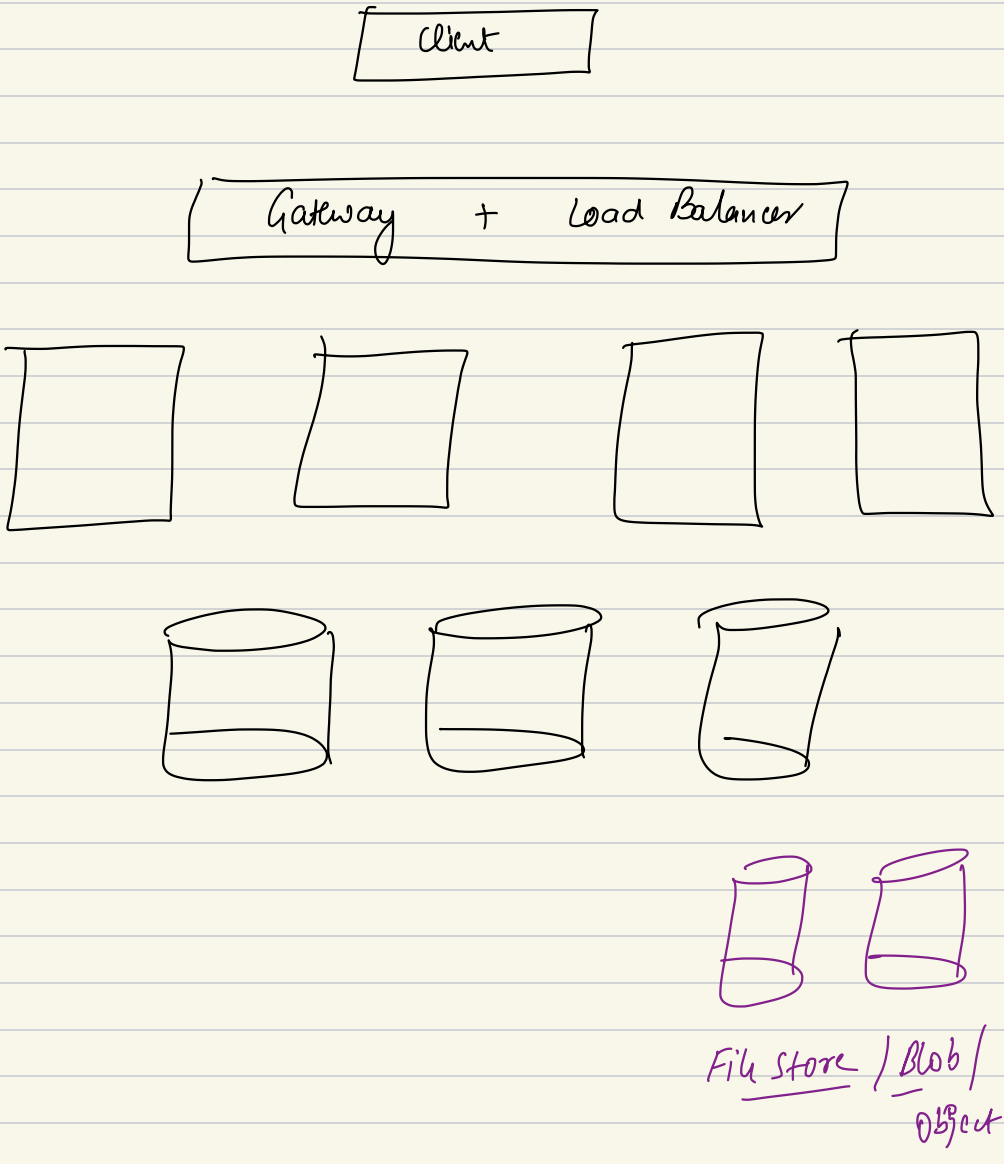
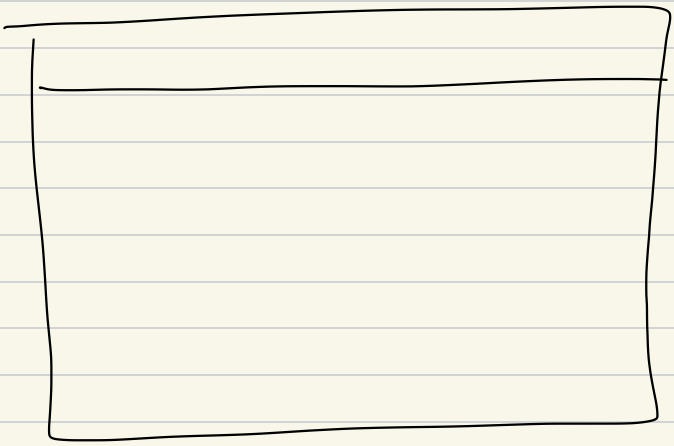
22 | Dec | 2023

STORING LARGE FILES | Quad Trees



Why NOT a SQL DB?



