Update on the HSF Conditions Database

Lino Gerlach[1], Ruslan Mashinistov[1], Michael Kirby[1]

[1]*Brookhaven National Lab (US)*

06 February 2024

@BrookhavenLab

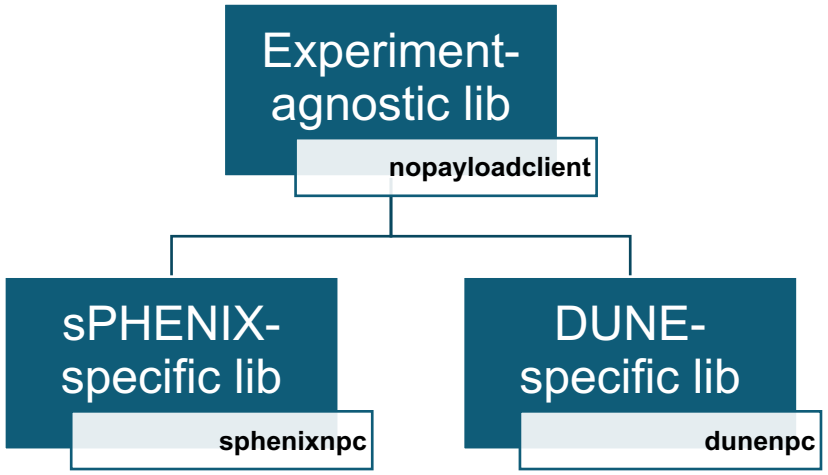
Introduction

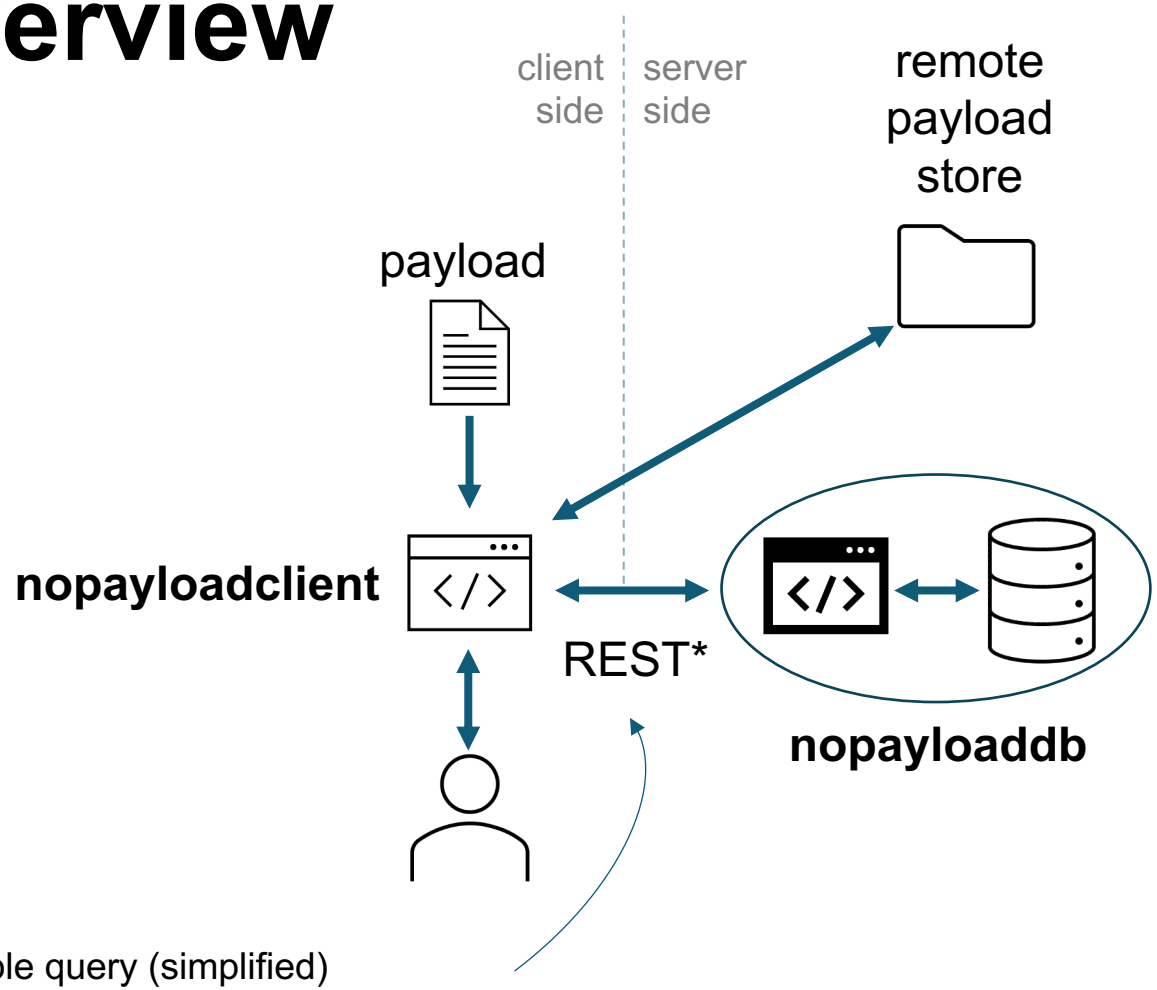
- Brief reminder of HSF Reference implementation
- Recent developments
  - Server-side caching
  - Single-container deployment
- Status of integration into DUNE
  - Test instance @CERN
  - Future production instance @BNL
- Conclusion & Outlook

