Cloud storage (II) — Google (cont.), Microsoft

Hung-Wei Tseng

Recap: GFS: Why?

- Conventional file systems do not fit the demand of data centers
- Workloads in data centers are different from conventional computers
  - Storage based on inexpensive disks that fail frequently
  - Many large files in contrast to small files for personal data
  - Primarily reading streams of data
  - Sequential writes appending to the end of existing files
  - Must support multiple concurrent operations
  - Bandwidth is more critical than latency

Recap: Data-center workloads for GFS

- Google Search (Web Search for a Planet: The Google Cluster Architecture, IEEE Micro, vol. 23, 2003)
- MapReduce (MapReduce: Simplified Data Processing on Large Clusters, OSDI 2004)
    - Large-scale machine learning problems
    - Extraction of user data for popular queries
    - extraction of properties of web pages for new experiments and products
    - large-scale graph computations
- BigTable (Bigtable: A Distributed Storage System for Structured Data, OSDI 2006)
    - Google analytics
    - Google earth
    - Personalized search

Recap: What GFS proposes?

- Maintaining the same interface
  - The same function calls
  - The same hierarchical directory/files
- Large chunks — Files are decomposed into large chunks (e.g. 64MB) with replicas
- Flat structure — Hierarchical namespace implemented with flat structure
- Architecture — Master/chunkservers/clients

Recap: Large Chunks

- How many of the following datacenter characteristics can large chunks help address?

  ① Storage based on inexpensive disks that fail frequently

  [✓ ②] Many large files in contrast to small files for personal data

  [✓ ③] Primarily reading streams of data

  [✓ ④] Sequential writes appending to the end of existing files

  ⑤ Must support multiple concurrent operations

  [✓ ⑥] Bandwidth is more critical than latency

  A. 1

  B. 2

  C. 3

  [✓ D. 4]

  E. 5

Current scoreboard

Red	Blue
21	19

Outline

- Google File System (cont.)
- Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency
- f4: Facebook's Warm BLOB Storage System

Flat file system structure

- How many of the following statements can flat file system structure help address?

  ① Storage based on inexpensive disks that fail frequently

  ② Many large files in contrast to small files for personal data

  ③ Primarily reading streams of data

  ④ Sequential writes appending to the end of existing files

  ✓⑤ Must support multiple concurrent operations

  fine-grained locking to reduce the chance of conflicts — you don't have to lock the whole path when acces

  ⑥ Bandwidth is more critical than latency

  Actually improve both.

  [✓ A. 1]

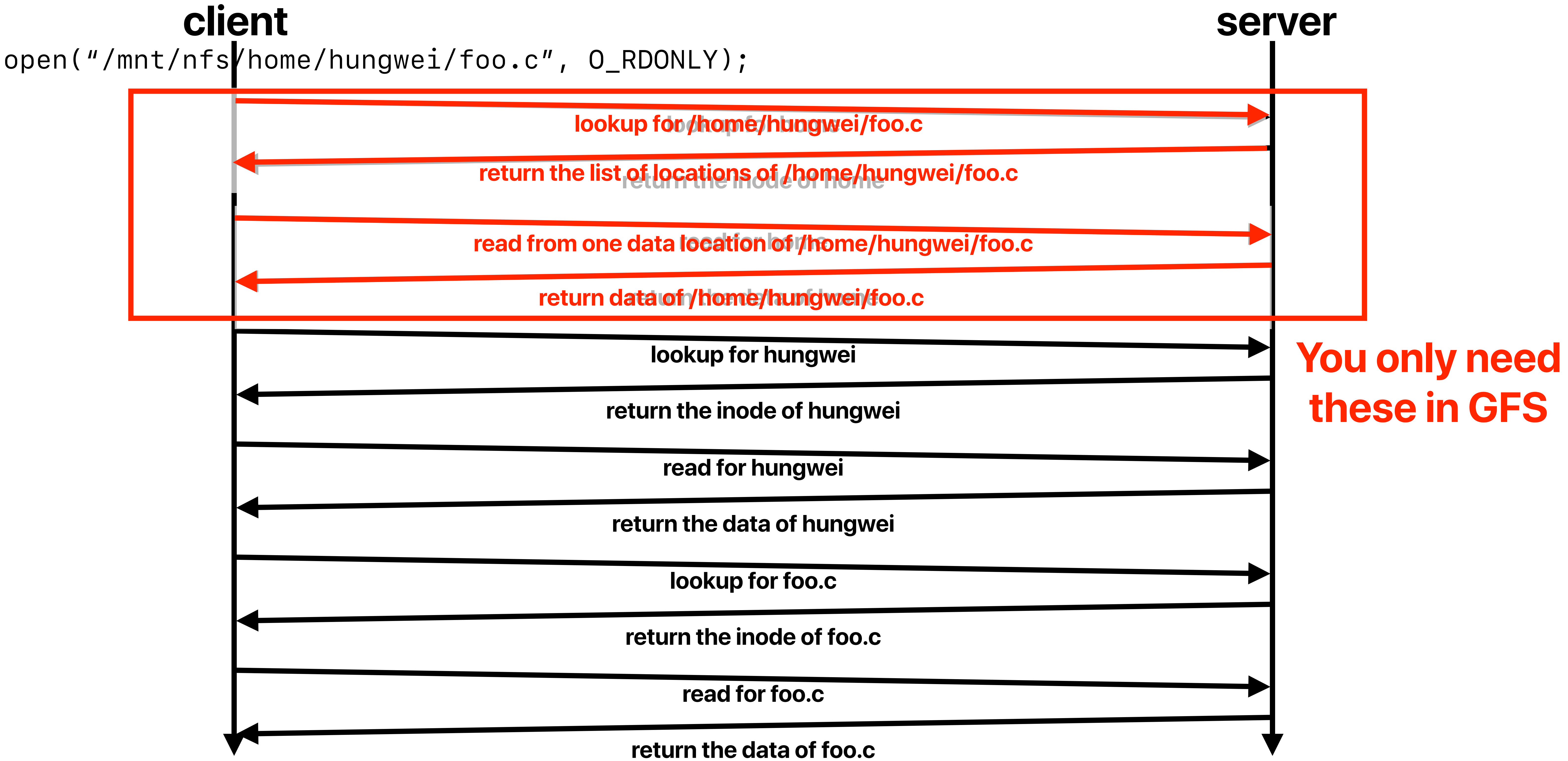
  B. 2

  C. 3

  D. 4

  E. 5

How open works with NFS



client
server
open("/mnt/nfs/home/hungwei/foo.c", O_RDONLY);
lookup for /home/hungwei/foo.c
return the list of locations of /home/hungwei/foo.c
read from one data location of /home/hungwei/foo.c
return data of /home/hungwei/foo.c
lookup for hungwei
return the inode of hungwei
read for hungwei
return the data of hungwei
lookup for foo.c
return the inode of foo.c
read for foo.c
return the data of foo.c

You only need these in GFS

How does GFS achieve its goals?

- Storage based on inexpensive disks that fail frequently — ???

Master/chunkserver architecture

- Many large files in contrast to small files for personal data — large chunk size
- Primarily reading streams of data — large chunk size
- Sequential writes appending to the end of existing files — large chunk size
- Must support multiple concurrent operations — flat structure
- Bandwidth is more critical than latency — large chunk size

Poll close in 1:30

GFS architecture

- Regarding the GFS architecture, how many of the following statements are correct?
  - ① The GFS cluster in the paper only has one active server to store and manipulate metadata
  - ② The chunkserver in GFS may contain data that can also be found on another chunkserver
  - ③ The chunkserver is dedicated for data storage and may not be used for other purpose
  - ④ The client can cache file data to improve performance

  A. 0

  B. 1

  C. 2

  D. 3

  E. 4

Poll close in 1:30

GFS architecture

- Regarding the GFS architecture, how many of the following statements are correct?
  - ① The GFS cluster in the paper only has one active server to store and manipulate metadata
  - ② The chunkserver in GFS may contain data that can also be found on another chunkserver
  - ③ The chunkserver is dedicated for data storage and may not be used for other purpose
  - ④ The client can cache file data to improve performance

