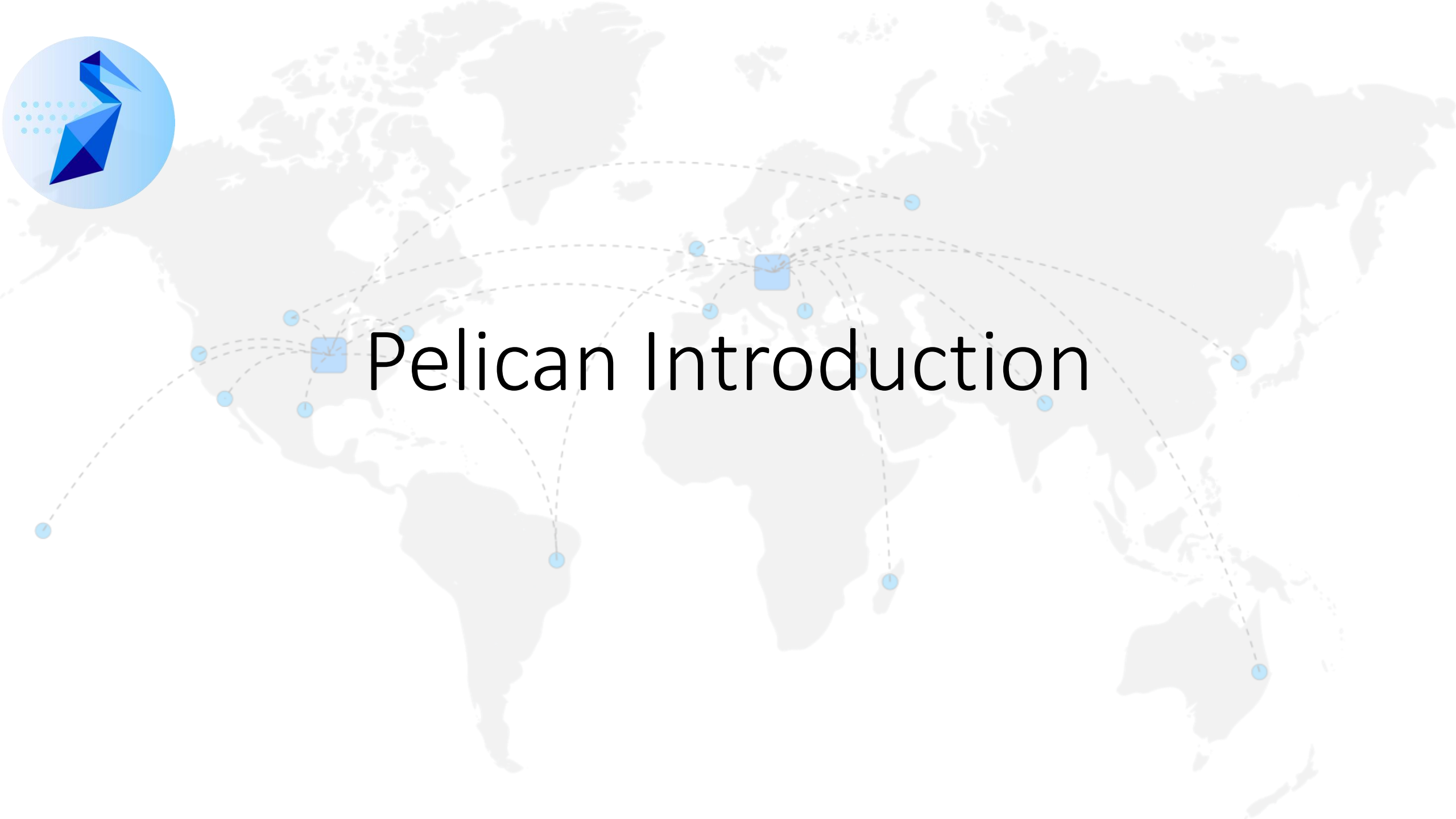




Pelican Introduction





Introducing the OSDF



The OSDF is a federated platform for delivering datasets from repositories to compute in an effective, scalable manner.



OSDF Integrates Independent Repositories into a common fabric

★ AWS
Open Data

★ NCAR

★ LIGO

★ OSPool

● jupyter

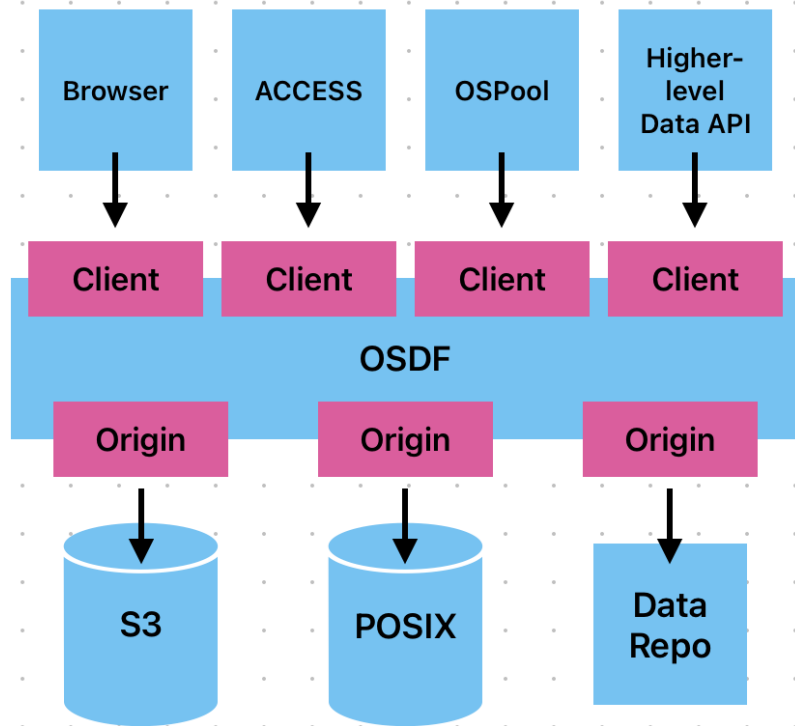
DeltaAI
I | NCSA

★ = existing integration

- About a dozen repositories integrated already, more on the way.
- Working to grow:
 - clients,
 - integrated resources, and
 - environments.



OSDF Architecture - Vision



Long-term vision:

We want OSDF to be an “all-science” CDN.

