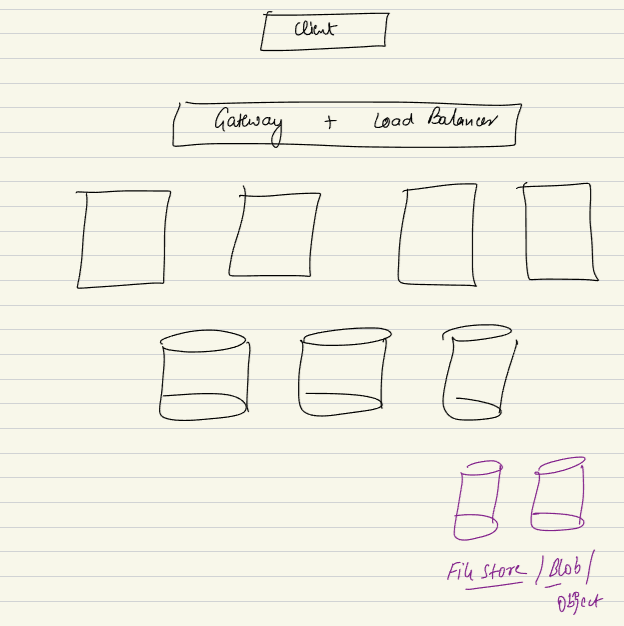


Client interacts with GWY + LB… beneath that we have app server machine and DB server layer.. DB can be SQl/noSQL/ federated DB (one kind of DB for some sub problem, another type for other sub problem), cold storage layer (put older data to maintaining records for long term 10 years)..

Beyond that we will have special data store, we call them as file store, BLOB store or OBJECT store..

If I have a big file, a big text file: log file. A system can put log and it can be very big.. Splunk or normal app, can be in GB and be bigger and bigger. Also if I want to store a movie, that will be in 100s of MB or in GB. That is a file which contains these frames... I cant those frames in normal sql or noSQL. I will put them and read it. But I will not go inside the video file and change a particular pixel. DB like sql/ no sql required to access point access and point update. Nature of accessing a big Text file or video is either we access the entire file or don’t. we don’t go at 20th frame and update. Its more like an object or file we dump in a big machine. So big file are not necessarily (rarely we do occasionally) they will be stored in file store/ object store or S3.

More likely to store as one chunk in a machine. We can persist docker image just like a file as some data in file storage/ blob storage.



**WHY not SQL or no SQL?**

We don’t want to store entire movie file of GB in size not in the Table. Nor we devide the video file into frames. A video is nothing but collection of frames. Bunch of images merger together. We wont store as we don’t do any processing on the images. There will be complication to store them as sQL.

No sql: we don’t store them as Key-Value in Redis. K-V for smaller data.

Wide-Column DB: a no sql db like WC like Casandra, every entry will be a combo of column families no point we would devide video in frames.

Document DB: makes some sense, as a Doc DB is nosql, store everything in once and ppull in once,

To store big file of 50 TB.. say 1000 of such files, a single achine will not be able to store all of these 1000 files. So right way to store them is not sql/nosql but rather a distributed file system.

**A common distributed file system is HDFS: Hadoop distributed file system**. Hadoop is way of managing/ imagining big data. HDFS is file system Hadoop uses.

If a machine cannot store all my data, we use shard, use bunch of machines. So we start thinking of cluster system, distributed system. When we think about cluster of m,achine. We come up with logical disctributed machines, when pieces are spread across. We still can access the whole thing.

**AWS S3:**

**Amazon Simple Storage Service (Amazon S3)** is an object storage service that offers industry-leading scalability, data availability, security, and performance. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere.

To store big file, image, use my S3 service, You send me stream of data, I will store that file, video, image not in DB but a service S3 I will push somewhere. S3 – is managed service.

Azure: blob. GCp: has parallel store.

Don’t get confused with CDN: cdn is a special kinda system which acts as a Cache. Where company buys hardware amdn manages infra all across. Another company uses the CDN so that his user can access data like Netflix.

However S3 is different, where we store data with S3.

Example - Scaler has a lot of media files. Lot of videos. Big test cases. Class recording. Not a lot of continuous process can happen on video. Once video created its stored. A 3hr video can be 3 GB of size.

