Improving Flash Cache Management for Virtualized System

Dong-Hyun Song[†§] Youngjae Lee[†] Jin-Soo Kim[†]
[†]Sungkyunkwan University, Korea [§]Samsung Electronics, Korea
{songdh418, yjlee, jinsookim}@csl.skku.edu

Motivation: Flash-based solid state drives (SDDs) are increasingly being used as a cache device in front of the back-end storage based on hard disk drives (HDDs) to accelerate I/O performance. In virtual environments, SSDs are widely deployed as shared cache devices for guest virtual machines (VMs). Previous studies have focused on cache space partitioning algorithm between VMs. S-CAVE [1] proposed how the hypervisor can fairly reallocate the cache space with a cache demand metric, called rECS (ratio of Effective Cache Space). However, S-CAVE did not consider the shared cache blocks. In the case of a plurality of VMs created from one base image, the same HDD blocks are requested frequently by different VMs and cached in a private cache space of each VM. A deficient cache management will reduce available cache space and degrade I/O performance. Therefore, the hypervisor has to take into account cache partitioning policies and cache replacement algorithms with the shared cache blocks to prevent I/O performance degradation.

Our Solutions: In this paper, we present a cache management architecture, which considers cache blocks shared among VMs. Specifically, we propose two cache management techniques: (1) a novel cache partitioning metric, called rCrS (ratio of re-accessed cache blocks with ratio of shared cache blocks), which provides better fairness than rECS, and (2) an efficient cache replacement algorithm called T-CLOCK (Tag-based CLOCK), which is an optimized CLOCK algorithm for multiple VMs. In every time interval, the hypervisor chooses two VMs by comparing the rCrS values of VMs to adjust their cache space: one is the highest scored VM with increasing demand; the other is the lowest scored VM with decreasing demand. The private CLOCK hand of the victim VM finds and evicts the coldest cache blocks forcefully using the T-CLOCK algorithm. In order to implement our techniques we define a global metadata array for all cached blocks.

Global Metadata: Each global metadata entry points to a cache block address (CBA) of the SSD, which is mapping with a logical block address (LBA) of the HDD by a hash table. An entry consists of two fields: VM tag bits and CLOCK bits. Each VM has one VM tag bit and one CLOCK bit as a metadata. These bits indicate which VM has accessed and re-accessed the corresponding cache block. Also they identify that the cache block is private to a VM or shared by VMs through VM tag bits. Once a VM accesses a cache block, its VM tag bit is set to 1. When the VM re-accesses the cache block, its CLOCK bit is changed to 1. On the other hand, if all bits are 0, the cache block is allocatable. If all CLOCK bits are 0, the cache block is evictable.

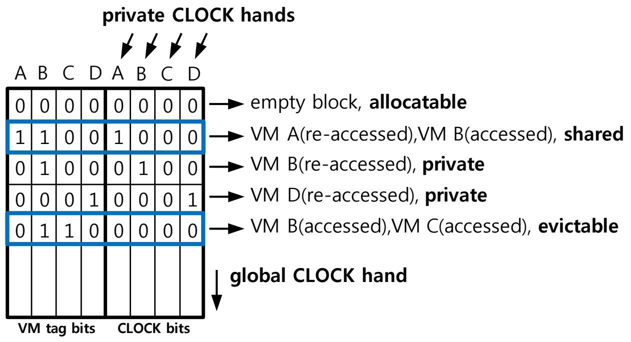


Figure 1: An example of global metadata array for 4 VMs

rCrS: The rCrS is an efficient cache partitioning metric with shared cache blocks. This value is a product of two ratios. The first one is the ratio of the number of re-accessed cache blocks to the total number of private cache blocks allocated to a VM. The second is the ratio of the number of shared cache blocks to the total number of re-accessed cache blocks. Even when the first ratio is lower than other VMs ratios, if the second ratio is higher, the calculated rCrS value can be higher than other VMs rCrS values. In other words, our cache demand metric avoids choosing a VM with a higher proportion of shared cache blocks as a victim.

