Virtual Device Farms for Mobile App Testing at Scale:
A Pursuit for Fidelity, Efficiency, and Accessibility

Hao Lin, Jiaxing Qiu, Hongyi Wang, Zhenhua Li, Liangyi Gong,
Di Gao, Yunhao Liu, Feng Qian, Zhao Zhang, Ping Yang, Tianyin Xu

清華大學 Tsinghua University
ByteDance
CNIC
USC
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

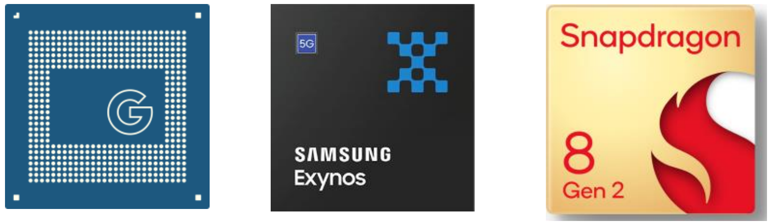
Mobile app testing in an open ecosystem is challenging

- ❑ Hundreds of new Android phone models are released every year

❑ Heterogeneous hardware

- Screen

- SoC



- Radio, camera, biometrics, sensors, etc.

❑ Highly-customized software

- Custom Android systems
- Unique service platforms

Solution of Douyin's team: physical device farm

- A massive physical device farm
  - Distributed across China and US
  - **5,918 device models** as of Jan. 2022
  - Popular models are updated every year
  - Cellular + WiFi access
  - A dedicated operation team of **15 engineers**

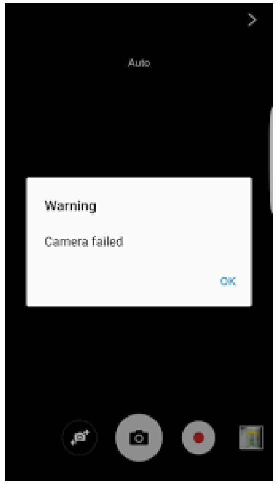
Solution of Douyin’s team: physical device farm

- Total cost of ownership (TCO) becomes untenable
  - TCO = team salary + device purchasing + carrier/WiFi plan + power usage + …
  - **> 1M dollars** for building, **> 0.6M dollars per year** for device purchasing alone
  - **Short lifespan** of mobile phones (~10 months)

Battery swelling

Screen wear out

USB port failure



Camera/sensor issues

Alternative solution: cloud-based testing service

- ❑ Rentable device farms managed by service providers

AWS Device Farm   Google Firebase Test Lab   SauceLabs   LAMBDATEST

+ **Reduced operation cost**

– **Insufficient device model diversity** (only 136 device models in AWS)

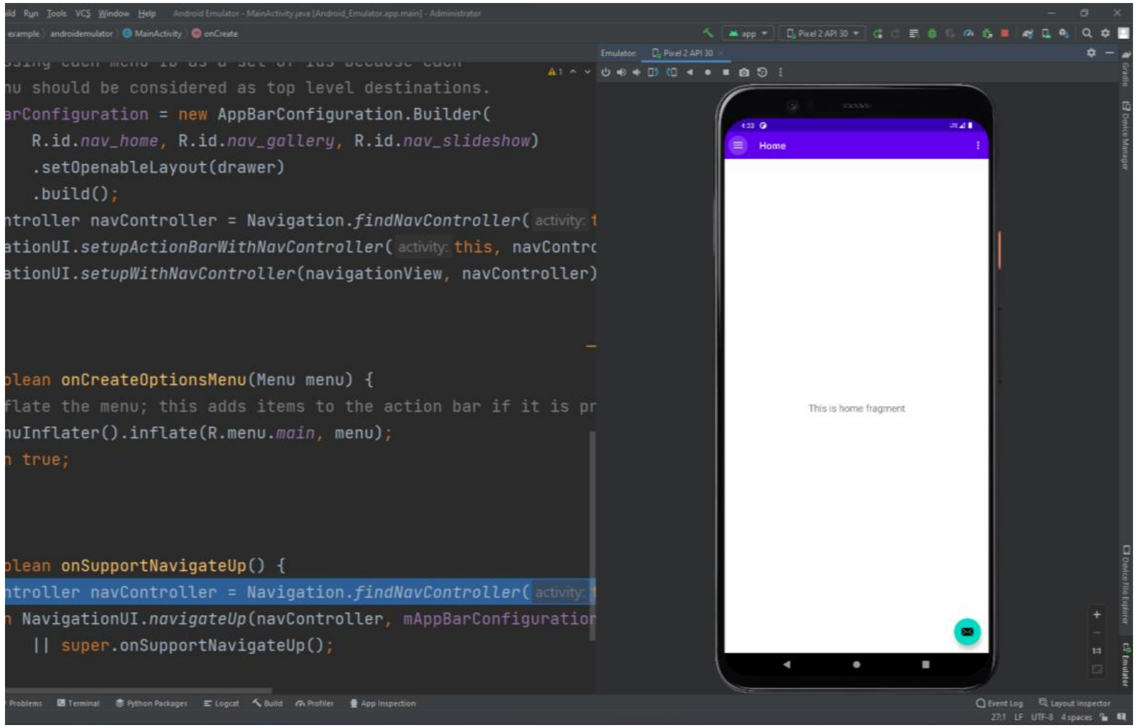
– **Considerable testing constraints** (max app size, max test time, etc.)