  A. 0

  B. 1

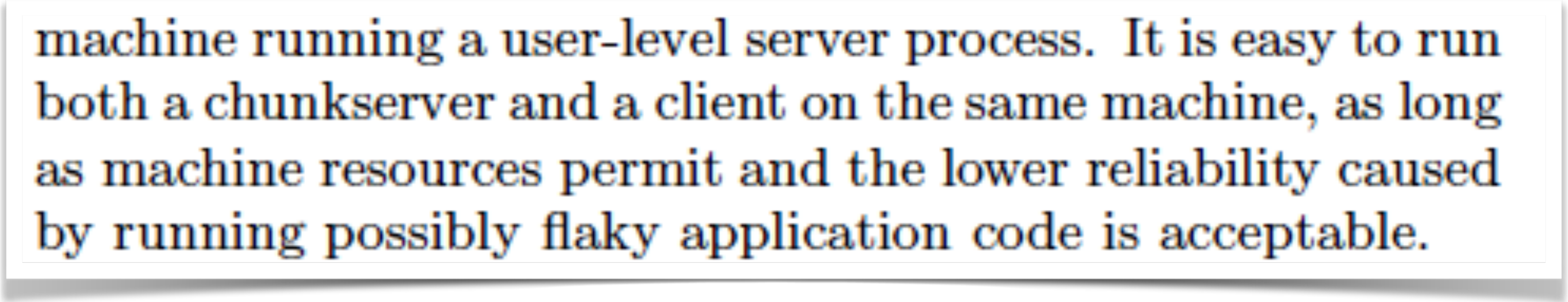
  C. 2

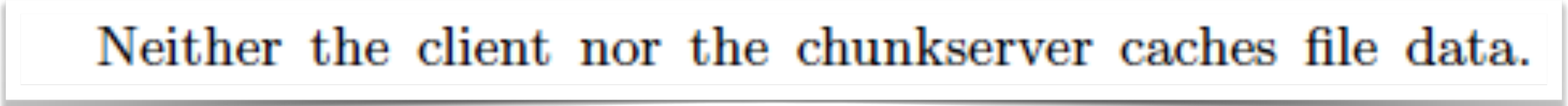
  D. 3

  E. 4

GFS architecture

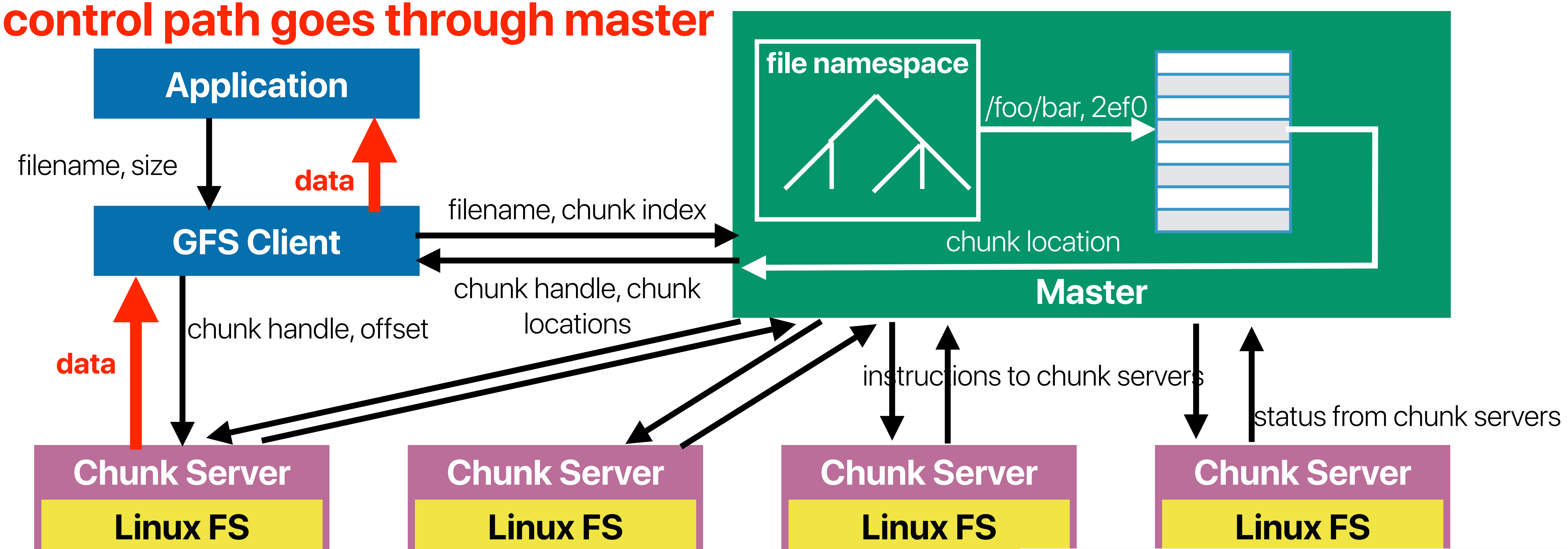
- Regarding the GFS architecture, how many of the following statements are correct?
  - ① The GFS cluster in the paper only has one active server to store and manipulate metadata — single failure point. They have shadow masters
  - ② The chunkserver in GFS may contain data that can also be found on another chunkserver — 3 replicas by default
  - ✗ The chunkserver is dedicated for data storage and may not be used for other purpose — improve the machine utilization — saving money!
  - ✗ The client can cache file data to improve performance — simplify the design
  - A. 0
  - B. 1
  - [✓ C. 2]
  - D. 3
  - E. 4

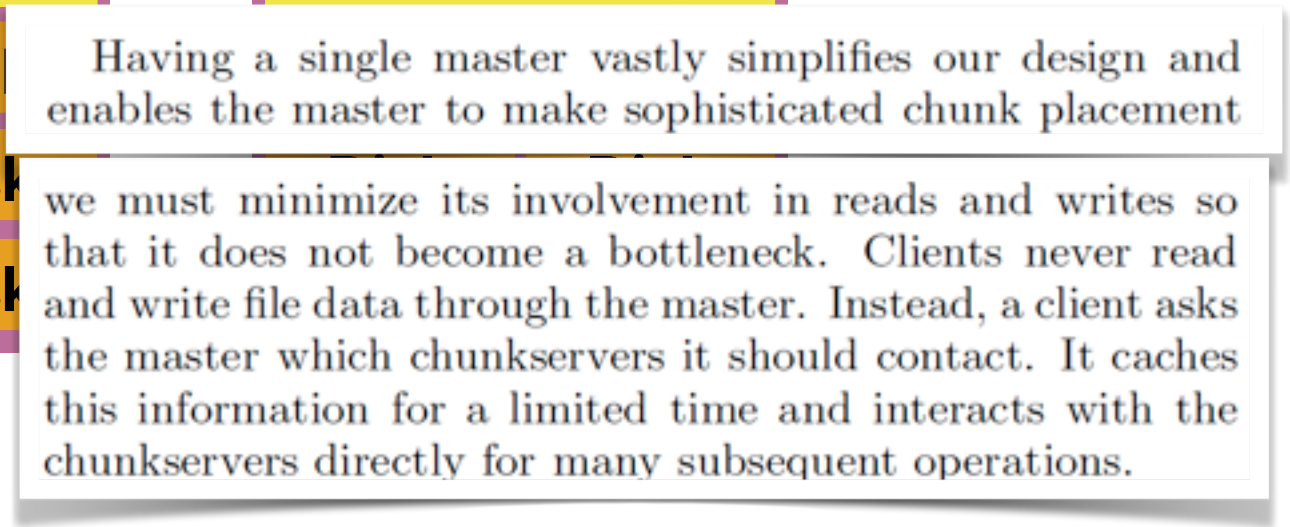

machine running a user-level server process. It is easy to run both a chunkserver and a client on the same machine, as long as machine resources permit and the lower reliability caused by running possibly flaky application code is acceptable.


Neither the client nor the chunkserver caches file data.

GFS Architecture

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





Having a single master vastly simplifies our design and enables the master to make sophisticated chunk placement

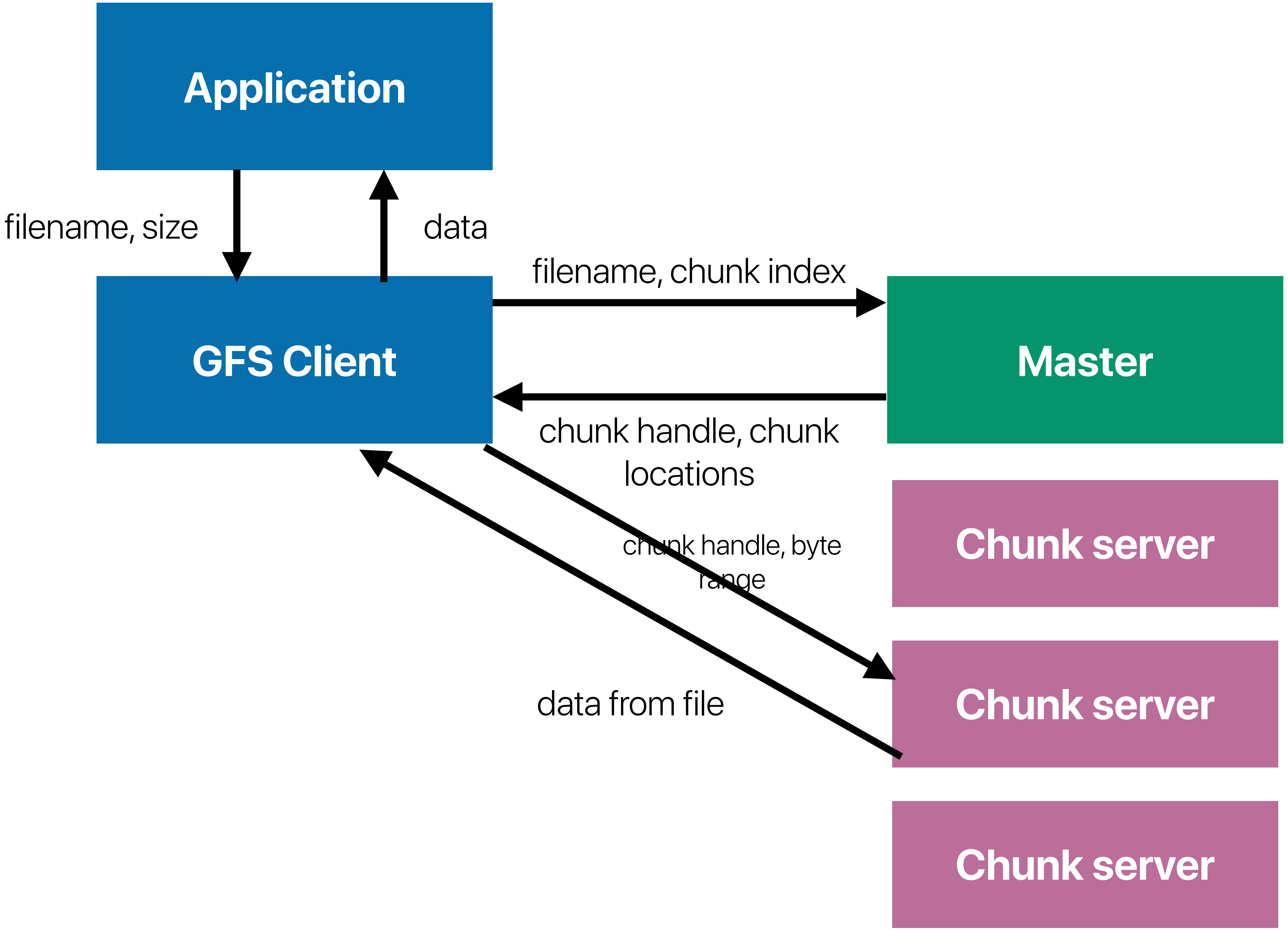
we must minimize its involvement in reads and writes so that it does not become a bottleneck. Clients never read and write file data through the master. Instead, a client asks the master which chunkservers it should contact. It caches this information for a limited time and interacts with the chunkservers directly for many subsequent operations.

