计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(4): 1063−1078, 2026

基于热点文件页交换和压缩预测的移动设备内存优化方法

龙家璇[1] 肖春华[1,2] 孙韬宁[1] 熊浩宇[1] 左笑寒[1] 吴 林[1] 陈延科[1]
[1](重庆大学计算机学院 重庆 401331)
[2](信息物理社会可信服务计算教育部重点实验室(重庆大学) 重庆 401331)
(longjiaxuan@cqu.edu.cn)

Memory Optimization Method for Mobile Devices Based on Hot File Page Swapping and Compression Prediction

Long Jiaxuan[1], Xiao Chunhua[1,2], Sun Taoning[1], Xiong Haoyu[1], Zuo Xiaohan[1], Wu Lin[1], and Chen Yanke[1]
[1](*College of Computer Science*, *Chongqing University*, *Chongqing* 401331)
[2](*Key Laboratory of Dependable Service Computing in Cyber Physical Society* (*Chongqing University*), *Ministry of Education*, *Chongqing* 401331)

Abstract The development of the mobile application ecosystem has intensified memory pressure, while traditional physical expansion solutions are limited. As a result, compressed memory has become the mainstream technology for memory expansion. However, existing compressed memory systems still suffer from low file page performance, high compression overhead, and suboptimal storage resource utilization, negatively impacting overall system performance. To address these issues, we propose two optimization frameworks: EC-MemOpt and APC-MSO. EC-MemOpt improves file page I/O performance and storage resource utilization by migrating frequently accessed file pages to compressed memory and leveraging compression prediction techniques. APC-MSO optimizes the compression swap strategy for anonymous pages, reducing unnecessary compression computation overhead. Furthermore, a collaborative management architecture for hybrid memory pages (EC-MemOpt + APC-MSO) is designed to achieve differentiated partition management. Compared with the QCOM 8350R kernel maintained by OnePlus, the proposed architecture improves application switching speed and space utilization by 10.5% and 9.8%, respectively. During the experiments, large volumes of anonymous and file pages are tested under diverse workload conditions. The results verify that the hybrid framework maintains stable performance across varying memory access patterns. Experimental results demonstrate that the proposed optimization solutions significantly enhance the overall performance of compressed memory systems in mobile environments.

Key words mobile devices; compressed memory; compression prediction; file page optimization; anonymous page optimization

摘 要 移动设备应用生态的发展导致内存压力加剧,传统物理扩容方案受限,压缩内存成为主流内存扩容技术。然而,当前压缩内存系统仍存在文件页性能低下、压缩开销较高及存储资源利用率方面不足的问题,对系统性能产生了负面影响。针对上述问题,提出 EC-MemOpt(efficient compression-based memory management framework for hot file pages)与 APC-MSO(anonymous page compression-based memory swapping optimization method)这 2 种优化框架。EC-MemOpt 通过迁移高频访问的文件页至压缩内存,并结合压缩性预测技术,有效提高了文件页 I/O 性能和存储资源利用率。APC-MSO 针对匿名页特性,优化了压缩交

收稿日期: 2025-03-24; **修回日期**: 2025-11-03
通信作者: 肖春华(xiaochunhua@cqu.edu.cn)

换策略,从而减少无效压缩计算开销。最后,研究构建面向混合内存页的协同管理架构(EC-MemOpt+APC-MSO),实现差异化分区管理。与 OnePlus 维护的 QCOM 8350R 内核相比,该架构在应用切换速率和空间利用率方面分别提升 10.5 个百分点和 9.8 个百分点。实验结果表明,提出的优化方法能显著改善移动端内存压缩系统的综合性能。

关键词 移动设备;压缩内存;压缩预测;文件页优化;匿名页优化

中图法分类号 TP316.81

DOI: 10.7544/issn1000-1239.202550229 **CSTR:** 32373.14.issn1000-1239.202550229

随着移动互联网技术的快速发展,手机已成为用户日常生产生活中必不可少的设备[1]。当前移动设备上的应用呈现出两大显著特征:应用并发运行需求持续增长,典型用户平均每日启动 9 个或 10 个应用程序以应对多样化场景需求[2-4];应用功能复杂化,单个应用内存需求可达到数百兆字节。应用负载特征与移动设备物理内存容量不足的矛盾日益增加。尽管各大手机厂商的旗舰机型扩展至 12 GB 以上内存容量,但中低端配置的设备的内存仍普遍维持在 4~8 GB,难以完全满足多任务场景下的内存需求。

传统解决方案通过增加物理内存的方式缓解资源压力,但受制于硬件成本控制与设备形态限制,以及市场倾向于低价格设备的趋势,单纯依赖硬件扩容难以实现可持续优化。因此基于软件层面的内存管理优化成为提升系统效能的关键突破口。当前 Linux 内存管理机制通过压缩内存技术缓解内存不足的问题,即将被淘汰的页面进行压缩,并将压缩数据存储至压缩内存 ZRAM(compressed random access memory)。当再次需要这些页面时,系统能够快速解压并取回,从而减少磁盘 I/O 操作的消耗。

在压缩内存技术的研究过程中,减少压缩带来的开销是一个重要的技术瓶颈。Tremaine 等人[5]提出了专用于内存压缩的硬件组件。这些硬件组件通常放置在主存储器和处理器之间。虽然这些专用组件可以有效压缩主存,且对性能的影响可以忽略不计,但由于其能耗和制造成本,很难适用于移动设备。

除了通过设计硬件组件提升压缩性能外,也有研究在软件层面进行优化。例如,Li 等人[6]通过并行压缩提高压缩速率。周新明[7]使用动态调整压缩页面粒度的方法来减少压缩开销。Kim 等人[8]提出了一种针对移动设备的应用程序的感知交换方法,该方法综合页面粒度与进程粒度来筛选淘汰页面。Gao 等人[9]将文件数据压缩为多个固定大小的块以缓解读放大问题,他们设计了一种内存高效解压方案以减少解压缩过程的内存使用。邱彬等人[10]根据虚拟机内存信息特性,结合多种分页压缩算法和多线程技术加速虚拟机迁移过程。

还有研究从预测角度入手,例如,Harnik 等人[11]试图预测大型存储系统的压缩和去重率,然而,移动设备通常为小型设备,其压缩任务主要集中在较小规模的数据集上。Son 等人[12]通过预测并预加载关键内存页,减少应用重启时的 I/O 开销,从而在内存压力下有效降低应用切换延迟。有研究[13-14]将信息熵理论引入页面压缩性预测领域,发现页面信息熵能够作为衡量页面数据压缩潜力的重要指标。然而,随着压缩算法的不断优化与加速,计算页面信息熵的时间可能会超过实际压缩页面的时间。

为了应对信息熵计算开销过大的问题,Song 等人[15]提出了基于采样的多阶段组合页面压缩预测方法 PGPI(prediction-guided performance improvement)。该方法设计了一种多阶段预测器,以较低的计算成本过滤出不可压缩页面,显著减少了无效的压缩操作。实验结果表明,该研究提出的预测器预测准确率达到了 97.4%,应用启动时间平均减少了 29.5%。

为了解决传统内存压缩交换缓存方案 ZSWAP(compressed swap cache)仅支持存储匿名页面的局限性问题,Kim 等人[16]提出了增强压缩交换方案 EZSWAP(enhanced compressed swap cache)。该方案同时支持匿名页和干净的文件页换出到压缩内存。EZSWAP 基于页面信息熵理论实现了压缩率预测技术和成本收益模型,能够选择性地存储高收益页面,从而提高了移动设备的内存利用率和系统响应能力。然而本研究发现,EZSWAP 接口的平均调用时长为 3 000 ns,而压缩单个页面的平均时间开销为 6 400 ns,EZSWAP 每次判断页面是否可压缩的开销接近于直接压缩开销的一半。并且,由于换出文件页的数量是匿名页的数倍,接口调用次数较多,系统无法承受如此大的计算开销,在实际的运行过程中,移动设备应用可能会发生崩溃现象。因此,优化接口调用效率将是提升 EZSWAP 实用性和系统性能的关键方向。

从研究现状来看，软件层面仍存在较大的优化空间和潜力。首先，内核级设计约束导致压缩空间存在 3 重缺陷：文件页交换时产生的 I/O 延迟较高，高频访问文件页的反复换入换出不可避免地造成性能损失；其次，现有机制缺乏压缩性预测模块，其会对进入的页面执行无差别的压缩操作，若进入压缩内存空间的页面不可压缩，不仅不会节约内存，还会带来不必要的压缩开销；再者，大容量压缩内存配置（超过 2 GB）下空间利用率不足问题突出，监控数据显示超过 50% 的压缩空间处于闲置状态。

因此，本研究针对上述问题，基于对大量页面交换规律的分析，设计了一种适用于 Android 移动设备的内存优化方法。该方法支持将热点文件页存储到压缩内存空间中，提升了文件页的使用性能和压缩内存的利用率，并对匿名页和热点文件页进行压缩预测以减少压缩开销。

1 研究动机

在实际应用中，压缩内存技术的使用机制暴露出了一些问题，具体包括 3 点。

1）文件页性能低效。由于 Linux 内核代码的设计，目前大多数 Android 移动设备上的压缩内存空间仅支持匿名页的存储，而不支持文件页的容纳。文件页缺失时可以从闪存存储空间中找到。然而，这种设计也带来了显著的缺陷：从闪存存储中读取文件页的速度较慢，可能会对用户体验产生明显的负面影响。实验显示，经过模拟用户行为充分使用手机后，并非所有文件页都是低访问频率的页面，在访问次数大于 1 次的文件页中，有超过 22% 的文件页在使用过程中访问超过了 10 次。如果高频页面频繁地换入换出，将不可避免地导致系统性能的显著下降。

2）页面压缩开销较大。当前的压缩内存空间并不能有效区分页面的压缩性。无论进入的页面是否具有压缩潜力，系统都会进行无差别的压缩处理。若进入压缩内存空间的页面是不可压缩页面，不仅无法节约内存，而且会带来额外的计算开销。

