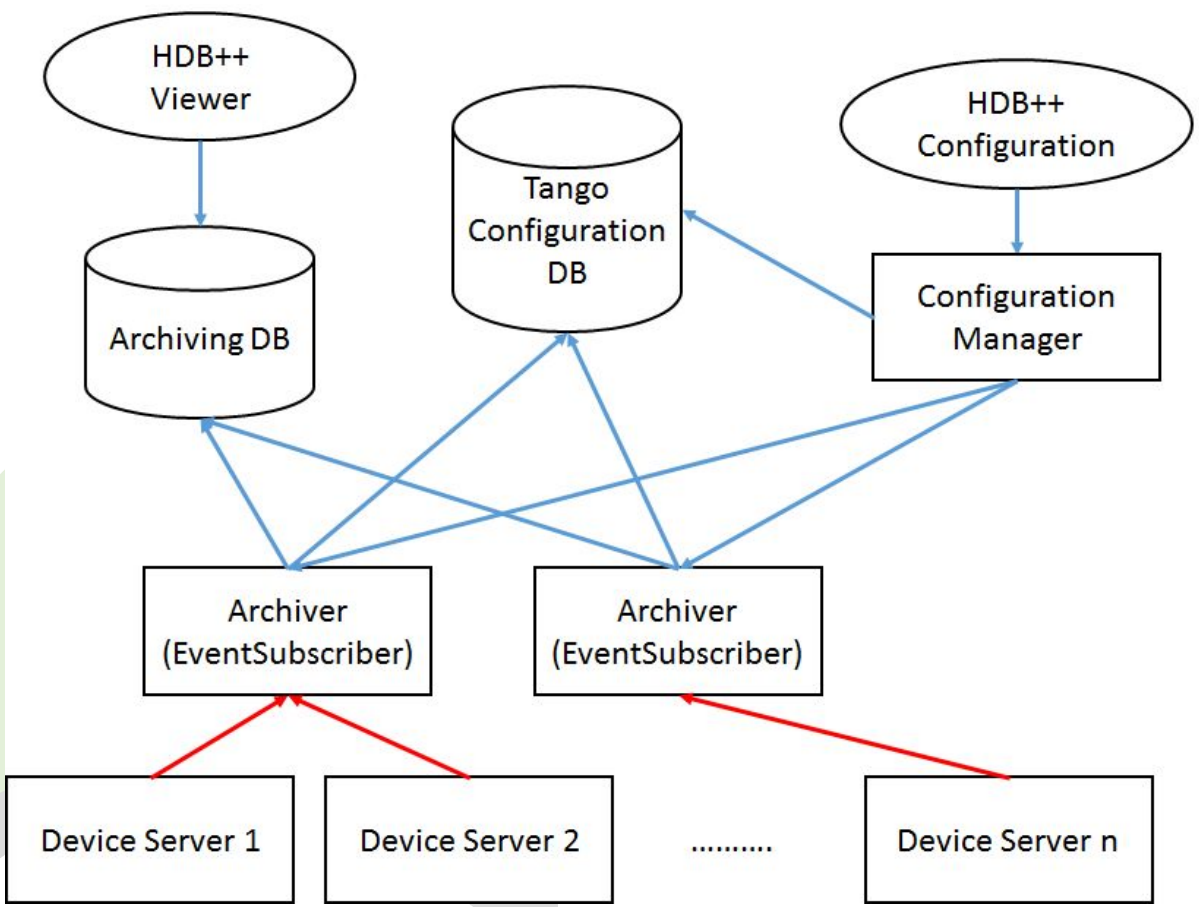
HDB++: What's new?

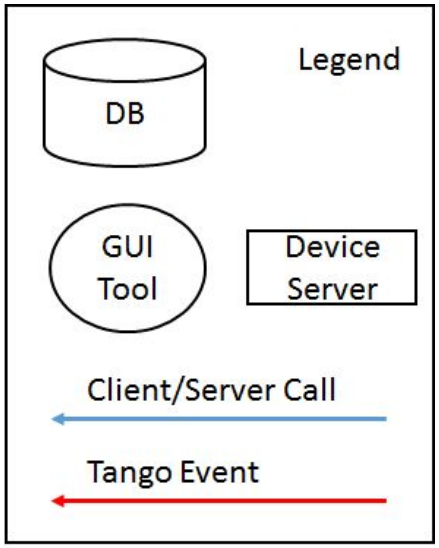
Reynald Bourtembourg
Johan Forsberg
Dmitry Egorov
Thomas Juerges
Damien Lacoste
Lorenzo Pivetta
Sergi Rubio-Manrique
Graziano Scalamera