NoSQL ✗

distributed file system ☺

HDFS

AWS S3

Ep Scaler

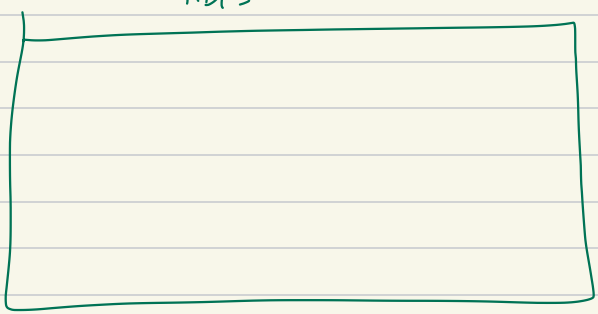
3GB X 10,000 videos

3GB x 10^4

30TB

replicate

HDFS



Or

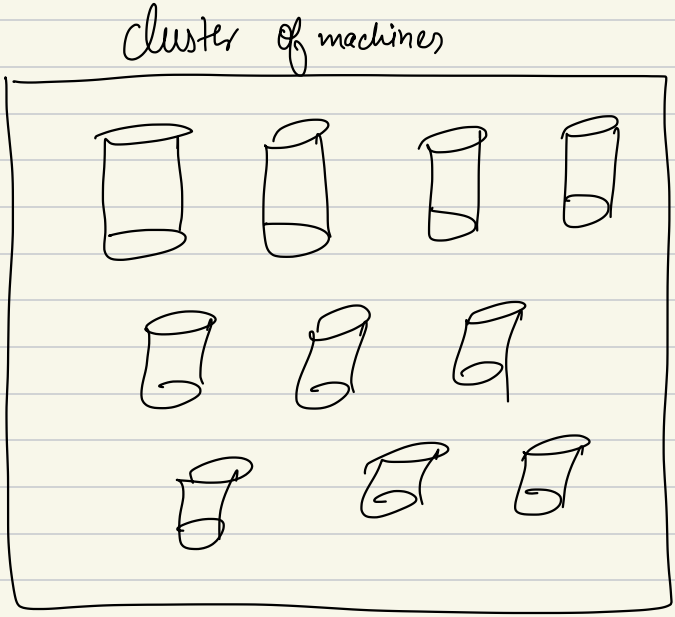
AWS S3

Files that are big → Text Files, Log Files, media files (videos, images), neural networks

ex. fun GB

~5GB × 100K

≃ 100TB



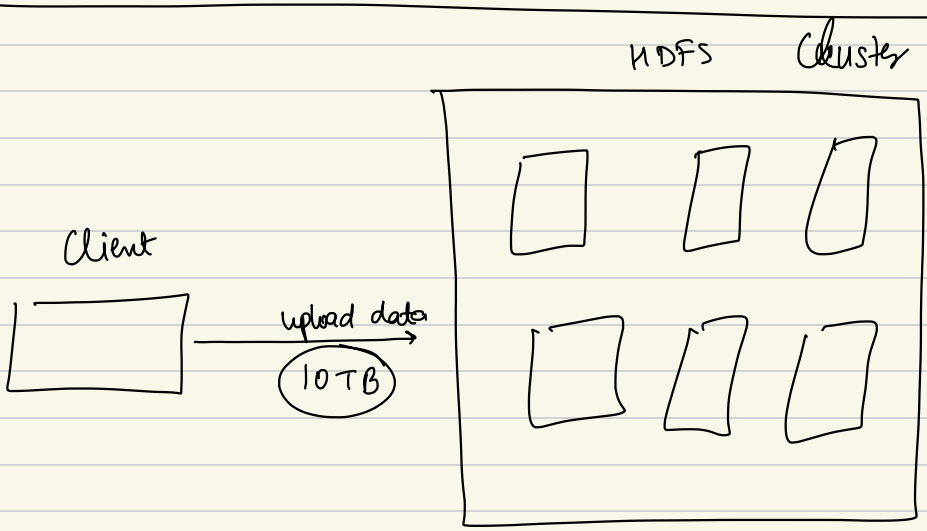
① Huge data storage capability

② Data Durability

③ Data Uploading should not require us to restart from 0 once network fails temporarily.

④ Downloading

– not require to restart from 0 after new failure

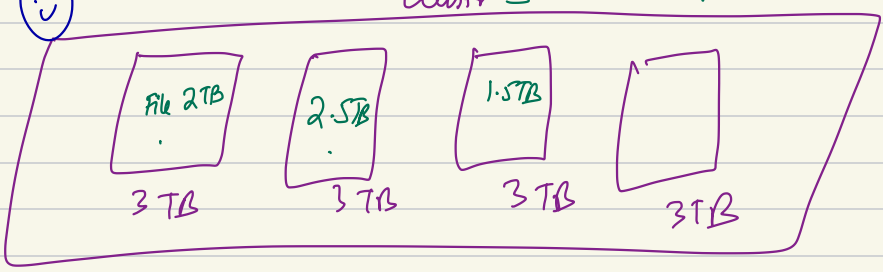


Problems

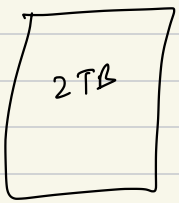
① 1 File itself might be too big to save in 1 machine.

