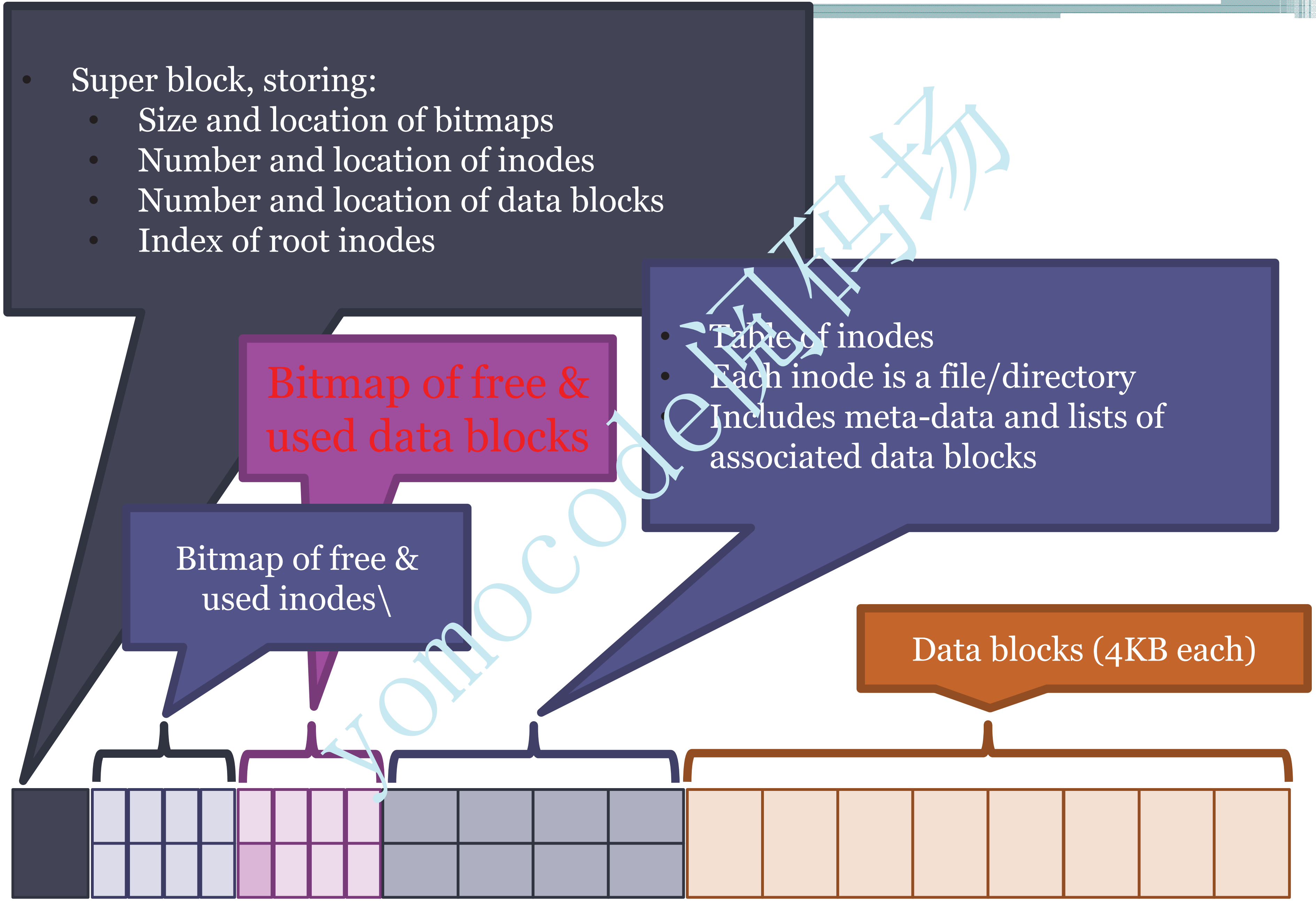
Linux铁三角之I/O(三)

麦当劳喜欢您来，喜欢您再来

文件系统的实现

*EXT2/3/4的layout
*文件系统的一致性:append一个文件的全流程
*掉电与文件系统一致性
*fsck
*文件系统的日志
*ext4 mount选项
*文件系统的debug和dump
*Copy On Write文件系统: btrfs



- Super block, storing:
  - Size and location of bitmaps
  - Number and location of inodes
  - Number and location of data blocks
  - Index of root inodes

