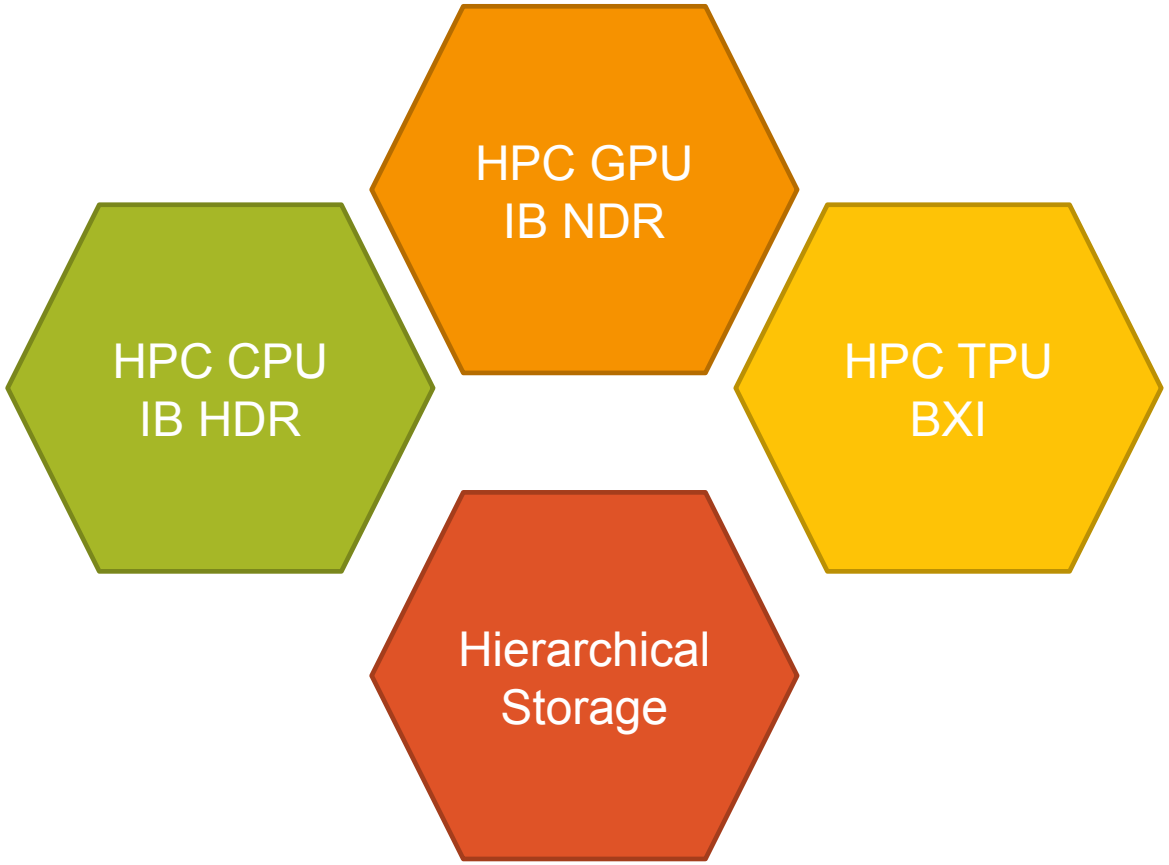
Context & Challenges

Modular SuperComputer Architecture

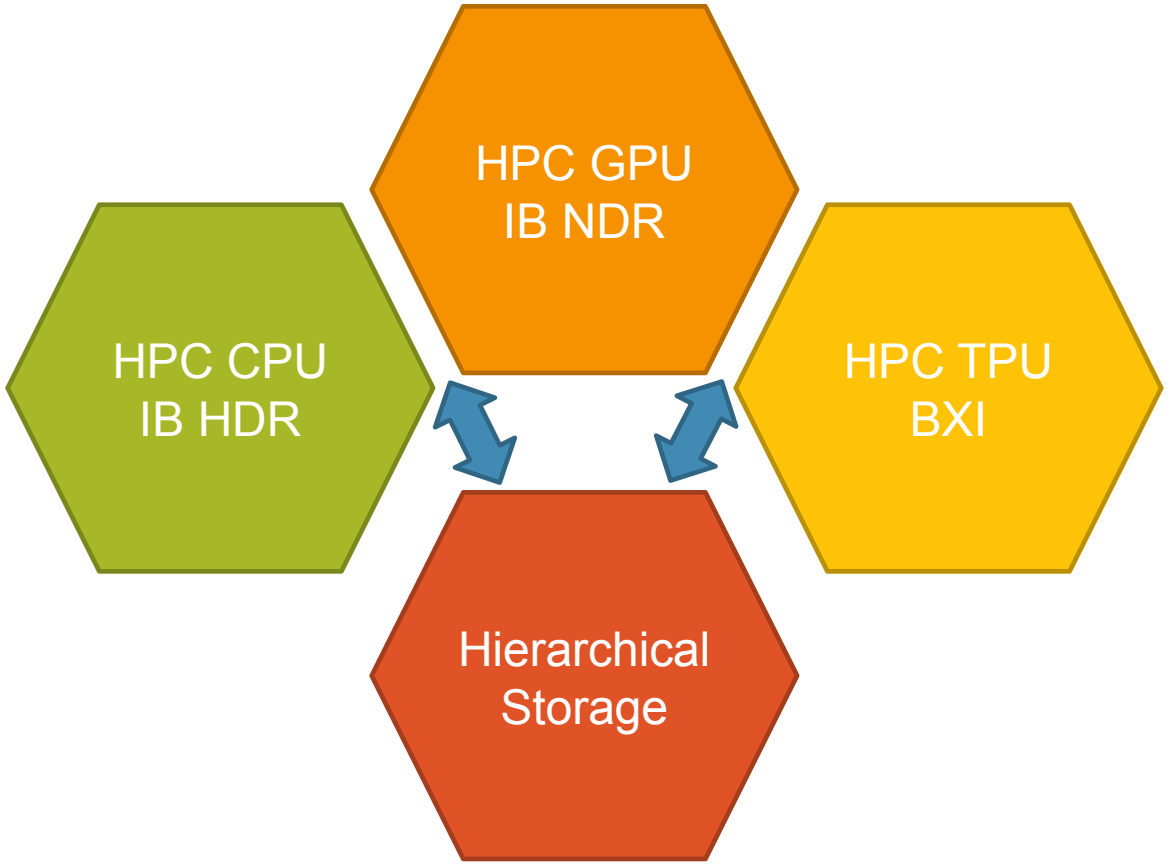
- Heterogeneous compute modules
- « Long Term » Storage module with multiple technologies
  - Object based
  - Hierarchical storage
- Low/medium bandwidth, high latencies connections between modules



