Performance measurement of ATLAS Data Collection software with QoS

Yoshiji Yasu, Yoji Hasegawa, Yasushi Nagasaka and Makoto Shimojima
On behalf of the ATLAS TDAQ DataFlow community[1]

***Abstract*— IP-based Quality of Service (QoS) in Linux has been applied to ATLAS Data Collection(DC) software with Gigabit Ethernet and then the performance has been measured. The measurement showed that the QoS improved the event building performance. The results are described.**

I. Introduction

CONGESTION avoidance of event data flow in ATLAS event builder network[2] is crucial[3]. Traffic management of the data flow is an essential point to avoid the congestion. Therefore, adopting congestion avoidance and flow control techniques for the event builder using switching network technologies are major issues.

On one hand, Gigabit Ethernet is one of the technologies which enables a high speed transfer for the event builder[4] and a major candidate of ATLAS event builder network. Ethernet provides a best-effort service to all of their applications, with few traffic shaping method in comparison with ATM. IP QoS technique, which is a control scheme at the level of event fragment such as traffic shaping for a packet-oriented network, had been investigated[5], [6], [7].

On the other hand, there are two different scenarios of data flow for an event builder, which is distinguished by whether the event manager informs the assignment to the sources or the destination. A push scenario is that the sources are responsible for initiating the data transfer, while a pull scenario is that the destination initiates the transfer by requesting the sources to send the data.

Both of the push and pull scenarios are implemented into the ATLAS event builder and selectable at execution for investigating the feasibility of the event builder architecture[8].

II. Configuration for Measurements on ATLAS Event Builder

A. Hardware configuration

The measurement used PCs connected to one BATM Gigabit Ethernet switches located at Building 513 testbed at CERN. The typical specification of PCs is following:

- CPU : Dual Xeon 2.2GHz/2.4GHz
- Memory : DDR 200/266-SDRAM 1GB
- OS : CERN RedHat 7.3.1 (Kernel 2.4.18-27) with CapServer patch
- NIC : Intel Pro 1000, driver e1000
- GbE switch : BATM
- kernel buffer size = 8MB
- kernel scheduling time : HZ parameter = 4096

Y.Yasu are with High Energy Accelerator Research Organization(KEK), Tsukuba, Japan (e-mail: Yoshiji.YASU@kek.jp)
Y.Hasegawa is with Shinshu University, Matsumoto, Japan
Y.Nagasaka is with Hiroshima Institute of Technology, Hiroshima, Japan
M.Shimojima is with Nagasaki Institute of Applied Science, Nagasaki, Japan

B. DataCollection Software Configuration

The DataCollection (DC) software[9] of the release DC-00-02-02 and the online software[10] of the release online-00-17-02 were used for this measurement.

The DC software parameters relevant to the measurement are following:

1) ROS parameters
   - Number of ROSs : variable
   - RoB data size (robDataSize) : variable
2) SFI parameters
   - Number of SFIs : variable
   - MaxNumberOfAssembledEventsHigh : 50
   - MaxNumberOfAssembledEventsLow : 45
   - DefaultCredits : 10
   - TimeoutSetup : 500
   - TrafficShapingParameter : 30
3) DFM parameters
   - Credits : 3
   - Timeout : 5000
   - MaxAssignqueueSize : 1000
   - ClearGroupSize : 50
   - DecisionGroupSize : 10
4) LVL2 decision : one LVL2 accept in one decision
5) Message passing protocol: UDP/IP

Each of ROSs, SFIs and DFM applications ran on a host exclusively. Root and DC controllers ran on a host. DFMautotester7 which generates L2accepts ran on the same host on which L2SV application was running. The L2accept means good event decided by L2 trigger.

III. Measurements without QoS

The measurements of event building performance were done for the push and pull scenarios with the same setup. The results for both scenarios are compared.

