CSCI 1380 : Day 19

Today

Distributed Transactions concluded

(a) 2PC

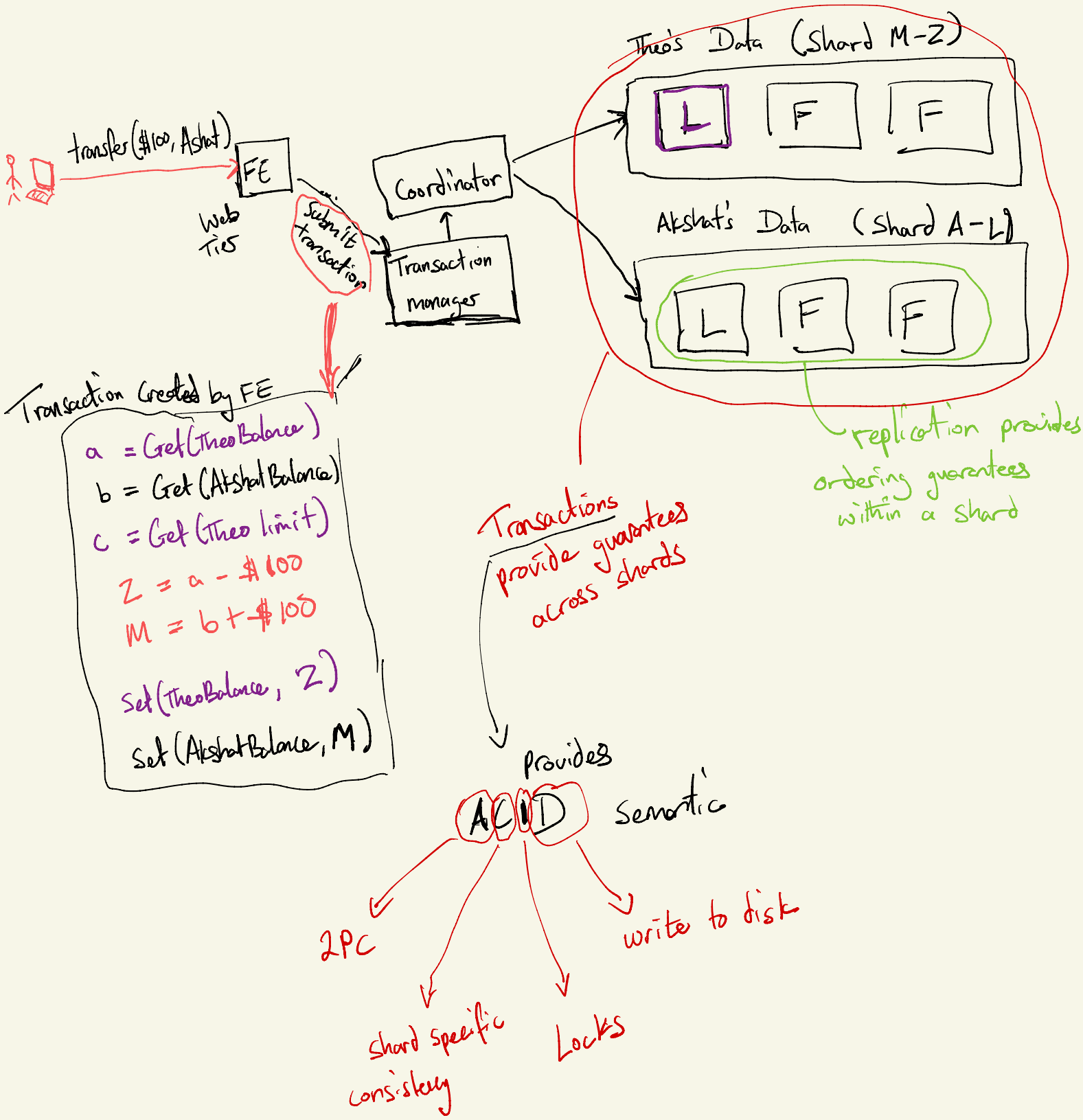
(5) Isolation (locking)

