CS 4414: Recitation 11

Sagar Jha

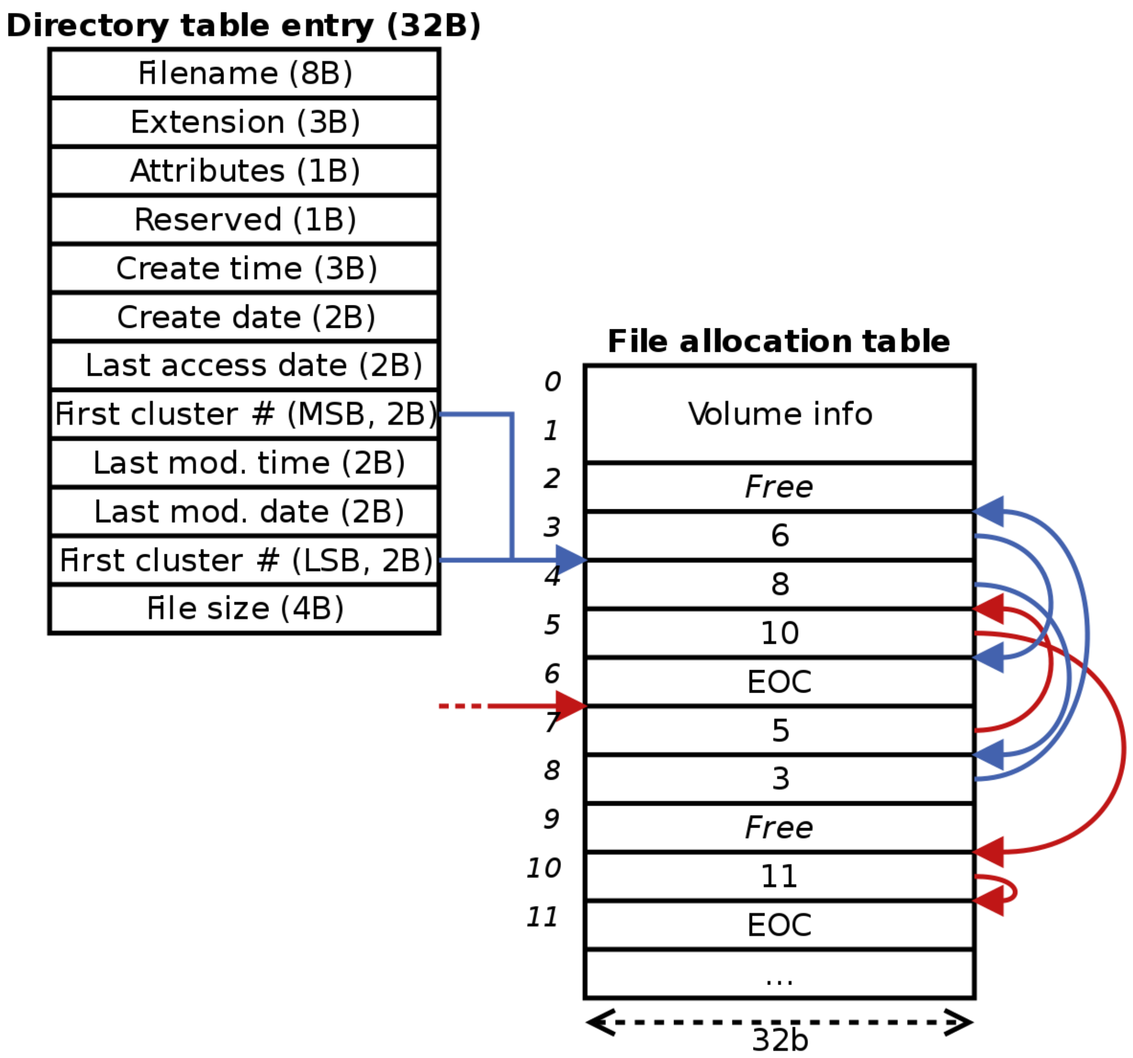
What is a filesystem?

- Manages data in files and directories
- Hierarchical structure: files in directories, directories have subdirectories
- Can store data on HDDs, SSDs or even RAM

What is a filesystem?

