THE CMS DETECTOR CONTROL SYSTEMS ARCHIVING

W. Karimeh[1], CERN, Geneva, Switzerland
C. Vazquez, CERN, Geneva, Switzerland
F. Glege, CERN, Geneva, Switzerland
M. Chamoun, Université Saint-Joseph de Beyrouth, Beirut, Lebanon
[1]also at Université Saint-Joseph de Beyrouth, Beirut, Lebanon

Abstract

The CMS experiment relies on its Detector Control System (DCS) to monitor and control over 10 million channels, ensuring a safe and operable detector that is ready to take physics data. The data is archived in the CMS Oracle conditions database, which is accessed by operators, trigger, data acquisition, and offline data reconstruction systems. In the upcoming extended year-end technical stop of 2023/2024, the CMS DCS software will be upgraded to the latest WinCC-OA release, which will utilise the SQLite database and the Next Generation Archiver (NGA), replacing the current Raima database and RDB manager. Taking advantage of this opportunity, CMS has developed its own version of the NGA backend to improve its DCS database interface. This paper presents the CMS DCS NGA backend design and mechanism to improve the efficiency of the read-and-write data flow. This is achieved by simplifying the current Oracle conditions schema and introducing a new caching mechanism. The proposed backend will enable faster data access and retrieval, ultimately improving the overall performance of the CMS DCS.

INTRODUCTION

Aimed at probing the deepest questions of fundamental physics, the Compact Muon Solenoid (CMS) is one of the flagship experiments at CERN's Large Hadron Collider (LHC). The Detector Control System (DCS) is integral to its functionality, enabling safe operations and efficient control of the LHC experiments. From the early stages of design, all LHC experiments have adopted WinCC Open Architecture (OA) [1] from ETM as their default Supervisory Control and Data Acquisition (SCADA) software.

At CMS, a myriad of sensors and control units require unwavering precision, addressed by the distributed and redundant DCS projects [2]. With its real-time data assimilation, the DCS projects play an indispensable role in ensuring operational consistency while also offering invaluable archival records. The vast, real-time data collected by the DCS not only helps in ongoing operations but also serves as an essential historical record, enabling researchers to understand anomalies, enhance the system, and plan future experiments.

Data collected by WinCC OA is archived in an Oracle database: The Conditions Database. The paper describes the evolution of the CMS DCS conditions database over the past 15 years of operations and unveils the latest development: the CMS Next Generation Archiver backend.

CMS CONDITIONS DATABASE

WinCC OA uses an RDB manager that serves as a bridge between its internal database, RAIMA, and external databases, notably the Oracle DB used at CERN. The conditions database schema, provided by ETM, categorizes data into metadata tables —representing WinCC OA projects internal datapoints— and real data tables, which hold events and alerts records.

At CMS, the vast volume of data in the event and alert tables presented a challenge, making data retrieval particularly time-consuming from the expansive tables that housed these records.

To address these challenges, the CMS conditions database schema was developed on top of the official one, incorporating PL/SQL scripts. Over the years, it has seen significant enhancements, introducing several key features:

DPT Tables

In WinCC OA, data structuring revolves around Data Points (DP), each being categorized under a specific Data Point Type (DPT). Each DP can encompass one or more Data Point Elements (DPEs), where every DPE represents a unique value or state.

To optimize the distribution and manage the extensive load of the events table, the CMS schema employs PL/SQL scripts. These scripts facilitate the creation of new DPT tables, where columns are designated for DPE names, and they systematically channel DPE values into their respective tables.

Last Value Tables

Due to the large size of DPT tables and the need for certain applications to promptly access the most recent values to understand the current state of the detector, a 'last value' (LV) mechanism has been incorporated into the CMS schema using PL/SQL functions.

This approach led to the creation of an LV table corresponding to each DPT table. Each LV table is consistently updated with the latest values for every datapoint element. When a new value is received, it's compared to its predecessor in the LV table. If the two values match, the new one isn't archived in the DPT table. Furthermore, a dead band configuration is introduced for every DPE within the LV table, ensuring the historical DPT tables only store significant data variations and filter out minor deviations.

