Hewlett Packard Enterprise

HPC Data Management with HPE DMF and SUSE Enterprise Storage

A large customer experience
Alberto Galli HPC Architect alberto.galli@hpe.com
Cedric Milesi HPC Storage & AI Architect cedric.milesi@hpe.com

Sept 2019

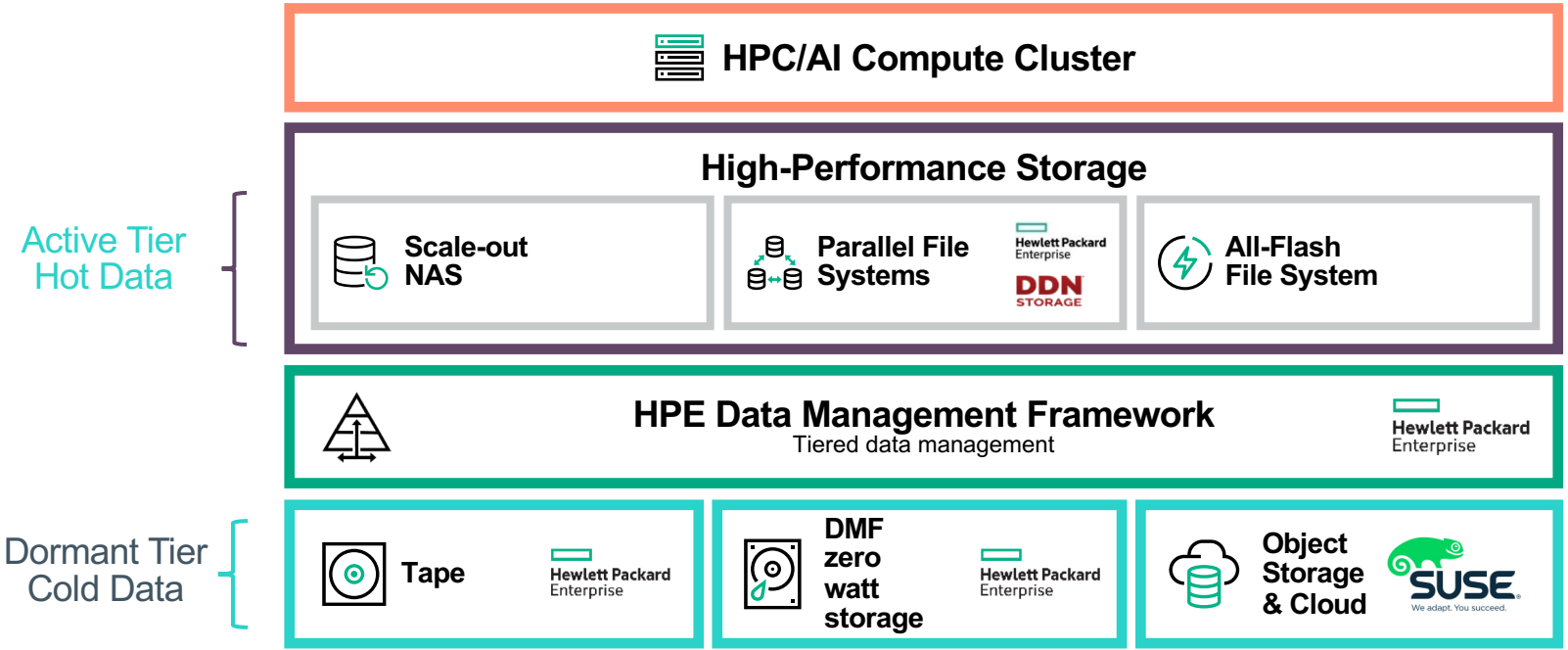
Agenda

- Technology basics
- DMF Workflow
- DMF Example
- Ceph backend

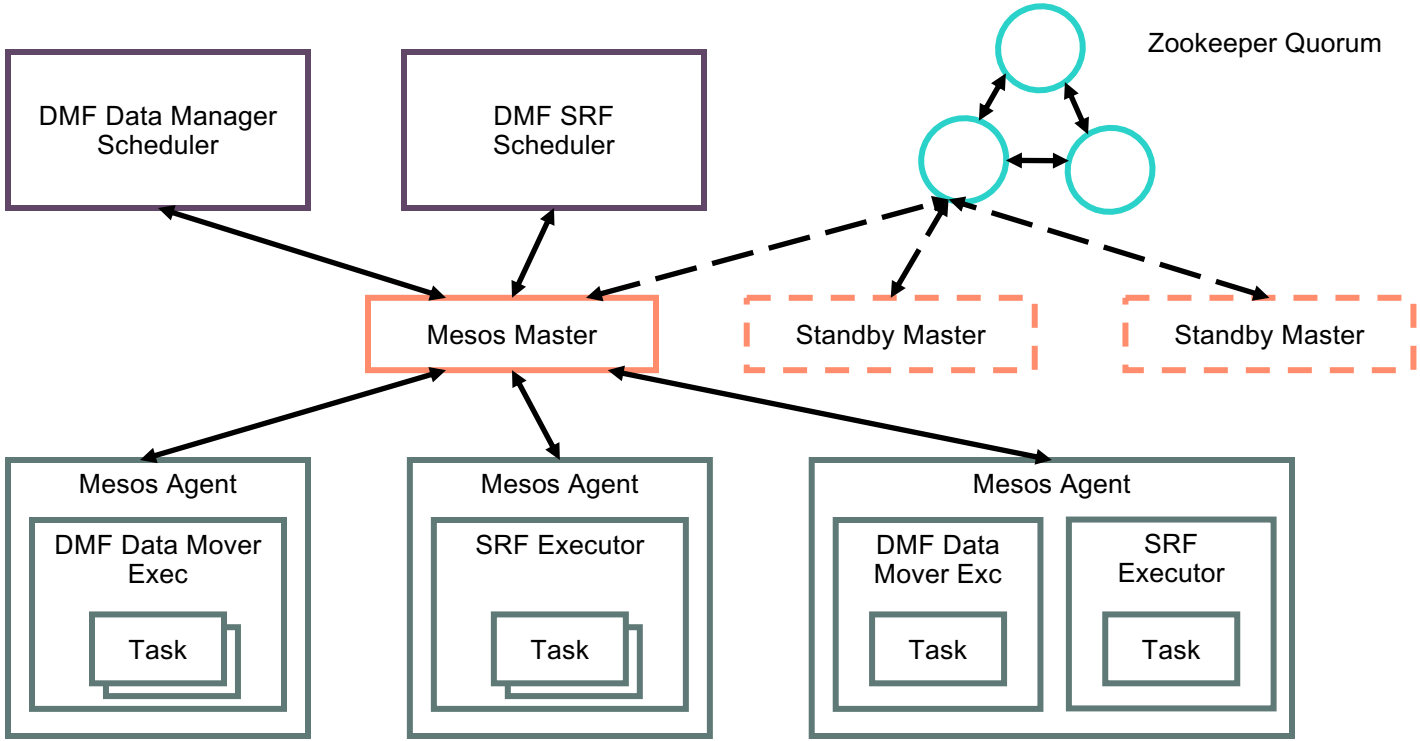
Technology Basics

Introducing Data Management Framework

Active & Dormant Data Forms



DMF Architecture



Request Processing Workflow

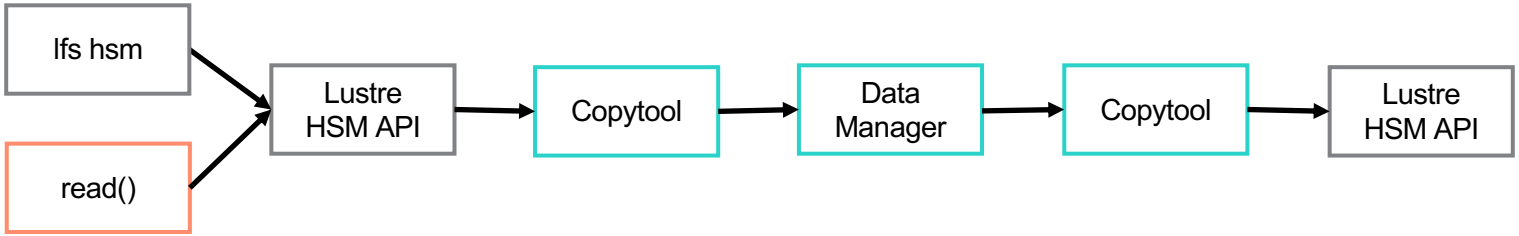
- DMF supports three (3) HSM command flows:
  - DMF CLI or API initiated data movement
  - Lustre lfs hsm command initiated data movement
  - Filesystem read() of an offline file initiated recall

Request Processing Workflow

DMF Initiated vs Lustre Initiated Data Movement

DMF CLI:

