# Inexpensive HMC+DRAM Hybrid Main Memory Architecture with LRU-based Data Distribution Management

Ying Zhou[1], Lin Zhang[1,*], Shuang Niu[1] and Shulin Zhao[2]
[1]Information Engineering, Shanghai Maritime University, Pudong District, Shanghai, China
[2]College of Computer Science, Shandong University, Jinan, Shandong, China
[*]Corresponding author

*Abstract*—**The Hybrid Memory Cube (HMC) is a 3-D-stacked DRAM architecture whose I/O interface achieves up to 320 GB/s of external bandwidth. Therefore, the HMC is a promising alternative to DDRx memory due to its potential to achieve substantially improved memory bandwidth. However, the high price of a HMC device compromises cost efficiency when the device is lightly utilized. The cost of a HMC device with 2GB capacity is about 5 times of a DDRx memory with the same capacity. In this paper, we propose a new inexpensive HMC+DRAM hybrid main memory architecture to reduce the cost consumption. In order to manage such hybrid memories, we develop a LRU-based data distribution mechanism to determine the destination of particular data flow. Evaluations show that our scheme reduces the cost consumption of Main Memory by 48% on average with a negligible performance degradation compared to a current HMC-based system. Also, our architecture outperforms a current DRAM-based system by 1.5 times, and reduces the response delay by 1.2 times.**

***Keywords-component; formatting; style; styling; insert (key words)***

## I. INTRODUCTION

The Hybrid Memory Cube (HMC) is a recent main memory technology that aims to overcome the memory wall problem [1] due to the limited bandwidth and long access latency in current memory systems. As shown in FIGURE I, a HMC device contains several DRAM dies stacked on top of a logic layer, where several DRAM memory controllers communicate with the host processor using an abstracted interface [2]. Through-Silicon Vias (TSVs) vertically connect memory controller to several memory banks in the DRAM stack, thereby providing about 1Tb/s of internal bandwidth and 320GB/s of external bandwidth [3]. Because the TSVs build a denser interconnect with shorter path lengths than traditional DDRx memories, the throughput between the controller and memory banks is higher.

On the other hand, the HMC can be denser, faster, and more power-efficient than DDRx, but it has problems with price cost. Such high bandwidth comes at a serious money consumption: The cost of a HMC device with 2GB capacity is about 20 times of a DDRx memory with the same capacity [4]. Although it is evident that the high bandwidth of HMCs could be a breakpoint to overcome the memory wall problem, such expensive price may not be worthwhile especially when the full bandwidth is underutilized.

In this paper, we propose an inexpensive HMC+DRAM hybrid main memory architecture with LRU-based data distribution management mechanism to extremely reduce the price cost with limited performance lost. We add a data-distribution module into the Linux Kernel to decide the destination of data block. And to minimize the extra delay incurred by this module, our scheme keeps track of the performance sensitivity to the distribution decision and determines the current destination based on the feedback and the basic LRU mechanism. In this work, we focus on read/write latencies, delay and bandwidth. Our goal is to present an auto-adapted inexpensive hybrid memory system and the key idea is to use the LRU-based data-distribution mechanism to choose HMC or DDRx to store data blocks.

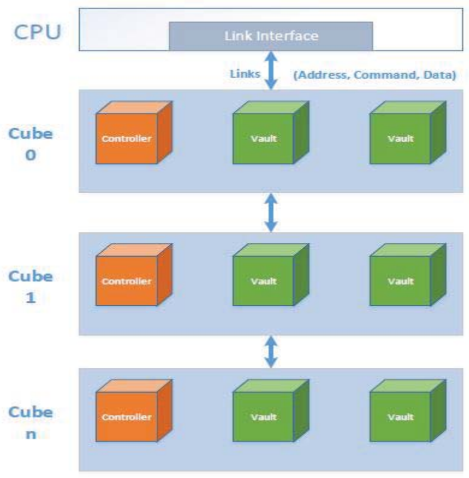


FIGURE I. THE ARCHITECTURE OF AN HMC MEMORY SYSTEM

The rest of this paper is organized as follows. Section II briefly introduces the HMCs and motivates the need for price-cost management of the current memory systems. Section III describes our LRU-based data-distribution module and HMC+DRAM hybrid main memory architecture in detail. Section IV evaluates the effectiveness of our approach. Finally, the conclusion is drawn in Section V.

