Tango Special Interest Group Meetings: What is going on?

Thomas Juerges (SKAO)

Why Special Interest Group (SIG) Meetings?

It always begins with a question:

- Where is Tango's protocol defined?
  - 👉 2022-06: RFC (S2Innovation, Krakow, Poland)
- How can I integrate Tango with Jupyter notebooks?
  - 👉 2022-09: jupyTango (SKAO HQ, Jodrell Bank, United Kingdom)
- How can HDB++ be used more efficiently?
  - 👉 2022-11: HDB++ (ASTRON Dwingeloo, The Netherlands)
- What is Tango's feature roadmap?
  - 👉 2023-05: IDL6 (ALBA, Cerdanyola del Vallès, Spain)

SIG Meetings so far

- How do we add encryption to Tango?
  - 👉 2024-03: Cyber Security (MAX IV, Lund, Sweden)
- Why is Tango's dated and sometimes incoherent documentation scattered over many locations on the web?
  - 👉 2024-10: Documentation workshop (ESRF, Grenoble, France)
- **What does Long Term Support (LTS) actually mean for our Tango LTS releases?**
  - 👉 **2025-11: LTS (DESY, Hamburg, Germany)**
- **How can we make the device server polling model more reliable?**
  - 👉 **2026-04: Device Server Polling Loop (MAX IV, Lund, Sweden)**

Long Term Support SIG Meeting @DESY

When, Where, Who

- 2026-11-03 - 2026-11-07 @DESY
- 12 participants
  - 8 on site
  - 4 remote
  - ALBA, byte physics, DESY, ELETTRA, ESRF, MAX IV, OSL, SKAO)

https://indico.tango-controls.org/event/521/

(Sorry for the bad photo that I took)

Proposed Features for next cppTango LTS version

👉**No fixed deadline, current LTS will be extended if needed (we would like it to be 2026 though)**

- Encryption (**sponsored by DESY**)✅
  - RFC, tests, implementation in cppTango (bonus Track: PyTango)
- Finish Open Telemetry
  - [Events (**needs a sponsor, i.e. RFC etc., will affect encryption**)]
  - Runtime configuration (almost merged)
- Dynamic attributes on device level (**sponsored by ALBA**)✅
- Code maintenance
  - Remove notifyd (TBD)
  - Deprecate DevPipe (compile warning, Pogo: nag screen that must be clicked away)
  - Refactoring of ABI (licensing, file split done, PIMPL ongoing)
- UTF-8
  - RFCs need to be updated (make encoding clear for all strings in Tango)
  - Migration path via new IDL version, preserve existing behaviour (new client/old device)

What does LTS mean for cppTango?

- What does it mean to support a first tier OS for cpptango:
  - We will provide patches for that OS during the LTS cycle for:
    - Bug fixes (if possible and deemed necessary by the community)
    - Security issues
  - Whatever OS version was picked at the begin of the LTS cycle will be the OS until the end of the LTS cycle, regardless of the OS vendor's support.
- What we will **NOT** provide:
  - Bug fixes that break ABI/API
  - New features
- What does it mean to support a particular version of an OS:
  - We will ensure that for the lifetime of the Tango LTS release:
  - Tango can be built on that version of the OS, using dependencies and build tools provided by that version of the OS (where available)
  - Critical bugs which effect that particular version of the OS will be fixed

Support time window

- **cppTango & pytango**
  - LTS Versions will be supported for 5 years
  - A new LTS version will be appointed every 3 years
  - Both LTS versions of cppTango & pytango will overlap for 2 years (to give time to switch versions)
- **❓ JTango ❓**

Example (not necessarily these versions):

