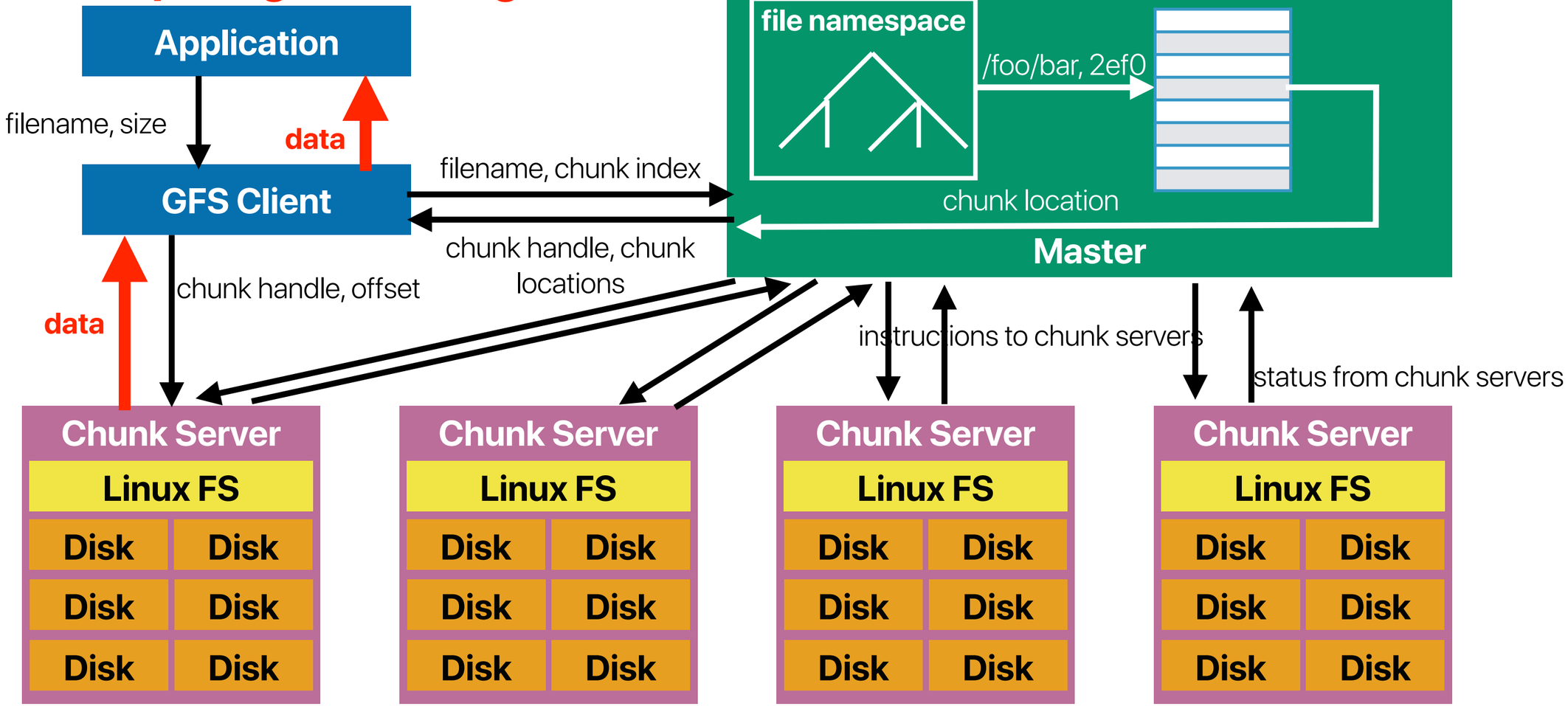
Microsoft Windows Azure Storage, Facebook's Data Storage and Google Search Architecture

Hung-Wei Tseng

Recap: GFS architecture

decoupled data and control paths —
only control path goes through master



load balancing, replicas among chunkservers

Recap: How does GFS achieve its goals?

- Storage based on inexpensive disks that fail frequently — master/chunkserver/client — MapReduce is fault tolerant
- Many large files in contrast to small files for personal data — large chunk size — MapReduce aims at processing large amount of data once
- Primarily reading streams of data — large chunk size — MapReduce reads chunks of large files
- Sequential writes appending to the end of existing files — large chunk size — Output file keep growing as workers keep writing
- Must support multiple concurrent operations — flat structure — MapReduce has thousands of workers simultaneously
- Bandwidth is more critical than latency — large chunk size — MapReduce only wants to finish tasks within "reasonable" amount of time

