# Recent development of a monitoring system for beam line status at the J-PARC slow-extraction beam line

A. Toyoda, K. Agari, H. Fujii, Y. Hayato, E. Hirose, M. Ieiri, Y. Igarashi, Y. Kato, H. Kodama, M. Minakawa, K. Nakayoshi, H. Noumi, Y. Sato, S. Sawada, Y. Suzuki, H. Takahashi, M. Takasaki, K. H. Tanaka, M. Tanaka, H. Watanabe, Y. Yamada, and Y. Yamanoi

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

## ABSTRACT

The 50-GeV proton synchrotron of J-PARC (Japan Proton Accelerator Research Complex) [1] now under construction aims to provide a high intensity proton beam (50 GeV, 15 μA) to various nuclear and particle physics experiments. For stable operation of the beam, it is necessary to monitor temperature, vacuum, and differential pressure from the gauges that are attached on apparatus at large beam loss points. It is also required to archive the data of the status of beam line and those apparatus to be able to get historical information and investigate cause of a beam failure. As part of the R&D program to construct such control system, we have recently developed a data logger with the EPICS [2] channel archiver [3] and online GUI display programmed in Python with the XMLRPC and the Tk interface. The performance and evaluation of this system are reported.

## INTRODUCTION

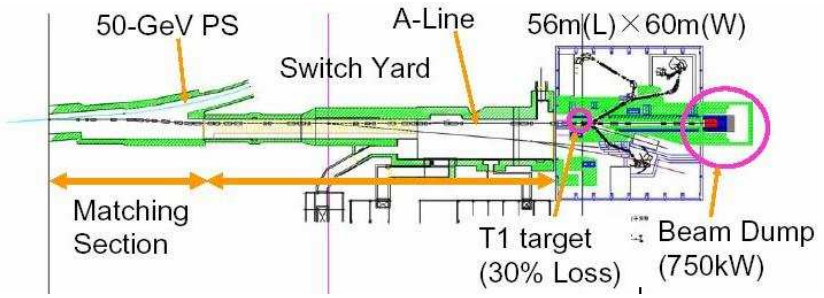


Figure 1: Schematic view of the J-PARC slow extraction beam line

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 3.4 seconds. The proton beam is extracted into HD hall (56 m (L) x 60 m (W)) via a switchyard, 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.

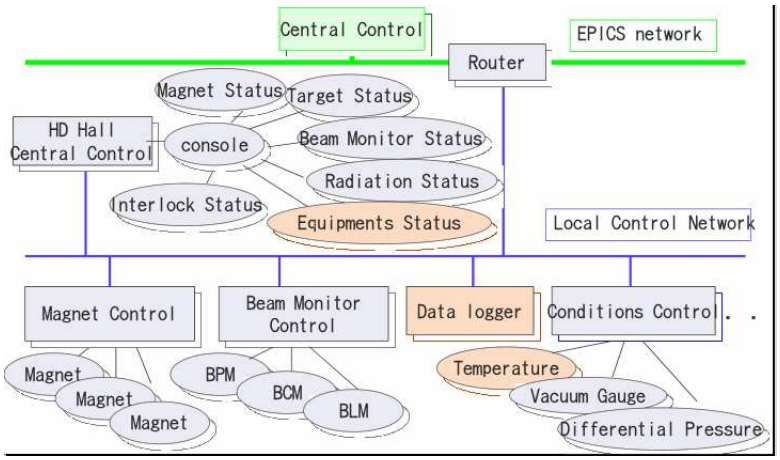


Figure 2: Control components of the J-PARC slow extraction beam line

Figure 2 shows the overview 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 for the control system*

In our beam line, there are several apparatus attached at large beam loss points such as the T1 target, the beam dump, a collimator, and so on. For the safety 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 to measure the temperature distribution. Thus the cost to measure 1 thermocouple should be as low as possible. The other point is that the scan rate to take data should be at least 1 beam spill cycle (about 3.4 seconds). To satisfy these two conditions, we chose the Agilent [5] data acquisition/ switch unit (34970A) and EPICS system. By this system, we achieved the relatively low cost of about $50 per channel. The measured scanning speed with normal readout precision of 0.1 centigrade (an integration time of 1 power cycle) is about 22 channels readout per second, which corresponds to about 3 seconds per scanning. We prepared the EPICS channel archiver for the data logging system. To display the real-time data and historical trend, we used the python program with XMLRPC, EpicsCA [6], and Tk interfaces.

