Comput Syst Sci & Eng (2013) 1: 73–83

International Journal of
Computer Systems
Science & Engineering

Towards an accurate evaluation of deduplicated storage systems

João Paulo, Pedro Reis, José Pereira and António Sousa

High-Assurance Software Lab (HASLab), INESC TEC and University of Minho
E-mail: {jtpaulo, jop, als}@di.uminho.pt, pedro.reiz@gmail.com

Deduplication has proven to be a valuable technique for eliminating duplicate data in backup and archival systems and is now being applied to new storage environments with distinct requirements and performance trade-offs. Namely, deduplication system are now targeting large-scale cloud computing storage infrastructures holding unprecedented data volumes with a significant share of duplicate content.
It is however hard to assess the usefulness of deduplication in particular settings and what techniques provide the best results. In fact, existing disk I/O benchmarks follow simplistic approaches for generating data content leading to unrealistic amounts of duplicates that do not evaluate deduplication systems accurately. Moreover, deduplication systems are now targeting heterogeneous storage environments, with specific duplication ratios, that benchmarks must also simulate.
We address these issues with DEDISbench, a novel micro-benchmark for evaluating disk I/O performance of block based deduplication systems. As the main contribution, DEDISbench generates content by following realistic duplicate content distributions extracted from real datasets. Then, as a second contribution, we analyze and extract the duplicates found on three real storage systems, proving that DEDISbench can easily simulate several workloads. The usefulness of DEDISbench is shown by comparing it with Bonnie++ and IOzone open-source disk I/O micro-benchmarks on assessing two open-source deduplication systems, Opendedup and Lessfs, using Ext4 as a baseline. Our results lead to novel insight on the performance of these file systems.

Keywords: Deduplication, Storage, Benchmark, Cloud Computing.

1. INTRODUCTION

Deduplication is now accepted as an effective technique for eliminating duplicate data in backup and archival storage systems [17] and storage appliances [20], allowing not only to reduce the costs of storage infrastructures but also to have a positive performance impact throughout the storage management stack, namely, in cache efficiency and network bandwidth consumption [13, 12, 10]. Moreover, a new trend where deduplication is applied to large-scale cloud computing large-scale infrastructures is emerging. In fact, recent studies show that up to 80% of space can be reclaimed for virtual machines with general purpose data [7, 4] and up to 95% for system images in a typical cloud provider [8].

Deduplication in cloud computing primary storage infrastructures raises new challenges for I/O performance and data consistency that are not addressed in backup deduplication. Some cloud applications will access and modify stored data, which we refer as *active data*, in a frequent basis and with strict disk I/O latency requirements. In contrast, *backup data* is immutable and consequently, shared data will never be modified, not needing copy-on-write mechanisms to ensure that data is not rewritten while being shared by several entities, which would lead to data corruption. However, copy-on-write mechanisms impair the performance of disk I/O operations by including additional computation overhead. Yet another challenge arises when deduplication is deployed on a decentralized infrastructure and performed across remote nodes, requiring distributed metadata for indexing the stored content and finding duplicates. Most systems perform in-band, also know as in-line, deduplication where disk I/O write requests to the storage are intercepted and shared before actually being stored. This way, only non-duplicate data is actually stored, sparing storage space but including metadata look up operations inside the critical I/O path, thus increasing I/O latency.

Having in mind these performance challenges and the recent sudden growth of work in this area, it is necessary to have proper benchmarking tools, with realistic workloads, for evaluating disk I/O performance of deduplicated storage systems. In fact, previous work analyzed 120 datasets used in deduplication studies and concluded that these datasets cannot be used for comparing distinct deduplication systems [18]. Most disk I/O benchmarks do not use a realistic distribution for generating data patterns and, in most cases, the patterns written either have the same content, for each write operations, or have random patterns with no duplicates at all [5, 9, 3]. If the same content is written for each operation, the deduplication engine will be overloaded with aliasing operations, which will affect the overall performance. Moreover, if this content is rewritten frequently, the amount of copy-on-write operations will increase considerably the I/O operations latency. On the other hand, writing always random content will generate few duplicates and the deduplication systems will be evaluated under a minimal load. Note that generating an unrealistic content workload will not only affect the disk I/O performance but also the space savings, sharing throughput and resource usage of the deduplication system [18]. Finally, deduplication is applied to storage environments with distinct distributions of duplicates that can also impact the evaluation accuracy. For instance, distinct duplicate content distributions will have an impact in the reference management and copy-on-write operations that, as explained previously, do not affect significantly the performance of archival/backup deduplication but will affect the performance of primary storage deduplication. This way, the workloads must simulate accurately the duplicate content distributions at the storage environment being targeted.

Some benchmarks address the duplicates generation issue by defining a percentage of duplicate content over the written records [14] or the entropy of generated content [2]. However, these methods are only able to generate simplistic distributions that are not as realistic as desired, or present still preliminary work where several details and proper implementation and evaluation are still missing [18].

We present DEDISbench, a synthetic disk I/O micro-benchmark suited for block based deduplication systems that introduces the following contributions:

- Generation of realistic content distributions, specified as an input file, that can be extracted from real datasets with DEDISgen extraction utility.
- Introduction of an hotspot random distribution, based on TPC-C NURand function, that generates access hotspots for disk I/O operations [19, 16].
- I/O operations, for each test, can be performed at a stress load, *i.e.* the maximum load handled by the test machine, or at a nominal load, *i.e.* the throughput of I/O operations is limited to a certain threadshold [16].

Note that DEDISbench simulates low-level I/O operations and does not focus on generating realistic directory trees and files like other benchmarks [6, 2, 18, 3, 9]. Nevertheless, such benchmarks are also referred along this paper and compared with DEDISbench in terms of content generation and accesses patterns. DEDISbench is evaluated and compared directly with Bonnie++ and IOzone, the two open-source micro-benchmarks that most resemble DEDISbench in terms of the suite and aim of disk I/O tests.

Our previous work on DEDISbench [16] is extended by considering duplicate content distributions from three storage environments with distinct access patterns and requirements, namely:

- An *Archival storage* where most files have a write-once policy, with non-significant updates, but with sporadic data deletion.
- A *Personal Files storage* where some files are updated and deleted frequently and the I/O requests latency is expected to be lower than the one found in archival storages.
- A *High Performance storage* where most files are updated and deleted frequently and I/O latency is expected to be as minimal as possible.

This study shows that each storage environment has specific requirements and duplicate content distributions. These differences must be simulated accurately when evaluating deduplication systems aiming at distinct storages. Moreover, we also show that extending DEDISbench with new workloads is simple and only requires having access to the real dataset.