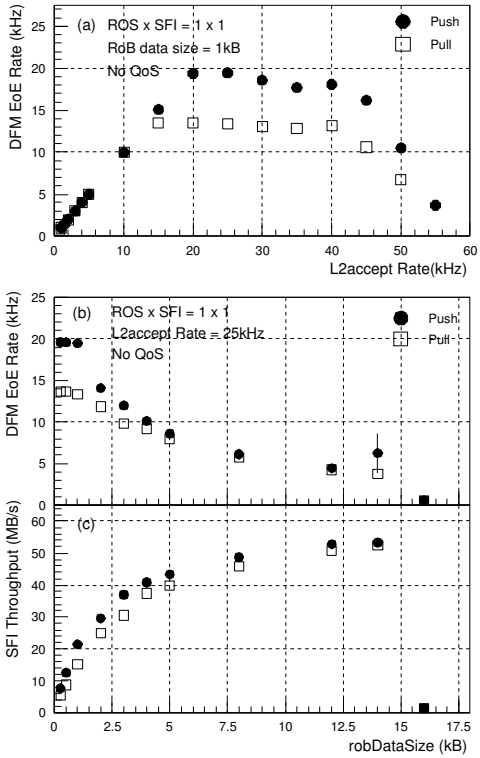

Fig. 1. End of Event rate as a function of (a) L2accept rate and (b) robDataSize and (c) throughput as a function of robDataSize are shown for the push and pull scenarios at 1 × 1 configuration.

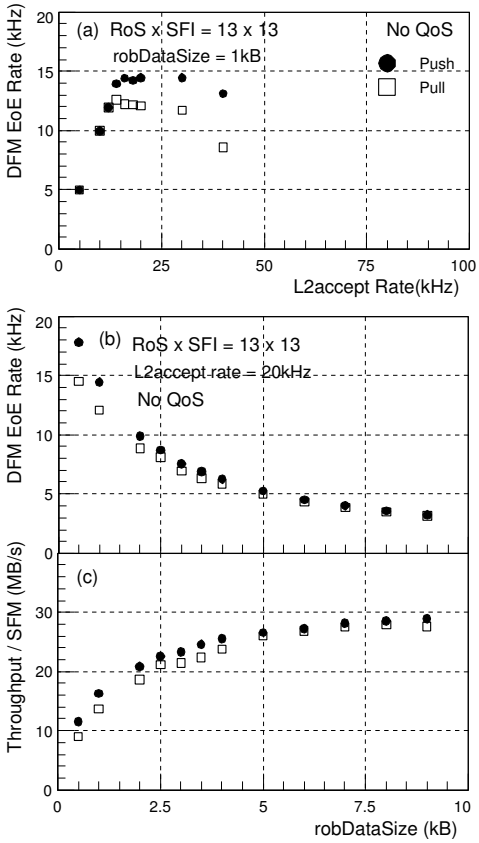

Fig. 2. End of Event rate as a function of (a) L2accept rate and (b) robDataSize and (c) throughput as a function of robDataSize are shown for the push and pull scenarios at 13 × 13 configuration. The measurements were performed at ICEPP, University of Tokyo. Thirty two PCs with Pentium III 1.4GHz and 512MB memeory and a GbE switch of BlackDiamond 6816 with 4G8Gi modules.

A. Results in Configuration without Packet Loss

Figure 1 shows the results in 1 ROS × 1 SFI configuration. Figure 1 (a) shows the rate of End of Event (EoE), which is received by the DFM, as a function of L2accept rate generated by DFMautotester7 with fixed robDataSize of 1 kB. The EoE means the event is successfully built. The robDataSize means the size of event fragment, which is sent from each ROS to a SFI. For the push scenario, the EoE rate increases to ∼20kHz according to increase of L2accept rate, then the rate is saturated for higher L2accept rate. For the pull scenario, the behavior is the same as that for the push scenario, but the maximum EoE rate is 14 kHz.

Figure 1 (b) and (c) show EoE rate and throughput at the SFI as a function of robDataSize with fixed L2accept rate of 25 kHz. As the robDataSize increase, the EoE rate decreases. Throughput increases up to 52 MB/s for robDataSize of 14 kB, then rapid drop of EoE rate and throughput was observed for larger robDataSize.

Generally, the pull scenario is thought to be better than the push scenario from the point of view of congestion avoidance of the network because the destination (SFI) initiates the data transfer from the sources (ROS) and we could easily make it serial. However, since an extra message is involved for each request in the pull scenario, this incurs some performance penalty. In other words, the push scenario has a potential to outperform the pull scenario if we could get rid of traffic congestion.

ATLAS event builder also implements an algorithm to avoid the coherent data flow to SFI. As an example, Figure 2 shows the results at 13 ROSs × 13 SFIs configuration. The behaviors of the EoE rate and the throughput at one SFI are similar to those in 1 ROS × 1 SFI configuration, though the absolute values are different. Thus, the data flow to SFI was not concentrated in 13 ROSs × 13 SFIs configuration.

B. Results in Configuration with Packet Loss

Figure 3 (a) shows EoE rate as a function of L2accept rate. The robDataSize was fixed to 1 kB.

The EoE rate is saturated at 5 kHz for the push scenario and 4 kHz for the pull scenario. This is due to high CPU utilization of the SFI application.

