ILLUSTRATE THE FLOW OF MONITORING DATA THROUGH THE MeerKAT TELESCOPE CONTROL SOFTWARE

M. Slabber*, SKA SA, Cape Town, South Africa
M.T. Ockards†, SKA SA, Cape Town, South Africa

Abstract

The MeerKAT telescope [1], under construction in South Africa, is comprised of a large set of elements. The elements expose various sensors to the Control and Monitoring (CAM) system, and the sampling strategy set by CAM per sensor varies from several samples a second to infrequent updates. This creates a substantial volume of sensor data that needs to be stored and made available for analysis. We depict the flow of sensor data through the CAM system, showing the various memory buffers, temporary disk storage and mechanisms to permanently store the data in HDF5 format on the network attached storage (NAS).

ELEMENTS THAT MAKE UP CAM

Sensor

Sensors are either physical sensors on devices, aggregated/calculated or internal metrics from software components. Some examples of physical sensors are tilt, temperature, wind speed and motor rpm. It is estimated that once MeerKAT is completed there will be close to 80 000 sensors in the system.

Device

Devices are the hardware that constitute the telescope. Devices are engineered to have Ethernet interfaces and can communicate via Karoo Array Telescope Communication Protocol (KATCP). Where devices do not have KATCP interfaces, translators are created [2]. The devices expose sensors and requests on the KATCP interface [3].

Component

CAM consists of many distinct components. Components are the standalone building blocks of the CAM system. Interaction with components are done with the KATCP protocol. The components expose sensors and requests over KATCP.

Node

Nodes are virtual containers on the host hypervisor [4]. The MeerKAT nodes run Ubuntu 14.04 LTS Linux operating system. When MeerKAT is fully deployed, it is expected to have 13 nodes.

Proxy

Proxy components make sensors and requests on devices available to the CAM system. Sensors and requests from the devices can be discarded or even rolled up into new sensors

* martin@ska.ac.za
† tockards@ska.ac.za

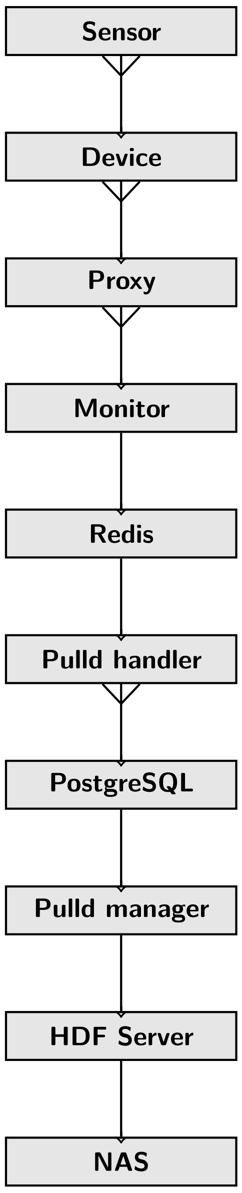

Figure 1: The progression of storing a sample, from sensor to archive.

and requests on the proxy component. The proxy component also exposes sensors and requests of its own.

Controller

Controller components connect to other subsystems. As with a proxy component, sensors and requests from the subsystem can be exposed or rolled up into new sensors and requests. Controllers always use KATCP to communicate to

ISBN 978-3-95450-148-9

the subsystem. The subsystem is responsible for presenting a central interface to which CAM can connect in order to gather sensor samples and use requests to initiate action. Examples of subsystems are the Correlator Beam Former and Science Data Processing.

Monitor

On each node in the CAM system a monitor component gathers sensor samples at a configured sampling rate from the other components on the same node. The gathered samples are written to a memory buffer on the node.

Redis

Redis is used as a memory buffer on each of the nodes in the CAM system. Samples are written to the memory buffer by the monitoring component.

Pulld

Pulld component on the storage node that moves the samples from the memory buffers to the central database. A sub-process of Pulld, Pulld handler, is started for each of the nodes in the system. Pulld will archive samples from the storage node and send it to the HDF Server to be archived. The archiving is done in the Pulld manager on a scheduled basis.

PostgreSQL

PostgreSQL is used as the central database for sensor samples and sensor metadata [5]. Samples are stored in the central database until they are archived to files on the network attached storage (NAS). The archived data is presented in the database as a foreign table; a foreign data wrapper (FDW) was developed to achieve this. Queries to the archive table are translated by the FDW and sent to the HDF Server; the response from the HDF Server is translated for presentation in the database. The FDW was developed to provide a read only interface to the HDF Server.

HDF Server

This component is responsible for performing all the read and write operations on the archive files. The archive files are HDF5 formatted files stored on the NAS. The files are stored in a hierarchical directory structure. An archive file is created for each component of the system per day. e.g. For *subarray1* on 7 March 2015 the file `2015/03/07/2015-03-07_subarray1.h5` will be created.

NAS

The NAS is a standalone storage unit attached to the network. An export on the NAS is mounted on the storage node using network file system (NFS). The NAS is accessible over a 10Gbps Ethernet network.