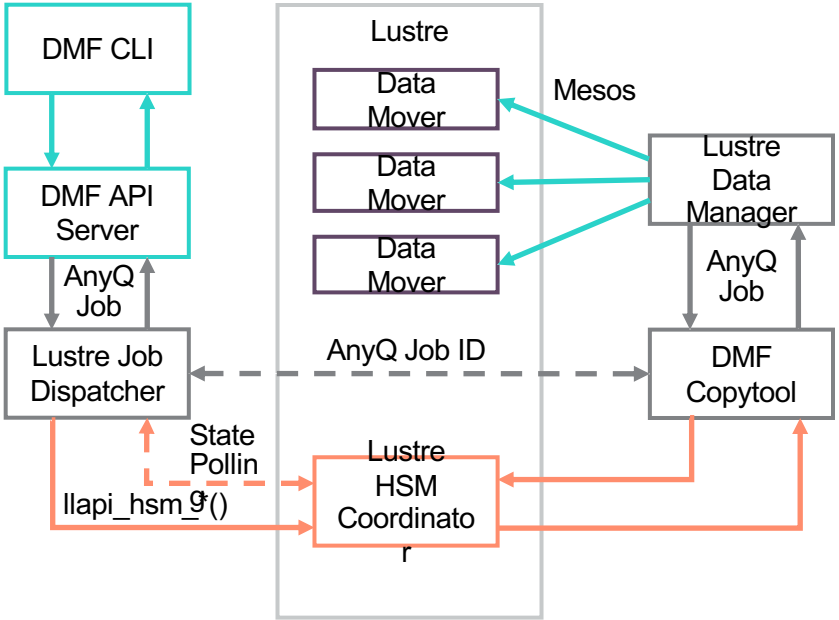
Lustre lfs hsm:
Read() of OFL:



Request Processing Workflow

DMF Get & Put Requests

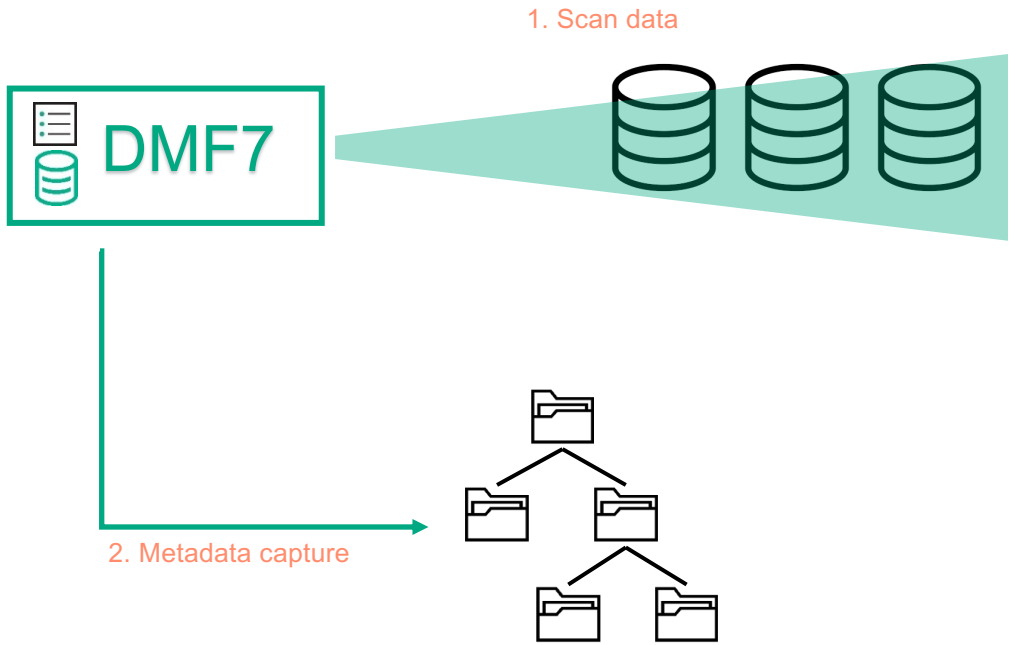
- Request is initiated via DMF CLI
- Dispatcher creates AnyQ job and calls Lustre HSM API to asynchronously submit request
- Lustre HSM processes request and calls DMF Copytool
- DMF Copytool locates associated AnyQ job and forwards it to DMF Lustre Data Manager
- Data Manager schedules data movement operation through Mesos framework
- Data Movers perform data copy. Upon completion, Data Manager marks the job accordingly and hands it back to DMF Copytool
- DMF Copytool calls Lustre HSM API to signal completion of transfer
- Job Dispatcher polls jobs that have completed transfer and validates file state transition
- Once file state has transitioned, Job Dispatcher marks AnyQ job as complete and hands it back to DMF API server that in turn signals completion to the API caller, i.e. CLI



DMF workflows

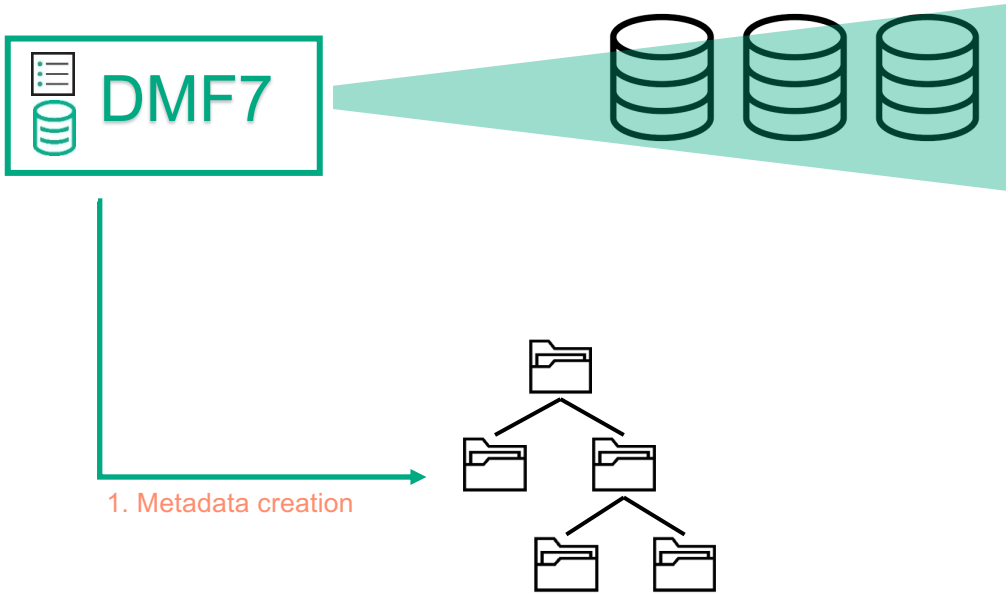
How DMF Works: scanning existing FS

When DMF is plugged in the front tier, it will scan all of the data and capture all the metadata info in a separate metadata repository



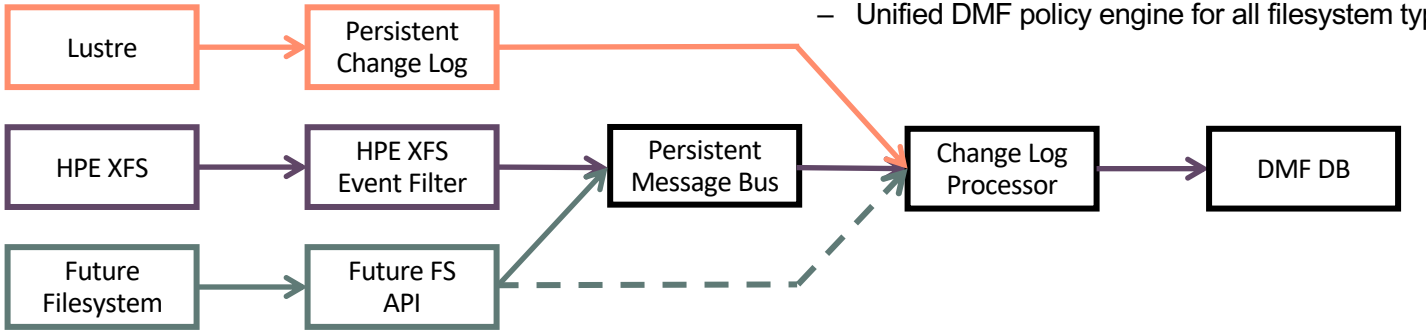
How DMF Works: Populate new FS with existing data

When DMF is plugged in the front tier, we can populate the metadata with content of the Cassandra database



Data Management | **DMF 7 Change Log**

- For HPE XFS:
  - Use DMAPI events to drive filesystem change log and filesystem reflection
  - Removes the need to scan the filesystem to drive the policy engine
  - Removes the need to backup (e.g. xfsdump) the filesystem to preserve the namespace
- For Lustre:
  - Natively process Lustre persistent change log via API
  - Policy engine and filesystem reflection directly out of DMF7 scale out database without needing RobinHood
- Others filesystems support:
  - Makes the DMF front-end filesystem independent
  - Persistent message bus use depends on filesystem API
  - Unified DMF policy engine for all filesystem types

