CE 528 Cloud Computing

Lecture 10: Cloud Storage
Spring 2026

Prof. Yigong Hu

BOSTON UNIVERSITY

Slides courtesy of Ata Turk

Ceph and Flat Data Center Storage

Cloud Storage

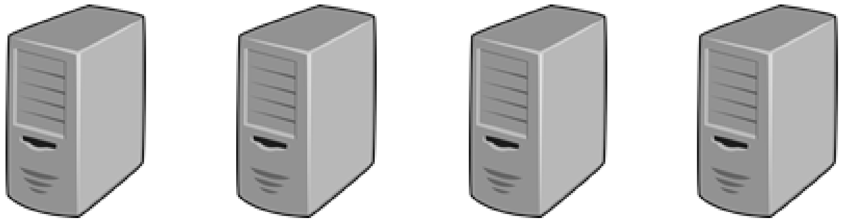
There are two approaches to find server responsible for data/key/…

- A centralized meta data server - e.g., GFS, various google platforms, Alibaba…
- Hashing - e.g., the two file systems we will talk about today, services designed Amazon…

What are the tradeoffs?

Hashing

hashValue = hashFunction(key)
serverIndex = hashValue % numberOfServers



Server 0	Server 1	Server 2	Server 3
key 0	key 1	key 2	key 3
key 4	key 5	key 6	key 7

What happens when we add or lose a server?

Adding Server 4

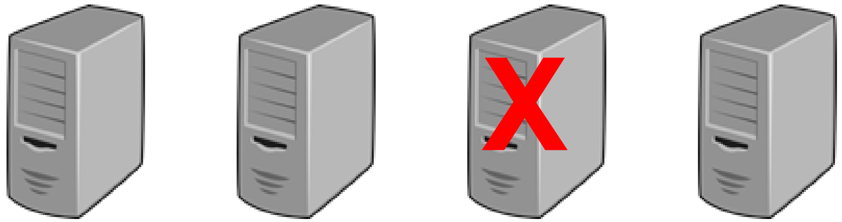
hashValue = hashFunction(key)

serverIndex = hashValue % numberOfServers

Server 0	Server 1	Server 2	Server 3	Server 4
key 0	key 1	key 2	key 3	key 4
key 5	key 6	key 7		

Server 2 fails

hashValue = hashFunction(key)
serverIndex = hashValue % numberOfServers

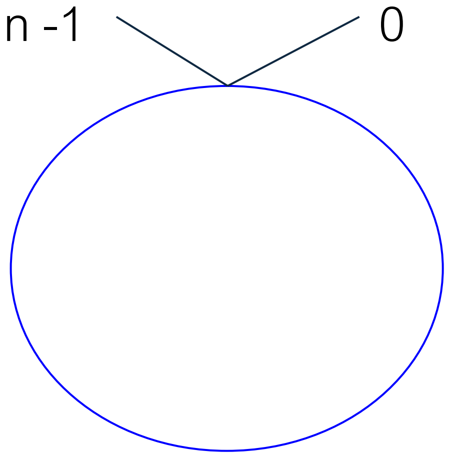


Server 0	Server 1	Server 2	Server 3
key 0	key 1		key 2
key 3	key 4		key 5
key 6	key 7		

What We Need?

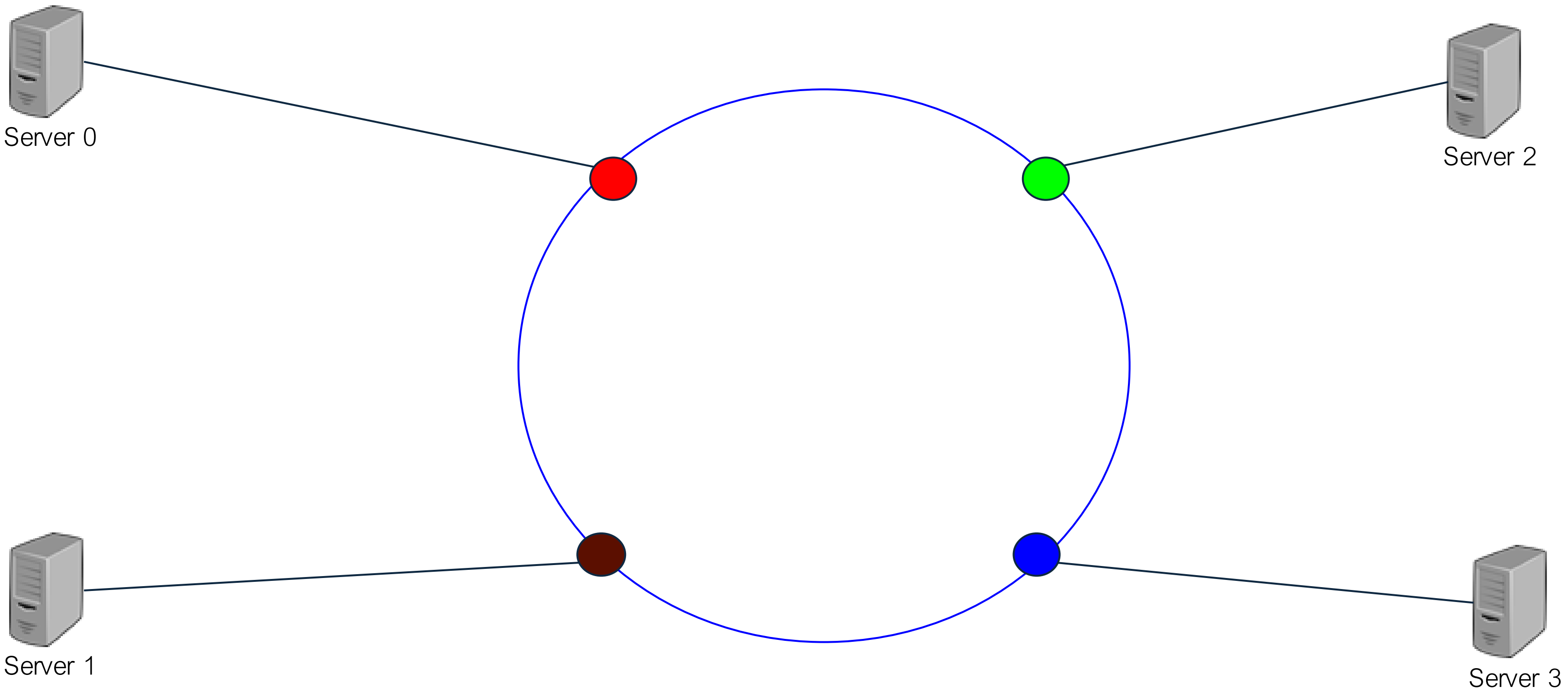
Technique that tries to minimize re-balancing when the number of buckets you hash across changes

