DAMNIT architecture

Thomas Kluyver, European XFEL

26th June 2025

Data And Metadata iNspection Interactive Thing

Table View

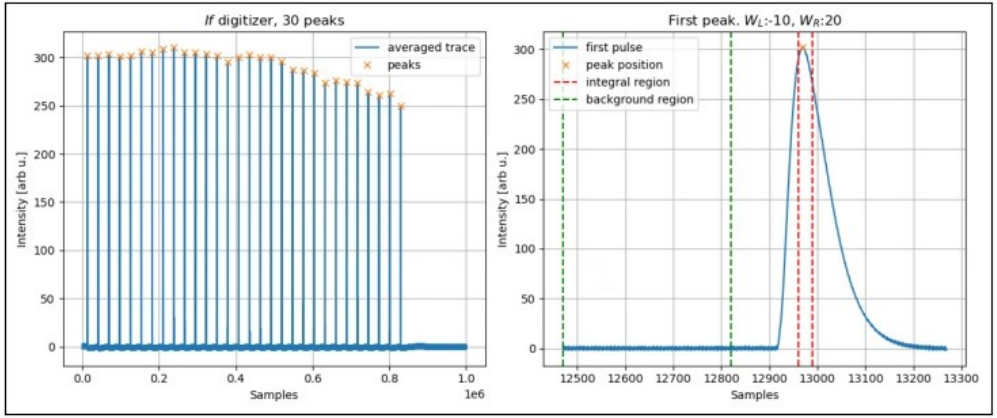
Run table | Context file

Plot | Time comment | Variables: 2 tags | Filters (0)

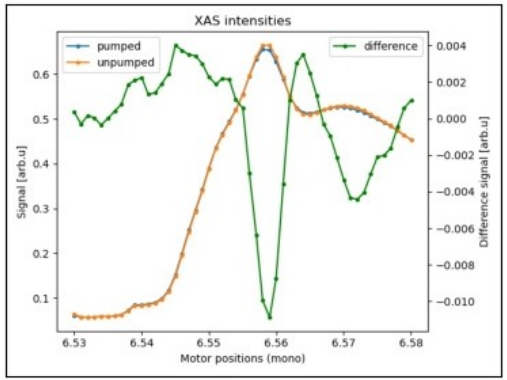
	Timestamp	mono_1_energy	undu_energy	BAM delay	PPODL	Pump pattern	Scan type	I0	If	I0 peaks: fig	If peaks: fig	Absorption	XAS pp	XAS scan
238	19:02:28 16/06/2025	**6.5400**	**6.4729**	**-143.1926**	-157	train	mono							
239	19:13:18 16/06/2025	**6.5400**	**6.4729**	**-149.6219**	-157	train	mono							
240	19:24:08 16/06/2025	**6.5400**	**6.4729**	**-157.5511**	-157	train	mono							
241	19:56:09 16/06/2025	**6.5550**	**6.4882**	**-176.6944**	-157	pulse	mono							
242	20:07:32 16/06/2025	**6.5590**	6.4922	**-183.4412**	-157	pulse	INCOUPLIN...							
243	20:15:58 16/06/2025	**6.5590**	6.4922	**-187.6522**	-157	pulse	INCOUPLIN...							
244	20:28:08 16/06/2025	**6.5590**	6.4922	**-190.0079**	-157	pulse	INCOUPLIN...							
245	20:32:34 16/06/2025	**6.5590**	6.4922	**-190.5333**	-157	pulse	INCOUPLIN...							
246	20:38:06 16/06/2025	**6.5550**	**6.4881**	**-195.1545**	-157	pulse	mono							
247	20:48:32 16/06/2025	**6.5550**	**6.4881**	**-196.2476**	-157	pulse	mono							
248	21:01:25 16/06/2025	**6.5600**	6.4936	**-199.1470**	**-162.2367**	pulse	ppl_odl							
249														
250														
251														
252														
253														
254														
255														
256														
257														
258														
259														

Doub

If peaks: fig (run 257)



XAS scan (run 257)



Display hover annotations

dcd)

DAMNIT!

