Integrated Host-SSD Mapping Table Management for Improving User Experience of Smartphones

Yoona Kim, Inhyuk Choi, Juhyung Park†, Jaeheon Lee†, Sungjin Lee†, Jihong Kim



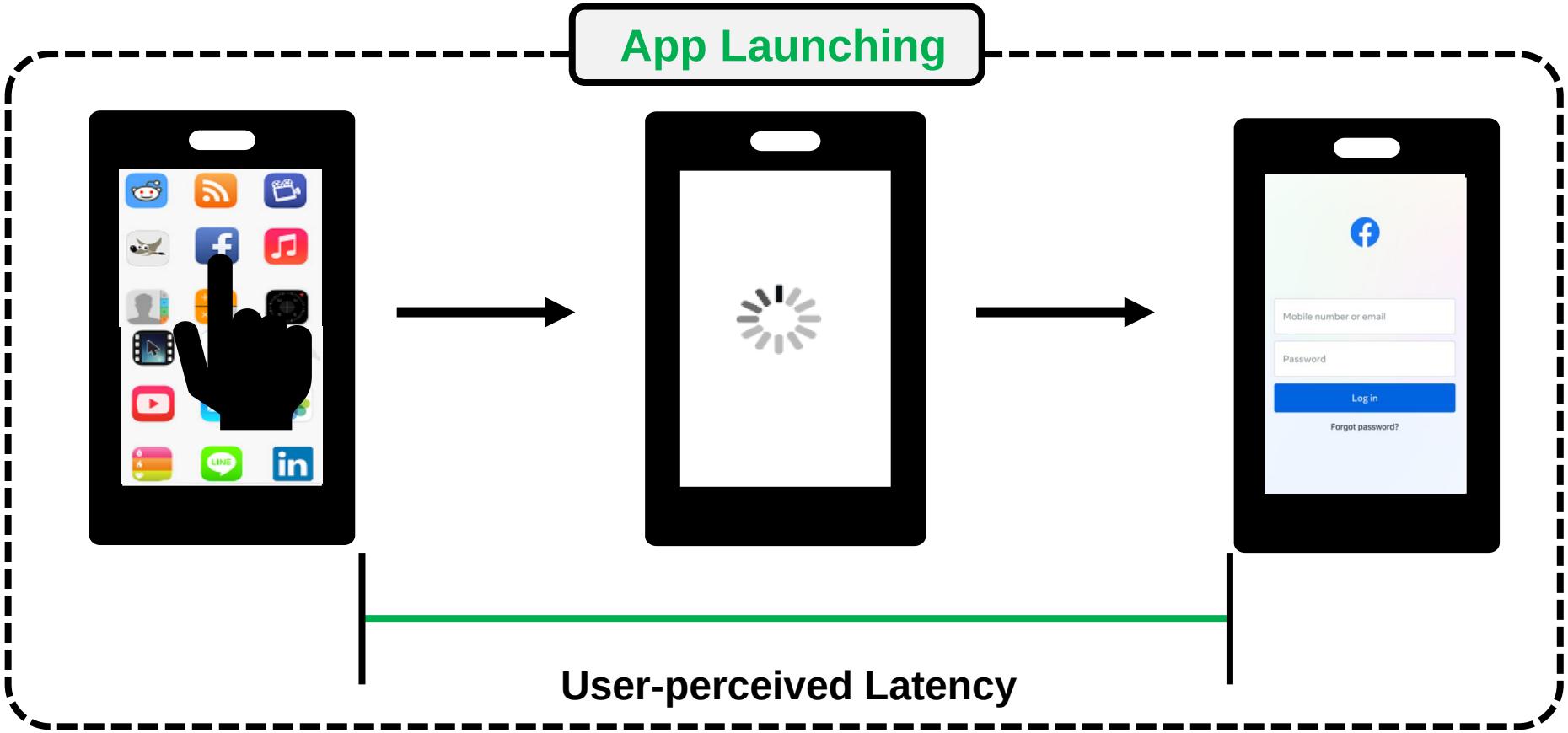
Seoul National University, † DGIST



The 21st USENIX Conference on File and Storage Technologies (FAST), 2023

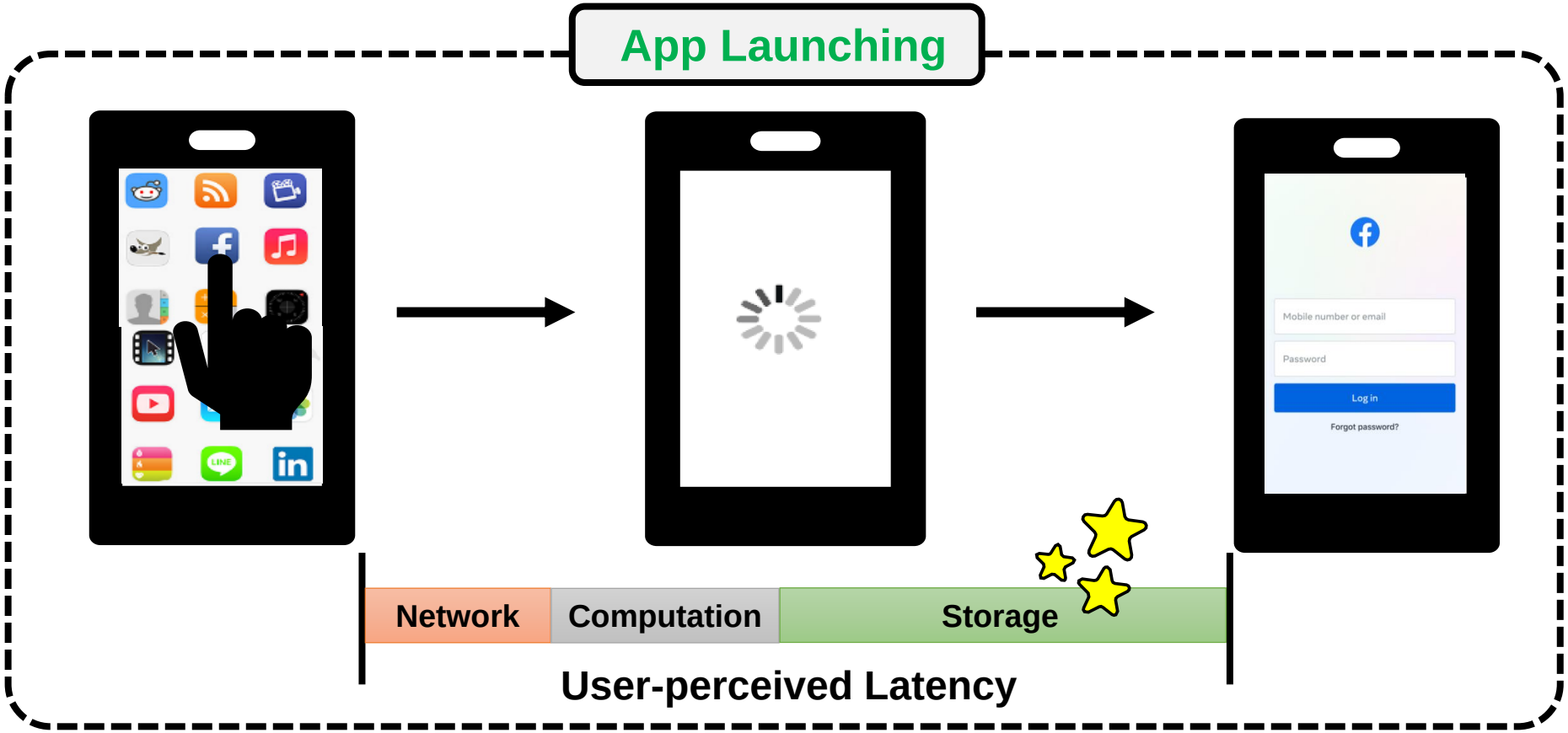
User Experience (UX) in Smartphones

- **User-perceived latency** (system responsiveness) determines *UX quality*.



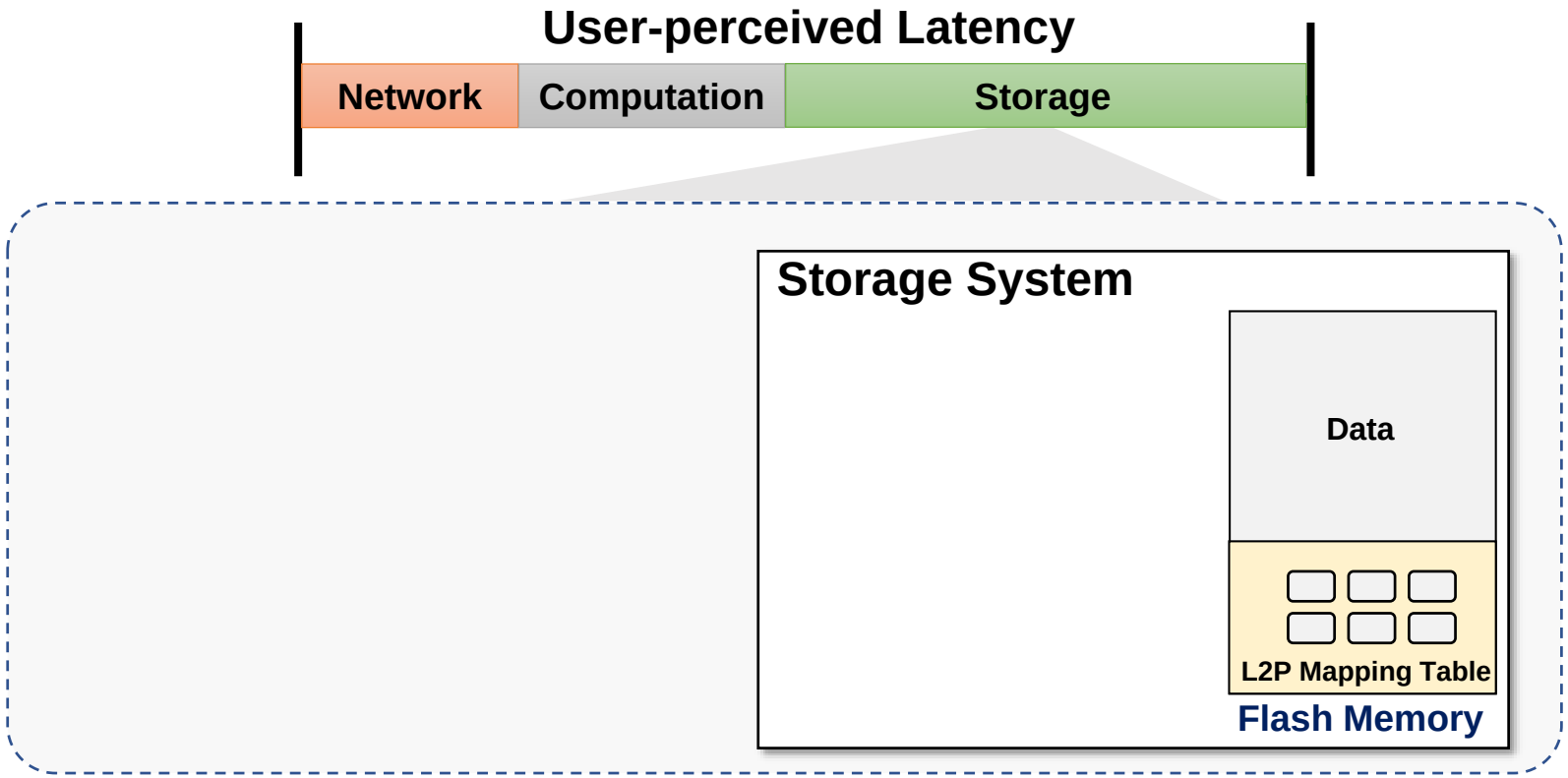
User Experience (UX) in Smartphones

- **User-perceived latency** (system responsiveness) determines *UX quality*.
- A long **storage latency** can significantly increase user-perceived latency during app *launching*, *switching*, and *loading* times.



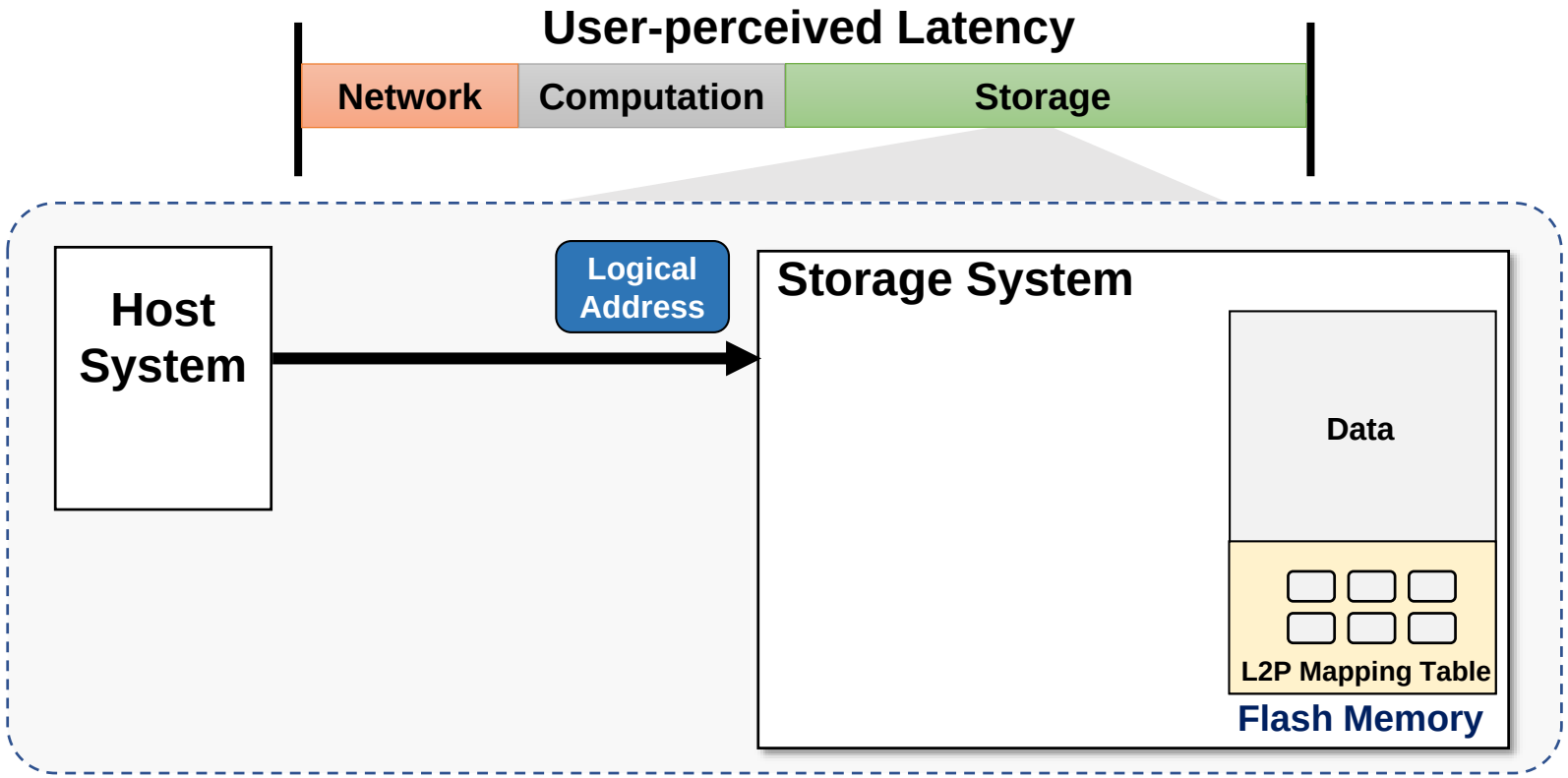
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



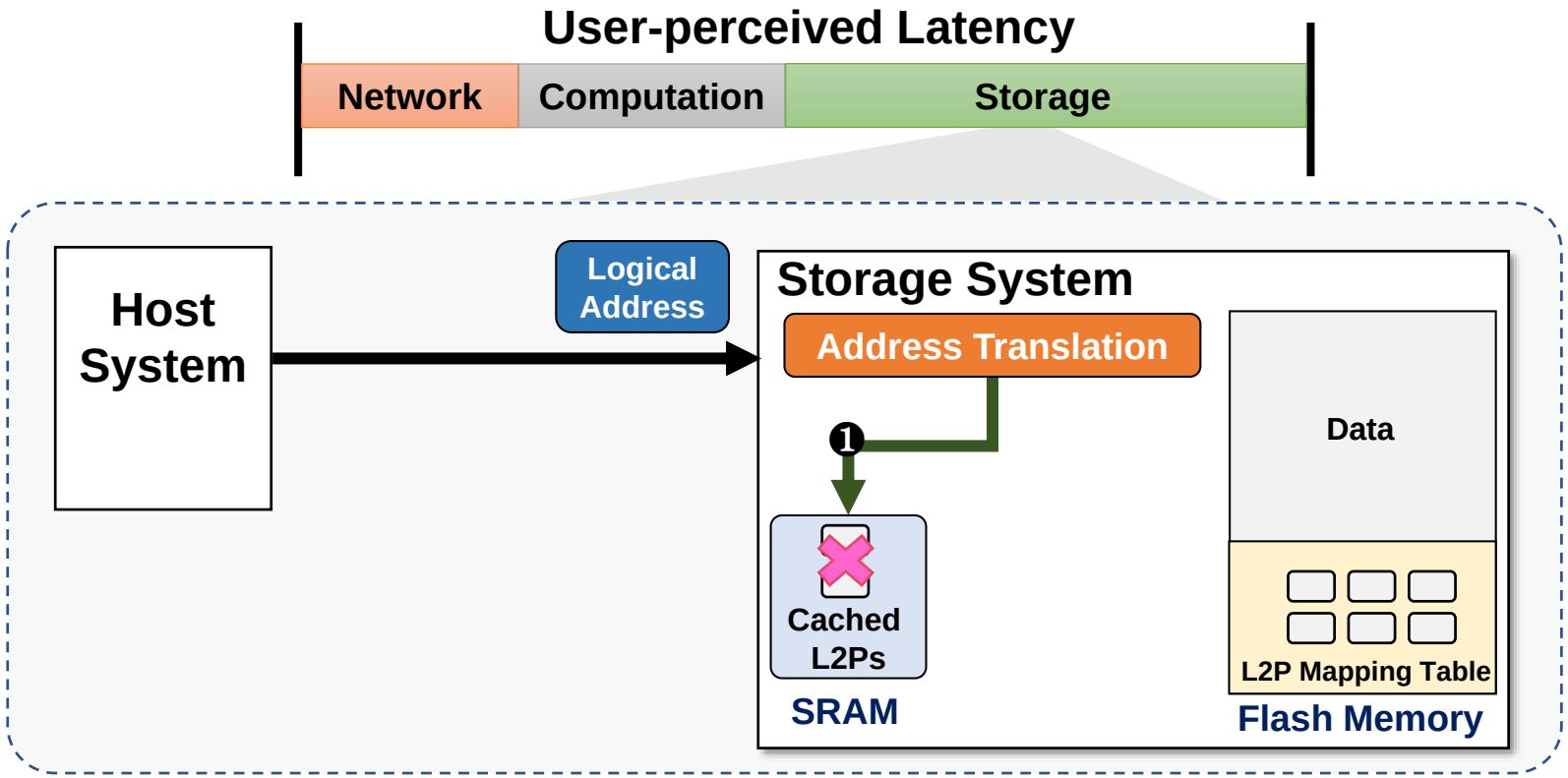
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



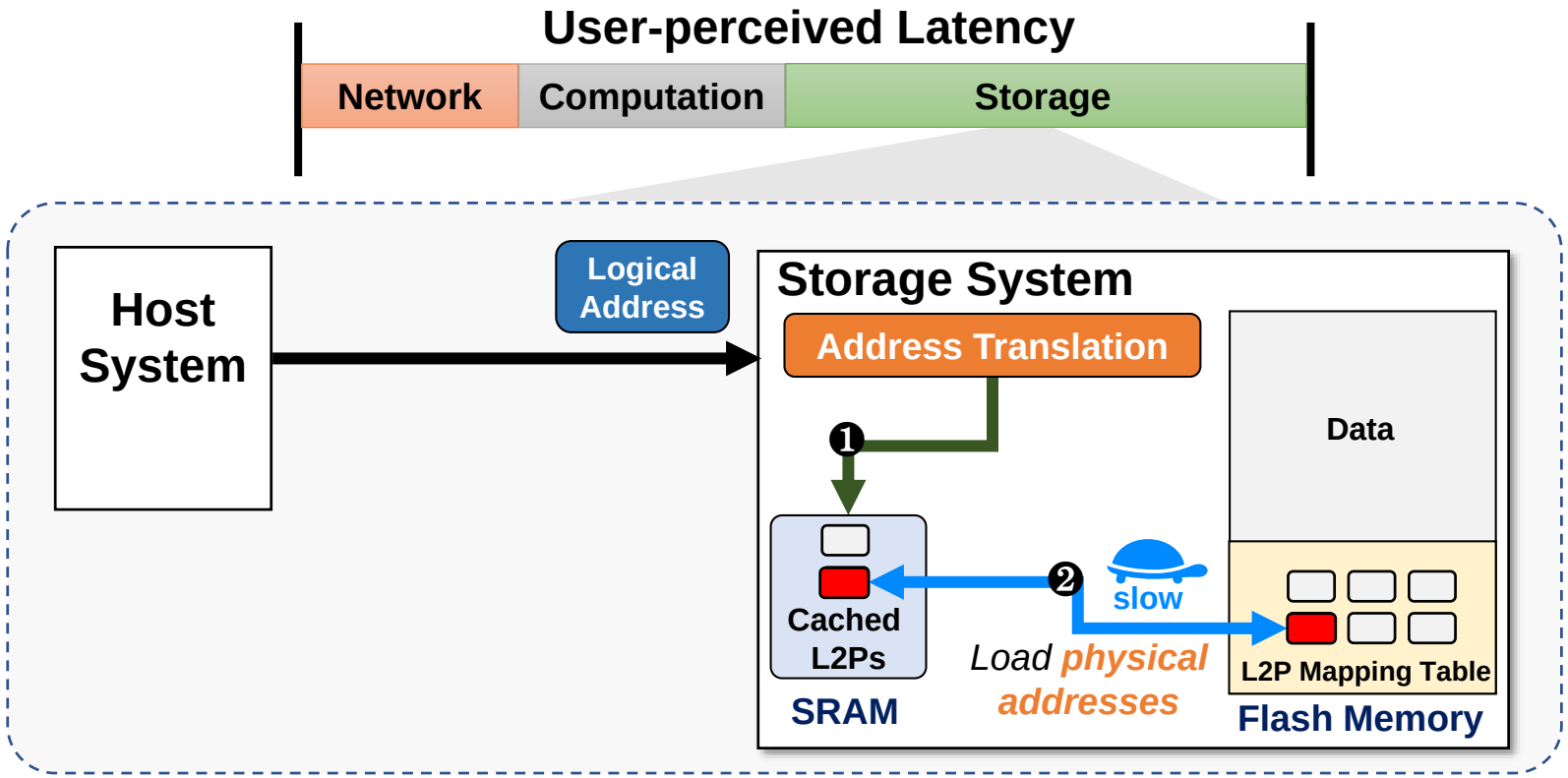
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



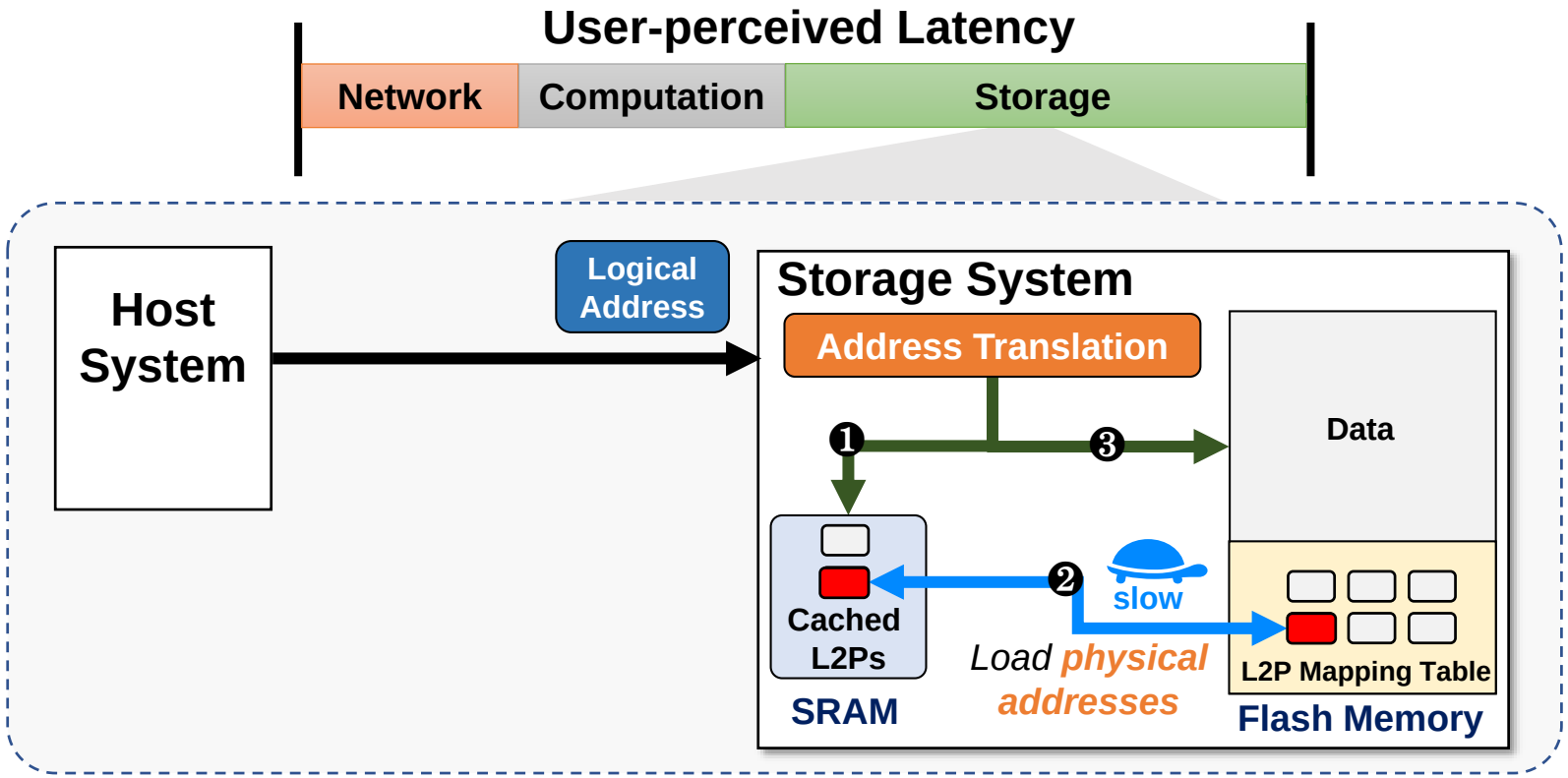
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



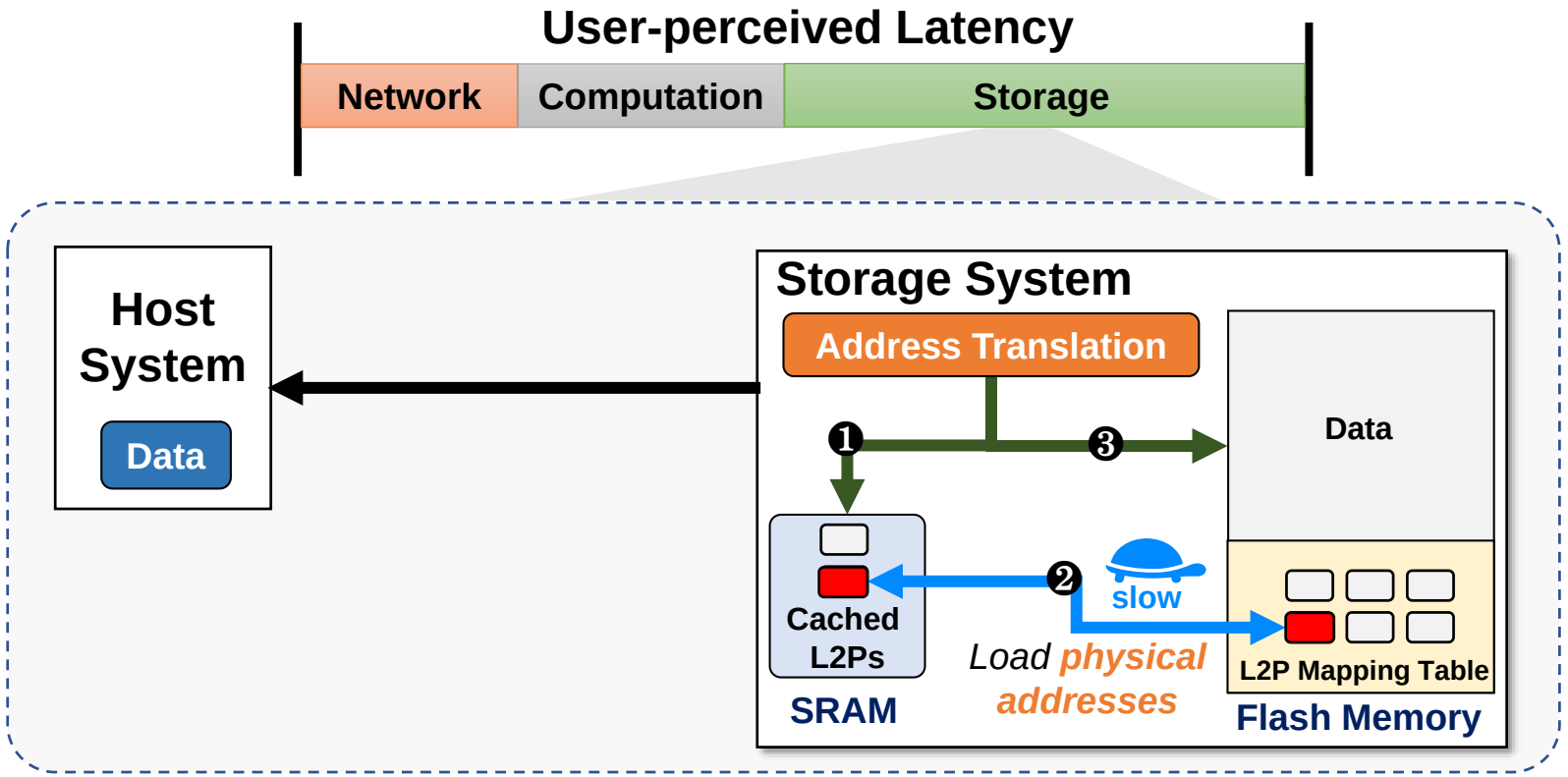
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



