CacheSifter: Sifting Cache Files for Boosted Mobile Performance and Lifetime

Yu Liang, Riwei Pan, Tianyu Ren, Yufei Cui, Rachata Ausavarungnirun,

Xianzhang Chen, Changlong Li, Tei-Wei Kuo, and Chun Jason Xue

Executive Summary

Problem: Unnecessary writes reduce flash lifetime and system performance

A large part of writes is contributed by cache files

Android systems write all cache files into flash storage

Not all cache files need to be written back for storing persistently

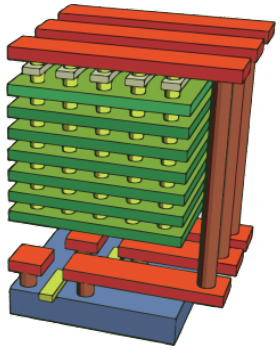
Our goal: To improve both system performance and lifetime of flash storage

CacheSifter: differentiate cache files and treat them according to their reuse behaviors and main-memory/storage usages

CacheSifter can reduce writes to flash storage more than 60% and thus prolong the flash lifetime more than 114% and improve write performance under intensive I/O workloads more than 18%.

Extensive Data Are Written into Flash Every Day!

Total writes of testing users is about 10GB on average and up to 30GB per day!



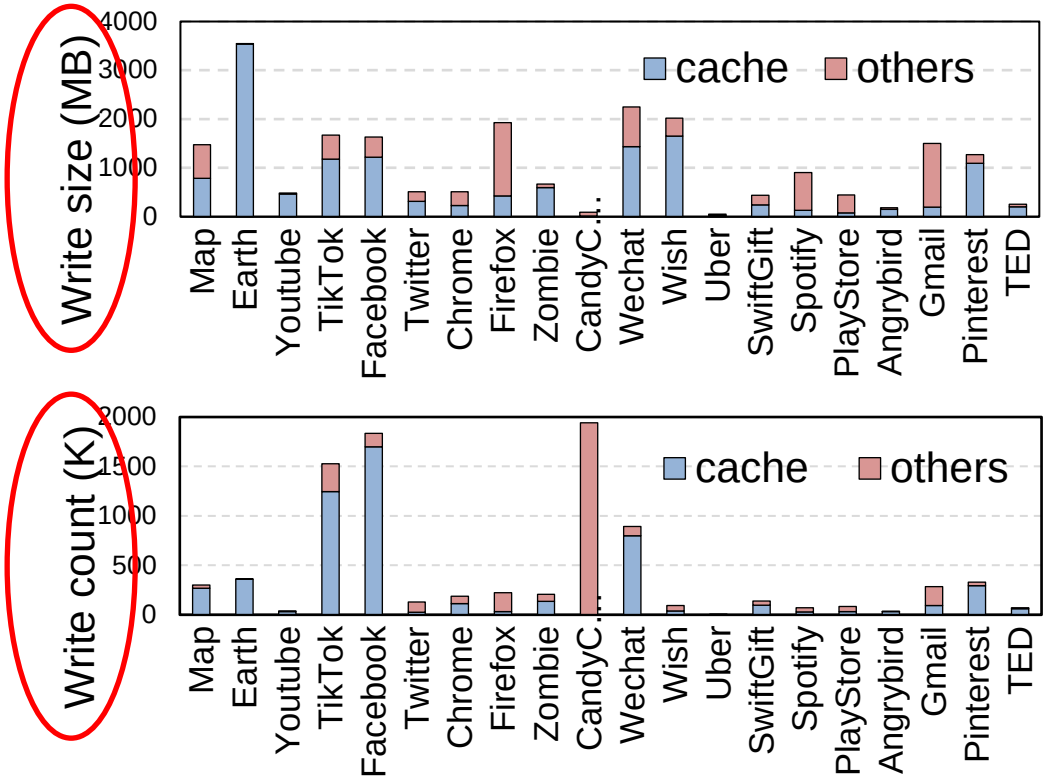
Writes could reduce lifetime of flash storage[1] and system performance[2].

[1] Tao Zhang, Aviad Zuck, Donald E. Porter, and Dan Tsafrir. Apps can quickly destroy your mobile's flash - why they don't, and how to keep it that way (poster). Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, page 207–221, 2019

[2] Congming Gao, Liang Shi, Mengying Zhao, Chun Jason Xue, Kaijie Wu, and Edwin H.-M. Sha. Exploiting parallelism in i/o scheduling for access conflict minimization in flash-based solid state drives. In 30th Symposium on Mass Storage Systems and Technologies (MSST), pages 1–11, 2014.

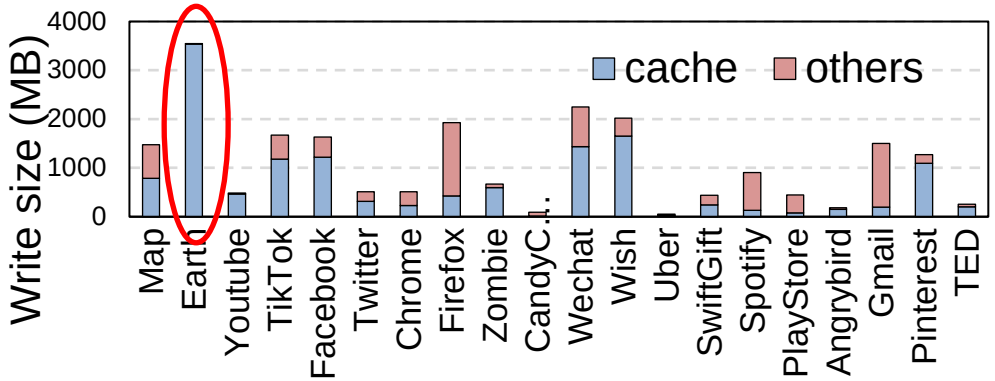
Lots of Cache File Writes

- Cache file writes account for a large part of total writes



Lots of Cache File Writes

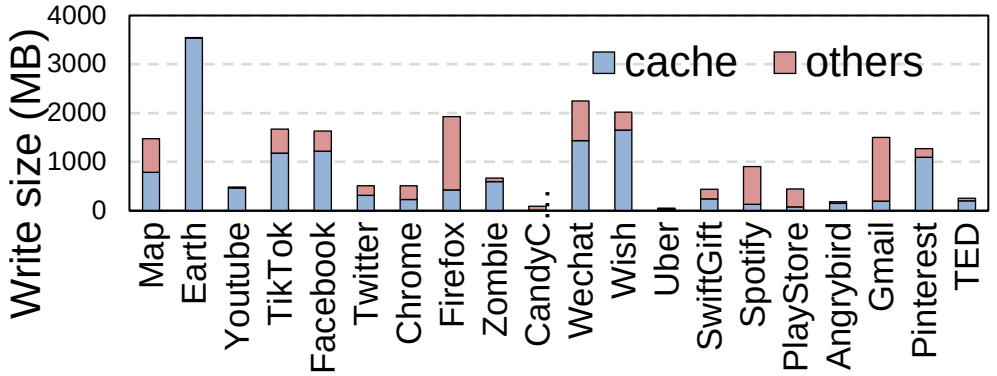
- Many cache file writes are produced within 2 hours



3.5GB/2h

Lots of Cache File Writes

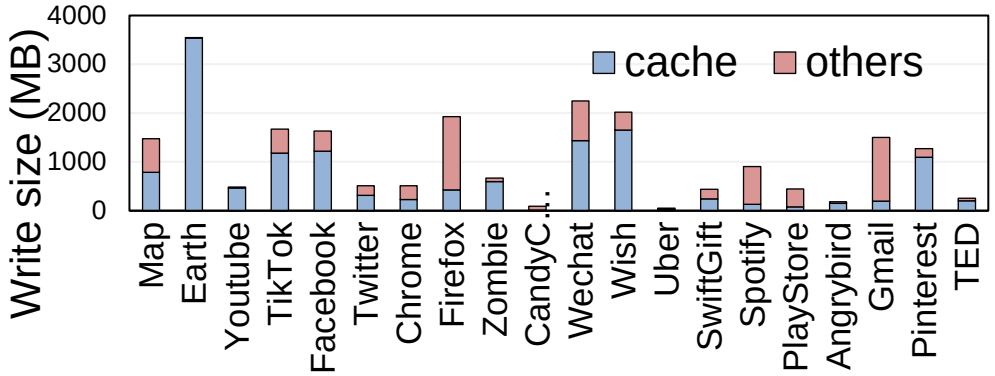
- Cache file writes account for a large part of total writes



Cache file writes represent an average of 64% of total writes.

Lots of Cache File Writes

- Too many cache file writes each day

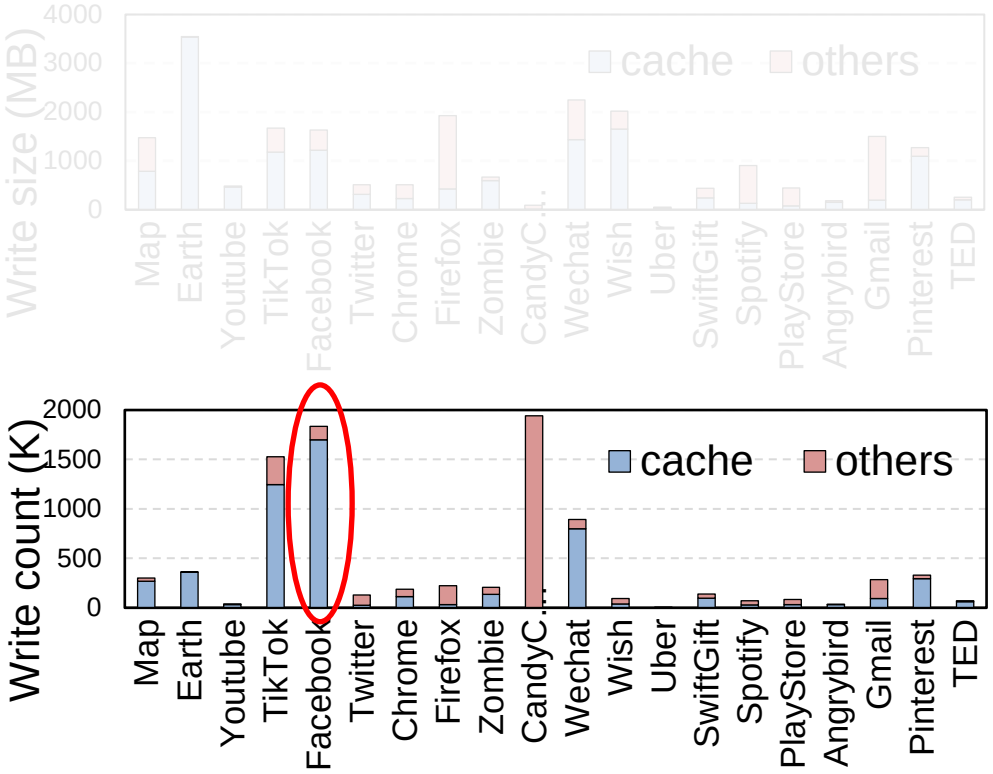


According to our survey, cache file writes are 6.4GB on average and up to 19.2GB each day.

Greatly hurt lifetime of flash storage!

Lots of Cache File Writes

- Many I/O operations for cache file writes in 2 hours



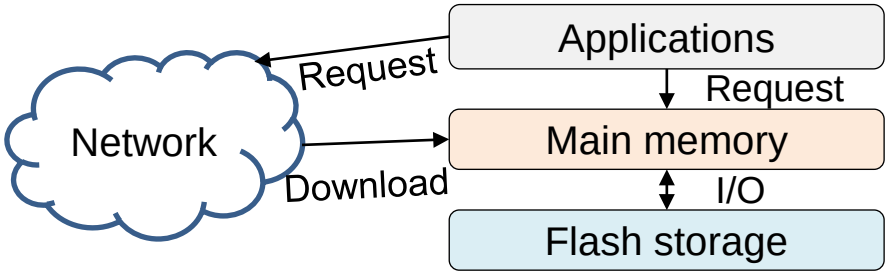
6.4GB on average and
up to 19.2GB each day

1.7 million/2h

Could degrade system performance, especially under extensive I/O workloads!