Simplest scheme consistent hashing, represents hash Space as a ring



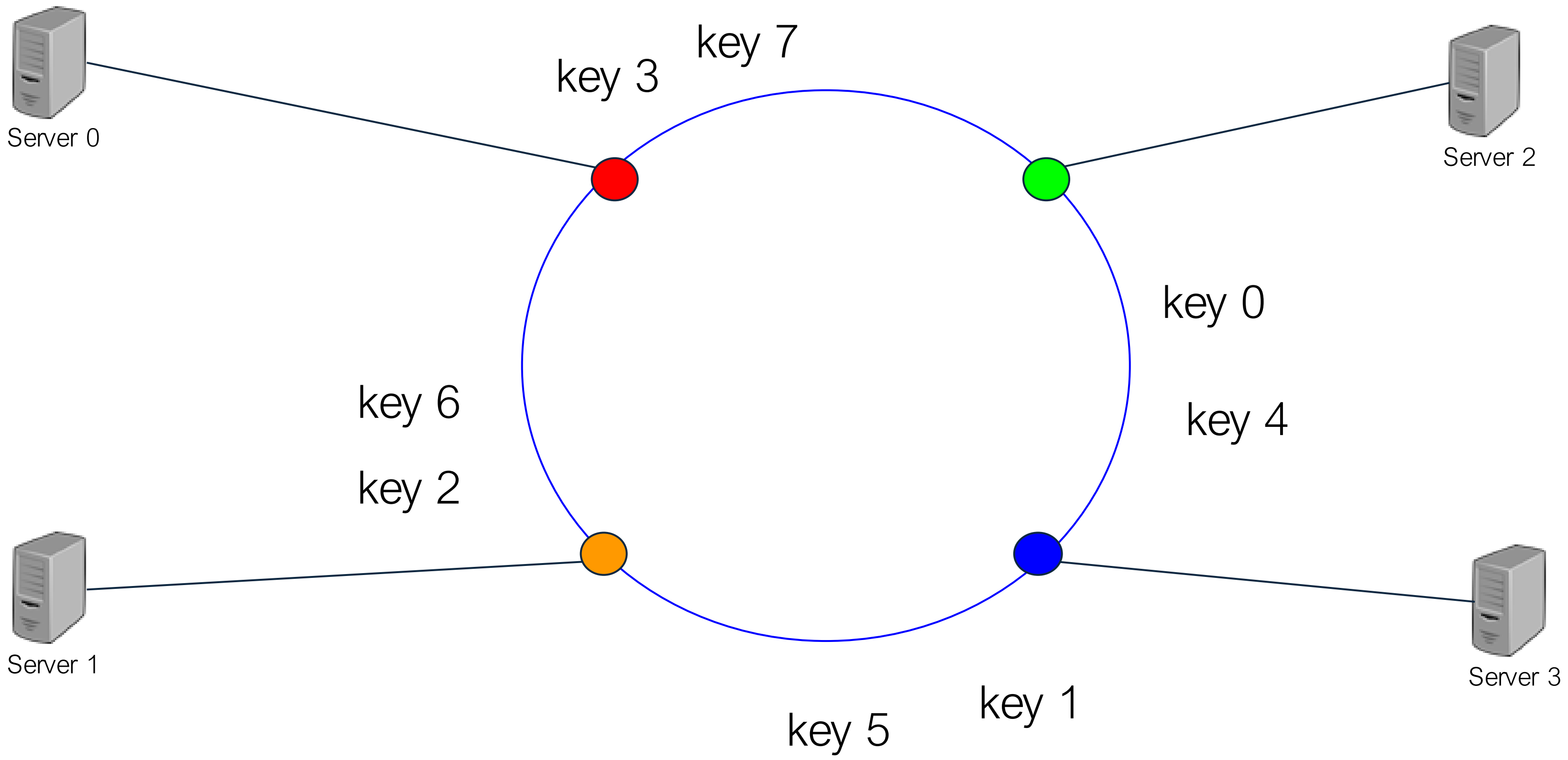
Hash Servers around Rings

based on IP address

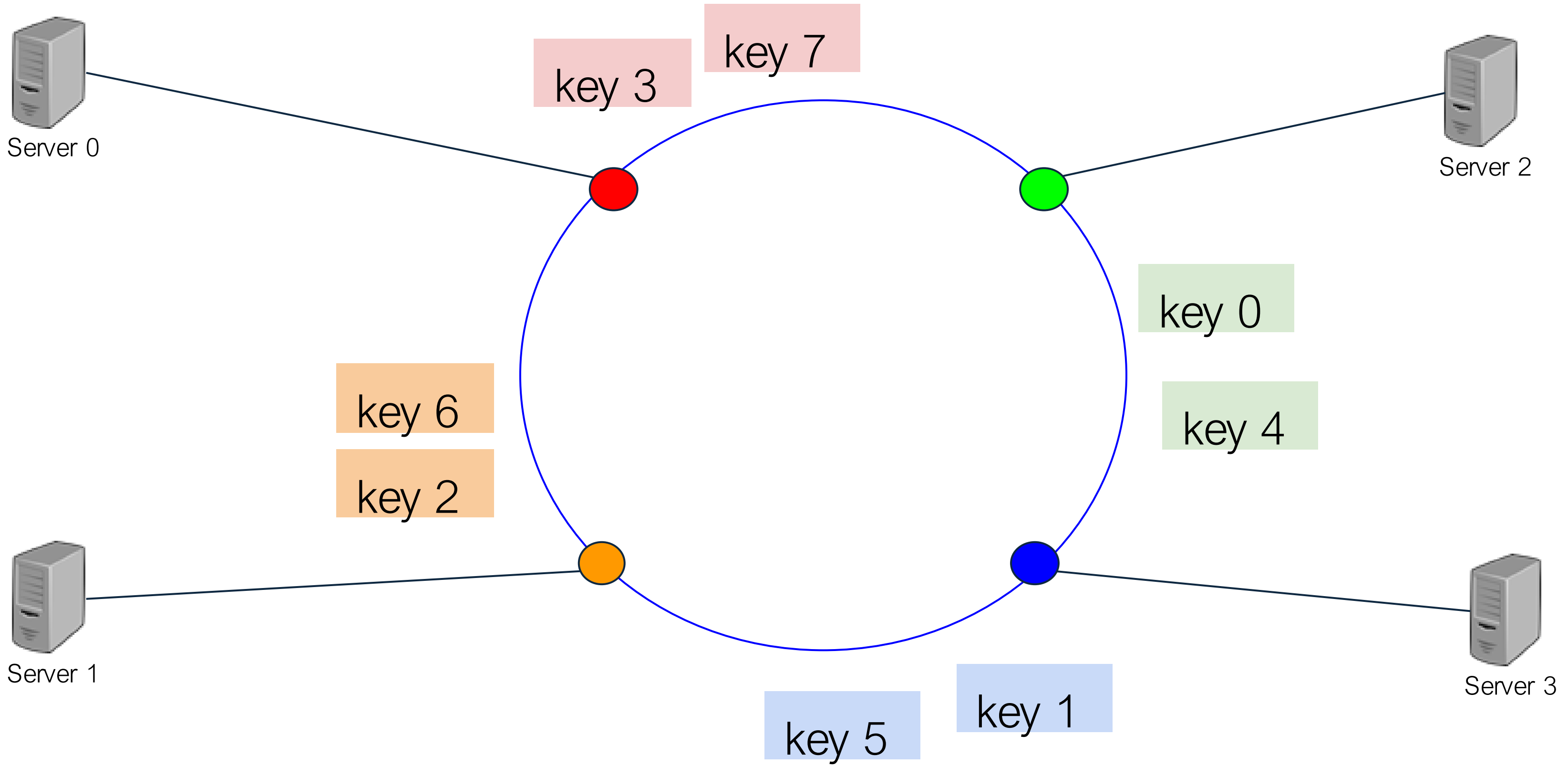


Hash Servers around Rings

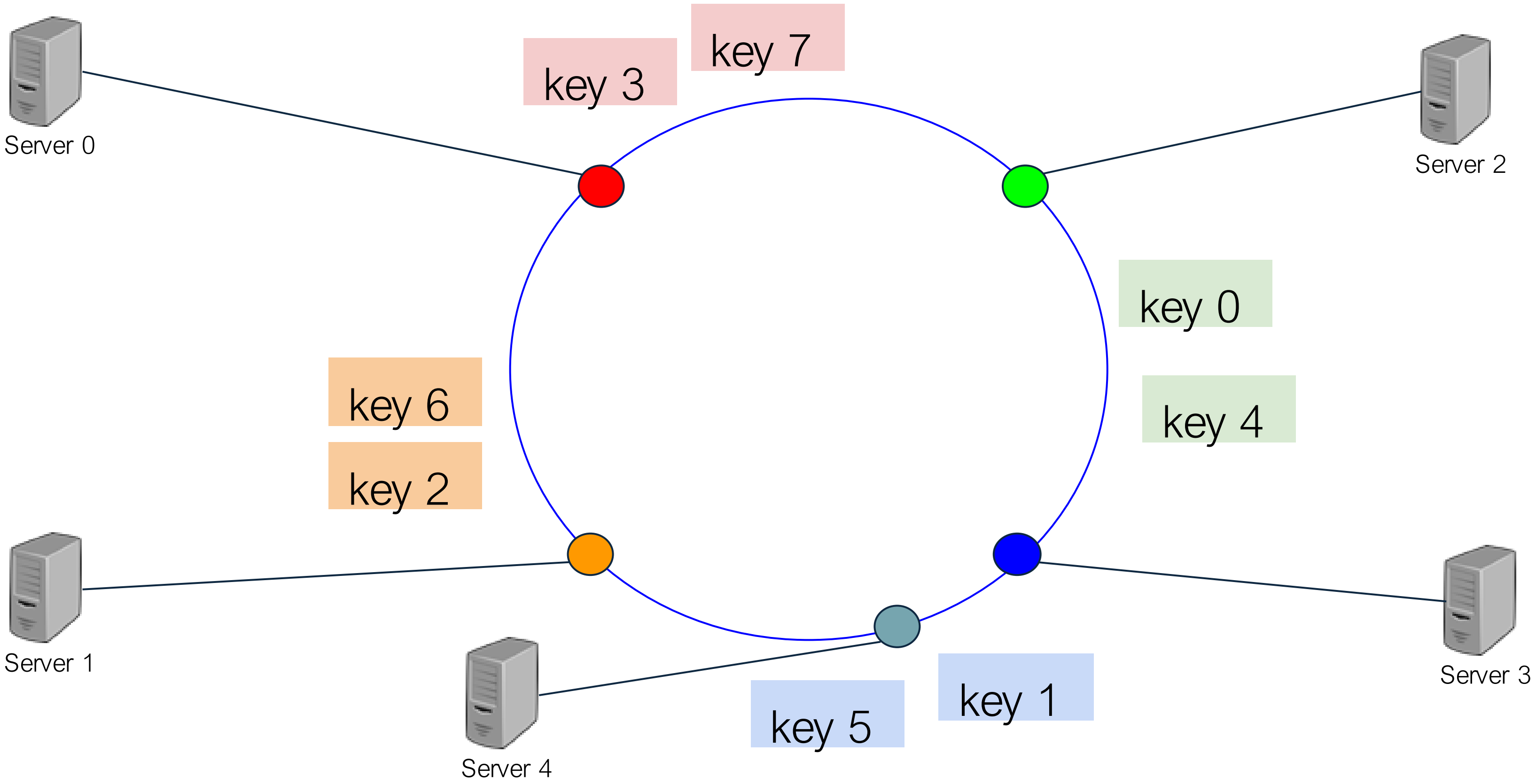
based on IP address



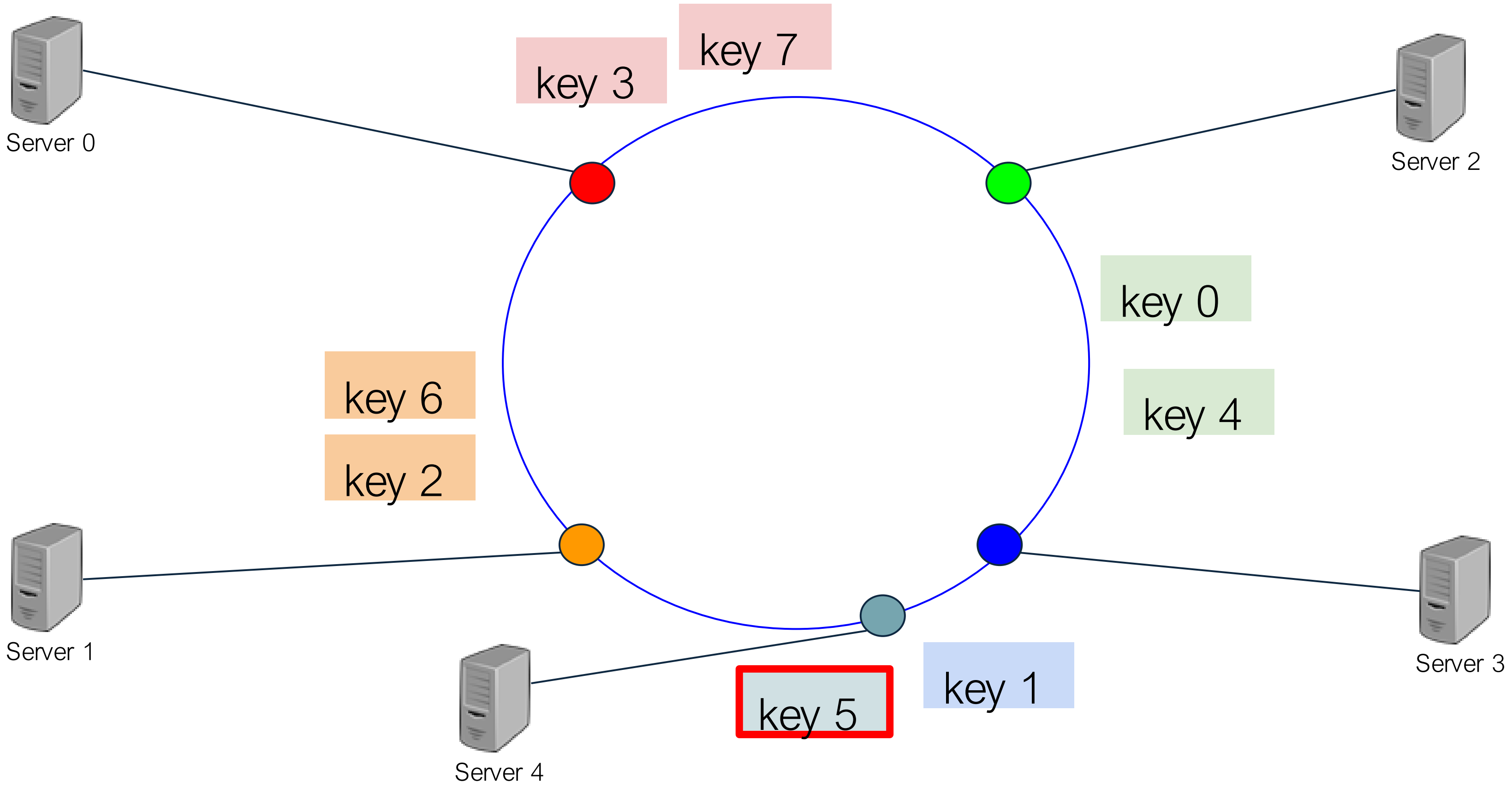
Server Responsible for Nearby Keys



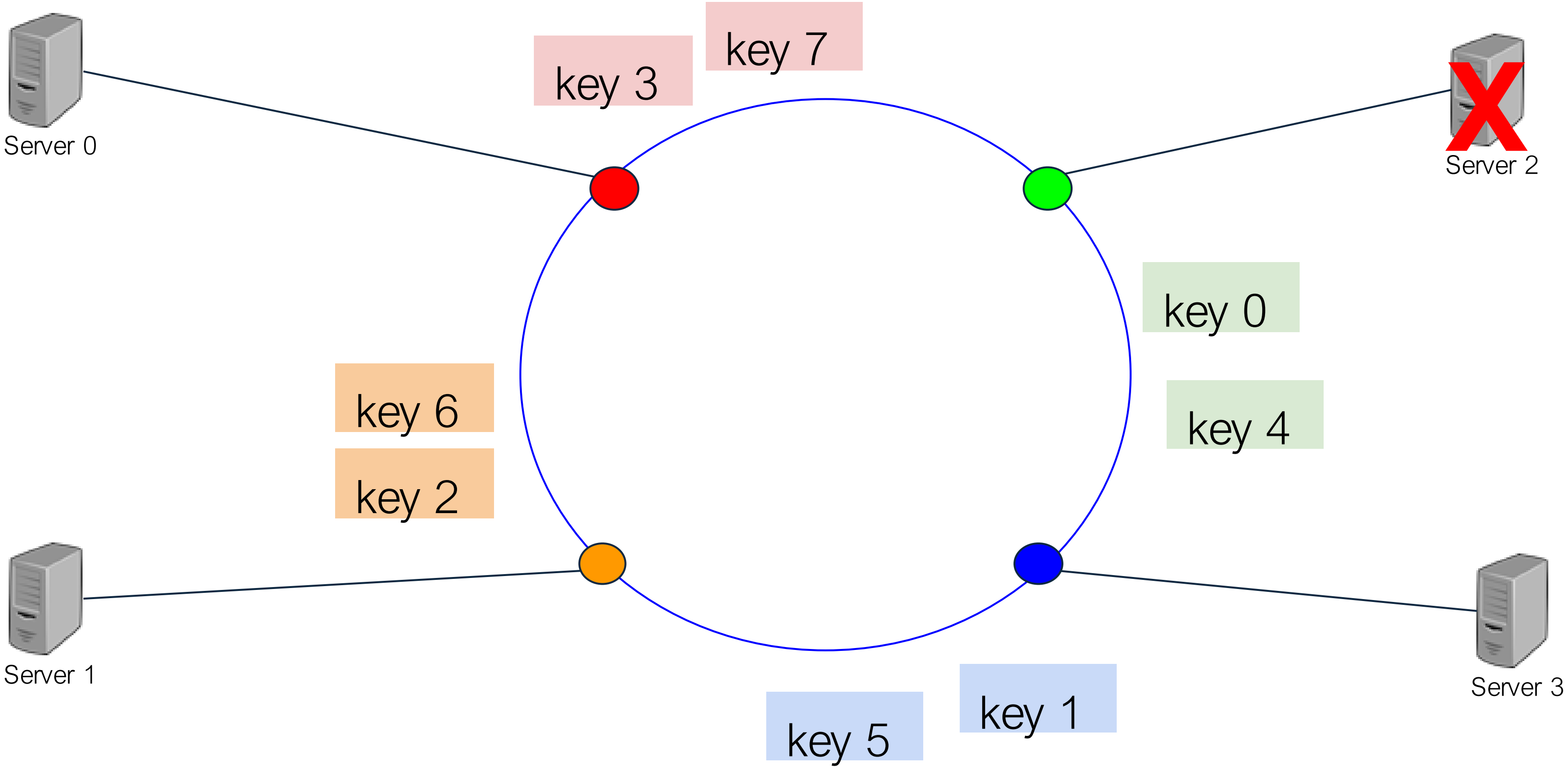
Servers Added



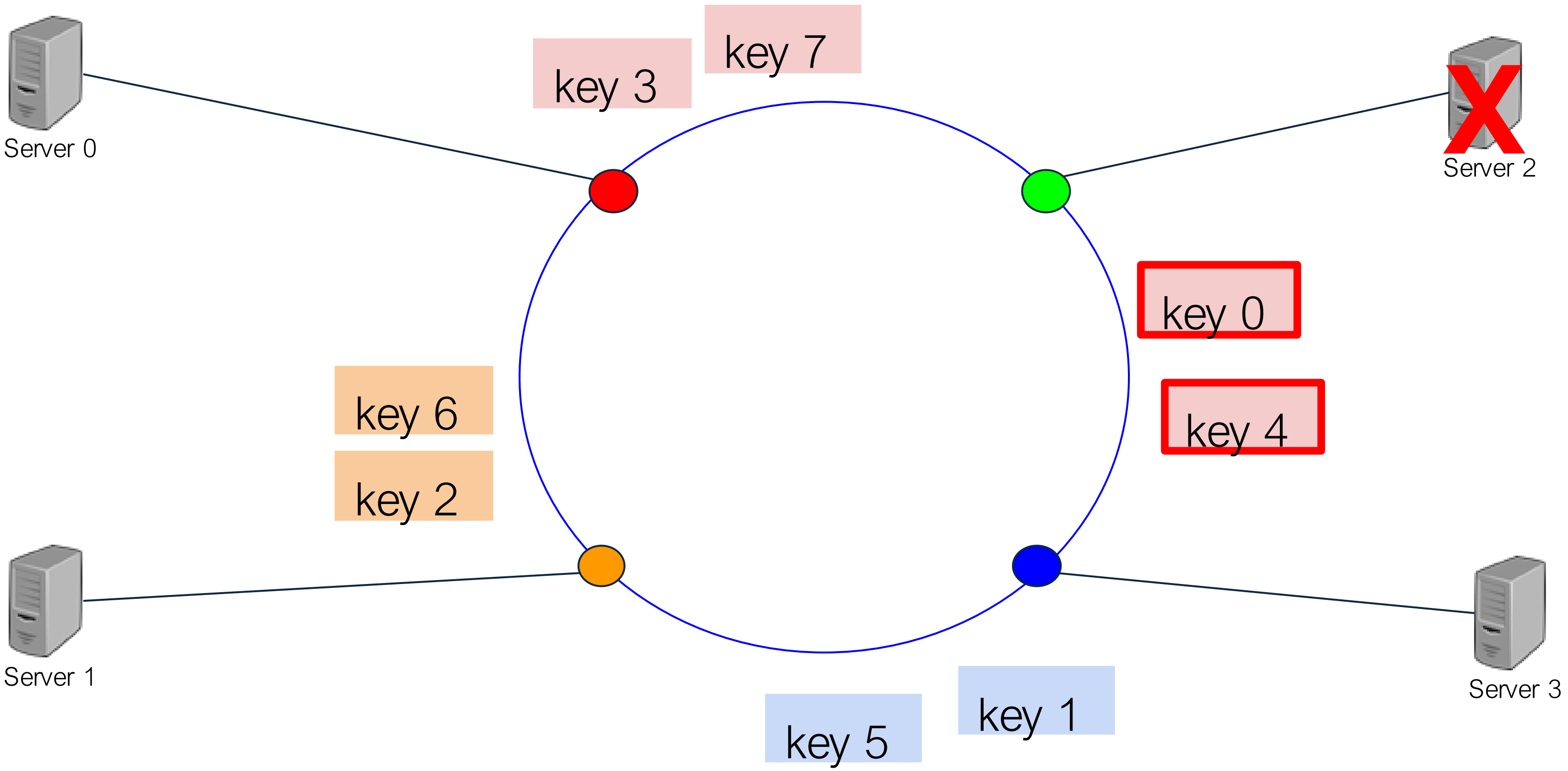
Move Just Keys in Range



Server Dies


Server 0
key 3
key 7
X
Server 2
key 0
key 4
key 6
key 2
Server 1
Server 3
key 5
key 1

Server Dies


Server 0
key 3
key 7
Server 2
key 0
key 4
key 6
key 2
Server 1
Server 3
key 5
key 1

Overview

Each 'host' hashed to find location on ring

To locate who is responsible for 'x' hash on ring;

- previous (or next) host is responsible for handling request

Advantages:

- When something changes minimize movement
- Avoids meta data server; only need hash function and lists of hosts to find server responsible for 'id'

