Acclaim: Adaptive Memory Reclaim to Improve User Experience in Android Systems

Yu Liang, Jinheng Li, Rachata Ausavarungnirun, Riwei Pan, Liang Shi, Tei-Wei Kuo, Chun Jason Xue

Executive summary

Problem: Address inefficient memory reclaim scheme in Android systems

High Page re-fault and direct reclaim

The size of reclaim scheme is too large for Android applications

Apps are still active in background and occupy memory

Improve user experience under intensive I/O requests

Key idea

Acclaim: Foreground aware and size-sensitive memory reclaim scheme

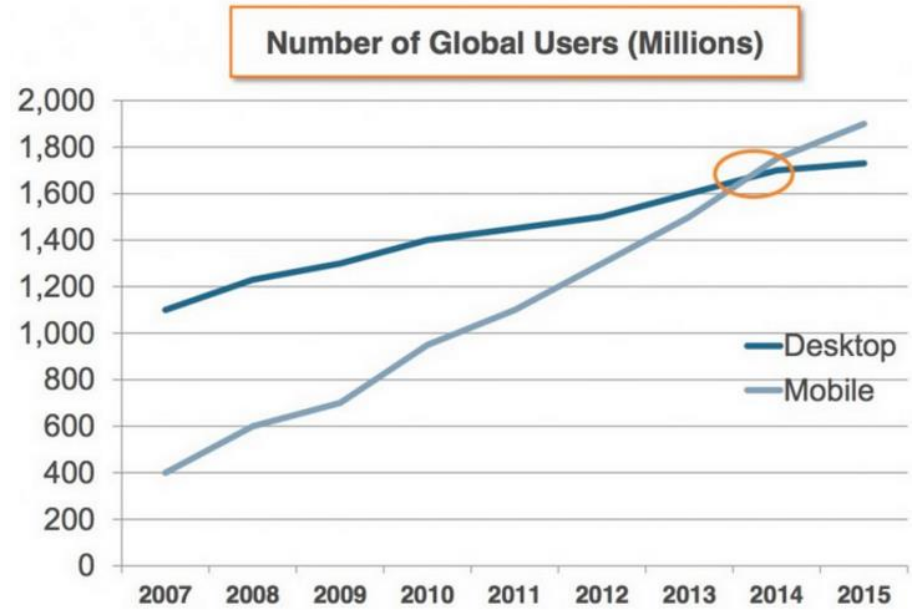
A: Lightweight prediction-based reclaim scheme (LWP)

B: Foreground aware eviction (FAE)

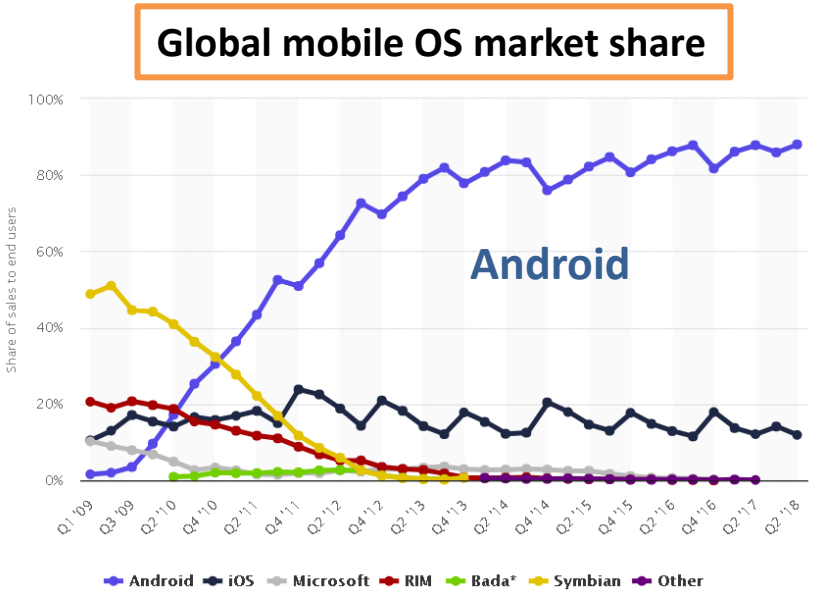
Acclaim reduces **application launch latency** up to 58.8% and improves the **write performance under intensive I/O requests** up to 49.3%.

Android mobile device is popular

Mobile devices are everywhere!



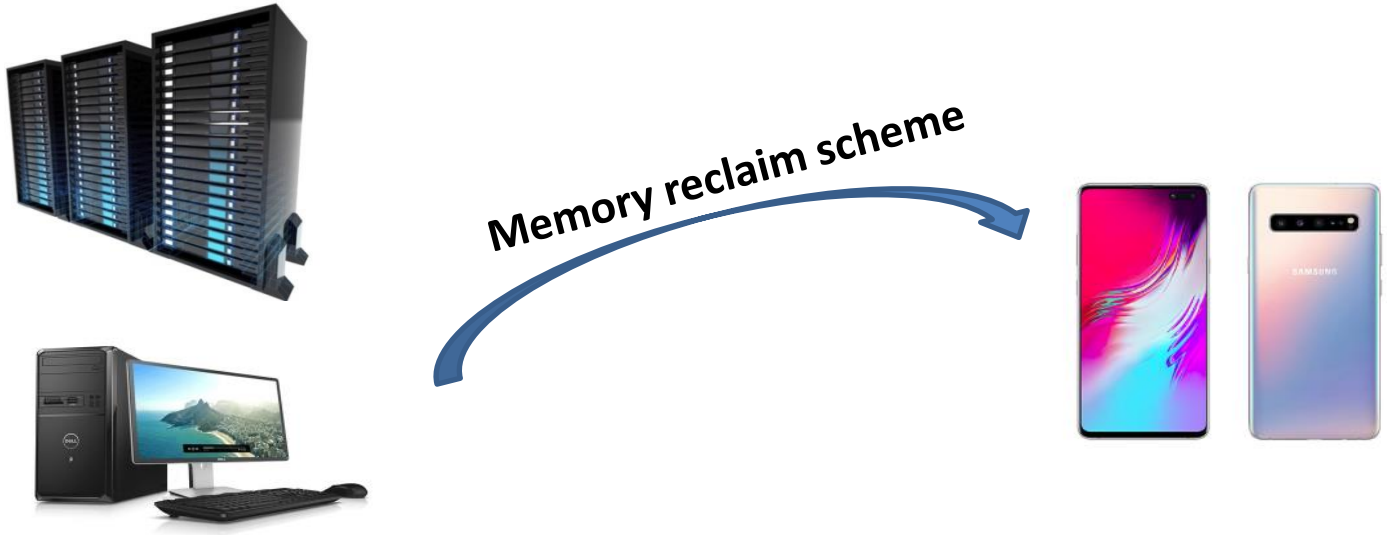
Industry report on mobile market [Morgan Stanley Research 2016]



Industry report on mobile market [© Statista 2018]

