CHINA LINUX KERNEL
中国Linux内核开发者大会

vivo

ZCACHE：异步文件压缩缓存管理方案

林泽生
vivo性能优化工程师

CHINA LINUX KERNEL
中国Linux内核开发者大会

vivo

目录

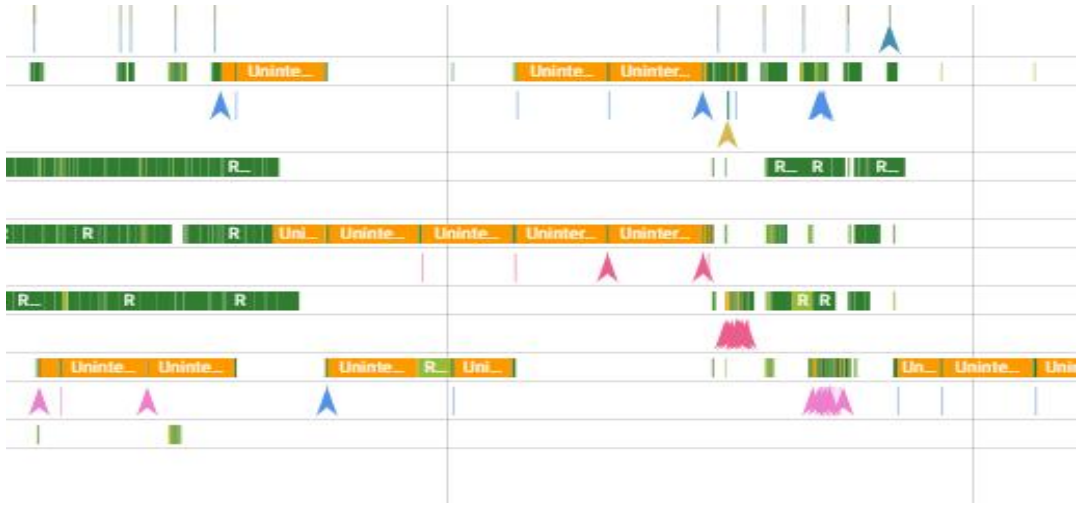
CONTENTS

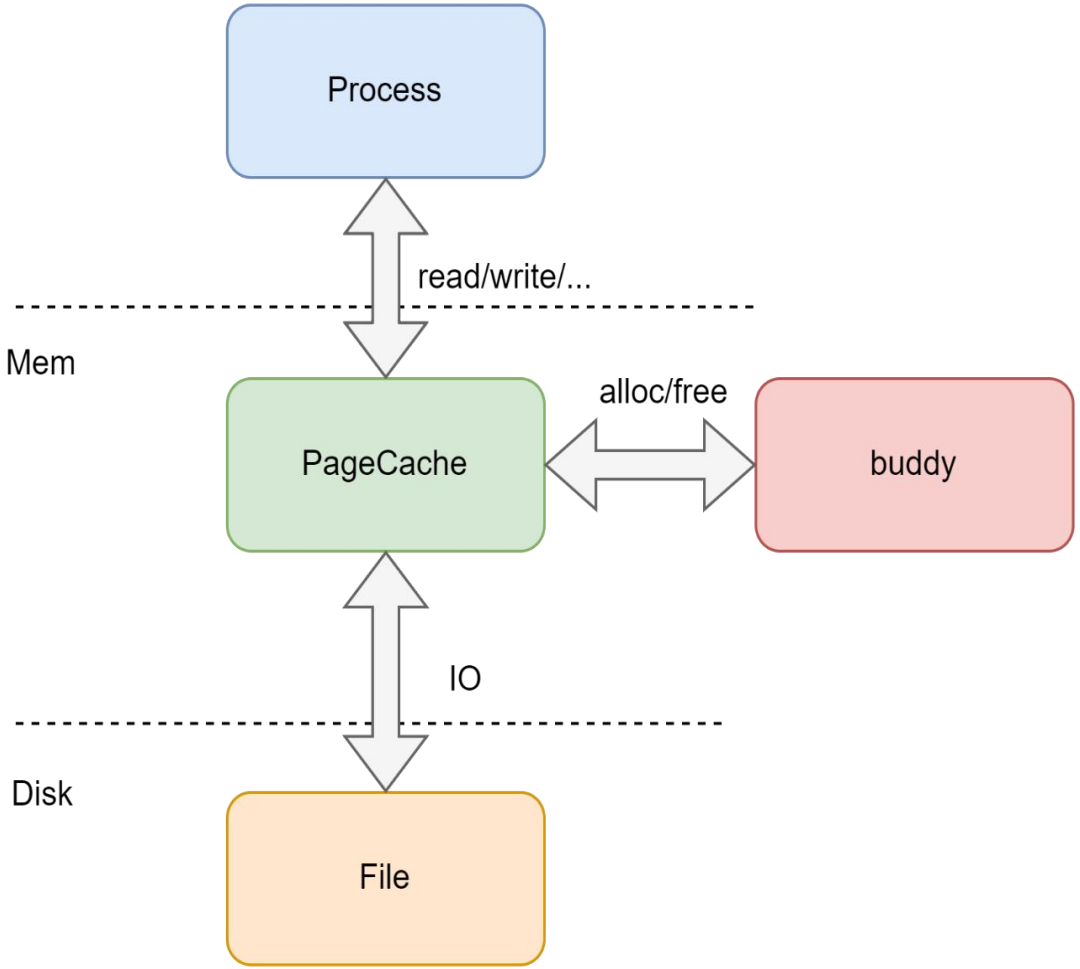
01 问题背景

问题背景 – 文件缓存现状（PageCache）

- 在文件读进来后以文件页的形式存在于PageCache中
- 内存紧张时回收文件页，待PageFault重新IO读回
- IO压力大时，出现瓶颈导致性能下降
- PageCache容量有限

