File System with logging

XV6 LOG

- 文件系统中存在log header块和log块
- header块中记录需要改变的块的编号；log块记录操作信息和文件meta data
- **最多只能存在一个事务；保证下面操作的原子性**
  - 写日志
  - 写数据
  - 提交数据
  - 清除日志

过程

保证原子性

- 一系列对文件系统的操作后，调用begin_op和end_op；此时对数据的改变都在buffer cache中
- end_op后，将数据块复制到日志块；所有日志块复制完成后，将编号写入Log header块（commit point）
- 写入后，依据log header块将改变的内容**写入文件系统对应的位置**

崩溃

- 系统崩溃后，恢复软件查看log header块；如果有编号，就将内容写入原位置

WAL

- 预写日志：Write-Ahead-Log
- **将所有的更改先写入日志，日志提交后；依据日志对文件进行修改**

Free Rule

- 要求一个事务的日志全部写入；日志不能重用或覆盖
- 完成一个事务；**必须将log header块中的块编号全部擦除**

EXT3

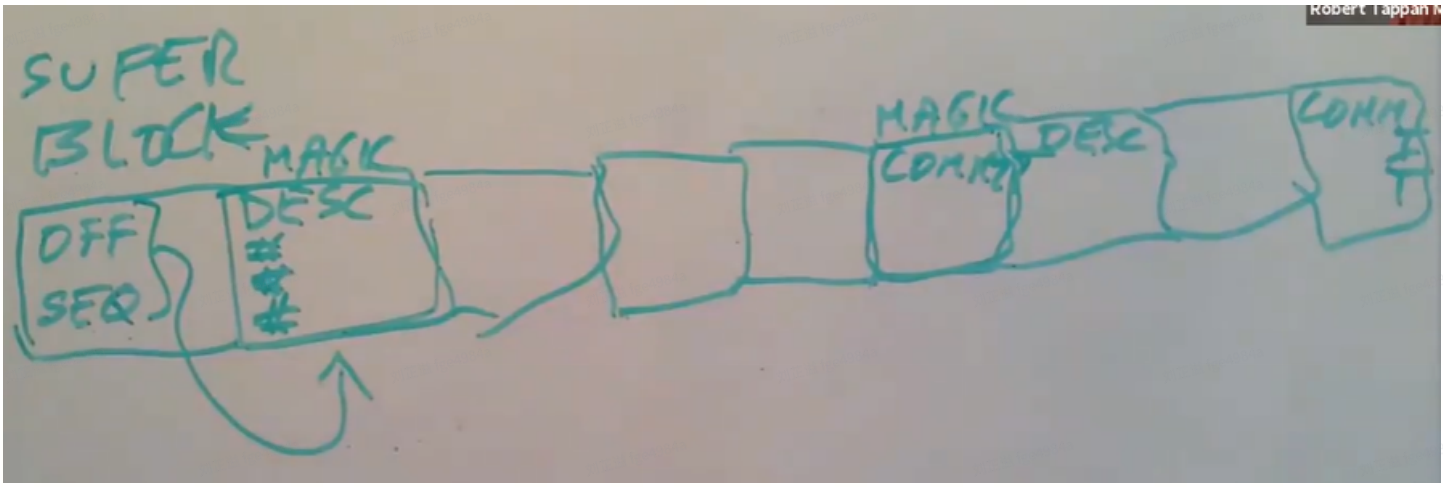
- ext2 + jourual (log)

MEM

- 主存
- 块缓存：存放修改的副本；以及需要回写的块
- 跟踪的事务meta data（可以同时跟踪多个事务）
  - 事务序列号
  - 修改的块号
  - 同步处理的函数handles

Log Format

- **磁盘上存放完成的事务，只有内存中存放一个执行的事务**
- Super block
  - 第一个有效事务的偏移量
  - 第一个有效事务的事务序列号
- 每个事务有事务描述块，存放块编号（类似header）
- 修改的块
- 提交块（后面可能其他事务的块）
- **使用MAGIC NUM来区分super block和commit block**



提高性能方式

ASYNC SYS CALLS

- 异步系统调用
- IO Concurrency + Batching
- 但是崩溃时数据并没有写入磁盘

fsync(fd)：将所有的输出写入磁盘

BATCHING

- 总有一个打开的事务；一个事务提交一次（接近5s的系统调用）
- 摊销系统调用成本
- 减少写入块的数量（多次写同一个块）
- Disk scheduling（磁盘大规模写入）

CONCURRENCY

- Syscalls in parallel
- Many old transactions
  - One open transaction: **copy all blocks that it will be modified**
  - Commiting to log transactions
  - Writing to home transactions
  - Freed transactions

