Performance measurement of DAQ-Middleware

Hiroshi Sendai, Kazuo Nakayoshi, Yoshiji Yasu and Eiji Inoue

Institute of Particle and Nuclear Studies,
High Energy Accelerator Research Organization (KEK)
1-1 Oho, Tsukuba, Ibaraki 305-0801 Japan

E-mail: `hiroshi.sendai@kek.jp`

Abstract. DAQ-Middleware is a software framework of network-distributed data acquisition system. DAQ-Middleware was developed based on Robot Technology Middleware (RT-Middleware), which is an international standard of Object Management Group (OMG) in Robotics. OpenRTM-aist is an implementation of RT-Middleware developed by the National Institute of Advanced Industrial Science and Technology. New implementation of DAQ-Middleware has been done according to the new OpenRTM-aist released early 2010 while DAQ-Middleware has been improved. Then, we measured the performance of the new DAQ-Middleware compared with previous one. We measured the throughput of DAQ-Middleware on several conditions. We observed improvement of performance in the new DAQ-Middleware.

1. DAQ-Middleware

DAQ-Middleware [1] is a software framework of a network-distributed DAQ system based on Robot Technology (RT) Middleware. RT-Middleware (RTM) is an international standard of the Object Management Group (OMG) not only for Robotics but also for embedded systems [2]. RT-Middleware provides the integration of RT systems as a Robotic Technology Component (RTC), which is the software unit. An RTC is a logical representation of a hardware and/or software entity that provides well-known functionality and services. OpenRTM-aist is a software package of RT-Middleware which was developed by the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan [3]. DAQ-Middleware was developed based on this package which implements in C++ and Python as programming languages and CORBA as communication protocol.

Why did we adopt RT-Middleware? Many DAQ systems so far could share the device driver and the library. Thus, they could make their own DAQ system using the common software. However, they could not share the software on the DAQ-Component level. Sometimes experimental groups could share a common DAQ system. We rather like to use not the level of the DAQ system, but the DAQ-Component because of its flexibility and reusability. Why could we not share the DAQ-Component? Because of there was no international standard of the DAQ-Component level so far. RT-Middleware is an international standard of DAQ-Component level. Thus, DAQ-Middleware is one of the solutions.

Figure 1 shows the architecture of the DAQ-Middleware. DAQ-Component is a software unit used to build an integrated DAQ system. DAQ-Operator is a special DAQ-Component which controls other DAQ-Components. A basic DAQ unit usually resides on a PC and consists of a set of components required to read and store data that are typically a Gatherer component,

