The Data-Logging System of the Trigger and Data Acquisition for the ATLAS Experiment at CERN

Andreas Battaglia, Hans Peter Beck, Marc Dobson, Szymon Gadomski, Kostas Kordas, Wainer Vandelli

ATL-DAQ-CONF-2008-003
25 February 2008

***Abstract*—The ATLAS experiment is getting ready to observe collisions between protons at a centre of mass energy of 14 TeV. These will be the highest energy collisions in a controlled environment to-date, to be provided by the Large Hadron Collider at CERN by mid 2008. The ATLAS Trigger and Data Acquisition (TDAQ) system selects events online in a three level trigger system in order to keep those events promising to unveil new physics at a budgeted rate of $\sim$200 Hz for an event size of $\sim$1.5 MB. This corresponds to a reduction of $\mathcal{O}(10^5)$ from the initial bunch-crossing rate of 40 MHz at nominal operating conditions. This paper focuses on the data-logging system on the TDAQ side, the so-called "Sub-Farm Output" (SFO) system. It takes data from the Event Filter farm, which is the third level trigger, and it streams and indexes the events into different files, according to each event's trigger path. The data files are moved to CASTOR, the central mass storage facility at CERN. The final TDAQ data-logging system has been installed using 6 Linux PCs, holding 24 disks of 500 GB each, managed by three RAID controllers on each PC. The data-writing is managed in a controlled round-robin way among three independent filesystems associated to a distinct set of disks, managed by a distinct RAID controller. This design allows fast I/O, which together with a high speed network permits to minimize the number of SFO nodes. We report here on the functionality and performance requirements on the system, our experience with commissioning it and on the performance achieved.**

***Index Terms*—Data acquisition, Data handling, Disks**

I. Introduction

THE ATLAS Trigger and Data Acquisition (TDAQ) system is based on three levels of online event selection [1], [2], [3]. Each trigger level refines the decisions made at the previous level and, where necessary, applies additional selection criteria. Starting from an initial bunch-crossing rate of 40 MHz, corresponding to an interaction rate of $\sim 10^9$ Hz at a luminosity of 10^{34} cm^{-2}s^{-1}, the rate of selected events must be reduced to $\mathcal{O}(200)$ Hz for permanent storage. This requires an overall rejection factor on the trigger level of 10^5 against minimum bias events, while retaining the rare new physics processes, such as Higgs boson decays.

Manuscript received November 23, 2007.

A. Battaglia, H. P. Beck and K. Kordas are with the Laboratory for High Energy Physics (LHEP), University of Bern, Switzerland, e-mails: andreas.battaglia@cern.ch, hans.peter.beck@cern.ch, kostas.kordas@cern.ch.

M. Dobson and W. Vandelli are with the European Organization for Nuclear Research (CERN), e-mails: marc.dobson@cern.ch, wainer.vandelli@cern.ch.

S.Gadomski was with the Laboratory for High Energy Physics (LHEP), University of Bern, Switzerland, now with the Département de physique nucléaire et corpusculaire (DPNC), University of Geneve, Switzerland, and with the Institute of Nuclear Physics, Cracow, Poland, e-mail: szymon.gadomski@cern.ch.

The LVL1 trigger reduces the event rate to 75 kHz (upgrade-able to 100 kHz) based on an initial selection using reduced granularity information from a subset of detectors. High transverse momentum muons are identified using only the muon-trigger chambers, resistive-plate chambers (RPCs) in the barrel, and thin-gap chambers (TGCs) in the end-caps. The calorimeter selections are based on reduced granularity information from all the calorimeters (electromagnetic and hadronic; barrel, end-cap and forward).

The LVL2 trigger reduces the event rate further down to $\sim$3 kHz based on a selection using the full detector granularity information inside a small region in pseudorapidity-azimuth coordinates around the trigger objects identified by the LVL1 trigger. About 2% of the event data volume needs to be accessed by the LVL2 trigger processing farms.

After LVL2, the event is fully assembled by the Event Builder [3], [4], [5] and then sent to the last stage of the online selection, the Event Filter. The Event Filter employs offline algorithms and methods, adapted to the online environment. It uses the most up to date calibration and alignment information and an accurate magnetic field map, to make the final selection of physics events. The output rate from LVL2 is reduced by an order of magnitude, giving $\sim$200 Hz.