Requires:

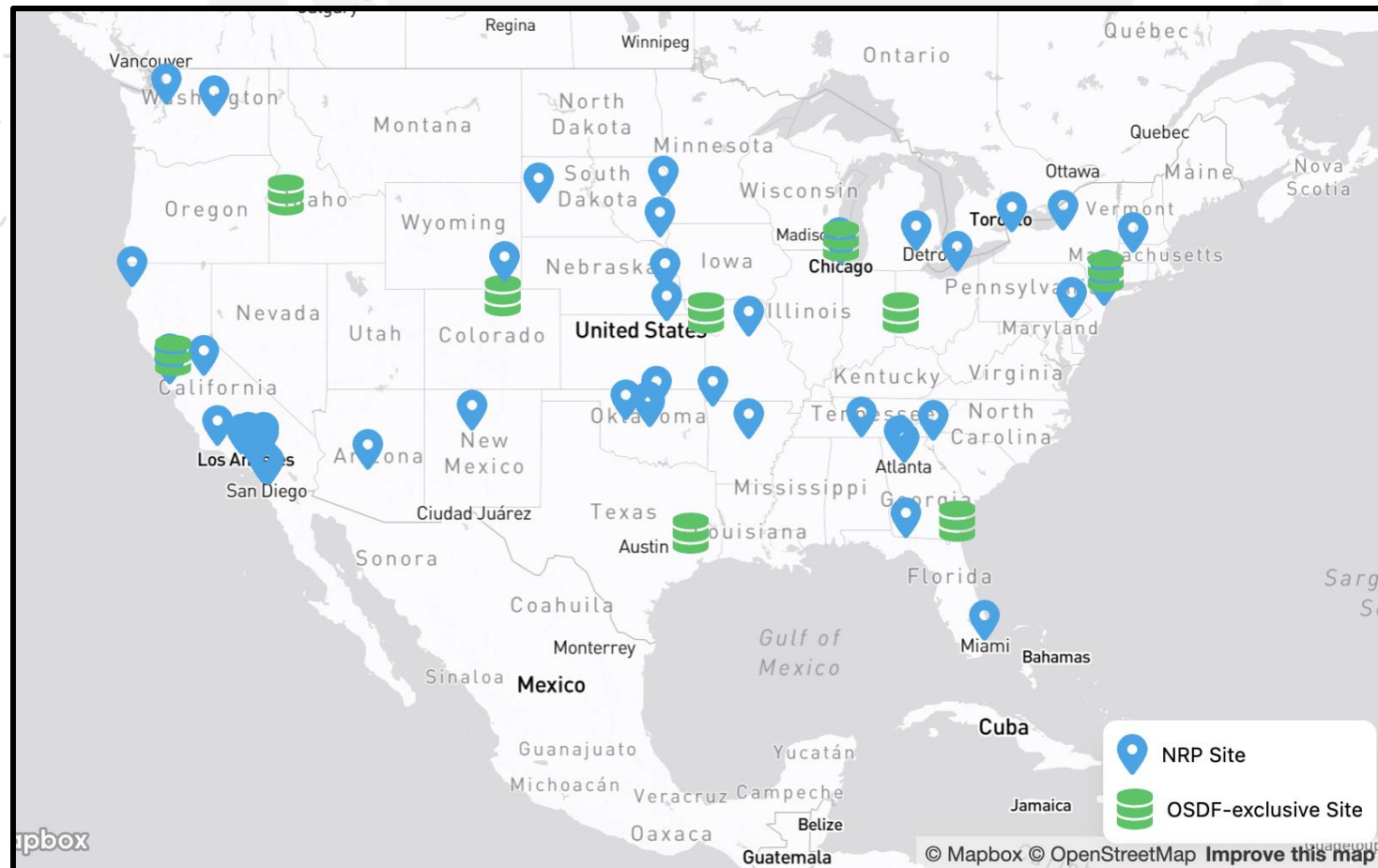
- Connect many repositories to the distribution fabric.
- Provide clients that enable as many use cases.

And benefit from the network effects.



A bit on the distribution layer...

- Anyone can run a cache!
- However, the OSDF centrally runs regional caches, mostly at network locations.
- Builds on top of a distributed Kubernetes cluster run by the National Research Platform (NRP).
 - Single, uniform interface to run services across the country.
- “Typical” cache hardware is ~100GbE / 20TB NVMe.





The OSDF: Connecting your repository

The OSDF provides an “adapter plug”, connecting your science repository to the national and international cyberinfrastructure.

The OSDF is operated by



Using hardware from



And integrates a wide range of open science,



As part of the OSG Consortium’s Fabric of Services



OSDF & Pelican

- You may have seen prior presentations about the OSDF – it (or predecessors) have existed for ~10 years.
- We split out the technology powering the OSDF and christened it the “**Pelican Platform**”.
 - Same components as before, just integrated into a standalone platform.





The Pelican Project

The OSDF is operated by  using hardware from  and others.

Who develops the software?

The Pelican project (OAC-2331480) is a newly-funded, \$7M/4-year project with the following goals:

1. Strengthen and Advance the OSDF.
2. Expand the types of computing where OSDF is impactful.
3. Expand the science user communities.
 - With a particular driver of the climate community.