EXT3 CODE

- 抽象层次

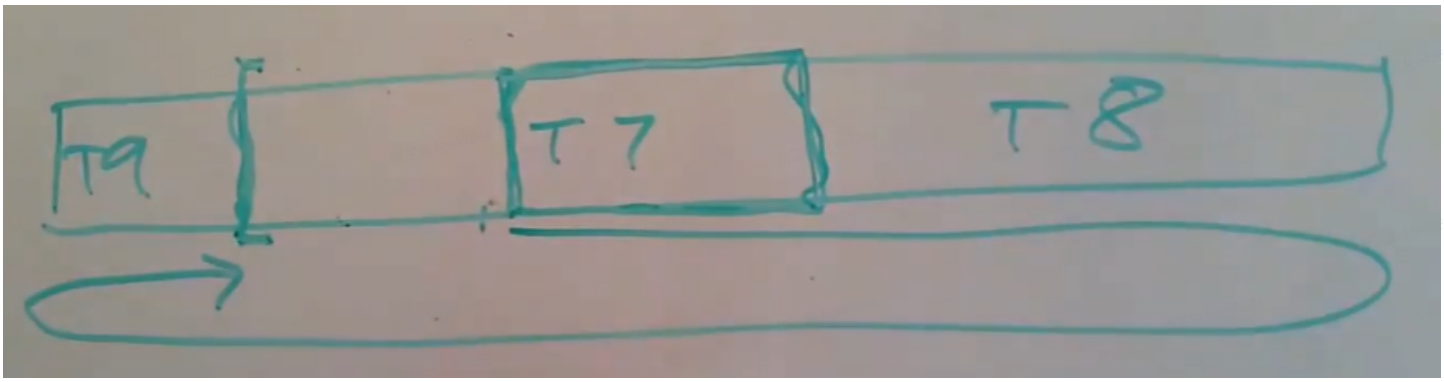
```
1   sys_unlink()
2   handle = start()        // 事务的开始
3   get(handle, blocks)     // 获取事务处理的块，将事务与块编号映射起来
4   modified blocks in cache
5   stop(handle)
```

提交事务

后台线程运行这些操作

- Block new system calls
- Wait for outstanding system calls
- Open new transaction
- Write described block and data blocks
- Write blocks to log (实际写入的是副本)
- Wait for writing blocks
- Write commit block

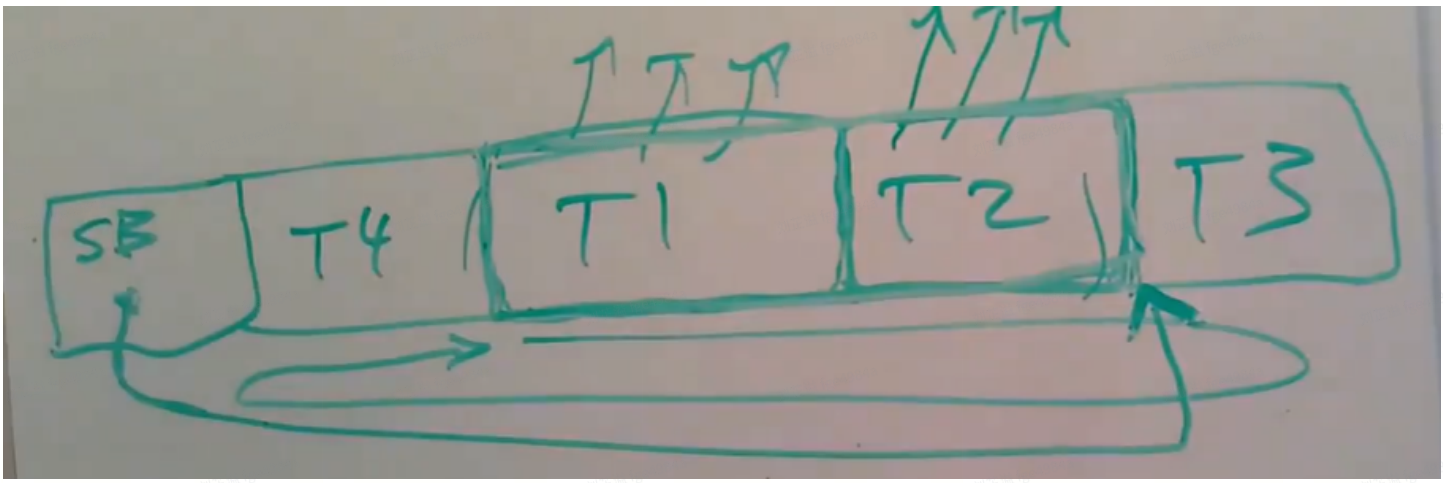
- Wait for commit write: **This is commit point**
- Write to home locations
- Re-use log