## II. Related Work

### A. Hybrid Memory Cube

Because TSV is an emerging technology, there has been a recent surge of academic and industry research about HMC’s impact on performance from the architecture perspective [5]. Mushfique Junayed Khurshid et al. [6] proposed a method to reduce the maximum temperature and variation by using data compression. Using this method, data compression on the processor side can cause reduced temperature as well as increased performance in the Hybrid Memory Cube. Yinhe Han et al. [7] proposed a data-aware refresh control scheme, Trial and Error (Trial-n-Error), which leverages the data-pattern dependence characteristics of the cells’ retention time to reduce refresh operations. It makes use of the reliability and communication resources inside the HMC logic base, and strikes a balance between hardware reliability and refresh overheads, which is especially suitable to be applied to HMC that suffers from the negative effects caused by frequent refreshes. Shibo Wang et al. [8] introduced a new technique that alleviated the long wake-up penalty of an HMC by employing erasure codes. Gwangsun Kim et al. [9] proposed a memory-centric network in which all processor channels are connected to HMCs and not to any other processors as all communication between processors goes through intermediate HMCs. Seth H Pugsley et al. [10] presented a high-level description of the Near Data Computing (NDC) hardware and accompanying software architecture, which presented the programmer with a MapReduce style programming model. Junwhan Ahn et al. [3] proposed an adaptive mechanism to partially disable off-chip links of HMCs to reduce the energy consumption of the off-chip links. Yarui Peng et al. [11] presented a cross-domain CAD/architectural platform that addresses DC power noise issues in 3D DRAM targeting stacked DDR3, Wide I/O, and hybrid memory cube technologies.

### B. Hybrid Memory Architecture

Hybrid memory architecture has been studied in the area of storage systems recently. Kai Chen et al. [12] proposed a hybrid global memory architecture by using both DRAM and PCM memory technologies. Such an architecture has the performance and dynamic power benefits from DRAM as well as the leakage power benefits from PCM. Luis Angel Bathen et al. [13] presented HaVOC: a run-time memory manager that virtualizes the hybrid on-chip memory space and supports efficient sharing of distributed ScratchPad Memoies (SPMs) and Non-Volatile Memories (NVMs). They introduced the notion of data volatility analysis and proposed a dynamic filter-based memory allocation scheme to efficiently manage the hybrid on-chip memory space. Meikang Qiu et al. [14] proposed a novel hybrid on-chip scratchpads memory (SPM) that consists of a static random access memory (RAM), a magnetic RAM (MRAM), and a zero-capacitor RAM for CMP systems by fully taking advantages of the benefits of each type of memory. To reduce memory access latency, energy consumption, and the number of write operations to MRAM, they also proposed a novel multidimensional dynamic programming data allocation (MDPDA) algorithm to strategically allocate data blocks to each memory. Soyoon Lee et al. [15] presented a new memory management technique for hybrid phase change memory (PCM) and DRAM memory architecture that efficiently hides the slow write performance of PCM. They presented a new page replacement algorithm called CLOCK-DWF that estimates future write references accurately, allowing frequent memory writes to be absorbed by DRAM. Xiangyong Ouyang et al. [16] proposed an SSD-Assisted Hybrid Memory that expands RAM with SSD to make available a large amount of memory. Hybrid Memory works as an object cache and it manages resource allocation at object granularity, which is more efficient than allocation at page granularity.

### C. Motivation

HMC technology has already been studied on their impact on performance and energy efficiency. However, the present researches merely focused on the potential application of HMCs and failed to notice the price-cost of the whole memory system. Also, to the best of our knowledge, the hybrid memory architecture technology has not been considered in building a storage system based on HMC. Therefore, we focus on the trade-off of the whole system, and propose a new inexpensive HMC+DRAM hybrid main memory architecture to reduce the cost consumption using a LRU-based data distribution module in Operating System Kernel level.

## III. System Design and Implementation

The present Inexpensive HMC+DRAM Hybrid Main Memory

Architecture has the performance and latency benefits from HMCs as well as the price-cost benefits from DRAM. The benefits of our design reside on four aspects:

- In order to reduce the latency and accelerate the I/O speed, the frequently used data which labeled hot data will be loaded to HMC.
- A LRU-based data-distribution module is added to Linux Kernel to label a particular data flow as hot or cold, then determine the destine of this data flow.
- An improved LRU algorithm is developed to reduce the complexity from O(n) to O(1), where four pages’ activeness level, inactive-inreferenced, inactive-referenced, active-inreferenced and active-referenced are introduced to decide where to distribute this page.
- Spin-lock is used in this LRU-based data-distribution module to avoid deadlock, which is indispensable to maintain the file system consistency.