Bitmap of free & used data blocks

Bitmap of free & used inodes\

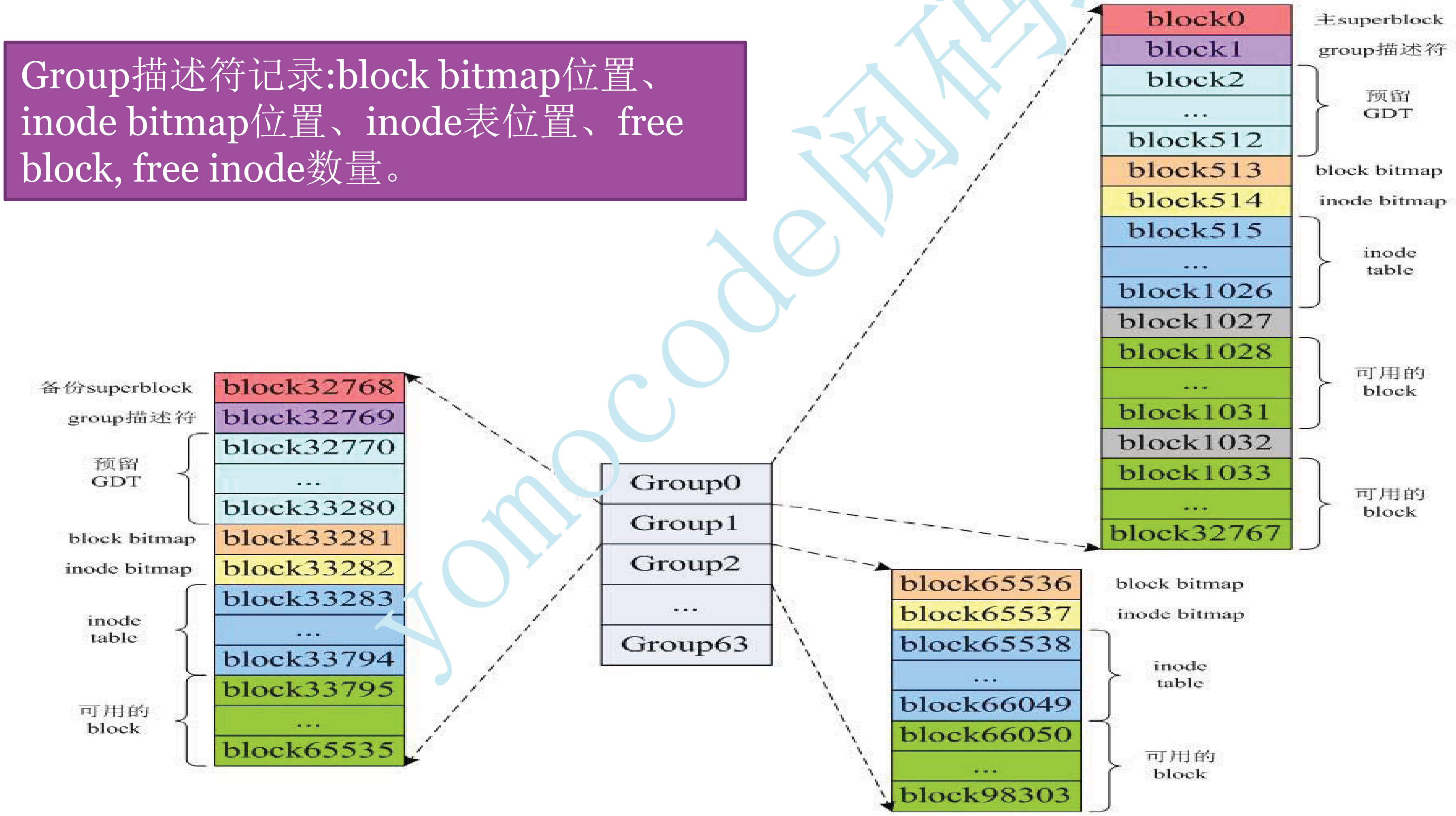
- Table of inodes
- Each inode is a file/directory
- Includes meta-data and lists of associated data blocks

Data blocks (4KB each)