Data Movements

- Heterogeneous workflows
- Main processing steps on HPC implies data movement

- Challenges :
  - **Organize data**
  - **Limit data movements**
  - **Give control to users**



The IO-SEA Proposal

Exploratory work…

Datasets & Namespaces

- **Datasets** are **data containers** hosting objects
  - No data organization, just collections of objects…
  - Could be seen as ***private*** file systems or object stores

- Objects in Datasets are organized with **Namespaces**
  - Different Namespaces can be created in each Dataset
  - Namespaces can expose the same objects in the dataset through different protocols
    - POSIX, S3, …

Datasets & Namespaces

A Dataset is stored as 1 S3 Bucket

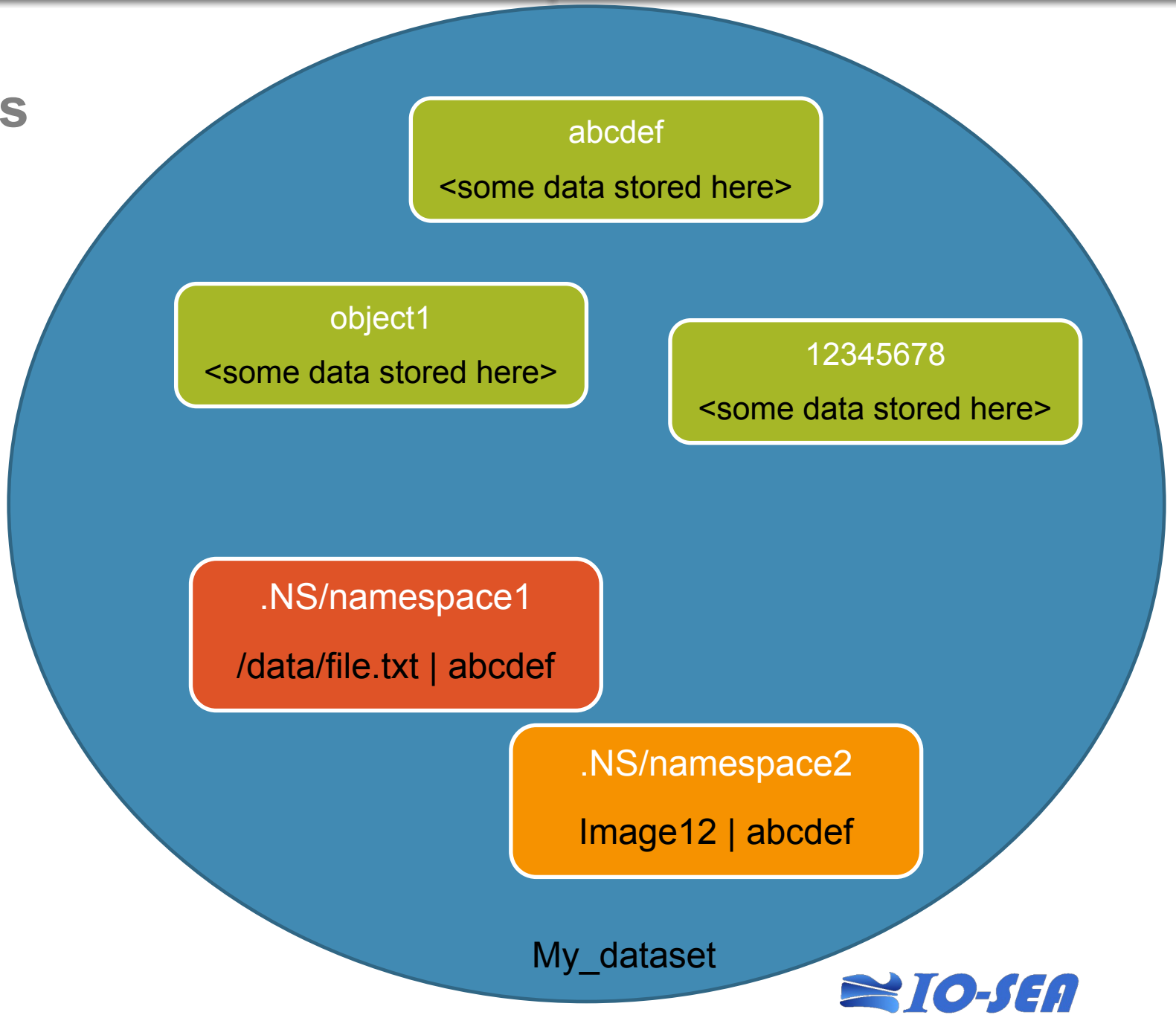
Files stored as S3 object

Namespaces stored in « metadata » objects

Think of it as a set of directories and symbolic links to objects in dataset

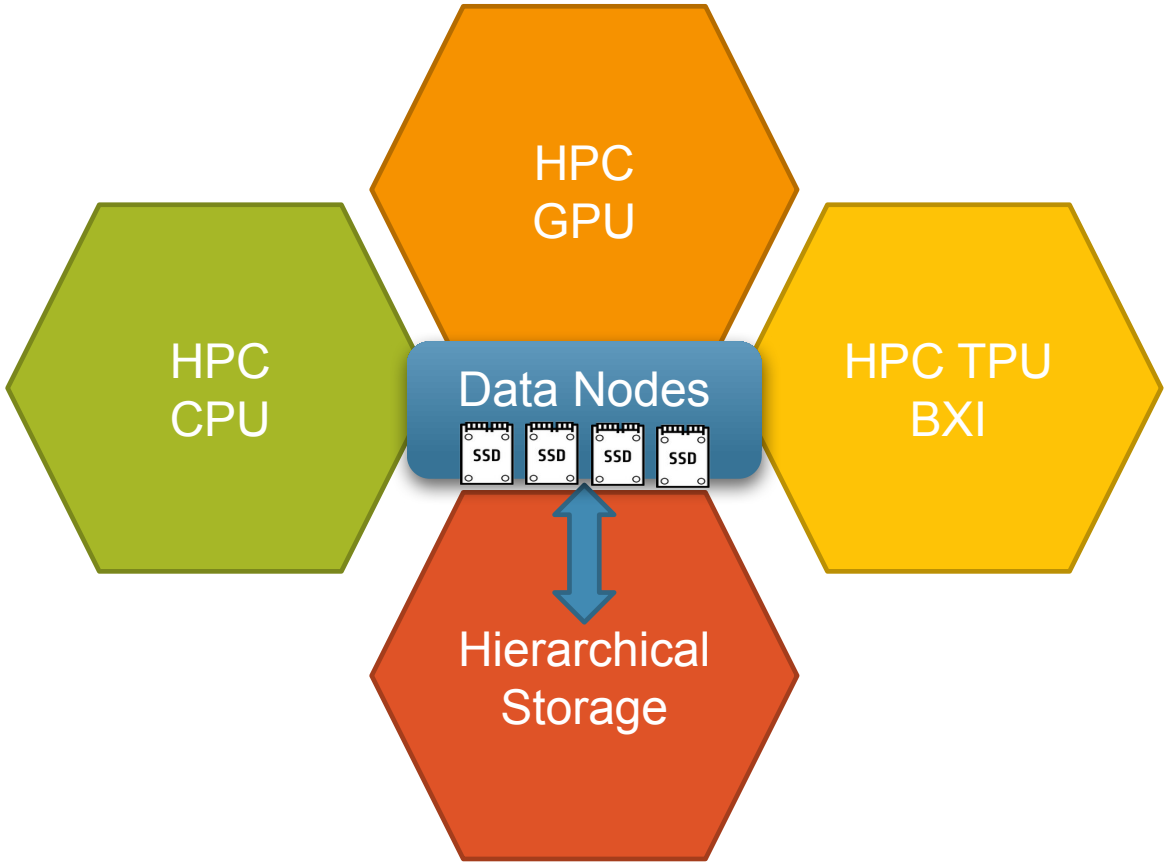
.NS/namespace1 : POSIX

.NS/namespace2 : S3



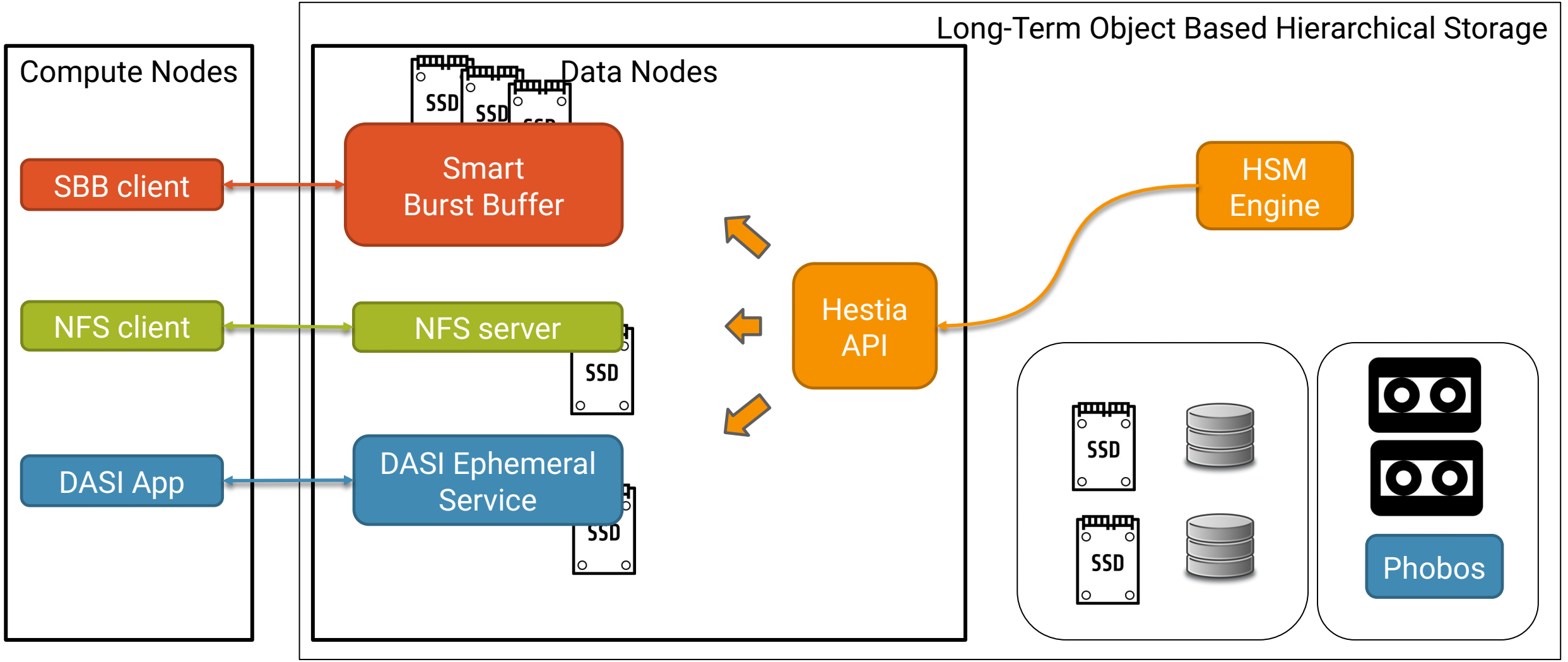
Data nodes for High Speed Access from HPC workflows

