ECE5658: Operating Systems Design

Paper Reading Assignment

2019712410 Jeon Hyung-Joon

mClock: Handling Throughput Variability for Hypervisor IO Scheduling

Summary:

The objective of this paper is to present mClock, a tag-based scheduling algorithm for input-output (IO) resource allocation of Virtual Machines (VMs). It is composed of three modules: Tag Assignment, Tag Adjustment, and Request Scheduling. The Tag Assignment module assigns one of three tags—the R tag (for "reservation"), the L tag (for limit), and a P tag (for proportional share)—to each of the IO requests from VMs. The Tag Adjustment module calibrates P tags with respect to real time. The Request Scheduling module makes decisions on IO scheduling based on the existence of tags for each VMs. By utilizing multiple real-time clocks, the tags are assigned on behalf of each clock, and the mClock algorithm executes dynamic selection of the tags to perform either constraint-based or weight-based scheduling. In addition, the distributed version of this algorithm—dmClock—is also introduced in the paper. It uses the mechanism as similar as that of mClock, and it is designed for use in clustered storage environments.

Strengths and Weaknesses:

Based on the experiments using the prototype implementation in the VMware ESX server hypervisor, the mClock algorithm effectively provides VM-specific high quality of service (QoS) for each VM regardless of whether the overall throughput fluctuates. It can support proportional-share fairness across VMs in terms of minimum reservations and maximum limits on IO allocations. Such controls are quite effective in isolating performance amongst VMs and reducing latency of applications, and similar strengths can be concluded in case of dmClock, a variation of the algorithm for use in clustered system architectures.

However, it should be noted that the paper implicitly assumes similar priorities and task independence between VMs. For some systems, the VMs have dependence in performing operations, and the VMs may not have equal priorities in the need for allocating IO resources. Plus, it is assumed that the system capacity is always sufficient for serving the total minimum reservations of all authorized clients, but this may not be the case in practice. Therefore, it might be possible to cast doubt on the effectiveness of the mClock algorithm when it comes to the cases that were not considered throughout the paper.

Suggestions for Improvement:

I suggest augmenting the algorithmic details to deal with a set of VMs requiring non-uniform, prioritized requests for IO resource allocation. From such improvement, I believe the algorithm can be more generalized and become usable in even more various types of machine system architectures. Moreover, as the authors of the paper state, the effectiveness of mClock for array-level scheduling or for other hardware resources can be explored further in the future. The paper can give insights on how the mechanisms of the algorithm sets parameters to meet service-level agreements (SLAs) to the application-level.