CHINA LINUX KERNEL
中国Linux内核开发者大会

vivo

MD RAID无锁位图优化机制

MD RAID Lockless bitmap Optimization Mechanism

华为 OS内核实验室

郑琪星
余快（特性作者）

目 录

CONTENTS

背景

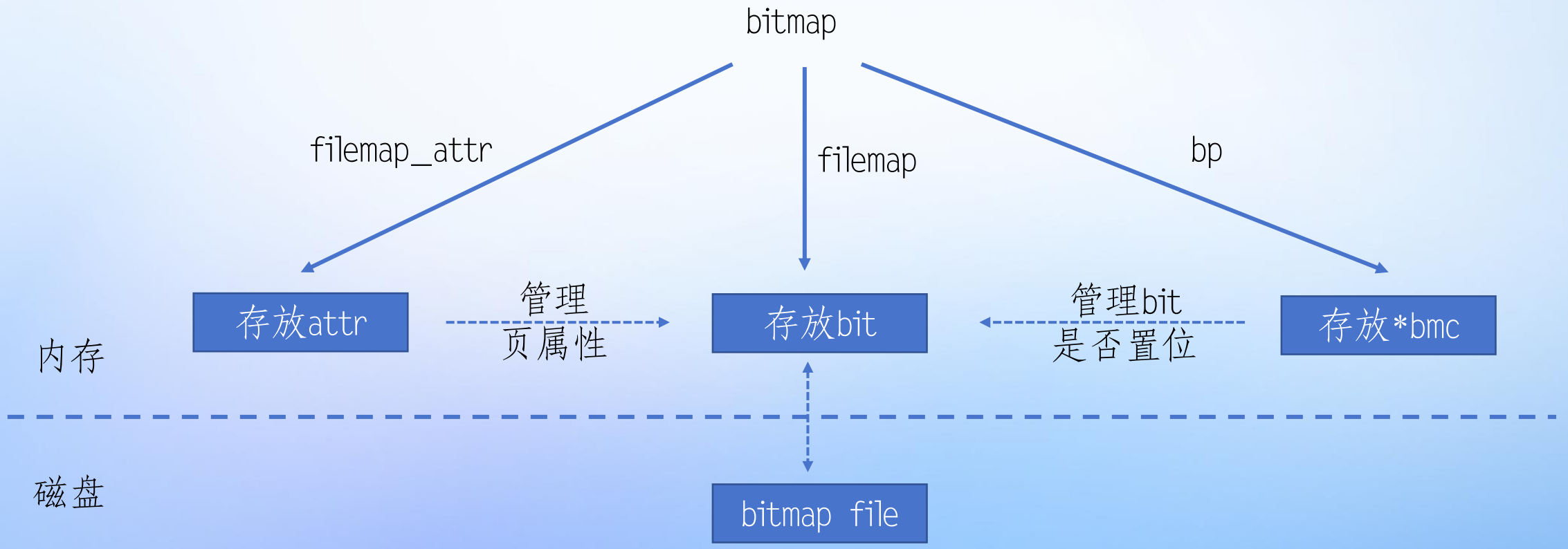
RAID系统需要维护多份数据副本或校验数据来提供容错能力，
但多副本之间可能因为**断电**、**磁盘损坏**、**系统故障**等原因出现数据不一致。

引入位图（bitmap）机制

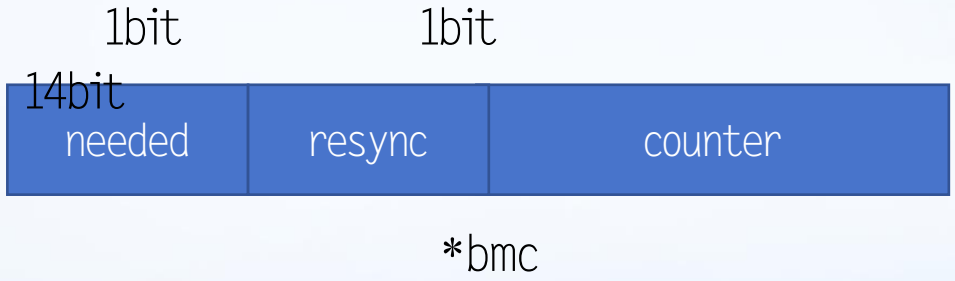
- **追踪**哪些数据可能不一致
- **同步**不一致的数据
- **避免**全盘恢复

