ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THSDSC05

THE SKAO ENGINEERING DATA ARCHIVE: FROM BASIC DESIGN TO PROTOTYPE DEPLOYMENTS IN KUBERNETES

Thomas Juerges, SKA Observatory, Jodrell Bank, United Kingdom
Aditya Dange, Tata Consultancy Services, Pune, India

Abstract

During its construction and production life cycles, the Square Kilometre Array Observatory (SKAO) will generate non-scientific, i.e. engineering, data. The sources of the engineering data are either hardware devices or software programs that generate this data. Thanks to the Tango Controls software framework, the engineering data can be automatically stored in a relational database, which SKAO refers to as the Engineering Data Archive (EDA). Making the data in the EDA accessible and available to engineers and users in the observatory is as important as storing the data itself. Possible use cases for the data are verification of systems under test, performance evaluation of systems under test, predictive maintenance and general performance monitoring over time. Therefore we tried to build on the knowledge that other research facilities in the Tango Controls collaboration already gained, when they designed, implemented, deployed and ran their engineering data archives. SKAO implemented a prototype for its EDA, that leverages several open-source software packages:

- with Tango Controls' HDB++
- the Timescaledb time series database
- and Kubernetes at its core.

In this overview we will answer the immediate question "But why do we not just do, what others are doing?" and explain the reasoning behind our choices in the design and in the implementation.

INTRODUCTION

The Square Kilometre Array Observatory (SKAO) [1] with its headquarters in Jodrell Bank, UK, is currently constructing two large-scale radio telescope arrays, SKA-Mid (under construction)[1] [2, 3] in South Africa and SKA-Low (under construction)[2] [2, 4] in Western Australia.

During construction, commissioning, and operation engineers, operators, and scientists will need to inspect a range of non-scientific data with a range starting as simple as temperature and humidity to highly specific ones like a SKA-Mid receptor's delay coefficient. All this data is required to be handled in a uniform manner. This is where the EDA [5] software comes into the picture.

To emphasize, even during the ongoing construction phase, the already existing software that is continuously tested and integrated in Kubernetes [6] clusters and the prototype hardware generate engineering data. This data has at this moment value for debugging the software, its performance, and finding mismatches between how the hardware behaves and the described behaviour in its interface control documents. When a simple line of log cannot tell the whole story, engineering data is able to complete the picture. Engineering data has the power to provide, even independent of any logs, a realistic representation of the current state of a system.

Naturally had SKAO's initial EDA design aged somewhat in the fast moving software technology age and was ready for a fresh look. Therefore SKAO invited partner facilities in the Tango Controls collaboration to explain and demonstrate their EDAs in order to modify the existing design for the SKAO EDA with the help of the already existing knowledge about newer technologies.

The demonstrations were very enlightening and it became almost immediately clear that the existing design for SKA's EDA would really benefit from a fresh look, replacing technologies that had been shown to not work for the use case at hand as had been proven at the partner facilities and also make use of technologies that SKAO had adopted in the meanwhile.

EARLY IDEAS, THE DESIGN AND THE ARCHITECTURE

The SKAO's EDA design was based on the Tango Controls [7] HDB++ archiver [8], consisting of the HDB++ Configuration Manager [9] and the HDB++ Event Subscriber [10], an EDA Controller and Cassandra [11] as the EDA database back end (as shown in Fig. 1).

This design focused on addressing the functional requirements and did not consider the deployment aspect. One reason was the expected shift in software deployment from bare metal deployments to cloud-based ones.

Evolution Over Time

The geographical distribution of the observatory's computers and clusters already added significant complexity to deployment of, running and maintaining the software needed to operate the entire observatory with the two telescopes. This informed SKAO's decision to containerize all software that does not have to run on bare metal and also deploy

[1] SKA-Mid will be a radio telescope array consisting of 192 fully steerable dishes: 13 315 m SKA dishes and the already operating 6413.5 m Meerkat dishes. The receivers are sensitive at frequencies between 350 MHz and 15.4 GHz. The geographical distribution of the dishes allows for maximum baselines of 150 km.