Static Datapoint ID

While the standard WinCC OA DB schema identifies a datapoint through its internal project ID, the CMS version adopts a schema that employs a static identifier for each datapoint. This strategy mitigates potential remapping issues that might arise during routine activities like project and/or datapoint recreation.

As a solution, a dedicated table, DP_Name2id, has been introduced to catalogue unique IDs—generated using a database trigger and sequence—for every datapoint. This ensures the conditions database remains consistent, even in the face of potential ID changes in the WinCC OA projects.

CMS NGA BACKEND

The DCS projects at CERN will be upgraded to the latest WinCC OA version 3.19 which introduces the new SQLite internal database (replacing the RAIMA database) and the NGA replacing the RDB manager. In a collaborative effort between CERN and ETM, the NextGen Archiver (NGA) [3] project was initiated to craft a modular and scalable archiving solution tailored for CERN's SCADA systems. Building upon the core achievements of the NGA, the CMS embarked on the development of a specialized backend software, tailored for its need and special schema.

With the planned upgrade of CMS DCS projects to WinCC OA version 3.19 during the year-end technical shutdown of 2023-2024, two pivotal changes will take place: the phasing out of the RAIMA database in favour of the SQLite internal database, and the substitution of the RDB manager with the NGA. The latter has been a collaborative endeavour between CERN and ETM, aimed at delivering a modular and scalable archiving solution tailored for CERN's SCADA systems.

Building on the foundational achievements of the NGA, CMS has initiated the creation of specialized backend software. This new development is designed to cater to the requirements of CMS.

MOTIVATION

The existing structure of the CMS schema organizes data into unique DPT and LV tables, as highlighted earlier. While this system has effectively served its purpose, the introduction of NGA spotlighted areas for improvements. With the computational demands of PL/SQL scripts exerting pressure on the shared database side, there was a clear incentive to reallocate the PL/SQL logic from the centralized database configuration to the more distributed NGA infrastructure. However, it was imperative to make this transition without altering the existing schema, ensuring backward compatibility.

The objective behind this move was two-fold: to introduce a more cohesive method for data management and to alleviate considerable operational burdens from the central database, thus paving the way for an improved and sturdy archiving strategy within the CMS DCS architecture.

DESIGN

In the design phase of the CMS NGA backend, several recurrent events became evident:

- **Database Structure Verification** The repeated need to ascertain the existence or absence of specific tables or columns in the database.
- **Datapoint Validation** Recurrent requirements to validate the presence of the CMS static DP IDs and retrieve the ID for a given datapoint.
- **Data Storage Path Determination**: Identify the table and column intended for the storage of data for a given DPE.
- **Access to the LV Data** A standing necessity to swiftly retrieve the latest value of a designated DPE.

To face the above recurrent challenges and to achieve superior performance, the CMS NGA backend incorporated a new feature: in memory caching mechanism for the metadata and LV data structures. The QT framework's tools:

- **QMap** Red-Black tree-based associative container, offering an ordered data storage making it suitable for frequent key-based lookups.
- **QHash**: A hash table-based container, ensuring average constant-time lookups and insertions, prioritizing retrieval speed.
- **QExplicitlySharedDataPointer** Providing efficient and safe sharing of data between objects. This not only prevented unnecessary overhead from deep copying but also ensures that an object's data is duplicated only when modifications are made, optimizing both memory and computational resources.

This has led to the introduction of the following caching data structures:

Metadata Caching

- **DPE Metadata QMap (elementIdAndDpeMetadataMap)** A mapping of the project's DPE IDs to their respective metadata.
- **Table and Column List QHash (tableAndColumnListHash)** A mapping which correlates tables with their corresponding list of columns.
- **DP Internal ID QMap (dpInternalIdMap)** A QMap which correlates DP static database ID with their corresponding DP name.

LV Caching

- **DPE Data QMap (elementIdAndDpeDataMap)** An explicitly shared data pointer map that associates the WinCC OA project DPE IDs with their LV data.
- **DPE Dynamic Data Map (elementIdAndDpeDynDataMap)** This map correlates the WinCC OA project DPE IDs of dynamic arrays to their LV data.
- **Global Table and DPE Data QHash (GlobalTableAndDpeDataHash)** This mapping structure serves as a quick reference for all the LV entries defined in the database LV tables.

