Improving Performance by Bridging the Semantic Gap between Multi-queue SSD and QEMU I/O Virtualization [*]

Tae Yong Kim[†*], Dong Hyun Kang[†], Dongwoo Lee[†], Young Ik Eom[†]
[†]*Sungkyunkwan University, South Korea*
[*]*Samsung Electronics, South Korea*
{taeyongkim, kkangsu, dwlee, yieom}@skku.edu

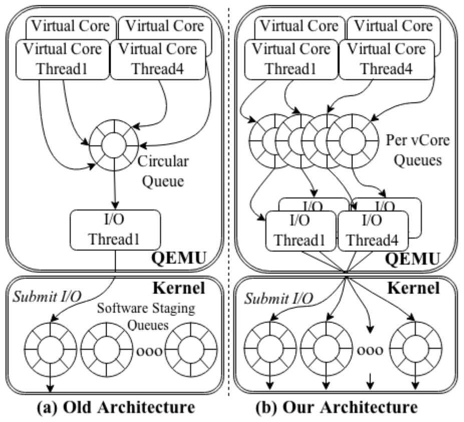

Figure 1: Comparison of system architectures consisting of host and guest

Motivations and Introduction: Multi-queue Solid State Drive (SSD) with PCI Express dramatically accelerates the I/O performance of secondary storage by exploiting the parallelism in SSD (e.g., NVM express SSD). Unfortunately, it does not directly result in the overall system performance. This is because the block layer of operating system waits to hold the lock whenever an I/O request is sent from the single request queue to the multi-queue SSD. To address this *lock contention* problem, previous work proposed a block layer that efficiently improves the system performance by using two levels of queue, which consist of software staging queue and hardware dispatch queue. However, in virtualized environment based on Quick Emulator (QEMU), the proposed mechanism can not take the advantages of multi-queue SSD because QEMU also has another *lock contention* problem, which is incurred by using the Virtio-Blk-Data-Plane technique. This technique has been introduced to improve the performance of virtual machines and it provides a dedicated thread each virtual core instead of using replicated layers between the host and guest (e.g., I/O scheduler and block layer). The dedicated thread directly issues I/O requests of its virtual core to the shared circular queue in QEMU after holding the lock on the single shared circular queue. In addition, a dedicated I/O thread, which is responsible for submitting I/O requests in the shared circular queue to the host kernel, periodically tries to hold the lock on the shared circular queue. As a result, QEMU significantly suffers from the frequent *lock contentions*. To understand the impact on the performance of virtual machines using the multi-queue SSD, we measured the random read performance under FIO benchmark and found that the IOPS of 4KB read decreases as increasing the number of jobs in FIO benchmark. is higher. This observation clearly reveals that existing Virtio-Blk-Data-Plane has *scalability* issue when multi-queue SSD is used as a storage, where we call it *semantic gap* between the host and guest.

Multi-queue and Multi-IO Thread: In this paper, we propose a novel approach to solve both *lock contention* and *scalability* issue for multi-queue SSD. Figure 1 shows the architecture and the overall abstractions of our design. To distribute the *lock contention*, we first eliminate a single circular queue in QEMU and then allocate a dedicated queue for each virtual core. To mitigate the *scalability* issue, We also improve the parallelism of I/O request by using multi-IO threads, which can submit I/O requests of each dedicated queue to host kernel simultaneously. For evaluation, we developed our new approach based on Linux 3.17 and modified both Virtio-blk front-end driver of QEMU 2.1.2 and Virtio-blk back-end driver of guest OS for multi-queue and multi-I/O thread architecture. All experiments were performed on null block device, which simulates a multi-queue virtual device by receiving I/O requests and acknowledges I/O completions immediately. We compared the performance of our architecture and existing architecture on multi-core system with FIO benchmark. We also performed experiments by varying the number of jobs from 1 to 8 for investigating the *scalability*. As a result, in case of 4KB random read, IOPS was significantly improved by up to 238%. Moreover, performance degradation completely disappeared even though the number of jobs was increased.

[*]This research was supported by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA (National IT Industry Promotion Agency).