- **如果日志没有足够空间，新的事务必须等待其他事务commit到磁盘；来释放足够的空间容纳新的事务信息写入日志系统**

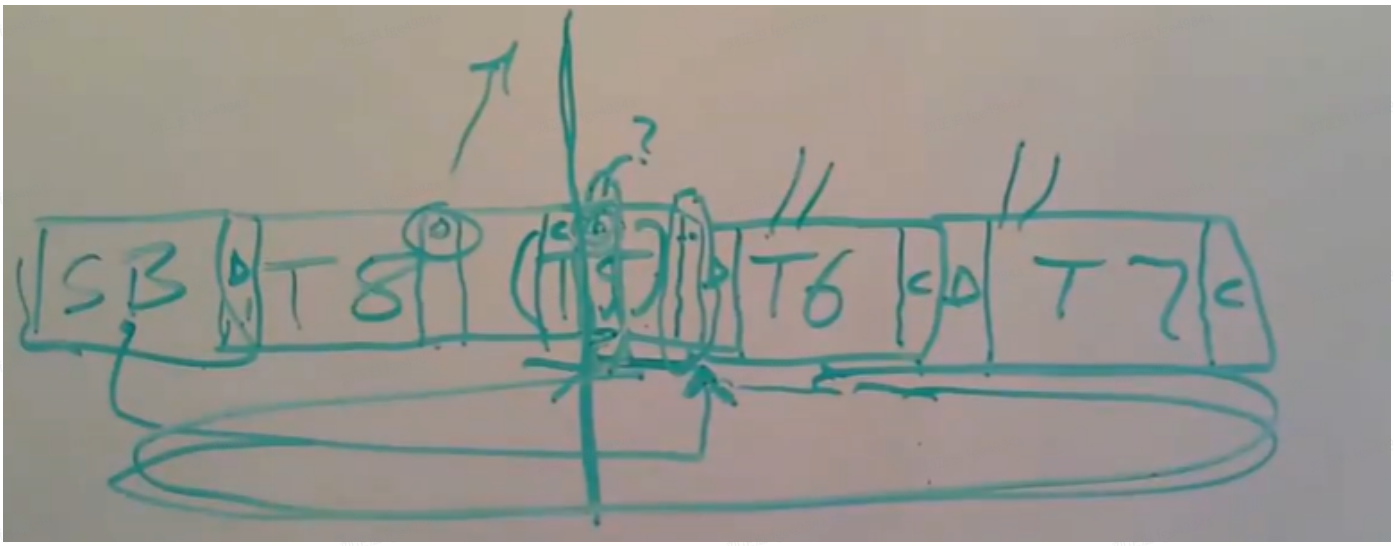
事务崩溃

- 文件系统选择重用或者释放某些块（事务占据的）
- 重写super block，将最新事务更新到T3

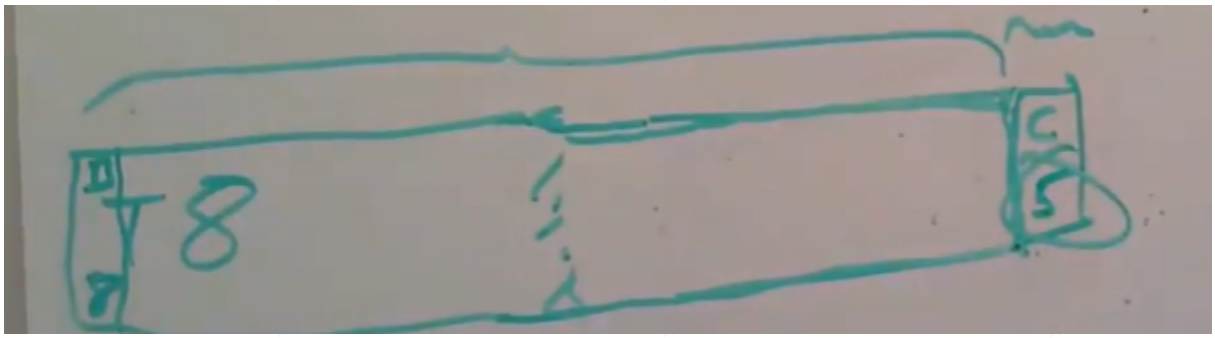


恢复日志

- 从第一个有效事务开始对所有空间进行扫描
- 判断事务的起止位置：第一个块是描述块，最后一个块是提交块
  - 判断Magic number
  - 区分数据块magic number：**将数据块的magic number替换成0；记录在describe block中，如果恢复块的话，将其中的0替换为magic number**
- 部分提交的事务恢复时放弃；
- 重写日志中已经提交的块