Section 2 describes DEDISbench design, implementation and features. Section 3 presents the duplicate content distributions found for three real storage environments and extends DEDISbench workloads. Section 4 compares the content distributions generated by DEDISbench, Bonnie++ and IOzone. Additionally, Opendedup [15] and LessFS [11] deduplication systems are evaluated with these three benchmarks and their performance is compared with Ext4, a file system without deduplication. Section 5 introduces relevant work and the main differences from DEDISbench while Section 6 concludes the paper and points DEDISbench main contributions.

2. DEDISBENCH

This section starts by presenting a global overview of DEDISbench design and implementation and then, the generation of realistic content and access pattern distributions are described in more detail.

2.1 Basic Design and Features

The basic design and features of DEDISbench resemble the ones found in Bonnie++ [5] and IOzone [14] that are two open-source synthetic micro-benchmarks widely used to evaluate disk I/O performance. DEDISbench is implemented in C and allows performing either read or write block disk I/O tests, where the block size is defined by the user. I/O operations can be executed concurrently by several processes with independent files, being the number of processes and the size of process files pre-defined by the users. Moreover, the benchmark can be configured to stop the evaluation when a certain amount of data has been written or when a pre-defined period of time has elapsed, which is not common in most I/O benchmarks. Yet another novel feature of DEDISbench is the possibility of performing I/O operations with different load intensities. In addition to a stress load where the

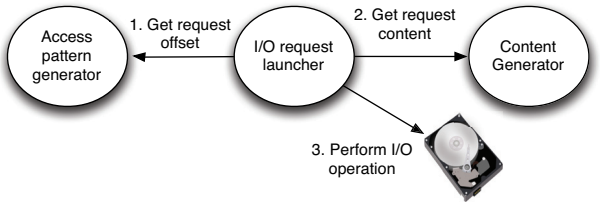

Figure 1 Overview of I/O request generation.

benchmark issues I/O operations as fast as possible to stress the system, DEDISbench supports performing the operations at a nominal load, specified by the user, thus evaluating the system with a stable load. Few I/O benchmarks support both features, as stated in Section 5.

Figure 1 depicts an overview of DEDISbench architecture. For each process, an independent I/O request launcher module launches either read or write I/O block operations, at nominal or peak rates, until the termination condition is reached. For each I/O operation, this module must contact the access pattern generator for obtaining the disk offset for the I/O operation (1) that will depend on the type of access pattern chosen by the user and can be sequential, random uniform or random with hotspots. Next, the I/O request launcher module contacts the content generator module for obtaining an identifier for the content to generate (2). Since DEDISbench is aimed at block-based deduplication, this identifier will then be appended as an unique pattern to the block's content, ensuring that blocks with different identifiers will not have the same content. The generated identifiers will follow the input file provided for DEDISbench with the information about duplicate content distribution. Note that this step is only necessary for write I/O requests because read requests do not generate any content to be written. Finally, the operation will be sent to the storage (3) and the metrics regarding operations throughput and latency will be monitored in the I/O request launcher module. Both content and access patterns generation are further detailed next.

2.2 I/O Accesses Distribution

DEDISbench can generate random uniform and sequential disk access patterns, as in IOzone and Bonnie++, and introduces a novel pattern based on TPC-C NURand function that simulates access hotspots, where few blocks are accessed frequently while the majority of blocks are accessed sporadically. This allows achieving a more realistic pattern, for most applications, where random disk arm movement is tested while leveraging the advantages of caching mechanisms, thus allowing to uncover distinct performance issues [16].

2.3 Duplicate Content Distribution

DEDISbench main contribution is the ability to process, as input, a file that specifies a distribution of duplicate content, which can be extracted from a real dataset, and using this information for generating a synthetic workload that follows such distribution. As depicted in Figure 2 the input file states the number of blocks for a certain amount of duplicates. In this example there are 5000 blocks with 0 duplicates, 500 blocks with 1 duplicate, 20 blocks with 5 duplicates and 2 blocks with 30 duplicates. This file can be populated by the users or can be generated automatically with DEDISgen, an analysis tool used for processing a real dataset and extracting from it the duplicate content distribution, as we detail in Section 3. DEDISbench then uses the input file for generating a cumulative distribution with the probability of choosing a certain block identifier, where two blocks with the same identifier are duplicates. Then with the aid of a random generator, a cumulative function is used for calculating, for each I/O operation, an identifier and consequently the content to be written.

Table 1 Duplicates found at the Archival, Personal Files and High Performance Storages.

		Archival	Personal Files	High Performance
Total space (GB)		486	113	1528
% Blocks w/ duplicates		90	76	69
% Duplicate blocks	distinct	3	6	6
	copies	7	18	25

3. EXTENDING DEDISBENCH DUPLICATE CONTENT DISTRIBUTIONS

There is extensive work focusing on the duplicates found at real storage systems [7, 4, 8]. However, the information provided in these studies does not present the necessary details in order to generate the input content distributions used by DEDISbench. This section details how DEDISbench can be extended with additional duplicate content distributions and presents the distributions found for three distinct real storage systems that are available in DEDISbench.

DEDISgen is an open-source utility for analyzing and extracting duplicate content distributions from real storage systems and generating suitable input distributions for DEDISbench [16]. Next, we analyze the distributions for three distinct storage environments extracted with DEDISgen for a block size of 4 KB. These three distributions have distinct access patterns and performance requirements that, as explained previously, limit the deduplication approach to be used.

3.1 Archival Storage

The first dataset analyzed has mainly archival and some backup data from the members of our research group. Most data is accessed sporadically and the number of updates on stored data is extremely low, thus not significant. However, data can be deleted which is not assumed in some archival systems. This way, this dataset's requirements are similar to the ones found in traditional archival/backup deduplication systems where write-once data is assumed and I/O throughput is leveraged over I/O latency [17].

As depicted in Table 1, the dataset has approximately 486 GB, 90% of the blocks do not have any duplicate, 3% of the blocks

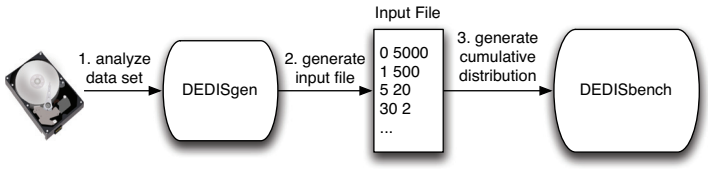

Figure 2 Generating and processing input content distribution file in DEDISbench.

have duplicates with distinct content and 7% of the blocks are copies that can be eliminated with deduplication. This storage has a small percentage of duplicate content, when compared to some of the literature [7, 8], because duplicated files at this storage are as reduced as possible due to space requirements.

