PERFORMANCE EVALUATION OF EPICS CHANNEL ARCHIVER VIA PYTHON XMLRPC INTERFACE

A. Toyoda, Y. Sato, H. Noumi, Y. Igarashi, K. Nakayoshi,
Y. Hayato, M. Tanaka, H. Fujii and H. Kodama
KEK, 1-1, Ooho, Tsukuba, Ibaraki, 305-0801 Japan

Abstract

J-PARC (Japan Proton Accelerator Research Complex) [1] under construction aims to provide the high intensity proton beam (50 GeV, 15 μA) and extend high intensity frontiers of particles and nuclear studies. To handle primary beams of the slow-extraction beam line at J-PARC, it is required for the control system to display real-time DC signals from a large number of thermometers, vacuum gauges and so on. It is also necessary to archive such kind of data every few seconds to get historical information on each apparatus. As part of the R&D program to construct such control system, we have developed a prototype system to archive data with the EPICS [2] Channel Archiver [3] and to display them via Python XMLRPC interface. The performance and evaluation of this system are reported.

CONTROL SYSTEM OF THE J-PARC SLOW EXTRACTION BEAM LINE

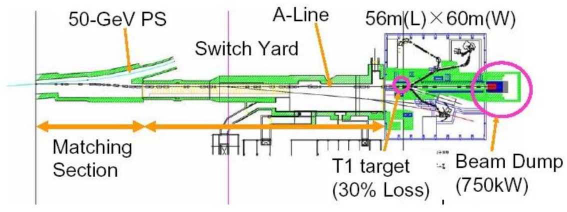

Figure 1: J-PARC slow extraction beam line.

Figure 1 shows a schematic figure of J-PARC slow-extraction beam line[4]. A proton beam is accelerated to 50 GeV in linac and 2 synchrotrons, and extracted into this beam line. The extracted beam duration will be about 0.7 seconds and the accelerator cycle will be 3.4 seconds or so. The proton beam is extracted into HD hall via a switch yard, and focused on a T1 target as a proton target. Secondary particles such as kaons, pions, and so on are transported into a secondary beam line, and the residual primary proton beam is introduced into 750 kW beam dump.

Figure 2 shows all of our control components. There are many control components such as a monitoring system for various status of each apparatus, a control system for magnets and several kinds of beam monitors, data logging system, and so on. In this article, we focused on the temperature monitoring system and its data logging system.

Necessary specifications

In our beam line, there are several apparatus located at a big beam loss point such as the T1 target, the beam dump, a collimator, and so on. For the safe beam operation, it is necessary to observe the temperature distribution and its historical trend for such apparatus. We are planning to set up more than a thousand of thermocouples for this purpose. Thus the unit cost to measure 1 thermocouple should be as low as possible. The other important point is to measure temperature distribution with the rate of at least 1 scan per 1 beam spill ($\sim$ 3.4 seconds). To satisfy these two conditions, we chose the Agilent [5] data acquisition / switch unit (34970A) and EPICS system. For data logging system, we prepared the EPICS channel archiver. To display real-time data and historical trend, we used the python with XMLRPC, EpicsCA [6], and Tk interfaces.

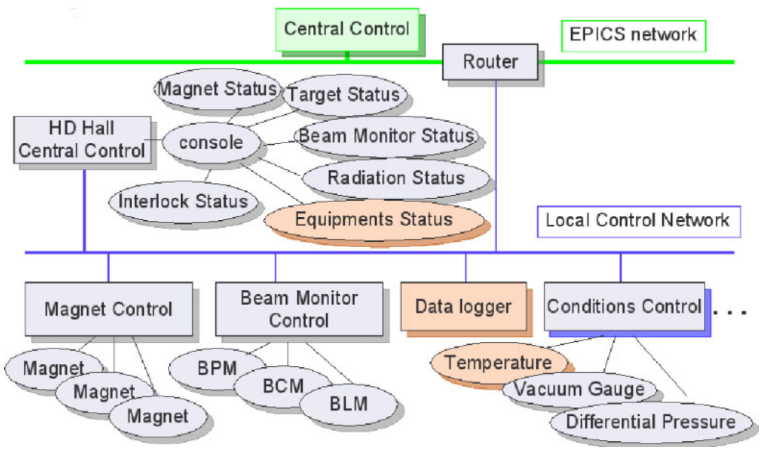

Figure 2: Control components.