- File metadata: name, size, location etc.
- File data: actual contents
- Data ops: Actual file I/O – reading and writing the file
- Metadata ops: creating, deleting or renaming a file
- Blocking: A file is divided into blocks of usually 4 KB

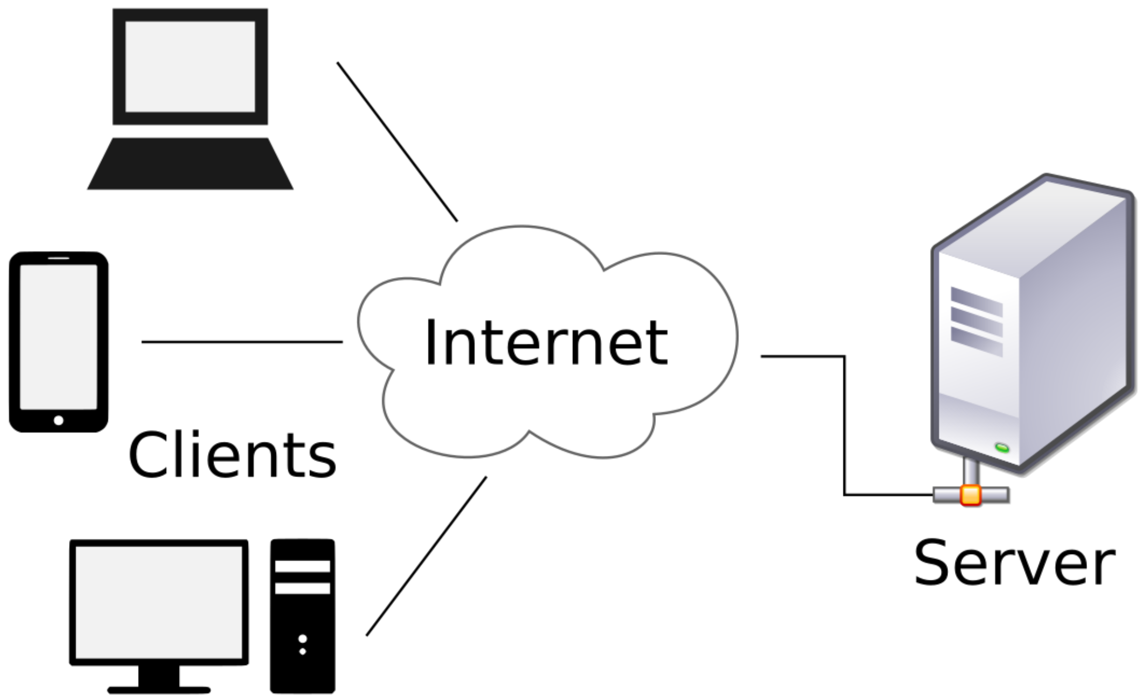
File allocation table (FAT)



Linux exiftool

```
File Name                       : forest-fire-4k-yf-1620x1080.jpg
Directory                       : .
File Size                       : 489 kB
File Modification Date/Time     : 2019:06:06 14:05:02-04:00
File Access Date/Time           : 2020:11:30 15:46:16-05:00
File Inode Change Date/Time     : 2020:08:25 20:52:37-04:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 300
Y Resolution                    : 300
Image Width                     : 1620
Image Height                    : 1080
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
```

Distributed file system (DFS)



- Filesystem that is accessed over the network from multiple clients
- A remote server supports the same filesystem interfaces

