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

Jingxin Feng, Jiri Schindler
NetApp Inc.
{*Jingxin.Feng, Jiri.Schindler*}*@netapp.com*

***Abstract*—Flash memory-based caches inside VM hypervisors can reduce I/O latencies and offload much of the I/O traffic from network-attached storage systems deployed in virtualized data centers. This paper explores the effectiveness of content deduplication in these large (typically 100s of GB) host-side caches. Previous deduplication studies focused on data mostly at rest in backup and archive applications. This study focuses on cached data and dynamic workloads within the shared VM infrastructure. We analyze I/O traces from six virtual desktop infrastructure (VDI) I/O storms and two long-term CIFS studies and show that deduplication can reduce the data footprint inside host-side caches by as much as 67%. This in turn 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 storms when most VM instances perform the same operation such as virus scans, OS patch installs, and reboots.**

I. Introduction

Deduplication is a well-known technique to improve storage efficiency and reduce the cost of on-line storage in corporate data centers [17, 21, 25]. For virtualized environments, and in particular for virtual desktop infrastructure (VDI), centrally managed networked storage can greatly reduce the overall data footprint because the virtual machine (VM) disk images have largely the same content. However, the network storage systems may also experience bursts of extremely high load when hundreds of VM instances perform essentially the same operations such as virus scans, OS patch installs (a.k.a. update storms), and reboots (a.k.a. boot storms).

To shed load from the shared infrastructure, recent work by Byan et al. [2] suggested employing flash memory-based host-side block caches and showed they could be effective for read-mostly workloads with stable working sets. However, it is not clear how well they would work in a dynamic environment where virtual machines migrate frequently from one physical server (hypervisor) to another or where VM working sets change over time. First, since these caches are large, warming them after a VM boot or migration can take as much as 12 hours [2]. 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 cache and reducing the overall cache hit rate.

Previous deduplication studies focused on primary [3, 21], backup [24, 25] and archival [16, 17] storage or on reducing the network traffic [20]. Our work explores the effectiveness of cache content deduplication for large host-side caches in a virtualized data center environments where virtual machines can migrate among a set of hypervisors and working sets change frequently. Previous work [2] suggested this approach without addressing how such caches would be structured. Our goal is to provide hints for designing more effective host-side caches. Thus, we consider different deduplication techniques, cache organizations (i.e., fix-sized blocks vs. variable-length extents) and analyze traces captured from VDI environments and corporate environment with hundreds of clients (see Section II).

Our trace analysis aims to highlight the opportunities for cache content deduplication intrinsic to the data and specific workloads. We understand that the cache size and replacement policies affect the cache hit rate. Thus our study provides an upper bound on the effective cache hit rate based on the cache organization alone. We show that deduplication at host-side flash caches can save 54% to 67% of cache space for VDI workloads and 24% to 31% for long-term CIFS workloads. This can increase the effective cache hit rate, reduce network traffic, and offload I/Os from the shared storage system.

II. Host-side cache and deduplication

We review the trends in virtual server environments, including the emergence of host-side flash caches to motivate our cache content deduplication study that focuses on transient caches with highly dynamic workloads.

A. Data sharing through consolidation

In a virtualized infrastructure, each VM disk image is a separate logical entity, yet with mostly identical content: an operating system image, runtime libraries, and the server application code. The only data that is typically unique to each VM instance is a handful of configuration files (e.g., in UNIX typically residing in the `/etc` directory). Thus, with proper support from the back-end storage system [8, 21, 24], each VM disk image file can be cloned from a master image and include just the unique blocks with references to the common ones. Even if the storage system does not collaborate explicitly with the virtualization infrastructure (e.g., the individual hypervisors), it can still eliminate duplicate content (blocks) in the background and reap the benefits of (opaque) duplicate content elimination.