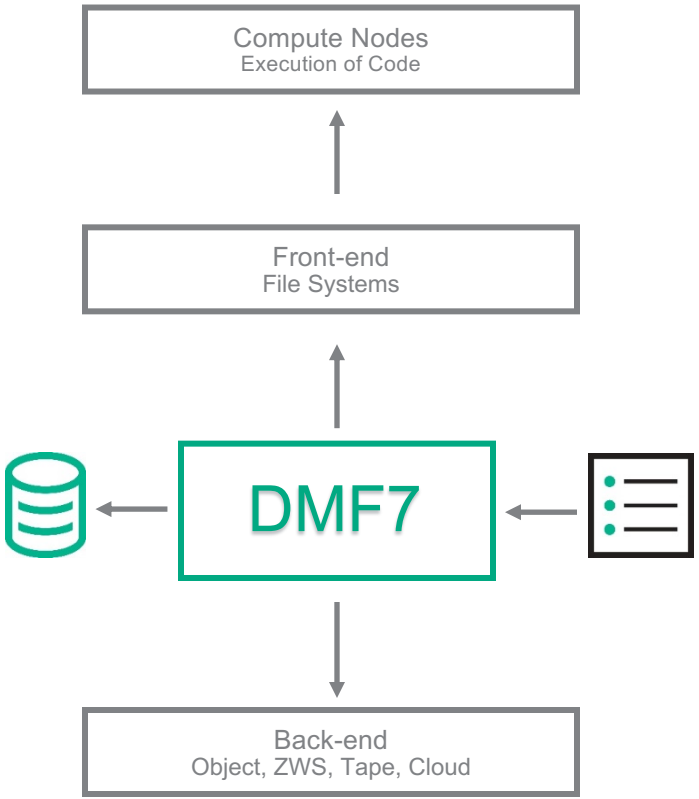


Data Management | **Lustre ChangeLog Processing in DMF 7**

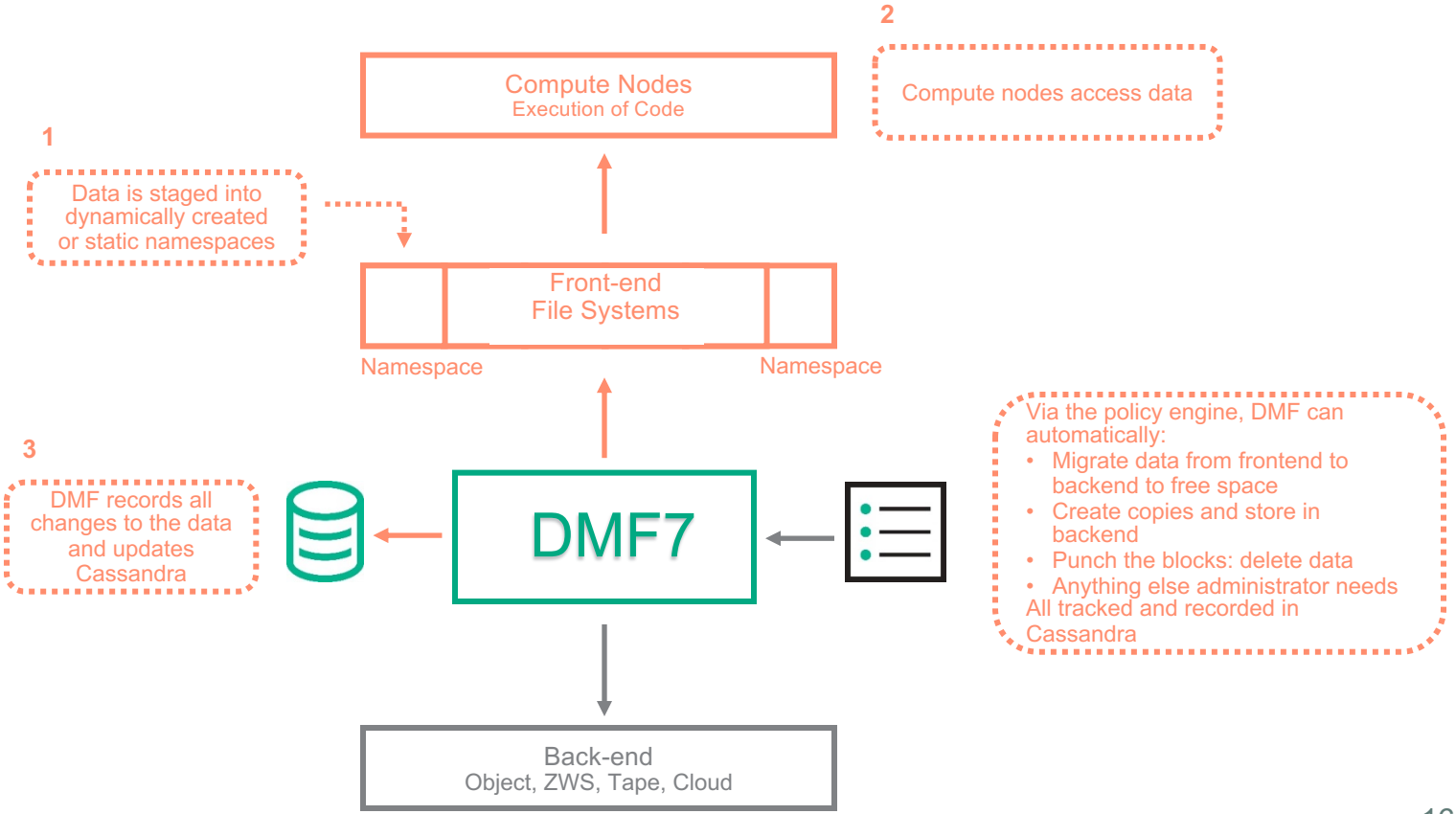
- ChangeLog feature records events that change the file system namespace or file metadata
  - See table on the right
- Changes such as file creation, deletion, renaming, attribute changes, etc. are recorded with the target and parent file identifiers (FIDs), the name of the target, and a timestamp
- DMF 7 uses ChangeLog entries to exactly replicate changes in the file system reflection
  - No additional software (such as RobinHood) is necessary, all work is done natively by DMF 7 ChangeLog processor
  - Native processing of ChangeLog into distributed database enables scalability and Spark queries

Event Type	Description
MARK	Internal recordkeeping
CREAT	Regular file creation
MKDIR	Directory creation
HLINK	Hard link
SLINK	Soft link
MKNOD	Other file creation
UNLNK	Regular file removal
RMDIR	Directory removal
RNMFM	Rename, original
RNMTO	Rename, final
IOCTL	ioctl on file or directory
TRUNC	Regular file truncated
SATTR	Attribute change
XATTR	Extended attribute change
UNKNW	Unknown operation

