Efficiently Storing Virtual Machine Backups

Stephen Smaldone, Grant Wallace, and Windsor Hsu
Backup Recovery Systems Division
EMC Corporation

Abstract

Physical level backups offer increased performance in terms of throughput and scalability as compared to logical backup models, while still maintaining logical consistency [2]. As the trend toward virtualization grows, virtual machine backups (a form of physical backup) are even more important, while becoming easier to perform. The downside is that physical backup generally requires more storage, because of file system meta-data and unallocated blocks. Deduplication is becoming widely accepted and many believe that it will favor logical backup, but this has not been well studied and the relative cost of physical vs. logical on deduplicating storage is not known. In this paper, we take a data-driven approach using user data to quantify the storage costs and contributing factors of physical backups over numerous generations. Based on our analysis, we show how physical backups can be as storage efficient as logical backups, while also giving good backup performance.

1 Introduction

Today, physical backups are becoming even more important and easier to perform as virtualization technologies are more broadly adopted. The main advantage of physical (block-based backups) is that a file system directory traversal is not required; one can just copy the virtual disk files (e.g., via the VMware backup API). This almost completely removes the need to perform any random seeks, dramatically increasing the speed in which backups can be completed [2]. However, even with the emergence of efficient deduplicating storage systems such as the Data Domain backup appliance [10], there is a larger storage cost associated with physical backups as compared to logical backups of the same data. Aside from the larger amounts of meta-data and unallocated blocks that must be stored, file system churn can also introduce additional storage overhead. How these effect deduplication is not well understood.

In this paper, we analyze user data to quantify the storage costs and contributing factors across generations of physical backup. The data we utilize is a set of backups collected from user workstations. We use this data to evaluate the backup storage efficiency through simulation, comparing physical and logical backups. We identify key factors that can make physical backup storage efficiency comparable to logical, in the context of virtual machine (VM) data protection. We show that in a steady state of multiple backup generations, deduplication and compression can push the storage efficiency of physical backup beyond that of logical.

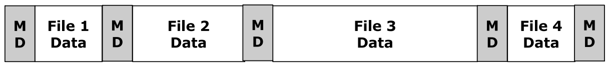


Figure 1: Logical Backup Layout

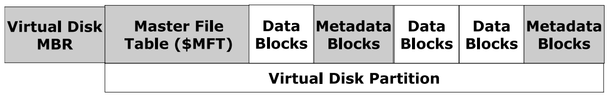


Figure 2: Physical Backup Layout

2 Background and Methodology

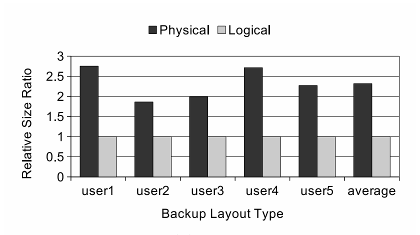
2.1 Physical vs. Logical Backup

There are two general approaches for backing up a virtual machine. The logical backup approach relies on the existence of client backup software executing within each VM. This approach copies the individual files stored within the VM's file systems and sends them in a *logical backup layout* to an external backup storage system. In this (Figure 1) individual files are concatenated with each other in a deterministic order, determined by directory tree traversal. Relevant meta-data (MD in the figure) is inserted between each file. As time passes, new files are added, others are deleted, and some are modified. However, large portions of each subsequent backup generation remain similar and this locality property has been leveraged by deduplicating storage systems [10].

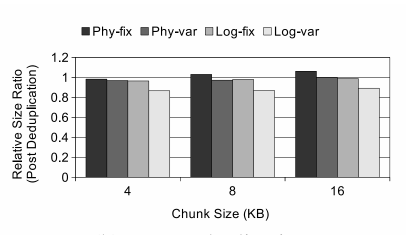
The physical backup approach relies on backup software at the virtual machine monitor (VMM) level. Under this model, copies of the VM disks (*virtual disks*) are sent to an external backup storage system. Figure 2 shows a simplified view of a virtual disk formatted with NTFS. In the figure, the low order blocks are the MBR (partition table and boot sector), and are followed by disk partitions. Three important components in the NTFS physical layout are shown: the *(i)* Master File Table ($MFT) (inodes), *(ii)* meta-data blocks (meta-data external to $MFT), and *(iii)* data blocks (logical data). Typically, the size of an $MFT record is 1 KB, and the size of a file system block is 4 KB. We note that files larger than the size of a block will not necessarily be stored contiguously on the virtual disk. In fact, it is quite possible for meta-data and data blocks to be interleaved and small files may be stored within their respective $MFT records.

