Journal of Physics: Conference Series **513** (2014) 012020 doi:10.1088/1742-6596/513/1/012020

Control functionality of DAQ-Middleware

H Maeda[1], Y Nagasaka[1], H Sendai[2], E Inoue[2], E Hamada[2], T Kotoku[3], N Ando[3], S Ajimura[4] and M Wada[5]

[1] Hiroshima Institute of Technology, 2-1-1 Miyake, Saeki-ku, Hiroshima 731-5193 Japan
[2] High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801 Japan
[3] The National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568 Japan
[4] Osaka University, 1-1 Machikaneyama, Toyonaka 560-0043, Osaka Japan
[5] Bee Beans Technologies Co., Ltd., Sengen 2-1-6, Tsukuba, Ibaraki 305-0047 Japan

E-mail: nagasaka@cc.it-hiroshima.ac.jp

Abstract. DAQ-Middleware is a software framework for a network-distributed data acquisition (DAQ) system that is based on the Robot Technology Middleware (RTM). The framework consists of a DAQ-Component, that is implemented as a data transfer module, a data gather module, a data record module, etc., and a DAQ-Operator, that is implemented as a control module of other components. The basic functionalities, that are necessary as a DAQ system, such as transferring data, starting and stopping the system, etc., are already prepared in the framework. But one of control functionalities, i.e., the functionality of changing parameter values on the DAQ-Components, wasn't provided yet. In order to implement the functionality, the framework has to have the communication method to transfer data from the DAQ-Operator to the DAQ-Component, and the new state to realize the functionality because it should be separated from a normal state to acquire data. Then we developed and added the new functionality in the DAQ-Middleware to transfer data from DAQ-Operator to DAQ-Components in the new state. The new DAQ-Middleware framework allows us to implement easily not only functionality of acquiring data but also that of controlling component modules.

1. Introduction

A high performance data acquisition system is used to collect data in many high energy and nuclear physics experiments. Moreover, most of the systems are connected to each other by using network technologies such as Ethernet and TCP/IP. A system called network-distributed DAQ is normally used to gather event fragment data from read-out modules of detectors.

Although the functionality of the DAQ system is almost the same, the system has been developed individually for each experiment so far, and it caused the system development efficiency to become smaller. We have developed a software framework for DAQ systems, called DAQ-Middleware in the following.

The DAQ system can be easily implemented by using this framework, and we can collect to data taken by an experiment. The role of the DAQ system is not only to collect data but also to control the system. The basic control functionalities such as starting and stopping the system is provided by the framework, but the functionality to transfer data from the DAQ-Operator to the DAQ-Components

was not implemented. It is useful, for example, when we change threshold values on readout modules or high-voltage values via modules. The functionality to set the first values on DAQ-Component was provided, but we were not able to change values after the DAQ system started, and the system had to be restarted when we changed them.

We developed the functionality to transfer data from the DAQ-Operator to the DAQ-Components to change values in the DAQ-Components. In order to enable the new functionality, the new state, SET, should be introduced in the DAQ-Middleware. The new functionality was implemented in the state and it allows us to transfer data, which are stored in a parameter file on the DAQ-Operator, to the specified DAQ-Components. By using this functionality, any parameters can be transferred and set with the DAQ-Middleware without restarting the system.

2. DAQ-Middleware

DAQ-Middleware is a software framework to develop for a network-distributed DAQ system easily[1-3]. It is a network-distributed DAQ framework and based on the Robot Technology Middleware, RTM[4], developed by the National Institute of Advanced Industrial Science and Technology, AIST. The framework is also based on an object-oriented technology.

DAQ-Middleware consists of two kinds of nodes, a DAQ-Component and a DAQ-Operator. These are objects and communicate with each other with CORBA. The basic functionalities such as transferring data, starting and stopping the system, etc., are already implemented in the DAQ-Component and the DAQ-Operator. The DAQ-Component is especially used as a core component to read, transfer, and record data. On the other hand, the DAQ-Operator is used as a controller for the DAQ-Components. The framework also works as a state machine system and it has four states; LOADED, CONFIGURED, RUNNING and PAUSED. The transition between these states occurs by a command that is transmitted by the DAQ-Operator.