From the Event Filter events are sent to the data-logging system, the so-called "Sub-Farm Output" (SFO) system. A data-logger application implemented in C++ and running on the SFO nodes buffers the events in memory, decodes and associates them to streams and to Luminosity Blocks. The streams are classes of events defined by the event's trigger path. The Luminosity Blocks are time intervals during which the instantaneous luminosity can be assumed constant. The SFO system writes the events into raw data files according to stream and Luminosity Block information, and sends the data files to the CERN Advanced STORage system (CASTOR) for permanent storage. From there, they are retrieved by the CERN computing center, Tier-0, to perform a full first-pass reconstruction analysis, whose result is sent to other computing centres up to local institutes for final analyses.

II. The SFO system

The SFO system represents the final element in the TDAQ chain; it receives the selected event data from the Event Filter system, writes them into raw data files and copies the files to CASTOR for permanent storage.

The requirements of the system are dictated by the Event

Filter output rate of ∼200 Hz. For an ATLAS event size of 1.5 MB, the SFO system has to receive and write the events to local disks at a speed of 300 MB/s, and to feed CASTOR with at least at the same rate to avoid filling up the disks.
Apart from the steady-state operations, we want to be able to keep collecting data at the TDAQ site for about 48 hours, even when the output towards CASTOR is problematic. Thus, beyond the performance requirements on the writing and reading speeds, we need at least 52 TB of local disk space for a 48-hours buffering.
The data handling effort in the Tier-0 reconstruction center increases with the number of files produced in a given run, defined by:

$$\frac{\text{Data volume in the run (MB)}}{\text{Average file size (MB)}}$$

We want to allow streaming (see Section II-B4) and to respect Luminosity Block boundaries of $\mathcal{O}$(min) (see Section II-B3). The number of files written is proportional to the number of streams and inversely proportional to the Luminosity Block duration. Since each SFO node works independently from each other and serves events from all streams, the number of files is also proportional to the number of SFO nodes. A typical file size is given by:

$$\frac{\text{Data rate (MB/s)} \cdot \text{Luminosity Block size (s)}}{\text{Number of SFO nodes} \cdot \text{Number of streams}}$$

Therefore, to minimize the number of files produced, we need to minimize the number of SFO machines. Thus we need to maximize network and a disk I/O per SFO node.
We can define the overall SFO system to be composed of three main entities, each devoted to address some of the above requirements/tasks:

- the hardware, i.e. the SFO machines;
- the SFO application, running on each SFO machine;
- the CASTOR client script, running on each SFO machine.

A. SFO Hardware

ATLAS installed 6 SFO PCs; each one is a rack mountable PC of 5U height and 54 kg weight, equipped with (see Fig. 1):

- an Intel S5000PSL motherboard;
- two Intel E5130 dual-core 2.0 GHz CPUs;
- 4 × 1 GB RAM;
- three 3ware SATA RAID Controllers (one 9550SXU-12MI and two 9650SE-8LPML);
- 24 SATA disks (Hitachi HDS725050KLA360) of 500 GB each, equally shared among the 3 RAID controllers. Considering an effective capability of 450 GB per disk, 5 machines should therefore assure about 54 TB of disk space;
- five Gigabit Ethernet (GE) NICs (Network Interface Controller); 2 on the motherboard, 2 on an Intel EXPI9402PT PCI-e, dual-port Network Card and 1 on an a Intel EXPI9400PT PCI-e, single-port Network Card.

From the data-flow point of view, each SFO machine has four 1 Gbit/s Ethernet links (see Fig. 2): two links for the input from the Event Filter system, towards a switch (called BackEnd Switch) to which the Event Filter nodes are connected; two links for the output to CASTOR, towards a second switch to which CASTOR connects via one 10 Gbit/s link. The Gigabit Ethernet links will be bonded [6] and should assure sufficiently high bandwidth.
The fifth link is connected to the control network to handle the Run Control commands and to allow administering the nodes remotely, e.g. powering up and down via IPMI (Intelligent

Fig. 1. The six final SFO machines, mounted in two racks.

