Distributed Systems

COMP90015 2023 Semester 1
Tutorial 10

Today’s Agenda

- Distributed File System (DFS) questions (Overview)
- Case Study: Drop Box

1. Name and explain transparencies that should be addressed by distributed file systems.

1. Name and explain transparencies that should be addressed by distributed file systems.

Access transparency

- Client programs don’t know if the file is local of remote

Location transparency

- Client programs don’t know where the file is stored
- Files can be relocated without changing their pathname

Mobility transparency

- Neither client programs nor system administration tables in client nodes need to be changed when files are moved

1. Name and explain transparencies that should be addressed by distributed file systems.

Performance transparency

• Maintain acceptable performance while the load on the service varies within a specified range

Scaling transparency

• Service can be expanded without loss of performance

2. What are the advantages and disadvantages of using absolute names as a naming strategy?

2. What are the advantages and disadvantages of using absolute names as a naming strategy?

- **Absolute name**
- provides a complete address to a file including both the server and path names: <machine name: path name>
- **Advantages**
- Trivial to find a file once the name is given
- No additional state must be kept since each name is self contained (No global state)
- Greater scalability
- Easy to add and delete new names
- **Disadvantages**
- No location transparency
- File is location dependent and cannot be moved
- Less resilient to failure

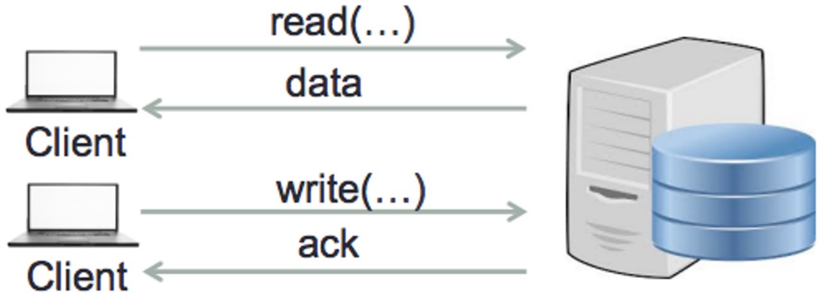
3. What are the advantages and disadvantages of a naming strategy based on mount points?

3. What are the advantages and disadvantages of a naming strategy based on mount points?

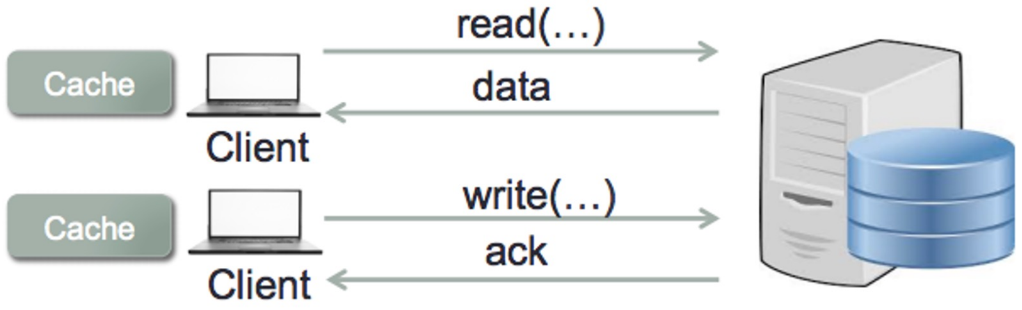
- **Mount Points (used by Sun’s Network File System - NFS)**
- The client machine creates a set of “local names" which are used to refer to remote locations: mount points
- At boot time, the local name is bound to the remote name
- The operating system must maintain a table to maintain the mapping of what server and path are mapped to each mount point
- **Advantages:**
- Names do not contain information about the file location
- Remote location can change between reboots
- **Disadvantages:**
- Hard to maintain
    - What happens when machines fail?
    - What happens when files are migrated?
- Can lead to confusion since two different local names may map to the same file on a remote system

4. What are the advantages and disadvantages of the following two types of distributed file systems?

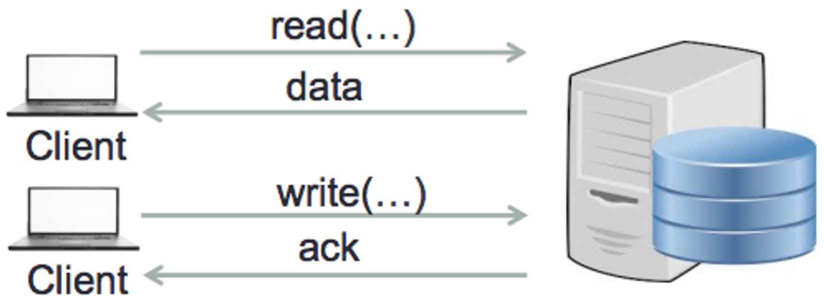
(A)