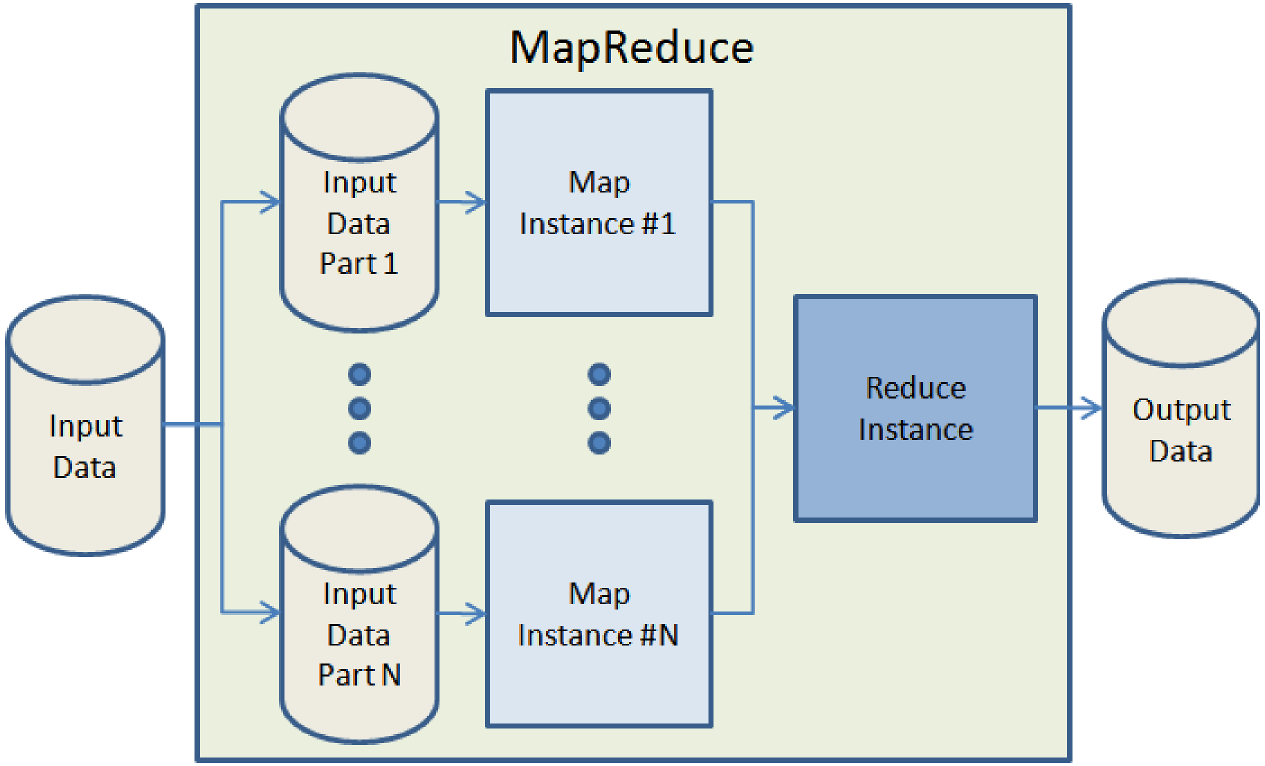
Distributed file system (DFS)

Main goal: Provide the abstraction of a filesystem but one that is accessible from multiple clients simultaneously

- Access transparency: Same API for the clients as if they were accessing a local filesystem
- Support for concurrency: Clients see a consistent view of the filesystem when multiple clients are accessing it simultaneously
- Fault-tolerance and scalability

Applications and advantages of a DFS

Applications and advantages of a DFS



- Batch processing of big data
- Processing big data using MapReduce
- Large scale ML
- Don't want to copy or move too much data around

