22/Feb/2022

Metadata Server. The primary role of this server to manage all required metadata information required for files/photo uploaded into system by user.

Every information regarding the files/photos/folders, their chunks, their storage location, number of copies, order of chunks with timestamp etc.

Finally storage of these metadata takes place on database (SQL/NoSQL).

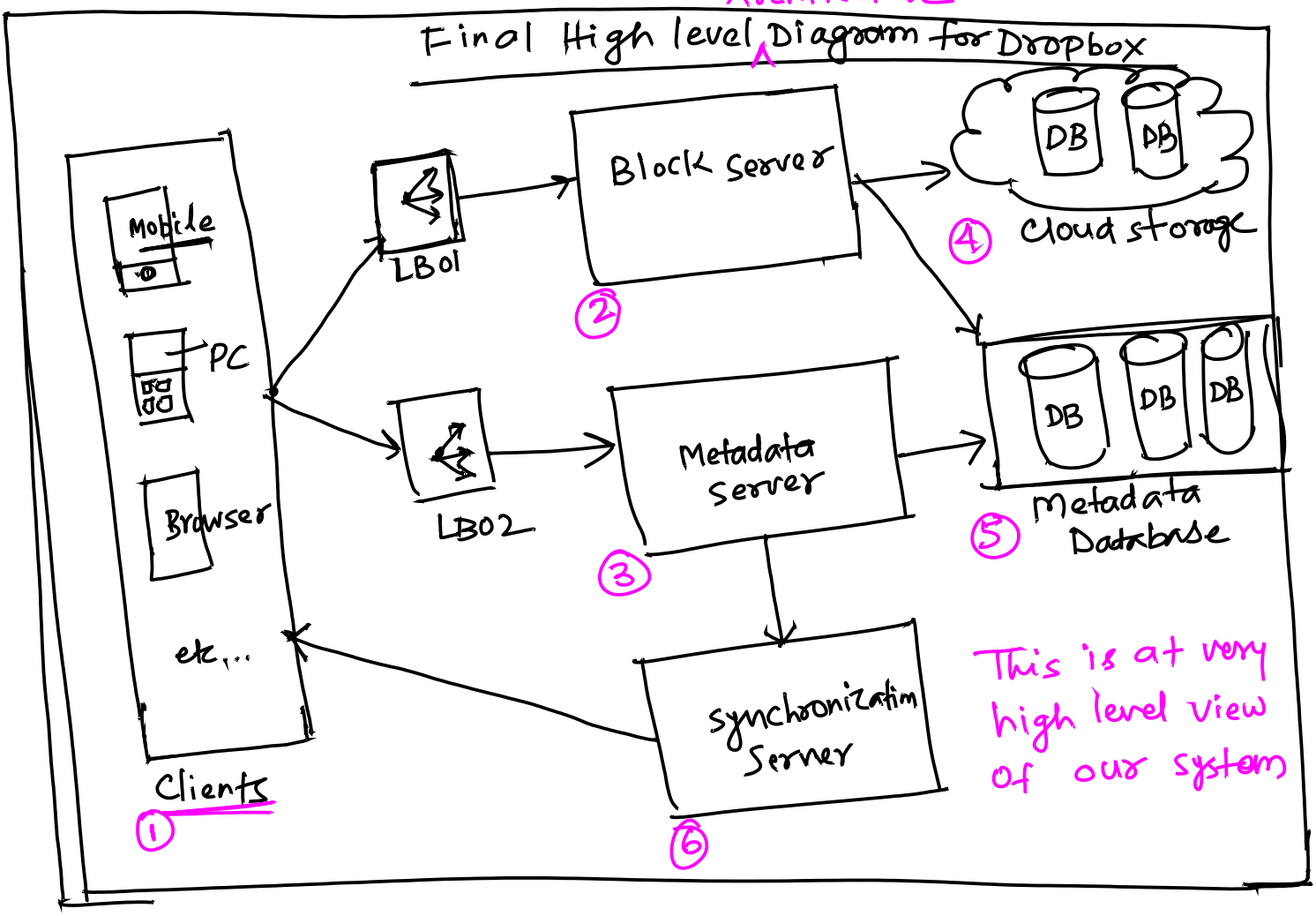
synchronization server. The primary role of this server is in helping to get updated content for User in their all different target device.

for eX: If User A have 03 different device first is mobile, second is PC, 03rd is webApp from Anywhere. Now Imagine you might modified your some of the content inside your dropbox with may be adding some files, deleting some files, modifying some files through your PC. Now how your device mapped folder will get information about modification and auto update for you?