T-CLOCK: The T-CLOCK algorithm tends to prolong the lifetime of shared cache blocks. Whenever it needs a cache space reduction or cache replacement, the private CLOCK hand of each VM traverses the global metadata array to find the coldest of the cache blocks. The private CLOCK hand tests exclusively its CLOCK bit. If the CLOCK bit is 1, then it is reset to 0 and the CLOCK hand will point to the next array index. If all CLOCK bits are 0, then the T-CLOCK algorithm reclaims the free cache block and the CLOCK hand points to next index. However, if its own CLOCK bit is 0 but another CLOCK bit is set to 1, this shared cache block survives until all CLOCK bits becomes 0.

Evaluation: We evaluate the performance of the proposed scheme on a trace-driven simulator with real-world traces. We chose read intensive workloads and executed two workloads on two VMs separately with limited cache space. The experimental results show that our solution achieves 31%-63% higher cache hit ratios than S-CAVE. In addition, the proposed scheme achieves 3.17x higher I/O throughput than S-CAVE.

References: [1] Tian Luo, Siyuan Ma, Rubao Lee, Xiaodong Zhang, Deng Liu, and Li Zhou. S-CAVE: Effective SSD Caching to Improve Virtual Machine Storage Performance. In PACT13.

Improving Flash Cache Management for Virtualized System

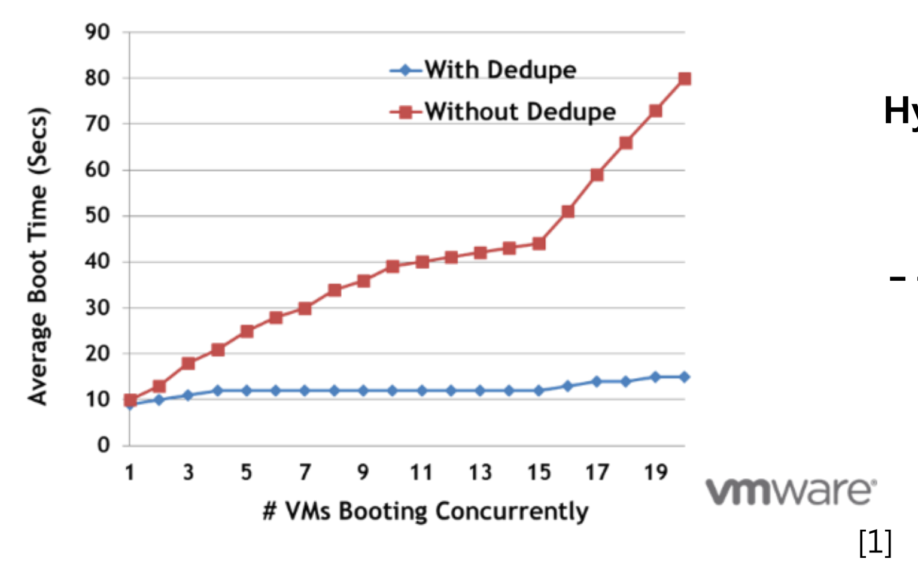
Dong-Hyun Song*, Youngjae Lee and Jin-Soo Kim