The introduced caching mechanism primarily targets performance enhancement by facilitating the rapid retrieval of DPE metadata and the most recent values. This ensures optimal system responsiveness. One of its other vital advantages is the notable reduction of strain on the primary database and network. By caching data that's accessed frequently, it not only mitigates the database's operational pressures but also optimizes network efficiency. A defining feature of this mechanism is its scalability. It's adeptly designed to handle growing data volumes, ensuring that performance remains consistent regardless of the increase.

DATA FLOW

Within the CMS NGA backend, data management and processing are orchestrated via two distinct flows: Metadata Handling and Value Archiving. These defined handlers ensure a structured and efficient approach to consistently synchronise the database and the local cache structures.

Metadata Handling

The CMS NGA backend has outlined a structured procedure to manage incoming DPE metadata updates from the NGA frontend. This can be broken down into a few distinct steps:

1. **Identifier Transformation**

Upon receipt, the DPE metadata undergoes a transformation to fit the constraints of the Oracle database. Notably, elements like DPT and LV tables names have a character limit, which is capped at 30 characters.

2. **Integrity Assessment**

The system first checks the elementIdAndDpeMetadataMap cache to determine if the DPE is already known.

For New DPEs**:**

- The dpInternalIdMap cache is consulted to verify if an internal identifier already exists for the DPE. If not, the backend creates a new ID, saves it in the DP_NAME2ID table in the database, and subsequently updates the dpInternalIdMap.
- The system verifies the presence of the associated DPE table and column within the tableAndColumnListHash cache. If these aren't present, they're established in the database, and the cache is updated accordingly.

For Known DPEs**:**

- The metadata attributes - specifically, the table name, column name, and data type - are reviewed. If there are mismatches or updates needed, the system updates both the database and cache accordingly.

3. **LV Synchronization**

As a final step, the last-value (LV) data associated with the DPE is synchronized using the GlobalTableAndDpeDataHash global cache. This step ensures that the system always maintains a consistent representation of the data across its various components.

Value Archiving

Upon receipt of data batches from the NGA frontend, the CMS NGA backend rigorously processes each DPE value. It does so by first establishing intermediary structures, which subsequently act as binding values for the SQL queries. This ensures efficient updating of the LV and DPT tables in the database. The series of operations involved include:

1. **Metadata and LV retrieval**

The first step of the DPE value processing is the retrieval of its metadata from the cache along with its LV data.

2. **Temporary data structure building**

- The incoming DPE value is compared with the LV record taking the dead band into account. According to this comparison, the temporary structures are updated.
- Special cases, such as when the DPE value returns NaN (Not a Number), are handled with caution to ensure they are correctly stored in the Oracle database without causing discrepancies.

3. **Database and cache update**

- Once all the DPE data values have been processed, the backend initiates SQL batch scripts to update the DPT and LV tables using the temporary data structures. This is followed by an update to the cache structures.

By adhering to this organized approach, the CMS NGA backend ensures the accurate and efficient handling of DPE values. This contributes to a more reliable and robust data management system, capable of handling large volumes of incoming data without compromising on data integrity.

FUTURE DEVELOPEMNTS

Current developments are undergoing extensive testing and performance evaluations. Preliminary results indicate significant improvements, especially in distributed systems with a high data archiving load. These enhancements, once fully validated, can further optimize the CMS NGA backend system, paving the way for more robust and scalable implementations in future iterations.

CONCLUSION

This paper detailed the design and implementation of an advanced in-memory caching mechanism for the CMS NGA backend, highlighting its efficacy in metadata and last-value data structures management. By leveraging QT framework's data structures, the system efficiently addresses several recurrent challenges, ensuring rapid data retrieval and optimal performance. As testing and system improvement continues, the promise of even greater efficiency in high-load, distributed system.

REFERENCES

[1] Simatic WinCC Open Architecture SCADA Software from ETM (Siemens subsidiary), `http://www.etm.at`

[2] R. Gomez-Reino *et al.*, "The Compace Muon Solenoid Detector Control System", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper MOB005, pp. 10-12.

[3] P. Golonka, M. Gonzalez-Berges, J. Guzik, and R. Kulaga, "Future Archiver for CERN SCADA Systems", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1442-1445. `doi:10.18429/JACoW-ICALEPCS2017-THPHA037`