load balancing, replicas among chunkservers

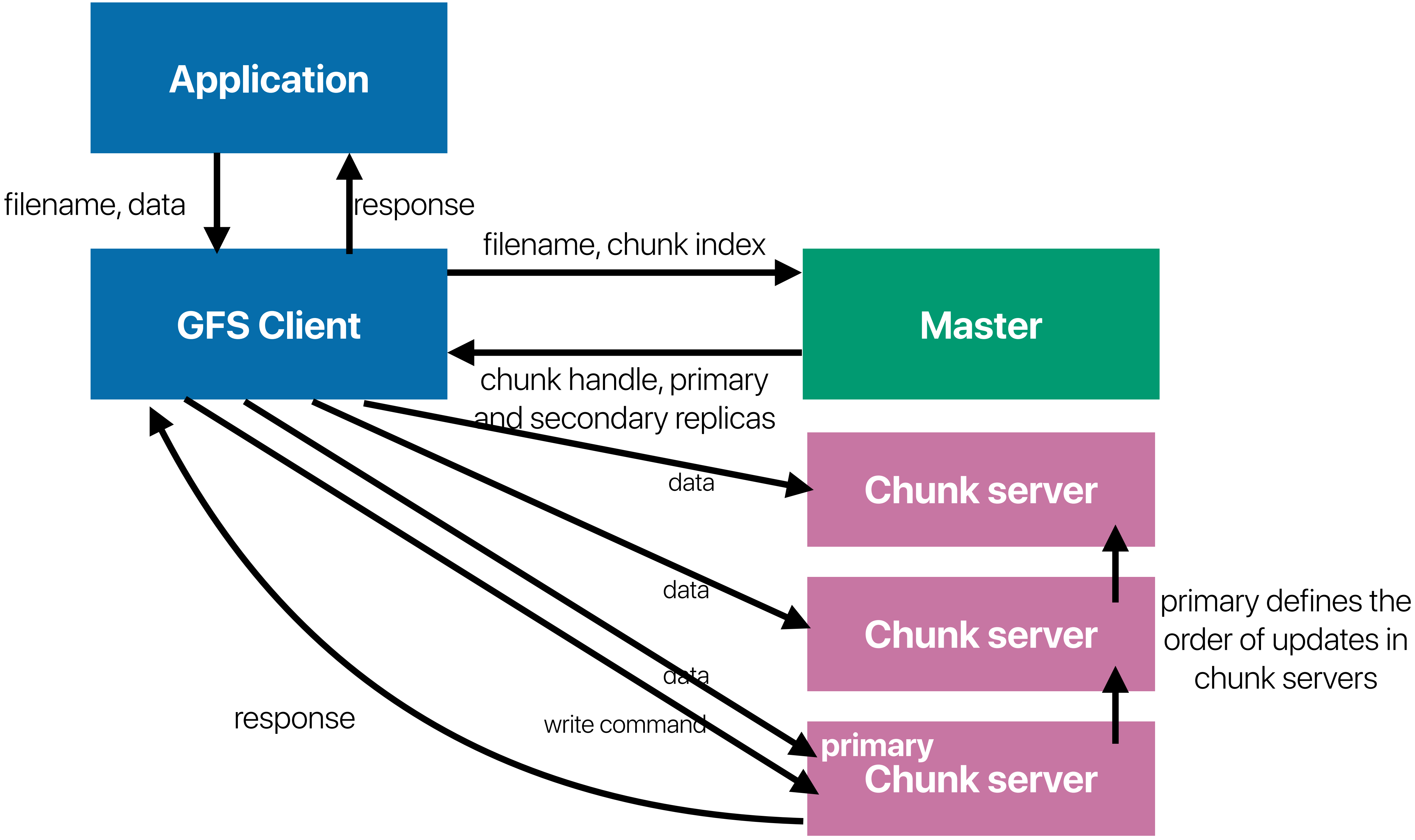
Distributed architecture

- Single master
  - maintains file system metadata including namespace, mapping, access control and chunk locations.
  - controls system wide activities including garbage collection and chunk migration.
- Chunkserver
  - stores data chunks
  - chunks are replicated to improve reliability (3 replicas)
- Client
  - APIs to interact with applications
  - interacts with masters for control operations
  - interacts with chunkservers for accessing data
  - Can run on chunkservers

Reading data in GFS



Writing data in GFS



GFS: Relaxed Consistency model

- Distributed, simple, efficient
- Filename/metadata updates/creates are atomic
- Consistency modes

	Write — write to a specific offset	**Append — write to the end of a file**
Serial success	Defined	Defined with interspersed with inconsistent
Concurrent success	Consistent but undefined	
Failure	inconsistent	

  - Consistent: all replicas have the same value
  - Defined: replica reflects the mutation, consistent
- Applications need to deal with inconsistent cases themselves

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat
Google

MapReduce

- Regarding MapReduce, please identify the number of correct statements from the following
  - [✓ ①] Each worker in MapReduce reads MBs of chunks from an input file
  - [✓ ②] Each worker works independently but does not share data
  - [✓ ③] Each worker does not overwrite its input in general
  - [✓ ④] MapReduce aims at computation throughputs, instead of latency

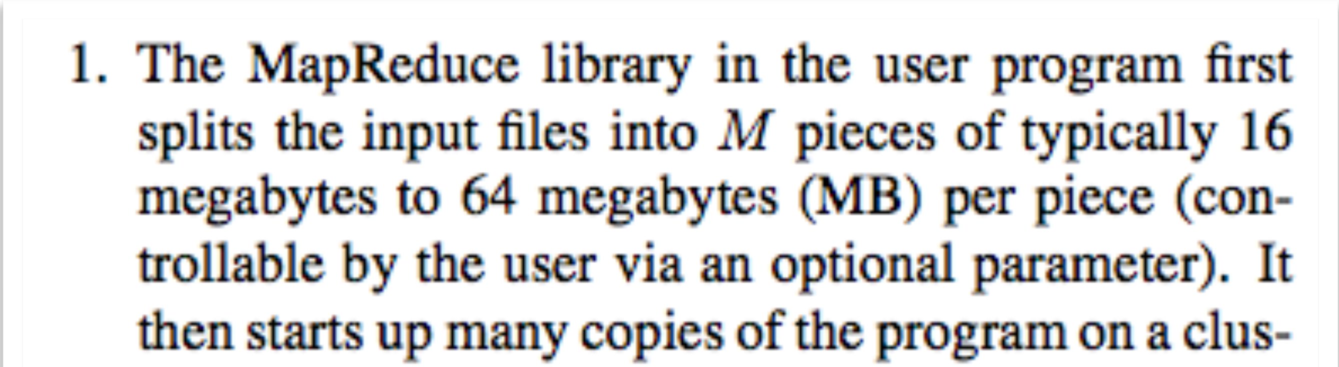
A. 0

B. 1

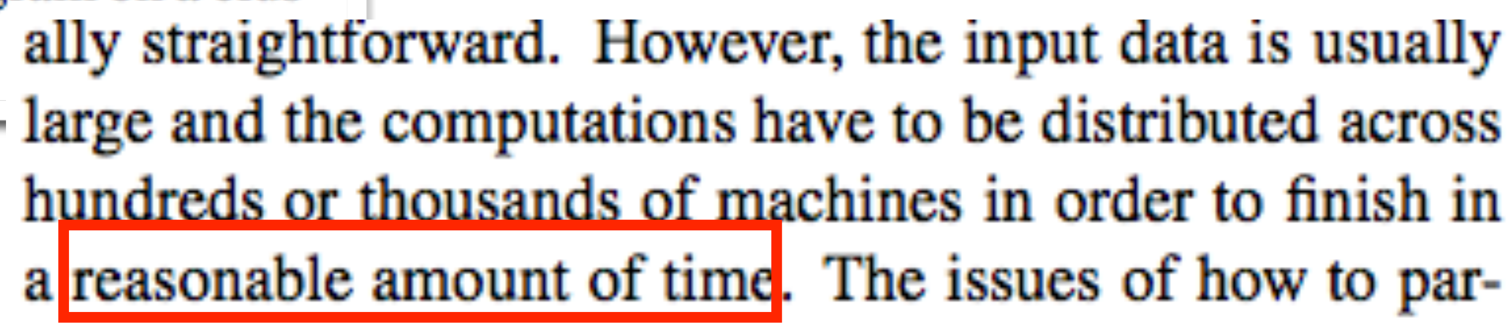
C. 2

D. 3

[✓ E. 4]

