技术洞察：面向安卓应用热启动的虚拟机GC和换页系统协同优化

部门：语言虚拟机实验室
作者：唐博文
日期：2024.11.28

Security Level:

HUAWEI

目录

1. 论文1-Marvin *(End the Senseless Killing: Improving Memory Management for Mobile Operating Systems)*
   - 背景 & 动机
   - 核心思路和关键技术
   - 实验评估

2. 论文2-Fleet *(More Apps, Faster Hot-Launch on Mobile Devices via Fore/Background-aware GC-Swap Co-design)*
   - 背景 & 动机
   - 观察
   - 核心思路和关键技术
   - 实验评估

3. 总结

HUAWEI

USENIX Annual Technical Conference *(ATC) 2020*

End the Senseless Killing: Improving Memory Management for Mobile Operating Systems

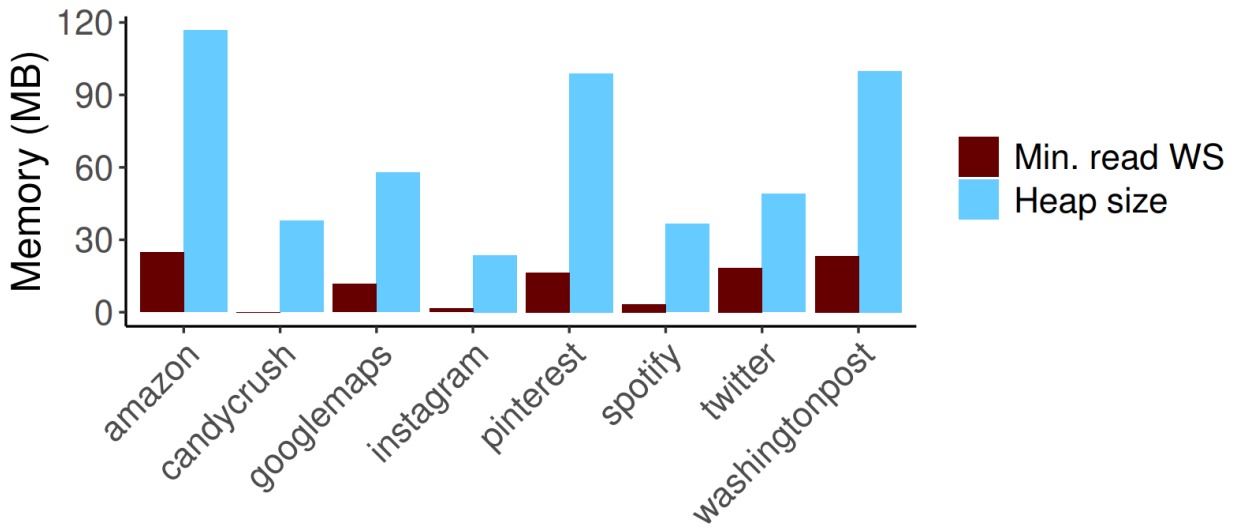
Niel Lebeck[1], Arvind Krishnamurthy[1], Henry M. Levy[1], Irene Zhang[2]

1 *University of Washington*
2 Microsoft Research

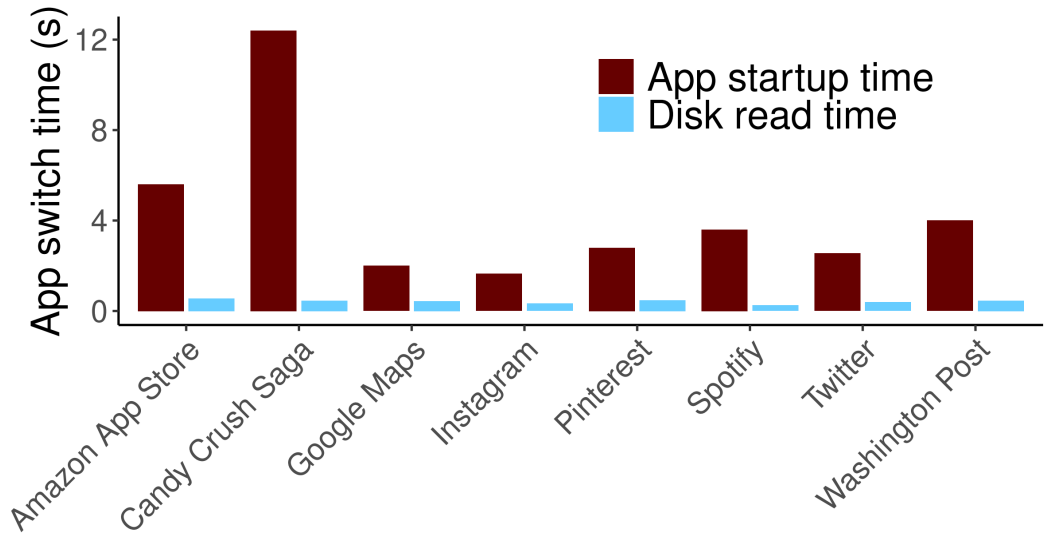
URL: https://github.com/UWSysLab

背景 & 动机

- **移动应用面临的问题：**
  - 每个应用都占用固定的大块的内存，但工作集（Working Set）只占一小部分
  - 用户切换应用时，系统会杀死前台应用，再重新启动后台应用
  - 应用需要增加生命周期管理代码，确保被杀死时能保存状态，以及被重启时能再恢复状态