(B)

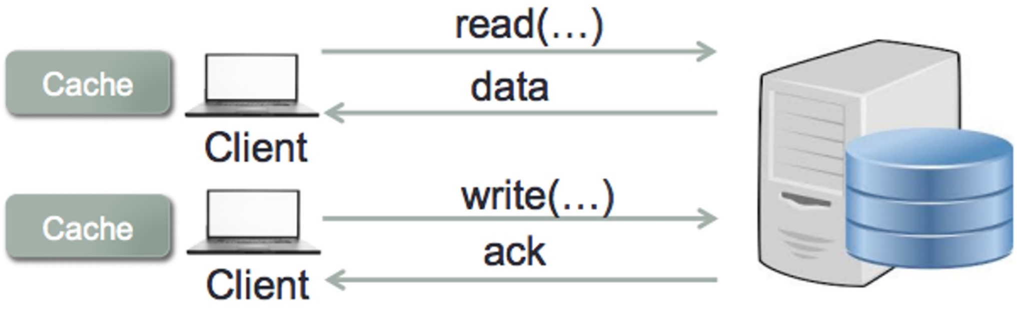


(A)



- RPC to access file system calls
- No client/local caching
- Advantages
- Consistent view of file system
- Disadvantages
- Performance (network access, server becomes potential bottleneck)

(B)



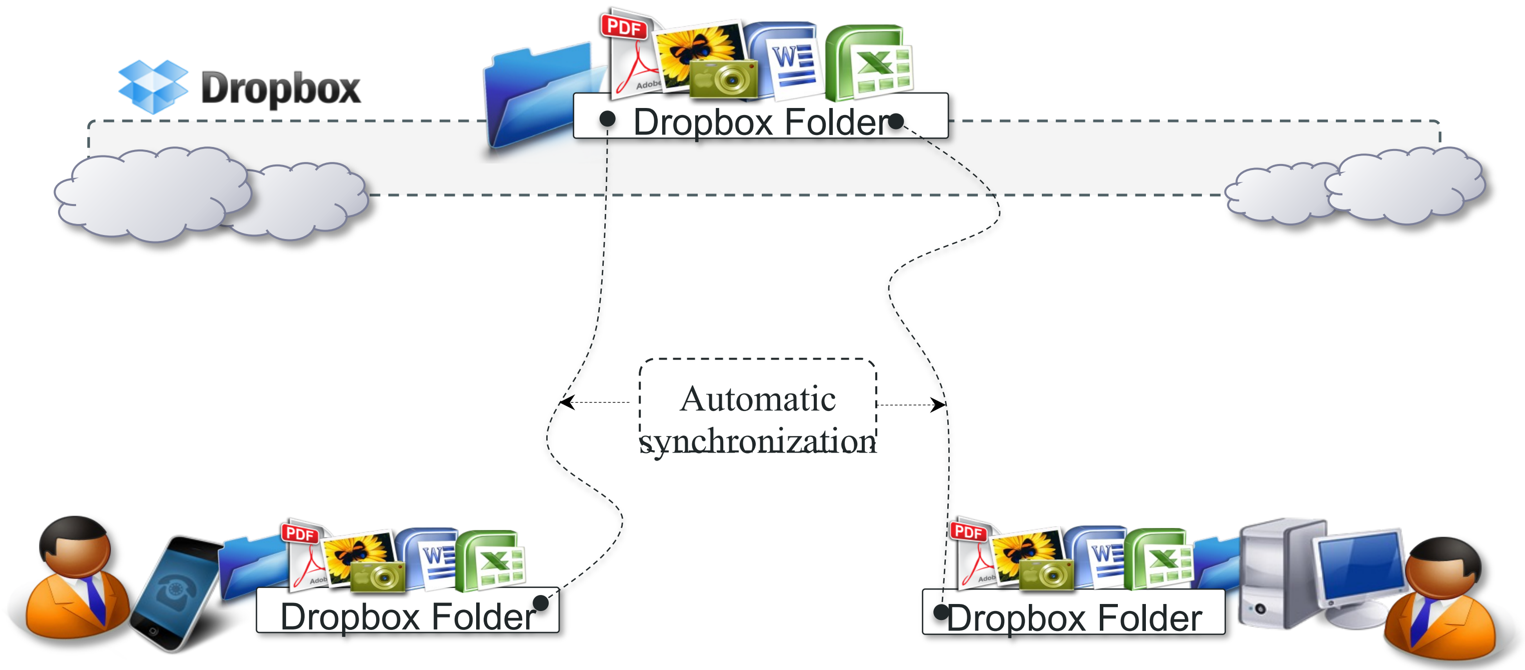
- Cache files on clients and perform local file system operations
- Advantages
- Local operations = better performance
- Disadvantages
- What happens when the client fails?
- Where should the client store the cached files?
- Difficult to keep local copy consistent with remote copy