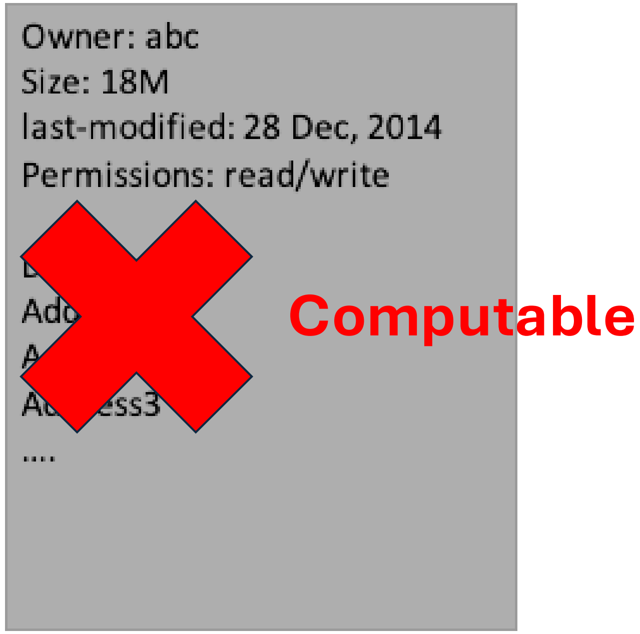
Ceph and FDS are two file systems designed based on Hashing that minimizes movement

***Ceph*: A Scalable, High-Performance Distributed File System**

Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn

OSDI 2006

File System Metadata



Owner: abc
Size: 18M
last-modified: 28 Dec, 2014
Permissions: read/write

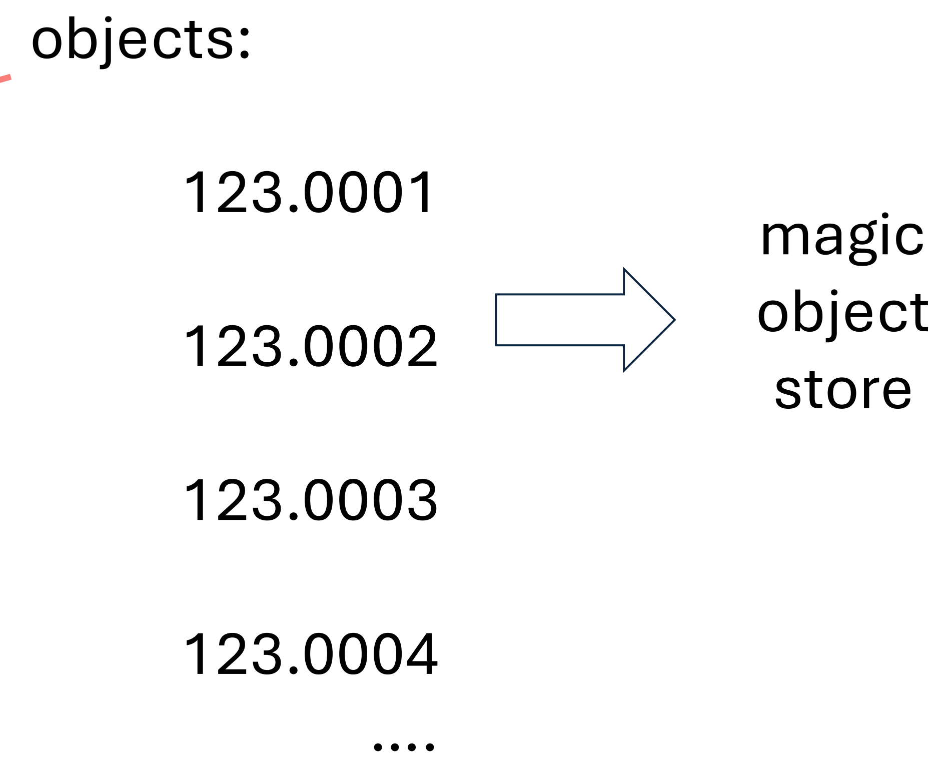
D
Add
A
Ad ess3
....

Computable

How It Works

directory

name	inode#	inode
file.txt	123	…
data.dat	456	…
other.xls	768	…



objects:

123.0001

123.0002

123.0003

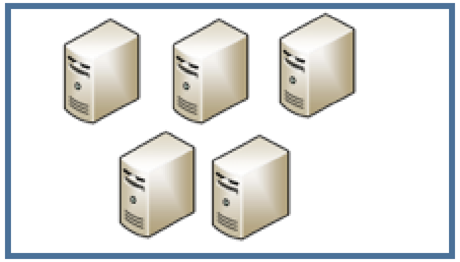
123.0004

….

magic
object
store

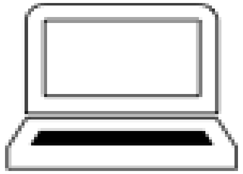
Ceph’s Key Components

- ***A cluster of metadata servers which has dynamic load balancing feature and fault tolerant***

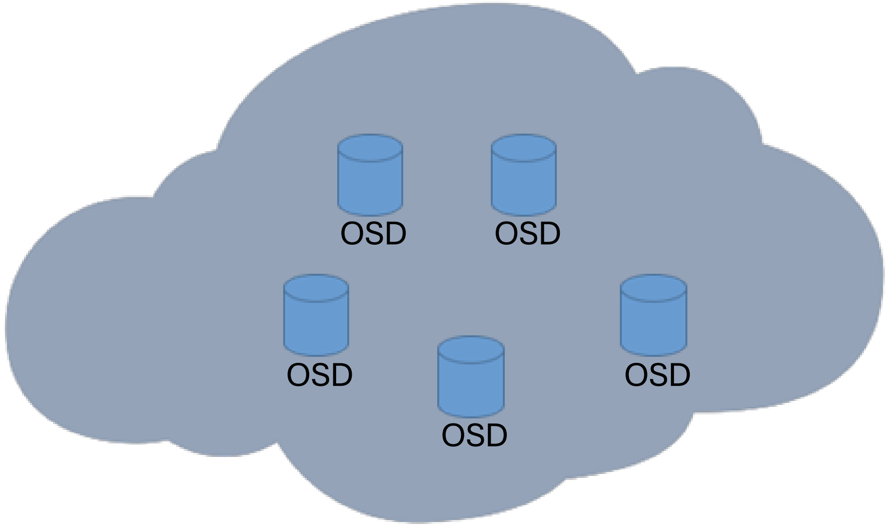


MDS Cluster

- ***Intelligent Storage device that handle data replication, failure detection and recovery***



Client



Object Storage Cluster

- ***A program that runs in userspace and exposes file system interface to the host***

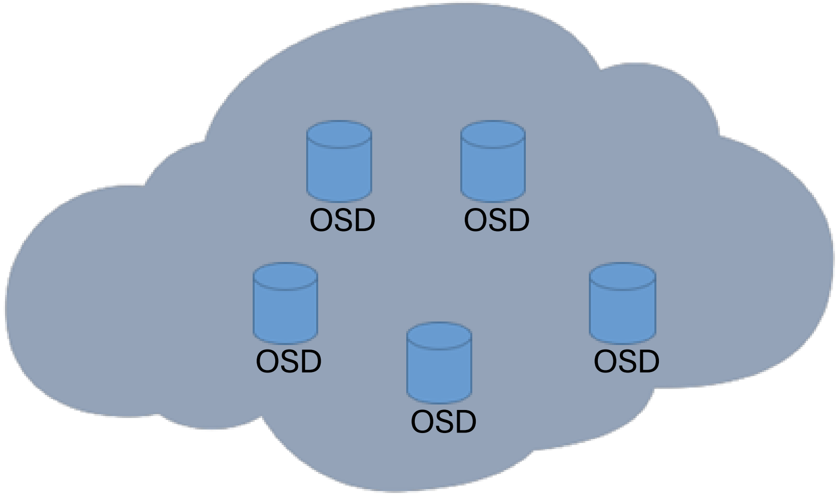
What’s Object Storage?

sort of a file (named, range read/write)

replacing block-based remote disks

Lustre, Parnassus, Ceph, …

Not to be confused with object storage (S3)



Object Storage Cluster

Client – a few Comments

User space program

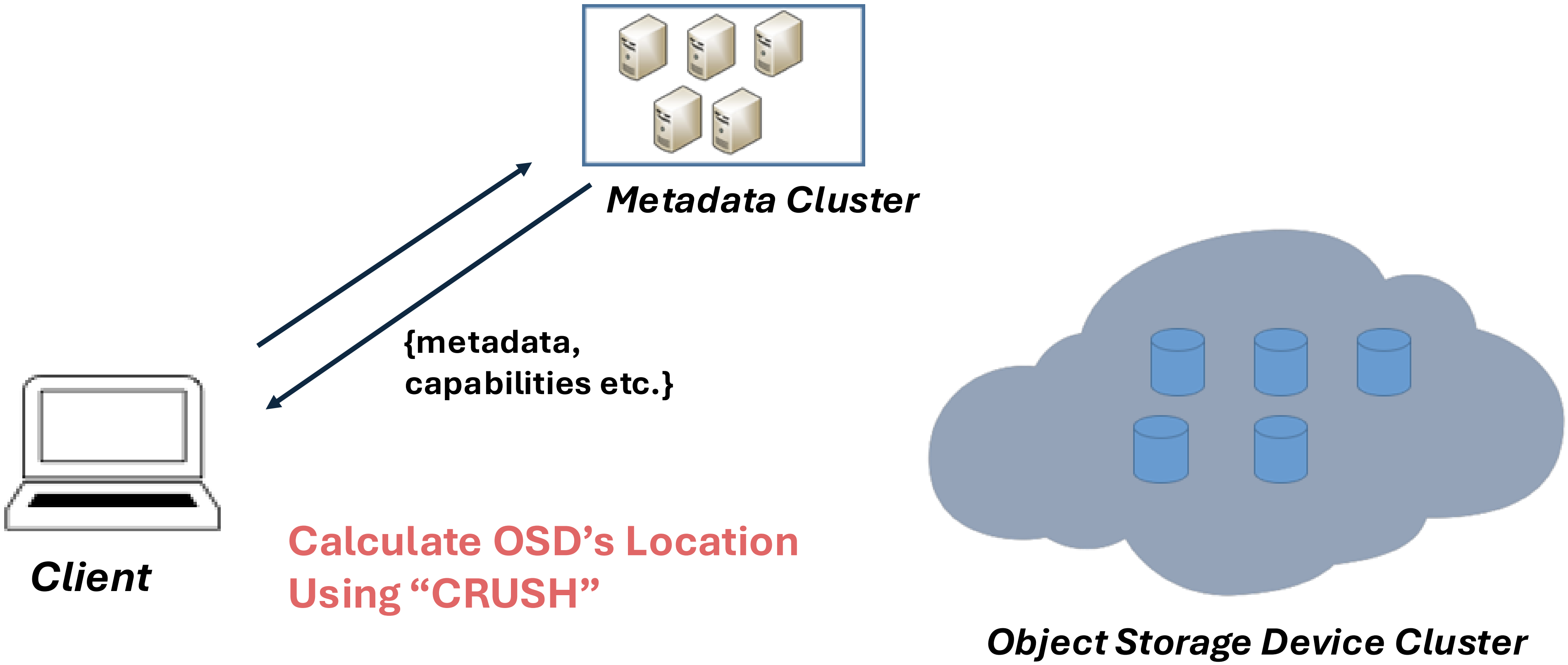
Near POSIX Interface

- Open, Read, Write, Close, …

Caching and buffering

- If one client, or multiple readers, data cached
- Otherwise, all operations synchronous to OSD
- Supports various HPC flags to enable application control

