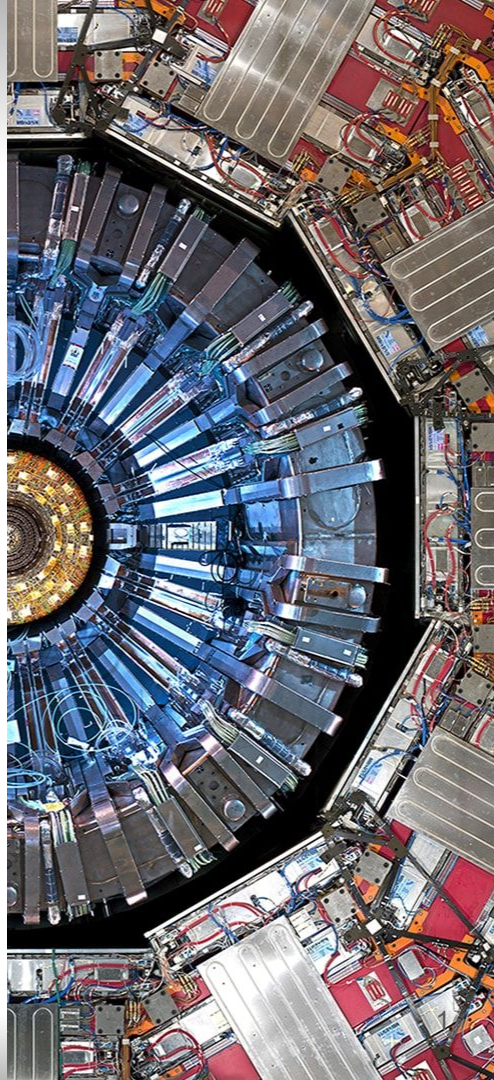
Verifying Replica Integrity Before Redirections

Rahul Chauhan
XRootD and FTS Workshop
September 14 - 18, 2026
CNRS Délégation Rhône Auvergne



The CMS XRootD Federation (aka AAA)

CMS AAA is a worldwide data access layer

- **xrootd** serves the bytes
- **cmsd** tells clients where those should come from

Enables remote read of disk resident data without local downloads

- LFN uniquely identifies the file around the globe
- Redirectors recursively query possible location
- Client is redirected to a valid location

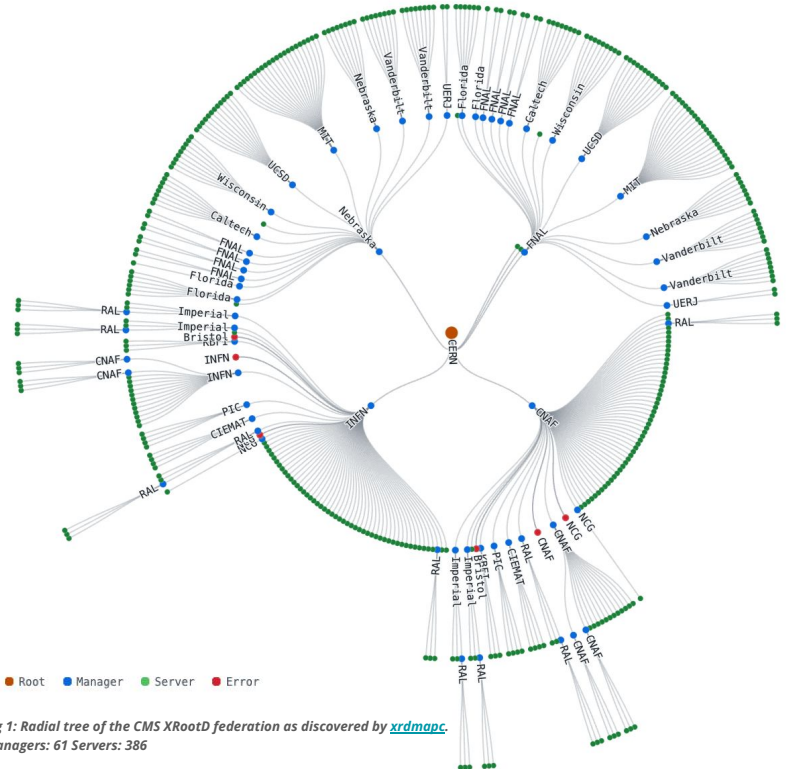


Fig 1: Radial tree of the CMS XRootD federation as discovered by xrdmapc.
Managers: 61 Servers: 386

CMS Use of AAA

Any data, Any time, Anywhere

- Also allows users to access uncatalogued data
- root://cms-xrd-global.cern.ch/{lfn}

Fallback file open

- CMSSW application initially use site configuration to attempt local open
- If local open fails, there is a “fallback” to appropriate AAA redirector
- Although CRAB & Production jobs are sent to data locations: *Rucio catalogue*
    - various situations lead to jobs not running at the site: *files temporarily not available at the site*