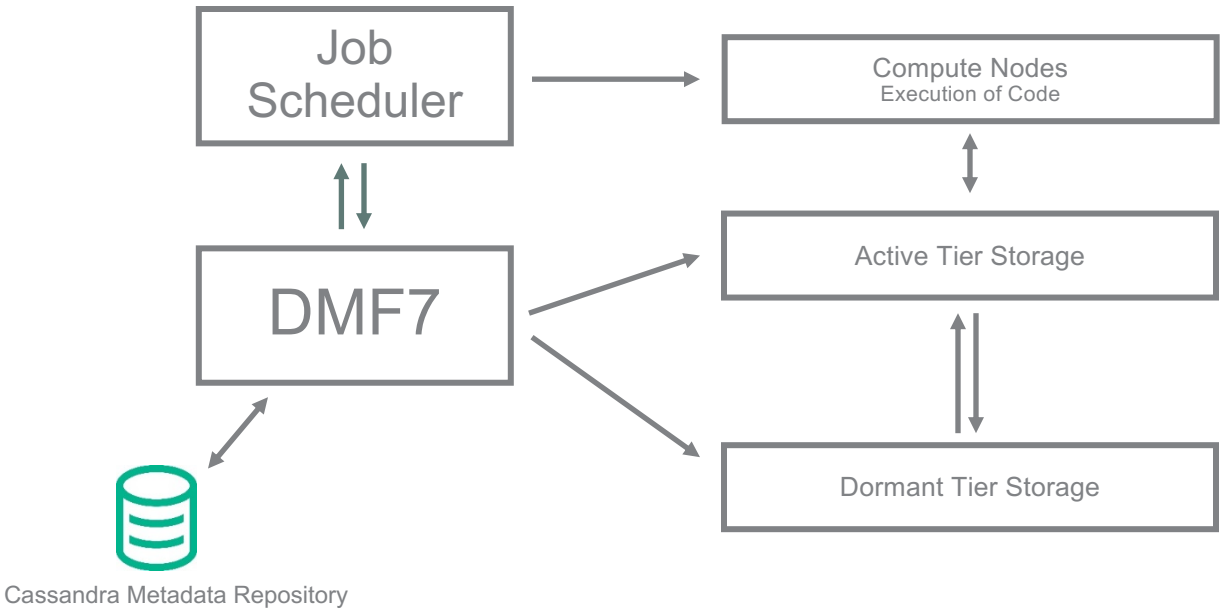
How DMF Works



How DMF Works

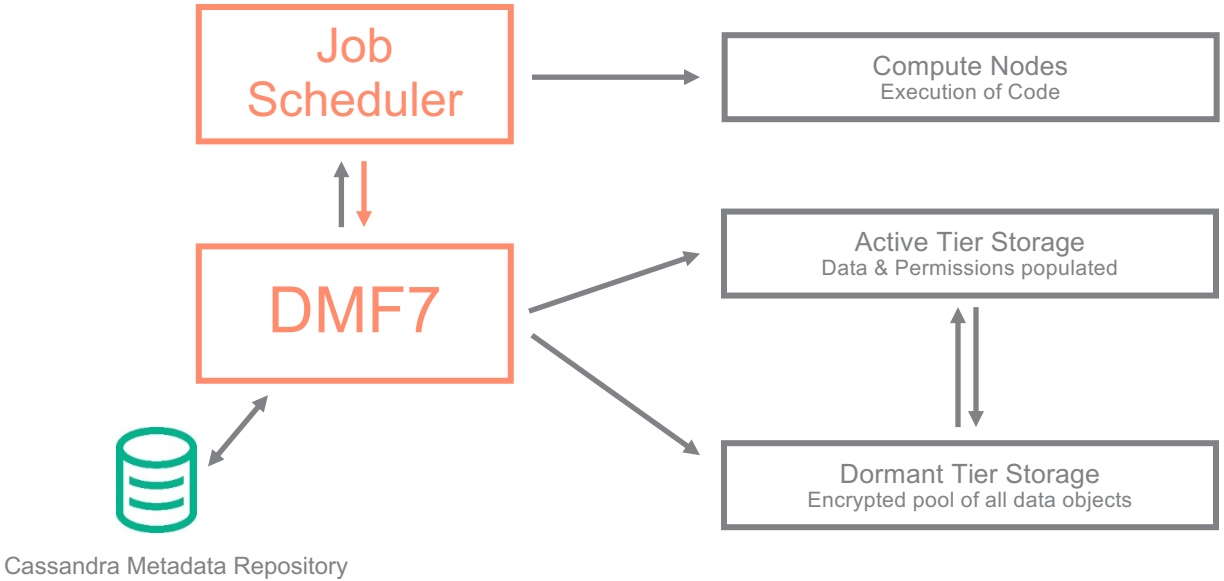


How DMF Works



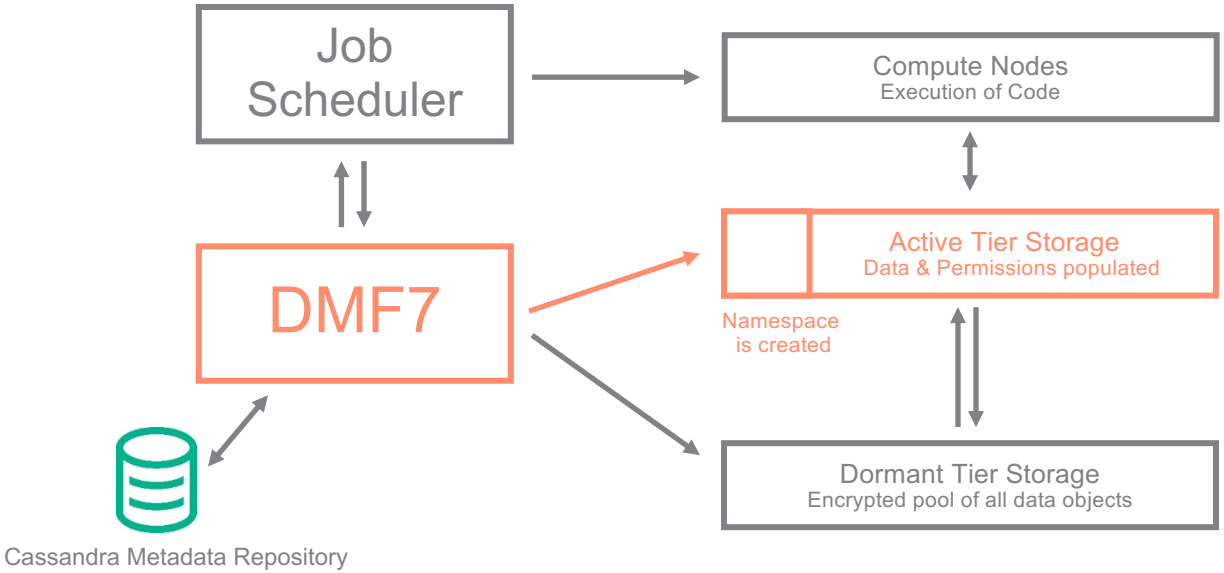
How DMF Works

Action: User schedules a job(s), which is then sent to DMF



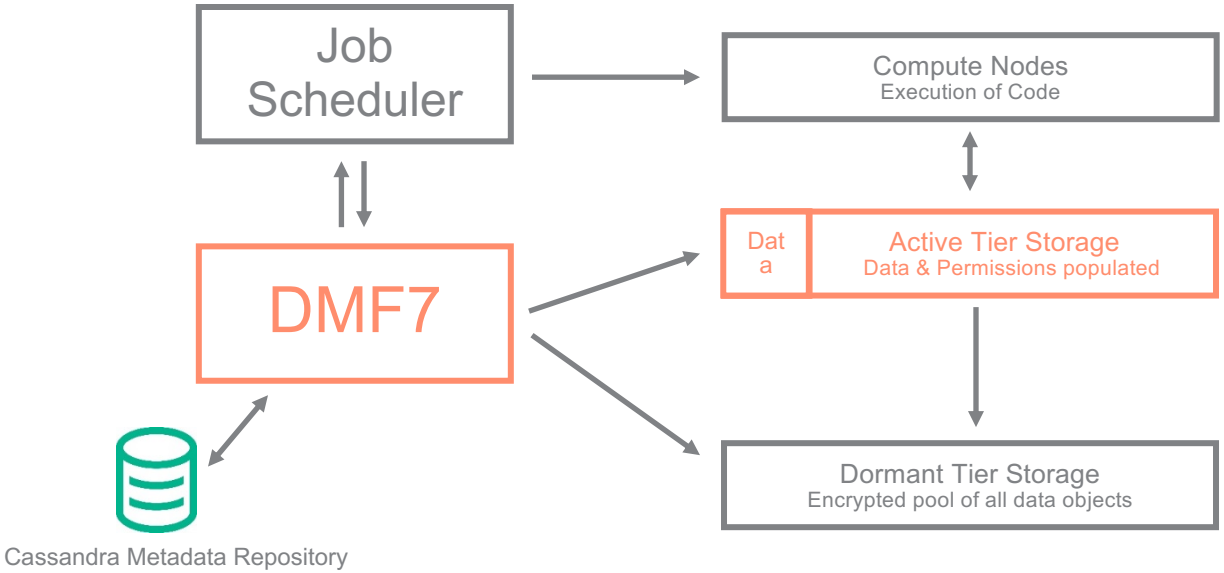
How DMF Works

Action: This triggers DMF to gather necessary resources and to create a namespace in the frontend or use an existing namespace



