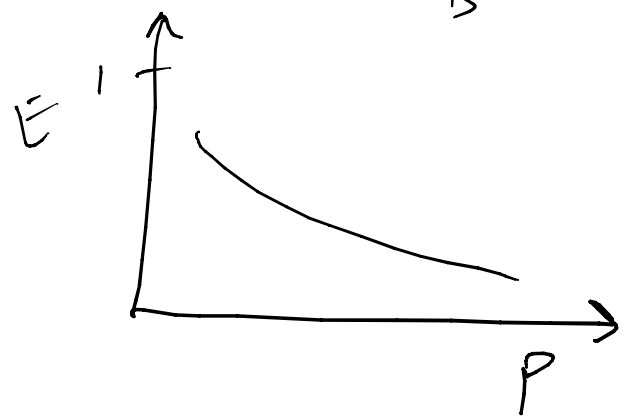
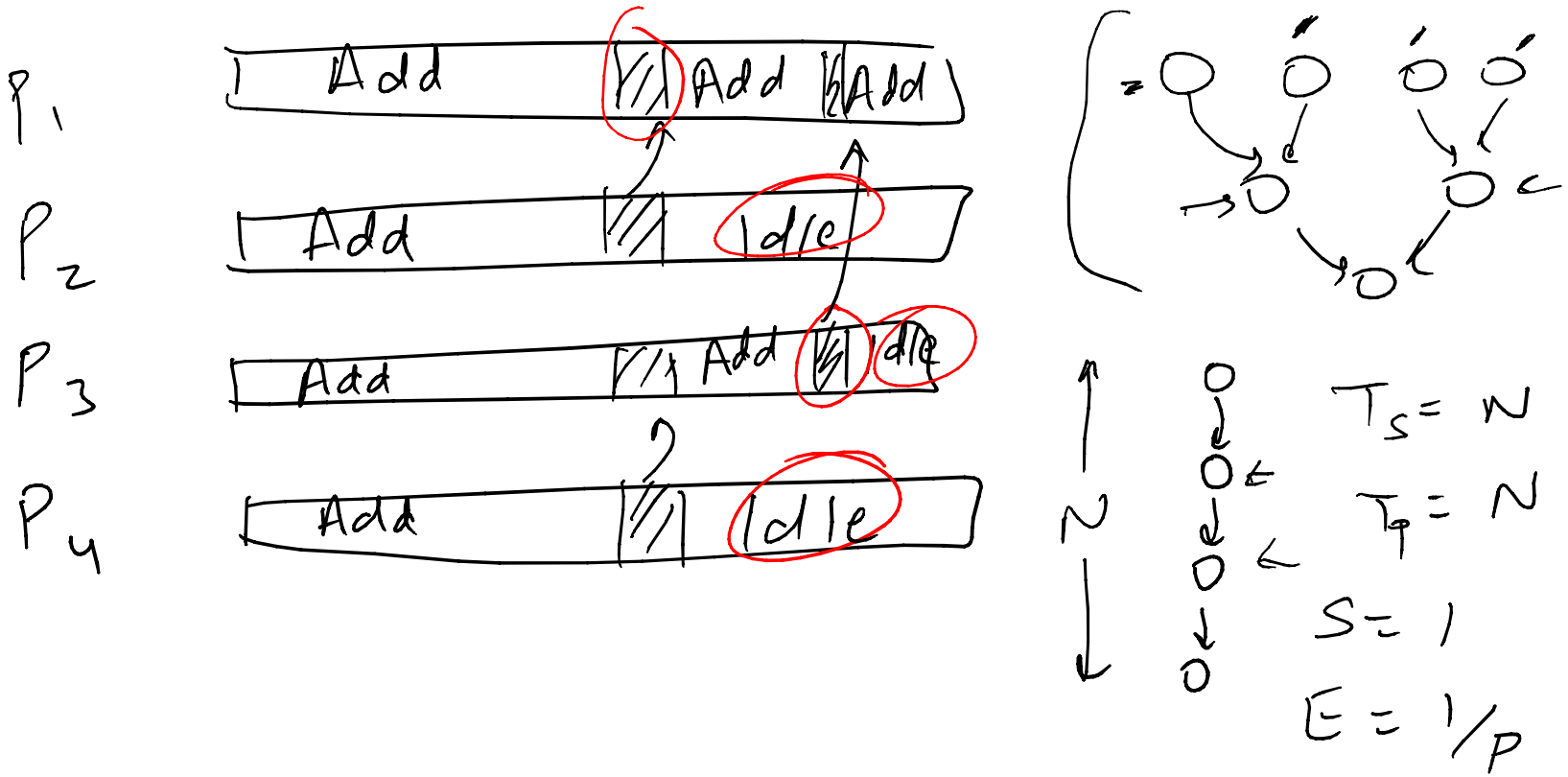
Note Title 06-Dec-22

