CS 423
Operating System Design: Persistence: Crash Consistency
04/25

Ram Alagappan

RECAP

FS CALLS

Basic: open, read, write, close
fsync, rename, link, unlink

How the FS implements these calls
We saw an example of VSFS

VSFS

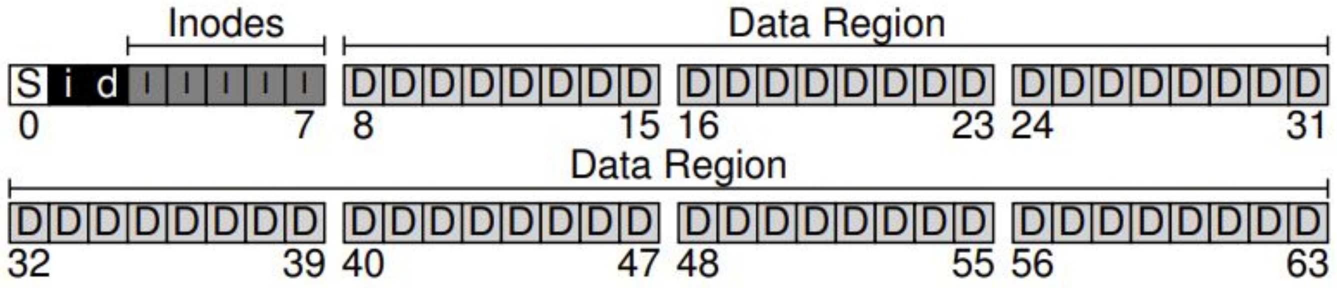
Very Simple File System

Two aspects:

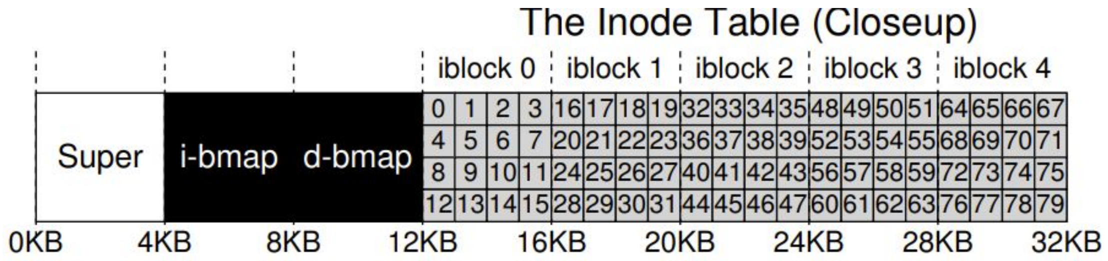
Data structures – how are files, directories, etc stored on disk

Access methods – how are high-level operations like open, read, write mapped to these DS operations

VSFS – Superblock (metadata)