Serving premix pileup to simulation production jobs

- Uses AAA, though in practice almost all files are either at CERN or FNAL

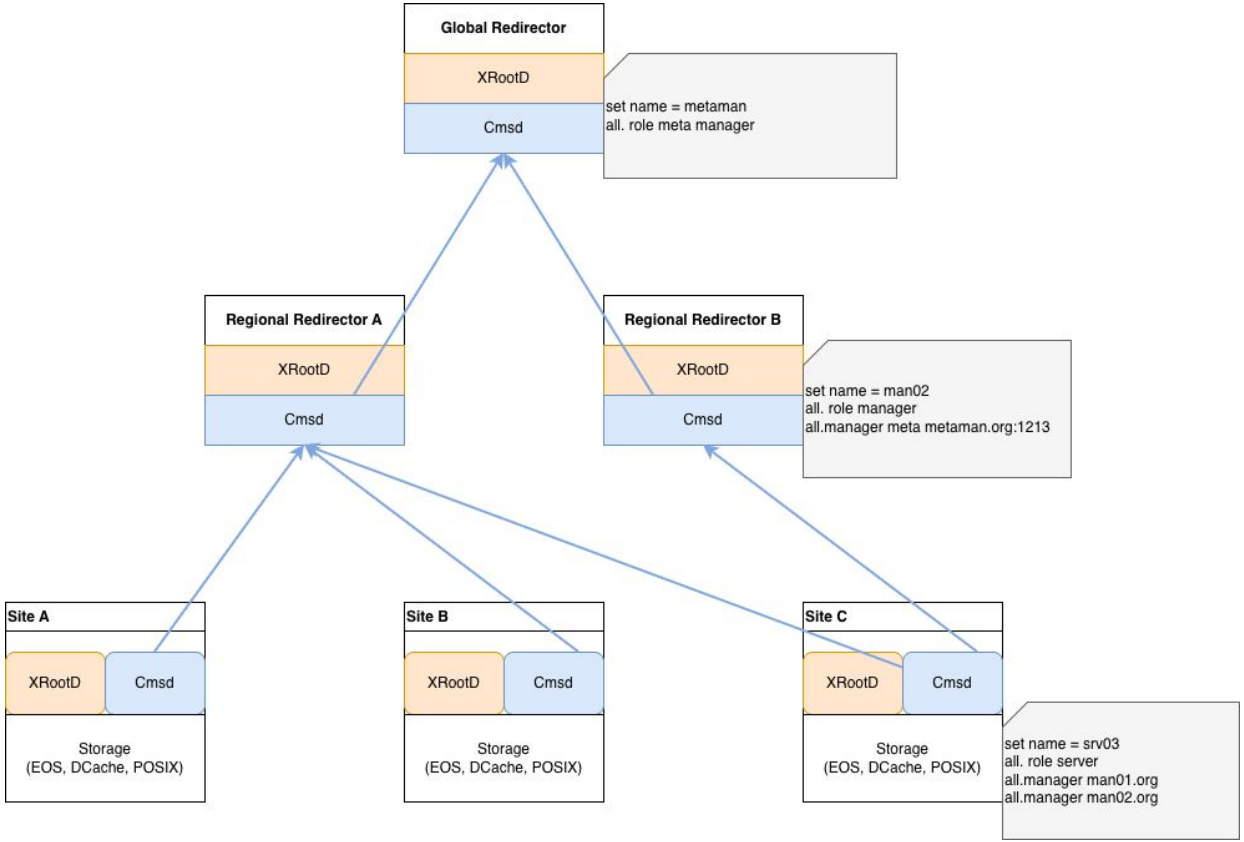
Federation Architecture

The federation consists of
- Metamanager nodes
- Manager nodes
- Server nodes

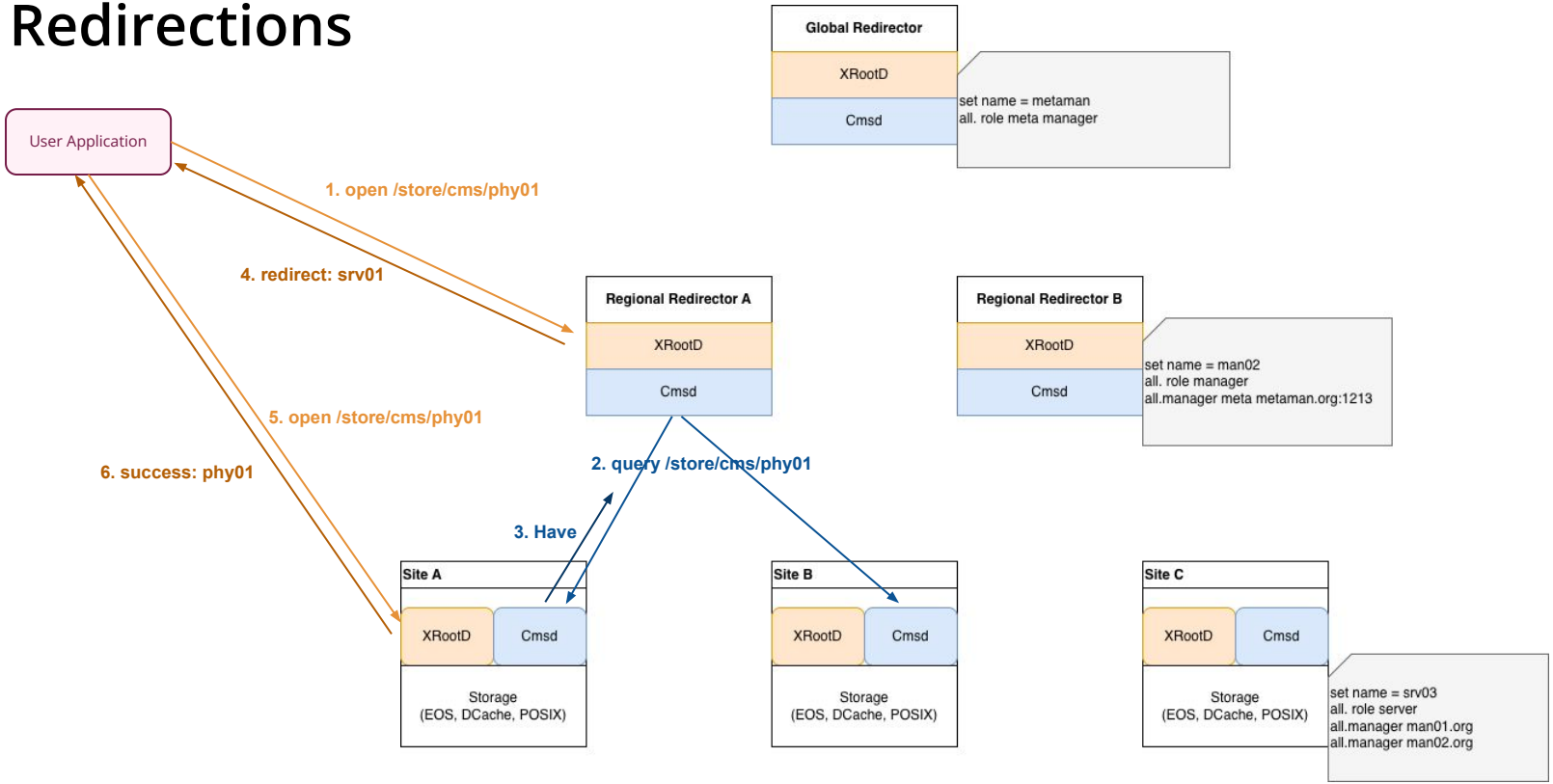
Many-to-many relationship between managers and server
- For redundancy

Sites can join in multiple patterns:
- Proxy (XrdPss)
- Federation Hosts

[Joining a dCache-based SE to the Xrootd service.]



