Introducing Pelican: Powering the OSDF

Introducing the OSDF

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

* ‘Compute’ is viewed broadly; everything from a browser to a cluster.

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 PATh

Using hardware from NRP

And integrates a wide range of open science,

As part of the OSG Consortium’s Fabric of Services

OSDF Integrates Independent Repositories into a common fabric

★ AWS Open Data

★ NCAR

LIGO ★

OSDF

jupyter

DeltaAI

I | NCSA

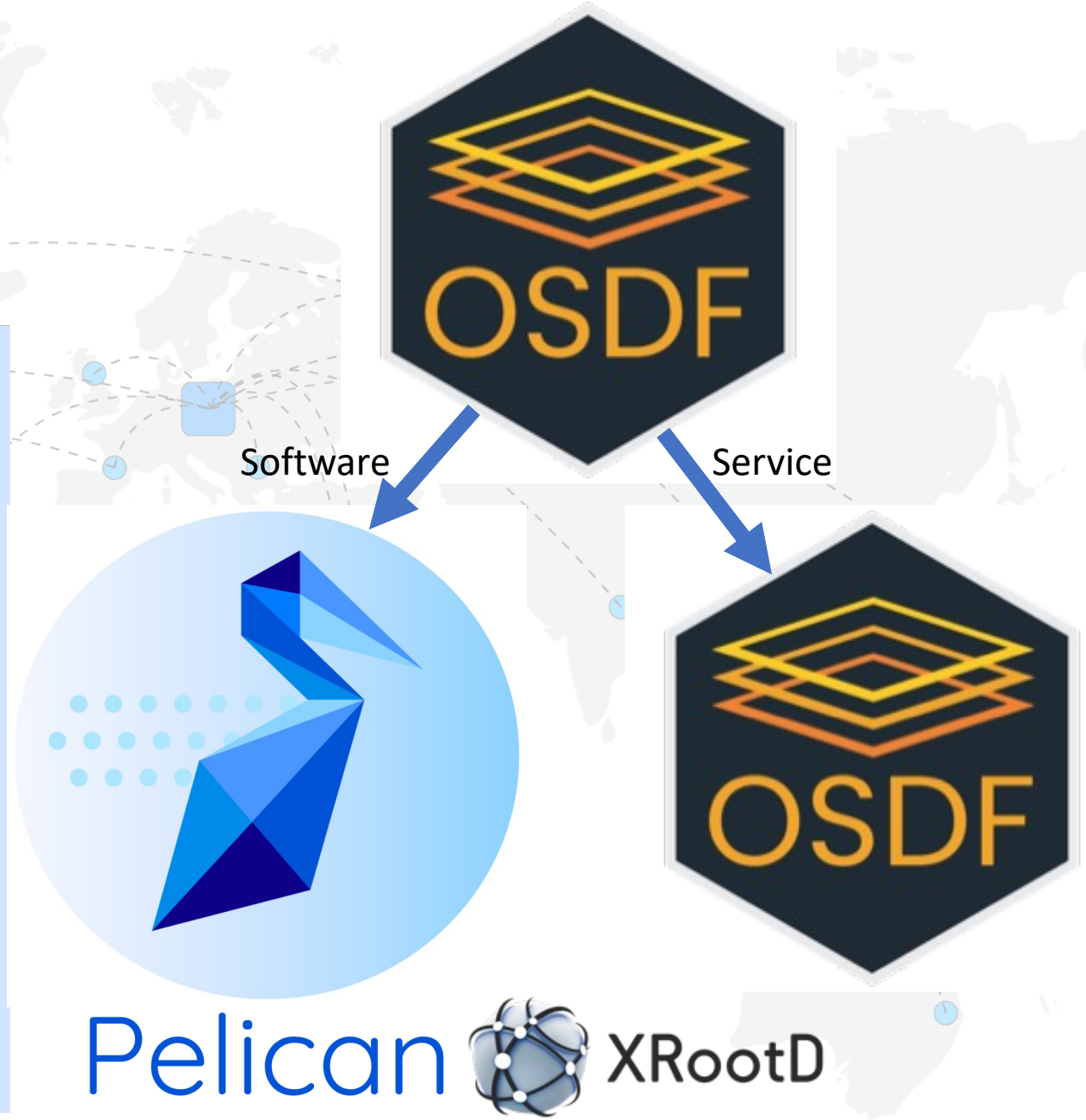
OSPool ★

★ = existing integration

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

OSDF & Pelican

- The next presentation will go into some highlights of what the OSDF has been delivering for science.
- 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