– **Limited customizability of the testing pipeline**

Cloud-based testing service is not a viable solution for Douyin

How about virtual devices?

- ❑ Emulate mobile devices on servers using virtualization techniques
- ❑ Already widely used by lab research
- ❑ Unique advantages and features
  - **Scalable, elastic, and cost-effective**
  - Useful features: **instrumentation, memory introspection, snapshot**



Virtual devices remain controversial in industry

The diverse, opaque, and ever-growing devices **are hard to emulate**

⬇

The fidelity concern: discrepancies between physical and virtual devices may lead to escapes of bugs and false alarms

⬇

Even a small number could have **magnified impacts** on global-scale apps like Douyin

Our study goal

1. **Quantitatively** understand virtual devices' **fidelity** and its **impact**

2. Explore how to **improve the efficiency and accessibility** of industrial mobile app testing with virtual devices

Contributions

- A large-scale study of virtual devices for mobile app testing
  - Analysis of **testing fidelity,** and **root causes** of discrepancies
- Design and implementation of a high-fidelity virtual device farm
- Techniques for improving virtual device fidelity
- **Efficiency**: virtual devices for **continuous mobile app testing**
- **Accessibility**: preliminary results of **virtual devices as a service**
- Artifact: https://github.com/Android-Emulation-Testing/emu-fidelity-ae

Study methodology

Comparatively analyze apps' test results on
virtual and physical device farms in production