XRootD Redirections



The Problem: Remote Read Failures

Redirector selection is based on limited inputs (load, proximity estimates): no feedback from the actual open outcome*

Pain points observed:

- Clients redirected to stale or corrupt replicas
    - open fails, duplicated retry overhead
- No per-server health signal feeds back into routing
- Clients sent to distant sites when local replicas exist

*clients can signal on the next request that they have tried a particular site and that it should be ignored with **tried** request parameter
For CMS Production jobs some knowledge is build into the XrdAdapter, that allows it to deal with data access failures much more gracefully

XRootD plugin layers in a redirecting cluster

Ofs plugin does different function calls based on node roles

Redirector: locate file, return redirect

- ofs uses cmsd client to locate file

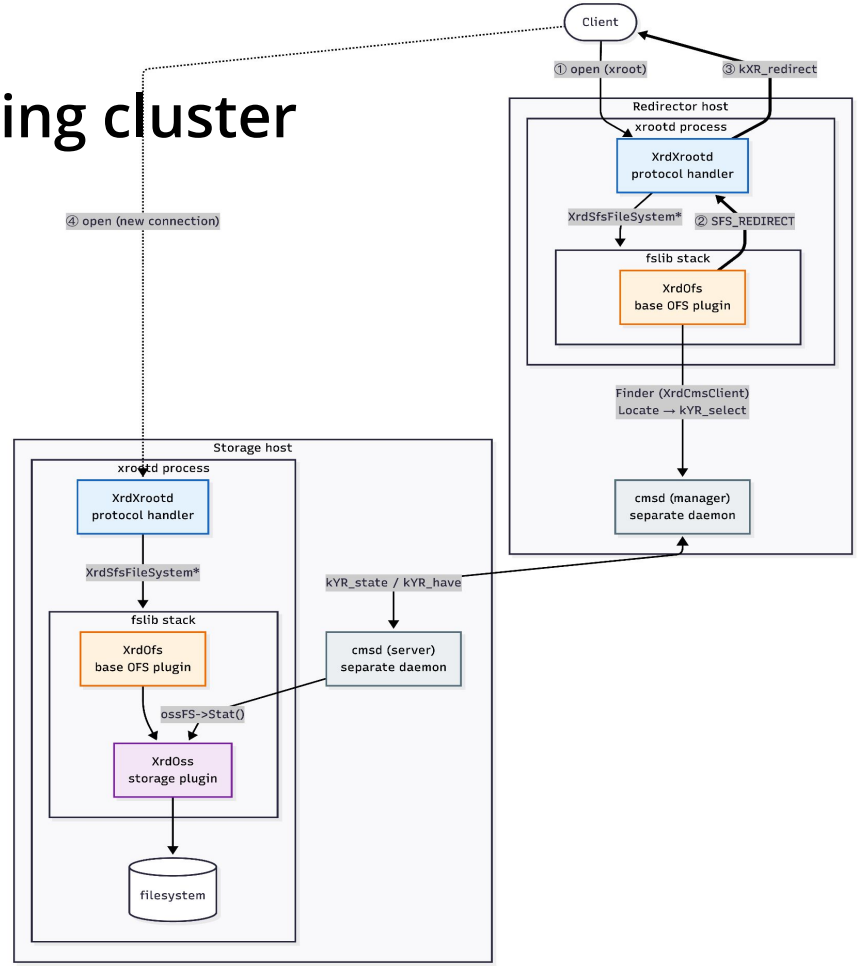
Storage: serve file via OSS - filesystem

- ofs uses oss to serve bytes

cmsd locate flow is often different from xrootd access flow

XrdOfs is the base OFS plugin

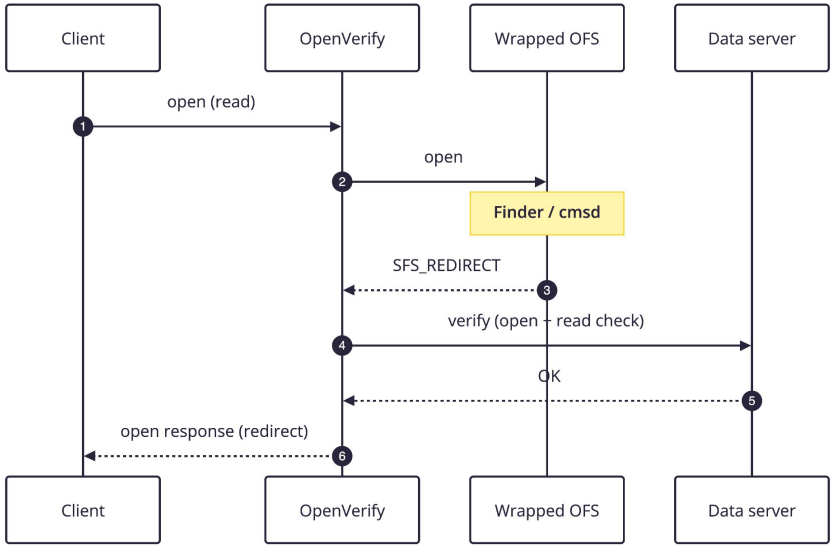
- fslib plugins are stackable: chained with **++**



XrdOfsOpenVerify Plugin

Verify the redirect target, returned by wrapped OFS after cmsd's selection

- Use XrdCl to perform a small *open+read* against the redirect target before returning *SFS_REDIRECT*
- On failure: injects the bad host into the *tried=* parameter, forcing the redirector to pick a different server
- Bypass for non-read operations



