ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THCR003

A LONG TERM STORAGE SOLUTION FOR TANGO ATTRIBUTE DATA AT SKAO

M. Zambrano*, T. Juerges, SKA Observatory, Jodrell Bank, United Kingdom
J. Venter, SARAO, Cape Town, South Africa

Abstract

At the Square Kilometre Array Observatory (SKAO), monitoring data is ingested from distributed subsystems via the Tango Controls archiver, with attribute data stored in the Engineering Data Archive (EDA). The EDA uses a PostgreSQL database with the TimescaleDB extension, offering a performant solution for time-series storage. However, as SKAO infrastructure scales, PostgreSQL becomes impractical for long-term retention due to cost and operational complexity. This paper outlines a long-term storage strategy based on S3-compatible object storage. The solution decouples operational and archival storage by exporting and serializing Tango attribute data into efficient formats like Apache Parquet for storage in S3. Metadata indexing ensures the data remains discoverable and retrievable over time. The approach draws from the MeerKAT telescope's experience, a precursor to SKAO operated by SARAO. MeerKAT faced similar challenges archiving large volumes of telemetry data and adopted a database and long term storage model. We also describe supporting tools and processes for managing data lifecycle transitions. The paper concludes with open challenges and future directions for integrating this approach into observatory-wide data access frameworks, ensuring engineering telemetry remains accessible throughout the SKAO system lifecycle.

INTRODUCTION

Modern telescopes are complex distributed systems. They are made of thousands of networked independent devices created by different developer teams. Each single component in the telescopes can generate a myriad of continuous steams of telemetry data that can be used for:

- Operational Reliability & Troubleshooting: to check if a device is functioning according the the specification and take a corrective approach if needed. Historical data could help to detect failures due to environmental variables and schedule maintenance.
- Commissioning & Calibration: during construction each system has to be characterized and tuned to match its specification. Long term archiving could help in detecting deviation from the expected performance.
- Performance Optimisation: by having all the telemetry data, the telescope team could find operational improvements.
- Root cause and forensic analysis: after an issue is reported, the story on how it was triggered can be reconstructed from the archive and lessons can be learn from it.
- Data-driven maintenance: by using predictive algorithms in large time series, components failures can be scheduled.

The Square Kilometre Array Observatory (SKAO) [1] is currently constructing two large radio telescopes, SKA-Mid [2] and SKA-Low [3]. Its global headquarters is in Jodrell Bank, UK. The SKA-Mid telescope, which is currently constructed in the Karoo desert in South Africa, will be composed of 197 15-meters dishes. The SKA-Low telescope in Western Australia will have 512 stations that consist of 131,072 log-periodic antennas. During its construction and production, the telescope will produce an enormous amount of engineering data that will support scientists and engineers building the Observatory. All of these data need to be preserved for fifty years, which is the planned lifetime of the observatory, but there is no specification about the final format of the data. The SKAO Software component in charge of doing this task is the SKAO Engineering Data Archive (EDA) [4].

THE MEERKAT ENGINEERING DATA ARCHIVE

MeerKAT [5], an SKAO precursor, consists of 64 radio telescopes that collectively generate approximately 6,000 engineering data sensor samples every second. The following architecture is used to support the long-term storage of these samples.

Each dish includes several KATCP device components, each generating samples at various rates and writing them to a shared messaging queue (NATS [6]) as shown on Fig. 1.

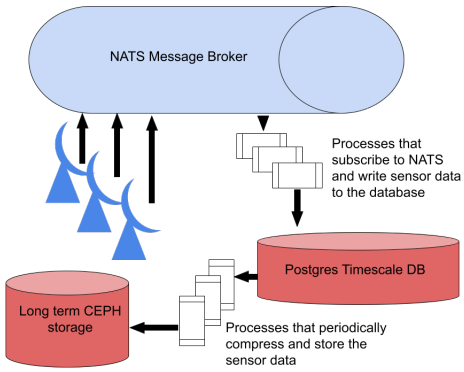

Figure 1: Schema of the MeerKAT engineering data archive.