Why bitmap?

bitmap可以记录raid数据一致性信息 → raid实现增量式数据同步/恢复



bmc：管理bitmap中bit是否置位



- needed——表示该bit对应的chunk是否需要同步
- resync——表示该bit对应的chunk是否正在同步
- counter——用来统计对应的chunk尚未完成的写请求的计数

filemap_attr：管理bitmap file缓存中page属性

bitmap file缓存自身的每一个page都有filemap_attr来管理页缓存状态，用于表示该缓存是否有效、是否脏等。

- BITMAP_PAGE_DIRTY——表示内存bitmap有bit被置位，但是bitmap file对应的位没有被置位，因此该page需要同步刷到磁盘，写磁盘完成才能继续；
- BITMAP_PAGE_PENDING——过渡状态，表示内存bitmap有可以清除的bit，则需要清除该bit，然后过渡到BITMAP_PAGE_NEEDWRITE状态；
- BITMAP_PAGE_NEEDWRITE——表示内存bitmap有bit被清除，但是bitmap file对应的位没有被清除，因此该page需要刷到磁盘，但是异步进行的，因为即使写失败，最多带来额外的同步，不带来数据的危害。

现有bitmap结构性缺陷

在io快速路径中，bitmap更新需要持同一把spin锁，高并发写IO锁竞争剧烈

瓶颈场景：

- 小随机写（4K~64K）+ 高并发：多核同时写不同 chunk，但被同一把锁串行化
- RAID5/6 写放大场景：本身有条带/校验访问，位图路径再加锁，争用更加明显
- NUMA 多节点：全局锁的 cacheline 在节点间抖动，跨节点一致性开销较大

无锁位图(llbitmap)机制

1. IO fast path 无锁化
   - 不需要维护设备级别自旋锁，标脏级别：页→块
   - 同一位/同一逻辑块短期多次写只计第一次，后续零额外bitmap开销
2. 只同步必要块区域
   - 新建/替换盘/断电重启时不做全盘同步/恢复
   - RAID4/5/6仅对写过的数据构建/恢复

