H3: An Application-level, Low-overhead Object Store

HPC I/O in the Data Center Workshop (HPC-IODC)

July 2, 2021

Antony Chazapis	chazapis@ics.forth.gr
Efstratios Politis	epolitis@ics.forth.gr
Giorgos Kalaentzis	gkalaent@ics.forth.gr
Christos Kozanitis	kozanitis@ics.forth.gr
Angelos Bilas	bilas@ics.forth.gr

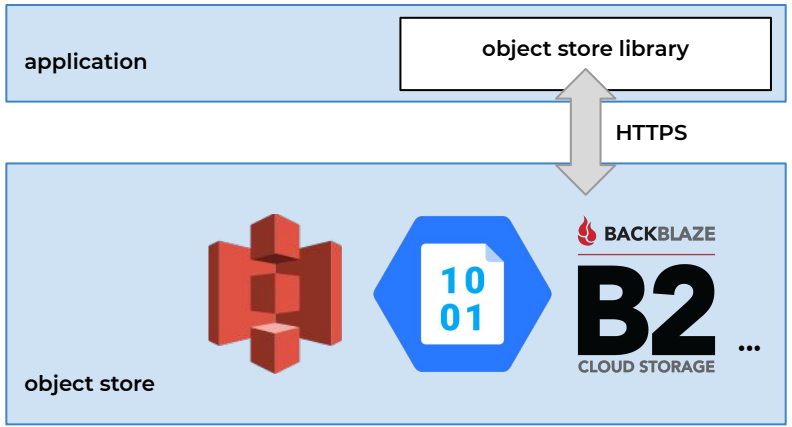
FORTH-ICS
Computer Architecture & VLSI Systems

Object stores

Most popular type of **storage-as-a-service**

Key points:

- Buckets that hold objects with no hierarchy
- Many Cloud offerings
  - HTTP-based RESTful APIs
  - No need for storage management
  - Unlimited capacity
- Popular solution for applications

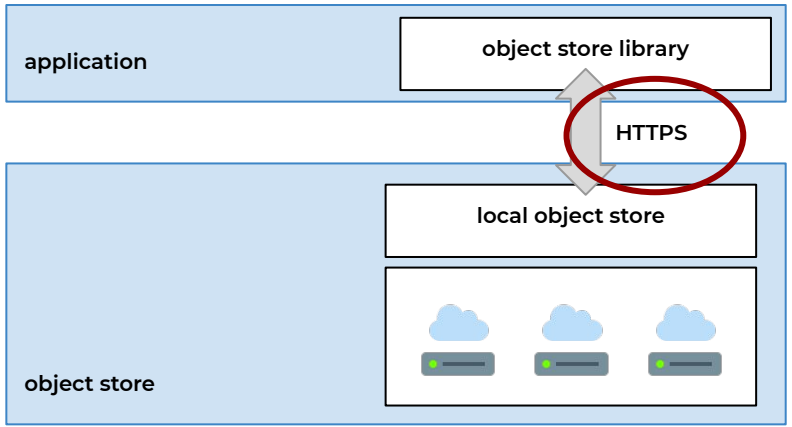


Multiple object stores currently available in the Cloud

Motivation

Issues:

- Run applications requiring an object store locally in an HPC environment
  - Avoid HTTP in the way
- Use storage as an interface between heterogeneous stages of the same workflow
  - Diverse software frameworks or languages
  - Different storage abstractions (files, objects)

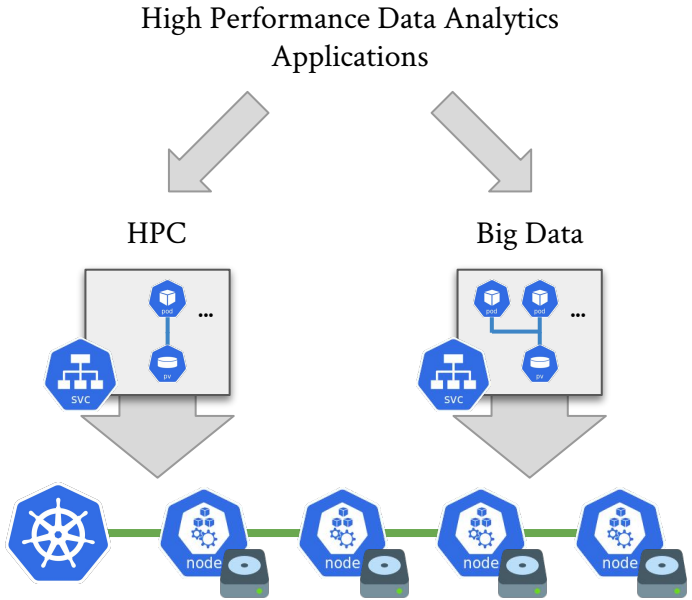


Avoid HTTP in the critical path

Motivation

Issues:

- Run applications requiring an object store locally in an HPC environment
  - Avoid HTTP in the way
- Use storage as an interface between heterogeneous stages of the same workflow
  - Diverse software frameworks or languages
  - Different storage abstractions (files, objects)