10K video can be 30TB. Also has ads on Youtube. These data, I wont store in sql/ no sql. Also I don’t want to lose it. I want to REPLICATE the data. So 100 of TB of data or more. These data either scaler can use a HDFS cluster in my end kind of system which is self hosted. Or they can use AWS s3. Whese scaler will keep on pushing and when I want to read.

This is different than CDN. CDN will be used if I want my user to access it esily. If scaler has user across india, scaler can ask AWS CDN to store copy of video on your EDGE machine. Even in Live class a CDN will be used. That CDN is for better service to end user. Cdn to show it quicker.

which file system does those edge servers use? CDN has their own file system.

S3 for Scaler to having that data.

If I cant store everything inside a single machine.. when every file is 5 GB and I want to store 100,000 files. 100 of TB of data I want to store, this cannt come in a single machine. Also this 100 TB is just the baseline I will also have to replicate the data to ensure the is not lost. So data backup available when some machine has issue. So single machine cant handle it. I need to build something we can think of files that are big. Big file ssuch as big LOG, TEXT, Media files, also Neural Netorl, NNL models they are storing data in the form of weights billion of weigth, every weight is byte of many TB of size.

So we think of S3 or HDFC. When we cant imagine a single machine to store it, I imagine Cluster of machines. When we devide the data and computer among themselves all them combined we call cluster fo machines. Ina cluster we will store this.

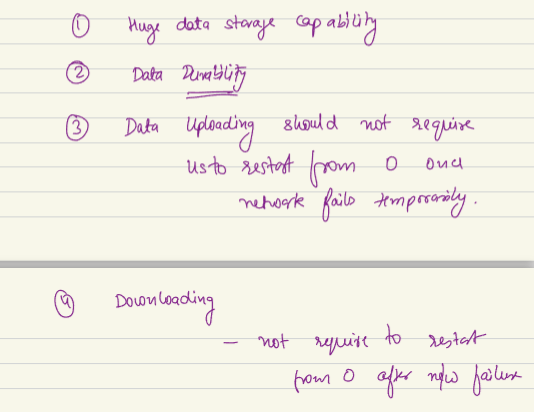
So I want below:

1. Hige data storage capacity
2. Data durability: cant loose data. File cant be lost.
3. Want data to be consistent, you store something and while readding data is something else. We wont change a file, but whatever we store needs to be durable.

Lets say I want to store 5TB in cluster, gonan happen in network.. To transfer it over network will take some time. Network is always unrealiable. Cant be up always. At times network will break. 5 TB can ake few minutes. But if network breaks I don’t want to updload whole file again when network is back up. If 1TB is uploaded I want to upload after that 1TB which is already uploaded.

1. Data Uploading should not require us to restart from 0. Once network fails temporarily. So whatever data is already uploaded, if not completed, I want to upload from there. Upload cant be atomic event.

Also for downloading, I want the following possible things, for downloading also even if something fails in between continue from where it failed. It should not require from restart from 0 after network failure.

