Exploring memory energy optimizations in smartphones

Ran Duan, Mingsong Bi, Chris Gniady
Computer Science
University of Arizona
Tucson, USA
Email: {tomduan, mbi, gniady}@cs.arizona.edu

***Abstract*—Recent development of sophisticated smartphones has made them indispensable part of our everyday life. However, advances in battery technology cannot keep up with the demand for longer battery life. Subsequently, energy efficiency has become one of the most important factors in designing smartphones. Multitasking and better multimedia features in the mobile applications continuously push memory requirements further, making energy optimizations for memory critical. Mobile RAM is already optimized for energy efficiency at the hardware level. It also provides power state switching interfaces to the operating system which enables the OS level energy optimizations. Many RAM optimizations have been explored for computer systems and in this paper we explore their applicability to smartphone hardware. In addition, we apply those optimizations to the newly emerging Phase Change Memory and study their energy efficiency and performance. Finally, we propose a hybrid approach to take the advantage of both Mobile RAM and Phase Change Memory. Results show that our hybrid mechanism can save more than 98% of memory energy as compared to the standard smartphone system with negligible impact on user experience.**

I. Introduction

As mobile phones become more versatile, they are being used not only to make phone calls and send messages but also to run a variety of applications. Subsequently, mobile computing is becoming an indispensible part of our personal and professional lives, as we are relying on smartphones to accomplish our basic computing and communication needs. We use them to access the Internet, watch videos, play games, read documents, and accomplish many other tasks on the go. The additional features and capabilities of mobile applications demand higher processing performance, which has a negative impact on a battery life. Since advances in battery technology continue to lag behind the demands placed for computing performance, power efficiency has become a critical consideration in the design of mobile systems.

While a smartphone contains many energy hungry components, such as CPU, display, and multiple radios, energy consumed by memory subsystem has been given limited consideration. Memory requirements in capacity and speed are increasing as smartphones tend to be used for multimedia applications which place higher demand on the main memory. For example, Windows 7 phone requires a minimum of 256MB main memory [1]. Larger memory will result in higher power consumption if its energy consumption is not properly addressed. Hardware manufacturers improve memory power efficiency by introducing multiple power states to balance the tradeoff between energy efficiency and performance. While power state transitions can be controlled by hardware, they are usually manipulated by the operating system, since the OS has a better knowledge with respect to what processes are currently running and their memory demands.

The task of designing efficient energy management techniques is challenging as the long power state transition latencies must be hidden from the application execution to preserve performance. Inefficient energy management may degrade user experience while interacting with the device, and potentially increase energy consumption since the tasks may take longer to execute, keeping other components on longer. Fortunately, full memory performance is usually not necessary to meet user's performance expectations, such as in the case of I/O bound applications. However, some CPU and memory bound applications may show significant degradation if memory is not in the highest performance state. Therefore, the key for memory energy management is to matching the memory performance/power state to the demand of the running application.

Many memory management mechanisms have been considered for servers, desktops, and portable computers [2], [3], [4], [5], [6], [7]. Different memory technologies [8], [9], [10] and hierarchies [11] have been investigated as well. The recent exploration of alternative memory technologies is driven by the development of phase change memory (PCM). PCM offers random access capability and performance comparable to DRAM, and non-volatility of flash memories. Subsequently, those characteristics allow PCM to be efficiently included in memory hierarchy or even replace DRAM to a certain extent. The tradeoffs between DRAM and PCM are being explored in conventional computing systems; however they have not been addressed in smartphones.

In this paper, we explore the efficiency of the existing energy management mechanisms on smartphones. We also propose a new memory organization for smartphones. We consider memory energy management based on applications, and include PCM in memory hierarchy to derive efficient

978-1-4577-1221-0/11/$26.00

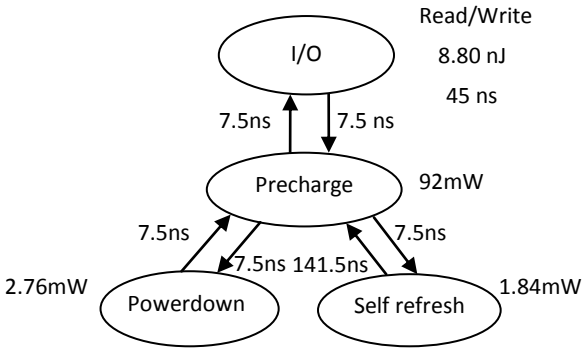

Figure 1. Power specifications for a 64MB 133MHz M-RAM device.