HDB++ High-performance event-based Archiving System for TANGO

HDB++ is a DB-agnostic archiving system that inserts Tango Archive (or Change) events into your database of choice (TimeScaleDB, MySQL, SQLite, ...) using several dedicated Tango devices for event subscribing and configuration.

https://tango-controls.readthedocs.io/en/latest/tools-and-extensions/archiving/HDB%2B%2B.html





The number of attributes per EventSubscriber is tuned depending on your events/s ratio and DB engine (see each institute slides).

Alternate "mods" like change-event archiving, manual insertion and client-based polling are also available

Supported backends and clients

Backends:

- Mysql/MariaDB
- TimescaleDB
- Cassandra Deprecated!
- ElasticSearch Status unknown!
- Mysql/MariaDB Legacy schema Deprecated!
- Postgresql Status unknown, timescale library should be compatible.
- SQLite

Clients:

Extraction libraries:

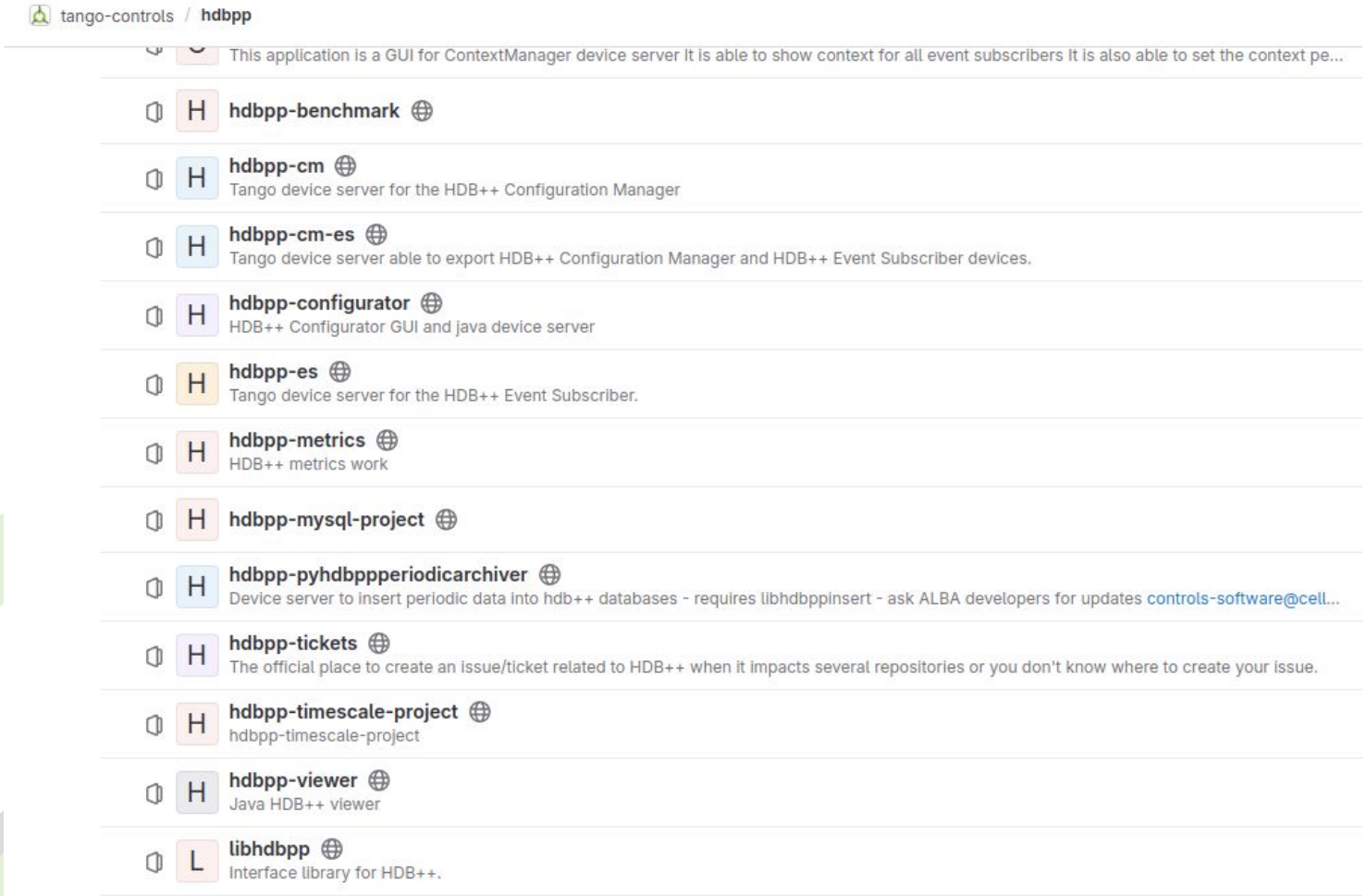
- Python extraction library
- Java extraction library, a matlab binding is available.
- Cpp extraction library, not up to date.

