, reporting the peak allocation over 8000 events.

Table V shows the evolution of the file size on disk per event. A jump is noticeable again between CMSSW version 4.2.8.p7 and 3.8.6: this is due to optimization in the event contents in the output file. A few container classes have been optimized, in order to handle the larger output files due to the expected higher event pile-up in the latest LHC collisions.

TABLE V. CMSSW FILE SIZE ON DISK PER EVENT EVOLUTION

Data-taking Period	CMSSW Release	MinBias RECO ^(*)	TTbar RECO ^(*)	TTbar PU RECO ^(*)
Fall 2009	3.3.6	107	477	833
Fall 2010	3.8.6	115	484	822
Fall 2011	4.2.8.p7	83	383	612
Fall 2011	4.4.1	78	380	630

^(*) File size on disk expressed in KB per event, averaged over 8000 events.

V. CONCLUSIONS

The CMSSW performance suite is an integral part of the CMS framework and of its release cycle. A large amount of performance data has been accumulated in the first LHC collision years, via automatic performance profiling in the CMSSW Release Validation and Integration Building process. Individual developers and software managers can access, both automatically and individually produced, in-depth performance results and regression information, boosting the performance improvement in the development process. Both the PerfSuite and PerfSuiteGUI have proven precious and flexible in monitoring the fast code evolution of CMSSW.

The CMS software framework has handled excellently the first LHC collision data and the harsh Monte Carlo production requirements. CMSSW developers have improved the physics performance of the code, while maintaining and improving the overall software quality in terms of CPU time, memory usage and file size on disk.

REFERENCES

- [1] Oliver Brüning *et al*, "LHC design report. Vol. I: The LHC main ring," CERN-2004-003.
- [2] CMS Collaboration, "The CMS experiment at the CERN LHC," *JINST* 0803 S08004.
- [3] CMS Collaboration, "The CMS Computing Model," CERN-LHCC-2004-035/G-083.
- [4] CMS Collaboration, "The Computing Project Technical Design Report," CERN/LHCC 2005-23.
- [5] CMS Collaboration, "CMS Physics Technical Design Report Vol I: Detector Performance and Software," CERN/LHCC 2006-001.
- [6] G. Eulisse, L. Tuura, "IgProf profiling tool", Proc. CHEP04, Computing in High Energy Physics, Interlaken, 2004.
- [7] L. Tuura, V. Innocente, G. Eulisse, "Analysing CMS software performance using IgProf, OProfile and callgrind," *J. Phys.: Conf. Ser.* 119 042030 (8pp), 2008, doi: 10.1088/1742-6596/119/4/042030.
- [8] Nicholas Nethercote and Julian Seward, "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation," Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007), San Diego, CA, USA, 2007.
- [9] Josef Weidendorfer, Markus Kowarschik and Carsten Trinitis. "A Tool Suite for Simulation Based Analysis of Memory Access Behavior," Proceedings of the 4th International Conference on Computational Science (ICCS 2004), Krakow, Poland, June 2004.
- [10] Julian Seward and Nicholas Nethercote., "Using Valgrind to detect undefined value errors with bit-precision," Proceedings of the USENIX'05 Annual Technical Conference, Anaheim, CA, USA, 2005.
- [11] Nicholas Nethercote, Robert Walsh and Jeremy Fitzhardinge, "Building Workload Characterization Tools with Valgrind," Invited tutorial, IEEE International Symposium on Workload Characterization (IISWC 2006), San Jose, CA, USA, 2006.
- [12] Gabriele Benelli, "The CMS software performance at the start of data taking," *IEEE Nucl. Sci. Symp. Conf. Rec.* (2008) 2667-2671
- [13] Oliver Gutsche, "Validation of software releases for CMS," *J. Phys.: Conf. Ser.* 219 042040, doi:10.1088/1742-6596/219/4/042040, 2010.
- [14] <http://cmssdt.cern.ch/qa/perfmondb/>
- [15] Michele Michelotto *et al*, "A comparison of HEP code with SPEC benchmarks on multi-core worker nodes," *J. Phys.: Conf. Ser.* 219 052009 doi:10.1088/1742-6596/219/5/052009, 2010.
- [16] Gabriele Benelli *et al*, "The CMSSW benchmarking suite: Using HEP code to measure CPU performance," *J. Phys.: Conf. Ser.* 219 052016 doi:10.1088/1742-6596/219/5/052016, 2010.