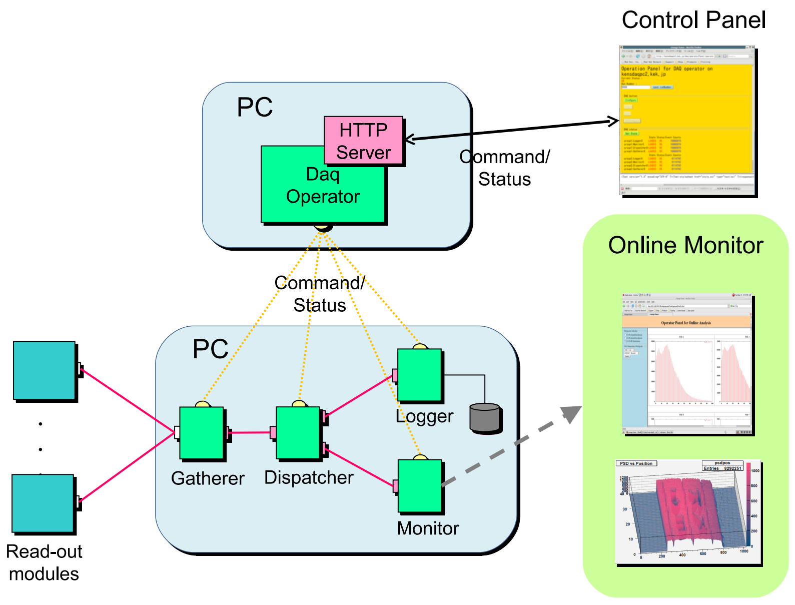

Figure 1. Systematic view of DAQ-Middleware

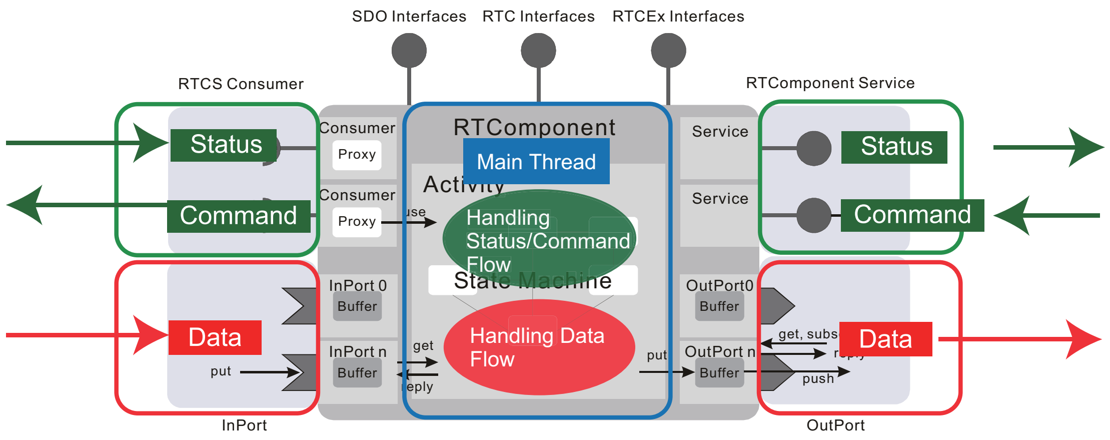

Figure 2. Features of DAQ-Middleware component

a Logger component, a Monitor component and a Dispatcher component. It is assumed that there are a DAQ operator component with system interface on a PC, and basic DAQ units on multiple PCs. A control panel of user interface on Web browser communicates with the DAQ operator. The DAQ operator reads the configuration file written in XML. After that, the DAQ operator can accept configure command from the control panel for system configuration, for an example. In same way, the control panel can issue begin and end commands to control the readout, the data logging and the online data analysis. The online histogram using ROOT [4] and online displaying on Web browser are shown in the figure.

RT-Component consists of several objects in Figure 2. DAQ-Middleware-Component is one of RT-Components. InPort and OutPort are data stream input and output, respectively. RT-Component Consumer and Service port are service interfaces for user command and reply to change internal attributes from outside. The DAQ-Component inherits the InPort,

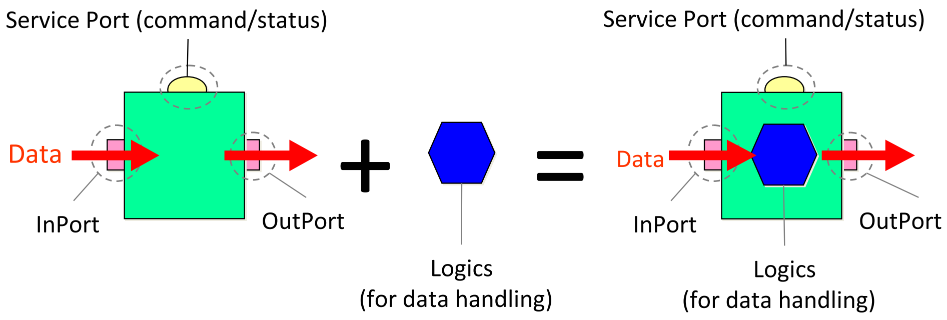

Figure 3. DAQ-Component

Table 1. Specification of test machines.

Model	Dell PowerEdge SC1430
CPU	Intel Xeon 5120 @ 1.86GHz 2 Cores $\times$ 2
Memory	2GB
Network	Intel Pro 1000 PCI/e (1GbE)
OS	Scientific Linux 5.4 (i386)

OutPort, Consumer and Service port of RTC. The InPort/OutPort are used for data path and Consumer/Service port are used for command/status path. The main thread is used for core logic with handling command/status flow and data flow. The thread does not work just in client/server model, but is autonomous with state machine. Users of DAQ-Middleware merely implement the component logics for data handling as their needs as described in Figure 3. DAQ-Middleware is used for data taking at the several beam lines in the Materials and Life Science Experimental Facility (MLF), Japan Proton Accelerator Research Complex (J-PARC). There are several gatherer components due to the difference of the read-out modules. There are several monitor components due to the difference of scientific work. But the other components are common [5, 6].

2. Performance measurement

Our aims of the measurements are to confirm the improvement of data transfer port on new DAQ-Middleware and to study the feasibility of DAQ-Middleware on a multi-core system. New implementation of DAQ-Middleware is based on OpenRTM-aist-1.0.0. So far, previous DAQ-Middleware is based on the special implementation of OpenRTM-aist-0.4.1.

We measured total throughput of data transfer on the new implementation and the previous one. The omniORB 4.0.7 implementation [7] was used as CORBA communication protocol.

DAQ-Middleware is used on not only multiple computers but also a multi-core system. Thus, we prepared two different communication channels, Gigabit Ethernet and loopback device.