（PageCache Size：文件页 Size = 1：1）





02 ZCACHE方案

ZCACHE方案 - ZCACHE（文件压缩）方案

● zcache

➢ 基于 cleancache 框架实现 ZCACHE 方案

➢ 回收时将干净文件页压缩缓存

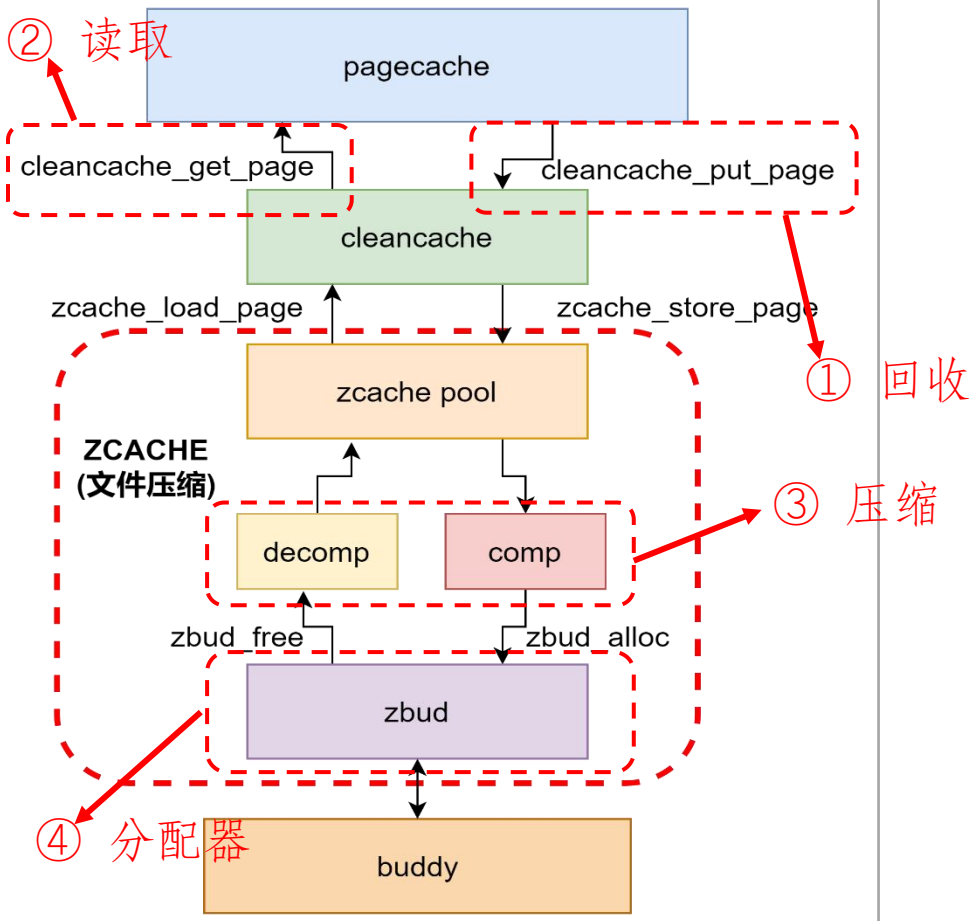
➢ 读取时优先从 zcache pool 读取

➢ 压缩缓存分配器采用 zbud 管理

● zcache 优点

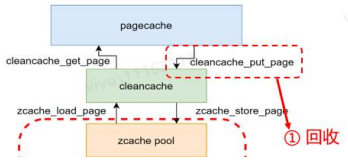
➢ IO压力大时，**降低IO吞吐量**