Ceph’s Key Components



Metadata Cluster

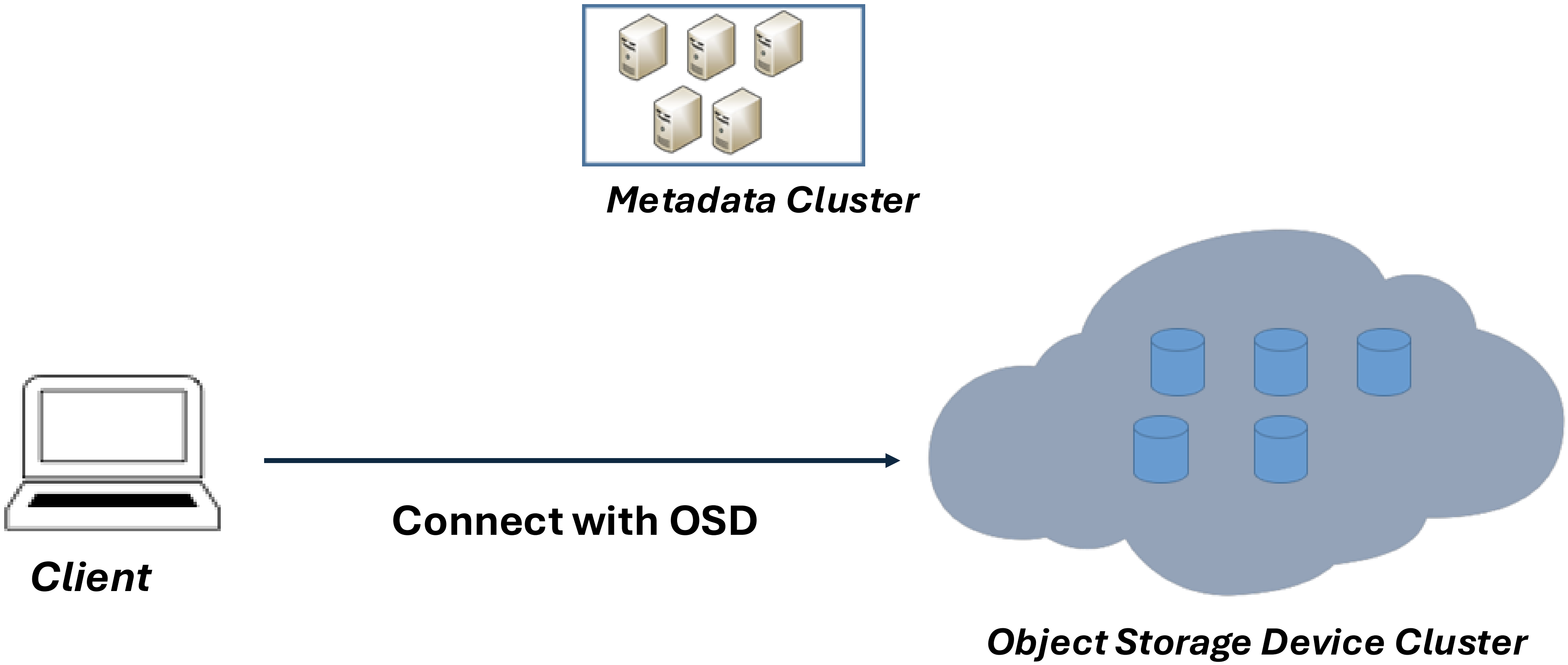
{metadata, capabilities etc.}

Client

Calculate OSD’s Location Using “CRUSH”

Object Storage Device Cluster

Ceph’s Key Components



Metadata Cluster

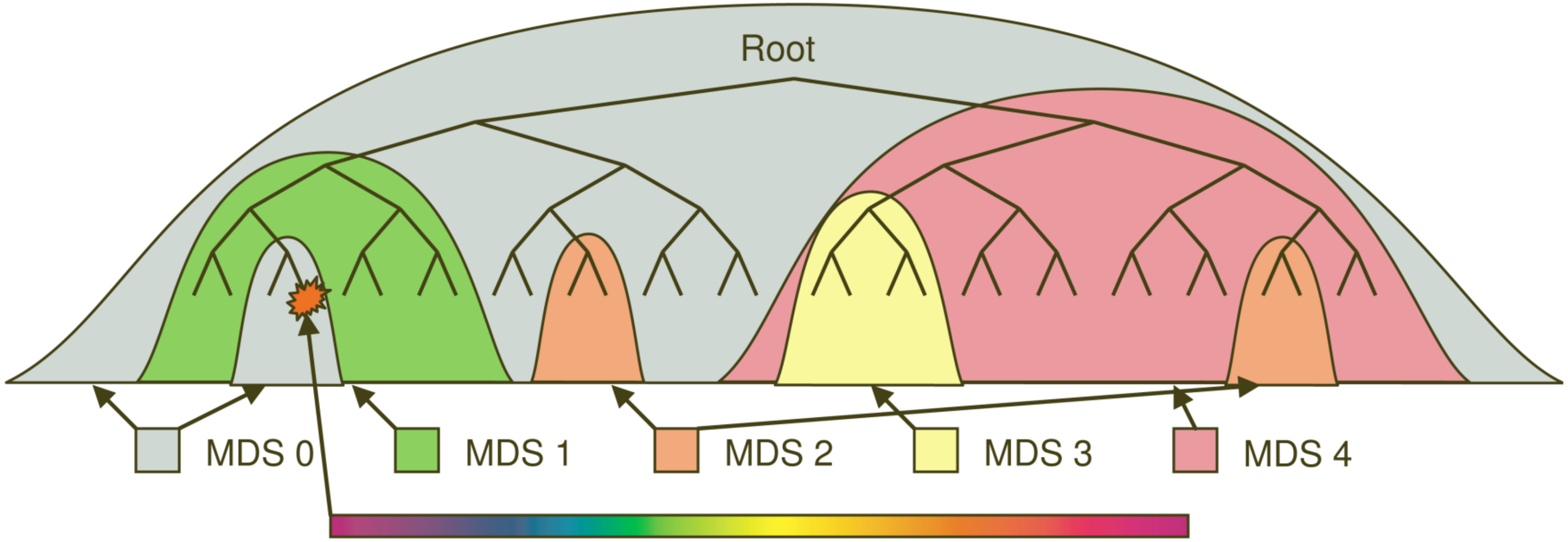
Client

Connect with OSD

Object Storage Device Cluster

Meta Data Servers

Adaptively distributes cached metadata across a set of MDS w.r.t popularity



Busy directory hashed across many MDS's

Object Storage Device (OSD)

Commodity server

- with CPU, memory and storage

Can make the replication and recovery decision

Large blocks of data (e.g., 4MB objects)

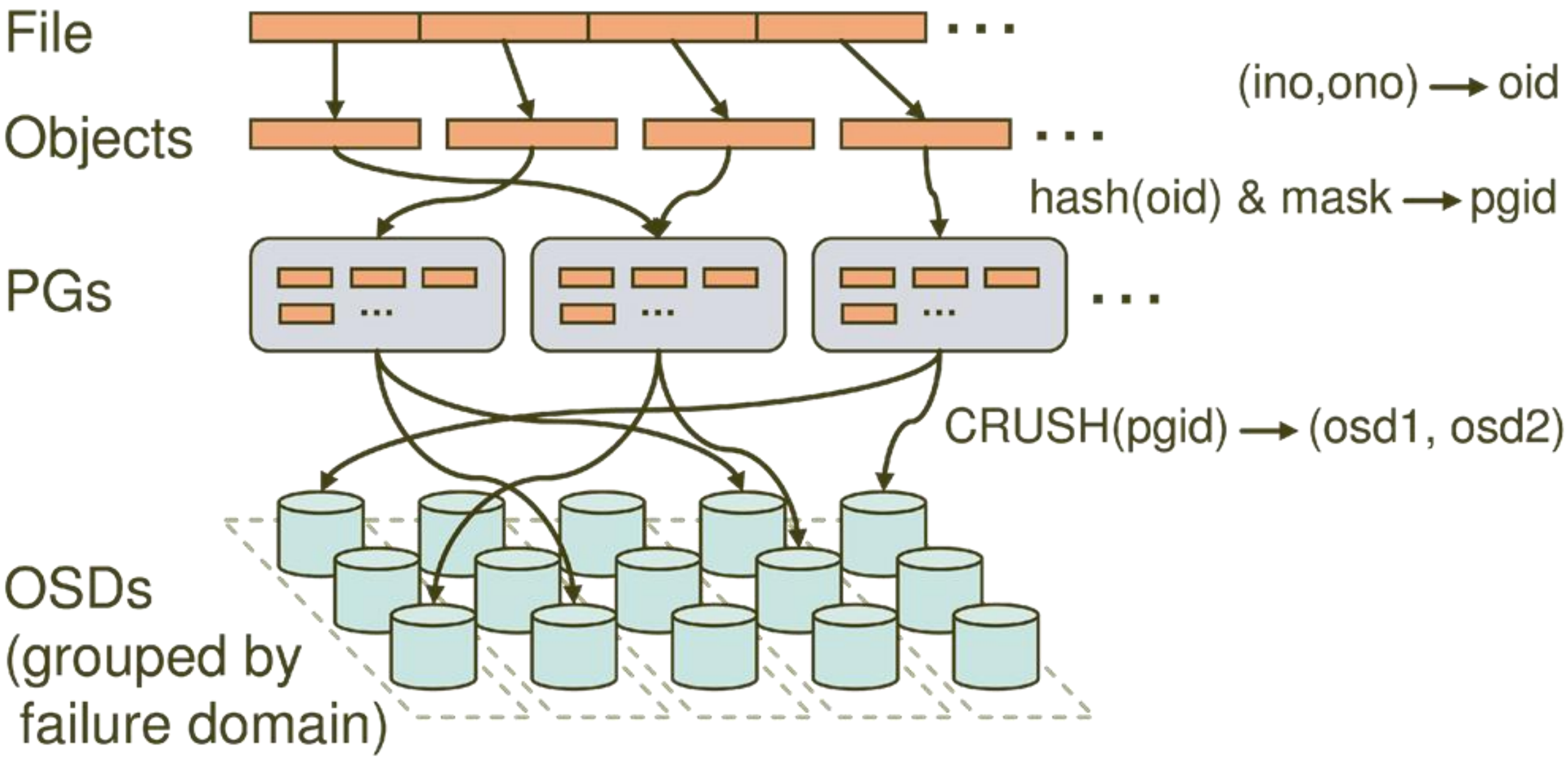
Distributed Object Storage

Files are split across objects

Objects are members of placement groups

Placement groups are distributed across OSDs.