② Fragmentation → under-utilizing your space

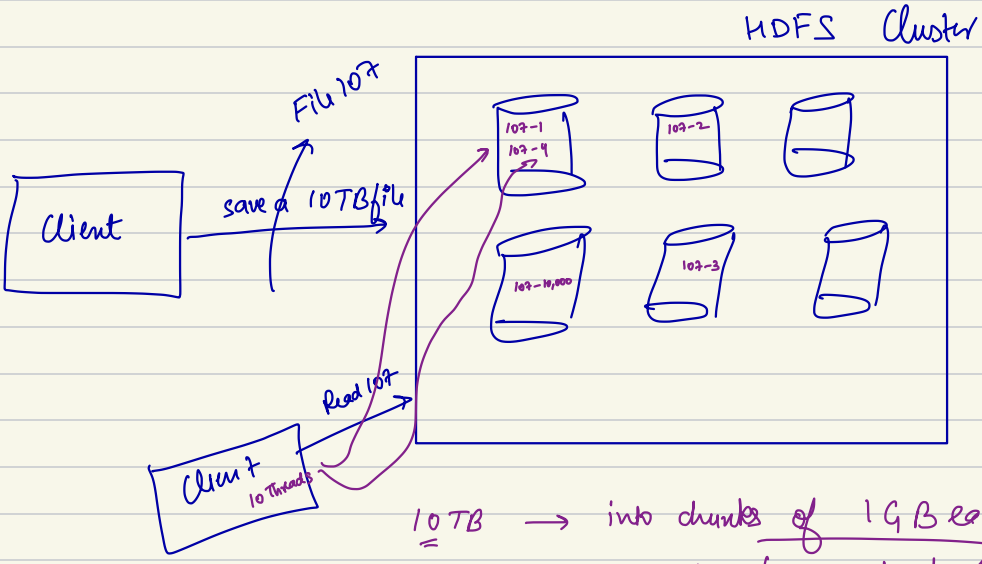
cluster



(3) Parallel downloading is not possible if you store entire file as 1 unit

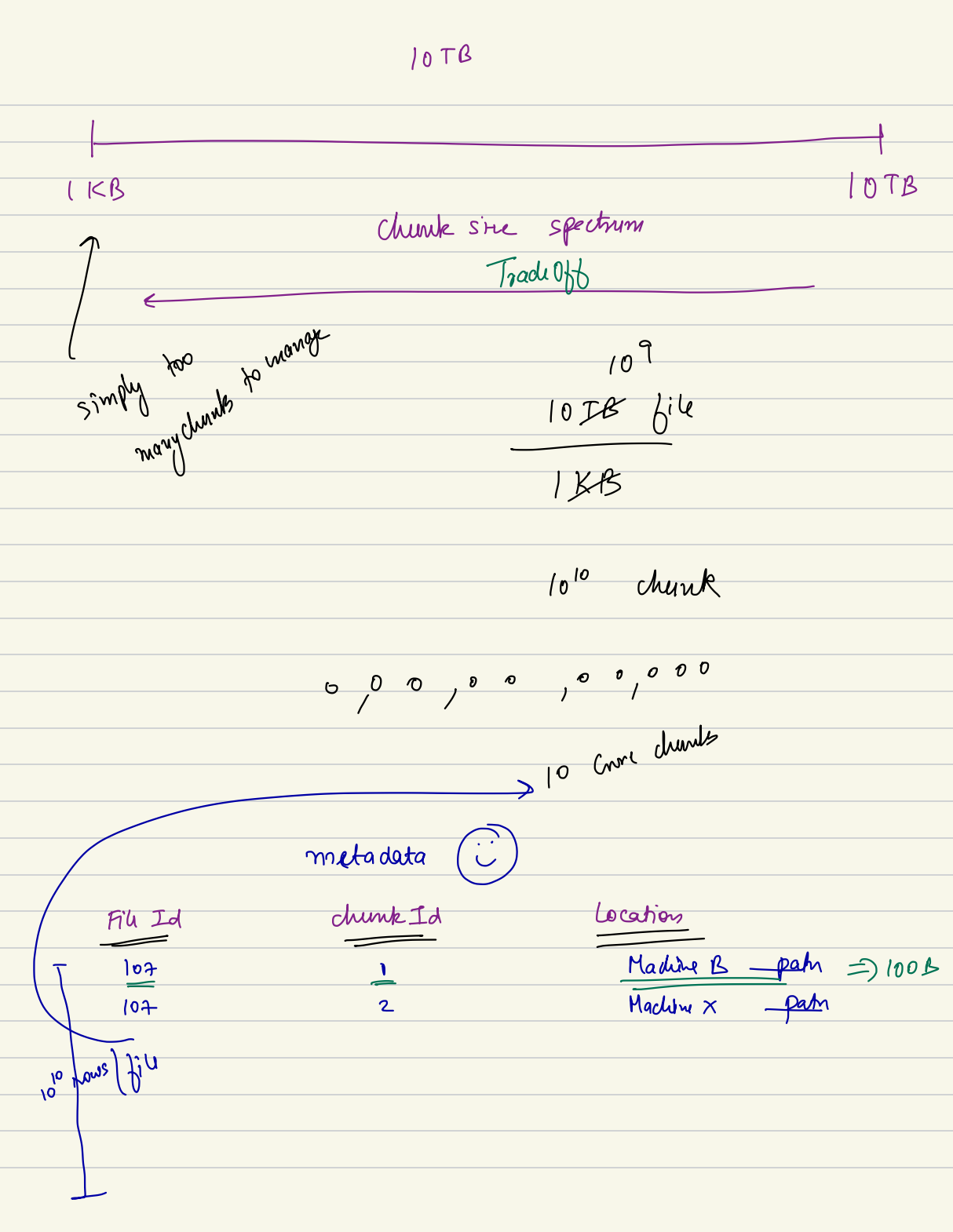


Files are going to chunkified



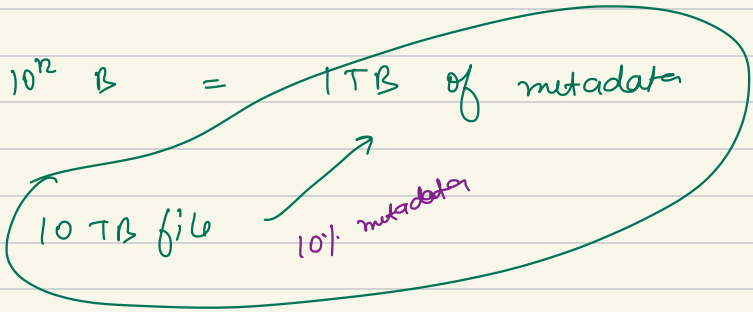
10TB → into chunks of 1GB each

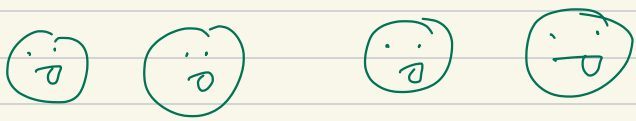
⇒ 10,000 chunks of 1GB each


10 TB
1 KB
10 TB
chunk size spectrum
Trade Off
simply too many chunks to manage
10^9
10 IB file
1 KB
10^{10} chunk
10 Crore chunks
metadata
File Id
chunk Id
Location
107
1
Machine B path
100 B
107
2
Machine X path
10^{10} rows | file

