中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

1924-2024 中山大學 世纪华诞 100th ANNIVERSARY SUN YAT-SEN UNIVERSITY

文件系统崩溃—致性II

SSE202/204: 操作系统原理

苏玉鑫

suyx35@mail.sysu.edu.cn

助教：龙玉丹 单诗雯 毛晨希 沈志轩 郑灿峰 胡伟峰

版权信息

- 部分内容来自：上海交通大学并行与分布式系统研究所操作系统课件
  - https://ipads.se.sjtu.edu.cn/courses/os/
- 其它参考资料：
  - 清华大学操作系统公开课
    - https://open.163.com/newview/movie/courseintro?newurl=ME1NSA351
    - 介绍标准内容，适合考研
  - 南京大学计算机软件研究所
    - http://jyywiki.cn/OS/2025/
    - https://space.bilibili.com/202224425/channel/collectiondetail?sid=192498
    - 比较有趣

单选题 1分

设置

SUN YAT-SEN UNIVERSITY

以下哪种操作可以确保文件系统缓存中的数据被写入磁盘，以保证数据的持久性?

close 函数关闭文件

使用 lseek 调整读写位置

调用 fsync 函数

调用 mmap 函数进行内存映射

提交

单选题 1分

设置

SUN YAT-SEN UNIVERSITY

在文件系统中，为了保证崩溃一致性，以下哪种操作是必要的?

定期对文件进行备份

在每次文件操作后立即将数据写入磁盘

采用事务机制，确保一组相关操作要么全部完成，要么全部不完成

只允许读操作，禁止写操作

提交

单选题 1分

设置

SUN YAT-SEN UNIVERSITY

文件系统崩溃一致性要求维护文件系统数据结构的内部不变量，以下哪种情况违反了这一要求？

数据块只在一个文件中

磁盘块既处于空闲状态又在一个文件中

仅有最近的一些操作没有被保存到磁盘中

没有顺序的异常

提交

单选题 1分

设置

SUN YAT-SEN UNIVERSITY

Linux 中使用 JBD2 实现的日志模式中，只记录元数据且一致性最差，但速度最快的是哪种模式?

写回 (writeback) 模式

日志 (journal) 模式

顺序 (ordered) 模式

以上都不是

提交

大纲

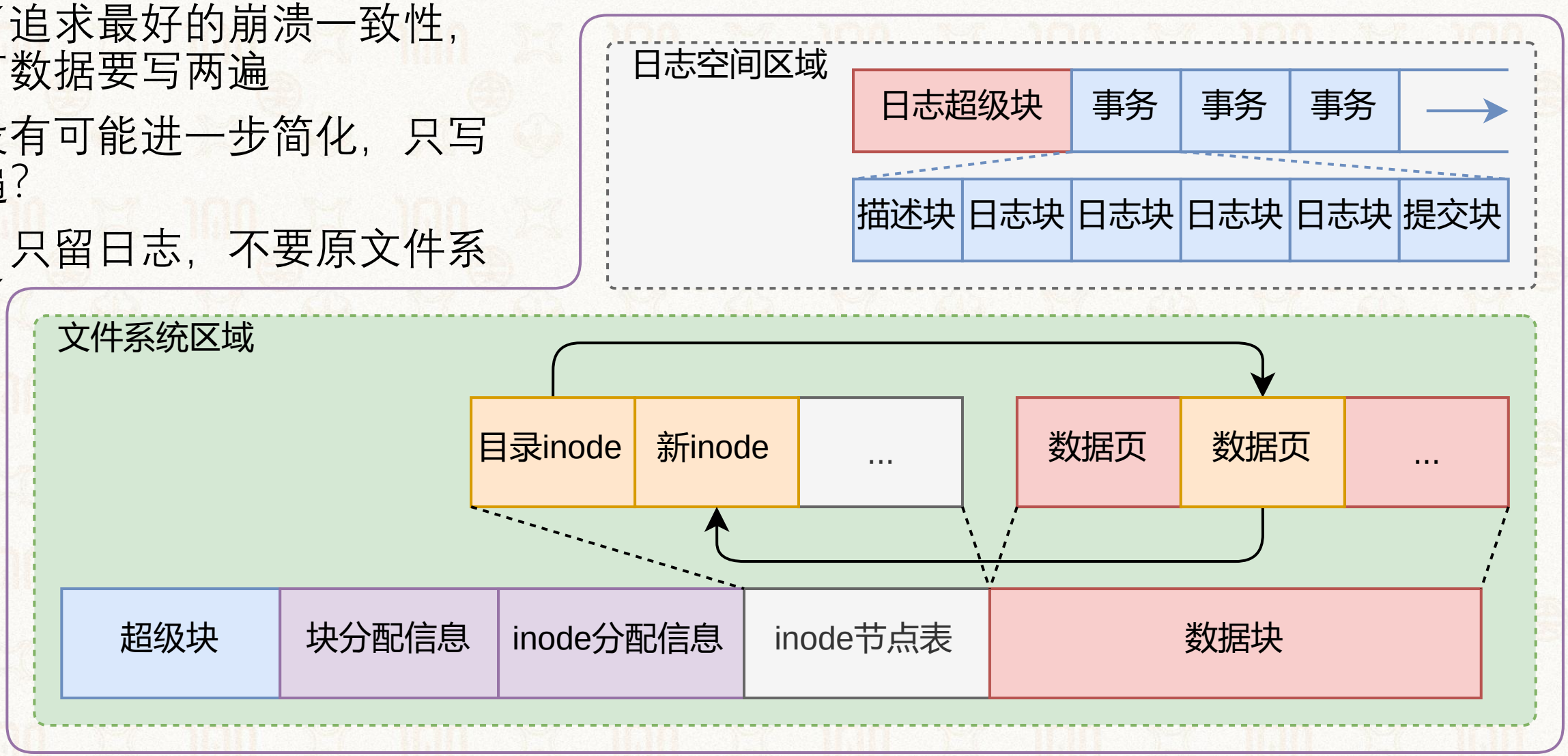
- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

日志文件系统

- 为了追求最好的崩溃一致性，所有数据要写两遍
- 有没有可能进一步简化，只写一遍？
- 有，只留日志，不要原文件系统了



日志文件系统（Log-structured FS）

The Design and Implementation of a Log-Structured File System

Mendel Rosenblum and John K. Ousterhout

Electrical Engineering and Computer Sciences, Computer Science Division
University of California
Berkeley, CA 94720
mendel@sprite.berkeley.edu, ouster@sprite.berkeley.edu

Abstract

This paper presents a new technique for disk storage management called a *log-structured file system*. A log-structured file system writes all modifications to disk sequentially in a log-like structure, thereby speeding up both file writing and crash recovery. The log is the only structure on disk; it contains indexing information so that files can be read back from the log efficiently. In order to maintain large free areas on disk for fast writing, we divide the log into *segments* and use a *segment cleaner* to compress the live information from heavily fragmented segments. We present a series of simulations that demonstrate the efficiency of a simple cleaning policy based on cost and benefit. We have implemented a prototype log-

magnitude more efficiently than current file systems.

Log-structured file systems are based on the assumption that files are cached in main memory and that increasing memory sizes will make the caches more and more effective at satisfying read requests[1]. As a result, disk traffic will become dominated by writes. A log-structured file system writes all new information to disk in a sequential structure called the *log*. This approach increases write performance dramatically by eliminating almost all seeks. The sequential nature of the log also permits much faster crash recovery: current Unix file systems typically must scan the entire disk to restore consistency after a crash, but a log-structured file system need only examine the most recent portion of the log.

1992 *ACM Transactions on Computer Systems*

John K. Ousterhout

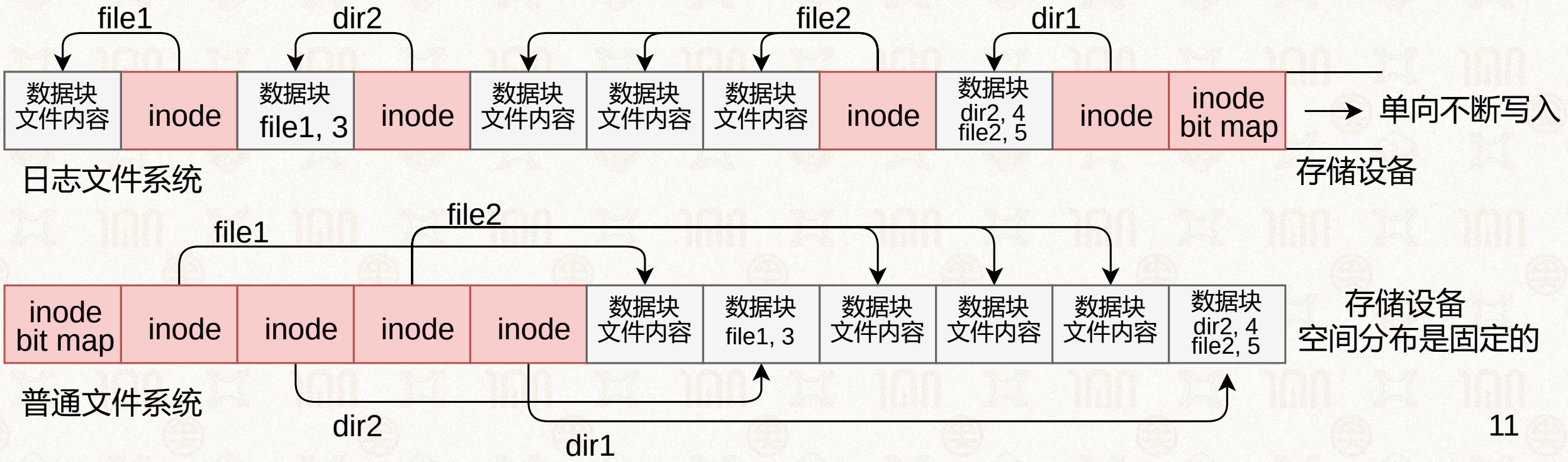
Sprite 负责人
TCL/TK 作者
Magic VLSI CAD 作者
Co-scheduling 提出者
Raft 一致性协议
目前是Stanford大学教授

Mendel Rosenblum

VMware 联合创始人
目前是Stanford大学教授

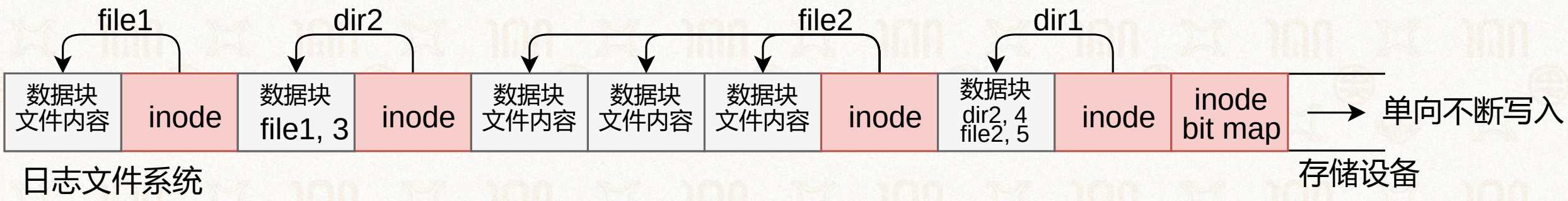
日志文件系统（Log-structured FS）

- 假设：文件被缓存在内存中，文件读请求可以被很好的处理
  - 于是，文件写成为瓶颈
- 块存储设备的顺序写比随机写速度很块
  - 磁盘寻道时间
- 将文件系统的修改以日志的方式顺序写入存储设备



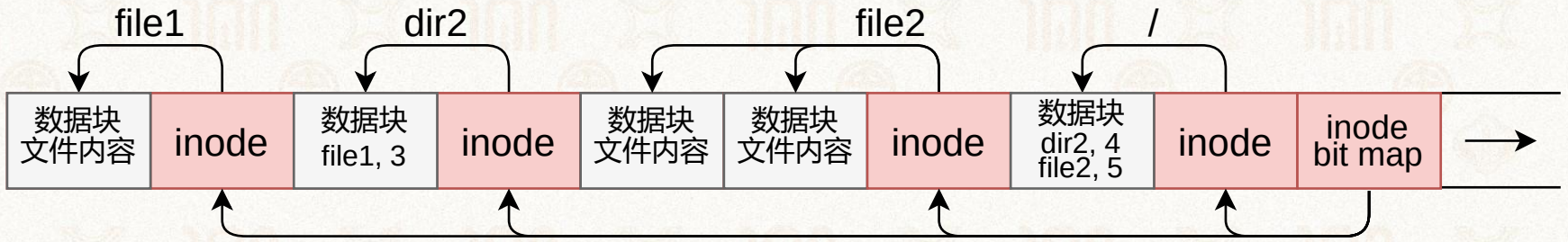
Sprite LFS的数据结构

- 固定位置的结构
  - 超级块、检查点（checkpoint）区域
- 以日志形式保存的结构
  - inode、间接块（索引块）、数据块
  - inode map：记录每个inode的当前位置
  - 段概要：记录段中有效块
  - 段使用表：段中有效字节数、段的最后修改时间
  - 目录修改日志