Source of Massive Cached Data

- Android performance optimization
  - Android systems cache all data to achieve high re-access performance

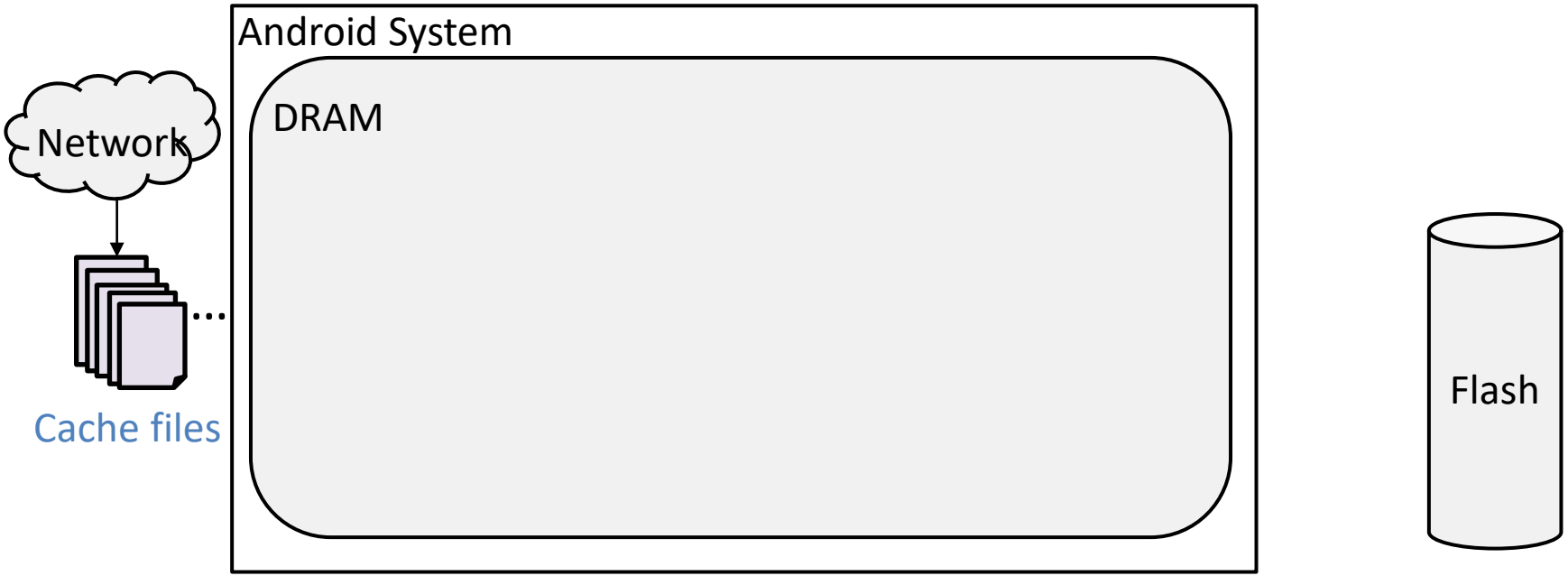


- Related works:
  - Store cache data in main memory[3]
    - Fast access and reduced writes if memory is sufficient
    - Memory is limited in practice
  - Handle cache files differently[4]
    - Still requires a solution

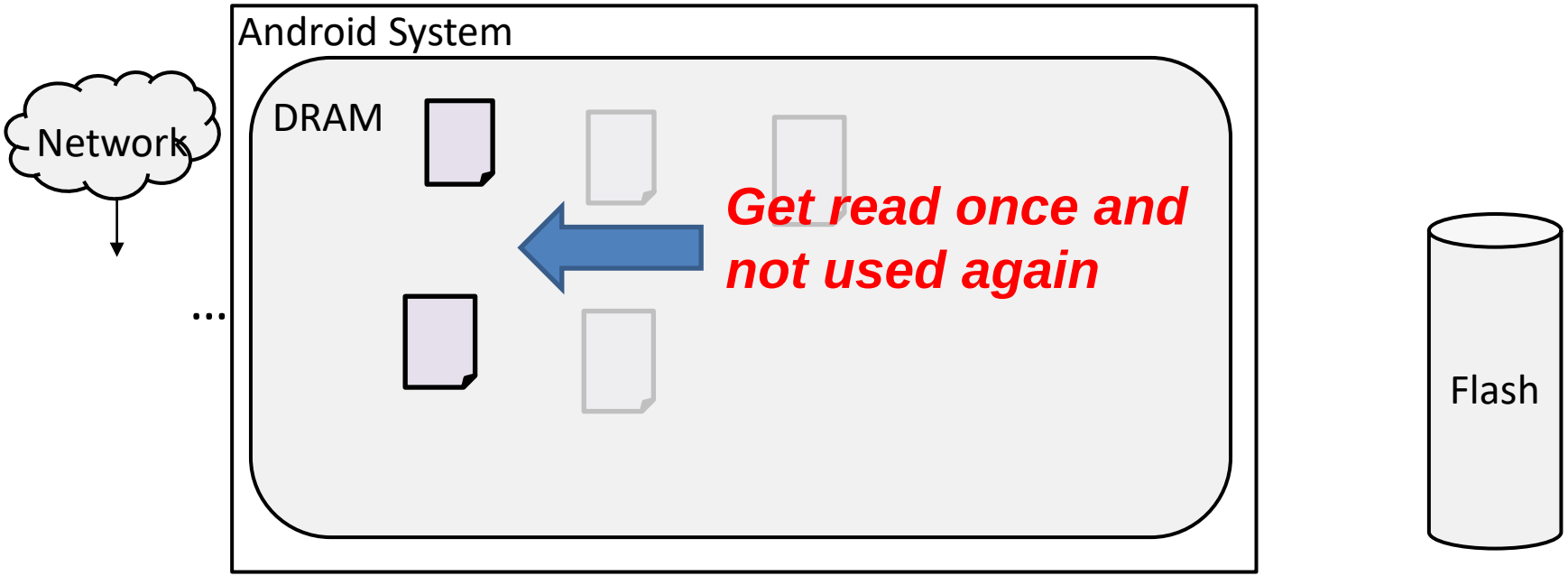
[3] Ngoan Nguyn. Ram disk: an app to mount a folder directly into the ram. https://apkpure.com/ramdisk/com.yz.ramdisk, 2019.

[4] Yu Liang, Jinheng Li, Xianzhang Chen, Rachata Ausavarungnirun, Riwei Pan, Tei-Wei Kuo, and Chun Jason Xue. Differentiating cache files for fine-grain management to improve mobile performance and lifetime. In 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20), July 2020.

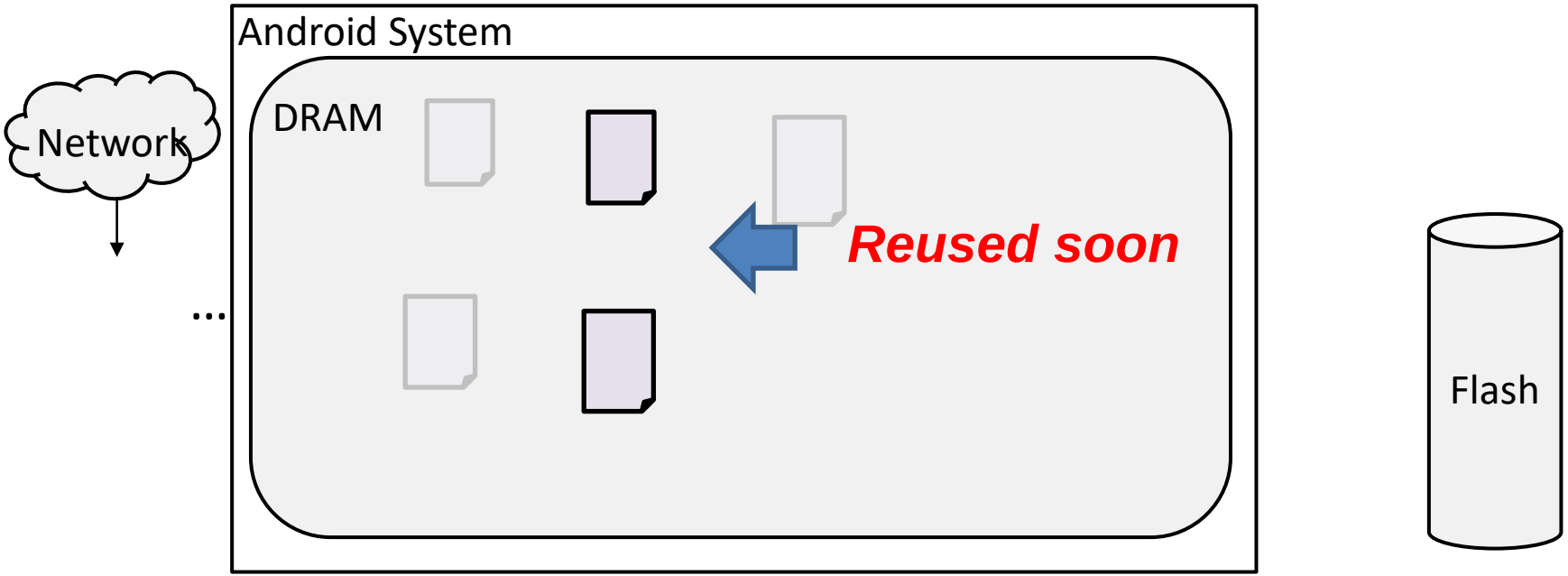
Observation: Cache Files Are Not All The Same



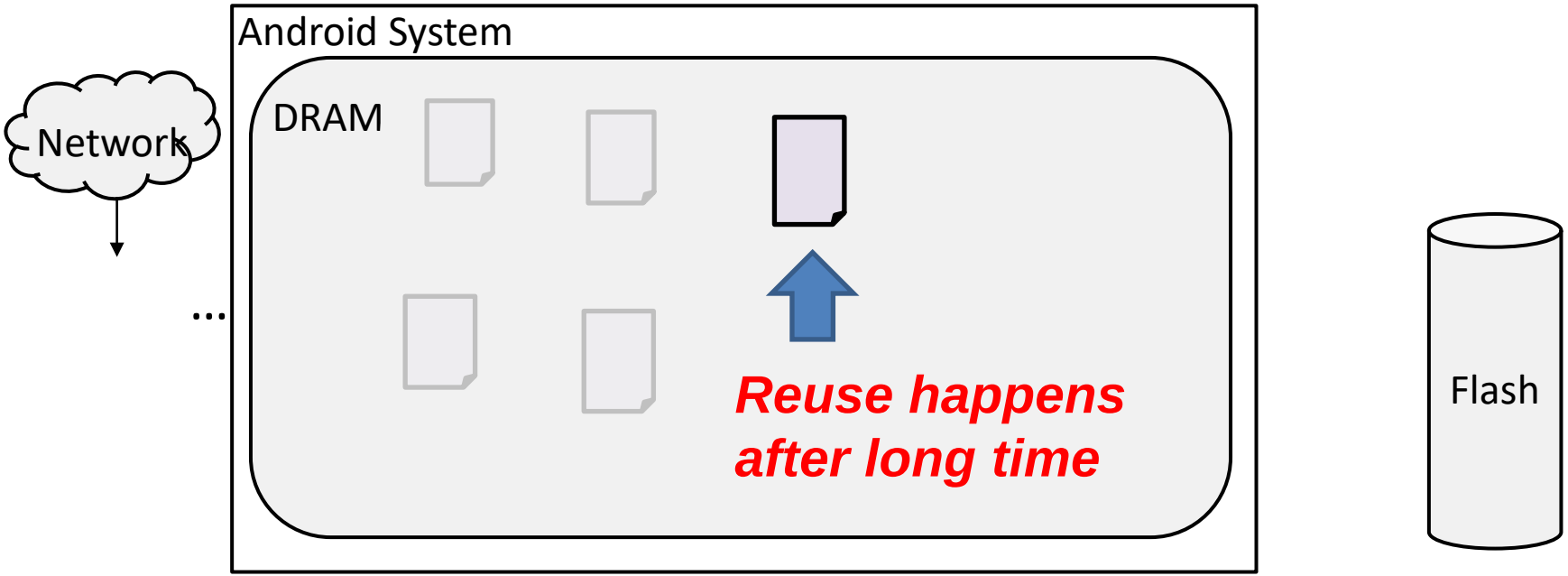
Observation: Cache Files Are Not All The Same



Observation: Cache Files Are Not All The Same



Observation: Cache Files Are Not All The Same



Differentiating File Types

- **Type 1:** Read once and not used again
  - Burn-after-read files (BAR)
  - **Strategy:** Do not need to cache these files

- **Type 2:** Frequent reuse that happen quickly
  - Transient files
  - **Strategy:** Cache them in DRAM → Benefit from the low latency