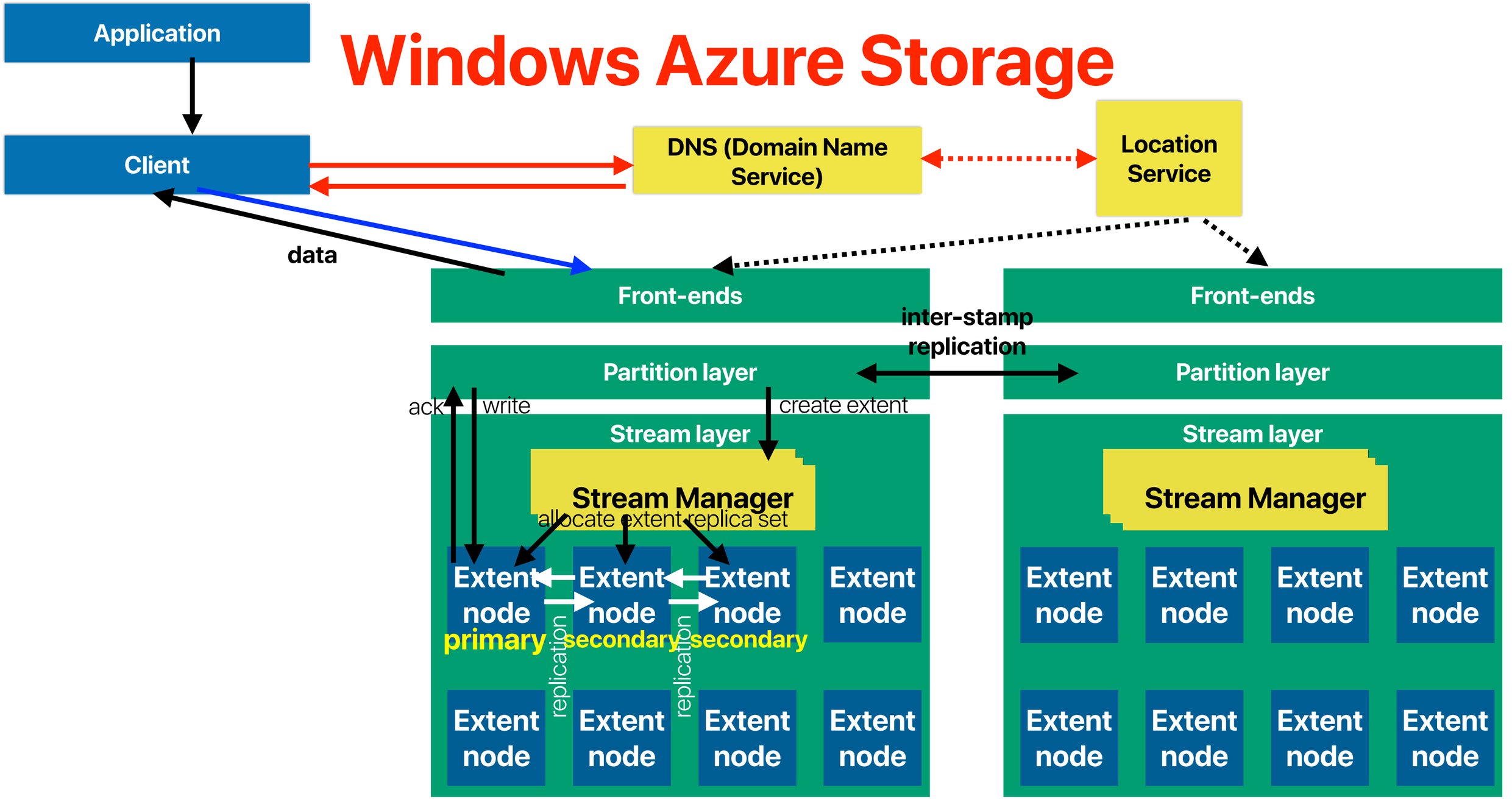
Recap: Why Windows Azure Storage

- Must tolerate many different data abstractions: blobs, tables and queues
- Learning from feedbacks in existing cloud storage
  - Strong consistency
  - Global and scalable namespace/storage
  - Disaster recovery
  - Multi-tenancy and cost of storage

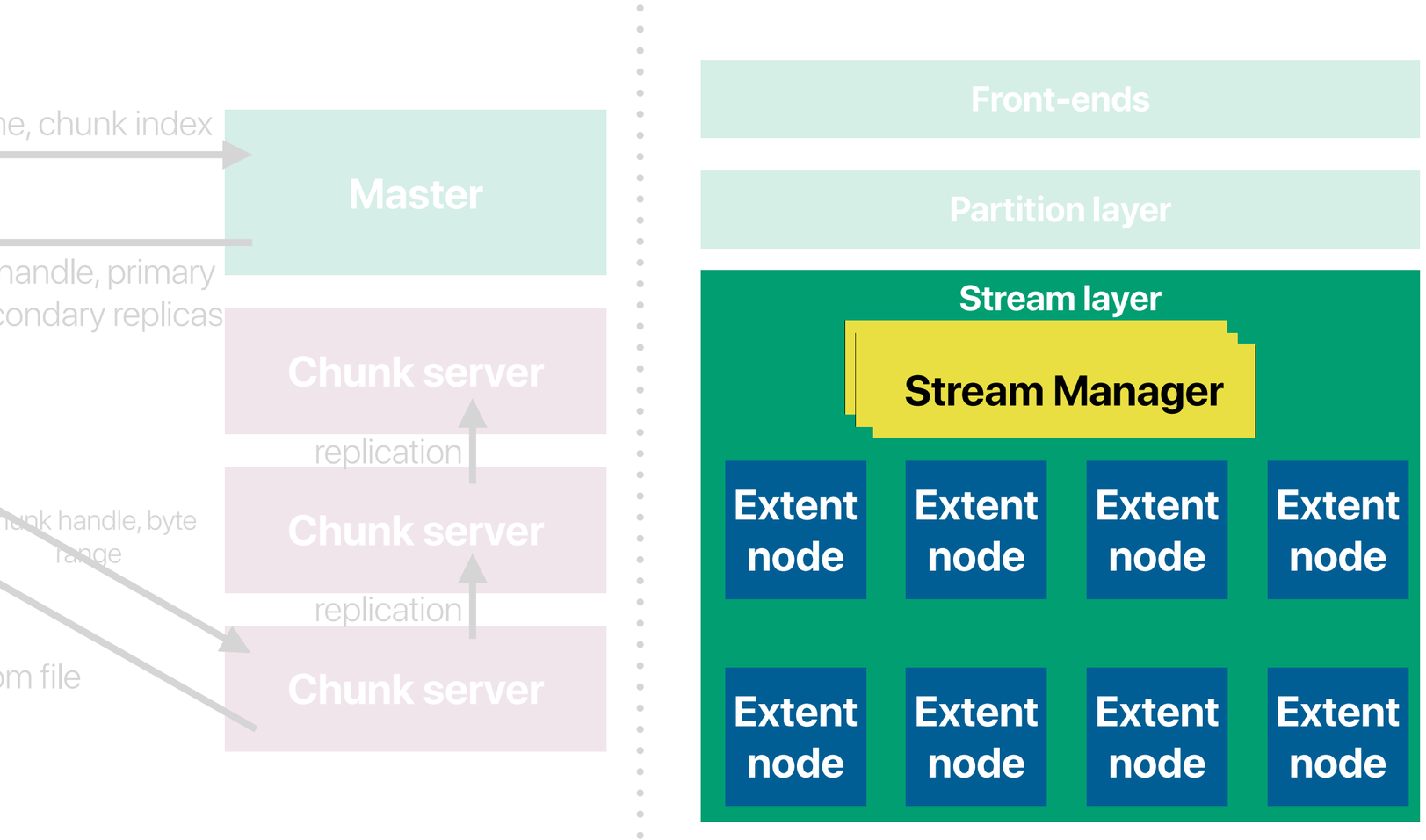
Outline

- Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency
- f4: Facebook's Warm BLOB Storage System
- Google Search

Windows Azure Storage



GFS v.s. stamp in WAS



What is a stream?

- Regarding a stream in WAS, please identify how many of the following statements is/are true
  - ① A stream is a list of extents, in which an extent consists of consecutive blocks
  - ② Each block in the stream contains a checksum to ensure the data integrity
  - ③ An update to a stream can only be appended to the end of the stream
  - ④ Two streams can share the same set of extents
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Similar to an extent-base file system. Shares the same benefits with EXT-based systems

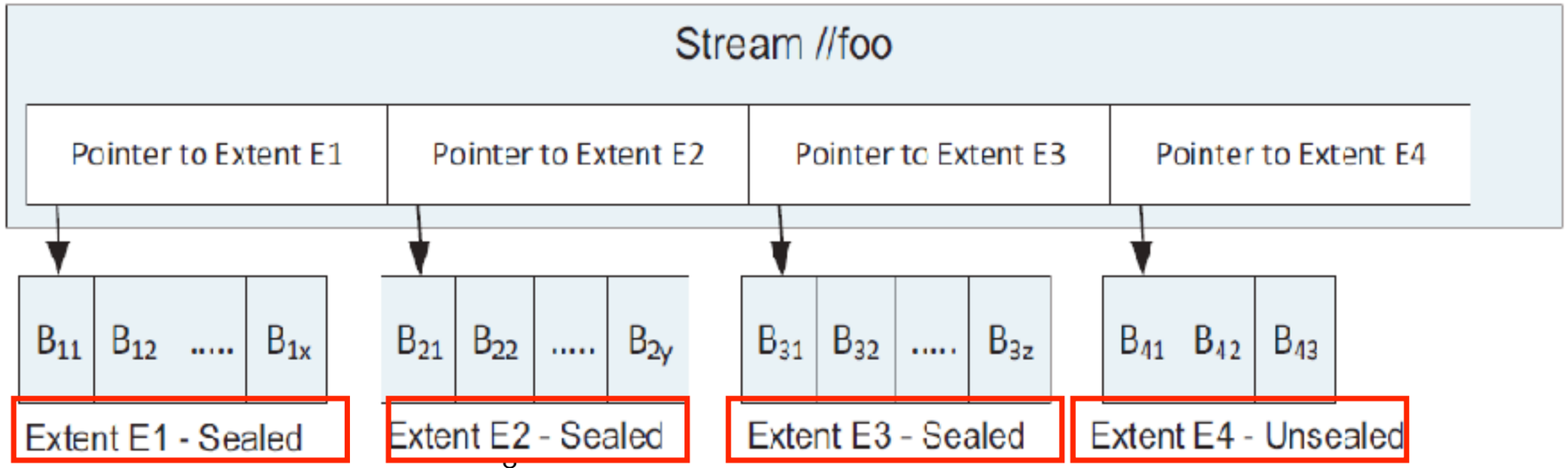
As a result, we need to read a whole block every time.... But not a big issue because ...

