CHINA LINUX KERNEL
中国Linux内核开发者大会

# PMR:并行内存回收的设计与实践

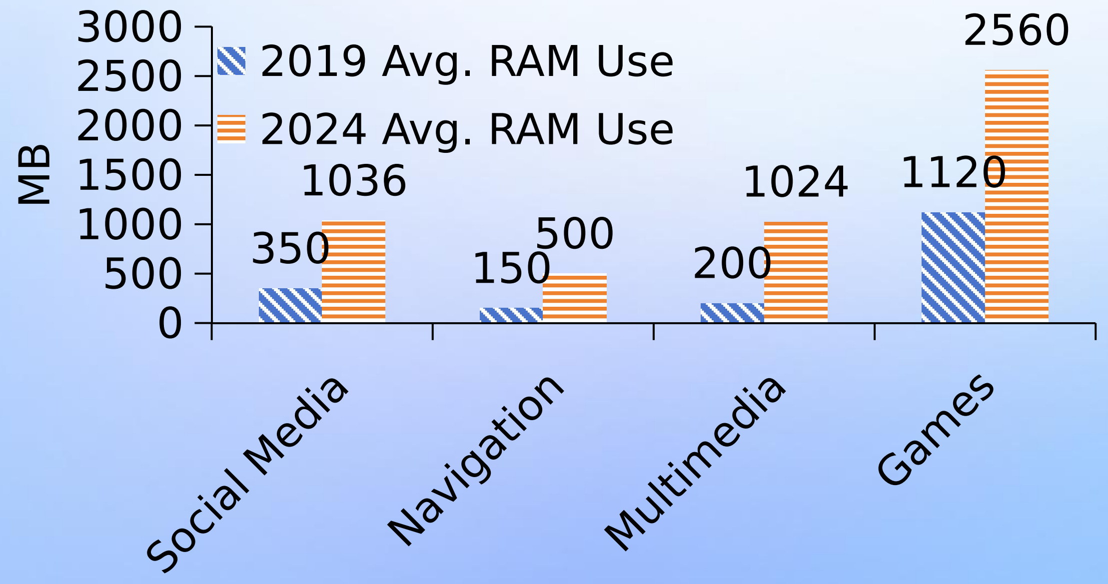
李文通
OPPO高级底层软件工程师
2025年11月1日

# 目录 CONTENTS

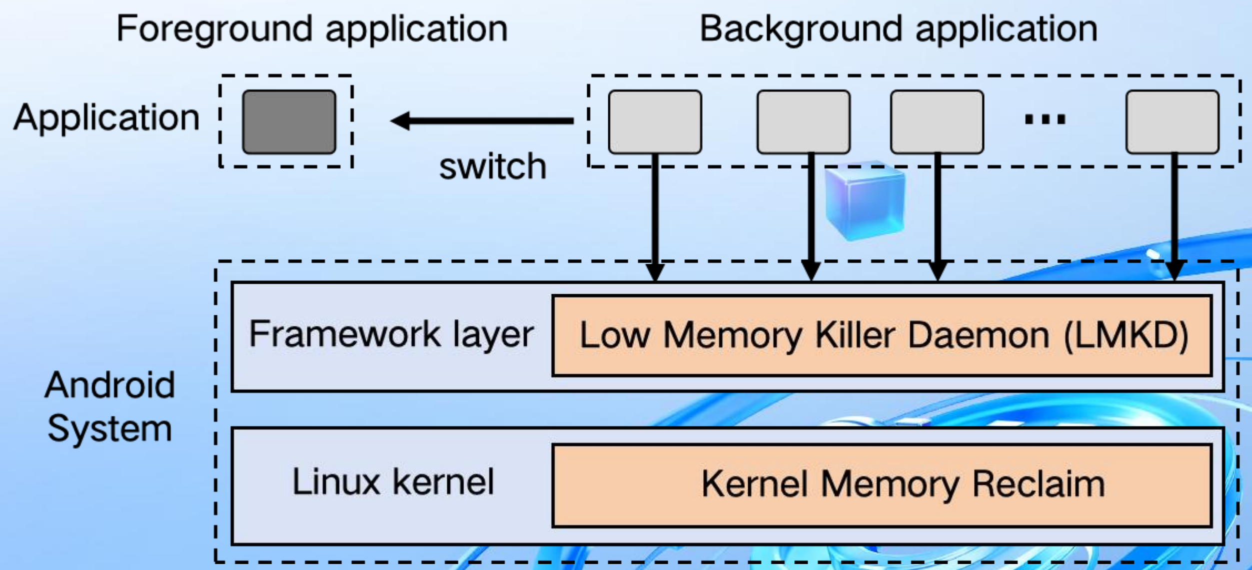


# Today’ s Mobile Devices are facing Memory Pressure

- Memory capacity is becoming a scarce resource on mobile devices
  - The application memory footprint have been growing.[1]
  - The physical memory capacity of mobile devices has seen only modest increase.
- Mobile systems need efficient memory reclaim to relieve memory pressure




