Ziting Liu
Shiyue Hou
EECE 5643

Progress Report

Introduction

Solid state drive (SSD) is a storage device that uses integrated circuit assemblies to store data. Unlike traditional HDD, it has no moving parts. There are two key components in an SSD: controller and memory. Controller is one of the most important factors that affects SSD performance. SSD controllers include electronics that bridge the flash memory components to the SSD input/output interfaces. There are two major memory types: Flash and DRAM. DRAM memory has advantages in terms of performance but usually cost more than NAND flash memory. Other than the two components mentioned above, there are Cache, Capacitor, and Host interface. A capacitor, or a battery, is used to make sure data integrity when power was cut off. Finally, Host interface is the physical connector managed by the controller, there are many different interface protocols. Some of the different types of SSD:

- SATA SSD
- M.2 SSD
- NVMe SSD
- SSD AIC

Due to SSDs' high throughput, low latency and increasingly lower costs, SSDs have become mainstream in many cloud enterprise such as Amazon AWS, Google Cloud etc, as well as consumer device such as laptop, workstation. As the IO requirement increase at cloud enterprise, data center and consumer applications, conventional SSD architecture can not match the high IO demand.

In the past years, most of HDD and SSD use of the Serial Advanced Technology Attachment (SATA) protocol, which is improvement to the existing Parallel ATA technology. Over this, SATA protocol have proven to be low efficient for fast IO operate and high bandwidth of SSDs. New protocol such as NVM Express have overcome the traditional HDD barriers, which can enable millions of IO operations per second (IOPS) and high bandwidth of SSDs. NVMe utilize multi-queue SSD (SSD), which can allow SSDs front end to manage IO request according to the SSDs free space.

As SSDs and their associate protocol development, the research needs to have an accuracy and stable emulator to emulate the SSDs characteristics. In our project, we select FEMU. FEMU is an emulator based on QEMU/KVM virtual machine. It can accurately emulate SSDs with multiple SSDs type. Customer can choose the SSDs type according to research request. Moreover, FEMU is an open-source emulator software. It is free provide to customers. FEMU also has great scalability, which can support 32 channels and chips.

In our project, we will try to figure out how flash memory controller affects the performance. A flush memory controller manages data stores on false memory and communicates with host. Upon initialization, the controller will make sure the device is running properly. During I/O operations, the host needs to communicate with the controller. I/O speed is limited by the performance of the controller. Inside the controller, the storage is managed by the Flash Translation Layer (FTL). This is like the Page Table in computer systems, which will translate between virtual and physical addresses. The physical location in and SSD are defined by block ID, page ID, and sector ID.

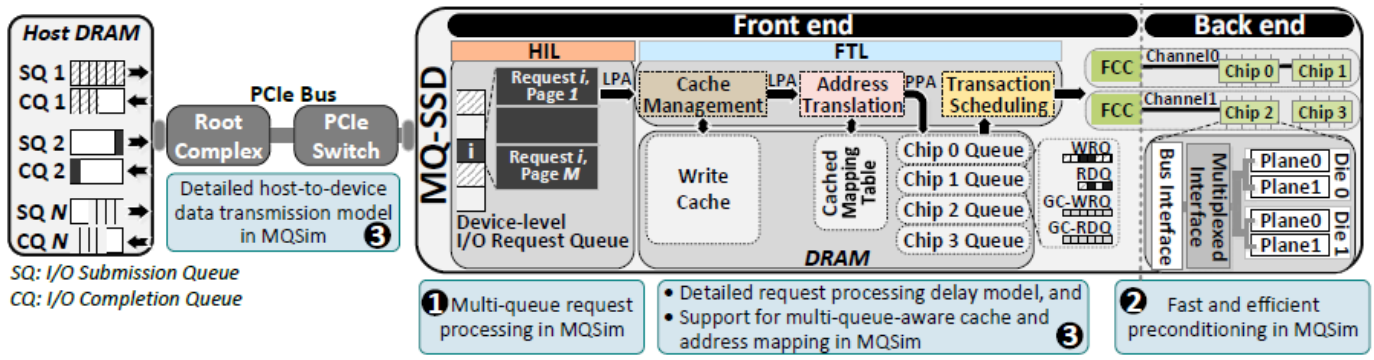

Figure 1: Architecture of SSDs [1]

Background

In this section, we will introduce a general background on multi-queue SSD device and FEMU emulator. Firstly, we will introduce the SSDs Host-Interface Logic (HIL) of SSDs front end. Second, we will talk about the Flash Translation Layer (FTL) of front end. Third, we will discuss the SSDs backend. Finally, we will discuss the FEMU emulator architecture. Figure 1 is architecture of SSDs.

