The HSF Conditions Database reference implementation

Andrea Formica[1], Lino Gerlach[2], Giacomo Govi[3], Paul Laycock[2], Ruslan Mashinistov[2], Chris Pinkenburg[2]

[1]Université Paris-Saclay (IRFU/CEA, FR)
[2]Brookhaven National Lab (US)
[3]Fermilab (US)

8-12 May 2023

@BrookhavenLab

Overview - Conditions data

“Conditions data is any additional data needed to process event data”

Changes over time	High access rates	Heterogenous data
• Repeat detector calibration with larger cosmic dataset • Improve calibration algorithms	• Distributed computing jobs access same conditions data simultaneously • Access rates up to ~kHz	• 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

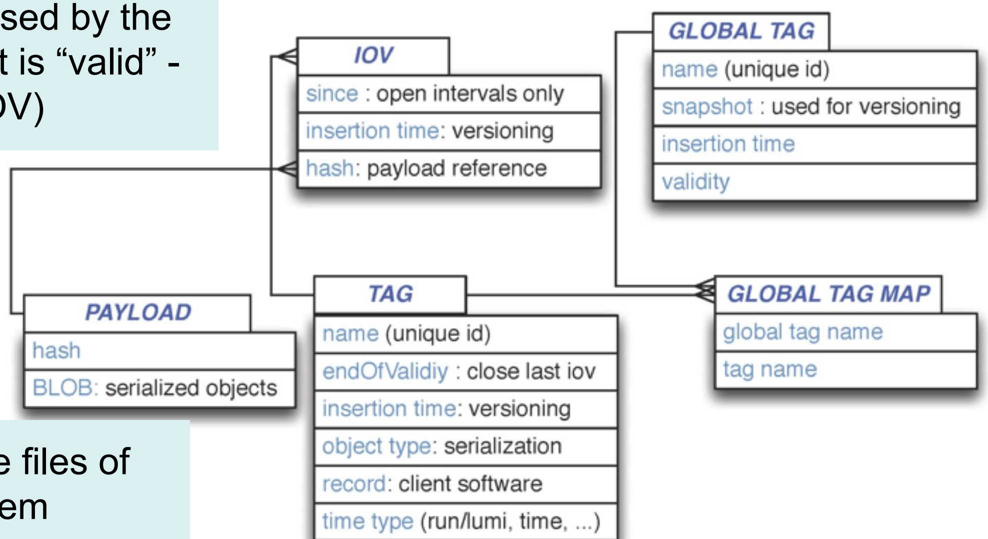
Brookhaven National Laboratory

Conditions Data – HSF Recommendations

HEP Software Foundation

- The HSF Conditions Databases activity is a CERN-based forum for cross-experiment discussions with as broad an audience as possible: https://hepsoftwarefoundation.org/activities/conditionsdb.html
- Key recommendations for conditions data handling
  - Separation of payload queries from metadata queries
  - General Conditions Database schema design (below) proposed by the White Paper [HSF-CWP-2017-03](HSF-CWP-2017-03)