Application memory footprint grows rapidly



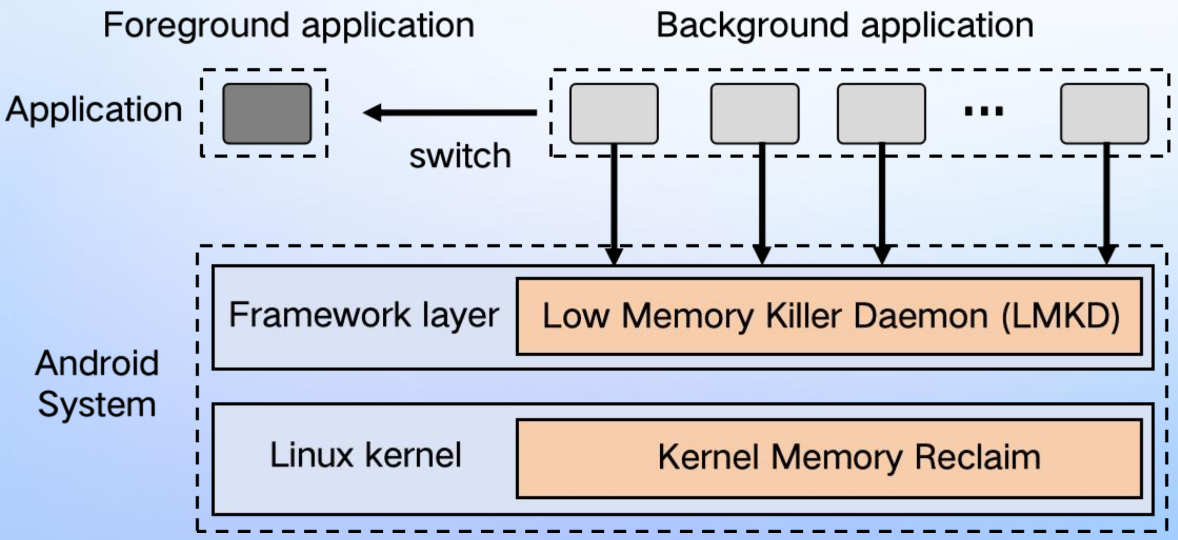

Android memory reclaim system

[1] data from DeepSeek

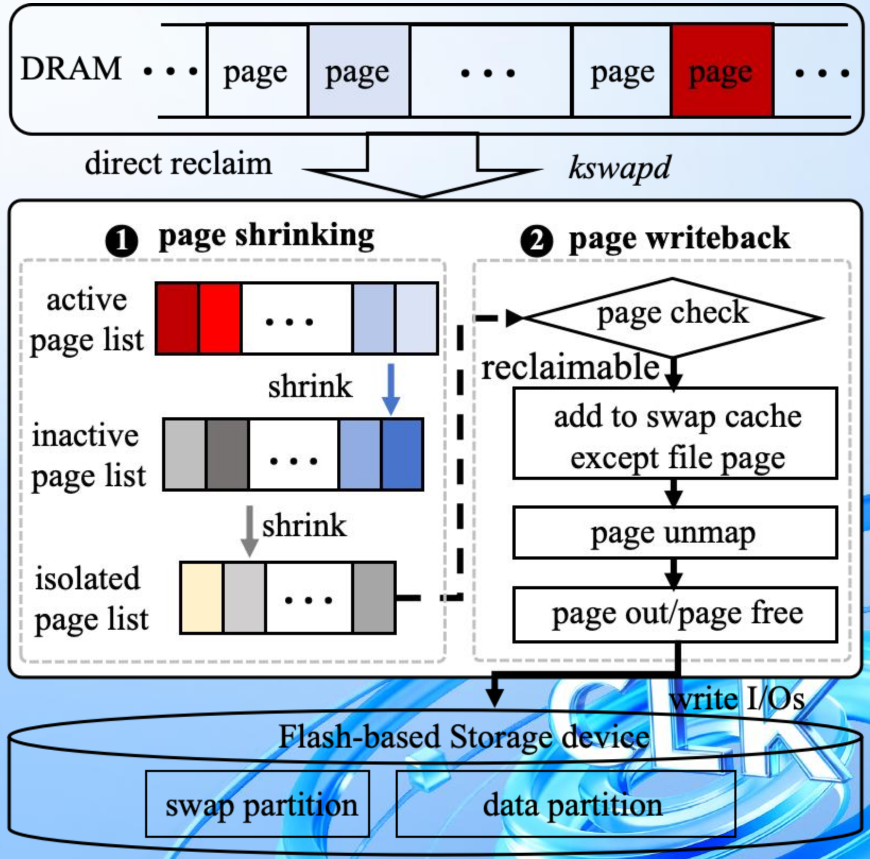
# Memory Reclaim on Mobile device

Kernel memory reclamation is more cost-efficient than LMKD

It inculdes memory swapping and direct reclaim, and both of them **follow the same reclaim path: page shrinking and page writeback**