Implementation – Overview

nopayloadclient:

- Client-side stand-alone C++ tool
- Communicates with **nopayloaddb** (server)
- Local caching
- Handling of payloads

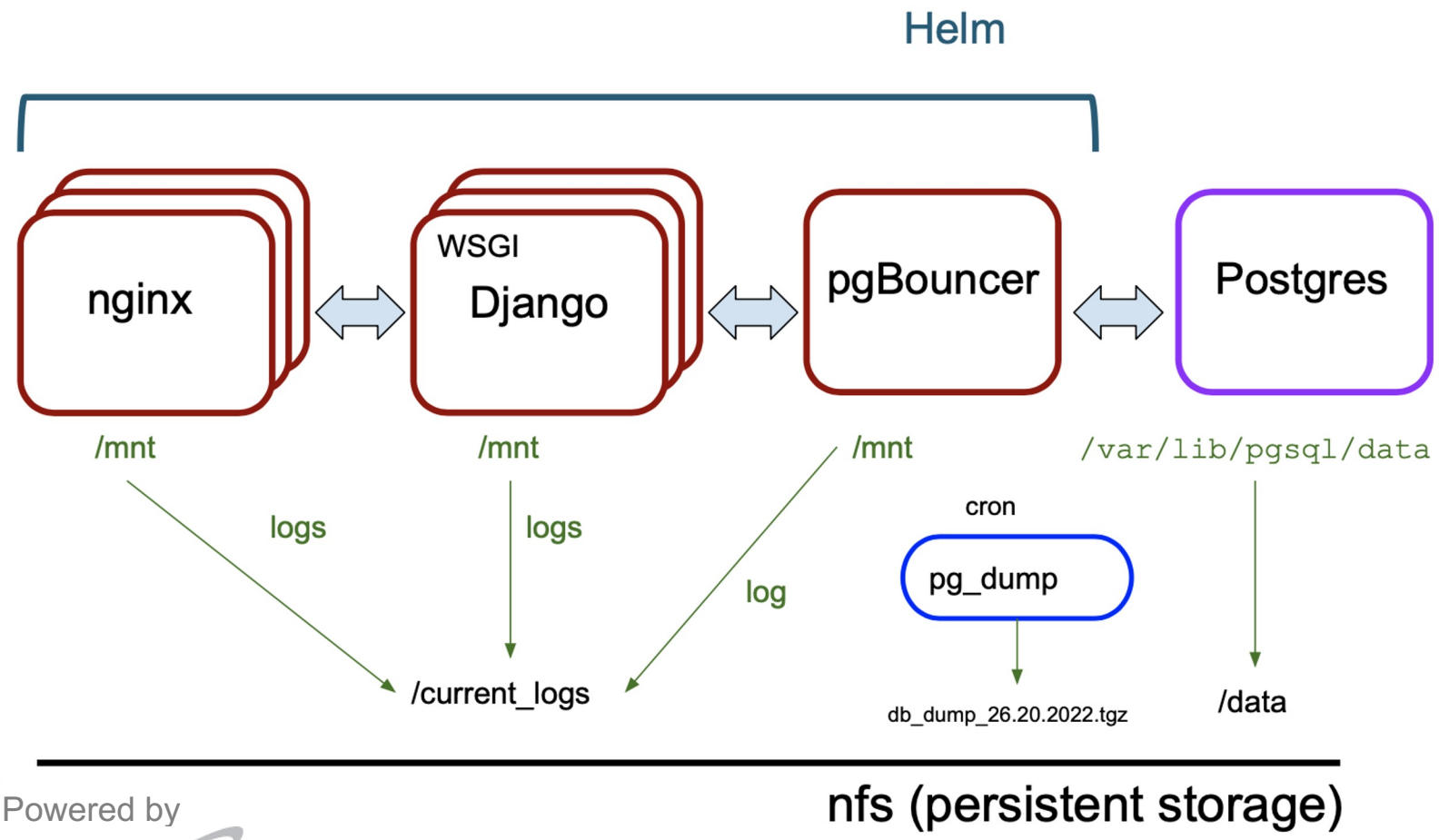




*Example query (simplified)

```
curl http://<host>/api/payloadiovs/?gtName=test_gt&iovNum=42
-> {type_1: url_1, type_2: url_2, …}
```

Deployment on OKD



- Automated deployment on OKD (Helm chart)
- Horizontally scalable
- Open Source only

Easily adoptable for various HEP experiments

Recent development – server-side caching

- Investigated different methods for server-side caching:
  - In webserver (nginx) layer, in REST-API layer, add dedicated caching layer (e.g. redis)
- Decided for **nginx**: easy to configure, high performance. But: no manual cache-invalidation
  - Find most frequently accessed endpoints from sPHENIX logs
  - Implemented selective in-memory caching for those
  - Set cache lifetime to 1 second
  - Repeating same request: response freq. >35kHz (1 nginx pod)

```
bash-4.2$ ./wrk -t12 -c400 -d30s http://npdb-dune.apps.usatlas.bnl.gov/api/cdb_rest/globalTags
Running 30s test @ http://npdb-dune.apps.usatlas.bnl.gov/api/cdb_rest/globalTags
  12 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    19.43ms   55.10ms   1.05s    95.53%
    Req/Sec     3.14k   445.11    21.66k    94.49%
  1122837 requests in 30.10s, 458.31MB read
Requests/sec:  37305.30
Transfer/sec:     15.23MB
```

Recent development – single container deployment

- Ruslan developed a single docker image that contains all server-side layers
  - Passed functionality tests, performance tests still awaiting
