Distributed Systems

08r. Assignment 5 Review

Paul Krzyzanowski

Rutgers University

Fall 2017

Question 1

What are three advantages of using a large chunk size in GFS?

1. Reduce interaction with master
   - It reduces clients' need to interact with the master because reads and writes on the same chunk require only one initial request to the master for chunk location information.
2. Reduce network overhead
   - Since on a large chunk, a client is more likely to perform many operations on a given chunk, it can reduce network overhead by keeping a persistent TCP connection to the chunkserver over an extended period of time.
3. Reduces the size of the metadata stored on the master.

Question 2

Explain the role of the GFS master.

The master maintains all file system metadata.

- This includes
  - the namespace
  - access control information
  - mapping from files to chunks
  - current locations of chunks
- It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunkservers.

Question 3

As Dropbox's design evolved, why did Dropbox split the original web server into two web servers? [What was the function of each server?]

- Dropbox ran out of capacity at the server because all uploads and downloads went to one server.
  - One server dealt with metadata.
  - Another dealt with file uploads and downloads.

Question 4

Why were notification servers added?

Notification servers were added to not require clients to poll the server to check for changes.

This reduces the load on the system since clients that don't have changes don't impose any traffic onto the servers.

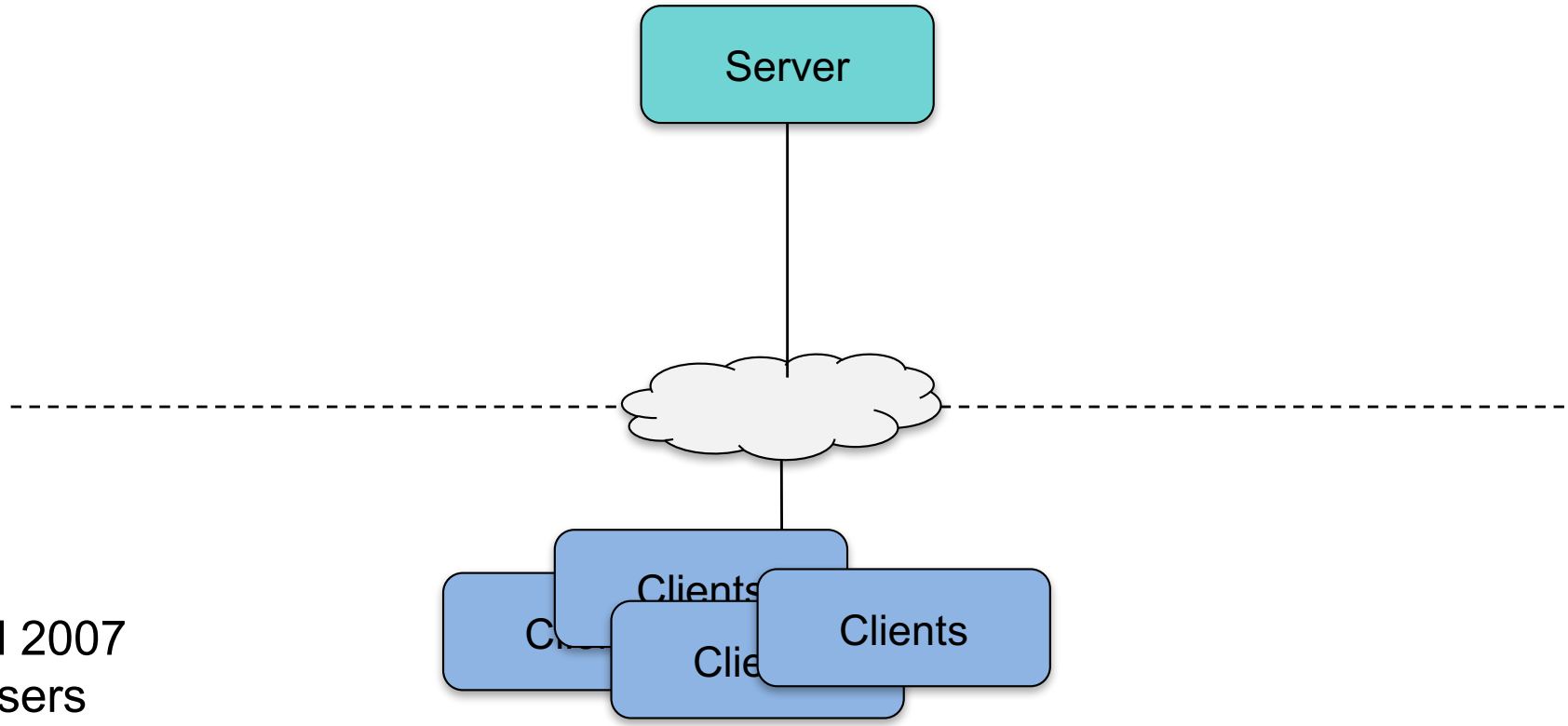
Question 5

Why was RPC-based communication added to the blockservers instead of having them talk to the database?

- Avoids multiple round-trip calls from the blockserver to the database: RPCs can contain higher-level commands

