AWS Simple Storage Service (S3)

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 10a

Yue Cheng

University of Virginia

Some material taken/derived from:

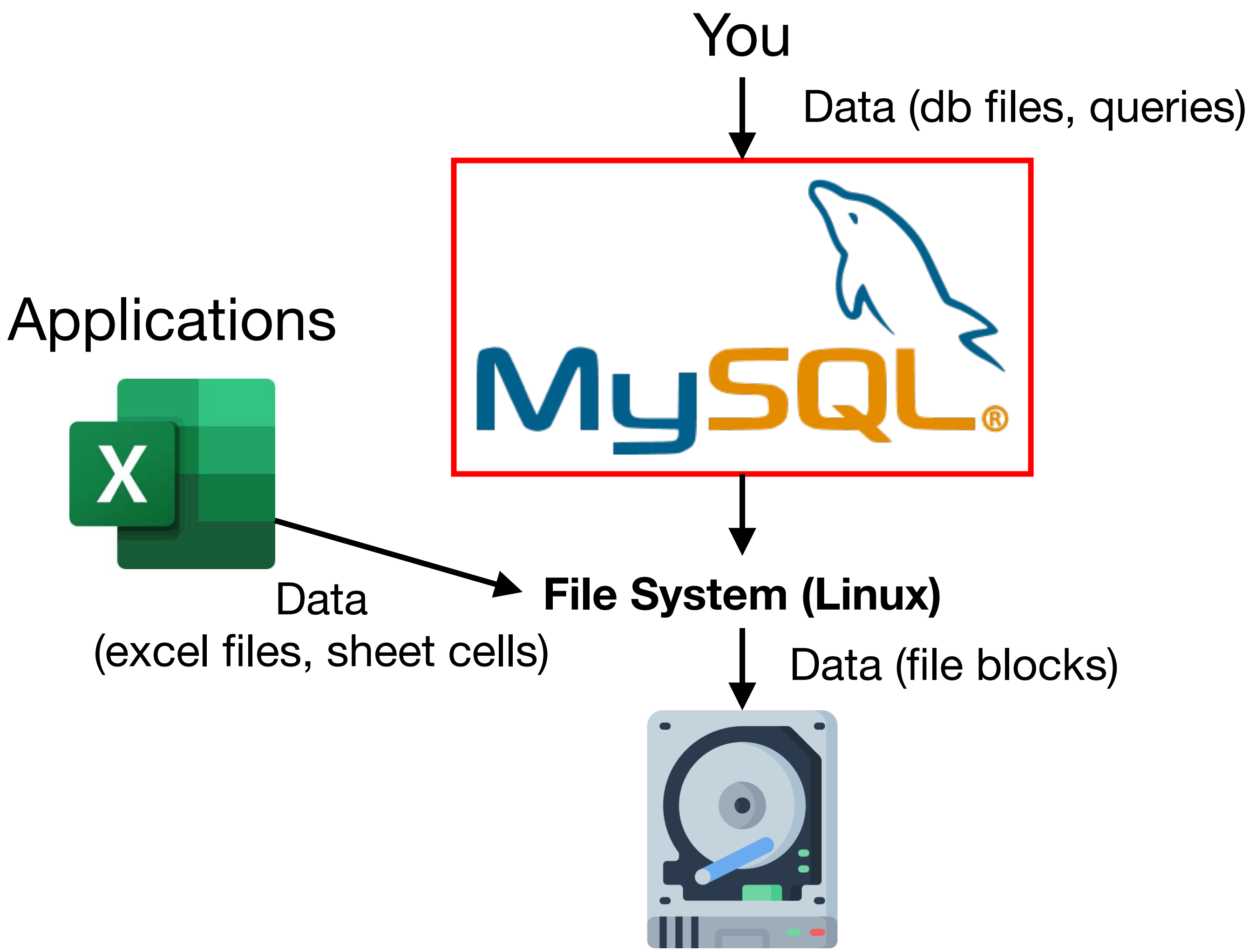
- Wisconsin CS-537 materials by Remzi Arpaci-Dusseau.

Learning objectives

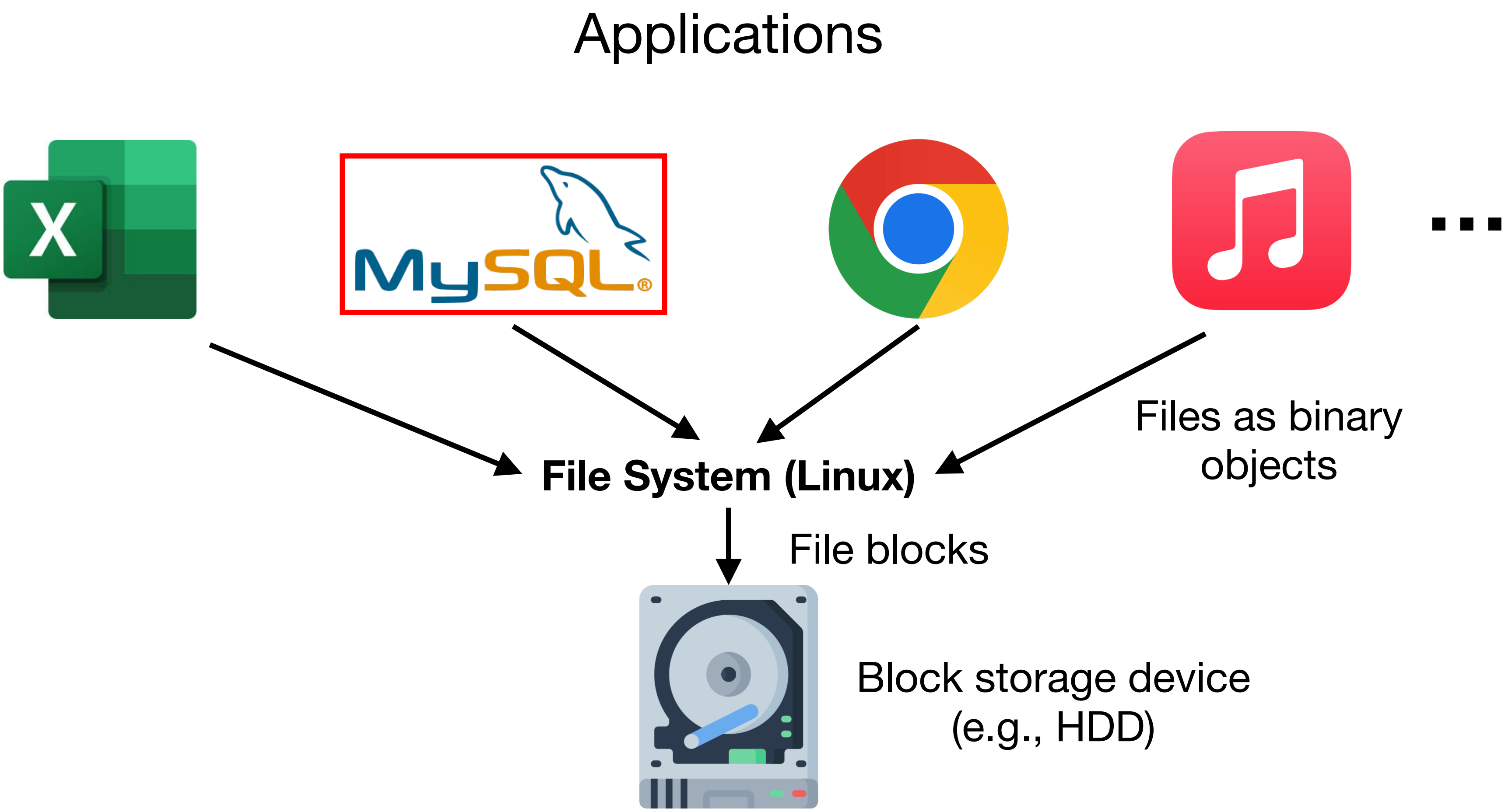
- Understand basic working mechanism of a hard disk drive
  - And why S3 is built primarily on HDDs but not SSDs
- Know different load balancing strategies
  - Replication-based
  - Striping-based (erasure-coding)
- Know basic RAID algorithms
  - Closely related to erasure coding algorithms such as Reed-Solomon