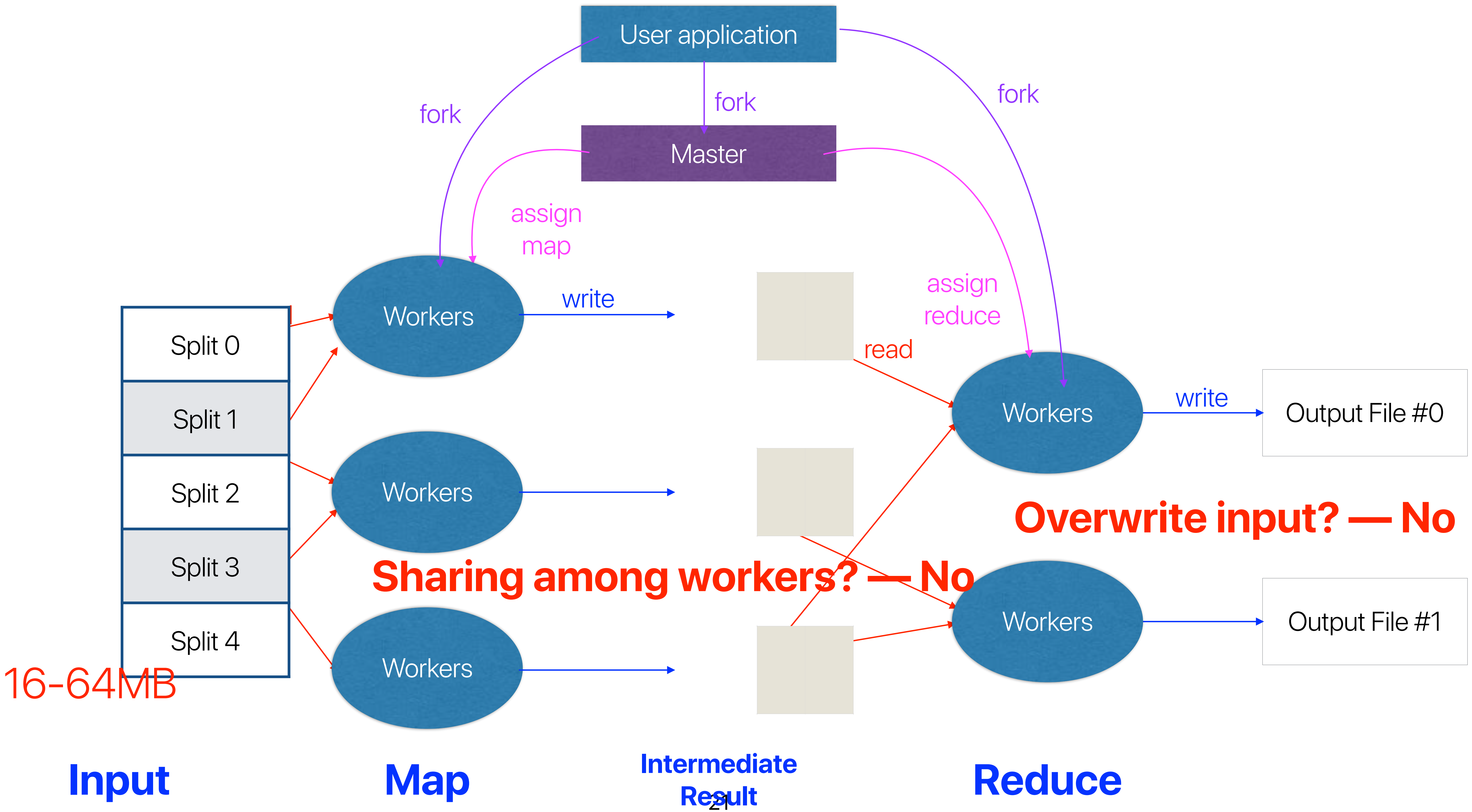


1. The MapReduce library in the user program first splits the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece (controllable by the user via an optional parameter). It then starts up many copies of the program on a cluster of machines.



ally straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to par-

MapReduce



16-64MB

Overwrite input? — No

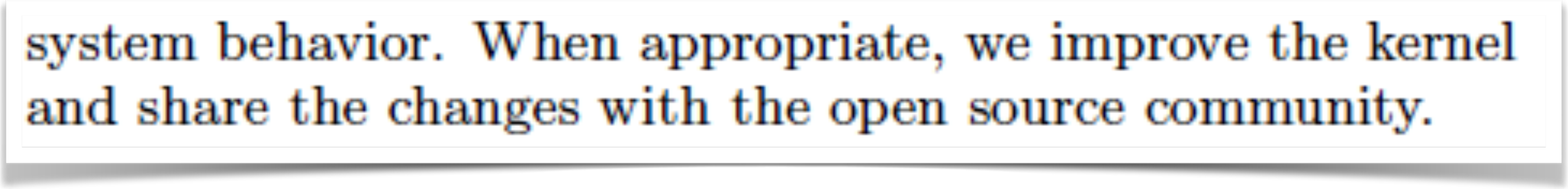
Sharing among workers? — No

Why we care about GFS

- Conventional file systems do not fit the demand of data centers
- Workloads in data centers are different from conventional computers
  - Storage based on inexpensive disks that fail frequently
    — MapReduce is fault tolerant
  - Many large files in contrast to small files for personal data
    — MapReduce aims at processing large amount of data once
  - Primarily reading streams of data — MapReduce reads chunks of large files
  - Sequential writes appending to the end of existing files
    — Output file keep growing as workers keep writing
  - Must support multiple concurrent operations
    —MapReduce has thousands of workers simultaneously
  - Bandwidth is more critical than latency
    —MapReduce only wants to finish tasks within “reasonable” amount of time

Real world, industry experience

- Linux problems (section 7)
  - Linux driver issues — disks do not report their capabilities honestly
  - The cost of fsync — proportion to file size rather than updated chunk size
  - Single reader-writer lock for mmap
  - Due to the open-source nature of Linux, they can fix it and contribute to the rest of the community
- **GFS is not open-sourced**



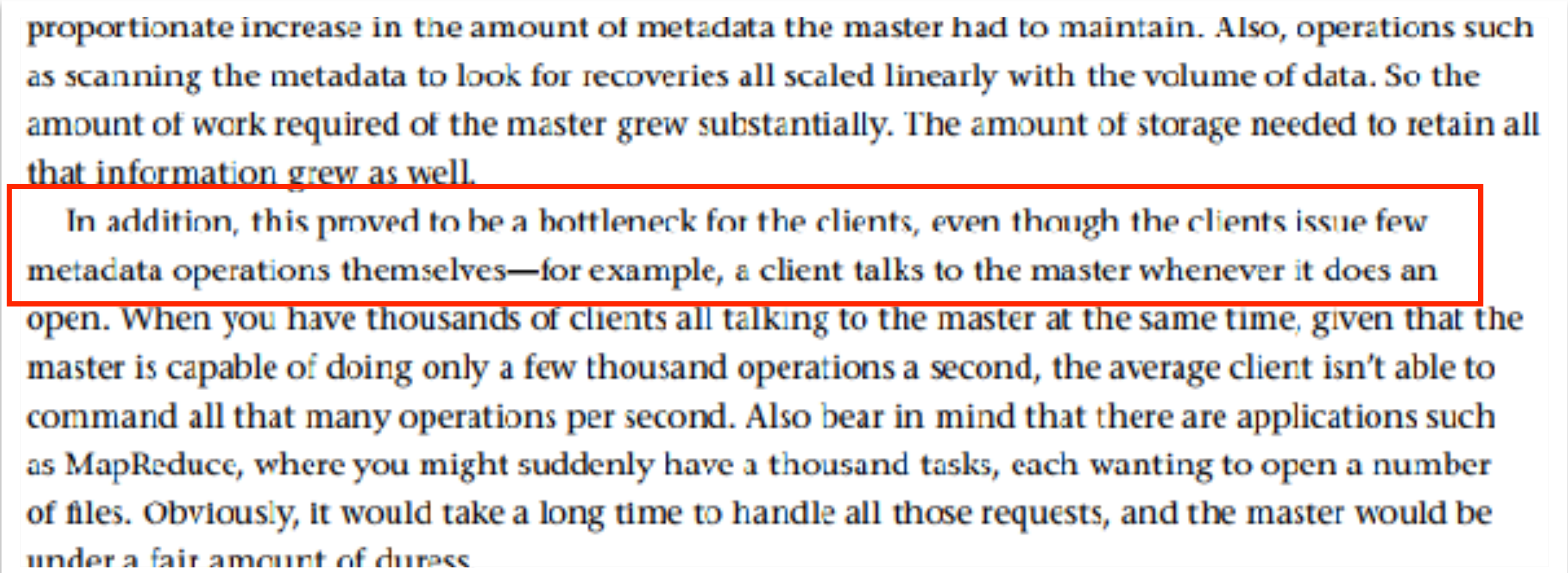
system behavior. When appropriate, we improve the kernel and share the changes with the open source community.