Full visualization clients:

- eGiga (web)
- HDB viewer (java)
- Grafana (web)
- archviewer (web)
- tango_browser (PyQt, taurus)

The HDB++ Ecosystem

All libraries and tools developed by the community are available at https://gitlab.com/tango-controls/hdbpp



Status: ALBA

ALBA - Accelerators

Running HDB++ since 2018 (HDB/TDB since 2008); configured using PyTangoArchiving, accessed using PyTangoArchiving and the new **pyhdbpp (libhdbpp-python)**

Main MariaDB host stores 6 months of undecimated data
(19779 attributes, 6 databases, 6TB in total, partitions every 15 days)

hdbacc: 943 attributes, 437 GB
hdbct: 3984 attributes, 576 GB
hdbdi: 3905 attributes, 2198 GB
hdbpc: 3127 attributes, 1112 GB
hdbrf: 3950 attributes, 525 GB
hdbvc: 4976 attributes, 1115 GB

Secondary MariaDB host stores all historical data decimated to max 1 value every 10 seconds (5TB in total since 2010).

Using 44 **CHANGE** Event Subscribers for latest devices and 52 Periodic Archivers for legacy systems.

ALBA - Beamlines

6 phase 2 beamlines (Tango9) using HDB++ (2034 attributes in 6 databases, using 189 GB)
7 phase 1 beamlines just migrated from legacy archiving to HDB++ last winter (but **notifd memleaks!**).

We use **PyHdbppPeriodicArchiver** and l**ibhdbpp-insert** for periodic/custom archiving insertion.
Old and new archiving working simultaneously while control system is migrated to Tango 9.

Status: Elettra

Fermi
HDB++ running since 2015
1 MySQL back-end, hdb++ schema
~17000 attributes from 8 Tango facilities
~9000 ev/minute; peaks up to 53.5K ev/minute
Context based archiving -> ~30 archiving strategies defined
59 EventSubscriber + 5 ConfigurationManager
~370 GB on disk - master (current + 2 previous years)
~370 + 870 GB on disk - replica

Elettra
HDB++ running since 2016
1 MySQL back-end, hdb++ schema (legacy HDB schema dropped 2021)
~6000 attributes
~5000 ev/minute
Context based archiving -> 7 archiving strategies defined
21 EventSubscriber + 1 ConfigurationManager
~460 GB on disk - master
~460 GB on disk - replica

Infrastructure (buildings facility)
1 MySQL back-end, hdb++ schema
~2800 attributes
5 EventSubscriber + 1 ConfigurationManager

ProxySQL
Used to make different DBs visible as a single one
For instance Fermi+Infrastructure, Elettra+Infrastructure, Fermi current (last 2 years) + Fermi archived (10 years) data

ESRF's Database setup (TimescaleDB)

- 1997 - 2018.
  - Moved to timescaledb backend.
  - 1 database engine with 2 databases.
  - 1.4To of compressed data + aggregates.
- From 2019.
  - Hdb++ with timescaledb backend.

- 19697 attributes, of which 18755 scalars, and 16881 doubles!
- 84 archivers.
- 3 configuration managers instances (45 devices).
- Database size ~ 7.3To for 5,5 years.
  - 2To compressed data (10To before compression).
  - ~ 500Go aggregates (not compressed yet).
- Stores about 700 events/s.
- Use of TTL with compression help reduce storage space but need to be carefully set up

Status: MAX IV

Finally shut down Cassandra around christmas 2024, now fully on TimescaleDB!