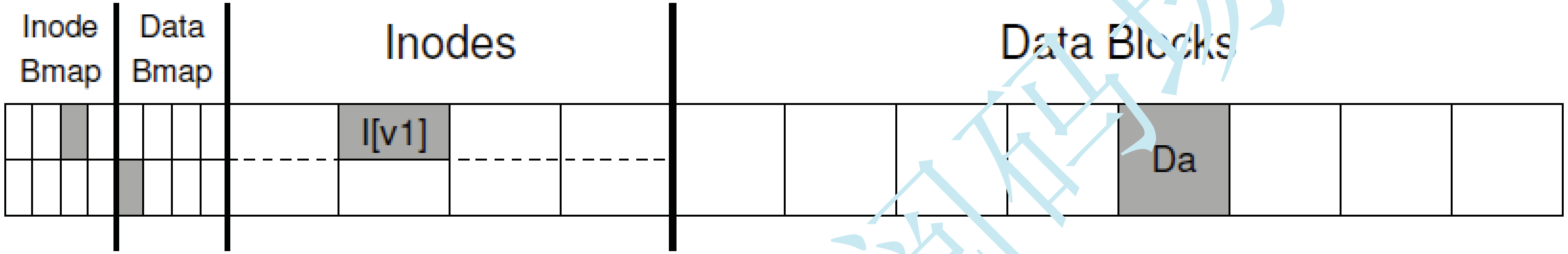
Group

superblock 记录文件系统的类型、block大小、block总数、free block数、inode大小、inode总数、free inode数、group的总数等,在多个group进行备份。

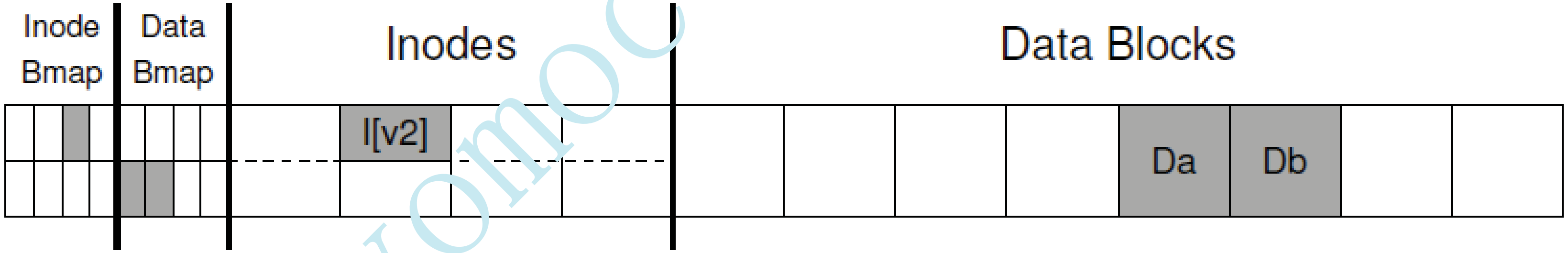
Group描述符记录:block bitmap位置、inode bitmap位置、inode表位置、free block, free inode数量。