Example


File
. . .
(ino,ono) → oid
Objects
. . .
hash(oid) & mask → pgid
PGs
...
...
...
. . .
CRUSH(pgid) → (osd1, osd2)
OSDs
(grouped by
failure domain)

Placement

Two levels of hashing:

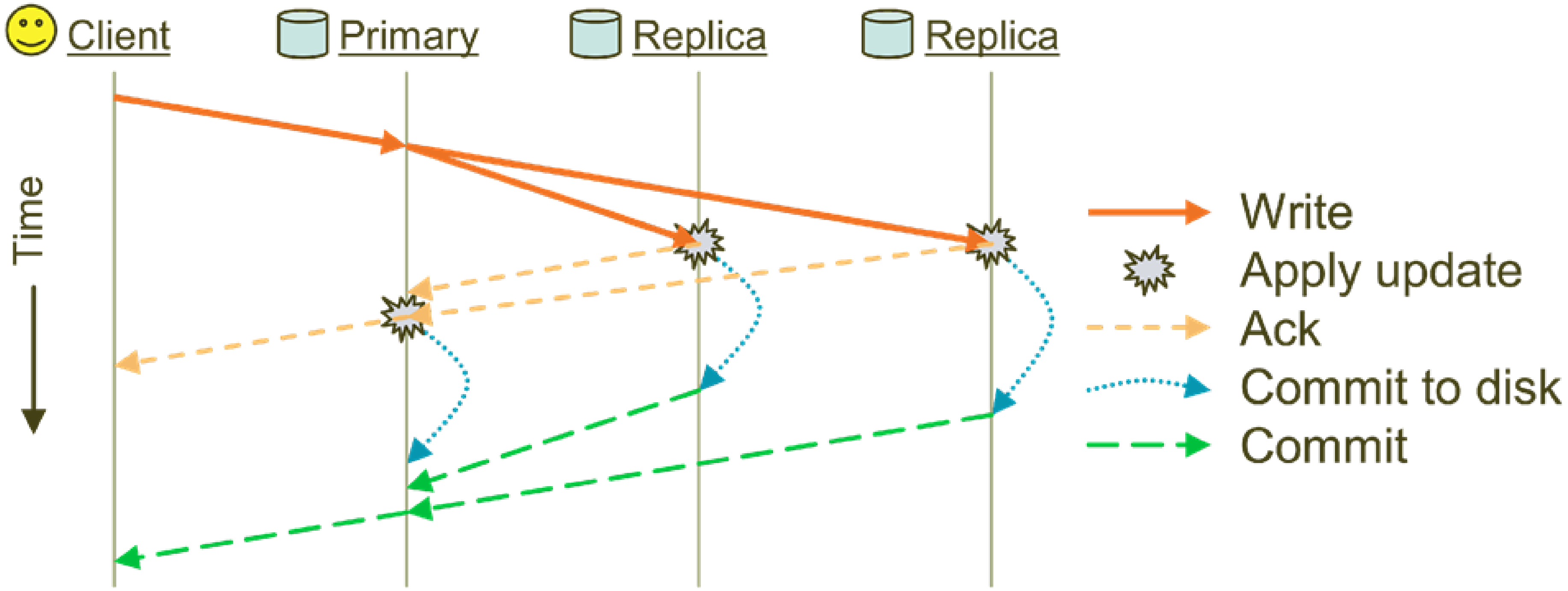
- object ID -> placement group (PG)
- PG -> OSD using CRUSH

Client requires hash algorithm, PG, OSD cluster map

- a compact, hierarchical description of the devices comprising the storage cluster.

Around 100 PG per OSD; when an OSD fails, different OSDs take over all the PG

Replication



Conclusion - Ceph

Demonstrated key value of consistent hashing for locating data

Only used meta-data for implementing traditional file system semantics

Achieved excellent performance and scalability

Flat Datacenter Storage

Edmund B. Nightingale, Jeremy Elson, Jinliang Fan, Owen Hofmann*, Jon Howell, and Yutaka Suzue

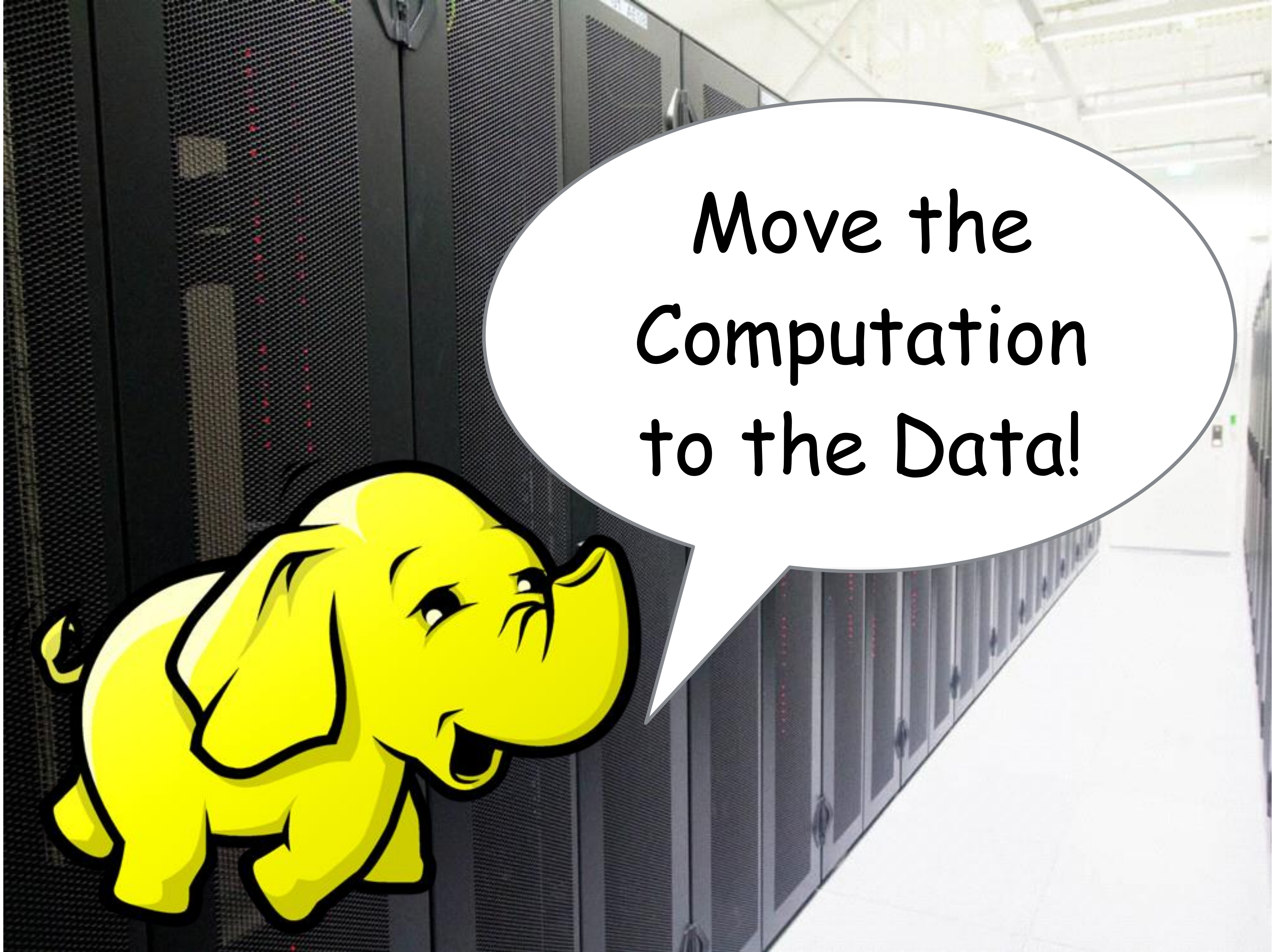
Microsoft Research

OSDI 2012

Motivation

Imagine world with "flat" data storage

- Simple easy to program



Move the Computation to the Data!

