COMPUTATIONS / STORAGE ON NOSQL

HW ← ① Storage in NoS…
② Comp. across sha… or machines.

① Read is fast
② write is fast
→ 500 GB

iphone

Key → value

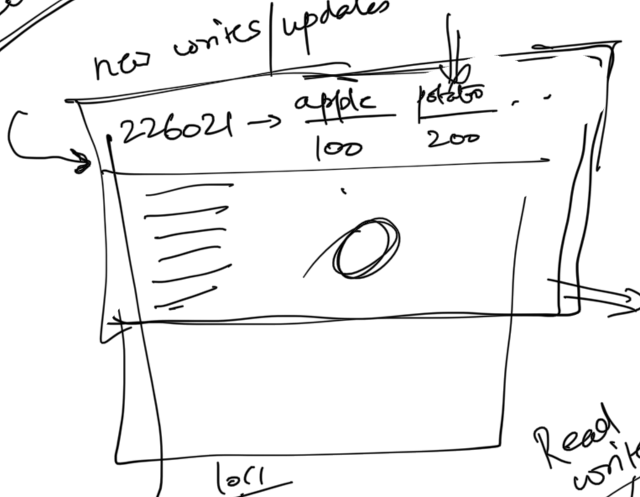
~~key~~ row → column family

~~document~~

Update | Immutable

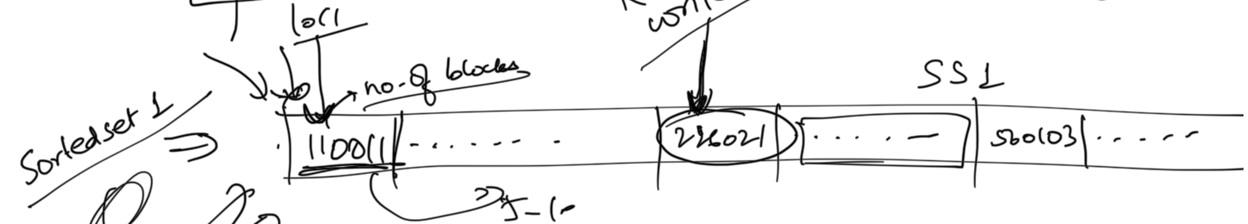
226021 ~~apple~~ apple: {"iphone 12", "iphone 13"}

apple: "iphone 14"