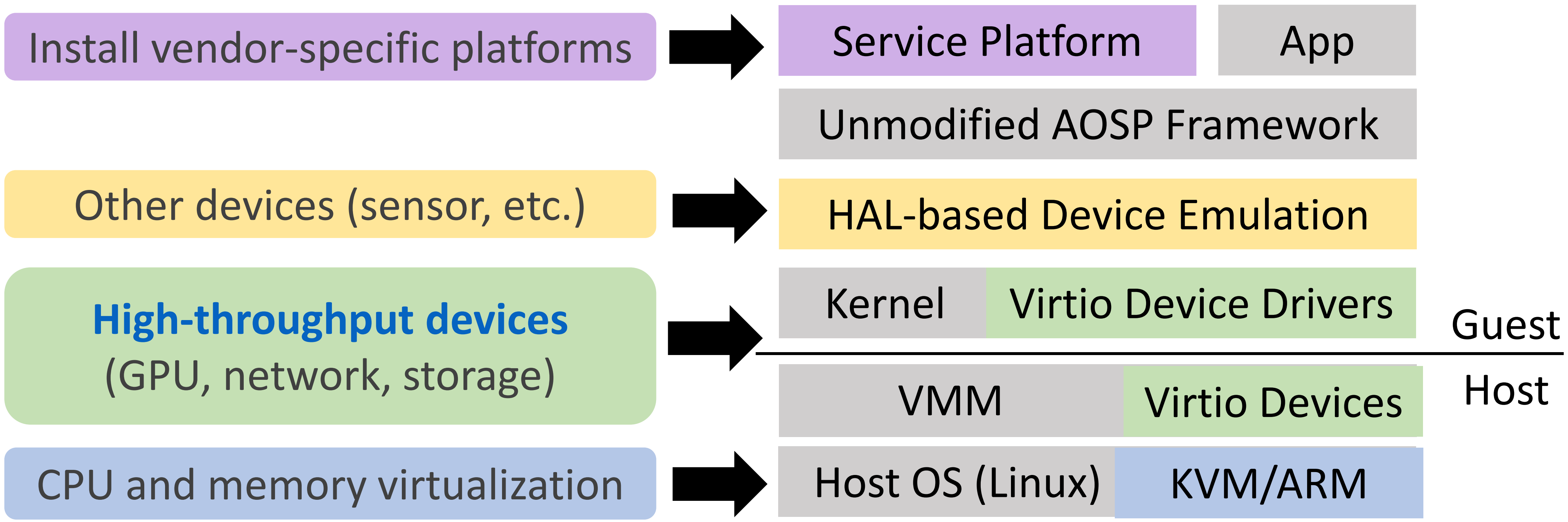
Designing a virtual device farm

- A **digital twin** of the physical farm
  - **5,918 virtual devices** on 395 ARM servers
  - Each virtual device mimics one physical device
- Major design considerations

▪ **Host Hardware**	▪ **Guest OS**	▪ **Guest App**
ARM servers	No Framework hooking	Same app service platform
Binary compatible with Android apps	Avoid changing Framework behaviors	Match vendors' app-related customizations

Building the virtual device farm

- ❑ Hardware: same configuration as the counterpart physical device
- ❑ Software: Cuttlefish Android Emulator with KVM



Testing and debugging tools

- Test case generation
  - **Model-based UI test** technique to generate streams of UI events
- Test failure (App failure) data collection
  - **Lightweight yet fine-grained** in-situ data collection via memory pruning
- Root cause analysis: debugging proprietary vendor components
  - **Binary taint backtracing** to reconstruct instruction and data flows

Study overview

- Study Period: **Jan. 1 to Mar. 31 in 2022**
- Studied apps: **Douyin and nine other global-scale apps**
- Each version release is tested on both physical and virtual farms

App	Functionality	# Users	# Releases	Test Time
Douyin	Video streaming, shopping, social media, map, education, etc.	842M	12	72 hours
Douyin Lite	Video streaming, communication, travel, photography, etc.	210M	12	72 hours
Xigua Video	Video streaming, payment, shopping, 3D gaming, etc.	180M	12	72 hours
Toutiao	News feed, shopping, web browsing, 3D gaming, etc.	530M	12	72 hours
Toutiao Lite	News feed, video streaming, security checking, payment, etc.	130M	12	72 hours
Lark	Communication, email, video conference, cloud storage, etc.	9.4M	5	30 hours
Helo	Social media, video streaming, communication, etc.	50M	12	72 hours
Fizzo Novel	E-book, shopping, 3D gaming, social media, etc.	10M	5	30 hours
Xingfu Li	E-commerce, video streaming, finance, communication, etc.	7.5M	12	72 hours
Resso Music	Music streaming, communication, social media, etc.	40M	9	54 hours

