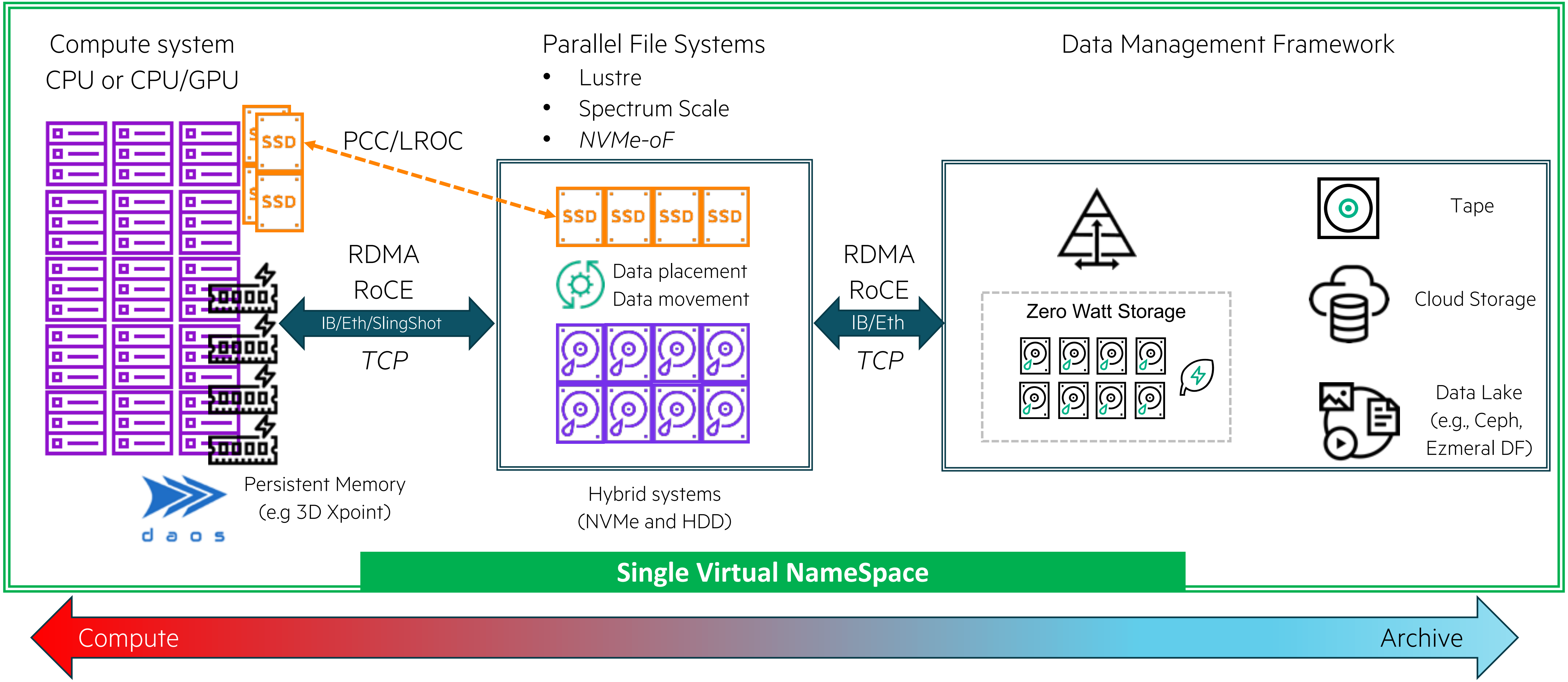
Data Mobility in HPC Storage

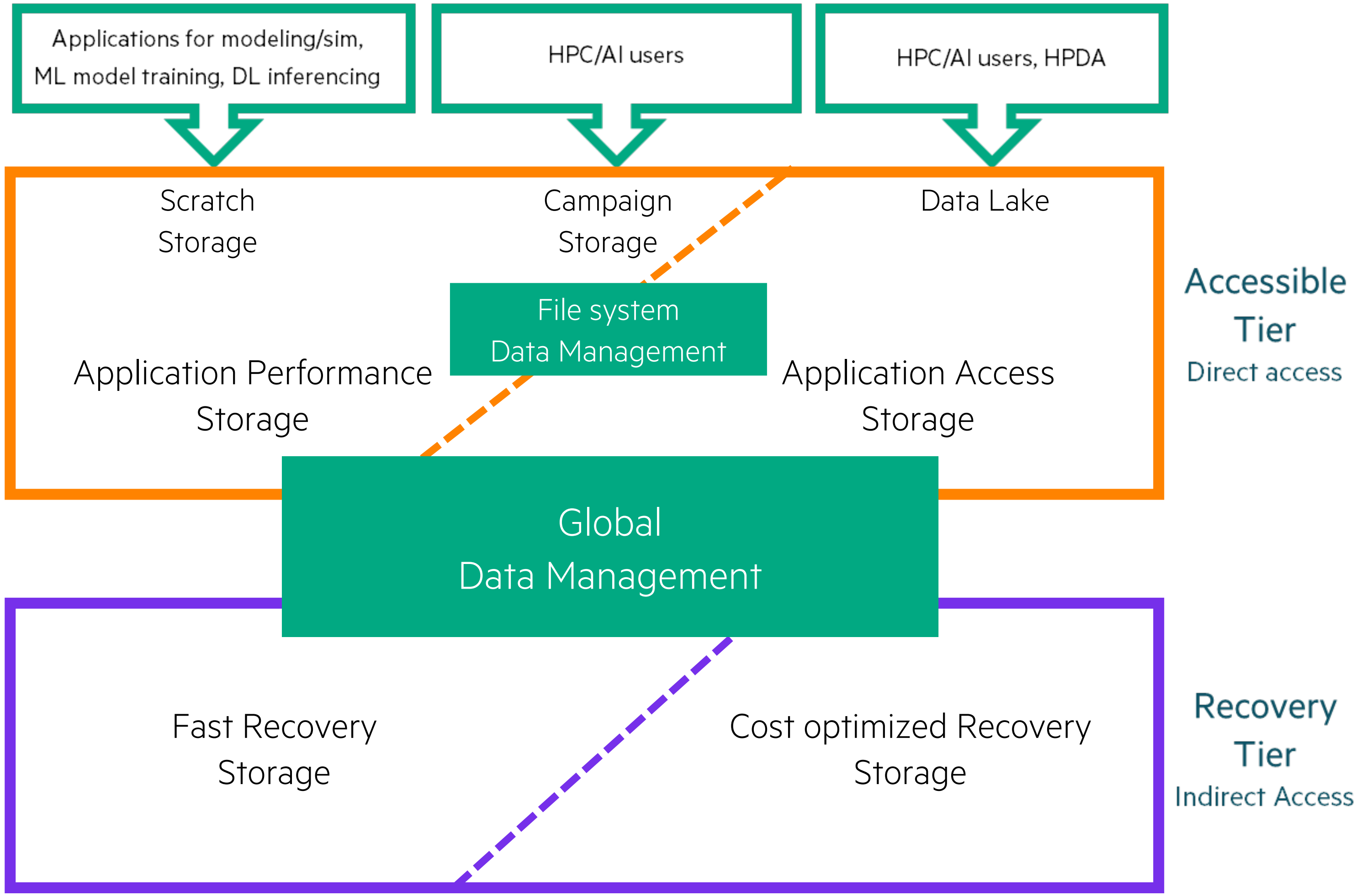
Torben Kling Petersen, PhD

Distinguished Technologist
Lead HPC Storage Architect - EMEA & APAC

THE "NEW" WORLD – TIERED STORAGE SOLUTIONS (ON PREM OR OFF …)



Why do we need Data Mobility?