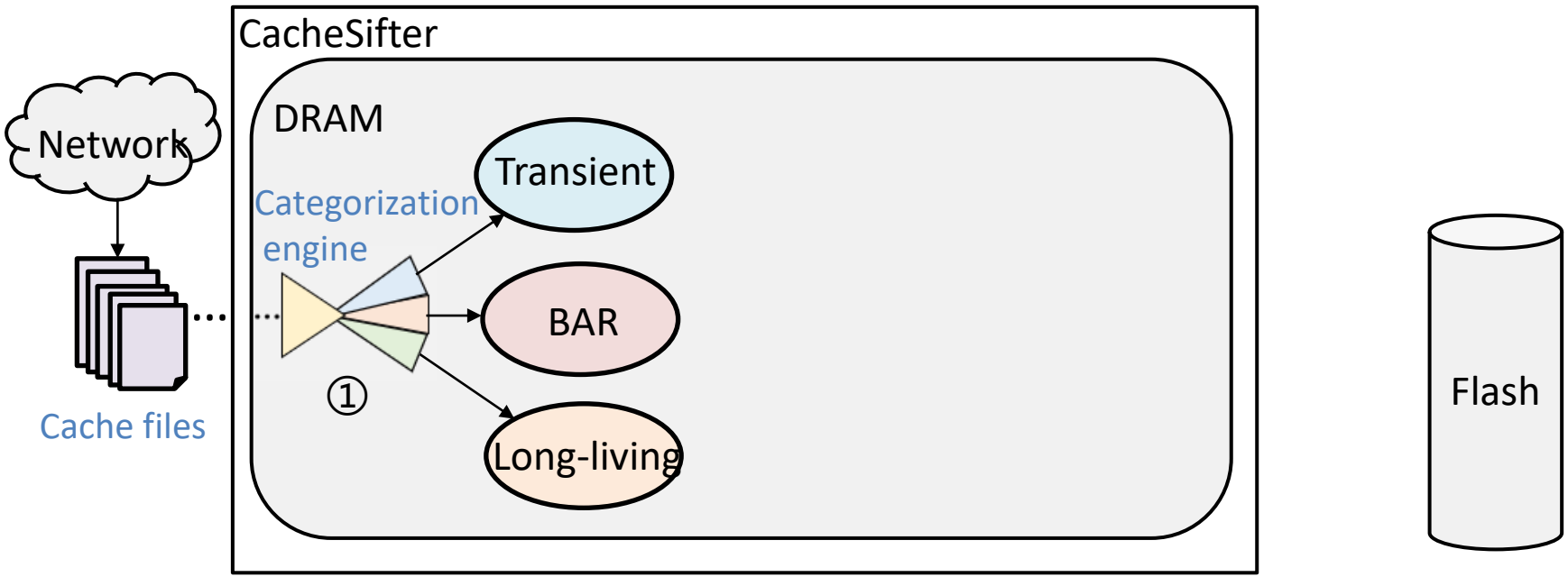
- **Type 3:** Reuse happens long after the first touch
  - Long-living files
  - Do not benefit from DRAM's low latency
  - **Strategy:** Put in storage → Reduce the transfer over the network

CacheSifter's Goal

- Categorize files into three types
  - Fast and application-transparent
- Manage files based on their types
  - Maximize DRAM's utility
- Adapt to changes in user behaviors
  - Recategorize files as needed
- Ensure safety when deleting files

