DATABASE SCHEME FOR UNIFIED OPERATION OF SACLA/SPring-8

K.Okada[†], M.Ishii, N.Hosoda, T.Sugimoto, M.Yamaga, Japan Synchrotron Radiation Research Institute, Hyogo, Japan
T.Fukui, T.Maruyama, T.Fujiwara, K.Watanabe, RIKEN SPring-8 Center, Hyogo, Japan
H. Sumitomo, SPring-8 Service Co. Ltd. Hyogo, Japan

Abstract

For a reliable accelerator operation, it is essential to have a centralized data-handling scheme, such as unique equipment IDs, archived and online data from sensors, and operation points and calibration parameters for restoration upon a change in operation mode. Since 1996, when SPring-8 began operation, a database system has been utilized for this role. However, as time has passed, the original design has become outmoded, and new features equipped upon request have increased the maintenance costs. For example, through the start of SACLA in 2010, we introduced a new data format for shot-by-shot synchronized data. In addition, the numbers of tables storing operation points and calibrations have increased with various formats. Facing project upgrades on-site, it is now time to overhaul the entire scheme. In this project, SACLA will be a high-quality injector for a new storage ring while in operation as the XFEL user machine. To handle multiple shot-by-shot operation patterns, we plan to introduce a new scheme wherein multiple tables inherit common parent table information. In this paper, we report a database design for a project upgrade and the transition status.

INTRODUCTION

The SPring-8 complex has been leading the synchrotron radiation community for almost two decades. SACLA [1], the XFEL facility, has been in operation since 2010, and SXFEL beamline [2] as a soft X-ray source has recently been added.

To help the community evolve, a project upgrade (SPring-8-II project) is under way, in which the oldest SPring-8 storage ring (SR) will be rebuilt into a fourth-generation X-ray source of a low-emittance beam in the early 2020s [3][4]. With this project, SACLA has been given an additional role as an injector into the new SR with its high-quality beam. Omitting the existing linac and 8-GeV booster synchrotron will also contribute to energy saving.

In terms of the control system, a large-scale overhaul for the SPring-8-II project is needed [5]. Currently, local evolutions exist in SPring-8 and SACLA, and in SPring-8 in particular, which has a long operation history, patches and roundabouts have accumulated.

This paper focuses on the database aspect. The rest of the paper is organized as follows. First, we describe the role of the database system at the site. Then, for the two main topics, a log database and parameter database, we describe the current status and its problem, followed by the new scheme we plan to replace it with. Before concluding, we show a transition strategy for the running facility. The SXFEL accelerator complex was selected as the first migration test location this summer.

DATABASE OF SACLA/SPRING-8

From the early design stage of the site, the database was set as the core technology of the control system [6]. The role of a database can be divided to three parts. First, the log database stores time-series data that are sent from each subsystem. Log data are mostly point signals, although some 1D and 2D data are also included. Next, the parameter database provides an intermediate location between applications. An example is the alarm system where a monitoring process goes around log data and keep the judges in the database, and a display process responds to them. The final part is used for management. For example, it manages the signal names and corresponding information.

LOG DATABASE

Current Scheme

There are three different systems currently running at this site in terms of data logging. Figure 1 shows a schematic of these systems.

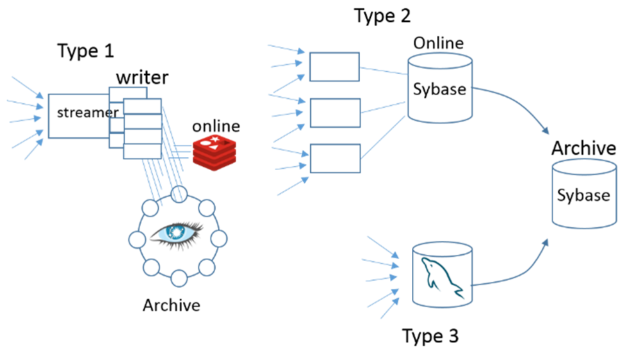

Figure 1: Current log data streams.

Type 1 [7]:

Data are stored in a NoSQL database (Cassandra). Multiple writer processes are connected to the NoSQL database, and all frontend equipment send data through the streamer. The connection between the equipment and streamer is based on ZeroMQ PUSH/PULL mode. To help with quick access to the data, an in-memory database (Redis) is running to keep the most recent data.