Deduplication exists in shared network-attached storage systems in the form of block deduplication across different VM OS images, and VM hypervisors in the form of sharing common memory pages of different guest OSes [5, 13, 23]. However, to the best of our knowledge there are no proposed or deployed systems where it occurs in the OS or file system buffer caches. Hence, without explicit provisions, a hypervisor-based buffer cache may store multiple copies of the same file simply because it belongs to different VM disk images. Polluting the shared buffer cache with the same data reduces its effective hit rate and in turn leads to increased I/O load on the networked storage system.

B. *New memory technologies*

Flash memory provides a new tier in the memory hierarchy between DRAM and hard disk. A flash memory-based SSD can be deployed effectively as a persistent host-side cache [2] to reduce data traffic and I/O between the host and network-attached disks. Host-side caches provide advantages for virtualized environments where VMs use similar data, run similar software stacks, or where VM migration from one physical server to another is common. A cache integrated with a hypervisor can obviate the need to read data from the shared storage infrastructure upon VM move.

As large host-side caches become more prevalent, we must understand how effective they will be in environments with dynamic workloads. For example, it takes more than 10 hours to warm up a 320 GB host-side cache and reach a steady-state hit rate after a live VM migration [2]. Recognizing shared pages that are already cached can shorten this period; content deduplication ensures storing them only once, leaving room for other data and thus increasing the effective cache hit rate.

C. *Deduplication approaches and cache organization*

There are three main hash-based methods to discover identical content within a dataset: (i) whole file content hashes, (ii) fixed size block hashes, and (iii) content-defined variable-length chunk hashes, for which Rabin fingerprinting [18] is an effective method used both in primary and secondary storage [12, 14, 16, 24].

Which method can be deployed in virtualized environments is governed by the shared host-side cache organization. File-based caching is not very practical – it would require the hypervisor-based cache to understand the format of the VM disk image to identify individual blocks as belonging to the same file. In contrast, fixed block-based cache designs are most prevalent and can transparently allow for duplicate content elimination of the same files in different VM disk images. Finally, variable extent-based cache organization meshes well with Rabin fingerprinting — it would define boundaries for the cached extents. While this approach might yield potentially better results, the cache organization for variable content-defined extents would be much more complex compared to fixed-size buffers. Therefore, we focus in our study on fixed-length chunks. However, we also do compare fixed-length and variable-length chunking in Section IV. This comparison allows designers to evaluate the trade-off between additional design complexity and a potentially higher deduplication ratio.

III. Improving cache effectiveness

We use a metric we term *deduplication degree*, which allows us to directly measure the effectiveness of content deduplication for a class of workloads in virtualized data centers and express the cache metadata overheads in managing buffers shared by multiple files.

A. *Deduplication degree metric definition*

We seek to answer three basic questions with our study: 1) how much space can be reclaimed by eliminating duplicate content, 2) how will the cache hit rate be affected, and 3) how to structure the cache metadata so that different contexts (e.g., buffer headers) can point to the same buffer. We also want to explore the trade-off whereby a smaller block size can lead to more duplicate content elimination, but increases metadata overheads.

Our metric is similar to those used in previous studies. For example, iDedup [21] uses deduplication ratio, expressed as percentage of the original data size, to denote the amount of data redundancy. Wallace [24] uses deduplication rate, such as 1x and 10x. Clements [3] uses storage reduction as a percentage of storage space. These metrics focus on space saving for primary or secondary storage and are similar to each other. In the context of caching dynamic workloads, we want to express directly the number of references to a unique 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 i-th block (i.e., the number of addresses that have the same contents). Recall that for each cached block, say 4 KB, the cache uses additional memory to store the buffer header information. With deduplication, it would additionally need to store the buffer content fingerprint, as well as references to all the files that share the same buffer.

Deduplication degree allows us to find the "right" number of references to each data block, and thus illuminate the trade-offs between content deduplication and metadata overheads. We thus express the distribution of the number of references for each unique block using the Cumulative Distribution Function (See Section IV-E). Based on a previous study of clustered SAN file systems [3], we expect the majority of blocks to be referenced fewer than 10 times with a few very popular blocks having 100s of references.