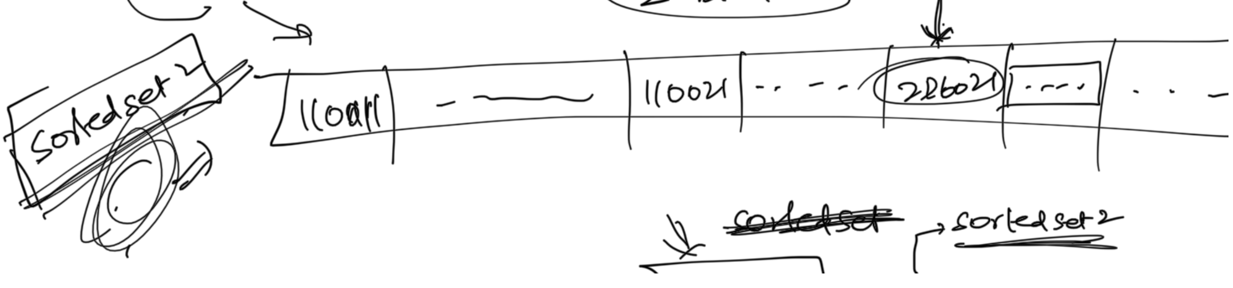
560103 ⇒ apple potato toma…

Lookup…
sorted set…
↳ 1011…
key → …

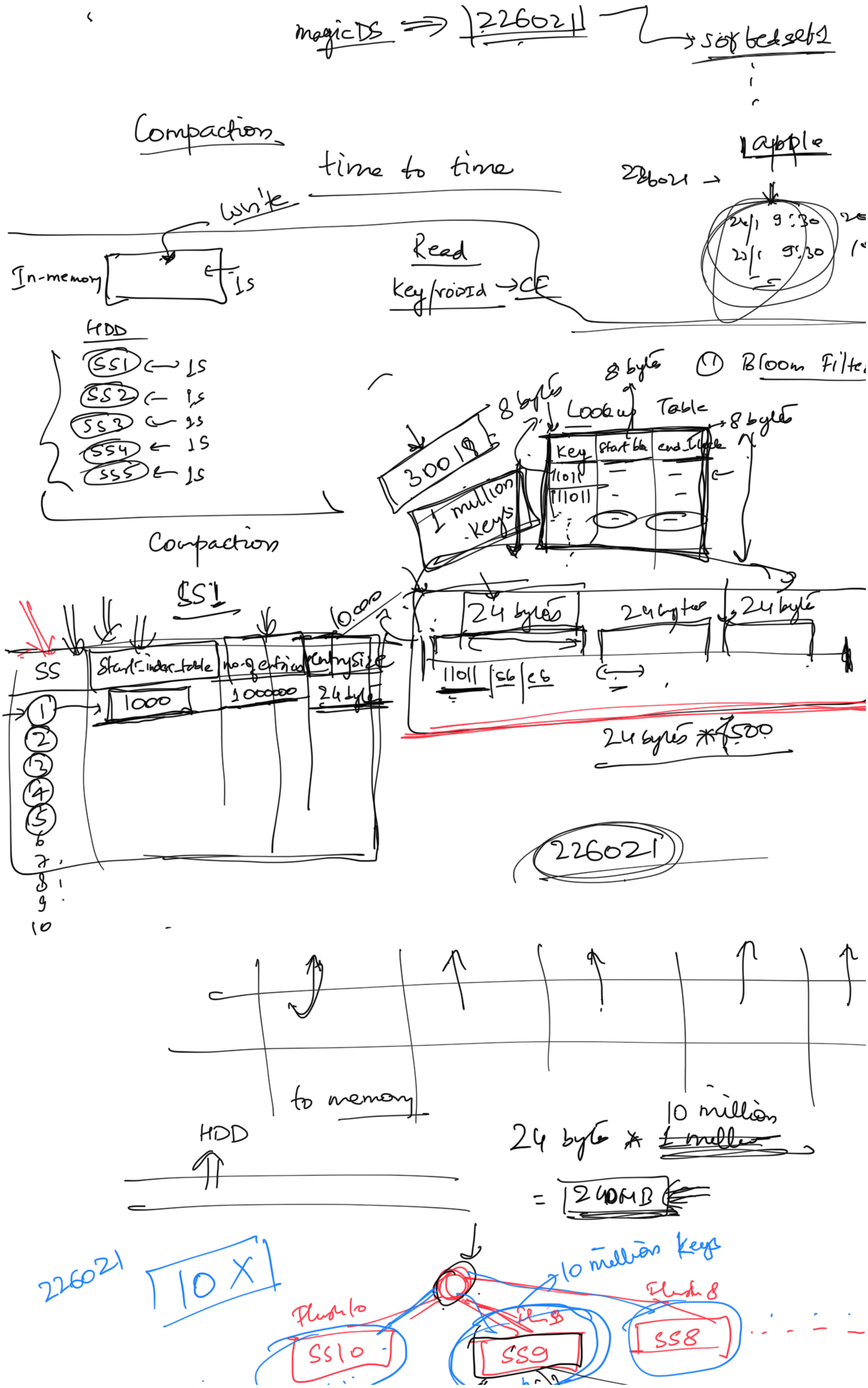


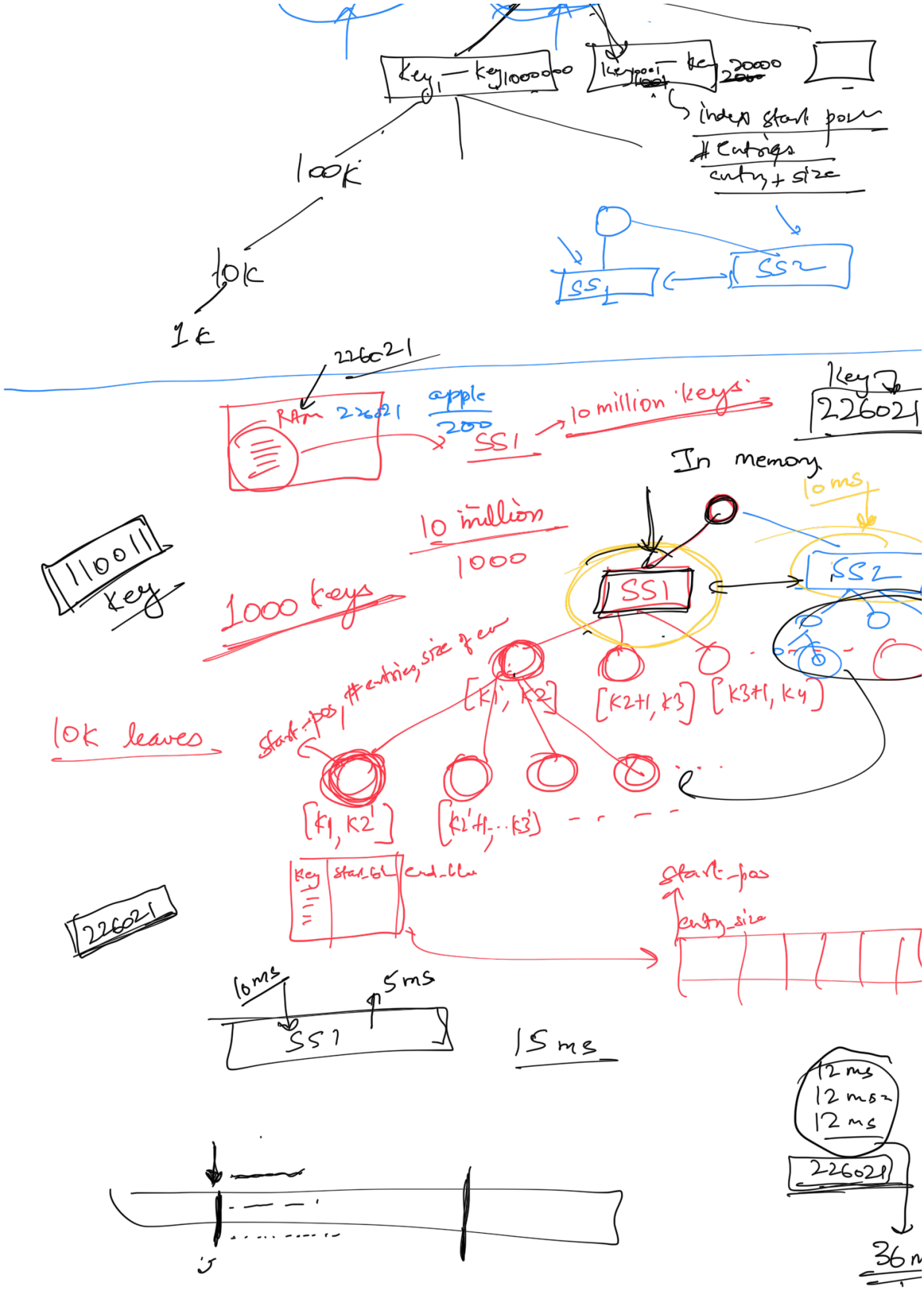
64 KB blocks.

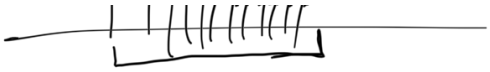
226021 → 120 KB
2 blocks.

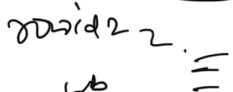


~~sortedset~~ → sortedset 2


magicDS
226021
sortedset1
apple
Compaction
time to time
write
Read
In-memory
1s
key/rowId
HDD
SS1
SS2
SS3
SS4
SS5
8 bytes
Bloom Filter
Lookup Table
key
start
end
300 18
1 million keys
Compaction
SS1
SS
start_index_table
no_of_entries
entrysize
1000
1000000
24 byte
24 bytes
24 bytes * 500
226021
to memory
HDD
24 byte * 10 million
= 240MB
226021
10 X
10 million keys
Flush 10
Flush 8
SS10
SS9
SS8

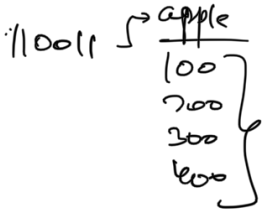

Key1 — key100000
Key100001 — key200000
index start pos
entries
entry + size
100K
10K
1K
SS1
SS2
226021
RAM 226021
apple
200
SS1
10 million keys
Key
226021
In memory
10 ms
10 million
1000
1000 keys
11001
key
SS1
SS2
10K leaves
start_pos, # entries, size per
[K1, K2]
[K2+1, K3]
[K3+1, K4]
[K1, K2']
[K2'+1, K3']
Key
start_pos
end_pos
226021
start_pos
entry_size
10ms
5ms
SS1
15 ms
12 ms
12 ms
12 ms
226021
36 m
Compaction :
rowid
4 bytes
SS1
SS2
rowid → merged data
rowid2 → merged



