HDB++: What's new?

Reynald Bourtembourg
Johan Forsberg
Thomas Jurges
Damien Lacoste
Jan David Mol
Lorenzo Pivetta
Sergi Rubio-Manrique
Graziano Scalamera

Status: ALBA

ALBA - Accelerators
Running HDB++ since 2018 (HDB/TDB since 2008); configured using PyTangoArchiving

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)

44 Event Subscribers + 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++ this spring.

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
~16700 attributes from 8 Tango facilities
~6500 ev/minute; peaks up to 53.5K ev/minute
Context based archiving -> ~30 archiving strategies defined
47 EventSubscriber + 5 ConfigurationManager
~350 GB on disk - master (current + 2 previous years)
~350 + 640 GB on disk - replica

Elettra
HDB++ running since 2016
1 MySQL back-end, hdb++ schema (legacy HDB schema dropped 2021)
~5700 attributes
~4700 ev/minute
Context based archiving -> 7 archiving strategies defined
21 EventSubscriber + 1 ConfigurationManager
~250 GB on disk - master
~250 GB on disk - replica

Infrastructure (buildings facility)
1 MySQL back-end, hdb++ schema
~275 attributes (new, growing to ~1000)

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.

- 16706 attributes, of which 16143 scalars, and 14068 doubles!
- 84 archivers.
- 3 configuration managers instances (45 devices).
- Database size ≃ 3.5To for 3,5years.
  - 983Go compressed data (4.8To before compression).
  - ≃ 700Go aggregates (not compressed yet).
- Stores about 700 events/s.

Status: MAX IV

Running HDB++ with Cassandra back-end, since late 2016

Currently running Cassandra and TimescaleDB **in parallel**
with “identical” config. So stats below are actually duplicated twice.

Data migration

Still ongoing, mainly limited by Cassandra readout problems
Beamlines done, machine ~50%.
Migration estimated to be finished during 2023.
Tuning of DB parameters…

Setup

One HDB++ setup per BL, one for accelerator (~90% of data)
Single database (cluster)

Some statistics

~ 20000 attributes
~ 2000 events per second
~ 50 archivers across 20 control systems

Status: SKAO

Engineering Data Archive (EDA) prototype ready

- Kubernetes & Helm
- Individual deployments (SKA-MID, SKA-LOW, SKA-MID-ITF, SKA-LOW-ITF, etc.)
  - Timescale in own namespace + persistent volume
  - Pods: HDB++, ArchiveViewer, ArchWizard,
  - Deployment as easy as

    ```
    make k8s-install-chart ARCHIVER_DBNAME=<dbname>
    ARCHIVER_TIMESCALE_HOST_NAME=<hostname>
    ARCHIVER_TIMESCALE_PORT=<port>
    ARCHIVER_TIMESCALE_DB_USER=<dbuser>
    ARCHIVER_TIMESCALE_DB_PWD=<dbpassword>
    ```
  - Configuration (upload, download, modification) with yaml file via web page: http://configurator.{KUBE_NAMESPACE}.svc.cluster.local:8003
  - Local pyhdbpp CLI supported VPN

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.

Clients:

Extraction libraries:

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

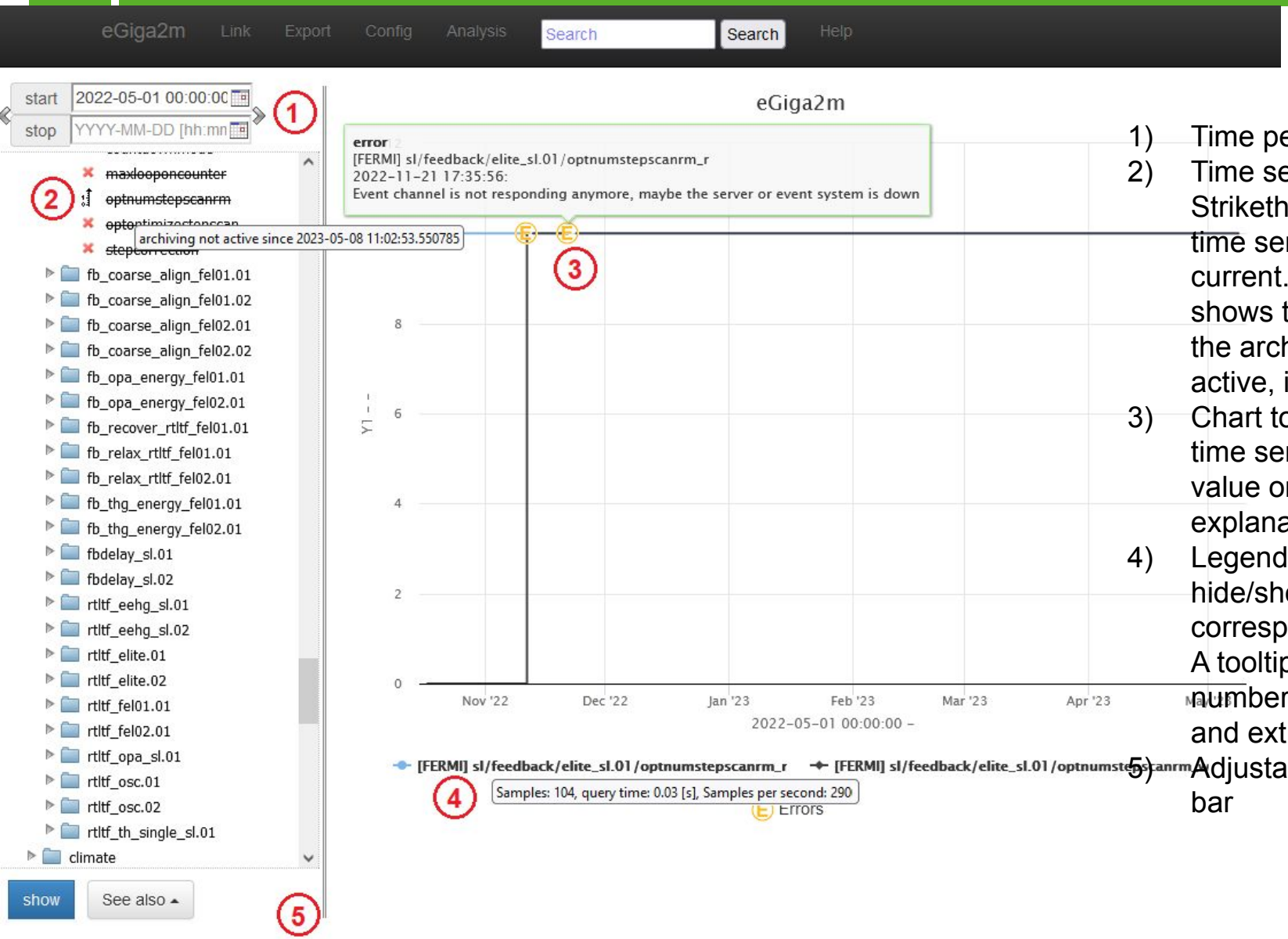
Full visualization clients:

- eGiga
- HDB viewer
- Grafana
- archviewer

Latest development news

- eGiga
- Libhdb++
- libhdbpp-timescale
- pyhdbpp
- Archviewer
- Latest development on TimescaleDB
- SQLite backend (current status of WIP)
- Conda packages

eGiga



1) Time period selection
2) Time series selection. Strikethrough means time series are not current. A tooltip shows the last date the archiviation was active, if available.
3) Chart tooltip shows time series numerical value or error explanation
4) Legend. Click to hide/show the corresponding curve. A tooltip shows the number of samples and extraction time
5) Adjustable separation bar