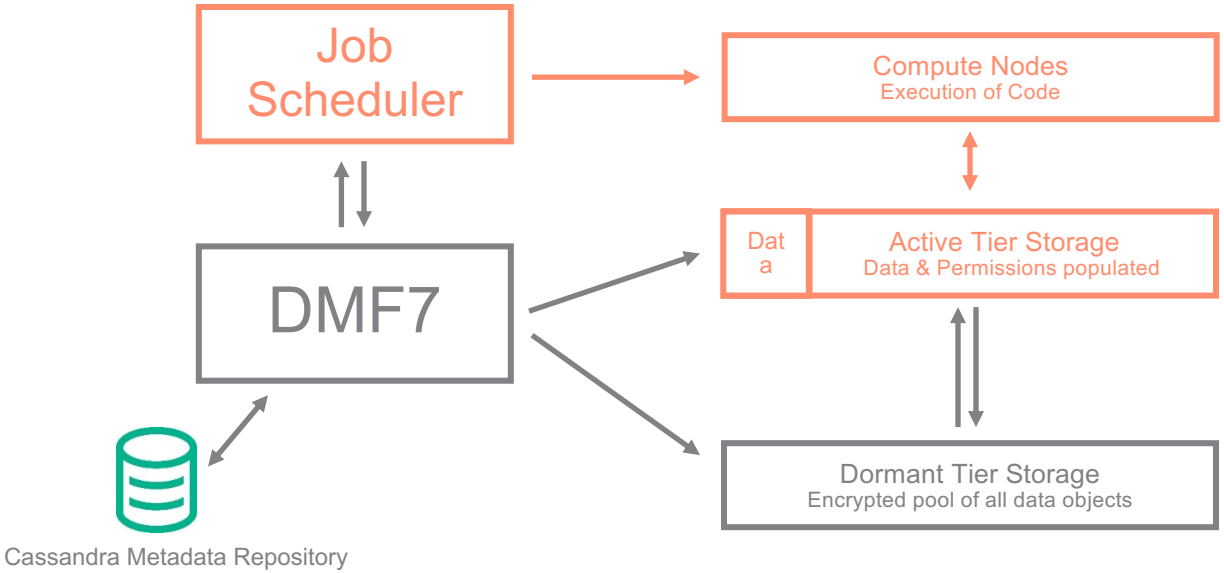
How DMF Works

Action: Once namespace is created, the data is staged into the namespace



How DMF Works

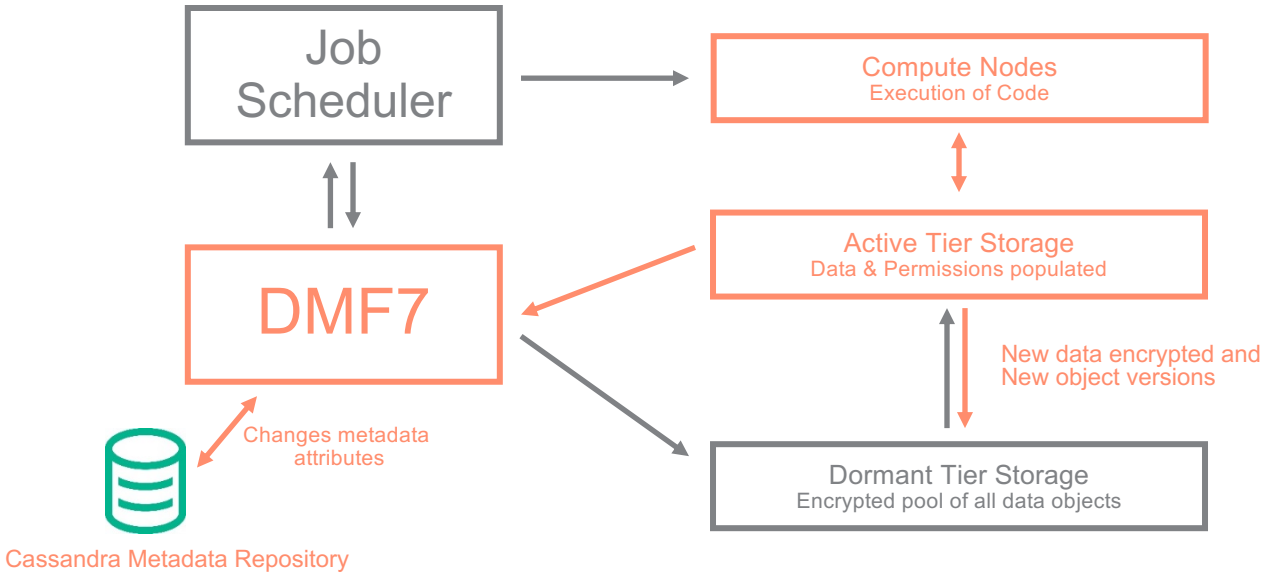
Action: Job from scheduler can run on compute nodes by accessing data from the namespace



At the same time …

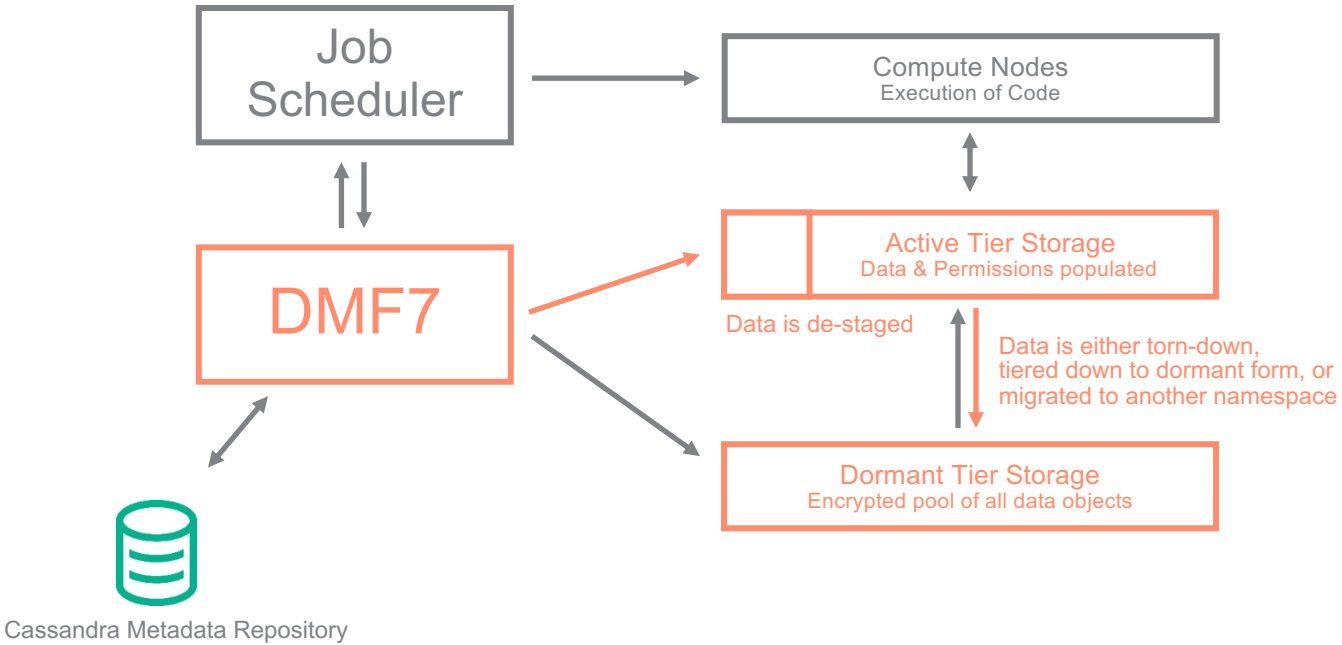
How DMF Works

Action: During job execution, DMF monitors the metadata changes according to the defined policies.



How DMF Works

Action: Once job is done, data is de-staged and tiered down, moved to another namespace, or torn down.



DMF at work – some examples

How DMF can help here

Migrate, Stage & De-stage Operations

Migration to dataset

```
dmf put --fs labfs01 --query "dir.path like '/smalldata/1*'" --set one
```