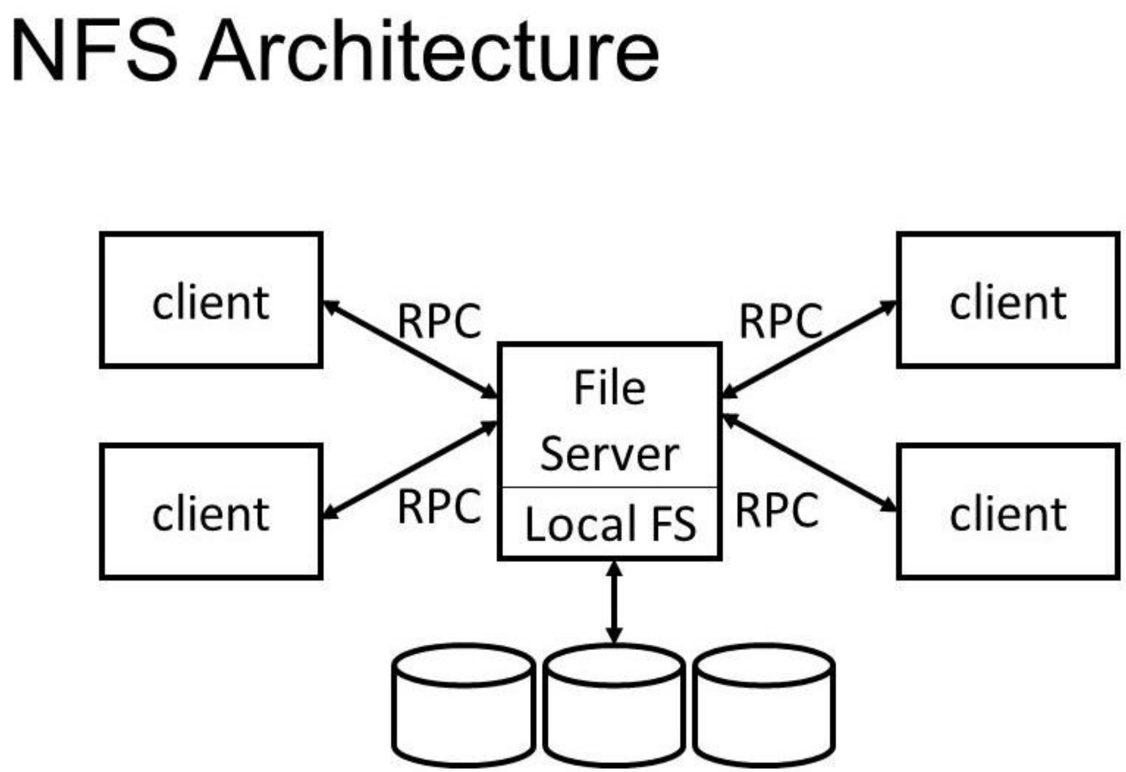
Applications and advantages of a DFS

- Multiple users can share files
- Can access files from multiple devices

Applications and advantages of a DFS

- Elasticity – Can scale to petabytes or more storage on demand
- Ease of access – Data can be accessed across multiple devices
- Centralized administration – makes it easier to offer consistency guarantees in a distributed setting
- Persistent way to store configuration files

Network file system (NFS)



- A simple implementation that combines local filesystem on multiple server nodes
- A client makes a request over the network that is fulfilled by exactly one server node

NFS development

- Originally developed by Sun Microsystems in 1984
- NFSv2: Stateless server with locking, UDP for sending requests (1989)
- NFSv3: 64-bit file sizes and offsets, asynchronous writes support, TCP for transport (1995)
- NFSv4: Security improvements, stateful protocol (2000)