Case Study: Dropbox

https://youtu.be/PE4gwstWhmc

Dropbox

Dropbox


Dropbox
Dropbox Folder
Automatic synchronization
Dropbox Folder
Dropbox Folder

Dropbox

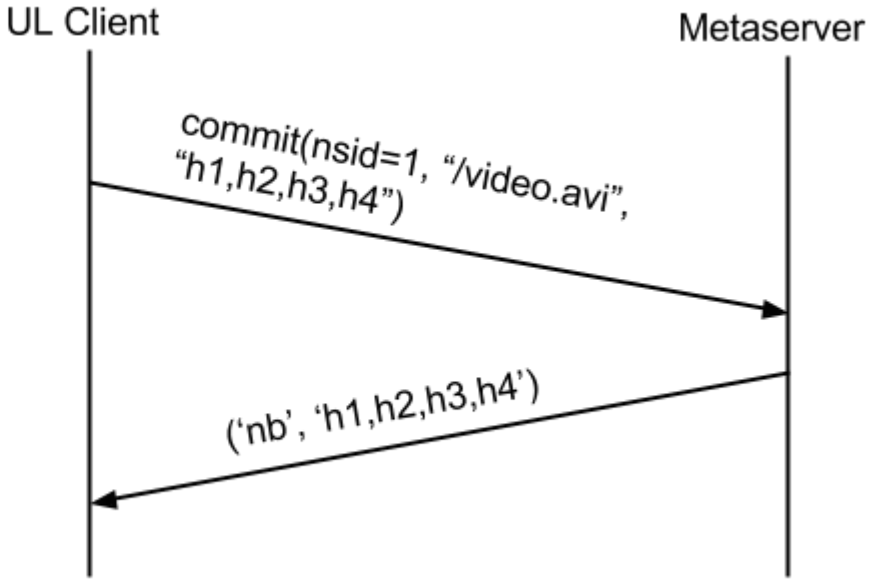
- Client runs on desktop
- Copies changes to local folder
  - Uploaded automatically
  - Downloads new versions automatically
- Huge scale – 100+ million users, 1 billion files/day
- Design
  - Small client, few resources
  - Possibility of low-capacity network to user
  - Scalable back-end
  - (99% of code in Python)

Dropbox

- Everyone’s computer has complete copy of Dropbox
  - Run daemon on computer to track “Sync” folder
- Traffic only when changes occur
  - Results in file upload : file download
  - Huge number of uploads compared to traditional service
  - Dropbox service’s read/write ratio is ***1:1***
- Uses compression to reduce traffic

