SC23

Denver, CO | i am hpc.

Accelerating Communications in Federated Applications with Transparent Object Proxies

Greg Pauloski*

Valerie Hayot-Sasson*, Logan Ward^, Nathaniel Hudson*, Charlie Sabino*, Matt Baughman*, Kyle Chard*^, and Ian Foster*^

*University of Chicago, ^Argonne National Laboratory

15 November 2023

FaaS and Workflow Systems

Enable programmers specify ***what*** tasks to perform without regard to ***where*** tasks are executed.

Control and Data Flow

Different problems with different solutions…

Control Flow

- Path the execution takes in an application
- Determining order of operations, scheduling, execution
- Tasks definitions are small

Data Flow

- How data moves through computations
- May accompany the control flow
- Data characteristics vary much more than task definitions

Cloud-hosted FaaS / workflow systems are ***good for control flow*** but ***bad for data flow***.

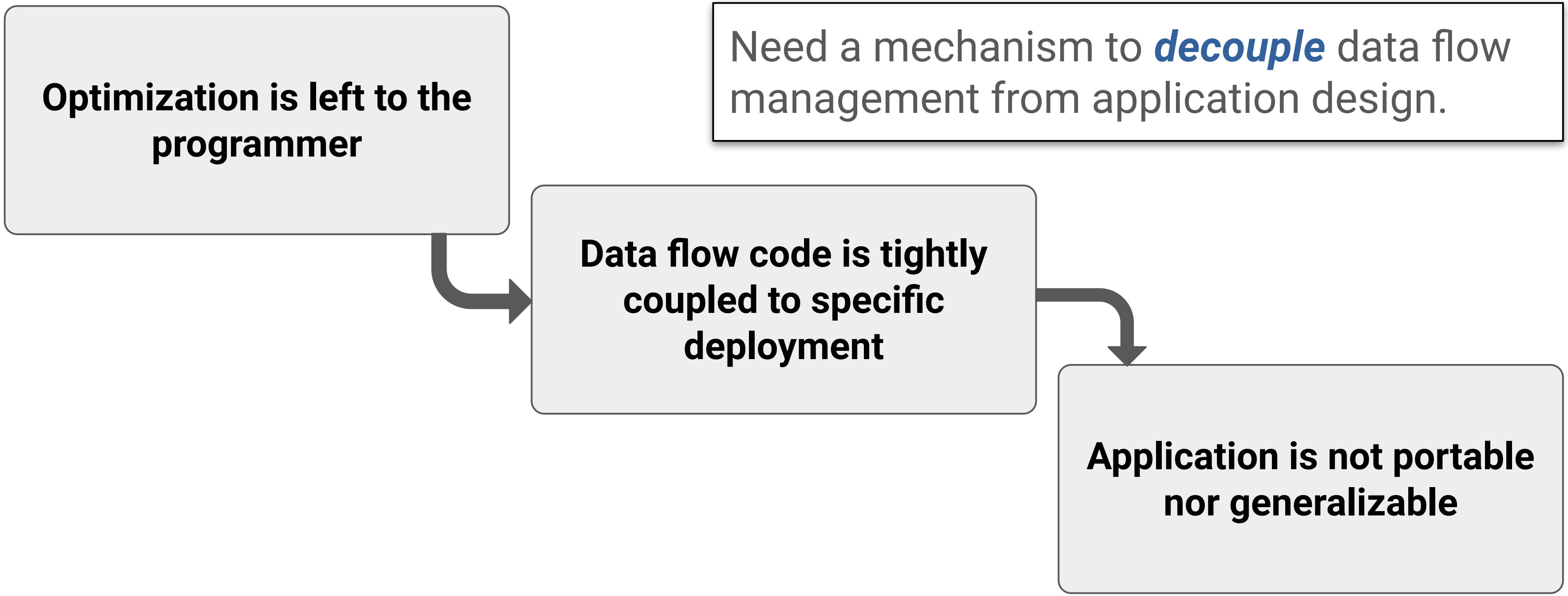
✓ Reliability and availability of cloud services

✗ Costs (time/money) increase with data flow due to ingress/egress

✓ Performance of workflow systems

✗ Restrictions are necessary to sustain “one-size-fits-all” approach

Managing Data Flow



Optimization is left to the programmer

Data flow code is tightly coupled to specific deployment

Application is not portable nor generalizable

Need a mechanism to ***decouple*** data flow management from application design.