Different types of storage systems

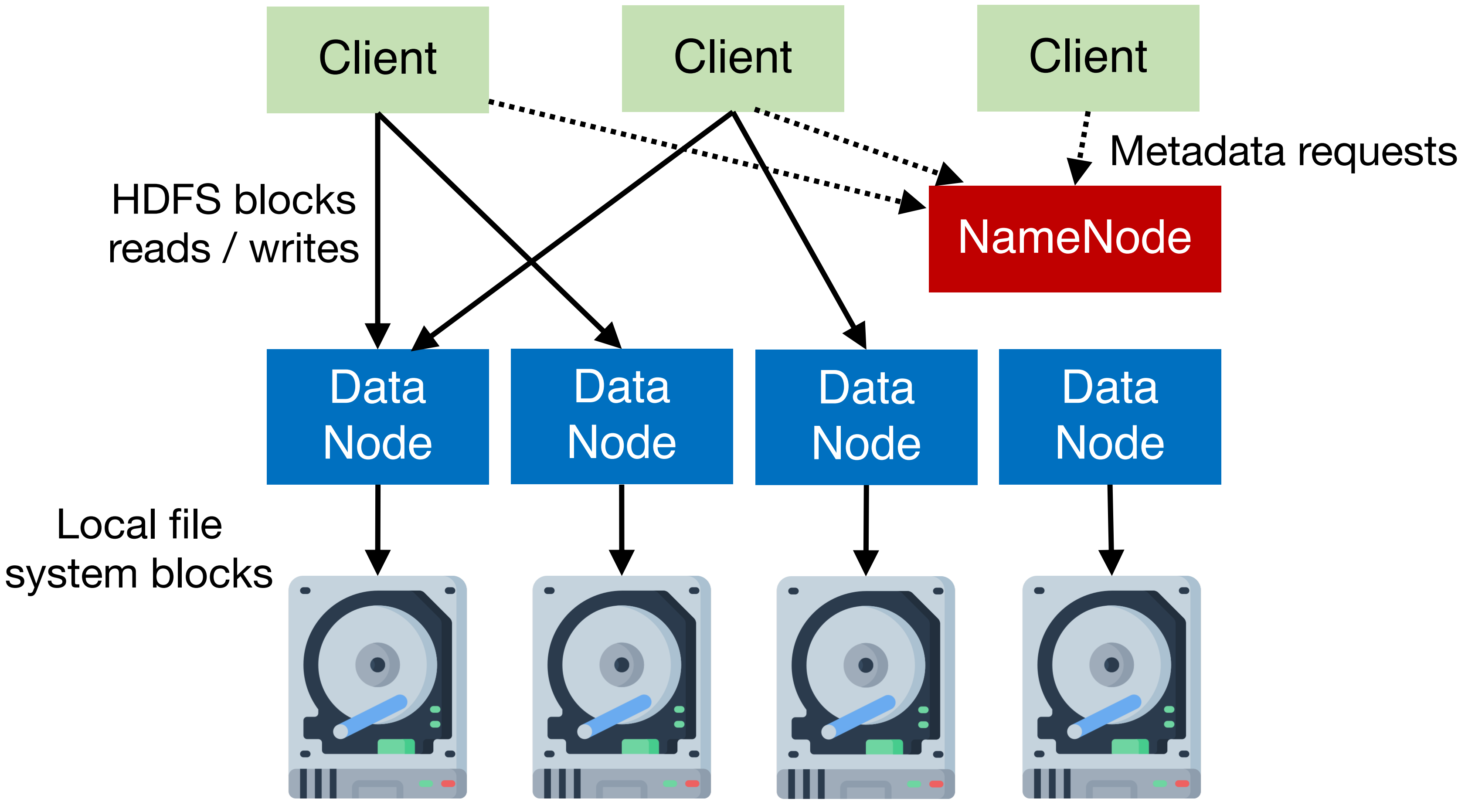
Local apps + local file systems



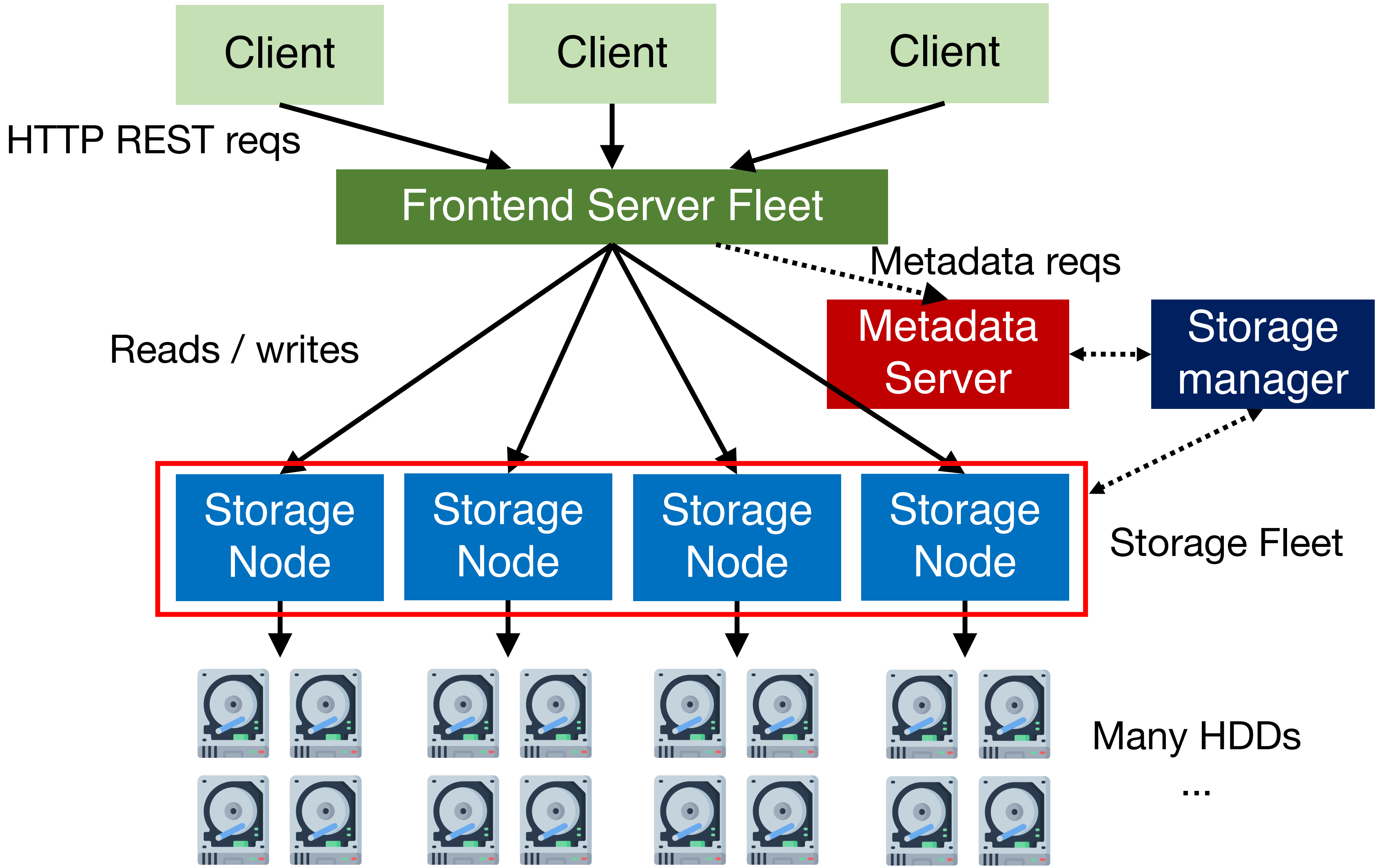
Local apps + local file systems



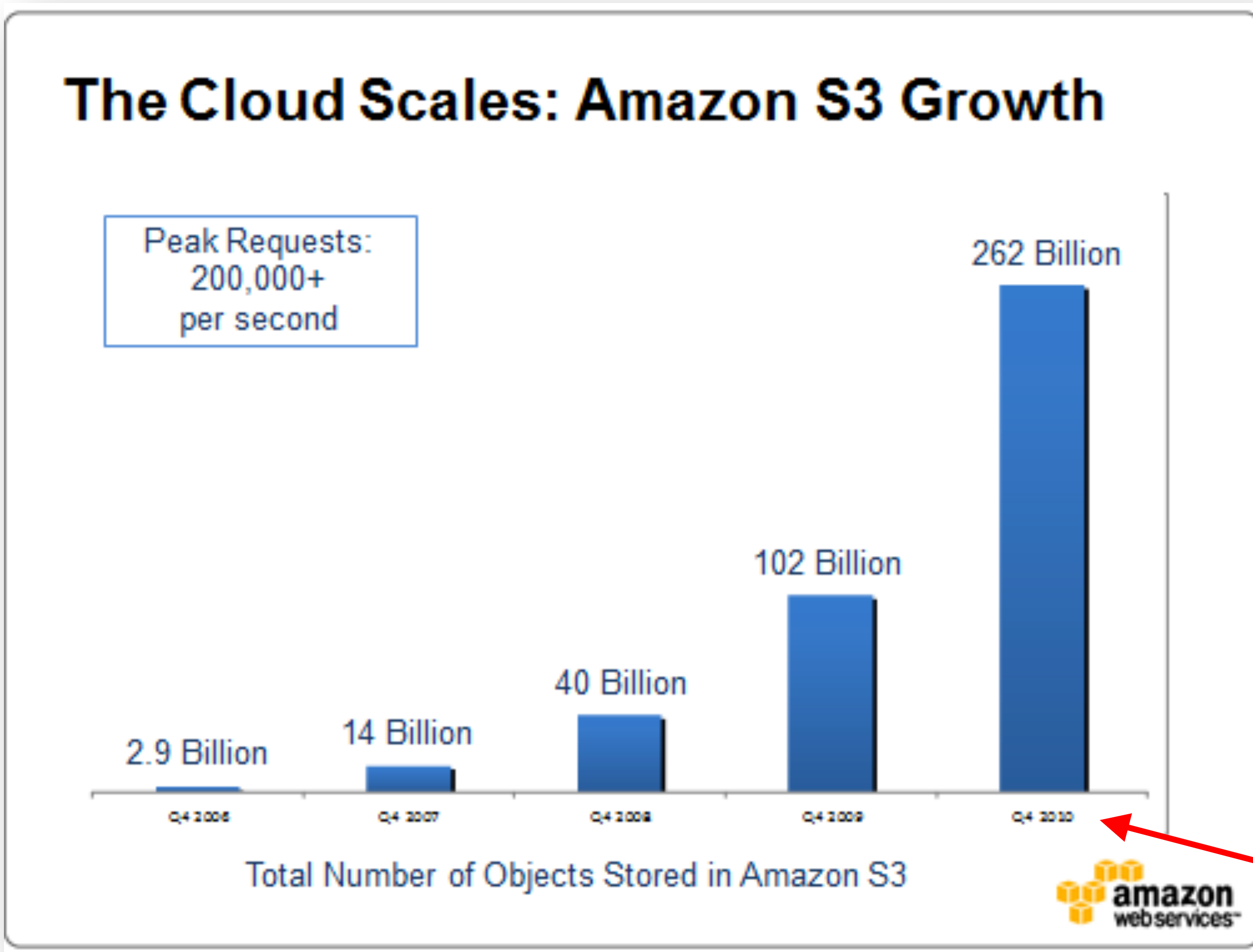
Distributed file systems



AWS S3 (Simple Storage Service)



Some S3 statistics



Circa 2010

https://www.pingdom.com/blog/amazon-s3-will-soon-store-a-trillion-objects/

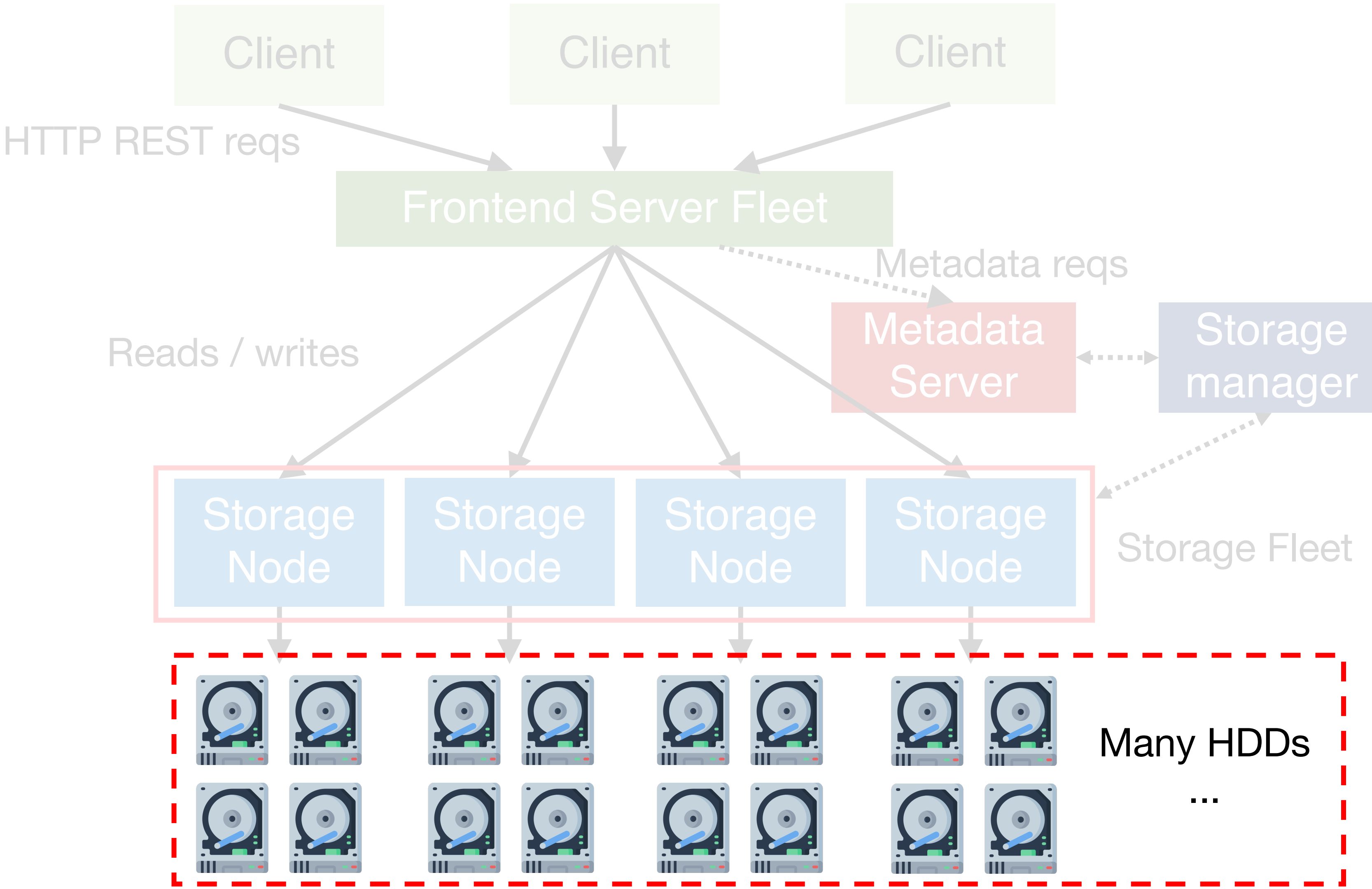
Some S3 statistics

2023

Capacity and throughput	Amazon S3 holds more than **280 trillion objects** and averages over **100 million requests per second**
Events	Every day, Amazon S3 sends over **125 billion event notifications** to serverless applications
Replication	Customers use Amazon S3 Replication to **move more than 100 PB** of data per week
Cold Storage Retrieval	Every day, customers **restore more than 1PB** from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes
Data Integrity Checks	Amazon S3 performs over **4 billion checksum computations per second**
Cost Optimization	On average, customers using Amazon S3 Storage Lens advanced metrics and recommendations have obtained **cost savings 6x greater** than the Storage Lens cost in the first six months of using it.
Flexibility	**Hundreds of thousands of data lakes** are built on Amazon S3

https://www.allthingsdistributed.com/2023/07/building-and-operating-a-pretty-big-storage-system.html

