A Deduplication Study for Host-side Caches with Dynamic Workloads in Virtualized Data Center Environments

Jingxin Feng, Jiri Schindler
NetApp Inc.

1 Introduction

Deduplication is a well-known method that improves storage efficiency and reduces the cost of storage in corporate data centers [3, 4]. For virtualized data centers, and in particular for virtual desktop infrastructure (VDI), centrally-managed networked storage can greatly reduce the overall data footprint because virtual machine (VM) disk images have largely the same content.

Recent work by Byan et al. [1] suggested employing flash memory-based host-side caches inside VM hypervisors to shed load from the shared storage infrastructure. They demonstrated that such caches can be effective for read-mostly workloads with stable working sets. However, it is not known if they can be as effective in a dynamic environment, when virtual machines migrate frequently from one physical server (hypervisor) to another or when their working sets change. First, since these caches are large (many 100s of GBs), re-warming them with the active content to reach a steady state cache hit rate after a VM migrates to another hypervisor can take as much as 12 hours [1]. Second, as each VM disk image is a separate entity, the caches might contain many copies of the same content even though the network-attached shared storage system would store only a single instance, unnecessarily polluting the host-side hypervisor caches and reducing their overall cache hit rate.

The goal of our study is to explore the effectiveness of deduplication for large host-side caches in virtualized data center environments running *dynamic VDI workloads*. To that end, we analyze traces captured from VDI deployments as well as general enterprise workloads [2]. Understanding the intrinsic properties of data duplication can help us design effective host-side cache policies. Previous deduplication systems and studies focused on data mostly at rest such as backups [4] and archives or on reducing network traffic across a WAN link.

Our study on eight traces in duration from two minutes to almost 100 days shows that deduplication can reduce the data footprint inside host-side caches by as much as 67%. This allows for caching a larger portion of the data set and improves the effective cache hit rate. More importantly, such increased caching efficiency can alleviate load from networked storage systems during I/O intensive workloads when most VM instances perform the same operation such as virus scans, OS patch installs (a.k.a. update storms), and reboots (a.k.a. boot storms).

2 Our Approach

We seek to improve cache hit rates for large host-side caches in VDI environments. As step towards our goal, we want to understand the real world workloads and assess how much opportunity for deduplication exists. We focus on discovering duplicate data intrinsic to these traces regardless of cache size or specific cache replacement policies. Our results provide an upper bound of cache space saved by deduplication; we realize that the specifics of cache organization and replacement policies may reduce the effective cache hit rate we report.

We analyze two sets of traces: two long term CIFS traces collected in a production enterprise data center [2] and six VDI traces with 3–15 VMs. We collected the VDI traces in our lab while the VMs performed a variety of typical tasks including login, patch update, and reboot.

We define a metric we term *deduplication degree*, which we believe is useful in measuring the effectiveness of content deduplication. In the context of caching dynamic workloads, our metric allows us to express directly the number of references to a unique cached block. Thus, we define deduplication degree, d, as

$$d = \frac{\sum_{i=1}^{n} L(i)}{n} \qquad (1)$$

where n is the number of unique cached blocks and $L(i)$ is the number of references to the ith block(i.e., the number of addresses that have the same contents).

By using deduplication degree, we can directly evaluate cache metadata overheads involved in tracking the various contexts such as files and offsets for the single instance of the shared cached block. Recall that block based caches use a buffer header for each cached block. The buffer header in a deduplicating cache would also need to keep additional information such as the block fingerprint and the backpointers to the context in which they exist. Deduplication degree can express the average

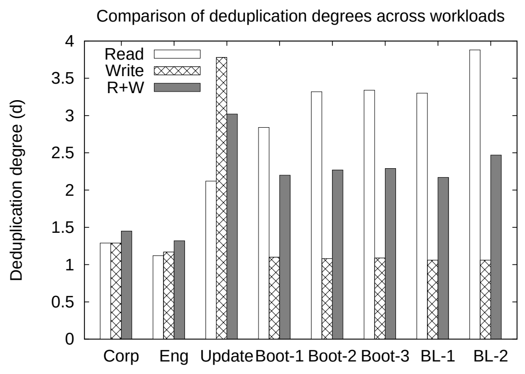

Figure 1: Comparison of deduplication degree for different workloads. Note the different profiles for read-heavy and write-heavy workloads.