Distributed file systems

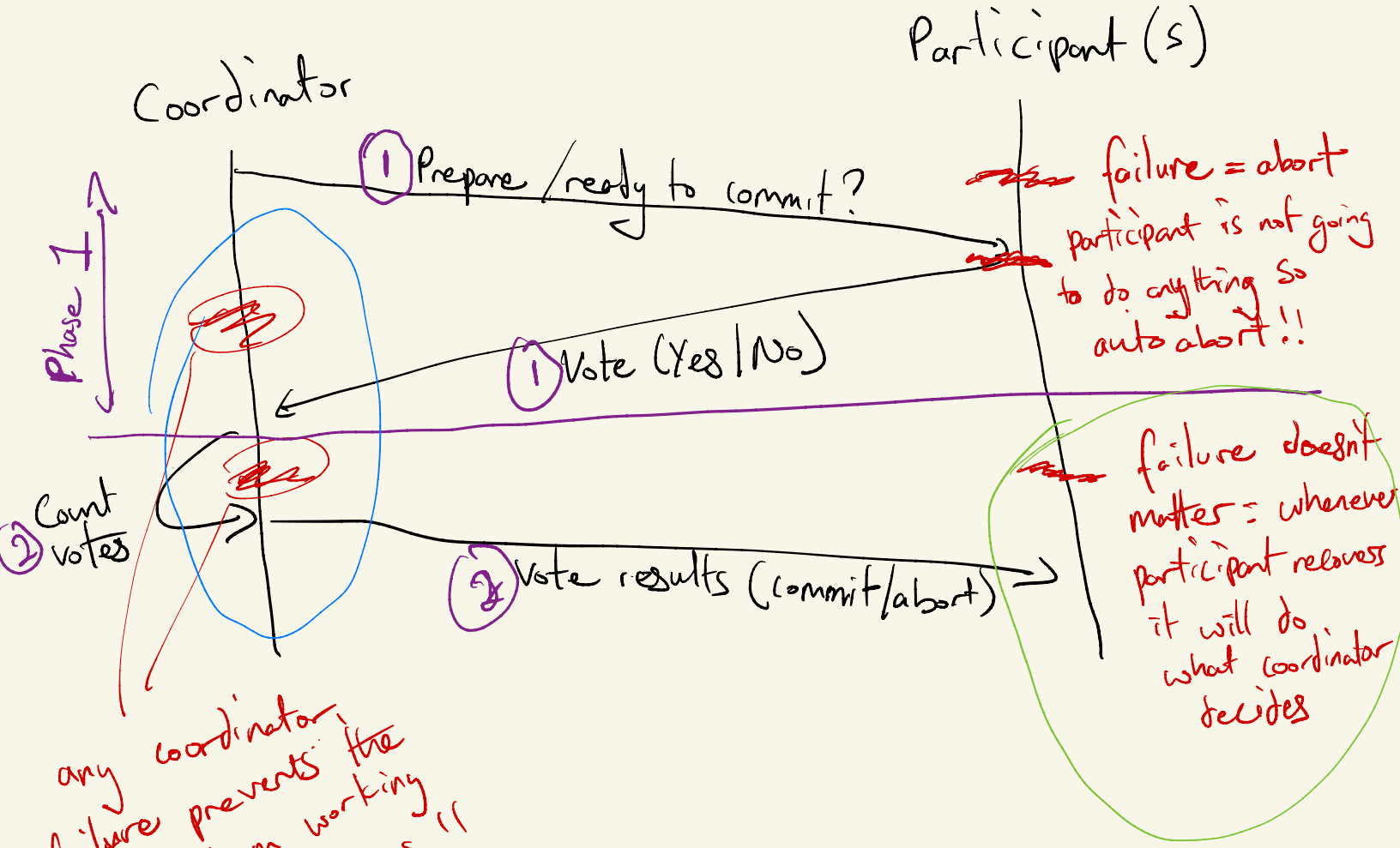
(1) Scalability: metadata V. data

(2) Performance: locks V leases & caching

(3) Security: access control list V. capabilities


transfer($100, Akshat)
FE
Web Tier
submit transaction
Coordinator
Transaction manager
Theo's Data (Shard M-Z)
L
F
F
Akshat's Data (Shard A-L)
L
F
F
replication provides ordering guarantees within a shard