3.2 Personal Files Storage

The second dataset has personal files from our research group and has distinct requirements from the archival dataset. Data is accessed frequently and some is updated and deleted sporadically, so it cannot be considered as write-once. Since the requirements of this dataset change, the deduplication systems targeting this storage type are also different from the ones considered in the last section [15, 11]. Namely, deduplication systems are expected to handle reference management more efficiently and protect updates on shared data with copy-on-write mechanisms that can have a significant impact in I/O latency, which is expected to be lower than the one tolerated in archival and backup storages.

As shown in Table 1, the dataset has approximately 113 GB and has a higher percentage of copies than the one found at the Archival storage, namely, 18%. Moreover, 76% of the blocks do not have any duplicate while 6% are duplicated blocks with distinct content. This storage has significantly more duplicates than the Archival storage because several personal files are duplicated. For example, project repositories are stored at this storage and in most cases each researcher has its own copy, generating several duplicates.

3.3 High Performance Storage

The last dataset analyzed is used as the primary storage for our group research projects, storing data from simulations and real applications. Data is updated frequently, thus generating more copy-on-write operations, that raise the complexity of reference management and the latency in I/O operations. However, I/O latency is expected to be as minimal as possible so, the deduplication systems suitable for the two storage types described previously are usually not suited for primary storages, leveraging the emergence of novel deduplication approaches.

As shown in Table 1, the dataset has approximately 1.5 TB, having the large size of the three storages analyzed and also the larger percentage of copies, namely, 25%. Additionally, 69% of the scanned blocks do not have any duplicate and 6% have duplicates with distinct content. This storage has the higher percentage of duplicates, which can be explained by the amount of duplicated runs from simulations and real systems benchmarks, necessary for the group's research work, that are kept persistent at this storage. Also some of these tests are not content-aware and, this way, simulate I/O storage content by writing only duplicate content, which also leverages the percentage of redundant copies found.

3.4 Discussion

Despite the distinct percentages of duplicates found for each storage environments, that will affect the number of duplicates generated and, consequently, the evaluation of deduplication systems, it is also important to look at the ratio of duplicates per block. In a storage scenario where blocks are updated frequently, having many blocks with one duplicate or having few blocks with many duplicates will change the number of copy-on-write operations, reference management and garbage collection. This way, it is important that the workload simulates not only the number of duplicates but also the duplication ratio found in storage systems.

The differences in the duplication ratios for the three storages are visible at Table 1. For example, both Personal Files and High Performance storages have 6% of duplicated blocks with distinct content but the percentage of duplicates (copies) is higher for the High Performance storage that, consequently, has a higher ratio of duplicates per block. In Figure 3, that shows the percentage of distinct content blocks with a certain range of duplicate blocks, these differences are even more noticeable. Note that the figure omits blocks with more that 500 duplicates for legibility reasons. In fact, all the storages have some blocks with more than 500 duplicates but the percentage is really small and not noticeable in the figure. In detail, Personal Files storage has 102 blocks with more than 500 duplicates, High Performance storage has 102 blocks with more than 500 duplicates and Archival storage has 118 blocks with 50 to 100 duplicates, 120 blocks with 100 to 500 duplicates and 58 blocks with more than 500 duplicates.

Looking at the figure, the Archival storage has few blocks with more than 100 duplicates while the Personal Files storage has the higher percentage of blocks with many duplicates. On the other hand, the High Performance storage has the higher percentage of blocks with few duplicates, which probably will increase copy-on-write operations and reference management complexity as many blocks will be shared and then updated in a frequent basis. This particular example helps understanding why deduplication systems must be evaluated with workloads that accurately simulate realistic content distributions.

To sum up, this study and comparison of duplicate content distributions is important to understand the percentage of duplicates and the ratio of duplicates per block for three storage en-

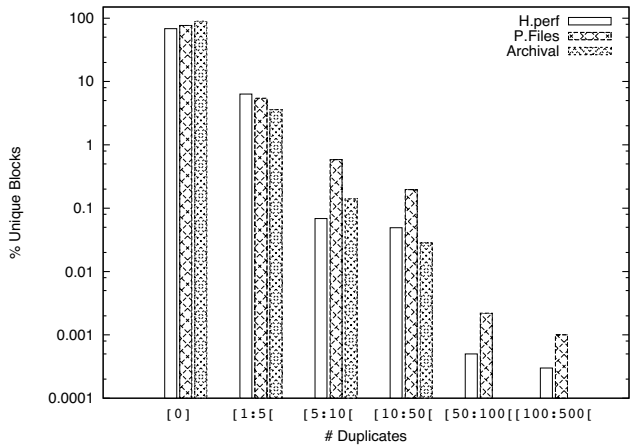

Figure 3 Distribution of duplicates ranges per distinct blocks for Archival, Personal Files and High Performance storages.

vironments with distinct access patterns and requirements. For instance, the different duplication ratios identified in this study can change the utilization patterns of reference management and copy-on-write mechanisms that may impact significantly the I/O latency overhead, which can pose as an unacceptable limitation for primary storage systems but still be tolerated in archival systems. Moreover, as we will show next, distinct duplicate content distributions will indeed affect the evaluation of deduplication systems so, it is extremely important to choose the correct workload.

4. EVALUATION

This section compares DEDISbench with IOzone and Bonnie++, which are the two micro-benchmarks with the closest design and features.

Bonnie++ [5] is a standard I/O micro-benchmark that performs several tests to evaluate disk I/O performance *in the following order*: Write tests assess the performance of single byte writes, block writes and rewrites while read tests assess byte and block reads, all with a sequential access distribution. Seek tests perform random block reads and, in 10% of the operations, block writes by following an uniform random distribution. The size of blocks, the number of concurrent Bonnie++ processes and the size of the file each process accesses are defined by the user. All these tests are performed with a stress load and run until an amount of data is written/read for each test. However, it is not possible to specify the content of written blocks. Bonnie++ also tests the creation and deletion of files, which is not contemplated in this evaluation because it is not supported by IOzone or DEDISbench.

IOzone [14] is the I/O micro-benchmark that most resembles DEDISbench and allows performing sequential and random uniform write and read tests. The block size, number of concurrent processes and the size of the files of each process, are also defined by the users. Tests are performed at a stress load and, for each test, the user defines the amount of data to be written by each process. Unlike in Bonnie++ it is possible to define full random tests that perform either read or write random disk I/O operations. Additionally, the percentage of duplicate inter-file and intra-file content in each block can also be specified. However, as discussed in the next sections, this content generation mechanism does not allow specifying a content distribution with a realistic level of detail as in DEDISbench.