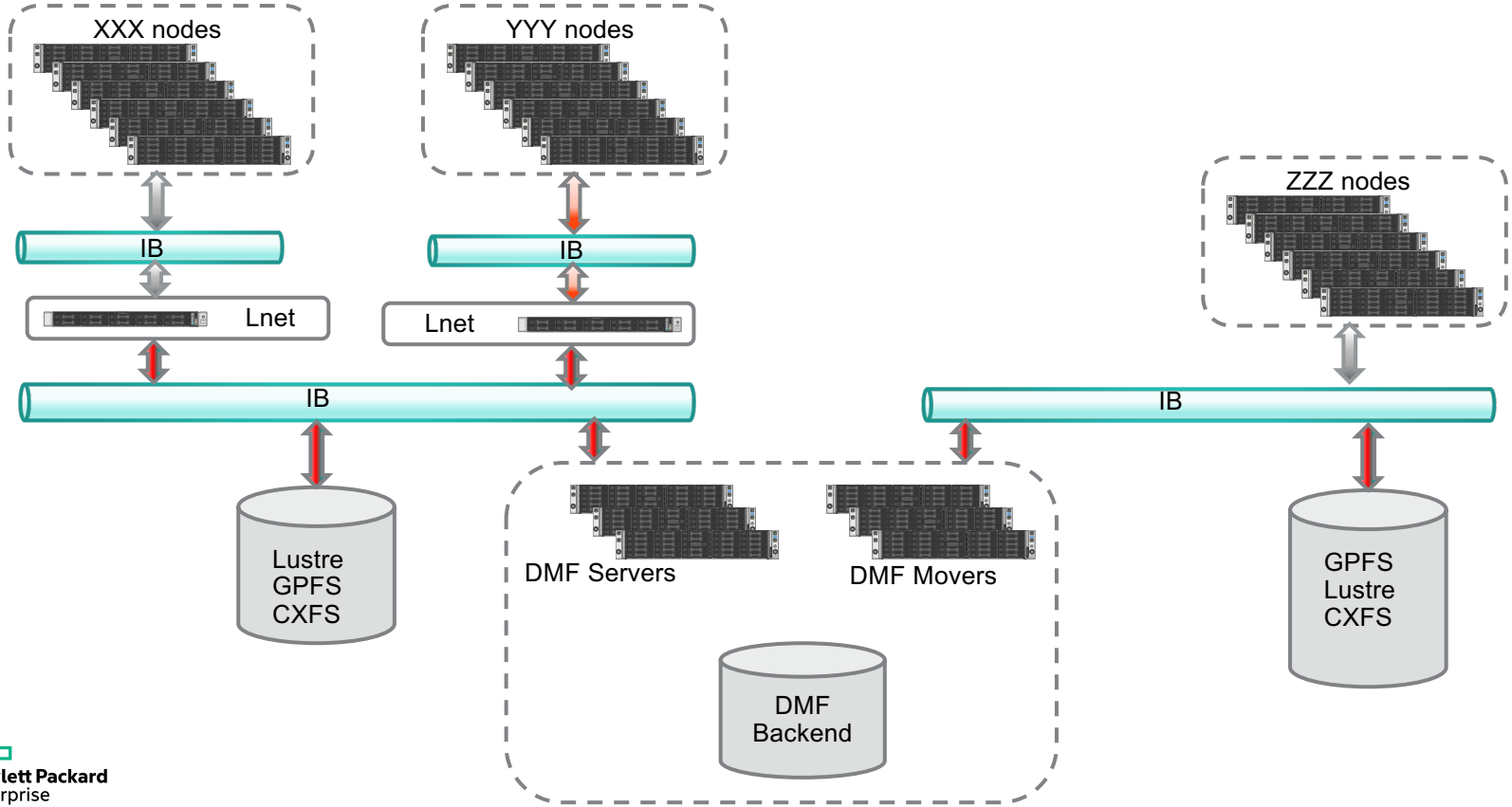
Destaging to dataset, locating it and staging

```
dmf destage --fs labfs01 --query "dir.path = '/SecLists/Fuzzing/Polyglots'" --set polyglot
dmf find --query "object.tags contains 'polyglot’”
dmf stage --set polyglot /labfs01/stage_dir
dmf find --query "object.size < 10000 and name like 'run200*'" --set res200
dmf stage --set res200 /labfs01/res_200
```

Listing Items

```
dmf list /labfs01/data/1/wholly_evidently_*
dmf list --fs aa305e6c-087a-43e1-a162-406c965021c4 --fid 000e0000-0000-0005-0000-000000000105
dmf list --fs labfs01 --fid 000e0000-0000-0005-0000-000000000105
dmf list --set “polyglot*”
dmf list --obj f0e830d5-ef35-4576-98d6-909478da5713
```

Using stage and destage different clusters can share data



Removing data, undelete data

Offlining data (remove from front end)

Release the data blocks for the /cxfsusr1/10Mfilestest_files-11mb-41

dmf punch /cxfsusr1/10Mfilestest_files-11mb-41

(similar to) lfs hsm_release

Removing data (remove from back end)

Data needs destaged first

dmf remove --query "file.xattr['user']['project'] = 'ocean1’

Undeleting data:

After some data have been mistakenly removed

dmf stage /labfs01/1990_E1/ \ --query "file.xattr['user']['project'] = 'ocean1’

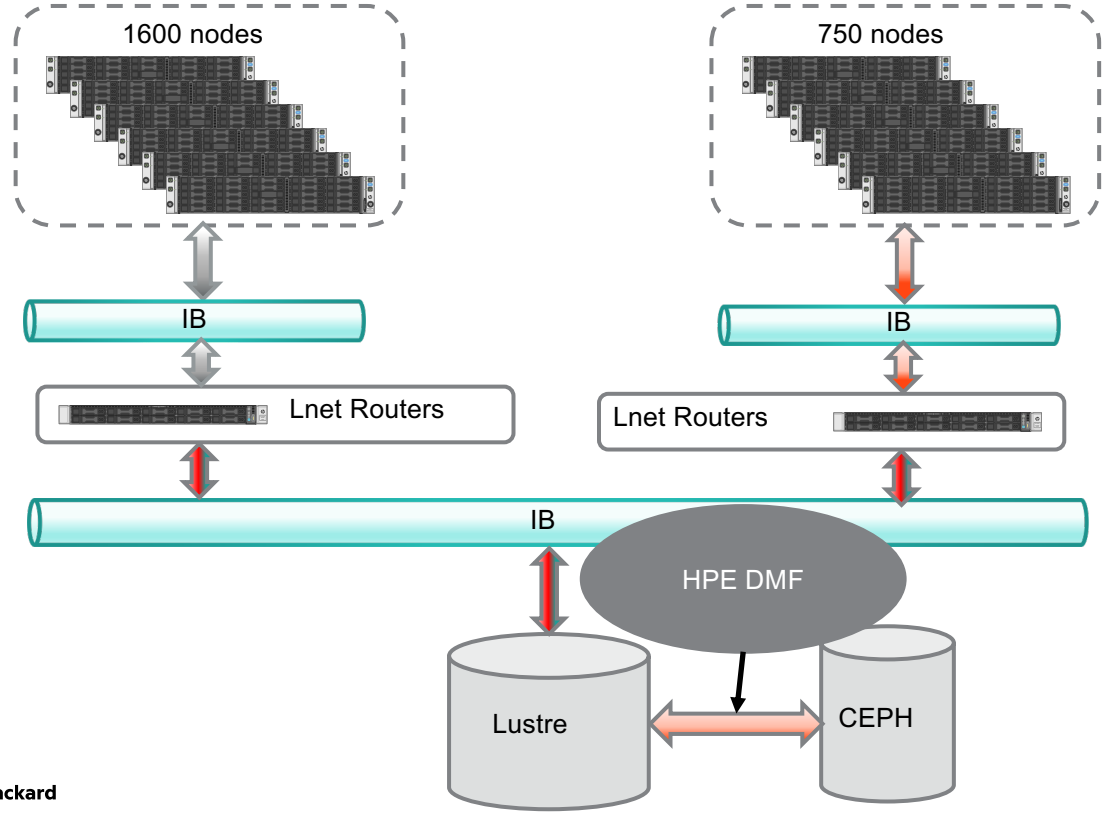
Restore file from project ocean1. Could also be used to get older version of a file(s)

DMF: backend Suse enterprise storage

Concepts

- HPE DMF (Data Management Framework)
  - Provides automatic movement of data based upon policies between storage tiers
  - Permits archiving i.e. safe enough long term storing of data
  - Frees up Tier 0 (costly and not bullet proof) for its main use, i.e. fast access “scratch areas”
  - New features will sum up and will provide “magic movement” between tiers at the same level
- SUSE Enterprise Storage – powered by Ceph
  - It’s an object storage that provides long term archiving using commodity hardware
  - Not really designed for speed but designed for data retention
  - Easily expandable
  - Provides a good numbers of different gateways giving a good flexibility in its accessibility