应用占用的堆空间（RSS）和工作集对比

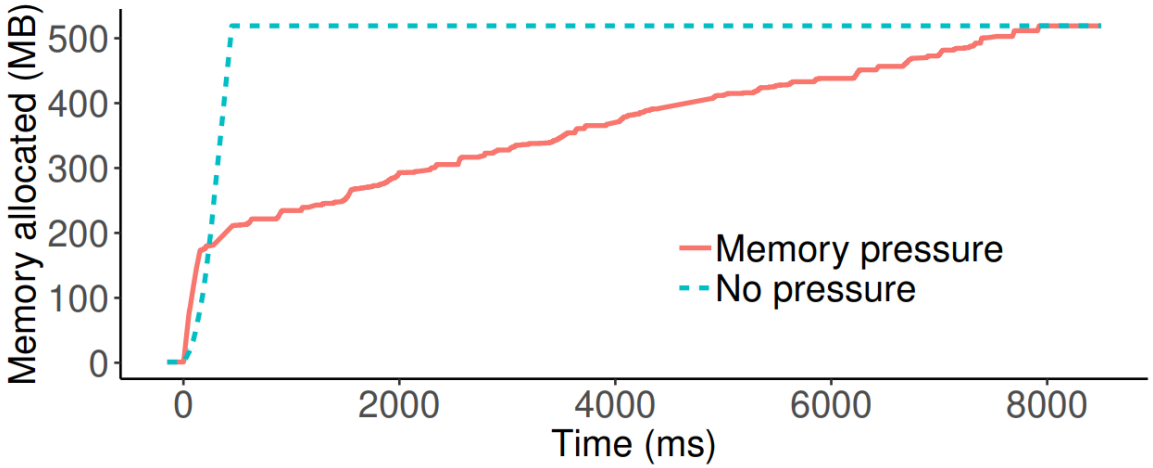


应用启动时延和磁盘换入时延的对比

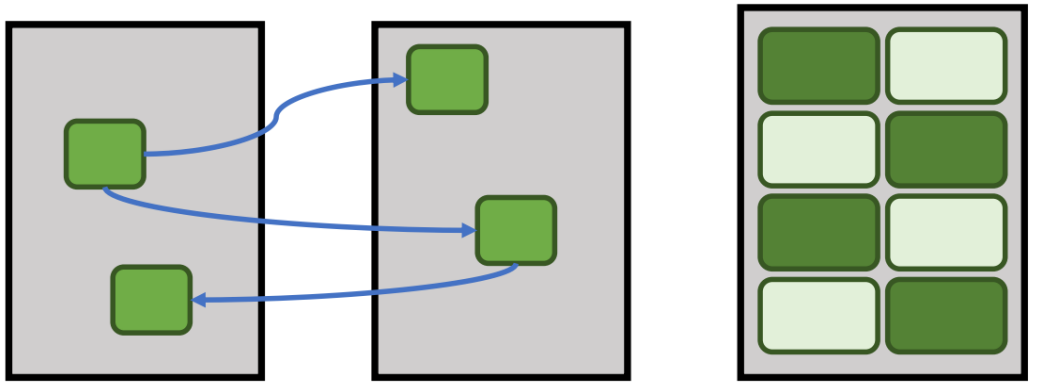
HUAWEI

背景 & 动机

- **应用热启动面临的一些挑战：**
  - 热启动（Hot Launch）通过进程保活+系统换页，绕过冷启动的开销
  - 传统的换页系统需要花费几秒钟换出前台应用的内存，才能获得足够内存来换入后台应用的内存
  - 换页系统是以页为粒度，并不感知应用的object，而虚拟机GC可能对GC带来频繁扰动



内存分配时延（包含换出和换入）

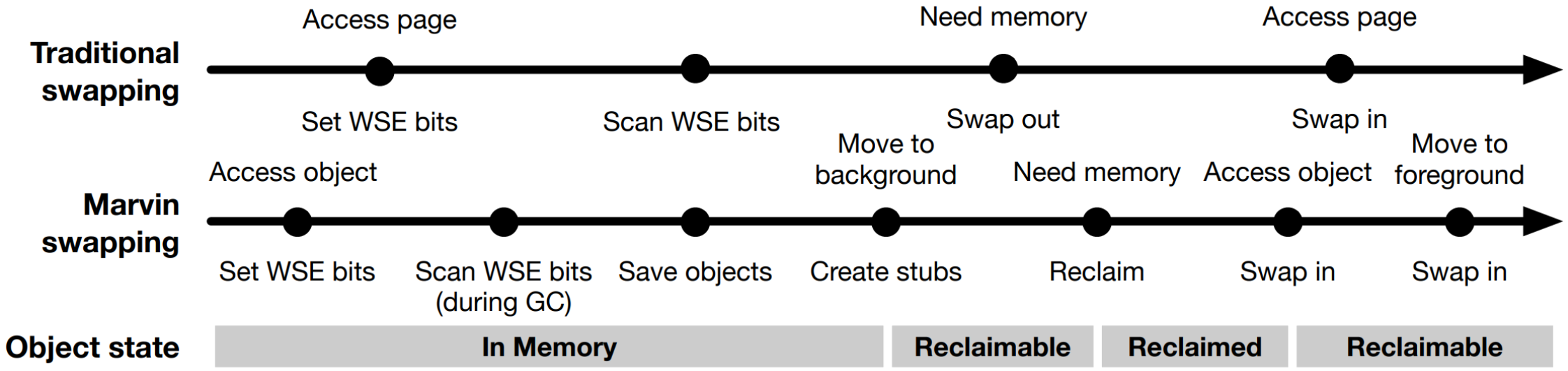


虚拟机GC回收遍历会对换页效果带来扰动

HUAWEI

核心思路和关键技术：

- **虚拟机和换页协同优化：**
  - 对象级的工作集估测（Object Level Working Set Estimation）
  - 书签GC（BookMarking GC，Hertz 05’）
  - 预换页（Ahead-of-time Swap）

