MobiCom 2024
The 30th Annual International Conference On
Mobile Computing And Networking

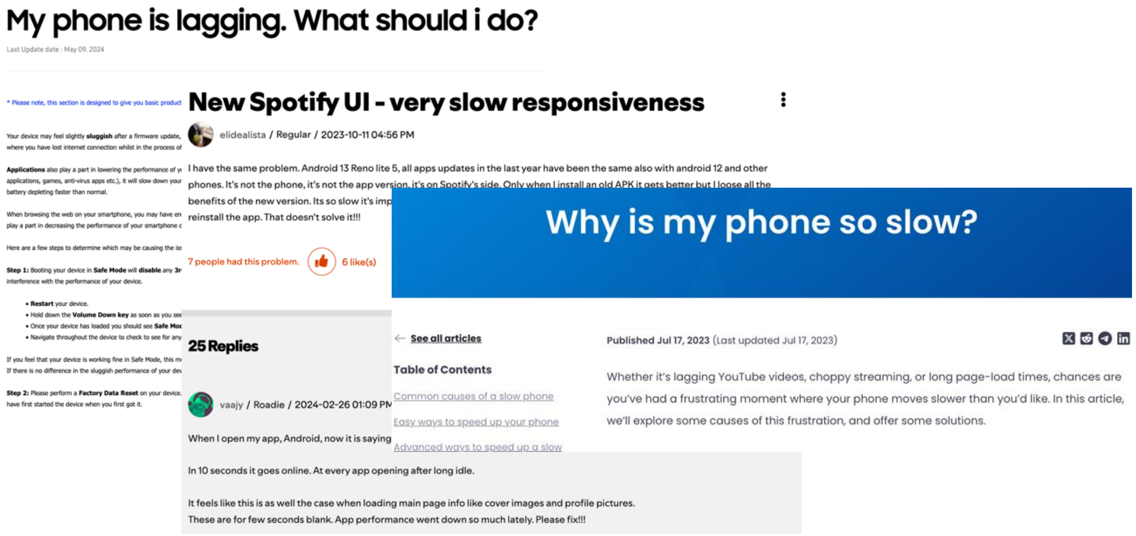
Rethinking Process Management for Interactive Mobile Systems

Jianwei Zheng, Zhenhua Li, Feng Qian, Wei Liu🎙, Hao Lin
Yunhao Liu, Tianyin Xu, Nan Zhang, Ju Wang, Cang Zhang

清華大學 Tsinghua University
xiaomi
USC University of Southern California
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

1. Slow UI Responsiveness

- □ **Interactive Mobile Systems** support emerging highly interactive apps, e.g., AR/VR/MR, 3D games, and metaverse
- □ **Responsiveness: a key metric to measure the quality of user experience**
- □ Till now **slow UI responsiveness (SUR)** on interactive mobile systems is still prevalent



My phone is lagging. What should i do?

Last Update date : May 09. 2024

* Please note, this section is designed to give you basic product

Your device may feel slightly **sluggish** after a firmware update,
where you have lost internet connection whilst in the process of

Applications also play a part in lowering the performance of y
applications, games, anti-virus apps etc.), it will slow down your
battery depleting faster than normal.

When browsing the web on your smartphone, you may have en
play a part in decreasing the performance of your smartphone c

Here are a few steps to determine which may be causing the is

Step 1: Booting your device in **Safe Mode** will **disable** any **3r**
interference with the performance of your device.

- **Restart** your device.
- Hold down the **Volume Down key** as soon as you see
- Once your device has loaded you should see **Safe Moc**
- Navigate throughout the device to check to see for any

If you feel that your device is working fine in Safe Mode, this m
If there is no difference in the sluggish performance of your dev

Step 2: Please perform a **Factory Data Reset** on your device.
have first started the device when you first got it.

New Spotify UI - very slow responsiveness

elidealista / Regular / 2023-10-11 04:56 PM

I have the same problem. Android 13 Reno lite 5, all apps updates in the last year have been the same also with android 12 and other
phones. It's not the phone, it's not the app version, it's on Spotify's side. Only when I install an old APK it gets better but I loose all the
benefits of the new version. Its so slow it's imp
reinstall the app. That doesn't solve it!!!

7 people had this problem. 6 like(s)

25 Replies

vaajy / Roadie / 2024-02-26 01:09 PM

When I open my app, Android, now it is saying

In 10 seconds it goes online. At every app opening after long idle.

It feels like this is as well the case when loading main page info like cover images and profile pictures.
These are for few seconds blank. App performance went down so much lately. Please fix!!!

Why is my phone so slow?

