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

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

An overview of the Android I/O stack.

Influence of page fault and reclaim on application launch latency.
Breakdown Android I/O latency

Android I/O Latency: read procedure as an example

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

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

- 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.

- 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.

<table>
<thead>
<tr>
<th>Percentage</th>
<th># of background applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>N&lt;2</td>
</tr>
<tr>
<td>15%</td>
<td>2&lt;=N&lt;5</td>
</tr>
<tr>
<td>75%</td>
<td>5&lt;=N&lt;10</td>
</tr>
<tr>
<td>10%</td>
<td>N&gt;10</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th>Percentage</th>
<th># of background applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>N&lt;2</td>
</tr>
<tr>
<td>15%</td>
<td>2&lt;=N&lt;5</td>
</tr>
<tr>
<td>75%</td>
<td>5&lt;=N&lt;10</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Operations</th>
<th>Memory</th>
<th>Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Launch and use an application for 5 minutes</td>
<td>Avail.</td>
<td>Light</td>
</tr>
<tr>
<td>3B+A</td>
<td>3 background applications</td>
<td>Avail.</td>
<td>Moderate</td>
</tr>
<tr>
<td>8B+A</td>
<td>8 background applications</td>
<td>Full</td>
<td>Moderate</td>
</tr>
<tr>
<td>15B+A</td>
<td>15 background applications</td>
<td>Full</td>
<td>Heavy</td>
</tr>
</tbody>
</table>

• 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.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Operations</th>
<th>Memory</th>
<th>Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Launch and use an application for 5 minutes</td>
<td>Avail.</td>
<td>Light</td>
</tr>
<tr>
<td>3B+A</td>
<td>3 background applications</td>
<td>Avail.</td>
<td>Moderate</td>
</tr>
<tr>
<td>8B+A</td>
<td>8 background applications</td>
<td>Full</td>
<td>Moderate</td>
</tr>
<tr>
<td>15B+A</td>
<td></td>
<td></td>
<td>Heavy</td>
</tr>
</tbody>
</table>

Based 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

![Graph showing page re-fault number and re-fault ratio](image-url)

- **Page re-fault number (pages)**
- **Refault ratio (%)**

- **others**
- **foreground**
- **refault ratio**

<table>
<thead>
<tr>
<th>Memory</th>
<th>A</th>
<th>3B+A</th>
<th>8B+A</th>
<th>15B+A</th>
<th>A</th>
<th>3B+A</th>
<th>8B+A</th>
<th>15B+A</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5G memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

- 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:

- 99% of allocation sizes are 1 page (order=0).

The distribution of reclaim sizes:

- 80% of reclaiming sizes are larger than 32 pages (order=5).
- The size of a block of order $n$ is $2^n$. 
Allocation sizes of mobile devices are usually small

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

The distribution of allocation sizes:

99% of allocation sizes are 1 page (order=0)

The distribution of reclaim sizes:

80% of reclaiming sizes are larger than 32 pages (order=5).

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.
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.

How to solve these problems?

Re-fault pages produced by each part.
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

![Diagram of LWP system]

**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.

Framework of foreground aware evict scheme.

- Music or video players can be excluded in background apps.
Foreground aware eviction (FAE)

- FAE lowers the priority of pages of background applications.

Gets space from background applications to the foreground application.

- Music or video players can be excluded in background apps.
Benefit to read/write performance.

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

![Chart showing application launch performance comparison between Baseline and Acclaim](chart.png)
Evaluation results: app launching

- Reduce launch time in 20 out of 24 benchmarks.

Benefits most of application launchings.

<table>
<thead>
<tr>
<th>CandyCrush</th>
<th>Twitter</th>
<th>Facebook</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Angrybirds</th>
</tr>
</thead>
</table>

Application launch performance.
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.
    – Check the configure file to get the UIDs. (Once)
    – Foreground UID deliver.
    – Check background UID during each page eviction.

Both memory overhead and performance overhead are trivial.

• Two parts for LWP
  – Lock-free sliding window.
  – Prolong the wake-up time of the background reclaim.
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