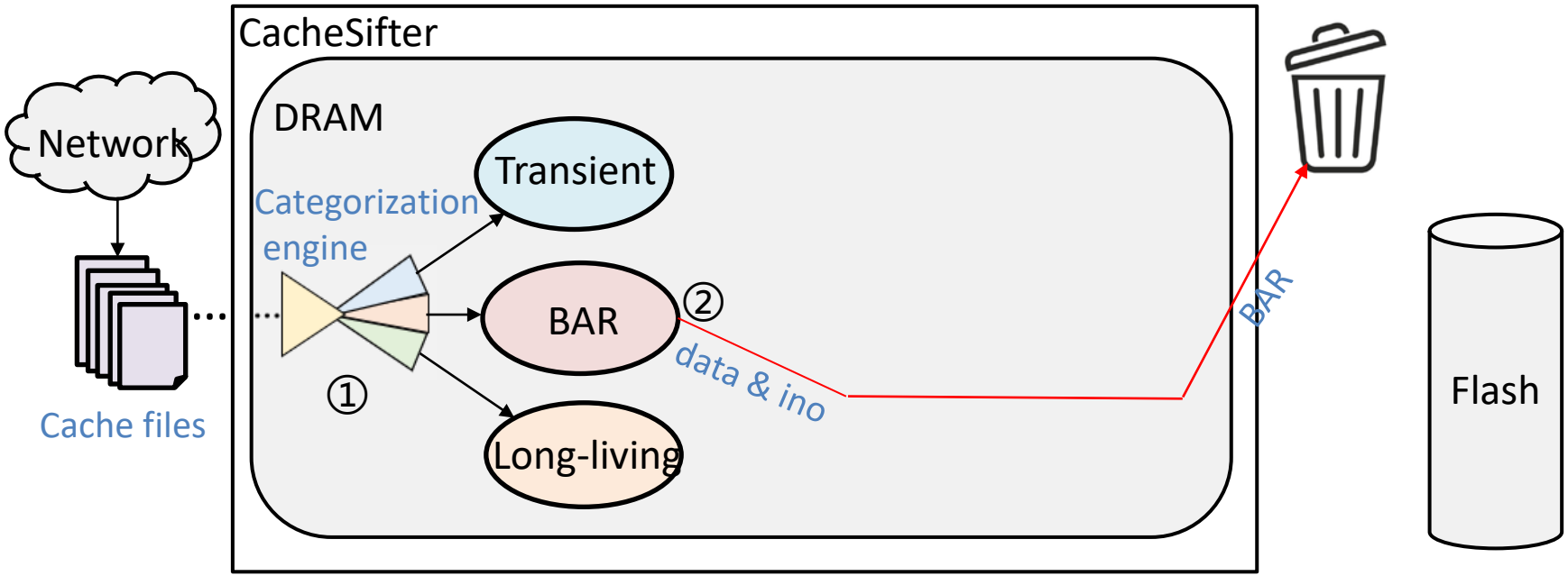
Framework of CacheSifter

- Lightweight ML-based categorization
  - Quickly and adaptively categorization



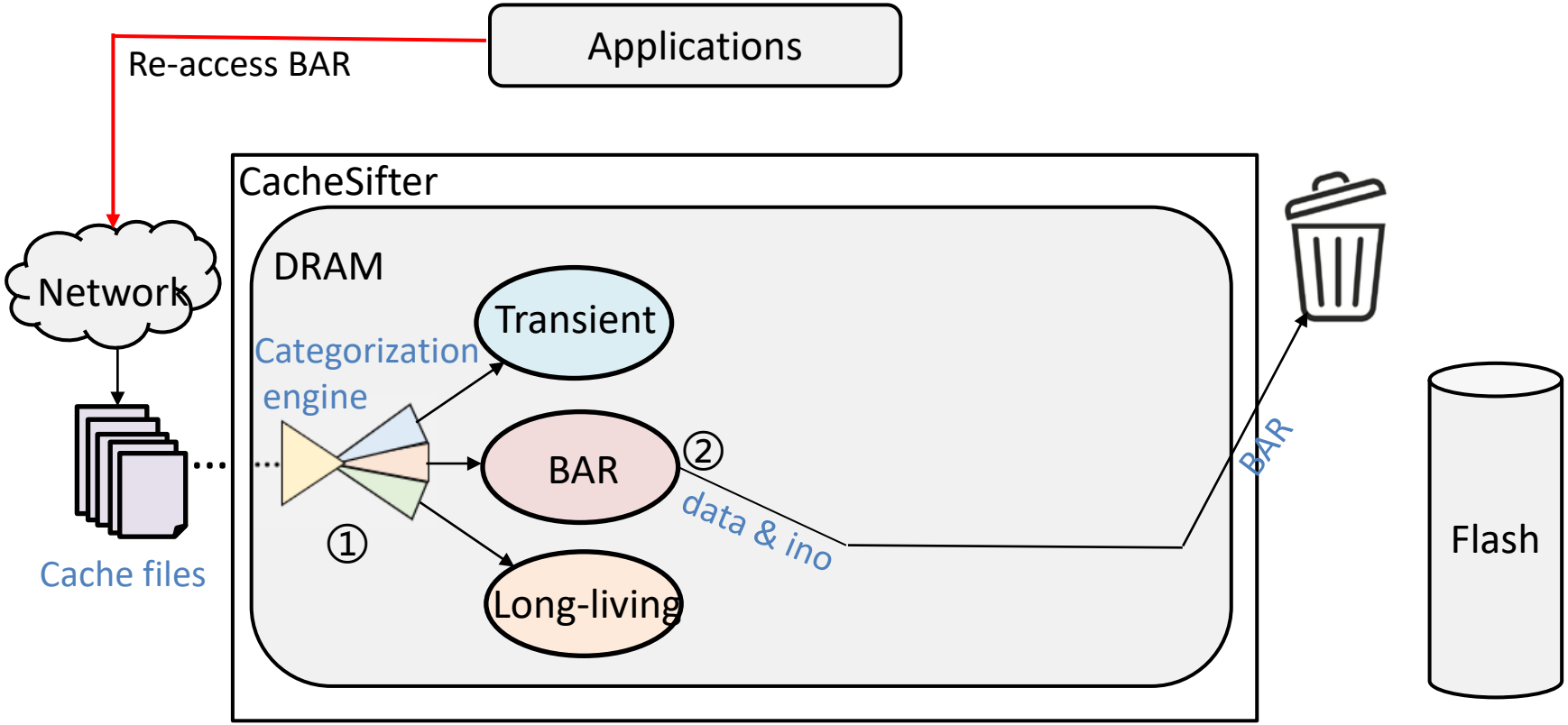
Framework of CacheSifter

- BAR file management
  - Delete them before they are written back to flash



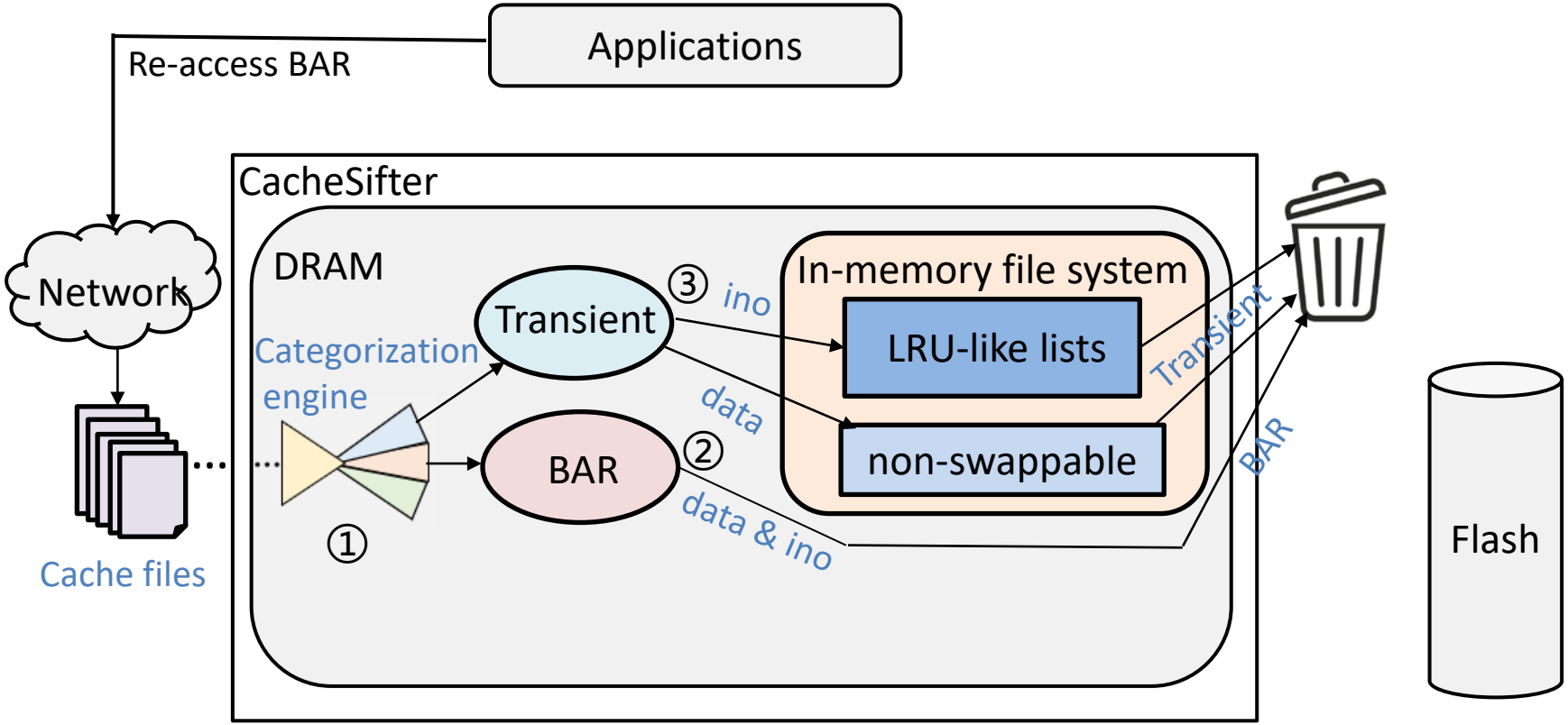
Framework of CacheSifter

- BAR file re-access



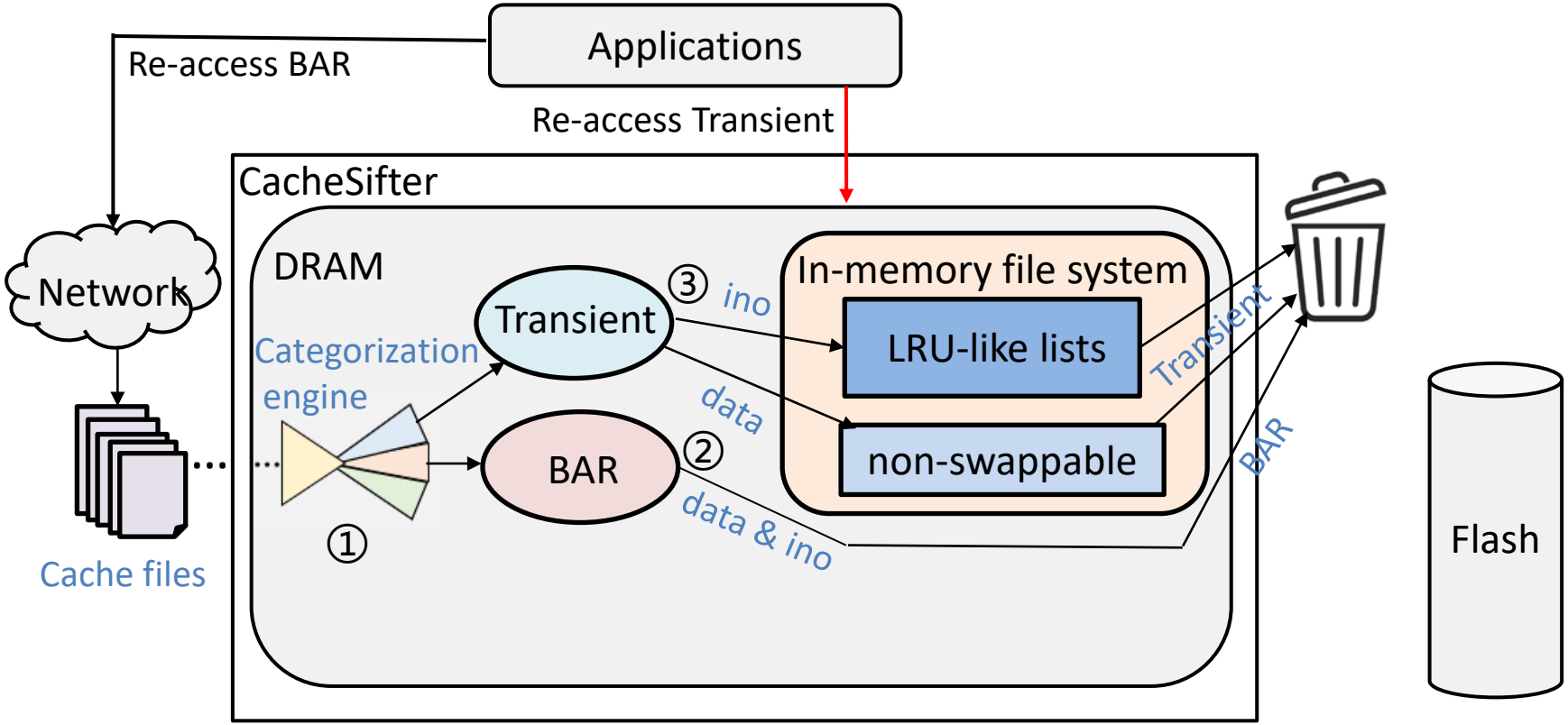
Framework of CacheSifter

- Transient file management



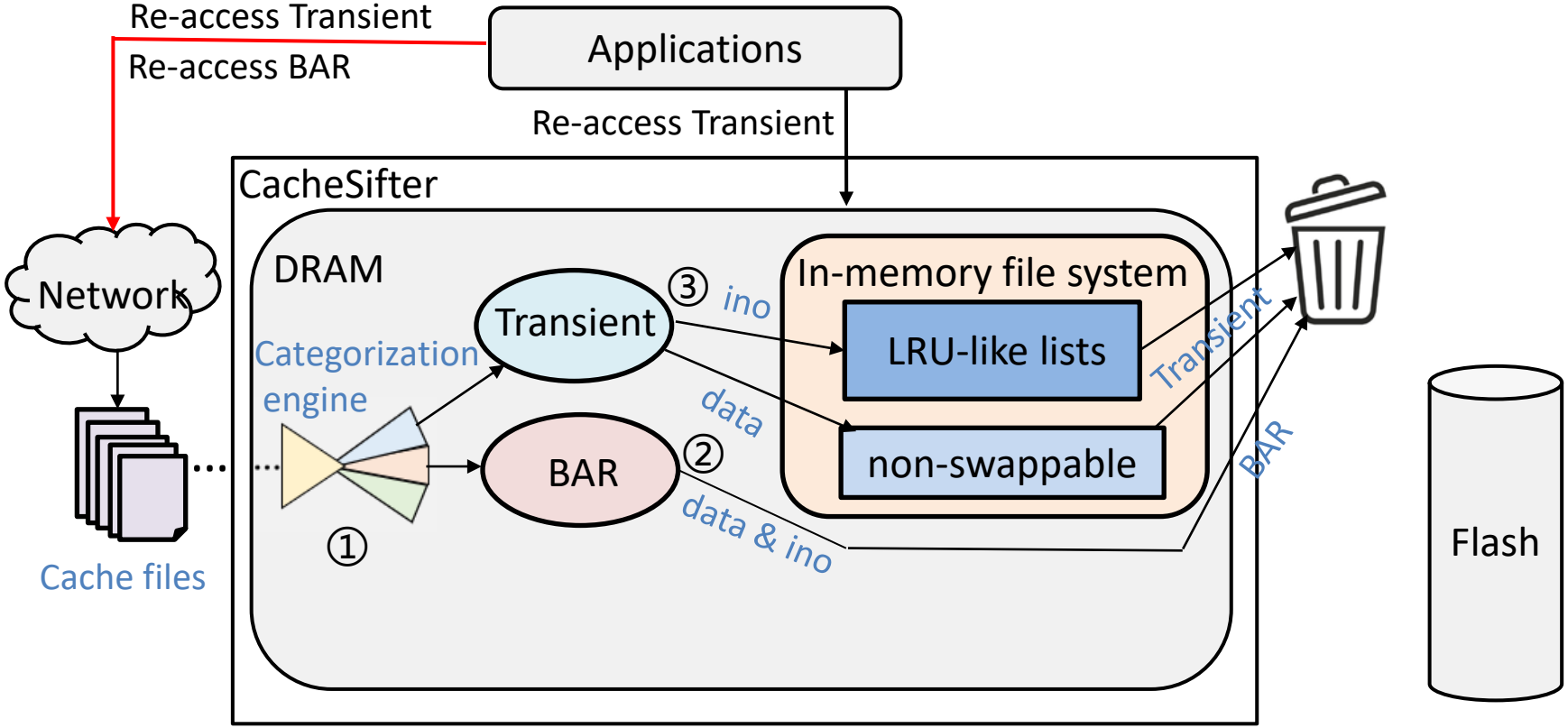
Framework of CacheSifter

- Transient file re-access in memory



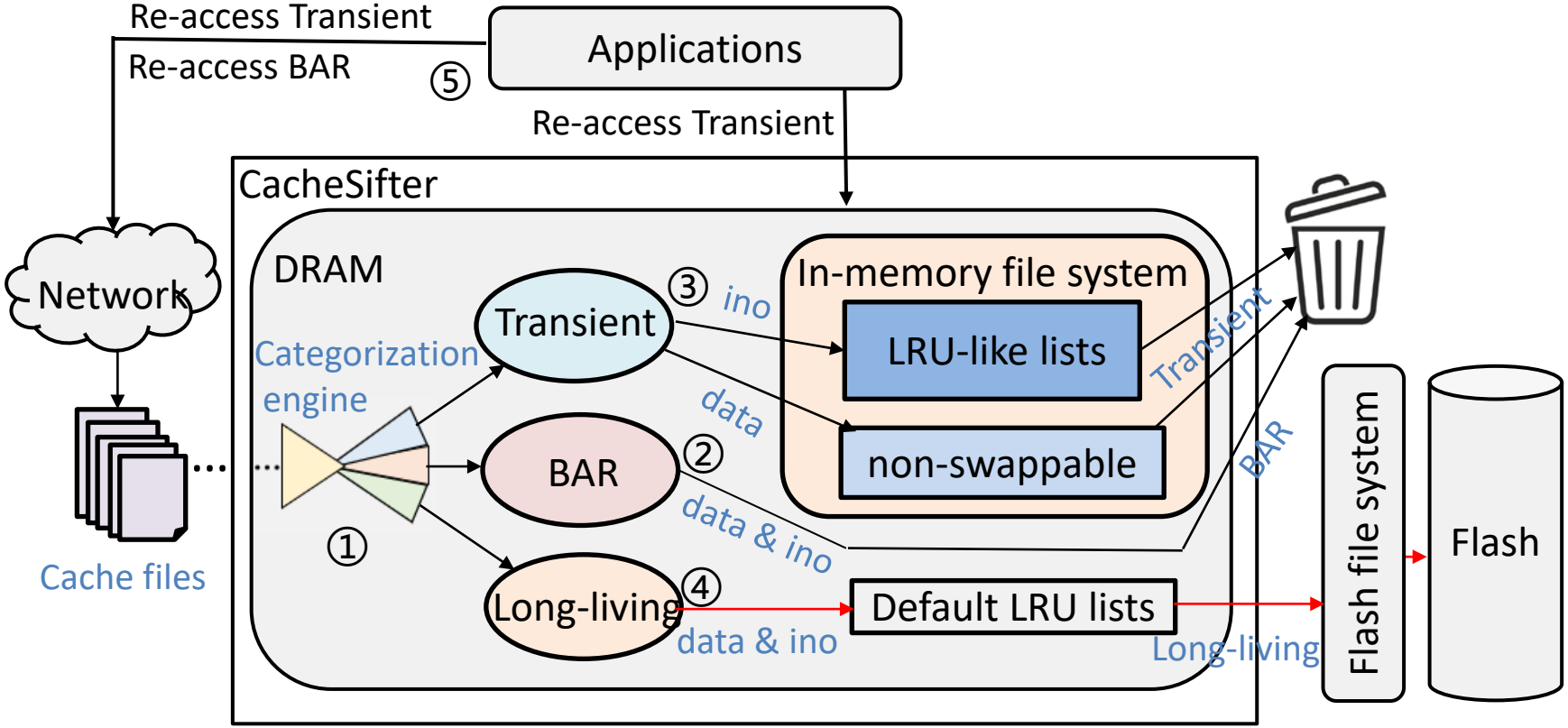
Framework of CacheSifter

- Transient file re-access after deleting



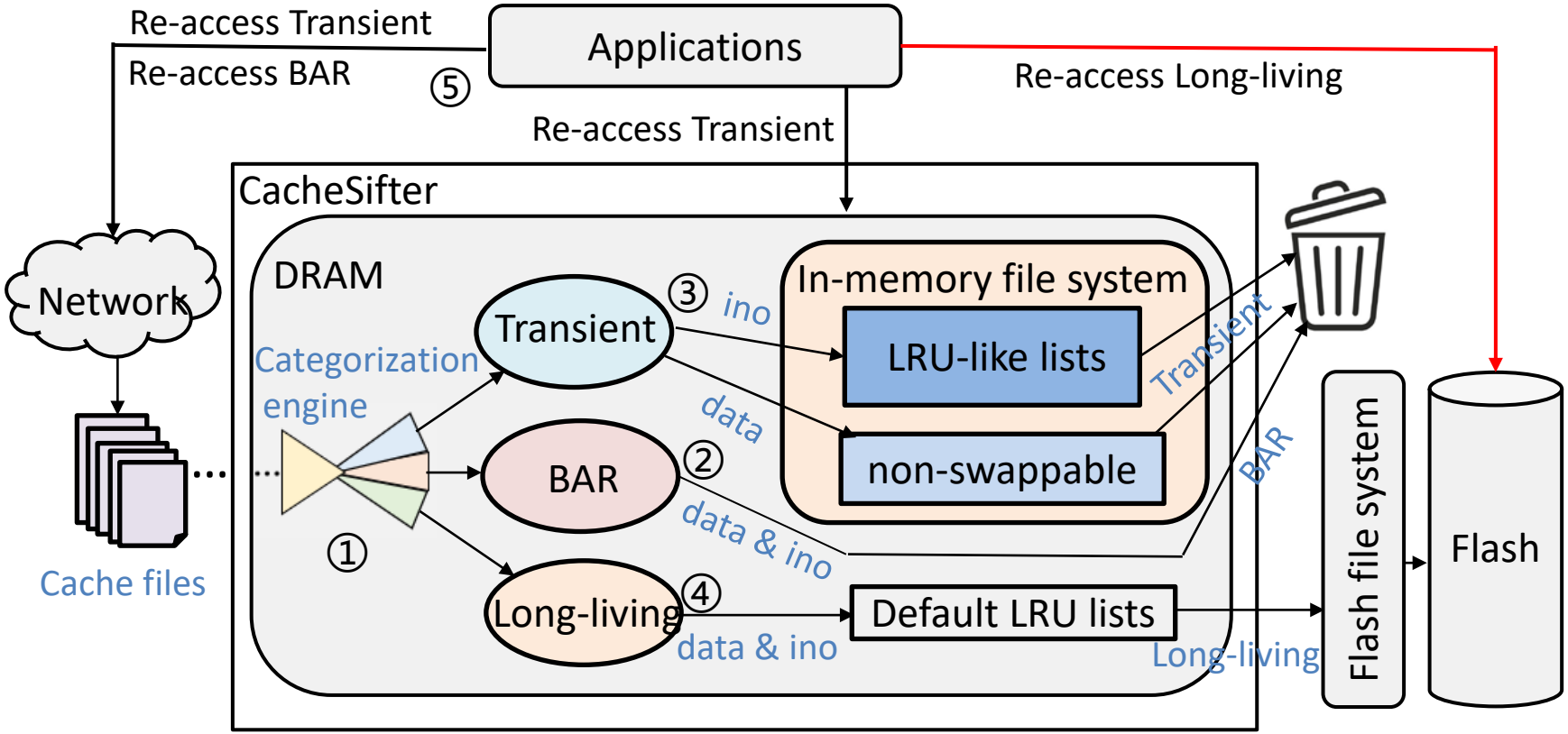
Framework of CacheSifter

- Long-living file management