← See all articles

Table of Contents

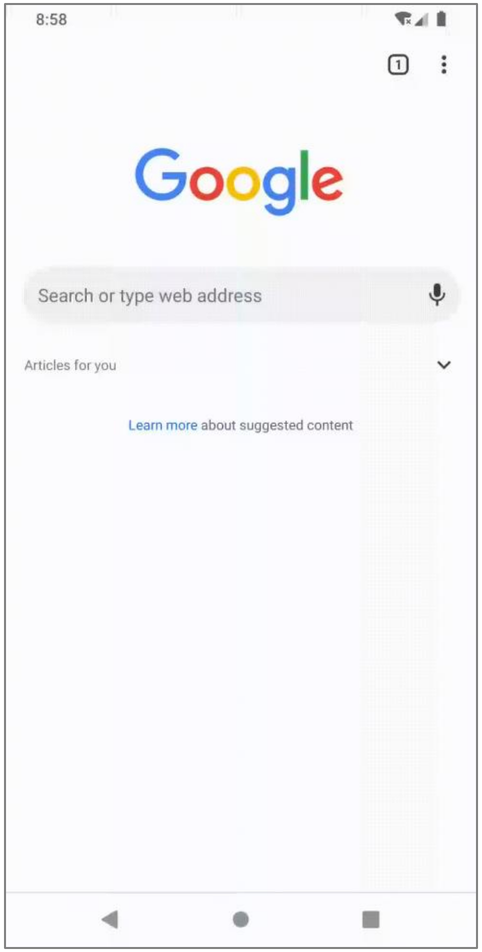
Common causes of a slow phone

Easy ways to speed up your phone

Advanced ways to speed up a slow

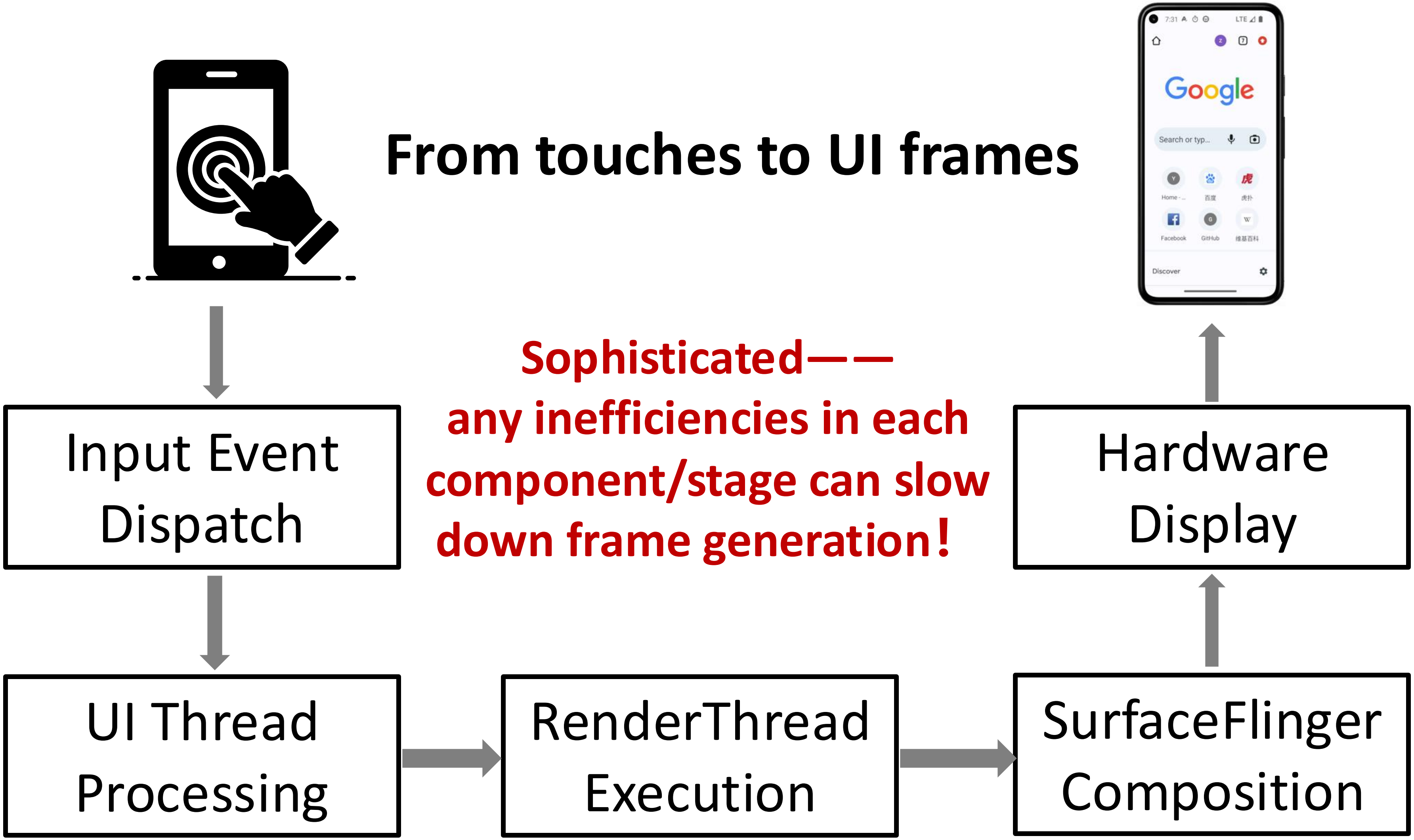
Published Jul 17, 2023 (Last updated Jul 17, 2023)

Whether it's lagging YouTube videos, choppy streaming, or long page-load times, chances are you've had a frustrating moment where your phone moves slower than you'd like. In this article, we'll explore some causes of this frustration, and offer some solutions.


8:58
Google
Search or type web address
Articles for you
Learn more about suggested content

Typical slow UI responsiveness

2. Frame Rendering Pipeline of Android



From touches to UI frames

Input Event Dispatch

UI Thread Processing

RenderThread Execution

SurfaceFlinger Composition

Hardware Display

Sophisticated——
any inefficiencies in each component/stage can slow down frame generation!

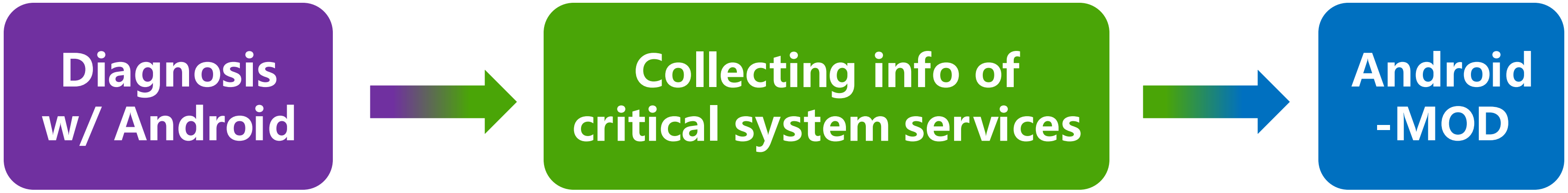
2. Continuous Monitoring Infrastructure

□ Android’s original diagnostic mechanism is not enough

- □ No formal definition of “perceptible” SUR events
- □ Lacking insights into critical system services

□ Our continuous monitoring infrastructure

- □ We collaborate with Xiaomi, a major phone vendor in China
- □ Android-MOD: a customized Android system
- □ Modifying vanilla Android versions 10.0, 11.0, and 12.0



