HOW CASSANDRA IMPROVES PERFORMANCES AND AVAILABILITY OF HDB++ TANGO ARCHIVING SYSTEM

R. Bourtembourg, J.L. Pons, P. Verdier, ESRF, Grenoble, France

Abstract

The TANGO release 8 led to several enhancements, including the adoption of the ZeroMQ library for faster and lightweight event-driven communication. Exploiting these improved capabilities, a high performance, event-driven archiving system, named Tango HDB++, has been developed. Its design gives the possibility to store archiving data into Apache Cassandra: a high performance scalable NoSQL distributed database, providing High Availability service and replication, with no single point of failure. HDB++ with Cassandra will open up new perspectives for TANGO in the era of big data and will be the starting point of new big data analytics/data mining applications, breaking the limits of the archiving systems which are based on traditional relational databases. The paper describes the current state of the implementation and our experience with Apache Cassandra in the scope of the Tango HDB++ project. It also gives some performance figures and use cases where using Cassandra with Tango HDB++ is a good fit.

INTRODUCTION

The TANGO archiving system is a tool allowing TANGO users to store the readings coming from a TANGO based control system into a database. The archived data are essential for the day by day operation of complex scientific facilities. They can be used for long term monitoring of subsystems, statistics, parameters correlation or comparison of operating setups over time.

To take advantage of the fast and lightweight event-driven communication provided by TANGO release 8 [1] with the adoption of ZeroMQ [2], a novel archiving system for the TANGO Controls framework [3], named HDB++ [4], has been designed and developed, resulting from a collaboration between Elettra and ESRF.

HDB++ design allows TANGO users to store data with microsecond timestamp resolution into traditional database management systems such as MySQL [5] or into NoSQL databases such as Apache Cassandra [6].

APACHE CASSANDRA

Apache Cassandra is an open source distributed database management system available under the Apache 2.0 license. Cassandra's masterless ring architecture where all nodes play an identical role, is capable of offering true continuous availability with no single point of failure, fast linear scale performance and native multi-datacentre replication. All these advantages over the traditional relational databases made Cassandra a very good candidate to store the historical data coming from the ESRF TANGO control systems.

HDB++ DESIGN

HDB++ is a novel TANGO Controls archiving system. Its architecture is mainly based on the two following Device Servers and takes advantage of the TANGO events mechanism:

- The Event Subscriber Device Server subscribes to TANGO archive events, which are ZeroMQ events in the latest TANGO releases, and stores the received events in the historical database. It provides diagnostics data as well.
- The Configuration Manager Device Server configures the attributes to be archived and defines which Event Subscriber is responsible for a set of TANGO attributes to be archived. It provides diagnostics data as well.

An abstraction library named *libhdb++* decouples the interface to the database back-end from the implementation.

To be able to store data into Apache Cassandra, a C++ library named *libhdb++cassandra*, implementing the methods from the *libhdb++* abstract layer for Cassandra, has been developed at the ESRF. It was inspired by the work done by Elettra who implemented the HDB++ MySQL back-end support shared library. The HDB++ Device Servers design is shown in Fig. 1.

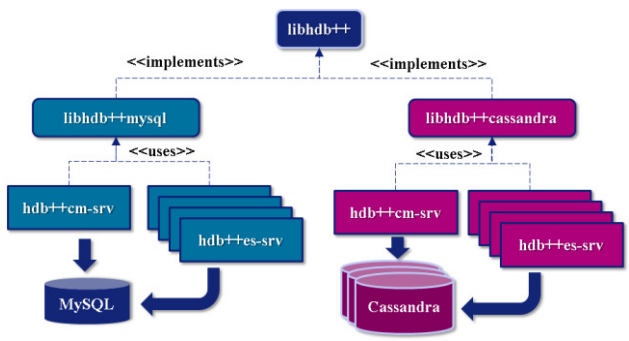

Figure 1: HDB++ Device Servers design.

The Cassandra HDB++ libraries allow us to reuse the Event Subscriber and Configuration Manager Device Servers as well as the Graphical User Interfaces (GUI) interacting with these Device Servers (HDB++ Configurator GUI shown on Fig. 2 and HDB++ Diagnostics GUI), without changing their source code. Linking the two Device Servers with *libhdb++cassandra* was sufficient to be able to store data into Cassandra.