Test failure events and root causes

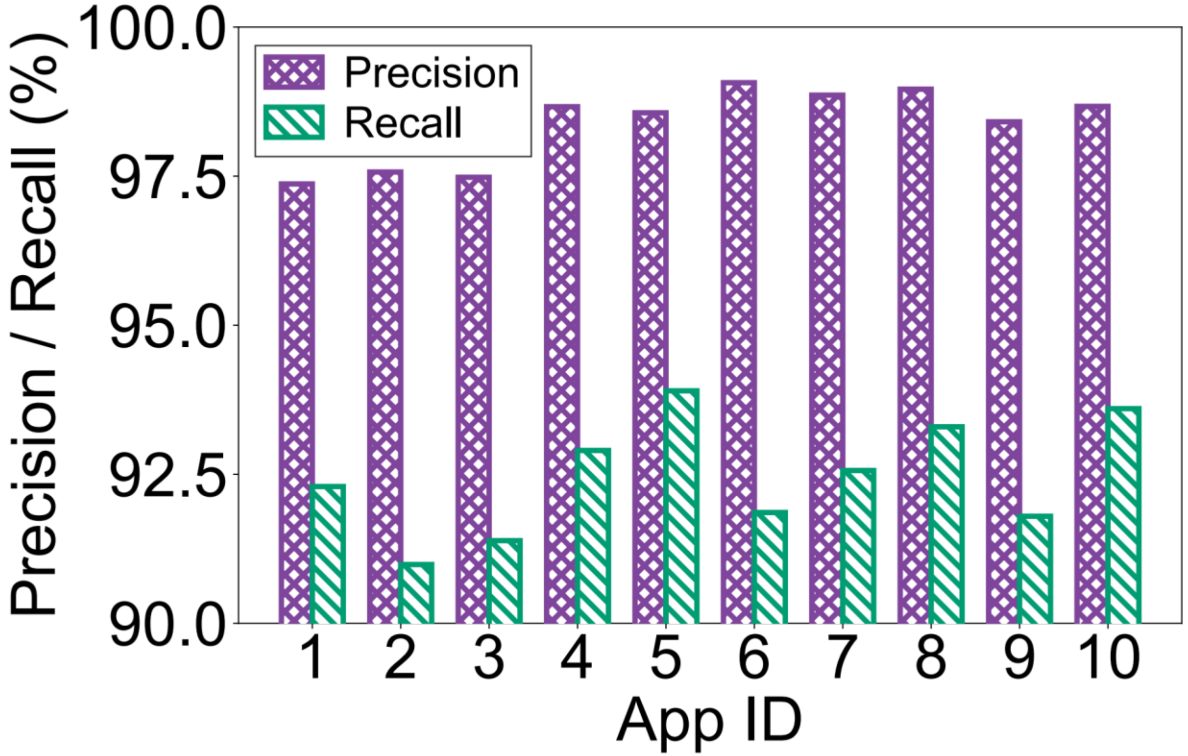
- Test failure events
  - **390K events** on physical devices, **415K events** on virtual devices
  - **2.5% are hardware-specific**, covering all the common mobile hardware
- Root causes
  - A total of **873** root causes
  - Top-10 account for 81% events

No.	Exception/Signal	Root Cause
1	NullPointerException (Java)	Bad resource handling during activity lifecycle shifts
2	NullPointerException (Java)	Defects in OPPO market SDK
3	NullPointerException (Java)	Null object reference in app module
4	NullPointerException (Java)	Attempt to cast null reference to non-null Kotlin class
5	ClassNotFoundException (Java)	Failed resolution of app Java classes
6	NullPointerException (Java)	Method parameter specified as non-null is null
7	ClassCastException (Java)	Incompatible Java class casts
8	OutOfMemoryError (Java)	Out of memory when allocating Bitmap objects
9	NullPointerException (Java)	Method invocation on null app objects
10	OutOfMemoryError (Java)	Out of memory when creating new threads

- *Top-10 most frequent root causes*

Quantitative fidelity: surprisingly good

- ❑ Virtual devices can capture **92.4% failures** on physical devices
- ❑ Only **1.8% of failures** on virtual devices are false alarms