2021	v9.3	-----	-----	-----
2022	v9.3	-----	-----	-----
2023	v9.3	-----	-----	-----
2024	v9.3	-----	-----	-----
2025	v9.3	-----	-----	-----
2026	v9.3	v11.1	-----	-----
2027	v9.3	v11.1	-----	-----
2028	-----	v11.1	-----	-----
2029	-----	v11.1	v13.0	-----
2030	-----	v11.1	v13.0	-----
2031	-----	-----	v13.0	-----
2032	-----	-----	v13.0	v15.1
2033	-----	-----	v13.0	v15.1
2034	-----	-----	-----	v15.1
2035	-----	-----	-----	v15.1
2036	-----	-----	-----	v15.1

First / Second / Third tier OS Support

- What does it mean to support an OS as "**first tier**" (Debian stable/oldstable, Ubuntu LTS)?
  - We will ensure that Tango LTS is supported on new (LTS) versions of selected first-class OS (not all).
  - For example, we support Debian as a first class OS. This means that when there is a new release of debian stable during the LTS lifecycle we will ensure that the cppTango LTS release builds and is tested on this new version of debian stable.
- What does it mean to support an OS as "**second tier**": (windows, macOS)
  - We will ensure that there is some version (probably the latest) of the OS where Tango LTS is supported.
  - Your mileage may vary on other versions of the OS.
- What does it mean to support an OS as "**third tier**": (FreeBSD, LE PowerPC Linux, ARM if we can't provide CI, ...)
  - No active support for any version
  - We will accept reasonable looking patches that are 100% compatible with the supported OSes (first & second tiers).

PyTango LTS Version

- PyTango LTS versions would be limited
  - Few releases, only for critical/security bugfixes
- Only a subset of Python versions will be supported with no new versions added
- Docker images for cppTango LTS will be kept to build PyTango on Python versions
- A PyTango LTS branch will be created to be able to submit MR and build wheel binaries from
- Tests will be run on MR only for the aforementioned subset of Python versions using Docker
- Platforms: at least Linux **x86_64** and Linux **aarch64** (using manylinux).
- PyTango will only publish source distributions to PyPI for LTS releases

Device Server Polling SIG Meeting @MAX IV

(Benjamin: Many thanks for the slides!)

When Where, Who

- 2026-04-14 - 2026-04-16 @MAX IV
- 14 participants
  - 12 on site
  - 2 remote
  - ALBA, byte physics, DESY, ELETTRA, ESRF, MAX IV, OSL, SKAO

https://indico.tango-controls.org/event/740/overview

(A much better photo taken by somebody else)

Tango device server polling in a nutshell

- Periodically executing command on a device
- Reading device attribute & storing the results (or the thrown exception) in circular polling buffer

Why device server polling?

- Speed-up response time for slow devices
- Keep history of device command outputs or attribute values
- Be data source for Tango event system
- Detect when something changes.

Periodic events are heavily relying on the polling to be done at precise time intervals

Tango Polling - Use case

Command polling exists, but not often used:

- Only commands without input argument can be polled
- It is not possible to poll the Init command

