Characteristics of Backup Workloads in Production Systems

Grant Wallace Fred Douglis Hangwei Qian* Philip Shilane Stephen Smaldone
Mark Chamness Windsor Hsu

Backup Recovery Systems Division
EMC Corporation

Abstract

Data-protection class workloads, including backup and long-term retention of data, have seen a strong industry shift from tape-based platforms to disk-based systems. But the latter are traditionally designed to serve as primary storage and there has been little published analysis of the characteristics of backup workloads as they relate to the design of disk-based systems. In this paper, we present a comprehensive characterization of backup workloads by analyzing statistics and content metadata collected from a large set of EMC Data Domain backup systems in production use. This analysis is both broad (encompassing statistics from over 10,000 systems) and deep (using detailed metadata traces from several production systems storing almost 700TB of backup data). We compare these systems to a detailed study of Microsoft primary storage systems [22], showing that backup storage differs significantly from their primary storage workload in the amount of data churn and capacity requirements as well as the amount of redundancy within the data. These properties bring unique challenges and opportunities when designing a disk-based filesystem for backup workloads, which we explore in more detail using the metadata traces. In particular, the need to handle high churn while leveraging high data redundancy is considered by looking at deduplication unit size and caching efficiency.

1 Introduction

Characterizing and understanding filesystem content and workloads is imperative for the design and implementation of effective storage systems. There have been numerous studies over the past 30 years of file system characteristics for general-purpose applications [1, 2, 3, 9, 15, 20, 22, 26, 30, 31], but there has been little in the way of corresponding studies for backup systems.

Data backups are used to protect primary data. They might typically consist of a full copy of the primary data once per week (i.e., a weekly full), plus a daily backup of the files modified since the previous backup (i.e., a daily incremental). Historically, backup data has been written to tape in order to leverage tape's low cost per gigabyte and allow easy transportation off site for disaster recovery. In the late 1990s, virtual tape (or "VTL") was introduced, which used hard disk storage to mimic a tape library. This allowed for consolidation of storage and faster restore times. Beginning in the early 2000s, deduplicating storage systems [10, 34] were developed, which removed data redundancy and extended the benefits of disk-based backup storage by lowering the cost of storage and making it more efficient to copy data off-site over a network for disaster recovery (replication).

The transition from tape to VTL and deduplicating disk-based storage has seen a strong adoption by the industry. In 2010 purpose-built backup appliances protected 468PB and are projected to protect 8EB by 2015, by which time this will represent a $3.5B market [16]. This trend has made a detailed study of backup filesystem characteristics pertinent for system designers if not long overdue.

In this paper we first analyze statistics from a broad set of 10,000+ production EMC Data Domain systems [12]. We also collect and analyze content-level snapshots of systems that, in aggregate, are to our knowledge at least an order of magnitude larger than anything previously reported. Our statistical analysis considers information such as file age, size, counts, deduplication effectiveness, compressibility, and other metrics. Comparing this to Meyer and Bolosky's analysis of a large collection of systems in Microsoft Corp. [22], we see that backup workloads tend to have shorter-lived and larger files than primary storage. This is indicative of higher data churn rates, a measure of the percentage of storage capacity that is written and deleted per time interval (e.g., weekly), as well as more data sequentiality. These have implications for the design requirements for purpose-built backup systems.

*Current affiliation: Case Western Reserve University.

While summary statistics are useful for analyzing overall trends, we need more detailed information to consider topics such as performance analysis (e.g., cache hit rates) or assessing the effect of changes to system configurations (e.g., varying the unit of deduplication). We address this with our second experimental methodology, using simulation from snapshots representing content stored on a number of individual systems. The content metadata includes detailed information about individual file content, but not the content itself. For example, deduplicating systems will break files into a series of chunks with each chunk represented by a strong hash, sometimes referred to as a fingerprint. We collect the lists of chunk fingerprints and chunk sizes that represent each file as well as the physical layout of these chunks on disk. These collections represent almost 700TB of data and span various data types including databases, emails, workstation data, source code, and corporate application data. These allow us to analyze the stream or file-wise behavior of backup workloads. This type of information is particularly helpful in analyzing the effectiveness of deduplication parameters and caching algorithms.

This study confirms and highlights the different requirements between backup and primary storage. Whereas primary storage capacities have grown rapidly (the total amount of digital data more than doubles every two years [13]), write throughput requirements have not needed to scale as quickly because only a small percentage of the storage capacity is written every week and most of the bytes are longer lived. Contrast this with the throughput requirements of backup systems which, for weekly full backups, must ingest the entire primary capacity every week. The implication is that backup filesystems have had to scale their throughput to meet storage growth. Meeting these demands is a real challenge, and this analysis sheds light on how deduplication and efficient caching can help meet that demand.

To summarize our contributions, this paper:

- analyzes more than 10,000 production backup systems and reports distributions of key metrics such as deduplication, contents, and rate of change;
- extensively compares backup storage systems to a similar study of primary storage systems; and
- uses a novel technique for extrapolating deduplication rates across a range of possible sizes.

The remainder of this paper is organized into the following sections: §2 background and related work, §3 data collection and analysis techniques, §4 analysis of broad trends across thousands of production systems, §5 exploring design alternatives using detailed metadata traces of production systems, and §6 conclusions and implications for backup-specific filesystem design.

2 Background and Related Work

We divide background into three areas: backups (§2.1), deduplication (§2.2), and data characterization (§2.3).

2.1 Backups

Backup storage workloads are tied to the applications which generate them, such as EMC NetWorker or Symantec NetBackup. These backup software solutions aggregate data from online file systems and copy them to a backup storage device such as tape or a (deduplicating) disk-based storage system [7, 34]. As a result, individual files are typically combined into large units, representing for example all files backed up on a given night; these aggregates resemble UNIX "tar" files. Many other types of backup also exist, such as application-specific database backups. Backups usually run regularly, with the most common paradigm being weekly "full" backups and daily "incremental" backups. When files are modified, the incremental backups may have large portions in common with earlier versions, and full backups are likely to have many of their comprising files completely unmodified, so the same data gets written to the backup device again and again.

2.2 Deduplication and other Data Reduction

In backup storage workloads the inherent high degree of data redundancy and need for high throughput make deduplicating techniques important. Deduplication can be performed at the granularity of entire files (e.g., Windows 2000 [5]), fixed blocks (e.g., Venti [29]), or variable-sized "chunks" based on content (e.g., LBFS [24]). In each case, a strong hash (such as SHA-1) of the content, i.e., its "fingerprint," serves as a unique identifier. Fingerprints are used to index content already stored on the system and eliminate duplicate writes of the same data. Because content-defined chunks prevent small changes in content from resulting in unique chunks throughout the remainder of a file, and they are used in the backup appliances we have analyzed, we assume this model for the remainder of this paper. Backup data can be divided into content-defined chunks on the backup storage server, on the backup software intermediary (e.g., a NetBackup server), or on the systems storing the original data. If chunked prior to transmission over a network, the fingerprints of the chunks can first be sent to the destination, where they are used avoid transferring those chunks already present [11, 24].

Traditional compression, such as *gzip*, complements data deduplication. We refer to such compression as "local" compression to distinguish it from compression obtained from identifying multiple copies of data, i.e., deduplication. The systems under study perform local compression after deduplication, combining unique chunks into "compression regions" that

are compressed together to improve overall data reduction.

There is a large body of research and commercial efforts on optimizing [14, 21, 34], scaling [8, 10], and improving the storage efficiency [17] of deduplicating backup systems. Our efforts here are mostly complementary to that work, as we are characterizing backup workloads rather than designing a new storage architecture. The impact of the chunk size has been explored in several studies [17, 18, 22, 28], as has delta-encoding of content-defined chunks [18]. However, our study of varying chunk sizes (§5.1) uses real-world workloads that are substantially larger than those used in previous studies. We also develop a novel technique for using a single chunk size to extrapolate deduplication at larger chunk sizes. This is different from the methodology of Kruus, et al. [17], which decides on the fly what chunk size to use at a particular point in a data stream, using the actual content of the stream. Here, we use just the fingerprints and sizes of chunks to form new "merged chunks" at a coarser granularity. We evaluate the effectiveness of this approach by comparing metrics from the approximated merged chunks and native chunking at different sizes, then evaluate the effectiveness of chunking various large-scale datasets over a range of target chunk sizes.