Transaction Created by FE
a = Get(TheoBalance)
b = Get(AkshatBalance)
c = Get(Theo limit)
Z = a - $100
M = b + $100
Set(TheoBalance, Z)
Set(AkshatBalance, M)
Transactions provide guarantees across shards
provides
ACID
Semantic
2PC
write to disk
Shard specific consistency
Locks

2PC

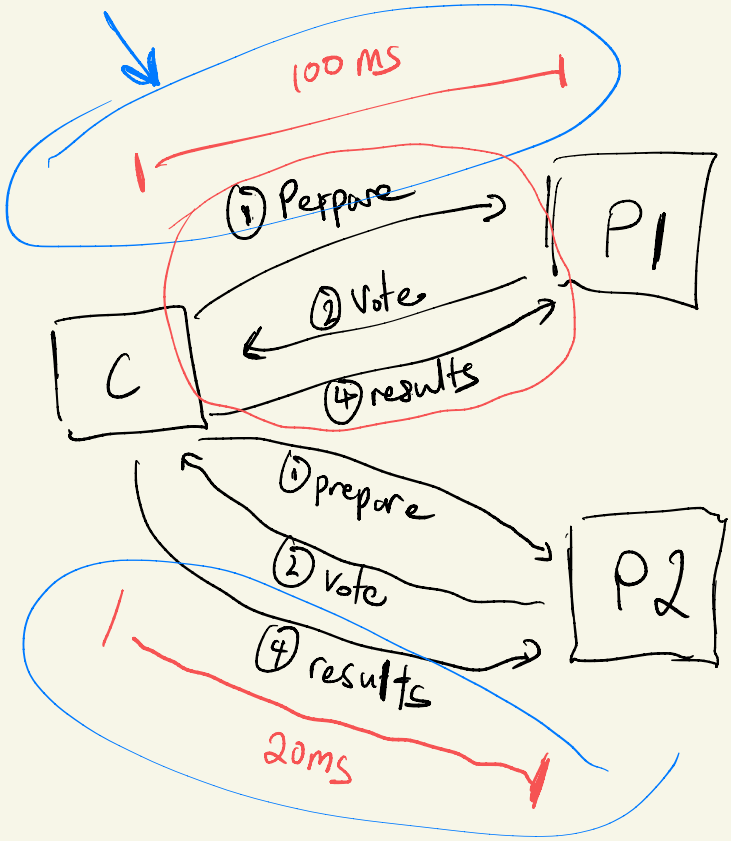


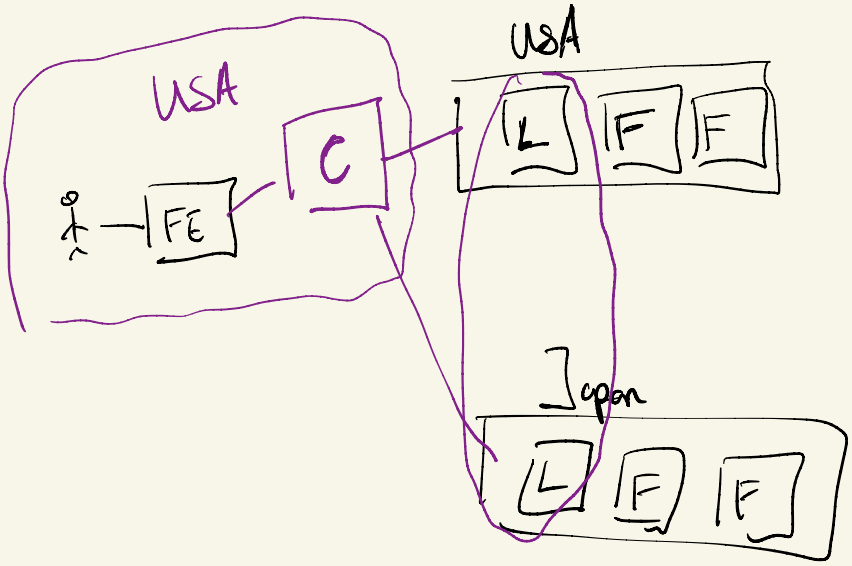
With 2PC which CAP guarantees do you get?