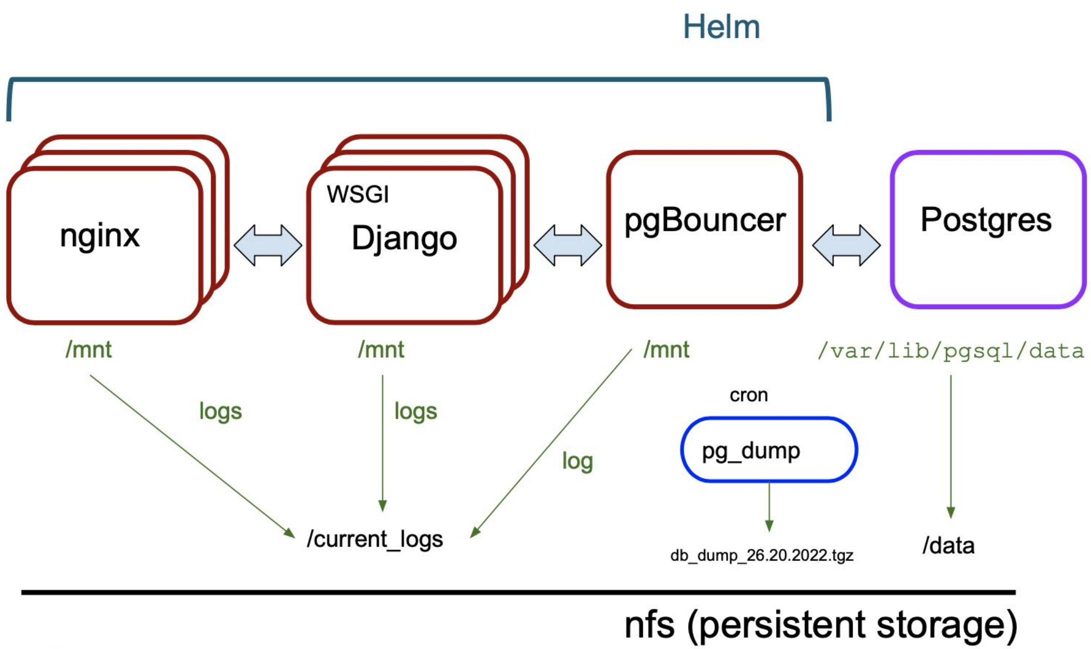
Payload is characterised by the period of time when it is “valid” - Interval of Validity (IOV)



Global Tag (GT) is the top-level configuration of all conditions data.

Typically Payloads are the files of data per detector subsystem

NoPayloadDB Deployment



- Postgres DB with persistent storage on nfs