2.2 Experimental Methodology

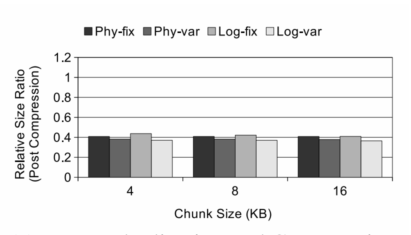
Backup Dataset. For our experiments we use backups taken from five software engineering workstations. This logical dataset consists of approximately 2.25 TB of backup data stored in 1800 large logical backup files. Depending on the user, the time span covered is roughly



(a) Base

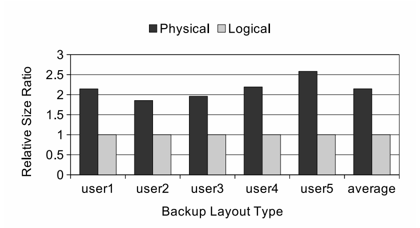


(b) Post Deduplication

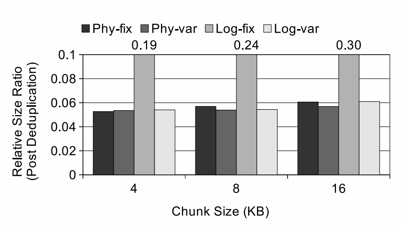


(c) Post Deduplication and Compression

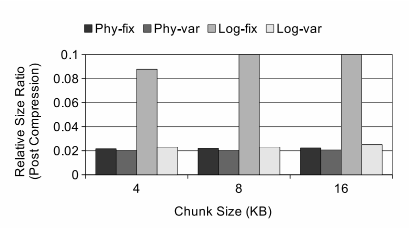
Figure 3: Relative Size Ratios - First Generation



(a) Base



(b) Post Deduplication



(c) Post Deduplication and Compression

Figure 4: Relative Size Ratios - Late Generation

25 to 40 weeks of full backups. This dataset was in turn converted into over 90 TB of virtual disk images for the various experimental cases we studied, representing a non-trival amount of data to be processed. Finally, the logical data (among others) was previously studied w.r.t. deduplication [1, 4, 7] but none of these studies considered physical vs. logical backup.

Virtual Disk Generation. Starting from the original logical backup files, we created a set of virtual disks (specifically VMware VMDKs) in the following way. To create the zeroth generation physical backup (denoted as gen_0), we create a blank virtual disk, which is loopback mounted to our host computer, then create a single NTFS file system partition, and extract (restore) the first full logical backup into the file system. Since virtual disk sizing can affect the results of deduplication across generations, we address this by normalizing the results presented in this paper (discussed further in Section 3.2).

To create the n^{th} physical backup (denoted as gen_n) from the respective full logical backup, we clone the gen_{n-1} virtual disk as its basis. Then, we extract the n^{th} full backup into a temporary storage area, and apply the updates to gen_n using `rsync`[1] to preserve meta-data attributes and perform in-place data file updates. This maintains the file system block allocations between generations for file overwrites, appends, creations, and deletions. However it does not capture all of the churn since we only have backups at 1 week intervals.

Deduplication Simulation. We utilize an in-house deduplication simulator to read in a set of backup files (logical or physical), divide them into chunks (either fixed or variable-sized), calculate the fingerprints of each chunk, and store the fingerprints in an in-memory index. In addition to deduplication, the simulator can also perform GZ compression (which is Lempel-Ziv with Huffman coding) on each unique chunk it finds, as well as delta encoding. As output it reports the backup size resulting from any combination of the three forms of compression. Although our simulator supports other compression types, we only report GZ due to space and time constraints. Differences in compression algorithms typically amount to a trade-off between resulting compressed size and time to compress. Since we are not examining performance in this study, we chose the algorithm that resulted in the best data reduction ratio.