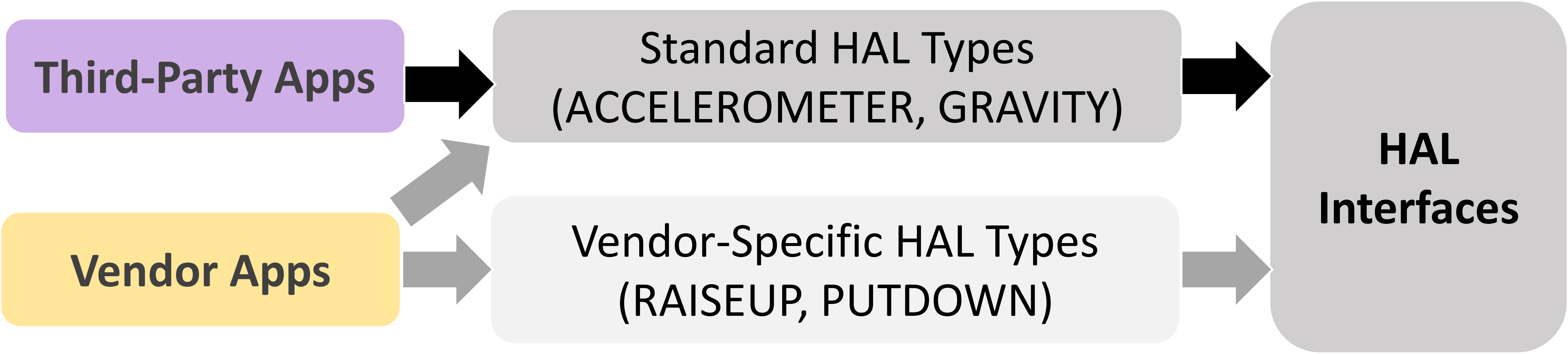
- *Precision and recall per app*

With sensible design, virtual device farms can achieve high-fidelity testing

- ❑ **Still, they are not perfect**

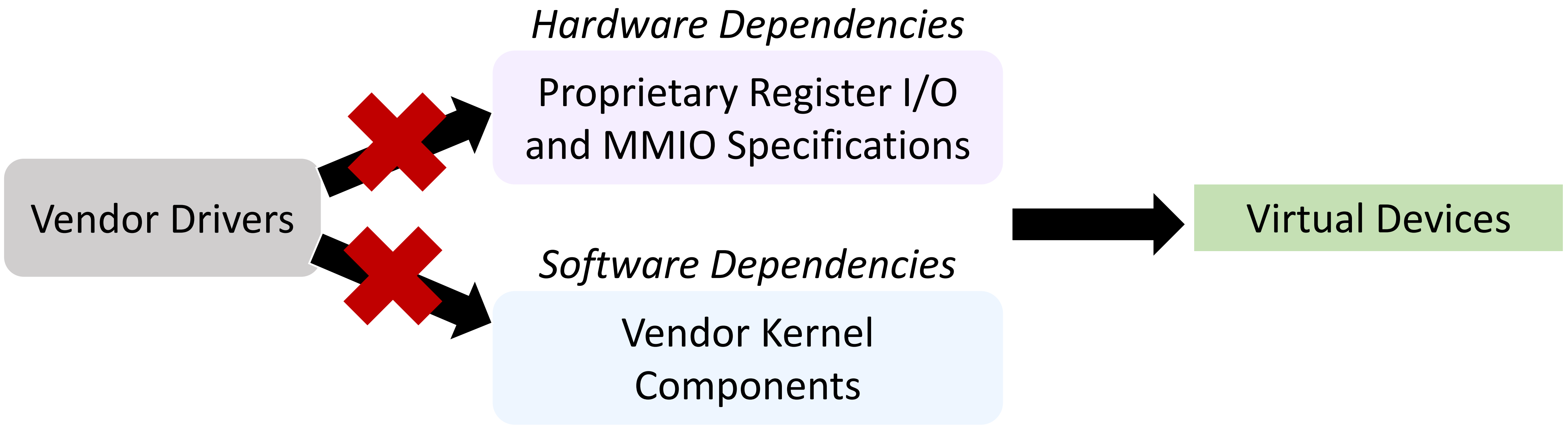
Hardware-level discrepancies

- ❑ ~~Common belief: vendor-specific hardware greatly impairs fidelity~~
- ❑ ***Vendor-specific hardware does not result in major discrepancies***
  - Vendor-specific hardware types **are not defined** by standard Android HAL



Hardware-level discrepancies

- ❑ ***Bugs in common hardware drivers caused 28% false negatives***
  - Errors in MediaTek GPU drivers cause **the third most frequent FN**
- ❑ It is hard for virtual devices to incorporate vendor drivers



Software-level discrepancies

- ***Customizations on vanilla Android components rarely hurt***
- Thanks to Android Compatibility Test Suite and Vendor Test Suite

CTS + VTS → Powered by android +

Compatibility Test Suite | Vendor Test Suite | Trademark | GMS

- CTS/VTS-incompliant models show significantly reduced fidelity

Vendor	# Models	Region	C/VTS	Precision	Recall
Sony	39	Europe	Y	97.9%	89.5%
Oneplus	38	India	Y	96.1%	90.7%
Smartisan	29	China	**N**	**88.7%**	**83.0%**
Vsmart	28	Vietnam	Y	96.6%	89.2%

Software-level discrepancies

- ***Vendor-specific system services incur considerable discrepancies***
- CTS/VTS do not check interfaces between stakeholders
  - Usually **break specification of other stakeholders**

No.	Percent.	Location	Root Cause
1	53.4%	AOSP, Emulator	Graphics resource format inconsistency
2	7.3%	Emulator	Missing graphics buffer allocator
3	6.8%	AOSP, Emulator	Graphics buffer overrun (due to graphics format inconsistency)

- *Top False positives*

No.	Percent.	Location	Root Cause
1	14.9%	AOSP	Integer overflow during implicit conversions
2	9.1%	Meizu	Improper null-terminations of C/C++ strings in vendor modules