Figure 3 (b) and (c) show EoE rate and throughput at the SFI as a function of robDataSize, respectively. L2accept rate was fixed to 5 kHz at which the event building rates reached to their plateau for both of the push and pull scenarios.

According to increase of robDataSize, EoE rate gradually decreases, on the other hand throughput at the SFI increases. Throughput of the pull scenario can reach ∼90 MB/s for robDataSize of 7 kB and above. The congestion caused by concentration in the pull scenario was not observed in 6 ROSs × 1 SFI configuration, because the SFI reads data from sources sequentially.

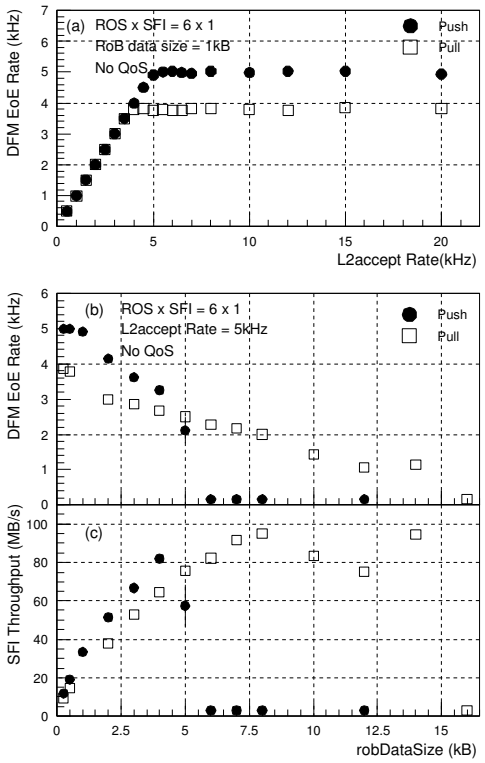

Fig. 3. End of Event rate as a function of (a) L2accept rate and (b) robDataSize and (c) throughput as a function of robDataSize are shown for the push and pull scenarios.

For the push scenario, throughput increases to 80 MB/s according to increase of robDataSize. However, when robDataSize is larger than 4 kB, throughput drops rapidly and almost no traffic was observed for robDataSize larger than 6 kB.

At this situation a lot of re-asks that an SFI requests ROSs to send packets again were observed, so that events can not be built at all. This implies that packet loss occurred in the push scenario is due to congestion or buffer overflow at the SFI.

IV. Measurements with QoS in Push scenario

Taking into account of the results shown in the preceding section, the traffic shaping technique without any modification of the event builder software was investigated. As the results, IP QoS based traffic shaping technique had been investigated as one of the control schemes at the level of event fragment. The IP QoS has been implemented in standard Linux kernel and can manage the data flow at the IP level of the network[11].

A. What is QoS

QoS stands for Quality of Service. In elements of QoS technology, packet classification, packet scheduling and traffic shaping techniques are important for the event builder. The packet classification is to classify packets in groups, such as Class Based Queueing (CBQ)[12]. The packet scheduler arranges the scheduling for outgoing packets according to the queueing method and the buffer management selected. Token

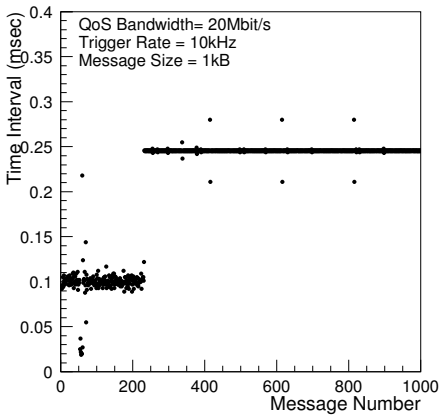

Fig. 4. Scheduling time interval between two messages sending sequentially as a function of message id number. Messages whose size is fixed to 1kB are generated in 10kHz. QoS assigned the bandwidth 20Mbit/s.

Bucket Filter (TBF) is one of the ways. The outgoing packet is sent at the rate determined by the size of the token buffer and the rate in which tokens are supplied. The traffic shaper is a technology to make the burst flat.

Linux standard kernel implements those QoS functionalities[11]. It is important to note that only the outgoing packets are controlled in the output queues. Incoming packets are accepted in a best effort basis.