B. *Effective cache hit rate*

Ideally, we would like to express directly how employing deduplication in a shared cache improves the effective cache hit rate. Unfortunately, the hit rate depends not only on the cache size but also on the policies it employs. Given our traces, we can express only the effective cache size i.e., the increased cache

TABLE I. GENERAL TRACE STATISTICS.

	CIFS [10]		Virtualized Desktop Infrastructure (VDI)					
	Corp	Eng	Upd	B-1	B-2	B-3	BL-1	BL-2
Clients/VMs	5261	2654	3	10	10	15	10	15
Duration	65 days	97 days	8 mins	2 mins	2 mins	2 mins	2 mins	2 mins
Data read	364.3 GB	723.4 GB	1.34 GB	92.0 MB	117.9 MB	131.8 MB	138.0 MB	216.9 MB
Data written	177.7 GB	364.4 GB	1.99 GB	22.5 MB	35.0 MB	41.8 MB	46.3 MB	61.6 MB
R:W bytes ratio	2.1	2.0	0.7	4.1	3.4	3.2	3.0	3.5
R:W I/O ratio	3.2	2.3	1.3	7.5	5.8	5.4	5.0	5.7

size perceived due to the elimination of the duplicate content. System designers can then take our results and use cache simulators configured with their cache sizes and policies to determine the *actual* cache hit rate. For fixed-length chunking systems, effective cache size is computed as $s = d * s'$, where s' is the real space size for cached data, d is the deduplication degree, and s is the effective cache size, which is the space needed without deduplication while the same hit ratio is guaranteed.

For variable-length chunking, deduplication degree cannot be used to derive effective cache size directly because the size of every chunk is different. To get an accurate number, we need to sum the space saved by each chunk. We can approximate this number by using the average chunk size.

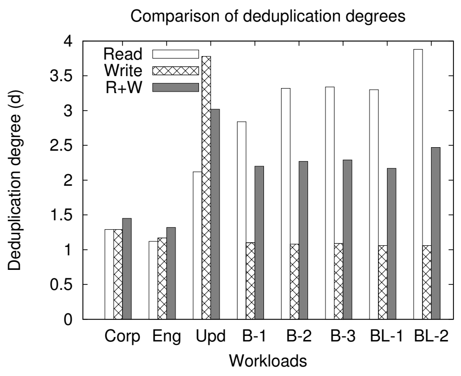


Fig. 1. Comparison of deduplication degree for different workloads. Note the different profiles for read-heavy and write-heavy workloads.

IV. EVALUATION

A. *Traces*

We analyzed two sets of traces: two long-term CIFS traces collected in a production enterprise data center [10] and six VDI traces with 3–15 VMs. The CIFS traces capture activities within the corporate administrative and engineering departments of a single company and represent systems accessing over 22 TB of storage accessed by workstations of 1500+ employees. We collected the VDI traces in a lab environment while the VMs performed a variety of typical tasks including users logging in, propagating patch updates, and rebooting. For the Upd trace, we applied the KB2604521 patch set to Windows Server 2008 R2. The B-1, B-2 and B-3 are three different boot traces with 10 and 15 VMs booting simultaneously, and BL-1 and BL-2 are traces from VM booting followed immediately by a user login to VMs running MS Windows 7 guest OS.

Collectively, the traces are a cross-section of activities in a virtualized data center: VDI environments with many similar instances and disk images may generate a lot of similar read and write requests while the CIFS traces represent traffic to network file systems deployed inside enterprises. Table I summarizes these traces and lists their names we use throughout this section.

