sPhenix Conditions Database

Ruslan Mashinistov, Paul Laycock

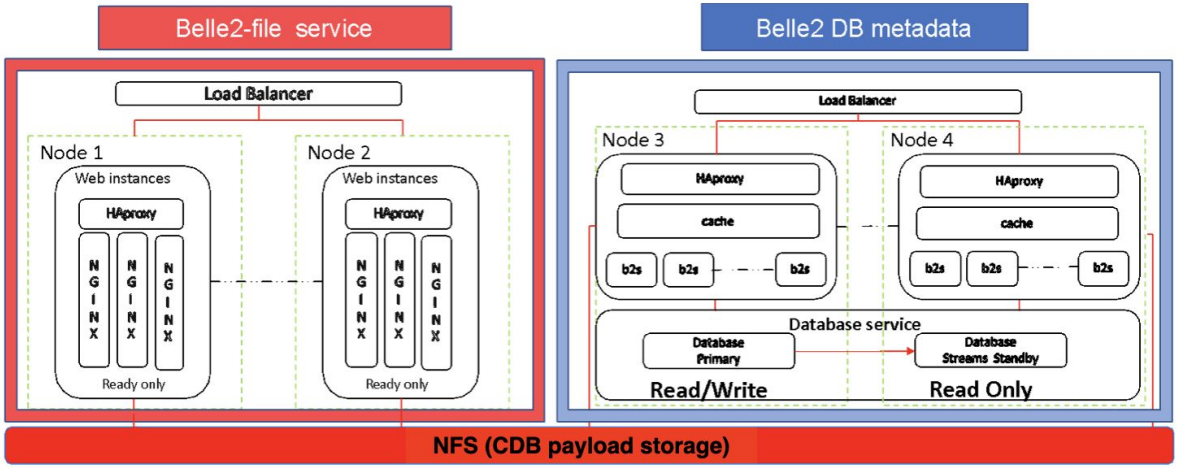
09/03/21

Introduction

- As a starting point we used the BelleII Conditions Database
  - NPPS software, SDCC operations
  - Working stably in production since 2018
  - Originally from PNNL, written in Java (I'm main developer now)

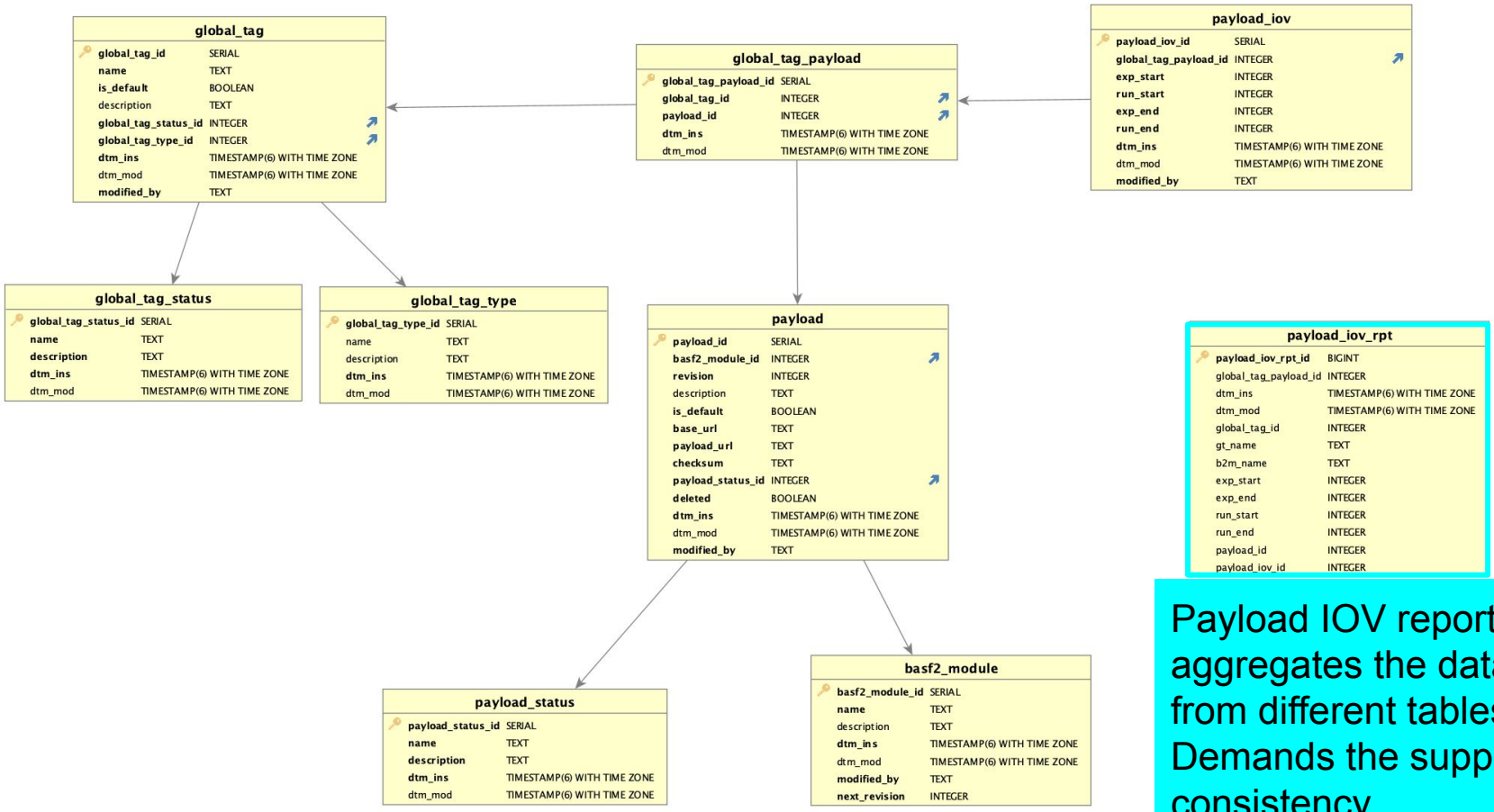
- Design concept: separate metadata from payloads
  - The metadata DB is separated from payloads so the CDB service effectively returns a list of URLs to retrieve the payloads

