Modeling and Validating Time, Buffering, and Utilization of a Large-Scale, Real-Time Data Acquisition System

Alejandro Santos^{*‡}, Pedro Javier García[†], Wainer Vandelli^{*}, and Holger Fröning[‡]

^{*}Physics Department, CERN, Geneva, Switzerland

[†]Computer Systems Department, University of Castilla-La Mancha, Albacete, Spain

[‡]Institute of Computer Engineering, Ruprecht-Karls University of Heidelberg, Germany

Abstract—Data acquisition systems for large-scale high-energy physics experiments have to handle hundreds of gigabytes per second of data, and are typically implemented as specialized data centers that connect a very large number of front-end electronics devices to an event detection and storage system. The design of such systems is often based on many assumptions, small-scale experiments and a substantial amount of over-provisioning.

In this paper, we introduce a discrete event-based simulation tool that models the dataflow of the current ATLAS data acquisition system, with the main goal to be accurate with regard to the main operational characteristics. We measure buffer occupancy counting the number of elements in buffers; resource utilization measuring output bandwidth and counting the number of active processing units, and their time evolution by comparing data over many consecutive and small periods of time. We perform studies on the error in simulation when comparing the results to a large amount of real-world operational data. We show which efforts are required to minimize the error for such a configuration, and explain possible reasons for the most important outliers we are observing. Furthermore, we use this tool to derive an operational envelope of the system, which describes the minimal amount of resources required to fulfill certain real-time guarantees.

Index Terms—CERN, ATLAS, Data Acquisition System, Simulation, Modeling, Omnet++.

I. INTRODUCTION

The ATLAS experiment [1] at CERN is a major experiment at the Large Hadron Collider (LHC). In the experiment, bunches of protons are collided at a center of mass energy of 14 TeV 40 million times per second for periods ranging up to 20 hours. The ATLAS data acquisition system receives 150 GB/s of physics data that are processed and analyzed in quasi-real time.

Future upgrades of the experiment [2], planned to happen in approximately one decade, will see an increase of the required data volume to be transported from 150 GB/s to 5 TB/s, a 33-fold increase driven by experimental conditions and the installation of new detector hardware components. There is a huge uncertainty on technology scenarios and availability, and it is not always possible to make precise predictions of the computing technology evolution. Different technology scenarios need to be considered. A methodologically sound approach based on predictive design techniques is essential to address these considerations.

Besides the need for technology exploration tools, the existing data acquisition system, called ATLAS Phase-0, has

proven to be effective processing many years of physics data. While new and increased data rates and volumes delivered by the detector will make the existing architecture unviable for the next-generation system, many general concepts and lessons should be taken into consideration for the upgrade. For that purpose, a fundamental understanding of the existing architecture, in particular its input data pattern and real-time behavior, is mandatory.

Predictive design techniques are based on different concepts, ranging from analytical models to simulation models to emulation to prototyping. Different techniques differ fundamentally in aspects including accuracy, flexibility, performance, ease and scalability [3]. For instance, analytical models are very fast in their prediction, very scalable, and quite flexible, but these advantages come at the cost of loss of accuracy when making predictions. The other extreme is prototyping, which is quite fast, usually not scalable, and with limited flexibility, but very accurate when predicting performance. System simulation is another well-known approach to predict performance, either based on functional simulation, which usually has implications on accuracy as time is poorly modeled, or based on single cycles of computing systems, with improved performance but reduced speed. In general, their accuracy is better than analytical models, but as they are based on simplifications there are obviously small absolute and relative errors compared to the final system. Simulators are particularly useful for network-centric distributed systems, as the investment in hardware and time needed to deploy such systems for testing is therefore not needed. Simulators do, however, require development time. This is particularly true when they have to be built from scratch. Fortunately, some simulation and network simulators as well as network-simulation frameworks are available [4], [5], [6].

There is little related work in the area of system modeling and simulation for data acquisition systems, believed to be the case due to the complex nature of the system and operational constraints. Ongoing work [7] also using the ATLAS data acquisition system as a case study aims at providing a scalable and complete simulation framework based on a combination of discrete and continuous models. At this stage however the discrete component dominates, limiting the performance and therefore the capability to effectively study a large spectrum of system-level operating conditions.

