Hash functions

- cryptographic hash functions

$$h : \Omega \longrightarrow \{0, \ldots, n-1\} = [n]$$

→ $i \in [n]$: $\Pr[h(x) = i] = \frac{1}{n}$

$\{h(x_1), \ldots, h(x_k)\}$ — random subset of $[n]$ with k elements

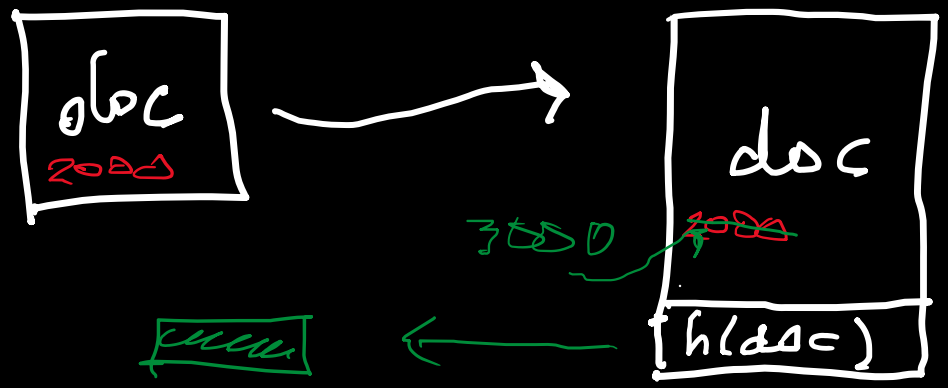
→ "salty"

If we know $y = h(x)$

then it is very diff to find x^*

s.t $y = h(x^*)$

Example. Suppose doc is a document (txt)



h(doc /2000 → 3000)
It wish
h(doc)

Signature
10103537G173...
33559...
5599AFB553

Typical examp.: MD5
SHA-X

X = 128, 256, 512, 1024

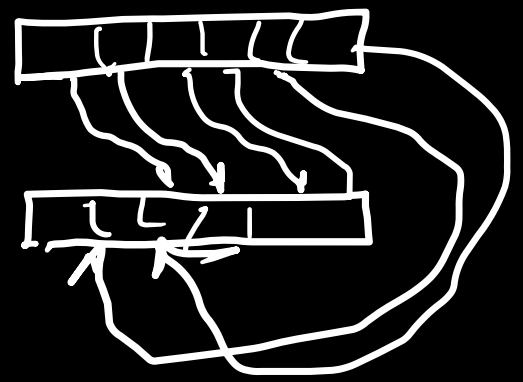
- non-crypt. functions:

k-independent for ~~reg~~ large k

Popular: Murmur hash function

{ Mu ← multiplication
  r ← rotate

(wiki)



ref. space.

Scala

```
import scala.util.hashing.{MurmurHash3 => MH3}
...
val seed = 1953553   // unsigned
myHash(data, mySeed) {
  val n = MH3.stringHash(data, mySeed)
  n
}
```

this must be given // indep. of the program

independence

```
val seeds = [3535, 3510, 1033]   // random uint values
```

myHashDelex(data, i) { myHash(data, seeds[i]) }

Suppose $h : \{0,1\}^* \longrightarrow [n]$ is good function.

$$\underbrace{h(x_1), h(x_2) \ldots h(x_k)}_{\approx \text{ random numbers from } \{0,\ldots,n-1\} \text{ with distribution.}} \quad \begin{matrix} x_i \neq x_j \\ \forall i \neq j \end{matrix}$$

$\approx$ random numbers from $\{0,\ldots,n-1\}$ with distribution.

$\rightarrow X_1, X_2, X_3, \ldots$ round. numb for $[n]$ indep.

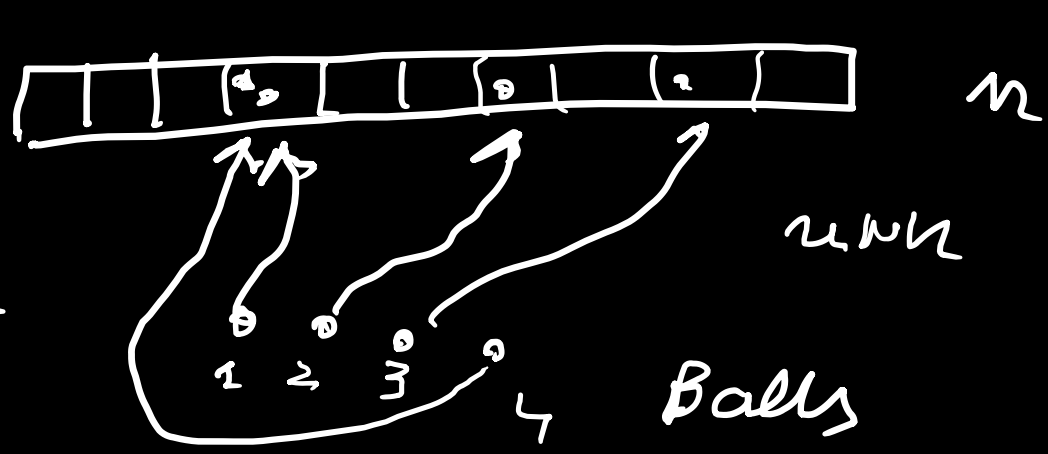
- $B = \min\{k : X_k \in \{X_1, \dots, X_{k-1}\}\}$

