Functional tests of a prototype for the CMS-ATLAS common non-event data handling framework

A. Formica[1], A. Pfeiffer[2], G. Govi[3], G. Franzoni[2], M. Musich[4], R. Sipos[5]
For the ATLAS and CMS Collaborations

[1] CEA/IRFU,Centre d'etude de Saclay Gif-sur-Yvette
[2] European Organization for Nuclear Research, CERN
[3] Fermi National Accelerator Laboratory
[4] Université Catholique de Louvain (UCL)
[5] MTA-ELTE Momentum CMS Particle and Nuclear Physics Group

CHEP 2016 - San Francisco

Overview

- Intro
  - Conditions data
  - The collaboration
  - Goal of the project
- Prototype
  - Details
- Client side
  - CMS software (CMSSW) integration
- Functional testing
- Outlook

Conditions data

Non-event data, used to describe the CMS and ATLAS detectors and constitute an essential ingredient to reconstruct events optimally and exploit the detectors' potential.

<u>Data categories</u>:

- Detector calibrations and alignments
- Control systems (HV, LV,...)
- Run information
- Beam information
- Trigger configuration
- Detector status

The collaboration

CMS went through a substantial change for Run2

- New system developed during LS2 to handle conditions data
- Exploited lessons learned in Run1

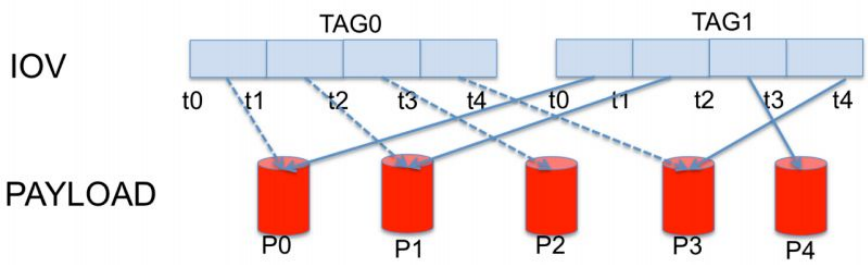
Collaboration with ATLAS started in June 2014

- Found several commonalities in the overall approach
- Discussion lead to a common view of the Data Model
- A common strategy on the architecture under discussion

The development that are shown in the following slides are coming from a joint effort of ATLAS and CMS.

Data model

- Payload
  - Container class defining conditions
  - Persisted/Stored as a whole (BLOB)
- IOV - Interval of validity
  - Time interval of events for which a given payload is to be consumed
- Tag
  - Label assigned to an IOV sequence
  - Identify the condition data for a specific system
- Global Tag
  - Consistent set of Tags, targeted for a specific workflow



Objective

Goals:

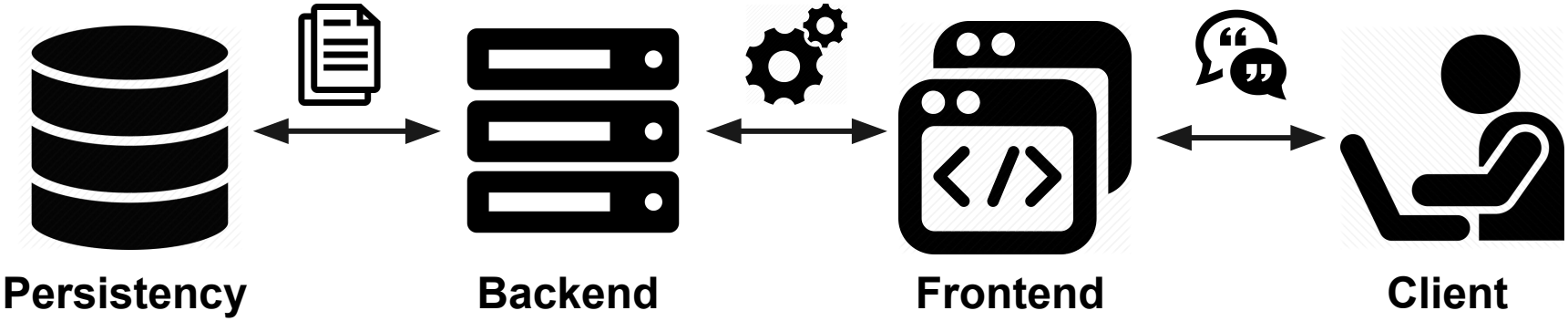
- Offer a service to access the conditions data fully integrated with a data caching system
- Disentangle the client code from specific storage solutions
- Aggregate resources for the interested experiments to address common requirements

Strategy:

Don’t repeat yourself (DRY) principle!