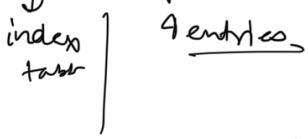


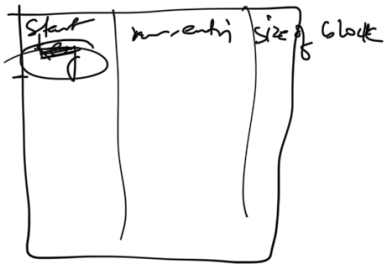
index table

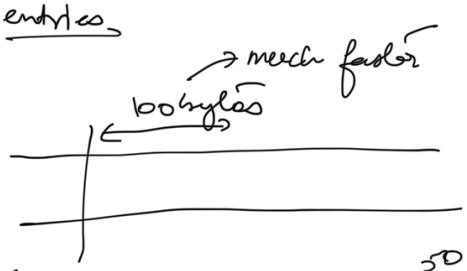
Case 1: no. of sorted sets * x

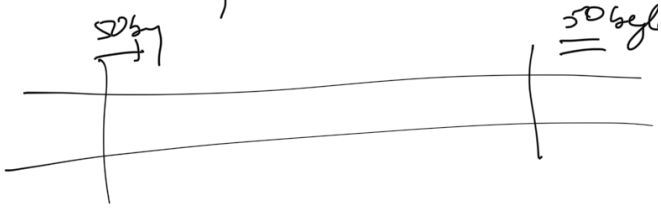


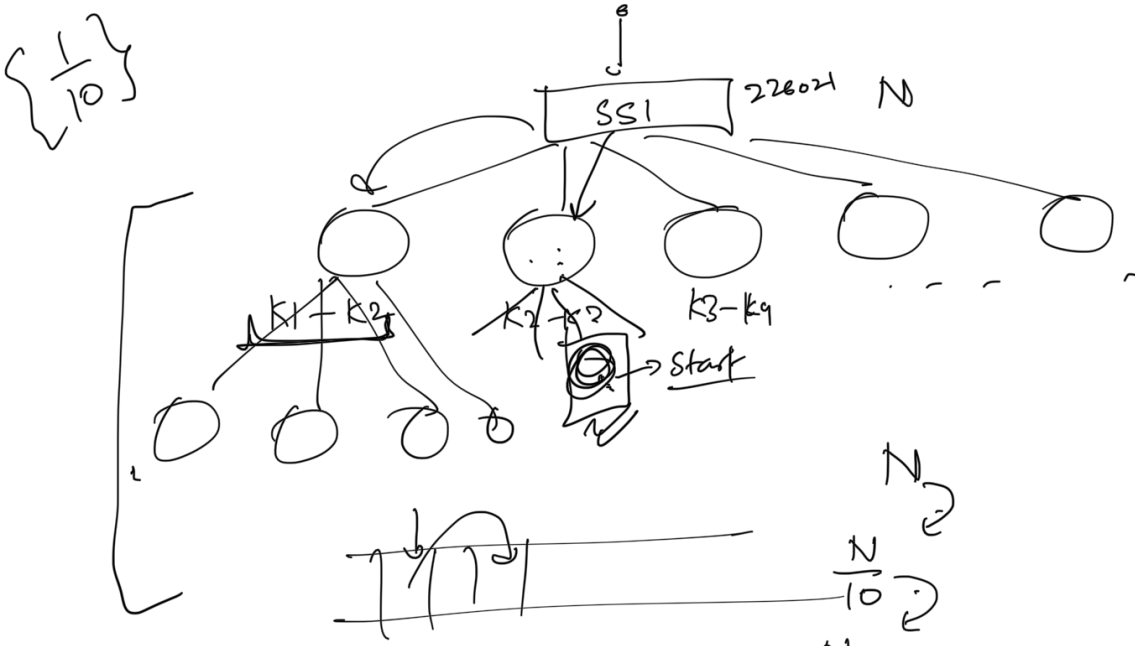
Case 2: 1 sorted set → HDD once.