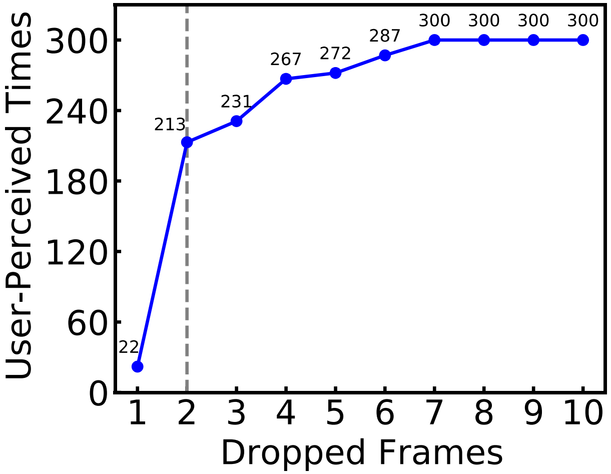
Diagnosis w/ Android → **Collecting info of critical system services** → **Android -MOD**

Lacking insights into critical system services

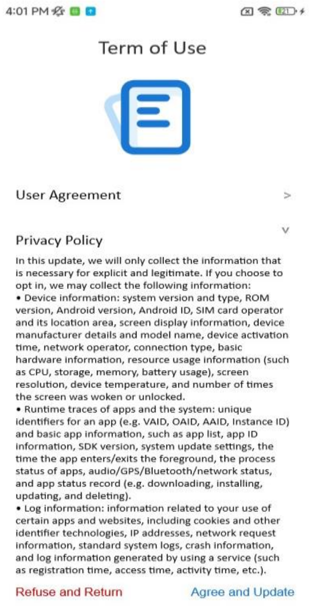
Need to modify the Android framework

2. Continuous Monitoring Infrastructure

- □ **SUR definition**: we did a user study by recruiting a variety of volunteers to identify perceptible SUR events (i.e., rendering delay > 50 ms)
- □ **System service instrumentation**: we modify the code of critical system services to insert monitoring hooks, e.g., monitoring the lock contention
- □ **Cross-layer in-situ information tracing**: e.g., CPU/memory/IO utilization
- □ Lightweight: negligible runtime overhead



User-perceived times of consecutively dropped frames



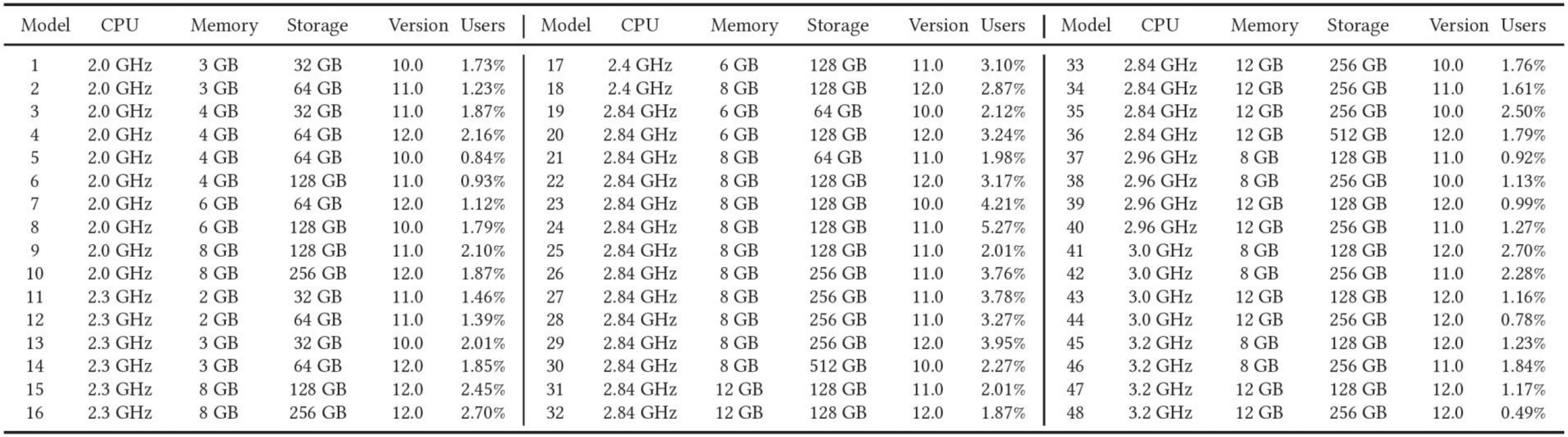
In this update, we will only collect the information that is necessary for explicit and legitimate. If you choose to opt in, we may collect the following information:

- Device information: system version and type, ROM version, Android version, Android ID, SIM card operator and its location area, screen display information, device manufacturer details and model name, device activation time, network operator, connection type, basic hardware information, resource usage information (such as CPU, storage, memory, battery usage), screen resolution, device temperature, and number of times the screen was woken or unlocked.
- Runtime traces of apps and the system: unique identifiers for an app (e.g. VAID, OAID, AAID, Instance ID) and basic app information, such as app list, app ID information, SDK version, system update settings, the time the app enters/exits the foreground, the process status of apps, audio/GPS/Bluetooth/network status, and app status record (e.g. downloading, installing, updating, and deleting).
- Log information: information related to your use of certain apps and websites, including cookies and other identifier technologies, IP addresses, network request information, standard system logs, crash information, and log information generated by using a service (such as registration time, access time, activity time, etc.).