pyhdbpp

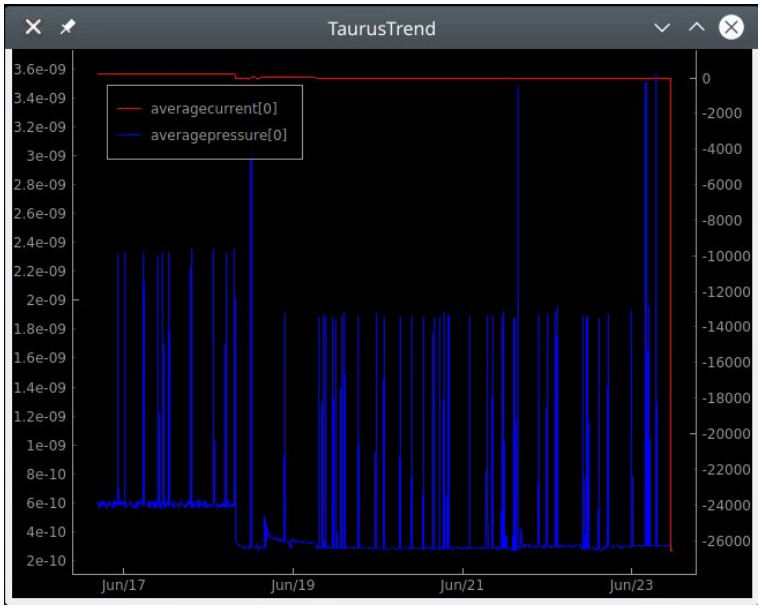
Python3 package for data extraction

> **pip3 install pyhdbpp**

- Common API for MariaDB, MySQL and TimeScaleDB
- AbstractReader object provides generic extraction interface
- 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), …
```



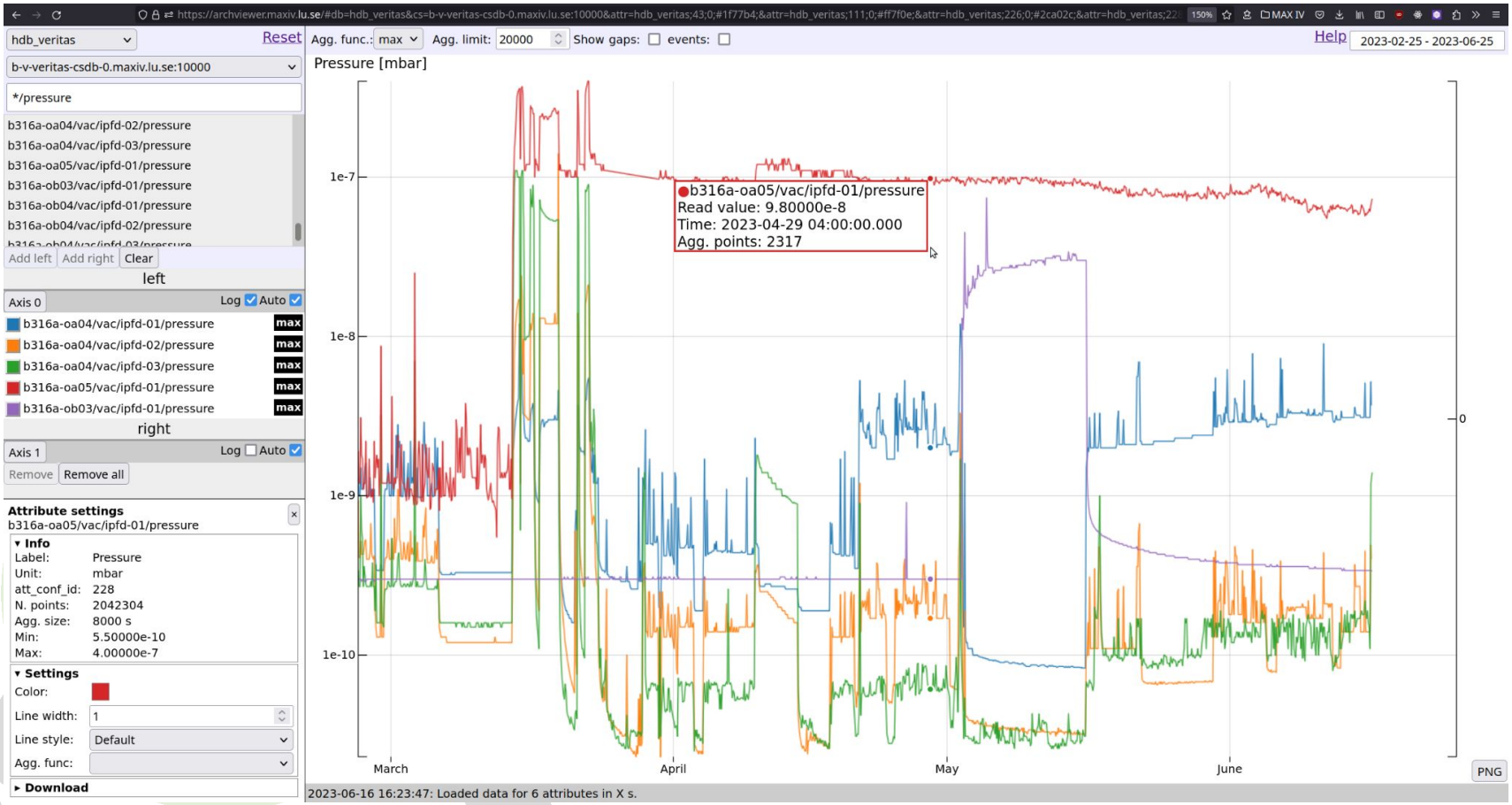
- Elastic? SQLite? … just inherit AbstractReader and contribute with your own implementation!
- ArchivingBrowser and SnapGUI also being migrated to python3 (ongoing)
- Configuration API still pending (PyTangoArchiving)

Archviewer 1.0

Web based archive viewer, **supporting only TimescaleDB**

Frontend based on (P)React and backend in Python

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



TimescaleDB latest development

- Compression support improved. It is now possible to alter, to some extent, data in compressed chunks.
- Added support for compression in aggregates.
- Multi node mode. So far with our current workload there is no need for such a setup, but it could be used to increase ingestion rate.
- Introduction of user-defined actions. Allows to launch postgresql procedure on schedule.
    - Will be used to replace python jobs for ttl and such
    - Could be used to perform post-processing, and manage data retention

SQLite Backend

Work in Progress by Lisa Banihachemi (Training Course at the ESRF)

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
```