- pgBouncer - DB pooler
- NoPayloadDB Django-application running under gunicorn (WSGI server). 5 Pods
- nginx - web server. 5 Pods

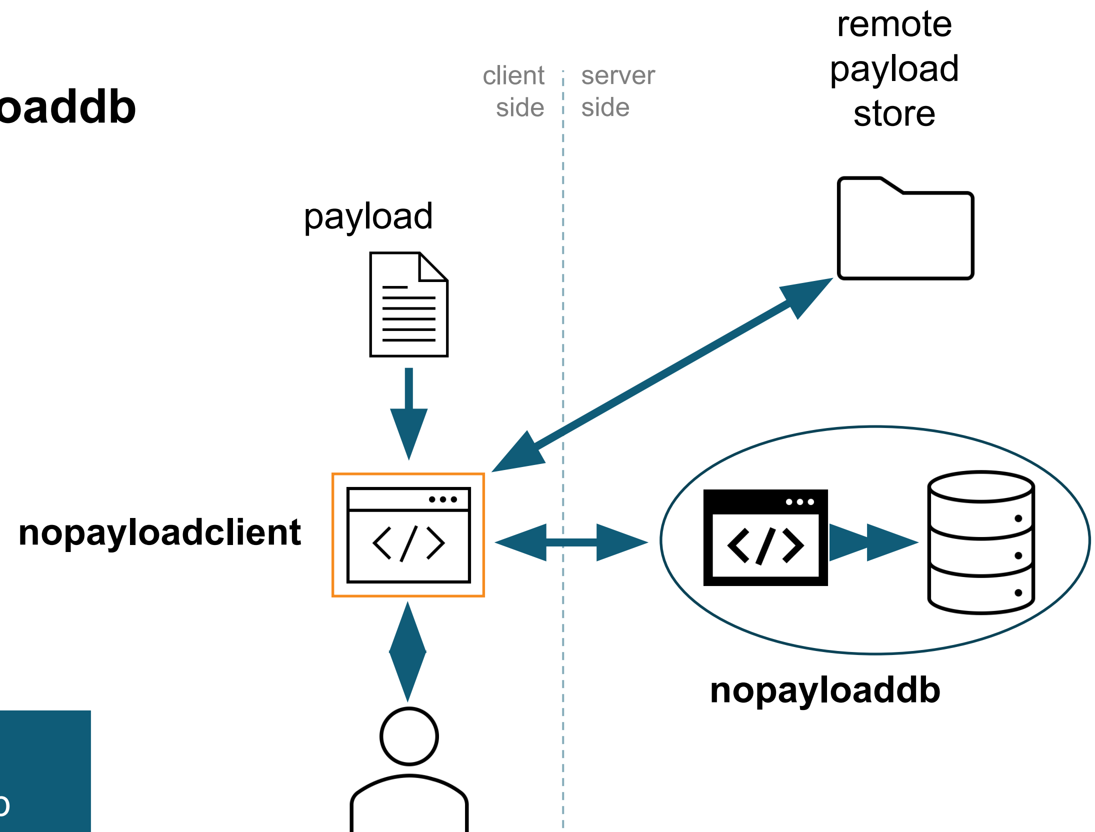
- Fully automated deployment (Helmchart based) at the OKD (open source container application platform) makes NoPayloadDB an easily adoptable, scalable and attractive solution for HEP experiments

