



Hewlett Packard  
Enterprise

# DMFUG 2019

DMF7 Database In-depth

Zsolt Ferenczy



# Confidentiality Notice

- **The information contained in this presentation** is proprietary to Hewlett Packard Enterprise (HPE) Company and is offered in confidence, subject to the terms and conditions of a Confidential Disclosure Agreement
- **HPE makes no warranties regarding the accuracy of this information.** This document contains forward looking statements regarding future operations, product development, product capabilities and availability dates. This information is subject to substantial uncertainties and is subject to change at any time without prior notification. Statements contained in this document concerning these matters only reflect Hewlett-Packard Enterprise's predictions and / or expectations as of the date of this document and actual results and future plans of Hewlett-Packard Enterprise may differ significantly as a result of, among other things, changes in product strategy resulting from technological, internal corporate, market and other changes. This is not a commitment to deliver any material, code or functionality and should not be relied upon in making purchasing decisions.



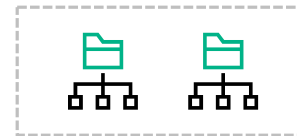
# Evolving DMF

From Traditional Archive System ...

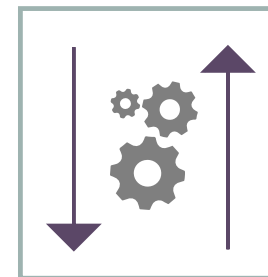
## Ye Olde HSM

- Transparently Tiers Data
  - Scans Filesystem
  - Migrates files to free up space
  - Recalls on user access or via GET command
- Focuses on Space Management

## Static Filesystems



DMF 6



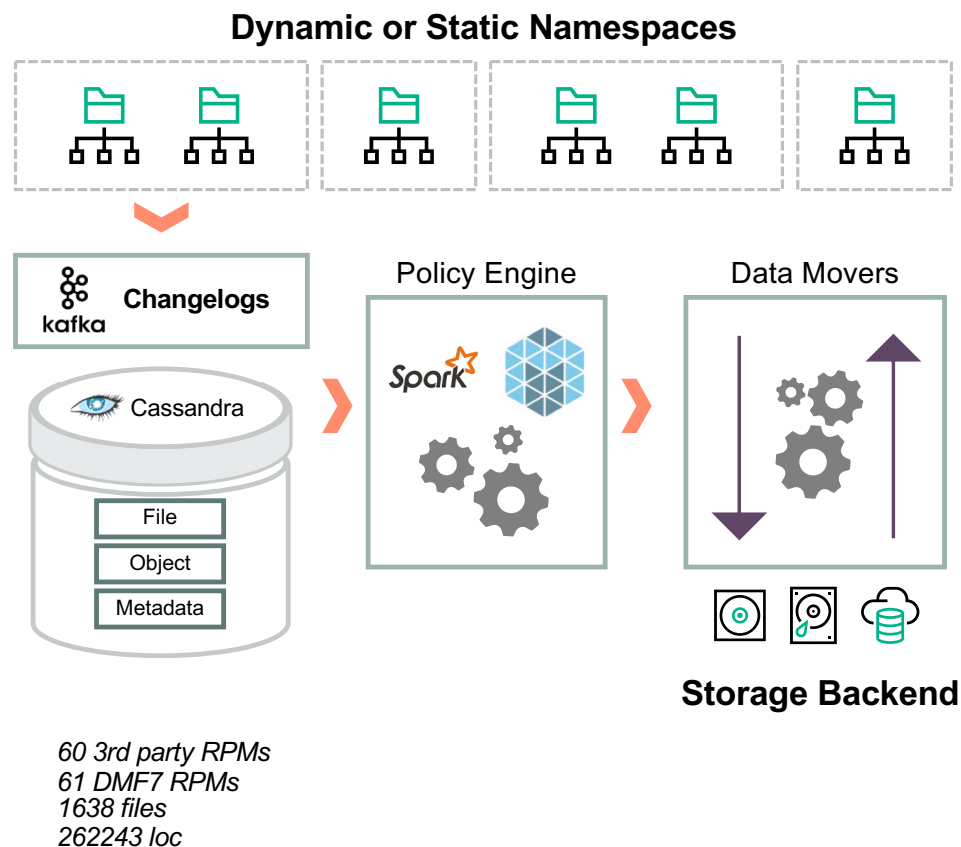
## Storage Backend

# Evolving DMF

## ... To Data Curator

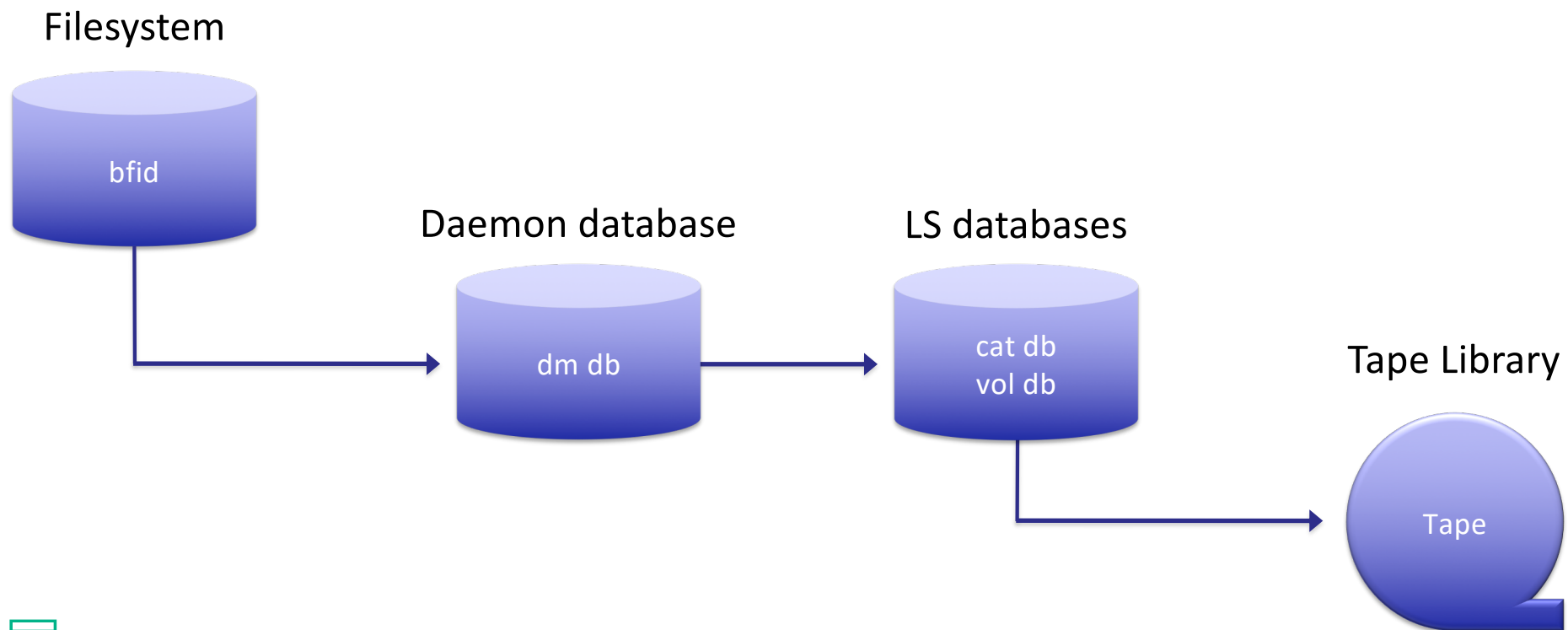
### Manage more data workflows (IML)