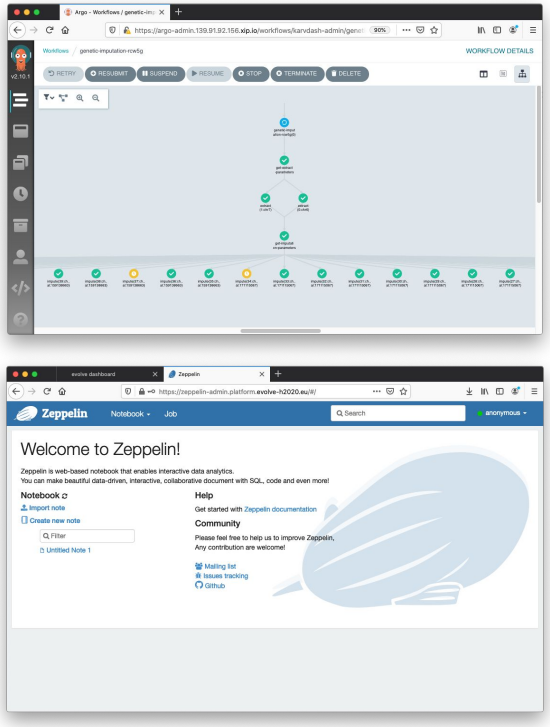


Containerize, separate software stack from hardware resources

Motivation

Issues:

- Run applications requiring an object store locally in an HPC environment
  - Avoid HTTP in the way
- Use storage as an interface between heterogeneous stages of the same workflow
  - Diverse software frameworks or languages
  - Different storage abstractions (files, objects)

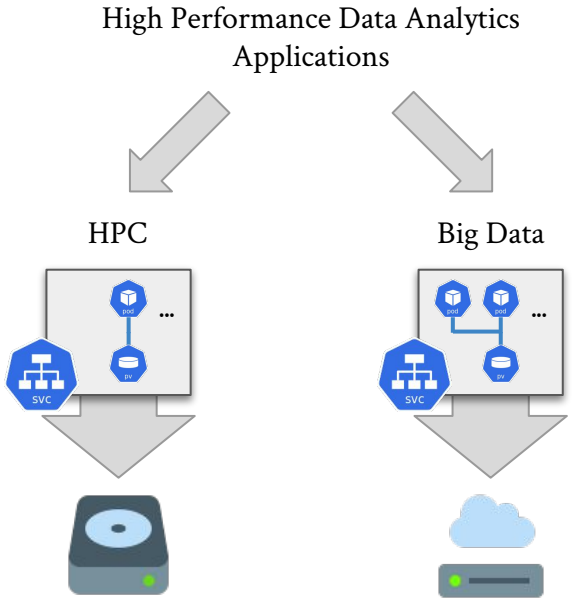


Workflows & Notebooks

Storage heterogeneity

Different requirements:

- HPC applications typically require a shared filesystem
- Big Data frameworks may use object stores
- Workflows exchange "artifacts"

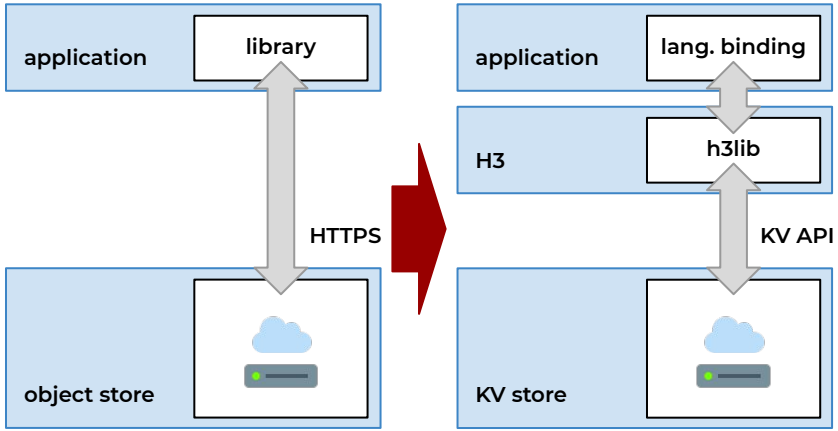


Different storage requirements

H3: An embedded object store

Key points:

- H3 is an embedded object store (library)
- API calls for objects converted to KV operations implemented by plug-ins
- Plug-ins for filesystem, Kreon, Redis, RocksDB
- Clean API with Python and Java bindings
- CLI for management
- FUSE-based filesystem for file semantics
- S3proxy for S3 compatibility