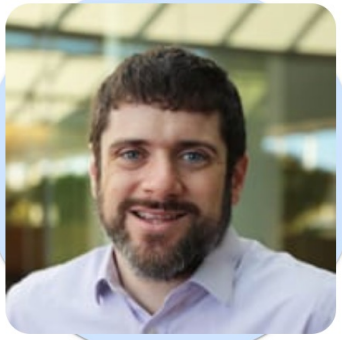
The OSDF is operated by PATh using hardware from NRP and others.

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.

Meet the team



Brian Bockelman
Principal Investigator

Morgridge Institute for Research

Miron Livny
Co-Principal Investigator

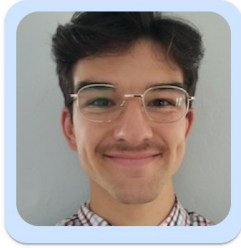
University of Wisconsin–Madison

Frank Wuerthwein
Co-Principal Investigator

University of California San Diego

Meet the team

Tae Kidd
Project Manager
Morgridge Institute For Research



Cannon Lock
Web Developer
Morgridge Institute for Research

Justin Hiemstra
Software Engineer
Morgridge Institute For Research

Brian Lin
OSG Software Area Coordinator
University of Wisconsin–Madison

Emma Turetsky
Software Engineer
Morgridge Institute for Research

Alja Mrak Tadel
Analytic Programmer
University of California San Diego

Rich Wellner
SDx Director
San Diego Supercomputer Center

William Swanson
Research Cyberinfrastructure Specialist
University of Wisconsin–Madison

Matevz Tadel
Project Scientist
University of California San Diego

Haoming Meng
Research Software Engineer
Morgridge Institute For Research

Christina Koch
Lead Research Computing Facilitato
University of Wisconsin - Madison

Lili Bicoy
Student Science Writer
Morgridge Institute For Research

Mátyás Selmeci
Software Integration Developer
University of Wisconsin–Madison

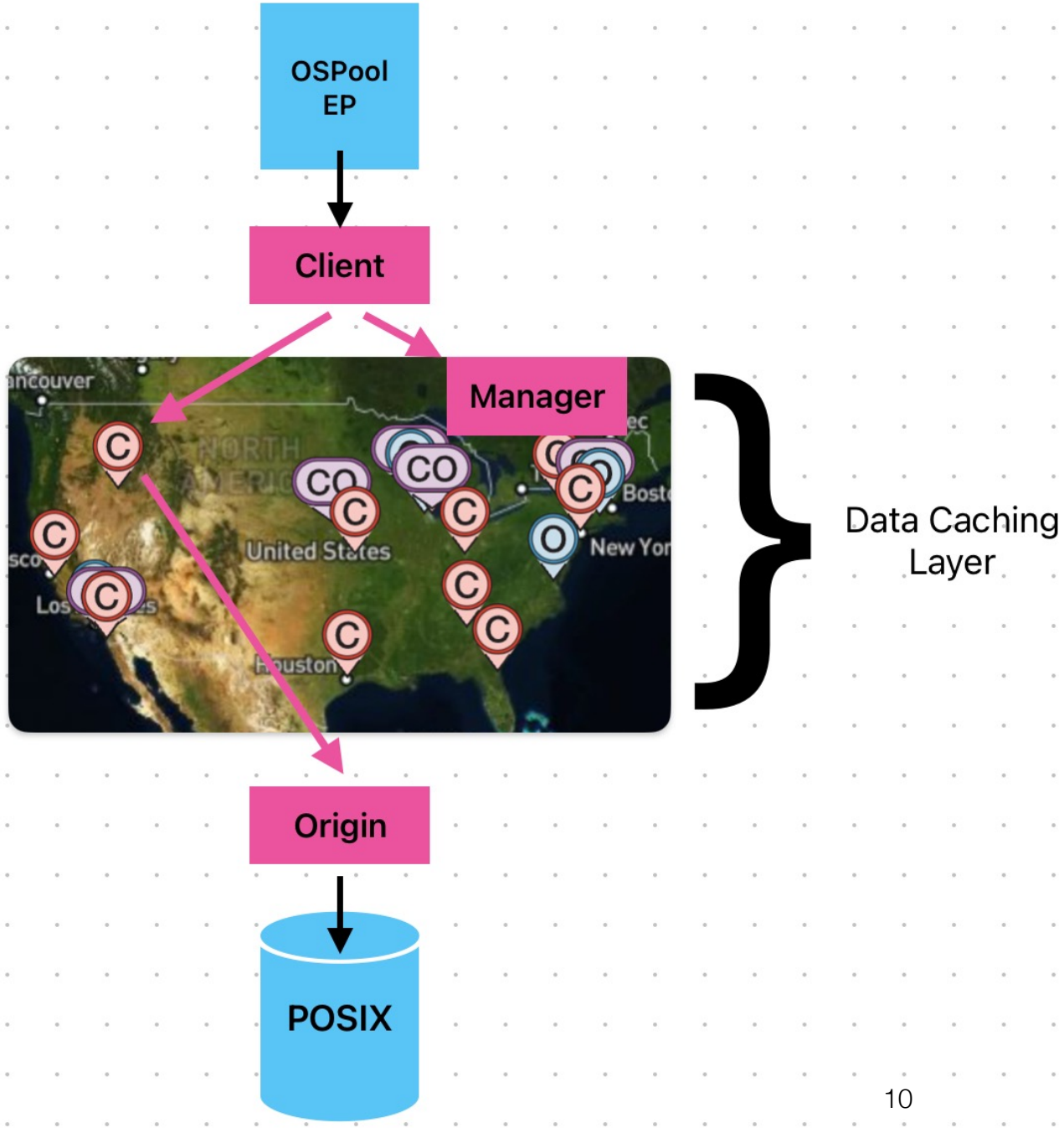
How does the OSDF work?