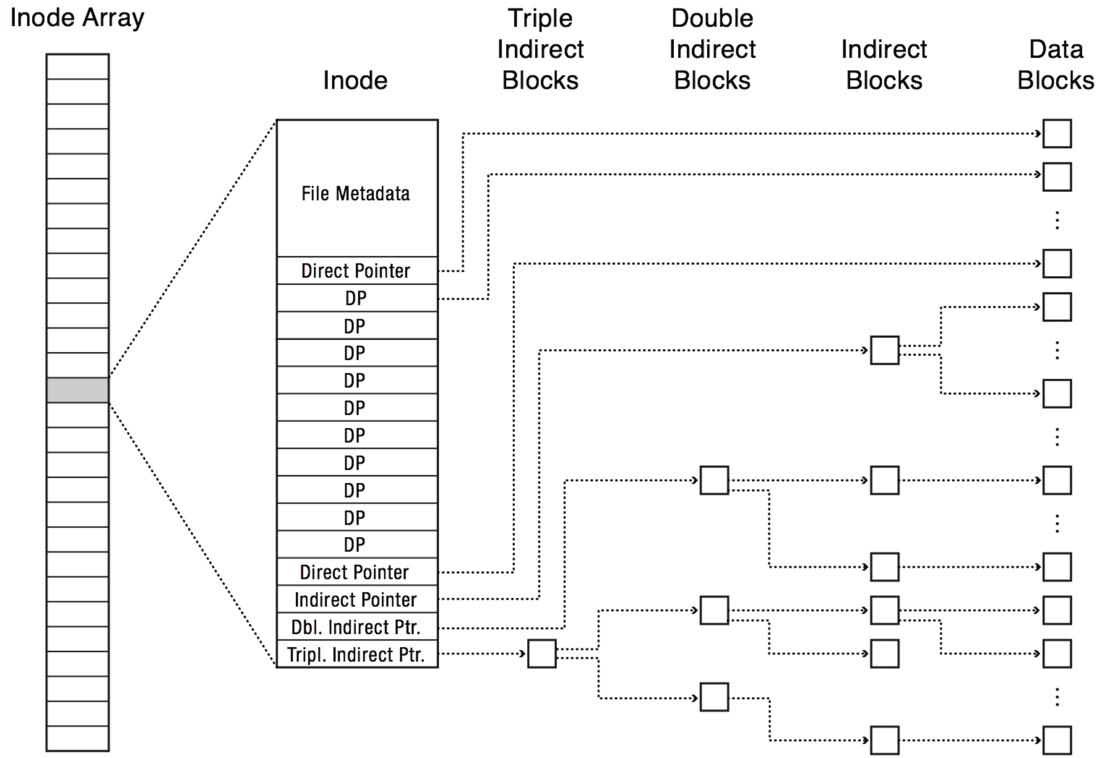


INODE

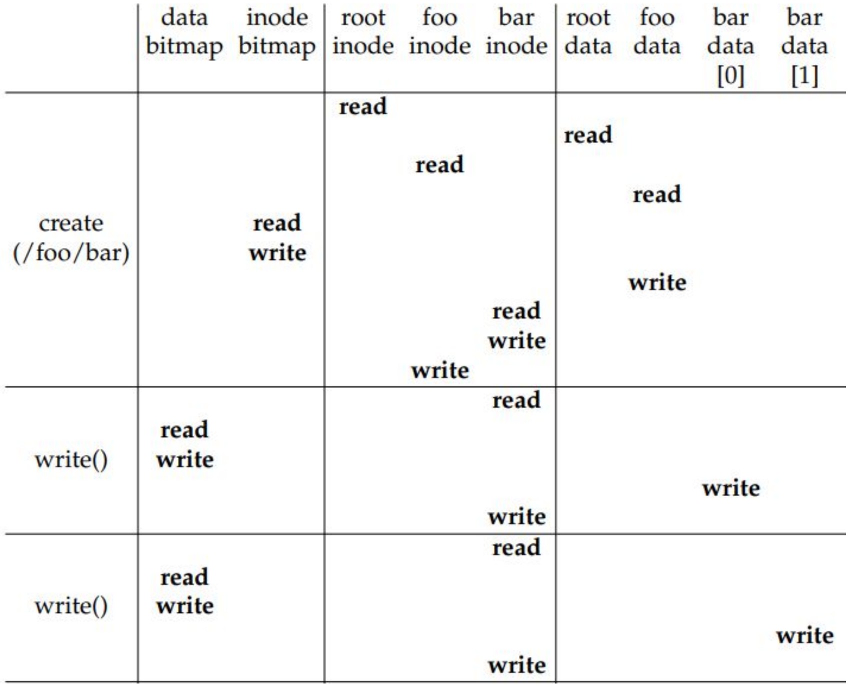


Implicitly know the block/sector number

Direct and Indirect Pointers



Creating and Writing File



	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]
create (/foo/bar)			read						
						read			
				read					
							read		
		read							
		write							
							write		
					read				
					write				
				write					
write()					read				
	read								
	write								
								write	
					write				
write()					read				
	read								
	write								
									write
					write				

Why read foo data?

What is written in foo data?

What is written in foo inode?

why is bar inode written upon data write?