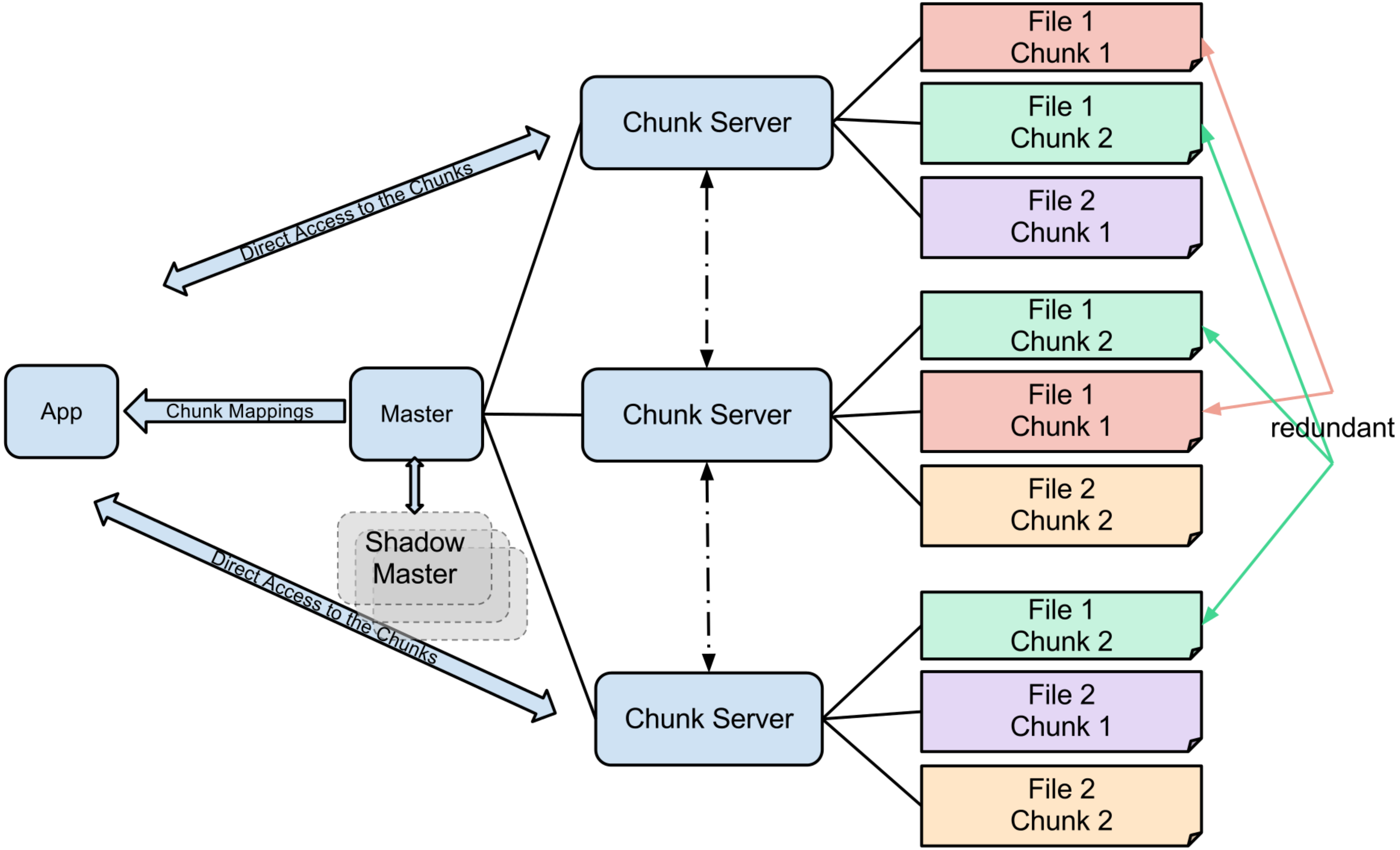
Limitations of NFS

- Synchronous I/O: All read/write operations finish only when the data has been written to disk on the server side
  - write to nonvolatile RAM and asynchronously later to disk
  - batching writes: gather multiple write requests from different clients to amortize I/O costs
- Centralized design: Poor performance for large files as read/write is not parallelizable