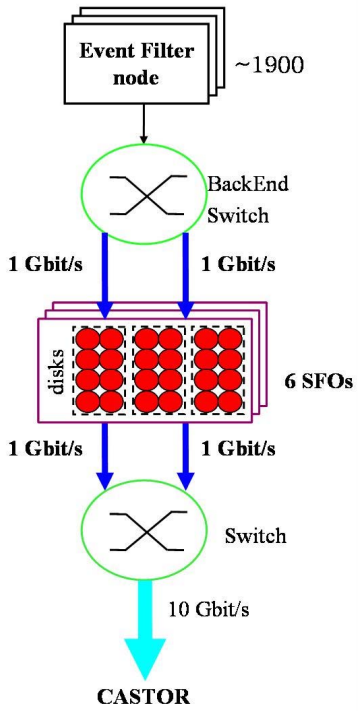

Fig. 2. Network topology with Event Filter nodes (EFDs), SFO nodes and CASTOR. In the final system there are 6 SFO machines, each connected via two 1 Gbit/s links to the BackEnd Switch and via two 1 Gbit/s links to the switch towards CASTOR.

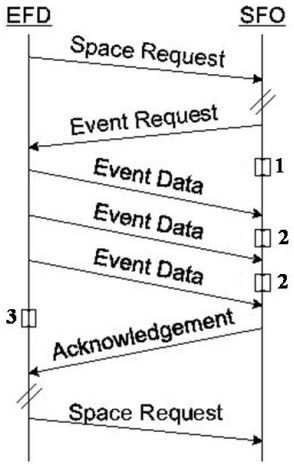

Fig. 3. Sequence of message exchange between EFD and SFO. Three boxes (labelled 1, 2, 3) indicate three timeout values that allow to determine the non-reception of the event data, the non-reception of still outstanding event data and the non-reception of the acknowledgment.

Platform Machine Interface [7]).
Five of the six SFO PCs are used for data-taking at any given moment; the sixth PC serves as a live-spare. Each SFO participating in the data-taking has to deal with $\sim$40 events per second.

B. SFO Application

The SFO Application software, written in C++, runs on an SFO machine and is responsible for:

- buffering in memory the event data received from the Event Filter system;
- organizing the event data in file structures;
- saving files on disk.

1) Event Buffering: The communication between the SFO and the Event Filter is based on the EFIO protocol using TCP/IP [8]. The SFO application acts as a server to the Event Filter Dataflow (EFD) processes. An EFD process sends a space request to the SFO (see Fig. 3); if sufficient buffer space is available, the SFO application returns an event request, to which the EFD process responds by sending the event; after the complete reception of the event, the SFO application sends an acknowledge message back to the EFD process, which then clears the event from its internal buffers.
Multiple EFD processes connect to an SFO application simultaneously.
The number and the size of the internal buffers used by the SFO application to store event data are configurable parameters; each buffer holds one event.

2) Data Saving: The SFO application receives events from the Event Filter in a "byte stream" format, i.e. a vector of 32 bit words containing the event information, organized according to the event format convention [9]. As soon as an event is stored in an internal buffer, the SFO application writes the bare "byte stream" data into raw data files [10]. After the data are saved to disk, the buffer is released and made available for the next event. In case the disks are full, the SFO application will fill up its buffers and consequently will ignore further space requests from the Event Filter; this way back-pressure is propagated to the Event Filter.
The file creation is fully data driven: after the event reception and buffering, the SFO uses the information contained in the event header to associate the event to one or more streams, and eventually to a given Luminosity Block.

3) Luminosity Blocks: Instantaneous luminosity depends on quantities which are time dependent. One can assume that this dependence is small enough so that these values can approximately be considered constant over a short time period, which we refer to as a Luminosity Block [11]. We define a Luminosity Block as a time interval for which the integrated, dead-time-corrected and pre-scale-corrected luminosity can be determined. A given Luminosity Block is identified by a unique index, called Luminosity Block number, which is reported in every event header as a 16 bit word.
Luminosity Blocks allow to keep the losses to a minimum in case of failures in the Data Acquisition system, data production, analysis, detector and machine operation; this can be done by excluding from the analysis those Luminosity Blocks in which failures occurred. It has to be possible to calculate the integrated luminosity of a given data-taking run even with missing Luminosity Blocks. The amount of tolerable losses due to such failures sets an upper limit on the duration of a Luminosity Block. The recommended Luminosity Block duration is $\mathcal{O}$(min) [11].

