Survey on Deduplication Techniques in Flash-Based Storage

Ilya Chernov, Evgeny Ivashko, Alexander Rumiantsev, Vadim Ponomarev, Anton Shabaev
Institute of Applied Mathematical Research, Karelian Research Centre of RAS
Petrozavodsk State University
Petrozavodsk, Russia
{chernov,ivashko,ar0}@krc.karelia.ru, vadim@cs.karelia.ru, ashabaev@petrsu.ru

***Abstract*—Data deduplication importance is growing with the growth of data volumes. The domain of data deduplication is in active development. Recently it was influenced by appearance of Solid State Drive. This new type of disk has significant differences from random access memory and hard disk drives and is widely used now. In this paper we propose a novel taxonomy which reflects the main issues related to deduplication in Solid State Drive. We present a survey on deduplication techniques focusing on flash-based storage. We also describe several Open Source tools implementing data deduplication and briefly describe open research problems related to data deduplication in flash-based storage systems.**

I. Introduction

Volume of data is growing exponentially. The total amount of digital data worldwide is expected to grow from 16 ZB in 2016 to 163 ZB in 2025 [1]. At that, several researches (e.g., [2]) show that data redundancy in enterprise storages is rather high: from 50% up to 95%. Pieces of duplicate data are found on file as well as disk level, whereas both deduplication and compression technologies address this redundancy. During deduplication, a piece of data is checked, prior to writing, to have an identical copy already stored, and in positive case referencing the existing copy is used instead of writing a new one. This reduces the traffic, increases data density, and decreases energy consumption. Note that deduplication is a reciprocal of replication: the former technology eliminates redundancy while the latter exploits it it for the sake of safety, reliability, robustness, etc.

Deduplication is an emerging technology, accelerated by development of new storage media, and currently it gets increasing attention, thanks to the widely used Solid State Drives (SSD). In this regard, deduplication technology should address the specifics of an SSD, which include high random read speed (in contrast to performance degradation due to fragmentation on a conventional hard disk drives (HDD)), write amplification (related to the erase before write technique), memory wear-out (with limited number of rewrites before failure), page size reads and block size writes, and necessary garbage collection (the necessary details on SSD architecture and specifics may be found e.g. in [3]).

In this work we present a survey on deduplication and compression research works considering SSD specifics. As a result, we present an enhanced taxonomy, which is most relevant to the particular subfield of flash-based storage systems. We also briefly describe several data deduplication tools implemented in Linux and rise several open issues in the domain of a stand-alone device as well as an SSD equipped storage system.

The structure of the paper is as follows. Section II describes the general deduplication process. In Section III the paper selection and sources are characterized. Section IV contains analysis of general survey papers on data deduplication and presents a new taxonomy. In Sections V and VI the techniques proposed at single (sub)device level and at storage/network levels are described. A number of Open Source data deduplication tools implemented in Linux operating system are given in Section VII. In Section VIII we describe several important open issues promising for deduplication improvement. Finally, in Section IX several concluding remarks are given.

II. General deduplication process

Deduplication is aimed at reducing data redundancy. The idea behind it is to store only unique data pieces using references on existing data pieces when necessary. A general deduplication process consists of four stages (see Fig. 1):

1) *chunking* refers to granularity;
2) *fingerprinting* refers to unicity of data;
3) *searching* refers to exploring existing pieces of data;
4) *referencing* refers to linking pieces of data to files.

In general, a deduplication process starts with splitting the input data stream (e.g., a file) into data pieces called *chunks*. This stage can be omitted in the case of file-level deduplication. However, chunk-level deduplication shows better deduplication ratio [4]. Size of the chunks can be either fixed or dynamic (the latter provides higher deduplication ratio). In SSD-based storage systems the chunk size is usually set equal to the SSD page size (4 KB) to improve the write performance, garbage collection, and wear leveling. Still, there is a variety of chunk size choices considered by researches; to name a few:

- *a page* (for example, in [5] page-sized chunking is a base for a novel deduplicating Flash Translation Layer (FTL) scheme),
- *several pages* (for example, in [6] where different sizes of chunks are used to compare performance),
- *a block* (for example, in [7] where a new deduplication scheme called “block-level content-aware chunking” is proposed to extend the lifetime of SSDs).

ISSN 2305-7254

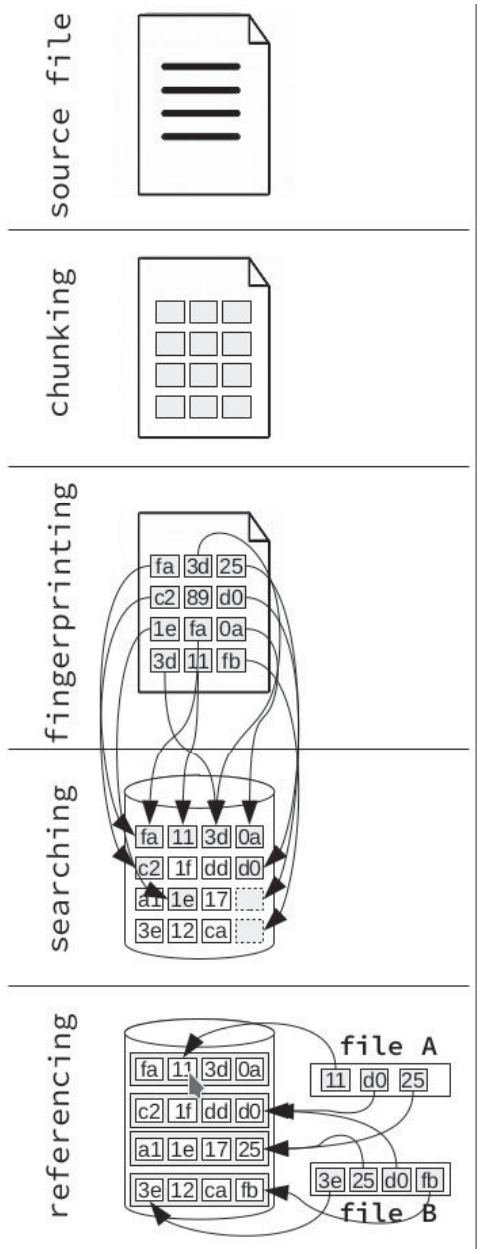

Fig. 1. General deduplication process

At the next stage a special algorithm produces *fingerprints* of data chunks. Usually fingerprinting is based on a hash-function like SHA-1, MD5, etc, but also other functions can be used for higher speed, lower resource (CPU cycles and memory) consumption, and uniform distribution of fingerprints.