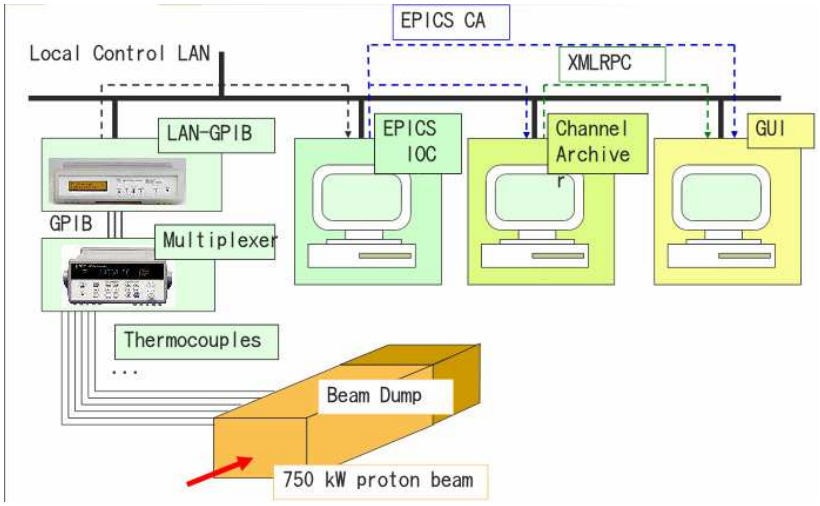


Figure 3: Status viewer framework

*Framework of our status viewer*

Figure 3 shows the framework of our status viewer as the temperature monitoring system and its data logging system. Many K-type thermocouples are connected to the apparatus such as a beam dump. These thermocouples are measured with three modules of 20-channel multiplexers (34901a) by the Agilent data acquisition unit. 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, and show the result with the Tk interface.

## PERFORMANCE MEASUREMENT

First of all, we describe specifications of the EPICS IOC. The EPICS IOC is run on a PC/AT compatible machine (CPU; Pentium IV 2.8 GHz, memory; 500 MB, HDD; 5400 rpm 40 GB ATA100). The operating system is Linux (Debian GNU/Linux sarge; gcc 3.3.5/kernel 2.6.7-2-686). A local control LAN is 100 base-T Ethernet. An EPICS base version is R3.14.7, and the data record type of the EPICS channel is waveform record. We used async driver whose version is 4-2-1 as an EPICS GPIB interface.

Secondly, we describe specifications of the channel archiver server. The channel archiver is run on a PC/AT compatible machine (CPU; Pentium IV 2.8 GHz, memory; 1 GB, HDD; 7200 rpm 120 GB

ATA133). The operating system is Linux (Debian GNU/Linux sarge; gcc 3.3.5/kernel 2.6.7-2-686). We used the channel archiver of version 2.3.0, 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 2.3.5-4.

### *Measurement of the XMLRPC client performance*

In this section, we evaluate a performance of XMLRPC as a client of the channel archiver server. To suppress an overhead by the server processing speed as low as possible, we optimised extracted data size to avoid the averaging process on the server. A CPU load of the server is always less than 5 %, so that the consumption time to extract data is almost consumed by the client side.