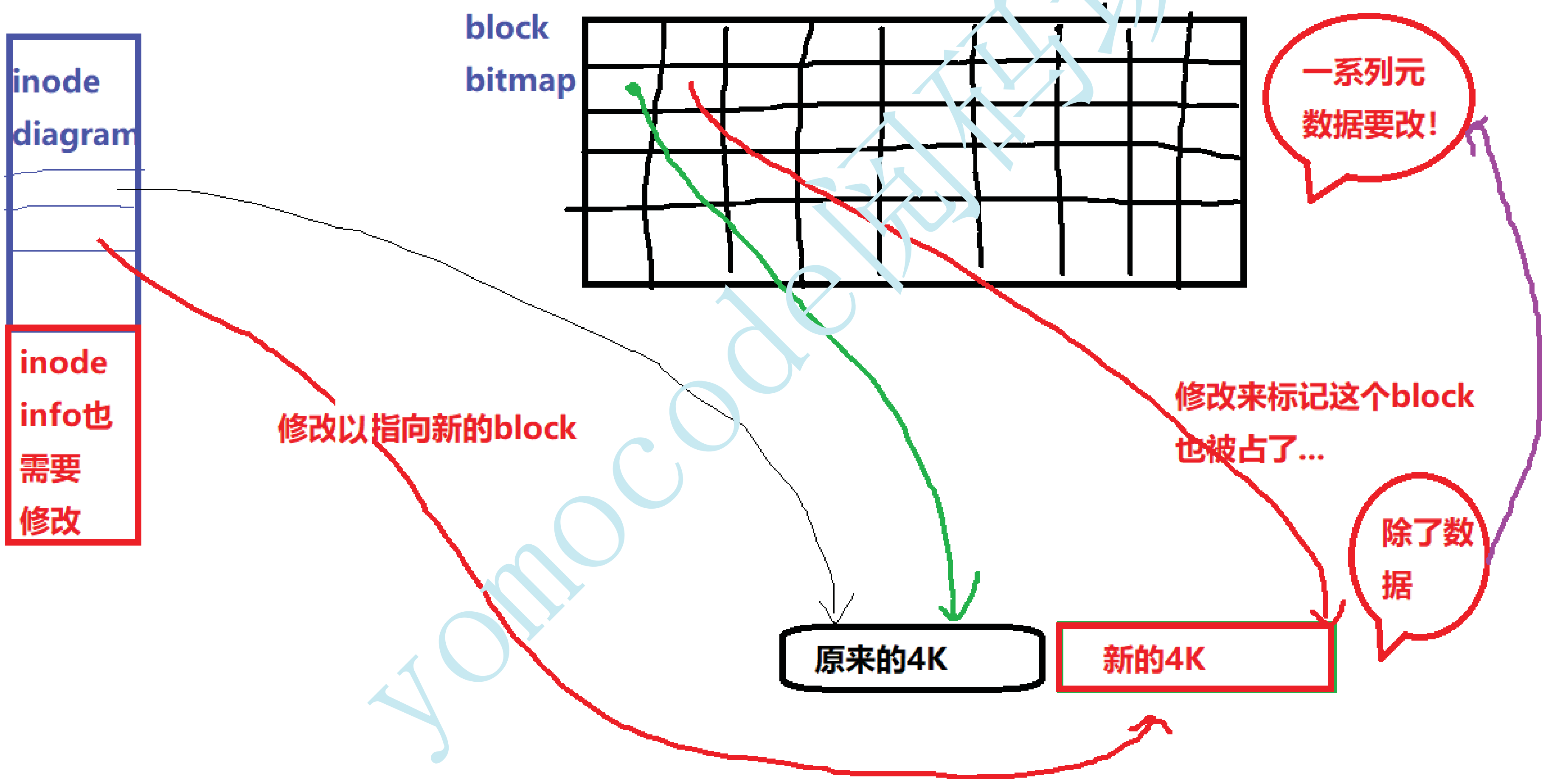
Append file一个文件



文件系统必须完成三件事：inode (I[v2]), bitmap (B[v2])和data block (Db).
crash 也许中途发生，并没有完成三件事件，文件系统在一个funny的状态。



硬件不可能原子执行


inode
diagram
inode
info也
需要
修改
block
bitmap
修改以指向新的block
一系列元
数据要改！
修改来标记这个block
也被占了...
除了数
据
原来的4K
新的4K

掉电与文件系统一致性

1. 任何的软件技术都无法保证掉电不丢数据！
只能保证一致性(元数据+数据的一致性，或者仅元数据的一致性)！

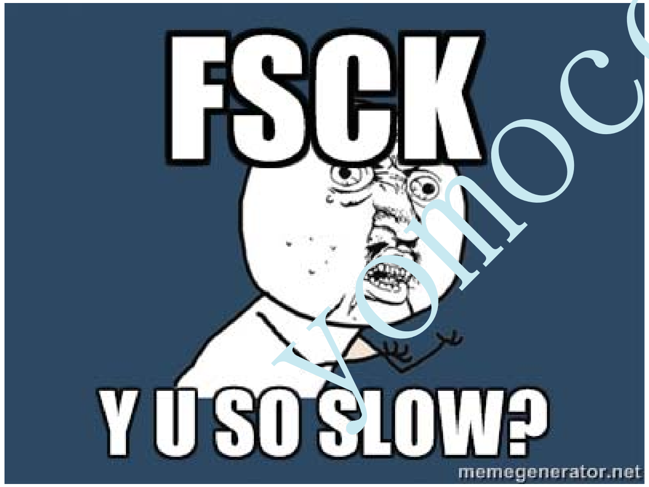
2. dirty_expire_centisecs、DIRECT_IO、SYNC IO的调整，不影响丢或者不丢数据，只影响丢多少数据。

3. fsck、日志、CoW文件系统等技术，帮忙提供一致性。

fsck (file system consistency check)

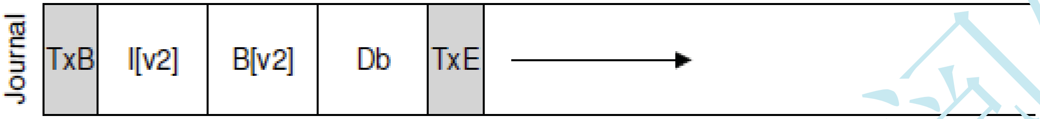
unclean shutdown后自动运行，或者手动运行；检查**superblock**、**inode**和**free block bitmap**、 所有**inode**的**Reachability**（比如删除**corrupted** 的**inode**）、验证目录的一致性；

