Journal of Supercomputing manuscript No.
(will be inserted by the editor)

Optimizing the Data-collection Time of a Large-scale Data Acquisition System through a Simulation Framework

Tommaso Colombo · Holger Fröning · Pedro Javier Garcìa · Wainer Vandelli

the date of receipt and acceptance should be inserted later

Abstract The ATLAS detector at CERN records particle collision "events" delivered by the Large Hadron Collider. Its data-acquisition system identifies, selects, and stores interesting events in near real-time, with an aggregate throughput of several 10 GB/s. It is a distributed software system executed on a farm of roughly 2000 commodity worker nodes communicating via TCP/IP on an Ethernet network. Event data fragments are received from the many detector readout channels and are buffered, collected together, analyzed and either stored permanently or discarded. This system, and data-acquisition systems in general, are sensitive to the latency of the data transfer from the readout buffers to the worker nodes. Challenges affecting this transfer include the many-to-one communication pattern and the inherently bursty nature of the traffic. The main performance issues brought about by this workload are addressed in this paper, focusing in particular on the so-called TCP incast pathology. Since performing systematic studies of these issues is often impeded by operational constraints related to the mission-critical nature of these systems, we developed a simulation model of the ATLAS data-acquisition system. The resulting simulation tool is based on the well-established, widely-used OMNeT++ framework. This tool was successfully validated by comparing the obtained simulation results with existing measurements of the system's behavior. Furthermore, the simulation tool enables the study of the theoretical behavior of the system in numerous what-if scenarios and with modifications that are not immediately applicable to the real system. In this paper, we take advantage of this to analyze the behavior of the system using different traffic shaping and scheduling policies, and with network hardware modifications. This analysis leads to conclusions that could be used to devise future system enhancements.

T. Colombo · W. Vandelli
Physics Department, CERN, Geneva, Switzerland
E-mail: Tommaso.Colombo@cern.ch

T. Colombo · H. Fröning
Institut für Technische Informatik (ZITI), Universität Heidelberg, Mannheim, Germany

P. Garcìa
Dep. de Sistemas Informàticos, Universidad de Castilla-La Mancha, Albacete, Spain

Acknowledgements This work was partially supported by the Ministry of Economy and Competitiveness of Spain (project TIN2012-38341-C04) and the European Commission (ERDF).

1 Introduction

ATLAS [10] is a high-energy physics experiment installed at CERN, Geneva, Switzerland. Its scientific program is mainly focused on the discovery and study of rare particle physics phenomena. The experiment's detectors observe proton-proton collision events delivered by the the Large Hadron Collider (LHC) at a design frequency of 40 MHz. Each event corresponds to 1-2 MB of data, which are used to reconstruct the physical processes produced by the collisions. If all collision events were to be read out and acquired, ATLAS would produce 80 TB/s.

The sheer amount of data produced by ATLAS cannot be all stored for later analysis. Instead, as with many other large scale experiments, a real-time data-acquisition and data-selection system is necessary. These systems are usually implemented with a mix of custom hardware and software running on commercial off-the-shelf (COTS) hardware, and their performance and reliability have a strong impact on the experiment as a whole. Failures and poor performance result in the permanent loss of extremely valuable experimental data.

Due to the mission-critical nature of these systems, a systematic study of their performance envelope is often impeded by operational constraints, such as system availability requirements or limited opportunities of performing hardware or system software modifications. A simulation model can thus be a worthwhile alternative, assuming that it is accurate enough to reliably reproduce the key traits of the system.

This publication reports on the development of a simulation of the ATLAS data-acquisition system, used as a case study. The most significant source of performance degradation in data-acquisition networks, the TCP incast phenomenon, is described along with the traffic patterns that cause it. Measurements showing its significance are presented, and the currently employed mitigation strategy is described. The paper introduces the basic assumptions underlying the simulation model and presents the validation tests undertaken to ensure that the model can reliably reproduce the behavior of the real system.

Having established that the most important features of the system are accurately simulated, the focus shifts to exploring what-if scenarios that would otherwise be difficult to study with the real system. Various traffic scheduling policies are simulated and evaluated; the system performance is assessed under realistic conditions; the impact of the network hardware buffer depth is gauged. These experiments lead to conclusions that will prove useful in improving the performance of the ATLAS data-acquisition and of other systems experiencing the same traffic patterns.

2 Background: the ATLAS Trigger and Data-Acquisition system

The ATLAS Trigger and Data-Acquisition (TDAQ) system [7] is responsible for the selection of interesting collision events (*triggering*, in high-energy physics jargon) reducing the initial frequency of 40 MHz to ~1 kHz of stored events. This requires an

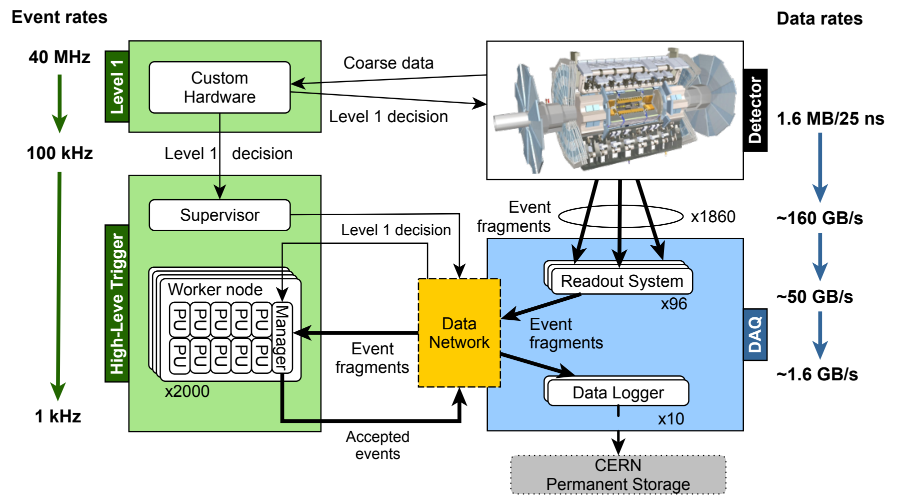


Fig. 1 Logical view of the ATLAS Trigger and Data-Acquisition system.

overall trigger rejection factor of the order of 10^4 against unremarkable events, while retaining potential candidates containing new physics processes, such as Higgs boson decays. The TDAQ system, outlined in Figure 1, is based on the combination of a hardware-based first stage and a software-based second stage.

The first stage, called Level-1 trigger, is a synchronous pipelined electronics system, with a guaranteed maximum latency of 2.5 µs. It selects events using coarse-grained data from a subset of the experiment's detectors. It triggers the readout at a maximum rate of 100 kHz. As part of the selection process, it identifies regions-of-interest (RoI): parts of the detector that recorded interesting signals that are used as a seed for the second stage.

When an event is accepted by the Level-1, its data fragments (1860 fragments of variable sizes around 1 kB, corresponding to different regions of the experiment) are readout by the detector electronics. Each fragment is pushed via a custom point-to-point optical link into a specific hardware buffer in one of the ~100 "readout system" nodes, which act as the coupling between the first stage's custom electronics and the second stage's COTS hardware.