So solving all these concerns we must have some service, which will help all these nitty gritty for all our system User's (400 million)

This is exactly the same, our new service with name "synchronization service" doing for us.

Architecture



So, for now, I feel let's explain first the component briefly to explain their purpose with Interviewer.

① client. The client Application monitors the workspace folder on the user's machine and sync all files/folders in it with the remote cloud storage, Associated with particular user id.

- client Application co-ordinate with Block Server for uploading/download files/folder/photo etc to and fro for user.

- client also interact with remote synchronization service to handle any file metadata updates. As an example, change in filename, size, modification date etc.

Top 03 Key operation of Client App

1. Upload and Download files.
2. Detect file change in workspace folder
3. Handle conflict between offline or online or concurrent modifications.

Let's discuss briefly, how do you handle file transfer efficiently?

- Break each files into smaller chunks
- transfer only those chunks whose ~~overloading~~/ uploading/ downloading / modification have some issue, not the whole file. It help better efficiency & experience to your system user.

- Let's say you decided over chunk size as 4MB. Do we have any data or mechanism to support our decision? there are approach to be used for estimating the Chunksize but those data are dynamic for ex: bandwidth, serever processing capacity, I/O ~~spe~~ of the server device, Throughput of server, Latency of network etc. But more or less these can't give you any definite size so on an average, decision over chunk are copied from standard architecture like Google Drive or Dropbox would be better.

how do you decide, on, Can we keep a copy of metadata with client?

Keeping a local copy of metadata helps us in doing modification locally to save lots of round trip with remote metadata. Here our synchronization service keep to

sync our local data with remote metadata and user will not feel of any stale data at any point of time.

How can client efficiently listen to changes happening with other clients?

Let's discuss and give a shot on this problem.

How client will know is their any change in server?
The obvious answer is client will fetch this information from server at certain regular time interval.
Yes, this could be done

but two key problems

- You are making server busy for nothing
- Also sometime you will face some delay in getting update because of your regular interval gap.
- Also, dependency of one over other may degrade your scale requirement, meaning pulling of information most of time is not scalable so well.

- Than what you will suggest solution of above problem?

OK let me think, for this problem, I am suggesting HTTP based "Long Polling" concept to be used for above mentioned problem.

So what is the working process of HTTP Long polling, with long polling, the client requests information from the server with the expectation that the server may not respond immediately. If the server has no new data for the client when poll received, instead of sending empty response, the server holds the request open and waits for response information to become available. Once it does have new information, the server immediately sends an HTTP/s response to the client

Completing the open HTTP/s request. Now client upon receiving the response, ① Immediately raise another request and then ② process/act upon the response received from Server.

Seems lots of text above, so let me put in small text but more detail form of process into client side.