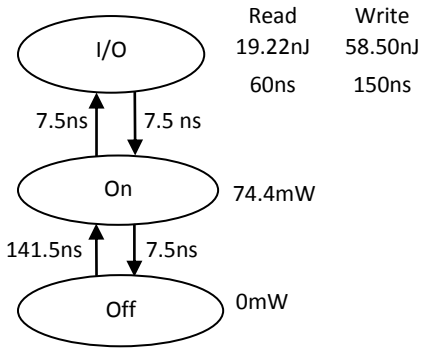

Figure 2. Power specifications for a 64MB 133MHz PCM device.

combination of policies and organizations. Subsequently, we make the following contribution in this paper: (1) We investigate the performance/energy profile for popular mobile applications; (2) We compare and evaluate performance and energy efficiency of several energy management mechanisms, that were designed for other systems, on a smartphone; (3) We propose a hybrid energy management mechanism which combines the traditional DRAM and emerging PCM.

II. Memory Technologies

The growing demand for mobile applications has enticed developers to consider smartphones as a platform of choice for application development. Currently there are over 100,000 applications developed for Android operating system, and Table I lists some of the most popular ones. Memory requirements range from 2.6MB for a simple Calculator to as much as 13.3MB for Google maps that needs to buffer map data in memory. According to [12], the memory subsystem in a smartphone can consume 34.5% of the overall phones energy when executing applications, resulting in a much higher energy consumption than the CPU which contributes 15.5% to the overall energy consumption. Therefore, it is critical to improve energy efficiency of main memory in smartphones.

Mobile RAM (M-RAM) is the most widely used memory technology in mobile devices. A recent contender for main memory technology is Phase Change Memory (PCM) that does not require state refreshes as M-RAM does, however offering lower performance than M-RAM.

A. Mobile RAM

The smallest power management unit in M-RAM is the rank and all devices in one rank are operating at the same power state [13]. A rank can operate in several different power states: (1) I/O: when memory is reading or writing data; (2) Precharge (PRE): where the next I/O can take place immediately without any delay; (3) Powerdown (PD): where several subcomponents of a rank are disabled to reduce power, such as I/O buffers, sense amplifier, row/column decoder, etc.; (4) Self Refresh (SR): where in addition to Powerdown state, the external clock and on-die termination are disabled to reduce power even further. Figure 1 shows the power states of a rank and their associated transition latencies as well as the energy consumption to perform one bus read or write.

According to Figure 1, a rank in the PD state only consumes 1/30 power of PRE state, while the SR state offers 33% additional power reduction to the PD state. The existence of low-power states provides the opportunity to save energy during idle periods. However, a rank in a low-power state (the PD or SR state) has to be transitioned to the PRE state before performing any I/O. These transitions incur high resynchronization latencies, especially between the SR state and the PRE state. Large number of resynchronizations may degrade overall system performance and even increase overall energy consumption. Therefore, the goal is to match memory power states to application's performance demand, such that the delays due to power state transitions are not exposed to the application execution.

B. Phase Change Memory

While an M-RAM cell uses capacitors to store data bits, a PCM cell uses phase change material to store data bits [8]. There are two stable structural states for the phase change material that can be changed with application of heat: amorphous and crystalline which differ greatly in resistivity. The difference of the two resistivities is used to store 0 and 1 states. When the temperature is above the crystallization point but below melting point for a period of time, the material enters the crystalline state which stands for logical 1. When the temperature rises above the melting point then drops quickly enough, the material enters the amorphous state which stands for logical 0. To read a cell, the bitline needs to be precharged to a certain voltage called the read voltage. If a cell is in the crystalline state, current will flow through the storage element and access transistor, causing the bitline to discharge. If the cell is in the amorphous state, the current cannot get through the storage element and the bitline will not be discharged.

Application name	Memory image size	Memory I/Os per second	Number of tasks	Long task percentage	Avg. long task length	Avg. short task length	Busy time percentage
Amazon mp3	6.05MB	105	5775	3%	131.02ms	1.74ms	5.4%
Calculator	2.59MB	140	5188	1%	286.27ms	3.76ms	9.2%
Facebook	6.24MB	281	10267	2%	200.07ms	2.20ms	13%
Online **Joke**	8.10MB	121	5208	1%	166.27ms	1.58ms	7%
Google **Maps**	13.27MB	704	4096	2%	225.43ms	3.23ms	15%
MSN messenger	9.06MB	206	3727	2%	222.04ms	3.47ms	9%
Music player	4.53MB	89	3969	1%	200.10ms	3.70ms	4%
Newspaper	10.31MB	636	4706	6%	504.49ms	3.08ms	29%
Stock quote	3.59MB	291	3983	3%	356.82ms	3.14ms	16%
Toodo	5.15MB	207	3882	3%	239.54ms	3.54ms	11%
Twidroid	11.08MB	112	4117	2%	166.93ms	1.87ms	5%
Youtube	5.79MB	181	3968	2%	205.91ms	1.85ms	9%

Table I
APPLICATION TRACE STATISTICS.