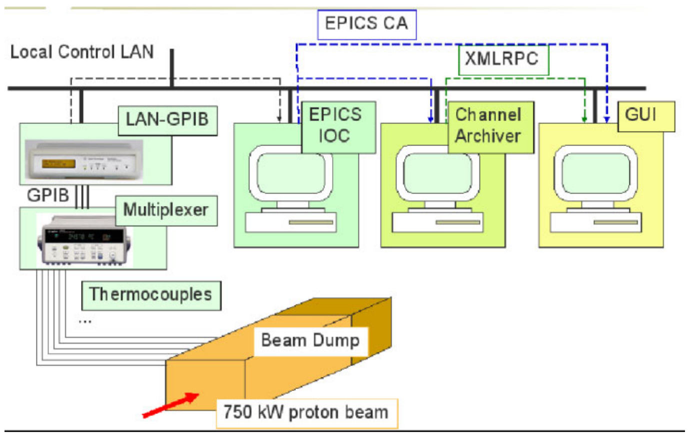

Figure 3: Status viewer framework.

Framework of our status viewer

Figure 3 shows the framework of our status viewer. Many K-type thermocouples are connected to the apparatus such as a beam dump. These thermocouples are measured by an Agilent multiplexer. This multiplexer is connected to a LAN-GPIB Ethernet gateway (E5810A) via the GPIB interface. By this module, An EPICS IOC (Input/Output Controller) can access data via a local control LAN. The EPICS channel archiver as a data logger and a GUI display extract data from an EPICS IOC via an EPICS channel access. The GUI display also extracts archived data from the data logger via a XMLRPC protocol.

EPICS IOC SPECIFICATIONS

First of all, we describe the EPICS IOC in detail. Used hardware for a thermometer scanner are a data acquisition unit (Agilent 34970a), three sets of 20 channel multiplexer modules (Agilent 34901a), and a LAN-GPIB Ethernet gateway (Agilent E5810a). The EPICS IOC is run on PC/AT compatible (CPU: Pentium IV 2.8 GHz; memory: 500 MB; HDD: 5400 rpm 40 GB ATA). The operating system is Linux (debian GNU/Linux sarge; gcc 3.3.4/ kernel 2.6.6-1-686). An EPICS version is R3.14.6, and the data record type of the EPICS channel is the waveform record. We used async driver whose version is 4-2 as an EPICS GPIB interface.

By combining the above components, we achieved the relatively low cost of 5000 Japanese yen per channel and the acceptable scanning speed of 22 channels readout per second which corresponds to 3 seconds per scanning. This relatively slow scanning speed is mainly dependent on the slowness of the multiplexer module readout, not on the EPICS IOC performance, a LAN interface, a GPIB interface, and so on.

DATA LOGGER

Secondly, we explain the EPICS channel archiver as a data logger. We used the channel archiver of version 2.1.8, a XMLRPC library of version 0.9.10-4, a Xerces XML library of 2.4.0-3, and python as a XMLRPC client of version 2.3.5.

Performance test of EPICS channel archiver

First of all, we explain the scheme to extract data from the channel archiver briefly. The XMLRPC client extracts archived data from the data logger via the XMLRPC protocol. The data logger always stored data every second. The XMLRPC client requests 50 data points within the specified time window from a starting time to an end time. According to this request, the data logger server averaged the archived data within each time window cell of 1/50 of the time window, and send the resulting 50 data points back to the client side.

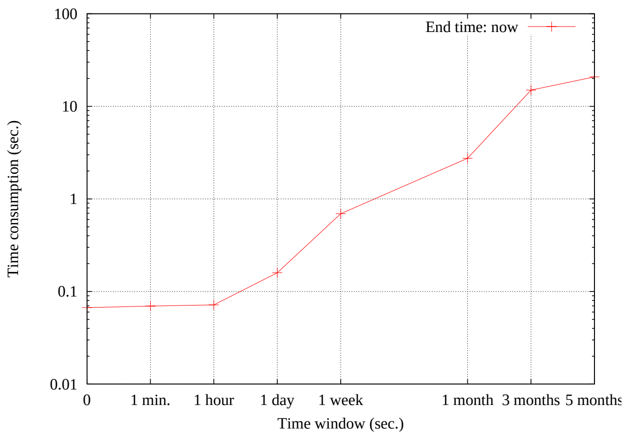

Figure 4: Time-window dependence of time consumption to extract all data. The horizontal axis is for the time window from 1 minute to 5 months. The vertical axis is for the time consumption in the unit of second to extract all requested data from the EPICS channel archiver.

Figure 4 shows the time-window dependence of the time consumption to take all of the requested data from the channel archiver. The conditions of this test are listed as follows.

- Number of thermocouples connected is 2.
- Data retrieval method of the channel archiver is averaged.
- Number of retrieved data points is 50.
- Data writing rate of the channel archiver is 1 writing per second.

As shown in this figure, there is a large dependence on the time window size. For a good response to retrieve data within 1 second, a time window should be smaller than a few weeks.