$$1 \le B \le n+1$$

$$E[B] = \sqrt{\frac{\pi \cdot n}{2}} + \frac{3}{2} + O\left(\frac{1}{\sqrt{n}}\right)$$

$$\simeq \sqrt{n}$$

Birthday Paradox.



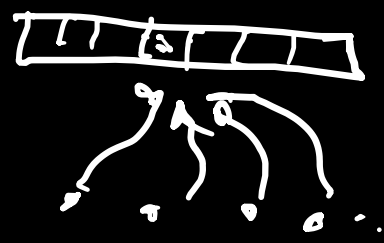
$$\sqrt{\frac{\pi}{2}} \simeq \sqrt{\frac{3.14}{2}} \simeq \sqrt{1.6}$$

$$\simeq 1.3 \simeq 1$$

- Coupon Coll Problem

$$C = \min\{k : \{X_1, \dots, X_k\} = [n]\}$$

$$P[C = \infty] = 0$$



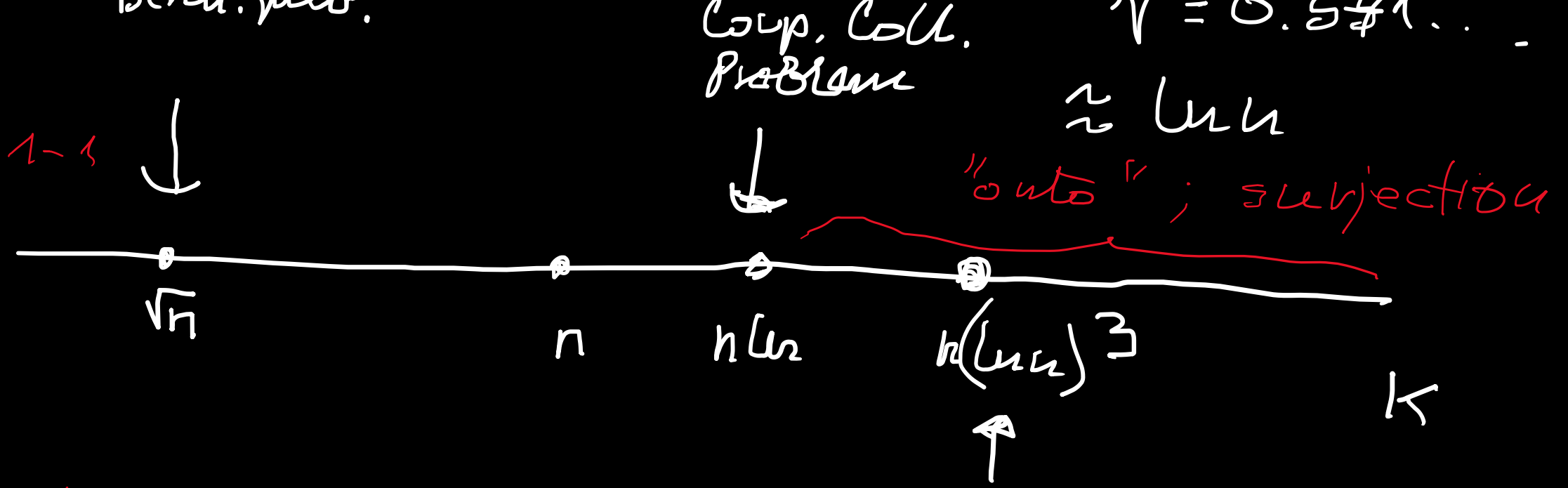
$$E[C] = n \cdot H_n \simeq n \ln n$$

, $H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$

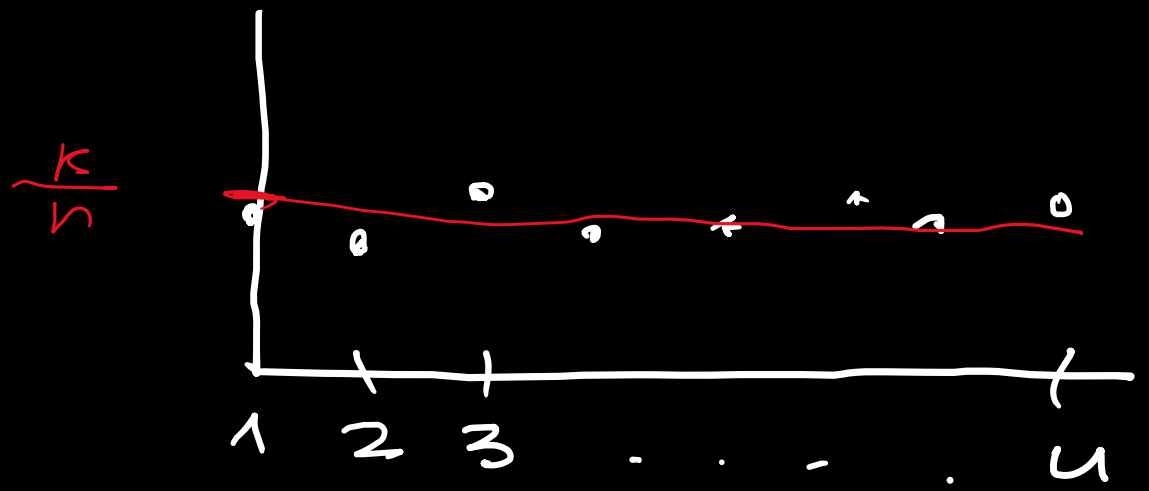
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right)$$