- Retains transparent tiering (HSM)
- Captures and stores filesystem metadata
- Provides metadata queries
- Provides metadata-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



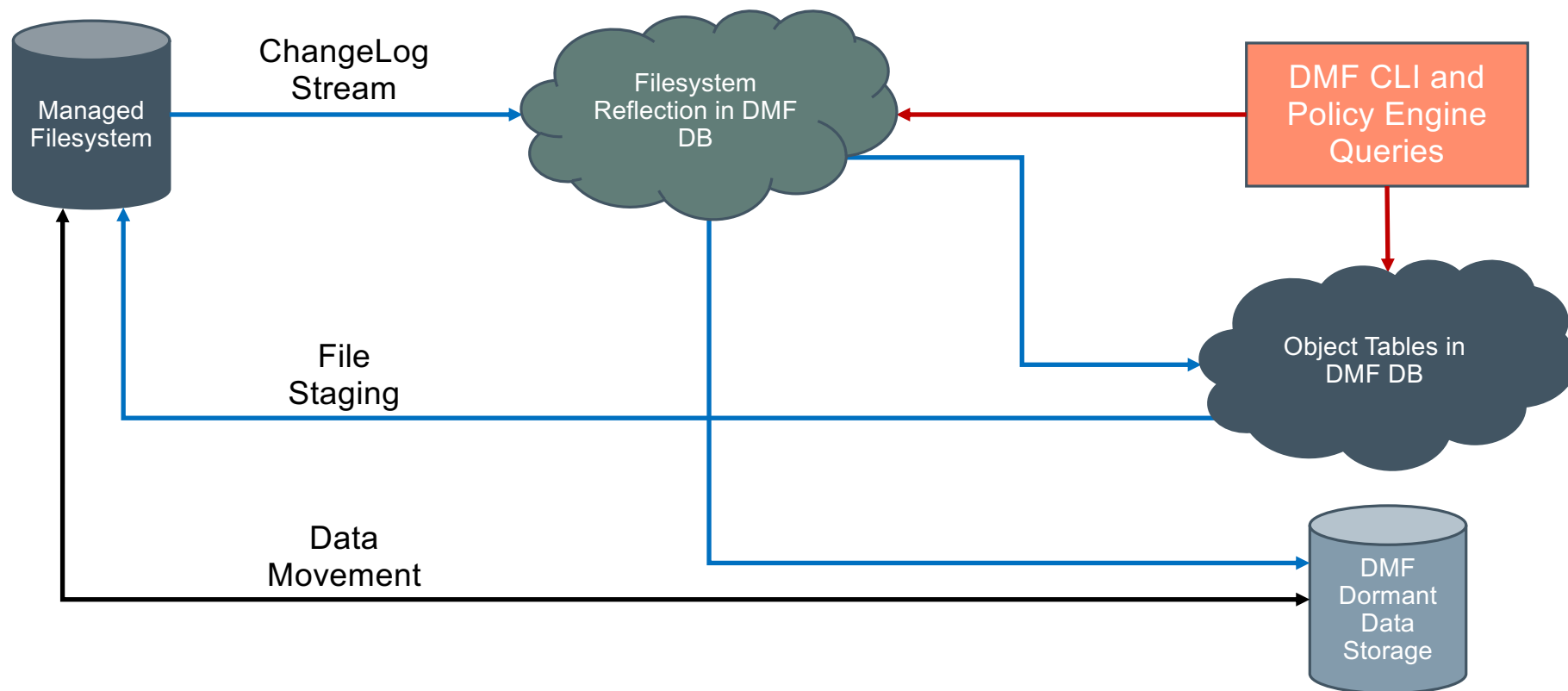
# Evolving DMF

From a Simple Data Model ...



# Evolving DMF

... To Full Metadata Management



---

# System-Wide Components

## Apache™ Cassandra Database

- Open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure <http://cassandra.apache.org/>
- De-centralized, durable, fast, elastic, fault tolerant, proven
  - Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data
  - Instagram, eBay, GitHub, Netflix and many others use Cassandra
- BASE Architecture
  - Basically Available, Soft state, Eventual consistency
    - Requests can fail or data may be changing
    - System state changes over time
    - If input stops, the state eventually reaches consistency
- CQL, cqlsh
  - Column family
  - No joins or subqueries



# System-Wide Components

## Cassandra Database

- DMF7 DB has 2 main keyspaces
  - Filesystem Reflection
    - One record per inode in the managed filesystem
  - Object
    - One record per managed version

Purpose	Keyspace	Description	Tables
Filesystem metadata	dmf_fs_<filesystem-id> per filesystem	Reflects current state of the filesystem provided by filesystem scanner, changelog processor and pathfinder. Includes working storage for capturing staging query results	file_inode, file_name, file_xattr, file_dirs, managed_file, file_to_object, file_policy, file_policy_state, file_staging, filesystem_usage
Backend DMF objects	dmf_object	Contains timestamped snapshot of filesystem metadata and pointers to object's data in external repositories	object_version, data_set, object_metadata, object_policy, object_policy_state, object_chunk, object_attr