END RECAP

Page Cache

Disk access is expensive

Can cache blocks in memory – all FS do this

Integrated with virtual memory

- can balance fs cache vs. vm

Also helps write buffering (need to fsync for persistence)

Flushing deamon

Crash Consistency

Basic problem:

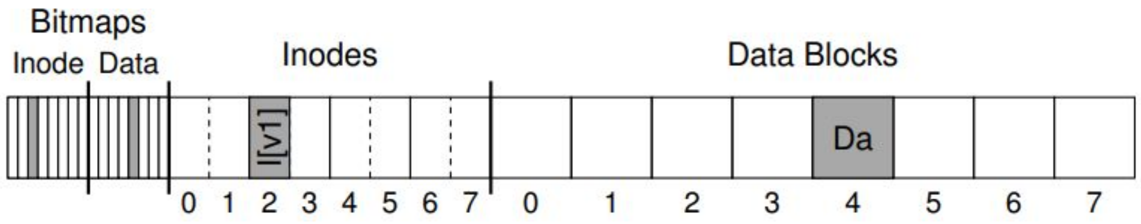
Must update many data structure on disk as a unit

What if failure happens in the middle

Types of failure:

- kernel panic
- power failures

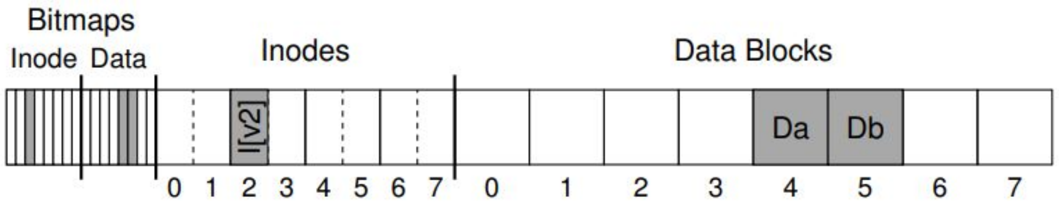
Append a Block Example



How many blocks do we need to write to accomplish the append?

Which ones?

Problems



What if only Db is written?
Only i[V2] is written to disk? (2 problems)
Data bitmap is alone written to disk?
Bitmap and data are written:
Data and inode are written:
Bitmap and inode are written:

What's special about the last case?

Metadata vs. Data

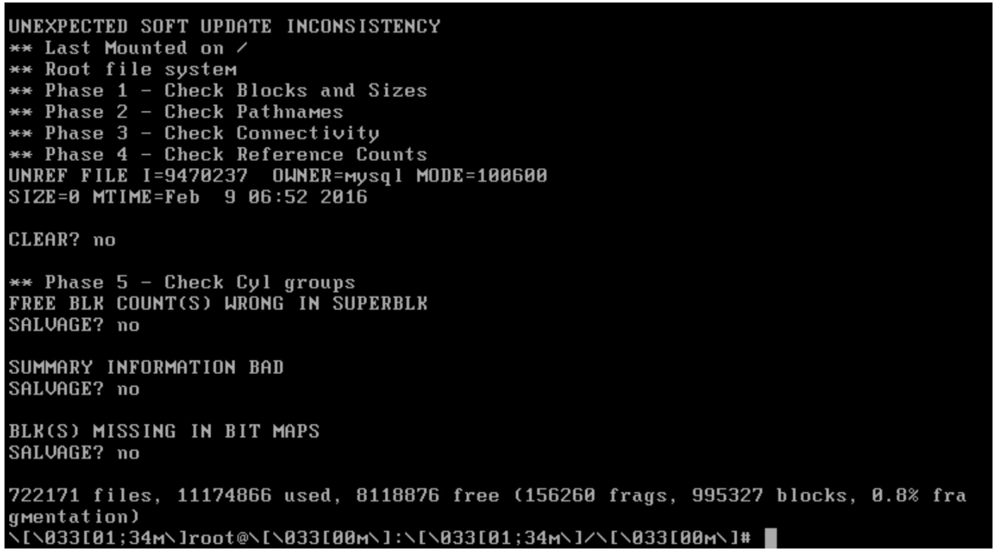
FS Metadata consistency vs. Data consistency