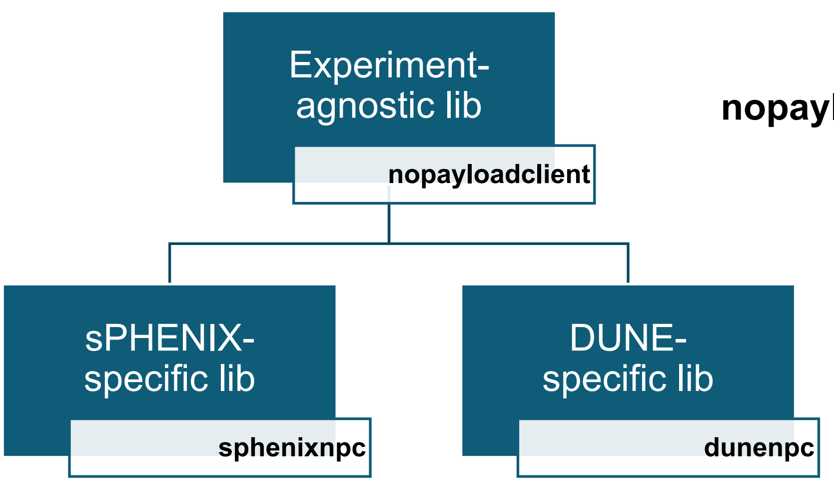
Brookhaven National Laboratory

HSF Reference Implementation - Client

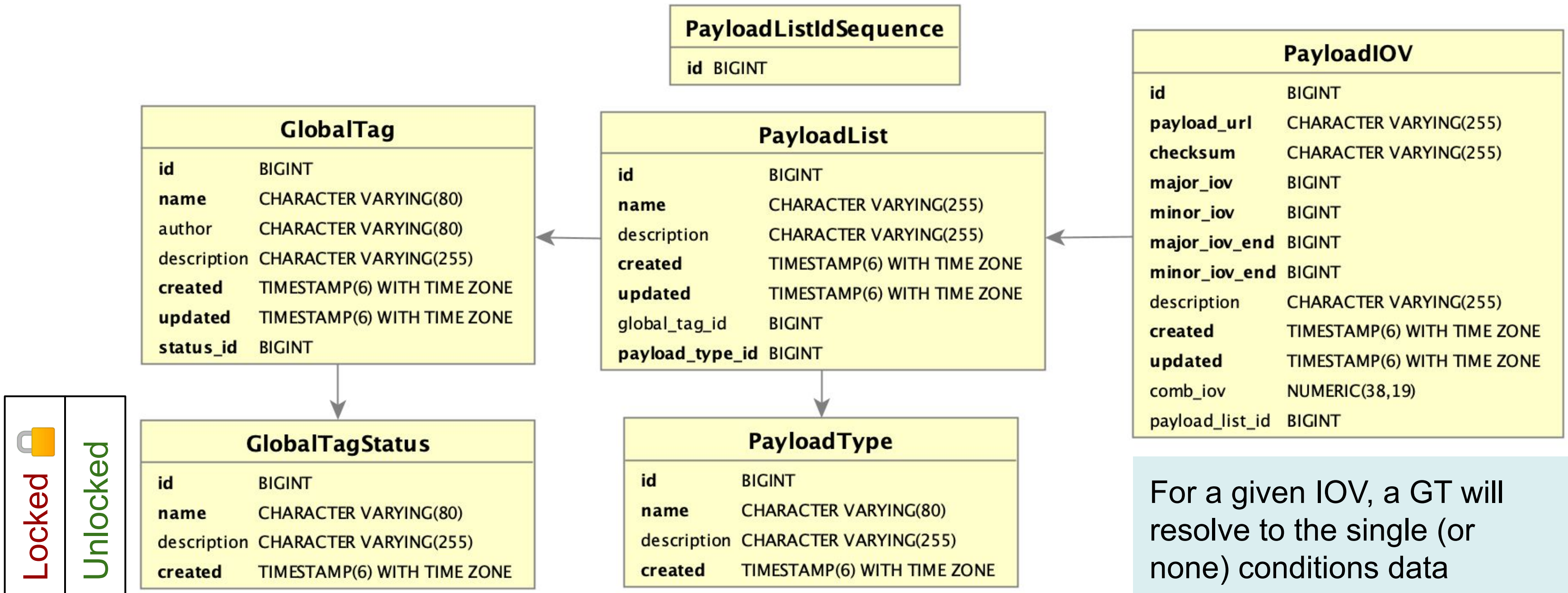
nopayloadclient: Client-side stand-alone C++ tool

- Experiment unspecific
- Communicates with **nopayloaddb**
- Local caching
- Handling of payloads





NoPayloadDB DB schema



For a given IOV, a GT will resolve to the single (or none) conditions data payload per each detector subsystem

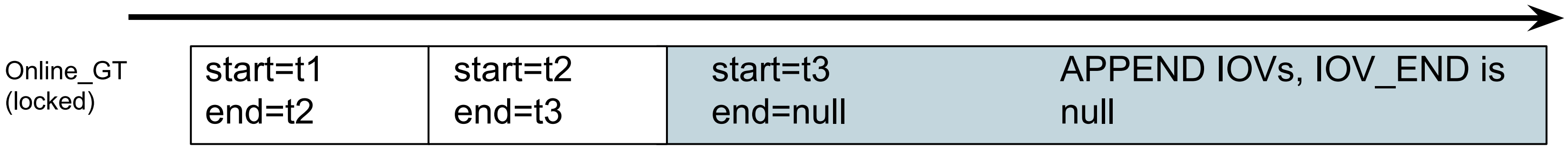
PayloadType allows to collect all of the calibrations of the same detector subsystem

- IOVs presented by two fields: **major** and **minor IOVs**
- Combined IOV - integer part for **major** and fractional parts for **minor IOV**
- PayloadIOVs also has ending IOVs

