

# Update on conditions DB deployment

Lino Gerlach, Paul Laycock

26.10.2022

# Reminder

## Long term goal:

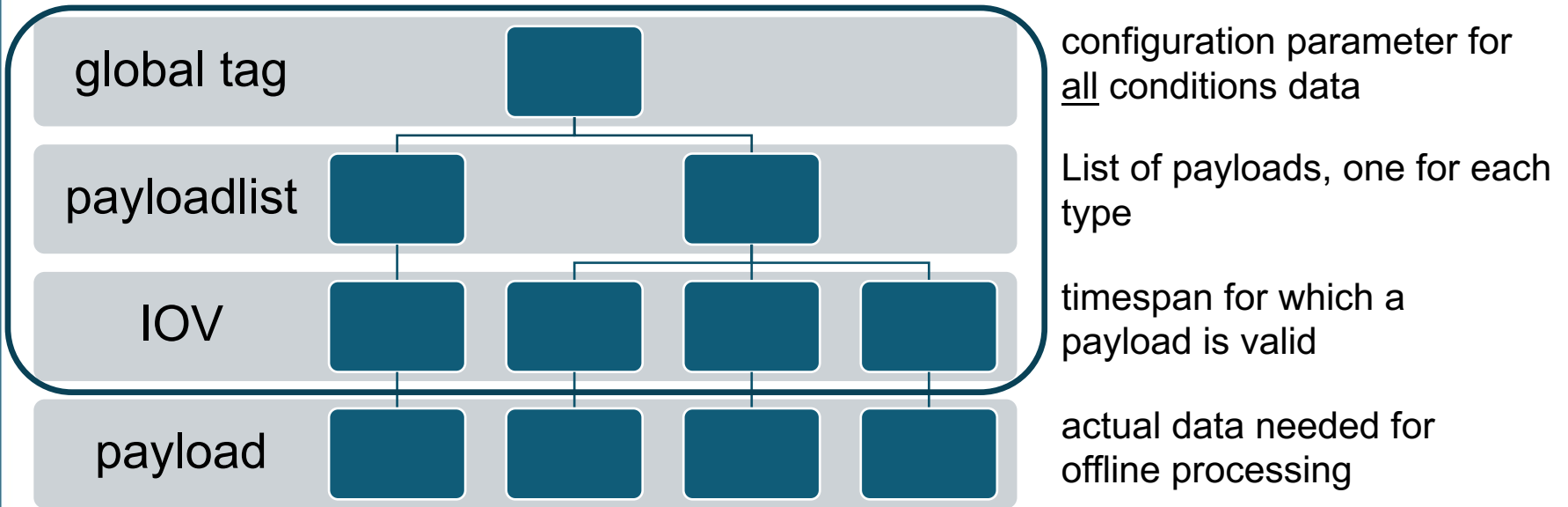
- Develop conditions DB for DUNE
  - ProtoDUNE as 'testing ground'
- Use existing, experiment-unspecific database 'nopayloadddb'
  - Designed according to HSF recommendations
  - Developed by Ruslan Mashinistov
- Deploy on Fermilab resources

## What I am currently working on:

- Deployed test instance of nopayloadddb at BNL (SDCC)
- Run performance tests on that instance
- Develop simple client-side C++ library to use the REST API

# nopayloaddb - Overview

## Postgres + Django application

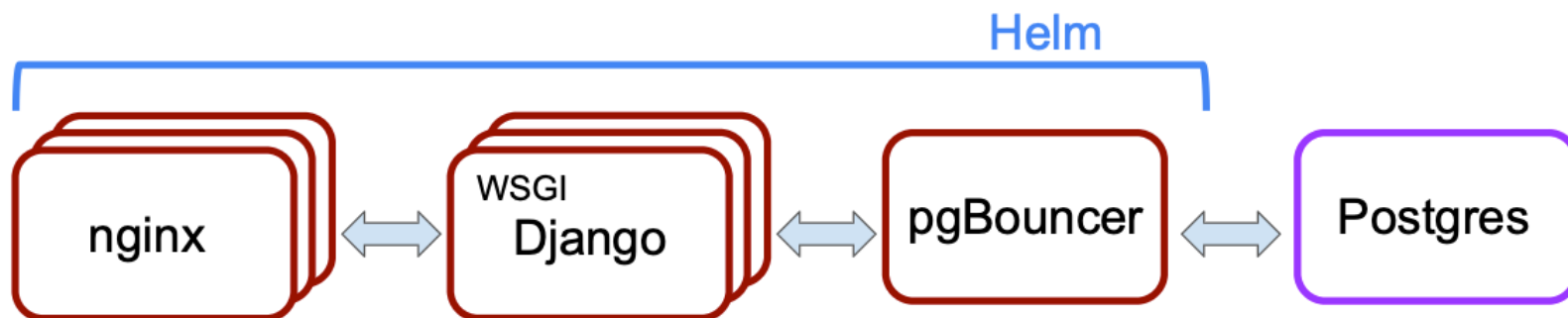


url of a payload can be retrieved via curl:

```
curl http://<host>/api/cdb_rest/payloadiovs/?gtName=test_gt&majorIOV=42
```

... and then pick the correct type from the resulting dictionary

# Deployment of nopayloaddb via OKD



From Ruslan Mashinistov

- Single command-line deployment
- Helm configuration (values) defines main parameters: OKD (Open Shift) project name, db credentials, URL ...