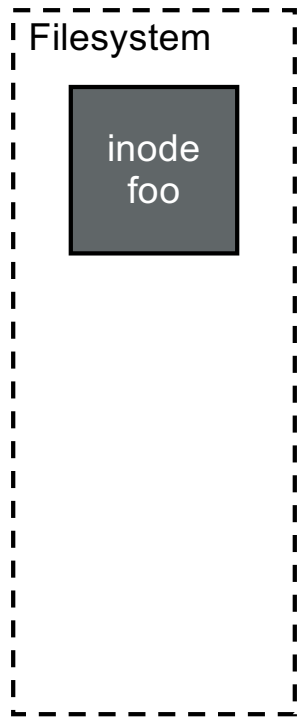
---

## DMF7 Filesystem Reflection

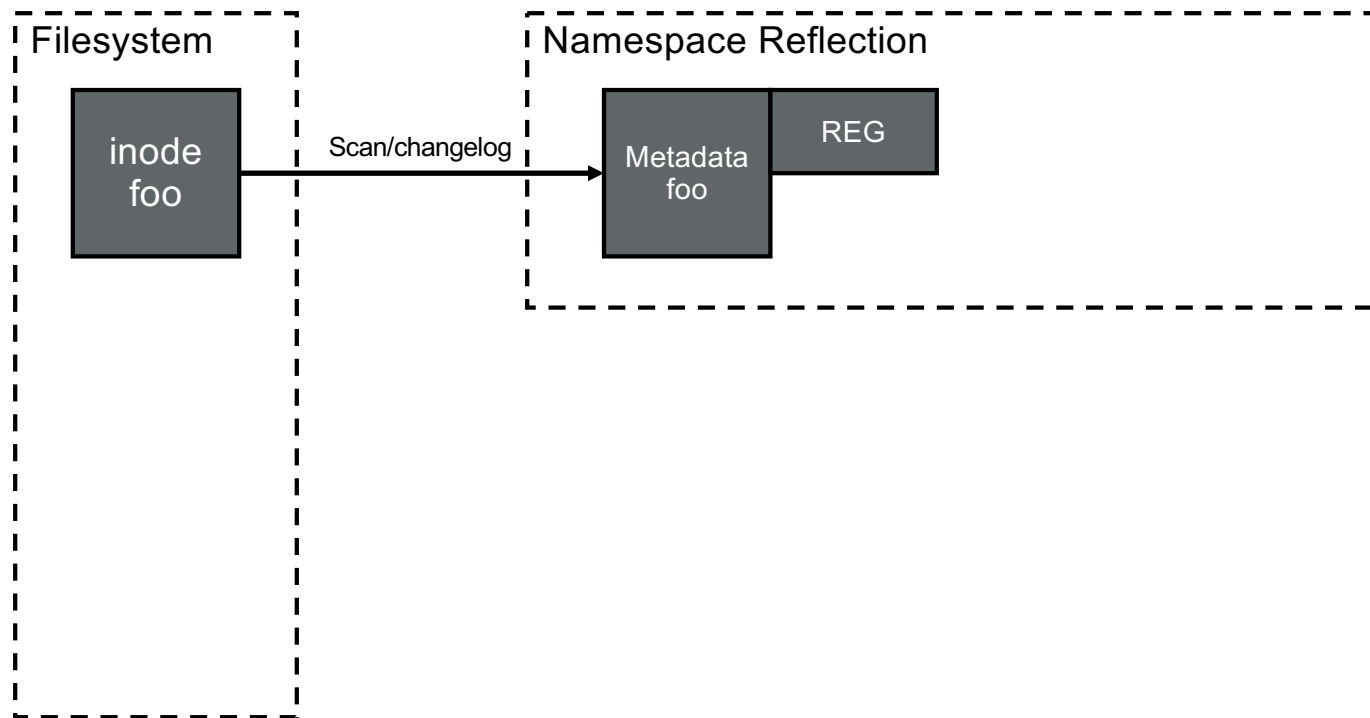
- Independent keyspace per filesystem namespace
- Stores metadata for every inode in the managed filesystem
- Populated by scan and changelog stream
- Stores the HSM state for every inode
- Stores a pointer to the staged object version for every managed file
- Stores POSIX inode, xattr, filename, and directory metadata separately
- Directory hierarchy is maintained by a dedicated component called the DMF7 PathFinder
- All query and policy operations use only the reflection
- DMF7 is not looking at the filesystem itself to make policy decisions