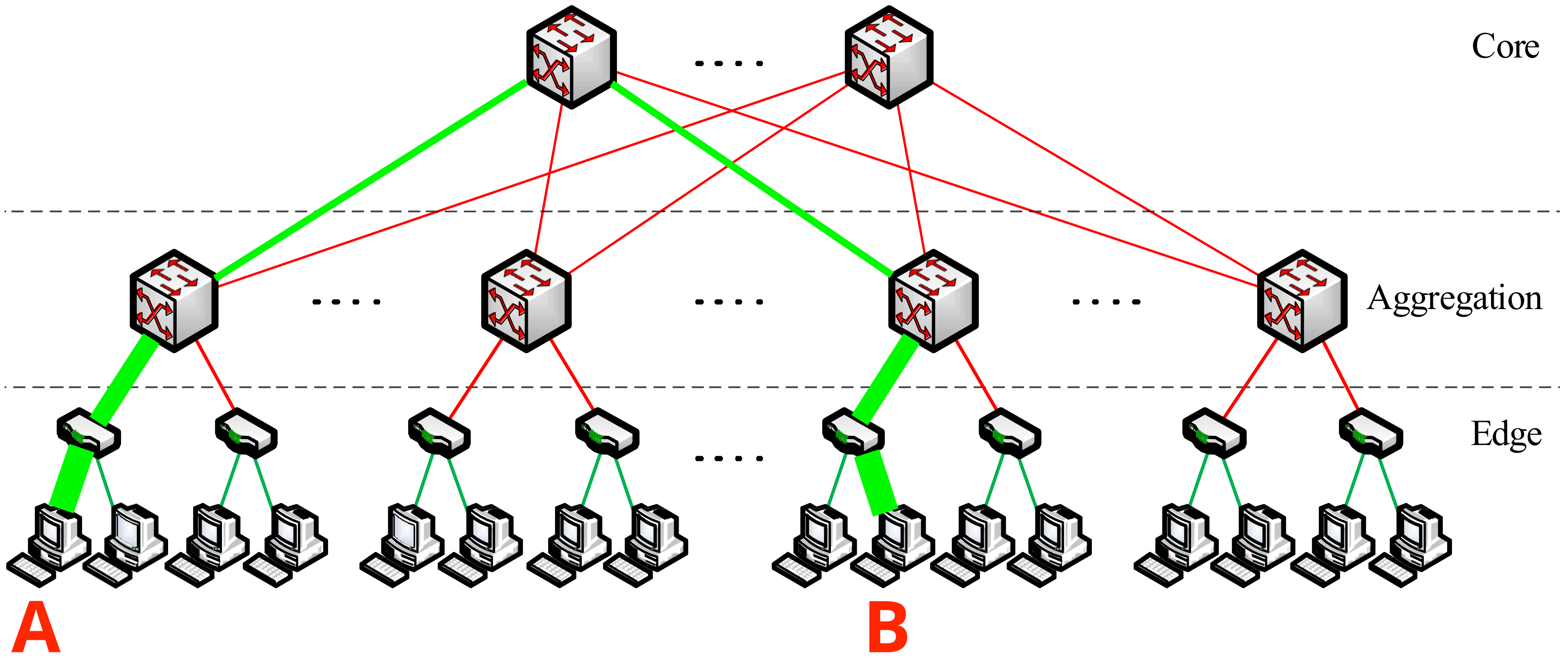
Motivation

Imagine world with "flat" data storage

- Simple easy to program

Unfortunately, data center networks where oversubscribed

- Lots of work on programming models that move computation to data, e.g., MapReduce



Aggregate Bandwidth **Above** Less Than

Aggregate Demand **Below**

Sometimes by 100x or more

Motivation

Imagine world with "flat" data storage

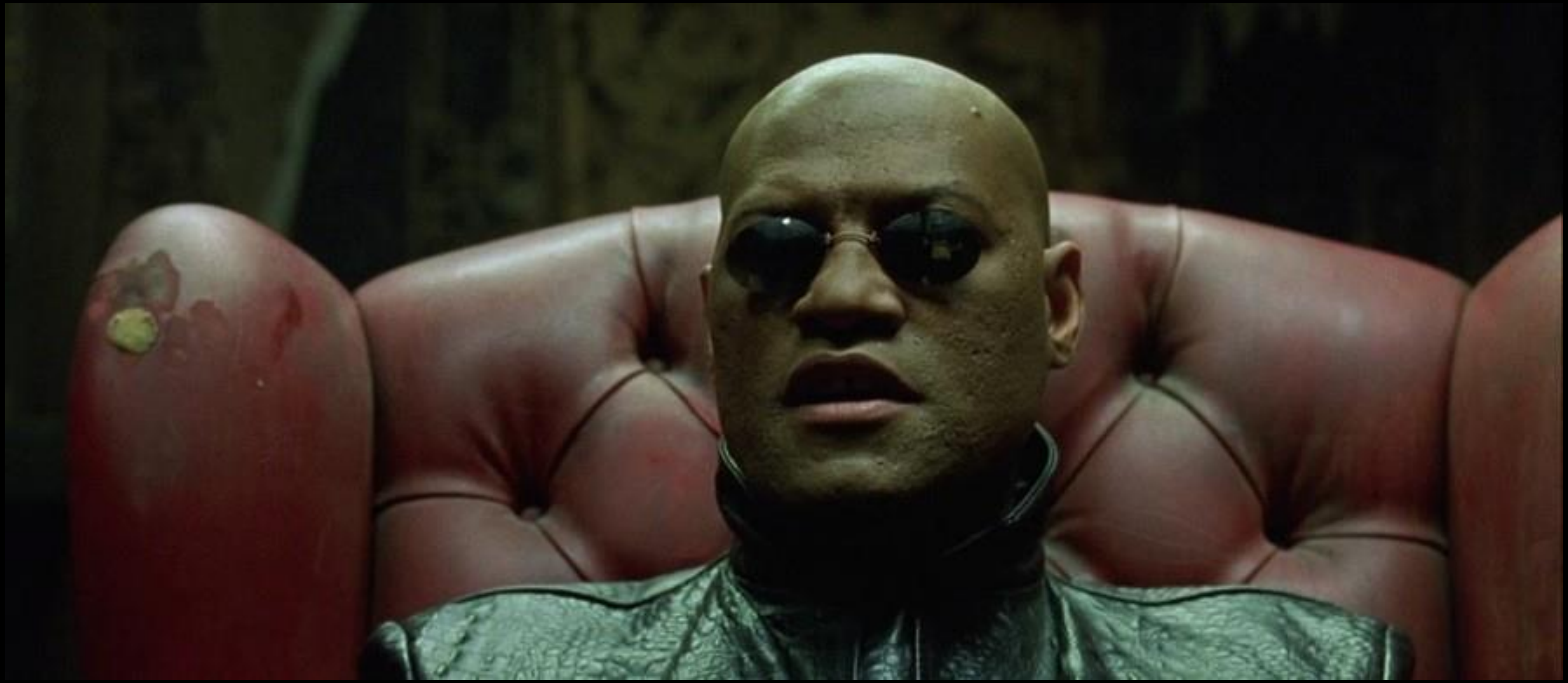
- Simple easy to program

Unfortunately, data center networks where oversubscribed

- Lots of work on programming models that move computation to data, e.g., MapReduce

Locality constraints hinder efficient resource utilization:

- Can't just schedule work wherever computation is available



What if I told you the network isn’t oversubscribed?

However

Data Center Networks got faster!

- Map Reduce designed for 100mbit

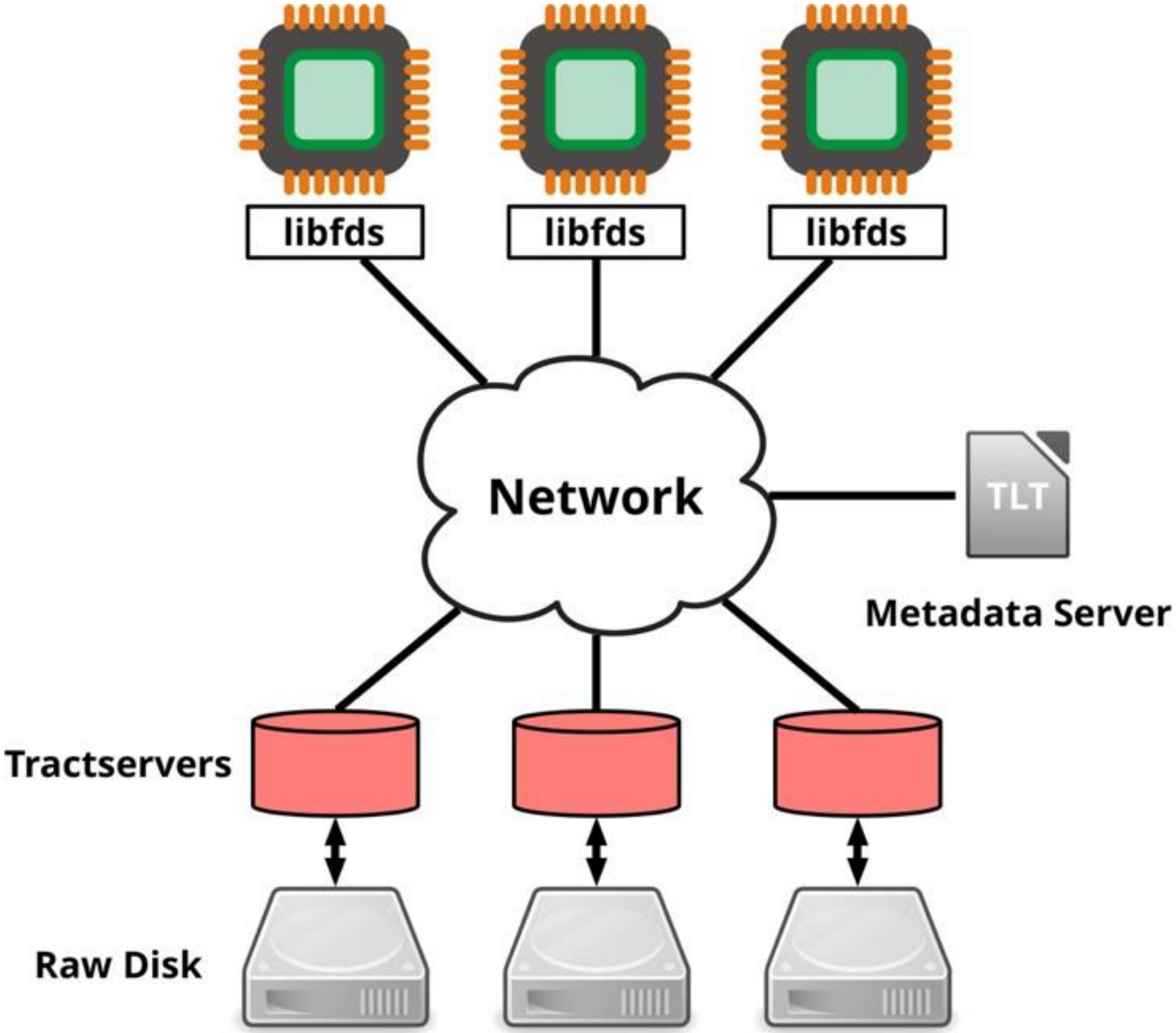
New topologies enabled "Full Bisection Bandwidth"