The second stage, called High-Level Trigger (HLT), is a distributed software system running on around 2000 Linux PCs interconnected by an Ethernet network. Each HLT worker node hosts one HLT "processing unit" per CPU core and a single "data-collection manager", which handles communications with the rest of the system on behalf of the node's processing units. A central scheduler, the HLT "supervisor", receives regions-of-interest information from the Level-1, and assigns it to one of the available HLT processing units. Using the regions-of-interest as starting points, the processing unit incrementally retrieves and analyses event fragments, until a decision can be taken. The event can be rejected even without analyzing all its fragments, thus limiting the fraction of data to be retrieved. Fragments of rejected events are deleted from the readout systems buffers, while accepted events are transferred to one of the Data Logger nodes for storage. The rate of events that can be accepted is mostly de-

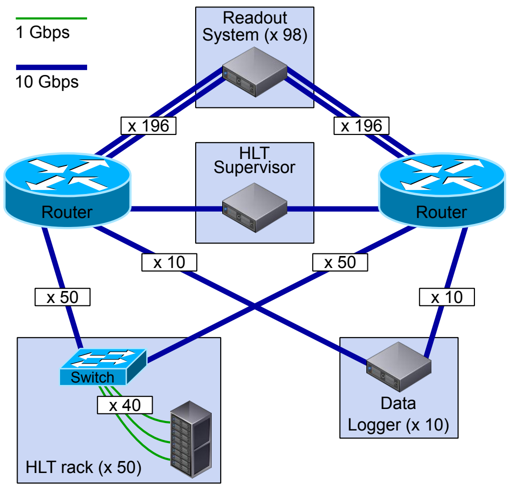


Fig. 2 Network architecture of the ATLAS Data-Acquisition and High-Level Trigger system.

termined by the availability of computing and storage resources for subsequent data analysis, and is foreseen to be around 1 kHz for the 2015–2018 data taking period.

The physical layout of the Data-Acquisition and High-Level Trigger system is represented in Figure 2. The core of the system consists of two large network routers with a maximum capacity of several hundred 10GbE ports. Readout systems are directly connected to both core routers with 4 10GbE links. HLT worker nodes are organized in racks of at most 40 nodes. Each node in a rack is connected to an aggregation switch with a GbE link. The rack switches have 10 GbE links to both core routers. For the rationale behind this design, please refer to [21].

3 Performance issues in data-acquisition networks

3.1 Traffic pattern

Data-acquisition networks have to deal with a particularly problematic traffic pattern:

– The communication pattern is many-to-one.
– Data are transmitted in multiple bursts, rather than a smooth flow.

These two characteristics do not reflect a design error, but rather the nature of data acquisition itself. The purpose of a data-acquisition system is to gather together data fragments from the different components of an experiment, therefore at least in one stage of the system the traffic pattern will be necessarily many-to-one. The data transfers are also normally driven by the experiment itself, which in general does not produce a continuous flow of data. Instead frequent bursts of data are generated in correspondence with the experiment detecting a phenomenon.

In the specific case of ATLAS, event data are striped over all the readout systems, since each system buffers data from a specific region of a detector. This is a hard constraint: the detectors are connected to the readout systems via many point-to-point

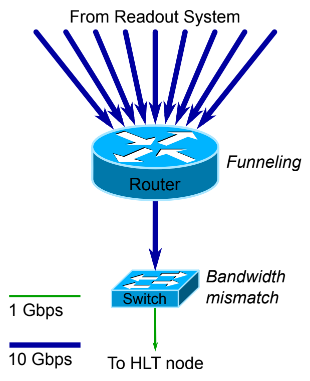


Fig. 3 Visualization of the potential network congestion issues.

links, rather than a network. Each link carries data fragments from a specific module of a detector to a specific readout system (on average, each readout system handles ~20 point-to-point link inputs). On the other hand, the selection algorithms running on a single HLT processing unit normally need fragments from multiple detector modules at the same time. As the fragments are already available in the buffers of multiple readout systems and are sent as soon as the request reaches the readout systems, many nodes will start sending fragments at the same time to the same destination, thus creating instantaneous network congestion (see Figure 3).

3.2 The TCP *incast* pathology

As mentioned in Section 1, the ATLAS data-acquisition system is based on Ethernet. The messaging among applications relies on the TCP protocol. The combination of TCP, a lossy link layer such as Ethernet, and the traffic pattern described above is subject to a well-known TCP pathology called *incast* [20]: first observed in data-center storage networks, it occurs “when a client simultaneously receives a short burst of data from multiple sources, overloading the switch buffers associated with its network link such that all original packets from some sources are dropped”.

TCP has both an active packet-drop detection mechanism, in which the receiver detects the lost packet and causes the sender to react very quickly, and a passive mechanism, which is based on retransmission timeouts (*RTO*) on the sender side [9]. The RTO value is calculated starting from the estimated round-trip time [19]. However, to protect against spurious retransmissions and interference with other TCP features such as delayed acknowledgments, the RTO has a fixed minimum, specified as 1 s in [19], with lower values used in actual implementations (200 ms in Linux, 30 ms in FreeBSD). These values are orders of magnitude larger than the round-trip time of a data-center network, which is usually in the sub-millisecond range. In addition, the active mechanism requires at least some of the original packets to reach the destination for the packet loss to be detected. When this is not the case, the delivery of packets incurs the large delay caused by the RTO.

3.3 Impact and mitigation

The data-collection latency (i.e., the latency of the data transfer between the data sources and the processing nodes) is critical to the performance of a data-acquisition system as a whole. In the case of ATLAS, each of the HLT processing units operates exclusively on the single event assigned to it, with the HLT selection proceeding iteratively starting from the region-of-interest identified by the Level-1, collecting data fragments incrementally as needed. A processing unit is blocked while it waits for the data fragments to be collected, which means that the data-collection latency effectively translates into lost CPU time. With average per-event processing times of the order of 100 ms, waiting for a TCP RTO is very wasteful.

The TCP incast problem is well studied in literature, with many solutions being proposed. In general the solutions point in the direction of either specialized hardware (i.e. switches with huge buffers), or alterations of the TCP implementation. Due to the operational constraints mentioned in Section 1, none of these can be readily deployed in the ATLAS data-acquisition system, as they would require significant modifications to the existing hardware or to core system software (specifically the kernel TCP implementation). Even assuming that the existing networking hardware can be replaced, cost and availability constraints might prevent the purchase of deep-buffered switches, which are an increasingly niche product in the datacenter market.

A less invasive, but less general, solution is application-level traffic shaping. In ATLAS, smoothing the rate of data requests generated by a HLT node can alleviate the network congestion by controlling the maximum size of the traffic burst from the readout systems [13]. Obviously such a smoothing mechanism imposes a trade-off: excessive smoothing can increase the data-collection time by unnecessarily delaying the requests for data, whereas insufficient smoothing will not eliminate packet drops.

The ATLAS data-acquisition software currently uses a credit-based traffic-shaping algorithm, implemented in the Data-Collection Manager running on every HLT node. Its basic rules are as follows.

- Each HLT worker node has a fixed number of credits available. All the HLT processing units on a node share these credits.
- Each data request from a processing unit to a readout system uses as many credits as the number of fragments it asks for.
- Each response returns the credits used by the corresponding request.
- If all available credits are used, further requests are blocked until the necessary credits become available.

The number of fragments in a request gives a rough estimation of the size of the corresponding response. Therefore, this algorithm effectively limits the maximum burst size of data transfers directed to the same HLT node. However, it relies on the assumptions that all event fragments are similar in size, and that this size is known beforehand. These assumptions are reasonable for ATLAS under normal operating conditions, but not necessarily for other systems or scenarios.