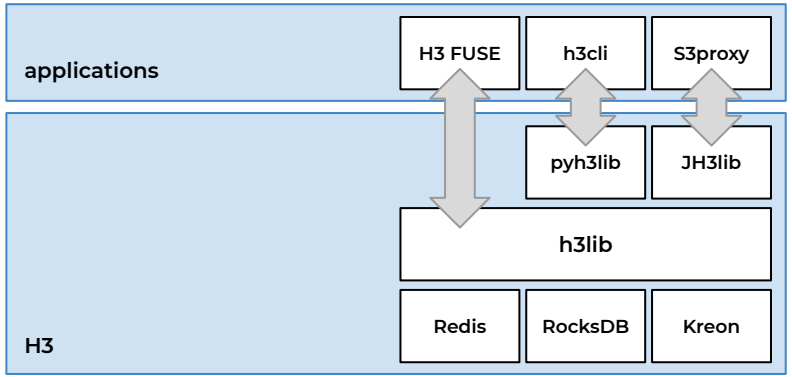


Moving the object store in the application

H3: An embedded object store

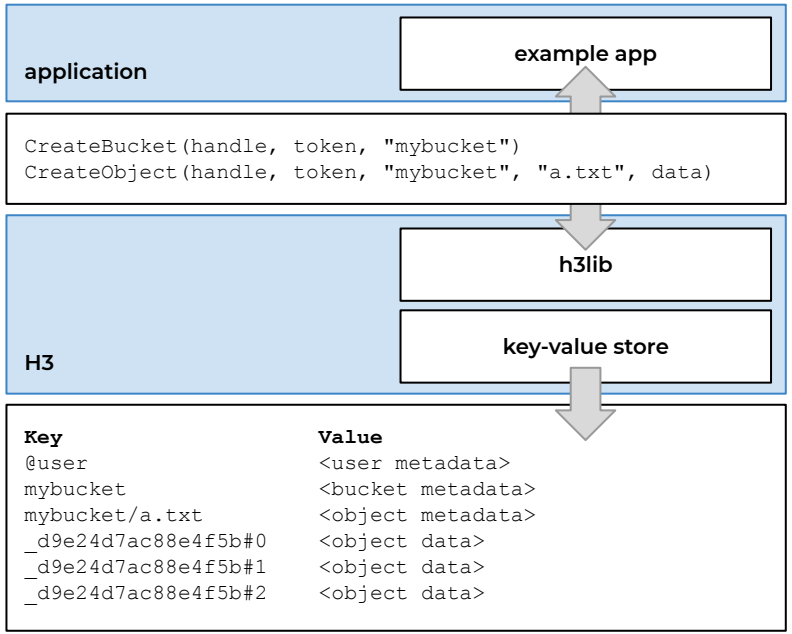
Key points:

- H3 is an embedded object store (library)
- API calls for objects converted to KV operations implemented by plug-ins
- Plug-ins for filesystem, Kreon, Redis, RocksDB
- Clean API with Python and Java wrappers
- CLI for management
- FUSE-based filesystem for file semantics
- S3proxy for S3 compatibility



H3 components

Key-value translation



Example of H3 object to key mapping

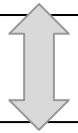
Language bindings and CLI

Python

```
from pyh3lib import H3

h3 = H3('redis://127.0.0.1:6379')
h3.create_bucket('mybucket')
h3.create_object('mybucket', 'a.txt', data)

h3.list_objects('mybucket') # Returns ['a.txt']
```



Bash

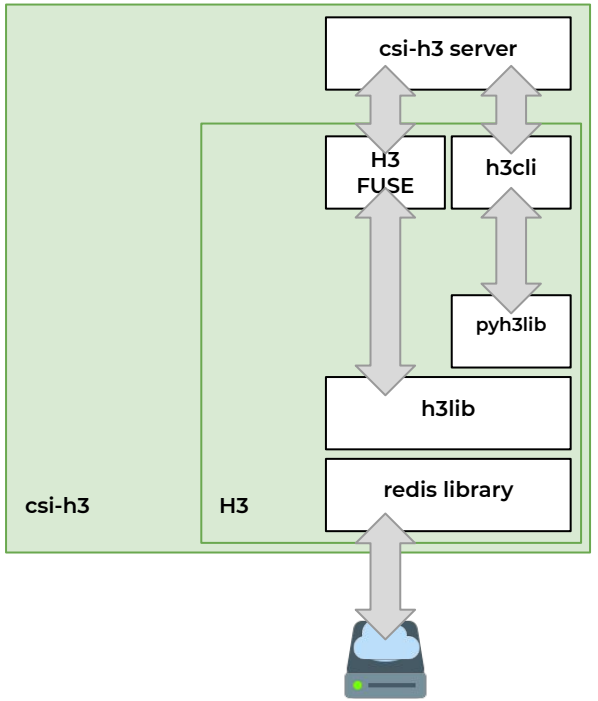
```
# h3cli --storage "redis://127.0.0.1:6379" mb h3://mybucket
# h3cli --storage "redis://127.0.0.1:6379" cp a.txt
h3://mybucket/a.txt
# h3cli --storage "redis://127.0.0.1:6379" ls h3://mybucket
a.txt
#
```

Example of H3 object to key mapping

Integration with Kubernetes

Key points:

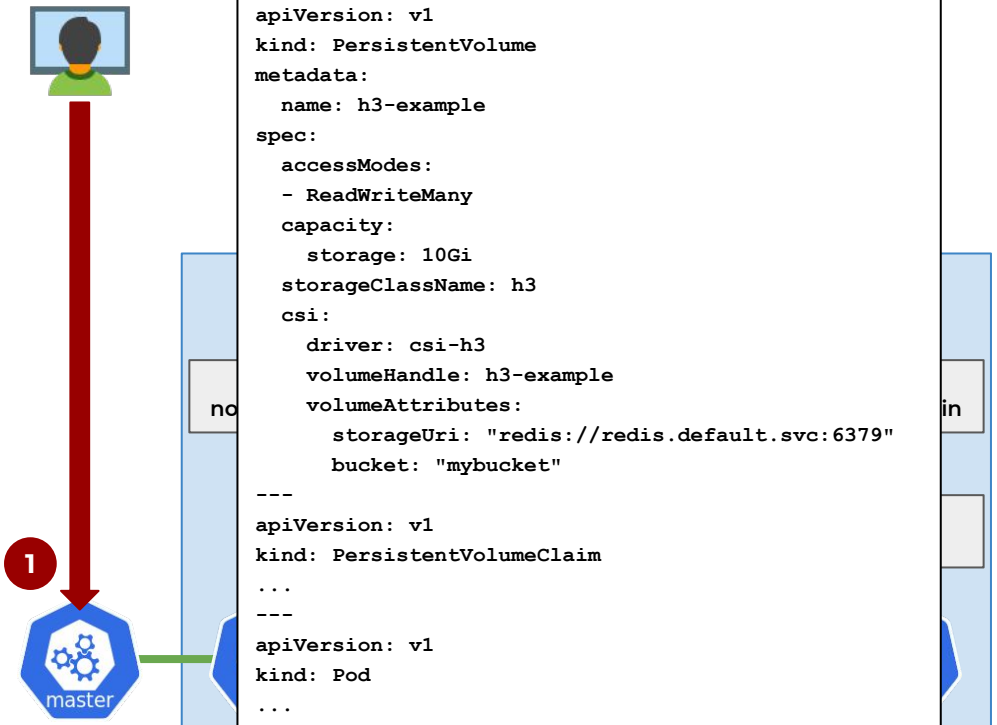
- Implementation of a Container Storage Interface (CSI) plugin
- Easy provisioning of storage to Kubernetes containers via H3 FUSE
- Single controller and nodeplugin
- Based on available H3 container



CSI-H3 container

CSI flow

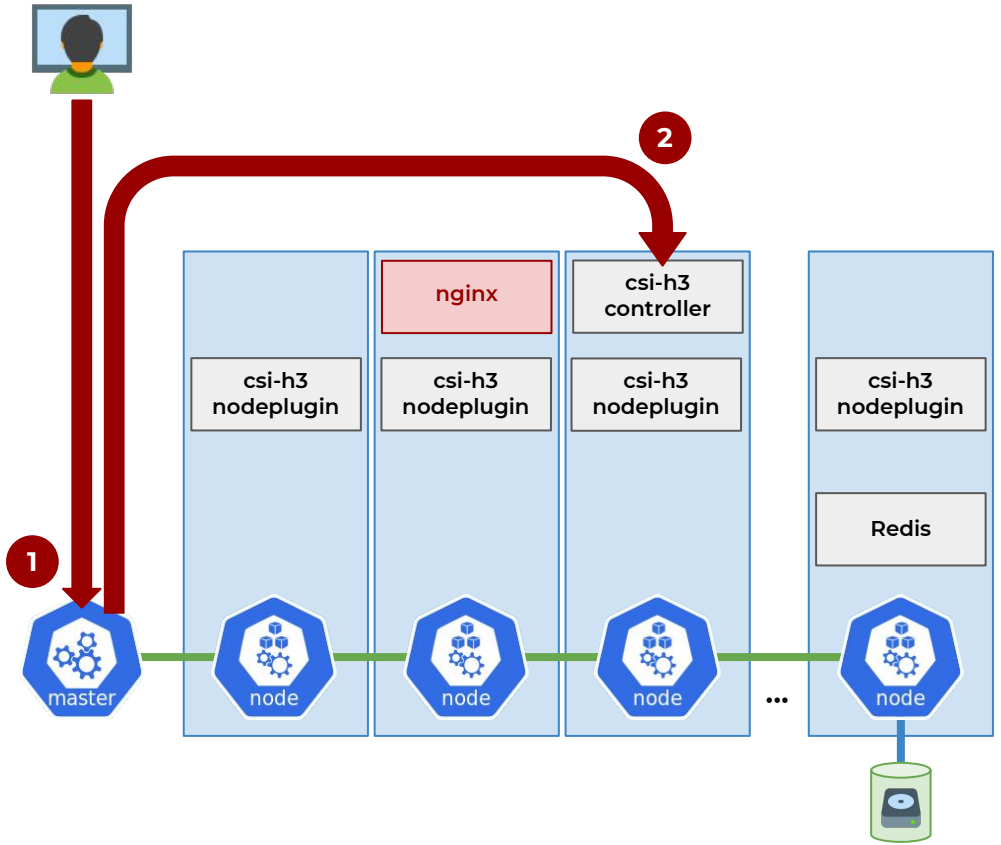
1 deploy service




```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: h3-example
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  storageClassName: h3
  csi:
    driver: csi-h3
    volumeHandle: h3-example
    volumeAttributes:
      storageUri: "redis://redis.default.svc:6379"
      bucket: "mybucket"
---
apiVersion: v1
kind: PersistentVolumeClaim
...
---
apiVersion: v1
kind: Pod
...
spec:
  ...
  volumes:
  - name: h3-example
    persistentVolumeClaim:
      claimName: h3-example
```