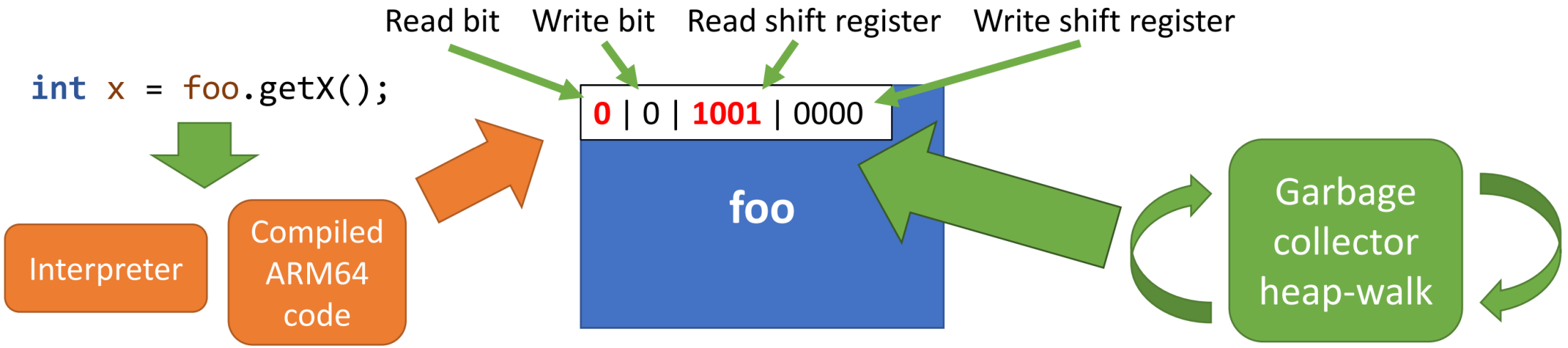


工作流程对比：传统GC换页 vs Marvin系统

HUAWEI

核心思路和关键技术：

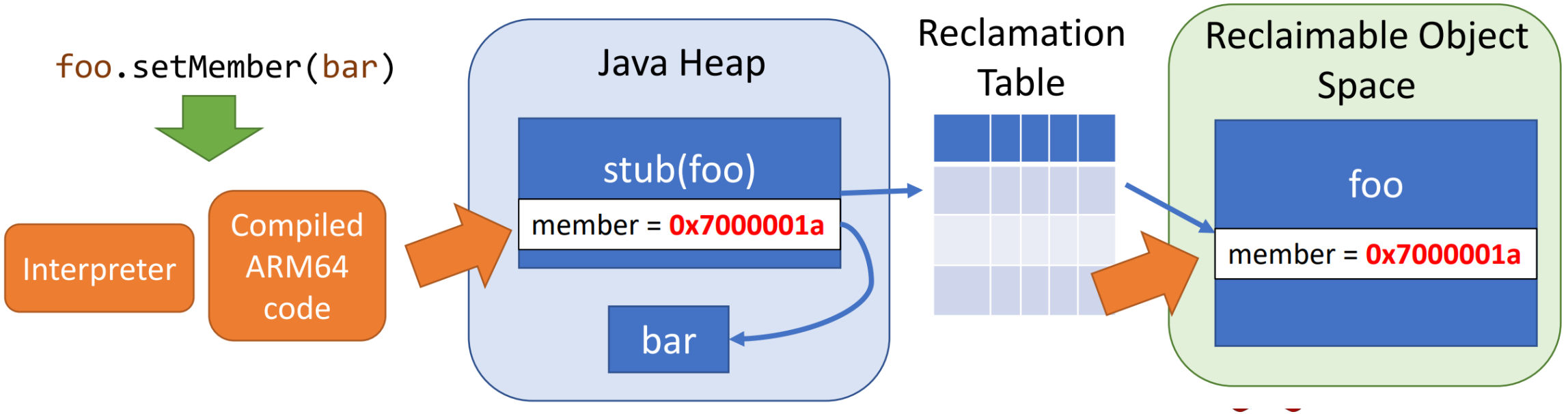
- **对象级的工作集估测（Object Level Working Set Estimation）：**
  - 通过对象访问插桩（Object Access Interposition）完成估测
  - Object头部增加2-bit记录读写情况（修改JDK和ART）
  - Object头部增加两个4-bit的移位“寄存器“，记录前4次扫描的结果
  - 应用前台时，异步线程定时全量扫描堆，将object头部读写情况移入“寄存器“并清零



HUAWEI

核心思路和关键技术：

- **书签GC（BookMarking GC，Hertz 05’）：**
  - 应用切到后台后，识别出冷对象
  - 冷对象移入待回收区域（页对齐）
  - 创建冷对象的替代对象（Stub Object），并修正引用关系
  - 后续GC遍历时只需要遍历替代对象，无需访问真正的对象，避免

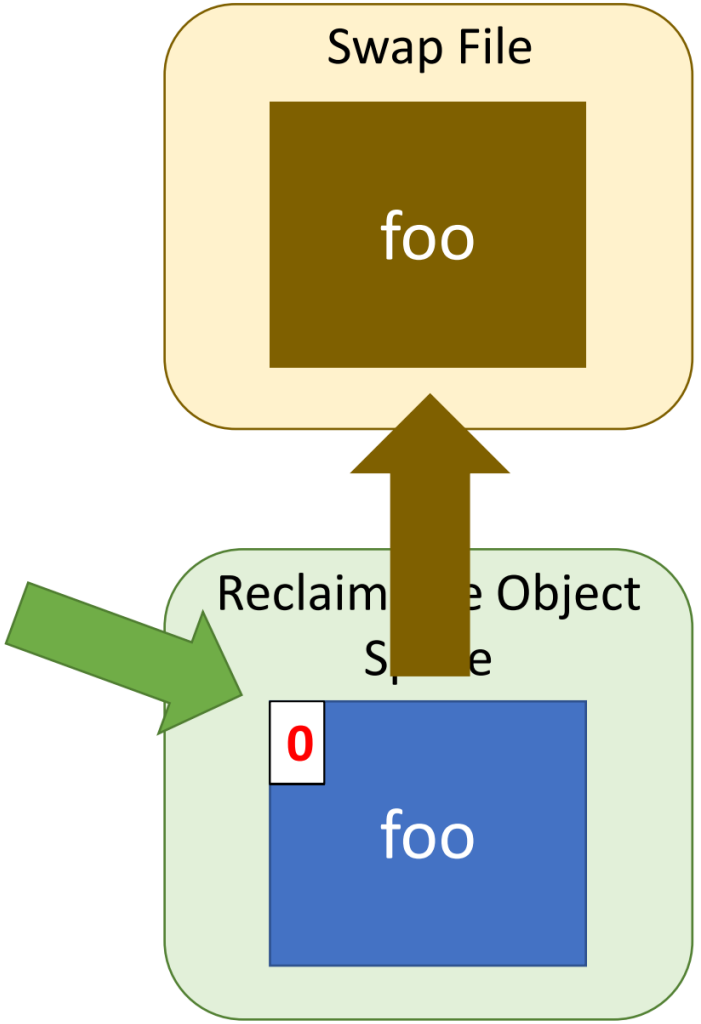


核心思路和关键技术：

- **预换页（Ahead-of-time Swap）：**
  - 应用被切到后台时，扫描待回收对象表（Reclaimation Table）
  - 将RT里面的对象写入磁盘备份文件，实现备份和真正回收解耦
  - 如果前台应用申请内存时，可以直接选中备份过对象所在的页
  - 如果后台应用被切回到前台时，根据stub更新相关对象的引用