These are the steps we can think

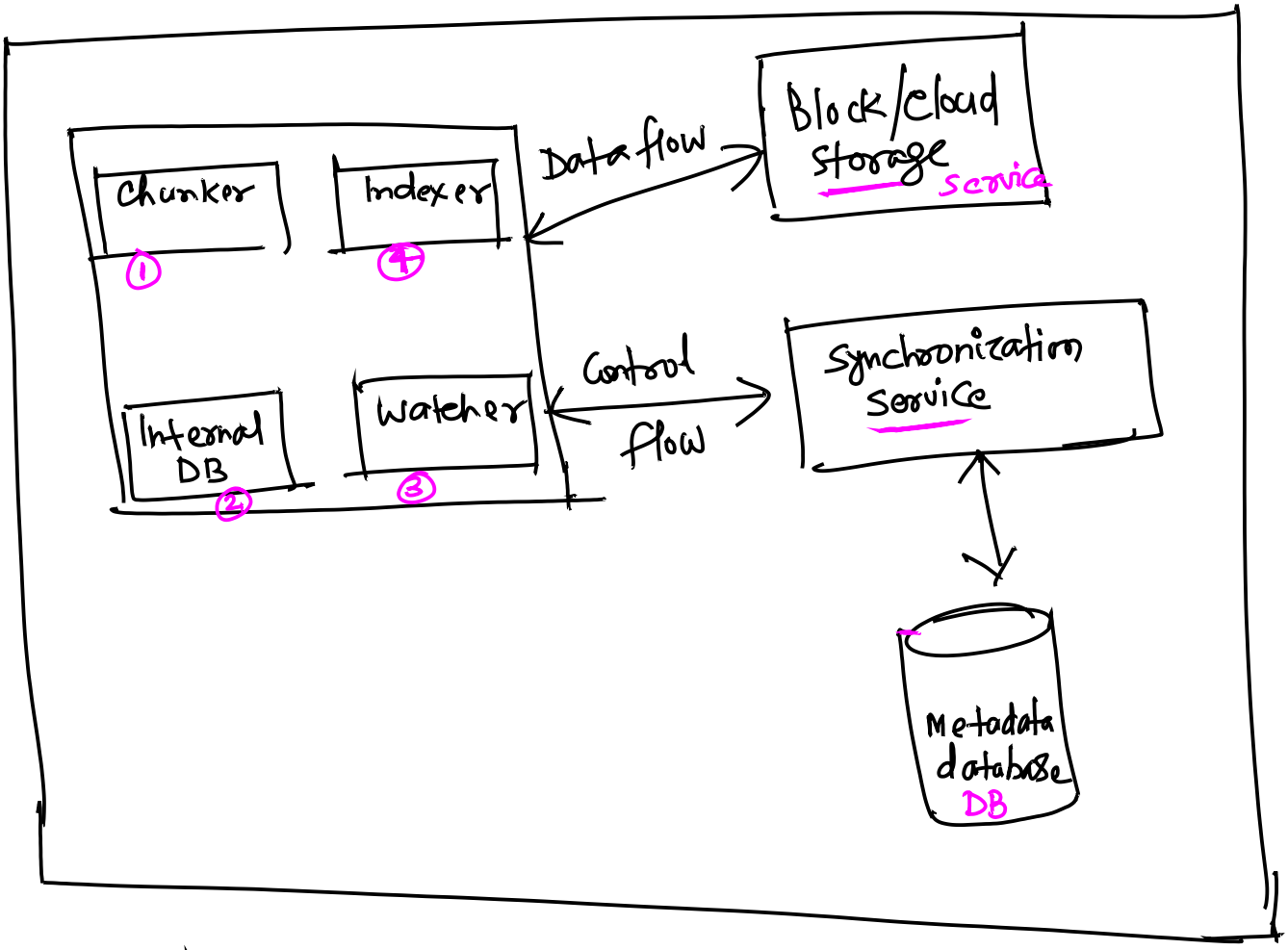
I. Internal Metadata Database will keep track of all the files, chunks, their versions and their location in the file system.

II. Chunker will split the files into smaller pieces like chunks. It will also be responsible for reconstructing a file from its chunks. Our chunking algorithm will detect the parts of the files that have been modified by the user and only transfer those chunks/parts to update and handle the modification. It will help to save the bandwidth and synchronization time.

III. Watcher will monitor the local workspace folders and notify the Indexer for user action. Meaning when user create, delete or update files or folders, watcher listen and inform Indexer. Another key role of watcher is listening of broadcast done by synchronization service.

IV. Indexer will process the events received from watcher, and update the internal metadata database with information about chunks of the modified files.

After update of Internal metadata, alongwith chunk upload/download, Indexer will communicate with remote synchronization service to broadcast changes to the other client and update the remote metadata database.



2. Metadata Database.

- Responsible for maintaining and managing/versioning about files/chunks, users, and workspaces.

- Frankly Speaking Metadata definately require ACID property in transaction, but it also need scale. So here Architect has to work solution with system estimation. Records are in billion and so for scaling of relational database might be challenge. Sometime NoSQL DB sometime not good for ACID like transaction.

  So as an architect we have to balance the different options and document the selected choice

The Metadata Database should be storing information about the following objects

1. Chunks
2. Files
3. User
4. Devices
5. Workspace (sync folder)

3. Synchronization service

This is key component in overall system design.

Before going to explore the role of synchronization service, I would suggest, let's learn first what kind of synchronization requirement this system have.

- One major requirement is if user have created multiple workspace on multiple device then updating content from one workspace have to reflect on other workspace on other devices.

Workspace Synchronization

- Our synchronization service is also have to deal with data transfer requirement for modification of cloud storage data. We can't afford to send the complete file again & again. So you have to create some algorithm to identify the modified chunk and transfer only the modified chunk. This will help in less network bandwidth consumption as well as better performance in data synchronization.