Single master design

- GFS claims this will not be a bottleneck
- In-memory data structure for fast access
- Only involved in metadata operations — decoupled data/control paths
- Client cache
- What if the master server fails?

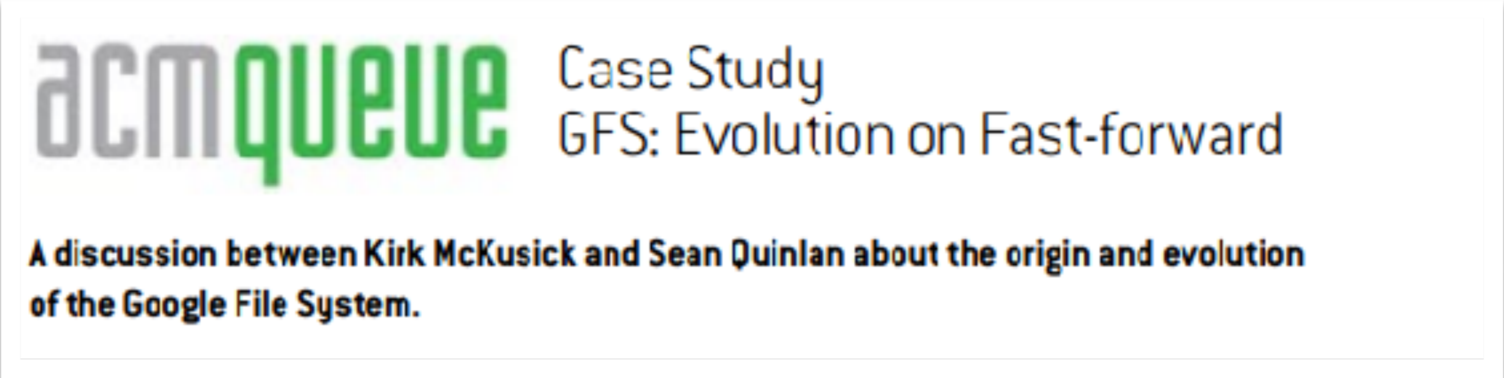
The evolution of GFS

- Mentioned in “Spanner: Google's Globally-Distributed Database”, OSDI 2012 — “tablet’s state is stored in set of B-tree-like files and a write-ahead log, all on a distributed file system called Colossus (the successor to the Google File System)”
- Single master



proportionate increase in the amount of metadata the master had to maintain. Also, operations such as scanning the metadata to look for recoveries all scaled linearly with the volume of data. So the amount of work required of the master grew substantially. The amount of storage needed to retain all that information grew as well.

In addition, this proved to be a bottleneck for the clients, even though the clients issue few metadata operations themselves—for example, a client talks to the master whenever it does an open. When you have thousands of clients all talking to the master at the same time, given that the master is capable of doing only a few thousand operations a second, the average client isn't able to command all that many operations per second. Also bear in mind that there are applications such as MapReduce, where you might suddenly have a thousand tasks, each wanting to open a number of files. Obviously, it would take a long time to handle all those requests, and the master would be under a fair amount of duress

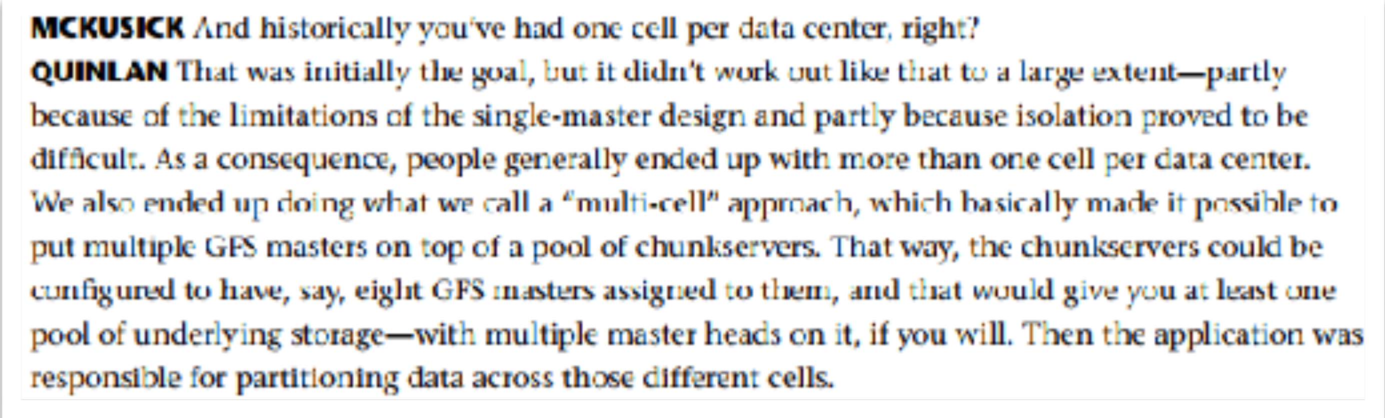


acmqueue

Case Study

GFS: Evolution on Fast-forward

A discussion between Kirk McKusick and Sean Quinlan about the origin and evolution of the Google File System.

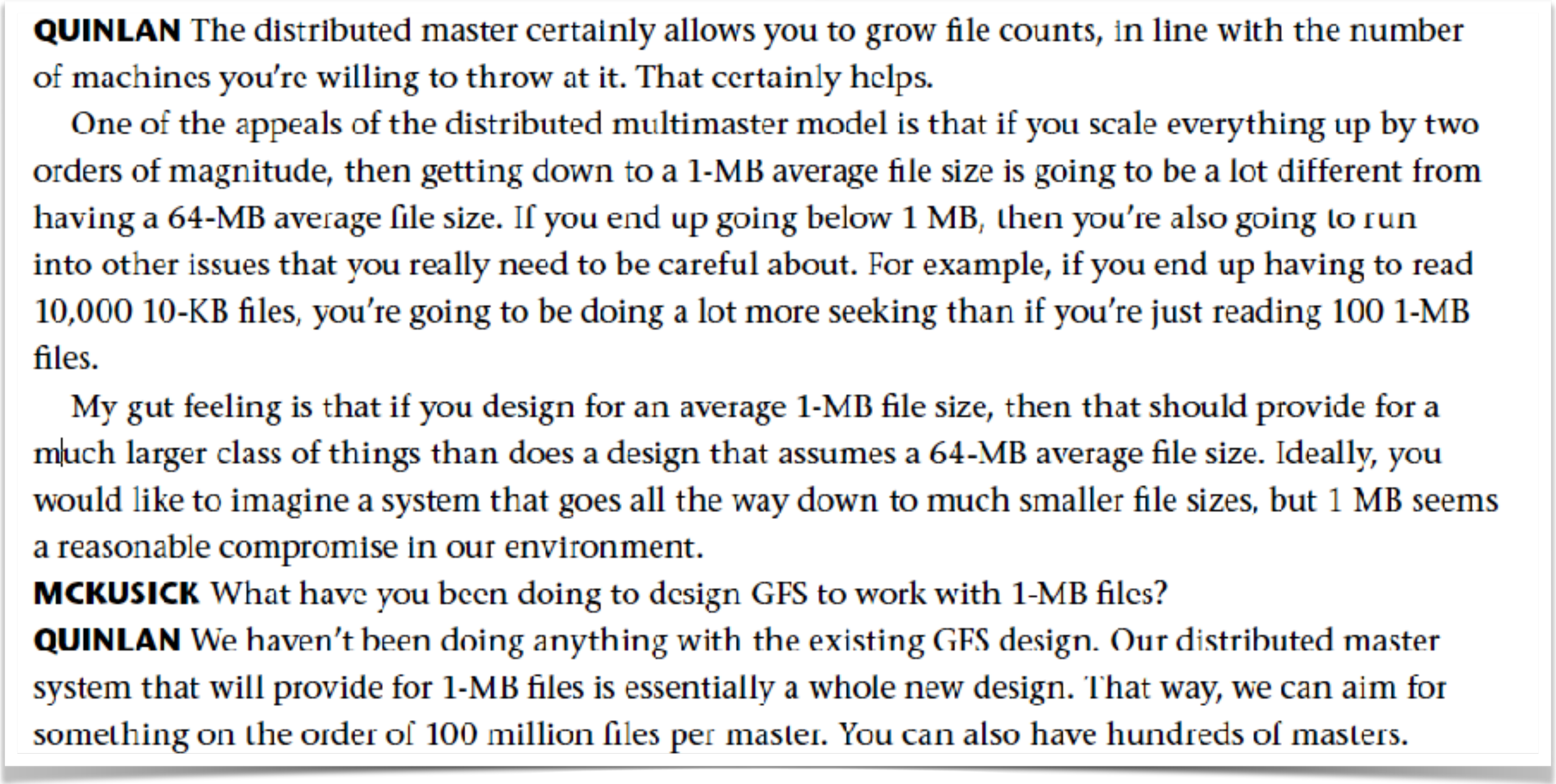


MCKUSICK And historically you've had one cell per data center, right?