Query Interface

Two query interfaces were developed to provide API's for accessing historical sensor data. The first uses KATCP as the access protocol and runs on the storage node, this interface is used by other CAM components. The second interface was developed as part of the portal system and provides an HTTP interface for the Graphical User Interface (GUI) [6] and external systems. Both query interfaces use the PostgreSQL database for all queries.

SENSOR SAMPLE

It is important to understand what a sample is in order to comprehend how the samples are transported, stored and made available for queries. Each sensor reading processed by the CAM system is called a sample. A sample always has a sensor name, sample timestamp, value timestamp, status and a value.

SENSOR_NAME A normalised form of the sensor's name. Non-alphanumeric characters are replaced by the '_' character; the case of alphabetic characters is maintained. The original KATCP name of the sensor is maintained as an attribute of the sensor with the sensor meta data.

SAMPLE_TS The timestamp of when the sample was first processed by the CAM system. It is a UNIX timestamp, the time in seconds since the epoch of 1 January 1970 00:00 UTC. SAMPLE_TS is always unique for a sensor and presented as a floating point number, the number of decimal places (fraction of a second) used is dependent on the host operating system.

VALUE_TS A UNIX timestamp when the acquisition was performed. VALUE_TS and SAMPLE_TS are the same or within milliseconds of one another if the acquisition was performed in sync with the CAM sampling strategy. When a sensor is over-sampled it is possible to have the same consecutive VALUE_TS, STATUS and VALUE.

STATUS An indication of the state of the sensor when the sample was taken. Can have a value of UNKNOWN, WARN, NOMINAL, FAILURE, INACTIVE, ERROR, UNREACHABLE as defined in the KATCP [3] specifications.

VALUE The value of the sensor at the time of acquisition (VALUE_TS). The data type of VALUE is typically one of float, integer, string, or boolean; additional types are allowed by the KATCP specifications but they are less commonly used. It was decided that the storage system will treat all values as strings until the archiving is performed. In the archive, data is stored as float, integer, boolean or string; all other types are stored as strings in the archived files.

SAMPLING STRATEGIES

CAM inherits the sampling strategies of the KATCP protocol. A client connected to the KATCP interface on a device or a component (the server) can set a sampling strategy for the sensors on the server; sensor samples will be sent by the server at the requested sampling rate. Several sampling strategies are available in KATCP. KATCP defines the strategies `none`, `auto`, `period`, `event`, `differential`, `event-rate` and `differential-rate`. Examples are `<period P>` the value is reported every `P` seconds, `<event>` the value is reported when it changes and `<event-rate SP LP>` the value is reported when it changes or every `LP` seconds. For the full details on sampling strategies see the KATCP specifications [3].

The sampling strategy used by the monitor component can be set per sensor in the central configuration. Sensors that do not have a configured sampling strategy are sampled every 10 seconds; for sensors of boolean and string types `event-rate 0 600` is used.

PROGRESSION THROUGH THE STORAGE SYSTEM

The following key words are used in Figures 2, 3 and 4.

- **Monitor** component has an instance running on every node.
- **Redis** memory database is installed on every node.
- **Pulld handler** has an instance on the storage node for every node in the system.
- **Pulld manager** has one instance on the storage node.
- **DB** is the central database on the storage node.
- **HDF Server** is running only on the storage node.
- **NAS** is the file storage system made available over the network.
- **Query** interface has instances on the storage node and the portal node.

Monitoring Component to Central Database

The monitor process connects to all the CAM services on the node on which it is running . Based on configuration, the monitor process subscribes to sensors and receives updates based on the requested sampling strategy. In Figure 2 (it is shown) for every new update the monitor gets from a sensor, the sample is written to the memory database (1). The memory database is a buffer, it allows for very fast writes and prevents blocking the monitor process. Because of the memory buffer, Pulld does not need any knowledge of the write rate of the monitor process. Pulld queries (2) the memory database and writes the received samples (3) to the database (4) on the storage node. When the write to the database on the storage node is successful (5) the samples are deleted (6) from the memory database on the relevant node.

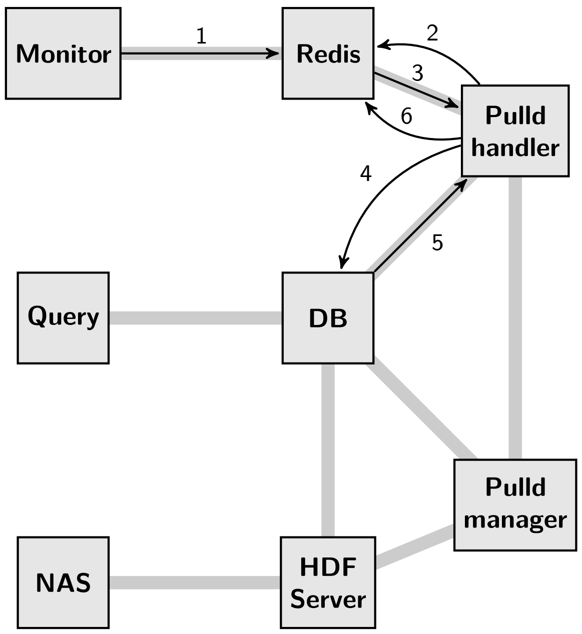

Figure 2: Moving samples from monitoring component to the central database.

