ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THMBCM001

NEW DEVELOPMENTS ON HDB++, THE HIGH-PERFORMANCE DATA ARCHIVING FOR TANGO CONTROLS

D. Lacoste, L. Banihachemi, R. Bourtembourg, ESRF, Grenoble, France
S. Rubio-Manrique, ALBA-CELLS, Cerdanyola del Vallès, Spain
J. D. Mol ,ASTRON, Dwingeloo, Netherlands
L. Pivetta, G. Scalamera, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
J. Forsberg, MAX IV Laboratory, Lund, Sweden
T. Juerges, SKA Observatory, Jodrell Bank, United Kingdom

Abstract

The Tango HDB++ project is a high performance event-driven archiving system which stores data with micro-second resolution timestamps. HDB++ supports many different back-ends, including MySQL/MariaDB, TimescaleDB, a time-series PostgreSQL extension, and soon SQLite. Building on its flexible design, with the latest developments supporting new back-ends is even easier. HDB++ keeps improving with new features, such as batch insertion, and by becoming easier to install or setup in a test environment, using ready to use docker images and striving to simplify all the steps of deployment. The HDB++ project is not only a data storage installation, but a full ecosystem to manage data, query it, and get the information needed. In this effort a lot of tools were developed to put a powerful back-end to its proper use and be able to get the best out of the stored data. Moreover, the latest developments in data extraction, from low level libraries to web viewer integration, such as Grafana, will be presented, pointing out strategies in use in terms of data decimation, compression and others to help deliver data as fast as possible.

INTRODUCTION

Since about ten years, the TANGO HDB++ archiving system is developed as a collaborative project between different institutes using Tango Controls (Alba, Astron, Elettra, ESRF, INAF, MaxIV, SKAO,...).

HDB++ provides tools and components to store Tango attribute values into the database back-end of your choice, with micro-second resolution timestamps. Tools and libraries are also provided for the extraction and visualization of the stored data, for the configuration of the attributes to be stored and for the monitoring of the archiving system health. The following back-ends are currently supported: MySQL/MariaDB, PostgreSQL, TimescaleDB and SQLite. New Database back-ends can be easily added, thanks to HDB++ modular design.

Design

The EventSubscriber Tango device subscribes to a list of Tango attribute events and stores the attribute values received with these events into a database. The ConfigurationManager Tango device helps managing the list of attributes to be stored in the historical database.

These two Tango devices, written in C++, use the libhdb++ abstraction library to decouple the interface to the database back-end from the implementation. HDB++ provides the libraries implementing the libhdb++ interface for the supported back-ends. Thanks to this design, the same tools can be used to manage and monitor the archiving system whatever back-end is used. Adding a new database back-end is quite easy, because it is just matter of creating the library implementing the libhdb++ interface for the new back-end.

DATABASE BACK-ENDS

One of the main strengths of HDB++ is the use of proven database engines for storage back-ends. Building an abstraction layer to interact with different back-ends is a key strategy that allows to select the preferred back-end, based on performance, system footprint, preferred technology or simply in-house expertise. The HDB++ community is currently supporting five different back-ends, in production in the various institutes, facing additional requests to support new ones. A small comparison of the different back-ends is presented in Fig. 1. A skeleton project is available to developers to help supporting new back-ends, standardizing the building steps and making the integration simpler.

MariaDB/MySQL

MySQL has been the database engine used for the Tango Database server since the very first releases of Tango due to its versatility and ease of installation. The fully compatible open source MariaDB variant is available in all major operating systems and Linux distributions, becoming the default option for small installations. MySQL/MariaDB also demonstrated performance and scalability on large installations, like ALBA or Elettra synchrotrons, where it is used to support the whole archiving system for accelerators and beamlines. MySQL/MariaDB installations support different schema, either using the legacy schema or the new schema introduced in HDB++. Deployment can use either one single database in a single host, database clusters or multiple databases. Using different API's like ProxySQL or PyTangoArchiving helps configuring and extracting data from the databases in a transparent way.