Android inherits Linux kernel

- Linux memory reclaim scheme is transplanted to Android smartphones



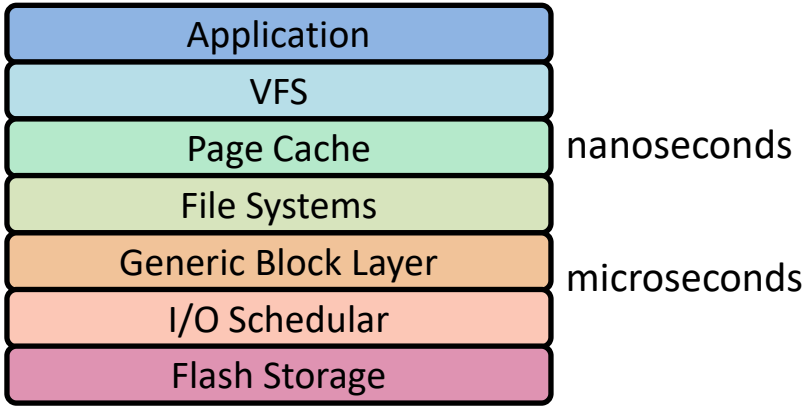
Android inherits Linux kernel

- Linux memory reclaim scheme is transplanted to Android smartphones

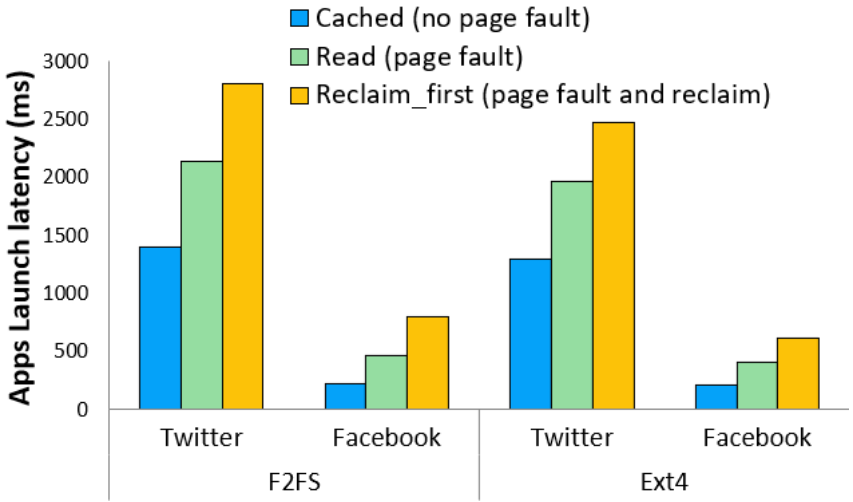
How these differences can impact the launch time of mobile applications?

Breakdown Android I/O latency

- Android I/O Latency: read procedure as an example



An overview of the Android I/O stack.

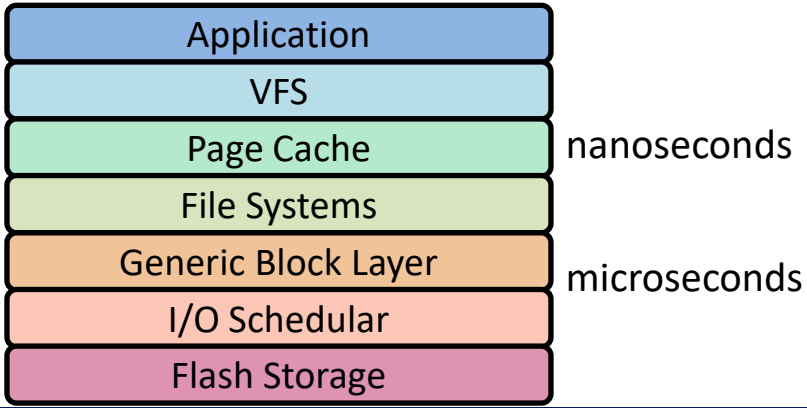


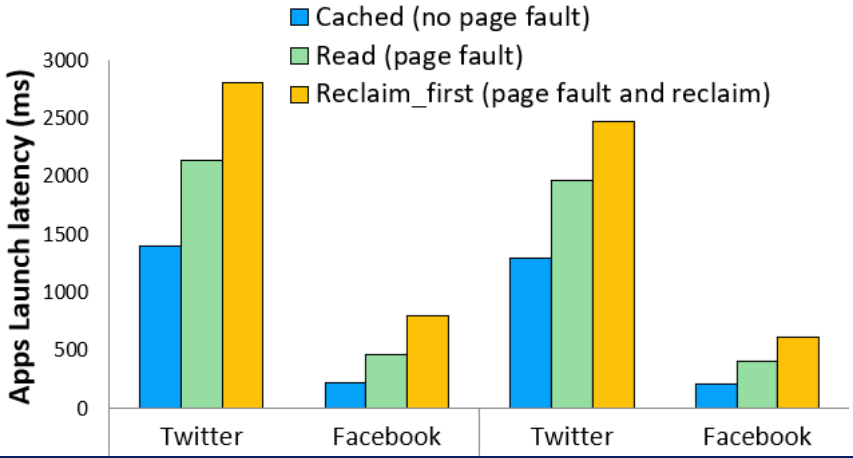
Influence of page fault and reclaim on application launch latency.

- If cache miss, page fault
- If memory is full, reclaim is conducted

Breakdown Android I/O latency

- Android I/O Latency: read procedure as an example





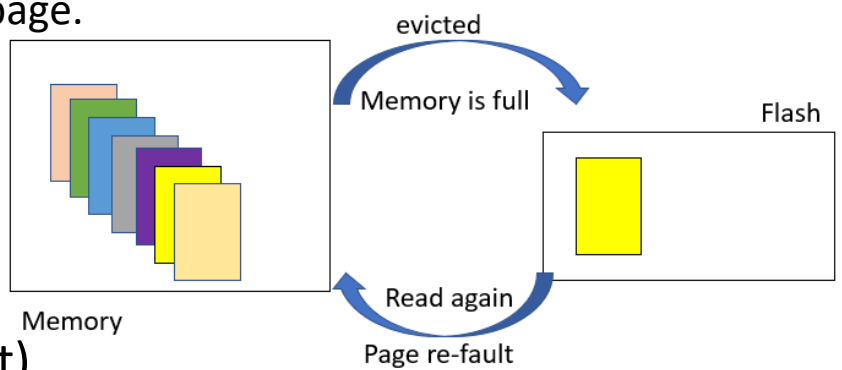
Both page fault and reclaim scheme are the key factors that impact I/O latency.

- If cache miss, page fault
- If memory is full, reclaim is conducted

Page re-fault causes page thrashing

- Page fault happens in three cases

1. Reading a page for the first time
   - Physical memory is not allocated for this page.
2. Reading a wrong address
   - This process will be killed.
3. Reading an evicted page (page re-fault)
   - It causes page thrashing.

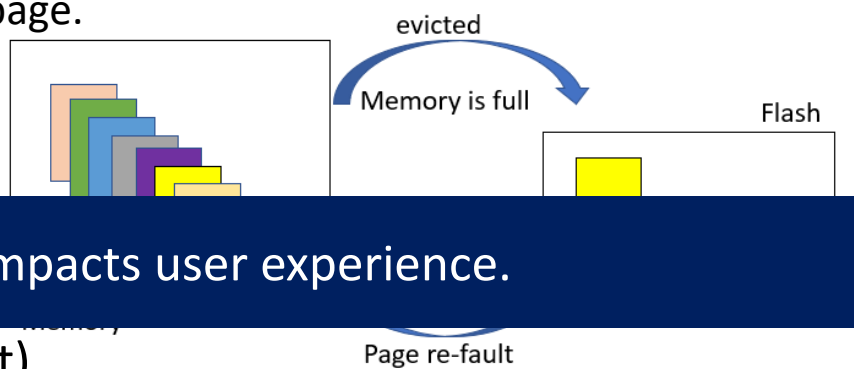


Page re-fault causes page thrashing

- Page fault happens in three cases

1. Reading a page for the first time
   - Physical memory is not allocated for this page.
2. Reading a wrong address
3. Reading an evicted page (page re-fault)
   - It causes page thrashing.

