# A Survey on the Development of Data Deduplication Technology (Postprint)

**Authors:** Wang Shupeng, Yun Xiaochun, Guo Li

**Date:** 2017-03-10T00:00:00+00:00

## Abstract

As informationization continues to advance, data volumes are experiencing explosive growth. This places increasing pressure on data storage management, particularly as duplicate content within data leads to significant waste of storage space. To enhance storage space utilization, deduplication (data redundancy elimination) technology has emerged. This article provides a comprehensive summary and analysis of the background, classification, key technologies, application scenarios, current development status, and future trends of deduplication technology.

## Full Text

## Preamble

Vol. 9 No. 3 | Information Technology Letters | A Survey on the Development of Data Deduplication Technology
Wang Shupeng, Yun Xiaochun, Guo Li

## Abstract

As information technology continues to advance, data volumes are exploding, placing increasing pressure on data storage management. In particular, redundant content within data causes significant waste of storage space. To improve storage utilization, data deduplication (data redundancy elimination) technology has emerged. This paper provides a comprehensive analysis and summary of the background, classification, key technologies, application scenarios, current development status, and future trends of data deduplication technology.

**Keywords:** Data Deduplication, Data Redundancy Elimination, Storage Optimization

## 1. Technical Background

The explosive growth and large-scale centralization of data have exacerbated the problem of space waste caused by data redundancy, prompting the emergence and development of data redundancy elimination technology. Current research in this area has achieved valuable results in improving deduplication ratios, optimizing performance, and ensuring reliability, effectively promoting the application of this technology.

In service-oriented cloud storage systems, the large-scale centralization of user data makes data redundancy elimination even more necessary while also presenting new challenges. To meet different user requirements for deduplication ratio, reliability, and performance, storage systems must be able to effectively control and adjust the quality of service for these attributes. Additionally, as storage system scale increases, energy conservation, quality of service (QoS) guarantee, and power reduction have become critical issues that need to be addressed in deduplication storage. Current research results cannot fully solve these problems or meet the practical needs of service-oriented large-scale storage systems. The Natural Science Foundation project applied for by the authors, "Research on Data Redundancy Elimination Storage Technology for Service Quality and Energy Consumption Optimization," focuses specifically on this issue, aiming to establish a theoretical and technical system for data redundancy elimination oriented toward service quality and energy consumption optimization. We first studied the key attributes of data redundancy elimination and proposed a quality of service description and evaluation methodology; then we studied the energy consumption of data redundancy elimination and proposed energy consumption analysis methods and optimization strategies; finally, we studied multi-objective data redundancy elimination technology and implemented service quality and energy consumption optimization based on service quality benefits and energy cost functions. This research aligns with the development needs of storage technology and holds significant theoretical and practical value.

Below, we analyze and elaborate on the relevant background and development status of data deduplication technology. The purpose of data deduplication (data redundancy elimination) is to eliminate redundant data globally within storage systems, including both intra-file and inter-file redundancy, whereas traditional data compression can only eliminate redundancy within individual files. In comparison, data deduplication technology achieves significantly better compression effects, with deduplication ratios reaching 300:1 or even higher for specific application data, while data compression technology only achieves ratios of approximately 2:1. Because data deduplication technology plays a crucial role in data storage, it has received widespread attention in recent years and become a hot research topic in the data storage field [1,3,4,5,6,7,8], ranking first among five hot research issues in data storage and protection.

In the early stages, extensive research focused on improving deduplication ratios by continuously reducing deduplication granularity. EMC's Centera system

[13] and Windows' Single Instance Storage system [14] adopted file-level deduplication methods. These methods offer simple implementation and fast computation but suffer from coarse detection granularity and poor deduplication effectiveness. To improve deduplication ratios, researchers proposed fixed-size chunking methods that divide files into fixed-length data blocks for redundancy elimination. These methods provide fast computation and are sensitive to data changes, but their main drawback is that inserting or modifying content within files severely impacts deduplication effectiveness. This method was applied in the Venti archival storage system [5]. To further improve deduplication ratios, researchers proposed variable-size chunking methods that use Rabin fingerprinting or similar functions to determine block boundaries [15], dividing changed content into separate blocks. Typical applications include Shark [8] and Deep Store [7]. Additionally, researchers have proposed byte-level deduplication mechanisms [11] that first identify highly similar data blocks and then use delta compression to calculate differences between blocks, storing only the differential content. Common delta compression algorithms include zDelta [12].

