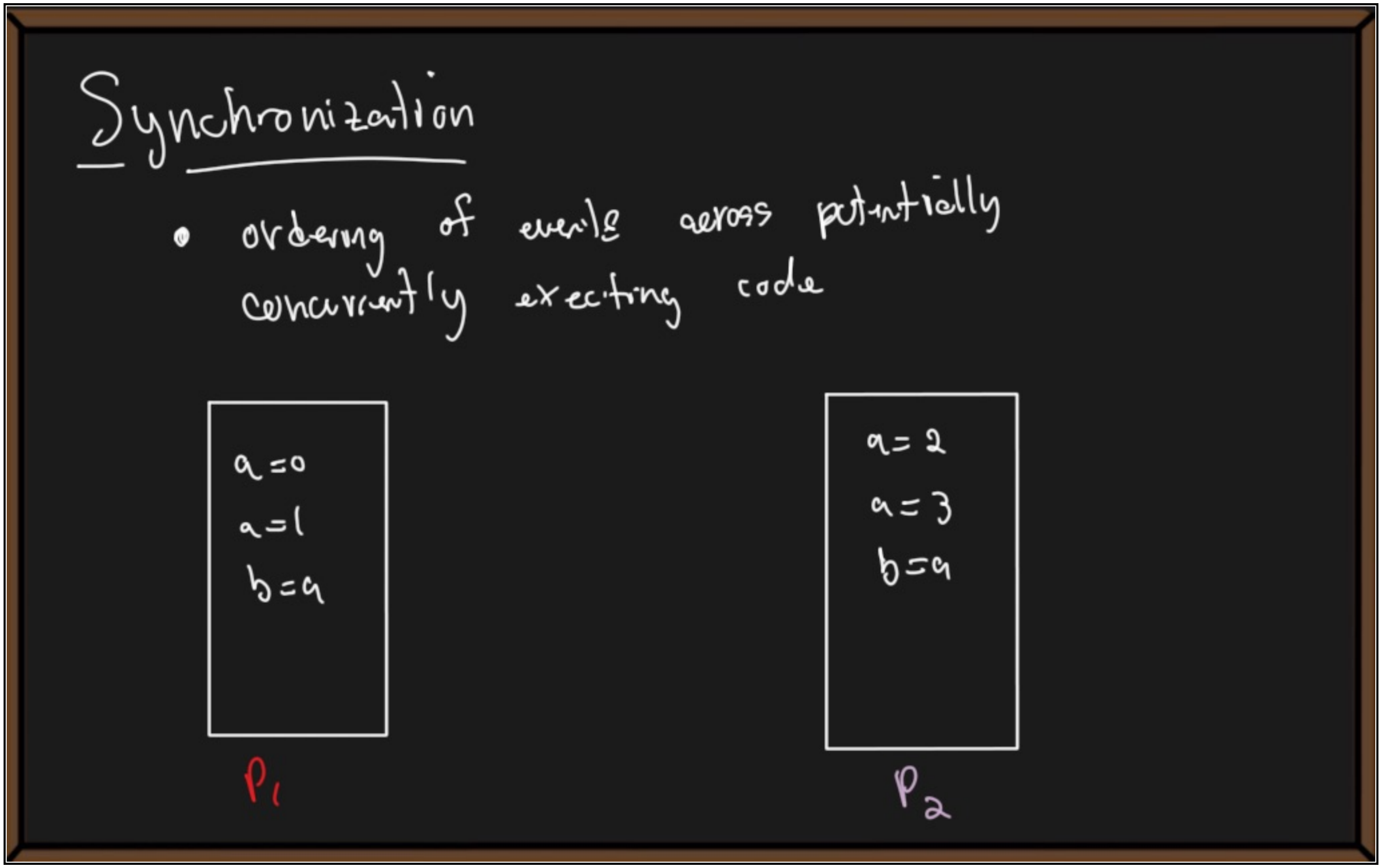


Synchronization

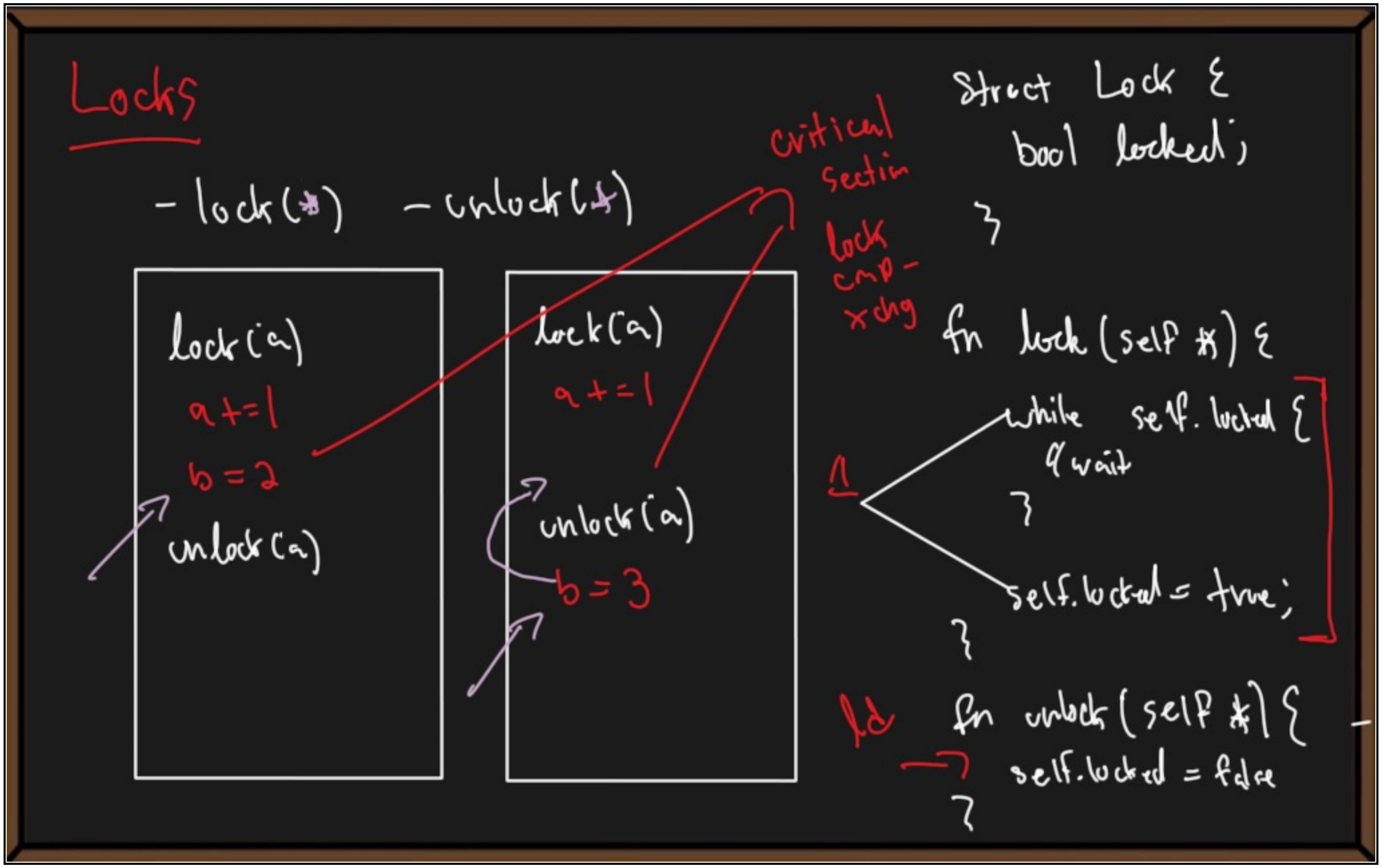
- ordering of events across potentially concurrently executing code

```
a=0
a=1
b=a
```