$10^{10} \times 100\,B$

$= 10^{12}\,B = 1\,TB$ of metadata





HDFS

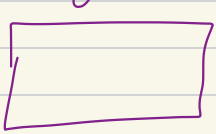
default chunk size 1.0 → 64 MB chunk

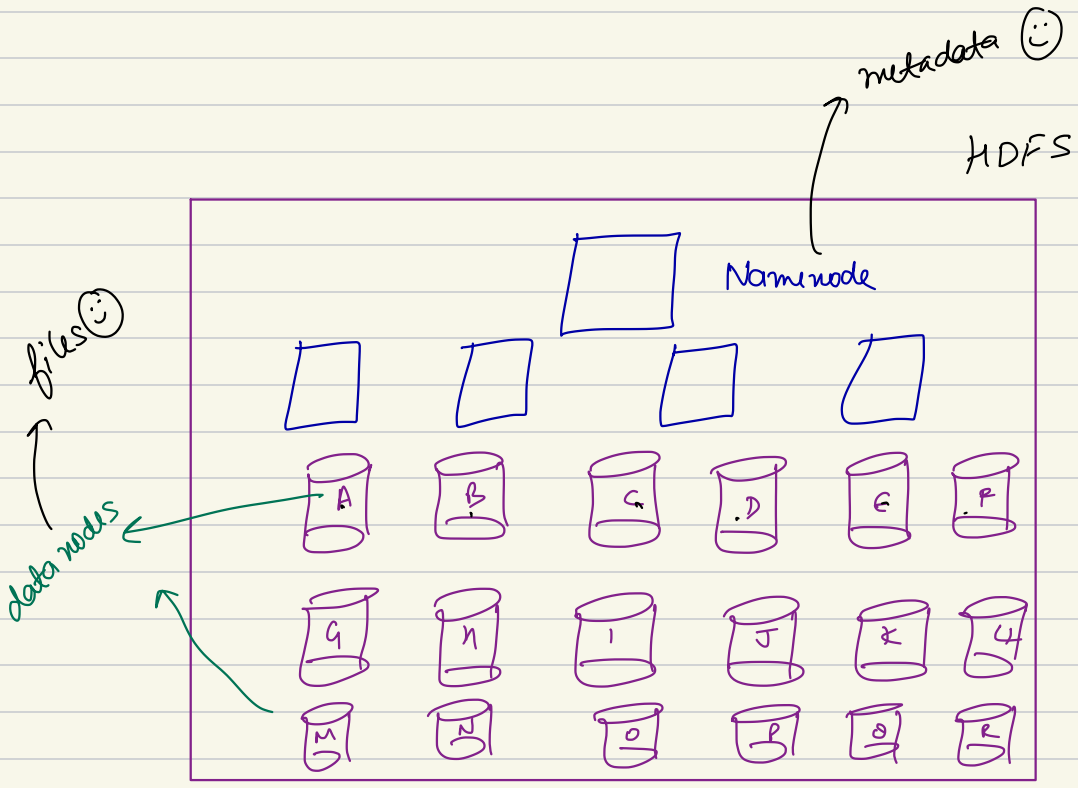
2.0 → 128 MB chunk

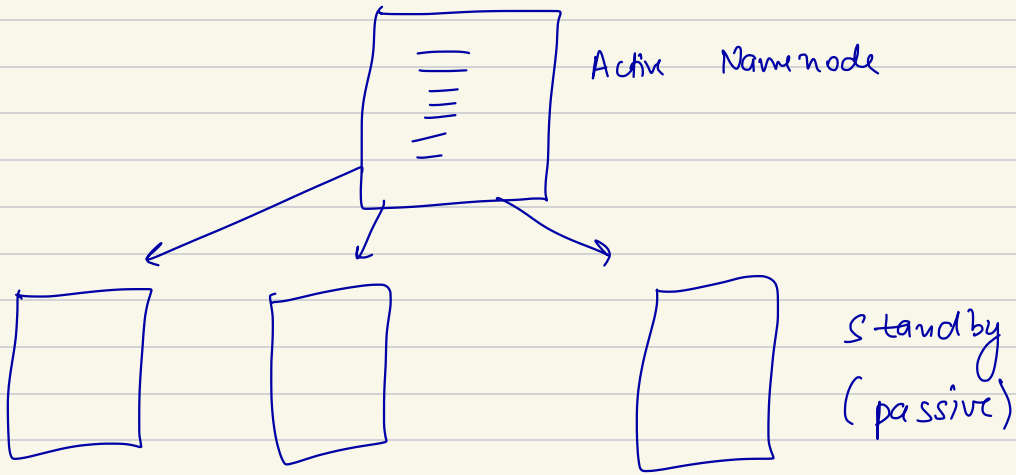
HW: 10 TB divide 128 MB chunk.

metadata 100 bytes

metadata = ?




metadata
HDFS
Namenode
files
data nodes
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R



metadata

	File Id	# chunks	Total Size	Timestamp
(I)	107	4	400MB	

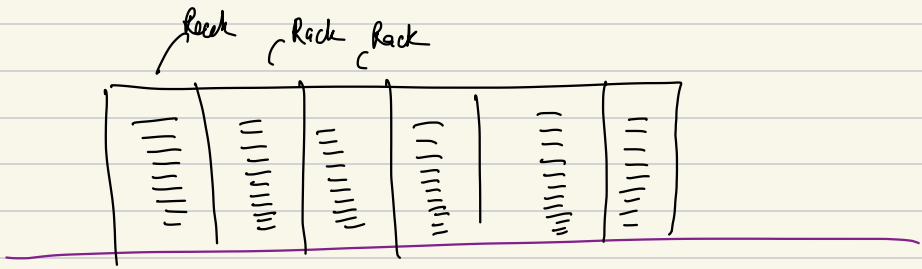
(II) chunk_size = 128 MB ☺

(III)