Figure 1: ATLAS Phase-0 data acquisition system architecture.

Previous work regarding data acquisition optimizations by reducing packet loss in lossy commodity networks while operating under hard real-time constraints can be found in [8]. Processing elements were not modeled in this previous work, but are mandatory for system-level simulation.

This paper presents the design, implementation, and validation of a simulation tool modeling the existing ATLAS data acquisition system. It makes it possible to explore different scenarios and to ultimately make the necessary changes to accommodate for the upcoming architectural design in a future upgrade, and to investigate architecture, provisioning, and advanced techniques. The simulation model follows the design of the existing system, and is validated by analyzing existing operational data stored in the experiment’s operational monitoring archives. Results include an exploration comparing metrics on a set of key elements of the existing system, and validation is done by cross-checking the output of the simulation with existing data.

The long term plan for the model is to focus on future system upgrades in order to explore architecture, provisioning, and advanced techniques like compression, storage, memory hierarchies and more. With this work being the first step in this direction, it could prove valuable to the simulation community as an early insight from the perspective of model design, analysis of problems faced and validation.

The main contributions of this work are:

- A summary of insights when analyzing input data for such large-scale experiments.
- Findings regarding simulation accuracy and limitations, comparing simulation results with real-world monitoring data.
- A conceptual exploration that re-uses the model to assess the provisioning of the current system referred to as *operational envelope*, making it possible to identify resource over-provisioning.

A procedure for modeling a complex data acquisition system using discrete event simulation will be presented, demonstrating the feasibility of this approach for the modeling and design of future systems.

II. ATLAS DATA ACQUISITION

Data acquisition systems for particle physics detectors need to handle a very large amount of data, as all the sensors

Figure 2: Implementation of the High-Level Trigger data flow and event management infrastructure for ATLAS Phase-0 data acquisition system.

combined can send aggregated data in the order of hundreds of gigabytes per second. The most common approach to handling such big workloads is to have an on-line filtration system, with one or more intermediate steps selecting in real-time the interesting data relevant to the physics goals of the experiment before sending the selected data to permanent storage. This initial filtering of data is known as the *trigger* system.

In ATLAS, most of the data produced by the detector follows well-known physics phenomena. The Trigger and Data Acquisition System [9] (TDAQ) receives the data from the detector, selects interesting collisions in real-time, and stores the selected data in permanent storage. The data captured at each collision is called an *event*. The event rate is consequently reduced, from the initial rate of 40 MHz to ~ 1 kHz.

The ATLAS TDAQ system combines both hardware and software trigger levels, shown in Figure 1. The first level (Level-1 trigger) is based on custom electronics: specialized hardware with precise real-time requirements. The Level-1 system reduces the initial collision rate to roughly 100 kHz of events, corresponding to ~ 150 GB/s, that are passed on to the next level.

A. High-Level Trigger

The next level, known as the High-Level Trigger (HLT), is based on commodity off-the-shelf multicore computer processors. The HLT architecture is shown in Figure 2. Each computer core runs a dedicated Processing Unit (PU) process, the set of PUs being collectively called *PU farm*. In 2016 there were ~ 40 thousand cores installed.

Each event is formed by ~ 1800 fragments, each produced by part of the front-end electronics of a detector system for that event. During event processing, the corresponding data fragments are stored in the Read-Out System (ROS) distributed across all servers. The ROS bridges the custom detector electronics and the commercial network domain, at the same time implementing the event data buffering needed to sustain the HLT decision latency. The ROS is composed of ~ 100 servers, each equipped with custom interface cards [10] for data reception and buffering as well as conventional 10GbE NICs for data output. Each custom card is equipped with 8 GB

of memory which, in the typical configuration, corresponds to a buffering capability of roughly 350000 fragments per input channel.

In order to process and select the assigned event, the PU farm fetches data fragments from the ROS, applying to them physics-based selection algorithms. The time to process and select an event corresponds to the HLT latency. On each PU server the access to the ROS is coordinated by a dedicated application, known as the Data Collection Manager (DCM). The DCM is responsible for managing the network communication for all PU processes on the same server, shielding the complex algorithmic code and providing a safe and reliable data-handling infrastructure. In 2016 there were ~2200 DCMs in the system, one for each physical PU server.