- No support for consistency with multiple clients

Object-based file systems

- A file is stored as a collection of distributed, variable-sized objects instead of fixed-sized blocks
- Object storage servers store the objects, service read/write requests
- A separate metadata server (MDS) performs metadata operations (open, rename)

Google File System (GFS)



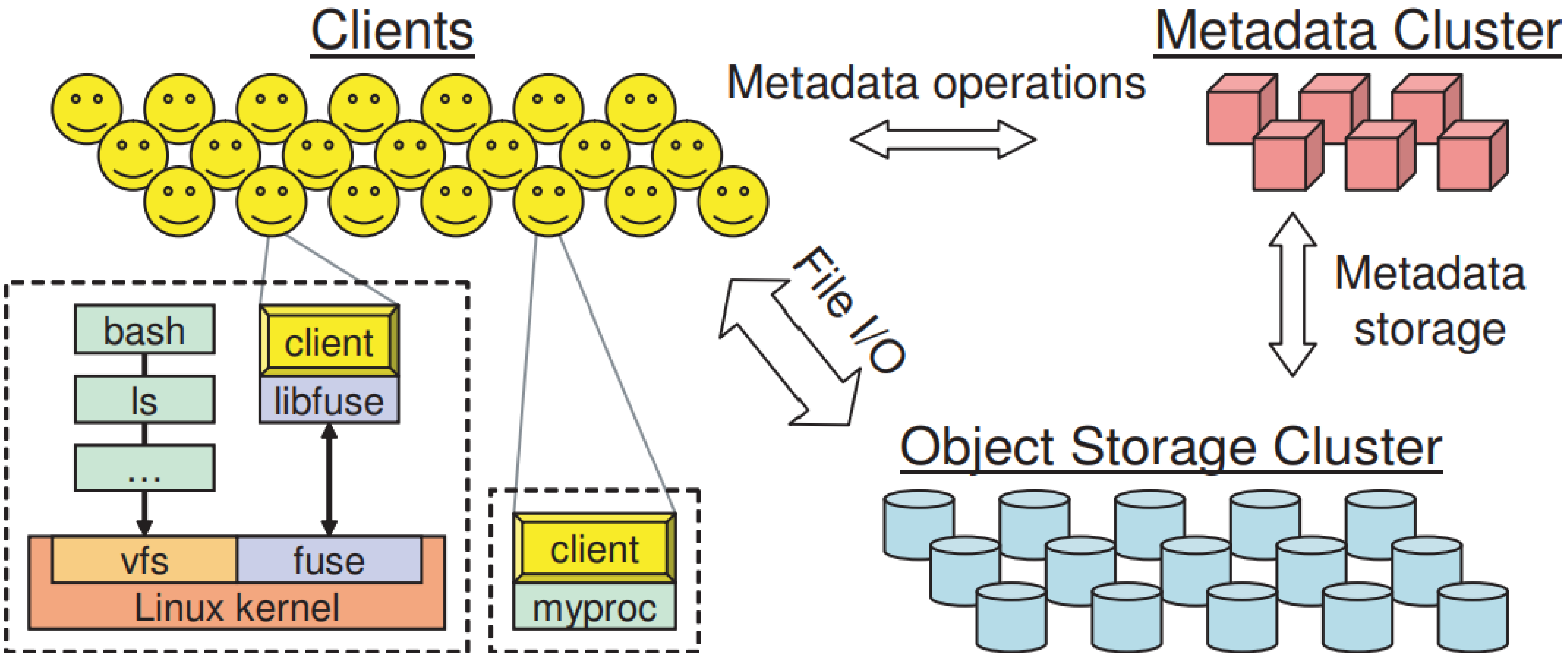
What is Ceph?