Hereafter, searching is used to find out if the same data chunk already exists in the storage. It should be as fast as possible to provide better response on write operations.

If the current data chunk is unique, it is stored in the system; otherwise, just a reference to the existing data is saved. It is a referencing stage.

Data chunks are aggregated into containers to provide better read and garbage collection performance. A data stream is usually collected from different containers and distributed among the storage data chunks. Container approach is motivated by high random read speed of SSD. However, from the point of view of performance it is still important to reduce the number of container reads required to restore the file. For details on general deduplication process we refer the reader to [4].

Note that the deduplication process might be performed on various levels, from a sub-block level in the device up to storage level of the cloud. Moreover, since deduplication is mostly a computationally intensive process, the exact device that performs deduplication depends on a suggested technique. Deduplication might be performed, e.g.,

- in FTL (for example, in [5], [8] novel deduplicating FTL schemes are proposed),
- in a specific hardware block (for example, in [9] a special software defined network (SDN) controller is used to implement in-network deduplication),
- on host (the most common way, used, for example, in [10] to implement application-aware deduplication).

While some researches propose changes in the deduplication stages (like eliminating, introducing new, or combining any of stages) in order to improve deduplication techniques, the general deduplication process seems to be conventional by the moment.

III. Scientific papers on deduplication

We used the search engine of Web of Knowledge to harvest scientific papers of deduplication. The time span is 1995–2017, though stable interest originates in 2010 with a jump in 2014 (determined by growth of interest to SSD) and slight decrease in the recent years. Fig. 2 shows the dynamics of the number of publications, which are considered as the most relevant and included in this survey. One half of the papers surveyed here was published in journals and proceedings of IEEE, some papers in issuance of ACM. Publications are evenly distributed between proceedings of international symposia and conferences (60%) and scientific journals (40%).

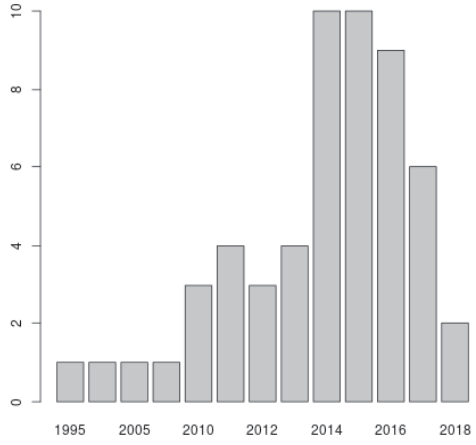

Fig. 2. Amount of publications for 2005–2017

Fig. 3 demonstrates wordcloud gathered from abstracts of

research papers on SSD-focused data redundancy management. We note some highlights easy following from the wordcloud:

- there clearly is strong connection among deduplication and compression techniques;
- redundancy management is among the primary aims of the deduplication and compression methods;
- write overhead, memory footprint, and lifetime are among the primary efficiency measures;
- read and write caching is a popular satellite technique, especially for large storage, such as cloud and backup devices;
- blocks and files are popular deduplication units.

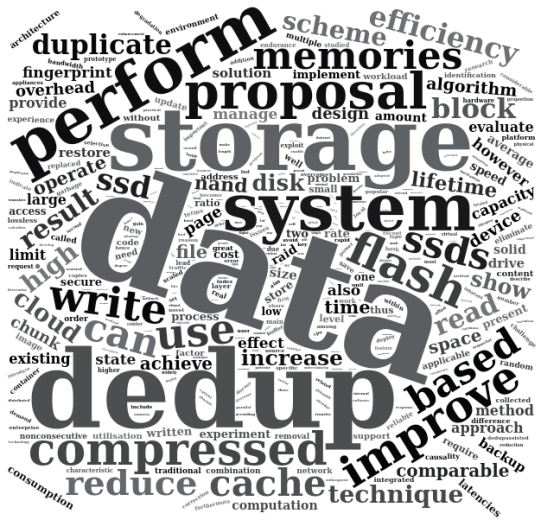

Fig. 3. Wordcloud of deduplication related keywords

IV. Surveys on deduplication

There is a number of previous research papers aimed at different aspects of deduplication. We briefly describe them in this section to highlight their differences with this work.

A. Surveys

Recently several survey papers performed an overview of the research on deduplication algorithms, including some SSD specifics. In [11] a new taxonomy on data deduplication is presented. This taxonomy is based on [12] and is further extended. The authors consider six categories: granularity, locality, timing, indexing, technique, and scope (these categories are expanded below); an overview of existing deduplication systems, grouped by storage type, is presented: Backup and Archival Storage, Primary Storage, Random Access Memory, and SSD.

In [4] deduplication is considered as evolution of data compression. The authors discuss the main applications and trends in data deduplication and list several publicly available Open Source projects, data sets, and traces to study deduplication; some open problems are described.

The paper [13] considers security-related issues of data deduplication in cloud storage systems. Basing on the proposed classification, the authors describe security risks and inside/outside attack scenarios. The modern techniques of safe deduplication are presented; to provide safety, these techniques use both crypto solutions and protocols. The authors highlight four approaches to design safe deduplication systems: data granularity, deduplication location, storage architecture, and cloud service model. The existing taxonomy is extended using these approaches. The existing solutions in secure deduplication are classified into four categories: message-dependent encryption, proof of ownership, traffic obfuscation, and deterministic information dispersal. Each of these categories is analyzed from the points of view of security and effectiveness, pros and contras are described. The authors also describe open problems in the domain of secure deduplication.

Comparing to these surveys on data deduplication, our research is more concentrated on flash-based storage. We propose a new taxonomy, review SSD-related research, and highlight several open problems in the domain of deduplication of flash-based storage.

B. Classification

There are several taxonomies on data deduplication. One of the most cited is the one presented in [12] and extended in [11]. The latter identifies six categories:

- *granularity*: refers to the method used for partitioning data into chunks,
- *locality*: locality assumptions are commonly exploited in storage systems to support caching strategies and on-disk layout,
- *timing*: refers to when detection and removal of duplicate data are performed, in particular, if duplicates are eliminated before or after being stored persistently,
- *indexing*: provides a data structure that supports the discovery of duplicate data,
- *technique*: refers to deduplication of either exact copies or “similar” data chunks,
- *scope*: refers to local or distributed deduplication.