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

OSDF in Practice

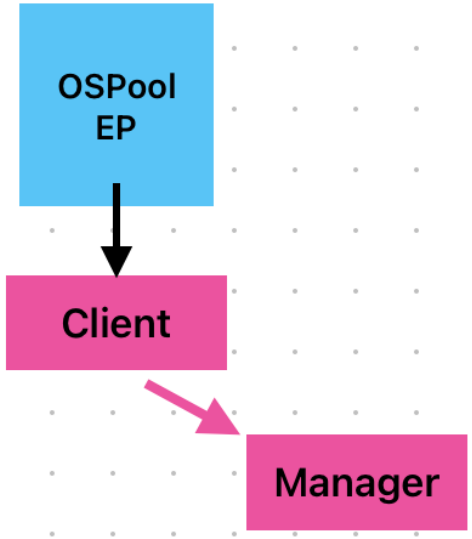
- Currently, the most common use for the OSDF is managing inputs to OSPool (or similar pools).
- Clients include:
  - **HTCSS plugin**, allowing “osdf://” URLs in a HTCondor submit file.
  - **Standalone CLI** with “cp”-like semantics
  - **Python** fsspec implementation, linking Pelican to the Python ”data science” ecosystem.

Let’s run through a HTCondor Example



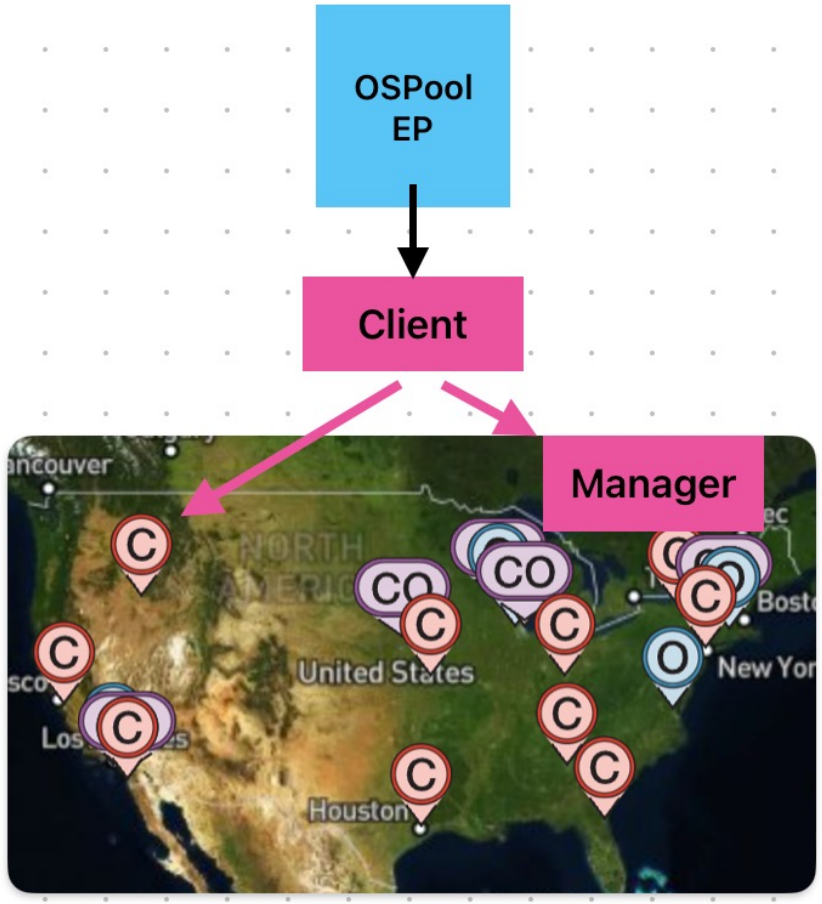
OSDF In Practice

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