The physics of storage: HDDs



Basic interface of disks

- A magnetic disk has a sector-addressable address space
  - You can think of a disk as an array of sectors
  - Each sector (logical block) is the smallest unit of transfer

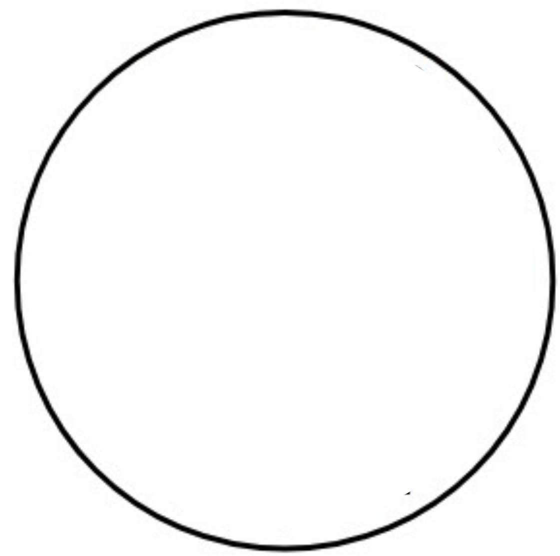
- Sectors are typically 512 or 4096 bytes

- Main operations
  - Read from sectors (blocks)
  - Write to sectors (blocks)

Disk structure

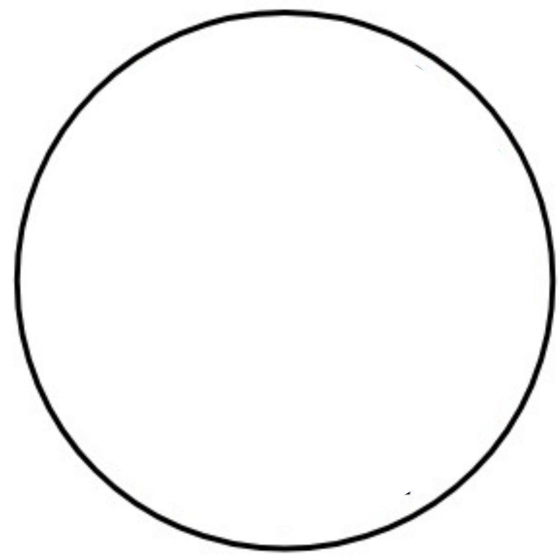
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors

Internals of hard disk drive (HDD)



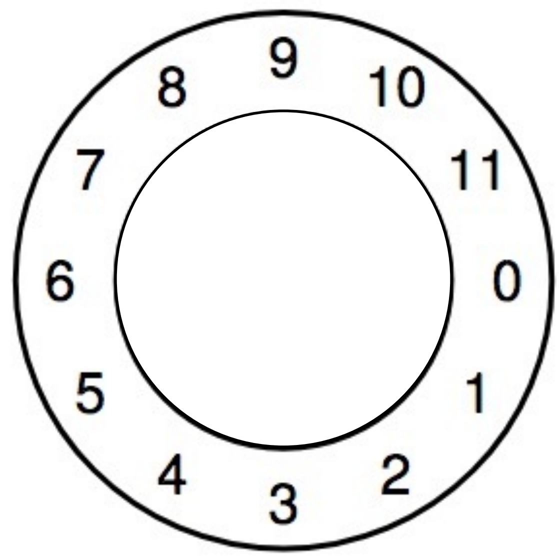
Internals of hard disk drive (HDD)

Platter
Covered with a magnetic film



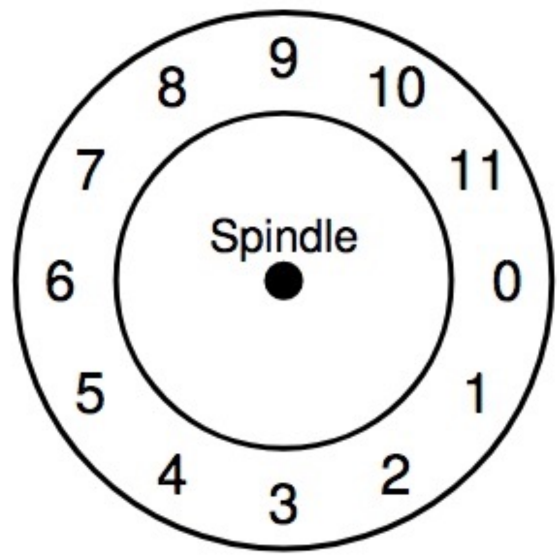
Internals of hard disk drive (HDD)

A single track example



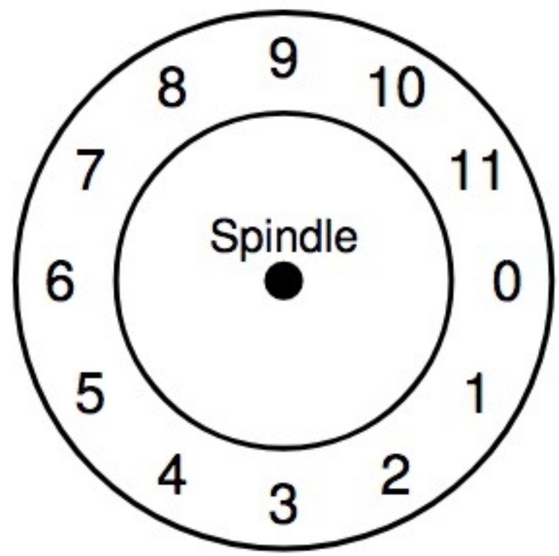
Internals of hard disk drive (HDD)

Spindle in the center of the surface



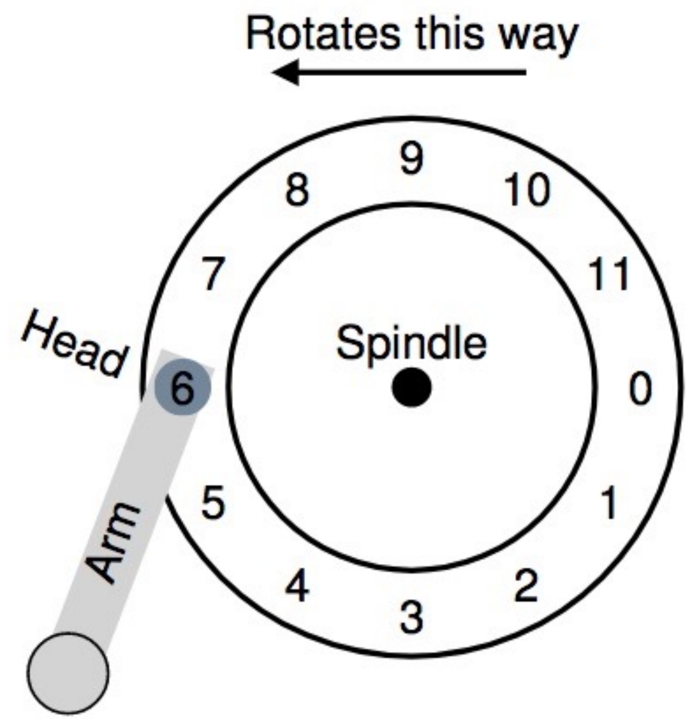
Internals of hard disk drive (HDD)

The track is divided into numbered sectors

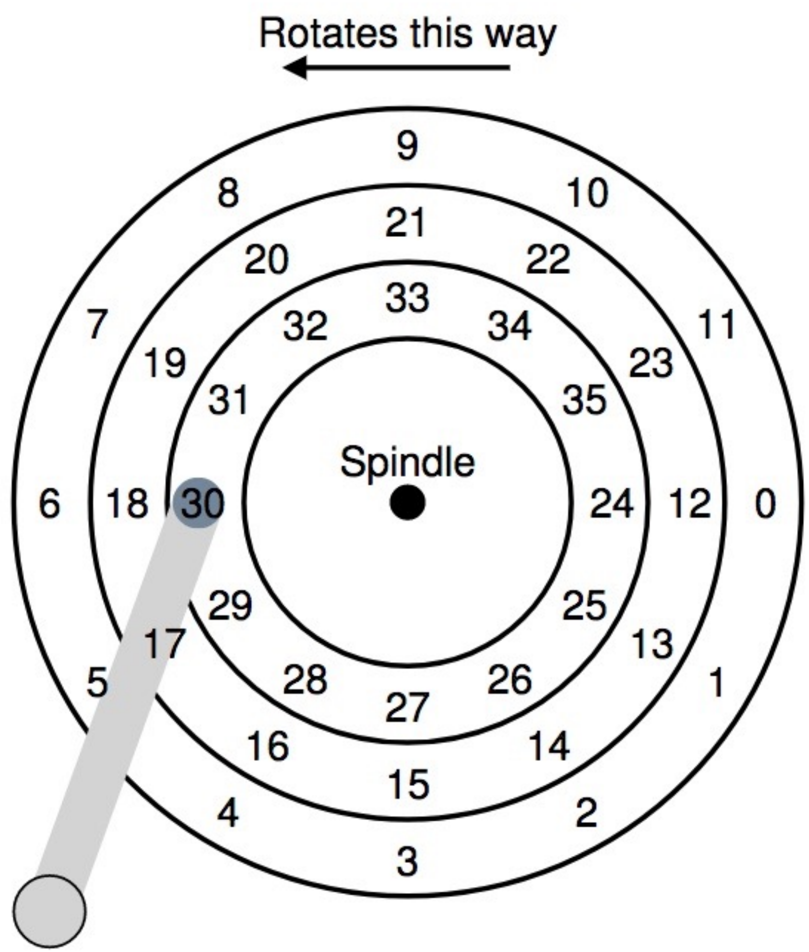


Internals of hard disk drive (HDD)