➢ **增大了 pagecache 容量**

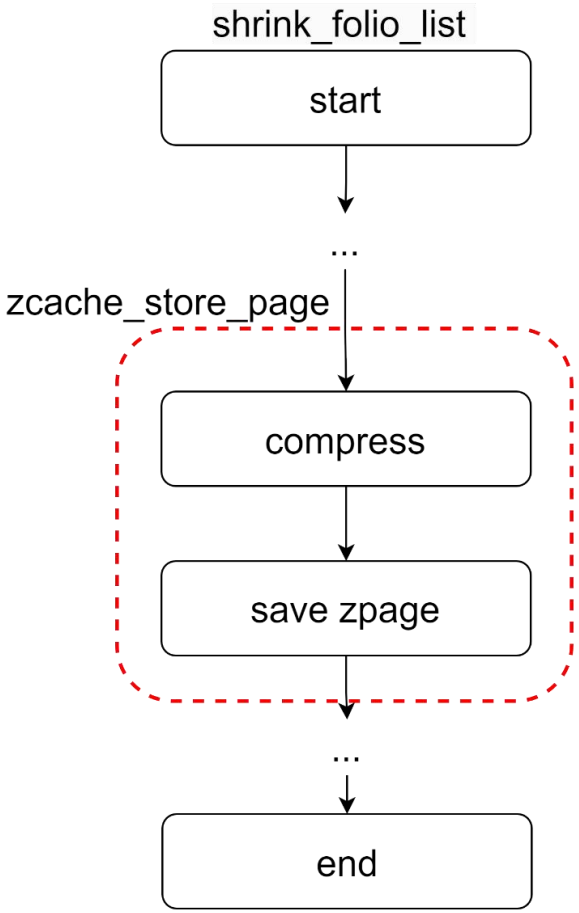


03 优化探索

优化探索 - 异步压缩



●问题分析



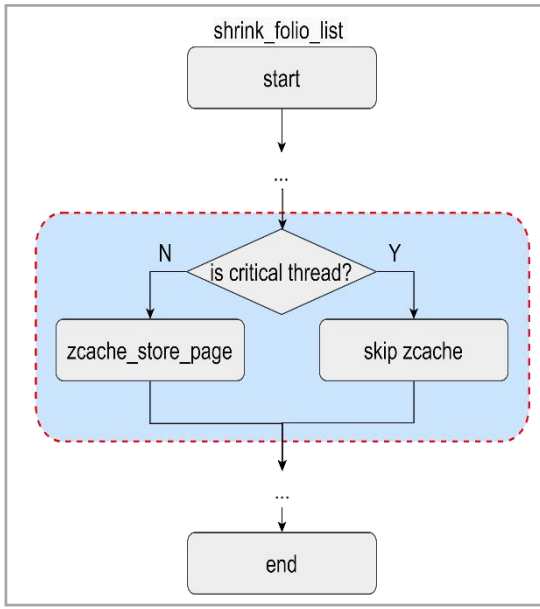
原生 zcache 压缩路径

●关键线程 slowpath 阶段要求内存快速回收，当前文件压缩路径会降低关键线程的内存回收速度

●解决方案

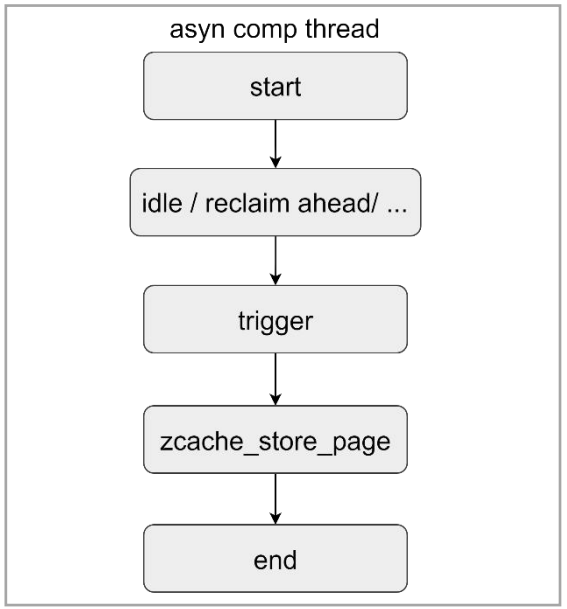
同步压缩

精细化同步压缩

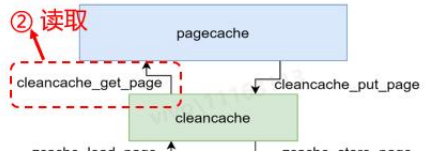