P_1

```
a=2
a=3
b=a
```

P_2



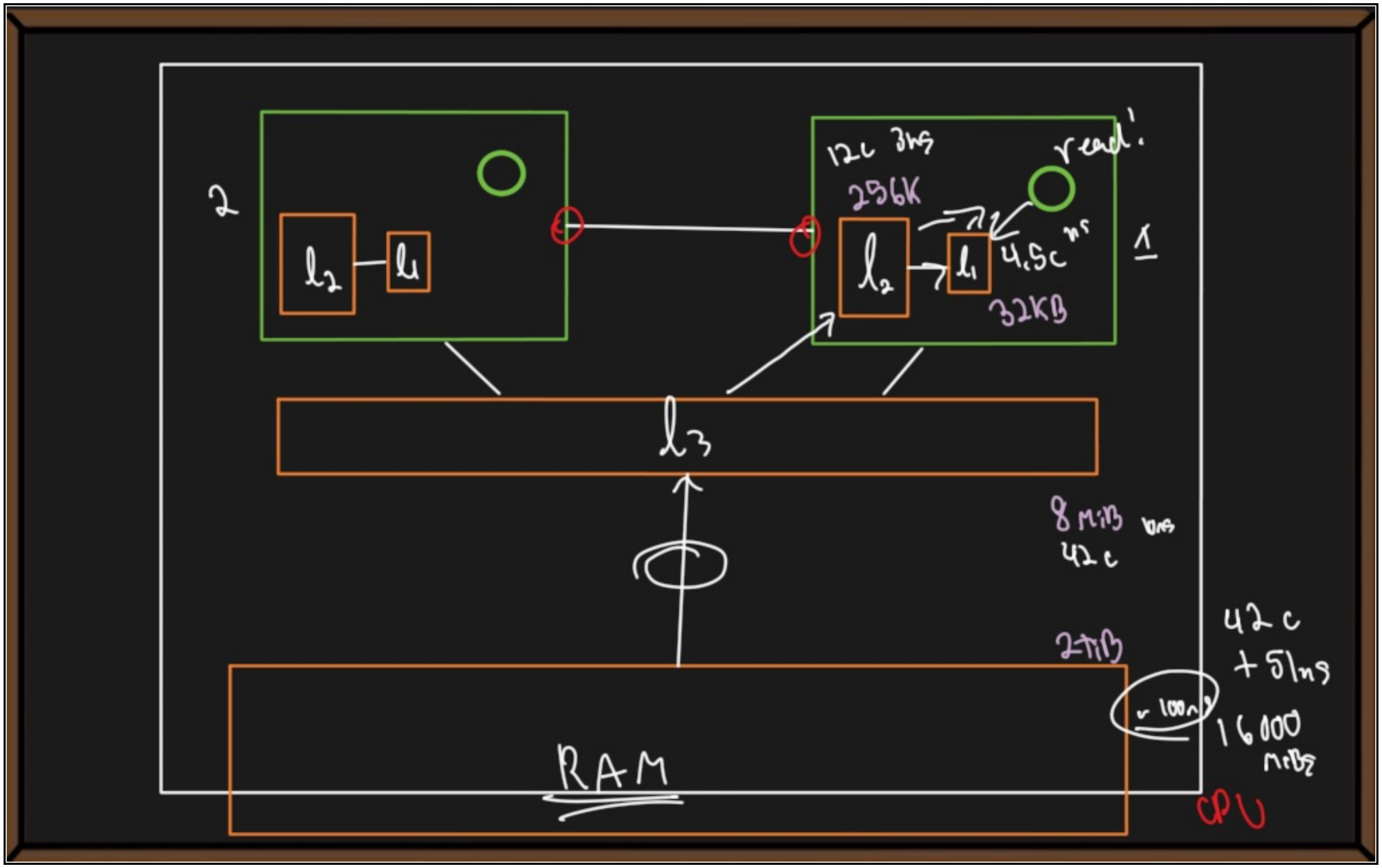
Locks

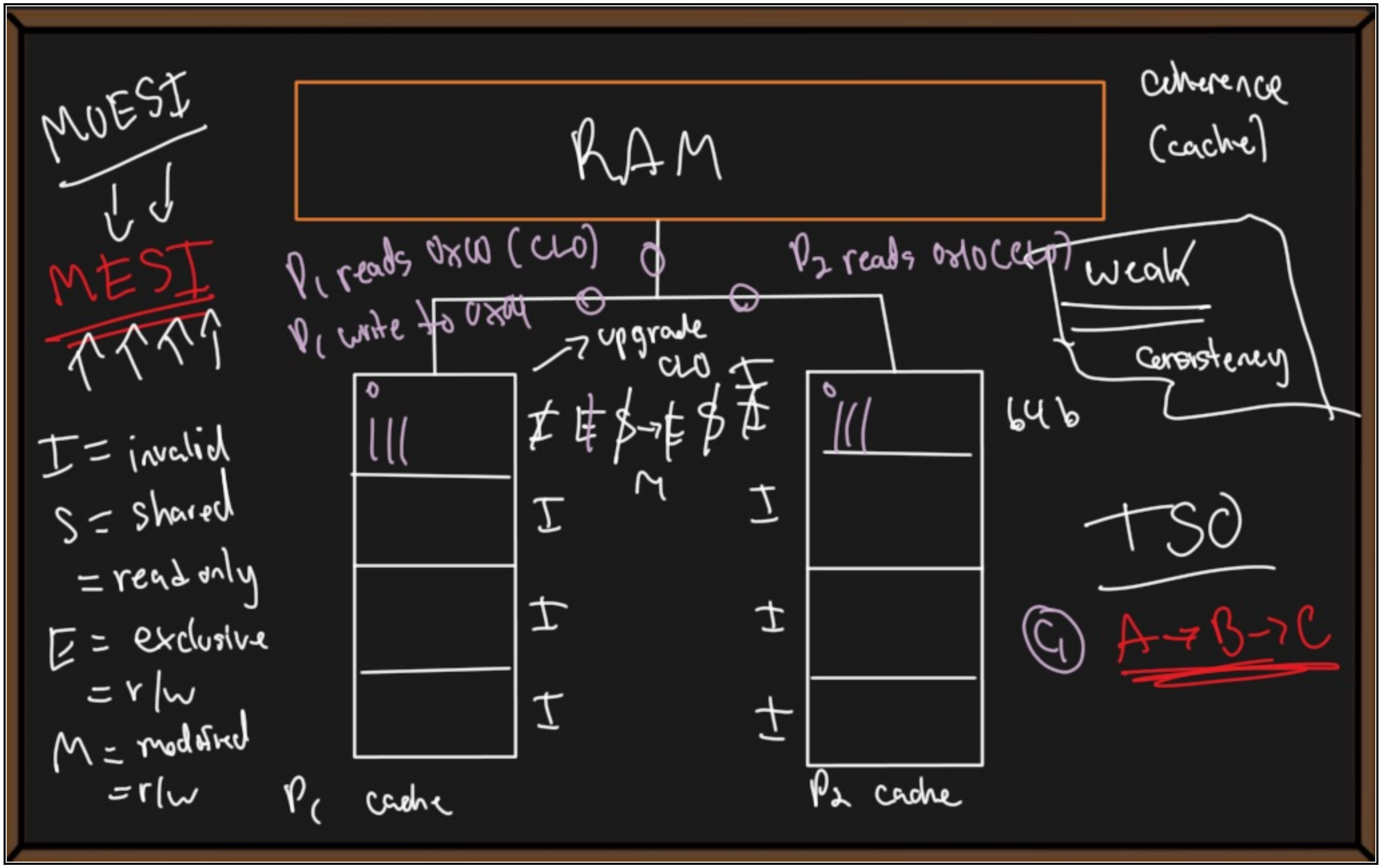
- lock(*) - unlock(*)

```
lock(a)
a+=1
b=2
unlock(a)
```

```
lock(a)
a+=1
unlock(a)
b=3
```