It is also important to realize that packets are scheduled at most in the granularity allowed by the Linux kernel scheduler, which is governed by the parameter called HZ. The average input rate of Atlas event builder is $\sim$3 kHz. The data will be expected to come to the event builder at $\sim$3 kHz. The data should be scheduled at least over $\sim$3 kHz for the traffic shaping. However, the default value of the HZ parameter is 100 or 512 depending on the kernel version. The HZ parameters of 100 and 512 corresponds to the scheduling frequency of 100 and 512 Hz, respectively. This means Linux kernel with the default value can not shape the traffic of the incoming data to the event builder. Therefore, we made Linux kernel with the HZ parameter of 4096. The Linux kernel with the value can shape the traffic of event builder because the parameter of 4096 makes the Linux kernel scheduling frequency 4096 = 4 kHz.

Figure 4 shows interval of scheduling time of two sequential message sent from a host to another host as a function of message id number. Once the buffer for outgoing messages is fully filled, messages are sent out in the constant time interval of 0.25 msec. This means that the outgoing messages are scheduled at 4 kHz.

B. QoS Configuration

Figure 5 shows the QoS configuration of each ROS. The configuration defines a queueing discipline, CBQ as a root with handler of 1: and its daughter class 1:1 which is defined in the root. Bandwidth is set 1000Mbit/s to each of two classes. The class 1:1 has another scheduler using TBF. For this

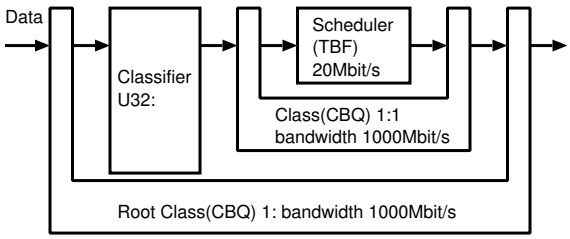

Fig. 5. QoS configuration used in this measurement. Because only one daughter class(1:1) is defined in the root class (1:), the classifier do nothing in this case. Bandwidth of this configuration is controlled by the scheduler with the TBF queueing discipline.

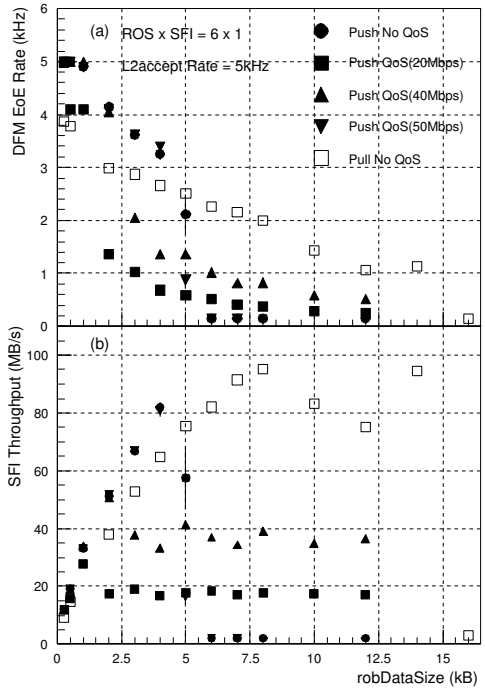

Fig. 6. (a) End of Event rate and (b) throughput as a function of robDataSize and QoS parameters are shown for the push scenario.

configuration, the network bandwidth is essentially controlled by the TBF scheduler.

C. Results

Figure 6 (a) and (b) show End of Event rate and throughput at SFI as a function of robDataSize in 6 ROSs $\times$ 1 configuration in case of the push scenario with bandwidth of 20Mbit/s, 40Mbit/s and 50Mbit/s assigned by QoS as well as the case of no QoS applied.

In case of bandwidth of 20 Mbit/s or 40 Mbit/s, the throughput became constant and was independent of robDataSize when the throughput reached to the value limited by QoS. In this region, re-asks decreased while there was a lot of re-asks observed in case of no QoS applied.

However when the bandwidth was assigned 50 Mbit/s, the traffic control did not work and the similar behavior was observed as that for no QoS applied.

The QoS technique applied to the ROS PCs enabled the improvement of the Atlas event builder shown in Figure 6 even if there is no global traffic control of the event builder data flow.

V. Conclusions

The performance of ATLAS event builder software was measured for case of the push scenario with QoS and the effect of QoS to the software were evaluated.

In the condition in which no packet loss occurs, the push scenario outperforms the pull scenario, since the pull scenario takes overhead due to sending extra control messages.

On the other hand, in the push scenario packet loss at the SFI occurs in the condition with larger message size at higher trigger rate and no QoS applied. As a result event building could not be done. However a suitable bandwidth applied to ROSs by QoS makes the event builder working even in the condition which events could not be built without QoS.