- Can be used for easy deployment within HPC
  - Nodes w/o internet can have access to the service
- Singularity can also run docker images
  - Still some technical issues w/ mounting file system
  - Should be fixed soon -> possible deployment at FNAL?

Test Deployment @ CERN

- Deployed **nopayloaddb** on VM @ lxplus (Apache & bare Django):
  http://vm-01.cern.ch:8000/api/cdb_rest/
- Installed **nopayloadclient & dunenpc** in shared location
  /eos/user/l/ligerlac/shared/releases
- Corresponding configuration file for **nopayloadclient**
  /eos/user/l/ligerlac/shared/config/for_lxplus.json
- Wrote **art** service: **ConditionsDataService**
  https://github.com/ligerlac/ifdh-art/tree/feature/hsf_condb_service

Ready to use on lxplus*

*After:
- set additional env variables
- replace **ifhd_art** by my fork

Backend Deployment @ BNL

- Fermilab vetos usage of Docker
- BNL is already operating a **nopayloaddb** instance (for sPHENIX)
  - Deployed on SDCC's OKD cluster
- Deployed an additional, DUNE-specific instance
  - Still need access granted from outside the network
- Got the ball rolling on this already in last year's autumn
- Personnel changes w/ unfortunate timing delayed progress
- Recently, the discussion picked up speed again
- Should be resolved soon

Integration into DUNE software stack

- **nopayloadclient** has been included into SciSoft
  - link to ticket: here
- Can now be used on **dunegpvm's (**CLI and c++ library)
- Next steps
  - Also include **dunenpc** in scisoft
  - Find a place for art service and merge it
  - Implement reading of payloads
    - Should the service do this or the respective calibration module?

