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 advantages of the RDBMS, such as joining tables , become bottlenecks when scaled out. The general method for improving server performance is to change to a high-spec server , but hardware costs are continuously increasing.
- **The data management is complex.**
  To improve writing performance, a number of log data are placed on one row to reduce the number of SQL statements needed when they are stored in RDBMS. When new data are registered on the RDBMS, a new table has to be created.
- **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.

New Data Logging Sytem

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.

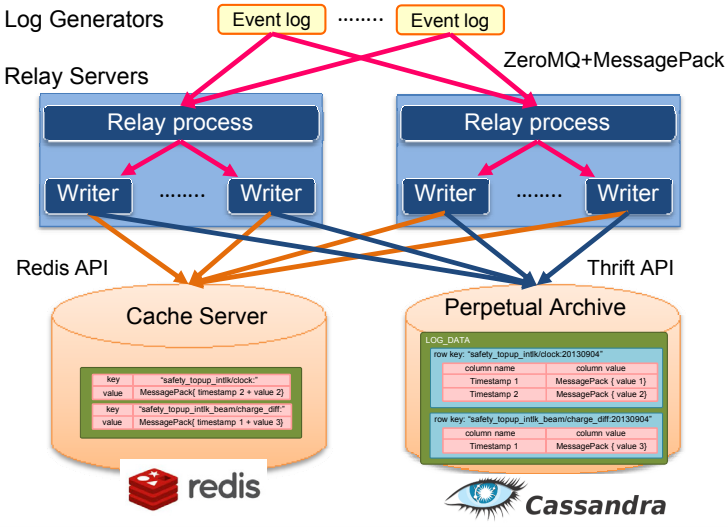
The system features are;

- **Scale-out**
  The system can easily grow the performance by adding more low-cost servers.
- **High Reliability**
  There was no single point of failure (noSPOF).
- **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.

*A **NoSQL** database is defined as a new type database management system that is non-relational. It provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability.

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 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 patten 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.

Perpetual Archive

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.

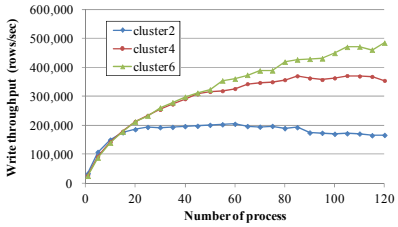


Figure 1: The write throughput when the access load is increased. Hardware specifications of a node are Xeon 2.93 GHz, 8 GB RAM, 64 bit Centos 6.2. Cassandra is version 1.0.5. Data size per row is 20 bytes, and a client inserts 1,000 rows at once.

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.

Cache Server

Redis is adopted for real time data cache. Redis is an in-memory key-value store, and is its support of a various data structures such as list, set, sorted set, and hash. Redis provides the real time data to the control GUI and is used to complement Cassandra's consistency .

Redis works a single process. Therefore, multiple Redis servers are parallelized by an access library that we developed.

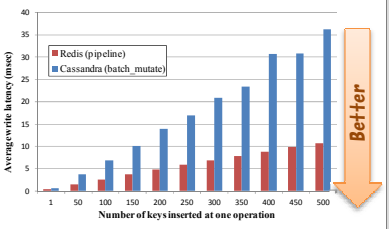


Figure 2: The write latency of Redis compared to Cassandra.

Long-tem Test

The table on the right shows the test parameters. These data are actually generated in SPring-8. The acquisition cycle is faster than the current one.

Testing Parameters

Item	Specification
Hardware	Intel Xeon X3470 2.93 GHz 4 Core CentOS 6.2 64 bit
Redis	Version 2.6.10, Number of process: 4
Apache Cassandra	Version 1.1.5, 6 nodes cluster (replica: 3) OracleJavaVM 1.6.0
Data Acquisition	Number of clients: 240, relay processes: 4, writer processes: 24
Event Log	Number: 47,397, Cycle: 1 Hz Average message size: 60 bytes

Write Test

The writing test was conducted for 3 months.

- ✓ No data loss during the test
- ✓ No impact on write performance even when the server was executed a forced termination.
- ✓ Throughput of one relay process: ~180,000 ops/sec, one writer: ~5,000 ops/sec

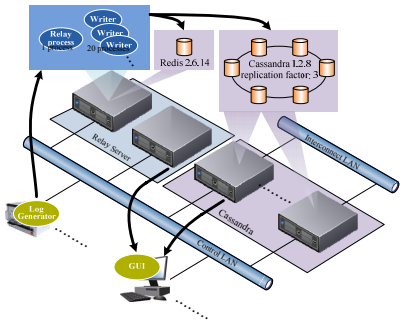
Read Test

The reading test was conducted along with the writing test.

Redis; Read latency of the latest value		Cassandra; Read latency of the data for 1 day (86,400 points)	
Average (1,000,000 events)	0.26 ms	Average (10,000 events)	1.01 sec
Standard Deviation	0.14 ms	Standard Deviation	0.18 sec

Current Status & Conclusion

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



Specifications of Servers

Item	Relay Server	Cassandra Server
Processor	Intel Xeon E5-2430, 2.2 GHz, 12 core	
Memory	16 GB	
OS	CentOS 6.4 64 bit	
Storage 1	SAS 15 Kr/m 450 GB × 2 Raid 1	SAS 15 Kr/m 450 GB × 2 Non Raid
Storage 2	N/A	SATA 7,200 r/m 3 TB × 2 Non Raid
Network	1 Gb Ethernet × 2 ports	
Power Supply	Dual, Hot-plug, Redundant power supply, 550 W	Single, Hot-plug, 550 W

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.