- *Top False negatives*

Regional discrepancies

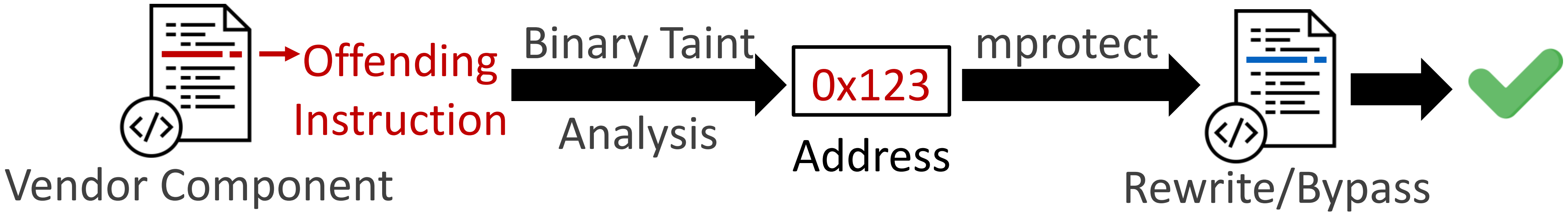
❑ ***Frequency discrepancies are specific to regional ecosystems***

- Up to **1,025× more frequent** occurrences of certain failures on some regional physical device models

Lacking well-regulated app stores → Users are prone to malicious/rogue apps → Aggressive defense mechanisms → Side effects on regular apps