Append only, copy-on-write ... (Doesn't this sound familiar?) Improved bandwidth, data locality

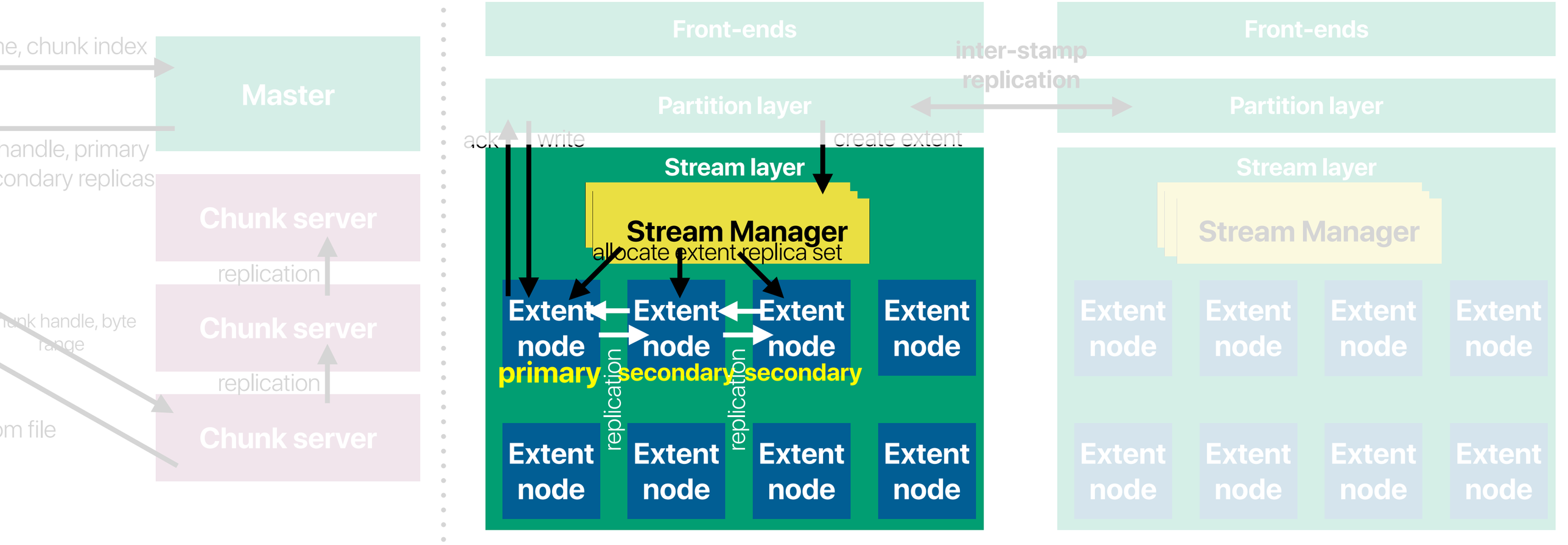
LogFS

Minimize the time when creating a new file

De-duplication to save disk space

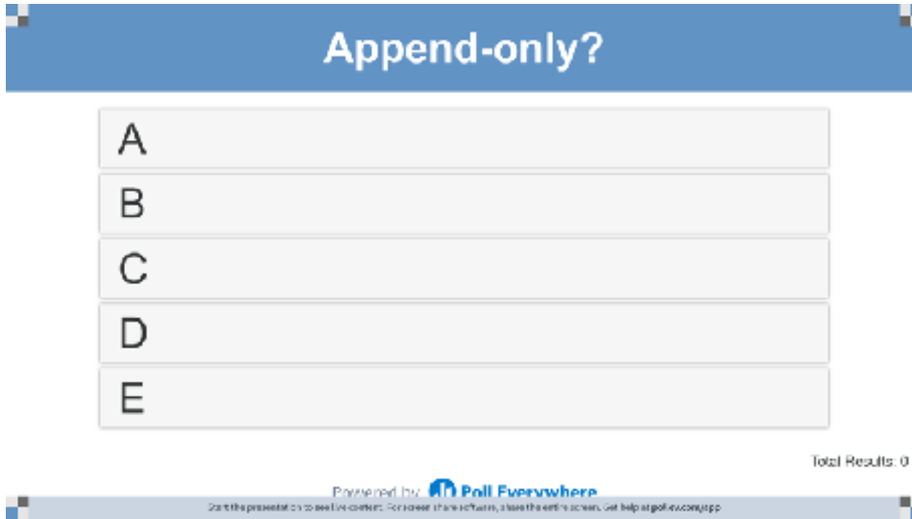


GFS v.s. stamp in WAS



Why “append-only” and “sealing”?

- In WAS, the stream is append only. The stamp will “seal” extents and extents will become immutable once sealed. How many of the following can sealing contribute to?
  - ① Must tolerate many different data abstractions: blobs, tables and queues
  - ② Strong consistency
  - ③ Global and scalable namespace/storage
  - ④ Disaster recovery
  - ⑤ Multi-tenancy and cost of storage
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5



Why “append-only” and “sealing”?

- In WAS, the stream is append only. The stamp will “seal” extents and extents will become immutable once sealed. How many of the following can sealing contribute to?
  - ① Must tolerate many different data abstractions: blobs, tables and queues
  - ✓ ② Strong consistency
  - ③ Global and scalable namespace/storage
  - ✓ ④ Disaster recovery
  - ✓ ⑤ Multi-tenancy and cost of storage
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