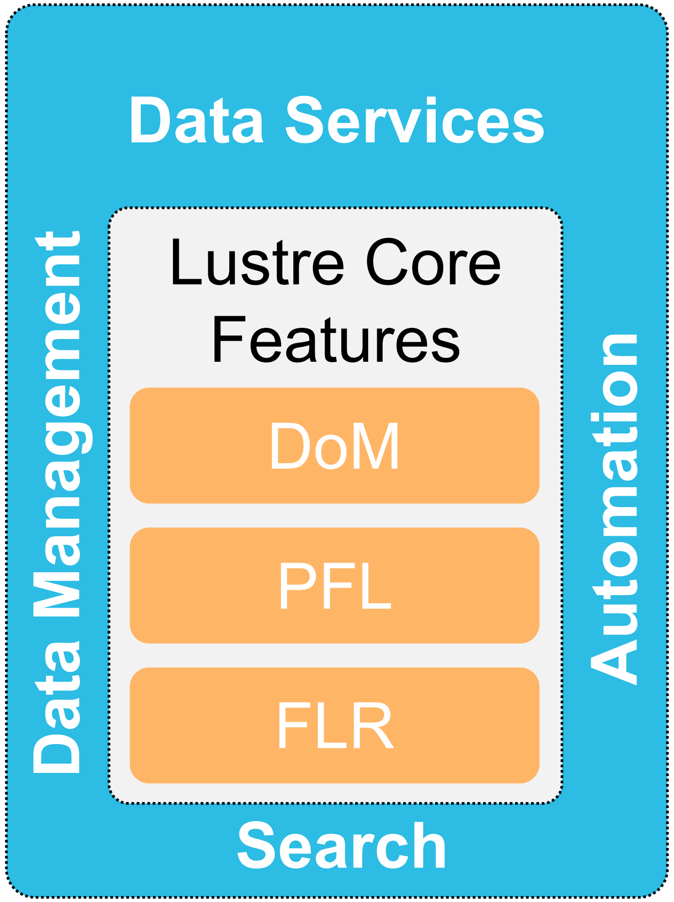


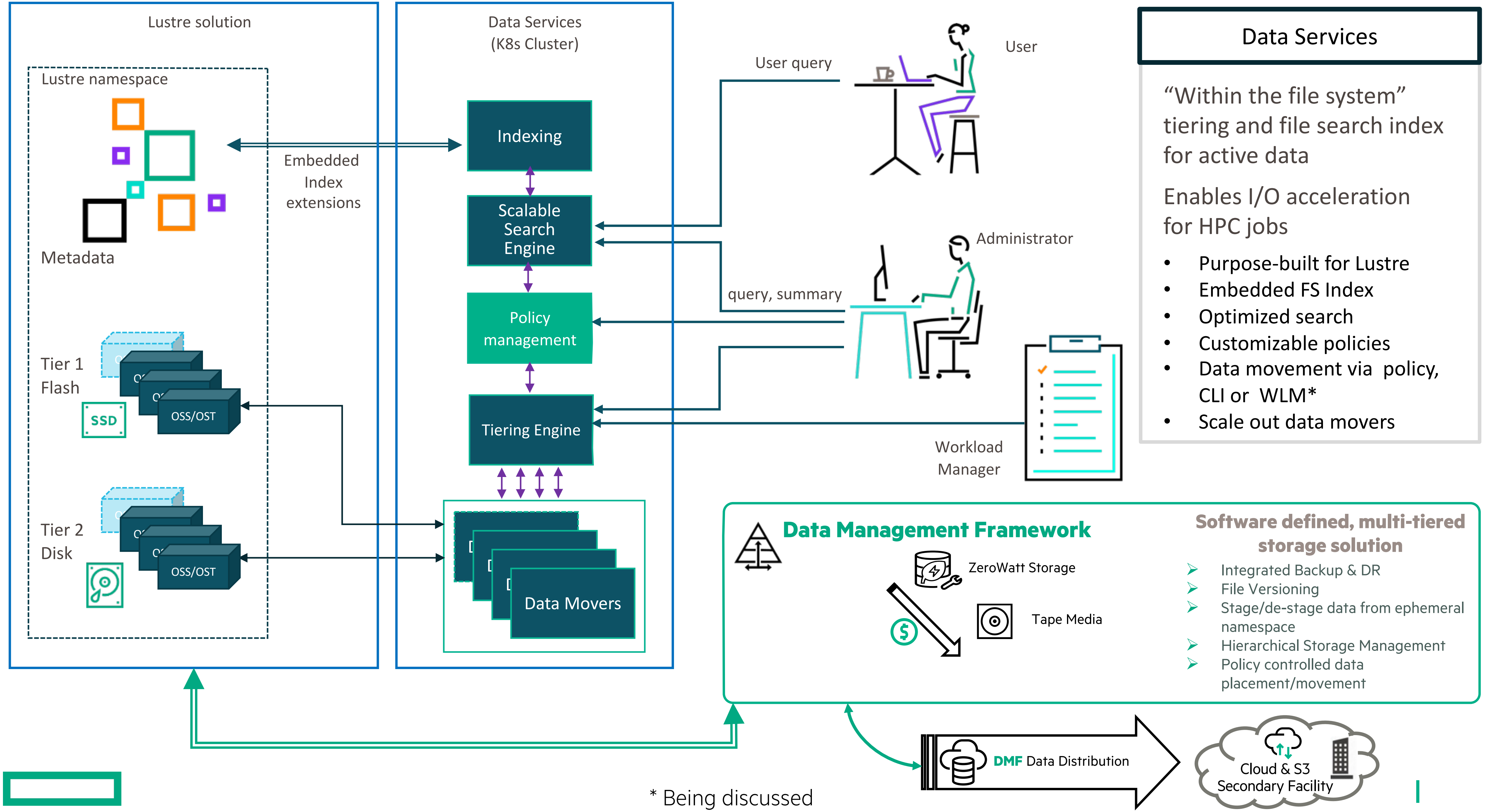
- Data management challenges are exacerbated in HPC
- Data is unstructured and is not managed by business systems: e.g. data base, email
- Data created by many sources: applications, sensors, technical instruments
- Data at volume, commonly in tens of petabytes
- Data exists in several states: e.g. hot, warm, cold
- Data has a blast radius that includes versions and recovery copies
- We need data management to:
  - Reduce the Total Cost of Curation
  - Inventory/locate data
  - Relocate data

Intra File System Data Mobility

INTRA FILE SYSTEM DATA SERVICES OVERVIEW

- Improved use of New Lustre Features
- Cohesiveness
  - Reduce complexity for users
  - No more component soup
- Scale
  - Move beyond scale limits of current solutions
  - Target petascale to exascale
- Integration
  - Direct integration with Lustre
  - Built-in management and monitoring