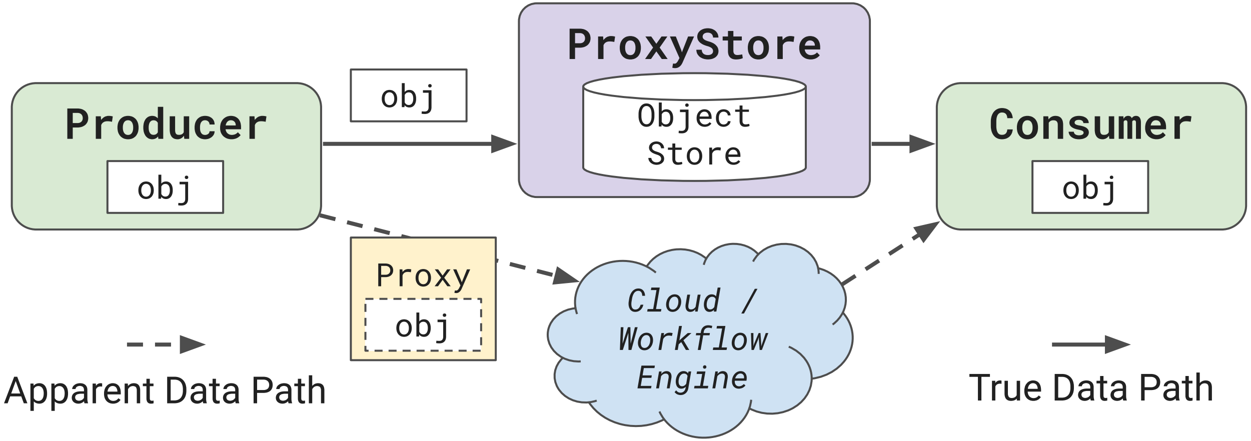
ProxyStore

A framework which abstracts the management and routing of data between processes in distributed and federated Python applications.

Goals

- Enable developers to ***focus on logical data flow*** rather than physical details of where data reside and how data are communicated.
- Dynamically select different data movement methods, depending on ***what*** data are moved, ***where*** data are moved, or ***when*** data are moved
- Transparently provide ***pass-by-reference*** semantics and ***just-in-time*** object resolution to consumers.

ProxyStore: Proxies + Object Stores



- Elegant ***pass-by-reference*** in distributed Python apps
- Mechanism for ***transparently*** decoupling control and data flow
- ***Abstract*** any (via plugins) object communication/storage

Concepts

Proxy + **Factory**

- Pass-by-reference
- Just-in-time, self-resolution

Store + **Connector**

- **Store**: high-level interface, used to create proxies
- **Connector**: low-level interface to ***mediated*** communication channel

Proxy Objects

- Transparently wrap ***target*** objects
- Acts like a wide-area ***reference***
- Initialized with a ***factory***
- ***Just-in-time*** resolution

```python
import numpy as np
from proxystore.proxy import Proxy

x = np.array([1, 2, 3])

# Proxy(Callable[[], T]) -> Proxy[T]
p = Proxy(lambda: x)

# A proxy is an instance of its wrapped object
assert isinstance(p, Proxy)
assert isinstance(p, np.ndarray)

# The proxy can do everything the numpy array can
assert np.array_equal(p, [1, 2, 3])
assert np.sum(p) == 6
y = x + p
assert np.array_equal(y, [2, 4, 6])
```

```python
from proxystore.connectors.redis import RedisConnector
from proxystore.store import Store

my_object = MyData(...)

with Store(
    name='my-store',
    connector=RedisConnector(‘localhost’, 6379),
    # other optional parameters
) as store:
    p = store.proxy(my_object)
```