## DMF7 DB: A File foo

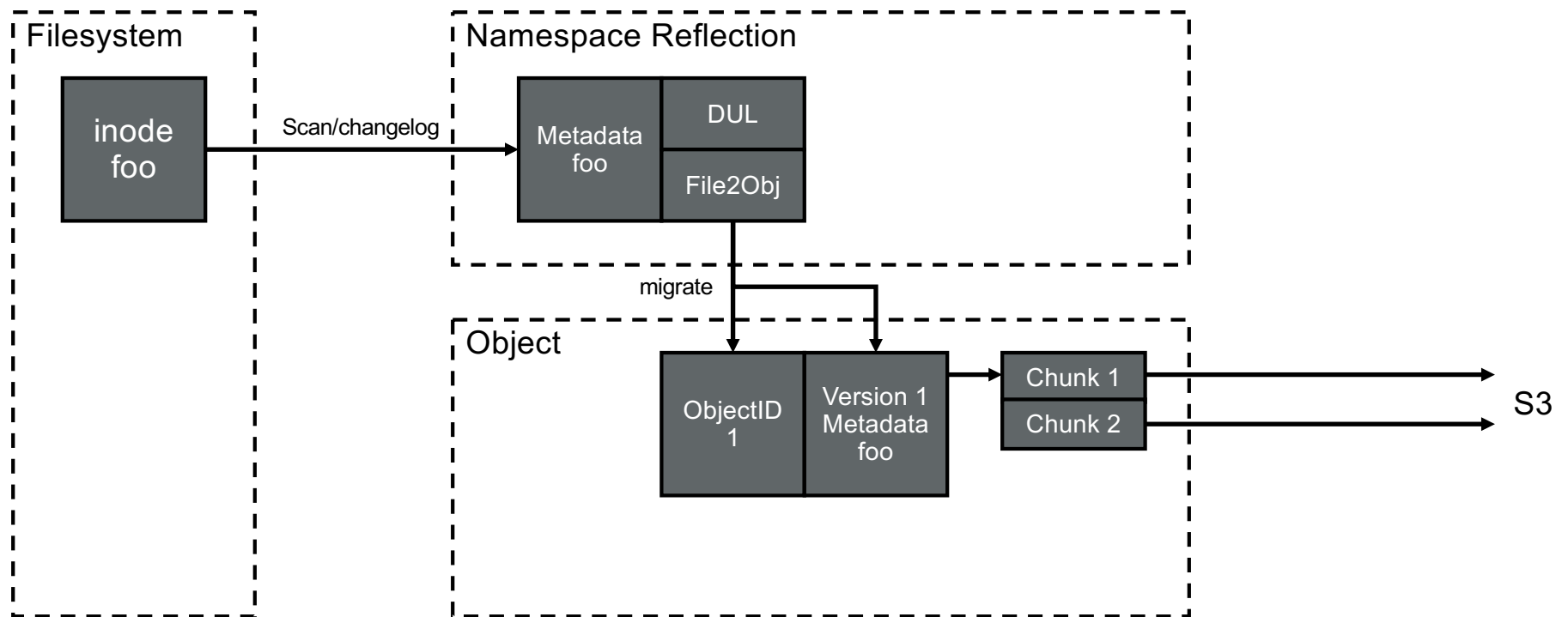
---



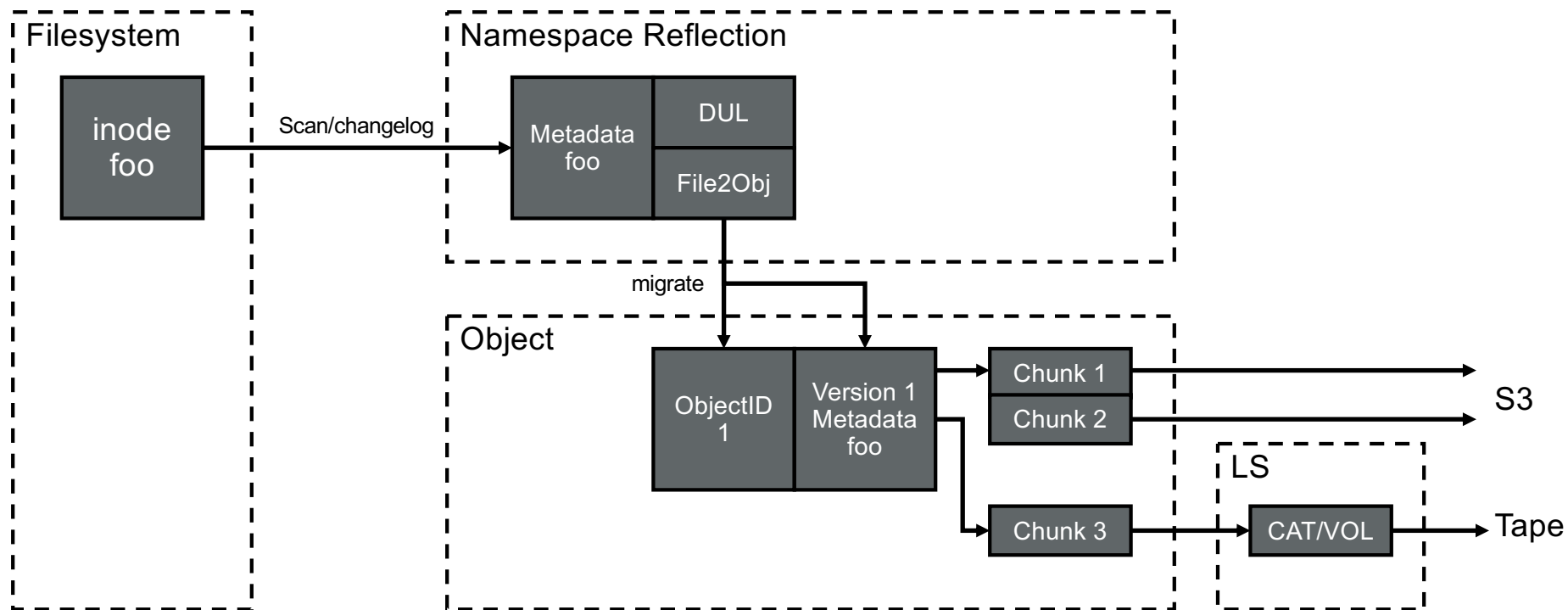
## DMF7 DB: A REG File foo in a DMF Managed Namespace