2. Once an extent is sealed, any reads from any sealed replica will always see the same contents of the extent.

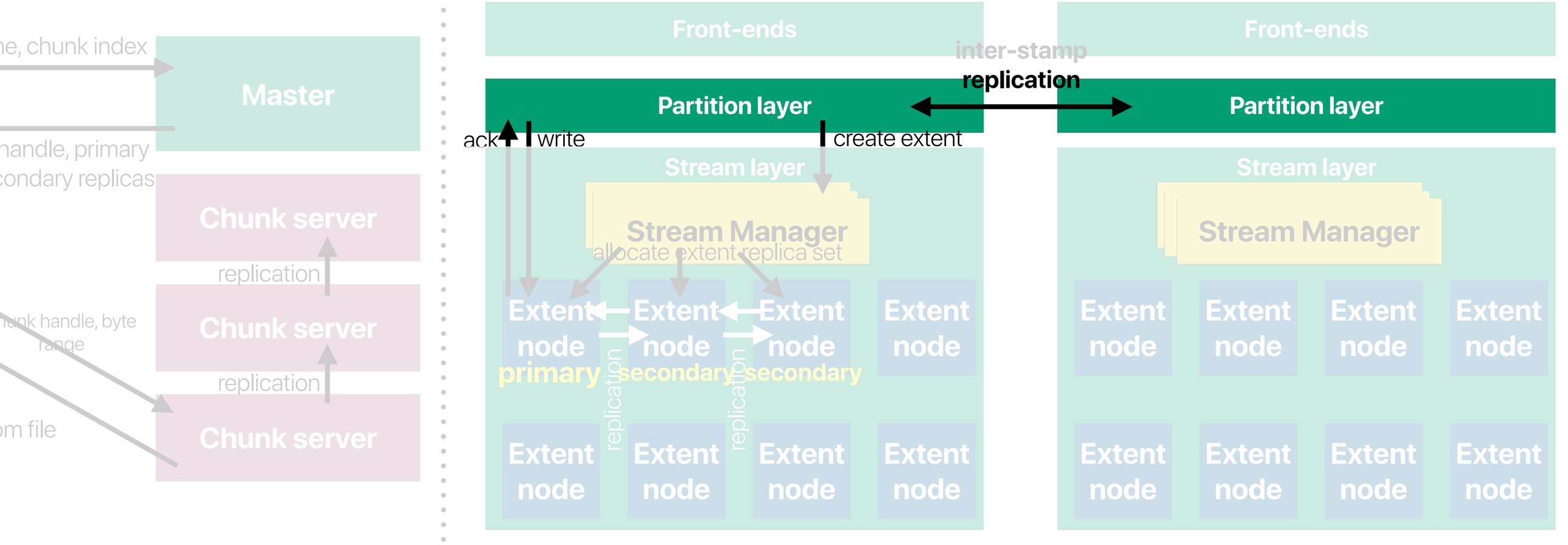
Append-only System – Having an append-only system and sealing an extent upon failure have greatly simplified the replication protocol and handling of failure scenarios. In this

Erasure coding sealed extents is an important optimization, given the amount of data we are storing. It reduces the cost of storing data from three full replicas within a stamp, which is three times the original data, to only 1.3x – 1.5x the original data, depending

Write failure

- Consider the case where 1 of 3 nodes handling a write fails and the current extent is sealed at latest commit boundary (end of extent) — that data will be on failed node
- new extent created
- SM chooses **three** new replicas to store extents
- client retries via new primary among the three new replicas
- failed node, upon restart, will coord w/ SM to synchronize its extent to the commit length decided upon

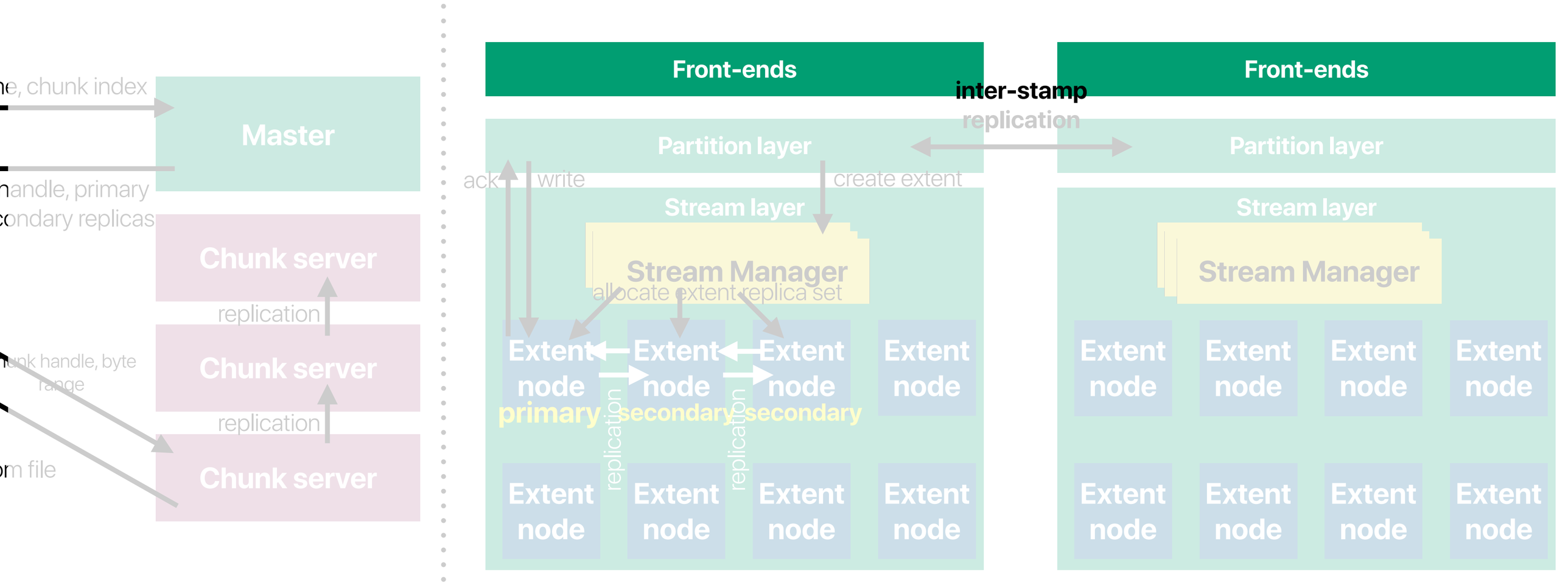
GFS v.s. stamp in WAS



Partition layer

- Managing high-level data abstractions — tolerate many different data abstractions: blobs, tables and queues
- Providing scalable object namespaces — Global and scalable namespace/storage
- Providing transaction ordering and strong consistency for objects — Strong consistency
- Storing object data on top of the stream layer — tolerate many different data abstractions: blobs, tables and queues
- Cache object data to reduce disk I/O
- Inter-stamp data replications — Strong consistency