A separate server runs multiple processes that subscribe to the message queue and write the incoming data to a buffer table in a database. Every few hours, all samples

* mauricio.zambrano@skao.int

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THCR003

older than one day are compressed and archived into block storage (CEPH [7]) from the database. The key used in CEPH for samples follows the following format <*days since epoch*>:<*sensor name*>, and the data is compressed when writing.

Clients access a single REST query interface that transparently searches and merges results from both the database and the object store, returning the data in JSON format. The REST interface supports custom filters.

THE SKAO ENGINEERING DATA ARCHIVE

The SKAO Engineering Data Archive is a scalable solution that consists of open source software components. Most of them are provided by the Tango Controls collaboration [8–10] as part of HDB++ (Historical Database Plus-plus) [11, 12]. They are:

- HDB++ Configuration Manager [13]
- HDB++ Event subscriber [14]
- libhdbpp-timescale [15]
- TimescaleDB [16]

HDB++ Configuration Manager

The HDB++ Configuration Manager is a component of the Tango Controls system designed to configure and manage the archiving of Tango attribute data. It is a key part of the HDB++ archiver suite, which is responsible for storing the event-based stream of telemetry data from devices (called Archive Events) within a Tango-based control system.

The core function of the Configuration Manager is to provide a user-friendly interface for engineers to:

- Specify which attributes to archive: Engineers can select specific device attributes (e.g. a motor's temperature, a sensor's voltage) that they want to monitor and store.
- Set archiving parameters: For each attribute, they can define the archiving rate (how often a value should be recorded), if the value should be stored on change or both. This allows for flexible data collection, from high-frequency real-time data to low-frequency long-term trends.
- Manage the archiver's state: It allows for starting, stopping, and viewing the status of the archiving process for different attributes or groups of devices.

In essence, the HDB++ Configuration Manager acts as the central control panel for the data archiving pipeline, allowing operators to control what is stored in the backend (e.g. TimescaleDB) without having to manually edit configuration files or interact directly with the backend.

HDB++ Event Subscriber

The HDB++ Event Subscriber is a component of the Tango Controls system that is responsible for listening to Archive Events published by Tango devices. These events contain attribute data, which the Event Subscriber submits to the storage backend. Therefore it acts as the primary data ingestion point for the HDB++ archiving solution. Note that it is not essential that an HDB++ Configuration Manager is running in a Tango Controls system. One can configure the archiving of attributes with an HDB++ Event Subscriber alone, but as soon as multiple Event Subscribers are needed for load-balancing, a Configuration Manager becomes a vital part in the Tango Controls system.

How It Works

- Subscription: The HDB++ Event Subscriber connects to one or more Tango devices and subscribes to Archive Events of attributes of these devices. The events need to have been configured for archiving.
- Data reception: When a device's attribute changes value, or a specified amount of time has passed, the Tango device's event system publishes an Archive Event. The subscriber receives this event, which contains, among other information, the attribute's name, its new value, and a timestamp.
- Data Ingestion: The subscriber processes the received data and writes it to a designated backend. Typically this is a database like PostgreSQL with the TimescaleDB extension. Other backend options are available.

In simple terms, while the Configuration Manager tells the system what to archive, the Event Subscriber is the worker that actively listens for Archive Events and writes the received data to long-term storage. As mentioned before, multiple subscribers can be run to handle high volumes of data and ensure redundancy.

libhdbpp-timescale

Libhdbpp-timescale is a Tango Controls library that serves as the interface for the database backend for the HDB++ archiver. Its main purpose is to provide the functionality for the HDB++ Event Subscriber to store time-series data into a PostgreSQL database with the TimescaleDB extension.

Key Functions

- Database Interface: It provides an abstraction layer that allows the HDB++ archiver to communicate with a PostgreSQL database with a TimescaleDB extension without needing to know the low-level details of the database's schema or query language.
- Data Ingestion: The library handles the efficient writing of Tango attribute events (data points) into TimescaleDB's hyper-tables. It is optimized to take advantage of TimescaleDB's features, such as data compression and chunking, for high-performance ingestion.
- Schema Management: It defines the database schema required for storing Tango data, including tables for attribute values, timestamps, and metadata. This ensures that the data is stored in a structured way that is optimized for time-series queries.
- Query Capabilities: It also provides the ability to query historical data from the database, allowing other components or tools to retrieve archived attribute values. Tools of the tango ecosystem typically require this library to retrieve the monitoring data.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THCR003

In essence, libhdbpp-timescale is the specific, optimized connector that bridges the generic HDB++ archiving system to the powerful, time-series-oriented storage capabilities of TimescaleDB.