PostgreSQL/TimescaleDB

TimescaleDB is a PostgreSQL extension for time series. It manages partitioning and offers a lot of features dedicated to time series. There are actually two implementation libraries to support insertion in TimescaleDB and PostgreSQL, with the difference in the underlying schema used. Both libraries could be used with one back-end or the other, provided that the schema is correct. TimescaleDB back-end is actively developed. The regular maintenance happens, to follow updates in the dependencies, namely *cpptango* and *libpqxx*, for interfacing with the database. Moreover, additional work is extending HDB++ support of the feature set provided by TimescaleDB, such as the powerful compression engine. More detail will be provided in the long-term archiving strategies section. Another interesting feature is the introduction of *jobs* in TimescaleDB to run some processing on the data. TimescaleDB *jobs* will replace some scripts that currently run outside of the database engine to perform some periodic operations. This opens the door to interesting ideas, such as running algorithms for data decimation directly in the back-end, for maximum performance.

SQLite

The latest addition to the list of HDB++ supported back-ends, SQLite has been requested to help deploy faster test setups for HDB++. SQLite has been an interesting exercise to test in real condition the effectiveness of the skeleton provided to develop new back-ends. Adding the support for SQLite faced two main challenges: the definition of the schema and the implementation. Different HDB++ back-ends share a similar schema, as far as possible, but not all the features are supported, such as arrays, nor data types. Therefore, a schema working with SQLite has been designed. Exploiting a well defined and compact interface, and a project ready with only some gaps to fill, reduced the burden and allowed to focus on learning the intrinsics of SQLite insertion library. The SQLite back-end, outcome of a fruitful internship at the ESRF, is still work in progress, with the need to refine the project and add some test, but can be already used to quickly setup a test environment.

Elasticsearch

Support for Elasticsearch has been added as a proof of concept, but it is no more maintained. Elasticsearch HDB++ back-end is currently deprecated.

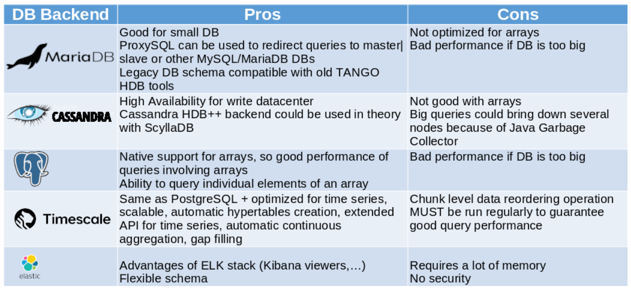
Cassandra

Cassandra support in HDB++ is deprecated.

DEPLOYMENT

Repositories

HDB++ related projects can be found on Gitlab [1]. Most, if not all, the projects document how to build and install the software. For convenience, the data insertion libraries are usually done with two repositories, one containing the code for the insertion library itself and a single repository to install a full system directly. Taking MySQL as example, the repository with the code for the library is available [2],



DB Backend	Pros	Cons
MariaDB	Good for small DB ProxySQL can be used to redirect queries to master|slave or other MySQL/MariaDB DBs Legacy DB schema compatible with old TANGO HDB tools	Not optimized for arrays Bad performance if DB is too big
Apache CASSANDRA	High Availability for write datacenter Cassandra HDB++ backend could be used in theory with ScyllaDB	Not good with arrays Big queries could bring down several nodes because of Java Garbage Collector
	Native support for arrays, so good performance of queries involving arrays Ability to query individual elements of an array	Bad performance if DB is too big
Timescale	Same as PostgreSQL + optimized for time series, scalable, automatic hypertables creation, extended API for time series, automatic continuous aggregation, gap filling	Chunk level data reordering operation MUST be run regularly to guarantee good query performance
elastic	Advantages of ELK stack (Kibana viewers,...) Flexible schema	Requires a lot of memory No security

Figure 1: HDB++ back-ends pros and cons.

together with another repository designed to build the entire system [3]. The *mono-repository* builds the insertion library specific to the selected back-end and the *hdbpp-es* and *hdbpp-cm* device servers. It contains as well the schema for the database and the instructions to setup the database. A recipe to build a ready to use docker image with the back-end is available. The *mono-repository* is the preferred place to start experimenting with HDB++.

