Poster: META: Memory Exploration Tool for Android Devices

Nisarg Parikh
L.D. College of Engineering
Ahmedabad
nisargnparikh@gmail.com

Varun Gohil
Indian Institute of Technology
Gandhinagar
gohil.varun@iitgn.ac.in

Manu Awasthi
Indian Institute of Technology
Gandhinagar
manua@iitgn.ac.in

ABSTRACT

Handheld devices are becoming increasingly common in today's world. With every passing year, the diversity of applications (apps) being supported by mobile platforms is growing manyfold. In addition, Android, the most popular handheld OS in the market is releasing a new version every year, with a newer and richer set of APIs, enabling the next generation of feature-rich applications.

To support these apps, hardware requirements of these devices are increasing rapidly. Mobile SoCs need a larger number of faster cores, better GPUs and above all, higher DRAM capacities to do justice to user experience. To augment capacity requirements, non-volatile memories (NVMs) have been proposed as a potential addition to LPDDR variants, which have been the mainstay of mobile SoCs. However, few tools exist to carry out architectural design space exploration of main memory hierarchies featuring NVMs for newer Android and app versions. In this paper, we present META, a trace based tool for facilitating the exploration of memory hierarchies in mobile devices. META uses an enhanced version of Android emulator for generating raw instruction traces. These traces are then fed into a cache hierarchy and memory simulation modules to carry out design space exploration for a wide variety of apps and memory technologies.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computer systems organization** → **Architectures**;

KEYWORDS

Memory Hierarchy; Simulation; Computer Architecture

MobiCom '18, October 29-November 2, 2018, New Delhi, India

ACM ISBN 978-1-4503-5903-0/18/10.
https://doi.org/10.1145/3241539.3267737

ACM Reference Format:
Nisarg Parikh, Varun Gohil, and Manu Awasthi. 2018. Poster: META: Memory Exploration Tool for Android Devices. In *The 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18), October 29-November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3241539.3267737

1 INTRODUCTION

Mobile computing has grown substantially over the past decade with more than 2 billion mobile devices being used today [11]. Despite the rapid pace of advancements, and adoption of devices in emerging economies, understanding the architecture of handheld devices, especially from an application's perspective is lacking. Software updates for handhelds have been happening very frequently – a new version of the popular OS for handheld devices, Android, is now being released every year [1] with each version having significant upgrades over the last. For example, Android Oreo had a complete redesign of entire OS stack to make it more modular. Incorporating these changes have also led to newer, feature rich applications, as well as changes in the access patterns of existing applications.

Mobile applications are expanding to have large memory footprints and demand more resources. A report from PwC expects the capacity requirements of mobile DRAM per device to increase by 170x every five years, starting 2010 [6]. With DRAM being a primary contributor to SoC energy consumption [5], such dramatic increases in memory capacity requirements are bound to increase the power budget of the SoC, leading to much faster draining of the battery. Hence, newer architectures for providing higher memory capacities with lower energy consumption have to be devised.

Despite the growing importance of these devices, and the numerous challenges being faced for architecting these, there is a significant lack of research in this domain. A recent report [11] compared the number of papers published in leading conferences that have focused on exploration of mobile systems, and found the number to be significantly lacking, as compared to server architecture papers.

A major reason for this discrepancy is the lack of tools for carrying out architectural level exploration, benchmarking,

and evaluation. Exploratory studies are done using tools like full system architectural simulators, which are able to run the most recent versions of the operating systems and applications. The server architecture exploration space is replete with a number of these tools, capable of carrying out wide range of analyses from statistical, event-queue based simulation, to system-call emulation modes, all the way to full system simulation [4] and emulation. There are a wide variety of tools available for modeling individual subsystems, from pipelines, to caches, DRAM, and SSDs.

Such a variety of tools is sorely missing for handheld devices. The few available ones [8, 10] are not being maintained actively, and have a dwindling user base. Moreover, these tools are outdated - none of them support the newer versions of Android (> 6), or the associated apps.

In this paper, we target issues associated with exploring a very specific part of the entire handheld SoC design space - the main memory subsystem. We propose to fill in the gap in exploration mechanisms by developing a tool that can evaluate memory hierarchies for handhelds quickly, and keep pace with ever-upgrading system and application software.