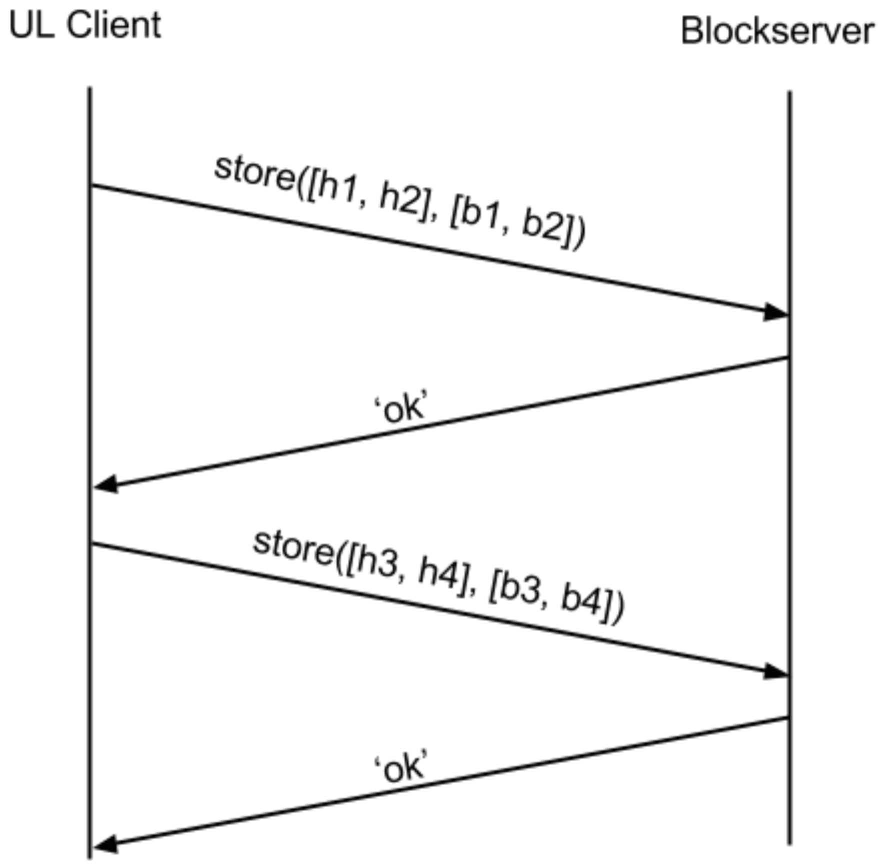
Dropbox - Upload

- Client attempts to "commit" new file
  - Breaks file into blocks, computes hashes
  - Contacts Metaserver
- Metaserver checks if hashes known
- If not, Metaserver returns that it "needs blocks" (nb)



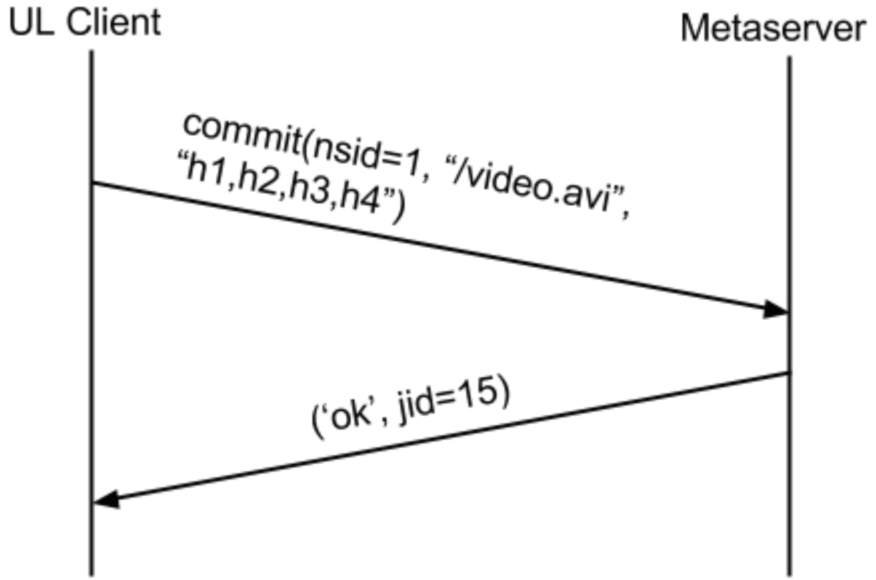
Dropbox - Upload

- Client talks to Blockserver to add needed blocks
- Limit bytes/request (typically 8 MB), so may be multiple requests