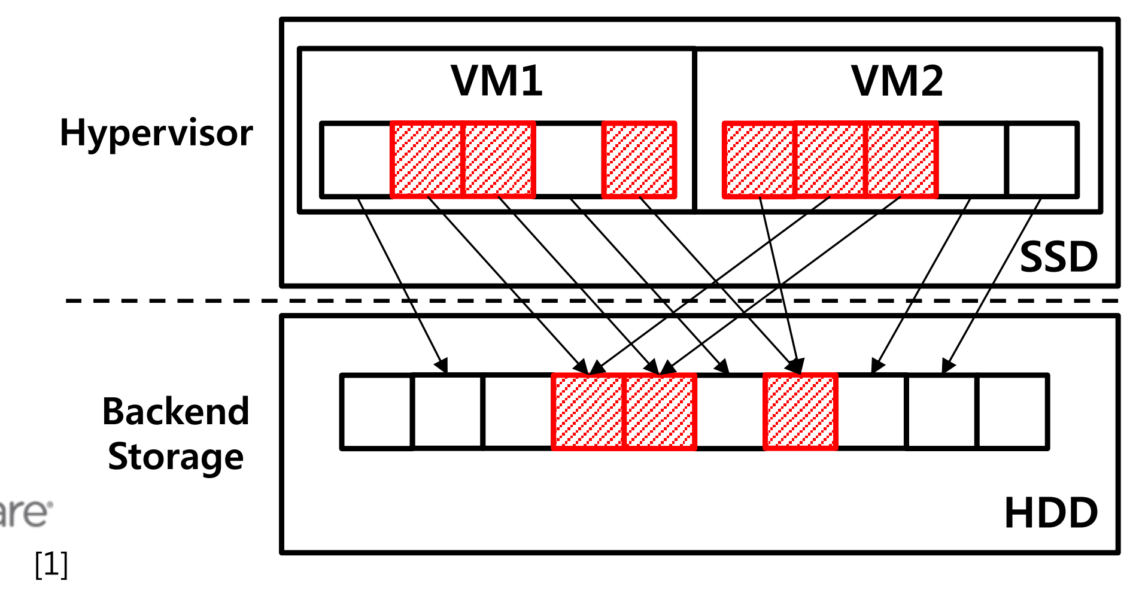
SungKyunKwan University, South Korea

Motivation

Inefficiencies when the same block is cached by multiple VMs

- **Reduce available cache space**
- **Increase cache space competition between VMs**
- **I/O performance degradation**





Challenges and Approaches

✓ How can the hypervisor fairly partition the cache space among VMs?

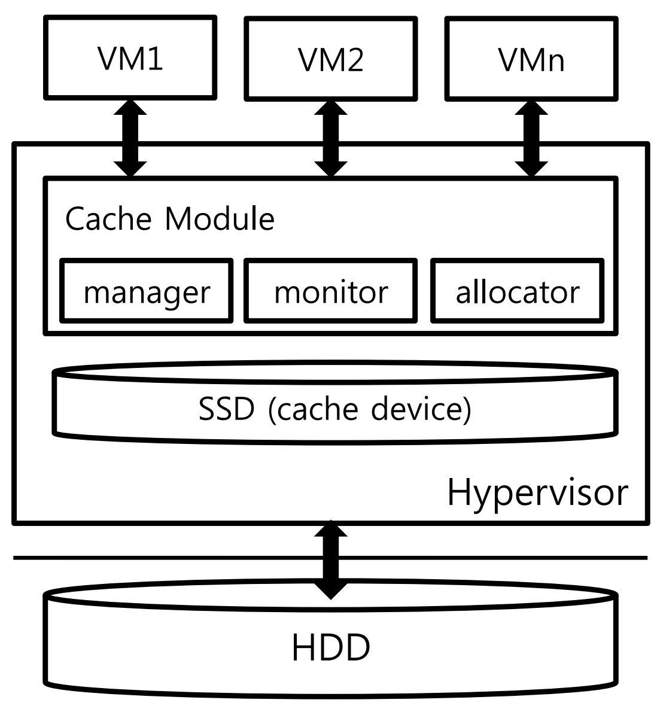
✓ How can the hypervisor efficiently manage the shared cache blocks?