Dropbox: architecture evolution: version 1

One server: web server, app server, mySQL database, sync server

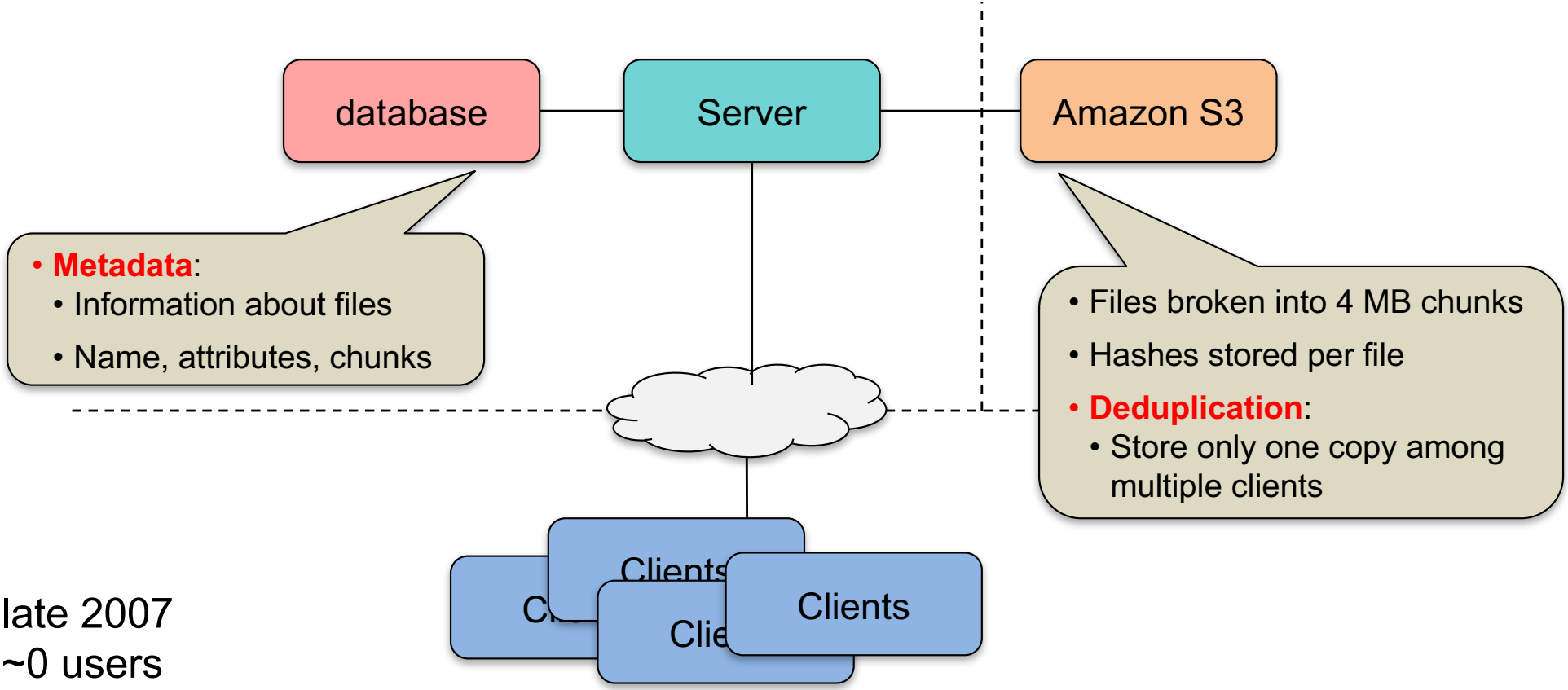


mid 2007
0 users

See http://youtu.be/PE4gwstWhmc

Dropbox: architecture evolution: version 2

- Server ran out of disk space:
  moved data to Amazon S3 service (key-value store)
- Servers became overloaded: moved mySQL DB to another machine
- Clients periodically polled server for changes