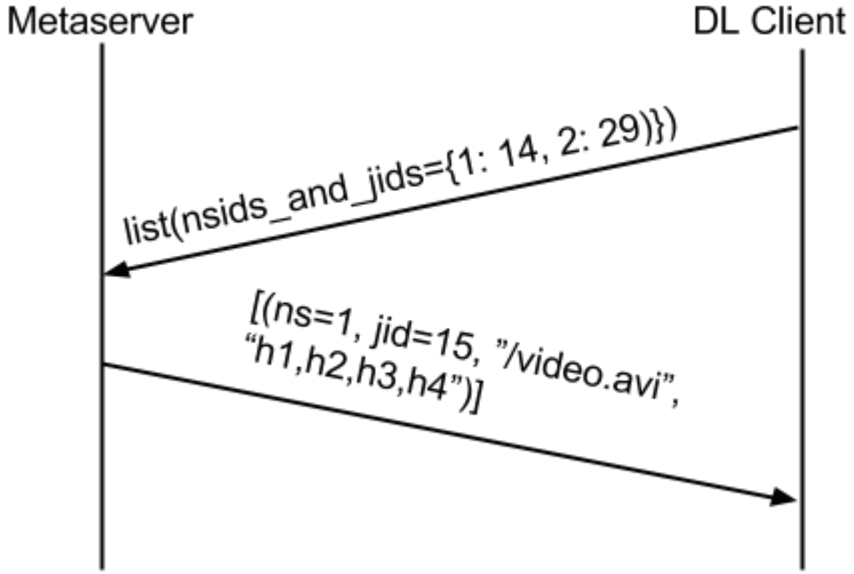
Dropbox - Upload

- Client commits again
  - Contacts Metaserver with same request
- This time, ok



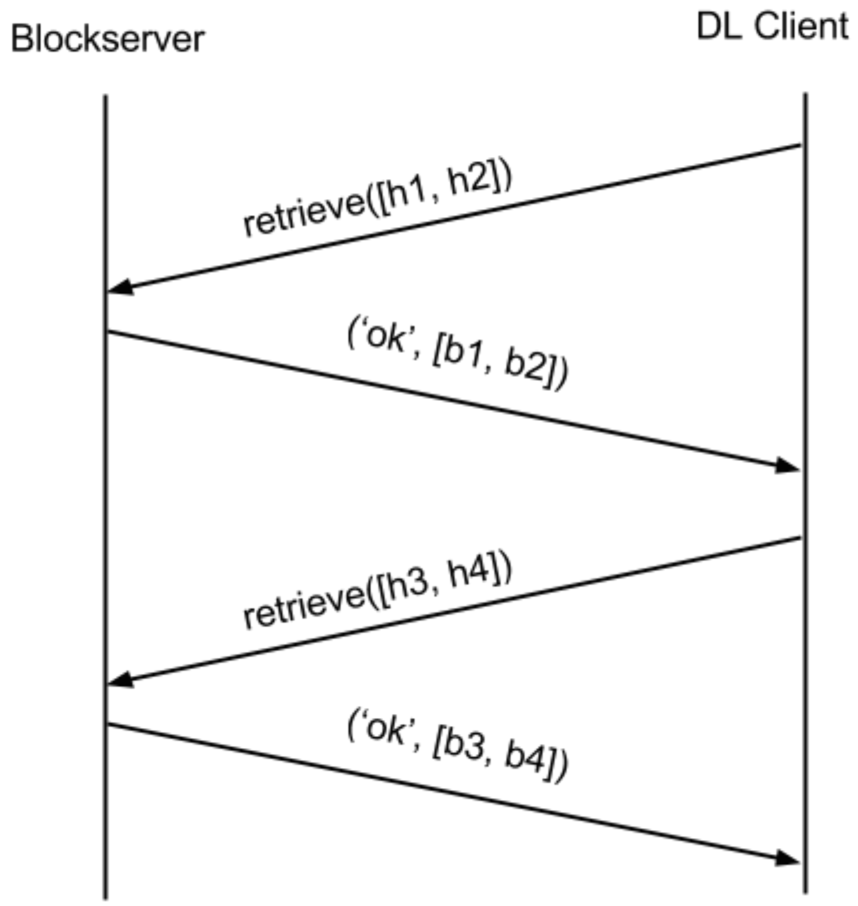
Dropbox - Download

- Client periodically polls Metaserver
    - Lists files it “knows about”
- Metaserver returns information on new files


Metaserver
DL Client
list(nsids_and_jids={1: 14, 2: 29)})
[(ns=1, jid=15, "/video.avi", “h1,h2,h3,h4”)]