- A distributed file system built upon object storage devices
- Written in C++

Ceph Goals

- Performance: Read/write throughput, high throughput for metadata operations
- Reliability: Resistant to node failures, adapts with shifting workloads
- Scalability: High performance with many clients and large data sizes

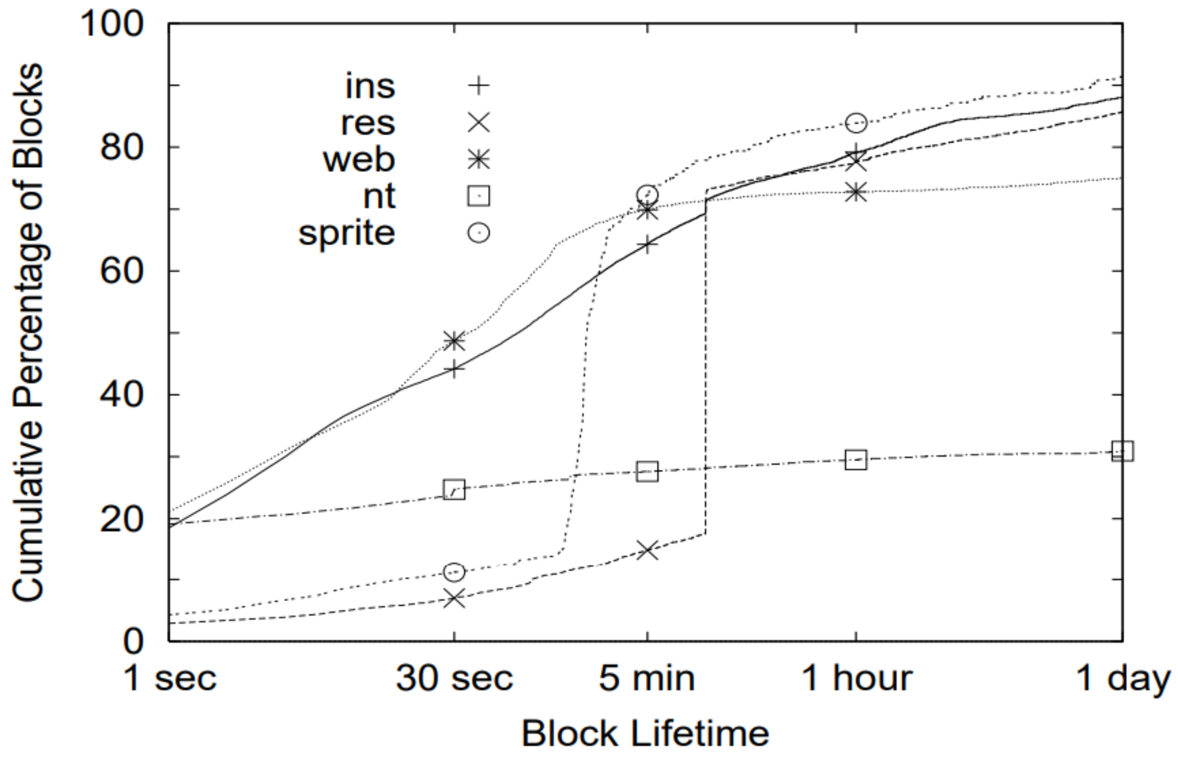
Ceph architecture



Ceph’s main insight

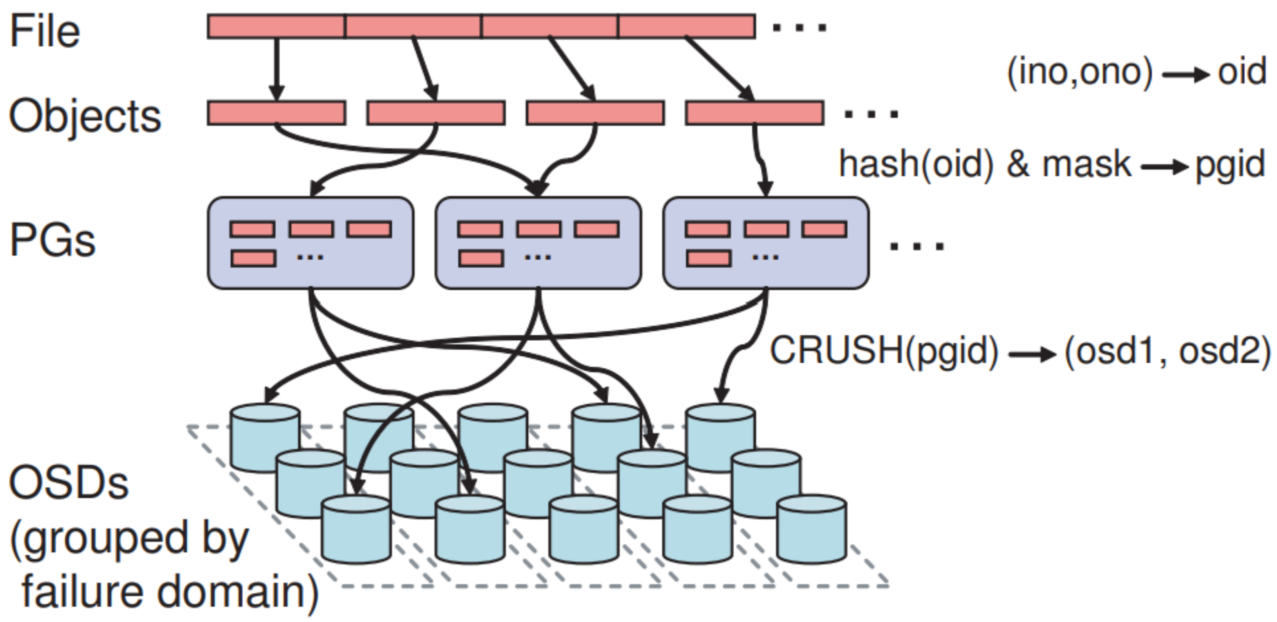
- Object-based file systems are bottlenecked for metadata operations
- Metadata storage needs to be distributed as well
- Delegate intelligence to object storage devices (OSDs) to minimize the number of metadata operations and improve parallelism

How important are metadata operations?



- Filesystem metadata operations can make up to 50% of an application workload
- In UNIX systems, most blocks die within an hour

Optimization 1: Data distribution with CRUSH



- Controlled replication under scalable hashing
- File divided deterministically into objects using generating functions
- Object locations can be independently calculated, no need to contact the metadata server