## DMF7 DB: DUL File foo with One Copy in S3

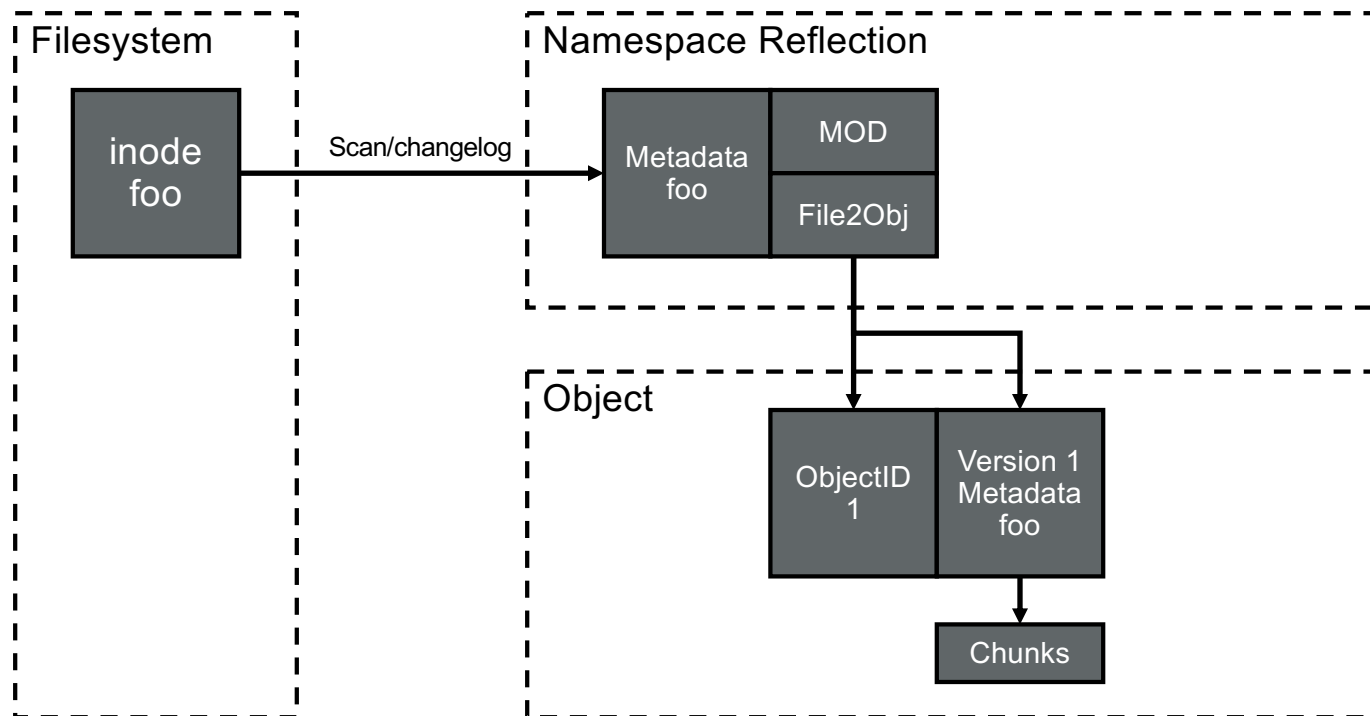


## DMF7 DB: DUL File foo with Two Copies

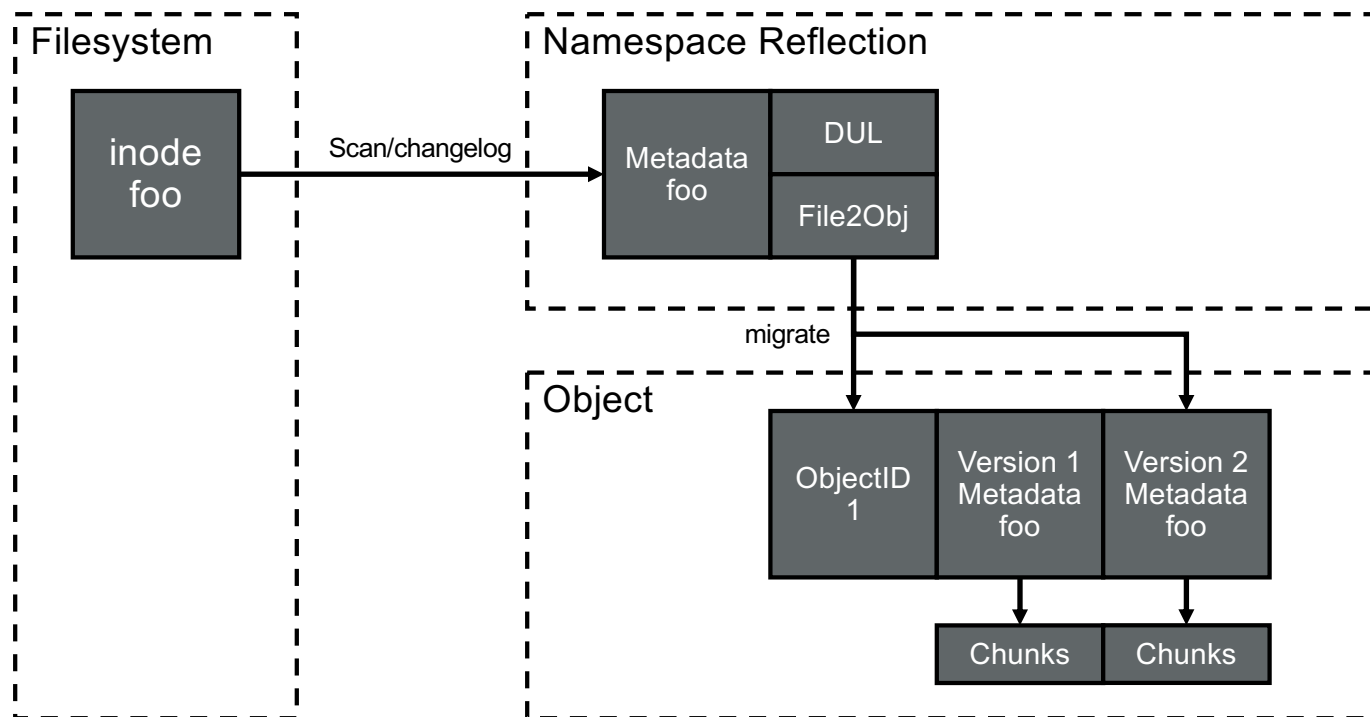




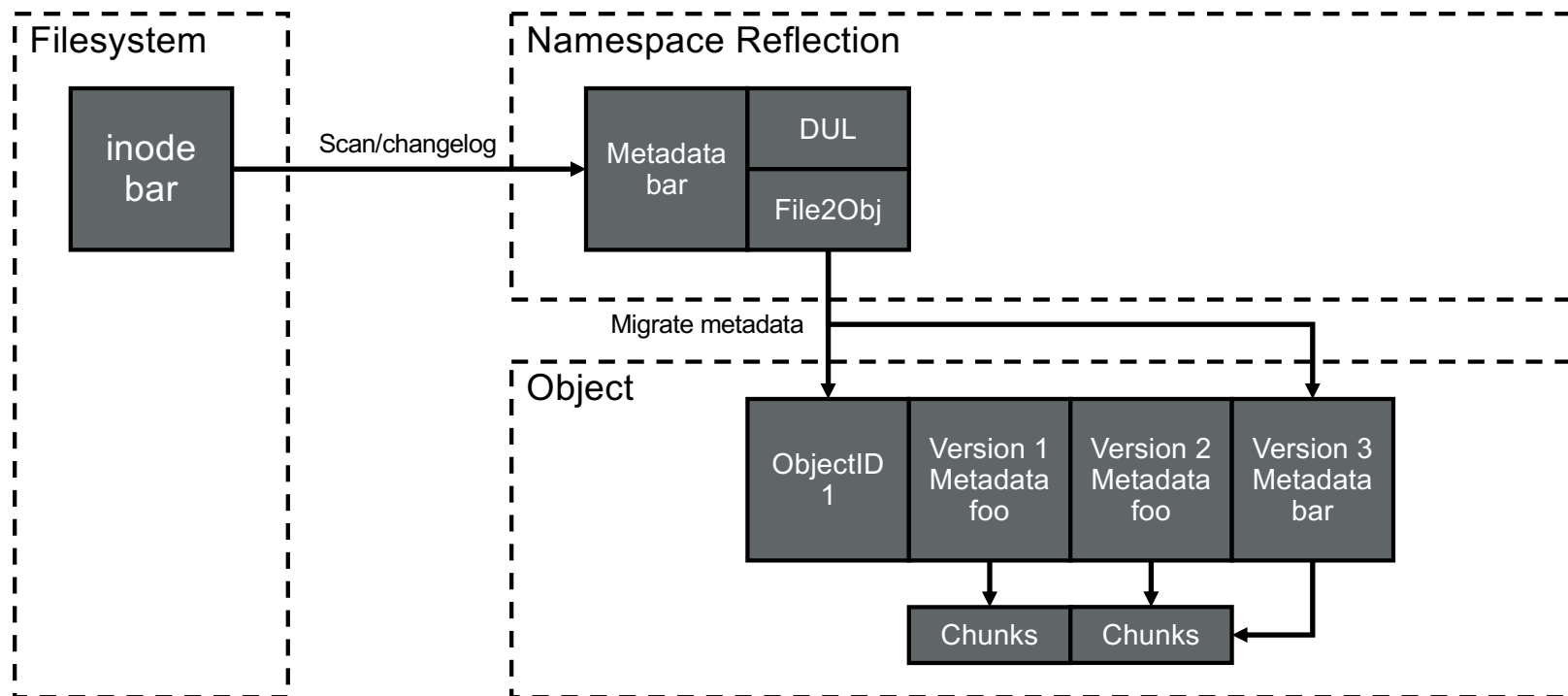