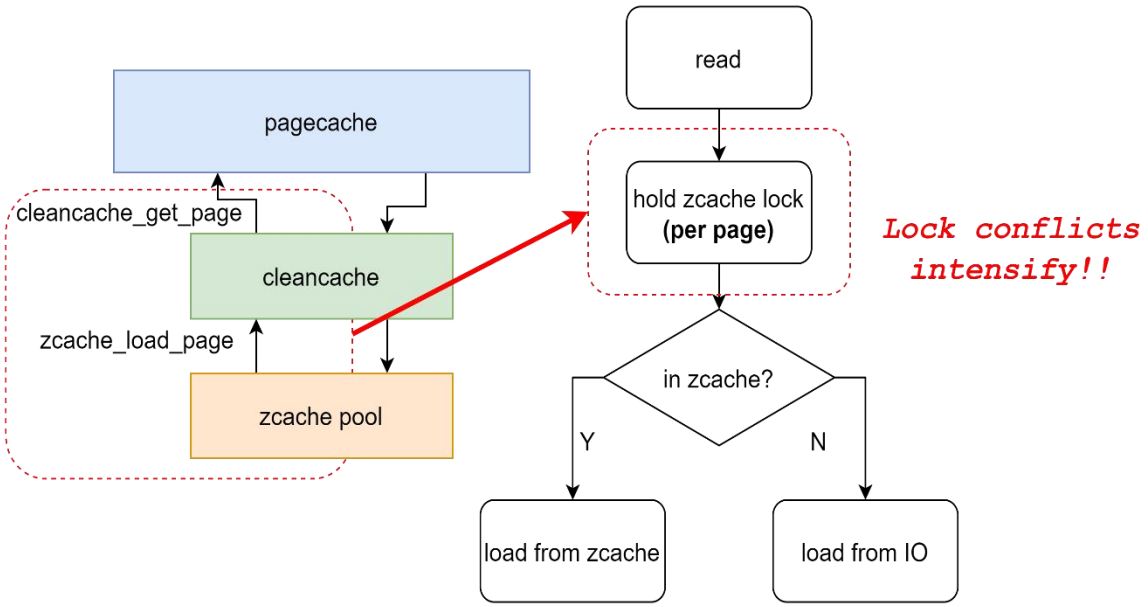
异步压缩

异步压缩



优化探索 - ZCACHE 标记

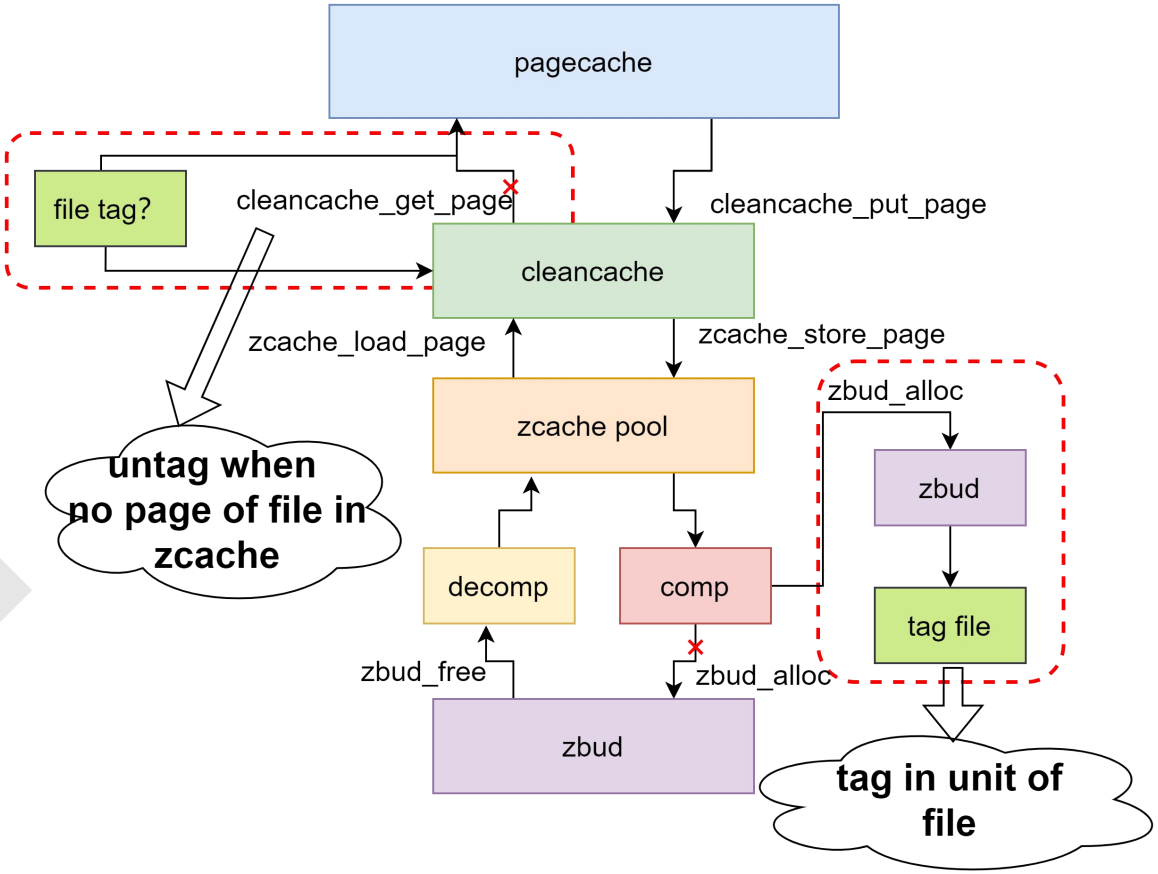




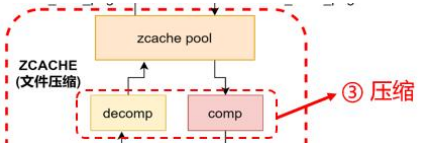
Lock conflicts intensify!!

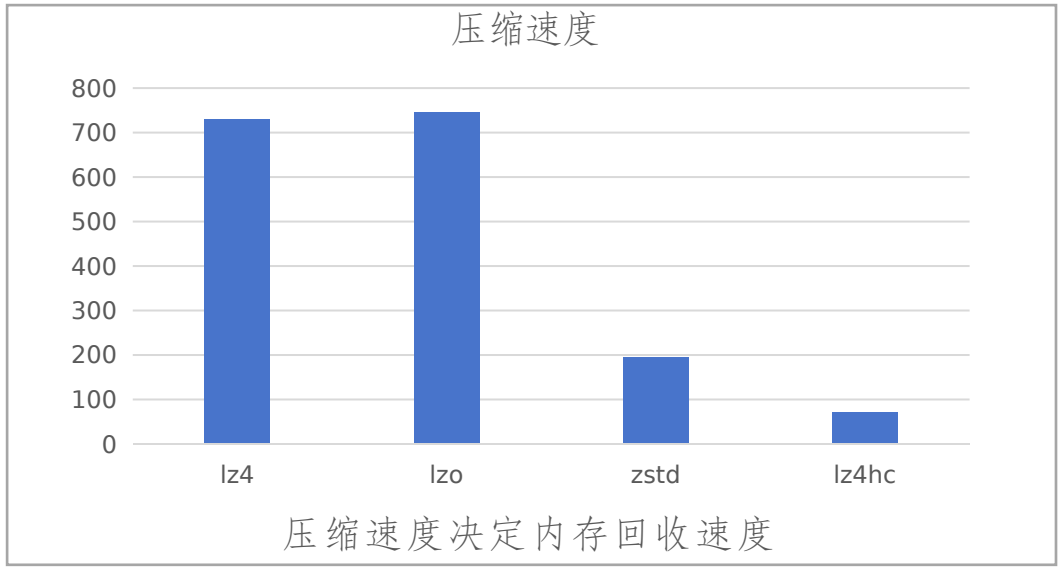
- 以文件为单位为例，zcache pool 打上标签
  - 减少查询时锁冲突
  - 提升系统读取效率