- Runs Ephemeral Services for a workflow **close to compute nodes**, enabling access to datasets through namespaces
  - Connected on HPC interconnects, enabling RDMA transfer speeds
- Multiple Data Nodes in parallel to reach target bandwidth
  - For some ephemeral services



IO-SEA

Ephemeral I/O Services, to expose a namespace to clients
Hestia API, as the main interface to Hierarchical Storage



A USER'S TOUR OF THE IO-SEA ENVIRONMENT

DEEP-SEA Seminar

1 March 2024 | Eric B. Gregory (JSC) & Philippe Couvee (Eviden) |

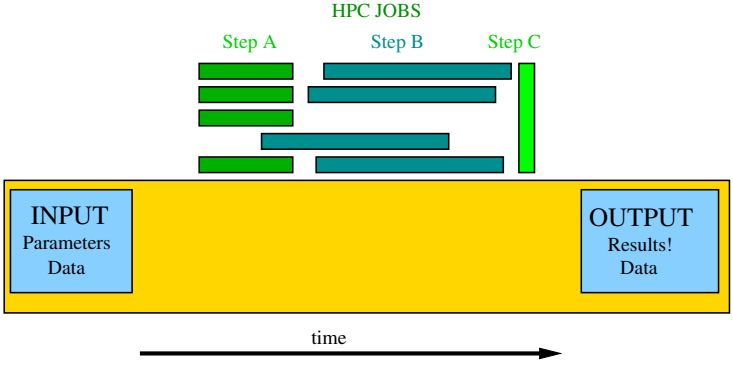
THE PLAN

- The workflow
- Setting up an IO-SEA Workflow
- Choosing ephemeral services
- The Workflow Description File
- Running a workflow
- Results

THE WORKFLOW

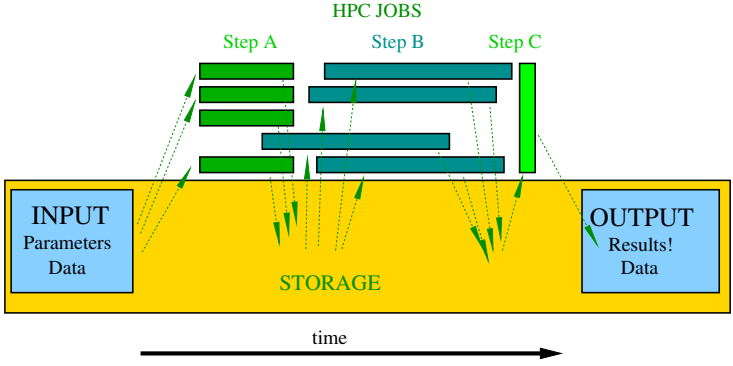
- The user's interaction with the IO-SEA system is through the concept of *WORKFLOWS*.

THE WORKFLOW



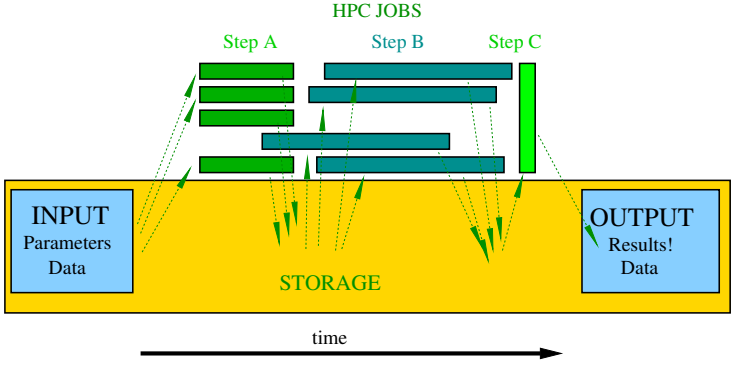
- The user's interaction with the IO-SEA system is through the concept of *WORKFLOWS*.
- Workflows are composed of *STEPS* — classes of HPC jobs.

THE WORKFLOW



- The user's interaction with the IO-SEA system is through the concept of *WORKFLOWS*.
- Workflows are composed of *STEPS* — classes of HPC jobs.
- Workflow steps generally exchange data through the storage system.

THE WORKFLOW



- The user's interaction with the IO-SEA system is through the concept of *WORKFLOWS*.
- Workflows are composed of *STEPS* — classes of HPC jobs.
- Workflow steps generally exchange data through the storage system.

In the IO-SEA environment, workflow steps can take advantage of *ephemeral storage services* to facilitate data movement between jobs and in the storage hierarchy.

