Generating Realistic Datasets for Deduplication Analysis

Vasily Tarasov[1]$(student, presenter)$, Amar Mudrankit[1]$(student)$, Will Buik[2]$(student)$
Philip Shilane[3], Geoff Kuenning[2], and Erez Zadok[1]
[1]*Stony Brook University,* [2]*Harvey Mudd College, and* [3]*EMC Corporation*

The amount of data that enterprises need to store increases faster than storage prices decrease, causing businesses to spend ever greater funds on storage [3]. One way to reduce storage costs is deduplication, in which repeated data is detected and replaced by references to a unique copy. Deduplication has been shown to be very effective in cases where the data is highly redundant [7, 8, 11]. For example, typical backup data contains multiple copies of the same files captured at different times, resulting in deduplication ratios as high as 95% [6]. Virtualized environments often run multiple similar virtual machines, which produces high duplication rates [7]. Even in primary storage, where one might expect data to be more unique, deduplication can be highly effective [9]. In fact, distinct users often share similar data, such as common project files or MP3 recordings of popular songs.

The significant space savings offered by deduplication have made it an almost mandatory part of the modern enterprise storage stack [4, 10]. Though the general concept of deduplication is the same in every product, there are many differences in how it is implemented and which optimizations are applied. Because of this variety and the high number of recently published papers in the area, it is important to be able to accurately compare the performance of deduplication systems.

The standard approach to deduplication is to divide the data into "chunks," calculate chunk hashes, and look up the result in an index to discover duplicates. The hashing step is straightforward; chunking is well understood but sensitive to various parameter settings. The indexing step is the most challenging because of the immense number of chunks found in real systems.

The chunking parameters and indexing method lead to three primary evaluation criteria for deduplication systems: (1) space savings, (2) performance (e.g., throughput and latency), and (3) resource usage (disk and CPU utilization, memory footprint, etc.). All three metrics are affected by the selection of data used for the evaluation and the specifics of the hardware configuration. Whereas previous storage systems could be evaluated based on traces of I/O operations, deduplication systems need the actual content (or a realistic recreation) to exercise caching and index structures.

We surveyed 120 datasets used in deduplication studies published in USENIX FAST, ATC, and several other conferences. We found that 76% of the datasets were not usable for cross-system evaluation, mostly because

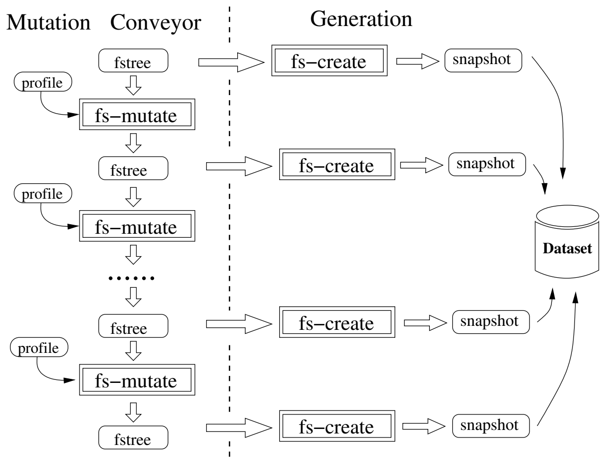

Figure 1: Mutation conveyor used to generate a dataset of several snapshots.

the data was either private or non-reproducible. Another 16% of the datasets were smaller than 1GB, which is too small to stress a full-fledged deduplication system. All remaining datasets (8%) contained various operating system distributions in different formats: installed, ISO, or VM images. However, such data cannot be considered as representative for the general user population. As a result, neither academia nor industry have wide access to representative datasets that can be used for unbiased comparison of deduplication systems.

We are developing a framework for *controllable data generation* that is suitable for evaluating deduplication systems. The key insight that we exploit is that deduplication is typically applied to multiple similar but slightly modified *snapshots*. (In this discussion "snapshot" is a generic term.) Snapshots can represent periodic backups; initially identical Virtual Machine disk images that become different because of their different usage; etc. If one can realistically emulate changes in the snapshots, then the whole dataset can be easily generated.

Our dataset generator operates at the file-system level, which is a common denominator across almost all deduplication systems. In fact, even block- and network-level deduplication solutions usually process data that is a file system at a higher level. We maintain an in-memory *fstree* object that represents file system tree along with the files' content. We do not store whole content, however, but only chunk's hashes. The generator uses an existing file system to create a first fstree object. It then *mutates* the fstree in accordance with a *mutation profile*.