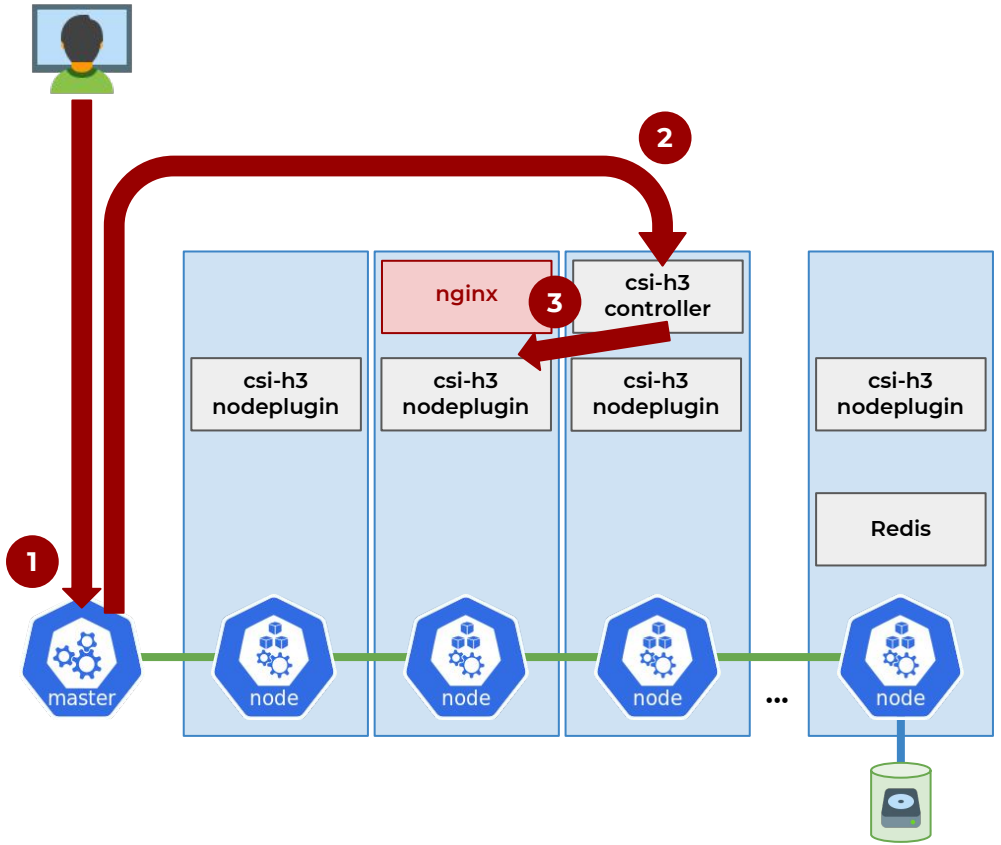
CSI flow

1. deploy service
2. need volume



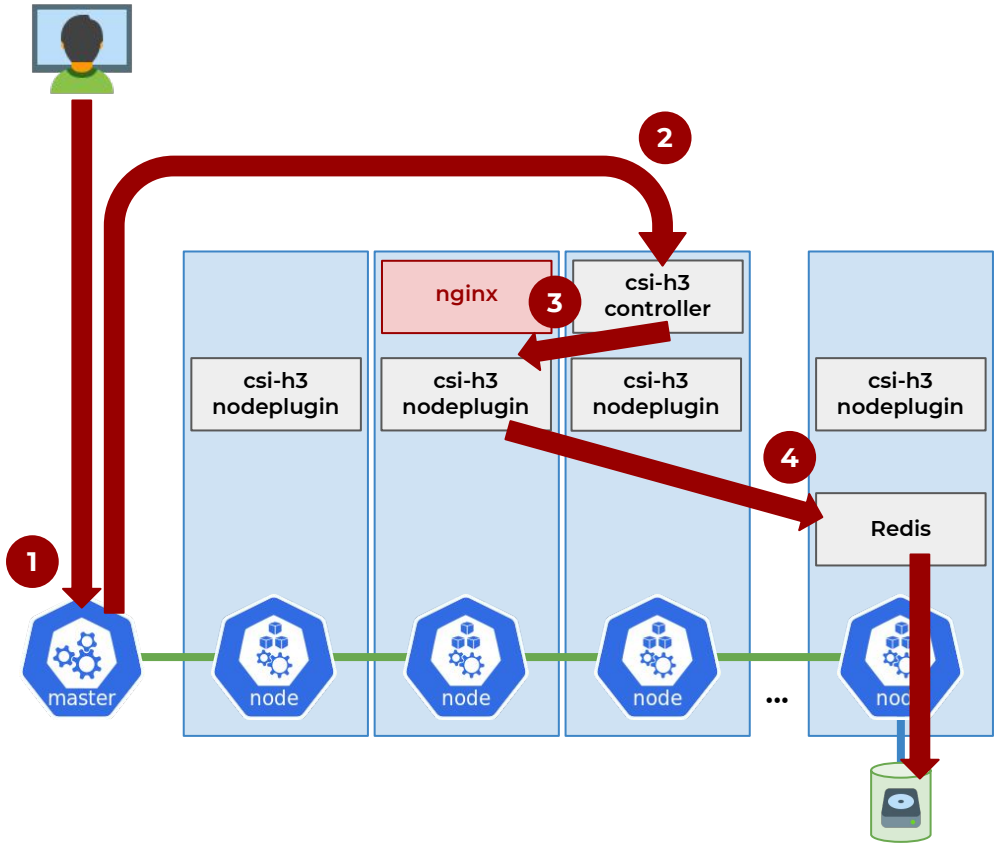
CSI flow

1. deploy service
2. need volume
3. create volume



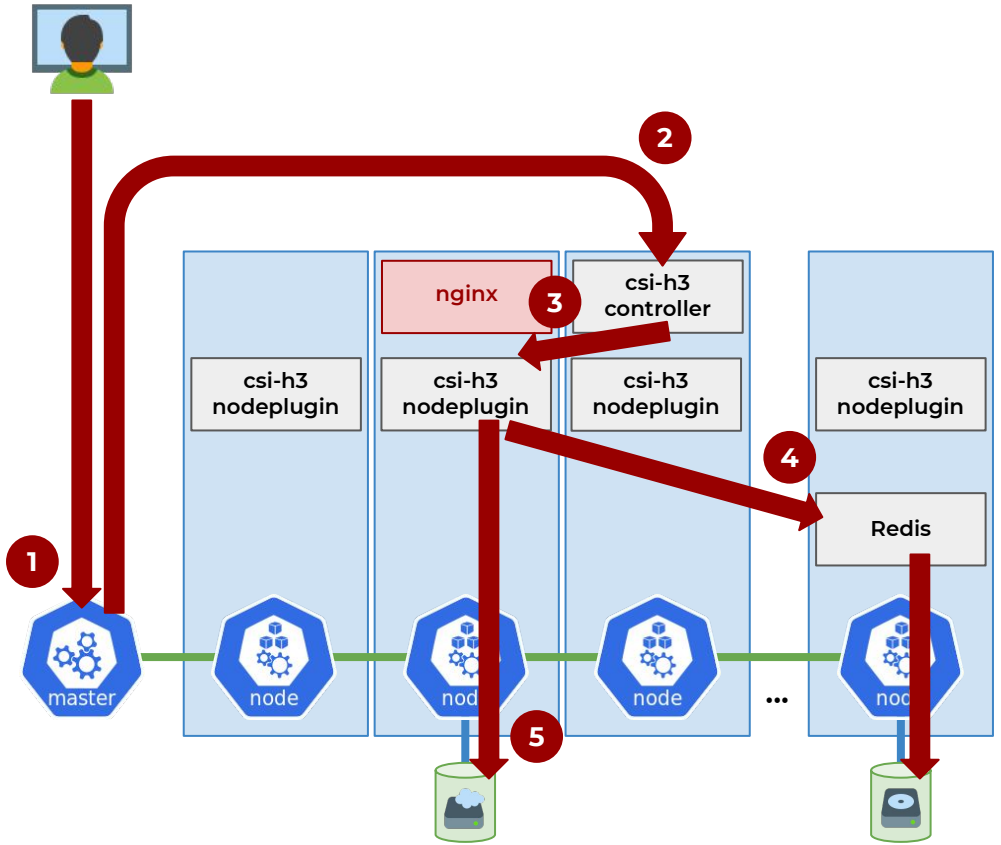
CSI flow