C A ~~P~~

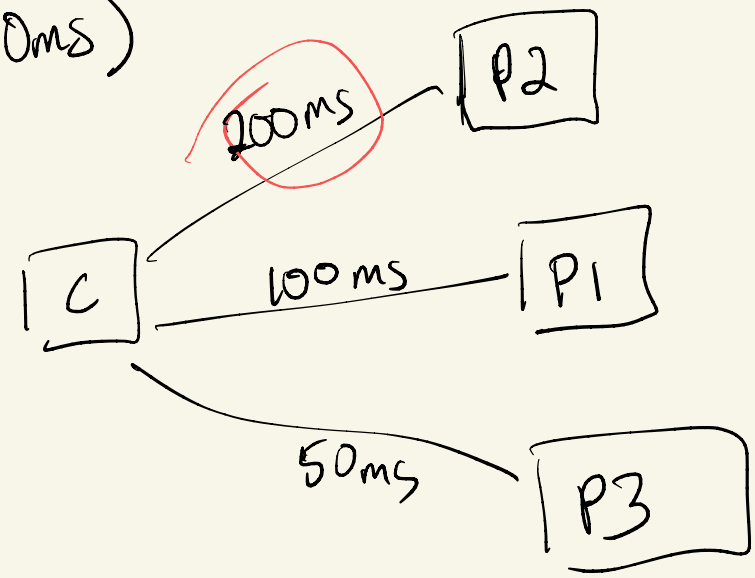
Problems With 2PC

The speed of 2PC is as fast as the slowest network latency





2PC = 3 (200ms)



Providing Isolation (locks)

```
a = Get(TheoBalance)
b = Get(AkshatBalance)
c = Get(Theo limit)
Z = a - $100
M = b + $100
Set(TheoBalance, Z)
Set(AkshatBalance, M)
```

Pessimistic

1. get all locks
2. execute all actions
3. commit
4. release all locks

Problems

1. Deadlocks
2. slow performance (transactions execute one at a time)

Optimistic

1. read all the data you need
2. execute all actions

atomic (locks)

