Fabric Attached Memory Emulation (FAME)의 성능 평가 및 분석

Safdar Jamil[1], 강현구[1], 안진우[1], 김영재[1], 마진석[2], 오명훈[2], 김학영[2]
[1]Sogang University, [1]ETRI, South Korea
{safdar, hyeongu, jinu37, youkim}@sogang.ac.kr, {majinsuk, mhoonoh, h0kim}@etri.re.kr

Fabric Attached Memory Emulation (FAME) Evaluation Study

Safdar Jamil[1], Hyeongu Kang[1], Jinwoo Ahn[1], Youngjae Kim[1], Jinseok Ma[2], Myeong-Hoon Oh[2], Hakyoung Kim[2]
[1]Sogang University, [2]ETRI, South Korea

요 약

With the introduction of non-volatile memories (NVMs), the concept of memory-driven computing (MDC) is gaining more attention. In MDC, *thousands of cores* will be accessing the global memory pool directly. Several ongoing projects on MDC are Hewlett Packard Enterprise (HPE) The Machine, Intel Rack-Scale Architecture, and UC Berkeley FireBox. HPE has introduced an open source emulator, FAME which is a collection of modules and virtual machines (VMs) that emulate HPE The Machine. Although, FAME provides sufficiently accurate emulation for application development for MDC, but still it has some modules such as librarian that can cause a major scalability and performance bottleneck for The Machine. In this study, we evaluate FAME, particularly librarian module which manages the allocation and deallocation of memory pool for system on chips (SoCs) and each SoC has to communicate with librarian to access memory pool. So, we run MDTest and Fio benchmark to test metadata and overall performance of FAME and found that as the number of SoCs increase the performance of librarian degrades gradually due to metadata contention.

1 Introduction

With emerging Non-Volatile Memory (NVM) technology, hybrid memory systems (HMS) are being proposed which is comprised of DRAM and NVM [1, 2]. NVM possesses characteristic such as persistency, byte addressability and fast read/write access. These characteristics make it a suitable candidate to be placed on processor memory bus so that processor can directly access that NVM. HMS proposed systems also address the memory-driven computing (MDC) paradigms. In MDC, a large collection of memory forms a memory pool and system on chips (SoCs), such CPUs, GPUs, and FPGAs, can directly access that memory pool without being dependent on processor memory controller [3]. Various future systems [4–6] have been proposed which will be comprised of *thousands of cores* and large DRAM and NVM.

Hewlett Packard Enterprise (HPE) The Machine [6] is one of the future system, which is based on paradigms of MDC. It is enabling SoCs to access that memory pool directly. HPE has also provided an open source emulation environment, FAME [7], to emulate The Machine. FAME deploys QEMU-KVM based virtual machines as SoCs and generates a memory pool on the host using Inter Virtual Machine Shared Memory (IVSHMEM) [8]. FAME also provide a collection of modules such as Librarian [9], Libpmem [10] which is based on Persistent Memory Development Kit [11] and Libfamatomic [12]. In The Machine, NVM memory pool is managed in terms of books and shelves as in a library. Book is the smallest granularity that can be assigned to each SoC and these books are organized on shelves. Librarian module of the FAME manages the allocation and deallocation of books for each SoC and each SoC has to communicate with it to read from or write to memory pool.

In this paper, we present an evaluation study of FAME as it is the only platform yet for development of MDC-based application. Thus, evaluation of this platform will help application developers design and develop applications by considering the performance and scalability. We show that FAME has various overheads such as VM switching and metadata management overhead. Meanwhile in the prototype of the Machine, real-time SoCs are deployed instead of VMs, hence there is no any VM overhead in rea-time system. But metadata management server (librarian) is deployed on top of SoCs and as it manages metadata of books so there is an overhead of handling metadata which can directly impact the performance of The Machine. Our finding in this study shows that the average metadata overhead of librarian is 79.47% for 8 nodes and the overall performance degradation of FAME is 80.65% for 8 nodes.

2 Background

2.1 Memory-Driven Computing

With the time, the focus of system infrastructure is moving from Compute- or Processor-Driven architecture to Memory-Driven architecture. In processor-driven architecture, every memory access is required to flow through the processor memory controller, which makes both processor and memory dependent of each other [3]. On the other hand, memory-driven architecture refers to an architecture where compute components can directly access memory without going through the processor memory controller.