$S = T_S / T_P$ $\frac{1.5}{3}$ 0.5 $E = S/P = \frac{T_S}{P T_P}$

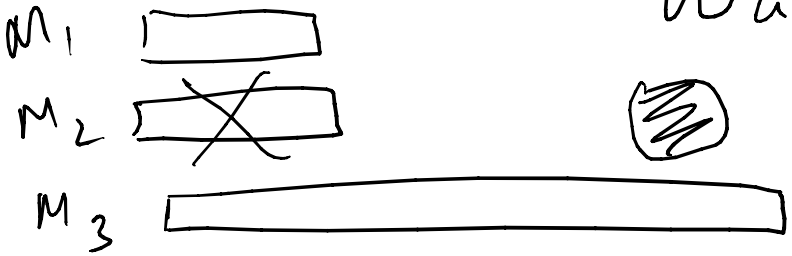


T_S P W =


P1
P2
P3
P4
Add
Add
Add
Idle
Add
Add
Idle
Idle
N
$T_s = N$
$T_p = N$
$S = 1$
$E = 1/p$

Implicit assumptions

- Uniform memory access
- No straggler



load balancing

faults

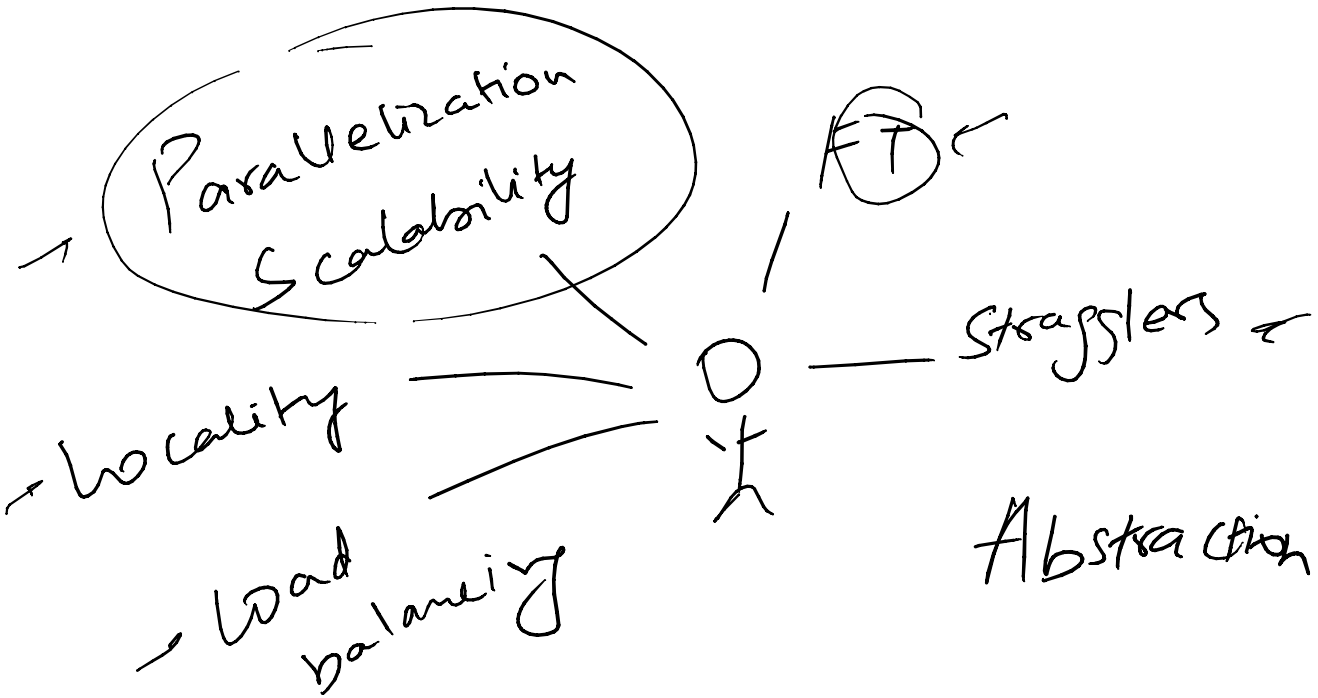
Locality matters

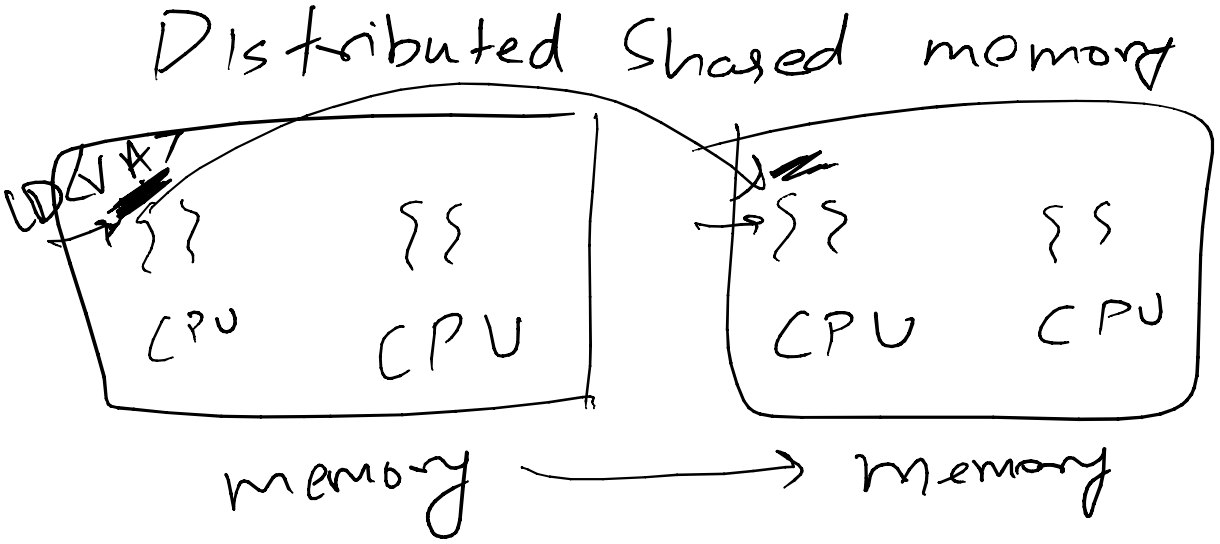
CPU → Cache ~1ns

DRAM ~100ns

D/C ~60μs

Cross D/C ~5-200ms


Parallelization
Scalability
FT
Stragglers
Locality
Load balancing
Abstraction



Page-based DSM

TLB

VA	PA

→ Page table

VA	PA	valid	Present

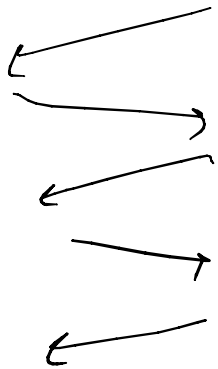
False sharing

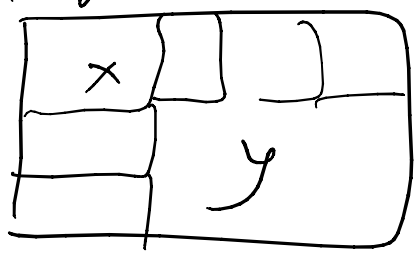
