A Study of Thread Level Parallelism on Mobile Devices

Cao Gao[*], Anthony Gutierrez[*], Ronald G. Dreslinski[*], Trevor Mudge[*], Krisztian Flautner[†] and Geoffery Blake[†]

[*]Advanced Computer Architecture Laboratory, University of Michigan, {caogao, atgutier, rdreslin, tnm}@umich.edu
[†]ARM Ltd., {krisztian.flautner, blakeg}@arm.com

***Abstract*—Mobile devices continue to increase the number of cores in an attempt to meet the needs of performance-demanding applications. However, the increasing number of cores does not necessarily translate into performance gain and/or power reduction. In this paper we investigate how multi-core mobile devices are utilized by applications. Our results demonstrate that mobile applications are utilizing less than 2 cores on average, which shows that multi-cores are generally underutilized by today's mobile applications. Unless application developers can significantly improve core utilization, further increasing core counts will result in little gain.**

I. Introduction

Given the growing hardware demand from modern mobile applications, mobile devices vendors have started shipping smartphones and tablets embedded with multi-core CPUs in volume. However, despite the great computation potential that resides in multi-core CPUs, it is not clear how much they can be utilized for mobile devices. In order to take advantage of a multi-core system, software developers have to divide their program into parallelized threads, which is difficult. In a similar desktop situation, Blake et al. [2] performed a study on a suite of representative desktop applications. Their results suggested that the number of cores that can be profitably used are less than 3 for most commonly used applications.

In this work, we analyze a broad range of popular mobile applications on two up-to-date development boards to determine how the cores are utilized on mobile devices. We calculate the *Thread Level Parallelism* (TLP) of these applications. Our results show an average TLP of 1.4 for a quad-core system. It suggests that mobile applications are utilizing less than 2 cores on average, even with several applications running concurrently. In fact, some recent mobile CPUs [1] are made with 2 cores and still provide the desired performance. We also measure the same metrics for a broad spectrum of configurations, including various number of cores in the system, core frequencies, and different CPUs. We observe a modest TLP scalability for most applications, and increasing the number of cores has little return on TLP. In addition, CPUs with higher frequencies tend to exhibit less TLP, which suggests that exploiting parallelism will only be more challenging in the future. In all, these studies suggest an underutilization of multi-core CPUs in mobile devices. It seems that software developers are lagging behind in exploiting parallelism in mobile applications, and increasing the number of CPU cores may have diminishing returns until that changes.

II. Methodology

A. Metrics We use *Thread Level Parallelism (TLP)* [2], [3], which is defined in Equation 1 as the machine utilization over the non-idle portions of the benchmarks execution:

$$TLP = \frac{\sum_{i=1}^{n} c_i i}{1 - c_0} \tag{1}$$

where c_i is the fraction of time that i cores are concurrently running different threads, and n is the number of cores. Specifically, c_0 is the idle time fraction. To calculate TLP, we collect all the context switch events using *ftrace*, a Linux kernel internal tracer.

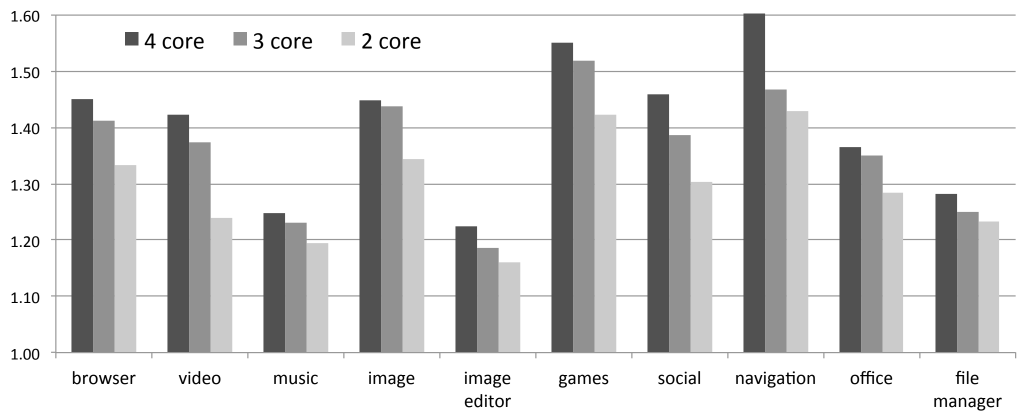
B. System setup We choose two development boards that are representative of the latest mobile device technology. Most of the experiments are done on the Samsung *Origenboard*. It contains a *Exynos 4412* SoC with a 1.4GHz quad-core Cortex-A9 CPU and Mali-400 GPU. For comparison, we also use a Qualcomm *Dragonboard* with a 2.3GHz quad-core Krait CPU.

C. Benchmarks We choose 16 popular applications from the Google Play Appstore and 4 native ones in the Android OS. This means they have a large user base and are thus representative of current mobile software. They represent applications from 10 different commonly used categories (shown in Fig. 1a). The testing actions on these applications usually last for 30 seconds and cover most typical functions of the application under test. We found 30 seconds is long enough to cover all common actions for the benchmark applications. All experiments are repeated at least 5 times, and we observe a low standard deviation of TLP results. Before testing, we kill all the running and background applications to reduce experimental errors. Besides single applications, we also choose four applications from the suite, and run them concurrently with a set of other applications in the background in order to simulate multi-tasking scenarios.