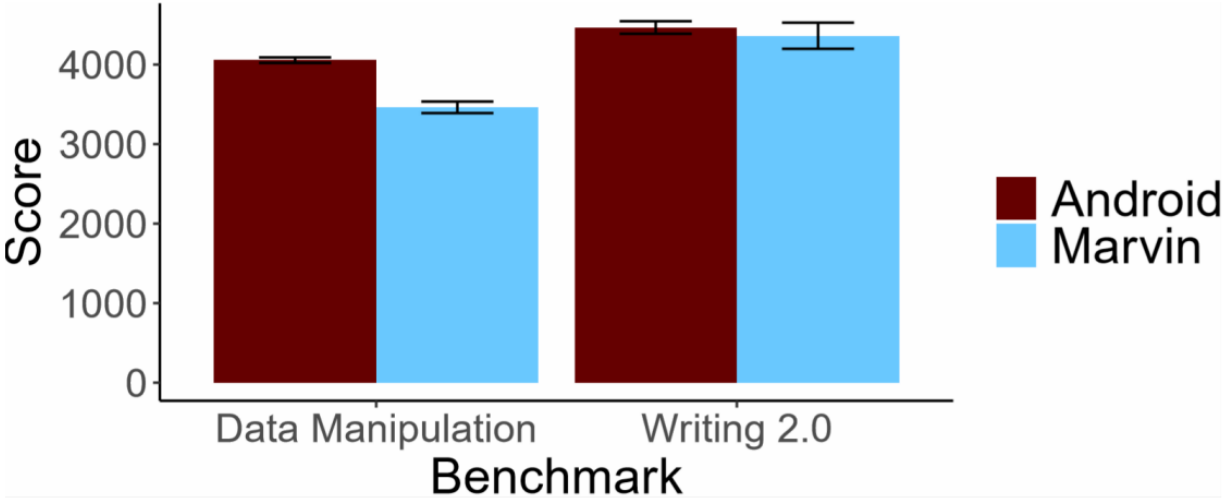
Reclamation Table

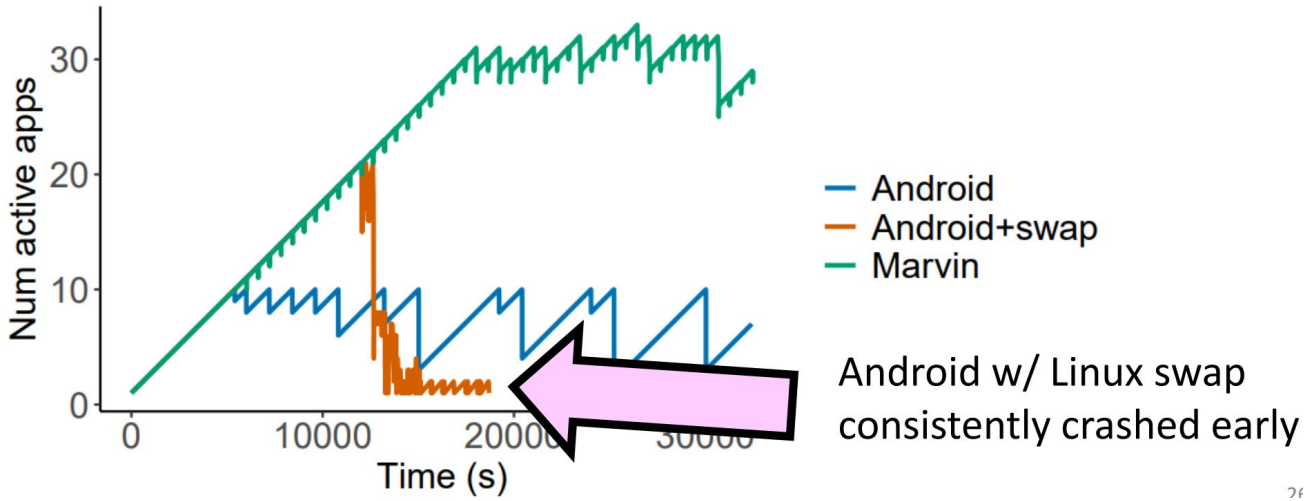
address	size	res	app lock	kernel lock
0xc00d00a0	512	1	0	0
0xc00e1410	128	1	2	0
0xc0002320	8192	1	0	0

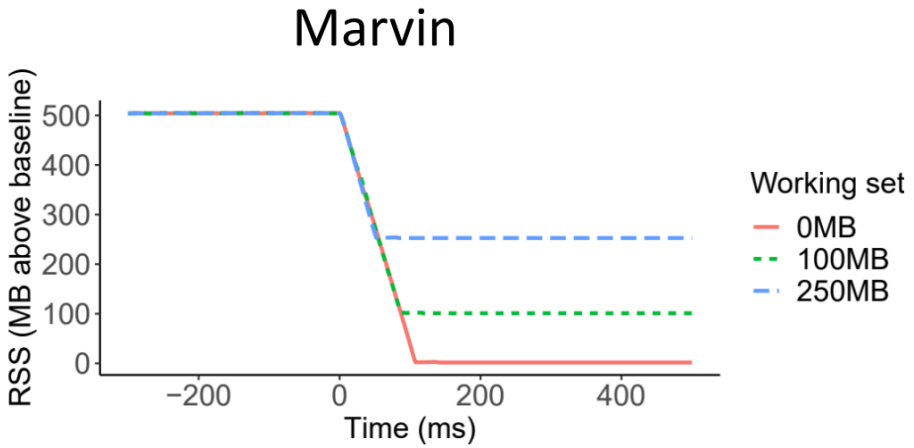


实验评估：

- **收益和开销：**
  - 同时运行应用的数量提升**2X**
  - 内存回收可在**100ms**左右达到WS的规模
  - 运行时开销不超过**15%**







