On Selective Compression of Primary Data

Gabriel Alatorre, Nagapramod Mandagere, Yang Song and Heiko Ludwig
IBM Almaden Research Center, San Jose, CA
Email: (galatorr,pramod,yangsong,hludwig)@us.ibm.com

***Abstract*—With the advent of social media, Internet of Things (IoT), widespread use of richer media formats such as video, and generally increased use of mobile devices, volume of online data has seen a rapid increase in recent years. To cope with this data explosion, businesses and cloud providers are scrambling to lower the cost of storing data without sacrificing the quality of their service using space reduction techniques such as compression and deduplication. Capacity savings, however, are achieved at the cost of performance and additional resource overheads. One drawback of compression techniques is the additional computation required to store and fetch data, which may significantly increase response time, i.e., I/O latency. Worse yet, inefficient compression algorithms that fail to compress data satisfactorily suffer from the latency penalty with marginal capacity savings, e.g., deciding to compress data that is encrypted or already compressed. Therefore, from a data center administrator's perspective, we should pick the set of volumes that will yield the most compression space saving with the least latency for a given amount of computation capacity, without exhaustively inspecting the data content of volumes. To fill this void, this paper proposes an approach to manage compression for a very large set of volumes. It maximizes capacity savings and minimizes latency impact without scanning the actual data content (to avoid security concerns). Our pilot deployments show significant capacity savings and performance improvements compared to benchmark compression strategies.**

I. Introduction

With hyper connectivity, pervasive computing, Internet of Things (IoT), we are experiencing an ever increasing rate of data generation in social media, commerce, finance, and many other industries & domains. In 2014, according to Domo, email users sent 200 million messages, Facebook users shared 2.5 million pieces of content, and Instagram users posted nearly 220,000 photos *every minute* [14] — and this is just social media. We live in a world where every click, phone call, financial transaction, etc. is logged. Businesses and cloud providers must find ways to store these massive (and continuously growing) amounts of data in a manner that lowers the cost rate to themselves and their clients while also not sacrificing the quality of their service.

Currently, many effective solutions exist such as storage tiering, deduplication, compression, the use of magnetic tape, or the use of commodity storage — in many cases, best results are achieved when employing a combination of these solutions (e.g. deduplication then compression [10]). Space reduction techniques such as compression and deduplication can help improve operational efficiency by significantly reducing the data footprint. Capacity savings though come at the cost of performance and additional resource overheads. Focusing on compression, we found it widely deployed on inactive & infrequently accessed data as well as on particular types of data such as multimedia (e.g. photos, videos, etc.) but application to active data in production data centers was practically non-existent. We found the idea of deploying compression across all data, both active and inactive, having the potential to yield immense storage capacity savings.

With recent advancements in real-time compression algorithms, we explored how well real-time compression fared when deployed on both active and inactive data. We explored using IBM's Real-time Compression [19] since it benchmarked well when compared to other industry solutions [16]. IBM's Real-time Compression operates by compressing associated data, accessed at approximately the same time, together into chunks. For example, a set of project files typically opened together would be compressed within the same chunk. One drawback of real-time compression is that it is computationally more expensive to read and write compressed data, response times may significantly increase, which brings to light the commonly seen fragile balance between space savings and performance — one is typically attained at the cost of the other. Real-time compression turns out not to be impervious to these issues. Further, compression of certain types of data such as encrypted data and multimedia files that tend to use some internal form of compression do not yield any capacity savings, however, on the flip side consume resources and add to overall latency of access.

Because of limited computation resources, data center storage devices generally have a hard limit to the number of data volumes that can be compressed. This additional constraint requires any compression deployment solution to not only compress the optimal subset of volumes at a given point in time but to maintain the optimal subset of volumes compressed over time (e.g. a compressed volume with high capacity savings today may provide poor savings in the future).

In this paper we present an effective approach to both small and large-scale deployment of data compression in a cloud or enterprise data center in a manner that yields significant capacity savings without incurring the heavy performance cost. We formulate this as an optimization problem and propose a system that efficiently finds and compresses a near optimal subset of volumes over time. Our solution finds and compresses volumes with high compression ratios using a heuristic-based compression prediction algorithm. Over time, it uncompresses volumes in favor of others yielding better capacity savings. The remainder of this paper is organized as follows, Section II provides background on compression, Section III highlights the motivation and challenges for online compression, Section IV presents our proposed heuristic based selective compression approach, Section V provides results of initial evaluation and that of a pilot deployment, Section VI provides summary and future directions.