TimescaleDB

TimescaleDB is a time series PostgreSQL database extension created in November 2018 by TigerData (formerly known as Timescale Inc) under a sourceavailable license. You can use it for your project but you cannot offer it as a service to third parties. They also offer the core version under the Apache 2.0 license. A time series table in TimescaleDB is called an hypertable. Each data table should have a way to be partitioned, for example, by date. You define a partition size called chunk and you create a policy for it. This policy triggers a periodic job on older data. Jobs are managed by its own scheduler. TimescaleDB uses a worker thread for each job, so you should probably tune their values from the PostgreSQL defaults to match your setup. If a job fails, it will be re-tried later. A typical failure could be that the job needs extra temporary space to write a compressed chunk. There are some recommendations about the size of the chunks. Active chunks should fit in 25% of the memory of the database server to deliver good performance when querying the data.

SKAO EDA: An Integral Part of the Observatory

The SKAO EDA is the software component at the SKAO that is responsible for collecting and preserving engineering telemetry data from the SKA-Mid and SKA-Low telescopes for long-term use.

It is a crucial system for the observatory's operations, as it stores a continuous stream of data from thousands of devices, which is used for:

- Operational reliability: Monitoring device health and detecting failures.
- Performance optimization: Finding ways to improve the telescope's efficiency.
- Troubleshooting: Reconstructing events to perform root cause analysis after an issue.

The EDA is a hybrid system built on open source technologies, including Tango Controls components and a PostgreSQL database with the TimescaleDB extension, which has already been described earlier. This system is crucial for Scientists and Engineers during the construction of the telescopes.

THE SKAO COMPUTING ENVIRONMENT

The SKAO runs its computing resources on top of Kubernetes. The preferred way to run services there is via operators. They allow us to do repeatable installations of high available databases, backups to S3, major and minor upgrades among other things. We have selected the Open Source StackGres operator as our main operator for PostgreSQL since it provides a curated set of extensions, including TimescaleDB and there is also good support from the StackGres developers and community. It uses Patroni as the main high-availability (HA) technology. Patroni is a Python-based open source tool for creating and managing a HA PostgreSQL cluster. It is a template for building a custom, HA solution for PostgreSQL using a distributed consensus system.

SKAO CHALLENGES

There are many archival challenges during the telescope construction phase. We will list them and review some of the approaches we are using to tackle them.

The Amount of Data to Store

During the construction phase, at SKA-Low a single testing station has 16031 engineering monitoring points with only four of the 512 planned stations. The typical data rate is between 0 and 11 transactions per second. We could be storing all this data, but eventually we might run out of physical space when trying to add more disks or run out of budget to buy more disks.

TimescaleDB provides compression and decimation policies to manage its growth. In our tests, the SKAO data compression ratio is between 87 and 1.2 depending of the table as seen in Fig. 2. Another important parameter is the size of the TimescaleDB partitions called chunks. We started by using a default of 28 days. It created operational issues given the large amount of data we are receiving. When compressing data, TimescaleDB has to read old original data to create the new file. It creates temporary files where the data is compressed. Then the old data is removed from the original uncompressed table. This requires to tune the maximum size of the temporary files to up to 50GB from the defaults and to wait for a vaccum process to run if too many rows are deleted. We reduced the size of the chunks to one day to handle this process in smaller sizes.

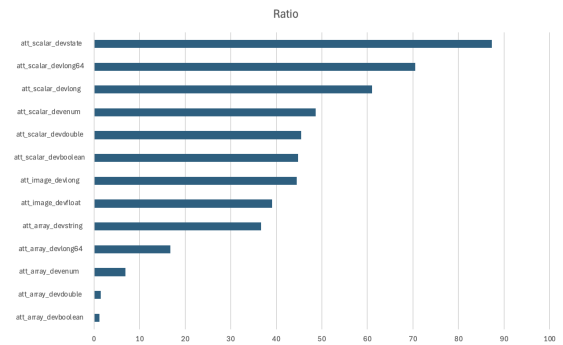

Figure 2: Current compression ratios with no reordering applied.