FXE p7925 - Raj Pandya

Probing ultrafast electron-polaron dynamics in lithium-ion battery cathodes

Web interface

Table

	Run	undu_energy	BAM delay	PPODL	Pump pattern	Scan type	I0	If	Laser intensity	I0 peaks: fig	If peaks: fig	Absorption	XAS pp	XAS scan	JF burst mode	JF 1M preview	JF ROI prev
134	136	6.48	-69.1	-157	pulse	mono									0		
135	137	6.48	-77.4	-157	pulse	mono									0		
136	138	6.49	-88.9	-157	pulse	INCOUPLING_Y									0		
137	139	6.49	-95.2	-157	pulse	INCOUPLING_X									0		
138	140	6.49	-99.9	-157	pulse	INCOUPLING_Y									0		
139	141	6.49	-107.1	-157	pulse	INCOUPLING_X									0		
140	142	6.49	-117.1	-157	pulse	INCOUPLING_Y									0		
141	143	6.49	-127.5	-157	pulse	INCOUPLING_Y									0		
142	144	6.49	-134.4	-157	pulse	INCOUPLING_X									0		
143	145	6.48	-142.5	-160	pulse	mono									0		
144	146	6.48	-150.0	-157	pulse	mono									0		
145	147	6.51	-158.1	-157	pulse										0		
146	148	6.51	-159.1	-157	pulse										0		
147	149	6.51	-159.3	-157	pulse										0		
148	150	6.46	-166.9	-157	pulse	mono									0		
149	151	6.47	-1	-157	train	mono									0		
150	152	6.47	-173.3	-157	train	mono									0		
151	153	6.47	-177.6	-157	train	mono									0		
152	154	6.47	-180.0	-157	train	mono									0		
153	155	6.47	-181.5	-157	train	mono									0		
154	156	6.47	-179.9	-157	train	mono									0		
155	157	6.47	-178.7	-157	train	mono									0		
156	158	6.47	-180.1	-157	train	mono									0		

Plot: data
BAM delay

Data And Metadata iNspection Interactive Thing

Table View

Run table | Context file

Context file

```
66  @Variable(title="Trains", tags=['metadata'])
67  def n_trains(run):
68      return len(run.train_ids)
69
70  @Variable(title="Run length", tags=['metadata'])
71  def run_length(run):
72      ts = run.train_timestamps()
73      delta = ts[-1] - ts[0]
74      delta_s = int(delta / np.timedelta64(1, 's'))
75      return str(timedelta(seconds=delta_s))
76
77  def run_size_tb(path):
78      run_size_bytes = sum(f.stat().st_size for f in path.rglob('*'))
79      return run_size_bytes / 1e12
80
81  @Variable(title="Raw size [TB]", tags=['metadata'])
82  def raw_size(run, proposal_path: "meta#proposal_path", run_no: "meta#run_number"):
83      run_path = proposal_path / "raw" / f"r{run_no:04}"
84      return run_size_tb(run_path)
85
86  @Variable(title="Proc size [TB]", data="proc", tags=['metadata'])
87  def proc_size(run, proposal_path: "meta#proposal_path", run_no: "meta#run_number"):
88      run_path = proposal_path / "proc" / f"r{run_no:04}"
89      return run_size_tb(run_path)
90
91  # Values from myMDC
92
93  @Variable(title="Run type", tags=['metadata'])
94  def get_run_type(run, run_type: "mymdc#run_type"):
```

