Available on CMS information server

CMS CR -2013/380

The Compact Muon Solenoid Experiment

Conference Report

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

29 October 2013 (v2, 30 October 2013)

Prototype of a File-Based High-Level Trigger in CMS

Gerry Bauer[6], Ulf Behrens[1], James Branson[4], Olivier Chaze[2], Sergio Cittolin[4], Jose Antonio Coarasa[2], Georgiana-Lavinia Darlea[6], Christian Deldicque[2], Marc Dobson[2], Aymeric Dupont[2], Samim Erhan[3], Dominique Gigi[2], Frank Glege[2], Guillelmo Gomez-Ceballos[6], Robert Gomez-Reino[2], Christian Hartl[2], Jeroen Hegeman[2], Andre Holzner[4], Lorenzo Masetti[2], Frans Meijers[2], Emilio Meschi[2], Remigius K. Mommsen[5], Srecko Morovic[2,a], Carlos Nunez-Barranco-Fernandez[2], Vivian O'Dell[5], Luciano Orsini[2], Wojciech Ozga[2], Christoph Paus[6], Andrea Petrucci[2], Marco Pieri[4], Attila Racz[2], Olivier Raginel[6], Hannes Sakulin[2], Matteo Sani[4], Christoph Schwick[2], Andrei Cristian Spataru[2], Benjanin Stieger[2], Konstanty Sumorok[6], Jan Veverka[6], Christopher Colin Wakefiled[2], Petr Zejdl[2]

Abstract

The DAQ system of the CMS experiment at the LHC is upgraded during the accelerator shutdown in 2013/14. To reduce the interdependency of the DAQ system and the high-level trigger (HLT), we investigate the feasibility of using a file-system-based HLT. Events of $\sim$1 MB size are built at the level-1 trigger rate of 100 kHz. The events are assembled by $\sim$50 builder units (BUs). Each BU writes the raw events at $\sim$2 GB/s to a local file system shared with $\mathcal{O}(10)$ filter-unit machines (FUs) running the HLT code. The FUs read the raw data from the file system, select $\mathcal{O}(1\%)$ of the events, and write the selected events together with monitoring meta-data back to a disk. This data is then aggregated over several steps and made available for offline reconstruction and online monitoring. We present the challenges, technical choices, and performance figures from the prototyping phase. In addition, the steps to the final system implementation will be discussed.

Presented at *CHEP2013 Computing in High Energy Physics 2013*

[1] DESY, Hamburg, Germany

[2] CERN, Geneva, Switzerland

[3] University of California, Los Angeles, Los Angeles, California, USA

[4] University of California, San Diego, San Diego, California, USA

[5] FNAL, Chicago, Illinois, USA

[6] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

[a] Now at Institute Rudjer Boskovic, Zagreb, Croatia

Prototype of a File-Based High-Level Trigger in CMS

G Bauer[6], U Behrens[1], J Branson[4], O Chaze[2], S Cittolin[4], J A Coarasa[2], G-L Darlea[6], C Deldicque[2], M Dobson[2], A Dupont[2], S Erhan[3], D Gigi[2], F Glege[2], G Gomez-Ceballos[6], R Gomez-Reino[2], C Hartl[2], J Hegeman[2], A Holzner[4], L Masetti[2], F Meijers[2], E Meschi[2], R K Mommsen[5], S Morovic[2a], C Nunez-Barranco-Fernandez[2], V O'Dell[5], L Orsini[2], W Ozga[2], C Paus[6], A Petrucci[2], M Pieri[4], A Racz[2], O Raginel[6], H Sakulin[2], M Sani[4], C Schwick[2], A C Spataru[2], B Stieger[2], K Sumorok[6], J Veverka[6], C C Wakefiled[2] and P Zejdl[2]

[1] DESY, Hamburg, Germany
[2] CERN, Geneva, Switzerland
[3] University of California, Los Angeles, California, USA
[4] University of California, San Diego, California, USA
[5] FNAL, Chicago, Illinois, USA
[6] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