SBB EPHEMERAL SERVICE

SBB — Smart Burst Buffer

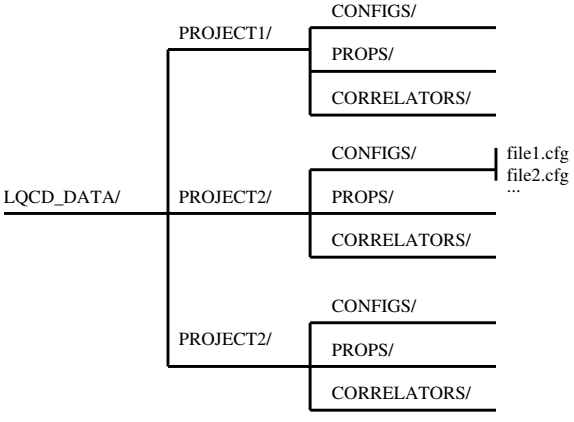
✓

- Provides I/O optimization.
- Requires no code or input modification.
- Can be applied to existing directory structure.
- Minimizes data traffic to and from disk during workflow.
- Optimizes “data re-use”.
- Can isolate workflow from a busy storage system.

X

- No data-management benefits.
- No datasets.
- Limited RAM resources on datanodes.

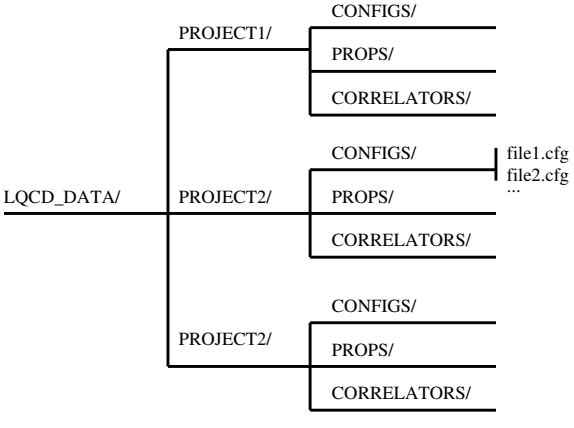
THE SMART BURST BUFFER EPHEMERAL SERVICE



THE SMART BURST BUFFER EPHEMERAL SERVICE

Specify a target directory, e.g.:

`<full-path-to>/LQCD_DATA/PROJECT2/`

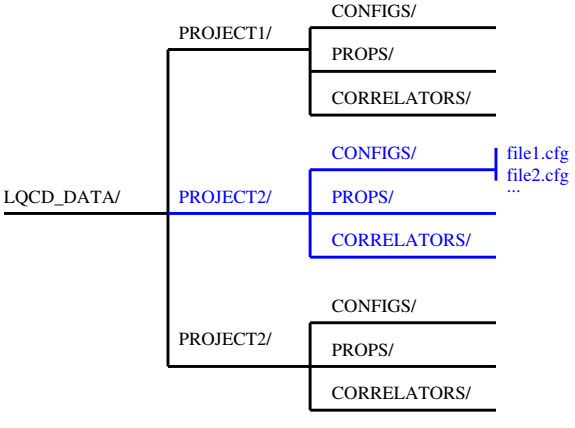


THE SMART BURST BUFFER EPHEMERAL SERVICE

Specify a target directory, e.g.:

`<full-path-to>/LQCD_DATA/PROJECT2/`

Data *reads* and *writes* in target and its sub-directories are intercepted by the SBB.



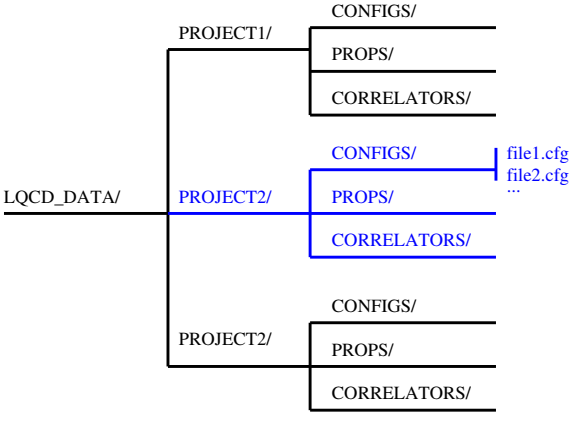
THE SMART BURST BUFFER EPHEMERAL SERVICE

Specify a target directory, e.g.:

`<full-path-to>/LQCD_DATA/PROJECT2/`

Data *reads* and *writes* in target and its sub-directories are intercepted by the SBB.

Output is stored in fast RAM or NVMe storage, ready for re-use or to be flushed to disk when convenient.



NFS EPHEMERAL SERVICE

- Files are stored in a “dataset” — private filesystem
- Provides I/O optimization.
- Requires no code modification.
- Minimizes data traffic to and from disk during workflow.
- Optimizes “data re-use”.
- Can isolate workflow from a busy storage system.
- Allows automated migration through storage hierarchy

- Cannot access data outside of a workflow session.
- Parallel I/O (many nodes $\longleftrightarrow$ one file) not currently possible (NFS is not fully Posix compliant).

DASI EPHEMERAL SERVICE & INTERFACE

- Stores data in dataset.
- Data accessible by sematic keys (“`beta=3.6,mass=0.2,...`”).
- User defines key schema.
- Can be accessed with C++ or Python API directly from the application.
- CLI exist for post- or pre-run archiving/extracting.

- Cannot access data outside of a workflow session.
- Requires code modification for ’native use’ (for C or python).

HOW DO WE DO IT?

To move an existing workflow into the IO-SEA environment we need:

HOW DO WE DO IT?

To move an existing workflow into the IO-SEA environment we need:

- Your Slurm batch scripts to launch jobs for each step.