Host-Interface Logic (HIL) play a very important role to utilize NAND flash memory parallel to improve SSDs throughput, IOPS and bandwidth, as decrease response time. The SATA protocol general used for traditional SSDs. SATA adopt Native Command Queue, which support parallel execute IO request based on SSDs space available. NVMe is design for SSDs. NVMe enable SSDs scalable, high throughput and bandwidth, lower response time within communication by PCIe bus. At Host DRAM, application submit a job directly insert into the submission queue (SQ), SSDs select job form submission queue and perform IO operate and return operate result into completion queen (CQ). Modern SSDs have broadly adopted NVMe protocol.

Flash Translation Layer (FTL) run at microprocessor with internal of SSDs. When SSDs select a job from host DRAM submission queue, HIL insert job into Device-Level IO Request Queue. FTL will check job read/write operate. For write operate, the job writes trigger FTL write cache management unit to store data and ask HIL to prepare a response. For read operate, FTL translate the job logic address into physical address to process. When the job finish, FLT will ask HIT to send a response to the host. Wearing level and garbage collection also execute at FTL. Wearing level is a SSDs technique to increase SSDs lifetime. The principle of wear level is to evenly distribute write at all of SSDs blocks, only concentrated in one part of blocks. All cell received the same number of writes, to avoid writing too often in specific one. Garbage collection is an optimize machine to release invalid block. When data deleted from file system, FTL marked erase pages as invalid, moving valid pages into a free block and then erase invalid block.

The modern SSDs are produced by NAND flash memory chip. Inside of NAND flash memory chip, A bunch of pages grouped as a block. Data erases operate take place at block. Flash writes take place at pages. Figure 2 show the structure of backend. The back end includes multiple bus channel. The channel connects frontend and backend. Each channel connects multiple chips. Each chip contains multiple die. Every die can independently perform memory command. Each die separates into multiple planes. Each plane divided into multiple blocks. Block made up multiple pages. One page default 4KB.

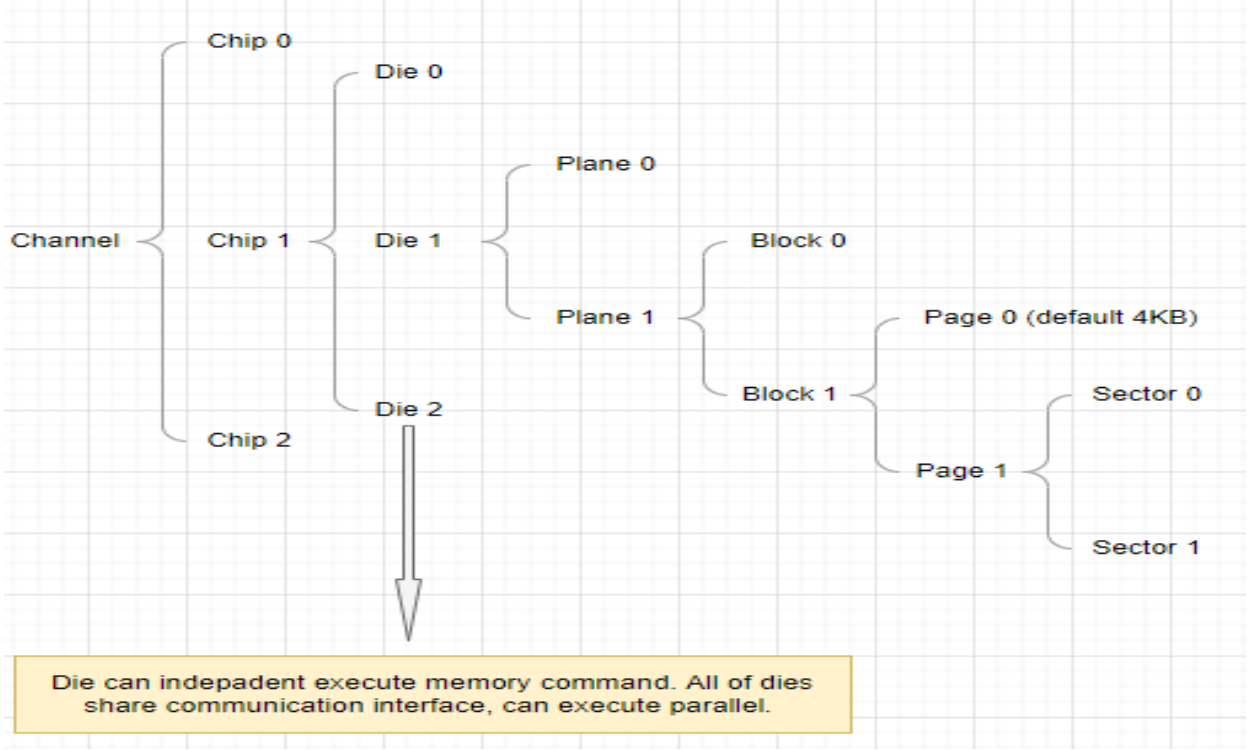

Figure 2: Structure of backend

For the fast, cheap, accurate, scalable and extensible FEMU Emulator. FEMU exposed to the Guest OS as an NVMe block device. FEMU provide multiple SSDs type:

- Whitebox mode: emulate Software-Defined SSD or OpenChannel SSD with Flash Translation Layer.
- Blockbox mode: emulate traditional NVMe SSD with Flash Translation Layer. It also contains a page-level mapping.
- ZNS mode: expose NVMe interface for host to directly read and write.
- NoSSD mode: emulate a convention NVMe SSD without FTL.