As data deduplication technology has been applied in massive storage systems, its impact on storage system throughput has become apparent, and performance issues have gradually attracted researchers' attention. References [4,16,20,24] have conducted research on this problem: Reference [4] proposes using Bloom Filters and locality-based caching to reduce disk I/O operations during deduplication and improve deduplication speed; Reference [16] improves data read/write performance by packaging data blocks into fixed-length objects; Reference [20] proposes a two-stage deduplication mechanism that improves throughput by converting random small disk I/Os into sequential large I/Os; Reference [24] solves the bottleneck of duplicate block lookup using locality principles, achieving high throughput with limited memory; Reference [25] proposes a method to reduce query operations during deduplication based on file similarity characteristics for systems lacking data locality features.

While improving storage space utilization, data deduplication technology causes a data block to be referenced by multiple file objects. The loss of a single data block can compromise file availability, thereby reducing data storage reliability. To meet the reliability requirements of critical storage systems, researchers have proposed using redundancy replication or erasure coding methods to improve storage reliability: L. You' s work in reference [7] quantitatively analyzes that blocks with high reference counts should have higher redundancy levels and provides a method for improving reliability through redundancy replication, but does not provide quantitative methods for calculating redundancy levels or evaluate data reliability and storage overhead after redundancy mechanisms; D. Bhagwat builds upon this work by providing a method to calculate redundancy levels based on block reference counts [22] and adopts redundancy replication to improve reliability, but does not conduct overall evaluation of block redundancy, storage overhead, and reliability, nor does it provide optimal redundancy calculation methods; Chuanyi Liu from Tsinghua University proposes R-ADMAD [23], a reliability guarantee mechanism for deduplication-based archival stor-

age systems that uses ECC encoding to encode storage objects and distributes them across storage nodes in a redundancy group, with failure recovery through a distributed dynamic recovery mechanism.

### 2.1 Classification Based on Duplicate Content Identification Methods

**(1) Hash-Based Identification**
This method uses data hash values to identify duplicates. For each new data block, a hash is generated. If the block' s hash matches a hash in the storage device' s hash index, the block is considered duplicate. Data Domain, Falcon, and Quantum' s DXi series devices all use hash algorithms like SHA-1 and MD-5 for deduplication.

Hash-based methods have inherent scalability issues. To quickly identify whether a block has been stored, these methods maintain a hash index in memory. As the number of data blocks increases, the index grows accordingly. Once the index exceeds the device' s memory capacity, performance drops sharply as disk searches are much slower than memory searches. Consequently, most current hash-based systems are standalone, maintaining a balance between memory required for stored data and disk space. This design prevents the hash table from becoming too large.

**(2) Content-Based Identification**
This method uses file system metadata embedded in data to identify files and performs byte-by-byte comparison with other versions in the data repository to find differences, creating delta files for changed data. This approach avoids hash collisions but requires application devices that support this functionality to extract metadata.

**(3) ProtecTier Virtual Tape Library (VTL) Technology**
Similar to hash-based products, this method divides data into chunks and uses proprietary algorithms to determine whether a given chunk is similar to other chunks, then performs byte-by-byte comparison with data in similar chunks to determine if the chunk has already been stored.

### 2.2 Classification Based on Deduplication Granularity

**(1) Full-File Level Deduplication**
This approach detects and eliminates duplicates at the entire file level by computing the hash value of the complete file and checking if an identical file exists in the storage system. The advantage is fast computation on standard hardware; the disadvantage is that it cannot eliminate duplicate data within different files even if they share substantial identical content.

**(2) File Chunk Deduplication**
This method divides a file into chunks using various approaches and performs detection at the chunk level. The advantages are fast computation and sensitivity to data changes.

**(3) Byte-Level Deduplication**
This approach identifies and removes duplicate content at the byte level, typically using delta compression strategies to generate differential content. Byte-level deduplication offers high deduplication ratios but suffers from slow processing speeds.

### 2.3 Classification Based on Deduplication Execution Order

**(1) Inline Deduplication**
Inline deduplication performs redundancy elimination before data is written to disk. Its main advantage is cost-effectiveness, reducing storage capacity requirements and eliminating the need to retain non-deduplicated datasets. However, inline processing slows data throughput, and because deduplication occurs before disk writes, the process itself becomes a single-point bottleneck.

**(2) Post-Processing Deduplication**
Post-processing deduplication, also known as offline deduplication, performs redundancy elimination after data has been written to disk. Data is first written to temporary disk space, after which deduplication begins, and finally the deduplicated data is copied to final disk storage. Since deduplication occurs after data writing on separate storage devices, it does not impact normal business operations. Administrators can schedule deduplication processes arbitrarily. Typically, backup data is retained on disk before deduplication, allowing enterprises to access recently stored files and data more quickly when needed. The main drawback of post-processing is the requirement for additional disk space to store the complete non-deduplicated dataset.

### 2.4 Classification Based on Implementation Level

**(1) Software-Based Deduplication**
At the software level, deduplication can be integrated in two ways: either by installing software products on dedicated servers or by integrating them into backup/archival software. Software-based deduplication has lower deployment costs but can disrupt operations during installation and is difficult to maintain. Products include EMC' s Avamar, Symantec' s Veritas NetBackup, and Sepaton' s DeltaStor storage software.

**(2) Hardware-Based Deduplication**
Hardware-based deduplication is performed by the storage system itself, integrating deduplication mechanisms into virtual tape library systems, backup platforms, or general-purpose storage systems like Network Attached Storage (NAS). The advantages are high performance, scalability, and relatively non-disruptive deployment, with deduplication operations being transparent to upper-layer applications. The disadvantage is higher deployment cost compared to software-based solutions.

Current hardware-based deduplication systems primarily include Virtual Tape Library (VTL) and NAS backup products, such as Data Domain' s DD410 series,

Diligent Technologies' ProtecTier VTL, Quantum's DXi3500 and DXi5500 series, Falcon's VTL, ExaGrid Systems'NAS backup products, and NetApp's NearStore R200 and FAS storage systems.

## 3. Identical Data Deduplication Technology

Identical data deduplication technology partitions data to identify identical portions and replaces identical data with pointers.

### 3.1 Identical File Deduplication Technology

Identical file deduplication technology identifies duplicate data at the file granularity [6]. As shown in Figure 1 [Figure 1: see original paper], it computes the hash value (SHA-1 or MD5) for the entire file and compares it with stored hash values. If a matching hash is found, the file is considered duplicate and not stored; otherwise, the file is new and both the file and its hash value are stored in the system.

EMC's Centera system [1] and Windows'Single Instance Storage system [2] adopt this method. Testing on a server with 20 different Windows NT images using Windows 2000's SIS (Single Instance Storage) technology showed a total storage space savings of 58%. The advantage of this method is fast deduplication speed; the disadvantage is its inability to eliminate identical data within different files.

### 3.2 Fixed-Length Chunking Deduplication Technology

The fixed-length chunking method, shown in Figure 2 [Figure 2: see original paper], divides a data object (file) into non-overlapping fixed-size chunks, computes each chunk's hash value (SHA-1 or MD5), and compares these hash values with stored values. If a matching hash is found, the chunk is considered duplicate and only its hash value and reference information are stored; otherwise, the chunk is new and the chunk itself, its hash value, and reference information are stored.

The main problem with this method is that inserting or deleting data from a data object causes misalignment of chunk boundaries, severely impacting deduplication effectiveness. As shown in Figure 3 [Figure 3: see original paper], Version 1 of a data object generates n fixed-size chunks D1, D2, ···, Dn. When Version 2 inserts some content (shown in shadow) based on Version 1, the chunking of Version 2 produces chunks D1, D' 2, ···, D' n, where only D1 is duplicate. None of D' 2 through D' n are duplicate chunks, causing all duplicate data from the insertion point to the end to remain undetected and reducing the deduplication ratio.

This method has been applied in many systems, with a typical application being the Venti archival storage system for network storage [3], which achieved approximately 30% space savings using this technology.

### 3.3 Content-Defined Chunking Deduplication Technology

To address these problems, researchers proposed content-defined chunking deduplication methods [7] (shown in Figure 4 [Figure 4: see original paper]). This approach uses a sliding window to determine chunk boundaries, employing Rabin fingerprinting to compute the fingerprint of the sliding window. When predetermined conditions are met, the window' s starting position is marked as a chunk boundary. By continuously sliding the window and computing fingerprints, the data object is partitioned. To avoid extreme cases of overly long or short chunks, minimum and maximum chunk sizes can be set. Each resulting chunk' s hash value is then compared to identify duplicates, following the same process described above.

Because chunks are defined by content rather than length, this method effectively solves the problems of fixed-length chunking. When content is inserted or deleted, if the change occurs outside the sliding window boundary region, the boundary remains unchanged. When inserted content creates a new boundary, one chunk splits into two; otherwise, chunks remain unchanged. If changes occur within the sliding window, they may break boundary chunks, causing two chunks to merge or boundaries to shift and create new chunks. Therefore, insertions or deletions only affect one or two adjacent chunks, leaving other chunks unaffected, enabling detection of more duplicate data between objects. As shown in Figure 5 [Figure 5: see original paper], when content is inserted into a file, the chunking process places the inserted content into a separate chunk, preserving subsequent chunks and ensuring that later duplicate chunks can be eliminated.

Typical applications of this method include Shark [4] and Deep Store [5], which are used in Low Bandwidth File Systems (LBFS). In LBFS, minimum and maximum chunk length boundaries are applied to avoid excessively long or short chunks.

### 3.4 Sliding Block Deduplication Technology

Content-defined chunking solves byte insertion and deletion problems but introduces complexity in storing variable-size chunks. To address this, sliding block deduplication detection and elimination methods [8] were developed (shown in Figure 6 [Figure 6: see original paper]), solving problems inherent in both fixed-size and content-defined chunking.

The sliding block method uses rsync Checksum and sliding window techniques for chunking. The rsync Checksum algorithm offers fast computation and high efficiency. The computed checksum values are compared with previously stored values. If a match is found, the data block' s SHA-1 value is computed for comparison to detect duplicates. When duplicate blocks are found, they are recorded and the sliding window moves past the duplicate block to continue detection. Data between the previous chunk' s end and the newly detected duplicate block is recorded and stored. If neither the checksum nor hash values

match, the detection process continues. If the sliding window moves a distance equal to the fixed chunk length before finding a duplicate block, the block' s hash value is computed and stored for future verification.

The sliding block method solves data insertion problems by detecting every block of an object. If content is inserted into a data object, only surrounding blocks change, while subsequent blocks can still be identified and detected. Similarly, when content is deleted, blocks after the deleted portion remain unaffected and can still be detected.

### 3.5 Fingerdiff Algorithm-Based Deduplication Technology

To address the high storage overhead of content-defined chunking algorithms, researchers proposed the fingerdiff algorithm [6]. Its core idea is to merge unchanged blocks as much as possible to reduce metadata storage overhead. The technique consists of three main processes: (1) a file is partitioned using content-defined chunking; (2) each sub-block is merged according to fingerdiff' s maximum sub-block count setting; (3) each block' s fingerprint is computed using a hash function and compared with stored fingerprints. If matching fingerprints are detected, the corresponding block is eliminated; otherwise, the large block is split to find the smallest different block for storage while remaining blocks stay merged.

### 3.6 Data Feature-Based Deduplication Algorithm

Although content-defined chunking strategies solve some fixed-size chunking problems, they still cannot achieve optimal chunking for specific data file types. To address this, data feature-based chunking strategies have emerged. For example, for PPT files, the strategy divides files into chunks based on each slide, effectively eliminating identical slides. Others have proposed deduplication technology that dynamically selects different chunking strategies based on data type: using file feature-based deduplication for PPT and DOC files while employing fixed-size chunking for executable files.

## 4. Similar Data Deduplication Technology

In addition to eliminating identical data, storage space can be saved and utilization improved through similar data detection and encoding. Similar data deduplication includes two phases: detection and encoding. Similar data detection techniques include:

Shingle detection technology simplifies document similarity to set similarity by extracting a set of features for each document [9]. Shingle detection is simple to implement and widely applicable but has high computational overhead, and its detection accuracy depends on shingle sampling techniques, which can introduce significant bias.

Bloom Filters are bit-array-based sets [10] that support membership queries. Bloom Filters overcome the high computational overhead of shingle detection, achieving a balance between performance and similarity detection accuracy. They perform fast bit operations for matching with minimal computational overhead.

Pattern matching can also detect similar data by mining data features. Pattern matching algorithms use a certain number of common strings for similarity search and discrimination between files. This technique requires scanning entire files, resulting in relatively high overhead.

Based on similar data detection, encoding data with high similarity can save substantial storage space for the entire system. However, similar data compression technology faces challenges in encoding efficiency and applicability.

## 5. Performance Enhancement Techniques for Data Deduplication

While improving storage space utilization, data deduplication impacts system data access performance because duplicate detection processes consume substantial system resources, severely affecting storage system access performance. Several solutions have emerged to address this problem.

To handle the issue of memory space being unable to accommodate all data indexes, Data Domain employs a three-level lookup [11]: Bloom Filter filtering, hash cache query, and hash file query. First, the in-memory Bloom Filter is searched—a hash summarizes the presence information of n block fingerprints in the block index using an m-bit vector. If the Bloom Filter indicates a block does not exist, it definitely does not exist; if it indicates the block may exist, the hash cache is searched next. If found there, the block exists; otherwise, disk lookup is performed. For on-disk data organization, stream-based block arrangement is used to effectively exploit data locality and improve cache hit rates.

For systems without obvious data access locality characteristics, researchers have proposed reducing query operations during deduplication based on file similarity features to improve performance. Others have adopted a two-stage deduplication mechanism [12] that improves throughput by converting random small disk I/Os into sequential large I/Os. Some have also employed two-level indexing technology to reduce disk I/O operations [13] and improve deduplication throughput.

Analysis of existing technologies reveals that the key to improving deduplication throughput is reducing disk I/O operations, with current methods employing various strategies to minimize disk reads and writes during block retrieval.

### 6.1 Data Backup Systems

Data deduplication technology has brought revolutionary breakthroughs to data protection, effectively improving the cost-effectiveness of disk-based data protection. Traditional data protection could not achieve deduplication, often relying on inexpensive tape libraries as backup devices. Tape backup struggles to meet user requirements for backup windows and recovery speeds. Today, disk-based data protection solutions such as virtual tape libraries are widely adopted and will continue to grow. Backing up to virtual tape libraries or other disk-based systems has shortened backup windows and improved backup and recovery capabilities. However, as data volumes continue to increase, we face mounting capacity pressure. Data deduplication technology provides an effective method to minimize storage capacity requirements.

### 6.2 Archival Storage Systems

Data deduplication is also crucial for archival storage. As reference data volumes continue to grow, regulatory compliance requirements demand longer online data retention, and high-performance needs require disk-based archiving, enterprises face cost issues when implementing archival storage. An ideal archival storage system should meet long-term data preservation needs while maintaining lower total cost of ownership than production systems. Data deduplication technology enables efficient archival storage by eliminating redundancy to achieve the lowest possible cost. Currently, deduplication technology in archival storage systems is primarily hash-based, with products sold mainly as Content Addressable Storage (CAS) technology in both pure software and storage system forms.

### 6.3 Remote Disaster Recovery Systems

In remote disaster recovery systems, large amounts of data must be migrated to remote sites. As data volumes grow, data transmission pressure increases. By detecting and eliminating duplicate data before transmission, data deduplication technology can effectively reduce transmitted data volume and improve transmission speed, with typical products such as Falcon' s MicroScan software.

## 7. Summary and Development Trends

The above analysis shows that researchers have conducted extensive studies on deduplication ratio, performance, and reliability according to practical application requirements, achieving many valuable results. However, in service-oriented storage systems, different users or applications have varying requirements for deduplication ratio, performance, and reliability attributes. To meet these diverse needs, dynamic adjustment of these attributes based on user requirements is necessary [21]. Current research in this area remains preliminary and cannot meet the application requirements of service-oriented large-scale storage systems.

Furthermore, as storage scale continues to expand, energy consumption issues become increasingly prominent. Storage system energy consumption has reached 40% of IT system energy consumption and continues to rise. Although researchers have conducted studies on storage system energy consumption and optimization methods, achieving valuable results, they have overlooked the impact of data deduplication on storage system energy consumption. Our preliminary research shows that deduplication mechanisms increase storage system energy consumption and affect the effectiveness of existing energy optimization mechanisms, creating a contradictory relationship between service quality and energy consumption.

Therefore, when studying the service quality of data deduplication, energy consumption factors must also be considered to achieve effective balance and adjustment between service quality and energy overhead.

In summary, the following research problems need to be addressed: (1) How to mine characteristics of different data types to quickly and accurately detect duplicate data while effectively reducing space overhead; (2) How to overcome limitations in similar data detection technology design, integrate various technical features, and combine statistical and data mining techniques to fully analyze and mine data characteristics, improving overall system performance through better understanding of data patterns; (3) How to leverage emerging compression theories, technologies, or more effective mathematical models to develop new technologies or combine existing ones to optimize storage space; (4) How to overcome limitations in storage overhead and system performance of existing reliability technologies based on redundancy addition, and appropriately increase redundancy or introduce other mechanisms to improve reliability for different data types; (5) How to make trade-offs between simplicity and performance when applying data deduplication technology, providing generality, scalability, and adaptability while minimizing system overhead from duplicate detection and elimination.

## References

[1] H. M. Sung, W. Y. Lee, J. Kim, and Y. W. Ko, Design and Implementation of Clustering File Backup Server Using File Fingerprint, Soft. Eng., Arti. Intel., Net. & Para./Distri. Comp., 2008, pp.61-73.

[2] C. Liu, D. Ju, Y. Gu, Y. Zhang and D. Wang, Semantic Data De-duplication for Archival Storage Systems, the 13th Asia-Pacific conference on Computer Systems Architecture Conference, Aug. 2008, pp.1-9.

[3] Y. Won, R. Kim, J. Ban, and J. Hur, PRUN: Eliminating Information Redundancy for Large Scale Data Backup System, Proceedings of the 2008 International Conference on Computational Sciences and Its Applications, 2008, pp 139-144.

[4] B. ZHU, H. LI, AND H. PATTERSON, Avoiding the disk bottleneck in

the data domain deduplication file system. In Proceedings of the 6th USENIX Conference on File And Storage Technologies (FAST' 08), San Jose, California, February 2008, pp. 1-14.

[5] S. Quinlan, S. Dorward. Venti: A new approach to archival storage. In Proceedings of the 2002 Conference on File and Storage Technologies (FAST), Monterey, California, USA, 2002, pp.89-101.

[6] L. L. You. Efficient Archival Data Storage. Techncial Report UCSC-SSRC-06-04, University of California, Santa Cruz, June 2006.

[7] L You, K Pollack and D. Long, Deep store: An archival storage system architecture, Proceedings of the 21st IEEE International Conference on Data Engineering, IEEE Press, Santa Cruz, CA, USA, 2005, pp. 804–815.

[8] S. Annapureddy, M. J. Freedman, and D. Mazières. Shark: Scaling file servers via cooperative caching. In Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05), Boston, MA, May 2005, pp. 129-142.

[9] A. Z. Broder. Identifying and filtering near-duplicate documents. In: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. Montreal, Canada: Springer-Verlag New York, Inc., Jun. 2000. 1-10.

[10] M. W. Storer, K. M. Greenan, D. D. E. Long, and E. L. Miller. Secure data deduplication. In Proceedings of the 2008 ACM Workshop on Storage Security and Survivability, Oct. 2008, pp 1-10.

[11] G. Forman, K. Eshghi, J. Suermondt, Efficient Detection of Large Scale Redundancy in Enterprise File Systems, HPL-2008-30R2, HP Laboratories, 2008.

[12] D. Trendafilov, N. Memon, and T. Suel. zdelta: An efficient delta compression tool. Technical Report TR-CIS-2002-02, Polytechnic University, June 2002.

[13] H. S. Gunawi, N. Agrawal, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and J. Schindler. Deconstructing commodity storage clusters. In Proceedings of the 32nd Int' l Symposium on Computer Architecture, June 2005, pp. 60–71.

[14] W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur. Single instance storage in Windows 2000. http://research.microsoft.com/farsite/WSS2000.pdf.

[15] M. O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[16] C. Liu, Y. Lu, C. Shi, and G. Lu, ADMAD: Application-Driven Metadata Aware De-duplication Archival Storage System, the 15th IEEE international workshop on Storage Network Architecture and Parallel I/O(SNAPI ' 08), 2008, pp.29-35.

[17] F. Douglis, A. Iyengar. Application-specific deltaencoding via resemblance detection. In Proceedings of the 2003 USENIX Annual Technical Conference, San Antonio, Texas, June 2003.

[18] L. Xu, Hydra: A Platform for Survivable and Secure Data Storage Systems, Proc. Int' l Workshop Storage Security and Survivability, Virginia, USA, Nov. 2005.

[19] J.J. Wylie, M.W. Bigrigg, J.D. Strunk, and G.R. Ganger. Survivable Information Storage Systems, IEEE Computer, 33(8), 2000: 61-68, Aug 2000.

[20] T. Yang, H. Jiang, D. Feng and Z. Niu. DEBAR: A Scalable High-Performance De-duplication Storage System for Backup and Archiving, Technical Report TR-UNL-CSE-2009-0004, HUST, 2008.

[21] 舒继武, 网络存储领域若干技术发展与启示, http://www.ccf.org.cn/web/resource/shujiwu.pdf, 2008.

[22] D. Bhagwat, K. Pollack, D. D. E. Long, and T. Schwarz, Providing High Reliability in a Minimum Redundancy Archival Storage System, Proceedings of the 4th ACM international workshop on Storage security and survivability, Virginia, USA, 2008, pp. 1-10.

[23] Chuanyi Liu, Yu Gu, Linchun Sun, Bin Yan, Dongsheng Wang: R-ADMAD: high reliability provision for large-scale de-duplication archival storage systems. ICS 2009: 370-379.

[24] Mark Lillibridge, Kave Eshghi, Deepavali Bhagwat, Vinay Deolalikar, Greg Trezise, and Peter Camble. Sparse indexing: large scale, inline deduplication using sampling and locality. In FAST ' 09: Proceedings of the 7th conference on File and storage technologies, page 111-123, Berkeley, CA, USA, 2009. USENIX Association.

[25] Deepavali Bhagwat, Kave Eshghi, Darrell D.E. Long, Mark Lillibridge. Extreme Binning: Scalable, Parallel Deduplication for Chunk-based File Backup. In IEEE MASCOTS 2009, London, UK, September, 21st, 2009.

## Author Biographies

**Wang Shupeng:** Ph.D., Head of Data Storage and Protection Group, Information Security Research Center, Institute of Computing Technology, Chinese Academy of Sciences. wangshupeng@software.ict.ac.cn

**Yun Xiaochun:** Professor, Institute of Computing Technology, Chinese Academy of Sciences

**Guo Li:** Director and Professor, Information Security Research Center, Institute of Computing Technology, Chinese Academy of Sciences

*Note: Figure translations are in progress. See original paper for figures.*