Poster: Towards Reducing Smartphone Application Delay through Read/Write Isolation

David T. Nguyen*, Gang Zhou*, Guoliang Xing†
*College of William and Mary, USA †Michigan State University, USA
{dnguyen, gzhou}@cs.wm.edu, glxing@cse.msu.edu

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies

Keywords

Smartphone App Delay; I/O Optimizations; App Launch

1. INTRODUCTION AND APPROACH

A recent analysis[3] indicates that most user interactions with smartphones are short. Specifically, 80% of the apps are used for less than two minutes. With such brief interactions, apps should be rapid and responsive. However, the same study reports that many apps incur significant delays during launch and run-time. This work addresses two key research questions towards achieving rapid app response. (1) *How does disk I/O performance affect smartphone app response time?* (2) *How can we improve app performance with I/O optimization techniques?*

We address the questions via following contributions. First, through a large-scale measurement study based on the data collected within 130 days from 1009 Android devices using an app[2] we developed, we find that Android devices spend a significant portion of their CPU active time (up to 58%) waiting for storage I/Os to complete, also known as *iowait* (Figure 1). This negatively affects the smartphone's overall app performance, and results in slow response time. Further investigation reveals that a read experiences up to 626% slowdown when blocked by a concurrent write. Additionally, the results indicate significant asymmetry in the slowdown of one I/O type due to another. Finally, we study the speedup of concurrent I/Os, and the results suggest that reads benefit more from concurrency.

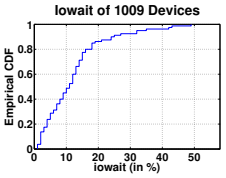

Figure 1: Iowait.

Second, we implement a system called SmartIO that shortens app delays by prioritizing reads over writes. SmartIO allows reads to be completed before writes, and delays writes as long as there are reads, while avoiding write starvation. To achieve this, a third level of I/O priority is added into the current block layer[1], assigning higher priority to reads and lower to writes. This third priority level has a lower priority than the original block layer's priorities.

MobiSys'14, June 16–19, 2014, Bretton Woods, New Hampshire, USA.
ACM 978-1-4503-2793-0/14/06.
http://dx.doi.org/10.1145/2594368.2601458.

Write starvation is avoided by applying a maximal period of time assigned to a process (100ms). Additionally, SmartIO issues I/Os with optimized concurrency parameters that are obtained during installation. The parameters include the optimal number of sequential or random reads (writes) that benefit most from concurrency.

2. PERFORMANCE EVALUATION

We evaluate our system using 20 popular apps and show that SmartIO reduces launch delays by up to 37.8%, and run-time delays by up to 29.6% (Figure 2). SmartIO's performance gain during launch is due to its read-intensive nature. Specifically, the average number of reads observed during launch on the 20 apps is five times higher than writes. The smaller performance gain during the run-time is caused by its modest I/O activity. The average number of I/Os observed during launch is 2 times higher than during run-time. SmartIO's read performance improvement comes with little cost due to the read/write discrepancy nature of the flash storage (reads take much faster to complete). Although, writes may encounter slight slowdowns. In another experiment, we install the 20 apps researched, and the writes are on average 7.6% slower with SmartIO. However, at the same time, many other processes in the background may benefit from SmartIO. Based on our large-scale study, there are on average 255 processes running on each device at any point of time, from which 98 have some I/O activity and generate a workload. These processes are expected to have faster response time with SmartIO.

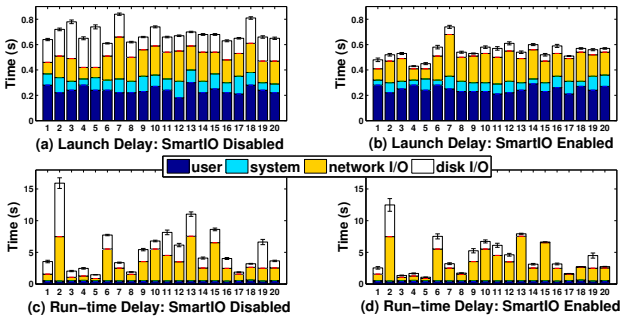

Figure 2: Launch and Run-time Delay. **1:**Angry Birds; **2:**GTA; **3:**Need for Speed; **4:**Temple Run; **5:**The Simpsons; **6:**CNN; **7:**Nightly News; **8:**ABC News; **9:**YouTube; **10:**Pandora; **11:**Facebook; **12:**Twitter; **13:**Gmail; **14:**Google Maps; **15:**AccuWeather; **16:**Accelerometer M.; **17:**Gyroscope Log; **18:**Proximity Sensor; **19:**Compass; **20:**Barometer.

3. REFERENCES

[1] Block layer. `http://goo.gl/SwdLZ5`, 2014.
[2] Storebench download. `http://goo.gl/ava9eV`, 2014.
[3] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices. In *Proc. of ACM MobiSys, 2012*.