The paper [13] proposes a specific taxonomy of secure deduplication solutions based on four categories: message-dependent encryption, proof of ownership, traffic obfuscation, and deterministic information dispersal. In [14] a taxonomy based on location, time, and chunk is developed. To address the specifics of SSD, we extended the taxonomy proposed in [11] (see Fig. 4).

The proposed taxonomy reflects the main issues related to deduplication in SSD.

C. Deduplication vs. compression

Both deduplication and compression are aimed at reducing data redundancy. In [4] deduplication is considered as evolution of compression. While traditional compression eliminates redundancy within a file or a small group of files, deduplication eliminates both intrafile and interfile redundancy over large

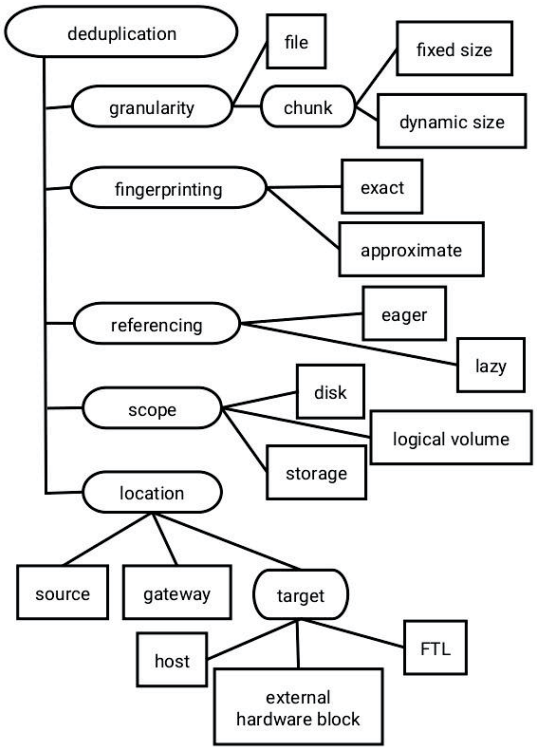

Fig. 4. Taxonomy of data deduplication

datasets, which could be stored by different users and across multiple distributed storages. Some works implement compression as an additional stage of deduplication (for example, see [5]).

We note that in general deduplication, as well as compression, is not always profitable. This means that the overhead caused by both techniques might be more significant, than the obtained gain. To address this important issue, we refer the reader to deduplication effectiveness and efficiency evaluation performed in the works [15], [16].

D. Goals of deduplication and compression

Several goals are specifically related to SSD as compared to general deduplication goals (such as memory footprint reduction):

- *Obtain extra tangible capacity* is the first goal of data deduplication considered in research works while SSD are still more expensive than HDD;
- *Reduce the number of write operations* [5], [6] is the second goal related to performance: the fewer write operations are performed, the lower response time with write operations the storage has;
- *Reduction of write amplification* [17] is a consequence of reducing the number of write operations: as write amplification heavily harms SSD shortening its lifetime and decreasing performance, reduction of write amplification is considered as one of the main features of data deduplication on SSD.
- *Improvement of write speed* [18] is an another consequence of reducing the number of write operations: as some write operations are replaced by less time-costly referencing operations, write speed increases;
- *Extension of the lifetime* [19] is the third consequence of reducing the number of write operations number and its corresponding write amplification;
- *Increase reliability* [20].

Specifics of SSD leads to new efficiency criteria compared to deduplication in HDD. This requires new technologies and methods of deduplication.

V. Single device level techniques

In the simplest case deduplication is narrowed to a single device. This approach is convenient to study the general deduplication process and its stages.

A. Deduplication and compression

It is known, that SLC (Single Level Cell) and MLC/TLC (Multi Level Cell/Three Level Cell) regimes of an SSD cell possess diverse reliability in terms of erase/write cycles. This diversity is utilized to perform write buffering to cache containing blocks working in SLC mode, thus extending the overall device lifetime. In order to perform wear leveraging, the SLC cache is rotated over all memory blocks. To enhance such a caching technology, an in-place delta compression technique was suggested in [21]. By performing intra-page delta compression (keeping the updates of the page contents in the same page by using the so-called partial programmability SLC mode), and lossless intra-sector data compression, the authors claim to decrease the write stress for the intra-SSD write cache working in SLC mode, at the price of a hardware sophistication. A somewhat similar idea of in-place update without block erase, based on the so-called multi-write coding (when a single page can be written more than once, without the need of intermediate block erasing) enhanced by the lossless data compression technique resulting in lesser write amplification and memory wear is discussed in [22].

In many papers the deduplication technique is suggested to increase lifetime rather than reclaim extra capacity. The reason for that is a possibility of deduplication cancellation due to data update. Thus, the free capacity is conventionally used for internal routines such as garbage collection and wear levering. A novel FTL scheme addressing the problem of obtaining extra free space by using a combination of deduplication and compression techniques was suggested in [5]. A prediction technique for compression rate applied to the so-called cold (rarely used) data is used to obtain the limits for available extra capacity, with respect to which the free space reclaimed by deduplication is offered to the operating system level. A similar compression rate based hot/cold data identification technique is addressed in the paper [23].

The reclaimed free space might also be used to increase redundancy. An in-block data compression method for reliability increase is suggested in the works [24], [20]. A block-level lossless compression (by a combination of a modified move-to-front algorithm with Huffman coding) allows to use the obtained in-block extra capacity to switch to a higher redundancy (by increasing the size of error correction codes (ECC) information stored in the block). A similar approach

is suggested in [18]. By applying in-block lossless compression and enhanced ECC, the authors suggest to apply faster writing, thus increasing the overall write speed without losing reliability. Prior to compression, a compressibility estimation is performed.

The works [17], [6] apply compression for increasing lifetime and performance, decreasing power consumption. Both papers suggest new FTL software supporting selective compression, whereas the paper [6] additionally requires a specific hardware module for high online compression performance. The content-aware in-block deduplication is considered in [7], namely, the variable chunks within a fixed size block are used to determine duplicated data, in order to increase the SSD device lifetime.