Approaches

➢ **Cache management architecture**

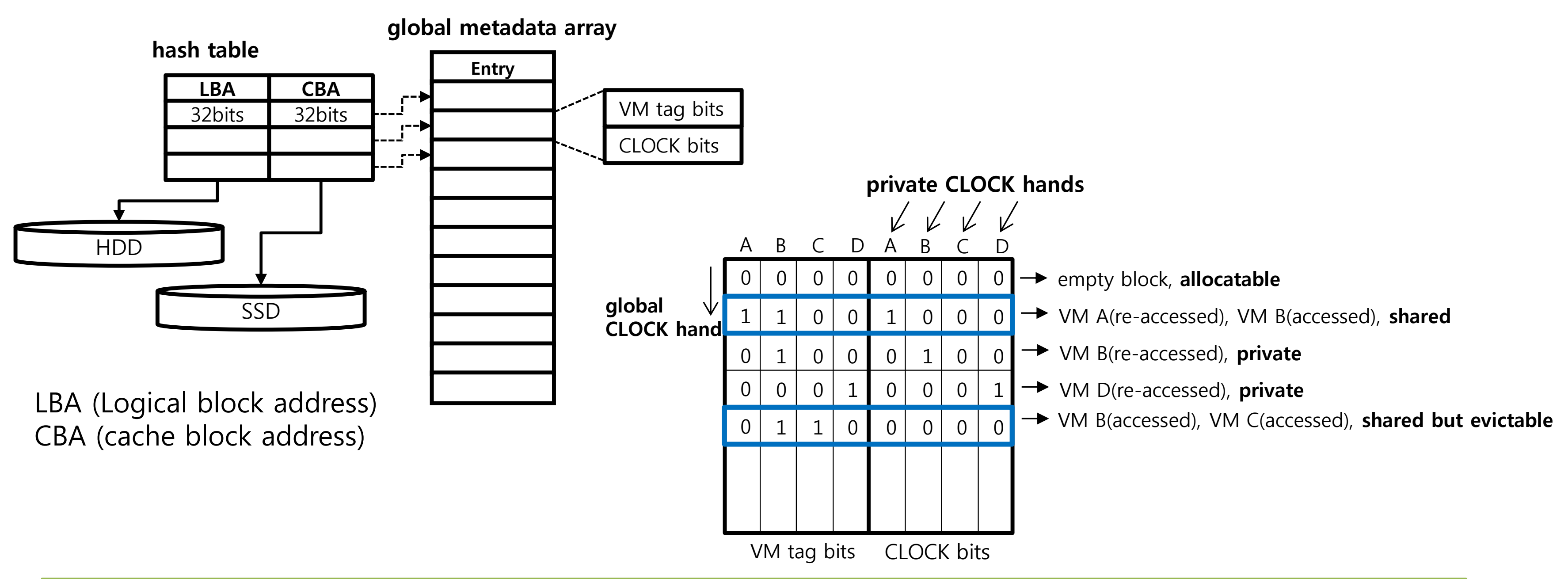
➢ **Cache partitioning metric**

➢ **Cache replacement algorithm**



Our solution

Global metadata



LBA (Logical block address)
CBA (cache block address)

Cache partitioning metric

$$rECS = \frac{m}{N} \Rightarrow rCrS = rECS \times \frac{r}{m} = \frac{m-r}{N-S}^{(1)} \times \frac{r}{m}^{(2)}$$

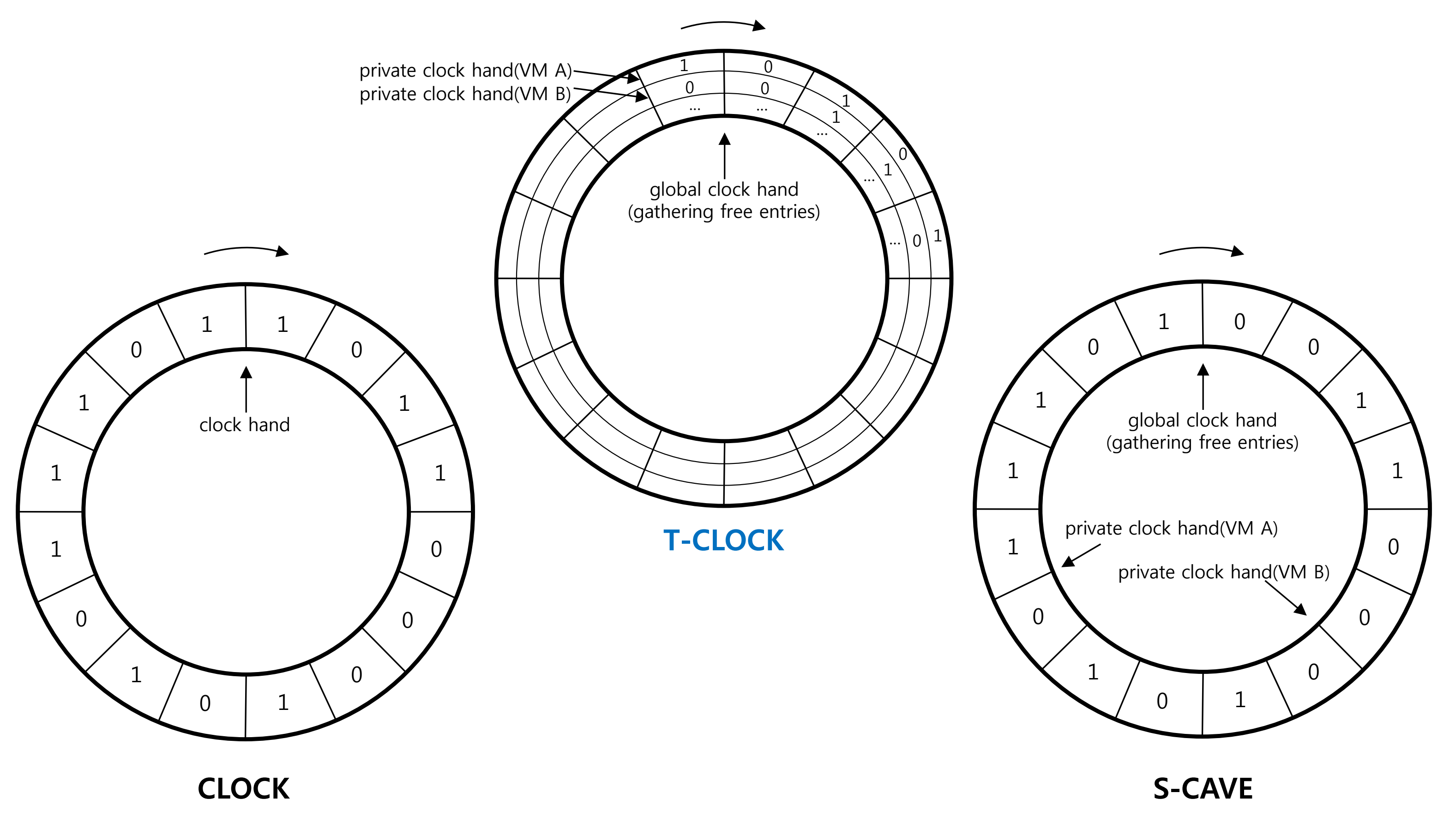
(S-CAVE[2])