Caching

Cache avoids repeating XrdCl work for the same file on the same redirect target

Cache key: *<host>:<port>//<path>* stored in a path-segmented trie

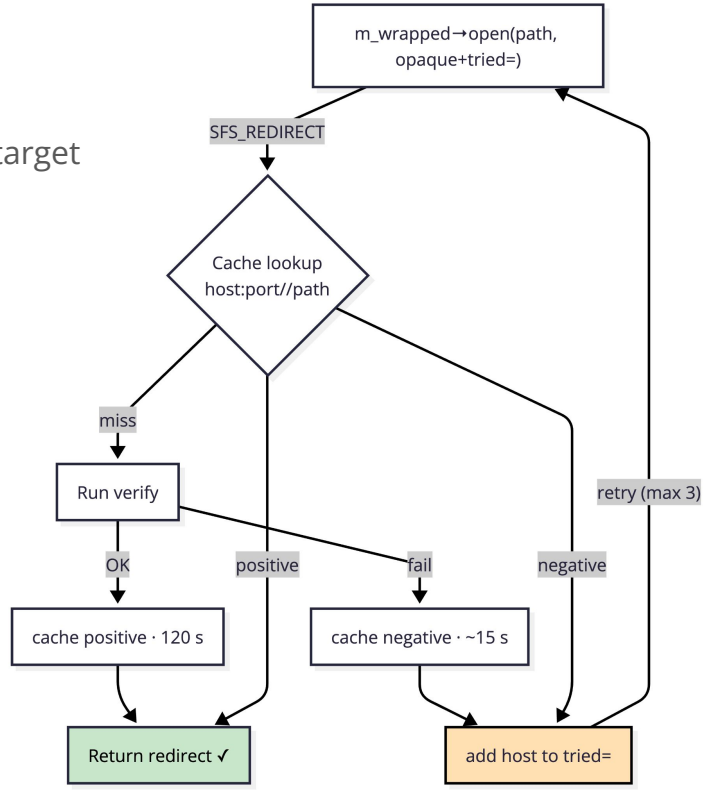
Positive entries (server has the file)

- TTL 120s: no re-verification needed on subsequent opens

Negative entries (open/read failed for the host+file)

- TTL (15 s ± 20% jitter) : allows recovery as replicas get fixed
- Jitter prevents negative entries from expiring together

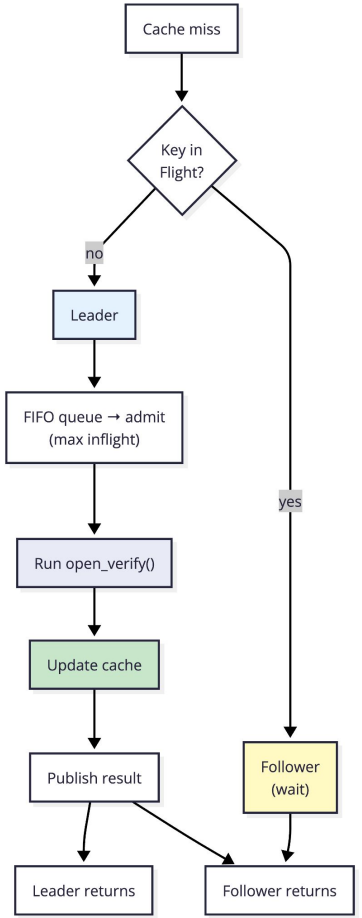
Miss: Run *open_verify()*



Single-Flight and Concurrency Control

Under high concurrency, many clients may open the same file simultaneously against a cold cache key triggering redundant verifications against the same server

- Only one leader executes *open_verify()* per *(file, host, port)*
  - all followers wait and reuse the result
- Global cap on concurrent verifies (separate keys can still run in parallel, up to the limit):
  - *XRD_OPENVERIFY_MAX_INFLIGHT* - default 32
  - *XRD_OPENVERIFY_MAX_WAITERS* - default 128
  - *XRD_OPENVERIFY_QUEUE_TIMEOUT_MS* - default 5000



Host Reliability - Scoring and Quarantine

We maintain an EWMA health score (higher = worse)

- Updated after every verify
  - Success: score decays towards 0
  - Failure: score increases weighted by error class
- Hysteresis band to prevent oscillation between states
  - Quarantine when *ewma >= 0.6*
  - Recover when *ewma <= 0.4*
  - Min attempts before quarantine is enforced = 20
- Probe cooldown
  - one probe request allowed for quarantined hosts every 60s + 20% jitter
  - Failed probe resets the window

Parameter	Value	Meaning
Min attempts	20	Before quarantine/recover can flip state
Quarantine	EWMA ≥ 0.6	Host marked unhealthy
Recover	EWMA ≤ 0.4	Host marked healthy again
Probe cooldown	60 s ± 20%	Gap between probe attempts
α fail / α success	0.05 / 0.10	Failures linger Successes recover faster

Fig 2: EWMA scoring parameters and quarantine thresholds

```
OnSuccess: ewma_health *= (1 - α_success)
OnFailure: ewma_health = α_fail × penalty + (1 - α_fail) × ewma_health
      penalty ∈ {1.0, 0.7, 0.6} by error class (network / protocol / xrootd)
```