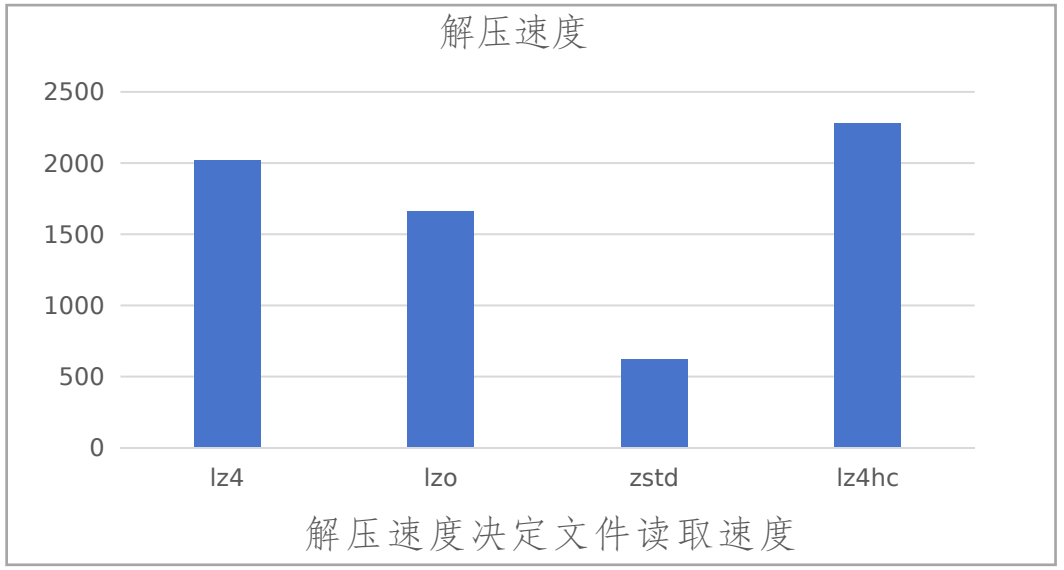
系统性能相较无 zcache 版本差

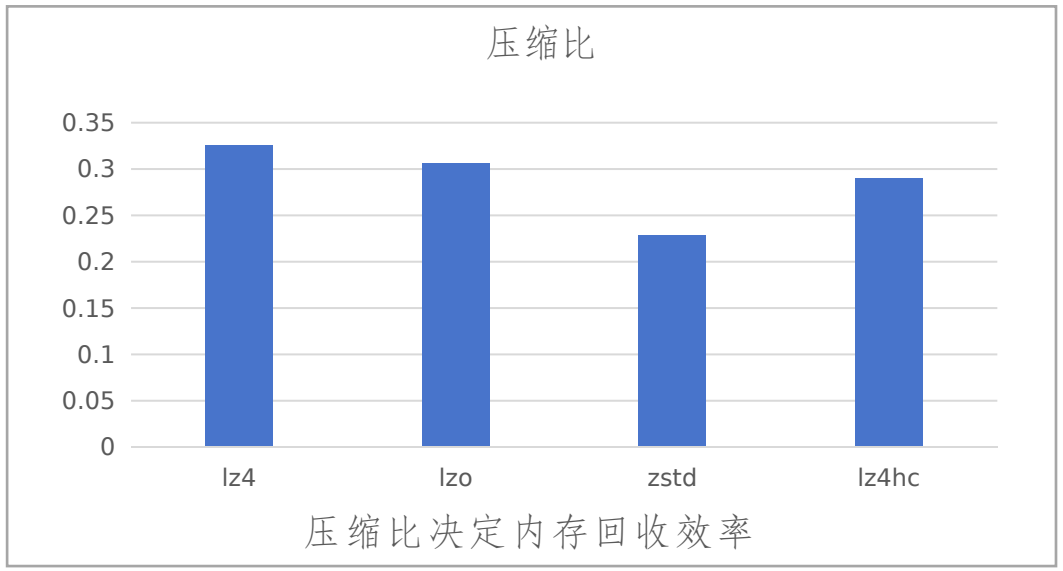


优化探索 - 压缩算法



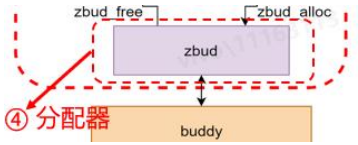






- 文件压缩尽可能满足 CPU 占用率小、内存回收效率高
- 原生 zcache 使用 lzo，相比 lz4，解压速度差距较大 =》CPU 占用率高

优化探索 - 内存分配器



- 除了压缩算法，**内存压缩率还与 空间利用率 有关**
- 空间利用率 与 **内存分配器有关**

- zcache 方案通过分配器**内部 lru 区分压缩缓存冷热**程度，在 zcache pool 满时需**淘汰旧缓存，存储新缓存**

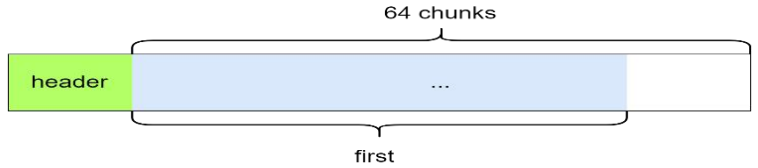
- 目前主线上适用于 zcache 的分配器如下：

不同分配器与 zcache 适配情况

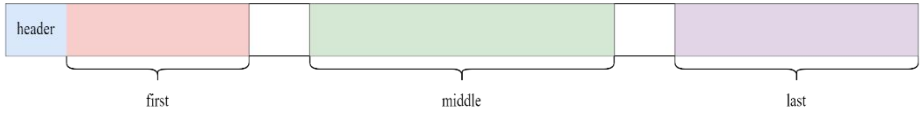
	zbud	z3fold	zsmalloc
zcache	√	√	×

- 分配器原理
  - ➢ zbud： 一个page支持最多存储两个 zpage
  - ➢ z3fold：一个page支持最多存储三个 zpage

➢ zbud 分配器原理图



➢ z3fold 分配器原理图



● zbud vs z3fold 结果

压缩情况对比

	zbud	z3fold
压缩前pages数量	108768	101296