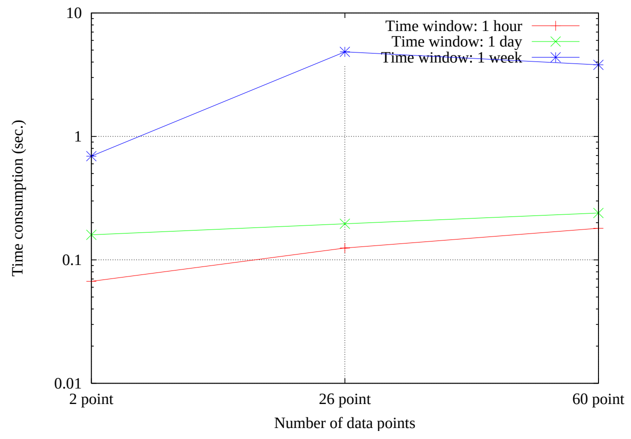

Figure 5: Dependence of a time consumption on number of thermocouples connected. A blue line is for the dependence with a time window of 1 week. A green line is for the dependence with a time window of 1 day. A red line is for the dependence with a time window of 1 hour.

In the next test, we measured a dependence of a time consumption on a number of thermocouples connected as shown in figure 5. In the case of dozens of thermocouples connected, it takes one order longer time especially for the relatively large time window of 1 week or so. For good response such as a time consumption of 1 second, we need

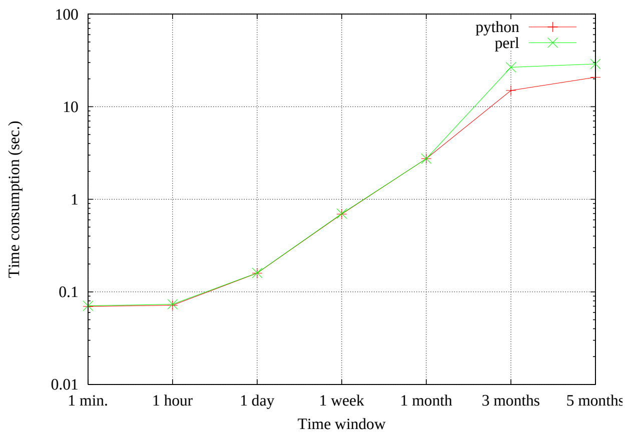

Figure 6: XMLRPC client dependence of the time consumption. A green line is for the perl client with perl of version 5.8.4 and Frontier library of version 0.07b4. A red line is for the python client whose version is 2.3.5.

to set the time window to be less than a few days or so with 60 thermocouples connected.

Lastly, we tested a XMLRPC client dependence on the time consumption. As shown in figure 6, almost no dependence on a client side was observed. This result indicates that the data extraction speed is mainly limited by the channel archiver server performance in our archiving setting.

EXPERIMENTAL MEASUREMENT

In this section, we described an experimental measurement with our newly-developed data viewer which is programmed with python.

Figure 7 shows the experimental setup. A copper block whose size is 1000 $\times$ 200 $\times$ 50 millimeters was prepared as a mockup of a beam dump. To warm up this copper block uniformly, fifteen heaters were installed from the top side of the copper block. The heater power was controlled from 5 to 10 kW. To cool the heated copper block, we made a hole from left to right as shown in the figure, and a running water line was connected to this hole. The water flow rate was controlled to be 20 l/m. To measure the temperature distribution, 56 K-type thermocouples (4 $\times$ 14) were

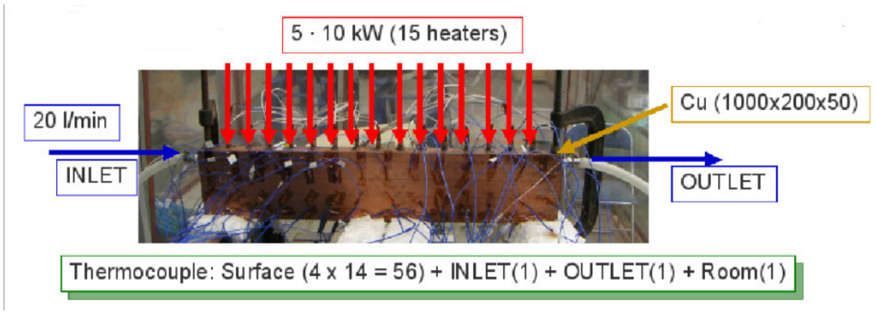
5 · 10 kW (15 heaters)
20 l/min
INLET
Cu (1000x200x50)
OUTLET
Thermocouple: Surface (4 x 14 = 56) + INLET(1) + OUTLET(1) + Room(1)

Figure 7: Experimental setup.

uniformly put on the surface of the copper block. The thermocouples were also set up at inlet and outlet line of the running water. One thermocouple was prepared to measure the room temperature.

For calibration, we also prepared an infrared thermometer. To make sure the emissivity to be constant, a black tape was put on the copper block.