Conda Packages

Several HDB++ Conda packages can be installed from conda-forge [4], only for Linux x86_64 architecture at the moment. Available packages are listed below:

- *libhdbpp*: the interface library for HDB++,
- *libhdbpp-timescale*: HDB++ library for TimescaleDB database backend,
- *hdbpp-es*: HDB++ EventSubscriber Tango device,
- *hdbpp-cm*: HDB++ ConfigurationManager Tango device.

hdbpp-es and *libhdbpp-timescale* debug symbols have been moved to separate packages, named respectively *hdbpp-es-dbg* and *libhdbpp-timescale-dbg*, keeping the main packages small but allowing users to debug by just installing these packages.

CONFIGURATION

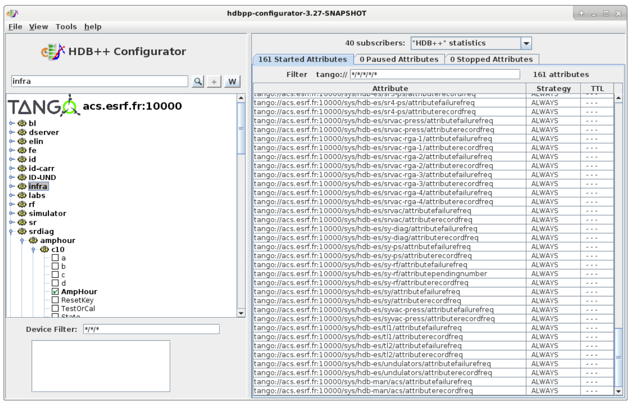
HDB++ is a complete ecosystem that allows to set up an archiving system for any Tango based control system painlessly providing a set of tools for configuration and management.

HdbConfigurator

HdbConfigurator, shown in Fig. 2, is the legacy configuration manager for HDB++. It is a java application, that helps configuring attributes within the control system to enable archiving. HdbConfigurator, although not strictly necessary to set up archiving, can swiftly orchestrate the configuration of Tango Attributes for the Tango devices to be archived and, at the same time, the proper EventSubscriber device.

Archwizard

For monitoring and troubleshooting a running HDB++ installation, MAX IV developed a simple web application



infra

acs.esrf.fr:10000
- bl
- dserver
- elin
- fe
- id
- id-carr
- ID-UND
- infra
- labs
- rf
- simulator
- sr
- srdiag
  - amphour
    - c10
      - ☐ a
      - ☐ b
      - ☐ c
      - ☐ d
      - ✓ AmpHour
      - ☐ ResetKey
      - ☐ TestOrCal

Device Filter: */*/*

40 subscribers: "HDB++" statistics

Filter tango:// */*/*/*/* 161 attributes

Attribute	Strategy	TTL
tango://acs.esrf.fr:10000/sys/hdb-es/sr4-ps/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sr4-ps/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-press/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-press/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-1/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-1/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-2/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-2/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-3/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-3/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-4/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac-rga-4/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/srvac/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-diag/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-diag/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-ps/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-ps/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-rf/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-rf/attributependingnumber	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy-rf/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/sy/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/syvac-press/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/syvac-press/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/tl1/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/tl1/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/tl2/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/tl2/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/undulators/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-es/undulators/attributerecordfreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-man/acs/attributefailurefreq	ALWAYS	- - -
tango://acs.esrf.fr:10000/sys/hdb-man/acs/attributerecordfreq	ALWAYS	- - -

Figure 2: HdbConfigurator GUI.

dubbed *Archwizard* [5], which lists managers, archivers and archived attributes, and allows inspecting settings and errors, with search and filter options.

yaml2archiving

Since HDB++ archiving configuration can be quite large, and is stored in the Tango database, MAX IV introduced an additional option for handling the configuration. A file in YAML format can be used to describe the archiving configuration, e.g. which Tango device Attributes have to be archived and their settings; a script [6] is available to process the YAML file and update the proper archiver configuration.