HUAWEI

ACM International Conference on *Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2024*

More Apps, Faster Hot-Launch on Mobile Devices via Fore/Background-aware GC-Swap Co-design

Jiacheng Huang[1,2], Yunmo Zhang[1], Junqiao Qiu[1], Yu Liang[3], Rachata Ausavarungnirun[4], Qingan Li[2], Chun Jason Xue[5]

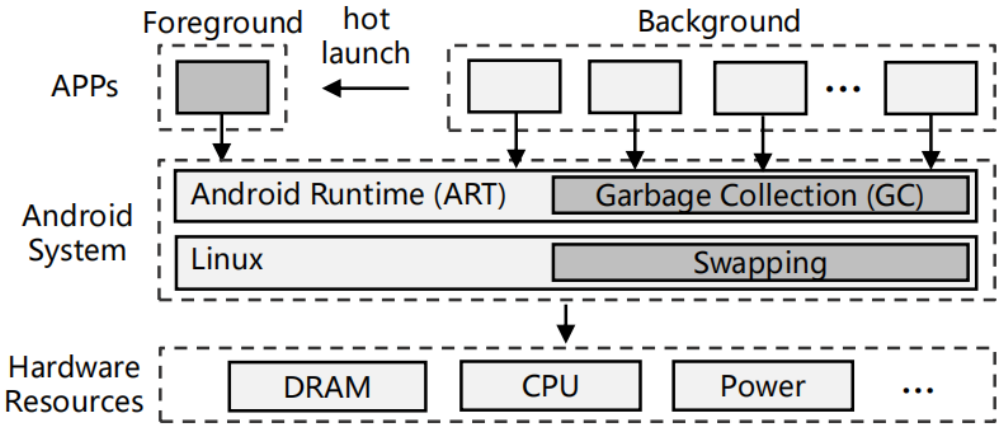
1 *City University of Hong Kong*
2 Wuhan University
3 ETH Zürich
4 King Mongkut's University of Technology North Bangkok
5 Mohamed bin Zayed University of Artificial Intelligence