[†] email address: k.okada@spring8.or.jp

Type 2 [6]:

Data are stored in two stages of relational databases (RDBs) (Sybase). The second stage is for archiving after a decimation process reducing the data volume. Several processes collecting data (CCs) are connected to the first stage RDB. A CC and other equipment communicate using the RPC protocol.

Type 3 [8]:

This type is utilized in combination with Type 2. Here, the first stage is another RDB (MySQL). Frontend equipment is connected to the database using the MySQL C API, and sends data individually. This was introduced to handle 60 Hz shot-by-shot data. The second archiving stage is the same as Type 2.

Table 1: The Volume of Online Log Data for Each Type. The Data Acquisition Rate in Type 3 Shown Here is the Maximum Case. "SXFEL" in the Table Indicates the Accelerator for SXFEL Beamline.

System	Type	#signals	Cycle (s)	Rate
			max, min	kHz
SPring-8	1	30000	60, 1	9.0
SACLA	2	57000	60, 1	18.9
SACLA	3	1900	1/60, 1/60	114
SXFEL	2	4600	60, 1	1.6
SXFEL	3	160	1/60,1/60	9.6

Concept of the Next Scheme

Clearly, it is not an ideal situation to maintain multiple systems, especially when we plan to use SACLA as an injector to the next SPring-8-II storage ring.

None of these systems is superior to any other, nor can they cover future demand without an upgrade. We decided to build a new system based on experience. Figure 2 shows a schematic view of the future logging system.

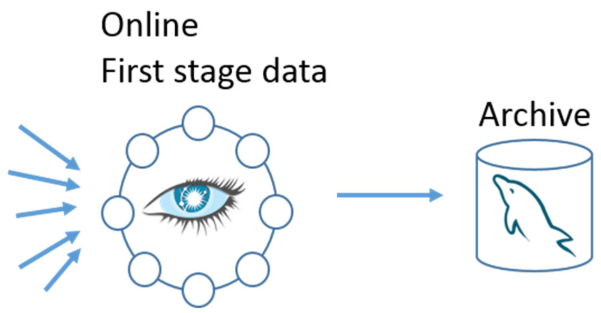

Figure 2: New logging system.

With this system, frontend equipment communicates with the management database directly and knows what log data and with what cycle to be taken, as opposed to the current system, which utilizes an intermediate text file.

These frontends connect to a NoSQL database (Cassandra). This scheme allows us an easy scaling for an increase in the number of signals. However, compared to a Type 1 system, we sacrifice flexibility in the choice of the frontend. If the demand for a new frontend platform arises, it will be necessary to build a new database handler fit to it. Thus far, Solaris and Ubuntu are available. Herein, the Redis online system in Type 1 for quick access to the most recent values is also omitted for simplicity. Instead, another database area is prepared on the same Cassandra to keep the most recent values of all signals. Because such recent data are mostly on Cassandra's caching stage in memory (called a memtable), the access speed is proved to be sufficient for a fast scanning purpose, such as an alarm system. However, there is the potential that we may reconsider such an in-memory database to deal with an intelligent on-the-fly analysis.

The next and final destination is archive storage. The archiving process is planned to skim data points and reduce the volume by about a factor 200 for 60 Hz data, or a factor ≤5 for normal polling data.

Storage Format

Table 2 summarizes the role of each database in the new system.

The Cassandra system stores data for at least one term of accelerator operation (~1/3 of a year). Because Cassandra or a NoSQL database in general has only minimum features for accessing data, an online area is prepared, in addition to the main storage area. The data in the online area are referenced to obtain the latest values, and an entry per day for each signal is used as an index to search the main area for a long period of multiple days.

The archive database used is MariaDB. Background processes skim the data in the first stage and put them into the archive. To save storage space, up to 60 consecutive data points are packed. The number here is a compromise between the packing factor and the access speed for the probable search period.

Table 2: The Role of Each Database

Online / index use	
DB	Cassandra 2.x (replication factor = 3)
Contents	Signal_id, time, (event#), V_{latest}
Key unit	Date+signal_id
Role	Keep the latest value of each signal. An index of date where data written.
1st stage	
DB	Cassandra 2.x (replication factor = 3)
Contents	Signal_id, time, (event#), V
Key unit	Date+signal_id
Role	Store data for one term (~1/3 year). Managed by monthly unit.
Archive	
DB	MariaDB 10.1.x
Contents	Signal_id, time, (event#), [[Δt, V],…]
Unit	1 signal/table
Role	Store archive data after skimming. Data packed by up to 60 points.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUBPA03

Access to Data

Some basic access such as browsing time-series data, or plotting signals to see a correlation, can be achieved on a web browser. A set of CGI functions was written in Python.