VIEWERS / EXTRACTORS

Grafana

With powerful features but keeping simplicity at its core, Grafana quickly became a de-facto standard for data visualization and dashboards interface using web technologies. Once again, the adoption of widely used database engines as back-end proved useful, as Grafana natively supports all the HDB++ back-ends. Grafana integration has been straightforward; exploiting the web interface it is easy to setup a Grafana instance to connect to HDB++ back-end and provide basic data visualisation. The community is now working on HDB++ dedicated synoptic, based on Grafana, or more generic data viewers; Grafana support is still in the development stage and not ready for production.

eGiga2m

eGiga2m is a web graphic data viewer. Data are supposed to be organized as a set of unevenly spaced time series. Time series are taken from a web service, which typically extracts data from a structured database, or can be loaded from a CSV file; drag-and-drop on the plot area is supported. HDB++ is one of the possible data sources, in particular MySQL/MariaDB schema and TimescaleDB schema used in HDB++ are supported. Each time series is identified by a unique name and by an ID. Time series names are displayed in a hierarchical tree, whose branches are dynamically loaded when the tree is expanded. Two vertical axes are supported, furthermore correlation is also possible assigning time series to the horizontal axis. An example is shown in Fig. 3. Users can configure a wide range of parameters. All

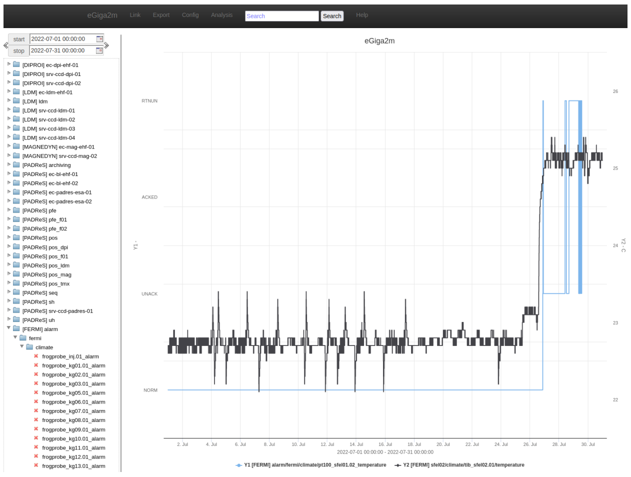


Figure 3: eGiga2m temperature plot with related alarm state.

settings are interfaced through URI parameters. This allows to send unique links to a specific graph, or exported files, and eGiga2m can be included, embedded or used by external resources. Configurations include:

- two decimation algorithms: maxmin and downsample,
- three graphic libraries: Chart.js, Flot Charts, Highcharts,
- several chart styles: scatter, line, spline, etc.

Some improvements available in the last release are:

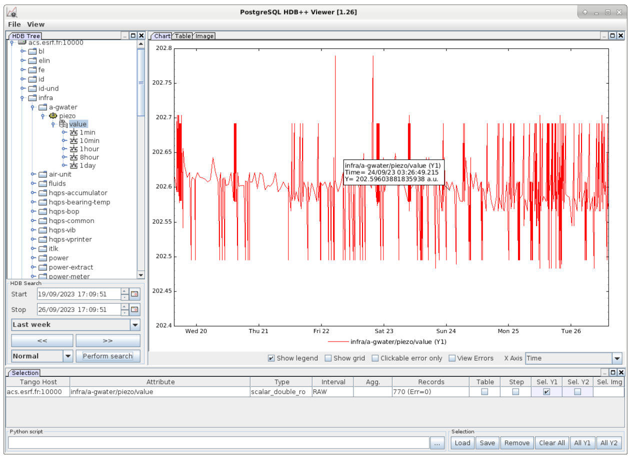
- support for TimescaleDB HDB++ back-end
- improved information in the chart tooltips, such as number of decimated samples displayed, total number of samples in the period, sample rate per second and query time

Archviewer

The *Archviewer* [7] is a web application for viewing HDB++ data archived in TimescaleDB. It supports multiple databases, and has a simple interface supporting ”pan/zoom” mouse operations. It relies on the database aggregation features to decimate data when its size reaches a given limit. Currently supported aggregation is min, max and average. It also supports downloading the plotted data as TSV or JSON. All parameters are stored in the URL to enable sharing and bookmarking. Archviewer currently only supports displaying *scalar* attributes, but *spectrum* attribute, e.g array, support is planned.

