Achieving Coherent and Aggressive Caching in DFS, GlusterFS

Poornima G
Red Hat

Soumya Koduri
Red Hat

AGENDA

- Intro to GlusterFS
- Caching in DFS
- Caching in GlusterFS and limitations
- Methods to achieve coherency
  - Leases : Design and Challenges
  - File system notifications : Design and challenges
- New cache mechanism proposed

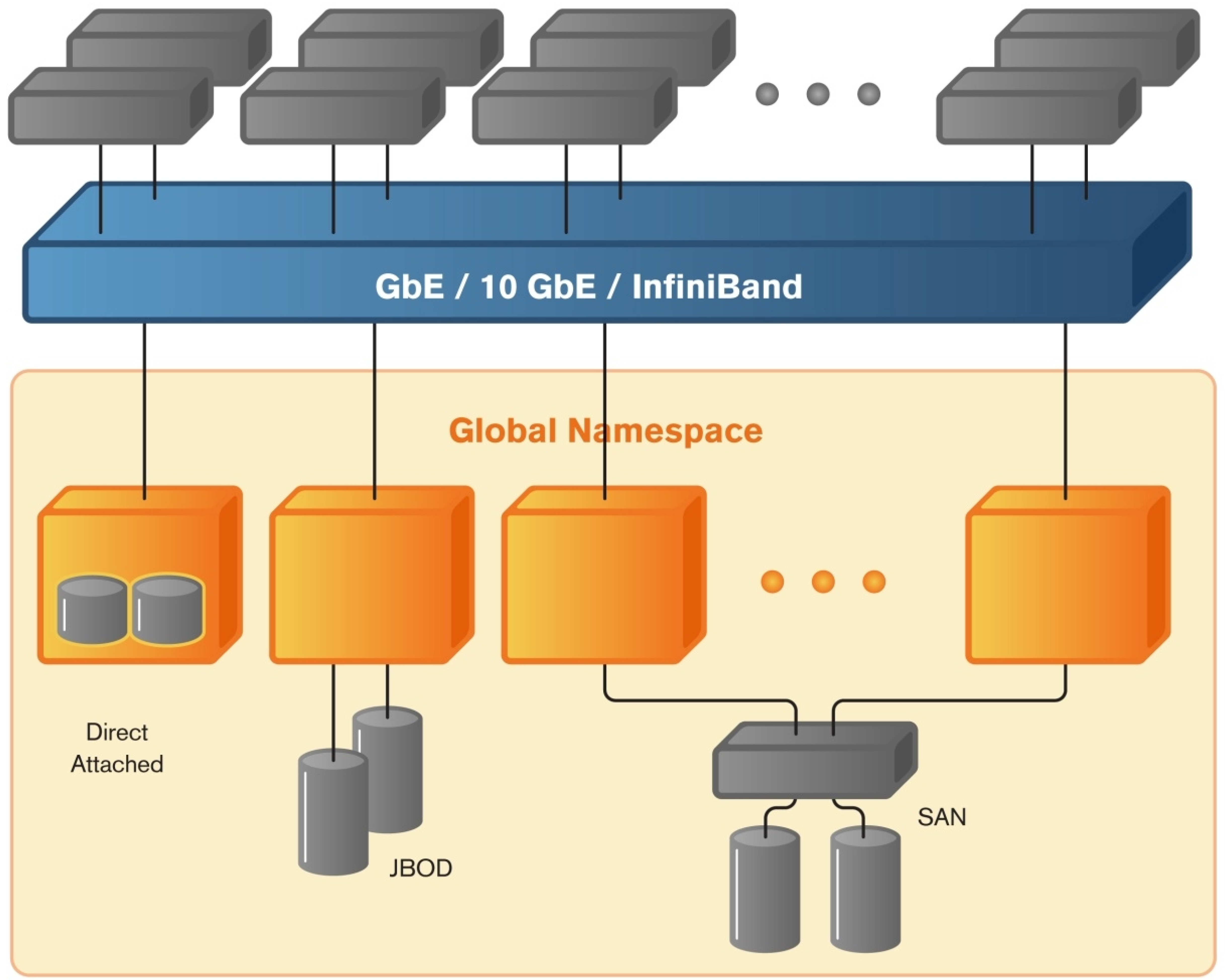
GlusterFS

- An open source, scale-out distributed file system
- Software Only and operates in user-space
- Aggregates Storage into a single unified namespace
- No metadata server architecture
- Provides a modular, stackable design
- Runs on commodity hardware