Configuration using “yaml2archiving”

Interest in archiving at beamlines is growing, work on standardization

Setup

One HDB++ setup (single archiver) per BL
One for accelerator (42 archivers, ~90% of data volume)
Single Postgres database cluster, 3 nodes (1 write, 2 read-only)
Separate schema for each control system

Some statistics

~ 20000 attributes
~ 3000 events per second
~ 20 control systems
~ 250 billion rows in TimescaleDB
~ 8TB disk space used (with compression)

Status: SKAO 1/2

Engineering Data Archive (EDA)

- Current deployments: SKA Low (Australia), SKA Mid (South Africa), CI/CD clusters
- Based on Kubernetes & Helm/Helmfile
  - Timescale server in dedicated namespace
    - PV for Timescale data storage
    - Individual DBs per deployment available
      - Not used :-/
      - Not good when used in CI/CD pipelines (one DB, many runs → lot or garbage in DB)
- Retrieval: archviewer & pyhdbpp

Status: SKAO 2/2

SPC cluster	MCCS cluster total	MCCS per station
60 devices	399 devices	79 devices
57 device servers	227 device servers	20 device servers
220 attributes archived	8149 attributes archived	1499 attributes archived

First results @SKA Low (~1300 of ~130k dipoles, "The 1% telescope" 😆)

- Single ES not enough (exactly as predicted by HDB++ community)
  - Will have to deploy more ES
    - Unfortunately deployment not flexible enough at the moment (Helm charts)
    - Currently "hacked" deployments

<u>Despite all the issues: Users are really happy to have historical attribute data</u>

Challenges:

- Data retrieval without leaking credentials
- "The data is not plotted the way I want it!"

5 YEARS WITH TIMESCALE!!!

- **The system is scaling fine, no decrease in insert or extraction time.**
- **Timescale proved to be a featurefull backend:**
  - Compression is very efficient.
  - Aggregates are a fast and efficient way to provide decimation
  - Easy configuration
  - Low maintenance, it just works
  - Jobs backend to ease automation
  - In place updates working flawlessly
- **Extensive with custom procedures for TTL, decimation and so on.**
- **Migration from Cassandra achieved for 2 institutes on big databases.**
- **Chunks size must be monitored as the system grow to avoid performances drops.**
- **Maintenance tasks, such as deleting attributes can take a lot of time and disk space, but they are manageable.**

Latest development news

- Libhdb++-timescale
  - Support for newer libpqxx version
  - MacOS support
- HdbEventSubscriber:
  - Bug fixes
  - Support latest cppTango version
  - Windows support merged
- libhdbpp-sqlite
- pyhdbpp (mysql/mariadb/timescale)
  - Grafana plugin
- Archviewer: multiple Y-axes support

libhdbpp-sqlite

- **libhdbpp-sqlite now available (for CI/CD, TDD, small setups, ...)**
- **Supported on Linux, macOS**
  - **Windows unsupported**
- **Try it and play with it!**

pyhdbpp

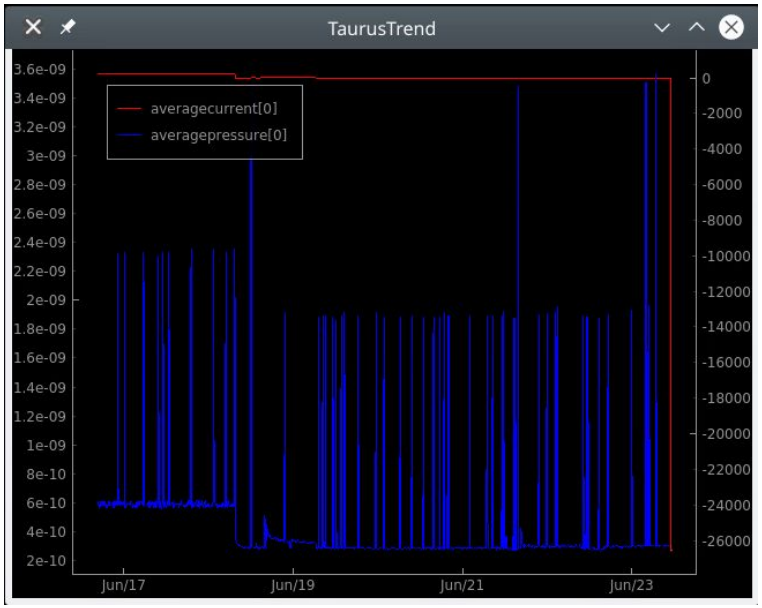
Python3 package for data extraction

> **pip3 install pyhdbpp**

https://gitlab.com/tango-controls/hdbpp/libhdbpp-python
https://gitlab.com/tango-controls/hdbpp/libhdbpp-tangobrowser

