EPICS Data Archiver at SSRF Beamlines (Post-print)

Authors: HU Zheng, MI Qing-Ru, ZHEN Li-Fang, LI Zhong

Date: 2023-06-18T00:00:00+00:00

Abstract

The control system of SSRF (Shanghai Synchrotron Radiation Facility) is based on EPICS (Experimental Physics and Industrial Control System). The storage of operational data for synchrotron radiation facilities is crucial for status monitoring and analysis. At SSRF, operational data were previously stored as index files recorded by the traditional EPICS Channel Archiver. However, index files are unsuitable for long-term maintenance and pose challenges for data analysis. Currently, the RDB Channel Archiver and MySQL are employed for archiving SSRF beamline operational data, thereby enhancing data storage reliability and usability. The implementation of a novel uploading mechanism for the RDB Channel Archiver has improved its write performance. Additionally, a web-based GUI (Graphical User Interface) has been developed to facilitate database access.

Full Text

Preamble

ChinaXiv Cooperative Journal
NUCLEAR SCIENCE AND TECHNIQUES 25, 020103 (2014)

EPICS Data Archiver at SSRF Beamlines

HU Zheng (胡正),[1],[2] MI Qing-Ru (米清茹),[1] ZHEN Li-Fang (郑丽芳),[1] and LI Zhong (黎忠)[1],*
[1]Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China

(Received August 23, 2013; accepted in revised form November 26, 2013; published online March 20, 2014)

The control system of SSRF (Shanghai Synchrotron Radiation Facility) is based on EPICS (Experimental Physics and Industrial Control System). Operation data storage for synchrotron radiation facilities is crucial for status monitoring and analysis. At SSRF, operation data were previously recorded as index files by the traditional EPICS Channel Archiver. However, index files are not suitable for long-term maintenance and are difficult to analyze. Currently, RDB Channel Archiver and MySQL are employed for SSRF beamline operation data archiving to enhance data storage reliability and usability. By implementing a new uploading mechanism in RDB Channel Archiver, its writing performance has been improved. Additionally, a web-based GUI (Graphical User Interface) has been developed to facilitate database access.

Keywords: Database, RDB Channel Archiver, Markov Auto-Complete, EPICS, SSRF
DOI: 10.13538/j.1001-8042/nst.25.020103

Introduction

Shanghai Synchrotron Radiation Facility (SSRF) is capable of supporting numerous beamlines [1] and experimental stations for physical, chemical, and biological research. For each beamline, over 200 signals providing information on vacuum status, temperature, cooling water flow, and other parameters must be monitored and archived accurately and reliably. This requirement also applies to SSRF accelerators, resulting in a substantial number of signals that need to be archived.

At SSRF, the control system software is based on EPICS, and the Channel Archiver is used to archive operation data. The RDB Channel Archiver represents a newly upgraded version of the Channel Archiver. Implemented in Java, it utilizes relational databases to provide superior data storage compared to the original index files. MySQL, Oracle, and PostgreSQL are the three relational database systems supported by RDB Channel Archiver. In practice, RDB Channel Archiver communicates with databases via JDBC (Java Database Connectivity) to ensure reliable connections, and it supports any database that provides a Java API.

II. EPICS Structure

EPICS is a set of open-source software tools, libraries, and applications widely used to create distributed soft real-time control systems for scientific instruments [2]. As shown in Fig. 1 [Figure 1: see original paper], EPICS primarily consists of three components: IOC (Input/Output Controller), OPI (Operator Interface), and CA (Channel Access) [3]. CA is a set of communication protocols based on TCP/IP that enables multiple clients to access multiple servers. A fundamental data unit in CA is the PV (Process Variable), which typically represents a signal. To establish a data access channel, a client broadcasts across the network for a targeted PV, and the IOC containing that PV responds by

establishing a channel. This enables the client to read values from the PV, write values to the PV, and monitor the PV.

III. RDB Channel Archiver and Its Improvement

A. Structure of RDB Channel Archiver

The RDB Channel Archiver has been implemented to provide effective data archiving for SSRF beamlines. It is a major component of CCS (Control System Studio), an open-source toolset developed under Eclipse by Kay Kasemir from Oak Ridge National Laboratory in the USA [4]. With the primary objective of transmitting PV data from IOCs to a relational database, RDB Channel Archiver uses JCA to connect to EPICS IOCs. JCA is a Java Channel Access protocol that works seamlessly with the original EPICS Channel Access [5]. Although its operation mechanism—reading and storing data through periodic scanning or PV monitoring, as illustrated in Fig. 2 [Figure 2: see original paper]—is similar to the traditional Channel Archiver, all data access in RDB Channel Archiver utilizes interface layers, with implementations provided in separate plug-ins [6]. This design makes it easily compatible with different databases.

RDB Channel Archiver supports Oracle, PostgreSQL, and MySQL, all of which are full-featured database management systems. We selected MySQL for SSRF beamlines because Oracle requires a paid license, while PostgreSQL's performance is not as high as MySQL's. Although MySQL has some functional limitations, these do not affect our data archiving requirements. The database E-R diagram is shown in Fig. 3 [Figure 3: see original paper], where the tables `smpl_{eng}`, `chan_{grp}`, and `channel` represent PV channel information and configuration, while the tables `sample` and `array_{val}` store the values of archived PVs.