压缩后pages数量	76711	76637
压缩比 = 压缩前/压缩后	1.4	1.3

● 分析

压缩率很低大约 94% 的pages压缩后的大小超过 1/3 * PAGE_SIZE

```
zcache_input_pagenr 1226105
zcache_exceed_half_pagenr 1151666
```

压缩后的大小 > 1/3 * PAGE_SIZE的页面比例

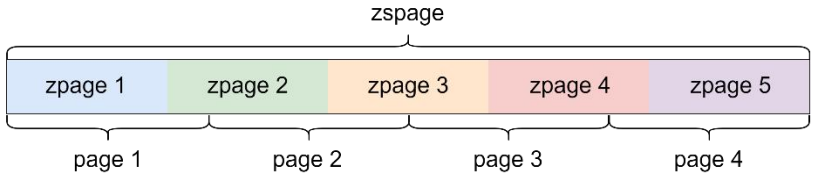
```
path:/app/~~riisBmWvClko7Y18R3i67Q==/com.ss.android.ugc.aweme-VpZbU_zjpkY-X5OaJKzWLg==/oat/arm64/base.vdex  size:3135
path:/app/~~riisBmWvClko7Y18R3i67Q==/com.ss.android.ugc.aweme-VpZbU_zjpkY-X5OaJKzWLg==/oat/arm64/base.vdex  size:3201
path:/app/~~riisBmWvClko7Y18R3i67Q==/com.ss.android.ugc.aweme-VpZbU_zjpkY-X5OaJKzWLg==/oat/arm64/base.vdex  size:3181
```

三方app文件页压缩后大小

很多资源类文件是经过编码的，文件页压缩后的大小普遍大于 1/3 * PAGE_SIZE。因此采用 zbud 或 z3fold 存储这类压缩文件页的话，会导致出现**较多内存碎片，压缩比较差**。