User consent form

2. Crowdsourcing Measurement

- □ We invited 500M Xiaomi users to participate, and 47M opted in
- □ They upgraded the OS to Android-MOD to record SUR event data
- □ The measurement lasted for four months (06-09/2022), involving a wide range of phones across 48 different models and 1M+ apps



Model	CPU	Memory	Storage	Version	Users	Model	CPU	Memory	Storage	Version	Users	Model	CPU	Memory	Storage	Version	Users
1	2.0 GHz	3 GB	32 GB	10.0	1.73%	17	2.4 GHz	6 GB	128 GB	11.0	3.10%	33	2.84 GHz	12 GB	256 GB	10.0	1.76%
2	2.0 GHz	3 GB	64 GB	11.0	1.23%	18	2.4 GHz	8 GB	128 GB	12.0	2.87%	34	2.84 GHz	12 GB	256 GB	11.0	1.61%
3	2.0 GHz	4 GB	32 GB	11.0	1.87%	19	2.84 GHz	6 GB	64 GB	10.0	2.12%	35	2.84 GHz	12 GB	256 GB	10.0	2.50%
4	2.0 GHz	4 GB	64 GB	12.0	2.16%	20	2.84 GHz	6 GB	128 GB	12.0	3.24%	36	2.84 GHz	12 GB	512 GB	12.0	1.79%
5	2.0 GHz	4 GB	64 GB	10.0	0.84%	21	2.84 GHz	8 GB	64 GB	11.0	1.98%	37	2.96 GHz	8 GB	128 GB	11.0	0.92%
6	2.0 GHz	4 GB	128 GB	11.0	0.93%	22	2.84 GHz	8 GB	128 GB	12.0	3.17%	38	2.96 GHz	8 GB	256 GB	10.0	1.13%
7	2.0 GHz	6 GB	64 GB	12.0	1.12%	23	2.84 GHz	8 GB	128 GB	10.0	4.21%	39	2.96 GHz	12 GB	128 GB	12.0	0.99%
8	2.0 GHz	6 GB	128 GB	10.0	1.79%	24	2.84 GHz	8 GB	128 GB	11.0	5.27%	40	2.96 GHz	12 GB	256 GB	11.0	1.27%
9	2.0 GHz	8 GB	128 GB	11.0	2.10%	25	2.84 GHz	8 GB	128 GB	11.0	2.01%	41	3.0 GHz	8 GB	128 GB	12.0	2.70%
10	2.0 GHz	8 GB	256 GB	12.0	1.87%	26	2.84 GHz	8 GB	256 GB	11.0	3.76%	42	3.0 GHz	8 GB	256 GB	11.0	2.28%
11	2.3 GHz	2 GB	32 GB	11.0	1.46%	27	2.84 GHz	8 GB	256 GB	11.0	3.78%	43	3.0 GHz	12 GB	128 GB	12.0	1.16%
12	2.3 GHz	2 GB	64 GB	11.0	1.39%	28	2.84 GHz	8 GB	256 GB	11.0	3.27%	44	3.0 GHz	12 GB	256 GB	12.0	0.78%
13	2.3 GHz	3 GB	32 GB	10.0	2.01%	29	2.84 GHz	8 GB	256 GB	12.0	3.95%	45	3.2 GHz	8 GB	128 GB	12.0	1.23%
14	2.3 GHz	3 GB	64 GB	12.0	1.85%	30	2.84 GHz	8 GB	512 GB	10.0	2.27%	46	3.2 GHz	8 GB	256 GB	11.0	1.84%
15	2.3 GHz	8 GB	128 GB	12.0	2.45%	31	2.84 GHz	12 GB	128 GB	11.0	2.01%	47	3.2 GHz	12 GB	128 GB	12.0	1.17%
16	2.3 GHz	8 GB	256 GB	12.0	2.70%	32	2.84 GHz	12 GB	128 GB	12.0	1.87%	48	3.2 GHz	12 GB	256 GB	12.0	0.49%

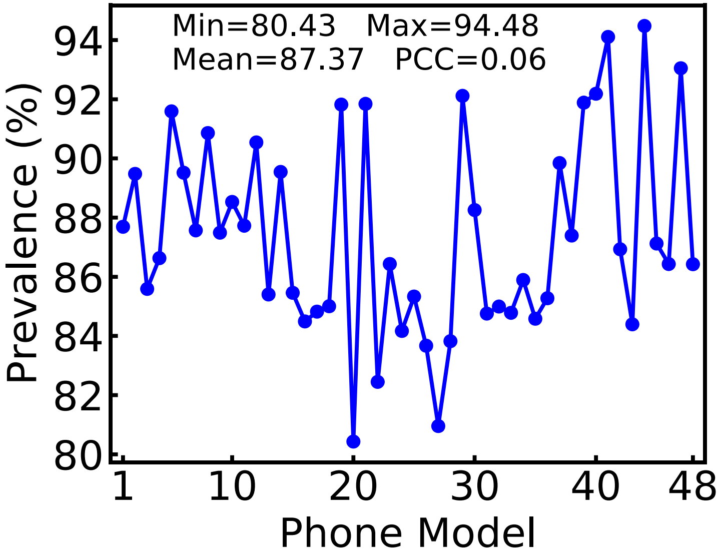
Hardware and OS configurations of our measured phone models

3. Key Findings: Hardware

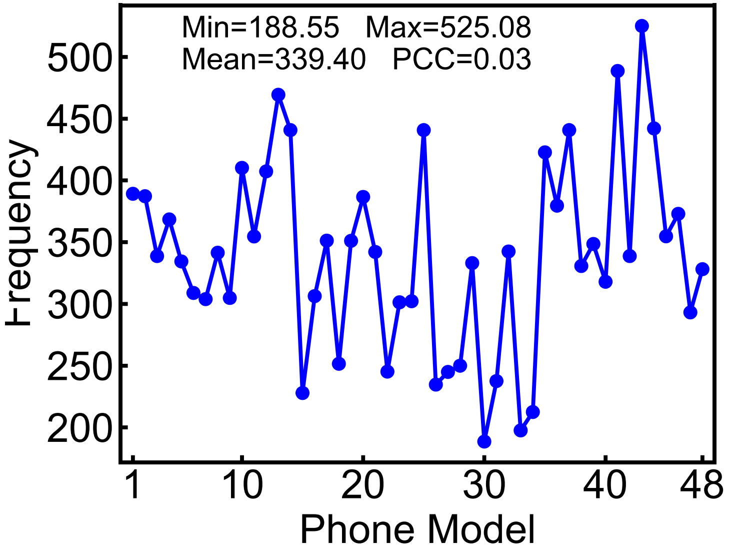
□ SUR events occur prevalently on all the 48 models per day (ranging from 80.42% to 97.73% with an average of 86.95%)

□ SUR events happen frequently on each specific model per day (ranging from 179.59 to 554.68 with an average of 338.28)

