EPJ Web of Conferences **245**, 04030 (2020) https://doi.org/10.1051/epjconf/202024504030
CHEP 2019

Development of the JUNO Conditions Data Management System

Xingtao Huang* On behalf of the JUNO Collaboration
Institute of Frontier and Interdisciplinary Science, Shandong University, Qingdao, Shandong, China

Abstract. The Jiangmen Underground Neutrino Observatory (JUNO) experiment is designed to determine the neutrino mass hierarchy and precisely measure oscillation parameters with an unprecedented energy resolution of 3% at 1 MeV. It is composed of a 20 kton liquid scintillator central detector equipped with 18000 20-inch PMTs and 25000 3-inch PMTs, a water pool with 2000 20-inch PMTs, and a top tracker. Conditions data, coming from calibration and detector monitoring, are heterogeneous, different type of conditions data has different write rates, data format and data volume. JUNO conditions data management system (JCDMS) is developed to homogeneously treat all these heterogeneous conditions data in order to provide easy management and convenient access with both Restful API and web interfaces, support good scalability and maintenance for long time running. The paper describes the status and development of JCDMS including the data model, workflows, interfaces, data caching and performance of the system.

1 Introduction

The Jiangmen Underground Neutrino Observatory (JUNO) experiment is designed to determine the neutrino mass hierarchy and precisely measure oscillation parameters with an unprecedented energy resolution of 3% at 1 MeV. The JUNO detector consists of a Central Detector (CD), a Water Cherenkov (WC) detector and a Top Tracker (TT). The CD filled with 20 kton liquid scintillator (LS) and equipped with 18000 20-inch Photomultiplier Tubes (PMTs) and 25000 3-inch PMTs is submerged in a water pool for shielding from natural radioactivity. The water pool is quipped with 2000 20-inch PMTs. A detailed description of the experimental apparatus can be found elsewhere [1].

Comparing with event data, condition data are non-event data. They play very important roles during almost all event data processing and physics analysis. The paper describes the status and development of the JUNO Conditions Data Management System (JCDMS) including the data model, workflows, interfaces, data caching and performances.

* Corresponding author : huangxt@sdu.edu.cn

2 Conditions data

The conditions data mainly come from detector configuration, calibration as well as detector running and monitoring. They are produced or updated with different frequency, stored in different formats, and their sizes are also different depending on the type of conditions data, from several kilobytes to several tens of gigabytes per year. On the other hand, the conditions data are also accessed with different frequency during event data processing and physics analysis. In short, the conditions data are heterogeneous, and they all vary with time. Therefore, it is a crucial task on how to effectively manage the conditions data, not only for short term access to these data, but also for long term maintenance and evolution of conditions data.

3 Conditions database system

The JUNO Conditions Data Management System (JCDMS) is developed in order to homogenously manage these heterogeneous conditions data. The system is designed with the Client-Server Structure, as illustrated in Fig. 1. In the server side, the MySQL or SQLite is used for underlying management of the conditions data, and the file system is also supported for the conditions data before they are validated or tested. In the client side, the system provides several ways to access to the conditions data via web interface or Conditions Database (CondDB) Service, the intermediate layer is added and implemented with Frontier/Squid [2-3] to provide data caching.

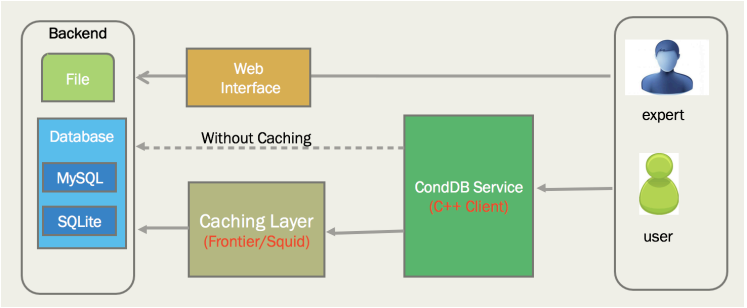

Fig. 1. Overview of JUNO Conditions Data Management System

3.1 Data model

The conditions data are logically divided into the conditions data objects and meta-data, which describes the object's validity and versioning. The data model at the level of MySQL consists of 6 tables, as shown in Fig. 2: Payload, IOV (Interval of Validity), Tag, Global_Tag, Tag_IOV_Map and Global_Tag_Map. The definition of the first four tables are similar with the ones of ATLAS and CMS experiment [4-5], the latter 2 tables are introduced to build the relations between Global_Tags, Tags and IOVs. It is noted that the Payload here can directly hold the conditions data object (in term of BLOB type) if the object has small data size, or external file references (such as the name and path of the files) for the conditions data object with larger data size. The current design will ease the schema and technology migrations during the long-term experiment running.