B. *Deduplication degree*

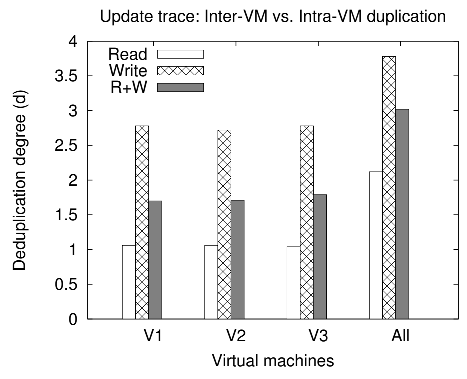
Figure 1 shows the deduplication degree for the eight different traces. The values range from 1.2 to 3 for combined reads and writes. For the VDI traces, this translates to saving between 54% and 67% of space. Larger values for deduplication degree occur when we consider reads and writes separately.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

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 (B-1 and B-2 vs. B-3). For the boot+login storm traces, deduplication degree increases slightly for the reads as more VMs are added (BL-1 vs. BL-2).

We are also interested in determining whether intra-VM redundancy is more prevalent than redundancy across VM disk images. Generally, if most of the deduplication occurs within a single VM image, the trend of increasing consolidation (i.e., more VMs running on a single hypervisor) will not yield more opportunities for deduplication within the shared host-side cache.

Figure 2 compares deduplication degree within each VM image and across all VMs combined for the Upd and B-2 traces. Due to space constraints, we show only one boot trace; the other boot and login traces exhibit similar trends regardless of number of VMs. The right-most group of columns shows deduplication degree for all the VMs combined, the other groups show the deduplication degree for each individual VM image (intra-VM deduplication).

We notice that for the Upd trace, the intra-VM deduplication degree is only slightly larger than 1 for reads while it is about 2.7 for writes. More importantly, the inter-VM deduplication degree is larger for both reads and writes. Similar trends hold for the B-2 trace. There are almost no opportunities for deduplication in



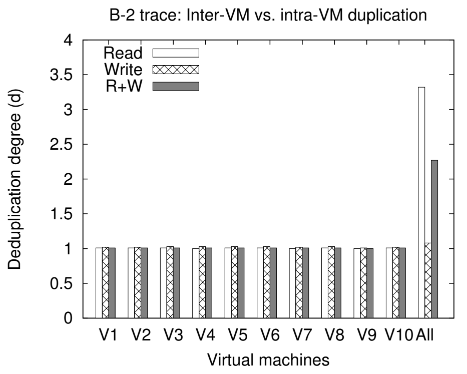


Fig. 2. Intra-VM vs. inter-VM deduplication for the Update and B-2 traces. Other boot traces show similar profile. The Vx bars shows the intra-VM deduplication for each VM, while the All bar shows the inter-VM deduplication when all VMs share the same cache.

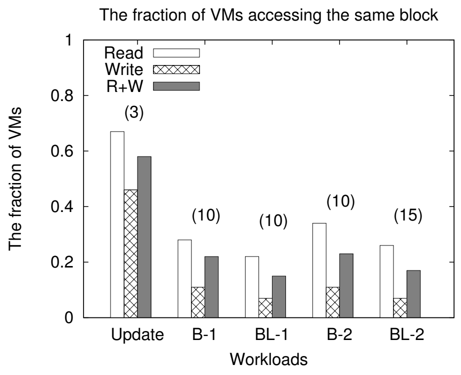


Fig. 3. The (number) shows the total number of VMs. A small fraction of VMs access the same block because of VM-unique content.

a single VM, while the inter-VM deduplication degrees are much higher. There is a difference between the intra-VM data of the two traces because update storms involve more data transfer than boot and login storms. The inter-VM results suggest that more VM consolidation would yield better content deduplication and likely increase the effective cache hit rate.

C. Similarity of VDI traffic

The number of VMs that access one block is less than or equal to the reference count of that block. The average number of VMs that touch each block suggests whether all VMs read or write the same data. As shown

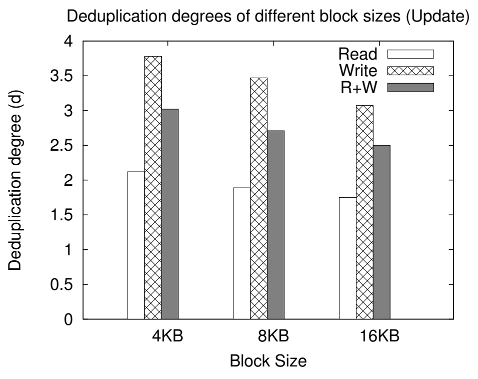


Fig. 4. Deduplication degree as a function of block size for the Upd trace. Other traces show similar trends.

in Figure 3, with the exception of the Upd trace, the average number of VMs that access each block is much lower than the number of total VMs running on the same hypervisor, because the fractions are all less than 0.4. The reason is that different VMs may access the same files, but these files have VM-unique content.