Page thrashing

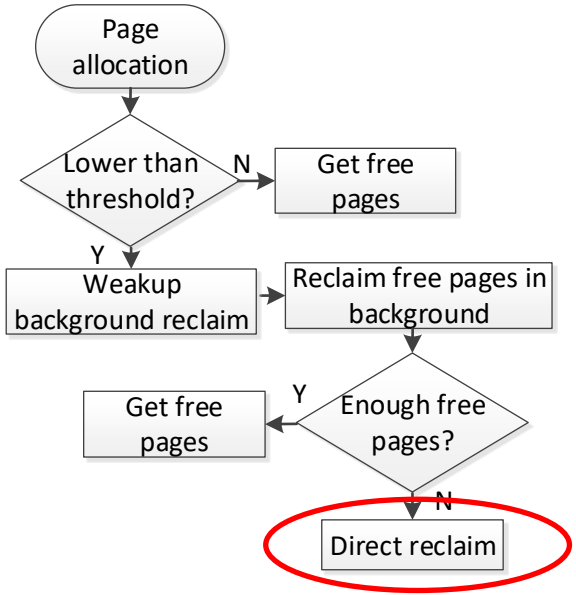


Page re-fault is a key factor that impacts user experience.

Direct reclaim can much delay page allocation

- There are mainly two reclaim schemes

1. Background reclaim (kswapd)
   - Asynchronous reclaim
   - Free pages is lower than a threshold.
2. Direct reclaim
   - Synchronous reclaim
   - Not enough free space for the system's demands.



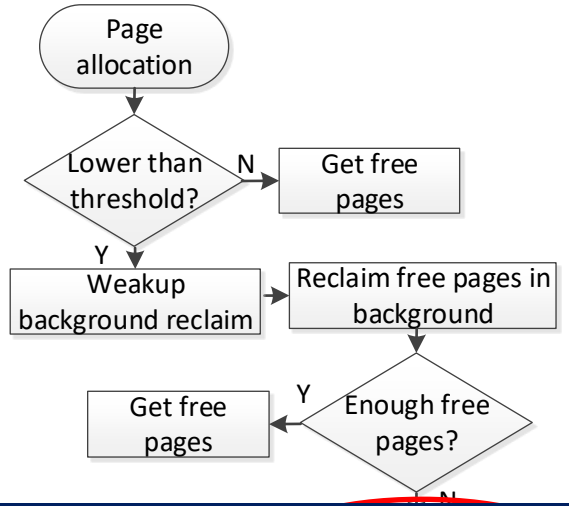
Heavy-cost

- Memory is extremely scarce
- Background apps will be killed by the Android low memory killer (LMK)

Direct reclaim can much delay page allocation

- There are mainly two reclaim schemes

1. Background reclaim (kswapd)
   - Asynchronous reclaim
   - Free pages is lower than a threshold.
2. Direct reclaim
   - Synchronous reclaim
   - Not enough free space for the system's demands.



Direct reclaim is another key factor that impacts user experience.

Heavy cost

- Memory is extremely scarce
- Background apps will be killed by the Android low memory killer (LMK)

Analysis of Android memory reclaim

- Survey of application usage patterns.

Collected data from 52 real phones.

Percentage	# of background applications
0%	N<2
15%	2<=N<5
75%	5<=N<10
10%	N>10

- Fifty-two users
- More than twenty smartphone models
- Over a two-month period

https://github.com/MIoTLab/Accliam

Analysis of Android memory reclaim

- Survey of application usage patterns.

Collected data from 52 real phones.

Percentage	# of background applications
0%	N<2
15%	2<=N<5
75%	5<=N<10

We design evaluation scenarios according to the survey.

- Fifty-two users
- More than twenty smartphone models
- Over a two-month period

https://github.com/MIoTLab/Accliam

Analysis of Android memory reclaim

- Experimental workloads

Application combinations used in experiments.

Applications	Operations	Memory	Workloads
A	Launch and use an application for 5 minutes	Avail.	Light
3B+A	3 background applications	Avail.	Moderate
8B+A	8 background applications	Full	Moderate
15B+A	15 background applications	Full	Heavy

  - Twenty apps (i.e. social media, browser, map, game, news, and multimedia applications).

- Experimental setup

Huawei P9 smartphone with 3GB and 2.5GB RAM, running Android 7.0 on Linux kernel version 4.1.18.

Analysis of Android memory reclaim

- Experimental workloads

Application combinations used in experiments.

Applications	Operations	Memory	Workloads
A	Launch and use an application for 5 minutes	Avail.	Light
3B+A	3 background applications	Avail.	Moderate
8B+A	8 background applications	Full	Moderate

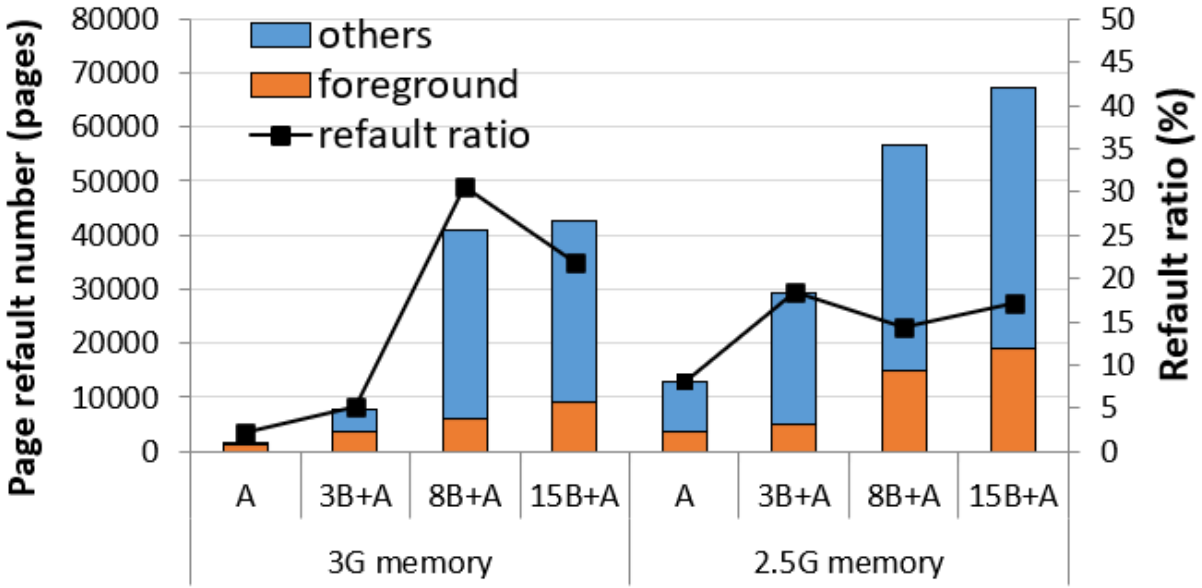
Base on these evaluation scenarios we analyze page re-fault and direct reclaim.

- Experimental setup

Huawei P9 smartphone with 3GB and 2.5GB RAM, running Android 7.0 on Linux kernel version 4.1.18.

Analysis of Android memory reclaim

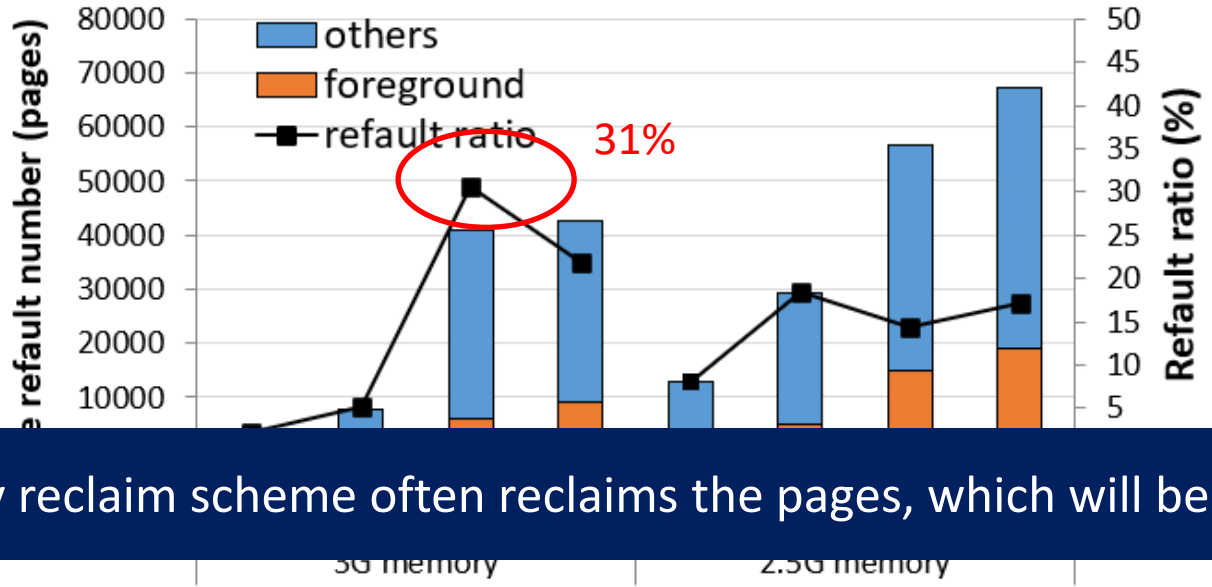
- Re-fault and direct reclaim on mobile devices