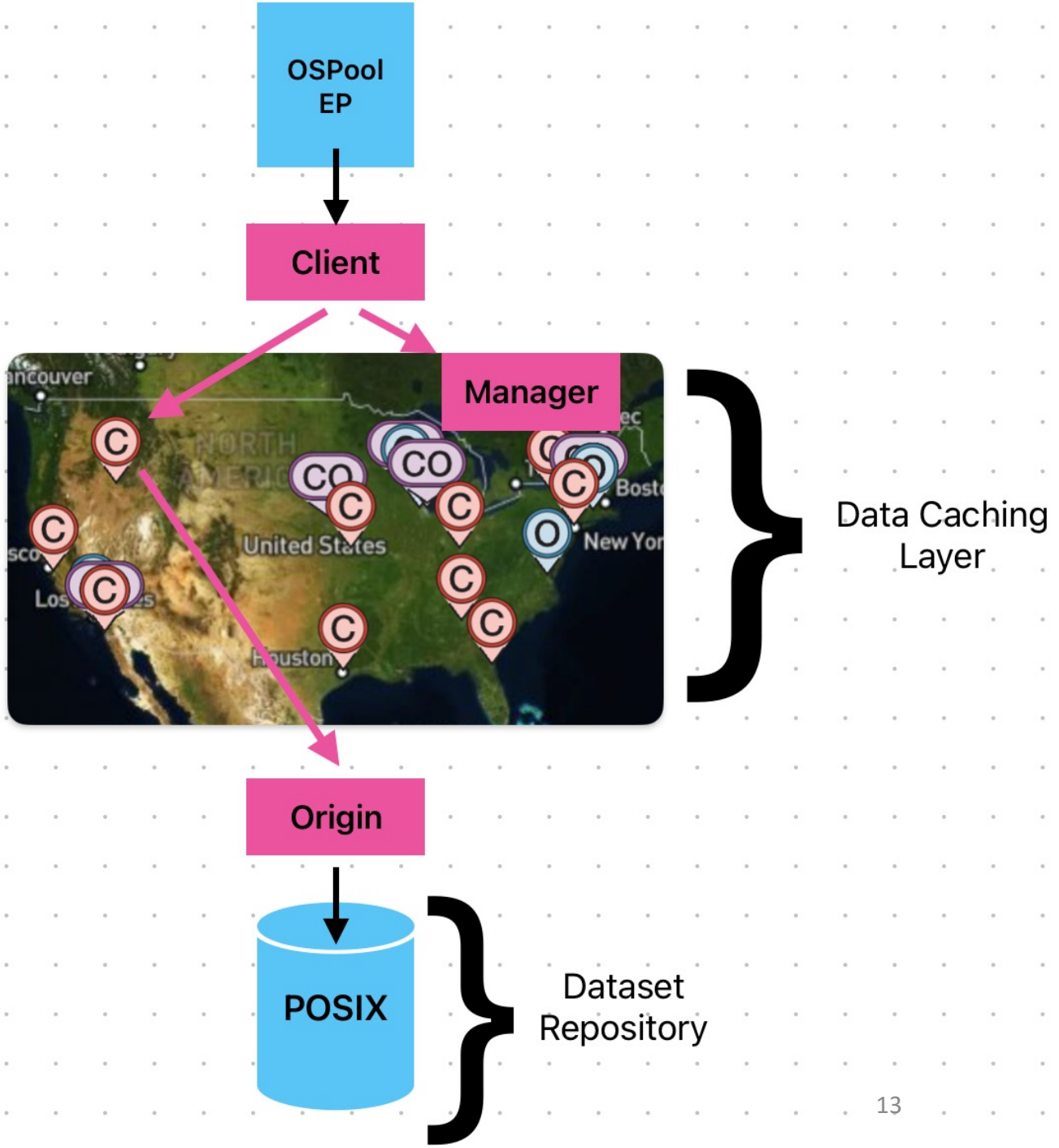
OSDF In Practice

- 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…



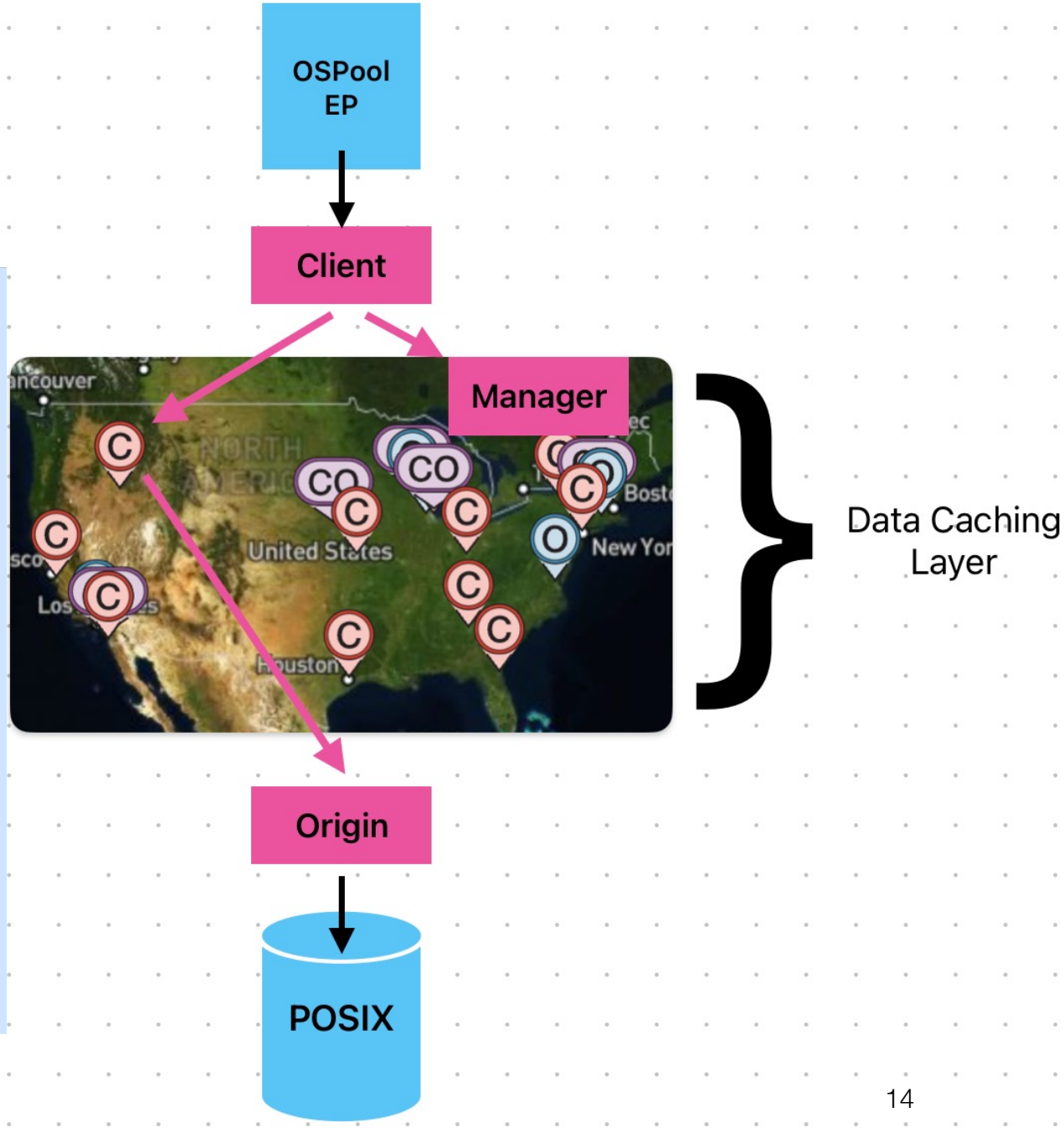
OSDF In Practice

- 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 POSIX filesystem – but other backends exist!