4 Measurements

Detecting buffer overflows in an Ethernet network is a relatively simple task: the Linux kernel provides the total number of packet retransmissions that have occurred in a TCP connection, and some commercial switch models report cumulative counts of dropped packets per port via SNMP. In-depth analysis of the root causes is not as straightforward: important metrics, such as switch buffer occupancies, are either not available at all or too coarse-grained to be useful. This, combined with a protocol with sophisticated congestion control algorithms, such as TCP, leads to an extremely intricate landscape. On the other hand, generating synthetic traffic patterns, which can be tightly controlled and are known in advance, proves instrumental in reducing the complexity down to a manageable level. The tests described in the following paragraphs employ this approach.

4.1 Test set-up

The measurements were performed using some of the hardware available in the ATLAS data-acquisition system. At the time of these tests, the infrastructure was still under consolidation, so the test set-up is slightly different from the one described in Section 1. In particular, readout systems were not directly connected with 10GbE links to the core routers. They were instead connected with GbE links to an intermediate aggregation switch with a 10GbE uplink to each core. The intermediate switch was kept under-subscribed so that no congestion could appear there.

Given these constraints, the following test set-up, shown in Figure 4, was chosen:

- 10 readout system groups:
  - Each group consists of 16 nodes connected to one switch (160 total).
  - 12 event fragments of 1.1 kB are served by each node (1920 total fragments, 2112 kB full event size).
- 1 HLT rack:
  - It consists of 39 PCs connected to one switch.
  - Each PC hosts 24 HLT processing units (936 total).

This configuration was chosen because it provides a realistic model of the expected network buffer usage in the final system topology. In particular, the congestion at the rack-level switch is well represented. The small amount of HLT nodes in use with respect to the complete system does not prejudice the usefulness of this set-up: the network congestion phenomena individually affect the output buffers of the network ports that are actually in use. The obtained results should scale reliably with the higher number of HLT racks in the complete system.

The core routers are Brocade MLXe-32 [1] modular switches with 10GbE line cards. They use input-buffering with a peculiar implementation of switch-level virtual output queuing: input ports are grouped in modules of 8 ports at most and each module maintains multiple, distinct queues to every output port on the router. As these routers are geared towards Internet Service Providers, they are equipped with very deep buffers: each module has a packet memory of 1.5 GB.

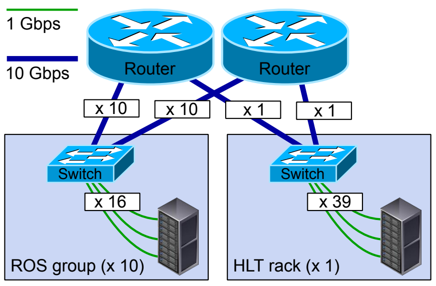


Fig. 4 Test set-up.

The top-of-rack switches use output-buffering. Different models were tested. This paper focuses on two representative models:

- Switch 1 (HP ProCurve 6600-48G-4XG [4]): a switch with per-port dedicated buffers of 750 kB, of which ~600 kB are available for standard-priority packets
- Switch 2 (Brocade VDX 6740T [2]): a switch with two buffers of 12 MB each, shared by 32 ports each, with a per-port limit of 8 MB

Among the synthetic traffic pattern tested, this paper reports on the so-called full event building pattern:

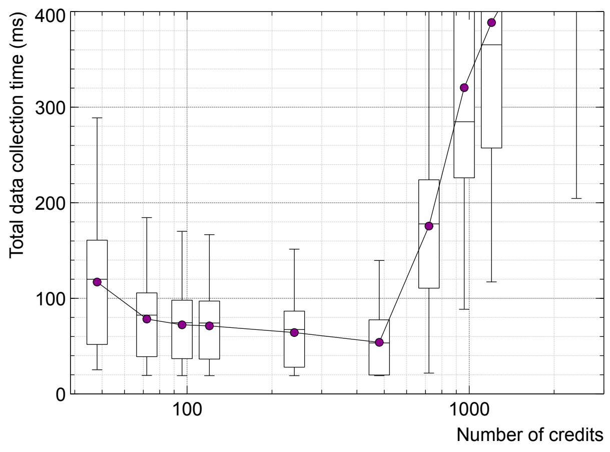
- Events are assigned by the HLT supervisor to HLT processing units at a constant rate.
- The processing units immediately collect all fragments of assigned events, process them for a fixed amount of time, and ask for a new assignment.

It should be noted that this pattern is the harshest in terms of generated traffic bursts: since the data corresponding to an event are collected all at once, the maximum burst size corresponds to the event size. On the other hand, the fixed-size fragments enable the simple traffic-shaping algorithm to operate most effectively.

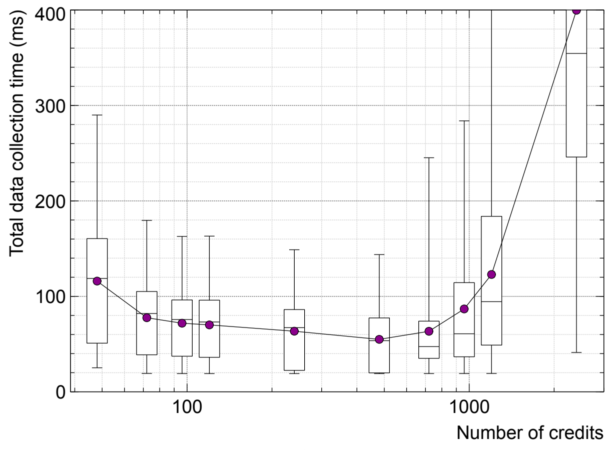
The assignment rate parameter for the presented tests was selected with the goal of utilizing a sizable portion of the available bandwidth of the HLT rack uplinks, while maintaining a comfortable margin to avoid the effects of link saturation. The chosen 750 Hz assignment rate corresponds to a total throughput of $\sim$ 13 Gb/s. Due to performance constraints, the only policy that the supervisor can use when choosing the processing unit to which an event is assigned is first-come first-served (FCFS), i.e. events are assigned to processing units in order of arrival of their assignment requests.

4.2 Results

The results of the measurements are shown in Figure 5. The total data-collection time per event is influenced both by the network conditions and by the traffic-shaping mechanism. With too few traffic-shaping credits available, the large data-collection time is due to collection inefficiency because the HLT nodes cannot fully utilize the network bandwidth. This is analogous to a TCP connection with a congestion window smaller



(a)



(b)

Fig. 5 Data-collection latency as a function of the number of traffic-shaping credits, for (a) switch 1 and (b) switch 2. The test conditions are detailed in Section 4.1. The bullets represent the average values. The horizontal box lines represent the first quartile, the median, and the third quartile. The box whiskers represent the first and the 99th percentile.

than the bandwidth-delay product. With too many traffic-shaping credits the top-of-rack switch buffers are overflowed, triggering the TCP incast pathology.

Packet drops are not an issue in the core router due to the deep buffers it offers. Nevertheless, deep buffers were not the main driver behind that choice of hardware, but rather the capability of aggregating hundreds of ports in the same router was. With a different model, packet drops in the core router would certainly influence the data-collection performance of the system and would need to be taken into account.