There could be some room to improve compression ratios by executing first chunk reordering. This operation rewrites the chunk content by putting together devices with their collected data arrays as explained in Fig. 3. This process also improves queries performance since you usually do queries on a single specific device during a period of time.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THCR003

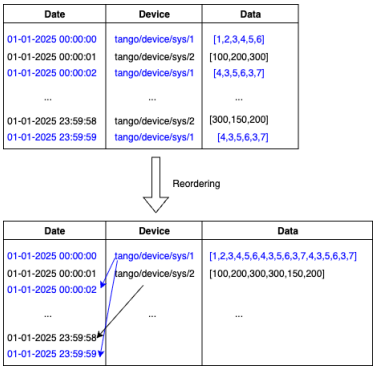

Figure 3: Changes on the original table chunk while reordering. Data related to specific devices is kept contiguous. That would increase usual queries.

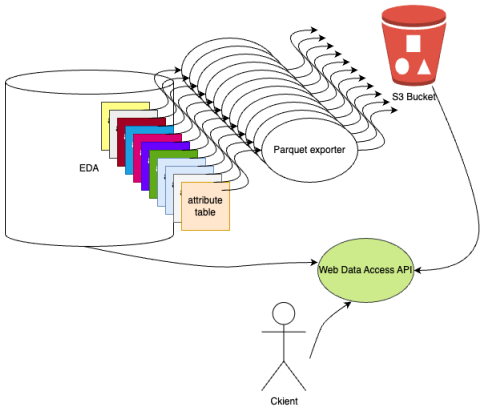

Figure 4: Schema of an EDA DLM solution with the parquet exporters retrieving data from the EDA and saving parquet file on an S3 bucket. An API can do queries on the S3 parquet files and the online EDA.

EDA Data Life-Cycle Management and Growth Management

Three approaches will help us. First, monitor the ingestion rates. For each HDB++ table, get the ingestion rates by device. This will also help to get a clear idea of what would be a useful data rate for debugging and during normal operations. We have created a Python exporter and a Grafana dashboard to monitor devices with attributes with high numbers of archive events. Integration teams should take actions to adjust the events' rates. This would also prevent us from leaving debugging monitoring configurations for too long. Second, decimation policies to limit the amount of data storage. TimescaleDB provides policies to decimate or down-sample data. Integrations teams have been asked to look with engineers and scientists at what original data should be preserved and what data should be preserved only decimated in the long term.

Finally, we can export, before decimation, old data in full to another format, like Apache parquet and move it, out of the database, to an S3 bucket as shown in Fig. 4. We have our own tool [17] to do this. A Kubernetes job can run on the EDA to export the data for each table. Each HDB++ table is kept as a parquet file on its own directory and data is distributed by day. Parquet files must be created only after all the data on the source table has been made available since you cannot update them later.

There is also an important discussion between engineers and scientists taking place to understand what is required by each team to do their job. Sometimes, the initial archival rates are not what is needed for debugging. There will be an agreement on the required configuration and the data format needed to store the long-term decimated data on the online EDA on PostgreSQL. There is no obligation to keep the data with the same format once it has been decimated. The decimation in TimescaleDB is a process triggered by a materialized view.

This solution also includes a web API [18] that knows the location of all the S3 files at startup and that can query data on parquet files and PostgreSQL live data when a query is requested. Since the structure of the parquet files is known, the location of a specific S3 file can be dynamically built.