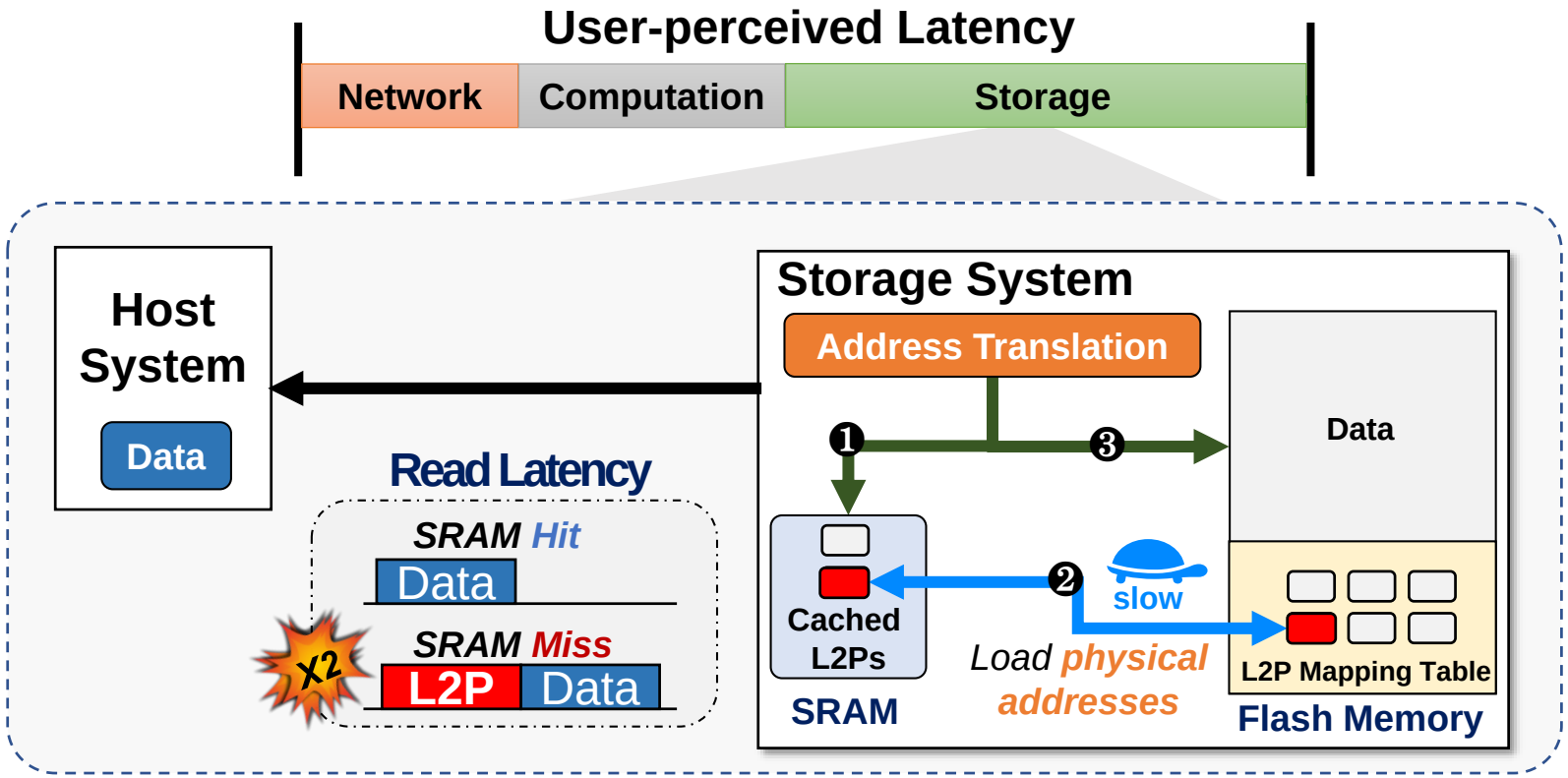
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



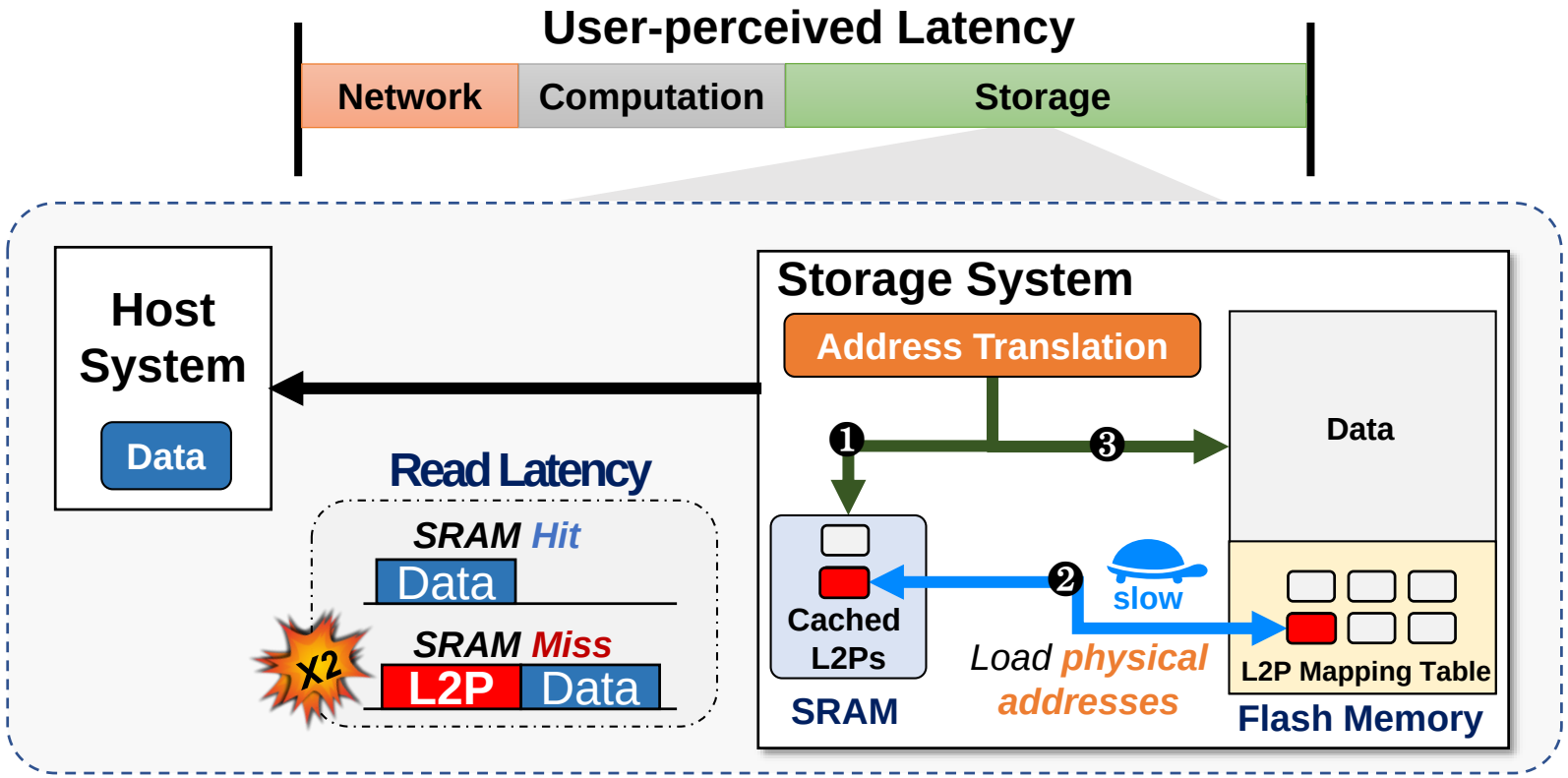
What Causes a Long Storage Latency?

- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



What Causes a Long Storage Latency?

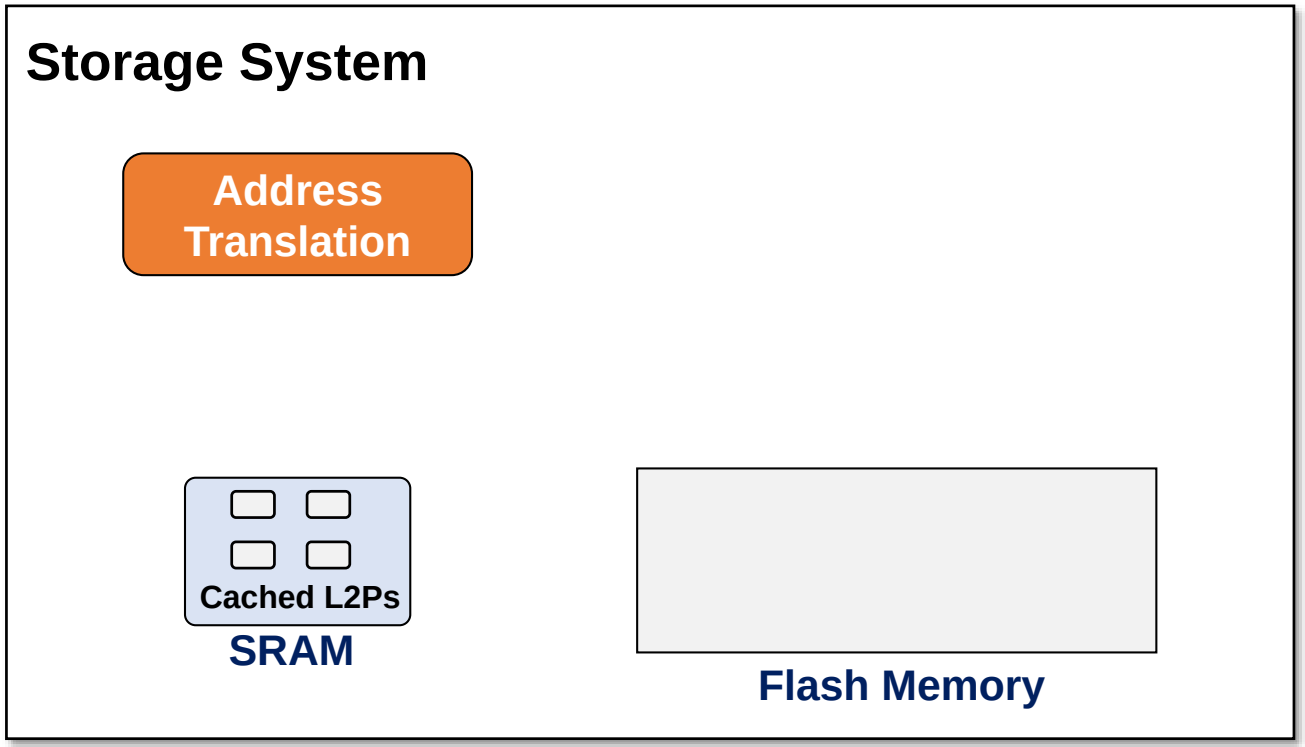
- **Logical-to-physical** (L2P) address translation plays an important role in deciding the read latency.



An L2P cache miss is a key contributor for a long storage latency.

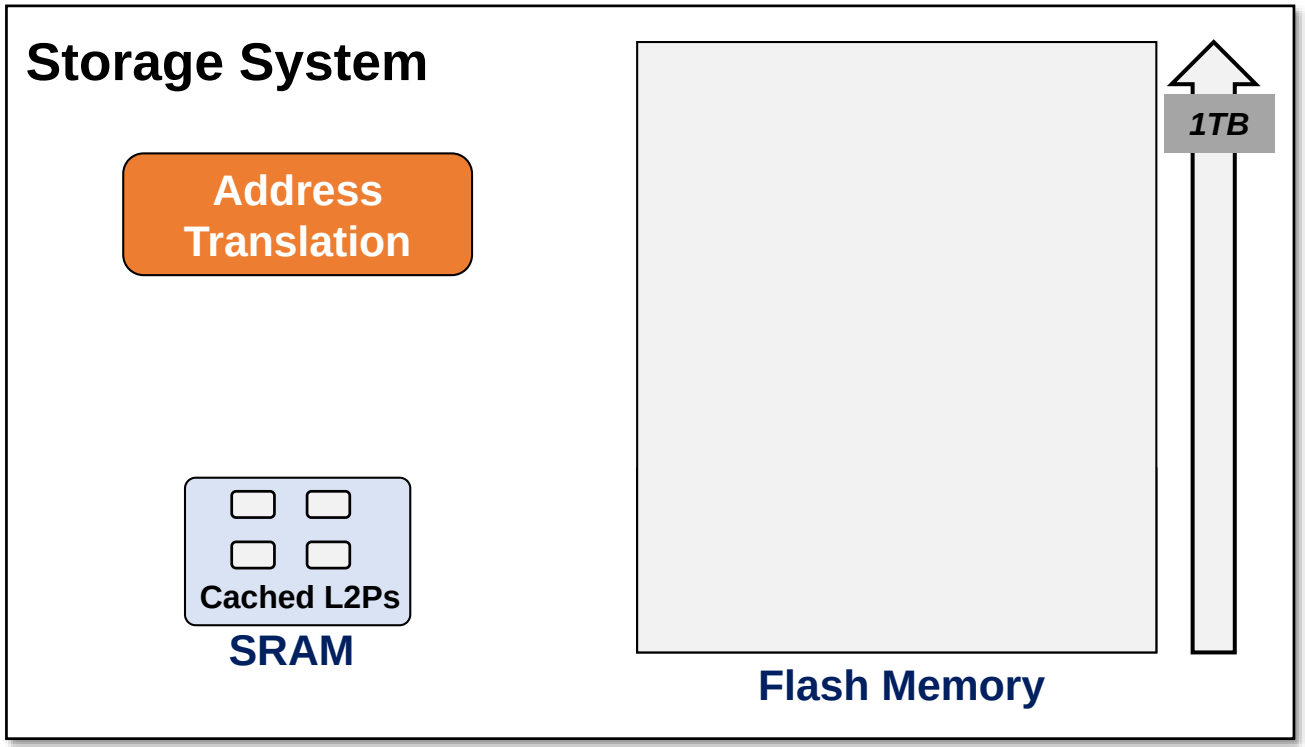
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



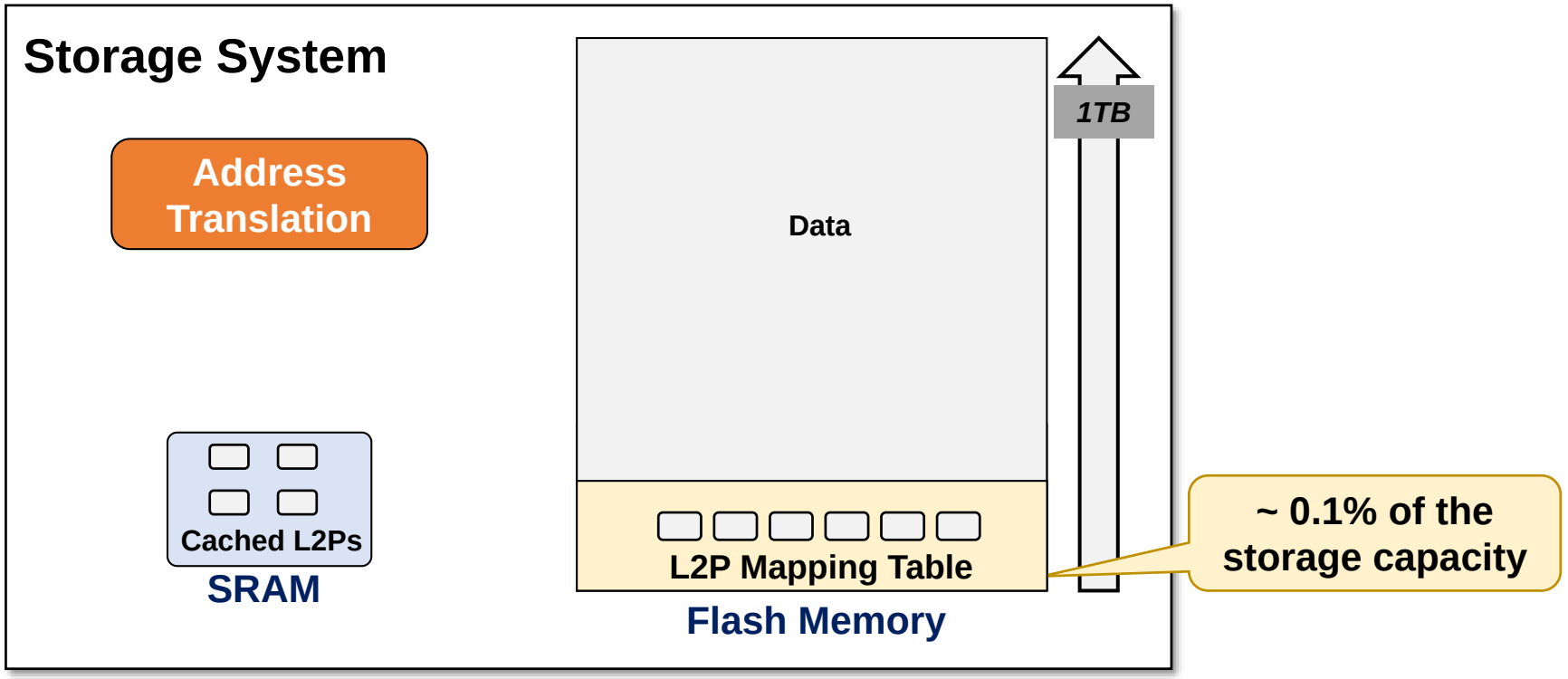
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



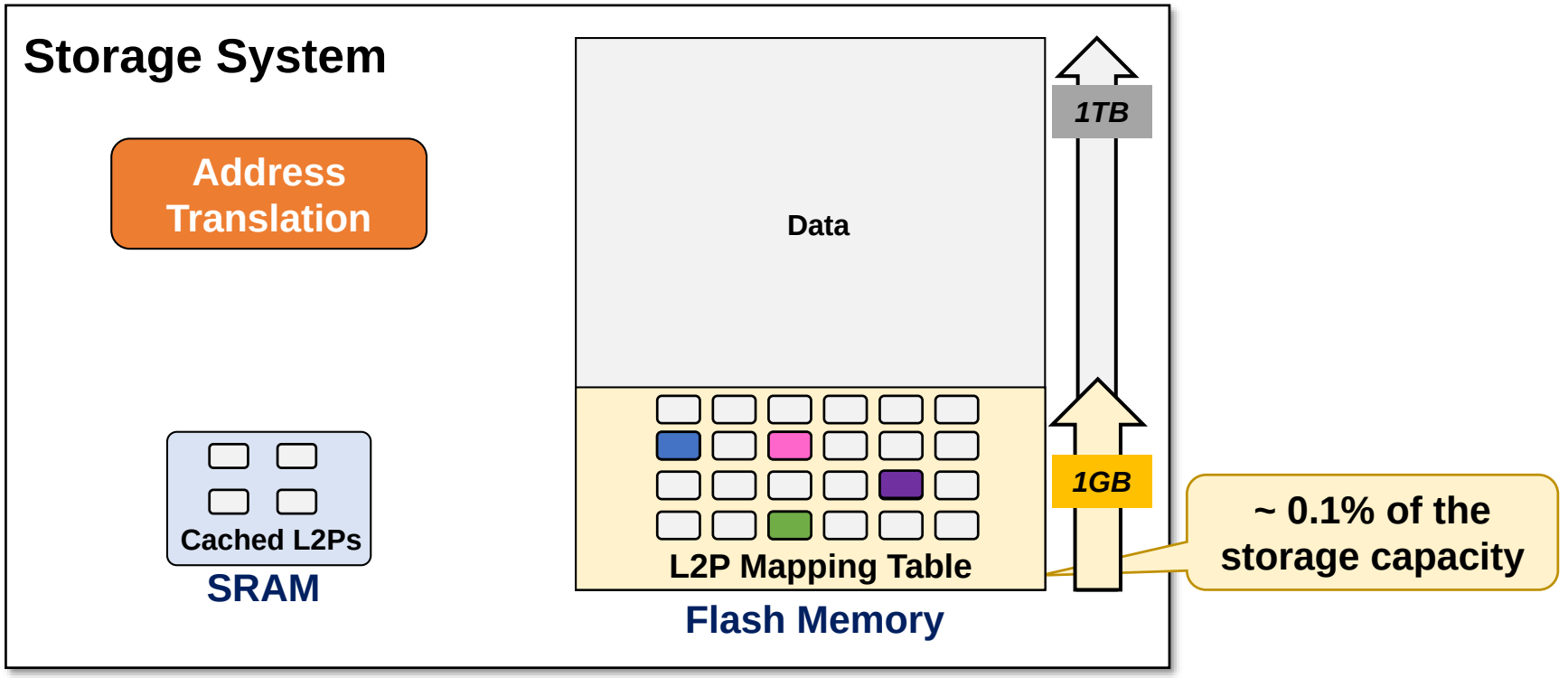
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



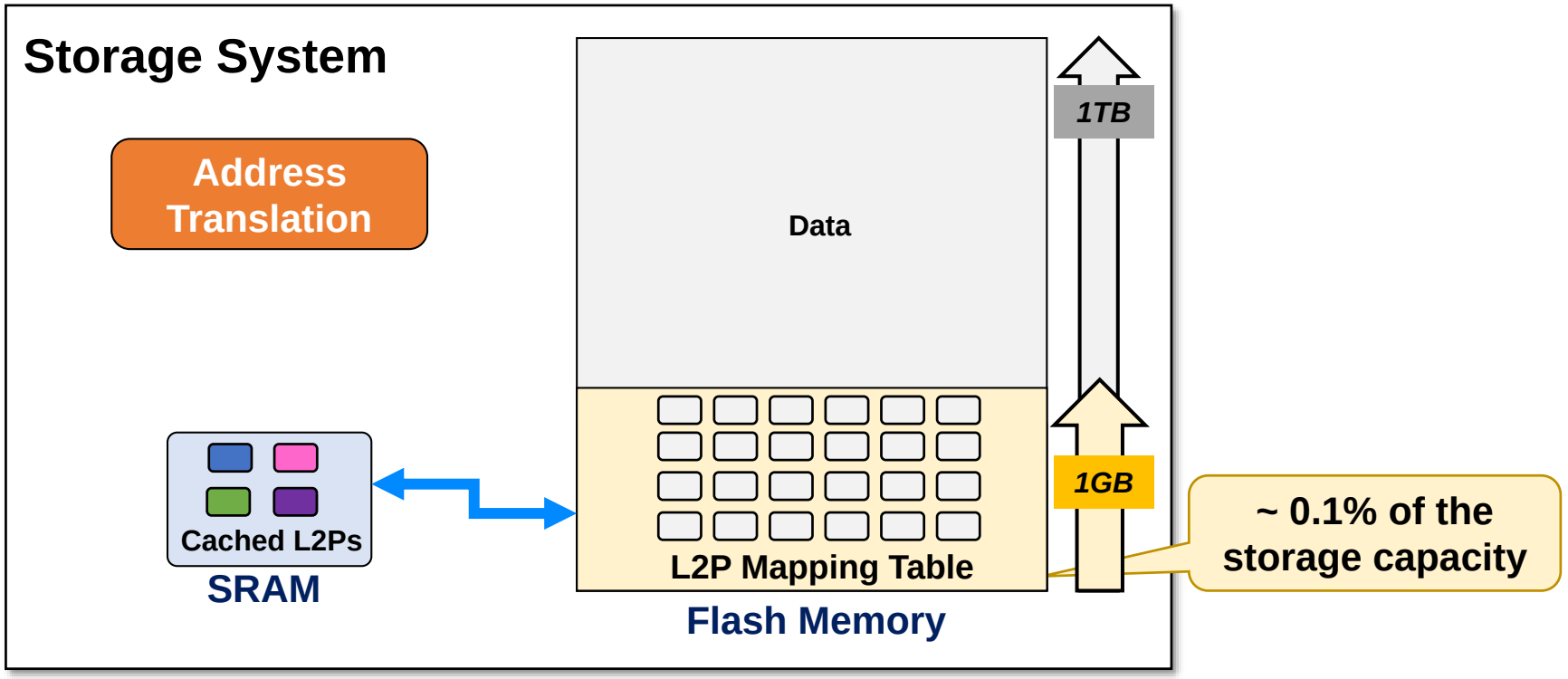
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



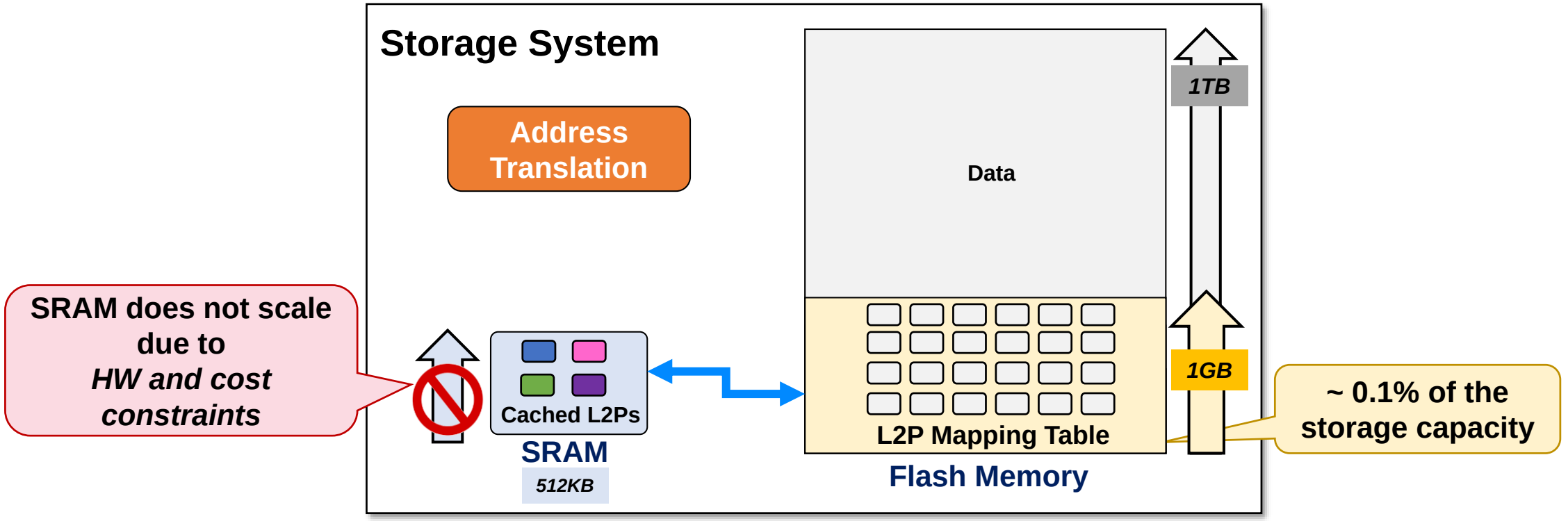
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



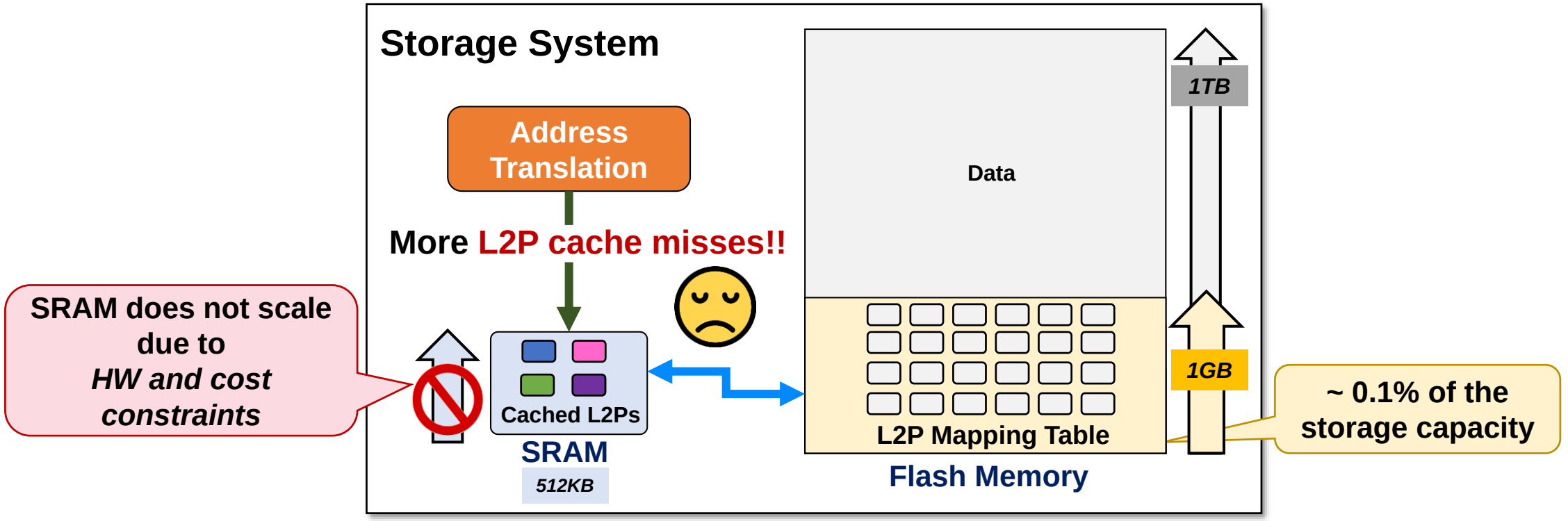
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



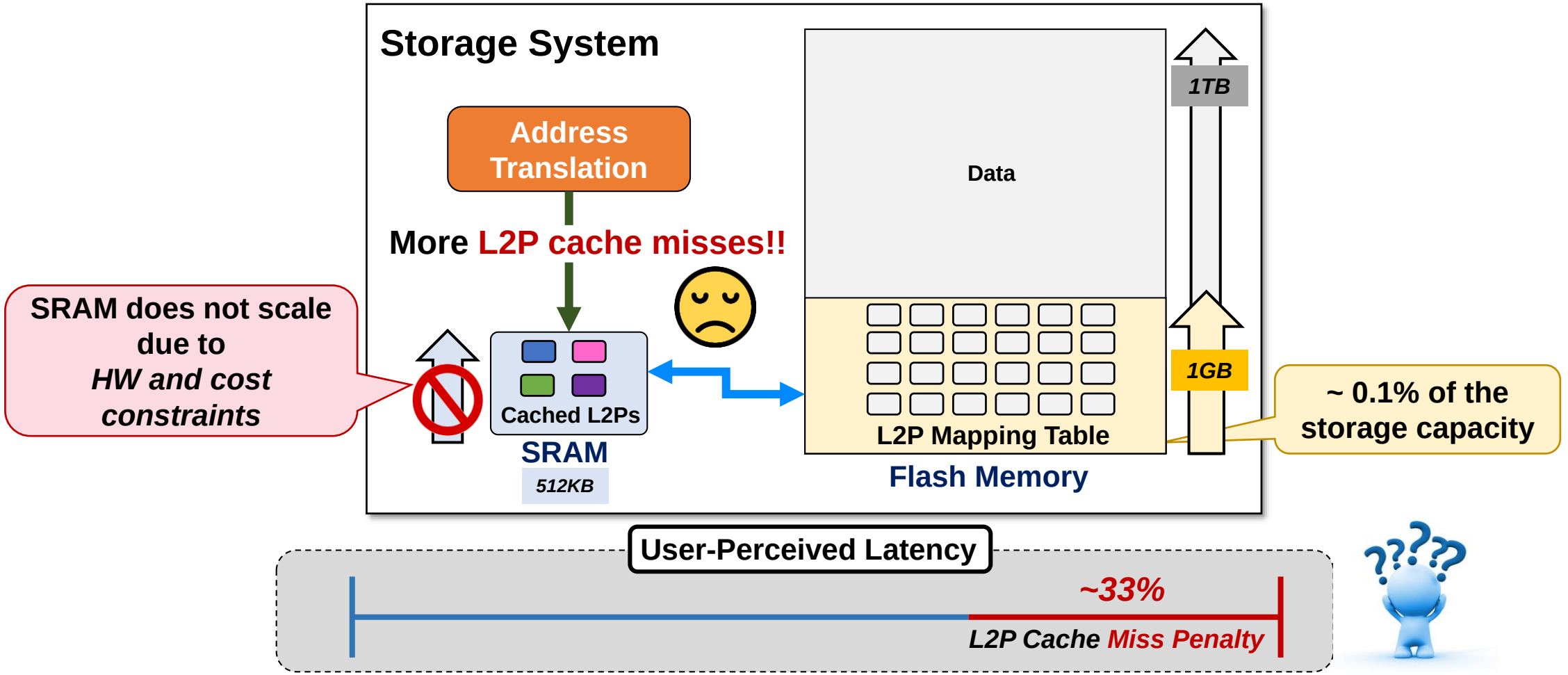
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



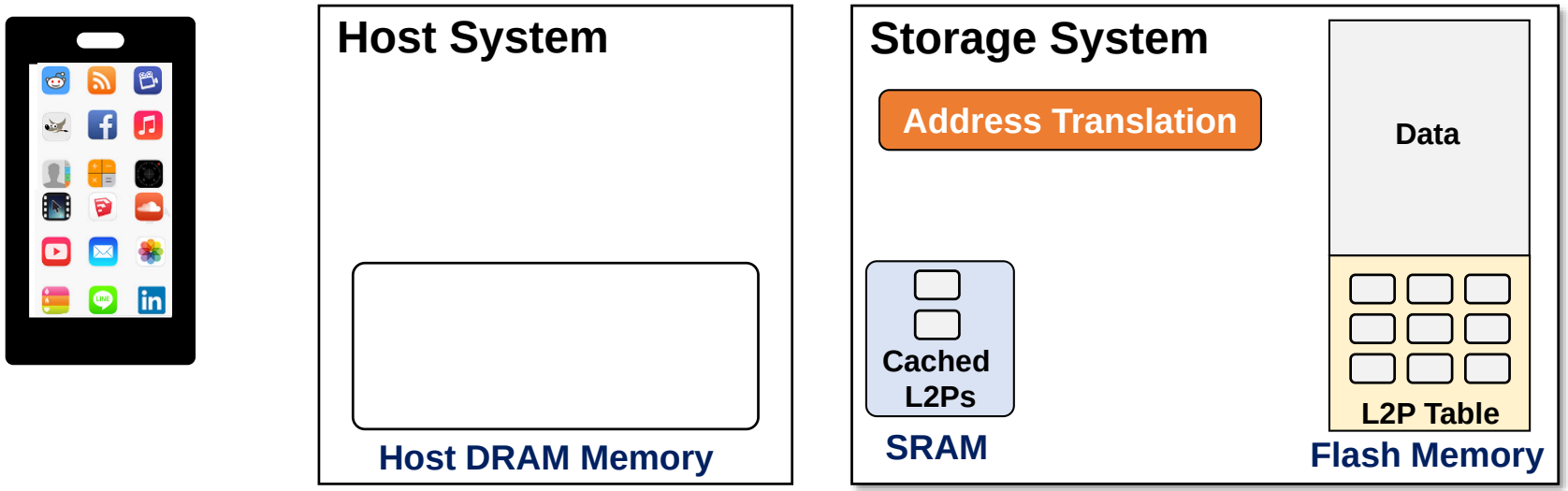
Number of L2P Cache Misses are Increasing

- An increase in storage capacity increases the number of L2P cache misses.



