HMDS: A Novel View of Data System Based on Hybrid Memory Architecture With Non-Volatility

Hao Liu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
liuhaosjtu@sjtu.edu.cn

Linpeng Huang
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
lphuang@sjtu.edu.cn

***Abstract*— In now Big Data era, rapidly growing data size and limited computing capability of current computer system have brought sharp contradiction between these two issues. So it is urgent to improve the data storage and processing ability for current computer system. Emerging byte-addressable, non-volatile memory (NVM) technology, can work like DRAM but with non-volatility, is expected to bring new opportunity of computing capability improvement in the near future. According to the hardware changes, it is necessary to reconsider the changes of system software and data storage methodology, especially the role of the data in the system. In this paper, we add the NVM to the computer architecture, and base on the hardware changes, we reconsider the design method of system software, among this we propose the concept of Data System, design a software defined interface, reconfig the system architecture without any real hardware changes. Finally, we present HMDS, a flexible hybrid memory data system architecture. Data system consider data as the center of a whole system, data storage is the foundation, data accessing and processing is the superstructure and all system parts as a service for the data. We think our architecture is novel and can bring data storage and processing a new perspective.**

Keywords—Big Data; NVM; Data System

I. INTRODUCTION

Nowadays, extremely huge volume of data is produced every day, data in computer system becomes more and more difficult to control, to access and to process. How to store and to process data is a key problem in now-days and it has also brought us many confuses. One important reason is that data processing ability of current computer system can't catch up with the rapid growth of data. To solve this problem, many approaches have been proposed on different aspects. Recent years rapid developed SCM technology, has brought new opportunity to solve the problem above. On system software level, several in memory file systems and databases were proposed which focus on the improvement of OS kernel and corresponding parts of system software to sup-port non-volatile memory. On programming mode level, Mnemosyne[11] provided a lightweight persistent memory program interface, ensure data consistency through a lightweight transaction mechanism. Above these work, different approaches are focus on different aspects, but all of these approaches only focus on one aspect, it lacks a general ideology about system design. In this article, we proposed the concept of data system, a novel system design perspective, change the design principle from traditional computing central to novel data central. We designed the Hybrid Memory Data System (HMDS), a data system based on hybrid memory architecture with both DRAM and SCM. We designed a software defined architecture to adapt the architecture and data access method changing. This paper was organized as follows: First, we describe the concept and design philosophy of data system (A), then present the system architecture overview will be presented in(B), software defined hybrid memory architecture will be presented in(C), multi mode data access interface in(D), and hybrid memory data system will be presented in(E). Finally, we give our comments on implementation and future work, and a brief summary of related research work from which HMDS was inspired.

Ⅱ. DATA SYSTEM

A. Data System Philosophy

In traditional computer system, hardware, software and applications compose the main components of the system. Data in the system, play a role of passive processing object. During the system running and data processing, data is transformed from storage side (disk, memory, file system, database) to processing side (CPU, GPU). But due to the huge performance gap between two parts, a large access latency has been produced and the whole system efficiency was severely impacted because of the deficient utilizing of the computing resource. Especially in now big data era, explosively increasing data volume makes the above contra-diction more and more seriously. To solve the problem, a lot of technologies and architectures are proposed, but these methods are mainly focus on a specific aspect, and it lacks a universal methodology and new perspective. Thus, we proposed the concept of data system as a new system design perspective. Ordinary data management method in existing computer systems is that organize data by file systems or databases. In the file system cases, no matter traditional block-based file system (ext3, ntfs, fat32) or temporary in memory file system (tmpfs, ramfs), data

is composed by many files, a file is a part of the data and the file system is a part of the system software. In database cases, data is standardized by a certain format and saved as many records. Both of the two methods illustrate that data is managed by a tool and this tool is only a part of the system. Figure 1 shows a traditional architecture of a computer system. In the system, data is isolated from the whole system and just as a part of sys-tem. This pattern has been used for many decades and have made few changes. Different with the traditional thinking, we first emphasize the data as the central of a system, other data management software such as file systems or databases in the system are just one existing form or management method of the data, supplying corresponding service for the data. *Figure 2* shows a philosophical image of the data system, it shows a high-level overview architecture of proposed data system, and the data sys-tem design is based on the following main principles:

1) Data is the central of the system.

2) Data storage is the foundation of the data system.

3) Data accessing and processing is the superstructure above the data storage.

4) Other parts of the system (hardware, software, etc.) is a service of the data in the system.

By this design methodology, we consider the data as the central of a system, liberate the data from the limited computing resource and bring a novel system design perspective face to the future.