In the following sections, these measurements are used as the baseline for verifying that the simulation models developed correctly reproduce the system behavior. For a more in-depth look at similar tests on the ATLAS data-acquisition system, please refer to [13]. One key consideration is worth reporting: it is possible to estimate the minimum possible value for the data-collection latency to use as a reference point. For this purpose, the time it takes for data requests to reach the readout systems and the time it takes for the readout systems to prepare the data can be neglected[1]: the dominant component is the time it takes to transmit the data fragments on the slowest link in the path. In this set-up, this corresponds to the amount of time it takes to transmit a full event on the GbE link from the rack-level switch to the HLT node, which is ~18 ms. It is worth noting that while the credit-based traffic-shaping algorithm is successful in keeping the data-collection latencies under control when configured correctly, it also prevents the average data-collection latency from reaching that minimum, as evidenced by the latency distributions shown in the figures.

5 Simulation model development

5.1 Simulation framework

The simulation presented in this paper uses the OMNeT++ discrete event simulation[2] framework [23]. OMNeT++ is free for non-commercial use and its source code is available. It was chosen for two main reasons: its wide acceptance in the academic community and its ease of use for the purpose of modeling different types of networks, such as computer networks, wireless networks, sensor networks, interconnection networks of High-Performance Computing (HPC) systems, or interconnection networks supporting massive-storage or big-data systems [5, 17, 3, 24, 18]. In OMNeT++ simulations are composed of modules, defined in the declarative NED language, which communicate exchanging messages via module-to-module channels. So-called simple modules are the active components of the simulation and are implemented in C++, leveraging the class hierarchy provided by the simulation framework. Modules can be grouped together to form compound modules and networks. The development of network simulations is aided by built-in support for physical channels, with latency, transmission delay, and message loss properties.

The simulation model described here is based on the INET Framework of OMNeT++. INET is a protocol model library which includes detailed implementations of all network layers, from the MAC layer onwards. In particular, the Ethernet, IP, and TCP [22] implementations are used.

5.2 Hosts and applications

Data-acquisition applications are implemented on top of the standard INET host model, as shown in Figure 6. Applications interact only with the TCP module. A useful feature of INET's TCP module is that it supports a data transfer mode that preserves application-level message boundaries: e.g. if an application sends a 1 MB message

[1] Measurements show that both time intervals are smaller than 0.5 ms.

[2] In this particular sentence, the word "event" refers to simulation events. To avoid confusion, throughout the rest of the paper, "event" will only be used in its high-energy physics meaning, i.e. "collision event".

the receiver application will receive the same message, after TCP has completely finished simulating the transmission of 1 MB over the connection. This greatly simplifies the modeling of message-based applications, like those in the ATLAS data-acquisition software.

In order to simulate the test system presented in Section 4.1, four applications need to be modeled: the readout systems, the HLT supervisor, the data-collection manager, and the HLT processing units.

In the real system, the HLT supervisor, which assigns events to processing units, and the readout systems, which serve event data fragments, are data-driven applications: their behavior is dependent on that of the experiment and the Level-1 trigger. In principle, this would require models of those applications to follow traces recorded on the real system. However, this is not necessary when trying to reproduce the synthetic traffic patterns described in Section 4.1. The readout systems become trivial server applications, responding to fragment requests with a configurable delay and response size. The supervisor instead is reduced to a periodic scheduler. The processing units send a message to the supervisor when they are available, i.e. when they are ready to start processing another event. The supervisor stores this information and uses it to assign events to processing units at a configurable global rate, by sending assignment messages.

As already mentioned, OMNeT++ simple models are implemented using the C++ programming language. Since the same language is used throughout the ATLAS data-acquisition software, the application-level code that is relevant to the simulation model can be ported to the simulation environment with minimal changes. This ensures a bug-for-bug compatible reproduction of the applications' behavior within the model. This approach is used for modeling the processing units, which generate the data requests for an event, and the per-node data-collection managers, which act as proxies between the node's processing units and the readout systems. In particular, a processing unit can simulate per-event iterative collection and processing of data: after it receives an event assignment, it can request data from the readout systems with a configurable pattern, emulate processing by waiting for a configurable amount of time, and repeat this process several times before considering the event fully processed and asking for another one from the supervisor. Just like in the real system, the processing units of a worker node do not interface directly with the TCP module: their communications are mediated by the per-node data-collection manager. Its most relevant functions in the context of this model are the mapping of each data request from the processing units to messages to multiple readout systems the enforcement of the traffic-shaping algorithm described in Section 3.3.

5.3 Network switches

For the purposes of this simulation, the most relevant aspect of the network hardware is packet buffering. The internal architecture of the switches is not modeled in detail. It is assumed that the switch speedup is high enough to prevent input head-of-line blocking, and to make the packetization delay of the switch cells negligible. With these assumptions, switches are modeled as follows. For each switch port, an INET

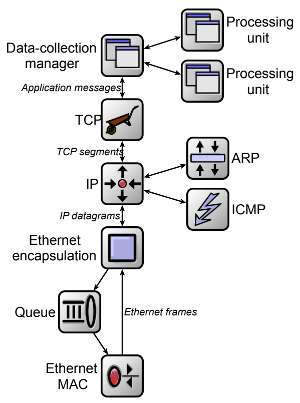


Fig. 6 Example of a model a host: HLT working node with 2 processing units.

Ethernet MAC module acts as the interface between the physical transmission channel and the switch. It sends incoming frames to an ideal frame relay unit, which maintains the switch's MAC address table and instantaneously forwards frames towards their destination port (or broadcasts them if the destination is not yet present in the address table). One or more "packet droppers" intercept frames between the relay unit and the switch output queues. These modules selectively drop frames, depending on the total buffer space available in the queues that are connected to their outputs, effectively defining the switch buffering scheme. Frames that were not dropped are stored in the switch's output queues, waiting for the Ethernet MAC to pull them from the queue when the transmission channel is ready.

Two basic buffering schemes are considered: dedicated and shared. In the dedicated buffers model, shown in Figure 7a, there is a dropper for every output port, effectively modeling tail-drop queues. In the shared buffer model, shown in Figure 7(b), a single dropper guards all the output ports. The two basic schemes can naturally be integrated to model more complex architectures, e.g. with buffer space limits both on a per-switch basis and on a per-port basis.

5.4 Complete model

The components described in the previous sections are assembled to create a model of the test system described in Section 4.1. Figure 8 shows the complete model as displayed by the simulation framework's graphical environment. The observed simulation run-time depends on the rate of events scheduled by the supervisor. The relation between simulated time t and simulation runtime T is approximately given by: $T = t \cdot r/R$, where r is the supervisor assignment rate, and R is a constant which is roughly equal to 5 Hz when running the simulation on a modern CPU. As an example, this gives a runtime of ca. 100 minutes for simulating 30 s of a system running at 1 kHz.

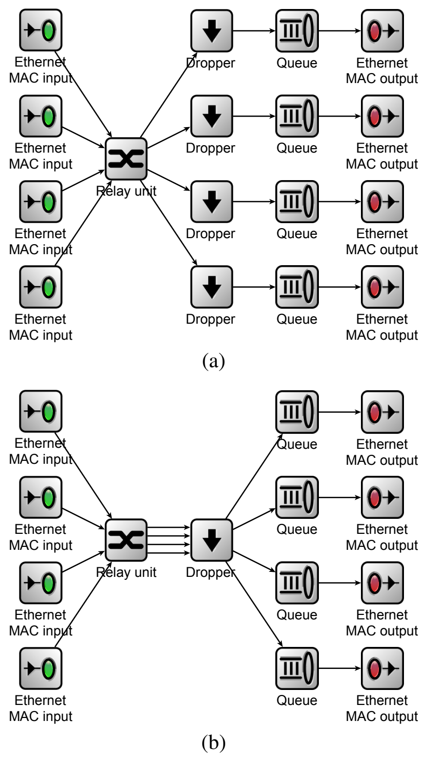


Fig. 7 Models of output buffered switches with (a) dedicated per-port buffers and (b) shared buffers.

