Upgrade to Tango9 and HDB++ at ALBA
Upgrade to Tango9 and HDB++ at ALBA

+ some update on PyTangoArchiving developments
Tango @ ALBA Synchrotron

ALBA is a 3rd Generation Synchrotron Lightsource delivering beam for users 24/7 since 2012 to its 8 beamlines, with 3 more to be added in the next 2 years.

The whole facility is managed by several Tango Control Systems:

- Accelerator Control System (7084 devices) running Tango 7 on SuSE 11
- 5 Beamlines running Tango 7 on SuSE 11 and 12
- 2 Beamlines running Tango 8 on SuSE 11 and 12
- 1 Beamline running Tango 9 on SuSE 42.1
Tango 9 @ ALBA Synchrotron

ALBA participates in several open source projects within the Tango Community (Taurus, Sardana, Panic, Archiving), and the need to keep up with the current state-of-the-art software technologies became a headache in the last years.

Although we implemented continuous integration, testing and delivery (Jenkins, Travis, Git, etc.) the previous attempts to upgrade our systems were unsuccessful (Tango 8 stalled due to ZMQ and Tango 7 incompatibilities and SuSE Leap discarded due to lack of 32 bit support).

This delay to upgrade just amplified our gap with the state-of-the-art, forcing us to a full-stack upgrade: from Tango 7 to 9, from Taurus 3 to 4, and from a 2010 OS to the just-released Debian 9.
Tango 9 @ ALBA Synchrotron

Doing a full-stack upgrade of the whole facility was discarded due to lack of return on migrating the oldest hardware to a new OS.

Instead, our approach is to migrate all the clients and middle-layer devices, while keeping only the legacy hardware on Tango 7.

The upgrade basically consisted on:

- Migrate 80 applications from Taurus 3 to 4 (API Change)
- Package 224 additional device servers and dependencies as debian packages
- Test on simulators before deploying it in the control room
- Deploy and test everything again with operators
Tango 9 & HDB++, from polling to events

In parallel to migrating GUI's to Tango 9, we continued deploying HDB++ as the new archiving system for Accelerators.

It meant enabling event pushing in many devices, with some consequences:

- The CPU usage in IOC's increased, this behaviour was unexpected but we found out that was caused by the checking of change conditions on push_event.

- For many devices the pushed event was ignored by HDB++, as only change event was pushed. It forced us to have redundant polling or modify the devices.

- Clients were suddenly overloaded, either by too many events (diagnostics equipment) or by too many subscriptions (sleeps at subscribe time).
Tango 9 & HDB++, from polling to events

Only 1 database for HDB or TDB
10000 attrs on each

HDB

Tango DS

Control IOC

Java Archiver performs filtering, polling and insertion
Periods on 1-3 seconds

TDB

Archiver DS

HDB++ (~1000 attr)
HDBVCT (~3000)
HDBRF (~1000)
HDB < 2017

multiple DBs can coexist

Control IOC

Event-based
Filtering and/or polling done internally in the device server
Frequency depends on HW

Tango DS

Tango DS

Archiver DS
Actions taken

- Pull request done to cpp Tango to reduce the sleeps on event subscription from 10 ms to 1 ms.
- Added `push_archive_event(...,detect=False)` argument in those devices pushing at more than 1 Hz (change criteria to be managed internally by the device).
- Data reduction implemented on PyTangoArchiving.Reader, to reduce the number of points to be shown by trends.
- Data loading split in small intervals of fixed size, in order to avoid buffer overloading by a single query.
- CPU and Memory monitoring added to most IOC/CPCI hosts using ProcessProfiler device server (stats stored in HDB++ itself).
Issues mixing T7 & T9

We discovered that periodic event configuration only worked from Jive/PyTango when using exactly the same tango version used by the device server.

This problem seems solved since we upgraded the Tango DB to Tango 9; but we had to keep it running on an old server due to being tied by TANGO_HOST hardcoded on many client models!!