[2] SKA-Low will be an aperture array radio telescope that consists of 512 stations each with 256 log-periodic dipoles. The dipoles are sensitive at frequencies between 50 MHz and 350 MHz. The configuration of the stations and their geographical distribution allow for baselines between 0.7 km and 70 km at a nominal frequency of 140 MHz.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THSDSC05

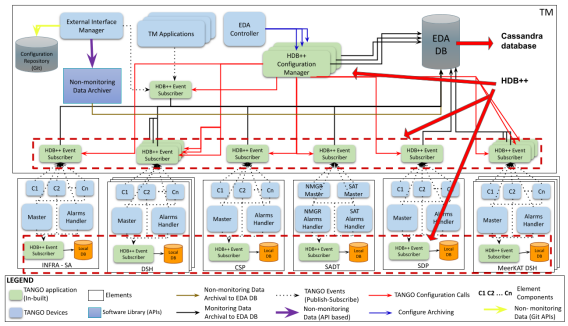

Figure 1: The initial EDA design in the context of the Tango Controls devices needed for SKA-Mid and SKA-Low.

the operational software in Kubernetes environments. The logical conclusion for the EDA team was therefore to also containerize the EDA software as well and deploy it in Kubernetes environments. This would immediately simplify software deployment and its maintenance, especially upgrading of software versions at run time without interrupting the operation of the observatory.

After consulting with other large-scale experiments (Light sources: ALBA [12], Elettra [13], MAX IV [14]; Scientific Telescope: ASTRON's LOFAR2.0 [15]) it was decided to replace Cassandra with Timescale [16]. It was also decided that a non-monitoring data archiver would be out of scope for the EDA.

In short, the changes that were made to the design are as follows:

- Non-monitoring data archiver: out of scope, potentially be replaced later with an online data storage solution (Parquet files [17]) that makes direct access from the database possible.
- EDA Controller: This application was replaced with the Configurator and the new application ArchWizard [18].
- The new data visualisation GUI ArchViewer [19] was added.
- Cassandra: The Cassandra NoDB was replaced by the Timescale timeseries database.
- Fully containrise the entire EDA software.
- Deployment of the EDA software with Helm [20] charts.

Individual Deployments

Figures 2 and 3 show the current deployment of the EDA for the development pipeline and SKA Dish-specific deployment respectively. There are a few differences between the functional blocks. The EDA Controller envisaged to manage the EDA operations, evolved into two separate components namely Archwizard and Configurator. The Non-monitoring data archiver was descoped. The database backend was also changed to Timescale as mentioned earlier. The main

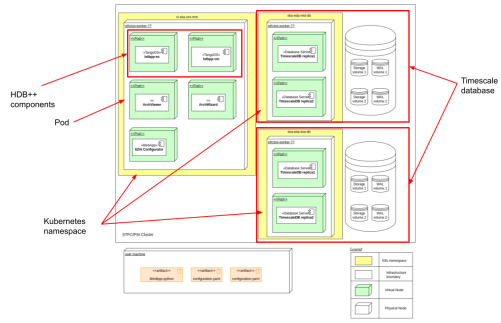

Figure 2: EDA deployment for development pipelines.

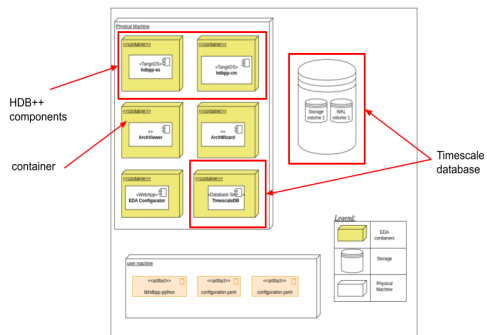

Figure 3: EDA deployment for Dish.

archiver component, the HDB++, remained the same of which the latest available version were to be used.

There is considerable work on the deployment aspect where deployment differs across various computing environments. The main factors behind the tailored deployments are: the availability of computing resources in different facilities, the heterogeneous nature of subsystems and physically distributed sites. The deployment aspect is further discussed in the next section.

FROM PLAN TO REALITY: DEPLOYMENT AND THE WORKING SYSTEM

The current trend in software deployment is containerization of the applications and software packages in order to be able to deploy them either in the cloud or on bare metal. Containerised software makes automated maintenance of a large-scale software system easier, at the same time enhancing the security of the deployed software. This in part addresses the SKAO policy that software shall be conscious of the large attack surface that the SKAO network and its computing resources offer. Containerisation also allows usage of heterogeneous development platforms the SKA developers work on/are proficient in because the containers are self-sufficient enough that they are almost OS and even CPU architecture independent.

Software that is not available as an image is "containerised" by SKAO developers. They create the files and scripts necessary to build an OCI [21] image of the software from a stable release in SKAO's CI/CD pipeline. Pre-configured jobs in the CI/CD pipeline ensure the software quality is

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THSDSC05

maintained by executing a set of test cases on the build. After review, the resulting image is released as a standardized product on SKAO's artifact repository from where it can be downloaded for deployment.