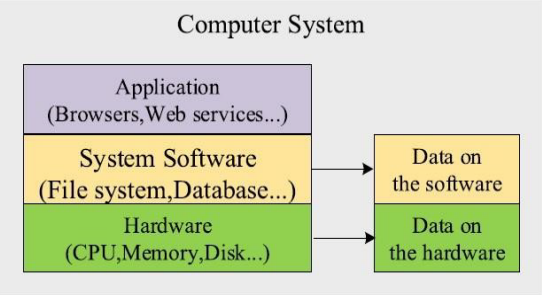

Figure 1: Traditional Computer System

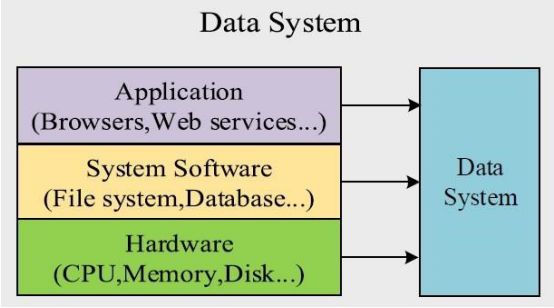

Figure 2: Novel Data System

B. Architecture Overview

In this paper, we first proposed a hybrid memory architecture as the main memory system. It uses both DRAM and SCM as working memory and data storage. We designed a software define interface library to define the level relationship between DRAM and SCM without any hardware layer changes. Then we designed hybrid Memory Data System (HMDS) to echo the central idea of this paper, HMDS is similar to an in memory persistent file system, and is the crucial part of our work. In HMDS, we extend operating system VMM mechanism to manage both DRAM and SCM, make it compatible with POSIX standard. In HMDS, we improved the access mode of traditional read/write data operation, developed a new SCM aware read/write data access mode. Besides, HMDS also optimized the memory mapped I/O by mapping unified memory directly into an applications address space, made load/store operation available and fast. This can avoid memory mapping overhead due to the first copy accessed page to DRAM in traditional file system. We call the SCM aware read/write and load/store access mode as multi-mode access mode. This access mode exploits SCMs byte addressable property to memory devices directly, therefor avoid the overheads of the block-based data access in traditional file systems, realized efficient access to SCM devices by applications. The system architecture overview is shown in *Figure 3*, and the other parts of the system will be illustrated in the next sections.

C. Software Defined Hybrid Memory Architecture

In this part, we designed a software-defined hybrid memory architecture to maximize the advantages of each kind of memory, DRAM and SCM. According to the different features of the upper application, software define interface chooses different memory architecture without any changes to the real hardware. We have implemented two system architectures: i) DRAM as the buffer of the SCM, ii) DRAM as the equal part of SCM. The comparison of the two architectures is illustrated in Figure 4. In the case of DRAM as the buffer of the SCM, metadata of the data system is mainly allocated in DRAM and data is allocated in SCM. This is equivalent to adding a layer of DRAM between SCM and CPU as a buffer. This design can take advantage of the merits of DRAM: fast, wear-leveling and maintain volatility when there is no need to guarantee data consistency in some specific applications. In the case that DRAM as one equal part of SCM, we can deployment unified byte-addressing mechanism then we can use the same CPU instructions to access two different memory devices. We defined a software defined interface library (SDILib), which

Table 1: Software Defined Interface Library

Command	Parameter	Comment
showarch	-all	Show the configuration of both dram and scm in the system.
	-scm	Show the configuration of scm in the system.
	-dram	Show the configuration of dram in the system.
defarch	-dram/scm	Define the system architecture that dram as the buffer of scm.
	-dram*scm	Define the system architecture that dram and scm as an equal part.

includes two shell commands showarch and defarch. It makes the HMDS can be aware of the existence of DRAM and SCM devices. Showarch is used to show the DRAM configuration in the system or SCM configuration in the system or both of them, with a parameter of -darm, -scm, and -all respectively. In defarch command, we implemented the reconfigurable memory architecture which mentioned before with the parameter of -dramscm and -darmscm respectively. The details of the two shell commands is depicted in *Table 1*.

D. Multi Mode Data Access Interface

In this part, we present two improved data access mode: SCM data mode and SCM memory mode. In SCM data mode, we access data through SDILib and HDMS. This mode improved the data access operations, including data read, write, open, close and so on, all of these operations are implemented based on POSIX standard. In SCM memory mode, HDMS designed a hybrid memory library (HMLib), provided serval APIs about memory mapping operations and implemented memory-like access to SCM, simplified the direct access to memory address by applications. These APIs are dedicated to memory allocating, unallocating, mapping and unmapping, complete the direct accessing to under persistent memory. The details of the defined APIs are illustrated in *Table 2*.