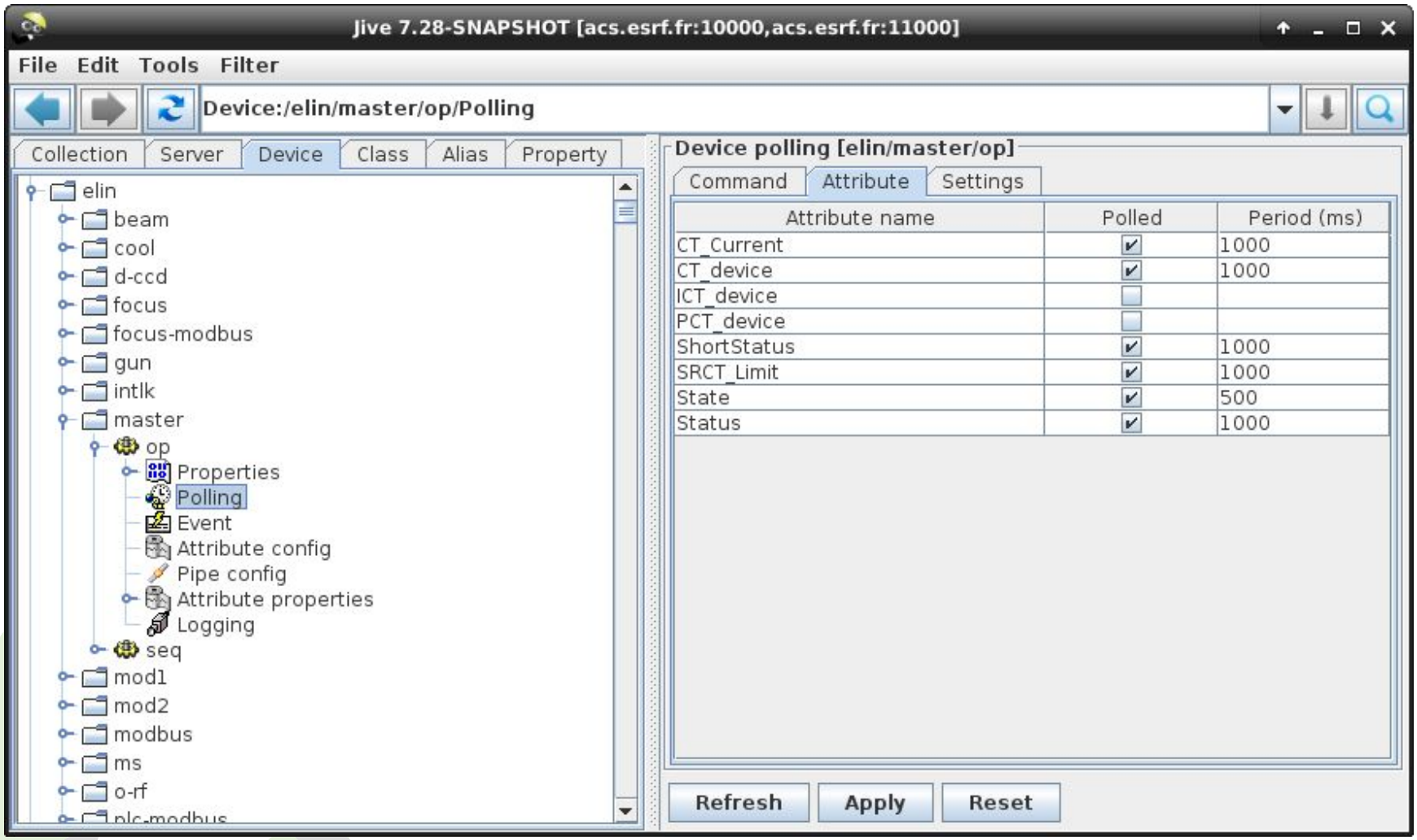
Attribute polling is very useful when:

- Hardware slow to read
- Device has many clients
- Data can be read from cache (last value read by polling)
  - But: History of read values is expected to have fixed time spacing (polling buffer)

When event communication is used:

- The event will be sent during polling phase

Tango Polling - Server Configuration

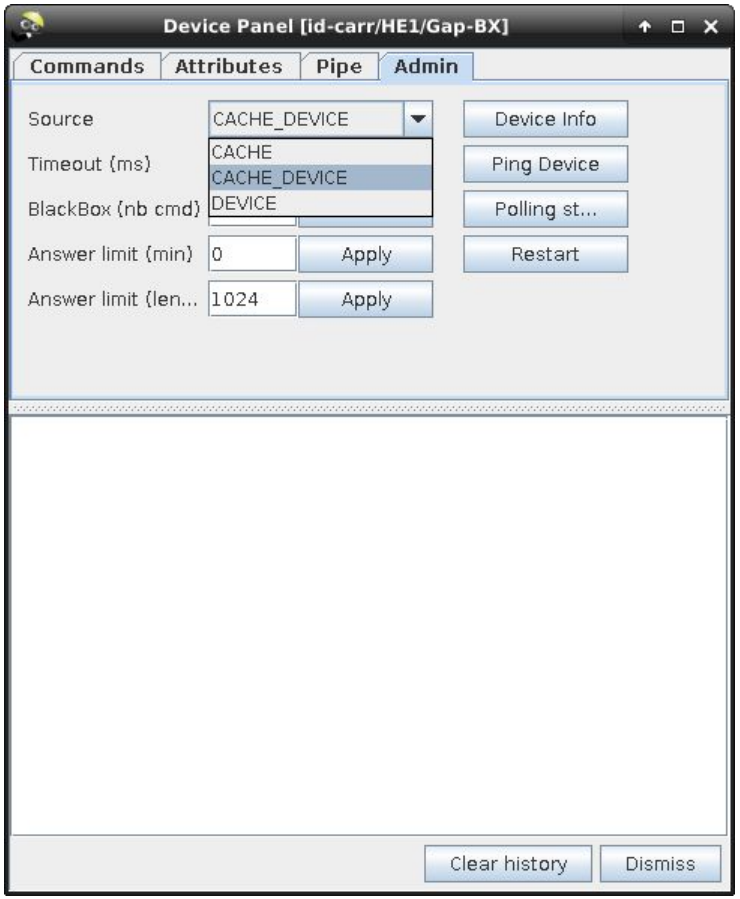


Reading an attribute:

A client is able to read data from

- The real device (DEVICE)
- The last record in the polling buffer (CACHE)
- The polling buffer and in case of error from the real device (CACHE_DEV)