3 Evaluation

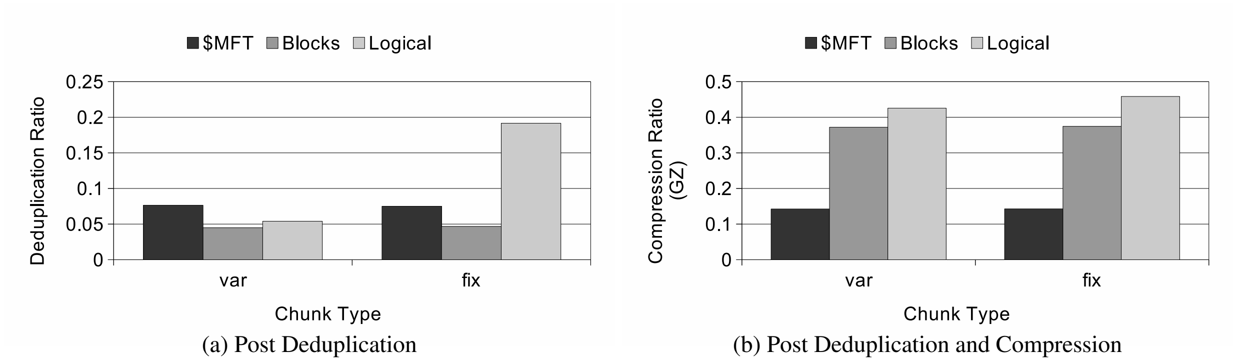
This section presents the results of our data-driven simulation. For each experiment, we report the ratio of the physical size that would be stored on disk vs. the logical data size. For all results, shorter bars indicate better compression. The goal of our evaluation is to answer the following questions w.r.t. physical backup storage efficiency: *(i)* How efficient is it compared to logical backup? *(ii)* How much is it affected by file system metadata? *(iii)* How does file system churn affect it over time? *(iv)* What optimizations can be employed to improve it?

3.1 Efficiency of Physical Backup

To examine the efficiency of physical backup over time, we start with two cases: early and late generation. For early, we consider the results of storing the first backup generation, while for late we consider at least 25 (up to 40 for some users) in order to approximate the steady-state in a backup system which has a multi-week rolling backup window. We compare the results of simulation for the logical backup layout vs. physical. To explore a reasonable range of possibilities, we vary two deduplication parameters: *(i)* chunk size (4, 8, and 16 KB) and *(ii)* chunk type (fixed and variable-sized). When referring to the size of variable chunks, we mean the average.

Figures 3 (a), (b), and (c) report the early generation results, while Figures 4 (a), (b), and (c) report the late

[1]rsync –times –hard-links –executability –acls –xattrs –perms –links –stats –inplace –delete -r source dest



(a) Post Deduplication (b) Post Deduplication and Compression

Figure 5: The Compressibility of Physical vs. Logical - Late Generation

generation results. The height of each bar reports the average size normalized to the logical backup size, and all standard deviations are less than 2%. Figures 3a and 4a present the relative backup data size, for each user. By looking at the right-most pair of bars, we observe that the average physical backup size across the five users is between 2 and 2.5 times that of the logical backup average. Some of this overhead is due to the fact that the virtual disks are not 100% utilized. We expect this to be the common case, since running nearly full disks has many downsides, including performance degradation.

Figures 3b and 4b report the effects of deduplication for both early and late generations. The bars are grouped by chunk size, and present the results from fixed and variable-sized chunking for both physical and logical backups. Early generations do not deduplicate well, and physical backups do worse than logical. This is largely due to the limited opportunity for deduplication in a single backup generation. This also shows the additional file system meta-data overhead in physical vs. logical. Late generations physical backup always does better than logical (Figure 4b). Logical deduplicates almost as well for variable-sized chunking, but does poorly for fixed-sized (exceeds chart scale). Physical does best when the block alignment matches the chunking (i.e., fixed 4 KB). The transition point when physical backup does better than logical occurs after 5 generations, on average.

From Figures 3c and 4c, we observe that including GZ compression halves the utilized storage, as expected. Interestingly, the best results are now given by physical backup variable-chunking across the three chunk sizes. This finding is somewhat counter-intuitive and we explore the reasons for this in the next section.

Although matching the file system block size can seem important for deduplication, using larger variable-sized blocks does as well by getting better GZ compression. Most importantly, physical backup does as well or better than logical backup, giving the combined benefits of faster backup and good storage savings.