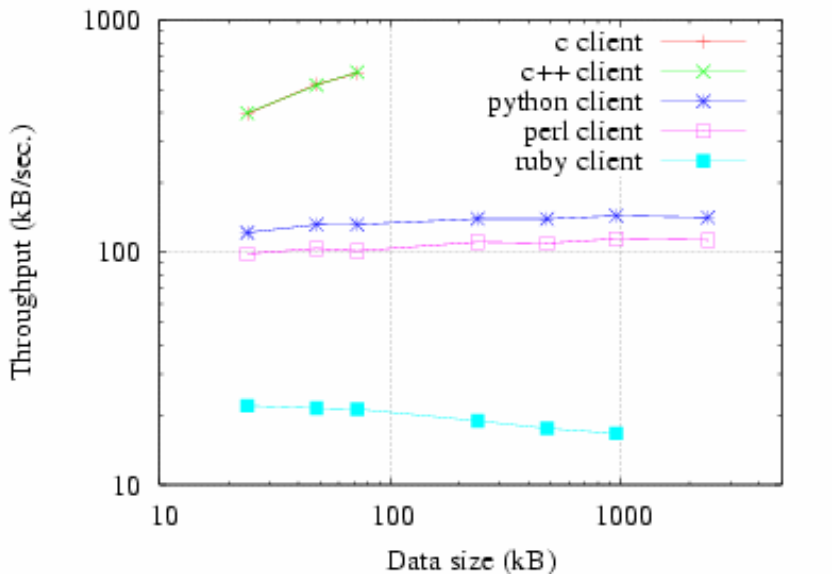


Figure 4: Result of XMLRPC throughput for each XMLRPC client. For c and c++ clients, there is a limit for data size of the XMLRPC response, so there is no data for large data size.

Figure 4 shows a result of XMLRPC throughput for each XMLRPC client. The version of c client and c++ client is 0.9.10-4. The version of the python client is 0.8.4-1. The XMLRPC parser of the python client is sgmlop [7]. The perl client is Frontier::RPC XMLRPC module whose version is 0.07b4-3. The perl version is v5.8.7. The XMLRPC parser of the perl client is a wrapper of the expat library written in c (version 1.95.8). The ruby version is 1.8.2-1. The XMLRPC parser of the ruby client is also a wrapper of the expat library. In this test, both the client and the server is run on the same machine (Pentium IV 2.8 GHz with SMP setting) to minimize a network bottleneck. We measured the connection speed with iperf (version 2.0.2-1). The result is 3.79 Gbits/s for localhost and 94.1 Mbits/s over Ethernet, thus there is almost no network bottleneck. As shown in this figure, the performance of c and c++ clients is the best (about 600 kB/s throughput). The clients using script language such as python and perl have about 1/4 performance of the programming language such as c. A performance of the python client is about 25 % better than that of the perl client. The performance of the ruby client is about 1/7 of the other clients using script language.

We checked the XMLRPC performance with changing the CPU power running the client program. For this purpose, we prepared another PC/AT compatible machine (Pentium III 500 MHz, memory 256 MB, HDD 34 GB 7200 rpm ATA66). We already checked the memory performance and the HDD performance do not affect the XMLRPC performance, so the performance difference comes from the CPU difference. For all client programs, the performance with Pentium III 500 MHz machine is 1/6 of that with Pentium IV 2.8 GHz machine. This ratio is about the same as the ratio of two CPU clocks. To check the SMP (Symmetric Multiple Processor) effect, we changed the OS setting for the SMP and measured a difference of performance. For all client programs, the performance with the SMP setting is always 10 % better than that without the SMP setting.

*Measurement of performance of the Channel Archiver server*

In this section, we evaluate a performance of the channel archiver server. For this purpose, we fixed a number of extracted data points to 50. The data extraction method is averaging over each time window cell. The channel archiver server always stored 60 channel temperature data every 5 seconds. The XMLRPC client requests 50 data points within a specified time window. According to this request, the server averaged the archived data within each time window cell of the 1/50 time window, and sent the resulting 50 data points back to the client.