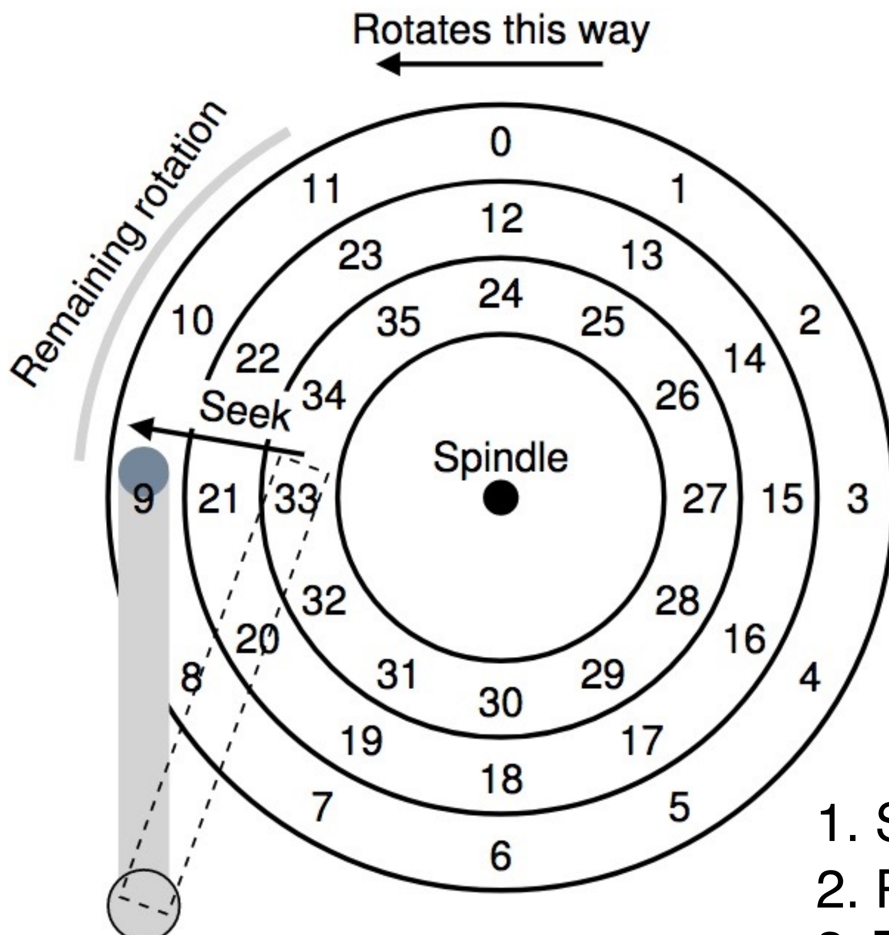
A single track + an arm + a head



Let's read sector 0

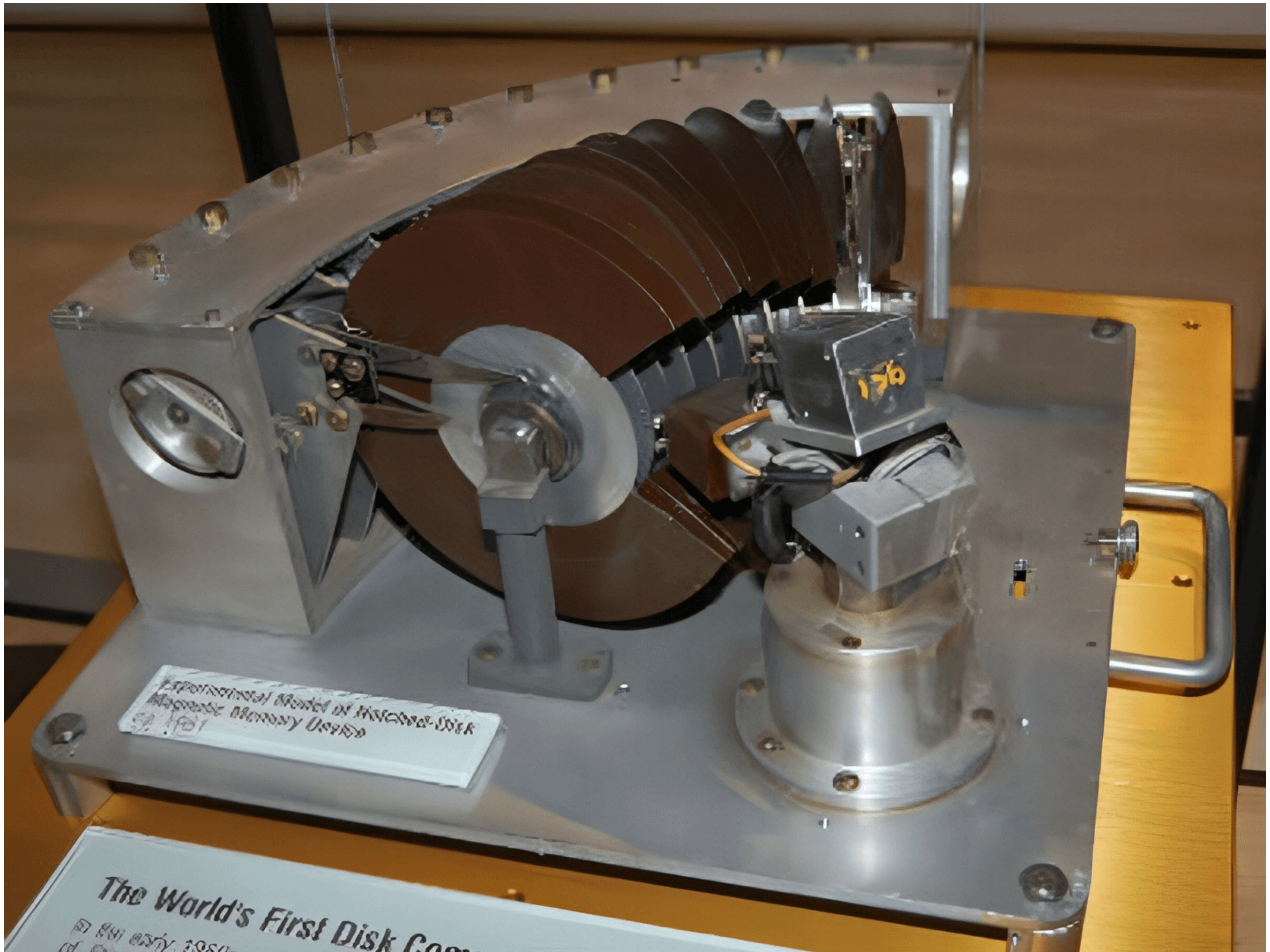


Let’s read sector 0



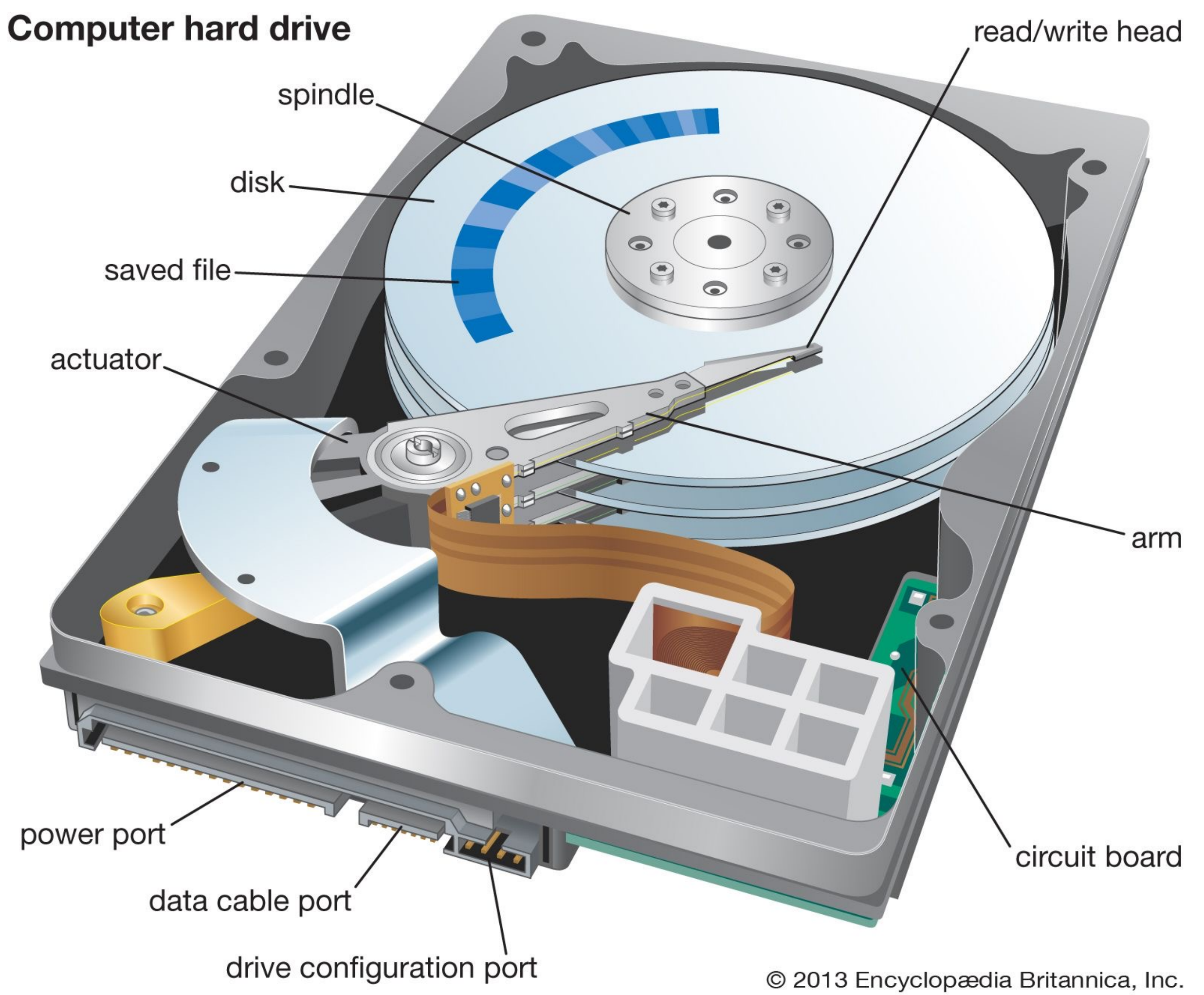
1. Seek for right track
2. Rotate (sector 9 → 0)
3. Transfer data (sector 0)

The first magnetic memory device



https://www.computerhistory.org/storageengine/rabinow-patents-magnetic-disk-data-storage/

3D view of a modern disk



https://www.britannica.com/technology/hard-disk

Don’t try this at home!