As hardware design, FEMU support run at Application+OS+NVMe system structure. FEMU can enable Software Defined Flash to modify application, OS and SSD FTL. Moreover, FEMU also modify SSDs FTL to apply new FTL algorithms to emulate NVMe SSD. Figure 3 is FEMU structure.

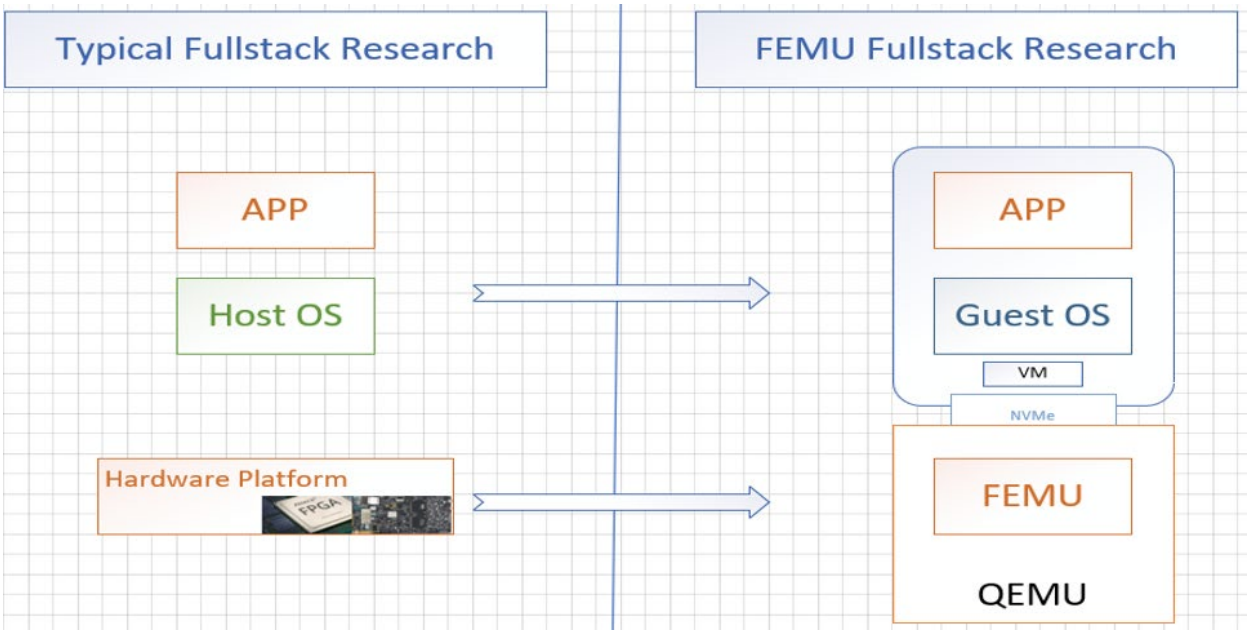

Figure 3: FEMU structure

Project Progress

In our project, we will be using FEMU, a QEMU-based and DRAM-backed NVMe SSD Emulator. FEMU is a fast, accurate, scalable, and extensible NVMe SSD emulator. Based upon QEMU/KVM, FEMU is exposed to Guest OS as an NVMe block device. It supports emulating different types of SSDs, including Whitebox mode, Blackbox mode, ZNS mode, and NoSSD mode. In this project, we will be focusing on the Blackbox mode, with FTL managed by the device. With this mode, we will be able to emulate current commercial SSDs.

Although we have learned the concepts about SSD and FEMU, but none of us had experience running qemu on a Linux machine. Therefore, we are running into many troubles when we try to get the FEMU running. The condition for running a qemu is very strict, the FEMU installation process uses GUI, which is not supported by remote servers. After that, I tried setting up a virtual machine using VirtualBox on windows machine, but it also doesn't work, because a virtual machine Linux does not support KVM (kernel-level virtual machine). Out plan for now is Shiyue will borrow a machine from his colleague, which already has FEMU environment ready, when they are done using it.

Reference

1. Tavakkol, Arash, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose and Onur Mutlu. "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices." *FAST* (2018).
2. Li, Huaicheng, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Matias Bjørling, and Haryadi S. Gunawi. "The {CASE} of {FEMU}: Cheap, accurate, scalable and extensible flash emulator." In *16th USENIX Conference on File and Storage Technologies (FAST 18)*, pp. 83-90. 2018.