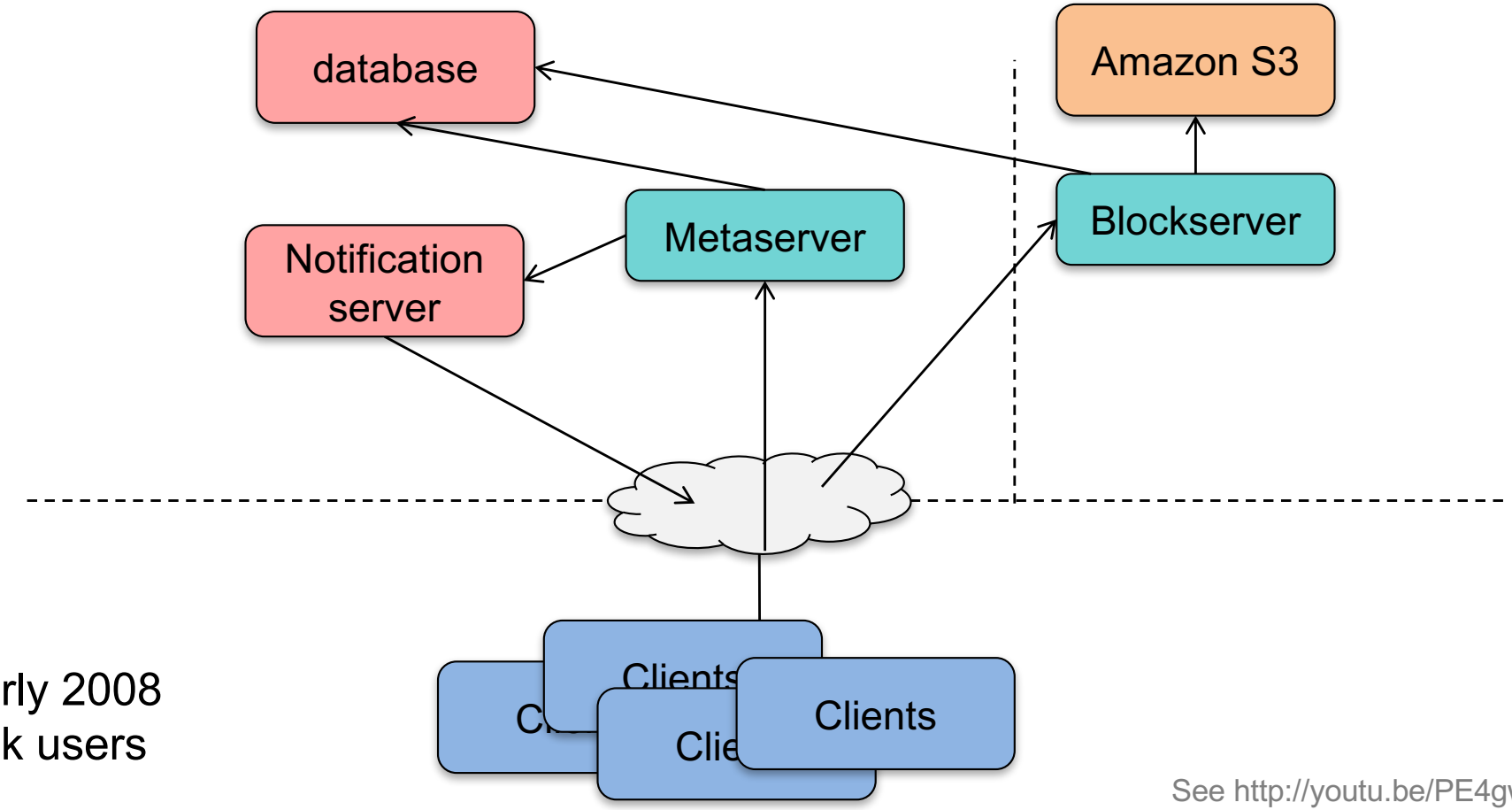


late 2007
~0 users

See http://youtu.be/PE4gwstWhmc

Dropbox: architecture evolution: version 3

- Move from polling to notifications: add **notification server**
- Split web server into two:
  - Amazon-hosted server hosts file content and accepts uploads (stored as blocks)
  - Locally-hosted server (at Dropbox, not Amazon) manages metadata