W1 for

$x \leftarrow x+1$

W2 for

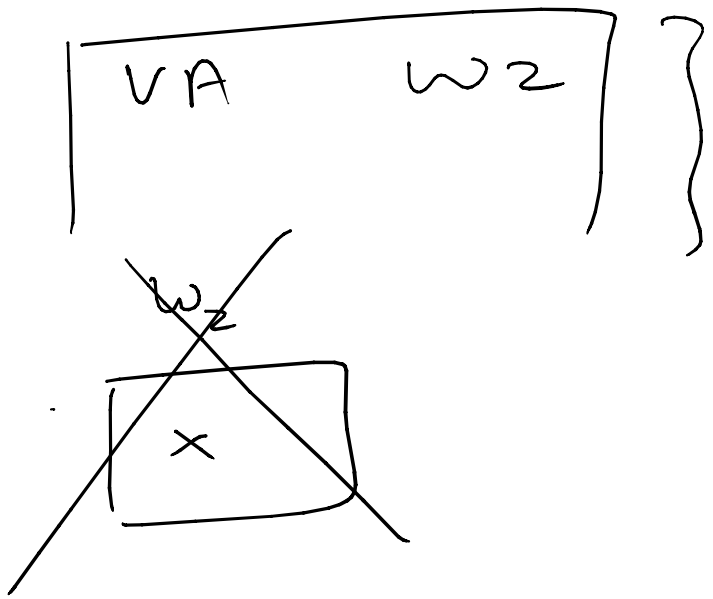
$y \leftarrow y+1$





Who has the page?

Worker ⟶ Central Server ← bottleneck



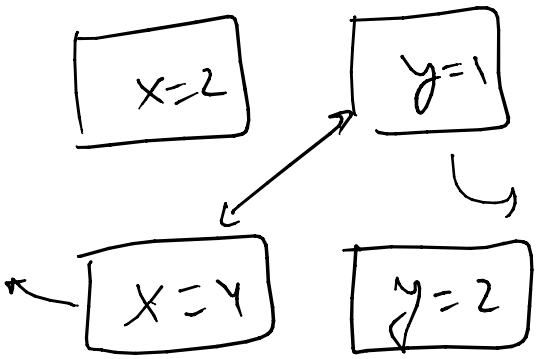
FT/ Straggler

for

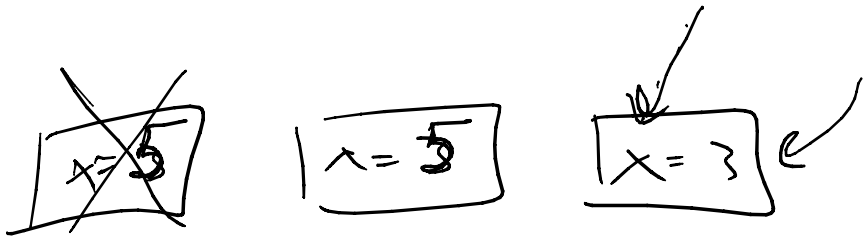
$x \leftarrow x+1$

Fault Tolerance

- Checkpoint
  - overhead
  - regularly
  - consistency
  - Rollback



- Replication



Batch —

Streaming —

ML training —

Graph —

grep word

map (file, file contents)
for if Emit (line)

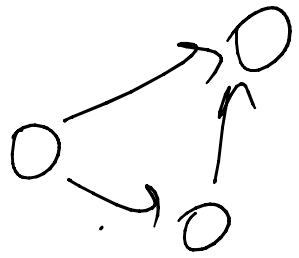
reduce:
append

Map Reduce

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Reverse web link graph



which doc point to me?