DEDISbench, IOzone and Bonnie++ have several features in common but also differ in specific details, as shown in Table 2. In DEDISbench, a cumulative function extracted from a real dataset allows simulating not only the percentage of duplicate and non-duplicate blocks but also the distribution of duplicates per unique block. Such is not possible with the IOzone's approach that only allows defining the intra and inter-file duplication percentage or in Bonnie++ where the content distribution cannot be specified by the user. The content distributions generated by the three benchmarks are evaluated and further discussed in Section 4.2.

The three benchmarks support sequential and random uniform access tests. DEDISbench introduces a novel hotspot access pattern where few blocks are accessed frequently while the majority of blocks are accessed sporadically. I/O tests in DEDISbench can be pre-defined to terminate when a specific amount of data was written, as in IOzone and Bonnie++, or when a specific amount of time has elapsed. Moreover, DEDISbench tests can perform I/O operations with stress or nominal load intensities, allowing to evaluate storage systems with stress or stable loads. In Bonnie++, it is possible to perform block and byte I/O tests while, in the other two benchmarks, tests are only performed at the block granularity. Most of these details are also evaluated and discussed in Section 4.3 as they influence the results of the evaluation of deduplication systems.

4.1 Evaluation Scope and Setup

The experiments discussed in this section aim at validating two distinct points: First, we want to show that DEDISbench simulates more accurately a real content distribution than IOzone and Bonnie++. As a second goal, we want to show that this enhanced content generation mechanism along with the other novel features of DEDISbench allow uncovering new issues in deduplication systems, thus proving that such features are important for an accurate evaluation.

All tests ran in a 3.1 GHz Dual-Core Intel Core Processor with hyper-threading, 4GB of RAM and a local SATA disk with 7200 RPMs. Unless stated otherwise, for each benchmark test, the amount of data read/written was 8GB distributed over 4 concurrent processes, each reading/writing 2GB in an independent file.

The three benchmarks were configured in order to simulate as accurately as possible the content distribution of the Personal Files dataset described in Section 3. Namely, since the Personal Files dataset was analyzed with a block size of 4KB, the block size chosen for DEDISbench and Bonnie++ was also 4KB. DEDISbench used the input distribution file generated by DEDISgen to simulate the realistic distribution while, in Bonnie++, it is not possible to specify the content to be written.

On the other hand, for IOzone, the block size chosen was 16KB, thus defining that each block would have 25% of its data (4KB) duplicated across distinct process files. With this configuration and using 4 independent files, each block of 16KB

Table 2 Comparison of DEDISbench, IOzone and Bonnie++ features.

	DEDISbench	IOzone	Bonnie++
Content Generation	Cumulative function of a real distribution	Intra and Inter-file duplication percentage.	Cannot be specified
Access Patterns	Sequential, uniform random and hotspot random	Sequential and uniform random	Sequential and uniform random
Termination	Data written and time	Data written	Data written
Intensity	Stress and nominal	Stress	Stress
Granularity	Block	Block	Byte and block

has a distinct 4KB region with three duplicates, one for each file, which resembles the average number of 2.76 duplicates per block found at Personal Files workload. Globally, IOzone generates 18.5% of copies while the remaining 75% of the blocks do not have any duplicate, which also resembles the values shown at Table 1 for the Personal Files data. By choosing 16KB for the block size, the duplicate blocks are generated with a size of 4KB as in the real dataset, which would not be possible if the IOzone block size was defined as 4KB. Finally, even thus IOzone allows defining intra and inter-file duplicates, which would also allow to generate regions with many duplicates, this would lead to increasing greatly the block size and the complexity of the configuration to only be slightly closer to the distribution found at the real dataset. However, even with these modification this would only allow to simulate two or three distinct types of blocks with a distinct proportion of duplicates while in DEDISbench it is possible to simulate as many types as specified in the input distribution file, thus increasing hugely the distribution detail.

Note that we have chosen the Personal Files dataset for evaluating the benchmark's accuracy but we could have chosen any one of the other two datasets described in the previous section. As explained in Section 4.3, the evaluated deduplication systems, namely Opendedup [15] and Lessfs [11], were developed for storage workloads with requirements resembling our Personal Files one. This way, we choose to use this real distribution as a baseline for the complete evaluation procedure in order to increase the paper's uniformity and clarity. However, the results and consequent conclusions of Section 4.2 would have been similar if one of the other distributions was chosen and the benchmarks were configured to simulate them. Finally, thorough this section, we also refer to the Personal Files dataset as the real dataset.

4.2 Duplicate Content Distribution

In order to compare DEDISbench, IOzone and Bonnie++ content generation mechanisms we have used DEDISgen for analyzing the data generated by each benchmark for a sequential disk I/O write test. We choose the sequential I/O test over a random test because there are no block rewrites, enabling the extraction of precise information about all the written blocks and their contents. Note that, as explained previously, these benchmarks were all configured to simulate as accurately as possible the content of the Personal Files workload.

Figure 4 presents the percentage of unique blocks with a certain range of duplicates (*i.e.* equal to 0, between 1 and 5, 5 and 10, 10 and 50 and so on) for Bonnie++, IOzone and DEDISbench generated content and for the Personal Files dataset. All the distinct blocks generated by Bonnie++ have between 1 and 5 duplicates, in fact, each unique block has precisely 3 duplicates because every file is written with the exact same content, meaning that, all blocks in the same file are distinct but are duplicated among the other files. Consequently, as shown in Table 3, with Bonnie++ 75% of the written space can be deduplicated which may introduce a significant load in the deduplication mechanisms. Note that, Figure 4 shows the number of duplicates generated for each *unique block* written by the benchmarks while Table 3 shows the percentage of blocks without any duplicate, blocks with distinct duplicates and duplicate blocks for *all the blocks* written by the benchmarks, thus explaining why the percentages differ.

The results for IOzone in Figure 4 show that most unique blocks do not have any duplicate, while the remaining blocks have mainly between 1 and 5 duplicates and a very small percentage has between 5 and 10. In fact, the remaining distinct blocks should have 3 duplicates each, which happens for almost all the blocks with the exception of 216 blocks that have only 1 duplicate and 3 blocks that have 7 duplicates. If the content of unique blocks is generated randomly, it is possible to have these collisions, which are not significant for the 8GB written by the I/O tests. In Table 3, IOzone percentage of duplicates and unique blocks is closer to the real distribution percentages.