Despite the similarity of both techniques, there still is a possibility to obtain additional gain by a proper combination of them. Namely, the so-called Dedup Assisted Compression (DAC) technique was suggested [25], which allows to extend the lifetime of SSD by lowering the required number of writes, requiring no additional hardware modules. A study of the available techniques and their combinations (such as lossless compression, deduplication and performance throttling), focused solely on the lifetime extension of an SSD device, is performed in the paper [26]. A sophisticated approach combining the deduplication, compression and caching techniques, the so-called DedupeSwift, implemented in OpenStack Swift storage (widely used in the cloud storage systems) to reduce the memory footprint as well as increase read performance is designed in the work [27].

B. Fingerprinting

A known tradeoff between the accuracy of the duplicate detection and speed is studied in many papers. Since the fingerprinting is based on hash functions, the possibility of a hash collision depends on the properties of hash function, which in practice means that a reliable fingerprint is both performance- and space-consuming. To overcome this difficulty, pre-hashing (based on fast unreliable hash functions, such as CRC32) might be applied, together with an offline deduplication of the possibly duplicated pages [28]. As an alternative technique of duplicate detection, the byte to byte comparison of write page with its potential duplicate is suggested in [29]. The authors claim that high read speed and easy comparison implies less overhead, than the conventional fingerprinting techniques do. A novel approach based on a specific device for keeping the metadata, the so-called Content addressable memory, in order to avoid fingerprinting, is suggested in [30].

Note also, that efficiency of fingerprinting depends not only on the speed of a single lookup, but also on the properties of workload; this means that multiple consequent lookups may lead to high disk utilization resulting in a performance bottleneck. To address this issue, a buffering technique called “lazy” deduplication is suggested in [31]. It is also suggested to use parallel computing and general purpose graphical processing unit (GPGPU) to facilitate the fingerprinting process. A similar technique related to buffering and delaying the process in order to get better deduplication rate is introduced in [32]. Namely, the so-called containers, uniting several deduplicated blocks, are constructed after a delay so as to increase the utilization (in terms of the amount of data restored from the container). A similar page-level compression-based technique was suggested earlier in [33]. We also note the work [34] studying the fine-grained deduplication technique. An approach to route the containers to multiple disks (as an alternative to RAID (Redundant Array of Independent Disks) striping) in order to speedup the reading performance is suggested in the work [35].

C. Application-specific techniques

Several papers address the issues of compression and deduplication related to specific application fields. Compression algorithms and hardware (FPGA – Field-Programmable Gate Array) implementation for astronomy data storage are studied in [36]. High-speed SSD-based RAID storage with integrated compression for handling image data is presented in [9]. Deduplication techniques for Non-Linear Editing (NLE) based on causality between operations is studied in [37]. A word sequence-based decoding scheme based on statistical properties of a natural language allowing to increase the efficiency of error correction of traditional ECC schemes with respect to text file decoding is presented in the paper [38]. An intermediate compression layer lowering the write response time in multimedia devices is studied in [39], exploiting the same idea as suggested earlier in the works [40], [41]. The paper [42] addresses the problem of highly duplicated data in the logging mechanism of a filesystem, by suggesting a scheme for invalidating multiple copies of the same data in a common journaling module.

D. Other types

An interesting approach to deduplication management is suggested in [10]. The authors apply code analysis techniques (focusing on loops as main sources for duplicate data) and block/page based deduplication in order to reduce the in-place updated data.

The security issues of organizing data storage with deduplication over unreliable provider is addressed in [43]. The cryptography-enhanced scheme with a trusted proxy and group authentication is applied, and a prototype is implemented on a popular Dropbox web storage hosting.

It remains to note, that a Zynq-based platform for experiments to test and validate compression algorithms is presented in [44]

VI. Storage and network level

Deduplication on several linked devices raises a variety of new research problems. One of the most SSD-related among them is difference in deduplication on All-Flash and heterogeneous storages.

A. All-flash storage

Large scale storage applications require many SSD devices to work in parallel. In many cases the devices are organized into the RAID array, which suffers from the known problem of reliability degradation due to frequent parity updates. In the paper [45] the all-flash array reliability is studied under consideration of the prevalence of long sequential writes in the

storage workload. The proposed Dedup-assisted RAID (DA-RAID), similarly to the Diff-RAID technique, differentiates the amount of data written to each SSD in an array, thus minimizing the probability of multiple correlated failures, and, thanks to deduplication, allows to reduce the amount of parity updates, thus significantly extending the SSD lifetime. A similar idea related to parity data deduplication for SSD array lifetime extension in an OpenStack-based cloud is presented in [46].

Another problem of RAID-based all-flash arrays is related to separate treatment of deduplication and replication (for redundancy purposes), resulting in a suboptimal design of the storage. Simultaneous treatment of these two aspects by means of a dynamical content-aware replication management is performed in [47]. A similar problem of deduplication and replication coordination, in the context of HDFS-based (Hadoop Distributed File System) all-flash distributed storage, is addressed in [48] by means of new routing algorithms for the HDFS protocol.

B. Heterogeneous storage

Heterogeneous storage systems utilize large and inexpensive HDDs as permanent storage, while using the low latency and high bandwidth of the SSD devices for caching purposes. However, conventional caching mechanisms induce write stress, thus significantly shortening the lifetime of the SSD cache. Specific policies were developed, such as D-ARC, D-LRU, in order to fight this problem. These algorithms utilize deduplication to decrease the number of writes during cache replacement, thus increasing the cache endurance, as well as increasing the cache hit ratio [49]. Another possible solution is based on increasing the data persistence together with cache hit ratio for the SSD cache, as studied in [50], by means of the so-called PLC-cache technique, amplifying the proportion of popular and long-term cached (PLC) data in the SSD cache, resulting in read access latency decrease together with SSD lifetime extension. A specific solution based on dedup-aware caching algorithms and sophisticated metadata management scheme (by combining cache data and metadata management) is studied in [51]. Moreover, additional gain, as suggested by the authors, might be obtained by a combination of compression and deduplication techniques for flash caching. Somewhat similar approach is exploited in [52], where the SSD disk is used to speedup the deduplication process on a HDD-based storage. The paper [53] also addresses the performance issue of deduplication on an HDD-based storage, however, in the Cloud storage context, and tries to segregate performance-oriented small writes from capacity-oriented large ones.