  - Workflow integration through workload managers

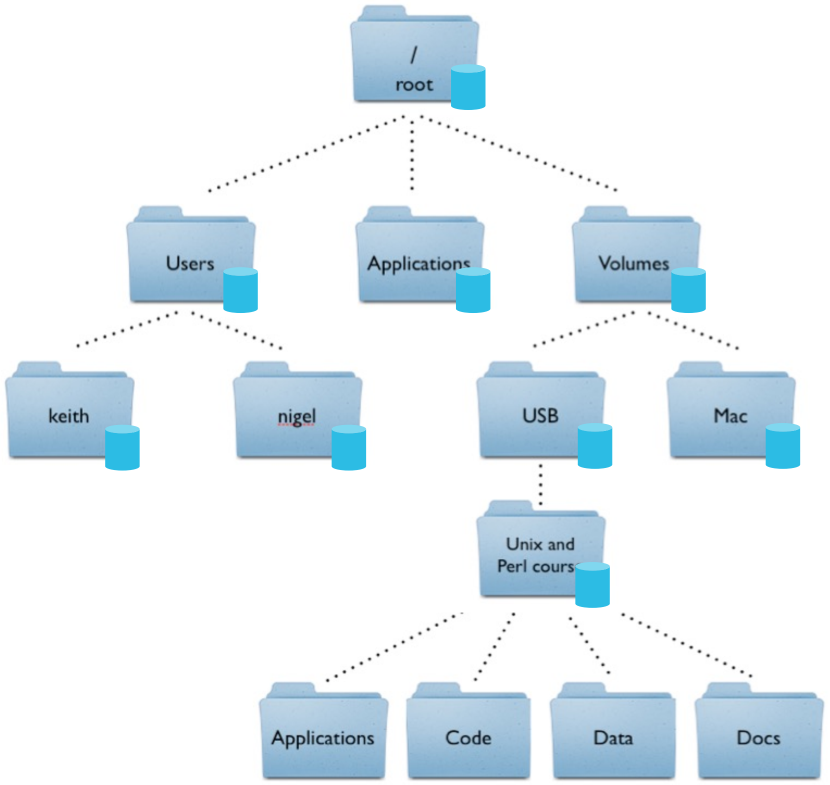




Lustre solution
Lustre namespace
Metadata
Embedded Index extensions
Tier 1 Flash
SSD
OSS/OST
Tier 2 Disk
OSS/OST
Data Services (K8s Cluster)
Indexing
Scalable Search Engine
Policy management
Tiering Engine
Data Movers
User query
User
query, summary
Administrator
Workload Manager
Data Services
"Within the file system" tiering and file search index for active data
Enables I/O acceleration for HPC jobs
• Purpose-built for Lustre
• Embedded FS Index
• Optimized search
• Customizable policies
• Data movement via policy, CLI or WLM*
• Scale out data movers
Data Management Framework
ZeroWatt Storage
Tape Media
Software defined, multi-tiered storage solution
➢ Integrated Backup & DR
➢ File Versioning
➢ Stage/de-stage data from ephemeral namespace
➢ Hierarchical Storage Management
➢ Policy controlled data placement/movement
DMF Data Distribution
Cloud & S3
Secondary Facility
* Being discussed