Hdbpp-Viewer

Hdbpp-viewer [8] is one of the first viewers, developed for HDB first and then extended to support HDB++. It is a standalone java application, as can be seen in Fig. 4, built on top of the java extraction library [9]. Hdbpp-viewer is available on maven central, thanks to the work done in the tango community to upload all java tools to maven central. It supports some HDB++ back-ends, like MariaDB/MySQL, TimescaleDB, and Cassandra. Advanced features are available for TimescaleDB, such as support for continuous aggregates and efficient index in array extraction. Hdbpp-viewer supports multiple data visualization on the same plot, two vertical axes, table view, CSV export, plot zoom, data selection and much more.



infra/a-gwater/piezo/value (Y1)
Time= 24/09/23 03:26:49.215
Y= 202.59603881835938 a.u.

HDB Search
Start 19/09/2023 17:09:51
Stop 26/09/2023 17:09:51
Last week
Normal

[✓] Show legend ☐ Show grid ☐ Clickable error only ☐ View Errors X Axis Time

Selection

Tango Host	Attribute	Type	Interval	Agg.	Records	Table	Step	Sel. Y1	Sel. Y2	Sel. Img
acs.esrf.fr:10000	infra/a-gwater/piezo/value	scalar_double_ro	RAW		770 (Err=0)	☐	☐	✓	☐	

Figure 4: hdbpp-viewer attribute plot.

pyhdbpp / AbstractReader

PyTangoArchiving python API and Reader objects have been used at ALBA Synchrotron to extract data from MySQL/MariaDB HDB and HDB++ archiving systems, either extracting raw data or linking graphical applications to the storage engine. Starting 2020, a new Python3 API, able to extract data from any HDB++ back-end, miming the approach adopted for the archivers that support different back-ends, has been developed. The AbstractReader class, provided by the python-pyhdbpp module, defines a minimum set of methods to be implemented for each back-end in order to provide data inspection, extraction and decimation from any Python application. The library ships with MySQL, MariaDB, PostgreSQL and TimescaleDB support out-of-the-box and encourages developers to write their own Reader objects to extract data from any other HDB++ back-end, or even other archiving systems. Moreover, the new multi-db schema allows to setup multiple database Reader objects dynamically, allowing queries in parallel to different engines.

Taurus

Latest releases of Taurus, the python graphical interface toolkit for Tango Control Systems, have shifted from Qwt / PyTangoArchiving plotting tools to pyqtgraph / pyhdbpp. Thanks to the new python API all graphical tools that used to be available for MySQL/MariaDB archiving systems are now available for any HDB++ back-end. Those tools include not only plotting but database browsing, using the tango browser tool to explore the attributes available in Tango and HDB++ databases.

LONG-TERM ARCHIVING STRATEGIES

Compression and Post-Processing in TimescaleDB

TimescaleDB development is quite active and offers interesting features for time series databases. Compression is a useful feature for large databases. TimescaleDB team boasts some interesting numbers in terms of compression, reproduced on test, as can be seen in Table 1 and production setups: compression ratio is good for scalar quantities, whilst the performance on array is much lower, with almost no compromise in terms of speed. TimescaleDB compression algorithm is so efficient that, on large data-sets, the time required to query the compressed data and uncompress is shorter than the time required to query the uncompressed data. In the production setup at the ESRF no impact on performances has been measured, while a lot of space is saved.