Ratio and number of page re-faults

Analysis of Android memory reclaim

- Re-fault and direct reclaim on mobile devices

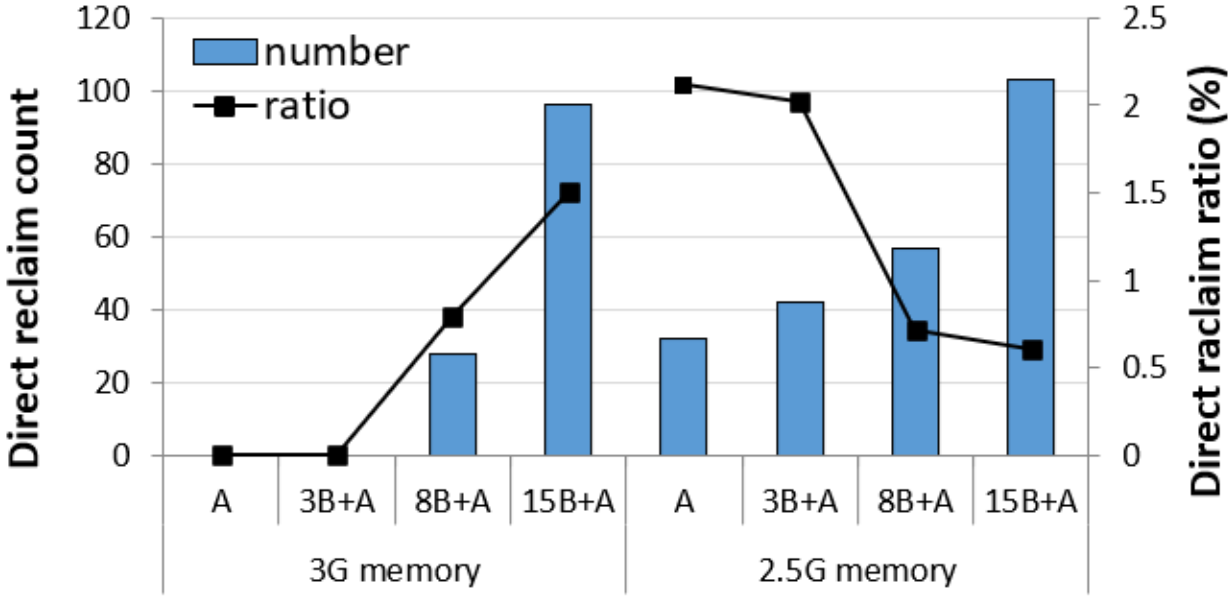


Memory reclaim scheme often reclaims the pages, which will be used soon.

Ratio and number of page re-faults

Analysis of Android memory reclaim

- Re-fault and direct reclaim on mobile devices

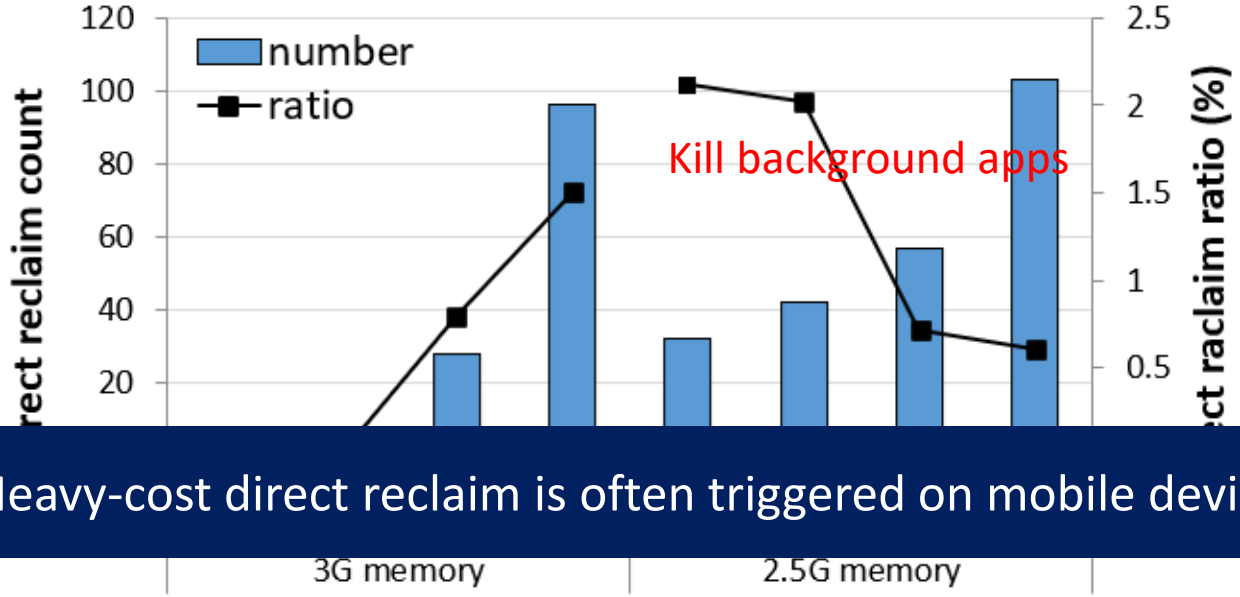


Ratio and number of direct reclaims

- Servers also have the problems.

Analysis of Android memory reclaim

- Re-fault and direct reclaim on mobile devices



Heavy-cost direct reclaim is often triggered on mobile devices.

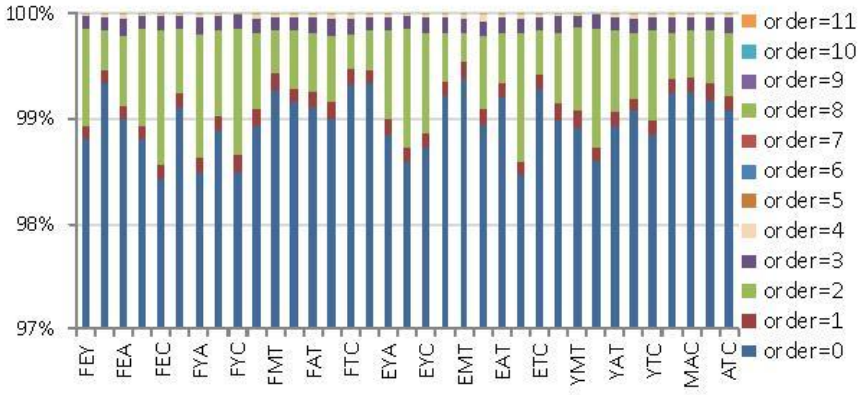
Ratio and number of direct reclaims

- Servers also have the problems.