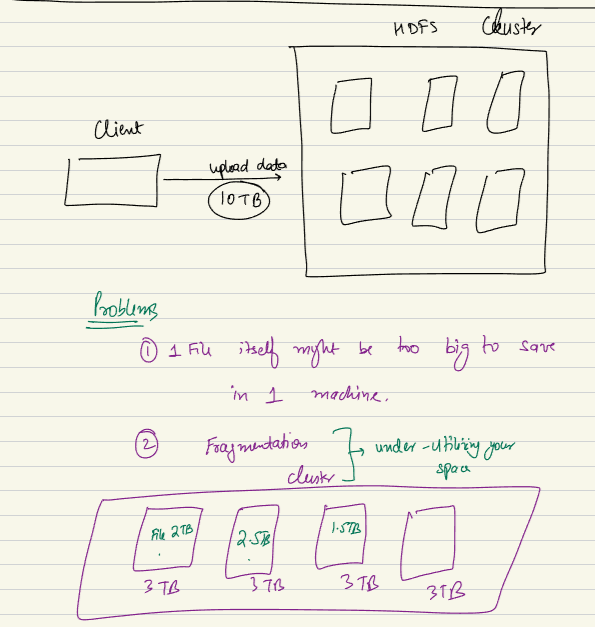


**HDFS CLUSTER:**

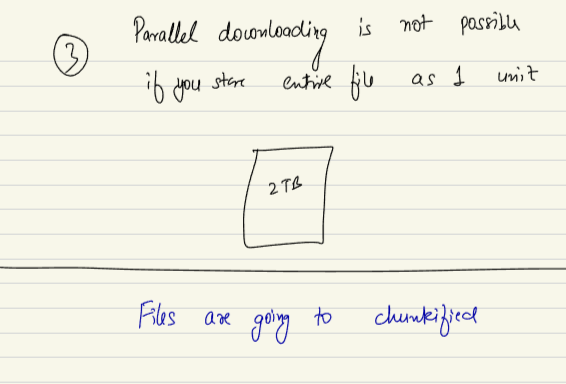
I am a client of HDFS service,, I will us ethe service. I want to upload a 10 Tb file. File can be so big it might not be stored entire file in a single machine.

If my machine is 3 TB in size, to store big big file the fragmentations will be problem.

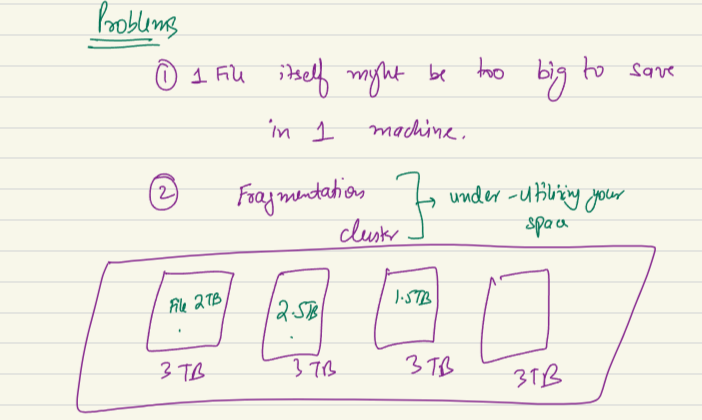
Lest say in cluster of HDFC each machine will store 3 TB of data. In first machine the file comes is 2 TB. Next file is 2.5 TB I will use next machine to store as I cant fit in 1st machine. So there is problem of fragmentation. When you store big blob files problem of fragmentation will come, you will underuse your storage.



If my files are treated, stored, downloaded not as a single file but as chunky files. Every file will be devied as small small chunks. Devide as smaller piece, each piece is chunk. Torrent like system.



I have HDFS system, I am a client who want to uload data, I wantto save a 10 TB file. Idea of chunky file means… rather than storing file107 in hdfs cluster. Rather storeing this file as one. I will arther split the file in chunks. 10TB file gets devided into chunks into 1 GB each. So 10,000 chunks of 1 GB each. So I will devide the file across many machines… this file is combo of small units. Each unit is..



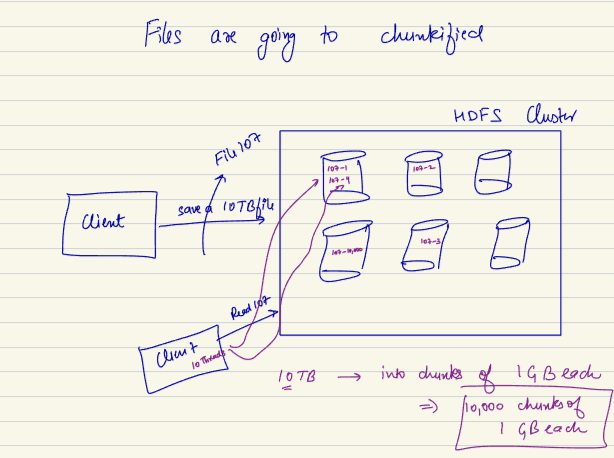
Frag: I had 3 TB, I store 2 TB and 1 TB is waster. If I store the 1 GB files I reduced the chance of fragmentations. There will be still I will waste lesser space compare to bigger space.

Parallel downloading: if I divide the file in 10k pieces, if a client want to download... and has 10 threads which can parallel read something. Each 10 threads can read diff piece of 107-1, 2nd thread can read 107-2. 10 threads read 10 chunks. The moment one thread can finish can go and request new file. If more than 10 threads I can do even more. Yes there can be network bandwidth…. A upper band.

I can keep on filling missing pieces. Every component come at me as ID. If network threshold is bottleneck, that can be problem. Usually is rate at which one machine can send data..

