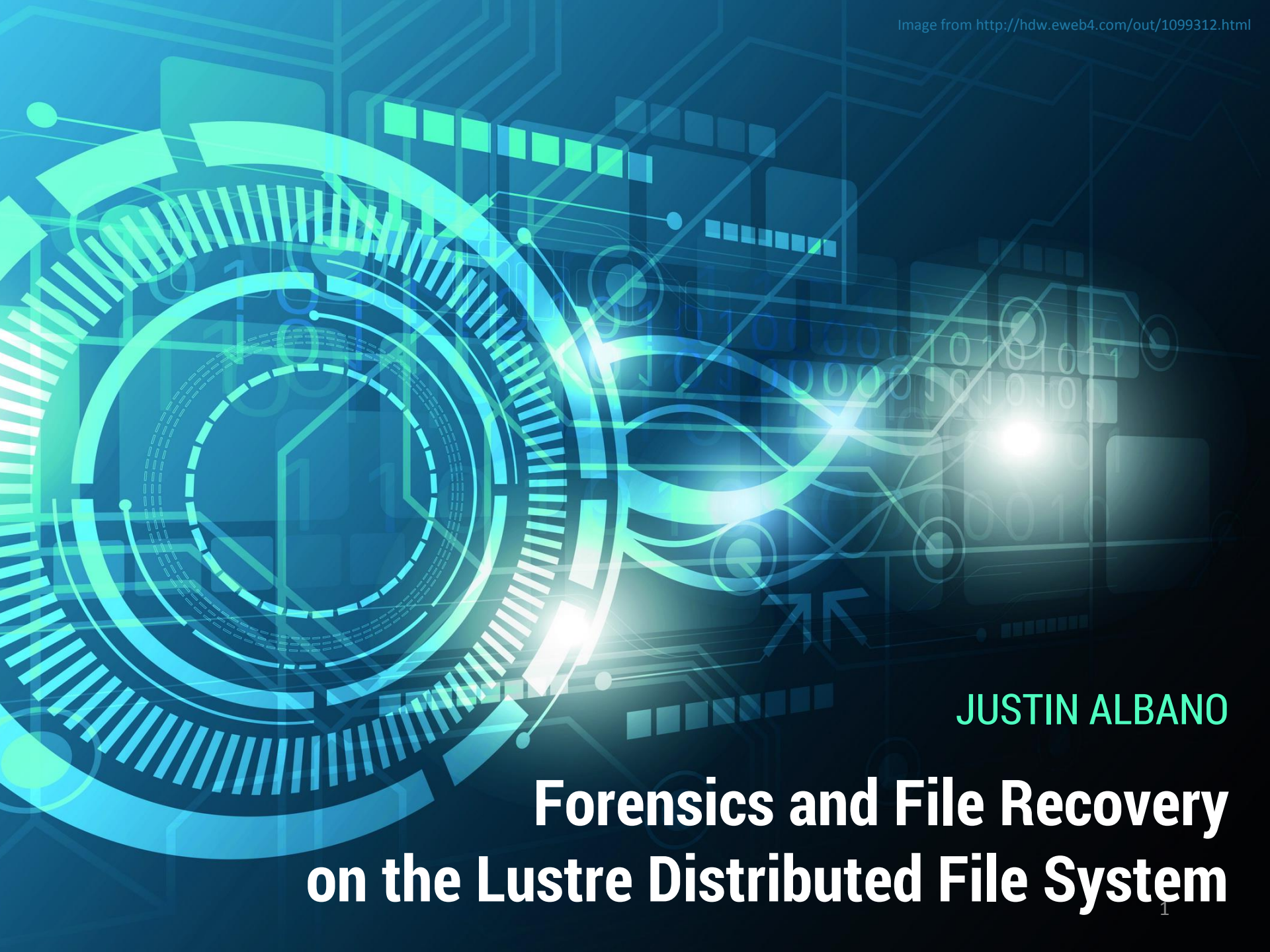


JUSTIN ALBANO

Forensics and File Recovery on the Lustre Distributed File System

Overview

Overview of Lustre

- The anatomy of a distributed file system
- The components of a Lustre file system

Background Information

- File Striping
- Linux Virtual File System

Problem & Solution

- Description of the problem being solved
- The conceptual solution and solution architecture

Purpose

Creating a systematic process for recovering a file from the Lustre distributed file system

Originally, the goal was to create a tool to automate this process, but the constrained timeline for this research limited this scope

In the end, a conceptual solution and solution architecture were devised, making the implementation of this desired tool possible with future research

Problem Statement

What is the problem that is being solved?

- While a great deal of research has been completed on distributed file systems, there is a lack of research into forensics and file recovery on these distributed systems
- This challenge is important for various customers:
  - Intelligence agencies
  - Enterprises and companies
  - Law enforcement
  - Individuals

Overview of Lustre File System

What is Lustre?

- The Lustre file system is an object-based distributed file system capable of petabytes per second of aggregate bandwidth and petabytes of file storage

Why is Lustre important?

- As cloud computing and distributed systems grow in popularity, a file system is needed that can support the massive storage and network bandwidth of these systems

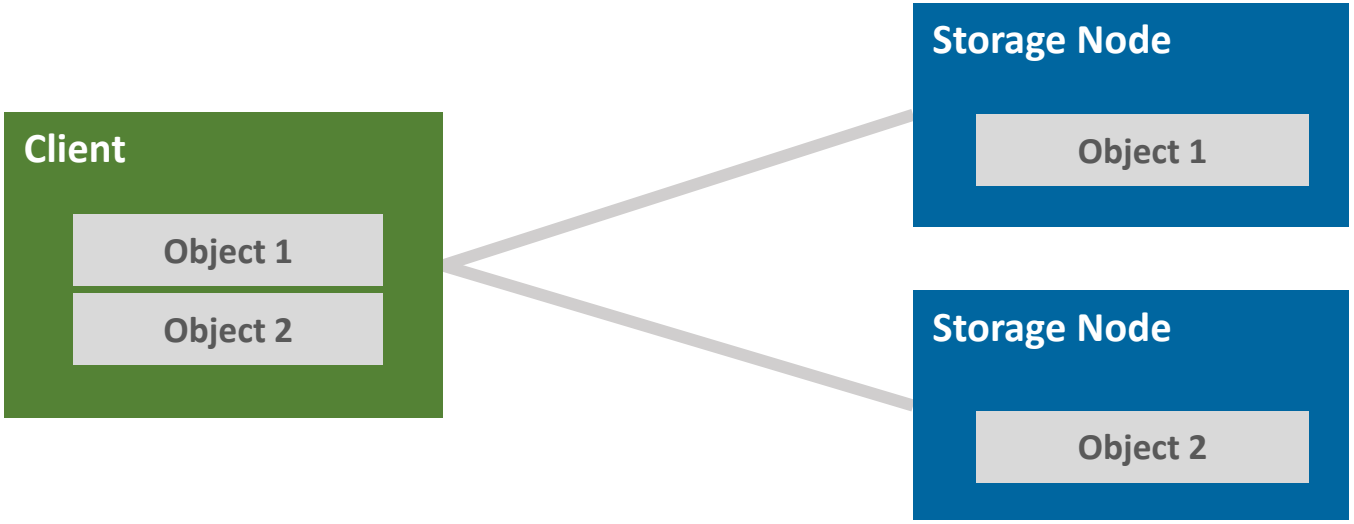
How long has Lustre been around?

- Originally created in 1999 by Peter Braam at Carnegie Mellon University
- Purchased by Sun Microsystems in 2007 and later by Oracle in 2010
- Now supported by OpenSFS, Intel, and Seagate

Overview of Lustre File System

What is an object-based distributed file system?

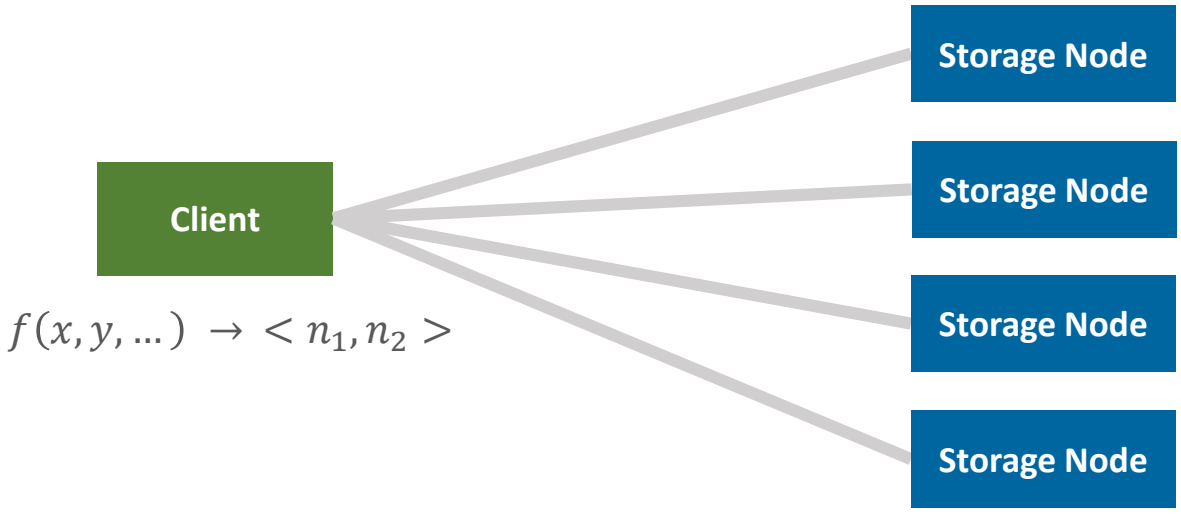
- Files are divided into “objects” and placed on various nodes in a network
- Instead of accessing a file serially, files can be accessed in parallel



Overview of Lustre File System

How does a client know where the objects are stored?

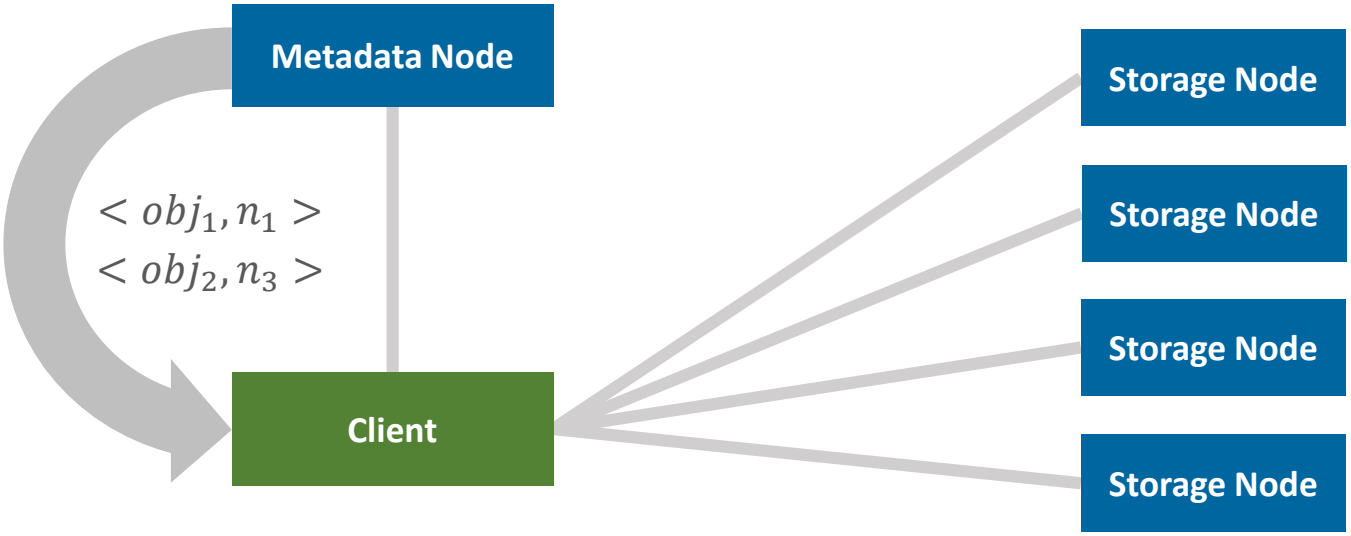
- Using a deterministic algorithm, such as CRUSH, on each of the clients to determine where the objects associated with a file can be found



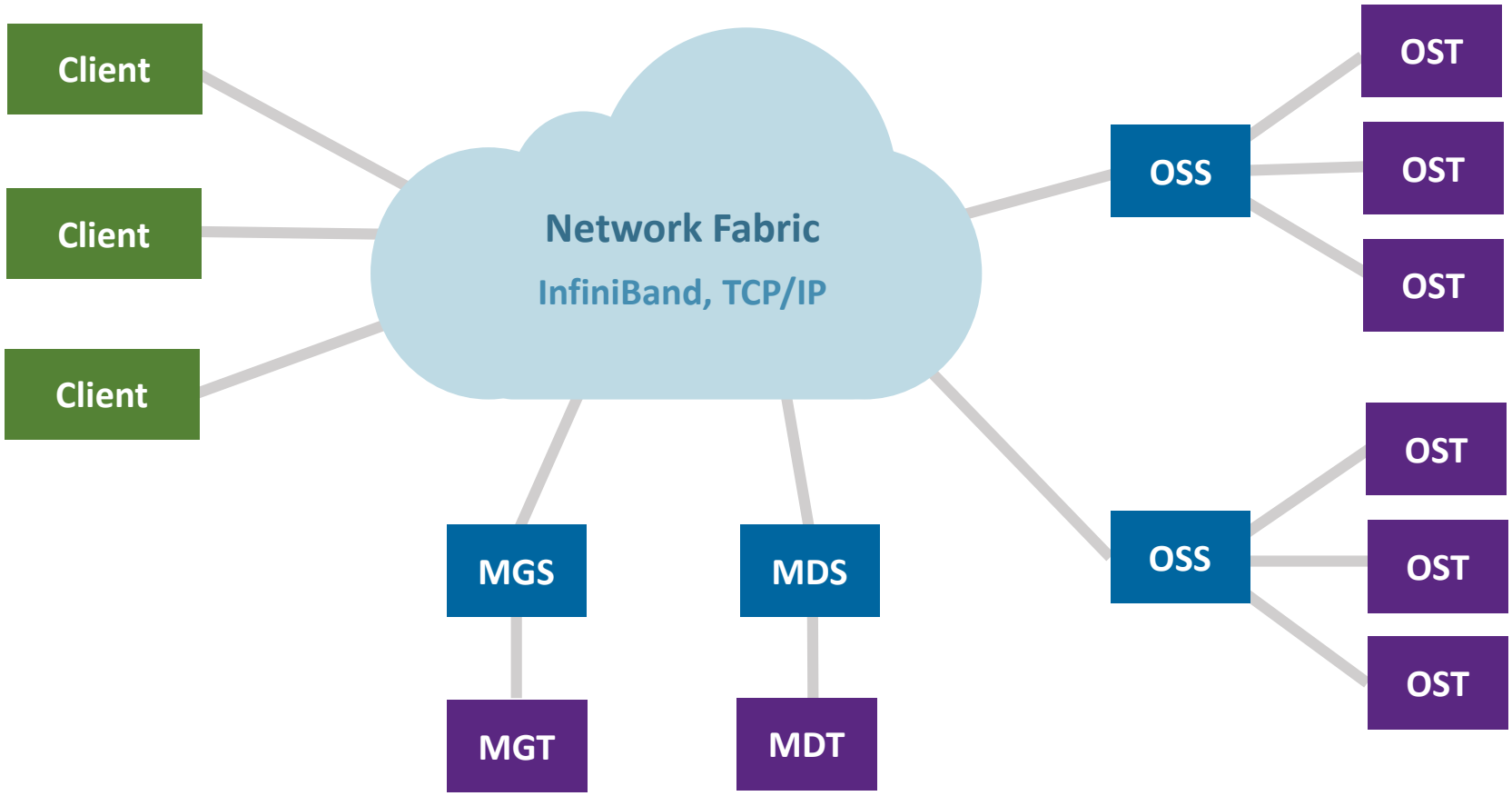
Overview of Lustre File System

How does a client know where the objects are stored?

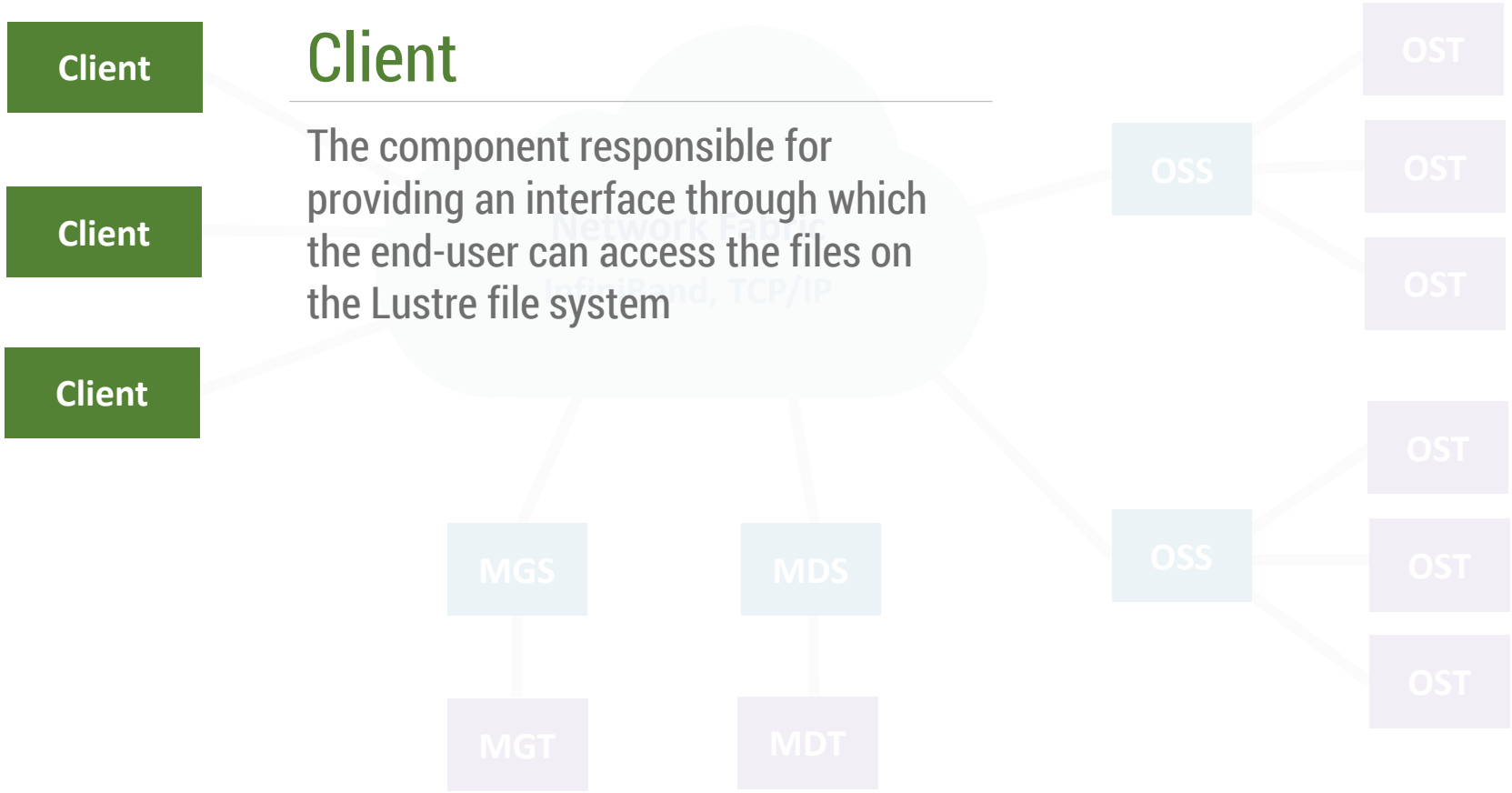
- Using a metadata node that provides the client with a mapping of objects to the storage nodes on which the objects reside