URL: https://github.com/ jiachengh/Fleet

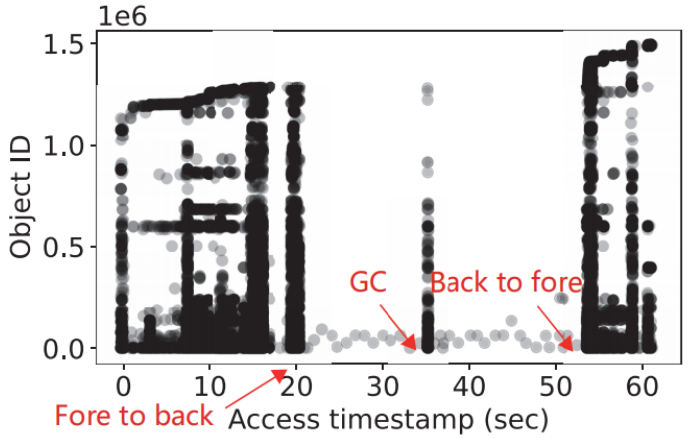
HUAWEI

背景 & 动机

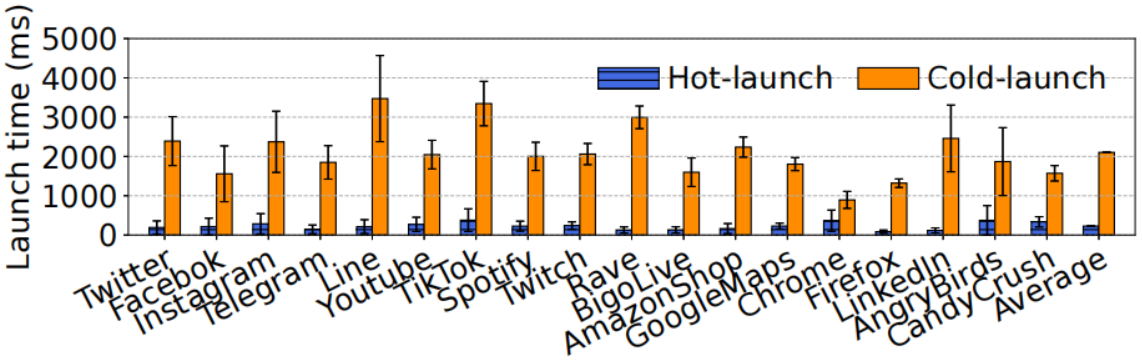
- 热启动（Hot Launch）原理

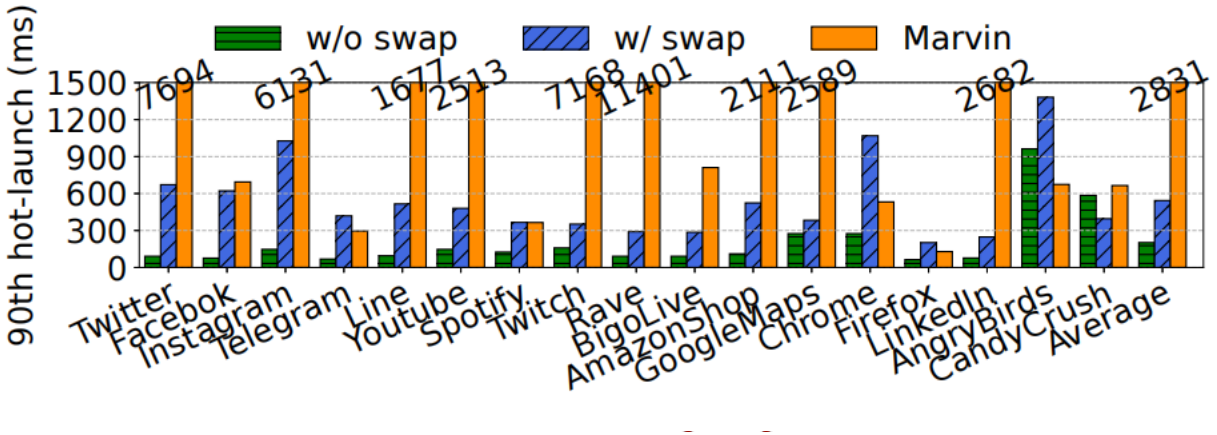


- 热启动存在的问题：
  - GC和swap存在冲突 —> Marvin (ATC’20)
  - Swap本身会影响热启动性能



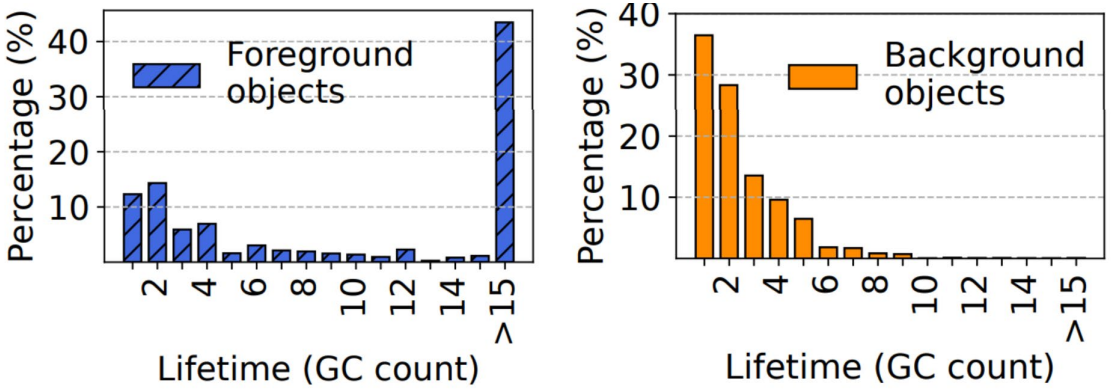
- 热启动 vs 冷启动存在8.75X性能差异



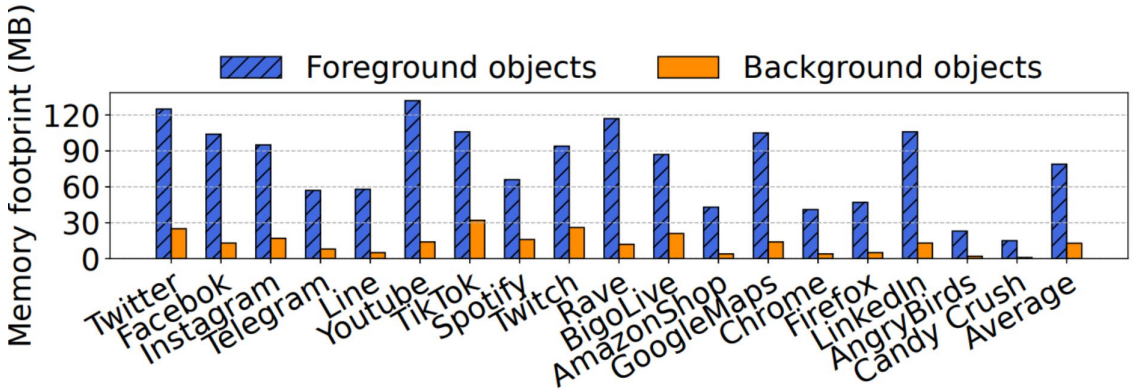


观察

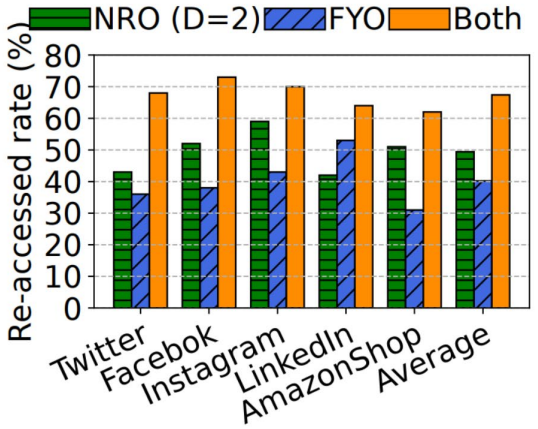
- **前台对象（Foreground Objects，FDO）vs 后台对象（Background Objects，BGO）**
  - FDO生命周期明显长于BGO
  - FDO占内存中的多数

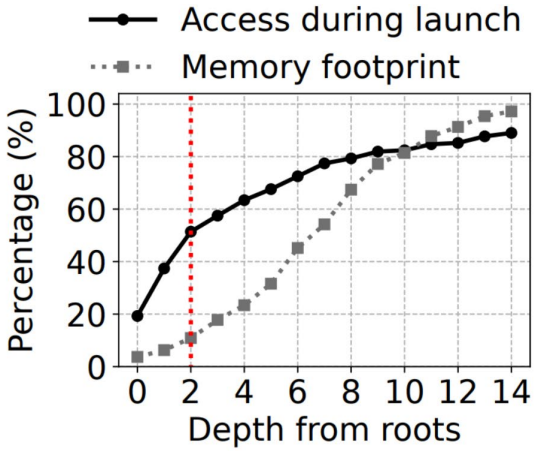


(a) Foreground object lifetime (b) Background object lifetime



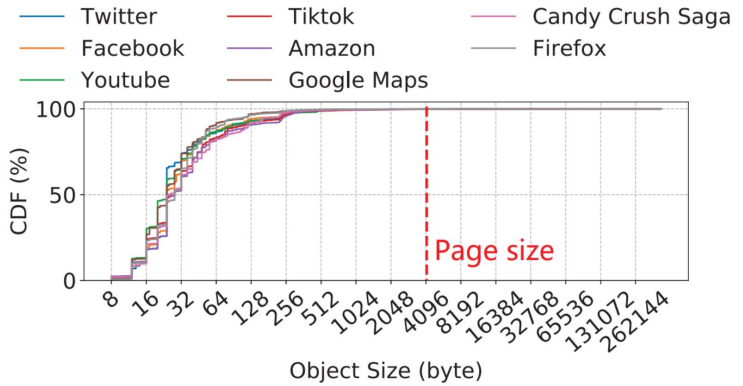
- **近根对象（Near Roots Objects，NRO）vs 前台年轻对象（Foreground Young Objects，FYO）**