Central Database to the Archive

Based on configuration, Pulld will start an archive task at a scheduled time (Fig. 3). The archive task will determine what data should be archived (1); based on a configurable sample age parameter. Pulld will retrieve data (2) from the database and write this data to the HDF Server (3). When the HDF Server acknowledges (5) that data has been successfully written to the NAS (4), the data will be deleted from the database (6).

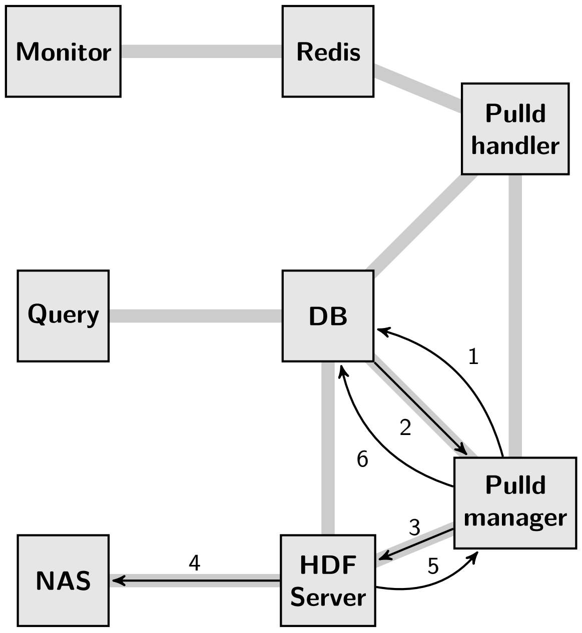

Figure 3: Archiving data from storage database to NAS.

ISBN 978-3-95450-148-9

Query the Samples

Requests from other components for historical sensor samples are done against the query interface (Fig. 4). The query interface is connected to the database and will create the appropriate SQL query and run the SQL against the database (1). If the query requires archived data, the database has a foreign table that contains no data, but provides an interface to the HDF server. It uses the FDW to achieve this. The FDW will rewrite the query into a syntax that can be interpreted by the HDF Server (2). The HDF Server will read the appropriate files from the NAS (3) and return the samples to the database (4). The database will add the returned samples to the response and return (5) to the query interface.

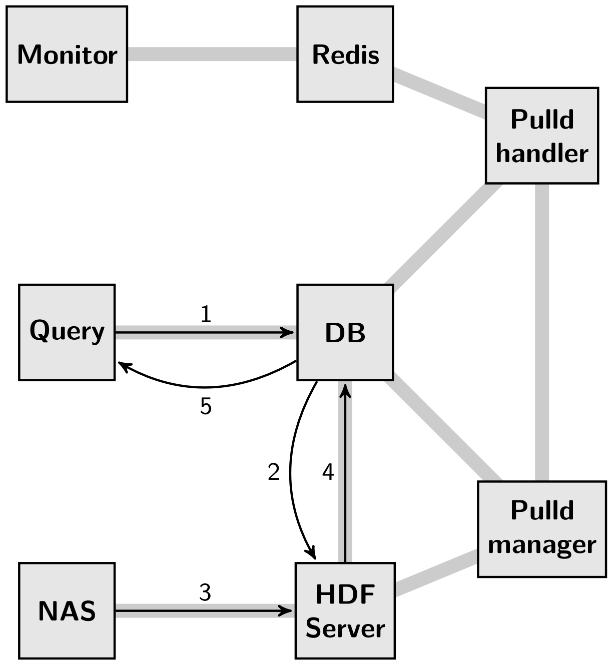

Figure 4: Query data across storage database and archive on NAS.

REFERENCES

[1] R.S. Booth, W.J.G. de Blok, J.L. Jonas, and B. Fanaroff, "MeerKAT Key Project Science, Specifications, and Proposals," *ArXiv e-prints*, pp. 1–16, 2009. [Online]. Available: http://arxiv.org/abs/0910.2935

[2] L. van den Heever, "MeerKAT Control And Monitoring - Design Concepts and Status," in *Proceedings of ICALEPC 2013*, paper MOCOAAB06, 2013.

[3] S. Cross, R. Crida, T. Bennett, M. Welz, and T. Kusel, "Guidelines for Communication with Devices," 2012. [Online]. Available: http://pythonhosted.org/katcp/_downloads/NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[4] N. Marais, "Virtualization and deployment management for the KAT-7 / MeerKAT control and monitoring system," in *Proceedings of ICALEPC 2013*, paper THCOBA06, 2013.

[5] M. Slabber, "Overview of the monitoring data archive used on the MeerKAT telescope," presented at ICALEPCS'15, Melbourne, Australia, paper THHD3O06, *these proceedings*, 2015.

[6] M. Alberts and F. Joubert, "The MeerKAT graphical user interface technology stack," presented at ICALEPCS'15, Melbourne, Australia, paper THHC3O01, *these proceedings*, 2015.