Computer Modeling the ATLAS Trigger/DAQ System Performance

Robert Cranfield, Piotr Golonka, Anna Kaczmarska, Krzysztof Korcyl, Jos Vermeulen, and Sarah Wheeler

Abstract—In this paper, simulation (“computer modeling”) of the Trigger/data acquisition (DAQ) system of the ATLAS experiment at the LHC accelerator is discussed. The system will consist of a few thousand end nodes, which are interconnected by a large Ethernet network. The nodes will run various applications under the Linux operating system (OS). Predictions for the latency, throughput and queue development in various places have been obtained. Results are presented with respect to the application of traffic shaping to reduce the probability of possible frame loss (which may cause severe performance degradation).

Index Terms—Data acquisition, discrete-event simulation, local area network, modeling, queuing analysis, traffic control.

I. INTRODUCTION

The trigger/data acquisition (DAQ) system for the ATLAS detector at the CERN LHC will be based on a large local area network (LAN). More than 1600 Readout Buffers (ROBs) transmit detector data on request via the network some tens or hundreds of level 2 processing units (L2PUs) and of the order of 100 event filter (EF) subfarm interfaces (SFIs). The SFIs build complete events and pass them on request to one of several thousand EF farm processors. Commodity Ethernet has been chosen for the network because of its cheapness, widespread usage and availability over a range of transmission speeds and physical media. The L2PUs and SFIs are commodity PCs running the “DataCollection software,” and, in the case of the L2PUs, the Level 2 (LVL2) trigger software, under the Linux OS. An example of the network architecture for the Trigger/DAQ system, which we used in our modeling studies, is presented in Fig. 1.

On the basis of information provided by the Level 1 Trigger, part of the detector data from events accepted by the Level 1 Trigger and stored in the ROBs are fetched by the SFIs from the ROBs to the L2PUs and EFs is at maximum about an order of magnitude). The expected total rate of data flowing from the ROBs to the L2PUs and EFs is at maximum about 5 GB/s.

II. MODELING

Two computer modeling tools are used: at2sim [2] based on the Ptolemy framework [3] and simdaq—a dedicated C++ program [4], [5].

The model of the Trigger/DAQ system implemented in both tools is an object-oriented model, in which most objects represent hardware (e.g., switches, computer links, processing nodes), software (e.g., low level network communications in Linux OS, data collection applications), or data items (e.g., Ethernet packets).

The type of simulation used for the computer models is known as “discrete event simulation.” Basically, the simulation program maintains a time-ordered list of “events,” i.e., points in time at which the simulated system changes state in a way implied by the type of “event” occurring. Only at the time of occurrence of an event is the modeled system allowed to change its state; in most cases only a small part of the state of the simulated system needs to be updated. The state change can result in the generation of new events for a later time; these events are then entered at the correct position in the event list. The simulation program executes a loop in which the earliest event is fetched from the event list and then handled.

Because of the size of the network it is not possible to build and test a full size prototype prior to constructing the final system. Apart from checking averages to be equal to “paper model” results there is therefore no way to check the computer model results other than by comparison of results from both tools. “Paper model” results are results from straightforward calculations of average message frequencies, bandwidth requirements and CPU capacity requirements using first level trigger rates and details of the trigger processing and mapping of the detector onto the ROBs.

At the time of submission of this paper the models implemented with both tools do not yet allow comparison of the results obtained for the full system, but the results do complement each other. The focus of the at2sim program is currently a proper simulation of the event building, while in simdaq details more relevant to the LVL2 trigger are taken into account (in particular proper handling of the step-wise execution of the trigger algorithms and associated requesting of event data from the ROBs).
A. Parameterization of System Components

The models of the system components are kept as simple as possible, but are sufficiently detailed to reproduce behavioral aspects relevant to the issues studied. Each model has measurable parameters.

We have developed a parameterized model describing the behavior of the members of a class of typical Ethernet switches. This type of switch is used in ATLAS Trigger/DAQ test setups and could be used in the final system. The switches operate in the store-and-forward mode and have a modular architecture: modules contain interconnected groups of ports and provide intramodule transfers while the intermodule communication proceeds via a backplane. The model has ten parameters such as the amount of input and output buffering, transfer limits when moving packets to and from the backplane for intermodule transfers and transfer speed for inter and intramodule transfers. These parameters have been identified to determine transfer latency and bandwidth limitations in case of congestion. The switch model supports flow-control and provides statistics on the usage of flow-control. It also offers very detailed statistics with respect to queue development in the output ports of the switch. This has proven to be very useful for quantifying the effect of various traffic shaping schemes.

The models of all components other than switches are built around a parameterized model of a multitasking OS with interrupt-driven network communication. The behavior of the Linux OS running multiple threads on a single-processor machine has been successfully modeled (Fig. 2).