- 如果T8并没有提交，而T5仍然有自己的提交块
  - **恢复软件需要查看此时describe块的seq**

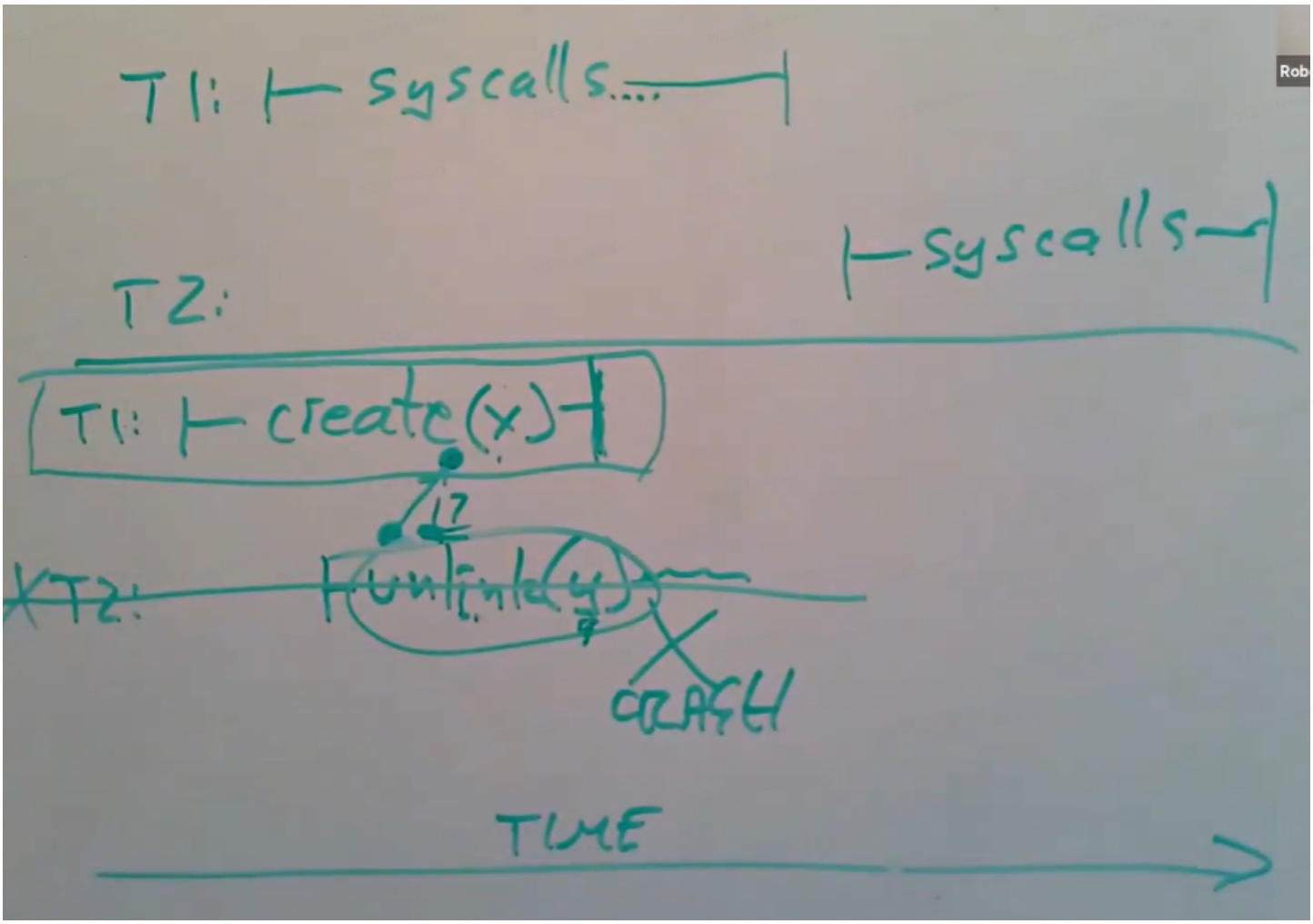


细节

直到某一事务执行完当前syscall后，其他事务才允许执行其他syscall

内存中的打开事务，缓存改变的块

- 如果create和unlink在两个事务中执行；unlink释放inode后，create获取这个inode；两个syscall使用相同的inode，create提交，unlink未提交；如果系统crash，恢复后**相当于两个文件在同一个inode上**
- 系统调用失去其原子性



EXT4解决方案

- 同时写入数据块和commit block
- **在commit block中加入校验和**；恢复时用于判断该事务是否需要重放