MOPPC130
ICALEPCS2013
San Francisco,USA.

A Message based Data Acquisition System

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

00 Abstract

In SPring-8, we are constructing MADOCA II, the next generation accelerator control framework. It will be installed in the spring of 2014. We describe the part of the data acquisition system of MADOCA II. The old MADOCA data acquisition system was built on the bases of ONC-RPC for communication between embedded processes and data collector.

We designed the new data acquisition system with the long experience on MADOCA. We employ ZeroMQ messages packed by MessagePack for communication. We obtained a high performance, highly reliable, well scalable and flexible data acquisition system.

01 System

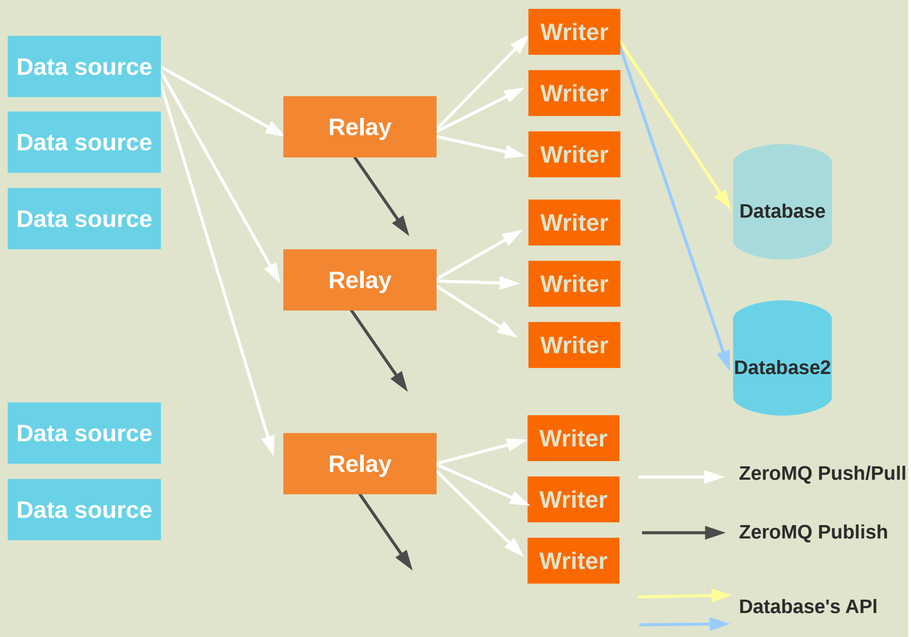
The system is built on the three-tier standard model of the accelerator control system.

The data source on embedded computers send messages to the relay processes on the server computers via Ethernet networks.

The relay processes relay messages to the writing processes which convert messages to database formats and write them to the databases. The relay servers also publish messages to other processes which subscribe to messages by arbitrary keywords.

The writer receives messages, process them and send to the database using the individual API. One relay server sends messages to one or more writers and no cross connections between relays and writers i.e one writer received from one relay servers. In the writer, the receiving thread and the writing threads are separated.

The receiving thread sends messages using in-process publish and subscribe pattern. Currently two writing threads write data to the databases. Relay server and writers has been prototyped by Python and implemented in C++.



02 Messaging

The messaging is asynchronous, one-way and no call back, that enables fast and no-wait messaging. We use ZeroMQ messaging library, which has multiple messaging pattern in multi operating system and multi language environments.

We use ZeroMQ's pipeline (PUSH/PULL) and publish-subscribe patterns with multi-part feature. One message has three parts one is signal name, second is meta-data of the data and the last one is data.
Data and meta-data are packed by MessagePack, a object serialization library which pack scalar, list, map and their combination structure into one binary string.

LGsr_mag_ps_b/current_adc
{tm:1380849642000000000}
1234.32

Structure of one message

One massage consist of three parts.
1st: Two letter keyword+signal name
2nd: Meta-data. Currently it is time stamp only, but extendable.
3rd: data

The two letter keyword express kind of data. For example, “LG” means log data and “AL” means alarm data. One can subscribe specific kind of data using the keyword.

2nd and 3rd data are serialized by MessagePack. Therefore, data type is not limited only scaler but also structured data like list,map and their combinations.

03 High-Performance

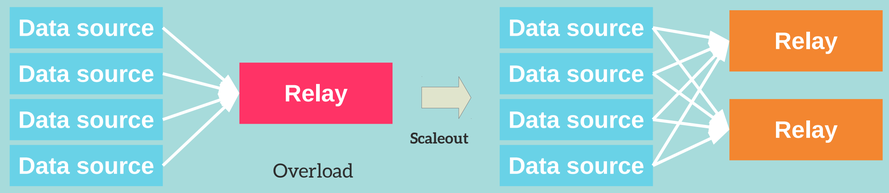
We performed a long term test run and measured its performance. The names of signals are taken from real 47,397 SACLA signals. Simulated data sources generated messages in 1Hz.