1. deploy service
2. need volume
3. create volume
4. create bucket and mount



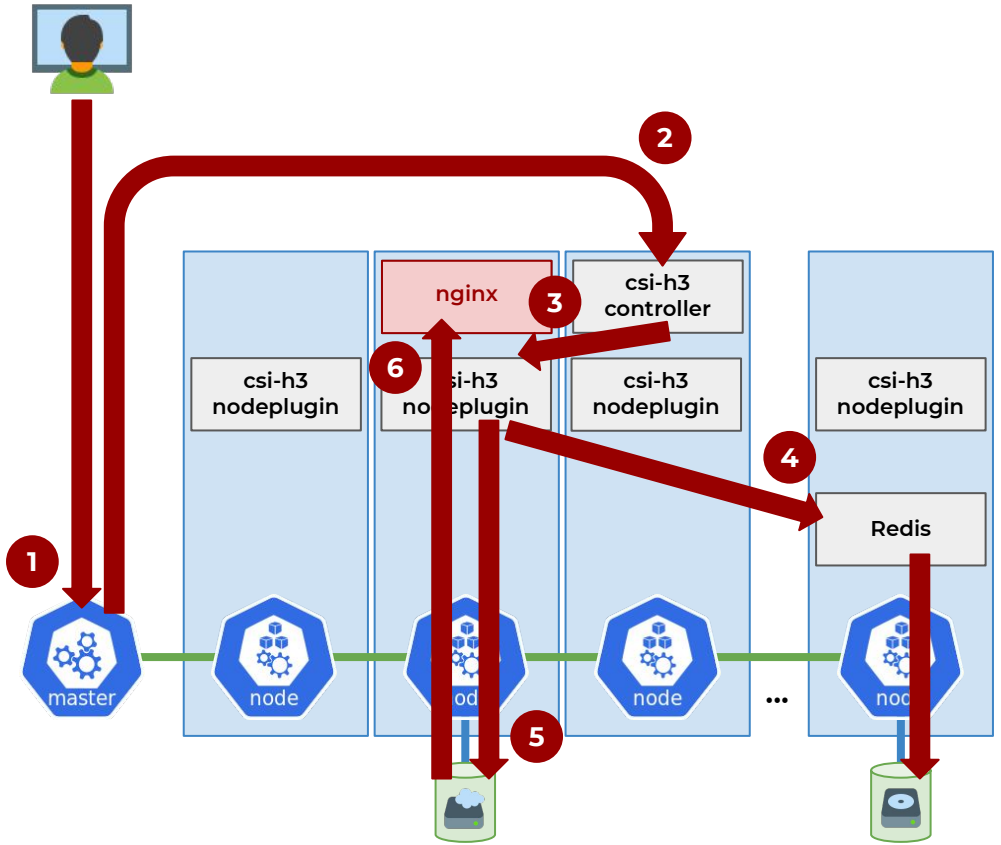
CSI flow

1. deploy service
2. need volume
3. create volume
4. create bucket and mount
5. mount volume to host