6 Analysis and comparison of measured and simulated results

The model outlined in Section 5 has some obvious approximations, the most important one being the simplified switch architecture. This is not only a design choice: details on the architecture of commercial network equipment are extremely scarce and, when available, are geared towards marketing rather than engineering. Another potential source of inaccuracies in the simulation is the TCP congestion avoidance algorithm. The most commonly used protocol model libraries only provide the "traditional" TCP variants: Tahoe, Reno, Vegas and New Reno. Support for more recent variants such as TCP CUBIC [14] (the algorithm currently in use in Linux) relies on models built by modifying the Linux kernel TCP stack for use in the simulation. To avoid this extra complexity, for the results presented here New Reno [15] was selected. While none of the mentioned algorithms can effectively prevent the incast problem, their impact on other facets of the simulation might render it unreliable. For these reasons, it is crucial to validate the model against the measured behavior of the system, before it is used to draw conclusions on scenarios that cannot easily be tested in practice.

As explained in Section 3.3, the key application performance metric is the average data-collection latency per event. Therefore, the focus here is on comparing the mea-

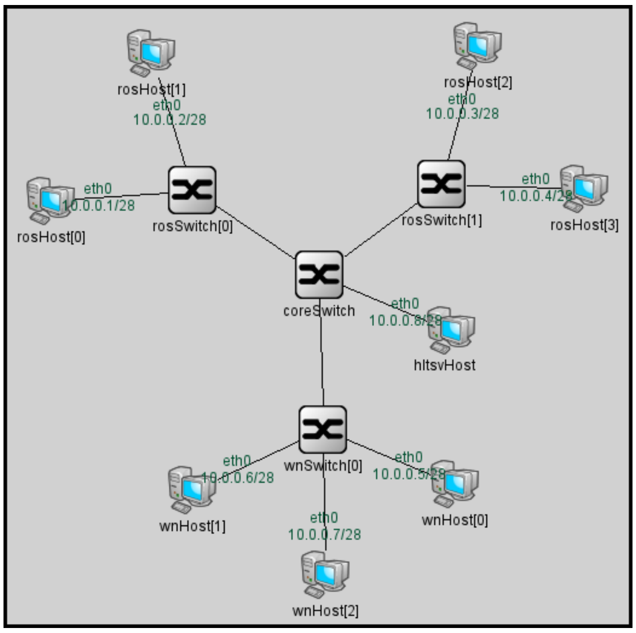


Fig. 8 Screenshot of the OMNeT++ graphical representation of the simulation model. The number of hosts was greatly reduced for clarity.

sured data presented in Section 4.2 with the simulated latencies, as shown in Figure 9.

The most important feature to reproduce is the onset of the TCP incast pathology, evidenced by the abrupt increase in data-collection latency as the traffic-shaping algorithm allows an excessive number of data requests to be in-flight. The model is successful in this. When simulating Switch 1, the onset point is identical at 480 traffic-shaping credits, corresponding to a maximum per-node burst size of ~550 kB. This is expected, as the switch has 600 kB per-port output buffers. When simulating Switch 2, the incast onset points of measurement and simulations are close, but not identical, with the simulation being more forgiving. This can be explained by an overly optimistic choice of buffering parameters in the simulation. Taking the manufacturer's parameters at face value, the switch's shared buffer memory is 12 MB. However, some of that memory is reserved for quality-of-service purposes and cannot be used by standard-priority packets. Indeed, with a simulated buffer size of 10 MB, the incast onset point coincides in simulation and measurement.

The model of the traffic-shaping algorithm benefits from the code sharing with its actual implementation. As a consequence, the simulation is particularly accurate in the region of the parameter space where the latency is heavily influenced by the shaping, i.e. where the algorithm is effective in preventing packet drops. This can be confirmed comparing the distributions of the data-collection times, rather than just their mean values.

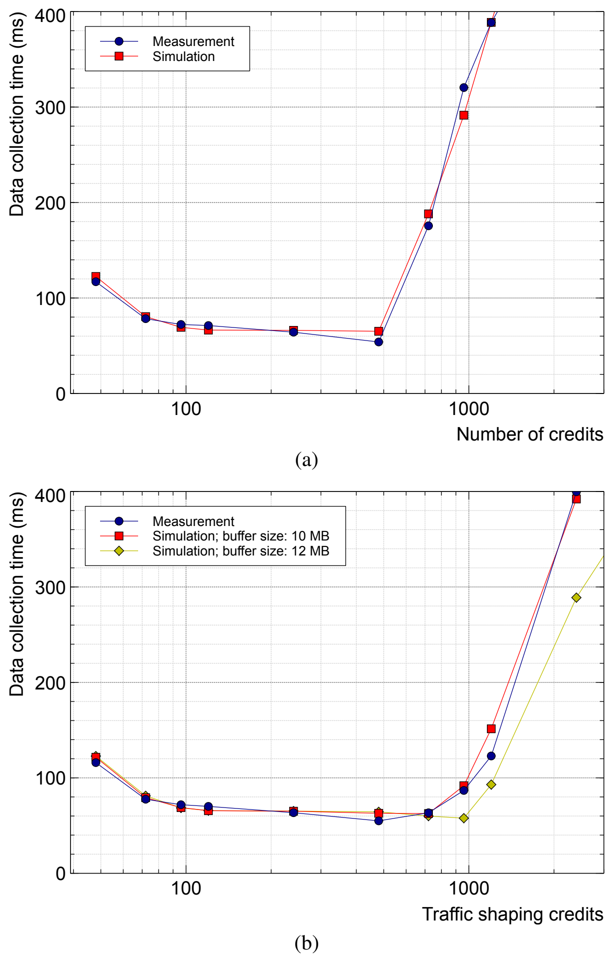
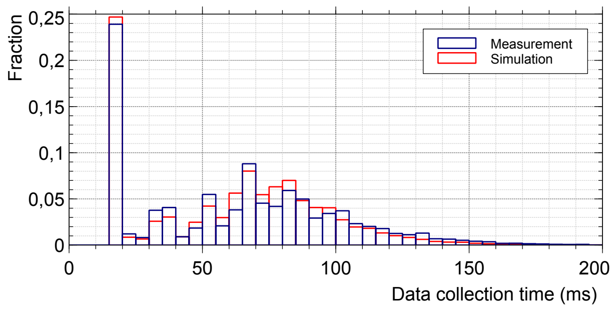


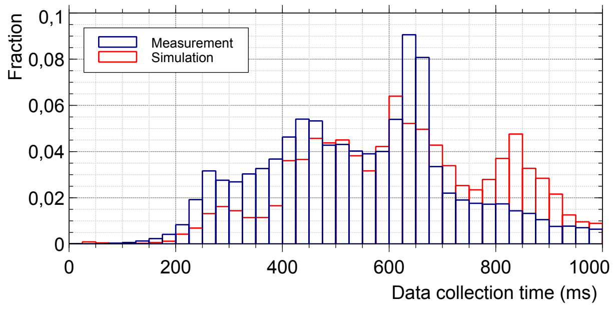
Fig. 9 Comparison of measured and simulated data-collection latencies for different settings of the traffic-shaping algorithm, using (a) Switch 1 and (b) Switch 2.

