

EXT4块分配可扩展性优化

李保坤
华为操作系统部

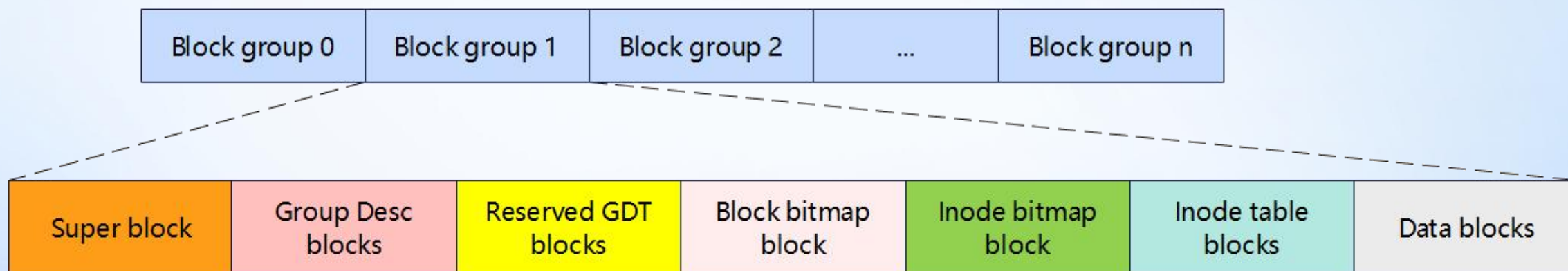


背景

众核高密场景通过 will-it-scale 测试内核可扩展性，发现 ext4 在 fallocate2 用例（8k追加 fallocate到1M，然后truncate 为 0）中 IOPS 随容器数量增加下滑严重。火焰图中可以观察到严重的锁竞争。

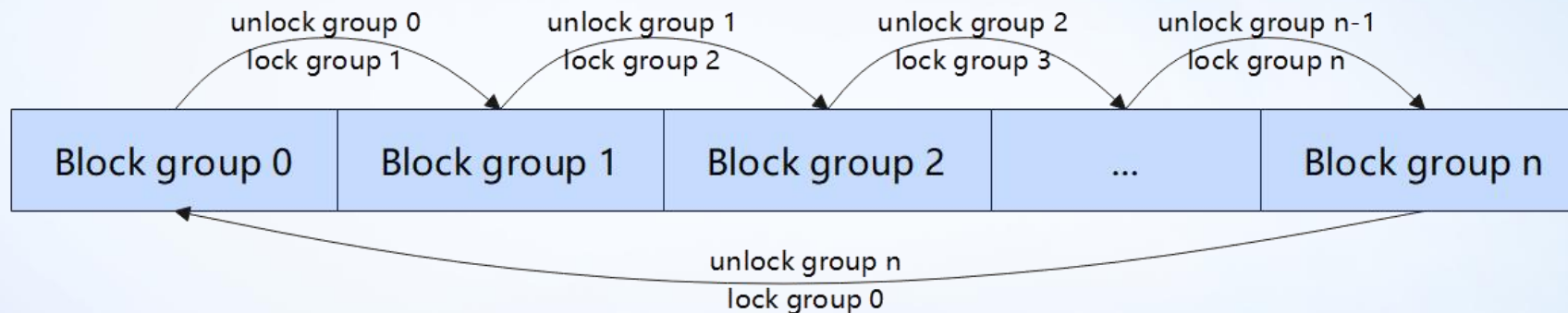


知识背景



- EXT4 文件系统把磁盘分为多个块组
- Inode 和 空闲块都是从块组中分配
- 块分配开始目标为 inode 所在组或已有extent所在组
- 目标分配失败则遍历所有块组，直到找到空闲块