The details of recent improvements of the Linux networking subsystem (NAPI, interrupt coalescence, flow control) are also taken into account [6]. The CPU time consumption due to communication, with multiple-level processing of the incoming messages, can be reliably estimated (hardware and software protocol stack interrupts and specific overheads of the high-level data-formatting routines are properly modeled).

The ATLAS Trigger/DAQ system applications are built around a common software framework, the Data Collection Software, executed on multiprocessor PC computers running the Linux OS. The applications have one or multiple execution threads. Their high-level message-passing subsystem, responsible for proper message formatting, packetization and buffer management, may either be steered sequentially or executed as a dedicated Input Thread. Both models of message-passing approaches are properly modeled. The specific functionality of the data-collection and of the higher-level trigger applications is concentrated in models of “tasks” being run by the model of the OS. The tasks model the activities and state changes due to incoming messages or computations.

In the L2PU (see Fig. 3) the input thread is responsible for receiving messages from the network. The messages are passed to the message dispatcher. Various handlers may subscribe to the Dispatcher and they will be notified when the requested type of message arrives. The LVL1 result handler passes the LVL1 result message to the LVL1 Result Queue. When a Worker Thread finishes processing an event and becomes free, it checks the status of the LVL1 result queue and fetches the next event from the head of the queue.
The parameters used for parameterization of the OS and applications are described in [7].

B. Model Calibration

All parameterized models of system components are calibrated using results from dedicated, small setups.

To calibrate the switches we used Ethernet frame generators based on FPGA devices or on programmable NICs (which support user modification of the firmware). We developed procedures to measure the values of all ten parameters of the switch model.

The values of the parameters of the low-level communication models have been determined with simplified setups where maximum achievable message rates were measured. We have observed that the data frames in a simplistic streaming scenario start to be lost at a certain rate, due to saturation of the available CPU capacity. The inverse of this rate provides an estimate of the average CPU time needed to process a single incoming data frame. The model correctly predicts this quantity, taking interrupt coalescence and multiple stages of processing of the message into account (see Fig. 4).

The Trigger/DAQ application parameter values were obtained by inserting time stamps into the application’s code. The log files with time stamps were later analyzed to find the time intervals between the starts of different but successive actions of the application. Wherever possible, a small setup was used, consisting of only two machines connected back to back with the calibrated application running on one machine and a tester application, sending messages in a predefined order, on the other. The times calculated from the time stamps were used to predict the maximum rate the application can sustain and were
cross-checked with results of maximum rate measurements. Various parameters were studied with respect to a possible impact on the performance of the application in question. For example, the plot on the left side of Fig. 5 shows that the DFM model is also sensitive to the number of LVL2 accepts inside the group of events received from the LVL2 Supervisor.

C. Model Validation

The biggest challenge for modeling is to predict the scalability of the final system. Correct modeling of testbeds of various sizes increases our confidence in the models used.

The plot in Fig. 6 shows good agreement between model predictions and measurement results obtained with a test setup aimed at testing the scalability of the event builder part of the system. The maximum event building rate scales linearly with the number of SFIs.

The two lines in the plot represent different system configurations. In one configuration, each of the ROBs is connected directly to the network, giving rise to 1600 access points. In the other configuration groups of eight ROBs are formed, with one network interface per group, so that the number of access points is reduced to 200. The data from a single event can be requested with a single message per group, the response consists of event fragments retrieved from all eight ROBs aggregated in a single message. The smaller number of access points requires a smaller number of request messages. This in turn results in a smaller fraction of the CPU capacity spent on generation of requests and increases the total number of events that can be processed per second. The number of responses received is also smaller than for direct connection of the ROBs to the network, however, the CPU time spent in receiving the event data depends more strongly on the number of frames than on the number of messages received. For the results shown in Fig. 6 the total number of frames per event does not depend on the system configuration.

III. FULL SIZE MODEL RESULTS

Results for the full size system were produced. Various ideas on traffic shaping aimed at improvement of performance and at avoidance of performance degradation due to frame losses were evaluated.
A. Tests of a Credit-Based Traffic Shaping Strategy

The number of frames that can be stored in buffers in switches rather than the number of bytes is bound to a maximum. Avoiding buffer overflow is essential to prevent frame loss. This is important, as the data from lost frames have to be retransmitted, which can result in a large amount of retransfers, which, in turn, may cause the whole system to jam. By limiting the number of outstanding requests, the SFIs can control the buildup of reply queues in the central switch. In Fig. 7 the maximum queue length in the central EB switch is shown as a function of the number of “credits” (maximum number of outstanding requests) for different scenarios for reading out the detector data. Collecting data from buffers which aggregate detector data from more than one readout link results in a proportional increase in the number of frames waiting in the switch for port availability, since one frame is sent per readout link. The event building rate is identical for all data points corresponding to more than five credits per SFI—the rate is limited by the time needed to transfer the fragments from all buffers (∼2 MB) over the Gigabit Ethernet connection between the EB central switch and an SFI.