2.3 Data Characterization

The closest work to ours in topic, if not depth, is Park and Lilja's backup deduplication characterization study [27]. It uses a small number of truncated backup traces, 25GB each, to evaluate metrics such as rate of change and compression ratios. By comparison, our paper considers a larger set of substantially larger traces from production environments and aims at identifying filesystem trends related to backup storage.

There have been many studies of primary storage characteristics [1, 2, 3, 9, 15, 20, 22, 26, 30, 31], which have looked at file characteristics, access patterns and caching behavior for primary workloads. Our study measures similar characteristics but for backup workloads. It is interesting to compare the different characteristics between backup and primary storage systems (see §4). For comparison data points we use the most recent study from Microsoft [22], which contains a series of large-scale studies of workstation filesystems. There are some differences that arise from the difference in usage (backup versus day-to-day usage) and some that arise from the way the files are accessed (aggregates of many files versus individual files). For example, the ability to deduplicate whole files may be useful for primary storage [5] but is not applicable to a backup environment in which one file is the concatenation of terabytes of individual files.

3 Data Collection and Analysis Techniques

In conducting a study of file-system data, the most encompassing approach would be to take snapshots of all the systems' data and archive them for evaluation and analysis. This type of exercise would permit numerous forms of interesting analysis including changes to system parameters such as average chunk size and tracking filesystem variations over time.

Unfortunately, full-content snapshots are infeasible for several reasons, the primary one being the need to maintain data confidentiality and privacy. In addition, large datasets (hundreds of terabytes in size each) become infeasible to work with because of the long time to copy and process and the large capacity required to store them. The most practical way of conducting a large-scale study is to instead collect filesystem-level statistics and content metadata (i.e., data about the data).

For this study we collect and analyze two classes of data with the primary aim of characterizing backup workloads to help design better protection storage systems. The first class of data is *autosupport* reports from production systems. Customers can choose to configure their systems to automatically generate and send autosupports, which contain system monitoring and diagnostic information. For our analysis, we extract aggregate information from the autosupports such as file statistics, system capacity, total bytes stored, and others.

The second type of information collected is detailed information about data contained on specific production systems. These collections contain chunk-level metadata such as chunk hash identifiers (fingerprints), sizes, and location on disk. Because of the effort and storage needed for the second type of collection, they are obtained from only a limited set of systems.

The two sets of data complement each other: the autosupports (§3.1) are limited in detail but wide in deployment, while the content metadata snapshots (§3.2) contain great detail but are limited in deployment.

3.1 Collecting Autosupports

The Data Domain systems that are the subject of this study send system data back to EMC periodically, usually on a daily basis. These autosupport reports contain diagnostic and general system information that help the support team monitor and detect potential problems with deployed systems [6]. Over 10,000 of these reports are received per day, which makes them valuable in understanding the broad characteristics of protection storage workloads. They include information about storage usage, compression, file counts and ages, caching statistics and other metrics. Among other things, they can help us understand the distribution of deduplication rates, capacity usage, churn and file-level statistics.

For our analysis, we chose autosupports from a one-week period. From the set of autosupports, we exclude some systems based on certain validation criteria: a system must have been in service more than 3 months and have more than 2.5% of its capacity used. The remaining set consists of more than 10,000 systems with system ages ranging from 3 months to 7 years and gives a broad view of the usage characteristics of backup systems.

We consider these statistics in the aggregate; there is no way to subdivide the 10,000 systems by content type, backup software, or other similar characteristics. In addition, we must acknowledge some possible bias in the results. This is a study of EMC Data Domain customers who voluntarily provide autosupport data (the vast majority of them do); these customers tend to use the most common brands of backup software and typically have medium to large computing environments to protect.

3.2 Collecting Content Metadata

In this study, we work with deduplicated stores which enable us to collect content metadata more efficiently. On deduplicated systems a chunk may be referenced many times, but the detailed information about the chunk need be stored just once. Figure 1 shows a schematic of a deduplicated store. We collect the file recipes (listing of chunk fingerprints) for each file and then collect the deduplicated chunk metadata from the storage containers, as well as sub-chunk fingerprints (labeled "sub-fps") as described below. The file recipe and per-chunk metadata can be later combined to create a per-file "trace" comprised of a list of detailed chunk statistics as depicted on the bottom right of the figure. (Note that this "trace" is not a sequence of I/O operations but rather a sequence of file chunk references that have been written to a backup appliance, from oldest to newest.) Details about the trace, including its efficient generation, are described in §3.2.3.

In this way, the collection time and storage needed for the trace data is proportional to the deduplicated size. This can lead to almost a 10X saving for a typical backup storage system with 10X deduplication. In addition, some of the data analysis can be done on the deduplicated chunk data. This type of efficiency becomes very important when dealing with underlying datasets of hundreds of terabytes in size. These systems will have tens of billions of chunks and even the traces will be hundreds of gigabytes in size.

3.2.1 Content Fields Collected

For the content metadata snapshots, we collect the following information (additional data are not discussed due to space limitations):

- Per-chunk information such as size, type, SHA-1 hash, subchunk sizes and abbreviated hashes.

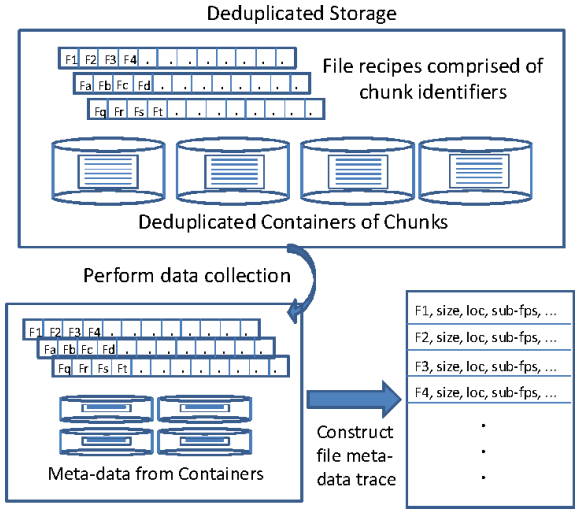


Figure 1: Diagram of Data Collection Process

- Per-file information such as file sizes, modification times, and fingerprints of each chunk in the file.
- Disk layout information such as location and grouping of chunks on disk.

One of the main goals for these collections was to look at throughput and compression characteristics with different system configurations. The systems studied were already chunked at 8KB on average with the corresponding SHA-1 hash values available. We chose to sub-chunk each 8KB chunk to, on average, 1KB and collected abbreviated SHA-1 hashes for each 1KB sub-chunk. Sub-chunking allowed us to investigate deduplication rates at various chunk sizes smaller than the default 8KB, as described in §5.1.

3.2.2 Creating Traces from Metadata Snapshots

The collected content metadata can be used to create per-file traces of chunk references. These traces are the ordered list of chunk metadata that comprise a file. For example, the simplest file trace would contain a file-ordered list of the chunk fingerprints and sizes that comprise the file. More detailed traces might also include other per-chunk information such as disk location. These file traces can be run through a simulator or analyzed in other ways for metrics such as deduplication or caching efficiency.

The per-file traces can be concatenated together, for example by file modification time (mtime), to create a representative trace for the entire dataset. This can be used to simulate reading or writing all or part of the system contents; our analyses in §5 are based on such traces.

For example, to simulate a write workload onto a new system, we could examine the sequence of fingerprints in order and pack new (non-duplicate) chunks together into

storage containers. The storage containers could then be used as the unit of caching for later fingerprints in the sequence [34]. This layout represents a pristine storage system, but in reality, chunk locality is often fragmented because garbage collection of deleted files causes live chunks from different containers to be merged into new containers. Instead of using the pristine layout, we could use the container layout of chunks as provided by the metadata snapshot from the production system, which gives a more realistic caching analysis.

To simulate a read workload, we would examine the sequence of fingerprints in order and measure cache efficiency by analyzing how many container or compression-region loads are required to satisfy the read. Compression-regions are the minimal unit of read, since the group of chunks have to be uncompressed together, but reading whole containers may improve efficiency. While reading the entire dataset trace would be a complete restore of all backups, perhaps more realistically, only the most recent full backup should be read to simulate a restore.