□ Better hardware cannot effectively reduce SUR events



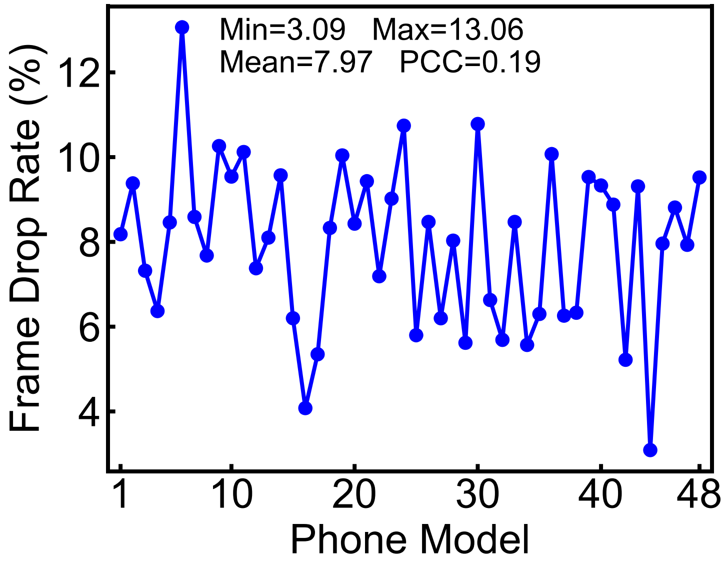
Prevalence of SUR events on each model of phones



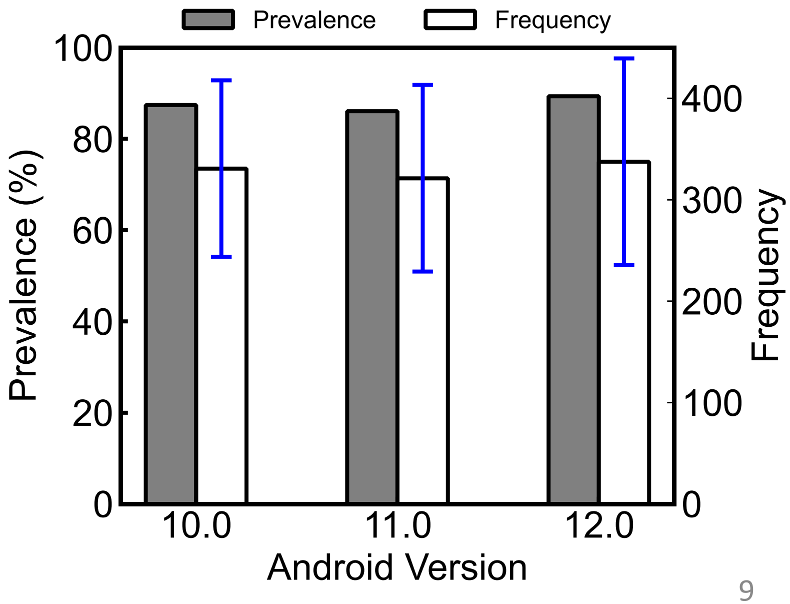
Frequency of SUR events on each model of phones

3. Key Findings: Frame Drop Rate & OS

- Frame drop rates of specific models are worryingly high: ranging from 3.09% to 14.12% with an average of 7.91%
- Newer OS cannot effectively mitigate SUR events, owing to higher stability & robustness of older OSes (Android 10 and 11) and that Android 12 was still undergoing constant patches and required mobile apps to adapt to the newly-provided APIs



Frame drop rate of each phone model



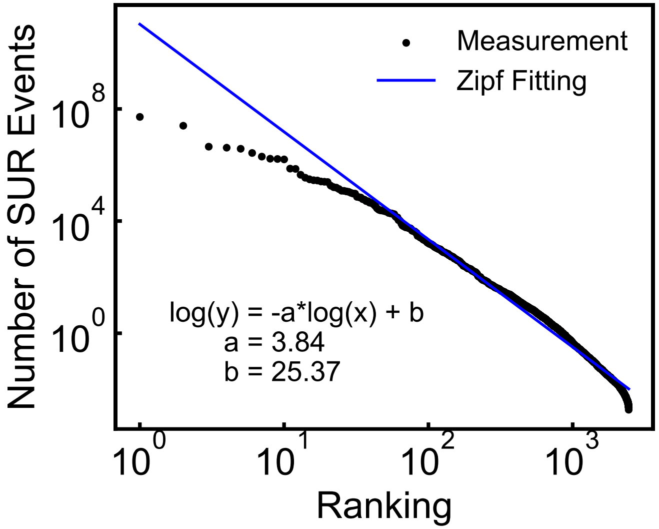
SUR prevalence and frequency of each Android version

3. Key Findings: Mobile Apps

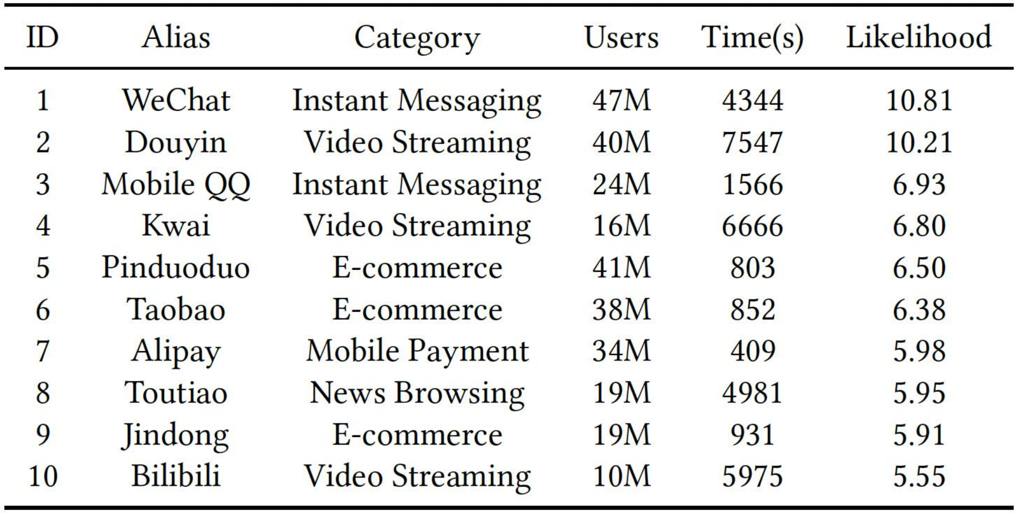
□ SUR event occurrences on different apps are skewed