Figure 2 shows the power states for PCM and their associated transition latencies [9], as well as the energy consumption to perform one bus read and write. PCM can operate in three states: (1) I/O: when memory is reading or writing data; (2) ON, where the memory device is ready to accept I/O instructions in the next cycle; (3) OFF, where all peripheral subcomponents are turned off. PCM consumes significantly less idle power than M-RAM, especially in the low-power state where the power consumption is reduced to 0. However, PCM consumes more energy to perform I/O operations, particularly write operations, since the cell state has to be changed. Therefore, we should leverage the tradeoffs between performance and energy efficiency to apply PCM technology in mobile devices.

III. CHARACTERIZING MOBILE SOFTWARE

Table I lists 12 most popular Android applications selected from the Android market along with their trace statistics. We used a T-Mobile G1 smartphone to collect the application traces. Each trace consists of task intervals with the task execution length and the number of memory I/Os. We used timestamp counters to measure the task CPU cycles and utilized the performance counters to count the memory I/Os. A task is defined as the CPU processing and memory activity between two consecutive schedulings of the idle process. Furthermore, a task completing within 50ms is referred to as a short task, since its execution is not noticeable to user, while a task running longer than 50ms is referred to as a long task, which execution time is noticeable to the user.

The bold part of an application name in Table I will be used to refer to the particular application throughout the paper. Memory image size represents the largest memory footprint of an application during normal execution. Larger image size usually implies more data-centric applications. Number of memory I/Os per second is obtained by dividing the number of total memory I/O operations by the total execution time. Higher memory access intensity imply more memory I/O operations and thus more I/O energy. Number of tasks shows the total number of tasks during the entire execution period. The long task percentage shows the portion of tasks longer than 50ms. Average long/short task length is calculated as the total long/short task time divided by the number of long/short tasks. Busy time percentage is the ratio of total task time over the total execution time.

The selected applications are:

- *Amazon mp3* is a music application that allows to preview, buy a single song, or a full album from Amazon.com. Songs are displayed together with their name and album. The previews of songs are downloaded to memory before being played for the user.
- *Caculator* is a calculator application capable of doing scientific calculation.
- *Facebook* allows user to access their Facebook account and the perform many activities such as reading news from friends, previewing pictures, inserting comments, viewing profiles of other users, etc.
- *Online joke* allows users to read, publish, and rate jokes online. Most network activity is for transferring small size text and therefore incurs short delays.
- *Google maps* allows users to view maps, street views, satellite images, and perform route planning. All map information is stored online and only the information near the current view point will be cached locally.
- *MSN messenger* provides online chat. CPU workload and memory accesses are limited since the applications execution consists mainly of short text transfers over the network.
- *Music player* plays audio stored locally on the phone and only short fragment of the current song is buffered in memory.
- *Newspaper* is online news reader that provides latest news from several news providers. The content includes pictures, texts, and hyperlinks.
- *Stock quote* provides real time stock quotes, latest news,

CPU	Qualcomm MSM7201A, 528MHz
OS	Android 1.6 (Donut)
ROM	256MB
RAM	192MB (3 ranks), 133MHz

Table II
SYSTEM CONFIGURATION OF T-MOBILE G1.

and historical price for the New York Stock Exchange.

- *Toodo* is a planner providing reminders and note taking features. Memory and CPU activity is present when create new schedule or edit existing ones. When scheduled event arrives, memory and CPU activity is limited to loading and playing a notification sound.
- *Twidroid* allows direct access to users twitter accounts. Users can read messages, follow messages as well as search for content.
- *Youtube* allows video searching and playing from Youtube on the phone.

Smartphone applications are highly interactive, therefore strictly CPU and memory bound applications are far less common. Compared to the CPU speed, human interactions are extreme slow, such that a mobile system is idle and waiting for user input for the majority of time. Prior study has shown that human perception threshold is between 50ms and 100ms [14] and any event shorter than the perception threshold appears instantaneous to the user. Completing task execution earlier than the perception threshold is meaningless since the user will not notice this amount of time and cannot initiate new tasks any sooner. This observation is the key to enabling energy optimizations without impacting observed application performance.

According to our measurement, the majority of tasks are very short as more than 90% of all tasks complete within 10ms. Moreover, 95% of all tasks are shorter than 50ms, indicating that these tasks can be extended to the 50ms perception threshold deadline without any performance penalty. Similarly, for the remaining 5% of long tasks, any additional extension less than 50ms will not be noticed by the user, avoiding performance degradation. These observations provide great opportunity and flexibility in memory energy management, since memory can be potentially put into a low-power state during idle times, as long as the task extension does not introduce any user noticeable delays.