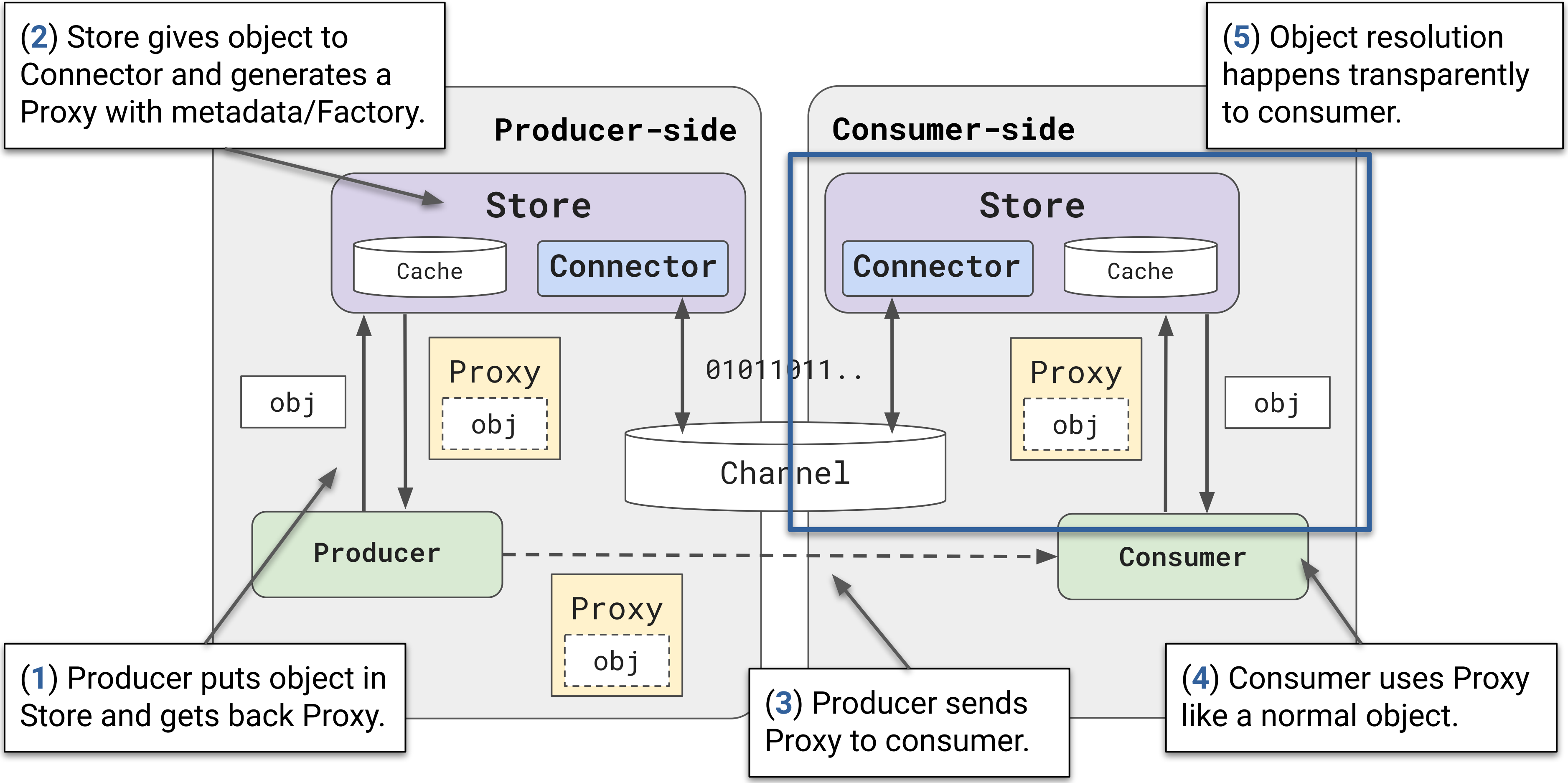
```python
from proxystore.proxy import Proxy

def my_function(x: MyData) -> ...:
    # Resolve of x deferred until use
    assert isinstance(x, MyData)
    # More computation...

assert isinstance(p, Proxy)
my_function(p)
```

Why lazy resolution with proxies?