When we inspected the block content, we found a large amount of data that is only slightly different on each VM. Thus, variable-length chunking can increase deduplication degree as we show below. Delta differencing could also be an effective technique, but it does not lend itself to an easy integration to existing cache designs. Thus, we do not consider it here.

D. Sensitivity to cache block size

For a given workload, system designers must choose an appropriate block size that improves the effective cache hit rate without excessive metadata overheads. Figure 4 describes the sensitivity of deduplication degree to block size. In VDI update storms, as block size increases up to 16 KB, deduplication degree drops; this drop is not significant for update storms. Thus, systems with well-provisioned networks may opt to use larger blocks to improve cache metadata efficiency.

Although content-based variable-length chunking complicates design of storage caches, we examine how much additional opportunity for content deduplication it could provide compared to fixed-length chunking. We use Rabin fingerprinting with the minimum, mean, and maximum chunk sizes set to 2 KB, 4 KB, and 16 KB respectively, and a sliding window of 48 bytes, which has been shown to provide good results [14].

Table II shows how much additional deduplication is possible with variable-length chunking relative to the best-case fixed-length 4 KB blocks. We use space savings because the deduplication degree metric does not apply directly to saved space when variable length is used. Variable-length chunking can provide additional savings ranging from 10% to 15% for most traces with the exception of the Upd trace for which it provides no benefit. If we consider read and write requests separately, differences of write in B-3 and BL-1 could as much as 40%. However, the volume of data written in these traces is very small: only 10s of MB.

TABLE II. ADDITIONAL SPACE SAVINGS ACHIEVED BY VARIABLE CHUNKS RELATIVE TO FIXED 4 KB CHUNKS. THE FIRST LINE SHOWS THE SPACE SAVINGS BY USING FIXED 4KB CHUNKS, THE LAST LINE SHOWS THE ADDITIONAL SPACE SAVINGS.

	Upd	B-1	B-2	B-3	BL-1	BL-2
Fixed	67%	54%	56%	56%	54%	60%
Variable	67%	66%	68%	71%	69%	70%
Incr.	0%	12%	12%	15%	15%	10%

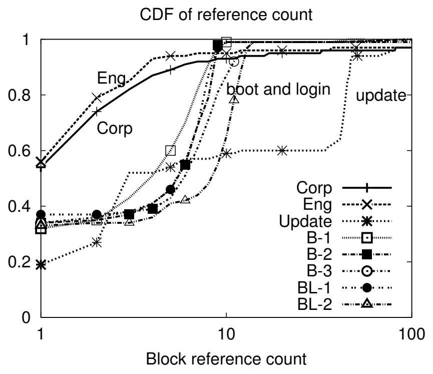


Fig. 5. CDF of block reference counts for all studied traces.

In contrast, Jin et al. [8] showed that, for stored VM disk images, fixed-length chunking leads to better deduplication compared to content-defined variable-length chunking. In our VDI workload traces, we observed that the majority of blocks in boot and login storms are less than or equal to 4KB. Given that the mean block size of variable-length chunking is less than 4 KB, this may lead to more space savings than fixed-length chunking.

E. Reference count distribution

Figure 5 shows the reference count CDFs for our traces. As shown in Table III, 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 fewer than 10 times, yet a few blocks are referenced more than a million times. Thus a cache design could target less than 10 references and provide special handling for a few blocks (e.g., an exceptions list) that are highly referenced to get additional space savings.