Now the conclusions are as follows;

1) The pull scenario is better than the push scenario even if the degradation of the performance in the pull scenario was a little bit observed, although large scale ATLAS event builder is so complicated and the results from a small scale test-bed can not be extrapolated easily.
2) IP QoS technique is a good traffic shaping one from viewpoint of the method without modification of the event builder software, but the shaping technique is not always effective on the event builder network. Thus, the QoS technique should be investigated toward the future ATLAS event builder.

Acknowledgment

We thank to Takahiko Kondo and Masaharu Nomachi for his support and encouragement. We also thank to Tetsuro Mashimo, Hiroshi Matsumoto, Stefan Stancu, Marian Zurek, Hanspeter Beck, Christian Haeberli and David Francis, for their support.

References

[1] The ATLAS TDAQ DataFlow community http://atlas.web.cern.ch/-Atlas/GROUPS/DAQTRIG/DataFlow/DFlowAuthors.pdf[1]
[2] ATLAS TDAQ: A Network-based Architecture, HP.Beck, R.W.Dobinson, K.Korcyl, M.LeVine, DC-59, February 2000.
[3] Atlas High-Level Triggers, DAQ and DCS; Technical Proposal. CERN/LHCC/2000-17, March 2000.
[4] Yoji Hasegawa, Yasushi Nagasaka, Yoshiji Yasu, DAQ/EF-1 Event Builder system and Linux/Gigabit Ethernet, ATL-DAQ-2000-008, March 2000
[5] Y.Yasu, Y.Nagasaka, A.Manabe, M.Nomachi, H.Fujii, Y.Watase, Y.Igarashi, E.Inoue, H.Kodama, Evaluation of Gigabit Ethernet with Quality of Service for event builder, the 11th IEEE Trans. on Nucl. Sci., Santa Fe, New Mexico, June 14-18, 1999
[6] Y.Yasu, Y.Nagasaka, Y.Hasegawa, A.Manabe, M.Nomachi, H.Fujii, Y.Watase, Quality of Service on Gigabit Ethernet for Event Builder, the 3rd International Data Acquisition Workshop on Networked Data Acquisition Systems, Lyon, France, October 20, 2000
[7] Y.Yasu, A.Manabe, Y.Nagasaka, Y.Hasegawa, M.Shimojima, M.Nomachi, H.Fujii and Y.Watase on behalf of the Atlas Trigger/DAQ group, Quality of Service on Linux for the Atlas TDAQ Event Building Network, International Conference on Computing in High Energy and Nuclear Physics CHEP2001, Beijing, September 3-7, 2001
[8] Message Flow: High-Level Description, HP.Beck, C.Haeberli, DC-12, June 2002
[9] ATLAS DataCollection home page, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/DataCollection.html

[10] ATLAS Online software home page, http://atlas-onlsw.web.cern.ch/Atlas-onlsw/
[11] Internet Protocol - Quality of Service, http://qos.ittc.ukans.edu/howto
[12] S.Floyd and V.Jacobson, Link-sharing and Resource Management Models for Packet Networks, IEEE/ACM Transactions on Networking, Vol.3 No.4, August 1995

[1] M. Abolins, A. Dos Anjos, M. Barisonzi, H.P. Beck, M. Beretta, R. Blair, J. Bogaerts, H. Boterenbrood, D. Botterill, M. Ciobotaru, E. Palencia Cortezon, R. Cranfield, G. Crone, J. Dawson, B. DiGirolamo, R. Dobinson, Y. Ermoline, M.L. Ferrer, D. Francis, S. Gadomski, S. Gameiro, P. Golonka, B. Gorini, B. Green, M. Gruwe, C. Haeberli, Y. Hasegawa, R. Hauser, C. Hinkelbein, R. Hughes-Jones, P. Jansweijer, M. Joos, A. Kaczmarska, G. Kieft, E. Knezo, K. Korcyl, A. Kugel, A. Lankford, G. Lehmann, M. LeVine, W. Liu, T. Maeno, M. Losada Maia, L. Mapelli, B. Martin, R. McLaren, C. Meirosu, A. Misiejuk, R. Mommsen, G. Mornacchi, M. Muller, Y. Nagasaka, K. Nakayoshi, I. Papadopoulos, V. Perez-Reale, J Petersen, P. de Matos Lopes Pinto, D. Prigent, J. Schlereth, M. Shimojima, R. Spiwoks, S. Stancu, J. Strong, L. Tremblet, J. Vermeulen, P. Werner, F. Wickens, Y. Yasu, M. Yu, H. Zobernig, M. Zurek