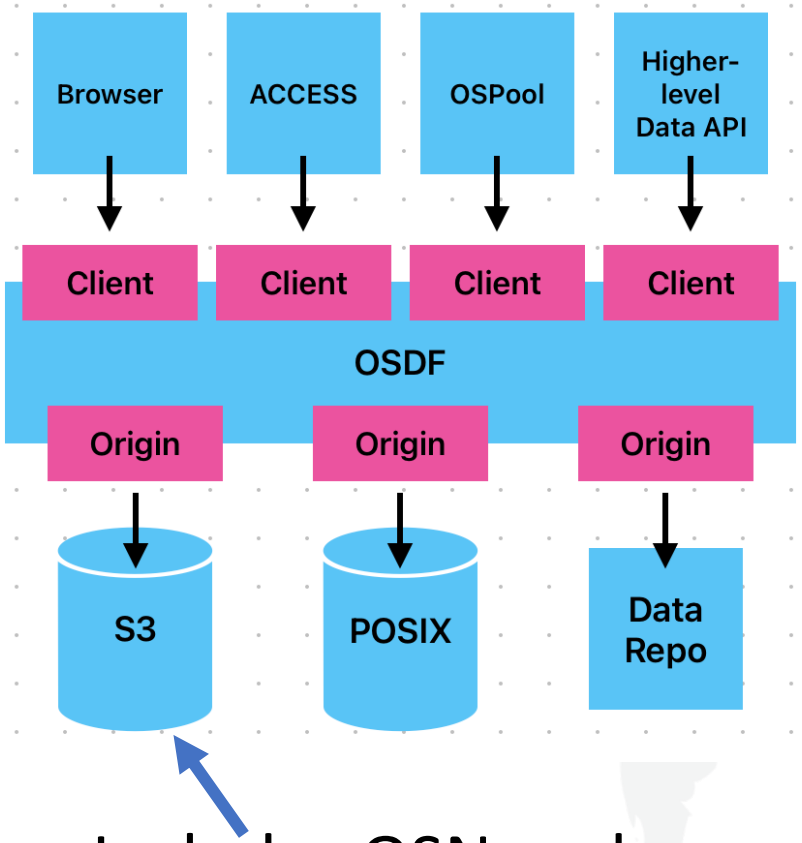


Architecture: Recap

- 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.



OSDF Architecture - Vision



Includes OSN pods – anything S3 compat.

- **Long-term vision**: Pelican Platform, provides a 'transport bus', connecting a broad range of dataset providers to consumers.
  - We aim to broaden the supported clients (Python, Browser-based) and provide integrated services with other projects.
- Become a platform beyond the initial context of a service supporting distributed High Throughput Computing.

Pelican: One Year Down the Line

What are the big project achievements after ~1 year?

Rearchitecting Core Services

- We reimplemented the central services of the federation. There's now two main central components:
  - The **Director** receives the client's HTTP requests (GET, PUT) for objects and selects an origin/cache to service the request.
  - The **Registry** maintains the list of approved caches, origins, and known namespace prefixes.
    - Each registry entry is associated with a public key; the public key is used to sign tokens authorizing actions.
- Other federation-level activities exist:
  - Periodically test all caches and origins for functionality.
  - Connection brokering.
  - Monitoring activity (embedded Prometheus).