□ 16.8% SUR events are attributed to top-10 (<0.001%) apps

□ Heavy workloads incurred by high-resolution media streaming, embedded WebView browsers, and complex UI functionalities



Ranking of apps by their number of SUR events per day



ID	Alias	Category	Users	Time(s)	Likelihood
1	WeChat	Instant Messaging	47M	4344	10.81
2	Douyin	Video Streaming	40M	7547	10.21
3	Mobile QQ	Instant Messaging	24M	1566	6.93
4	Kwai	Video Streaming	16M	6666	6.80
5	Pinduoduo	E-commerce	41M	803	6.50
6	Taobao	E-commerce	38M	852	6.38
7	Alipay	Mobile Payment	34M	409	5.98
8	Toutiao	News Browsing	19M	4981	5.95
9	Jindong	E-commerce	19M	931	5.91
10	Bilibili	Video Streaming	10M	5975	5.55

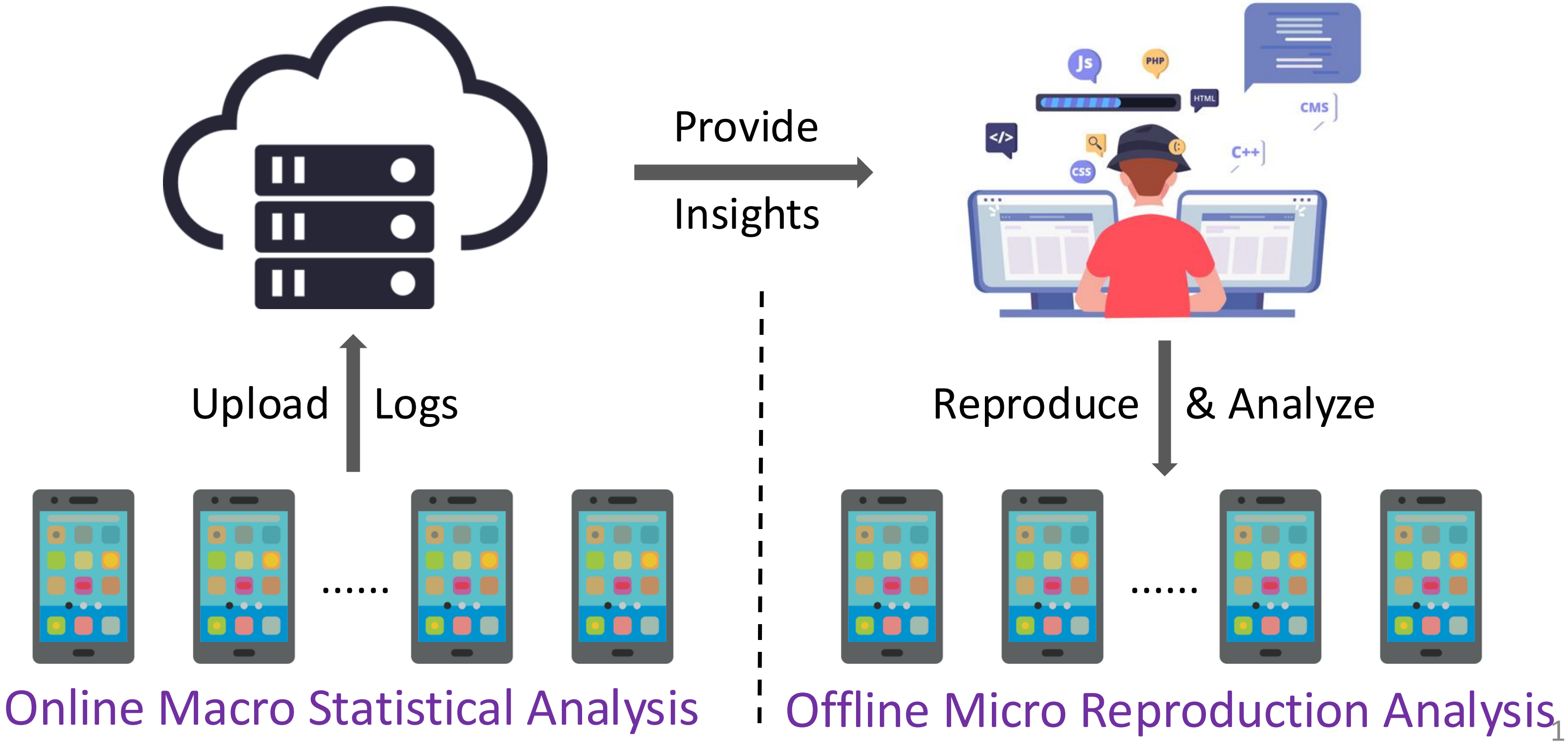
Top-10 apps ordered by the frequency (or simply likelihood) of SUR events **after normalization**

3. Key Findings: Root Cause Analysis

□ System/App developers usually analyze SUR logs by hand

□ We develop a semi-automatic two-phase analysis pipeline

□ The first phase classifies SUR events with the same root cause to a cluster, and the second phase pinpoints the root cause



