THE STUDY OF ACCELERATOR DATA ARCHIVING AND RETRIEVING SOFTWARE

Yusi Qiao[1], Ge Lei, Zhuo Zhao, Institute of High Energy Physics, Beijing, China
[1]also at University of Chinese Academy of Sciences, Beijing, China

Abstract

This paper presents a novel archiving and retrieving software designed for BEPC-II and other particle accelerators. At BEPC-II, real-time data are stored as index files recorded by traditional EPICS Channel Archiver. Nevertheless, index files are not suitable for long-term maintenance and difficult for data analysis. The NoSQL database MongoDB is used for this new system due to aging technologies, so as to promote the data storage reliability, usability, and possible future advanced data analysis. A cross-platform UI (User Interface) has also been developed to make it quicker and easier to access the database. The writing and query performance are tested for this software.

INTRODUCTION

The Beijing Electron-Positron Collider II (BEPC-II) consists of various equipment. Capturing live BEPC-II data is important for its status monitoring and post mortem analysis. These signals provide status information for power supplies, RF devices, vacuum, beam diagnostics, timing system, etc. which should be monitored and archived accurately and reliably. At BEPC-II, the control system software is based on the Experimental Physics and Industrial Control System (EPICS). EPICS provides a set of Open Source software tools, libraries and applications. It is widely used to create distributed soft real-time control systems for scientific instruments [1]. One of the EPICS tools, Channel Archiver, has been used in the present BEPC-II data archiving system, but it has issues of capability, extensibility, data migration and so on. To deal with the problem, we proposed a novel archive system using MongoDB, a document-oriented NoSQL database. The new database is required to have high availability, high performance, and high flexibility of storage expansion.

ARCHIVING TOOLS ALTERNATIVES

For comparison purpose, we did some research on several mainstream archiving tools of particle accelerators.

Channel Archiver records data from several channels, each producing samples at a different rate. The data is stored in binary index and data files. The designing goal of Channel Archiver is I/O speed, and the retrieving tool is easy to utilize [2].

With matured relational database (RDB) technologies, such as MySQL, Oracle and PostgreSQL, many new archiving tools are developed and adopted based on RDB. These tools significantly improve data access and retrieval performance comparing with the original indexed file based Channel Archiver. However, relational database is not enough in availability, performance, and flexibility for increasing data volume, which becomes a great limitation when facing big data and various data structure nowadays.

NoSQL databases are developed with the challenge of mass data storage and processing, as well as high performance, especially in large scale and high-concurrency applications. MongoDB is a high performance and very scalable document-oriented database that stores data in a BSON format, a dynamic schema document structured like JSON [3]. With powerful query language and high-speed access to massive data, MongoDB is used as the underlying storage database of our system.

ARCHITECTURE

The system has two main subsystems, the archiving tool and retrieving tool, as shown in Fig. 1. PV (Process Variable) is an important data unit of EPICS communication protocols, which usually represents a signal. Now, about 8000 PVs are being collected for monitoring and controlling of BEPC-II equipment. To establish a data communication channel, engine model broadcasts across the network for the targeted PV, and the IOC (Input/Output Controller) holding the PV will response to the request and then establish a communication channel, which allows reading values from the PV. The values and other parameters received from read threads will be stored in a buffer temporarily and then bulk-write to a MongoDB database. With an appropriate data structure and indexes, the data can be stored efficiently and located quickly through retrieving model, and the users can display the data using a convenient and efficient UI.

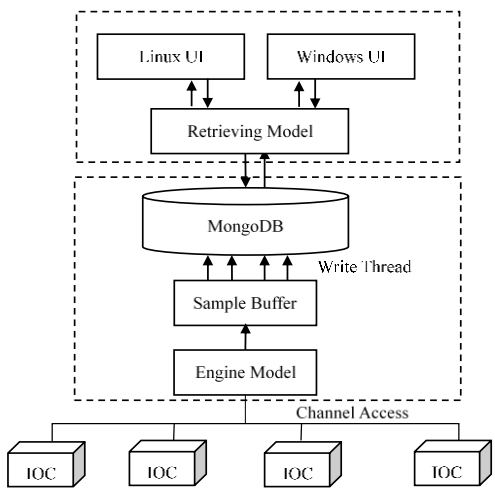

Figure 1: Architecture of system.

ISBN 978-3-95450-182-3

DATA ARCHIVING

The configuration file is in XML format, including archive engine name, which is also the database name in MongoDB, and targeted PV names with corresponding sampling rate. There are several read threads and one write thread. The read threads read the data via periodic scanning the PV, and send it to the buffer. Two buffers are applied for this archive tool, which operate alternatively. When a buffer reaches a certain volume, the write flag calls the write thread and the other buffer would be ready to store the next coming data.

By applying the bulk-write mechanism, the writing performance has been improved. For real-time data archiving, PV scanning rates are various but no faster than 1 Hz. In our test, we run two IOCs which contain 1600 PVs for archiving, with scan periods among 1 s, 2 s, 5 s and 10 s. The archive server deploying MongoDB is built with an HP workstation, a quad-core, eight threads, clocked at 3.4 GHz Intel CPU is equipped, along with 8GB memory and a 500 GB Western Digital disk. The maximum data transmission speed is 126 MB/s.

The data must be transformed into a BSON object to store in MongoDB. On average, a BSON document consumes about 92 bytes per sample to store the value, timestamp, status and severity. So based on existing devices, 6×10^4 to 6×10^4 documents were inserted per second, and the I/O throughput of archive engine is 6MB/s - 9MB/s. Because of the high speed processing and reasonable compression rate, we choose Snappy as compression and decompression library [3]. Each database of the archiving system consists of two kinds of collection: the collection named colsum for recording collections' names and their time spans and the collection contains raw data. This structure makes the data easy to classify and query. Once a query is submitted to database, the system first finds in the colsum collection according to the engine name, PV name and time span, and then find the requested data from the collection contains raw data. The query only loads the value and timestamp because in most cases users only use the data to plot.