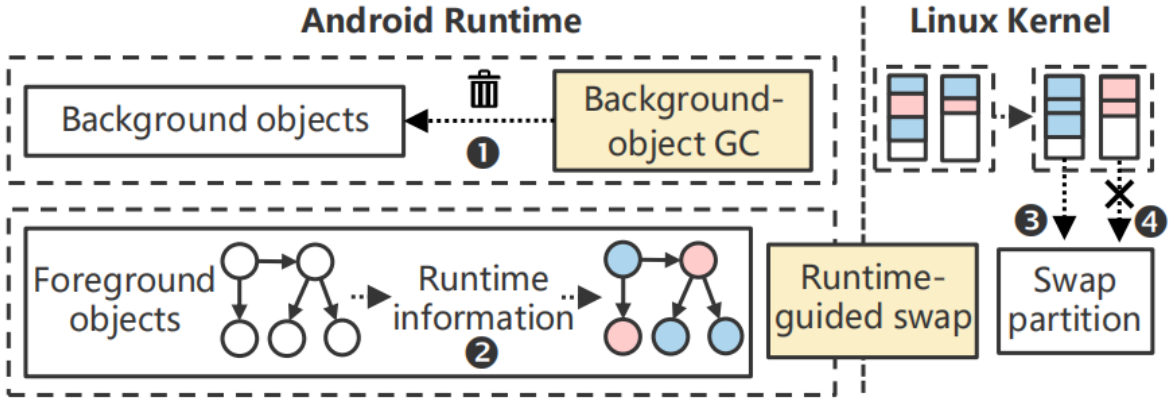
(a) Re-accessed objects (b) Depth analysis of NRO

- **绝大部分对象体积都非常小**



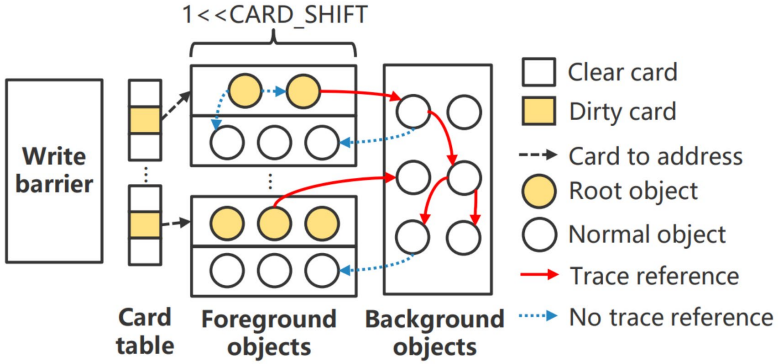
核心思路 & 关键技术

- BGO比FDO生命周期更短，BGO更可能是垃圾，所以后台GC可以只考虑回收BGO对象
- FDO内存占比很大，必须对其进行换页，换出非NRO和FYO的对象更合适
- 考虑到app后台运行的需要，后台工作对象（Working Set Objects，WSO）也不应被换出

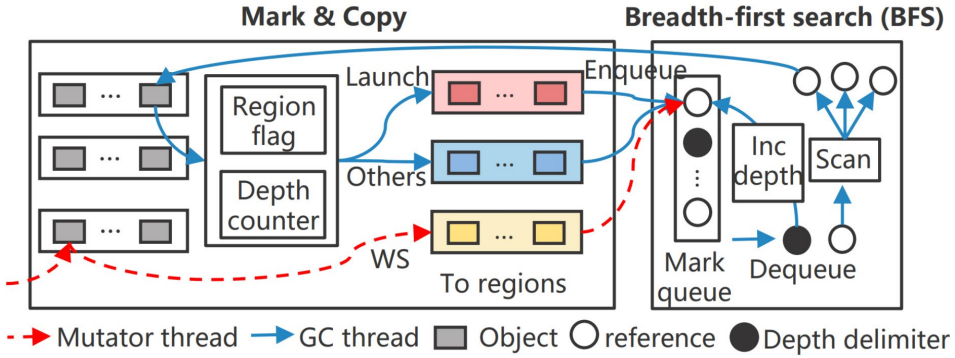


Fleet整体架构：BGC + RGS

- BGC
  - 首次BGC时做全量GC，分离BGO和FDO
  - 利用write barrier，建立FDO region到BGO region的Card-Table



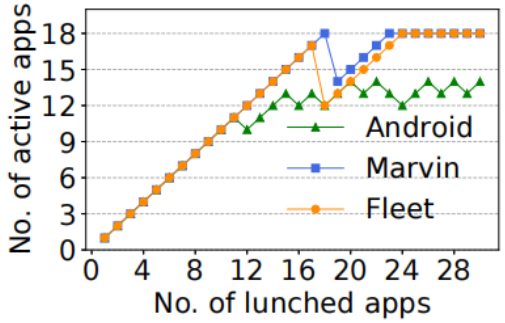
- 识别并区分NRO、FYO、WSO：通过BFS识别NRO；通过原有flag识别FYO；利用read barrier识别WSO