DATA SERVICES **SCALABLE SEARCH***

- A better way to search…
  - Linear search is too slow
  - Singular search database is unwieldy
  - Scalable within a tree and across trees
- Distributed search
  - Distributed indices across file system
  - Parallel cross file system search
  - Summarization of trees to optimize search
  - Adheres to POSIX permissions for search
- Queries & Reports
  - Rapid generation of full reports based on any searchable criteria
  - Fully scriptable queries on usage based on users, groups etc.
  - Use standard sql syntax



* See Nathan Rutman's presentation "Fast Scanning and Scalable Search" from LAD2019

POLICY ENGINE

- A policy defines an action to be taken on set of candidate files that match a set of selection criteria
- Requires a triggering event, such as a particular state of the filesystem or a simple timer
- Uses established RobinHood syntax

```
fileclass largeflash {
        definition { size > 100MB and pool = flash}
}

flash_maintenance_rules{
        rule migrate_large {
                target_fileclass = largeflash;
                action = migrate;
                action_params {
                        template = .cray/cds/template/disk_pool;
                }
                condition { last_access > 2d }
        }
}
# trigger FM policy if any OST in pool 'flash' exceeds 85% of disk usage.
flash_maintenance_trigger {
        check_interval = 600;
        trigger_on = pool_usage(flash);
        high_threshold_pct = 85%;
}
define_policy flash_maintenance {
        default_action = migrate;
}
```

QUERY – FILE PURGING

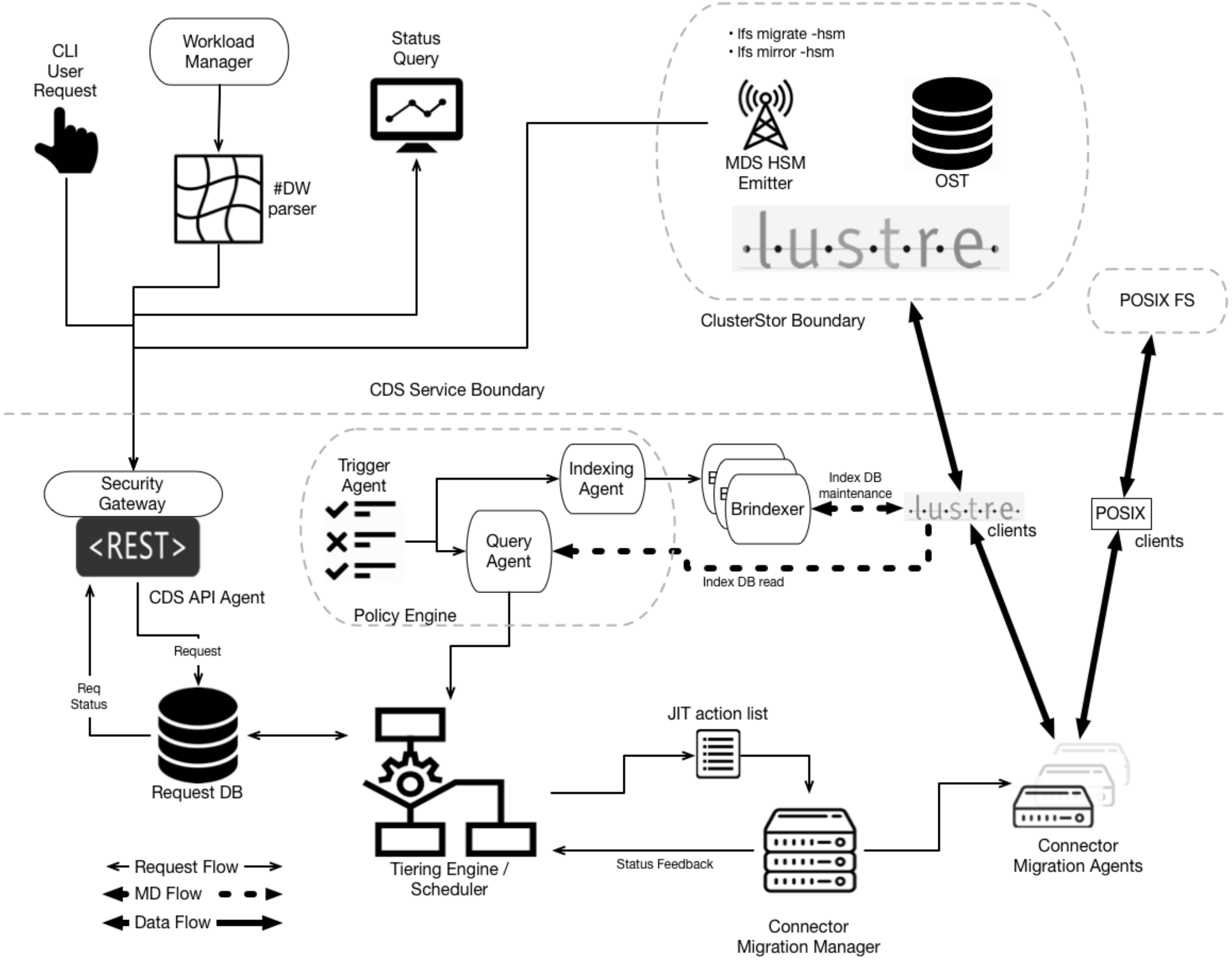
- Essentially a query with added delete

```
client# /opt/cray/brindexer/bin/query -q "select %s from %s where size > 30000" --delete /lus
Delete file:/lus/next copy/myfile
Delete file:/lus/next copy/3file
Delete file:/lus/next copy/another
Delete file:/lus/tiny/onefile-0
Delete file:/lus/tiny/onefile-0
Delete file:/lus/next/3file
Delete file:/lus/next/another
Delete file:/lus/next/myfile
Delete file:/lus/level1/level2/tiny copy/onefile
Delete file:/lus/level1/level2/tiny copy/onefile-0
Delete file:/lus/next copy/tiny/onefile-0 copy
Delete file:/lus/next copy/tiny/onefile-0
Delete file:/lus/tiny/next/3file
Delete file:/lus/tiny/next/another
```

```
client# /opt/cray/ /brindexer/bin/query -q "select %s from %s where name like '%%.tmp' and atime > 86400" --delete /lus
Delete file:/lus/next copy/myfile.tmp
Delete file:/lus/next copy/3file.tmp
Delete file:/lus/next copy/another.tmp
Delete file:/lus/tiny/onefile-0.tmp
Delete file:/lus/tiny/onefile-0.tmp
Delete file:/lus/next/3file.tmp
Delete file:/lus/next/another.tmp
Delete file:/lus/next/myfile.tmp
Delete file:/lus/level1/level2/tiny copy/onefile.tmp
Delete file:/lus/level1/level2/tiny copy/onefile-0.tmp
Delete file:/lus/next copy/tiny/onefile-0 copy.tmp
Delete file:/lus/tiny/next/another.tmp
```