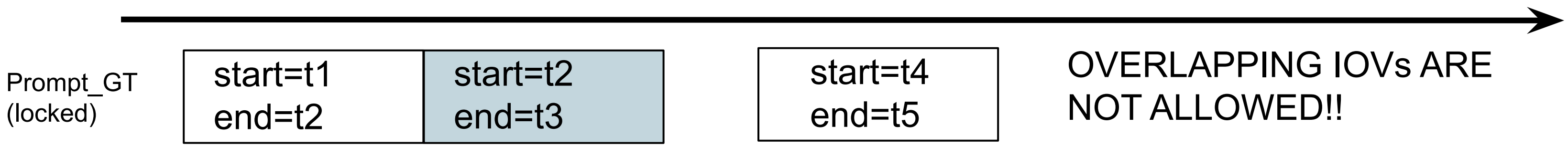
Brookhaven National Laboratory

GT workflows

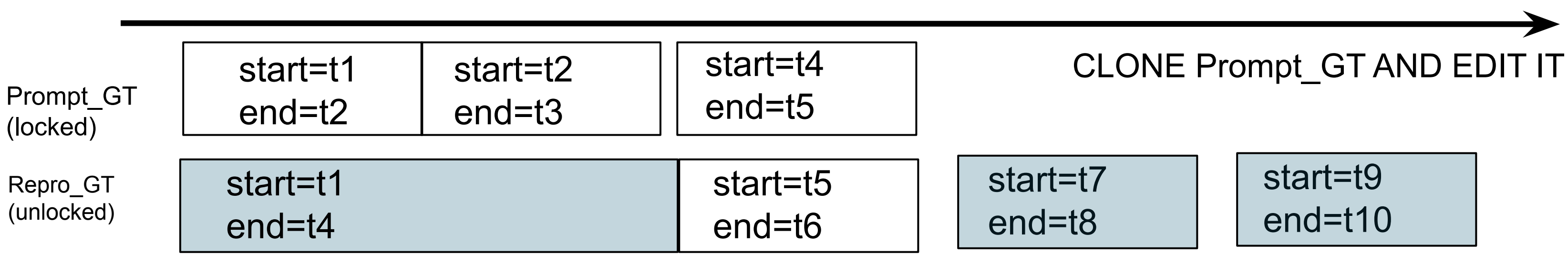
- Online GT - global tag consequently growing at time and returning the latest conditions



- Prompt GT - overwriting isn't allowed



- Prompt GT - overwriting isn't allowed



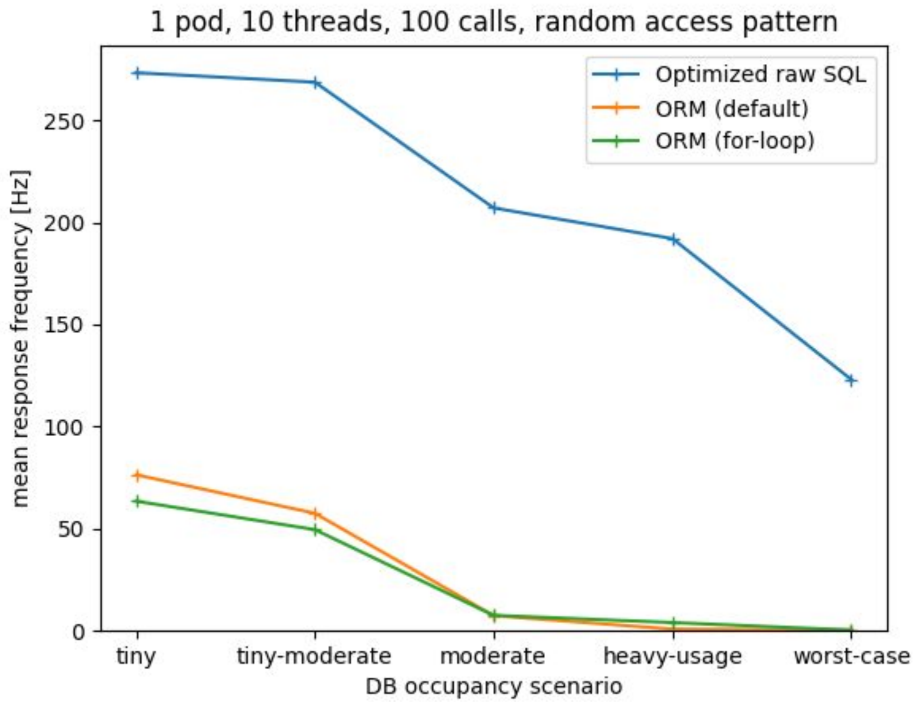
While editing, the service will take care of removing unnecessary PIOVs