HOW DO WE DO IT?

To move an existing workflow into the IO-SEA environment we need:

- Your Slurm batch scripts to launch jobs for each step.
- A YAML-format Workflow Description File (WDF).

HOW DO WE DO IT?

To move an existing workflow into the IO-SEA environment we need:

- Your Slurm batch scripts to launch jobs for each step.
- A YAML-format Workflow Description File (WDF).
- That's pretty much it.

HOW DO WE DO IT?

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three LQCD workflow steps:

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three LQCD workflow steps:

STEP	READS IN	WRITES OUT

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three LQCD workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three LQCD workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs
B	1 configuration file	multiple propagator files

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three LQCD workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs
B	1 configuration file	multiple propagator files
C	1 configuration groups of propagators	correlation arrays

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs
B	1 configuration file	multiple propagator files
C	1 configuration groups of propagators	correlation arrays

Note that we have data re-use in the lattice gauge field configuration files and the propagator files.

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs
B	1 configuration file	multiple propagator files
C	1 configuration groups of propagators	correlation arrays

Note that we have data re-use in the lattice gauge field configuration files and the propagator files.

HOW DO WE DO IT?

Assume I already have Slurm batch scripts prepared for three workflow steps:

STEP	READS IN	WRITES OUT
A	initial configuration file	Markov chain of configs
B	1 configuration file	multiple propagator files
C	1 configuration groups of propagators	correlation arrays

Note that we have data re-use in the lattice gauge field configuration files and the propagator files.

Since you were wondering, in our tests:

configuration file	1.2GB
propagator	4.6 GB
correllator	few 10^2 kB

... but could be *much* bigger in modern production runs!

PREPARE A WDF — SBB EXAMPLE

Workflow Description File is a YAML-format file describing how ephemeral services will be used by the workflow steps.

PREPARE A WDF — SBB EXAMPLE

Workflow Description File is a YAML-format file describing how ephemeral services will be used by the workflow steps.

```yaml
workflow:
  name: LQCD_ABC

services:
  - name: lqcd-sbb1
    type: SBB
    attributes:
      targets: /afsm/iosea/LQCD_DATA/PROJECT2/
      flavor: high
      datanodes: 4
      location: dp-esb

steps:
  - name: step_A
    command: "sbatch ~/sub_LQCD_stepA.sh"
    services:
      - name: lqcd-sbb1

  - name: steps_B_and_C
    command: "sbatch ~/sub_LQCD_stepBC"
    services:
      - name: lqcd-sbb1
```

PREPARE A WDF — SBB EXAMPLE

Workflow Description File is a YAML-format file describing how ephemeral services will be used by the workflow steps.

```yaml
workflow:
  name: LQCD_ABC

services:
  - name: lqcd-sbb1
    type: SBB
    attributes:
      targets: /afsm/iosea/LQCD_DATA/PROJECT2/
      flavor: high
      datanodes: 4
      location: dp-esb

steps:
  - name: step_A
    command: "sbatch ~/sub_LQCD_stepA.sh"
    services:
      - name: lqcd-sbb1

  - name: steps_B_and_C
    command: "sbatch ~/sub_LQCD_stepBC"
    services:
      - name: lqcd-sbb1
```

← Configures the SBB service

PREPARE A WDF — SBB EXAMPLE

Workflow Description File is a YAML-format file describing how ephemeral services will be used by the workflow steps.

```yaml
workflow:
  name: LQCD_ABC

services:
  - name: lqcd-sbb1
    type: SBB
    attributes:
      targets: /afsm/iosea/LQCD_DATA/PROJECT2/
      flavor: high
      datanodes: 4
      location: dp-esb

steps:
  - name: step_A
    command: "sbatch ~/sub_LQCD_stepA.sh"
    services:
      - name: lqcd-sbb1

  - name: steps_B_and_C
    command: "sbatch ~/sub_LQCD_stepBC"
    services:
      - name: lqcd-sbb1
```

← Configures the SBB service

← Instructions for launching step

PREPARE A WDF — SBB EXAMPLE

Workflow Description File is a YAML-format file describing how ephemeral services will be used by the workflow steps.

```yaml
workflow:
  name: LQCD_ABC

services:
  - name: lqcd-sbb1
    type: SBB
    attributes:
      targets: /afsm/iosea/LQCD_DATA/PROJECT2/
      flavor: high
      datanodes: 4
      location: dp-esb

steps:
  - name: step_A
    command: "sbatch ~/sub_LQCD_stepA.sh"
    services:
      - name: lqcd-sbb1

  - name: steps_B_and_C
    command: "sbatch ~/sub_LQCD_stepBC"
    services:
      - name: lqcd-sbb1
```

⟵ Configures the SBB service

⟵ Instructions for launching step

⟵ Associates ephemeral service with the step

WDF — EPHEMERAL SERVICES EXAMPLES

```yaml
workflow:
  name: LQCD_ABC
services:
  - name: lqcd-sbb1
    type: SBB
    attributes:
      targets: /afsm/iosea/LQCD_DATA/PROJECT2/
      flavor: high
      datanodes: 4
      location: dp-esb

  - name: lqcd-nfs
    type: NFS
    attributes:
      namespace: HESTIA@/afsm/iosea/USE_CASES/LQCD/DATASETS//lqcd-bm-dataset-2024-02-09-160348-ID2
      mountpoint: /afsm/iosea/USE_CASES/LQCD/MNT/
      storagesize: 50GiB
      datanodes: 1
      location: dp-esb

  - name: lqcd-dasi
    type: DASI
    attributes:
      dasiconfig: /p/project/iosea/USE_CASES/LQCD/DASI/lqcd_dasi.yaml
      namespace: HESTIA@/afsm/iosea/USE_CASES/LQCD/DATASETS/DASI-{{ INDEX }}/
      storagesize: 50GiB
      datanodes: 1
      location: dp-esb
```