IV. MECHANISM COMPARISON

M-RAM needs to refresh the storage cells regularly for data retention, therefore consuming non-negligible power even in the low-power state. PCM is able to completely eliminate idle power due to non-volatile nature. We will evaluate the efficacy of various energy management mechanisms on M-RAM and PCM under the same execution environment. We built a trace-driven simulator to model a T-Mobile G1 smartphone as shown in Table II. The memory subsystem consists of a memory controller and three 64MB ranks(192MB totally), for either M-RAM or PCM, according to the specifications of the Infinion HYB25L512160AC7.5 M-RAM [13] and the PCM memory [9]. The simulator feeds with the traces, determines the memory power state, and conducts task execution under the current CPU and memory state. The memory controller conducts memory I/O operations, and executes power state transitions for each rank based on the energy management mechanism.

A. Power-Aware Virtual Memory

Idle periods are common in mobile applications when the smartphone is waiting for user input, therefore energy consumption can be reduced by powering down the memory devices during system idle time. A simple and effective way to provide energy management is the Power-Aware Virtual Memory (PAVM) [2], which keeps the memory devices occupied by the currently running process in the active state while keeping all other memory devices in a low-power state to save energy. Memory devices used by the newly scheduled process are powered up during the context switch time to minimize the delays exposed to the user due to power state transitions.

Figure 3 shows the energy consumption of M-RAM and PCM for the PAVM mechanism and the standard system with no energy management (ON). PCM consumes about 2 times more I/O energy than M-RAM for each application which is consistent across mechanisms. Nevertheless, idle energy consumption accounts for the majority of energy consumption since idle time dominates each application execution. In the standard system, PCM consumes 20% less energy than M-RAM, since PCM does not consume energy due to frequent rank refreshes and bitline precharging. Evaluated applications have smaller footprint than one rank and can easily fit in the rank occupied by the operating system structures. Subsequently, the PAVM mechanism keeps the only one used rank in active idle state and two unused ranks in the low-power state, reducing memory energy consumption by 94% and 95% respectively for M-RAM and PCM. Compared to M-RAM with the PAVM mechanism, PCM with the PAVM mechanism offers the extra energy savings because PCM does not consume any energy for maintaining memory state in the ranks powered down.

PCM has higher I/O latencies that can potentially degrade performance. Through detailed study, we found negligible performance impact (less than 1%) on the application execution. Therefore, PCM turns out to be a viable alternative for M-RAM as main memory in mobile devices. Furthermore, utilization of the PAVM mechanism does not introduce any additional delays into application execution. This is because the initial transition overhead is hidden by the context switch when the process is scheduled in, and thus there are no additional power state transitions during the execution. Subsequently, the PAVM mechanism provides energy savings without any performance penalty and will

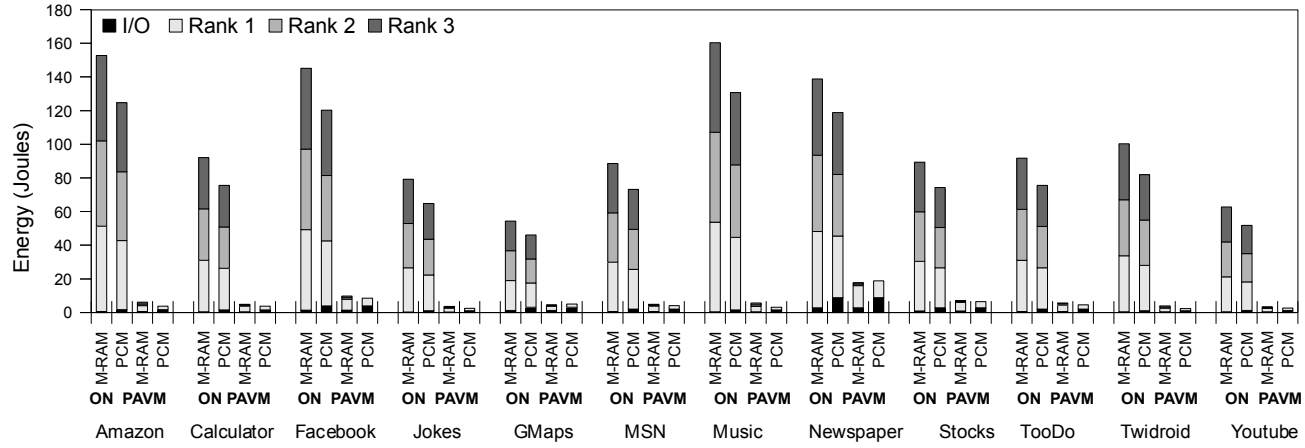

Figure 3. Memory energy consumption with a standard system (ON) and the PAVM mechanism. The left two bars for each application show the energy of M-RAM and PCM in standard system, while the right two bars show the energy for the PAVM mechanism.

