Cassandra

Saturday, 20 March 2021 12:59 PM

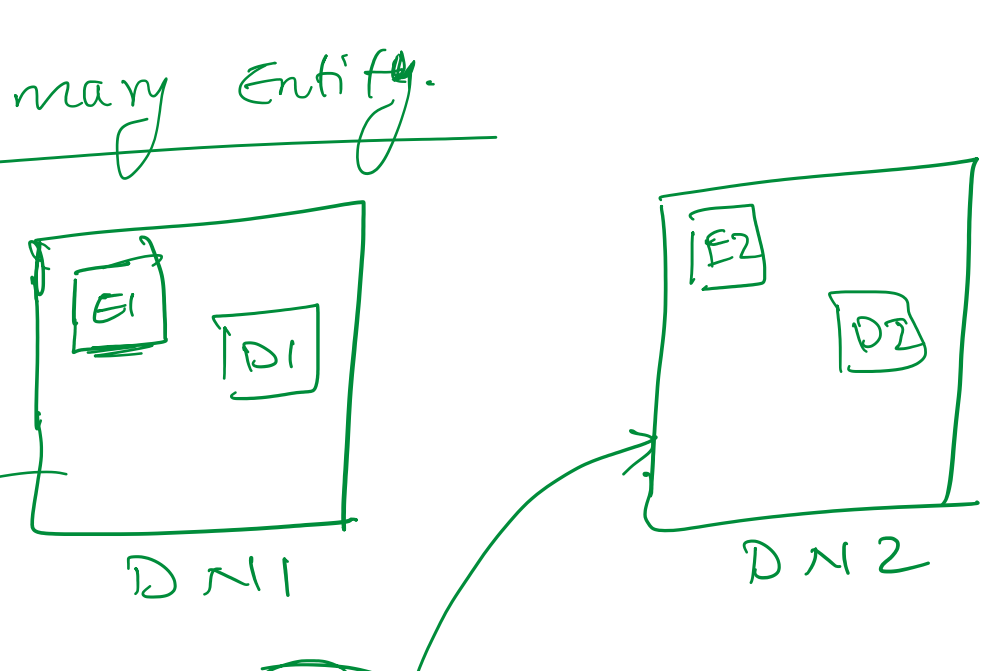
- search type ahead
- why shard is data
- types of NoSQL
  - KV
  - Document
  - Column Based
  - Graph

○ Challanges associated with tradinal RDBMS
- * Shard automatically
- * Query routing should be automatic
- * Choice between C & A.

Emp		
id	name	dept-id

Dept		
id	name	desc

Emp: primary Entity.



→ id, Name, D2

Normalizatⁿ & sharding doesn't go hand in hand

① Denormalize. ( )

Emp: { id: —
name: —
dept: { id:
name:
desc:
}
}

updates: O(n) time
insert: O(1)
read: O(1)
delete: O(1)

Gmail

userId: {

}

Trade-offs
1. Require more space
2. update anomalies
3. otherwise, works well
4. there won't be any joins.

② Storage

i) Tables: RDBMS

ii) K.V: key-value (Redis/Aerospike, mem cache)

key: {value}

updating (Mohit 4)

→ user_id: { emails: [{ sender: mohit@scaler.com, recipient: vaibav, body: 2 }] }

O(1) < O(n) (KV) stores?

(user_id: int)

iii) Column Family Store: (Cassandra, HBase, Druid)

GMAIL

Row Key: (database)	small table 1	small table 2	small table 3
user_id:	column family: contacts (K:V, K:V)	CF2: Email (K-V, K-V)	CF3: Threads

"Yogesh": Contacts (name: jaydeep, email: jaydeep@sc..., table) | Email (Thread_id: 01, # conversati: 10, sorted set) | threads (thread_id: 0 { })

iv) Document store (Mongo DB)

{ — : — : — : }

file: _id:

{
"name": "yogesh"
"age": 20
organisatⁿ: { name: Google
#emp: 10,000
}
}

{ "item_id": —
"item_val": —
}

Doc1 Doc2

RDBMS	Document
Database:	Database
→ Table:	Collection
Row:	Document

Node + React + MongoDB

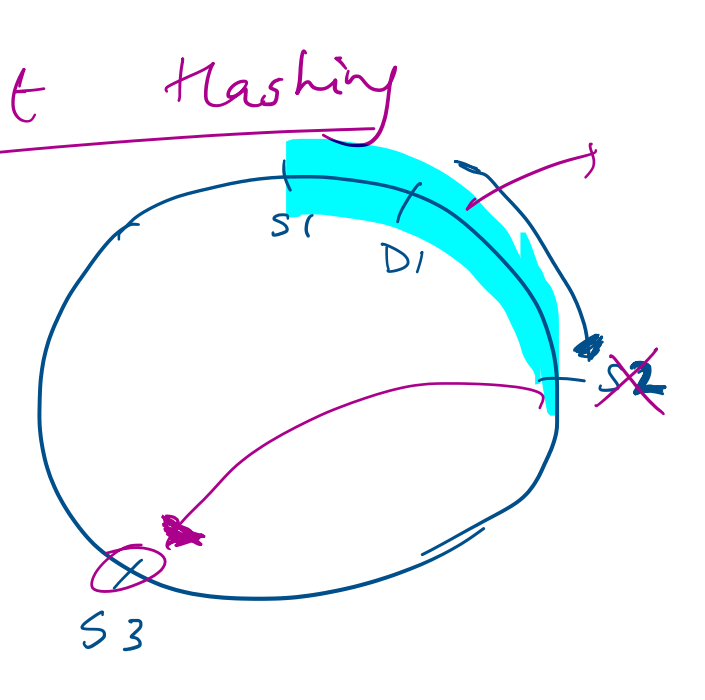
json

KV > Column family > Document

1. GMail: user
   Accounts: accountid
   Bank: account
   Amazon: product/items