4) Streaming: The SFO application writes the data into streams, which are characterized by:

- a type, that states if an event has been accepted as a Physics, Calibration or Debug event:
  Physics events will be used for reconstruction at Tier-0 and physics analysis in the local research centers and institutes;
  Calibration events are meant to calculate a new set of calibration constants, such as detector alignment;
  Debug events are those that need to be looked at with special care, as they were identified as problematic;
- a name, that characterize with more precision the streams of a given type; for instance, Physics streams [12] can be subdivided into: Electrons&Photons; Muons & B Physics; Jets; Taus & Missing E_T; Minimum Bias; Express streams.
  The Express stream [13] contains a subset of the physics data (roughly 10%) that will go with the highest priority through the offline reconstruction at Tier-0. This allows for quick feedback whether meaningful results are obtained before the main reconstruction starts;
- a boolean, which tells if the stream obeys Luminosity Block boundaries.

The main motivation for streaming is extra flexibility for prioritized reconstruction and re-reconstruction at Tier-0 and Tier-1 centres [12]. The streaming information is contained

in two C-strings and a boolean, reported in the event header. A single raw data file corresponds to a unique stream identified with stream type and stream name. For the streams which obey Luminosity Block boundaries, the Luminosity Block number also characterizes the raw data file. Luminosity Block assignment is necessary for streams meant for use in physics analyses, and optional for other streams.
As one event can belong to more than one stream, the Physics stream definitions have to be chosen in such a way that duplication of the same event in different streams are kept lower than a few percent, to reduce unnecessary extra storage.

5) Data files: The raw File names obey the following convention:

daq.<name tag>.<run number>.<stream type>.<stream name>.LB<luminosity block number>.<SFO ID>._<file number>.data

for closed files; the name of still open files ends with a *.data.writing* suffix. The stream type and name, the run number and the Luminosity Block number are decoded by the SFOs from the event header. All the files which belong to the same run, stream and Luminosity Block are labelled using a file index number. An example file name is given by:

daq.Cosmics.1532.Physics.Muon.LB0012.SFO-01._0123.data

For streams which obey Luminosity Block boundaries, the files are opened and closed at the Luminosity Block boundaries. Files are also closed and new ones opened when the maximum size or the maximum number of events per file (both configurable parameters) are reached. A size limit of 2 GB is currently used. It will be increased when 64-bit Linux will be deployed ATLAS wide.
Given a size limit of 2 GB and a required writing speed of 60 MB/s per SFO, in case of no streaming an SFO application writes, on average, one file every 34 s. In case of streaming, the file writing rate depends on the sharing of events among the various streams; if we suppose a balanced streaming with 10 streams, i.e. an average rate per stream of 6 MB/s, then an SFO application completes a file every 340 seconds for streams not obeying Luminosity Block boundaries. Otherwise the file writing rate is determined by the Luminosity Block size, currently to one file per $\mathcal{O}$(min).

6) Filesystem writing and reading: Simultaneous writing to and reading from the same disk can degrade the performance: as writing has usually priority, reading may be blocked. Moreover, simultaneous writing and reading forces an excess in mechanical movements of the disk access arms carrying the read and write heads, which may lead to a faster ageing. For these reasons, the SFO application organizes the writing in three different directories, each corresponding to an independent filesystem using a distinct set of disks, managed by a distinct RAID controller [14].
The writing to the various directories/filesystems is done in a round-robin mode and the reading goes on in parallel from those directories/filesystems which are not involved with writing. The SFO application locks the filesystem in which it writes by creating a lock file. A reading application, like the CASTOR client script, simply notices the presence of the lock file and refrains from accessing the filesystem. After a configurable time interval or when the available disk space reaches a configurable threshold, the SFO application changes the writing filesystem; any new files that need to be opened are created from now on in the newly-locked filesystem.
The directory change does not take an immediate effect to files still open in the old directory; these files continue being filled with event data. Only after a configurable transition timeout, all the still open files are closed, the old filesystem is unlocked and file-copying is allowed.
The filesystem rotation mechanism described above allows for fast I/O, and contributes in minimizing the number of SFO nodes needed, and in turn the number of files produced (see Section II-B5). With a proper choice of the transition timeout, e.g. half of the filesystem rotation period, the transition time logic allows to minimize the number of files closed due to the rotation mechanism itself.
The SFO application requires a configurable fraction of free disk space in order to consider a directory writable. If all filesystems disk usage is above threshold, the SFO application stops writing and no longer responds to space request messages from the Event Filter.