The results of DEDISbench, in Figure 4, show that the number of unique blocks and their duplicates is distributed over several regions, resembling most the real distribution. DEDISbench generates most blocks with few duplicates and a small percentage of blocks with many duplicates. In fact, we omitted one value from the figure in the far end of the distribution tail, for legibility reasons, where a single block has 15665 duplicates. As depicted in this figure, DEDISbench distribution is closer to a real dataset which may impact the performance of deduplication systems. For example, having many blocks with few duplicates will increase the number of shared blocks that, after being rewritten, must be collected by the garbage collection algorithm. On the other hand, mixing blocks with different number of duplicates will also affect the size of metadata structures and the work performed by the deduplication engine. However, when looking at Table 3 the results are slightly more distant from the real values when compared to IOzone results. Since the real data set has approximately 100GB and the benchmark is only writing 8GB, even if the cumulative distribution generated by DEDISbench has a high probability for writing some blocks several times, which would generate a large amount of duplicates per block, these blocks are being written fewer times than expected. Figure 5 and Table 4 compare the results of running DEDISbench sequential write tests for 16 and 32GB (divided by 4 files) and

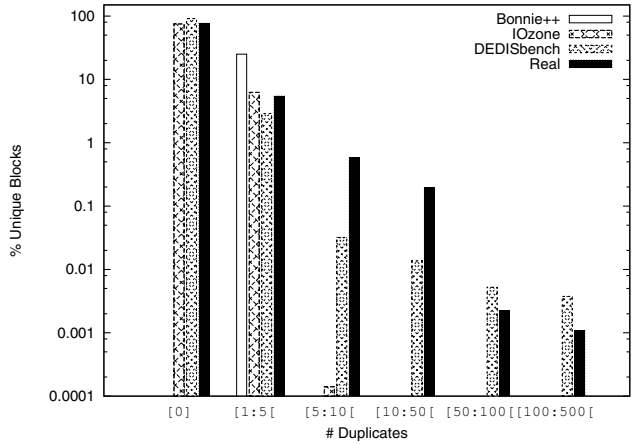

Figure 4 Distribution of duplicates ranges per distinct blocks for Bonnie++, IOzone, DEDISbench and the Real dataset.

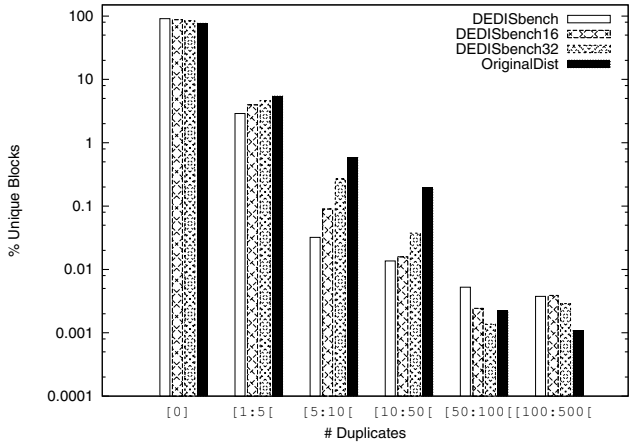

Figure 5 Distribution of duplicates ranges per distinct blocks for DEDISbench tests of 8,16 and 32 GB and for the Real dataset.

Table 3 Duplicates found in Bonnie++, IOzone,DEDISbench and the real dataset.

		Bonnie++	IOzone	DEDISbench	Real
% Blocks w/ duplicates		0	75	90	76
% Duplicate blocks	distinct	25	6	3	6
	copies	75	19	7	18

Table 4 Duplicates found in DEDISbench datasets with 8,16 and 32 GB and in the Real dataset.

		DEDISbench 8	DEDISbench 16	DEDISbench 32	Real
% Blocks w/ duplicates		90	87	83	76
% Duplicate blocks	distinct	3	4	5	6
	copies	7	8	12	18

shows that when the amount of written data is closest to the amount of data in the real dataset, the distribution generated by DEDISbench also becomes closer to the real one.

These results show that both Bonnie++ and IOzone do not simulate accurately the distribution of duplicates per unique blocks. On the other hand, DEDISbench design allows to overcome this limitation and simulates more accurately a real storage workload, thus proving our first evaluation goal. The lack of detail in Bonnie++ and IOzone can influence the load in the deduplication and garbage collection mechanisms of the deduplication system. For instance, a block shared by two entities or by one hundred determines the timing when garbage collection is needed, how often the copy-on-write mechanism must be used and the amount of information in metadata structures for sharing identical content. In Bonnie++ sharing an excessive amount of blocks will overload the deduplication engines while in IOzone, having all duplicated blocks with 3 or 4 distinct duplicates may also not be realistic and influence the evaluation. Such issues are assessed in the next section.

4.3 I/O Performance Evaluation

After comparing the content generation accuracy for the three benchmarks, we assessed how the distinct features of each benchmark would affect the evaluation of real deduplication systems. For this purpose, we chose to evaluate two open-source deduplication systems:

LessFS [11] is an open source single-host deduplication file system designed mainly for backup purposes but that can also be used for storing VM images and active data with moderate performance. LessFS uses FUSE for implementing file system semantics and only supports in-line deduplication. Data is stored as chunks with a fixed size of 4KB.

Opendedup [15] is an open-source deduplication file system that supports single and multi-host deduplication. Deduplication can be performed in-line or in batch (off-line) mode, where data is shared in an asynchronous fashion only after being stored. Opendedup file system is based on FUSE to export common file system primitives and uses Java for the deduplication engine. Data is stored as chunks and the block size can be parametrizable.

These two systems were chosen because they are mature open-source projects that, although not designed for high performance storages, export file systems supporting active data modification. In most archival/backup deduplication systems, write-once data is assumed and it is not possible testing the impact of copy-on-write and garbage collection mechanisms. Looking at the target storage workloads of these two systems and at the storage datasets discussed in Section 3, the Personal Files workload is the storage environment that fits best with the assumptions of these two deduplication systems. In fact, this was the main motivation for choosing the content distribution of this specific real dataset as the reference distribution simulated by the benchmarks in our experiments.