Architecture

- Data is stored on disk using native formats (e.g. ext4, XFS)
- Has client and server components
  - Servers, known as storage bricks, export local filesystem as volume
  - Clients, creates composite virtual volumes from multiple remote servers using stackable translators
  - Management service manages volumes and cluster membership

GlusterFS Deployment



Multiprotocol Client Support

- GlusterFS Native
- NFS
- CIFS
- HTTP
- WebDAV
- FTP

Clustered Storage Building Blocks

Flexible Back-end

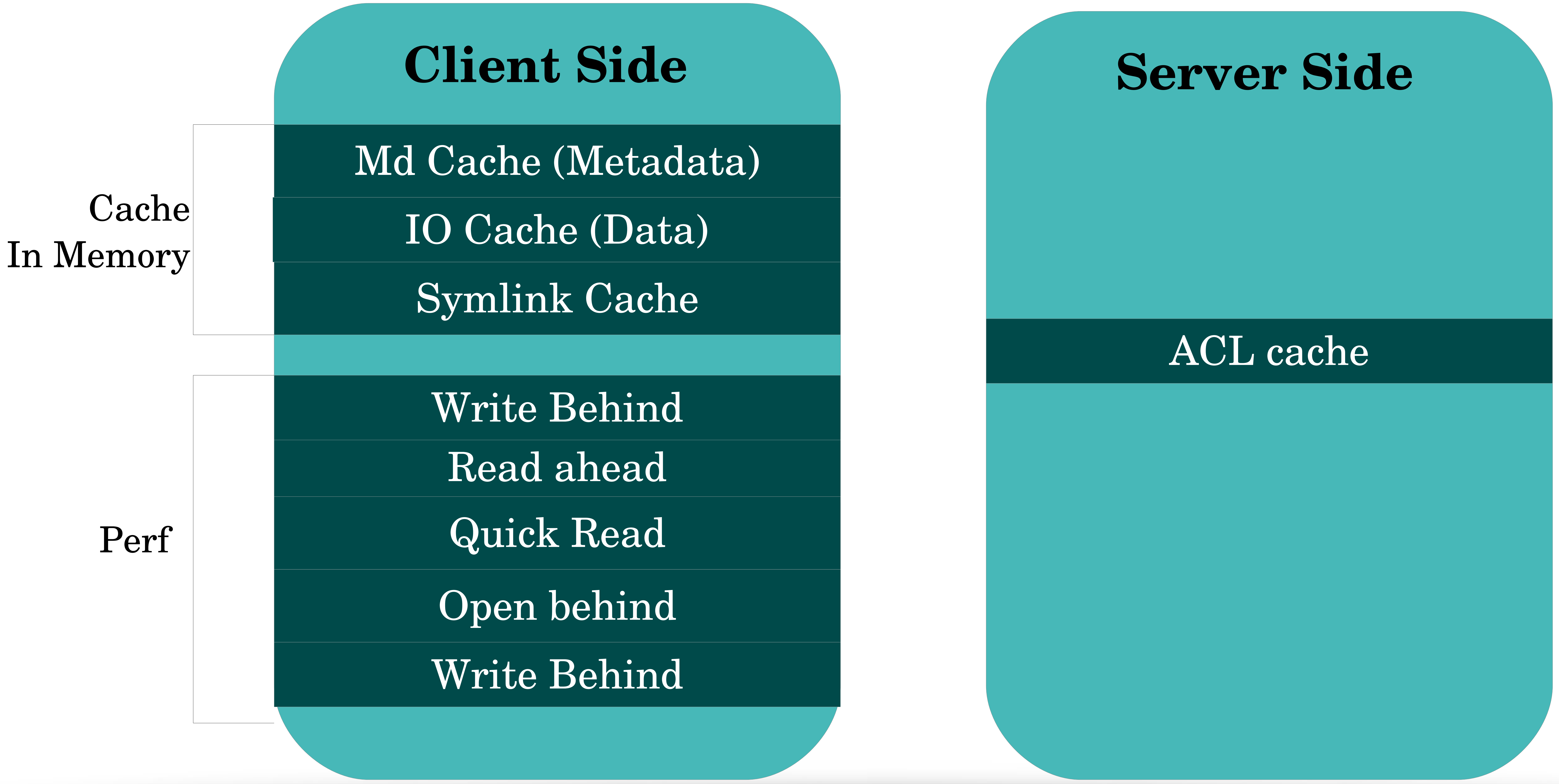
Caching

Caching in DFS

- Why caching?
  - Reduces network traffic.
  - Reduces contention for disk, resulting in increased performance.
- Cache policies:
  - location : server/client
  - where : in-memory/disk
  - type : file/block