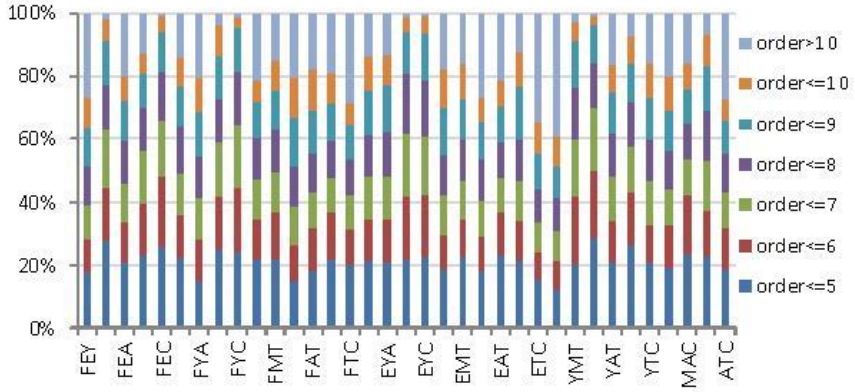
Allocation sizes of mobile devices are usually small

- Compared to allocation size, reclaim size is often too large.

The distribution of allocation sizes:



The distribution of reclaim sizes :

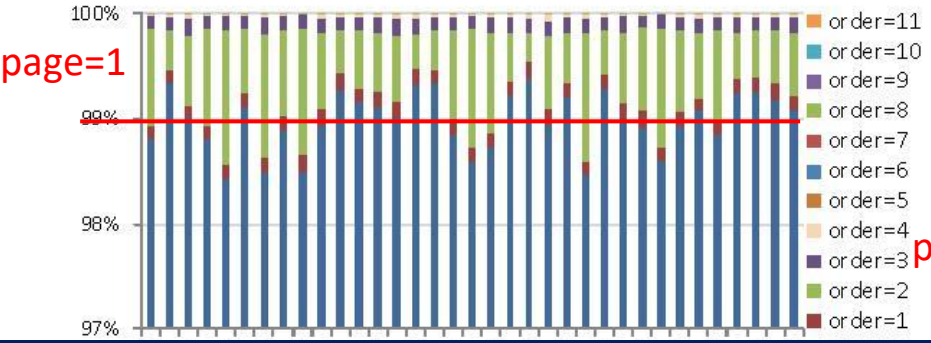


- The size of a block of order n is 2^n.
- 99% of allocation sizes are 1 page (order=0)
- 80% of reclaiming sizes are larger than 32 pages (order=5).

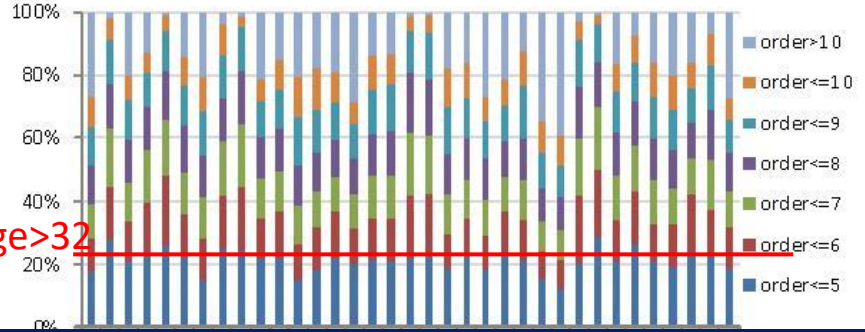
Allocation sizes of mobile devices are usually small

- Compared to allocation size, reclaim size is often too large.

The distribution of allocation sizes:



The distribution of reclaim sizes :

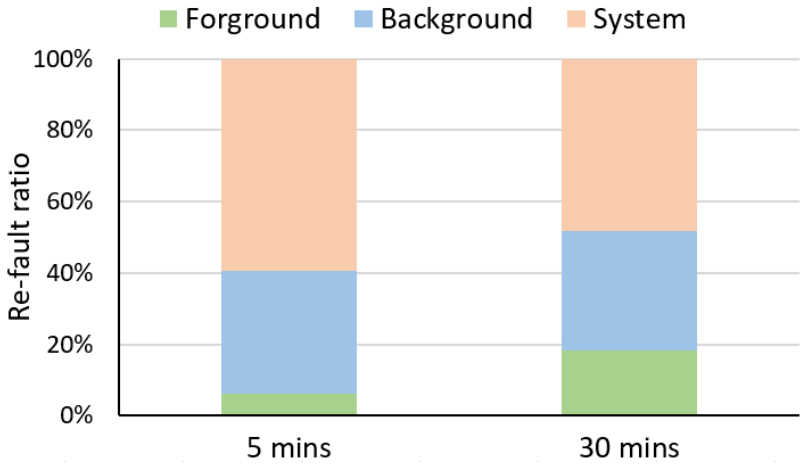


The reclaim size of background reclaim is often too large.

- The size of a block of order n is 2^n.
- 99% of allocation sizes are 1 page (order=0)
- 80% of reclaiming sizes are larger than 32 pages (order=5).

Foreground apps more important to user experience

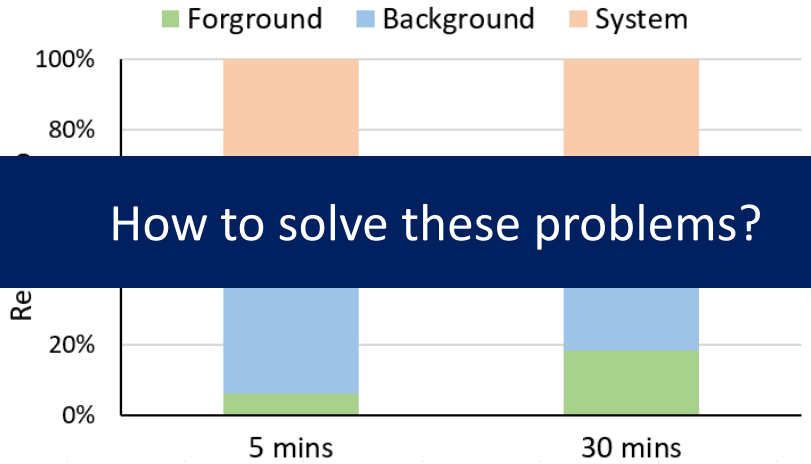
- Background applications keep consuming free pages.
  - Anonymous pages from background apps thrash file pages of foreground apps.
    - Anonymous pages are more important to a process than file pages.
  - Background applications are still active.



Re-fault pages produced by each part.

Foreground apps more important to user experience

- Background applications keep consuming free pages.
  - Anonymous pages from background apps thrash file pages of foreground apps.
    - Anonymous pages are more important to a process than file pages.
  - Background applications are still active.



Re-fault pages produced by each part.

How to solve these problems?

Our Solution: Acclaim

- Design goals:
  - The reclaim size of background reclaim is just right.
  - Background applications have lower priority

- Acclaim, foreground aware and size-sensitive reclaim scheme.

  - Lightweight prediction-based reclaim scheme (LWP)
    - Target: Reclaim size of the background reclaim is too large.
    - Solution: Tunes size and amount of background reclaims according to the predicted allocation workloads.

  - Foreground aware eviction (FAE)
    - Target: Background applications keep consuming free pages.
    - Solution: Gets space from background apps to the foreground app.

Our Solution: Acclaim

- Design goals:
  - The reclaim size of background reclaim is just right.
  - Background applications have lower priority

- Acclaim, foreground aware and size-sensitive reclaim scheme.