- Common API for MariaDB, MySQL and TimeScaleDB
- AbstractReader object provides generic extraction interface
- MultiDB reader allows to merge data from multiple data sources
- Dedicated DB back-end implementation is loaded at runtime
- Connection setup is stored in .yaml or Tango properties
- Taurus Widget available! (pyqtgraph)

```
import pyhdbpp

rd = pyhdbpp.get_default_reader()

rd.get_attribute_values('sr/vc/01/pressure',

                        '2023-06-03','2023-06-04')


Out:  ((1685782812.169, 1.4e-08, 0),

      (1685782842.011, 1.5e-08, 0), …
```



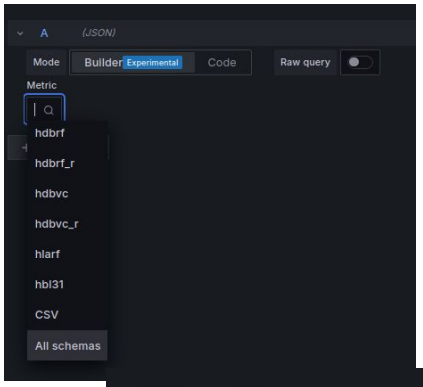
- Elastic? SQLite? … just inherit AbstractReader and contribute with your own implementation!
- New browser based on it : https://gitlab.com/tango-controls/hdbpp/libhdbpp-tangobrowser
- Configuration API still pending (PyTangoArchiving)
- Grafana plugin

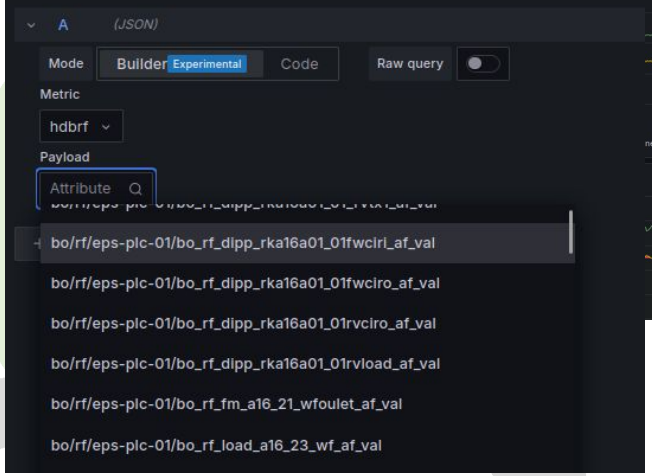
Pyhdbpp Grafana plugin

In addition to tango_browser and taurus, now grafana is available for plotting HDB++ (from mysql, mariadb, postgresql or timescaleDB backends).

Using pyhdbpp.multidb schema, it allows to plot data from multiple HDB++ databases.