For events selected for permanent storage by the PU processes, the DCM collects any not yet requested fragments of the event from the ROS and conveys the now complete event to a temporary storage infrastructure, the Data Logger. Event data are aggregated by the logger into large files and asynchronously transferred to the CERN permanent storage system for off-line analysis.

B. Data-flow and Supervision

The HLT is a large, heterogeneous, loosely-coupled, distributed computer system, where events from different collisions are analyzed in parallel. In order to guarantee event bookkeeping, data safety and load balancing a supervising entity is needed.

The coordination of the data acquisition is performed by a dedicated process called the High Level Trigger Supervisor (HLTSV). The HLTSV keeps track of PU availability, assigning events accepted by the Level-1 trigger and receives PU decisions in response.

Upon acceptance by the Level-1 trigger, event fragments are pushed by the detector electronics into the ROS buffers. At the same time, the Level-1 system informs the HLTSV, providing trigger-specific event metadata. The HLTSV in turns assigns the event to the next available PU through a message to the respective DCM. The PU, based on the received event metadata, performs the necessary data requests to the ROS.

For the model result discussion, it is important to notice that not all event fragments are required for the PU to reach a decision. For this reason, for any given event, not all ROS servers are requested. In addition, those requested may be asked only for a subset of the served channels.

Once the PU processing is completed, the HLTSV is notified, such that the processing slot becomes newly available. Furthermore, for all completed events, the HLTSV notifies the ROS, such that the corresponding event fragments are dropped from the data buffer releasing the space.

III. MODEL INPUT AND OUTPUT DATA

The implemented model is driven by ATLAS operational data. Similarly, the results of the model are validated against a different set of operational parameters. Therefore two sets of operational values can be defined, namely the input and output variables of the model.

Figure 3: ROS request size distribution. The horizontal axis represent the ROS servers, while each column provides the corresponding request size distribution. The maximum number of fragments is bounded by the physical number of channels per ROS of 24.

Figure 4: Average fragment size distribution for the ROS system for a specific five minute interval used in the simulation.

Figure 5: Correlation between PU processing time and number of processed events for a specific simulation period. These data are used to correct the overall processing time as described in Section III-B.

The input parameters chosen for the model are summarized in Table I. They were selected based on the focus of the current

Parameter	Description	Source
Number of ROS servers	Total ROS count.	Database
Number of DCMs	Total DCM count.	Database
Number of PUs	Total PU count.	Database
Level-1 Rate	DAQ Input data rate.	Database
Fragment Sizes	Fragment sizes for each ROS link.	Database
Request Rate	Probability for each ROS to receive a request.	Database
ROS Request Size	ROS request size distribution.	Histogram
PU Delay	Processing time for each PU.	Database
PU Weight	Probability for each PU to get an event.	Database
Number of PU per DCM	PU count for each DCM.	Database

Table I: Simulation input parameters.

model: the role of ROS system in the data acquisition.

Most of the simulation input data are obtained from an ATLAS database keeping a record of the experiment's operational state over time [11]. For each operational variable, the database provides time-series information at the original publication rate. Most of the variables, like Level-1 rate or ROS request rate, are updated every five seconds. In order to suppress instantaneous fluctuations in the published values, time-series are averaged over a period of five minutes. Over such a time period, the experiment's operating conditions can be considered constant.

In addition to averaging, some of the input variables need additional processing, either to extract the required parameter or to correct distortions when combining multiple inputs.

One exception amongst the input parameters is the ROS request size, namely the number of fragments requested to a specific ROS. This information is derived from on-line histograms, as shown in Figure 3.