块组线性遍历



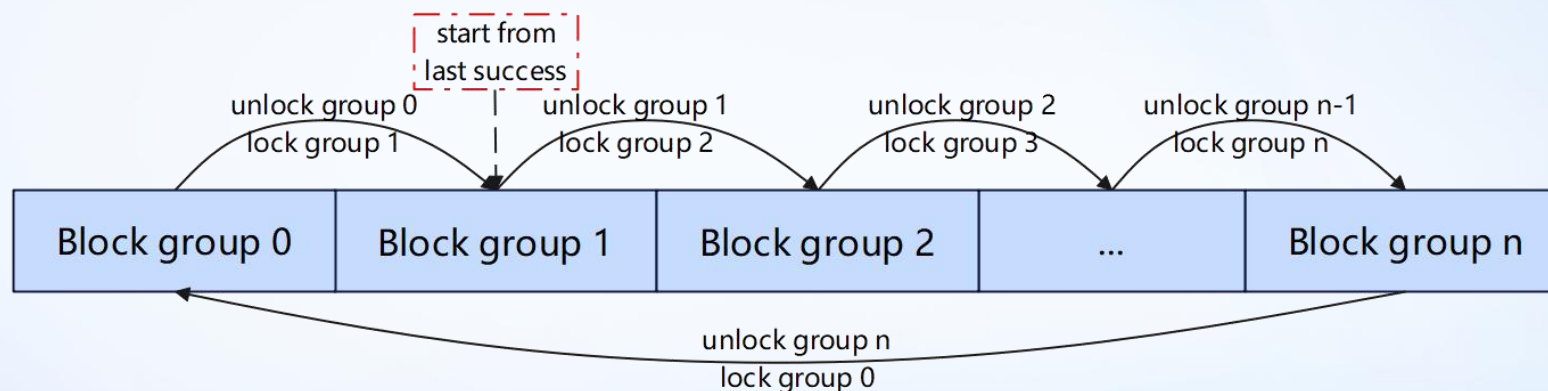
问题：

多个进程分配块遍历组时，先持锁的进程会阻塞后续进程。多个块组在同一个group中分配/释放块导致严重的 group 锁竞争。

优化：

遍历 group 时都是非目标块组，不是一定要在某个 group 中分配块，引入 trylock 机制来跳过繁忙组，这也可以避免一些不同进程后续块释放时的 group 锁竞争。

stream allocation



问题：

该特性在块分配前持有 `s_md_lock` 获取上次块分配成功的位置，设置为块组遍历的起始位置，并在块分配后持有 `s_md_lock` 更新该全局目标。当多个进程分配块时，**单一全局目标导致 `s_md_lock` 竞争严重。**

优化：

- 由于该全局目标只是 hint，因此**删除 `s_md_lock`** 改为 `READ_ONCE/WRITE_ONCE`。
- 将**全局目标拆分为多个**，按照 inode 号 hash 分桶，避免无效遍历。

mb_optimize_scan

max_free_order_lists/avg_fragment_size_lists

list_order_0_groups	group 37	group 12	group 45
list_order_1_groups	group 8	group 23	group 4
list_order_2_groups	group 50	group 29	group 6
list_order_3_groups	group 41	group 0	group 47
list_order_4_groups	group 15	group 33	group 9
list_order_5_groups	group 26	group 51	group 18
list_order_6_groups	group 2	group 39	group 31
list_order_7_groups	group 22	group 10	group 48
list_order_8_groups	group 7	group 35	group 14
list_order_9_groups	group 1	group 43	group 28
list_order_10_groups	group 19	group 5	group 32
list_order_11_groups	group 25	group 11	group 46
list_order_12_groups	group 21	group 40	group 17
list_order_13_groups	group 3	group 36	group 27

```
// 遍历次数不超过group的总数量
for (i = 0; i < ngroups; i++)
    for (i = ac_2order; i < max_order; i++)
        // 遍历 i 对应 order list
        spin_lock(order list);
        if (ext4_mb_good_group(bb_group)))
            break;
        spin_unlock(order list);
```