The problem of backup and restore operations (e.g. in the virtualized environment) in a Cloud storage, related to high probability of backup fragmentation resulting in multiple disk I/O operations, is addressed in the works [54], [55]. Namely, the authors suggest to use an SSD disk as a read cache for non-sequential unique small data chunks with high reference. However, traditional cache replacement policies lead to fast SSD wearout, and this problem might be solved by applying special policies, such as keeping the long-term popular data, as suggested in [19]. A similar solution (by creating an SSD based cache in the HDD based storage) for improving the efficiency of backup operation is suggested in [56].

Finally, we note a few papers addressing the deduplication issues at an even higher level, such as the large virtual machine hosting cloud [57] and software-defined network [8].

VII. Linux deduplication implementations

Probably, the first example an approach similar to deduplication in UNIX is the so called “hard link”. Hard links give ability to have multiple names for one file, allowing to save some disk space occupied by identical files (see Fig. 5). The major consequence of the “one file — multiple names” semantic is that changing file using one name affects other file names too. Such behaviour makes hard links usable only for a limited set of scenarios, for example, by incremental backup software(https://sourceforge.net/projects/backuppc/), where any file which has not changed between two consecutive snapshots will appear as a single data area with multiple hard links in backup pool. Hard links can be created using the `link()` function.

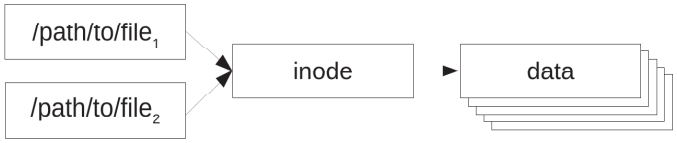

Fig. 5. Hard link

The so called “symbolic link” appeared later; a symbolic link contains a text string that is automatically interpreted and followed by the operating system as a path to another file or directory (see Fig. 6). Symbolic links are commonly used as pointers to the “real object location” for compatibility purposes, but their usage for deduplication is doubtful.

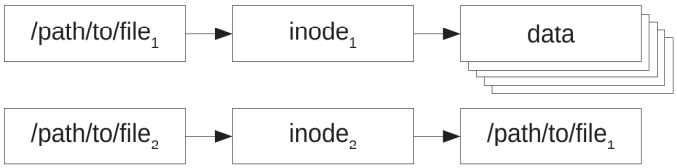

Fig. 6. Symbolic link

Recently the so called “reflink” has appeared in Linux filesystem development. Reflinks give ability to share data blocks (extents) among multiple files. Unlike the hard links, each file can be modified independently using copy-on-write (Fig. 7). This approach was traditionally used for memory management (cloning process memory on `fork()` calls, for example) but with modern file systems Linux is able now to clone also files efficiently enough.

Reflinks first appeared in the btrfs file system with `BTRFS_IOC_CLONE ioctl()` operation, but recently were included into the virtual file system layer of the Linux kernel with `FICLONE` and `FICLONERANGE ioctl()` operations. Another deduplication `ioctl()` operation is `FIDEDUPERANGE`, allowing block-level out-of-band deduplication. It could be used by any userspace program: first, to find out that some blocks in files are identical, and then to tell the kernel to merge them using reflinks. Ioctl operations

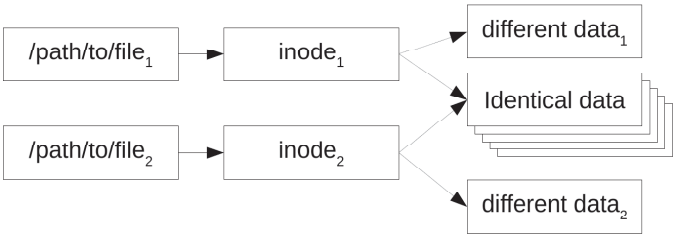

Fig. 7. Lightweight copy

mentioned above first appeared in Linux kernel 4.5(http://man7.org/linux/man-pages/man2/ioctl_ficlone.2.html). At the time of writing, the lightweight copy and deduplication are supported by btrfs, xfs, and ocfs2 file systems.

At least one storage vendor, NetApp Inc (multinational storage and data management company) have the so called "FlexClone" and "FlexVol", which exploit exactly the "multiple files sharing identical data blocks" approach described above [58]. The ability to easily clone production data volumes for development or backup purposes is obviously useful in many scenarios, and should be present in new and currently developed storage systems.

For all implementations described above the user or an application must explicitly require the Linux kernel to clone a file or merge some parts of two files. This is good for deduplication-aware applications (or users who know about "`cp --reflink`"): such application can take control over the process and specify exactly what and when to merge (or avoid creating duplicate data at all by using the file cloning feature).

However, for deduplication-unaware applications some kind of "duplication finder" is required. There are many of them, e.g., `duperemove`(https://github.com/markfasheh/duperemove). Such tools can save some disk space but they are not able to save time for data writing nor to decrease flash memory wear-out. There is a need for some in-line deduplication software for finding and storing duplicate data effectively and transparently for applications and users.

Two most noticeable implementations of open-source in-line deduplication software are OpenDedup (sdfs file system) [59] and Virtual Data Optimizer (VDO) [60].

The OpenDedup project has developed the sdfs file system – a clustered deduplicated file system licensed under GPLv2. It detects duplicate blocks using some kind of hash for each block. The project uses Java, which results in high CPU and memory overhead and relatively bad performance. Data and workload could be distributed across multiple servers. Data blocks can be stored also redundantly (which is exactly the opposite idea with respect to deduplication).

Another Linux inline deduplication implementation is VDO. It was developed by Permabit Technology Corporation and was not open source until 2017, when Red Hat had acquired the assets and technology of Permabit and opened VDO to the community. It currently has some problems related to its closed source origin (conformance with the Linux kernel coding standards, support for non-x86-64 platforms, refactoring of platform layer abstractions, and other changes requested by upstream maintainers etc). But the technology itself is thoroughly tested, mature, and in production use since 2014 in its previous proprietary form. With the high demand of the Linux community for a mature open-source inline deduplication software and the Red Hat support, we expect VDO to be ready for integration with the Linux kernel in the nearest future. Some performance observations for VDO used together with DRBD (Distributed Replicated Block Device) are available in [61]. In a testing environment VDO reduced the replication network and storage utilization (amount of transferred and written bytes) by about 85% while only increasing the load (the so called "la", load average) by about 0.8.