A database access library written in C is also provided for dedicated analyses. Example applications include GUIs for various operations.

The sources are under librarian control, and thus the database is protected from dangerous operations.

PARAMETER DATABASE

There are various occasions for keeping and retrieving variables. Herein, we describe items other than an alarm system, including the calibration parameters of different equipment, the DAC setting of magnet currents, and results of geometry survey. These demands can be summarized as having common features. The goal of the parameter database scheme is to build a platform to store variables in a standardized format. The design includes RDB tables and access functions.

Problems in the Current Scheme

There has been no movement to formulate the usage of this parameter treatment. Over a long operation history, every time a new request arises, a new RDB table is created and the corresponding access functions are made. As a result, ~50 types of RDB tables and ~250 access functions have been accumulated. This situation increases the maintenance costs. To make matters worse, the reluctance to add another form, and few chances to update a large-size library, allow users to take an ad hoc approach, which leads to additional mess and personalization problems.

Requirements

From an assessment of the use cases, we narrowed down the usage to the following key features.

1. group management
2. version management
3. parent and child groups
4. access to identification (ID) number and relation between IDs

1. To limit the number of functions handling the data, a common structure is applied to the data tables. Group management is used to point to a proper set of RDB tables.
2. One of the main reasons to use a database is to keep a record of the updates and restore a past record upon request. This requires a rule to tag a set of values at a certain time with the version number and an associated comment.
3. There are cases in which sets of parameters are all in the same category, but the frequencies of the updates are significantly different. By defining a tree relation between groups, we can store them in different groups separately, and load them together.
4. We realized that it is not practical to allow all parameters to belong to "eqip_id," which is a unique identification number of the equipment. To allow the use of different ID numbers, we set the management method to handle the ID numbers and the relation between two IDs.

Design

For these requirements, we designed a standardized platform. Figure 3 describes the concept of this parameter set operation scheme. Management tables take care of the parameter set groups, types of ID numbers, and their relations. Groups can be associated, and versions are kept in one place. Each group has a storage space where the values are kept with the ID number and key. The users access the workspace.

A set of data tables is assigned to each group. We took this structure because, with one big data table design, the access speed may suffer and some disastrous operations affecting all data may occur.

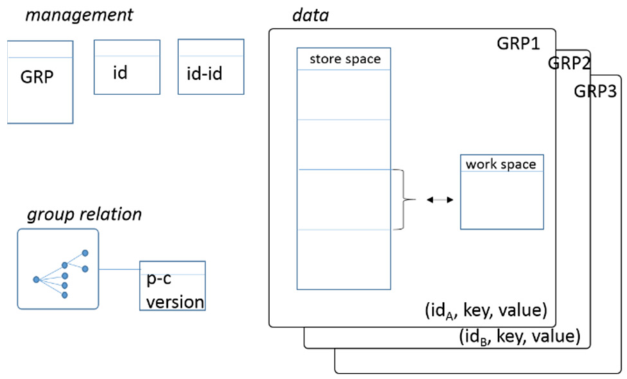

Figure 3: Structure of parameter set on RDB. "p-c version" maintains the relation between the parent and child groups. Applications access to the work space and a set of values is saved to, or loaded from the store space.

Items not Covered

However, we gave up several features and allowed them to be taken care of in the upper layer.

1. Handling a combination key.
2. A mechanism to prevent building a triangle relation between IDs.
3. A safety mechanism to prevent a loop in the parent-child relation.
4. Speed.

Regarding point #4, compared to a custom-made table and function pair, this scheme with key-value type storage and some management tables requires more steps to obtain the same data. However, because most cases using this system are for record tracking of operations, it is rare for very fast responses to be required.