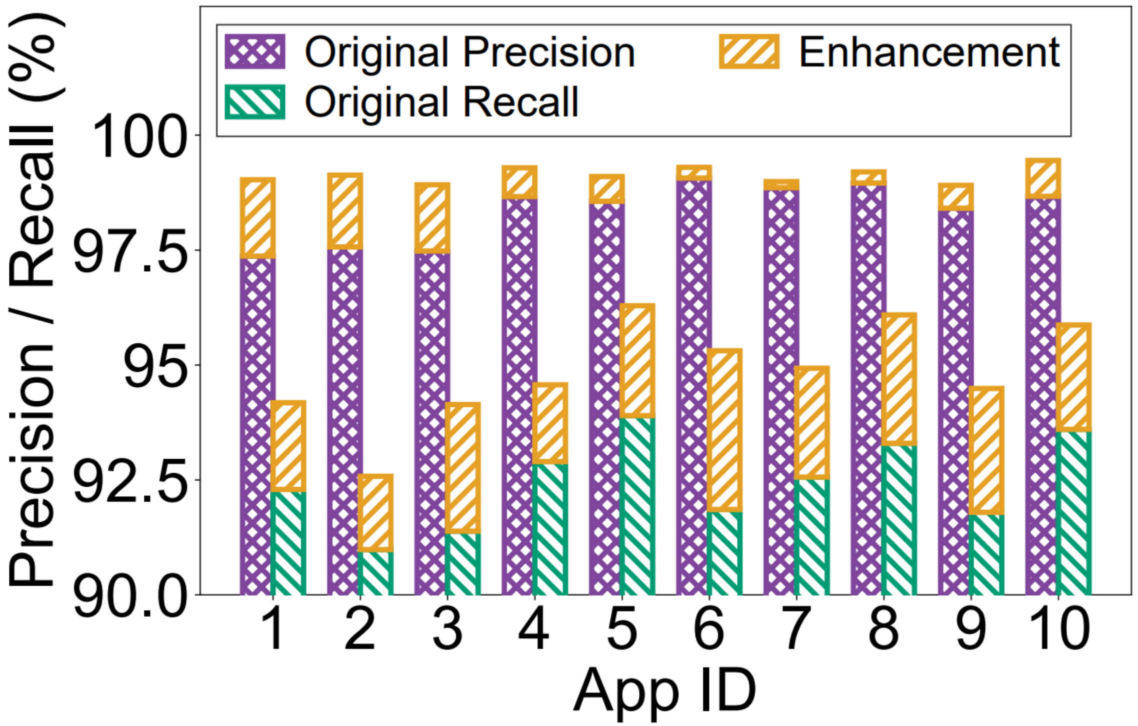
Improving virtual device fidelity

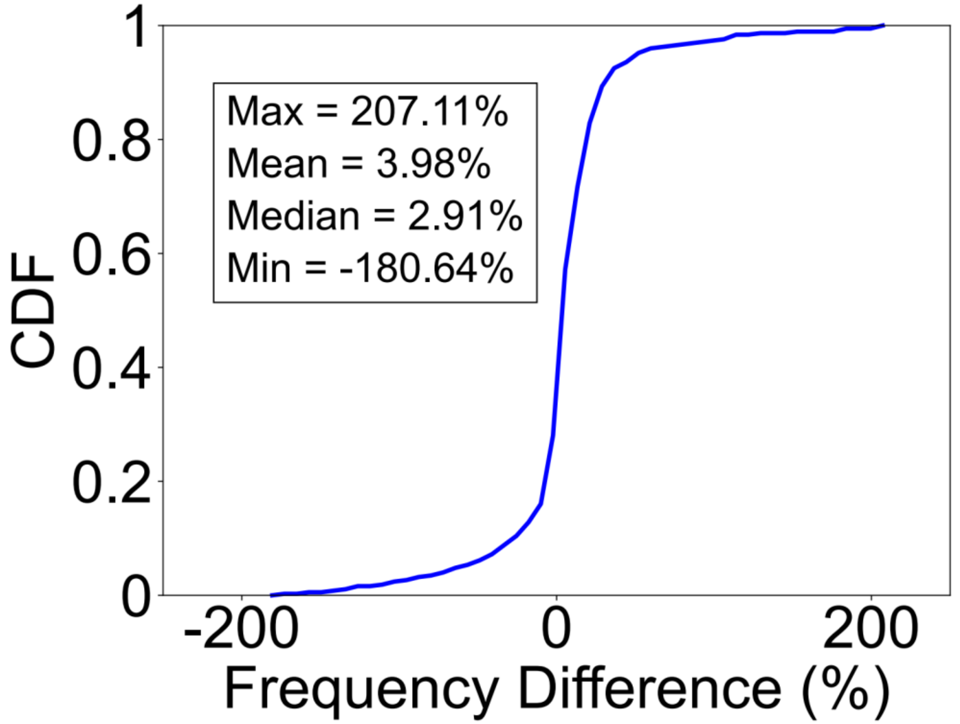
- ❑ Emulator side: **adapt and fix the implementation**
  - Support vendors' malicious app defenses in AOSP
  - Fix & report defective mechanisms
- ❑ Vendor side: **active outreach and communication**
  - Challenge: vendors are **not motivated to fix seemingly app-specific issues**
  - Solution: **dynamic binary patching** to provide proof of causality



Evaluation

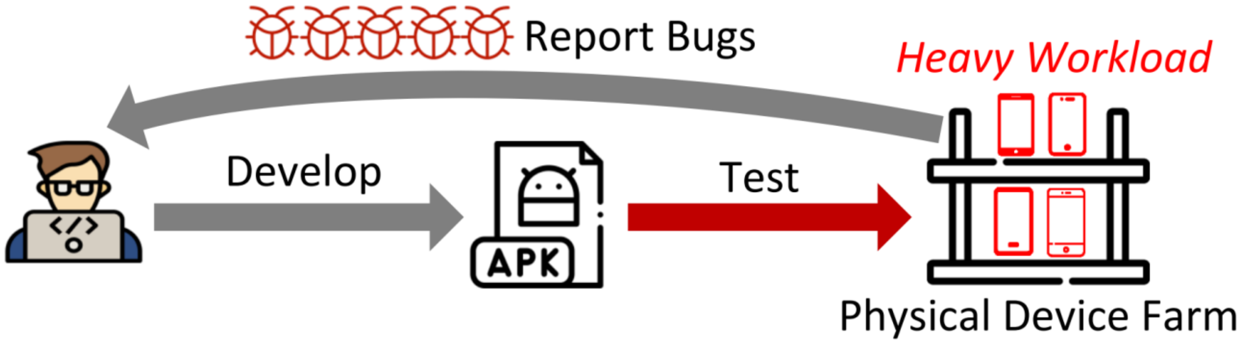
- ❑ 63% of reports have been confirmed and fixes have been merged
- ❑ Remeasure the fidelity from **Jul. 1st to Sep. 30th in 2022**
- ❑ Recall: **92.4% → 94.7%**; Precision: **98.2% → 99.1%**