3）存储系统资源利用率低下。压缩内存空间的利用率未能得到充分发挥。本研究在 OnePlus 9 Pro 手机配置下开展实验，系统采用 Linux 内核中的最近最少使用（least recently used, LRU）链表页面淘汰策略。该策略依据页面访问频率将其划分为活跃与非活跃 2 类，优先从非活跃链表中选择冷页面进行淘汰。同时，基于对主流移动设备内存的调研，实验将系统 ZRAM 大小设定为 4 GB，该设置在当前主流移动设备中具有代表性，能够较好地平衡压缩性能与系统资源开销。为模拟接近真实的多任务运行场景，本研究依次启动 2 轮 12 个目标应用程序，应用程序名称如表 1 所示。随后选定主应用（抖音）运行于前台，并使用 trace 工具记录 ZRAM 区域占用情况。图 1 展示了 ZRAM 占用随应用使用的变化示意图，图 1 的横坐标前 24 个点对应 12 个应用的 2 次循环启动的采样时间点，后 25 个点记录主进程持续运行阶段的采样时间点。可以发现，尽管启动了大量应用构建了内存压力的环境，大量页面被持续淘汰，但在后续

Table 1 Application List

表 1 应用程序列表

编号	应用	编号	应用
1	百度	7	时钟
2	便签	8	天气
3	地图	9	图库
4	电话	10	微博
5	计算器	11	文件管理
6	系统设置	12	相册

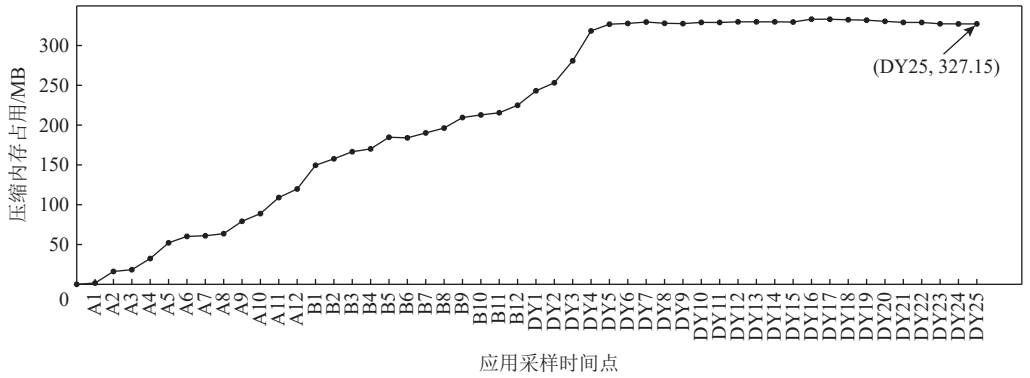


Fig. 1 Compressed memory space occupancy varies with application usage

图 1 压缩内存空间占用随应用使用的变化

运行过程中，ZRAM 的占用空间趋于稳定。考虑到 ZRAM 中其他数据结构的大小，ZRAM 的实际占用空间不足 1 GB，压缩内存空间利用率不足 25%，有严重的空间浪费问题。

理论上，动态调整压缩区容量可以根据系统运行状态(如内存压力、页面压缩效率等)进行灵活扩展或收缩，以提升系统压缩内存的利用率。然而在资源受限的移动设备环境中，该机制面临诸多挑战。首先，动态调整依赖对系统状态(如内存压力、页面压缩比等)的持续监控，这会带来不可忽视的 CPU 与内存开销。其次，自适应策略往往与用户使用习惯密切相关，缺乏通用标准，难以在不同场景间高效泛化。此外，压缩区容量调整通常伴随内存重分配或页面迁移，可能引发显著的瞬时延迟，影响应用响应。更重要的是，负载波动具有高度动态性，容量刚被调整后即可能失效，而若调整策略响应过慢，又无法有效提升性能，甚至加剧系统不稳定性。综上所述，本研究采用静态压缩内存容量的配置展开移动设备内存优化方法。

2 页面特性分析

2.1 文件页内存迁移与压缩性的分析

为了验证文件页内存优化方法的设计思路，本研究设计了一套白盒 trace 工具，以监测文件页在页面链表中以及文件页回收的迁移规律。

1) 文件页迁移模式与驻留时间特性分析

在 Linux 内核中，LRU 链表用于管理页缓存，实现高效的内存回收机制。内核采用双链表(活跃链表与非活跃链表)结构，依据页面的访问频率动态调整页面在链表中位置。当内存压力增加时，内核优先淘汰非活跃链表中的页面，从而提高缓存命中率并减少 I/O 开销。这种设计确保了系统在资源受限环境下的稳定性与性能。

通过模拟用户行为习惯，开启多个后台应用，并长时间使用 1 个前台应用，本研究设计的 trace 工具获取了 205 万个文件页在内存的 LRU 链表中迁移和驻留情况。在排除 trace 异常打印的无效记录的基础上，本研究对文件页行为进行分析。图 2 展示了在内存中文件页在 LRU 链表中的迁移行为统计，图 2 中迁移行为表示 1 个页面迁移前后所处的位置或状态。其中，“非活跃”表示处于非活跃链表、“活跃”表示处于活跃链表、“回收”表示页面由于内存紧张而被回收、“释放”表示页面由于没有进程映射而被释放。可以发现，内存中文件页面最多的迁移行为是页面回收，占比高达 88.72%。内存中文件页面迁移至活跃链表的比例较低，仅有 5.27%。也有少量文件页因为缺少进程映射而被释放，占比 5.01%。

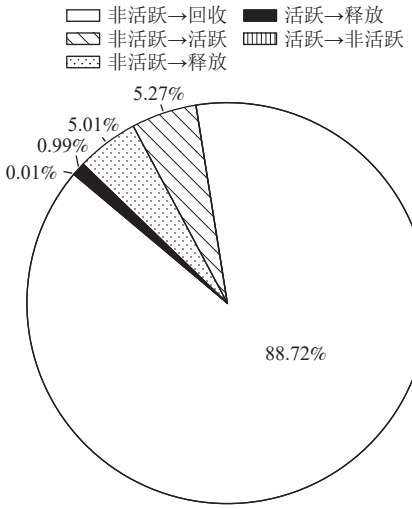


Fig. 2 Proportion of file page migration behavior in memory

图 2 内存中文件页迁移行为占比

本研究统计了文件页在 LRU 链表的驻留时间。处于非活跃链表的文件页，被回收前驻留时间平均为 6.37 s，被释放前驻留时间平均为 2.12 s。在被迁移至活跃链表之前，文件页在非活跃链表驻留时间平均为 1.1 s。处于活跃链表的文件页，在被回收前驻留时间平均为 6.6 s，被释放前驻留时间平均为 3.68 s。

2)文件页压缩性与时间局部性关联分析

本研究将文件页的可压缩比阈值设定为 0.75，即若压缩后的文件页大小低于该值，为可压缩页面，否则为不可压缩页面。图 3 展示了连续被淘汰的 1 000 个文件页的压缩性分布，其中黑色区域表示可压缩页面，而白色区域则表示不可压缩页面。可以发现，被淘汰的文件页在压缩性方面表现出显著的时间局部性。例如，连续淘汰的 3 个页面 A, B, C，若页面 A 和页面 B 均为不可压缩页面，则页面 C 大概率也为不可压缩页面。

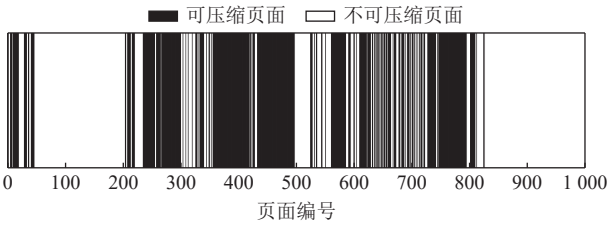


Fig. 3 Distribution of compressibility for consecutively evicted file pages

图 3 连续被淘汰的文件页面压缩性分布

2.2 匿名页面特性与压缩性的分析

1)页面平均字节值与页面压缩性的关系

在计算机体系结构中，字节是最小的可寻址内存单元，每个字节的值为0~255。在内核中，一个页面的大小通常为4 KB，页面平均字节值的计算方法如式(1)所示。

$$\mu = \frac{1}{n}\sum_{i=0}^{n} P_i , \tag{1}$$

其中 P_i 代表页面中第 i 个字节的数值，n 为页面的总字节数，n=4 096。

为了模拟用户行为，我们依次运行游戏类应用“王者荣耀”、视频类应用“哔哩哔哩”、文字博客类应用“知乎”和通信类应用“Tim”这4个常见应用，对产生的约44万个匿名页进行统计。本研究将匿名页面可压缩比阈值定义为0.875，基于上述统计进一步分析页面平均字节值与压缩率存在的关联性。图4展示了匿名页平均字节值与压缩率的关系散点图，每个点代表1个匿名页，落在顶层灰色区域的页面即为不可压缩页面，落在下方白色区域的页面即为可压缩页面，页面的聚集程度越高，颜色越深。可以发现，可压缩页匿名面的平均字节值通常小于30或大于210，这一特征可作为判断页面压缩性的依据之一。

2)页面字节值方差与页面压缩性的关系

页面的字节值方差也可用于衡量页面数据的整体分布情况，其计算方法如式(2)所示。

$$\sigma^2 = \frac{1}{n}\sum_{i=0}^{n} (P_i^2 - \mu^2) , \tag{2}$$

其中 P_i 代表页面中第 i 个字节的数值，取值范围为0~255，μ表示页面平均字节值，n 为页面的总字节数，n=4 096。

本研究采用上述44万个匿名页作为测试样本，并对这些匿名页的字节值方差特征进行分析。图5展示了匿名页平均字节值方差与压缩率的关系散点图。通过对统计数据进行分析可以发现，当匿名页的字节值方差小于2 000或大于15 000时，页面较高概率为可压缩页面，仅有少量页面不符合此规律。