Brookhaven National Laboratory

ORM vs SQL tests

Two reference read APIs with Django ORM were implemented with the following logics:

- Group Payloads by type. Then descending order by IOV and distinct per type
- Get maximal IOV per type and append to the final output



Scenario	Global Tags	Payload Types	Payload IOVs (per type)	Update Rate
tiny	1	10	100 (10)	
tiny-moderate	1	10	2000 (200)	
moderate	1	100	20000 (200)	1 day
heavy-usage	1	100	500000 (5000)	1 hour
worst-case	1	200	5200000 (26000)	10 minutes

Brookhaven National Laboratory

PayloadIOV read API

Retrieving latest Payloads of each type for the given GlobalTag and IOVs

```
SELECT pi.payload_url, pi.major_iov, pi.minor_iov,
pt.name, …
FROM "PayloadList" pl
JOIN "GlobalTag" gt ON pl.global_tag_id = gt.id AND
gt.name = %(my_gt)s
JOIN LATERAL (
  SELECT payload_url, major_iov, minor_iov, …
  FROM   "PayloadIOV" pi
  WHERE  pi.payload_list_id = pl.id
    AND pi.comb_iov <= CAST(%(my_major_iov)s +
CAST(%(my_minor_iov)s AS DECIMAL(19,0)) / 10E18 AS
DECIMAL(38,19))
  ORDER BY pi.comb_iov DESC
  LIMIT 1
) pi ON true
JOIN "PayloadType" pt ON pl.payload_type_id = pt.id;
```

For each PayloadList (Type)

Get Payloads descending ordered by combined IOV - arranged from last to first

Limit return to 1 line - latest Payload for a given IOVs

And then append the results of each subquery to create the final output

Use raw SQL

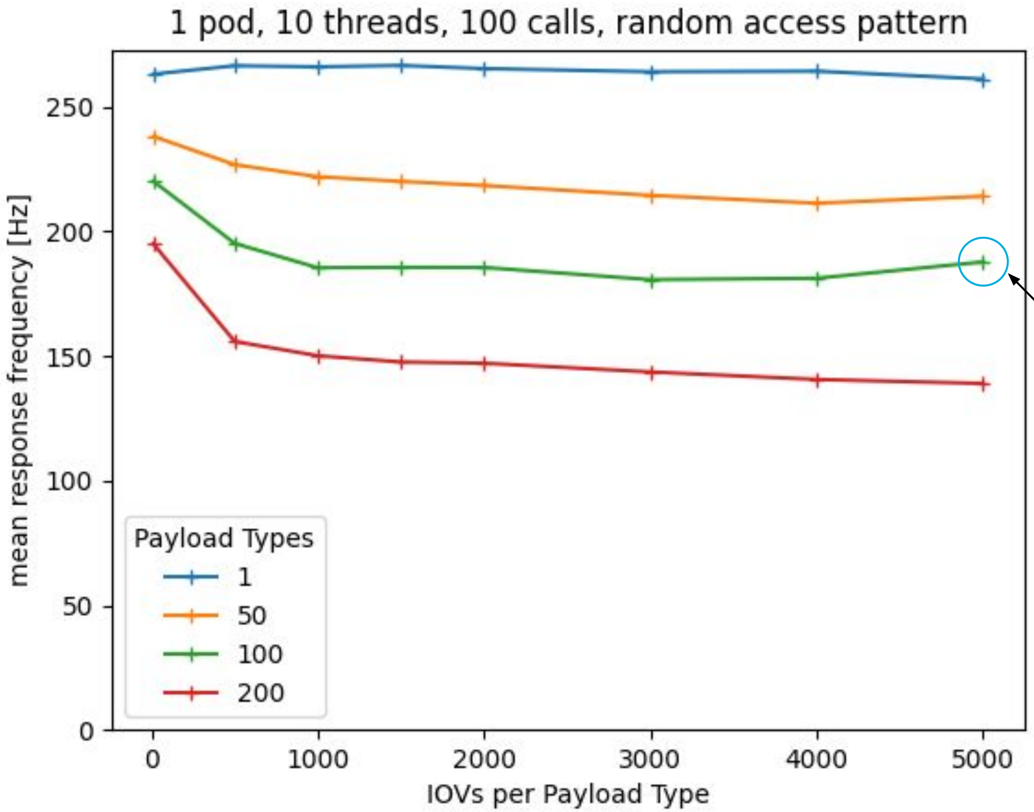
Django Documentation on 'Database Access Optimization':