OSDF by the numbers

Over the last 12 months, the OSDF transferred

**230_{PB} &
125 req/s**

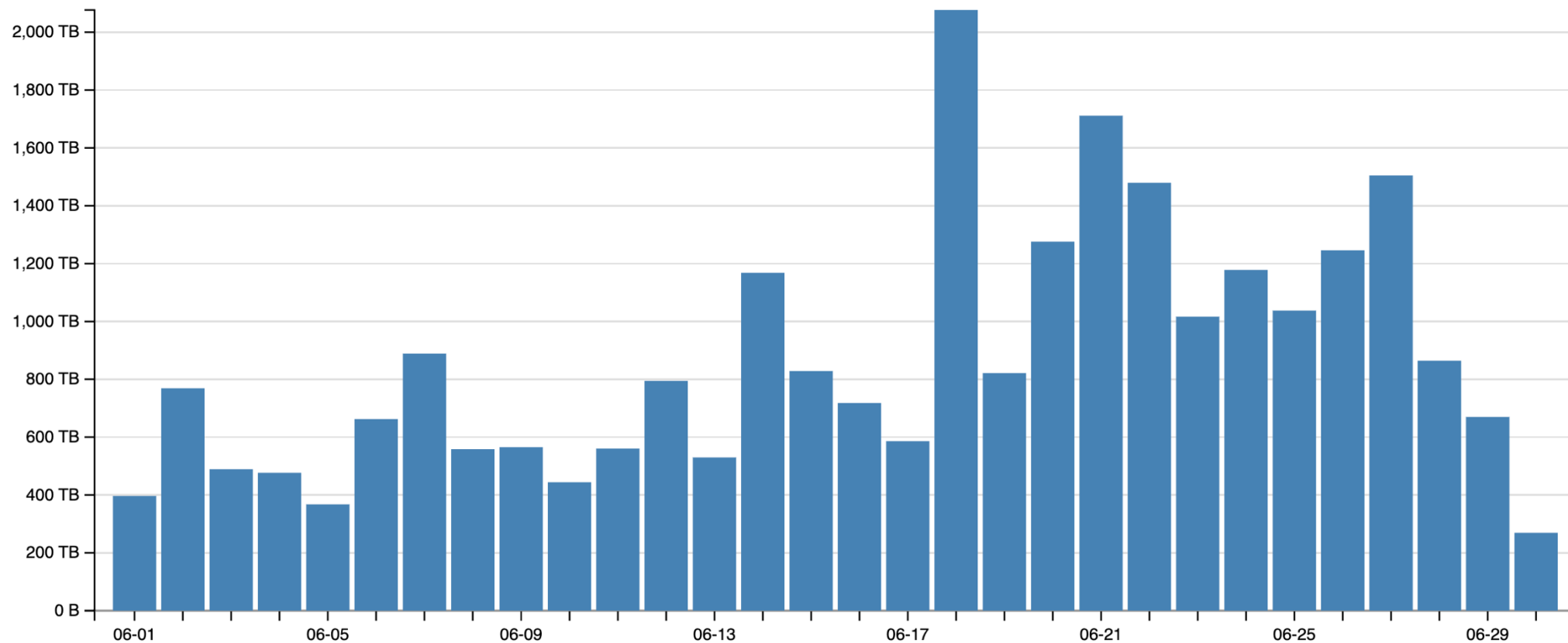
Data used by

**15 science
collaborations &
~120 OSPool
users**





Example Daily Volume – June 2024



Note: individual experiments can still dominate a day's activities.



How does the OSDF work?

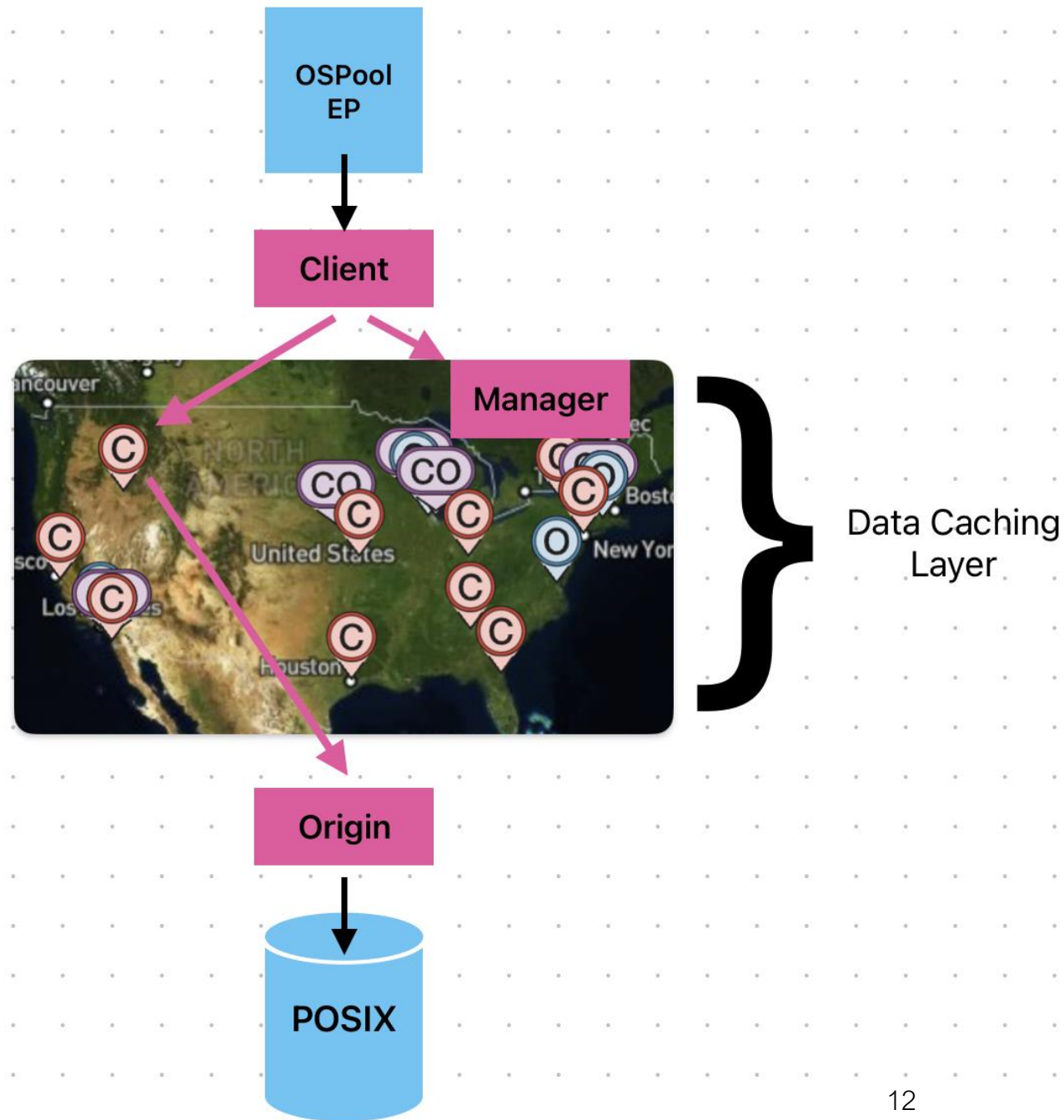
A brief tour through the Pelican architecture as implemented by the OSDF.



OSDF in Practice

- Currently, the most common client for the OSDF is the OSPool.
- The OSPool is a distributed High Throughput Computing service, part of the OSG Consortium and run by PATH.
 - The OSPool is a distributed HTCondor pool, run across ~60 US sites, including **28 CC* awardees** (active + 'alumni').

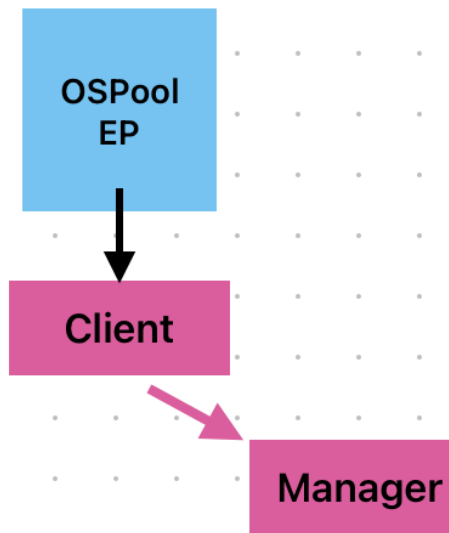
Let's run through a HTCondor Example





OSDF In Practice

- If HTCondor needs an object – say, a container – for a job, the first step is to start the OSDF client.
- The OSDF client contacts the **manager**, requesting to read the object.