Table 2: Hybrid Memory Library

API	Parameter	Comment
hmalloc	(size t size)	Allocate a hybrid memory address space.
hmfree	(void *ptr)	Free a hybrid memory page address space.
hmmap	(void *start, size t len, int prot, int flags, int fd, off_t offset)	Mapping the physical memory pages to the application address space directly.
hmunmap	(void *start, size t len)	Recycle physical memory pages mapped to the application address space.
hmsche	(void *page start, int page flag, int page index)	Schedule a V_page or a NV_page between DRAM and SCM.

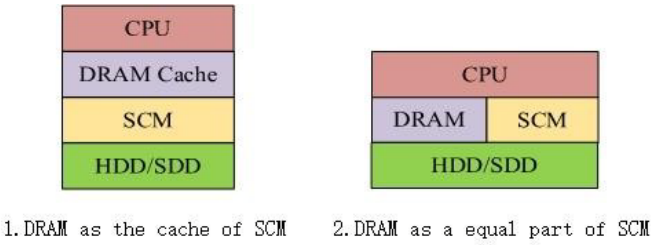

Figure 4: Software Defined Hybrid Memory Architecture

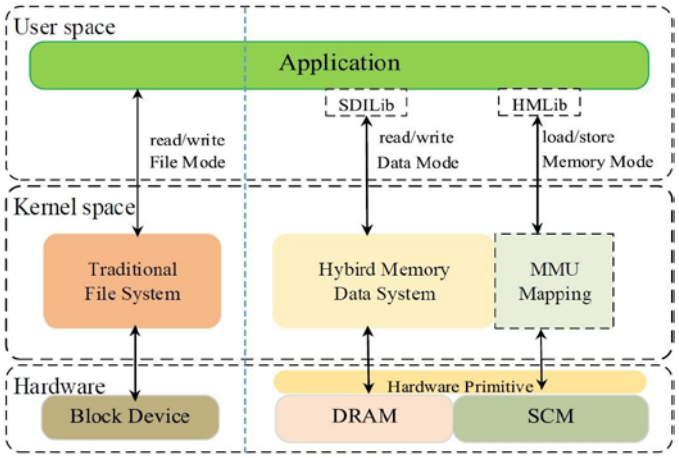

Figure 3: System Architecture Overview

E. Hybrid Memory Data System

In this part, the most important part of our work HMDS is described, a hybrid memory data system which support POSIX standard interface. The whole system layout is shown in Figure 5. On physical address space, there are metadata, memory mapping table, data log and data address space. The metadata part contains the information such as the size of DRAM and SCM and other root information about data system. The memory map-ping table part is used to maintain memory mapping management by using MMU mechanism in SCM memory mode. The information in the metadata and memory mapping table are used to build the mapping between the physical address and the virtual address. The data log part is used to save the system logging data which guarantee the data consistency. In the data address space part, the layout of HMDS is very similar to existing file systems, and HMDS divided it to four parts further which refer to super page, inode table, page bitmap and data pages respectively. In the super page, HMDS stores the pointer to the start address of the inode table. In the inode table part, a fixed size entry of 256 bytes is reserved for each inode, and it is simple to get a data pages metadata through its inode number and the start address in the inode table. The inode stores several pieces of metadata including checksum, owner uid, group gid, file mode, page count of SCM and DRAM, data size, access time and so on. The bitmap part stores three bitmap flags for each data page, NV_Bitmap, Use_Bitmap and Size_Bitmap. NV_Bitmap represents a data page in the memory space is volatile or non-volatile, Use_Bitmap represents a page that is already allocated in the memory address space has been used or still free. Size

Bitmap represents the size of the page, HMDS support for three types of page size: 4KB, 2MB and 1GB. As for the page layout, all metadata and inode table in HMDS is organized by a B-tree. The B-tree is used to represent both of the inode table and the data in inodes. By default, all meta data uses the 4KB size page and the data leaf pages can use all of the three kinds of size. An inode presents both of a directory data page ordinary data page. The directory data page is stored as the ordinary data page, except that their contents are lists of inode numbers. The data address space is composed by virtual address space that already allocated by HMDS, and the virtual address space is divided into the mapped space and the unmapped space. In the mapped space, there are used data page space and free page space. In the null data page, there is no really used space in mapped area. The HMDS data page layout is shown in Figure5.