Table 1 shows specification of PC used for the measurement. The configurations and the results will be shown in the following.

2.1. Setup with Ethernet and the result

Configuration 1: Run a source component and a sink component in the separate PC. The source component sends data to the sink one. We changed the one data chunk size from 1 kB to 1 MB. The sink component receives data from the source component and then

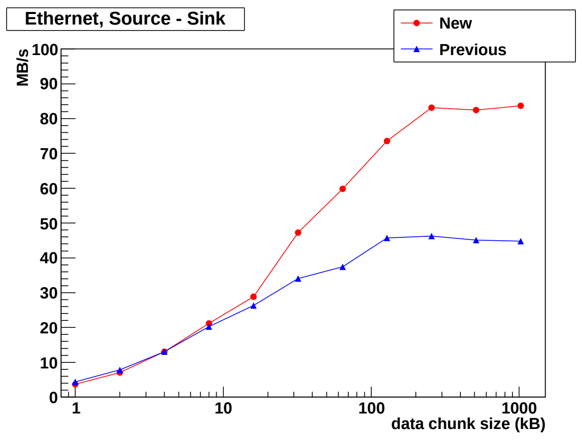

Figure 4. Result of Ethernet Source - Sink connection

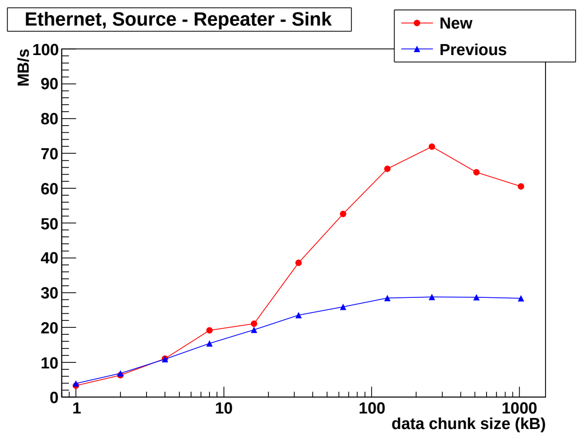

Figure 5. Result of Ethernet Source - Repeater - Sink connection

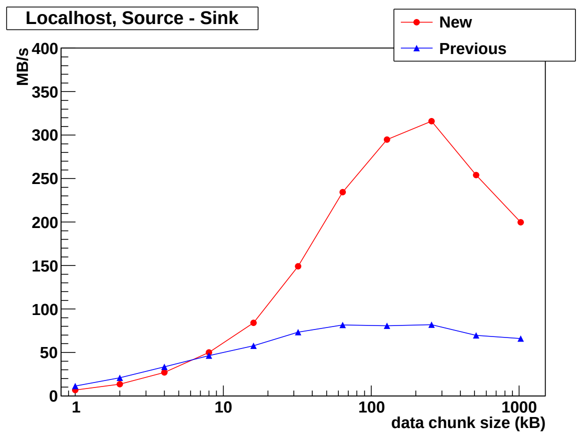

Figure 6. Result of Localhost Source - Sink connection

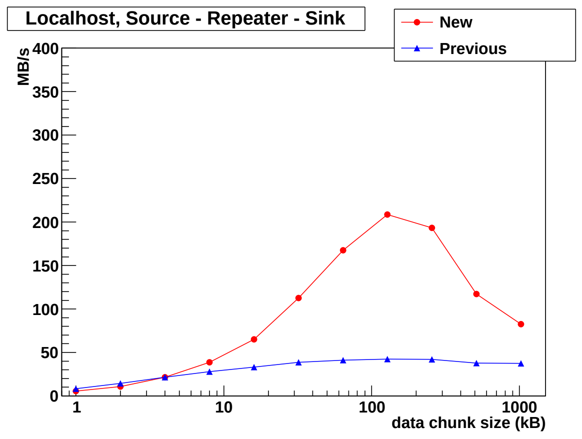

Figure 7. Result of localhost Source - Repeater - Sink connection

discards data without saving to disk to measure network total throughput. Ethernet frame size is 1500 bytes.

Configuration 2: A filter component that just passes data from one component to other, is necessary. It is called repeater. Put a repeater component between the Source component and the Sink one.

The result is shown Figure 4 and Figure 5. Blue solid line shows the performance using previous DAQ-Middleware and red solid line shows that using new DAQ-Middleware. Figure 4 and Figure 5 shows the total throughput on new DAQ-Middleware has been improved. There are at least two points of the reason why it has been improved. One is that a temporary buffering mechanism in flush mode on the OutPort of new OpenRTM-aist has been removed. This means that the number of data copy on the data transfer is reduced. Another is that the data type for the data transfer has been changed from CORBA any to CORBA oct. It reduced the serialization time of data.