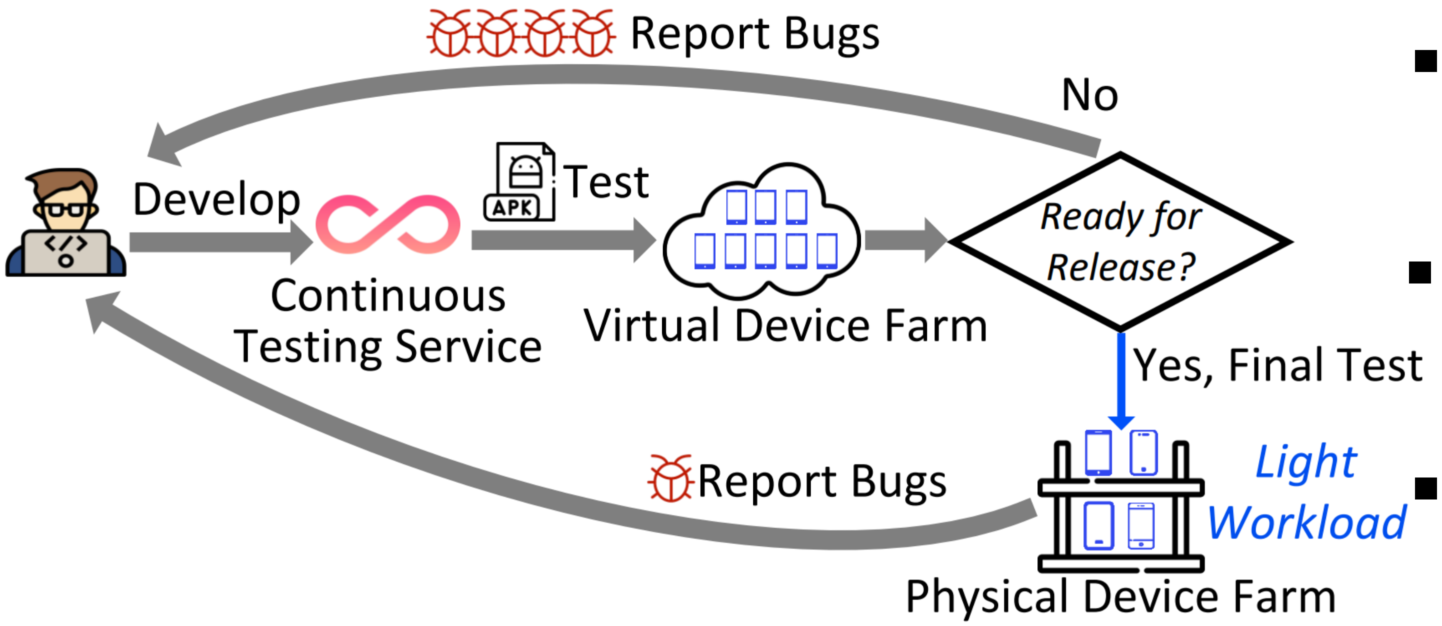


Virtual Devices for Continuous Testing

❑ Reshaping the testing infrastructure of Douyin



- Traditional physical-based mobile app testing infrastructure



- Modern continuous integration and deployment (CI/CD) pipeline
- **Continuously tests every code change** on virtual devices first
- Testing efficiency: **40%↑**
  Total operation cost: **3x↓**

Virtual Devices as a service (VDaaS)

- Recently started to share the virtual device farm as a service
  - Targeting individual or startup developers
- Feedback from preliminary users
  - **28 apps** were tested From Jan. 1st to Feb. 28th 2023
  - VDaaS helped detect 3× to 10× more bugs
  - **Most of our findings can be generalized to a broader range of apps**

Problems for future study

- Solutions for vendor-specific discrepancies
  - **Possible direction**: remoting apps' interactions (e.g., function call, system call, and I/O operation) with proprietary components to physical devices
- Developing cross-component compatibility tests
  - **Possible direction**: allow app developers to enrich CTS tests
- Issues of regional mobile app ecosystems
  - **Possible direction**: a more systematic understanding of the conflicts of interest among stakeholders

Conclusion

- ❑ A quantitative understanding of the virtual device testing fidelity
- ❑ In-depth analysis of discrepancy root causes
- ❑ Design and implementation of a high-fidelity virtual device farm
- ❑ Practices and experiences of using virtual devices to improve testing efficiency and accessibility
- ❑ Artifact: https://github.com/Android-Emulation-Testing/emu-fidelity-ae