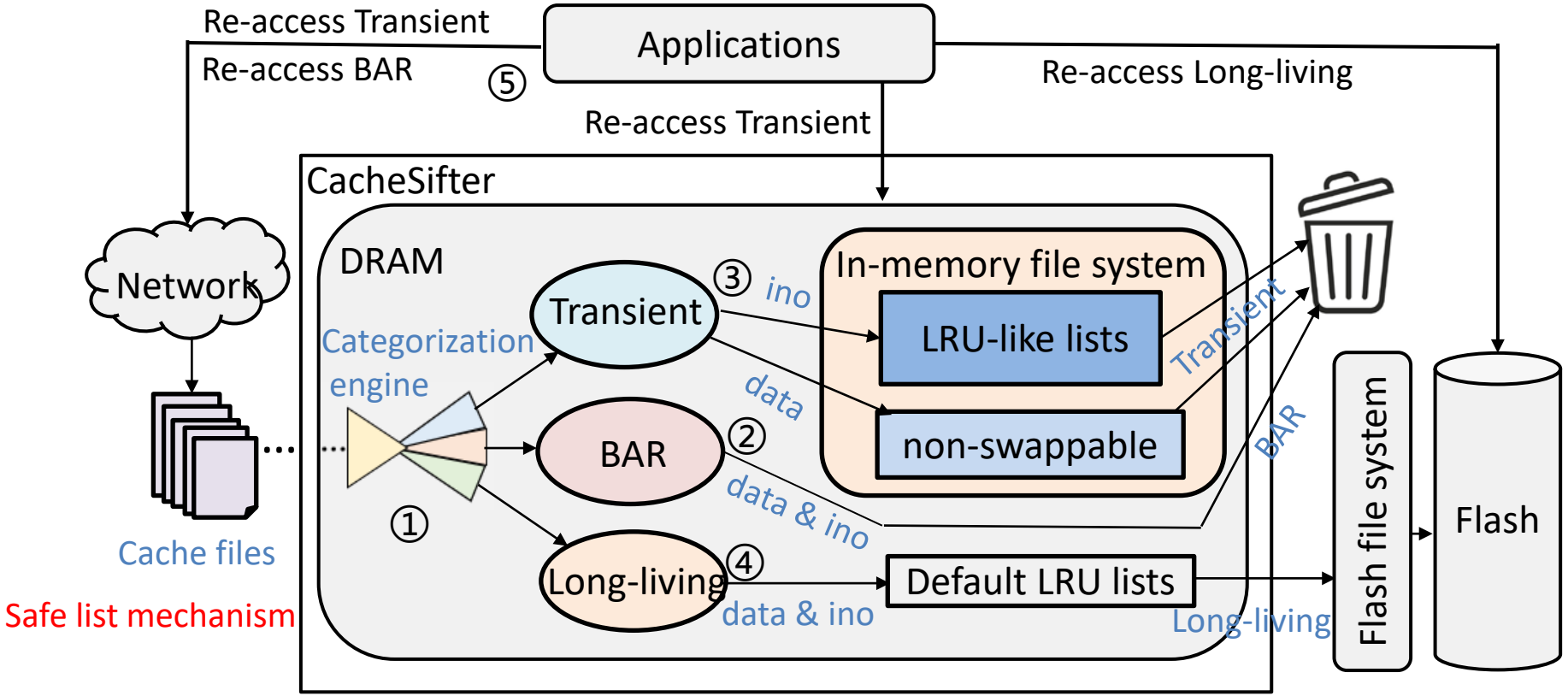
Framework of CacheSifter

- Long-living file re-access



Framework of CacheSifter

- Long-living file re-access

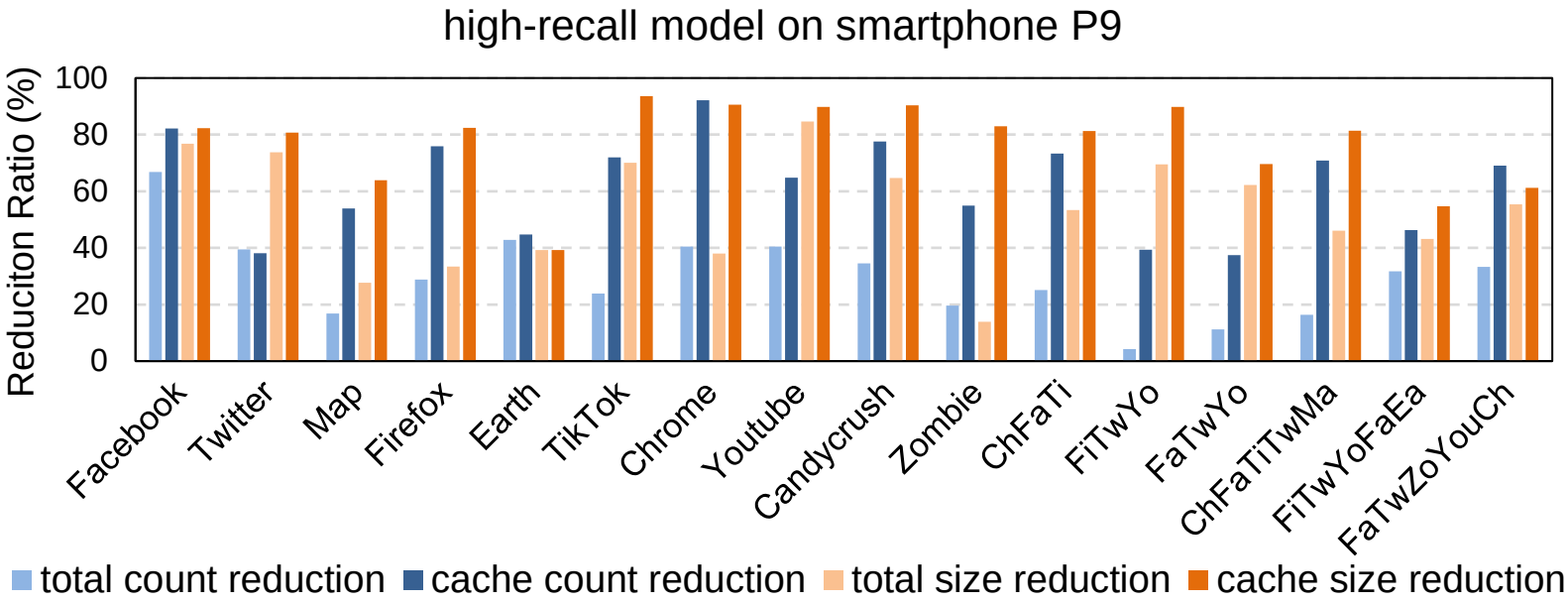


Experimental Setup

- Real smartphones
  - P9
    - P9 equipped with an ARM Cortex-A72 CPU, 32GB internal flash memory and 3GB DRAM running Android 7.0 with Linux kernel version 4.1.18.
  - Mate30
    - Mate30 equipped with an ARM Cortex-A76 CPU, 128GB internal flash memory and 8GB DRAM running Android 10 with Linux kernel version 4.14.116.
- Comparisons
  - Default Android
  - High-accuracy model
  - High-recall model

Evaluation Results: Cache Files' Writes

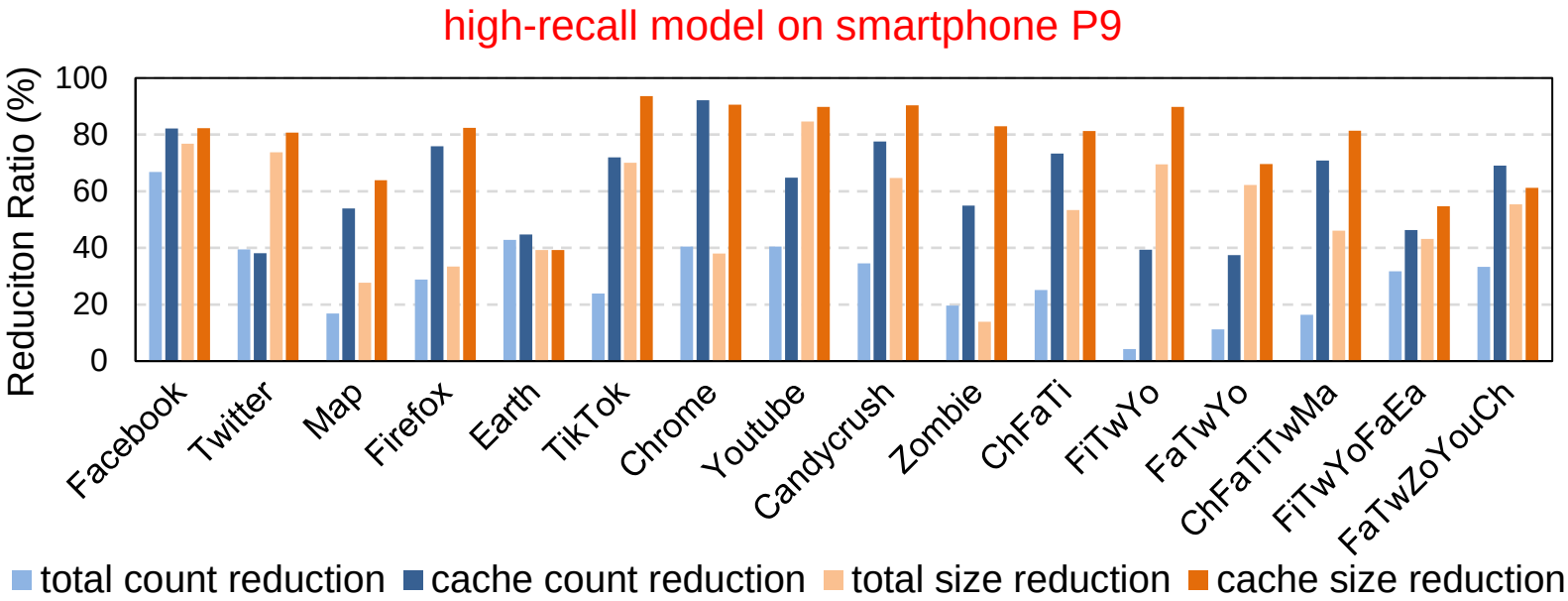
- Reduction in cache file writes and total writes to flash storage.



Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

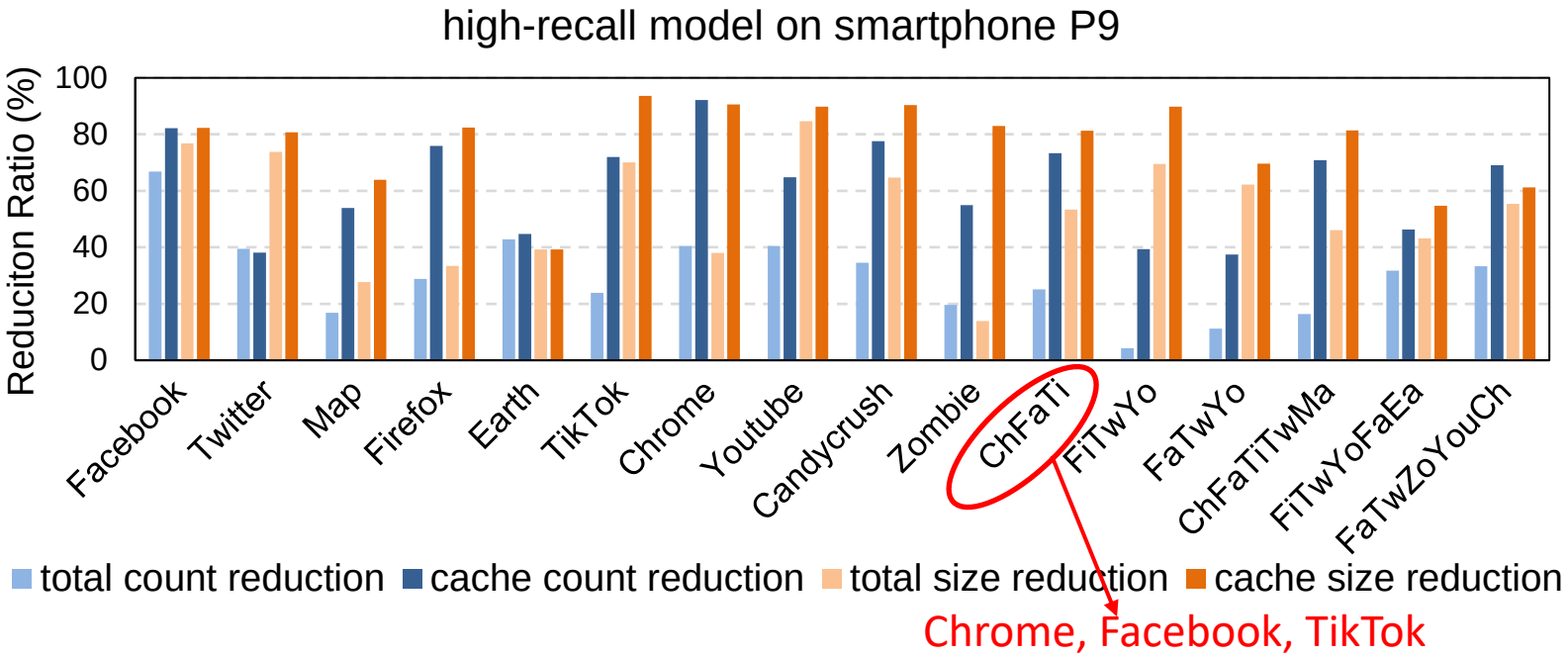
- Reduction in cache file writes and total writes to flash storage.



Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

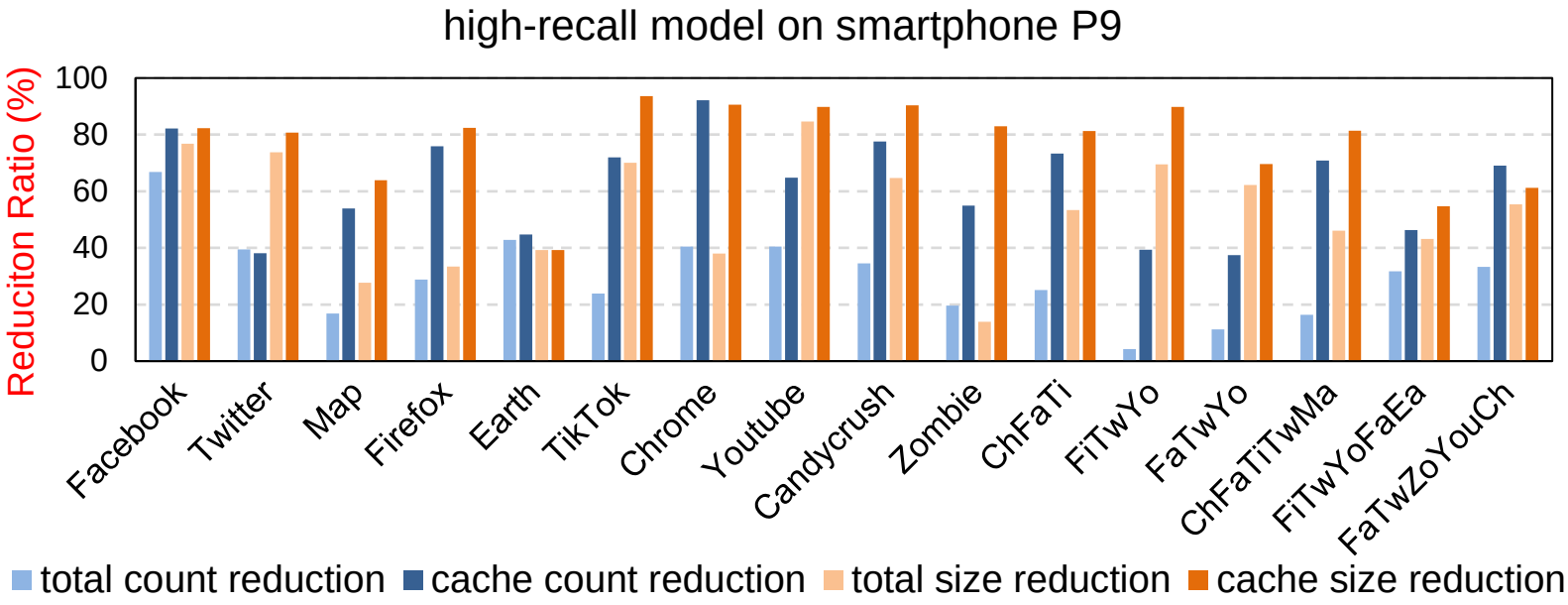
- Reduction in cache file writes and total writes to flash storage.



Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

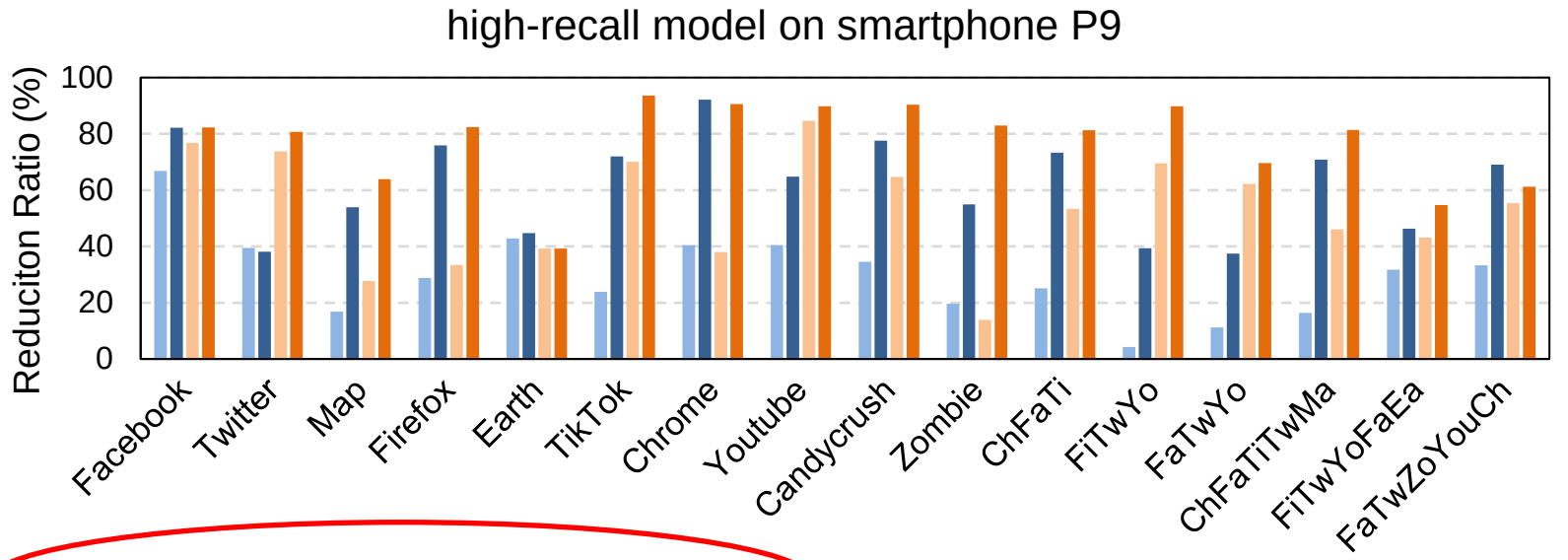
- Reduction in cache file writes and total writes to flash storage.



Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

- Reduction in cache file writes and total writes to flash storage.