A. Fragment Size

Figure 4 shows the ROS fragment size distribution. The fragment size depends on the experimental data taking conditions at the time. The presented values are derived from the input bandwidth per ROS channel divided by the rate of events through the data link. Both values are archived every five seconds, therefore for a time interval of t seconds there are roughly $N_b = t/5$ records. For each bandwidth record b_i and input fragment rate r_i , the average fragment size for each data link is:

$$\frac{1}{N_b} \sum_{i=1}^{N_b} \frac{b_i}{r_i} \quad (1)$$

B. Processing Time

Figure 5 shows the average processing time calculated over five minute sampling intervals. The average processing time for the simulation is derived from the average processing time statistics archive, where each record for the average processing time p_i and the number of processed events q_i are stored every 30 seconds as the average over that time interval. For a time interval of t seconds there are roughly $N_p = t/30$ records.

Figure 6: ATLAS TDAQ OMNeT++ Simulation Design.

Since data are already averaged over 30 seconds, data has to be normalized by the number of events processed in each 30 second interval to be averaged over larger intervals. Therefore, the average processing time over N_p records is:

$$\frac{\sum_{i=1}^{N_p} p_i q_i}{\sum_{i=1}^{N_p} q_i} \quad (2)$$

The number of events processed by each PU is used to build the probability table for each PU to accept a new event.

C. Output Parameters and Simulation Validation

The parameters used to validate the simulation results are the ROS output bandwidth and the number of fragments stored in the ROS internal buffers. The simulation results are once again compared against the real operational data.

While the number of fragments is fetched from the operational time-series database, the ROS bandwidth data are provided by network monitoring tools that regularly poll the networking devices. For the purposes of this work, one shortcoming of these data sources is precision loss over time. While recent data are kept as five minute averages, older data are eventually stored as the average of 15, 30, and 60 minutes, depending on the age of the data.

IV. MODEL CONSTRUCTION

The TDAQ model and simulation for Phase-0 is built with OMNeT++ [4], a robust and user-friendly simulation framework, very popular in the network simulation community. In OMNeT++, simulation building blocks are defined as modules, implemented by C++ classes, and relationships between modules are defined in specialized configuration files.

The basic simulation model for the ATLAS TDAQ system is shown in Figure 6 as an OMNeT++ design, and follows what has been described in Section II. The design shown in Figure 6 is actually a simplified version of the actual OMNeT++ model. A simulation runs with ~ 1800 data links, ~ 100 ROS servers, ~ 2200 DCMs, and $\sim 40,000$ PUs. There is one Level-1 component, one HLTSV, and one Data Logger.

The model and configuration files are constructed dynamically by a script program which takes the data from the real

Figure 7: Single simulation error ratio distribution for the ROS output bandwidth. Positive values indicate the simulation results are larger than the operational data. Highlighted ROS servers present higher TCP retransmission rates in the real system.

system (described in section III) and sets the parameters for the simulation to run.

The network is assumed to be ideal and infinite in capacity, with no packet loss. Network protocol overhead is modeled as a fixed 46 byte increase every 1454 bytes.

The Level 1 Trigger is the source of data of the simulation, producing events at the configured L1 rate. For each event, a corresponding fragment is sent through all the data links to all the ROS buffers, and a single message to the HLTSV notifying it of the new event. The HLTSV in turn selects a free DCM based on the input data distribution, sends a new event message, and the DCM assigns the event to the next free PU.

The PU farm is a heterogeneous system, following the real system where there are machines with 8, 12, or 24 cores. In the simulation, each PU has its own average processing time and a distinct probability of taking a new event. First, the next available PU is selected from the set of available PUs, depending on its probability. Then, when the event is taken by the PU, a random number from an exponential distribution is drawn following the mean value assigned to the specific PU. The PU starts a timer to simulate the processing of an event. Finally, when the timer completes, the PU informs the HLTSV through the DCM that the event is finally processed. The simulation does not perform the actual processing of event data.

V. MODEL VALIDATION AND REFINEMENT

The model was initially used to predict the system operation for individual five-minute periods. Each simulation is run over 60 simulated seconds, for a total wall-time of about six hours on a Xeon E5645 CPU.

In this section the results of the model for a single period will be discussed, followed by a review of tests over longer periods.

Figure 8: Single simulation error ratio distribution for the number of buffered fragments. Positive values indicate the simulation results are larger than the operational data.

Figure 9: Comparison of the real operational data and model results for the ROS output bandwidth. Each value corresponds to the average over the ROS system. The comparison spans a time period of 120 minutes corresponding to 24 individual simulation runs.