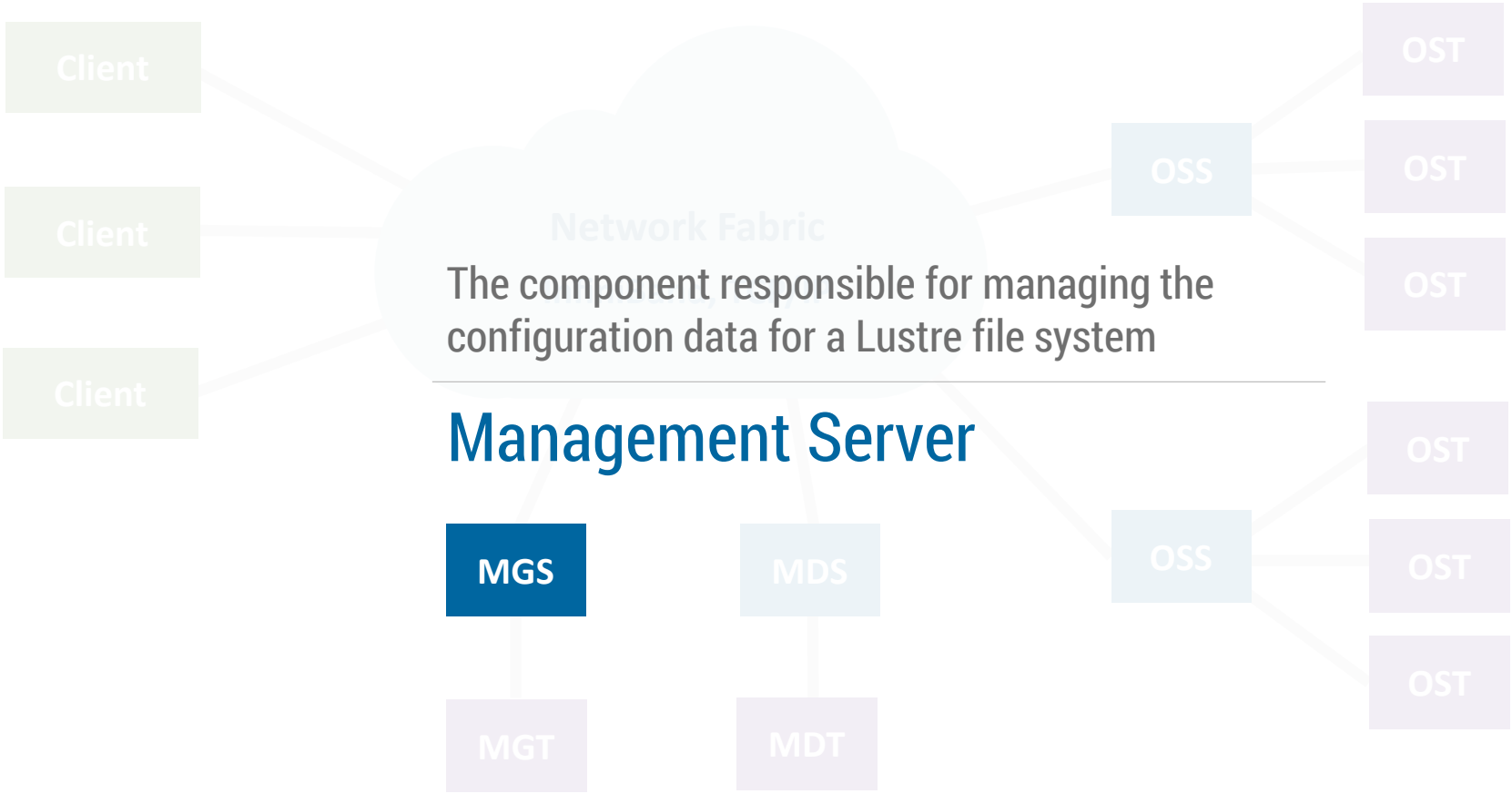
Overview of Lustre File System



Overview of Lustre File System



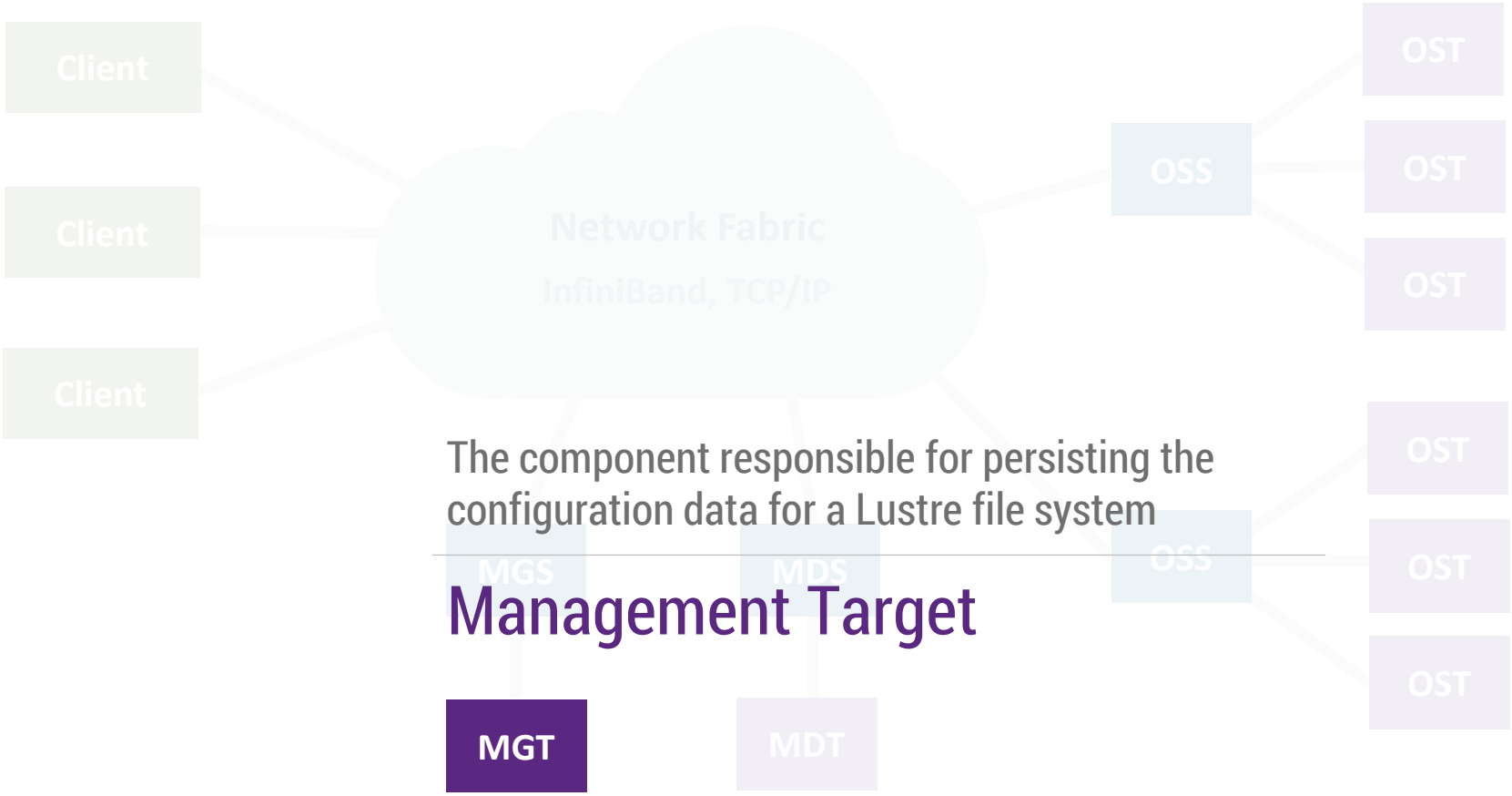
Overview of Lustre File System



The component responsible for managing the configuration data for a Lustre file system

Management Server

Overview of Lustre File System

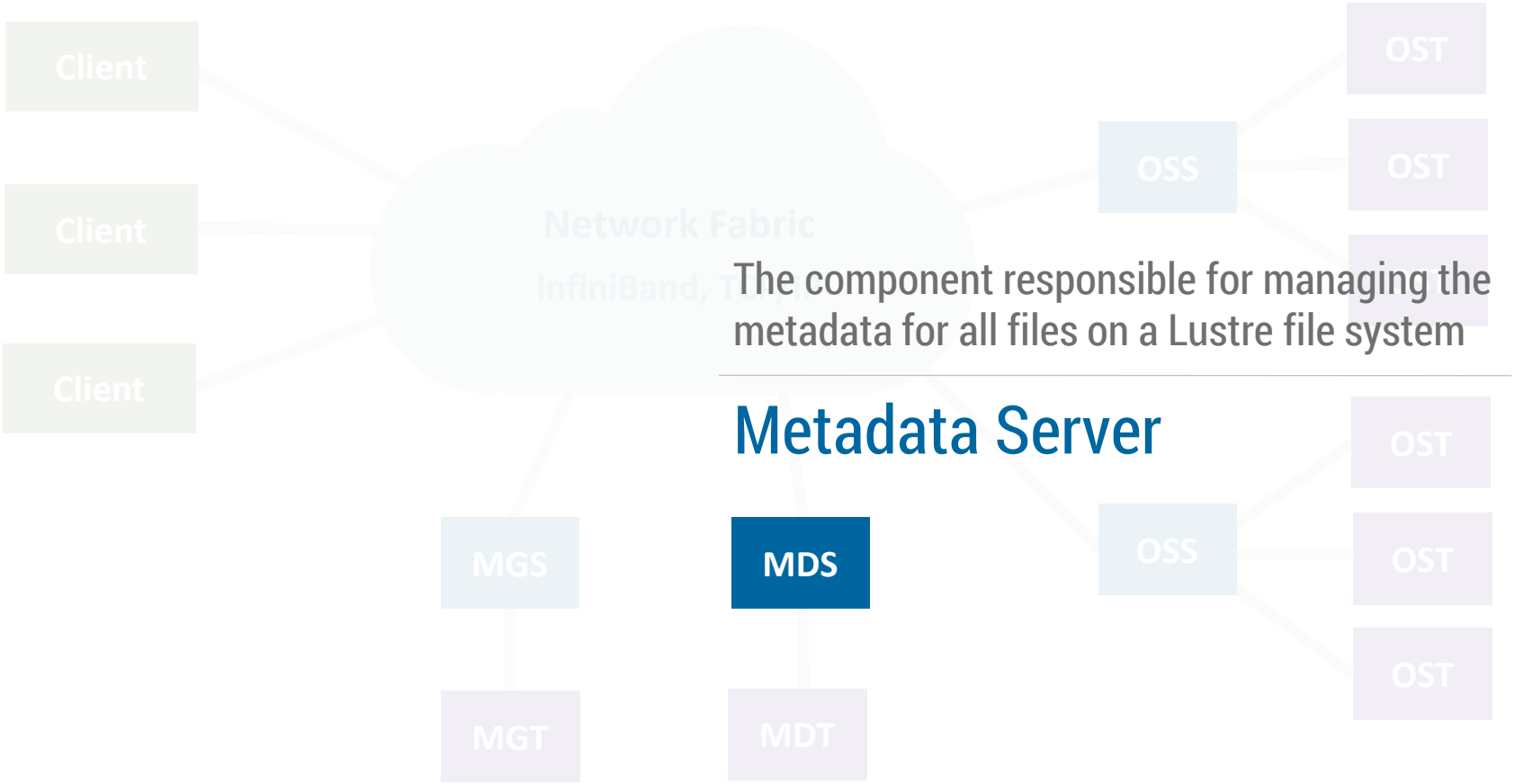


The component responsible for persisting the configuration data for a Lustre file system

Management Target

MGT

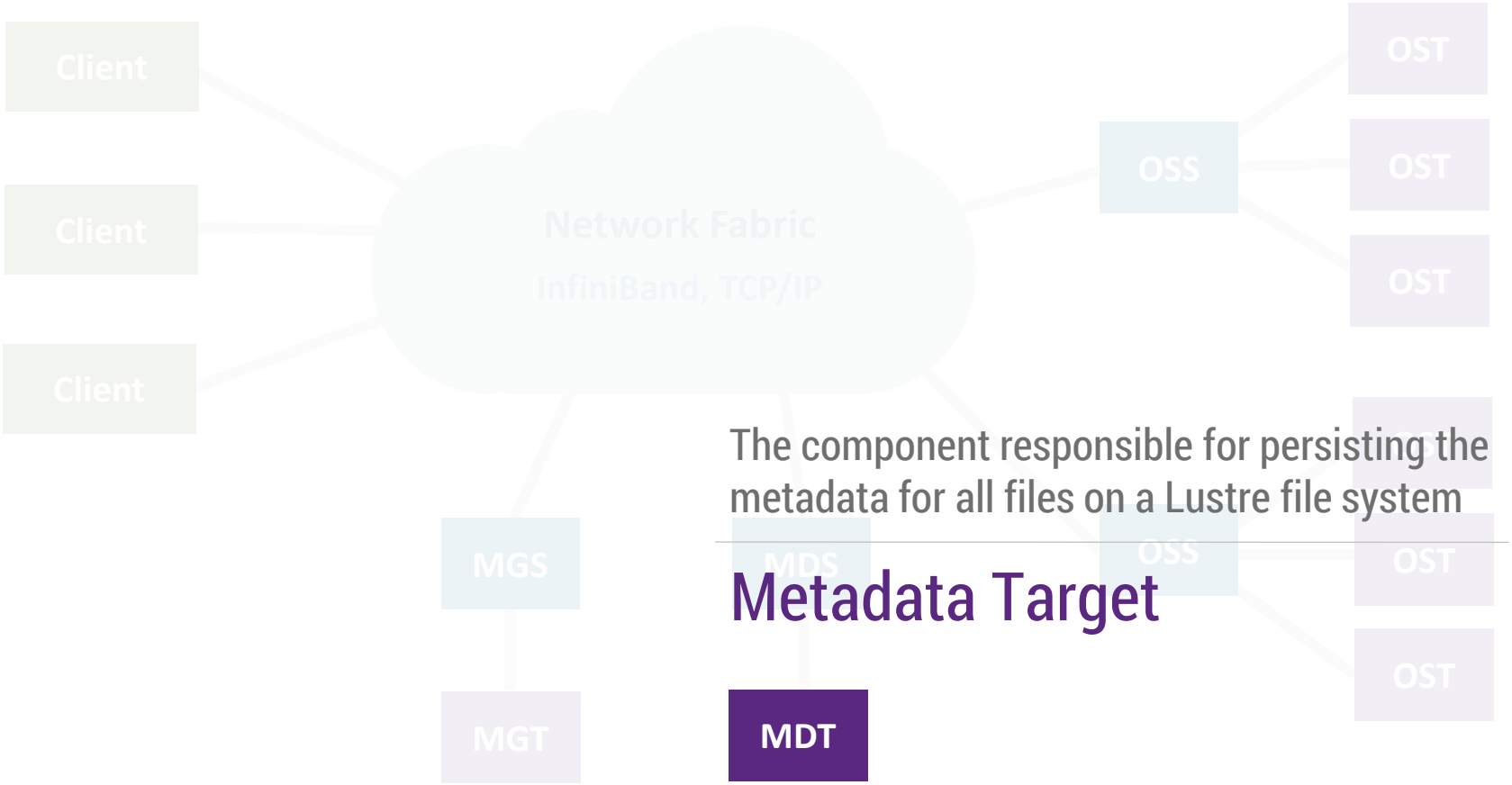
Overview of Lustre File System



The component responsible for managing the metadata for all files on a Lustre file system

Metadata Server

Overview of Lustre File System

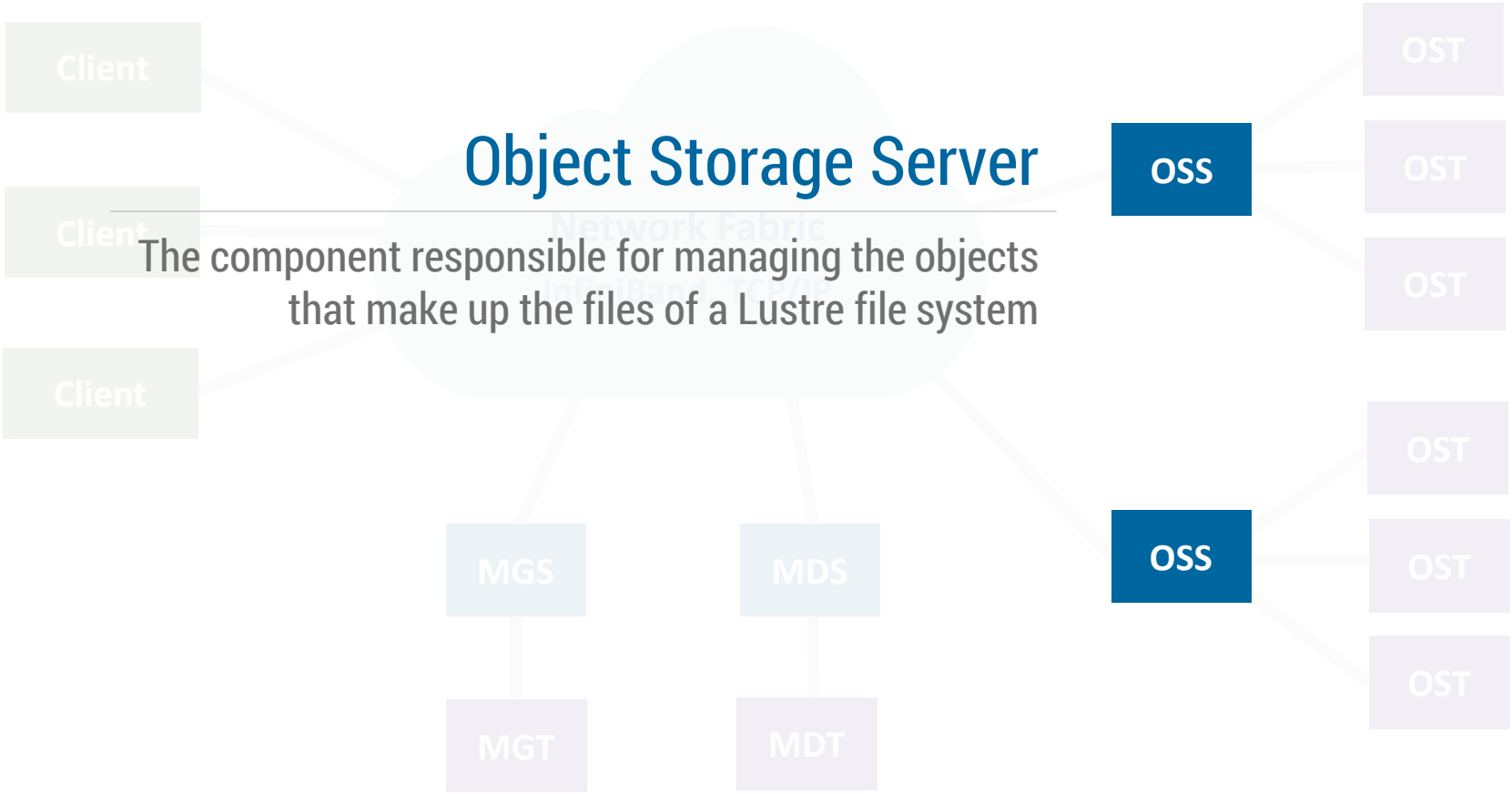


The component responsible for persisting the metadata for all files on a Lustre file system