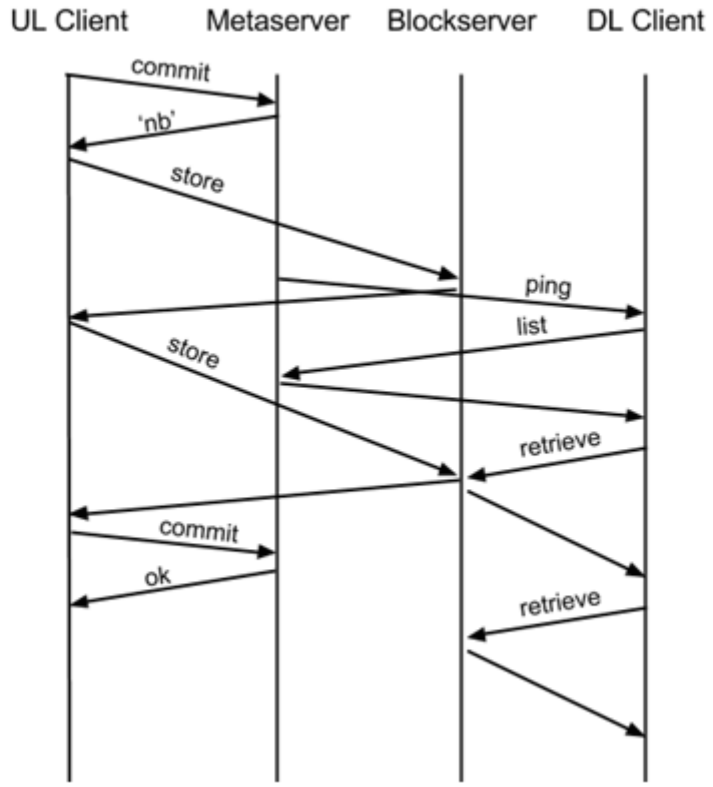
Dropbox - Download

- Client checks if blocks exist
  - For new file, this fails
- Retrieve blocks
- Limit bytes/request (typically 8 MB), so may be multiple requests
- When done, reconstruct and add to local file system
  - Using local filesystem system calls (e.g., open(), write()...)



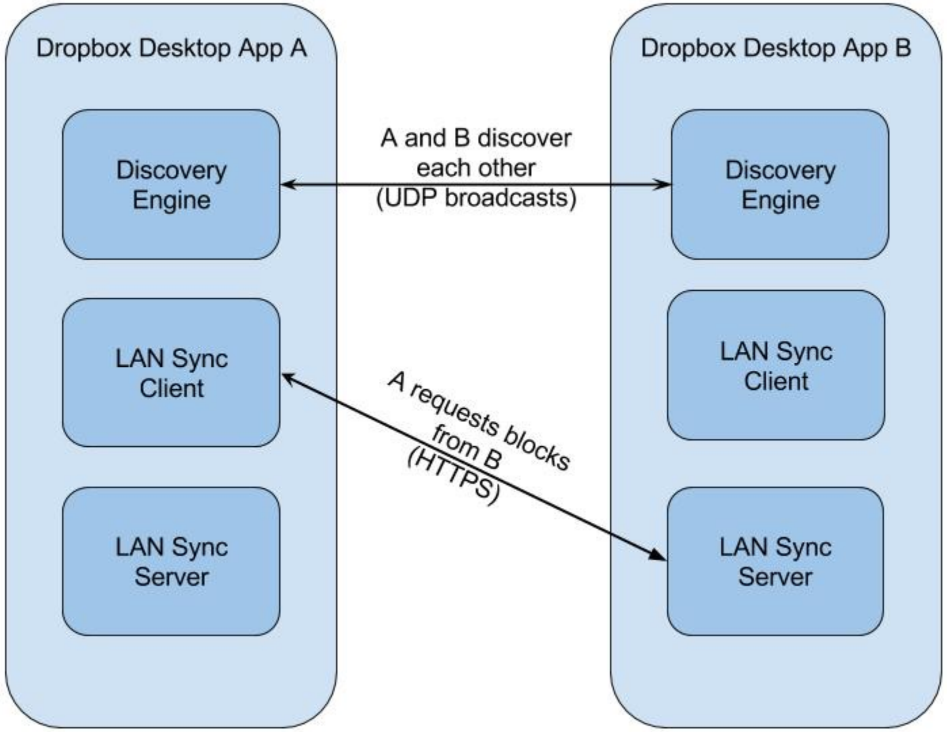
Dropbox – Streaming Sync

- Normally, cannot download to another until upload complete
  - For large files, takes time "sync"
- Instead, enable client to start download when some blocks arrive, before commit
  - Streaming Sync