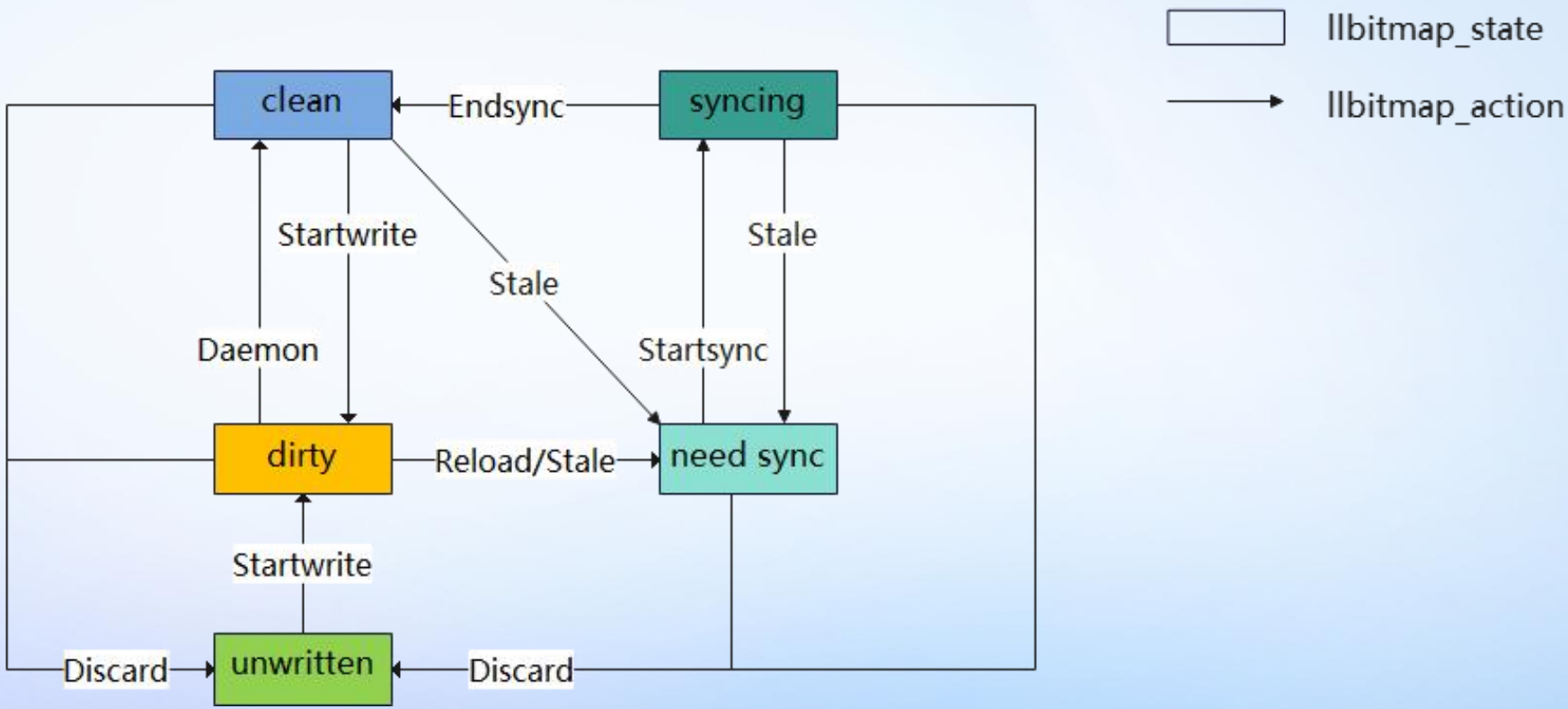
IO fast path

针对高并发的读写raid场景，io路径中需要尽量保证无锁、轻量化操作以保证数据吞吐，发挥高性能SSD设备的优势

- 无锁快路径 + 块级脏追踪

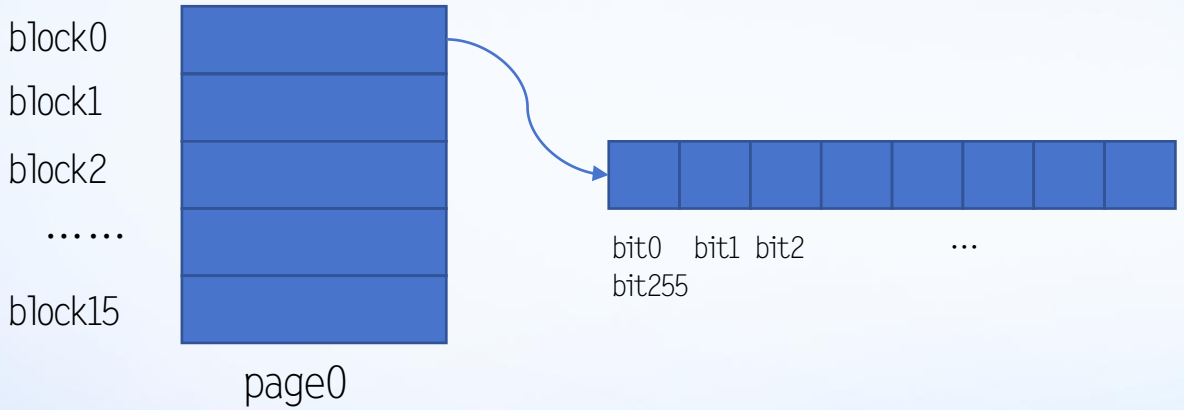
自旋锁保护的bitmap页管理 → 依赖状态机转换的**无锁**bit更新、按块标记“脏”