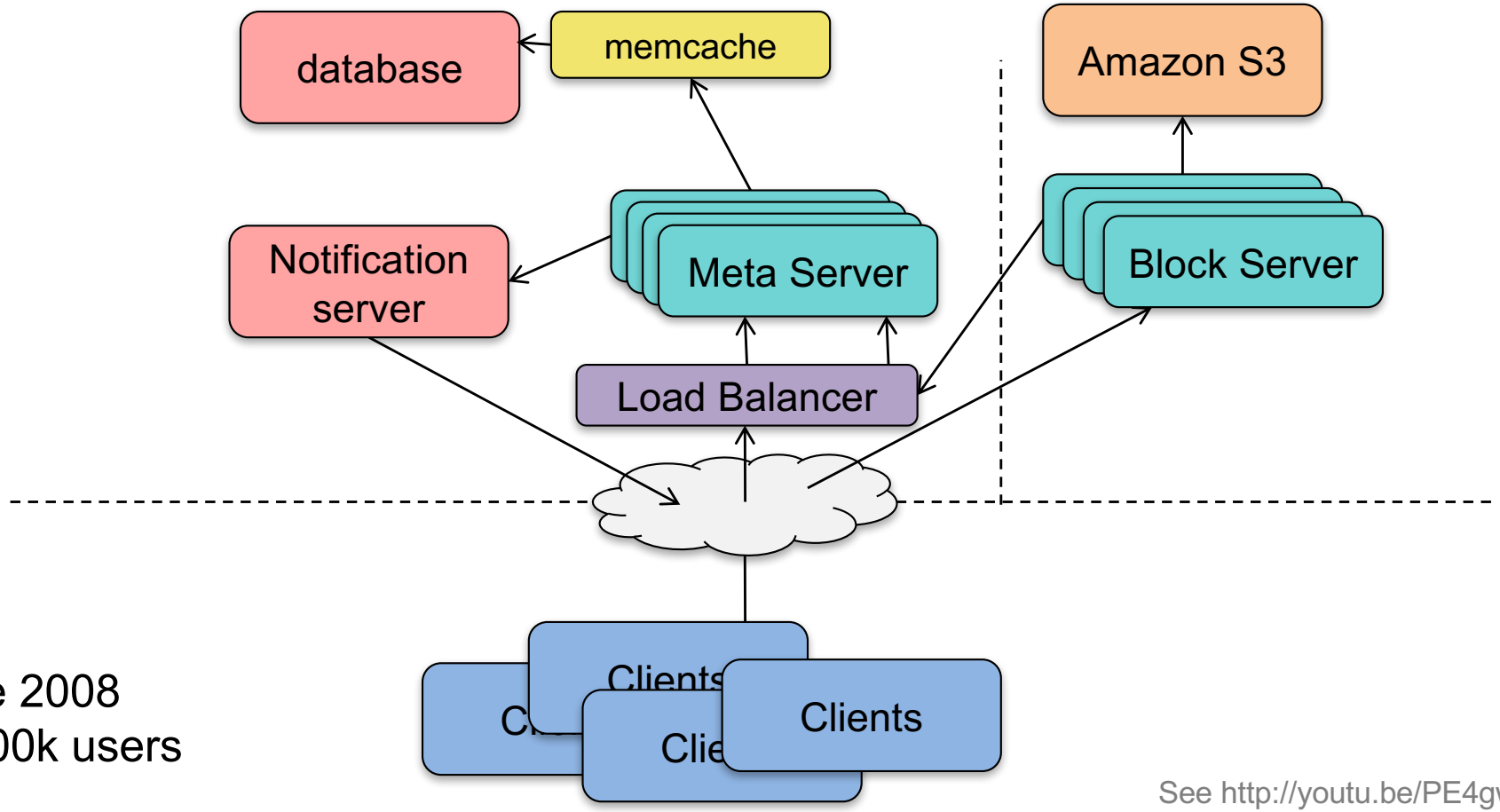


early 2008
50k users

See http://youtu.be/PE4gwstWhmc

Dropbox: architecture evolution: version 4

- Add more *metaservers* and *blockservers*
- Blockservers do not access DB directly; they send RPCs to metaservers
- Add a memory cache (memcache) in front of the database to avoid scaling the database: keep common query results in memory cache

