HDB++: A NEW ARCHIVING SYSTEM FOR TANGO

L. Pivetta, C. Scafuri, G. Scalamera, G. Strangolino, L. Zambon,
Elettra Sincrotrone Trieste, Trieste, Italy
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 lightweigh event-driven communication. Exploiting these improved capabilities, a high performance, event-driven archiving system written in C++ has been developed. It inherits the database structure from the existing TANGO Historical Data Base (HDB) and introduces new storage architecture possibilities, better internal diagnostic capabilities and an optimized API. Its design allows storing data into traditional database management systems such as MySQL or into NoSQL database such as Apache Cassandra. The paper describes the software design of the new HDB++ archiving system, the current state of the implementation and gives some performance figures and use cases.

INTRODUCTION

The TANGO archiving system is a tool that allows 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 large facilities, such as long term monitoring of subsystems, statistics, correlation of parameters 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] has been designed and developed in collaboration between Elettra and ESRF.

DESIGN GUIDELINES

A number of requirements have been taken into account during the design phase. The HDB++ archiving system must fully comply to the TANGO device server model, with two immediate benefits. First, all the required configuration parameters are stored to and retrieved from the TANGO database; some of these parameters are, for user convenience, duplicated into a dedicated table of the HDB++ schema by a mechanism that guarantees the consistency of the copy. Second, the HDB++ archiving system inherits the TANGO scaling capability: any number of EventSubscriber instances can be deployed according to the desired architecture and overall performance.

The publish/subscribe paradigm is available in TANGO via the event subsystem. More in detail, the *archive* event is provided for archiving purposes and can be triggered on threshold comparison and/or periodic basis. The HDB++ architecture is fully event based; therefore, a part of HDB++ setup consists of conveniently configure TANGO device servers to send events as required.

Two TANGO device servers have been developed. The EventSubscriber, also referenced as archiver, is in charge of gathering the values from the TANGO devices and storing them into the historical database. To address the requirements coming from large systems the need to distribute the workload over a number of archivers shows up. A ConfigurationManager TANGO device server will assist in the operations of adding, editing, moving and deleting an Attribute to/from the HDB++ archiving system. A specific library, exposing a suitable API, addresses the historical data extraction from the archive.

The task of each HDB++ archiving system component is sharply defined; low layer devices, e.g. archivers, have no dependency against the ConfigurationManager, the extraction library or the graphical user interfaces and, possibly, can be deployed standalone. Also, the HDB++ architecture has been designed to easily support different SQL and NoSQL database engines: an abstraction library decouples the interface to the database back-end from the implementation. Adding a new back-end is just matter of writing the code for the specific implementation; this has been done, as an example, during last year to introduce the support for Cassandra [4].

EVENT SUBSCRIBER

The EventSubscriber TANGO device server is the core of the HDB++ archiving system. It subscribes to archive events for the specified Attributes list, stored into a Property in the TANGO database, as well as a number of additional parameters, such as the hostname and port number where the back-end is running, the name of the database and the username and password to be used.

A dedicated thread is in charge of event subscription and callbacks execution; the callbacks, acting as producers, put the complete data structure of the received events in a FIFO queue, protected by a suitable locking mechanism. The thread and the callbacks must be able to handle an arbitrary number of events, possibly limited just by the available memory and the required performances. One additional thread, acting as consumer of the FIFO, is in charge of writing the data into the database. Moreover, a high-mark threshold is setup on the FIFO queue to alert for an overloaded EventSubscriber.

The EventSubscriber device server allows to perform the following operations:

- add/remove an Attribute to/from archiving
- start/stop the archiving for all Attributes
- start/stop the archiving for one Attribute
- read the status of an Attribute
- read the list of Attributes currently archived (started)

ISBN 978-3-95450-148-9

Pre-Press Release 23-Oct-2015 11:00
Copyright © 2015 CC-BY-3.0 and by the respective authors

- read the list of Attributes currently not archived (stopped)
- read the number/list of Attributes in charge
- read the configuration parameters of each Attribute
- read the number/list of working Attributes
- read the number/list of faulty Attributes
- read the number/list of Attributes pending in the FIFO

Working at the EventSubscriber level implies that the database entry and the archive event parameters have to be already configured. Besides, no action is performed on the archived data when removing an Attribute, which means that the data remain available in the historical database.