- Performance (pass-by-reference, async resolve, skip unused objects)
- Avoid writing shims/wrapper functions
- Partial resolution of large objects with nested proxies
- Access control (only resolve data where permitted)



(1) Producer puts object in Store and gets back Proxy.

(2) Store gives object to Connector and generates a Proxy with metadata/Factory.

(3) Producer sends Proxy to consumer.

(4) Consumer uses Proxy like a normal object.

(5) Object resolution happens transparently to consumer.

Connectors

- Many **mediated** methods supported
- `Connector` = Python **`Protocol`**
- **`MultiConnector`**: Policy-based routing between instances

Protocol	Storage	Intra-Site	Inter-Site	Persistence
File	Disk	✓		✓
Redis/KeyDB	Hybrid	✓		✓
Margo	Memory	✓		
UCX	Memory	✓		
ZMQ	Memory	✓		
Globus	Disk		✓	✓
DAOS	Disk*	✓		✓
P2P Endpoint	Hybrid	✓	✓	✓

Examples

Intra-Site Communication with RDMA

Goal: Data-intensive workflows on HPC clusters

Idea: Leverage/aggregate local node storage

- Each node runs a storage server process
- Storage servers communicate via RDMA
- Elastic—storage processes spawned as proxies are propagated between nodes
- Downstream code unaware RDMA is being used