SOFTWARE ARCHITECTURE

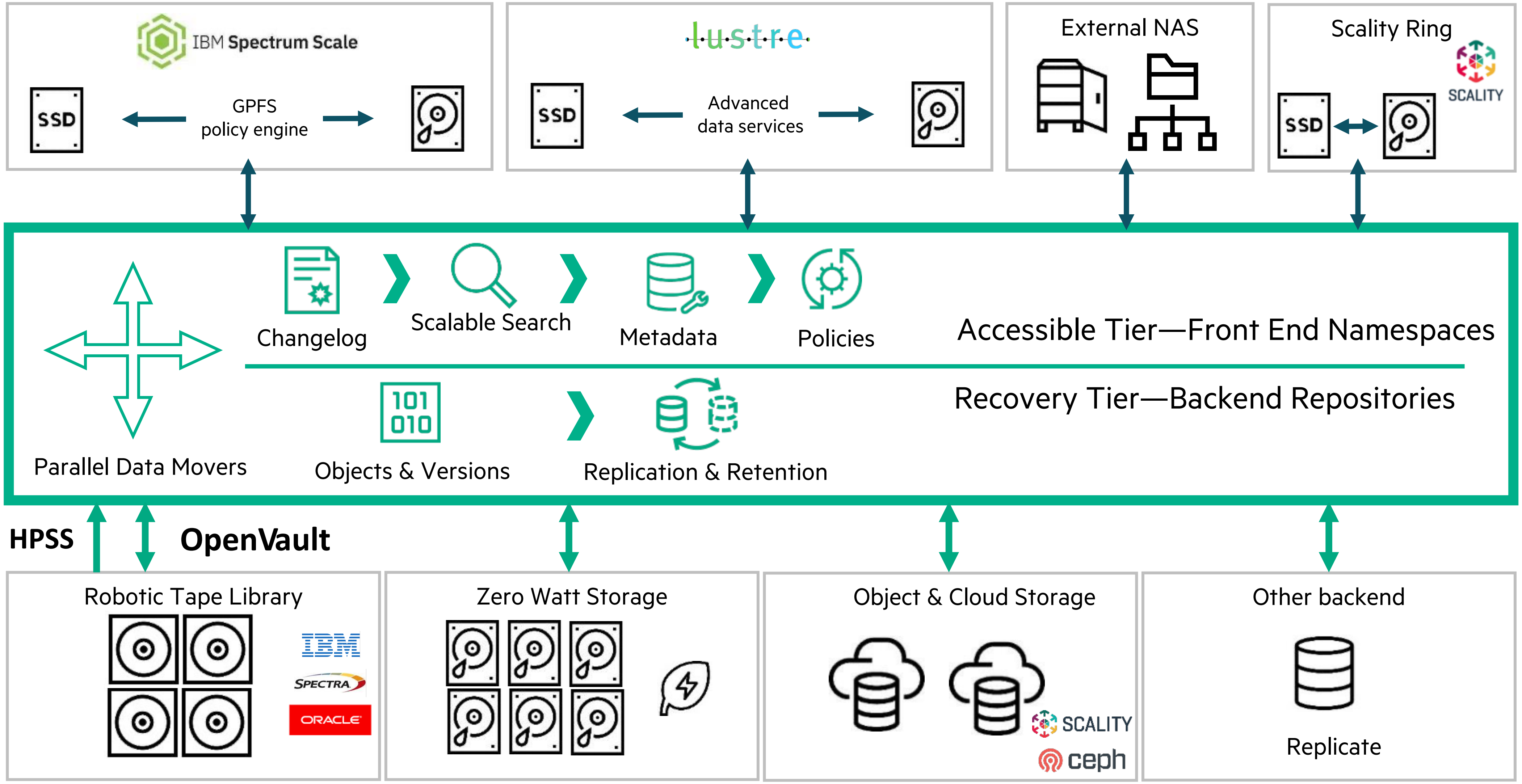
- **Lustre client** – sends data movement requests via the ClusterStor Emitter software, running on the Metadata Servers (MDS)
- **API Agent** – provides a RESTful interface to initiate and data movement requests from Lustre, command-line tools, etc.
- **Policy Engine** – processes policies defined as text files (RobinHood syntax) and initiates data movement and other operations through the Tiering Engine
- **Tiering Engine** – processes data movement and indexing actions; orchestrates the Data Movers (Connector)
- **Scalable Data Movers** (aka Connector)–executes data movement requests issued by the Tiering Engine via standard Lustre client