  - content : data/metadata
- Challenges:
  - Achieve cache coherency, without adversely affecting performance.

Caching in Gluster

- Existing cache mechanisms:



Caching in Gluster - Limitations

- Not distributed cache coherent
- In memory caching makes it less aggressive
- Cache is timeout based, hence cache is invalidated in short intervals
- Could lead to degraded performance, for one time access workloads
- Network disconnects may lead to silent data corruptions

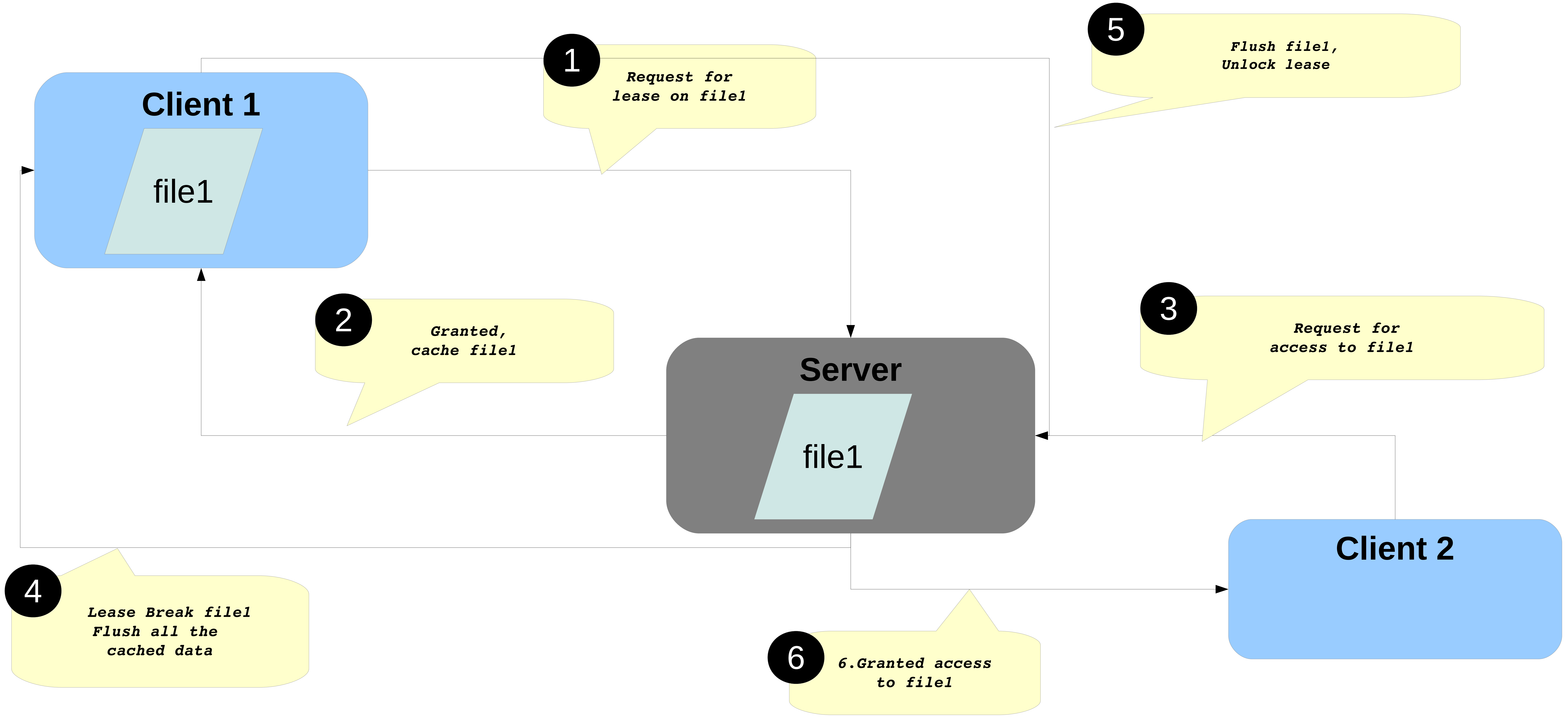
Methods to achieve Cache coherency

Leases

Leases

- Locking mechanism, which guarantees the client, certain semantics with respect to the sharing of the file/directory with other clients.
- Granted on regular files (file/directory).
- Protocol for cache consistency and aggressive caching.