We also found archive event configuration less flexible than previous HDB configuration (e.g. you can set absolute change ranges of a DevLong as float with no warning, but events will not be pushed).

Other problems found (high cpu, events lost after a while) were solved by community pull requests on cppTango (many thanks!!).
Multiple HDB++ Databases at ALBA

Currently 5000 out of 10000 attributes has been migrated from HDB/TDB to HDB++. Maximum frequency is 20 Hz, but only few attributes are archived at these rates.

As frequency and volume of data increased, backups and partitioning of the database consume much more time (we add a partition for each big hdb++ table every month).

In addition, accelerators team wanted to increase the keeping period for diagnostics data (from 1 month to 6 months, ~2000 attributes affected).

We decided to split the archiving in 3 different MariaDB databases: diagnostics, radiofrequency and vacuum/protection systems. All three use MyISAM and do not store diagnostic timestamps.

From 200-300 attributes per event subscriber for diagnostics to 1500 attributes/subscribers for vacuum (slow attributes).
Only 1 database for HDB or TDB

Archiving01
HDB

Archiving03:
TDB
HDB < 2016

Control Workstation

Mambo / Taurus / Script

HDB OR TDB

Tango DB

Selection by origin and date, data can be concatenated

Multiple DBs can coexist

New Control Workstation

Taurus / Script / Web

Archiving04:
HDB++ (~1000 attr)
HDBVCT (~3000)
HDBRF (~1000)
HDB < 2017

New Control Workstation

Taurus / Script / Web

Only 1 database for HDB or TDB

Archiving01
HDB

Archiving03:
TDB
HDB < 2016
PyTangoArchiving 8: Multi-Database APIs

New APIs built on previous ArchivingAPI, ArchivingDB and HDBpp. The target is to provide a single set of commands to manage and extract attributes from different database schemas.

- PyTangoArchiving.Schema: allows setup of archiving schemas not only by db configuration but also by loading external python modules if needed.

- PyTangoArchiving.Reader: uses Schema config to retrieve archived data from different db's, querying several db's simultaneously if wanted.

- PyTangoArchiving.multi: implements "agnostic" methods for managing either HDB/TDB or HDB++ databases (start/stop archiving, check availability).
PyTangoArchiving 8: Multi-Database APIs

Free properties [PyTangoArchiving]

Property name | Value
--- | ---
hdb | reader=PyTangoArchiving.Reader('hdb')
hdb++ | db_name=hdblite
       | user=...
       | host=...
       | password=...
radb | check=attribute.startswith('radb/'),
     | user=...
     | host=...
     | password=...
     | reader=radb.Reader
     | method=get_device_values_2s
sqlserver |
radb |
hdb++ |
tdb |
hdb |

Schemas

sqlserver
radb
hdb++
tdb
hdb
Schema selection from Taurus ArchivingBrowser

<table>
<thead>
<tr>
<th>Label/Value</th>
<th>Device</th>
<th>Attribute</th>
<th>Alias</th>
<th>Archiving</th>
</tr>
</thead>
<tbody>
<tr>
<td>BO01/CCG-01</td>
<td>BO01/VC/VGCT-01</td>
<td>P1</td>
<td></td>
<td>HDBVCT/TDB/HDB</td>
</tr>
<tr>
<td>4.20e-10</td>
<td>millbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>BO01/VC/VGCT-01</td>
<td>P2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00e+00</td>
<td>millbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BO01/CCG-02</td>
<td>BO01/VC/VGCT-02</td>
<td>P1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.70e-10</td>
<td>millbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BO01/CCG-03</td>
<td>BO01/VC/VGCT-02</td>
<td>P2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.40e-10</td>
<td>millbar</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PANIC Alarms Logging with HDB++
PANIC Alarms Logging with HDB++

Formula:

\[ 0.5 \times \max \{ \text{sys/profile/alb.1/loadaverage} \} \]

Values of attributes used in the Alarm formula:

Trend graph showing data from ALBA03_CPU.
PANIC Convergence with AlarmHandler

- Elettra and ALBA had unified its criteria for defining alarms so both systems can interoperate following the IEC 62682 standard.