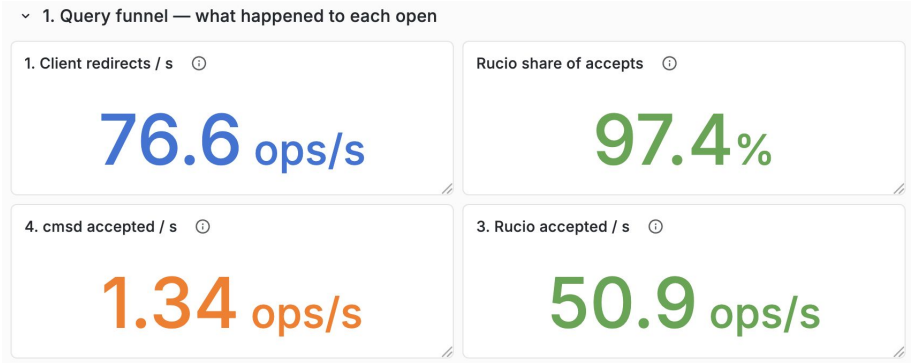
Quarantine burst protection

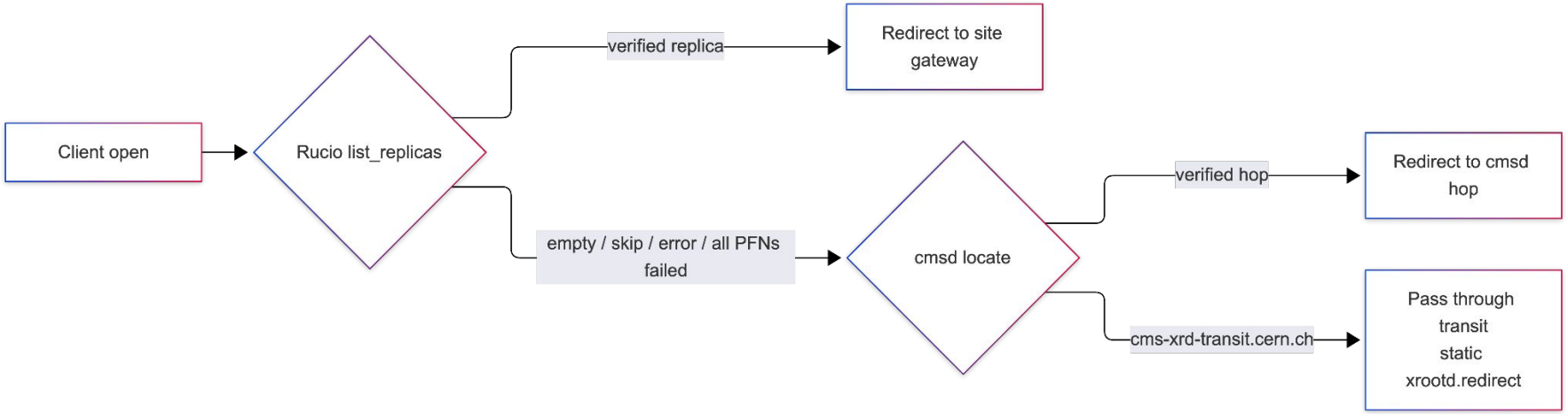
Rucio First Locate

Perform a list_replica call against rucio

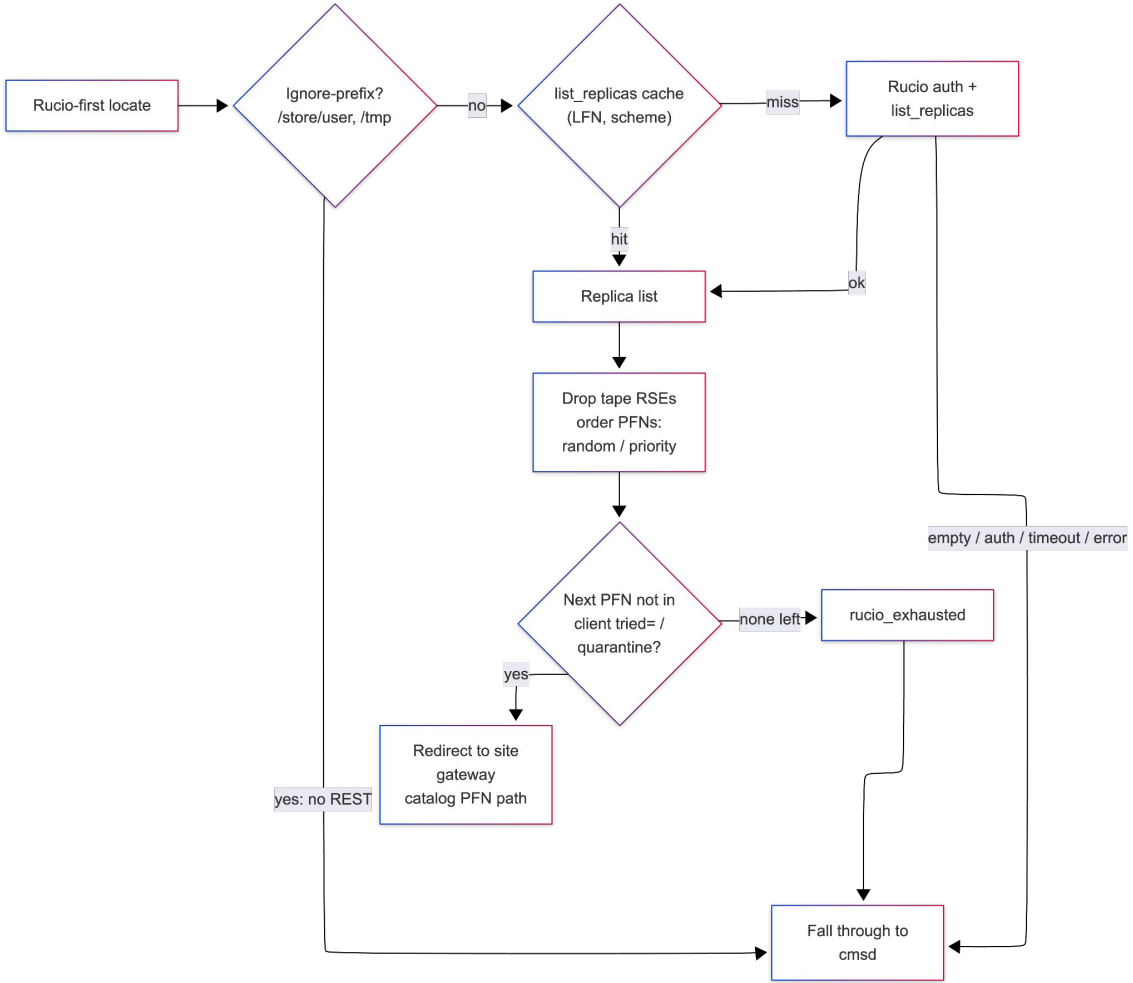
- use –protocol <http|root> as on the wire