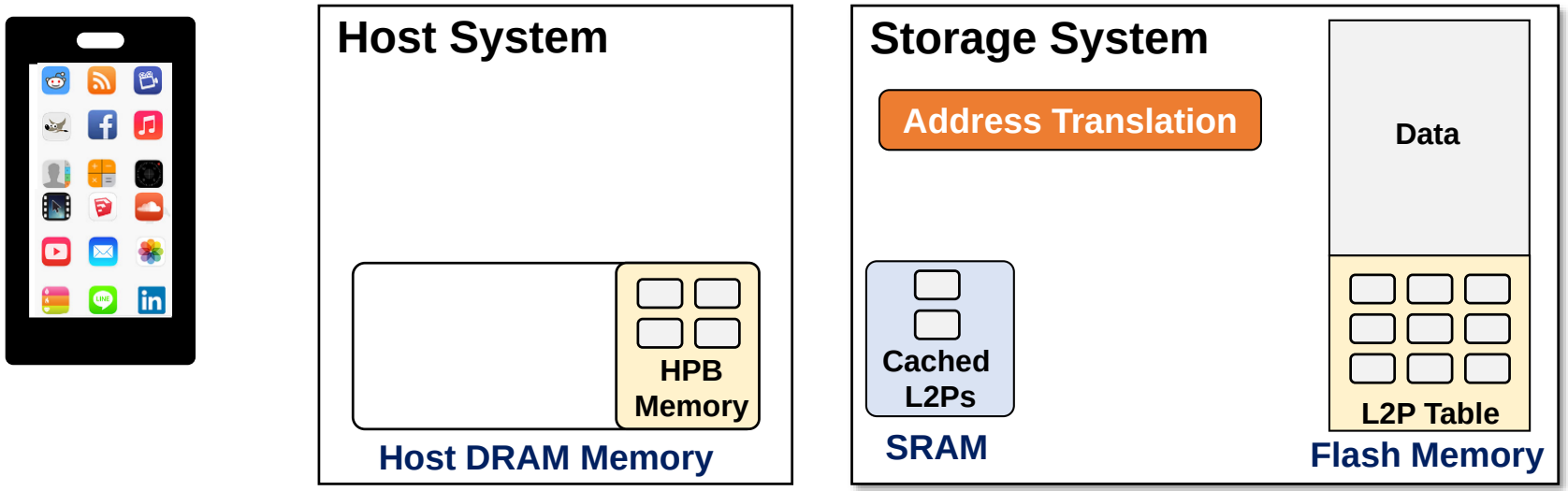
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.



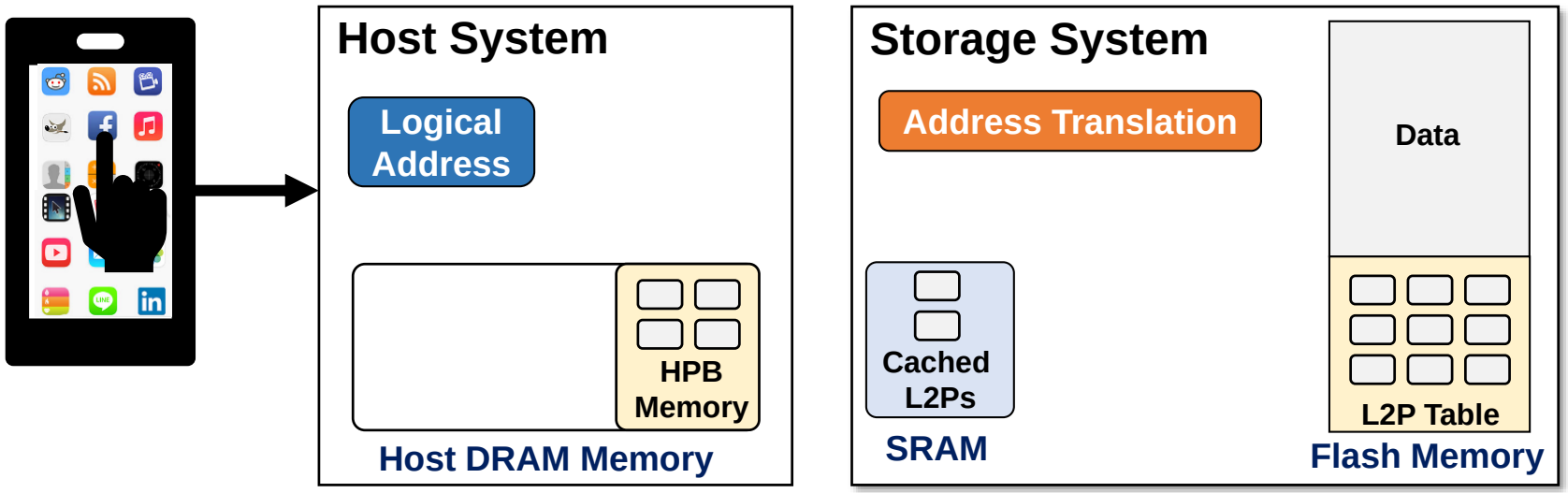
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.



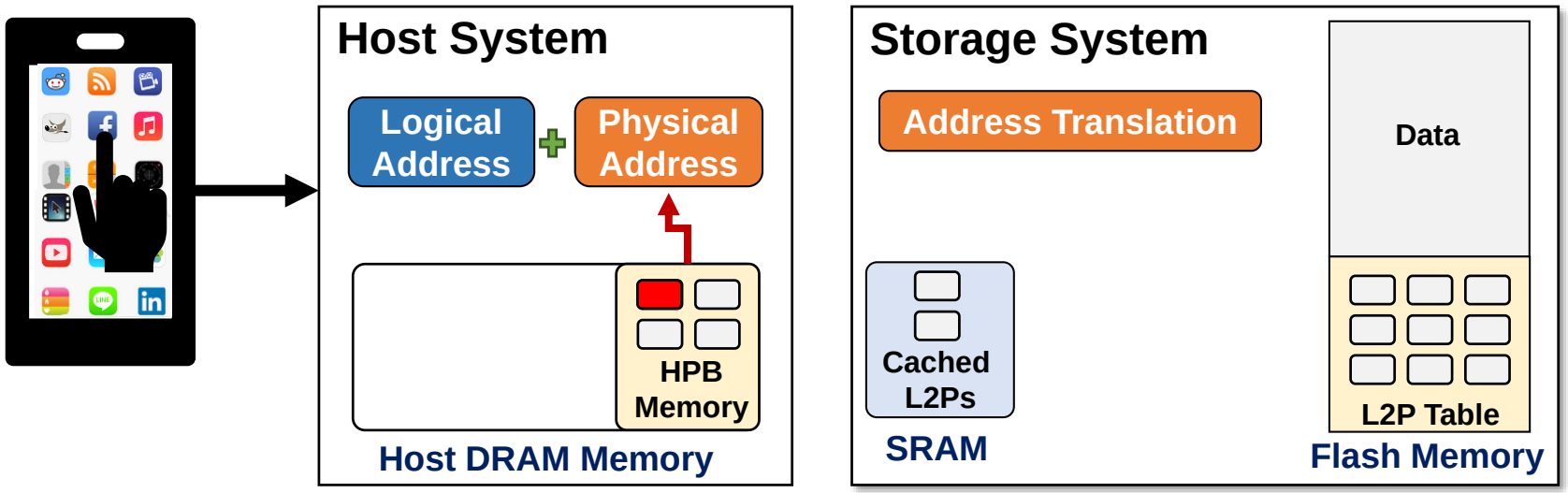
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.



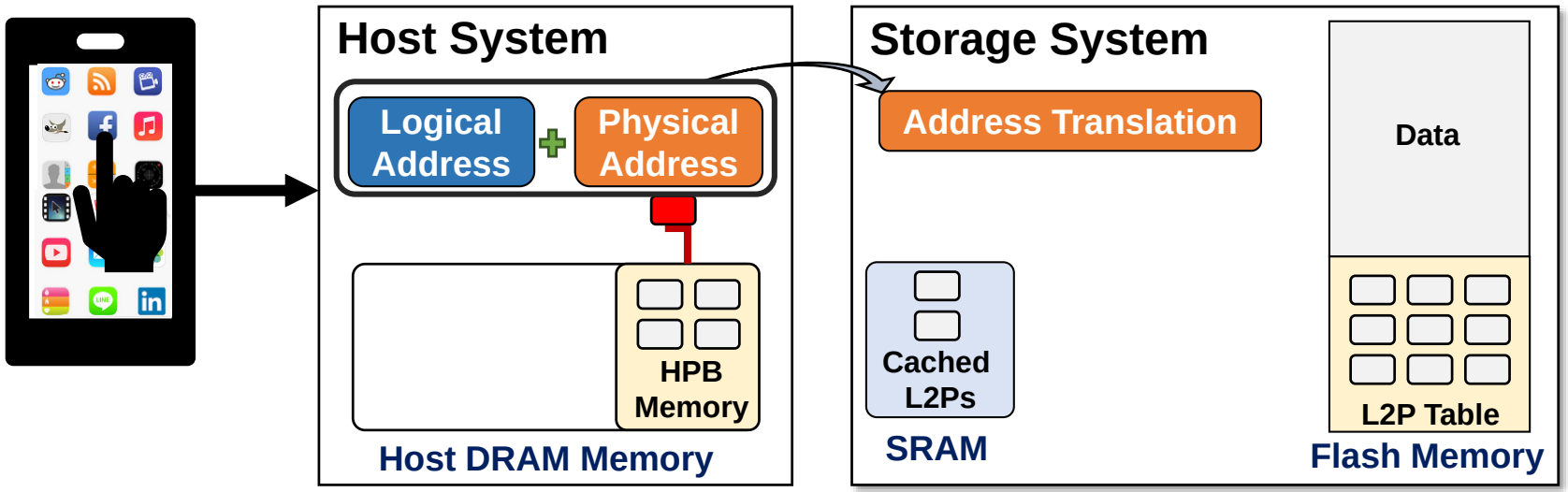
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.



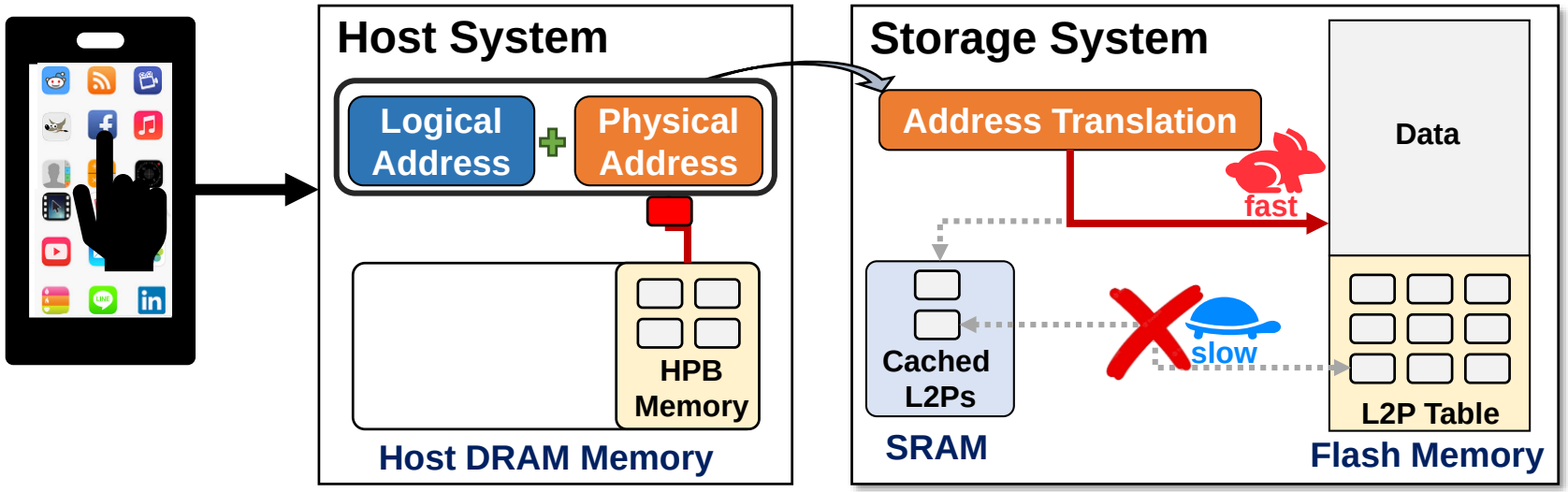
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.



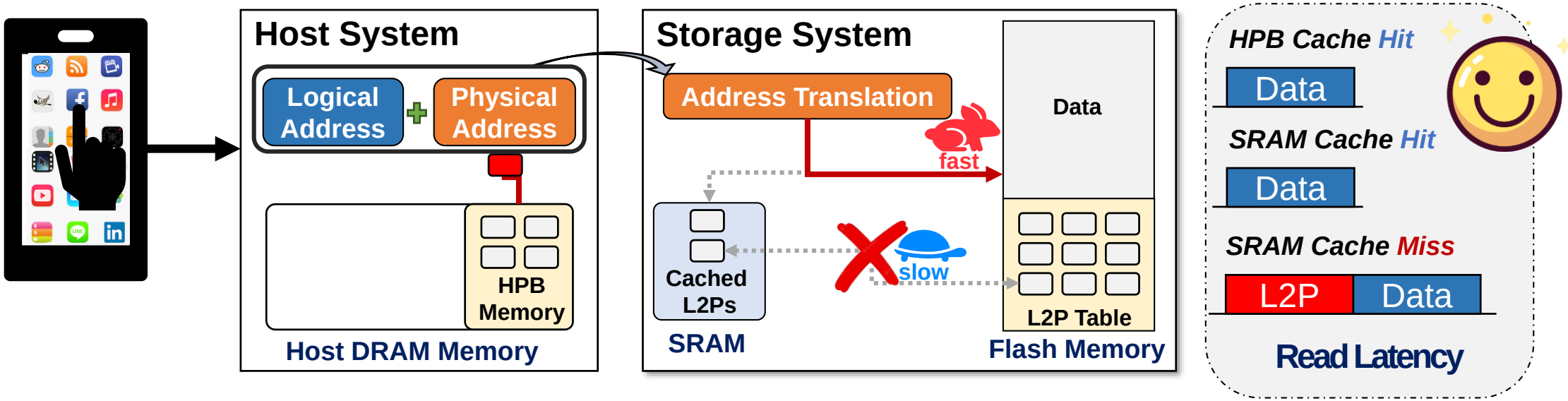
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.
  - Slow flash read operations for missed L2P entries can be avoided.



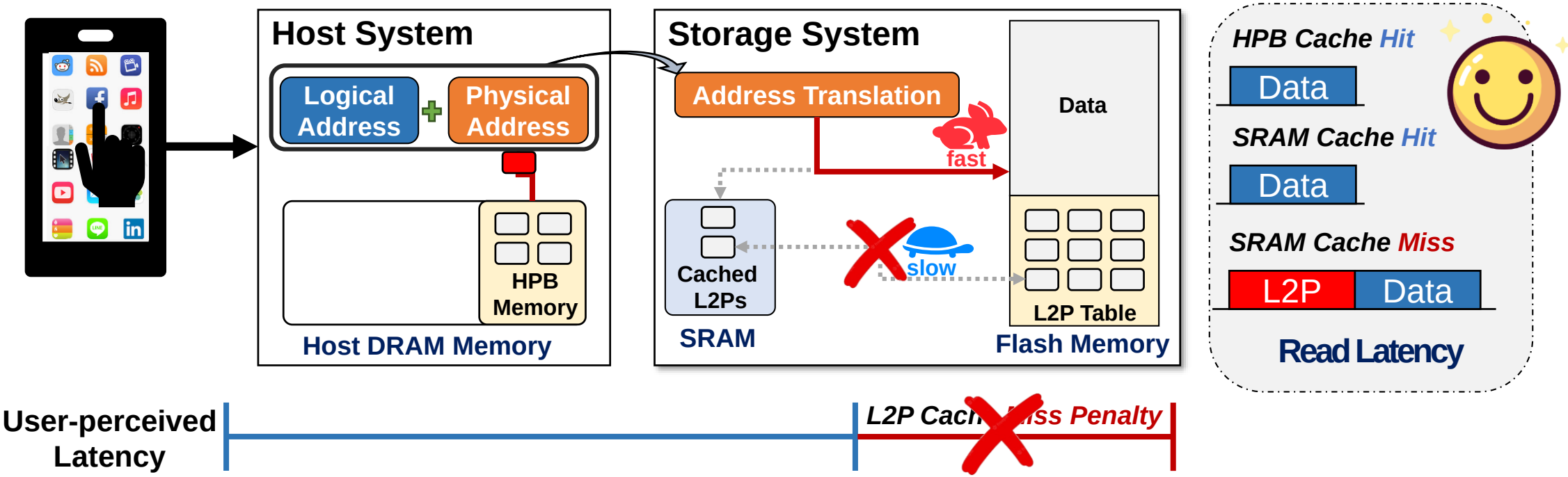
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.
  - Slow flash read operations for missed L2P entries can be avoided.



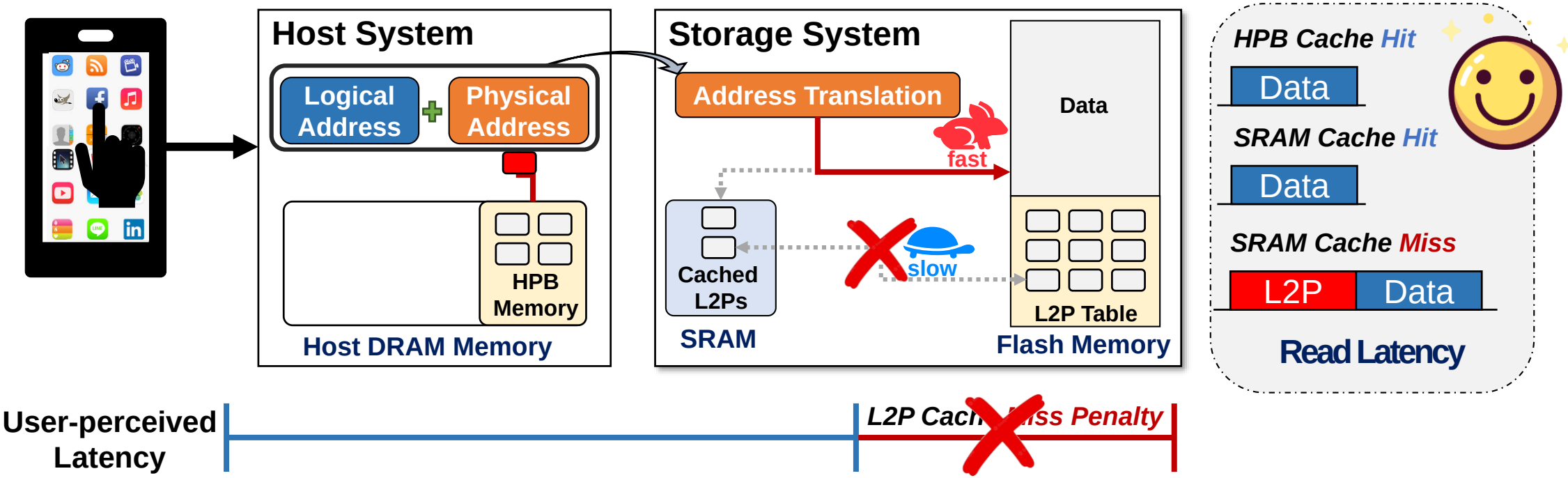
Existing Solution: Host Performance Booster (HPB)

- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.
  - Slow flash read operations for missed L2P entries can be avoided.



Existing Solution: Host Performance Booster (HPB)

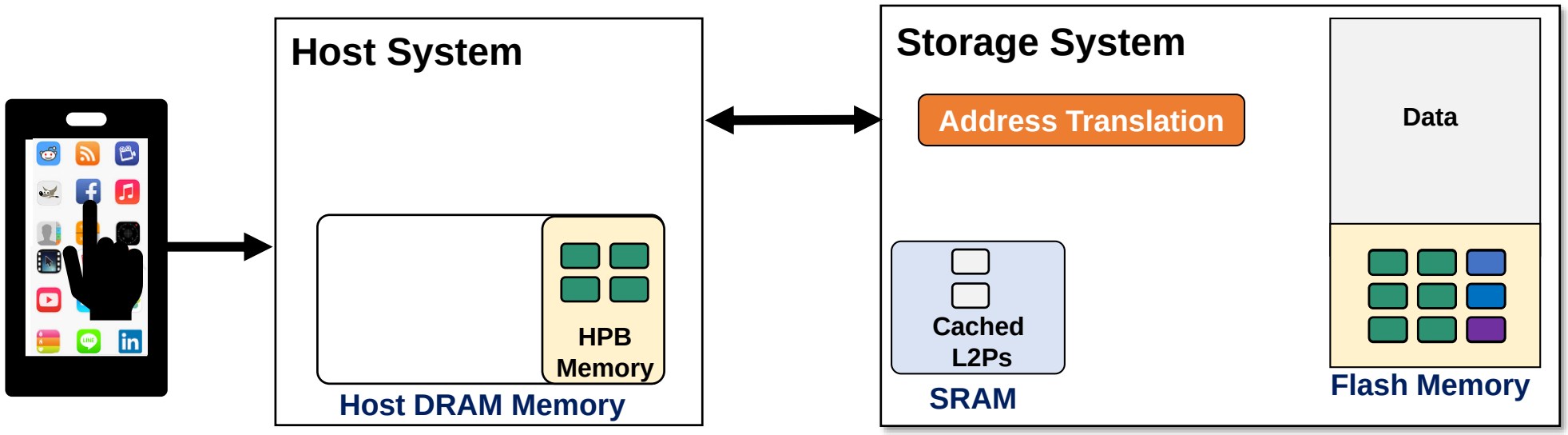
- HPB caches L2P entries in the host-side DRAM to reduce L2P misses.
  - Slow flash read operations for missed L2P entries can be avoided.



Integrating HPB can possibly provide better UX quality by reducing L2P cache misses.

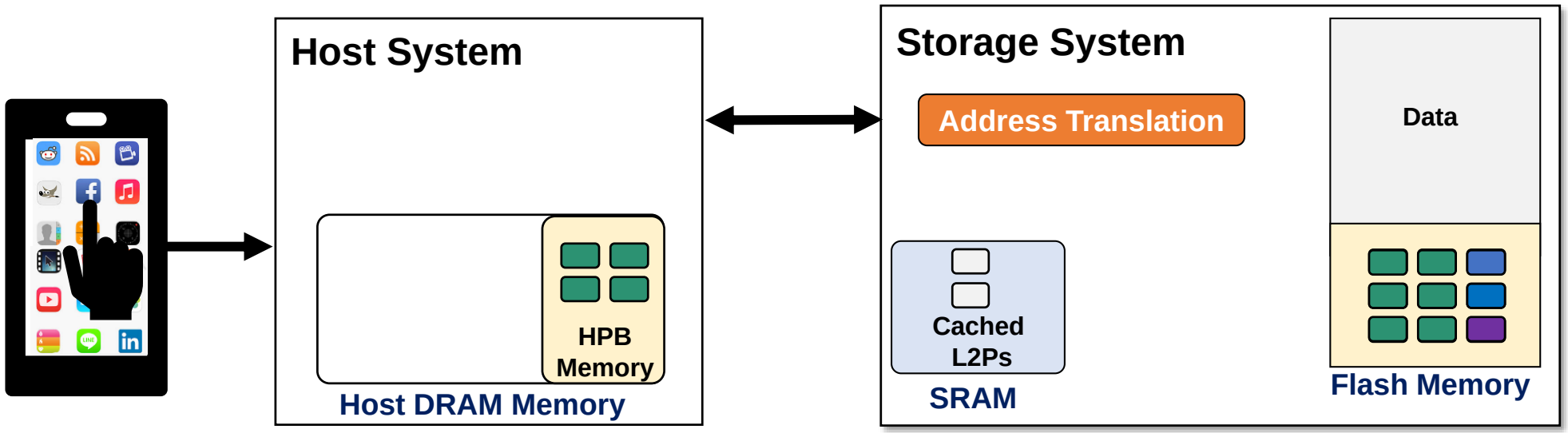
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.



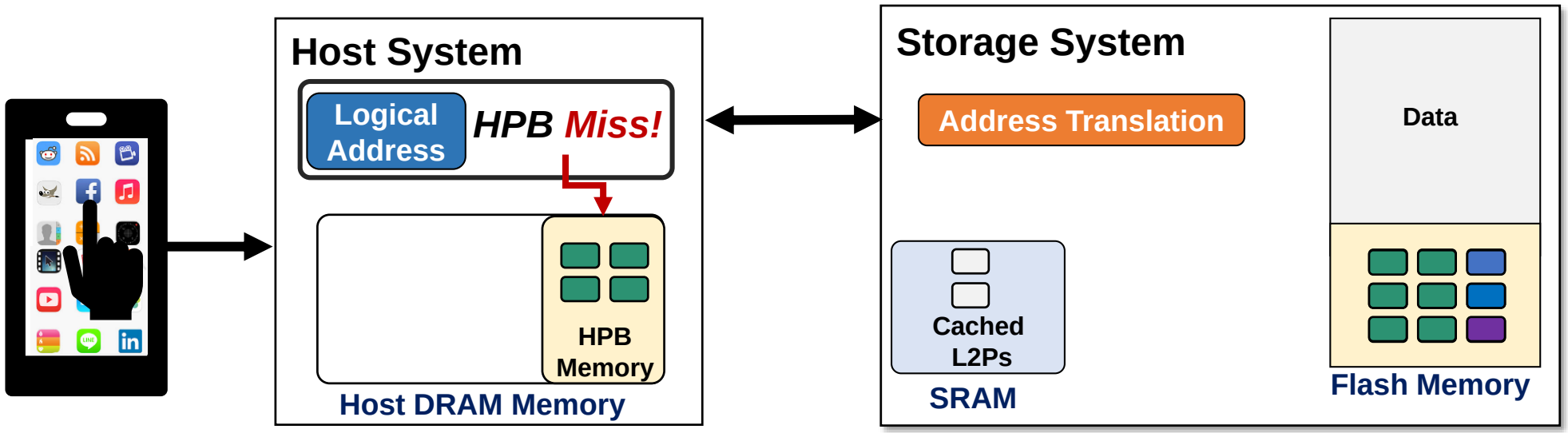
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*



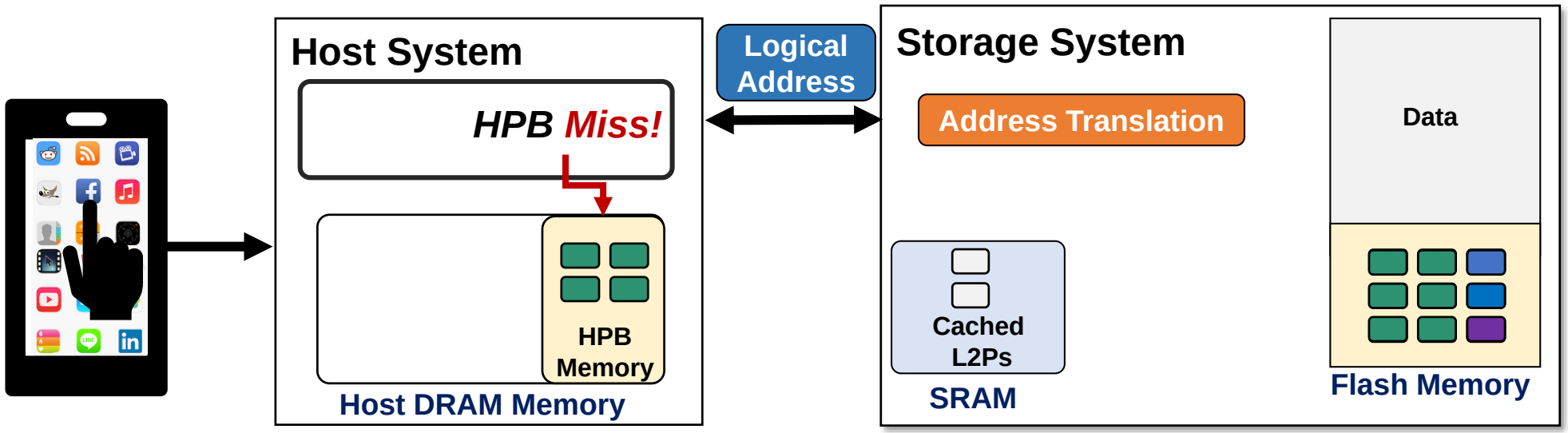
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*



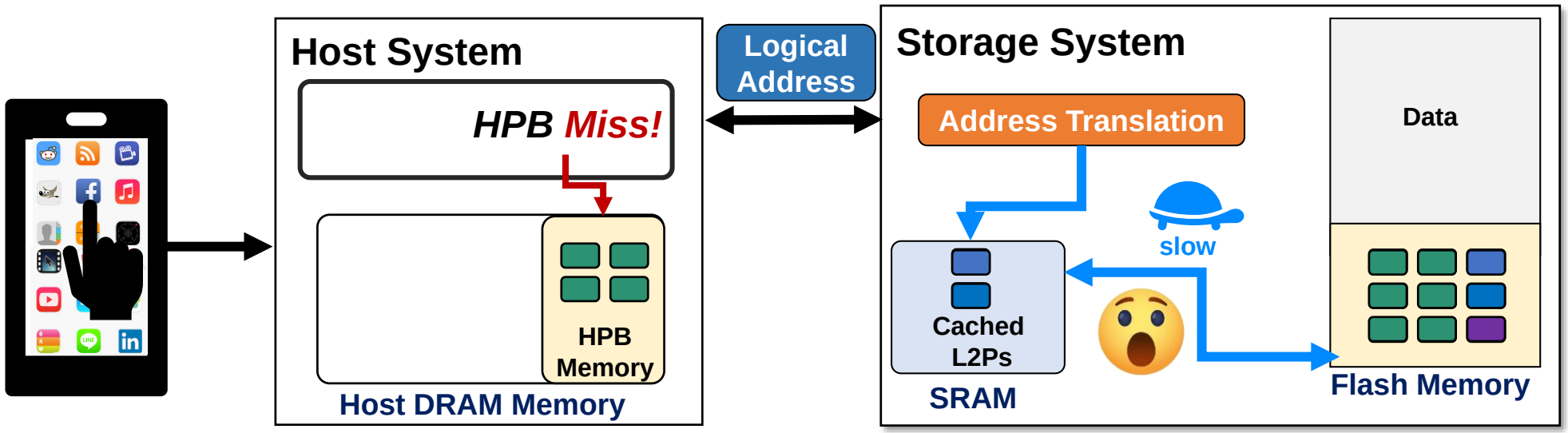
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*