The impact of having distinct I/O benchmarks, with specific designs, evaluating deduplication systems cannot be measured by comparing directly the results of each benchmark. Firstly, each benchmark has distinct implementations for similar tests and for the calculation of I/O throughput and latency values as well as CPU and RAM usage. Moreover, each benchmark performs some tests that are not present in the others, for example, only Bonnie++ has single byte tests and only DEDISbench uses an hotspot access distribution. As another example, saying that Opendedup achieves higher throughput in Bonnie++'s sequential block write test than in IOzone and DEDISbench's sequential block tests may not be significant because each benchmark has distinct implementations.

Table 5 Evaluation of Ext4, LessFS and Opendedup with Bonnie++.

	Ext4	Lessfs	Opendedup
Sequential byte write (KB/s)	1100	76	56
Sequential block write (KB/s)	72035	13860	155496
Sequential block rewrite (KB/s)	17319	1016	62744
Sequential byte read (KB/s)	3029	1262	72
Sequential block read (KB/s)	73952	60064	144614
Urandom seek (KB/s)	170.9	127.1	115.8

This way, we choose to evaluate each deduplication system as it would be evaluated in a traditional scenario. Namely, the two deduplication systems were compared against Ext4, a file system without deduplication. By running the three benchmarks for Opendedup, LessFS and Ext4 we can observe the overhead of the deduplication systems over a traditional file system that does not perform deduplication. This way, it is possible to compare the amount of overhead introduced by the deduplication systems in Bonnie++, IOzone and DEDISbench tests and compare these overhead values. For instance, if the single byte write tests of Bonnie++ introduce higher I/O latency overhead than the sequential block write tests of all benchmarks, we could conclude that writing single bytes is inefficient and that this specific test uncovers a problem that is not evaluated by the other tests. On the other hand, if sequential block writes in Bonnie++ have less I/O latency overhead than in IOzone or in DEDISbench block write tests, we can compare these values and justify this difference because Bonnie++ writes a higher percentage of duplicates than the other two benchmarks. To sum up, by comparing the overheads of the deduplication systems and Ext4 it is possible to extract meaningful information even if the implementation of each benchmark is different. Note that this evaluation is not intended for assessing which benchmark consumes less resources or has higher throughput but to check how each deduplication system is evaluated by these benchmarks and which issues are uncovered.

All the file systems were mounted in the same partition, with a size of 20GB, that was formatted before running each benchmark. Also, all the deduplication file systems were configured to have a block size of 4KB and perform deduplication in-line. By performing in-line deduplication, data is shared immediately and the consequent overheads are also visible immediately, which would not be possible in batch mode deduplication. Finally, in Bonnie++ the tests were performed in the following order: single-byte write, block write and block rewrite in sequential mode, single-byte read and block read in sequential mode and, finally, the random seek test. For IOzone and DEDISbench we choose the tests order to be as similar as possible to Bonnie++. In IOzone the order was: Block write, block rewrite, block read and block reread in sequential mode and block read and block reread in random uniform mode. For DEDISbench the order was exactly the same as in IOzone but we introduced two more tests before ending the benchmark that were the block read and block write with the NURand distribution.

Table 5 shows the results of running Bonnie++ on Ext4, LessFS and Opendedup. By comparing the deduplication systems with Ext4 it is possible to conclude that writing sequentially one byte at a time is inefficient because, for each written byte, a block of 4KB will be modified and will be shared by the deduplication system, thus forcing the deduplication system to process a single block 4096 times. This is also true for sequential byte reads where, in each operation, it must be made an access to the metadata that tracks the stored blocks for retrieving a single byte. In this last test, the overhead introduced by Opendedup, when compared to LessFS overhead, is considerably higher and can be caused by retrieving the whole block to memory in each byte read operation instead of taking advantage of a caching mechanism. In sequential block write and rewrite Opendedup outperforms Ext4 by taking advantage of Bonnie++ writing the same content in all tests. Data written in sequential byte tests was already shared and Opendedup algorithm only requires consulting the in-memory metadata for finding duplicate content and sharing it, thus avoiding the need of actually writing the new blocks to disk. On the other hand, LessFS implementation does not seem to take advantage of such scenario. Opendedup also outperforms Ext4 in sequential block reads probably with a cache mechanism, efficient only at the block granularity. Finally, in random seek tests both deduplication systems present worse results that Ext4, with LessFS slightly outperforming Opendedup. RAM and CPU usages while Bonnie++ was running are depicted in Table 6. Both Opendedup and LessFS consume a significant amount of RAM, meaning that most metadata is loaded in memory and explaining, for example, the performance boosts of Opendedup in sequential block read and write tests. Moreover, the increase in CPU consumption with Opendedup can be a consequence of Bonnie++ writing a high percentage of duplicate content, thus generating an unrealistic amount of duplicate data to be processed.

Table 6 CPU and RAM consumption of LessFS and Opendedup for Bonnie++, IOzone and DEDISbench.

		LessFS	Opendedup
Bonnie++	CPU	22 %	163 %
	RAM	2.2 GB	1.8 GB
IOzone	CPU	9 %	25 %
	RAM	1.25 GB	2.1 GB
DEDISbench	CPU	15.7	19.5 %
	RAM	2.2 GB	1.9 GB

Table 7 Evaluation of Ext4, LessFS and Opendedup with IOzone.

	Ext4	Lessfs	Opendedup
Sequential block write (KB/s)	74463	5525,24	19760,8
Sequential block rewrite (KB/s)	74356,88	373,28	29924,84
Sequential block read (KB/s)	67159,36	7777,48	10464,4
Sequential block reread (KB/s)	67522,48	11495,48	10403,72
Urandom block read (KB/s)	2086,4	1304,08	1766,24
Urandom block write (KB/s)	2564,76	162,4	1608,04

Table 7 shows the results of running IOzone on Ext4, LessFS and Opendedup. Unlike Bonnie++, this benchmark does not write always the same content explaining why Opendedup does not outperforms Ext4 in block rewrite operations. Although some of the data was shared already, the content written is not always the same and most requests are still written to disk. With IOzone, Opendedup outperforms LessFS in almost all tests with

Table 8 Evaluation of Ext4, LessFS and Opendedup with DEDISbench.

	Ext4	Lessfs	Opendedup
Sequential block write (KB/s)	86916.82	5025.352	77508.424
Sequential block rewrite (KB/s)	76905.028	658.324	18852.732
Sequential block read (KB/s)	78648.964	7527.196	18591.672
Sequential block reread (KB/s)	78620.46	11788.792	20404.88
Urandom block read (KB/s)	791.356	2055.228	511.62
Urandom block write (KB/s)	1416.016	123,232	n.a.
NURandom block read (KB/s)	2287.208	1829.704	1350,304
NURandom block write (KB/s)	1246.336	151.556	n.a.

the exception of block reread test where LessFS is slightly better. LessFS decrease in performance is more visible in sequential and random write tests and mainly in rewrite tests. In Table 6, the RAM and CPU usages drop significantly which can be a consequence of writing less duplicate content. The RAM usage in Opendedup is an exception and the value is higher than in Bonnie++ tests.