Polaris @ ALCF

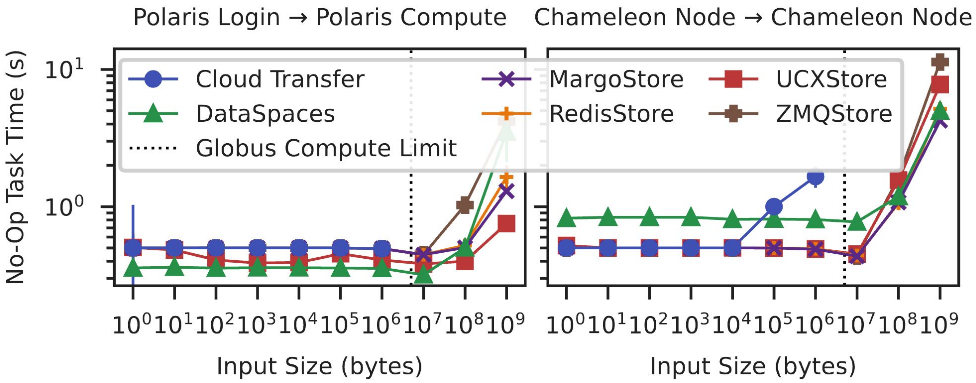
UCX-Py

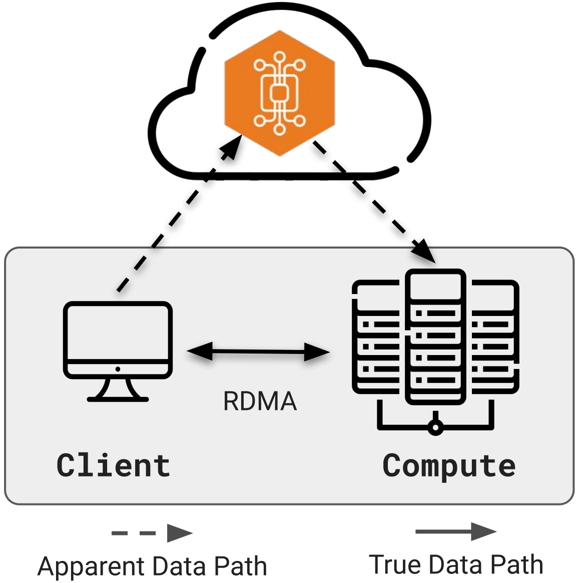
Mochi

ØMQ

Intra-Site Communication with RDMA

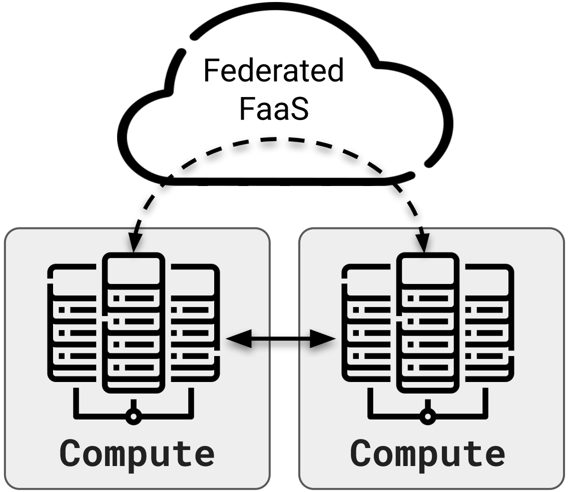
RDMA with Federated Functions as a Service

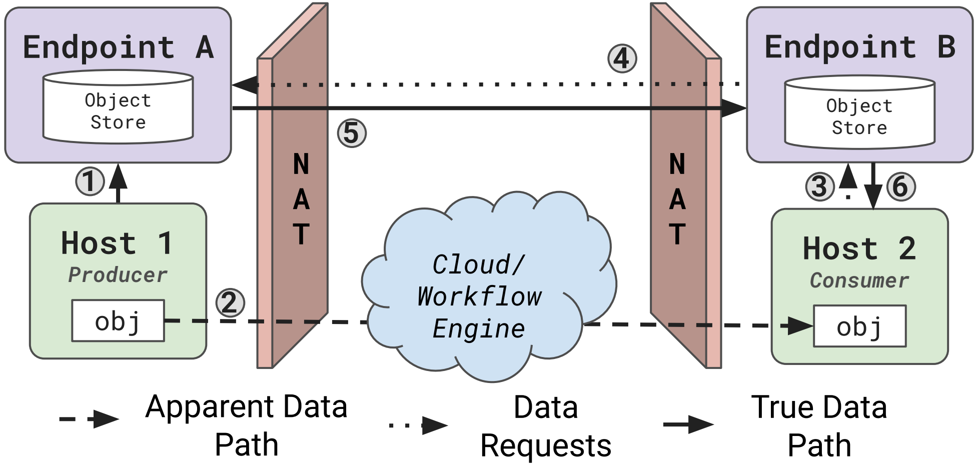




docs.proxystore.dev/main/guides/globus-compute
github.com/proxystore/benchmarks

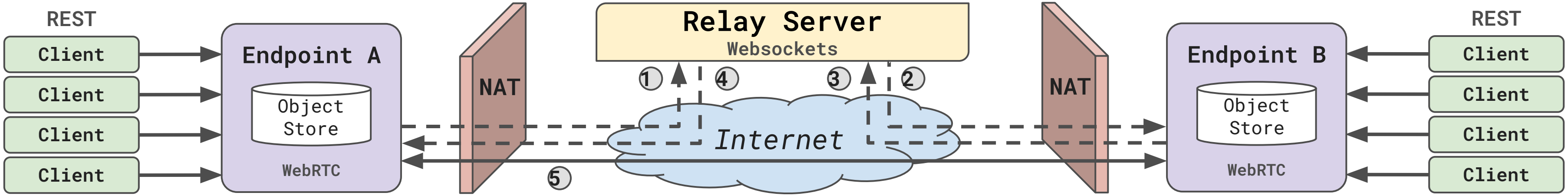
P2P Endpoints: Easy* Multi-Site Workflows





