GREM: Dynamic SSD Resource Allocation In Virtualized Storage Systems With Heterogeneous VMs
Zhengyu Yang¹, Jianzhe Tai², Ningfang Mi¹
Northeastern University¹ VMware Inc²

Abstract
In a shared virtualized storage system that runs heterogeneous VMs with diverse IO demands, it becomes a problem for the hypervisor to cost-effectively partition and allocate SSD resources among multiple VMs. We design a Global SSD Resource Management solution - GREM, which aims to fully utilize SSD resources as "smart" cache under the consideration of performance isolation.

Background
There are two straightforward approaches: (1) Equally assigning SSDs to each VM; and (2) Managing SSD resources in a fair competition mode. Unfortunately, they cannot fully utilize the benefits of SSD resources, particularly when the workloads frequently change and bursts or spikes of I/Os occur from time to time.

Design
GREM takes dynamic IO demands of all VMs into consideration to split the entire SSD space into a long-term zone and a short-term zone and cost-effectively updates the content of SSDs in these two zones.

![Diagram of GREM architecture](image)

1. GREM can adaptively adjust the reservation for each VM inside the long-term zone based on their IO changes.
2. GREM can dynamically partition SSDs between long- and short-term zones during runtime by leveraging the feedbacks from both cache performance and bursty workloads.

![Diagram of GREM partitions](image)

Methods
GREM has a decision maker which detects bursty and non-bursty I/Os. It is based on both workload and cache hit feedbacks. During bursty phase, GREM aggressively allocate more space of short-term zone based on workload change, otherwise, it conservatively allocate space for short-term zone based on the cache hit status.

![Workflow of decision maker](image)

Experimental Results

![Graphs of IO costs of traces](image)

Dynamic Partitioning

![Graphs of Hit Ratio](image)

Conclusions
Experimental results show that GREM can capture the cross-VM IO changes to make correct decisions on resource allocation and thus obtain high IO hit ratio and low IO costs, compared with both traditional and recent caching algorithms.

References