4) SYSTEM DESIGN OF GOOGLE DRIVE / DROPBOX

1) Requirements & Clarifications:
(i) User can upload & download files / folders
(ii) If user is logged in with mutiple devices, all modifications should be reflected in all devices(synchronization)
(iii) Total 500M users and Daily 100M active users
(iv) Avg File/ Photos user has is 200 and file size is 100kb

Total Storage :
100M * 200 * 100 KB = 10 PetaBytes

2) High Level Design :
Read Write will be heavy so 3 application servers

(i) Block Server - (upload/download)
We will upload data in chunks of 4MB, so in case of failure of chunk, we don't need to upload whole file/photos rather we can upload that particular chunk

(ii) Meta-Data Sever (Updating metadata)

(iii) Synchronization Server (All devices will be synced by this)

3) Component Details

(i) **Client** -
We can have 4 components
a) **Internal Local meta-data db** - Store local changes here to avoid going to server often.

b) **Chunker** - Chunker algorithm will be used here to chunk data and optimize it (data deduplication techniques)

c) **Watcher** - watches changes made by user(create, update, delete) and informs indexer. Also it listens to changes broadcasted by synchronization server

d) **Indexer** - It processes instructions given by watcher and also updates the local Db with instructions about chunks of modified files and also communicates with synchronization server about the modifications so that it cab broadcast to other shared users

(ii) **Synchronization Server** - It is an important part and it communicates with Meta-data

server regarding the modifications.
we will need to add asynchronous messaging queue, to efficiently handle synchronization between client and synchronization server

(iii) **Messaging Queue**-
We can have single Global Request Queue which accepts all incoming requests

We can have individual Response Queues, as if we have single queue, and if data is received by client then it will be deleted, so other clients won't be having the latest change

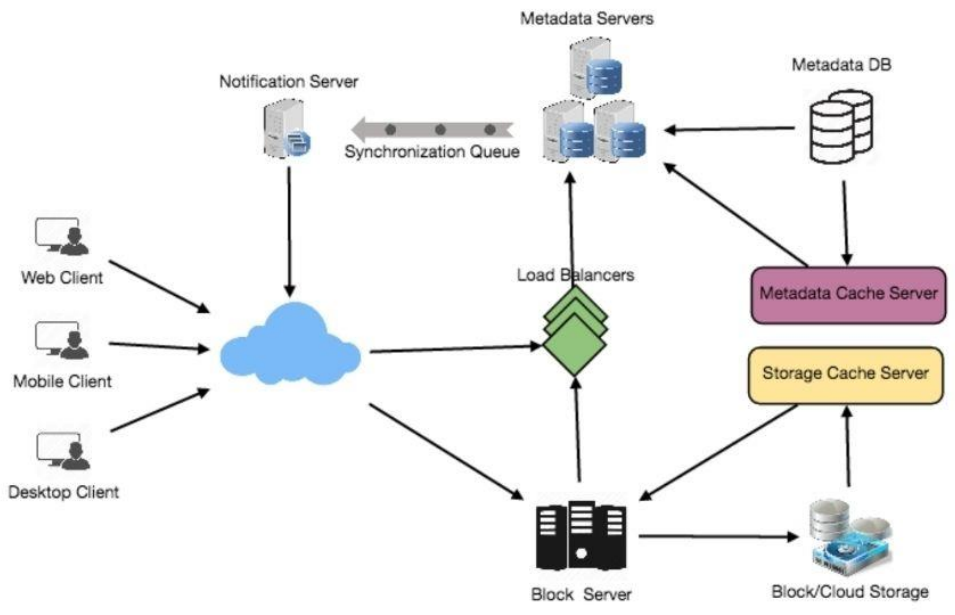
4) Data deduplication
It is a technique used for eliminating duplicate files. we calculate hash and compare it with previously saved hashes to optimize storage

It has 2 methods
a) Post deduplication- As the name suggests we store the copy and then we can have a service which identifies the copy.Not so efficient process as we are unnecessary using the network bandwidth

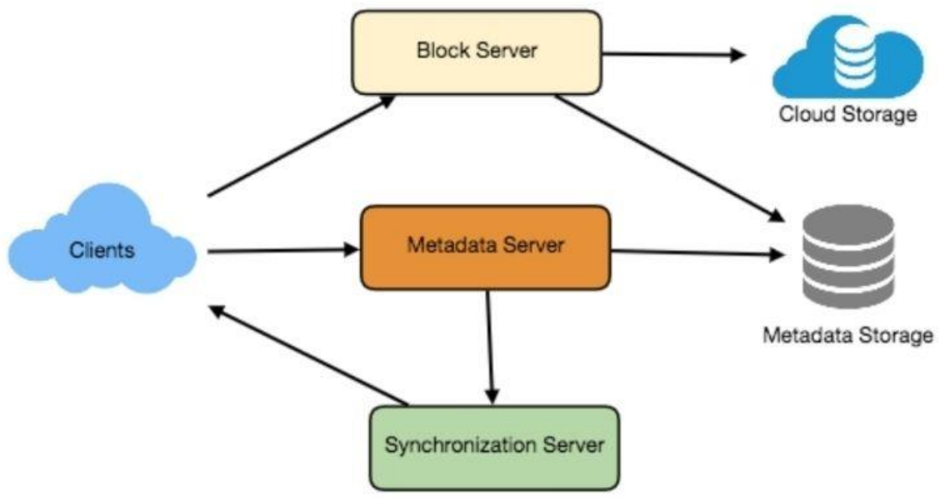
b)In - line deduplication- We can calculate hash at run- time , that is when user modifies the content we can calculate hash and compare with previous hash. If previous hash is present then we just send reference of it in meta-data object rather than sending the whole chunk thus saving network bandwidth and optimizing storage

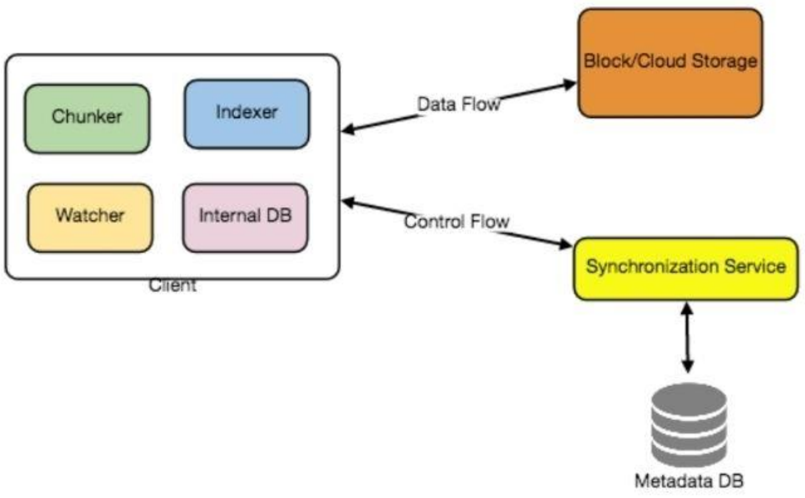
5) Load Balancer:
a) Client -> Meta-Data server
b) Client -> Block server

6) Cache
We can add 2 cache servers (Block cache server) and (Meta-Data cache server)



Detailed component design for Dropbox




Chunker
Indexer
Watcher
Internal DB
Client
Data Flow
Control Flow
Block/Cloud Storage
Synchronization Service
Metadata DB