- Several bugs have been solved in the GUI thanks to the collaboration of the Tango community on github (Solaris, S2, tmc-ska, phiLAL).

Screenshot by Graziano Scalamera (Elettra), that successfully achieved to interoperate AlarmHandler and PyAlarm devices from the PANIC Gui.
Web Reports at ALBA

We do not have web-based tools to interact directly with the control system.

We provide indirect access, read-only, by extracting data from CS to .json files.

This .json files are then used as data-source by the different reports/applications.

Intermediate devices (WebTornadoDS/PyExtractor) are in charge of extracting/generating the needed .json files (parsing .html code for attr names).
PyTangoArchiving + WebTornadoDS Reports

WebTornadoDS generates new web reports on demand.

The device provides a web frontend to add new attributes to a cfg file.

Via PyExtractor, the DS will query the attributes to the archiving system (from a machine outside TangoCS).

Data will be exported to a .json file and later loaded by the web front-end.

To completely isolate CS and WWW, visualization/exporting can be separated in two instances, only sharing the cfg/json files.
Web Reports at ALBA

Current 179.86 mA

Operation mode
Top up

Lifetime
19h 52m

Avg. pressure
3.8e-10 mbar

Current x lifetime
3522 mAh

Beam for BLs
Time to inject: 00:03:42

Beamline Status
ID Gap

<table>
<thead>
<tr>
<th>Beamline</th>
<th>Status</th>
<th>ID Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL01</td>
<td>MIRAS</td>
<td>24.06</td>
</tr>
<tr>
<td>BL04</td>
<td>MSPD</td>
<td>B - 1.80 T</td>
</tr>
<tr>
<td>BL09</td>
<td>MISTRAL</td>
<td>5.86 mm</td>
</tr>
<tr>
<td>BL11</td>
<td>NCD</td>
<td>6.68 mm</td>
</tr>
<tr>
<td>BL13</td>
<td>XALOC</td>
<td>13.00 mm</td>
</tr>
<tr>
<td>BL22</td>
<td>CLAESS</td>
<td>41.52 mm</td>
</tr>
<tr>
<td>BL29</td>
<td>BOREAS</td>
<td>31.47 mm</td>
</tr>
</tbody>
</table>
Pending Things (help wanted)

Taurus Archiving schema, having to redefine the authority to accommodate multiple schemas

Partitioning and decimation, using TTL syntax as it is used in Cassandra schema

Using existing hdb++ extractors from PyTangoArchiving

Testing and CI!!

Python 3, fandango and panic already have python 3 branches, at least for the server side, so work on PyTangoArchiving can be started

Documentation!! See PyTangoArchiving_UserGuide.rst on github
PyTangoArchiving 8

This package allows to: * Integrate Hdb and Snap archiving with other python/PyTango tools. * Start/Stop Archiving devices in the appropriated order. * Increase the capabilities of configuration and diagnostic. * Import/Export .csv and .xml files between the archiving and the database.
Conclusions

PROs:

- Maximum archiving frequency increased from 1 Hz to XX Hz
- Devices capable of pushing events are not polled anymore
- Number of processes reduced, java removed from IOCs
- Splitting of databases facilitates maintenance

CONs:

- Volume of data to manage has increased dramatically in server and clients, this triggered refresh problems in trends
- Devices had to be modified to push both archive and change, to avoid extra polling
- Clients subscribing to many attributes became slower, sleeps had to be reduced
- Applications had to be adapted to work with sub-second timestamps (some errors on time operations)

push_archive_event(attr,implemented=True,detect=False) to enable high frequency pushing.
Questions?

https://www.tango-controls.org/community/forum/
https://www.pypi.org/project/pytangoarchiving
https://www.github.com/tango-controls/pytangoarchiving
https://www.github.com/tango-controls/fandango
https://www.github.com/tango-controls/panic