Special care has been reserved to the error management. One NULL value with time stamp is inserted whenever the archiving of an Attribute is stopped due to error. Moreover, if an error occurred, the corresponding Attribute is marked as faulty in the archiving engine and the error description stored. In case the archiving was suspended due to an error, it is automatically resumed when valid data is available again. The quality factor of the Attribute is also stored into the historical database. Exploiting these features, a client could be fully aware of the archiving status of an Attribute; in addition, dedicated alarms can be configured in the TANGO Alarm System to asynchronously inform about the status of the archiver device.

The EventSubscriber TANGO device server also exposes some additional figures of merit, such as:

- for each instance, total number of records per time
- for each instance, total number of failures per time
- for each Attribute, number of records per time
- for each Attribute, number of failures per time
- for each Attribute, time stamp of last record

These numbers can sum up in a counter, which can be reset every hours/days/weeks, to rank each Attribute in term of data rate, error rate etc. This allows preventive maintenance and fine tuning, detecting, for instance, when an Attribute configuration is wrong because the variation threshold is lower than the noise level. These statistics are a key element for qualifying the health of the system. All these Attributes are archived themselves to enable a follow-up versus time. For each Attribute, the EventSubscriber TANGO device server also computes the minimum and maximum processing and storing times, which helps discovering possible bottlenecks.

CONFIGURATION MANAGER

Adding an Attribute to the archiving system may require creating the new entry into the database tables, setting up the Attribute archive event configuration and assigning the Attribute to one of the archivers. Moreover, large systems may need to distribute the workload over several EventSubscriber device servers. A special TANGO device server, the ConfigurationManager, has been developed to simplify the above steps and to help monitoring the whole HDB++ archiving system. Adding an EventSubscriber device to the ConfigurationManager pool enables the management. This leaves open the possibility of deploying un-managed archivers, if needed.

The ConfigurationManager device server is able to perform the following operations on the managed EventSubscriber pool:

- handle the request of archiving a new Attribute
  - create an entry in the database if not existing
  - setup the Attribute archive event configuration
  - assign the new Attribute to one of the archivers
- move an Attribute from one archiver to another
- show the Attribute/archiver coupling
- start/stop the archiving of an Attribute
- remove an attribute from archiving

The ConfigurationManager also exposes some Attributes to keep trace of the global statistics:

- total number of EventSubscribers
- total number of working/faulty attributes
- total number of events per second
- overall minimum and maximum processing and storing time

These attributes could be themselves archived to enable a follow up versus time. The statistics window GUI for MySQL back-end at the ESRF is shown in Fig. 1.

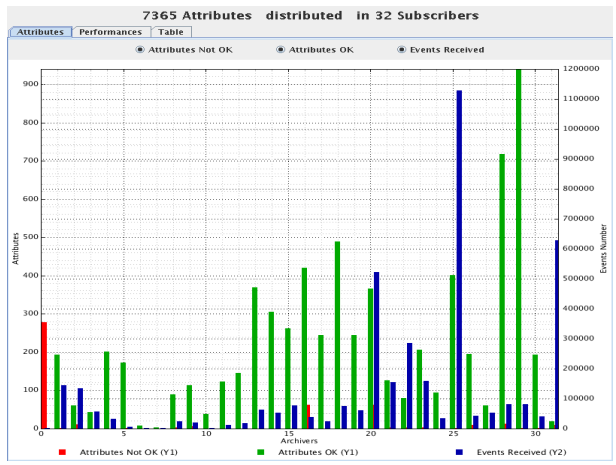

Figure 1: MySQL back-end statistics. X: archivers, Y1: Attributes [red faulty/green ok], Y2: events number [blue].

To guarantee the consistency of the archiving setup, the ConfigurationManager implements a strict sequence when adding an Attribute to the archiving system. More in detail, at first the historical database entry for the Attribute is created, then the archive event parameters are configured, or checked if already existing, and finally the Attribute is assigned to the desired archiver.

HISTORICAL DATABASE

Currently the HDB++ archiving system supports MySQL [5], a relational database management system, and Cassandra as back-ends. Cassandra is a distributed NoSQL database where the data is spread over a number of nodes. Its architecture provides replication, high availability with no single point of failure and linear scalability, meaning that

Pre-Press Release 23-Oct-2015 11:00

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

higher performances can be achieved simply adding more nodes. A detailed description of the Cassandra support in HDB++ is available in [6].