Here client is not getting the file at the same time watching t. assume client is storing component and client arrange them based on ids in sequential manner. Then get the file together. if network goes down, then client starts from last id.

So solved all 3 problems..



Lets see more Praises of fragmentation..

Not having to REDO everything…

Say I have a file of 10 TB. How small I want the chunk or how big it’s a spectrum… from 1KB to 10 TB..

If I have bigger chunks, I am not utilizing the idea of fragmentation/ chunking… if I have more smaller chunks. Problem of 3 is solving more.. HLD its not a free choice in SYSTEM.. always a trade off…

When chunks are small, probvle are simply too many chunks to manage… if I have a 10 TB big file I want to have 1 KB of chunks.. how many ???

Number of files=1KB10TB​

First, convert 10 TB to kilobytes:

10 TB=10×1,024×1,024×1,024 KB10TB=10×1,024×1,024×1,024KB

10 TB=10,737,418,240 KB10TB=10,737,418,240KB

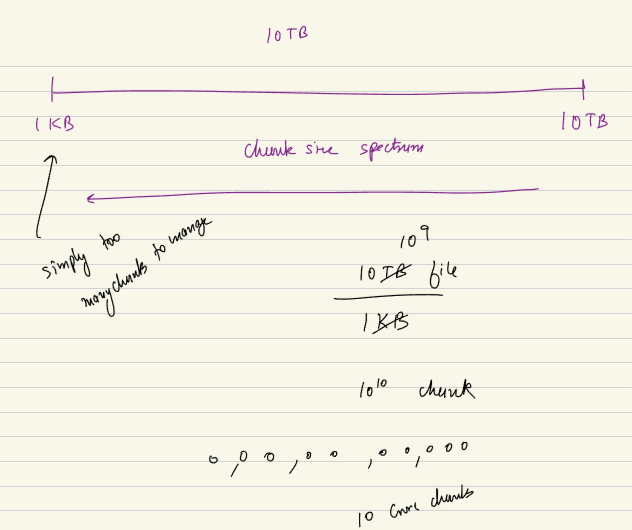
Now, calculate the number of files:

Number of files=10,737,418,240 KB1 KBNumber of files=1KB10,737,418,240KB​

Number of files=10,737,418,240Number of files=10,737,418,240

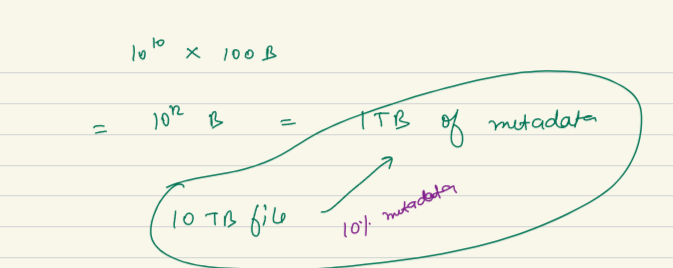
So, if each file is 1 kilobyte in size, you would have approximately 10,737,418,240 files in

10 crore chunks..



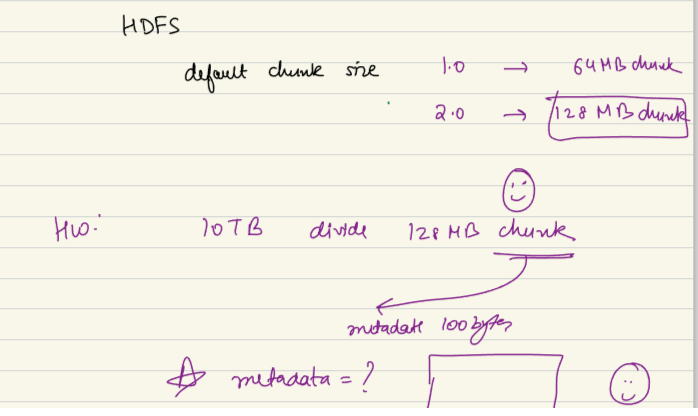
Metadata will be 10^10 enties will eb stored.