Figure 1 shows a typical example of a system architecture that is developed with using DAQ-Middleware. The system consists of one DAQ-Operator and four DAQ-Components on two computers, PC(UI) and PC(DAQ#1).

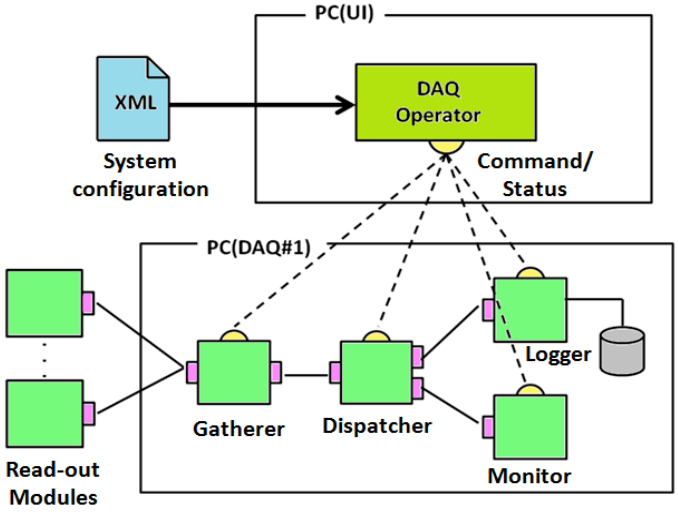

Figure 1. Typical example of a system architecture based on DAQ-Middleware

The PC(UI) is used for a user interface and the DAQ-Operator. The user controls the system via the DAQ-Operator with using a system configuration file written in XML. The DAQ-Operator transmits the control command to each DAQ-Components. On the other hand, four DAQ-Components are implemented on the PC(DAQ#1). These four components, Gatherer, Dispatcher, Logger, and Monitor shown in the figure, are typical components of a DAQ system and are developed by using the DAQ-Component of DAQ-Middleware. The Gatherer gathers event fragments from Read-out Modules, builds an event and transfers it. The Dispatcher duplicates event data that are received from the Gatherer and transferred to two components, the Logger and the Monitor. The Logger stores the event into the database, and the Monitor shows the data for checking them. In this figure, it shows these components on only one computer, but it is also possible to implement them on several computers separately because the communication between components is implemented with CORBA.

3. Control functionality

We developed and added the new control functionality into the DAQ-Middleware. It enables the DAQ-Operator transfer data to the DAQ-Components without restarting the system. And we also introduced the new state, SET, for this new functionality.

A state diagram of a DAQ system based on the DAQ-Middleware is shown in Figure 2. DAQ-Middleware has four states originally, which are shown as blue boxes in the figure. The transitions of these states are triggered by commands, *configure*, *unconfigure*, *start*, *stop*, *pause* and *resume*. The data acquisition is normally performed in the RUNNING state. As these four states were prepared only for collecting data, we introduced the new state, SET, which was shown as a red box in the figure. The state is transited from the PAUSED state by the command *set* and to the PAUSED state by the command *return*.

When the DAQ-Component receives the command *set* and data form the DAQ-Operator, the *daq_setting( )* method is called, and the state of the component transits to SET state. After finishing the state transition, the *daq_set( )* method is called repeatedly until receiving command *return* to transit to the PAUSE state. A user can implement functionality to change values on the DAQ-Component by using these two methods.

The data to transfer from the DAQ-Operator are stored in a file on the operator. The file is specified in the process on the DAQ-Operator. The file consists of value's names, their values, DAQ-Component IDs, etc., which are written in XML in order to transfer several different values to different components. The format is basically the same as that of the configuration file, which is used in the configuration process at the beginning when starting the system.

By using this functionality, users can easily implement the functionality to control the DAQ-Components and communication method between DAQ-Operator and DAQ-Components.

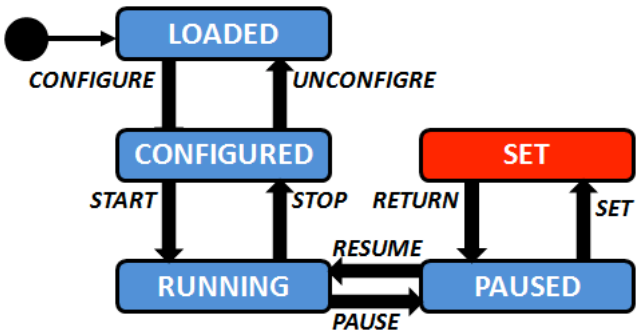

Figure 2. State diagram of a DAQ system based on DAQ-Middleware

4. Performance measurements

4.1. *Setup*

The transfer times after opening a file that stores transfer values until finishing transferring it from the DAQ-Operator to the DAQ-Component were measured to evaluate a performance of the new functionality.