The Belle II Conditions DB



- A file service (left) serves the conditions data payloads
- The metadata service returns the list of payloads URLs. On the client side it is trivial to modify the path with a different prefix, so the preferred location of payloads is usually cvmfs

B2 Conditions Database schema



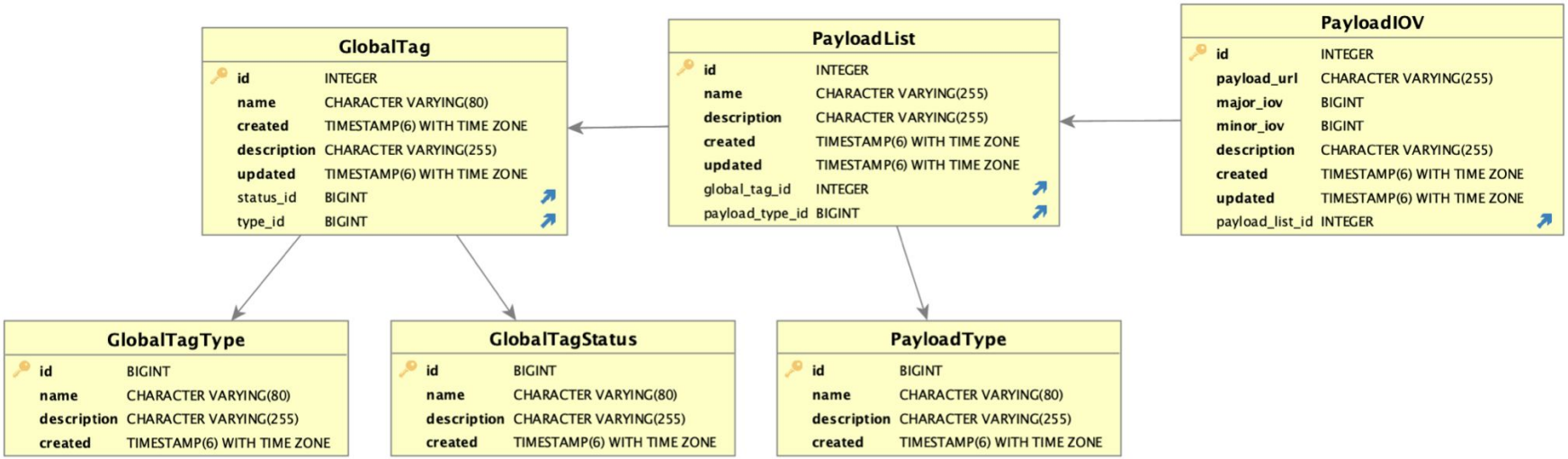
Payload IOV report table aggregates the data from different tables
Demands the support for consistency

Brookhaven National Laboratory

Streamlining the Server code and DB schema

- Use this as an opportunity to simplify both the code base and DB schema
- Change from Java to Django for simplicity and easier to support
- This is an opportunity for Belle II that we are happy to take

New CDB schema



- Implemented schema provides simple navigation from the Global Tag to the Payloads
  - GT has Type and Status (Locked || Unlocked)
  - Payloads are grouped by Payload Lists which collect all of the calibrations of the same type. List also has Type attribute
  - Payloads stored together with IOVs presented by two BIGINT fields: major and minor IOVs - start of the Payload validity.
    - One IOV ends where the next IOV starts