HPE The Machine is an example of MDC. The Machine is a rack-based collection of nodes and each node comprises of SoCs, local DRAM, and large collection of NVMs. Figure 1 shows the overview of The Machine architecture. NVMs collectively form a memory pool and every SoC in The Machine can access that memory pool. The Machine exploits advanced optical intercon-

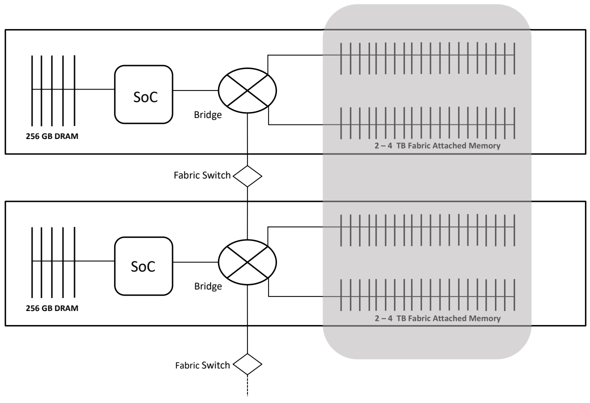


그림 1: Overview of The Machine Architecture

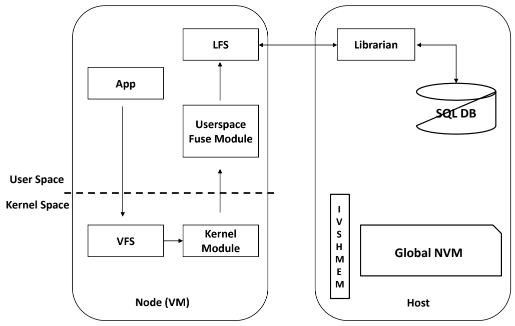


그림 2: FAME Working Flow

nect, which is termed as memory fabric [13]. Since it uses fabric, the latency of accessing memory pool is low i.e., near-uniform.

2.2 Fabric Attached Memory Emulation (FAME)

FAME emulates The Machine architecture. As The Machine contains nodes, FAME emulates those nodes in terms of QEMU-KVM based virtual machines (VMs). FAME exploits Inter-Virtual Machine Shared Memory (IVSHMEM) [8] feature of QEMU-KVM to generate memory pool on the host. Each VM is configured through QEMU-Command line to access that memory pool. FAME also consists of a customized Linux Kernel, Linux-L4FAME, which is installed in the host. The major customization in Linux-L4FAME is the drivers to communicate with memory fabric. Librarian, a tailored file system and metadata management server is also a part of the FAME environment. FAME also provides some low-level APIs such as libpmem and libfam-atomic for writing MDC-based application.

2.2.1 Librarian

Librarian consists of three modules, a fused-based file system, a database, and an asynchronous metadata management server. Librarian is named on the minimum granularity of memory portion, called book, that can be allocated to SoCs in The Machine. It only deals with metadata regarding allocation and deallocation of books to SoC. It implements a view of the memory pool and provides an interface for managing books in memory pool. In FAME, the SQLite database and metadata management server (librarian) runs on host while the file system daemon (LFS) runs on the VMs. Figure 2 shows the overall working flow of librarian. When an application on VM wants to access memory pool, it has to first get the metadata from librarian server's managed SQLite database using LFS and then it accesses the memory pool. The librarian server and LFS both are developed using python.

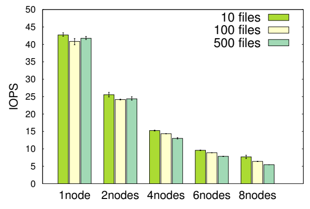
I *Librarian File System:* Librarian file system (LFS) is a fuse-based file system and provides all the functionalities that a file system provides. In The Machine, LFS runs on the SoC and communicates with librarian to get metadata for writing and reading on memory pool. But on FAME, LFS runs on VMs. In Figure 2, LFS is the daemon that runs on VM and communicates with Librarian server. The communication between librarian server and LFS is based on JSON formatted messages.

II *Librarian Server and SQLite Database:* In the prototype of the Machine, each rack deploys a separate librarian server that runs over the SoCs. The server is named as Top of Rack Management Server (TORMS). SQLite database is also deployed in that TORMS. On FAME, librarian server and SQLite database run on host and provide an interface of the memory pool. The metadata of books allocated to SoCs is stored in SQLite database and managed by librarian server. When an application on the SoC needs to write or read data to/from memory pool, LFS daemon will communicate with the Librarian server and get the metadata regarding the allocated books for that SoC and then SoC will be able to read/write data from/to memory pool.

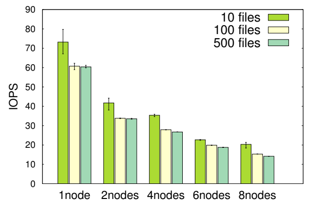
3 Evaluation

3.1 System Setup

We configured FAME on our in-house system which has Intel Xeon Processor E5-2640 v4 10 Cores and 64 GB DDR4 main memory. Due to limited resources, we only perform evaluation till 8 nodes. For testing scalability of FAME and metadata management of librarian, we used two benchmarks, Fio [14] and MDTest [15]. For scalability test, we used two workloads of Fio, one is when each job operates 4KB per IOs and other is 1MB per IO. For metadata evaluation, we choose MDTest as it is an HPC-based metadata benchmark and provides various insights related to metadata performance. We used three configurations of MDTest, (i) 10 files per directory, (ii) 100 files per directory, and (iii) 500 files per directory.