- m: a number of re-access blocks of a VM
- r: a number of re-access shared blocks of a VM
- N: total blocks of a VM
- S: total shared blocks of a VM

(1) The number of re-accessed cache blocks to the total number of private cache blocks allocated to a VM.
(2) The number of shared cache blocks to the total number of re-accessed cache blocks.

✓ **The rCrS metric avoids choosing a VM with a higher proportion of shared cache blocks as a victim**

Cache replacement algorithm

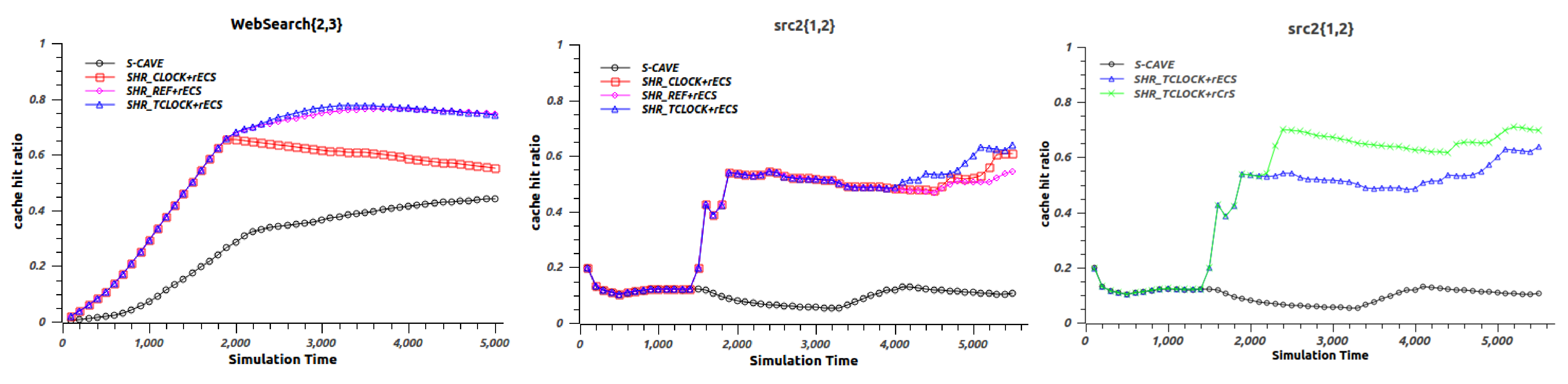


(1) If the CLOCK bit is 1, then it is reset to 0.
(2) If all CLOCK bits are 0, then the T-CLOCK algorithm reclaims the free cache block.
(3) If its own CLOCK bit is 0 but another CLOCK bit is set to 1, this shared cache block survives until all CLOCK bits becomes 0.