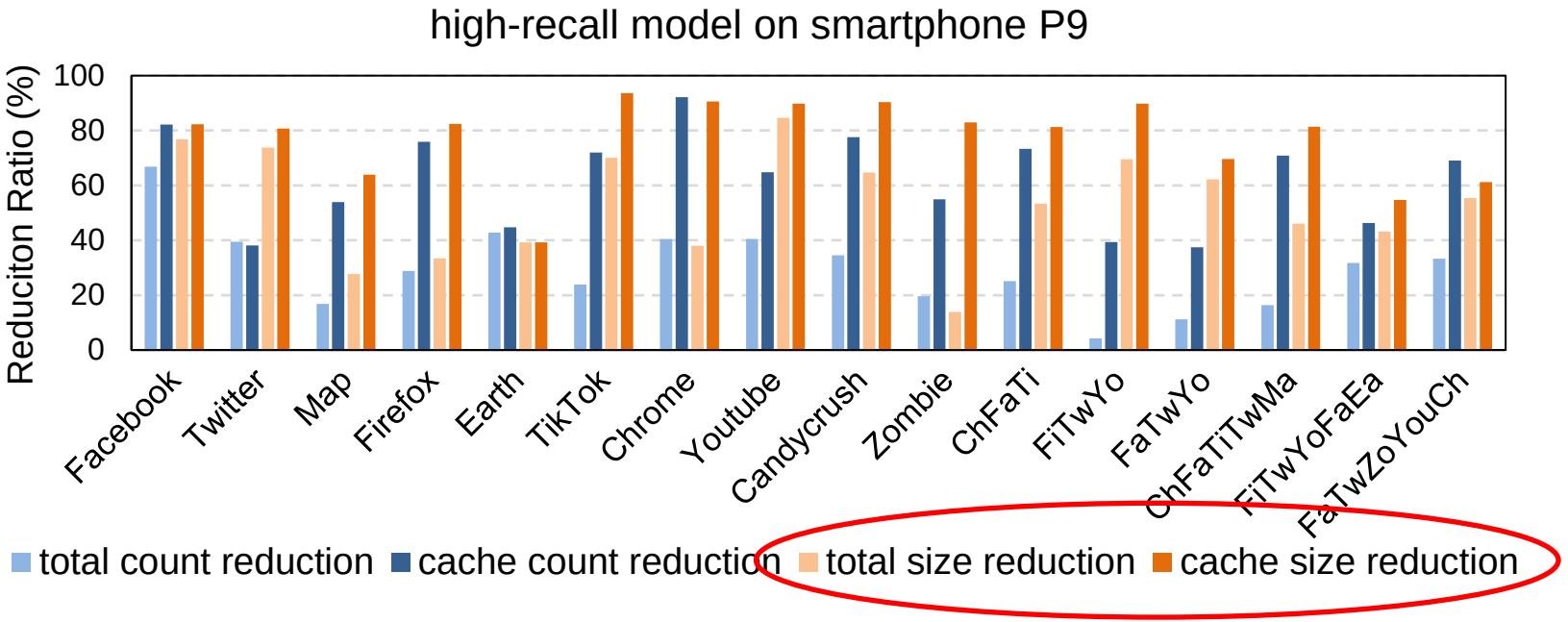


I/O number reduction

Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

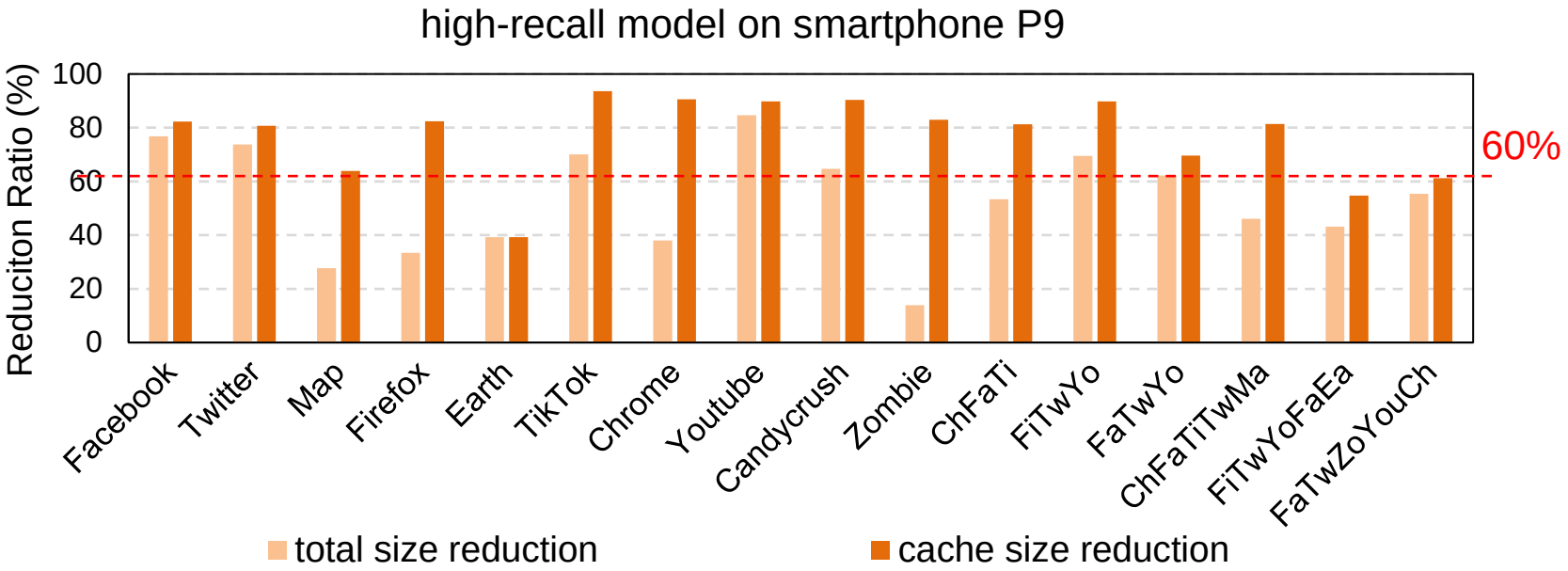
- Reduction in cache file writes and total writes to flash storage.



Normalized reduction ratio of cache files' writes and total I/Os.

Evaluation Results: Cache Files' Writes

- Reduction in cache file writes and total writes to flash storage.

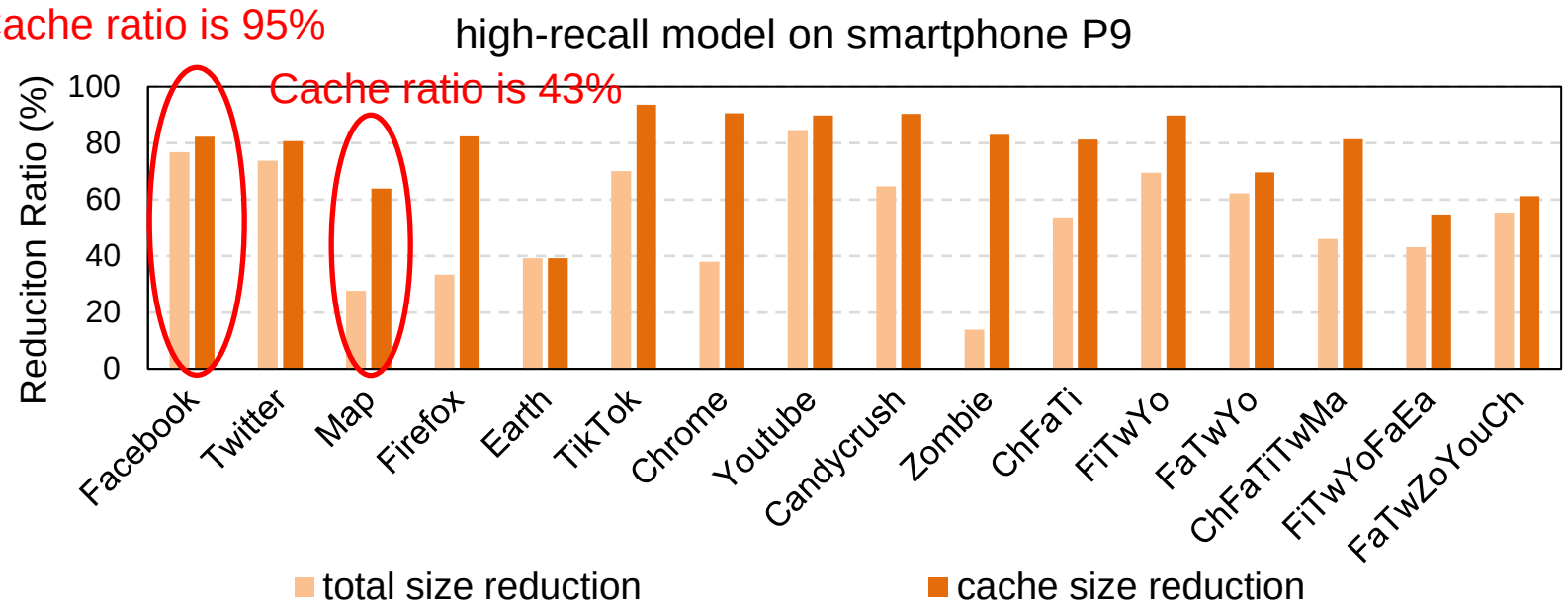


Cache file write can be much reduced.

Total write reduction varies by application.

Evaluation Results: Cache Files' Writes

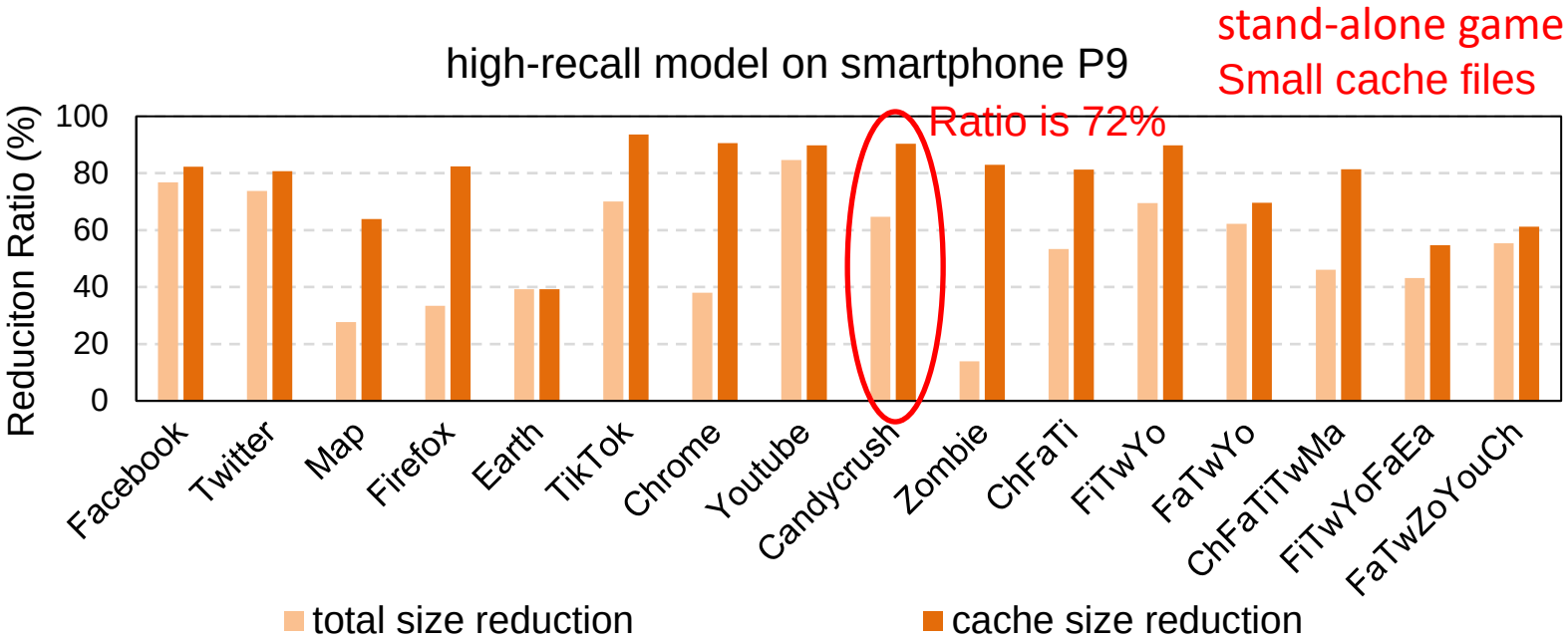
- Reduction in cache file writes and total writes to flash storage.



1) Cache ratio varies by application.
   - Cache ratio = cache writes/ total writes

Evaluation Results: Cache Files' Writes

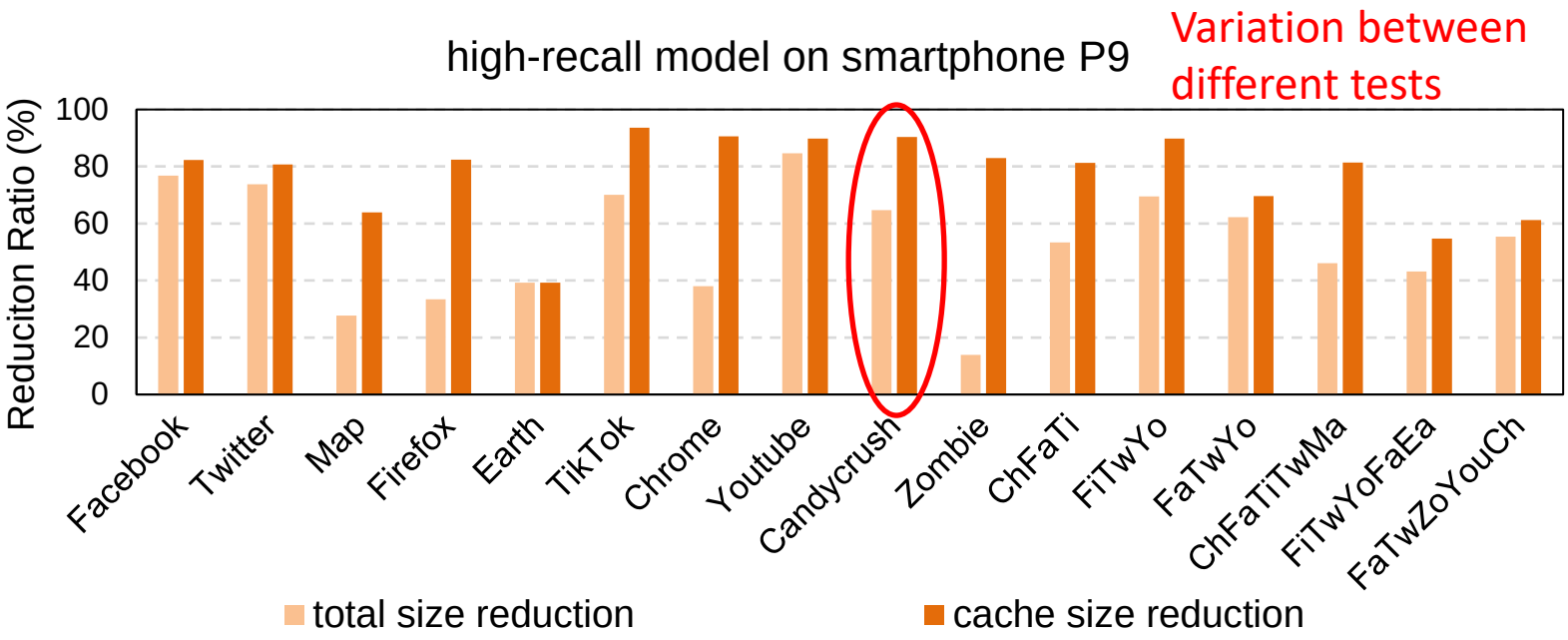
- Reduction in cache file writes and total writes to flash storage.