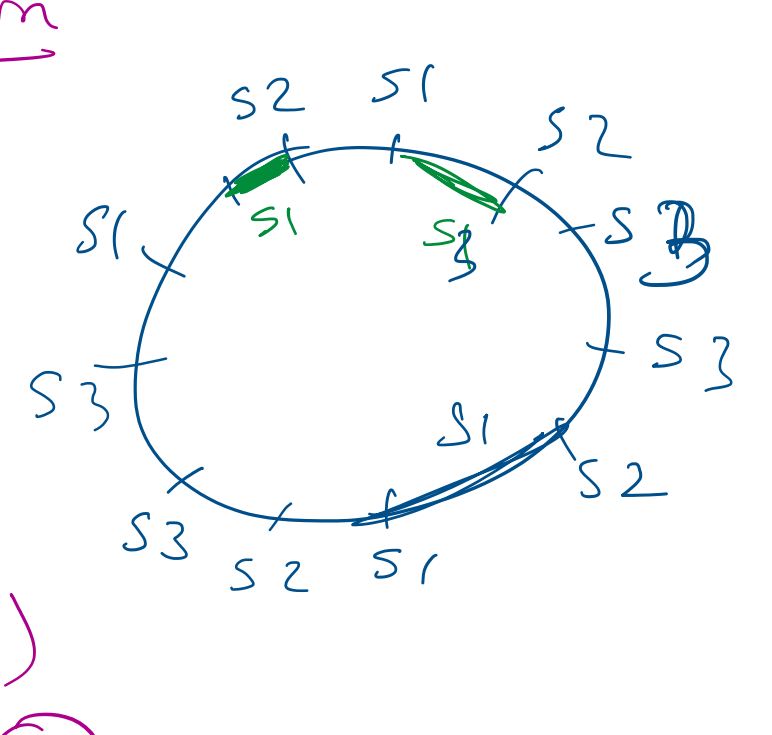
2. Sharding

Consistent Hashing



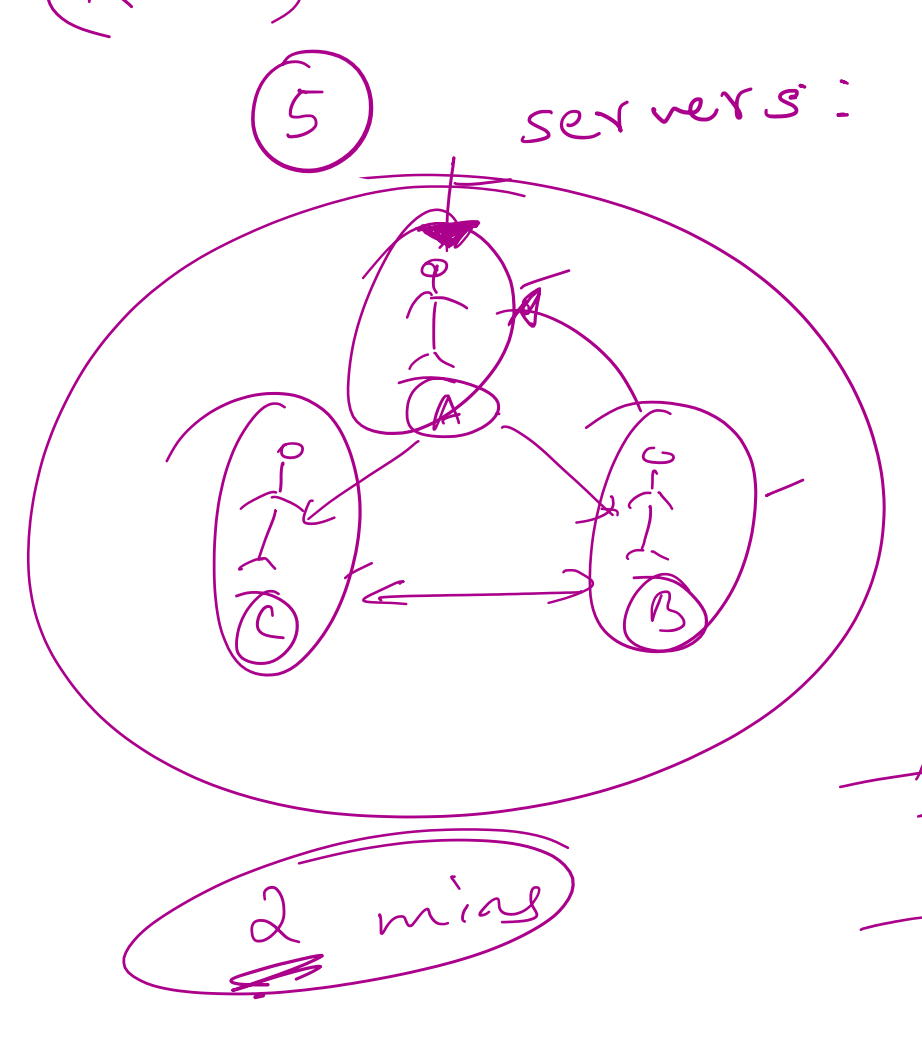
S1: ×
~~S2~~
S3: ✓
1/3 → S3
S1: 33%
S3: 66% ↑
50
50:50?

3. Replication



(R+W)

5 servers:



2 mins

③ RF

'xyz': (remind me to board a flight at 10 am tommorrow)

to take call: 1min

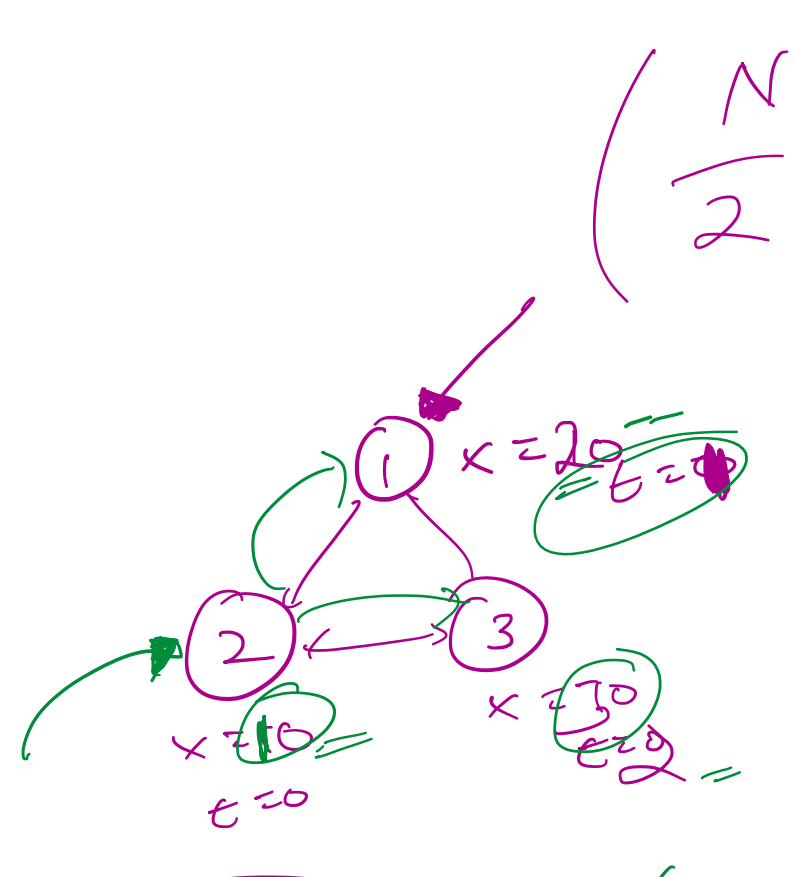
to inform others about a call: 30 sec

N

(N/2)