The simulation framework collects statistics being used to compare the average values of the output of the simulation with the average values of the real system metrics data set. Metrics to validate the output of the simulation are the ROS bandwidth, ROS buffer memory usage, and the number of busy PUs.

A. ROS Output Bandwidth

Figure 7 shows the simulation results for the ROS output bandwidth for a single period. The model output for most ROS servers is within 3% of the actual data. Given the precision limitations of the network bandwidth operational data discussed in Section III-C, a limited disagreement be-

Figure 10: Comparison of the real operational data and model results for the number of fragments buffered in the ROS. Each value corresponds to the average over the ROS system. The comparison spans a time period of 120 minutes corresponding to 24 individual simulation runs.

Figure 11: Comparison of the real operational data and model results for the number of active PUs. Each value corresponds to the average over the PU farm. The comparison spans a time period of 120 minutes corresponding to 24 individual simulation runs.

tween simulation and data is not unexpected. This will be re-emphasized in Section V-C.

In addition to this, individual ROS servers may be affected by other sources of errors. For example the main outlier with 30% error has only one active link carrying a very small fragment size (~ 75 bytes). Since the error is a relative error, small variations in the absolute number result in a larger error compared to links with larger fragment sizes.

Another factor affecting the results are TCP retransmissions, whose level depends on the individual system load as well as the quality of the physical links. These are not considered in

the model, while it is known that in the real system some ROS servers show higher retransmission rates than others. For example the group of four highlighted in Figure 7.

B. Number of Buffered Fragments

Figure 8 presents the results for the number of fragments buffered in the ROS. The simulation results are mostly within 5% of the real data. A systematic bias on all the results can be appreciated. The bias represents a missing fixed latency of ~ 10 ms in the simulation.

Currently the model only accounts for the PU processing latency, which is the main contribution to the buffering time in the ROS. On the other hand, in the real system additional elements contribute to this time. Once the PU has reached a decision, the information needs to be propagated from the HLT farm to the ROS. This involves two network messages and three different software applications. Additionally, accepted events have to be built, contributing even more latency for data collection. Event building is the process of joining all the fragments of an event together in a single unit of data.

C. Longer term Simulation Results

Figure 9, Figure 10, and Figure 11 show the average results of 24 simulations taking consecutive segments of 5 minutes, corresponding to 120 minutes in total. Each point corresponds to one simulation. All the three figures show an outlier at minute ~ 70 . The real system had to stop taking data and the Level-1 rate had a drop due to external conditions. This violates the assumption that conditions are constant within each five minute slot and therefore explains the larger inconsistency between simulation results and real data.

Figure 9 shows the comparison between simulation and real metrics data for the average ROS output bandwidth. As discussed in Section III-C, the network bandwidth data have a limited resolution due to the archiving technology being used. In this case, the resolution is one hour, which explains the two constant values reported in the figure. They represent two different one hour slots within the chosen 120 minute period. The simulation results are bounded by the limited resolution values provided by monitoring of the real system.

Figure 10 shows the comparison between simulation and real metrics for the average number of pages buffered in the ROS system. There is a systematic bias in all the results of $\sim 5\%$, where the simulation is not taking into account all the latencies involved in the process, as described in the previous sub-section V-B.

Figure 11 shows the comparison between simulation and real metrics for the average number of busy PUs in the system. The number of busy PUs directly depends on the average processing time of the events in the real system, which is modeled in the simulation as an input parameter. The processing time of each event is represented in the simulation with high accuracy.

VI. OPERATIONAL ENVELOPE OF THE SYSTEM

The operational envelope of the system characterizes its efficiency under a combination of resource characteristics.

Figure 12: Operational envelope of the system for the combination of 1.5 MB event size and 100 kHz input data rate.

For a given input data rate and event size, the usage of the computing farm and memory buffers are evaluated as a function of the buffer size. The efficiency of the system is then defined as the product of farm usage and memory occupancy. Figure 12 shows the individual components of farm and memory usage, and the product of both curves as the efficiency of the system.