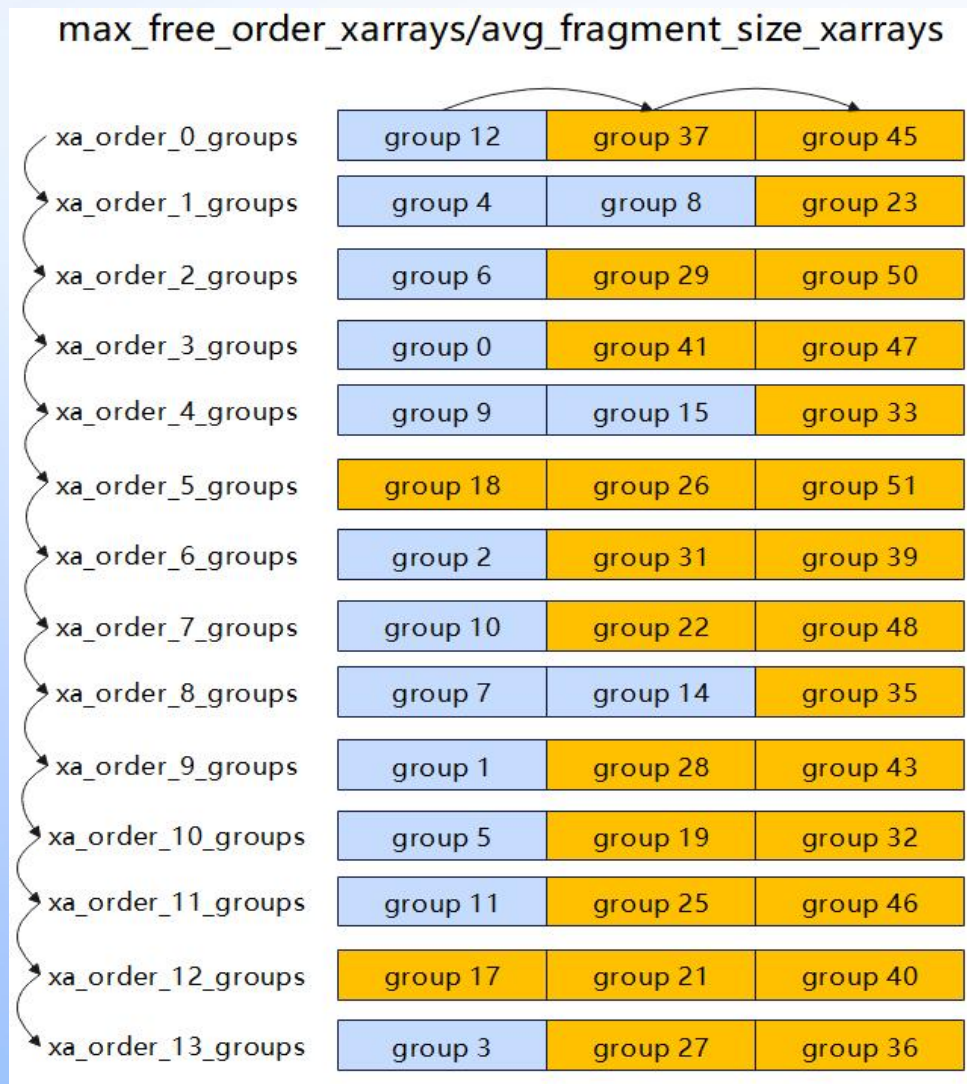
ext4_mb_load_buddy(grp); // 可能睡眠

```
if (!ext4_try_lock_group(group) || !ext4_mb_good_group(grp))
    ext4_unlock_group(group);
    continue;
// 扫描空闲块
ext4_mb_scan_group(grp)
ext4_unlock_group(group);
```