状态机转换

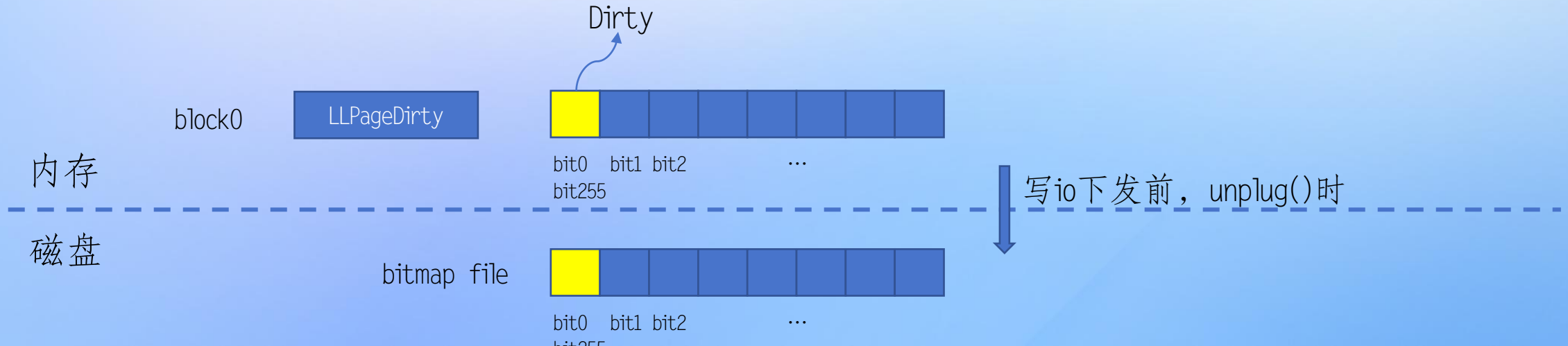


- Unwritten：从未写过（新阵列/被discard的区域） → 恢复/重建时可直接**跳过**
- Clean：数据一致
- Dirty：已写过，但阵列健康，后台稍后清为 Clean → 为了聚合与减抖，不急着清
- NeedSync：必须同步/恢复（例如降级时新写、掉电重装后曾经的 Dirty、RAID5/6 首写要“懒构建”校验）
  → 交由resync同步线程处理
- Syncing：正在同步/恢复中