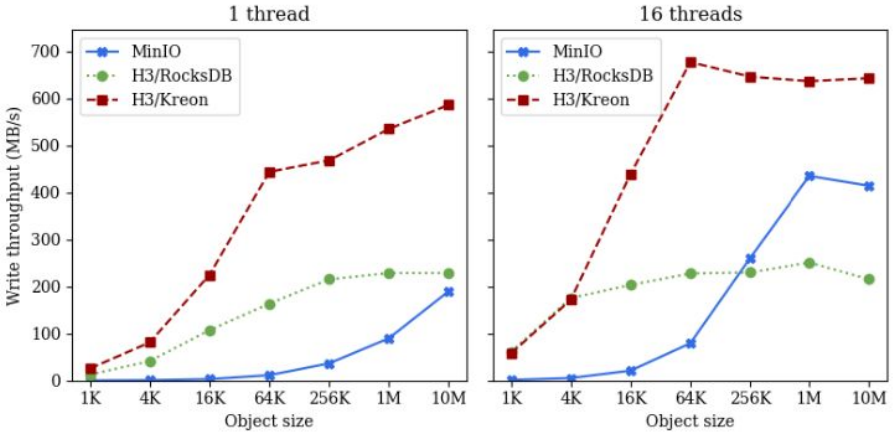


CSI flow

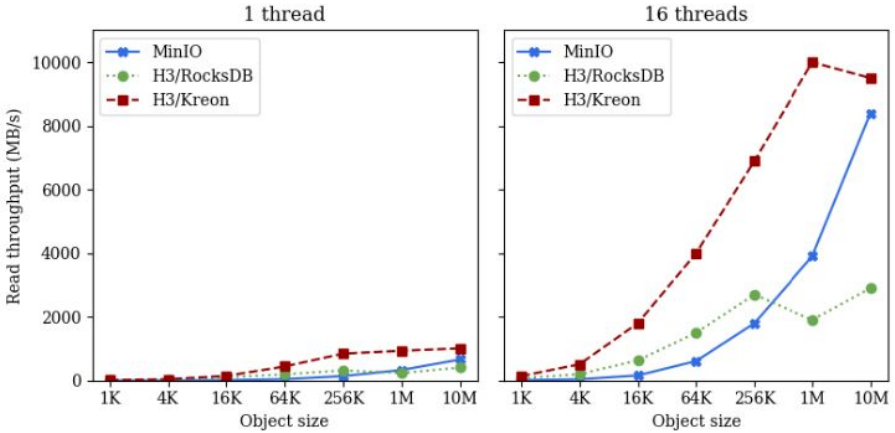
1. deploy service
2. need volume
3. create volume
4. create bucket and mount
5. mount volume to host
6. attach volume to container