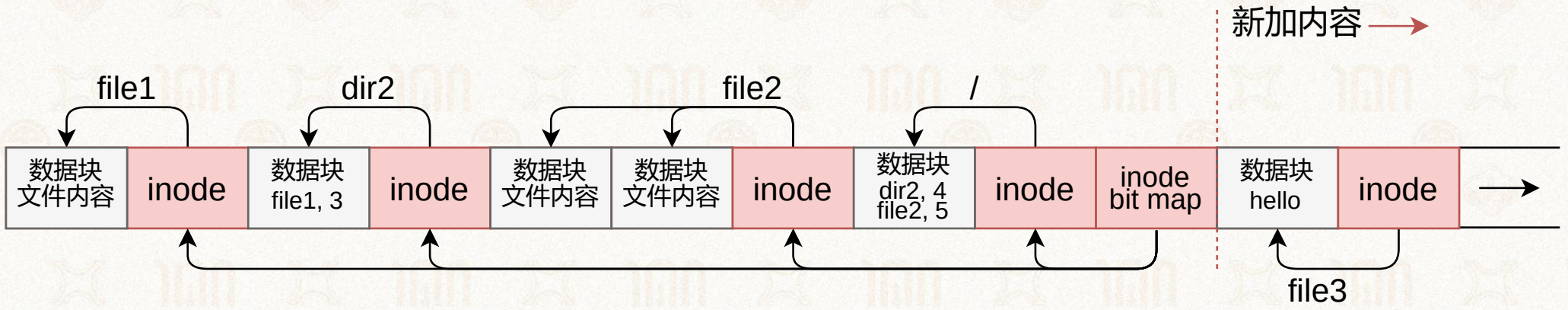
举例

- 一个日志文件系统
  - 有4个inode，位置记录在inode map中
  - 对应4个文件分别为：/，/dir2，/file2，/dir2/file1



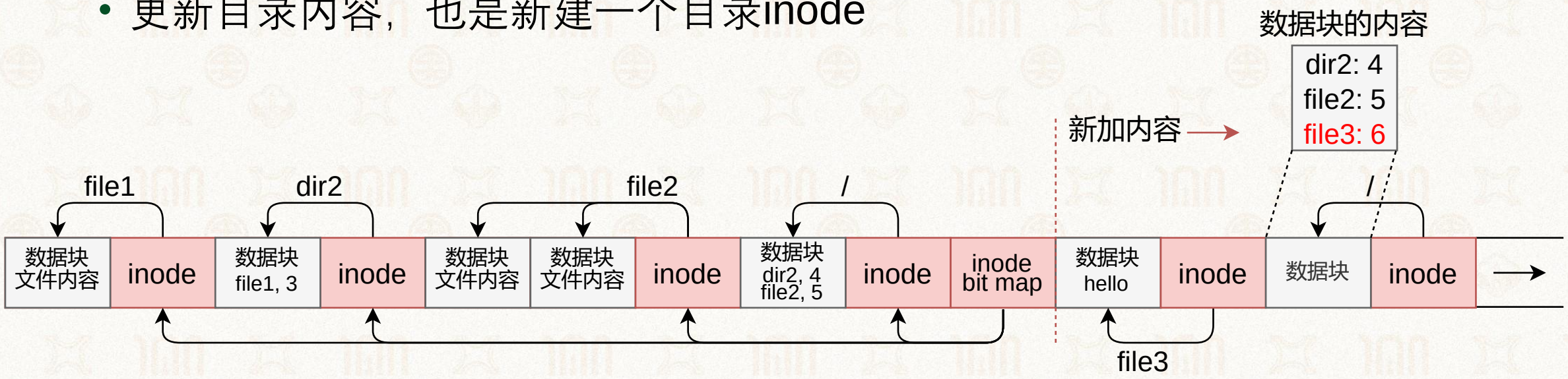
创建文件

- 创建一个文件：/file3
- 修改文件内容为：hello
- 日志系统只能以追加(append)模式向后写
  - 创建新inode，创建新数据块，写内容到数据块中



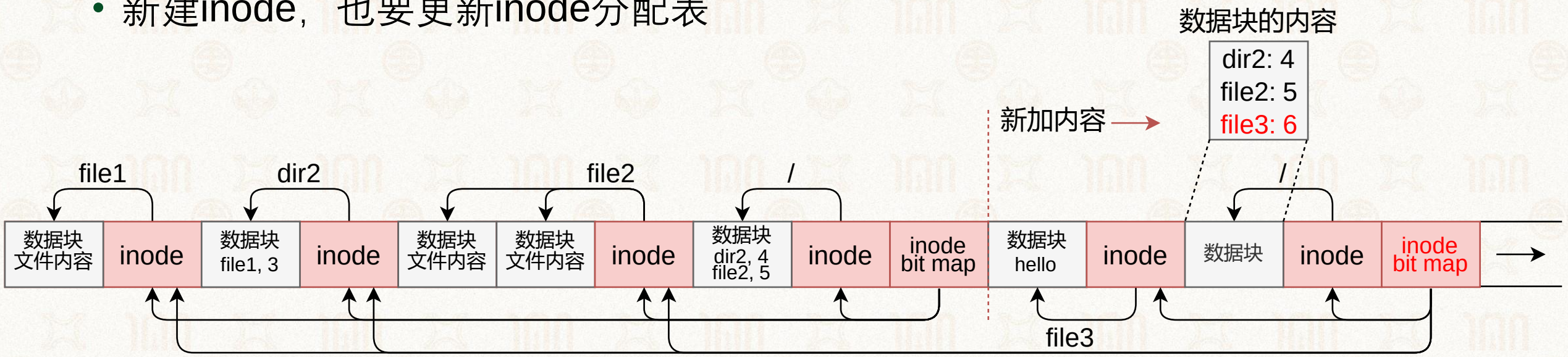
创建文件

- 创建一个文件：/file3
- 修改文件内容为：hello
- 日志系统只能以追加(append)模式向后写
  - 更新目录内容，也是新建一个目录inode



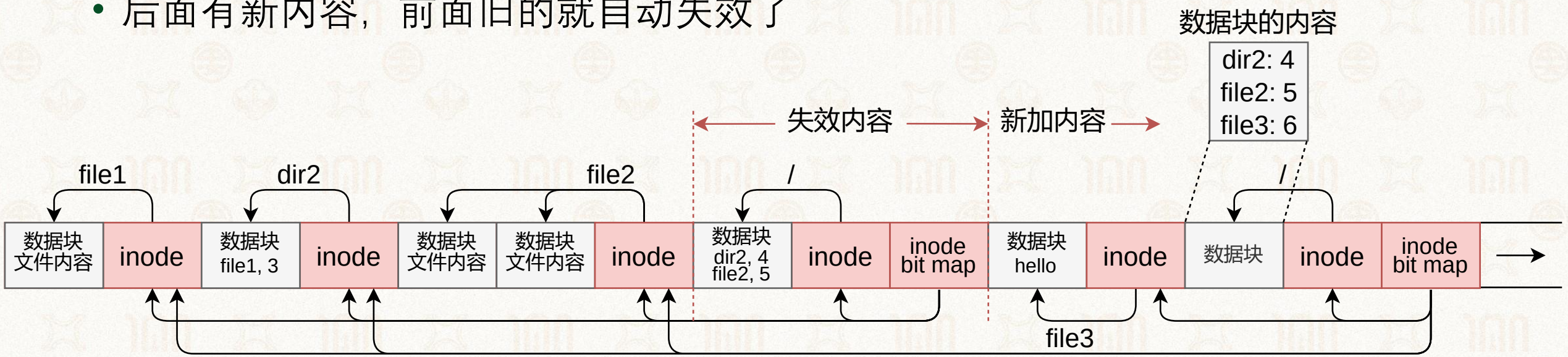
创建文件

- 创建一个文件：/file3
- 修改文件内容为：hello
- 日志系统只能以追加(append)模式向后写
  - 新建inode，也要更新inode分配表



创建文件

- 创建一个文件：/file3
- 修改文件内容为：hello
- 日志系统只能以追加(append)模式向后写
  - 后面有新内容，前面旧的就自动失效了

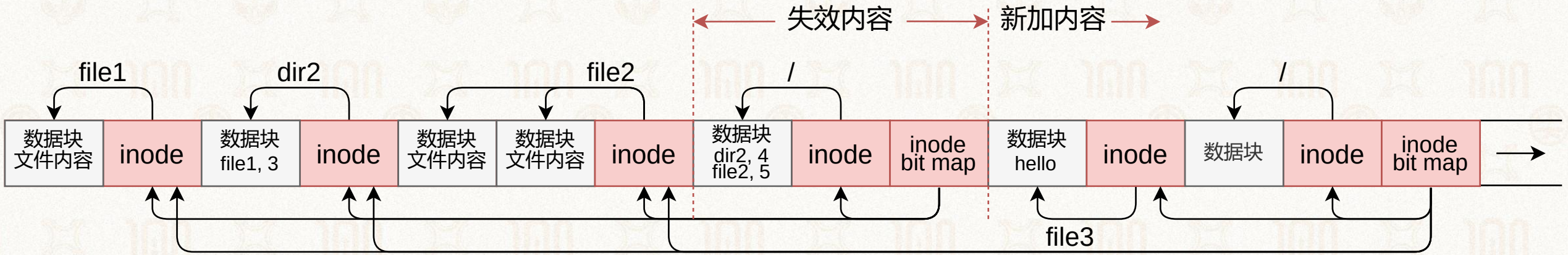


大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

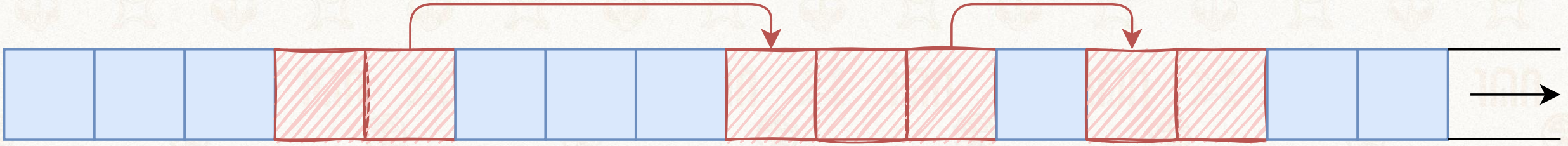
空间回收利用

- 存储设备最开始是一个连续的空闲空间
- 随文件系统的使用，日志写入位置会接近存储设备末端
- 需重新利用前面的设备空间
  - 文件系统数据被修改后，此前的块被无效化
  - 如何组织前面无效空间，以重新利用它们？

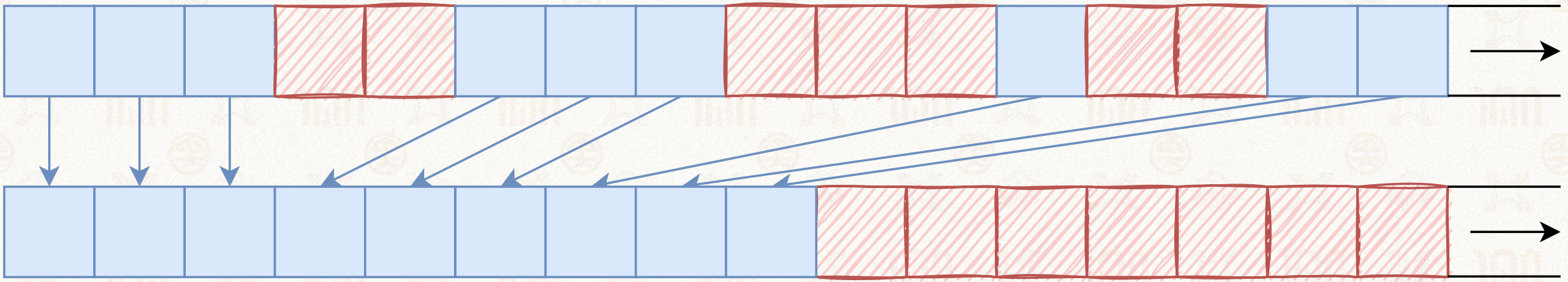


空间回收管理方法

- 串联：将所有空闲空间用链表串起来
  - 磁盘空间会越来越碎，影响到日志文件系统大块顺序写的性能



- 拷贝：将所有的有效空间整理拷贝到新的存储设备
  - 数据需要拷贝



空间回收管理方法：段(Segment)

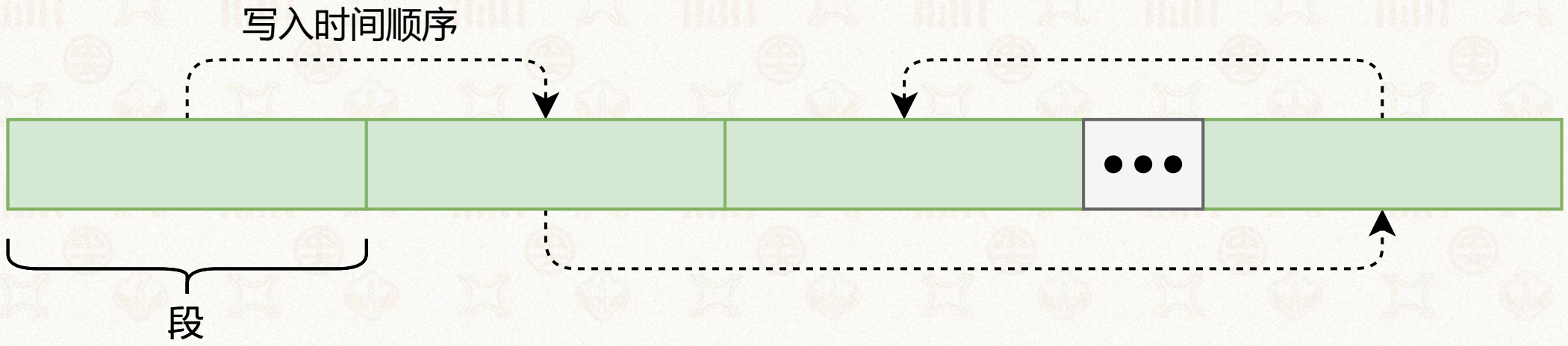
- 串联
  - 磁盘空间会越来越碎，影响到日志文件系统大块顺序写的性能
- 拷贝
  - 数据需要拷贝
- 串联和拷贝两种方法的结合：段
  - 系统领域的经典思维：在两个极端之间“和稀泥”

空间回收管理方法：段(Segment)

- 一个设备被拆分为定长的区域，称为段
  - 段大小需要足以发挥出顺序写的优势，512KB、1MB等
- 每段内只能顺序写入
  - 只有当段内全都是无效数据之后，才能被重新使用
- 干净段用链表维护（对应串联方法）