## DMF7 DB: foo is Modified



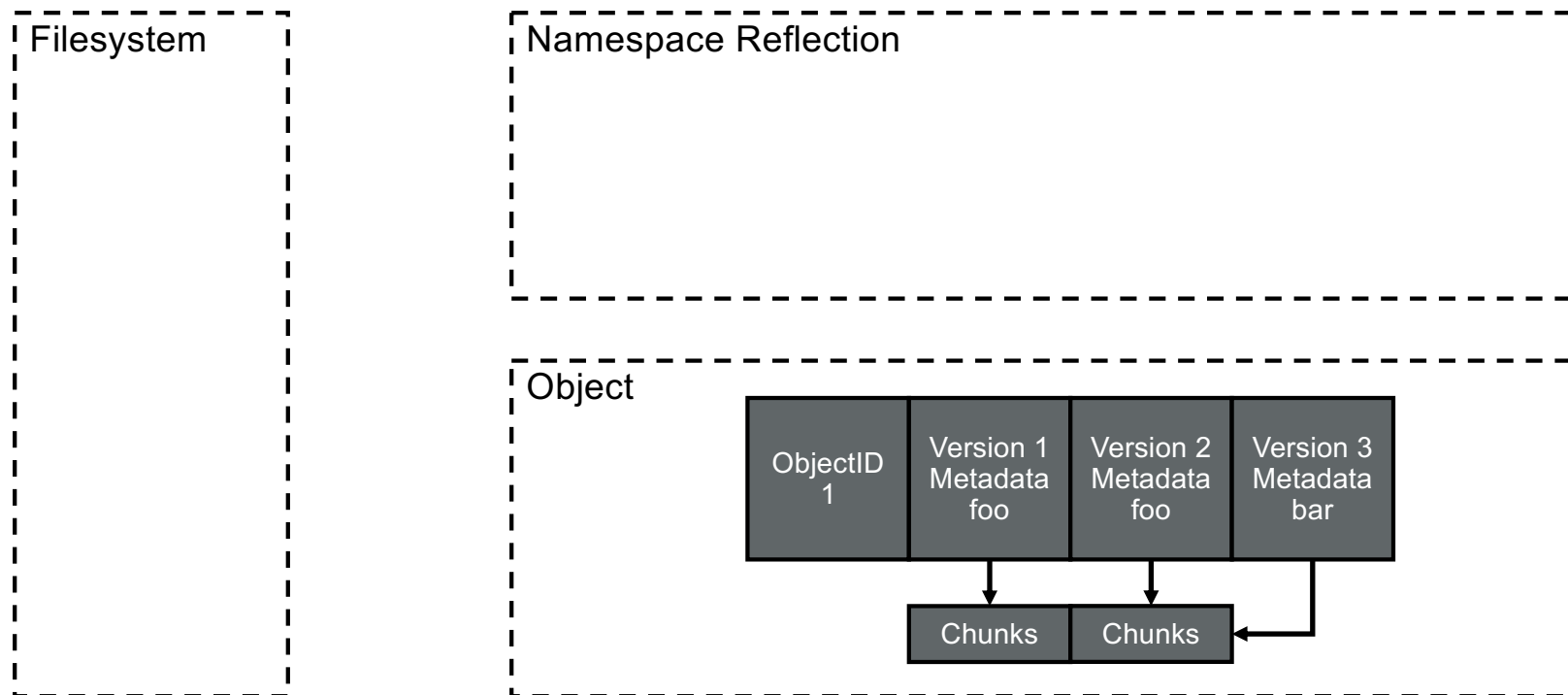
## DMF7 DB: foo is Versioned by Changing the Data



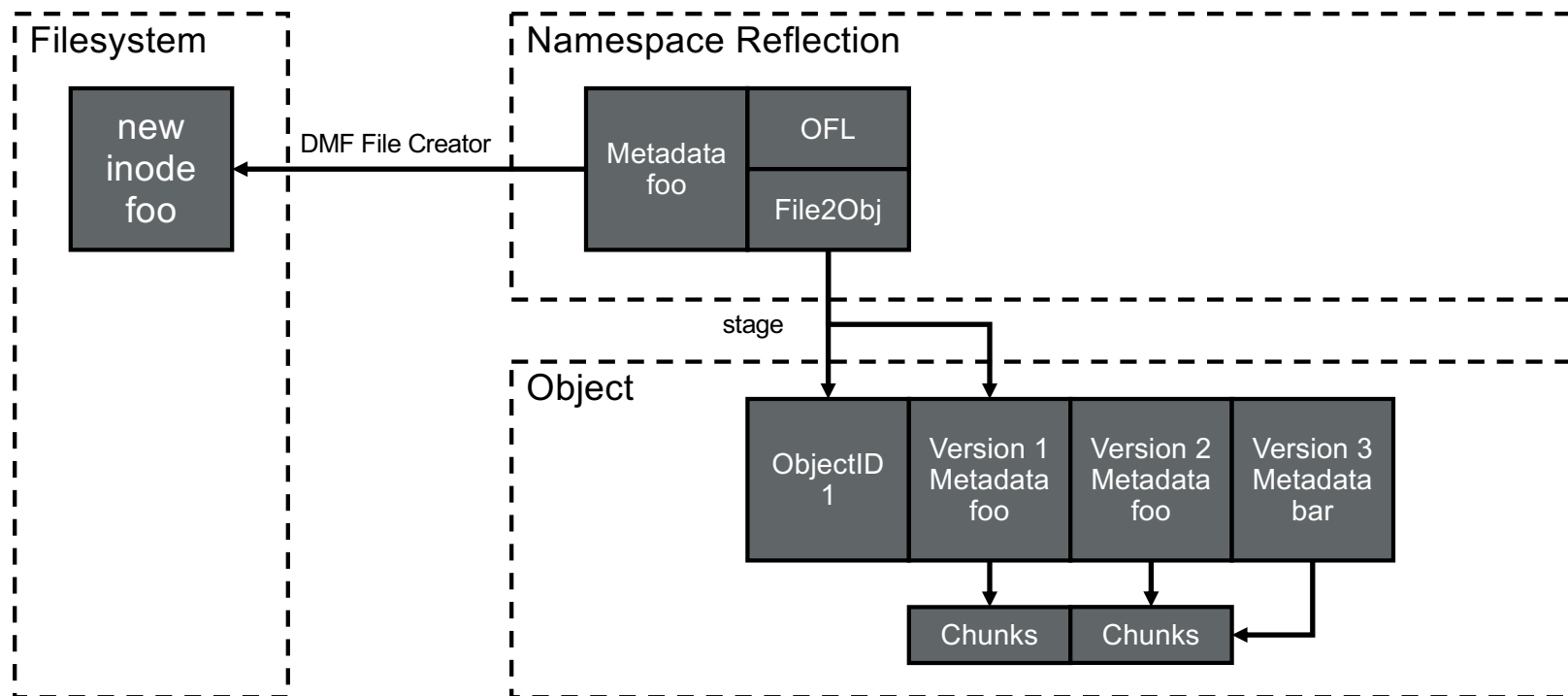
## DMF7 DB: foo is Versioned by Changing Name to bar