To approximate the read or write of one full backup of the trace requires knowledge of what files correspond to a backup. Since we don't have the backup file catalog, we are not able to determine a full backup at file granularity. Instead we divide the trace into a number of equal sized intervals, with the interval size based on the deduplication rate. For instance, if the deduplication rate is 10X then we estimate that there are about 10 full backups on the system, i.e., the original plus 9 identical copies. In this example we would break the trace into 10 intervals approximating about one backup per interval. This is an approximation: in practice, the subsequent backups after the first will not be identical but will have some data change. But this is a reasonable approach for breaking the caching analysis into intervals, which allows for warming the cache and working on an estimated most-recent backup copy.

3.2.3 Efficient Analysis of Filesystem Metadata

The file trace data collected could be quite large, sometimes more than a terabyte in size, and analyzing these large collections efficiently is a challenge. Often the most efficient way to process the information is by use of out-of-core sorting. For instance, to calculate deduplication ratios we sort by fingerprint so that repeated chunks are adjacent, which then allows a single scan to calculate the unique chunk count. As another example, to calculate caching effectiveness we need to associate fingerprints with their location on disk. We first sort by fingerprint and assign the disk location of the first instance to all duplicates, then re-sort by file mtime and offset to have a time-ordered trace of chunks, with container locations, to evaluate.

Even the process of merging file recipes with their associated chunk metadata to create a file trace would be prohibitively slow without sorting. We initially implemented this merge in a streaming fashion, looking up chunk locations and pre-fetching neighboring chunks into a cache, much as an actual deduplication system would handle a read. But the process was slow because of the index lookups and random seeks on an engineering workstation with a single disk. Eventually we switched this process to also use out-of-core sorting. We use a four-step process of (1) sorting the file recipes by fingerprint, (2) sorting the chunk metadata collection by fingerprint, (3) merging the two sets of records, and (4) sorting the final record list by logical position within the file. This generates a sequence of chunks ordered by position within the file, including all associated metadata.

4 Trends Across Backup Storage Systems

We have analyzed the autosupport information from more than 10,000 production deduplicated filesystems, taken from an arbitrary week, July 24, 2011. We compare these results with published primary storage workloads from Microsoft Corp. [22]. The authors of that study shared their data with us, which allows us to graph their primary workload results alongside our backup storage results. The Microsoft study looked at workstation filesystem characteristics for several different time periods; we compare to their latest, a one month period in 2009 which aggregates across 857 workstations.

Backup storage file characteristics are significantly different from the Microsoft primary workload. Data-protection systems have generally larger, fewer and shorter lived files. This is an indication of more churn within the system but also implies more data sequentiality. The following subsections detail some of these differences. In general, figures present both a histogram (probability distribution) and a cumulative distribution function (CDF), and when counts are presented they are grouped into buckets representing ranges, on a log scale, with labels centered under representative buckets.

4.1 File Size

A distinguishing characteristic between primary and backup workloads is file size. Figure 2 shows the file size distribution, weighted by bytes contained in the files, for both primary and backup filesystems. For backup this size distribution is about 3 orders of magnitude larger than for primary files. This is almost certainly the result of backup software combining individual files together from the primary storage system into "tar-like" collections. Larger files reduce the likelihood of whole-file deduplication but increase the stream locality within the system. Notice that for backup files a large percentage of the space is used by files hundreds of gigabytes in

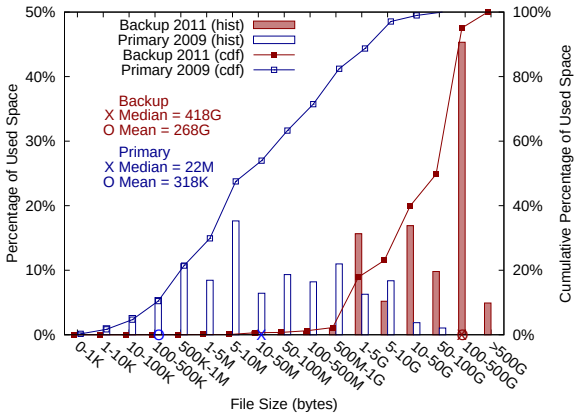


Figure 2: File size

size. Small file optimizations that make sense for primary storage such as embedding data in inodes or use of disk block fragments may not make sense for backup filesystems where large allocation units can provide more efficient metadata use.

4.2 File and Directory Count

File and directory counts are typically much lower in backup workloads. Similar to the effect of large file sizes, having a low file count (Figure 3(a)) results from having larger tar-type concatenations of protected files. The low directory count (Figure 3(b)) is a result of backup applications using catalogs to locate files. This is different from typical user-organized filesystems where a directory hierarchy is used to help order and find data. Looking at the ratio of file to directory count (Figure 3(c)), we can see again that backup workloads tend to use a relatively flat hierarchy with several orders of magnitude more files per directory.

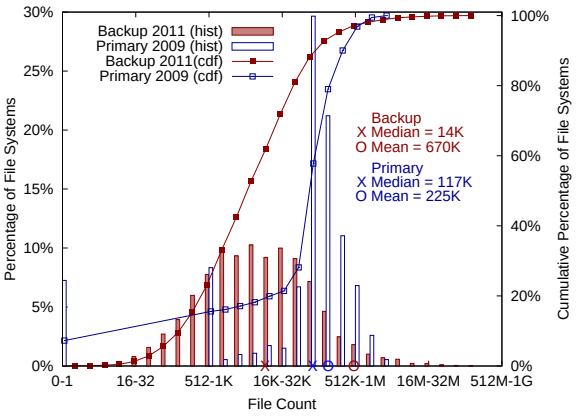
4.3 File Age

Figure 4 shows the distribution of file ages weighted by their size. For backup workloads the median age is about 3 weeks. This would correspond to about 1/2 the retention period, implying data retention of about 6 weeks. Short retention periods lead to higher data churn, as seen next.

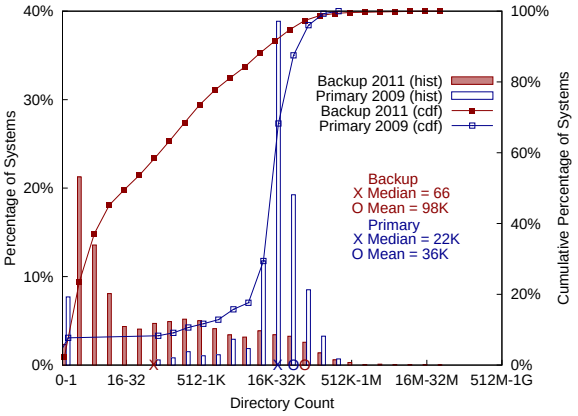
4.4 Filesystem Churn

Filesystem churn is a measure of the percentage of storage that is freed and then written per time period, for instance in a week. Figure 5 shows a histogram of the weekly churn occurring across the studied backup systems.

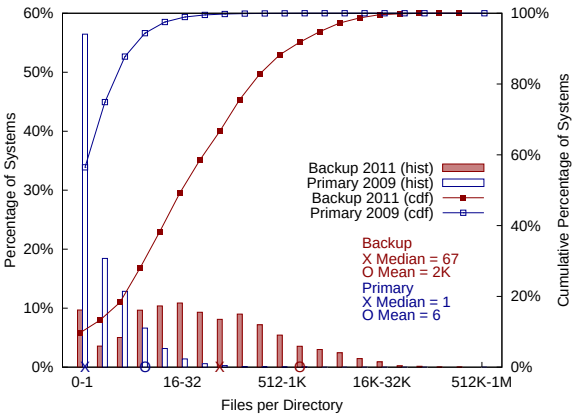
On average about 21% of the total stored data is freed and written per week. This high churn rate is driven by backup retention periods. If a backup system has a 10-week retention policy, about 10% of the data needs to be



(a) File count



(b) Directory count



(c) Files per directory

Figure 3: File and directory counts

written and deleted every week. The median churn rate is about 17%, corresponding to almost a 6-week retention period, which correlates well with the median byte age of about 3 weeks.