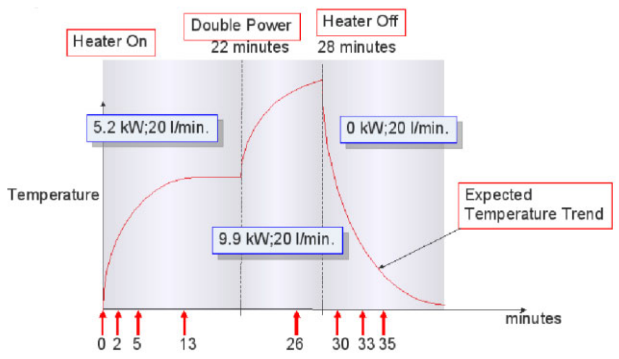

Figure 8: Experimental sequence. Horizontal axis is for the clock time, vertical axis is for the temperature in the arbitrary unit. Solid line is the expected temperature trend.

Figure 8 shows the experimental sequence. At time zero, we switched on the heater with the power of 5.2 kW. 22 minutes after the heater on, the heater power was doubled (9.9 kW). 26 minutes after the heater on, we switched off the heater.

The GUI data viewer extracted real-time data via EPICS channel access with EpicsCA interface whose version is 1.4.5. A historical trend was extracted from the EPICS channel archiver via XMLRPC protocol.

Real-time distribution GUI

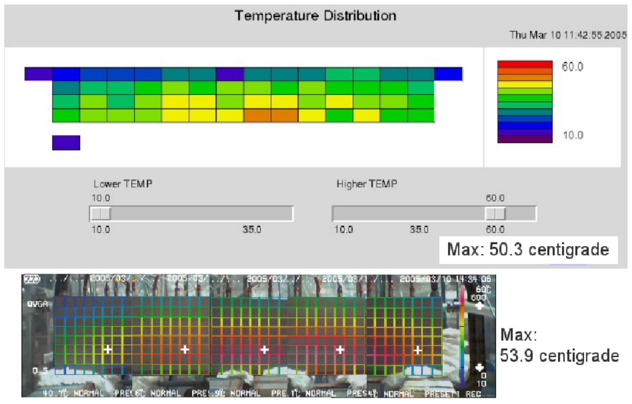

Figure 9: Real-time distribution GUI. These data are measured at 13 minutes after the heater on. The top figure is for a GUI window of our data viewer for the real-time temperature distribution. The bottom figure is for a picture taken by the infrared thermometer for calibration.

Figure 9 shows the real-time temperature distribution with our data viewer. The temperature distribution is almost the same as each other. The maximum temperature is also the same as each other. Our data viewer for the real-time distribution worked well.

Historical Trend GUI

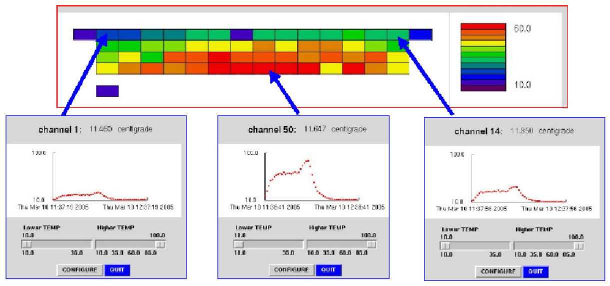

Figure 10: Historical trend GUI.

Figure 10 shows our historical trend viewer. When we clicked on the real-time data viewer described in the previous subsection, a window of the historical trend viewer is pop up, and archived data are extracted from the data logger. We achieved to get the nice historical trend of the corresponding channel as shown in this figure.

SUMMARY

We developed a status display to monitor a thousand of thermocouples by combining the Agilent data scanner system, the EPICS IOC, and python Tkinter for the J-PARC slow-extraction beam line. The data acquisition speed of about 3 seconds per scan is enough, and the channel unit cost of 5000 Japanese yen is low enough. We achieved to log and extract data with the EPICS channel archiver. We analyzed the channel archiver performance, and found that the time window needed to be less than a few days for a good response with 60 (full) thermocouples connected. The data extraction speed is limited by not the client but the performance of a channel-archiver host computer. We also experimentally measured the real-time temperature distribution and its historical trend, and confirmed that it worked well by comparing with data of an infrared thermometer.

REFERENCES

[1] http://j-parc.jp/index.html
[2] http://www.aps.anl.gov/epics/
[3] http://ics-web1.sns.ornl.gov/ kasemir/archiver/
[4] K.H. Tanaka et al., ìTec hnical design report II for the slowextraction beam facility at the 50-GeV in J-PARCî, KEK internal report 2004-3
[5] http://www.agilent.com
[6] http://cars9.uchicago.edu/ newville/Epics/Python/