[a] Also at Institute Rudjer Boskovic, Zagreb, Croatia

E-mail: remigius.mommsen@cern.ch

Abstract. The DAQ system of the CMS experiment at the LHC is upgraded during the accelerator shutdown in 2013/14. To reduce the interdependency of the DAQ system and the high-level trigger (HLT), we investigate the feasibility of using a file-system-based HLT. Events of ∼1 MB size are built at the level-1 trigger rate of 100 kHz. The events are assembled by ∼50 builder units (BUs). Each BU writes the raw events at ∼2 GB/s to a local file system shared with $\mathcal{O}(10)$ filter-unit machines (FUs) running the HLT code. The FUs read the raw data from the file system, select $\mathcal{O}(1\%)$ of the events, and write the selected events together with monitoring meta-data back to a disk. This data is then aggregated over several steps and made available for offline reconstruction and online monitoring. We present the challenges, technical choices, and performance figures from the prototyping phase. In addition, the steps to the final system implementation will be discussed.

1. Introduction

The Compact Muon Solenoid (CMS) experiment at CERN, Switzerland, is one of the two general purpose experiments located at the LHC. CMS is designed to study both proton-proton and heavy ion collisions at the TeV scale [1]. The detector comprises about 55 million readout channels. The online event-selection is performed using two trigger levels: a hardware-based first-level trigger accepting up to 100 kHz of events and a software-based high-level trigger (HLT) selecting $\mathcal{O}(1\%)$ of these events. The HLT operates on fully assembled events using the CMS software framework (CMSSW) [2] running on a farm of computers.

The CMS data-acquisition (DAQ) system [3] is upgraded during the first long-shutdown of the LHC (2013/14). The main motivation for the upgrade is the ageing of the existing hardware

Table 1. The HLT farm consists of three generations of machines running the event selection code. The Dell C6100 and C6220 machines were commissioned during run 1 and will be used until their five-year life cycle is over. A new generation, dubbed 'PC-2015', will be bought at the beginning of run 2. The table gives the foreseen static allocation of machines to builder units (BUs) in the case of the same processing power as in DAQ 1. The allocation will be slightly different should more processing power be required.

Machine Type	Operation	CPU Config	Cores/ Node	Nodes	Total Cores	BUs	Nodes/ BU	Cores/ BU
C6100	2011-15	2×X5650	12	308	3,696	14	22	264
C6220	2012-16	2×E5-2670	16	256	4,096	16	16	256
PC-2015	2015-19	2×IvyBridge	24	240	5,760	24	10	240
Total				804	13,552	54		

(both PCs and network equipement are at least 5 years old), and the need to accommodate sub-detectors with upgraded off-detector electronics that exceed the original specification of the DAQ system.

2. DAQ 2

The new DAQ 2 system [4, 5] uses state-of-art network technology to collect event fragments from the sub-systems and to assemble them into complete events. The event assembly is done on $\mathcal{O}(50)$ builder units (BUs) connected to the event-builder network based on InfiniBand technology [6]. The BUs make the data available to the HLT farm (see table 1). Each BU serves a fixed number of filter units (FUs) which are running the selection code. The allocation of nodes to BUs is static and aims to balance the processing power attached to each BU, while keeping the different machine generations separated. This allows for an easier replacement of the FU nodes once they reach the end of their five-year life cycle.