3. check for conflicts
4. if no conflicts then commit

5. rollback if conflicts

Problems

1. can have lots of rollbacks if every one needs the same data

Pessimistic = whole transaction is protected by locks

Optimistic = only a small of the transaction is protected by locks

Optimistic ⇒ (1) perfect when transactions use different

(2) transactions operate in parallel

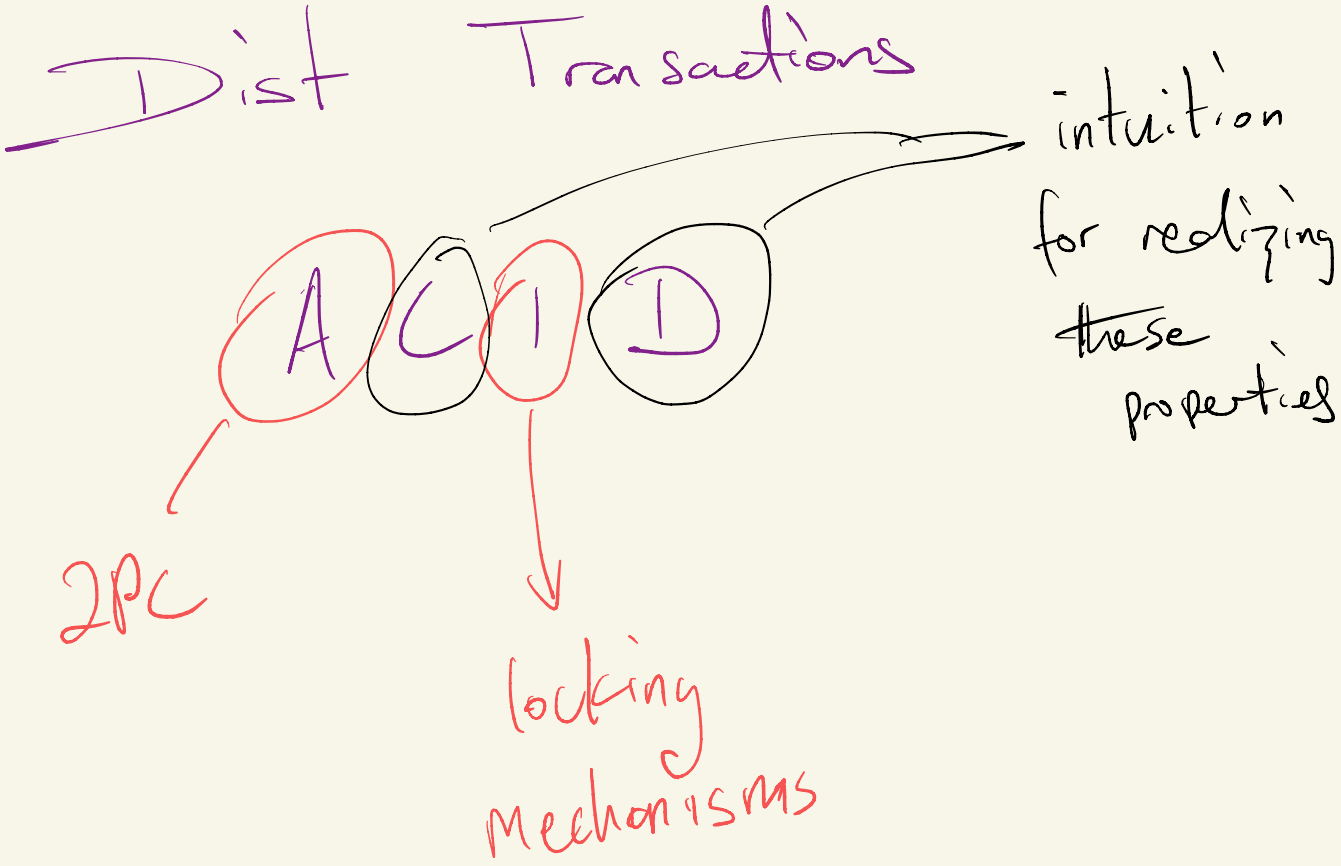
high level of concurrency