Table 8 shows the results of running DEDISbench on Ext4, LessFS and Opendedup. As explained previously, IOzone generates all duplicated blocks with exactly 3 duplicates while DEDISbench uses a realistic distribution where most blocks have few duplicates but some blocks have a large number of copies, which will help explaining the next results. In sequential tests both Opendedup and LessFS are outperformed by Ext4, as in IOzone evaluation. However, the results of Opendedup for the sequential write test show considerably less overhead when compared to the same IOzone test, which can be a consequence of DEDISbench generating some blocks with a large amount of duplicates that will require writing only one copy to the storage, thus enhancing the performance of Opendedup. On the other hand, in the sequential rewrite tests, Opendedup performance decreases since DEDISbench generates many blocks with few duplicates that will then be rewritten and will require garbage collection, thus increasing the overhead.

The most interesting results appear in the random I/O tests. Firstly, LessFS outperforms Ext4 in uniform random block read test, which is an harsh test for the disk arm movement, pointing one of the advantages of using deduplication. If two blocks stored in distant disk positions are shared, the shared block will then point to the same disk offset and a disk arm movement will be spared. In IOzone there are few duplicates per block and this operations does not occur so often but, in DEDISbench some blocks have a large number of duplicates which can reduce significantly the disk arm movement and consequently improve performance. Even in Opendedup where this improvement is less visible, the overhead for random uniform read tests is lower than the one for sequential read tests. With the NURand hotspot distribution the performance of read operations in Ext4 is leveraged because caching mechanisms can be used more efficiently, thus the performance LessFS and Opendedup are reduced but, nevertheless, achieve better performance than in sequential tests. The CPU and RAM consumptions, shown in Table 6, for LessFS and Opendedup are similar to the ones obtained with IOzone, with a slight reduction in Opendedup and increase in LessFS. These

Table 9 Evaluation of Opendedup with DEDISbench and a modified version of DEDISbench that generates the same content for each written block.

	DEDISbench Original	DEDISbench Modified
Sequential block write (KB/s)	77508.424	247428,092
Sequential block rewrite (KB/s)	18852.732	253817,508
Sequential block read (KB/s)	18591.672	412694,064
Sequential block reread (KB/s)	20404.88	418169,436
Urandom block read (KB/s)	511.62	106696,336
Urandom block write (KB/s)	n.a.	3638,368
NURandom block read (KB/s)	1350,304	73385,616
NURandom block write (KB/s)	n.a.	3288,78

variations can be explained by the design and implementation of each deduplication system and how these process the distinct generated datasets.

The other interesting results are visible in the uniform and NURand random write tests. The performance of LessFS when compared to Ext4 decreases significantly while Opendedup system blocks with a CPU usage of almost 400%, not being able to complete these tests. Realistic content distribution in DEDISbench uncovered a problem in Opendedup that could not be detected with simplistic content distributions in IOzone and Bonnie++. To further prove this point, Table 9 tests Opendedup with the default DEDISbench and a modified version that writes always the same content, in each I/O operation, and, as we can see by the results, Opendedup completes successfully all the tests and greatly increases the performance, even when compared to the Ext4 results with the default DEDISbench version. However, the drawback of processing a fully duplicate dataset is visible in the CPU and RAM usage of Opendedup that increase to 272% and 2.6 GB respectively, which can be a serious limitation for deduplication in cloud commodity servers. Furthermore, these results show that using a realistic content distribution is necessary for a proper evaluation of deduplication systems and that Opendedup is not thought for datasets with a higher percentage of non-duplicated data.

This section states that using realistic content and accesses distributions influences significantly the evaluation of deduplication systems. Moreover, generating a realistic content distribution is necessary for finding performance issues and system design fails, like the ones found in Opendedup, but also for finding deduplication advantages, such as the boost in performance of uniform random read tests in LessFS. Moreover, it is useful having a benchmark that can simulate several content distributions ranging from fully duplicate to fully unique content and, most importantly, that is able to generate a content distribution where the number of duplicates per block is variable and follows a realistic distribution. To our knowledge, this is only achievable with DEDISbench.

5. RELATED WORK

Despite the extensive research on I/O benchmarking, to our knowledge and as discussed in previous published work, I/O

benchmarks that allow defining content distributions are vaguely addressed in the literature and are either limited to generating simplistic distributions [14, 6] or are still preliminary work [18].

A lot like DEDISbench does, IOzone [14] and Bonnie++ [5] test disk I/O performance by performing concurrent sequential and random read and write operations in several files. Bonnie++ does not allow specifying the content generated for I/O operations, in fact, it writes the same content in each disk I/O test and for each file. On the other hand, IOzone allows specifying the percentage of duplicate data in each record (block). It is possible to subdivide further this duplicate percentage and detail the amount of this percentage that is found among other records in the same file (intra-file), among records on distinct files (inter-file) and in both intra and inter file. In other words, these parameters allow defining the percentage of intra and inter-file duplicate content, meaning that, it is possible to achieve some control over the number of duplicates per record and have different regions of a record with a different number of duplicates as in DEDISbench. However, achieving such distributions can be complex and the level of detail achieved is not as realistic as the one provided by DEDISbench. Both IOzone and Bonnie++ use either sequential or random uniform distributions for the access pattern of I/O operations and are only able to perform stress testing. In Bonnie++ and IOzone, tests are performed at a peak/stress rate and random tests follow an uniform random distribution that balances equally the I/O operations per file region. DEDISbench introduces an hotspot access pattern distribution based on TPC-C NURand function and allows to perform I/O operations at a nominal throughput, that may be more realistic settings for most applications. To our knowledge, Bonnie++ and IOzone are the closest synthetic micro-benchmarks to DEDISbench in terms of design principles and evaluation parameters, which is why we compare our benchmark directly with both in Section 4.