Fallback to cmsd





Rucio redirection flowchart



Rucio redirect policies

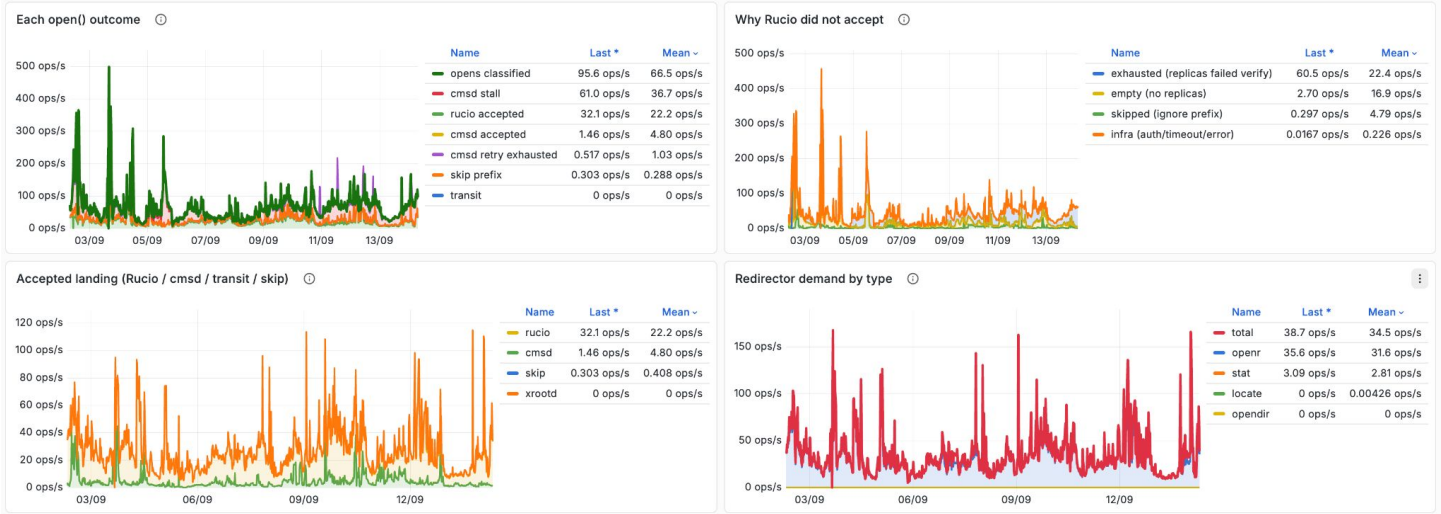
- **PFN, not LFN.** Rucio returns a site prefix (`/eos/cms/store/...`). Verify and redirect use that path.
- **Full-URL redirects** (`SFS_REDIRECT` port `-1`) so HTTP Location is followable.
- `davs://` → `https://` (and `dav` → `http`). Clients will not follow a `davs` Location.
- **Drop tape.** PFN `type=TAPE` or RSE `*_Tape` . Disk and Tape often share a gateway host with different paths.
- **Try-order** is pluggable: `random` (default), `priority` , `geoip` reserved.

Observability

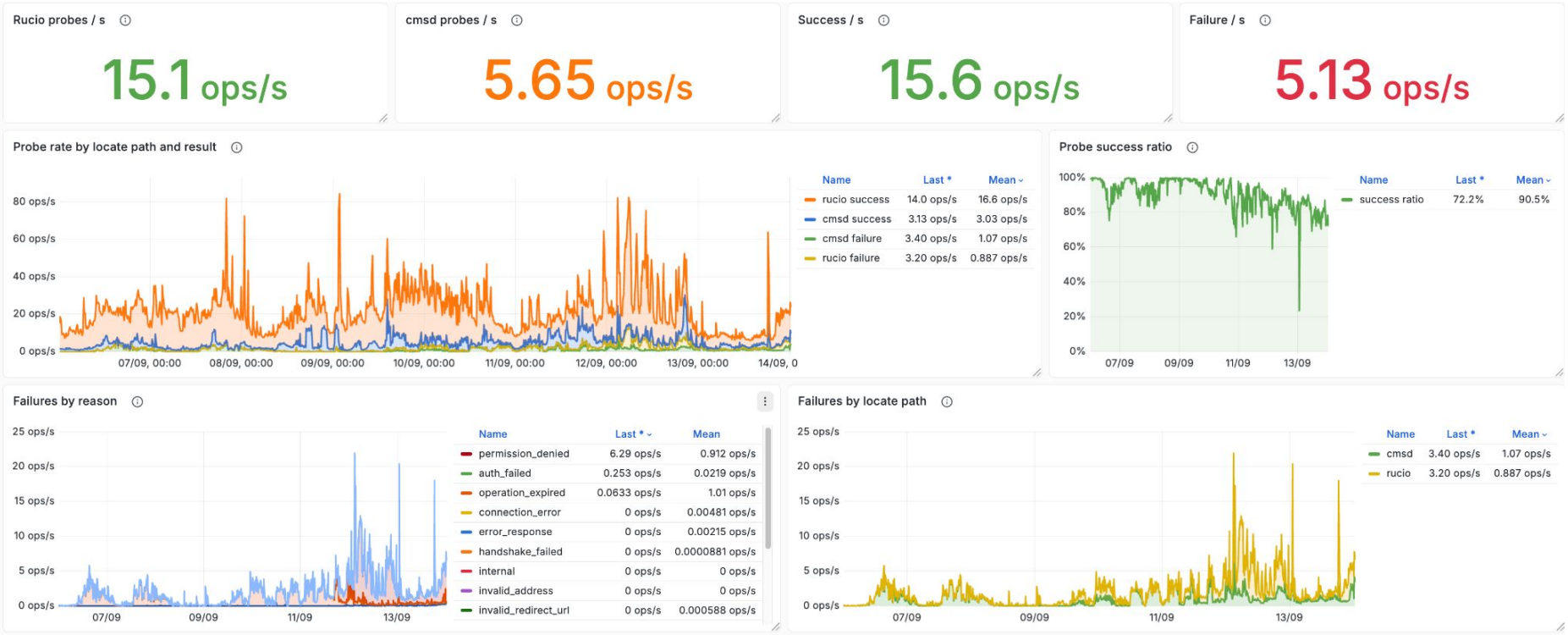
A textfile is generated in the prometheus metric format