VIII. Open problems

Despite the ongoing active development of deduplication technology, there are still open research problems that seem promising for significant improvement of deduplication efficiency. These problems are related to both general deduplication process and specific issues of disk drives. Here we briefly describe several of these problems directly related to the SSD specifics.

An **efficient hash function**; the modern SSDs are equipped by a processing unit, but the performance of this unit is relatively low. Chunk hash computing using SHA-1 or MD5 consumes much CPU power. So, a lightweight hash function could make it possible to move chunk hash computing directly to the FTL layer of the disk. This could improve performance of data deduplication. However, a new hash function should still have a property of uniform mapping of data chunks to hash values. This is necessary to reduce the probability of collisions, which lead to data loss and file corruption. Low probability of collision (when different data chunks produce the same hash) is crucial. This is tightly connected to the hash size: larger hashes are less likely to collide. However, large hashes mean large hash table of stored data chunks, low data density, slower search, etc.

The previous problem is related to a more general problem of SSD development with **enough resources on-board to perform deduplication inside the disk**. Besides the hash computing, data deduplication consumes memory to store the hash table. So, memory consumption also have to be addressed by internal data deduplication.

Distributed storage data deduplication brings up issues on storing the hash table of distributed chunks, which could be a problem in the case of hot swap and dynamic change of the storage structure. Another problem is fragmentation: a file could be distributed among several different disks, so, it should be appropriately recomposed.

Hybrid storage requires taking into account diverse disk characteristics: random access memory is fast but expensive and volatile, so it is good to store the hash table, but not file references on chunks; SSD is fast but harmed by wear out and write amplification, so it is good to store chunks but not the hash table or file references to chunks if files often change; HDD is nonvolatile and comparatively cheap but has slow random access and is harmed by fragmentation.

So, effective techniques of data deduplication in hybrid storage are of interest.

Effective solution of these open problems requires special mathematical models and algorithms to be developed.

IX. Conclusion

Since early 2000s, data deduplication plays an increasingly important role in storage systems. As data volumes grow and SSD are used more widely, the existing methods are adopted to specifics of flash-based storage. It allows to reduce the number of writes and increase the disk lifetime, provides faster write operations for high-load information systems, and saves storage space.

We performed a survey on data deduplication techniques, focusing on SSD-aware techniques. We hope that the presented enhanced taxonomy of the research works on data deduplication for this particular subfield, as well as several open research problems related to SSD-aware data deduplication, could be a good starting point to development of new techniques and approaches of data deduplication in SSD-based storage.

Acknowledgment

This work is supported by the Ministry of Education and Science of Russian Federation (project no. 14.580.21.009, unique identifier RFMEFI58017X0009).

References

[1] Seagate. Data age 2025: The evolution of data to life-critical, 2017.

[2] Dutch T. Meyer and William J. Bolosky. A study of practical deduplication. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 1–1, Berkeley, CA, USA, 2011. USENIX Association.

[3] Cagdas Dirik and Bruce Jacob. The performance of pc solid-state disks (ssds) as a function of bandwidth, concurrency, device architecture, and system organization. *SIGARCH Comput. Archit. News*, 37(3):279–289, June 2009.

[4] Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, and Yukun Zhou. A comprehensive study of the past, present, and future of data deduplication. *Proceedings of the IEEE*, 104(9):1681–1710, September 2016.

[5] Bon-Keun Seo, Seungryoul Maeng, Joonwon Lee, and Euiseong Seo. DRACO: A deduplicating FTL for tangible extra capacity. *IEEE computer architecture letters*, 14(2):123–126, JUL-DEC 2015.

[6] Sungjin Lee, Jihoon Park, Kermin Fleming, Arvind, and Jihong Kim. Improving performance and lifetime of solid-state drives using hardware-accelerated compression. *IEEE transactions on consumer electronics*, 57(4):1732–1739, NOV 2011.

[7] Jin-Yong Ha, Young-Sik Lee, and Jin-Soo Kim. Deduplication with block-level content-aware chunking for solid state drives (SSDs). In *2013 IEEE 15TH international conference on high performance computing and communications & 2013 IEEE international conference on embedded and ubiquitous computing (HPCC_EUC)*, pages 1982–1989, New York, USA, 2013. IEEE.

[8] Yu Hua, Xue Liu, and Dan Feng. Smart in-network deduplication for storage-aware SDN. *ACM sigcomm computer communication review*, 43(4):509–510, OCT 2013.

[9] Wang Shiming, Xu Zhiyong, Zhang Yao, and Fu Chengyu. PCIE interface design for high-speed image storage system based on SSD. In Tang, C and Chen, S and Tang, X, editor, *XX international symposium on high-power laser systems and applications 2014*, volume 9255 of *Proceedings of SPIE*, Bellingham, WA USA, 2015. SPIE-INT soc optical engineering.

[10] Joon-Young Paik, Tae-Sun Chung, and Eun-Sun Cho. Application-aware deduplication for performance improvement of flash memory. *Design automation for embedded systems*, 19(1-2):161–188, MAR 2015.

[11] João Paulo and José Pereira. A survey and classification of storage deduplication systems. *ACM computing surveys*, 47(1):1–30, June 2014.

[12] Nagapramod Mandagere, Pin Zhou, Mark A Smith, and Sandeep Uttamchandani. Demystifying data deduplication. In *Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, Companion '08, pages 12–17, New York, NY, USA, 2008. ACM.

[13] Youngjoo Shin, Dongyoung Koo, and Junbeom Hur. A survey of secure data deduplication schemes for cloud storage systems. *ACM computing surveys*, 49(4):1–38, January 2017.

[14] E. Manogar and S. Abirami. A study on data deduplication techniques for optimized storage. In *Advanced computing (ICoAC), 2014 Sixth International Conference on*, pages 161–166. IEEE, 2014.

[15] Ruijin Zhou, Ming Liu, and Tao Li. Characterizing the efficiency of data deduplication for big data storage management. In *2013 IEEE International symposium on workload characterization (IISWC 2013)*, International Symposium on Workload Characterization Proceedings, pages 98–108, New York, USA, 2013. IEEE.