Android memory reclaim system

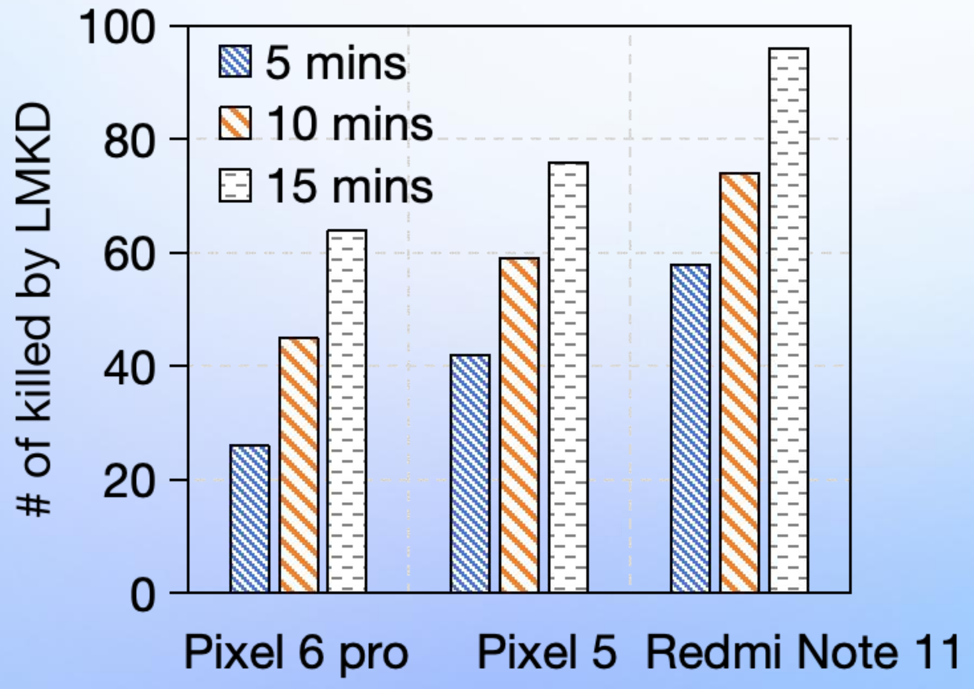



Kernel memory reclaim path

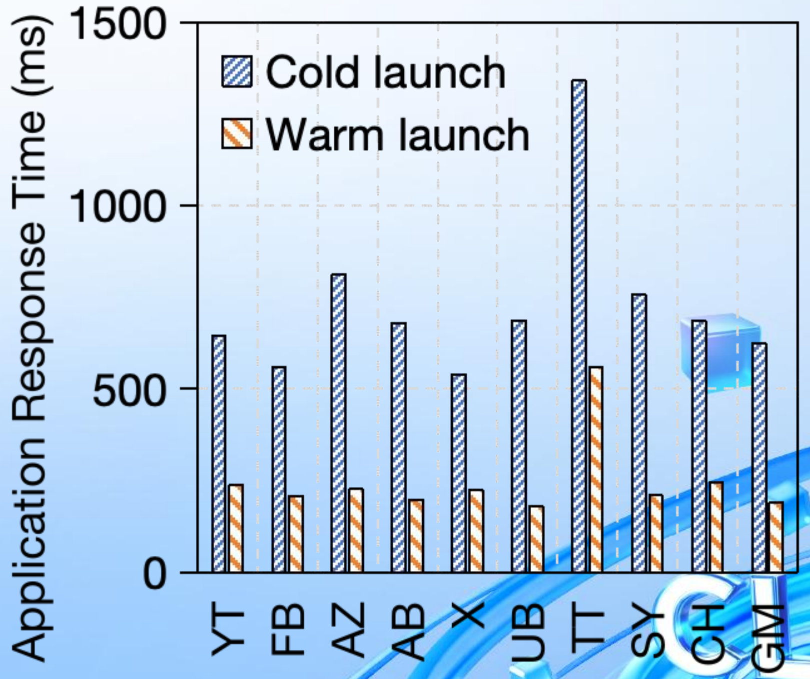
# Preliminary Study of System Memory Reclaim

## Frequent, Expensive Process Killing

LMKD is frequently triggered, causing the application to cold launch, accompanied by a longer response time.




Many processes are killed among different mobile devices



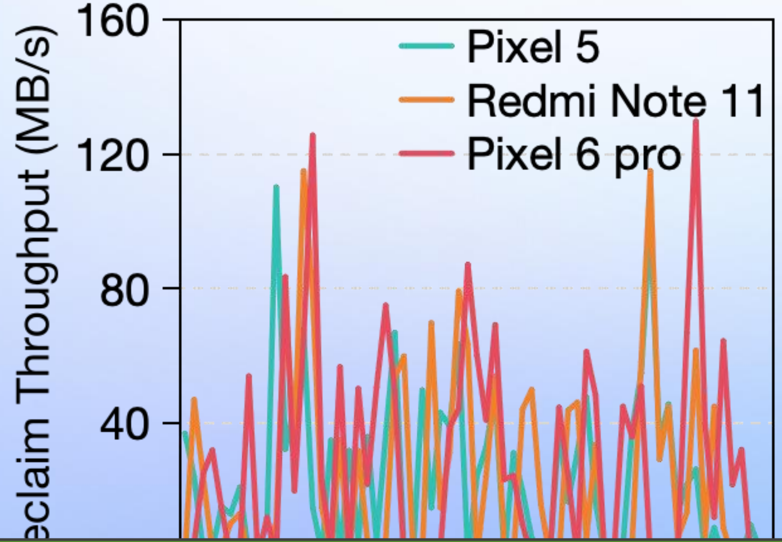

YT: YouTube, FB: Facebook, AZ: Amazon, AB: Angry Birds, X: Twitter, UB: Uber, TT: TikTok, SY: Spotify, CH: Chrome, GM: Gmap

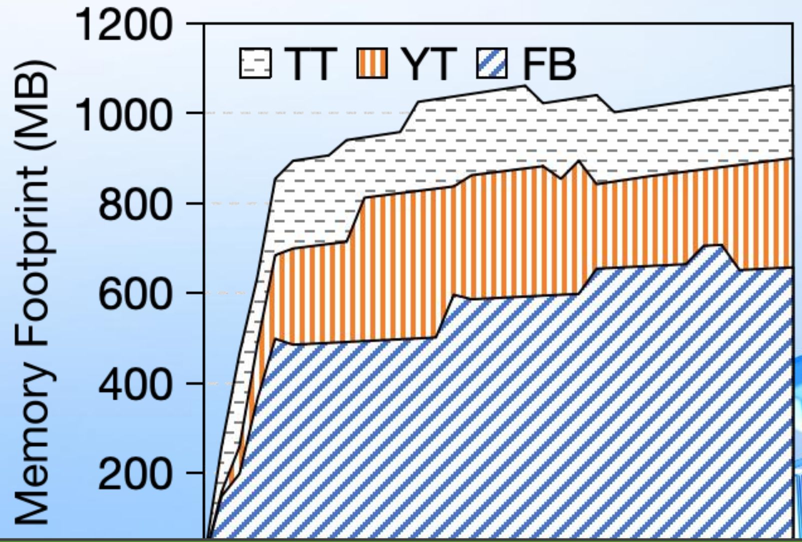
# Preliminary Study of System Memory Reclaim

## Sluggish Kernel Memory Reclaim

With the upgrade of mobile devices, the memory reclaim throughput has grown slowly.

The maximum throughput is only over 100 MB, which is far lower than the memory requirements of the popular applications.




While app memory footprint is increasingly high, the hardware advance does not benefit the efficiency of kernel memory reclaim.