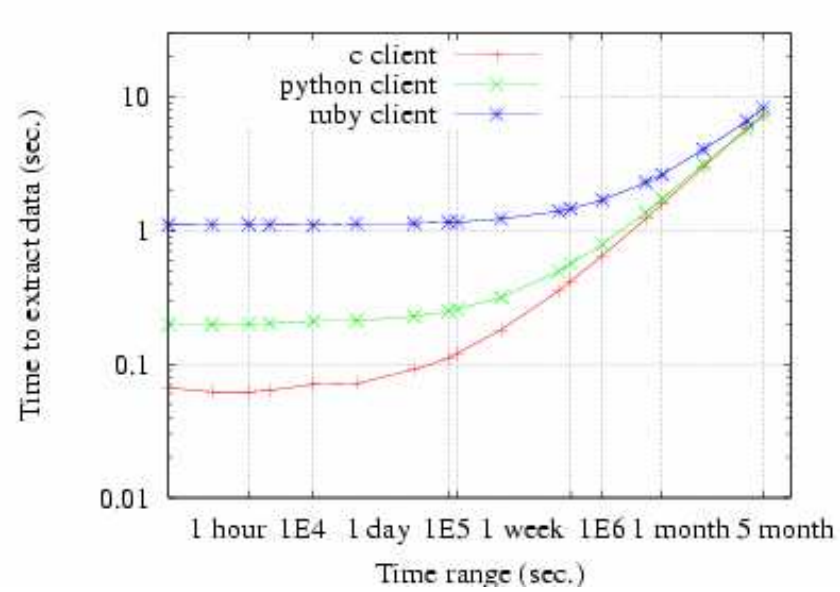


Figure 5: Dependence of consumption time to extract data on the time range of the extracted

Figure 5 shows the dependence of the consumption time to extract data on the time window size. As shown in this figure, the fastest c client is the most sensitive to the performance of the server. The time to extract data is linearly dependent on the time window. To get the averaged data within the time window more than 1 month, the response gradually becomes slow (about 1 second or more).

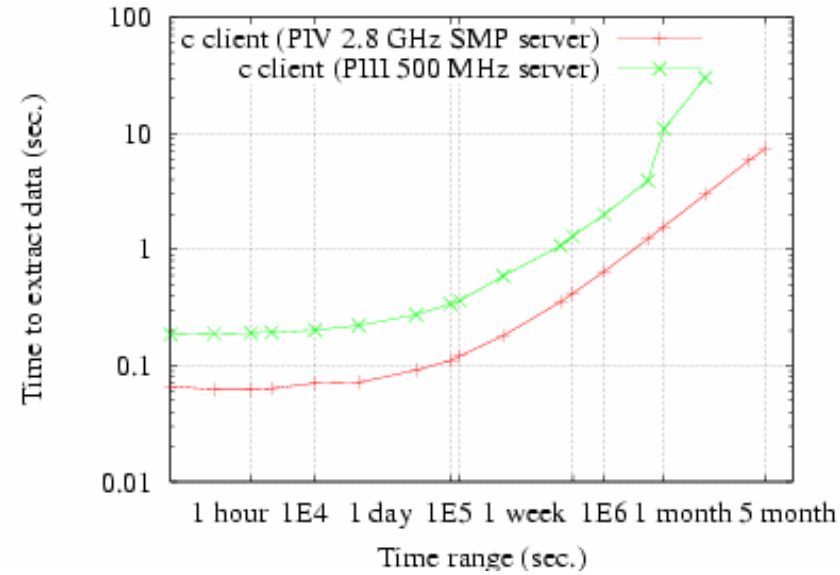


Figure 6: Dependence of consumption time to extract data on the time range of the extracted data for Pentium IV 2.8 GHz PC server and Pentium III 500 MHz PC server, respectively.

Figure 6 shows the server performance with changing the CPU power. For the time to extract data below 4 seconds, the performance of the Pentium III 500 MHz server is always 3 times lower than that

of the Pentium IV 2.8 GHz server. This ratio is about half of the CPU clock ratio. For the time to extract data more than 4 seconds, the performance of the Pentium III 500 MHz server rapidly becomes worse. To understand this phenomenon, we investigated an effect of the SMP setting.