Metadata Target

MDT

Overview of Lustre File System



Object Storage Server

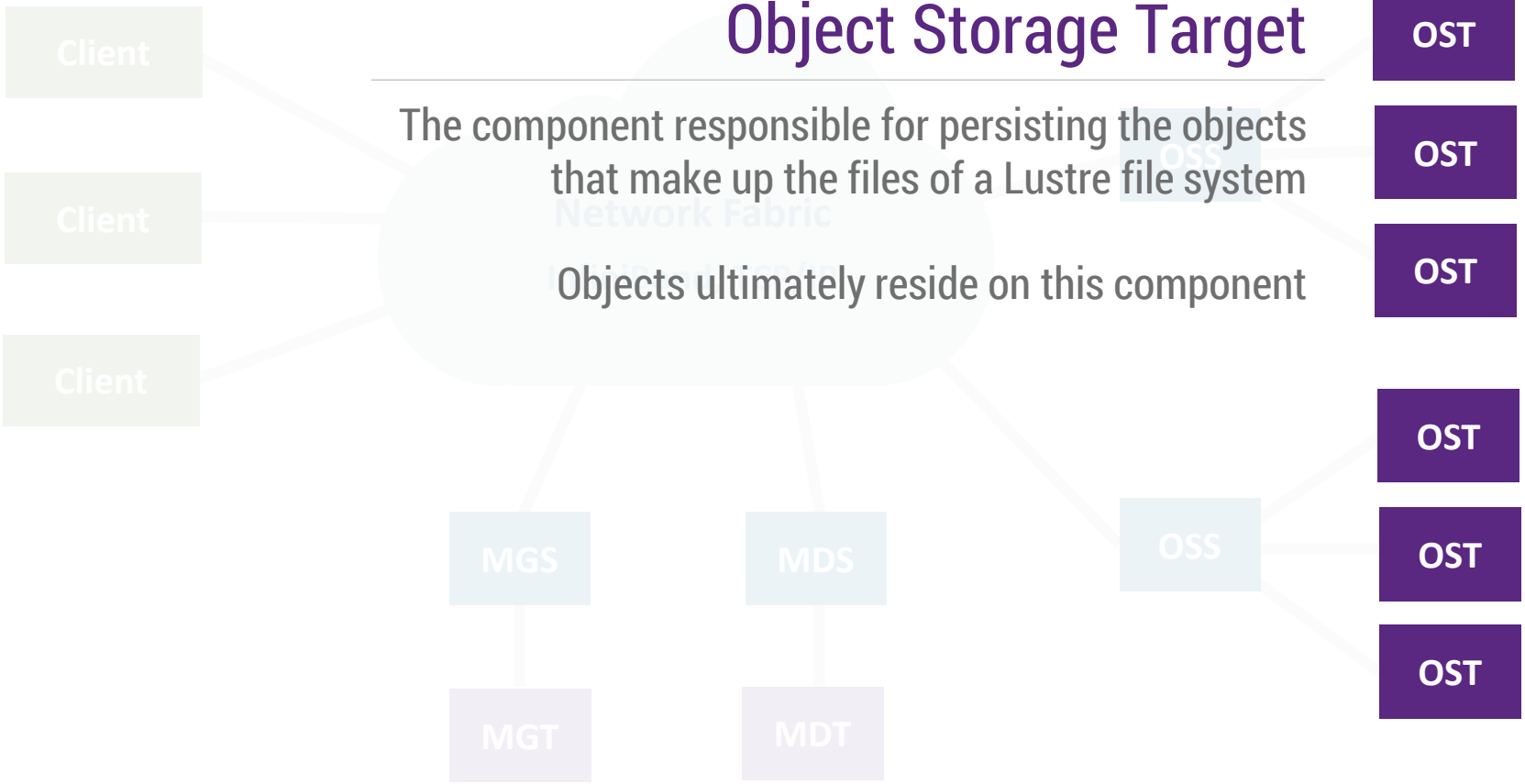
The component responsible for managing the objects that make up the files of a Lustre file system

Overview of Lustre File System

Object Storage Target

The component responsible for persisting the objects that make up the files of a Lustre file system

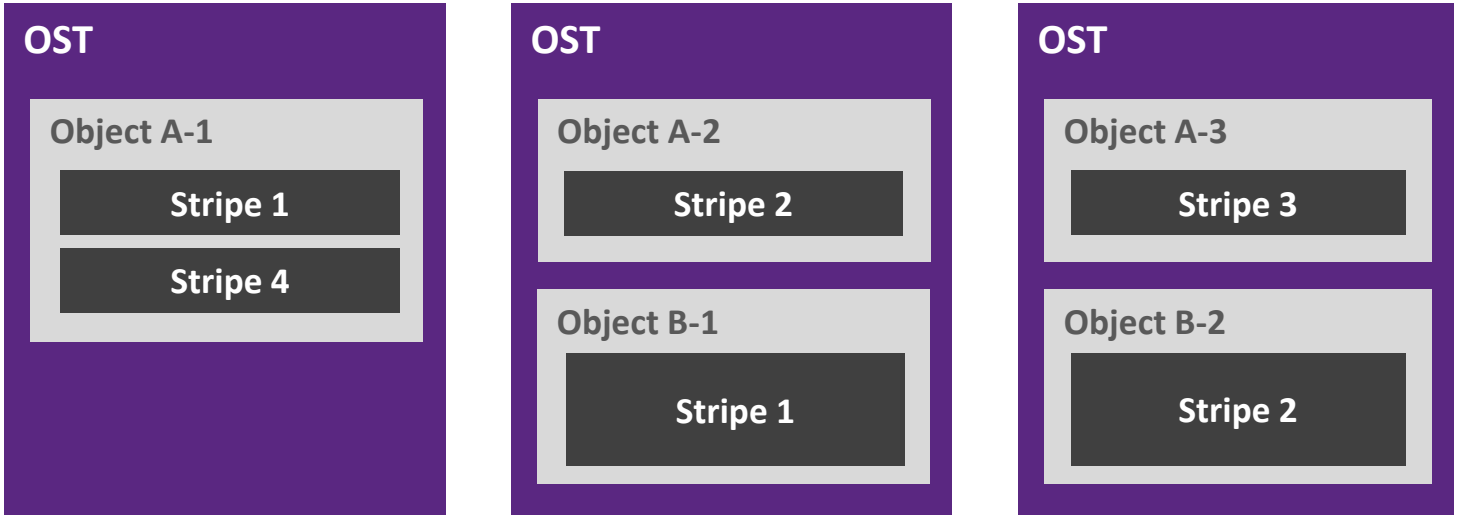
Objects ultimately reside on this component



Background Information

How are objects distributed on each OST?

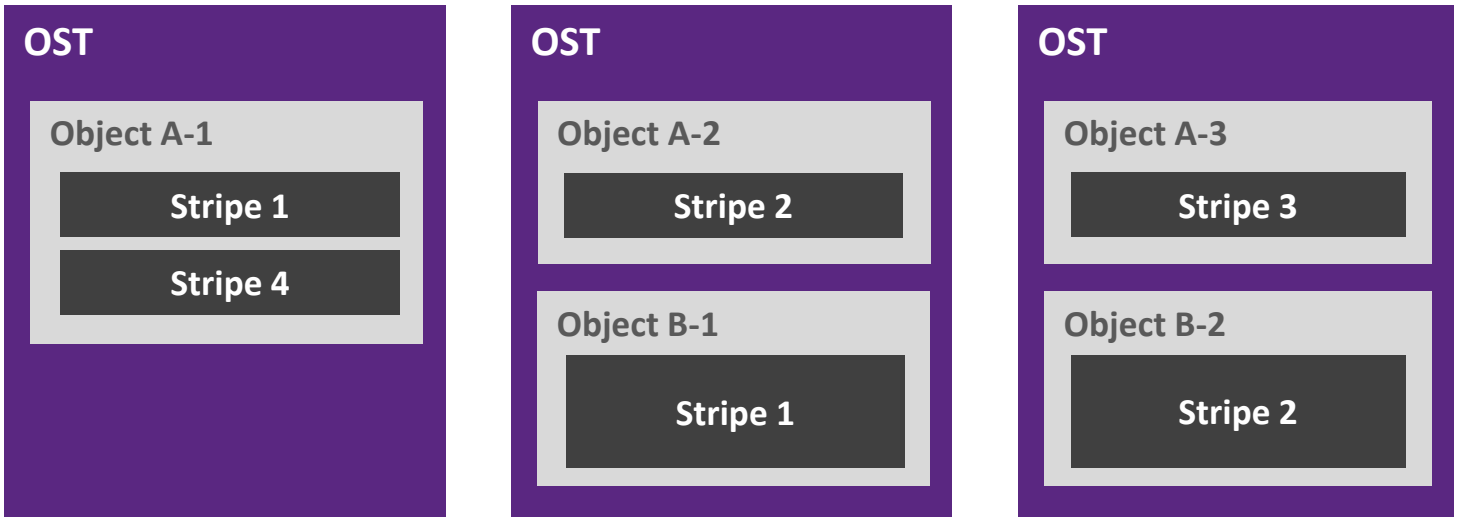
- OSTs are selected to store each object associated with a file
- Each object contains stripes that are written to the object in a round-robin fashion, similar to RAID 0 on a local disk array



Background Information

Object Striping Parameterization

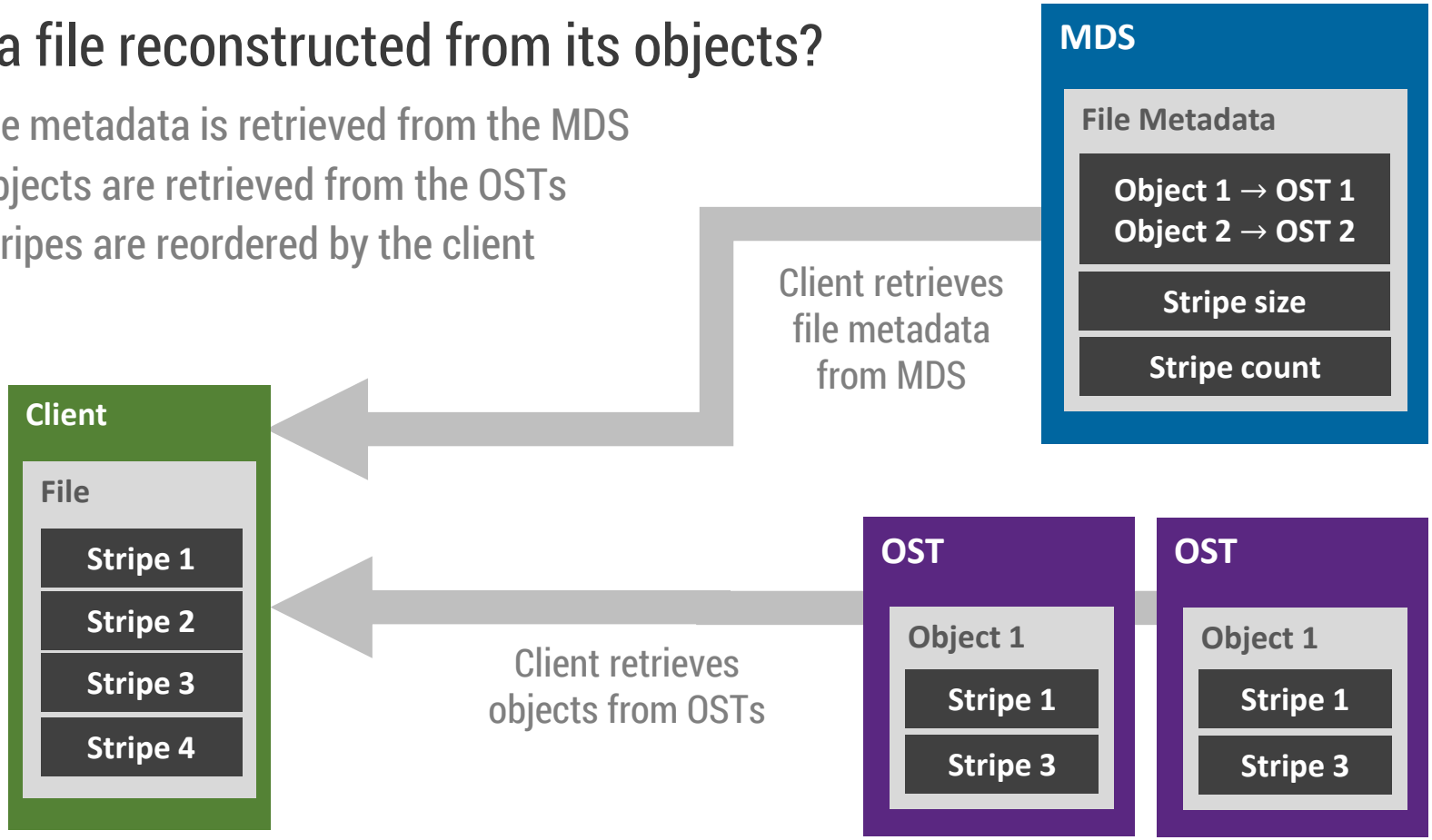
- **Stripe size**: the number of bytes containing in a single stripe
- **Stripe count**: the number of *objects* over which the file is striped



Background Information

How is a file reconstructed from its objects?

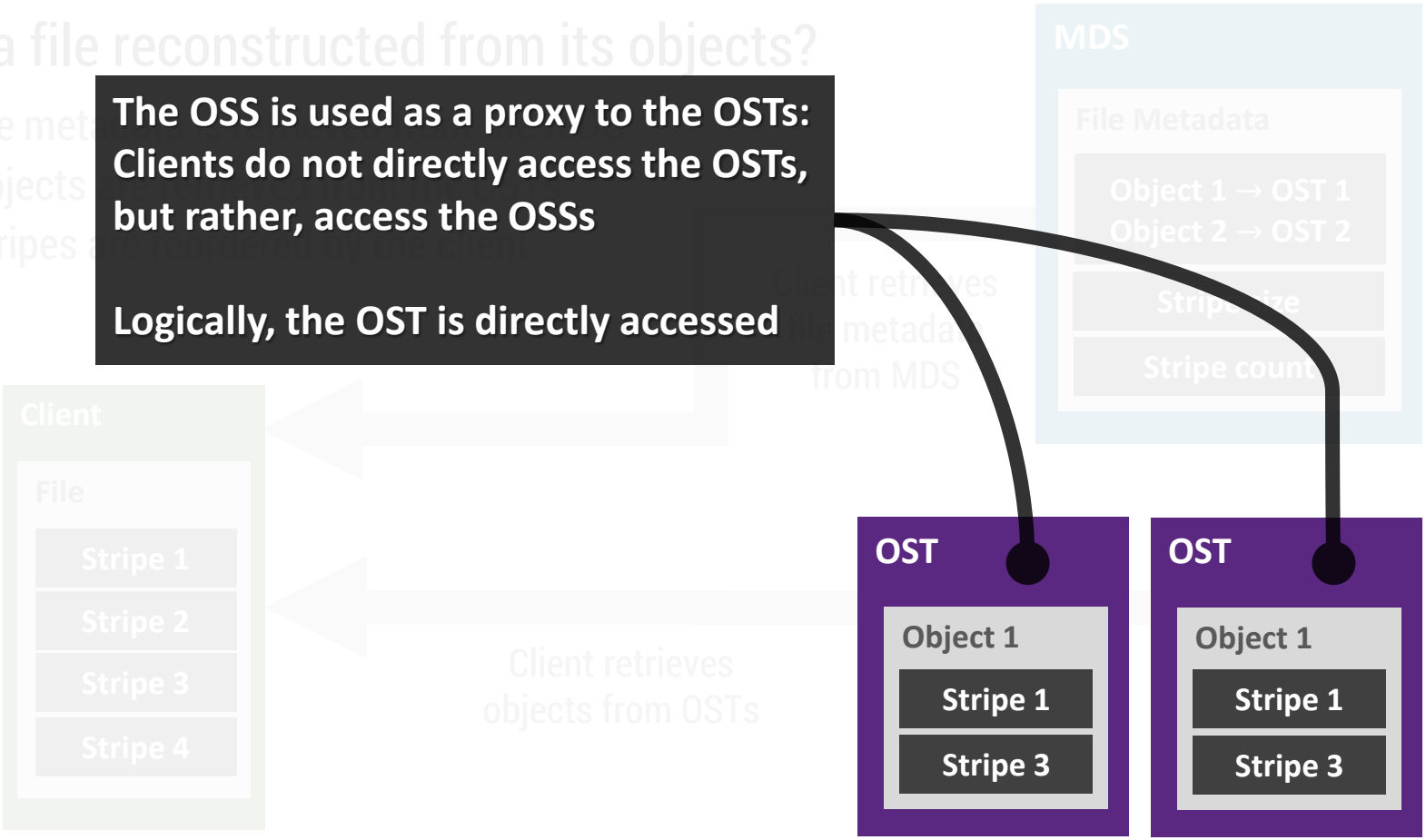
1. The file metadata is retrieved from the MDS
2. The objects are retrieved from the OSTs
3. The stripes are reordered by the client



Background Information

The OSS is used as a proxy to the OSTs: Clients do not directly access the OSTs, but rather, access the OSSs