llbitmap 页面管理器 pctl 将位图 page 切分为 block，使用 pctl->dirty[] 按块标脏

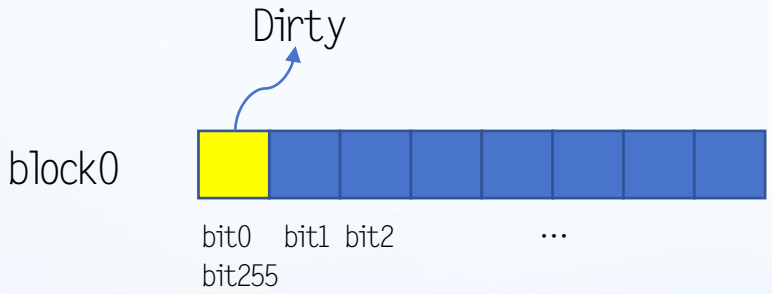


处理写请求时，更新对应bit为Dirty/NeedSync，不拿锁，写io下发前将对应block元数据落盘



若同一 block 在 barrier_idle 窗内再次被触及，将此 block 内所有bit一次性提升到Dirty/NeedSync

T1



T2

test_and_set_bit(block, pctl->dirty) -> 返回非0

T3

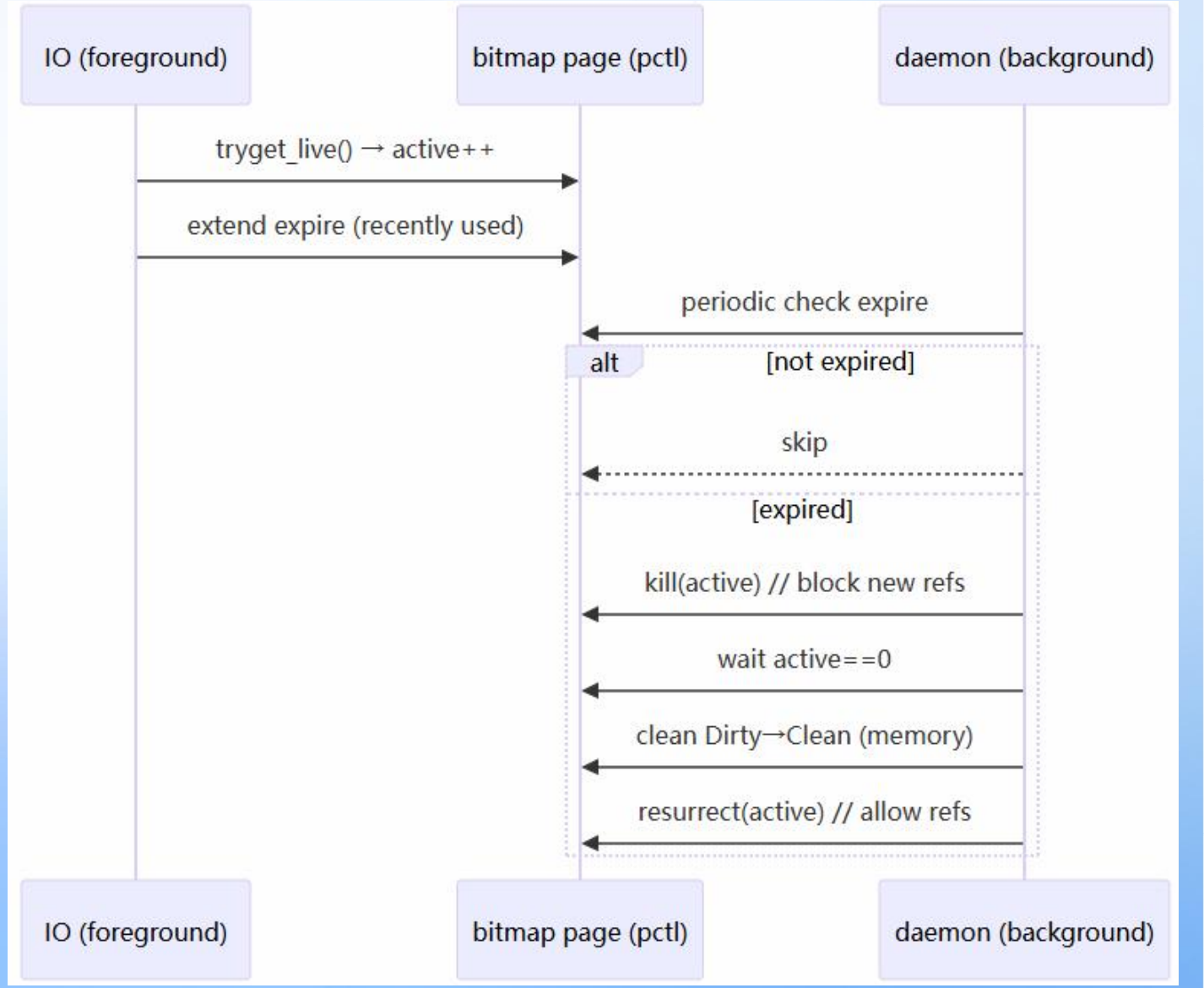
顺序写的场景下，同一sub page中256次原子操作