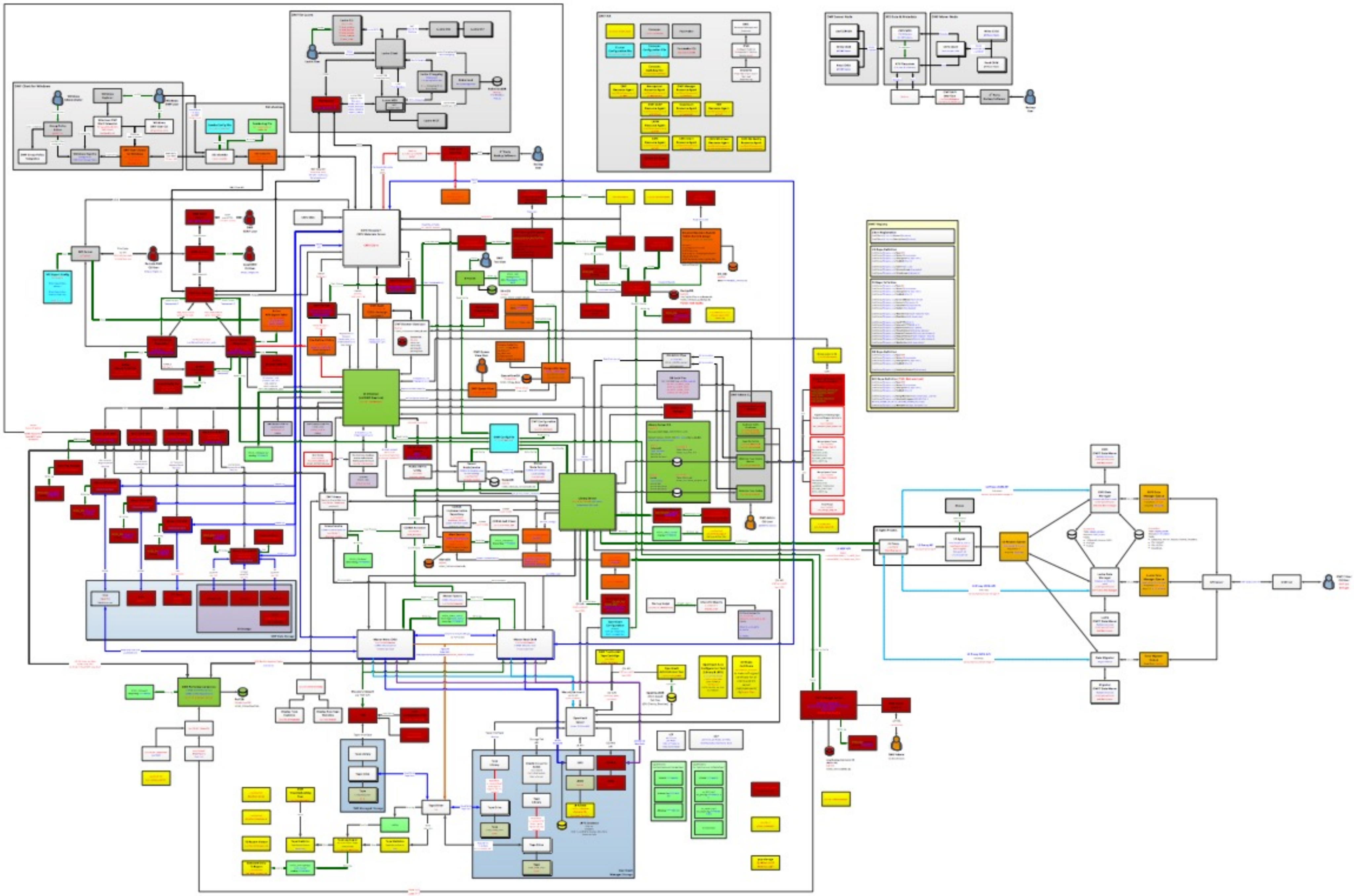
Inter File System Data Mobility

DATA MANAGEMENT FRAMEWORK

Data management solution for parallel storage across heterogenous namespaces



DMF 7.x Library Server Diagram



CORE TECHNOLOGY FRAMEWORKS

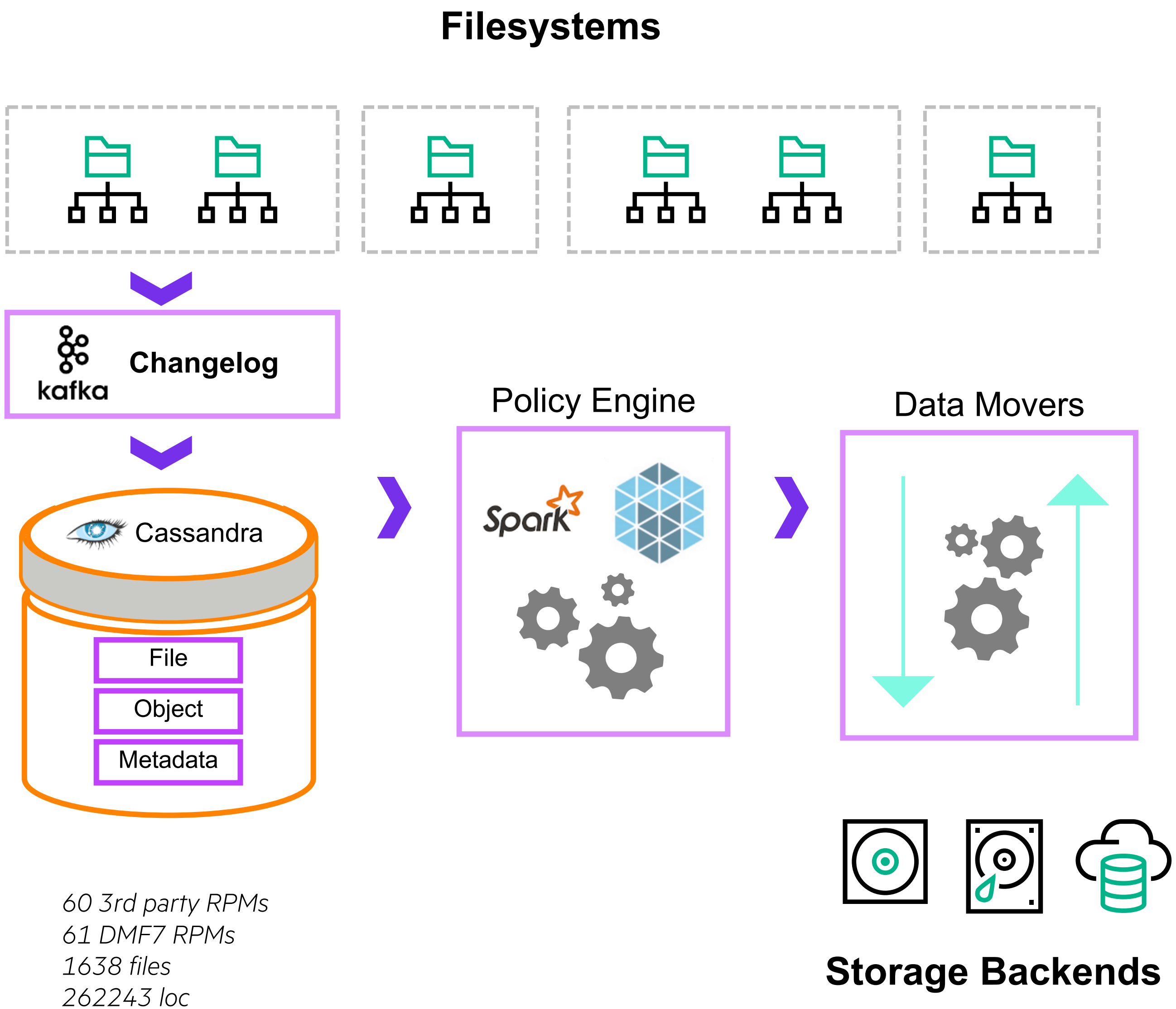
- **Cassandra Database**
  - Distributed NoSQL DMBS for big data
  - HA with no Single Point Of Failure
  - Tunable Consistency
- **Redis**
  - Distributed in-memory key-value store
  - Foundation for AnyQ – DMF Queueing Framework
- **Spark**
  - Policy Engine with Custom DSL
  - Distributed Metadata Queries
- **Kafka**
  - Changelog Event Processing for GPFS and EXFS
  - Filesystem Synchronous Event Processing