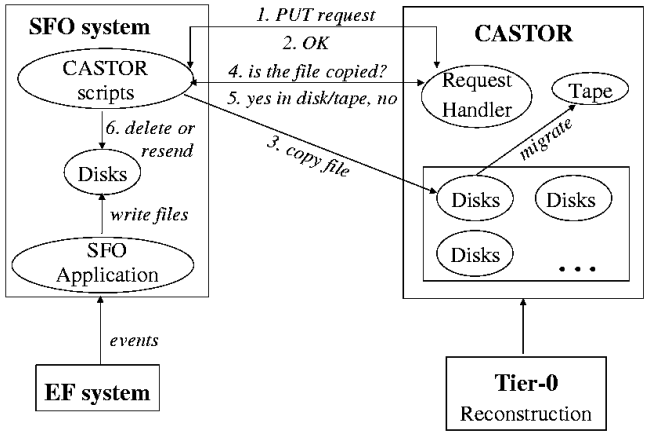

Fig. 4. Schema of the communication between the SFO system and CASTOR.

C. CASTOR client script

A CASTOR client script is responsible for copying the raw data files from the SFO disks to CASTOR and for deleting them from the SFO disks. This script runs on each SFO machine, fully decoupled from the SFO application.
For unlocked filesystems, a handshake mechanism which assures proper file transmission and deletion is schematized in Fig. 4. The file transfer is based on the RFIO protocol [15].

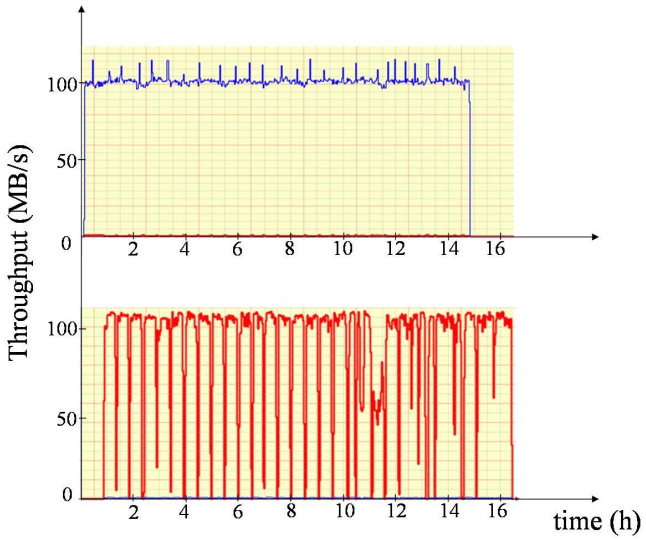

Fig. 5. On top, throughput out of the Event Filter-SFO 1 Gbit/s link. On bottom, throughput out of the SFO-CASTOR 1 Gbit/s link.

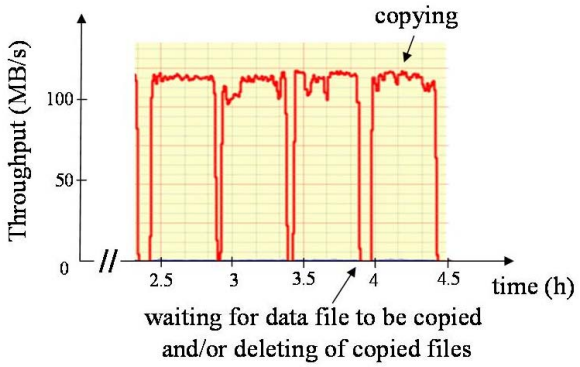

Fig. 6. Zoom-out of the bottom plot in Fig. 5; the deletion and copying periods are marked.

III. Measurements

The SFO system functionality and performance have been tested during several ATLAS TDAQ Technical Runs as a standard procedure of the ATLAS commissioning task. Test patterns, Monte Carlo and cosmic data have been exercised. In particular, the communication between Event Filter and SFO systems and between the SFO system and CASTOR has been exercised several times, while the file writing according to streaming and Luminosity Block boundaries and the filesystems rotation mechanism were tested in the ATLAS September 2007 Technical Run for the first time.

The measurements shown in Figs. 5 and 6 refer to a configuration in which one SFO node receives simulated events of 1 MB each from one Event Filter node and sends data files towards CASTOR; differently from the final system, the SFO node is currently connected by one 1 Gbit/s input link and one 1 Gbit/s output link. The file size limit was set to 1.5 GB, the filesystem rotation period was set to 30 minutes, the transition completion timeout was set to 15 minutes.