The choice is selected by `DeviceProxy.set_source`



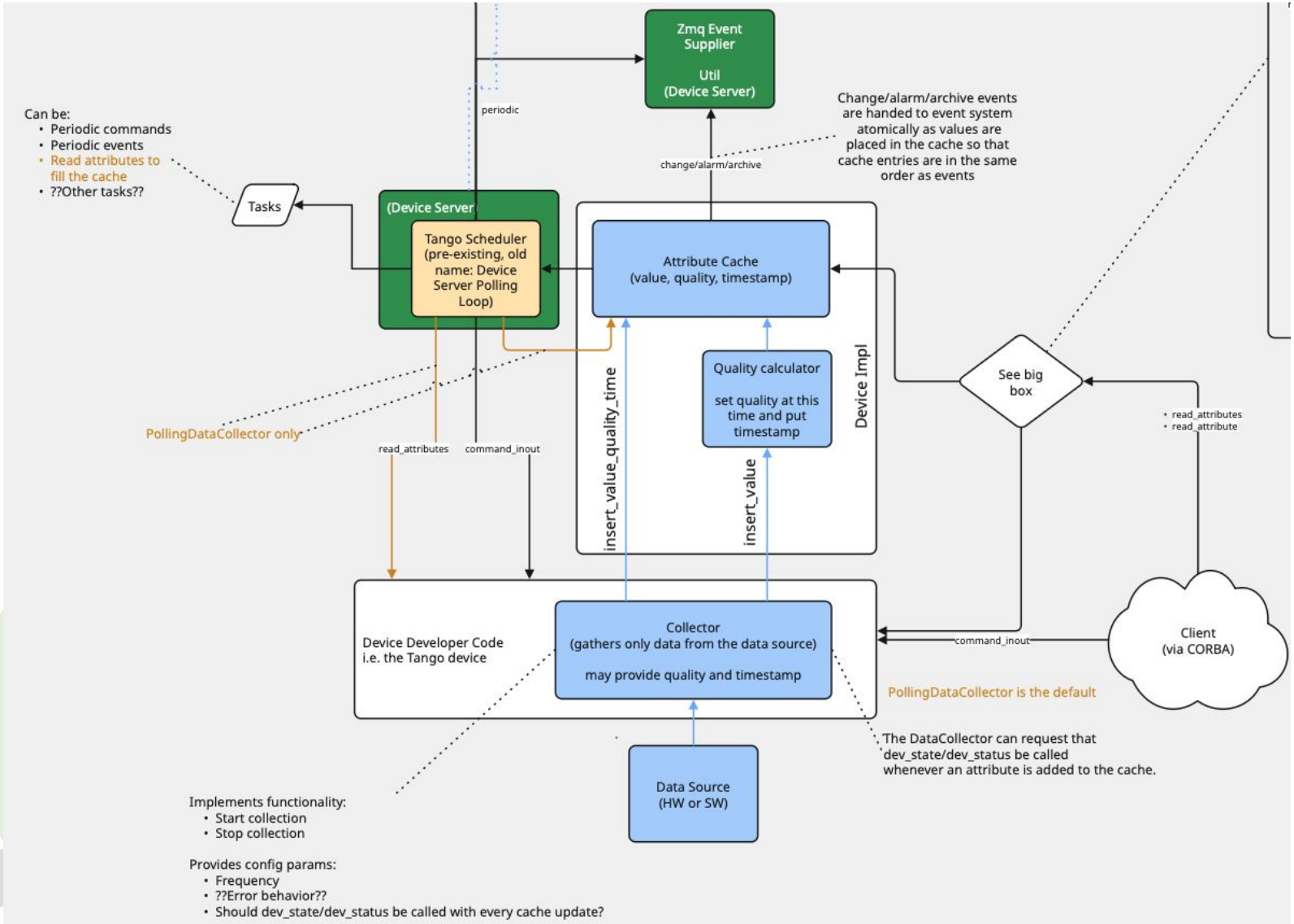
Deficiencies

- Difficult to tune
- Getting it wrong means items discarded
- Polling loop is not fair when it is mis-configured (punish items at the back of the queue)
- Client can get old data from cache when they expect latest (Sardana)
- Current polling has 3 responsibilities: getting data, putting it in the cache, periodic updates

