OVERVIEW OF THE MONITORING DATA ARCHIVE USED ON MeerKAT

Design of KATStore - THHD3O06

Martin Slabber
22 October 2015
SKA SA, Cape Town, South Africa
Table of Contents

1. Overview
2. Goals
3. Architecture
4. Conclusion
Overview
Ethernet is used as the field bus.

THN contains ’nearly’ all the equipment needed to operate the telescope.
CAM system communicates with THN, external systems and users.

Users: Engineers, operators and scientists.

KAT-7 and MeerKAT each run an independent CAM system.

On completion MeerKAT CAM will have nearly 80,000 sensors. Physical, system metrics and aggregated.

KATCP is used within CAM and between CAM and THN.
What is in a sample?

- Sensor’s Name - Name of the sensor.
- Sample Timestamp - Process time.
- Value Timestamp - Acquisition time.
- Status - Status of sensor when processed.
- Value - The value of the sensor.

A sensor is a KATCP primitive.

pi,1445398685.612134,1445398675.371092,nominal,3.141593
Sensors have meta data

Typical Sensor’s meta data:

- Value type - E.g. float, integer, string, booleans, etc.
- Description - A text description of the sensor.
- Unit of measure - E.g. Celsius, RPM.
- Parameters - Specifics of the sensor and value type. E.g. min and max allowed numbers.

DATA = Sensor samples + Sensor meta data

KATStore places no limitation on sensor attributes.
Goals
Some key goals

- Gather samples using sampling strategies defined in configuration.
- Cope with sudden high sample volumes. Events.
- Store all samples and sensor meta data permanently.
- Archive files to NAS. CAM nodes are transient.
- Query samples. Queries should be able to span all stored samples.

The astronomy data (correlated receiver voltages) are stored by a different system.
Architecture
- **Memory buffer** on each node. (Redis)
- Monitor process writes to **memory buffer**.
- **Central database** on storage node. (PostgreSQL)
- Process on storage node pulls data from **memory buffer** and writes to **central database**.
- Data is archived from **central database** to NAS.
- Data is stored in HDF5 files on NAS.
Monitor

- A Monitor process on every node. Including the storage node.
- Using KATCP, connect to all components on the same node.
- Subscribe to sensors using the configured sampling strategy.
- Write samples to Redis.
Memory Buffer - Redis

- Very fast.
- Stores the sensor data until removed by Pulld.
- Allows the control activity to be decoupled from storage.
- Copes well with monitor writing at undetermined rates e.g. events.
Pull Daemon (Pulld)

- Moves data from Redis to PostgreSQL in batches.
- Starts a child process for every node in the system. Many *pull_handler* are managed by one *pull_manager*.
- Periodically *pulld_manager* will move *old* samples to archive.
- Messaging facility between *pulld_manager* and *pull_handler* and via DB notify.
- *pull_manager* is a KATCP server and is monitored.
- Stores sensor samples until archived.
- Stores sensor meta data.
- Samples are stored in multiple tables. Horizontal partition a.k.a sharding.
- Archived data are presented in the database as a foreign table via a FDW.
- Allow for queries over stored samples and archived samples.
HDF Server

- Handles all read and write operations to HDF5 files.
- HDF5 files are stored on the NAS.
- For queries, HDF Server is exposed as a foreign table in the database.
- On the wire: Concise Binary Object Representation (CBOR) RFC 7049 and Blosc compression.
Query Interface

Two interfaces:

- KATCP interface used by CAM components.
- RESTful interface used by CAM GUI and external systems.
- Mechanisms to send the samples that was in flight, after the query.
- Based on query input construct SQL.

Query the database for current and historic data. For historic data, query the foreign table.
Conclusion
Conclusion

A complete system was developed to efficiently move sensor samples from the nodes where they were collected on to the central storage node from where the samples can be archived and queried.

The system was designed to be comprised out of several independent elements. Each element is concerned with a specific function. Thus, while developing each element, it was possible to focus exactly on solving the problem at hand. The elements make testing, fault finding and performance tuning easier.
MeerKAT will be a 64-receptor aperture-synthesis interferometric radio telescope array. MeerKAT will be the most sensitive L-band radio telescope in the world. It should be completed in 2017. MeerKAT is located in the sparsely populated semi-arid Karoo region of the Northern Cape Province, South Africa.

The MeerKAT Control and Monitoring (CAM) System Architecture includes a set of processes on each of the 64 MeerKAT digital front end (DFE) receivers. The processes are divided into two rough categories: (i) devices, i.e., the 64 DFEs, which are responsible for operating the receivers, and (ii) a CAM server, which is responsible for managing and processing the sensor data. The devices and the CAM server are connected via a dedicated network, which is designed to handle the large volume of data generated by the 64 DFEs.

The devices are responsible for monitoring and controlling the 64 DFEs. They collect sensor data from the DFEs, which includes information about the health and performance of the DFEs. The sensor data is then sent to the CAM server via the network. The CAM server processes the sensor data and sends it to a KATCP server, which is responsible for managing the sensor data. The KATCP server then sends the sensor data to a KATCP client, which is responsible for displaying the sensor data to the operator.

The sensor data is stored in a central storage node, which is responsible for archiving and querying the sensor data. The sensor data is also sent to a KATCP server, which is responsible for managing the sensor data. The KATCP server then sends the sensor data to a KATCP client, which is responsible for displaying the sensor data to the operator.

The CAM server is responsible for managing the sensor data. It processes the sensor data and sends it to a KATCP server, which is responsible for managing the sensor data. The KATCP server then sends the sensor data to a KATCP client, which is responsible for displaying the sensor data to the operator.

The sensor data is also sent to a KATCP server, which is responsible for managing the sensor data. The KATCP server then sends the sensor data to a KATCP client, which is responsible for displaying the sensor data to the operator.