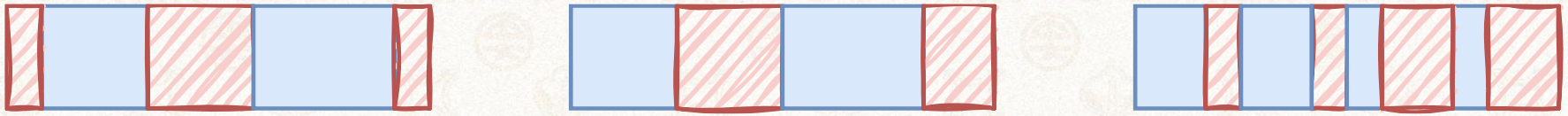
段使用表

- 记录每个段中有效字节数
  - 归零时变为干净段
- 记录了每个段最近写入时间
  - 将非干净段按时间顺序连在一起，形成逻辑上的连续空间



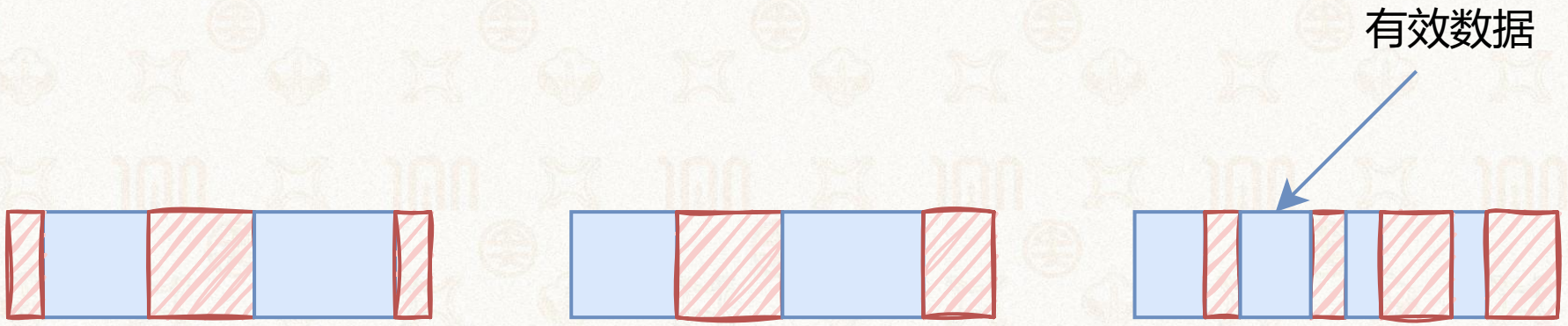
段清理

- 1. 将一些段读入内存中准备清理



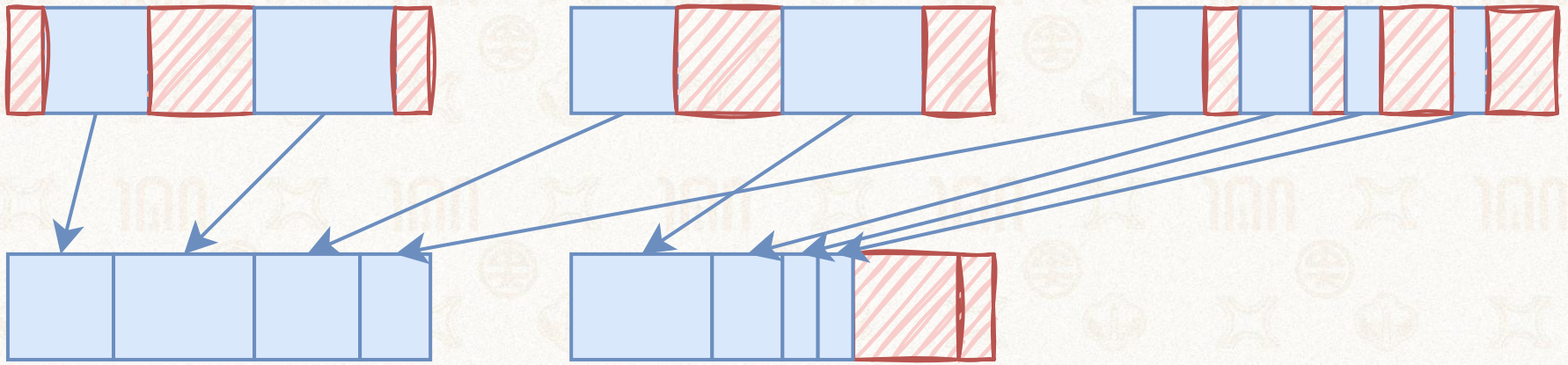
段清理

- 1. 将一些段读入内存中准备清理
- 2. 识别出有效数据



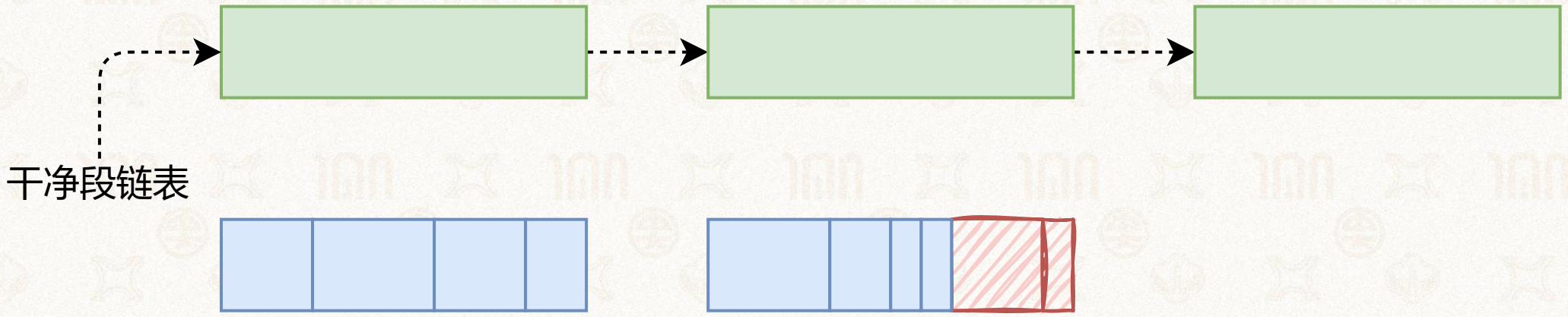
段清理

- 1. 将一些段读入内存中准备清理
- 2. 识别出有效数据
- 3. 将有效数据整理后写入到干净段中（对应拷贝方法）



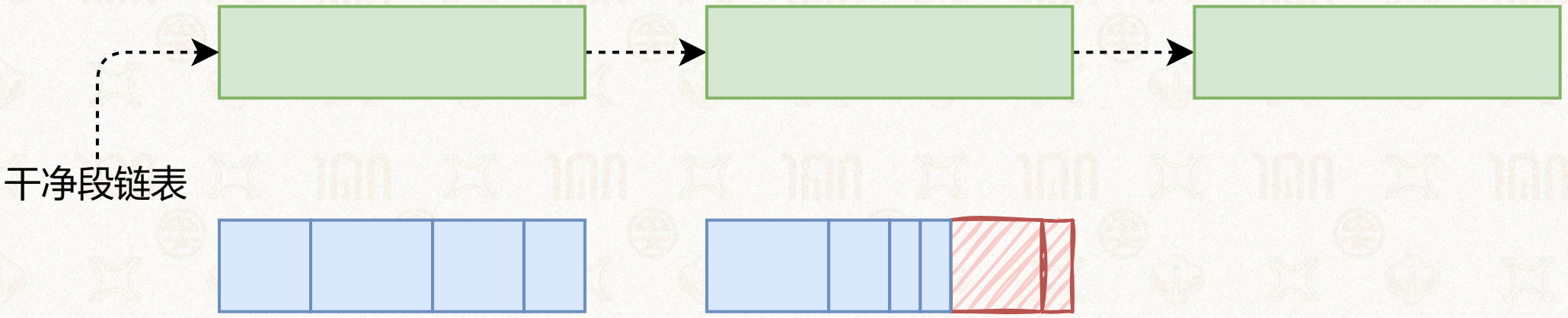
段清理

- 1. 将一些段读入内存中准备清理
- 2. 识别出有效数据
- 3. 将有效数据整理后写入到干净段中（对应拷贝方法）
- 4. 标记被清理的段为干净



段清理

- 1. 将一些段读入内存中准备清理
- 2. 识别出有效数据
- 3. 将有效数据整理后写入到干净段中（对应拷贝方法）
- 4. 标记被清理的段为干净

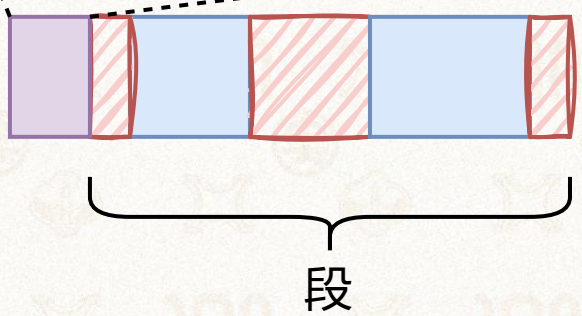


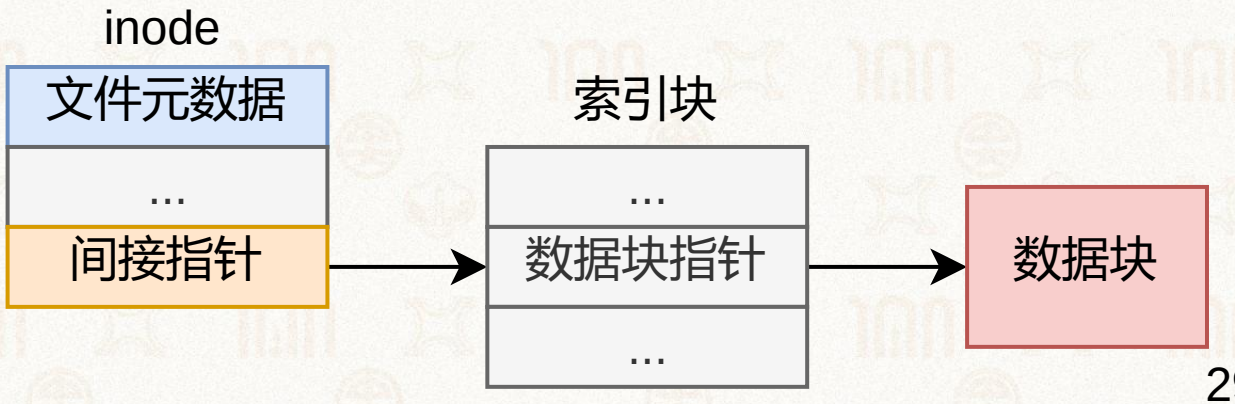
识别有效数据

- 每个段中保存有段概要
  - 每个块被哪个文件的哪个位置所使用
    - 如：数据块可使用inode号和第几个数据块来表示位置

段概要：

第1块	第2块	第3块	•••	第N块
类型：数据块 inode号：5 数据块序号：13	类型：索引块 inode号：5 数据块序号：1	类型：索引块 inode号：n/a 数据块序号：n/a		类型：数据块 inode号：8 数据块序号：2





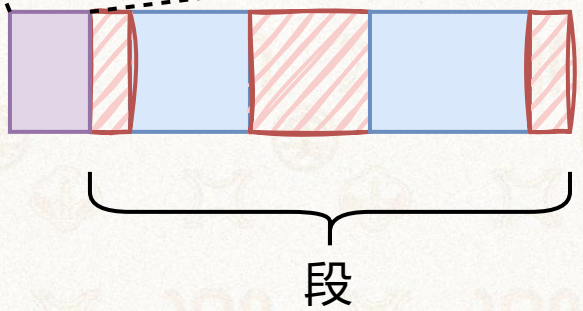
识别有效数据

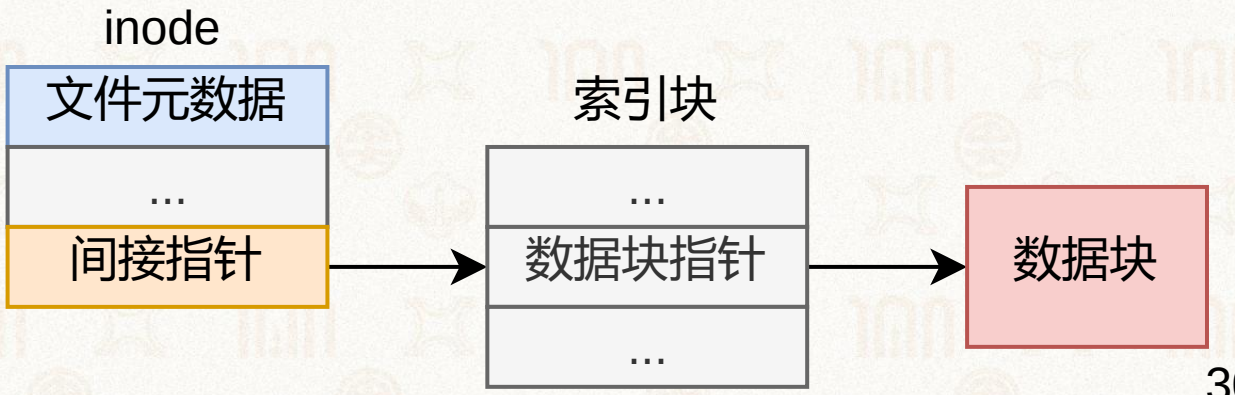
- 每个段中保存有段概要
  - 每个块被哪个文件的哪个位置所使用
    - 如：数据块可使用inode号和第几个数据块来表示位置
  - 块有效性通过比对该位置上现有指针判断

8号inode中第2个数据块不指向此位置=>无效块

段概要：

第1块	第2块	第3块	•••	第N块
类型：数据块 inode号：5 数据块序号：13	类型：索引块 inode号：5 数据块序号：1	类型：索引块 inode号：n/a 数据块序号：n/a		类型：数据块 inode号：8 数据块序号：2





大纲

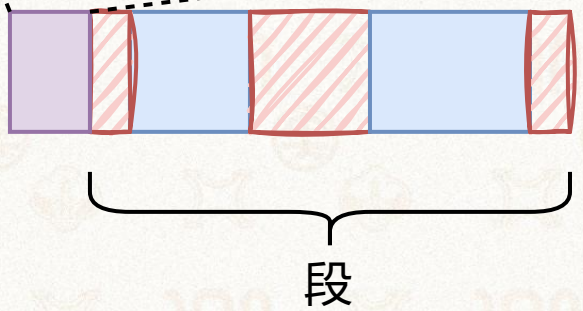
- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