B. Improvement of RDB Channel Archiver

In the original RDB Channel Archiver, a single write thread scans sample buffers and sends data to the database. At SSRF, we have implemented an additional write thread and an uploading controller called the uploader. The improved operation mechanism allows the two write threads to work alternately with the buffers, while database access is controlled by the uploader. The uploader manages the upload method for each write thread by incorporating their upload methods into a Java synchronized function, thereby preventing conflicts when both write threads upload simultaneously.

As shown in Fig. 4 [Figure 4: see original paper], components drawn with solid lines represent original RDB Channel Archiver components, while dashed-line components are our additions. The 'SQL' symbol in Fig. 4 [Figure 4: see original paper] indicates the upload SQL query sent to the database. Since preparing an SQL query typically consumes less time than executing it, uploading becomes a

more continuous process and achieves greater efficiency when two write threads work together.

The upload performance of both original and improved RDB Channel Archivers was tested using various write periods and batch sizes, which are key parameters of the RDB Channel Archiver. The database server was an IBM System x3850 running Red Hat Enterprise Linux Server release 5.3 (Tikanga). Simulated PVs were used to test both archivers at a scanning rate of 2000 values per second, which approximates a high operational load at SSRF. The write speed of the improved RDB Channel Archiver was compared with the original version using batch sizes of 500, 1000, 1500, and 2000 samples over write periods of 5 s, 10 s, 15 s, and 20 s. The results are shown in Fig. 5 [Figure 5: see original paper]. On average, the uploading performance improved by more than 10%.

IV. Web-Based GUI

A. Framework of the GUI

A web-based GUI (Graphical User Interface) was developed for SSRF beamline users (Fig. 6 [Figure 6: see original paper]). Users can access the database from any networked computer via a web browser without installing additional applications, providing excellent accessibility and convenience.

The server was deployed on a LAMP stack, which works effectively with the MySQL database. The interface is entirely driven by PHP, JavaScript, and HTML, with chart elements based on the Highcharts library, which offers better compatibility and speed than Flash.

The web-based GUI consists of four main parts. Part I is the archived PV chart, which includes two sub-charts: a master chart displaying a large time scope to indicate data availability, and a detailed chart showing the selected time span from the master chart with basic zoom and pan capabilities. Exact PV data for corresponding timestamps can be displayed in tooltips when users hover over the detailed chart, and the detailed chart can be exported to a file and saved to a desired location.

Part II provides a live PV chart that refreshes every second, showing current values of a specified PV. This chart displays 20 values representing the last 20 seconds for that PV. Delay time measured in milliseconds is also shown as text; if the delay exceeds 20 minutes, the text indicates that no live PV data is available.

Part III is a PV selection area that enables users to select PVs by attributes such as name, beamline region, and monitor types. When these attributes are specified, facilities with corresponding PVs are displayed as options (Fig. 7 [Figure 7: see original paper]). Users can select a specific PV by clicking a radio button. These radio buttons are not pre-deployed but are dynamically generated based on user input and database queries.

Part IV is an input box for directly typing PV names, serving the same purpose as Part III.

B. Markov Auto-Complete and Its Implementation

A novel Auto-Complete (or suggestion) method has been implemented for the GUI input box. Unlike conventional Auto-Complete that provides a set of suggested words once users type a few letters, this method provides suggested fields for all possible words at each step until all fields compose the complete suggestion. For example, to select the PV "X14W:FE:FM1:TC1:AI", users type "X" and select "X14W:", then select "X14W:FE:", "X14W:FE:FM1:", and finally "X14W:FE:FM1:TC1:AI" to obtain the target PV. All user selections are automatically displayed in the input box, and users need only press 'up' or 'down' arrows and 'enter' to complete the selection. This process is illustrated in Fig. 8 [Figure 8: see original paper]. This approach offers significant advantages over conventional Auto-Complete, particularly when too many suggestion words start with the same letter (e.g., "X"). In essence, the complete word is divided into several parts and treated as a Markov chain.

V. Conclusion

RDB Channel Archiver represents a promising upgrade for Channel Archiver, offering a mature solution for long-term operation data storage with improved accessibility. The upload mechanism of RDB Channel Archiver has been remarkably improved at SSRF beamlines. A web-based GUI featuring a novel user input method ("Markov Auto-Complete") has been developed to provide enhanced reliability and usability. Users can now easily and quickly access operation data from any networked computer via a web browser.

References

[1] SSRF Home Page. http://ssrf.sinap.ac.cn/, September 10, 2013.
[2] EPICS Page. http://www.aps.anl.gov/epics/, September 10, 2013.
[3] Dalesio L R, Kraimer M R, Kozubal A J. EPICS architecture, in Proc ICALEPCS. 1991, 91: 92–15.
[4] Kasemir K, Gabriele Carcassi. Control System Studio Guide. http://cs-studio.sourceforge.net/docbook/, September 10, 2013.
[5] JCA Home Page. http://jca.cosylab.com/, September 10, 2013.
[6] Kasemir K. Control System Studio (CSS) Data Browser, in Proc PCaPAC08, Ljubljana, Slovenia, 2008, pp. 99–101.
[7] del Campo M, Giacchini M, Giovannini L, et al. EPICS HyperArchiver: initial tests at ESSBilbao, in Proc. IPAC2011, San Sebastián, Spain, 2011, pp. 2343–2345.

Note: Figure translations are in progress. See original paper for figures.