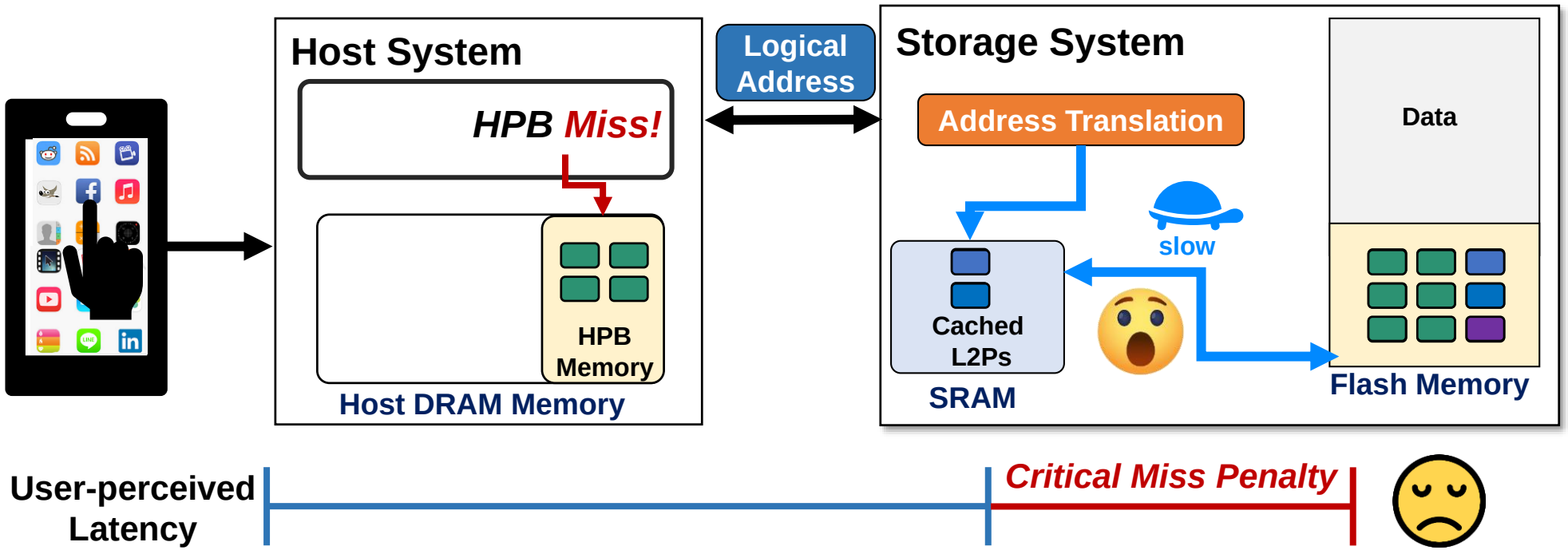
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*



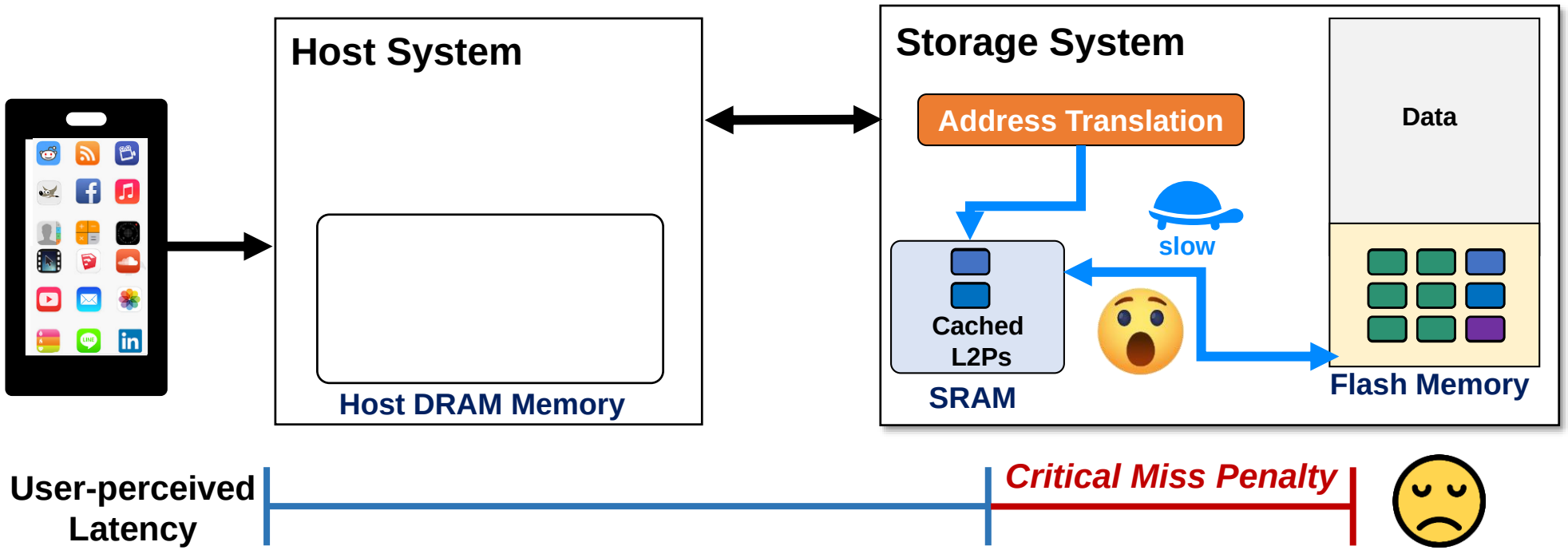
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.



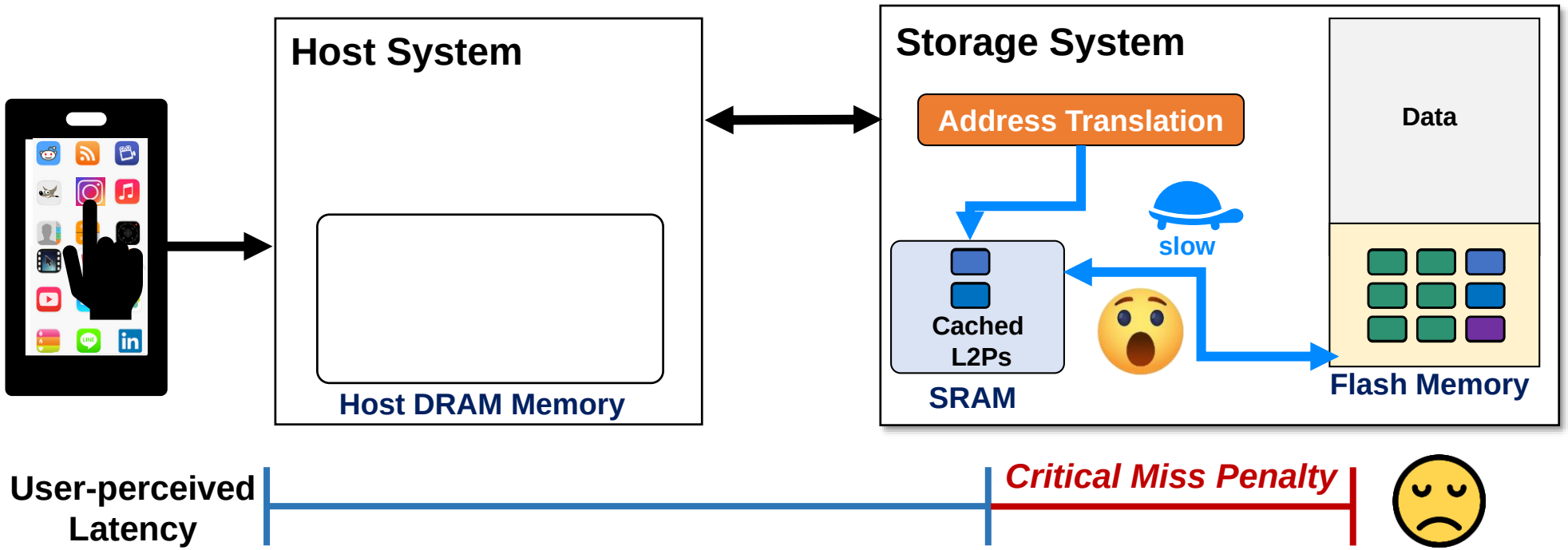
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



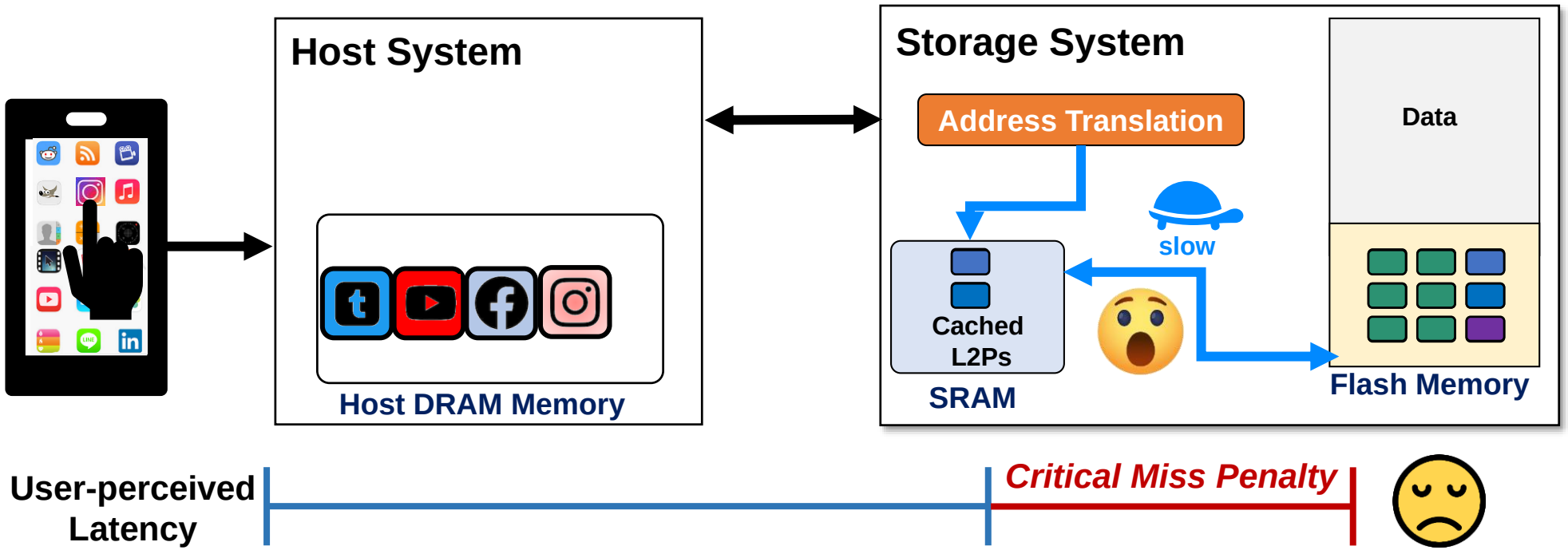
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



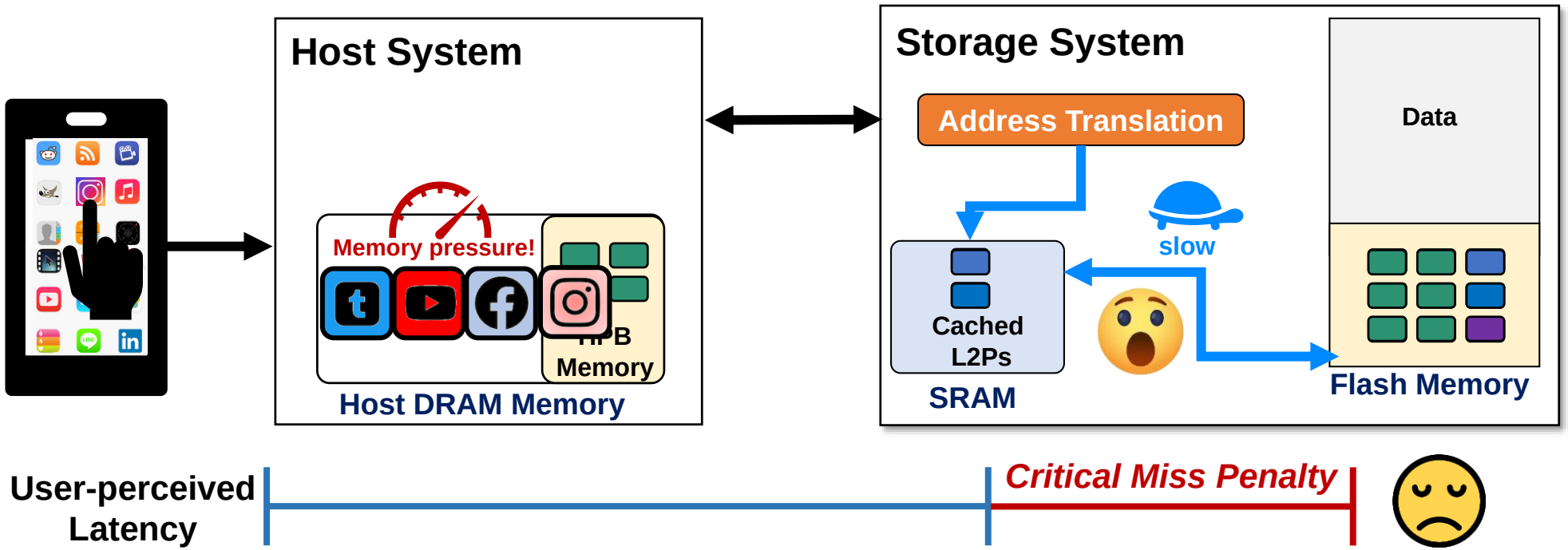
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



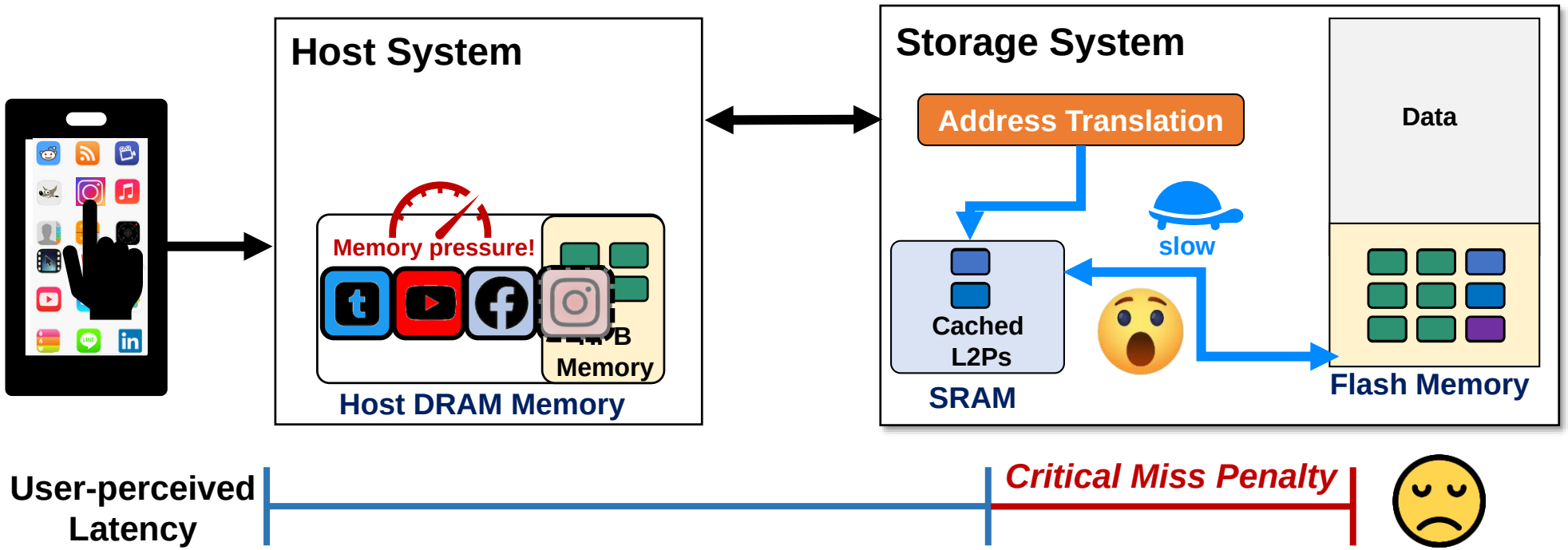
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*
  - **Limitation 2**: HPB may **kill** additional **apps**.



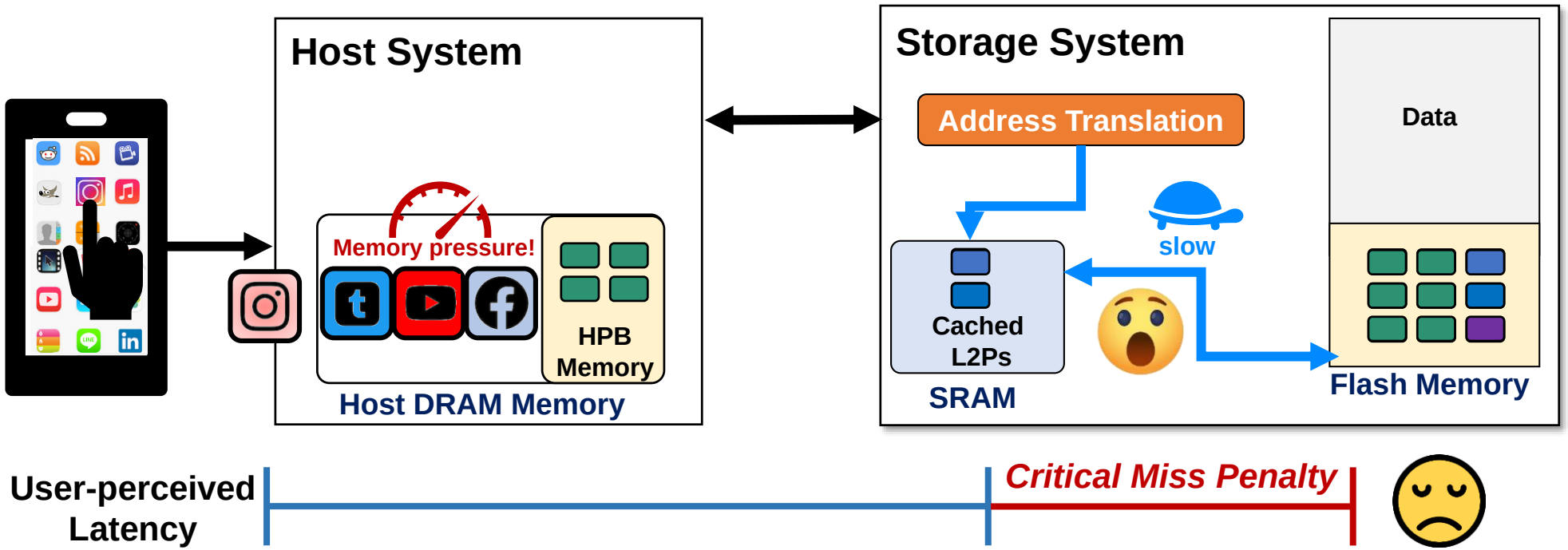
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB and device-side SRAM memory.*
  - **Limitation 2**: HPB may **kill** additional **apps**.



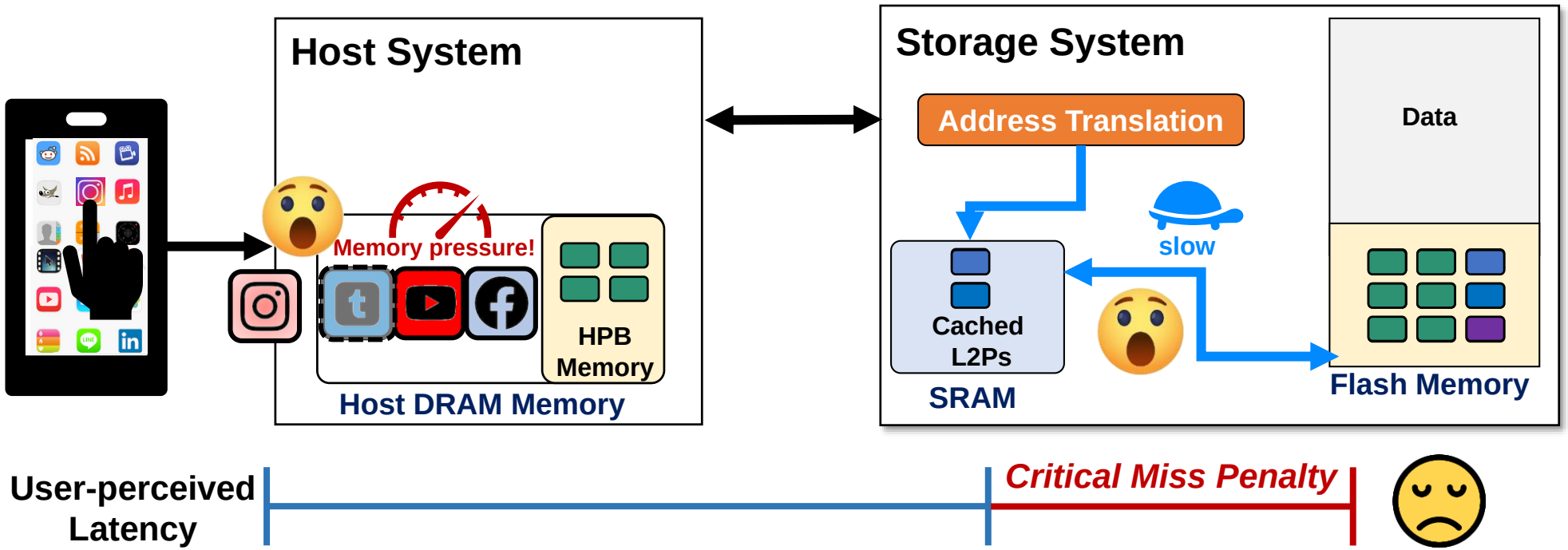
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



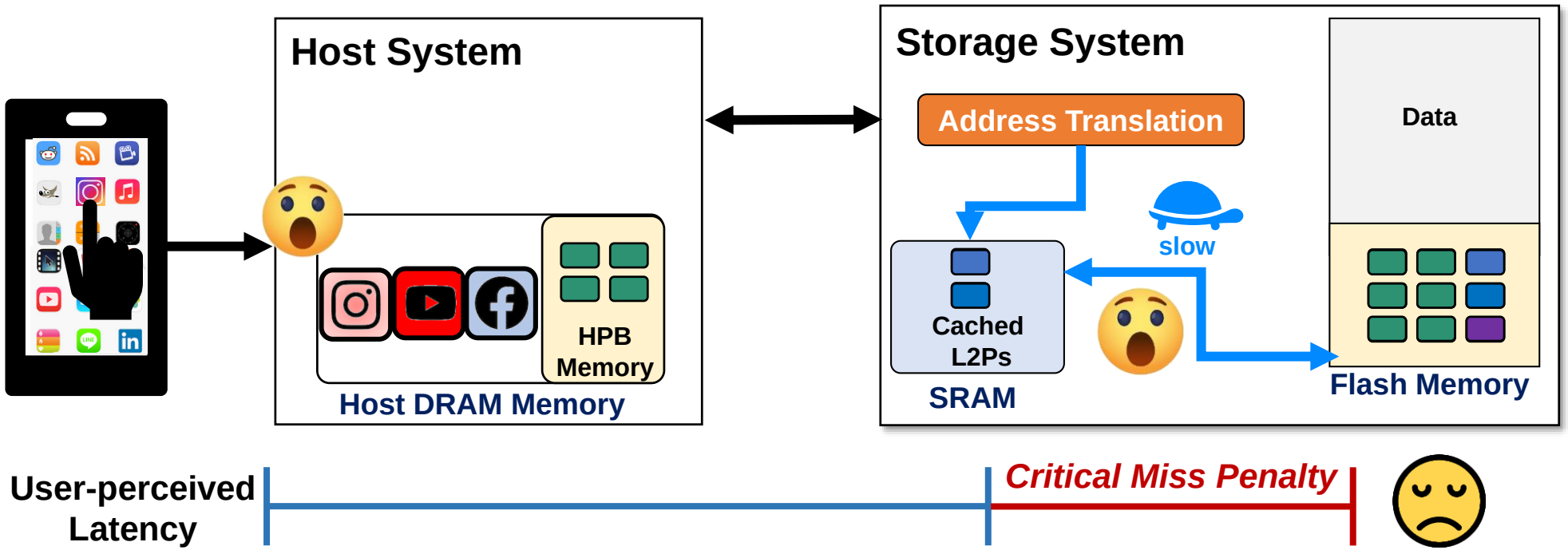
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



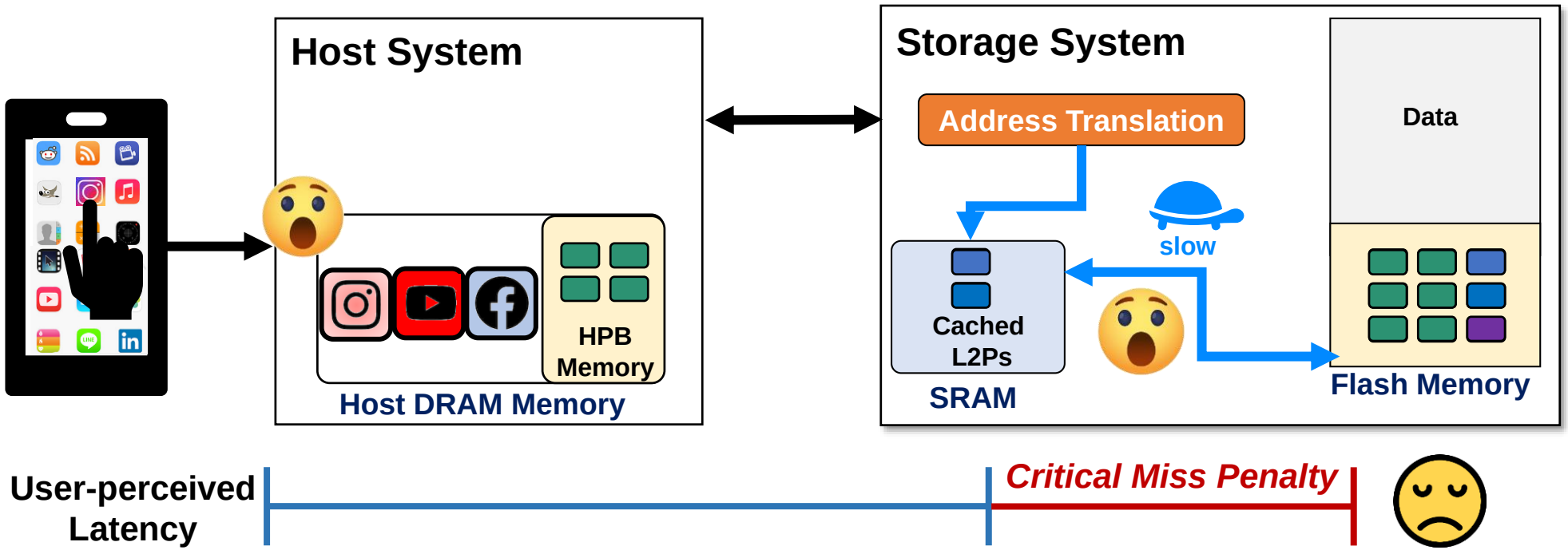
Limitations of the Existing HPB Implementation

- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



Limitations of the Existing HPB Implementation

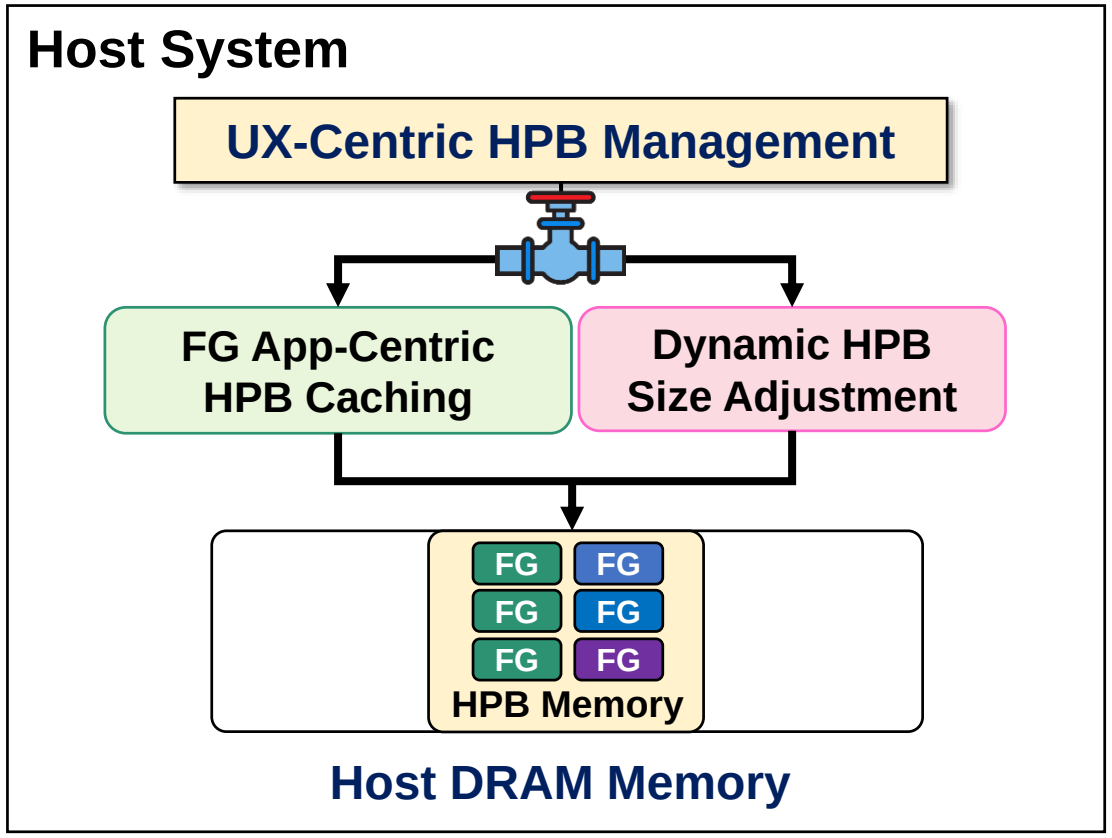
- Manages HPB memory in a ***UX-oblivious storage-centric*** manner.
  - **Limitation 1**: A significant amount of **critical misses** exist.
    - **Critical miss:** *L2P cache misses from both HPB* and device-side *SRAM memory*.
  - **Limitation 2**: HPB may **kill** additional **apps**.



We need a UX-centric HPB management policy.

Design Goal of Proposed Work

- **HPBvalve**: Manage HPB memory in a **UX-Centric Manner**!
  - HPB memory should cache L2P entries of FG apps over BG apps.
  - User apps should not suffer from reduced working memory space.