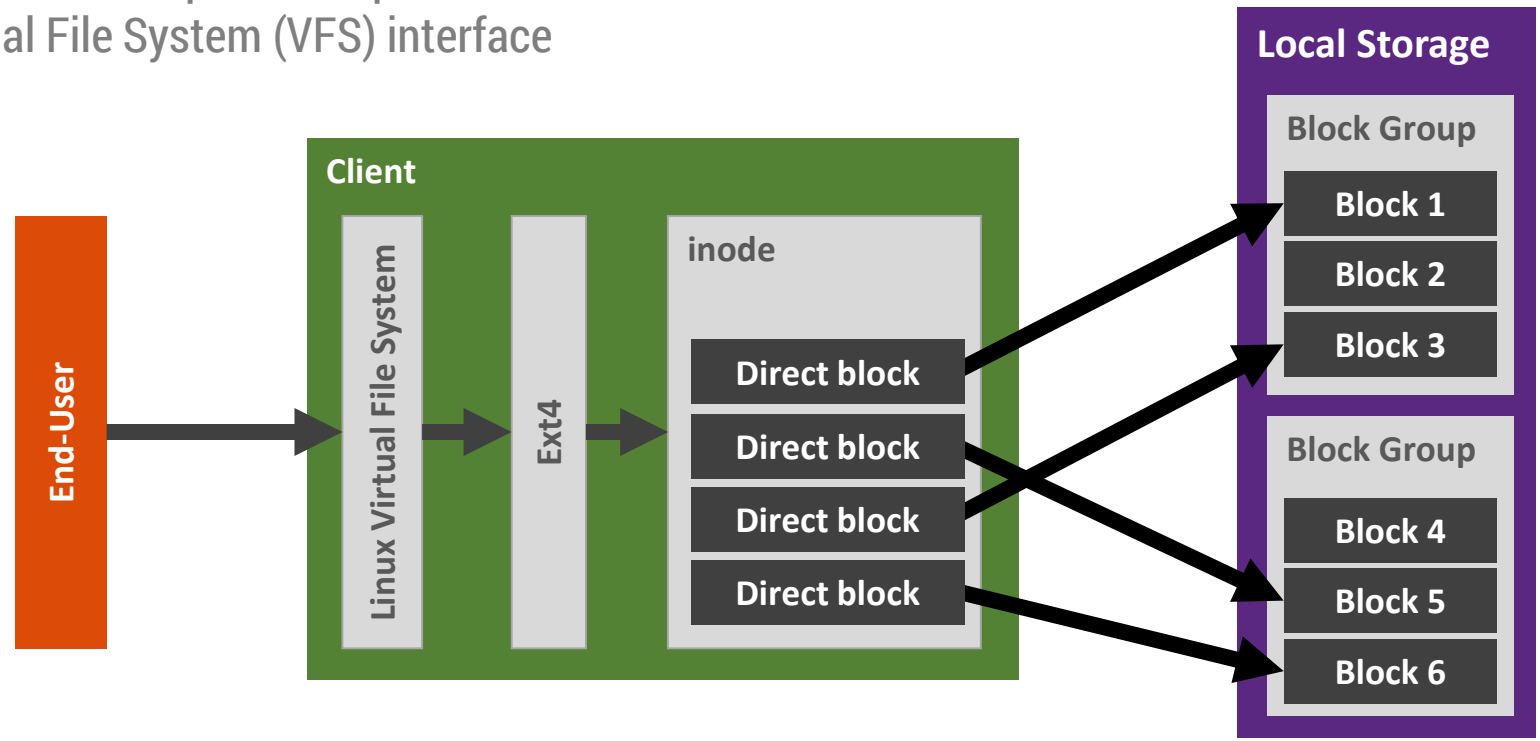
Logically, the OST is directly accessed



Background Information

How does the mounted file system work?

- The client component implements the Linux Virtual File System (VFS) interface



Background Information

What is the Linux VFS?

- An abstraction of the file system that allows clients to access a mounted file system without knowing the implementation details of the file system

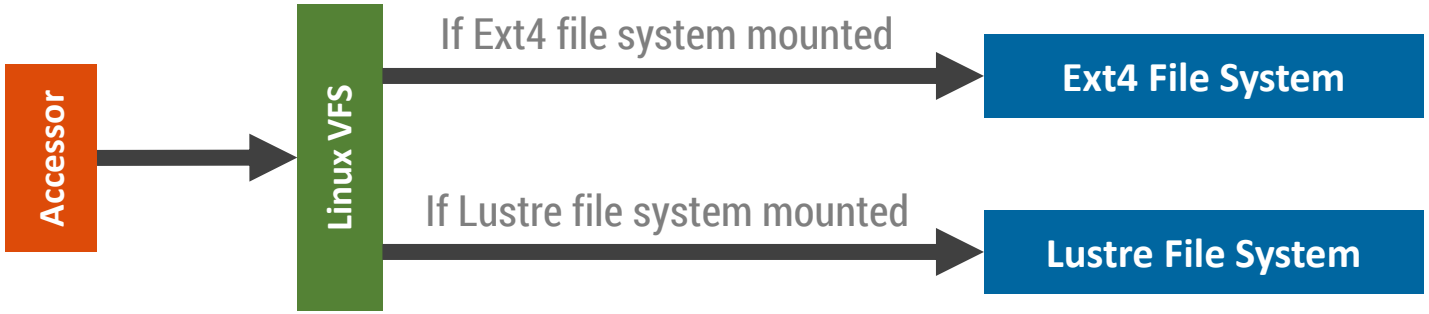
```
// Obtain reference to mounted file system and read a file
FileSystem fs = kernel.getMountedFileSystem("/home/joe/");
fs.read("/documents/foo.txt");
```

- The manner in which the file is accessed depends on the object returned from the getter function: For example, `Ext4FileSystem` or `LustreFileSystem`

Background Information

What is the Linux VFS?

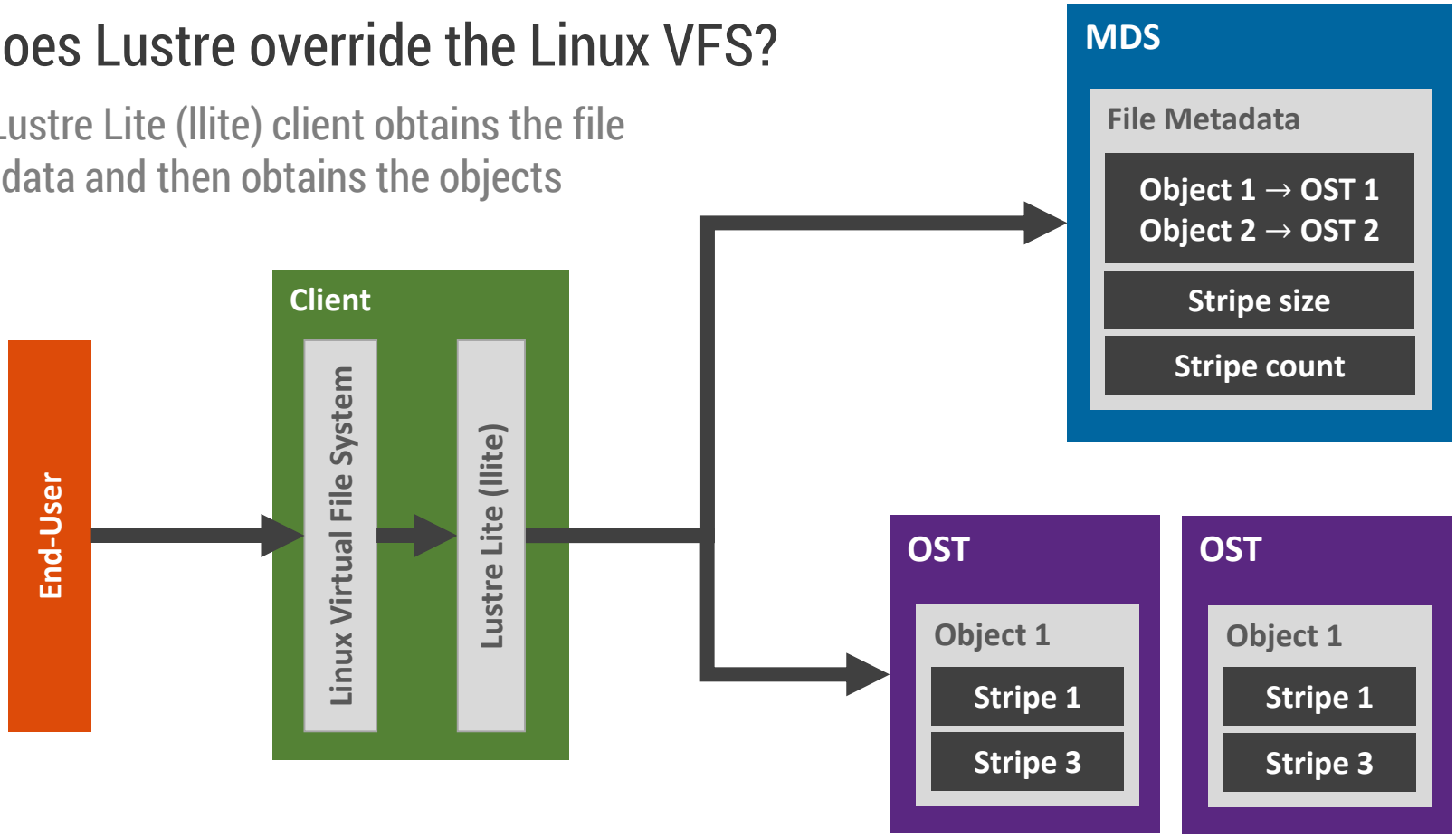
```
// Obtain reference to mounted file system and read a file
FileSystem fs = kernel.getMountedFileSystem("/home/joe/");
fs.read("/documents/foo.txt");
```



Background Information

How does Lustre override the Linux VFS?

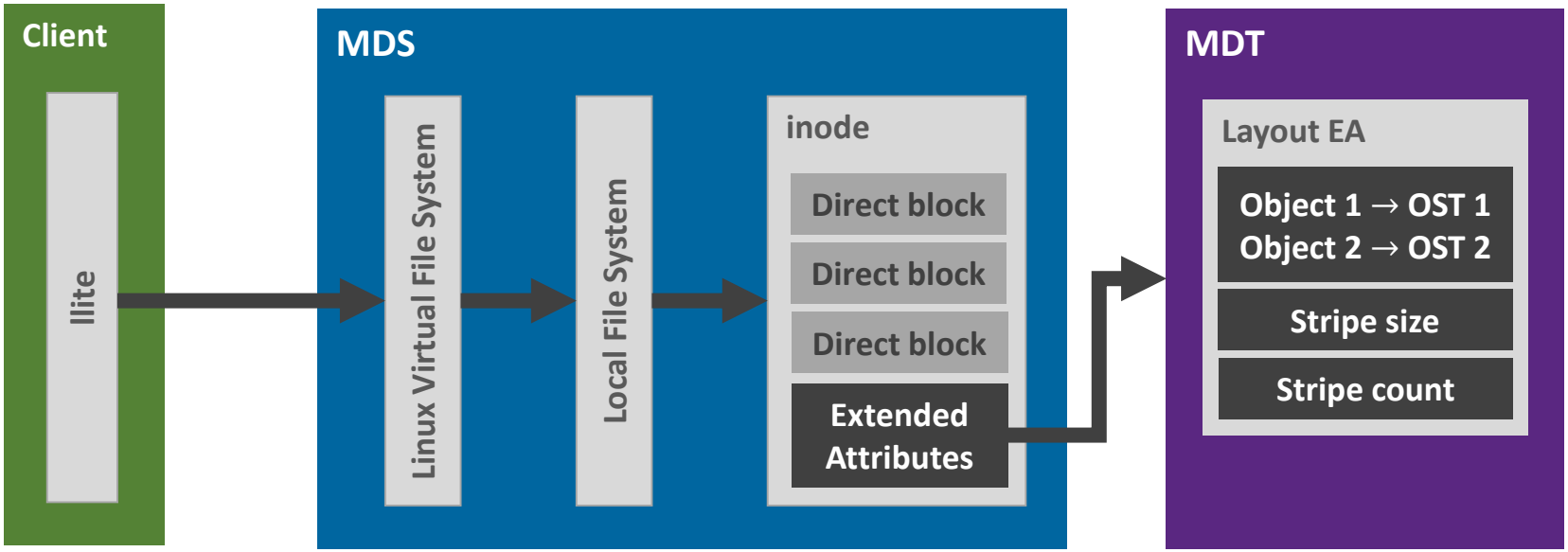
- The Lustre Lite (llite) client obtains the file metadata and then obtains the objects



Background Information

How does metadata reside on the MDT?

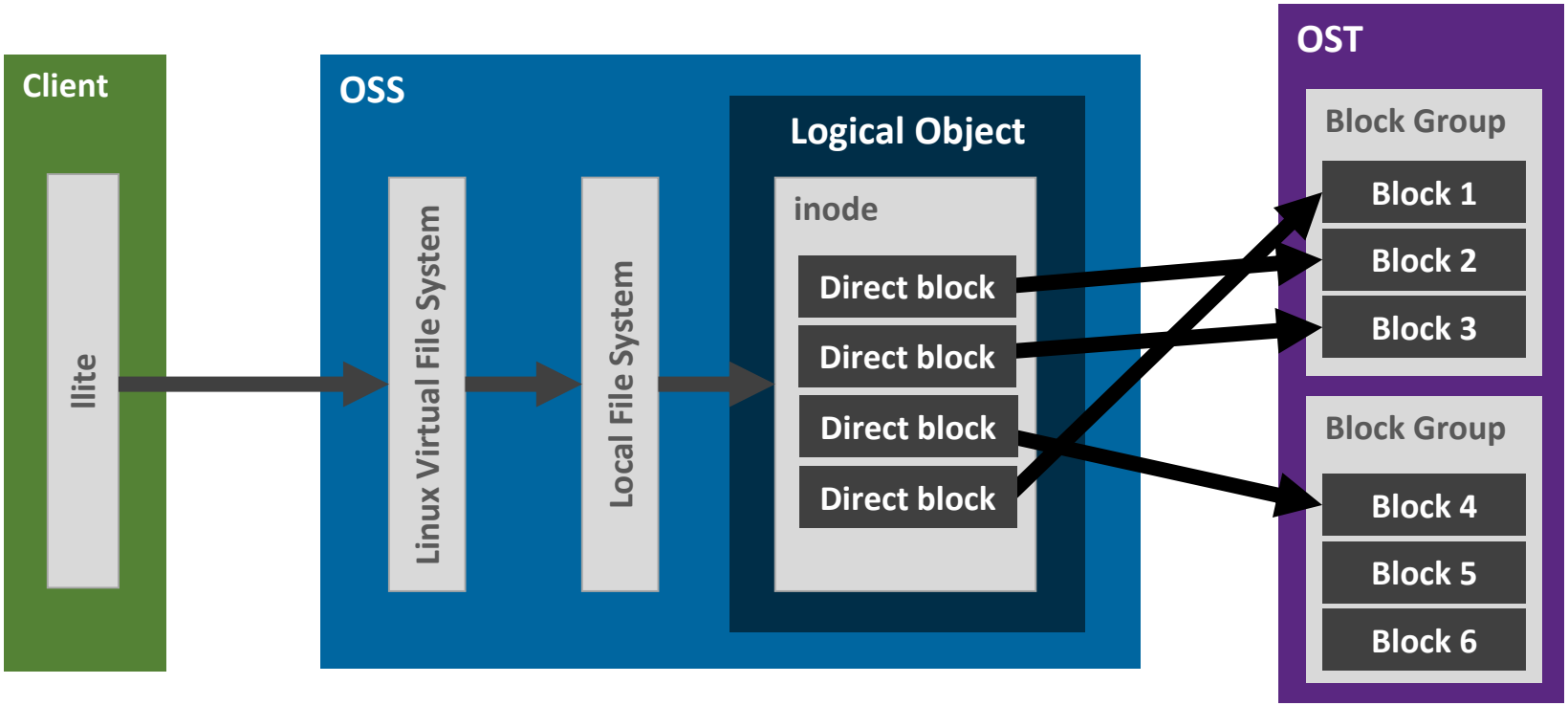
- The entire Lustre file system is represented by bare inodes, where the metadata for the file are stored in Layout Extended Attributes (EAs)



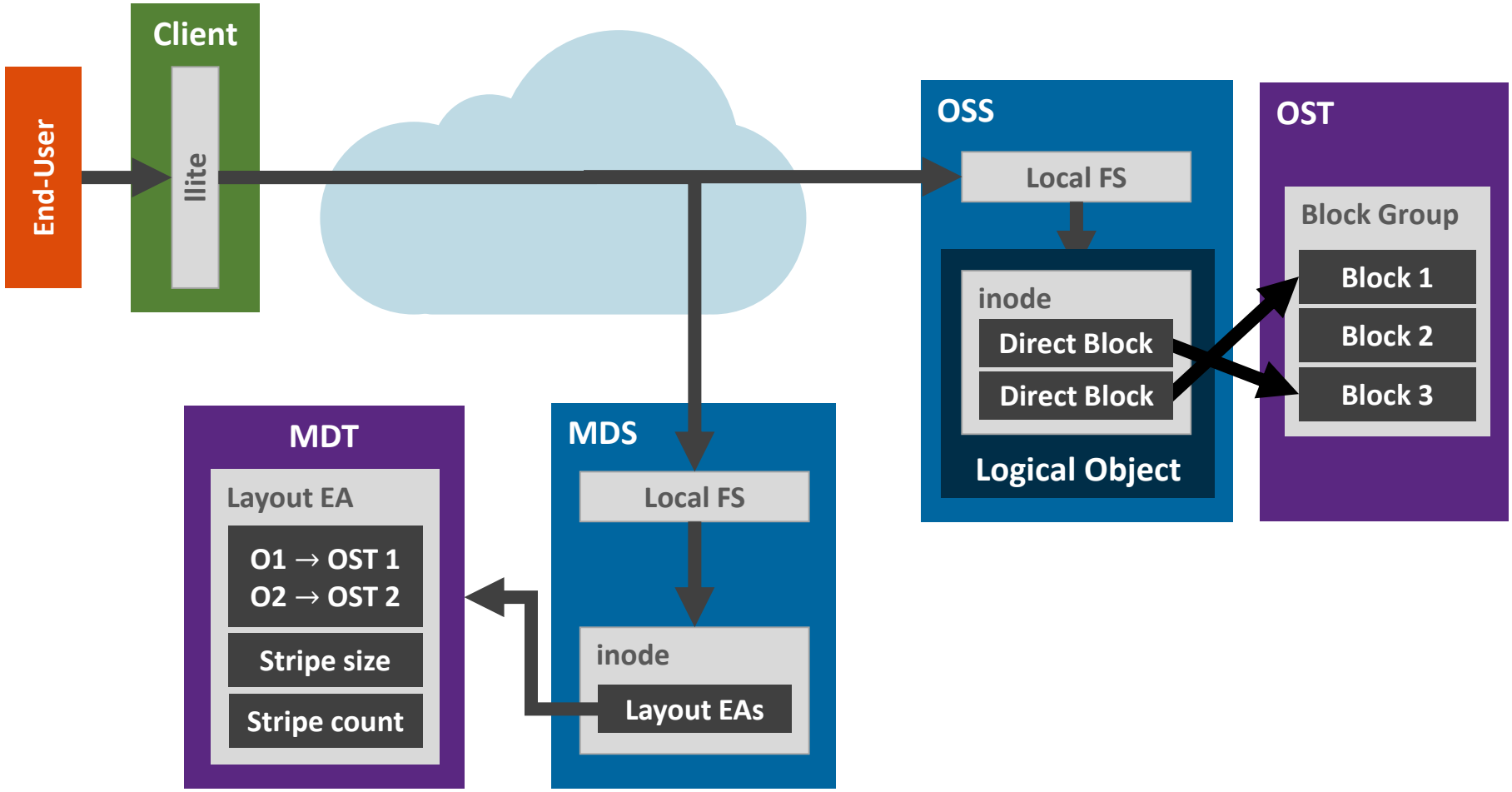
Background Information

How do objects reside on OSTs?