$\left\{\frac{1}{10}\right\}$

$\frac{N}{10^h} = 1000$

$10^h = \frac{N}{1000}$

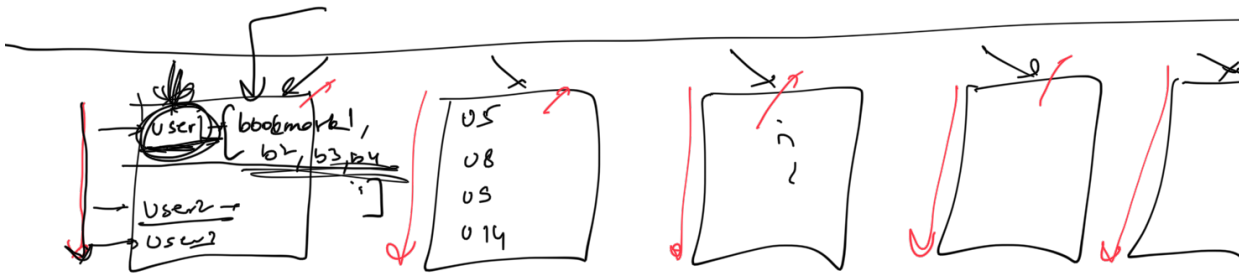
$h = \left(\log_{10} \frac{N}{1000}\right)$

$\frac{N}{1000} \rightarrow \leq 1000$

$h = 0 \ldots 10^7 \ldots 4$

$\log_{10} 1000 = 10$

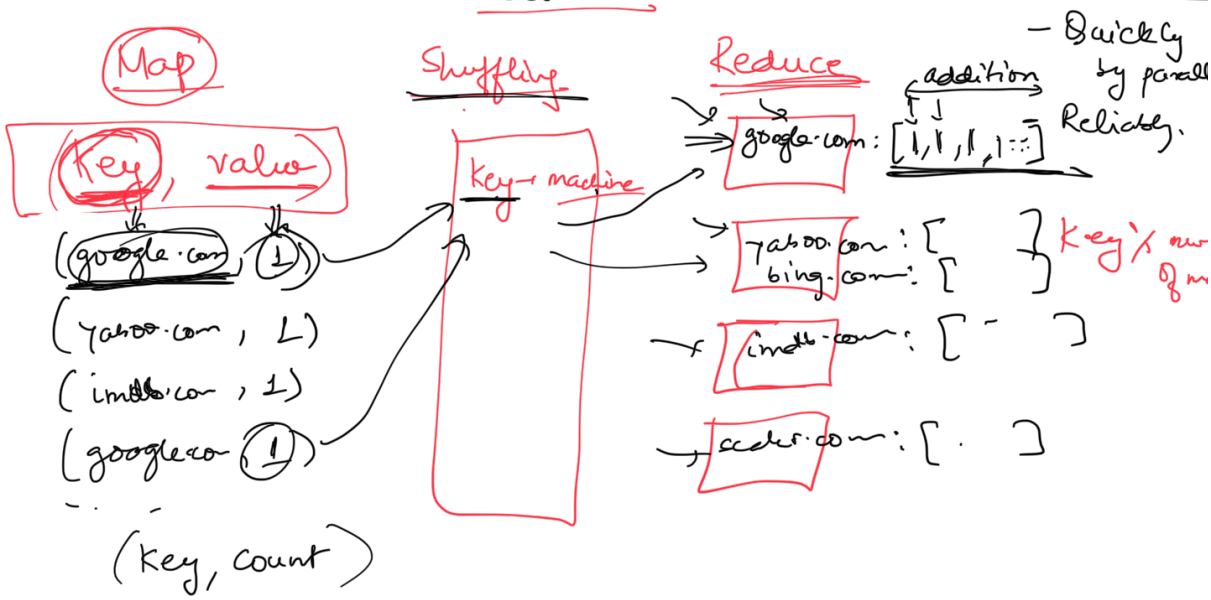
(4)



bookmarks → count

Map Reduce

- Quickly by parall
Reliably.



(key, count)

map(string line) { // 0 or more (key, value)

key → bookmark

value → count

}

⇒ key, value

reduce(string key, list<int> value)

{

MR

MR scheduler

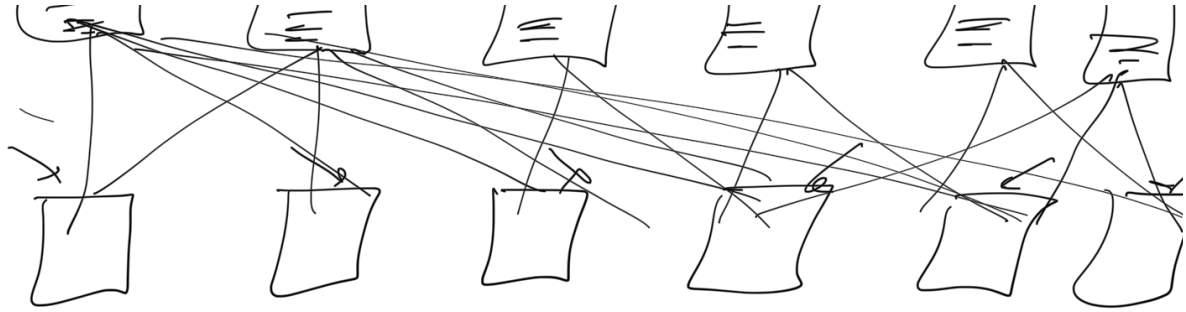
(map job

(shuffling + sorting

reduce

}

(key, count)