DASI NEEDS A COUPLE EXTRAS

```
---
schema: /p/project/iosea/USE_CASES/LQCD/DASI/schema
catalogue: toc
store: file
spaces:
  - roots:
    - path: /afsm/iosea/USE_CASES/LQCD/DASI_DATA
```

A schema file:

```
[ FermAct, GaugeAct, beta, volume, mass [ index, type [ dummy?  ]  ]  ]
```

- Gauge Action="Wilson"
- Fermion Action="Clover"
- $\beta = 3.6$
- mass=0.2
- volume=$32^3 \times 64$

describes experiment

Data:

config_1.lime
metadata_1.xml
config_2.lime
metadata_2.xml
config_3.lime
metadata_3.xml
....

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

Use the prepared workflow description file `lqcd_wdf.yaml` , to initialize the workflow:

```
> iosea-wf start -w lqcd_wdf.yaml -s lqcd_00
```

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

Use the prepared workflow description file `lqcd_wdf.yaml` , to initialize the workflow:

```
> iosea-wf start -w lqcd_wdf.yaml -s lqcd_00
```

Before running any steps, check that the service is started:

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

Use the prepared workflow description file `lqcd_wdf.yaml` , to initialize the workflow:

```
> iosea-wf start -w lqcd_wdf.yaml -s lqcd_00
```

Before running any steps, check that the service is started:

```
> iosea-wf status -s lqcd_00
SESSION                 WORKFLOW                    STATUS
lqcd_00                 LQCD_WFM_ABC                starting
```

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

Use the prepared workflow description file `lqcd_wdf.yaml` , to initialize the workflow:

```
> iosea-wf start -w lqcd_wdf.yaml -s lqcd_00
```

Before running any steps, check that the service is started:

```
> iosea-wf status -s lqcd_00
SESSION                 WORKFLOW                    STATUS
lqcd_00                 LQCD_WFM_ABC                starting

> iosea-wf status -s lqcd_00
SESSION                 WORKFLOW                    STATUS
lqcd_00                 LQCD_WFM_ABC                active
```

STARTING THE WORKFLOW SESSION

In current implementation, each user must have an instance of the persistent API server running in the background:

```
> wfm-api &
```

Use the prepared workflow description file `lqcd_wdf.yaml` , to initialize the workflow:

```
> iosea-wf start -w lqcd_wdf.yaml -s lqcd_00
```

Before running any steps, check that the service is started:

```
> iosea-wf status -s lqcd_00
SESSION                 WORKFLOW                    STATUS
lqcd_00                 LQCD_WFM_ABC                starting

> iosea-wf status -s lqcd_00
SESSION                 WORKFLOW                    STATUS
lqcd_00                 LQCD_WFM_ABC                active
```

Good to go!

RUNNING THE WORKFLOW STEP

RUNNING THE WORKFLOW STEP

```
> iosea-wf run -s lqcd_00 -t step_A
```

RUNNING THE WORKFLOW STEP

```
> iosea-wf run -s lqcd_00 -t step_A
```

Issues the slurm command described in the WDF for `step_A`:

```
sbatch ~/sub_LQCD_stepA.sh
```

RUNNING THE WORKFLOW STEP

```
> iosea-wf run -s lqcd_00 -t step_A
```

Issues the slurm command described in the WDF for `step_A`:

```
sbatch ~/sub_LQCD_stepA.sh
```

- Can run multiple instances of the same step concurrently.
- Current WMF version does not yet handle complicated job dependencies

RUNNING THE WORKFLOW STEP

```
> iosea-wf run -s lqcd_00 -t step_A
```

Issues the slurm command described in the WDF for `step_A`:

```
sbatch ~/sub_LQCD_stepA.sh
```

- Can run multiple instances of the same step concurrently.
- Current WMF version does not yet handle complicated job dependencies

```
> iosea-wf run -s lqcd_00 -t step_BC
```

RUNNING THE WORKFLOW STEP

```
> iosea-wf run -s lqcd_00 -t step_A
```

Issues the slurm command described in the WDF for `step_A`:

```
sbatch ~/sub_LQCD_stepA.sh
```

- Can run multiple instances of the same step concurrently.
- Current WMF version does not yet handle complicated job dependencies

```
> iosea-wf run -s lqcd_00 -t step_BC
```

Burst-buffer space is finite, so we should free the resource by stopping the session when the runs are complete:

```
iosea-wf stop -s lqcd_00
```

WHAT HAVE WE GAINED?

WHAT HAVE WE GAINED?

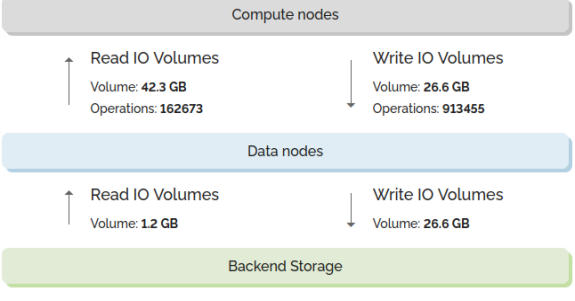
STEP	READS IN	WRITES OUT
A	1 config 1.2 GB	6 configs (1×1.2 GB) 6 GB
B	1 config 1.2 GB	4 props (4×4.8 GB) $\approx$ 19.3 GB
C	1 config 1.2 GB	
	8 props (4×4.8 GB) 38.7 GB	
TOTAL	42.3 GB	26.6 GB

WHAT HAVE WE GAINED?

