Efficient Unstructured Data Compression for Block Storage Systems

Hiroki Ohtsuji
Fujitsu Laboratories Ltd.
Kawasaki, Japan

Yoshiyasu Doi
Fujitsu Laboratories Ltd.
Kawasaki, Japan

I. Introduction

Data compression is a key technology to reduce the amount of data in large-scale storage systems. HPC applications mainly use well-structured data formats; however, there are still unstructured data sets in storage systems. Well-structured data can be compressed in a sophisticated manner, which can avoid degrading the I/O performance. The rest unstructured data sets will be compressed by file systems or underlying block storage systems. From a view point of storage systems, it is difficult to obtain the structural information of written data blocks. If the block storage system compresses them individually, there is a limitation in terms of the data compression ratio. Using larger data chunk size is an option to improve the compression ratio, at the same time, it also brings the I/O performance degradation problem, which is caused by the read amplification. This paper discuss a dynamic chunking method to optimize the size of compressed blocks to avoid I/O performance degradation.

II. I/O Performance degradation: Read amplification

The read amplification problem occurs when the application requires only a fraction of compressed data blocks. Typically, this problem is caused by random and stride access patterns. Fig. 1 describes the mechanism of the read amplification problem. The compression algorithm uses the reference pointers to previously appeared chunks to reduce the data length. Therefore, the storage system have to read the entire compress data block and decompress them to access a single data block. This behavior increases the amount of read access and degrades the both latency and bandwidth. In order to avoid this performance degradation, we have to split a group of consecutive written data blocks which are expected to have less relevance.

III. Design and Implementation

Fig. 2 describes the implementation of the target system. We modified the structure of the write cache page table to implement the dynamic chunking mechanism. The additional column is "write time stamp", which stores the time stamps of the write I/O operations. The system regards the write operations as independent if there is a time gap between two I/O operations. Independent written blocks will be individually compressed into compressed blocks. In addition, there is a mechanism to monitor read accesses in order to adjust the compression chnunk size (not described in the figure).

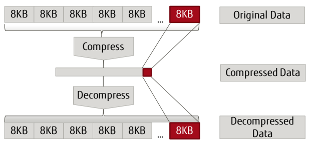
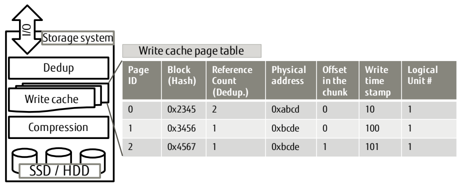


Fig. 1. Read amplification: Reading an entire compressed block is required to decompress only a single block.



Page ID	Block (Hash)	Reference Count (Dedup.)	Physical address	Offset in the chunk	Write time stamp	Logical Unit #
0	0x2345	2	0xabcd	0	10	1
1	0x3456	1	0xbcde	0	100	1
2	0x4567	1	0xbcde	1	101	1

Fig. 2. A storage system and the write cache page structure for the dynamic chunking mechanism.

IV. Evaluation

Fig. 3 depicts the preliminary evaluation result of the compression ratio with LZ4 [1] algorithm. If we use 64 KB compressed chunk size instead of 4 KB, we can improve the compression ratio by 17 percentage points. The read amplification is expected to be eliminated in the case of Fig. 4 because the method can split write data blocks into appropriate chunks.

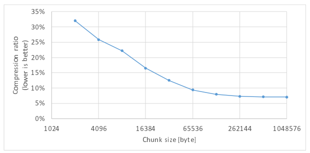


Fig. 3. Compression ratio: X-axis is the size of compressed blocks. Y-axis is the compression ratio.

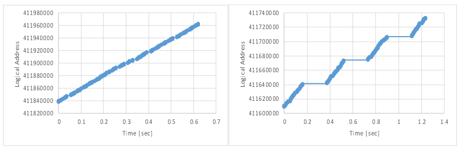


Fig. 4. An example of write trace log: Contiguous (left) and non-contiguous (right)

V. Conclusion and future work

This paper described the dynamic chunking method for the data compression mechanism in block storage systems. The method can improve the compression ratio while preventing the I/O performance degradation incurred by unnecessary read operations. Future work is evaluating the method with real workload with the complete implementation of the system.

References

[1] Yann Collet. Lz4: Extremely fast compression algorithm, 2013.