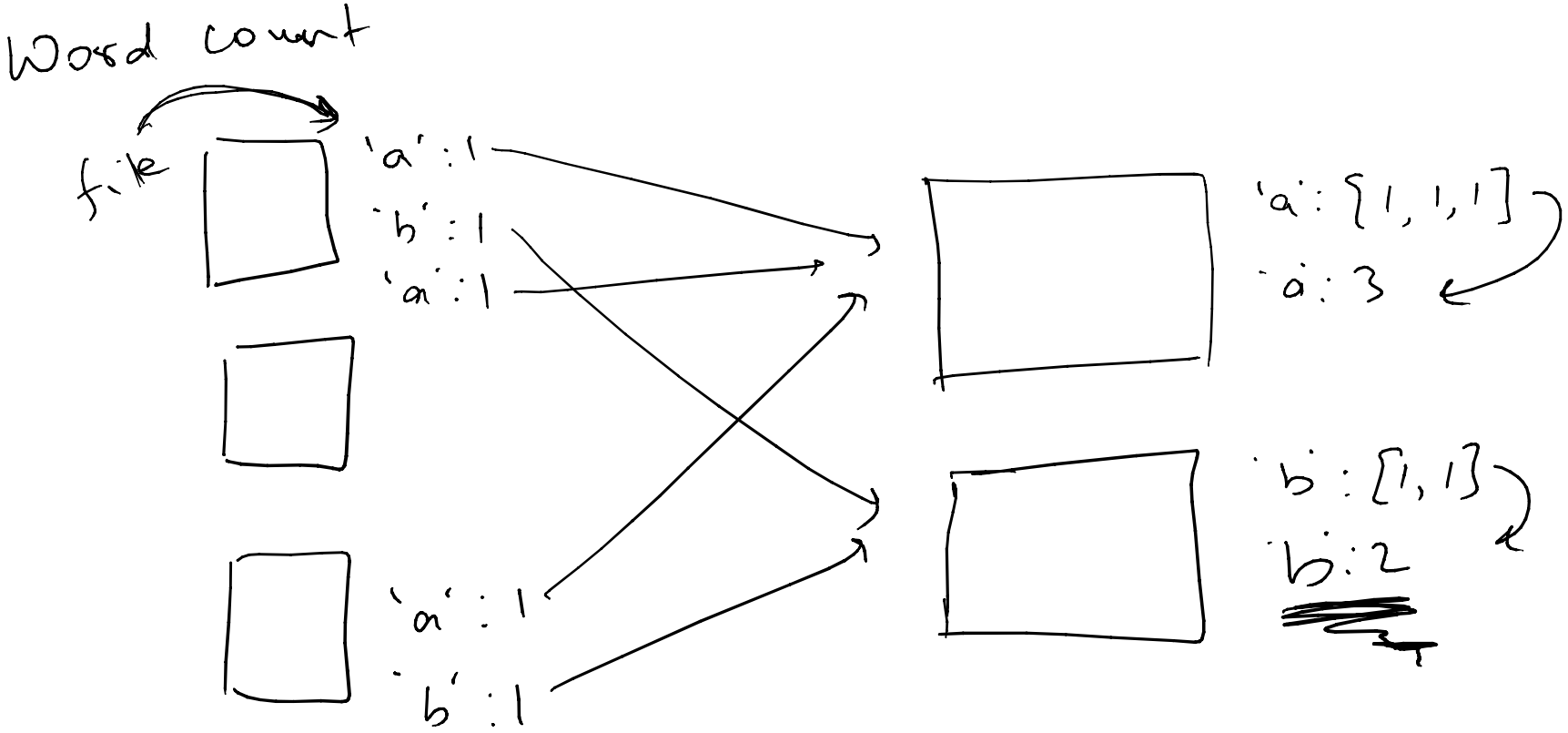
map(source doc)