Leases


Client 1
file1
1
Request for lease on file1
5
Flush file1, Unlock lease
2
Granted, cache file1
Server
file1
3
Request for access to file1
Client 2
4
Lease Break file1 Flush all the cached data
6
6.Granted access to file1

Leases

- Lease locks and its manifestations:

SMB		NFSV4 Delegations	Linux Lease Locks
Oplocks	**Leases**		
Level 1 (Exclusive)	Read	Read Delegations	Read lease
Level 2 (Read)	Read Handle	Write Delegations	Write Lease
Batch (Handle + exclusive)	Read/Write	Directory Delegations	
	Read/Write Handle		
	Directory Leases		

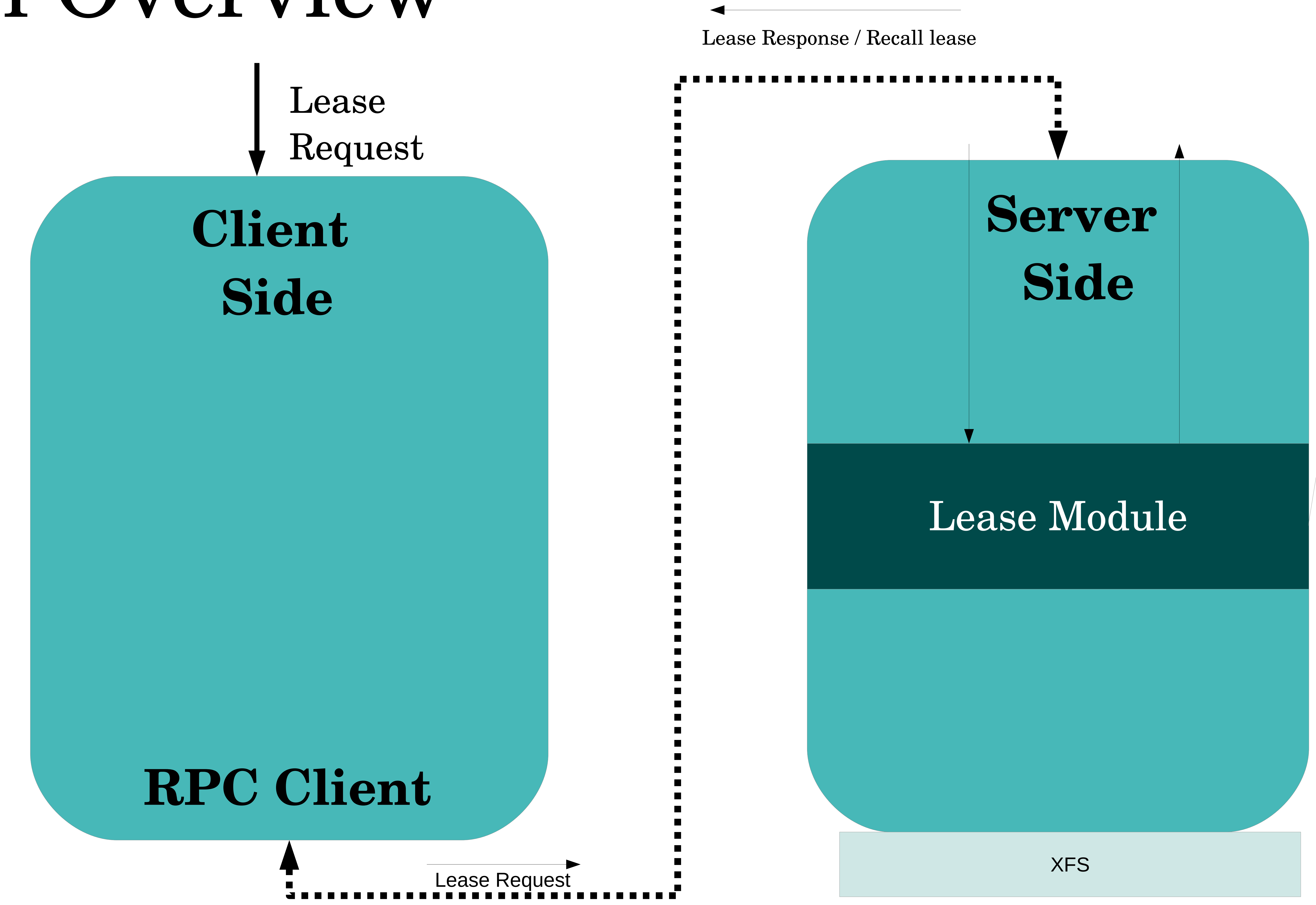
Leases on Gluster

Types	Break Conditions
Read	•The file is opened in a manner that can change the file •File data/size changed •Byte range lock of conflicting mode requested.
Read Write	• Open(read/write mode) from different client. • File data/size is changed
Handle	•Open from different client with conflicting access and share modes •Rename of the parent directory

Leases on Gluster

- Other polices:
  - Lease key based lease identification
  - Lease upgrade and downgrade
  - Timer based recall lease
  - Heuristics based lease grant

Design Overview



- Stores the lease info in memory
- Check conflict/ grant/recall lease
- Timer bound recall

Challenges involved

- Network partitions
- Integrating with other file system features like migration, healing etc.
- Multiprotocol access of file system
- Directory leases – non path based operations

Challenges involved

- **Network partitions**
- **Server/Client Reconnects:**

  - The client will replay all the lease locks it has held. Synchronization is a major problem here.
- **Server disconnect:**

  - In replica setup, leases will be served from the replica pair