FS metadata consistency: internal structures agree with each other

Data consistency: additionally, the data must “make sense” to applications and users

FSCK

Let inconsistencies happen and take care during reboot

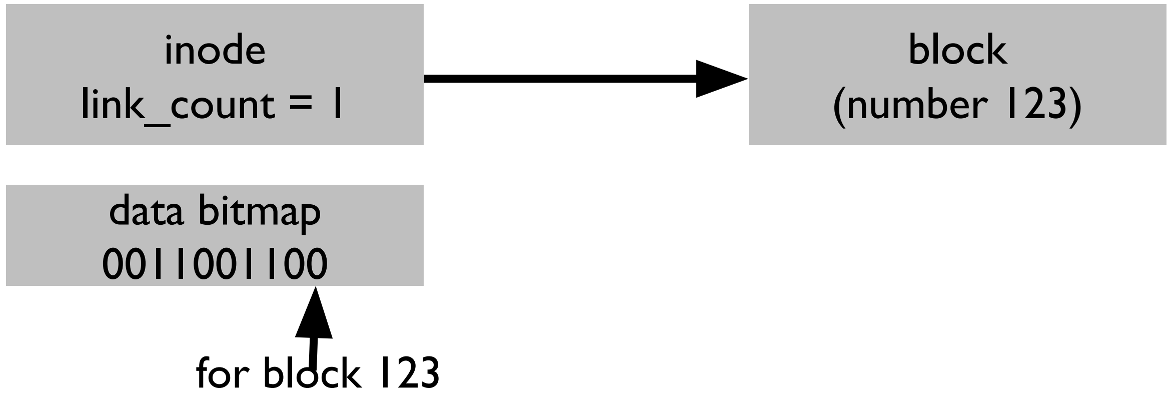


```
UNEXPECTED SOFT UPDATE INCONSISTENCY
** Last Mounted on /
** Root file system
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
UNREF FILE I=9470237  OWNER=mysql MODE=100600
SIZE=0 MTIME=Feb  9 06:52 2016

CLEAR? no

** Phase 5 - Check Cyl groups
FREE BLK COUNT(S) WRONG IN SUPERBLK
SALVAGE? no

SUMMARY INFORMATION BAD
SALVAGE? no

BLK(S) MISSING IN BIT MAPS
SALVAGE? no

722171 files, 11174866 used, 8118876 free (156260 frags, 995327 blocks, 0.8% fra
gmentation)
\[\033[01;34m\]root@\[\033[00m\]:\[\033[01;34m\]/\[\033[00m\]#
```

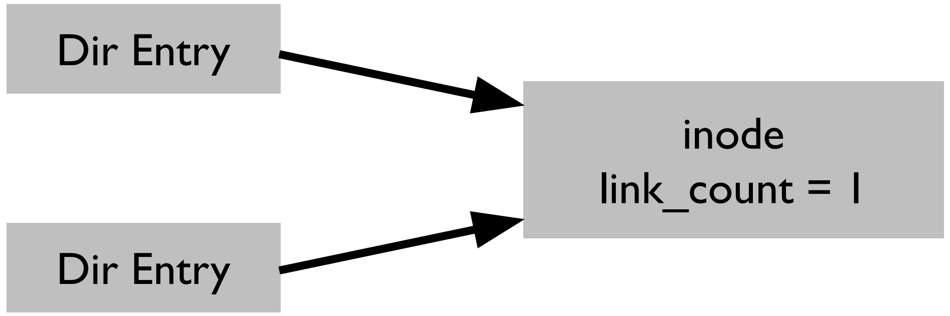
FSCK CHECKS

Do superblocks match?
Is the list of free blocks correct?
Do number of dir entries equal inode link counts?
Do different inodes ever point to same block?
Are there any bad block pointers?
Do directories contain "." and ".."?
…