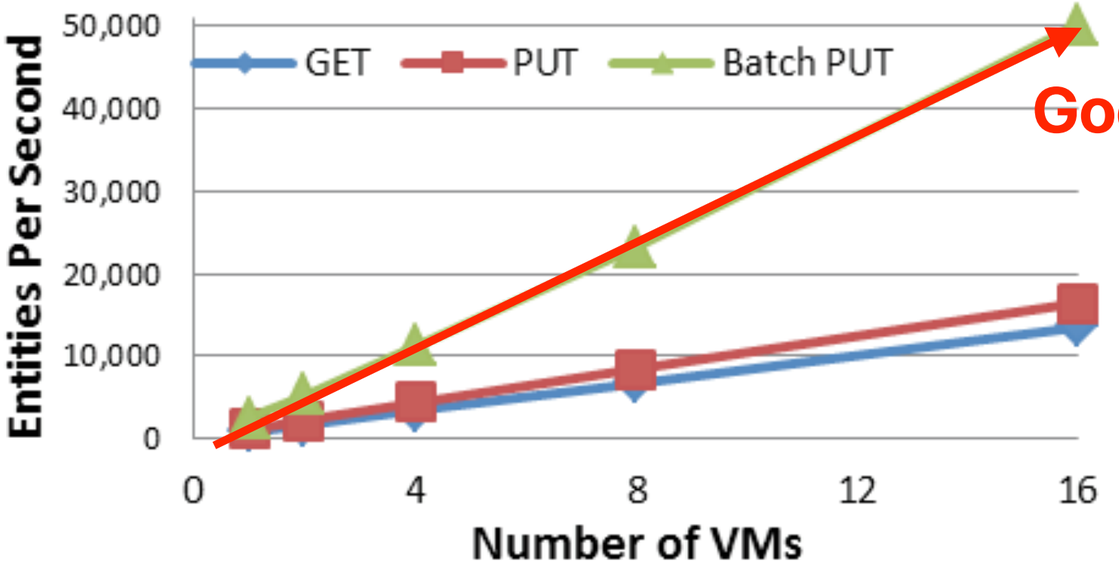
GFS v.s. stamp in WAS



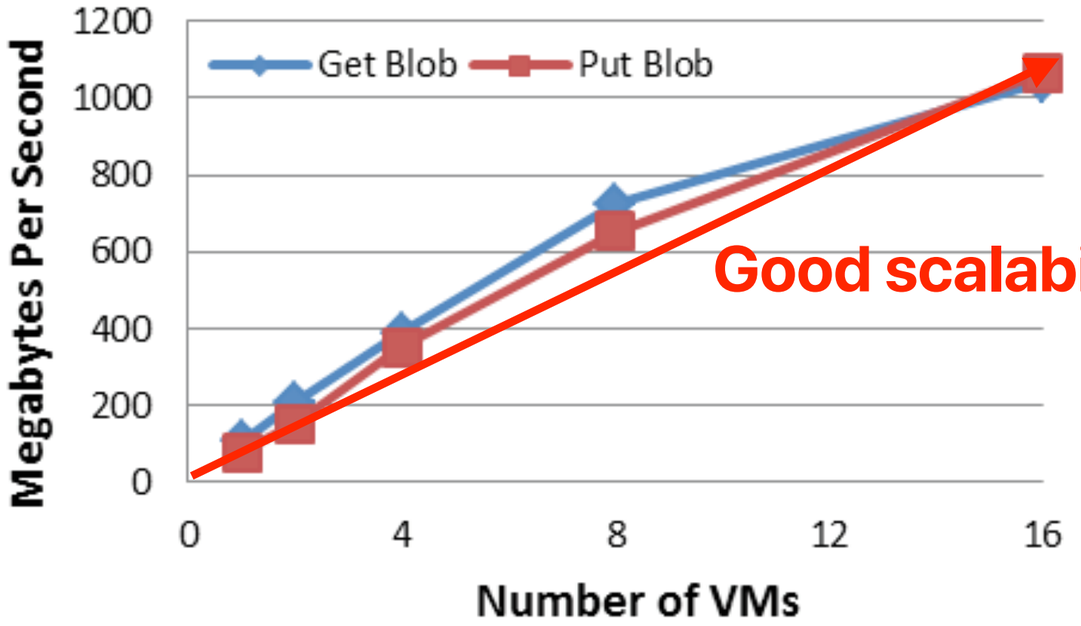
Front-end layer

- A set of **stateless** servers taking incoming requests
  - Think about the benefits of stateless in NFS
- Keep partition maps to forward the request to the right server
  - A stamp can contain 10—20 racks with 18 disk-heavy storage node per rack
- Stream large objects directly from the stream layer and cache frequently accessed data for efficiency

Are they doing well?



Good scalability



Good scalability

GFS v.s. WAS

	GFS (OSDI 2003)	WAS (SOSP 2011)
File organizations	file chunk block	stream extent record
System architecture	master chunkserver	stream manager extent nodes
Data updates		append only updates
Consistency models	relaxed consistency	strong consistency
Data formats	files	multiple types of objects
Replications	intra-cluster replication	geo-replication
Usage of nodes	chunk server can perform both	separate computation and storage

f4: Facebook's Warm BLOB Storage System

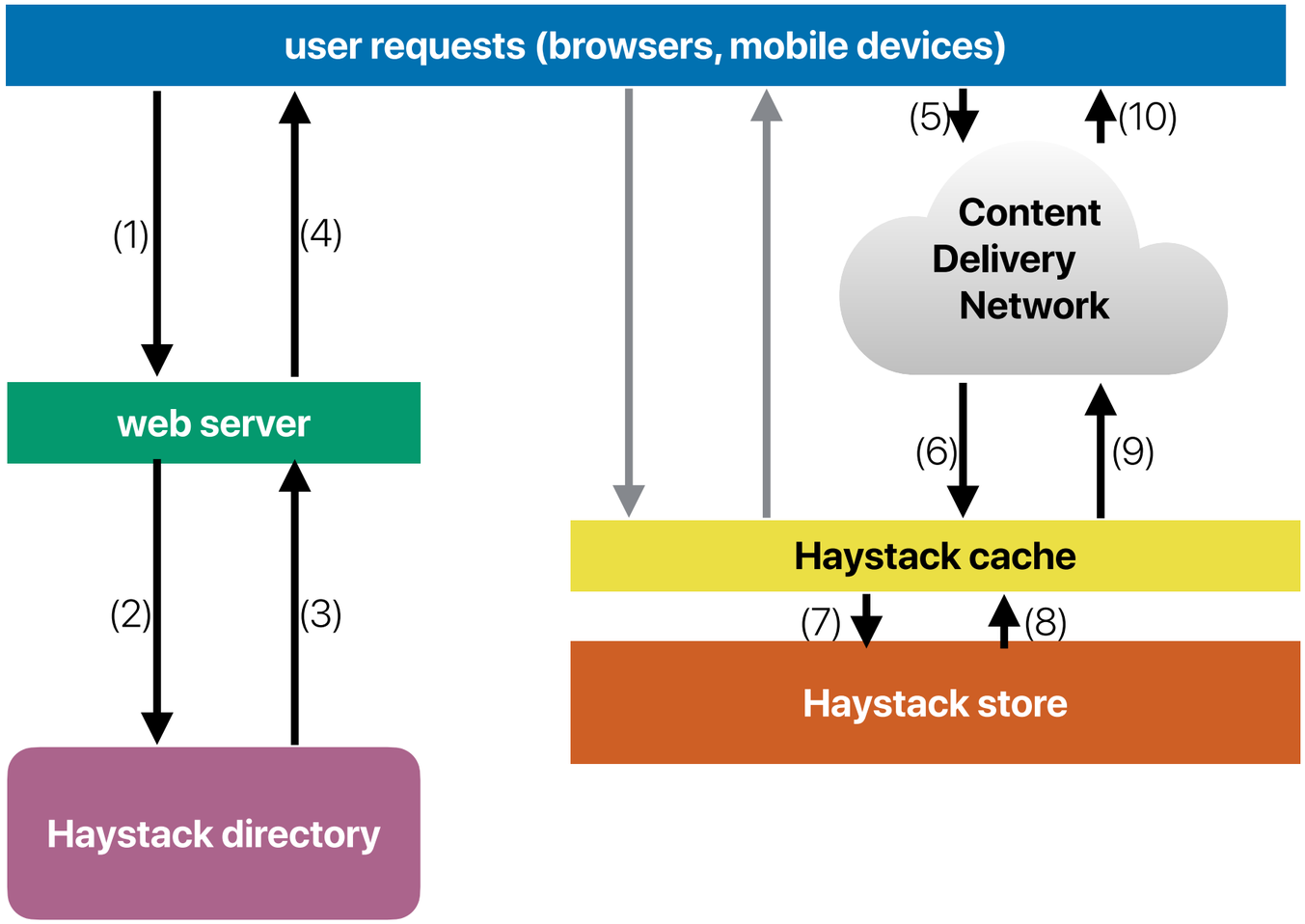
Subramanian Muralidhar, Wyatt Lloyd, Sabyasachi Roy, Cory Hill,
Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar,
Viswanath Sivakumar, Linpeng Tang, and Sanjeev Kumar.