Word Count

Map

deer car bear → [(deer, 1), (car, 1), (bear, 1)]

dog car deer → [(dog, 1), (car, 1), (deer, 1)]

bear dog bear → [(bear, 1), (dog, 1), (bear, 1)]

Reduce

bear: [1, 1, 1] → (bear, 3)

deer: [1, 1] → (deer, 2)

cat: [1, 1], dog: [1, 1] → (cat, 2), (dog, 2)

Map (key, value)

a: b c d

f: c

b: a c e

e: b

c: a b f

d: f g

(a, b) → [b c d]

(a, c) → (b c d)

(a, d) → (b c d)

(a, b) → [a c e]

(b, c) → [a c e]

(b, e) → (a c e)

Reduce

(a, b) → [c

(a, b) → [(b c d), (a c e)]

(a, b) → [c

(a, b) → [(...

(...

Key → Value (Tombstone)

Read

map(

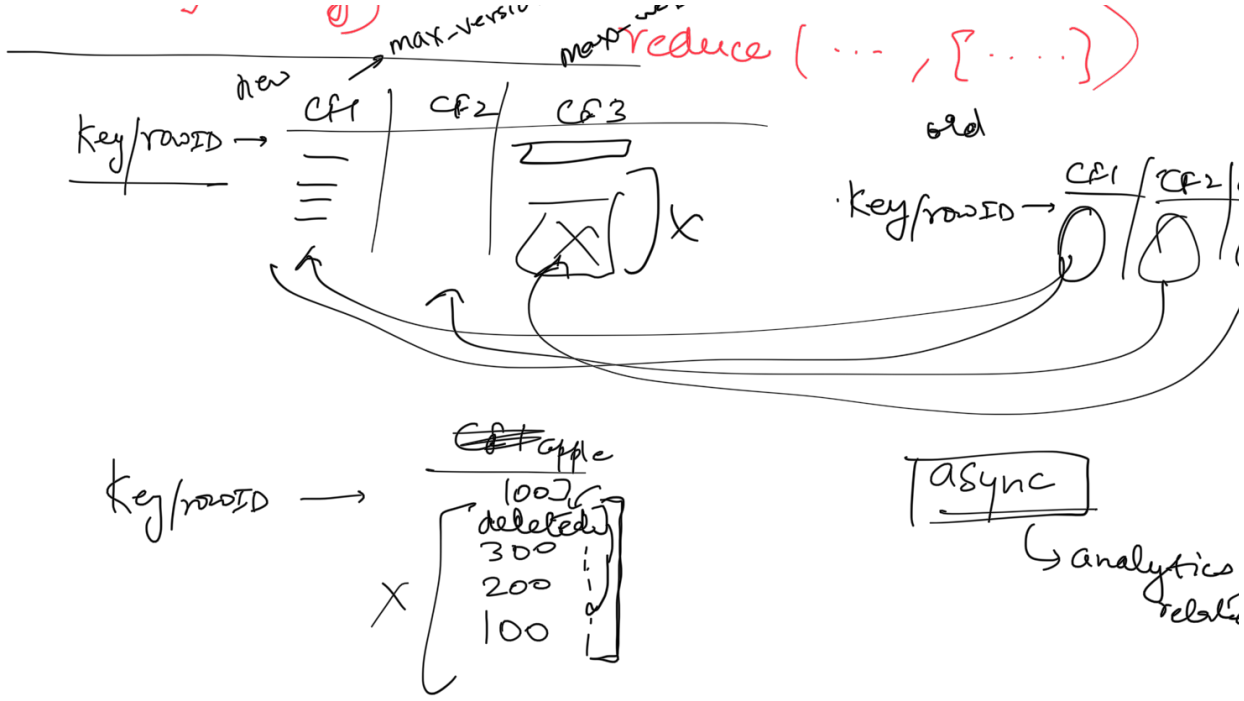
get(key)

HIVE

↳ HiveQL

↓

MR jo



Key/rowID → ~~CF1~~ apple

100
deleted
300
200
100

async

↳ analytics