  - In non replica the leases state is lost forever
- **Client disconnect:**

  - All the locks held by this client will be flushed on the server side

Challenges involved

- **Integrating with other file system features:**
- **Migration**

  - Rebalance, tiering leads the files to be migrated across servers, the lease state(fd, client id) of the file also needs to be migrated across servers.

- **Healing**

  - In case of replicated(replica or erasure coded) files when any of the replica servers is either offline or disconnected, and comes back up, the lease state needs to be healed.

  Any DFS with metadata server will not have these challenges

Challenges involved

- **Multiprotocol access of file systems:**
  - NFSV4, SMB, and Linux have different semantics of leases and converging all the standards without breaking either of the semantics.
- **Directory lease – non path based operations:**

  - In Gluster there is a concept of nameless operation, where only the UUID of the file is enough to perform any operations. In this case it is hard to identify the parent directory and break its lease.

File system Notifications

File system notifications

- A mechanism by which applications can be informed when events happen within a filesystem.

  Eg: Inotify/dnotify/fanotify etc.

- Events include
  - Modify, rename or delete of a file
  - Change in directory contents
  - Changes in metadata including extended attributes.

File-system Notifications in Gluster

- Cache-Invalidation
  - Clients/Applications are notified in case of any change in the file data or meta-data at the back-end
  - Consumers: NFS-Ganesha
- Recall-lease
  - Recall lease in case of any conflicting access on the same file.
  - Consumers: NFS-Ganesha, SMB