### A. Hybrid Memory Storage Architecture

In the present HMC+DRAM Hybrid Main Memory Architecture, the memory system is composed of three parts, Linux Kernel Data Distribution Module, traditional DRAM and HMC memory device. FIGURE II shows the hardware architecture of our system. When CPU sends several I/O requests to the Memory Bus, the Linux Kernel Data Distribution Module receives these requests and has to choose those that have the best ”return on investment” for migration to HMC. While the others have to be sent to DRAM. Both DRAM and HMC have the access to the shared Disk, so the spin-locks have to be used to keep consistency. The key design of our system relies on two aspects:

• How to distribute file system data between the two types of devices: DRAM and HMC.

• How to organize the data before writing to the Disk Layer to ensure no potential deadlock would occur.

These two points will be elaborated in the following sections.

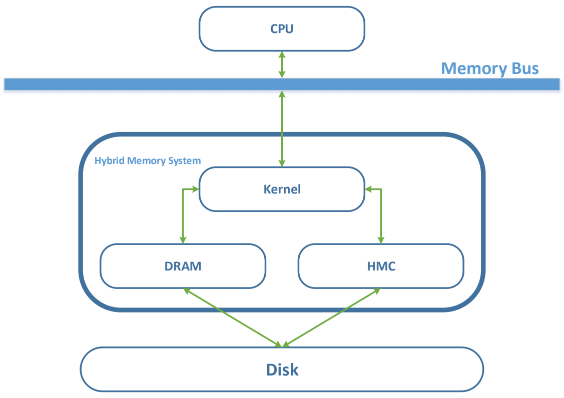


FIGURE II. HYBRID MEMORY STORAGE ARCHITECTUER

## B. *Deciding Which Pages to Put in HMC*

Within the whole set of candidate pages, the OS must choose those that have the relatively possibility to jump between different layers for migration to HMC. In this way, the hot data which are migrated frequently will be dealt with in HMC, the whole performance of this system will benefit from the HMC's extremely high speed. Four metrics are introduced to distribute the data flow [17]:

• Page types: OS tends to associate pages with type information. Stack pages, for example, are good candidates for migration to HMC because of the frequent changes; non-file pages shared between two processors are good candidate. While code pages are bad candidate because they are unlikely to be moved out of the memory layer.

• File reference modes: 80% of temporary files will be deleted within four seconds [18], so pages from files marked temporary should be migrated to HMC.

• Application-supplied page attributes: Memory-hungry applications with large workload, such as databases, are good candidate for migration to HMC.

• Page history: The OS could track the history for each page, and migrate pages with high recurrence rate to HMC.

## C. *Improved-LRU-based Data Distribution Module*

The Data-Distribution Module selects hot pages for HMC and cold pages for DRAM, using an improved LRU algorithm whose time complexity is $O(1)$. In this improved LRU algorithm, the number of lists to record different-activeness-level pages increases to four, indicating four labels: inactive-inreferenced, inactive-referenced, active-inreferenced and active-referenced. Each level corresponds to a list. The Data-Distribution Module levels up or down one page's activeness according to the four metrics mentioned above. And HMC simply select pages from the head of the active-referenced list, while DRAM from the tail of the inactive-inreferenced list. These operations with $O(1)$ time complexity will substantially reduce the latency.

## D. *Spin-lock: Deadlock Avoiding*

File system consistency is always an indispensable issue in file system design. As described above, both HMC and DRAM can access the shared Disk block. Therefore, there is a possibility of deadlock when HMC and DRAM are both trying to write a same Disk block. In order to avoid deadlock, we lock each page when loaded to memory level and unlock it after the I/O request has already submitted to the Disk layer. The procedure of the data-distribution strategy is described in Algorithm 1.