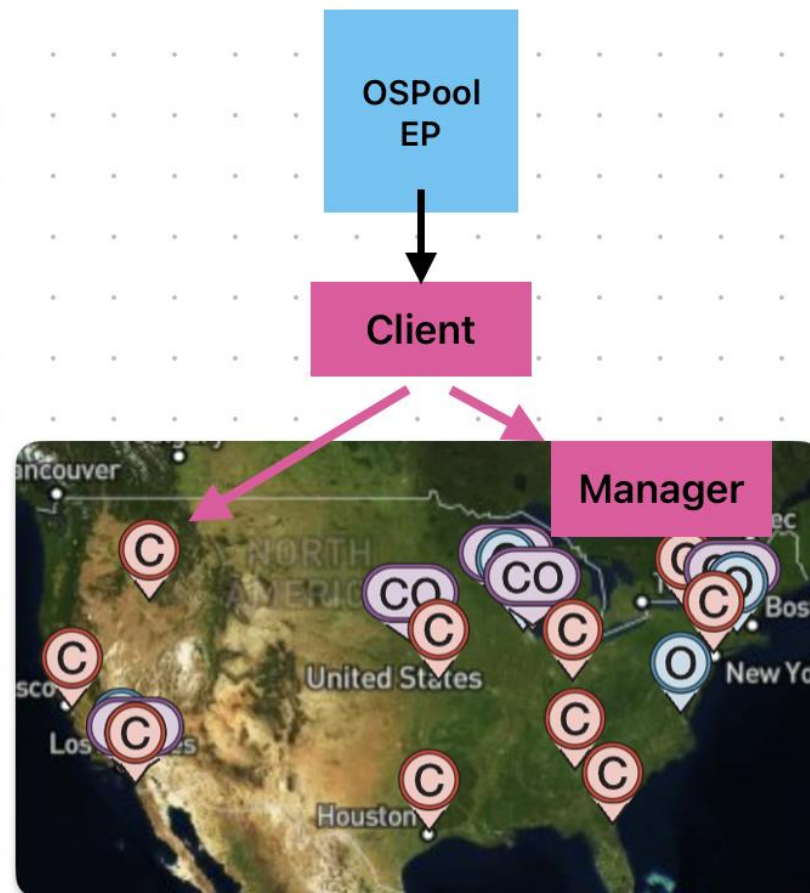




OSDF In Practice

- The manager determines a nearby **cache** to serve the object.
 - Every location in the lower 48 states is within 500 miles from an OSDF cache hosted by the NRP.
- If the object is in cache, it is served to the client immediately.
 - Otherwise...