Pessimistic ⇒ (1) transactions operate one at a time

(a) serial execution creates perf. problems

(b) locks can lead to deadlocks

Dist Transactions



Distributed File systems

Data V. meta Data

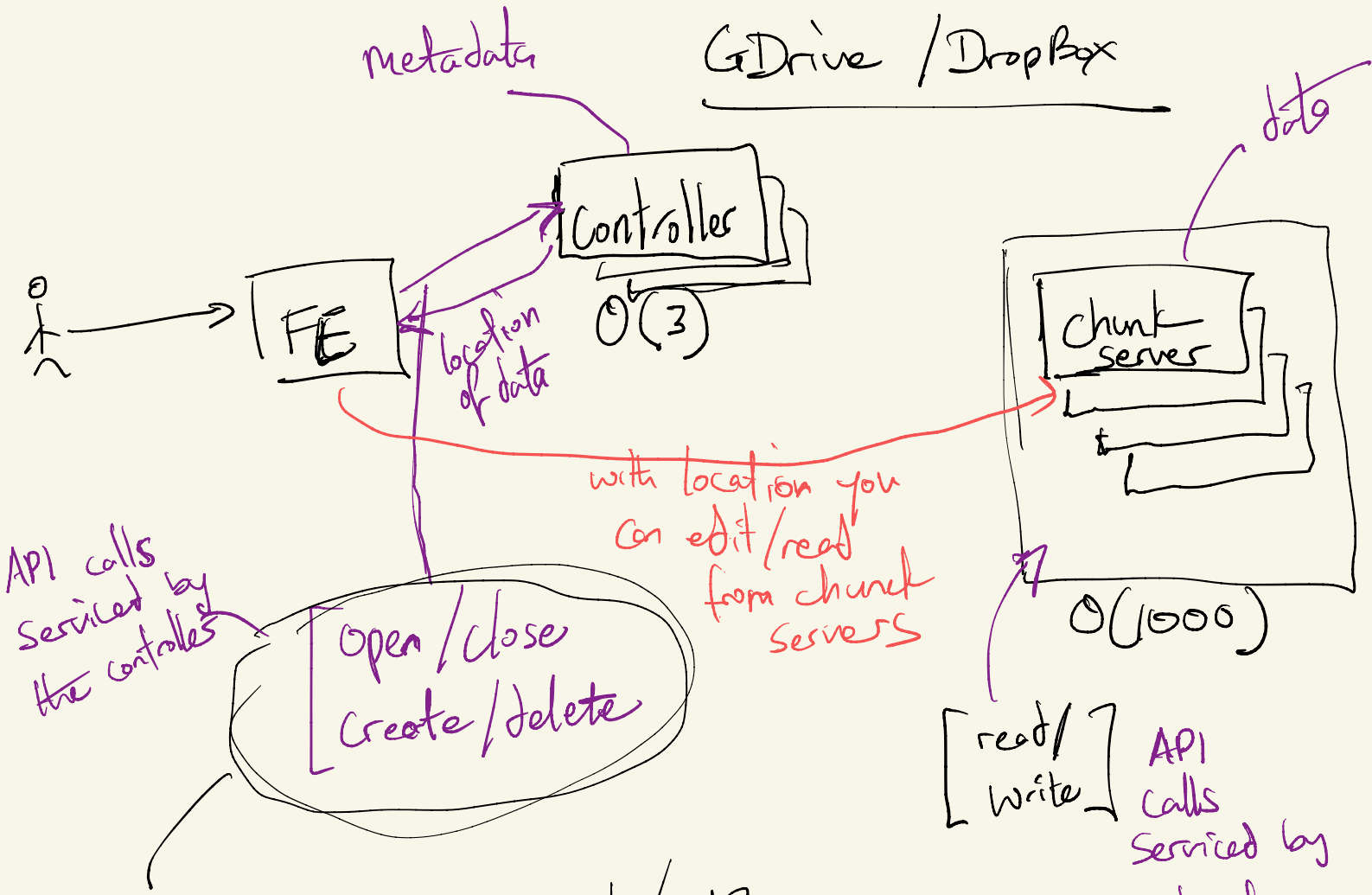
Data

contents of the file

meta Data

is data about the data

1. file size
2. Creation date
3. Access control
4. File type
5. Directory



(1) permissions: can you write/read?