DATA RETRIEVING

Indexes offer enhanced performance for read operations. These are useful when documents are larger than the amount of RAM available [4]. Indexes are defined when the collections are created. Queries that return results containing only indexed fields are called covered queries. These results can be returned without reading through the source documents. Based on the architecture and data structure, we designed appropriate indexes to make sure that only covered queries are called for better performance. While indexes will optimize system performance and scalability, they incur associated overhead in write operations, disk usage, and memory consumption to a certain degree. For now, we mostly focus on the retrieval performance.

The Table1 shows the data retrieval performance. The response time (R Time) represents the time a query request consume. The process time (P Time) represents the time from the find button clicked till the data is ready to plot, which includes converting EPICS Time stamp to local time. Since the mostly queried time span is less than a day [5], we did a performance test retrieving a day's worth of 1 Hz double data from 15 days of data and 30 days of data. That means finding 86.4 thousand samples from 1.296 million samples and 2.592 million samples.

Table 1: Query Performance

Condition	15d		30d	
	R Time	P Time	R Time	P Time
No Index	0.432s	1.289s	0.808s	1.749s
Cover Index	0.047s	0.866s	0.056s	0.896s

USER INTERFACE

A cross-platform GUI (Graphics User Interface) based on Qt has been developed for users. Users can access the database easily on Linux or Windows system. It is driven by C++, and the plot elements are based on Qcustomplot library. Figure 2 shows four major parts of the GUI. Part I is a connection dialog for server database. It can fulfil the connection table automatically via logging the past successful connection information. Part II is a PV selection area. It helps users to find the PV name quicker and easier by providing auto-filter function. It gives a set of suggested words that contain the input word. Comparing with the usual auto-complete, users do not need to know the initials or the first accurate part of the PV name. It makes user easier to find a specific PV because the PVs are often named regularly or with meaningful tags which are easy to remember.

Part III is for determining the time span. User can enter time or just click in the calendar. Part IV provides a plot area that shows the extract PV data with corresponding readable time stamp in line chart with scatter. User defined legend and title, three zoom modes using the mouse wheel are also provided. The plot and scatters can be exported, dragged or selected to see the details of each point. One more useful function is to allow users to export queried data in to a TXT file. This is helpful for performing off-line analysis.

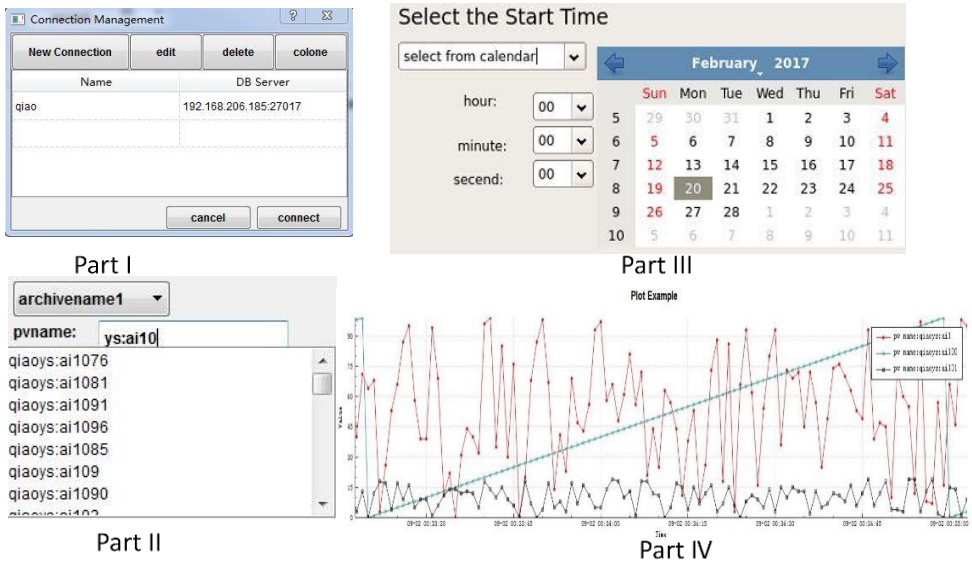

Figure 2: Main parts of retrieving user interface.

CONCLUSION

A novel archiving and retrieving system has been developed using MongoDB, a NoSQL document-oriented database. The system provides a new archive engine to archive EPICS records to MongoDB and a user-friendly interface to query the data. The simple UI allows users to plot the stored data or export for off-line analysis. On the basis of the existing laboratory test results, the system has been deployed and being tested in the BEPC-II runtime environment at present. A web-based query interface is also being developed as a future plan.

REFERENCES

[1] EPICS Home Page, http://www.aps.anl.gov/epics/.

[2] *Channel Archiver*, Aug. 2006, pp. 3-14; http://icsweb.sns.ornl.gov/kademir/archiver/manual.pdf

[3] *MongoDB Manua*l, https://docs.mongodb.com/manual/.

[4] Truică C O, Boicea A and Trifan I. CRUD operations in MongoDB., in *Proc. 2013 international Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*, Beijing, China, July 2013, paper AC1210, pp.347-350.

[5] G. Shen, Y. Hu, Marty Kraimer, Shroff Kunal and Dejan Dezman., "NSLS II Middlelayer Services", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, San Francisco, US, Oct. 2013, paper MOPPC155, pp. 467-470.

ISBN 978-3-95450-182-3