https://www.youtube.com/watch?v=9eMWG3fwiEU&feature=youtu.be&t=30s

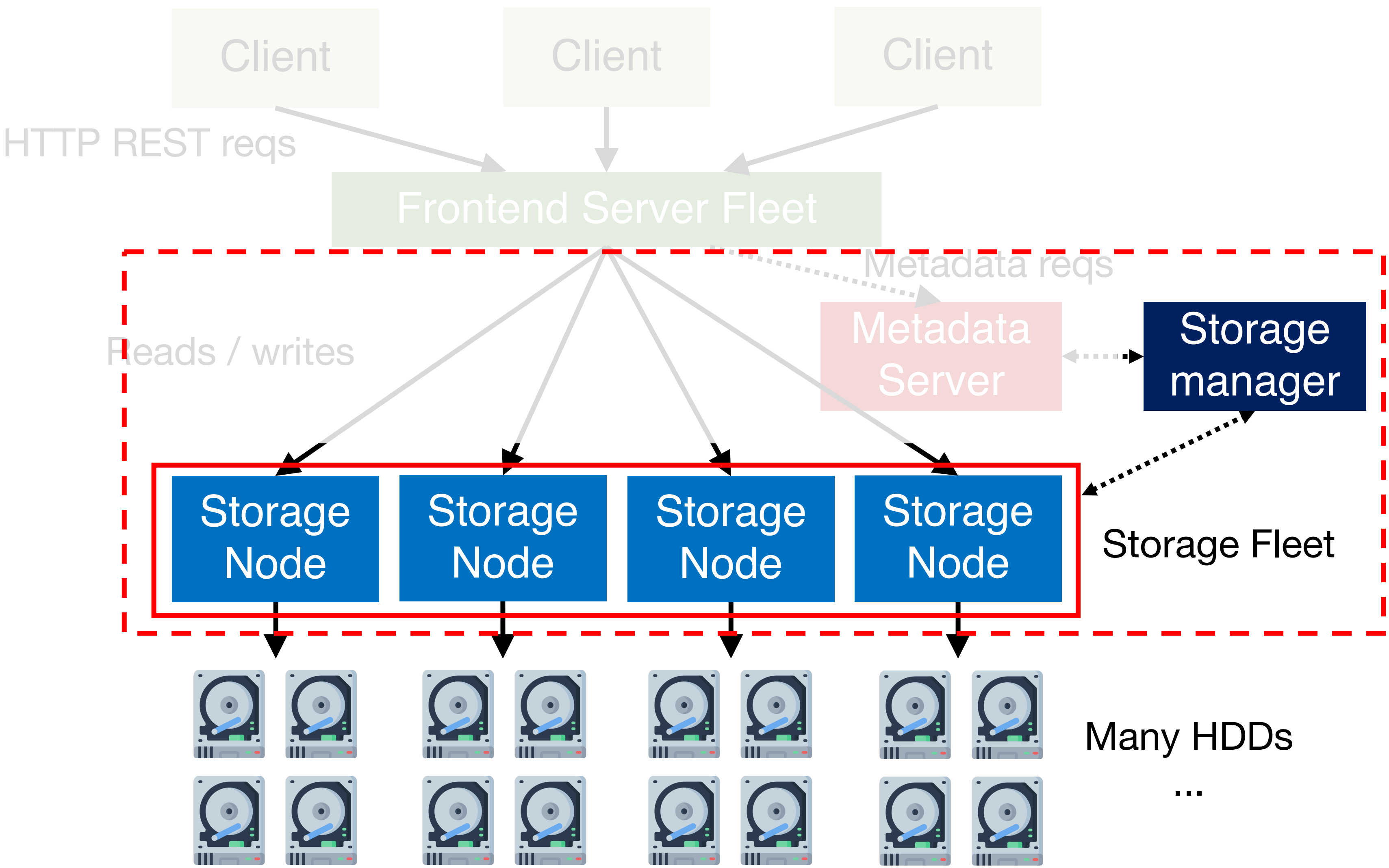
Summary of differences: SSD vs. HDD

	SSD	HDD
Stands for	*SSD* stands for Solid State Drive.	*HDD* stands for Hard Disk Drive.
How it works	SSDs store data on electronic circuits.	HDDs store data on mechanically moving, magnetic platters.
Read process	An SSD controller finds the correct address and reads its charges.	Ann HDD I/O controller sends a signal that moves the actuator arm. The read/write head then reads charges.
Write process	An SSD copies data to a new block, then erases the old block. It then writes new to the old block by changing its charges.	An HDD moves the read/write head to the nearest available location. It then writes data by changing the charge of bits in that area.
Performance	SSDs are faster. They're silent and run cooler.	HDDs are slower as their platters have to move around. They release more heat and are noisy.
Cost	SSDs are costlier.	HDDs are less costly and larger storage volumes are commercially popular.
Durability	SSDs are electrical, which makes them less prone to damage.	HDDs have moving mechanical parts that make them comparatively less durable.

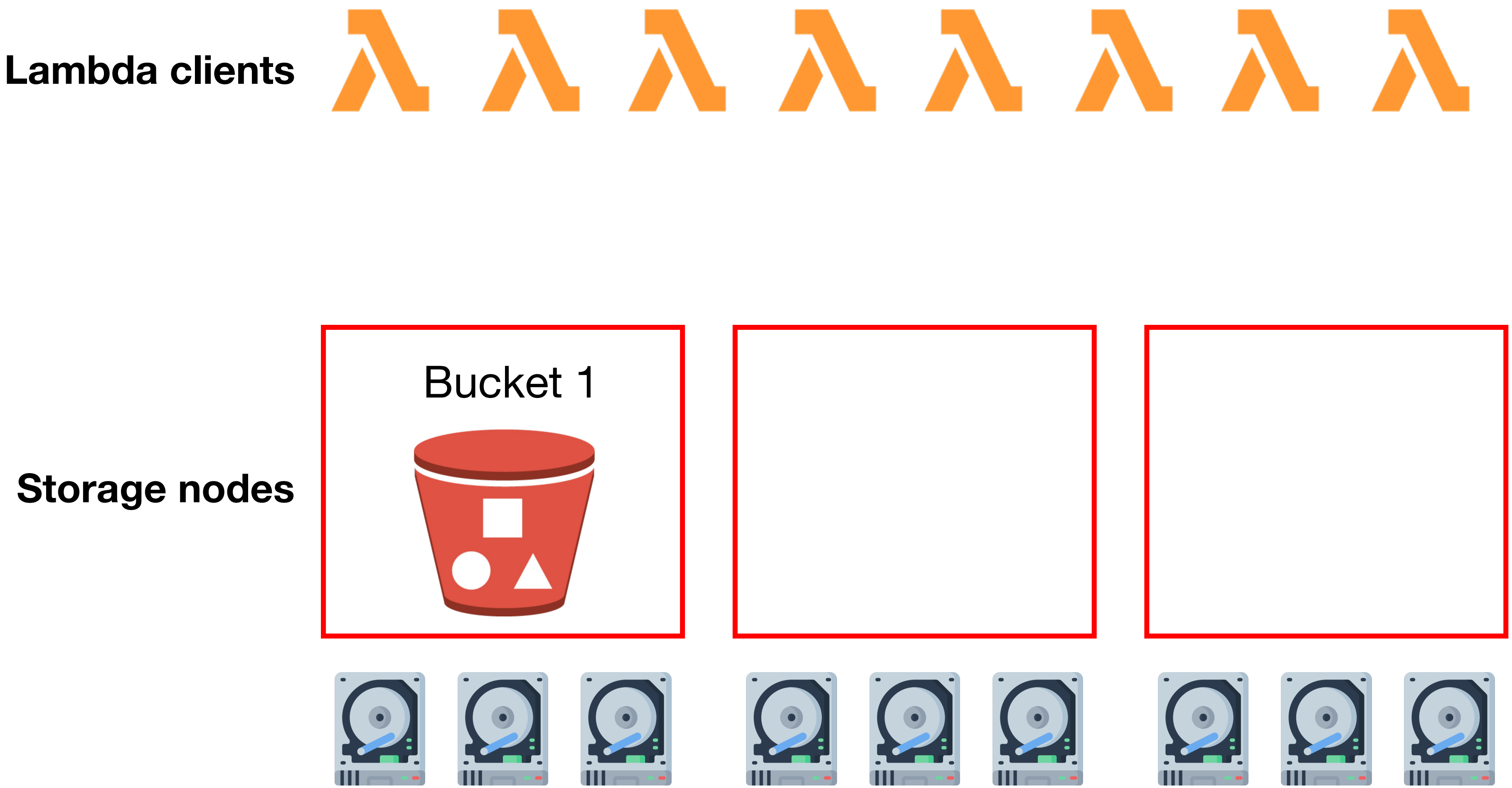
https://aws.amazon.com/compare/the-difference-between-ssd-hard-drive/

What makes HDDs an ideal storage for S3?