Dropbox – LAN Sync

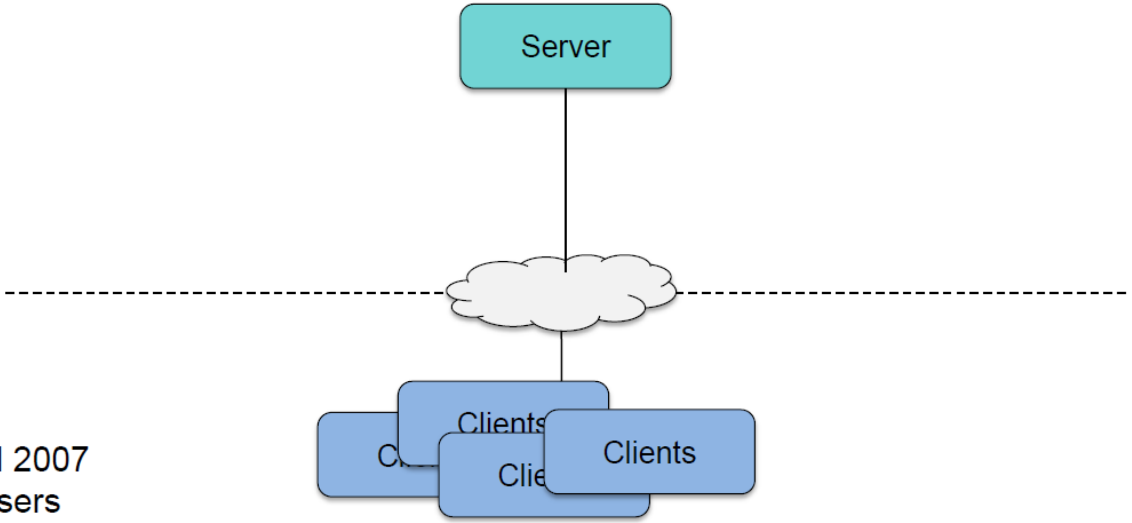
- LAN Sync – download from other clients
- Periodically broadcast on LAN (via UDP)
- Response to get TCP connection to other clients
- Pull blocks over HTTP



DropBox: Architecture Evolution

Dropbox Architecture – v1

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

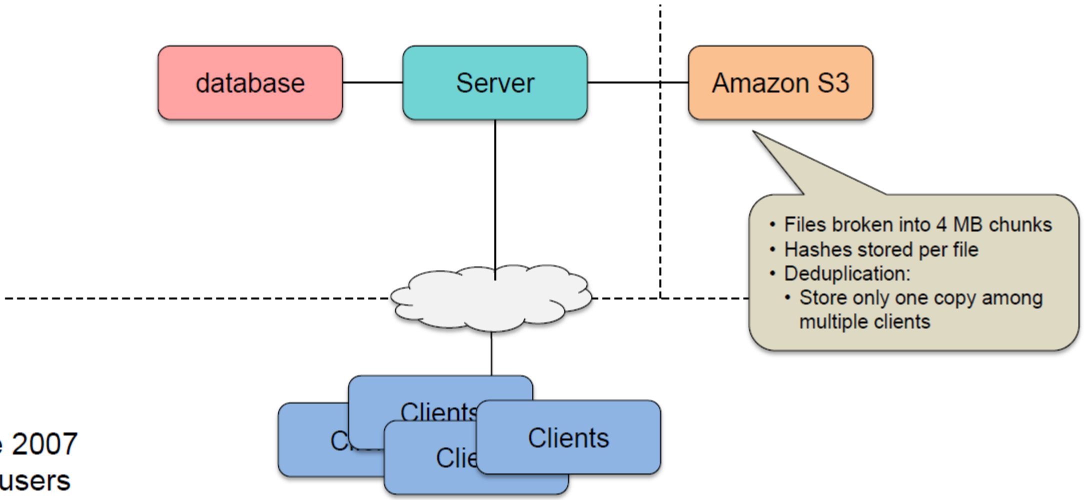


mid 2007
0 users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v2

- 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 polled server for changes periodically



- Files broken into 4 MB chunks
- Hashes stored per file
- Deduplication:
  - Store only one copy among multiple clients

late 2007
~0 users

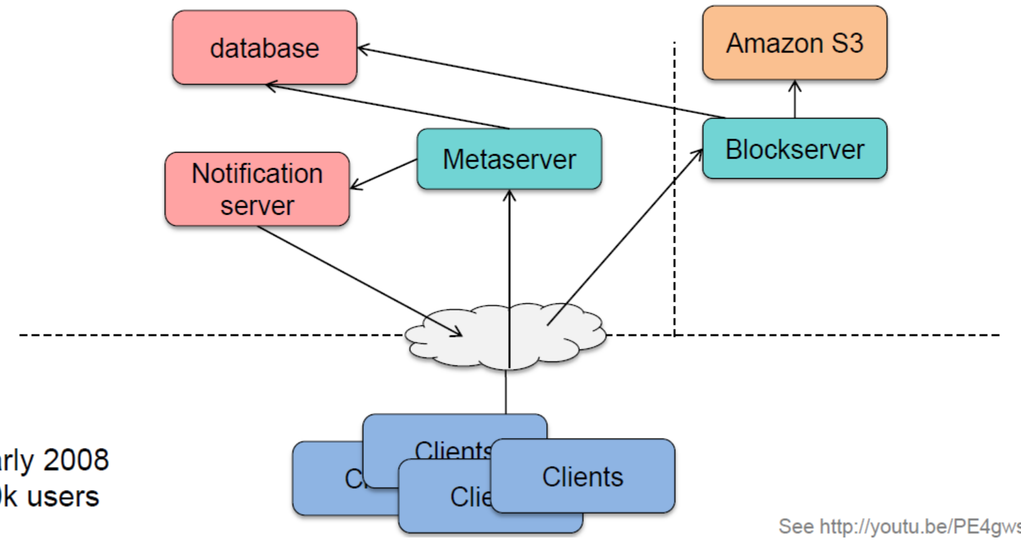
See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v3