3.2 Effects of File System Meta-Data

To understand how physical backup can be more storage efficient than logical, given enough generations, we divide an NTFS file system into two separate regions *(i)* $MFT and *(ii)* external block regions, and pass them into our simulator using 4 KB chunking. By doing so, we can examine the relative compressibility of each region in isolation of the other.

Figures 5 (a) and (b) report the results of the simulations, and they present the average deduplication and GZ compression ratios as $\frac{output\_size}{input\_size}$. All standard deviations are less than 1%. In each figure, we group by chunk type and include bars for the $MFT, external blocks, and logical backup. For the physical cases, *input_size* can be made arbitrarily large by simply creating overly large VMDKs. To avoid these effects, we normalize by using the size of the logical backups for all cases.

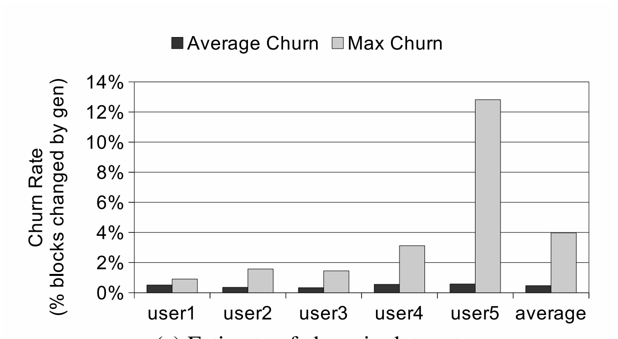
From the figures, we observe first that the $MFT does not deduplicate as well as either blocks or logical, but is more compressible. The $MFT will experience more frequent small changes (e.g., timestamps) making deduplication less effective, but has a lot of repetition of headers and other data structures, making it highly compressible.

We also observe that the external blocks region deduplicates better than the logical backup, while achieving slightly better compression as well. In the logical backup, meta-data is interleaved with data. This has two effects. First, since meta-data is more frequently updated, it reduces the deduplication. Second, without grouping the meta-data together in a contiguous region, it doesn't get the full benefit of GZ compression.

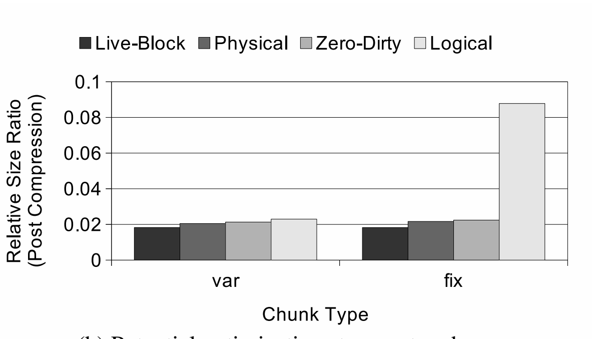
3.3 Effects of File System Churn

A distinguishing characteristic of physical backup is that it experiences file system churn. That is, as a file system ages it accumulates *dirty blocks* (blocks that once held live data, but have now been de-allocated). These dirty blocks are never present in a logical backup. Also, over time as a file system fills up, external fragmentation will occur, causing the data to become non-contiguous. In logical backups, all files are presented contiguously.

We estimate the amount of churn between generations in our data set by measuring the number of unique blocks added. This is an indicator of the level of *unique* chunks that must be stored after deduplication, and the results are shown in Figure 6a. The height of each bar in the chart reports the average churn rate, which is the ratio of unique changed blocks over the total number of blocks



(a) Estimate of churn in data set



(b) Potential optimizations to counter churn

Figure 6: Churn Effects - Late Generation

in a virtual disk. The first five groups of bars are the individual results for each of the five user data sets used in the study and the sixth pair reports the average. For each pair of bars, the left-most bar is the average churn between all subsequent pairs of backup generations, while the right-most bar is the maximum. From the figure, we observe that on average the rate of churn is low related to the size of the virtual disks, but the maximum across generations can be large.

To examine the effects of file system churn, we perform two modifications to the virtual disks. First, we identify and zero out all dirty blocks. Second, we coalesce all live file system blocks from the external block region, maintaining their original order. We also copy the $MFT and concatenate it to the live blocks. This excludes all non-essential portions of a file system (i.e. unallocated blocks, dirty or not). We perform these operations by comparing meta-data from two subsequent physical backup generations. We perform this comparison utilizing the block allocation bitmaps from physical backups to identify dirty and live blocks.