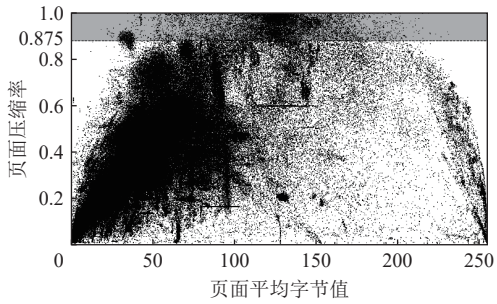


Fig. 4 Scatter plot of the relationship between average byte value of anonymous pages and compression ratio

图4 匿名页平均字节值与压缩率关系散点图

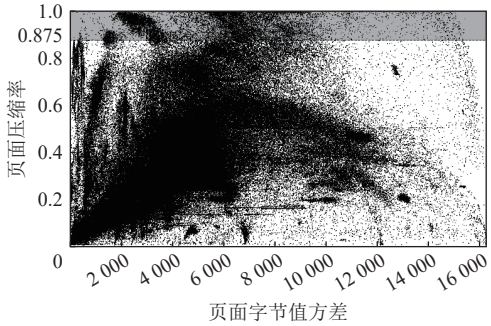
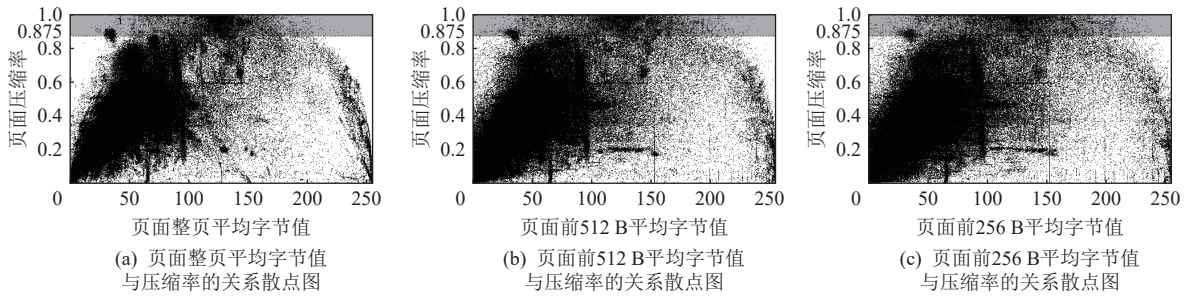


Fig. 5 Scatter plot of the relationship between byte value variance of anonymous pages and compression ratio

图5 匿名页面字节值方差与压缩率关系散点图

3)局部页面统计特征与压缩性关联分析

为了研究页面前512 B与整页压缩性之间的关系，本研究使用上述44万个匿名页面作为测试数据，并对页面进行局部采样，样本大小为每个页面的前512 B。根据图6可以观察到，页面整页平均字节值(图6(a))与页面前512 B的平均字节值(图6(b))与页面压缩性之间的关系散点图分布形态较为相似。具体来说，当页面的前512 B的平均字节值小于30或大于210时，页面通常可压缩，仅有少量稀疏的散点未能遵循该规律，与整页的分布规律一致。



(a) 页面整页平均字节值与压缩率的关系散点图

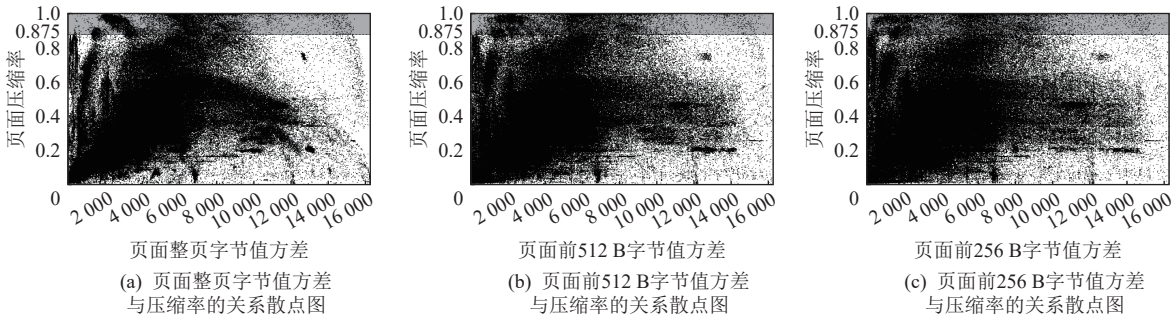
(b) 页面前512 B平均字节值与压缩率的关系散点图

(c) 页面前256 B平均字节值与压缩率的关系散点图

Fig. 6 Scatter plot of the relationship between the average byte value of page and compression ratio for different samples

图6 不同样本的页面平均字节值与压缩率的关系散点图

此外，本研究还对比了这些页面的整页字节值方差和前 512 B 的字节值方差与页面压缩性之间的关系。如图 7 所示，页面整页字节值方差(图 7(a))与页面前 512 B 的字节值方差(图 7(b))与页面压缩性之间的散点图分布形态较为相似。具体来说，在页面前 512 B 的字节值方差小于 2 000 或大于 15 000 时，页面通常可压缩，仅有少量稀疏的散点未能遵循这一规律，与整页的分布规律一致。



(a) 页面整页字节值方差与压缩率的关系散点图　(b) 页面前512 B字节值方差与压缩率的关系散点图　(c) 页面前256 B字节值方差与压缩率的关系散点图

Fig. 7　Scatter plot of the relationship between the variance of byte values of page and compression ratio for different samples

图 7　不同样本的页面字节值方差与压缩率的关系散点图

然而，当样本大小缩小至 256 B 时，这种规律性显著降低。图 6(c)展示了页面前 256 B 的平均字节值与页面压缩性之间的散点图；图 7(c)展示了页面前 256 B 的字节值方差与页面压缩性之间的散点图。可以发现，页面整页平均字节值和页面前 256 B 的平均字节值与页面压缩性之间的散点图分布形态相差较大。同样，页面整页字节值方差和前 256 B 的字节值方差与页面压缩性之间的散点图分布形态相差较大，且许多数据点未能遵循整页范围内呈现的规律。

3　框架设计与实现细节

3.1　面向热点文件页的高效压缩内存管理框架

为了解决压缩内存文件页性能低效和存储系统资源利用率较低的问题，本节设计了一种面向热点文件页的高效压缩内存管理框架 EC-MemOpt(efficient compression-based memory management framework for hot file pages)。该框架旨在通过将高频访问的文件页迁移至空闲的压缩内存空间，从而提升压缩内存的空间利用率，并通过压缩预测技术降低换出文件页带来的计算开销。

EC-MemOpt 的主要框架如图 8 所示。与原始 Linux 内核相比，EC-MemOpt 主要包括 3 个关键改进：

1)压缩内存文件页管理模块。传统的压缩内存空间仅允许匿名页的换入或换出，而该模块允许文件页换入或换出压缩内存空间。

2)热点文件页识别模块。该模块的设计用于判断文件页的重用概率，能够在常数时间复杂度下帮助系统决定哪些文件页应当进行压缩存储。

3)文件页压缩性预测模块。该模块基于对文件页访问模式和迁移规律的分析，利用历史页面的压缩性，在常数时间复杂度下判断当前页面的压缩性，从而减少系统压缩页面的计算开销。

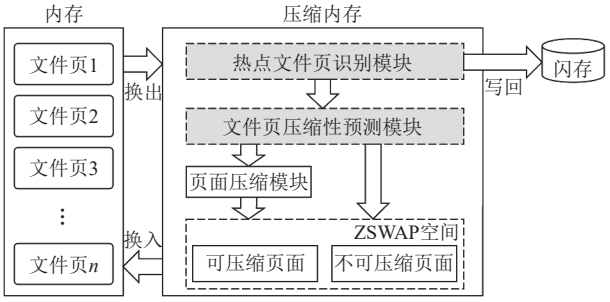


Fig. 8　EC-MemOpt framework

图 8　EC-MemOpt 框架

EC-MemOpt 的工作流程如下：

1)页面换出流程。当文件页被换出时，根据页面属性进行判断，脏文件页会被写回外存，以保证数据一致性。对于干净文件页，热点文件页识别模块会判断其重用概率。如果页面被识别为高重用概率页，则进一步通过文件页压缩性预测模块评估其压缩性，并存储在压缩内存中；如果页面重用概率较低，则直接丢弃页面。

2)页面换入流程。当缺页中断发生，需要换入文件页时，EC-MemOpt 会拦截 I/O 请求，检查所请求的文件页是否已存储在压缩交换池中，若所需的文件页已存储在压缩交换池中，块 I/O 操作将被忽略，匹配的压缩页面将被解压缩并换入内存。

3.1.1　热点文件页识别模块

在手机内存中，文件页的数量通常远超匿名页，

而当页面淘汰时，换出的文件页约为匿名页的 20 倍[16]，若不加以限制，将所有文件页都换出到压缩内存空间，会导致计算和存储资源的浪费，甚至可能引发系统崩溃。针对这一问题，本研究设计了热点文件页识别模块，旨在筛选出具有较高概率再次访问的热点文件页。准确识别此类页面，对文件页性能的提升和减少压缩资源的浪费起到重要作用。

具体来说，本研究设计的热点文件页识别模块通过使用页面结构体的 pageflag 中 3 个空闲的标记位作为“热位”、“次热位”和“首次进入位”，来标识文件页的访问热度。其中，当文件页在内存中的额外访问次数达到 2 次及以上时为“热位”；当文件页在内存中的额外访问次数达到 1 次时为“次热位”（热位、次热位统计次数时不包括页面第 1 次被换入内存时的访问）；“首次进入位”在页面第 1 次进入内存时被置位，当页面被换入压缩内存后，该标记位复位。该模块的工作流程图如图 9 所示，工作步骤如下：