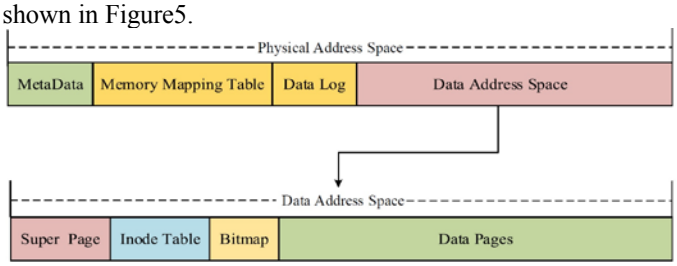

Figure5: Data System Layout

III. RELATEDWORK

A lot of work has been proposed about non-volatile memory architecture and system. We borrowed their ideas in our work but to the best of our knowledge, data system is a novel concept that haven't been proposed. On hardware level, PDRAM[4] designed a hybrid memory architecture based on PCM and DRAM, incorporated PCM into the main memory hierarchy of the computer system. NVMDuet[8], proposed a novel unified working memory and persistent store architecture. On system software level, several file systems have been designed for hybrid memory or Flash devices. PMFS[5] implemented a file system that exploits SCM's byte addressability to avoid overheads of block-oriented storage and enable direct memory access by applications with memory-mapped I/O. SCMFS[12] is implemented on the virtual address space and utilizes the existing memory management module of the operating system to help manage the file system space. In NVMFS[10], data storage was based on SSDs and resolved the problem of random write issue of SSDs. BPFS[3], used the technique called short-circuit shadow paping to provide atomic, fine-grained updates to persistent storage. On operating system level, [1] and [9] gave some implications about OS modification. On programming mode aspect, NV-Heaps[2] presented a light weight, high performance persistent object approach, provided a model for persistence and transactional semantics to guarantee system persistency and consistency, it is also an important issue in HMDS. Mnemosyne[11] declared global persistent data structure by the *pstatic* keyword and implemented several primitives for directly modifying these persistent variables and support consistent updates through a light weight transaction mechanism. On novel system architecture aspect, Memorage[7] leveraged the OS virtual memory manager to improve the performance of memory intensive workloads, provided an evolutional path from the memory-storage dichotomy to integration and co-management of memory and storage resources.

IV. CONCLUSION AND FUTURE WORK

This paper proposed a new concept of data system, designed a novel framework dedicate to data storage and processing based on hybrid memory architecture. We believe that it offers a new view or an opportunity to finally update the several decade years old computer system, and promote the redesign and innovation of current operating system to support new SCM memory device.

ACKNOWLEDGMENT

The work described in this paper was supported by the National Natural Science Foundation of China under Grant No.61472241 and the National High-tech R&D Program of China(863 Program) under Grant No.2015AA015303.

REFERENCES

[1] BAILEY,K.,CEZE,L.,GRIBBLE,S.D.,ANDLEVY,H.M.Operating system implications of fast, cheap, non-volatile memory. In Proceedings of the 13th USENIX conference on Hot topics in operating systems(2011),USENIXAssociation,pp.2–2.

[2] COBURN,J.,CAULFIELD,A.M.,AKEL,A.,GRUPP,L.M.,GUPTA,R.K., JHALA,R.,ANDSWANSON,S. Nv-heaps: making persistent objects fast and safe with next-generation,non-volatile memories. In ACM SIGARCH Computer Architecture News(2011),vol.39,ACM,pp.105–118.4

[3] CONDIT,J.,NIGHTINGALE,E.B.,FROST,C.,IPEK,E.,LEE,B.,BURGER,D.,ANDCOETZEE,D.Better i/o through byte-addressable, persistent memory. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (2009), ACM, pp.133–146.

[4] DHIMAN, G., AYOUB, R., ANDROSING, T. Pdram: a hybrid pram and dram main memory system. In Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE (2009), IEEE, pp.664–669.

[5] DULLOOR,S.R.,KUMAR,S.,KESHAVAMURTHY,A.,LANTZ,P.,REDDY,D.,SANKARAN,R.,ANDJACKSON,J.System software for persistent memory. In Proceedings of the Ninth European Conference on Computer Systems (2014),ACM,p.15.

[6] JUNG,J.Y.,ANDCHO,S. Memorage: emerging persistent ram based malleable main memory and storage architecture. In Proceedings of the 27th international ACM conference on International conference on super computing (2013), ACM, pp.115–126.

[7] LIU,R.-S.,SHEN,D.-Y.,YANG,C.-L.,YU,S.-C.,ANDWANG,C.-Y.M. Nvmduet: Unified working memory and persistent store architecture. In Proceedings of the 19th international conference on Architectural support for programming languages and operating systems (2014),ACM, pp.455–470.

[8] MOGUL,J.C.,ARGOLLO,E.,SHAH,M.A.,ANDFARABOSCHI,P. Operating system support for nvm+dram hybrid main memory. In HotOS (2009).

[9] QIU,S., ANDREDDY, A.N. Nvmfs: A hybridfile system for improving random write in nand-flash ssd. In Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on (2013),IEEE,pp.1–5.

[10] VOLOS,H.,TACK,A.J.,ANDSWIFT,M.M. Mnemosyne: Lightweight persistent memory. In ACM SIGARCH Computer Architecture News (2011), vol.39, ACM, pp.91–104.

[11] WU,X.,ANDREDDY,A. Scmfs: afile system for storage class memory. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (2011), ACM, p.39.