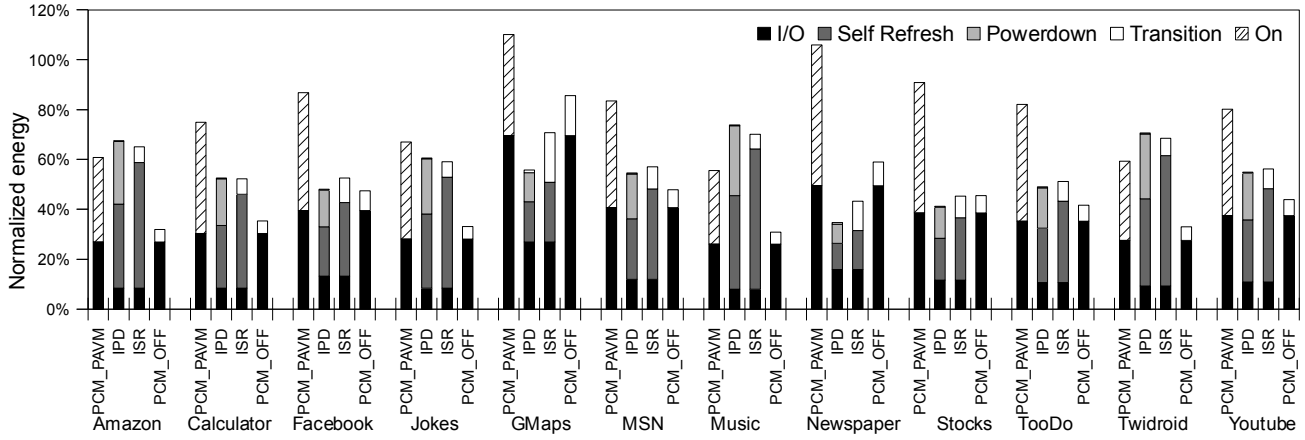

Figure 4. Memory energy consumption for on-demand mechanisms normalized to the PAVM mechanism on M-RAM.

be used as a base case for comparison with more aggressive energy management mechanisms in the following section.

B. On-demand Mechanisms

Despite of PAVM's benefits to the standard system, it fails to address the energy efficiency of the active rank accessed during the process execution. Therefore, we consider more aggressive optimizations that offer energy savings even for the currently active rank. Immediate Power Down (IPD) mechanism and Immediate Self Refresh (ISR) [15] mechanism have been proposed for RAM to provide on-demand power state transitions and improve energy efficiency of active ranks.

The IPD and ISR mechanisms rely on the fact that the user may not be able to perceive delays of certain length introduced by power state transitions. Therefore, energy consumption may be further reduced by powering down the active ranks during idle periods. Initially, all ranks are kept in the SR states, with those mechanisms. As soon as an I/O request arrives at the memory controller, the rank to be accessed is transitioned to the PRE state, and transitioned back to a low-power state (the PD state for IPD, or the SR state for ISR) immediately after the I/O completes.

This on-demand approach can be applied to different memory technologies, yielding different energy management mechanisms. M-RAM has the PD and SR states and the mechanisms are applied as in the original design. PCM has only one OFF state where all of the supporting devices are powered off, and it is comparable to the SR state in M-RAM. Subsequently, the PCM_OFF mechanism for PCM is a counterpart of the ISR mechanism for M-RAM.

Figure 4 shows the energy consumption of M-RAM and PCM for each application and each mechanism. Each energy bar is normalized to M-RAM with the PAVM mechanism. We show in the first bar the energy consumption of PCM with the PAVM mechanism and in other bars the energy of on-demand mechanisms. We first examine the energy efficiency on M-RAM. By keeping the active rank in the low-power state, the IPD and ISR mechanisms reduce the majority of idle energy consumed in the PRE state. On