3.2 Work flow

The conditions data are categorized into different sub-system according to their features, such as frequency of writing and reading. There is one administrator or manager for each sub-system to create new Payloads, IOVs and Tags when new conditions data are produced or updated. One unique hash value defined in both Payload and IOV is used to correlate the Payload with its IOVs. The Tag_IOV_Map is used for Tag to collect a series of IOVs. The Top Manager (also called GT_Admin) can create a new Global_Tag for the whole set of conditions data. The procedures to release a new set of conditions data is described below:

- The Top Manager creates a new Global_Tag for a certain purpose,
- Then notify subsystem managers to request the specific tag for each sub-system.
- Each sub-system manager may create a new Tag or choose an old Tag, and send it to the Top Manager.
- The well-defined validation and tests will be launched on the set of conditions data.
- Global Tag will be frozen and released for the whole collaboration If the new set of conditions data pass through all the tests.

The workflows for writing, reading and updating conditions database have been supported by the web interface and python interface. The python interface is used to create Payloads and IOVs by Sub-System Managers, and the web interface is used to create Tags and Global Tags by Sub-System Managers and Top Managers.

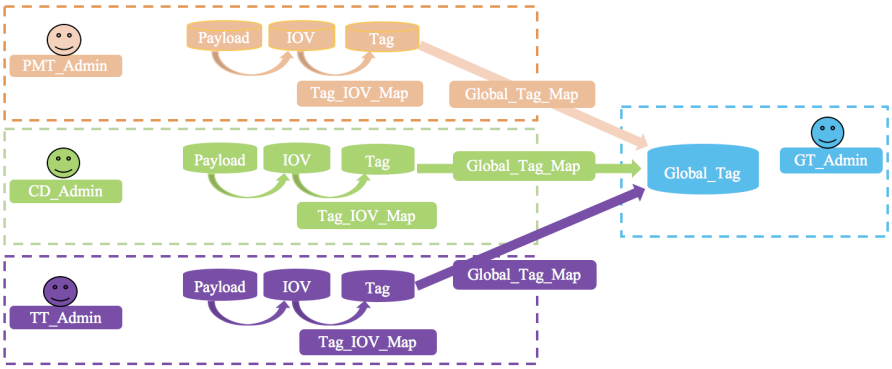

Fig. 2. Data model and workflow management

3.3 CondDB service

The conditions data will be mostly accessed by the offline software system which is developed based on the SNiPER framework [4]. Since algorithm, the basic event data processing unit, usually access to the data in terms of C++ objects (called transient conditions objects), a conditions database (CondDB) service is developed to query the meta-data and payload from the conditions database (called persistent conditions data) with the Global_Tag or Tag information, and then convert the persistent conditions data into the transient conditions objects. Streamers or Converters are provided for transient and persistent object conversion.

During the event loop, the CondDB service will automatically prepare the corresponding transient condition objects for algorithms according to the current processing event information (such as timestamp of the event) and the job configuration. For the moment, both the database and file system (such as JSON file) are supported to store conditions data. The function of switching from the database to the file system is very useful for the validation of new conditions data. The new condition data can be temporarily stored in the file for validations, once they pass through the validations, then these data will be inserted into the official central database.

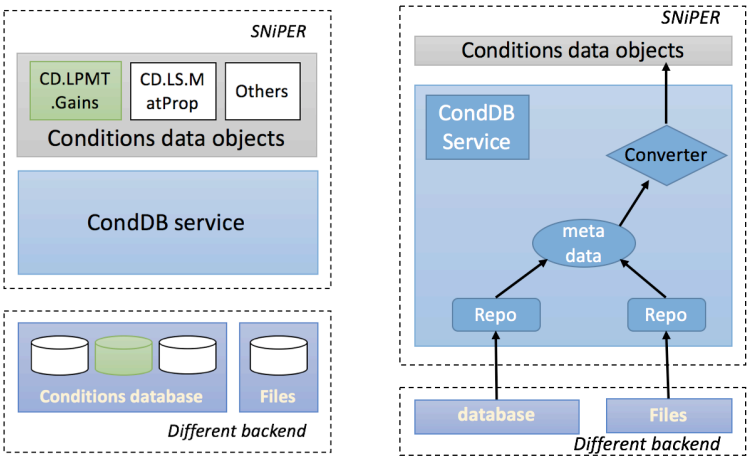

Fig. 3. The architecture of CondDB Service (left) and the conversion between transient conditions object and persistent condition data (right)

3.4 Data caching

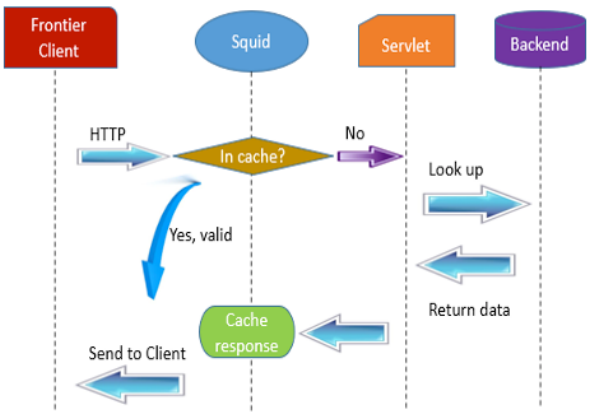

Fig. 4. Data caching of Frontier/Squid

During event data processing, algorithms use CondDB service to request conditions data object, CondDB will directly send the query to the center database in case of no data