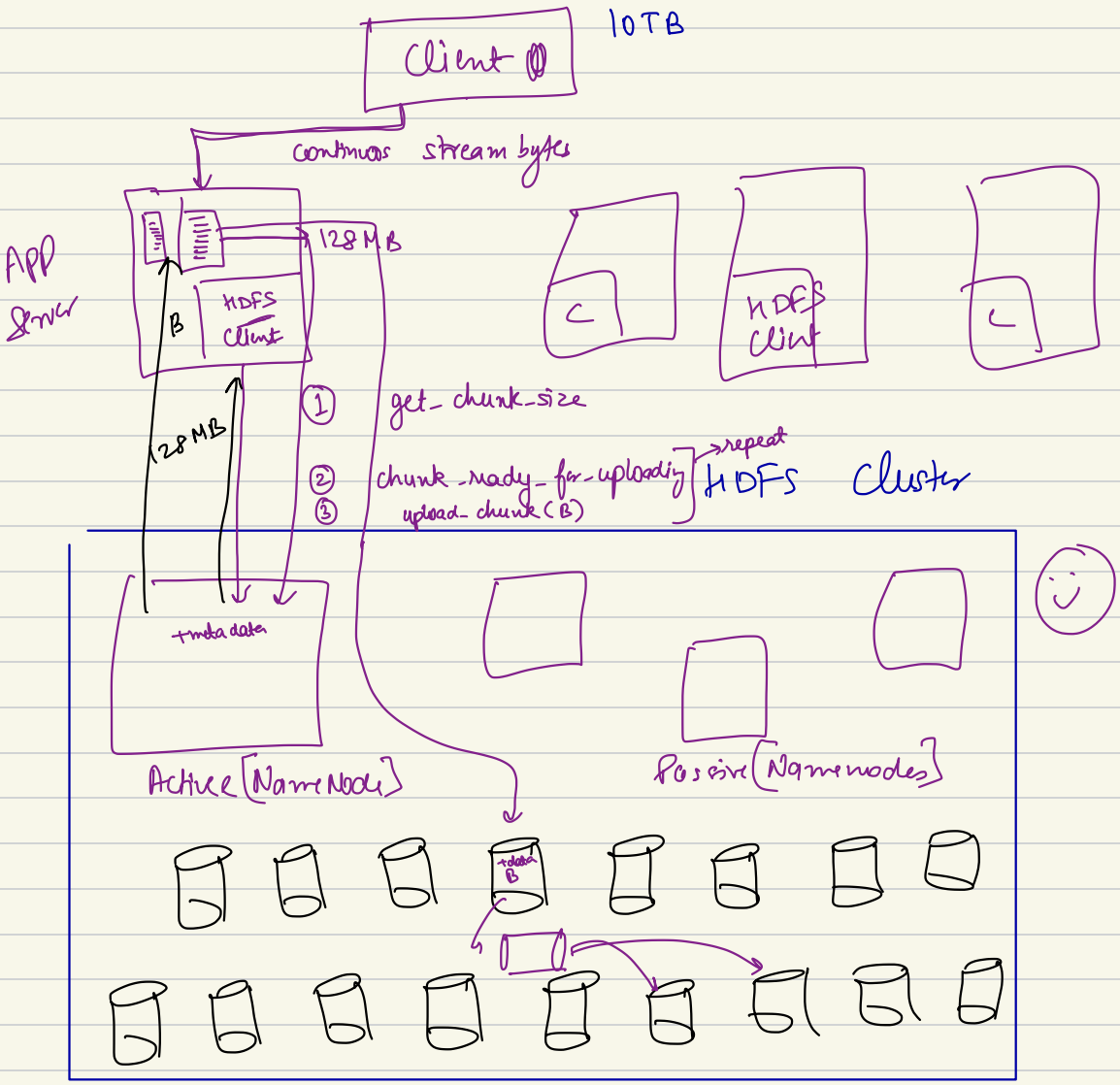
File Id	Chunk Id	Primary Copy	1st Replica	2nd Replica
107	1	C	E	A
107	2	M	O	P
107	3	A	D	F
107	4	C	E	H