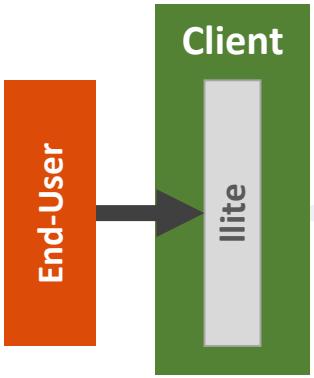
- Objects are stored as files on the local file system (such as ldiskfs or ZFS) of the OST



Background Information

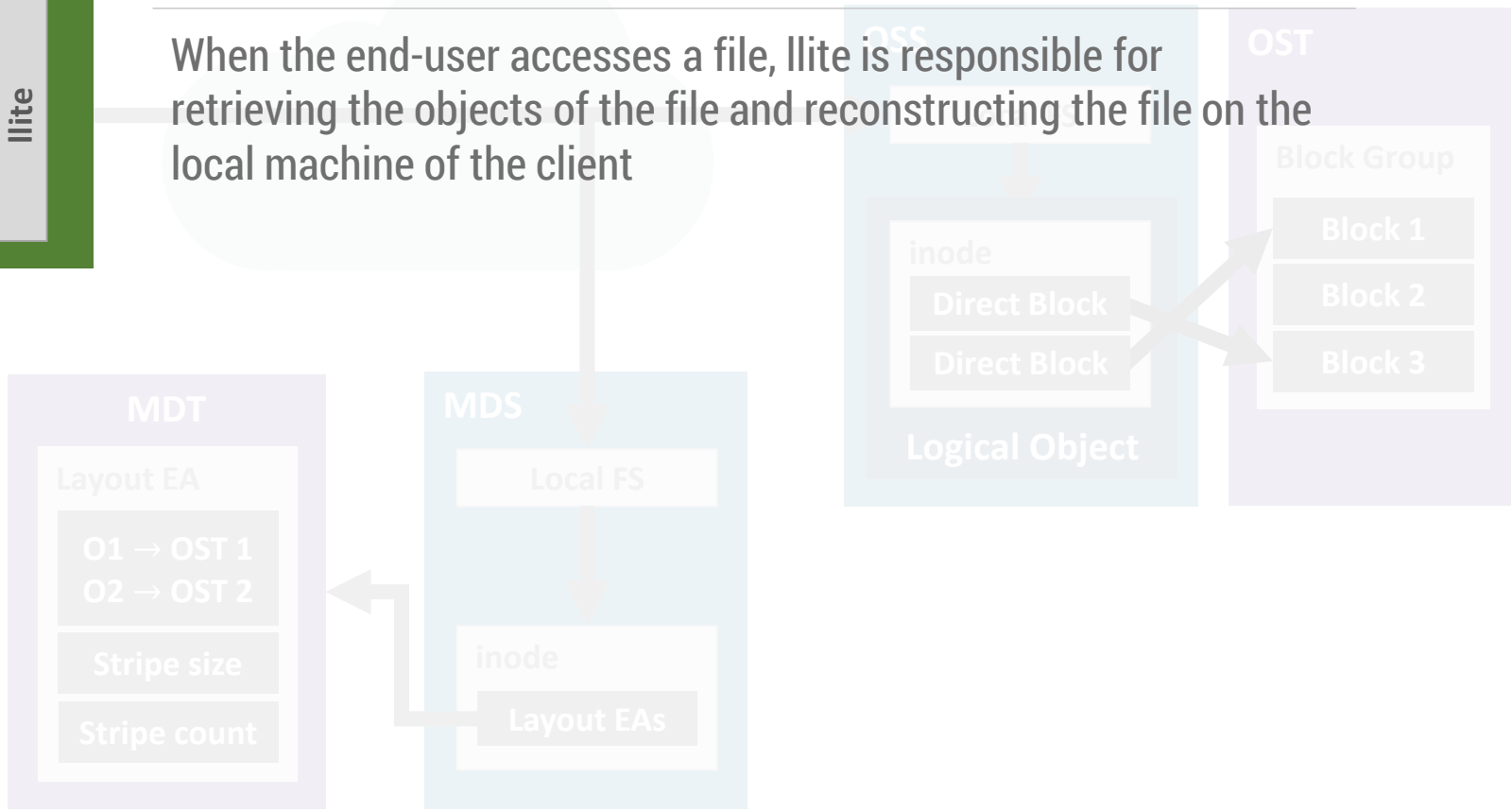


Background Information



llite

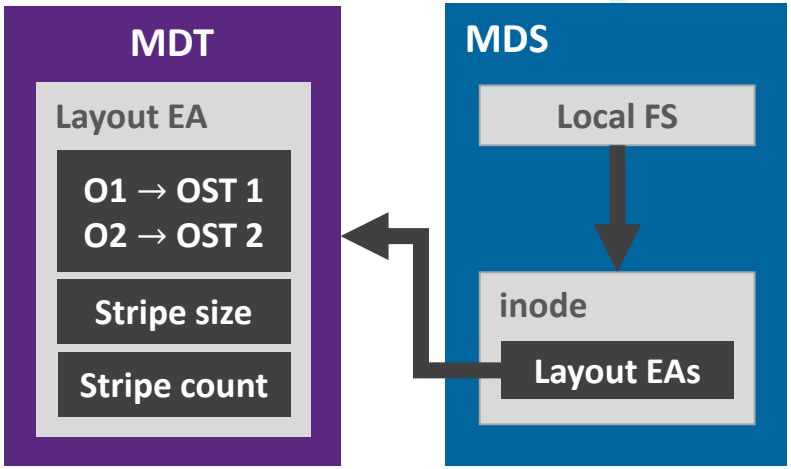
When the end-user accesses a file, llite is responsible for retrieving the objects of the file and reconstructing the file on the local machine of the client



Background Information

Obtain the file metadata

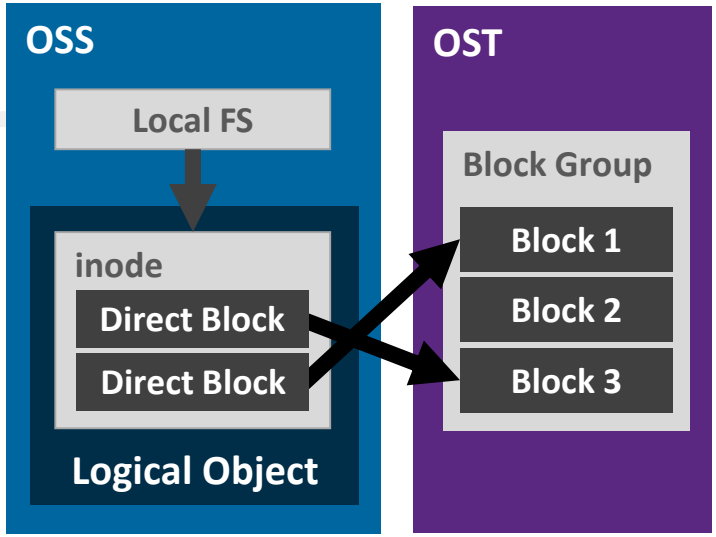
The llite component of the client fetches the metadata for the file in order to find the objects of the file and reconstruct the complete file from these constituent parts



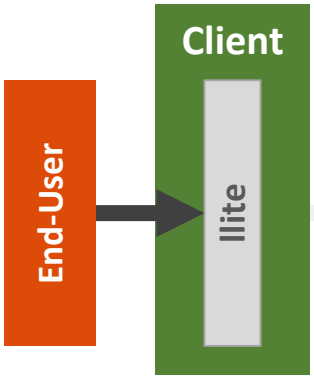
Background Information

Obtain the objects

Using the metadata, the llite component then fetches the objects directly from the OSSs and OSTs that manage and store the objects, respectively

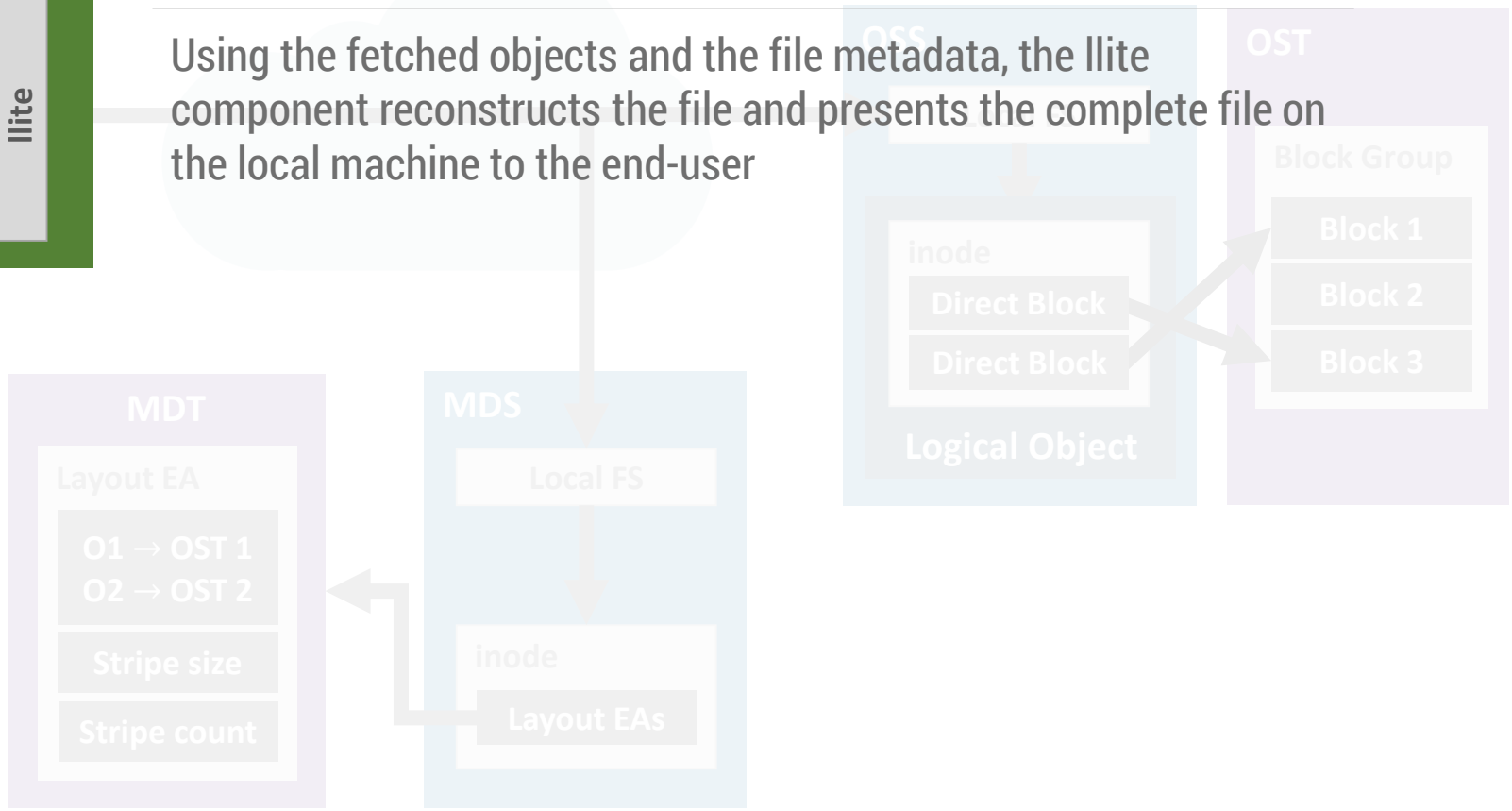


Background Information



Reconstruct the file

Using the fetched objects and the file metadata, the llite component reconstructs the file and presents the complete file on the local machine to the end-user



Problem Statement

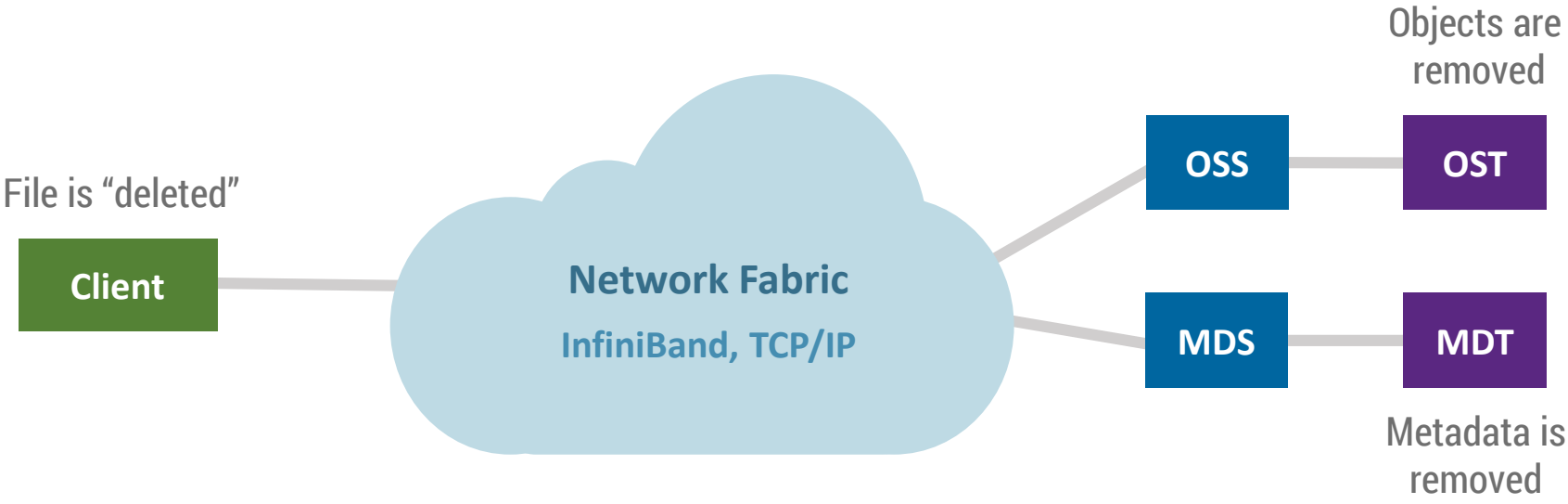
What is the problem that is being solved?

- While a great deal of research has been completed on distributed file systems, there is a lack of research into forensics and file recovery on these distributed systems
- This challenge is important for various customers:
  - Intelligence agencies
  - Enterprises and companies
  - Law enforcement
  - Individuals

Problem Statement

What happens when a file is deleted in Lustre?

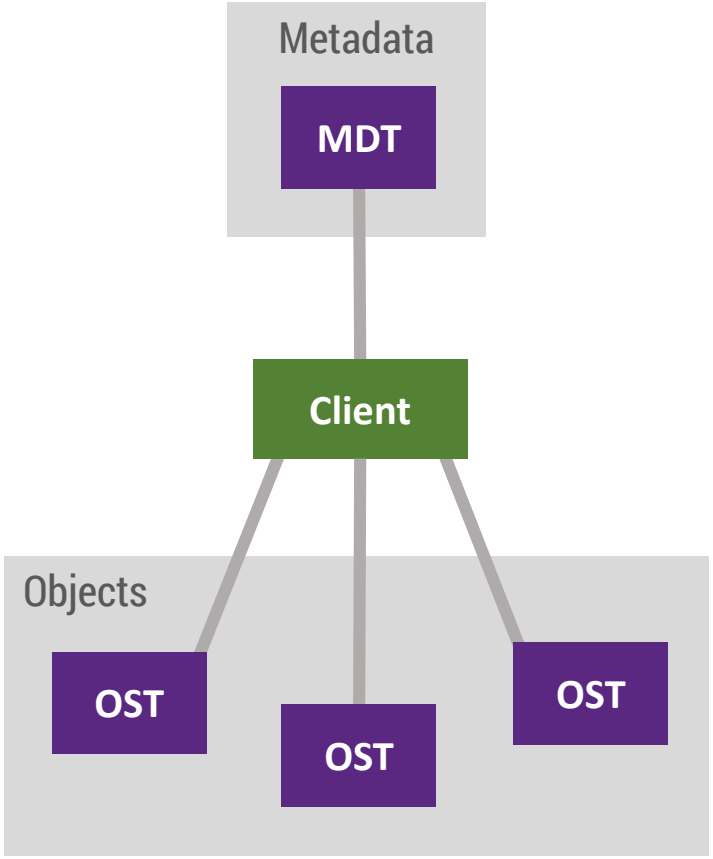
- When a file is deleted, or *unlinked*, the inode containing the metadata is removed from the MDT and the objects associated with the file are removed from the OSTs
- Once the process of removing the metadata and objects is complete, the file is considered unlinked from the Lustre file system



Problem Statement

What must be done to recover a file?