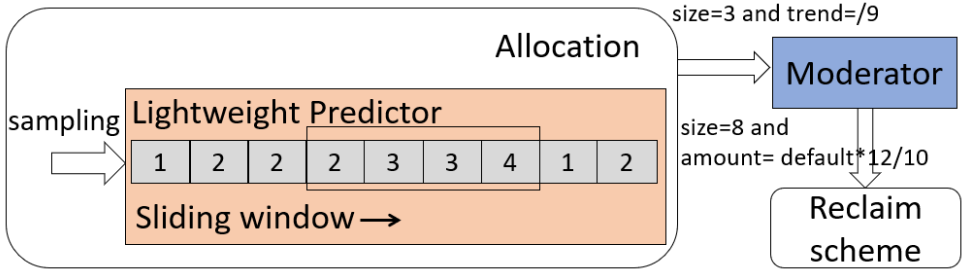
  - Lightweight prediction-based reclaim scheme (LWP)
    - Target: Reclaim size of the background reclaim is too large.
    - Solution: Tunes size and amount of background reclaims according to the predicted allocation workloads.

Two schemes targets two problems.

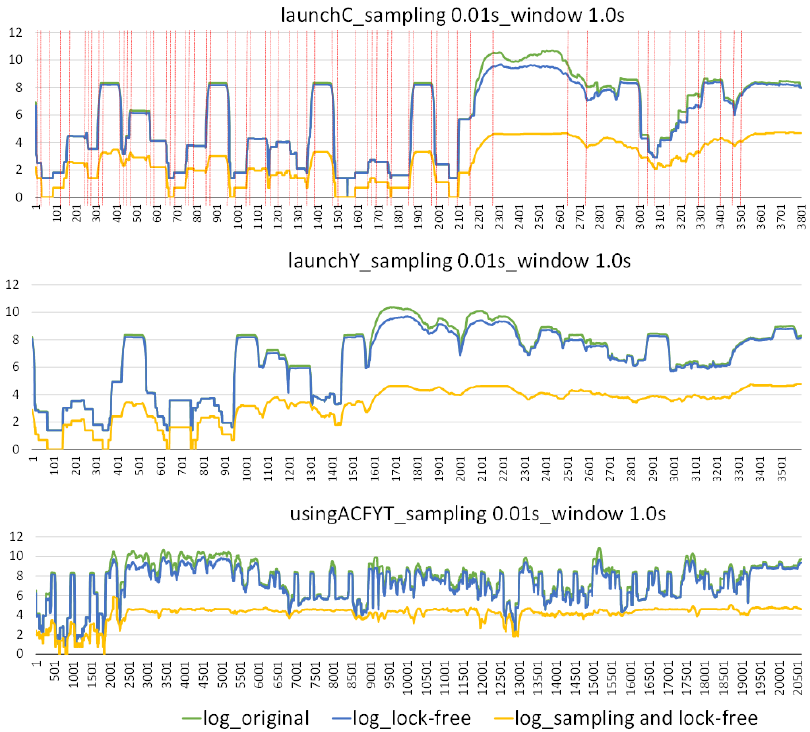
    - Target: Background applications keep consuming free pages.
    - Solution: Gets space from background apps to the foreground app.

Lightweight prediction-based reclaim scheme (LWP)

- LWP predicts allocation workloads



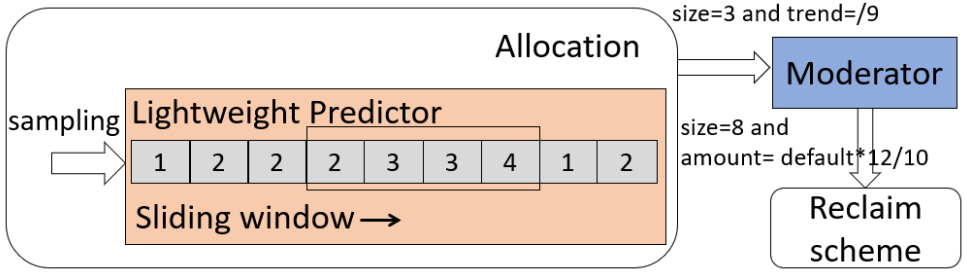
An example of LWP.



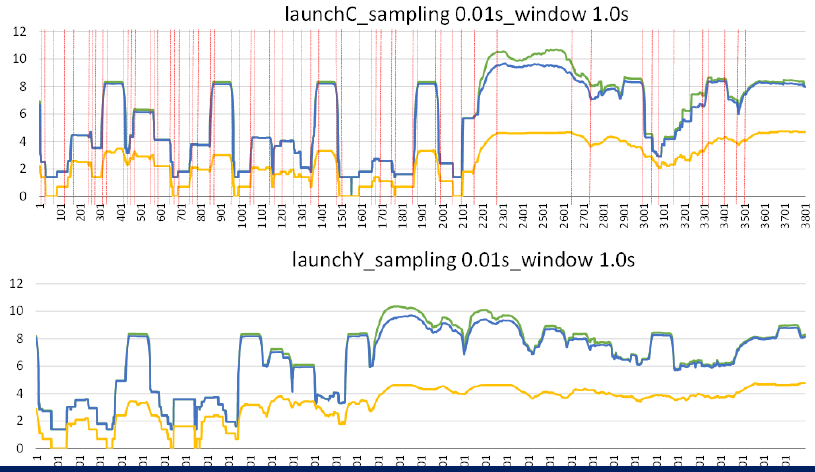
- *Predicted size* =average of sizes in window
- *Predicted amount trend*=sum/last sum

Lightweight prediction-based reclaim scheme (LWP)

- LWP predicts allocation workloads

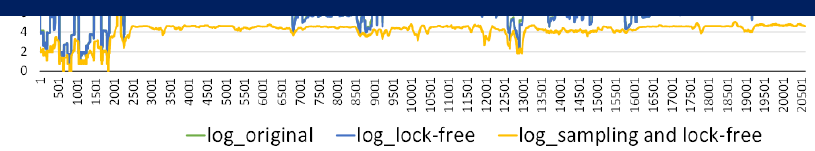


An example of LWP.



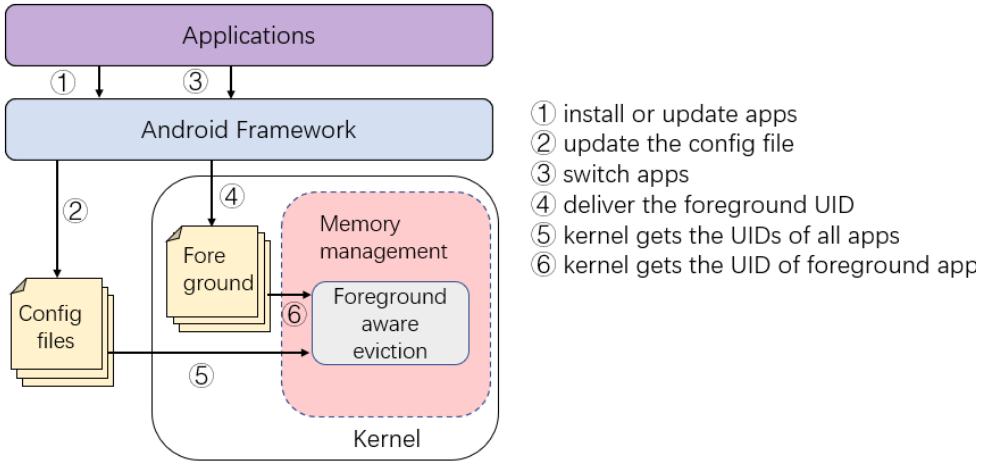
Tune size and amount of background reclaims according to the predicted allocation workloads.