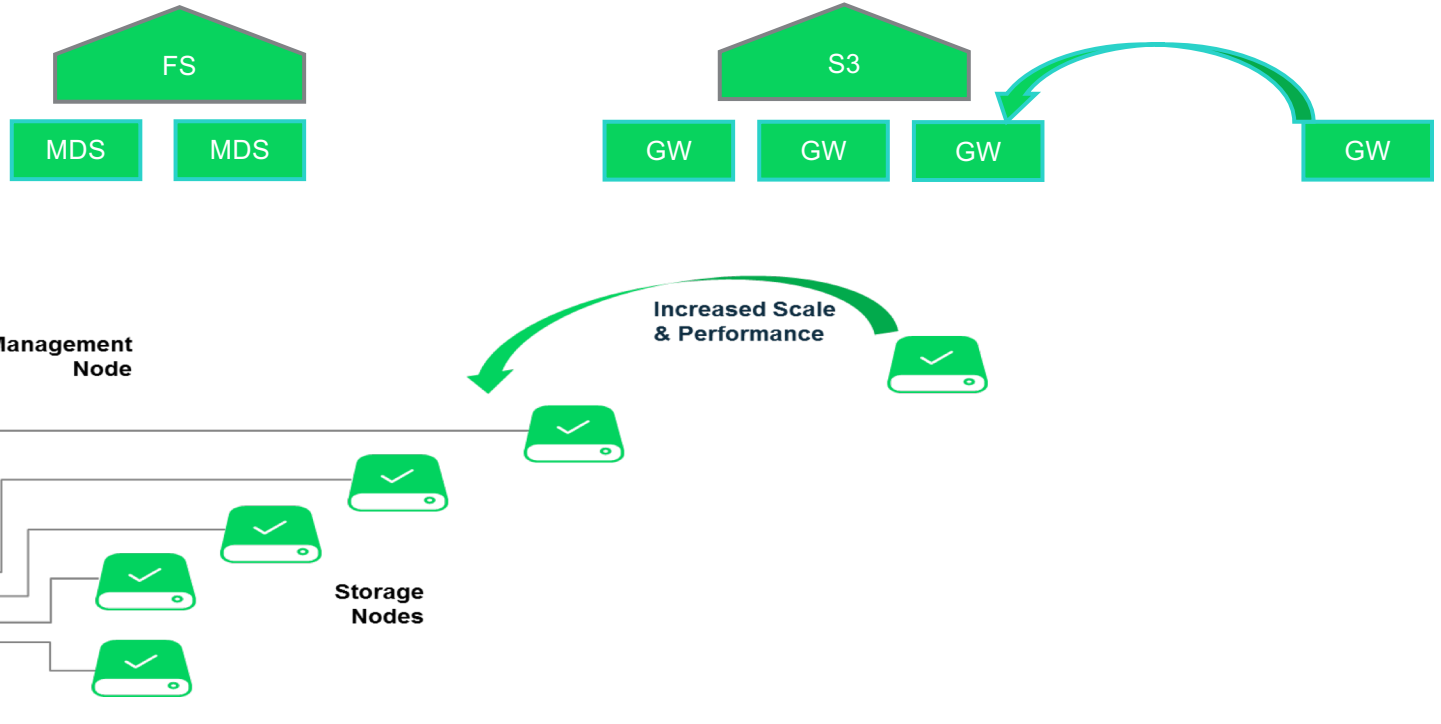
Solution at work



– HPE DMF uses Big Data technology to deal with – indeed – big data
  – Robustness
  – Scalability
  – High performance
– Ceph can scale as well

SUSE Enterprise Storage

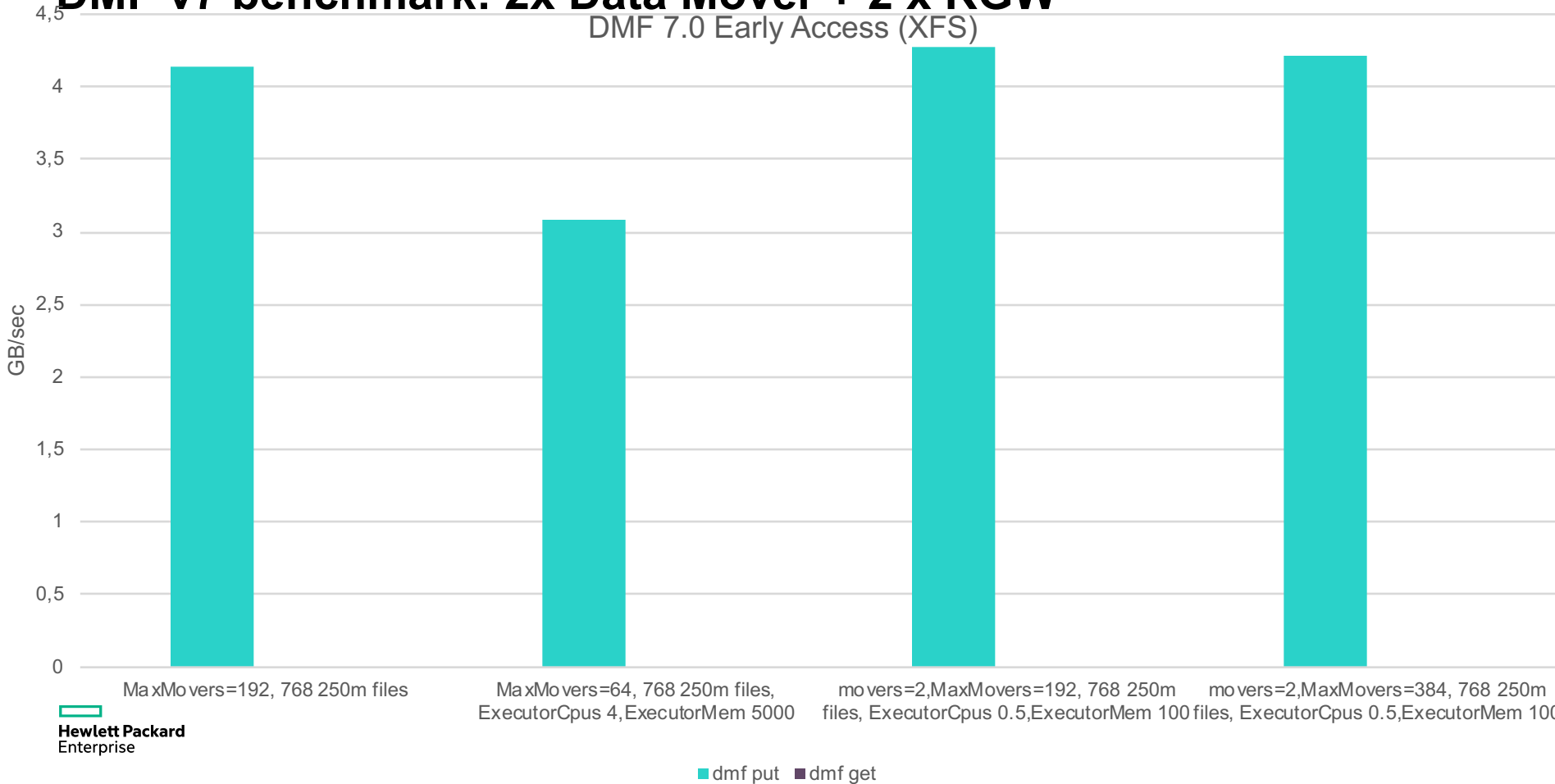
Anatomy



Why Object storage (SUSE Enterprise Storage)

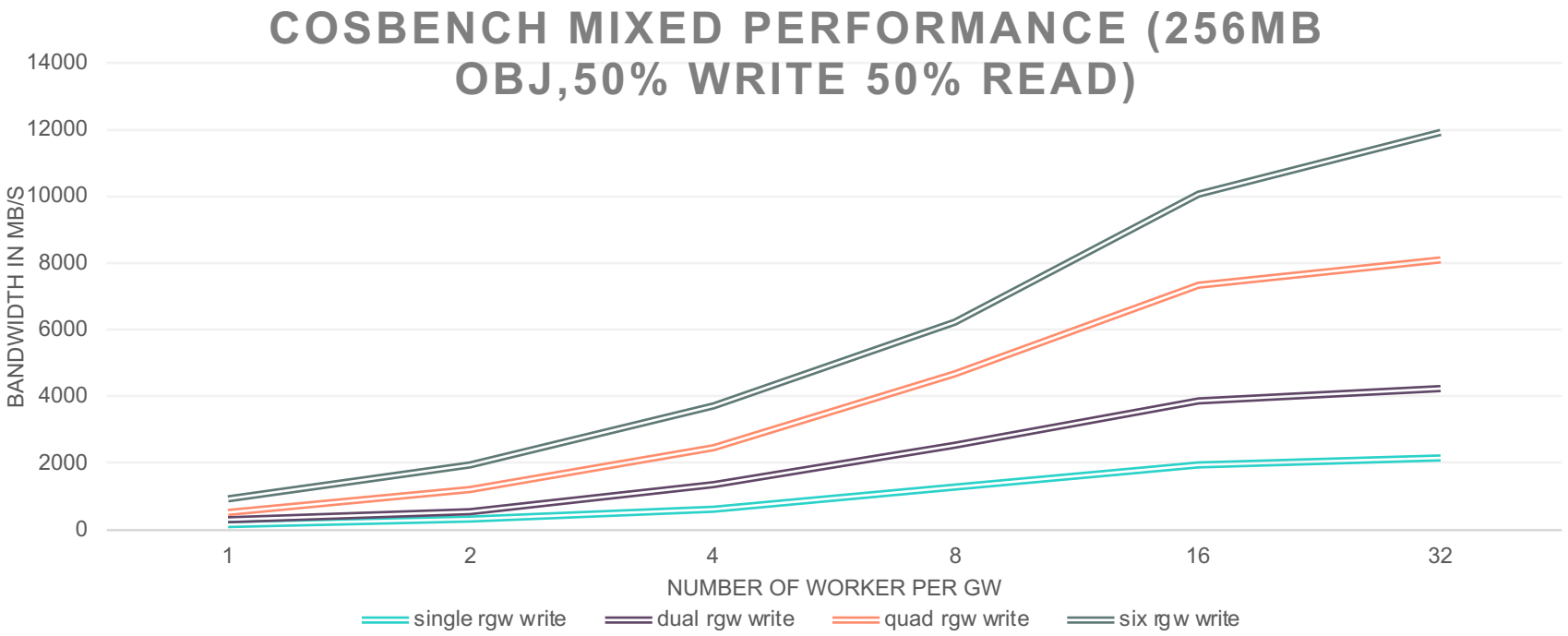
- New technology that can easily used not only for archiving
  - Object storage can extend its use to different areas inside customer site
- Protocol neutrality
  - Using S3 we can dare to say that we are “cloud ready”
- Sill maintain an easy expandability
  - Blocks can be added on the fly to the pool
- Why Ceph
  - It is largely used
  - Open Source
  - Price affordable