Evaluation: Single-node setup

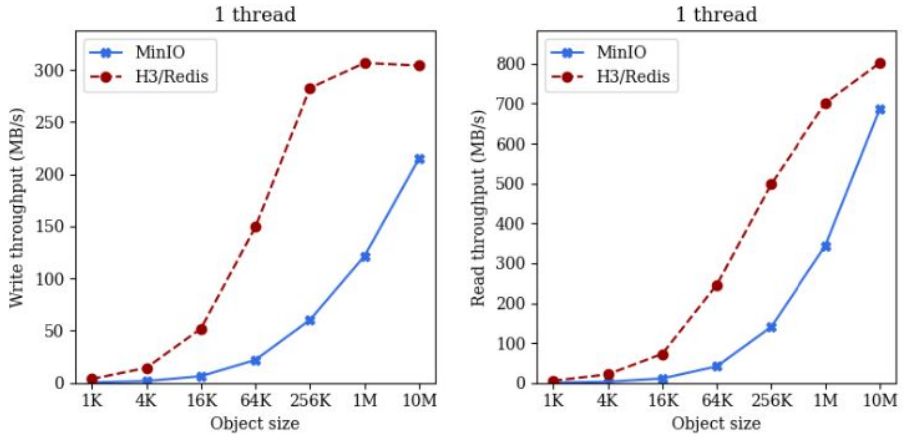


(a) Put operations



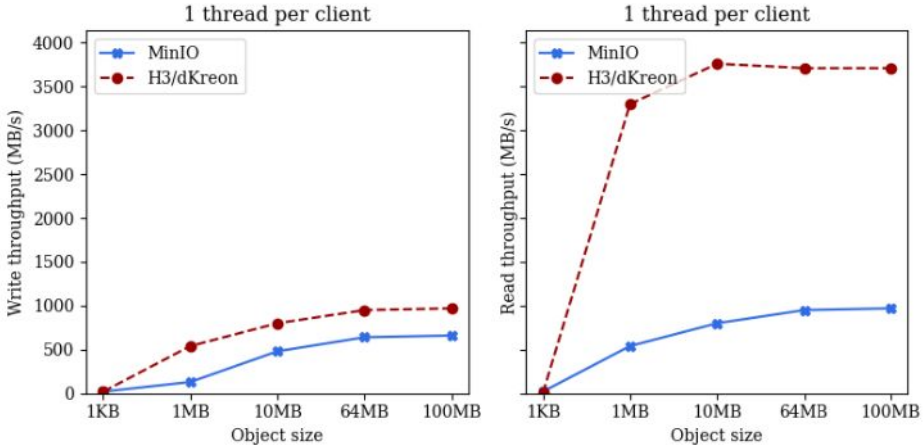
(b) Get operations

Evaluation: Single-node setup

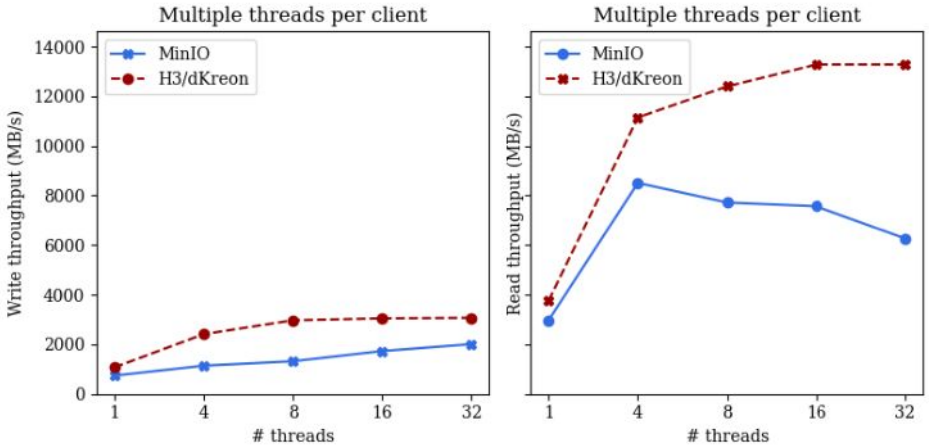


(a) Put operations (b) Get operations

Evaluation: Multi-node setup



(a) Varying value sizes



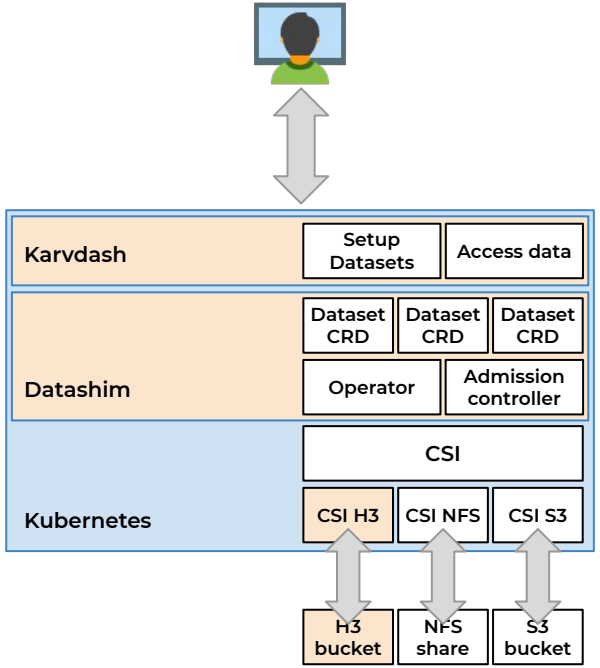
(b) Varying number of threads

Ongoing and future work

H3 is now part of the Unified Storage Layer for Kubernetes (presented at the CHEOPS workshop of EuroSys 2021)

Future directions:

- Additional key-value back ends
- Extended attributes and controllers
- Plug-ins for programming frameworks



USL components

Thank you

Links:

- H3:

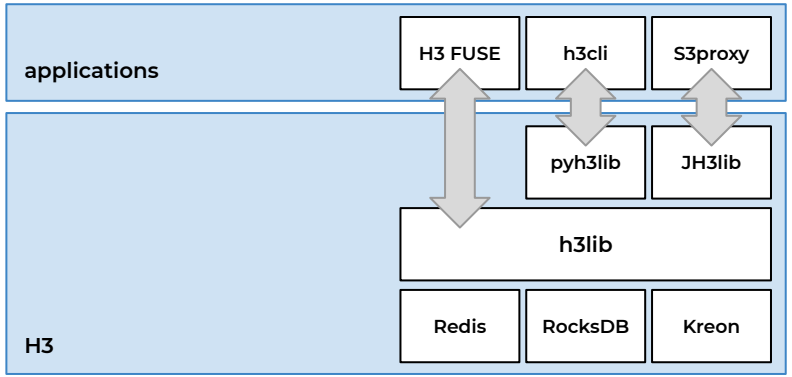
  https://github.com/CARV-ICS-FORTH/H3
- CSI driver for H3:

  https://github.com/CARV-ICS-FORTH/csi-h3
- Performance test for H3:

  https://github.com/CARV-ICS-FORTH/h3-benchmark
- Kreon key-value store:

  https://github.com/CARV-ICS-FORTH/kreon
- EVOLVE:

  https://www.evolve-h2020.eu



USL components