问题：

链表无序遍历导致**弹跳**；可能先从目标组之前的组中分配块，存在潜在的分配-释放、释放-释放**竞争**。

mb_optimize_scan



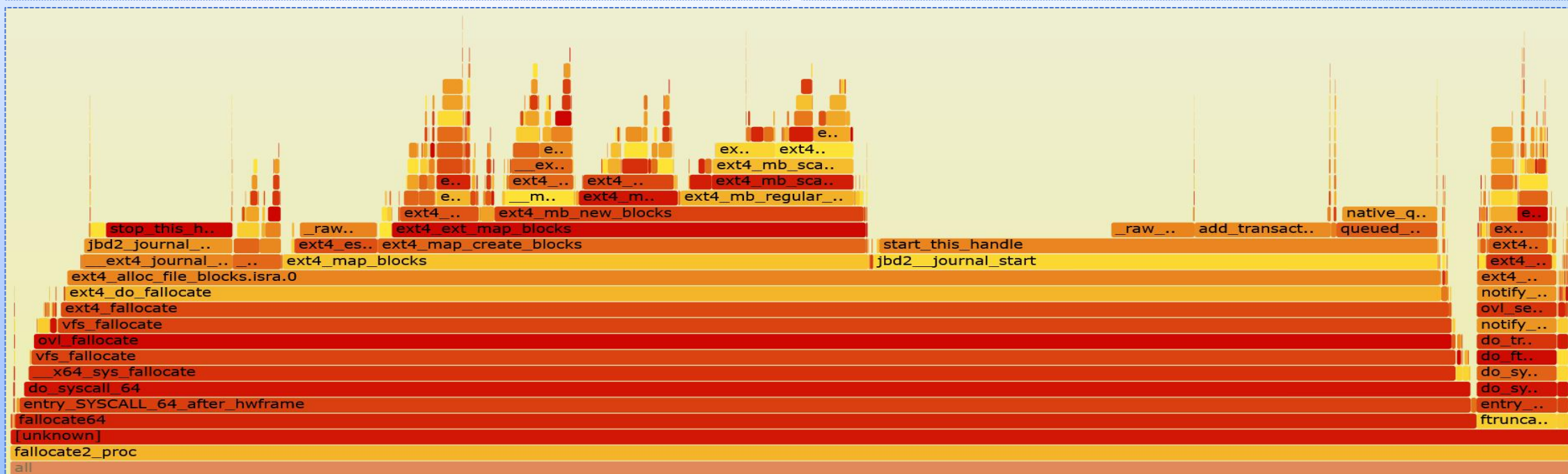
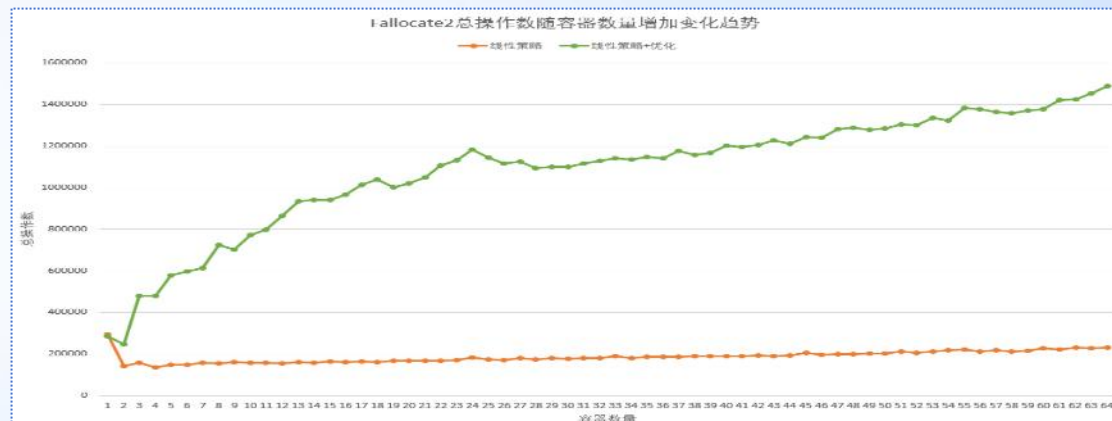
优化：

- 将无序的 list head 转换为有序的 xarry, group number 做为 xarray 的索引。有序队列即使遍历时放锁，也可以知道下一个 group，因而不会重复获取相同 group 导致弹跳。
- 单向遍历：先遍历大于 goal group 的块组再遍历小于 goal group 的块组。比如 goal group 是 16，先遍历所有list中黄色组再遍历所有list中蓝色组。这和线性遍历保持一致，可以减少竞争，并且对于 HDD 很友好。

优化效果

Test: Running will-it-scale/fallocate2 on CPU-bound containers.
Observation: Average fallocate operations per container per second.

CPU: Kunpeng 920	P80		P1	
Memory: 512GB				
960GB SSD (0.5GB/s)	base	patched	base	patched
mb_optimize_scan=0	2667	20049 (+651%)	314065	316724 (+0.8%)
mb_optimize_scan=1	2643	19342 (+631%)	316344	328324 (+3.7%)



碎片化影响评估

测试 200G EXT4 文件系统 fio 64 进程 4k 追加 fallocate 到 1G, 补丁对于碎片化的影响:

mb_optimize_scan	0		1	
	base	patched	base	patched
bw(MiB/s)	217	5718	217	5626
Avg. free extent size(KB)	1943732	1316212	1943732	1171208
Num. free extent	71	105	71	118
Avg. extents per file	261967	588	261973	570
Avg. size per extent(KB)	4	1780	4	1837
Fragmentation score	100	2	100	2

社区状态

- 社区状态：特性已合入Linux kernel v6.17

- <https://lore.kernel.org/linux-ext4/20250729033748.GA367490@mit.edu/>

- 补丁集链接：

- <https://lore.kernel.org/linux-ext4/20250714130327.1830534-1-libaokun1@huawei.com/T/>

- 社区News：

- [EXT4 Shows Wild Gains With Better Block Allocation Scalability In Linux 6.17](#)

- [Linux 6.17 With EXT4 Showing Some Nice Performance Improvements](#)

THANKS