https://gitlab.com/tango-controls/hdbpp/libhdbpp-grafana-connector



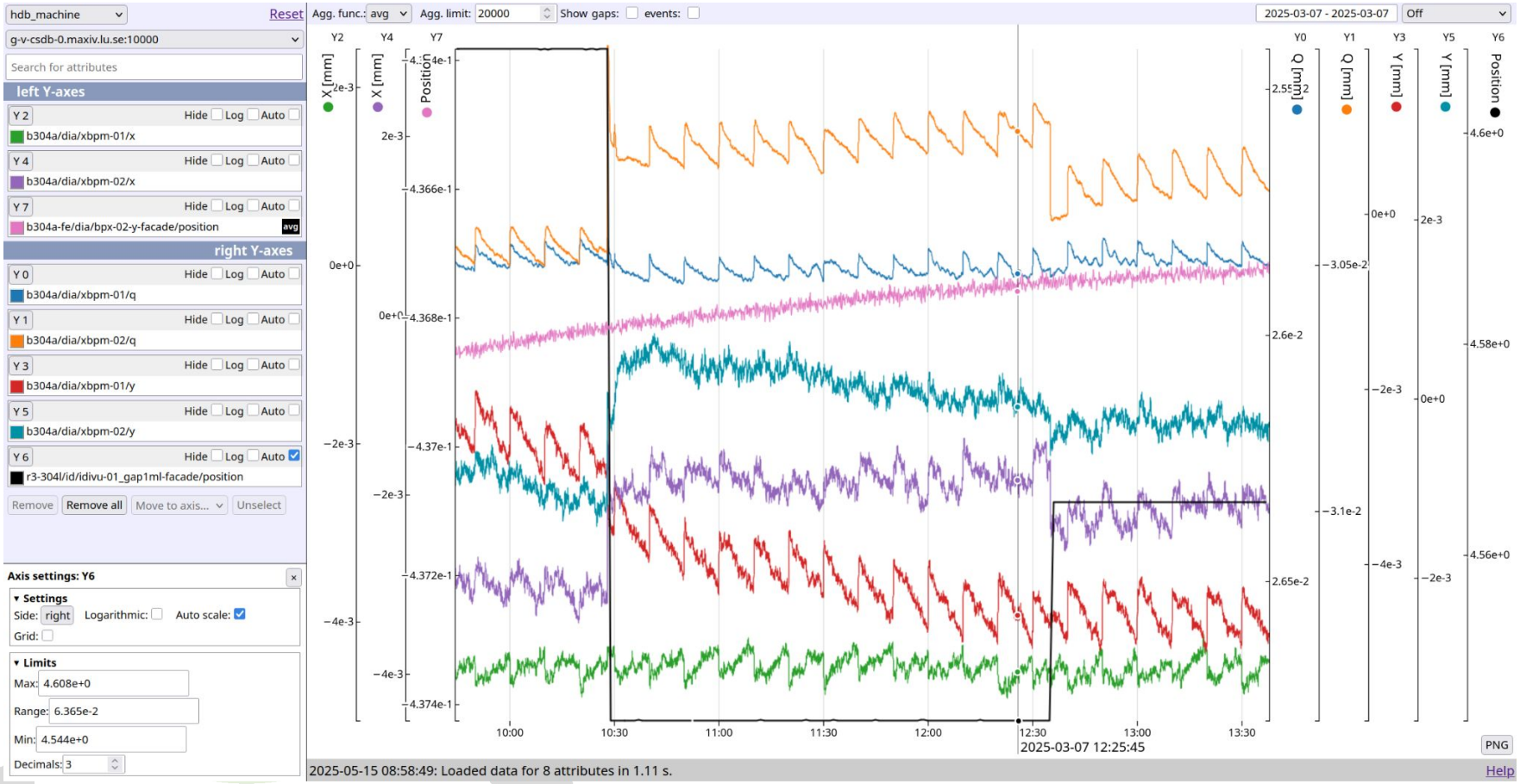


Archviewer 2.0

Web based HDB++ viewer, **supporting only TimescaleDB**

https://gitlab.com/tango-controls/hdbpp/archviewer (currently a mirror only)

New feature: support for any number of independent Y-axes



Conda packages

HDB++ archiver and configuration manager device servers (only for Timescale) available from Conda on conda-forge channel:

```
conda install -c conda-forge libhdbpp-timescale hdbpp-cm hdbpp-es
```

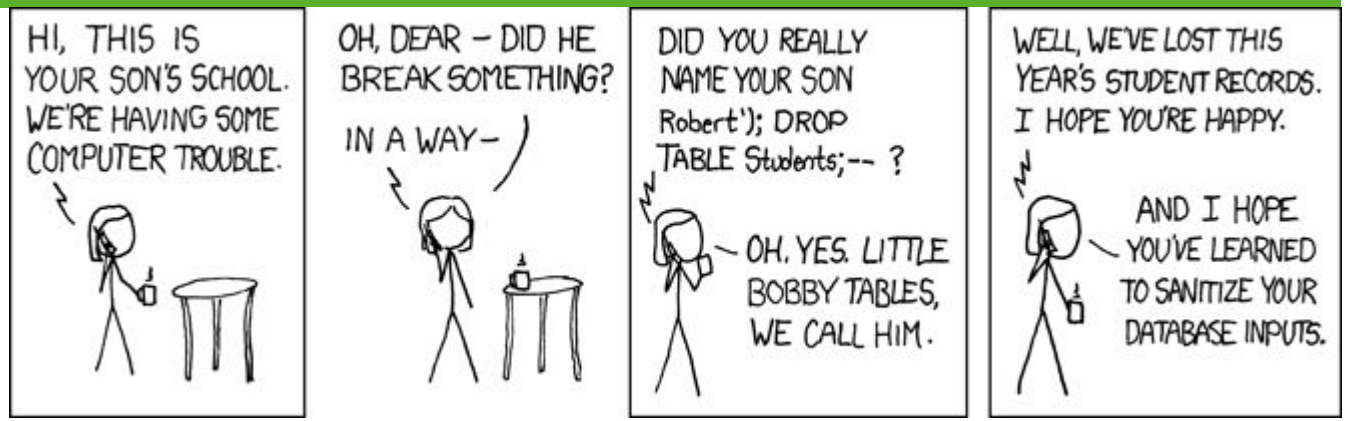
Debug versions also available:

```
conda install -c conda-forge libhdbpp-timescale-dbg hdbpp-es-dbg
```