B. The Impact of the L2PU Assignment Strategy

Results for the second-level trigger decision times for a model of the full system, running at the LHC design luminosity, and for a first-level trigger rate of 75 kHz, have been obtained.

The model is based on a realistic trigger menu; the mapping of the detector and relevant details of the various steps of the second-level trigger algorithms have been taken into account. Groups of 12 ROBs are assumed to be connected via Gigabit Ethernet links, one per group, to the LVL2 central switch. Data from the same event are sent as a single message which may consist of more than one frame if involving several ROBs. Groups of 5 LVL2 trigger processors (dual-CPU 4 GHz PCs) are connected via small Gigabit Ethernet switches to the same central switch.

Each LVL2 trigger processor can run at maximum four threads, with an event assigned to each thread. The algorithm execution times and acceptance factors of the various steps have been obtained by extrapolating results from algorithm benchmarks. The values of other parameters of the model are estimates for 4 GHz PCs. The Ethernet switches are assumed to be nonblocking. Distributions for the LVL2 decision time for an average L2PU utilization of about 80% (100 L2PUs) are presented in Fig. 8.

The distribution with the long tail arises from round-robin (RR) assignment of events to the L2PUs. The two additional distributions for least queued assignment are for limiting the number of events handled at the same time to a maximum of four for each L2PU (lq-4) and for a combination of this with limiting the number of outstanding requests per L2PU to a maximum of 4 (lq-4-4).

Fig. 7. Central EB switch queue buildup for various scenarios for reading out the detector data.

Fig. 8. Distributions for the LVL2 latency for RR and least queued (lq) assignment of events to the L2PUs. The two additional distributions for least queued assignment are for limiting the number of events handled at the same time to a maximum of four for each L2PU (lq-4) and for a combination of this with limiting the number of outstanding requests per L2PU to a maximum of 4 (lq-4-4).
events at the time of assignment. This assignment scheme is referred to as \textit{least queued} assignment. The long tail arising from RR assignment can also be suppressed by only allowing up to a certain maximum number of events to be handled by each L2PU at the same time. In that case, LVL1 accepts need to be stored by the supervisor if no L2PU is available for assignment and the event can only be assigned once an L2PU is available (signaled by the reception of a LVL2 trigger decision). The effect of this is small for least-queued assignment, as can be seen from the figure. The figure also shows that limiting the number of outstanding data requests to a maximum of four per L2PU only has a small effect on the decision time distribution. It has to be noted that, in the model, I/O and request formulation are handled at a higher priority than trigger algorithm execution. The peaks in the distributions arise from the different steps in the trigger algorithms (for each step a fixed computation time is assumed, though in reality these computation times may vary from event to event). In Fig. 9 distributions are shown for the number of events assigned by the LVL2 supervisor to a L2PU at the time that a new request for assignment is made. From the distribution for least-queued assignment without further constraints it can already be seen that it is not necessary to assign more than four events to the same L2PU. If the number of outstanding requests per L2PU is limited to a maximum of four a shift of the peak of
the distribution toward a higher number of events assigned can be observed. This shift shows that on average the L2PU needs somewhat more time to reach a decision: the average decision time is 3.8 ms instead of 3.2 ms for the least queued scenario without further constraints.

In Fig. 10, the sizes of the queues for the ports of the central Ethernet LVL2 switch connecting to the small switches as found with the model are presented. Each message flowing through these ports incremented the queue size upon arrival of the last frame of the message. Only by limiting the number of outstanding requests per L2PU can a reduction of the queue sizes be achieved. For a maximum of four outstanding requests at maximum, somewhat more than 30 frames are queued. This is more than the number of five times four one would naively expect, as the data sources may send multiframe messages and also as requests from the LVL2 supervisor to the L2PUs are transferred via the same ports.

IV. CONCLUSION

The behavior of the calibrated component models is in good agreement with the behavior of the real components in small test setups. The first experimental results for event building in a larger test setup also show an encouraging agreement with the model predictions. However, further validation is required. Models for the full-scale system already allow determination of possible problem areas and investigation of techniques for preventing buildup of queues in switches and processors. In the models, buildup of queues may result in long second-level trigger decision times or event-building times; message loss due to queue overflow may in reality also occur. Models have been run for the full system, for event building based on the calibrated component models and for the LVL2 trigger so far based on “paper model” assumptions.

It has been shown that credit-based pull scenarios for event building and for collection of input data for the second-level trigger are essential for limiting queue lengths and most likely can be applied without compromising the throughput. It has also been shown that long LVL2 trigger decision times can be avoided in cases of a high average L2PU processor utilization by the use of the least queued assignment strategy and by limiting the number of events simultaneously handled by each L2PU.

REFERENCES