Standalone Software

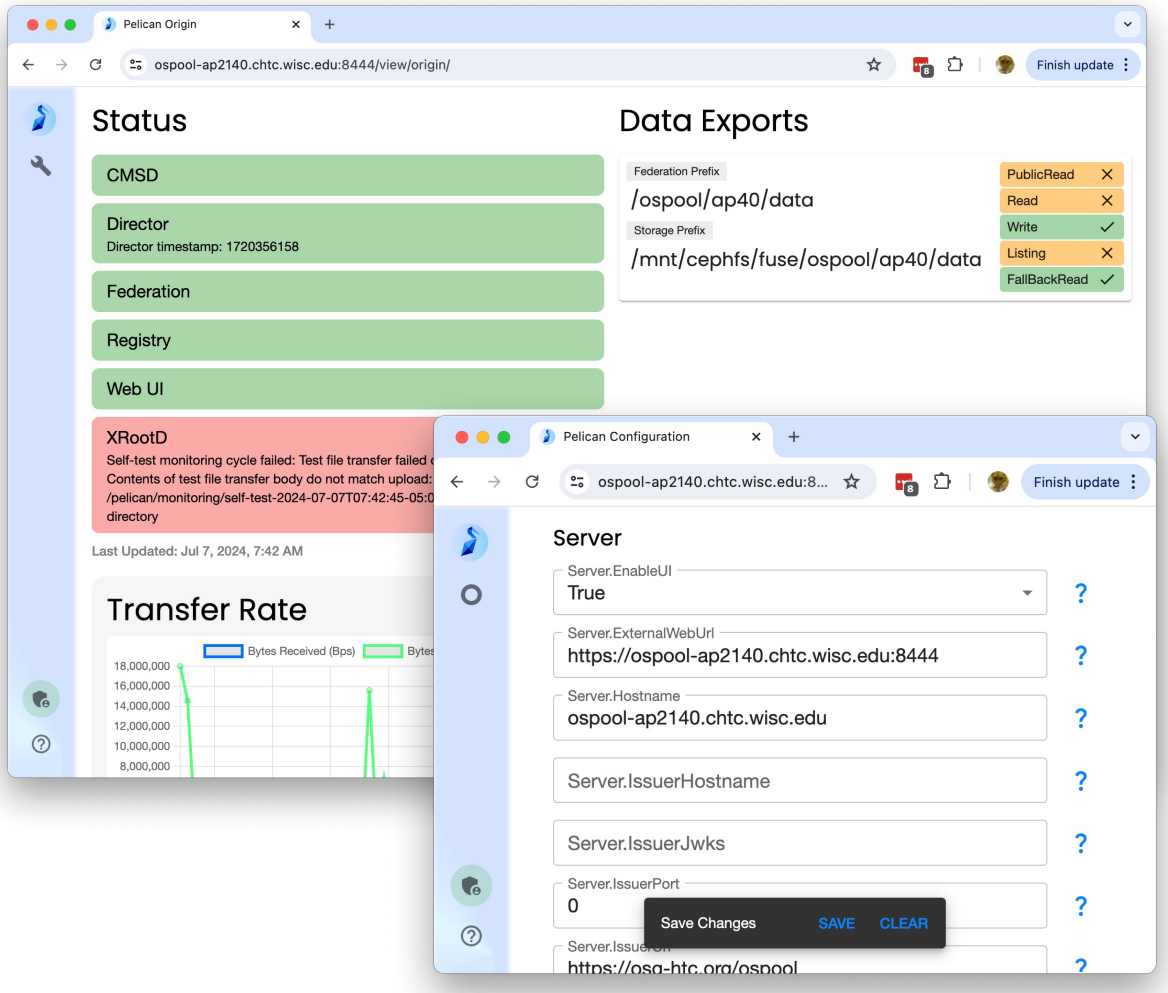
- Pre-Pelican, it was not possible to setup your own data federation.
  - The venerable OSDF client, `stashcp`, had hardcoded hostnames and OSDF-specific logic. Deeply tied into the OSG Topology application and WLCG GeoIP services.
- As Pelican, it's now a proper software project:
  - Software is managed on GitHub in a single organization. Monthly feature releases, ~weekly bugfix releases.
  - You can setup your own, self-contained data federation!
  - There are dozens of unit tests and end-to-end integration tests run with each commit (~55% test coverage).
- While "unit test coverage" isn't a user-visible feature, it allows us to start deliver new and evolving functionality with confidence.

Converting OSDF to Pelican

- We are rolling out new services and protocols via a new software stack … onto the existing infrastructure!
  - E.g., a Pelican-based cache must be 100% compatible with old and new origins and clients.
  - No “flag day” option, cannot force client upgrades.
- Transition of services is >50% done.
  - Slower than anticipated. Familiar story: periodically pause to implement previously-unknown use cases, cleanup old messes in topology and containers.
  - Until we’ve 100% cutover, Pelican carries the burden of supporting both old and new clients.