More Reliable

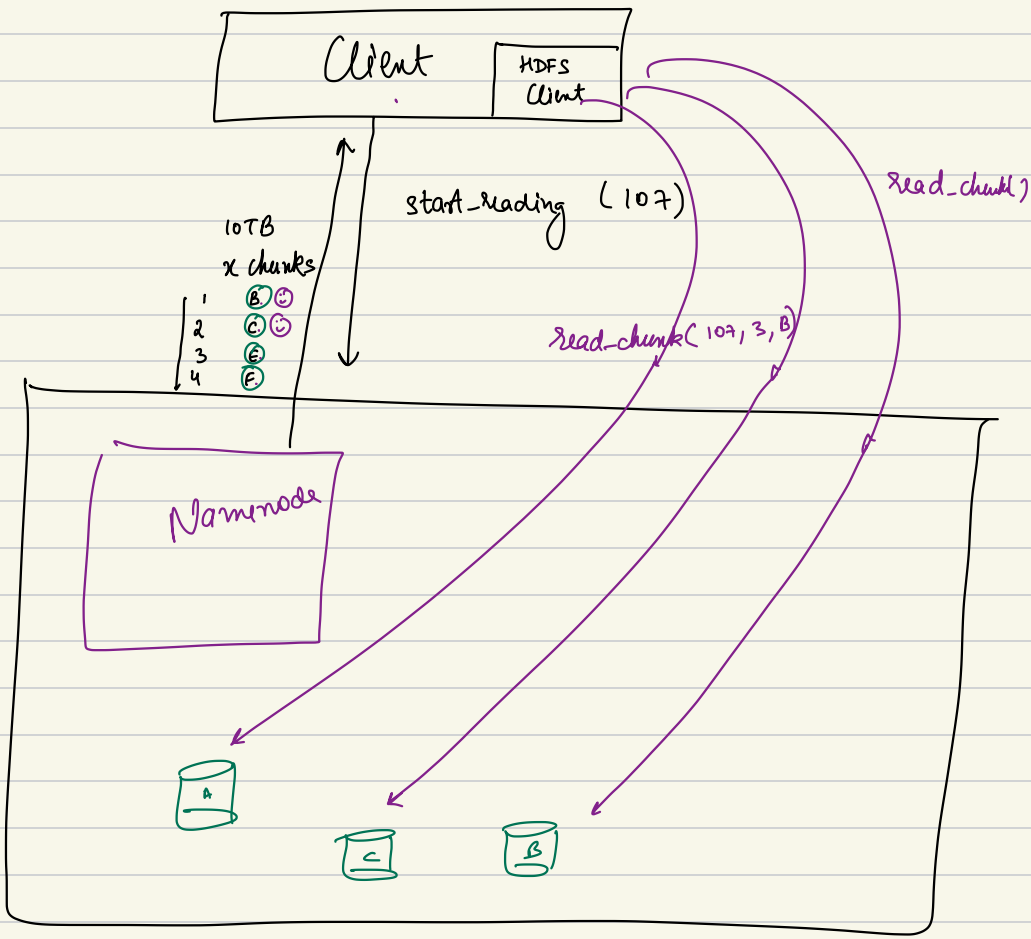
Rack Aware Algorithm

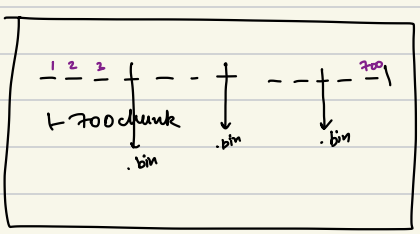


Write Flow | Upload File

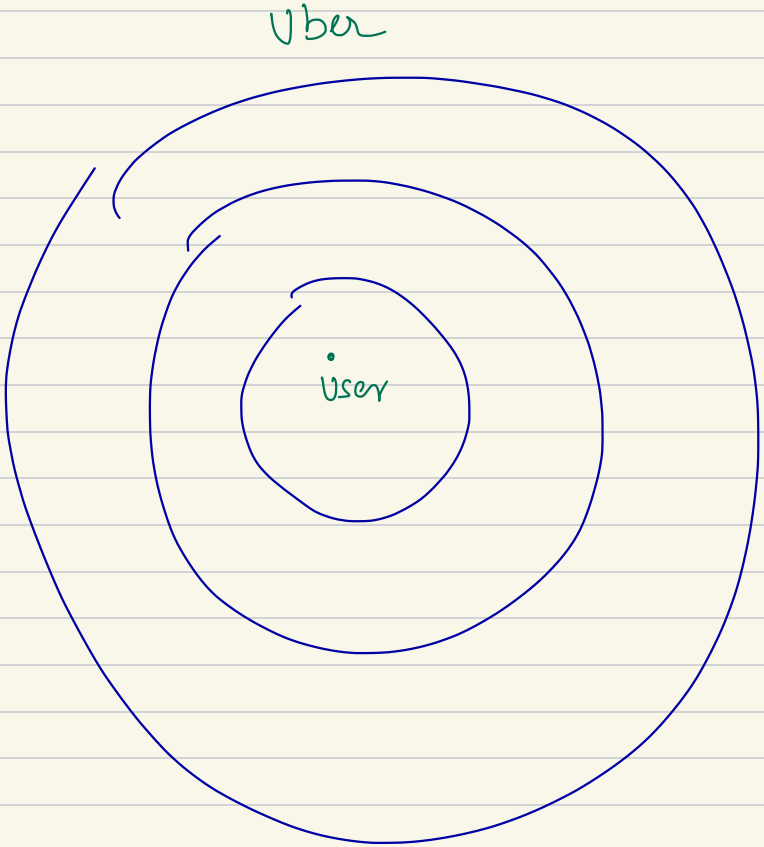


Read Flow | Download



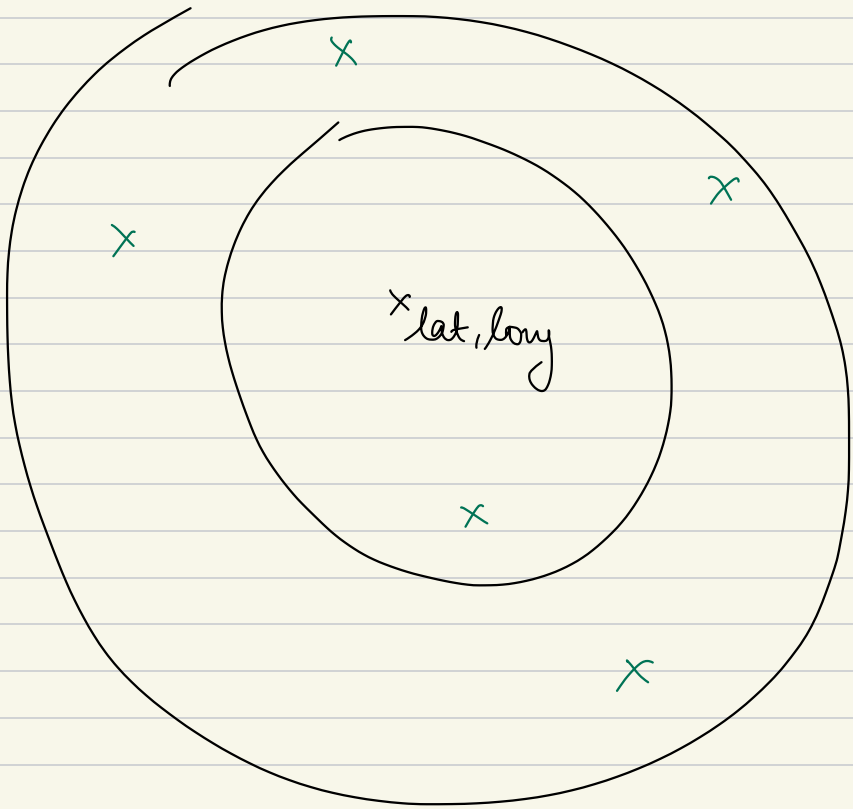


QUAD TREES



google maps

barbers shops near me

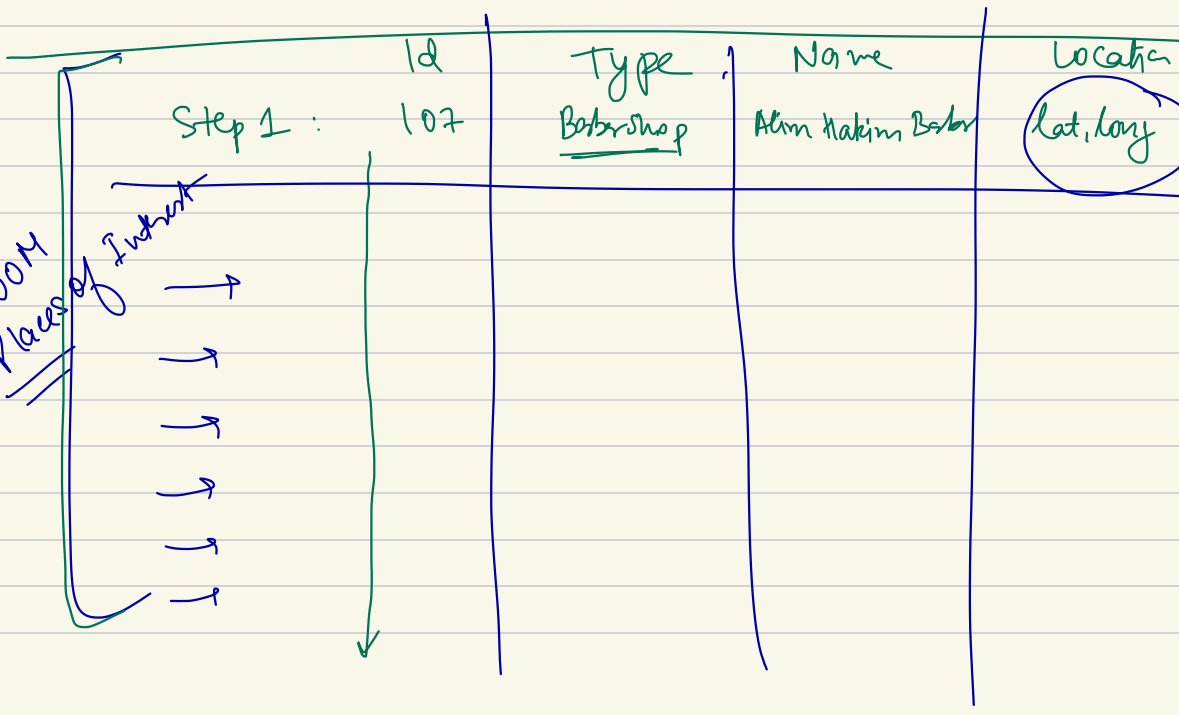


Places near me

Place of interest near me