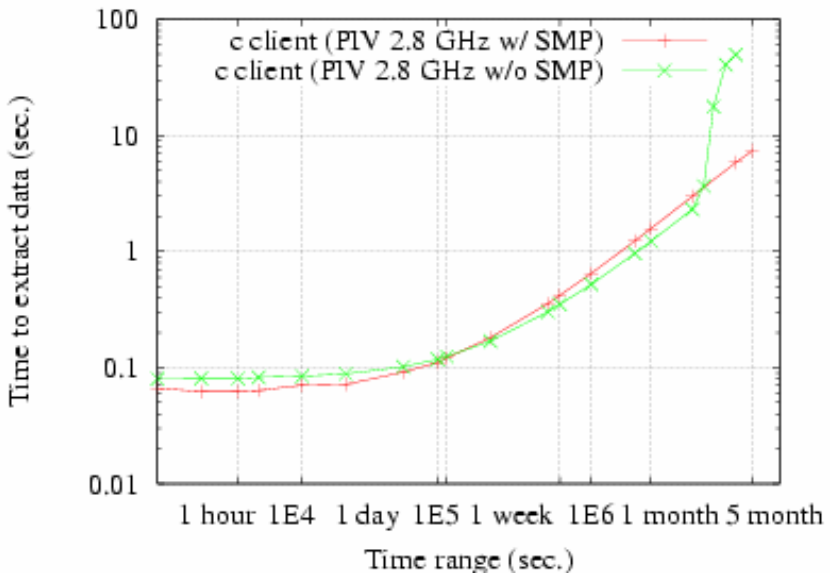


Figure 7: Dependence of consumption time to extract data on the time range of the extracted data for Pentium IV 2.8 GHz PC server and Pentium III 500 MHz PC server, respectively.

Figure 7 shows the result of the performance test for the SMP setting. As shown in this figure, there is no large difference below the time to extract data of 4 seconds. However, the performance difference rapidly becomes large for the time to extract data more than 4 seconds. The server simultaneously runs a program to archive data and a program to extract archived data, thus the SMP becomes effective. It is commented that the server archives data every 5 seconds which is about the same as the above threshold of 4 seconds.

## GUI DEVELOPMENT

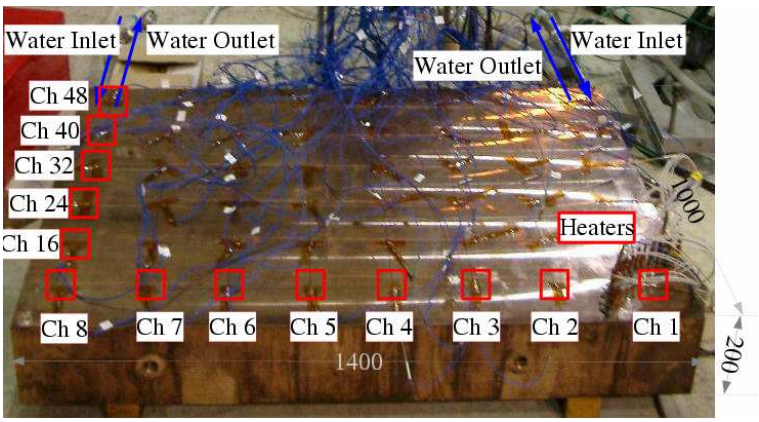


Figure 8: Experimental setting to measure temperature distribution of a mock up for the HD hall beam dump.

In this section, we describe an GUI viewer development of our system and its application. One of our GUI viewer applications is the temperature-distribution viewer for the beam dump mock up. Figure 8 shows the experimental setting of the mock up. This is the quarter part of one slab of the beam dump composed of an oxygen-free copper block. The dimension of this slab is 1400 mm x 1000 mm x 200 mm. The real beam dump is composed of 25 slabs of the copper block (2000 mm width x 1000 mm height x 5000 mm thick). This mock up has two of cooling water lines for redundancy. From a result of a thermal analysis, a heat deposit of each slab is 50 kW at maximum. Thus this quarter mock up is heated with the power of 12.5 kW by 15 heaters placed at around the corner. The flow rate

of the cooling water is adjusted to be 15 litters/s just below the critical flow rate to avoid the water-flow damage to the copper block. We attached 48 K-type thermocouples to measure the temperature distribution of the surface of the mock up.