3. A file-based HLT farm

In the original DAQ system used during run 1, the event selection code was embedded into the data-acquisition application [7]. This tight coupling required to use the same compiler version and externals, to align the release schedules, and to work around the different state models and services of the XDAQ [8] and CMSSW frameworks. Therefore, a separation of the data-flow and the event selection applications would be beneficial. The CMSSW framework running the event selection code is mainly used for offline reconstruction and analysis. It is geared to use files as input and output. Therefore, the idea arose to use files for the HLT, too. A file system provides a natural interface between the DAQ and HLT world. It provides some time decoupling by buffering the data, which allows to finish processing data from a previous run while a new run is already starting. In addition, a network filesystem provides an abstraction from network protocols. It also provides bookkeeping, resource accounting and arbitration which we do not need to invent. Last but not least, having a clear separation between DAQ and HLT domains clarifies the responsibilities and simplifies the debugging.

3.1. The BU Appliance

In the previous DAQ system, each filter unit was connected to the event-builder network and ran the event building locally. The high-performant InfiniBand network used for DAQ 2 makes

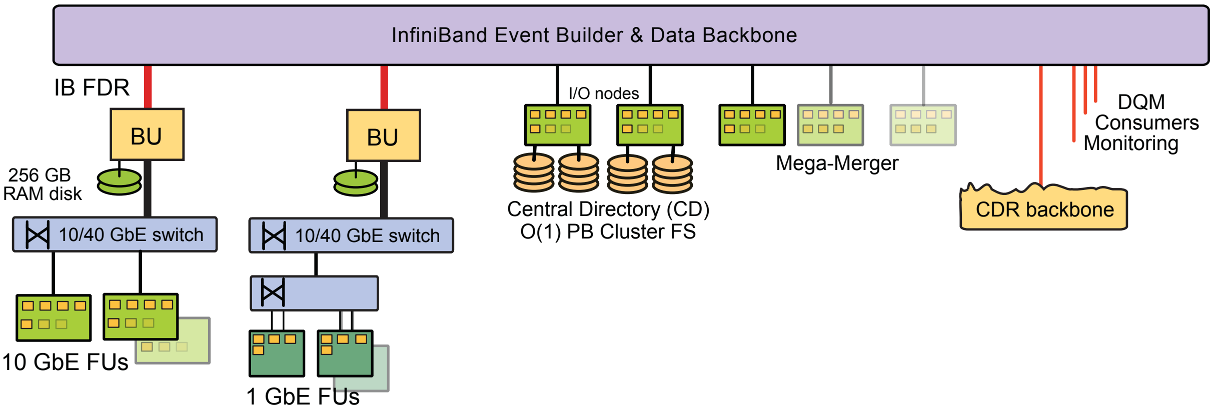

Figure 1. Sketch of the file-based HLT system. One the left, 2 BU appliances are shown. A builder-unit (BU) machine is connected to the event-builder network based on InfiniBand (IB). The filter-unit (FU) machines are attached to the BU via Ethernet. The latest generation machines have a 10 GbE connection using a 40 GbE uplink to the BU. Older generation machines will be connected with bonded 1 GbE links using legacy switches connected over 10 GbE to the 40 GbE uplink. On the right, the components of the storage and transfer system are shown.

this approach inefficient. Therefore, ∼54 builder machines (BUs) will be connected to the event-builder network and write complete events to disk. Each BU has a statically allocated number of filter units (FUs). The FU machines mount the disk from the BU over a networking filesystem. They read the files from this disk and process them. The accepted events are written to the local disk. We call the unit consisting of one BU and its associated FUs an appliance (figure 1). Each appliance is a stand-alone unit with clear boundaries to the surrounding systems. It has a single function of building and filtering events. In addition, different hardware generations can work independently. In the future, it could be that some appliances provide additional features like GPUs or Xeon-Phi co-processors.

The required throughput per BU is ∼2 GB/s at a level-1 trigger rate of 100 kHz. A standard harddisk cannot cope with this. Writing in parallel to SSD media was considered. However, as the lifetime of the media for our use case is questionable, we decided to use a RAM disk. A RAM disk of 256 GB would allow to buffer data for about 2 minutes or 4-5 luminosity sections of ∼23 s. The luminosity section is the smallest unit used for assessing the data quality in CMS. Events from different luminosity sections are written into separate files. In addition, multiple files are written per luminosity section. This is done for two reasons: firstly, the BU uses multiple threads to build and to write events independently, which results in one file per thread. Secondly, each HLT process reads one file at the time. Thus, having multiple files per luminosity section allows for a more efficient use of the FUs as each process quickly gets a file to work on.