FACEBOOK

The original NFS-based FB storage

- Within a data center with high-speed network, the round-trip latency of network accesses is not really a big deal
- However, the amount of metadata, especially directory metadata, is huge — cannot be cached
- As a result, each file access still requires ~ 10 inode/data requests from disks/network nodes — kill performance

Haystack

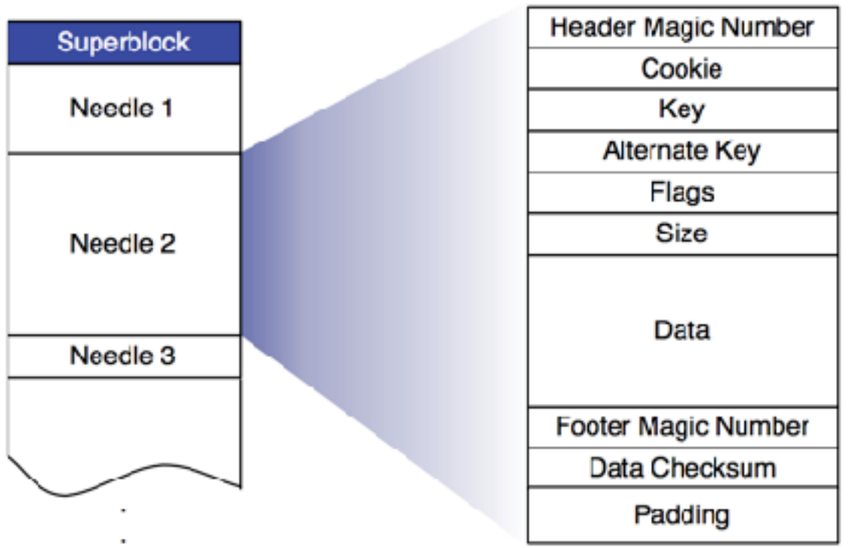


http://<CDN>/<Cache>/<Machine ID>/<Logical volume,Photo>

Finding a needle in Haystack: Facebook's photo storage. Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel. OSDI 2010

Haystack

- Each storage unit provides 10TB of usable space, using RAID-6 — 20% redundancy for parity bits
  - Each storage split into 100 physical volumes (100GB)
  - Physical volumes on different machines grouped into logical volumes
  - A photo saved to a logical volume is written to all corresponding physical volumes — **3** replicas
- Each volume is actually just a large file
  - Needle represents a photo
  - Each needle is identified through the offset
  - Sealed (the same as WAS) one the file reaches 100GB



GFS v.s. WAS

	GFS (OSDI 2003)	Facebook Haystack (OSDI 2010)	WAS (SOSP 2011)
File organizations	file chunk block	volume needle	stream extent record
System architecture	master chunkserver	directory haystack store	stream manager extent nodes
Data updates			append only updates
Consistency models	relaxed consistency		strong consistency
Data formats	files	photo/needle	multiple types of objects
Replications	intra-cluster replication	RIAD-6 & geo-replication	geo-replication
Usage of nodes	chunk server can perform both		separate computation and storage

Why FB wants f4 in addition to Haystack

- Regarding the optimization goals of f4, please identify how many the following statements is/are correct.
  - ① f4 is optimized for the throughput of accessing data
  - ② f4 is optimized for the latency of accessing data
  - ③ f4 is designed to reduce the degree of data replications in the system
  - ④ f4 is designed to tolerate various kind of failure in the system
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Why FB wants f4 in addition to Haystack

- Regarding the optimization goals of f4, please identify how many the following statements is/are correct.
  - ① f4 is optimized for the throughput of accessing data
  - ② f4 is optimized for the latency of accessing data
  - ③ ✓ f4 is designed to reduce the degree of data replications in the system
  - ④ ✓ f4 is designed to tolerate various kind of failure in the system

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Storage Efficiency One of the key goals of our new system is to improve storage efficiency, i.e., reduce the effective-replication-factor while still maintaining a high degree of reliability and performance.

Fault Tolerance Another important goal for our storage system is fault tolerance to a hierarchy of faults to ensure we do not lose data and that storage is always available for client requests. We explicitly consider four types of failures:

What kind of data is f4 optimized for?

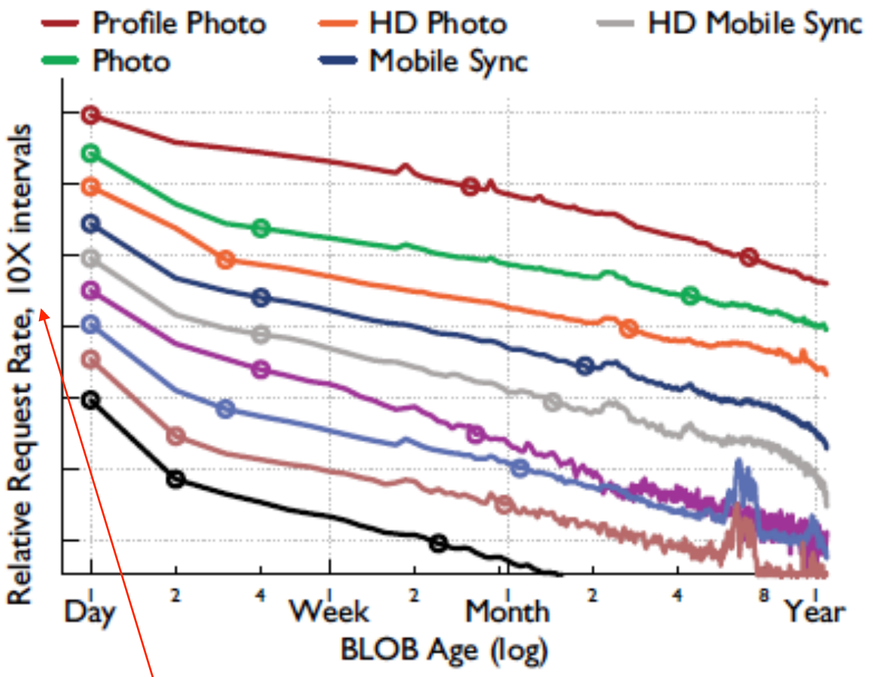
- Regarding the type of data that f4 aims at, please identify how many the following statements is/are correct.
  - ① f4 is optimized for most frequently requested data in Facebook services
  - ② f4 is optimized for frequently created, deleted data
  - ③ f4 is optimized for reducing the access latency of long-term storage
  - ④ f4 is optimized for read-only data
  - ⑤ f4 is optimized for data that are not accessed very frequently
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