** Easy = no SSH tunnels/firewall restrictions, one-time setup, no cloud storage costs*

P2P Endpoints: UDP Hole-Punching



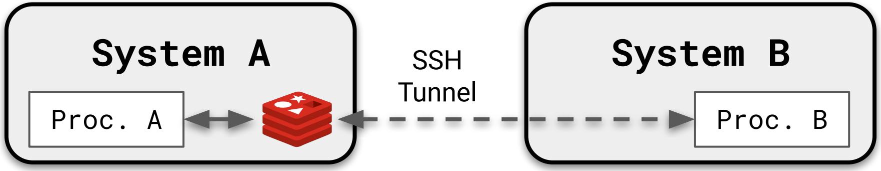
```
$ proxystore-endpoint configure example --relay-server wss://relay.proxystore.dev
$ proxystore-endpoint start example    # Runs as a daemon process
```

docs.proxystore.dev/main/guides/endpoints

P2P Endpoints: Benchmarks

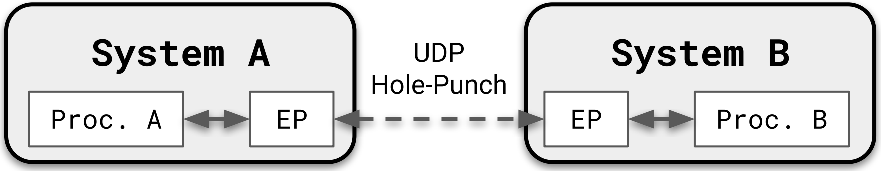
How to access shared data between multiple computing sites?

Redis + SSH



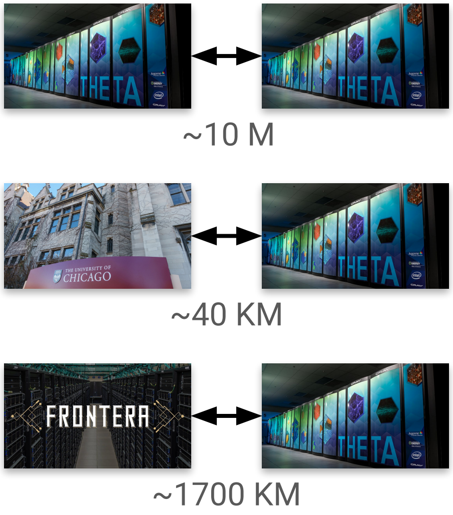
$O(n^2)$ SSH Tunnels (n systems)

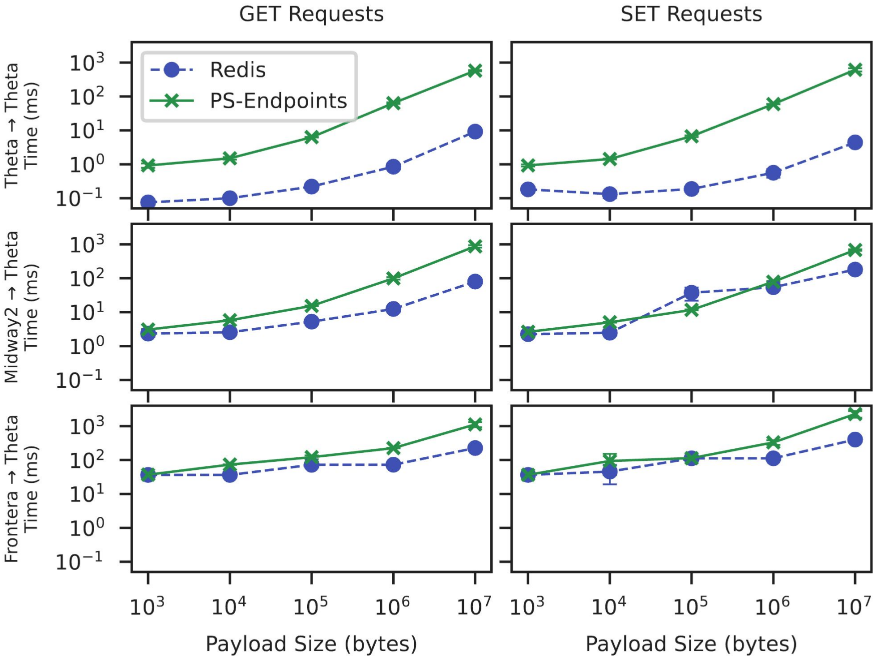
P2P Endpoints



$O(n)$ ProxyStore P2P Endpoints (n systems)