降低到

2次

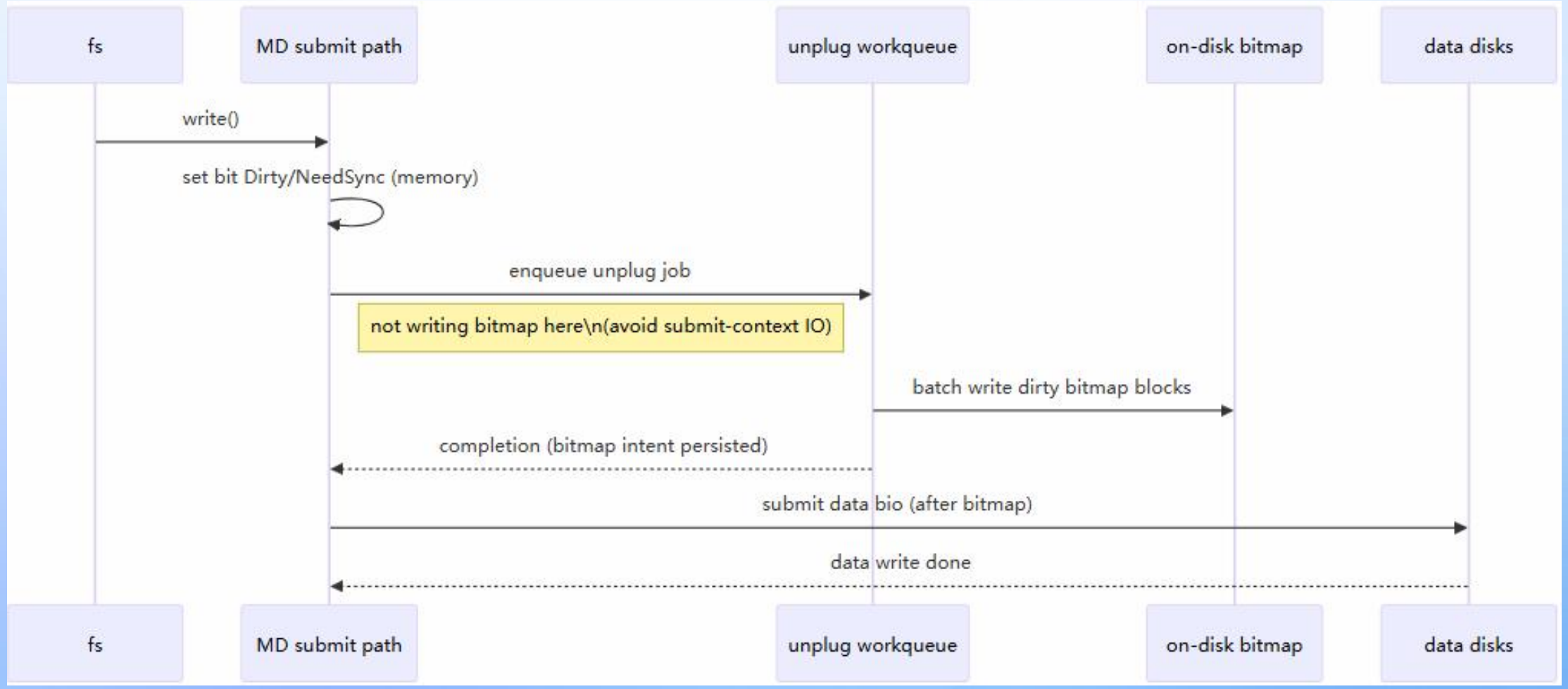
ú 页级屏障 + 后台清理

- llbitmap 每个位图页有一个引用计数屏障 active 与到期时间 expire，表示当前页近期是否在使用
- 后台进程定期处理到期的bitmap页，挂起页做清脏处理



ú 批量写回 + 工作队列提交

- 写回按页内block粒度批量提交，遍历pctl->dirty[]将内存中置脏的block同步到磁盘bitmap区
- 专用工作队列负责写回，避免在io提交上下文更新磁盘bitmap



技术收益

- **正常业务io：** bitmap 更新变为 lockless，不再阻塞 fast path
- **初始化 / 恢复：** 跳过unwritten区域，减少同步范围