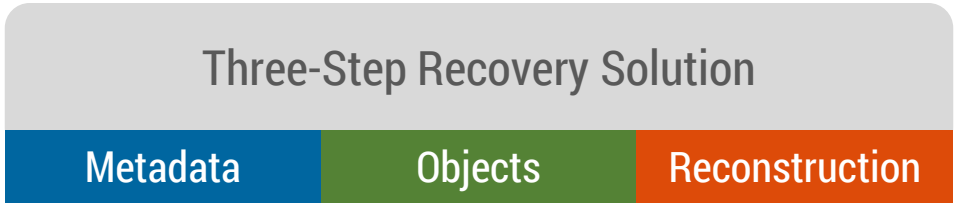
1. Discover where the objects that make up the file reside
2. Mount the OSTs containing the objects and retrieve the objects for the deleted file
3. Reconstruct the file from the objects



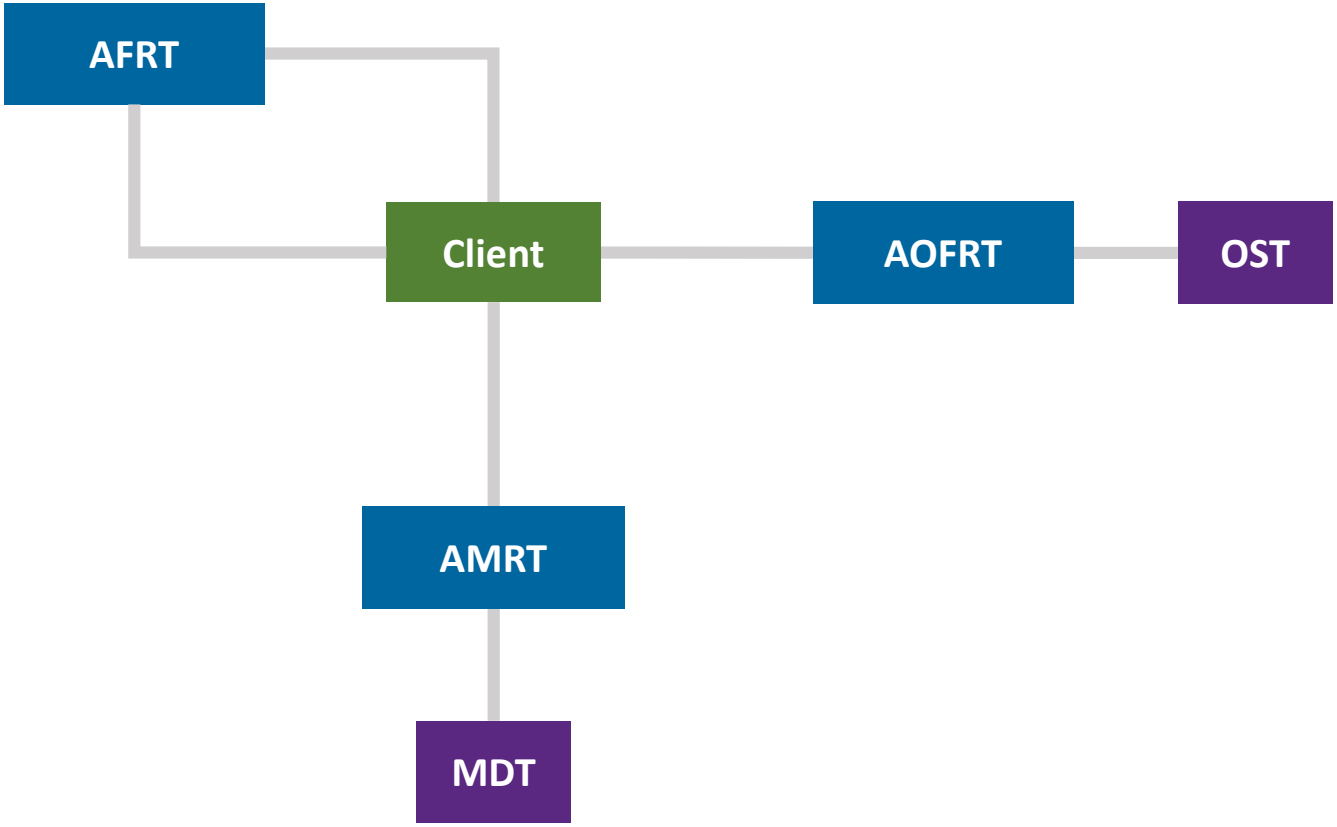
Solution

What is the approach?

- Divide the problem into steps for which solutions have already been devised:
  1. Recover the metadata for the file from the local file system of the MDT, which is a simple recovery of an inode from a local file system
  2. Recover the objects from the local file system of the OSTs, which is a simple recovery of a file from a local file system
  3. Reconstruct the file from the recovered objects, for which code already exists in the llite component of the Lustre file system client



Solution



Solution

AFRT

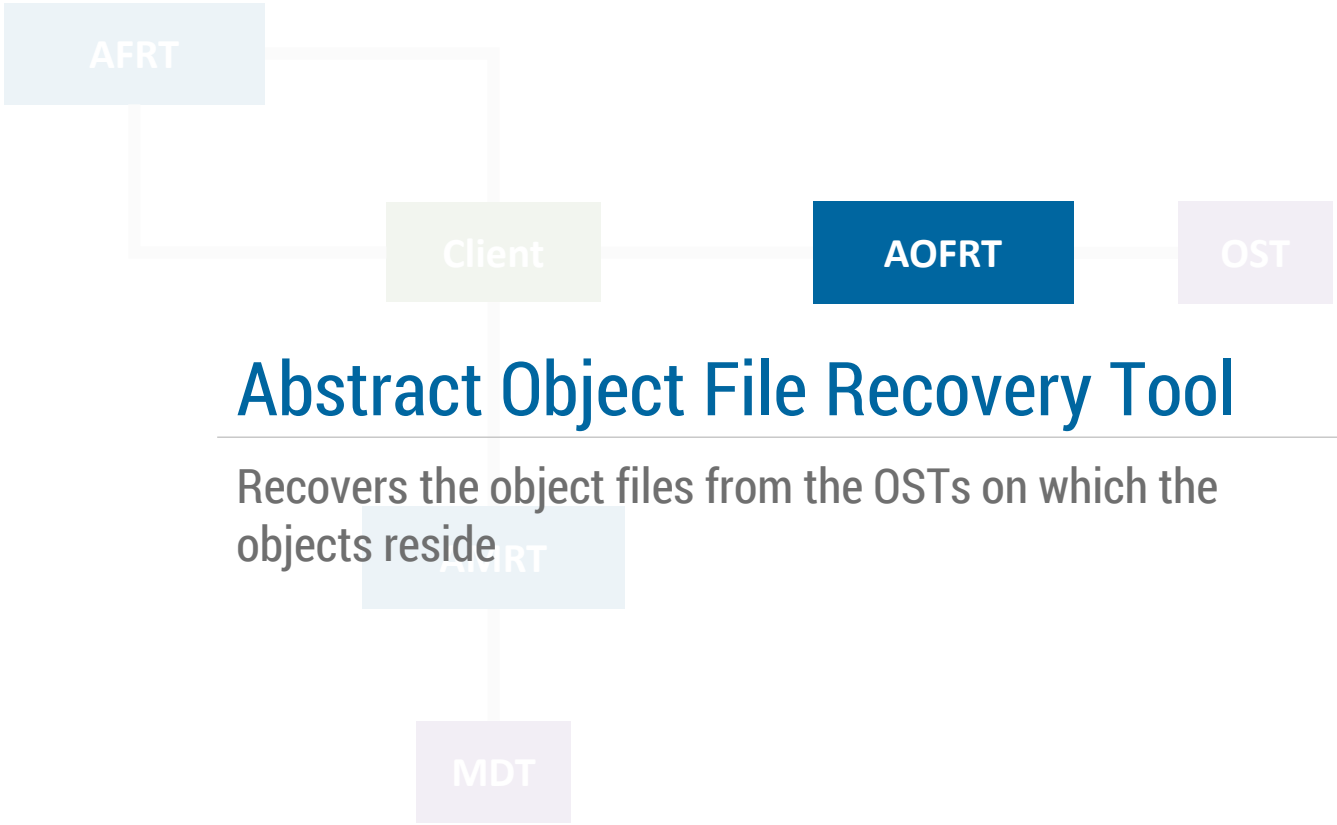
Abstract Metadata Recovery Tool

Recovers the inode and layout extended attributes associated with the deleted file from the MDT

AMRT

MDT

Solution



Abstract Object File Recovery Tool

Recovers the object files from the OSTs on which the objects reside

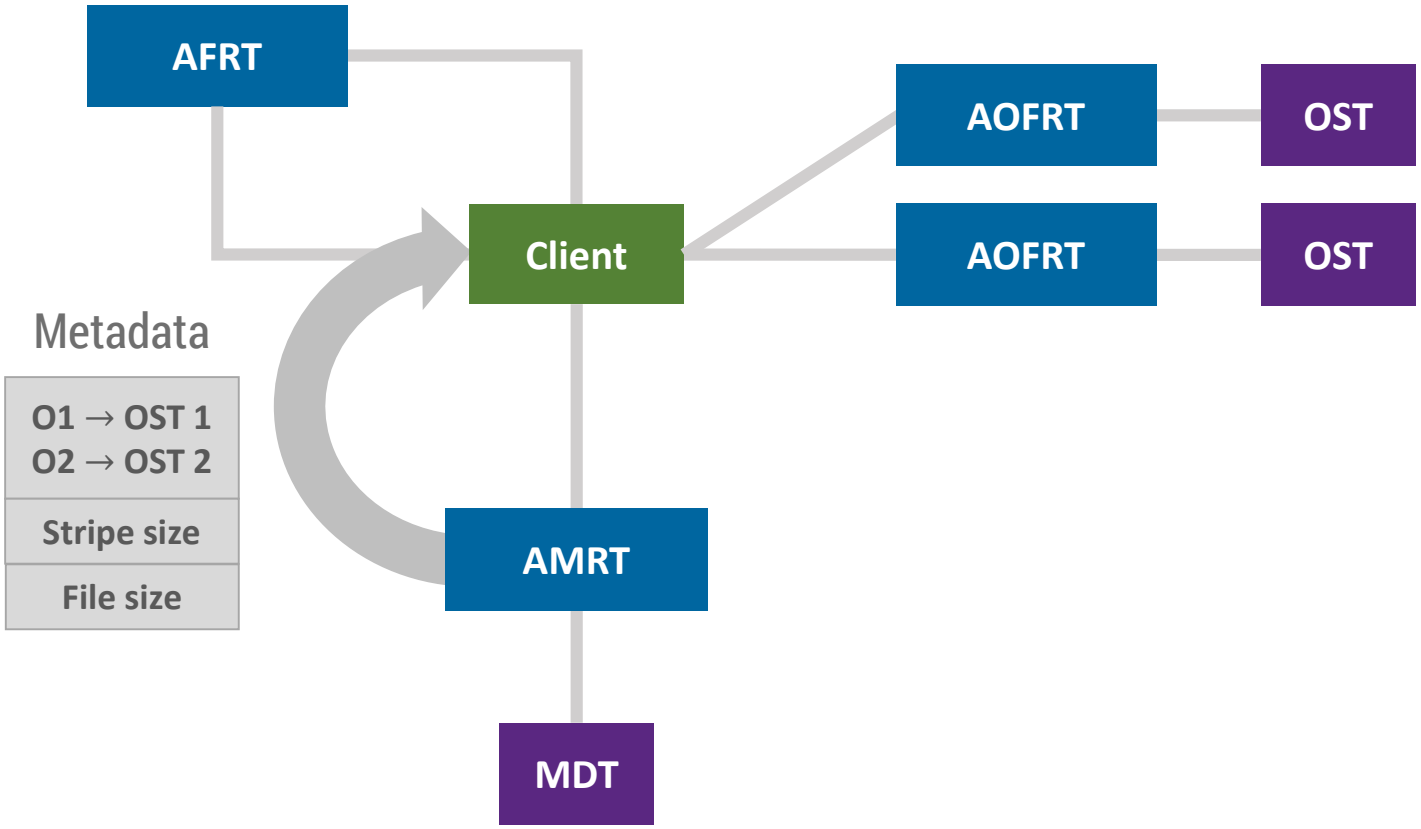
Solution

AFRT

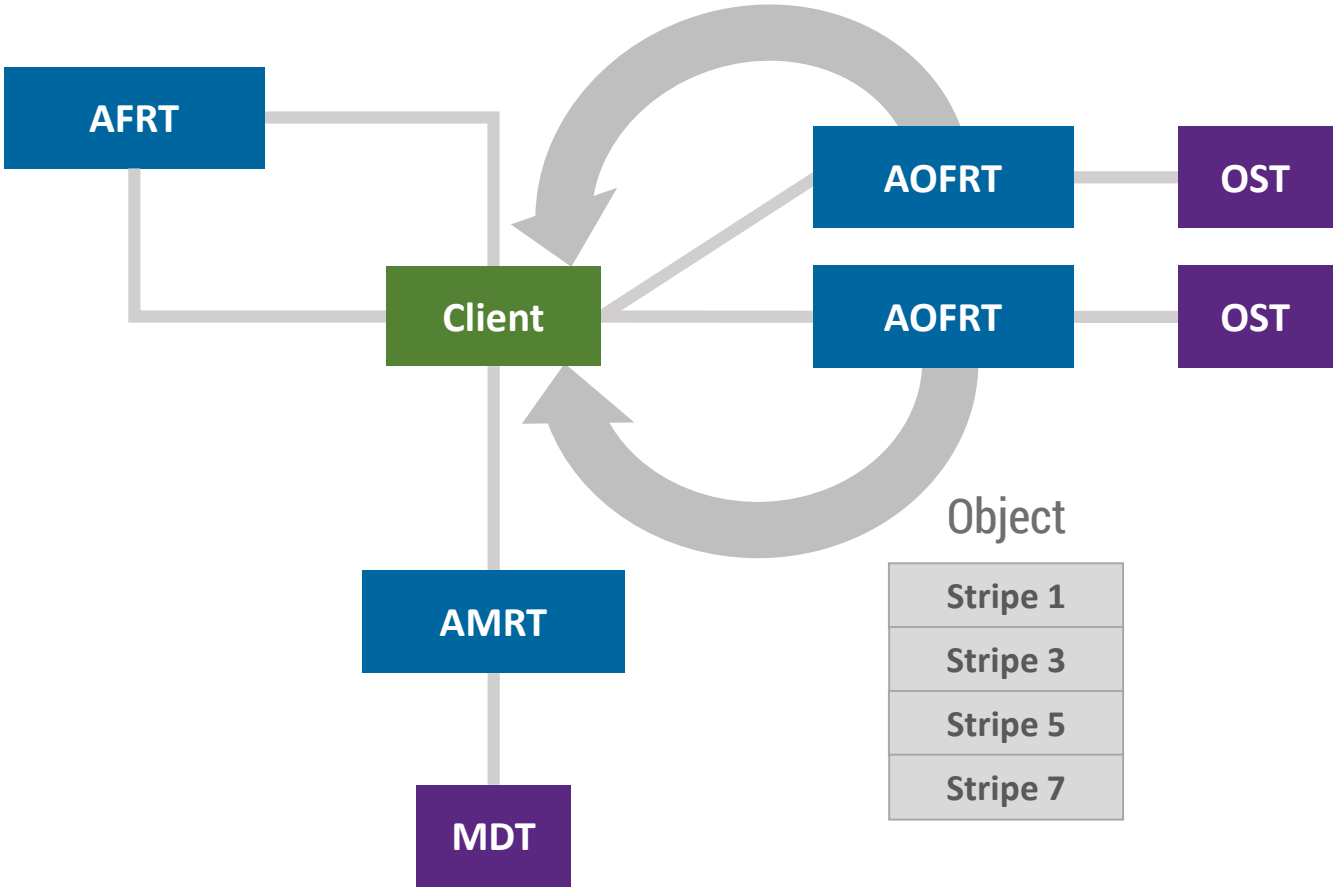
Abstract File Reconstruction Tool

Reconstructs the deleted file from the metadata recovered by the AMRT and the objects recovered by the AOFRT using the existing logic in the llite component

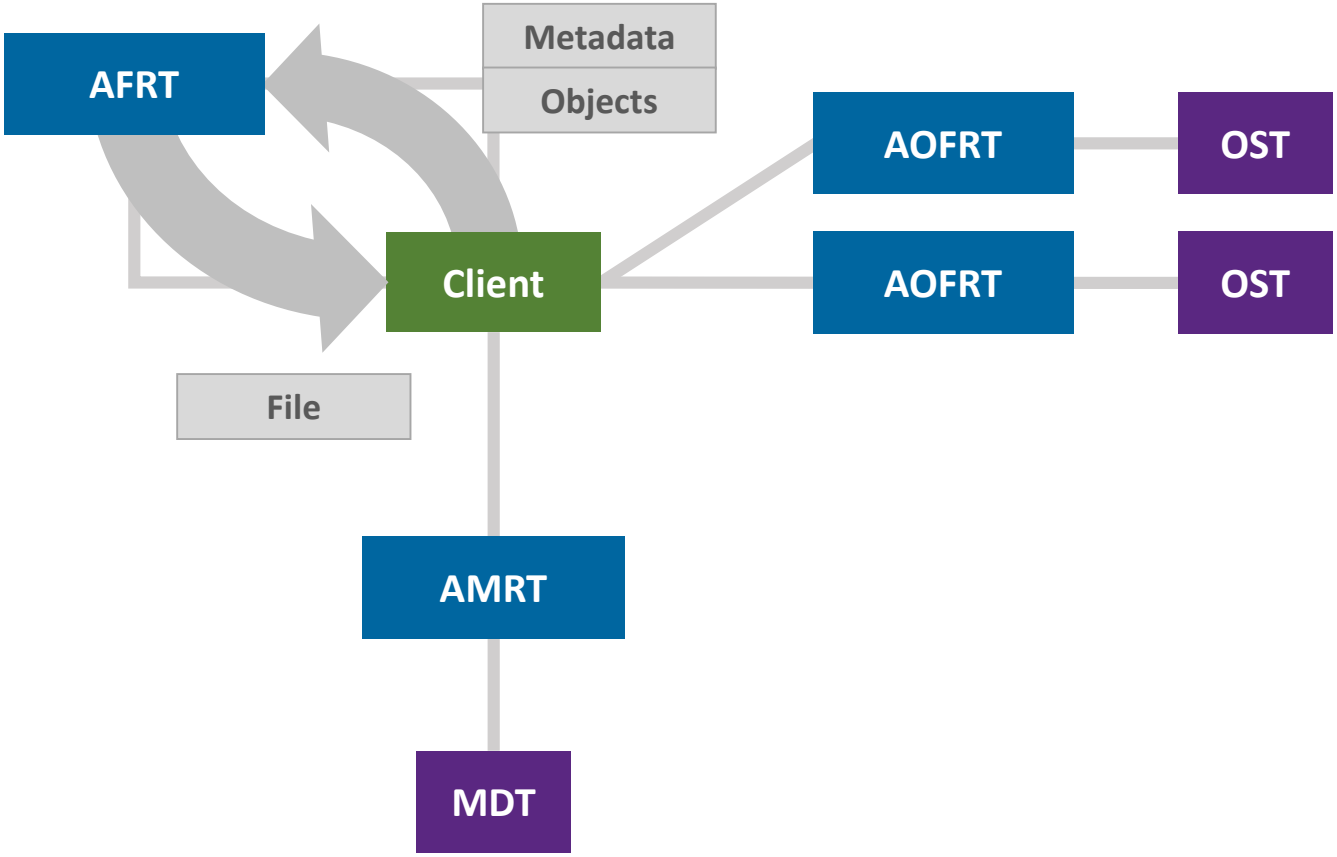
Solution


AFRT
AOFRT
OST
Client
AOFRT
OST
Metadata
O1 → OST 1
O2 → OST 2
Stripe size
File size
AMRT
MDT

Solution


AFRT
Client
AMRT
MDT
AOFRT
AOFRT
OST
OST
Object
Stripe 1
Stripe 3
Stripe 5
Stripe 7

Solution


Metadata
Objects
AFRT
AOFRT
OST
Client
AOFRT
OST
File
AMRT
MDT

Solution

What are the advantages?