挂载和恢复

- 扫描所有日志，重建出整个文件系统的内存结构
  - 大量无效数据也被扫描

段概要：

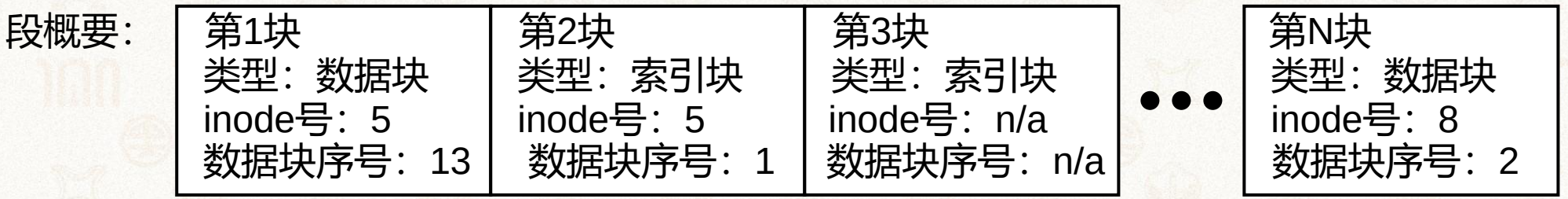
第1块	第2块	第3块	•••	第N块
类型：数据块 inode号：5 数据块序号：13	类型：索引块 inode号：5 数据块序号：1	类型：索引块 inode号：n/a 数据块序号：n/a		类型：数据块 inode号：8 数据块序号：2



挂载和恢复

- 问：在单机游戏社区中，玩家常说的S/L大法指什么？

- 定期写入检查点（checkpoint）
  - 写入前的有效数据，可以通过检查点找到
  - 只需扫描检查点之后写入的日志
  - 减少挂载/恢复时间

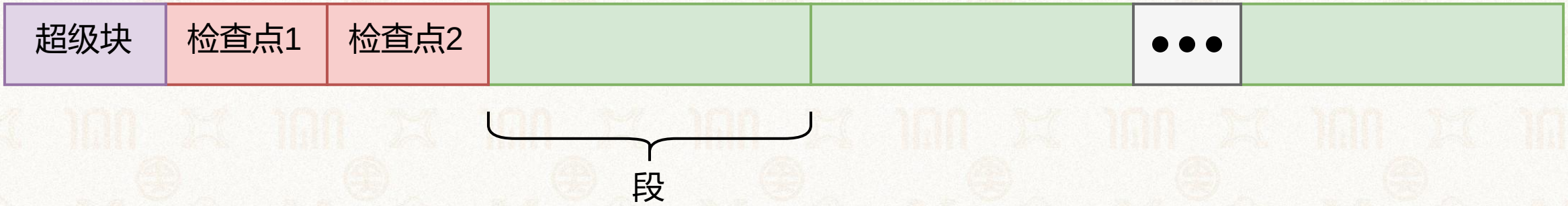


检查点

- 检查点内容
  - inode map的位置（可找到所有文件的内容）
  - 段使用表
  - 当前时间
  - 最后写入的段的指针

为什么需要两个检查点区域?

日志文件系统全貌:



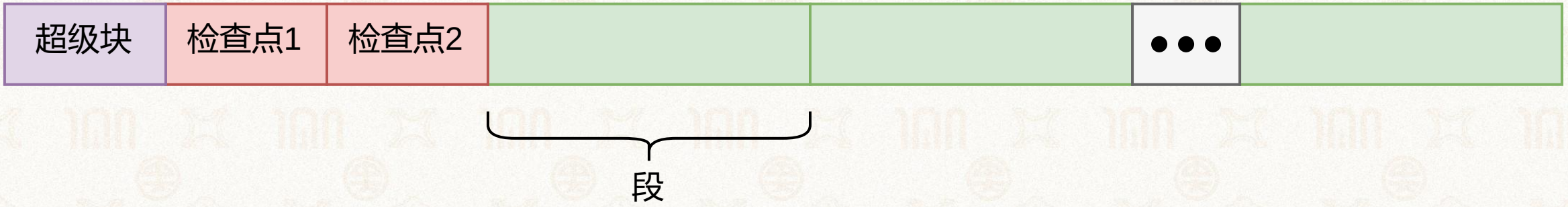
检查点

- 检查点内容
  - inode map的位置（可找到所有文件的内容）
  - 段使用表
  - 当前时间
  - 最后写入的段的指针

为什么需要两个检查点区域?

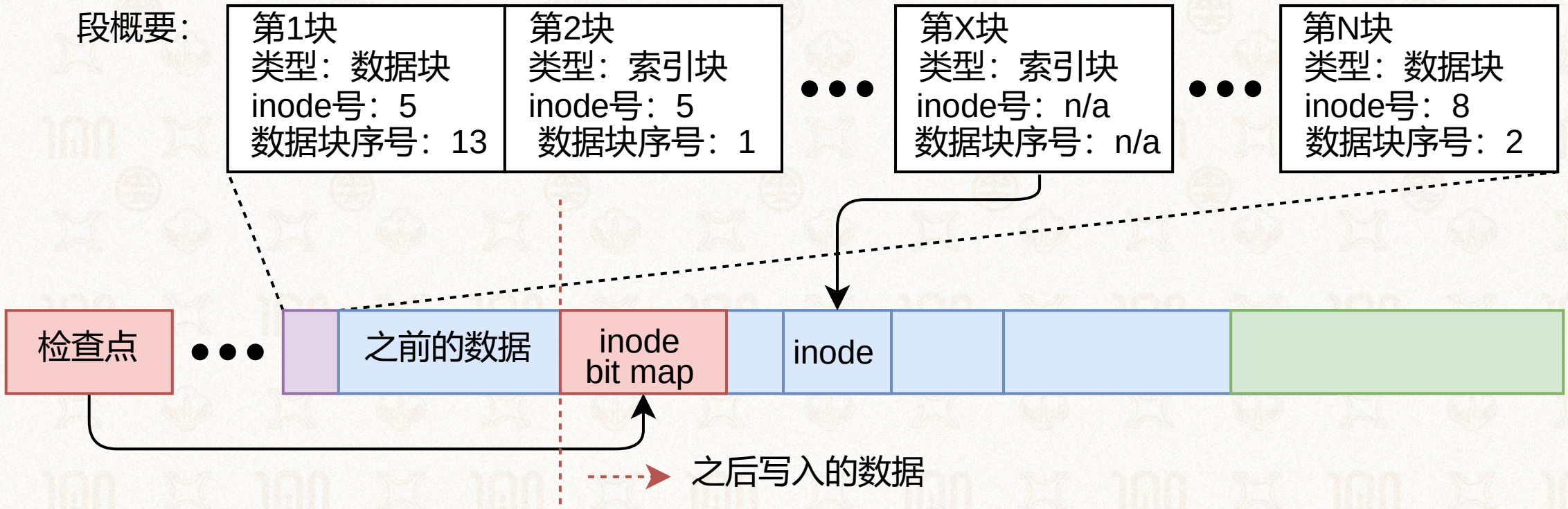
答：要假定系统随时可崩溃，万一写检查点时崩溃，至少还有另一个可用

日志文件系统全貌：



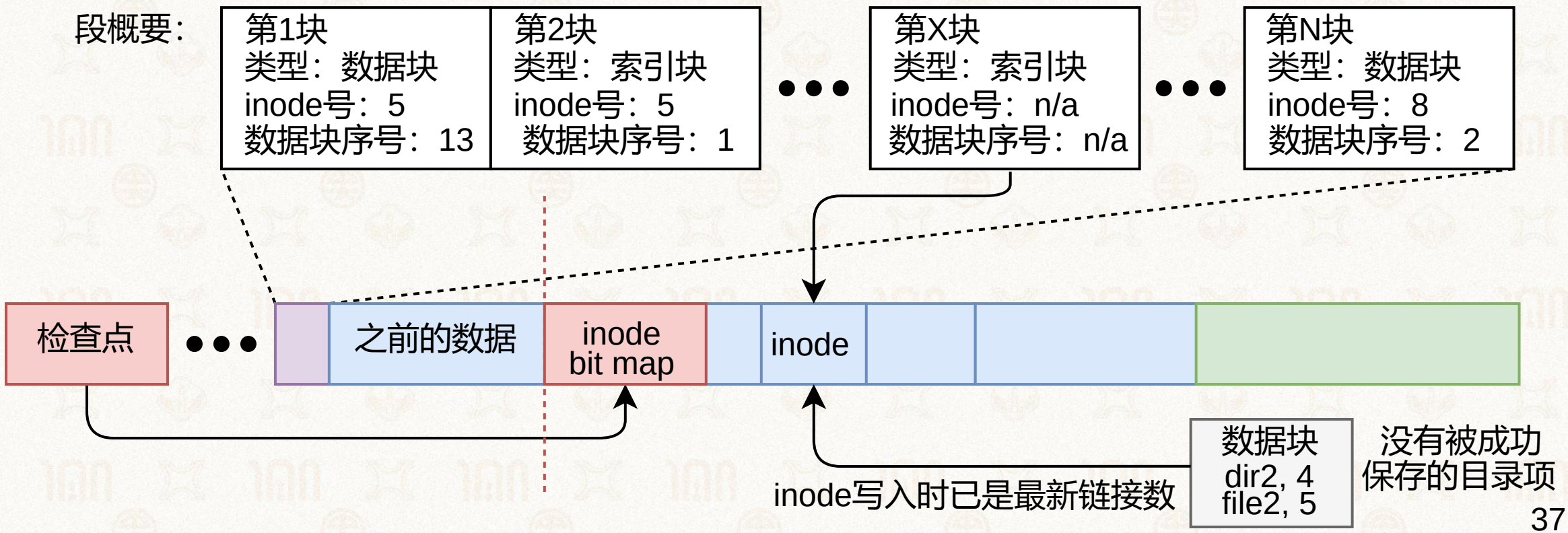
恢复：前滚（roll-forward）

- 尽量恢复检查点后写入的数据
- 通过段概要里面的新inode，恢复新的inode
  - 其inode中的数据块会被自动恢复
- 未被inode"认领"的数据块会被删除



恢复：前滚（roll-forward）

- 段概要无法保证inode的链接数一致性
  - 如：inode被持久化，但是指向其的目录项未被持久化
- 解决方案： 目录修改 日志 (套娃)



恢复：目录修改日志

- 目录修改日志
  - 记录了每个目录操作的信息
    - create、link、rename、unlink
  - 以及操作的具体信息
    - 目录项位置、内容、inode的链接数
- 目录修改日志的持久化在目录修改之前
  - 恢复时根据目录修改日志保证inode的链接数是一致的

其它日志文件系统的实现

- 光盘
  - UDF
- Flash/SSD
  - JFFS, JFFS2
  - UBIFS
  - LogFS
  - YAFFS, YAFFS2
  - F2FS
- 非易失性内存
  - NOVA
- RAID
  - WAFL (by NetApp)
- NVM
  - NOVA
- 云
  - 基于FUSE的ObjectiveFS

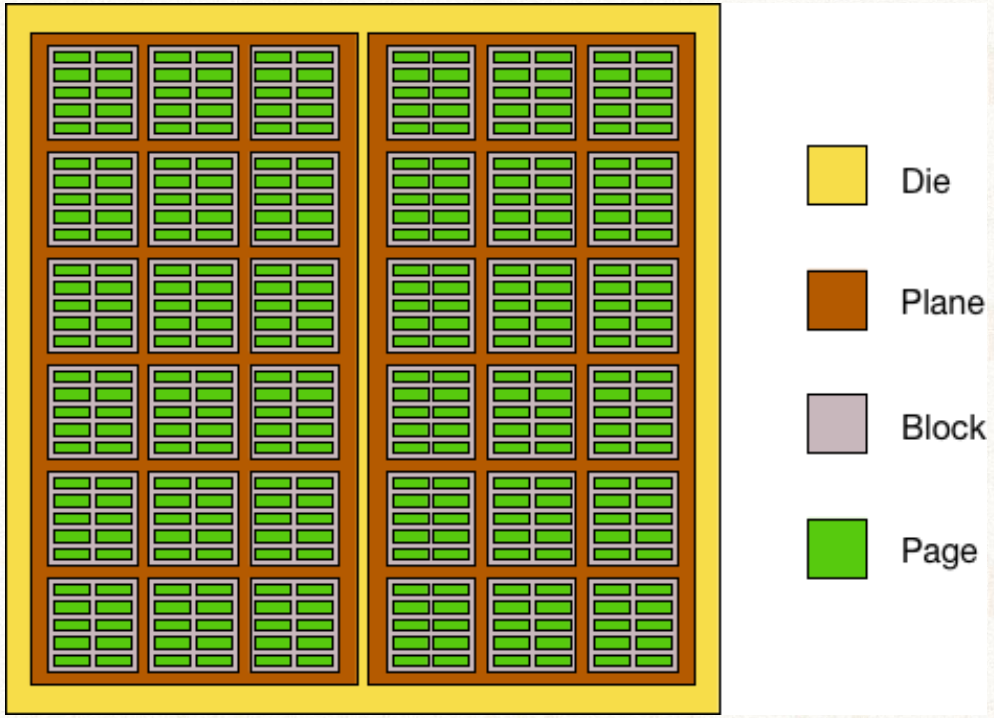
https://en.wikipedia.org/wiki/List_of_log-structured_file_systems

大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

闪存盘(NAND)的组织

- A chip/package
  - => 1/2/4 dies
  - => 1/2 planes
  - => n blocks （块）
  - => n pages （页）
  - => n cells
  - => 1/2/3/4 levels