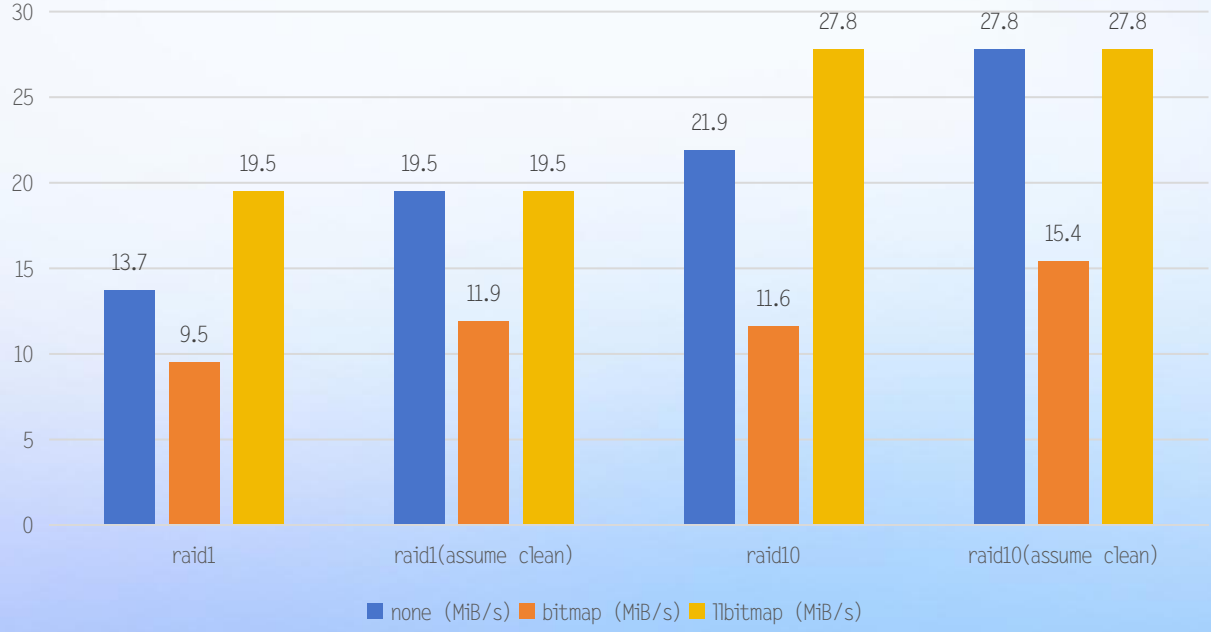
技术验证场景： VM中通过fio随机写测试来构建一个20GB的内存盘阵列

- 使用RAM disk是为了排除磁盘的瓶颈
- fio randwrite 是为了模拟阵列创建及同步过程中的随机写压力

技术收益

ú 镜像类：raid1/raid10

llbitmap 的情况下几乎与”none bitmap”性能持平，比bitmap提升60%~130%

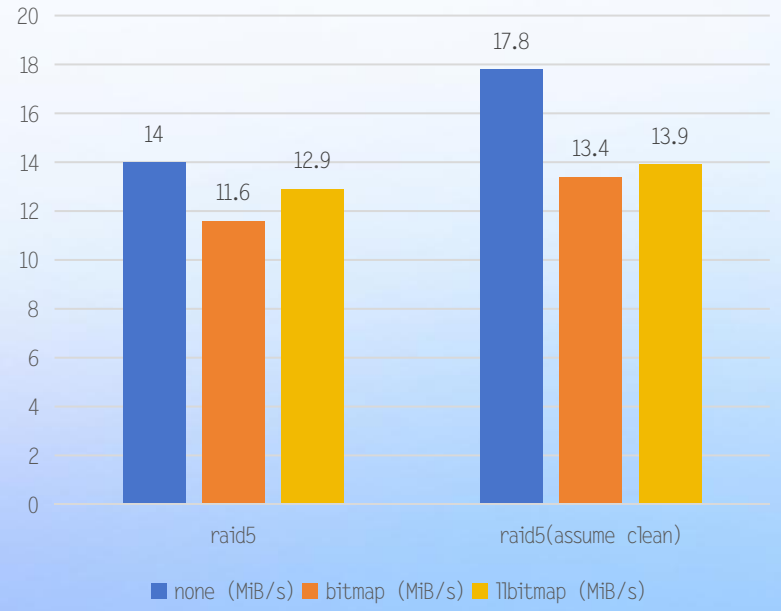


- bitmap最慢：主要为锁竞争带来的额外开销
- llbitmap较快，在数据不可信情况下构建性能超过 none bitmap

技术收益

ú 校验类：raid4/raid5/raid6

llbitmap 收益有限，但比旧 bitmap 开销小，提升4%~11%



- RAID5 的瓶颈主要在校验写路径（RMW / 重算XOR），即使是无锁仍有位图操作开销

总 结 与 讨 论

收益场景

- **高并发写入的数据库 / 日志系统**：当大量短小写请求集中命中同一数据段（例如某些写热点）时，无锁位图避免跨CPU抢锁和cacheline bounce，延迟显著降低。
- **云盘与分布式存储后端**：在云环境下，常见磁盘替换与重建场景多发；仅针对写入过的数据做重同步能缩短恢复时间窗口并减少网络/磁盘 IO。
- **恢复窗口优化**：创建新阵列或换盘时，避免立即进行全盘重同步，有助于提高服务可用性并加快可写入就绪时间。

使用约束

- 目前只支持在磁盘上（内部元数据）持久化bitmap file，不支持外部元数据

Linux社区

- 社区状态：特性已于Linux 6.18-rc1合入主线
- 补丁链接： https://git.kernel.org/pub/scm/linux/kernel/git/Torvalds/linux.git/commit/drivers/md?h=v6.18-rc1&id=e1b1d03ceec3
- News：

  https://www.phoronix.com/news/Linux-6.18-Block-IO-Uring

Merge tag 'for-6.18/block-20250929' of git://git.kernel.org/pub/scm/linux/kernel/git/axboe/linux

```
Pull block updates from Jens Axboe:

 - NVMe pull request via Keith:
     - FC target fixes (Daniel)
     - Authentication fixes and updates (Martin, Chris)
     - Admin controller handling (Kamaljit)
     - Target lockdep assertions (Max)
     - Keep-alive updates for discovery (Alastair)
     - Suspend quirk (Georg)

 - MD pull request via Yu:
     - Add support for a lockless bitmap.

       A key feature for the new bitmap are that the IO fastpath is
       lockless. If a user issues lots of write IO to the same bitmap
       bit in a short time, only the first write has additional overhead
       to update bitmap bit, no additional overhead for the following
       writes.

       By supporting only resync or recover written data, means in the
       case creating new array or replacing with a new disk, there is no
       need to do a full disk resync/recovery.
```

Linux 6.18 Block Code Introduces Lockless Bitmap For Software RAID

Written by Michael Larabel in Linux Storage on 8 October 2025 at 03:27 PM EDT. 28 Comments

Last week the block subsystem and IO_uring updates were merged for the Linux 6.18 kernel with a few items to draw attention to.

Via the Multiple Device "MD" software RAID support is now a new lockless bitmap option in Linux 6.18. Huawei engineer Yu Kuai led the work on this MD lockless bitmap code and explained with the patch series:

"Redundant data is used to enhance data fault tolerance, and the storage method for redundant data vary depending on the RAID levels. And it's important to maintain the consistency of redundant data.

Bitmap is used to record which data blocks have been synchronized and which ones need to be resynchronized or recovered. Each bit in the bitmap represents a segment of data in the array. When a bit is set, it indicates that the multiple redundant copies of that data segment may not be consistent. Data synchronization can be performed based on the bitmap after power failure or readding a disk. If there is no bitmap, a full disk synchronization is required.

Key Features

THANKS