Figure 6b presents the combined results after simulation for both dirty block zeroing (bars labeled *Zero-Dirty*) and external live block coalescing (bars labeled *Live-Block*), along with the results for the base physical and logical backup. Again, we use 4 KB chunking for all the results and the height of each bar reports the relative size ratio after GZ compression.

From the figure, we observe that *Zero-Dirty* exhibits slightly worse storage efficiency than the base physical backup case. We speculate that this is because the average churn rates are low and most dirty blocks, once created, will remain unchanged through many backup generations. By modifying them, we introduce a higher rate of change (i.e. more non-duplicate blocks) than would otherwise exist, reducing storage efficiency.

Alternatively, we observe that *Live-Block* is more effective than the base physical and logical backup cases. We believe that this is because it fully segregates data and meta-data, which may be updated at different frequencies. This eliminates scenarios where small meta-data updates (such as resulting from file reads) can actually affect the chunking of data sections. Also, live block coalescing removes dirty blocks by excluding them (and not introducing zero-filled blocks), further reducing churn.

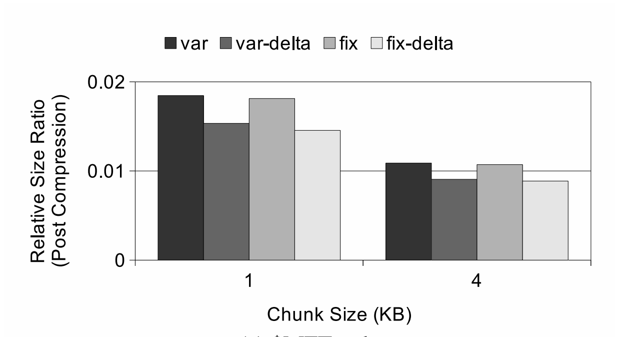
3.4 Potential Optimizations

We now explore two possible optimization techniques to improve physical backup storage efficiency. Within the $MFT, the record size is fixed when the file system is first formatted (default 1 KB). In a typical file system, every block within the $MFT contains four records (with default 4 KB block). A small number of frequently updated records affect a larger number of records due to collocation within the same block. We want to evaluate different chunking sizes. Additionally, applying similarity-based compression (i.e. delta encoding) may be able to take advantage of similarity across records. The key to this is the fact that $MFT records and attributes are very similar in terms of field types and locations (e.g., standard headers, limited number of record types, pervasive use of timestamps, etc.) such that there is likely to be a lot of redundancy, even when they are not identical.

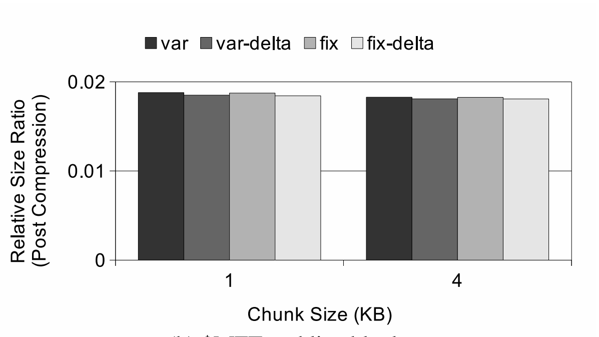
Figure 7a shows the $MFT post compression ratio grouped by chunk size with bars representing variable and fixed-chunking with and without delta encoding. From the figure, we observe that delta encoding improves efficiency when applied to the $MFT, and that it is better to use the larger chunk size (4 KB) even though smaller chunk sizes are generally associated with increased deduplication. In this case, the gain in GZ compression from larger chunks outweighs the loss in deduplication. Although delta encoding can have a positive impact on the storage efficiency of the $MFT, the $MFT is only a portion of the file system. As Figure 7b suggests, the gains from delta encoding only apply to the $MFT and may not translate to overall improvements in storage efficiency when the $MFT is a relatively small.