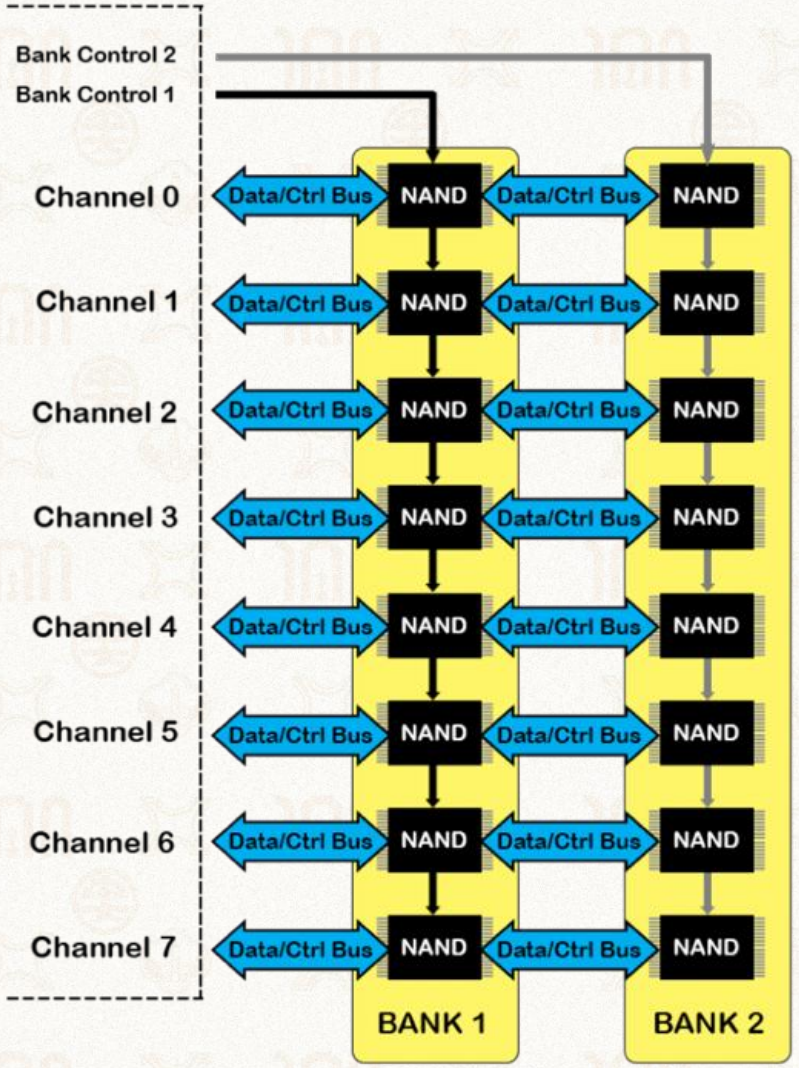


https://www.extremetech.com/electronics/274944-samsungs-new-v-nand-could-mean-cheap-plentiful-multi-terabyte-ssds

闪存盘的组织

- 通道（Channel）
  - 控制器可以同时访问的闪存芯片数量

- 多通道（Multi-channel）
  - 低端盘有2或4个通道
  - 高端盘有8或10个通道



NAND Banks and Channels Illustration

https://www.cactus-tech.com/resources/blog/details/solid-state-drive-primer-8-controller-architecture-channels-and-banks/

闪存盘的性质

- 非对称的读写与擦除操作
  - 页 (page) 是读写单元 (8-16KB)
  - 块 (block) 是擦除单元 (4-8MB)

顺序读、写速度不一致

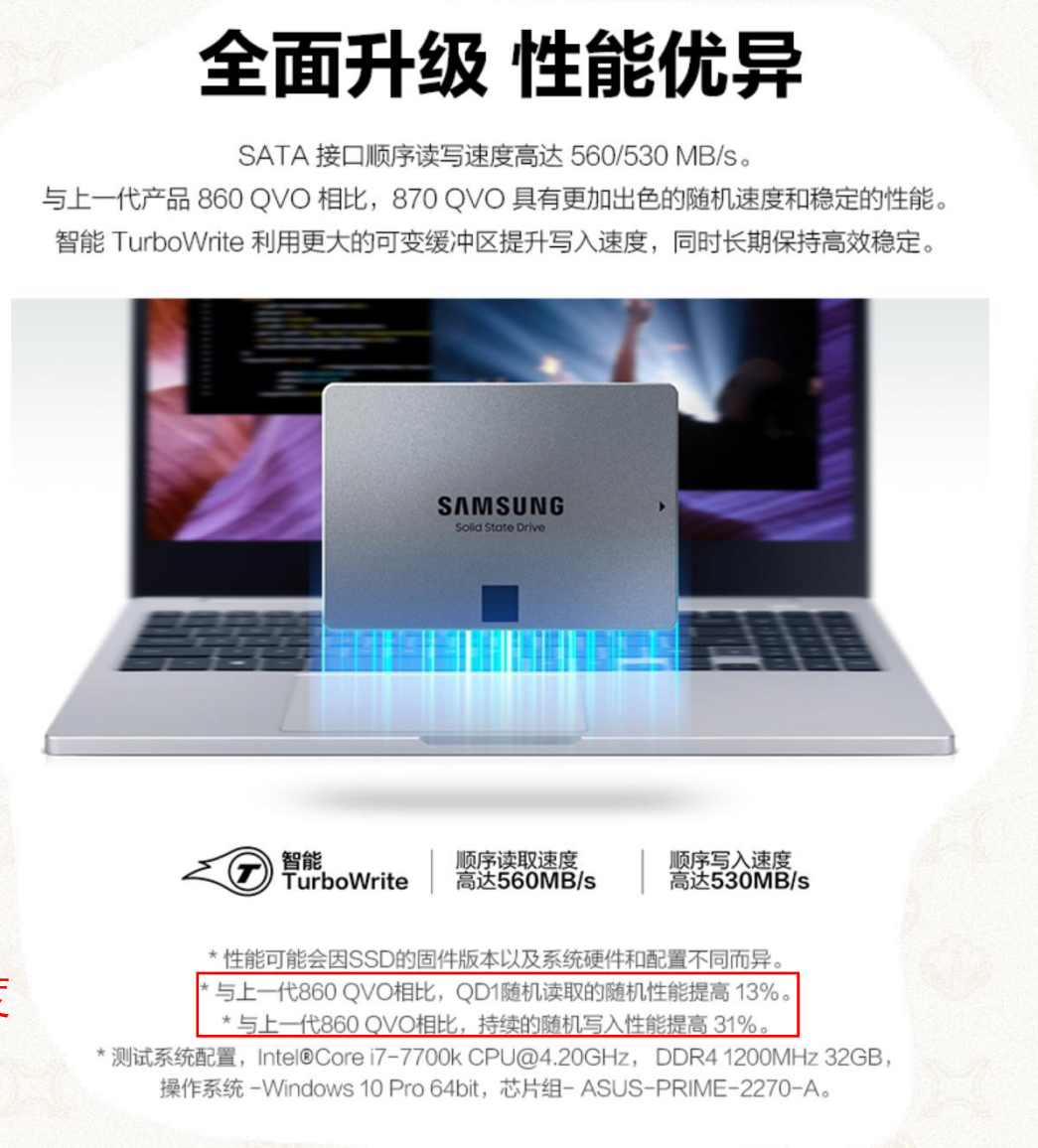
https://item.jd.com/100007847303.html

闪存盘的性质

- 非对称的读写与擦除操作
  - 页 (page) 是读写单元 (8-16KB)
  - 块 (block) 是擦除单元 (4-8MB)

- 随机访问性能
  - 没有寻道时间
  - 随机访问的速度提升，但仍与顺序访问有一定差距

很鸡贼地不标随机读写速度

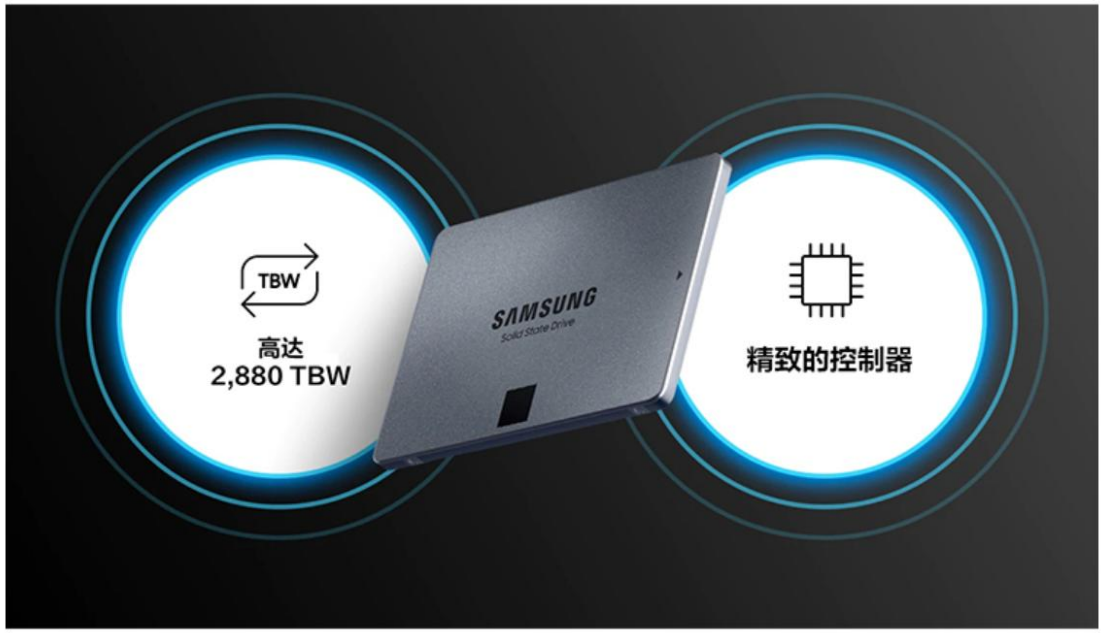


https://item.jd.com/100007847303.html

闪存盘的性质

- 非对称的读写与擦除操作
  - 页 (page) 是读写单元 (8-16KB)
  - 块 (block) 是擦除单元 (4-8MB)
- 随机访问性能
  - 没有寻道时间
  - 随机访问的速度提升，但仍与顺序访问有-
- Program/Erase cycles
  - 写入前需要先擦除
  - 每个块被擦除的次数是有限的

TBW是什么意思?



* TBW: 总写入字节数
* 保修TBW:1TB SSD 为 360 TBW，2TB SSD 为 720 TBW，
4TB SSD 为 1,440 TBW，8TB SSD 为 2,880 TBW
*3年或TBW的有限保修，以先到者为准。有关保修的更多信息，请参考包装中随附的保修文件
* ECC：纠错码。