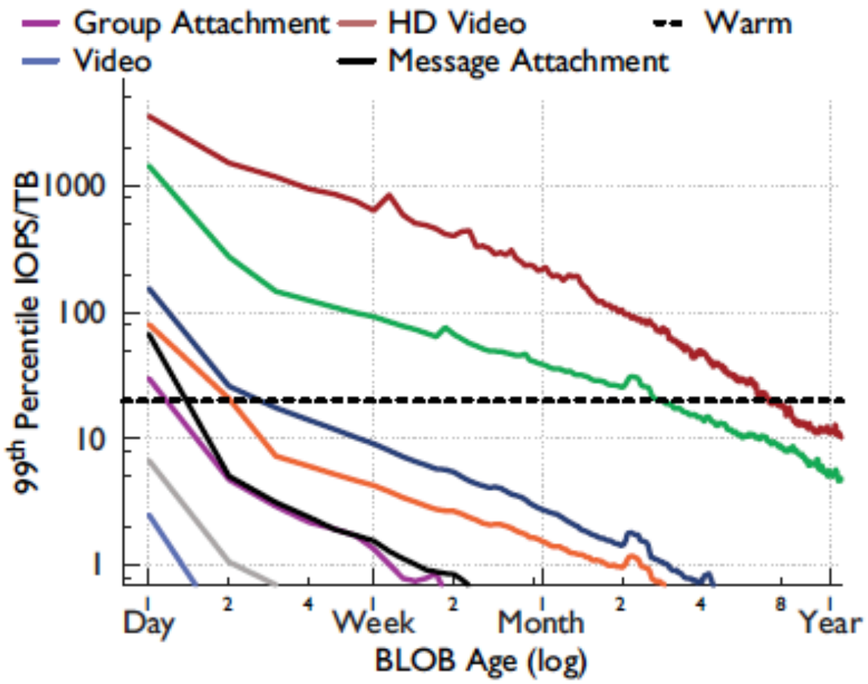
What kind of data is f4 optimized for?

- Regarding the type of data that f4 aims at, please identify how many the following statements is/are correct.
  - ① f4 is optimized for most frequently requested data in Facebook services
  - ② f4 is optimized for frequently created, deleted data
  - ③ f4 is optimized for reducing the access latency of long-term storage
  - ④ f4 is optimized for read-only data
  - ⑤ f4 is optimized for data that are not accessed very frequently

  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

"Temperature" of data



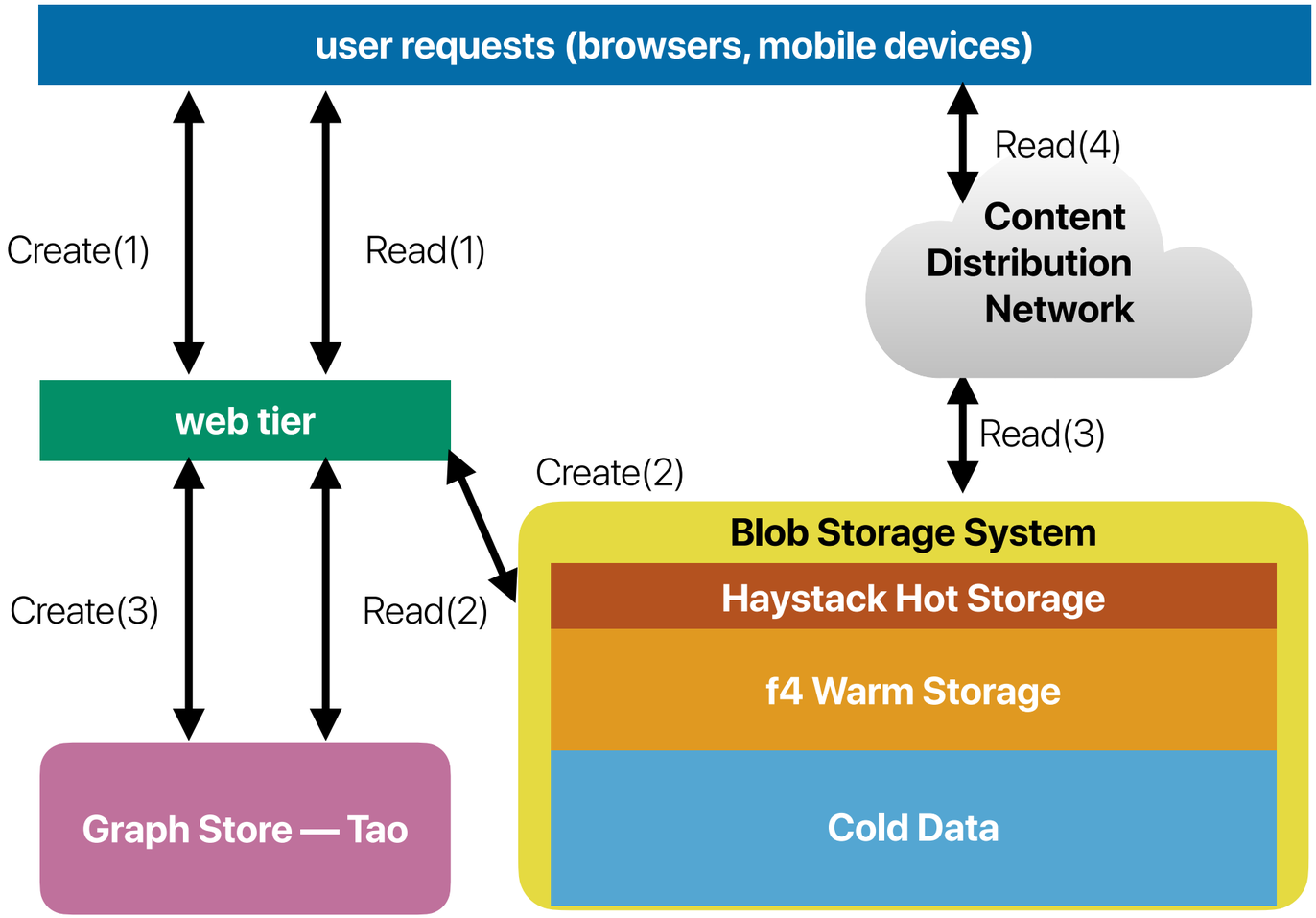


log scale — not encouraged to graph like this if you're writing a technical document or scientific paper

“Temperature” of data

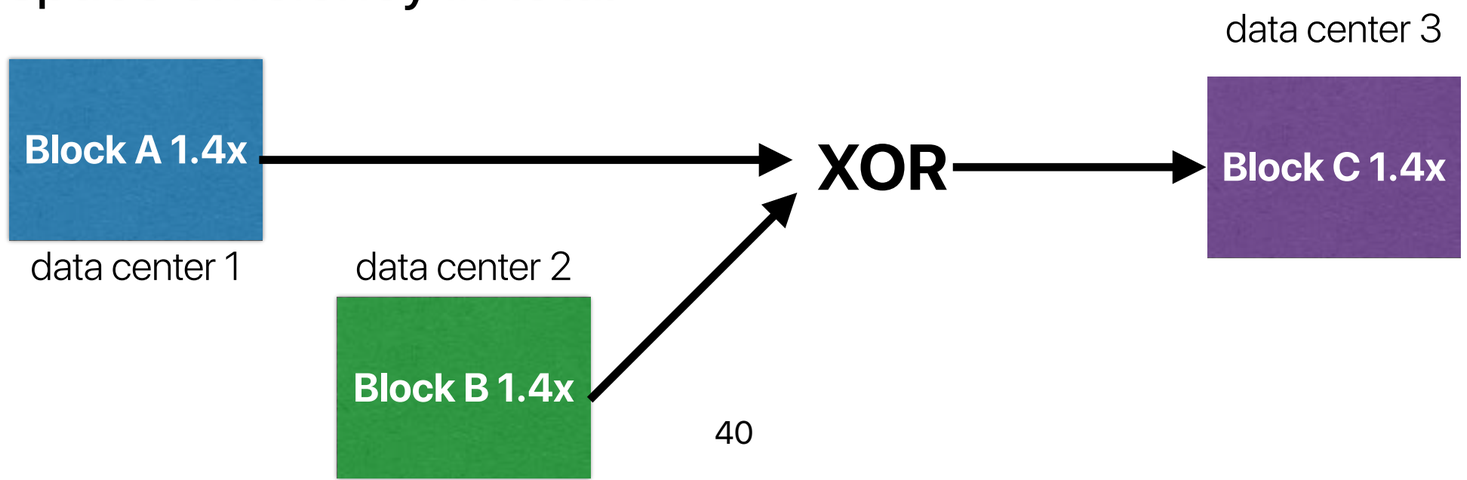
	Hot	Warm	Cold
Access Frequency	Most frequent	Less frequent	Rare
Pattern	Created often, delete often	Not so frequently read Not so frequently deleted Maybe read-only	Long-term storage, usually takes hours to retrieve
Size		65PB in 2014 and growing rapidly	