Each data file is accompanied by a meta-data file containing the number of events in the file. Additional meta-data files are written for each luminosity section and at the end of the run. We are using the JSON (JavaScript Object Notation) format which is lightweight, easily readable by humans and can be displayed on web pages for monitoring purposes.

The control of the event-selection processes is also using the file system. At each run start, the BU creates a run directory containing the run number and the configuration for the HLT processes. A daemon discovers the creation of the run directory and starts the HLT processes. The daemon assures that the required resources are available, i.e. it does not start new processes if the FU is still processing data from a previous run, or if resources on the FU are allocated for other tasks, e.g. for running the CMS cloud. Once the HLT processes are started, they keep looking for files to appear on the BU disk. When a HLT process grabs a file, the corresponding meta-data file is renamed. To avoid any race condition while renaming the file, a unique lock file is used. The HLT process stops processing events when all files have been grabbed and the end-of-run meta-data file from the BU has appeared.

Each HLT process writes the accepted events to the local disk. Accepted events are sorted into different streams. One stream contains physics events, while up to 10 streams are used for different calibration events, data-quality monitoring, and other tasks. A separate file is written for each stream and luminosity section. For each output file, a meta-data file is created containing the number of processed and accepted events. In addition, a set of histograms used for online data-quality monitoring is written for each luminosity section .

The bandwidth requirement per FU for the output is ∼3 MB/s, where we assume a 1 kHz HLT accept rate, an event size of 1 MB and a conservative factor 2 for calibration and monitoring data. However, the number of files created is daunting. In the worst case scenario where each core runs a separate process, up to 24 files are created per second and machine. This rate would be reduced by an order of magnitude if the currently being developed multi-threaded CMSSW framework [9] could be used for the HLT.

3.2. The Storage and Transfer System

The files written to the local disks on the FUs by the HLT processes need to be aggregated. Thereby, the data belonging to one luminosity section and stream is treated separately and independently. The aggregation is done in 3 steps (see figure 1):

(i) The data from all processes on one filter-unit is merged. The resulting files from each filter-unit are written to a disk local to the appliance, either in the BU machine or on a NAS.

(ii) The data from all FUs attached to a given BU is merged. The resulting files are written to a global file system visible from all BUs.

(iii) The final merge (mega-merger) takes the files from all BUs and merges them into a single file on a global file system. These files are then either copied to tier 0 for offline calibration and reconstruction, or made available to online calibration and monitoring applications.

Each step starts as soon as new files appear. Once all input files are processed, they can be deleted. The meta-data files written by the BU and HLT processes are used to decide if all data has been processed. For example, all data for a given luminosity section has been processed when the event count of processed files as reported by the HLT processes matches the number of events written by the BU in this luminosity section.

The requirement for the global file system used for the final merge is high. Each second, about 25 files are concurrently written from $\mathcal{O}(50)$ sources. The total bandwidth is ∼1 GB/s. These files need to be read by the final merge and the result written back to disk. Finally, the files have to be copied over to tier 0, or they are accessed by multiple online consumers. Therefore, a total read/write bandwidth of ∼5 GB/s from $\mathcal{O}(100)$ clients needs to be sustained. The total disk space of $\mathcal{O}(250\,\mathrm{TB})$ will be similar to run 1. It allows to cache data for several days in case that tier 0 cannot digest the data.