File id – 8 byte Chunk id = 8 byte | All can take 100 byte.  
If I have 10^8 rows it will be 1 TB of metadata. | For a 10 TB file I am storing 1 tb metadata……. Bad Architect.. So we cant keep on going left side of spectrum.. its a trade off



HDFS: being a designer they have decided a default chunk size.. many analysis. Hdfs v1.0 has decided 64MB chunk.. 2.0 version decided 128 MB of chunks.

When hdfs 1.0 coming up they had 64 mb of chunk. Now their chunk/ SPLIT size is now 128 MB… it is in a way sign of time, as time goes by data is going bigger and bigger… with machine learning model and more and more values in them this is going bigger. As of now 128 mb is latest chunk/ split size.



Now, calculate the number of files:

Number of files=10,485,760 MB128 MBNumber of files=128MB10,485,760MB​

Number of files=81,990Number of files=81,990

First, calculate the total size in bytes:

Total size in bytes=81,990×100Total size in bytes=81,990×100

Total size in bytes=8,199,000 bytesTotal size in bytes=8,199,000bytes

Now, convert the total size to gigabytes:

Total size in gigabytes=8,199,000 bytes1,024 MB×1,024 KB×1,024 bytesTotal size in gigabytes=1,024MB×1,024KB×1,024bytes8,199,000bytes​

Total size in gigabytes≈0.0076 GBTotal size in gigabytes≈0.0076GB

Therefore, the total size for 81,990 files, each with a size of 100 bytes, is approximately 0.0076 gigabytes.

**HDFS:**

Hdfs has data nodes. 2 kind of nodes. Data node and Metadata nodes..  
Data Node: The places where actually the data/ file will be stored…

Other kind of machine HDFS has called as Name node.. A name mnode means the machine where I will store entire MetaData.. when we divide a big file to small files I need metadata storing machines.

Name node: STORES METADATA… is a single machine.. which whenever I hear a single machine I can imagine SPOF. To solve singl;e point of failure they agve option of making master-slave. Or the basic default option, name node following the active-passive config.. 1 machine which is acting as a main source. Other nodes are right now as stand by. We call them standby or passive. Primary - Standby pair.

Active ka data is copied, when active dies the passive will take place and becomes active name node.. all rwequest comes to active name nodes, and data is copied to active. When active dies passive becomes an active.

**is active passive nodes different than master slave architecture?**

Master-slave means an active master and active slave. All the machines are active. Slave helps with reads. Slave assisting master with reads.

Active-passive: only active do read and write. Passive are just copies and waiting active to die.

Usually HDFS donot prefer master-slave, prefer active passive. As the reads are very small reads. A single machine can do it. It just want to make sure data is consistane. Bothe user gets same data. They try to be consistence. So try to be active-standby/ passive.

You can use Kafka like queue in between so the load is reduces… read about it.

**What metadata will be stored??**

I should be storing the file id, for the file how many no of chunks I have created.. timestamp it was stored. This kind of info will be in metadata.

Also there will eb ariable of chunk size, whats the configuraton of chunk id. Here 128 mb.. it is configurable. By default its 128 mb.

For this file id, for this chunk id, at what place you are storing the primary copy, where you storing 1st replica and where 2nd replicat.. if my replication fator is 3 the chunk will be stored 3 times. Again it is configurable.

For file 107 for first chunk the 1st replica stored in machine C, E A machine

Same for 107 2nd chunk is stored at machine M, O, P

All these will make system realiable.

looks like SQL data in namenode.. metadata is always like SQL.. but using diff SQL use NameNode of same Product HDFS.

No chunk has info of next or previous chunk. But metadata knows everything..

Usually S3, which is like HDFS ka cloud version…

**RACK AWARE ALGORITHM:**

A data center is nothing but a big building, it has many halls. Inside each halls there are different RACKS has different machines.. each of them are called RACK> all machine in rack share the same network line/cable, same switch… so probability of machine at same Rack going down is higher.. if a rat chews the cable all machines will be disconnected.

All machine in same rack shares same network. Goes down together. This is why system like HDFS and S3 try to keep secondary machines separately..

So all primary at same rack. All secondary at diff rack etc. copies are stored at diff rack so they do not go bad at same time.

Availability Zone: I have a data center at some part of city, another part in diff city, if flood comes at a place other place ka data center is fine. A system like HDFS uses RACK Aware algo.

<https://www.reddit.com/r/SiliconValleyHBO/comments/4oxy1j/this_scene_had_me_rolling/?rdt=33619>