Table 2. 8 Cores PC Specification

Model	HP xw8600
CPU	Intel Xeon 5420 @ 2.50 GHz 4 Cores × 2
Memory	8GB
Network	Intel Pro 1000 PCI/e (1GbE)
OS	Scientific Linux 5.4 (i386)

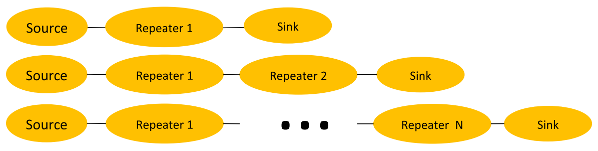

Figure 8. Many Repeaters on 8 cores CPU

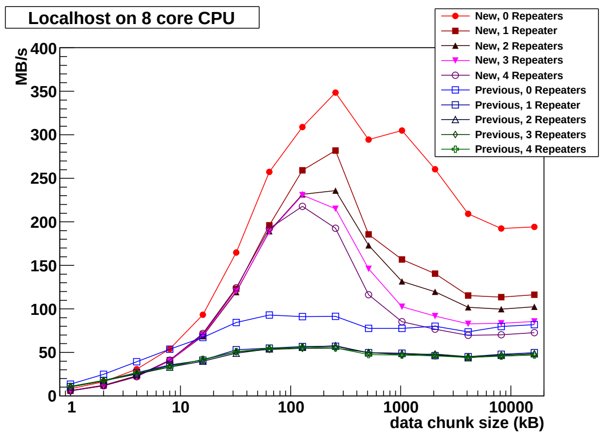

Figure 9. Result of many repeaters on 8 cores CPU machines

2.2. Setup with Loopback device (localhost) and the result

We have measured the total throughput on a multi-core system. The result is shown in Figure 6 and Figure 7. Much higher throughput of new DAQ-Middleware has been observed in comparison with previous DAQ-Middleware.

2.3. Setup with a PC with 8 CPU cores and the result

As we observed the high performance of DAQ-Middleware on a multi-core system, we prepared a PC with 8 CPU cores. Table 2 shows the specification of the PC. Several repeaters between the Source component and the Sink one were used as shown in Figure 8. Figure 9 shows the result. The total throughput on conditions with more than one repeater had similar trend. This means that similar performance will be expected on multi-core system even if the number of component increases.

Throughout all results of the measurement except that in Figure 4, the measurements found a drop of throughput for a data chunk size above 100 kB. Thus, we discuss this point. The packet drop was not observed in all of the measurements. DAQ-Middleware has OutPort and InPort as data path and uses CORBA as data transfer protocol as already described. The InPort is a thread while it has a ring buffer to store data temporarily. Therefore, data copy speed is important for DAQ-Middleware while the protocol enables the flexibility of system configuration against shared memory or message queue mechanism in an Operating system. In all of conditions in the measurements, CPU cache effect should be considered. However, we could not find the reason of the drop of throughput yet in the measurements.

3. Conclusions

Improvement of performance of the new DAQ-Middleware was verified on not only multiple computers but also a multi-core system. DAQ-Middleware will be expected that it works well on a multi-core system. The adoption of multi-core PC instead of 1 Gbps Ethernet may be one of the solutions for DAQ-Middleware from performance point of view. The measurements found a drop of throughput for a data chunk size above 100 kB. More investigation is necessary to find out the reason.

Acknowledgments

The authors wish to thank N. Ando, T. Kotoku and S. Hirano of the National Institute of Advanced Industrial Science and Technology, Japan. The authors also wish to thank people in Open-It, M. Tanaka, T. Uchida, M. Ikeno and KEK electronics system group for their help. This work was performed in the framework of Open-It.

References

[1] Yasu Y, Nakayoshi K, Sendai H and Inoue E 2010 *IEEE Trans. Nucl. Sci.* **57** 487
[2] Object Management Group 2008 *Specification of Robotic Technology Component (RTC)* URL `http://www.omg.org/spec/RTC/1.0/`
[3] Ando N, Suehiro T, Kitagaki K, Kotoku T and Yoon W 2005 *Proc. of IEEE Int. Symp. on Computational Intelligence in Robotics and Automation* 457
[4] The ROOT Team, ROOT URL `http://root.cern.ch/`
[5] Nakayoshi K *et al.* 2009 *Nucl. Instr. and Meth.* A **600** 173
[6] Nakayoshi K *et al.* 2010 *Nucl. Instr. and Meth.* A **623** 537
[7] omniORB URL `http://omniorb.sourceforge.net/`