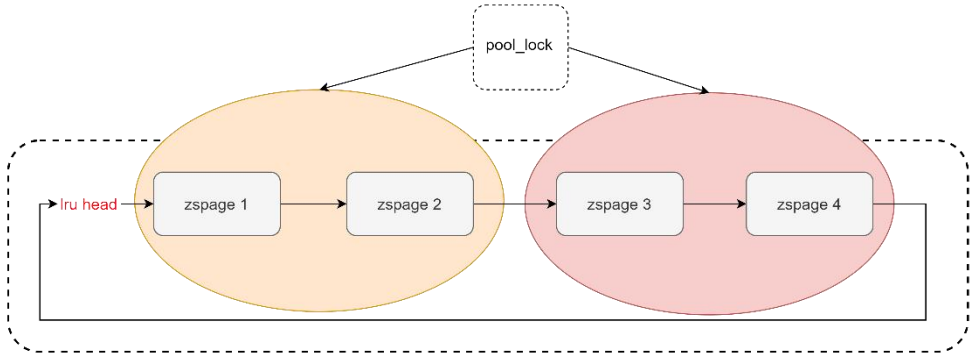
优化探索 - 内存分配器

● zsmalloc 允许跨页存储，增加空间利用率，减少内存碎片



zsmalloc 原理

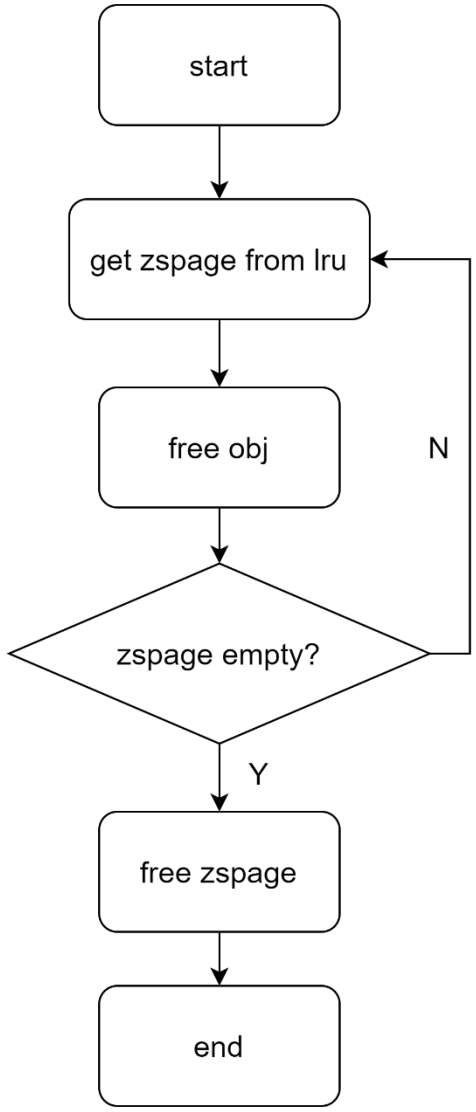
● 支持冷热机制 zsmalloc



● 压缩结果

压缩情况对比

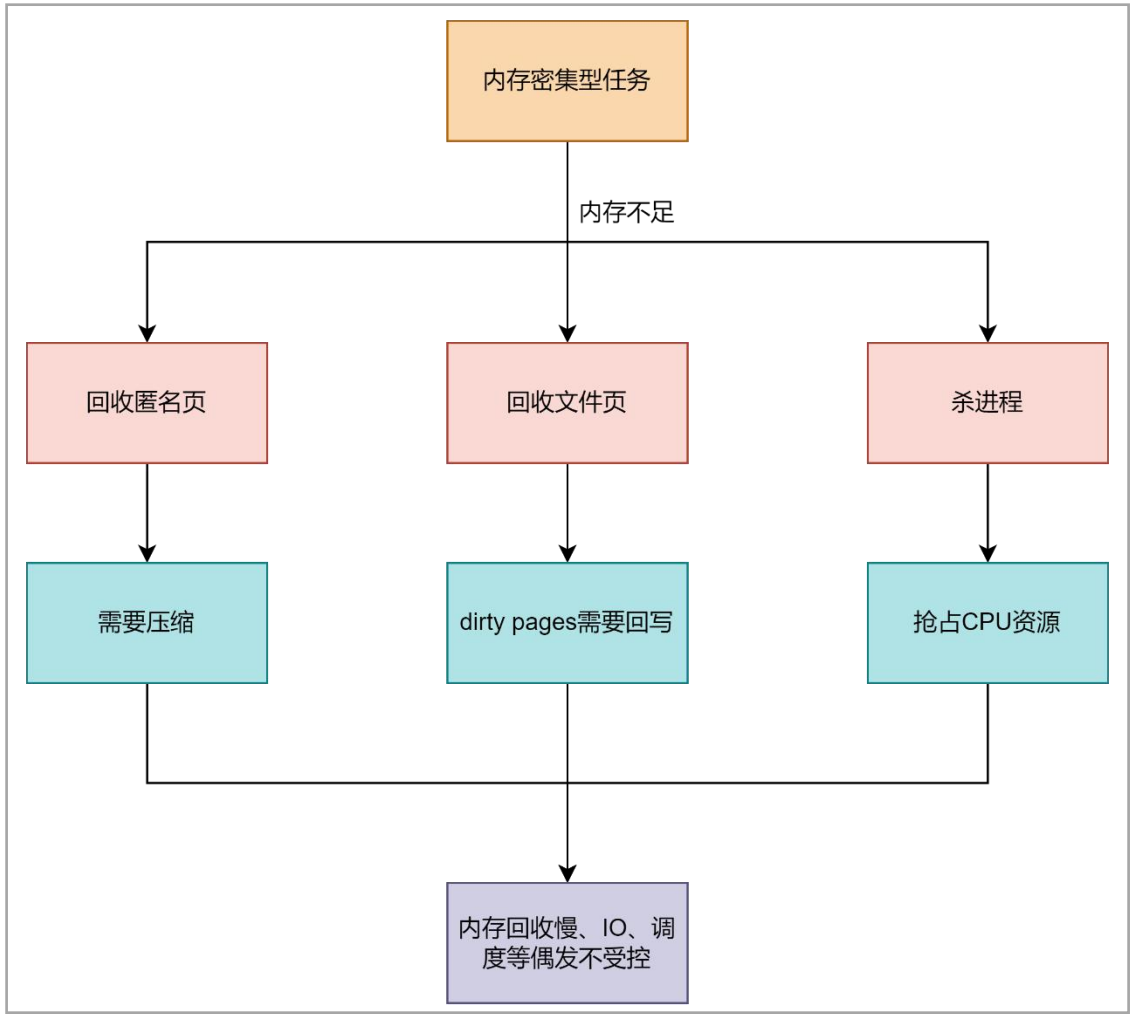
	zbud	zsmalloc
压缩前pages数量	108768	122016
压缩后pages数量	76711	61,678
压缩比	1.4	2.0