There are two distinct regions of operation for the system. On the left side of the efficiency peak, the system does not have enough buffer space to saturate the allotted computing resources. Since the limiting resource availability is the buffering space, an increase of any of the attributes (event size, input data rates, processing power) further reduces the efficiency of the system.

On the right side of the efficiency peak, the system is limited by the available computing power. Therefore an increase in deployed memory buffer does not contribute additional efficiency.

VII. RELATED WORK

Simulations of the ATLAS network can be found at [12] and [7], where packet drops, network latency, TCP retransmissions, and TCP incast pathology are analyzed.

Simulations related to the initial designs of the ATLAS trigger and data acquisition system can be found in [13] and [14]. In both works, the main concern was the overall understanding of system design choices, and the analysis of the latency distribution for the system while processing events under different conditions. In [13] some initial analysis for queuing effects is provided, and in [14] some initial analysis regarding packet loss, retransmissions, and traffic shaping are offered.

VIII. CONCLUSION

A simulation model for an existing data acquisition system was presented, reproducing with very high accuracy real data from previous operation of the system. Simulation errors were

understood, motivating the next stage of the work with the inclusion of missing latencies on the system and further validation of results over larger periods of time. The current model provides a very good basis for this study. The operational envelope represents the simplified provisioning of a complex data acquisition system, where the plan exists to revisit the concept and apply it to future data acquisition systems.

REFERENCES

- [1] ATLAS Collaboration, "Performance of the ATLAS detector using first collision data," *JHEP*, vol. 09, p. 056, 2010.
- [2] "ATLAS Phase-II Upgrade Scoping Document," CERN, Geneva, Tech. Rep. CERN-LHCC-2015-020. LHCC-G-166, Sep 2015. [Online]. Available: <https://cds.cern.ch/record/2055248>
- [3] K. L. Spafford and J. S. Vetter, "Aspen: A domain specific language for performance modeling," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 84:1–84:11. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2388996.2389110>
- [4] A. Varga, "Omnnet++," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 35–59.
- [5] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.
- [6] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls *et al.*, "The structural simulation toolkit," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 4, pp. 37–42, 2011.
- [7] M. Bonaventura, D. Foguelman, and R. Castro, "Discrete event modeling and simulation-driven engineering for the ATLAS data acquisition network," *Computing in Science & Engineering*, vol. 18, no. 3, pp. 70–83, 2016.
- [8] T. Colombo, H. Fröning, P. J. García, and W. Vandelli, "Optimizing the data-collection time of a large-scale data-acquisition system through a simulation framework," *The Journal of Supercomputing*, vol. 72, no. 12, pp. 4546–4572, 2016. [Online]. Available: <http://dx.doi.org/10.1007/s11227-016-1764-1>
- [9] Astigaraga, ME Pozo (on behalf of the ATLAS Collaboration), "Evolution of the ATLAS trigger and data acquisition system," in *Journal of Physics: Conference Series*, vol. 608, no. 1. IOP Publishing, 2015, p. 012006.
- [10] W. Vandelli, A. Borga, M. Joos, J. Vermeulen, J. Schumacher, J. Panduro-Vasquez, L. Tremblet, F. Wickens, G. Crone, A. Kugel *et al.*, "Evolution of the ReadOut System of the ATLAS experiment," *PoS*, p. 205, 2014.
- [11] I. Soloviev and A. Sicoe, "New Persistent Back-End for the ATLAS online information service," in *Real Time Conference (RT), 2014 19th IEEE-NPSS*. IEEE, 2014, pp. 1–6.
- [12] T. Colombo, H. Fröning, P. J. García, and W. Vandelli, "Modeling a large data-acquisition network in a simulation framework," in *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*. IEEE, 2015, pp. 809–816.
- [13] J. Vermeulen, S. Hunt, C. Hörtanag, F. Harris, A. Erasov, R. Dankers, and A. Bogaerts, "Discrete event simulation of the atlas second level trigger," *IEEE Transactions on Nuclear Science*, vol. 45, no. 4, pp. 1989–1993, 1998.
- [14] R. Cranfield, P. Golonka, A. Kaczmarek, K. Korcyl, J. Vermeulen, and S. Wheeler, "Computer modeling the atlas trigger/daq system performance," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 532–538, 2004.