QUINLAN That was initially the goal, but it didn't work out like that to a large extent—partly because of the limitations of the single-master design and partly because isolation proved to be difficult. As a consequence, people generally ended up with more than one cell per data center. We also ended up doing what we call a “multi-cell” approach, which basically made it possible to put multiple GFS masters on top of a pool of chunkservers. That way, the chunkservers could be configured to have, say, eight GFS masters assigned to them, and that would give you at least one pool of underlying storage—with multiple master heads on it, if you will. Then the application was responsible for partitioning data across those different cells.

The evolution of GFS

- Support for smaller chunk size — gmail



QUINLAN The distributed master certainly allows you to grow file counts, in line with the number of machines you're willing to throw at it. That certainly helps.

One of the appeals of the distributed multimaster model is that if you scale everything up by two orders of magnitude, then getting down to a 1-MB average file size is going to be a lot different from having a 64-MB average file size. If you end up going below 1 MB, then you're also going to run into other issues that you really need to be careful about. For example, if you end up having to read 10,000 10-KB files, you're going to be doing a lot more seeking than if you're just reading 100 1-MB files.

My gut feeling is that if you design for an average 1-MB file size, then that should provide for a much larger class of things than does a design that assumes a 64-MB average file size. Ideally, you would like to imagine a system that goes all the way down to much smaller file sizes, but 1 MB seems a reasonable compromise in our environment.

MCKUSICK What have you been doing to design GFS to work with 1-MB files?

QUINLAN We haven't been doing anything with the existing GFS design. Our distributed master system that will provide for 1-MB files is essentially a whole new design. That way, we can aim for something on the order of 100 million files per master. You can also have hundreds of masters.

Lots of other interesting topics

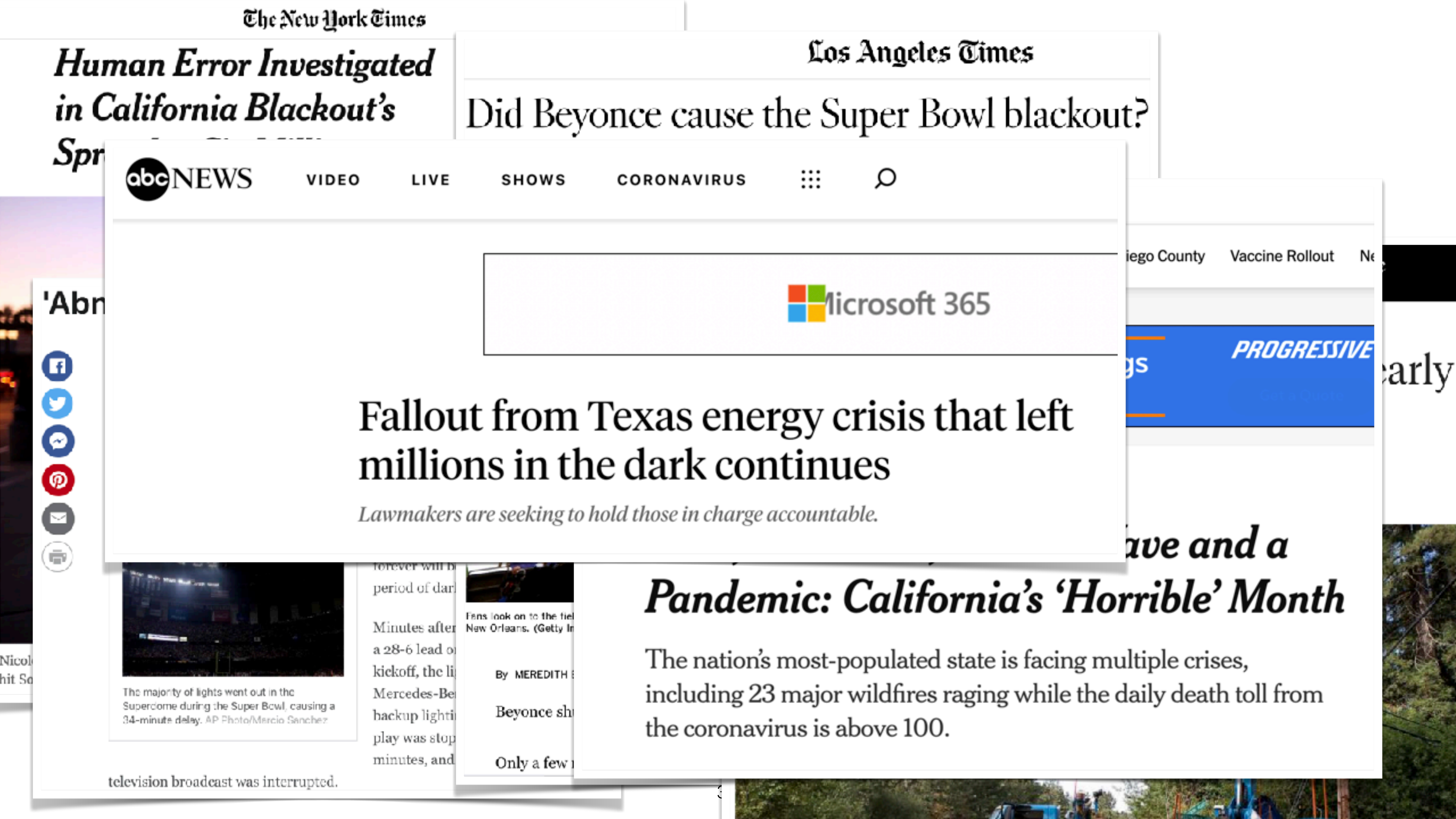
- snapshots
- namespace locking
- replica placement
- create, re-replication, re-balancing
- garbage collection
- stable replica detection
- data integrity
- diagnostic tools: logs are your friends

Do they achieve their goals?

- Storage based on inexpensive disks that fail frequently — replication, distributed storage
- Many large files in contrast to small files for personal data — large chunk size
- Primarily reading streams of data — large chunk size
- Sequential writes appending to the end of existing files — large chunk size
- Must support multiple concurrent operations — flat structure
- Bandwidth is more critical than latency — large chunk size

What’s missing in GFS?

- GFS only supports consistency models
- Scalability — single master
- Only efficient in dealing with large data
- No geo-redundancy



The New York Times

Human Error Investigated in California Blackout's Spr

Los Angeles Times

Did Beyonce cause the Super Bowl blackout?

Fallout from Texas energy crisis that left millions in the dark continues

Lawmakers are seeking to hold those in charge accountable.

'Abn

The majority of lights went out in the Superdome during the Super Bowl, causing a 34-minute delay. AP Photo/Marcio Sanchez

forever will b

period of dar

Minutes after a 28-6 lead o kickoff, the li Mercedes-Be backup lighti play was stop minutes, and

television broadcast was interrupted.