P2P Endpoints: Benchmarks

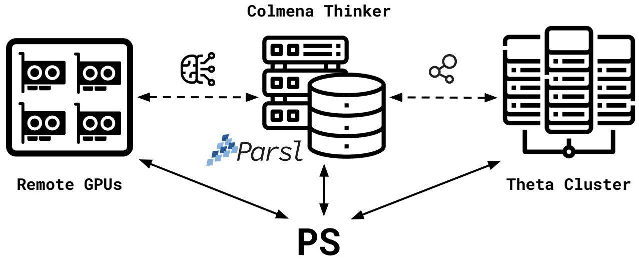




Increasing Distance = Better P2P Performance

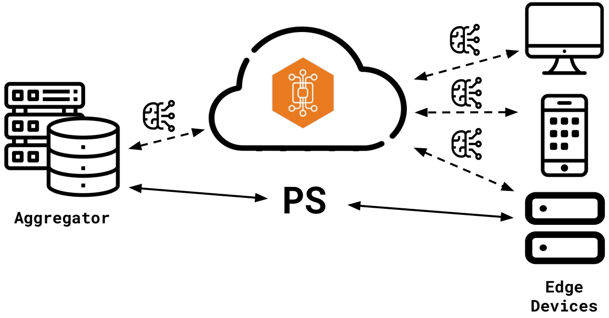
github.com/proxystore/benchmarks

Reducing Overheads in Science Applications

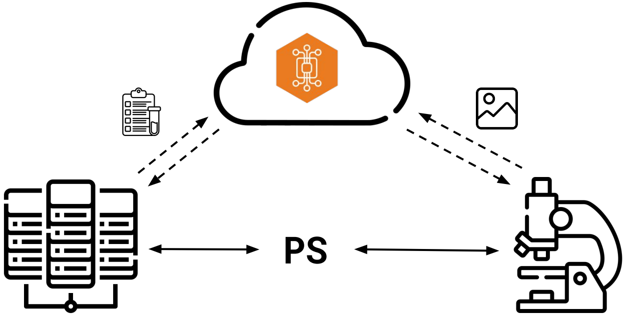


Multi-site Workflows

Federated Learning

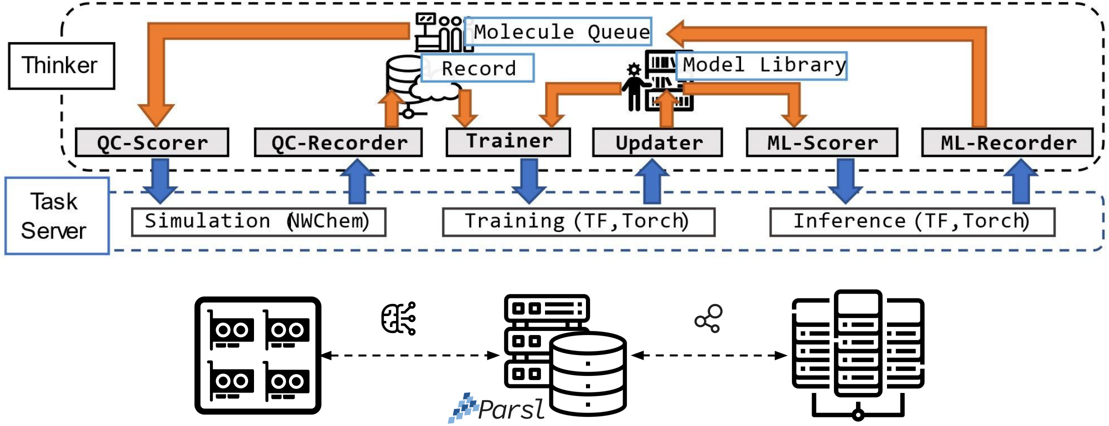


Real-time Science



Multi-site Active Learning

Science Goal: Use quantum chemistry simulations and surrogate ML models to efficiently identify electrolytes with high ionization potentials in a candidate set.



