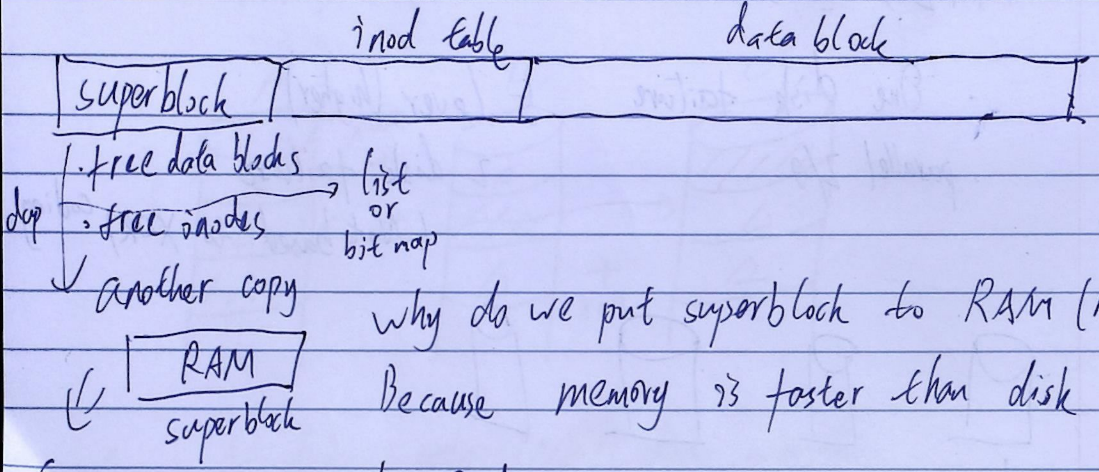
UNIX File System Consistency



why do we put superblock to RAM (memory)

Because memory is faster than disk

Consistency when Os boot, Os copy it to RAM

a data block is either on the free list (super block)

or assigned to a single file.

X

if the kernel ① freed a disk data block (in a file)

② returning the block number to the in-core copy of the superblock

③ and ③ allocated the disk data block to a new file

"disk crashes" X assume the file does not need this data block

why Consistency ?

case 1): if the kernel wrote the inode and blocks of the new file to disk, but crashed before updating the inode of the old file to disk

Lead to => two inodes would address the same disk block number

case 2) if the kernel wrote the super block and to free list to disk and crashed before writing the old inode out

this data block still in free list vs. the data block still in old inode

↓ so issue occurs.

case 3) if the old file was written on disk and crashed before the superblock was written to disk.

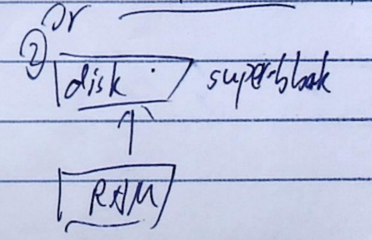
-> this data block appear no where

① the old file delete the inode which point to this data block

② before update this data block to free list in ① superblock (RAM) or ②

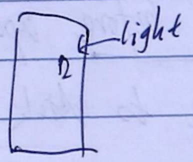
③ Os crashed

④ this data block will be no where



How to solve these issues?

"Journalling"



yellow blinking ⇔ "periodically"

5 ~ 10 seconds

Ch6 Synchronization in DS vs. "Centralized Sys"

Properties of distributed algorithms:

1) ~~#~~. The relevant info is scatterred among machines

2) Each process make decisions based on local information

3) a single point of failure in the Sys should be avoided.

4) No common clock or global time source exists

↓ E.g. A, B share same disk

machine A 1	time	clock is slower than A machine 2
A> make ⋮		B>
A> a.out →6PM	→5:50pm	B> vim .c (change the file) ↓ 5:51 (time stamp)
A> vim .c 6:01 PM	↓	B> make all up to date
A> make gcc -c .. a.out		B>

a processor timer (usually quartz crystal)

generate interrupts periodically.

. Each interrupt is called a clock tick

not perfect ⇒ clock skew

physical clock: agree with real time

logical clock: all machines agree on the same time order.