$\gamma = 0.5771\ldots$

$\simeq \ln n$



$X_1 \ldots X_k \quad k \ll \sqrt{n}$



uniformity of deployments

$k > n(\log n)^3$

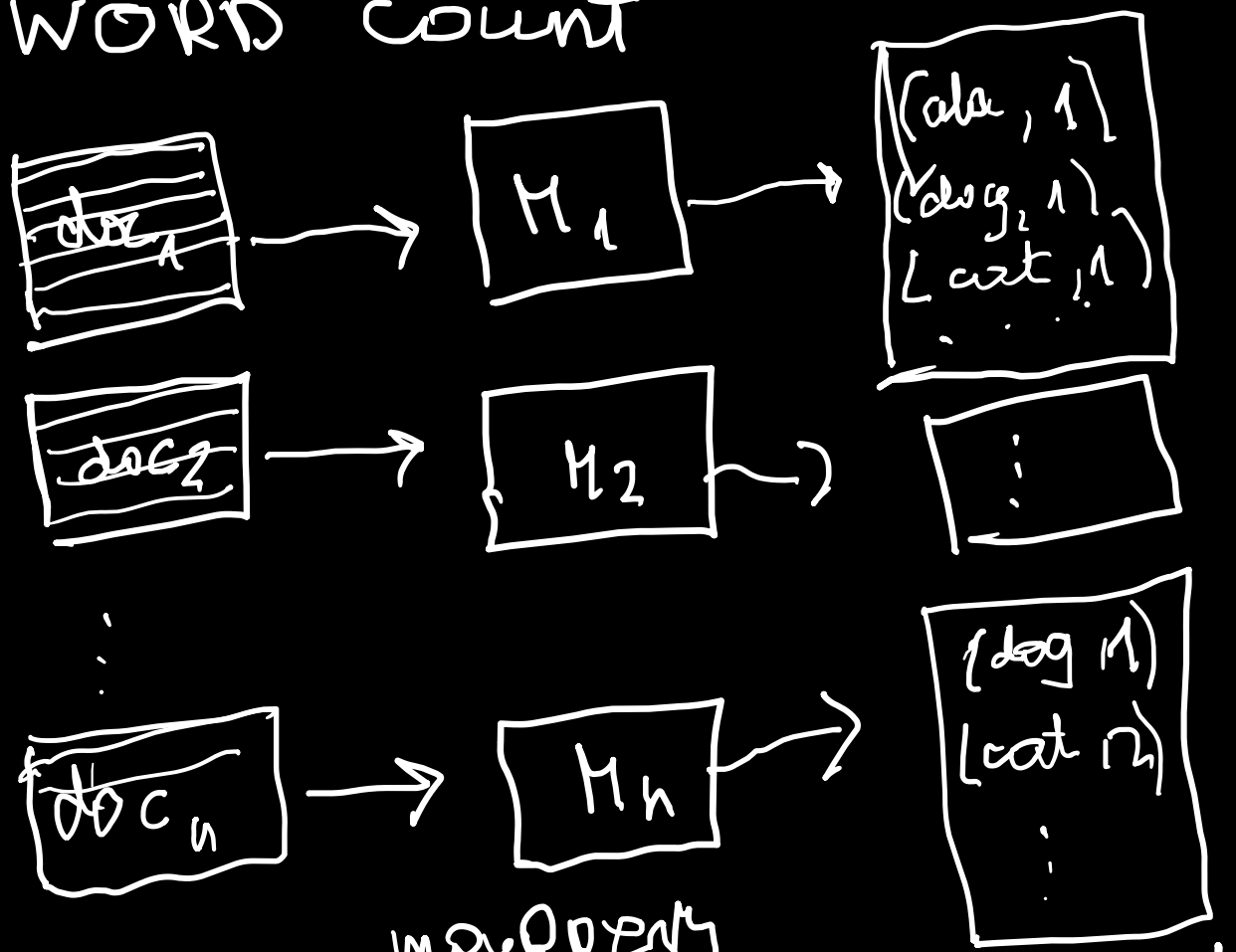
m_i = number of balls in ith urn

$m_1 + \ldots + m_n = k$

mean number of balls in bins : $\frac{k}{n}$

MAP - REDUCE MODEL OF COMPUT.