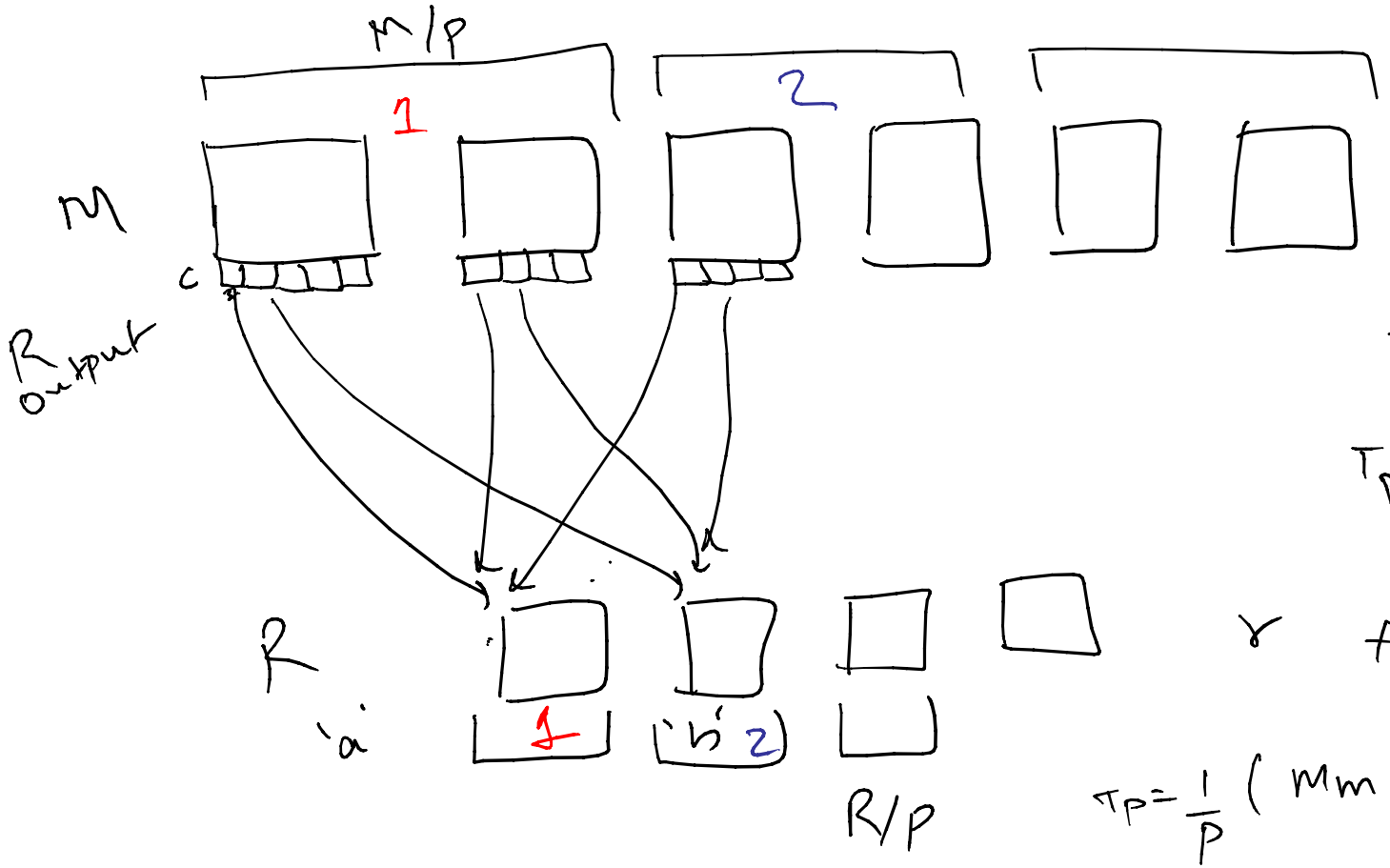
parse

if link Emit (<target, source>)

Reduce (target,)

(target, list(sources))


Word count
file
'a' : 1
'b' : 1
'a' : 1
'a' : 1
'b' : 1
'a' : [1, 1, 1]
'a' : 3
'b' : [1, 1]
'b' : 2