The two superimposed histograms in Figure 10a show the measured and simulated distributions for Switch 1 at 240 traffic-shaping credits, i.e. with packet drops completely prevented. The histograms are in very good agreement and demonstrate the effect of the traffic-shaping algorithm. A sizable portion of the events are fully collected within 20 ms, which is compatible with the minimum latency estimated in Section 4.2 (18 ms), presumably because all their fragments were collected when no other events were competing for the same credits. The other events need to wait for enough credits to become available, hence the long tail of the distribution.

By contrast, Figure 10b compares the data-collection times at 2400 credits, i.e. with a mostly ineffective traffic-shaping. The histograms are still similar, however some remarkable differences appear. In both distributions, very few events manage to be fully collected within the first 200 ms, which means that during most data-



(a)



(b)

Fig. 10 Comparison of measured and simulated data-collection latency distributions, using Switch 1, at (a) 240 traffic-shaping credits and (b) 2400 traffic-shaping credits. Note that the scales of the two x-axes are different.

collections at least one connection to a readout system suffers enough packet drops to cause a TCP RTO. Compared with the simulated distribution, the measured distribution has a greater fraction of the events being fully collected before 600 ms (i.e. three TCP RTOs), where it also exhibits a stronger peak. These differences are due to the different TCP congestion control algorithms being used: CUBIC for the measurements, New Reno for the simulations. After a timeout, CUBIC can quickly grow its congestion window back to the previous size, while New Reno enters its slow-start phase, which grows the congestion window much less aggressively. This explains why the simulated distribution is smoother and its tail contains a greater fraction of the events. Notwithstanding these discrepancies, the simulation still reproduces the most significant consequence of the incast phenomenon: almost all data-collections are affected by at least one TCP RTO.

7 Parameter studies

7.1 Supervisor assignment policies

With the consistency of the simulation model with the real system measurements reasonably established, the focus can shift to more simulated scenarios that could not easily be enacted in practice.

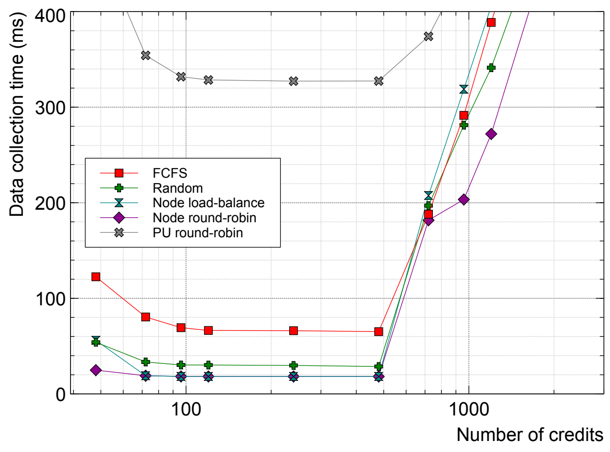
One such scenario is applying a different event-scheduling policy in the HLT supervisor. As mentioned in Section 4.1 performance limitations currently prevent event-scheduling policies more complex than FCFS from being implemented in the supervisor. The simulation is not affected by such issues so alternative policies can be modeled. Besides FCFS, four event-scheduling were implemented:

- Random: the supervisor chooses a random processing unit out of all of those that requested an event assignment, without regard for the order in which they did so.
- Node round-robin: the supervisor chooses a processing unit running on the next node in a fixed sequence of nodes.
- Node load-balancing: the supervisor chooses a processing unit running on the node with the most idle units.
- Processing Unit round-robin: the supervisor chooses the next processing unit in a fixed sequence of units, where units running on the same worker node are consecutive

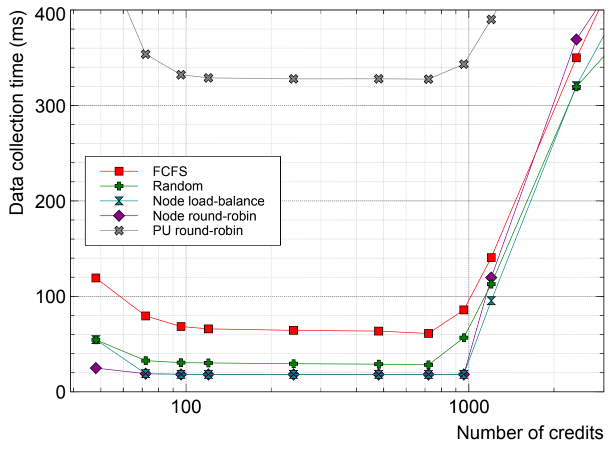
The results are shown in Figure 11. The change in policy from FCFS to random leads to a very significant reduction of the data-collection latency, especially in the region of the parameter space where the traffic-shaping algorithm prevails. The explanation for this difference lies in the mapping of processing units to nodes. While the traffic-shaping credits limit is applied on a per-node basis, the event scheduling (and associated data collection) happens on a per-processing-unit basis. In the particular set-up used, there are 24 units per node (see Section 4.1). The random policy reduces the probability that two or more events will be assigned in a short interval to units hosted by the same node. Units on different nodes do not compete for the same credit pool and switch buffer space, ultimately resulting in lower average data-collection latency.

The node load-balancing policy further reduces the chance of two events being assigned to the same node, which translates to further reductions of the data-collection time in the region where traffic-shaping is effective. The node round-robin policy, which completely prevents the supervisor from choosing the same node consecutively, unsurprisingly performs better than all the others when the nodes have very few credits available. It also has a clear advantage over the others in the incast-affected region when using Switch 1, does not have the same advantage when using Switch 2. This is caused by the different buffering schemes in the two switch models: with per-port dedicated buffers assigning two events to the same node consecutively can have disastrous consequences: if the buffers are still filled with the fragments of the first event when the second one is assigned, all the fragments of the second event are going to be dropped. With a shared buffer the consequence of consecutive same-node assignments is just an increase in buffer occupancy, which has less drastic effects.

Finally, the processing unit round-robin policy is an example of the results of a bad policy choice: by concentrating the data-collection on the same node at the same time, all 24 processing units on a node compete for the same credits, resulting in abysmal performance.