Economy of data movement!

STEP	READS IN	WRITES OUT
A	1 config 1.2 GB	6 configs (1×1.2 GB) 6 GB
B	1 config 1.2 GB	4 props (4×4.8 GB) $\approx$ 19.3 GB
C	1 config 1.2 GB	
	8 props (4×4.8 GB) 38.7 GB	
TOTAL	42.3 GB	26.6 GB

IOI says:

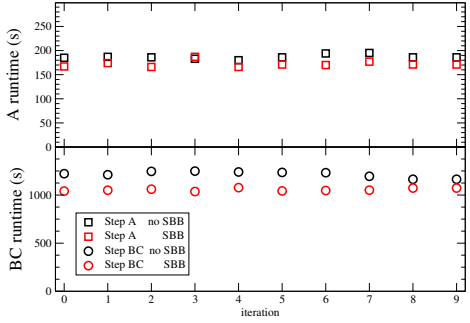


PERFORMANCE

Test setup is not for high performance runs:

- Proof-of-concept of IO-SEA environment
- Datanodes are ∼9 year old hardware
- Connected to DEEP by two long IB cables

Nevertheless, performance improvement is visible compared to direct accfess to JSC NFS storage.



Performance with the SBB service does not beat direct access to the all-flash storage module.

TESTS ON A BUSY STORAGE SYSTEM

TESTS ON A BUSY STORAGE SYSTEM

- Reserve the system

TESTS ON A BUSY STORAGE SYSTEM

- Reserve the system
- Start *N* instances of IOR benchmark —
  loads the storage with constant, repeated writes

TESTS ON A BUSY STORAGE SYSTEM

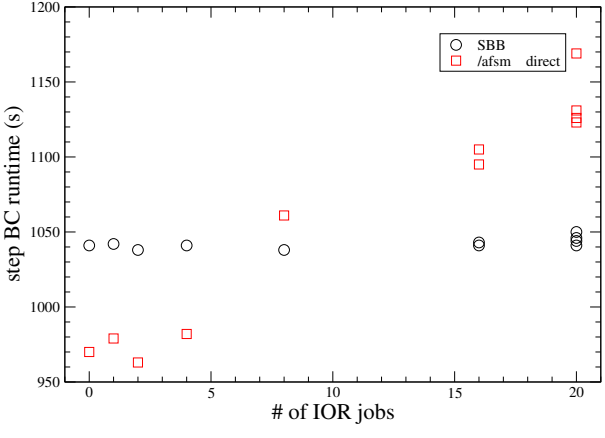
- Reserve the system
- Start *N* instances of IOR benchmark —
  loads the storage with constant, repeated writes
- Run the LQCD workflow with and without the SBB service

TESTS ON A BUSY STORAGE SYSTEM

- Reserve the system
- Start *N* instances of IOR benchmark —
  loads the storage with constant, repeated writes
- Run the LQCD workflow with and without the SBB service
- Repeat with increasing *N*

TESTS ON A BUSY STORAGE SYSTEM

- Reserve the system
- Start *N* instances of IOR benchmark —
  loads the storage with constant, repeated writes
- Run the LQCD workflow with and without the SBB service
- Repeat with increasing *N*



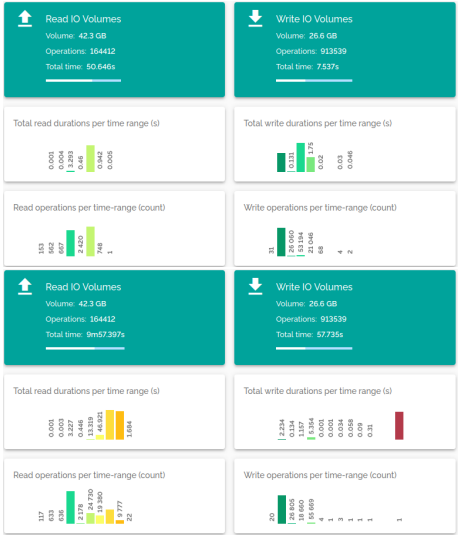
TESTS ON A BUSY STORAGE SYSTEM

Direct AFSM

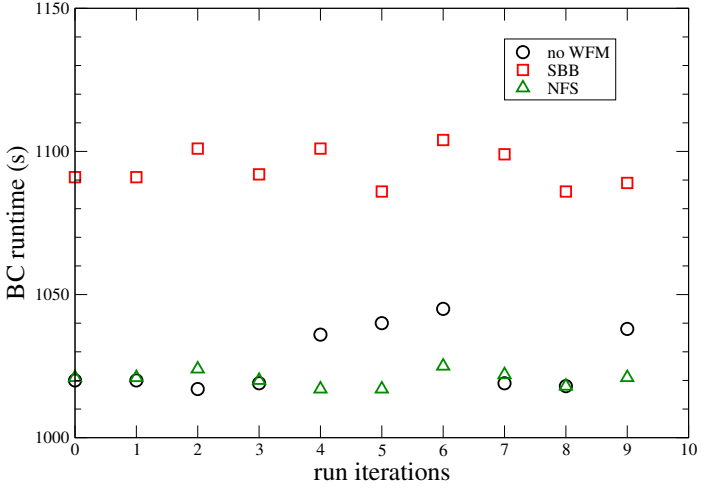
SBB

unperturbed

perturbed (20x IOR)



PERFORMANCE COMPARISON



SUMMARY

- The IO-SEA WFM provides ephemeral storage services to optimize the I/O and data management of workflows in the exascale era.
- Currently three ephemeral storage services are deployed on the IO-SEA prototype on the DEEP system at JSC: SBB, NFS, DASI.
- Each has different characteristics, performance, and purpose.

Demo, if time...