- *Predicted size* =average of sizes in window
- *Predicted amount trend*=sum/last sum



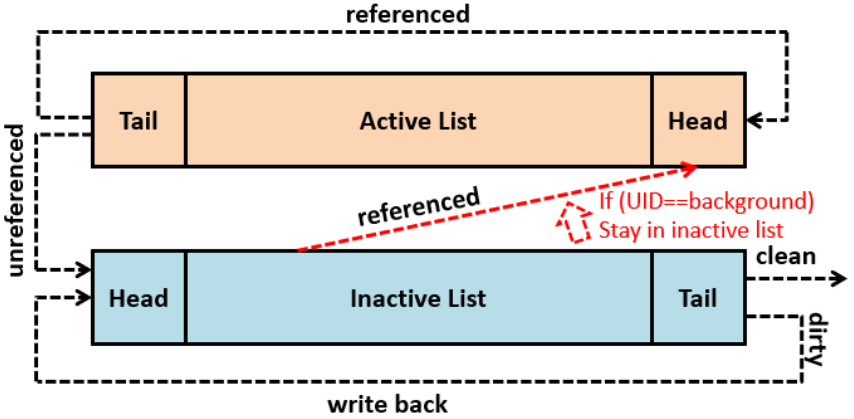
Foreground aware eviction (FAE)

- FAE lowers the priority of pages of background applications.



① install or update apps
② update the config file
③ switch apps
④ deliver the foreground UID
⑤ kernel gets the UIDs of all apps
⑥ kernel gets the UID of foreground app

Framework of foreground aware evict scheme.

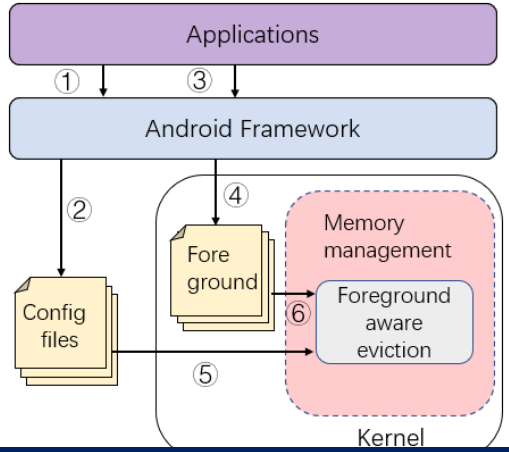


Foreground aware evict scheme of LRU lists.

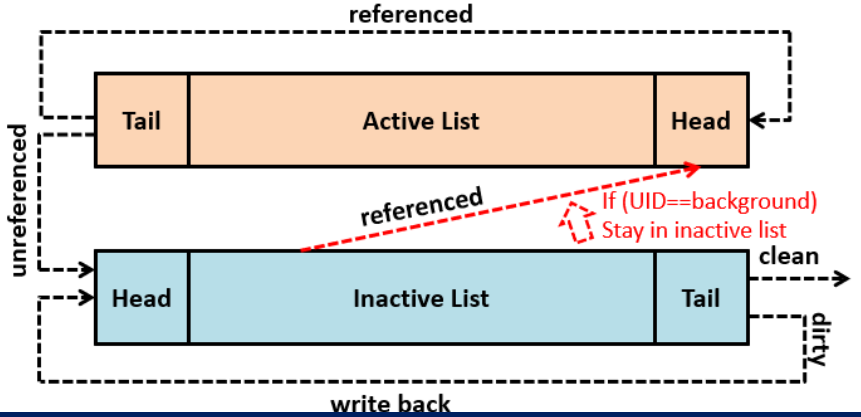
- Music or video players can be excluded in background apps.

Foreground aware eviction (FAE)

➢ FAE lowers the priority of pages of background applications.



① install or update apps
② update the config file
③ switch apps
④ deliver the foreground UID
⑤ kernel gets the UIDs of all apps
⑥ kernel gets the UID of foreground app

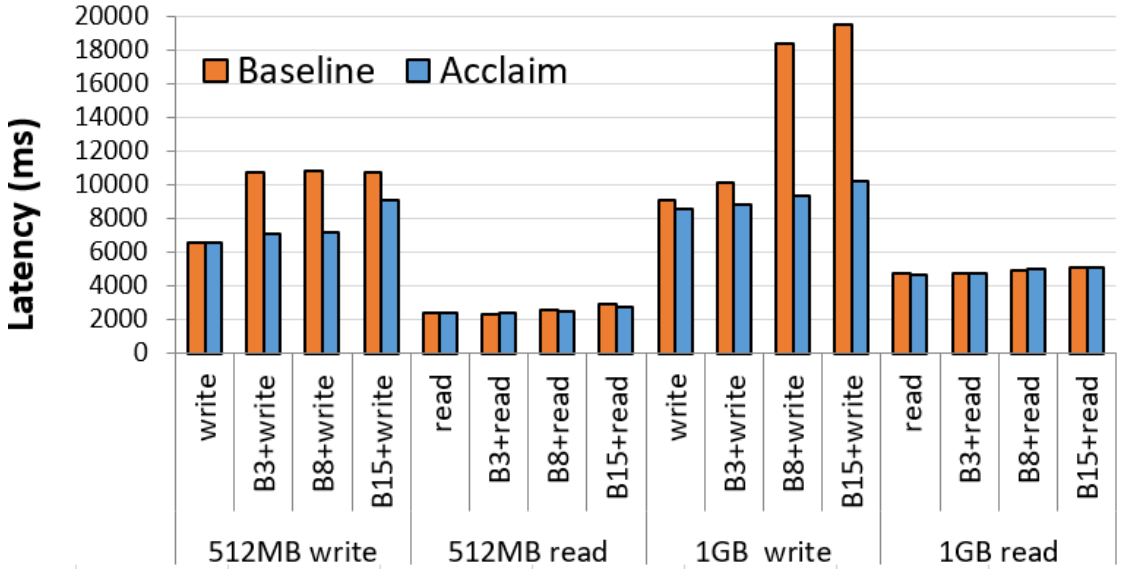


Gets space from background applications to the foreground application.

- Music or video players can be excluded in background apps.

Evaluation results: read/write performance

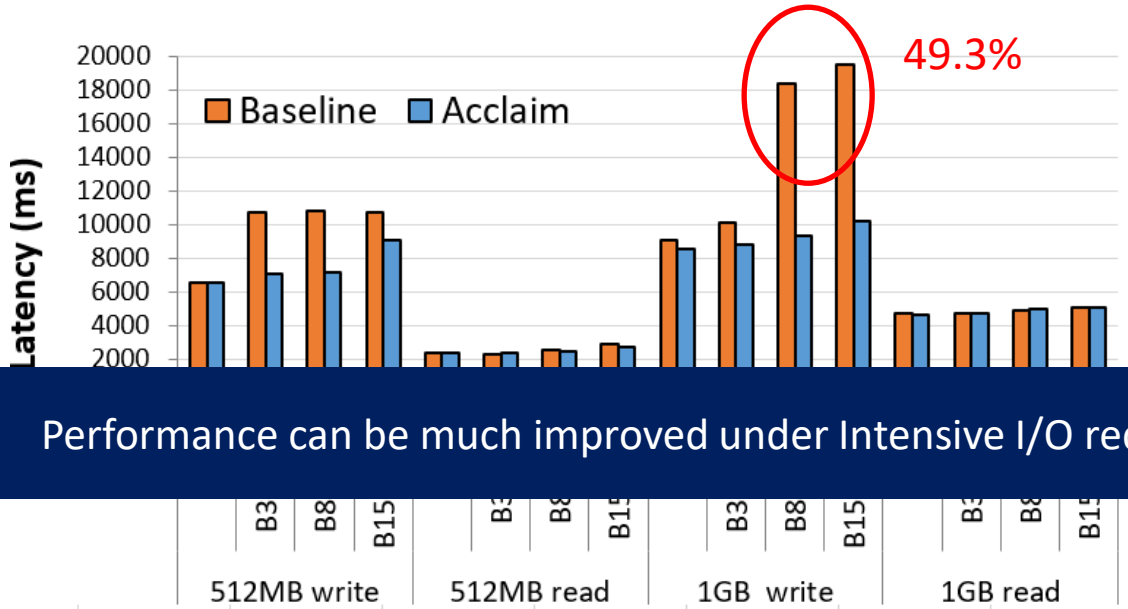
- Benefit to read/write performance.
  - We write and read 512MB and 1GB of data in size of 4KB.



