1. mClock: Handling Throughput Variability for Hypervisor IO Scheduling (OSDI 2010)

This paper introduces a novel algorithm for IO resource allocation in a hypervisor. Hypervisors are responsible for multiplexing the underlying hardware resources among virtual machines (VMs) while providing them the desired degree of isolation using resource management controls. There are three controls that hypervisor multiplexes hardware resources between VMs. First, Reservation means minimum guarantee. Second, Limits means maximum allowed. Third, Shares means proportional allocation. Existing methods provide many knobs for allocating CPU and memory to VMs, but support for control of IO resource allocation has been quite limited. IO resource management in a hypervisor introduces significant new challenges and needs more extensive controls than in commodity operating systems. Their algorithm, mClock, supports proportional-share fairness subject to minimum reservations and maximum limits on the IO allocations for VMs. mClock is an IO scheduler that provides all three controls mentioned above at a per-VM level. It uses two main ideas: multiple real-time clocks and dynamic clock selection.

This paper has a strength that mClock is the first scheduler to provide such controls in the presence of capacity fluctuations at short time scales. In addition, they also demonstrate dmClock, a distributed version of the algorithm that can be used in clustered storage systems, where the storage is distributed across multiple nodes.

But it is weakpoint that the network is assumed to be over-provisioned and in a clustered environment, a host level algorithm alone cannot control the LUN capacity available to a specific host due to workload variations on other hosts in the cluster. Hence, any solution local to a single host is unable to provide guarantees across a cluster and is not sufficient for our use case.

2. FlashFQ: A Fair Queueing I/O Scheduler for Flash-Based SSDs (ATC 2013)

NAND Flash storage devices achieve fast I/O without mechanical seek/rotation delay. But, fairness is important in multi-task systems and clouds. Existing fair I/O schedulers are mostly timeslice-based (e.g., Linux CFQ, FIOS). These timeslice schedulers may exhibit poor responsiveness, particularly when there are large number of co-running tasks. So, timeslice schedulers can't easily exploit device parallelism.

This paper presents FlashFQ, a new Flash I/O scheduler that attains fairness and high responsiveness at the same time. It enhances the start-time fair queueing schedulers with throttled dispatch to exploit restricted Flash I/O parallelism without losing fairness. It also employs I/O anticipation to minimize fairness violation due to deceptive idleness.

FlashFQ algorithm is based on the SFQ(D) (Depth-controlled Start-time Fair Queuing) algorithm. FlashFQ maintains the virtual time for each input / output stream to manage the progress of it in proportion to the assigned weight. Virtual time represents the total amount of service received by the stream, and fair queuing algorithms aim to schedule all streams to maintain the same virtual time for fair service proportional to the weight.

FlashFQ has strength that in their real applications evaluations, they used mix of different request sizes and types showing that improve the fairness among multiple concurrent tasks. Also, it exploits fast random access of flash storage by issuing IO requests from different tasks while maintaining overall fairness based on fair queueing.

However, it is weakpoint that FlashFQ assumes only a single queue SSD. FlashFQ needs to know and calculate the minimum VT for every request at every moment. In multi-queue SSDs, it can occur scalability issue.