Thinker

Molecule Queue

Record

Model Library

QC-Scorer

QC-Recorder

Trainer

Updater

ML-Scorer

ML-Recorder

Task Server

Simulation (NWChem)

Training (TF,Torch)

Inference (TF,Torch)

Parsl

20 GPU Workstation
- Training Tasks
- Inference Tasks

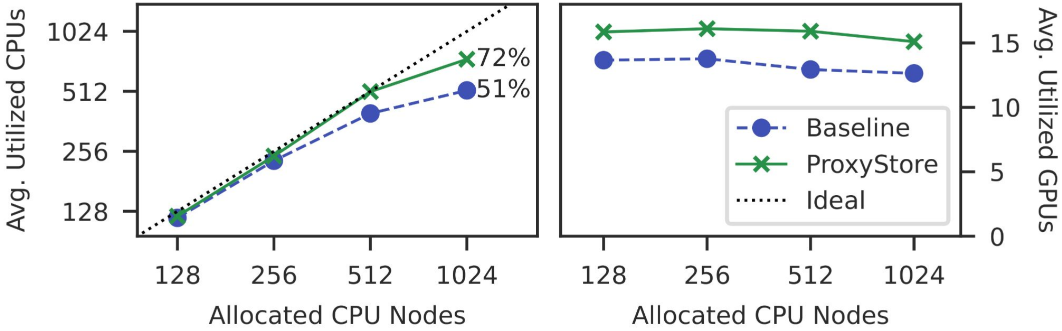
Workstation/Head Node
- Submit work
- Process results

1024 Theta KNL Nodes
- Simulation Tasks

Logan Ward, J. Gregory Pauloski, Valerie Hayot-Sasson, Ryan Chard, Yadu Babuji, Ganesh Sivaraman, Sutanay Choudhury, Kyle Chard, Rajeev Thakur, and Ian Foster. *Cloud services enable efficient AI-guided simulation workflows across heterogeneous resources*. In Heterogeneity in Computing Workshop at IPDPS. IEEE Computer Society, 2023.

Multi-site Active Learning

Systems Goal: Reduce task communication overheads in workflow system to increase system utilization and task throughput.

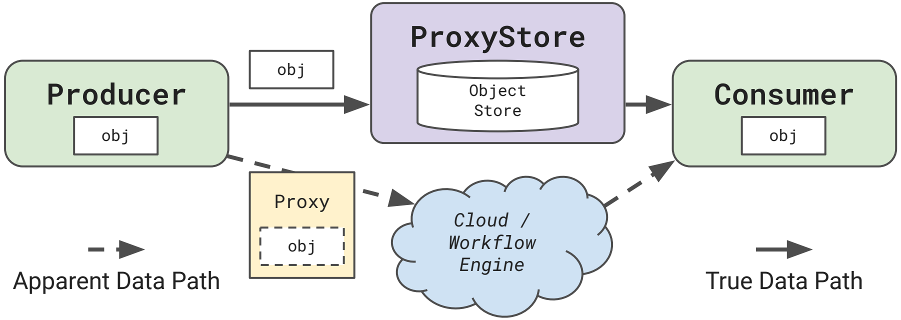


`MultiConnector` Configuration

- Simulation: Redis
- Training: ProxyStore P2P Endpoints
- Inference: Globus Transfer / ProxyStore P2P Endpoints

Takeaways

- Reduce overheads
- Re-used data only communicated once
- Orchestrator can choose ideal communication method
- No changes to task code needed



```
$ pip install proxystore[all]
$ pip install proxystore-ex
```

Questions?

Contact:
jgpauloski@uchicago.edu
github.com/proxystore/proxystore/issues

Publications:
docs.proxystore.dev/main/publications

Acknowledgements:

Funding

- Department of Energy (DOE) Contract DE-AC02-06CH11357
- ExaWorks project and ExaLearn Co-design Center of the Exascale Computing Project (17-SC-20-SC)
- NSF Grant 2004894

Compute

- Argonne Leadership Computing Facility
- Texas Advanced Computing Center
- National Energy Research Scientific Computing Center
- University of Chicago Research Computing Center
- Chameleon Cloud