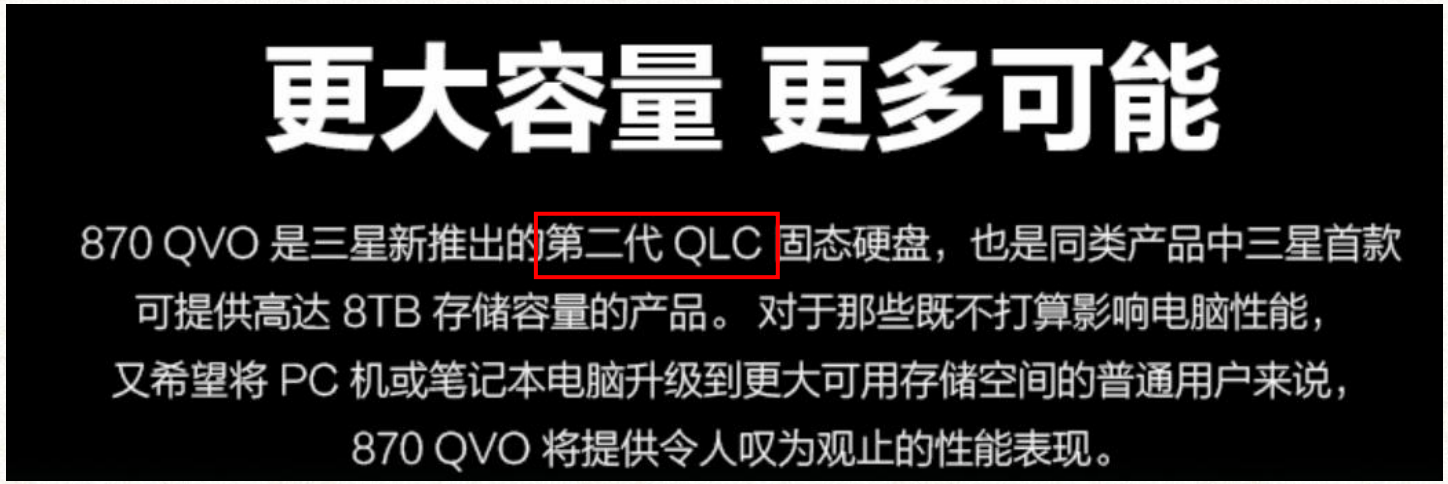
对于1T的固态硬盘，平均写360遍，硬盘就可能报废了

https://item.jd.com/100007847303.html

闪存盘的性质

- 多通道
  - 高并行性

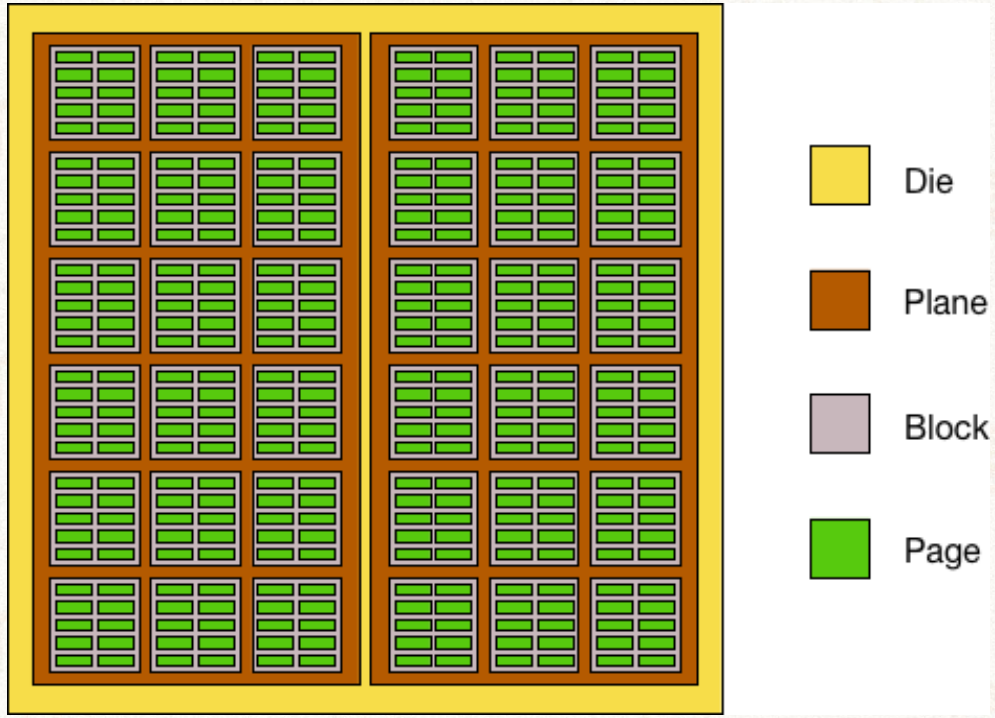
- 异质Cell
  - 存储1到4个比特：SLC 、MLC、TLC、 QLC



https://item.jd.com/100007847303.html

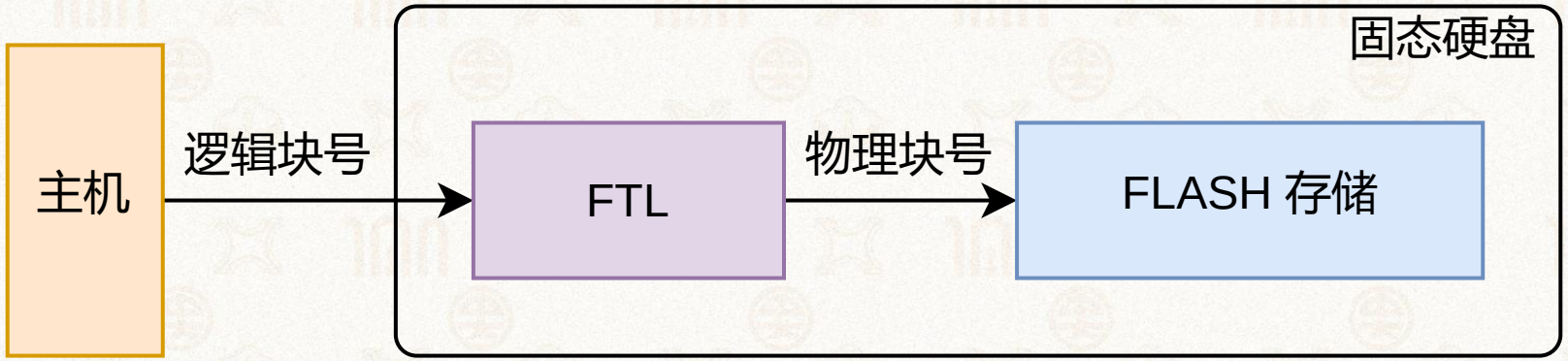
闪存盘的性质

- 多通道
  - 高并行性
- 异质Cell
  - 存储1到4个比特：SLC 、MLC、TLC、 QLC
- 磨损均衡
  - 频繁写入同一个块会造成写穿问题
  - 将写入操作均匀的分摊在整个设备



Flash Translation Layer (FTL)

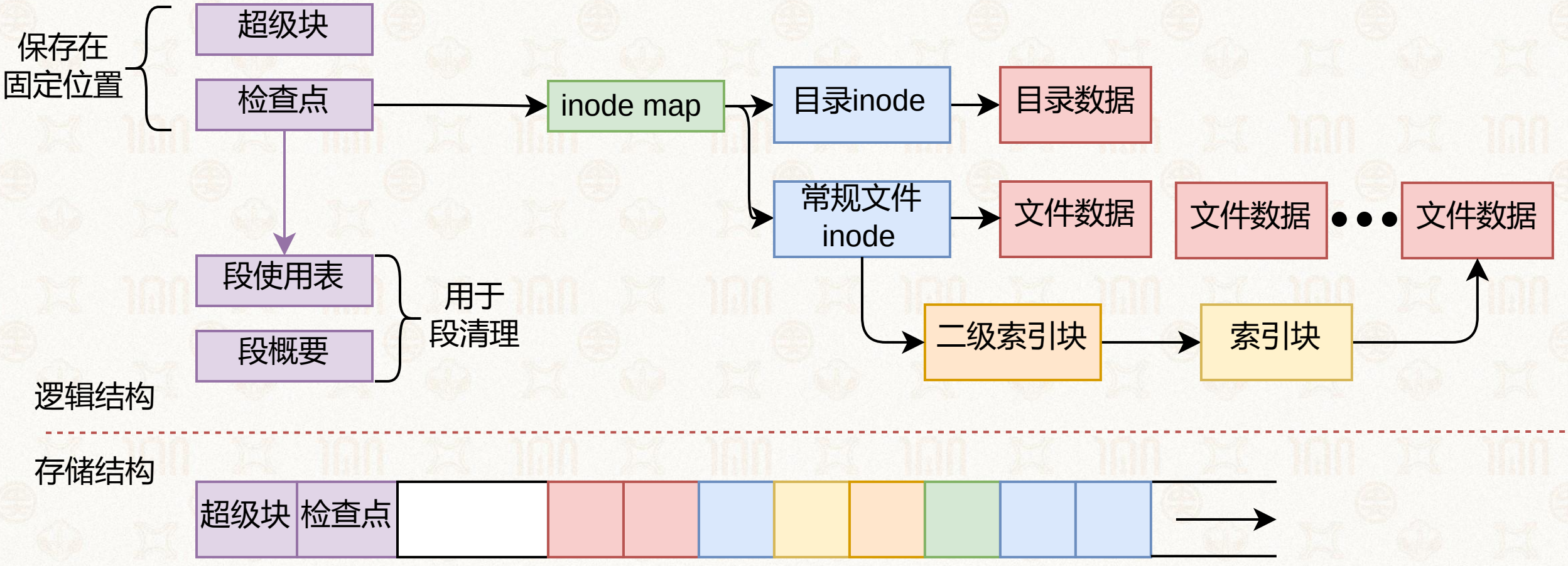
- 逻辑地址到物理地址的转换
  - 对外使用逻辑地址
  - 内部使用物理地址
  - 可软件实现，也可以固件实现
  - 用于垃圾回收、数据迁移、磨损均衡（wear-levelling）等



如何利用这些性质为闪存设计文件系统?

- 日志文件系统
  - 段清理后才能使用
  - 顺序写入
  - 需要清理（垃圾回收）
- 闪存
  - Block擦除后才能使用
  - 需要考虑磨损均衡
  - 需要垃圾回收

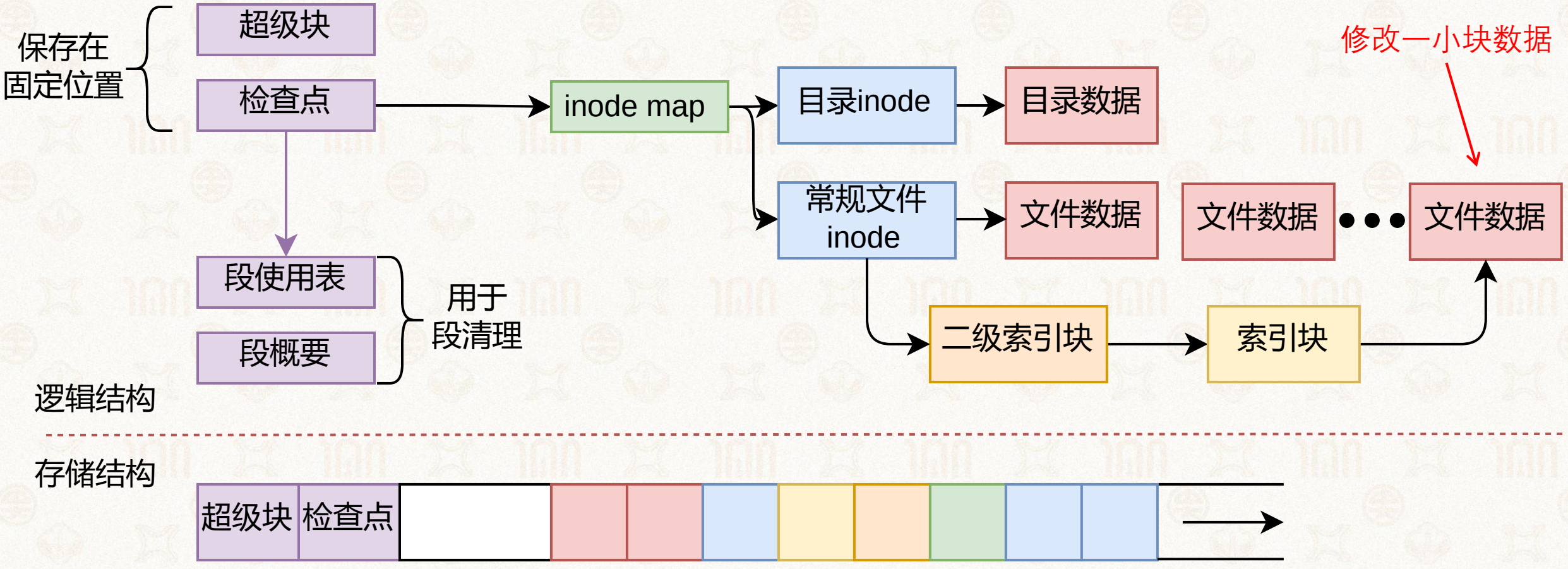
日志文件系统的结构


保存在
固定位置
超级块
检查点
inode map
目录inode
目录数据
常规文件
inode
文件数据
文件数据
文件数据
段使用表
段概要
用于
段清理
二级索引块
索引块
逻辑结构
存储结构
超级块
检查点

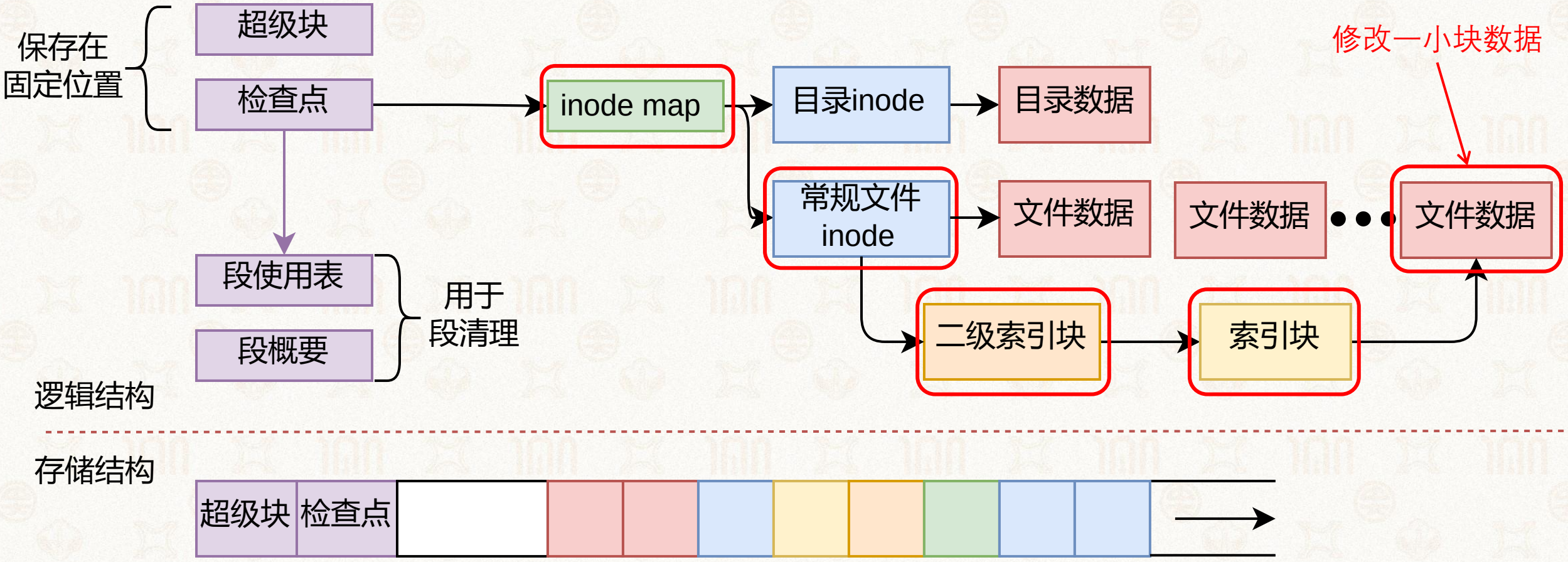
大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