- This solution is simple, leveraging the existing solutions to the problem of file recovery on a local file system (stands on the shoulders of localized file recovery)
- This algorithm nearly mimics the algorithm used by the Lustre file system to reconstruct a file when an end-user accesses a file through the client

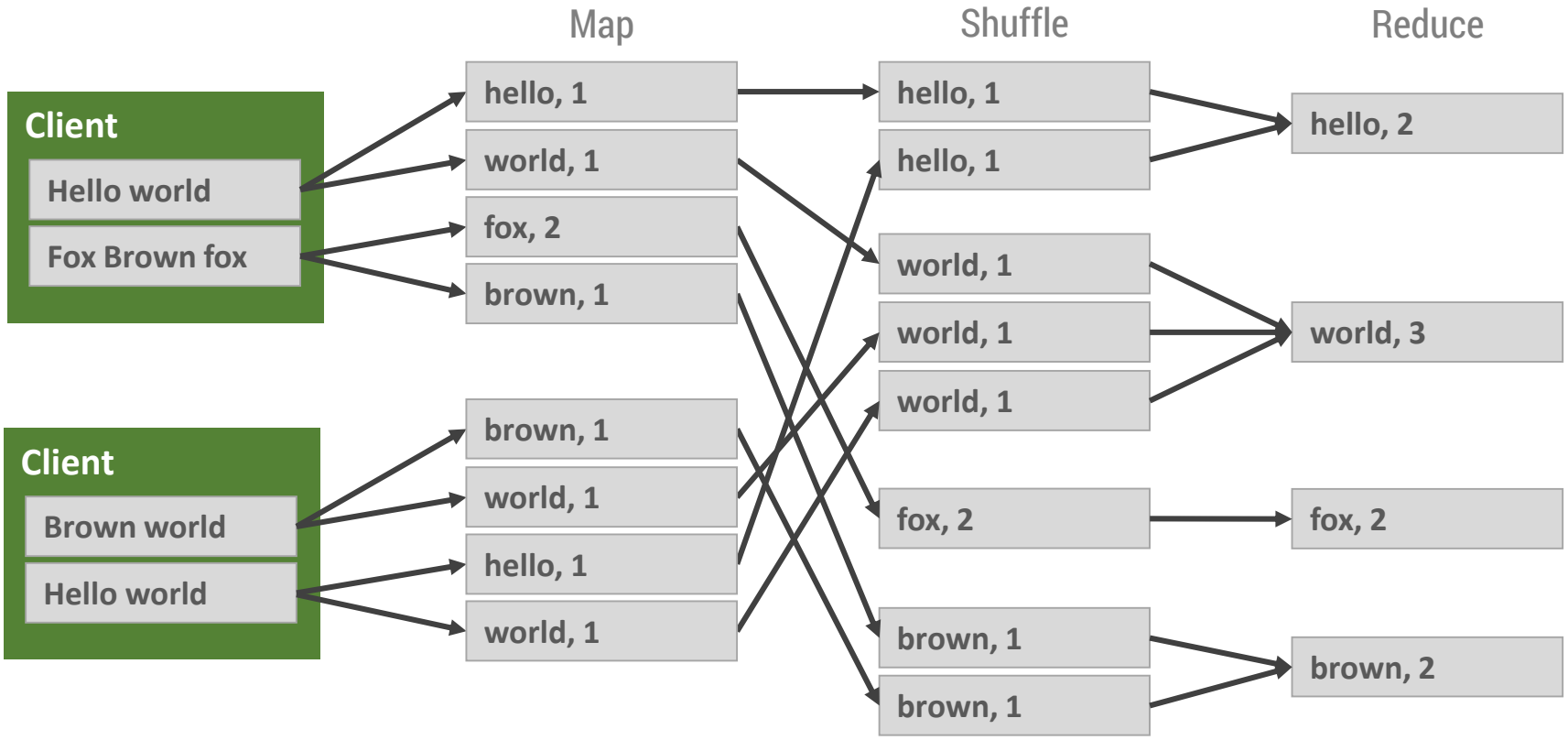
What are the disadvantages?

- This solution requires that all OSTs containing objects for the deleted file be directly mounted to the client system recovering the file
- Essentially a localized algorithm for use in a distributed environment

Improvements can be made by making this a distributed algorithm →

Solution

What is MapReduce?



Solution

How does this help?

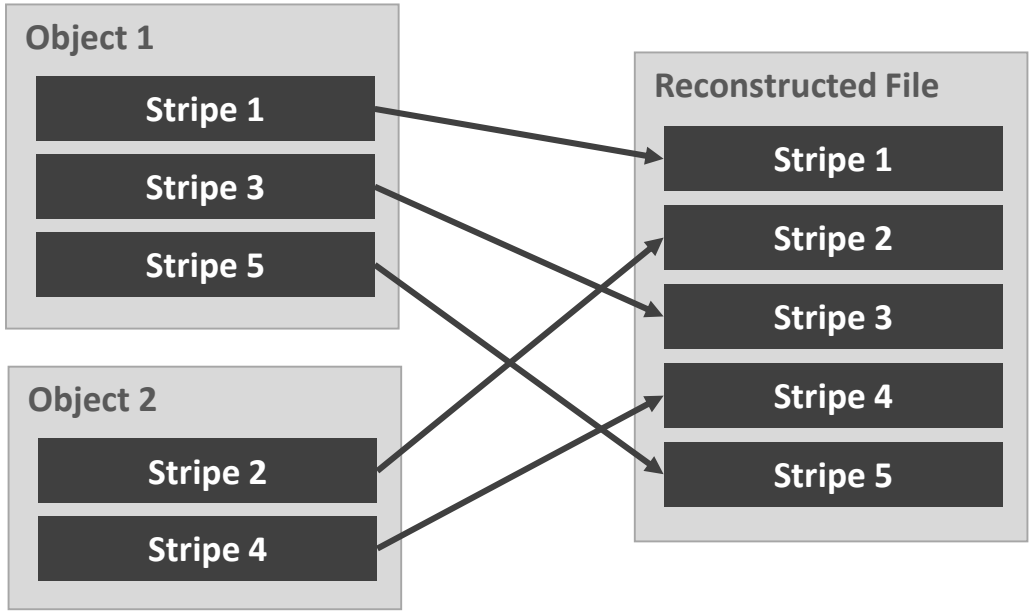
- What if the parts of a file can be mapped and the MapReduce process be used a way to aggregate the parts into a file?
- Combining objects does not produce the reconstructed file, since the data in a file is *striped* across the objects: The data in an object is non-continuous

- A unit of finer granularity is needed to be able to combine the parts of the file into the reconstructed file using MapReduce

Solution

What is the correct unit of granularity?

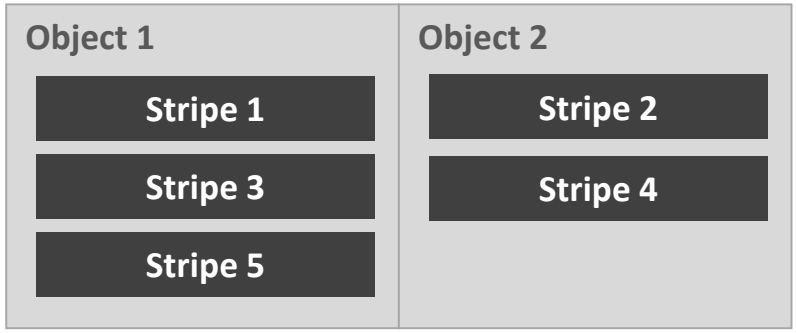
- The stripes of a file can be combined in order to reconstruct the file
- Using stripes requires the metadata (stripe size, file size, and ordered list of objects)



Solution

Where do the stripes belong?

- The striping of a file across objects can be viewed as a table where columns are the objects (or the OST on which the object resides) and the rows are one round-trip in the round-robin striping algorithm
- Essentially, the striping algorithm is reversible if the stripe size, file size, and ordered list of objects is known

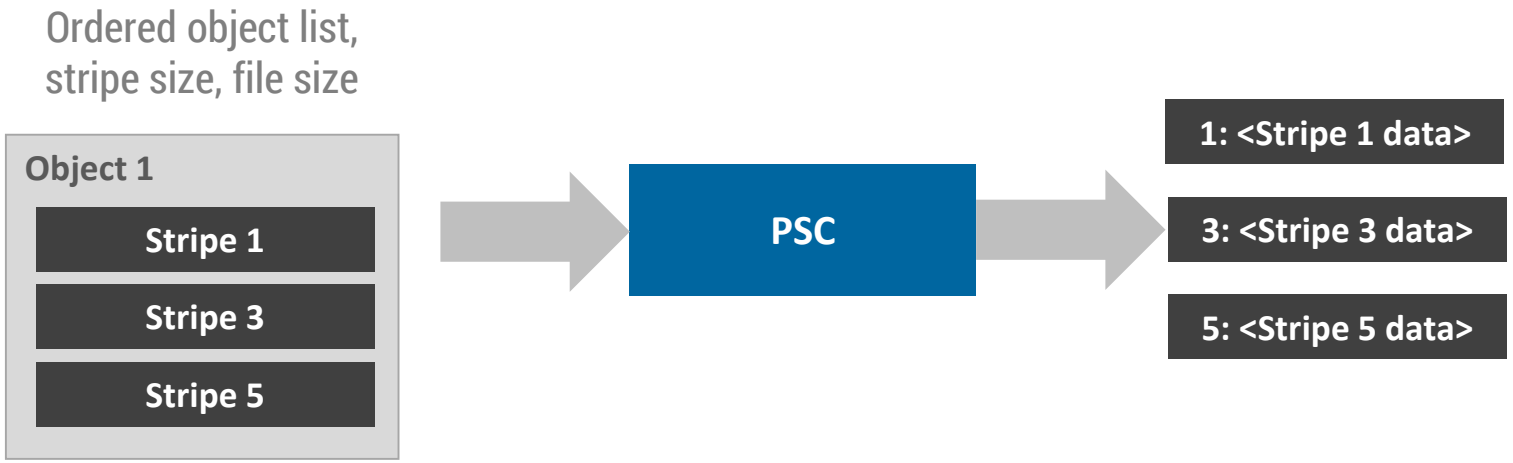


- Read until stripe size is read
- Read from next object until strip size is read
- Continue until the number of bytes read is equal to the file size

Solution

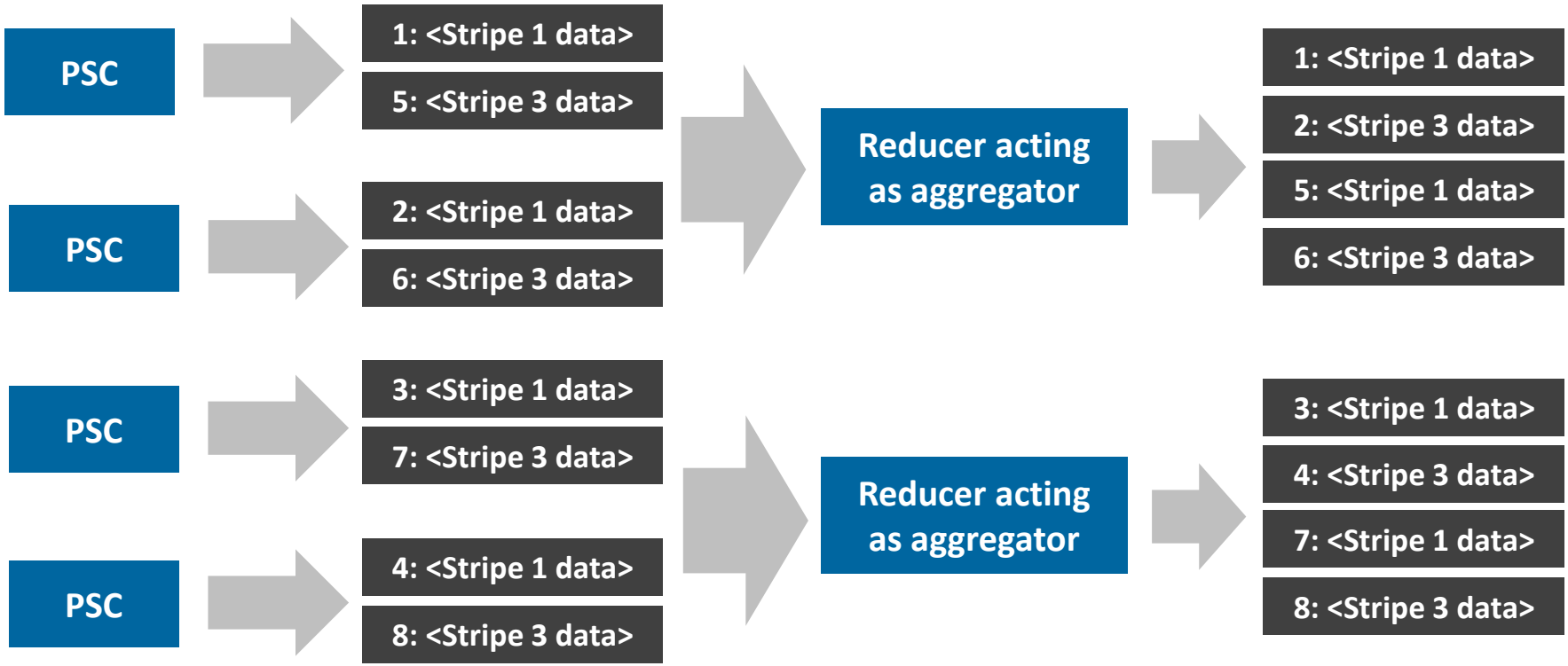
How can the stripes be obtained?

- A component, called the Partial Striping Component (PSC), is an extension of the AOFRT that produces the individual stripes contained in a recovered file
- Using the stripe size, file size, and ordered list of objects, the stripe data can be recovered and keyed by the stripe index



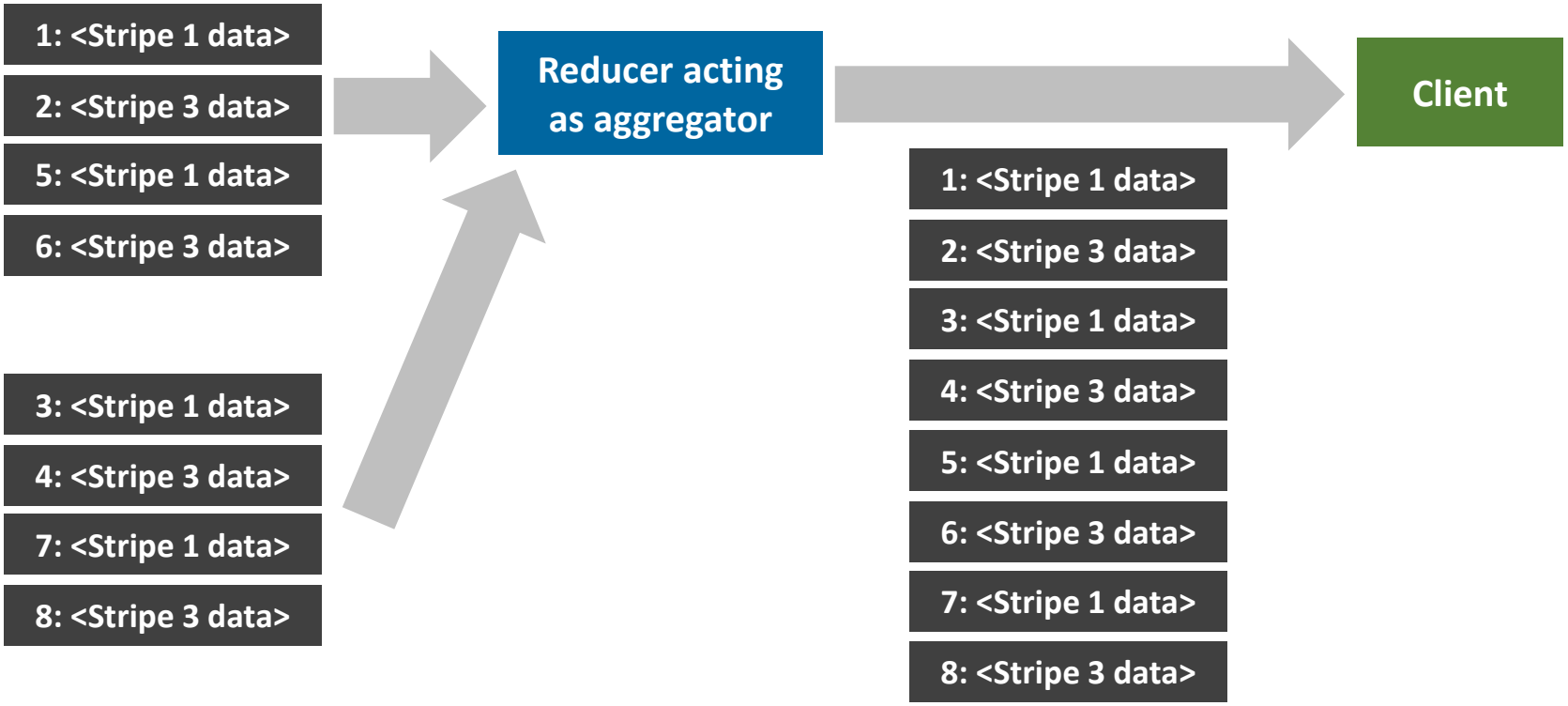
Solution

How does relate to MapReduce?