Table 1: Compression Ratio

Data type	Compression ratio	Speedup
Scalar double	11.14	0.69
Array double	1.54	0.98

With release 2.0 TimescaleDB introduced the concept of *jobs*. It is a generic mechanism to run a procedure within the back-end. Recurring tasks, such as continuous aggregation, chunks management, compression can conveniently be run as jobs. As already noted, within HDB++ some periodic tasks run as cron jobs: this is, for instance, how the Time To Live (TTL) feature is implemented, requiring extra infrastructure to run. HDB++ TTL feature is a perfect match for TimescaleDB *jobs*. TimescaleDB *jobs* design is generic enough to be adopted by developers for their own purposes. A custom procedure in PostgreSQL can be defined to run as a TimescaleDB *job* on a certain schedule. Some development has been done for data post-processing using this approach.

The amount of available data is increasing day by day. Extracting large amounts of data can become a challenging operation, in particular for visualisation purposes. Data decimation is an efficient and elegant solution that can be applied at different levels, but one of the most efficient is to decimate data running a post-processing *job* in the back-end. This approach keeps the data in its usual place, ready to be queried by the users. Currently TimescaleDB decimation *job* for HDB++ has been setup to generate decimated data while keeping the raw data as well, but different strategies can be easily implemented, such as pushing the decimated data to cold storage on to another database.

Multidatabase Setup

HDB++ is easily scalable by design; adding an EventSubscriber to handle more events is as simple as deploying a new Tango device. Concerning the back-end, it is possible to use more than one database to separate the data by subsystem or for performance reasons. Not all extraction tools are able to manage multiple back-ends, but the generic Python extraction tool, pyhdbpp, provides this support. This approach is convenient, as it separates the concerns effectively. System administrators can set up the databases as required, whilst the user can access the data in a transparent way. Exploiting this approach, interesting scenarios emerge, such as moving data between different clusters, with different specification, while decimating data in the process or setting up a small dedicated cluster to host applications with high archiving data rates but short time-to-live. The multidatabase setup enables great flexibility while providing the same interface to the users.

COLLABORATION

The HDB++ developers and users meet online on a regular basis, once every two months, to discuss issues share tools and methodology [10]. These meetings are often superseded by in person meetings where time is used to discuss the pressing issues. In 2023, Special Interest Group (SIG) meetings have been introduced within the Tango community; one SIG meeting, hosted by Astron in the Netherlands in November 2022, was dedicated to HDB++. The future roadmap of Tango archiving system has been discussed and the idea for the development of the SQLite back-end proposed. The HDB++ group is actively participating in the Tango community meetings, presenting the latest developments and animating workshops and discussions.

CONCLUSIONS

Using the strength of existing database engines, the HDB++ project built a complete ecosystem of tools and practices to ingest, manage, and extract data from the supported databases. More than a project, HDB++ is a thriving collaboration between several different institutes, which enabled to create a free, configurable solution to set up archiving for Tango as needed, with focus on solid back-ends, raw performance, simplicity of use, lightweight deployment and user requirements. HDB++ is improving through constant interaction within a free and open collaboration: your contribution is welcome.

ACKNOWLEDGEMENTS

The authors would like to thank the Tango Controls HDB++ community members for their contribution and great ideas.

REFERENCES

[1] HDB++ repository on gitlab, `https://gitlab.com/tango-controls/hdbpp/`

[2] HDB++ MySQL insertion library repository on gitlab, `https://gitlab.com/tango-controls/hdbpp/libhdbpp-mysql`

[3] HDB++ MySQL mono-repo on gitlab, `https://gitlab.com/tango-controls/hdbpp/hdbpp-mysql-project`

[4] Conda-Forge Community, "The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem", *Zenodo*, 2015. `doi:10.5281/zenodo.4774216`

[5] Archwizard repository, `https://gitlab.com/tango-controls/hdbpp/archwizard`

[6] yaml2archiving repository, `https://gitlab.com/tango-controls/hdbpp/yaml2archiving`

[7] Archviewer repository, `https://gitlab.com/tango-controls/hdbpp/archviewer`

[8] hdbpp-viewer repository, `https://gitlab.com/tango-controls/hdbpp/hdbpp-viewer`

[9] libhdbpp-extraction-java repository, `https://gitlab.com/tango-controls/hdbpp/libhdbpp-extraction-java`

[10] hdbpp collaboration minutes repository, `https://gitlab.com/tango-controls/hdbpp/meeting-minutes`