Write your own custom SQL to retrieve data or populate models. Use **`django.db.connection.queries`** to find out what Django is writing for you and start from there.

- LATERAL joining. Without LATERAL, each sub-SELECT is evaluated independently and so cannot cross-reference any other FROM item
- Covering index on Payload table including combined IOV and reference to the PayloadList

Brookhaven National Laboratory

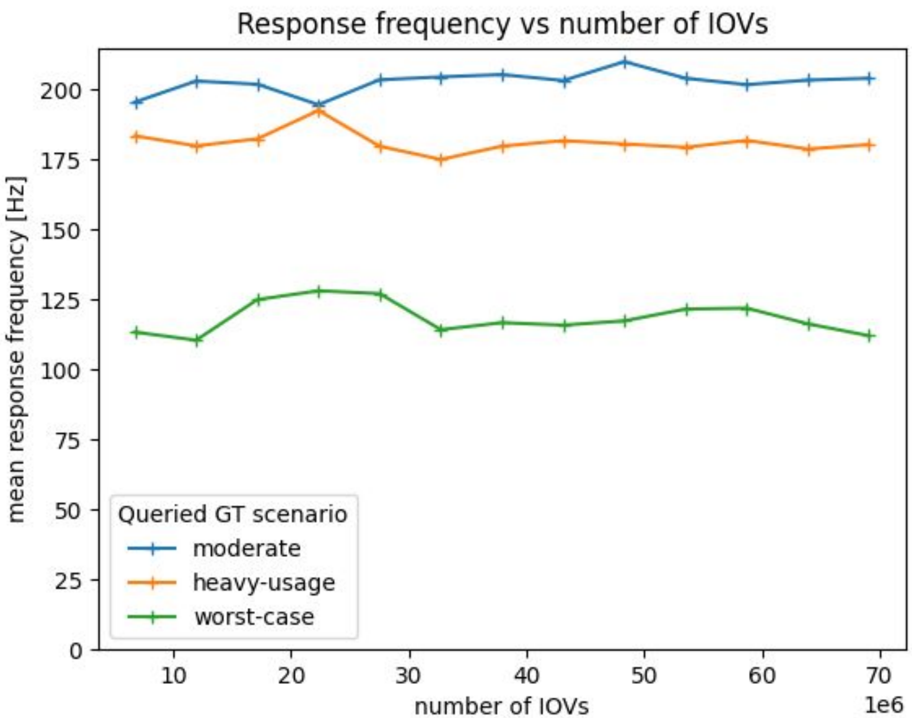
Scalability tests 1/2



- Each dot on the plot is independent GT
    - All 32 GTs was created at the same time
- Testing dependance from the number of payload types in the GT
    - Different colours vary in number of payload types
- The x-axis is the number of IOVs per payload type

Heavy-usage GT:
200 Payload types x 5000 IOVs

Scalability tests 2/2



- Query Payloads for the given GT and random IOV
  - Testing 3 scenarios for the used GTs
- Populating the DB by cloning "worst-case" GT
  - After 12 clones (13*5.2M = ~68M rows), postgres pod reached the limit for the persistent volume (20 GB)