Provide Insights

Upload Logs

Reproduce & Analyze

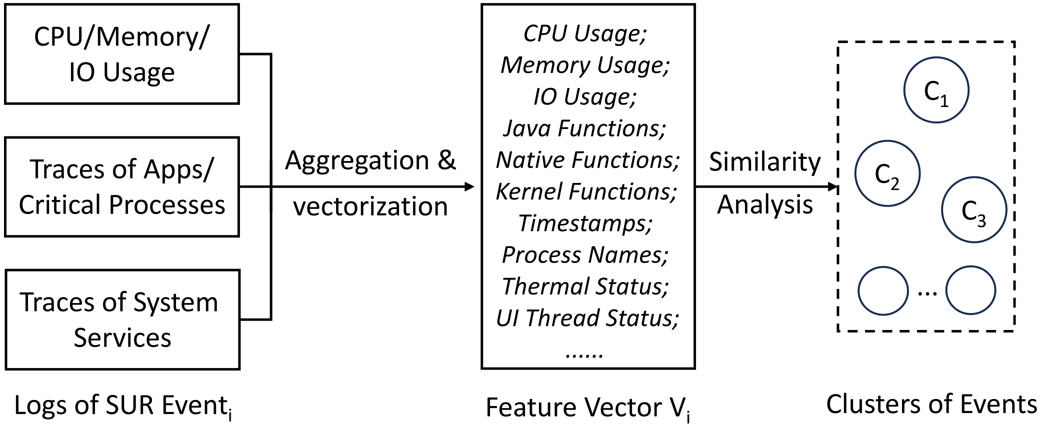
Online Macro Statistical Analysis

Offline Micro Reproduction Analysis

3. Key Findings: Root Causes Analysis

□ Phase 1: Online Macro-level Statistical Analysis

□ Clustering results: long CPU scheduling delay (20.98%), slow I/O transactions (11.32%), insufficient memory (26.70%), and app-specific defects (41%)



□ Phase 2: Offline Micro-level Reproduction Analysis

□ Define the time window around an SUR event (±1s)

□ Quantitatively assess the correlation between the occurrence time of SUR events and the lifespan of low-priority processes (> 0.91)

□ **Key insight**: The persistent survival of numerous low-priority processes of **hogging apps** leads to system resource under-provisioning and contention, and thus causes SUR events

3. Key Findings: Hogging Apps

□ Major Keep-Alive Patterns

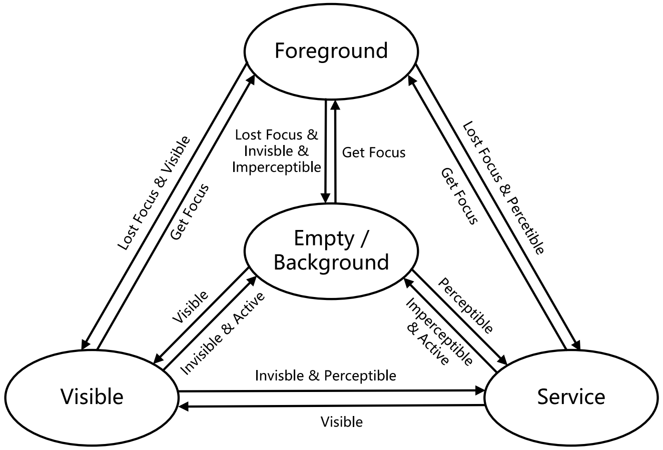
- □ Abuse foreground services (e.g., GPS, Audio, Bluetooth, and Network) to deliberately increase their processes' priorities
- □ Conduct dual-process co-awakening via process binding

□ Commercial Motivations

- □ User retention & engagement: escalate user retention and engagement rates, and thus potentially boost the revenue
- □ In-app advertisements: continually display ads or push notifications to users, thereby generating revenue
- □ Data harvesting: continuously collect user data
- □ Cross-app awakening: leverage the sustained presence to promote or awaken other apps from the same developer or affiliated partners

4. Rethinking Process Management

- Manage the lifecycle of each app process
- Decide which process(es) should be kept alive or killed when system resources become constrained
- Priority: Foreground > Visible > Service > Background > Empty



The state machine that models the process management in Android

Over-optimistic Assumption

Process State Transition Is Deterministic

Real-world Scenario

State Transition Should Consider Processes' Actual Behaviors

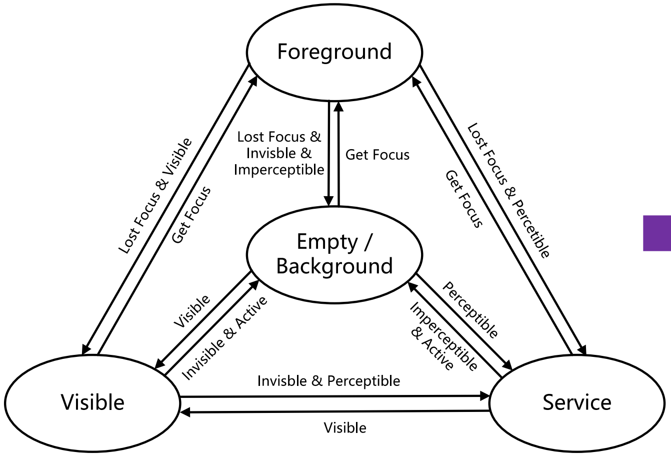
4. Remodeling Process Management

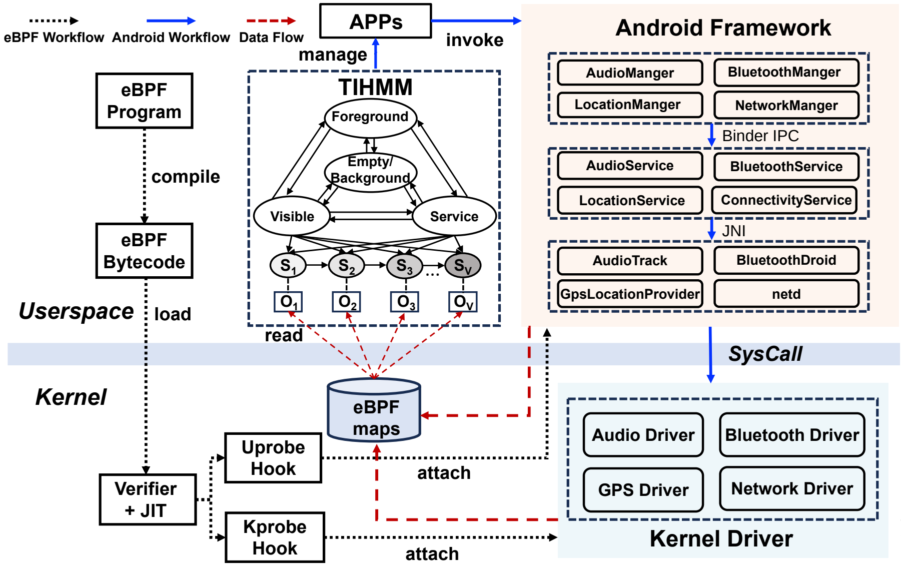
□ TIHMM-based Process State Modeling

- □ Add new hidden states (“hogging”) to the original state machine
- □ Formalize the process transition as a **Time-inhomogeneous Hidden Markov Model (TIHMM)** transition in a time-sensitive manner