Parent-child Group Inheritance Scheme

Because we plan to use SACLA in multi-mode operation where on-demand injection to SR is required while delivering multiple XFEL beamlines, RF components and pulse magnets should be programmed for each 60 Hz bunch cycle. For this type of situation, a feature of a parent-child group inheritance scheme is implemented herein. Figure 4 shows the concept of this scheme. The parent group (p) stores the common settings in all operation modes. The children (e.g., c1 for beamline-1, and c2 for beamline-2) keep uncommon keys. When users access the data through a child group, there is no need to be aware of which values belong to the parent. Changes applied to common keys are stored in the parent group, and are reflected properly to other child groups.

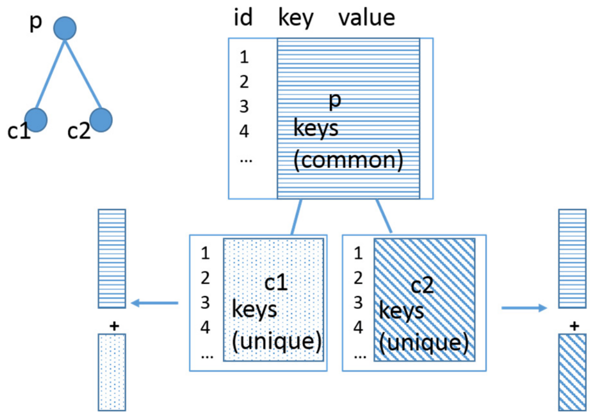

Figure 4: Concept of parent-child inheritance scheme. When a parent-child relation is set using inheritance mode, users accessing a child group do not need to be aware of the common part. Changes are properly reflected to another child group.

Application to SP8 SR Magnet Operation

We have started to rearrange the current parameter treatment. Figure 5 shows the design of a magnet operation. There are three categories under the corresponding IDs, and two id-id connections are defined. For the series-magnets, multiple element_ids are associated with a single eqip_id corresponding to the power supply.

TRANSITION PLAN

Requirement

Because the complex at the site is an operating facility, a blackout period for the upgrade has to be minimized. Because much of the blackout period should be assigned to the physical construction, the control system including the database operation described in this paper needs to be ready ahead of the currently expected period of the early 2020s.

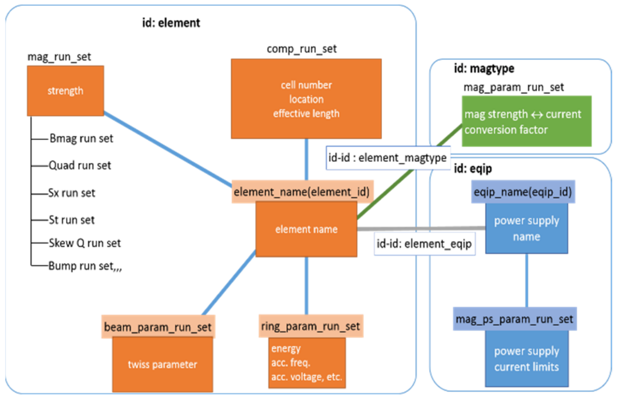

Figure 5: Design of SP8 SR magnet operation. Three boxes correspond to three different IDs. They are connected with id-id relation tables.

Wrapper Functions

It is impractical to build an entire system from scratch because we already have a large number of running applications with 20 years of operation. The method that should be applied to migrate the current system is shown in Fig. 6. Wrapper functions for the existing access functions directly connected to various database tables will take care the changes to the database scheme. In addition, some complicated functions for reading intermediate text files, or files with hard-coded values, will be cleaned up.

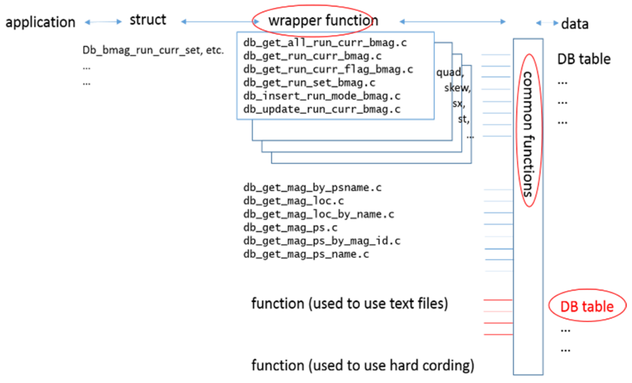

Figure 6: Migration of the current applications. Wrapper functions take care of the changes to the database scheme.

Accelerator for SXFEL Beamline in 2017

Because the size of the SXFEL accelerator is compact and the accelerator is rather independent, we chose it as the first target of the transition.