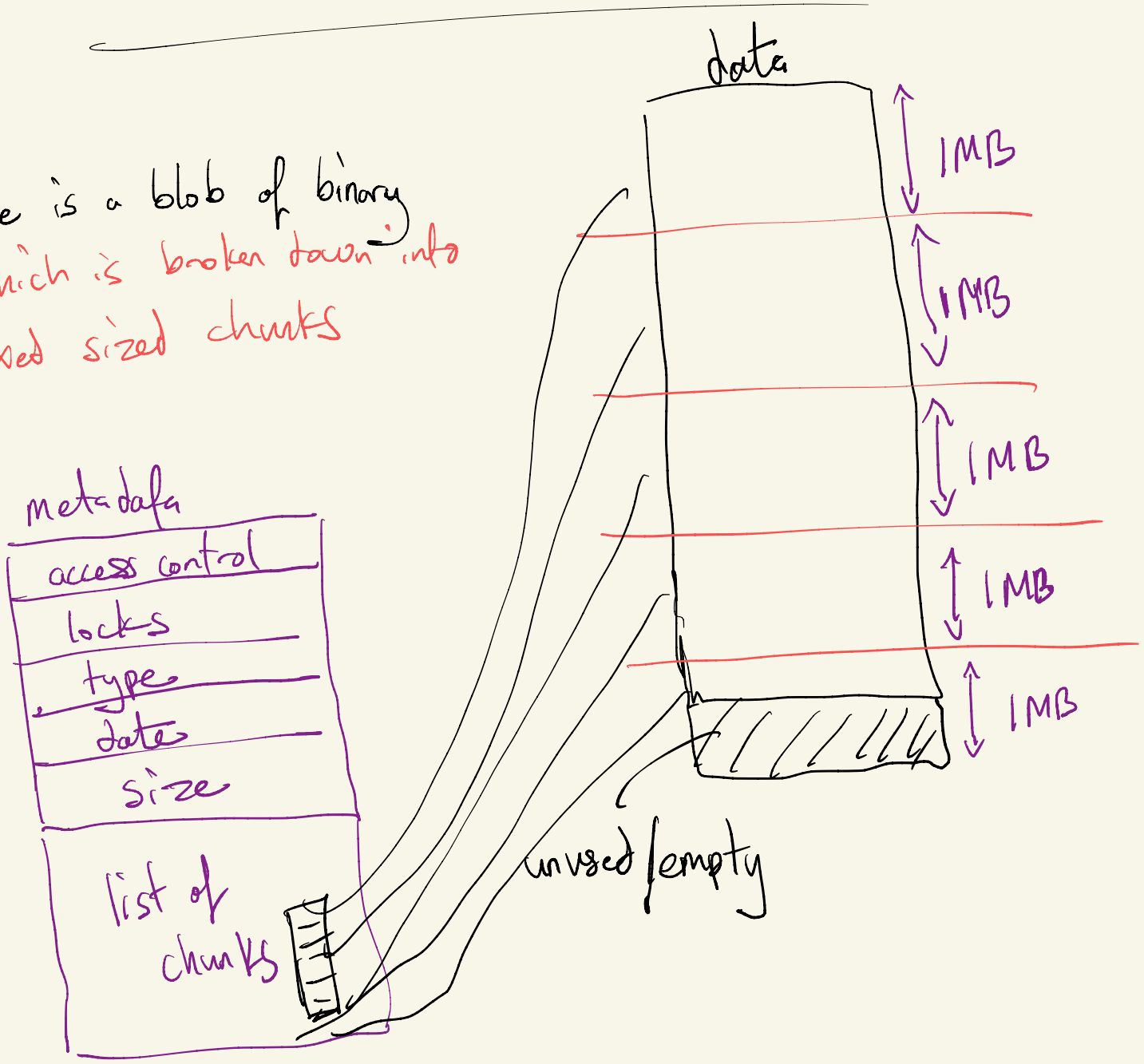
(2) acquire locks: gives exclusive access ~~to~~ chunk servers!!

DFS have a small # of controllers because they only manage metadata & thus only involved in a small # of interactions

Most of the interactions (reading/writing data) happen at the chunk servers!!