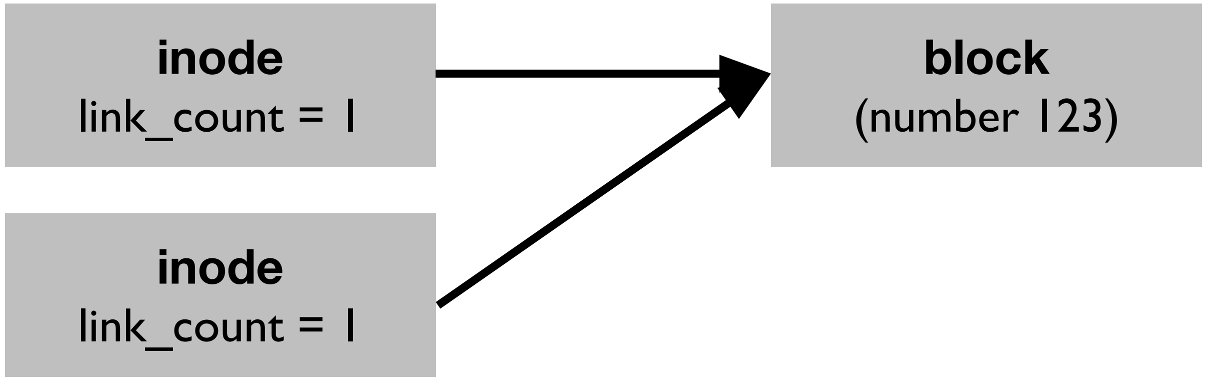
Free Blocks Example



Link Count Example



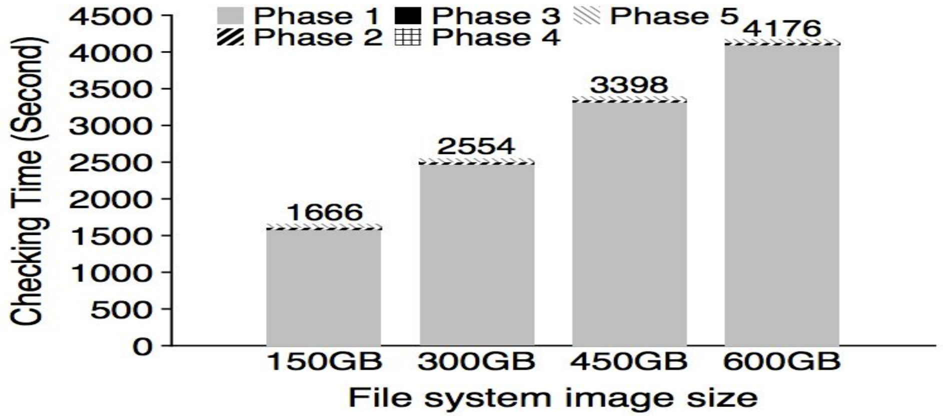
DUPLICATE POINTERS



FSCK PROBLEMS

Not always obvious how to fix file system image - don't know "correct" state, just consistent one

Simply too slow!



Checking a 600GB disk takes ~70 minutes

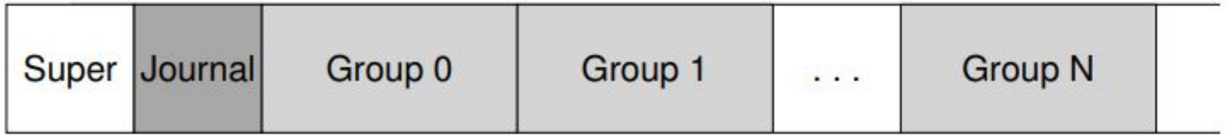
ffsck: The Fast File System Checker
Ao Ma, Chris Dragga, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau

Journaling or WAL

Main idea: write a “note” to a well-known location before actually writing the blocks