WORD COUNT



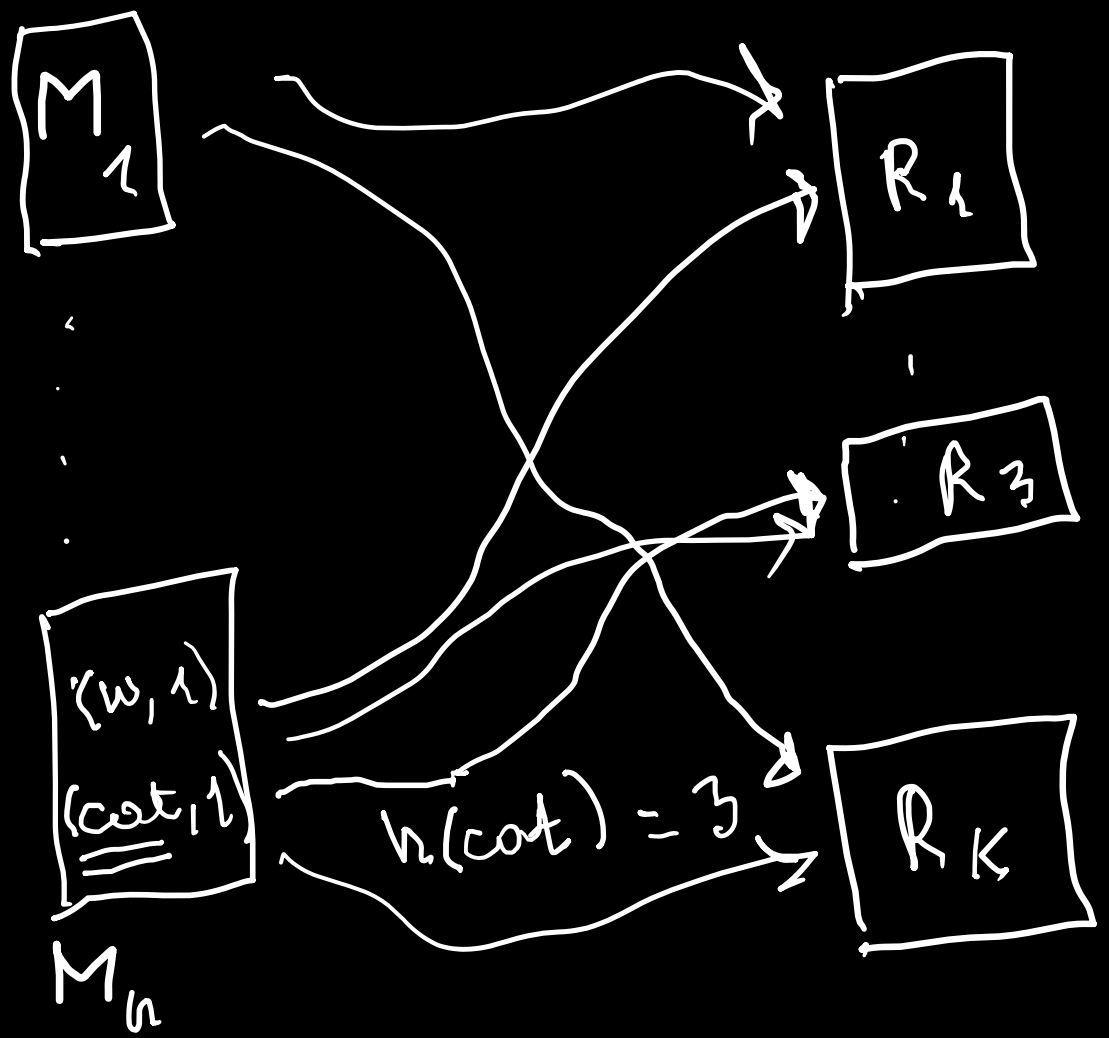
STEP 1

!!!

```
map (L) {
for each w ∈ L do {
  emit ((w, 1))
}
}
```