Scenario	Global Tags	Payload Types	Payload IOVs (per type)	Update Rate
tiny	1	10	100 (10)	
tiny-moderate	1	10	2000 (200)	
moderate	1	100	20000 (200)	1 day
heavy-usage	1	100	500000 (5000)	1 hour
worst-case	1	200	5200000 (26000)	10 minutes

Brookhaven National Laboratory

Summary

- NoPayloadDB is the first HSF reference implementation of conditions database
- Fully automated deployment at the OKD makes NoPayloadDB an easily adoptable, scalable and attractive solution for HEP experiments.
    - NoPayloadDB was chosen by sPHENIX experiment and was commissioned for production at the beginning of May 2023.
    - Moreover, other experiments, such as the protoDune and Belle2, express interest in the service

Backup

-

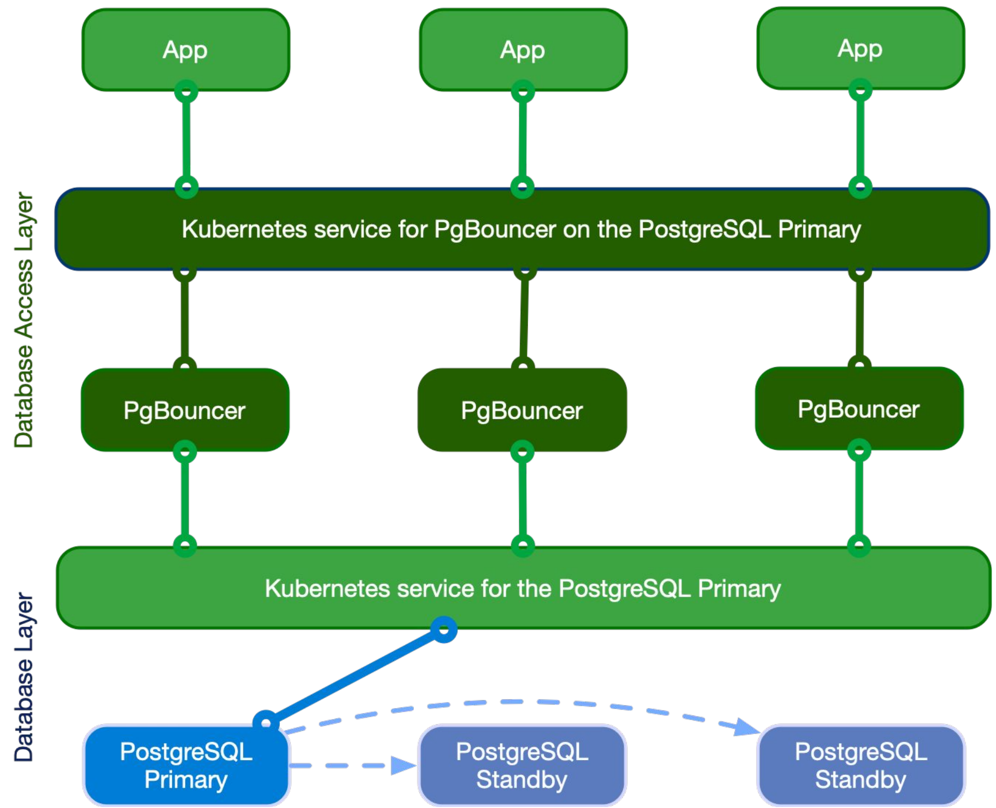
PostgreSQL HA cluster

We're considering using the CloudNativePG

Open source Kubernetes operator for HA PostgreSQL

CloudNativePG provides:

- Covers the full lifecycle of a HA PostgreSQL cluster
- Primary/standby architecture, using native streaming replication
- Native support for connection pooling with PgBouncer



Brookhaven National Laboratory