For the data extraction part, new Java classes were developed for HDB++ Cassandra, extending the features provided by an abstract layer hiding the database back-end to the Java applications.

ISBN 978-3-95450-148-9

A Java GUI named HDB++ Viewer (Fig. 3) was developed, using these Java classes, to visualize the data stored in the historical database.

The GUI can be configured to read data from HDB++ Cassandra, from HDB++ MySQL or from the ESRF Oracle old Historical Database.

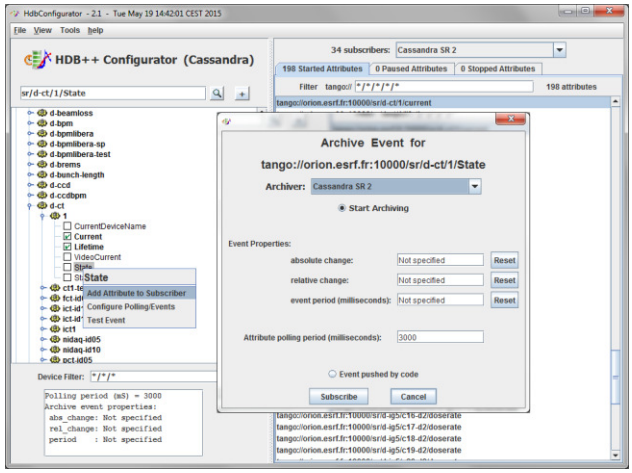

Figure 2: HDB++ Configurator GUI.

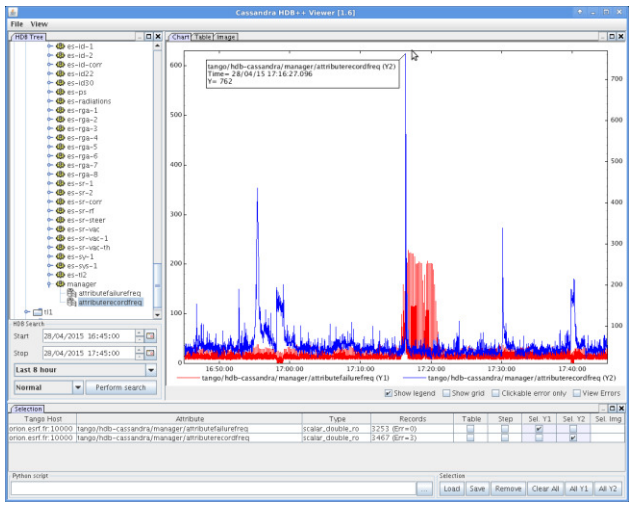

Figure 3: HDB++ Viewer GUI showing the biggest events peak (762 events/s) observed in 2015.

The HDB++ Cassandra Java classes are retrieving the data directly from the Cassandra cluster and are written in a way to take advantage of the replication. A data request is split into several smaller requests to ensure each query targets one and only one Cassandra partition to maximize performances. The requests are sent asynchronously and in parallel, taking advantage of the fact that the data is distributed over several Cassandra nodes. The data model has been designed in such way that there is one partition per attribute per period of time. This period of time is adjustable in the Event Subscriber Device Server code and is currently set to one day. In the future, this period will be set to one hour, meaning one partition will contain one hour of data of a specific attribute.

A C++ extraction library for HDB++ Cassandra will be implemented soon, inheriting from the C++ HDB++ abstract extraction library already developed by Elettra.

THE CASSANDRA EXPERIENCE

The Cassandra basics were learnt very quickly thanks to the numerous tutorial online videos and to the Datastax academy web site [7].

Since the Event Subscriber and Configuration Manager Device Servers were written in C++, the *libhdb++cassandra* library described above had to be implemented in C++. To do this, it was necessary to use the Cassandra C++ driver, which was still in Beta version at that time. The Cassandra C++ driver appeared to be stable for our usage. The version 2.0.1 of the driver, which provides useful features like latency-aware routing, is currently in use. Newer versions, which will be used soon, are providing retry policies, making the code simpler and more robust to failures.