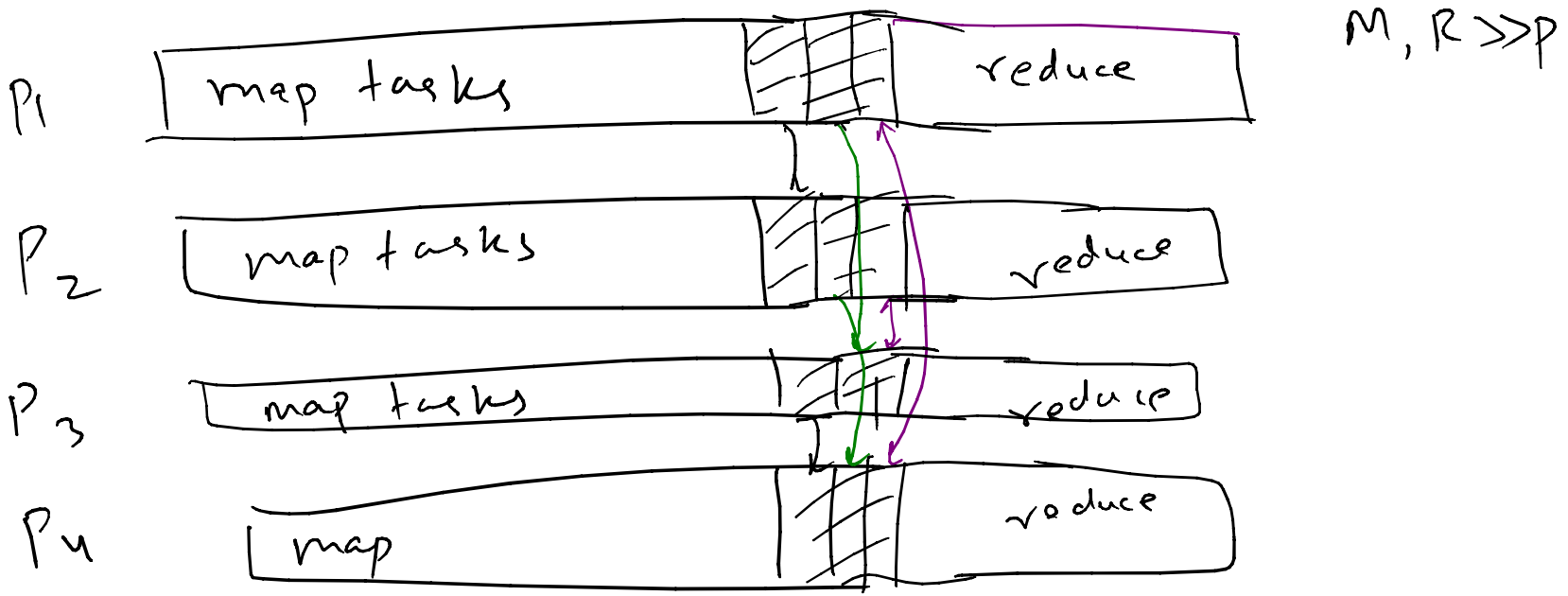
$$T_S = Mm + Rr$$

$$T_P = \frac{M}{P}m + \frac{R}{P}r + \frac{R}{P}\left(M - \frac{M}{P}\right)C$$

$$T_P = \frac{1}{P}\left(Mm + Rr + MRC\frac{(P-1)}{P}\right)$$

$$T_p = \frac{1}{f}\left(Mm + Rr + MR\left(\frac{(p-1)}{p}\right)\right) \qquad T_s = Mm + Rr$$

$$\eta = \frac{Mm + Rr}{Mm + Rr + MR_c\frac{(p-1)}{p}} = \frac{1}{1 + \left(\frac{MR_c\frac{(p-1)}{p}}{Mm + Rr}\right)}$$


M, R >> p
P1
map tasks
reduce
P2
map tasks
reduce
P3
map tasks
reduce
P4
map
reduce