24 hrs

$\frac{1 \times 60 \times 24 \times 3}{2} =$



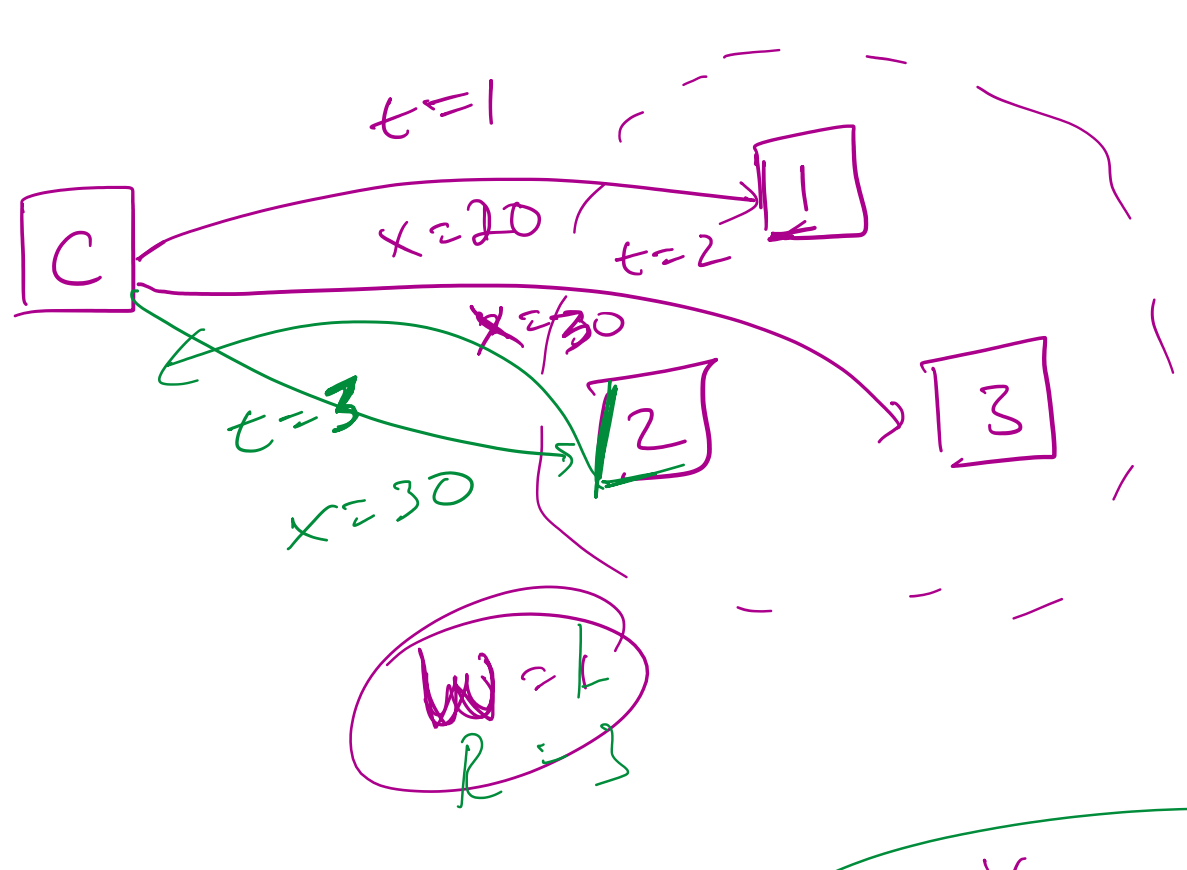
t=3 x=30 t=2 → 30

1 Node:

write: write to 1 Node

read: read from all Nodes

my system is still strongly consistent.