After a few tests archiving a few attributes, it was decided early in the project to store real data coming from the accelerator control system. So the system was configured to archive all the accelerator control system TANGO attributes which were already being archived in the old HDB ESRF database. The HDB++ Configuration GUI (Fig. 2), really helped to simplify this task.

Following the Apache Cassandra website advices, the project was started with Cassandra 2.1.0, with 3 recycled servers (See Table 1). The problem is that useful data was stored almost from the start of the project and Cassandra 2.1.x branch appeared to be a development branch with some remaining bugs so an upgrade was necessary each time a new version was released until version 2.1.7. The bugs didn't cause any loss of data. The most annoying bug was CASSANDRA-8321 [8] which was filling up the disks with temporary files not removed after use. A cron job had to be created in order to remove the oldest of these temporary files until a new version fixing the bug was released. The upgrade was always smooth with no downtime of the HDB++ system. To upgrade a Cassandra cluster, one needs to do a rolling upgrade, meaning each node is upgraded one after the other so there is always only one node down at a given moment. Missing data is then streamed by the other nodes once the node is back online thanks to Cassandra consistency repair features [9].

Table 1: Recycled Servers' Specifications

Item	Description
CPU	Intel Xeon E5440, 2.83GHz, 4 cores, 2 CPUs
Memory	48 GB
OS	Debian 7 (Wheezy) - 64 bits
Storage 1 (OS)	SAS 10K RPM 2x72 GB – RAID 1
Storage 2 (Data)	SAS 10K RPM 2x146 GB - RAID 0 SAS 10K RPM 1x146 GB
Network	1 Gb Ethernet x 1 port

ISBN 978-3-95450-148-9

Data Model

The schema and tables names were inspired by the HDB++ MySQL tables.

Cassandra provides very high performances but in order to maintain these performances, some operations which are allowed in traditional relational databases are forbidden in Cassandra, like the JOIN queries for instance or SELECT queries with a WHERE clause not containing the partition key. For this reason, it was not possible to reuse directly the same database schema as the one created for the HDB++ MySQL version and some additional tables had to be created to overcome this difficulty. Cassandra Query Language collection types were also used to store the Tango array types.

Partition keys

The partition key is a part of the primary key which will be used by Cassandra to decide on what nodes data having a primary key containing this partition key will be stored. It was decided to use the HDB++ attribute configuration ID and a text field named *period* as partition key for all the tables storing historical values. At the beginning, the name of the current month was used for this *period* field (e.g. "2015-10"). The partition sizes were too big and this was degrading the performances, causing long garbage collector pauses when users were requesting a full month of data for an attribute, making a Cassandra node unresponsive during the garbage collection process period, which could last for more than one minute sometimes. It was decided to insert the current day in the *period* field to reduce the partitions sizes. This improved the performances but is still not perfect. This is why one partition per attribute and per hour will be created in the future (and the already stored data will be migrated) to keep the partition sizes below 100MB, following the Cassandra experts' recommendations.

Cassandra Administration

At the ESRF, Datastax OpsCenter Community Edition, shown in Fig. 4, is used to monitor the Cassandra cluster. OpsCenter is a web-based visual monitoring tool for Apache Cassandra.

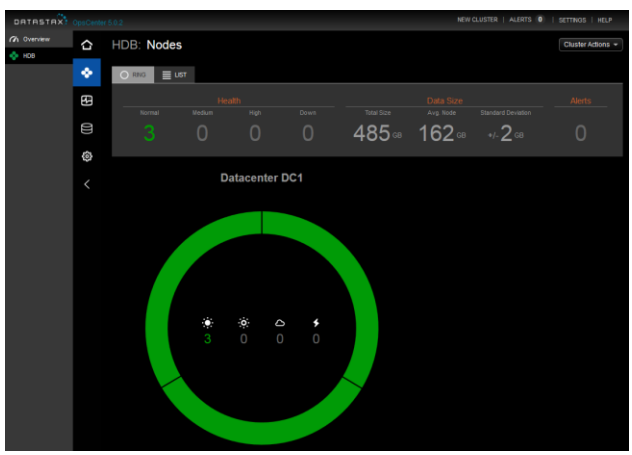

Figure 4: Datastax OpsCenter monitoring tool showing HDB++ Cassandra cluster status at the ESRF.