4. Prototypes

In order to evaluate the feasibility of the proposed file-based system and to test software and hardware solutions, 2 small-scale prototyping systems have been assembled (see figure 2). The BU appliance consists of a Dell R720 machine with 288 GB RAM acting as BU and 8 Dell C6220 machines mounting the RAM disk on the BU over 10 GbE using NFS 4. The appliance is connected over InfiniBand to 4 Dell C6220 acting as readout-units (RUs). Each RU emulates a fixed-size event fragment of 256 kB which is assembled by the BU and written to disk. The

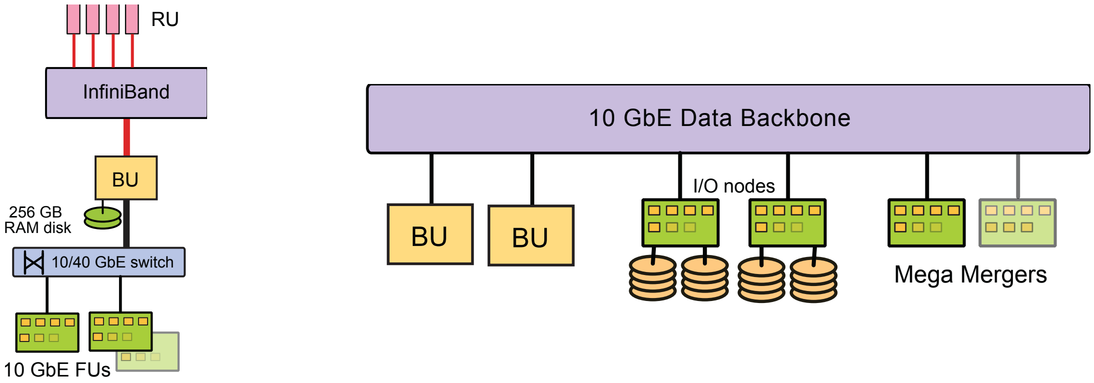

Figure 2. Sketch of the 2 prototyping systems: on the left the prototype of the BU appliance connected to 4 readout units (RUs), on the right of the storage and transfer system.

other prototype emulates the storage and transfer system: it consists of a number of machines emulating the output of a BU appliance, a cluster file system, and a few machines acting as merger nodes.

4.1. Prototype of the BU Appliance

A first test performed on the BU-appliance prototype measured the writing performance. Figure 3 shows the measured throughput from 4 RUs to the BU. The event building and disk writing is done in a configurable number of threads. Each thread writes a fixed number of events into separates files on the 256 GB RAM disk mounted on the BU. The throughput raises linearly with the number of threads until it plateaus at about 5.2 GB/s when using 6 or more threads, which is well above the requirement of 2 GB/s. There is a small difference in throughput if the RAM disk is empty or 90% full. This is due to the fact the event building is done on one CPU and the memory for the RAM disk is preferentially allocated from the local memory. Only if the local memory is exhausted, the memory banks attached to the other CPU are used, which lowers the performance by ∼7%. The figure also shows that the event building can be done at 6.2 GB/s independently of the number of threads. However, it is expected that the event building will become more demanding when more RUs send smaller event fragments. The lower figure shows that the throughput is not affected by the number of files generated. For each data file, a meta data file is generated. It is interesting to note that even writing each event into its separate file has only a small affect on the throughput.

In the test for the writing performance, the raw-data files are deleted either when the disk becomes full or immediately after closing. The next test uses HLT processes consuming the files. The HLT processes are started automatically by the HLT daemon who detects the appearance of the run directory. However, instead of doing real processing, the processes read the data and then sleep for a variable amount of time. They write ∼1% of the events accompanied by the corresponding meta data into 2 streams on the local disk. The meta data is used by a lightweight ruby-on-rails application server deployed in each machine. The server displays the status of the HLT processes on each FU node. The meta data is also aggregated by the server on the BU to summarise the data flow through the BU appliance. In future, this information will be used for aggregating the data files.