This has implications for backup filesystems: such filesystems must be able not only to write but also re-

claim large amounts of space on a weekly basis. Storage technologies with limited erase cycles, such as flash memory, may not be a good fit for this workload without care to avoid arbitrary overwrites from file system cleaning [23].

To some extent, deduplicating storage systems help alleviate this problem because less physical data needs to be cleaned and written each week. The ratio of data written per week to total stored is similar whether those are calculated from pre-deduplication file size or post-deduplicated storage size; this is expected as long as the deduplication ratio is relatively constant over time.

Note also that backup churn rates increase quickly over time. They follow the same growth rate as the underlying primary data, (i.e., doubling every two years). To meet the high ingest rates, backup filesystems can leverage the high data redundancy of backup workloads. In-line deduplication of file streams can eliminate many of the disk writes and increase throughput. Doing so effectively requires efficient caching, which is studied further in §5.

4.5 Read vs Write Workload

Data-protection systems are predominately write workloads but do require sufficient read bandwidth in order to stream the full backup to tape, replicate changed data offsite, and provide for timely restores. Figure 6 shows the distribution of the ratio of bytes written vs total I/O bytes, excluding replication and garbage collection. About 50% of systems have overwhelmingly more writes than reads (90%+ write). Only about 20% of systems have more reads than writes.

These I/O numbers underestimate read activity because they do not include reads for replication. However, since during replication an equal number of bytes are read by the source as written by the destination, the inclusion of these statistics might change the overall percentages but not change the conclusion that writes predominate.

This is the opposite of systems with longer-lived bytes such as primary workloads, which typically have twice as many reads as writes [20]. The high write workloads are again indicative of high-churn systems where large percentages of the data are written every week.

4.6 Replication

For disaster recovery, backup data is typically replicated off-site to guard against site-level disasters such as fires or earthquakes. About 80% of the production systems replicate at least part of their backup data each week.

Of the systems that replicate, Figure 7 shows the percentage of bytes written in the last 7 days that are also replicated (either to or from the system). On average almost 100% of the data is replicated on these systems.

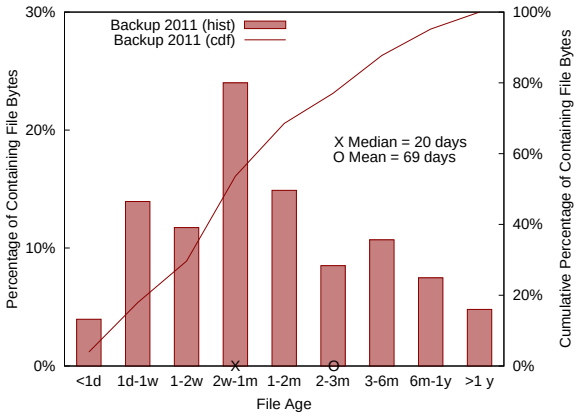


Figure 4: File age

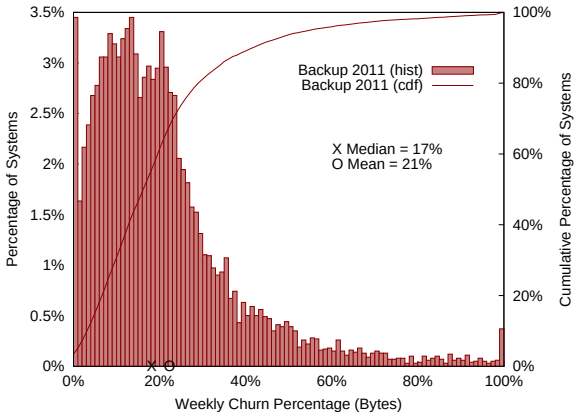


Figure 5: Weekly churn

Notice that some systems replicate more data than was written in this time period. This can be due to several causes: some systems replicate to more than one destination and some systems perform cascaded replication (they receive replicated data and in turn replicate it to another system).

The high percentage of replicated data increases the need for read throughput, resulting in a slightly more balanced read to write ratio than one might expect from just backup operations (write once, read rarely). This implies that while backup systems must provide excellent write performance, they cannot ignore the importance of read performance.

In concurrent work, cache locality for delta compression is analyzed in the context of replication, including information from production autosupport results [32].

4.7 Capacity Utilization

Figure 8 shows the distribution of storage usage for both primary and backup systems. Backup systems skew toward being more full than primary systems, with the

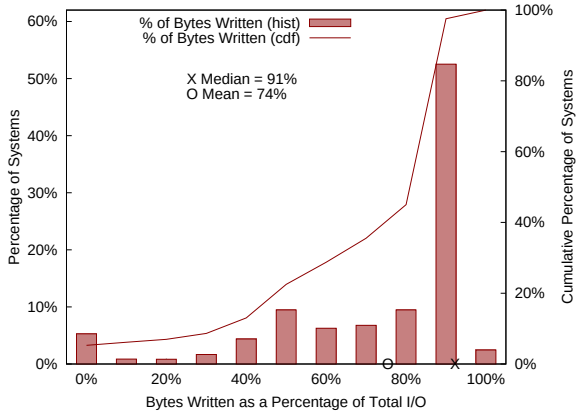


Figure 6: Read/write byte ratio

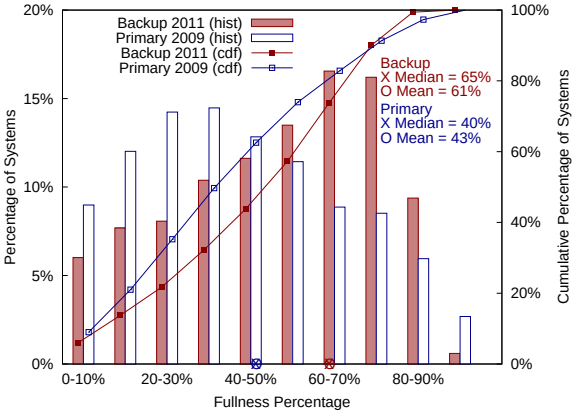


Figure 8: Fullness

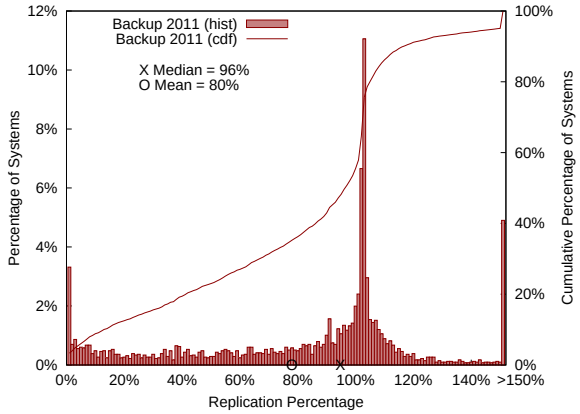


Figure 7: Percent of data replicated

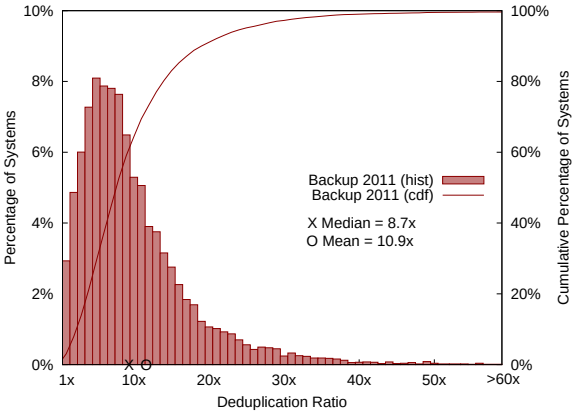


Figure 9: Deduplication

backup system modal (most frequent) utilization about 60–70% full. In contrast primary systems are about 30–40% full. The gap in mean utilization may reflect differences in the goals of the administrators of the two types of systems: while performance and capacity are both important in each environment, there is a greater emphasis in data protection on dollar-efficient storage, while primary storage administrators may stress performance. Also, backup administrators have more flexibility in balancing the data protection workloads across systems, as they can shorten retention periods or reduce the domain of protected data. Achieving higher utilization helps to optimize the cost of overall backup storage [6].