步骤 1. 对即将进入压缩内存空间的干净文件页，该模块首先判断页面是否有进程映射，若无映射，则表明页面短时间内不会被进程访问，将其丢弃至外存，退出模块；若有进程映射，进入步骤 2。

步骤 2. 判断页面“热位”是否置位，若满足条件，则判定为热点页，退出模块；若不满足，则进入步骤 3。

步骤 3. 判断页面“次热位”是否置位，若满足条件，则进入步骤 4；若不满足，则丢弃页面，退出模块。

步骤 4. 判断此页面“首次进入位”是否为 1（即是否为第 1 次进入压缩内存空间），若满足条件，则判定为热点页，并修改此页面“首次进入位”为 0，退出模块；若不满足，则丢弃页面，退出模块。

3.1.2 文件页压缩性预测模块

通过将文件页迁移至压缩内存空间，可以有效减少系统的 I/O 开销，并提升文件页的性能。然而，大量文件页被换入压缩内存空间，如果不限制其计算开销，则很有可能导致系统崩溃。针对这一问题，本节结合 2.1 节文件页压缩性特征与时间局部性的关联分析，设计了基于时间局部性和滑动窗口的文件页压缩性预测模块。该模块利用历史页面的属性，在常数级时间复杂度内判断文件页的压缩性。

具体而言，当文件页被淘汰至压缩内存时，模块会进行判断：若此文件页与历史页面保持连续且历史页面均判定为不可压缩页面，则跳过该页面的压缩步骤，直接复制页面到压缩内存，并判定此页面不可压缩。若不满足上述条件，则视为无法判断页面的压缩性，仍执行页面的压缩操作。

此模块引入了滑动窗口最大尺寸（m）、存储历史页面记录的数量（n）和压缩性预测模块的页面可压缩比阈值（TH）3 个参数。图 10 展示了压缩性预测模块工作流程图，具体工作步骤如下。

步骤 1. 对于进入文件页压缩性预测模块的页面，若同时满足：1）前 n 个历史页面压缩后大小均大于设定的阈值 TH；2）此页面与存储的历史页面连续；3）当前滑动窗口大小 m_1>0，则判定页面不可压缩，m_1 值减 1，更新历史页面信息，退出模块；否则，进入步骤 2。

步骤 2. 压缩页面，若页面保持连续则重置当前滑动窗口大小 m_1 为初始值 m，并更新历史页面信息；否则，进入步骤 3。

步骤 3. 若页面不可压缩，更新历史页面信息，退出模块；否则，直接退出模块。

我们通过一个典型案例来描述文件页压缩性预测模块的工作机制。实验参数设置如下：存储历史页面记录的数量 n=2，滑动窗口最大尺寸 m=3，页面可压缩比阈值 TH=3 072，当前滑动窗口大小 m_1=2，当前压缩性预测模块已缓存的历史页面 A（*inode*=100，*offset*=100，不可压缩）和 B（*inode*=100，*offset*=101，不可压缩）。

此时，对于刚被淘汰的页面 C（*inode*=100，*offset*=102），文件页压缩性预测模块会通过已缓存的 2 个历史页面 A，B 的压缩性来判定 C 的压缩性。其空间

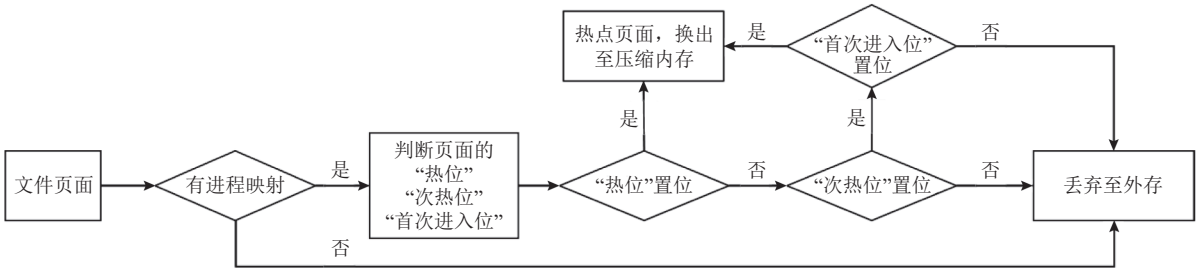


Fig. 9 Workflow of the hotspot file page recognition module

图 9 热点文件页识别模块的工作流程

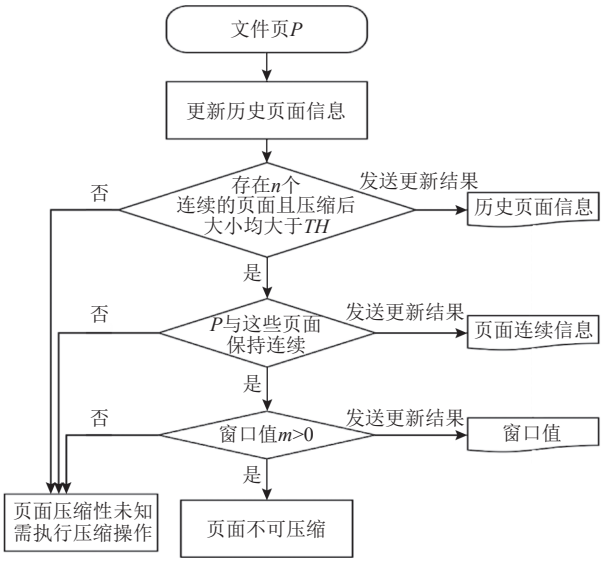


Fig. 10 Workflow diagram of file page compressibility prediction module

图 10 文件页压缩性预测模块工作流程图

连续性满足同一 *inode* 下连续偏移量访问序列且当前滑动窗口 m_1>0, 因此判定页面 *C* 压缩性为不可压缩, 同时将当前滑动窗口 m_1 减 1, 移除历史最早的页面 *A*, 新增当前页面 *C* 以维持 *n*=2 的缓存规模。该决策过程规避了冗余的压缩计算操作, 有效提升了系统效率。

在不同的应用场景中, 用户可以根据实际需求调整参数以便寻求符合场景的最佳配置。其中 *m* 值的大小决定了无需判定压缩性的连续页面数量, 该值越大, 判定为不可压缩的页面比例越高, 判定准确率越低, 减少的压缩开销越多, 但同时浪费的压缩内存空间越多; *n* 值和 *TH* 值的大小则决定了判断一个页面压缩性的严苛程度, *n* 值或 *TH* 值越大, 条件越严格, 判定准确率越高, 判定为不可压缩的页面比例越低, 减少的压缩开销越少, 但同时浪费的压缩内存空间越少。当用户应用场景更在意时间开销而忽略压缩内存的损失时, 选择较大的 *m* 值比较合适。而用户应用场景更在意压缩内存空间存储的数据密度而忽略时间开销时, 选择较大的 *n* 值或 *TH* 值比较切合。通过这种动态调整参数的机制, 模块能够根据具体的工作负载和内存管理需求, 提供更加合适的压缩预测, 从而进一步优化系统的性能表现。

3.2 基于匿名页的压缩内存交换优化方法

针对压缩开销较大的问题, 本节基于对匿名页面特性的深入分析, 设计了一种基于匿名页的压缩内存交换优化方法(anonymous page compression-based memory swapping optimization method, APC-MSO)。该方法旨在减少压缩那些不可压缩页面而带来的不必要的计算开销, 优化系统性能。

图 11 展示了 APC-MSO 的框架。与传统 Linux 内核不同的是, APC-MSO 框架引入了匿名页压缩性预测模块和匿名页压缩性标识模块。其中匿名页压缩性预测模块通过利用匿名页的特性, 在不实际压缩页面的前提下快速判断出页面的压缩性。此外, 框架的预测过程采用了循环展开技术[17], 进一步降低了压缩开销。

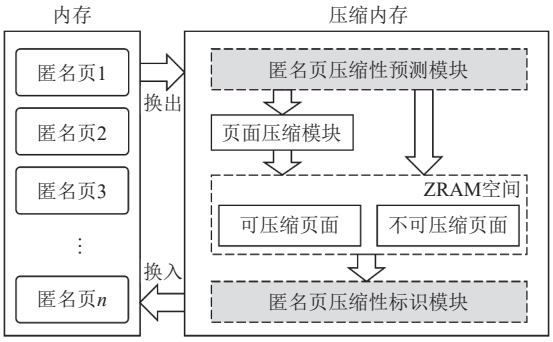


Fig. 11 Framework structure of APC-MSO

图 11 APC-MSO 的框架结构

匿名页压缩性标识模块的功能是对进入压缩内存的页面进行压缩性标识。当页面再次被淘汰到压缩内存时, 系统可以直接查阅该标记信息并判定其压缩性, 而无需重新经过压缩预测模块的计算流程, 从而有效减少不必要的计算开销, 进一步优化系统资源的利用。