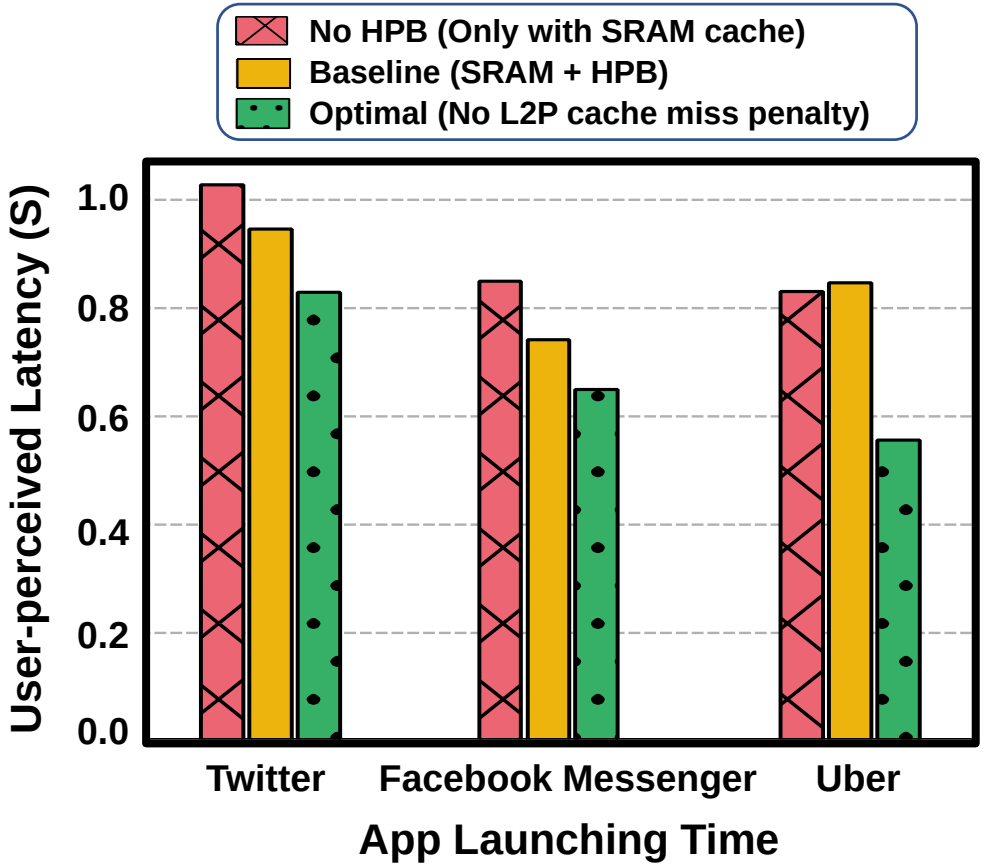


Outline

- Root Cause Analysis of the Existing Solution
- Design of HPBvalve
- Evaluation Results

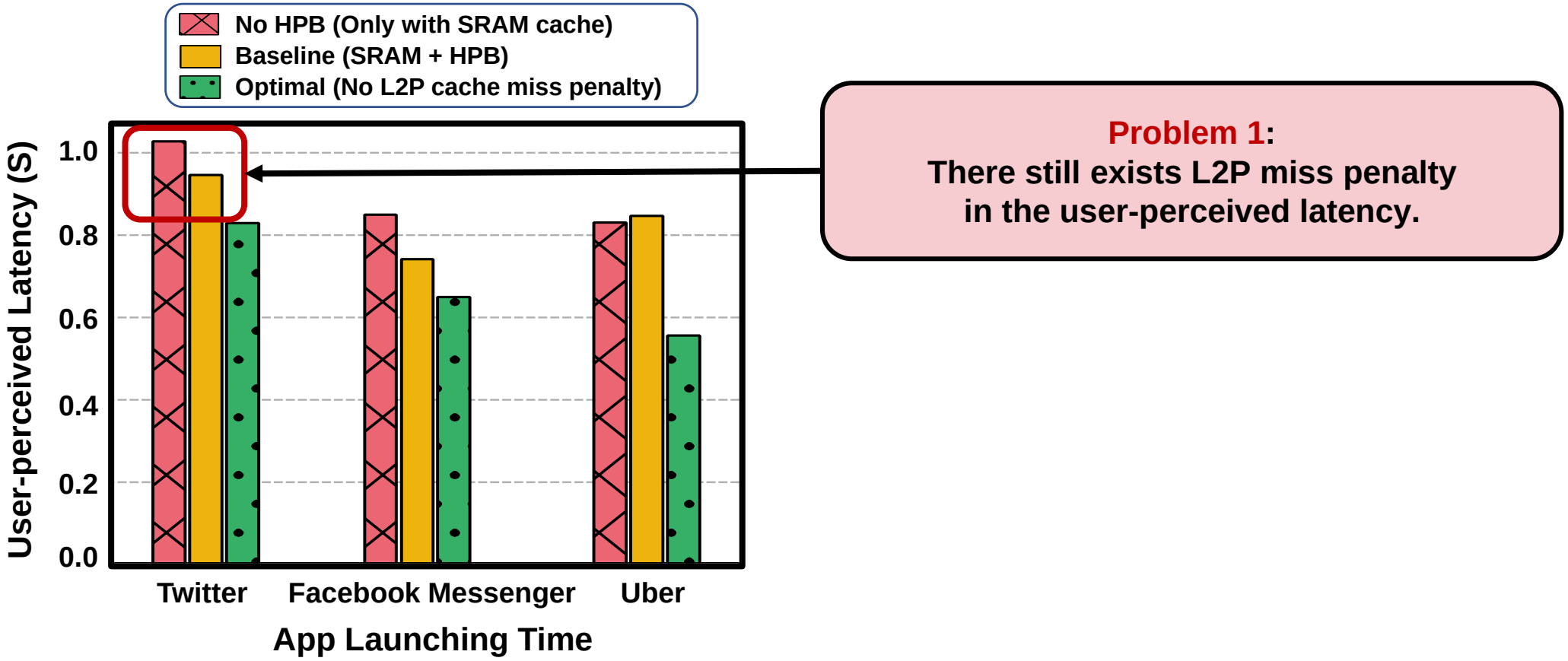
Evaluation of the Existing HPB Solution

- Simply integrating HPB **does not guarantee** improved quality of UX.



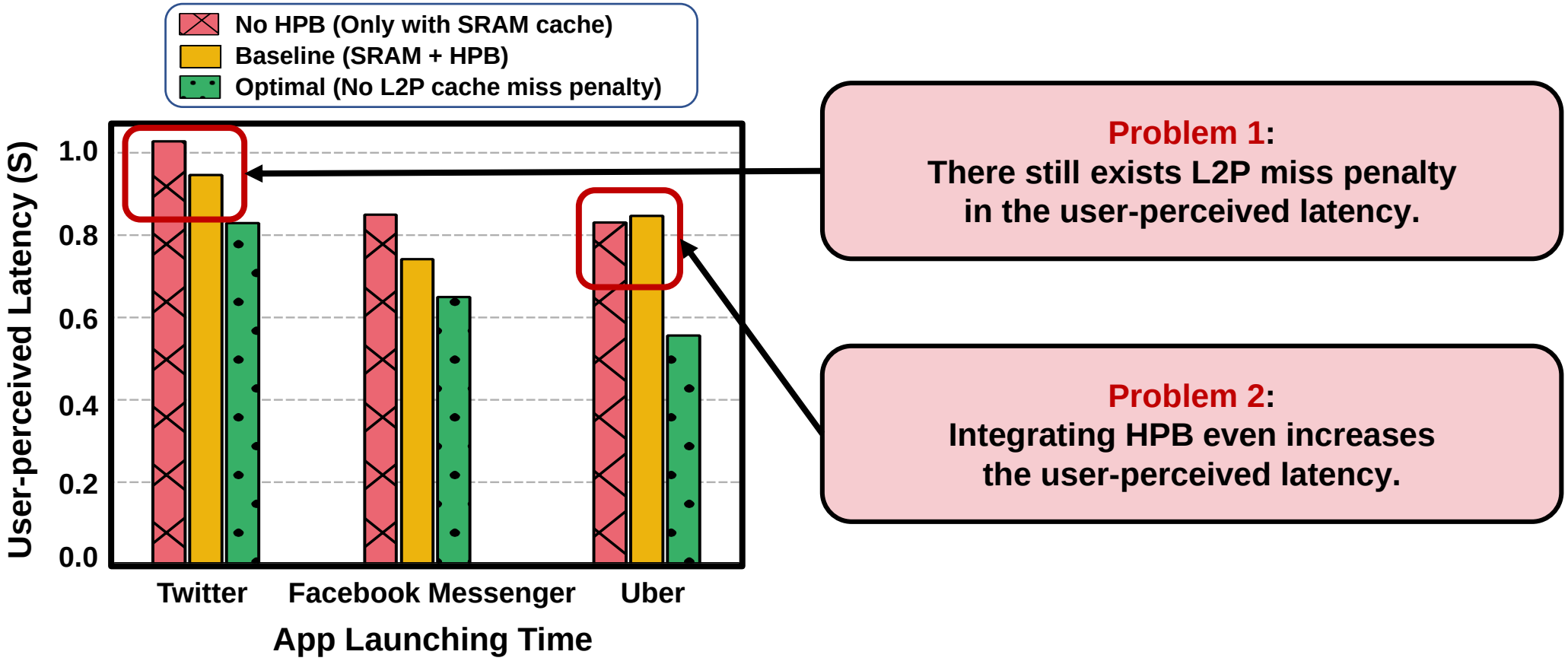
Evaluation of the Existing HPB Solution

- Simply integrating HPB **does not guarantee** improved quality of UX.



Evaluation of the Existing HPB Solution

- Simply integrating HPB **does not guarantee** improved quality of UX.



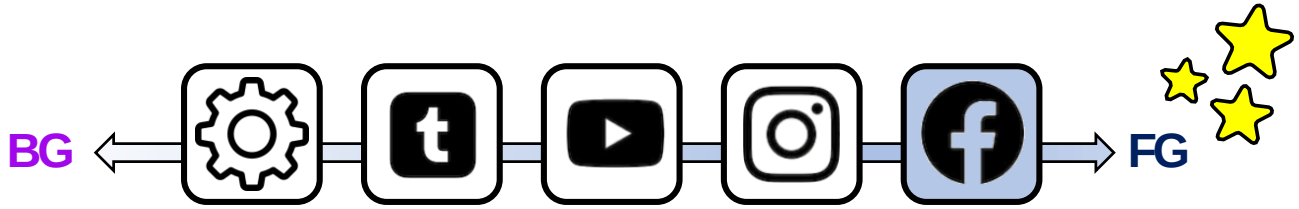
A large room for improvement exists in the current HPB management scheme.

Root Cause 1: App Priority-Oblivious HPB Caching

- Current HPB caching policy results in **priority inversion** between *foreground (FG)* and *background (BG) apps*.

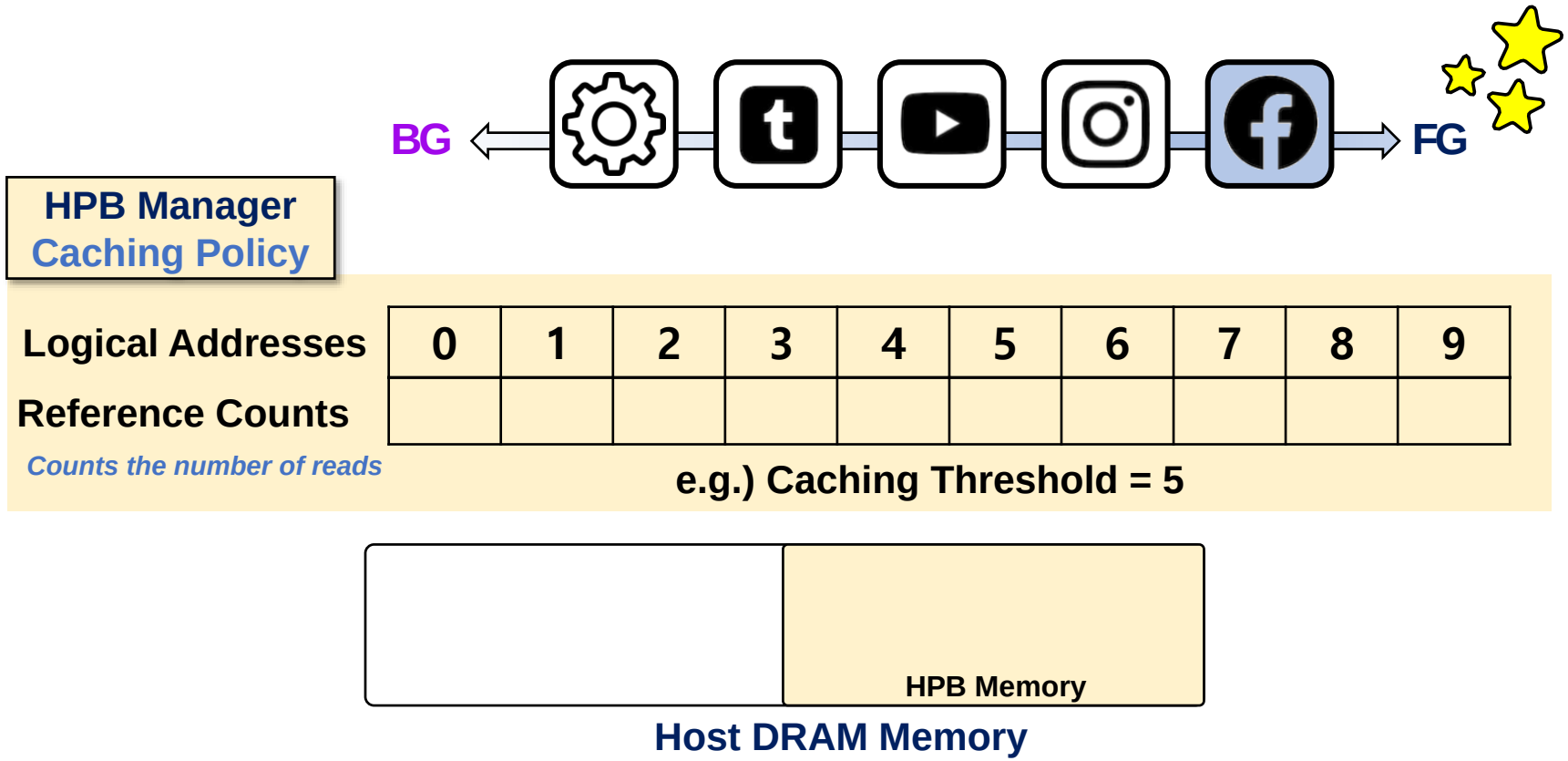
Root Cause 1: App Priority-Oblivious HPB Caching

- Current HPB caching policy results in **priority inversion** between *foreground (FG)* and *background (BG) apps*.



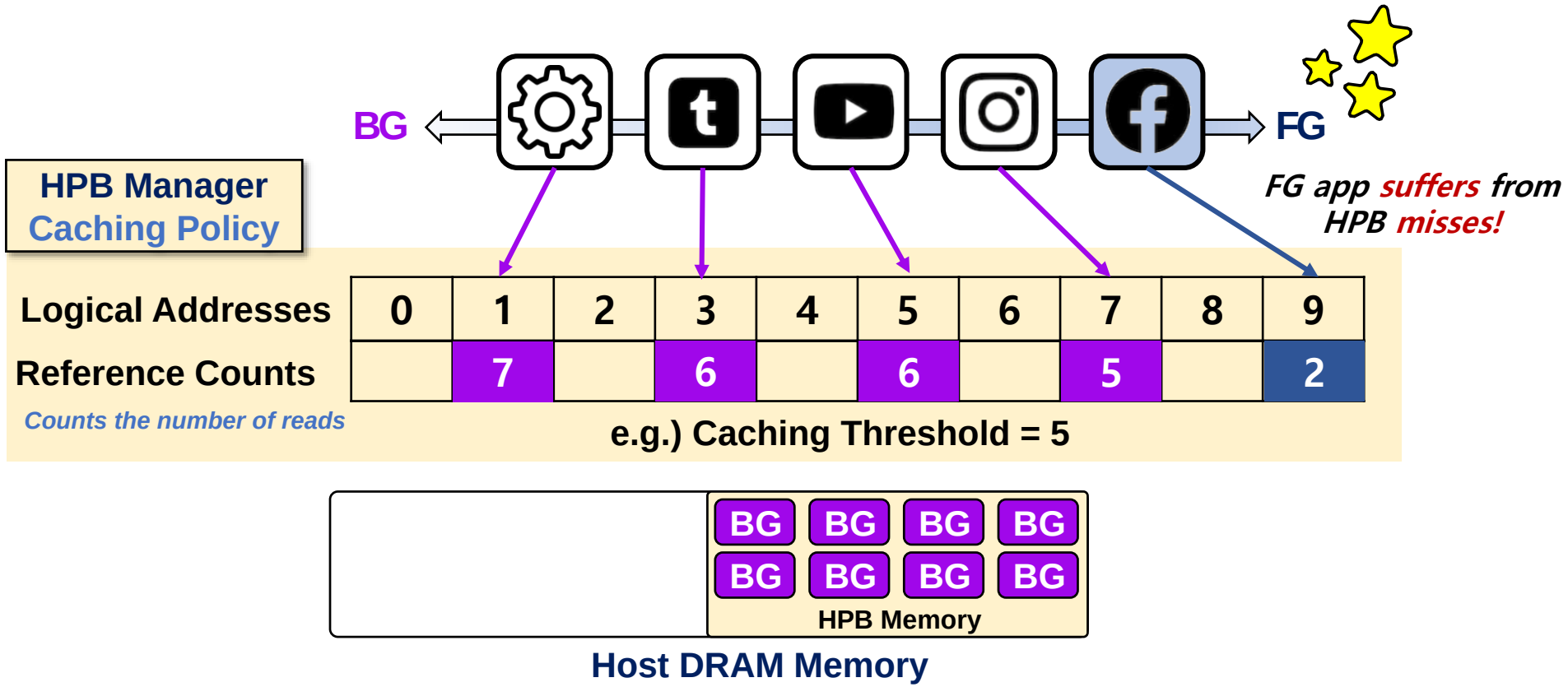
Root Cause 1: App Priority-Oblivious HPB Caching

- Current HPB caching policy results in **priority inversion** between *foreground (FG)* and *background (BG) apps*.



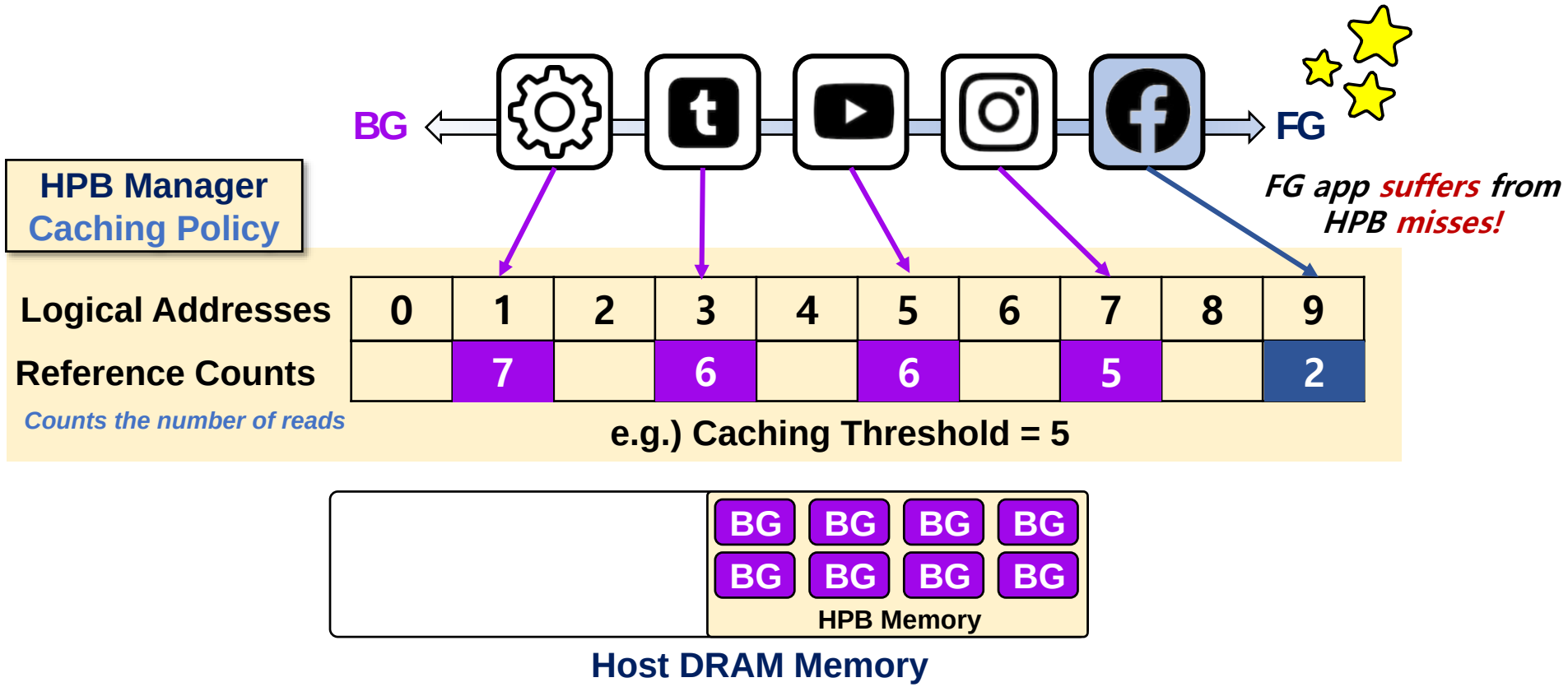
Root Cause 1: App Priority-Oblivious HPB Caching

- Current HPB caching policy results in **priority inversion** between *foreground (FG)* and *background (BG) apps*.



Root Cause 1: App Priority-Oblivious HPB Caching

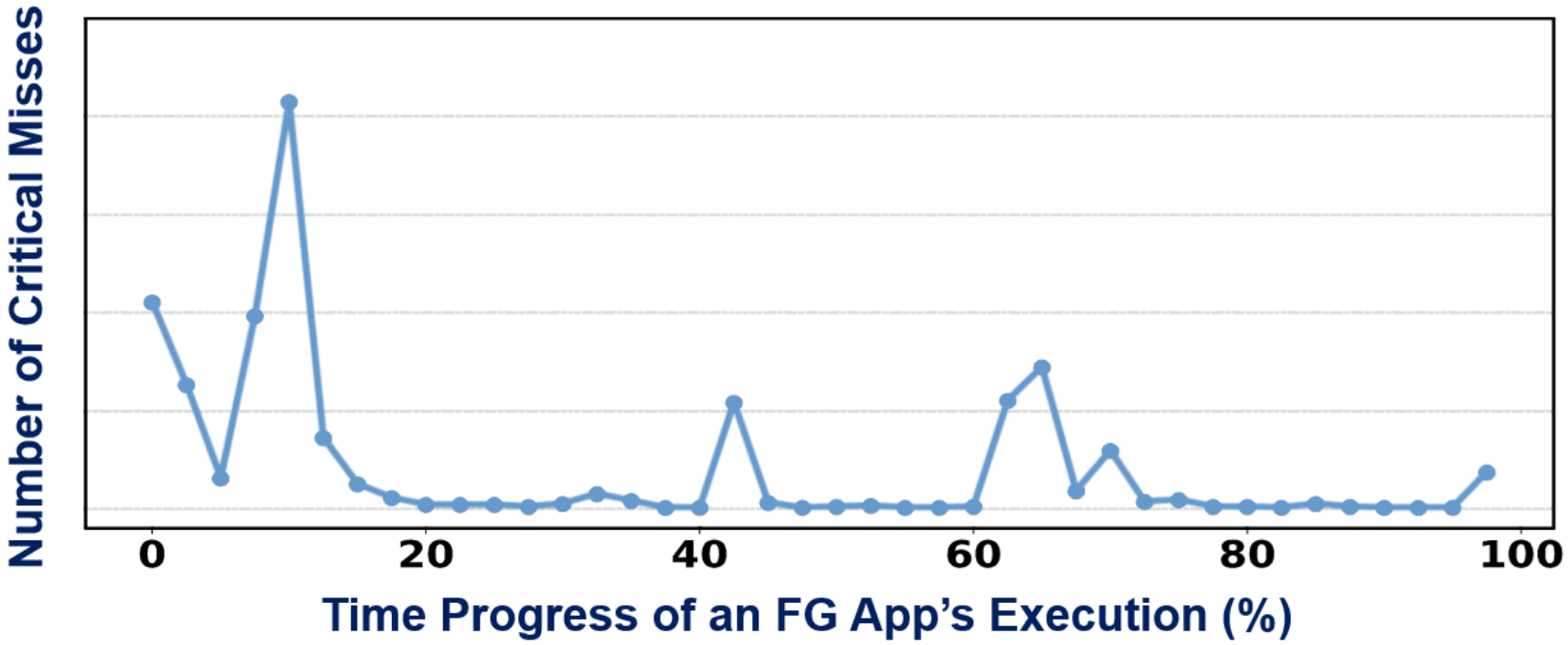
- Current HPB caching policy results in **priority inversion** between *foreground (FG)* and *background (BG) apps*.



HPB fails to cache L2P entries of FG apps.

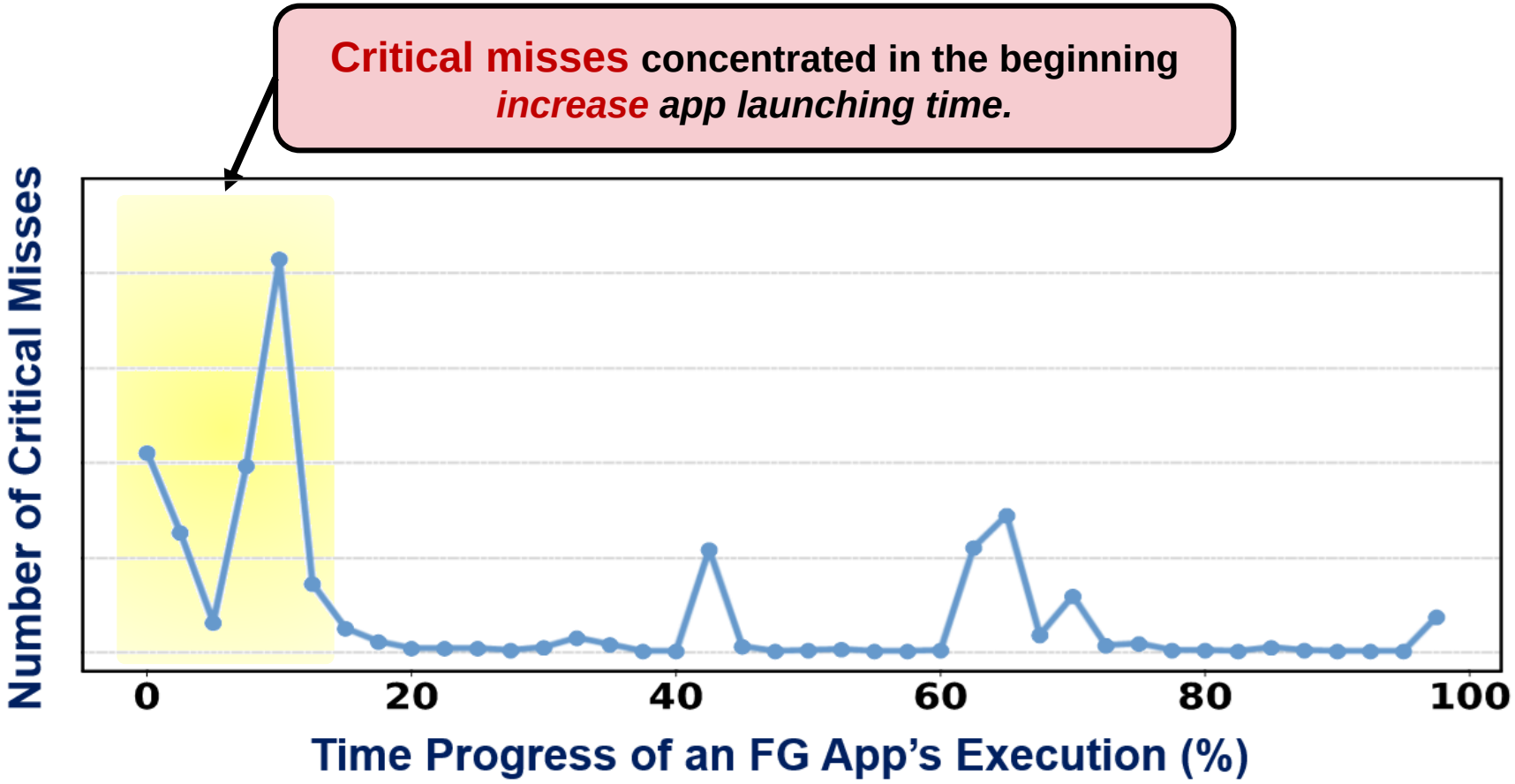
Impact of HPB Caching on App Launching Time

- HPB fails to cache ***UX-critical L2P entries*** in the beginning of FG app's execution due to *priority inversion* between FG and BG apps.



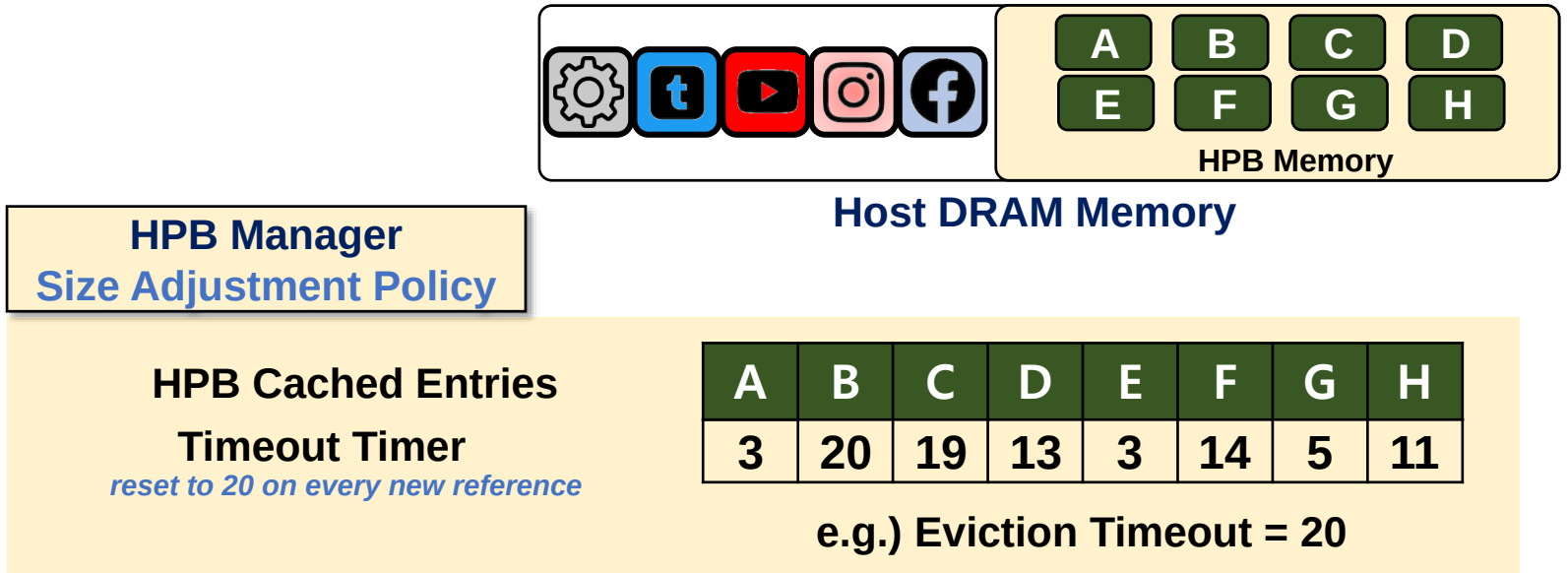
Impact of HPB Caching on App Launching Time

- HPB fails to cache ***UX-critical L2P entries*** in the beginning of FG app's execution due to *priority inversion* between FG and BG apps.