Facebook storage architecture



Storage efficiency

- Reed-Solomon erasure coding
  - Strips: 10GB data + 4GB parity — 1.4x space efficiency
  - One volume contains 10 strips
- XOR Geo-replication
  - Use XOR to reduce overhead further (e.g., Azure makes full copies)
  - Block A in DC1 + block B in DC2 -> parity block P in DC3
  - Any two blocks can be used to generate the third
  - 1.5x space efficiency
- 1.4*1.5 = 2.1x space efficiency in total



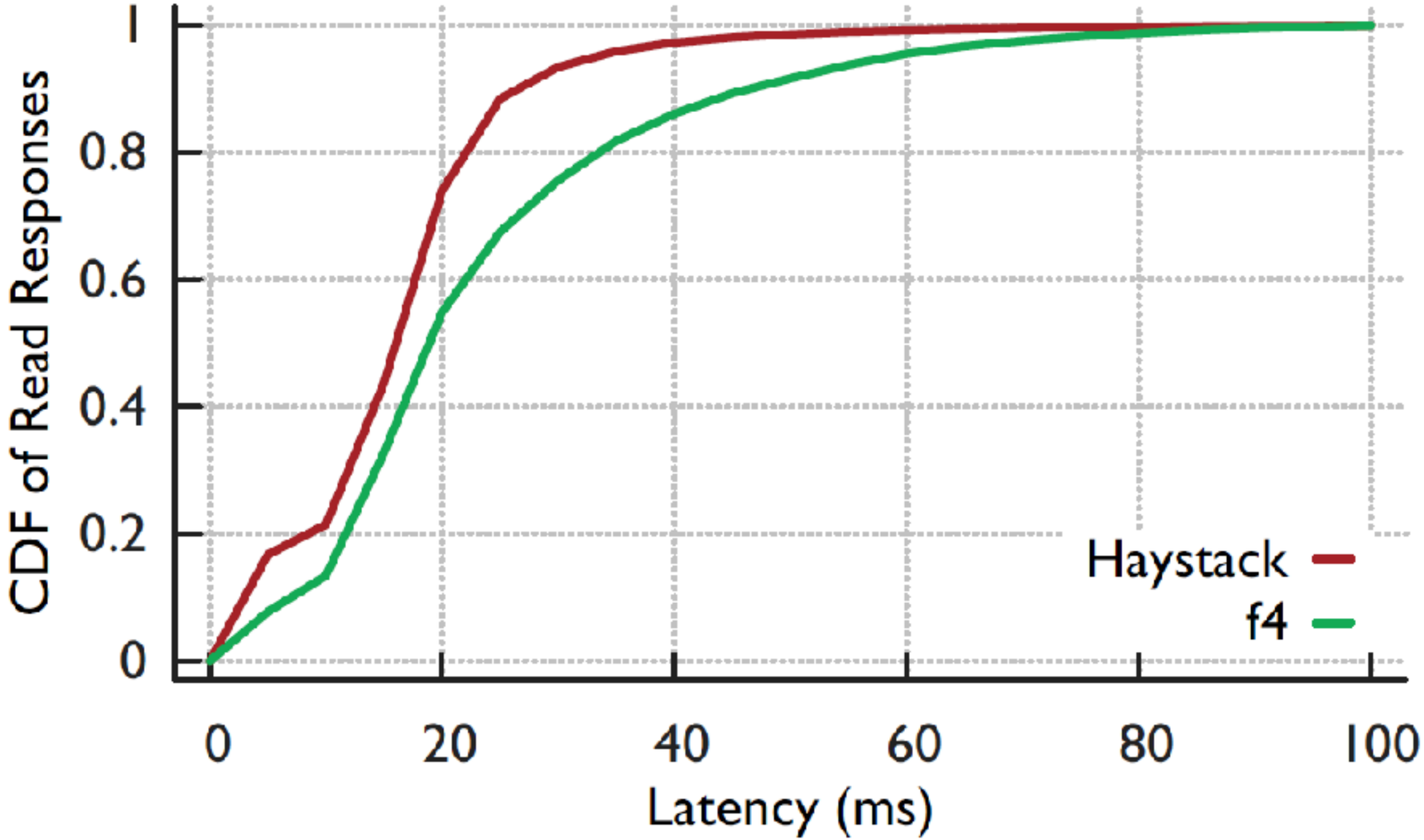
Fault tolerance

- 1%-2% HDD fail in a year
  - replicate data across multiple disks
  - Use erasure coding for storage efficiency
    - n blocks -> n + k blocks, can tolerate k simultaneous failures
    - higher cost for recovering data when there is a failure
- Host failures (periodically)
  - replicate coded blocks on different hosts
- Rack failures (multiple times/year)
  - replicate coded blocks on different racks
- Datacenter failures (rare, but catastrophic)
  - replicate blocks across data centers
  - use XOR to reduce overhead further (e.g., Azure makes full copies)
    - block A in DC1 + block B in DC2 -> parity block P in DC3
    - any two blocks can be used to generate the third
- Index files
  - use normal triple replication (tiny, little benefit in coding them)

What happens if fault occurs?

- Drive fails
  - Reconstruct blocks on another drive
  - Heavy disk, Network, CPU operation
  - one in background
- During failure, may need to reconstruct data online
  - rebuilder node reads BLOB from data + parity, reconstructs
  - only reads + reconstructs the BLOB (40KB), not the entire block (1GB)

Performance of f4



Cells

- Each cell contains 14 racks of 15 hosts, each host contains 30 4TB H.D.Ds.
- A unit of acquisition, deployment
- Storage for a set of volumes
- Similar to the idea of stamps

Announcement

- Project due this Thursday
  - Please submit your current progress
  - You will have another week of "revision period"
    - Allows you to revise your project with 30% of penalty on the unsatisfactory parts/test cases after the first-round of grading (firm deadline 3/11)
    - Say you got only 60% in the first-round, and you fixed everything before 3/11 — you can still get 60% +70%*40% = 88%
- "Very last" reading quiz due next Tuesday
- iEVAL — count as an extra, full-credit reading quiz
- Final — contains two parts (each account for 50%)
  - Part 1: unlimited time between 3/11-3/17, open-ended questions
  - Part 2: 80 minute multiple choices/answers questions + two problem sets of comprehensive exam questions
  - Will release some practice for time-limited part next week — please don't discuss with other people than TA and me

Computer
Science &
Engineering

202

つづく