Architecture

Main **elements** to handle Conditions data:



Prototype

An application that decouples the mentioned elements, using state-of-art technologies and frameworks, and developing a client in the CMS software.

We use the *Java Platform, Enterprise Edition (Java EE)* to support a generic infrastructure for handling conditions.

Reference:

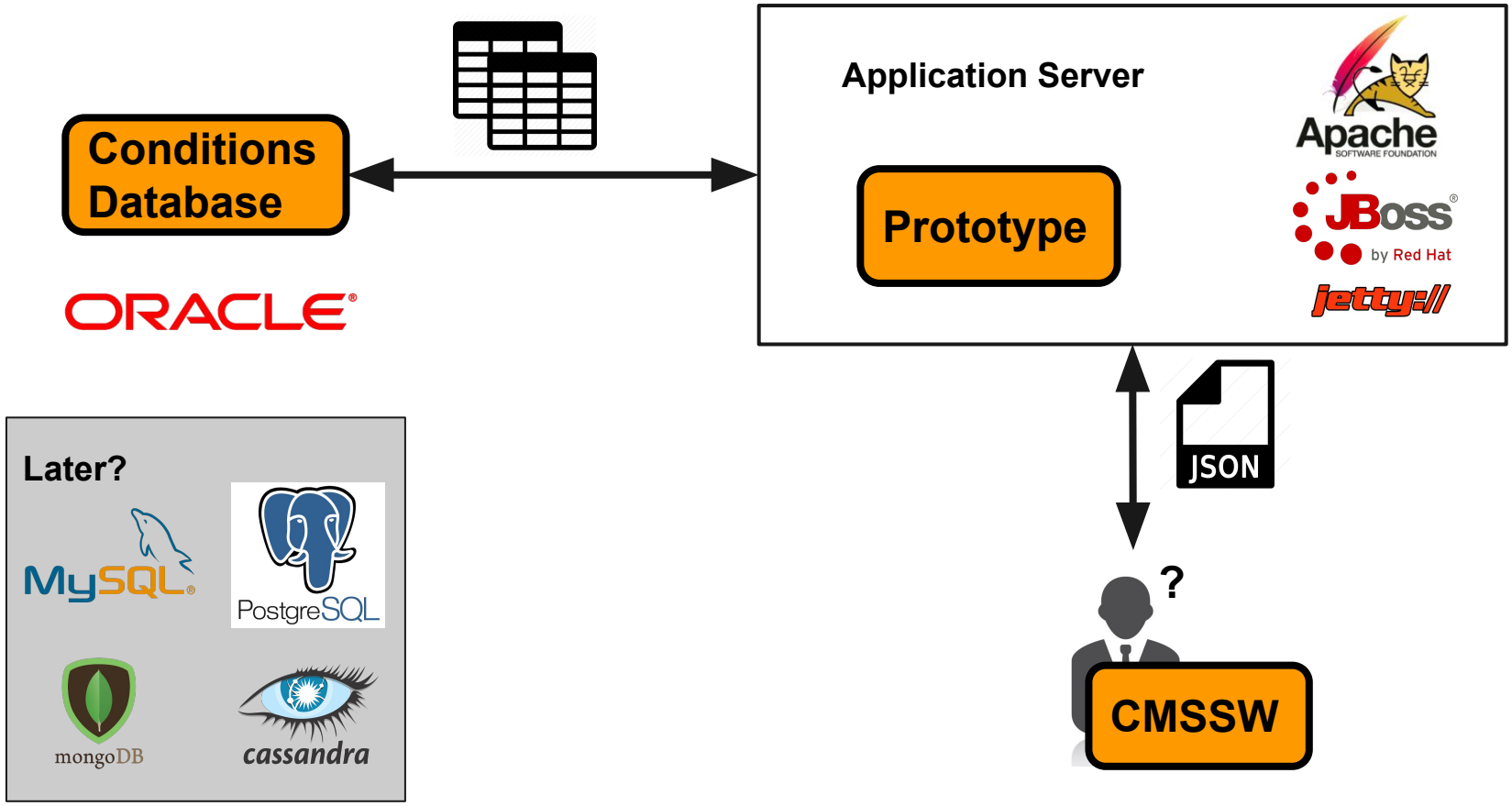
https://indico.cern.ch/event/496146/contributions/1174798/