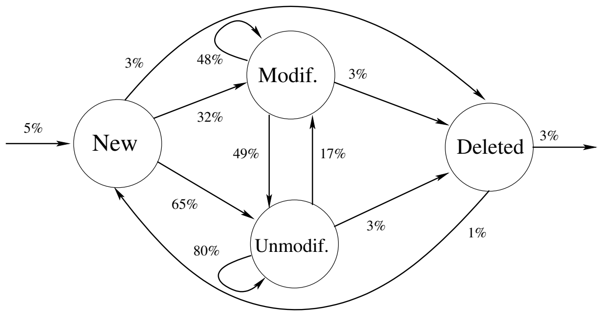

Figure 2: Markov model for handling file modifications. The probabilities of transitions are based on the Kernels 2.6.0–2.6.39 dataset.

A mutation *conveyor* can be created as depicted in Figure 1 to emulate long-term file system evolution. Each fstree object servers as an input to the next mutation and to the *fs-create* tool, which generates an on-disk snapshot. The important property that fs-create preserves is that the content of every generated chunk corresponds uniquely to that chunk's hash. This allows preserving the distribution of duplicates in the created file system.

Mutation should emulate both data and meta-data changes, because real deduplication systems are sensitive to both. E.g., it is known that the run-lengths of unique or duplicated chunks affect the throughput of many deduplication systems. Run lengths in turn depend on the ways files are modified and file size distribution.

Our current fs-mutate tool uses a Markov model to emulate file-level changes and multi-dimensional statistical distributions for in-file changes. Figure 2 shows a Markov model with probabilities for the Linux kernel versions 2.6.0–2.6.39. New files are created with 5% probability; existing ones stay unmodified from one snapshots to another with 80% probability. Only 17% of the files are modified if they were not changed in the previous snapshots. Using these probabilities, we can accurately select files for creation, deletion and modification. To predict the sizes of created files, the distribution of duplicates within them, the type of modification, etc., we use a multi-dimensional statistical distribution.

To create profiles, we analyzed both data and meta-data changes in several public and private datasets: home directories, system logs, Mac OS X-based email and Web servers, and a version control repository. We are currently experimenting with our model and various datasets to identify its accuracy range. Preliminary numbers show that the error of emulated parameters is within 15%. We believe our generator can become a useful tool for fair and versatile comparison of deduplication systems.

To the best of our knowledge there has been no research focusing on emulating file system content and meta-data evolution. File system benchmarks usually pre-create a file system to perform operations on it. Fstress [2] and Filebench [5] have the ability to specify file size and directory depth distributions for the pre-creation phase, but the data written to the files is either all zeros or random. Agrawal et al. presented a more detailed attempt to approximate the distributions encountered in the real-world file systems [1], but again, no attention was given in their study to generating duplicated content.

References

[1] Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Generating realistic impressions for file-system benchmarking. In *FAST '09: Proccedings of the 7th conference on File and storage technologies*, pages 125–138, Berkeley, CA, USA, 2009. USENIX Association.

[2] D. Andrerson. Fstress: A flexible network file service benchmark. Technical Report TR-2001-2002, Duke University, May 2002.

[3] R. E. Bohn and J. E. Short. How much information? 2009 report on american consumers. `http://hmi.ucsd.edu/pdf/HMI_2009_ConsumerReport_Dec9_2009.pdf`, December 2009.

[4] EMC Corporation. EMC Centra: Content Addressed Storage Systems. Product description guide, 2004.

[5] Filebench, July 2011. `http://filebench.sourceforge.net`.

[6] Advanced Storage Products Group. Identifying the Hidden Risk of Data Deduplication: How the HYDRAstorTM Solution Proactively Solves the Problem. Technical Report WP103-3_0709, NEC Corporation of America, 2009.

[7] K. Jin and E. Miller. The Effectiveness of Deduplication on Virtual Machine Disk Images. In *Proceedings of the SYSTOR Conference*, 2009.

[8] D. Meister and A. Brinkmann. Multi-Level Comparison of Data Deduplication in a Backup Scenario. In *Proceedings of the SYSTOR Conference*, 2009.

[9] D. Meyer and W. Bolosky. A Study of Practical Deduplication. In *Proceedings of the FAST Conference*, 2011.

[10] NetApp. NetApp Deduplication for FAS. Deployment and Implementation, 4th Revision. Technical Report TR-3505, NetApp, 2008.

[11] N. Park and D. Lilja. Characterizing Datasets for Data Deduplication in Backup Applications. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, 2010.