Archiving visualization is available in conda via Taurus and pyhdbpp:

```
conda install -c conda-forge taurus pyhdbpp pyqtgraph
```

Upcoming in HDB++



https://xkcd.com/327

- **CI/CD**
  - In an effort to simplify and help installation CI will be set up on most of the hdb++ repos. This time we'll get to it!
- **Backend for a generic producer/consumer architecture**
- **Further integration of snap, time-machine into HDB++**
- **Explore the new Tango features:**
  - Alarm_event
  - Telemetry
  - Software versions

Current Databases Benchmark

Unifying the format in which we report our databases to facilitate comparison and architectural design for new facilities

(in progress)

Institute	ALBA	ESRF	SKAO MID-ITF	SKAO MID-PSI	SKAO LOW-ITF	SKAO AAVS3	ELETTRA	ELETTRA
Db Engine	MariaDB	TimescaleDB	TimescaleDB	TimescaleDB	TimescaleDB	TimescaleDB	MySQL/InnoDB	MySQL/InnoDB
Used for	vacuum	Generic	Generic	Generic	Generic	Generic	Fermi FEL	Elettra SR
mode	:hange_events, inserts	events	ents+periodic (periodic is discouraged)	+periodic (periodic is discouraged)	periodic is discouraged)	eriodic is discouraged)	archive events	archive events
Hdb++ version		1.x	current	current	current	current	current	current
Packaging/distribution	debian package	compile-in-place	docker images	docker images	docker images	docker images	compile-in-place	compile-in-place
OS	Debian 9	ubuntu 20.04	Kubernetes	Kubernetes	Kubernetes	Kubernetes	Ubuntu 18.04	Ubuntu 18.04
RAM	128 Gb	128 Gb	Varies(32G)	Varies(2x128G)	Varies(32G)	Varies(2x128G)	32 Gb master/64 Gb replica	16 Gb master/64 Gb replica
N CPUs	24	64	Varies(16)	Varies(2x64)	Varies	Varies(2x128)	8 master/4 replica	8 master/4 replica
Db Size (Gb)	514	4200	1.8	0.4	77	5	350	332
Timespan	3 months	4 years	4 months	6 months	9 months	1 month	2 years	2 years
N attributes	5159	17772					17326	6067
N attributes pushing events	4505	all?					all	all
N scalar double attributes	3461	15006						
from N devices	806	6579	~10	~10	~10	~10		
N subscribers	16	84	1	1	1	1	59	19
N attributes/subscriber	**322.4375**	**211.5714286**	**0**	**0**	**0**	**0**	**293.6610169**	**319.3157895**
partitioning	30 days	rom couple of days for double to a month					2 months for double, 1 year c	2 months for double, 1 year othe
max partition size	64G	~64Gb					29 Gb	35 Gb
events/second (hdb++es)	137.133	700	Not measured yet	Not measured yet	Not measured yet	Not measured yet	150	85
events/second (stored)								
events/second/attribute (average)	0.03						0.008	0.014
events/second/subscriber (average)	**8.5708125**	**8.333333333**	**#VALUE!**	**#VALUE!**	**#VALUE!**	**#VALUE!**	**2.542372881**	**4.473684211**
max events/second/subscriber	14.36						**16.5**	**17**
max events/second/attribute	1						2.8	1.5
max events/second/attribute (stored)								
max events/second/device								
bytes/event								
bytes/second	452							
	62018							

Get in touch with the HDB++ Community

- #hdbpp channel on Mattermost https://mattermost.hzdr.de/tango-controls/channels/hdbpp
- Get in touch with Reynald Bourtembourg to be added to the mailing list
- Report issues on https://gitlab.com/tango-controls/hdbpp/hdbpp-tickets
- Source code available under https://gitlab.com/tango-controls/hdbpp