Figure 4 shows the RAM disk usage over a simulated run. When the run starts, the data starts accumulating on the disk. After about 30 s, the first files have been processed by the HLT and are deleted. From then on, the disk usage stays constant while the BU continues to write at ∼3 GB/s. The small increase in disk usage seen in the figure is due to the meta-data and

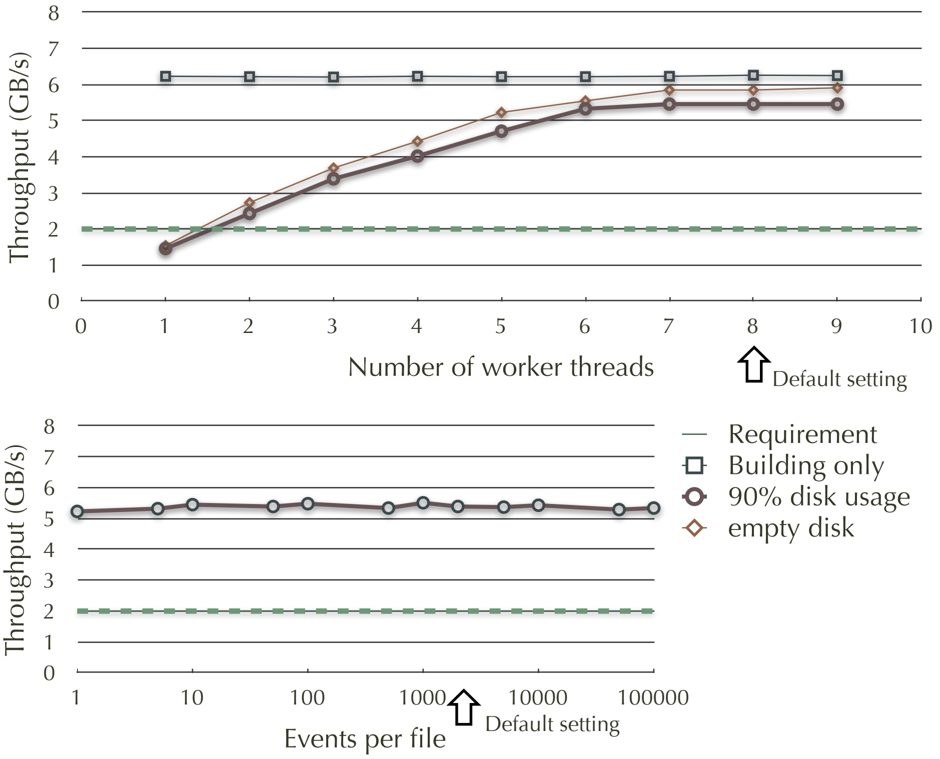

Figure 3. Measurement of the throughput to the RAM disk on the BU in the prototype system. The figure on the top shows the achievable throughput as number of worker threads. Each thread builds events from 4 sources and writes 2000 events into separate files. The lower figure shows that the throughput to disk using 8 worker threads is not affected by the number of events written into a single file.

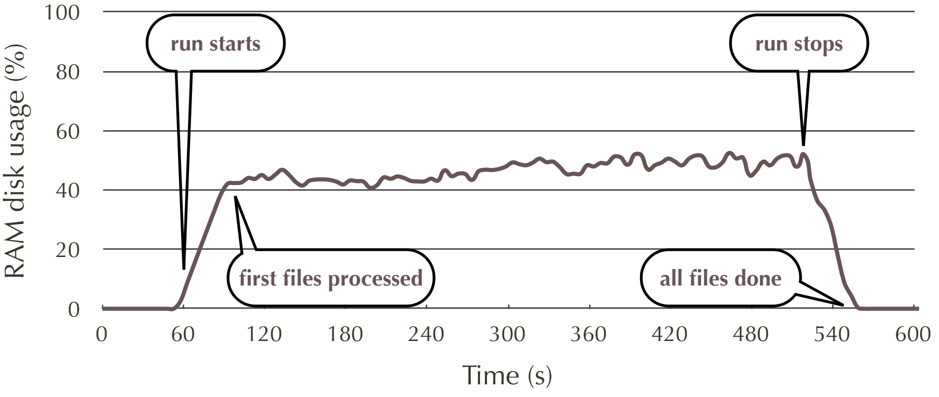

Figure 4. The usage of the RAM disk over a simulated run (see text). The small increase in disk usage is due to meta-data files not consumed in this prototype.