3.2.1 匿名页压缩性预测模块

匿名页压缩性预测模块同时考虑到了式(3)提到的收益模型和式(4)提到的内存损失模型, 目的是最大化压缩收益并最小化压缩损失。同时, 本研究还特别注重压缩预测的精度最大化, 以确保预测结果的准确性。

$$Benefit=\sum_{i=1}^{N_{ny}+N_{nn}}Comp(P_i)-\sum_{i=1}^{N_{all}}Pred(P_i), \tag{3}$$

$$Penalty=\sum_{i=1}^{N_{ny}}(OrigSZ(P_i)-CompSZ(P_i)), \tag{4}$$

其中 *Benefit* 表示匿名页压缩性预测模块的收益, *Penalty* 表示匿名页压缩性预测模块带来的内存损失, N_{ny} 表示模块预测为不可压缩且实际为可压缩页面的数量, N_{nn} 表示预测为不可压缩且实际为不可压缩页面的数量, N_{all} 表示页面的总数量, P_i 表示页面 *i*, $Comp(P_i)$ 和 $Pred(P_i)$ 分别表示页面 *i* 实际压缩和预测的计算开销, $OrigSZ(P_i)$ 和 $CompSZ(P_i)$ 分别表示页面

i 的原始大小和压缩后的大小。

基于 2.2 节的匿名页页面特性分析，匿名页压缩性预测模块的工作流程如图 12 所示。模块输入为待测匿名页。模块的工作过程划分为通过页面平均字节值判定压缩性、通过页面字节值方差判定压缩性、通过局部采样判定压缩性 3 个工作阶段。只有在所有流程都判断页面为不可压缩时，预测器才会跳过压缩过程，判定页面为不可压缩。当任何一个阶段判断页面为可压缩时，预测将立即终止并判定页面为可压缩。同时，模块在计算过程中采用了循环展开技术，确保在减少计算开销的同时，最大限度地减少预测损失，从而提高整个系统的内存管理效率。

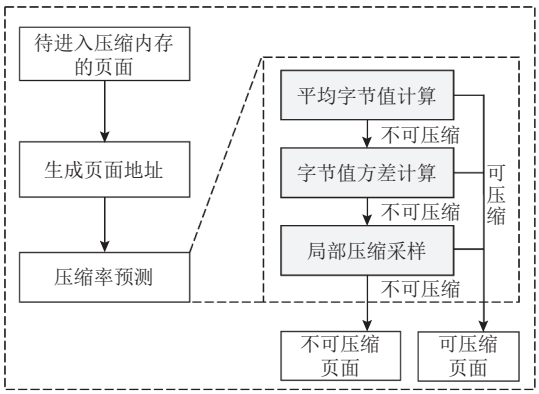


Fig. 12 Workflow diagram of the anonymous page compressibility prediction module

图 12 匿名页压缩性预测模块工作流程图

前 2 个阶段通过计算页面的平均字节值 μ 和字节值方差 σ^2 来判定页面的可压缩性。根据 2.2 节的分析，当满足 $\mu<30$，$\mu>210$，$\sigma^2<2\,000$，$\sigma^2>15\,000$ 中的任意一项时，页面较高概率可压缩，模块判定其为可压缩页，结束预测。

局部压缩采样主要用于处理前 2 个阶段仍无法精准预测压缩性的页面。例如，当页面的平均字节值处于 30~210，且字节值方差处于 2 000~15 000 时，前 2 个阶段预测方法存在盲区。针对这种情况，局部压缩采样抽取页面的部分字节进行压缩测试，并基于该样本的压缩性推测整页的压缩性。

局部压缩采样的优势在于其基于实际压缩过程进行预测，因此预测精度较高。然而，由于涉及实际压缩操作，其计算开销相较前 2 个阶段有所增加。为了平衡预测准确性与计算开销，本研究选择了页面前 512 B 作为采样样本。随着采样大小的增加，预测精度将进一步提升，但相应的计算开销也会显著增长。因此，在选择采样大小时，需确保预测的准确性，同时避免对系统计算资源造成过度负担。

3.2.2 匿名页压缩性标识模块

由于页面在系统中的流动性，换入压缩内存的页面往往在不久之后可能会再次被内存访问，当这些被换入内存的页面由于页面回收策略被再次淘汰到压缩内存时，压缩性预测模块会再次进行预测。若页面没有被改动，2 次预测的结果是完全相同的，这会造成不必要的系统资源浪费。为了解决这一问题，本研究设计了匿名页压缩性标识模块，旨在对已经完成预测的页面进行标记，使得当页面再次进入压缩内存时，能够直接利用已有的预测结果跳过重复预测，从而优化系统性能。

具体而言，当内存页面被淘汰而首次进入压缩内存时，APC-MSO 的匿名页压缩性标识模块会根据压缩性预测模块的预测结果，以标记位的形式存储在压缩内存对应的内存块中。当发生缺页中断而再次读取该页面时，会依据该标记位信息和页面在压缩内存中的大小对页面的标志位 pageflag 进行设置。经过上述操作，当该页面再次被淘汰而进入压缩内存时，系统可以直接通过已有的标志位知晓页面的压缩预测结果，从而跳过匿名页压缩性预测模块，节省计算开销。然而，如果页面重新换入内存后经历过数据修改，此时页面的压缩性可能已经发生变化。因此，当页面在内存中被修改后，需要重置标志位的压缩性标识信息。若该页面再次进入压缩内存，仍需重新进行压缩预测，以确保预测结果的准确性。通过这种机制，匿名页压缩性标识模块有效减少了重复预测计算开销，最大程度地优化了内存管理策略。

3.3 面向混合内存页的协同管理架构

通过整合 EC-MemOpt 和 APC-MSO 这 2 个内存优化组件，本研究构建了面向混合类型内存页的 EC-MemOpt+APC-MSO 协同管理架构，文件页压缩存储路径设置为 ZSWAP 分区，匿名页压缩存储路径设置为 ZRAM 分区，如图 13 所示。

该架构采用存储空间隔离策略，分别针对文件页与匿名页的内存特征实施差异化管理。其核心设计原理可分解为 2 个关键流程：

1）页面换出优化流程。当系统触发内存回收机制时，首先通过页表项中的 PG_Anon 标志位进行页面类型识别，建立路由决策机制。将文件页导入 EC-MemOpt 框架的热点文件页识别模块，该模块将访问频率超过阈值的干净文件页迁移至 ZSWAP 空间，脏页回写到持久化存储设备，而将非活跃文件页直接丢弃，以此实现 I/O 的负载分流；匿名页则通过 APC-MSO 框架的匿名页压缩性预测模块判断页面的压缩

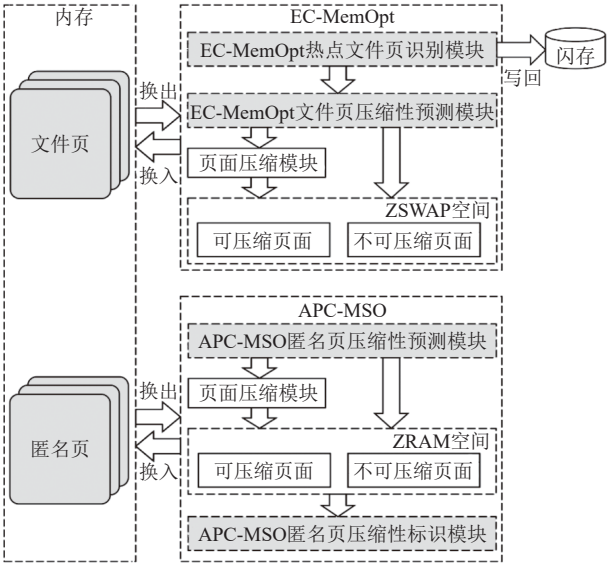


Fig. 13 EC-MemOpt+APC-MSO collaborative optimization architecture

图 13 EC-MemOpt+APC-MSO 协同优化架构

性并将其存 ZRAM 空间。

2)页面换入恢复流程。当发生缺页中断，需要从压缩内存重新换入页面时，框架对不同类型的页面实施不同的检索策略。对于文件页换入请求，框架通过文件页的索引编号 inode 和文件偏移量 offset 构建唯一的索引键值，在 ZSWAP 空间中进行匹配检索。对于匿名页换入请求，框架通过设备号“type”和逻辑偏移量“offset”构建唯一索引键值，在 ZRAM 空间执行查找操作。通过这种分区管理，可以对文件页和匿名页设置不同的可压缩比阈值，避免了跨类型页面的竞争冲突，确保文件页与匿名页的优化策略和数据结构互不干扰，更有效地利用压缩内存资源。

4 实验结果与分析

本研究所选用的测试验证平台如表 2 所示。实验数据采集方面，通过 Android 调试桥(adb)[18]将设备端捕获的页面行为追踪数据实时传输至 Windows 11 工作站，最终在 PyCharm 开发环境中完成数据可视化分析。

本研究采用 OnePlus 9 Pro 商用旗舰机作为测试与验证平台，设备配置详见表 2。该设备搭载高通骁龙 888 移动处理器(SM8350，8 核)和 8 GB 主内存，运行 Android 11R 操作系统，配备 512 GB 存储空间。实验中压缩内存容量设置为 4 GB，采用 LZ4 压缩算法实现。此内存配置的选择基于该组合在当前移动设备中的代表性，能够反映资源受限环境下的系统运行特点。此外，本研究核心聚焦于内存页面换出策略的优化，其设计并不依赖于特定的内存或压缩区容量，因此具备较好的适应性和通用性。

Table 2 Testing and Verification Platform

表 2 测试验证平台

配置及参数	详细信息
CPU	高通骁龙 888 移动处理器（SM8350），8 核
内存	8 GB
操作系统	Android 11R（内核版本 QCOM 8350R[19]）
压缩内存	4 GB（LZ4 压缩算法）
磁盘	512 GB

本研究基于 Linux 内核的 ZSWAP 与 ZRAM 子系统构建双通道压缩内存实验环境，具体配置如下：文件页压缩存储路径设置为 ZSWAP 分区，匿名页压缩存储路径设置为 ZRAM 分区，压缩算法统一采用 LZ4。为全面评估系统性能，实验设计涵盖 2 个维度的测试方法：

1)复合型用户行为模拟测试。每次实验前重启移动设备以消除系统缓存及后台进程的干扰。为构建接近真实场景的多任务环境，本研究模拟用户行为，按照表 1 中的应用列表编号顺序，使用 adb 脚本依次启动 12 个应用程序，完成首轮启动后，重复执行第 2 轮循环启动以构建后台任务负载。本研究围绕多类主流应用的展开调研，最终选择抖音作为主应用。一方面，用户在该类应用上的使用时长持续增长[20-21]；另一方面，其运行过程中匿名页频繁更新、文件页密集，视频内容通过网络持续缓冲，内存占用高[22]，能够构建出典型负载场景。相比之下，其他类型应用由于负载强度较低或页面访问模式不具有代表性，难以反映压缩机制在高压场景下的稳定性与收益，因此未作为主应用纳入测试。通过定制的 trace 工具实时采集系统或页面信息。该测试方法旨在模拟真实的用户使用场景下的系统表现。