- Temple
- Restaurant
- Metro station

pt (lat, long)

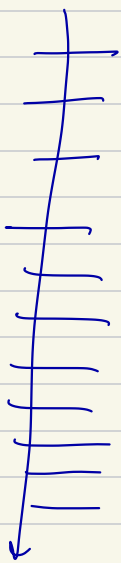


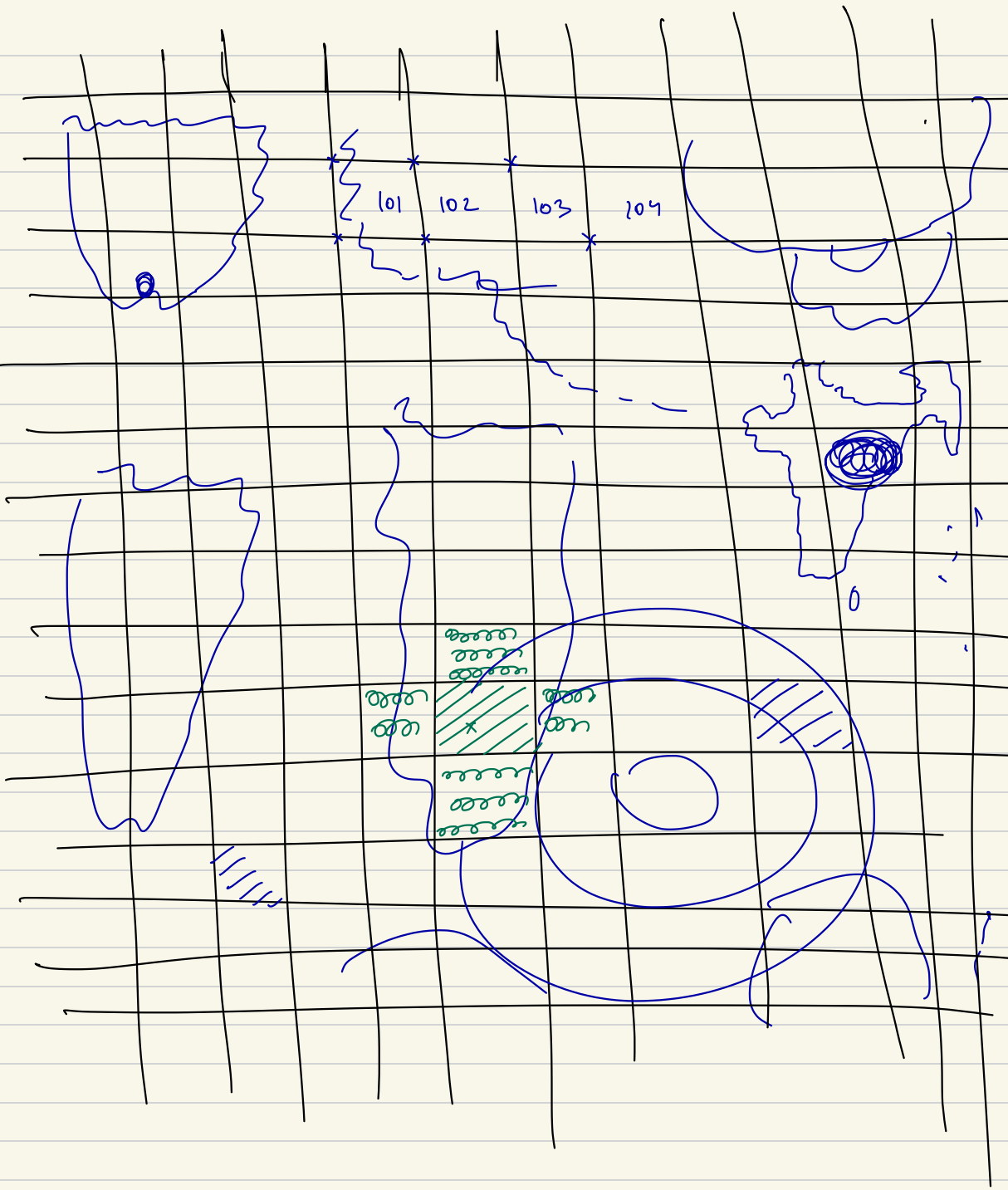
Query : X nearby petrol pumps

(lat, long)

$\sqrt{(v_x - l_x)^2 + (v_y - l_y)^2}$

v_x , v_y




101
102
103
104

Iteration 1

Simple indexing OR sharding will not solve the issue as; even after it you will have to consider 10s of Thousands of comparisons.