Technologies

<u>Proposed technologies for the prototype:</u>

1. Persistency
   a. Database, *Oracle*
   b. Object relational mapping (ORM), *Hibernate*
2. Backend
   a. Model-view-controller fwk., *Spring*
3. Frontend
   a. REST-API, *JAX-RS/Jersey*
4. Client,
   a. Simple clients, *Swagger*
   b. CMSSW, *CPR curl wrapper* *(C++ Requests: Curl for People)*

Application overview



CMSSW integration

Development of a test bed in the CMS software.

- Using the *CPR* tool *(C++ Requests: Curl for People)*,
  - Easy to install, using new standards,
  - It is basically a *libcurl wrapper*.
- JSON parsing
  - *Boost::property_tree*,
  - or header-only JSON parsers.
- Payloads are read with a callback function through CURL.

<u>Using REAL data and REAL production workflows for the qualification of the prototype.</u>

Functional tests

Verify, that for a standard set of CMS validation workflows, the consumed conditions are exactly the same between the tested (prototype) and the reference (Frontier connection) version.

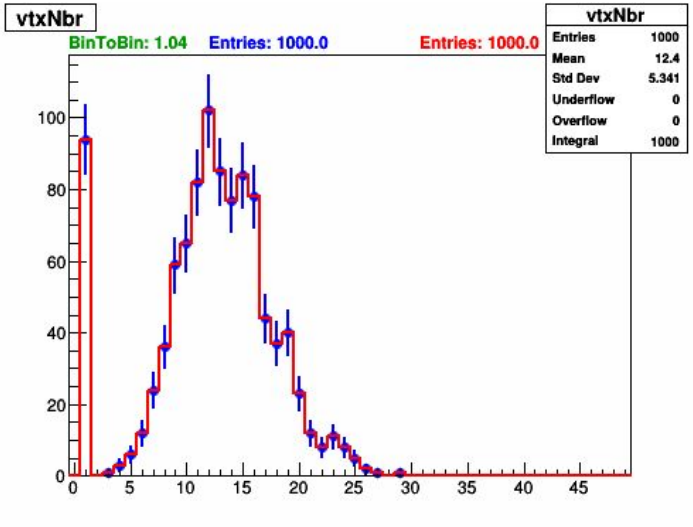
Workflow includes 2 processing steps:

1. High level trigger, whose output is fed to step (2)
2. Reconstruction of raw data
    a. Equipped with large set of diagnostic plots filled with the reconstructed quantities

Results

The results of these tests:

- We verified, that all such diagnostic plots are identical to the reference point.



We processed:

- <u>1000 events,</u>
- <u>with 92000 plots.</u>

Example: a histogram with the number of vertices which the reconstruction program has identified in p-p data. Physics quantity which is very sensitive to many of the non event-data consumed in CMSSW.

Conclusion

- Verified functionality
  - Identical results compared to the original solution
- Unifies the Condition handling for the experiments
- Flexibility
  - Following the DRY principle:
    Different elements are independent
    - Example: With the current prototype, the replacement of Oracle with some other database would have no impact on the client

Outlook

- Integration
  - ATLAS software framework
    - And deployment for functional testing
  - Frontier
- Deployment for testing in larger scale
  - Dedicated resources for the application
- Fine-tuning
  - Host and application configuration
- Performance testing
  - Load time of conditions for different workflows

End

Thank you for your attention!

Any questions, suggestions are welcome!