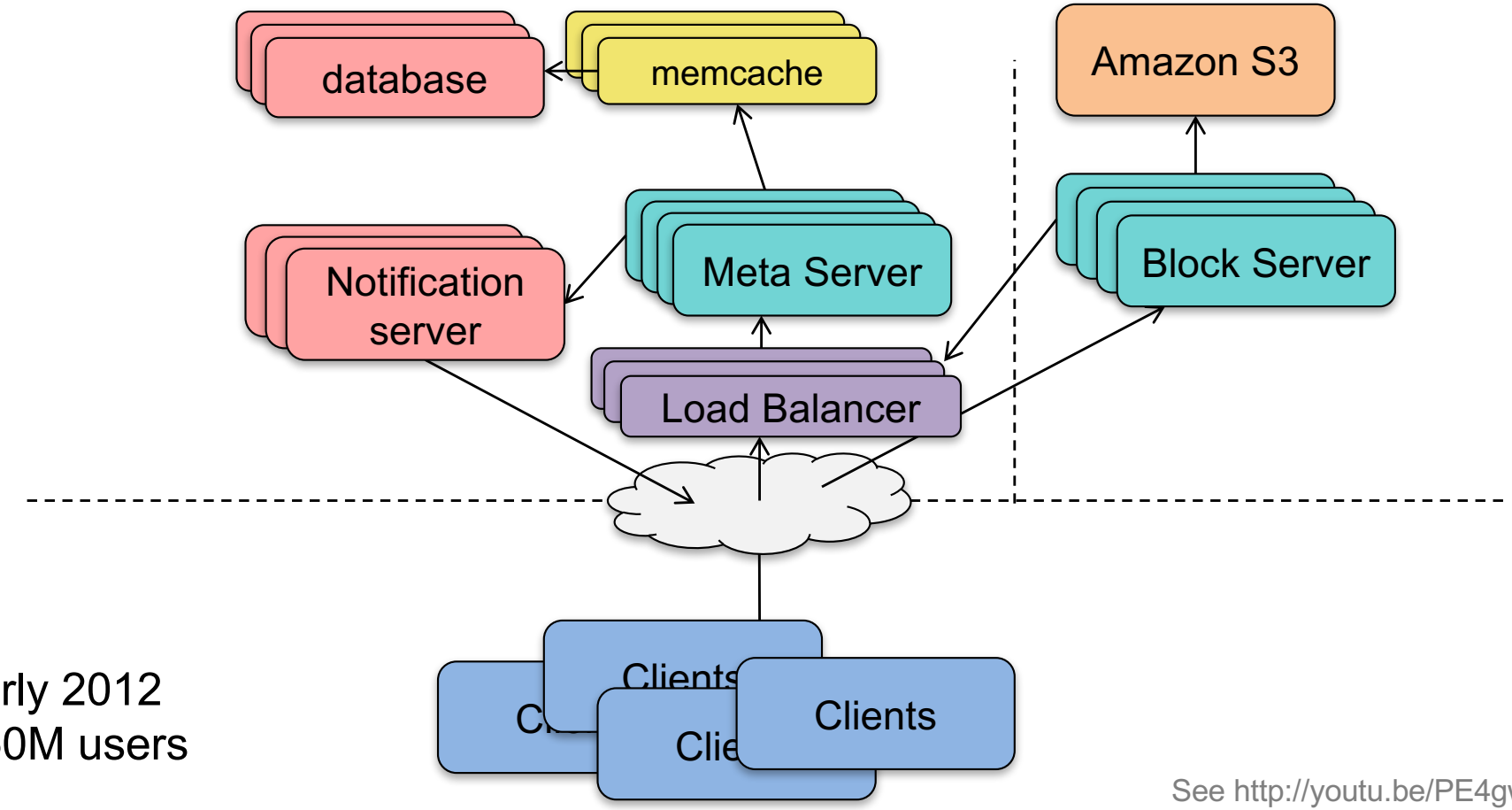


late 2008
~100k users

See http://youtu.be/PE4gwstWhmc

Dropbox: architecture evolution: version 5

- 10s of millions of clients: Clients have to connect before getting notifications
- Add 2-level hierarchy to notification servers: ~1 million connections/server



early 2012
>50M users

See http://youtu.be/PE4gwstWhmc

Exam 2 Study

Exam 2 Topics

- **Mutual exclusion algorithms**
- **Election algorithms**
- **Consensus** (Paxos & Raft)
- **Transactions**
  - Commit protocols (2PC, 3PC, CAP theorem)
  - Concurrency control
  - Distributed deadlock
- **Network file systems**
  - Concepts behind NFS, AFS, Coda, DFS, SMB
  - Chubby
- **Distributed file systems** (file may be spread across multiple systems)
  - Dropbox
  - Google File system (GFS), Hadoop Distributed File System (HDFS)
- **Distributed lookup**
  - Coordinator, Flooding, Distributed Hash Tables (CAN, Chord, Amazon Dyamo)

Fall 2016: Question 2

Why is a transaction log crucial in providing fault tolerance in a two-phase commit protocol?

Hint: the answer has nothing to do with aborts

It allows the transaction to continue from where it left off when the system restarts.

A transaction cannot change its mind even if it dies and restarts. The log keeps the transaction's state.

Bad answers:

- *"Revert if a sub-transaction fails" or "enable rollback"*
  - *That's only done in the case of aborts*
- *Recovery server can take over*
  - *A recovery server cannot access a failed coordinator's log*

Fall 2016: Question 3

How does Eric Brewer's CAP theorem affect the design of highly-available systems that can withstand network partitions

If you need availability and partition tolerance, you cannot have consistency.

In distributed systems, we usually expect network partitions to occur occasionally.

Therefore, the choice is *consistency across replicas* or *high availability*.

Fall 2016: Question 4

How did callbacks in AFS enable it to scale to support more clients than NFS?

They allow each client to support long-term caching – no need to check with the server; it will tell you if a file has been modified.

Bad answer:

Explain what callbacks are without explaining why they enable AFS to scale

Fall 2016: Question 5

Under what conditions would consistent hashing be unnecessary for a distributed hash table?

If you never need to expand or shrink the table – i.e., you never need to add or remove servers

The whole purpose of consistent hashing is to create hash keys that don't change if you change the size of the table.

Bad answers:

- *"If there are fewer keys than the number of slots"*
- *"If there are no collisions" or any answer related to collisions.*

Fall 2016 – Exam 3 – Question 1

Unlike the design of Chord, Amazon Dynamo stores the entire list of nodes on each system. Explain the advantage of doing this instead of using a finger table.