Microsoft Copilot’s interpretation of “changing the engine while the Pelican is flying”

“Batteries Included” Origin



We aim to simplify the art of running an origin:

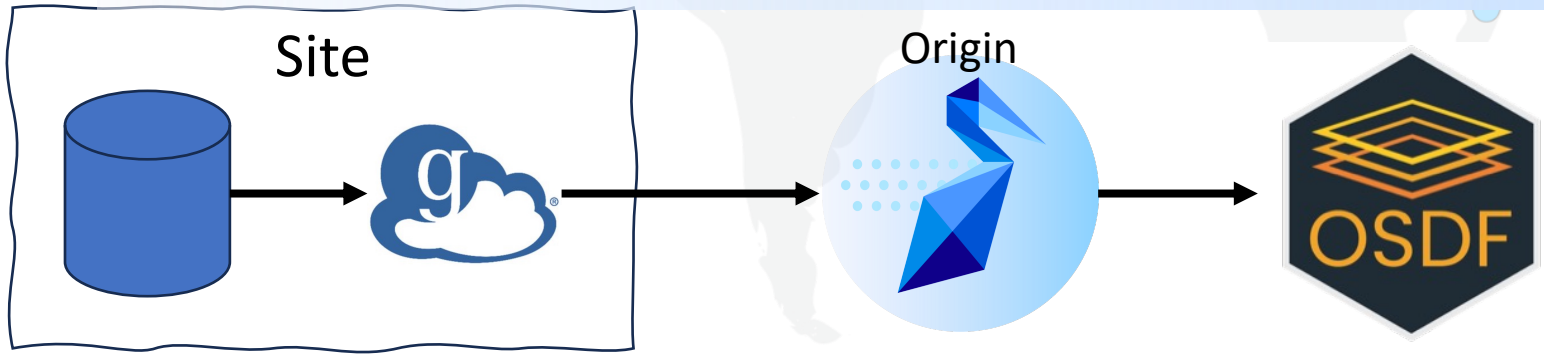
- New web UI for viewing, monitoring, and configuring the origin.
- Origin runs built-in health checks
- Can use “connection reversing” so incoming firewall port / hostname / host certificate not needed.

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.



amazon S3

XRootD

Pelican: Looking forward

More Backends!

We want to connect any reasonable scientific data repository to the OSDF. This includes:

- Continuing to mature the Globus backend: more functionality, more example integrations.
  - For WLCG folks: aim is sufficient integration to use with FTS transfers.
- **DataVerse**: Data repository software (https://dataverse.org/) often used by institutional libraries. Embed OSDF links directly into DataVerse instances.
- Improve origin monitoring and throttling. E.g., NASA datasets are available through HTTP but access must be strictly rate-limited.

Work in-progress by summer CHTC Fellow

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? Chase down a Pelican team member!

Deeper HTCSS Integration

- Each team HTCondor uses Pelican to transfer a file, a summary ClassAd is created (and can be ingested to ElasticSearch; see Jason's talk tomorrow).
  - Working on a common schema so the AP can make more informed decisions about retries or alternate transfer methods.
- 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.
  - Goal is to deploy this to the OSPool over the summer

HTCSS and Pelican are a single team!

Collections Management

- Pelican/OSDF are inherently about immutable *objects*. Upload, download, stat – works at the individual object level.
- Next year, we plan to expose a *collections API*, representing a group of objects.
  - Analogous to “buckets” in S3 but decoupled from the namespace.
  - Collections would be mutable, and may define a set of objects that are the result of a database query.
  - Provides a way to transport metadata to higher-level cataloguing systems.
  - Access to collections will be separately managed, allowing for simple authorization management.

Conclusions

Pelican Year 1 – quite the whirlwind!

- We reengineered the origin and cache services, added new central services, and greatly improved the OSDF's integration with HTCSS.
  - OSDF saw corresponding enormous growth, with some days moving >2PB.
- We've picked up new science partners (notably, NCAR) and supported some great science (NRAO).
- Working to provide more visibility into the system: what's my hardware doing? who's using my objects? who am I impacting?
  - Expect us to take inspiration from OSPool!

Pelican is a happy member of the HTC community and has big plans to move forward data management with HTC.

The OSDF: Connecting to your datasets

To acknowledge all of the partners working together...

OAC-2331480

Provides the software

OAC-2030508

Operates the OSDF services

OAC-2112167

Operates (most of) the OSDF hardware

The OSG Consortium is the “umbrella” we work within.

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.