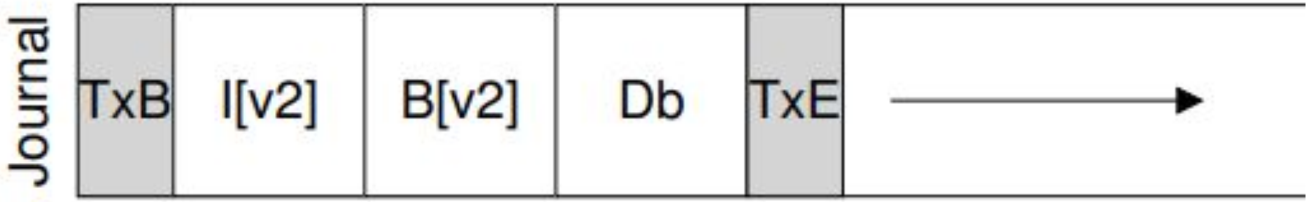
If crash, know what to fix and how to do so from the note (instead of scanning the entire disk)

Journaling in Linux ext3



Append a block to an existing file example

Journal Transaction



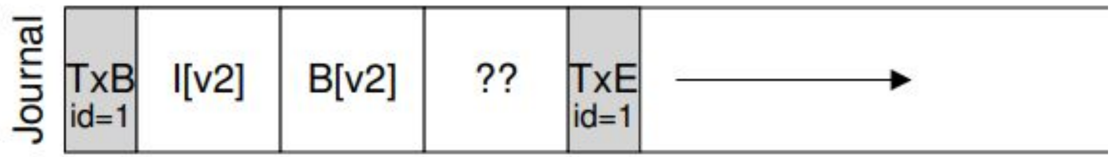
Data journaling vs. metadata journaling

Journaling or WAL

First write the txn to journal
Once that is safe, write the actual blocks (this is called checkpointing)

What if crash happens during journal write?

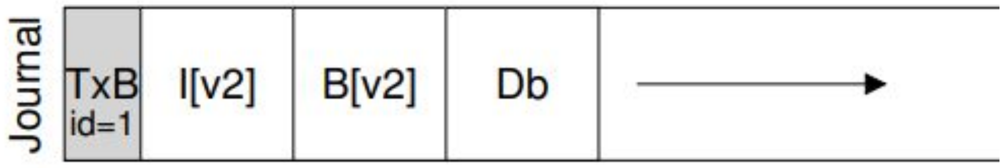
Journal Writes

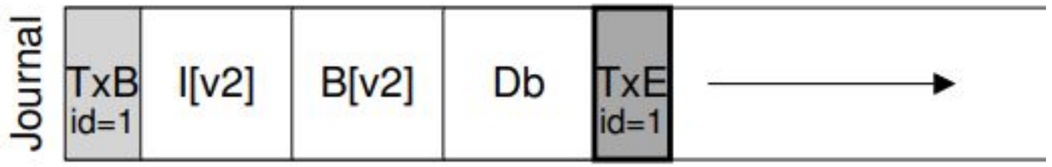


How to solve this?

Can issue one write at a time but is too slow
Must maximize how many writes can be concurrently sent
But writing all 5 blocks together is problematic

One solution





Barriers
Incurs a wait or flush between TxB + Data and TxE…
How to do without waiting?

Solution without Wait

Recovery

Scan the journal

Checkpoint completed transactions

Discard otherwise

Will the system be safe if crash happens during recovery

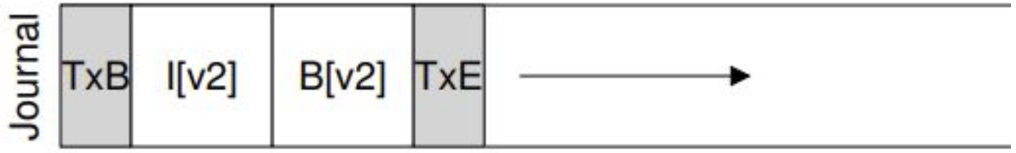
Batching for Efficiency

What is the problem with DJ?

Think about performance…

Which workload will suffer the most?

Metadata Journaling



Data blocks written in "FS proper" (in place)
Metadata goes via journal

What is the order of writes?

Order of Writes

D □ JM □ JC □ M
First data, write metadata to journal, write commit block, then checkpoint metadata

D || JM □ JC □ M (|| means concurrent)
Is this safe?