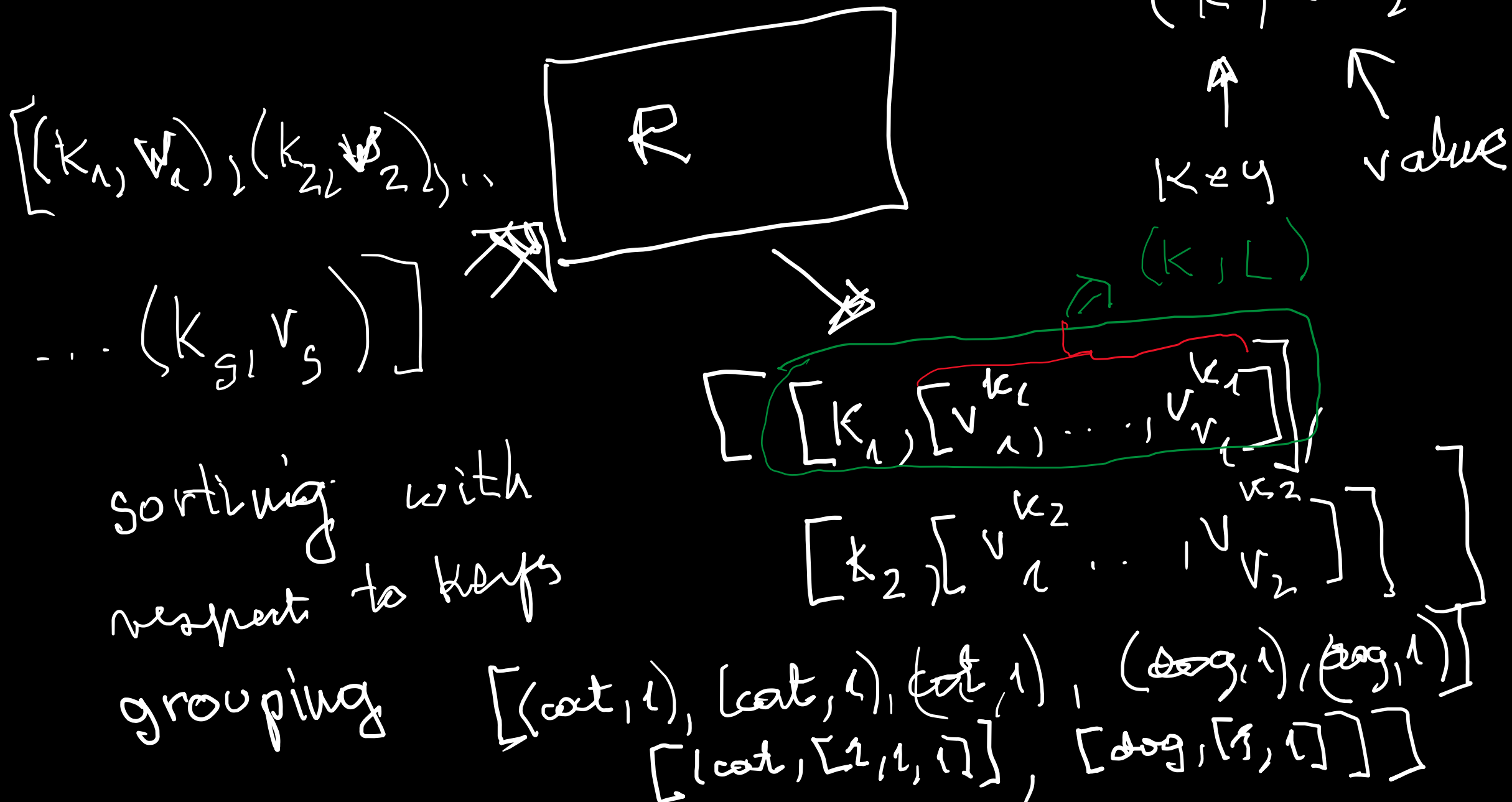
Hadoop
Apache Spark

STEP 2



$h : \Omega \rightarrow \{1, \ldots, k\}$

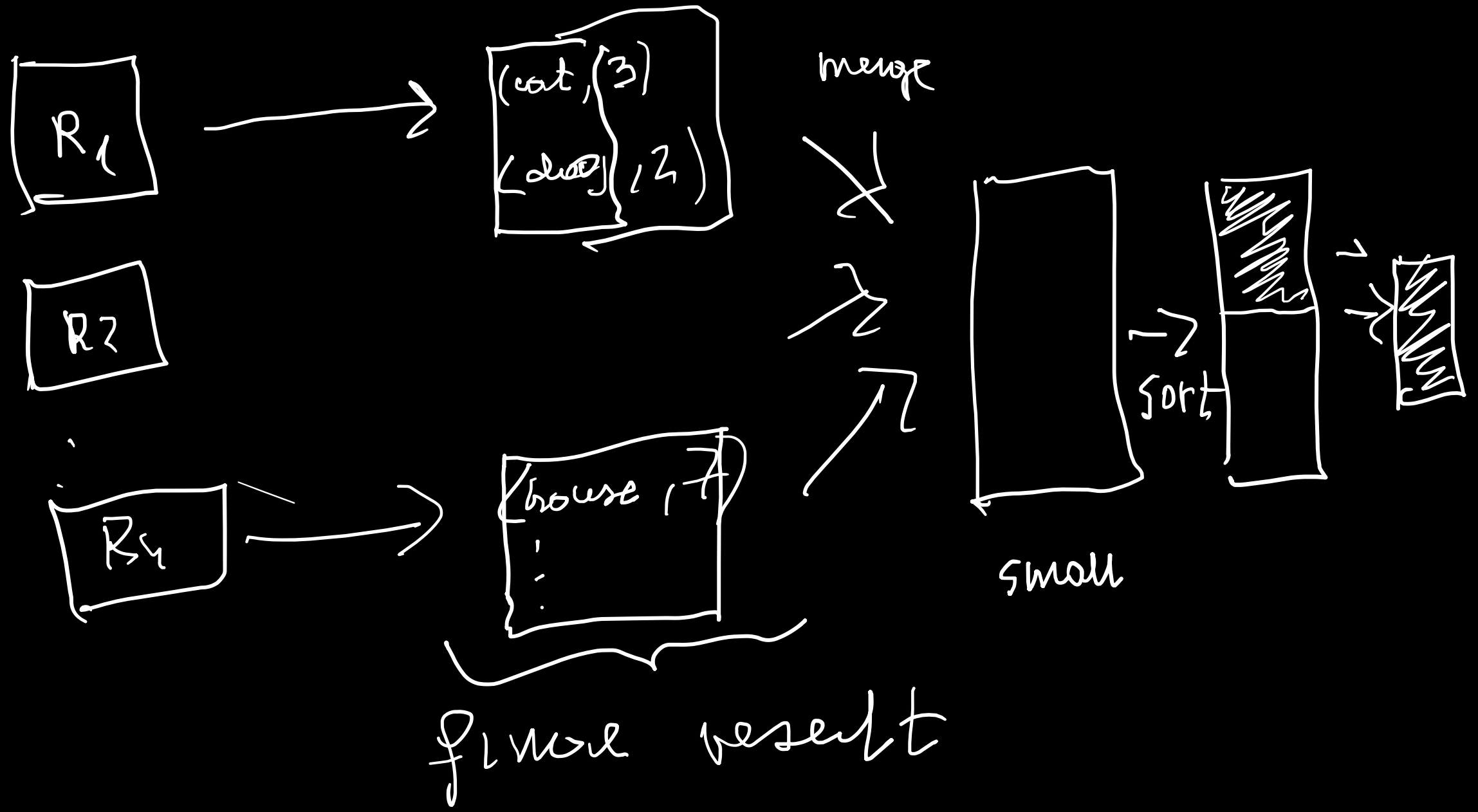