Conclusion & Outlook

Conclusion

- Progress on HSF Reference Implementation
  - server-side caching, single-container deployment for HPC (potentially w/o docker)
- Progress on DUNE instance deployment @ BNL
  - Request has been approved

Outlook

- I will leave DUNE computing within this month
  - Will continue working on HSF Reference Implementation, but not on DUNE integration
- BNL is looking for a replacement to take over
  - Difficult to give a time scale at this point

Thank you all very much for the nice time in the Database group!

Backup

ConditionsDataService - Description


```
[ligerlac@lxplus790 LArSoftDev]$ lar --print-description ConditionsDataService
==========================================================================
    service : ConditionsDataService
        provider: user
        source  : /afs/cern.ch/user/l/ligerlac/LArSoftDev/srcs/...
        library : /afs/cern.ch/user/l/ligerlac/LArSoftDev/build_slf7.x86_64/...
    Allowed configuration
    ---------------------
        ## Any parameters prefaced with '#' are optional.
        ConditionsDataService: {
           ## global configuration parameter for all conditions data
           global_tag: <string>
           ## override url's for given condition types [[type1, url1], ...]
           # override_pairs: [
           #    [
           #       <string>,
           #       <string>
           #    ],
           #    ...
           # ]
        }
==========================================================================
```

ConditionsDataService – Example Config

```
ConditionsDataService: {
    global_tag: "test_gt"
    override_dict: [
        ["space_charge_effect", "my_local_file.root"]
    ]
}
```

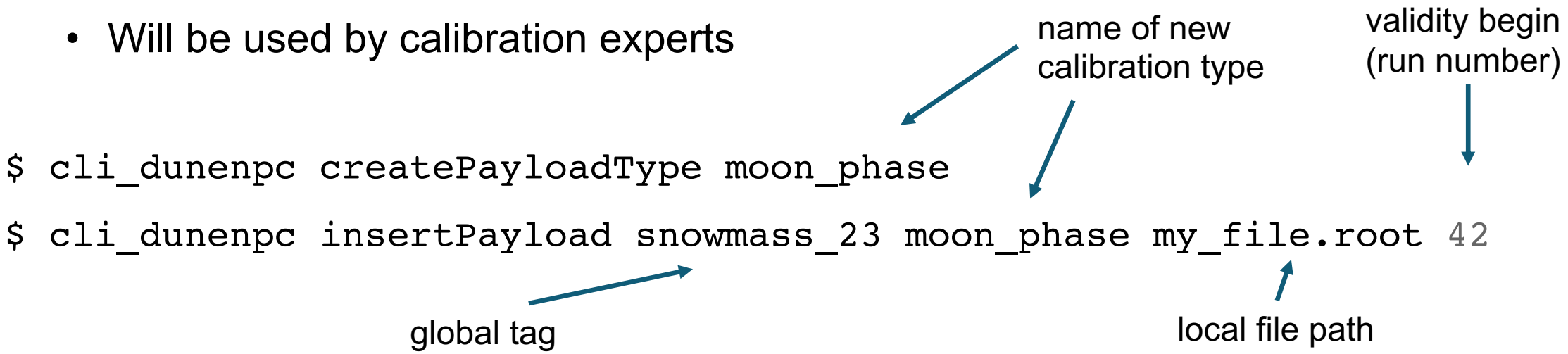
Single configuration parameter for all conditions data

Optional: overriding

- Use ‘my_local_file.csv’ for electron lifetime, regardless of DB content
- All other contions data according to global tag ’test_gt’
  - Isolate impact of new calibrations before inserting into DB

Integration into Software Stack

- **dunenpc** comes with CLI for managing conditions data
  - Will be used by calibration experts



- LArSoft (art) service to access conditions data within a job
- Not aware of any ‘prompt’ processing use case
  - Read-only service should be sufficient

ConditionsDataService – Implementation

```
#include <dunenpc/dunenpc.hpp>

class ConditionsData  {

private:
  dunenpc::DuneClient client_;

public:

  ConditionsData(Config const& config) {
    client_.setGlobalTag(config.global_tag());
    for (const auto& pair : override_pairs) {
      client_.override(pair.type, pair.url);
    }
  }

  std::string getUrl(const std::string& type, int run_number) const {
    return client_.getUrl(type, run_number);
  }
};
```