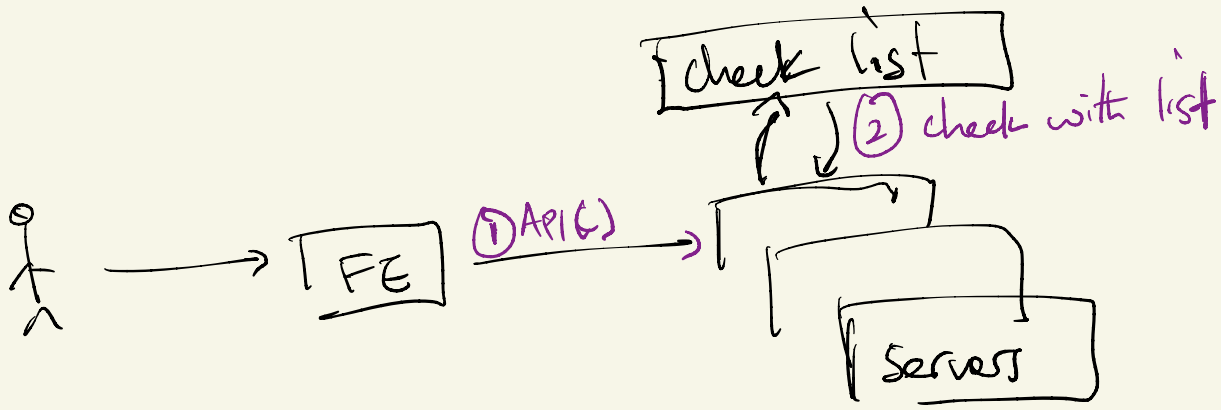
Files

File is a blob of binary which is broken down into fixed sized chunks



Security (Access control lists / capabilities)

Access control list → this is a list that's maintained at servers. each API call the servers check the list to see if the client is authorized



Easy to change the list & change permissions in the system

Quick to propagate security changes

checking ACL could add significant latency for each call!!!

Capabilities

Capability is a token which encodes permissions

anyone with the token can use any APIs the token enables

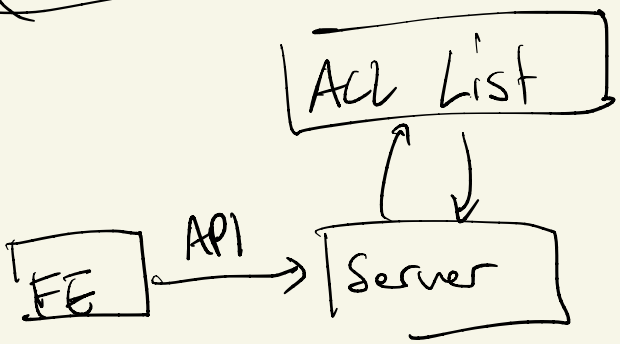
Servers use the token to validate permissions

you can make copies & distribute tokens

To change permissions: you need to track / find all tokens

Security technique for distributed file system

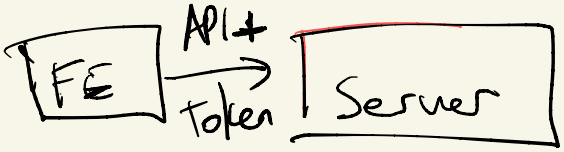
ACL (access control lists)



① easy to permissions

② perf degradation because each call requires using the ACL

Capabilities (Tokens)

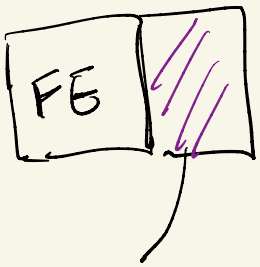


① nice & fast to verify

② Hard to change permissions because you need to track all tokens

- attach a timeout to Tokens
- have a revocation list of bad tokens

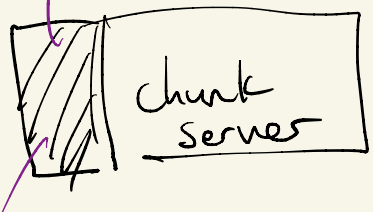
Caching



client side cache

① reduces load on the servers (because you don't need to go to server)

② all operations to cache are extremely fast because they are local (no network or server is involved !!!)



server side cache

server side cache is an in memory data

The goal of this cache is to speed up reads/writes because the operations are in memory.

Memory >>> Disk

Since memory is faster than disk ⇒ if operations are in memory then they are faster

Today

1. 2PC
2. Isolation
3. Distribute file system