Recent studies [6, 11] reported that 1% of chunks accounted for almost 30% of total space savings. We have not observed such a high rate in our traces. The CIFS traces show that around 22% of the total space savings come from 1% unique blocks and VDI traces show even less. The reason is that related works studied primary/secondary storage data, which demonstrates higher deduplication degrees than our I/O data set. In addition, our VDI traces are too short for the top 1% blocks to become outstanding.

To get a better understanding of which blocks contribute to the distribution, we examined the content of these blocks from the traces available to us. Blocks

TABLE III. REFERENCE COUNT DISTRIBUTION QUANTILES.

	25%	50%	75%	90%	95%	99%	100%
Upd	2	3	44	47	80	17433	
B-1	1	3	7	9	9	12	56
B-2	1	6	8	9	9	20	119
B-3	1	6	9	11	12	29	128
BL-1	1	9	11	13	13	22	140
BL-2	1	9	11	13	13	22	180
Corp	1	1	3	6	23	2936	382102
Eng	1	1	2	4	9	1952618	

with high reference counts were blocks containing all 0s or 1s. Additionally, other blocks in this group had regular bit patterns repeated across the entire block and containing mostly 0s. We speculate that these blocks hold file system metadata such as bitmaps, but, given our traces, we could not map back the block content to the file they belonged to.

V. DESIGN HINTS

We conclude our study by providing some hints for designing an optimized host-side cache with deduplication. Our results show that there is a substantial degree of duplication in the studied dynamic workloads that can be translated into increased effective cache hit rate. This lowers system cost by reducing the real space cache size for a targeted effective cache size or providing better performance at a fixed cost. This optimization in turn provides various benefits such as reducing network traffic between hypervisors and shared storage systems, lowering load on storage systems during VDI storms, or decreasing cache warm-up time during VM migration.

Fixed-length chunking is prevalent in cache design and our results show that using 4 KB blocks can yield substantial savings. We can improve the cache hit rate by using variable-sized chunks; however the modest gains do not warrant the added design and implementation complexity of variable-extent-based caching.

Another potential issue with deduplicating fixed-length blocks is that of misaligned I/Os or partial block sharing. By choosing a suitable cache block size, system designers can choose deduplication level of the system for a given workload and cache replacement policy. With our study, they can weigh the trade-off between increased effective cache hit rate and space overheads for metadata, especially when the metadata such as buffer headers are kept in DRAM. Larger block sizes lower the demands on DRAM that is taken away from applications running inside VMs. A potential optimization to increase cache hit rate is to provide support for small (partial-block) reads for caches with deduplication that have a large block size.

Deduplication degree 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.

VI. RELATED WORK

Data deduplication is an active research area. There are many previous works that have studied the effects of duplicate content elimination and proposed different approaches and systems to improve data storage efficiency in primary [21] and secondary storage [17, 25]. LBFS [14] was designed to deduplicate network traffic in a low-bandwidth network file system, inspired by the original work of Spring and Wetherall [20].

CZIP uses hashes of variable chunks combined with chunk compression to reduce storage footprint in content-delivery networks (CDNs) [15]. We use a similar approach for studying the opportunities for eliminating duplicate content of actively used data that is cached. Kochut and Karve [9] studied provisioning space for VM images for VDIs with locally attached storage. Our work looks at data actively accessed for the same VDI workloads; however, we assume VM disk images provisioned on a shared storage system and study the opportunities in eliminating redundant data that is cached, i.e., data in active use by these VMs.

Our work shares the same goals of previous research on techniques for hypervisors to eliminate guest OS memory pages with the same content [13, 23]. We do not claim our work invents a new technique. Its contribution is in conducting a workload-specific study for large (storage-based) caches and providing design hints for these caches that are not integrated with the hypervisor virtual memory subsystem using page tables. In addition, these previous works typically proposed background scanners [1, 4, 5] for eliminating duplicate pages. In contrast, we envision flash-memory-based caches to use in-band techniques as suggested by a previous work on the Mercury system [2].