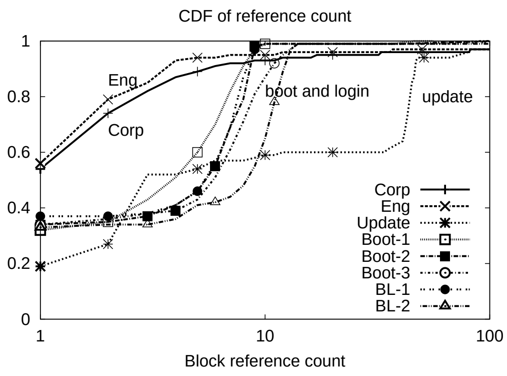

Figure 2: CDF of block reference count for all traces with three distinct profiles.

number of references to each data block and thus estimates additional metadata overheads.

3 Preliminary Results

Figure 1 shows the deduplication degree for the 8 traces. The values range from 1.2 to 3 for combined reads and writes. For the VDI traces, this translates to saving between 54% and 67% of space. Larger deduplication degrees can be observed when we consider reads and writes separately. As expected, the boot and login storms exhibit different behavior from update storms. For write-heady update storms, we observe more duplication in write requests compared to the five boot traces, which show little or no duplication in write requests. The deduplication degree for writes in these read-heavy traces is close to one, meaning that, on average, each unique block is seen only once by the hypervisor.

The write-heavy update storms show high deduplication degree for reads as well. Yet, we see little or no increase for the boot storm traces as the number of virtual machines per hypervisor increases from 10 to 15 (Boot-1 and Boot-2 vs. Boot-3). For the boot+login storm traces, deduplication degree increases slightly for the reads as more VMs are added (BL-1 vs. BL-2).

Figure 2 shows the reference count CDFs for our traces. For most workloads, more than 90% of the data blocks are referenced no more than 15 times. The CDFs have long tails, which means that a few blocks have large number of duplicates. That is, a single instance of the block in a cache would have a large number of references. For example, in CIFS Eng trace, 95% of blocks are referenced less than 10 time. At 100%, the reference count jumps to almost two million. In a deduplicated cache, all the addresses that have the same content need to be stored in memory as references. If one pointer takes 4 or 8 bytes, as shown in Figure 2, 95% of data blocks use less than 60 or 120 bytes to store references respectively, while some may take 7 or 14M.

4 Conclusion

We believe that deduplication degree captures a useful concept for evaluating cache effectiveness for dynamic workloads; It is directly correlated with the reference count (pointers) needed to link the single instance copy of the data shared among multiple contexts (e.g., VM disk images). Our data suggests a design point where reference counts of less than 100 capture over 99% of all duplicates for most workloads. Besides duplication degrees in different workloads, we are also analyzing similarity of VDI traffice, deduplication sensitivity to cache block size and other aspects that may provide us design insights on host-side cache in VDI environment.

References

[1] S. Byan, J. Lentini, A. Madan, L. Pabon, M. Condict, J. Kimmel, S. Kleiman, C. Small, and M. Storer. Mercury: Host-side flash caching for the data center. In *Proceedings of the 28th Symposium on Mass Storage Systems and Technologies (MSST)*, 2012.

[2] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller. Measurement and analysis of large-scale network file system workloads. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, 2008.

[3] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. iDedup: latency-aware, inline data deduplication for primary storage. In *Proceedings of the 10th USENIX conference on File and Storage Technologies*, 2012.

[4] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, 2008.