Three relay serves and 24 writer processes wrote to six node Apach Cassandra cluster and two Redis servers. The speed of writes are over six times higher than current SPring-8 writing speed

We continued writing tests for 24x7 in three months. We examined Cassandra data and no data were dropped. We also examined Redis data and no time reversal happened in this test term. One relay process itself can handle over 180,000 messages per second at Intel Xeon X3740 2.93GHz processor .

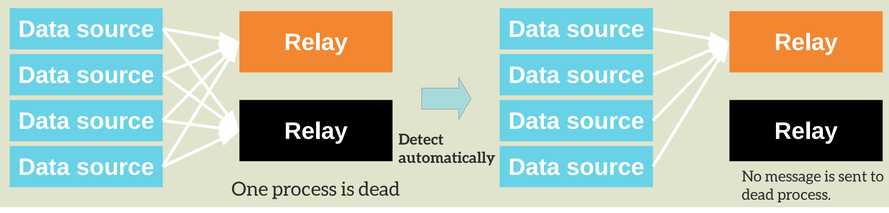
04 Scalable

Data sources send messages to multiple servers in round robin way that enhances the scalability and reliability. The multiple relay server may cause broken the order of the message. The older messages may come after new messages. We observed no order breaking at long run test but it could happen with high density data.



05 Reliable

We examined behavior in the process fails. When one relay process failed, the writer process sent messages to the other relay process after the buffer was filled. After the relay process brings back the messages in the buffer sent to the relay process. Because the size of the buffer was adjustable, we were able to minimize the delay of the message.



06 Flexible

Flexible data format

By using MessagePack, user can send data structure like lists, maps and their combinations without breaking down structure. Also user send just a string. The system can replace syslog system to store system logs into database.

Complex data structure is converted into 69byte string by MessagePack.

```
{'dataA': [1.23, 3.56, 7.65], 'dataB': [9.87, 3.56, 55.22]}
```

```
'\x82\xa5dataA\x93\xcb?\xf3\xae\x14z\xe1G\xae\xcb@\x0cz\xe1G\xae\x14{\xcb@\x1
e\x99\x99\x99\x99\x99\x9a\xa5dataB\x93\xcb@#\xbdp\xa3\xd7\n=\xcb@\x0cz\xe1G\x
ae\x14{\xcb@K\x9c(\xf5\xc2\x8f\\'
```

Flexible data source

The main component of data source consist of ZeroMQ and MessagePack, both are open-source, and run under multi-platform and multi-language environments. So user can choose their platform and language as their like. And the data source can send their messages at arbitrary time.

ØMQ

40+ languages including C, C++, Java, .NET, Python.
Most OSes including Linux, Windows, OS X.

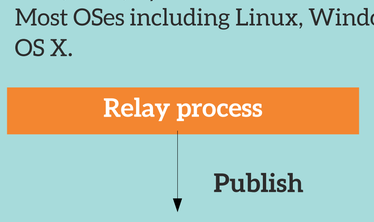
MessagePack

17+ languages including C, C++, Java, C#, Python.
Most OSes including Linux, Windows, OS X.

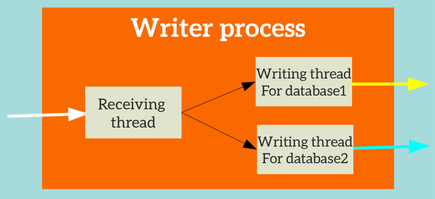
Flexible data output

There are two ways to output message. One from writer process. It is suitable to synchronous output. Another is subscribing from relay server asynchronously.

A writer process have a receiving thread and number of writer threads. The receiving thread sends messages to writer threads with PUBLISH/SUBSCRIBE pattern. If one want to add another output, another thread can be added without modification to receiving thread. User can subscribe message from the relay servers in asynchronous way. User can specify keywords tom subscribe messages.



Relay process publish message. User can subscribe message by header keyword and signal names. For example if user want to see alarm of signal name yyzz, one can receive message by subscribing by “ALyyzz” keyword. It is suitable asynchronous application like Websocket server.



In the writer process, receiving thread publishes message to writing threads. Writing threads are separated and another thread can be added easily. It is suitable synchronous output like databases.

07 Conclusion

We constructed flexible, reliable, scalable and high performance data acquisition system. This system will acquire and store data of SPring-8 during the run in the spring 2014.

For long run test, the new system coexist with the old system. We expect the new system will serve the new generation accelerator like old MADOCA system has been serving SPring-8 for a long time.