(a)

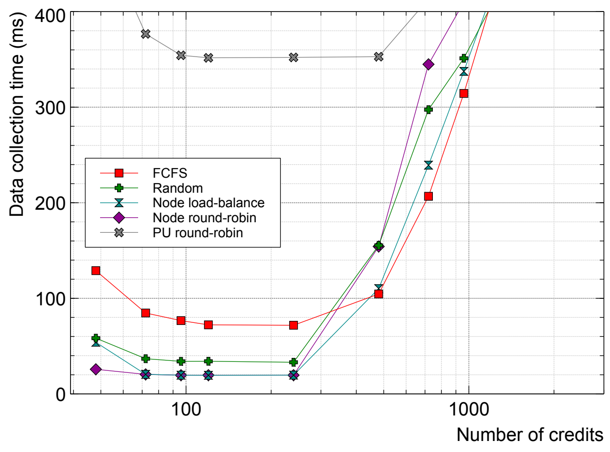


(b)

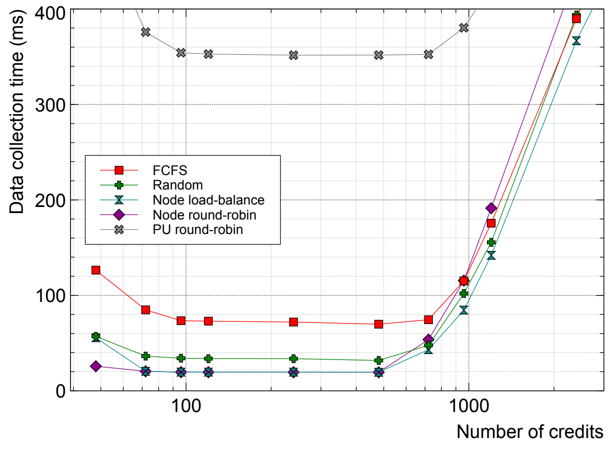
Fig. 11 Comparison of the effect of different HLT supervisor event assignment policies on the data-collection latencies for different settings of the traffic-shaping algorithm, using (a) Switch 1 and (b) Switch 2.

7.2 Variable fragment sizes

Both the measurements described in Section 4.1 and the simulation results presented so far use a fixed single size for all the data fragments (1.1 kB). In less controlled conditions, however, fragments sizes depend on the characteristics of the particular phenomenon observed by the detectors. In high-energy physics, the fragment size distribution is generally quite asymmetric with a long tail: "ordinary" events tend to be small and appear rather frequently; events corresponding to rare interesting phenomena in the detector tend to have bigger fragments. Moreover, within an event, fragment sizes are not all identical: they depend on a multitude of factors, including the observed phenomenon and the region of the detector they correspond to.



(a)



(b)

Fig. 12 Data-collection latencies with variable event sizes, for different event assignment policies and different settings of the traffic-shaping algorithm, using (a) Switch 1 and (b) Switch 2.

Given that the traffic-shaping algorithm presented in Section 3.3 uses the number of fragments in a data request as an estimate of the size of the corresponding response, its performance is expected to worsen when the fragment sizes are not always the same. The simulation model can help quantify this.

In this particular example, the fragment sizes are modeled as the sum of two components. The first component is different for every event, while the second component is different for every fragment of an event. The first component is extracted from a log-normal distribution[3], modeling the long-tail distribution described above. The second component instead is extracted from a normal distribution with mean 0, modeling the fragment-specific size variability.

[3] Location parameter: 0; Scale parameter: 0.25

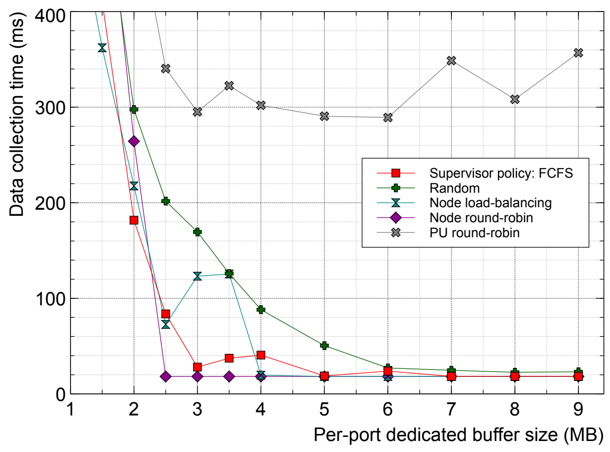
The results are presented in Figure 12. As expected, the incast onset point is reached with fewer available credits than with fixed-size fragments: the traffic-shaping algorithm cannot prevent packet drops as effectively as with fixed-size fragments. Indeed, with fixed-size fragments, the maximum burst size is fixed to the product of the available traffic-shaping credits and the fragment size. With variable fragment sizes, instead, the maximum burst size varies as well, possibly exceeding that value and leading to packet drops. However, for both switch models, an optimal operating region can still be reached. In particular, if the chosen event assignment policy effectively spreads the events on all the available nodes, like in the case of the node load-balancing and round-robin policies, the minimum data-collection latency value is identical to the value that can be reached with fixed-size fragments.

7.3 Buffer space

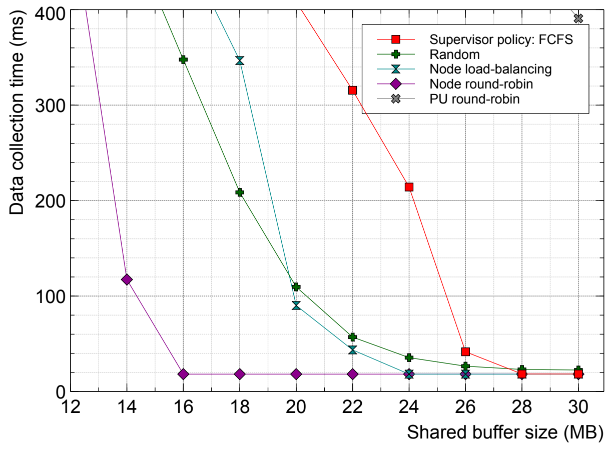
The simulation also enables studying the effects of hardware parameters that cannot so easily be modified in practice. One of these is the size of the packet buffers in the top-of-rack switches. The simulation can be used to determine the relation between the amount of memory and the data-collection latency, with the workload described in Section 4.1 and with the traffic-shaping algorithm disabled.

The results for a switch with dedicated per-port memory are shown in Figure 13a. When employing the FCFS assignment policy, buffers of at least 5 MB effectively prevent packet drops and the latency can reach its lowest value (slightly lower than 20 ms, compatible with the minimum latency estimated in Section 4.2). However, with the random assignment policy, the lowest latency is only reached with 8 MB buffers. This discrepancy, and in general the better performance of FCFS in this scenario, can be explained. If two events are consecutively assigned to two processing units on the same worker node, the data for the second event incurs a larger delay, since the buffer of the worker node's switch port still contains data from the first event, leading to queuing delays or overflow. This effect changes the order of the supervisor's FCFS queue: over time, entries in the queue referring to different units on the same worker node distance themselves. The available switch buffer memory is therefore used more efficiently, and less of it is necessary to prevent packet drops. As expected, the node load-balancing and round-robin algorithms outperform both FCFS and random, with node round-robin reaching the lowest latency with just 2.5 MB buffers, i.e. buffers that can contain a single full event.

The results for a switch with shared memory are shown in Figure 13b. In this scenario, both with the FCFS and random assignment policies, packet drops are prevented with a buffer size of at least 28 MB. However, for smaller buffer sizes, the random assignment policy performs better than FCFS. This is explained by the fact that, in a switch with fully shared memory, packet drops are not directly related to the occupancy of their output queue. Therefore, the beneficial effect described in the above paragraph does not apply, and the random assignment policy ensures a more uniform usage of the switch outputs, thus reducing packet drops. This interpretation is further reinforced by the results of the node load-balancing and round-robin poli-



(a)



(b)

Fig. 13 Average data-collection latency (with no traffic-shaping) as a function of the top-of-rack switch buffer size, using (a) dedicated per-port buffers and (b) shared buffers.

cies, which enable reaching the minimum latency with buffers of just 24 and 16 MB respectively.