Some shared libraries provide the methods for writing to the database back-end. These libraries, written in C++, are addressed to the EventSubscriber TANGO device server and their main purpose is to provide an abstraction layer. Actually, some shared objects are available implementing the abstraction layer and the specific interface:

- *libhdb++*: database abstraction layer
- *libhdbmysql*: legacy HDB schema support for MySQL back-end
- *libhdb++mysql*: HDB++ schema support for MySQL back-end
- *libhdb++cassandra*: HDB++ schema support for Cassandra back-end

These libraries allow reusing the EventSubscriber, the ConfigurationManager and the GUIs without changes. The HDB++ archiving system can be easily extended to support additional database engines, such as Oracle, PostgreSQL or other NoSQL databases, just writing the specific support library.

The database schema characteristics are common to both MySQL and Cassandra back-end. The *att_conf* table associates the attribute name with a unique ID and selects the data type; it's worth noting that the *att_name* row always contains the complete FQDN, e.g. with the hostname and the domain name. The *att_history* table stores the timestamps of the operations made on each Attribute, such as adding, removing, starting or stopping the archiving. Each TANGO data type has a dedicated table containing the Attribute ID, the Attribute data timestamp, the event timestamp, the database insert timestamp and the data value. Comparing the timestamps, any possible performance bottleneck can be easily detected. With respect to the legacy HDB schema, the new HDB++ schema introduces some relevant changes:

- μs timestamp resolution
- no per-attribute additional tables; the number of tables used is fixed and does not depend on the number of archived attributes
- specific TANGO data type support

CONFIGURATION TOOLS

A graphical user interface for the ConfigurationManager has been developed. Written in Java, the HdbConfigurator GUI presents a Jive-like interface, showing on the left side the device tree and on the right side the selected archiver with the lists of the relevant started and stopped Attributes. On bottom left, the archive event parameters of the selected Attribute appear. A screenshot of the HdbConfigurator GUI is shown in Fig. 2.

To use the HdbConfigurator, the `HdbManager` environment variable, containing the ConfigurationManager TANGO device server to be used, has to be exported. Then, once the desired archiver has been selected, the device tree can be browsed for the requested Attribute. A right-click on

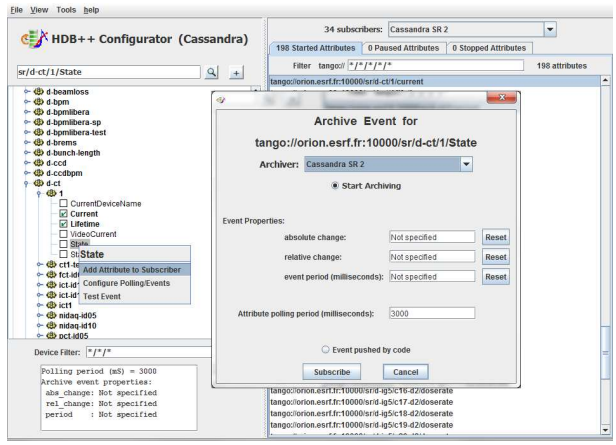

Figure 2: HdbConfigurator GUI.

the Attribute name opens a pull-down menu; in the popup window the user can specify the archive event parameters, if needed, or mark the event pushed by code flag. A list of Attributes, stored in a file, can be added to the archiving system using the File/Open menu of the HdbConfigurator GUI. Moreover, whenever a large number of Attributes has to be added to the archiving system, using a programmatic approach could be convenient. A client application for the ConfigurationManager TANGO device server can consist of a few lines of Python or Java and address in a very efficient and effective way the above requirement, especially when also the archive event parameters for each Attribute have to be specified.

DATA EXTRACTION

A specific API has been defined to address the historical data extraction. The data extraction library shall be able to deal with event based archived data. The possible lack of data inside the requested time window shall be properly managed:

- returning some *no-data-available* error: in this case the reply contains no data and an error is triggered;
- enlarging the time window itself to comprehend some archived data: the requested time interval is enlarged to include some archived data. No fake samples are introduced to fill the values in correspondence of the requested timestamps;
- returning the value of the last archived data anyhow: the requested time interval is kept and the last available data sample is returned. The validity and the consistency of the data is guaranteed when the archive event change threshold is configured; care must be taken with the dataset in case of periodic archiving event;

Two libraries have been developed implementing this API: the first, written in C++ is dedicated Qt/Qtango based GUIs or to C++ TANGO device servers; the second, written in Java, has been used for the HdbViewer GUI and is a native choice for Java device servers.

ISBN 978-3-95450-148-9

Pre-Press Release 23-Oct-2015 11:00
Copyright © 2015 CC-BY-3.0 and by the respective authors

The HdbExtractor++ multithread library allows fetching the data from the legacy HDB and the new HDB++ MySQL schema in a simple Object Oriented way. An additional module provides a Qt interface to the HdbExtractor++ and a dedicated GUI, exploiting the MathGL framework, aimed at displaying mono and bidimensional data over time. Also, possible errors conveniently stored in the database can be found an displayed. Figure 3 and 4 show a multiline plot and a surface plot.