✓ **The private CLOCK hand tests exclusively its CLOCK bit**
✓ **The T-CLOCK algorithm tends to prolong the lifetime of shared cache blocks**
✓ **If there are no shared cache blocks, T-CLOCK algorithm is same as CLOCK**

Evaluation

Cache hit ratio: 31%-63% higher than S-CAVE

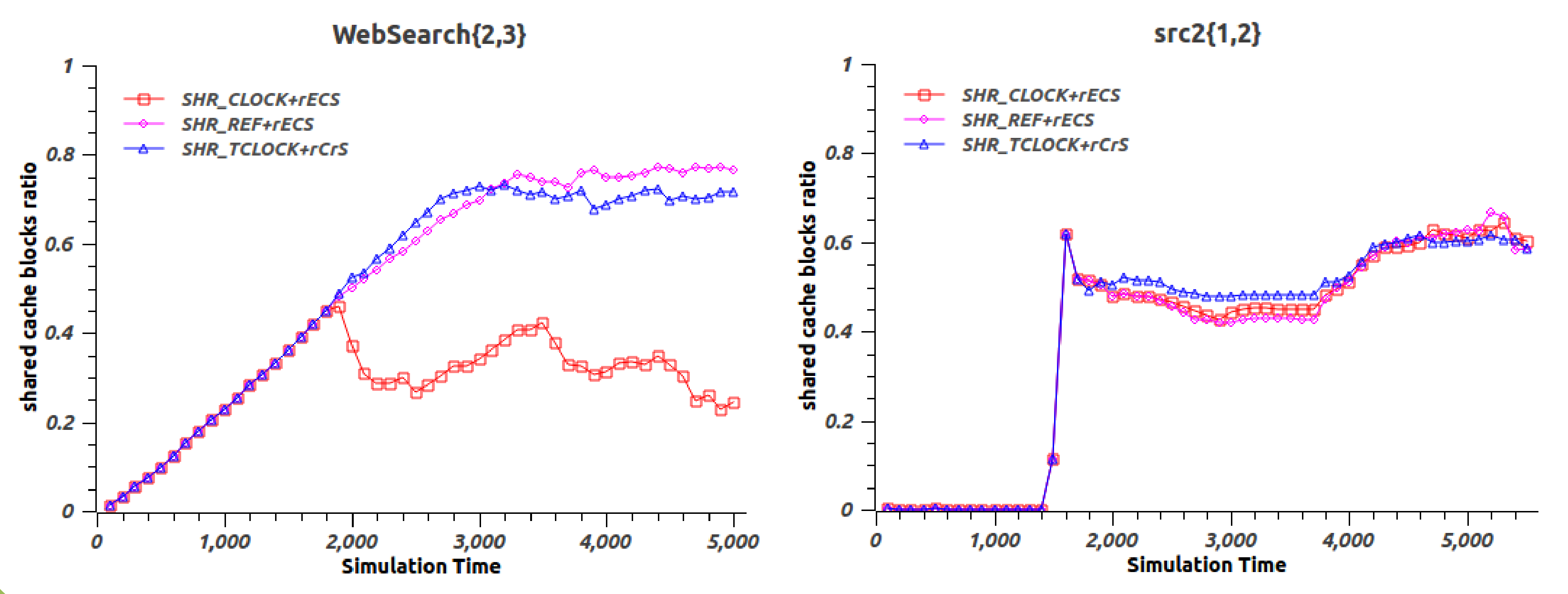


Trace-driven simulator
- Read intensive workloads
- Write-through
- Co-run: WebSearch2, WebSearch3
- Co-run: src2_1, src2_2