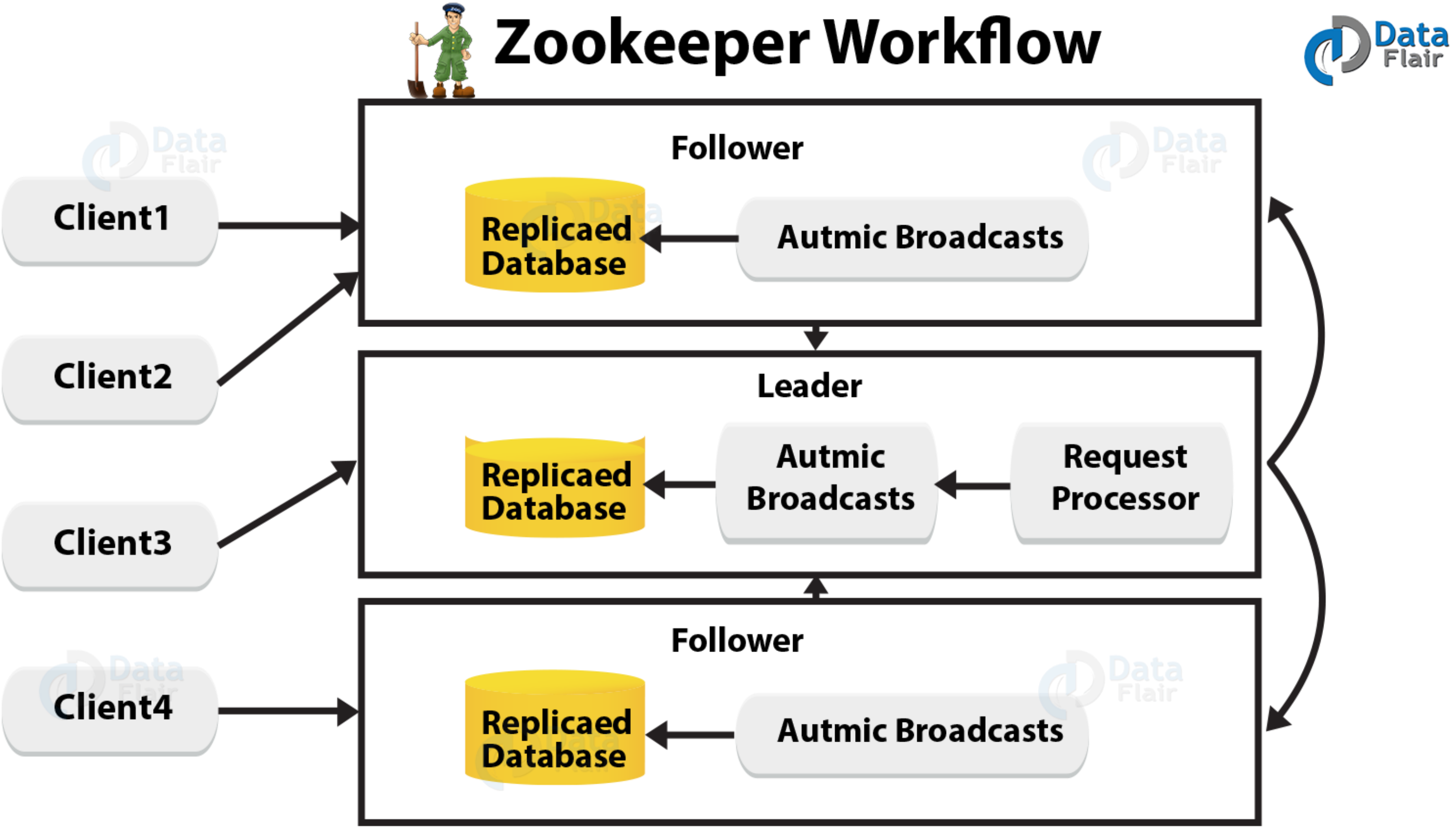
Optimization 2: Dynamic distributed metadata management

- Novel metadata cluster architecture based on Dynamic Subtree Partitioning
- Intelligent distribution of metadata workload among hundreds of metadata servers
- Dynamic load distribution based on access patterns

Optimization 3: Distributed reliability and high availability protocols

- Focus on effectively utilizing available devices at any point in time
- Replication guarantees across device failures
- Efficient data migration, replication, failure detection and recovery protocols

Zookeeper



Zookeeper Workflow
Follower
Replicaed Database
Autmic Broadcasts
Leader
Replicaed Database
Autmic Broadcasts
Request Processor
Follower
Replicaed Database
Autmic Broadcasts
Client1
Client2
Client3
Client4

References

- Ceph: A Scalable, High-Performance Distributed File System
- NFS Version 3 Design and Implementation
- Why NFS Sucks
- A Comparison of File System Workloads
- The Google File System