fixed hash function

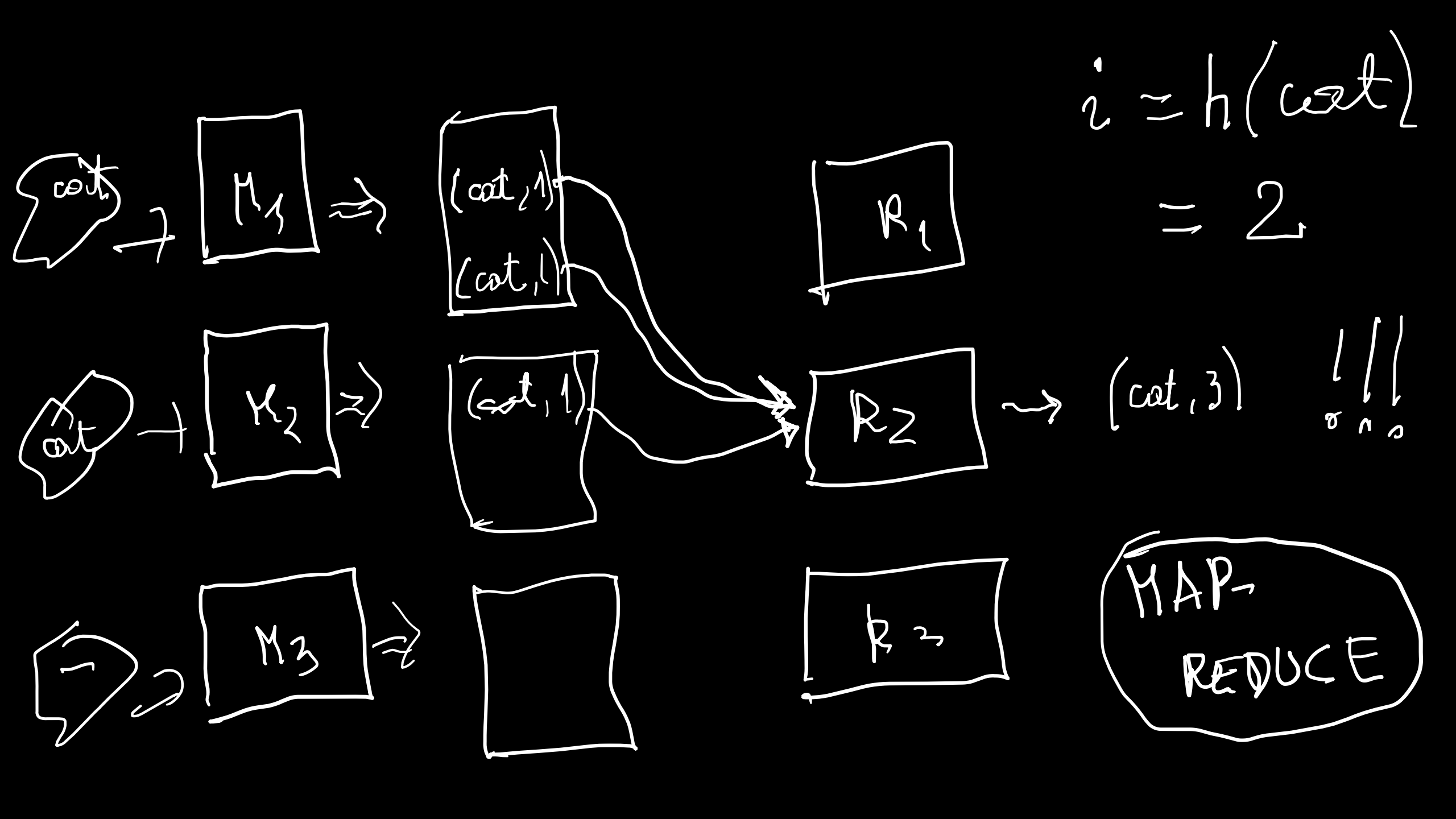

(k, v)
key
value
R
(k, L)
sorting with respect to keys