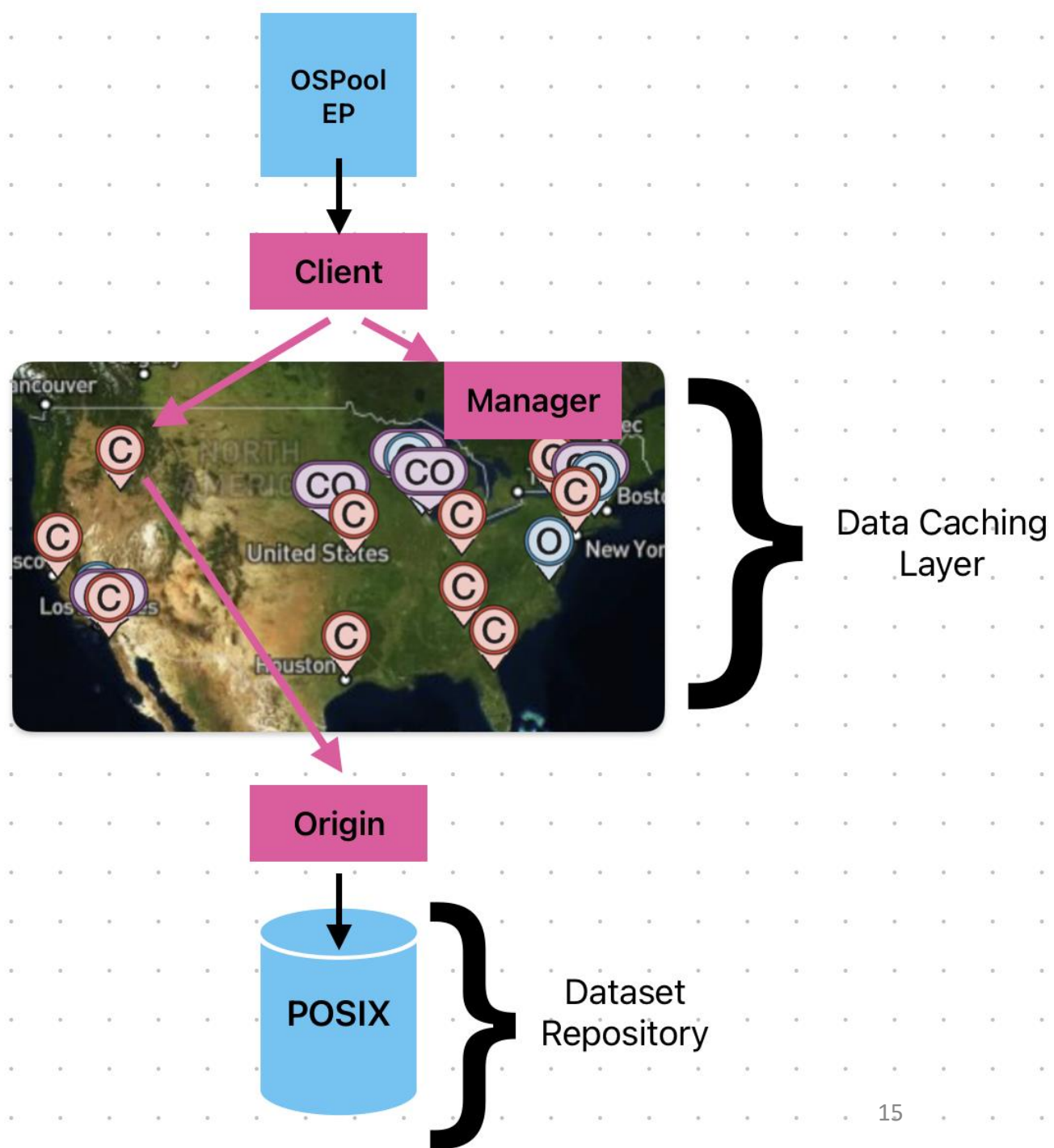
NRP





OSDF In Practice

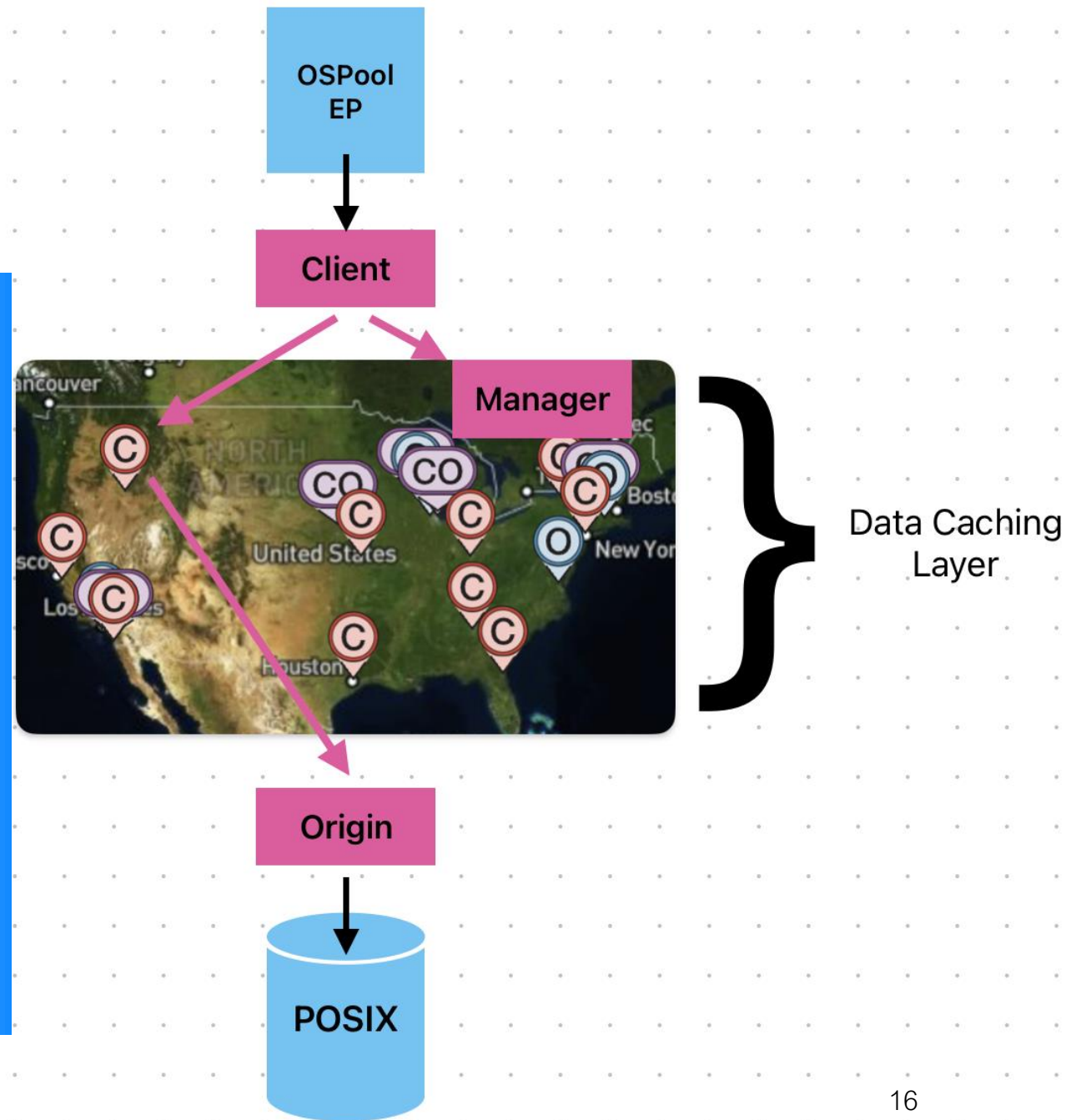
- The cache contacts the origin hosting the object.
 - The object prefix is used as a routing key to determine the correct origin.
- The origin will read the object from the underlying object store.
 - Typically, a filesystem – but expanding to many dataset repository types!





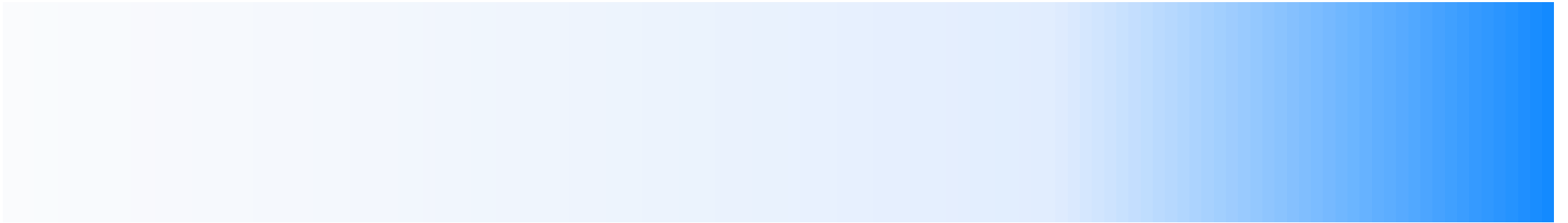
Architecture: Recap

- An **origin service** integrates the object store into the OSDF in the same way a CE integrates a batch system into the OSPool. Interfaces to move data and map authorizations.
- The **cache service** stores and forwards objects, providing scalability to the data access.
- The **manager** selects a source/sink of an object for clients and maintains the namespace.





Zooming in – Technical Components

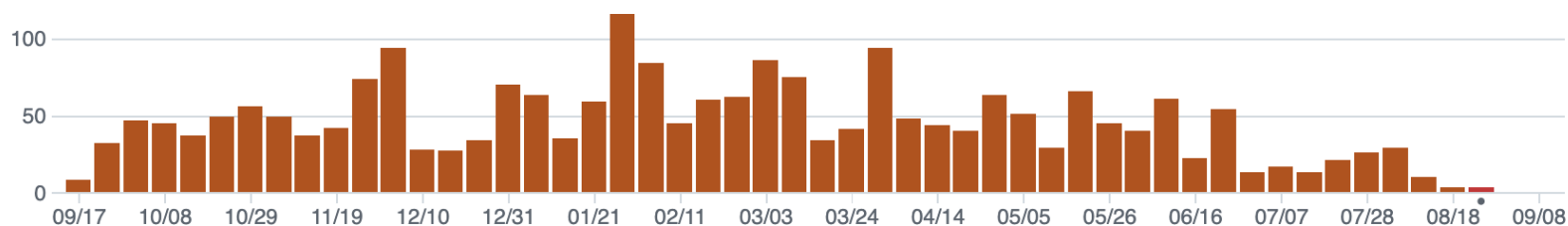




Pelican Implementation

<https://github.com/PelicanPlatform/pelican>

- The Pelican core is a standalone software project.
 - Golang for core; Next.js for web UI.
 - Shipped as a single statically-linked executable.
 - Fairly significant reasonable test suite (~50% code coverage).
- For origins/caches, forks & manages an XRootD process.
 - Dynamically generates XRootD configuration. One, YAML-based config file for admins to manage.
- All components have a web (management) interface.
- Distributed via RPM and containers. Majority use is containers.