Scraped by prometheus via the node exporter

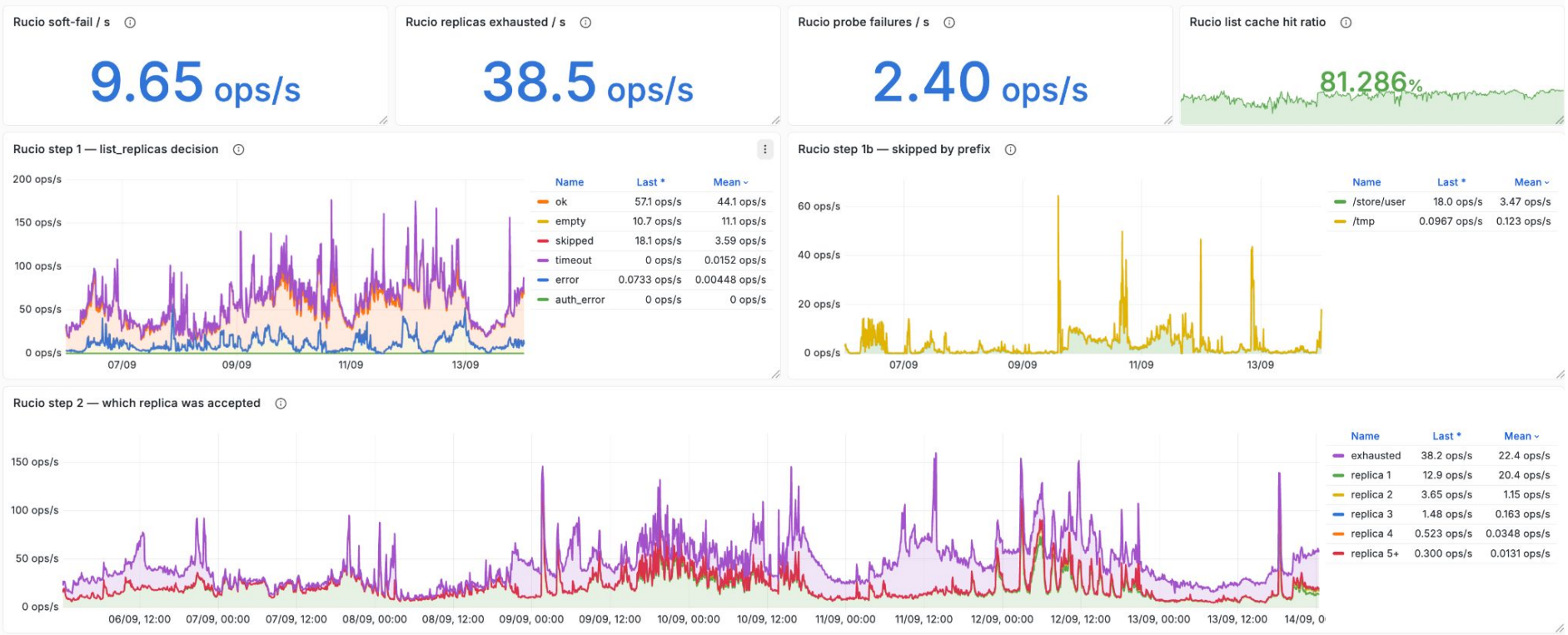
Observe mode *XRD_OPENVERIFY_OBSERVE=1*: records metrics but does not modify redirect behaviour