grouping

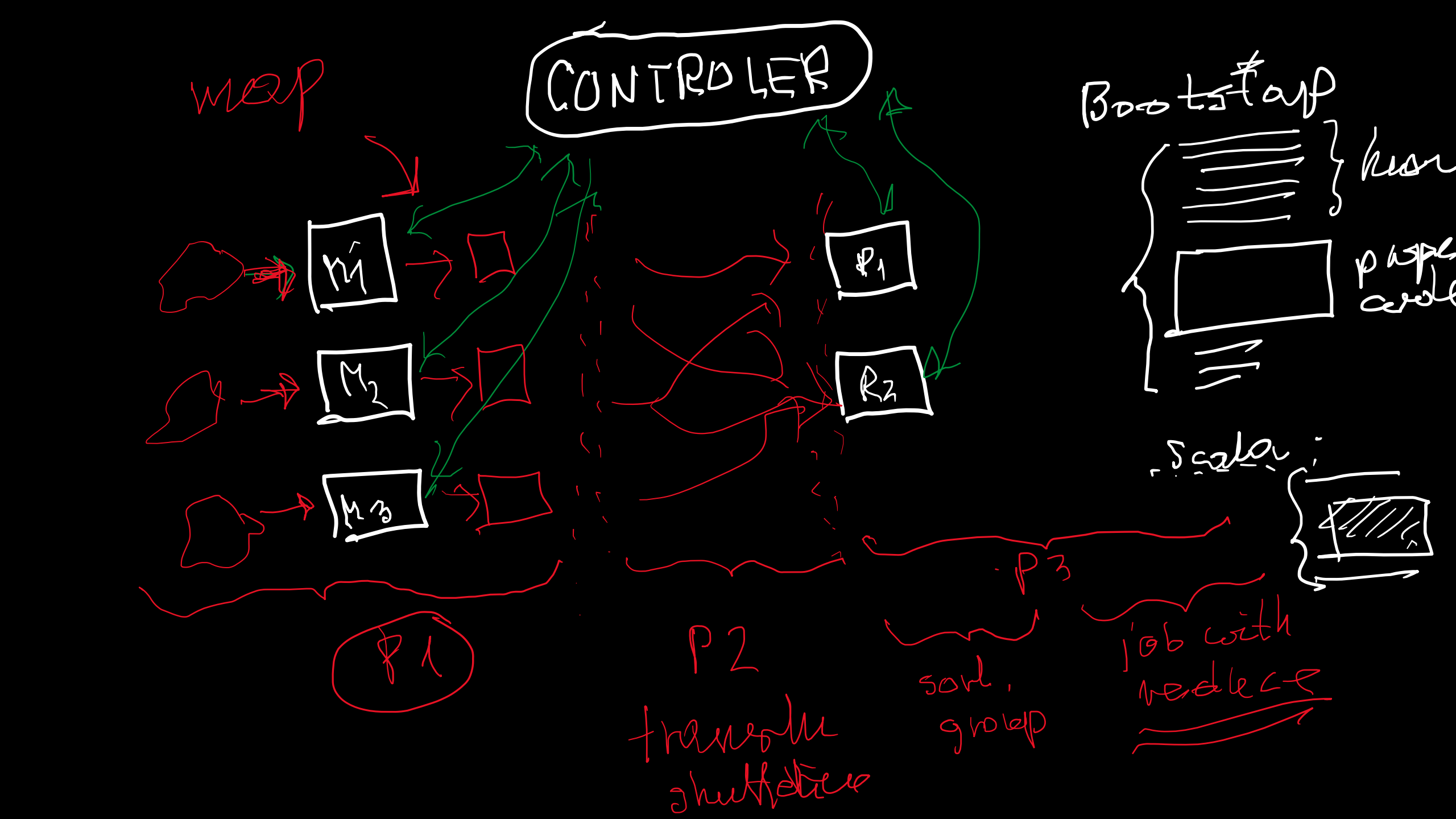
Programmers job with reducers

```
reduce(key, L){
    emit((key, length(L))
}
```