Each node knows exactly which node contains the desired key and can forward the query there directly. There is no need for multiple hops.

A finger table may require hops: O(log N) vs. O(1)

Bad answers:

- *Better fault tolerance*
- *Easier to add new nodes*
  (if every node stores the entire list, you still need to update all nodes)

Fall 2016: Question 8

A *Paxos proposer* may change its mind and pick a different proposal:

a) If it receives a higher-numbered proposal from any acceptor.
b) If it receives a different proposal number from the majority of acceptors.
c) If it receives a delayed but earlier proposal from a client.
d) Only after consensus has been reached.

If *any* acceptor responds with a higher # proposal, the proposer is obligated to use that for phase 2 of the protocol.

This is a way to share knowledge of the highest proposal received so far.

Lower proposer numbers are not chosen to avoid the problem of delayed messages.

Fall 2016: Question 9

Unlike Paxos, Raft:

a) Requires all requests to go through an elected leader.
b) Does not require a majority of servers to agree to the proposed value.
c) May not always succeed in achieving consensus even with a majority of servers functioning.
d) Provides a fault-tolerant framework for consensus.

Paxos makes an elected leader ("distinguished proposer") optional; Raft requires it.

Fall 2016: Question 10

To achieve consensus, Paxos requires the functioning of:

a) The majority of proposers
b) The majority of acceptors.
c) The majority of learners.
d) All of the above.

The algorithm supports multiple proposers, acceptors, and learners for fault tolerance.

At any time, only one proposer and one learner has to be active.
However, a majority of acceptors have to be alive.

Only one proposer is usually used (called the "distinguished proposer").

- With multiple proposers, there's a greater risk of the proposal being rejected with a higher proposal number from another proposer (which is OK, but will require another round of the protocol).

At least one learner is needed to propagate the knowledge.
Multiple learners will simply propagate redundant messages.

Fall 2016: Question 11

Which is not an ACID property of transactions?

a) *Atomic*. The intermediate state of a transaction is not visible.
b) *Consistent*. A transaction cannot leave the system in an inconsistent state.
c) *Isolated*. The net effect of concurrent transactions should be the same as if they ran in serial order.
d) *Distributed*. A transaction is made up of sub-transactions running on multiple systems.

Distributed – transactions do not have to be distributed
D stands for *durable*

Question 12

The *two-phase commit protocol* uses two phases to:

a) Perform a tentative commit followed by a permanent commit.
b) Give a chance for a sub-transaction to request more time.
c) Make sure everyone's vote is received before taking action.
d) Release resources used by each sub-transaction prior to committing.

Phase 1: Get a vote from everyone
Phase 2: Tell everyone what to do

(a) There is no such thing as a *tentative commit* in the protocol

(b) If a sub-transaction needs more time, it can choose to abort or delay responding until it is ready. The transaction manager (coordinator) will be happy to wait as long as necessary.

(c) Releasing resource locks is part of the commit/abort process.

Question 13

The *three-phase commit protocol*:

a) Allows a coordinator to abort all sub-transactions if any one does not respond to a vote.
b) Enables a recovery coordinator to query any sub-transaction for the state of the protocol.
c) Allows a participant to query another participant to decide to commit or abort if it does not hear from the coordinator.
d) All of the above.

3PC was designed to:

- Enable the use of a recovery coordinator
- Enable timeout-based aborts/commits (in cases where possible)

(a) If no commit/abort directives were issued, the coordinator can tell everyone to abort if there's a delay from any participant

(b) If any sub-transaction has a "pre-commit" vote, that means consensus was reached. If not, the recovery coordinator knows it can abort

(c) If a participant queries any other participant and gets the "pre-commit" vote, it knows the guaranteed outcome of the transaction.

Question 14

Two-phase locking means a transaction:

a) First checks if any other transaction holds the lock before requesting it.
b) First requests a read lock, followed by a write lock only if it needs to modify the resource.
c) Becomes the owner of the lock and can grant it to another transaction when it is finished.
d) Cannot request a lock if it already released any lock.