2 BACKGROUND AND MOTIVATION

The current state of the art simulation tools in the handheld SoC space rely on full system simulators which suffer from multiple drawbacks. We enumerate some of the most important ones to make a case for our tool's design. First, existing tools can simulate relatively older versions of Android. Some platforms like Gem5 support creation of disk images that can boot newer Android versions and run new apps. However, this is a tedious process which doesn't always work because of compatibility issues. Disk images for contemporary SoCs booting newer Android versions have also not been made publicly available, limiting usability.

Second, using older OS versions results in experimentation with older versions of apps, which means that the design space is being explored for older applications, and not the contemporary or future ones.

Third, full system simulators are very slow compared to native execution [12]. With detailed, full system simulation models, researchers can only simulate a few seconds of actual application execution. Since applications on handheld devices involve a significant user interaction, longer activity periods need to be simulated to get a better picture.

Fourth, every release of Android leads to a large number of API changes and updates, as well as the release of new APIs [2]. As a result, the behavior of existing applications also changes. This information needs to be fed back to the architectural evaluations for the next generation SoCs. In addition, new APIs lead to release of new applications with different behaviors.

Finally, one of the goals of META is to build a toolchain that will help explore the **memory hierarchy of handheld devices**. Existing tools [4, 10] are generic and are designed to explore a number of features in the handheld devices, including core micro-architecture, caches, coherence protocols and many other things. The focus of this tool is going to be explicitly on *quickly and comprehensively* exploring main memory hierarchies for handhelds.

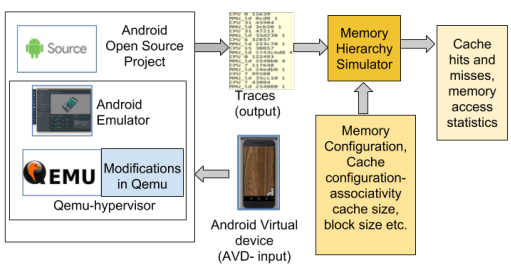
3 RELATED WORK

The space for architectural exploration tools for mobile device is limited. Of the three main choices, GemDroid is a comprehensive simulation framework for exploring SoC design space, targeted specifically for mobile systems [10]. It helps to capture system level interaction between multiple accelerators, cores and the operating system. The shortcoming of the GemDroid lies in the fact that it supports only Android 4.4 (KitKat) operating system. MofySim is a yet another simulation framework for energy consumption and performance analysis of mobile systems [8]. It is a setup with built-in benchmarks to perform prerecorded actions on user applications. Both these tools support *only* the much older, Android KitKat (4.4). QSim [7] is a framework that provides an API layer on top of QEMU which provides callbacks for extracting instruction-level execution information from QEMU. It boots a modified Linux kernel and supports ARM64 and x86-64 ISAs. Booting Android OS on top of QSim requires integration of Android and QEMU source code by performing certain modifications in both. This is a difficult task and has already been done by Google for building the Android emulator on top of QEMU.

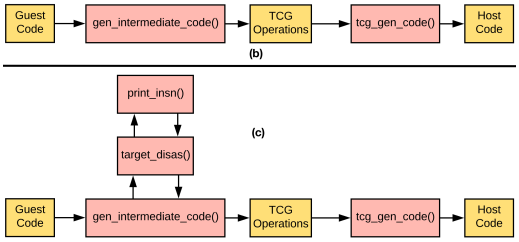
4 DESIGN OF META

META is built on top of Android Emulator, shipped with the Android Open Source Project (AOSP). This emulator, is built using a modified version of the QEMU virtualizer and emulator [3]. QEMU and AOSP were chosen because this combination serves the intent of designing this tool - quick compatibility with future versions of Android. The emulator works on Android Virtual Device (AVD), which describes the hardware profile, system image, storage areas etc. of the device being simulated. For future compatibility, newer AVDs will have to be provided, which are typically shipped with AOSP, making the integration much smoother.