Commit graph from the last 12 months



Pelican uses HTTP

- Pelican uses HTTP to move bytes.
- We hew to using standard HTTP where possible. While we *prefer* you use the Pelican client, any HTTP client suffices.
 - Downloading an object? => GET
 - Uploading an object? => PUT
 - Want to know if the object exists? => HEAD

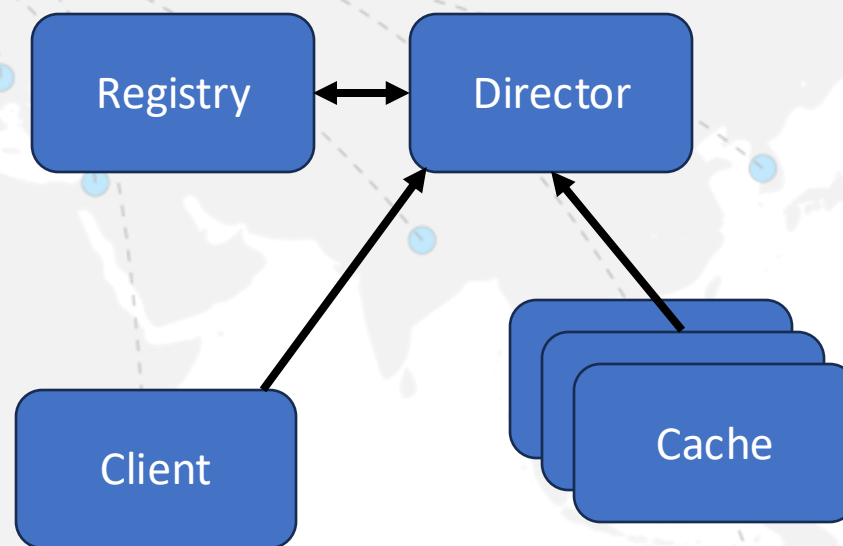
```
pelican -- -bash -- 80x24
[F4HP7QL65F:pelican bbockelm$ curl -L https://director-caches.osgdev.chtc.io/s3.amazonaws.com/us-west-1/hrrrzarr/sfc/20211016/20211016_00z_anl.zarr/2m_above_ground/TMP/2m_above_ground/TMP/6.2 > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left     Speed
 100  186    100   186     0     0    2534      0  --:--:-- --:--:-- --:--:--   2547
 100 22083   100 22083     0     0    97k      0  --:--:-- --:--:-- --:--:--  1960k
F4HP7QL65F:pelican bbockelm$
```



Pelican “Manager” Components

The central manager contains two components:

- The **Registry** maintains the authoritative list of known caches, origins, and namespaces.
 - Also associates each entity with a list of public keys.
 - Authorization is done by signing an appropriate token with the pubkey.
- The **Director** receives requests from clients / caches and selects an appropriate service.
 - All communication done over HTTP!





Example request from client to director

- > GET /chtc/staging/bbockelm/testfile HTTP/2
- > Host: osdf-director.osg-htc.org
- > User-Agent: curl/8.4.0
- > Accept: */*



Example director response

< HTTP/2 307

< content-type: text/html; charset=utf-8

< date: Mon, 08 Jul 2024 17:17:17 GMT

< link: <<https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=1; depth=3, <<https://stash-cache.osg.htc.io:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=2; depth=3,...

< location: <https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>

< x-pelican-authorization: issuer=<https://htc.cs.wisc.edu>

< x-pelican-namespace: namespace=/htc, require-token=true, collections-url=<https://origin-auth2000.htc.wisc.edu:1095>

< x-pelican-token-generation: issuer=<https://htc.cs.wisc.edu>, max-scope-depth=3, strategy=OAuth2

< content-length: 109



Example director response

< HTTP/2 307

< content-type: text/html; charset=utf-8

< date: Mon, 08 Jul 2024 17:17:17 GMT

< link: <<https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=1; depth=3, <<https://stash-cache.osg.htc.io:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=2; depth=3,...

< location: <https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>

< x-pelican-authorization: issuer=<https://htc.cs.wisc.edu>

< x-pelican-namespace: namespace=/htc, require-token=true, collections-url=<https://origin-auth2000.htc.wisc.edu:1095>

< x-pelican-token-generation: issuer=<https://htc.cs.wisc.edu>, max-scope-depth=3, strategy=OAuth2

< content-length: 109



Example director response

< HTTP/2 307

< content-type: text/html; charset=utf-8

< date: Mon, 08 Jul 2024 17:17:17 GMT

< link: <<https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=1; depth=3, <<https://stash-cache.osg.chtc.io:8443/htc/staging/bbockelm/testfile>>; rel="duplicate"; pri=2; depth=3,...

< location: <https://osdf-uw-cache.svc.osg-htc.org:8443/htc/staging/bbockelm/testfile>

< x-pelican-authorization: issuer=<https://htc.cs.wisc.edu>

< x-pelican-namespace: namespace=/htc, require-token=true, collections-url=<https://origin-auth2000.chtc.wisc.edu:1095>

< x-pelican-token-generation: issuer=<https://htc.cs.wisc.edu>, max-scope-depth=3, strategy=OAuth2