4 Related Work

The areas of backup storage, deduplication, and virtualization have been covered within the literature, for example in [5, 6, 9]. There are also studies that examine various combinations of these areas. Several papers have explored deduplication for backup storage systems [10, 4]. Jin *et al.* [3] investigates deduplication of virtual images within primary workload environments, while Hutchinson *et al.* [2] looks at backups of physical disks. However, the convergence of all three areas, that is backing



(a) $MFT only



(b) $MFT and live blocks

Figure 7: Potential Optimizations - Small Chunks and Delta Encoding

up virtual disk images to deduplicated storage has, to the best of our knowledge, not been studied. Finally, Tarasov *et al.* [8] utilizes virtual disk introspection to offer possible I/O performance enhancements.

Jin *et al.* [3] looked at deduplication across static virtual disk images which had different operating system or package libraries installed. They did not look at virtual disks with user data and how usage and churn can affect deduplication over time within physical backups. Our study looks at deduplication efficiency as physical backups change across up to forty weeks. Their study also concentrated on the unnormalized deduplication ratio which can be skewed by file system affects such as zero blocks. Instead we use as a baseline comparison the size of logical backups thus eliminating any virtual disk or file system affects which might artificially inflate deduplication such as unallocated blocks.

5 Conclusion

Optimizing the storage usage of physical (VM) backups is becoming increasingly important as more workloads are deployed in VMs. Physical backups are larger than logical backups but are faster to create because they avoid file system traversal. Our study on user workstation data has shown that by using deduplication and compression, physical backups can be compressed smaller than their logical counter-part. This is largely because meta-data is grouped in physical backups (i.e. $MFT area) and achieves greater GZ compression. Additionally, deduplication can be degraded in logical backups because the meta-data, which has more frequent updates, is interleaved with the data. Further analysis of $MFT showed that chunking it at smaller granularity and using delta compression had less impact than compressing using larger chunks. As future work, we intend to extend delta encoding to the entire physical backup, perhaps varying the parameters based on the region being encoded. We also intend to explore the differences in cost (i.e. CPU and memory) between physical vs. logical backups. Finally, we intend to explore the use of other file systems (e.g., ext3/4, btrfs, zfs, etc.) to understand if our current findings are more generally applicable.

6 Acknowledgements

The authors thank Ujwala Tulshigiri for her work on an early version of the virtual disk generation tool. We also thank Stephen Manley, Fred Douglis, Philip Shilane, and Hyong Shim, for helpful comments on earlier drafts, and the anonymous referees and our shepherd, Binny Gill, for their valuable comments and suggestions.

References

[1] DONG, W., DOUGLIS, F., LI, K., PATTERSON, H., REDDY, S., AND SHILANE, P. Tradeoffs in scalable data routing for deduplication clusters. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies* (2011), FAST'11.

[2] HUTCHINSON, N. C., MANLEY, S., FEDERWISCH, M., HARRIS, G., HITZ, D., KLEIMAN, S., AND O'MALLEY, S. Logical vs. physical file system backup. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation* (1999), OSDI '99.

[3] JIN, K., AND MILLER, E. L. The effectiveness of deduplication on virtual machine disk images. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference* (2009), SYSTOR '09.

[4] PARK, N., AND LILJA, D. J. Characterizing datasets for data deduplication in backup applications. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC'10)* (2010), IISWC '10.

[5] QUINLAN, S., AND DORWARD, S. Venti: A new approach to archival data storage. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies* (2002), FAST'02.

[6] ROSENBLUM, M., AND GARFINKEL, T. Virtual machine monitors: Current technology and future trends. *Computer 38*, 5 (May 2005).

[7] SHILANE, P., HUANG, M., WALLACE, G., AND HSU, W. WAN optimized replication of backup datasets using stream-informed delta compression. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies* (2012), FAST'12.

[8] TARASOV, V., JAIN, D., HILDEBRAND, D., TEWARI, R., KUENNING, G., AND ZADOK, E. Improving I/O Performance using Virtual Disk Introspection. In *Proceedings of the 5th USENIX Workshop on Hot Topics in Storage and File Systems* (2013), HotStorage'13.

[9] WALLACE, G., DOUGLIS, F., QIAN, H., SHILANE, P., SMALDONE, S., CHAMNESS, M., AND HSU, W. Characteristics of backup workloads in production systems. In *Proceedings of the 10th USENIX conference on File and Storage Technologies* (2012), FAST'12.

[10] ZHU, B., LI, K., AND PATTERSON, H. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies* (2008), FAST'08.