- **Mesos**
  - Cluster Management Framework (used by Spark)
  - Task-based Application Framework with Scheduler API
  - Partial Containerization in DMF

EVOLVING DMF

To More Advanced Data Management

Manage more data workflows

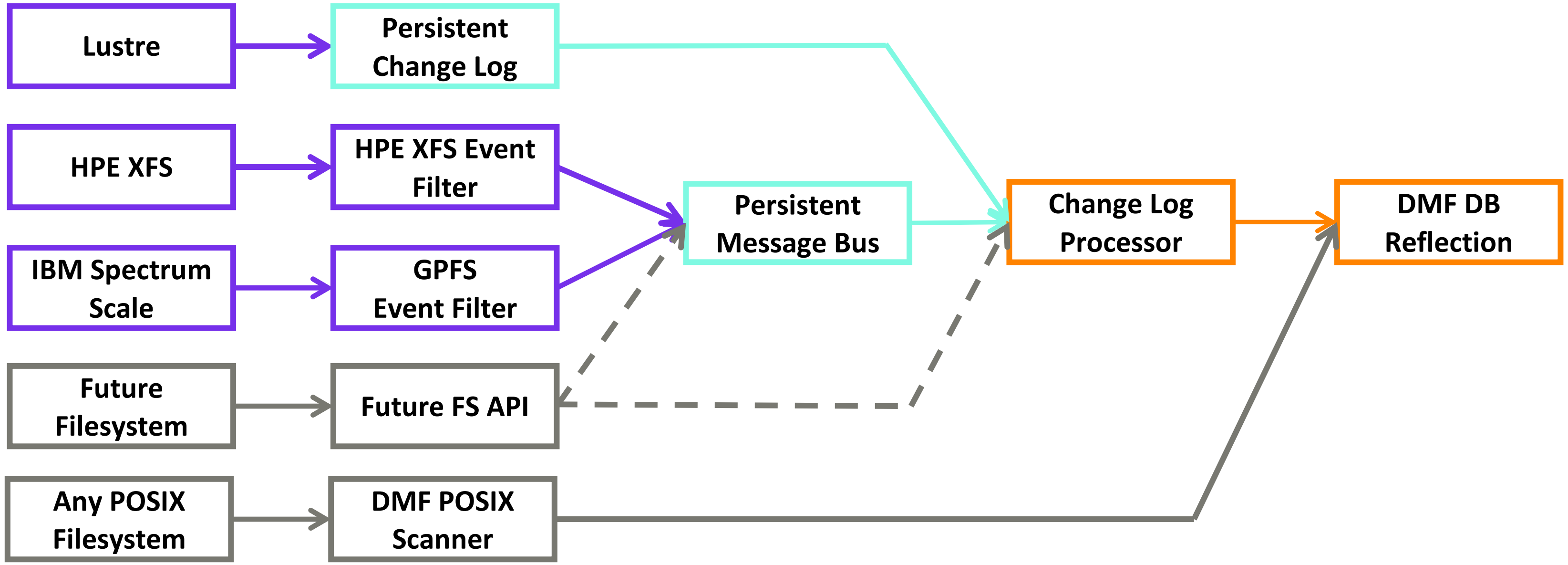
- Retains transparent tiering (HSM)
- Captures and stores filesystem metadata
- Provides metadata queries
- Provides metada-driven policies
- Versions data
- Can destage files and stage data from backend into filesystems
- Configures and creates namespaces
- Delivers scalability and HA
- Modular architecture – can accommodate multiple filesystem types (currently Lustre, GPFS, EXFS, Generic POSIX, and S3)