4.8 Deduplication Rates

The amount of data redundancy is one of the key characteristics of filesystem workloads and can be a key driver of cost efficiency in today's storage systems. Here we compare the deduplication rates of backup filesystem workloads with those of primary storage as reported by Meyer and Bolosky [22]. Figure 9 indicates that deduplication rates for backup storage span a wide range across system and workloads with a mean of 10.9x. This is dramatically higher than for primary workloads with a mean of about 3x in the Microsoft workload. The main difference is that backup workloads generally retain multiple copies of data.

Additionally, backups are usually contained within large tar-type archives that do not lend themselves to whole-file deduplication. When these larger files are subdivided into chunks for deduplication, the chunk size can have widely varying effects on deduplication effectiveness (see §5.1).

4.8.1 Compression

Data Domain systems aggregate new unique chunks into compression regions, which are compressed as a single unit (approximately 128KB before compression). Since there is usually spatial locality between chunks that are written together, the compressibility of the full region is much greater than what might be achieved by compressing each 8KB chunk in isolation.

#	Snapshot	Class	Data Type	Size (TB)	Dedup. Ratio	1-Wk Dedup.	MedAge (Weeks)	Retention Time	Update Freq.
1	homedirs	LT-B	Home Directories	201	14.0	3.0	3.49	1–3 years	MF
								5 weeks	DF
2	database1	B	Database	177	5.1	2.7	2.21	1 month	MF/DI
3	email	B	Email	146	9.6	1.1	1.36	15 days	DF
4	fileservers	B	Windows Fileservers	60	5.9	1.7	5.80	3 months	WF/DI
5	mixed1	B	Mixed DB/Email/User	47	6.0	2.4	3.24	1–3 months	MF/DI
6	mixed2	B	Workstations, Servers	43	11.0	3.0	9.44	4–6 months	WF/DI
7	workstations	B	Workstations	4.5	7.5	2.3	13.56	4 months	WF/DI
8	database2	B	Database	3.8	2.2	1.3	0.23	3 days	DF

Table 1: Collected Datasets. Class can be **B** "Backup," **LT-B** or"Long Term Backup." Retention can be **MF** "Monthly Full," **WF** "Weekly Full," **DF** "Daily Full," or **DI** "Daily Incremental." We report cumulative and one-week deduplication. **MedAge** is the mid-point at which half the data is newer or older.

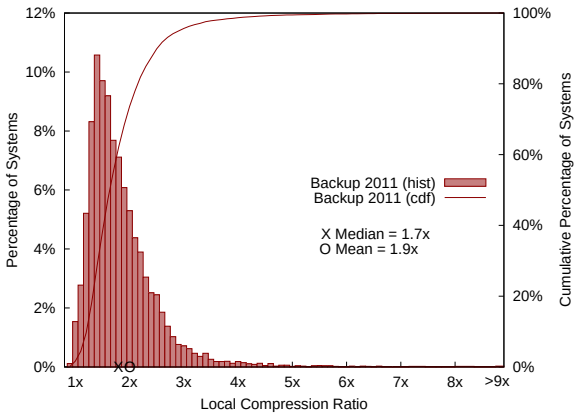


Figure 10: Local compression

As a result, the effectiveness of post-deduplication compression in a backup workload will typically be comparable to that of a primary workload. Figure 10 shows the local compression we see across production backup workloads, with a mean value of almost 2X as the expected rule of thumb [19]. But as can be seen, there is also a large variation across systems with some data inherently more random than others.

5 Sensitivity Analyses of Deduplicating Backup Systems

Deduplication has enabled the transition from tape to disk-based data protection. Storing multiple protected copies on disk is only cost effective when efficient removal of data redundancy is possible. In addition deduplication provides for higher write throughput (fewer disk writes), which is necessary to meet the high churn associated with backup storage (see §4.4). However, read performance can be negatively impacted by the fragmentation introduced by deduplication [25].

In this section we use trace-driven simulation to evaluate the effect of chunk size on deduplication rates (§5.1) and to evaluate alternatives for caching the fingerprints used for detecting duplicates (§5.2). First we describe the metadata collections, which are used for the sensitivity analyses, in greater detail. Table 1 lists the data sets collected and their properties, in decreasing order of pre-deduplication size. They span a wide range of sizes and deduplication rates. Most are straightforward "backup" workloads while one includes some data meant for long-term retention. They range from traces representing as little as 4–5TB of pre-deduplicated content up to 200TB. The deduplication ratio (using the default 8KB target chunk size) also has a large range, from as little as 2.2 to as much as 14.0; the data sets with the lowest deduplication have relatively few full backups, with the extreme case being a mere 3 days worth of daily full backups.

Deduplication over a prolonged period can be substantial if many backups are retained, but how much deduplication is present over smaller windows, and how skewed is the stored data? These metrics are represented in the **1-Wk Dedup.** and **MedAge** columns. The former estimates the average deduplication seen within a single week, which typically includes a full backup plus incrementals. This is an approximation of the intra-full deduplication which cannot be determined directly because the collected datasets do not provide information about full backup file boundaries. The median age is the point by which half the stored data was first written, and it provides a view into the retention and possible deduplication. For instance, half of the data in `homedirs` had

been stored for 3.5 weeks or less. With daily full backups stored 5 weeks we would expect a median age of 2.5 weeks, but the monthly full backups compensate and increase the median.

5.1 Effect of Varying Chunk Size

The systems studied use a default average chunk size of 8KB, but smaller or larger chunks are possible. Varying the unit of deduplication has been explored many times in the past, usually by chunking the data at multiple sizes and comparing the deduplication achieved [18, 22, 28]; it is also possible to vary the deduplication unit dynamically [4, 17]. The smaller the average chunk size, the finer-grained the deduplication. When there are long regions of unchanged data, the smaller chunk size has little effect, since any chunk size will deduplicate equally well. When there are frequent changes, spaced closer together than a chunk, all chunks will be different and fail to deduplicate. But when the changes are sporadic relative to a given chunk size, having smaller chunks can help to isolate the parts that have changed from the parts that have not.

5.1.1 Metadata Overhead

Since every chunk requires certain metadata to track its location, the aggregate overhead scales inversely with the chunk size. We assume a small fixed cost, 30 bytes, per physical chunk stored in the system and the same cost per logical chunk in a file recipe (where physical and logical are post-deduplication and pre-deduplication, respectively). The 30 bytes represent the cost of a fingerprint, chunk length, and a small overhead for other metadata.

Kruus, et al., described an approach to chunking data at multiple granularities and then selecting the most appropriate size for a region of data based on its deduplication rate [17]. They reported a reduction in deduplication effectiveness by a factor of $\frac{1}{(1+f)}$, where f is defined as the metadata size divided by the average chunk size. For instance, with 8KB chunks and 30 bytes of metadata per chunk, this would reduce the effectiveness of deduplication by 0.4%.

However, metadata increases as a function of both post-deduplication physical chunks and pre-deduplication logical chunks, i.e., it is a function of the deduplication rate itself. If the metadata for the file recipes is stored outside the deduplication system, the formula for the overhead stated above would be correct. If the recipes are part of the overhead, we must account for the marginal costs of each logical chunk, not only the post-deduplication costs. Since the raw deduplication D is the ratio of logical to physical size (i.e., $D = L/P$) while the real deduplication D' includes metadata costs ($D' = \frac{L}{P+fP+fL}$), we can substitute $L = DP$ in the latter equation to get:

$$D' = \frac{D}{1+f(1+D)}.$$

Intuitively, we are discounting the deduplication by the amount of metadata overhead for one copy of the physical data and D copies of the logical data. For a deduplication rate of 10X, using this formula, this overhead would reduce deduplication by 4% rather than 0.4%.

However, as chunks get much smaller, the metadata costs for increasing the number of chunks can dominate the savings from a smaller chunksize. We can calculate the breakeven point at which the net physical space using chunksize C_1 is no greater than using twice that chunksize (C_2, where $C_2 = 2C_1$). First, we note that $f_1 = 2f_2$ since the per-chunk overhead doubles. Then we compare the total space (physical post-deduplication P_i plus overhead) for both chunk sizes, using a single common logical size L:

$$P_1 + 2f(L+P_1) \leq P_2 + f(L+P_2).$$

Since $D_i = L/P_i$ we can solve for the necessary D_1:

$$D_1 \geq \frac{D_2(1+2f)}{1+f(1-D_2)}.$$

This inequality shows where the improvement in raw deduplication (not counting metadata) is at least as much as the increased metadata cost.[1] As an example, with the 30 bytes of overhead and 10X raw deduplication at **2**KB chunks, one would need to improve to 11.9X or more raw deduplication at the **1**KB chunk size to fare at least as well.