8 Related work

Characteristics of other large-scale data-acquisition systems can be found for example in [8] for the LHCb experiment and [12] for the ALICE experiment. Details of the CMS experiment's InfiniBand-based system are in [11].

Alternative general-purpose network simulation frameworks include the open-source ns-3 [6] and the commercial SteelCentral NetModeler (formerly OPNET Modeler). Ns-3 includes the Direct Code Execution (DCE) framework, which enables the execution, within the simulation, of existing implementations of network protocols or applications without source code changes. Using this framework to execute the Linux TCP implementation could improve the accuracy of the simulation in reproducing TCP behavior.

The TCP incast pathology received a considerable amount of attention in the academic community. See [25] for a review. The application of some of the proposed solutions to a small prototype of the ATLAS data-acquisition network is reported on in [16].

9 Conclusion and future directions

The simulation model presented in this paper was proven capable of reproducing the key performance traits of a complex data-acquisition system such as the ATLAS TDAQ system. The approach employed in the development of the model, aiming at keeping complexity at a minimum without sacrificing accuracy, can be extended to systems with similar traffic patterns. Indeed, while the ATLAS experiment was used as a case study throughout this paper and the numeric results presented here are specific to it, the issues discussed are common to data-acquisition systems in general. A system-specific model such as the one described in this paper can provide useful insights without the need of intrusive system modifications or testing campaigns.

Once trust in the simulation model is established, a wide variety of what-if scenarios can be easily implemented and explored. The examples described in this paper show that the results can prove extremely valuable in driving the future evolution of the real system. In the case of the ATLAS data-acquisition, the simulation results suggest that smarter traffic scheduling can result in significant performance improvements. Their implementation is currently hindered by practical limitations (not enough CPU power), but given the promising simulation results an effort to overcome these limitations is worthwhile. In other simulation runs, the impact of the network hardware buffer depth was quantified. The results are quite clear and can be used to guide future hardware purchase requirements.

Using this work as a base, more invasive solutions can be tested. In particular, three categories of solutions to the incast pathology can be modeled: more sophisticated application-level traffic shaping, alterations of the transport protocol itself (requiring kernel modifications in the real system) and of the link layer (requiring hardware changes). The second category includes solutions such as reducing TCP's minimum RTO limit or experimental incast-aware TCP variants. The third category mainly refers to alternative link-layer technologies such as Infiniband or recent additions to the Ethernet standards such as IEEE 802.1Q Congestion Notification. A thorough evaluation of these solutions will yield results that can guide the design of future data-acquisition systems.

References

1. Brocade MLX series. URL `http://www.brocade.com/en/products-services/routers/mlx-series.html`

2. Brocade VDX 6740 switches. URL `http://www.brocade.com/en/products-services/switches/data-center-switches/vdx-6740-switches.html`
3. Castalia wireless sensor network simulator. URL `https://castalia.forge.nicta.com.au`
4. HP ProCurve 6600 manuals. URL `http://www.hp.com/rnd/support/manuals/6600dc.htm`
5. INET framework open-source OMNeT++ model suite for wired, wireless, and mobile networks. URL `https://inet.omnetpp.org`
6. The ns-3 network simulator. URL `http://www.nsnam.org`
7. ATLAS high-level trigger, data-acquisition and controls. Technical Design Report ATLAS-TDR-016 CERN-LHCC-2003-022, CERN, Geneva (2003)
8. Alessio, F., et al.: The LHCb Data Acquisition during LHC Run 1. J. Phys.: Conf. Ser. **513**(1), 012,033 (2014). DOI 10.1088/1742-6596/513/1/012033
9. Allman, M., Paxson, V., Blanton, E.: TCP congestion control. RFC 5681 (2009)
10. ATLAS Collaboration: The ATLAS experiment at the CERN large hadron collider. J. Instrumentation **3**(08), S08,003 (2008). DOI 10.1088/1748-0221/3/08/S08003
11. Bawej, T., et al.: Boosting Event Building Performance using Infiniband FDR for the CMS Upgrade. Proc. Sci. **TIPP2014**, 190 (2014)
12. Carena, F., et al.: The ALICE data acquisition system. Nucl. Instruments and Methods in Physics Research A **741**, 130–162 (2014). DOI 10.1016/j.nima.2013.12.015
13. Colombo, T.: Data-flow Performance Optimisation on Unreliable Networks: the ATLAS Data-Acquisition Case. J. Phys.: Conf. Ser. **608**(1), 012,005 (2015). DOI 10.1088/1742-6596/608/1/012005
14. Ha, S., Rhee, I., Xu, L.: CUBIC: A new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. **42**(5), 64–74 (2008). DOI 10.1145/1400097.1400105
15. Henderson, T., Floyd, S., Gurtov, A., Nishida, Y.: The NewReno Modification to TCP's Fast Recovery Algorithm (2012)
16. Jereczek, G., Lehmann-Miotto, G., Malone, D.: Analogues between tuning TCP for data acquisition and datacenter networks. In: IEEE Int. Conf. Comm. (2015)
17. Köpke, A., Swigulski, M., Wessel, K., Willkomm, D., Haneveld, P.T.K., Parker, T.E.V., Visser, O.W., Lichte, H.S., Valentin, S.: Simulating wireless and mobile networks in OMNeT++ the MiXiM vision. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Simutools 2008, p. 71. ICST, Brussels (2008). DOI 10.4108/ICST.SIMUTOOLS2008.3031
18. Núñez, A., Fernández, J., Garcia, J.D., Prada, L., Carretero, J.: SIMCAN: A simulator framework for computer architectures and storage networks. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Simutools 2008, p. 73. ICST, Brussels (2008). DOI 10.4108/ICST.SIMUTOOLS2008.3025
19. Paxson, V., Allman, M., Chu, J., Sargent, M.: Computing TCP's retransmission timer. RFC 6298 (2011)
20. Phanishayee, A., et al.: Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In: Proc. of the 6th USENIX Conference on File and Storage Technologies, FAST'08, pp. 12:1–12:14. USENIX Association, Berkeley (2008)
21. Pozo Astigarraga, M.E.: Evolution of the ATLAS Trigger and Data Acquisition System. J. Phys.: Conf. Ser. **608**(1), 012,006 (2015). DOI 10.1088/1742-6596/608/1/012006
22. Reschka, T., Dreibholz, T., Becke, M., Pulinthanath, J., Rathgeb, E.P.: Enhancement of the TCP module in the OMNeT++/INET framework. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Simutools 2010, p. 24. ICST, Brussels (2008). DOI 10.4108/ICST.SIMUTOOLS2010.8834
23. Vargas, A.: OMNeT++. In: K. Wehrle, J. Gross, M. Günes (eds.) Modeling and Tools for Network Simulation, pp. 35–59. Springer-Verlag, Berlin, Heidelberg (2010). DOI 10.1007/978-3-642-12331-3
24. Yebenes, P., Escudero-Sahuquillo, J., Garcia, P.J., Quiles, F.J.: Towards modeling interconnection networks of exascale systems with OMNet++. In: Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP '13, pp. 203–207. IEEE Computer Society, Washington (2013). DOI 10.1109/PDP.2013.36
25. Zhang, Y., Ansari, N.: On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers. IEEE Commun. Survey Tutorials **15**(1), 39–64 (2013). DOI 10.1109/SURV.2011.122211.00017