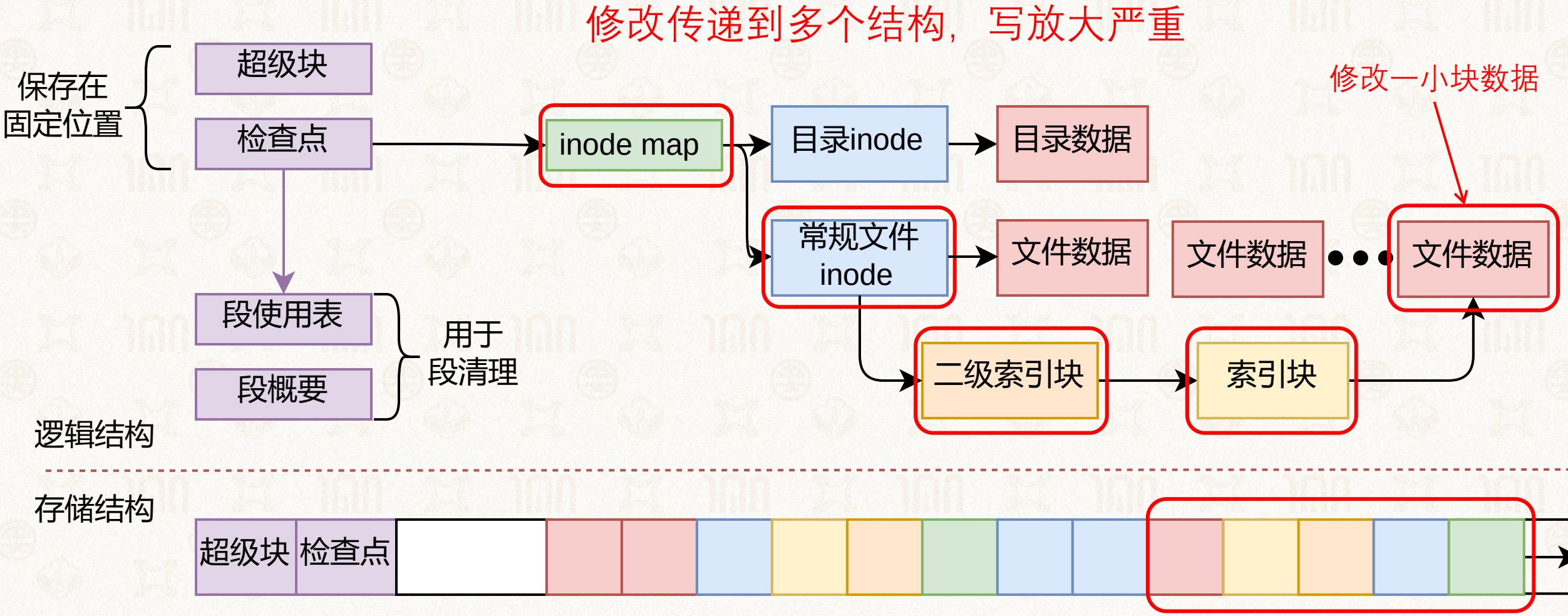
日志文件系统的问题1：递归更新



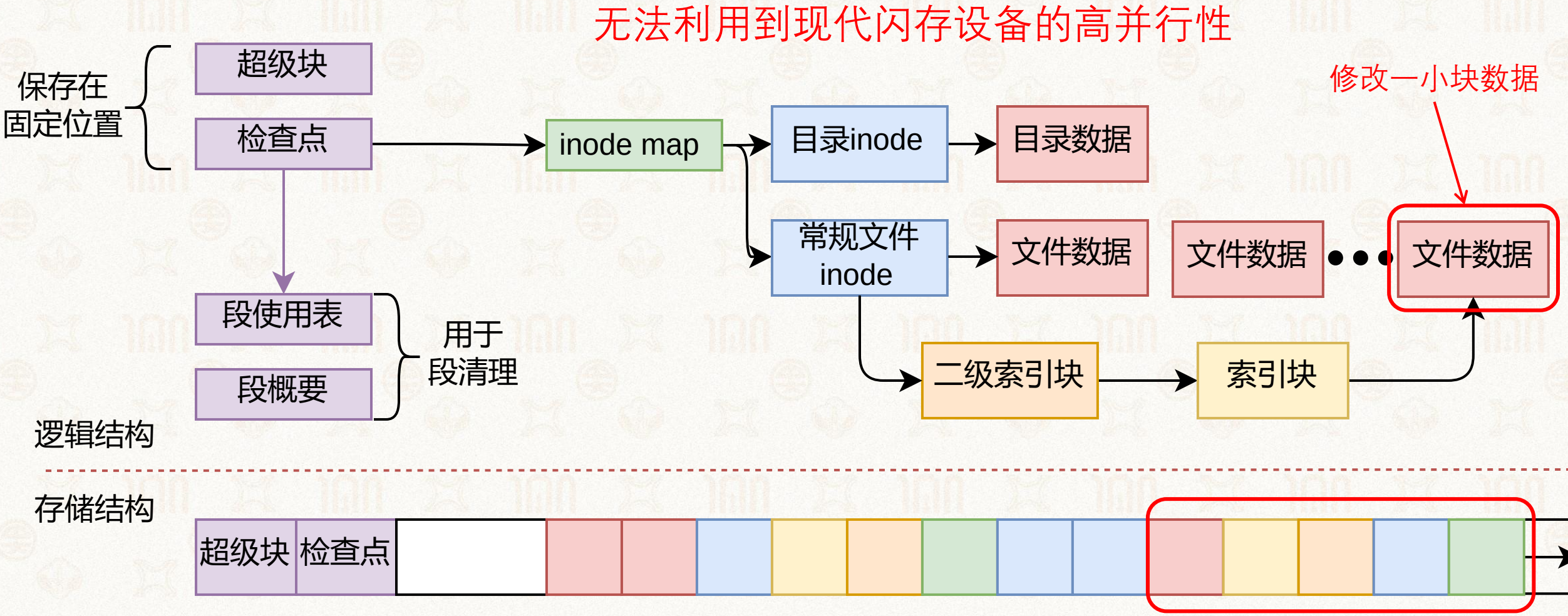
日志文件系统的问题1：递归更新



日志文件系统的问题1：递归更新



日志文件系统的问题2：单一日志顺序写入



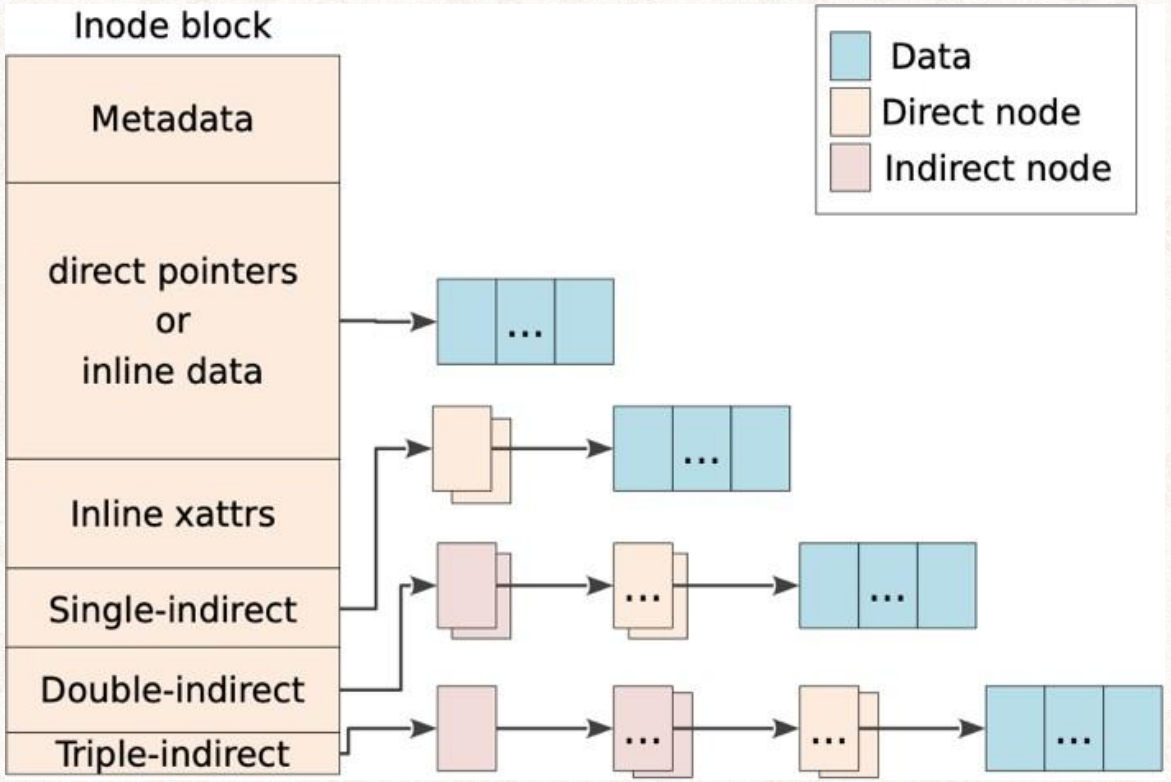
大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

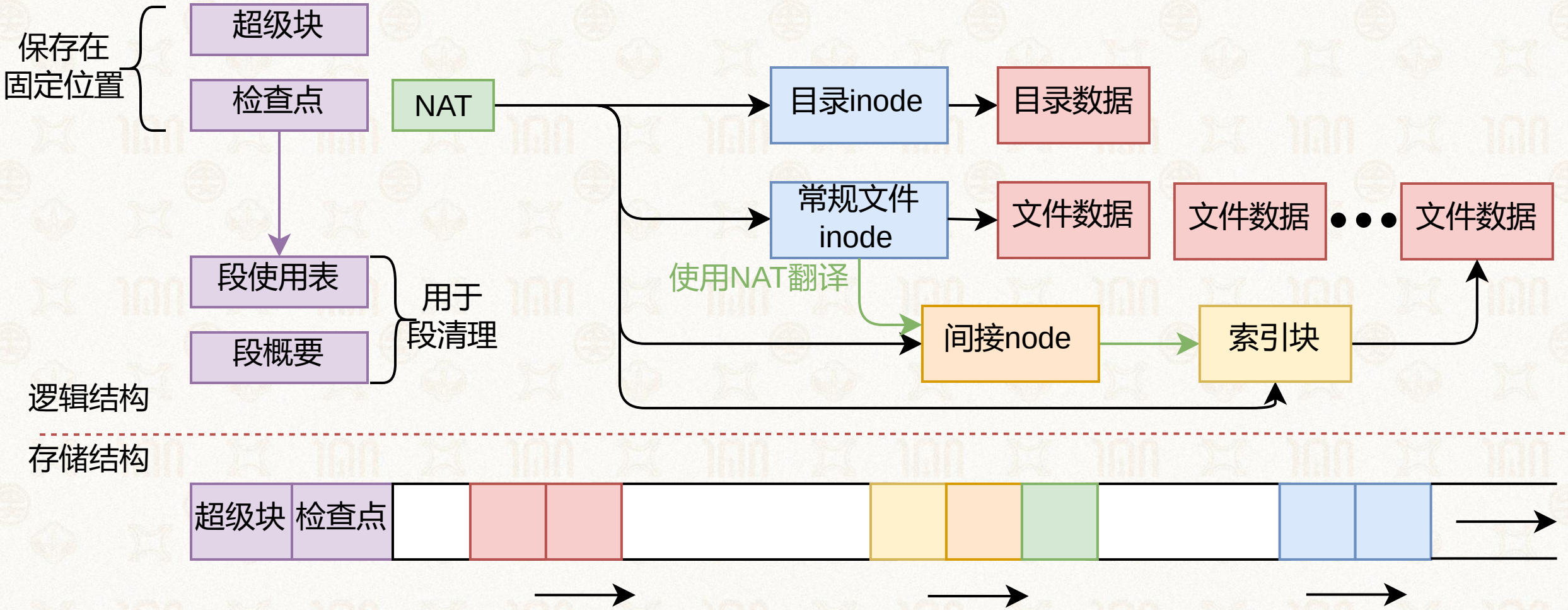
F2FS的改进1：NAT

- 引入一层 indirection：NAT(node地址转换表)
  - NAT：Node Address Table
  - 维护node号到逻辑块号的映射
  - Node号需转换成逻辑块号才能使用

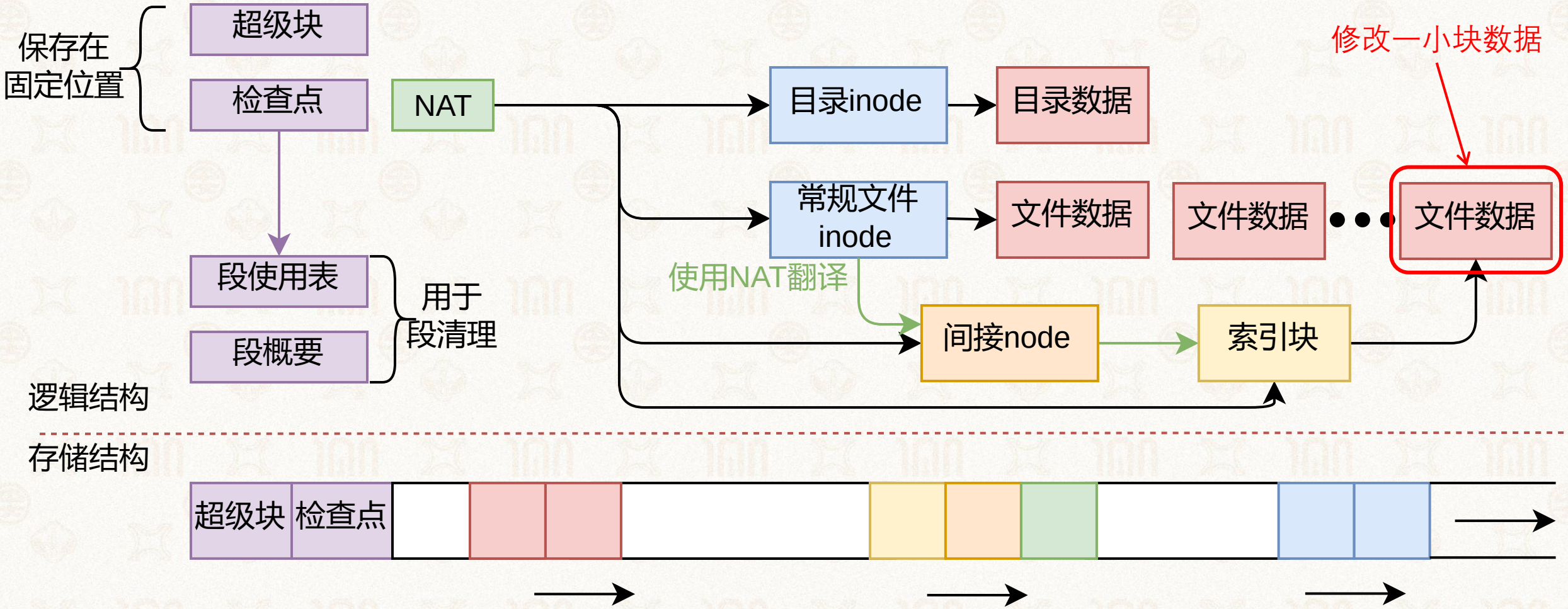
- F2FS中的文件结构
  - 直接node：保存数据块的逻辑块号
  - 间接node：保存node号(相当于索引块)
  - 数据块：保存数据