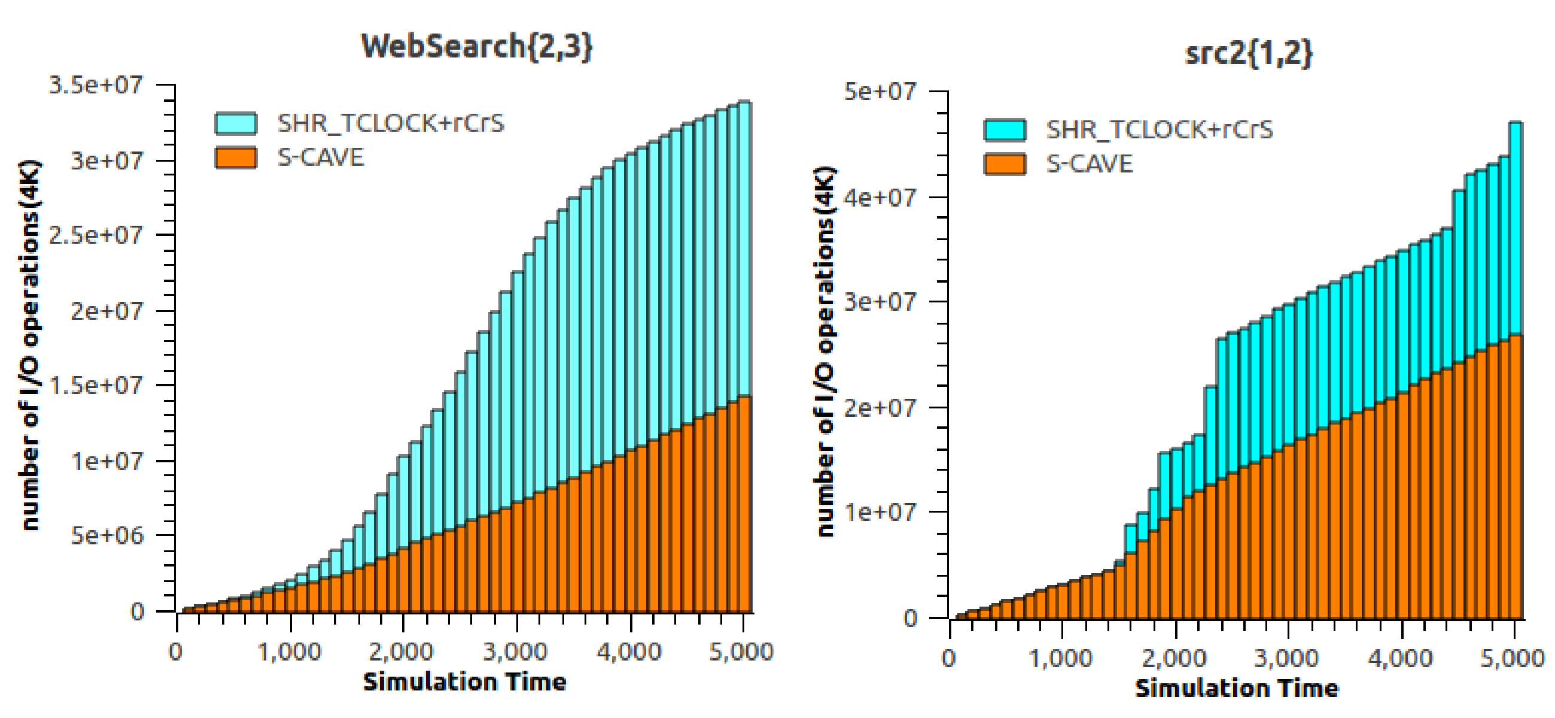
Cache partitioning metric
- rECS(S-CAVE)
- rCrS

Cache replacement algorithm
- S-CAVE(CLOCK with all blocks)
- SHR_CLOCK(CLOCK with shared blocks)
- SHR_REF(Reference counts with shared blocks)
- SHR_TCLOCK(T-CLOCK with shared cache bocks)

Shared blocks ratio: efficiently balanced with rCrS metric



I/O throughput: by 3.17x higher than S-CAVE



References

[1] http://permabit.com/turbocharging-hybrid-storage-with-data-efficiency-deduplication-and-compression-impact/
[2] Tian Luo, Siyuan Ma, Rubao Lee, Xiaodong Zhang, Deng Liu, and Li Zhou. S-CAVE: Effective SSD Caching to Improve Virtual Machine Storage Performance. In PACT'13