Read and write performance.

Evaluation results: read/write performance

- Benefit to read/write performance.
  - We write and read 512MB and 1GB of data in size of 4KB.

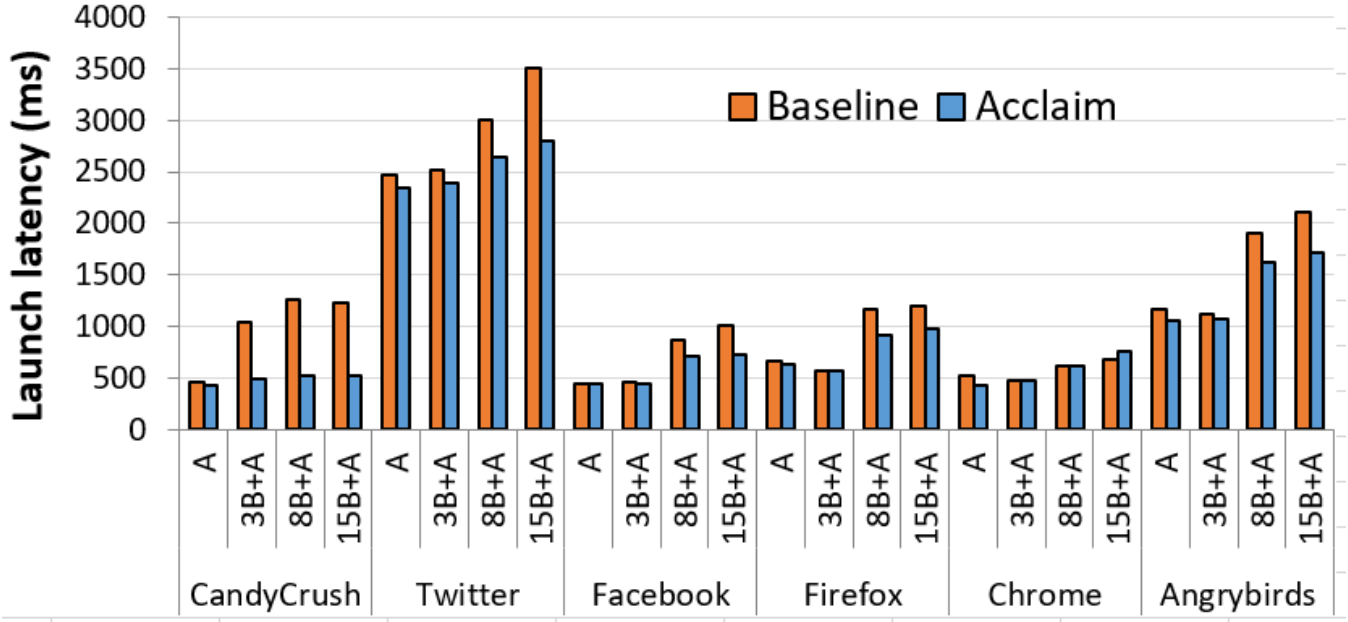


Performance can be much improved under Intensive I/O requests.

Read and write performance.

Evaluation results: app launching

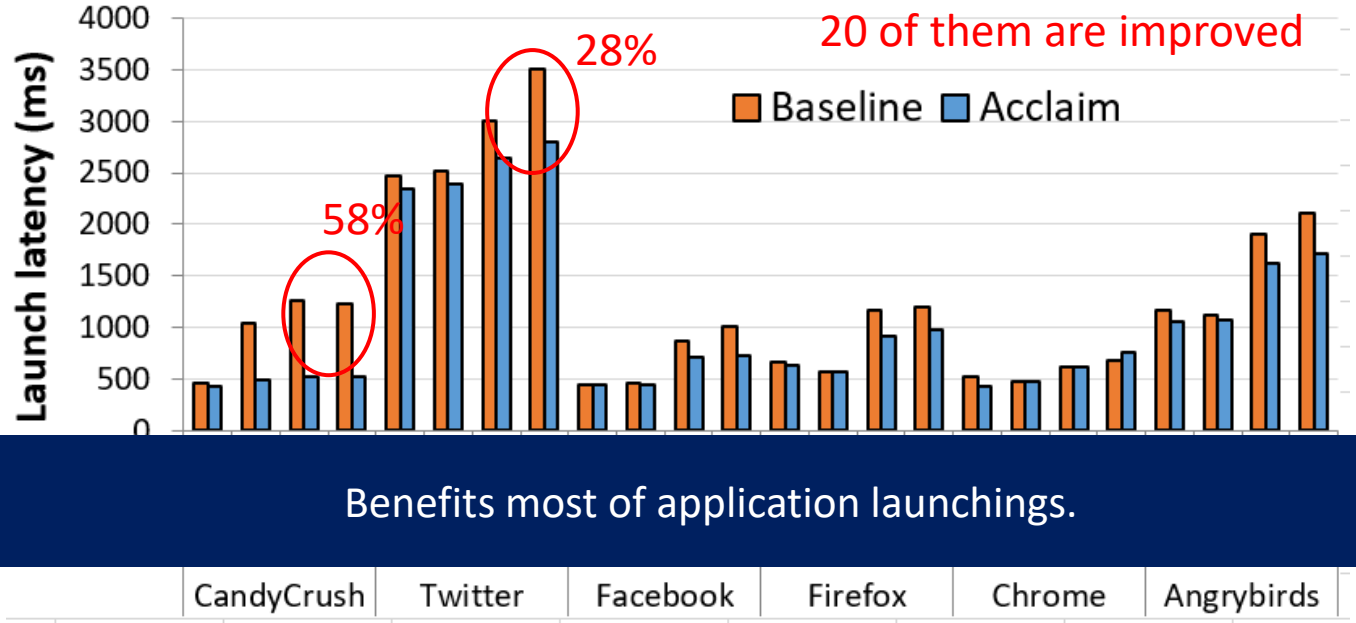
- Reduce launch time in 20 out of 24 benchmarks.



Application launch performance.

Evaluation results: app launching

- Reduce launch time in 20 out of 24 benchmarks.



Benefits most of application launchings.

Application launch performance.

Evaluation results: overhead

- Overhead of Acclaim.
  - Additional memory overhead.
    - Each page 4 Bytes for UID
      - 0.1% of memory capacity
    - 100 values in sliding window will take up 400B in LWP
  - Performance overhead.
    - FAE needs few comparisons.
      - Check the configure file to get the UIDs. (Once)
      - Foreground UID deliver.
      - Check background UID during each page eviction.
    - Two parts for LWP
      - Lock-free sliding window.
      - Prolong the wake-up time of the background reclaim.

Evaluation results: overhead

- Overhead of Acclaim.
  - Additional memory overhead.
    - Each page 4 Bytes for UID
      - 0.1% of memory capacity
    - 100 values in sliding window will take up 400B in LWP
  - Performance overhead.

Both memory overhead and performance overhead are trivial.

      - Check background UID during each page eviction.
    - Two parts for LWP
      - Lock-free sliding window.
      - Prolong the wake-up time of the background reclaim.

Conclusion

Problem: Address inefficient memory reclaim scheme in Android systems

High Page re-fault and direct reclaim

Improve user experience under intensive I/O requests

Key idea

Acclaim: Foreground aware and size-sensitive memory reclaim scheme

A: Lightweight prediction-based reclaim scheme (LWP)

B: Foreground aware eviction (FAE)

Acclaim reduces **application launch latency** up to 58.8% and improves the **write performance under intensive I/O requests** up to 49.3%.

Thank you!

yliang22-c@my.cityu.edu.hk

Acclaim: Adaptive Memory Reclaim to Improve User Experience in Android Systems

Yu Liang, Jinheng Li, Rachata Ausavarungnirun, Riwei Pan, Liang Shi, Tei-Wei Kuo, Chun Jason Xue