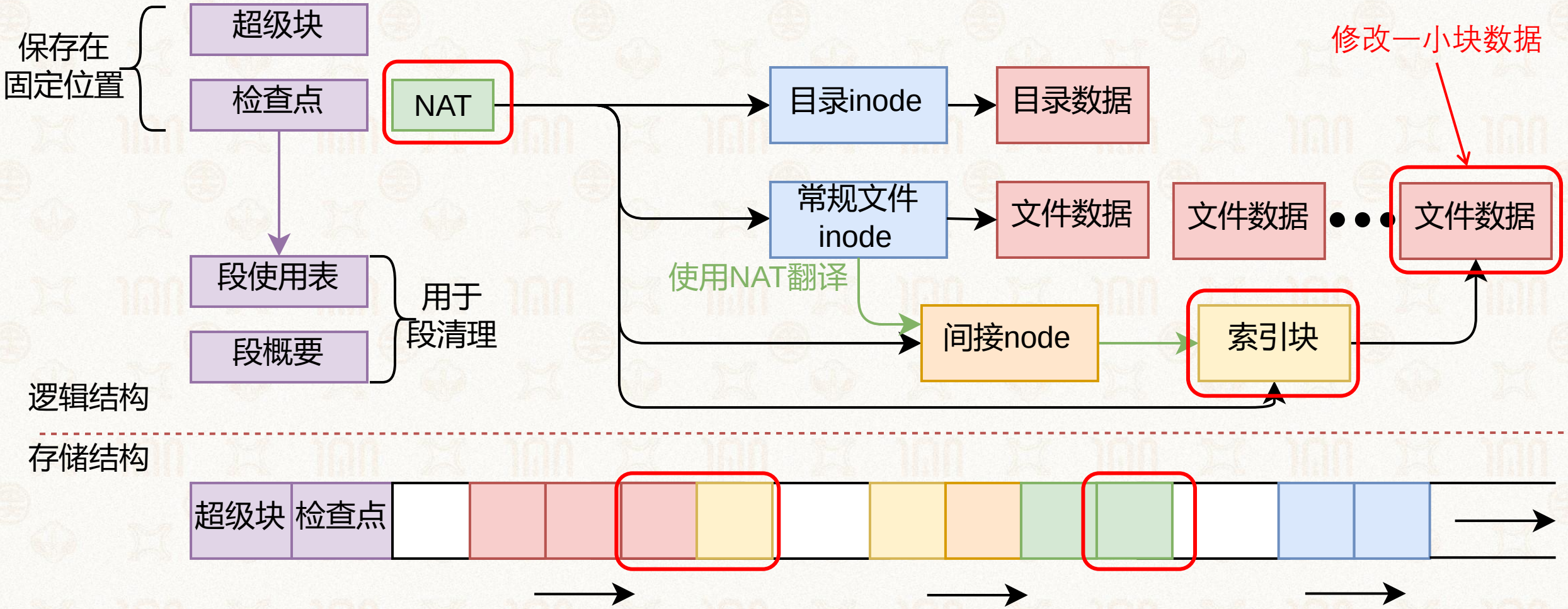
F2FS的改进1：NAT



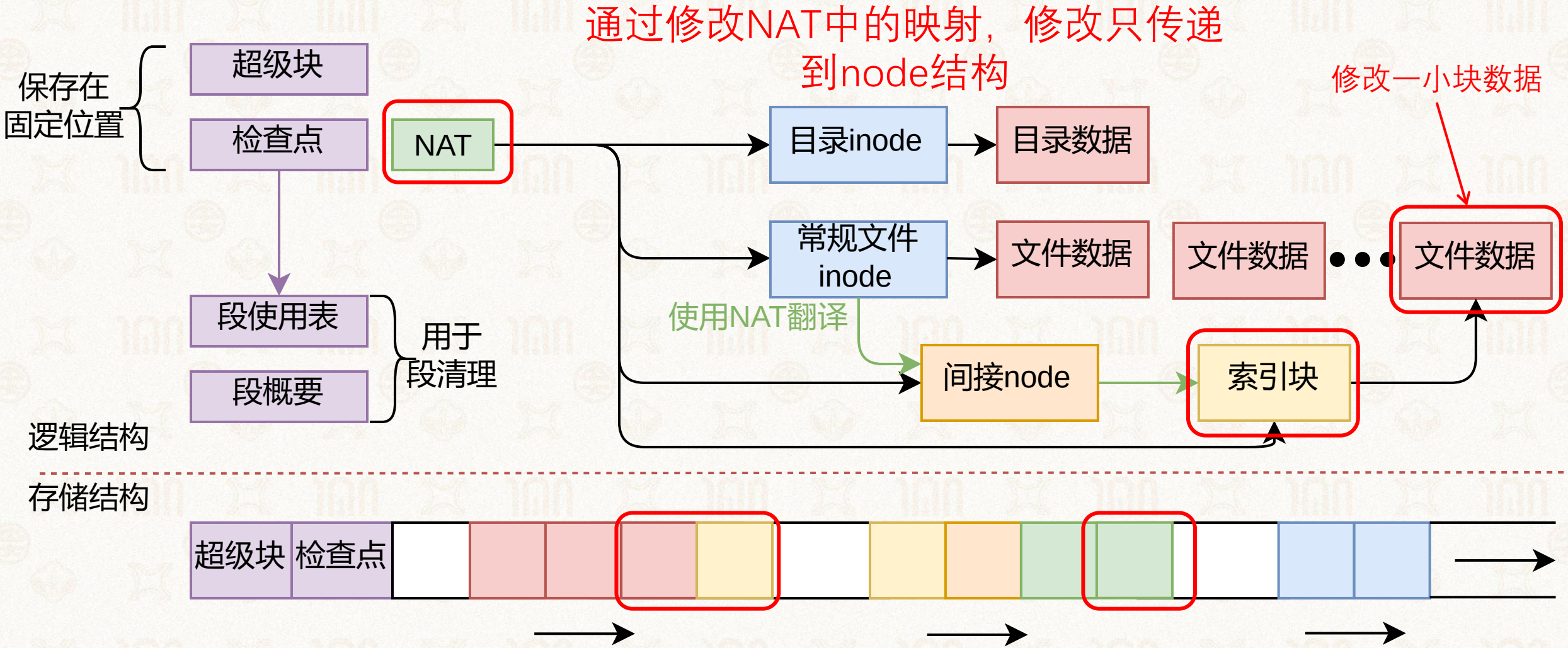
F2FS的改进1：NAT



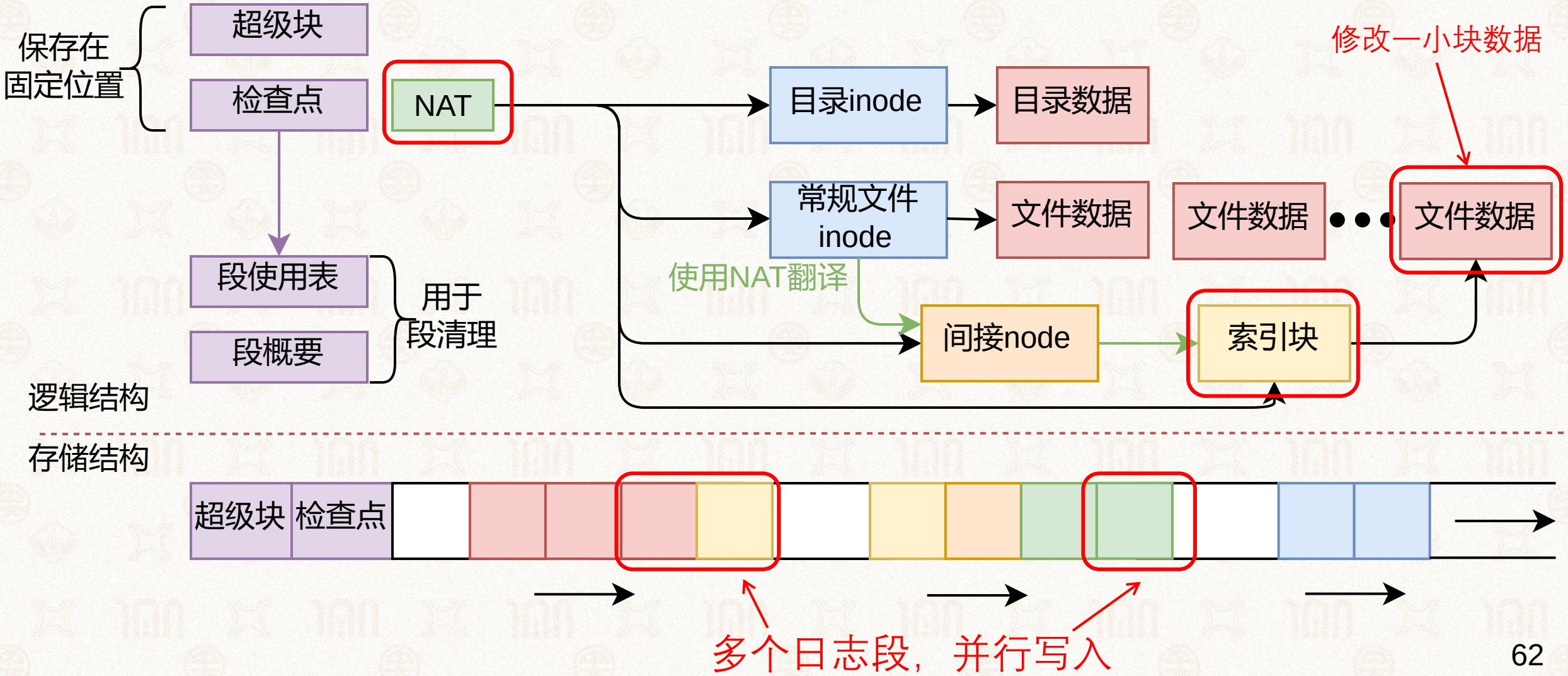
F2FS的改进1：NAT



F2FS的改进1：NAT

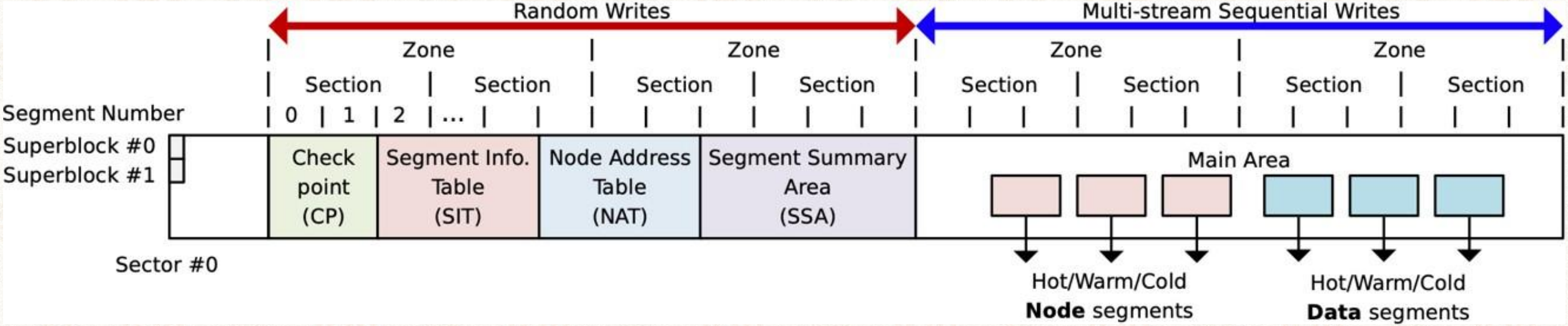


F2FS的改进1：NAT



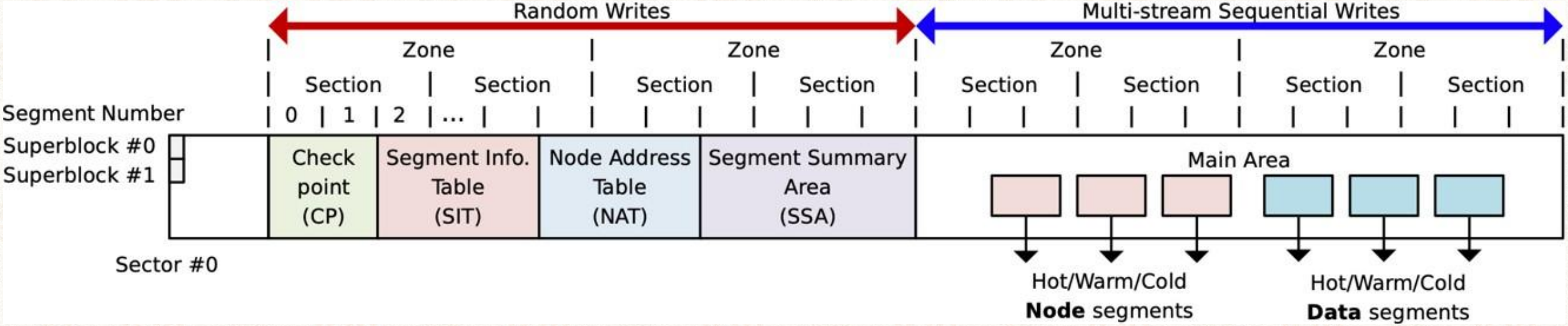
闪存友好的磁盘布局

- 组织层级
  - Block：4KB，最小的读写单位
  - Segment(段)：2MB
  - Section：多个segment（垃圾回收/GC粒度）
  - Zone：多个section



闪存友好的磁盘布局

- 系统元数据（随机写入）
  - 存放在一起：局部性更好
  - CP：检查点
  - SIT：段信息表
  - NAT：node地址转换表
  - SSA：段概要区域
- 数据区（多Log顺序写入）
  - 区分冷/温/热数据
  - 区分文件数据（data segment）与元数据（node segment）



大纲

- 文件系统崩溃一致性是什么
  - 文件系统一致性约束
  - 崩溃与恢复
- 原子更新技术
  - 日志
    - 日志系统JBD2
  - 写时复制
- Soft Updates
    - 不详细讲，太复杂，有兴趣同学自己看
- 日志文件系统
  - 基本特点
  - 空间回收
  - 崩溃恢复
- 闪存友好的文件系统
  - 闪存盘的性质
  - 传统日志文件系统的问题
  - 闪存文件系统的改进

谢谢

微信：suyuxin
钉钉：苏玉鑫
B站：https://space.bilibili.com/502854403
软工集市课程专区：https://ssemarket.cn/new/course
匿名提问箱：https://suask.me/ask-teacher/106/苏玉鑫