– 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 manages metadata

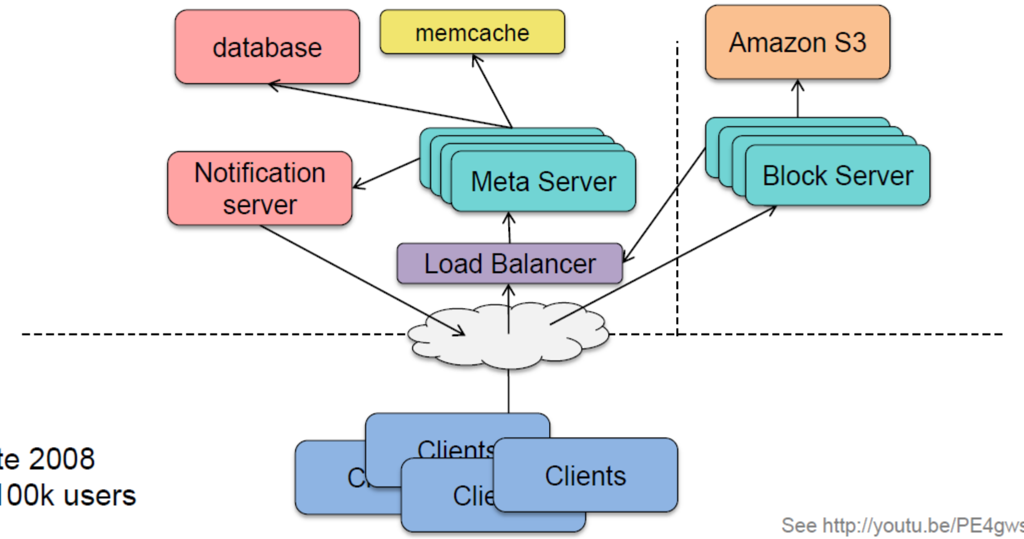


early 2008
50k users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v4

– 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

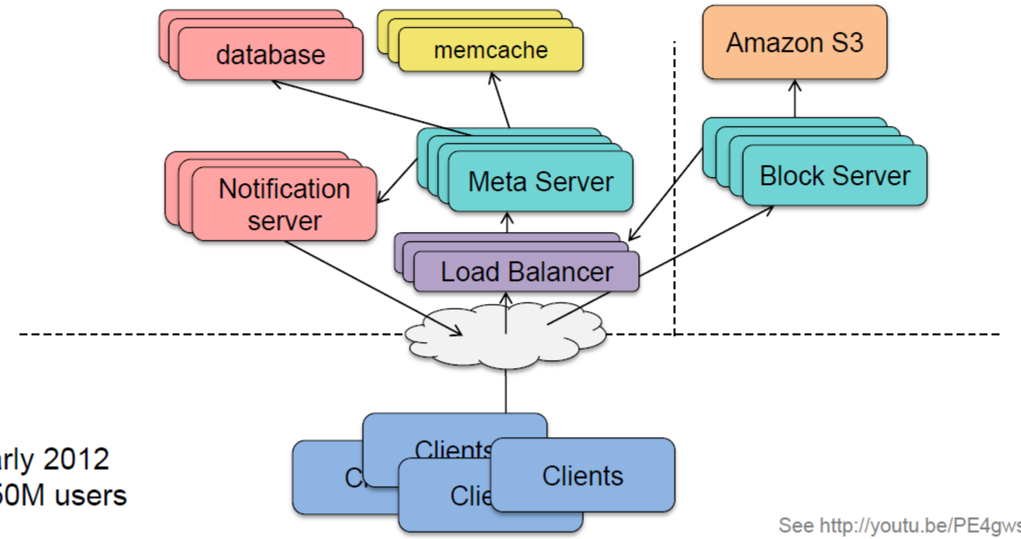


late 2008
~100k users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v5

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



early 2012
>50M users

See http://youtu.be/PE4gwstWhmc