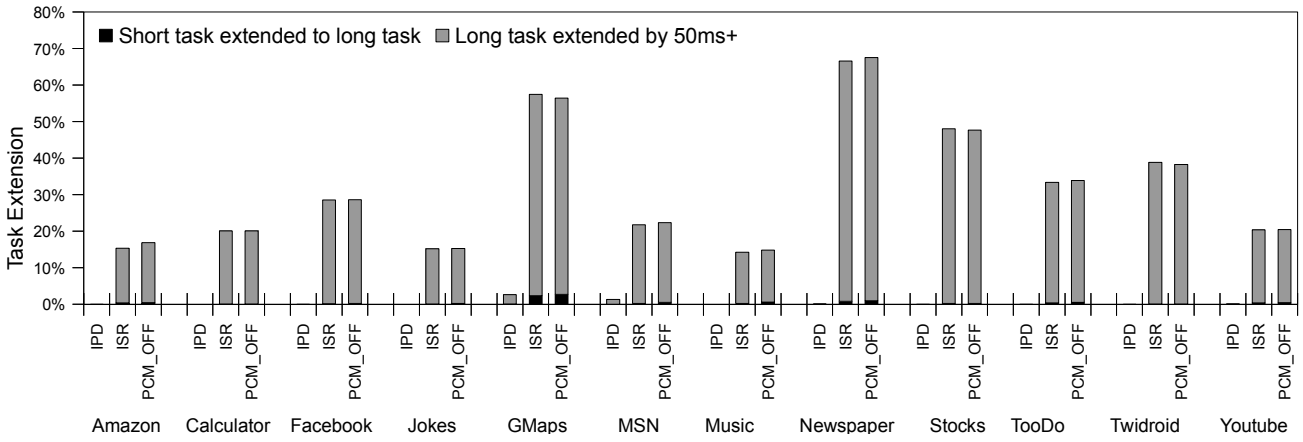

Figure 5. The distribution of extended tasks that expose delays for on-demand mechanisms.

average, IPD and ISR reduce 45% and 42% of the energy consumed by M-RAM with the PAVM mechanism. The I/O energy remains the same due to the constant amount of memory I/Os.

We also notice that the two on-demand mechanisms even outperform the PAVM mechanism on PCM. Due to the reduction of idle energy, I/O energy becomes a significant portion of overall energy consumption. As a result, PCM's inferior I/O efficiency can't offset its energy savings from idle periods, except for lightly loaded applications Amazon, Music and Twidroid. On average, the IPD and ISR mechanisms save 31% and 28% of energy consumed by PCM with the PAVM mechanism.

The IPD and ISR mechanisms are comparable in energy consumption, although the SR state consumes 33% less power than the PD state. This is because the 18 times longer transition time from the SR state incurs more transition overheads, which offsets the idle energy savings. In Table I, Facebook, GMaps, Newspaper and Stock are the top applications in memory access intensity, due to the involved heavy image processing. Therefore, it is not surprising to see that the IPD mechanism performs apparently better than the ISR mechanism for these four applications, resulting in 17% energy savings on average.

The PCM_OFF mechanism completely eliminates the active idle energy, resulting in 44% energy reduction over the PAVM mechanism on PCM. Compared to the IPD and ISR mechanisms on M-RAM, the PCM_OFF mechanism offers 18% and 22% energy savings respectively. However, for some applications, PCM_OFF is outperformed by the other two mechanisms. The reason is twofold. First, PCM consumes more I/O energy than M-RAM with the IPD and ISR mechanisms. Secondly, the PCM_OFF mechanism incurs comparable transition overheads to the ISR mechanism, much higher than the IPD mechanism. Therefore, in memory intensive applications, the PCM_OFF mechanism may consume more overall energy than the other mechanisms, since the idle energy savings cannot pay off its overheads. In worst cases of GMaps and Newspapers, PCM_OFF incurs 61% and 32% more energy consumption than the IPD and ISR mechanisms respectively.

The delays introduced by energy management mechanisms may be exposed to the user and adversely affect user experience. The resulting performance degradation may even erode the energy savings since the whole system stays on longer. As discussed before, we consider system performance is preserved when a short task is extended within 50ms, or a long task is extended by less than 50ms. Otherwise, user noticeable delays are exposed and the performance is degraded.

Figure 5 shows the ratio of extended tasks that expose delays to all tasks for each application and each mechanism. We can observe that the IPD mechanism achieves the best performance with negligible delays exposed. The ISR and PCM_OFF mechanisms, on the other hand, incur more evident degradation due to the 141.5ns long transition latency. For light-loaded applications such as Amazon, Calculator, Jokes and Music, the ISR and PCM_OFF mechanisms perform relatively well with only 14% to 20% long tasks extended. However, their performance drops significantly for heavy-loaded applications, especially for GMaps and Newspaper, where 55% and 66% of the tasks are extended with delays exposed to the user.

Through detailed evaluation, we can see that when memory energy is the only concern, the novel PCM technology with on-demand mechanism surpasses the traditional M-RAM. However, taking into account the performance as well, M-RAM still has the chance to beat PCM, for example when the IPD mechanism is applied on M-RAM. We therefore need an approach to balance energy and performance more efficiently than any standalone memory technology.

C. Hybrid Memory Architecture

From the above analysis, we can see that PCM is superior to M-RAM for its lower idle power consumption, while M-

RAM excels PCM for faster I/O speed and lower I/O energy. As suggested in [11], a hybrid memory consisting of M-RAM and PCM can improve both the energy efficiency and performance. To take advantage of both memory technologies, we therefore propose a hybrid memory architecture which combines M-RAM and PCM. Subsequently, a hybrid mechanism for this hybrid memory is proposed accordingly: The frequently invoked applications are moved into M-RAM to increase I/O performance, while less-frequently-executed applications are moved to PCM to reduce energy consumption in idle state.