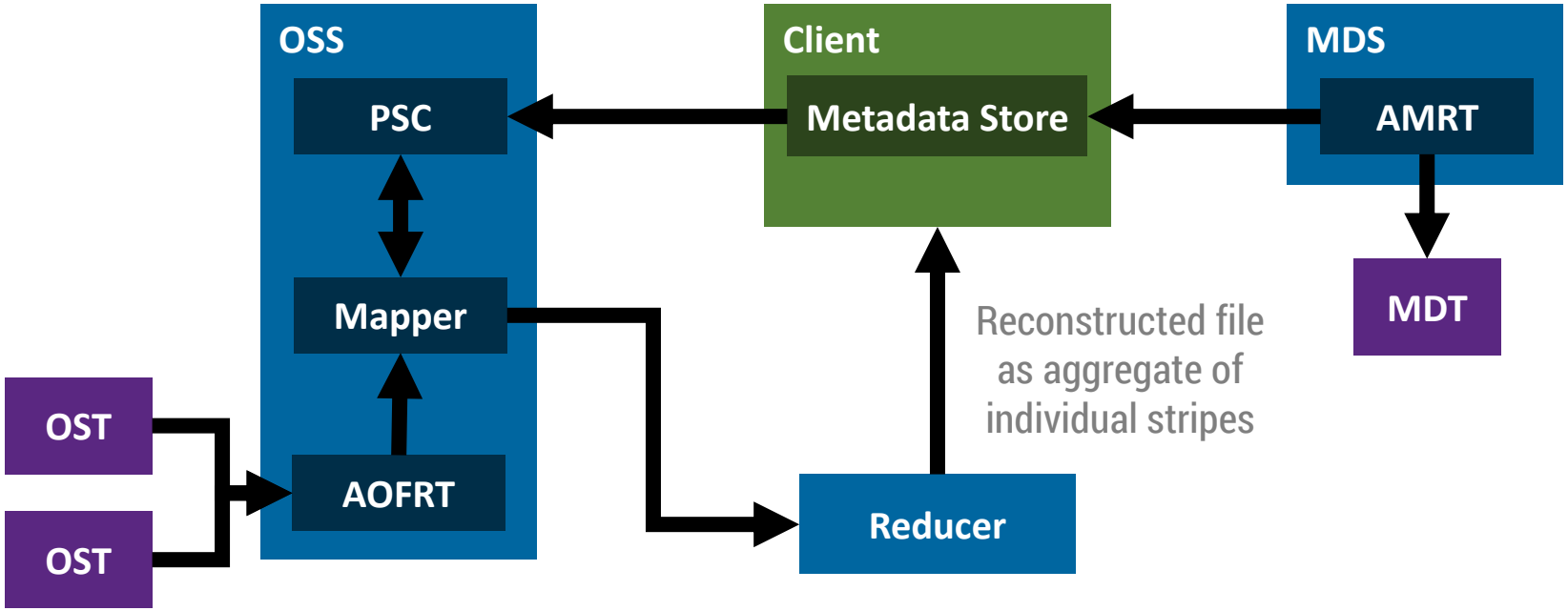


Solution

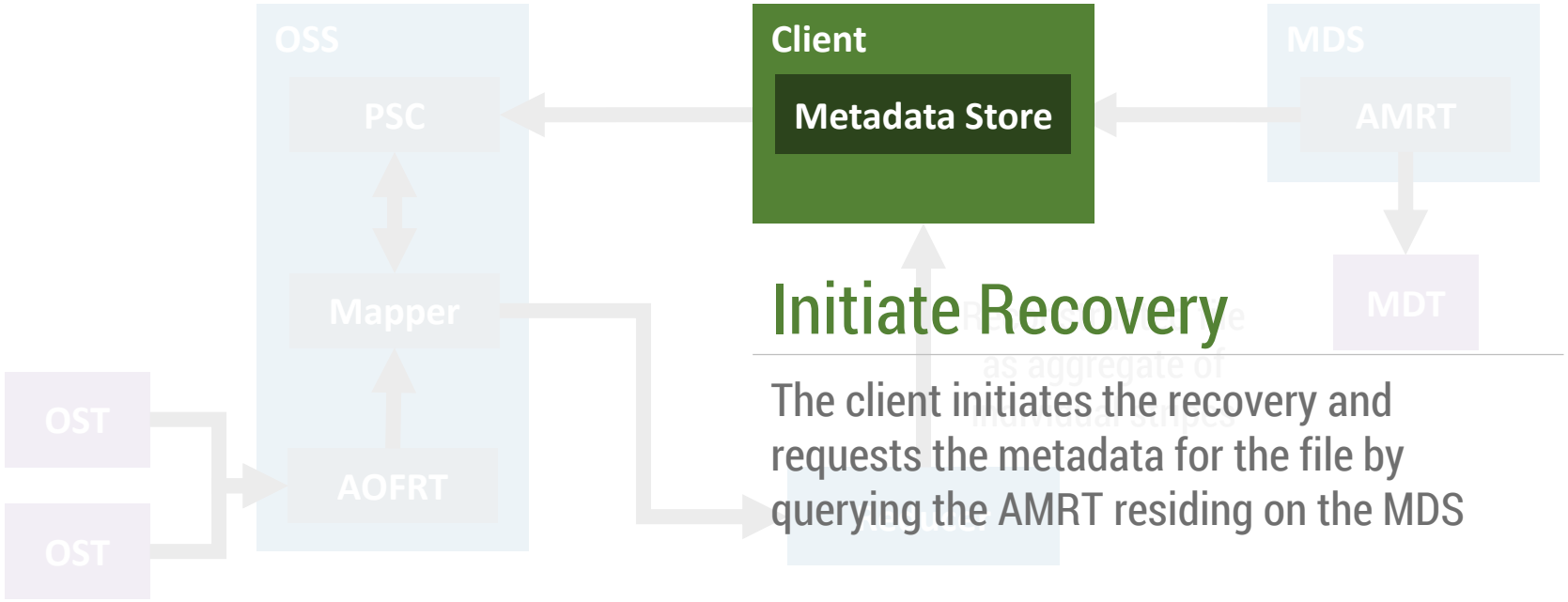
How does relate to MapReduce?



Solution


OSS
PSC
Mapper
AOFRT
OST
OST
Client
Metadata Store
MDS
AMRT
MDT
Reconstructed file as aggregate of individual stripes
Reducer

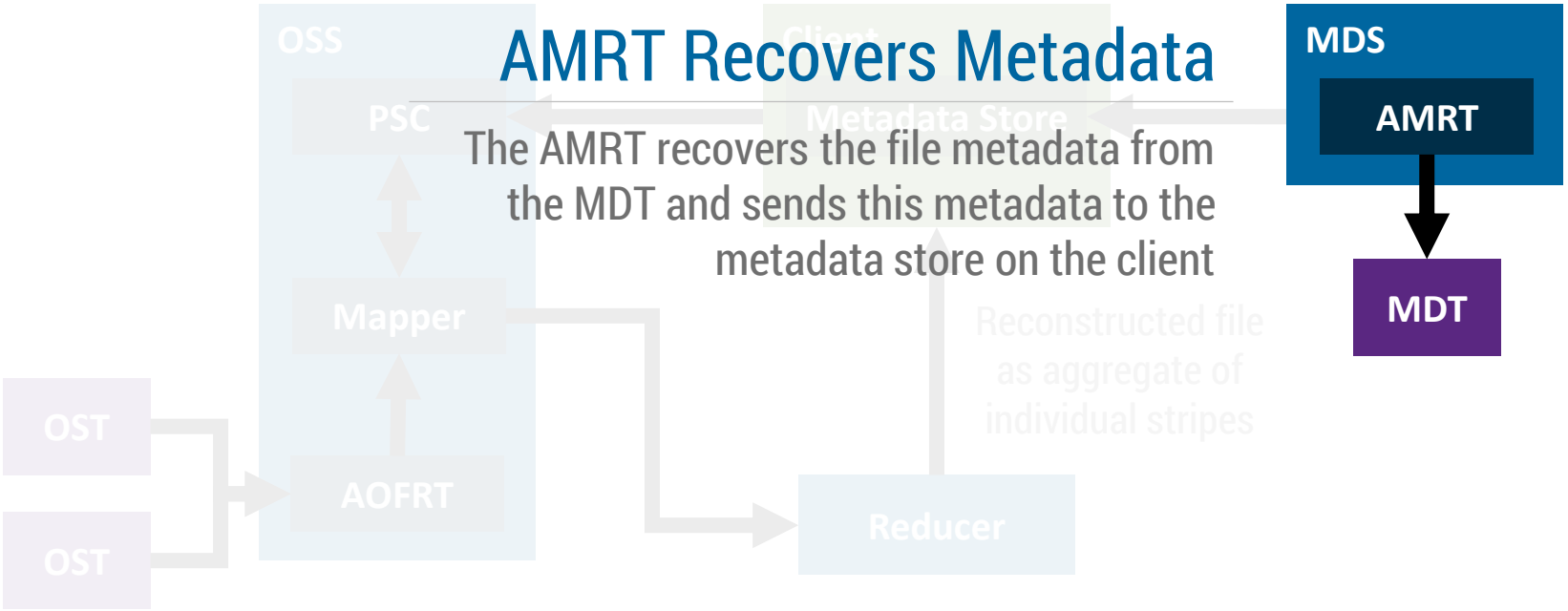
Solution


OSS
PSC
Mapper
AOFRT
OST
OST
Client
Metadata Store
MDS
AMRT
MDT
Initiate Recovery
The client initiates the recovery and requests the metadata for the file by querying the AMRT residing on the MDS

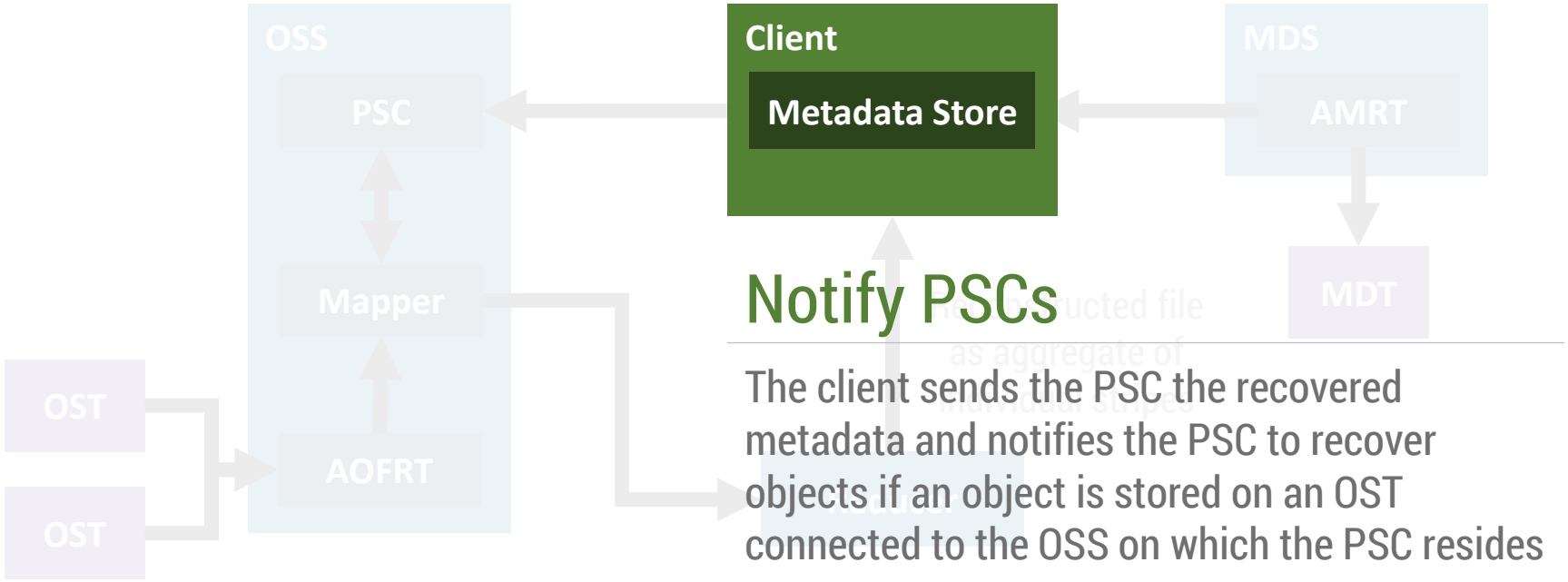
Solution

AMRT Recovers Metadata

The AMRT recovers the file metadata from the MDT and sends this metadata to the metadata store on the client



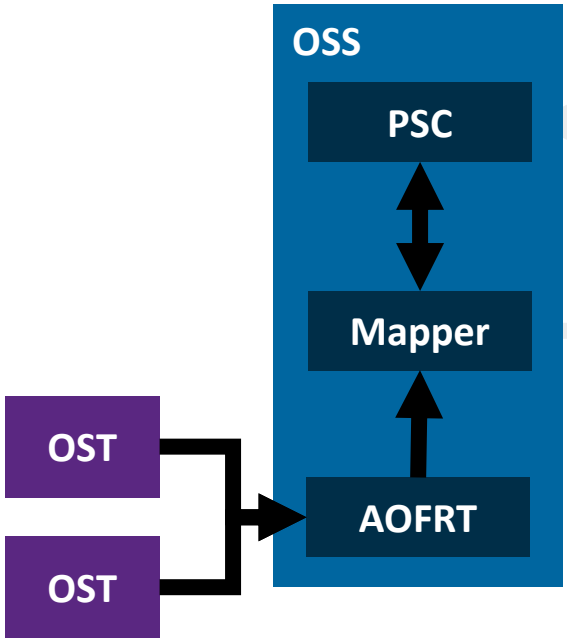
Solution



Notify PSCs

The client sends the PSC the recovered metadata and notifies the PSC to recover objects if an object is stored on an OST connected to the OSS on which the PSC resides

Solution



Objects Recovered and Mapped

If a needed object resides on an OST connected to the OSS on which the PSC resides, the AOFRT recovers the object from the OST

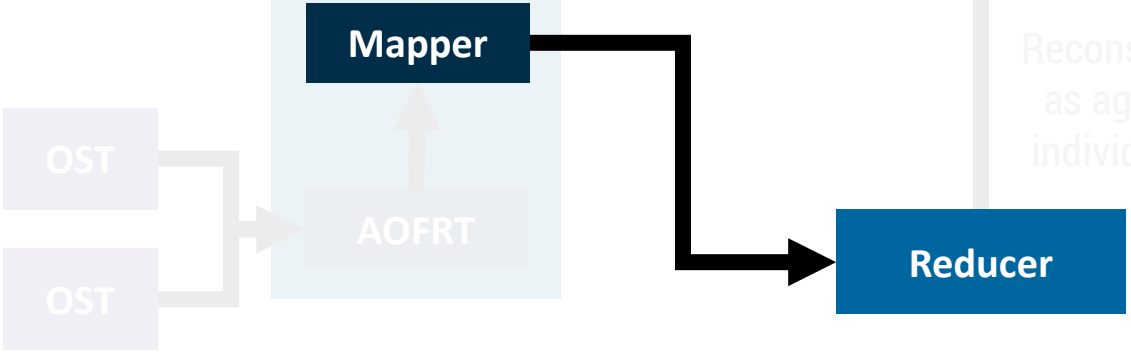
The object is then sent to the PSC and, using the metadata, the PSC extracts the stripes from the object

The mapper then keys each stripes by the index of the stripe extracted by the PSC

Solution

Keyed Stripes are Aggregated

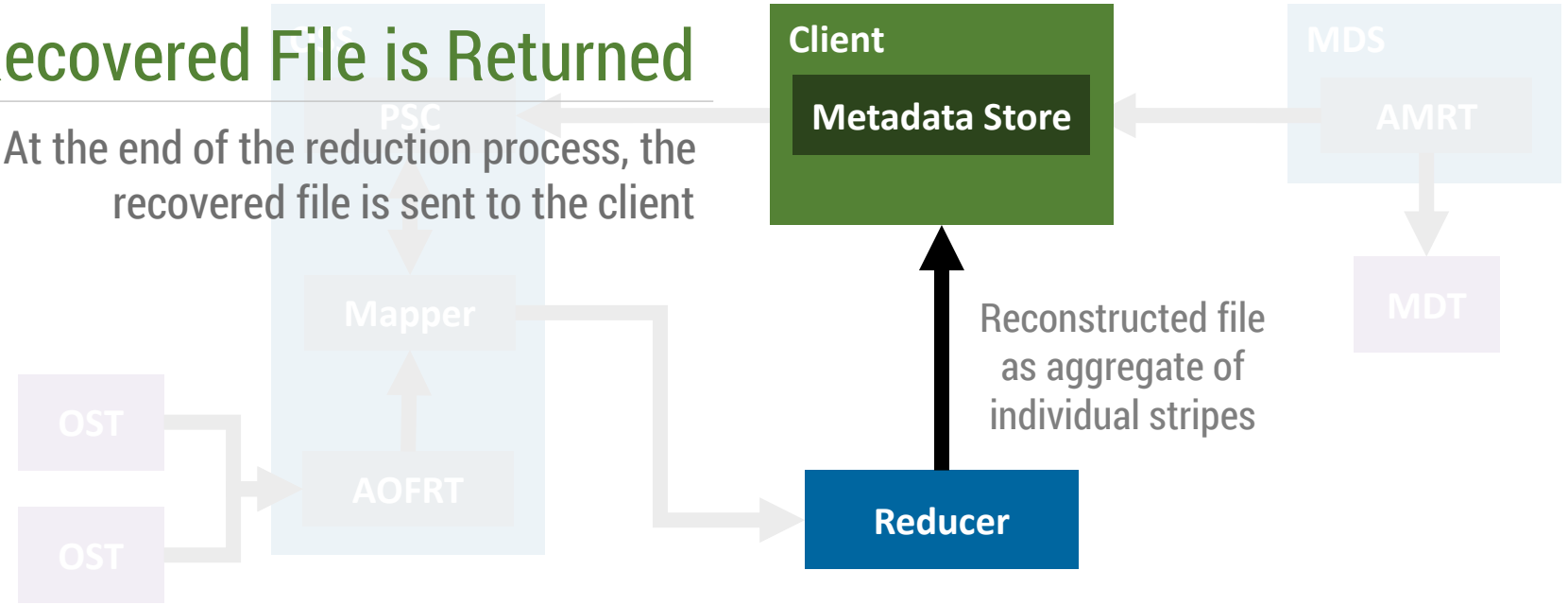
The keyed stripes are then sent to the reducer, where they are aggregated with the keyed stripes from other OSSs



Solution

Recovered File is Returned

At the end of the reduction process, the recovered file is sent to the client



Solution

What are the advantages over the simple solution?

- The MDT on which the metadata is stored does not have to be mounted to the client
- The OSTs on which the object files are stored do not have to be mounted to the client
- This solution is a distributed solution to a distributed problem
  - The solution can be scaled to the number of OSTs on which objects are stored by increasing the number of compute nodes performing the reduce jobs
  - The distributed solution provides the same benefits as a distributed file system: The reconstruction process can be completed in parallel using MapReduce

Side note

There is a tinge of irony: The Lustre file system is being explored as a way to improve the performance of MapReduce clusters

Conclusion

Research gap

- While distributed file systems, such as Lustre, are highly researched, research in forensics and file recovery on these systems is greatly lacking

Simplicity of solution

- Although file systems are complex software systems, they are essentially composites of local file systems, and therefore, the process of recovering a file is basically the process of recovering a file from a local file system, repeated multiple times

Future research

- While a solution architecture has been devised, it has not been implemented
- Future research should be conducted on how to improve this solution, implement the presented architecture, and gain further insight in the Lustre file system

Conclusion

A few comments on Lustre

- Impressively complex file system
- There is an overwhelming lack of documentation and technical detail:
  - Two levels of expertise: Novice or expert
  - There is either no documentation, documentation that lacks technical detail, or documentation that has technical detail, but is out-of-date
- It will be very interesting to see where the Lustre file system ends up in years to come

Conclusion

Acknowledgements

- Dr. Seker for his advisement, guidance, and patient support
- Dr. Sarp Oral at Oak Ridge National Laboratory for his guidance on the technical underpinnings of Lustre and his expertise during the research phase of this project
- Many other unnamed students and friends that have helped me during research, report writing, and presentation; without you, none of this project would have come together

Question & Answer