Local Deployment Requirements

These are challenges related to specific telescope requirements. For example, at SKA-Mid, each antenna has a single self-contained EDA on the only server that is installed in the base of the antenna's pedestal. This system must be able to work without an external network connection for an extended period of time. This EDA ingests all the archive events from all Tango devices that run on the same server. EDA compresses the data after one day. In case of a network outage it could send the data eventually back to a central archive. The current deployment emulates a single-node EDA database deployment with PostgreSQL and TimescaleDB. Due to their restricted resources and small footprint, it would be an attractive solution to create a standalone libhdbpp-parquet that would require no database at all and fewer resources. Some initial work has been done, but it should be extended to handle all the range of complex data types that Tango supports.

During commissioning, data could be stored on the pedestal server or on a central server. The archiver's configuration and the database configuration have to be able to handle all the combinations of operating modes required by the integration team. They need to be able to store data remotely on a central EDA, on a separate database for each device, or on the same EDA.

CONCLUSIONS AND FUTURE WORK

When monitoring data requires a huge amount of data and it becomes hard to manage in traditional ways, architectures that can delegate storage growth to specialized systems can prove to be convenient to simplify the problem. That has been the case at the Meerkat with Ceph and with S3 at the SKAO.

During the construction of a telescope, there are still important parameters to be tuned. Some agreement between

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-THCR003

the Engineers and Computing teams must be reached to achieve a meaningful monitoring policy that is economically viable. This will probably also define the format and rate of the data in the long term.

We have an initial DLM solution that could be extended to keep all the original data in an S3 bucket. We will also explore other data formats like iceberg to see if we can improve S3 file generation. We are currently self-hosting the S3 buckets via an S3 ceph gateway but we might also use cloud vendors service to keep our data. We should also plan how we will present the data to any predictive tool depending on the selected framework.

It is important to keep existing tango tools compatible with any new data storage development and some effort might be required to build any missing component but this work could benefit all the community and also solve the SKAO monitoring challenges.

ACKNOWLEDGMENTS

We would like to thanks the Square Kilometre Array Observatory, the South African Radio Astronomy Observatory and the Tango Control collaboration.

REFERENCES

[1] The Square Kilometre Array Observatory (SKAO), https://skao.int

[2] SKA-Mid, https://www.skao.int/en/explore/telescopes/ska-mid

[3] SKA-Low, https://www.skao.int/en/explore/telescopes/ska-low

[4] T. Juerges and A. Dange, The SKAO Engineering Data Archive: From Basic Design to Prototype Deployments in Kubernetes, in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 1590–1593. doi:10.18429/JACoW-ICALEPCS2023-THSDSC05

[5] M. J. Slabber, "Overview of the Monitoring Data Archive used on MeerKAT", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1155–1157. doi:10.18429/JACoW-ICALEPCS2015-THHD3O06

[6] NATS documentation, https://docs.nats.io/.

[7] Ceph Filesystem, https://docs.ceph.com/en/reef/cephfs/.

[8] A. Götz *et al.*, "State of the Tango Controls Kernel Development in 2019", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1234–1239. doi:10.18429/JACoW-ICALEPCS2019-WEPHA058

[9] The Tango Controls facilities, https://www.tango-controls.org/about-us

[10] The Tango Controls Gitlab project, https://gitlab.com/tango-controls

[11] D. Lacoste *et al.*, "HDB++, a retrospective on 5+ years using TimescaleDB", presented at ICALEPCS'25, Chicago, IL, USA, paper TUMR004, this conference.

[12] HDB++ GitLab group, https://gitlab.com/tango-controls/hdbpp

[13] HDB++ Configuration Manager, https://gitlab.com/tango-controls/hdbpp/hdbpp-cm

[14] HDB++ Event Subscriber, https://gitlab.com/tango-controls/hdbpp/hdbpp-es

[15] HDB++ backend library for the TimescaleDb extension to Postgresql, https://gitlab.com/tango-controls/hdbpp/libhdbpp-timescale

[16] TimescaleDB, a PostgreSQL extension for time series, https://docs.tigerdata.com/use-timescale/latest/hypertables/

[17] A PostgreSQL to parquet to S3 script, https://gitlab.com/ska-telescope/ska-databases-metadata-scripts/-/tree/main/pg/pg2pq2s3

[18] An API to query PostgreSQL and parquet files on S3, https://gitlab.com/ska-telescope/ska-databases-metadata-scripts/-/tree/main/pg/pg2pq2s3