Design Overview

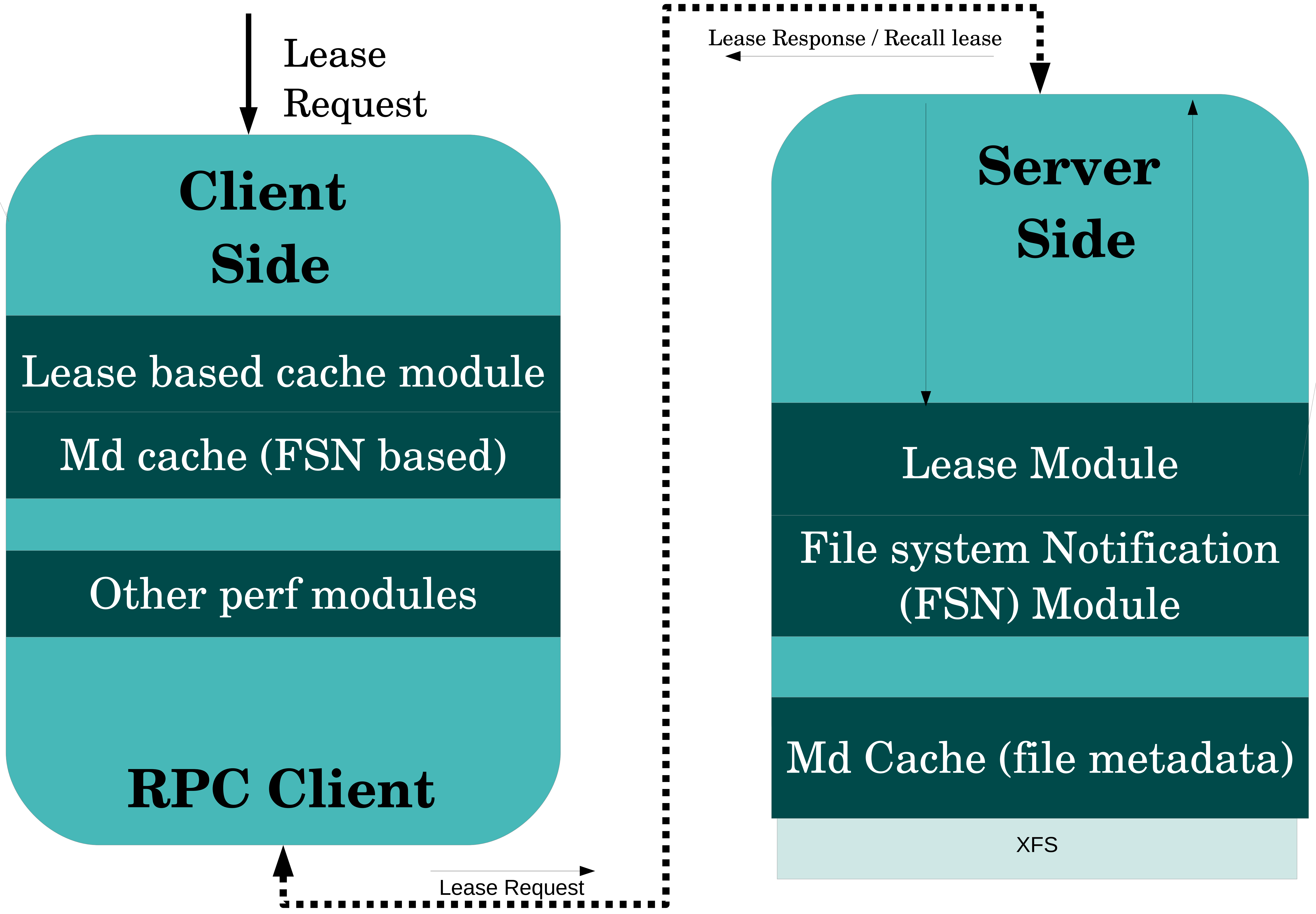
- State of the clients accessing the file is maintained by the Gluster server.
- A new callback RPC mechanism has been provided for the server to send notifications to its clients.
- Applications need to register/unregister and poll for different filesystem events.
- Notifications are sent only to the registered clients which have accessed the file with in the timeout period.

File-system Notifications

- Advantages:
  - Provide coherency for the applications' cache
- Limitations:
  - Not tolerant with network failures
- Challenges
  - Duplicate notifications need to be filtered
  - Integrating with other storage features (Data Tiering and Data Healing)

New Cache mechanism proposed

- New module to cache file data and directory content (readdirplus) based on leases
- New module to cache file metadata based on FSN and timer
- File data cache backed by disk



- New module to handle lease
- New module to maintain file access state
- New write through module to store the file metadta in memory instead of querying XFS

Caching enhancements proposed

- New Client side xlators to cache:
  - Data, metadata, readdir, readdirplus
- New Server side xlator to cache:
  - Metadata(stat, xattrs), ACLs
- File based cache, backed by disk, on the client
- Leases and cache invalidation for achieving coherency

Current state of the project

- The Server side lease, file system notification modules are WIP
- The new lease based client side module is in design phase

References

- Caching-gluster.org
- www.review.gluster.org for code
- Caching in Distributed File System – By Ke Wang

Thank You :)