We have modified the QEMU code shipped with AOSP to generate instruction level traces of the complete system, including the app being executed. These traces are then post processed via a cache and memory hierarchy simulator for understanding the application's memory behavior, as well as exploring the range of memory architectures using DRAM and NVMs. This tool, unlike QSim, supports both 32 and



(a) Different components for tool flow



(b) QEMU code structure (c) Modified QEMU code

Figure 1: Proposed Tool Flow

64-bit versions of ARM ISA. The tool consists of two modules: the Tracer Module and Memory Simulation Module. Figure 1a describes the flow of the tool. Next, we describe the functionalities of each of these modules.

4.1 Tracer Module

Every computer program on execution gets converted to assembly instructions before getting converted to machine code. The main purpose of the tracer module is to generate a comprehensive trace of all instructions executed by the emulator, including memory accesses, branch instructions, arithmetic instructions etc. The traces include the instructions executed by the app, as well as the OS.

Figure 1b shows the general workflow of QEMU. The guest instructions, executing within the emulator, are translated to an intermediate representation called TCG Operations using a function called `gen_intermediate_code()` . This intermediate representation is then translated to the host instructions by yet another function `tcg_gen_code()` [3]. We modify this codepath to derive the traces, as depicted in Figure 1c. During translation of guest instructions to TCG operations, the control is passed to a new function, `target_disas()` which disassembles the instruction. This calls `print_insn()` to redirect the decoded instructions to a trace file.

4.2 Memory Simulation Module

The generated traces are then fed to the cache simulation module, for exploration of on-chip cache hierarchies. The memory simulation module (MSM) takes as input a configuration file that describes the cache hierarchy. For example, for each level, it specifies the size, associativity, line size, replacement policies etc. among other things. The configuration file also specifies the relationships between the different levels of the hierarchy. Currently, every instruction in the trace, other than a memory instruction is ignored by MSM.

At the end of the run, the cache simulation module of the MSM provides two things (i) statistics about the cache characteristics like hit rate, cache misses, etc., and (ii) another trace file that can be fed to a main memory simulator (MMS).

5 CONCLUSIONS AND FUTURE PLANS

We present the design of META, a trace based tool for rapid exploration of memory hierarchies in Android based mobile devices, comprising of trace generation, cache hierarchy and main memory simulation modules capable of modeling DRAM, and non-volatile memory technologies like PCM and STT-RAM, by integrating already existing tools [9]. META will be able to carry out rapid analyses of the main memory design space for both conventional (DRAM), non-conventional (PCM, STT-RAM etc.) and hybrid / heterogeneous memory systems for handheld devices.

ACKNOWLEDGMENTS

This research is supported in part by SERB grant ECR/2017/000887

REFERENCES

[1] 2017. Android - History. https://bit.ly/2hglRnB
[2] 2018. Android 8.0 Features and APIs. https://bit.ly/2Mq7iy0
[3] Fabrice Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of USENIX ATC*.
[4] Nathan Binkert et al. 2011. The Gem5 Simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011).
[5] Aaron Carroll and Gernot Heiser. 2013. The Systems Hacker's Guide to the Galaxy Energy Usage in a Modern Smartphone. In *APSYS*.
[6] Raman Chitkara. 2012. Mobile Technologies Index Memory: The ever-predictable DRAM path. https://pwc.to/2LpCFsd
[7] Kersey Chad D et al. 2012. A universal parallel front-end for execution driven microarchitecture simulation. In *Proceedings of RAPIDO*.
[8] Minho Ju et al. 2016. MofySim: A mobile full-system simulation framework for energy consumption and performance analysis. In *ISPASS*.
[9] Poremba Matthew et al. 2015. Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems. *IEEE CAL* 14, 2 (2015).
[10] Chidambaram Nachiappan et al. 2014. GemDroid: A Framework to Evaluate Mobile Platforms. In *Proceedings of SIGMETRICS*.
[11] V. J. Reddi, H. Yoon, and A. Knies. 2018. Two Billion Devices and Counting. *IEEE Micro* 38, 1 (January/February 2018), 6–21.
[12] Ali Saidi and Andreas Sandberg. 2012. gem5 Virtual Machine Acceleration. https://bit.ly/2PplipV