Dennis Ritchie: “So fsck was originally called something else”
Question: “What was it called?”
Dennis Ritchie: "Well, the **second** letter was different"



日志：transaction

重启的过程中：transactions that are committed but not free are replayed in order



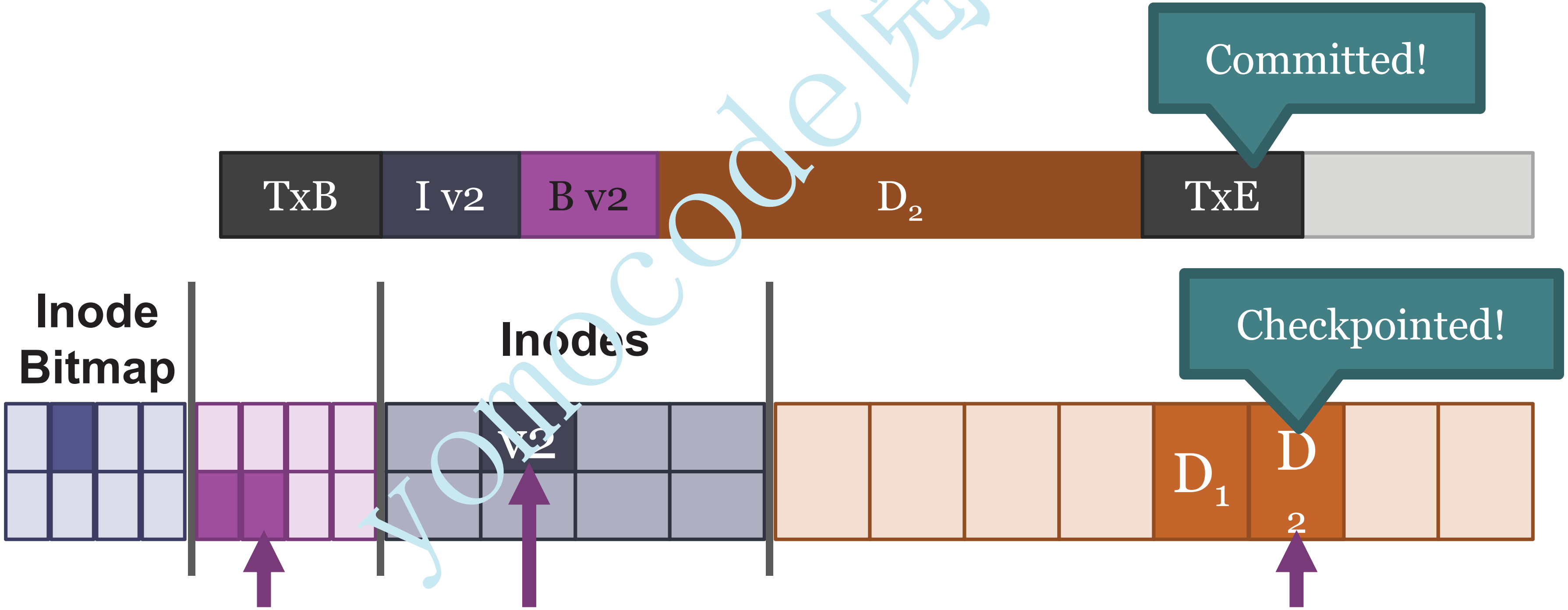
✓ The transaction begin (TxB) tells us about this update, including information about the pending update to the file system (e.g., the final addresses of the blocks I[v2], B[v2], and Db), as well as some kind of transaction identifier (TID).

✓ The middle three blocks just contain the exact contents of the blocks themselves;

✓ The final block (TxE) is amarker of the end of this transaction, and will also contain the TID.