Phase 1: acquire all locks
Phase 2: release all locks

Question 15

Unlike two-phase locking, *strict two-phase locking*:

a) Prohibits a transaction from getting a lock unless it first checks if it is held by another process.
b) Requires a transaction to get a lock for any resource it accesses.
c) Ensures that other transactions cannot access uncommitted data.
d) Ensures that transactions maintain serial order.

Phase 1: acquire all locks
Phase 2: release all locks at commit/abort

Strict two phase locking avoids the need for cascading aborts

Question 19

If two clients modify the same file concurrently using AFS:

a) Two distinct versions of the file will be created on the server.
b) The last client to close the file overwrites the other client's changes.
c) Changes will be applied to the server in the order that they are made by the clients.
d) The last client to open the file will be considered to have the latest version and its changes will win.

AFS uses session semantics: the last client to close a modified file wins – its changes override any previous changes.

Question 20

Coda's *client modification log* is used to:

a) Allow servers to keep track of which clients modified which files.
b) Make it easy to merge changes to the same file from multiple clients.
c) Propagate client changes to the server.
d) Provide a version control mechanism on the server.

When disconnected, clients log files that were change to the CML.

When reconnected, the log contains a list of files that need to be uploaded to the server(s).

Question 21

SMB *oplocks* (opportunistic locks) are:

a) Locks pushed by the server to force a client to wait before opening a file.
b) A technique to maximize concurrent file access without getting locks, resolving conflicts when the file is closed.
c) Permissions granted by the server to tell the client how it can cache file data.
d) Exclusive locks on a region of a file to ensure that only one client can modify data in that region.

---

oplocks control client caching behavior

Question 22

Which storage systems keep file metadata (information) and file data on separate servers?

a) GFS and Chubby.
b) Dropbox and GFS.
c) SMB and Chubby.
d) AFS and Coda.

Chubby: single-server file system – but replicated

SMB: protocol for single-server file systems

AFS: servers contain volumes, which store data & related metadata (the *volume location database* identifies where a volume lives)

Coda: Same as AFS

Dropbox: database for metadata and Amazon S3 for file data

GFS: master stores metadata; chunkservers store data

Question 23

GFS minimizes the time spent locking a file during appends by:

a) Using optimistic concurrency control techniques.
b) Allowing only one process to open a file at any time.
c) Using strict two-phase locking.
d) Uploading the data fully before making it part of the file.

Two phase (NOT two-phase locking) process used for writing

1. Send the data: propagated to N chunkserver replicas serially
   Client sends the data to one chunkserver, which then forwards a copy to another chunkserver, and so on …
2. Make the data part of the file:
   GFS Master locks the file
   Primary chunkserver sends *write* request to all replicas – ensures consistent order is maintained
   After all replicas are done, the GFS Master unlocks the file

Data flow is separated from the control flow.

Fall 2016 – Exam 3 – Question 8

What is an advantage of Amazon Dynamo's *virtual nodes* over simply having each physical machine be a node?

a) Performance is improved because you can create a massive number of virtual nodes.
b) It's cheaper because the node runs on a virtual machine that can do other things as well.
c) A virtual node can be easily migrated to another system.
d) You can shed load from multiple systems when you add a new physical machine..

Adding a new machine adds more virtual nodes that are scattered throughout the ring.

→ Each virtual node takes on some {key, value} sets from another node
On average, these other nodes are spread across multiple physical systems

Fall 2016 – Exam 3 – Question 9

Amazon Dynamo's *optimistic replication*:

a) Uses Paxos to ensure that all replicas contain identical copies.
b) Grabs locks on all replicas of objects while they are being updated.
c) Forces each transaction to check at commit time whether the data was modified by another transaction.
d) Allows some replication to fail, causing some of the replicas to have older versions.

Optimistic replication = it's ok to fail on replication

Dynamo is designed to be always available and eventually consistent.

The end