[16] Jonghwa Kim, Choonghyun Lee, Sangyup Lee, Ikjoon Son, Jongmoo Choi, Sungroh Yoon, Hu-ung Lee, Sooyong Kang, Youjip Won, and Jaehyuk Cha. Deduplication in SSDs: Model and quantitative analysis. In *2012 IEEE 28th symposium on mass storage systems and technologies (MSST)*, IEEE Symposium on Mass Storage Systems and Technologies Proceedings-MSST, New York, USA, 2012. IEEE.

[17] Youngjo Park and Jin-Soo Kim. zFTL: Power-efficient data compression support for NAND flash-based consumer electronics devices. *IEEE transactions on consumer electronics*, 57(3):1148–1156, AUG 2011.

[18] Ningde Xie, Guiqiang Dong, and Tong Zhang. Using lossless data compression in data storage systems: Not for saving space. *IEEE transactions on computers*, 60(3):335–345, MAR 2011.

[19] Jian Liu, Yun-Peng Chai, Xiao Qin, and Yao-Hong Liu. Endurable SSD-based read cache for improving the performance of selective restore from deduplication systems. *Journal of computer science and technology*, 33(1):58–78, JAN 2018.

[20] Juergen Freudenberger, Mohammed Rajab, Daniel Rohweder, and Malek Safieh. A codec architecture for the compression of short data blocks. *Journal of circuits systems and computers*, 27(2), FEB 2018.

[21] Xuebin Zhang, Jiangpeng Li, Hao Wang, Kai Zhao, and Tong Zhang. Reducing solid-state storage device write stress through opportunistic in-place delta compression. In *14TH USENIX Conference on file and storage technologies (FAST '16)*, pages 111–124, Berkeley, USA, 2016. USENIX ASSOC.

[22] Ashish Jagmohan, Michele Franceschini, and Luis Lastras. Write amplification reduction in NAND flash through multi-write coding. In MG Khatibm, X He, and M Factor, editors, *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, IEEE Symposium on Mass Storage Systems-Proceedings, New York, USA, 2010. IEEE.

[23] Kyuwoon Kim, Sanghyuk Jung, and Yong Ho Song. Compression ratio based hot/cold data identification for flash memory. In *IEEE International conference on consumer electronics (ICCE 2011)*, pages 33–34, New York, USA, 2011. IEEE.

[24] Juergen Freudenbrger, Alexander Beck, and Mohammed Rajab. A data compression scheme for reliable data storage in non-volatile memories. In *2015 IEEE 5th international conference on consumer electronics - Berlin (ICCE-BERLIN)*, pages 139–142, Bilbao, Spain, 2015. UNIV BASQUE COUNTRY UPV-EHU PRESS.

[25] Jisung Park, Sungjin Lee, and Jihong Kim. DAC: Dedup-assisted compression scheme for improving lifetime of NAND storage systems. In *Proc. of the 2017 design, automation & test in Europe conference & exhibition (DATE)*, Design Automation and Test in Europe Conference and Exhibition, pages 1249–1252, New York, USA, 2017. IEEE.

[26] Sungjin Lee, Taejin Kim, Ji-Sung Park, and Jihong Kim. An integrated approach for managing the lifetime of flash-based SSDs. In *Design, automation & test in Europe*, Design Automation and Test in Europe Conference and Exhibition, pages 1522–1525, New York, USA, 2013. Assoc computing machinery.

[27] Jingwei Ma, Gang Wang, and Xiaoguang Liu. DedupeSwift: object-

oriented storage system based on data deduplication. In *Trustcom/BigDataSE/I SPA, 2016 IEEE*, pages 1069–1076. IEEE, 2016.

[28] Eunsoo Park and Dongkun Shin. Offline deduplication for solid state disk using a lightweight hash algorithm. *JSTS: journal of semiconductor technology and science*, 15(5):539–545, October 2015.

[29] Zhengguo Chen, Zhiguang Chen, Nong Xiao, and Fang Liu. Nf-dedupe: A novel no-fingerprint deduplication scheme for flash-based ssds. In *2015 IEEE symposium on computers and communication (ISCC)*, pages 588–594, New York, USA, 2015. IEEE.

[30] Roman Kaplan, Leonid Yavits, Amir Morad, and Ran Ginosar. Deduplication in resistive content addressable memory based solid state drive. In *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26th International Workshop on*, pages 100–106. IEEE, 2016.

[31] Jingwei Ma, Rebecca J. Stones, Yuxiang Ma, Jingui Wang, Junjie Ren, Gang Wang, and Xiaoguang Liu. Lazy exact deduplication. *ACM transactions on storage*, 13(2):1–26, June 2017.

[32] Jian Liu, Yunpeng Chai, Chang Yan, and Xin Wang. A delayed container organization approach to improve restore speed for deduplication systems. *IEEE transactions on parallel and distributed systems*, 27(9):2477–2491, September 2016.

[33] KS Yim, K Koh, and H Bahn. A compressed page management scheme for NAND-type flash memory. In HR Arabnia and LT Yang, editors, *VLSI'03: Proceedings of the international conference on VLSI*, pages 266–271, Athens, GA USA, 2003. CSREA Press.

[34] Taejin Kim, Sungjin Lee, and Jihong Kim. FineDedup: A fine-grained deduplication technique for extending lifetime of flash-based SSDs. *Journal of semiconductor technology and science*, 17(5):648–659, OCT 2017.

[35] Chao Li, Shupeng Wang, Chunyun Xiao, Xiaoyang Zhou, and Guangjun Wu. MMD: an approach to improve reading performance in deduplication systems. In *9th IEEE International Conference on Networking, Architecture, and Storage (NAS)*, pages 93–97. IEEE, August 2014.

[36] Bo Peng, Xi Jin, Tianqi Wang, and Xueliang Du. Design of a distributed compressor for astronomy ssd. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, page 98, New York, USA, 2015. IEEE.

[37] Man-Keun Seo and Seung-Ho Lim. Deduplication flash file system with PRAM for non-linear editing. *IEEE transactions on consumer electronics*, 56(3):1502–1510, AUG 2010.

[38] Yue Li, Yue Wang, Anxiao (Andrew) Jiang, and Jehoshua Bruck. Content-assisted file decoding for nonvolatile memories. In MB Matthews, editor, *2012 conference record of the forty sixth asilomar conference on signals, systems and computers (ASILOMAR)*, Conference Record of the Asilomar Conference on Signals Systems and Computers, pages 937–941, New York, USA, 2012. IEEE.