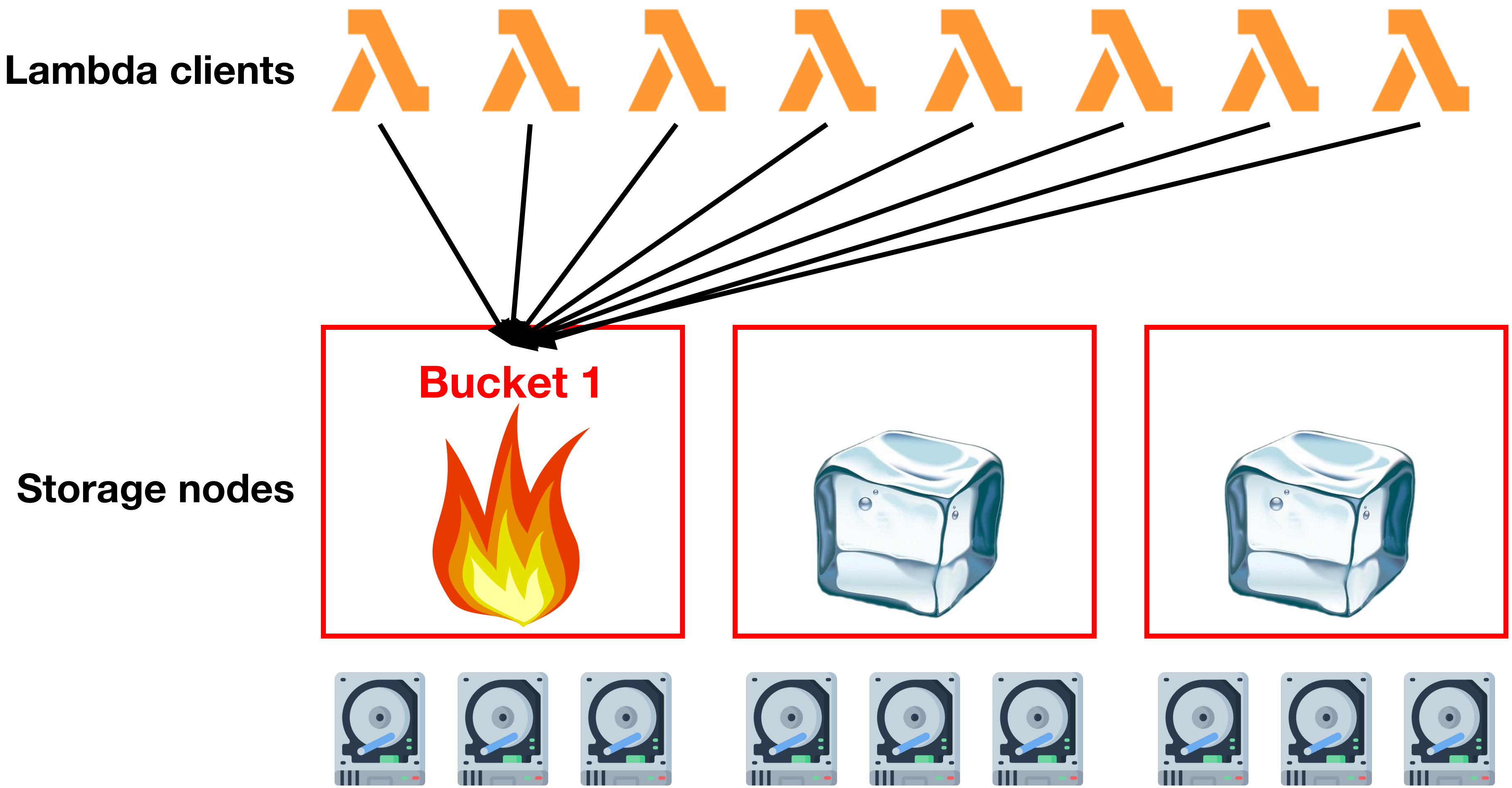
Performance and data placement



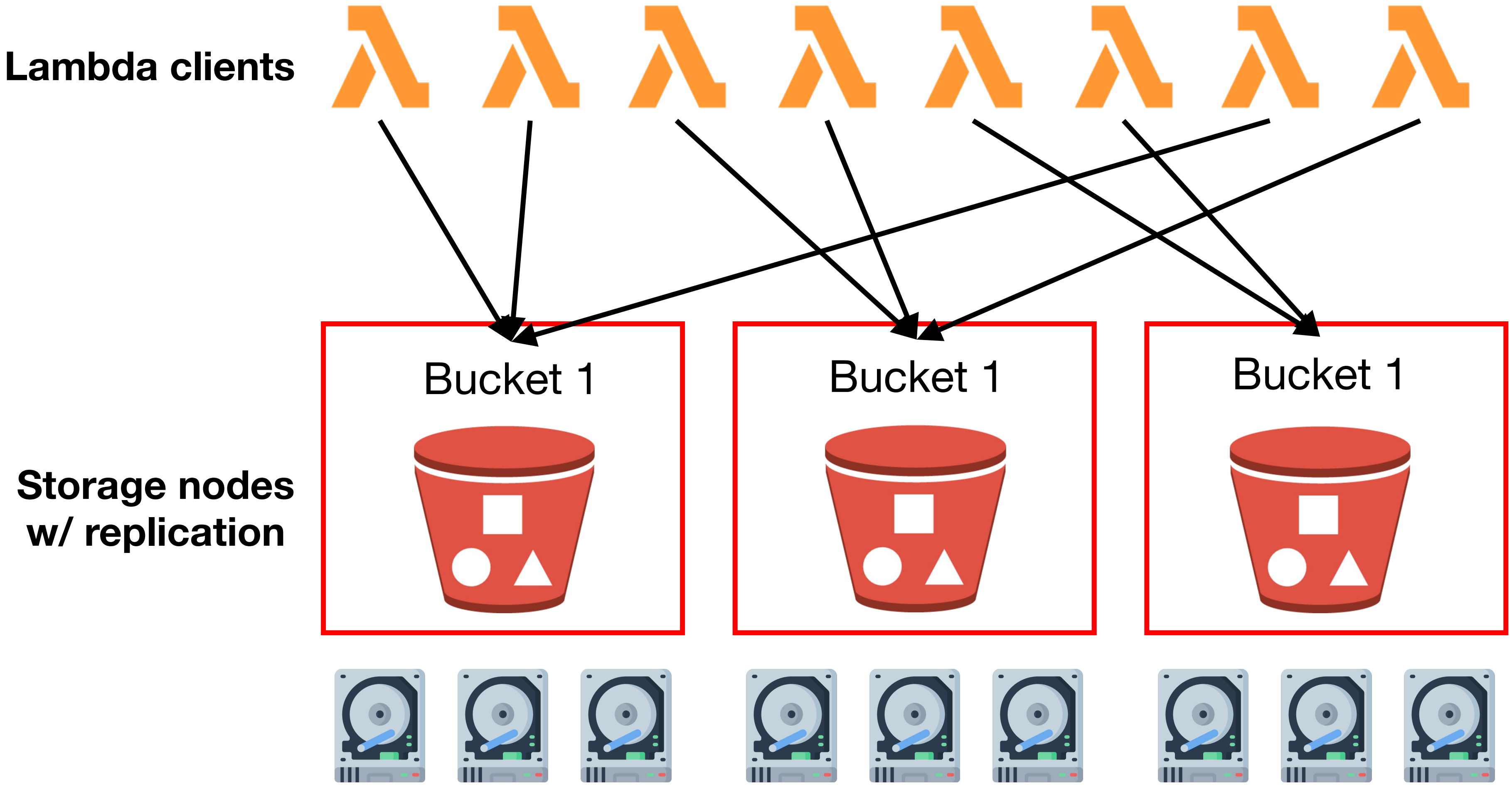
Hot data creates a hotspot



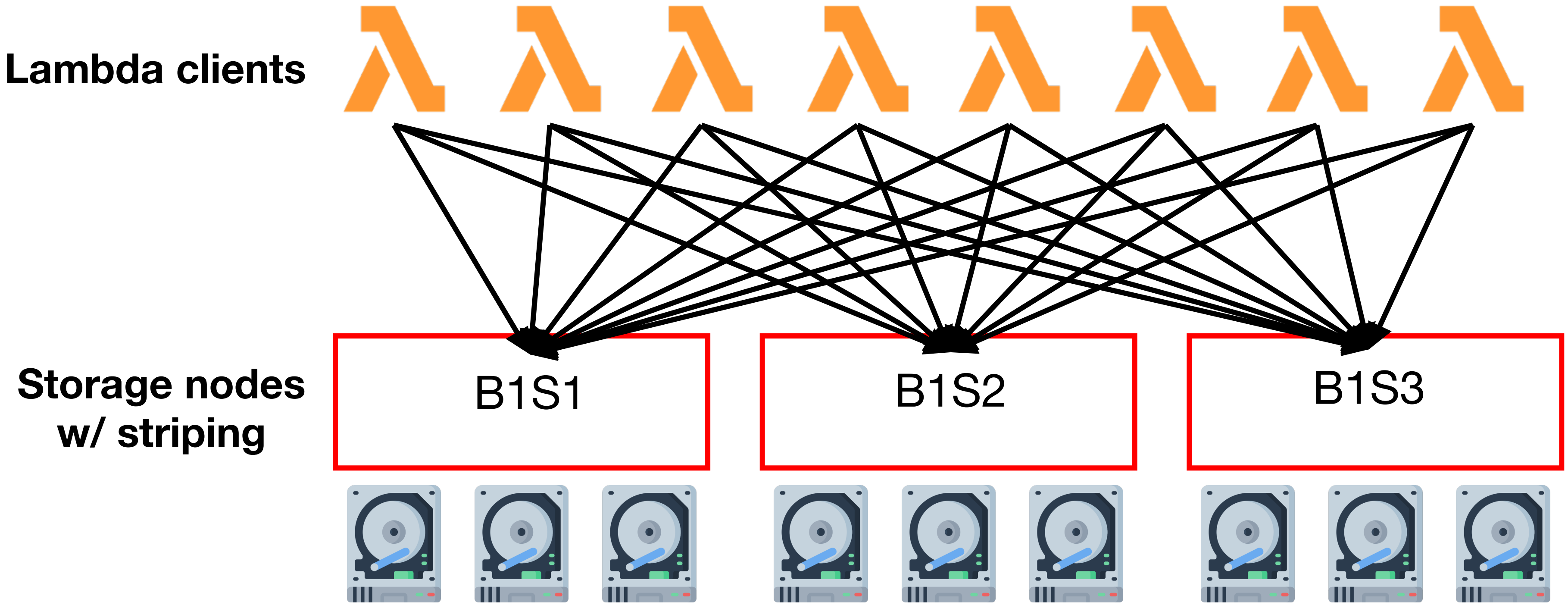
Hot data creates a hotspot



Replication helps balance the heat



Striping helps balance the heat



Why hotspots are bad for disks

Modeling disk performance

I/O latency of disks

$$L_{I/O} = L_{seek} + L_{rotate} + L_{transfer}$$

Disk access latency at millisecond level

Seek, Rotate, Transfer

- Seek may take several milliseconds (ms)

- Settling along can take 0.5 - 2ms

- Entire seek often takes 4 - 10ms

Seek, Rotate, Transfer

- Rotation per minute (RPM)
  - 7200 RPM is common nowadays
  - 15000 RPM is high end
  - Old computers may have 5400 RPM disks

- `1 / 7200 RPM = 1 minute / 7200 rotations =`

  `1 second / 120 rotations =` **`8.3 ms`** `/ rotation`

Seek, Rotate, Transfer

- Rotation per minute (RPM)
  - 7200 RPM is common nowadays
  - 15000 RPM is high end
  - Old computers may have 5400 RPM disks

- `1 / 7200 RPM = 1 minute / 7200 rotations =`

  `1 second / 120 rotations =` **`8.3 ms`** `/ rotation`

- Statistically, it may take 4.2 ms **on average** to rotate to target (0.5 * 8.3 ms)

Seek, Rotate, Transfer

- Relatively fast
  - Depends on RPM and sector density
- 100+ MB/s is typical for SATA I (1.5Gb/s max)
  - Up to 600MB/s for SATA III (6.0Gb/s)
- `1s / 100MB = 10ms / MB = 4.9us / sector`
  - Assuming 512-byte sector

Workloads

- Seeks and rotations are slow while transfer is relatively fast
- What kind of workload is best suited for disks?

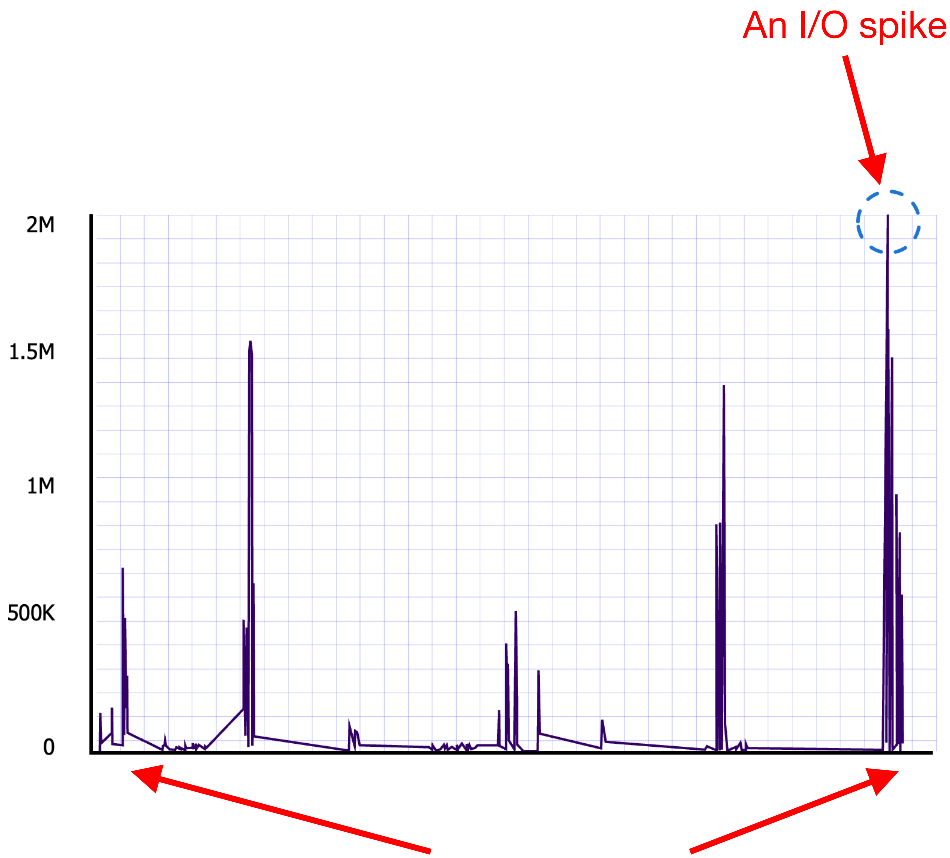
Workloads

- Seeks and rotations are slow while transfer is relatively fast

- What kind of workload is best suited for disks?
  - Sequential I/O: access sectors in order (transfer dominated)

- Random workloads access sectors in a random order (seek+rotation dominated)
  - **Typically slow on disks**

S3 workloads can be quite spiky



A large-scale data-intensive application (e.g., parallel data processing from thousands of Lambda functions)