**Algorithm 1** Improved-LRU-based Data Distribution Algorithm

```
Input: $P$: Whole set of pages from processors.
Output: Distribute every pages to HMC or DRAM.
 1: for each $page$ in $P$ do
 2:     $page \Rightarrow corresponding\_LRU\_list$
 3: end for
 4: $fork()$ 2 threads to concurrently handle HMC and DRAM
 5: for all $active\_referenced\_list.getHead(hotPage)$ do
 6:     $spin\_lock(hotPage)$
 7:     Load $hotPage$ to HMC
 8:     $wait\_complete(hotPage)$
 9:     $unlock(hotPage)$
10:     $adjust\_all\_lists$
11: end for
12: for all $inactive\_inreferenced\_list.getTail(coldPage)$ do
13:     $spin\_lock(coldPage)$
14:     Load $coldPage$ to DRAM
15:     $wait\_complete(coldPage)$
16:     $unlock(hotPage)$
17:     $adjust\_all\_lists$
18: end for
```

## IV. EVALUATION

## A. *Methodology*

We evaluate our design using a low-level simulator HMC-SIM, whose goal is to provide architectural simulation frameworks the ability to begin migrating current banked DRAM memory models to stacked HMC-based designs without a reduction in simulation fidelity or functionality [19]. TABLE I shows the details of the simulation configuration. We choose two comparison points that represent an upper and lower bound to show the validity of this system. The lower bound is a 4-GB quad-channel DDR4 memory system and the upper bound is a 4-GB HMC system, while our design is a hybrid 1-GB HMC + 3-GB DDR4 memory system. So the cost consumption of our memory system is only about 48% of the 4-GB HMC system [20]. In this section, we present the performance of our file system in four aspects: bank conflict, read and write request, crossbar latency and crossbar request stall.

TABLE I. SIMULATION CONFIGURATION

| | |
|---|---|
| **CPU** | **8 out-of-order x86 cores** |
| **L1 Cache** | 128 K L1-I / 128 K L1-D |
| **L2 Cache** | 2 MB shared L2 |
| **Operation System** | Ubuntu Linux 12.04 |
| **Banks** | 8 |
| **Vaults** | 16 |
| **Queue Depth** | 64 |
| **Crossbar Depth** | 128 |
| **Number of Requests** | 10240 |
| **Benchmarks** | Hmc physrand |
| | Decode physrand |
| | Gups |
| | stream |

## B. Bank Conflict

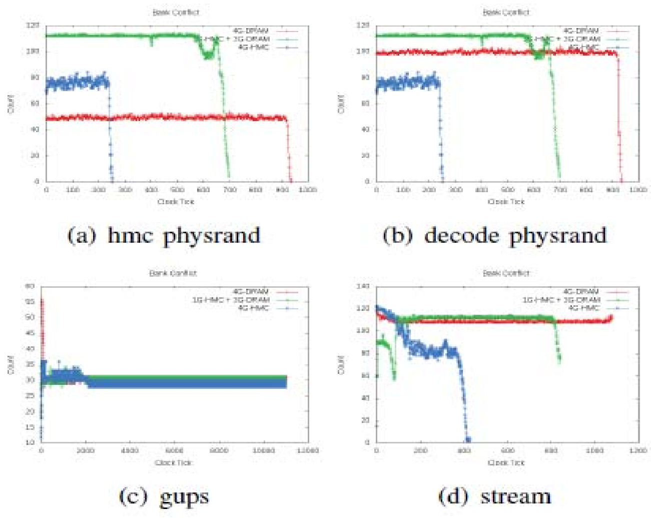


FIGURE III. BANK CONFLICT

FIGURE III describes the bank conflicts on each vault in DRAM, hybrid HMC+DRAM and HMC systems. Because the 4G-DRAM system only has two banks while the others both have eight banks, so in FIGURE III (a) and FIGURE III (b), the present hybrid HMC+DRAM system is inferior to DRAM and HMC systems. However, in FIGURE III (c) and FIGURE III (d), the bank-conflict performance of hybrid HMC+DRAM system is almost equivalent to the other systems.

## C. Read and Write Request

FIGURE IV describes the read and write requests in DRAM, hybrid HMC+DRAM and HMC systems, which directly reflect the throughout performance. We can observe that the throughput of hybrid HMC+DRAM system outperforms the DRAM system by about 1:5_. However, the HMC system still outperforms the hybrid HMC+DRAM system by about 2:1_. Considering the present system is consisted of 1G-HMC and 3G-DRAM while the HMC system of 4G-HMC devices, and the time spending on dead-lock avoiding, the throughout performance of our design is still effective.

