Development of the JUNO Conditions Data Management System

Xingtao Huang¹, Ziyun Deng², Wenhao Huang¹, Weidong Li², Tao Lin², Qiumei Ma², Wenshuai Wang², Hongmei Zhang², Xiaomei Zhang², Jiaheng Zou²

(On behalf of the JUNO collaboration)
¹Shandong University, Qingdao, China
²Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China

CHEP2019,
Nov. 4-8, 2019, Adelaide, Australia
Outline

- JUNO Experiment
- Conditions Data
- Conditions Database System
  - Data Model
  - Web Interface
  - CondDB Service
  - Frontier/Squid for data caching
- Performance Testing
- Summary
Jiangmen Underground Neutrino Observatory (JUNO) Experiment

- Main Physics Goal: determine the neutrino mass hierarchy and precisely measure oscillation parameters with an unprecedented energy resolution of 3% at 1 MeV

- JUNO Detector
  - Central Detector (CD)
  - Water Cherenkov (WC)
  - Top Tracker (TT)
Conditions Data

- Conditions Data includes
  - Detector Configuration
    - Geometry, Material Optical Parameters and Calibration data
  - Detector Monitoring
    - High Voltage, Electronics Channels, LS Volume and Temperature,...

- Conditions Data are heterogeneous and varying with time
  - Different write rate, access rate, data volume and data format

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Write Rate</th>
<th>Access Rate</th>
<th>Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>WC_PMT_Timing</td>
<td>Once Per Year</td>
<td>~ kHz</td>
<td>~ 100 MB Per Year</td>
</tr>
<tr>
<td>CD_PMT_Gain</td>
<td>Once Per Day</td>
<td>~ kHz</td>
<td>~ 10 GB Per Year</td>
</tr>
<tr>
<td>DqChannel</td>
<td>Once Per Mins</td>
<td>~ kHz</td>
<td>~ GB Per Year</td>
</tr>
<tr>
<td>Material_LS</td>
<td>Once Per Years</td>
<td>~ kHz</td>
<td>~ KB Per Year</td>
</tr>
</tbody>
</table>

How to effectively manage these Conditions Data is a crucial task!
Overview of Conditions Database System

- **Client-Server Structure**
  - Server side: underlying database is flexible
  - Client side: Several methods are provided for access to DB

- Data caching capability is implemented with Frontier/Squid, which has been used by ATLAS and CMS
**Data Model**

Like CMS, Six database tables are used to manage conditions data and describe their validity

- **4 Metadata tables**
  - **Payload**: holds conditions data or the path of the conditions data file
  - **IOV (Interval of Validity)**: describes the period of payload validity
  - **Tag**: collect a set of IOVs
  - **Global Tag**: collects a set of Tags

- **2 Auxiliary tables**
  - **Tag IOV Map**
  - **Global Tag Map**

With this design, Conditions Data is separated from Metadata, It is good for scalability and maintenance for the long-time running of Exp.
Data Management

- According to the features, conditions data is divided into different types, for example, PMT, Central Detector (CD) and Top Tracker (TT).
- There is one manager for each type of conditions data.
**User Interface**

- **User Interfaces** are setup for managers/experts/users
  - Web Interface
  - CondDB Service
Web Interface to create new payloads

- Web interface is developed for Data Manager to
  - Upload the data files into web server by ftp command
  - Create new payloads
  - Create new IOVs
  - Create new Tags
  - Add IOVs into Tag
Web Interface to create new IOVs and Tag

- Web interface is developed for Data Manager to
  - Upload the data files into web server by ftp command
  - Create new payloads
  - Create new IOVs
  - Create new Tags
  - Add IOVs into Tag
Web Interface to create a global tag

- Top Manager creates a global tag for all types of conditions data
  - Create a global tag for certain purpose
  - Notify subsystem managers
  - Subsystem managers decide which tag should be included for this global tag
  - Global tag will be frozen after validation and testing

![Table with operation columns](image1)

![Table with tag columns](image2)
- CondDB Service is provided for access to Database in the offline software system
- It is implemented with a service of SNiPER, which is the offline software Framework developed by JUNO collaboration
CondDB Service retrieves the suitable conditions data (Object) from Conditions Database according to the current event time.
**CondDB Service**

**Conditions Data**
- used by algorithms in term of C++ Object (Transient Condition Object)
- Varying with the time
- Need to declare which condition objects will be used in the algorithms

![Diagram](image-url)
**CondDB Service**

- **Conditions Data**
  - used by algorithms in term of C++ Objec (Transient Condition Object)
  - Varying with the time
  - Need to declare which condition objects will be used in the algorithms

- **Persistent Condition Object**
  - Support several backend with different format and priorities
    - Temporal Database/Files for testing and validation
    - Official Database for data production
CondDB Service

- **Conditions Data** used by algorithms in term of C++ Object (Transient Condition Object)

- **Persistent Condition Object** in Multiple format and managed by conditions database
  - Database
  - Files

- **CondDB Service**
  - Perform conversions from Persistent Object to Transient Object
  - Updates the condition objects during the event loop
Data Caching Layer

- An intermediate layer (Frontier/Squid) between the client and server is adopted to provide data caching capability.
- To decrease the heavy burden of center database when thousands of jobs query the same conditions data at the same time.
Frontier System

- **Frontier Client**
  - C++ API
  - Called by CondDB Service

- **Squids**
  - HTTP proxy
  - Cache data locally

- **Frontier server**
  - Decode requests
  - Contact with a backend

More details about Frontier: https://twiki.cern.ch/twiki/bin/view/Frontier/FrontierOverview
Workflow of Data Caching

- Frontier Clients firstly send query to Squids, not the Center Database.

- If the requested results has been cached in local squid, Frontier clients could get data directly from squid.

- The System automatically updates the cached data frequently at very low cost:
  - Make sure the cached data being updated in time.
  - Frequency for updating is configurable.
Testing System

Deployment

Center DB
- Center database in IHEP

Frontier Server
- Decoding requests
- Contacting a back-end

WAN

IHEP Squids
- Local Squids
- Caching data

Site1 Squids
- Making http requests to the proxy caches

Site N Squids

IHEP Frontier clients
- ConDB Service

Site1 Frontier clients
- ConDB Service

Site N Frontier clients
- ConDB Service
Performance Testing

- Every request is the same and needs to download some data
- The request with caching remain fast and stable with increased number of requests
Summary

- Conditions data takes very important role for the event data processing and physics analysis

- Conditions Data are heterogeneous and vary with time

- JUNO Conditions Database System is developed to homogeneously manage all Conditions Data
  - Data Model
  - Web Interface
  - CondDB Service
  - Frontier/Squid for data caching

- The System has been tested and optimized to have good performance, and is being used for JUNO M.C. Data Production
Thanks a lot!