monitoring files which are not yet processed by the downstream system in this prototype. Once the run stops, it takes another ∼30 s to finish the processing. It is also possible to start a new run immediately after the stop of the previous run in which case the processing of the new data starts while the remaining data is processed.

4.2. Benchmark for the Storage and Transfer System

A benchmark has been developed which emulates the traffic of the final merge step. Several clients emulate the output of the BU appliance with fake data files. A simple merger aggregates these files. This benchmark is used to evaluate hardware solutions for the global file system. The benchmarks was tested on the run-1 legacy hardware [10] consisting of 8 Nexsan SATABeasts, each featuring 4 volumes. A Lustre filesystem (version 2.1 on SLC6) was set up on one of the volumes. The measured bandwidth merging (15 LS)·(10 streams)·(50 files) into (15 LS)·(10 streams)·(1 file) is 93.4 MB/s, corresponding to ∼3 GB/s when scaled to the full system. This is in agreement with previous tests and operational experience.

The first real test system was provided by EMC Isilon. The x-series system consists of 4 NAS heads running the OneFS operating system and providing 42 TB of disk space. 3 clients are connected over 10 GbE using NFS 4. The measured throughput is ∼0.7 GB/s. Assuming a perfect scaling to the final system size, the throughput would be ∼4.2 GB/s. In addition to the marginal performance, the files written on one client appear in burst and with a noticeable delay on the other clients, which results in uneven CPU workload and disk access. Therefore, the OneFS which is designed for data protection rather than speed, is not the optimal solution for our needs. Systems from other vendors based on Lustre or GPFS (IBM) are being evaluated.

5. Summary

A file-based high-level trigger system is an ambitious project. It is evaluated in the context of the new CMS data-acquisition system being constructed during the ongoing long shutdown of the LHC. The advance in network technology and the availability of affordable high-performance storage enables the decoupling of the data-flow software from the event-selection framework. The use of a file system provides a natural buffer space and resource arbitration, and it allows to run the HLT processes similar to offline jobs.

We presented the concept of largely independent BU appliances doing the event building and filtering, and a global file system which is used to aggregate the data from the $\mathcal{O}(50)$ BU appliances. Prototypes of the most critical components have been developed to demonstrate the feasibility and to evaluate possible hardware solutions. No show-stoppers have been identified. However, many issues concerning the scalability, reliability, and error recovery have still to be addressed.

Acknowledgments

This work was supported in part by the DOE and NSF (USA) and the Marie Curie Program.

References

[1] Adolphi R *et al.* (CMS Collaboration) 2008 *JINST* **3** S08004
[2] Lange D J (CMS Collaboration) 2011 *J.Phys.Conf.Ser.* **331** 032020
[3] Mommsen R K *et al.* 2011 *J.Phys.Conf.Ser.* **331** 022021
[4] Petrucci A *et al.* 2012 *J.Phys.Conf.Ser.* **396** 012039
[5] Holzner A *et al.* 2013 *in these proceedings*
[6] InfiniBand Trade Association, see http://www.infinibandta.com
[7] Spataru A *et al.* 2012 *J.Phys.Conf.Ser.* **396** 012008
[8] Gutleber J *et al.* 2010 *J.Phys.Conf.Ser.* **219** 022011
[9] Jones C and Sexton-Kennedy E 2013 *in these proceedings*
[10] Mommsen R K *et al.* 2010 *J.Phys.Conf.Ser.* **219** 022038