5.1.2 Subchunking and Merging Chunks

We are able to take snapshots of fingerprints but not of content, so it is not possible to rechunk content at many sizes. While we could chunk data from a system at numerous sizes at the time the snapshot is created, that would require more processing and more storage than are feasible. Thus, to permit the analysis of pre-chunked data, for which we can later store the fingerprints but not the content, we take a novel approach. To get smaller chunks than the native 8KB size, during data collection we read in a chunk at its original size, *sub*-chunk it at a single smaller size (1KB), and store the fingerprints and sizes of the smaller sub-chunks along with the original chunk metadata. We can then analyze the dataset with 1KB chunks, or merge 1KB chunks into larger chunks

[1] There is also a point at which the deduplication at one size is so high that the overhead from doubling the metadata costs would dominate any possible improvement from better deduplication, around 67X for our 30-byte overhead. Also, the formula applies to a single factor of two but could be adjusted to allow for other chunk sizes.

(such as 2KB or 4KB on average). We can also merge the original 8KB chunks into larger units (powers of two up to 1MB). To keep the merged chunks distinct from the native 8KB chunks or the 1KB sub-chunks, we will refer to merged chunks as *m*chunks.

For a given average chunk size, the system enforces both minimum and maximum sizes. To create an *m*chunk within those constraints, we group a minimum number of chunks (or sub-chunks) to reach the minimum size, then determine how many additional chunks to include in the *m*chunk in a content-aware fashion, similar to how chunks are created in the first place. For instance, to merge 1KB chunks into 4KB *m*chunks (2KB minimum and 6KB maximum), we would start with enough 1KB-average chunks to create at least a 2KB *m*chunk, then look at the fingerprints of the next *N* chunks, where the *N*th chunk considered is the last chunk that, if included in the *m*chunk, would not exceed the maximum chunk size of 6KB.

At this point we have a choice among a few possible chunks at which to separate the current *m*chunk from the next one. We need a content-defined method to select which chunk to use as the breakpoint, similar to the method used for forming chunks in the first place within a size range. Here, we select the chunk with the *highest value fingerprint* as the breakpoint. Since fingerprints are uniformly distributed, and the same data will produce the same fingerprint, this technique produces consistent results (with sizes and deduplication comparable to chunking the original data), as we discuss next. We experimented with several alternative selection methods with similar results.

5.1.3 Evaluation

A key issue in this process is evaluating the error introduced by the constraints imposed by the merging process. We performed two sets of experiments on the sub-chunking and merging. The first was done on full-content datasets, to allow us to quantify the difference between ground truth and reconstructed metadata snapshots. We used two of the datasets from an earlier deduplication study [8], approximately 5TB each, to compute the "ground truth" deduplication and average chunk sizes. We compare these to the deduplication rate and average when merging chunks. (The datasets were labeled "workstations" and "email" in the previous study, but the overall deduplication rates are reported slightly differently because here we include additional overhead for metadata; despite the similar naming, these datasets should not be confused with the collected snapshots in Table 1.) Table 2 shows these results: the average chunk size from merging is consistently about 2–3% lower. For the workstations dataset, the deduplication rate is slightly higher, presumably due to smaller deduplication units, while for the email dataset the deduplication rate is somewhat lower (by 4–7%) with merging than when the dataset is chunked at a given size. However, the numbers are all close enough to serve as approximations to natural chunking.

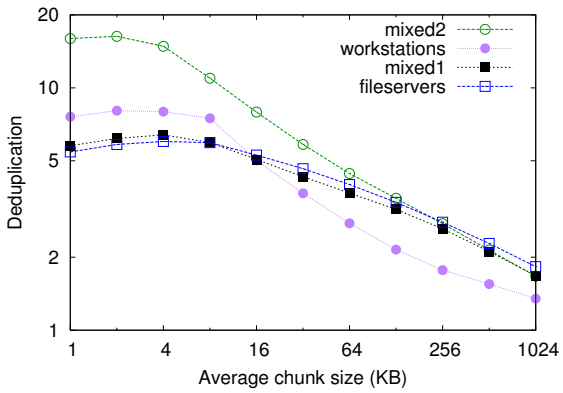


Figure 11: Deduplication obtained as a function of chunk size, using the merging technique. Both axes are on a log scale.

The second set of experiments, shown in Figure 11, was performed on a subset of the collected datasets (we selected a few for clarity and because the trends are so similar). For these we have no "ground truth" other than statistics for the original 8K chunks, but we report the deduplication rates as a function of chunk size as the size ranges from 1K sub-chunks to 1024KB (1MB) *m*chunks. The 1KB sub-chunks are used to merge into 2-4KB *m*chunks and the 8KB original chunks are used for the larger ones.

Looking at both the "ground truth" datasets and the snapshot analyses, we see that deduplication decreases as the chunk size increases, a result consistent with many similar studies. For most of the datasets this is an improvement of 20–40% for each reduction of a power of two, though there is some variability. As mentioned in §5.1.1, there is also a significant metadata overhead for managing smaller chunks. In Figure 11, we see that deduplication is consistently worse with the smallest chunks (1KB) than with 2KB chunks, due to these overheads: at that size the metadata overhead typically reduces deduplication by 10–20%, and in one case nearly a factor of two. Large chunk sizes also degrade deduplication; in fact, the `database1` dataset (not plotted) gets no deduplication at all for large chunks. Excluding metadata costs, the datasets in Table 2 would improve deduplication by about 50% when going from 2KB to 1KB average chunk size, but when these costs are included the improvement is closer to 10%; this is because for those datasets, the improvement in deduplication sufficiently compensates for the added per-chunk metadata.

Target Size (KB)	Workstations-full				Email-full			
	Ground Truth		Merged		Ground Truth		Merged	
	Dedup.	Avg Size (KB)	Dedup.	Avg Size (KB)	Dedup.	Avg Size (KB)	Dedup.	Avg Size (KB)
1	10.48	1.04	*N/A*		10.88	1.05	*N/A*	
2	9.29	2.08	9.30	2.02	9.54	2.09	9.19	2.03
4	7.28	4.15	7.28	4.09	7.84	4.18	7.48	4.08
8	5.49	8.34	*N/A*		6.61	8.33	*N/A*	
16	4.16	16.68	4.17	16.18	5.37	16.64	5.08	16.19
32	3.23	33.33	3.24	32.69	4.18	33.41	3.90	32.66
64	2.59	66.67	2.64	65.62	3.21	66.43	3.03	65.52

Table 2: Comparison of the ground truth deduplication rates and chunk sizes, compared to merging sub-chunks (to 4KB) or chunks (above 8KB), on two full-content snapshots. The target sizes refer to the *desired* average chunk size. The ground truth values for 1KB and 8KB average chunk sizes are included for completeness.

Data and analysis such as this can be useful for assessing the appropriate unit of deduplication when variable chunk sizes are supported [4, 17].

5.2 Cache Performance Analysis

In deduplicating systems, the performance bottleneck is often the lookup for duplicate chunks. Systems with hundreds of terabytes of data will have tens of billions of chunks. With each chunk requiring about 30 bytes of metadata overhead, the full index will be many hundreds of gigabytes. On today's systems, indexes of this size will not fit in memory and thus require an on-disk index, which has high access latency [34].

Effective caching techniques are necessary to alleviate this index lookup bottleneck, and indeed there have been numerous efforts at improving locality (e.g., Data Domain's Segment-Informed Stream Locality [34], HP's sparse indexing [21], and others). These studies have indicated that leveraging stream locality in backup workloads can significantly improve write performance, but their analyses have been limited to a small number of workloads and a fixed cache size. Unlike previous studies, we analyze for both read and write workloads across a broader range of datasets and examine the sensitivity of cache performance to cache sizes and the unit of caching.