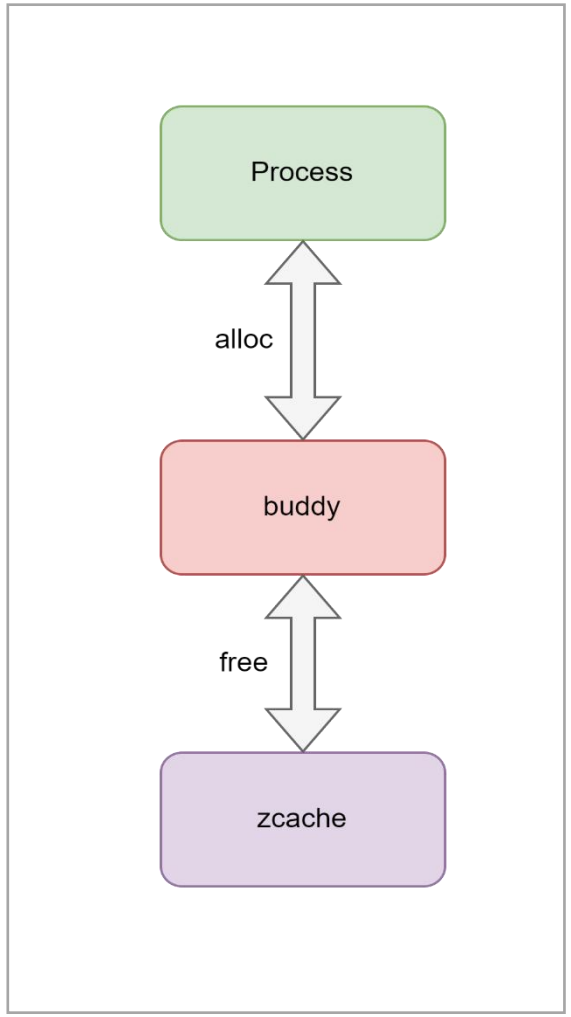
zsmalloc回收逻辑

优化探索 - 场景释放

- 内存密集性任务对内存回收速度敏感，内存回收速度慢会降低对应场景性能，影响用户体验
- zcache 存储干净文件页 =》 支持动态释放，**释放速度** 2.5G/S 左右



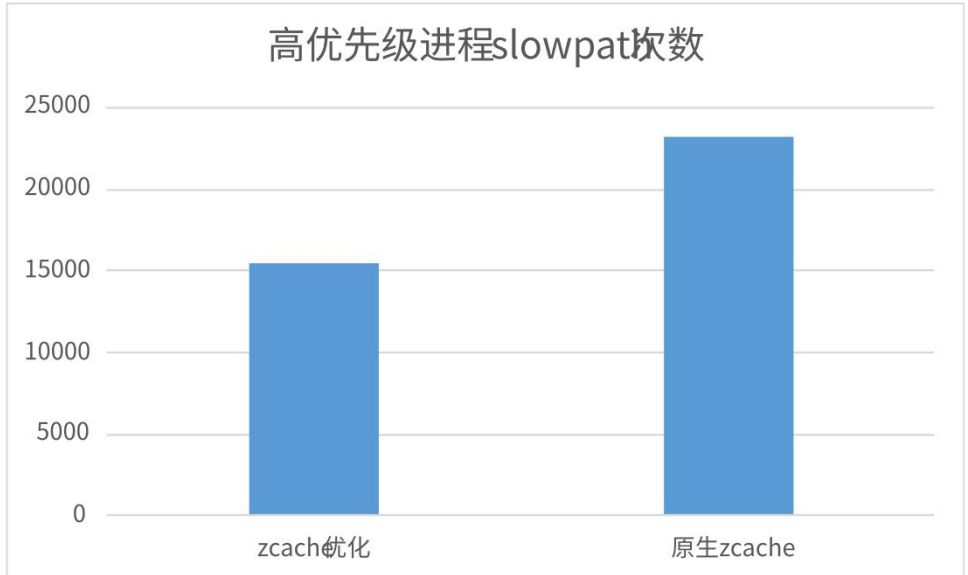
文件压缩缓存
存储干净页，
支持直接丢弃



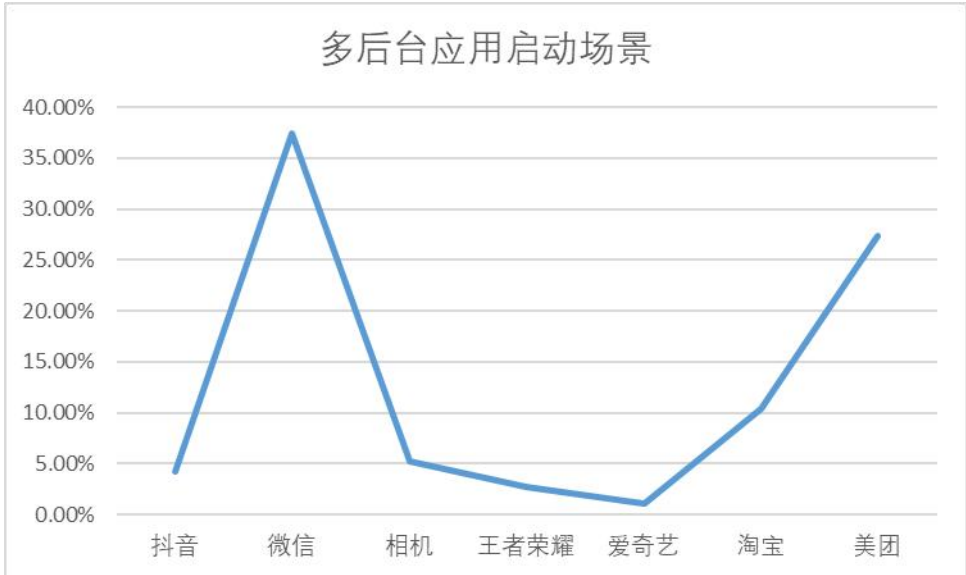
04 收益呈现

收益呈现

减少33.28% slowpath



应用启动速度平均优化了12.11%



THANKS