## DMF7 DB: Destage bar

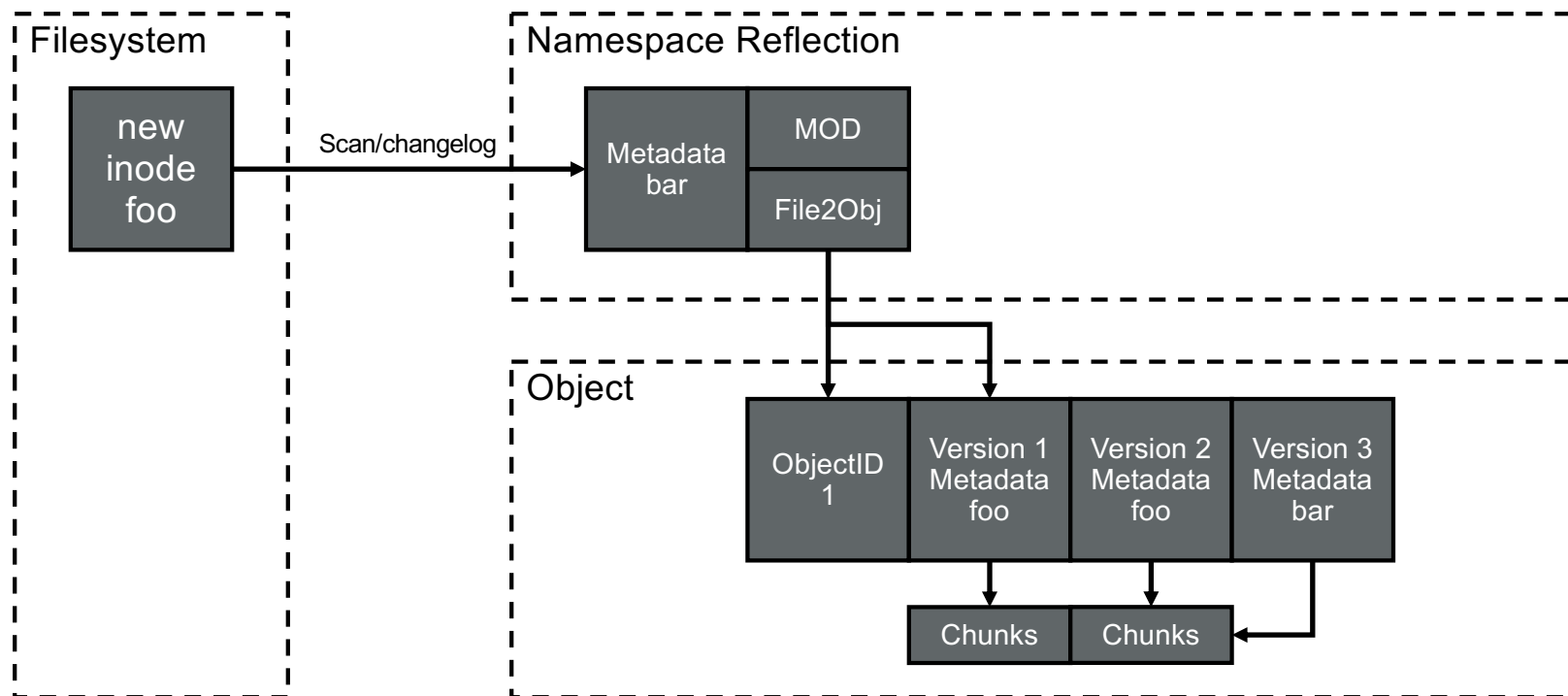


## DMF7 DB: Stage foo from Version 1

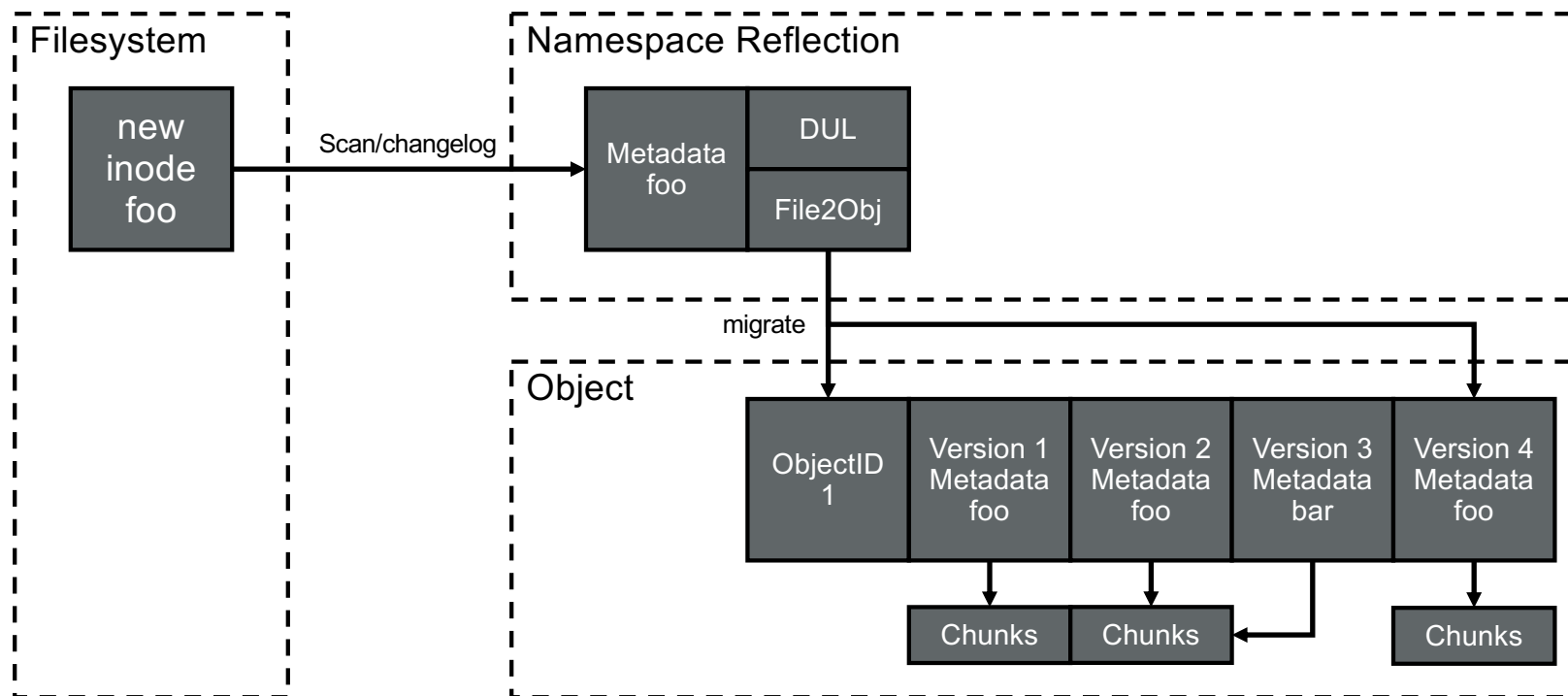




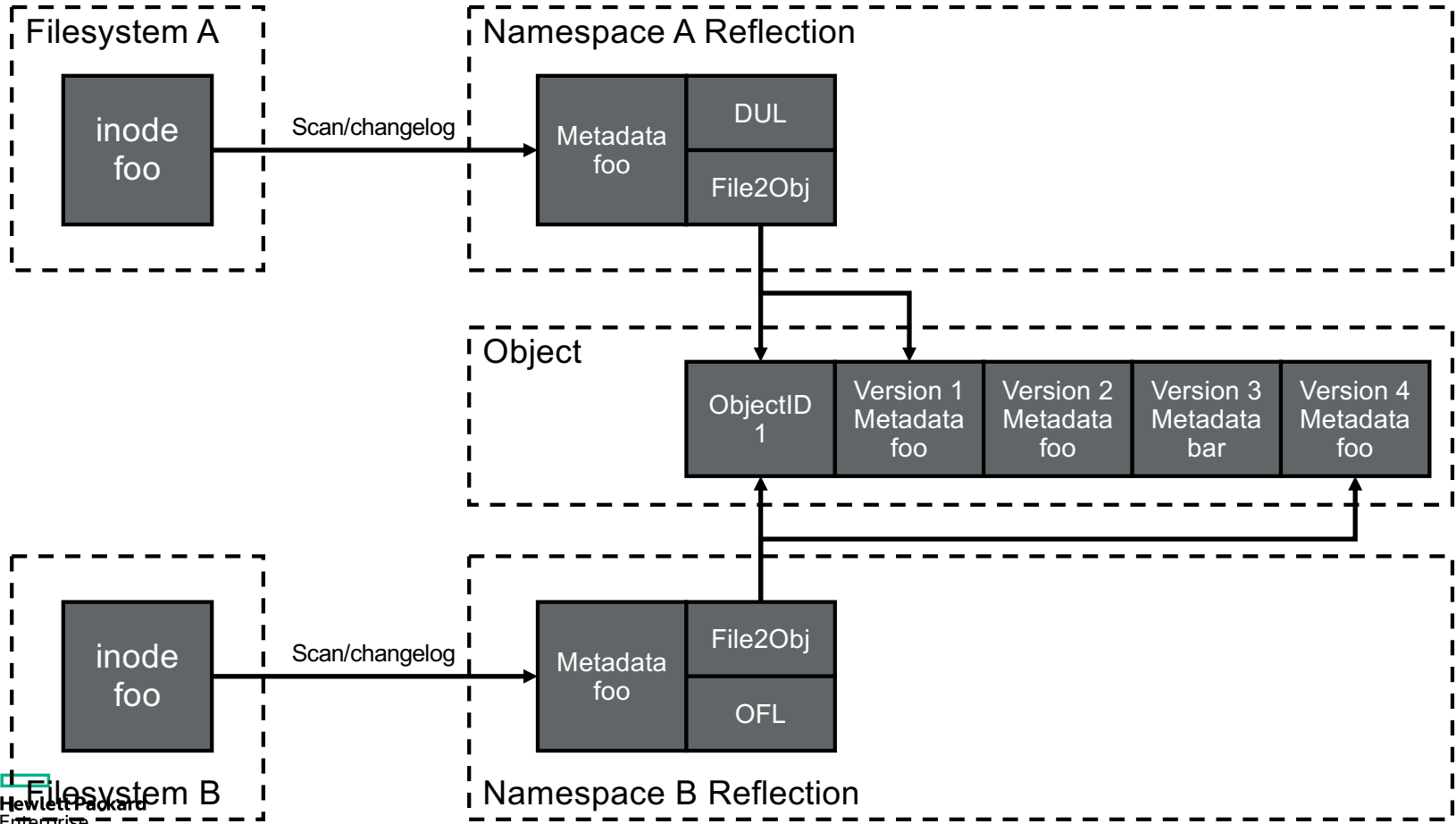
## DMF7 DB: foo is Modified Again



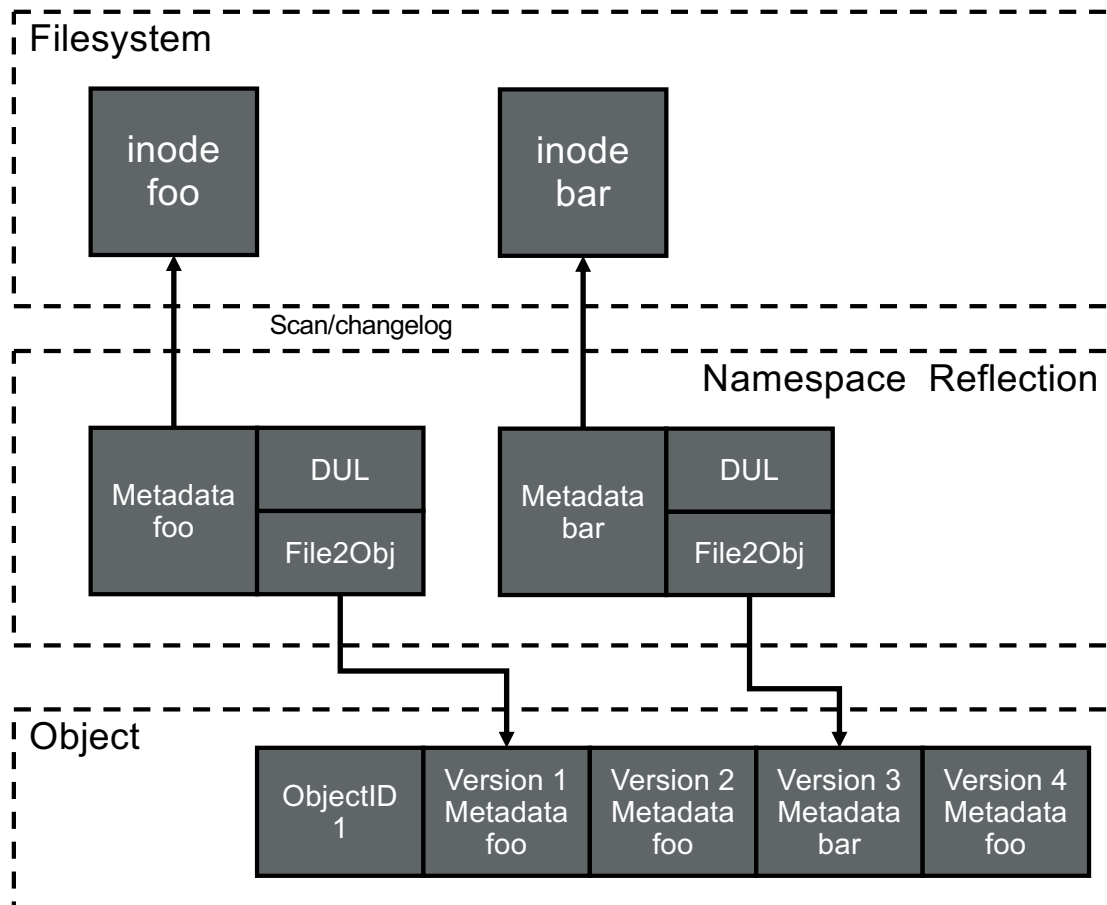
## DMF7 DB: foo is Versioned Again



# DMF7 DB: This is Possible



## DMF7 DB: This is Also Possible



---

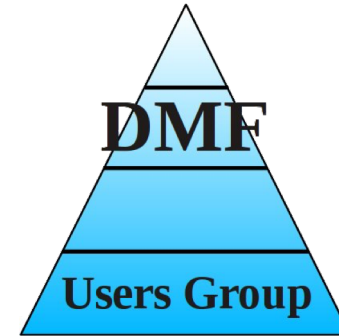
## DMF7 DB: Notes on Sizing

- The database is replicated
- The default Replication Factor for all keyspaces is 3
- Full POSIX metadata, including all extended attributes, are stored in both the reflection and object version
- Length of file and directory names effects size in DB
- Number and size of extended attributes effects size in DB
- Each Object Version carries full metadata
- Ultimate size of the database is driven by
  - Number of inodes in the managed filesystem(s)
  - Length of file and directory names
  - Size and number of extended attributes
  - Number of Object Versions





**Hewlett Packard**  
Enterprise



# Thank You