caching layer, as the dash line shows in Fig. 1. That means the large amount of simultaneous access to the center database will happen. In order to decrease the heavy burden of center database, an intermediate layer (Frontier/Squid) between the client and server is adopted to provide data caching capability. In this case, as illustrated in Fig. 4, CondDB firstly sends the query to Frontier client, not the center database. After receiving the query, Frontier client will send the query to the local squid. If the result of the query has been cached in the local squid, the squid will send them to the Frontier client, and the Frontier client delivers the result to the CondDB. If the result has not yet been cached in the squid, the squid will send query to the servlet and look up the result in the center database.

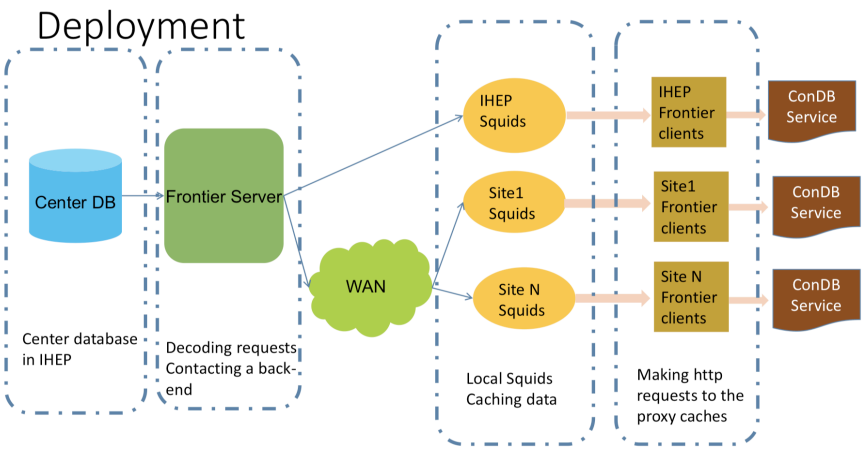

Fig. 5. The deployment of the center database, Frontier and Squid

The Frontier and Squid have been deployed as described in Fig. 5, both the center database and the Frontier Server are set up in IHEP, the local squid and Frontier clients are deployed in each site.

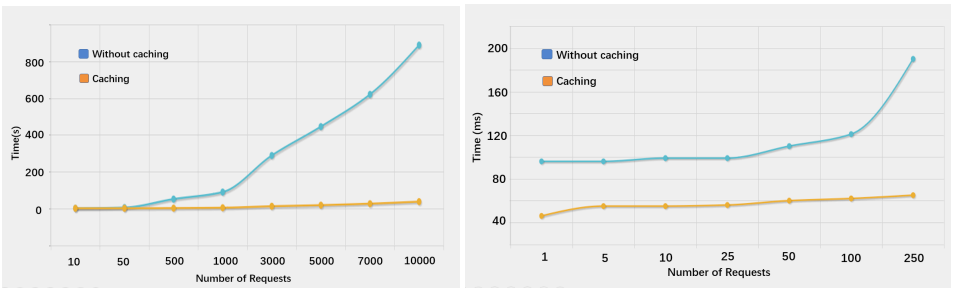

Fig. 6. Performance study and comparison with/without data caching

The comparison of performances of data transmission and response time with/without data caching have been made, as shown in Fig. 6. Every request is the same and needs to download some data, both yellow curves standing for the data caching show that the request with caching remain fast and stable with increased number of requests.

4 Summary

Conditions data plays a very important role for the event data processing and physics analysis. Conditions data are heterogeneous and vary with time. The design of the JCDMS can homogeneously treat all the conditions data and easily support its scalability and maintenance for JUNO long time running. The system is developed and provides several methods for the administrators and users to manage and access to the conditions data, and now being used for JUNO M.C. data production.

This work is supported by the Joint Large-Scale Scientific Facility Funds of the NSFC and CAS (U1532258).

References

1. JUNO collaboration, J. Phys. G: Nucl. Part. Phys. **43,** 030401 (2016)
2. D. Dykstra, J. Phys. Conf. Ser. **331**, 042008 (2011)
3. B. Blumenfeld, D. Dykstra, P. Kreuzer, R. Du, W.Z. Wang, J. Phys. Conf. Ser. **396**, 052014 (2012)
4. D. Barberis, A. Formica, E.J. Gallas, G. Govi, G. Lehman Miotto, A. Pfeiffer, J. Phys. Conf. Ser. **664**, 042015 (2015)
5. S. Di Guida, G. Govi, M. Ojeda, A. Pfeiffer, R. Sipos, J. Phys. Conf. Ser. **664**, 042024 (2015)
6. J. H. Zou, X.T. Huang, W.D. Li, T. Lin, K. Zhang, Z.Y. Deng and G.F. Cao, J. Phys. Conf. Ser. **664**, 072053 (2015)