scheduling algorithm and queue depth with optimal consumption, that combination is returned as a result of this phase. With power efficiency in mind, we want a computationally inexpensive matching approach that is at the same time precise. Having all types of I/Os coming to storage, simple intuition says that what matters most at the end are the total number of completed reads and number of completed writes in a given interval. Furthermore, it is necessary to take into consideration differences between characteristics of read and write I/Os. Some partitions will serve reads better than writes or vice versa. Some partitions will be read-only, other allow both read and write. Motivated by this, we decide to expand the simple intuition, and do the matching based on the proportions of rates of completed reads and completed writes.

For clarity, let us define *RCRate* as *number of reads completed per second.* and *WCRate* as *number of writes completed per second.* Further, let us define *Rate Proportion (RP)* as $RP = RCRate / WCRate$. If the Rate Proportion (RP) of the phone's I/O pattern is close to the RP of a benchmark, a match is found. Finally, the optimal scheduling algorithm is set in the block layer scheduler and the optimal queue depth is set in the device driver.

Performance Evaluation

This section evaluates the SmartStorage efficiency by comparing energy usage of the 20 most popular applications from the Android Market with and without SmartStorage.

In our experiments, we use the SmartStorage implementation in the Nexus One phone. To measure energy consumption, the Monsoon Power Monitor [2] is utilized. We run the experiments with the top 20 free applications from the Android Market as of August 7, 2012. During our experiments, all radio communication is disabled except for WiFi. The screen is set to stay awake mode with constant brightness and auto-rotate screen off. When SmartStorage is in use, it runs only in the background and its GUI is off.

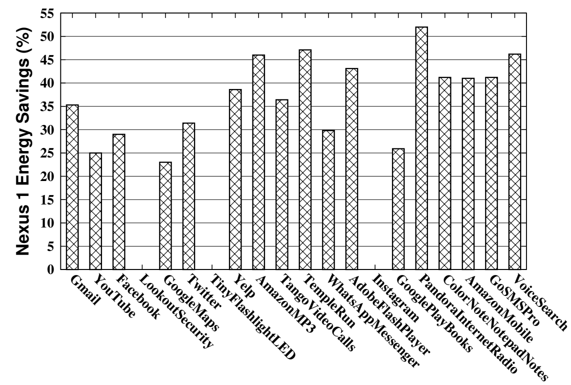

Figure 5: Energy Savings on Nexus One.

In order to address how much energy our solution saves in a typical use case, we run each of the 20 applications mentioned with SmartStorage in the background and compare with the case when the application is running with the default scheduling algorithm and queue depth (BFQ/128). A typical use case varies for applications. For instance, for Gmail, we read 20 emails and write 10 emails; for Amazon Mobile, we search for 20 products and read information about them; in Pandora, we listen to a channel for 30 minutes; on YouTube we search and listen to 5 songs; on Facebook, we read and write posts, etc. The Android Monkey tool is utilized to allow repeating the same behavior more times with and without SmartStorage so as to ensure fairness. The results of the total savings are in Figure 5. We can observe that the energy savings vary from 23% to 52% and the largest savings are with Pandora application (52%). The three applications with no energy values have the optimal configuration identical to the default parameters of the phone.

Remaining Steps

In the device driver layer, we benchmarked the phone with two different queue depths and found significant differences in energy consumption. Naturally, more research on combining queue depths and scheduling algorithms may yield higher savings. We proposed the *RP* metric that proved to be efficient at matching I/O patterns. However, we plan to research a machine learning based method in the future. Our pilot solution periodically measures the storage I/O and then matches the I/O fingerprint to that of benchmarks for locating the optimal storage policy to save energy. If this process happens too frequently, the cost may be unnecessarily high and the system may not be stable since application performance may be impacted during highly frequent storage policy transitions, which we also plan to evaluate. If such a process happens too sparsely, we will not save much energy. Hence, we plan to monitor application events such as application started and terminated, and use them to adapt the measurement and matching frequency.

The conventional wisdom is that storage contributes little (approximately 30%) to the total power consumption [5]. In our simple proof-of-concept solution, with the dynamic storage configurations, we are able to save from 21% to 52% of the total consumption. We attribute this to the performance impact of the storage on other components of the phone. We suspect that the interesting savings are triggered by the changes in the storage, and further propagated into other components in the phone. This naturally raises a question, how storage affects the performance of different smartphone subsystems. Thus, still more research needs to be done in this matter.

Acknowledgements

This work is supported in part by NSF grant CNS-1250180. The author would like to thank Dr. Zhou, William & Mary LENS research lab members, and anonymous reviewers for their valuable comments.

Biographical Sketch

David Nguyen has been working on his Ph.D. in Computer Science at the College of William and Mary since Fall 2011. He is advised by Dr. Gang Zhou, and his research interests are wireless networking and smartphone storage.

References

[1] Block i/o layer tracing: blktrace. *http://linux.die.net/man/8/blktrace*, 2012.

[2] Monsoon power monitor. *http://www.msoon.com*, 2012.

[3] Bovet, D., and Cesati, M. *Understanding the Linux Kernel, Third Edition*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2005.

[4] Brunelle, A. D. *blktrace User Guide*. USA, 2007.

[5] Carroll, A., and Heiser, G. An analysis of power consumption in a smartphone. In *Proc. ATC 2010*, USENIX Assoc. (2010).

[6] Kim, H., Agrawal, N., and Ungureanu, C. Revisiting storage for smartphones. In *Proc. FAST 2012*, USENIX Assoc. (2012).

[7] Nguyen, D. T., Zhou, G., Qi, X., Peng, G., Zhao, J., Nguyen, T., and Le, D. Storage-aware smartphone energy savings. In *Proc. UbiComp 2013*, ACM Press (2013).

[8] Pathak, A., Hu, Y. C., and Zhang, M. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proc. EuroSys 2012*, ACM Press (2012).