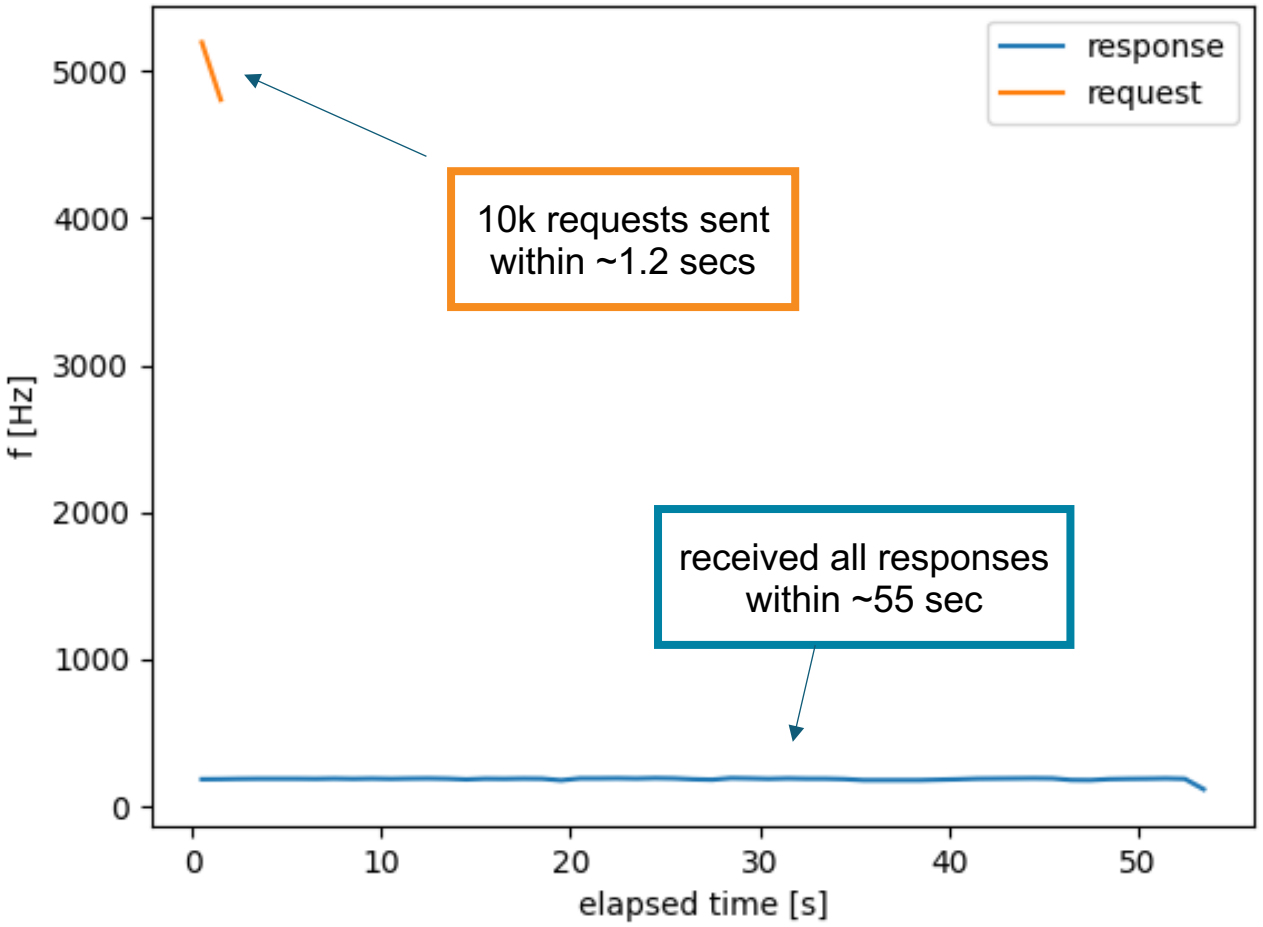
DUNE-specific version of **nopayloadclient** (stand-alone)

Does all communication w/ DB, caching, handling payloads, low-level configuration

Translate FHiCL parameter set into client configuration

Simple wrapper for getUrl()

Performance Testing – High Frequency



- Simulate offline reco use case
  - Many jobs launched at same time
- Cooperative multithreading (**asynchio**)
  - Send requests firsts
  - Process responses later
- Allows very high peak request frequency
- Server-side queuing of requests works