Table 1 shows the specification of two computers used for the performance test. Two computers were connected directly via 10 Gigabit Ethernet network without any switches. The data used for the performance measurement were binary one and were prepared before the measurements.

Table 1. Specifications of test computers

CPU	Intel(R) Xeon(R) CPU E5606 2.13 GHz
Memory	2048 MBytes
Network	Intel Corporation 82598EB 10-Gigabit AT2 Server Adapter
OS	Scientific Linux 6.2

4.2. *Results*

The results of our measurements are shown in Figure 3. The transfer times in the unit of msec are plotted as a function of parameter data sizes in the unit of bytes.

As shown in the figure, the transfer time becomes bigger for increasing data sizes, and the absolute values are from about 4.5 msec to about 6.0 msec. This transfer time includes a file access and a data transfer times in the process of the DAQ-Operator. The overhead of the file access in the process is about 4.5 msec. The transfer speed of parameter data is a few Mbps in the case of 1 kBytes of data size. As the DAQ-Middleware is based on object-oriented architecture and uses CORBA architecture to communicate each other, the data transfer speed is limited by its architecture and the test environment.

The performance test shows the new control functionality works well without failures. The new DAQ-Middleware with the functionality allows to set values to modules on the DAQ-Component without a restart procedure.

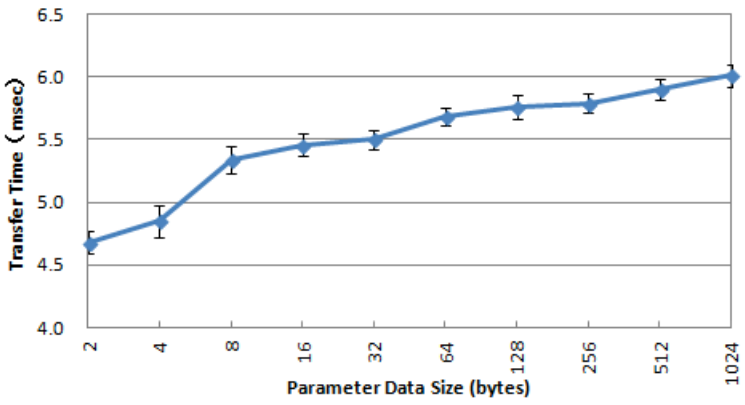

Figure 3. Transfer times as a function of data sizes of configuration file

5. Conclusions

We have developed the DAQ-Middleware, a software framework for the network-distributed DAQ system. The basic functionalities are already implemented and we can develop a DAQ system easily by using this framework. The control functionality to set values was not provided, so we developed it into the framework. A new state, SET, was introduced in the framework. The transfer times of data stored in the file were measured to evaluate the system. Although the transfer time is a few msec, it is obviously faster than original middleware, because the original system has to be restarted in order to set such values to the DAQ-Component. The new control functionality worked without failures, and it was useful to reduce the time to set values on the DAQ-Components.

References

[1] Y. Yasu, K. Nakayoshi, E. Inoue, H. Sendai, H. Fujii, N. Ando, T. Kotoku, *et al*., "A Data Acquisition Middleware," Proc. IEEE/NPSS Real Time Conference, pp. 1-3, May 2007.
[2] Y. Yasu, K. Nakayoshi, H. Sendai, and E. Inoue, "Functionality of DAQ-Middleware," IEEE Trans. Nucl. Sci., vol. 57, no. 2, pp. 487-490, 2010
[3] K. Nakayoshi, H. Sendai, Y. Yasu, E. Inoue, T. Kotoku, N. Ando, Y. Nagasaka, S. Ajimura and M. Wada, "DAQ-Middleware: Progress and status," J. Phys.: Conf., Ser. 331 02202318, 2011
[4] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W. K. Yoon, "RT-Middleware: distributed component middleware for RT (robot technology)," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 3933-3938, 2005