APIs

- Django implementation is based on rest_framework library
- Implemented APIs:
  - “Create” (POST) endpoints. Accept object as JSON
  - Query current Payloads for a given Global Tag and IOVs
    - Main call from software framework
  - Deep copy of the Global Tag

API calls

- API can be called with the standard (GET||POST) requests using cURL or program libraries
- An example of the cURL GET Request:

curl 'http://127.0.0.1:8000/api/cdb/payloadiovs/?gtName=TestMedGT5&majorIOV=0&minorIOV=1630825065'

[{"id":8977,"name":"TestMedPayloadList0","global_tag":{"id":116,"name":"TestMedGT5","description":"","created":"2021-09-02T07:33:57.612512","updated":"2021-09-02T07:33:57.612525","status":1,"type":1},"payload_type":{"id":1,"name":"Type1","description":"","created":"2021-08-04T13:54:47.835407"},"payload_iov":[{"id":6279079,"payload_url":"testPayloadMed4999_8977","major_iov":0,"minor_iov":1630576781,"payload_list":8977,"created":"2021-09-02T09:59:41.649975"}],"created":"2021-09-02T07:33:57.638153"},...

Get example

```
GET /api/cdb/payloadiovs/?gtName=ARICHdata&majorIOV=3&minorIOV=3
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 8246,
        "name": "ARICHModuleTest",
        "global_tag": {
            "id": 106,
            "name": "ARICHdata",
            "description": "",
            "created": "2021-08-27T10:39:03.282104",
            "updated": "2021-08-27T10:39:03.282180",
            "status": 3,
            "type": 1
        },
        "payload_type": {
            "id": 1,
            "name": "Type1",
            "description": "",
            "created": "2021-08-04T13:54:47.835407"
        },
        "payload_iov": [
            {
                "id": 792659,
                "payload_url": "dbstore/ARICHModuleTest/dbstore_ARICHModuleTest_rev_3.root",
                "major_iov": 0,
                "minor_iov": 2,
                "payload_list": 8246,
                "created": "2021-08-27T10:39:04.370853"
            }
        ],
        "created": "2021-08-27T10:39:03.741195"
    },
```

- GT and its Payloads were migrated from the BelleII CDB
- The output is the list of Payload Lists with PayloadIOVs

Payload URL,
major and minor IOVs

POST Request

- POST Requests accept JSON body with Objects definition
- Global Tag creation example. GT's definition as JSON:

```
{"name": "TestGT1",
 "status": 1,
 "type": 1}
```

curl --header "Content-Type: application/json" --request POST --data '{"name": "TestGT1","status": 1, "type": 1}' http://127.0.0.1:8000/api/cdb/gt

Brookhaven National Laboratory

Python example

```python
#Create PIOV
base_url = 'http://127.0.0.1:8000'
url = base_url + '/api/cdb/piov'

for i in range(0,5000):
    iov = int(time.time())
    pname = 'testPayload_%d' % (i)
    piov = {
           'payload_url': pname,
           'payload_list': payloadListId,
           'major_iov':  0,
           'minor_iov': iov
           }

    r = requests.post(url = url, json=piov)
    data = r.json()
    payload_id = data['id']
```

```python
# Get PIOVs
base_url = 'http://127.0.0.1:8000'
url = base_url + \
      "/b2s/rest/v2/iovPayloads/?gtName=%s&majorIOV=%d&minorIOV=%d" \
      % (gtname,majIOV,minIOV)

r = requests.get(url=url)
data = r.json()
```

- Examples of population the Payload List by 5k PayloadIOVs and retrieving of the PayloadIOVs

Brookhaven National Laboratory

Scalability tests

- Moderate usage of IOVs would be calibrations updated every day (26 weeks of running time)
  - 1/day, 7/week, ~200/year of running
  - This is 20k payloads assuming 100 payload lists
- Heavy usage of IOVs would be calibrations updated every hour
  - 1/hour, 24/day, 168/week, ~5k/year of running
  - This is 500k payloads assuming 100 payload lists
- Worst case scenario is that calibrations are updated every 10 minutes,
  - 6/hour, 144/day, 1k/week, ~26k/year of running
  - Worst case use 200 payload lists to make it painful!
  - This is 5 million payloads!

Conclusions

- "Demonstrator" prototype is implemented
- Next step is to run number of scalability tests including test with multiple clients running in parallel
  - Prepare 3 testing Global Tags according to the moderate, heavy and worst case scenarios
    - All Payloads have unique names
    - Use minorIOV only
- Looking at code optimization