5.2.1 Caching Effectiveness for Writes

As seen in §4, writes are a predominant workload for backup storage. Achieving high write throughput requires avoiding expensive disk index lookups by having an effective chunk-hash cache. The simplest caching approach would be to use an LRU cache of chunk hashes. An LRU cache relies on duplicate chunks appearing within a data window that is smaller than the cache size. For backup workloads, duplicate chunks are typically found once per full backup, necessitating a cache sized as large as a full backup per client. This is prohibitively large.

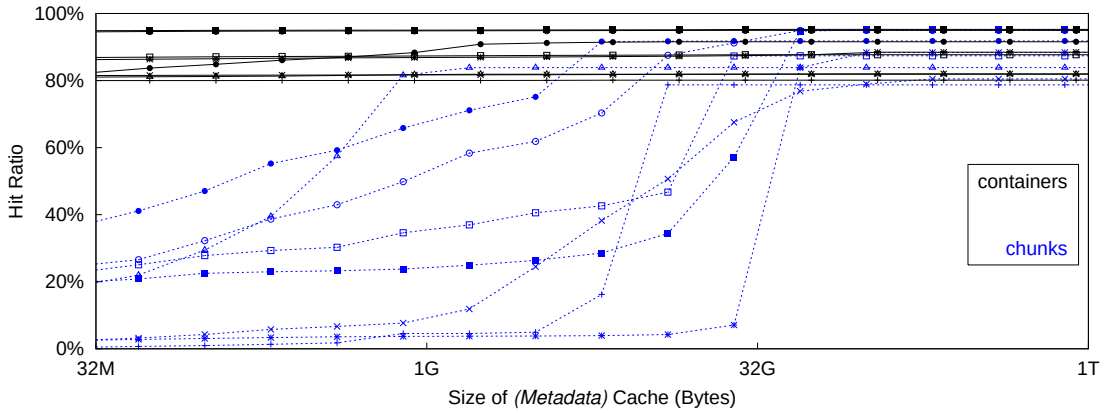
To improve caching efficiency, stream locality hints can be employed. [21, 34]. Files are typically grouped in a similar order for each backup, and re-ordering of intra-file content is rare. The consistent stream-ordering of content can be leveraged to load the hashes of nearby chunks whenever an index lookup occurs. One method of doing so is to pack post-deduplicated chunks from the same stream together into disk regions.

To investigate caching efficiency, we created a cache simulator to compare LRU versus using stream locality hints. The results for writing data are shown in Figure 12(a). The LRU simulator does per-chunk caching and its results are reported in the figure with the dotted blue lines. The stream locality caching groups chunks into 4MB regions called "containers" and its results are reported in that figure with solid black lines. We simulate various cache sizes from 32MB up to 1TB where the cache only holds chunk fingerprints (not the chunk data itself).[2] For these simulations, we replay starting with the beginning of the trace to warm the cache and then record statistics for the final interval representing approximately the most recent backup.

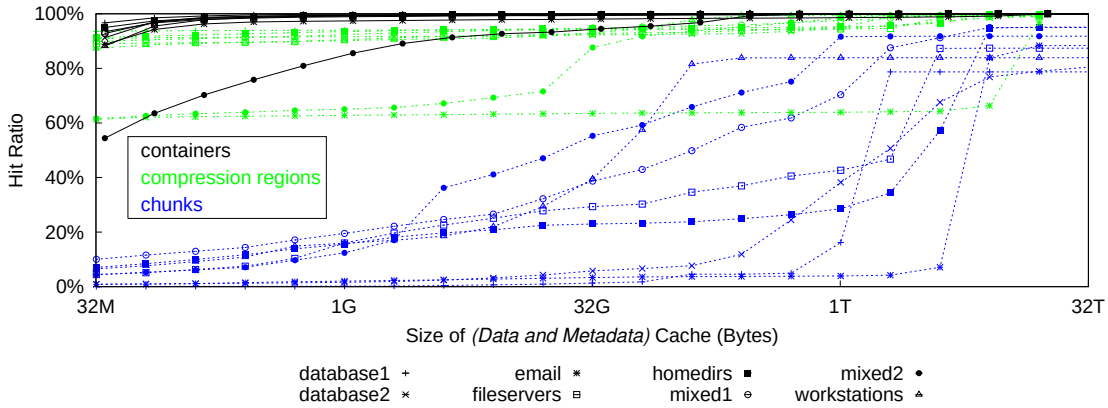
Note that deduplication write workloads have two types of compulsory misses, those when the chunk is in the system but not represented in the cache (duplicate chunks), and those for new chunks that are not in the system (unique chunks). This graph includes both types of compulsory misses. Because the misses for new chunks are included, the maximum hit ratio is the inverse of the deduplication ratio for that backup.

Using locality hints reduces the necessary cache size by up to 3 orders of magnitude. Notice that LRU does achieve some deduplication with a relatively small cache, i.e., 5-40% of duplicates could be identified with a 32MB

[2]To make the simulation tractable, we sampled 1 in 8 cache units, then scaled the memory requirement by the sampling rate. We validated this sampling against unsampled runs using smaller datasets. The cache size is a multiple of the cache unit for a type; therefore, data points of similar cache size do not align completely within Figure 12(a) and (b). We crop the results of Figure 12(a) at 32MB to align with Figure 12(b).



(a) Writes



(b) Reads

Figure 12: Cache results for writing or reading the final portion of each dataset. For writes, the cache consists just of metadata, while for reads it includes the full data as well and must be larger to have the same hit ratio. Differences in marks represent the datasets, while differences in color represent the granularity of caching (containers, chunks, or in the case of reads, compression regions).

cache (dotted blue lines). These duplicates which occur relatively close together in the logical stream may represent incremental backups that write smaller regions of changed data. However, effective caching is not typically achieved with the LRU cache until the cache size is many gigabytes in size, likely representing, at that point, a large portion of the unique chunks in the system. In contrast, using stream locality hints achieves good deduplication hit rates with caches down to 32MB in size (solid black lines across the top of the figure). Since production systems typically handle tens to hundreds of simultaneous write streams, each stream with its own cache, keeping the per-stream cache size in the range of megabytes of memory is important.

5.2.2 Caching Effectiveness for Reads

Read performance is also important in backup systems to provide fast restores of data during disaster recovery. In this subsection, we present a read caching analysis similar to that of the previous subsection.

There are three main differences between the read and write cache analysis. The first is that read caches contain the data whereas the write caches only needs the chunk fingerprints. The second is that reads have only one kind of compulsory miss, those due to cache misses, while writes can also miss due to the first appearance of a chunk. The third is that in addition to analyzing stream locality hints at the container level (which represents 4MB of chunks) we also analyze stream locality at the compression-region level, a 128KB grouping of chunks.

Figure 12(b) shows the comparison of LRU with stream locality hints at the container and compression-region granularity for read streams. The effectiveness of using stream locality hints is even more exaggerated here than for write workloads. Stream locality hints still

allow cache sizes of less than 32MB for container level caching (solid black lines), but chunk-level LRU (dotted blue lines) now requires up to several terabytes of cache (chunk data) to achieve effective hit rates. There is now a 4-6 order of magnitude difference in required cache sizes. Compression-region caching (dashed green lines) is as effective as container-level for 6 of the datasets, but 2 show significantly degraded hit ratios. These two datasets are from older systems which apparently have significant fragmentation at the compression-region level, which is smoothed out at the container level.

Fragmentation has two implications on performance. One is that data that appear consecutively in the logical stream can be dispersed physically on disk, impacting read performance [25]. Another is that the unit of transfer may not correspond to the unit of access; e.g., one may read a large unit such as a container just to access a small number of chunks. The impact of fragmentation on performance is the subject of recent and ongoing work (e.g., SORT [33]).

6 Conclusion

We have conducted a large-scale study of deduplicated backup storage systems to discern their main characteristics. The study looks both broadly at autosupport data from over 10,000 deployed systems and in depth at content metadata snapshots from a few representative systems. The broad study examines filesystem characteristics such as file sizes, ages and churn rates while the detailed study focuses on deduplication and caching effectiveness. We contrast these results with those of primary filesystems from Microsoft [22].