Save

Validate

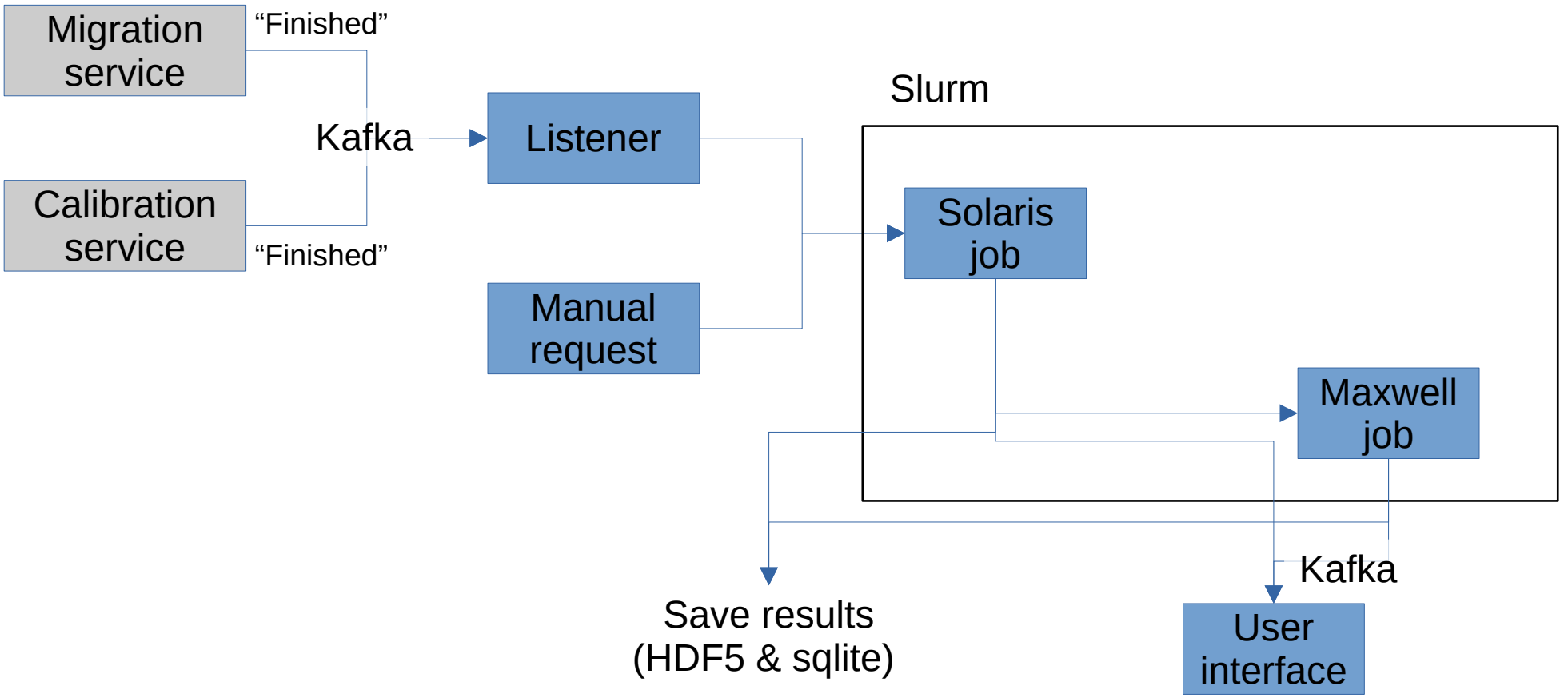
Reload from disk

Valid context.

Double-click on a cell to inspect results.

Getting updates (f40ec1ed0353a430f9d94d2c552dfaee5a401dcd)

Running the code



Directory structure

```
<proposal>/usr/Shared/amore

├── context.py
├── extracted_data
│   ├── p001234_r1.h5
│   ├── p001234_r2.h5
│   ...
├── process_logs
│   ├── r1-p001234.out
│   ├── r2-p001234.out
│   ...
├── runs.sqlite
```

API

```
from damnit import Damnit
db = Damnit(7925)
data = db[100, "i0"].read()
```

https://damnit.readthedocs.io/en/latest/api/

Automate initial analysis steps, retrieve data for further analysis

Services

- Current / classic model:
  - Per-proposal listener
- New model:
  - Web API server (with GPFS access)
  - Web UI server (accessible from outside)
  - Central listener for all proposals
  - Central DB writer?