⇒ Normal SQL / NoSQL dBs won't work ☹

Iteration 2

Divide the world into grids; and then based on the query, identify grid-id and then perform a place of interest search only in that grid cell

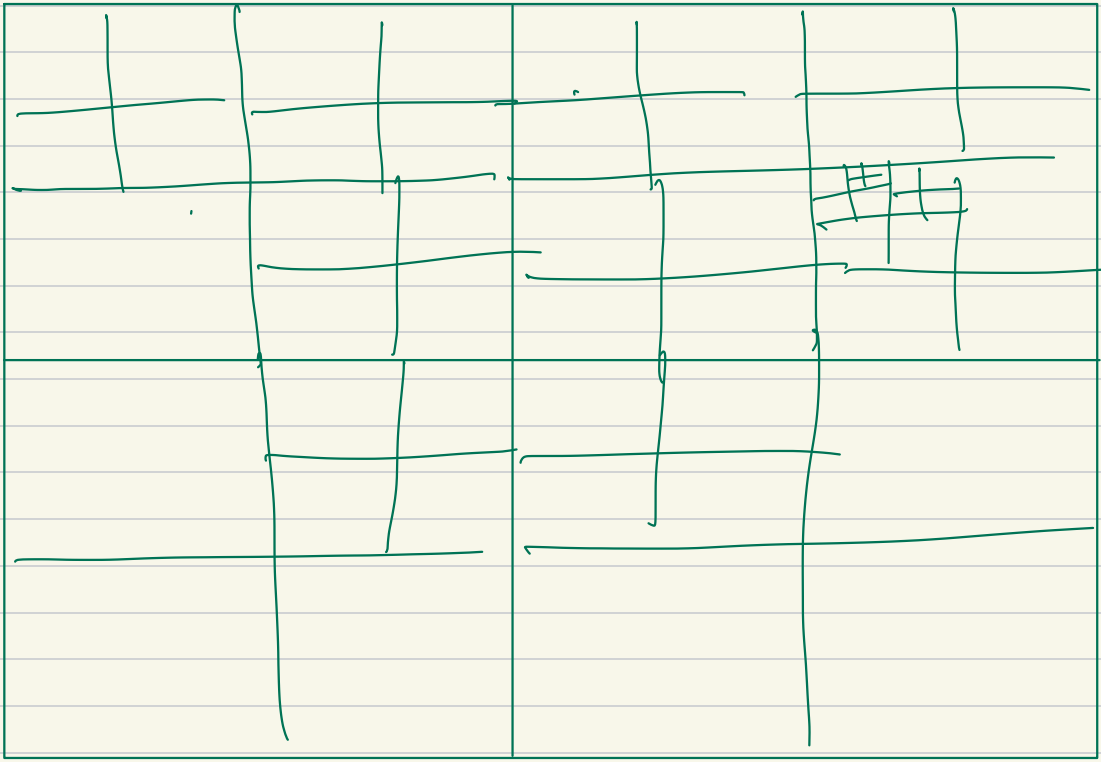
[+ neighbouring grid cells]

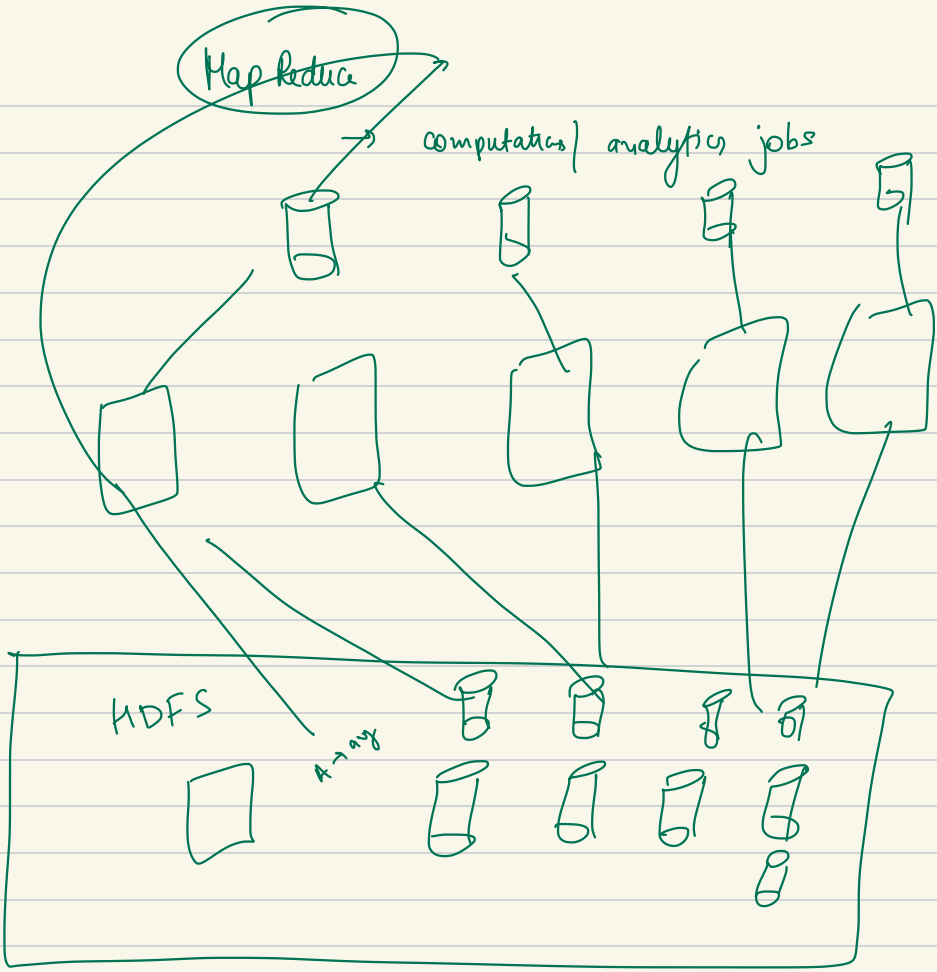
PROBLEM: . . . .

Even though a grid cell solution is a solution in the right direction, but it is NOT without problems

Unequal density of places of interest throughout the world

Flexible Grid Solution




Map Reduce
computations/ analytics jobs
HDFS
Array