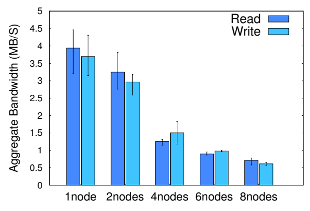


(a) File Creation

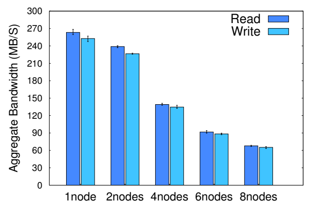


(b) File Deletion

그림 3: Libraraian Metadata Overhead



(a) 4KB IO



(b) 1MB IO

그림 4: Scalability of FAME

3.2 Metadata Overhead of Librarian

As librarian only deals with metadata, thus in this experiment we run MDTest which is an HPC-based metadata testing benchmark. In these tests, we only evaluated file creation (figure3(a)) and deletion (figure 3(b)) operations as these two operations are more critical in metadata management. Figure 3 shows the result of MDTest and we can observe that as the number of nodes is increasing the IO operations per second (IOPS) is decreasing linearly and this is caused due to overhead of SQLite and JSON based communication between VMs and librarian server. In provided MDC architecture, the management of memory allocation to each SoC will become major bottleneck as we have shown that the performance of management of metadata is degrading as the number of SoCs increases.

3.3 Scalability Evaluation of FAME

To evaluate how is the overall performance of FAME, we increased the number of nodes and run Fio benchmark on all the nodes simultaneously and measure the aggregated bandwidth of Read and Write operations. Our Fio configuration is explained in Table 1. Figure 4 shows that as we are increasing the number of nodes, read and write performance of FAME is linearly decreasing. As the number of VMs increase in FAME, the management of those VMs becomes crucial for librarian as it has to manage metadata for all the VMs which cause a severe performance degradation in overall. So, the performance of FAME decreases gradually as VMs increased.

4 Conclusion

The Machine is one of the ongoing projects that address MDC paradigms. FAME is the emulation environment of The Machine. The metadata management overhead cause performance degradation as number of SoC increases. In MDC, thousands of SoCs are going to access a large memory pool. But if the current system with all these overheads will be deployed to manage thousands of SoCs, then we will not be able to achieve high performance.

표 1: Fio description

Number of Jobs	16
Runtime	60 seconds
Types of IO	mmap, posixio, and sync
Job 1 - Job 5	mmap engine
Job 6 - 10	POSIXIO engine
Job 11 - Job 15	sync engine
Job 16	mmap engine with multithreading

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00503, Researches on next generation memory-centric computing system architecture).

참고 문헌

[1] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable high performance main memory system using phase-change memory technology,” in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA ’09, pp. 24–33, ACM, 2009.

[2] L. E. Ramos, E. Gorbatov, and R. Bianchini, “Page placement in hybrid memory systems,” in *Proceedings of the International Conference on Supercomputing*, ICS ’11, pp. 85–95, ACM, 2011.

[3] G.-Z. Consortium, “Gen-zcore specification 1.0,” tech. rep.

[4] K. Asanović, “Firebox: A hardware buidling block for 2020 warehuose-scale computers (keynote),” in *FAST*, 2014.

[5] Intel, “White paper: Intel rack scale design,” tech. rep.

[6] HPE, “The Machine.” https://www.labs.hpe.com/the-machine.

[7] HPE, “Fabric Attached Memory Emulation.” https://github.com/FabricAttachedMemory/Emulation.

[8] QEMU, “Inter-Virtual Machine Shared Memory.” https://doc.dpdk.org/guides-16.04/prog_guide/ivshmem_lib.html.

[9] HPE, “The Librarian File System Suite.” https://github.com/FabricAttachedMemory/tm-librarian.

[10] “Library for persistent memory programming.” https://github.com/FabricAttachedMemory/nvml.

[11] “Persistent Memory Development Kit.” http://pmem.io/pmdk/.

[12] HPE, “Library of fabric attached memory atomic.” https://github.com/FabricAttachedMemory/libfam-atomic.

[13] T. P. Morgan, “HPE’s Superdome gets and SGI NUMALINK Makeover.” https://www.nextplatform.com/2017/11/06/hpes-superdome-gets-sgi-numalink-makeover/.

[14] J. Axboe, “Flexible I/O Tester.” https://github.com/axboe/fio.

[15] MDTEST, “MDTest: HPC benchmark for metadata performance.” https://sourceforge.net/projects/mdtest/.