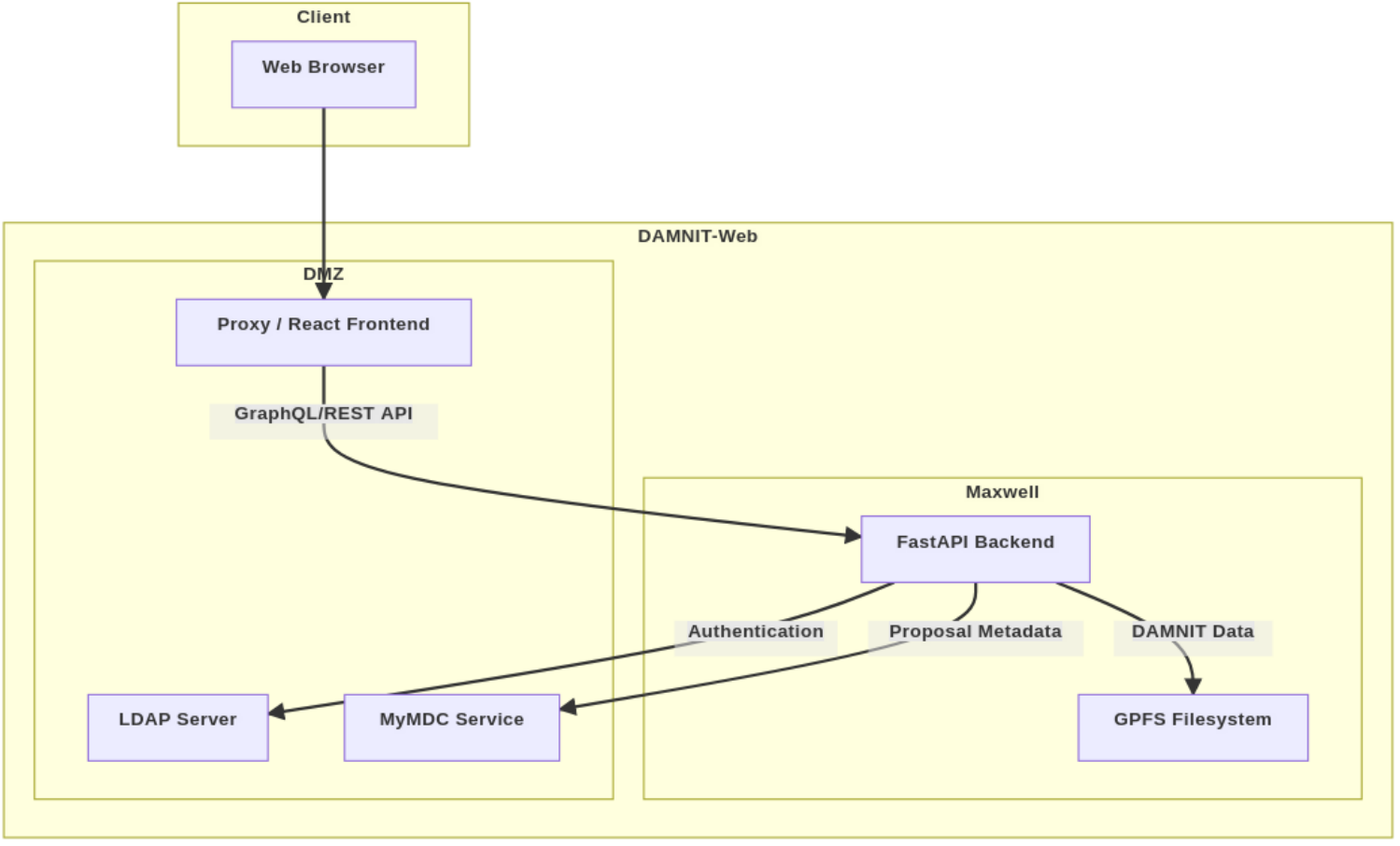
Web servers

- UI server
  - Serve JS files (compiled from Typescript)
  - Proxy to API server
- API server
  - GraphQL
  - Python & FastAPI
  - Mutual TLS (mTLS) allows requests only via UI server

Possible service accounts

- xdamnear (listener)
  - Broad permissions, no user code
  - Add to all upex Slurm reservations
- xdamnrun
  - Executes user code
  - Minimal base permissions
  - Sandboxing adds proposal access as needed
- xdamnweb (web API server)
  - Access to every proposal’s DAMNIT directory

Web server interactions



Web authorisation

- Sign in via keycloak
- Look up groups by username in LDAP
  - e.g. `60006736-dmgt, fxedata`
- Allow access only to corresponding proposals
- Future option: check user’s permissions in myMdC

Sandboxing

- Goal: restrict context file code to 1 proposal
- `xdamnear` (full access) starts process as `xdamnrun` (no access)
- Process adds itself to proposal group
- Sets ‘no new privileges’ flag
- Executes context file

Sandboxing detail

```
$ proposal-sandbox 7925 -- some-command --foo
```

- Run `some-command` with access to proposal 7925
- Uses setgroups() – needs CAP_SETGID capability
- seteuid() & setegid() to switch user
- prctl() to set no_new_privs flag

https://git.xfel.eu/dataAnalysis/proposal-sandbox/-/merge_requests/1

Sandboxing & Slurm

- Context file code can (& does) submit Slurm jobs
- Jobs would have no proposal access by default
  - Fail safe – but still fails
- Need to re-enter the same sandbox in Slurm job
  - Secure way to know proposal submitter belongs to?
  - MUNGE token?
  - Use Slurm task prolog?
  - Tweak sandboxing to change primary group? (passes through Slurm)

Online processing?

- Faster results in DAMNIT are desirable
- Online processing is interesting
- Obstacles
  - No Slurm on online cluster
  - Not safe to modify SQLite DB in synced `usr`

DB writer coordination

- Summary data in SQLite DB for each proposal
  - No need for a DB server
  - Data all together in proposal folder
- But write contention can be an issue
  - We limit concurrent jobs to mitigate this
- Centralised writer service?
  - Could batch writes when busy

HDF5 writer coordination

- Larger results stored in HDF5 file for each run
  - Concurrent writes less likely
  - Avoid unnecessary sending/copying of data
- Concurrent writes can corrupt files
  - HDF5 uses flock(), not effective across nodes
- Prevent concurrent jobs for 1 run?
- Separate output file for each job?
  - Recombine files later on when things are quiet?