Commits 和 Checkpoints

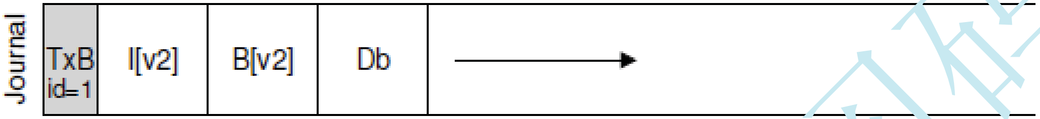
- 日志的写完成，transaction committed
- Transaction完成, OS checkpoint这次更新



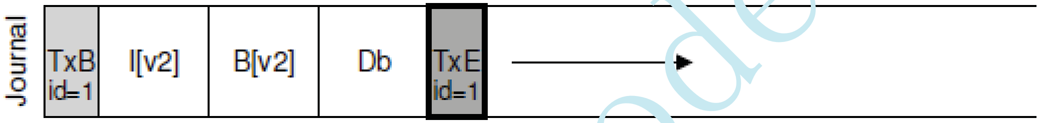
- Final step: free the checkpointed transaction

日志的4个阶段

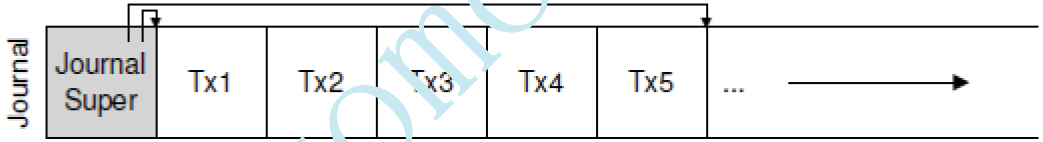
✓1. Journal write: Write the contents of the transaction (including TxB, metadata, and data) to the log; wait for these writes to complete.



✓2. Journal commit: Write the transaction commit block (containing TxE) to the log; wait for write to complete; transaction is said to be committed.



✓3. Checkpoint: Write the contents of the update (metadata and data) to their final on-disk locations.



✓ 4. Free: Some time later, mark the transaction free in the journal by updating the journal superblock.

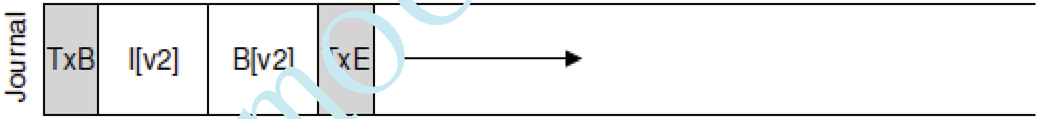
Metadata日志 - 5个阶段

完整的日志(data=journal)让系统变得很慢

✓1. Data write: Write data to final location; wait for completion (the wait is optional: data=writeback/ data=ordered).

✓2. Journal metadata write: Write the begin block and metadata to the log; wait for writes to complete.

✓3. Journal commit: Write the transaction commit block (containing TxE) to the log; wait for the write to complete; the transaction (including data) is now committed.



✓ 4. Checkpoint metadata: Write the contents of the metadata update to their final locations within the file system.