< content-length: 109



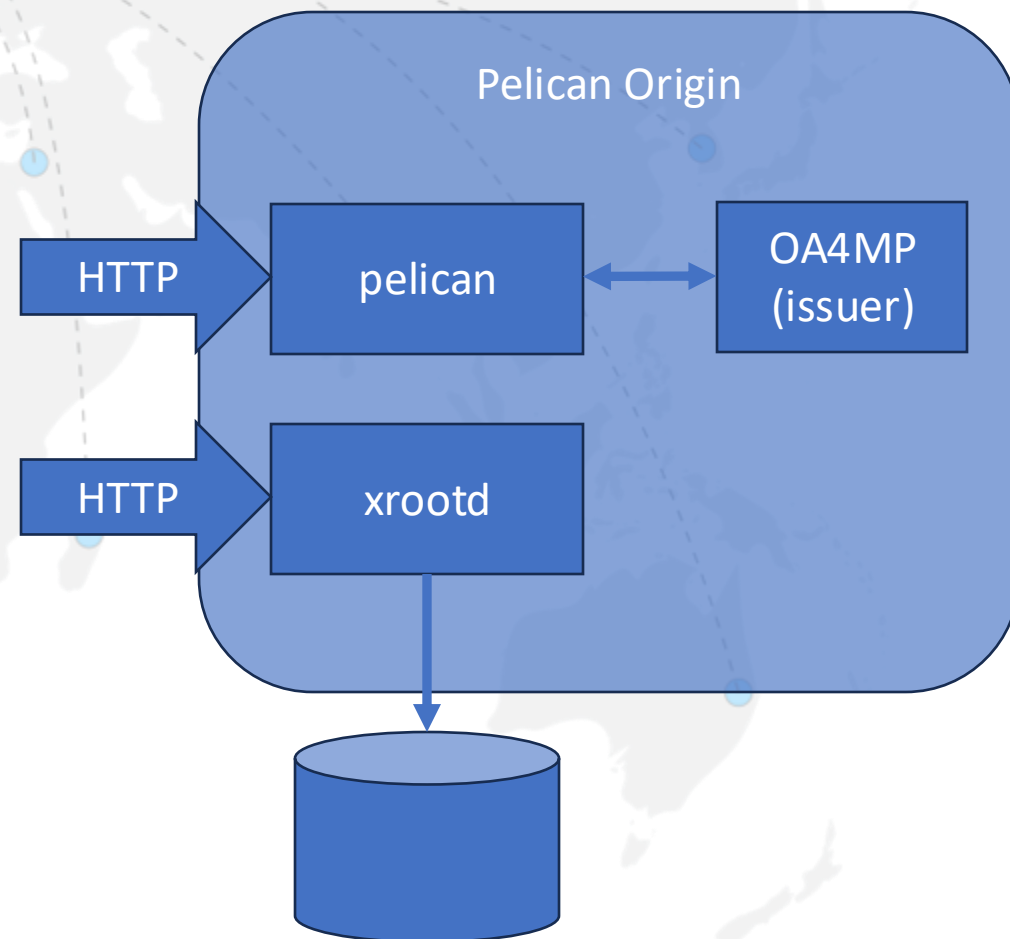
Director Response

- If you speak “plain HTTP”, you only understand the “blue” headers and will successfully access the data.
- If you are the “Pelican client”, you can interpret the “red” headers:
 - **X-pelican-authorization**: What token the client needs to successfully access the data.
 - **X-pelican-namespace**: What namespace the object is in. Informs client how to reuse the director response; no need to return to director for each object.
 - **X-pelican-token-generation**: If the client doesn't have a usable token, how to receive one.
 - **Link**: An ordered list of potential endpoints (caches) that can serve the requests. Actually, a standard RFC header (RFC 6249).



Pelican Origin

- Pelican daemon launches and manages the xrootd daemon.
 - However, HTTP data movement requests go straight to the xrootd process.
 - pelican's HTTP interface is used for monitoring, management, and token issuer.
- XRootD can be configured for a variety of backends.





“Batteries Included” Origin

The image displays two screenshots of the Pelican Origin web interface. The top screenshot shows the 'Status' page, which includes a 'Data Exports' section with a table of permissions:

Prefix	PublicRead	Read	Write	Listing	FallBackRead
Federation Prefix: /ospool/ap40/data	X	X	✓	X	✓
Storage Prefix: /mnt/cephfs/fuse/ospool/ap40/data			✓	✓	✓

The bottom screenshot shows the 'Server' configuration page with the following settings:

- Server.EnableUI: True
- Server.ExternalWebUrl: https://ospool-ap2140.chtc.wisc.edu:8444
- Server.Hostname: ospool-ap2140.chtc.wisc.edu
- Server.IssuerHostname: (empty)
- Server.IssuerJwks: (empty)
- Server.IssuerPort: 0
- Server.IssuerUrl: https://osa-htc.org/ospool

We aim to simplify the art of running an origin:

- New web UI for viewing, monitoring, and configuring the origin.
- Origin runs built-in health checks
- Can use “connection reversing” so incoming firewall port / hostname / host certificate not needed.

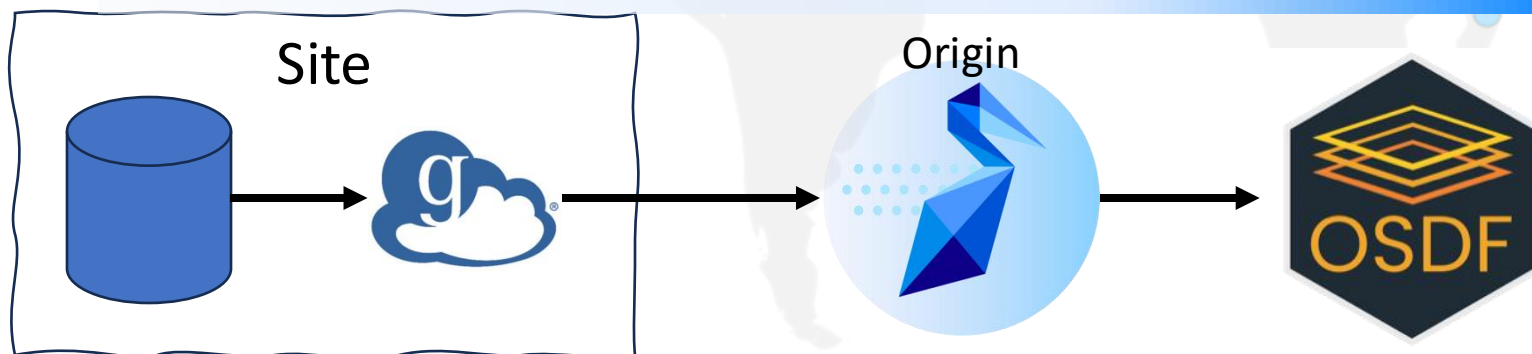


New Backends

Beyond the traditional POSIX storage, we've added the following backends:

- **S3**: Works with any S3-compatible endpoint
- **Generic HTTP**: Integrate existing HTTP endpoint into the OSDF.
- **Globus**: Users must authorize sharing a collection to the origin
- **XRootD**: Uses XRootD proxying module.

Note each of these backends can be used remotely – origin does not need to be present at the local site.