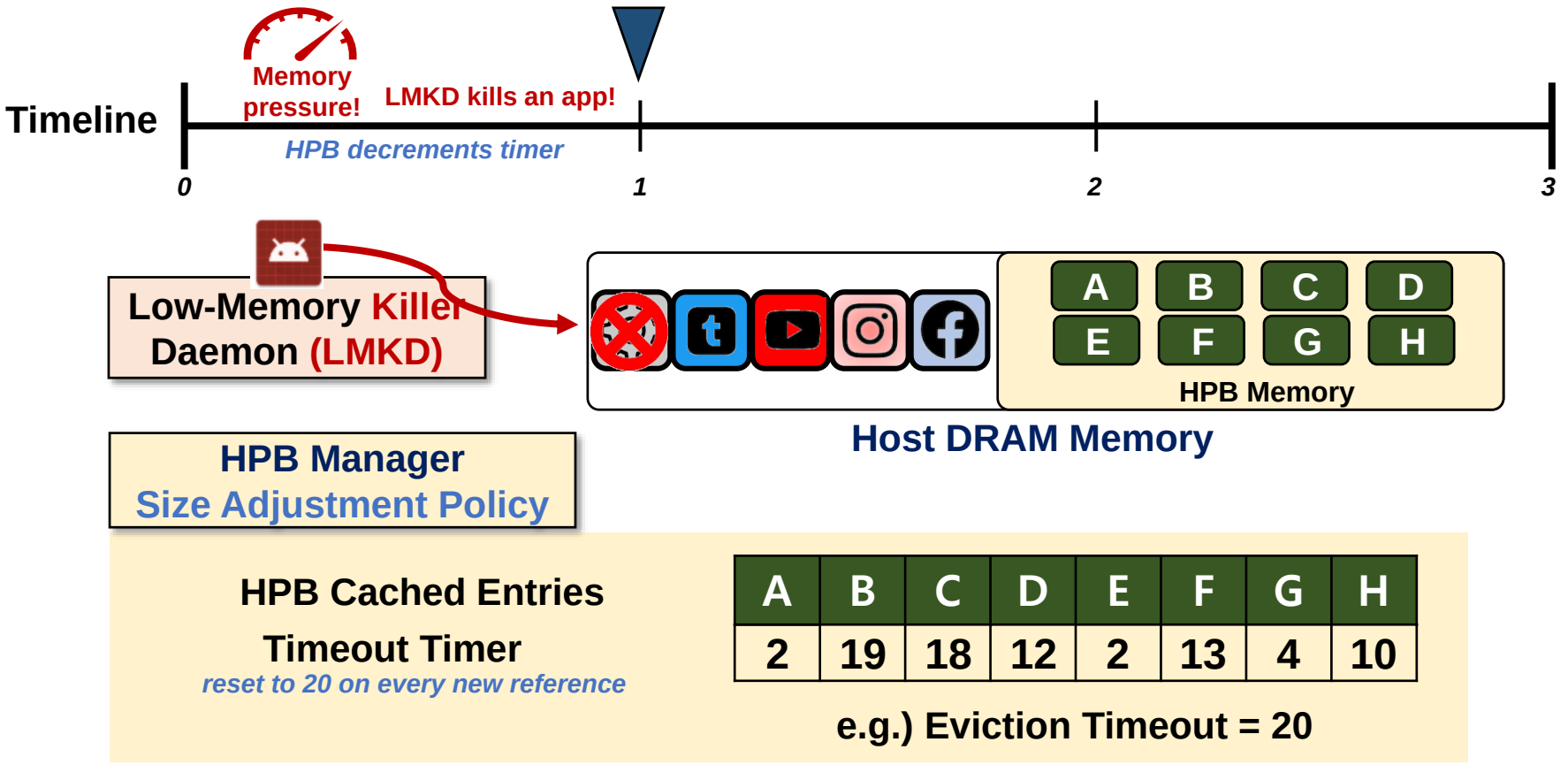
Root Cause 2: Mem. Pressure-unaware Size Adjustment

- Current HPB size adjustment policy does not address **memory pressure status**.



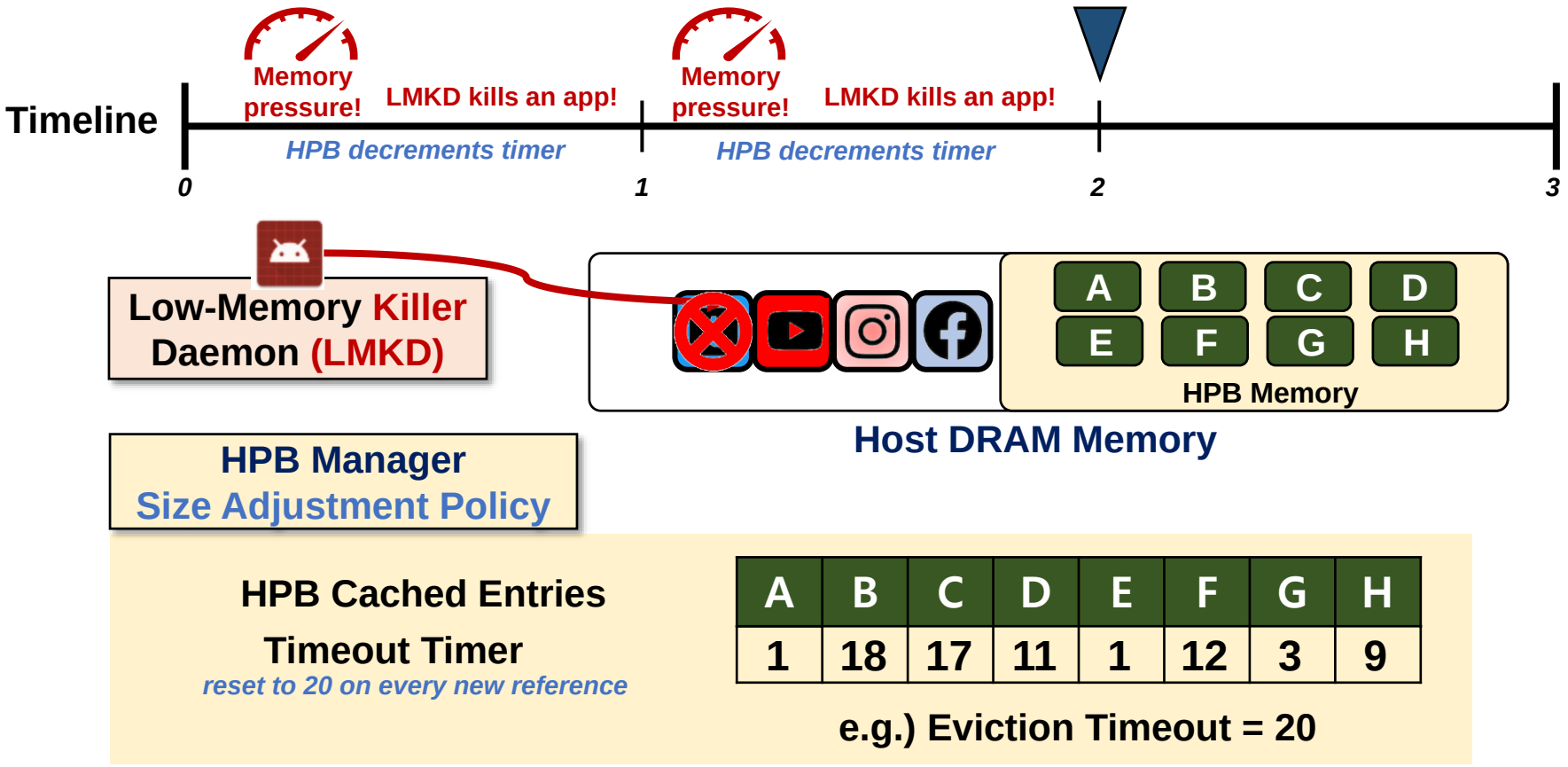
Root Cause 2: Mem. Pressure-unaware Size Adjustment

- Current HPB size adjustment policy does not address **memory pressure status**.



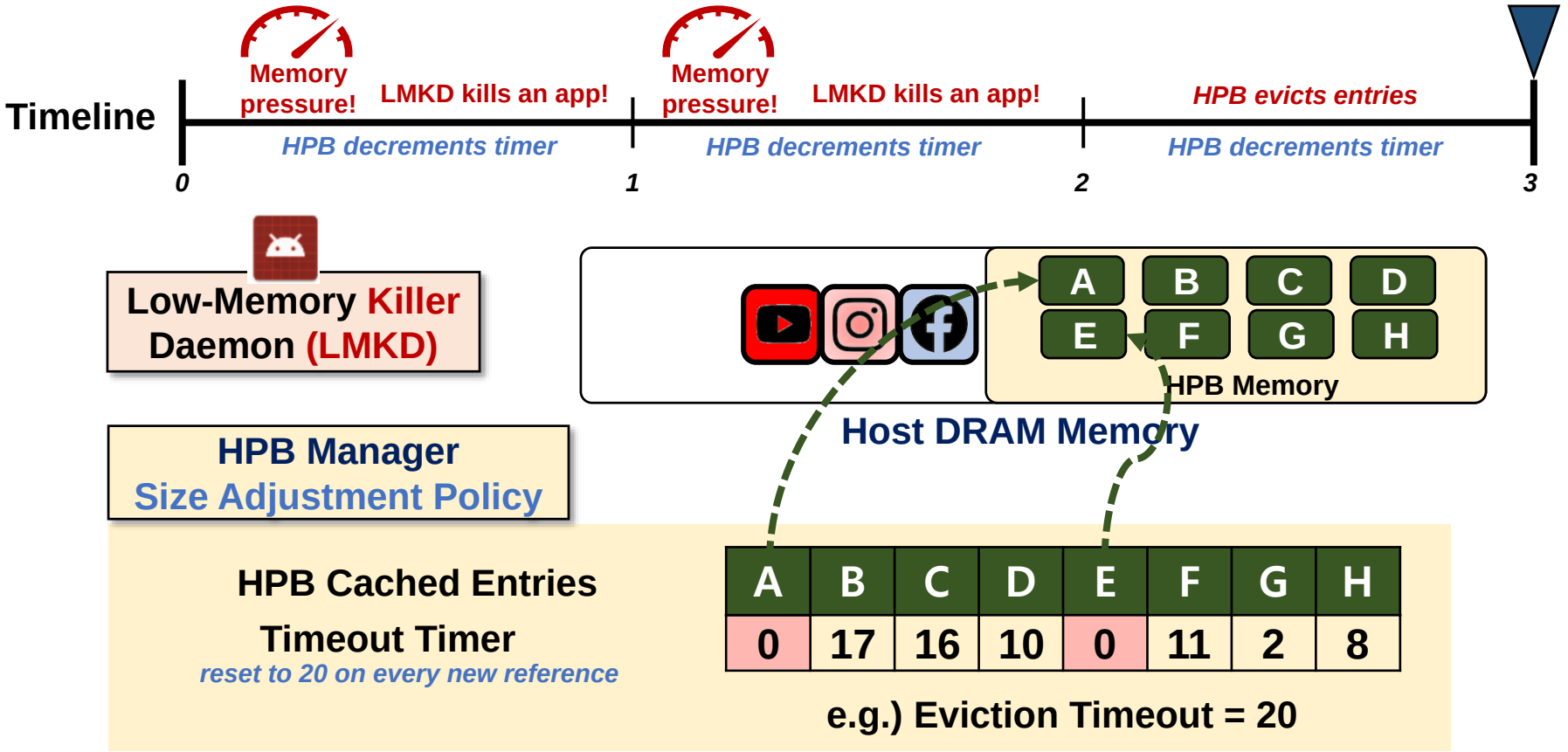
Root Cause 2: Mem. Pressure-unaware Size Adjustment

- Current HPB size adjustment policy does not address **memory pressure status**.



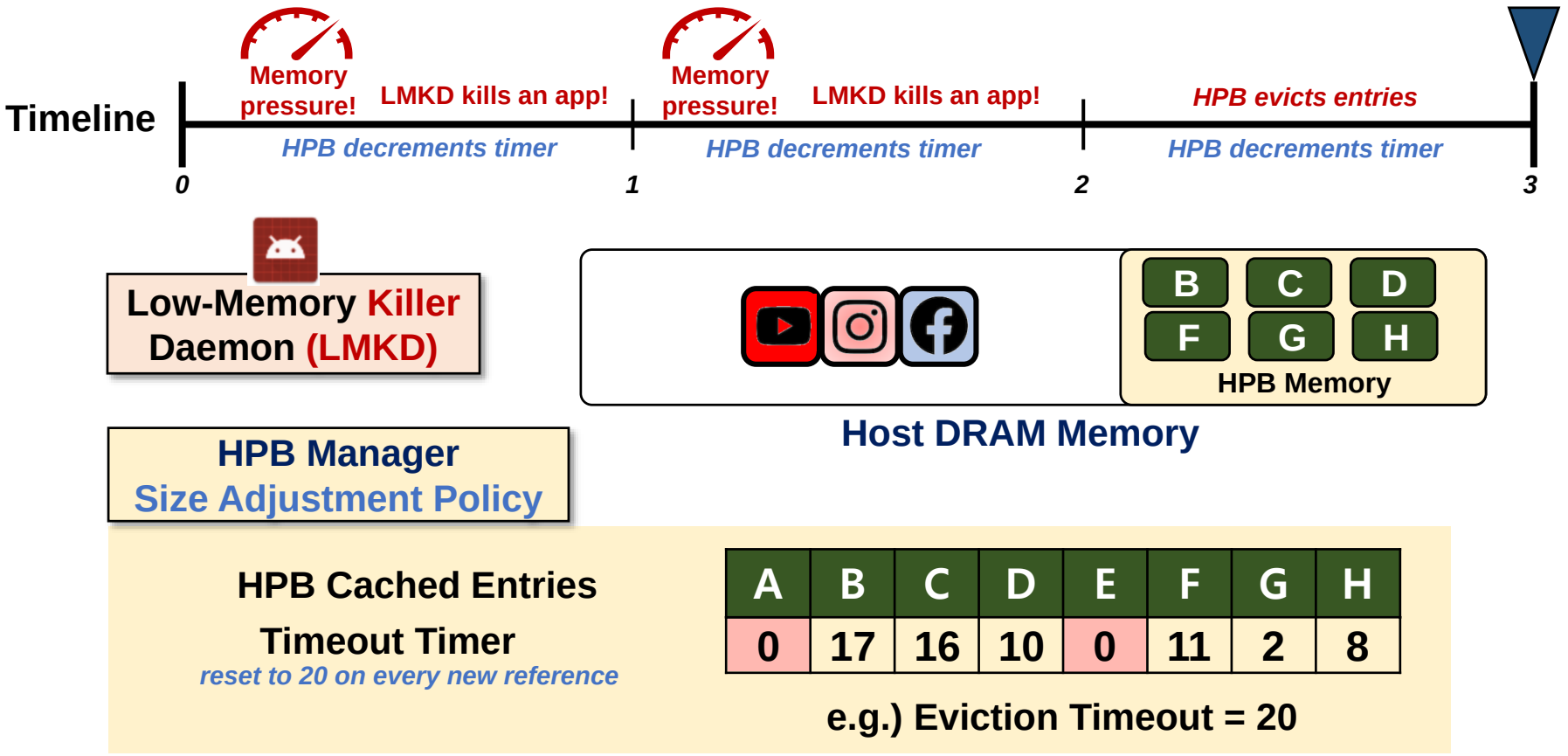
Root Cause 2: Mem. Pressure-unaware Size Adjustment

- Current HPB size adjustment policy does not address **memory pressure status**.



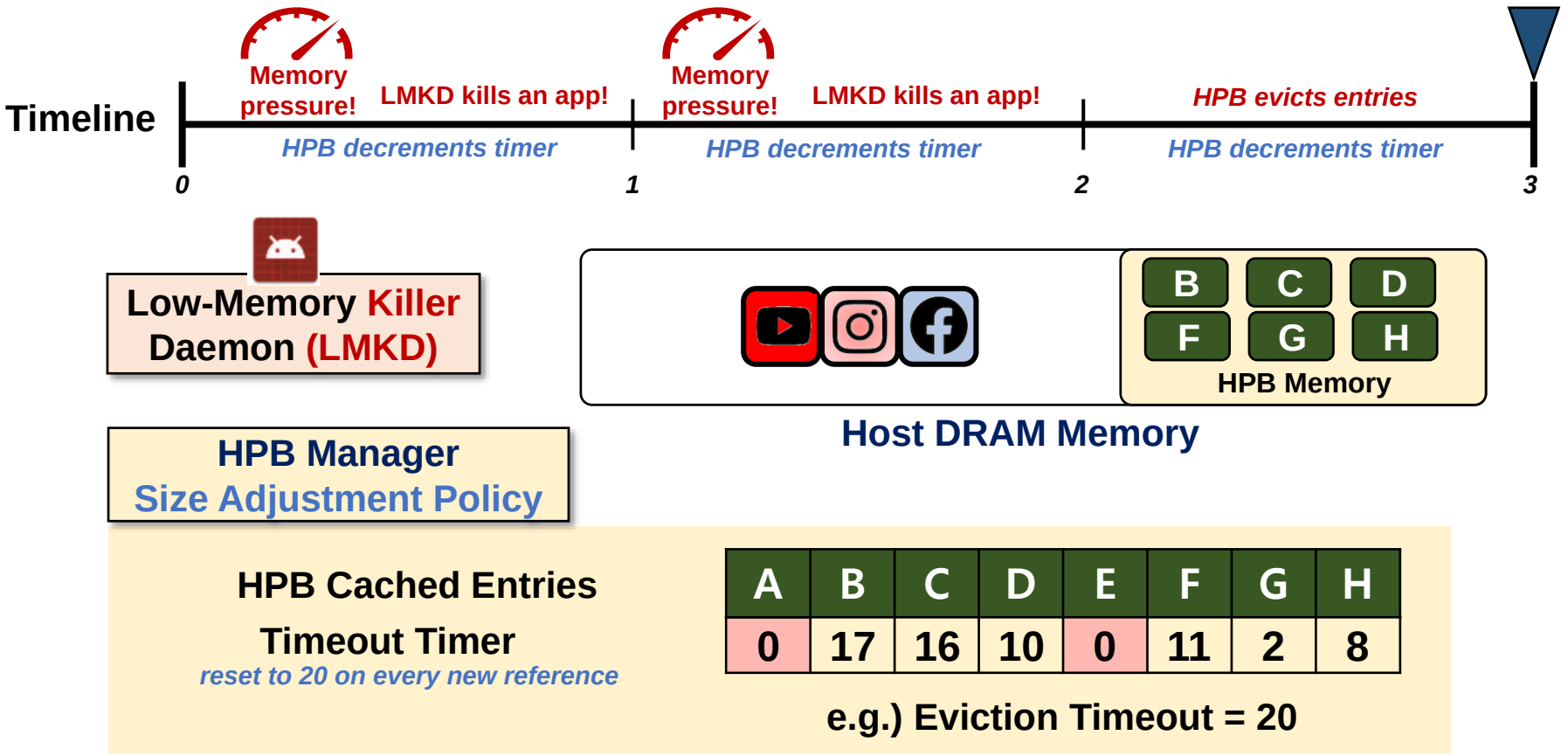
Root Cause 2: Mem. Pressure-unaware Size Adjustment

- Current HPB size adjustment policy does not address **memory pressure status**.



Root Cause 2: Mem. Pressure-unaware Size Adjustment

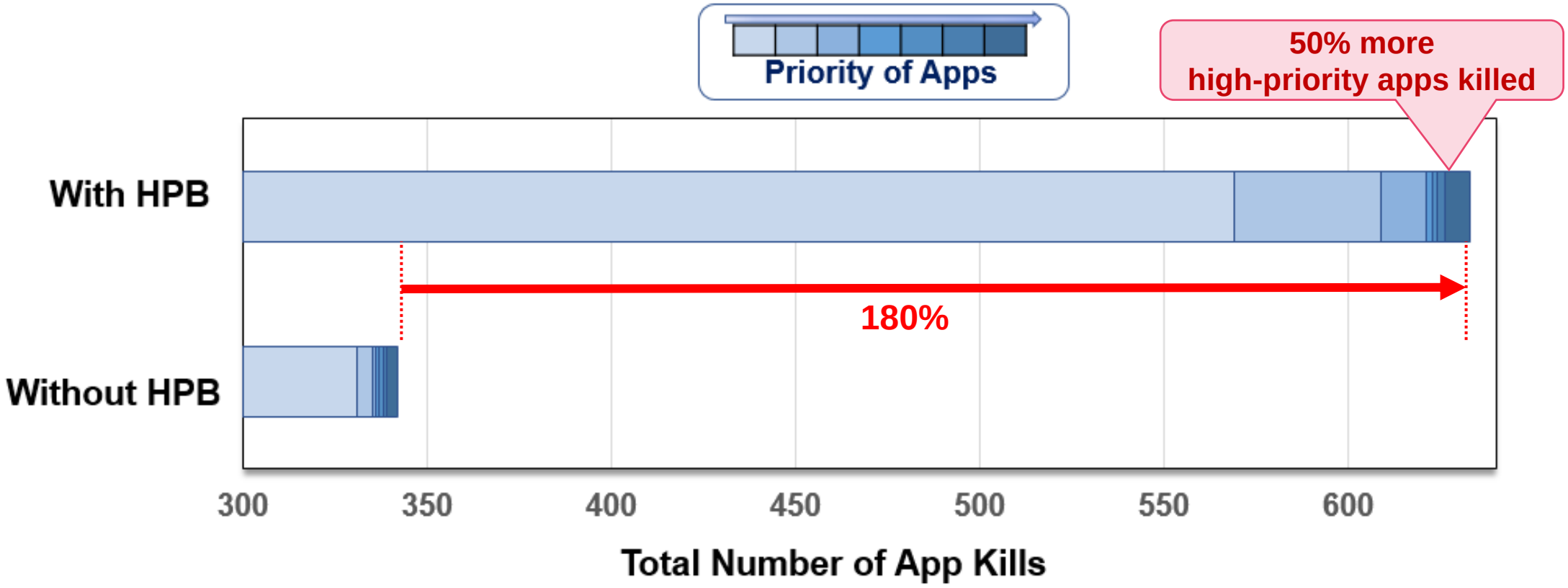
- Current HPB size adjustment policy does not address **memory pressure status**.



HPB increased memory pressure causes unintentional app kills.

Impact of HPB Memory Consumption on App Killing

- Some of high-priority (recently user-interacted) apps are unintentionally killed due to HPB induced *memory pressure*.
  - Increased number of high-priority app kills result *in longer user-perceived latency for future re-launches.*

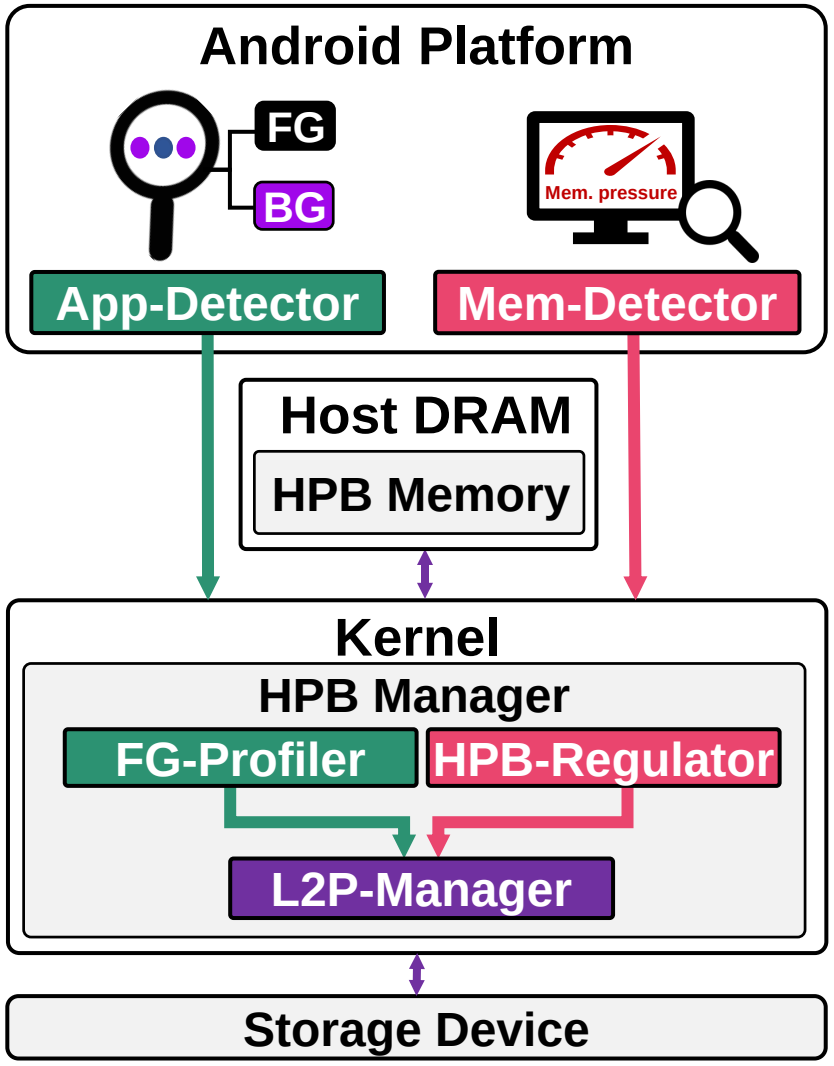


Outline

- Root Cause Analysis of the Existing Solution
- Design of HPBvalve
- Evaluation Results

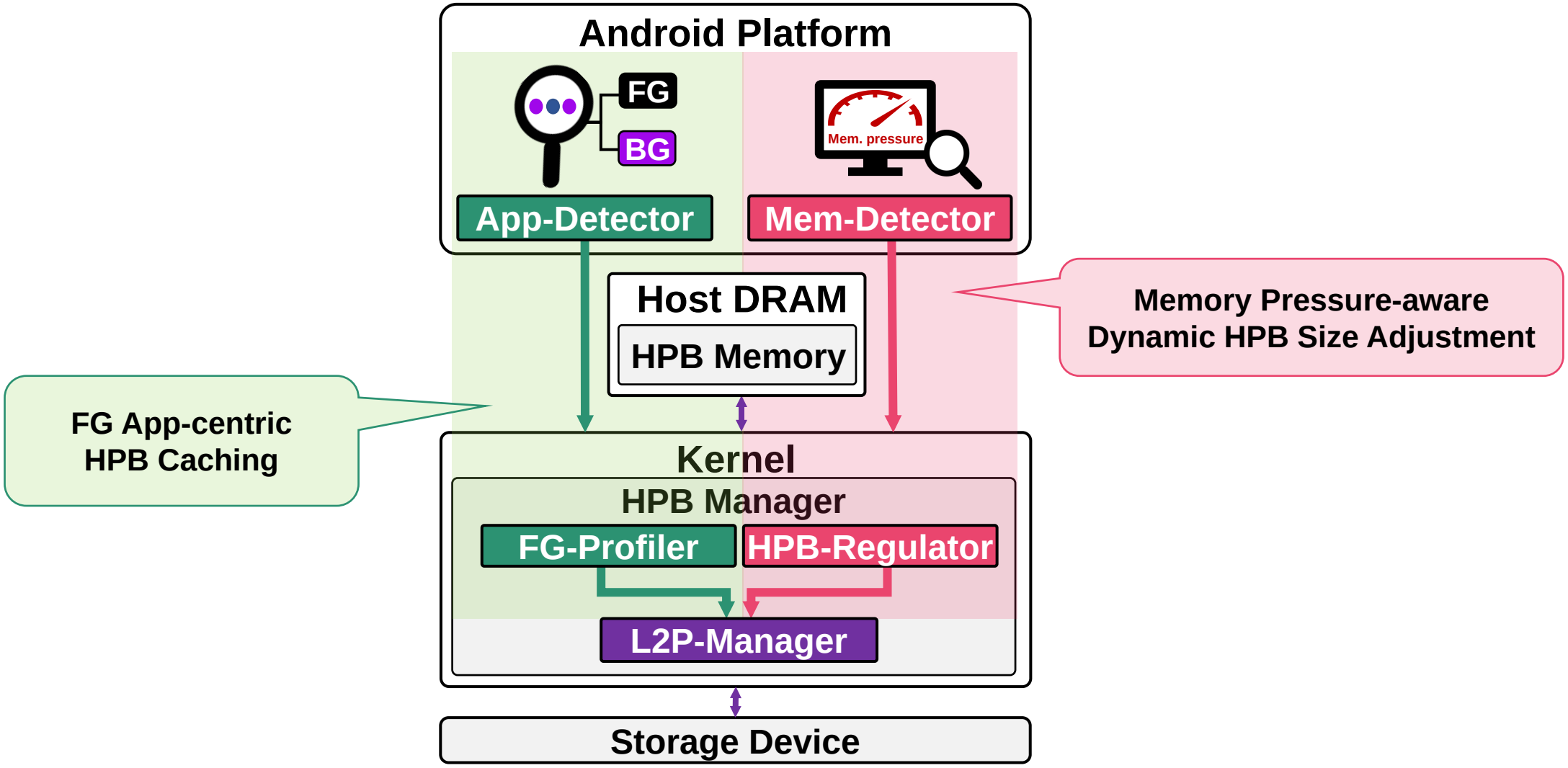
Overall Architecture of HPBvalve

- Manage HPB memory in a **UX-Centric** Manner.



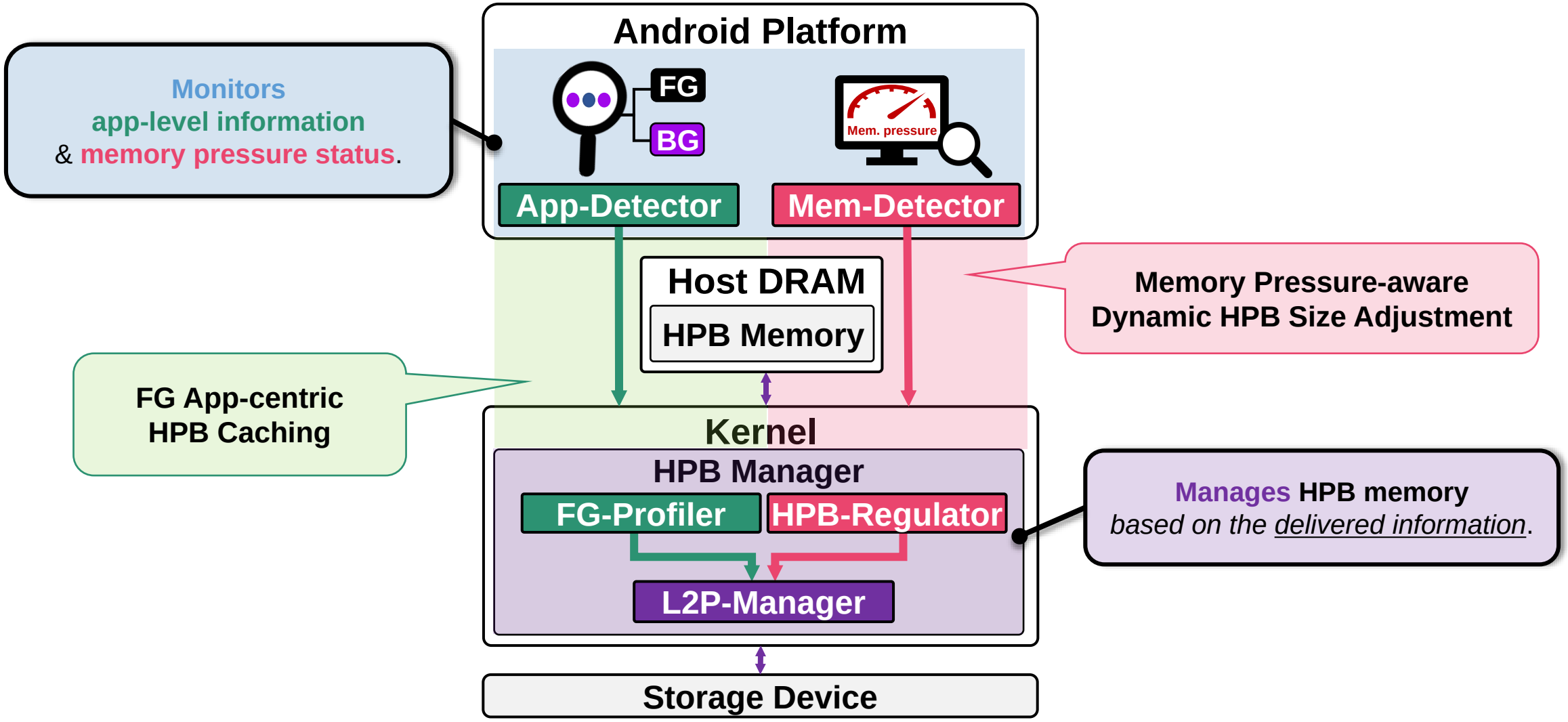
Overall Architecture of HPBvalve

- Manage HPB memory in a **UX-Centric** Manner.