2)应用热启动时延测试。每次实验前均重启移动设备，以清除系统缓存和后台进程干扰。随后，构建标准化测试序列，并通过 adb 自动化脚本重复执行该序列 10 次以上，记录各应用的热启动时延。本研究采用调整后的标准应用列表(移除了表 1 中编号为 2 的便签应用)，按编号依次启动 11 个应用以构成完整的测试序列。为提升测量精度，实验过程中采用以下优化措施：排除冷启动场景以确保测量对象为热启动时延；排除系统弹窗等干扰事件的影响；排除偏差过大的测试用例。该测试方法旨在测试不同

框架下的应用程序切换延迟。

4.1 EC-MemOpt 性能分析

4.1.1 热点文件页识别模块性能分析

基于 2.1 节中的结论，在当前的内存管理模式下，换入内存的绝大多数文件页不会经历迁移至活跃链表的过程，而是直接被回收，且其在内存中的驻留时间平均为 6.37 s。因此，本研究将 6.5 s 设定为文件页在内存中驻留时间阈值。若某页面的 2 次访问间隔超过 6.5 s，本研究将该页面标记为“压缩内存青睐页面”，表示该页面在被淘汰后会再次被换入，将此类文件页换出至压缩内存可以减少 I/O 开销。

在此设定的基础上，对 3.1.1 节中设计的热点文件页识别模块性能进行了测试。使用复合型用户行为模拟测试，每次统计约获取 200 万个文件页信息，重复进行了 5 次统计。在每次实验中，统一主应用为抖音，构造具代表性的复合型负载行为，并重复进行多轮数据采集与分析，测试环境与执行条件保持一致，以验证该模块在典型场景下的稳定性与准确性。在此基础上，统计了此模块筛选出的页面中压缩内存青睐页面的占比（即筛选页面精确率（*Precision*）），如式（5）所示，以及此模块筛选出的压缩内存青睐页面占全部压缩内存青睐页面的比例（即筛选页面召回率（*Recall*）），如式（6）所示。

$$Precision = \frac{N_{\text{select}} \cap N_{\text{favor}}}{N_{\text{select}}}, \tag{5}$$

$$Recall = \frac{N_{\text{select}} \cap N_{\text{favor}}}{N_{\text{favor}}}, \tag{6}$$

其中 N_{favor} 表示压缩内存青睐页面的总数，而 N_{select} 表示通过热点文件页识别模块筛选出的页面。通过对上述 200 万个文件页信息统计分析，得出了该模块筛选页面平均精确率（90.5%）和筛选页面平均召回率（57.2%）。

由于本研究提出的热点文件页识别模块采用固定逻辑规则，无需依赖可调节的模型参数或判定阈值，区别于传统的机器学习或启发式模型。因此，其精确率与召回率主要反映的是页面访问行为本身的特征，而非算法参数调整的结果。实验结果显示，该模块在多轮测试中取得了较高的精确率（平均 90.5%），表明其筛选出的页面大多具有较强的再访问可能性，具备明显的压缩价值。

然而，召回率相对较低（平均 57.2%），这主要源于模块在设计上优先强调识别结果的准确性，避免将非热点页面误标为热点页面。具体而言，为保证较高的精确率，模块的识别标准被设置得较为严格，相较于压缩内存青睐页面的实际范围偏窄，导致部分真实的热点页面未被及时识别。另一方面，部分最终表现为“再次被访问”的页面，在其首次进入内存时的访问频次较低，缺乏明显的访问特征，因而未能通过模块的判定逻辑而被排除。

随后，本研究统计了热点文件页识别模块筛选出的文件页比例随时间的变化趋势。使用复合型用户行为模拟测试，共统计了约 415 万个文件页流动信息。图 14 展示了进入压缩内存的干净文件页比例时序变化，在系统运行超过 400 s，保持稳定的情况下，约有 5.8% 的文件页能够通过模块筛选进入压缩内存，避免了对大量非热点文件页的无效换出。

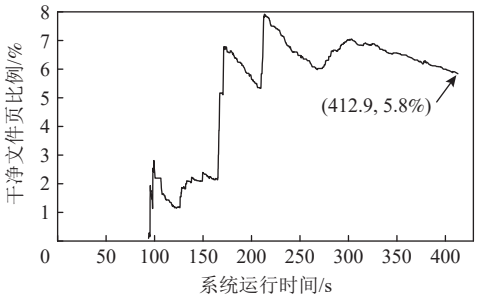


Fig. 14 Changes of the proportion of clean file pages entering compressed memory varies with time series

图 14 进入压缩内存的干净文件页比例时序变化

综合而言，该模块在精确率与召回率之间采取了偏保守的策略取舍，以降低误判所带来的资源浪费。尽管当前召回率尚有提升空间，但整体识别策略在系统稳定性与压缩资源利用效率之间实现了较为合理的平衡。

4.1.2 文件页压缩性预测模块性能分析

本次实验固定 n=2（只保留 2 个历史页面的信息），并调整不同的 m 值和 TH 值，以观察接口的准确率（*EC_Accuracy*）（如式（7））和减少无效压缩的比例（*Reduce_Ratio*）（如式（8））。

$$EC_Accuracy = \frac{N_{\text{nn}}}{N_{\text{nn}} + N_{\text{ny}}}, \tag{7}$$

$$Reduce_Ratio = \frac{N_{\text{nn}}}{N_{\text{noncomp}}}, \tag{8}$$

其中 N_{noncomp} 表示换出页面中实际为不可压缩页面的数量，N_{nn} 表示预测为不可压缩且实际为不可压缩页面的数量，N_{ny} 表示模块预测为不可压缩且实际为可压缩页面的数量。需要注意的是，本研究设计的文件页压缩预测接口仅能预测出页面状态为不可压缩或不确定其压缩性，而 3.2.1 节提到的多级页面压缩性预测接口能预测出页面状态为不可压缩或可压缩，

因此二者的准确率计算公式不同。

为了评估压缩预测接口的性能, 本研究采用复合型用户行为模拟测试, 获取到 94 万个文件页作为测试数据, 其中包括 371 393 个不可压缩页面。在实验中, 为了避免过多的存储开销, 将缓存页面数量设置为 2, 围绕 m 和 TH 开展了 3 组 2 048 个不同参数组合的实验, 并绘制了压缩预测接口的准确率, 如图 15 所示。通过压缩预测接口减少的无效压缩页面的比例, 完整趋势曲线如图 16 所示。

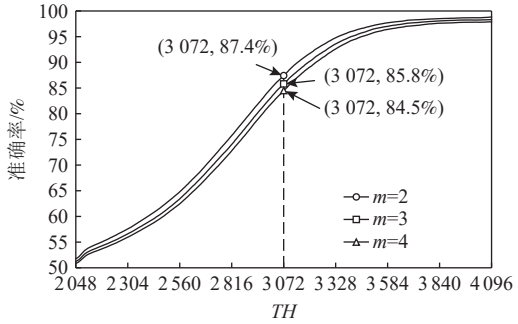


Fig. 15 Prediction accuracy of the compression prediction interface of file page

图 15 文件页压缩性预测接口的预测准确率

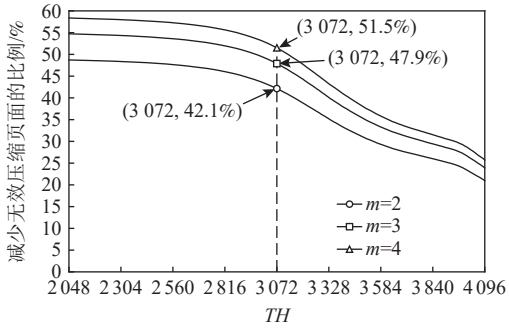


Fig. 16 Proportion of invalid compressed pages reduced by the compression prediction interface of file pages

图 16 文件页压缩性预测接口减少无效压缩页面的比例

结果显示, m 值越大, 条件越宽松, 减少无效压缩页面的比例越高, 但判定准确率越低; TH 值越大且条件越严格, 判定准确率越高, 减少无效压缩页面的比例越低。在不同的场景中, 可以根据需求调整这些参数。本研究在权衡了准确率和计算开销后, 在后续实验中设置了 TH=3 072, m=3 作为参数配置。在该配置下, 压缩预测接口的准确率达到了 85.8%, 减少无效压缩页面的比例达到了 47.9%

4.1.3 EC-MemOpt 框架整体性能分析

本研究同时启用了热点文件页识别模块和文件页压缩性预测模块进行 EC-MemOpt 框架整体效果测试, 使用复合型用户行为模拟测试进行 2 组实验, 2 组实验仅在使用主进程的使用时间上有所差别。由于热点文件页识别模块能够过滤掉 94% 的文件页, 因此在启用热点文件页识别模块的情况下, 实验中的数据量相较于未启用模块时减少了 1 个数量级。

第 1 组实验包含 67 675 条测试数据, 其中 36 941 个不可压缩页面(约占 55%)、22 353 个不可压缩且预测正确的页面, 减少无效压缩页面的比例高达 60.5%, 预测准确率为 96.5%。

第 2 组实验包含 100 623 条测试数据, 其中 67 493 个不可压缩页面(约占 67%)、43 135 个不可压缩且预测正确的页面, 减少无效压缩页面的比例高达 63.9%, 不可压缩页面预测准确率为 97.9%。