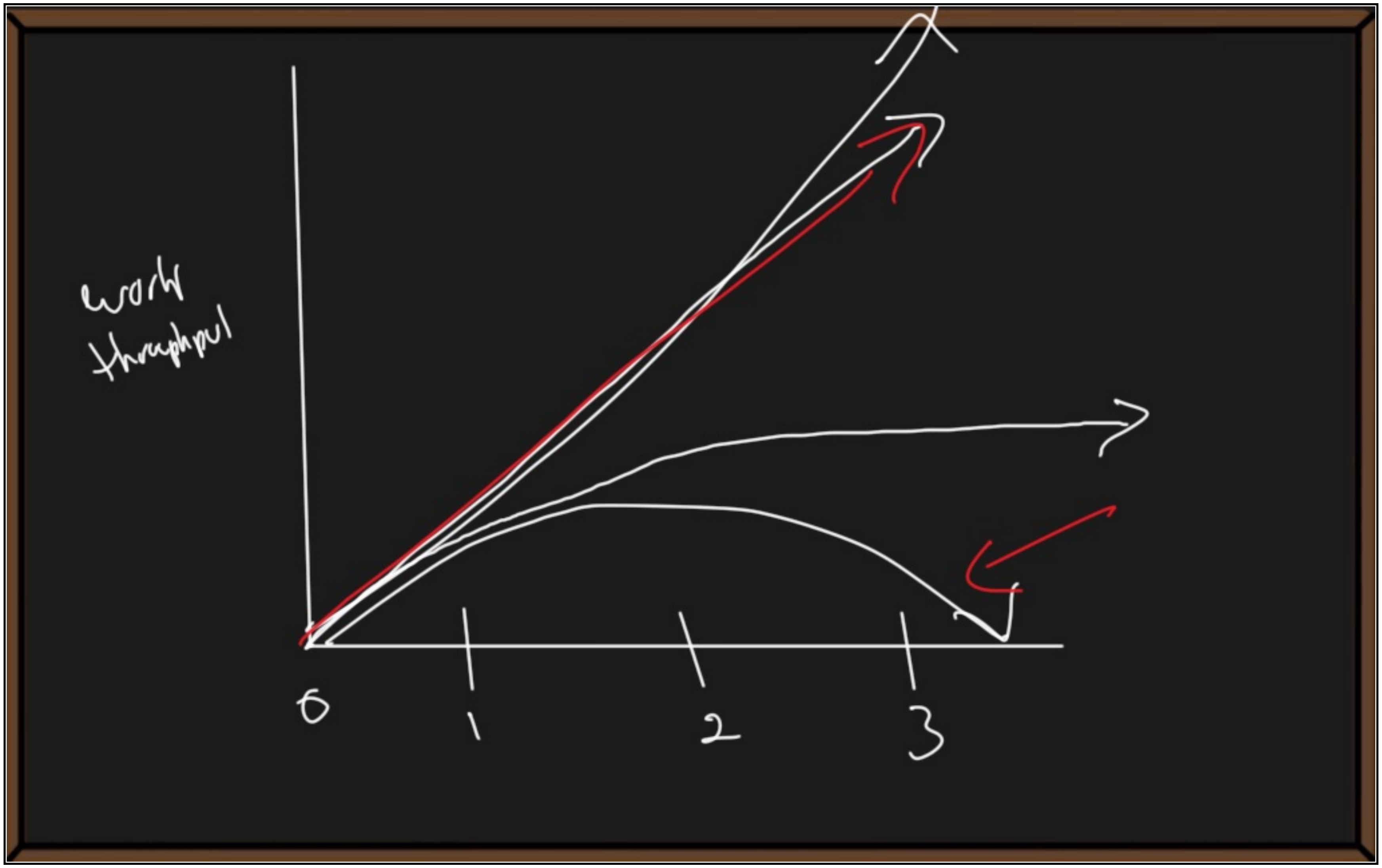
```
struct Lock {
    bool locked;
}

fn lock(self *) {
    while self.locked {
        // wait
    }
    self.locked = true;
}

fn unlock(self *) {
    self.locked = false
}
```


2
l2
l1
12c 3ns
256K
read!
l2
l1
4.5c ns
32KB
1
l3
8 MiB 10ns
42 c
2TiB
42 c
+ 51ns
~ 100ns
16000 MiB/s
RAM
CPU



I = invalid
S = shared
= read only
E = exclusive
= r/w
M = modified
= r/w

P1 reads 0x00 (CL0)
P1 write to 0x04
P2 reads 0x10 (CL0)


work throughput
0
1
2
3