An Efficient Data Deduplication based on Tar-format Awareness in Backup Applications

Baegjae Sung, Sejin Park*, Youngsup Oh*, Jeonghyeon Ma*, Unsung Lee* and Chanik Park*
Department of Computer Science and Engineering, POSTECH, Pohang, South Korea
{jays,baksejin, youngsup, doitnow0415, dnsrjd1212, cipark}@postech.ac.kr
** student authors*

Abstract

Disk-based backup storage system is utilized widely, and data deduplication is becoming an essential technique in the system because of the advantage of a space-efficiency. Usually, user′s several files are aggregated into a single Tar file at primary storage, and the Tar file is transferred and stored to the backup storage system periodically (e.g., a weekly full backup) [1]. In this paper, we present a boundary-aware chunking (BAC) technique in a data deduplication backup storage system to improve space-efficiency by using a hint of the storing Tar file.

Chunking technique has the role of making chunks. Because chunks are basic granularity for detecting and eliminating duplicate data by checking the hash value of the chunk, space-efficiency of data deduplication system is varied by chunking technique. Static chunking (SC) [2] divides storing data into fixed size (e.g.., 8 KB) chunks. In other words, locations of every multiples of 8 KB of storing data are chunk boundary. In order to improve space-efficiency by solving boundary shifting problem, content-defined chunking (CDC) [3, 4] is proposed. CDC divides the storing data into variable size (e.g., expected size 8 KB) chunks. In other words, CDC technique choose proper chunk boundary by examining a content of the storing data.

BAC technique is an extension of CDC to choose more proper chunk boundary by using a hint of the storing Tar file. The Tar file consists of a sequence of header part and content part. Header part size is fixed 512 B and contains information of sub-file, such as file name, file size. Content part size is N * 512 B and contains content of sub-file. BAC can know end location of header part and end location of content part by extracting file size information from header part. We define these locations as absolute chunk boundary (ACB) that is additional chunk boundary. Basic chunking flow of BAC is similar with CDC except utilizing the ACB – Figure 1. Hash value f is generated (e.g., Rabin fingerprint) by content of small fixed size sliding window (e.g., 48 B) from storing data. If hash value f meets particular condition (e.g., f mod 8192 = 0) then the location of window is chunk boundary, otherwise window is

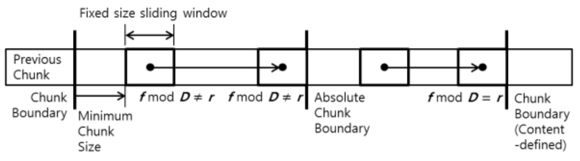

Fig. 1. Boundary-aware chunking technique

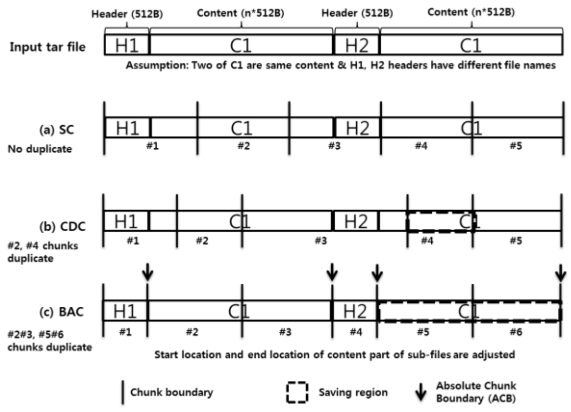
Header (512B)
Content (n*512B)
Header (512B)
Content (n*512B)
Input tar file
H1
C1
H2
C1
Assumption: Two of C1 are same content & H1, H2 headers have different file names
(a) SC
No duplicate
#1
#2
#3
#4
#5
(b) CDC
#2, #4 chunks duplicate
(c) BAC
#2#3, #5#6 chunks duplicate
#6
Start location and end location of content part of sub-files are adjusted
Chunk boundary
Saving region
Absolute Chunk Boundary (ACB)

Fig. 2. Chunking comparison of SC, CDC and BAC

shifted. If shifting window is reached to ACB then the location is chosen to chunk boundary even though hash value does not meet the condition. In order to prevent too small or too large chunk size, minimum and maximum chunk size restriction is utilized.

By using ACB for chunk boundary, start location and end location of sub-files are adjusted. It means first chunk included in a sub-file is always started from start location of sub-file regardless of any adding or removing or modification of previous sub-file. Likewise last chunk included in a sub-file is always ended to end location of sub-file regardless of any modification of current sub-file. Also, easily changeable header part is isolated from partially modifiable content part when chunk is created. In other words, BAC technique can redeem negative effect of adding or removing or modification of sub-files. We show a simplified example of BAC when storing input tar file including comparison of SC and CDC - Figure 2. We assume two files are aggregated into a tar file and the two files have same content. In (a) SC, five fixed-size chunks are generated. It does not find any duplicate chunks. In (b) CDC, it choose two chunk boundary on C1 content by examining content and generate five chunks. It finds a single duplicate chunk. In (c) BAC, it makes first chunk that contains header part by ACB, and it makes second chunk by examining content and third chunk by ACB under content part. Likewise other chunks are generated. It finds two duplicate chunks that contains more duplicate region than CDC.

To evaluate benefit of our BAC technique, we build upon the open-source Opendedup file system [5]. Because default chunking technique of Opendedup is SC, we modified the filesystem to utilize CDC technique or BAC technique. We use 8 KB expected chunk size, 2 KB minimum chunk size, and 14 KB maximum chunk size. Tested backup system is built upon Intel Xeon E5620 2.40GHz (2xQuad core), 16 GB memory, and 7200 RPM SATA disk. To eliminate network interference, we copy tar backup file from local ext4 file system to local deduplication file system. We tested a linux-20-dist workload that storing a single tar file that is aggregated different 20 versions of Linux distributions. The workload has similar characteristics of several full backup.

Deduplication ratio (Input size/Stored size) shows a space-efficiency of data deduplication system. We measured the result of deduplication ratio using linux-20-dist workload with SC, CDC and BAC – Figure 4. Deduplication ratio of BAC shows 7 times higher than SC, and 3 times higher than CDC. It shows benefit of BAC′s redemption of negative effect of adding or removing or modification of sub-files of a tar file. Write throughput also measured using linux-20-dist workload. We add ext2 naive file system write throughput for comparison. Ext2 and SC have almost same throughput even though SC has chunking overhead. We think the reason comes from parallel 20 threads write in Opendedup filesystem. Throughput of BAC and CDC shows half of SC. Interesting point is BAC is little higher than CDC, because actual writing chunk data to disk is 3 times lower than CDC even though BAC has more operational overhead such as reading file size from header part, more chunks.

BAC data deduplication file system is ongoing work. We need to evaluate our BAC technique using various workloads. After detailed evaluation, we plan to optimize the BAC data deduplication file system. In this paper, we concentrated on only Tar file format for extracting ACB hint. However, BAC can be applied to other formats such as VM image, ISO. We also plan to apply BAC technique for these formats.

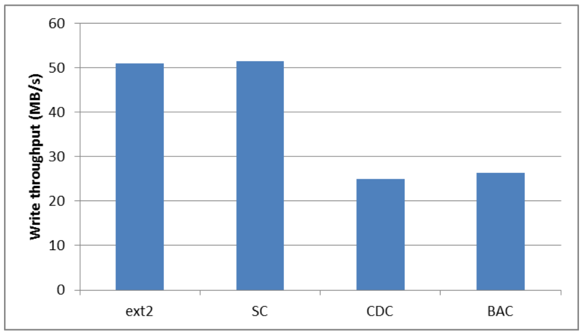

Fig. 3. Deduplication ratio

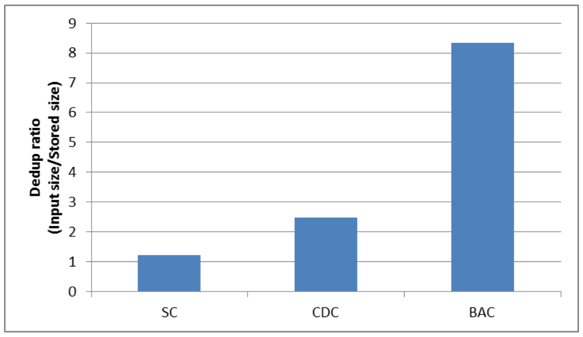

Fig. 4 write throughput

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MEST) (No. 2011-0016972) and IT Consilience Creative Program of MKE and NIPA (C1515-1121-0003)

References

[1] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characeiristics of Backup Workloads in Production Systems", In FAST'12: Proceedings of the 10th USENIX conference on File and Storage Technologies, 2012.

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

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

[4] F. Guo and P. Efstathopoulos, "Building a highperformance deduplication system", In Proceedings of the 2011 USENIX conference on USENIX Annual Technical Conference, 2011.

[5] Opendedup, http://opendedup.org/