Other work, with different assumptions from DEDISbench, IOzone and Bonnie++, leverages the simulation of actual file systems by generating directory threes and depth, the amount of files in each directory, distinct file sizes and multiple operations on files and directories. Most of these benchmarks use probabilistic distributions for building filesystem trees, choosing the operations to execute and the targets of each operation [3, 1, 9, 6, 18]. Fstress [3] presents several workloads with different distributions (*e.g.* peer-to-peer, mail and news servers) that run with a pre-defined nominal load like in DEDISbench. Moreover, Fstress also uses an hotspot probabilistic distribution for assigning operations to distinct files. Postmark [9] is designed to evaluate the performance of creating, appending, reading and deleting small files, simulating the characteristic workloads found in mail, news and web-based commerce servers. Target files and sizes are choosen by following an uniform distribution. Agrawal et all [1] work uses distinct probability models for creating new directories and files, for choosing the depth and number of files in each directory and for choosing the size and the access patterns to distinct files. However, none of these benchmarks allows specifying the content to be written, using instead a random or a constant pattern.

Filebench [6, 2] uses an entropy based approach for generating data with distinct content, for each I/O operation, that allows controlling the compression and duplication ratio of a dataset. Like in IOzone, this approach allows simulating the amount of duplicate data in a specific dataset but does not allow detailing further the distribution as in DEDISbench. Furthermore, we could not find the implementation details of this feature in the current version of Filebench. Tarasov *et al.* [18] preliminary work presents a framework for generating data content and metadata evolution in a controllable way. Their algorithm builds a base image with pre-defined directories and files and then uses a Markov model to perform file-level changes and multi-dimensional statistics to perform in-file changes that result in mutations of the base image. Metadata and data changes are loaded from pre-computed profiles, extracted from public and private data of different web servers, e-mail servers and version control repositories. This is still preliminary work and the generation and loading of the duplicate content distribution are neither detailed nor evaluated. Despite the different aims of DEDISbench and these filesystem benchmarks, our content generation algorithm is still different from the ones found in these systems and could be incorporated, with some design and implementation modifications, in any of these benchmarks.

To sum up, most I/O benchmarks do not support the generation of duplicate content writing either random or constant data patterns. To our knowledge, IOzone, Filebench and Tarasov et all [18] are the only I/O benchmarks supporting such feature but, when compared with DEDISbench, these benchmarks use different algorithms for generating duplicate content that are lacking design and implementation details or limiting the realism of generated distributions.

6. CONCLUSION

This paper discusses the characterization of duplicate content in storage systems and its impact in evaluating deduplication systems. This is achieved with DEDISbench, a synthetic disk I/O micro-benchmark that processes metadata extracted from real datasets for generating realistic content for I/O write operations. Previous I/O benchmarks, either do not focus on distinct content generation or generate limited distributions that, in most cases, do not simulate accurately real datasets. DEDISbench allows performing I/O tests with stress and nominal intensities and introduces an hotspot distribution, based on TPC-C NURand function, that allows testing random disk accesses while maintaining cache efficiency.

In detail, the duplicate content distributions of three real datasets, with distinct requirements and access patterns, are analyzed with DEDISgen, a tool for analyzing real datasets and extracting the duplicate content distributions used by DEDISbench. This process is straightforward and allows having workloads that are suited for distinct storage environments.

The comparison of DEDISbench with IOzone and Bonnie++ shows that DEDISbench simulates more accurately a real content distribution, allowing to specify in detail the proportion of duplicates per unique block. This increased accuracy was key for finding new performance advantages and drawbacks and also relevant issues in two deduplication file systems, LessFS and Opendedup, evaluated with the three benchmarks and compared to Ext4, a file system without deduplication. In fact, DEDISbench realistic content distribution uncovered an important limitation in Opendedup implementation.

7. AVAILABILITY

DEDISbench and DEDISgen documentation, source code and debian packages are available at: `http://www.holeycow.org/Home/dedisbench`.

Acknowledgments

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project RED FCOMP-01-0124-FEDER-010156 and FCT by Ph.D scholarship SFRH-BD-71372-2010.

REFERENCES

1. Agrawal, N., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Generating realistic impressions for file-system benchmarking. In: Conference on File and Storage Technologies (2009)
2. Al-Rfou, R., Patwardhan, N., Bhagavatula, P.: Deduplication and compression benchmarking in filebench. Tech. rep. (2010)
3. Anderson, D.: Fstress: A flexible network file service benchmark. Tech. rep. (2002)
4. Clements, A.T., Ahmad, I., Vilayannur, M., Li, J.: Decentralized deduplication in san cluster file systems. In: USENIX Annual Technical Conference (2009)
5. Coker, R.: Bonnie++ web page. `http://www.coker.com.au/bonnie++/`, may 2012
6. Filebench: Filebench web page. `http://filebench.sourceforge.net`, may 2012
7. Ganger, G.R., Wilkes, J.: A study of practical deduplication. In: Conference on File and Storage Technologies (2011)
8. White paper - complete storage and data protection architecture for vmware vsphere. Tech. rep. (2011), `http://www.ea-data.com/HP_StoreOnce.pdf`
9. Katcher, J.: Postmark: a new file system benchmark. Tech. rep. (1997)
10. Koller, R., Rangaswami, R.: I/o deduplication: utilizing content similarity to improve i/o performance. In: Conference on File and Storage Technologies (2010)
11. Lessfs: Lessfs web page. `http://www.lessfs.com/wordpress/`, may 2012
12. Muthitacharoen, A., Chen, B., Mazieres, D., Eres, D.M.: A low-bandwidth network file system. In: Symposium on Operating Systems Principles (2001)
13. Nath, P., Kozuch, M.A., Ohallaron, D.R., Harkes, J., Satyanarayanan, M., Tolia, N., Toups, M.: Design tradeoffs in applying content addressable storage to enterprise-scale systems based on virtual machines. In: USENIX Annual Technical Conference (2006)
14. Norcott, W.D.: Iozone web page. `http://www.iozone.org/`, may 2012
15. Opendedup: Opendedup web page. `http://opendedup.org`, may 2012
16. Paulo, J., Reis, P., Pereira, J., Sousa, A.: Dedisbench: A benchmark for deduplicated storage systems. In: In proceedings of 2nd International Symposium on Secure Virtual Infrastructures (DOA-SVI'12). Springer (2012)
17. Quinlan, S., Dorward, S.: Venti: A new approach to archival storage. In: Conference on File and Storage Technologies (2002)
18. Tarasov, V., Mudrankit, A., Buik, W., Shilane, P., Kuenning, G., Zadok, E.: Generating realistic datasets for deduplication analysis. In: USENIX Annual Technical Conference. Poster Session (2012)
19. Transaction processing performance council: TPC-C standard specification, revision 5.5. `http://www.tpc.org/tpcc/spec/tpcc_current.pdf`
20. Zhu, B., Li, K., Patterson, H.: Avoiding the disk bottleneck in the data domain deduplication file system. In: Conference on File and Storage Technologies (2008)