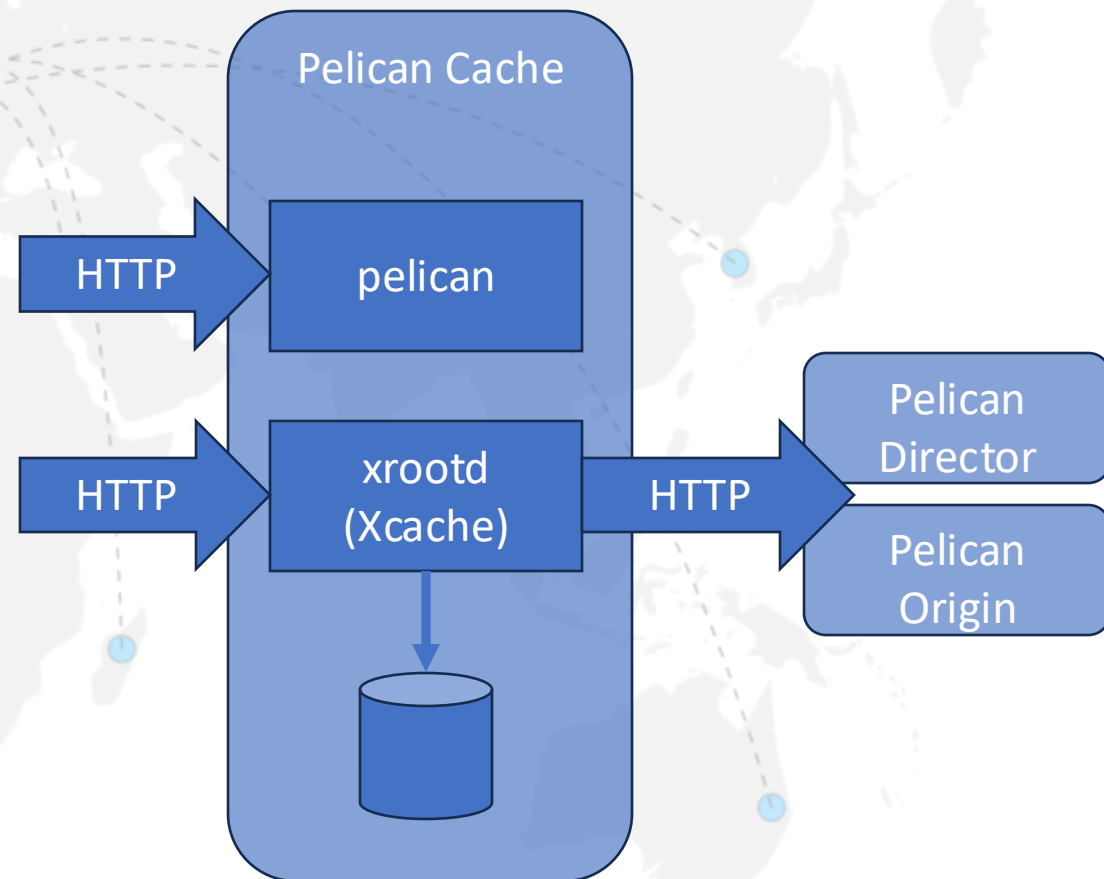


XRootD



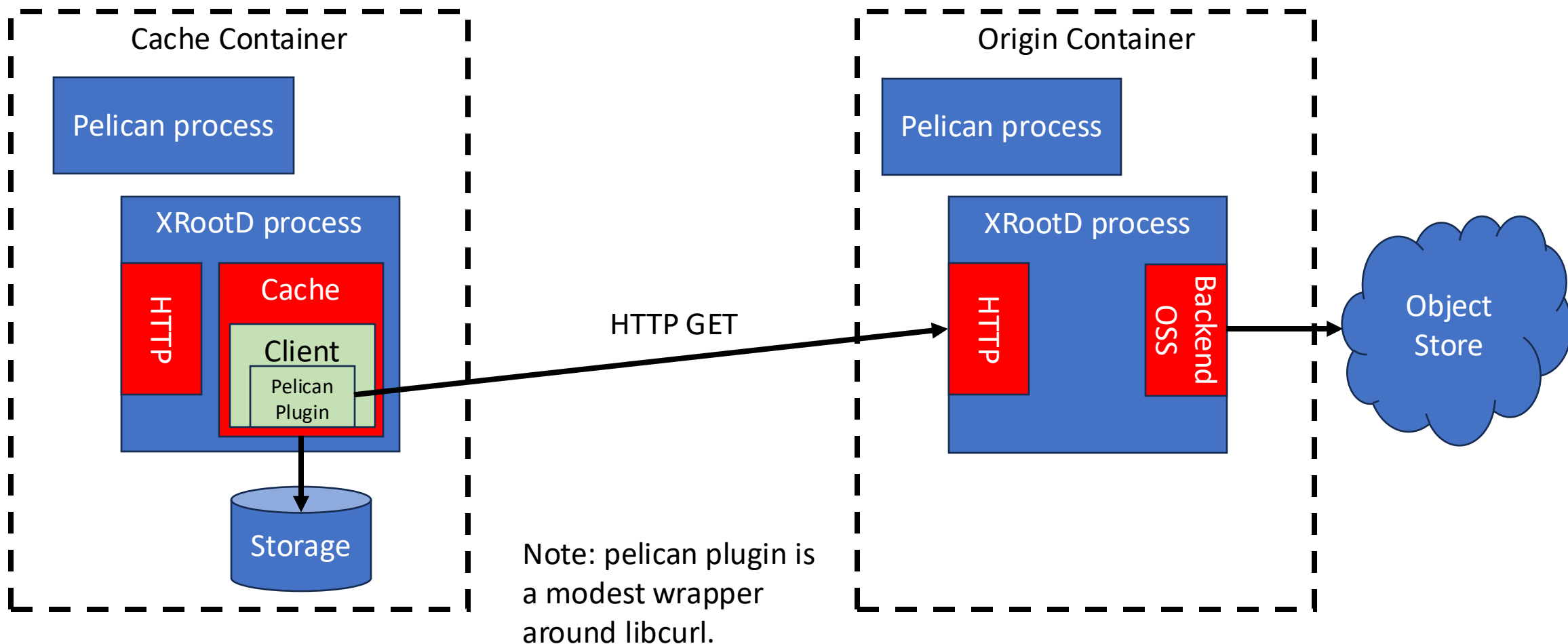
Pelican Cache

- Similar setup to the origin: two separate processes, two ports for HTTP.
 - Given the director and origin works exclusively over HTTP, the XCache must talk to them over HTTP as well.





A slide for the XRootD people out there...





Improved Dashboard and Monitoring



Patrick Brophy

Mentor(s):

Haoming Meng

Monitoring work is ongoing in two lines:

- Better communicating the data we have:
 - Generating improved graphs of data moved, number of requests, breakdown by project.
 - **Goal:** Clearly show how your institution is impacting/enabling others, just like the HTCondor-CE dashboard.
- Gathering more data:
 - XRootD has deep coverage of successful transfers. Little aggregation of filesystem errors; no monitoring of protocol-level (HTTP) events. Contributing patches upstream to expose this data.
 - What else do you want to see?



Client - CLI

- While curl *can* be used, we have quite a bit of specialized knowledge:
 - Immutable files means file download resumption is straightforward.
 - Parse the extra director headers to understand where backup caches are. Retry as necessary.
 - From the director headers, we know what tokens are required and how to generate them.
- The client can also do metadata operations (“stat”, “list”), recursive upload/downloads of directories.
- (Coming soon!) Trigger the prestaging of objects to a cache.
- The client is all in the same static binary as the server – the entire system is the one file.



Client - Python

- While we love CLIs, we want to tap into the Python community (which is more interactive/visualization focused).
- Accordingly, we started a [FSSpec for Pelican](#).
 - Summer student was able to use the FSSpec to run PyTorch against the OSDF.
- Allows us to tap into more communities (particularly, a large contingent of climate science).



Client - HTCondor



+



=



- The most commonly-used client is the HTCondor plugin!
- Allows you to specify “osdf:///” or “pelican:///” URLs for the input or output sandbox.
- HTCondor and Pelican are the same team: While separate code bases, features are developed jointly across the platforms.
 - Example: **Transfer ads.** Pelican emits detailed transfer statistics for each transfer attempt (successful or not) done at the EP.
 - These transfer statistics are copied to the AP.
 - condor_adstash can then aggregate the ads into an ElasticSearch database.
- Each month, we scan ElasticSearch to review hold messages generated by Pelican and attempt to make them more “human readable”
- HTCSS can start up a Pelican “local cache” daemon, setting aside EP space to be used for common input files.



Final Thoughts – Pelican & HTCondor

- In case it got lost in the details:
Pelican represents an investment into CHTC's HTC vision.
 - It is the “data component”, complimenting HTCSS.
- The project's first year focused on re-engineering the OSDF.
 - Implement new central components, Improve packaging, Increase observability, Expand backends.
- What's next? Tighter integration with HTCSS.



Questions?

This project is supported by the National Science Foundation under Cooperative Agreements OAC-2331480. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.