To identify the most frequently invoked application, we modify the Android system with a LRU (least-recently-used) list to keep track of the applications invoked. When an application is invoked and its image does not reside in M-RAM, it is loaded into M-RAM either from secondary storage or PCM, and the corresponding process identifier is put at the head of the LRU list. If there is no sufficient space in M-RAM to load the newly invoked application, the OS will keep swapping out applications from the tail of the LRU list to PCM until the required free space is reached. When an application is closed, its memory image will stay in M-RAM until it is swapped out.

We evaluated this hybrid approach on a memory subsystem consisting of 64MB M-RAM (1 rank) and 128MB PCM (2 ranks). From Figure 4 and Figure 5, the IPD mechanism achieves the best energy efficiency and performance on M-RAM, while the PCM_OFF mechanism saves most energy on PCM. Subsequently in the hybrid memory, we apply the IPD mechanism on the single M-RAM rank and the PCM_OFF mechanism on the two PCM ranks.

We compare the proposed hybrid approach against the IPD and ISR mechanisms on 192MB standalone M-RAM, as well as the PCM_OFF mechanism on 192MB standalone PCM. Figure 6 shows the energy consumption for each mechanism when randomly invoking the selected 12 applications for 10, 100 and 1000 times. The results are normalized to 192MB standalone M-RAM with the PAVM mechanism. As we can see, the hybrid approach exhibits the best energy efficiency and stays consistent over the progress of application execution. It saves 68% of I/O energy and incurs negligible transition energy as compared to the PCM_OFF mechanism, since memory I/O requests are actually served at the M-RAM rank. As compared to the IPD and ISR mechanisms, it also reduces 57% and 59% of idle and transition energy, because the unused PCM ranks are completely turned off. In addition, it incurs negligible swapping overheads due to rare occurrence and fast internal transfer between M-RAM and PCM. As a result, the hybrid approach saves 45%, 47% and 31% of the overall energy from the IPD, ISR and PCM_OFF mechanisms respectively.

From our study, the hybrid approach preserves more than 99% of IPD's performance since memory I/O is performed on M-RAM. The additional delays only come from the swapping operations between PCM and M-RAM. In conclusion, the hybrid approach achieves the best energy efficiency among all mechanisms while maintaining almost full memory performance. While either standalone memory technology has its own shortcoming, the hybrid approach provides an practical solution to memory energy management for future mobile devices.

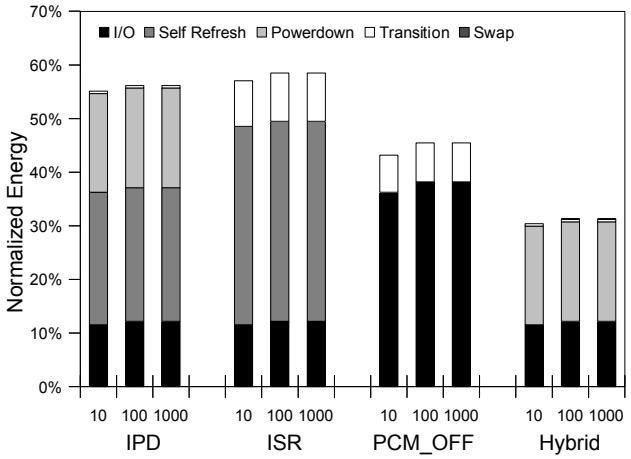

Figure 6. Normalized energy consumption when randomly executing the selected applications for 10, 100 and 1000 times respectively.

V. Related Work

As the upsurge in upgrading mobile hardware continues, energy management in mobile devices has become a hot area. For mobile devices that are used mostly for multimedia applications, [16] proposed a mechanism which combines a special real-time CPU scheduler with dynamic voltage scheduling to meet the specified QoS of each individual application. To reduce screen power consumption, Iyer et al. proposed in [17] an energy-adaptive display subsystem which tries to reduce the energy consumed by the unfocused part of the screen. Due to the mobile nature of handhold devices, accessing and caching remote data is of great importance. [18] proposed an energy-efficient data caching and prefetching mechanism by taking into account the characteristic of mobile system such as connection-disconnection, mobility handoff, data update and user request patterns.