W=1 R=3

writes: (W=1, R=RF) 1ms

	N1	N2	N3
t=1μs	x=10	x=10	x=10
t=2μs	x=20		
t=2μs		x=20 t=1	x=30
t=3μs		10 t=0	x=30 t=2

writes are fast
reads are slow

⇒ write intensive systems = (log systems, rating)

read intensive system (FB, insta, social)

W: RF
R: 1

	A	B	C
t=0	x=10	x=10	x=10
t=1	x=20	x=20	x=10
t=2			

W=2
R=1
R=2

(W+R) > RF (RF−W)+1

N

$\frac{n}{2} + \frac{n}{2}$ Quorum

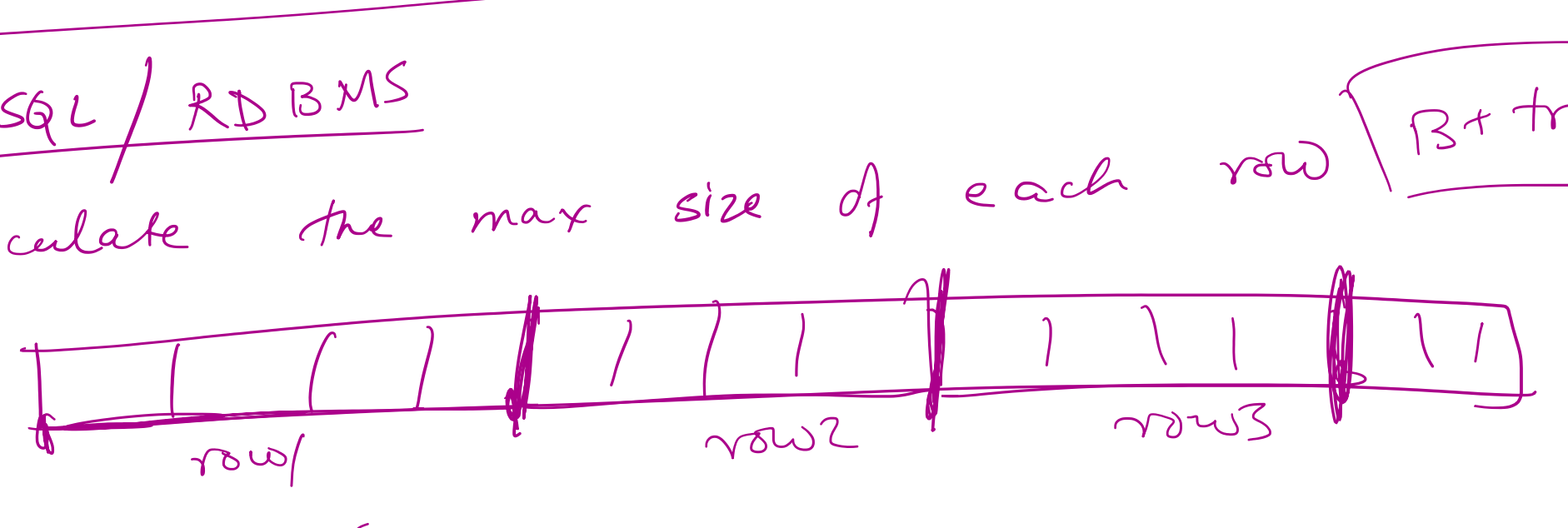
default config in cassandra: W=1 → A( ), R=1 → C( )