1) Cache ratio varies by application.
2) Variation between different tests

Evaluation Results: Cache Files' Writes

- Reduction in cache file writes and total writes to flash storage.

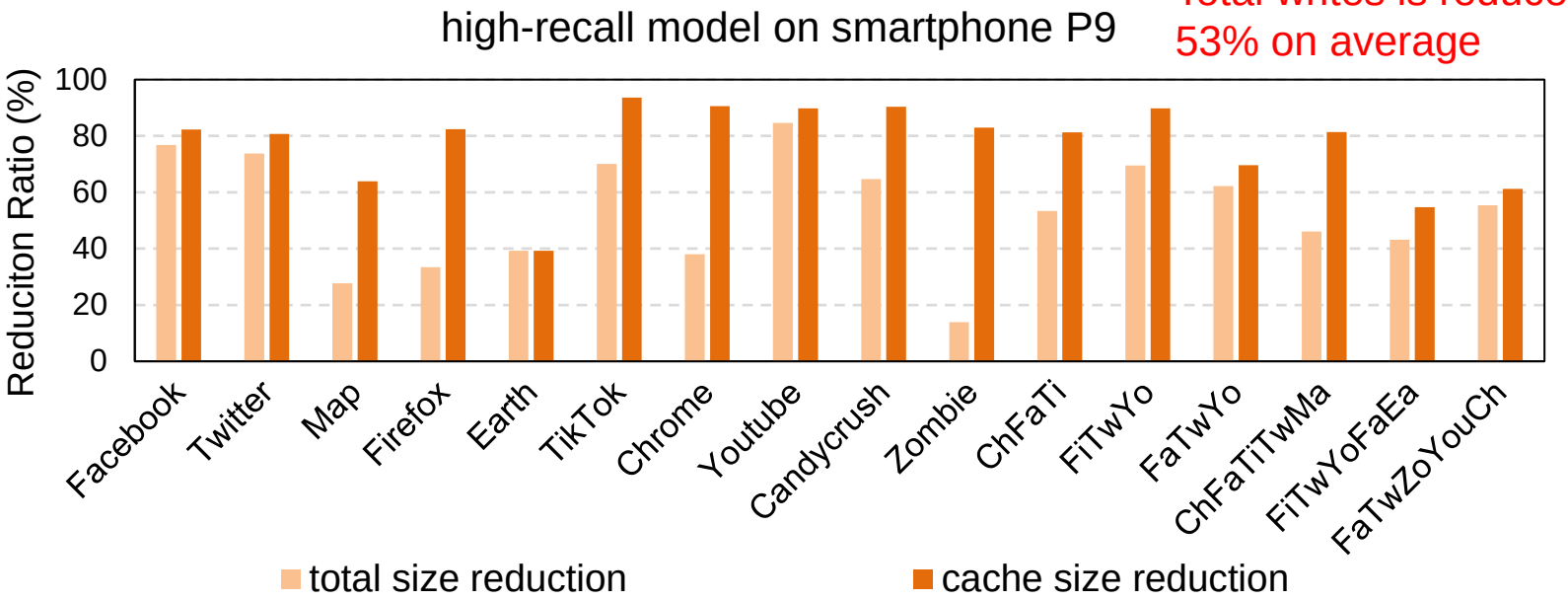


Cannot eliminate the variation by replaying trace because we can not obtain precise traces.

Evaluation Results: Cache Files' Writes

- Reduction in cache file writes and total writes to flash storage.

Total writes is reduced by 53% on average

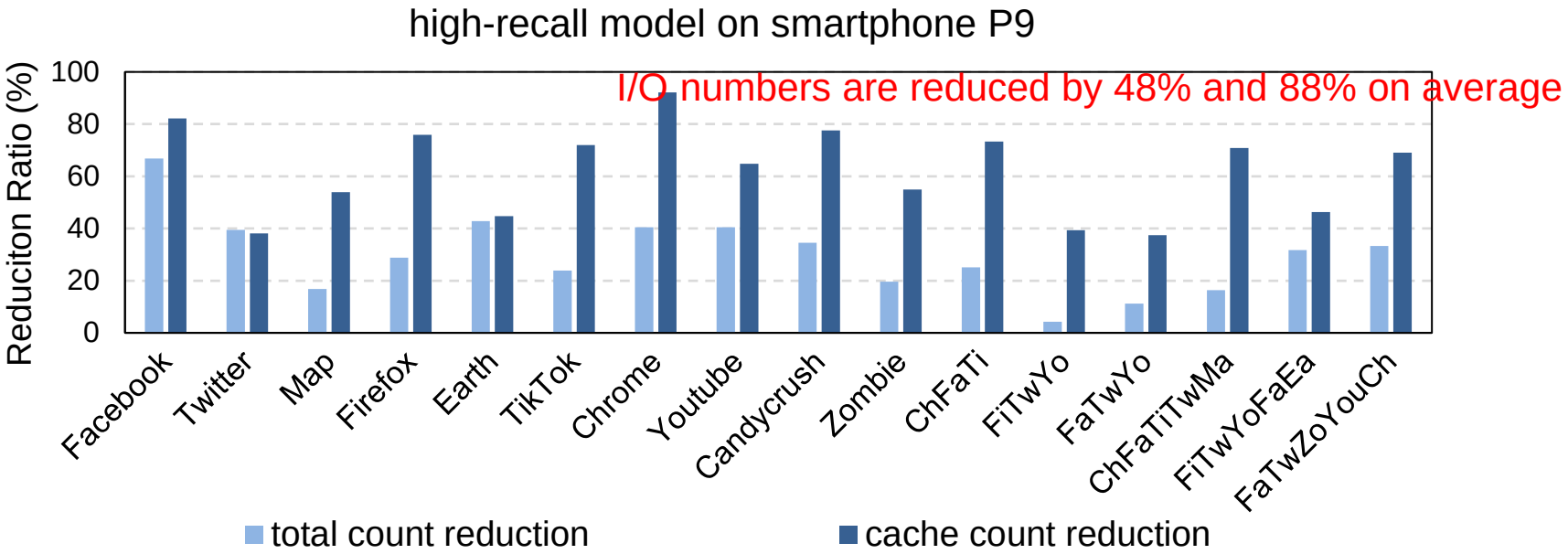


Lifetime is inversely proportional to the total writes.

Lifetime can be improved by an average of 113.7% (1/(1-53.2%)-1).

Evaluation Results: Cache Files' Writes

- Reduction in cache file writes and total writes to flash storage.



Improve system performance due to the reduction on I/O competition, especially under extensive I/O workloads.

Evaluation Results: Overhead

- Overhead of CacheSifter.
  - Network overhead
    - Redownload because of misclassification

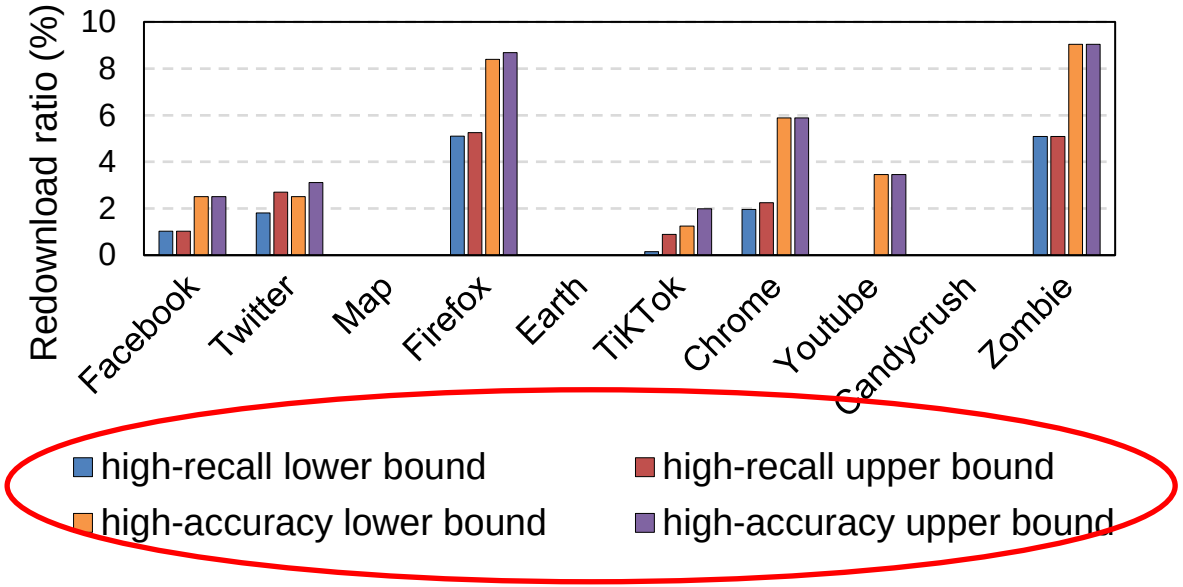
Six types of misclassifications : “BR->TR,LL”, “TR->BR,LL”, and “LL->BR,TR”.

“TR->BR” and “LL->BR,TR” could induce re-download.

1) “LL->TR” case has a small possibility to re-download (lower bound)
2) Other cases have a large possibility to re-download (upper bound)

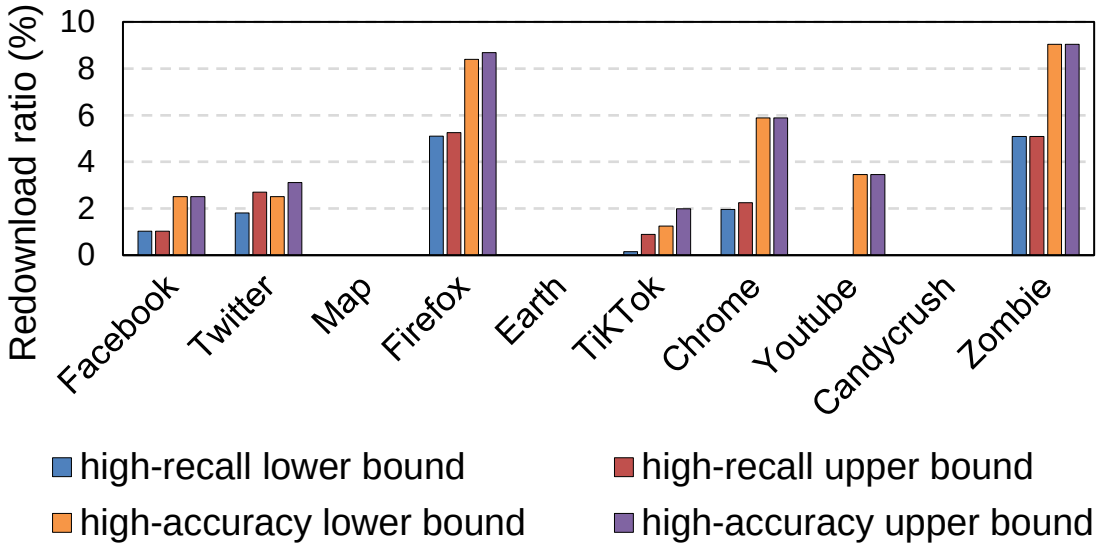
Evaluation Results: Overhead

- Overhead of CacheSifter.
  - Network overhead
    - Redownload because of misclassification



Evaluation Results: Overhead

- Overhead of CacheSifter.
  - Network overhead
    - Redownload because of misclassification



Smaller than 10%

Evaluation Results: Overhead

- Overhead of CacheSifter
  - Memory overhead
    - For categorization: 492KB
    - For maintaining Transient files: 10MB
    - For ML inference: 2MB

12.5MB DRAM

  - CPU time overhead
    - Training/retraining
      - 20h-data training per day on PC
    - Categorization
      - 82ms out of 10s on average
    - Manage cache files in memory
      - list move/insert operations
      - 1.9ms out of 10s on average

84ms out of 10s CPU time

Executive Summary

Problem: Unnecessary writes reduce flash lifetime and system performance

A large part of writes is contributed by cache files

Android systems write all cache files into flash storage

Not all cache files need to be written back for storing persistently

Our goal: To improve both system performance and lifetime of flash storage

CacheSifter: differentiate cache files and treat them according to their reuse behaviors and main-memory/storage usages

CacheSifter can reduce writes to flash storage more than 60% and thus prolong the flash lifetime more than 114% and improve write performance under intensive I/O workloads more than 18%.

Thank you!

yliang22@cityu.edu.hk

CacheSifter: Sifting Cache Files for Boosted Mobile Performance and Lifetime

Yu Liang, Riwei Pan, Tianyu Ren, Yufei Cui, Rachata Ausavarungnirun,

Xianzhang Chen, Changlong Li, Tei-Wei Kuo, and Chun Jason Xue