Bucket size: 3.7 PB
Peak throughput: 2.3M req/s

$\frac{3.7\text{ PB}}{26\text{TB/HDD}}$ **= 143 HDDs** (Storage constrained)

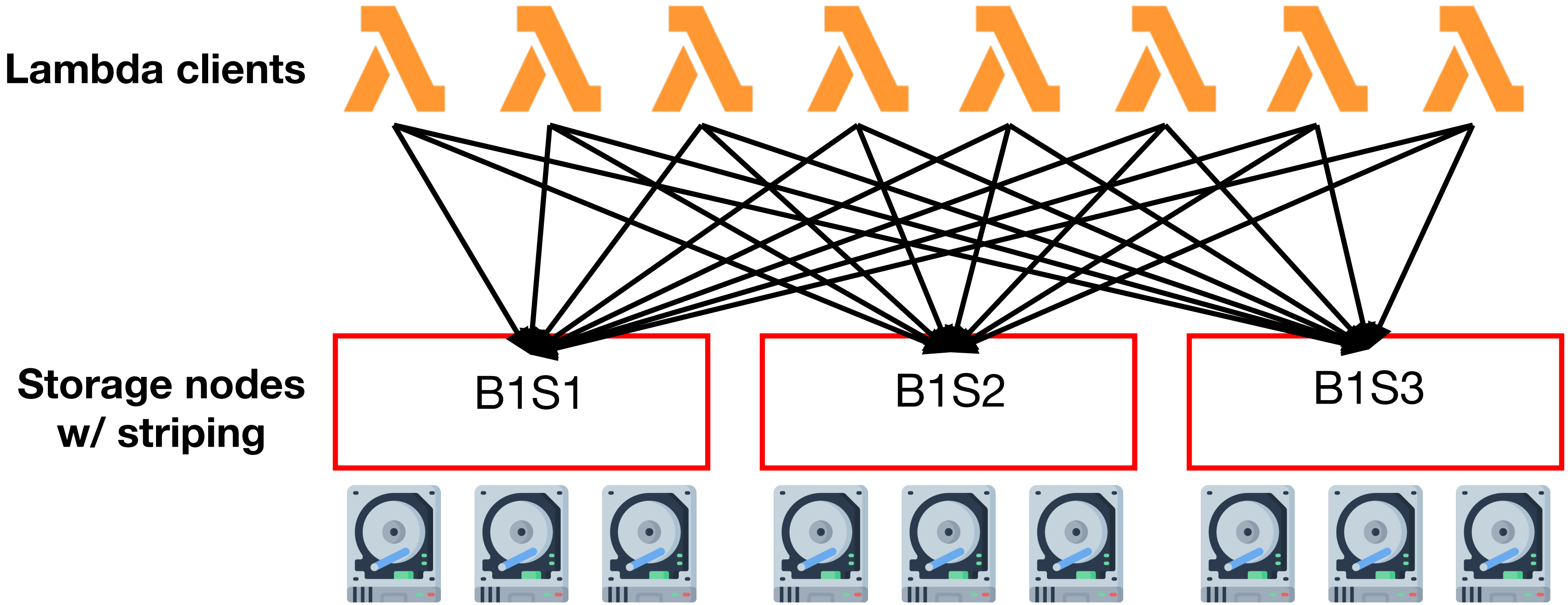
143 HDDs × 120 IOPS = 17,160 IOPS

$\frac{2.3\text{M req/sec}}{120\text{ IOPS}}$ **= 19,000 HDDs** (IO constrained)

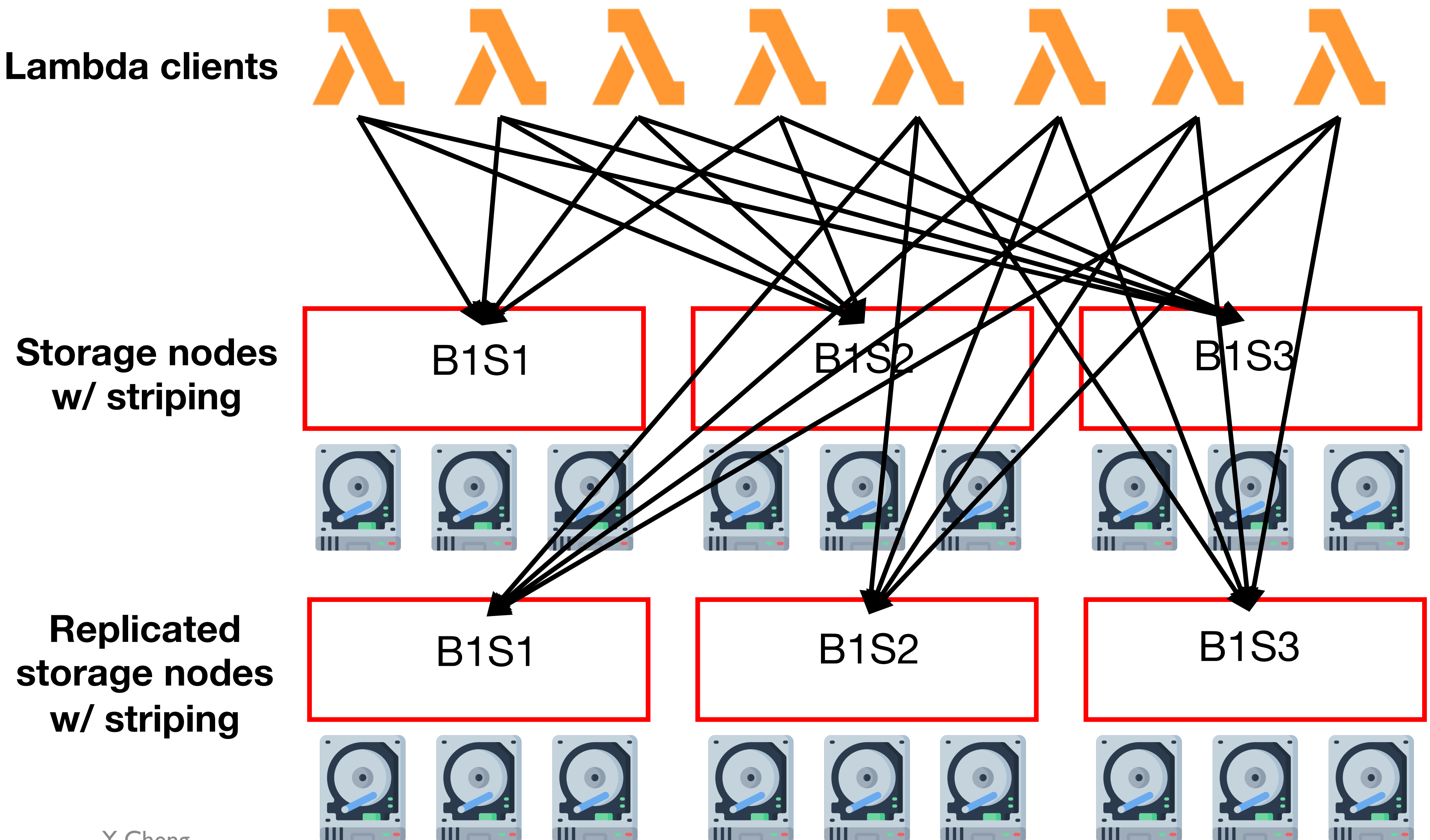
Balancing the load using scale

See the video example in https://www.allthingsdistributed.com/2023/07/building-and-operating-a-pretty-big-storage-system.html

Striping helps balance the heat

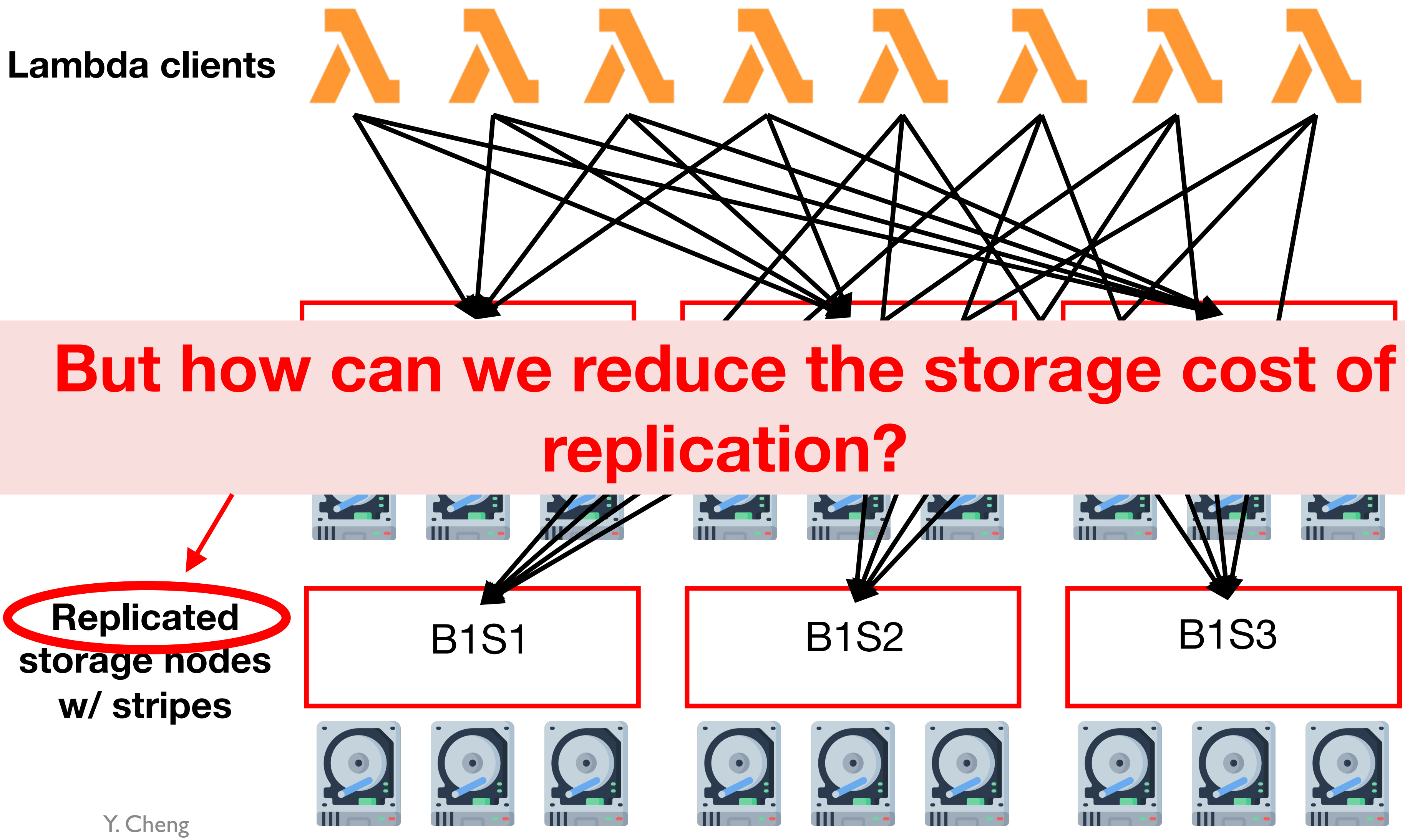


Striped data needs to be replicated



Y. Cheng

Striped data needs to be replicated



But how can we reduce the storage cost of replication?

Y. Cheng

RAID and erasure coding

Redundant array of inexpensive disks

4 disks

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

4 disks

	Disk 0	Disk 1	Disk 2	Disk 3
	0	1	2	3
stripe:	4	5	6	7
	8	9	10	11
	12	13	14	15

How to map?