□ eBPF-based Uniform Authentic Sensing

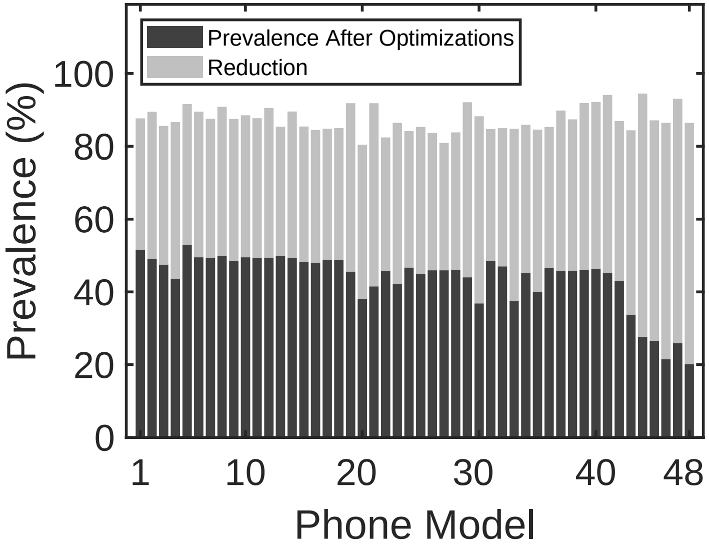
- □ Leverage eBPF to sense the real usage of user-perceptible foreground services as **observations**, by attaching probes to the corresponding codes across kernel and framework

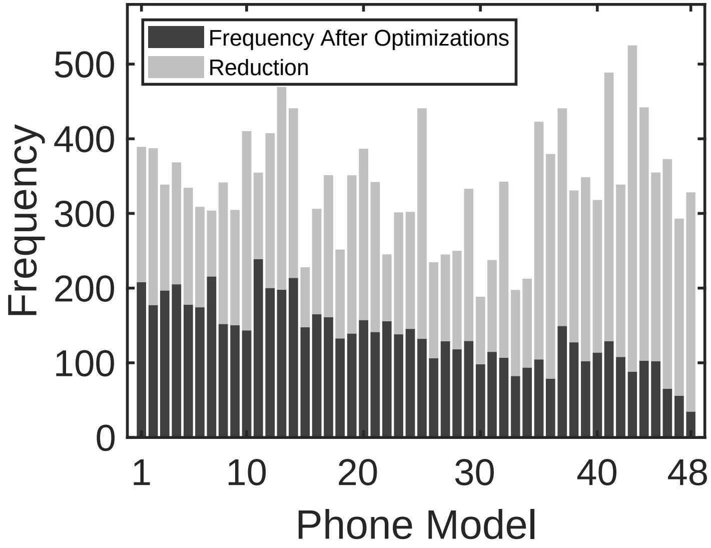




4. Real-world Deployment & Evaluation

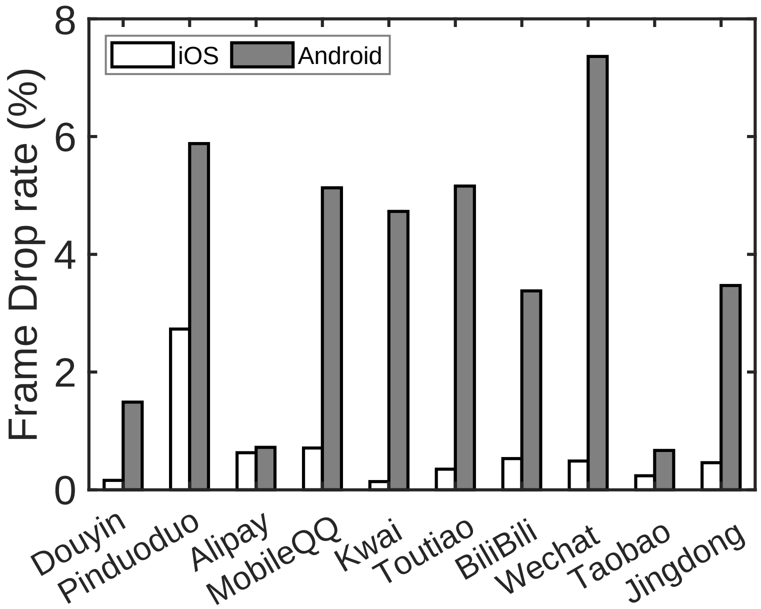
- □ Patched our proposed mechanism to Android-MOD
- □ Invited the original 47M users to upgrade (60% opted in)
- □ The evaluation spanned two months (Jan.–Feb. 2023)
- □ Reduce the prevalence of SUR events by 50%
- □ Reduce the frequency of SUR events by 60%
- □ Reduce the battery consumption by 10.7% due to the effective throttling of resource usages from hogging apps





4. Android vs. iOS

- With the similar hardware configuration, Android suffers far more (oftentimes 10X) SUR events than iOS according to our measurement
- Hardware and software co-design of iPhones: iOS can be fine-tuned to work perfectly with the specific hardware it runs on
- More stringent scrutiny policy: Apple's App Store has stricter guidelines and a more rigorous app review process than Google Play Store



SUR = Slow UI Responsiveness

5. Summary of Contributions

- Conduct the first large-scale measurement study on SUR (Slow UI Responsiveness) events for Android in the wild with the generous help from 47 million Xiaomi users; share our continuous monitoring infrastructure for capturing SUR events on user devices

- Present our semi-automatic analysis pipeline for deeply understanding SUR events; pinpoint the largest root cause of SUR to be the system-wise resource contention caused by the wide existence of hogging apps

- Remodel Android process management to effectively detect & suppress hogging apps; real-world deployment reduces the occurrence of SUR events by 60% and saves the battery consumption by 10.7%

- Code and data released at https://Android-SUR.github.io

Thanks!
Q & A