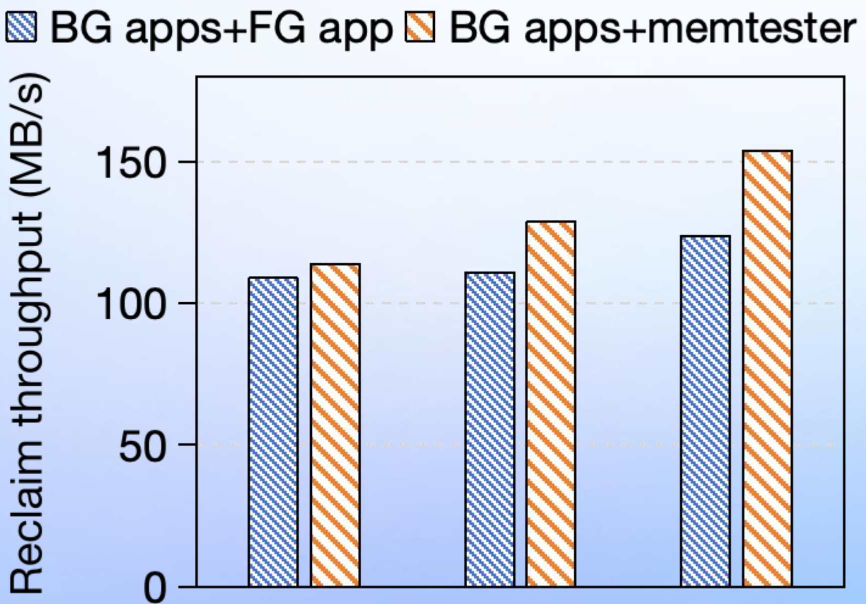
# Performance Bottleneck Analysis

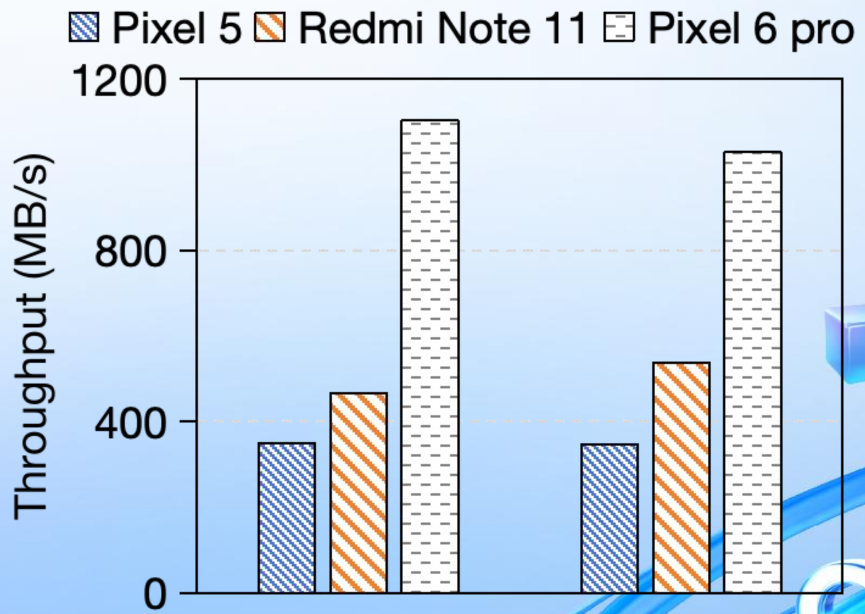
CHINA LINUX KERNEL
中国Linux内核开发者大会

## I/O Utilization

### Low Impact of I/O Conflicts

### Excessive Storage Bandwidth



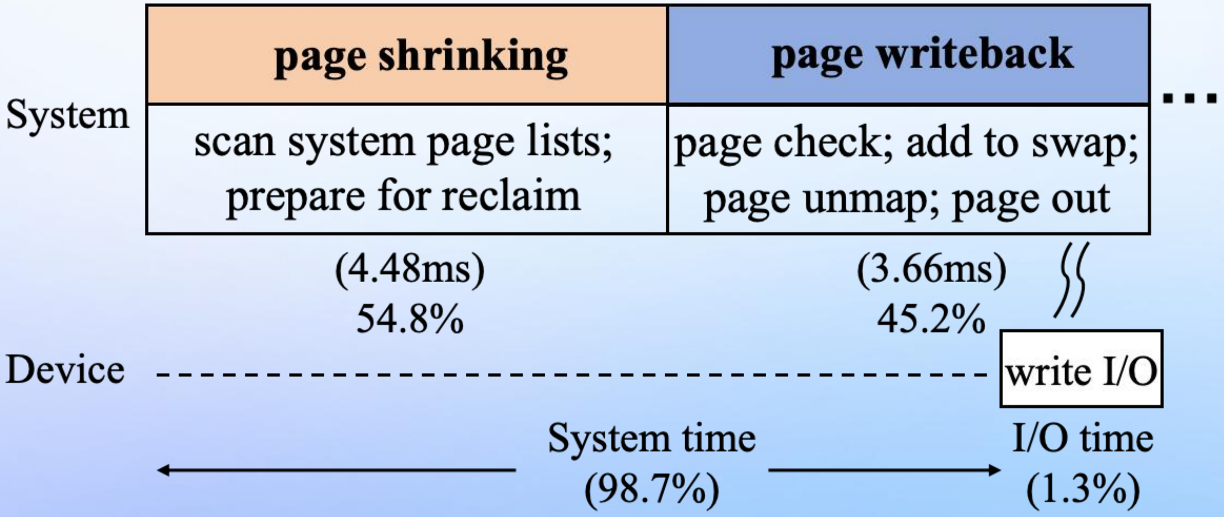




I/O Performance is not the bottleneck of kernel memory reclaim.

# Performance Bottleneck Analysis

## Sub-optimal Memory Reclaim Path

**Sequential Execution of Key Steps:** page writeback waits on the results of page shrinking, introducing unnecessary delays.