- Given logical address A:
  - Disk = …
  - Offset = …

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

How to map?

- Given logical address A:
  - Disk = `A % disk_count`
  - Offset = `A / disk_count`

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Mapping example: Find block 13

- Given logical address 13:
  - Disk = `13 % 4 = 1`
  - Offset = `13 / 4 = 3`

Offset	Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Mapping example: Find block 13

- Given logical address 13:
  - Disk = `13 % 4 = 1`
  - Offset = `13 / 4 = 3`

Problem with naïve striping is that there is no redundancy support.

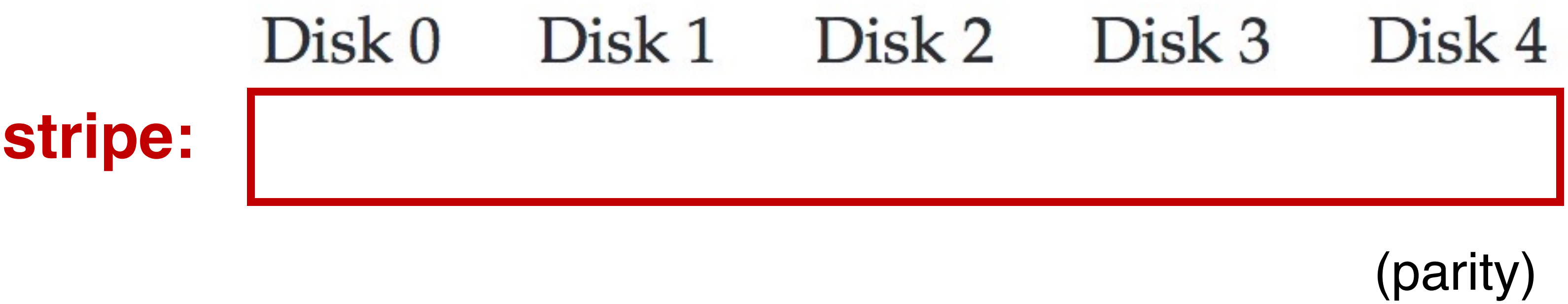
Offset	Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

5 disks

Parity disk

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

Example



Example

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	4	3	0	2	
					(parity)

Example

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	4	3	0	2	9
					(parity)

Example

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	X	3	0	2	9
					(parity)

Example

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	**4**	3	0	2	9
					(parity)

Parity function: XOR example

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

Parity function: XOR example

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity function: XOR example

	Block0	Block1	Block2	Block3	Parity
stripe:	00	10	11	10	11
	10	01	00	01	10

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity function: XOR example

	Block0	Block1	Block2	Block3	Parity
stripe:	X	10	11	10	11
	10	01	00	01	10

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity function: XOR example

	Block0	Block1	Block2	Block3	Parity
stripe:	X	10	11	10	11
	10	01	00	01	10

Block0 = XOR(10,11,10,11) = 00

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity function: XOR example

	Block0	Block1	Block2	Block3	Parity
stripe:	00	10	11	10	11
	10	01	00	01	10

Block0 = XOR(10,11,10,11) = **00**

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity function: XOR example

Q: How many disks can fail?

stripe: (first row)

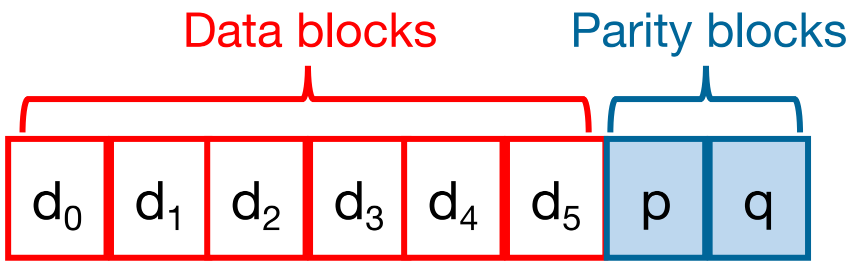
Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

Block0 = XOR(10,11,10,11) = **00**

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

RAID-6



RAID-6 can fail at most 2 disks at a time.

Encoding

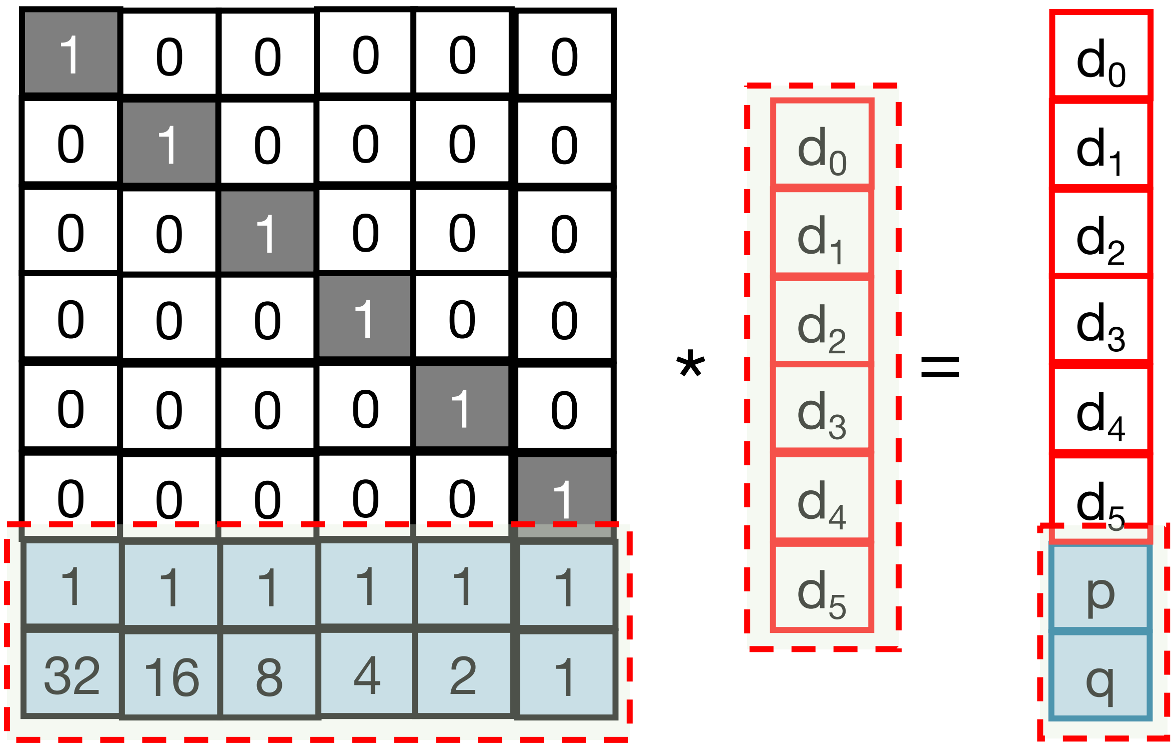
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 32 & 16 & 8 & 4 & 2 & 1 \end{bmatrix} * \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ p \\ q \end{bmatrix}$$

The product vector is called a codeword

Generator matrix

[8 x 6] * [6 x 1] = [8 x 1]

Encoding



$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \longrightarrow p$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \longrightarrow q$$

Decoding w/ a parity check matrix

Parity check matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 32 & 16 & 8 & 4 & 2 & 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ p \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus p = 0$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \oplus q = 0$$

Handling failures w/ decoding

$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus p = 0$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \oplus q = 0$$

Suppose disk1 (d_1) and disk4 (d_4) fail

Handling failures w/ decoding

$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus p = 0$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \oplus q = 0$$

Suppose disk1 (d_1) and disk4 (d_4) fail

Step 1: Put the failed data on the right of the equations.

$$d_0 \oplus d_2 \oplus d_3 \oplus d_5 \oplus p = d_1 \oplus d_4$$

$$32d_0 \oplus 8d_2 \oplus 4d_3 \oplus d_5 \oplus q = 16d_1 \oplus 2d_4$$

Handling failures w/ decoding

$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus p = 0$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \oplus q = 0$$

Suppose disk1 (d_1) and disk4 (d_4) fail

Step 2: Calculate the left sides, since those all exist.

$$d_0 \oplus d_2 \oplus d_3 \oplus d_5 \oplus p = S_0 = d_1 \oplus d_4$$

$$32d_0 \oplus 8d_2 \oplus 4d_3 \oplus d_5 \oplus q = S_1 = 16d_1 \oplus 2d_4$$

Handling failures w/ decoding

$$d_0 \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus p = 0$$

$$32d_0 \oplus 16d_1 \oplus 8d_2 \oplus 4d_3 \oplus 2d_4 \oplus d_5 \oplus q = 0$$

Suppose disk1 (d_1) and disk4 (d_4) fail

Step 3: Solve using Gaussian Elimination or Matrix Inversion.

$$S_0 = d_1 \oplus d_4$$

$$S_1 = 16d_1 \oplus 2d_4$$

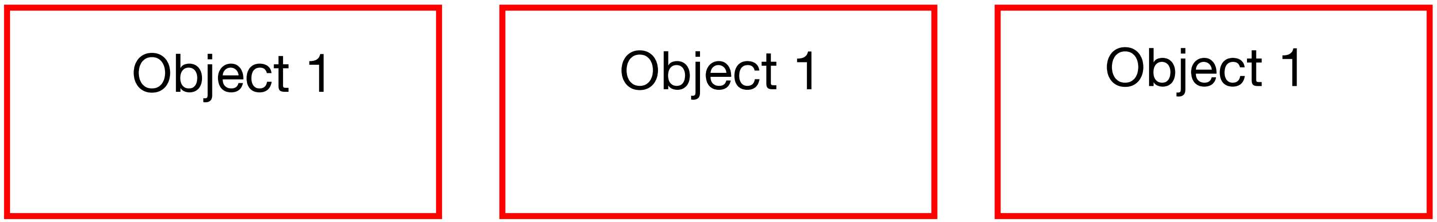
$$\longrightarrow$$

$$d_1 = \frac{(2S_0 \oplus S_1)}{(16 \oplus 2)}$$

$$d_4 = S_0 \oplus d_1$$

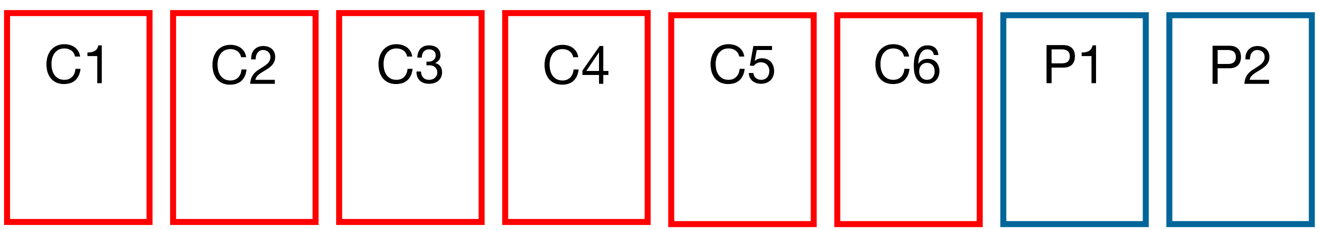
Replication vs. erasure coding

Storage nodes w/ 3-way replication



3-Way replication requires **3X** of the storage space for storing one object.
3-way replication can tolerate **2 failures** at a time.

Storage nodes w/ RS (6,2)



Reed-Solomon (6,2) requires **1.33X** of the storage space for storing one object. ← Storage-efficient redundancy
RS (6,2) can tolerate **2 failures** at a time.