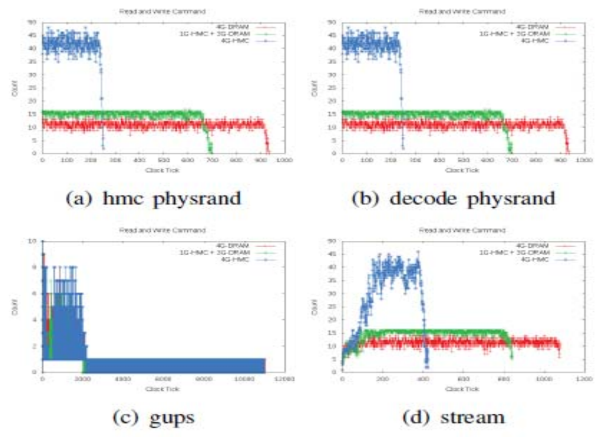


FIGURE IV. READ AND WRITE REQUEST

## D. Crossbar Latency

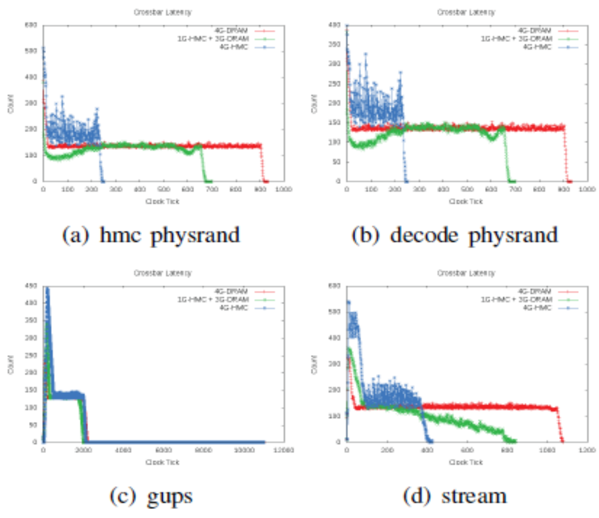
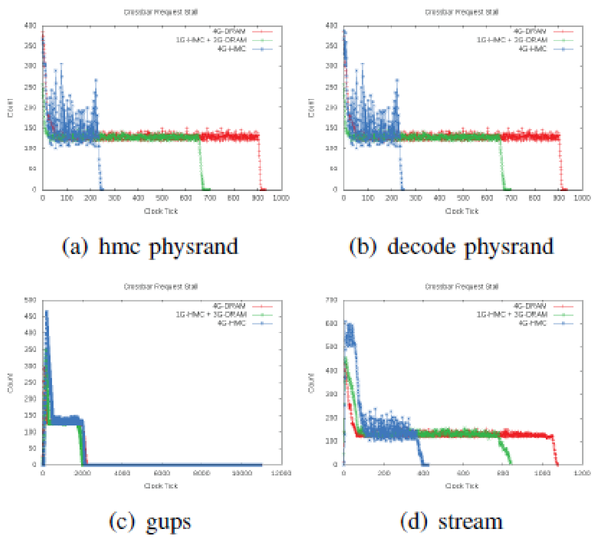


FIGURE V. CROSSBAR LATENCY

FIGURE V describes the crossbar latency in DRAM, hybrid HMC+DRAM and HMC systems, which directly reflect the I/O request delay in crossbar. In Figure 5(a), Figure 5(b) and Figure 5(c), the crossbar-latency performance of hybrid HMC+DRAM system is almost equivalent to the other systems. But in Figure 5(c), the present hybrid HMC+DRAM system outperforms DRAM and HMC systems by about 1.4 times.

## E. Crossbar Request Stall

FIGURE VI describes the crossbar request stalls in DRAM, hybrid HMC+DRAM and HMC systems, which directly reflect the I/O request response time in crossbar. We can observe that the I/O request response time in crossbar of hybrid HMC+DRAM system outperforms the HMC system by about 1.2 times, and is almost equivalent to the DRAM system.



(a) hmc physrand (b) decode physrand

(c) gups (d) stream

FIGURE VI. CROSSBAR REQUEST STALL

## V. Conclusion

The HMC is a promising alternative to DDRx memory due to its potential to achieve substantially improved memory bandwidth. However, the high price of a HMC device compromises cost efficiency when the device is lightly utilized. In this paper, we propose a new inexpensive HMC+DRAM hybrid main memory architecture to reduce the cost consumption. In order to manage such hybrid memories, we develop a LRU-based data distribution mechanism to determine the destination of particular data flow. Evaluations show that our scheme reduces the cost consumption of Main Memory by 48% on average with a negligible performance degradation compared to a current HMC-based system. Also, our architecture outperforms a current DRAM-based system by 2.1 times, and reduces the response delay by 1.9 times

## References

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," ACM SIGARCH computer architecture news, vol. 23, no. 1, pp. 20–24, 1995.