Average time breakdown of memory reclaim path

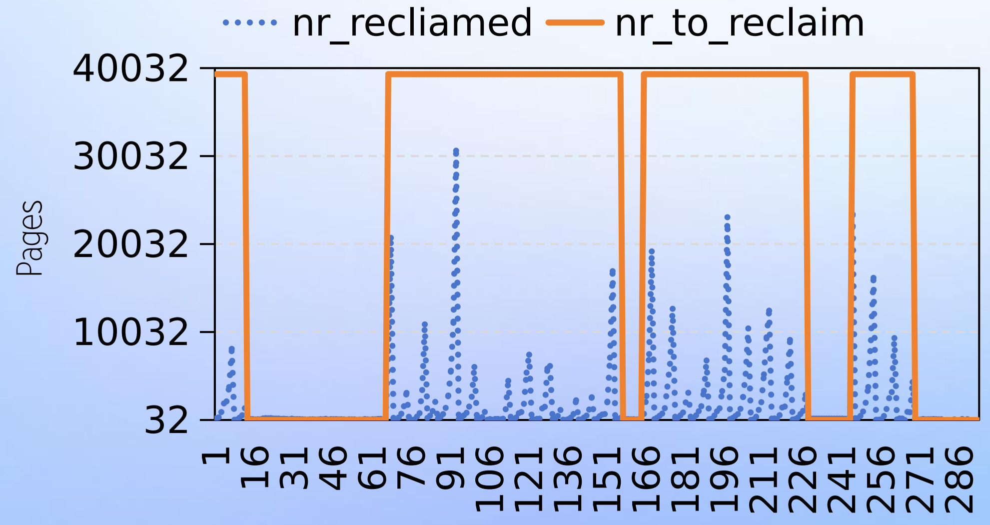
# Performance Bottleneck Analysis

CHINA LINUX KERNEL
中国Linux内核开发者大会

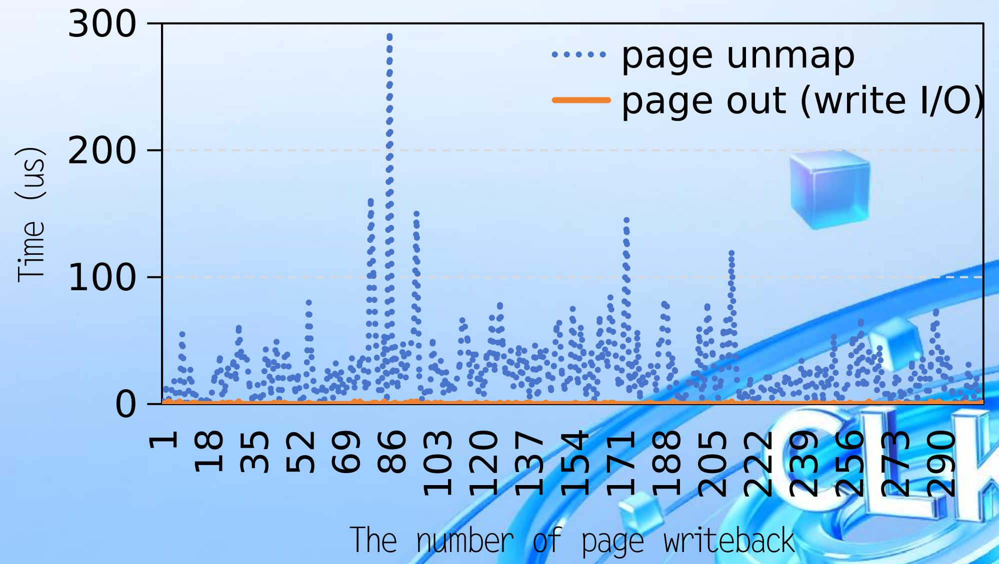
## Sub-optimal Memory Reclaim Path

**Inefficient Page Shrinking:** page shrinking suffers from inefficient reclaim and repeated invocation.

**Internally Blocked Page Writeback:** page unmap latency is highly unstable and produces small writes that cannot unleash the potential of flash storage.




nr_to_reclaim vs nr_reclaimed when system performs memory reclaim

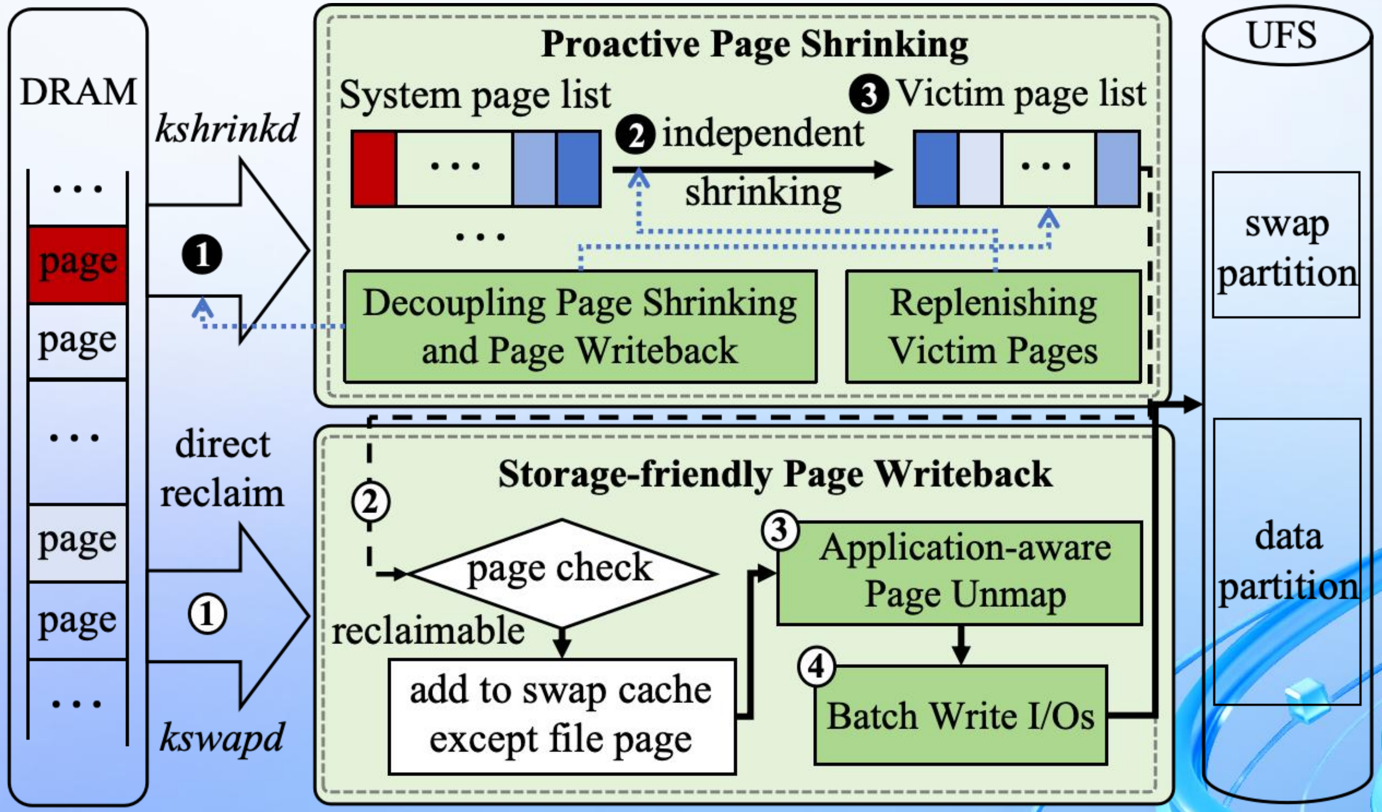



Time breakdown of page writeback (the time of page out is not zero but fluctuates under a few microseconds)

# PMR’s Design

PMR parallelizes key steps of memory reclaim to fulfill application memory demand

- Proactive Page Shrinking(PPS)
- Storage-friendly Page Writeback(SPW)




Overview of PMR

# PMR’s Design

## Proactive Page Shrinking(PPS)

**Decoupling Page Shrinking and Page Writeback:**

(1) Enable an independent *kshrinkd* Thread.

(2) Maintain a victim page list.

**Replenishing Victim Pages:**

(1)Design page shrinking watermark: The parameter *nr_ideal_victim_page* (i.e., δ in Algorithm 1) specifies the target number of victim pages that should be maintained in the new victim page list.

(2)Preform timely page shrinking.