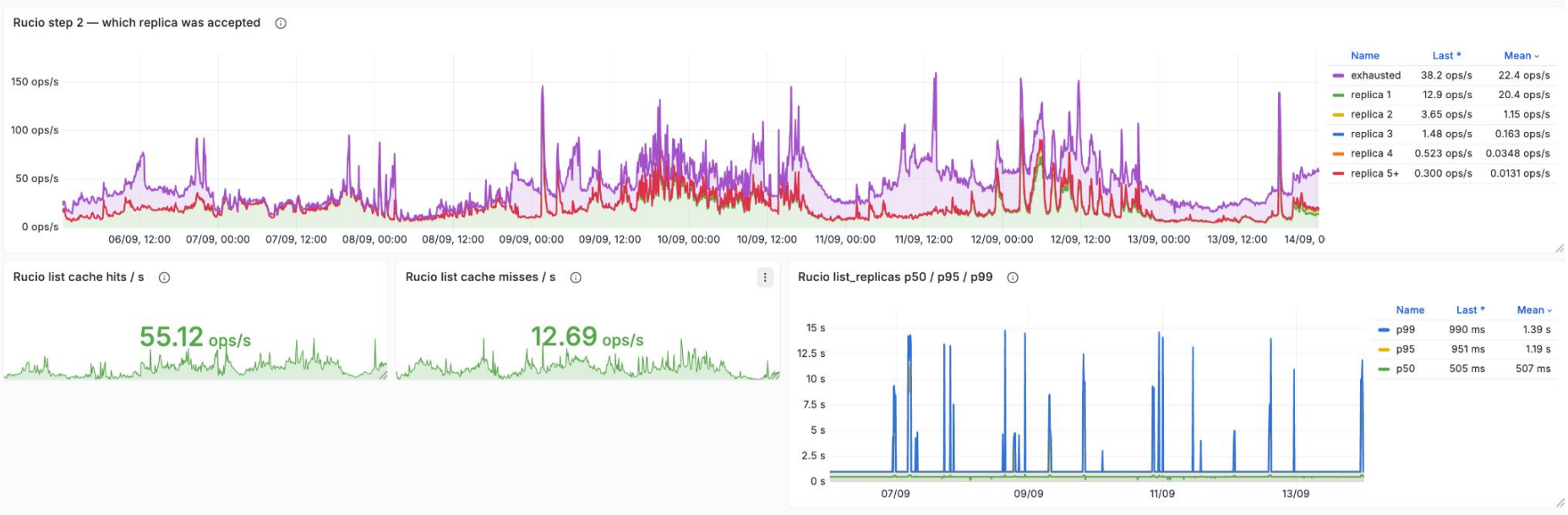
Open-verify probes monitoring



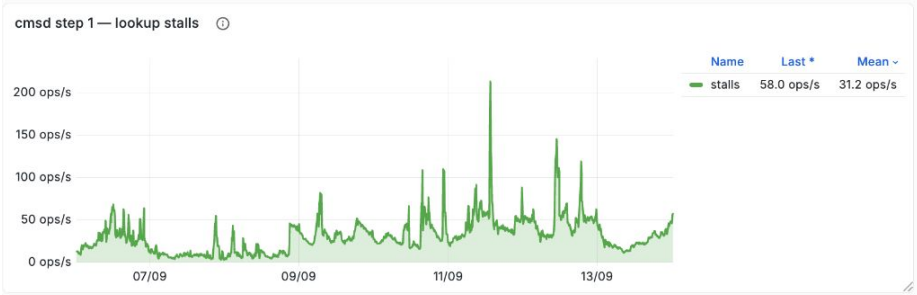
Zooming into Rucio redirected probes

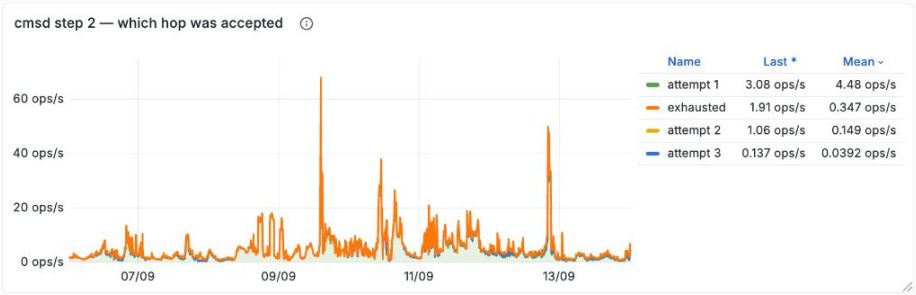


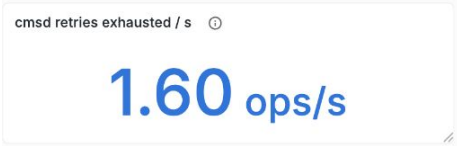
Which replica is accepted

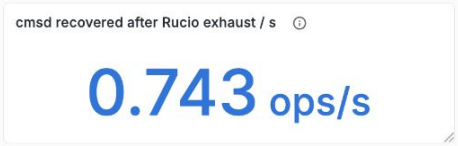


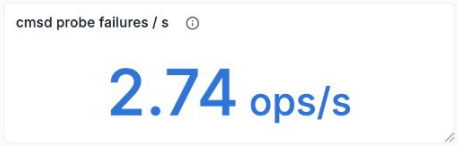
Zooming in CMSD redirected probes

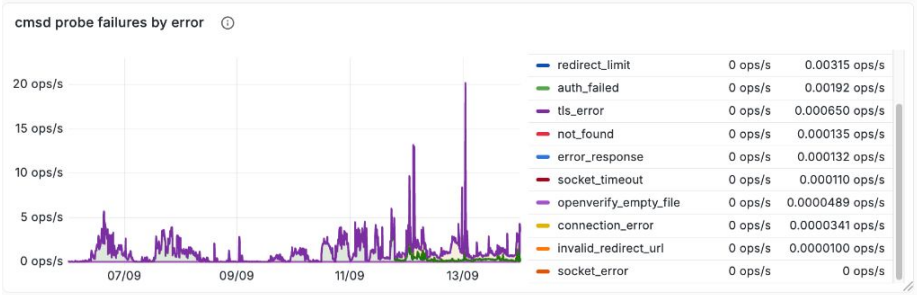


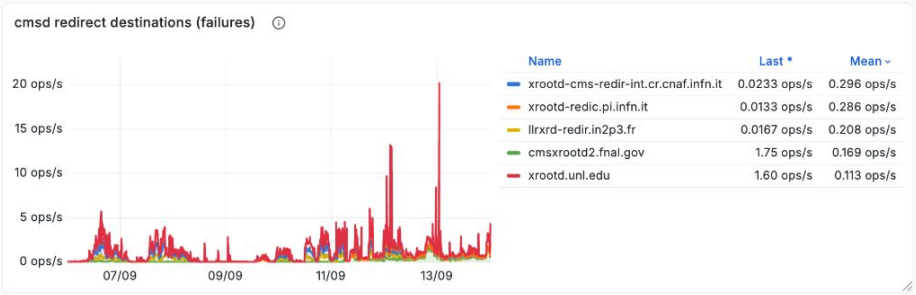








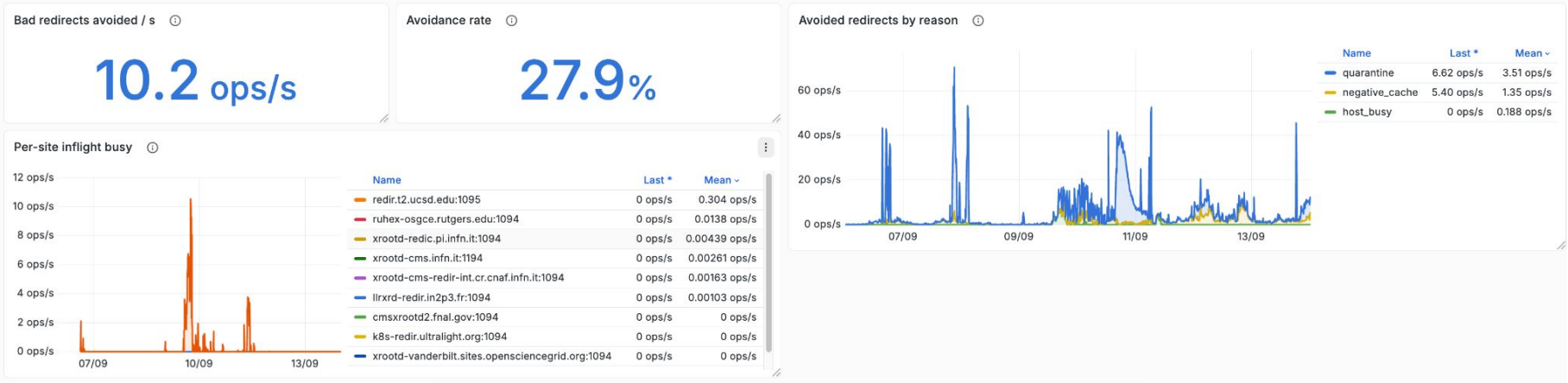




Summary

Still evaluating viability

- Fixing bugs as we go



Thank you!