Fans look on to the fiel New Orleans. (Getty In

By MEREDITH

Beyonce sh

Only a few

Nicol hit Sc

arly

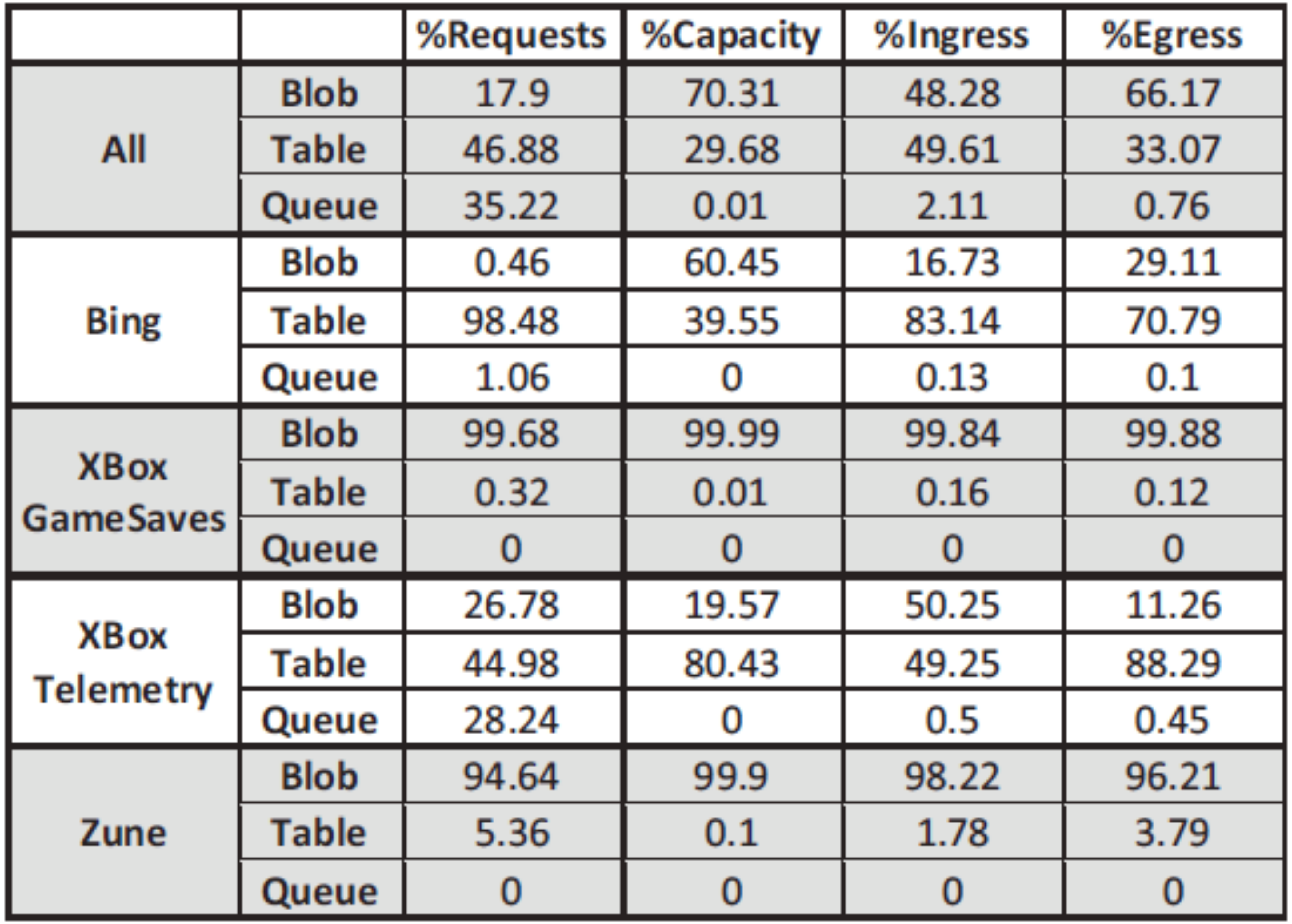
ave and a Pandemic: California's 'Horrible' Month

The nation's most-populated state is facing multiple crises, including 23 major wildfires raging while the daily death toll from the coronavirus is above 100.

Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency

Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, Leonidas Rigas
Microsoft

Data center workloads for WAS



		%Requests	%Capacity	%Ingress	%Egress
All	Blob	17.9	70.31	48.28	66.17
	Table	46.88	29.68	49.61	33.07
	Queue	35.22	0.01	2.11	0.76
Bing	Blob	0.46	60.45	16.73	29.11
	Table	98.48	39.55	83.14	70.79
	Queue	1.06	0	0.13	0.1
XBox GameSaves	Blob	99.68	99.99	99.84	99.88
	Table	0.32	0.01	0.16	0.12
	Queue	0	0	0	0
XBox Telemetry	Blob	26.78	19.57	50.25	11.26
	Table	44.98	80.43	49.25	88.29
	Queue	28.24	0	0.5	0.45
Zune	Blob	94.64	99.9	98.22	96.21
	Table	5.36	0.1	1.78	3.79
	Queue	0	0	0	0

Why Windows Azure Storage

- A cloud service platform for social network search, video streaming, XBOX gaming, records management, and etc. in M$.
  - Must tolerate many different data abstractions: blobs, tables and queues
  - Data types:
    - **Large** Blob(Binary Large OBjects) storage: pictures, excel files, HTML files, virtual hard disks (VHDs), big data such as logs, database backups -- pretty much anything.
    - **Large** Table: database tables
    - **Small** Queue: store and retrieve messages. Queue messages can be up to 64 KB in size, and a queue can contain millions of messages. Queues are generally used to store lists of messages to be processed asynchronously.

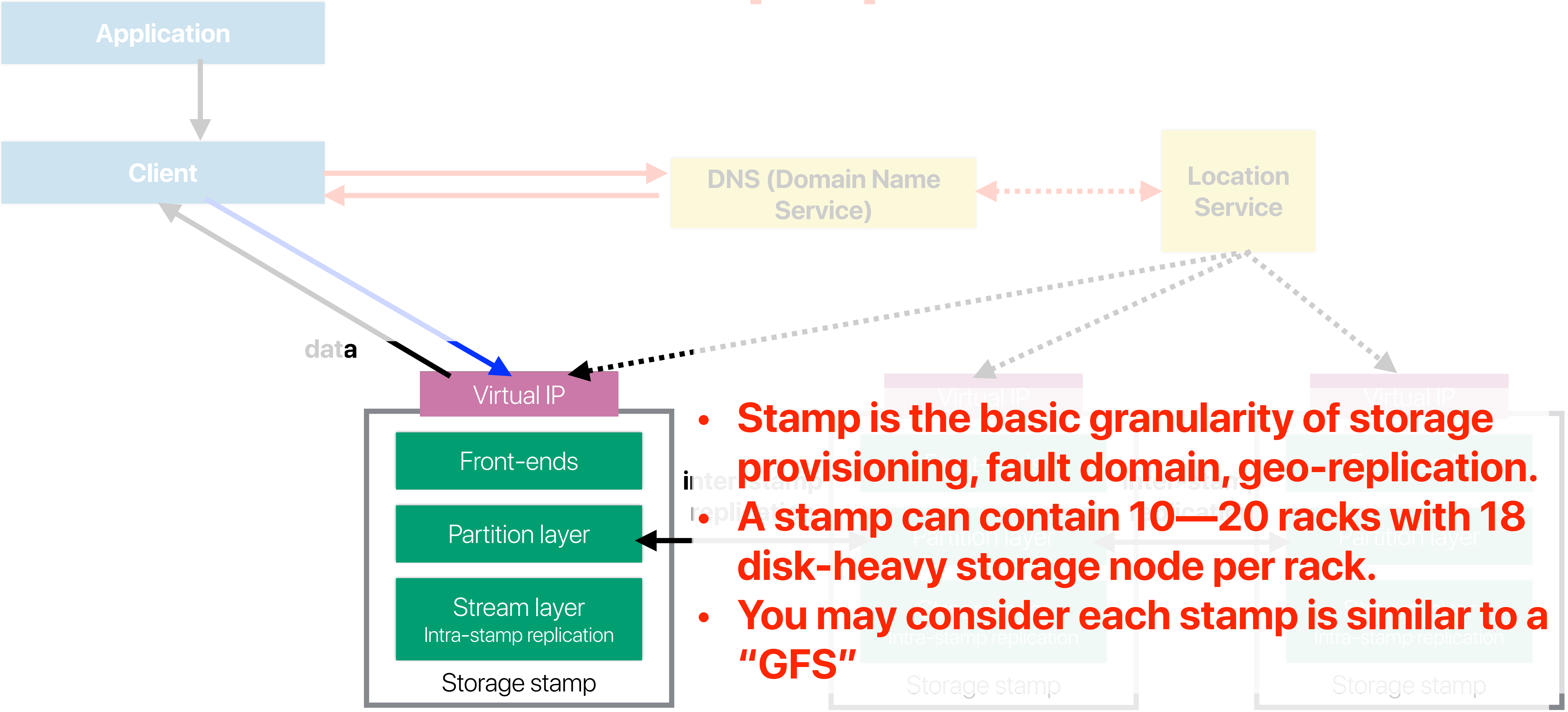
Why Windows Azure Storage (cont.)

- Learning from feedbacks in existing cloud storage
  - Strong consistency
  - Global and scalable namespace/storage
  - Disaster recovery
  - Multi-tenancy and cost of storage

All problems in computer science can be solved by another level of indirection

–David Wheeler

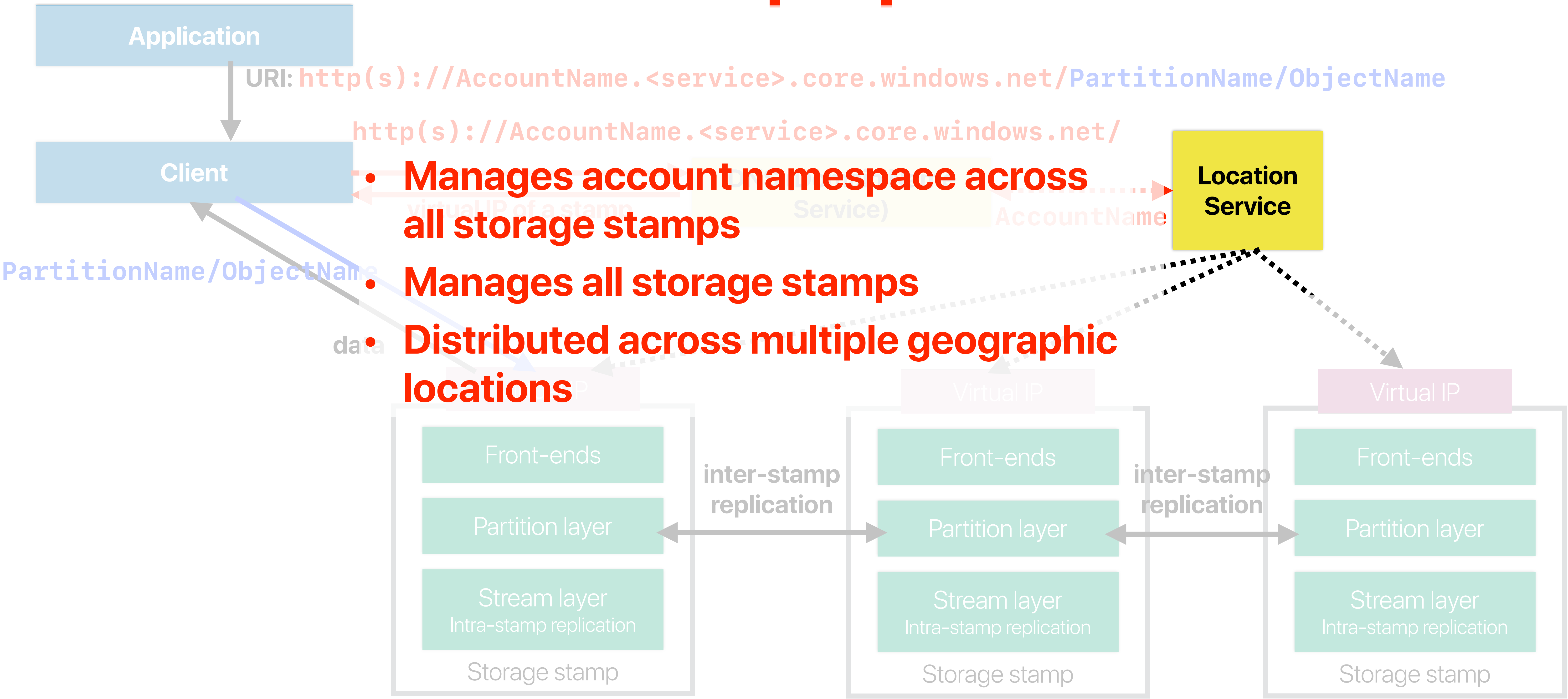
What WAS proposes?