Figure 3: HdbExtractor++ multiline plot.

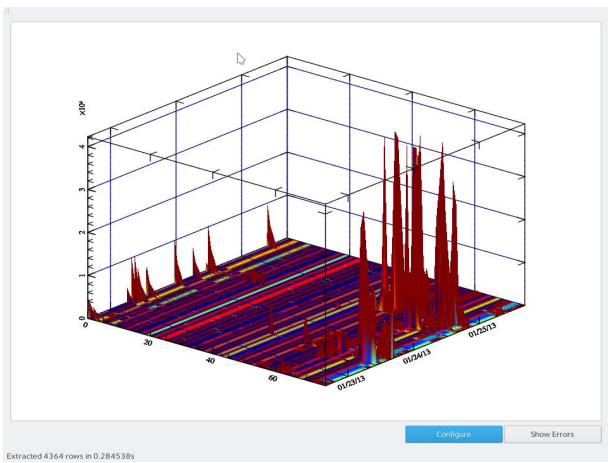

Figure 4: HdbExtractor++ surface plot.

The HdbViewer Java framework, in addition to the legacy ESRF historical database support, allows retrieving the data from the new Cassandra back-end as well as managing the Cassandra partitioning period. The classic table display of the HdbViewer GUI is shown in Fig. 5. Multiline bidimensional plots are also supported.

PERFORMANCE

Currently more than 6800 Attributes are archived with the HDB++ at Elettra, on both the legacy HDB schema and the new HDB++ schema using MySQL as back-end; respectively 30 and 20 instances of the EventSubscriber have been

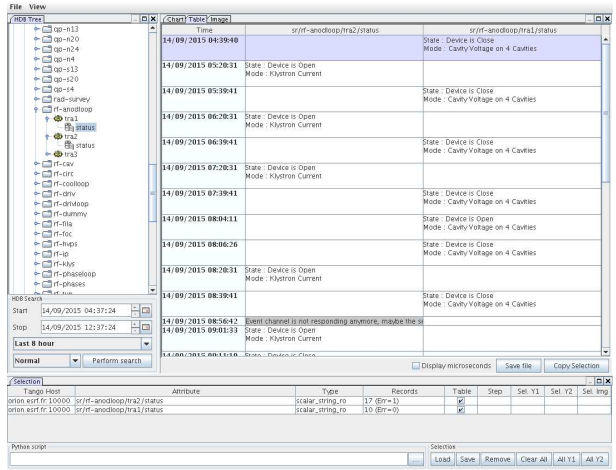

Figure 5: HdbViewer, classic table display.

deployed, each one managing a very different number of Attributes, spanning from just one to more than one thousand per archiver. A single EventSubscriber device server, running on an unloaded machine, is capable to handle thousands of events per second, sustained rate, with easy; peaks of a few tens of thousands can also be handled exploiting the FIFO for caching. However, in this scenario, or at even higher rates, care has to be taken with respect to the database back-end that can become the bottleneck.

Similarly, more than 7300 Attributes are archived with the HDB++ at the ESRF using the new HDB++ schema on both MySQL back-end, exploiting 32 archivers, and Cassandra back-end, using 34 archivers.

CONCLUSION

The HDB++ archiving system, though still under development, is in production at both Elettra and ESRF sites since almost two years. The ConfigurationManager device server in conjunction with the HdbConfigurator GUI greatly simplify the utilization. The administration of an HDB++ archiving system is quite easy, especially for anyone with some TANGO experience, although the installation is still somehow tricky. Some Debian packages are foreseen to simplify the installation procedure for selected platforms and will be made available in the next future.

REFERENCES

[1] A. Götz et al., "TANGO V8 - Another turbo charged major release", ICALEPCS'13, San Francisco, USA (2013).

[2] ZeroMQ: `http://zeromq.org`

[3] TANGO Controls: `http://www.tango-controls.org`

[4] Apache Cassandra: `http://cassandra.apache.org`

[5] MySQL: `http://dev.mysql.com`

[6] R. Bourtembourg et al. "How Cassandra improves performances and availability of HDB++ TANGO archiving system" WEM310, *These Proceedings*, ICALEPCS'15, Melbourne, Australia (2015).

Pre-Press Release 23-Oct-2015 11:00

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9