[2] P. Rosenfeld, "Performance exploration of the hybrid memory cube," 2014.

[3] J. Ahn, S. Yoo, and K. Choi, "Low-power hybrid memory cubes with link power management and two-level prefetching."

[4] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," in ACM SIGARCH Computer Architecture News, vol. 37, no. 3. ACM, 2009, pp. 267–278.

[5] J. S. Pak, J. Kim, J. Cho, K. Kim, T. Song, S. Ahn, J. Lee, H. Lee, K. Park, and J. Kim, "Pdn impedance modeling and analysis of 3d tsv ic by using proposed p/g tsv array model based on separated p/g tsv and chip-pdn models," Components, Packaging and Manufacturing Technology, IEEE Transactions on, vol. 1, no. 2, pp. 208–219, 2011.

[6] M. J. Khurshid and M. Lipasti, "Data compression for thermal mitigation in the hybrid memory cube," in Computer Design (ICCD), 2013 IEEE 31st International Conference on. IEEE, 2013, pp. 185–192.

[7] Y. Han, Y. Wang, H. Li, and X. Li, "Data-aware dram refresh to squeeze the margin of retention time in hybrid memory cube," in Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, 2014, pp. 295–300.

[8] S. Wang, Y. Song, M. N. Bojnordi, and E. Ipek, "Enabling energy efficient hybrid memory cube systems with erasure codes," in Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on. IEEE, 2015, pp. 67–72.

[9] G. Kim, J. Kim, J. H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in Proceedings of the 22nd international conference on Parallel architectures and compilation techniques. IEEE Press, 2013, pp. 145–156.

[10] S. H. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, F. Li et al., "Ndc: Analyzing the impact of 3d-stacked memory+ logic devices on mapreduce workloads," in Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on. IEEE, 2014, pp. 190–200.

[11] Y. Peng, B. W. Ku, Y. Park, K.-I. Park, S.-J. Jang, J. S. Choi, and S. K. Lim, "Design, packaging, and architectural policy co-optimization for dc power integrity in 3d dram," in Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE. IEEE, 2015, pp. 1–6.

[12] K. Chen, Z. Yu, C. Xu, J. Liu, and X. Li, "Improving power efficiency of gpgpu's global memory by a hybrid memory approach," in Information Science and Technology (ICIST), 2014 4th IEEE International Conference on. IEEE, 2014, pp. 660–664.

[13] L. A. Bathen and N. Dutt, "Havoc: a hybrid memory-aware virtualization layer for on-chip distributed scratchpad and non-volatile memories," in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 447–452.

[14] M. Qiu, Z. Chen, Z. Ming, X. Qin, and J. Niu, "Energy-aware data allocation with hybrid memory for mobile cloud systems."

[15] S. Lee, H. Bahn, and S. H. Noh, "Clock-dwf: A write-history-aware page replacement algorithm for hybrid pcm and dram memory architectures," Computers, IEEE Transactions on, vol. 63, no. 9, pp. 2187–2200, 2014.

[16] X. Ouyang, N. S. Islam, R. Rajachandrasekar, J. Jose, M. Luo, H. Wang, and D. K. Panda, "Ssd-assisted hybrid memory to accelerate memcached over high performance networks," in Parallel Processing (ICPP), 2012 41st International Conference on. IEEE, 2012, pp. 470–479.

[17] J. C. Mogul, E. Argollo, M. A. Shah, and P. Faraboschi, "Operating system support for nvm+ dram hybrid main memory." in HotOS, 2009.

[18] W. Vogels, "File system usage in windows nt 4.0," in ACM SIGOPS Operating Systems Review, vol. 33, no. 5. ACM, 1999, pp. 93–109.

[19] J. D. Leidel and Y. Chen, "Hmc-sim: A simulation framework for hybrid memory cube devices," Parallel Processing Letters, vol. 24, no. 04, p. 1442002, 2014.

[20] B. Jacob, S. Ng, and D. Wang, Memory systems: cache, DRAM, disk. Morgan Kaufmann, 2010.