Development of A Scalable and Flexible Data Logging System Using NoSQL Databases

M. Kago, A. Yamashita, JASRI/SPring-8, Hyogo, Japan

Introduction
The current data logging system for SPring-8 accelerator powered by a relational database management system (RDBMS) has been storing log data for 16 years. However, the recent improvement and future plans of the SPring-8 accelerator require the data logging system to increase writing performance and to handle large-volume data, but it is not easy to extend the system for the following reasons. Therefore, we developed a new data logging system.

- The RDBMS has no horizontal scalability.
- The data management is complex.
- The data acquisition has no flexibility.

New Data Logging System

New framework was developed. Two NoSQL (Not only SQL) databases, Redis and Apache Cassandra, were adopted to store log data. The data acquisition was designed on the basis of ZeroMQ messaging library and MessagePack serialization library.

- **Scale-out**: The system can easily grow the performance by adding more low-cost servers.
- **High Reliability**: There was no single point of failure (no SPOF).
- **Flexible Data Acquisition**: Users specify the data name and only send log data. Log data supports various data type such as integers, reals, strings, arrays, and maps.
- **Low Latency Access**: Users can take the latest data in microseconds order.

Architecture

The data logging system is designed on a three-layer model. All devices are connected through a network. The event log is sent to several relay servers at the client’s own timing, and is inserted into databases.

Log Generators

- **Event log** sent to several relay servers at the client’s own timing, and is inserted into databases.

Relay Servers

- **ZeroMQ + MessagePack**

ZeroMQ and MessagePack is inserted into databases. Through a network. The event log is sent to several relay servers at the client’s own timing, but it is not easy to extend the system for the following reasons. Therefore, we developed a new data logging system.

- **The RDBMS has no horizontal scalability.**
- **The data management is complex.**
- **The data acquisition has no flexibility.**

The polling system of the server-client model uses tight coupled ONC-RPC and is highly interdependent. Therefore, the data acquisition is limited by the OS and language environments.

Cassandra

Apache Cassandra is utilized for the perpetual archive. Cassandra is an open-source distributed database of the Apache project. Its features include scale-out (see Fig. 1), high write performance, fault tolerance, no SPOF, suitable for time-series data.

![Figure 1: The write throughput when the access load is increased.](image)

Cassandra's consistency provides a few guarantees and is called eventual consistency. (When the data are taken from six node cluster with a replication factor of three, we found that the time required for guaranteeing consistency is about 1 sec). Therefore, real-time data are provided by the cache server.

![Figure 2: The write latency of Redis compared to Cassandra.](image)

Log Generator

Users who want to store the log into this system insert the prepared function into their data acquisition programs. The code packs the messages using MessagePack, and pushes the messages using the Push/Pull pattern the ZeroMQ communication library. The data logging system uses MessagePack for all internal data representation. The Push/Pull pattern sends messages from one sender to several receivers by the round robin algorithm and can easily realize load balancing. When the sender detects a problem with a receiver, the message is sent excluding this receiver. Any platform or programming language that supports ZeroMQ and MessagePack can be a client of this system.

Relay Server

The relay server works as a gateway between the local computer and database. The relay process manages the pull socket for receiving messages from the client and transfers the received message to a writer process. The writer process converts the received message into a database command and inserts the data into Cassandra and the Redis in parallel.

Current Status & Conclusion

We are now migrating from the previous data logging system. The new system has been installed in the actual environment.

In the near future, data acquisition will begin small and the scale will gradually grow. Along with these future plans, we will construct an alarm system using this system and will pursue multi-platform support and web pages.