- Shown here: deployment configuration at BNL/SDCC (for sPhenix Experiment)
- Fermilab will have a separate Postgres service (outside of OKD)
- My test instance only has 1 Pod for each service (could be scaled up)

# Development of client-side tool

Idea: Stand-alone c++ tool to communicate with nopayloadddb

- Read & write operations on DB
- Handling of payload files: copy to remote storage, compare checksums
- Experiment unspecific
  - (Proto)DUNE specific stuff only in LArSoft Service

## Current Progress

- Basic functionality and error handling is implemented
- Wrote unit tests for envisioned workflow
- No checksum comparison yet
- No performance optimization and caching yet

# Testing 'nopayloadddb' performance

- Simulate typical use case
  - Access to service by many jobs in parallel
- Use HTCondor to create <n> jobs, making <m> calls to service each
  - Call to service via 'curl' bash command or custom compiled tool
- After all jobs finish, extract & summarize error codes & response times

## Example 'executable' for HTCondor job:

```
#!/usr/bin/bash
```

```
url=http://linostest.apps.usatlas.bnl.gov/api/cdb_rest/payloadiovs/?gt  
Name=test_gt&majorIOV=42
```

```
for i in $(eval echo {1..$1})  
do
```

```
    start=`date +%s.%N`
```

```
    httpcode=$(curl --write-out '%{http_code}' $url)
```

```
    end=`date +%s.%N`
```

```
    runtime=$(echo "$end - $start" | bc -l)
```

```
    echo runtime=$runtime
```

```
    echo httpcode=$httpcode
```

```
done
```


Define endpoint




Calls per job



Perform 'curl' (or  
compiled c++ code) &  
write out error code

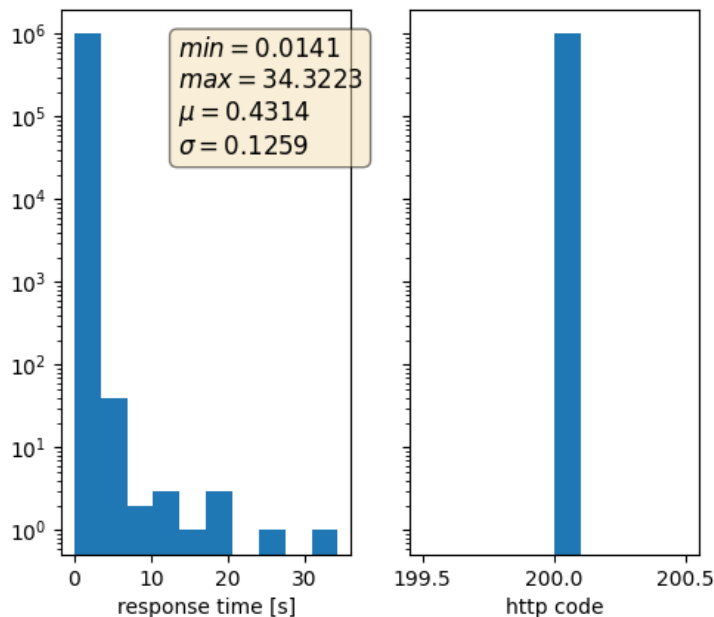


Write out error code &  
response time

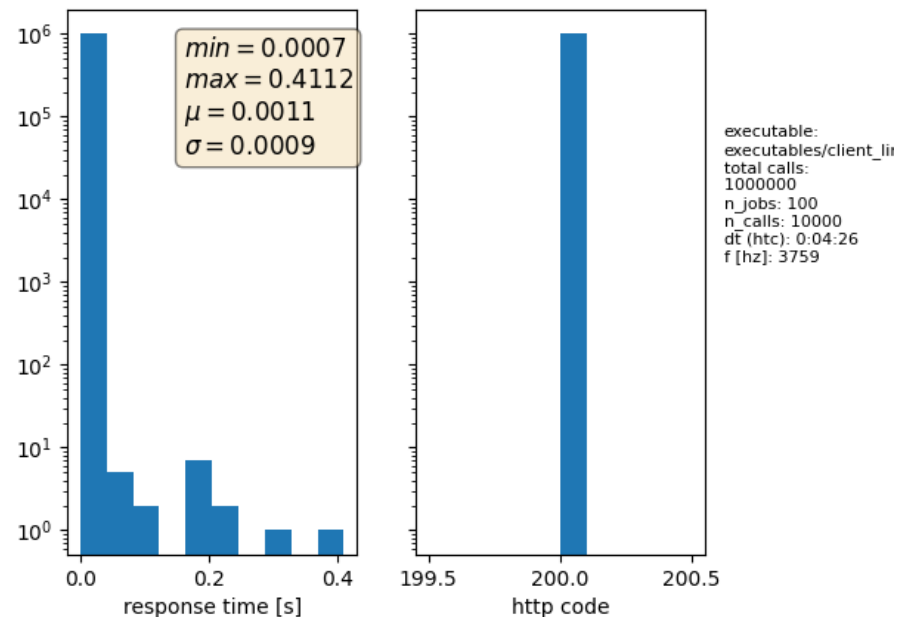


# Performance of test instance

Example: 100 jobs, making 10,000 calls each (1,000,000 calls in total)



via 'curl' bash command



via compiled c++ tool (based on libcurl)



# Conclusion & Outlook

## Conclusion

- Deployed test instance of nopayloaddb at BNL
- Tested its performance (with one pod for each service)
  - Looks solid:  $10^6$  accesses w/o error, quick response times
- Started implementing C++ client-side tool for nopayloaddb

## Outlook

- Further performance testing:
  - Additional pods, more realistic access patterns, fill DB w/ more data
- Let people work with client-side tool
  - Implement their feedback