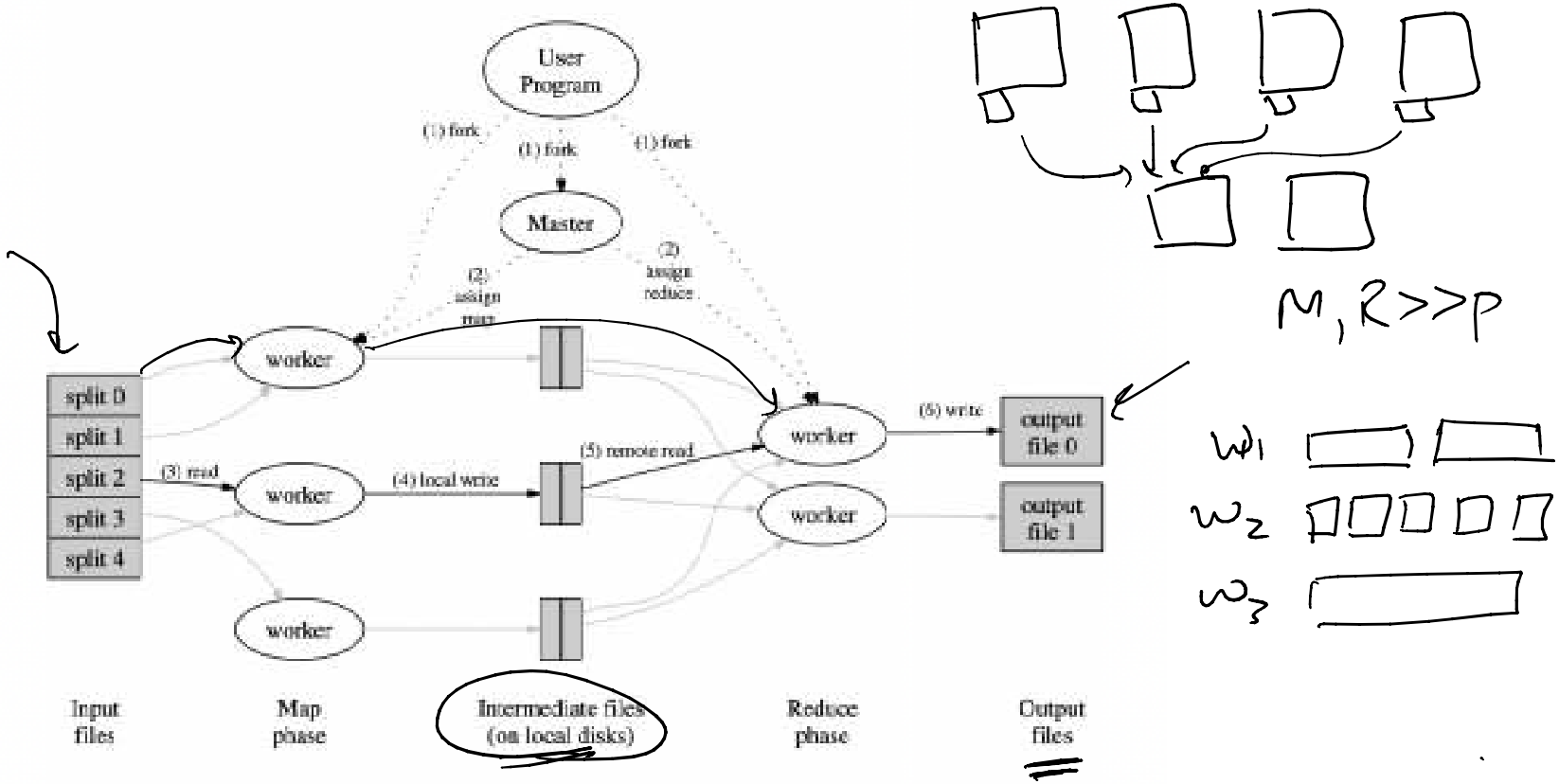


Figure 1: Execution overview

Master data structure

Tasks — Status — worker

M_1 — idle → in-progress → completed — <R files, R sizes>

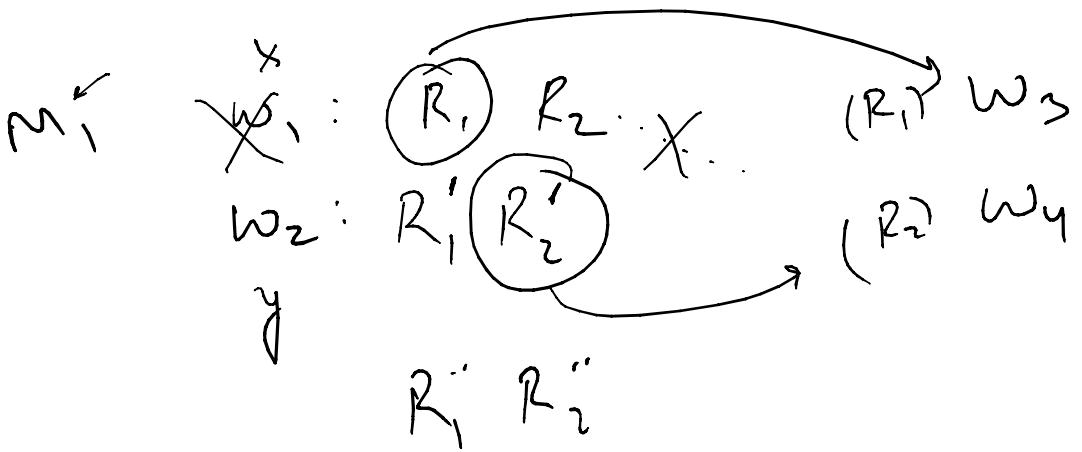
M_2 — idle

R_1

R_2

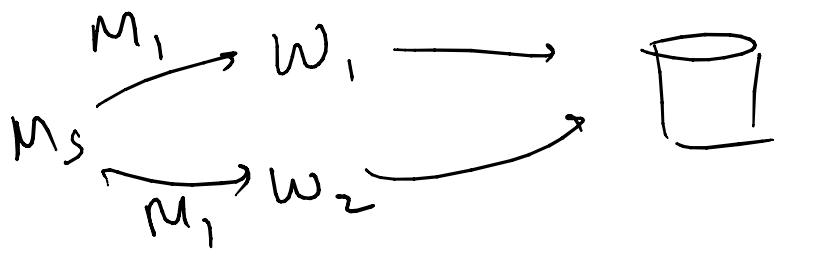
Fault tolerance -

- Rerun task on a separate worker
- HeartBeats



$R_2 \neq R_2'$

deterministic tasks



Idempotence