We took advantage of a week-long shutdown in May for a pilot test. Several VME frontends were replaced using the new data acquisition process, and several operation applications were tested. About 40% of all log data and one-half of the operation applications were tested.

This summer, the entire control system of the SXFEL accelerator was replaced. Along with other summer con-

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUBPA03

struction activities, and with the need of a restoring point, dedicated planning is required.

As of now, the system is operating under the new control scheme. Figure 7 shows the log data accumulation that has occurred since September. The machine and software specifications are provided in the Appendix.

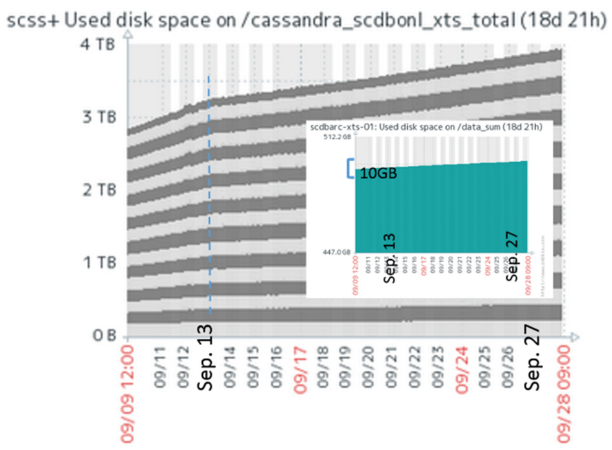

Figure 7: Log data accumulation using the new system on the SXFEL accelerator. A total of 18 nodes of the Cassandra database server are shown on top of each other. Inset: Archived data accumulation after data reduction. Additional data points from the SPring-8 SR trial before September 13 are included.

CONCLUSION

Toward fulfilling the SPring-8-II upgrade plan, we started the design of an accelerator control system. In the upgrade project, the new storage ring and SACLA will be operated synchronously. Although the current scheme had supported operations for almost two decades, we decided it was better to overhaul it for many reasons.

In 2017, the control system of the SXFEL accelerator was successfully replaced with the new scheme as the first application.

APPENDIX

Hardware and Software Specifications for SXFEL Accelerator

Table 3 shows the specifications of the databases used in the SXFEL accelerator control system. We plan to improve the redundancy of the scheme at a later date.

Table 3 Specifications of the SXFEL system in 2017

Online / index use / 1st stage	
H/W	HPCT R324s (*18 nodes)
CPU	Intel(R) Xeon(R) CPU E3-1241 v3 @ 3.50GHz 2core×HT
Memory	32GB
Storage	SSD 1TB
OS	CentOS7.2
DB	Cassandra 2.2.x
Parameter / Archive	
H/W	Dell Power Edge R530
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz 10core×2socket×HT
Memory	128GB
storage	SSD1.2TBx6 (RAID1)
OS	Redhat 6.8
DB	MariaDB 10.1.x

REFERENCES

[1] T. Ishikawa *et al.*, “A compact X-ray free-electron laser emitting in the sub-ångström region,” *Nature Photonics* 6 (2012) 540.

[2] K.Togawa *et al.*, “A soft X-Ray-Free-Electron Laser Beamline of SACLA,” in *Proc. IPAC’17*, Copenhagen, Denmark, 2017.

[3] H. Tanaka *et al.*, “SPring-8 Up-grade Project,” in *Proc. IPAC’16*, Busan, Korea, 2016.

[4] SPring-8-II Conceptual Design Report (2014). `http://rsc.riken.jp/pdf/SPring-8-II.pdf`

[5] T.Fukui *et al.*, “Status of the Control System for the SACLA/SPring-8 Accelerator Complex,” in *Proc. ICALEPCS’17*, Barcelona, Spain, 2017.

[6] R.Tanaka *et al.*, “The First Operation of Control System at the SPring-8 Storage Ring”, in *Proc. ICALEPCS*’97, Beijing, China, 1997.

[7] A. Yamashita, M.Kago, “MADOCA II Data Logging System Using NOSQL Database For SPring-8”, in *Proc. ICALEPCS’15*, Melbourne, Australia, 2015.

[8] M. Yamaga *et al.*, “Event-Synchronized Data-Acquisition System for SPring-8 XFEL,” in *Proc. ICALEPCS’09*, Kobe, Japan, 2009.