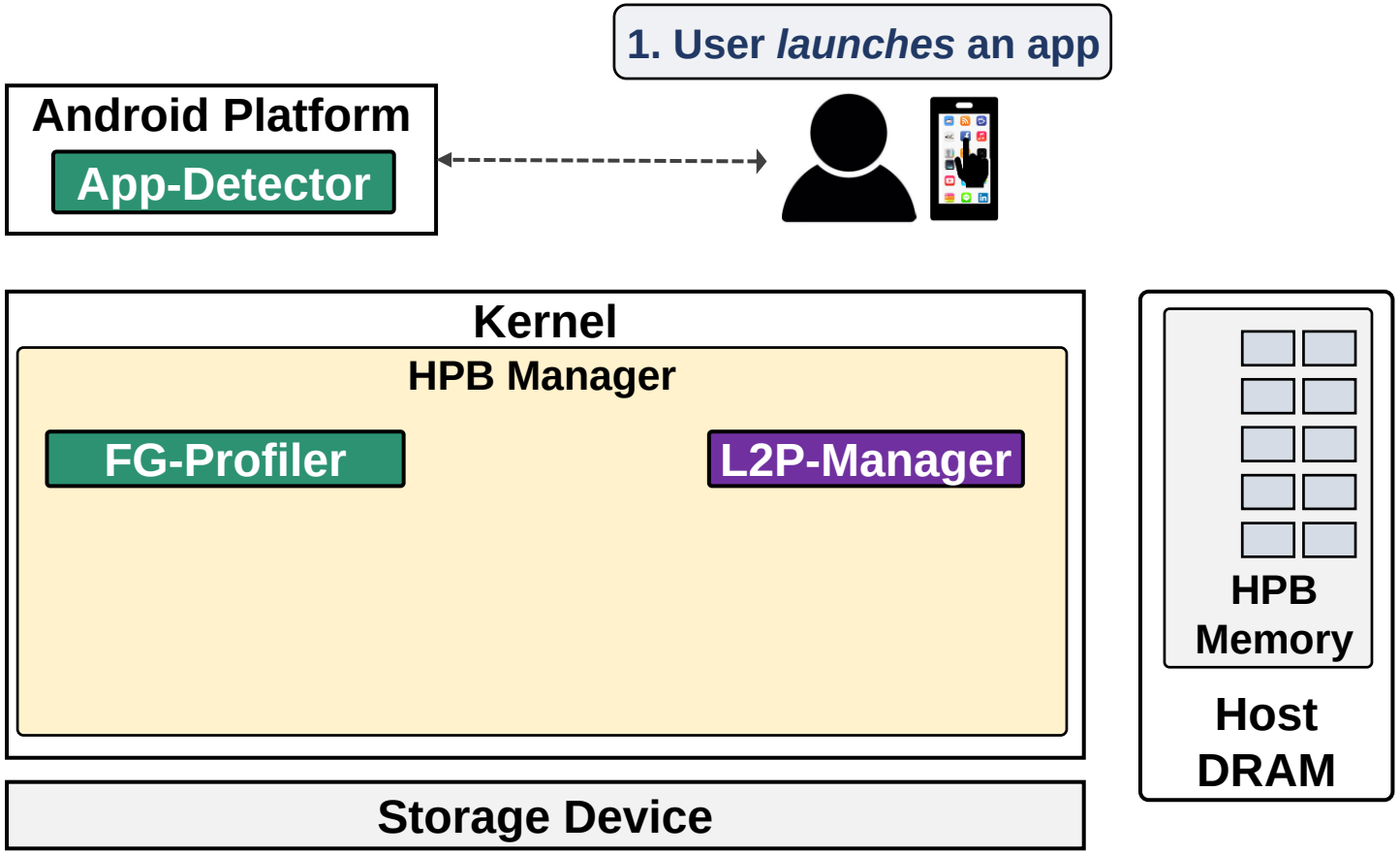
Overall Architecture of HPBvalve

- Manage HPB memory in a **UX-Centric** Manner.



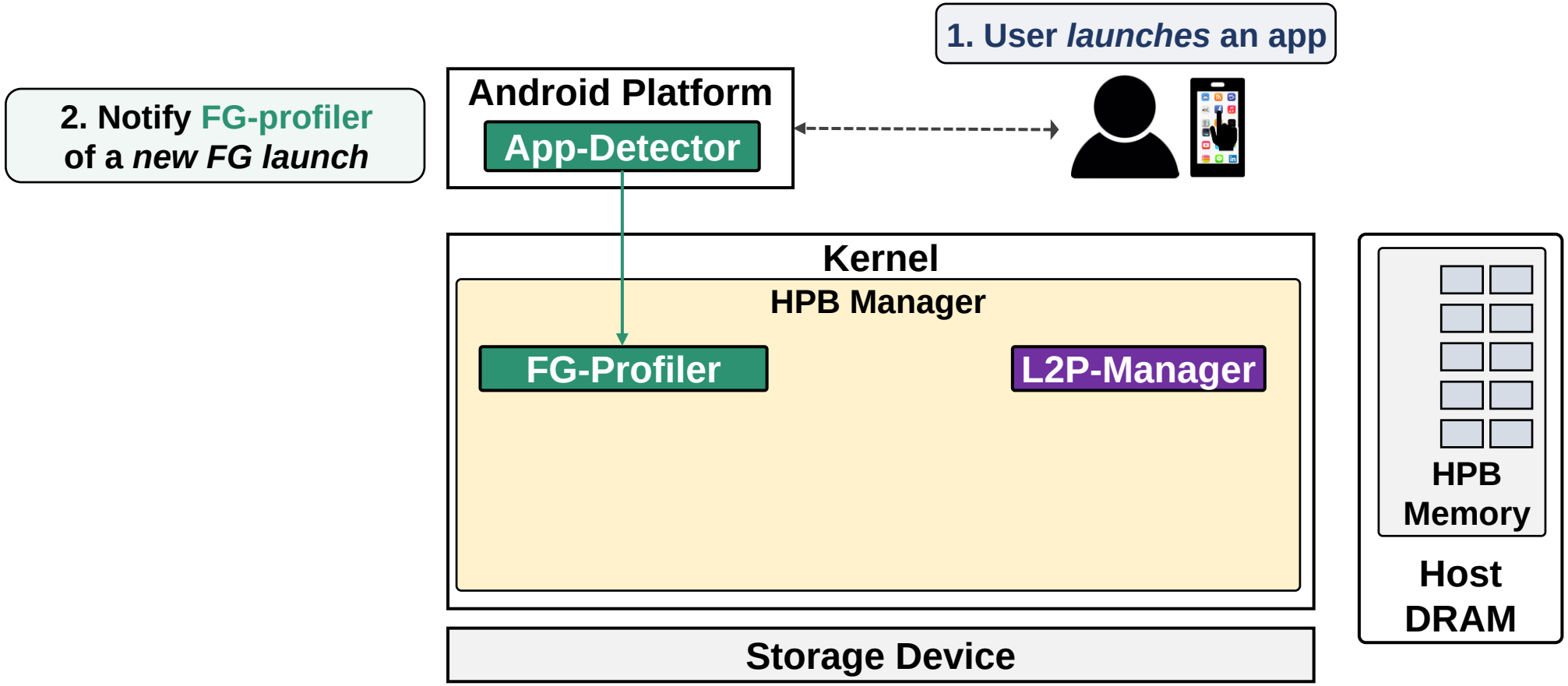
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



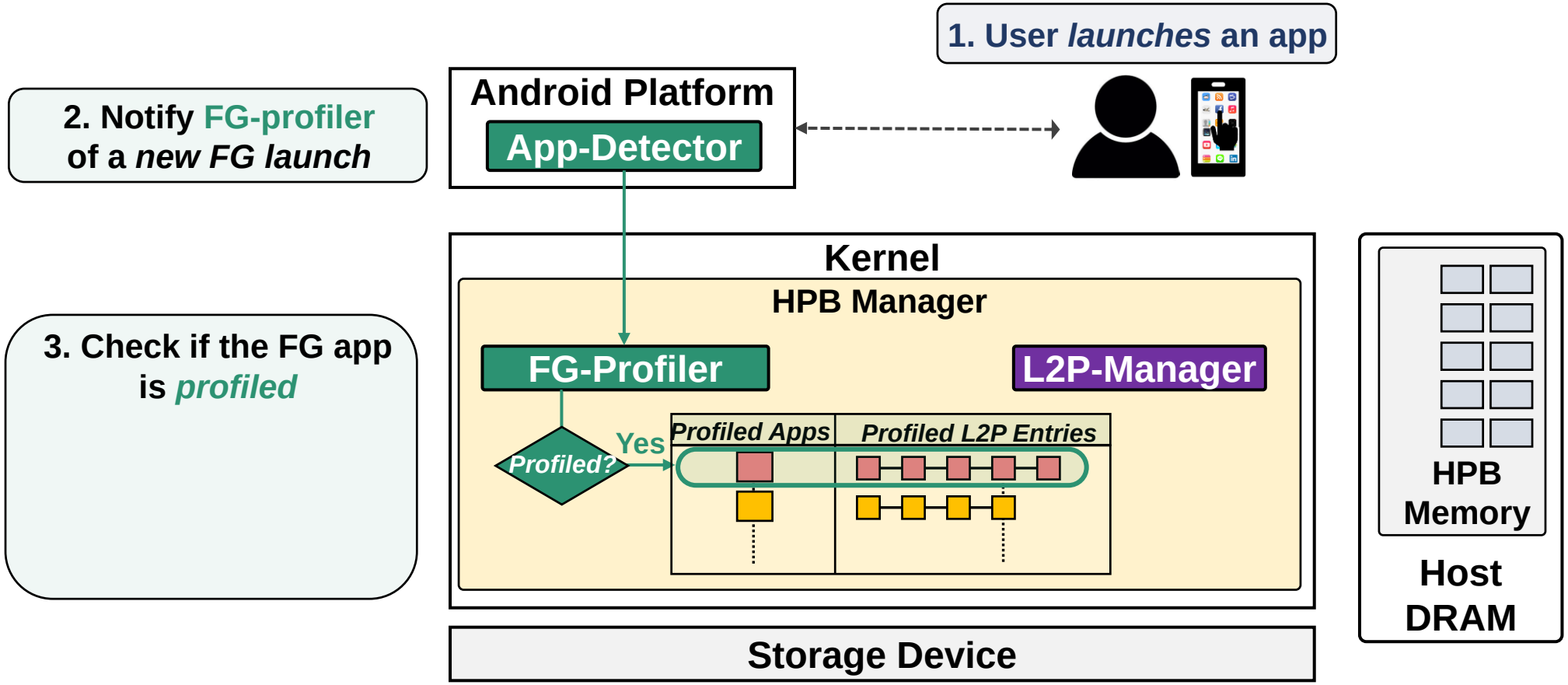
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



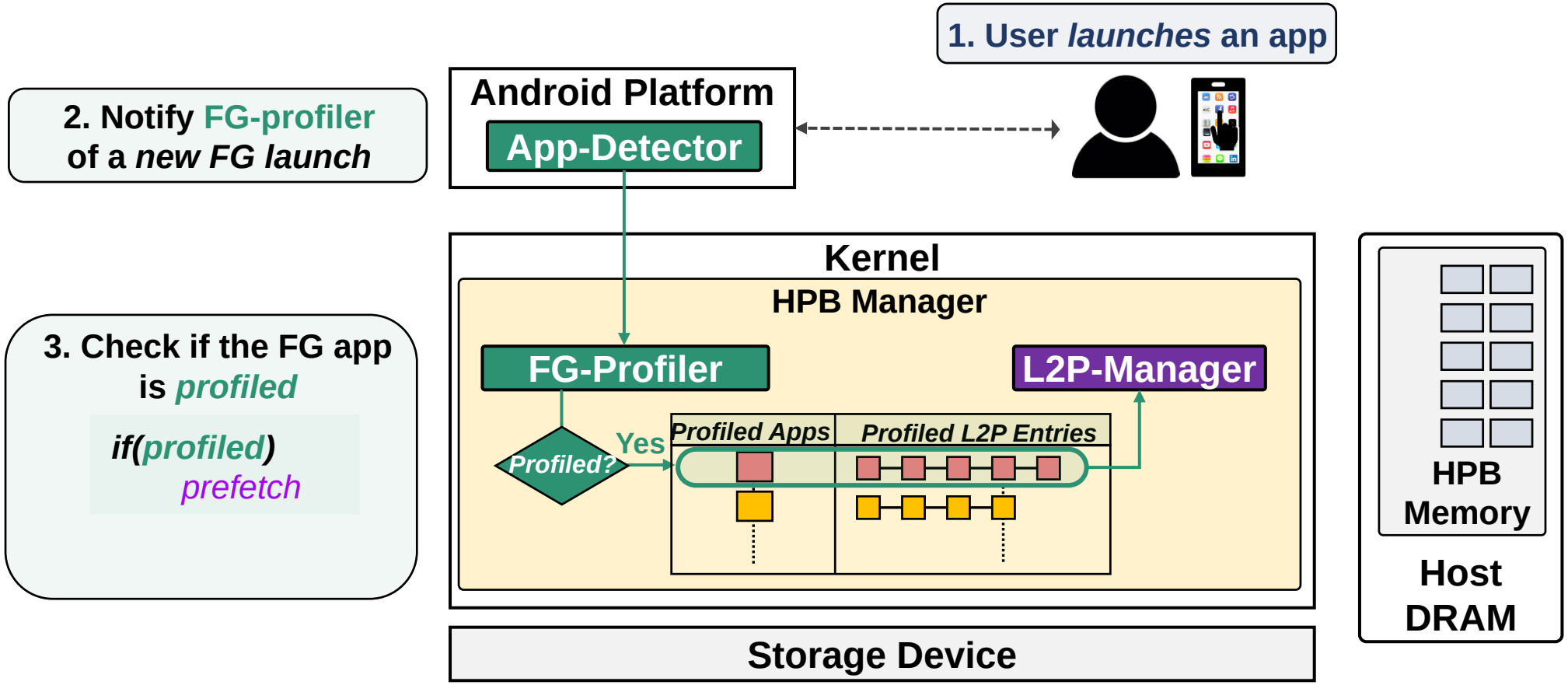
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



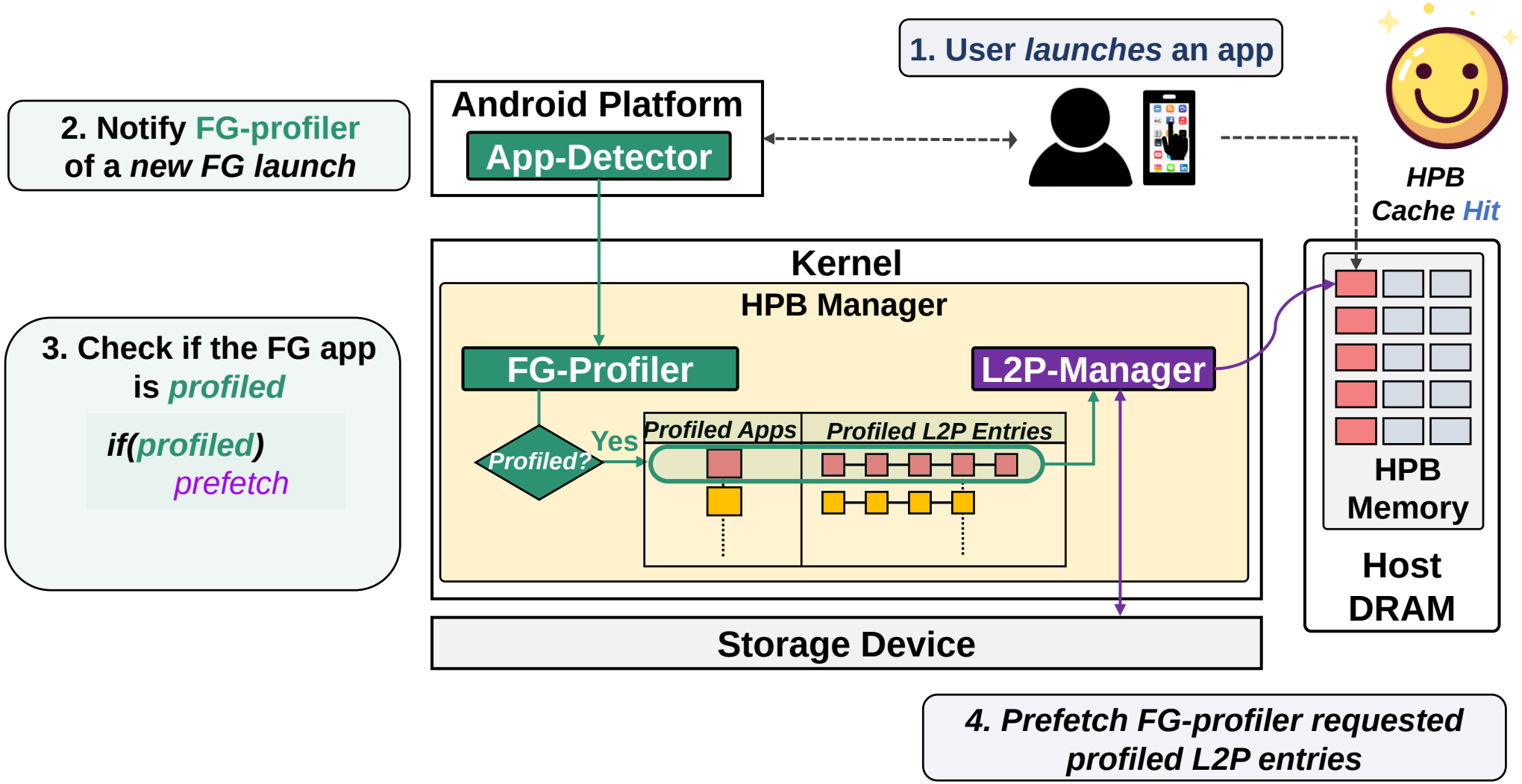
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



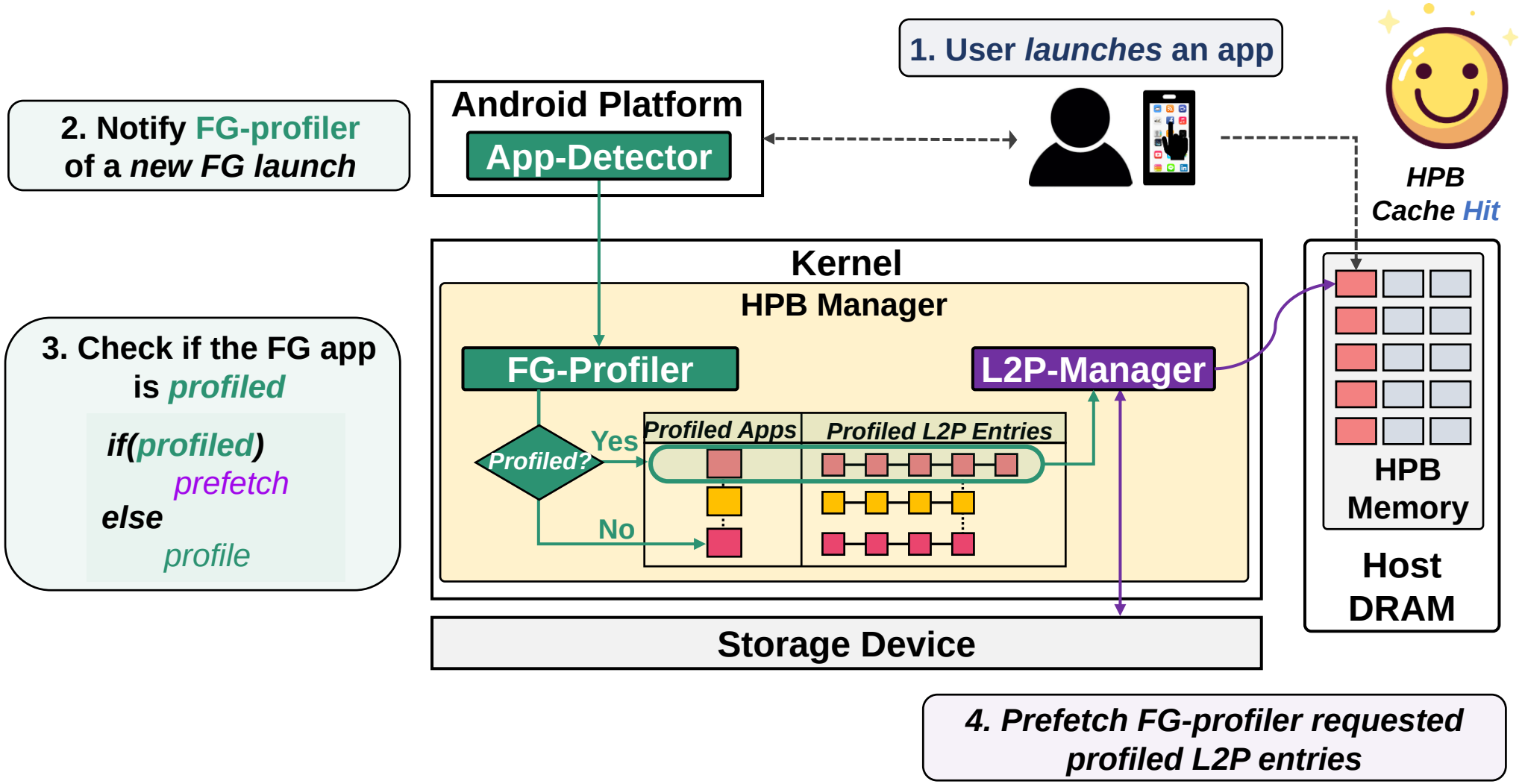
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



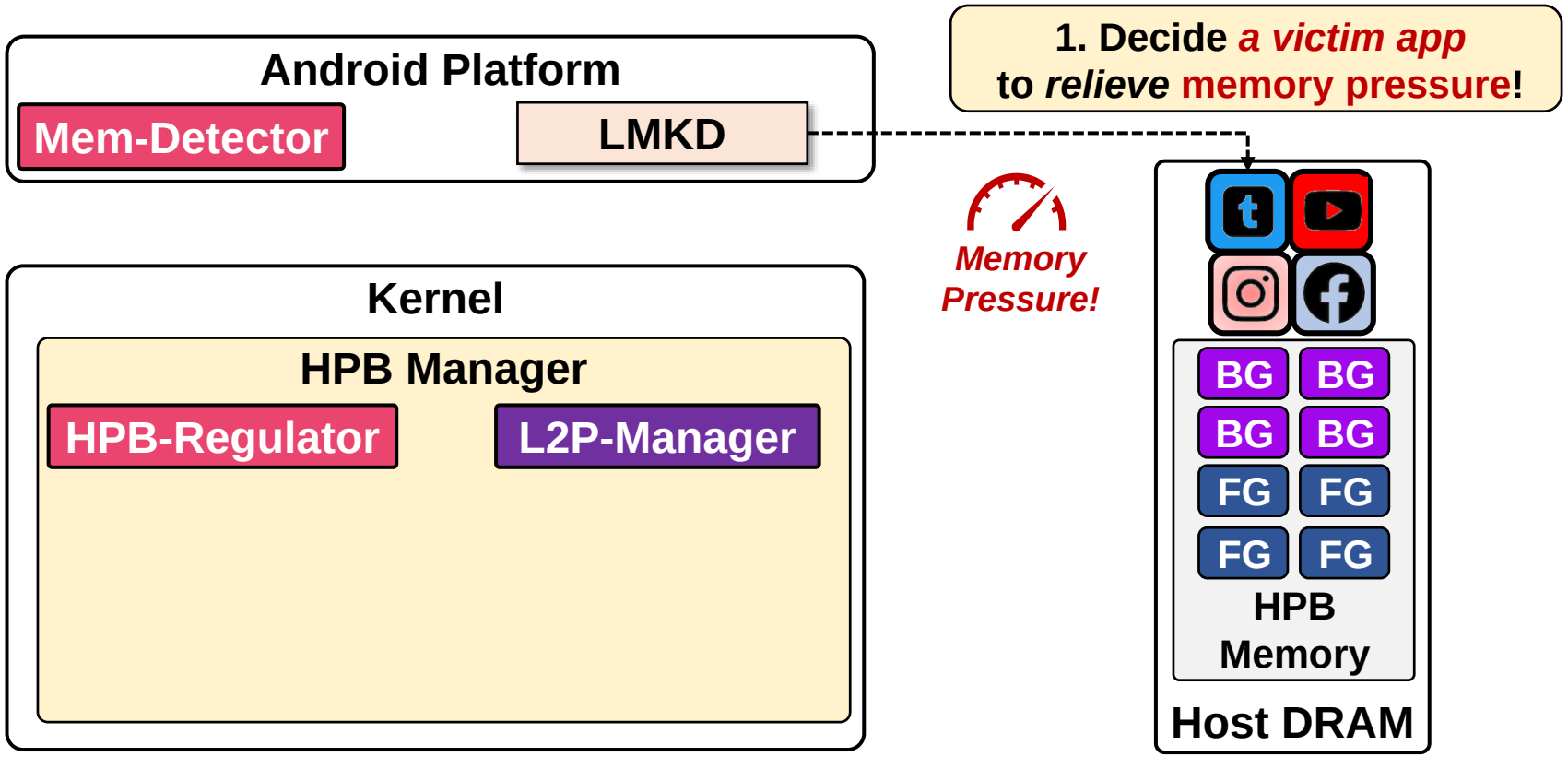
FG App-centric HPB Caching

- **Key Idea:** Prefetch important L2P entries of popular FG apps on their launch.



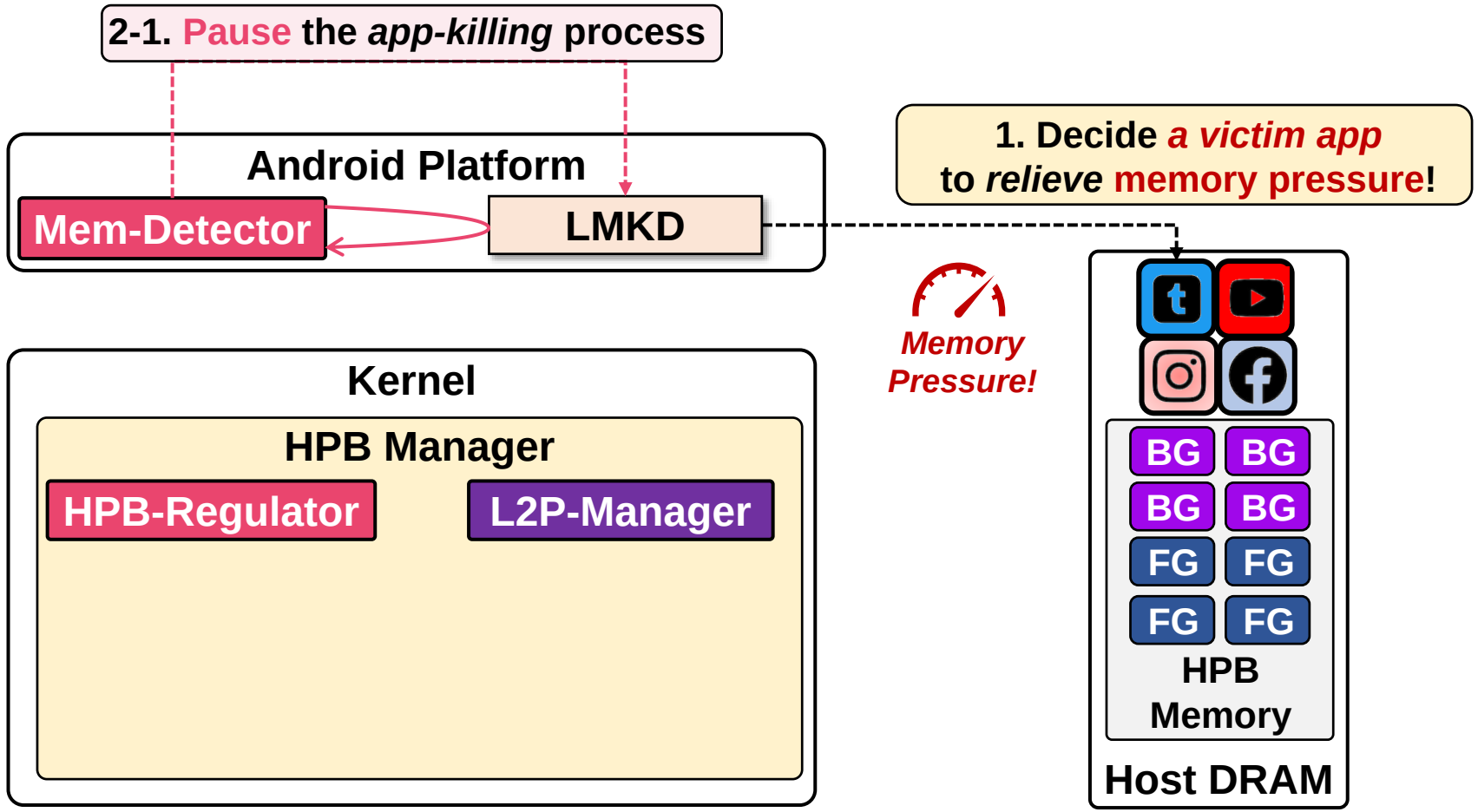
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



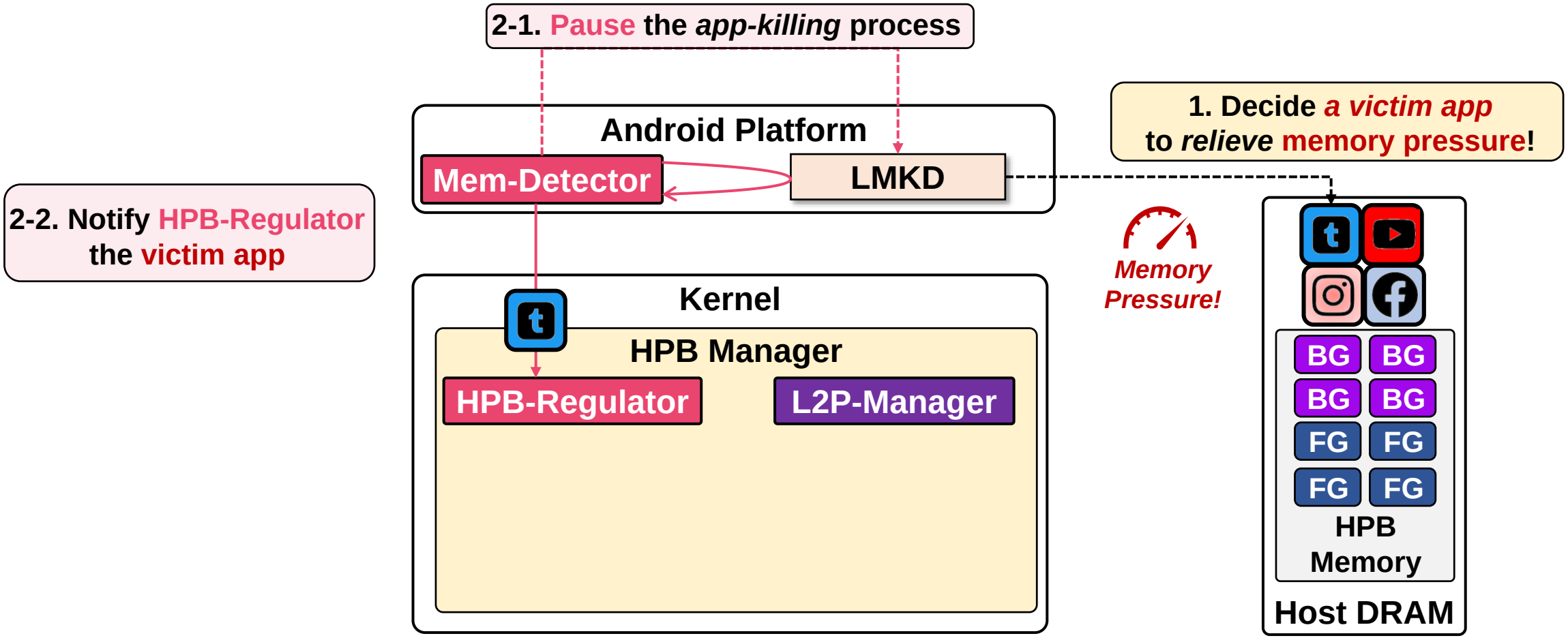
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