The actual deployment of the EDA is done with the help of Helm charts. In such a Helm chart the containerised Archiver applications (HDB++ Configuration Manager, HDB++ Event Subscriber, Configurator, ArchWizard, ArchViewer) are bundled together. To provide more flexibility in the deployment, it is done in two steps, using two individual charts: one for the Timescale database server and one for the Archiver applications. The split of the two serves also the purpose of protecting the Timescale databases when the Archiver applications receive an update or a bug fix and need to be redeployed.

Considering the different resource needs for SKA environments in different locations, there are multiple deployment configurations defined to save the time of the engineers to fine-tune the specific deployment configurations. It is also possible to enhance the configuration by providing individual parameters during the deployment of the Helm charts as can be seen in the example in Fig. 4.

```
# Clone the repository with the Timescale server deployment
# Specify a tag in the --branch parameter for reproducibility
git clone --branch=2.8.2 --depth=1 https://gitlab.com/ska-telescope/
    ska-tango-archiver.git
cd ska-telescope/ska-tango-archiver
# Deploy Timescale with an already existing and known persistent volume
    (default values are assumed)
# Note: Timescale deployment can also be configured with parameters on the
    command line
make timescaledeploy
# Deploy the Archiver applications, assuming default configuration values.
make k8s-install-chart
# The deployment step will log a URL at which a user can access the
    Configurator.
# The documentation is available at
# https://developer.skao.int/projects/ska-tango-archiver/en/latest/introduction/
    introduction.html
```

Figure 4: Deployment of Timescale and the Archiver with just a handful of commands.

It is possible to directly provide values that modify the default configuration of a deployment as shown in Fig. 5. With this, the Archiver applications are deployed in the Kubernetes namespace 'ska-tango-archiver'.

```
make k8s-install-chart TELESCOPE=<SKA-low/mid> \
    ARCHIVER_DBNAME=<dbname> \
    ARCHIVER_TIMESCALE_HOST_NAME=<hostname> \
    ARCHIVER_TIMESCALE_PORT=<port> \
    ARCHIVER_TIMESCALE_DB_USER=<dbuser> \
    ARCHIVER_TIMESCALE_DB_PWD=<dbpassword> \
    TELESCOPE_ENVIRONMENT=<environment name> \
    ARCHWIZARD_CONFIG=<full FQDN of configuration Manager>
```

Figure 5: Individual configuration options can be provided with the installation command.

A deployment can be removed, i.e. uninstalled, with a simple command:

```
make k8s-uninstall-chart
```

As can be seen in the examples above, it is possible to deploy multiple instances of the Timescale server and the Archiver applications in the same location by just specifying different namespaces.

Another graphical user interface is the Configurator, shown in Fig. 6. The tool allows to modify at run time what is archived. A YAML [22] file with the new or changed configuration can be uploaded and the changes are immediately applied. The current configuration can also be downloaded as a YAML file.

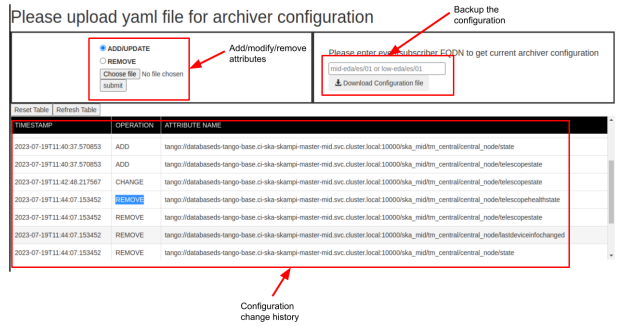

Figure 6: Screenshot of the Configurator in a running EDA deployment.

Data Extraction

Data extraction from Timescale databases can be accomplished in many ways. Due to the specific HDB++ schema that is used in Tango Controls to store attribute values in an HDB++ backend, a manual extraction in SQL would be inconvenient. Therefore is ArchViewer, a GUI for web-based data visualization purposes, provided with every deployment (Fig. 7).