We were able to exploit $\sim$99 MB/s and up to $\sim$113 MB/s respectively via 1 Gbit/s link for input and 1 Gbit/s link for output, as shown in Fig. 5. The used bandwidth is plotted as a function of time. The not stable explotation of the output link towards CASTOR can be seen in Fig. 6 which is a zoom-in the lower plot in Fig. 5. This is due to a combination of the time to wait for the next file becoming available for copying, and the actual implementation of the CASTOR client script, which does not allow for concurrent deletion and copying.

IV. Conclusion

The SFO system described in this paper satisfies the ATLAS Data-Logging requirements: we have demonstrated that, even with only one 1 Gbit/s link in and one 1 Gbit/s out, each SFO node can receive more than 100 MB/s from the Event Filter and feed data to CASTOR with roughly the same speed. In the final system we foresee to deploy two 1 Gbit/s link in and two 1 Gbit/s link out for stability and to provide enough headroom in case of temporary congestions.

Performance improvements can still be achieved by tuning the SFO machine configuration parameters, especially the filesystem and RAID controller configurations. A new implementation of the CASTOR client script is ongoing and will allow simultaneous copying and deleting of data files.

Studies devoted to understand the scaling properties of the current SFO-Event Filter communication protocol with many Event Filter nodes connecting to an SFO machine, are ongoing.

Acknowledgment

The authors would like to thank ATLAS Online Software group for providing a system and useful tools to control, configure and operate a large-scale distributed TDAQ system as it is in need for the ATLAS experiment.

References

[1] *High-Level Trigger Data Acquisition and Controls*, ATLAS TDR 016, 02 October 2003. [Online]. Available: http://atlas-proj-hltdaqdcs-tdr.web.cern.ch/atlas-proj-hltdaqdcs-tdr

[2] K. Kordas, "The ATLAS Data Acquisition and Trigger: concept, design and status," in *Nuclear Physics B - Proceedings Supplements*, vol. 172, October 2007, pp. 172–182. [Online]. Available: http://www.sciencedirect.com

[3] H. P. Beck, "Performance of the final Event Builder for the ATLAS Experiment," in *Proc. 15th IEEE RTC*, Fermilab, Batavia, IL, USA, 2007.

[4] K. Kordas, "ATLAS High Level Trigger Infrastructure, ROI Collection and Event Building," in *Proc. 15th CHEP*, Mumbai, India, 2006.

[5] W. Vandelli, "The ATLAS Event Builder," in *this Proc. IEEE NSS*, Honolulu, Hawaii, USA, 2007.

[6] Linux Ethernet Bonding Driver HOWTO. [Online]. Available: http://garr.dl.sourceforge.net/sourceforge/bonding/bonding.txt

[7] Intel Corporation. [Online]. Available: http://www.intel.com/design/servers/ipmi

[8] H. P. Beck, S. Gadomski, C. Häberli, S. Klous, C. Meessen, A. Negri, and S. Wheeler, *EFIO: Protocol Specification*, DataCollection Note 035. [Online]. Available: https://edms.cern.ch/document/391570

[9] C. Bee, D. Francis, L. Mapelli, R. McLaren, G. Mornacchi, J. Petersen, and F. Wickens, *The raw event format in the ATLAS Trigger & DAQ*, EDMS: ATL-D-ES-0019. [Online]. Available: https://edms.cern.ch/document/445840

[10] H. P. Beck and S. Gadomski, *Format of the data files written by EventStorage library of ATLAS TDAQ*, DataCollection Note 066. [Online]. Available: https://edms.cern.ch/document/580290

[11] S. Ask, D. Malon, T. Pauly, and M. Shapiro, *Report from the Luminosity Task Force*, ATLAS, 20 July 2006. [Online]. Available: https://edms.cern.ch/document/755341
[12] *Report on Data Streaming in ATLAS*, Data Streaming Study Group, 09 November 2007. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/Atlas/DataStreamingReport
[13] S. Gadomski, *Express Stream and the FDR*, Trigger & Physics Week, June 2007. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/Atlas/ExpressStream
[14] H. P. Beck, C. Häberli, and S. Gadomski, *The Sub-Farm Output of ATLAS Trigger and DAQ*, DataCollection Note 070. [Online]. Available: https://edms.cern.ch/document/834750
[15] The CASTOR project page at CERN. [Online]. Available: http://castor.web.cern.ch/castor