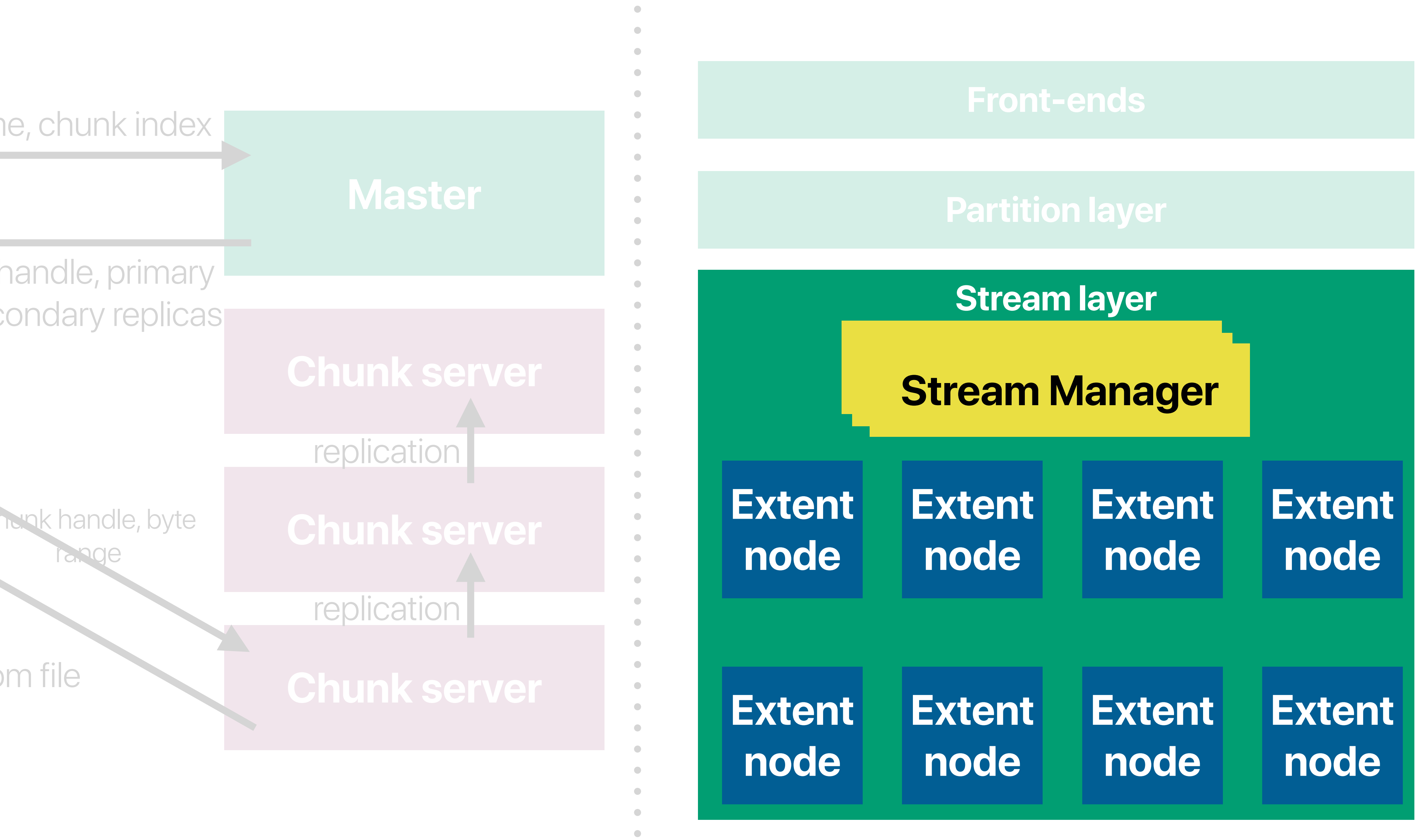
- Stamp is the basic granularity of storage provisioning, fault domain, geo-replication.
- A stamp can contain 10—20 racks with 18 disk-heavy storage node per rack.
- You may consider each stamp is similar to a "GFS"

What WAS proposes?

- Manages account namespace across all storage stamps
- Manages all storage stamps
- Distributed across multiple geographic locations



GFS v.s. stamp in WAS



Poll close in 1:30

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

Poll close in 1:30

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

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

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

② Each block in the stream contains a checksum to ensure the data integrity

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

③ An update to a stream can only be appended to the end of the stream

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

LogFS

④ Two streams can share the same set of extents

Minimize the time when creating a new file De-duplication to save disk space

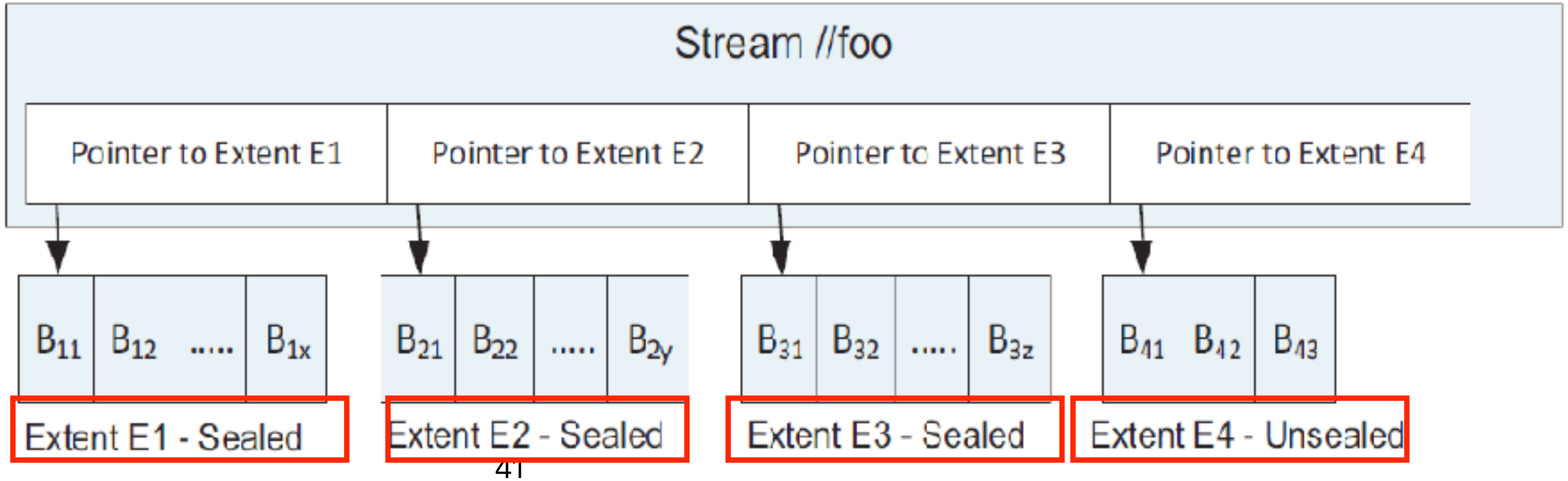
A. 0

B. 1

C. 2

D. 3

[✓ E. 4]



Announcement

- Project due tonight
  - Please submit your current progress
  - You will have another week of "revision period"
    - Allows you to revise your project with 20% 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%+80%*40% = 92%
- iEVAL — count as an extra, full-credit reading quiz
- Final — contains two parts (each account for 50%)
  - Part 1: unlimited time between 3/12-3/17, open-ended questions
  - Part 2: 80 minute multiple choices/answers questions + two problem sets of comprehensive exam questions

Computer Science & Engineering

202

つづく