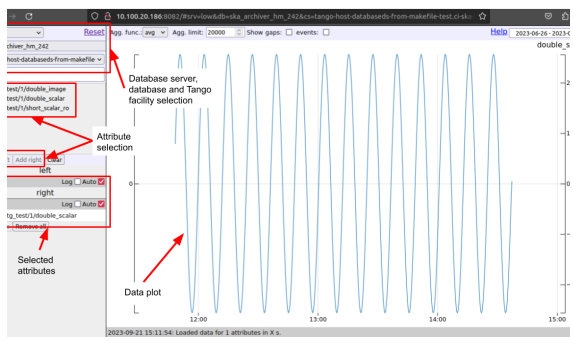

Figure 7: ArchViewer: The web-based data visualisation GUI for HDB++ Timescale databases.

```
>>> import pyhdbpp
>>> rd = pyhdbpp.reader(
apiclass = 'pyhdbpp.timescaledb.timescaledb.TimescaleDbReader',
config = 'admin:admin@10.200.10.109:5432/mid_archiver_db'
)

# rd.get_attribute_values('<attribute_name>', 'start_time', 'end_time')

>>> values = rd.get_attribute_values('tango://databaseds-tango-base.
    ci-ska-skampi-hm-213-mid.svc.cluster.local:10000/ska_mid/tm_subarray_node/1/
    obsstate', '2022-8-24', '2023-07-10')

>>> len(values)
79

>>> values[0]
(datetime.datetime(2023, 5, 9, 7, 24, 49, 9386, tzinfo=datetime.timezone.utc),
    2, None, 0, None)
```

Figure 8: Example data extraction from a Timescale HDB++ backend with PyHDBPP.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THSDSC05

For programmatic data extraction and data evaluation, the pyhdbpp Python module is recommended to the users [23]. The data access with pyhdbpp is user-friendly and straight forward as can be seen in Fig. 8.

CONCLUSION AND OUTLOOK

It has been very valuable to listen to the other facilities in the Tango Controls collaboration and to learn how they deploy and run their archives for engineering data and understand the designs they chose, what technologies they used and what their deployment and data maintenance strategies are. This informed SKAO's decisions to move away from Cassandra as a database back end for the EDA and to Timescale. Unlike the majority of facilities in the Tango Controls collaboration, SKAO runs a major part of its monitor and control software in Kubernetes clusters. The decision to also deploy the EDA in Kubernetes environments instead of bare metal integrates the EDA seamlessly into SKAO's software landscape. The EDA becomes easier to maintain because it is not a one off software that needs special attention.

First deployments have shown that the documentation has already reached a mature and satisfactory level, but still needs improvements where it is not explicit enough and thus has the potential to confuse users. These deployments will very likely expose performance bottlenecks, that the flexible design will be able to address at the database, Event Subscriber and Configuration Manager levels.

Timescale's native on-the-fly data compression is already being worked on as well as database server and data health supervision. Data migration into long-term storage capability is planned for, but it remains to be seen first, what data rates will be reached and how quickly the available data storage will be exhausted.

REFERENCES

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

[2] SKA Telescope Specifications, https://www.skao.int/en/science-users/118/ska-telescope-specifications##__otpm4

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

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

[5] Documentation SKAO Engineering Data Archive, https://developer.skao.int/projects/ska-tango-archiver/en/latest/

[6] Kubernetes, https://kubernetes.io

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

[8] L. Pivetta, R. Bourtembourg, J.L. Pons, C. Scafuri, G. Scalamera, G. Strangolino, *et al.*, "HDB++: A New Archiving System for TANGO", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WED3O04, pp. 652–655. doi:10.18429/JACoW-ICALEPCS2015-WED3O04

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

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

[11] Apache Cassandra, https://cassandra.apache.org/

[12] ALBA Synchrotron, https://www.cells.es/en/

[13] Elettra Sincrotrone Trieste, https://www.elettra.eu

[14] MAX IV Laboratory, https://www.maxiv.lu.se

[15] T. Juerges, J.J.D. Mol, and T. Snijder, "LOFAR2.0: Station Control Upgrade", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 31–36. doi:10.18429/JACoW-ICALEPCS2021-MOAR03

[16] Timescale, https://www.timescale.com

[17] Apache Parquet, https://parquet.apache.org

[18] ArchWizard: A webapp for viewing HDB++ archived Tango attribute data in a Timescale database, https://gitlab.com/tango-controls/hdbpp/archwizard

[19] ArchViewer: A simple web interface to the HDB++ archiving system, https://gitlab.com/tango-controls/hdbpp/archviewer

[20] Helm, the package manager for Kubernetes, https://helm.sh

[21] Open Container Initiative, https://opencontainers.org/

[22] YAML: Yet Another Markup Language, https://yaml.org

[23] PyHDBPP: The Python HdbReader, https://gitlab.com/tango-controls/hdbpp/libhdbpp-python