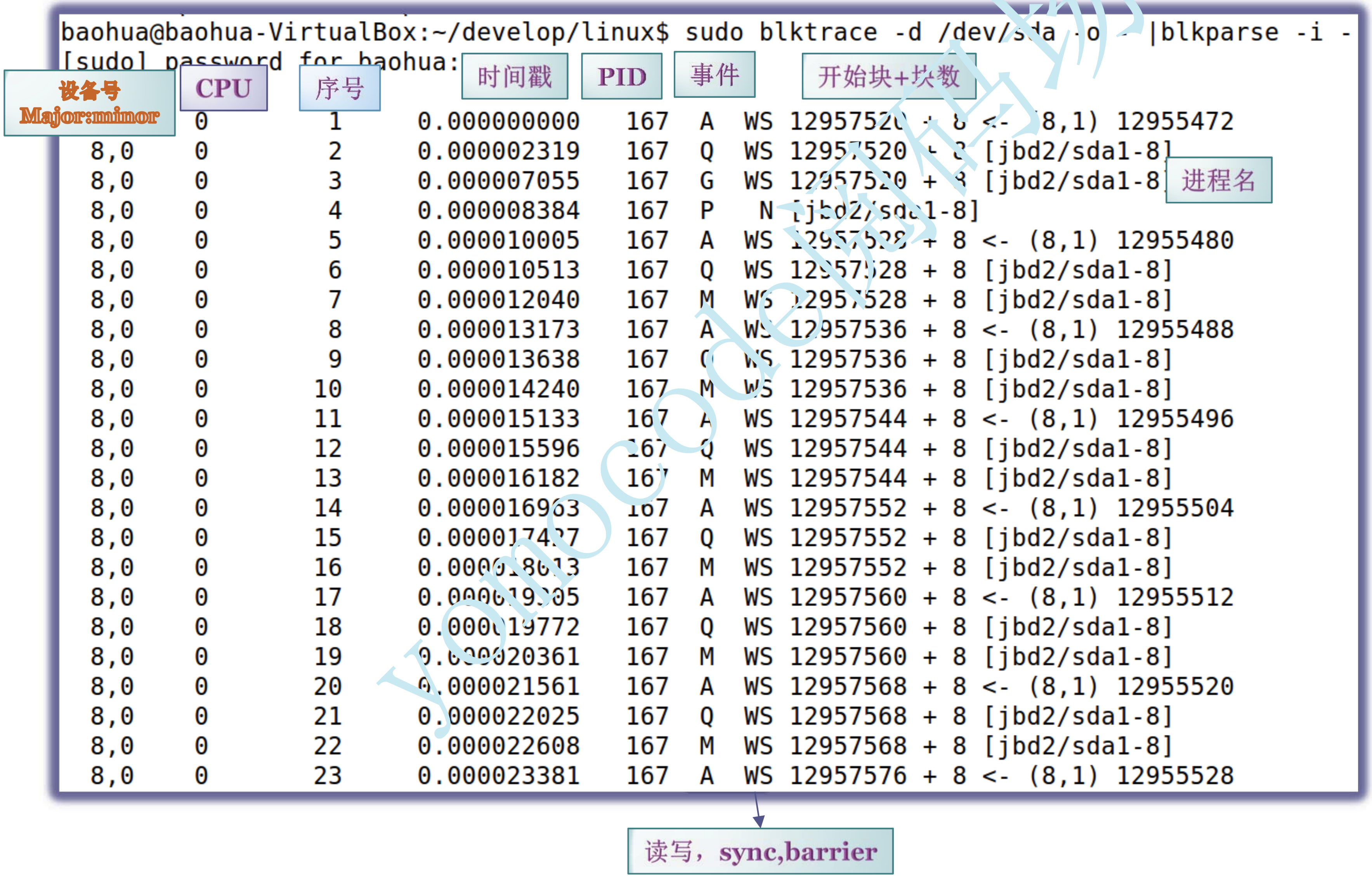
✓ 5. Free: Later, mark the transaction free in journal superblock.

常用工具

mkfs
dumpe2fs
blkcat
dd
debugfs:

- icheck: block号找inode
- ncheck:inode号找文件名

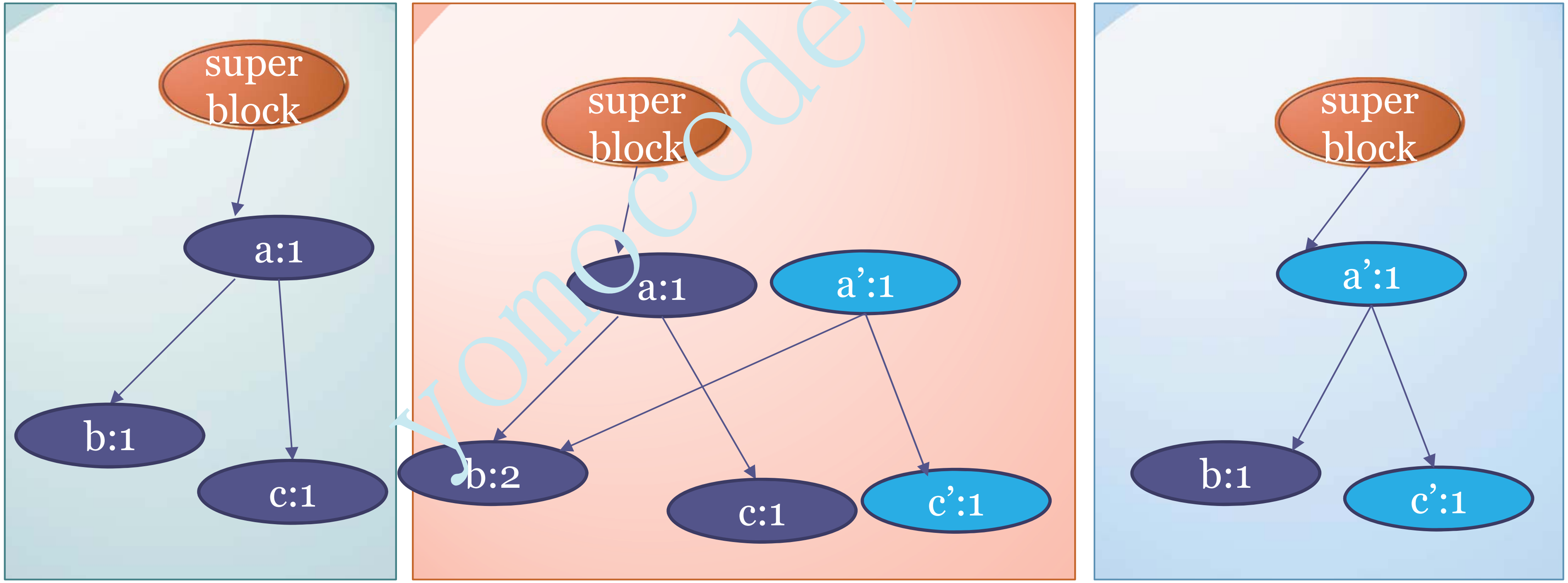
blktrace



```
baohua@baohua-VirtualBox:~/develop/linux$ sudo blktrace -d /dev/sda -o - |blkparse -i -
[sudo] password for baohua:
  8,0    0        1     0.000000000   167  A  WS 12957520 + 8 <- (8,1) 12955472
  8,0    0        2     0.000002319   167  Q  WS 12957520 + 8 [jbd2/sda1-8]
  8,0    0        3     0.000007055   167  G  WS 12957520 + 8 [jbd2/sda1-8]
  8,0    0        4     0.000008384   167  P   N [jbd2/sda1-8]
  8,0    0        5     0.000010005   167  A  WS 12957528 + 8 <- (8,1) 12955480
  8,0    0        6     0.000010513   167  Q  WS 12957528 + 8 [jbd2/sda1-8]
  8,0    0        7     0.000012040   167  M  WS 12957528 + 8 [jbd2/sda1-8]
  8,0    0        8     0.000013173   167  A  WS 12957536 + 8 <- (8,1) 12955488
  8,0    0        9     0.000013638   167  Q  WS 12957536 + 8 [jbd2/sda1-8]
  8,0    0       10     0.000014240   167  M  WS 12957536 + 8 [jbd2/sda1-8]
  8,0    0       11     0.000015133   167  A  WS 12957544 + 8 <- (8,1) 12955496
  8,0    0       12     0.000015596   167  Q  WS 12957544 + 8 [jbd2/sda1-8]
  8,0    0       13     0.000016182   167  M  WS 12957544 + 8 [jbd2/sda1-8]
  8,0    0       14     0.000016963   167  A  WS 12957552 + 8 <- (8,1) 12955504
  8,0    0       15     0.000017427   167  Q  WS 12957552 + 8 [jbd2/sda1-8]
  8,0    0       16     0.000018013   167  M  WS 12957552 + 8 [jbd2/sda1-8]
  8,0    0       17     0.000019305   167  A  WS 12957560 + 8 <- (8,1) 12955512
  8,0    0       18     0.000019772   167  Q  WS 12957560 + 8 [jbd2/sda1-8]
  8,0    0       19     0.000020361   167  M  WS 12957560 + 8 [jbd2/sda1-8]
  8,0    0       20     0.000021561   167  A  WS 12957568 + 8 <- (8,1) 12955520
  8,0    0       21     0.000022025   167  Q  WS 12957568 + 8 [jbd2/sda1-8]
  8,0    0       22     0.000022608   167  M  WS 12957568 + 8 [jbd2/sda1-8]
  8,0    0       23     0.000023381   167  A  WS 12957576 + 8 <- (8,1) 12955528
```

COW文件系统:btrfs

每次写磁盘时，先将更新数据写入一个新的 block，当新数据写入成功之后，再更新相关的数据结构指向新 block 。
没有日志，用COW实现文件系统一致性。



Transactions in Btrfs

- There is no journal as extN, or xfs has
- COW is used to guarantee consistency
- On fs tree or extent tree modification
  1. Tree is cloned and the branch in question is copied and modified
  2. New roots are added into root tree (identified by transaction ID)
  3. New root of root tree added into superblock
  4. Wait on all respective data and metadata to hit the disk
  5. Commit the superblock to the disk
  6. Original trees can be freed (decrease refcount)
- In case of crash, the original root tree is used (the one in the most up-to-date superblock)

Btrfs: subvolume和snapshot

subvolume可以单独挂载

```
#btrfs subvolume create sub1
#btrfs subvolume list .
#mount -o subvolid=256 image aaa
```

snapshot类似git branch，也具备subvolume特征

```
#btrfs subvolume snapshot snapshot1
#btrfs subvolume list .
#mount -o subvolid=257 image aaa
#btrfs subvolume set-default 257 snapshot1
```

谢 谢！