同时, 本研究模拟实现了 PGPI[15] 与 EZSWAP[16] 这 2 种代表性方法的核心预测策略。其中, EZSWAP 采用基于页面信息熵的压缩性估计方法, PGPI 则结合页面平均字节值与采样压缩率对页面压缩性进行预测。本研究在相同数据集上对这 2 种策略进行了模拟评估, 覆盖 20 万个文件页, 并以“不可压缩页面预测准确率”与“无效压缩页面减少比例”作为关键评估指标。

结果如表 3 所示, EC-MemOpt 的不可压缩页预测准确率为 97.9%, 无效压缩减少比例为 63.9%, 相比之下, PGPI 分别为 51.79% 和 95.45%, EZSWAP 为 49.31% 和 86.76%。尽管 EZSWAP 与 PGPI 在减少无效压缩方面表现突出, 但其准确率偏低, 说明其预测倾向于保守跳过页面压缩。相比之下, EC-MemOpt 在保持高准确率的同时, 也有效减少了冗余压缩操作, 在预测准确性与开销控制之间实现平衡。

Table 3 Compression Prediction Effectiveness Between EC-MemOpt and Other Methods

表 3 EC-MemOpt 与其他方法的压缩预测效果 %

方法	准确率	减少无效压缩页面的比例
EC-MemOpt(本文)	97.9	63.9
PGPI	51.79	95.45
EZSWAP	49.31	86.76

为了验证 EC-MemOpt 压缩内存空间利用率提升效果, 本研究采用复合型用户行为模拟测试进行实验, 压缩内存空间占用量时序变化结果如图 17 所示, 横坐标采用复合编码标注, 如 A2 表示第 1 次启动编号为 2 的应用程序, B4 表示第 2 次启动编号为 4 的应用程序(应用编号参见表 1)。实验数据显示, 在启用 EC-MemOpt 机制后, 当系统进入稳态运行阶段时, 压缩内存中文件页存储呈现显著稳定性特征, 对应

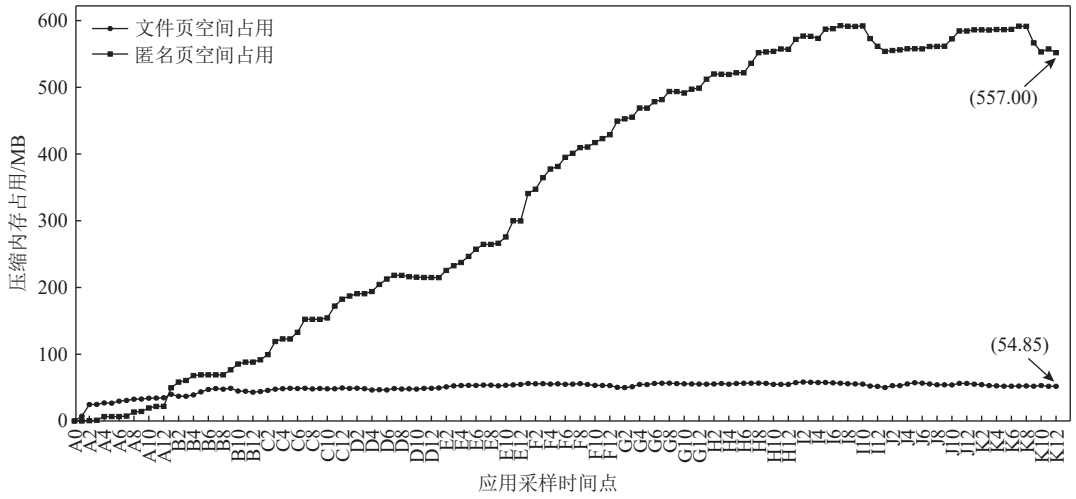


Fig. 17 Time series variation of compressed memory space usage

图 17 压缩内存空间占用量时序变化

压缩内存空间占用稳定在 55±5 MB 范围。该现象表明系统已达成动态平衡状态，即单位时间内存入压缩内存的文件页数量与置换出的文件页数量比值趋近于 1。此平衡态的建立证明了 EC-MemOpt 页面置换策略在用户行为模拟测试的长期有效性，被存入的压缩文件页具有高重用概率特征。与原始内核相比，压缩内存空间利用提升 9.8%。

4.2 APC-MSO 性能分析

4.2.1 匿名页压缩性预测模块性能分析

本研究使用复合型用户行为模拟测试，获取了连续的 163 864 个被内存淘汰的匿名页，并对匿名页压缩性预测接口进行分析。在匿名页被淘汰并转入压缩内存的过程中，匿名页压缩预测模块接口会被调用。为了评估 APC-MSO 的性能，本研究使用式(9)计算 APC-MSO 压缩预测接口的准确率：

$$APC_Accuracy = \frac{N_{yy} + N_{nn}}{N_{yy} + N_{nn} + N_{yn} + N_{ny}}, \quad (9)$$

其中 N_{yy} 表示模块预测为可压缩且实际为可压缩页面的数量，N_{nn} 表示预测为不可压缩且实际为不可压缩页面的数量，N_{yn} 表示预测为可压缩且实际为不可压缩页面的数量，N_{ny} 表示模块预测为不可压缩且实际为可压缩页面的数量。

图 18 显示了在不同配置下匿名页压缩预测接口调用时间和准确率，其中 PGPI 代表论文中提到的多级预测模块[15]，APC-MSO 的数字表示 3.2.1 节中匿名页多级页面压缩性预测模块前 2 个阶段的页面采样大小(该值仅代表页面平均字节值和页面字节值方差计算采样，而第 3 阶段的局部压缩采样大小固定为 512 B)。2.2 节提到，APC-MSO-256 的规律性较差，因此本研究使用 APC-MSO-512 作为临界测试值。

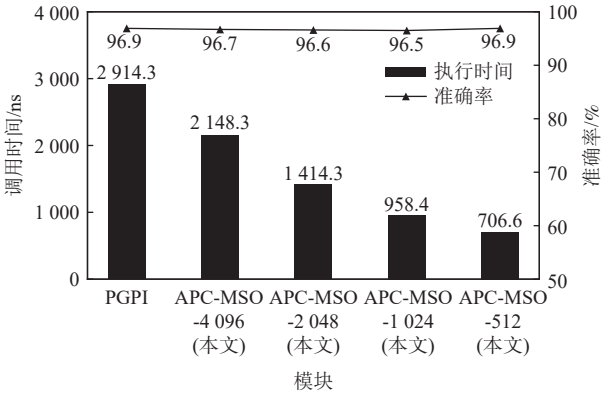


Fig. 18 Call time of compression prediction interface and accuracy under different configurations

图 18 不同配置下压缩预测接口调用时间和准确率

可以观察到，在几乎不降低预测准确率的前提下，APC-MSO 显著减少了预测开销。APC-MSO-512 接口调用时长仅为 706.6 ns，约占实际压缩 1 个页面开销的 11%，相比于 PGPI 减少了 75.8%。后续实验中本研究统一使用 APC-MSO-512 配置。

4.2.2 匿名页压缩性标识模块效果分析

本节对匿名页压缩性标识模块效果进行了测试，选择复合型用户行为模拟测试，通过模拟用户行为，分析了匿名页在内存与压缩内存之间交换的过程中，启用匿名页压缩性标识模块与未启用该模块时，匿名页压缩性预测模块接口的调用次数对比。

图 19 展示了启用和未启用模块时的调用页面压缩性预测接口次数归一化对比，结果表明，启用匿名页压缩性标识模块能够减少 13.6% 的匿名页压缩性预测接口调用。

4.3 混合内存优化架构性能分析

为验证混合内存优化架构对系统的优化效果，

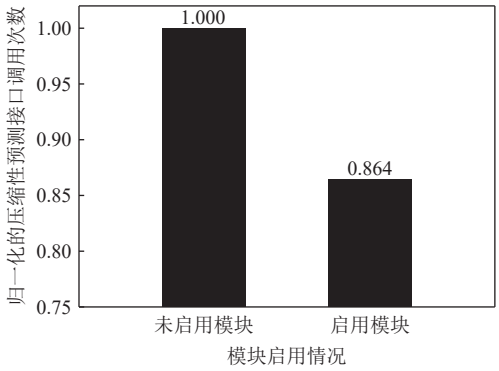


Fig. 19 Normalized comparison for invocation count of compression prediction interface under different configurations

图 19 不同配置下压缩预测接口调用次数归一化对比

本研究对 4 种系统配置进行应用热启动延迟测试对比分析: OnePlus QCOM 8350R 开源内核(Baseline)、仅 APC-MSO 优化方法、仅 EC-MemOpt 优化方法以及 EC-MemOpt+APC-MSO 协同优化方法。各实验组参数设置遵循以下原则: Baseline 实验组仅启用 4 GB ZRAM 作为压缩存储(即保持原始内核参数); APC-MSO 实验组仅启用 4 GB ZRAM 配置; EC-MemOpt 实验组仅启用 4 GB ZSWAP; EC-MemOpt + APC-MSO 协同优化组配置 2 GB ZRAM(APC-MSO 专用)与 2 GB ZSWAP(EC-MemOpt 专用), 确保压缩存储总容量与对照组一致。

图 20 通过归一化方式呈现各方法的平均切换速率。实验数据显示: 与 Baseline 相比, 仅 APC-MSO 方法使应用平均切换速率提升 3.9 个百分点; 仅 EC-MemOpt 方法获得 4.4 个百分点的平均切换速率提升; EC-MemOpt+APC-MSO 协同方法展现出显著的性能叠加效应, 实现 10.5 个百分点的平均切换速率提升。