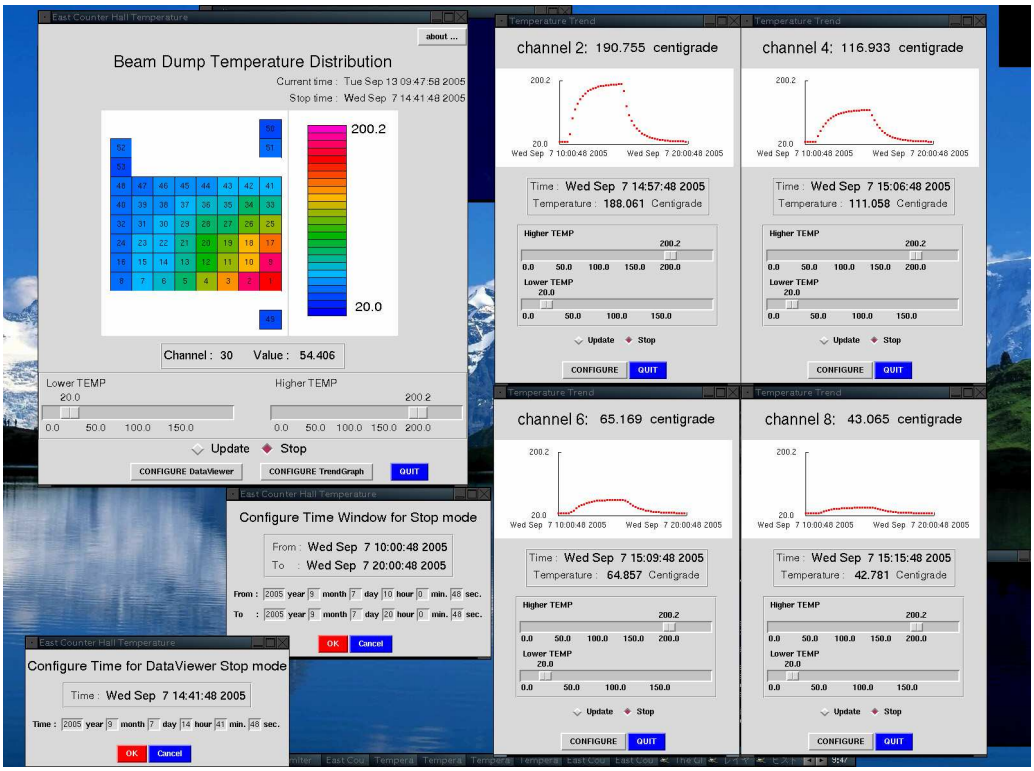


Figure 9: Screenshot of a newly developed temperature distribution viewer for a beam dump mock up. The top-left window is a main window to display a temperature distribution. The right 4 windows show the trend graph of each thermocouple data. The bottom-left 2 windows are for configuration of the time range for the temperature distribution and the trend graph.

Figure 9 shows the screenshot of the GUI viewer for this mock up. We achieved to monitor the temperature distribution and its historical trend by this viewer.

## SUMMARY

We developed a GUI viewer to monitor a thousand of thermocouples by combining the Agilent data scanner system, the EPICS framework, and python Tk interface for the J-PARC slow-extraction beam line. We also achieved to log and extract data with the EPICS channel archiver. We analysed the performance of the EPICS IOC, the channel archiver, and the XMLRPC client, and confirmed that this system satisfies the requested specifications. We applied this proto-type system for the mock up of the beam dump successfully.

## 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., “Technical design report II for the slow-extraction beam facility at the 50-GeV in J-PARC”, KEK internal report 2004-3.
[5] http://www.agilent.com
[6] http://cars9.uchicago.edu/~newville/Epics/Python/
[7] http://effbot.org/zone/sgmlop-index.htm