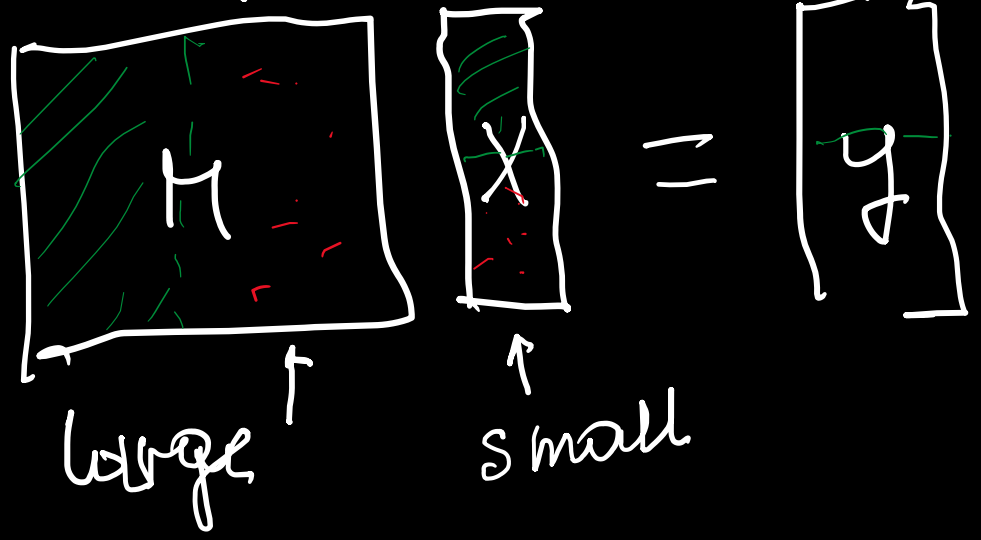
(cat, [1, 1, 1])


R1
R2
R4
(cat, 3)
(dog, 2)
(house, 7)
merge
sort
small
final result


i = h(cat)
= 2
cat
M1
(cat, 1)
(cat, 1)
R1
cat
M2
(cat, 1)
R2
(cat, 3)
!!!
M3
R3
MAP-
REDUCE


map
CONTROLER
M1
M2
M3
R1
R2
Bootstrap
Scala
P1
P2
P3
sort, group
job with reduce

Example



$$y_i = \sum_{j=1}^{n} m_{ij} \cdot x_j$$

what reducer gets

$$[\,[3, [m_{3,7} \cdot x_7, m_{3,9} \cdot x_9, \ldots]]\,]$$

input : (i, j, m_{ij})

```
map (i, j, m) {
    emit((i, m * x_j))
}
```

```
reduce (i, L) {
    emit (i, Σ L)
}
```