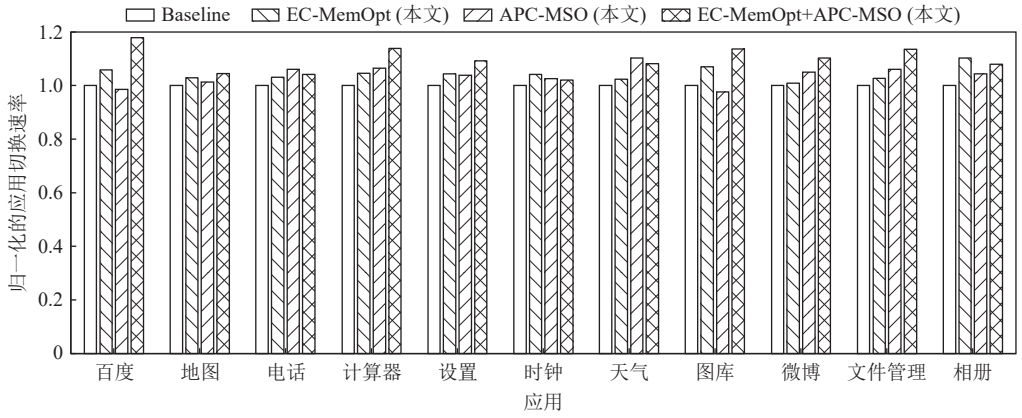


Fig. 20 Normalized application switch rate comparison under different configurations

图 20 不同配置下应用归一化切换速率对比

表 4 显示了 EC-MemOpt+APC-MSO 与目前代表性系统 EZSWAP 和 PGPI 的对比。在文件页面支持方面, EC-MemOpt+APC-MSO 和 EZSWAP 均支持文件页面压缩, 而 PGPI 则不支持文件页压缩。在空间利用率上, EC-MemOpt+APC-MSO 与 EZSWAP 相比, 具有中等水平的空间利用率, 而 EZSWAP 由于其较为积极的压缩策略在空间利用率上表现较高。PGPI 则不支持文件页换出, 导致空间利用率较低。关于开销, EC-MemOpt+APC-MSO 的单个页面压缩开销为约 700 ns(文件页约 10 ns), 而 EZSWAP 与 PGPI 的开销都高达 3 μs, 这表明 EC-MemOpt+APC-MSO 的混合内存优化架构在处理页面时计算开销更低, 能够在资源受限的环境下更高效地进行页面压缩。因此, EC-MemOpt+APC-MSO 能够更好地平衡准确性与性能开销, 适用于移动设备等资源受限的场景。

Table 4 Comparative Analysis of EC-MemOpt+APC-MSO and Other Methods

表 4 EC-MemOpt+APC-MSO 与其他方法的对比分析

对比项	EC-MemOpt +APC-MSO (本文)	EZSWAP	PGPI
文件页压缩支持	支持	支持	不支持
压缩性预测	支持	支持	支持
空间利用率	中	高	低
CPU 开销/ns	约 700 (文件页 10)	约 3 000	约 3 000

5 总结与展望

本研究针对当前压缩内存技术存在的文件页性能低下、压缩开销较高及存储资源利用率方面不足的问题, 设计了面向混合类型内存页的协同管理架构 EC-MemOpt+APC-MSO, 并进行了实验验证。结果表明, 相较于 OnePlus 维护的 QCOM 8350R 内核, 该架构在内存压力下, 应用切换速率和空间利用率方

面分别提升 10.5 个百分点和 9.8 个百分点。然而，本研究仍然存在一定局限性。首先，热点文件页识别模块主要依赖页面访问频率来判断页面热度。尽管这一方法在识别热点页面时具备较高的准确率，但在召回率方面仍然存在不足，导致部分热点页面未能被有效识别，从而影响系统的整体性能。此外，尽管本文设计的架构使资源利用率得到了提升，但仍有大量压缩内存资源未能得到充分利用。未来工作中，可以进一步优化热点文件页识别模块，以提高召回率，并探索更加高效的资源管理策略，以提升存储资源的利用效率。

作者贡献声明：龙家璇、孙韬宁提出优化方法并撰写论文；肖春华对论文提出指导意见；熊浩宇提供优化思路；左笑寒、吴林、陈延科对论文的撰写和修改提供帮助。

参 考 文 献

[1] Zhao Ziming, Liu Fang, Cai Zhiping, et al. Edge computing: Platforms, applications and challenges[J]. Journal of Computer Research and Development, 2018, 55(2): 327−337 (in Chinese)
(赵梓铭, 刘芳, 蔡志平, 等. 边缘计算: 平台、应用与挑战[J]. 计算机研究与发展, 2018, 55(2): 327−337)

[2] Li Tong, Xia Tong, Wang Huandong, et al. Smartphone app usage analysis: Datasets, methods, and applications[J]. IEEE Communications Surveys & Tutorials, 2022, 24(2): 937−966

[3] Falaki H, Mahajan R, Kandula S, et al. Diversity in smartphone usage[C]//Proc of the 8th Int Conf on Mobile Systems, Applications, and Services. New York: ACM, 2010: 179−194

[4] Shepard C, Rahmati A, Tossell C, et al. LiveLab: Measuring wireless networks and smartphone users in the field[J]. ACM SIGMETRICS Performance Evaluation Review, 2011, 38(3): 15−20

[5] Tremaine R B, Franaszek P A, Robinson J T, et al. IBM memory expansion technology (MXT)[J]. IBM Journal of Research and Development, 2001, 45(2): 271−285

[6] Li Changlong, Shi Liang, Liang Yu, et al. SEAL: User experience-aware two-level swap for mobile devices[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 4102−4114

[7] Zhou Xinming. Research of memory compression based on mobile system[D]. Shanghai: East China Normal University, 2022 (in Chinese)
(周新明. 基于移动操作系统的内存压缩研究[D]. 上海: 华东师范大学, 2022)

[8] Kim S H, Jeong J, Kim J S. Application-aware swapping for mobile systems[J]. ACM Transactions on Embedded Computing Systems, 2017, 16(5s): 182

[9] Gao Xiang, Dong Mingkai, Xie Miao, et al. EROFS: A compression-friendly readonly file system for resource-scarce devices[C]//Proc of the 2019 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2019: 149−162

[10] Qiu Bin, Qiu Shuwei. Research on real time migration mechanism of virtual machine based on memory compression technology[J]. China CIO News, 2023(12): 44−47 (in Chinese)
(邱彬, 邱树伟. 基于内存压缩技术的虚拟机实时迁移机制研究[J]. 信息系统工程, 2023(12): 44−47)

[11] Harnik D, Kat R, Sotnikov D, et al. To zip or not to zip: Effective resource usage for real-time compression[C]//Proc of the 11th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2013: 229−241

[12] Son S, Lee S Y, Jin Y, et al. ASAP: Fast mobile application switch via adaptive prepaging[C]//Proc of the 2021 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2021: 365−380

[13] Meguerdichian S, Noshadi H, Dabiri F, et al. Semantic multimodal compression for wearable sensing systems[C]//Proc of the 2010 IEEE Sensors. Piscataway, NJ: IEEE, 2010: 1449−1453

[14] Ji Cheng, Chang Lipin, Shi Liang, et al. Lightweight data compression for mobile flash storage[J]. ACM Transactions on Embedded Computing Systems, 2017, 16(5s): 183

[15] Song T, Kim M, Lee G, et al. Prediction-guided performance improvement on compressed memory swap[C]//Proc of the 40th IEEE Int Conf on Consumer Electronics. Piscataway, NJ: IEEE, 2022: 1−6

[16] Kim J, Kim C, Seo E. EZSWAP: Enhanced compressed swap scheme for mobile devices[J]. IEEE Access, 2019, 7: 139678−139691

[17] Huang J C, Leng T. Generalized loop-unrolling: A method for program speedup[C]//Proc of the 2nd IEEE Symp on Application-Specific Systems and Software Engineering and Technology. Piscataway, NJ: IEEE, 1999: 244−248

[18] Regupathy R. Unboxing Android USB: A Hands-on Approach with Real World Examples[M]. Berkeley, CA: Apress, 2014: 125−138

[19] OnePlusOSS. android_kernel_oneplus_sm8350[EB/OL]. (2021-03-24)[2025-08-25]. https://github.com/OnePlusOSS/android_kernel_oneplus_sm8350

[20] Chen Qiang, Li Changlong. Argus: Real-time HQ video decoding with CPU coordinating on consumer devices[C]//Proc of the 45th IEEE Real-Time Systems Symp. Piscataway, NJ: IEEE, 2024: 43−56

[21] Davis R. Short-form video market soars: People are spending more and more time on short-form video[EB/OL]. (2021-06-21)[2025-08-25]. https://variety.com/2021/streaming/news/china-short-video-market-study-1235001776/

[22] Li Zhuqi, Xie Yaxiong, Netravali R, et al. Dashlet: Taming swipe uncertainty for robust short video streaming[C]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 1583−1599



Long Jiaxuan, born in 1999. PhD candidate. His main research interest includes hardware and software co-design.

龙家璇, 1999 年生。博士研究生。主要研究方向为软硬件协同设计。



Xiao Chunhua, born in 1986. Professor. Her main research interests include MPSoCs, hardware and software co-design, and embedded systems.

肖春华, 1986 年生。教授。主要研究方向为多处理器系统芯片、软硬件协同设计、嵌入式系统。



Sun Taoning, born in 2000. Master candidate. His main research interest includes computer architecture.

孙韬宁, 2000 年生。硕士研究生。主要研究方向为计算机体系结构。



Xiong Haoyu, born in 2000. Master candidate. His main research interest includes computer architecture.

熊浩宇, 2000 年生。硕士研究生。主要研究方向为计算机体系结构。



Zuo Xiaohan, born in 2000. Master candidate. Her main research interest includes neural network hardware acceleration and parallel computing.

左笑寒, 2000 年生。硕士研究生。主要研究方向为神经网络硬件加速与并行计算。



Wu Lin, PhD. His main research interests include hardware/software codesign, file systems, and distributed systems.

吴　林, 博士。主要研究方向为软硬件协同设计、文件系统、分布式系统。



Chen Yanke, born in 1998. Master candidate. His main research interest includes recommender systems.

陈延科, 1998 年生。硕士研究生。主要研究方向为推荐系统。