There are several deduplication studies of real-world data and traces [12, 16]. Jayram et al. [7] studied the similarity of data at rest within VM images. Our work studies the implications when the images are cached and in active use. Clemens [3] suggested a block-level deduplication system for a clustered file system used in virtualized environments. Some research has been done to generate datasets to simulate real workloads [22]. Such trace or workload generators could be employed in our context.

Some previous studies of large-scale I/O traces provide comprehensive analysis on access, usage, and sharing patterns [10, 19]. However, these studies do not try to detect duplication related characteristics of these workloads. Our work expands on these and, in the case of the Leung et al. study, analyzes the same data.

Acknowledgements

We would like to thank Christopher Small who contributed to our initial investigation and helped us with the original concepts of this work. Our gratitude goes to John Strunk, James Lentini, Luis Pabon, and the anonymous reviewers for helping us refine this paper. We thank Scott Dawkins for his insights and support.

References

[1] A. Arcangeli, I. Eidus, and C. Wright. Increasing memory density by using KSM. In *OLS '09: Proceedings of the Linux Symposium*, 2009.

[2] 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 *MSST*, 2012.

[3] A. T. Clements, I. Ahmad, M. Vilayannur, and J. Li. Decentralized deduplication in SAN cluster file systems. In *USENIX Annual Technical Conference*, 2009.

[4] K. M. et al. Ksm++: Using i/o-based hints to make memory-deduplication scanners more efficient. In *Proceedings of the ASPLOS Workshop on Runtime Environments, Systems, Layering and Virtualized Environments (RESoLVE'12)*, 2012.

[5] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat. Difference engine: harnessing memory redundancy in virtual machines. *Commun. ACM*, 53(10):85–93, 2010.

[6] B. Hong and D. D. E. Long. Duplicate data elimination in a san file system. In *MSST*, 2004.

[7] K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei. An empirical analysis of similarity in virtual machine images. In *Middleware 2011 Industry Track Workshop*, 2011.

[8] K. Jin and E. L. Miller. The effectiveness of deduplication on virtual machine disk images. In *SYSTOR*, 2009.

[9] A. Kochut and A. Karve. Evaluation of redundancy driven provisioning for hypervisors with locally attached storage. In *MASCOTS*, 2011.

[10] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller. Measurement and analysis of large-scale network file system workloads. In *USENIX ATC*, 2008.

[11] D. Meister and A. Brinkmann. Multi-level comparison of data deduplication in a backup scenario. In *SYSTOR*, 2009.

[12] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. In *FAST*, 2011.

[13] G. Miłós, D. G. Murray, S. Hand, and M. A. Fetterman. Satori: enlightened page sharing. In *USENIX Annual Technical Conference*, Berkeley, CA, USA, 2009. USENIX Association.

[14] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *SOSP*, 2001.

[15] K. Park, S. Ihm, M. Bowman, and V. S. Pai. Supporting practical content-addressable caching with CZIP compression. In *USENIX Annual Technical Conference*, Berkeley, CA, USA, 2007. USENIX Association.

[16] C. Policroniades and I. Pratt. Alternatives for detecting redundancy in storage systems data. In *USENIX ATC*, 2004.

[17] S. Quinlan and S. Dorward. Venti: A new approach to archival storage. In *FAST*, 2002.

[18] M. O. Rabin. Fingerprinting by random polynomials. Technical report, Center for Research in Computing Technology, Harvard University, 1981.

[19] D. Roselli, J. R. Lorch, and T. E. Anderson. A comparison of file system workloads. In *USENIX ATC*, 2000.

[20] N. T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM, 2000.

[21] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. iDedup: latency-aware, inline data deduplication for primary storage. In *FAST*, 2012.

[22] V. Tarasov, A. Mudrankit, W. Buik, P. Shilane, G. Kuenning, and E. Zadok. Generating realistic datasets for deduplication analysis. In *USENIX ATC*, 2012.

[23] C. A. Waldspurger. Memory resource management in VMware ESX server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002.

[24] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu. Characteristics of backup workloads in production systems. In *FAST*, 2012.

[25] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *FAST*, 2008.