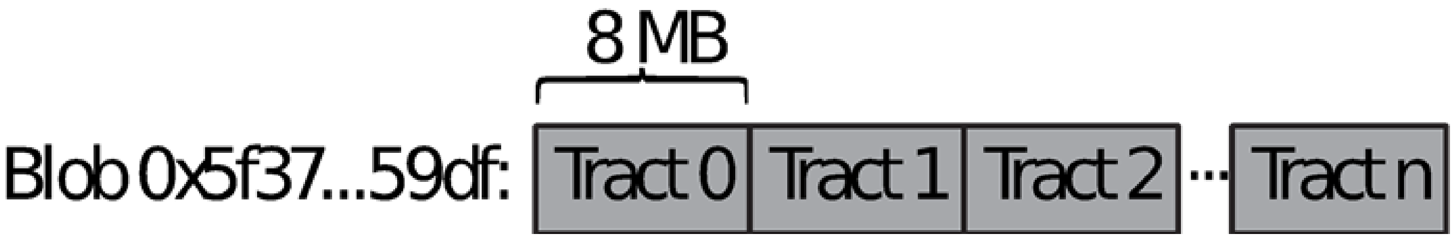
The lead to

Use consistent hashing directly from client

Result:

- read/write performance > 2GB/s
- recover 92GB in 6.2 second
- broke world record in sorting (2012)

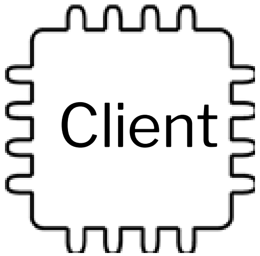

libfds
libfds
libfds
Network
TLT
Metadata Server
Tractservers
Raw Disk



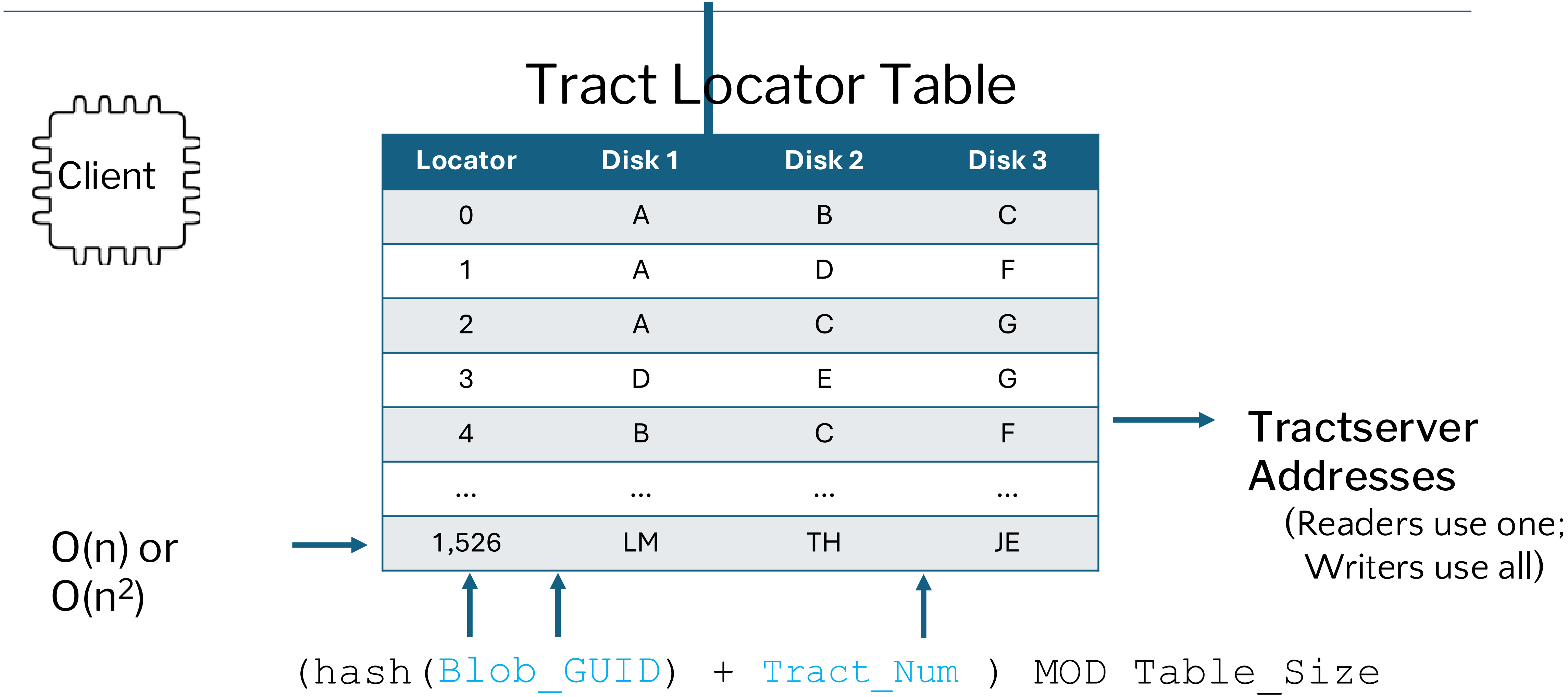
```
// create a blob with the specified GUID
CreateBlob(GUID, &blobHandle, doneCallbackFunction);

// Write 8mb from buf to tract 0 of the blob.
blobHandle->WriteTract(0, buf, doneCallbackFunction);

// Read tract 2 of blob into buf
blobHandle->ReadTract(2, buf, doneCallbackFunction);
```



Metadata Server



Tract Locator Table

Locator	Disk 1	Disk 2	Disk 3
0	A	B	C
1	A	D	F
2	A	C	G
3	D	E	G
4	B	C	F
…	…	…	…
1,526	LM	TH	JE

O(n) or
$O(n^2)$

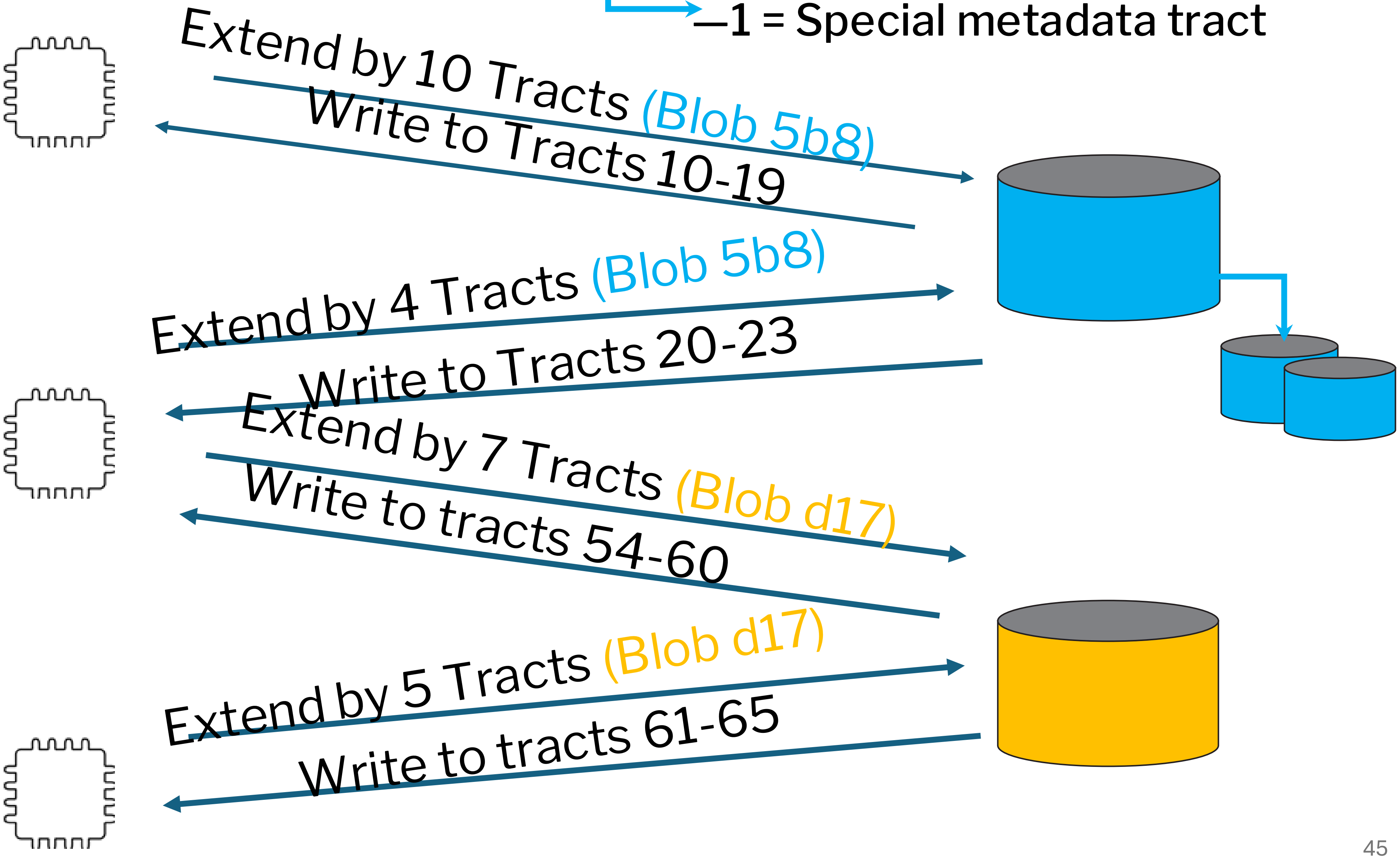
Tractserver Addresses

(Readers use one;
Writers use all)

`(hash(Blob_GUID) + Tract_Num ) MOD Table_Size`

```
(hash(Blob_GUID) + Tract_Num) MOD Table_Size
```

—1 = Special metadata tract



Extend by 10 Tracts (Blob 5b8)

Write to Tracts 10-19

Extend by 4 Tracts (Blob 5b8)

Write to Tracts 20-23

Extend by 7 Tracts (Blob d17)

Write to tracts 54-60

Extend by 5 Tracts (Blob d17)

Write to tracts 61-65

Conclusions

Ceph and FDS two great examples of using form consistent hashing to avoid centralized metadata for location

FDS and Ceph eventually focused on relaxed semantics for scale, i.e. object storage

Ceph discussed at-scale recovery, FDS did it.

Ceph had tightly coupled cluster, FDS assumed full bi-sectional BW

BW is, again, limited and SSD and NVME have changed the equation; locality now matters again.