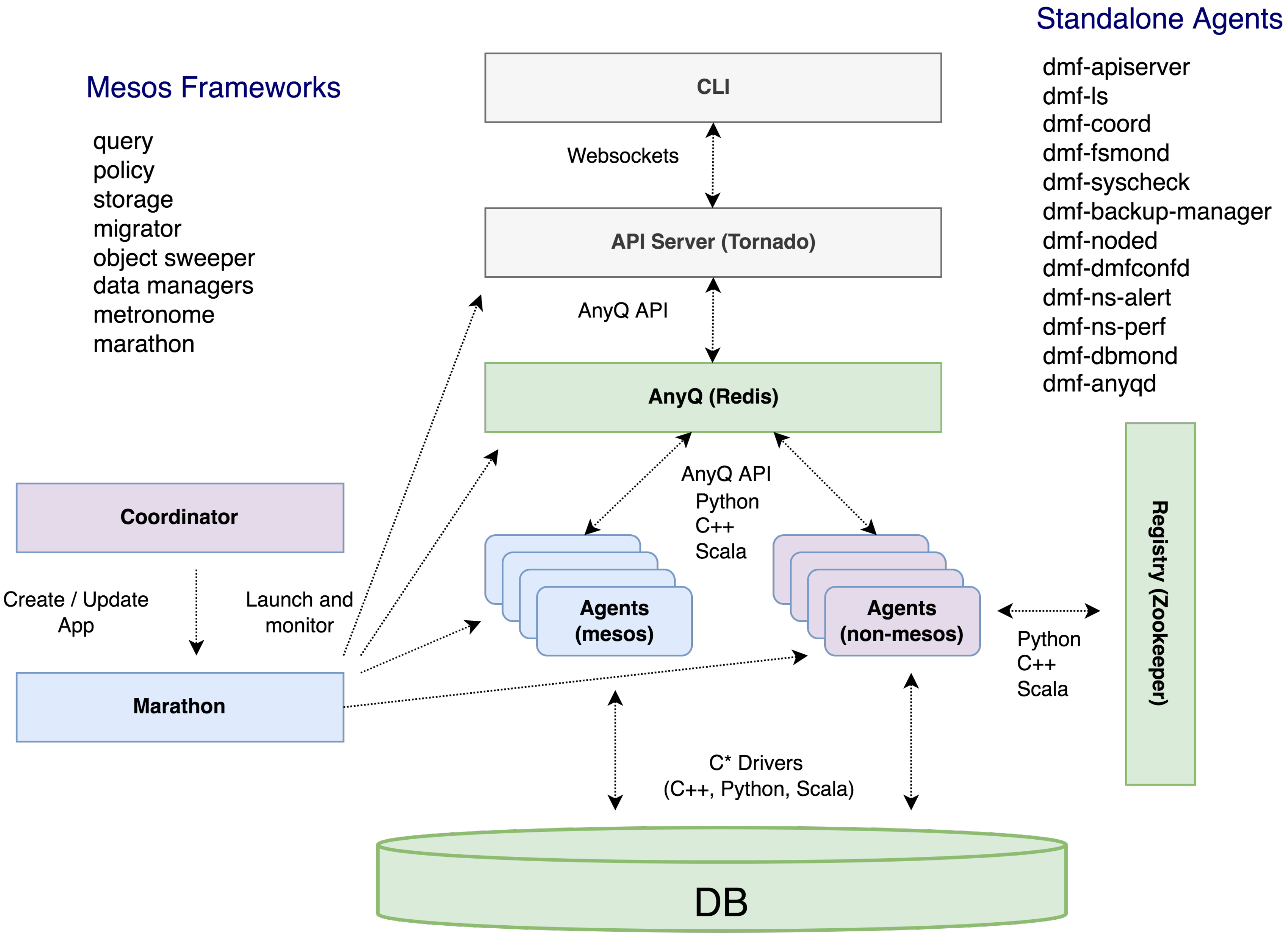
60 3rd party RPMs
61 DMF7 RPMs
1638 files
262243 loc

DMF 7 NAMESPACE REFLECTION & CHANGE LOG

- For HPE XFS & IBM Spectrum Scale:
  - Use DMAPI events to drive filesystem change log and filesystem reflection
  - Buffer filesystem events in scalable persistent message bus (Kafka)
  - 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
- Other filesystems support:
  - Makes DMF front-end filesystem independent
  - Persistent message bus use depends on filesystem API
  - Any POSIX filesystem can be simply re-scanned at any time
  - Unified DMF policy engine for all filesystem types



Architecture



SUMMARY

- Automated and resilient tools for data mobility (both intra and inter filesystem) is key to growth.
- One size does not fit all …
  - Modular deployment
  - Add-ons rather than monolithic
- Extensive user customization
  - Policy driven data mobility
- Database or not ??
  - Casandra vs index files

THANK YOU

(for listening to a madmans ramblings ….)

tkp@hpe.com