DUNE database development

Ana P. Vizcaya Hernandez and Lino Gerlach for the DUNE collaboration

CHEP 2023
8 – May – 2023
Content

• Brief introduction to DUNE and ProtoDUNE
• Introduction to ProtoDUNE’s databases
  – Database architecture
  – Metadata and Conditions data
• Run History Database
  – Versioning, workflow, and schema
• Conditions Database
  – HSF recommendations, performance optimization, and deployment
DUNE (Deep Underground Neutrino Experiment)

• DUNE is a next generation neutrino experiment that uses LArTPC technology
  • High-intensity (MW-scale) neutrino beam is produced at Fermilab as part of LBNF
  • Travels ~1300 km to a far detector at the Sanford Underground Research Facility
  • Measured by the near detector complex about 0.5 km from the neutrino production

• Physics: neutrino oscillation, detection of solar and supernova neutrinos, beyond the standard model physics.
ProtoDUNE at CERN

- ProtoDUNE-SP and -DP are DUNE’s large scale prototypes of its far detector modules.
- Located at CERN Neutrino Platform
- Critical to demonstrate viability of LArTPC technology
- 770 tons LAr mass each
- Exposed to test beams H4(SP) and H2(DP)
- Also take cosmic ray data
• DUNE will produce vast amounts of **metadata**, which describe the data coming from the read-out of the primary DUNE detectors.
**Purpose**

Centralized place to store all the conditions data

**Goals**

- Contain all the conditions metadata that can be relevant for offline analysis
- Prevents a priori schema

**Folders:**
- sp_protodune
- test_protodunell
- IFBeam

**Objects:**
- configuration
- blob-run#
- blob-time#

**Version:**
- Stored with run key
- Stored with validity

---

**Run History Database**
1. After each run, the metadata is extracted from the different databases and sent to the Master Store of Metadata.

2. A subset of metadata is sent to the Run History DB where it is stored with run granularity.

3. Users access the metadata in the database via MetaCat. An example query is given.

```sql
metacat filter dune_runsdb_incondb () (files from dune:all) where runs_history.hv = 120
```
Run History Database - Versioning

- New payloads can be added to the Run History Database, they will be treated as a new version.
- Changes of a single condition parameter in a run require a new entry in the database, the newest entry of each run is return except otherwise specified.

Database schema, where the conditions_# payloads represent all the conditions parameters in the database.
Conditions Database - Introduction

**Purpose:** provide all of the non-trigger-record data needed for offline data processing

### Changes over time
- Repeat detector calibration with larger cosmic dataset
- Improve calibration algorithms

### High access rates
- Distributed computing jobs access same conditions data simultaneously
- Access rates up to ~kHz

### Heterogenous data
- Granularity varies (time indexed, run-indexed, constant)
- Structure of payload varies (3D map, single number, …)

Versioning & configuration
Fast DB queries & effective caching
Payload agnostic by design

Similar challenges for various HEP experiments
Conditions Database – HSF Recommendations

• Dedicated HEP Software Foundation (HSF) conditions data activity: https://hepsoftwarefoundation.org/activities/conditionsdb.html

• Key recommendations for conditions data handling
  • Separation of payload queries from metadata queries
  • Schema below to enable appropriate configuration
Conditions Database - Implementation

**nopayloadclient**: client-side stand-alone C++ tool
- Experiment unspecific
- Communicates with server-side **nopayloaddb**
- Local caching
- Handling of payloads

*See Ruslan’s talk: ‘The HSF Conditions Database reference Implementation’ link*
Conditions Database – Performance

- Simulate expected DB occupancy
  - Vary number of Payload Types & IOVs
- Simulate access patterns:
  - Parallel requests using HTCondor or Multithreading
- Response times and -frequencies as performance metrics
Conclusions

• We are developing two databases, run history DB and conditions DB, that will help users get all the data files and conditions data they need for offline analysis.

• The Run History Database which consists of just a subset of metadata from the Master Store of Metadata will allow users to select runs/files of runs with specific characteristics.

• Experiment-agnostic conditions database with DUNE-specific client-side library.

• Conditions database performance has been investigated, looks good.
Thank you!

We acknowledge the support of the department of energy through the grant
Conditions Database – Performance

- Example results of a test campaign (absolute values don’t matter here)
- Scaling tests: single number as metric of whole campaign

![Graph showing mean response time and frequency with descriptive statistics](image)

**no caching**

total calls: 10000
n_threads: 10
n_calls: 1000
wall clock [s]: 42.2
avg. f [Hz]: 237.1
acc. pattern: first
n_pll: 1
n_iow: 100000
• Compared default Django ORM and optimized raw SQL query
• Horizontal scaling (deployment): response frequency plateaus around 3 pods
• >250 Hz for worst-case (>5M) occupancy, random access, no caching