[39] W. T. Huang, C. T. Chen, and C. H. Chen. The real-time compression layer for flash memory in mobile multimedia devices. In SS Kim, JH Park, N Pissinou, TH Kim, WC Fang, D Slezak, HR Arabnia, and D Howard, editors, *MUE: 2007 International conference on multimedia and ubiquitous engineering, proceedings*, pages 171+, Los Alamitos, CA USA, 2007. IEEE Computer soc.

[40] WT Huang, CT Chen, YS Chen, and CH Chen. A compression layer for NAND type flash memory systems. In X He, T Hintz, M Piccardi, Q Wu, M Huang, and D Tien, editors, *Third International Conference on Information Technology and Applications, Vol 1, Proceedings*, pages 599–604, Los Alamitos, CA USA, 2005. IEEE COMPUTER SOC.

[41] M Kjelso and S Jones. Memory management in flash-memory disks with data compression. In HG Baker, editor, *Memory management*, volume 986 of *Lecture notes in computer science*, pages 399–413, Berlin, Germany, 1995. Springer-Verlag Berlin.

[42] Seung-Ho Lim and Young-Sik Jeong. Journaling deduplication with invalidation scheme for flash storage-based smart systems. *Journal of systems architecture*, 60(8):684–692, September 2014.

[43] Wen Bing Chuan, Shu Qin Ren, Sye Loong Keoh, and Khin Mi Mi Aung. Flexible yet secure de-duplication service for enterprise data on cloud storage. In *International Conference on Cloud Computing Research and Innovation (ICCCRI)*, pages 37–44. IEEE, October 2015.

[44] Debao Wei, Youhua Gong, Liyan Qiao, and Libao Deng. A Hardware-Software Co-design Experiments Platform for NAND Flash Based on Zynq. In *2014 IEEE 20th international conference on embedded and real-time computing systems and applications (RTCSA)*, New York, USA, 2014. IEEE.

[45] Taejin Kim, Sungjin Lee, Jisung Park, and Jihong Kim. Efficient lifetime management of SSD-based RAIDs using dedup-assisted partial stripe writes. In *2016 5TH Non-volatile memory systems and applications symposium (NVMSA)*, IEEE Non-Volatile Memory Systems and Applications Symposium, New York, USA, 2016. IEEE.

[46] Huiseong Heo, Cheongjin Ahn, and Deok-Hwan Kim. Parity data deduplication in all flash array-based OpenStack cloud block storage storage. *IEICE transactions on information and systems*, E99.D(5):1384–1387, 2016.

[47] Yimo Du, Youtao Zhang, and Nong Xiao. R-Dedup: content aware redundancy management for SSD-based RAID systems. In *43rd International Conference on Parallel Processing (ICPP)*, pages 111–120. IEEE, September 2014.

[48] Binqi Zhang, Chen Wang, Bing Bing Zhou, and Albert Y. Zomaya. Inline data deduplication for SSD-based distributed storage. In *IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pages 593–600. IEEE, December 2015.

[49] Xian Chen, Wenzhi Chen, Zhongyong Lu, Peng Long, Shuiqiao Yang, and Zonghui Wang. A duplication-aware ssd-based cache architecture for primary storage in virtualization environment. *IEEE Systems journal*, 11(4):2578–2589, DEC 2017.

[50] Jian Liu, Yunpeng Chai, Xiao Qin, and Yuan Xiao. PLC-cache: Endurable SSD cache for deduplication-based primary storage. In *2014 30th symposium on massive storage systems and technologies (MSST)*, IEEE Symposium on Mass Storage Systems and Technologies Proceedings-MSST, New York, USA, 2014. IEEE.

[51] Wenji Li, Gregory Jean-Baptise, Juan Riveros, Giri Narasimhan, Tong Zhang, and Ming Zhao. Cachededup: In-line deduplication for flash caching. In *14th usenix conference on file and storage technologies (FAST '16)*, pages 301–314, Berkeley, USA, 2016. USENIX ASSOC.

[52] Dirk Meister and Andre Brinkmann. dedupv1: Improving deduplication throughput using solid state drives (SSD). In MG Khatibm, X He, and M Factor, editors, *2010 IEEE 26TH symposium on mass storage systems and technologies (MSST)*, IEEE Symposium on Mass Storage Systems-Proceedings, New York, USA, 2010. IEEE.

[53] Bo Mao, Hong Jiang, Suzhen Wu, and Lei Tian. POD: performance oriented I/O deduplication for primary storage systems in the cloud. In *IEEE 28th International Parallel and Distributed Processing Symposium*, pages 767–776. IEEE, May 2014.

[54] Bo Mao, Hong Jiang, Suzhen Wu, Yinjin Fu, and Lei Tian. Read-performance optimization for deduplication-based storage systems in the cloud. *ACM transactions on storage*, 10(2), MAR 2014.

[55] Bo Mao, Hong Jiang, Suzhen Wu, Yinjin Fu, and Lei Tian. SAR: SSD assisted restore optimization for deduplication-based storage systems in the cloud. In *IEEE 7th International Conference on Networking, Architecture and Storage (NAS)*, pages 328–337. IEEE, June 2012.

[56] Longxin Lin, Kun Xiao, and Wenjie Liu. Utilizing SSD to alleviate chunk fragmentation in de-duplicated backup systems. In *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*, pages 616–624. IEEE, 2016.

[57] Xun Zhao, Yang Zhang, Yongwei Wu, Kang Chen, Jinlei Jiang, and Keqin Li. Liquid: a scalable deduplication file system for virtual machine images. *IEEE transactions on parallel and distributed systems*, 25(5):1257–1266, May 2014.

[58] M Kilvansky. A thorough introduction to flexclone volumes. *NetApp, Technical White Paper*, 2004.

[59] Jeramiah Bowling. Opendedup: open-source deduplication put to the test. *Linux Journal*, 2013(228):2, 2013.

[60] Data deduplication and compression with vdo. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/vdo. [Online; accessed 24-Feb-2018].

[61] Albireo virtual data optimizer (vdo) on drbd. https://www.linbit.com/en/albireo-virtual-data-optimizer-vdo-on-drbd/. [Online; accessed 24-Feb-2018].