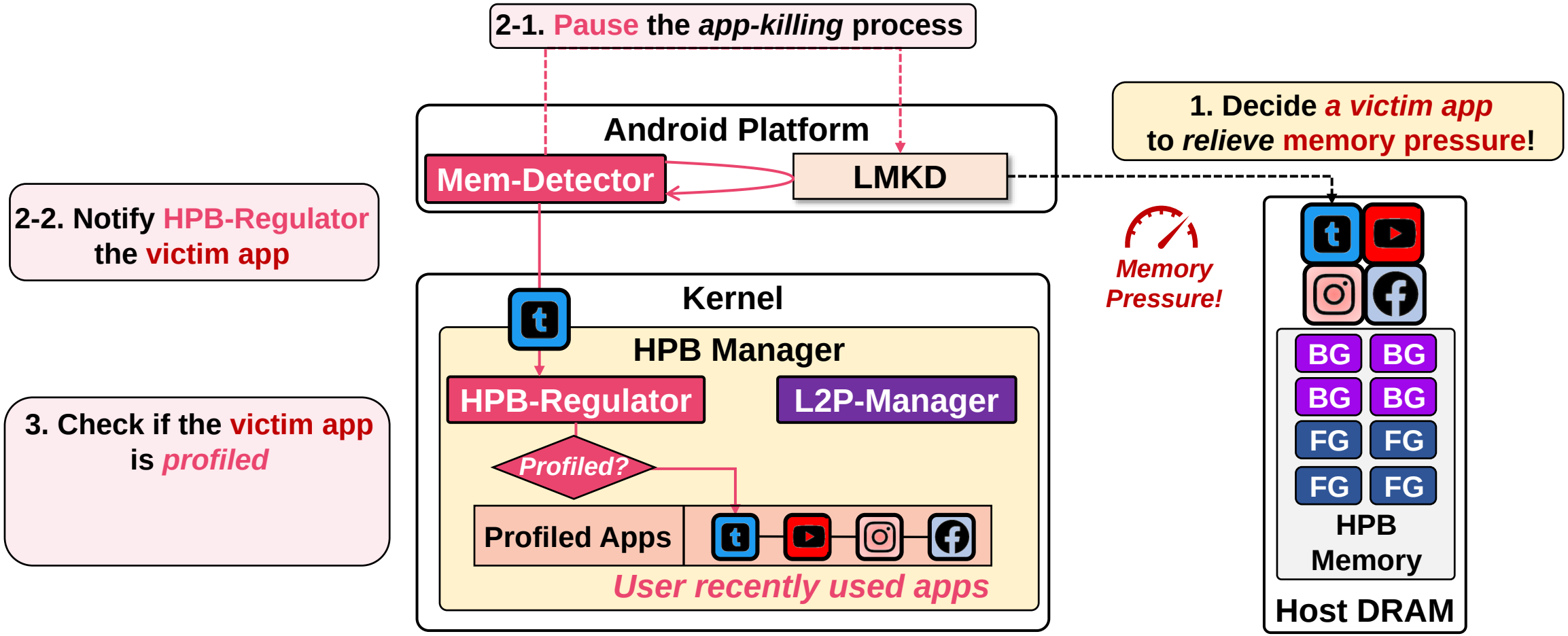
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



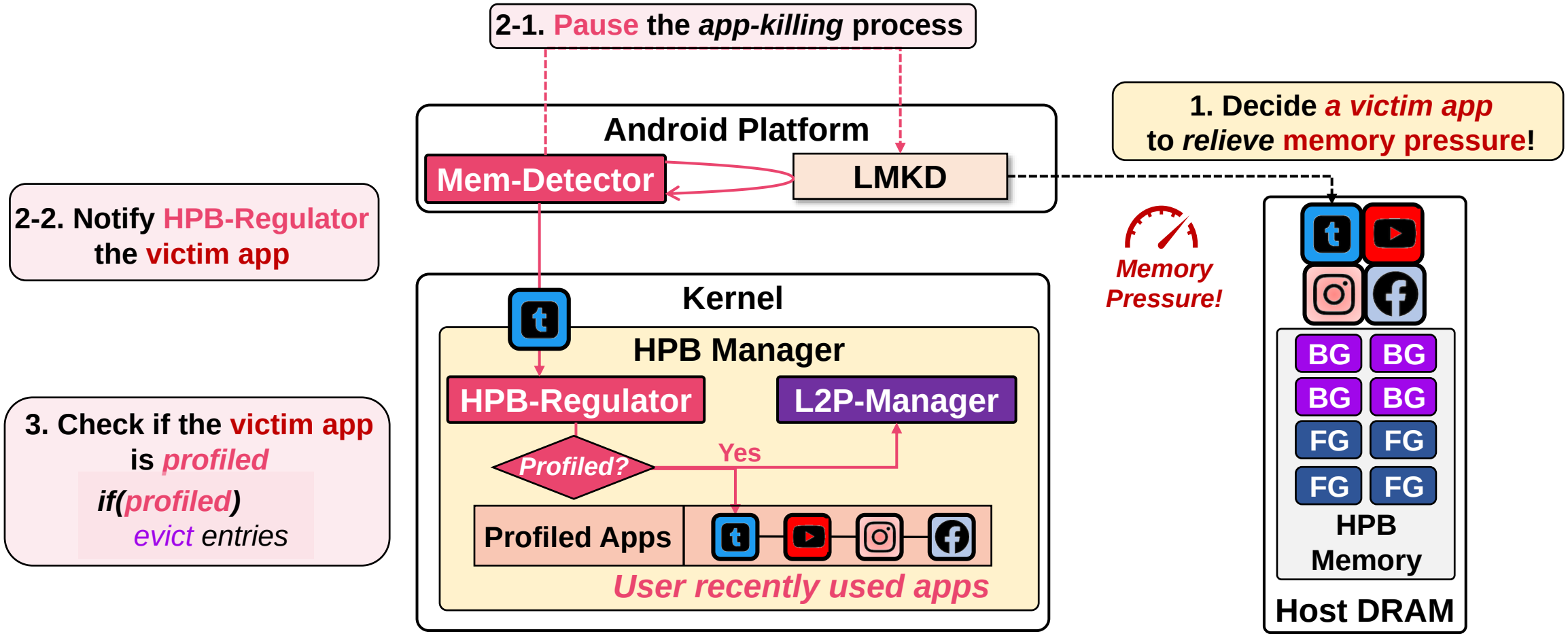
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



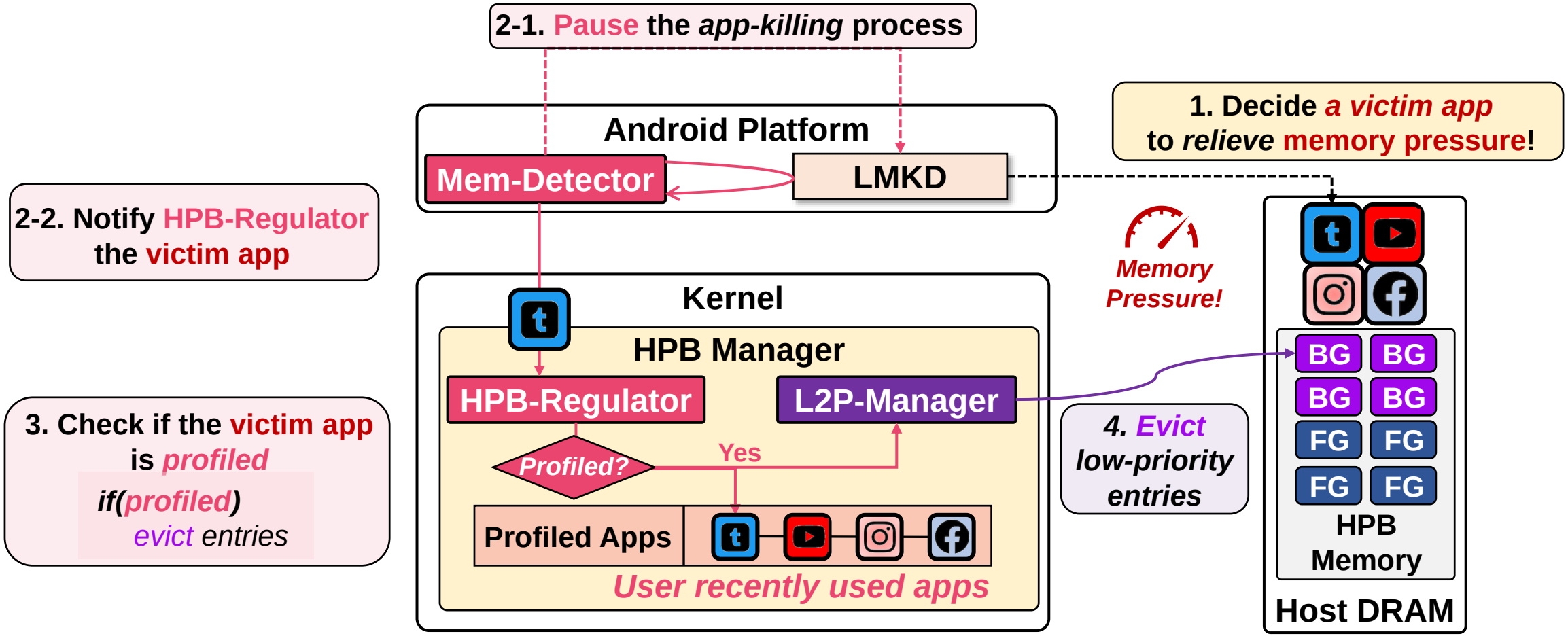
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



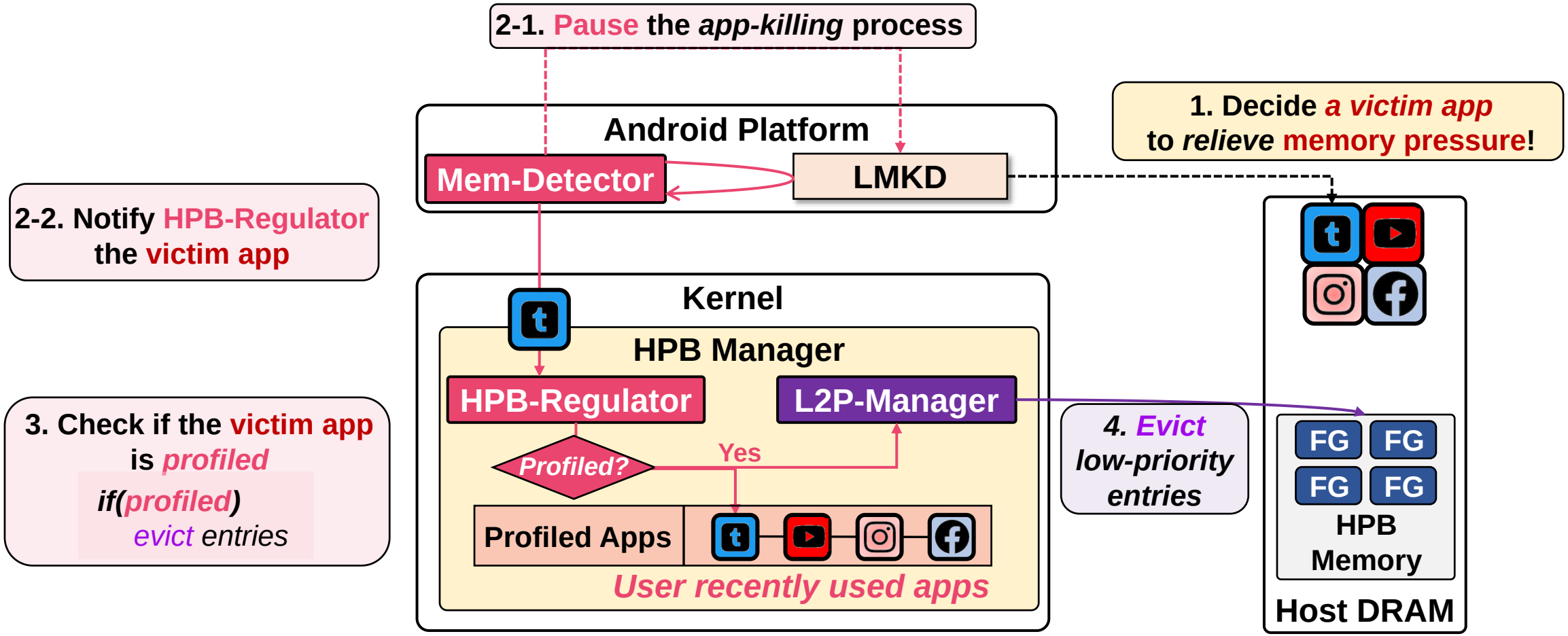
Mem. Pressure-aware Dynamic HPB Size Adjustment

- **Key Idea**: Return HPB memory before a high-priority app gets killed.



Mem. Pressure-aware Dynamic HPB Size Adjustment

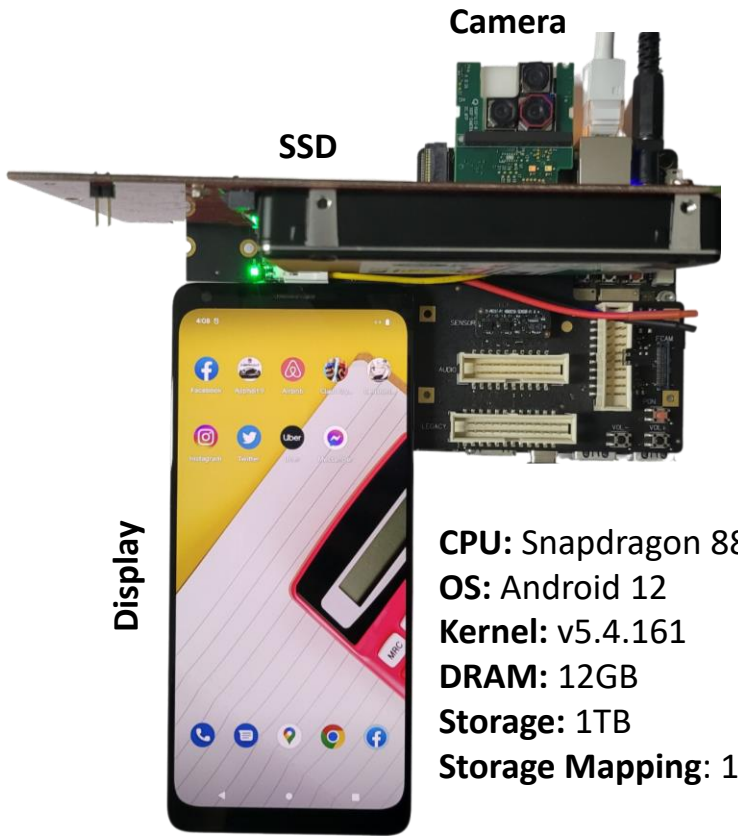
- **Key Idea**: Return HPB memory before a high-priority app gets killed.



Evaluation Scenarios and Settings

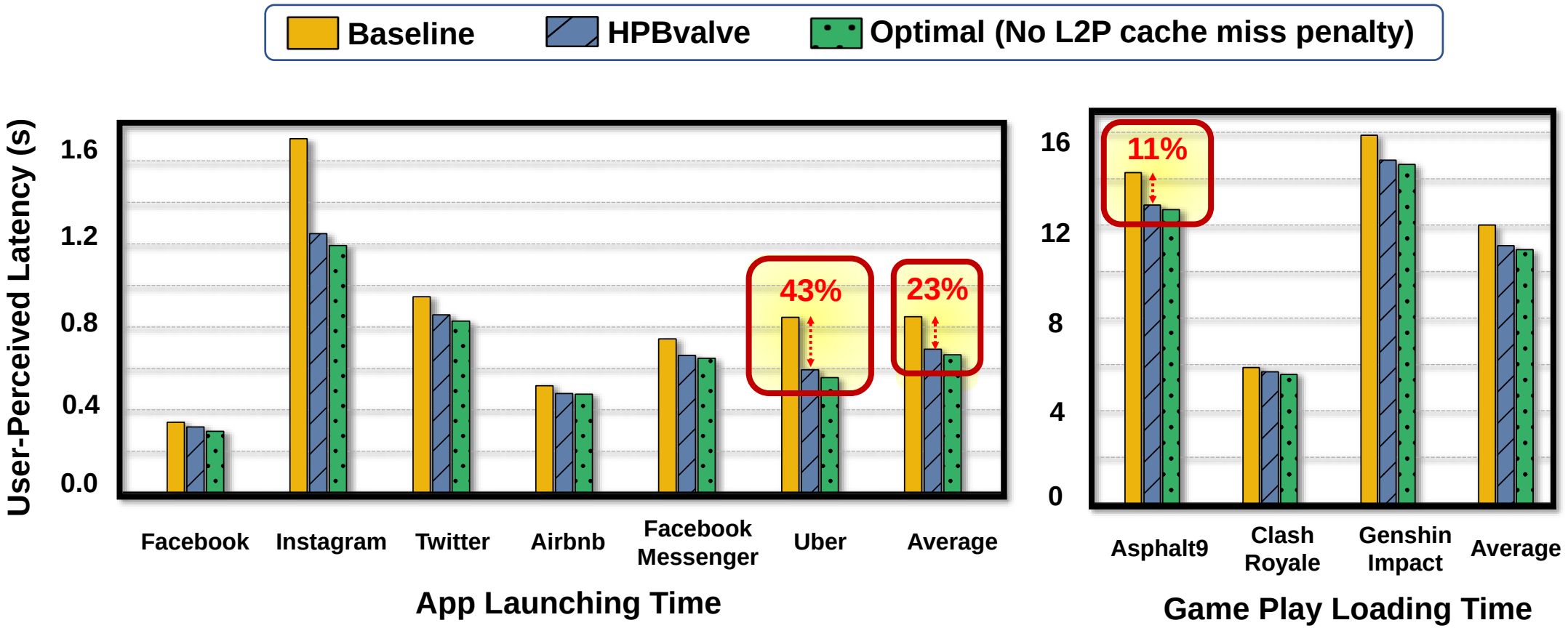
- We implemented HPBvalve on a mobile hardware development kit.
- We evaluated **nine popular apps** under predefined scenarios.

Category	Apps	App Usage Scenario
Games	***Asphalt9***	Play a stage.
	Clash Royale	Play a stage.
	Genshin Impact	Explore the map.
Social Media	***Facebook***	Browse Feed, Watch, Stories, Notifications, and other pages.
	Instagram	Browse Feed, Reels and write comments.
	Twitter	Browse Timeline, Trending, Notifications, and reply to a tweet.
Utilities	***Airbnb***	Browse the globe map and opens multiple listings.
	Facebook Messenger	Browse Chats, Calls, People, Stories, and send messages.
	Uber	Browse Activity, Account settings, and search the map.



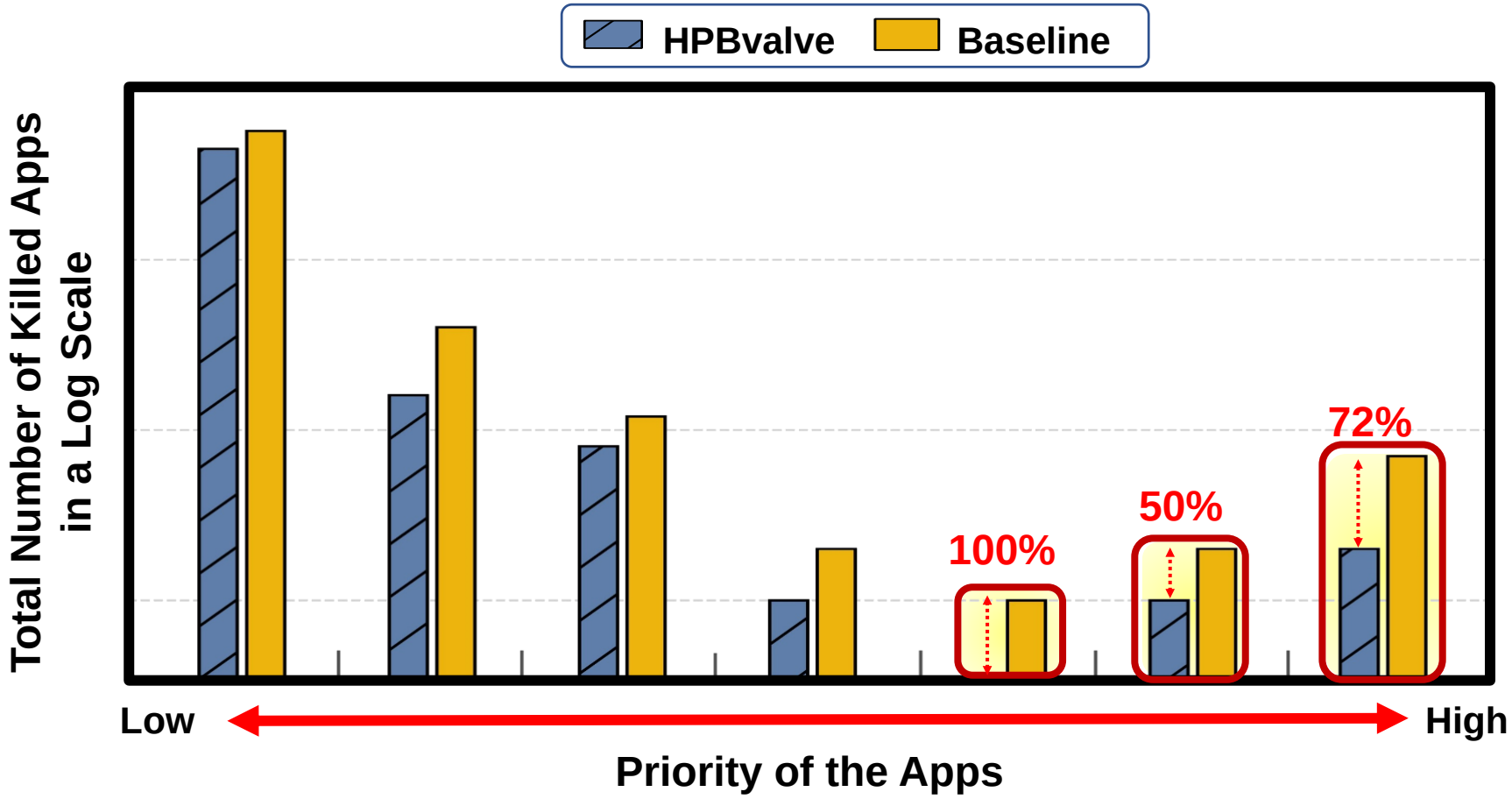
CPU: Snapdragon 888
OS: Android 12
Kernel: v5.4.161
DRAM: 12GB
Storage: 1TB
Storage Mapping: 1GB

Result 1: Improvement in User-Perceived Latency



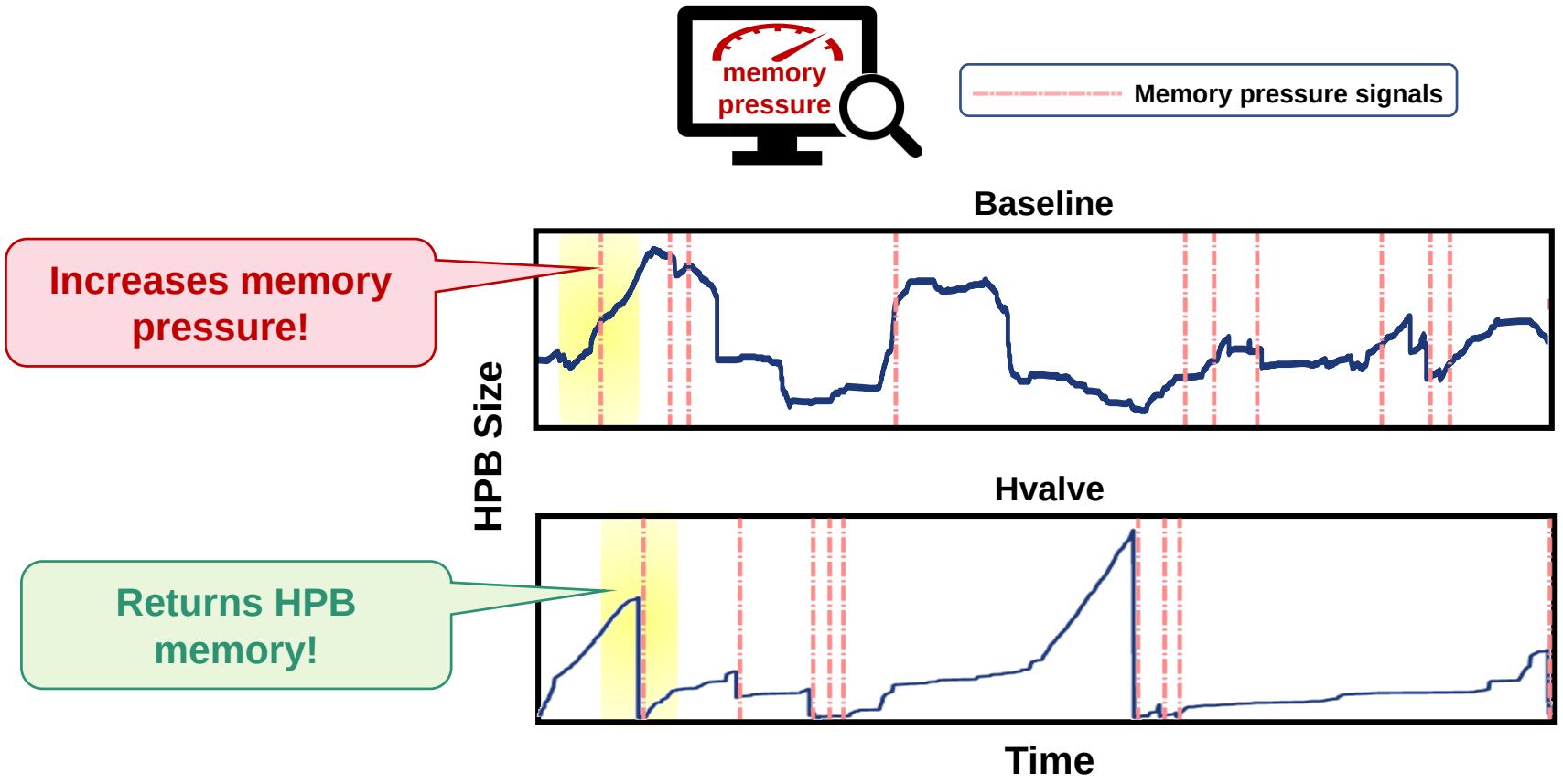
HPBvalve achieved almost identical user-perceived latencies as Optimal.

Result 2: Number of Killed Apps

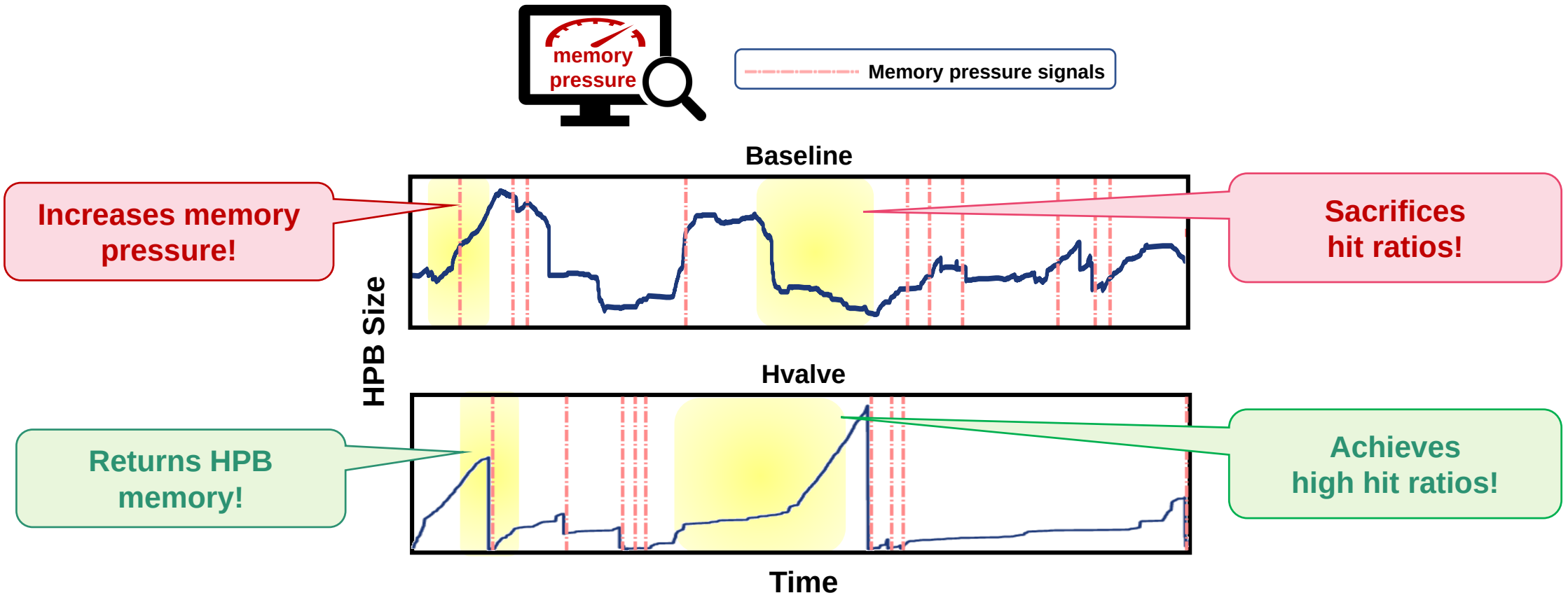


HPBvalve prevents 25% of total app kills and 70% of high-priority app kills.

Result 3: HPB Size Changes



Result 3: HPB Size Changes



HPBvalve successfully manages the HPB size in an harmonized manner with memory pressure status.

Conclusions

- Presented **HPBvalve that efficiently manages L2P entries in HPB**.
  - HPBvalve works in a ***FG app-aware fashion*** while considering the host ***memory pressure*** status.

- Achieved near-optimal performance level for most apps.
  - Reduced the ***user-perceived response time by up to 43%*** over a baseline scheme.
  - Prevented ***70% of unintentional UX-critical app kills*** due to HPB induced memory pressure over a baseline scheme.

- Future Work
  - Extend HPBvalve to manage the device-side SRAM L2P cache.

Thank You ☺

The source code of HPBvalve is available at
https://github.com/cares-davinci/Hvalve





The 21st USENIX Conference on File and Storage Technologies (FAST), 2023