W=2 (data loss) ⇒

MySQL / RDBMS B+ trees

○ calculate the max size of each row

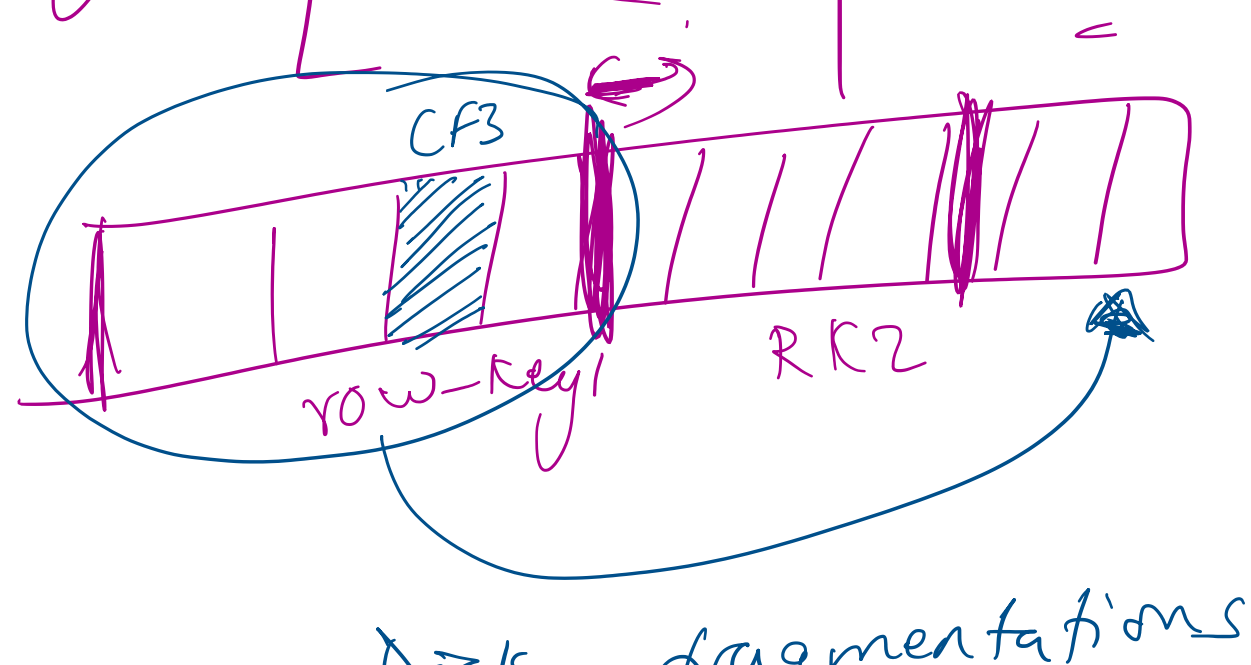
○



Column family

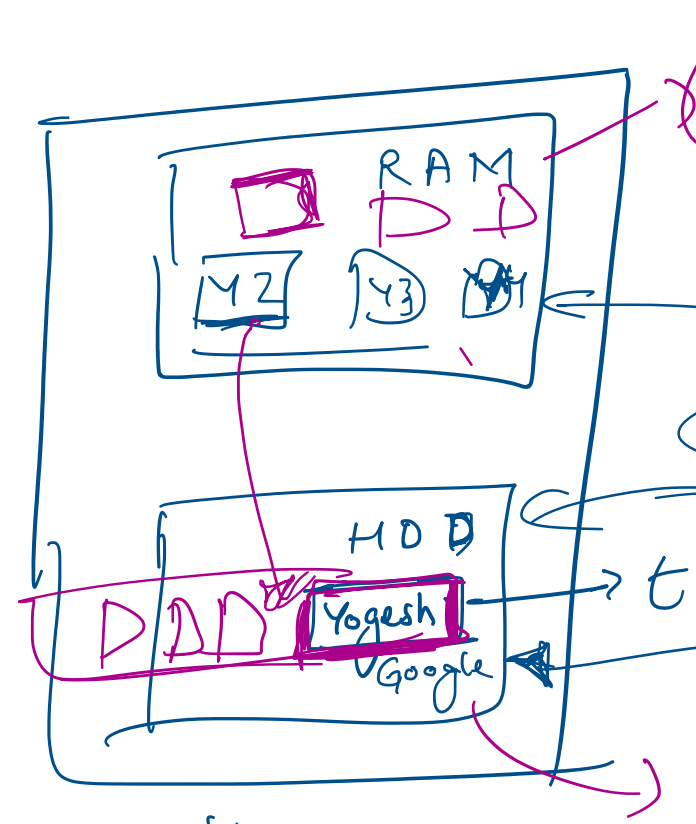
○ fixed size is not possible

row-key: CF1 | CF2 | CF3 | CF4



1. Add CF anomaly.
2. Remove CF anomaly.

Disk fragmentations: OS



HDD: 100 μs ( )

RAM: 10 μs. (random traversals easy)

GMail:
Yogesh: Email t=1 (Yogesh: facebook)

Data is always immutable

Reads: (RAM + HDD) Merge

↓ show it to the user.

Yogesh: { email1: { }
email2: { }
}

Yogesh: { email
E2
E3
E100
}

V6

Yogesh: { email 50
email 78
}

{ a=1, b=1, c=1 } V0 + { d=1, c=2 } V1 → Merge step ⇒ { a=1, b=1, c=2, d=1 } final

Merits: faster writes.

Demerits: ① Reads are slow. → COMPACTION (I/O)
② What if a server dies?
→ data flushed to disk periodically.

updates? → ○ Immutable
○ Sorted set (log n)

1%

flipkart