---

**Algorithm 1:** Always Ready Page Shrinking

**Input:** *nr_victim_page*: the actual number of pages in the victim page list (The initial value is 0);
*nr_ideal_victim_page*: the ideal number of pages in the victim page list (The initial value is δ);
*shrink_size*: the number of page shrinking each time ;

**if** *System Start* **then**
    **while** $nr\_victim\_page < nr\_ideal\_victim\_page$ **do**
        ***kshrinkd*** starts;
        $shrink\_size = nr\_ideal\_victim\_page$ ;
    ***kshrinkd*** sleep;

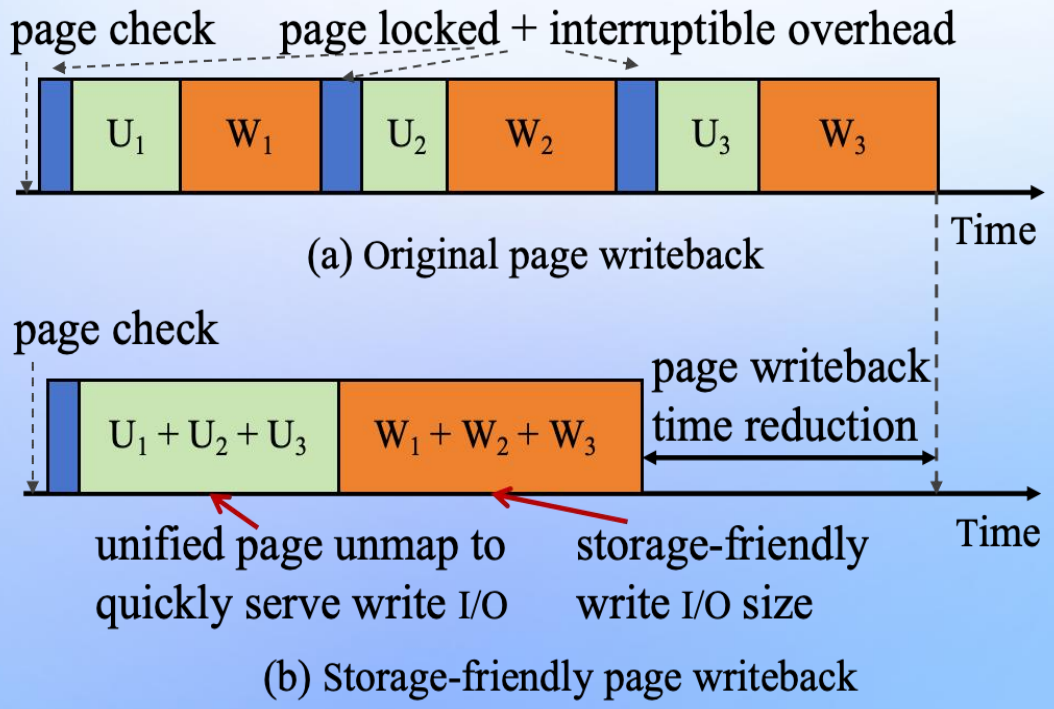
Once page writeback is required, provide victim pages;
**if** *Memory Swapping or Direct Reclaim Start* **then**
    **while** $nr\_victim\_page < nr\_ideal\_victim\_page$ **do**
        ***kshrinkd*** starts;
        $shrink\_size = nr\_to\_reclaim$ ;

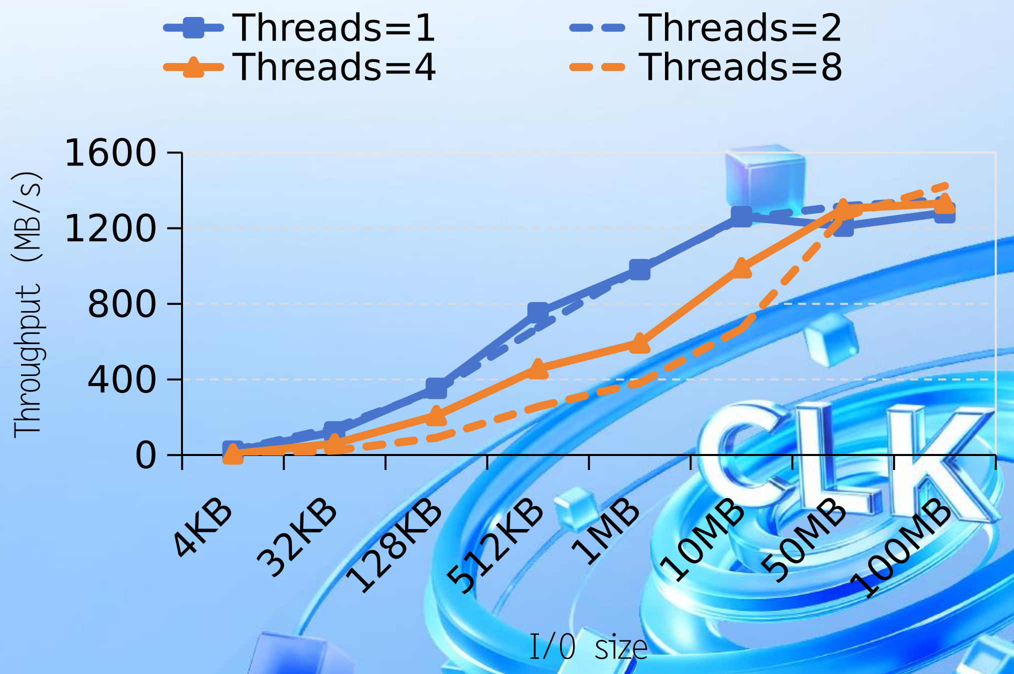
# PMR’s Design

## Storage-friendly Page Writeback(SPW)

**Application-aware Page Unmap:** It performs unified page unmap based on applications to quickly serve write I/Os.

**Batch Write I/Os:** It adopts most cost-effective I/O size based on the characteristics of flash-based storage devices.



# Evaluation



## Experiment Setup

**Evaluation Platforms and Workloads:** We adopt different real mobile devices and installed the pre-selected 36 applications.