As can be seen from §4, backup filesystems tend to have fewer, larger and shorter-lived files. Backups typically comprise either large repositories, such as databases, or large concatenations of protected files (e.g., tarfiles). As backup systems ingest these primary data stores on a repeating schedule they must delete and clean an equal amount of older data to maintain within capacity limits. This high data churn, averaging 21% of total storage per week leads to some unique demands of backup storage. They must sustain high write throughput and scale as primary capacity grows. This is not a trivial task as primary capacity scales with Kryder's law (about 100x per decade) but disk, network, and interconnect throughput have not scaled nearly as quickly [13]. To keep up with such workloads requires data reduction techniques, with deduplication being an important component of any data protection system. Additional techniques for reducing the ingest to a backup system, such as change-block tracking, are also important as systems scale further.

Backup workloads have two properties that help meet these challenging throughput demands. One is that the data is highly redundant between full backups. The other is that the data exhibits a lot of stream locality; that is, neighboring chunks of data tend to remain nearby across backups [34]. As seen in §5.2, leveraging these two qualities allows for very efficient caching, with deduplication hit rates of about 90% (including compulsory misses from new chunks).

Another interesting point is that backup storage workloads typically have higher demands for writing than reading. Primary storage workloads, which have less churn and longer-lived data, are skewed to relatively more read than write workload (2:1 as a typical metric [20]). However backup storage must be able to efficiently support read workloads, as well, to process efficient restores when needed and to replicate data offsite for disaster recovery. Optimizing for reads requires a more sequential disk layout and can be at odds with high deduplication rates, but effective backup systems must balance between both demands, which is an interesting area of future work.

The shift from tape-based backup to disk-based, purpose-built backup appliances has been swift and continues at a rate of almost 80% annually. By 2015 it is projected that disk-based deduplicating appliances will protect over 8EB of data [16]. Scaling write throughput at the same rate as data is growing, optimizing data layout, and providing efficiencies in capacity usage are challenging and exciting problems. The workload characterizations presented in this paper are a first step at better understanding a vital, unique, and under-served area in file systems research and we hope that it will stimulate further exploration.

Acknowledgments

We thank Stephen Manley, Hyong Shim, and Florin Sultan for helpful comments on earlier drafts. We are especially grateful to Bill Bolosky for sharing the statistical spreadsheet from the Microsoft FAST 2011 study, and to the anonymous referees and our shepherd, Florentina Popovici, for their feedback and guidance. We very much appreciate the many people within and outside EMC who enabled the metadata collection, particularly Hugo Patterson and Edgard Capdevielle.

References

[1] N. Agrawal, W. Bolosky, J. Douceur, and J. Lorch. A five-year study of file-system metadata. In *FAST'07: Proceedings of 5th Conference on File and Storage Technologies*, pages 31–45, February 2007.

[2] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout. Measurements of a distributed file system. In *Proceedings of the Thirteenth Symposium on Operating Systems Principles*, Oct. 1991.

[3] J. Bennett, M. Bauer, and D. Kinchlea. Characteristics of files in NFS environments. In *SIGSMALL'91: Proceedings of 1991 Symposium on Small Systems*, June 1991.

[4] D. R. Bobbarjung, S. Jagannathan, and C. Dubnicki. Improving duplicate elimination in storage systems. *Transactions on Storage*, 2:424–448, November 2006.

[5] W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur. Single instance storage in Windows 2000. In *Proceedings of the 4th conference on USENIX Windows Systems Symposium - Volume 4*, pages 2–2, Berkeley, CA, USA, 2000. USENIX Association.

[6] M. Chamness. Capacity forecasting in a backup storage environment. In *LISA'11: Proceedings of the 25th Large Installation System Administration Conference*, Dec. 2011.

[7] A. Chervenak, V. Vellanki, and Z. Kurmas. Protecting file systems: A survey of backup techniques. In *Joint NASA and IEEE Mass Storage Conference*, 1998.

[8] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane. Tradeoffs in scalable data routing for deduplication clusters. In *FAST'11: Proceedings of 9th Conference on File and Storage Technologies*, Feb. 2011.

[9] J. Douceur and W. Bolosky. A large scale study of file-system contents. In *SIGMETRICS'99: Proceedings of 1999 Conference on Measurement and Modeling of Computer Systems*, May 1999.

[10] C. Dubnicki, G. Leszek, H. Lukasz, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki. HYDRAstor: a scalable secondary storage. In *FAST'09: Proceedings of the 7th conference on File and Storage Technologies*, pages 197–210, February 2009.

[11] EMC Corporation. Data Domain Boost Software, 2010. `http://www.datadomain.com/products/dd-boost.html`.

[12] EMC Corporation. Data Domain products. `http://www.datadomain.com/products/`, 2011.

[13] J. Gantz and D. Reinsel. Extracting value from chaos. IDC Iview, available at `http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf`, June 2011.

[14] F. Guo and P. Efstathopoulos. Building a high-performance deduplication system. In *Proceedings of the 2011 USENIX conference on USENIX Annual Technical Conference*, 2011.

[15] W. Hsu and A. J. Smith. Characteristics of I/O traffic in personal computer and server workloads. *IBM Systems Journal*, 42:347–372, April 2003.

[16] IDC. Worldwide purpose-built backup appliance 2011-2015 forecast and 2010 vendor shares, 2011.

[17] E. Kruus, C. Ungureanu, and C. Dubnicki. Bimodal content defined chunking for backup streams. In *FAST'10: Proceedings of the 8th Conference on File and Storage Technologies*, February 2010.

[18] P. Kulkarni, F. Douglis, J. LaVoie, and J. M. Tracey. Redundancy elimination within large collections of files. In *Proceedings of the USENIX Annual Technical Conference*, pages 59–72, 2004.

[19] D. A. Lelewer and D. S. Hirschberg. Data compression. *ACM Computing Surveys*, 19:261–296, 1987.

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

[21] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble. Sparse indexing: large scale, inline deduplication using sampling and locality. In *FAST'09: Proceedings of the 7th Conference on File and Storage Technologies*, pages 111–123, 2009.

[22] D. Meyer and W. Bolosky. A study of practical deduplication. In *FAST'11: Proceedings of 9th Conference on File and Storage Technologies*, February 2011.

[23] C. Min, K. Kim, H. Cho, S.-W. Lee, and Y. I. Eom. SFS: Random write considered harmful in solid state drives. In *FAST'12: Proceedings of the 10th Conference on File and Storage Technologies*, 2012.

[24] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *SOSP'01: Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 174–187, 2001.

[25] Y. Nam, G. Lu, N. Park, W. Xiao, and D. H. C. Du. Chunk fragmentation level: An effective indicator for read performance degradation in deduplication

storage. In *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, HPCC'11, pages 581–586, Washington, DC, USA, 2011. IEEE Computer Society.

[26] J. Ousterhout, H. DaCosta, D. Harrison, J. Kuntze, M. Kupfer, and J. G. Thompson. A trace driven analysis of the unix 4.2 BSD file system. *Proceedings of the Tenth Symposium on Operating Systems Principles*, Oct. 1985.

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

[28] C. Policroniades and I. Pratt. Alternatives for detecting redundancy in storage systems data. In *Proceedings of the USENIX Annual Technical Conference*, pages 73–86, 2004.

[29] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *FAST'02: Proceedings of the 1st USENIX conference on File and Storage Technologies*, 2002.

[30] D. Roselli, J. Lorch, and T. Anderson. A comparision of file system workloads. In *Proceedings of 2000 USENIX Annual Technical Conference*, June 2000.

[31] M. Satyanarayanan. A study of file sizes and functional life-times. In *SOSP'81: Proceedings of 8th ACM Symposium on Operating Systems Principles*, December 1981.

[32] P. Shilane, M. Huang, G. Wallace, and W. Hsu. WAN optimized replication of backup datasets using stream-informed delta compression. In *FAST'12: Proceedings of the 10th Conference on File and Storage Technologies*, 2012.

[33] Y. Tan, D. Feng, F. Huang, and Z. Yan. SORT: A similarity-ownership based routing scheme to improve data read performance for deduplication clusters. *IJACT*, 3(9):270–277, 2011.

[34] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the Data Domain deduplication file system. In *FAST'08: Proceedings of the 6th Conference on File and Storage Technologies*, pages 269–282, February 2008.