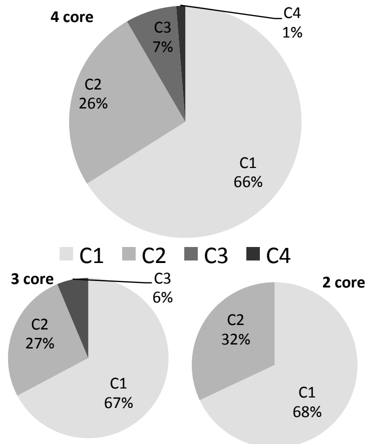
III. Results

In this section, we show that current mobile applications have a rather low TLP on modern mobile device platforms. We also observe a small return on TLP given the increase in the number of cores and less TLP for cores with higher frequencies.

We present the overall TLP results in Fig.1a. The results demonstrate that: **1) All the applications have some, but quite limited TLP.** We do see a TLP higher than 1.2 for almost all the applications under test. However, the parallelism we observed is quite low: for a 4-core system, on average, we see a TLP of **1.4**. The applications with high TLP, namely Games, Browser and Navigation, have TLPs around 1.5 to 1.6. Applications like Music and File Browser have rather low TLPs around 1.2 to 1.3. **2) Increasing number of cores has little return on TLP.** On average, TLP increases by 4.5%



(a) TLP for different application categories



(b) Average CPU time breakdown

Fig. 1: TLP result for the *Origenboard*. We change the number of active cores and repeat the same experiments. The "4 core" columns or pie chart shows the result when 4 cores are activated; the "3 core" ones are when the fourth core is shut down and the remaining 3 cores are kept activated; similarly, the "2 core" represent the result with half cores turned off. The left graph shows the TLP numbers of different categories of applications. The right graph represents a CPU time breakdown on the average of all applications: "C4" stands for the *c4* portion in Equation 1, which is the percentage of time that all the 4 cores are not idle. "C3" stands for 3 cores, etc.

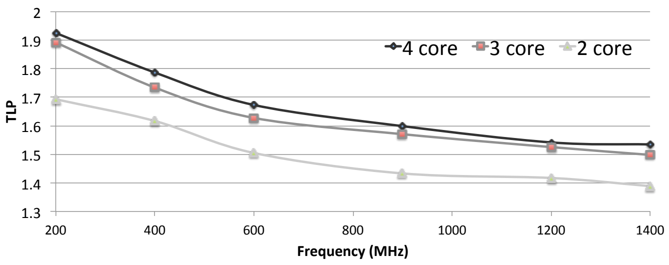


Fig. 2: TLP under different frequencies (BBench)

when we switch from a 2-core system to a 3-core system, and only 3.1% from a 3-core system to a 4-core system. Particularly, most applications hardly show any increase in TLP from 3-core to 4-core, which indicates the software does not generate enough concurrent parallel threads during its execution.

We also present a detail CPU time breakdown in Fig.1b. It shows the fraction of time that *i* cores are concurrently running threads. We observe a very low *C4* and *C3* number for all applications we tested, which means there is only a small amount of time that four or three cores are being utilized at the same time. On average, for only **1%** of the time are four cores executing threads and **7%** of time are three cores execuitng threads.

A. Frequency Scaling

Most modern mobile CPUs support *Dynamic Frequency and Voltage Scaling* (DVFS), which offers an opportunity to reduce power consumption of cores. We run BBench [4], an automatic webpage rendering benchmark, on the *Origenboard* for six different frequencies between 200MHz and 1400MHz. We show our results in Fig.2: the three lines represent system configurations with different numbers of cores activated. The results demonstrate that cores with higher frequency tend to have less TLP than those with lower ones for the same program. Nevertheless, we can see that even for a 4-core system of minimum frequency (200MHz), we still have a TLP of less than 2 for BBench. This suggests that software is still the limitng factor in exploiting TLP.

B. Alternative Architectures

We also perform the same set of TLP tests on the *Dragonboard* in order to see how a more powerful CPU would affect TLP. We observe a decrease in TLP in all applications but image editor, which remains unchanged; none of the applications has a TLP larger than 1.5. This echoes the observation that more powerful CPUs tend to have less TLP.

In all, these results imply that as individual CPU architecture designs continue to evolve and core counts increase, exploiting TLP will become harder. The CPUs will continue to be underutilized if software developers fail to produce better parallelized programs.

REFERENCES

[1] Apple's iphone 5s a7 chip is a benchmarking beast. [Online]. Available: http://www.forbes.com/sites/patrickmoorhead/2013/09/20/apples-a7-soc-is-a-benchmarking-beast

[2] G. Blake, R. G. Dreslinski, T. Mudge, and K. Flautner, "Evolution of thread-level parallelism in desktop applications," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010.

[3] K. Flautner, R. Uhlig, S. Reinhardt, and T. Mudge, "Thread-level parallelism and interactive performance of desktop applications," in *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, 2000.

[4] A. Gutierrez, R. G. Dreslinski, T. F. Wenisch, T. Mudge, A. Saidi, C. Emmons, and N. Paver, "Full-system analysis and characterization of interactive smartphone applications," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on*, 2011.