**Evaluated Schemes:** Five schemes are implemented and measured to show the effectiveness of PMR.

Table 1: Mobile Devices Under Evaluation.

| Model | Google Pixel 5 | Redmi Note 11 | Google Pixel 6 pro |
|---|---|---|---|
| CPU | Qualcomm Snapdragon 765G | MediaTek Dimensity 810 | Google Tensor |
| Memory | 8 GB | 6 GB | 12 GB |
| Storage | 128 GB/UFS 2.1 | 128 GB/UFS 2.2 | 256 GB/UFS 3.1 |
| System | Android 13 (Kernel 5.10) | Android 13 (Kernel 5.10) | Android 13 (Kernel 5.10) |
| Announced | 2020, September | 2022, January | 2021, October |

Table 2: Applications and automated user interaction.

| Category | Foreground Applications | Auto user inputs |
|---|---|---|
| Browser | Chrome | Browse/Read posts |
| Social Network | Facebook, Twitter | Browse/Read posts |
| Multimedia | YouTube, Tiktok | Watch videos |
| Business Utility | Amazon, Gmap, Uber, Spotify | Browse and search<br>Listen music |
| Game | Angry Birds | Play a stage |

∗Background applications: Browser (Firefox, Opera), Social Network (WhatsApp, Instagram, Skype, WeChat, LinkedIn), Multimedia (Spotify, MXPlayer, Netflix, Capcut), Online shopping (Taobao, eBay, AliPay, BOA, Paypal), Business Utility (Booking, Gmail, New York Times, BBC News, OfficeMobile, GoogleDrive), and Game (Hill Climb Racing, Boom Beach, ClashRoyale, Call of Duty).

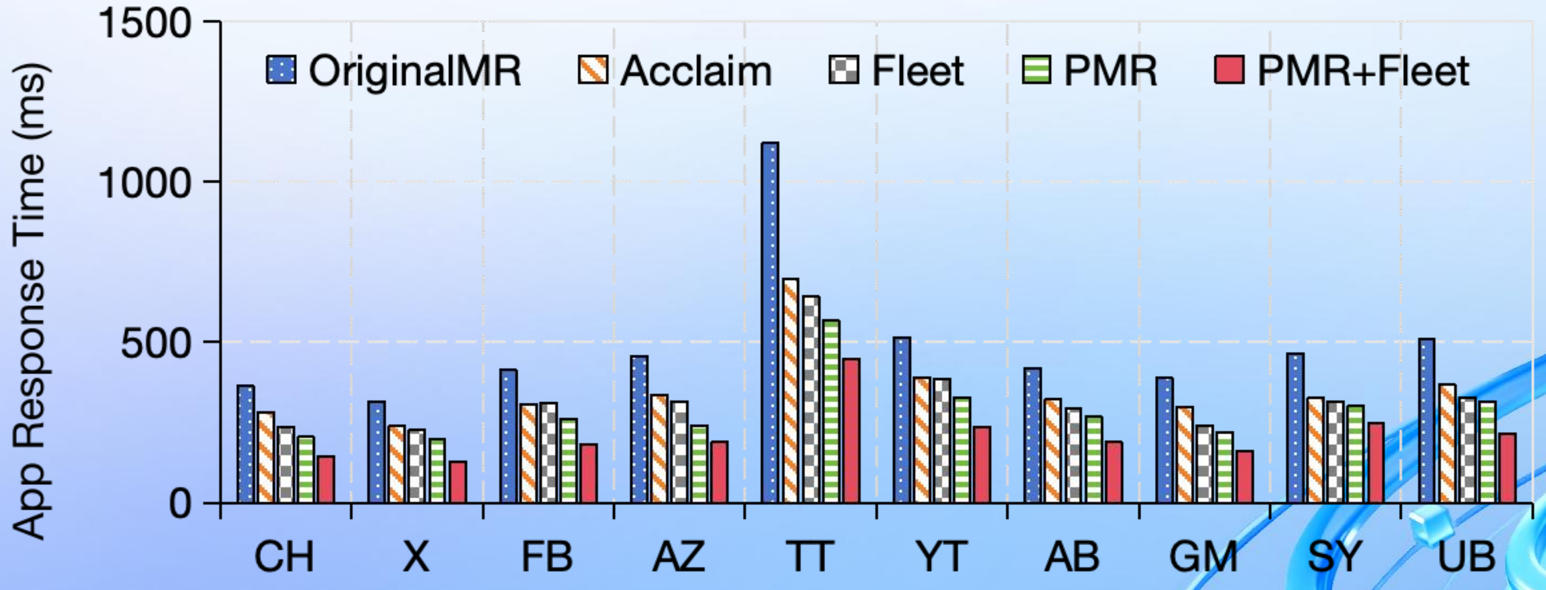
# Evaluation



## Application Response Evaluation

**Significant performance improvement:** The application response time decreases 43.6% compared with OriginalMR.

**Good compatibility:** When PMR and Fleet are used together, application response time is reduced by 38.9% compared to Fleet alone.



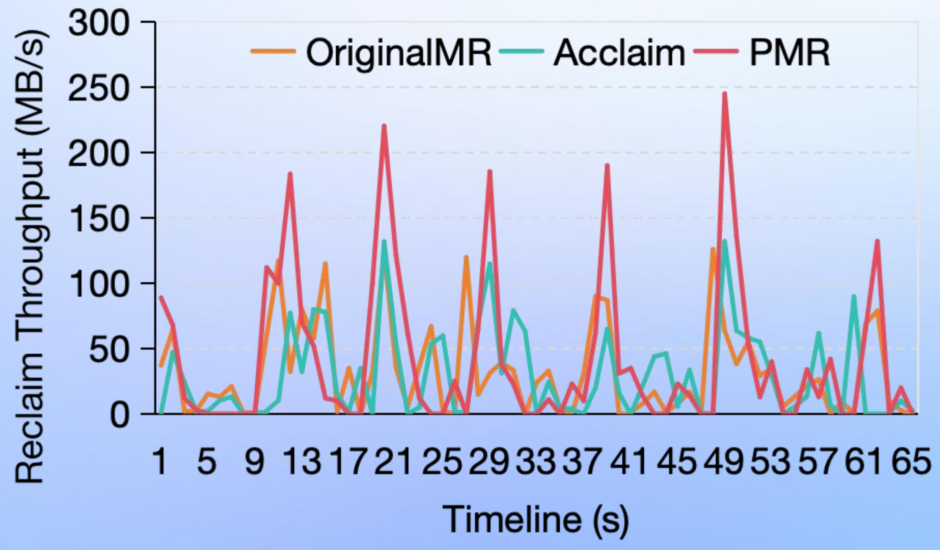

Application response time among different memory reclaim schemes on Google Pixel 6 pro