978-3-901882-77-7

II. Background

One of the first data compression techniques proposed was by Shannon et al.[17] and Fano et al.[12]. Symbols were sorted by frequency and encoded using a growing number of bits so more frequently seen symbols were encoded with less bits. Building on this work, Huffman et al.[13] proposed a more efficient encoding strategy that minimized the average output bits per symbol. The dawn of modern data compression came in 1977 when Abraham Lempel and Jacob Ziv proposed a technique for encoding repeating text patterns [21][22]. Building on this work, Welch et al.[20] proposed a more efficient technique that currently serves as the basis of many modern compression algorithms including the real-time algorithm used in our work.

In modern cloud computing, compression is traditionally used for infrequently accessed data (such as archived files, snapshots, & backups), multimedia, and to reduce the size of data transmission. For example, Nasuni appliances leverage compression to shrink snapshots and data transmission [6]. Omnicloud, a company focused on secure but not high performance storage, compresses data before encrypting & storing [7]. Cloud services such as Amazon Web Services, Google Cloud Storage, and Microsoft Azure enable users to manually compress files, snapshots, and backups but lack automated compression as well as real-time compression solutions for primary data [2][4][5]. Enterprise storage providers such as EMC, NetApp, and HP provide compression but advocate compressing backup and secondary but not primary data [8][15][9].

III. Motivation & Challenges

During our initial investigation into the use and deployment of compression within a data center, we discovered numerous challenges as well as how current deployment methods failed to address them. In this section, we list and explain each of these challenges.

When initially deploying the use of compression in a data center, data volumes that fail to produce significant capacity savings or exhibit significantly higher response latency when compressed should remain uncompressed. Typical deployment methods such as compressing data volumes that are accessed infrequently, stored in a particular storage pool, or belonging to a particular account fail to guarantee either goal.

The next challenge is efficiently transforming data volumes (from uncompressed to compressed and vice versa) since the transformation is not instant but, in fact, consumes computation, storage, and bandwidth resources over an extended period of time. When transforming a data volume, the volume must remain accessible without any loss of data. Volume transformation can take minutes (for a small volume) to hours (for a relatively large volume) so taking a volume offline is not an acceptable solution for most applications. If volume transformation involves having the volume partially uncompressed and partially compressed at any point in time, any form of device failure can result in complete data loss (or partial data loss if a backup of the data volume is taken before transformation began).

Once a volume is transformed, if compression failed to produce adequate results (i.e. significant space savings and low latency), the volume should be rolled back and remain uncompressed. Transforming volumes back to uncompressed should similarly be non-disruptive and result in no data loss. Observing that volume transformation consumes resources, accurately predicting the data volumes that compress well would drastically reduce overall resource usage during compression deployment. Compression prediction itself is made difficult by the fact that sampling (e.g. method used by NetApp's Space Savings Estimation Tool [15] and IBM's Comprestimator [3]) in many cases is not permissible due to security concerns (e.g. medical records, personal financial data, etc.).

In data centers, the use of compression is typically limited by the virtualization device(s) due to limited computation resources [19]. An additional challenge then is to compress the volume set resulting in the greatest cost savings (or capacity savings if all storage is priced the same). Many data centers typically have different tiers of storage ranging from high performance, mid-performance, and near-line with each tier varying in cost — typically the higher the performance capabilities the higher the cost. Compression of volumes on the higher cost, high performance devices can result in significantly sizeable cost savings but cloud providers are typically hesitant to compress data on these devices due to concerns over significant performance degradation. This scenario illustrates the need for a compression solution that can compress data volumes across the different tiers of storage in a manner that maximizes cost savings while ensuring acceptable performance.

Once compression has been deployed, there is the issue that data volumes change over time. Data volumes that had yielded high capacity savings and acceptable performance may no longer do so after some time. The compression solution deployed should include the discovery and uncompression of these data volumes. On the other end, there may now be volumes eligible for compression that initially did not meet the capacity savings or performance requirements. The compression solution should also include the discovery and compression of these data volumes.

In one data center we investigated, the policy to compress data volumes found to have low activity (i.e. low read & write I/O access) was deployed. Data volumes that had averaged 1.5ms response times rose to 4.5ms — a 290% increase! Of the 550 compressed volumes roughly 20% yielded no capacity savings — the performance cost was incurred with no benefit in return.

IV. Selective Real-time Compression

Our proposed Selective Real-time Compression system (referred to as Real-time Compression Automation (RTCA) tool) is designed with goal of optimal selection of data for compression. In this section, we describe how RTCA addresses the concerns in the previous section to efficiently, non-disruptively, and safely deploy data compression within a data center.

RTCA is implemented as a web service (deployed on a WebSphere Application Server Liberty Profile [11]) and uses a lightweight embedded database (Apache Derby [1])

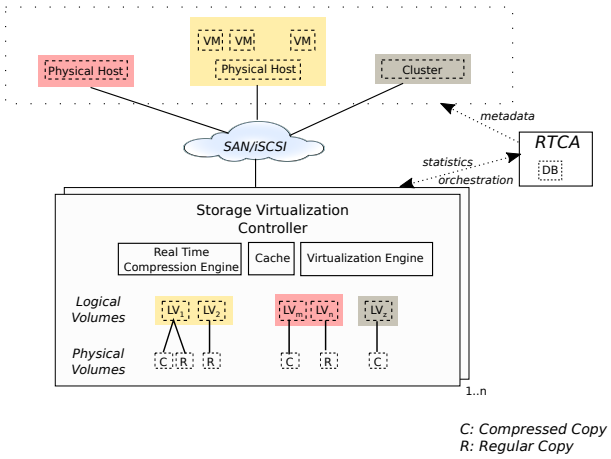

Fig. 1. Selective Real-time Compression (RTCA): System Model

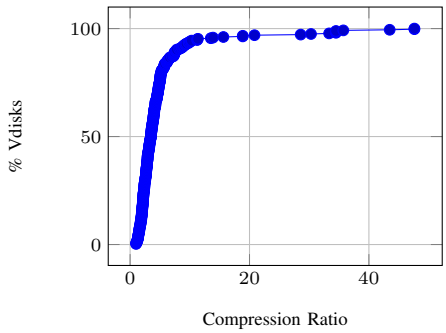

Fig. 2. Cumulative distribution of compression ratios among volumes within a data center

for persistently storing job status and execution history as shown in figure 1. RTCA connects to the storage virtualization device(s) within a data center to retrieve host & storage configuration data as well as to compress and uncompress data volumes. Since storage virtualization devices enable the non-disruptive transformation of data volumes from uncompressed to compressed (and vice versa), they are a key component in any online compression deployment solution.

Once a volume is compressed by RTCA, the storage virtualization device services I/O requests by decompressing the appropriate data chunks on reads and updating the appropriate data chunks on writes; making data compression invisible to applications running on the host devices. RTCA operates by having a user execute a job. Since an RTCA job uses computation, storage, and bandwidth resources, it is typically run during a maintenance or low-peak usage window. If the job does not complete within the time window, it can be run again during the next window. Since volume data changes over time (e.g. a compressed data volume yielding high capacity savings today may not do so in the future), an RTCA job can be executed periodically (e.g. daily, weekly, monthly, etc.) to maintain the optimal set of volumes compressed within the data center. An RTCA job is composed of the following three phases: sampling, planning, and execution. We now discuss each in turn.

1) Sampling: The sampling phase is only executed when RTCA is run for the first time in a data center. Sampling is executed to jump start the compression prediction algorithm RTCA uses to predict the capacity savings of a particular volume. Since sampling is resource intensive, we first looked into the range of compression ratios (uncompressed size / compressed size) of 359 volumes (accounting for 27.4TB) in two data centers to verify whether compression prediction was necessary. For example, if volumes skewed to having relatively high compression ratios then no compression sampling and prediction scheme is necessary since compressing the largest volumes would yield the greatest cost savings (assuming all storage is priced the same).

Plotting the compression ratios of the production data center volumes displayed a wide spectrum of results — some volumes showed no compressibility while others showed up to a compression ratio of 50 (i.e. on average every 50GB compressed down to 1GB). A wide distribution of compression ratios necessitated finding a method to predict compression ratios.

In analyzing the configuration (e.g. volume size, volume to host mapping, etc.) & performance (e.g. average read and write I/O requests per second) data of volumes within two production data centers, we found a correlation among the compression ratios of volumes mapping to the same host. This led to the hypothesis that applications generally stored similar data across most or all of their data volumes. To verify this hypothesis we looked at the variance of compression ratios among three volume sets: (1) volumes mapping to the same host, (2) similarly sized volumes, and (3) all volumes.

			Variance	
	Groups	Average Group Size	Average	Median
Host	51	12	20.0	2.3
Size	5	72	35.7	16.4
Overall	1	359	40.2	4.7

Table 1: *Variance among volumes grouped by host or size as well as all volumes*

As seen in Table 1, the average variance among host volumes is 20.0 (44% and 50% less than that of similarly sized volumes and all volumes, respectively). The median variance among host volumes is 2.3 (86% and 52% less than that of similarly sized volumes and all volumes, respectively).

RTCA's compression prediction algorithm leverages this host heuristic to predict the compression ratio of a given volume by averaging the known compression ratios of any other volumes mapping to the same host. To validate this prediction algorithm, we again used the compression ratio information of the 359 volumes. We measured the compression ratio differences of every pair of volumes that belonged to the same host (6,544 pairs in our study), denoted by vector A, and the compression ratio differences of every pair of volumes that belonged to different hosts (107,326 pairs in our study), denoted by vector B. The mean values of A and B were 3.8 and 4.4. The medians of A and B were 1.5 and 1.8, respectively. To evaluate the statistical significance of such differences, we performed unpaired Mann-Whitney U test on A and B where the alternative hypothesis was that the population mean of A is less than the population mean of B. Our test rejects the null

hypothesis at a significance level of 5% (confidence 95%), which corroborates our heuristic that the compression ratio differences of two volumes with the same host are usually smaller than those across different hosts.

When RTCA is first run in a data center, there are no known compression ratios so there is insufficient information to make predictions. To address this cold start problem, we execute a one-time sampling phase. During the sampling phase, a randomly selected volume from each host is compressed, their compression ratio is logged, and they are subsequently rolled back (to uncompressed).

2) Planning: Having compression ratio information, RTCA moves on to the planning phase where it predicts the compression ratio of every volume in the data center. It proceeds by selecting the subset of volumes with known or predicted compression ratios above a user-specified threshold and ordered by expected cost savings (cost/GB * Expected[capacity savings]), from highest to lowest.

3) Execution: In the final execution phase, RTCA starts from the top of the volume list and begins to transform (compress) data volumes — those with the highest expected cost savings are compressed first. To control data center resource usage, the user can limit the number of volume transformations running in parallel. To compress a volume, RTCA creates a compressed copy of the volume. Once the original and compressed copies are in sync, RTCA removes the original copy. This process requires up to the size of the volume in additional capacity for a second copy (worst case) and doubles the work of writes (writes are propagated to both copies) during the length of the process but it does guarantee no data loss, non-disruptive compression (and uncompression), and instant roll-back. If a volume's compression ratio was predicted inaccurately and falls below the acceptable threshold, it is instantly rolled back (compressed copy deleted).

During the execution phase, RTCA also finds volumes with compression ratios below the acceptable threshold and transforms them back to uncompressed. Optionally, RTCA can retrieve performance data from a monitoring program and transform compressed volumes exhibiting unacceptable response times back to compressed. To avoid re-compressing these volumes in subsequent RTCA jobs, RTCA tracks each volume's transformation history and feeds it into the compression prediction algorithm to improve further predictions and recommendations.

V. Evaluation

We deployed RTCA in a 190TB production data center consisting of 1,287 volumes virtualized by IBM's SAN Volume Controller [18], an in-band virtualization device, where compression (IBM's Real-time Compression [19]) was already in use. Compression had been deployed following the policy of compressing volumes found to have relatively low activity (i.e. low read & write I/O access). Following this policy, 551 volumes were compressed yielding 28TB of capacity savings (an average of a 40% capacity reduction per volume) but of the 551 compressed volumes roughly 20% yielded no capacity savings. Compressed volumes also experienced an increase in average response time from 1.5 to 5.8ms (approx. 290% increase).

We ran a daily RTCA job to uncompress volumes with a compression ratio below and to compress volumes predicted to have ratios above 1.7. Jobs ran during low peak hours (i.e. evening & night). RTCA jobs were limited to execute at most 20 transformations in parallel with each transforming a volume at 64MB/s leading to the use of at most 1.25GB/s of bandwidth during execution. With the average size of a volume being 151GB, there was an additional use of 3TB of storage capacity on average during execution. Negligible changes in volume response times suggest computation resources were not over-utilized or exhausted during the transformation windows.

	Low Activity Policy	RTCA
Compressed Count	551	381
Average Compression Ratio	1.7	3.3
Average Response Time (ms)	5.8	1.8
Total Savings (GB)	28,101	37,563

Table 2: *Comparison between low activity policy & RTCA*

After a week of daily RTCA jobs, the data center stabilized and volume transformations ceased. By compressing a more optimal set of volumes, RTCA reduced overall computation utilization, compressed fewer volumes (i.e. 551 to 386), reduced the average compressed volume response time (i.e. 5.8ms to 1.8ms), and increased capacity savings by 36% (i.e. 28TB to 38TB).

VI. Conclusion

In this paper, we first investigate the feasibility of deploying compression techniques in large scale data centers. Our study shows that poorly selecting data volumes to compress could result in significantly increasing response time latency while yielding marginal capacity savings. Moreover, limited computation resources allow only the compression of a limited number of volumes so selecting the right volumes to compress is of great importance from a data center administrator's standpoint. In addition, the transformation of volumes was a process that needed to be executed carefully to not lose data, waste resources, or disrupt actively running applications. To address these challenges, we developed RTCA which is a tool for deploying compression within a data center that used a host heuristic based compression prediction algorithm to compress a near optimal set of volumes. We validate the effectiveness of RTCA by deploying it in a 190TB production data center where compression was already in use. Our study shows that RTCA was able to compress fewer volumes thus reducing overall computation utilization, reduce the average compressed volume response time, and increase capacity savings.

Acknowledgment

The authors would like to thank Kevin McQuillan, Jim Olson, Dave Schustek, Laura Richardson, Ted Fay, and Sandeep Gopisetty for their insightful guidance, suggestions, and support throughout the execution of this work. Also special thanks to Nicolas Andre Druet, Dave Kodjo, and Lucy Kung for helping with development set up and testing.

REFERENCES

[1] Apache derby. http://db.apache.org/derby/. Accessed: June 2015.

[2] Best practices for using amazon s3. https://aws.amazon.com/articles/1904. Accessed: August 2015.

[3] Comprestimator utility version 1.5.2.2. http://www-304.ibm.com/webapp/set2/sas/f/comprestimator/home.html. Accessed: August 2015.

[4] Google cloud platform - performance. https://cloud.google.com/storage/docs/json_api/v1/how-tos/performance. Accessed: August 2015.

[5] Microsoft azure. https://azure.microsoft.com. Accessed: August 2015.

[6] The nasuni filer. http://www.nasuni.com/product/the-nasuni-filer/. Accessed: August 2015.

[7] Omnicloud: Secure and flexible use of cloud storage services. https://www.sit.fraunhofer.de/fileadmin/dokumente/Projektblaetter/OmniCloud_EN_FraunhoferSIT.pdf. Accessed: August 2015.

[8] Emc vnx deduplication and compression. http://www.emc.com/collateral/hardware/white-papers/h8198-vnx-deduplication-compression-wp.pdf, July 2012. Accessed: August 2015.

[9] Hp storeall storage with iternity compliant archive solution. http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA3-8113ENW&cc=us&lc=en, June 2014. Accessed: August 2015.

[10] Julian Cates. Storage efficiency in clustered data ontap. http://community.netapp.com/t5/Tech-OnTap-Articles/Storage-Efficiency-in-Clustered-Data-ONTAP/ta-p/86822, September 2013. Accessed: August 2015.

[11] Brent Daniel, Alex Mulholland, Erin Schnabel, and Kevin Smith. *WebSphere Application Server Liberty Profile Guide for Developers*. International Technical Support Organization, 2 edition, August 2013.

[12] Robert M Fano. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics, 1949.

[13] David A Huffman et al. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[14] Josh James. Data never sleeps 2.0. http://www.domo.com/blog/2014/04/data-never-sleeps-2-0/, April 2014. Accessed: June 2015.

[15] Sandra Moulton and Carlos Alvarez. Netapp data compression and deduplication. http://www.netapp.com/us/media/tr-3958.pdf, February 2014. Accessed: August 2015.

[16] Craig Norris, Barry Cohen, and Manny Frishberg. A comparative evaluation of block storage compression technology for multipurpose storage environments. http://www.theedison.com/pdf/2012_Samples_IBM_V7000_Block_Compression.pdf, July 2012. Accessed: August 2015.

[17] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[18] Jon Tate, Frank Enders, Torben Jensen, Hartmut Lonzer, Libor Miklas, and Marcin Tabinowski. *Implementing the IBM System Storage SAN Volume Controller V7.4*. International Technical Support Organization, 4 edition, April 2015.

[19] Jon Tate, Bosmat Tuv-El, Jorge Quintal, and Christian Burns. *IBM Real-time Compression in SAN Volume Controller and Storwize V7000*. International Technical Support Organization, 2 edition, March 2015.

[20] Terry A. Welch. A technique for high-performance data compression. *Computer*, 6(17):8–19, 1984.

[21] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[22] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530–536, 1978.