CURRENT STATUS

The HDB++ Cassandra version at the ESRF is currently storing data coming from about 7440 TANGO attributes managed by 34 Event Subscriber Device Server instances. The maximum number of attributes managed by a single Event Subscriber at the ESRF is currently 940.

The Figure 5 is showing the attributes distribution among the Event Subscriber instances as well as the number of received events per event subscriber during a period of about 1 day 1 hour and 20 minutes, on 12 -13 October 2015. The maximum number of events received for a single event subscriber during that period was a bit more than 1.5 million on that typical day.

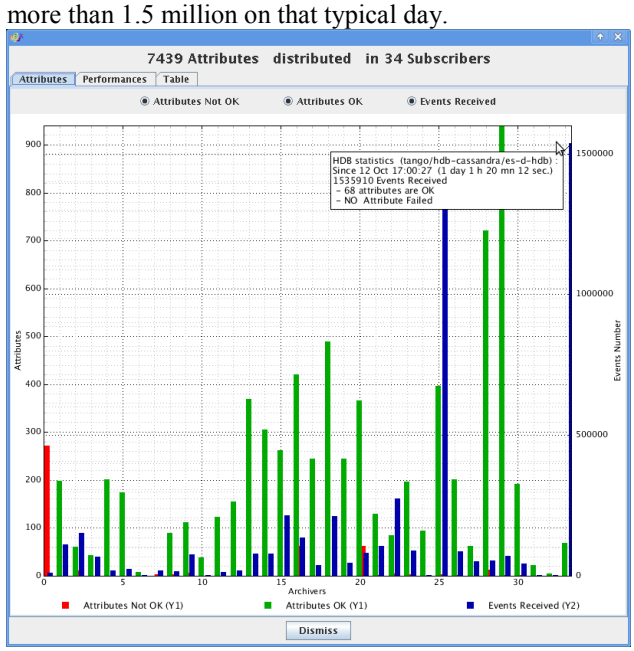

Figure 5: Distribution window from the HDB++ diagnostics GUI.

The Cassandra cluster is currently composed of 3 nodes running Cassandra 2.1.7. The average data size per node is about 162GB. The recycled servers described in Table 1 are still used for the moment.

The architecture of the HDB++ Cassandra system at the ESRF is described in Fig. 6 as well as the future evolution of the cluster.

FUTURE PLAN

New Nodes

3 new servers have been bought and will be replacing soon the 3 current nodes after having upgraded Cassandra to the latest stable version (probably 2.1.10).

Analytics Datacentre

3 additional servers with SSD disks have been bought to constitute a new datacentre dedicated to analytics. They will be located in a different room than the production datacentre for a better protection of the data, in case of fire for instance. The specifications of all these new servers are described in Table 2.

ISBN 978-3-95450-148-9

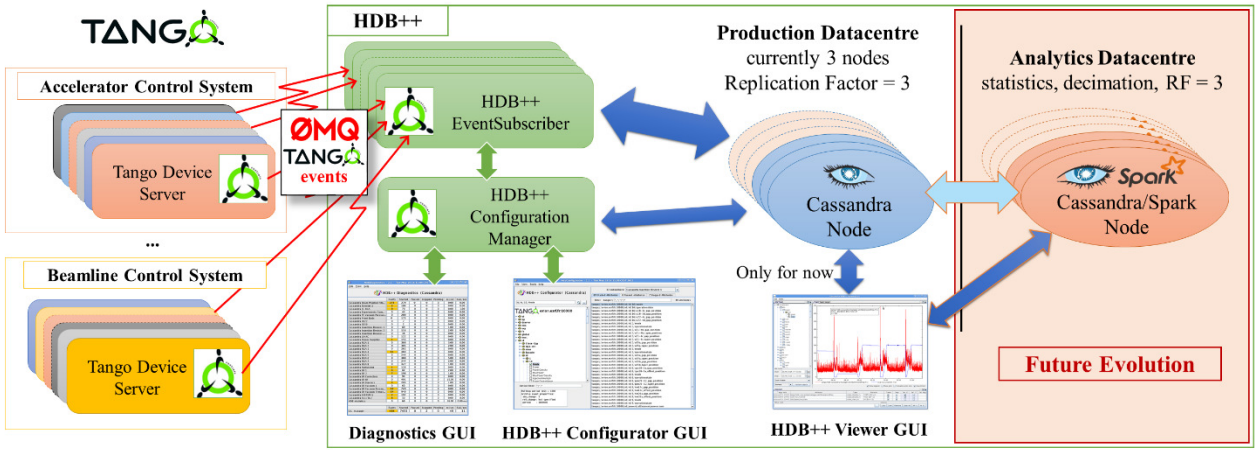

Figure 6: HDB++ Cassandra system architecture at the ESRF and future evolution.

Table 2: New Servers' Specifications

Item	Production DC	Analytics DC
CPU	Intel Xeon E5-2630, 2.4GHz, 8 cores	Intel Xeon E5-2630, 2.4GHz, 8 cores, 2 CPUs
Memory	64 GB	128 GB
OS	Debian 7 (Wheezy) - 64 bits	
Storage 1 (OS)	SAS 15K RPM 300GB x2 – RAID 1	
Storage 2 (data directories)	1x 1TB, SATA, 7200 RPM 1x 2TB NL-SAS 7200 RPM	SSD SATA MLC 800GBx4
Network	1 Gb Ethernet x 4 ports	

Apache Spark [10] will be installed on the *Analytics* datacentre Cassandra nodes. It will be used to compute statistics (minimum, maximum, average, received events count, errors count) per attribute and will compute decimation data which will be inserted into new Cassandra tables from a new keyspace dedicated to analytics data.

The HDB++ Viewer tool will be improved in order to return decimated data and statistics first when a user will want to retrieve a big amount of data. That way, the user will get quickly an overview of the data and might be able to narrow down the region of interest by zooming in and reducing the time interval. At some point, once the amount of data to retrieve will be below a reasonable number (to be defined), the user will see the real data.

CONCLUSION

HDB++ Cassandra proved it can replace the old ESRF Historical Database. It demonstrated it is easy to develop additional database back-end support for HDB++.

It might also become a good alternative to the current TANGO Temporary Database (TDB).

Apache Cassandra proved it is able to deliver continuous availability as long as the partitions sizes stay small to avoid long garbage collector cycles.

Using Apache Cassandra as HDB++ database back-end is a good fit for projects requiring big storage capacities and continuous availability, as well as for projects where more storage capacities and performances might be needed in the future, like at the ESRF for instance with the coming upgrade of the accelerator. For smaller projects, the HDB++ MySQL version is enough and cheaper because less servers are needed.

The analytics cluster using Apache Spark should allow to bring better user experience and read performances thanks to the new decimation tables and SSD disks.

ACKNOWLEDGMENT

The Elettra HDB++ team for their work on the HDB++ project and for their open mind.

DuyHai Doan, technical advocate at Datastax, for his advices and Cassandra expertise.

REFERENCES

[1] A. Götz et al., "TANGO V8 – Another turbo charged major release", ICALEPCS'13, San Francisco, USA (2013), http://jacow.org
[2] ZeroMQ: http://zeromq.org
[3] TANGO Controls: http://www.tango-controls.org
[4] L. Pivetta et al., "HDB++: A New Archiving System for TANGO", *These Proceedings,* WED3O04, ICALEPCS'15, Melbourne, Australia (2015).
[5] MySQL: http://dev.mysql.com
[6] Apache Cassandra: http://cassandra.apache.org
[7] Datastax Academy: https://academy.datastax.com
[8] CASSANDRA-8321 bug: https://issues.apache.org/jira/browse/ CASSANDRA-8321
[9] Cassandra consistency repair features: http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dmlConsistencyRepair.html
[10] Apache Spark: https://spark.apache.org

ISBN 978-3-95450-148-9