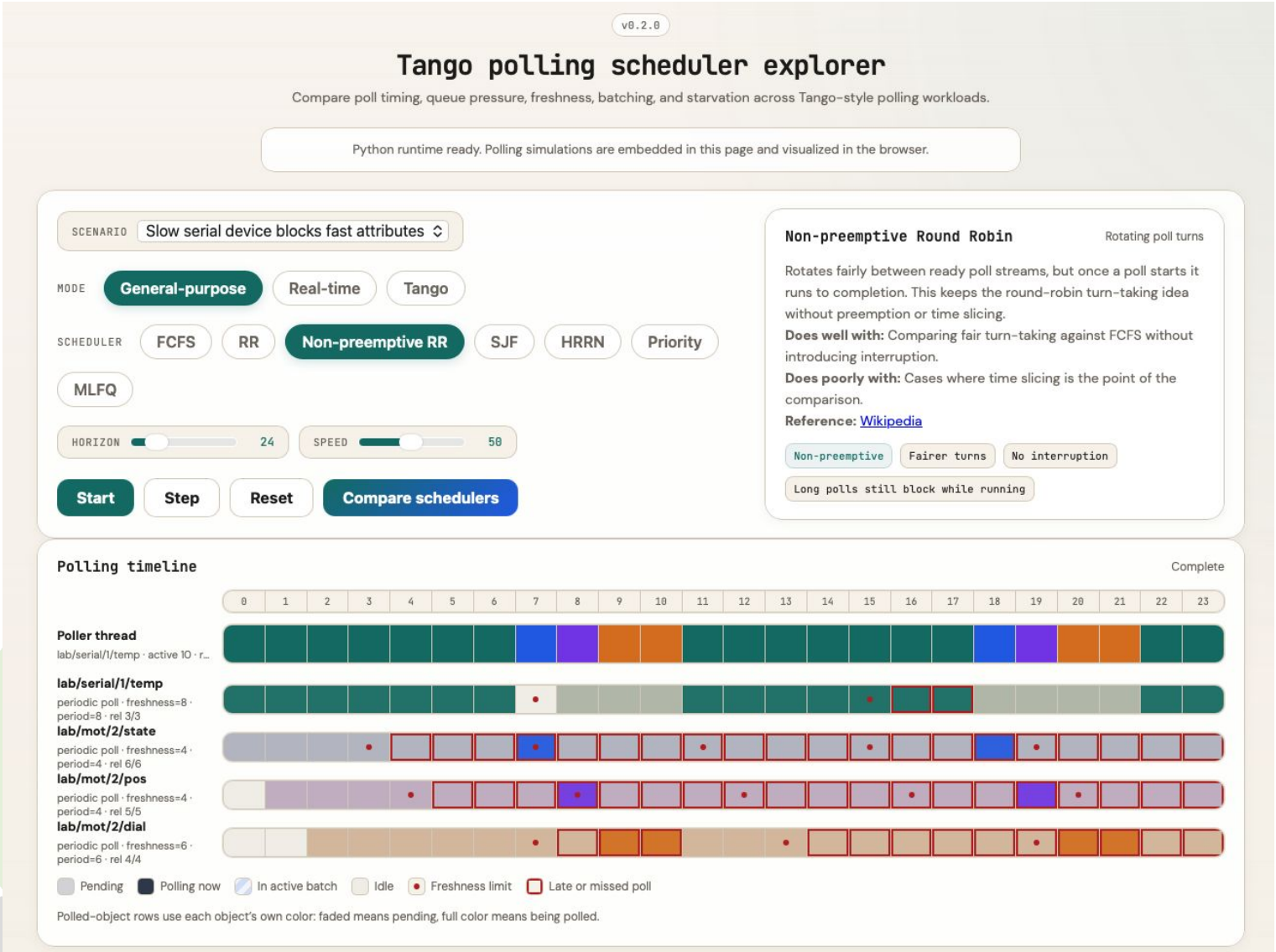
Way forward

- Cache separated from the polling loop
- Need backwards compatibility (allow old polling)
- Existing algorithm needs improvement

New Design


Zmq Event Supplier
Util
(Device Server)
periodic
change/alarm/archive
Change/alarm/archive events are handed to event system atomically as values are placed in the cache so that cache entries are in the same order as events
Can be:
• Periodic commands
• Periodic events
• Read attributes to fill the cache
• ??Other tasks??
Tasks
(Device Server)
Tango Scheduler (pre-existing, old name: Device Server Polling Loop)
Attribute Cache (value, quality, timestamp)
Quality calculator
set quality at this time and put timestamp
Device Impl
insert_value_quality_time
insert_value
See big box
• read_attributes
• read_attribute
PollingDataCollector only
read_attributes
command_inout
Device Developer Code
i.e. the Tango device
Collector
(gathers only data from the data source)
may provide quality and timestamp
command_inout
Client
(via CORBA)
PollingDataCollector is the default
The DataCollector can request that dev_state/dev_status be called whenever an attribute is added to the cache.
Data Source
(HW or SW)
Implements functionality:
• Start collection
• Stop collection
Provides config params:
• Frequency
• ??Error behavior??
• Should dev_state/dev_status be called with every cache update?

Tango polling scheduler explorer



Roadmap

- Changes in 10.4:
  - List of minor issues planned
  - Scheduler stats command?

- Changes for a future release (after 10.4.x):
  - New polling algorithm
  - RFC changes
  - Data collector
    - API changes
    - Cache changes

Bonus: What else did we discuss? State

PyTango: `DeviceProxy.state()` is different to `DeviceProxy.State()`!

❗ `state` is actually a "special" attribute in the IDL ❗

Three different ways of getting `state` from cppTango:

- CORBA `state`
- `State` command
- `State` attribute

Bonus: What else did we discuss? Timeout

DeviceProxy is in the same process as the Device?

Timeout doesn’t work

It becomes infinite! 😮