As application become more data centric, memory requirements for both speed and capacity has been increasing continuously. Subsequently, lots of research work has been done about memory energy management. Li et al. proposed in [7] a mechanism which guarantee the system performance by stopping power management strategy temporarily whenever the whole system performance is below the specified level. By using page information possessed by the operating system, [2] proposed Power-Aware Virtual Memory which manages power states of main memory on per-process basis. Bi et al. [19] targeted the buffer cache and utilized the OS I/O handling routines to hide power state transition

delays to minimize the impact of aggressive memory energy management. Trying to improve memory energy efficiency without losing performance, Liu et al. proposed in [20] a memory management mechanism through sacrificing the correctness of non-critical data. On the architecture level, Deng et al. proposed in [21] a mechanism to increase memory energy efficiency by applying DVFS to the memory controller and DFS to the memory channels.

PCM, as a newly developed RAM technique, has distinct advantages over traditional DRAM such as non-volatile nature and lower idle power consumption. To facilitate the programming on persistent memory, Volos et al. proposed in [22] a programming interface for creating and managing persistent memory as well as ensuring consistency. To further explore the usage of PCM, [8] and [10] proposed several data caching and write-back mechanisms to avoid PCM's shortcomings such as long latency, high-cost write and finite endurance. In [9], Zhang et al. analyzed the performance results through substituting PCM for DRAM in a 3D-integation technique to form an energy efficient and thermal friendly 3D stack architecture.

VI. Conclusion

Due to the increasing process power of smartphones, they are becoming an inevitable part of our everyday life. As mobile applications become more data-centric, current smartphones are equipped with larger and faster memory subsystem. In this paper, we evaluated several energy management mechanisms for main memory that were originally proposed for desktops and servers, on a mobile system. The evaluation included both the traditional M-RAM and the novel PCM. Results show the state-of-the-art PAVM mechanism exhibits excellent efficiency, remaining the same performance while saving more than 90% energy as compared to the standard system with no energy management. On-demand mechanisms offer around 40% additional energy savings over the PAVM mechanism, for both M-RAM and PCM, but at the cost of performance degradation to various extents. To further improve energy efficiency, we then proposed a hybrid approach with mixed memory technologies and mechanisms. We have shown that this proposed approach outperforms all other existing mechanisms, with 98% energy savings and negligible performance overheads as compared to the standard system. The analysis suggests that utilizing hybrid memory organizations may be a better choice in future smartphone design than using standalone memory technology only.

VII. Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 0844569.

References

[1] "Windows phone 7," http://en.wikipedia.org/wiki/Windows_Phone_7.

[2] H. Huang, P. Pillai, and K. G. Shin, "Design and implementation of power-aware virtual memory," in *ATC*, 2003.

[3] M. Lee, E. Seo, J. Lee, and J.-s. Kim, "Pabc: Power-aware buffer cache management for low power consumption," *IEEE Transactions on Computers*, 2007.

[4] V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini, "Dma-aware memory energy management." in *HPCA*, 2006.

[5] H. Huang, K. G. Shin, C. Lefurgy, and T. Keller, "Improving energy efficiency by making dram less randomly accessed," in *ISLPED*, 2005.

[6] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers," *Computer*, 2003.

[7] X. Li, Z. Li, Y. Zhou, and S. Adve, "Performance directed energy management for main memory and disks," *Transactions on Storage*, 2005.

[8] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ISCA*, 2009.

[9] W. Zhang and T. Li, "Exploring phase change memory and 3d die-stacking for power/thermal friendly, fast and durable memory architectures," in *PACT*, 2009.

[10] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, 2009.

[11] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *ISCA*, 2009.

[12] A. Rice and S. Hay, "Decomposing power measurements for mobile devices," in *PerCom*, 2010.

[13] Infineon, "Hyb25l512160ac-7.5 512mbit mobile-ram," 2004, www.infineon.com.

[14] B. Schneiderman, *Designing the user interface: strategies for effective human-computer interaction*, 1998.

[15] H. Huang, K. G. Shin, C. Lefurgy, K. Rajamani, T. W. Keller, E. V. Hensbergen, and F. L. R. III, "Software-hardware cooperative power management for main memory." in *PACS*, 2004.

[16] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," in *SOSP*, 2003.

[17] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan, "Energy-adaptive display system designs for future mobile environments," in *MobiSys*, 2003.

[18] H. Shen, M. Kumar, S. K. Das, and Z. Wang, "Energy-efficient data caching and prefetching for mobile devices based on utility," *Mob. Netw. Appl.*, 2005.

[19] M. Bi, R. Duan, and C. Gniady, "Delay-hiding energy management mechanisms for dram," in *HPCA*, 2010.

[20] L. Song, P. Karthik, M. Thomas, and G. Benjamin, "Flikker: Saving dram refresh-power through critical data partitioning," in *ASPLOS*, 2011.

[21] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: active low-power modes for main memory," in *ASPLOS*, 2011.

[22] H. Volos, A. J. Tack, and M. M. Swift, "Mnemosyne: lightweight persistent memory," in *ASPLOS*, 2011.