# Evaluation

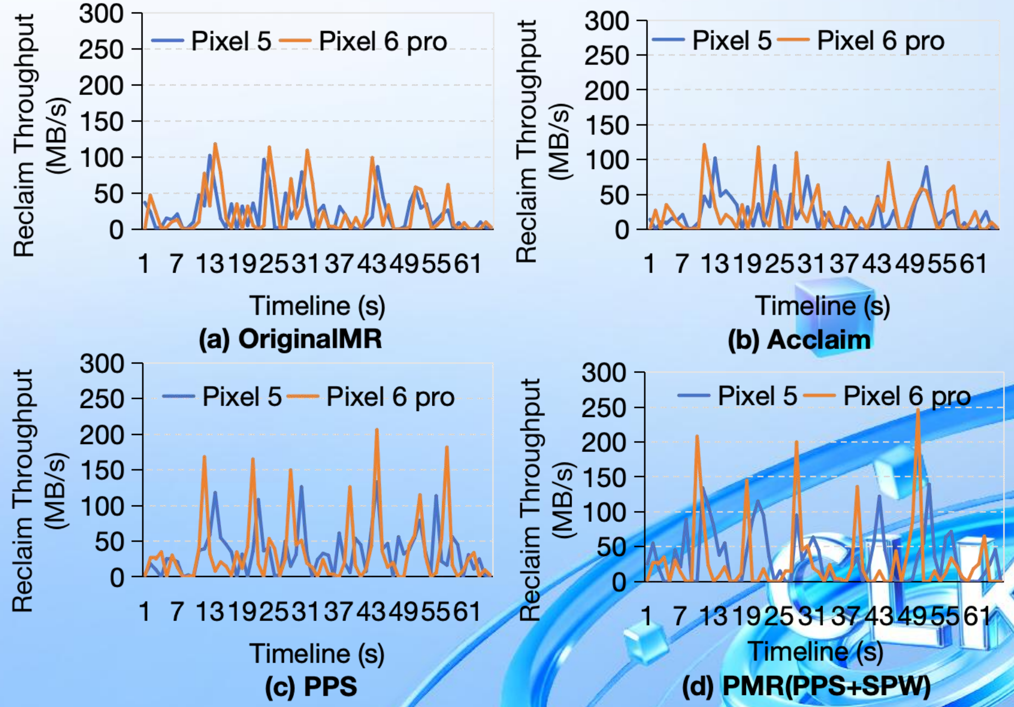
## Memory Reclaim Evaluation

**Reclaim Throughput Improvement:** The peak throughput of PMR is increased by 82.8% and 75.5% respectively compared with OriginalMR and Acclaim.

**Storage Device Utilization:** With PMR, the average throughput on the Pixel 6 pro is 65% of that on the Pixel 5.




Memory reclaim throughput among different schemes on Google Pixel 6 pro




Memory reclaim throughput for different schemes under different mobile devices

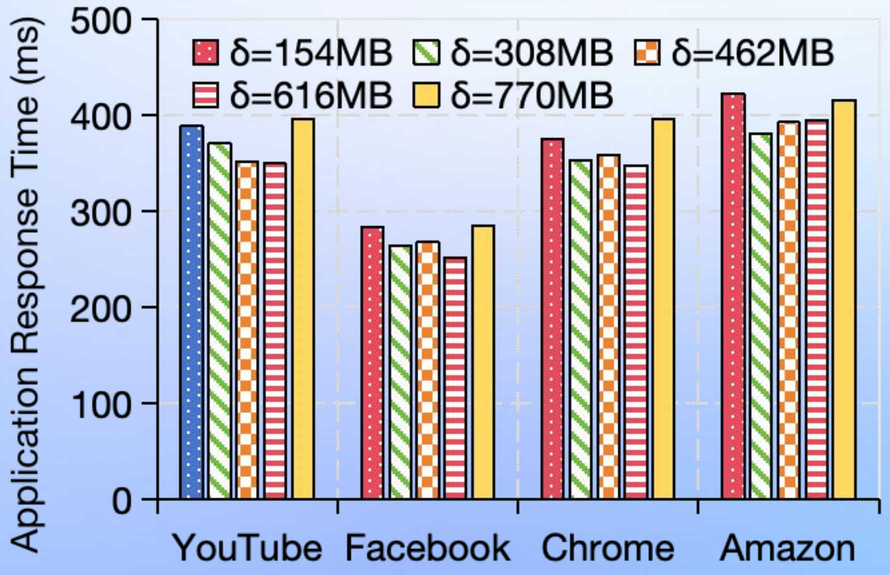
# Evaluation



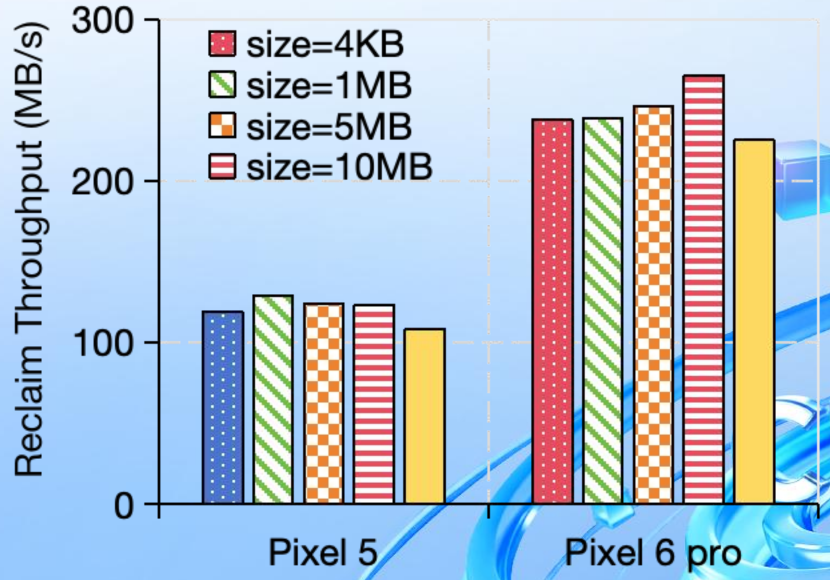
## Sensitivity Study

**The ideal pages in the victim page list:** δ aims to control the capacity of the victim page list, which directly affects the performance of memory reclaim.

**The size of the application-aware page unmap:** The size affects both the performance of the unmap and the efficiency of write I/O.




Application responsiveness by varying δ




Memory reclaim throughput by varying the size of the application-aware page unmap

# Conclusion

Motivation: Sluggish memory reclaim is mainly due to the sub-optimal memory reclaim path, including three aspects: (1) Sequential Execution of Page Writeback and Page Shrinking; (2) Inefficient Page Shrinking; (3) Internally Blocked Page Writeback.

Design: PMR parallelizes key steps of memory reclaim, including Proactive Page Shrinking and Storage-friendly Page Writeback.

Evaluation: Evaluations on real mobile devices with popular application show PMR achieves significant performance improvement.

# THANKS