DMF v7 benchmark: 2x Data Mover + 2 x RGW



Hewlett Packard Enterprise

Backend Performance: S3 Read/Write



Hewlett Packard
Enterprise

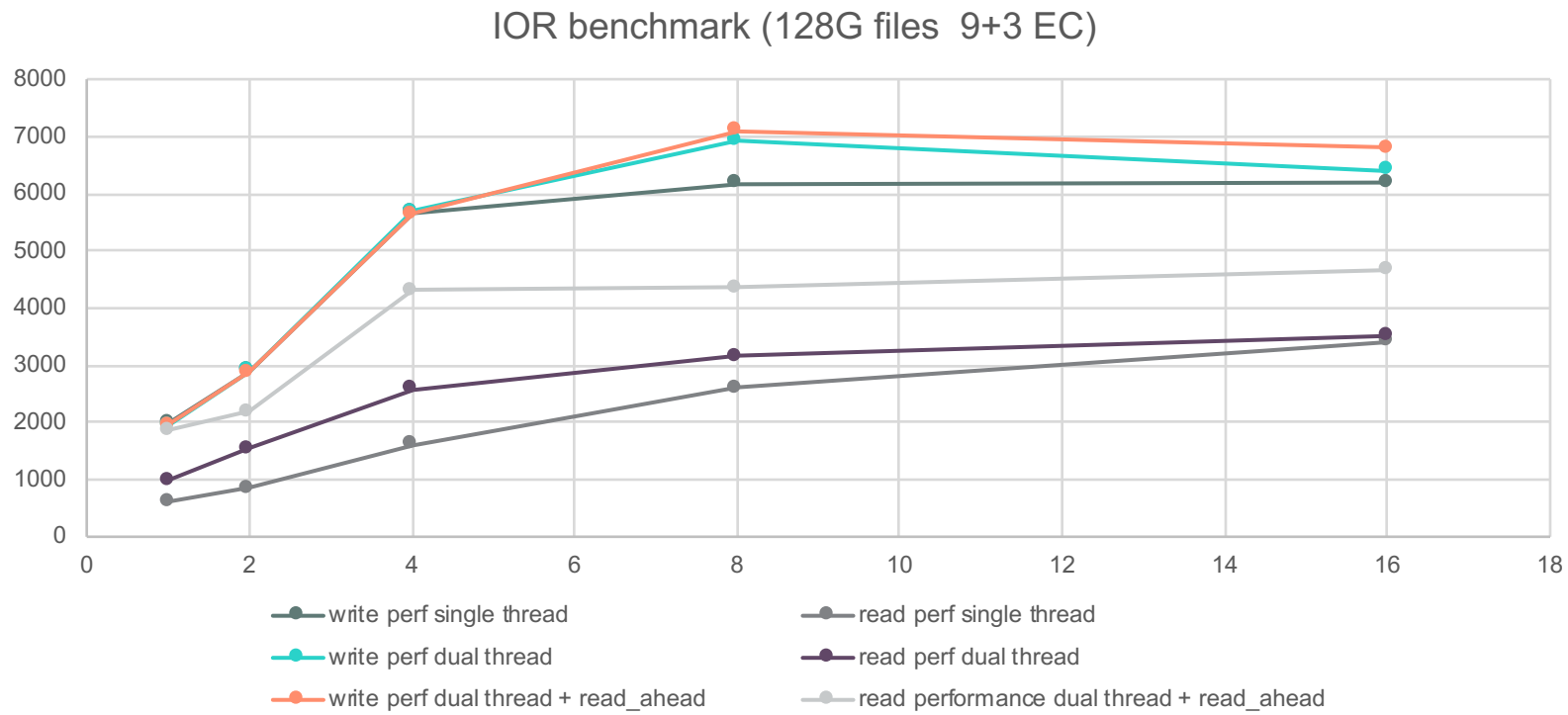
Thank You

cedric.milesi@hpe.com
alberto.galli@hpe.com

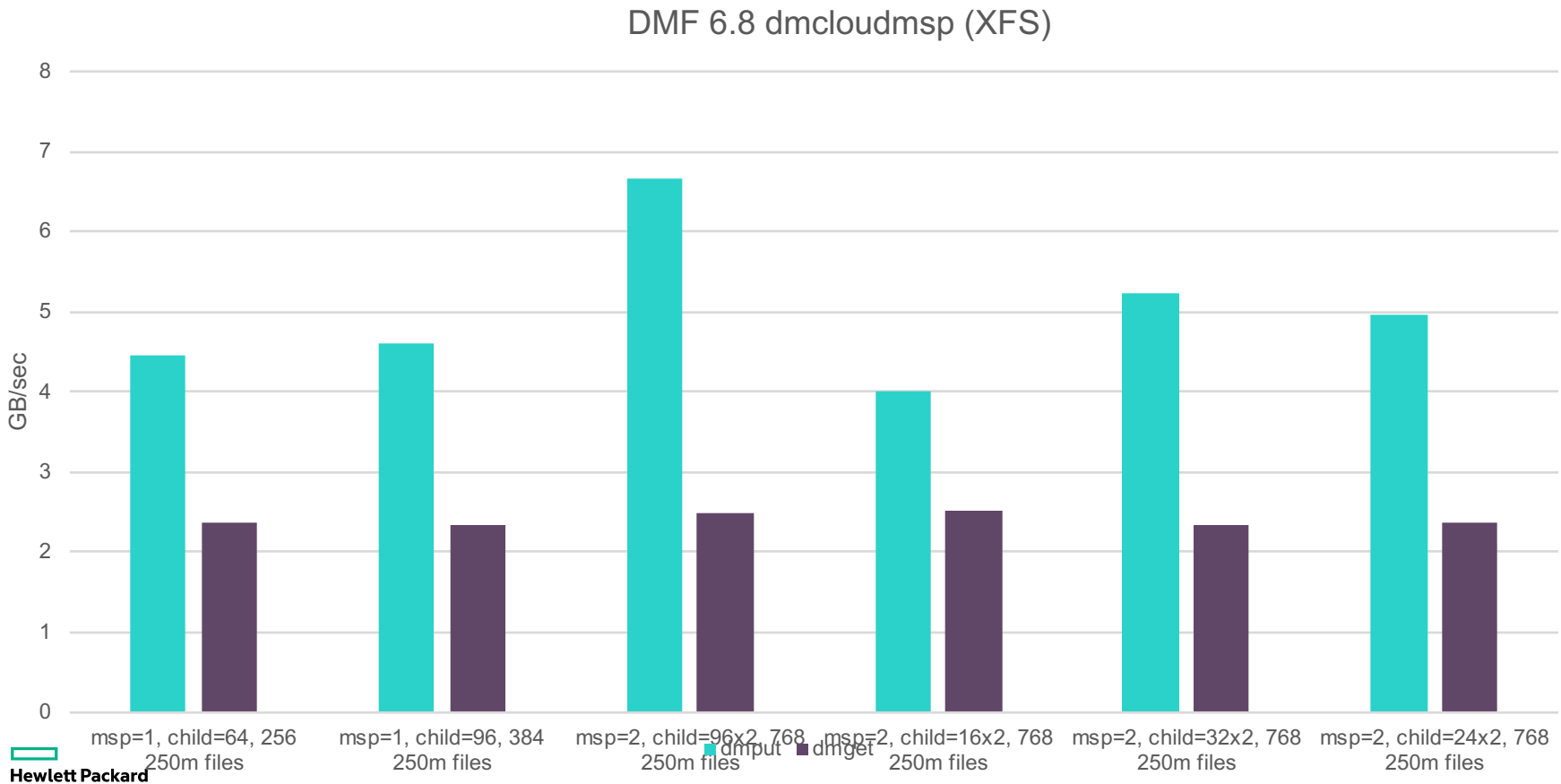
SUSE Enterprise Storage

Notes on the backend

CephFS capability



DMF v6 benchmark: 2 x RGW



Hewlett Packard Enterprise

Proof Of Concept outcome

- Ceph is a viable solution in terms of performance
- DMF + Ceph is a viable solution
- Basic configuration (EC scheme, network infrastructure, …) is known
  - EC: 9+3
  - Network Infrastructure: Seperated Public and Private network should be sufficient
  - Need NVMe in OSD
  - 1 x RGW is capable of 2GB/s

Just need to scale it

On site performance: S3 Write