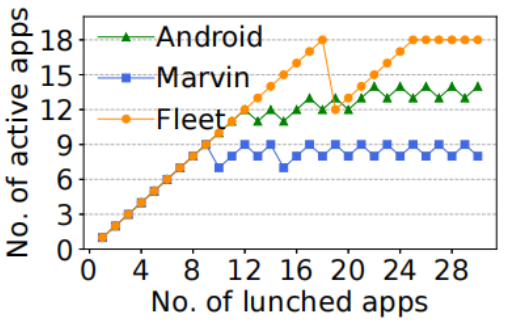
- RGS
  - 修改madvise系统调用：修改LRU策略，增加对HOT_RUNTIME和COLD_RUNTIME两种新类型内存页的考虑

实验评估

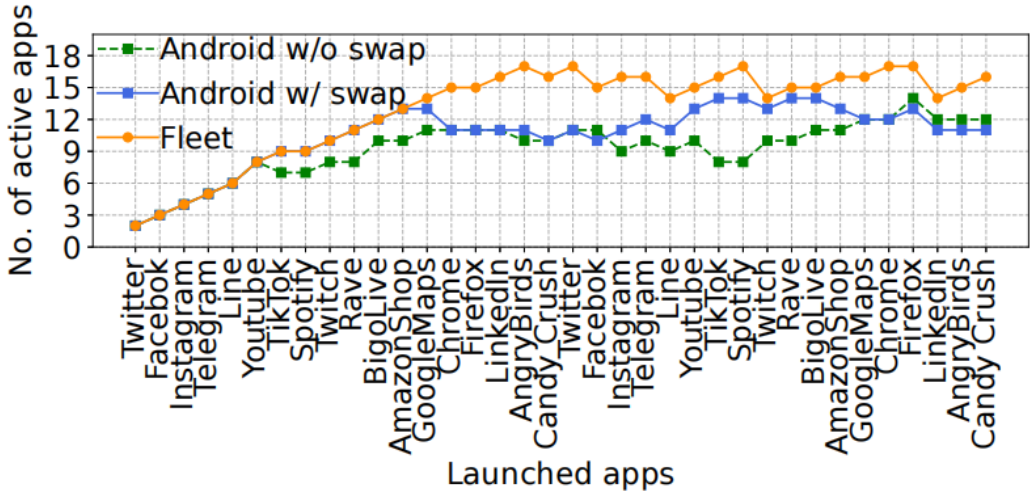
- 热启动承压能力



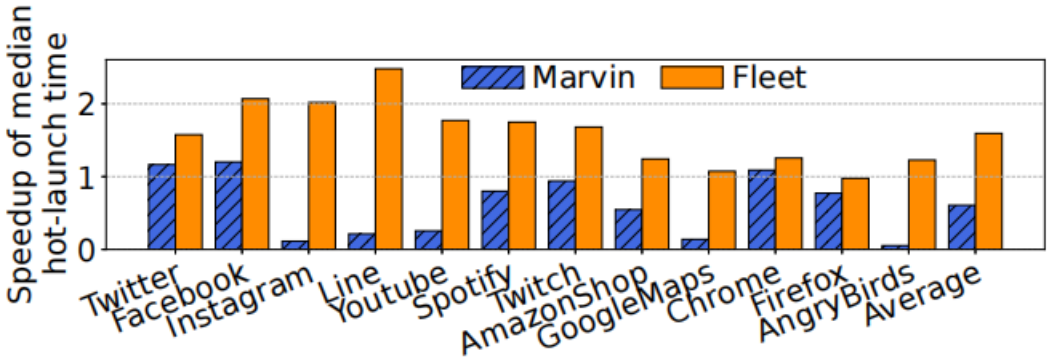
(a) Large object apps



(b) Small object apps

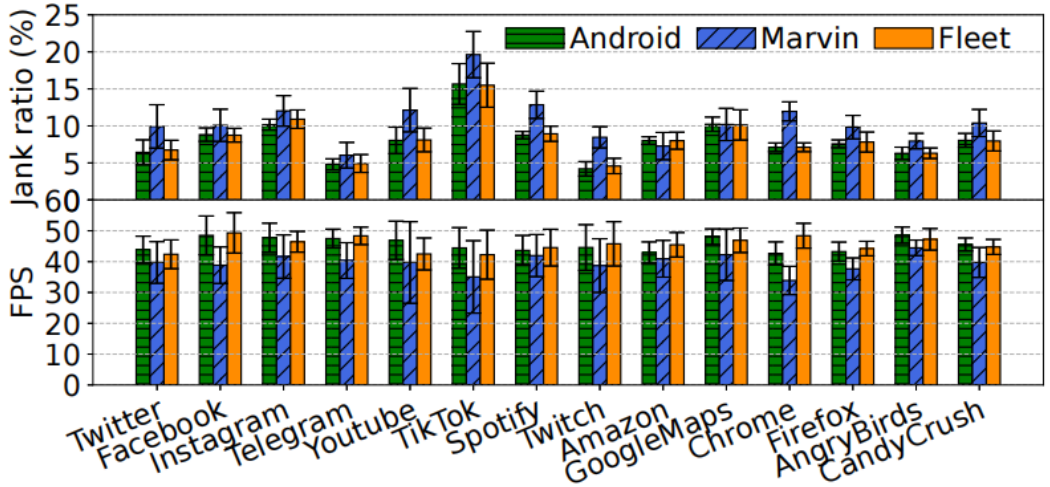


- 启动性能



(m) Median (50th) hot-launch time over Android

- 实际体验（丢帧率和滑动帧率）



总结

- **Marvin：**
  - 热启动是解决移动系统应用启动时延的关键技术
  - 热启动依赖虚拟机GC和换页系统密切配合
  - 虚拟机可以较为准确地评估出工作集，辅助换页系统
  - 换页机制中：页保存和被重用可以解耦，提前完成页保存可以节省

- **Fleet：**
  - 相比于是否在工作集中，分配在前/后台更能体现出对象的冷热程度
  - 近根对象、年轻前台对象、后台工作集对象都属于热对象，不适合被回收
  - 相比于Marivin估测工作集的方式OAI，Fleet只需要在全量GC时用read/write barrier即可完成冷、热对象的分离，极大改善了运行时开销

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI