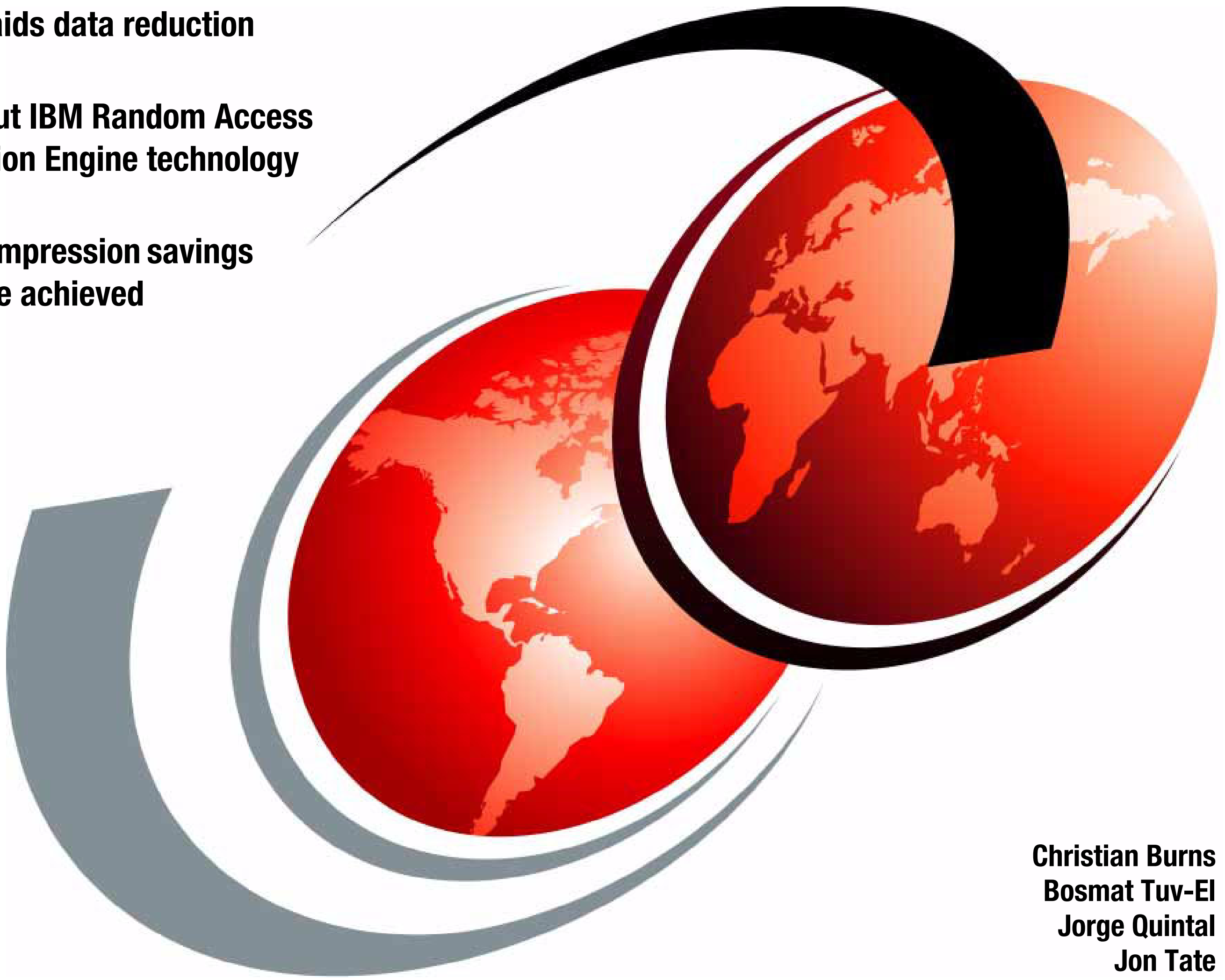
IBM Real-time Compression in IBM SAN Volume Controller and IBM Storwize V7000

Discover how embedded compression software aids data reduction

Learn about IBM Random Access Compression Engine technology

See the compression savings that can be achieved



Christian Burns
Bosmat Tuv-El
Jorge Quintal
Jon Tate

Redpaper

ibm.com/redbooks

International Technical Support Organization

IBM Real-time Compression in IBM SAN Volume Controller and IBM Storwize V7000

March 2015

REDP-4859-01

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

Second Edition (March 2015)

This edition applies to those hardware and software products as detailed in the paper only.

Contents

Notices

Trademarks

Preface

IBM® Real-time Compression™ software that is embedded in IBM SAN Volume Controller (SVC) and IBM Storwize® V7000 solution addresses all the requirements of primary storage data reduction, including performance, by using a purpose-built technology called *real-time compression*. This IBM Redpaper™ publication addresses the key requirements for primary storage data reduction and gives real world examples of savings that can be made by using compression.

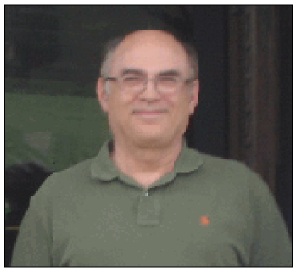
SVC and Storwize V7000 is designed to improve storage efficiency by compressing data by as much as 80% through supported real-time compression for block storage. This process enables up to five times as much data to be stored in the same physical disk space. Unlike other approaches to compression, IBM Real-time Compression is used with active primary data, such as production databases and email systems. This configuration dramatically expands the range of candidate data that can benefit from compression. As its name implies, IBM Real-time Compression operates as data is written to disk, avoiding the need to store data that is awaiting compression.

Authors

This paper was produced by a team of specialists from around the world working for the IBM International Technical Support Organization in Tel Aviv, Israel.

Christian Burns, at the time of writing, was an IBM Storage Solution Architect based in New Jersey. As a member of the Storage Solutions Engineering team in Littleton, MA, he worked with clients, IBM Business Partners, and IBMers worldwide, designing and implementing storage solutions that included various IBM products and technologies. Christian's areas of expertise include IBM Real-time Compression, SVC, IBM XIV®, and IBM FlashSystem™. Before joining IBM, Christian was the Director of Sales Engineering at Storwize, before its becoming IBM Storwize. He has over a decade of industry experience in the areas of sales engineering, solution design, and software development. Christian holds a BA degree in Physics and Computer Science from Rutgers College. Christian has since left to pursue a career outside of IBM.

Bosmat Tuv-El is a Manager of Development Support for IBM Real-time Compression in Israel. Bosmat has 10 years of IT, QA, and support experience in storage systems and networking. She joined IBM through the acquisition of Storwize in 2010. She manages the worldwide product support team for IBM Real-time Compression that provides analysis of complex customer problems and works to improve the overall customer experience through product improvements and documentation. Bosmat graduated from the Open University of Israel with a BA in Computer Science and Management.



Jorge Quintal is a Storage Managing Consultant currently providing Development Support for IBM Real-time Compression. He joined IBM through the acquisition of Sequent Computer Systems in 1999. During his time at IBM, he has worked for Storage Lab Services as one of the original members working with SAN File System, SVC, and NAS. He was lead for N-Series services development and implementations. Jorge also worked for an extended period as an IBM XIV Technical Advisor.

Jon Tate is a Project Manager for IBM System Storage® SAN Solutions at the International Technical Support Organization (ITSO), San Jose Center. Before joining the ITSO in 1999, he worked in the IBM Technical Support Center, providing Level 2 support for IBM storage products. Jon has 29 years of experience in storage software and management, services, and support, and is both an IBM Certified IT Specialist and an IBM SAN Certified Specialist. He is also the UK Chairman of the Storage Networking Industry Association.

Thanks to the authors of the previous version of this paper:

Bosmat Tuv-El
Jorge Quintal
Eyal Traitel
Barry Whyte

Thanks to the following people for their contributions to this project, and the previous version of this paper:

Eric Bartlett
John Fairhurst
Robin Findlay
Stuart Jones
John Lindley
Andy Martin
Craig McAllister
Paul Merrison
Simon Swindells
IBM Hursley, UK

Amir Lidor
Navin Manohar
Nir Rigai
Tzahi Shahak
Ori Bauer
IBM Israel

Thomas Vogel
IBM Germany

Bruce McNutt
Chris Saul
IBM US

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Summary of changes

This section describes the technical changes that are made in this edition of the paper and in previous editions. This edition also might include minor corrections and editorial changes that are not identified.

Summary of Changes
for IBM Real-time Compression in IBM SAN Volume Controller and IBM Storwize V7000
as created or updated on May 16, 2018.

March 2015, Second Edition

This revision includes the following new and changed information.

New information

This edition is updated to document the many changes to the hardware and software of the SAN Volume Controller (SVC) and Storwize V7000 family.

Changed information

The values that are associated with compression are updated to encompass the latest hardware and software changes.

1

Overview

Continued data growth and economic pressures are driving the rapid adoption of data reduction technologies. Although much of the attention around data reduction has focused on backup, many businesses are applying data reduction throughout the entire data lifecycle.

IBM Real-time Compression Software that is embedded in IBM System Storage SAN Volume Controller (SVC), IBM Storwize V7000, and IBM FlashSystem V840 addresses all the requirements of primary storage data reduction, including performance. It does so by using a purpose-built technology called real-time compression. This publication addresses the key requirements for primary storage data reduction.

This chapter describes the current challenges of data growth and addresses how to overcome these challenges by using compression.

This chapter includes the following sections:

- Current IT challenges
- The solution: IBM Real-time Compression
- Common use cases
- IBM Real-time Compression

1.1 Current IT challenges

Businesses and organizations around the world are challenged with tough economic conditions. The IT environment, which historically was viewed as an expense, is now viewed as a source of innovation that can drive future revenue. However, ever increasing data storage requirements consume the available resources and disrupt attempts to innovate the IT environment.

The modern IT department has these challenges:

- Support for increasing data storage requirements: Shrinking IT budgets are pressuring IT managers to increase the lifetime of existing storage systems. Traditional methods of cleanup of unneeded data and archival of files to auxiliary storage are time consuming. They shift one resource constraint, physical storage, to another: The human work of storage administrators.
- Power, cooling, and floor space: A data center provides the means to host the storage systems. However, the physical characteristics of hard disk drive-based systems limit the amount of data that can be stored per rack unit. High-power consumption and heat dissipation are major concerns for IT managers who must fit the storage systems into a limited data center. This conflicts with the increasing demand for computing power that is needed to support new types of applications.
- High availability of data: Digital information has become the basis for any service in use today. As a result, the underlying systems that provide access to digital information are expected to be online all the time. This requirement has made it impossible to introduce data reduction solutions that impose any type of downtime. This restriction is true whether it is an actual inability to access the data, or merely a major slowdown when accessing an optimized data set.

Compression of primary storage provides an innovative approach that is designed to overcome these challenges.

1.2 The solution: IBM Real-time Compression

The IBM Real-time Compression solution addresses the challenges that are listed in the previous section because it was designed from the ground up for primary storage. Implementing IBM Real-time Compression in SVC, Storwize V7000, or FlashSystem V840 provides the following benefits:

- Compression for active primary data: IBM Real-time Compression can be used with active primary data. Therefore, it supports workloads that are not candidates for compression in other solutions. The solution supports online compression of existing data. It allows storage administrators to regain free disk space in an existing storage system without requiring administrators and users to clean up or archive data. This configuration enhances the value of existing storage assets, and the benefits to the business are immediate. The capital expense of upgrading or expanding the storage system is delayed.
- Compression for replicated/mirrored data: Remote volume copies can be compressed in addition to the volumes at the primary storage tier. This process reduces storage requirements in Metro Mirror and Global Mirror destination volumes as well.

- No changes to the existing environment are required: IBM Real-time Compression is part of the storage system. It was designed with transparency in mind so that it can be implemented without changes to applications, hosts, networks, fabrics, or external storage systems. The solution is not apparent to hosts, so users and applications continue to work as-is. Compression occurs within the Storwize V7000, FlashSystem V840, or SVC system itself.
- Overall savings in operational expenses: More data is stored in a rack space, so fewer storage expansion enclosures are required to store a data set. This reduced rack space has the following benefits:
  - Reduced power and cooling requirements: More data is stored in a system, therefore requiring less power and cooling per gigabyte or used capacity.
  - Reduced software licensing for additional functions in the system: More data stored per enclosure reduces the overall spending on licensing.

  Tip: Implementing compression in SVC, Storwize V7000, and FlashSystem V840 provides the same benefits to externally virtualized storage systems.

- Disk space savings are immediate: The space reduction occurs when the host writes the data. This process is unlike other compression solutions in which some or all of the reduction is realized only after a post-process compression batch job is run.

1.3 Common use cases

This section addresses the most common use cases for implementing compression:

- General-purpose volumes
- Databases
- Virtualized infrastructures

Explanation: The expected compression ratios are based on samples that are taken from that industry. Real-life ratios can vary from the stated values, and depend on individual customers' environments and data structures.

For more information about expected compression savings and candidate data types, see Chapter 4, "Planning" on page 25.

1.3.1 General-purpose volumes

Most general-purpose volumes are used for highly compressible data types, such as home directories, CAD/CAM, oil and gas data, and log data. Storing such types of data in compressed volumes provides immediate capacity reduction to the overall consumed space. More space can be provided to users without any change to the environment.

There can be many file types that are stored in general-purpose servers. However, for practical information, the estimated compression ratios are based on actual field experience. Expected compression ratios are 50% - 60%.

File systems that contain audio, video files, and compressed files are not good candidates for compression. The overall memory savings on these file types are minimal.

1.3.2 Databases

Database information is stored in table space files. It is common to observe high compression ratios in database volumes. Examples of databases that can greatly benefit from real-time compression are IBM DB2®, Oracle, and Microsoft SQL Server. Expected compression ratios are 50% - 80%.

Attention: Some databases offer optional built-in compression. Generally, do not compress already compressed database files.

1.3.3 Virtualized infrastructures

The proliferation of open systems virtualizations in the market has increased the use of storage space, with more virtual server images and backups kept online. The use of compression reduces the storage requirements at the source.

Examples of virtualization solutions that can greatly benefit from real-time compression are VMWare, Microsoft Hyper-V, and KVM. Expected compression ratios are 45% - 75%.

Tip: Virtual machines with file systems that contain compressed files are not good candidates for compression, as described in 1.3.1, “General-purpose volumes” on page 3.

1.4 IBM Real-time Compression

The IBM Random Access Compression Engine (RACE) technology was first introduced in the IBM Real-time Compression Appliances. It is now integrated into the SVC, Storwize V7000, and FlashSystem V840 software stack. The following chapters focus on the benefits of IBM Real-time Compression, and describe how to plan for and configure compression. In addition, the reporting of compression savings, preferred practices, and troubleshooting tools are addressed.

2

Compression technology

This chapter provides a general overview of data compression and data efficiency technologies.

This chapter includes the following sections:

- ▸ Compression technology history
- ▸ Data efficiency technologies
- ▸ Data compression technologies

2.1 Compression technology history

For the last 100 years, IT technology has evolved from large systems that were built to run simple mathematical operations. Now, small devices can run, generate, and manipulate massive amounts of data. Devices that can store and capture data are growing exponentially from year to year. Data must be stored for a long time for long-term reference, compliance, and security purposes. Therefore, new ways to optimize the capacity utilization are needed. As data requirements have grown, the technologies that are available to reduce the amount of data stored or transported from one place to another have greatly improved.

One of the first methods for reducing the amount of data was the use of symbols and representations in mathematical format. For example, instead of writing the words "multiplied by," the related representation that is used is the asterisk character (*). In the same way, the word "minus" is represented with the dash character (-).

In 1838, the invention of Morse code allowed messages to be transmitted quickly over long distances. Roman letters and Arabic numbers were replaced with symbols formed from lines and dots. To reduce the number of dots or lines that is used to represent each letter, statistical analysis of the commonality of letters was performed. The most common letters are represented with a shorter combination of dots and lines. The commonality is different for each language, as is the Morse code. For example, in the English language, the letter "s" is represented in the Morse code by three dots, whereas the letter "h" is represented by four dots. The representation therefore consists of seven dots. However, in some languages "sh" is a common combination, so the dots were replaced by lines. In this scheme, "sh" is represented by four lines, effectively saving transmission time.

Later in the 20th century, the development of IT technologies raised the need for complex algorithms that reduce the amount of data. This compression is done by interpreting the information beyond the simple substitution of specific strings or letters.

One of the first techniques of mathematical data compression was proposed by Claude E. Shannon and Robert Fano in 1949. In the Shannon-Fano coding, symbols are sorted from the most probably to the least probable, and then encoded in a growing number of bits. If the source data contains A B C D E, where A is the most common and E is the least common letter, the Shannon-Fano coding is 00-01-10-110-111.

In 1952, a Ph.D. student at MIT named David A. Huffman proposed a more efficient algorithm for mapping source symbols to unique string of bits. In fact, Huffman proved that his coding is the most efficient method for this task, with the smallest average output bits per source symbol.

Later, Abraham Lempel and Jacob Ziv in 1977 proposed a method of replacing repeating words with code words. The method was applicable also to a pattern of text, such as expressions. This was the actual dawn of modern data compression. In 1984, Terry Welch improved the algorithm proposed by Lempel and Ziv (also known as LZ78) and developed a method that is known as LZW. Today, this algorithm is the basis of modern compression techniques that are used in PKZIP for general file compression, or within GIF and TIFF formats for images.

Over time, many data compression algorithms have been developed around the Lempel-Ziv method:

- LZ - Storer-Szymanski (LZSS)
- LZ with Arithmetic encoding (LZARI)
- LZ + Huffman (LZH) encoding, which is used by the ARJ utility

The IBM Real-time Compression Appliance™ also uses compression based on LZH. For more information about both algorithms, see the following links:

Details about Lempel-Ziv coding can be found at the following websites:

- Lempel-Ziv explained:

  http://www-math.mit.edu/~shor/PAM/lempel_ziv_notes.pdf
- Lempel-Ziv coding:

  http://www.code.ucsd.edu/cosman/NewLempel.pdf

Details about Huffmann coding can be found at the following websites:

- Huffmann coding explained:

  http://www.cs.cf.ac.uk/Dave/Multimedia/node210.html
- Detailed Huffman coding:

  http://homepage.smc.edu/kennedy_john/HUFFMAN.PDF

Software Version 7.2 introduced the implementation of LZ4 compression algorithm as part of the IBM Random Access Compression Engine (RACE). This new compression algorithm delivers significant performance improvement to many compressed workloads, especially sequential ones, and also reduces CPU utilization.

2.2 Data efficiency technologies

Data compression technologies can be found in various implementations over the last 15 years. These range from compression at the application level to in-band solutions that can compress data as it is transferred from a host to a storage device. Recently, many new technologies and new concepts have been developed to address the need for optimized storage space and more efficient capacity usage. Some of these technologies are addressed briefly in this chapter to avoid confusion between data compression and other space optimization methods.

2.2.1 Thin provisioning technology

Thin provisioning enables the storage to present the required capacity to the host while allocating only the actual used capacity in terms of space on the physical storage media. Figure 2-1 presents the difference between a traditional volume and a thin-provisioned volume.

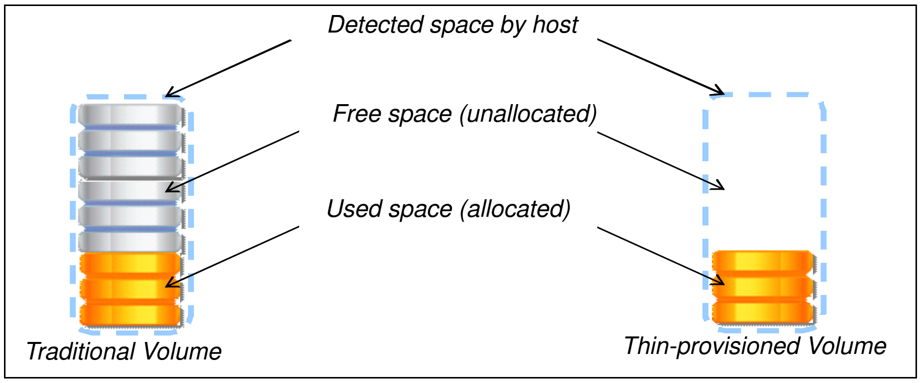


Figure 2-1 Traditional volume and thin-provisioned volume

2.2.2 FlashCopy (thin-provisioned snapshots)

IBM FlashCopy® is a technique that allows the multiplication of volume data and data instances without multiplying the required space with the number of copies. This technology allows the copy to depend on the source with all the data that is in common. When data is changed from the original volume, those changes are available and reflected only in the updated instance. The rest of the data remains as a common foundation.

2.2.3 IBM Easy Tier

IBM Easy Tier® is a technology that is optimizing the way data is placed across different types of drives (FC/SAS, SATA/SAS_NL, and SSDs). In environments without Easy Tier, the storage system is configured by using a rule set. Those rules are known by a specialist and they usually refer to the result that is expected after configuring a storage system. The hardest part is to address, in a price/performance way, the capacity needs over the performance needs.

Usually, when you need either capacity or performance, you add drives. There are many situations when you must add drives to accommodate a specific performance, but those drives are not needed for capacity. In these situations, the infrastructure is oversized in terms of capacity.

IBM Easy Tier (Figure 2-2) is a feature that can provide an optimized balance between capacity and performance. It provides balance by identifying specific hot zones and by moving them from slower drives to faster drives automatically. You can use a combination of solid-state drives (SSDs) with much slower but higher capacity SATA drives, instead of using many faster drives to accommodate the load.

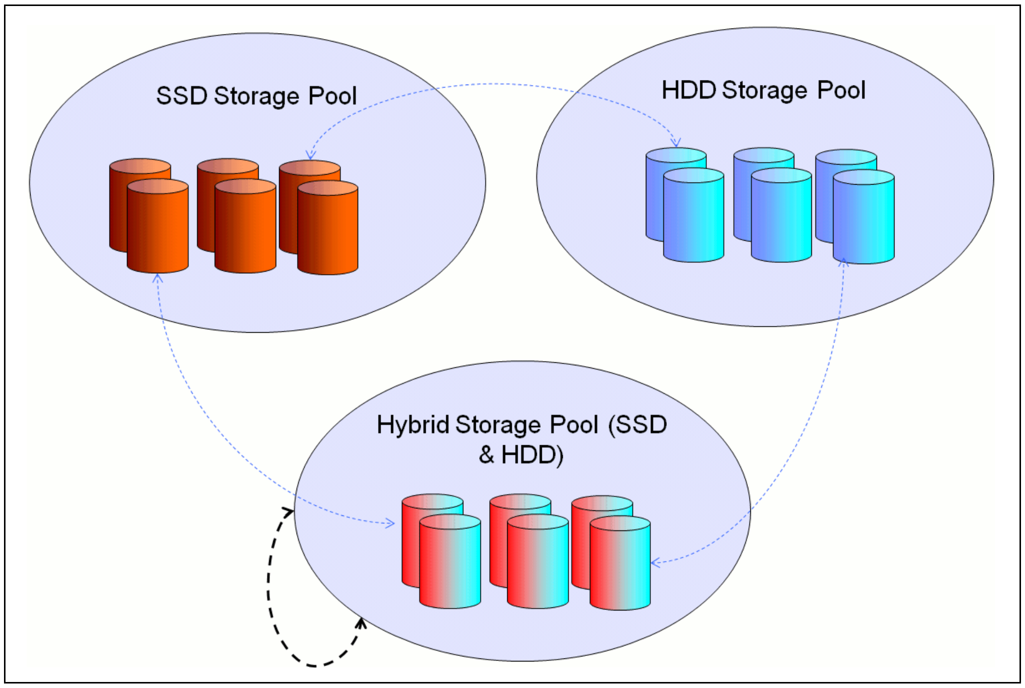


Figure 2-2 Easy Tier data allocation

2.2.4 Archiving and space management

Archiving and space management, also known as information lifecycle management (ILM), is a concept rather than a dedicated technology solution. It can be used in environments where you need long-term storage that optimizes the space that is occupied on different storage types, such as tapes and hard disk drives (HDDs). The old or rarely used data can be moved out of the production area and placed on much cheaper support. That is, moving data from enterprise storage systems to low-end storage systems, and from HDDs to tapes. The historical information is usually kept in a database that holds metadata to provide accurate information about the managed data.

2.2.5 Data deduplication

The data deduplication mechanism identifies identical chunks of data within a storage container and keeps only one copy of each chunk. All the other logically identical chunks are pointed to this chunk. There are various implementations of this method:

- In-line deduplication
- Post-processing

In-line solutions (Figure 2-3) deduplicate information as it is stored. Post-processing solutions deduplicate data after the information is stored in the original format at certain time intervals.

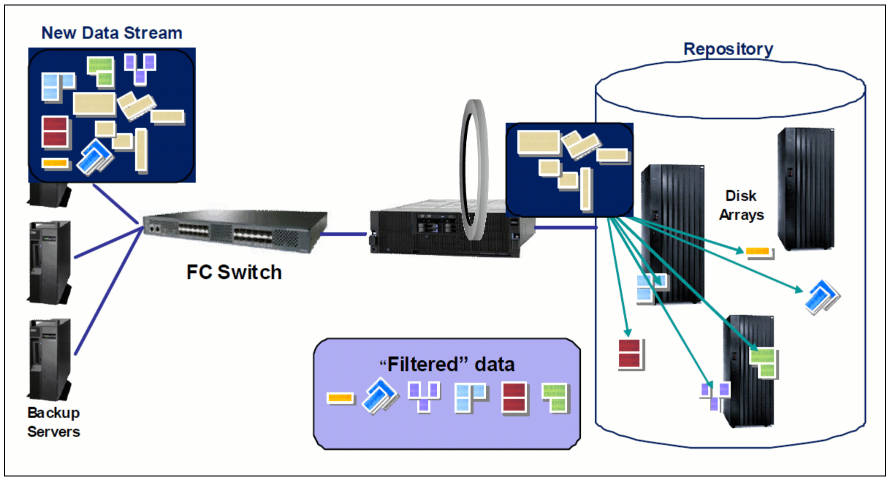


Figure 2-3 Data deduplication

Comparison: All the technologies that are presented in 2.2, "Data efficiency technologies" on page 7 are used to optimize the way that the available storage capacity is used. None of those technologies change information, but rather optimize how this information is stored or how much of this information is duplicated within the storage system. Data compression technologies can interpret data. They can provide storage optimization, both regarding the amount of data that is stored and the performance that is needed for the storage system that is used in back-end systems.

2.3 Data compression technologies

The compression of data has rapidly become a focus for the IT industry. Because of the different types of data and the reasons why data is compressed, two major compression methods are used:

- Lossless data compression: This method allows the information to be rebuilt completely with no effect on the quantity or quality of the original information.
- Lossy data compression: This method synthesizes the information and keeps only the data that is needed. The original information cannot be rebuilt completely to its original form when the data is extracted.

Examples of lossless data compression include audio, image, video compression, reports, and graphics that are generated to visualize large amounts of data and statistics. For example, the data might be runners at the Olympic Games who manage to go under 10.1 seconds in the 100-meter race. The system might retain the data that 35% of the runners managed to go under 10.1 seconds. However, this data does not include the times of individual runners.

In this way, lossless data compression offers the advantage of completely and accurately re-creating the input information. In comparison, the irreversible method offers only some specific information that is related to the original information.

Because of the massive amounts of data and calculations that are necessary for compressing data, there are two approaches:

- Real-time compression: This method processes the data before it is written to the storage device. The key advantage of this approach is that it reduces the storage resources that are required for a data set. If done correctly, the capacity-reduction application preserves the inherent performance of the storage environment. Already optimized data is written to storage, which mitigates the capacity explosion challenge at the point of origin. It accomplishes this mitigation by eliminating the need to allocate the additional storage capacity that is required by post-processing solutions. Because the primary storage is used, any compression technique must be run in real time and maintain the high availability features of the existing storage system.
- Post-processing optimization: These solutions eliminate the need to deal with the performance issues in real time, and usually do not have any advanced high availability capabilities. The challenge with post-processing optimization is that it uses storage resources for the capacity-reduction application, which requires significant storage I/O resources. Post-processing solutions require a full read of the original data. They then scan and compress the data writing out a new compressed copy before deleting the original data.

Over the years, IBM has introduced a series of lossless, real-time compression algorithms and solutions that are used in wide range of technologies:

- The LTO-DC algorithm that is used in IBM Linear Tape Open, formally known as LTO tape drives.
- The Streaming Lossless Data Compression (SLDC) algorithm that is used in IBM Enterprise-class TS1130 tape drives.
- The Adaptive Lossless Data Compression (ALDC) that is used by the IBM Information Archive for its disk pool collections.
- The RACE that is used inside Storwize V7000, SAN Volume Controller (SVC), and Real-time Compression Appliance.

3

IBM Real-time Compression

This chapter introduces the IBM Real-time Compression technology, which is fully embedded into the software stack of IBM SAN Volume Controller (SVC), IBM Storwize V7000, and IBM FlashSystem V840. It addresses the basics of the IBM Random Access Compression Engine (RACE) technology. RACE is seamlessly integrated into the existing software stack in a fully transparent way. This integration does not alter the behavior of the system, so previously existing features are supported for compressed volumes.

This chapter includes the following sections:

- Overview
- IBM Real-time Compression
- Software and hardware updates that enhance Real-time Compression

3.1 Overview

The industry need for data compression is for it to be fast, reliable, and scalable. The compression algorithm that is used must ensure data consistency and a good compression rate to be implemented. In addition, the data compression solution must also be easy to implement. The compression must occur without affecting the production use of the data at any time.

Based on industry requirements, the best model was chosen for the IBM Real-time Compression support. This is a combination of a lossless data compression algorithm with a real-time compression technology.

The RACE technology is the core of the IBM Real-time Compression solution, and is now available in these IBM storage systems:

- SVC
- Storwize V7000 Gen2
- FlashSystem V840

Tip: For simplicity, throughout this chapter we refer to these supported systems as "RtC-supported system" or "RtC-supported systems". In cases where information is provided that does not apply to all of these systems, we indicate as much.

Note: To use the IBM Real-time Compression feature on the FlashSystem V840, two compression accelerator cards are required. To use the IBM Real-time Compression feature on the SAN Volume Controller 2145-DH8 node, at least one compression accelerator card is required. For more information, see "Intel Quick Assist Acceleration Technology (Coletto Creek) compression acceleration cards" on page 23.

RACE technology is based on over 40 patents that are not about compression. Rather, they define how to make industry-standard LZ compression of primary storage operate in real-time and allow random access. The primary intellectual property behind this is the RACE engine. At a high level, the RACE component compresses data that is written into the storage system dynamically. This compression occurs transparently, so Fibre Channel and iSCSI connected hosts are not aware of the compression. RACE is an in-line compression technology, meaning that each host write is compressed as it passes through the RtC-supported system software to the disks. This has a clear benefit over other compression technologies that are post-processing based. These technologies do not provide immediate capacity savings, and therefore are not a good fit for primary storage workloads, such as databases and active data set applications.

3.2 IBM Real-time Compression

To understand the basic design of the IBM Real-time Compression technology, you must understand the basics of modern compression techniques, which include the Lempel-Ziv algorithm and Huffman coding.

RACE is based on the Lempel-Ziv lossless data compression algorithm that operates in a real-time method. When a host sends a write request, it is acknowledged by the write cache of the system, and then staged to the storage pool. As part of its staging, it passes through the compression engine, and is then stored in compressed format onto the storage pool. Writes are therefore acknowledged immediately after received by the write cache, with compression occurring as part of the staging to internal or external physical storage.

Capacity is saved when the data is written by the host because the host writes are smaller when written to the storage pool.

IBM Real-time Compression is a self-tuning solution, similar to the RtC-supported system itself. It is adapted to the workload that runs on the system at any particular moment.

3.2.1 Random Access Compression Engine

To understand why RACE is unique, you must review the traditional compression techniques. This description is not about the compression algorithm itself, that is, how the data structure is reduced in size mathematically. Rather, the description is about how it is laid out within the resulting compressed output.

Compression utilities

Compression is probably most known to users because of the widespread use of compression utilities such as zip and gzip. At a high level, these utilities take a file as their input, and parse the data by using a sliding window technique. Repetitions of data are detected within the sliding window history, most often 32 KB. Repetitions outside of the window cannot be referenced. Therefore, the file cannot be reduced in size unless data is repeated when the window "slides" to the next 32 KB slot.

Figure 3-1 shows compression that uses a sliding window.

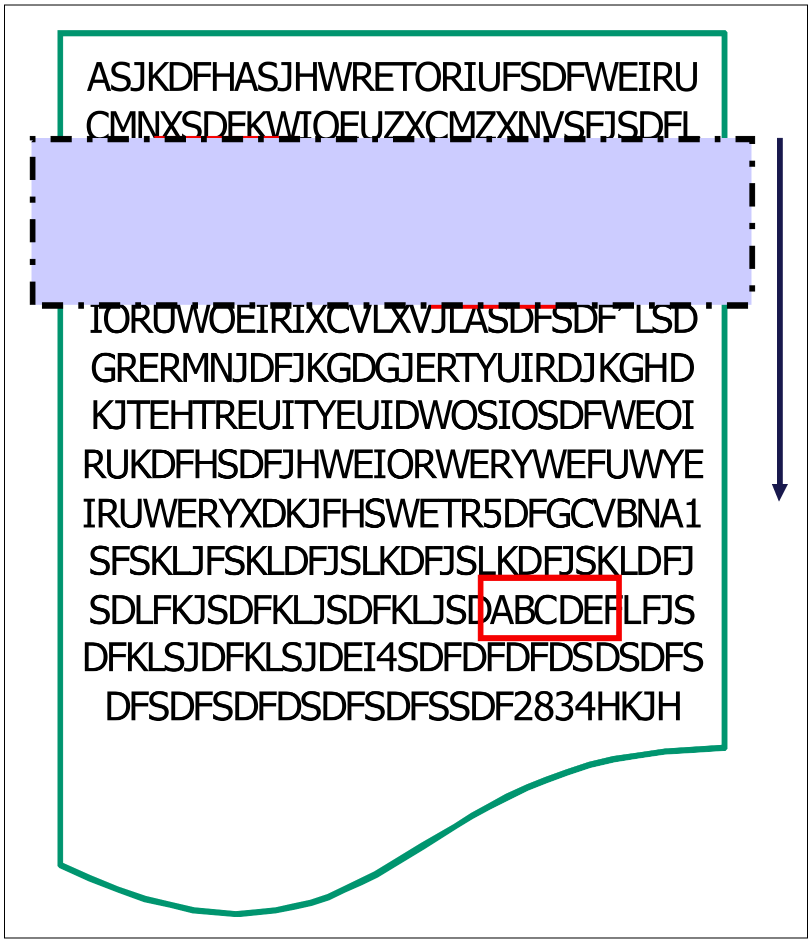

ASJKDFHASJHWRETORIUFSDFWEIRU
CMNXSDFKWIOEUZXCMZXNVSFJSDFL
IORUWOEIRIXCVLXVJLASDFSDF LSD
GRERMNJDFJKGDGJERTYUIRDJKGHD
KJTEHTREUITYEUIDWOSIOSDFWEOI
RUKDFHSDFJHWEIORWERYWEFUWYE
IRUWERYXDKJFHSWETR5DFGCVBNA1
SFSKLJFSKLDFJSLKDFJSLKDFJSKLDFJ
SDLFKJSDFKLJSDFKLJSDABCDEFLFJS
DFKLSJDFKLSJDEI4SDFDFDFDSDSDFS
DFSDFSDFDSDFSDFSSDF2834HKJH

Figure 3-1 Compression that uses a sliding window

Traditional data compression in storage systems

The traditional approach that is taken to implement data compression in storage systems is an extension of how compression works in the compression utilities previously mentioned. Similar to compression utilities, the incoming data is broken into fixed chunks, and then each chunk is compressed and extracted independently.

However, there are drawbacks to this approach. An update to a chunk requires a read of the chunk followed by a recompression of the chunk to include the update. The larger the chunk size that is chosen, the heaver is the I/O penalty to recompress the chunks. If a small chunk size is chosen, the compression ratio is reduced because the repetition detection potential is reduced.

Figure 3-2 shows an example of how the data is broken into fixed size chunks (in the upper-left side of the figure). It also shows how each chunk is compressed independently into compressed chunks (in the upper-right side of the figure). The resulting compressed chunks are stored sequentially in the compressed output.

Although this approach is an evolution from compression utilities, it is limited to low performance use cases. This limitation is mainly because it does not provide real random access to the data.

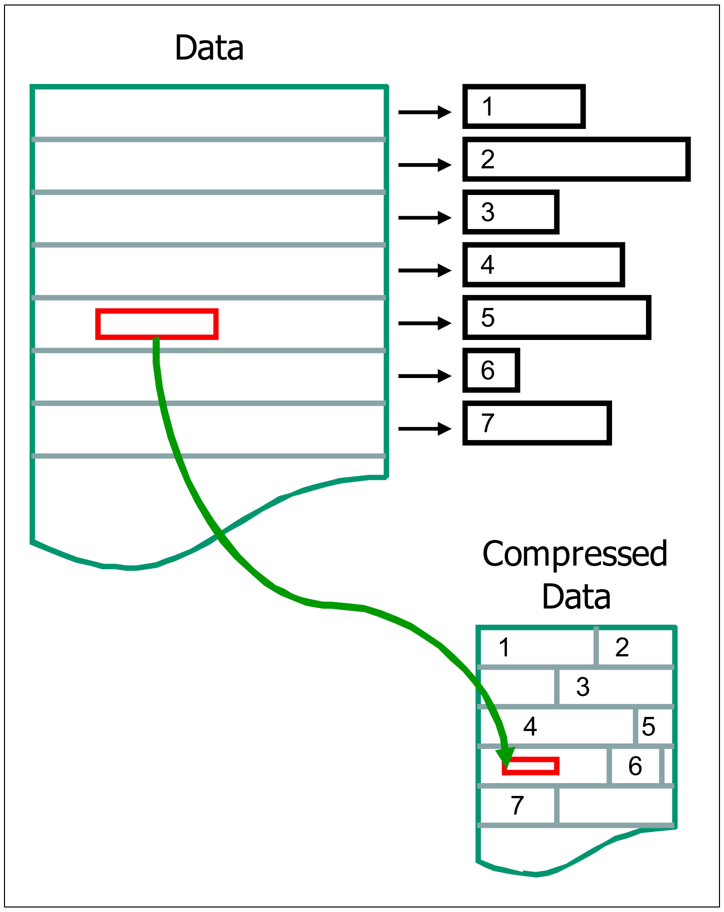


Figure 3-2 Traditional data compression in storage systems

Random Access Compression Engine

The RACE turns over the traditional approach to compression. It uses variable-size chunks for the input, and fixed-size chunks for the output.

This method enables an efficient and consistent method to index the compressed data because it is stored in fixed-size containers.

Figure 3-3 shows Random Access Compression.

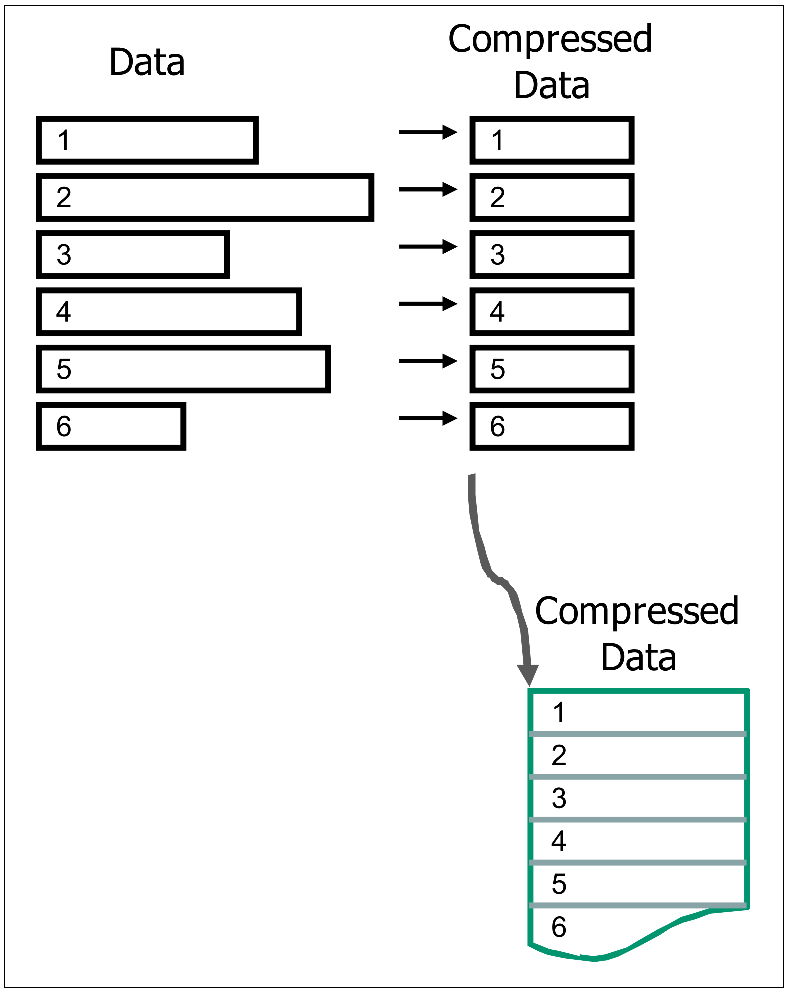


Figure 3-3 Random Access Compression

3.2.2 Location-based compression

Both compression utilities and traditional storage systems compression compress data by finding repetitions of bytes within the chunk that is being compressed. The compression ratio of this chunk depends on how many repetitions can be detected within the chunk. The number of repetitions is affected by how much the bytes stored in the chunk are related to each other. The relationship between bytes is driven by the format of the object. For example, an office document might contain textual information, and an embedded drawing (such as this page). Because the chunking of the file is arbitrary, it has no notion of how the data is laid out within the document. Therefore, a compressed chunk can be a mixture of the textual information and part of the drawing. This process yields a lower compression ratio because the different data types mixed cause a suboptimal dictionary of repetitions. That is, fewer repetitions can be detected because a repetition of bytes in a text object is unlikely to be found in a drawing.

This traditional approach to data compression is also called *location-based compression*. The data repetition detection is based on the location of data within the same chunk.

This challenge was also addressed with the *predecide* mechanism that was introduced in Version 7.1.

Predecide mechanism

Some data chunks have a higher compression ratio than others. Compressing some of the chunks saves a little space, but still requires resources such as CPU and memory. To avoid spending resources on uncompressible data, and to provide the ability to use a different, more effective (in this particular case) compression algorithm, IBM has invented a predecide mechanism that was first introduced in software Version 7.1.

The chunks that are below a given compression ratio are skipped by the compression engine, thus saving CPU time and memory processing. Chunks that are determined to not be compressed with the main compression algorithm, but that still can be compressed well with the other, are marked and processed accordingly. The result might vary because predecide does not check the entire block, only a sample of it.

Figure 3-4 shows how the detection mechanism works.

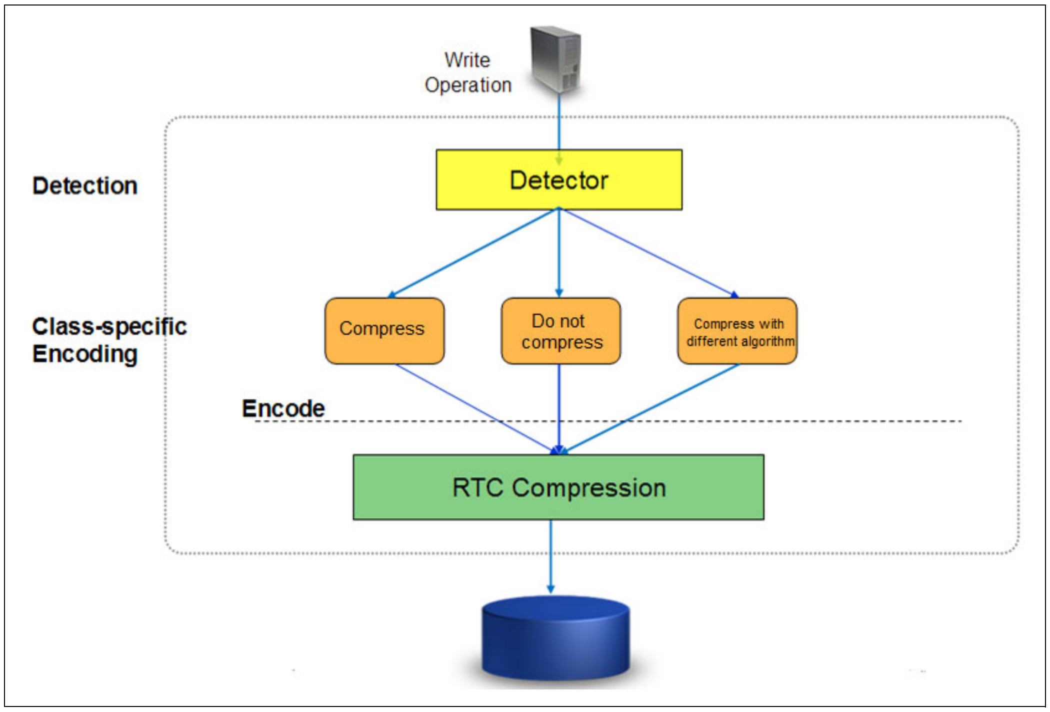


Figure 3-4 Detection mechanism

3.2.3 Temporal compression

RACE offers a technology leap beyond location-based compression: temporal compression.

When host writes arrive at RACE, they are compressed and fill up fixed size chunks that are called *compressed blocks*. Multiple compressed writes can be aggregated into a single compressed block. A dictionary of the detected repetitions is stored within the compressed block. When applications write new data or update existing data, it is typically sent from the host to the storage system as a series of writes. Because these writes are likely to originate from the same application and be from the same data type, more repetitions are detected by the compression algorithm.

This type of data compression is called *temporal compression* because the data repetition detection is based on the time the data was written into the same compressed block. Temporal compression adds the time dimension that is not available to other compression algorithms. It offers a higher compression ratio because the compressed data in a block represents a more homogeneous input data.

Figure 3-5 shows (in the upper part) how three writes that are sent one after another by a host end up in different chunks. They are compressed in different chunks because their location in the volume is not adjacent. This yields a lower compression ratio because the same data must be compressed non-natively by using three separate dictionaries. When the same three writes are sent through RACE (in the lower part of the figure), the writes are compressed together by using a single dictionary. This yields a higher compression ratio than location-based compression.

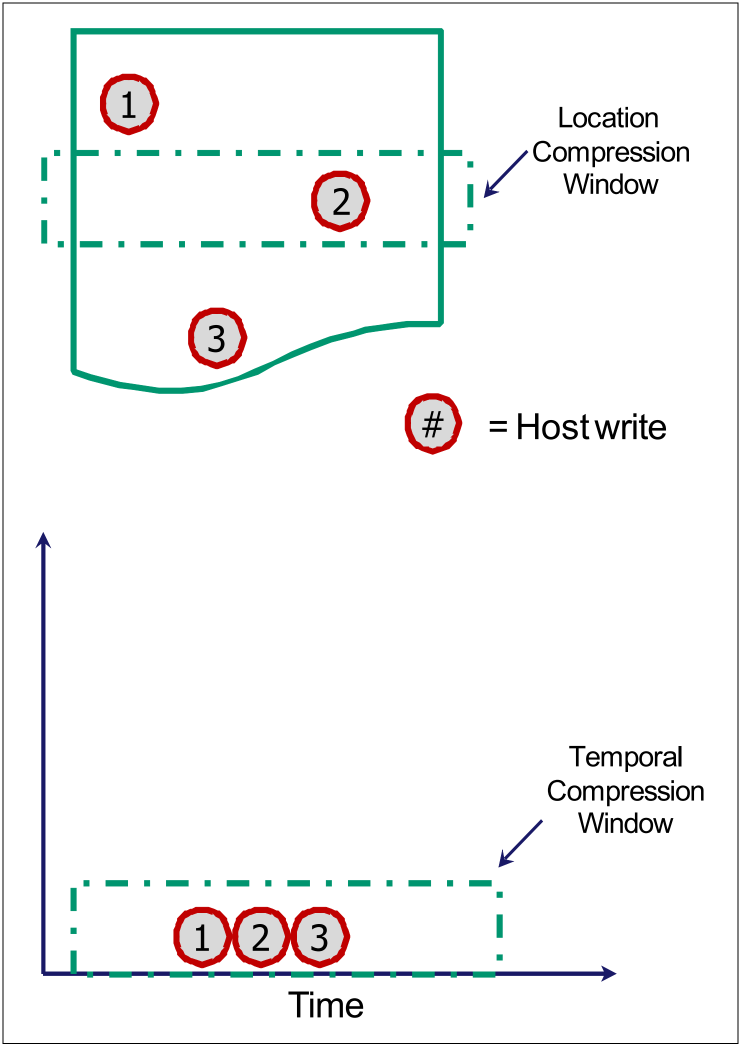


Figure 3-5 Location-based versus temporal compression

3.2.4 RACE in the SAN Volume Controller software stack

It is important to understand where the RACE technology is implemented in the software stack of the RtC-supported system. This location determines how it applies to system components and features.

RACE technology is implemented into the thin provisioning layer, and is an organic part of the stack. The software stack is shown in Figure 3-6 on page 19.

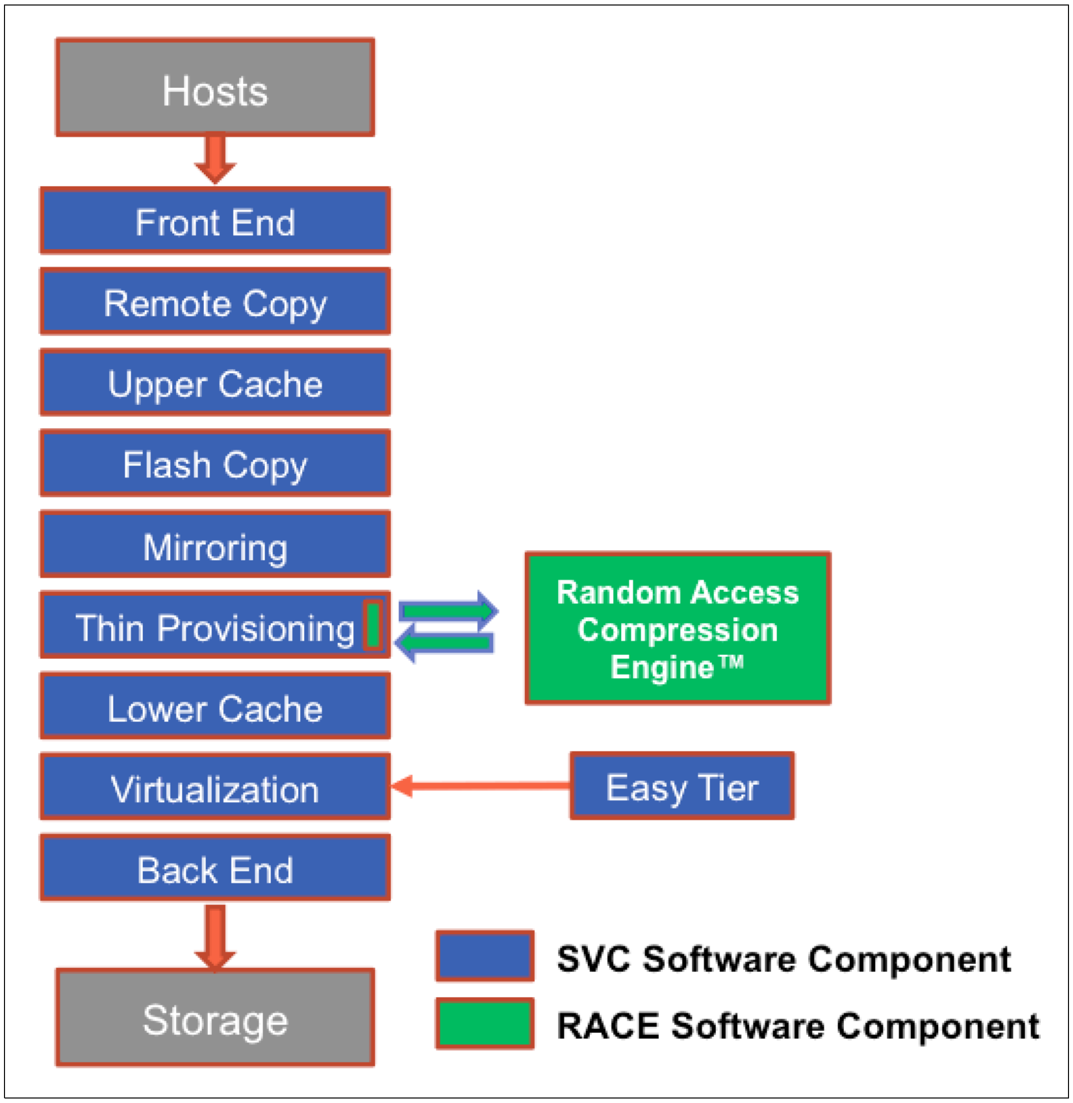


Figure 3-6 SAN Volume Controller V6.4.0 software stack

Compression is transparently integrated with existing system management design. All of the advanced features of the RtC-supported system are supported on compressed volumes. You can create, delete, migrate, map (assign), and unmap (unassign) a compressed volume as though it were a fully allocated volume. In addition, you can use IBM Real-time Compression along with IBM Easy Tier on the same volumes. This compression method provides nondisruptive conversion between compressed and uncommpressed volumes. This conversion provides a uniform user-experience and eliminates the need for special procedures when dealing with compressed volumes.

3.2.5 Data write flow

When a host sends a write request to a RtC-supported system, it reaches the upper cache layer. The host is immediately sent an acknowledgment of its I/Os.

When the upper cache layer de-stages to the RACE, the I/Os are sent to the thin-provisioning layer. They are then sent to RACE and, if necessary, the original host write or writes are compressed as they are sent to disk. The metadata that holds the index of the compressed volume is updated if needed, and is compressed as well.

3.2.6 Data read flow

When a host sends a read request to the RtC-supported system for compressed data, it is forwarded directly to the RtC component:

- If the RtC component contains the requested data, the RtC-supported system cache replies to the host with the requested data without having to read the data from the lower-level cache or disk.
- If the RtC component does not contain the requested data, the request is forwarded to the lower-level cache of the RtC-supported system.
- If the lower-level cache contains the requested data, it is sent up the stack and returned to the host without accessing the storage.
- If the lower-level cache does not contain the requested data, it sends a read request to the storage for the requested data.

3.2.7 Compression of existing data

In addition to compressing data in real time, it is also possible to compress existing data sets. This compression adds a compressed mirrored copy to an existing volume. You then delete the original copy after the synchronization to the compressed copy is complete. This process is nondisruptive, so the data remains online and accessible by applications and users.

This capability enables customers to regain space from storage pool, which can then be reused for other applications.

With virtualization of external storage systems, the ability to compress already stored data enhances and accelerates the benefit to users. It allows them to see a tremendous return on their storage investment. On initial purchase of a SVC, Storwize V7000, or FlashSystem V840 with IBM Real-time Compression, clients can defer their purchase of new storage. As new storage must be acquired, IT purchases a lower amount of the required storage before compression.

3.3 Software and hardware updates that enhance IBM Real-time Compression

For RtC-supported systems, recent hardware updates and software versions 7.3 and 7.4 introduced improvements that enhance and extend the applicability of the IBM Real-time Compression feature. This section provides an overview of these enhancements:

- Software enhancements:
  - Cache rearchitecture
  - RACE multi-threading
- Hardware enhancements:
  - Additional/enhanced CPU options
  - Increased memory options
  - Intel Assist Acceleration Technology (Coletto Creek) compression acceleration cards

3.3.1 Software enhancements

This section describes recent software enhancements in software versions 7.3 and 7.4 that relate to IBM Real-time Compression.

Cache

As described in*IBM SAN Volume Controller 2145-DH8 Introduction and Implementation*, SG24-8229 and *Implementing the IBM Storwize V7000 Gen2*, SG24-8244, Storwize software Version 7.3 introduced an enhanced, dual-level caching model. This model differs from the single level cache model of previous software versions.

In the previous model, the IBM Real-time Compression Software component sits below the single level read/write cache. The benefit of this model is that the upper level read/write cache masks from the host any latency that is introduced by the IBM Real-time Compression Software component. However, in this single level caching model, the de-staging of writes for compressed I/Os to disk might not be optimal for certain workloads because the RtC component is interacting directly with uncached storage.

In the new, dual-level caching model, the IBM Real-time Compression Software component sits below the upper-level fast write cache and above the lower-level advanced read/write cache. The advantages to this dual-level model with respect to IBM Real-time Compression are several:

- Host writes, whether to compressed or uncommpressed volumes, are still serviced directly through the upper level write cache, preserving low host write I/O latency. Response time can improve with this model, as the upper cache flushes less data to RACE more frequently.
- The performance of the de-staging of compressed write I/Os to storage is improved because these I/Os are now de-staged through the advanced lower-level cache, as opposed to directly to storage.
- The existence of a lower-level write cache below the IBM Real-time Compression component in the software stack allows for the coalescing of compressed writes and, as a result, a reduction in back-end I/Os because of the ability to perform full-stride writes for compressed data.
- The existence of a lower-level read cache below the IBM Real-time Compression component in the software stack allows the temporal locality nature of RtC to benefit from pre-fetching from the back-end storage.

- The main (lower-level) cache now stores compressed data for compressed volumes, increasing the effective size of the lower-level cache.
- Support for larger numbers of compressed volumes.

RACE multi-threading

Software Version 7.4 introduces multi-threading for the RACE process that handles metadata operations for compressed volumes. This approach allows each node to run multiple threads, allowing for more efficient metadata processing for compressed volumes, and as a result, improved performance for compressed workloads. For more information about the improvements to performance of compressed volumes in software Version 7.4, see Chapter 6, “Performance guidelines” on page 83.

RACE multi-threading in software Version 7.4 is supported on the following storage systems:

- SVC 2145-DH8 nodes
- Storwize V7000 Gen2 systems with 64 GB of memory
- FlashSystem V840

Note: RACE multi-threading is a transparent feature, that is, no user action is required to take advantage of this feature. For supported systems running software Version 7.4, the system automatically uses RACE multi-threading. For earlier systems running software Version 7.4. the system uses RACE in a single threaded manner.

3.3.2 Hardware enhancements

As described in*IBM SAN Volume Controller 2145-DH8 Introduction and Implementation*, SG24-8229 and *Implementing the IBM Storwize V7000 Gen2*, SG24-8244, there are numerous hardware enhancements. Several of these enhancements relate directly to the IBM Real-time Compression feature and offer performance and scalability improvements over previous hardware versions.

Additional/enhanced CPU options

The 2145-DH8 node offers updated primary CPUs that contain eight cores, compared to the 4- and 6-core processors that are available in previous hardware versions. Additionally, the 2145-DH8 node offers the option of a secondary 8-core processor for use with IBM Real-time Compression. The FlashSystem V840 comes standard with two of the same 8-core processors. This additional, compression-dedicated processor allows for improved overall system performance when using compression over previous hardware models.

Note: To use the IBM Real-time Compression feature on 2145-DH8 nodes and FlashSystem V840 systems, the secondary processor is required.

The Storwize V7000 Gen2 offers updated 8-core processors, as compared to the 4-core processor that is available in the previous hardware version.

Increased memory options

The 2145-DH8 node and the Storwize V7000 Gen2 each offer the option to increase the node memory from the base 32 GB to 64 GB, for use with IBM Real-time Compression. The FlashSystem V840 comes standard with 64 GB of memory. In all three cases, this additional, compression-dedicated 32 GB of memory allows for improved overall system performance when using compression over previous hardware models.

Note: To use the IBM Real-time Compression feature on 2145-DH8 nodes, the additional 32 GB memory option is required.

Intel Quick Assist Acceleration Technology (Coletto Creek) compression acceleration cards

The 2145-DH8 node offers the option to include either one or two Intel Quick Assist compression acceleration cards that are based upon the Coletto Creek chipset. The Storwize V7000 Gen2 comes with one Intel Quick Assist compression acceleration card. The introduction of these Intel based compression acceleration cards in the SAN Volume Controller 2145-DH8 node is an industry first, providing dedicated processing power and greater throughput over previous models.

Note: To use the IBM Real-time Compression feature on 2145-DH8 nodes, at least one Quick Assist compression acceleration card is required. To use the IBM Real-time Compression feature on the FlashSystem V840, both Quick Assist compression acceleration cards are required. With a single card, the maximum number of compressed volumes per I/O group is 200. With the addition of second Quick Assist card, the maximum number of compressed volumes per I/O group is 512.

4

Planning

This chapter describes planning guidelines that contribute to successful solution design. It addresses the requirements for configuring real-time compression and the best candidate data sets for using compression in IBM Storwize V7000 and IBM SAN Volume Controller (SVC).

This chapter includes the following sections:

- Candidate data sets for compression
- Requirements
- Comprestimator
- General guidelines
- Generic volumes that are created from reused extents

4.1 Candidate data sets for compression

There are common workloads that are good candidates for compression, and others that are not. Distinguishing between them requires you to understand your workload, and to plan to implement workloads that are suitable for compression.

4.1.1 Data types

The best candidates for data compression are data types that are not compressed by nature. These data types involve many workloads and applications, such as databases, character/ASCII based data, email systems, server virtualization, CAD/CAM, software development systems, and vector data.

Attention: Use compression for data with a compression ratio of 25% or higher. Do not attempt to compress data that is compressed by nature. Selecting such data to be compressed provides little or no savings while consuming processor resources by generating additional I/Os. Compression relies on reduced I/Os to achieve the same or better performance than an uncompressed volume. In addition, workloads that create compressed data should not be stored on compressed volumes. Use the Comprestimator tool or have a full understanding of the data types before attempting compression.

The following examples represent workloads and data that are already compressed.

Compressed audio, video, and image file formats
File types such as JPEG, PNG, MP3, medical imaging (DICOM), and MPEG2

Compressed user productivity file formats
Microsoft Office 2007 newer formats (pptx, docx, xlsx, and so on), PDF files, Microsoft Windows executable files (exe), and so on

Compressed file formats
File types such as zip, gzip, rar, cab, and tgz

In cases where data is encrypted by the client or the application that is used, no savings can be achieved by using *any* compression method. Because of the high entropy that is found in encrypted data, compression algorithms cannot find similarities or repetitions within them, and are ineffective. Do not spend system resources on compressing encrypted data or storing that data on compressed volumes.

The following common workloads are typically suitable for use with compression:

- Database applications: Oracle, IBM DB2, Microsoft SQL, and SAP
- Server/Desktop virtualization: VMware, KVM, and Hyper-V
- Messaging: IBM Notes, and Microsoft Exchange (if attachments are not compressed file types)
- Others: Engineering, collaboration, and seismic

4.2 Requirements

This section describes the requirements for using compression in Storwize V7000 and SVC. Compression works on a specific type of hardware node starting with SVC Version 6.4.0.

4.2.1 Supported hardware configurations

Version 7.3 supports the new hardware models Storwize V7000 Gen2 (2076-524) and SVC (2145-DH8). These models deliver outstanding performance for many workloads and are preferred when using IBM Real-time Compression.

The following hardware configurations are preferred with IBM Real-time Compression:

- Storwize V7000 Gen2 with 64 GB and 2X Compression Accelerator Cards (per node canister)
- SVC (2145-DH8) with one or two Compression Accelerator Cards (per node canister)

4.2.2 Hardware requirements

Compression is supported on Storwize V7000 and SVC on the models that are shown in Table 4-1.

Table 4-1 SAN Volume Controller and Storwize V7000 models that support compression

Model	**100**	**300**	**V7000 Gen2**	**8F2**	**8F4**	**8G4**	**CF8**	**CG8**	**DH8**
Storwize V7000	Yes	Yes	Yes	N/A	N/A	N/A	N/A	N/A	N/A
SAN Volume Controller	N/A	N/A	N/A	No	No	No	Yes	Yes	Yes[a]

a. Compression is supported on the SAN Volume Controller DH8 model only with an additional processor and at least one Compression Accelerator card.

Note: For the four core SAN Volume Controller CG8, treat these nodes per the guidelines for the SAN Volume Controller CF8. To check a node’s model type, run `svcinfo lsnodevpd`, and if the output returns Xeon 5630, it is a 4-core SAN Volume Controller CG8.

The minimum code level that is required to support compression is Version 6.4.0.

The system does not allow a non-supported node type, such as an 8F4, to be added to an I/O Group that is enabled for compression.

Remember: Compression is enabled when the first compressed volume is created, and disabled when the last compressed volume is removed from an I/O Group.

Compression requires dedicated hardware resources within the nodes that are assigned or de-assigned when compression is enabled or disabled. That is, when you create the first compressed volume in an I/O Group, hardware resources are assigned.

There are fewer cores that are available to the Fast Path I/O code (normal host to disk I/O processing cores). Therefore, avoid creating compressed volumes if the processor utilization (before creating compressed volumes) is consistently sustained above the percentages that are shown in Table 4-2.

Table 4-2 Maximum existing sustained processor utilization per node

Per node	**SAN Volume Controller CF8**	**SAN Volume Controller CG8**	**SAN Volume Controller CG8 Dual-CPU**	**SAN Volume Controller DH8**	**Storwize V7000**	**Storwize V7000 Gen2**
Processor already close to or above:	25%	50%	No CPU consideration	No CPU consideration	20%	50%

Attention: If any node in a particular I/O Group already has sustained processor utilization greater than those values in Table 4-2, do *not* create compressed volumes in this I/O Group. Doing so might affect existing non-compressed volumes that are owned by this I/O Group. If you have any questions, speak to your IBM representative.

If the nodes in your I/O Group are sustaining processor usage higher than the suggested maximum processor usage, this can cause problems when using compression. Add an I/O Group (or Storwize V7000 control enclosure) to your cluster before attempting to create compressed volumes.

For more information about what resources are assigned and the implications, see 6.2.3, “Benchmarking compression performance” on page 86.

4.2.3 Compressible data

Identify compressible data before using compression. Different data types have different compression ratios. An estimation of expected compression ratios can be performed by using these methods:

Using well-known ratios	Review your existing data sets and applications, and compare them with the well-known data types and ratios that are listed in Table 4-3 on page 29.
Tool-based	Scan the data for compression. A tool that is called Comprestimator can quickly scan existing volumes (even ones in other storage systems or local disks) to provide an accurate estimation of the expected compression ratio. For more information, see 4.3, “Comprestimator” on page 29.

Consideration: When a data format is already compressed, very little additional savings can be achieved by using storage system compression. Do not spend system resources on data that cannot be compressed further. Workloads that create compressed data should not be stored on compressed volumes.

Table 4-3 provides typical compression ratios for types of data.

Table 4-3 Data types

Data types/Applications	Compression ratio
Oracle / DB2	Up to 80%
Office 2003	Up to 60%
Office 2007	Up to 20%
CAD / CAM	Up to 70%
Oil / Gas	Up to 50%
IBM i ERP Data	Up to 75%
VMware: Windows OS	Up to 45 - 55%
VMware: Linux virtual OS	Up to 70%

4.3 Comprestimator

Comprestimator is a command-line host-based utility that can be used to estimate the expected compression rate for block-devices. The utility uses advanced mathematical and statistical algorithms to perform sampling and analysis efficiently. The utility also displays its accuracy level by showing the compression accuracy range of the results that are achieved based on the formulas it uses. The utility runs on a host that has access to the devices to be analyzed. It runs only read operations, so it has no effect on the data that is stored on the device. The following sections provide useful information about installing Comprestimator on a host and using it to analyze devices on that host.

This version of Comprestimator is supported to run on the following host operating systems:

- ESXi 4, 5
- Red Hat Enterprise Linux Version 5.x, 6
- HP-UX 11.31
- Sun Solaris 10 and 11
- SUSE SLES 11
- Ubuntu 12
- CentOS 5.x
- IBM AIX® V6.1, V7.1
- IBM i through VIOS 2.1 or 2.2
- Windows 2003 Server, Windows 7, Windows 2008 Server, Windows 8, and Windows 2012

4.3.1 Downloading Comprestimator

Comprestimator is available from the following website:

http://www14.software.ibm.com/webapp/set2/sas/f/comprestimator/home.html

4.3.2 Installing Comprestimator

Comprestimator can be installed only on supported Windows operating systems, as mentioned in 4.3, “Comprestimator” on page 29. After installation completes, the binary files for other supported operating systems are available in the Windows installation folder.

By default, the files are copied to the following locations:

- In Windows 64-bit: `C:\Program Files (x86)\IBM\Comprestimator`
- In Windows 32-bit: `C:\Program Files\IBM\Comprestimator`

4.3.3 Installing Comprestimator on Linux or AIX

To install Comprestimator on a Linux or AIX host, complete the following steps:

1. Log in to the host by using the root account.
2. Copy the Comprestimator binary file to any folder on the host by using an SCP utility.
3. Make sure that the file has execute permissions. To add execute permissions to the binary file, run **`chmod +x comprestimator`**.

4.3.4 Installing Comprestimator on IBM i through VIOS

To install Comprestimator on an IBM i client through VIOS, complete the following steps:

1. Download and use the AIX Comprestimator tool, and FTP it to VIOS in binary mode.
2. Log in to the VIOS host by using the padmin account.
3. Run **`oem_setup_env`** to exit the padmin shell to a `#` prompt.
4. Rename the executable file to `mv comprestimator_AIX comprestimator`.
5. Make sure that the file has execute permissions. To add execute permissions to the binary file, run **`chmod +x comprestimator`**.

4.3.5 Installing Comprestimator on ESXi

To install Comprestimator on ESXi, complete the following steps:

1. Enable SSH on the ESXi server. Skip this step if SSH is already enabled. Instructions are available at the following website:

   http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1017910

2. Copy the Comprestimator binary file to any folder on the ESXi server. You can copy the file to the ESXi server by using the Secure Copy Protocol (SCP), for example, **`scp`** on Linux or **`winscp`** on Microsoft Windows.

4.3.6 Installing Comprestimator on Windows server

To install Comprestimator on Windows, complete the following steps:

1. Log in to the server using the Administrator account.
2. Copy the appropriate (32-bit or 64-bit) Comprestimator binary file from the corresponding folder to any folder on the host.

4.3.7 Using Comprestimator

The procedures to use Comprestimator depend on the type of server.

Using Comprestimator on a Linux, ESXi, VIOS, or AIX server

To use Comprestimator on a Linux, ESXi, VIOS, or AIC server, complete the following steps:

1. Log in to the server by using the root account.
2. Obtain a list of device names by using one of the following procedures:
   - Use the **fdisk -l** command with a Linux host.
   - Use the **esxcli storage core device list | grep Dev** command with an ESXi 5 server.
   - Use the **lsdev -Cc disk** command with an AIX/VIOS host.
3. Run Comprestimator with the **-d <device_name>** flag to analyze a device or a partition. Example 4-1 shows the syntax for the **comprestimator** command with a Linux host.

Example 4-1 Comprestimator syntax for Linux

```
[root@lbsserv34 sbin]# comprestimator_linux -h
Comprestimator - version : 1.5.1.1 (Build u0087)
Usage :
comprestimator [ -h | -d device  [-c filename]  [-v] [-p number_of_threads]
[-P] [-I] [--storageVer=version] [--config=task_file]
-d device name          Path of device to analyze (e.g.: /dev/sdb)
-p number               Number of threads (default 10)
-c                      Export the results to a CSV file
-v                      Verbose output
-h                      Print this help message
-P                      Display results using a paragraph format
-I                      Allow larger scale of storage io-error threshold rate
(up to 5%)
--config=file           Configuration file that contains list of devices to
analyze
--storageVer=version    Target Storwize/SVC/Flex storage system version.
Options include 6.4, 7.1, 7.2, 7.3; default is 7.3

[root@lbsserv34 sbin]#
```

Using Comprestimator on a Windows server

To use Comprestimator on a Windows server, complete the following steps:

1. Log in to the server using the Administrator account.
2. Run Comprestimator with the **-l** flag to list the devices.
3. Run Comprestimator with the **-n** flag to analyze a device or partition. Example 4-2 shows the syntax for the **comprestimator** command with a Windows host.

Example 4-2 Comprestimator syntax for Windows

```
C:\RtCA\v7k_comp_redbook\Comprestimator\Comprestimator\Windows\>comprestimator..exe -h
Comprestimator version : 1.5.1.1 (Build w0087)
Usage :
comprestimator [ -h | -l | -d device | -n disk_number] [-c filename]  [-v] [-p
number_of_threads] [-P] [-I] [--storageVer=ver
-n                      Disk number
-l                      List devices
-d device name          Path of device to analyze
-p number               Number of threads (default 10)
-c                      Export the results to a CSV file
```

```
-v                      Verbose output
-h                      Print this help message
-P                      Display results using a paragraph format
-I                      Allow larger scale of storage io-error threshold rate
(up to 5%)
--config=file           Configuration file that contains list of devices to
analyze
--storageVer=version    Target Storwize/SVC/Flex storage system version.
Options include 6.4, 7.1, 7.2, 7.3; default is 7.3
C:\RtCA\v7k_comp_redbook\Comprestimator\Comprestimator\Windows>
```

Interpreting Comprestimator results

The explanations of the scan results are shown in Table 4-4.

Table 4-4 Explanations of scan results

Header	Explanation
Sample (#)	The number of the current sample that is reported.
Device	The device name that is used for the scan.
Size (GB)	The total size of the device as reported by the operating system in gigabytes.
Compressed Size (GB)	The estimated size of the device if it is compressed by using SVC or V7000 Real-time Compression.
Total Savings (GB)	The total estimated savings from thin-provisioning and compression in gigabytes.
Total Savings (%)	The estimated savings from thin provisioning and compression, in percentage of the size of the device. This value is calculated by the following method: Total Savings(%) = 1 - (Compressed Size(GB) / Size(GB))
Thin Provision Savings (%)	The estimated savings from thin provisioning (areas with zeros are stored using minimal capacity).
Compression Savings (%)	The estimated savings from compression.
Compression Accuracy Range (%)	The accuracy of the estimate that is provided by Comprestimator. The results that are provided are estimated based on samples from the device and therefore might be lower or higher than the actual compression that is achieved. The approximate accuracy of the results is represented as a percentage of the total size of the device. If the estimated Compression Savings (%) is 67%, and the Compression Accuracy Range is 5%, the actual compression savings is 62% - 72%.

Before implementing IBM Real-time Compression, use the Comprestimator utility to decide whether the volume is good candidate for compression. As a general guideline, enable compression on volumes that show 25% or more Compression Savings.

Evaluate the workload with compression for volumes with Compression Savings lower than 25%.

A Linux example for the Comprestimator utility is shown in Example 4-3 on page 33.

Example 4-3 Comprestimator utility for Linux

```
[root@swfc205 ~]# /comprestimator_1.5.1.1_u0087 -d /dev/sda
Version: 1.5.1.1 (Build u0087)
Start time: 27/08/2014 17:21:06
Device name: /dev/sda
Device size: 278.5 GB
Number of processes: 10
Exhaustive: no

 Sample# | Device   | Size(GB) | Compressed |   Total    |   Total    | Thin Provisioning | Compression |    Compression
         |  Name    |          |  Size(GB)  | Savings(GB)| Savings(%) |    Savings(%)     | Savings(%)  | Accuracy Range(%)
---------+----------+----------+------------+------------+------------+-------------------+-------------+------------------
    3516 |/dev/sda  |     278.5|      86.30 |       192.1|      69.0% |              22.3% |      60.1% |             4.7%
```

A Windows example for the Comprestimator utility is shown in Example 4-4.

Example 4-4 Comprestimator utility for Windows

```
C:\RtCA\v7k_comp_redbook\Comprestimator\Comprestimator_V1.5.1.1\Windows_32bit>comprestimator_win32.exe -l
Drive number [0]
\\?\ide#diskwdc_wd1200bevs-08ust0___________________02.01a02#5&1635548a&0&0.0.0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

C:\RtCA\v7k_comp_redbook\Comprestimator\Comprestimator_V1.5.1.1\Windows_32bit>comprestimator_win32.exe -n 0
Version: 1.5.1.1 (Build w0087)
Start time: 20/07/2012 14:56:25.265625
Device name: \\?\ide#diskwdc_wd1200bevs-08ust0___________________02.01a02#5&1635548a&0&0.0.0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
Device size: 100 GB
Number of processes: 10
Exhaustive: no

Sample# | Device                | Size(GB) | Compressed |   Total    |   Total    | Thin Provisioning | Compression |   Compression
        |  Name                 |          |  Size(GB)  | Savings(GB)| Savings(%) |    Savings(%)     | Savings(%)  | Accuracy Range(%)
---------+----------------------+----------+------------+------------+------------+-------------------+-------------+------------------
3102     |5&1635548a&0&0.0.0    | 100.0    |    17.875  |   82.125   |   65.30%   |  24.81%           |    76.22%   |        5.0%
C:\RtCA\v7k_comp_redbook\Comprestimator\Comprestimator_V1.5.1.1\Windows_32bit>
```

4.3.8 Available space

Compression can be configured only when a volume is created, or when a volume copy is added to an existing volume. To compress existing data, create a compressed volume copy and copy the existing data from the original volume copy to the compressed mirrored one. The mirroring process is not apparent to the host. During the synchronization between the volume copies, additional space is required to run the mirroring. After the data transfer is complete and the volume copies in sync, delete the original volume copy to free up space.

In addition, the volume mirroring capability enables the evaluation of system performance and an effective compression ratio with compressed volumes without deleting the original volume copy. You can then complete the migration to compressed format by deleting the original volume copy.

Remember: Available space is needed just in case you want to compress existing volumes.

Calculating the free space that is needed for compressed volume mirroring is easy. Because a compressed volume is basically a thin-provisioned volume, you create a much smaller volume. The amount of free space that is needed is the estimated compressed size of the data. The calculation depends on the type of data that is stored on the volume you want to compress.

As an example, a 100 GB volume stores several Linux VMware VMDK files, and the expected compression ratio is 45 - 55%. In that case, you need at least 55 GB of available space in the storage pool that contains the compressed volume copy. If you have a type of data that is not mentioned in Table 4-3 on page 29, use the Comprestimator tool to predict the amount of space that is needed.

Tip: In both cases, have about 10% of extra space in the pool for additional metadata and to gap the error margin.

4.3.9 Licensing

To use compression on Storwize V7000 or SVC, a software license is required.

In SVC, real-time compression is licensed by the total number of terabytes of virtual capacity, which is the amount of storage that the host sees. You do not need to purchase a compression license for the whole cluster. The effects of compression are within an I/O Group. Therefore, you can have one I/O Group enabled for compression while other I/O Groups are not.

In Storwize V7000, IBM Real-time Compression is licensed by the total number of enclosures. There is a new simplified licensing structure:

- Flexible options: Purchase a license per advanced Storwize V7000 function (such as Easy Tier and IBM Real-time Compression).
- Full Bundle: Purchase a license for all advanced Storwize V7000 for a lower price.

In both options, all enclosures should get a license for every feature.

For more information about how to install the compression license, see 5.2, “Software licensing for the SAN Volume Controller and Storwize V7000” on page 40.

4.4 General guidelines

There are various configuration items that affect the performance of compression in the Storwize V7000 and SVC.

Create compressed volumes by using the preferred settings that are described in Chapter 5, “Configuring compressed volumes” on page 39.

The **`migratevdisk`** command can be used to migrate a compressed volume from one storage pool to another. However, the extent sizes must be the same on both storage pools.

The **`addvdiskcopy`** command can be used to migrate or compress existing generic volumes by using the **`add mirrored copy`** function. The original copy can then be deleted if wanted and the compressed copy can be mapped to the host. The Volume Mirroring or Adding a Mirrored Copy is therefore available for converting between compressed volume copies and other kinds of volumes. This function can be used to compress or migrate existing data to a compressed volume, and is the preferred method. This method can even be used if the extent sizes of the two pools are not the same.

Follow the established guidelines that exist for SVC and Storwize V7000. For more information, see *IBM System Storage SAN Volume Controller and Storwize V7000 Best Practices and Performance Guidelines*, SG24-7521.

4.4.1 Compression ratio recommendations

Software Version 7.4 introduces performance improvements when using SAN Volume Controller 2145-DH8 or Storwize V7000 Gen2 models because of the new hardware advances and the dual IBM Random Access Compression Engine (RACE) architecture.

Use the threshold for volume compressibility values in Table 4-5 and Table 4-6 to help you decide whether to compress a volume.

Table 4-5 Recommendations when using SAN Volume Controller (2145-DH8) or Storwize V7000 Gen2 (2076-524) models with Version 7.4 and above

Data compression rate	Recommendation
Higher than 25% savings	Use compression.
Below 25% savings	Evaluate a workload with compression.

Table 4-6 Recommendations when using SAN Volume Controller (2145-CG8) or Storwize V7000 (2076-112, 2076-124)

Data compression rate	Recommendation
Higher than 40% savings	Use compression.
Below 40% savings	Evaluate a workload with compression.

4.4.2 Guidelines for getting a high compression ratio

Review the following guidelines to achieve the highest compression savings in your environment:

- To achieve a high compression ratio, use compressible data. Table 4-3 on page 29 lists common applications that provide a high compression ratio. Storing these data types on compressed volume saves disk space and improves the benefit of using compression with SVC or Storwize V7000.
- Avoid using any client, file system, or application compression when using SVC or Storwize V7000 compressed volumes. In most cases, data that is already compressed cannot achieve significant additional savings from compressing it again.
- Avoid using any client, file system, or application encryption when using a SVC or Storwize V7000 compressed volume. Encrypted data is not compressible.

4.4.3 Easy Tier

Easy Tier is a performance function that automatically migrates extents off a volume from one MDisk storage tier to another MDisk storage tier. Easy Tier monitors the host I/O activity and latency on the extents of all volumes in a multi-tiered storage pool over a 24-hour period.

Starting with Version 7.1, Easy Tier supports compressed volumes. A new algorithm is implemented to monitor *read* operations on compressed volumes instead of reads and writes. The extents with the highest number of read operations that are smaller than 64 KB are migrated to solid-state disk (SSD) MDisks. As a result, frequently read areas of the compressed volumes are serviced from SSDs. Easy Tier on non-compressed volumes operates as before and it is based on read and write operations smaller than 64 KB.

For more information about Easy Tier with compression, see *Implementing IBM Easy Tier with IBM Real-time Compression*, TIPS1072:

http://www.redbooks.ibm.com/abstracts/tips1072.html

4.4.4 I/O Groups and compressed volumes

When there are only a small number of compressed volumes (that is, 10 - 20), configure them on a single I/O Group rather than splitting them between different I/O Groups.

For larger numbers of compressed volumes in systems with more than one I/O Group, distribute compressed volumes across I/O Groups. If a clustered pair of Storwize V7000 control enclosures requires 100 compressed volumes, configure 50 volumes per I/O Group instead of 100 compressed volumes in one I/O Group. Also, ensure that the preferred nodes are evenly distributed across both nodes in the I/O Group. This is the default behavior when you create consecutive volumes.

Consideration: Each I/O Group of SVC DH8 model and Storwize V7000 Gen2 with 64 GB of memory can accommodate 512 compressed volumes. You can have up to 2048 compressed volumes in a full 8-node cluster. The older models (SVC CF8, CG8, and Storwize V7000 Gen1) are restricted to 200 compression volumes per I/O Group.

4.5 Generic volumes that are created from reused extents

Generic (fully allocated) volumes that are created without initial zero formatting might contain traces of old data at the block device level. Such data is not accessible or viewable in the file system level. However, it might affect compression ratios and system resources when using volume mirroring to migrate such volumes to compressed ones. Therefore, the compression ratio might be less than the estimate that is provided by Comprestimator if the source generic volume is created from non-zeroed extents.

4.5.1 How reused extents consume system resources

When adding a mirrored compressed copy to a generic volume that contains old data (that is, a volume that was not initially zeroed), the old data also is compressed. Therefore, system resources are wasted on compression of old data that is effectively inaccessible to users and applications.

4.5.2 How reused extents affect Comprestimator results

When using the Comprestimator utility to analyze such volumes, the expected compression results reflect the compression rate for all the data in the block device level. This data includes the old data. This block device behavior is limited to generic volumes, and does not occur when using Comprestimator to analyze thin-provisioned volumes.

Regardless of the type of block device that is scanned, it is also important to understand a few characteristics of common file systems space management. When data is deleted from a file system, the space it occupied before it was deleted is freed and available to the file system. It is available even though the data on disk was not deleted. When data is deleted in the file system/application level, the file system index and pointers are updated to reflect this change. The data is not deleted from the device that stores this file system. When using Comprestimator to analyze a block device that is used by a file system, all underlying data in the device is analyzed. The data is analyzed *regardless* of whether this data belongs to files that were deleted from the file system.

4.5.3 Avoiding this behavior

Generally, format new volumes during creation. This process zeros all blocks in the disks and eliminates traces of old data.

Creating a volume

To format a new volume from the GUI, complete the following steps:

1. Click **Pools** → **Volumes by Pools**.
2. Click **New Volume**.
3. Click **Advanced**, and select **Format Before Use**.

To format a new volume from the CLI, when creating a volume by using the **`mkvdisk`** command, add the **`-fmtdisk`** command-line argument.

Remember: The format process can take some time to complete. The volume remains offline until the format is completed.

Creating a volume more quickly

It is possible to create a zeroed fully allocated volume more quickly by using the following procedure:

1. Start with a thin-provisioned volume.
2. Add a fully allocated copy.
3. Mirroring copies the thin-provisioned volume to a fully allocated volume, which has zeros for unused extents.
4. When the fully allocated copy is in sync, delete the thin-provisioned copy.

This process creates a formatted volume faster than the zeroing option with rate control, and the volume is online immediately.

Converting fully allocated volumes

The volume mirroring procedure copies all of the existing blocks of the primary volume copy to the secondary volume copy. If the primary volume copy contains old data, this data is copied over to the new volume copy. Therefore, any space in the file system level on the original volume copy should be filled with zeros before using volume mirroring.

You can convert a fully allocated volume into a compressed (or thin-provisioned) volume nondisruptively by using the following procedure:

1. Start with a single copy of a fully allocated volume. Fill the free capacity on the volume with a file that contains all zeros. For example, create a file and use `/dev/zero` as the source file.
2. Add a compressed (or thin-provisioned) copy to the volume. Use a small real capacity, such as 2%, and the autoexpand feature.
3. Wait until volume mirroring synchronizes the copies.
4. Delete the fully allocated copy.

Any grains of the fully allocated volume that contain all zeros do not cause any real capacity to be allocated on the compressed (or thin-provisioned) copy.

5

Configuring compressed volumes

This chapter addresses how to create a compressed volume and then map the volume to defined hosts. You can use the command-line interface (CLI) or the graphic user interface (GUI) of the IBM Storwize V7000 or IBM System Storage SAN Volume Controller (SVC) to create and map the volume. The SVC and Storwize V7000 must have the correct licensing to use compression.

This chapter includes the following sections:

- Compression
- Software licensing for the SAN Volume Controller and Storwize V7000
- Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 GUI
- Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 CLI
- Interaction with other SAN Volume Controller and Storwize V7000 components

5.1 Compression

Compression is enabled on a volume-copy basis. Compressed volume copies are a special type of thin-provisioned volume copies. They are not only thinly provisioned, but also compressed. Volume copies are marked as compressed or not when they are created by using **`mkvdisk`** or **`addvdiskcopy`**. This setting cannot be changed.

Compressed volume copies can be freely mixed with fully allocated and regular thin-provisioned (that is, not compressed) volume copies. This mixture means that Volume Mirroring is available for converting between compressed volume copies and other types.

Compressed volumes can be moved between clusters in the same way as thin-provisioned volumes. They can be moved by using image mode and the import setting on **`mkvdisk`**. They can be distinguished from other types of volumes in both CLI and GUI views.

The first part of the chapter covers the creation of a compressed volume and mapping with both the CLI and GUI.

The second part of the chapter describes some of the interactions of a compressed volume and other SVC and Storwize V7000 components. These components include copy services, data migration, and thin provisioning.

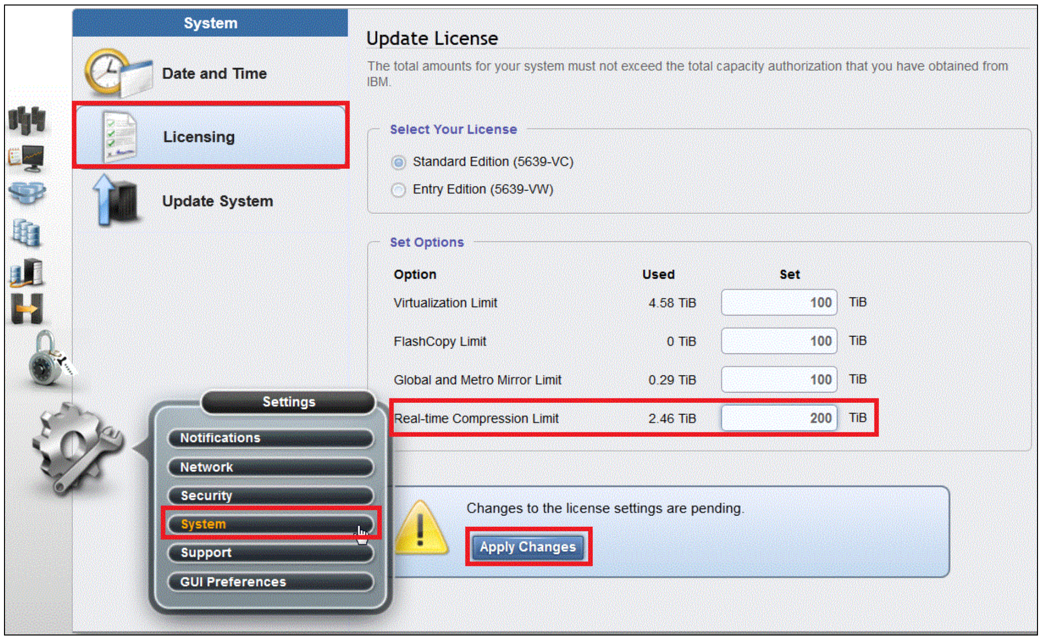
5.2 Software licensing for the SAN Volume Controller and Storwize V7000

To use compression on the SVC and Storwize V7000, licensing is required. With the SVC, real-time compression is licensed by capacity, per terabyte of virtual data. In the Storwize V7000, real-time compression is licensed per enclosure. For more information about licensing, see 5.2.1, “Licensing for the SAN Volume Controller using the GUI” on page 41.

5.2.1 Licensing for the SAN Volume Controller using the GUI

To apply your SVC compression license, complete the following steps:

1. Click the **Settings** icon and select **General** → **Licensing**, as shown in Figure 5-1.



Update License

The total amounts for your system must not exceed the total capacity authorization that you have obtained from IBM.

Select Your License

[✓ Standard Edition (5639-VC)] Entry Edition (5639-VW)

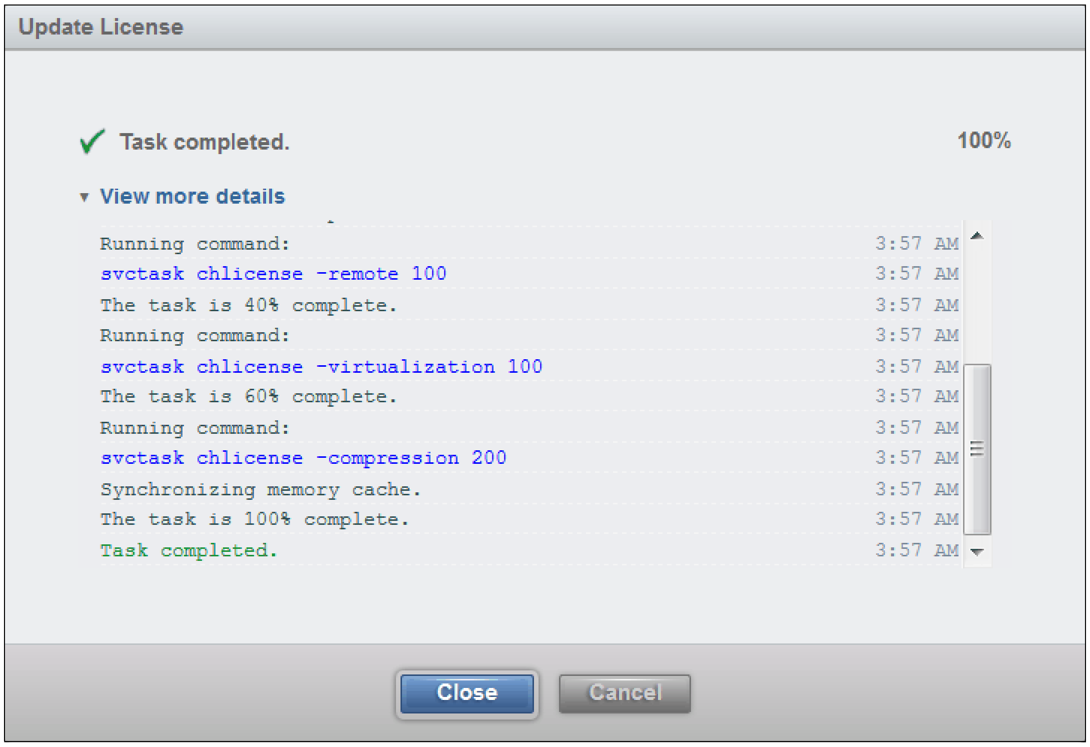
Set Options

Option	Used	Set	
Virtualization Limit	4.58 TiB	100	TiB
FlashCopy Limit	0 TiB	100	TiB
Global and Metro Mirror Limit	0.29 TiB	100	TiB
Real-time Compression Limit	2.46 TiB	200	TiB

Changes to the license settings are pending.

Figure 5-1 Applying the SVC compression license (GUI)

2. Enter the total number of terabytes of virtual capacity that is licensed for compression.
3. Click **Apply Changes** to complete the process, as shown in Figure 5-2.



Task completed. 100%

```
Running command:                                   3:57 AM
svctask chlicense -remote 100                      3:57 AM
The task is 40% complete.                          3:57 AM
Running command:                                   3:57 AM
svctask chlicense -virtualization 100              3:57 AM
The task is 60% complete.                          3:57 AM
Running command:                                   3:57 AM
svctask chlicense -compression 200                 3:57 AM
Synchronizing memory cache.                        3:57 AM
The task is 100% complete.                         3:57 AM
Task completed.                                    3:57 AM
```

Figure 5-2 SAN Volume Controller update license complete

5.2.2 Licensing for the SAN Volume Controller using the CLI

To apply your SVC compression license, enter the total number of terabytes of virtual capacity that is licensed for compression. Run **chlicense**, as shown in Example 5-1.

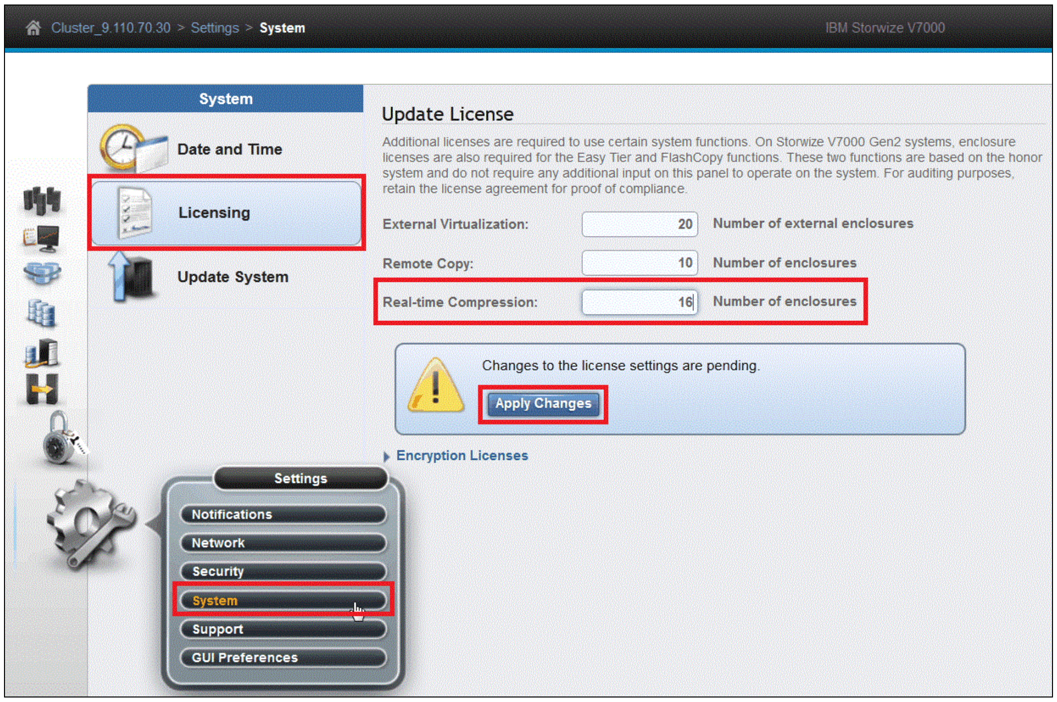
Example 5-1 The chlicense command

```
IBM_2145:ITSO_Remote_Cluster:superuser>chlicense -compression 200
IBM_2145:ITSO_Remote_Cluster:superuser>
```

5.2.3 Licensing for the Storwize V7000 using the GUI

To apply your Storwize V7000 compression license, complete the following steps:

1. Click the **Settings** icon and select **General** → **Licensing**, as shown in Figure 5-3.



Update License

Additional licenses are required to use certain system functions. On Storwize V7000 Gen2 systems, enclosure licenses are also required for the Easy Tier and FlashCopy functions. These two functions are based on the honor system and do not require any additional input on this panel to operate on the system. For auditing purposes, retain the license agreement for proof of compliance.

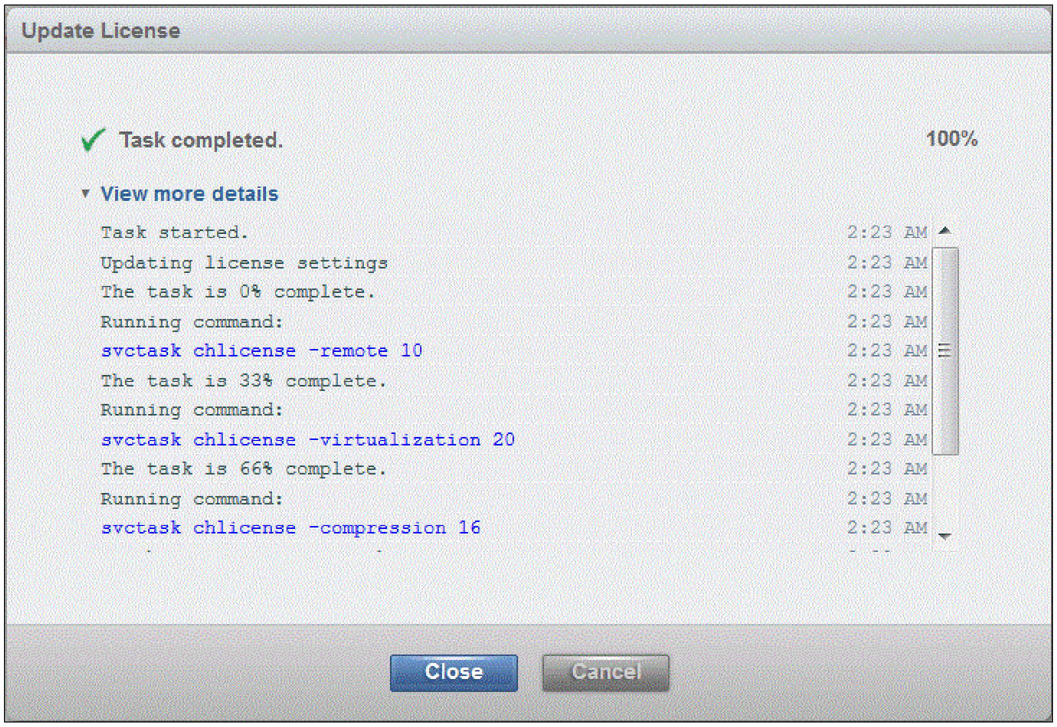
External Virtualization: 20 Number of external enclosures
Remote Copy: 10 Number of enclosures
Real-time Compression: 16 Number of enclosures

Changes to the license settings are pending.

Figure 5-3 Applying the Storwize V7000 compression license (GUI)

2. Enter the total number of enclosures that are licensed for compression.

3. Click **Apply Changes** to complete the process, as shown in Figure 5-4.



Task completed. 100%

View more details

```
Task started.                                  2:23 AM
Updating license settings                      2:23 AM
The task is 0% complete.                       2:23 AM
Running command:                               2:23 AM
svctask chlicense -remote 10                   2:23 AM
The task is 33% complete.                      2:23 AM
Running command:                               2:23 AM
svctask chlicense -virtualization 20           2:23 AM
The task is 66% complete.                      2:23 AM
Running command:                               2:23 AM
svctask chlicense -compression 16              2:23 AM
```

Figure 5-4 Storwize V7000 update license complete

5.2.4 Licensing for the Storwize V7000 using the CLI

To apply your Storwize V7000 compression license, enter the total number of enclosures that are licensed for compression. Run **`chlicense`**, as shown in Example 5-2.

Example 5-2 Apply the Storwize V7000 compression license (CLI)

```
IBM_Storwize:Cluster_9.110.70.30:superuser>
IBM_Storwize:Cluster_9.110.70.30:superuser>chlicense -compression 15
IBM_Storwize:Cluster_9.110.70.30:superuser>
```

5.3 Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 GUI

The steps in this section are limited to outlining the creation of a compressed volume and mapping that volume to a host. There are many other ways to create a volume by using the GUI that are not covered in this book. For more information about configuring your SVC or Storwize V7000, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

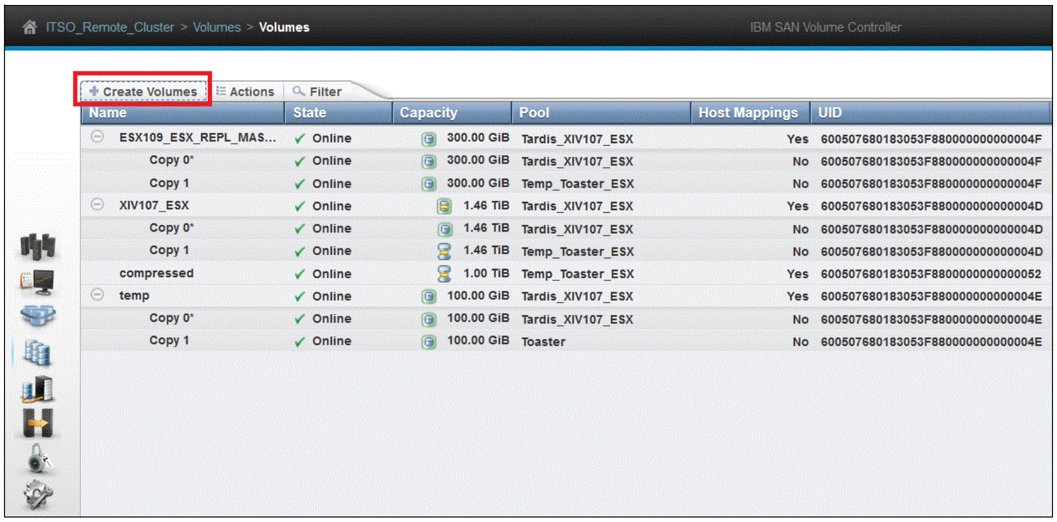
You can also explore SVC and Storwize V7000 IBM Redbooks publications, residencies, and workshops by subscribing to the IBM Redbooks weekly newsletter at the following website:

http://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

5.3.1 Creating a compressed volume

To create a compressed volume, complete the following steps:

1. Go to the Volumes panel under the SAN Volume Controller or Storwize V7000 Welcome panel and click **Volumes** → **Volumes**.
2. Click **Create Volumes**, as shown in Figure 5-5.


ITSO_Remote_Cluster > Volumes > Volumes
IBM SAN Volume Controller
Create Volumes
Actions
Filter

Name	State	Capacity	Pool	Host Mappings	UID
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004F
Copy 0*	✓ Online	300.00 GiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004F
Copy 1	✓ Online	300.00 GiB	Temp_Toaster_ESX	No	600507680183053F880000000000004F
XIV107_ESX	✓ Online	1.46 TiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004D
Copy 0*	✓ Online	1.46 TiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004D
Copy 1	✓ Online	1.46 TiB	Temp_Toaster_ESX	No	600507680183053F880000000000004D
compressed	✓ Online	1.00 TiB	Temp_Toaster_ESX	Yes	600507680183053F8800000000000052
temp	✓ Online	100.00 GiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004E
Copy 0*	✓ Online	100.00 GiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004E
Copy 1	✓ Online	100.00 GiB	Toaster	No	600507680183053F880000000000004E

Figure 5-5 New volume

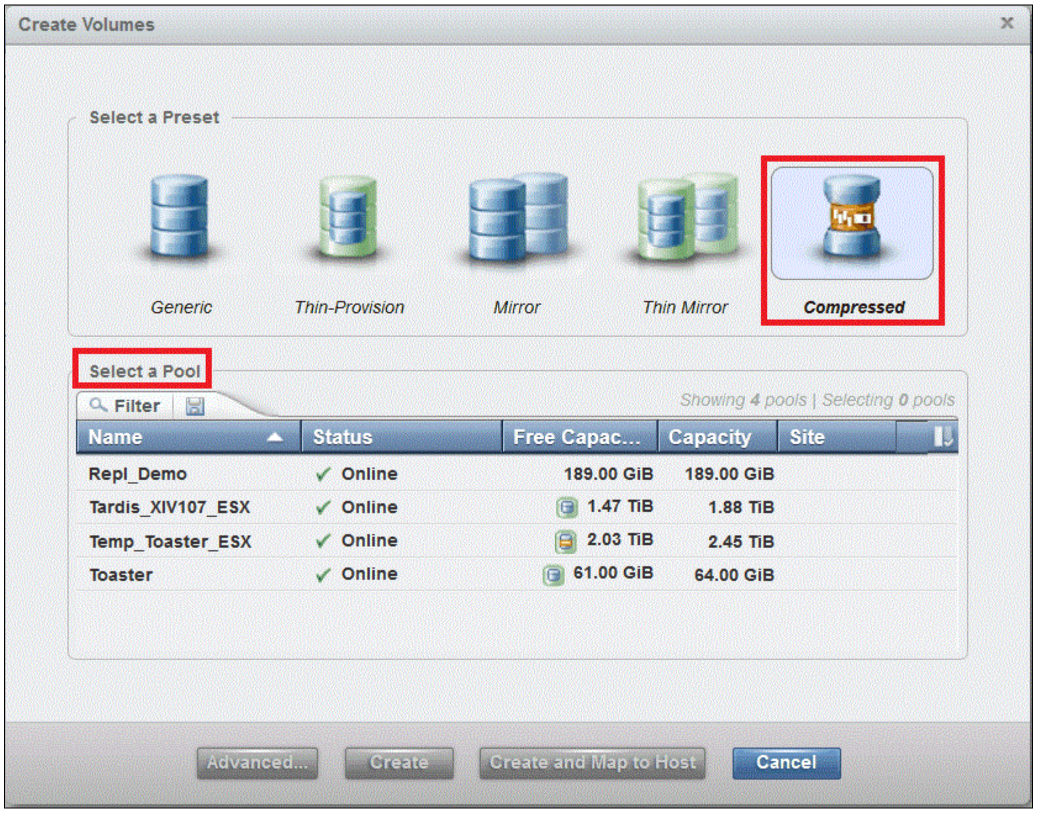
3. Select **Compressed** as the preset for the volume from the following presets:
   - **Generic**: Create volumes that use a set amount of capacity from the selected storage pool.
   - **Thin-Provision**: Create large-capacity volumes that use only the capacity that is written by the host application from the pool.
   - **Mirror**: Create volumes with two physical copies that provide data protection. Each copy can belong to a different storage pool to protect data from storage failures.
   - **Thin Mirror: Create** volumes with two physical copies to protect data from failures while using only the capacity that is written by the host application.
   - **Compressed**: Create volumes where data is compressed when it is written and stored on the disk.

Tip: You can create a compressed volume or compressed mirror by selecting any of the presets:

1. Select the preset that you want, such as **Generic**.
2. Click **Advanced**.
3. Click the **Capacity Management** tab.
4. Select **Compressed** and select the following options:
   - **Real Capacity** is set to 2% of Virtual Capacity.
   - **Automatically Expand** is selected.
   - **Warning Threshold** is set to 80%.

If the values are not set to these settings, the volume can go offline prematurely for thin provisioning and compression alike.

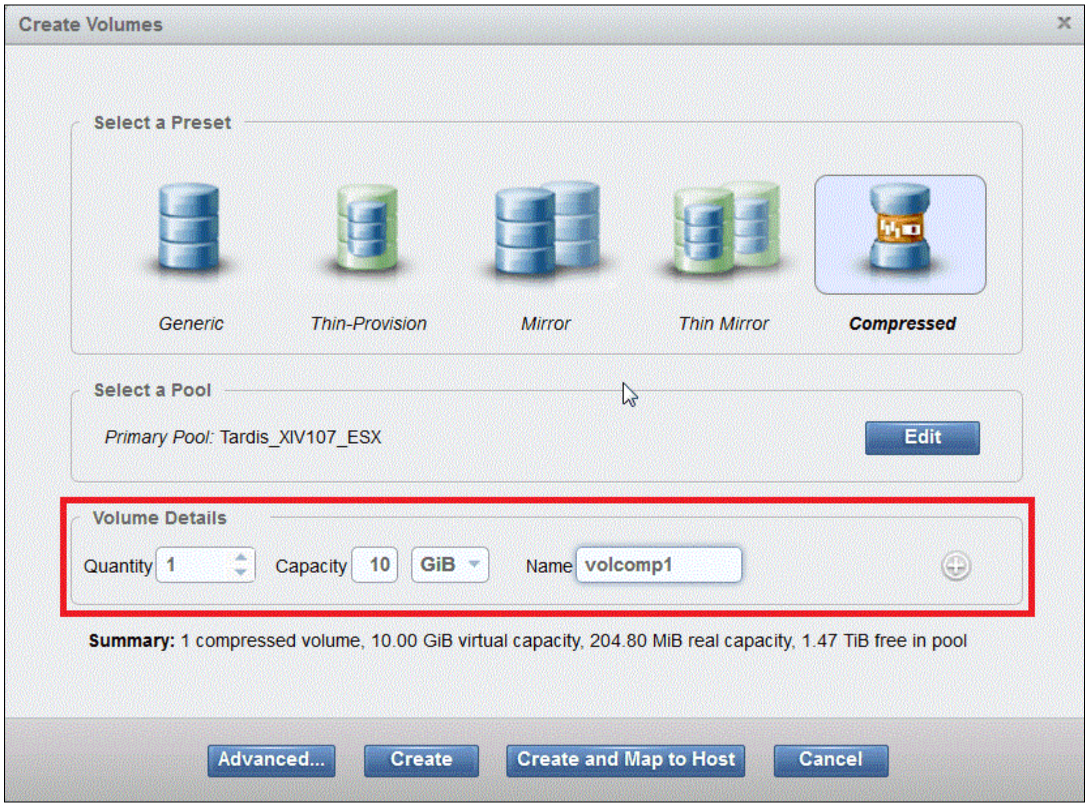
5. Select the pool that you want to use for the compressed volume, as shown in Figure 5-6.



Name	Status	Free Capac...	Capacity	Site
Repl_Demo	✓ Online	189.00 GiB	189.00 GiB	
Tardis_XIV107_ESX	✓ Online	1.47 TiB	1.88 TiB	
Temp_Toaster_ESX	✓ Online	2.03 TiB	2.45 TiB	
Toaster	✓ Online	61.00 GiB	64.00 GiB	

Figure 5-6 Selecting the compressed volume

6. Provide a name and size for the compressed volume, as shown in Figure 5-7.



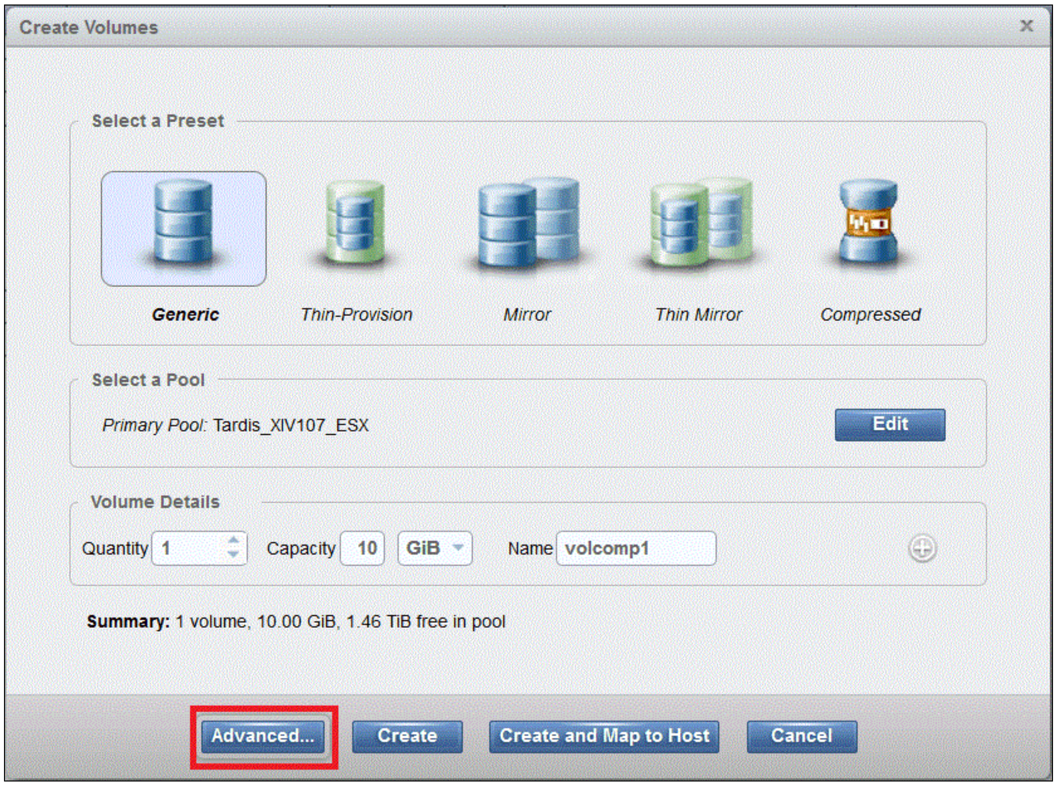
Primary Pool: Tardis_XIV107_ESX

Quantity 1 Capacity 10 GiB Name volcomp1

Summary: 1 compressed volume, 10.00 GiB virtual capacity, 204.80 MiB real capacity, 1.47 TiB free in pool

Figure 5-7 Providing the volume name and size

7. The Advanced section (Figure 5-8) provides a way to change the defaults for volume characteristics, capacity management, and mirroring.



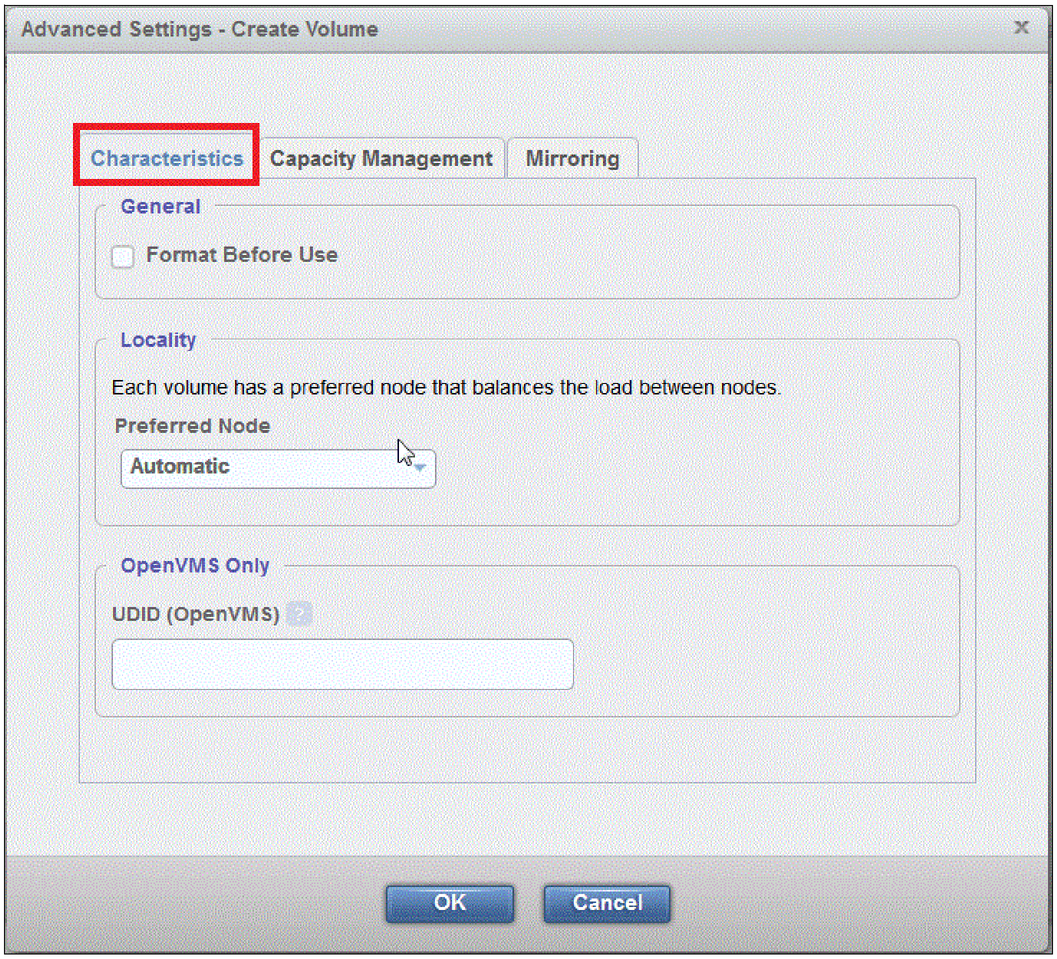
Primary Pool: Tardis_XIV107_ESX

Quantity 1 Capacity 10 GiB Name volcomp1

Summary: 1 volume, 10.00 GiB, 1.46 TiB free in pool

Figure 5-8 Advanced volume settings

8. Set the following characteristics, as shown in Figure 5-9 on page 48.
   - **Format Before Use**: Formatting is not required for compressed volumes, so this flag can be ignored. Formatting uses capacity (write data) on compressed volumes.
   - **Preferred Node**: Each volume within an I/O Group has a preferred node property. This property can be used to balance the I/O load between nodes in that group.
   - **UDID (OpenVMS)**: The unit device identifier (UDID) that is used by OpenVMS hosts to identify the volume.


Advanced Settings - Create Volume

General

Format Before Use

Locality

Each volume has a preferred node that balances the load between nodes.

Preferred Node: Automatic

OpenVMS Only

UDID (OpenVMS)

Figure 5-9 Volume characteristics

Remember: Compressed volumes have the same characteristics as thin provisioned volumes. You must specify that the volume is compressed, specify the rsize, enable autoexpand, and specify a warning threshold. If not configured properly, the volume can go offline prematurely. The preferred settings are to set rsize to 2%, enable autoexpand, and set warning to 80%.

9. Here are the options for capacity management when compressed is selected, as shown in Figure 5-10.

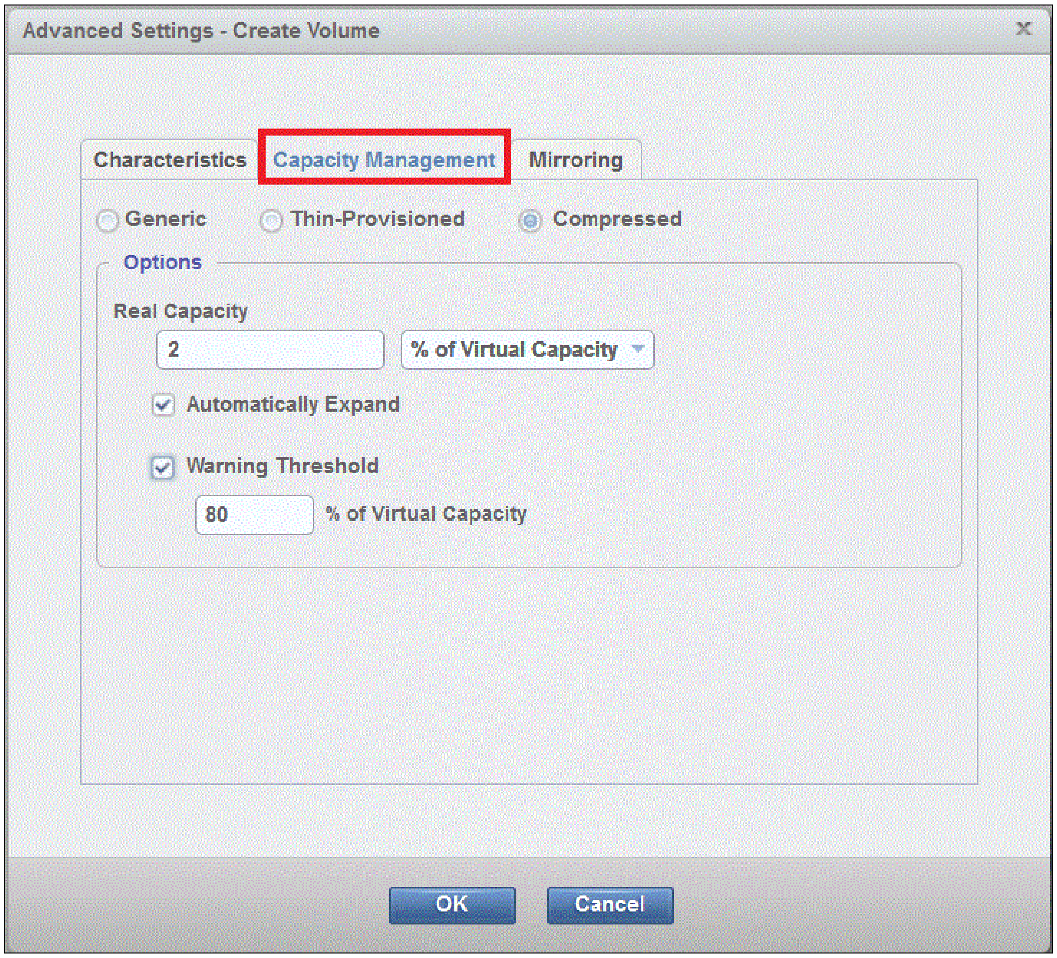


Figure 5-10 Volume capacity management

10. The capacity management settings provide several controls to gain an extra degree of protection against out of space conditions with compressed volumes.

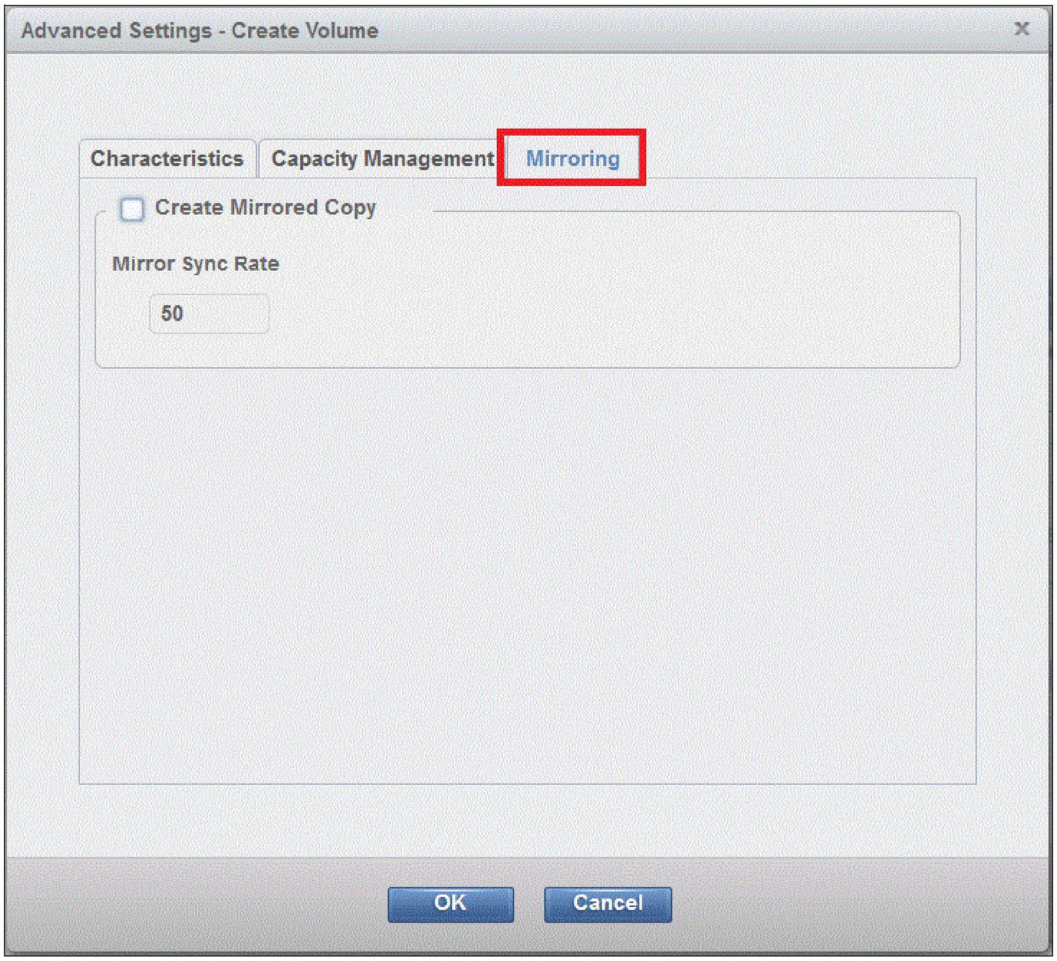
 – **Real Capacity**: The capacity to be allocated to each copy of the volume. This option is also called *rsize*, and provides a method for setting the amount of real capacity that is reserved at volume creation for the compressed volume. The setting can be specified in whole integer units (bytes, kilobytes, megabytes, gigabytes, terabytes, or petabytes) or as a percentage (0-100%). The default setting for rsize is 2%, which is required for compressed volumes.

 – **Automatically Expand**: Automatically increase the real capacity as data is written to the compressed volume. This option is also called *autoexpand*, and should be selected for a compressed volume. Autoexpand attempts to maintain a fixed amount of unused real capacity. This capacity is known as *contingency capacity,* which is initially set to the real capacity specified by rsize.

– **Warning Threshold**: When the used capacity first exceeds the warning threshold for volumes that do not have autoexpand set, an event is raised. This event indicates that additional real capacity is required. You can then manually expand the real capacity that is allocated to the volume. This option is set at creation time and defaults to 80% of the virtual capacity. It can be set to an absolute value or a percentage. The alerting facilities (event, alert, or email) are asynchronous, so the event might not be received before the volume goes offline. For this reason, consider an automated response to this condition. IBM Tivoli® Storage Productivity Center alerts provide a great mechanism for triggering automated responses. For more information, see the following website:

  `http://www-03.ibm.com/systems/storage/software/center/index.html`

11. You can create two copies of the compressed volume by using mirroring, as shown in Figure 5-11. If the copies belong to different pools, hosts can still access data on the volume even if either pool goes offline. The rate determines how quickly copies are synchronized. Mirroring is also used to migrate from compressed to uncompressed and from uncompressed to compressed and specifying which copy of the mirror is the primary copy.

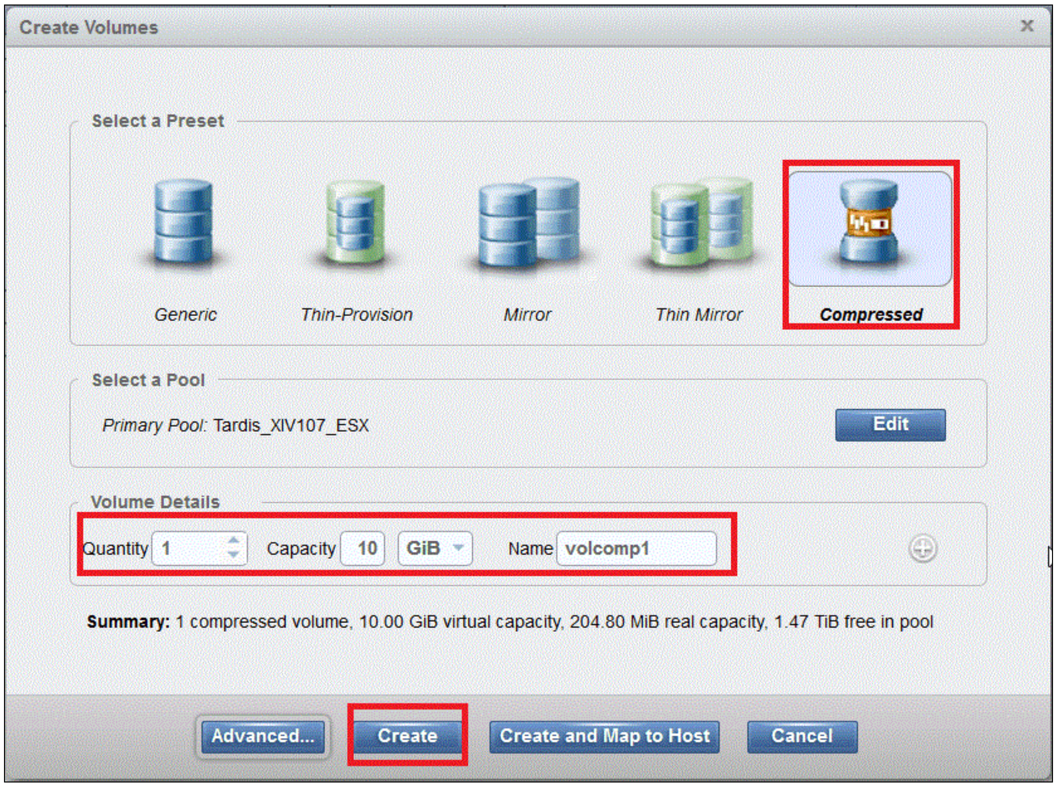


Create Mirrored Copy [ ]

Mirror Sync Rate: 50

Figure 5-11 Volume mirroring

12. You can create the volume at this time by selecting **Create** and entering the details, as shown in Figure 5-12. Mapping the volume to a host is described in 5.3.4, “Mapping a compressed volume to a host” on page 59.



Primary Pool: Tardis_XIV107_ESX

Quantity 1 Capacity 10 GiB Name volcomp1

Summary: 1 compressed volume, 10.00 GiB virtual capacity, 204.80 MiB real capacity, 1.47 TiB free in pool

Figure 5-12 Volume name and size

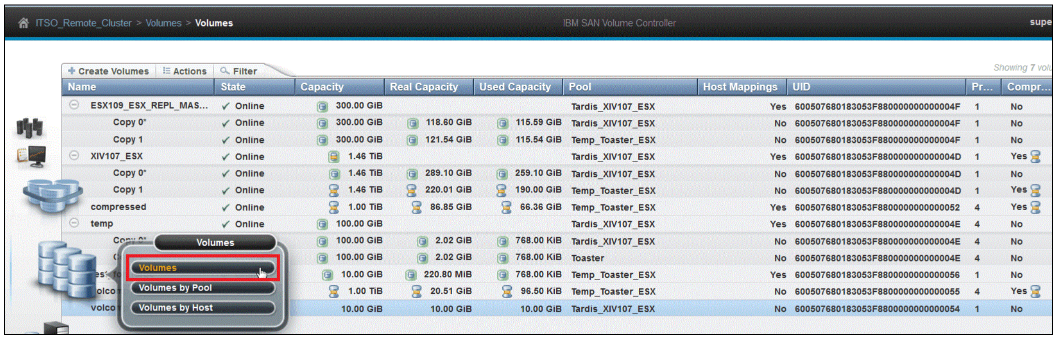
5.3.2 Creating a compressed volume using Volume Copy

As mentioned previously, compressed volume copies can be freely mixed with fully allocated and regular thin-provisioned (that is, not compressed) volume copies. Therefore, you can use Volume Mirroring or Adding a Mirrored Copy to convert between compressed volume copies and other kinds. This process can be used to compress or migrate existing data to a compressed volume or to move an already compressed volume back to a generic volume or uncompress.

This section shows how to create a compressed mirror volume by using one of the methods that are described in 5.3, “Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 GUI” on page 43. The new volume can be modified to be the primary copy and original volume can then be deleted if wanted after the mirroring completes.

To create a compressed volume by using Volume Copy, complete the following steps:

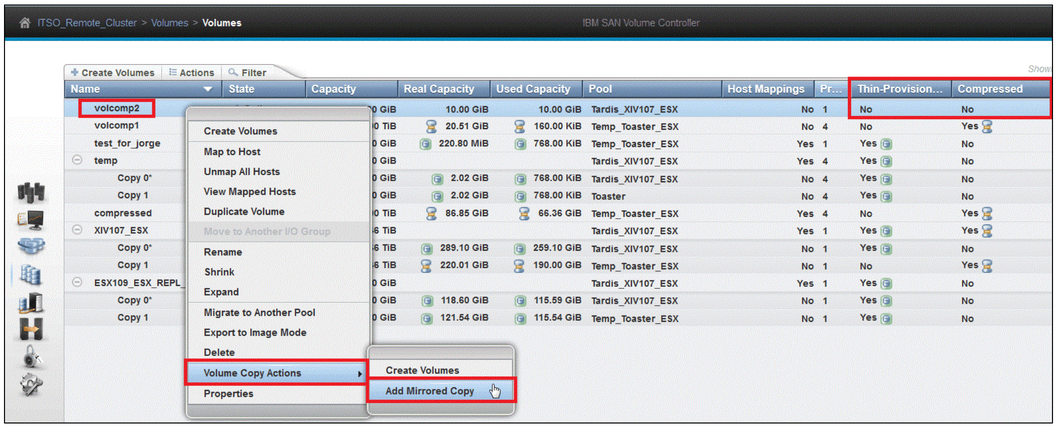
1. Go to the Volumes panel under the SAN Volume Controller or Storwize V7000 Welcome panel, as shown in Figure 5-13, and click **Volumes** → **Volumes**.



Name	State	Capacity	Real Capacity	Used Capacity	Pool	Host Mappings	UID	Pr...	Compr...
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB			Tardis_XIV107_ESX	Yes	600507680183053F88000000000004F	1	No
Copy 0*	✓ Online	300.00 GiB	118.60 GiB	115.59 GiB	Tardis_XIV107_ESX	No	600507680183053F88000000000004F	1	No
Copy 1	✓ Online	300.00 GiB	121.54 GiB	115.54 GiB	Temp_Toaster_ESX	No	600507680183053F88000000000004F	1	No
XIV107_ESX	✓ Online	1.46 TiB			Tardis_XIV107_ESX	Yes	600507680183053F88000000000004D	1	Yes
Copy 0*	✓ Online	1.46 TiB	289.10 GiB	259.10 GiB	Tardis_XIV107_ESX	No	600507680183053F88000000000004D	1	No
Copy 1	✓ Online	1.46 TiB	220.01 GiB	190.00 GiB	Temp_Toaster_ESX	No	600507680183053F88000000000004D	1	Yes
compressed	✓ Online	1.00 TiB	86.85 GiB	66.36 GiB	Temp_Toaster_ESX	Yes	600507680183053F880000000000052	4	Yes
temp	✓ Online	100.00 GiB			Tardis_XIV107_ESX	Yes	600507680183053F88000000000004E	4	No
Copy 0*		100.00 GiB	2.02 GiB	768.00 KiB	Tardis_XIV107_ESX	No	600507680183053F88000000000004E	4	No
C		100.00 GiB	2.02 GiB	768.00 KiB	Toaster	No	600507680183053F88000000000004E	4	No
es fo		10.00 GiB	220.80 MiB	768.00 KiB	Temp_Toaster_ESX	Yes	600507680183053F880000000000056	1	No
olco		1.00 TiB	20.51 GiB	96.50 KiB	Temp_Toaster_ESX	No	600507680183053F880000000000055	4	Yes
volco		10.00 GiB	10.00 GiB	10.00 GiB	Tardis_XIV107_ESX	No	600507680183053F880000000000054	1	No

Figure 5-13   Selecting a volume

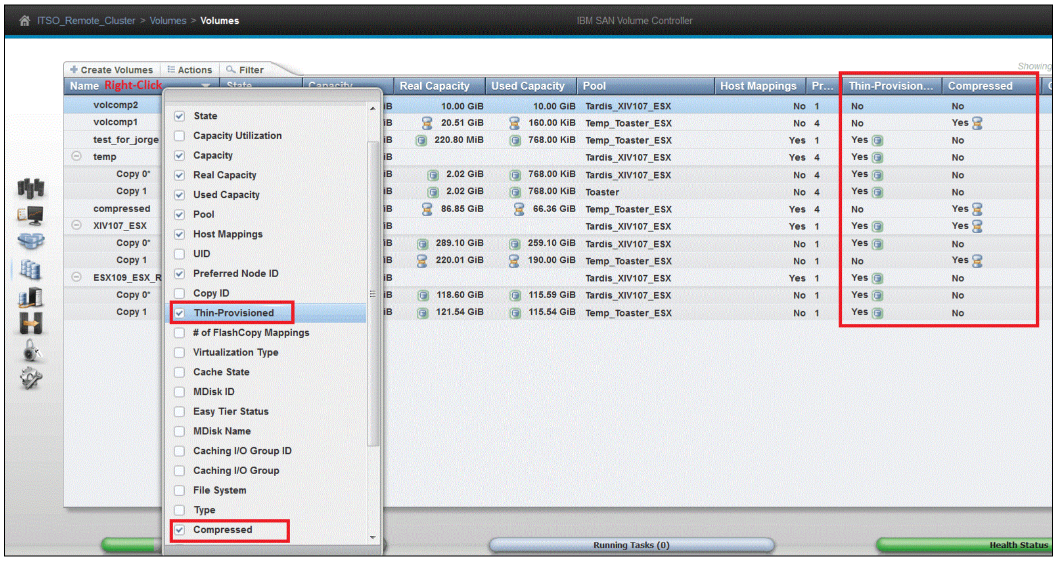
2. Right-click an existing volume of which you want to create a copy (volcomp2 in Figure 5-14) and select **Volume Copy Actions** → **Add Mirrored Copy**. The columns show that **volcomp2** is a generic volume, and not thin-provisioned or compressed.



Name	State	Capacity	Real Capacity	Used Capacity	Pool	Host Mappings	Pr...	Thin-Provision...	Compressed
volcomp2		0 GiB	10.00 GiB	10.00 GiB	Tardis_XIV107_ESX	No	1	No	No
volcomp1		0 TiB	20.51 GiB	160.00 KiB	Temp_Toaster_ESX	No	4	No	Yes
test_for_jorge		0 GiB	220.80 MiB	768.00 KiB	Temp_Toaster_ESX	Yes	1	Yes	No
temp		0 GiB			Tardis_XIV107_ESX	Yes	4	Yes	No
Copy 0*		0 GiB	2.02 GiB	768.00 KiB	Tardis_XIV107_ESX	No	4	Yes	No
Copy 1		0 GiB	2.02 GiB	768.00 KiB	Toaster	No	4	Yes	No
compressed		0 TiB	86.85 GiB	66.36 GiB	Temp_Toaster_ESX	Yes	4	No	Yes
XIV107_ESX		6 TiB			Tardis_XIV107_ESX	Yes	1	Yes	Yes
Copy 0*		6 TiB	289.10 GiB	259.10 GiB	Tardis_XIV107_ESX	No	1	Yes	No
Copy 1		6 TiB	220.01 GiB	190.00 GiB	Temp_Toaster_ESX	No	1	No	Yes
ESX109_ESX_REPL_		0 GiB			Tardis_XIV107_ESX	Yes	1	Yes	No
Copy 0*		0 GiB	118.60 GiB	115.59 GiB	Tardis_XIV107_ESX	No	1	Yes	No
Copy 1		0 GiB	121.54 GiB	115.54 GiB	Temp_Toaster_ESX	No	1	Yes	No

Figure 5-14   Volume copy add

The column names are not shown by default. You can add them by right-clicking anywhere on the column heading and selecting items to be displayed, as shown in Figure 5-15.


ITSO_Remote_Cluster > Volumes > **Volumes**

IBM SAN Volume Controller

Create Volumes | Actions | Filter

Name Right-Click	Real Capacity	Used Capacity	Pool	Host Mappings	Pr...	Thin-Provision...	Compressed
volcomp2	10.00 GiB	10.00 GiB	Tardis_XIV107_ESX	No	1	No	No
volcomp1	20.51 GiB	160.00 KiB	Temp_Toaster_ESX	No	4	No	Yes
test_for_jorge	220.80 MiB	768.00 KiB	Temp_Toaster_ESX	Yes	1	Yes	No
temp			Tardis_XIV107_ESX	Yes	4	Yes	No
Copy 0*	2.02 GiB	768.00 KiB	Tardis_XIV107_ESX	No	4	Yes	No
Copy 1	2.02 GiB	768.00 KiB	Toaster	No	4	Yes	No
compressed	86.85 GiB	66.36 GiB	Temp_Toaster_ESX	Yes	4	No	Yes
XIV107_ESX			Tardis_XIV107_ESX	Yes	1	Yes	Yes
Copy 0*	289.10 GiB	259.10 GiB	Tardis_XIV107_ESX	No	1	Yes	No
Copy 1	220.01 GiB	190.00 GiB	Temp_Toaster_ESX	No	1	No	Yes
ESX109_ESX_R			Tardis_XIV107_ESX	Yes	1	Yes	No
Copy 0*	118.60 GiB	115.59 GiB	Tardis_XIV107_ESX	No	1	Yes	No
Copy 1	121.54 GiB	115.54 GiB	Temp_Toaster_ESX	No	1	Yes	No

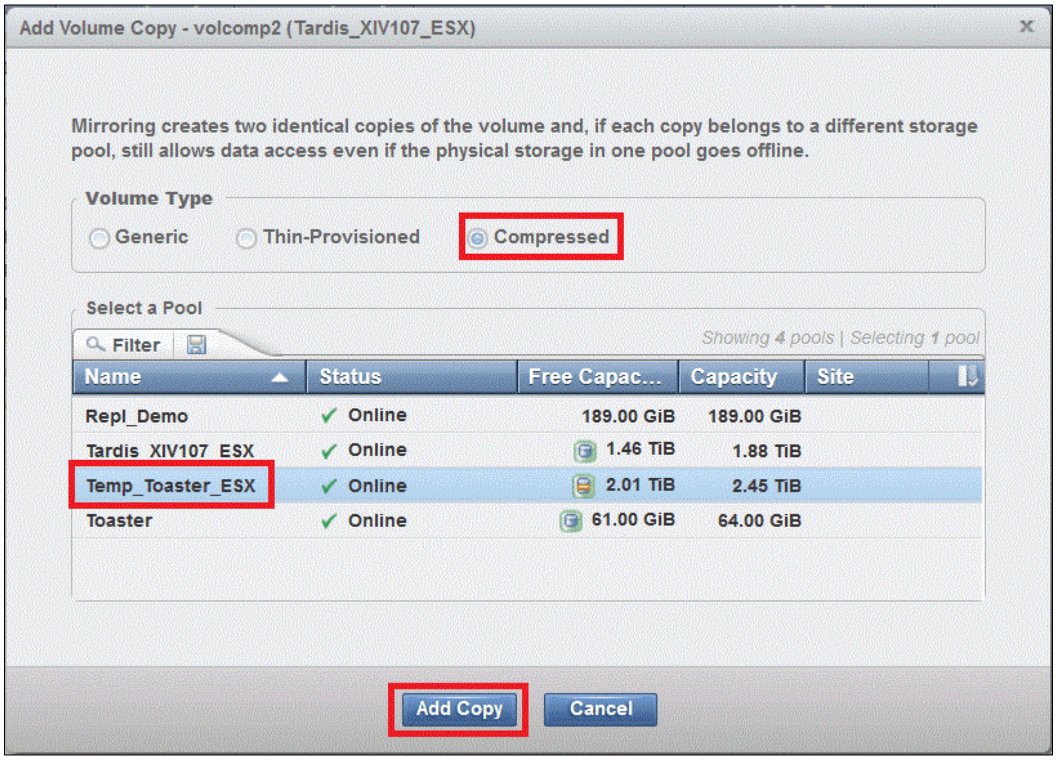
- [✓] State
- [ ] Capacity Utilization
- [✓] Capacity
- [✓] Real Capacity
- [✓] Used Capacity
- [✓] Pool
- [✓] Host Mappings
- [ ] UID
- [✓] Preferred Node ID
- [ ] Copy ID
- [✓] Thin-Provisioned
- [ ] # of FlashCopy Mappings
- [ ] Virtualization Type
- [ ] Cache State
- [ ] MDisk ID
- [ ] Easy Tier Status
- [ ] MDisk Name
- [ ] Caching I/O Group ID
- [ ] Caching I/O Group
- [ ] File System
- [ ] Type
- [✓] Compressed

Running Tasks (0)

Health Status

Figure 5-15 Selecting column display

3. Select the volume type as **compressed,** select the storage pool, and click **Add Copy**, as shown in Figure 5-16.


Add Volume Copy - volcomp2 (Tardis_XIV107_ESX)

Mirroring creates two identical copies of the volume and, if each copy belongs to a different storage pool, still allows data access even if the physical storage in one pool goes offline.

Volume Type

Generic   Thin-Provisioned   [✓ Compressed]

Select a Pool

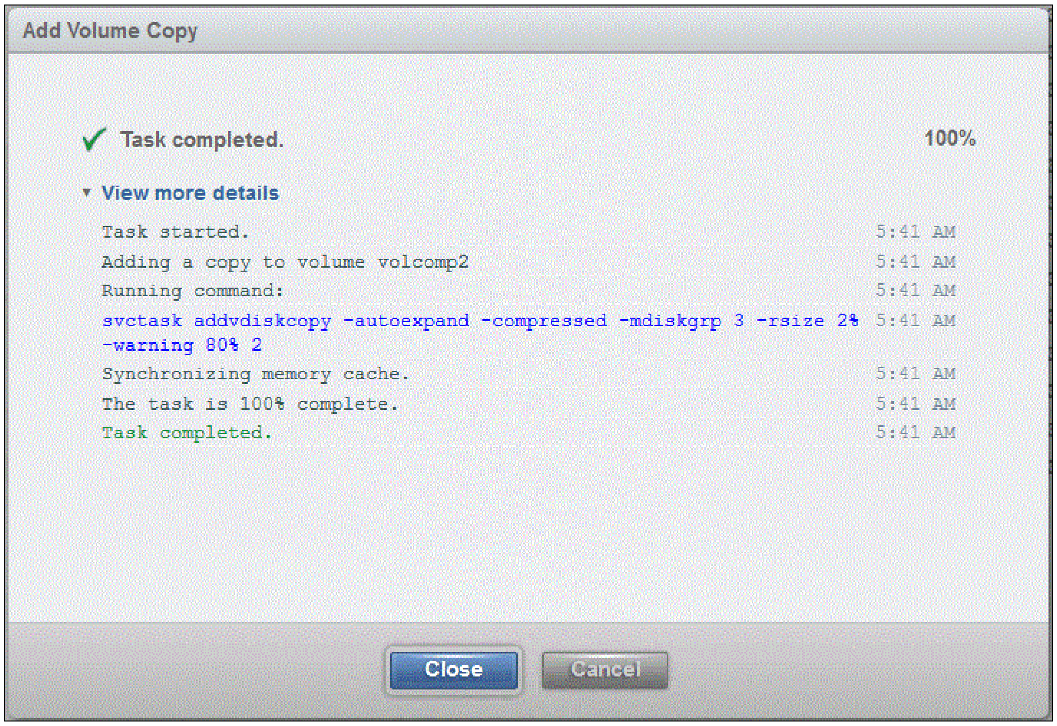
Filter

Showing 4 pools | Selecting 1 pool

Name	Status	Free Capac...	Capacity	Site
Repl_Demo	✓ Online	189.00 GiB	189.00 GiB	
Tardis_XIV107_ESX	✓ Online	1.46 TiB	1.88 TiB	
Temp_Toaster_ESX	✓ Online	2.01 TiB	2.45 TiB	
Toaster	✓ Online	61.00 GiB	64.00 GiB	

Figure 5-16 Add Copy

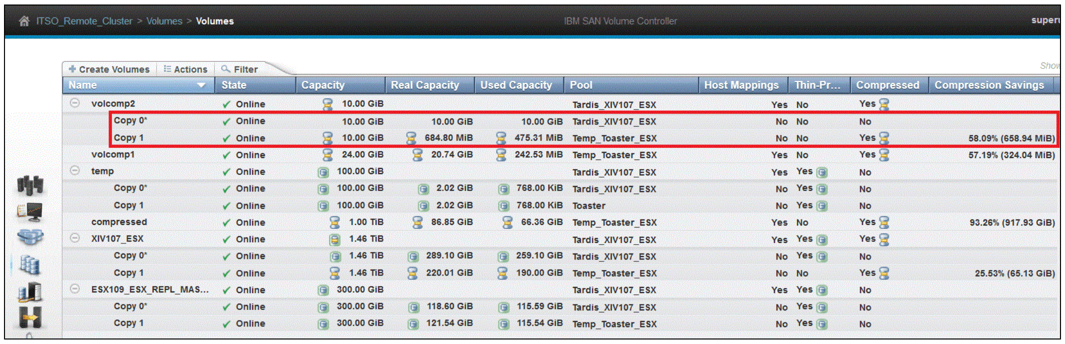
4. The Add Volume Copy process completes, as shown in Figure 5-17.



```
Task started.                                                   5:41 AM
Adding a copy to volume volcomp2                                5:41 AM
Running command:                                                5:41 AM
svctask addvdiskcopy -autoexpand -compressed -mdiskgrp 3 -rsize 2% 5:41 AM
-warning 80% 2
Synchronizing memory cache.                                     5:41 AM
The task is 100% complete.                                      5:41 AM
Task completed.                                                 5:41 AM
```

Figure 5-17 Add Volume Copy completes

5. Figure 5-18 shows two copies of volcomp2, which are 0 and 1. Volume copy 0 is the original generic volume, and 1 is the new compressed version.

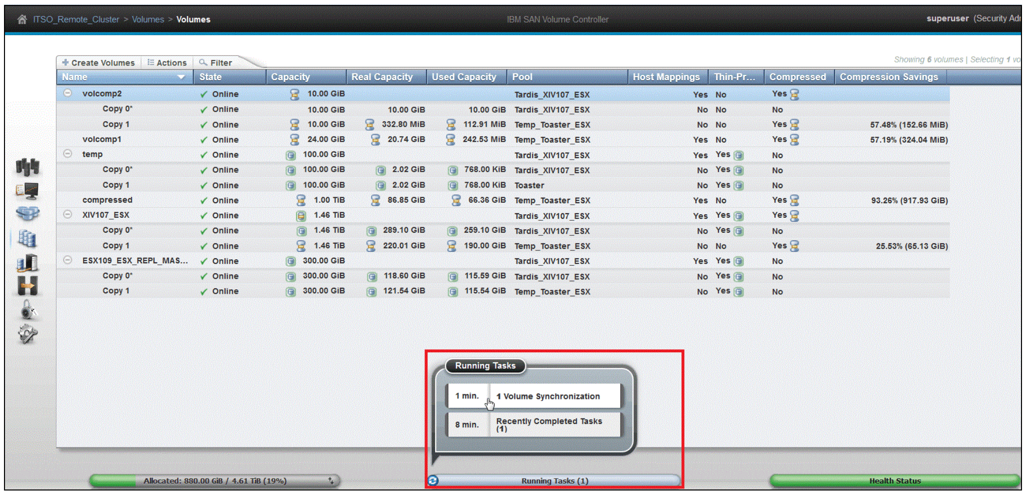

ITSO_Remote_Cluster > Volumes > Volumes
IBM SAN Volume Controller
super
Create Volumes
Actions
Filter
Show

Name	State	Capacity	Real Capacity	Used Capacity	Pool	Host Mappings	Thin-Pr...	Compressed	Compression Savings
volcomp2	✓ Online	10.00 GiB			Tardis_XIV107_ESX	Yes	No	Yes	
Copy 0*	✓ Online	10.00 GiB	10.00 GiB	10.00 GiB	Tardis_XIV107_ESX	No	No	No	
Copy 1	✓ Online	10.00 GiB	684.80 MiB	475.31 MiB	Temp_Toaster_ESX	No	No	Yes	58.09% (658.94 MiB)
volcomp1	✓ Online	24.00 GiB	20.74 GiB	242.53 MiB	Temp_Toaster_ESX	Yes	No	Yes	57.19% (324.04 MiB)
temp	✓ Online	100.00 GiB			Tardis_XIV107_ESX	Yes	Yes	No	
Copy 0*	✓ Online	100.00 GiB	2.02 GiB	768.00 KiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	100.00 GiB	2.02 GiB	768.00 KiB	Toaster	No	Yes	No	
compressed	✓ Online	1.00 TiB	86.85 GiB	66.36 GiB	Temp_Toaster_ESX	Yes	No	Yes	93.26% (917.93 GiB)
XIV107_ESX	✓ Online	1.46 TiB			Tardis_XIV107_ESX	Yes	Yes	Yes	
Copy 0*	✓ Online	1.46 TiB	289.10 GiB	259.10 GiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	1.46 TiB	220.01 GiB	190.00 GiB	Temp_Toaster_ESX	No	No	Yes	25.53% (65.13 GiB)
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB			Tardis_XIV107_ESX	Yes	Yes	No	
Copy 0*	✓ Online	300.00 GiB	118.60 GiB	115.59 GiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	300.00 GiB	121.54 GiB	115.54 GiB	Temp_Toaster_ESX	No	Yes	No	

Figure 5-18 Volume mirror copy

6. You can keep the compressed copy and delete the generic copy after both copies are synchronized. Select the compressed copy as the Primary copy by completing the following steps:

   a. Monitor the synchronization by selecting **Running Tasks** at the bottom of the GUI window, as shown in Figure 5-19. You see the current list of Running Tasks.

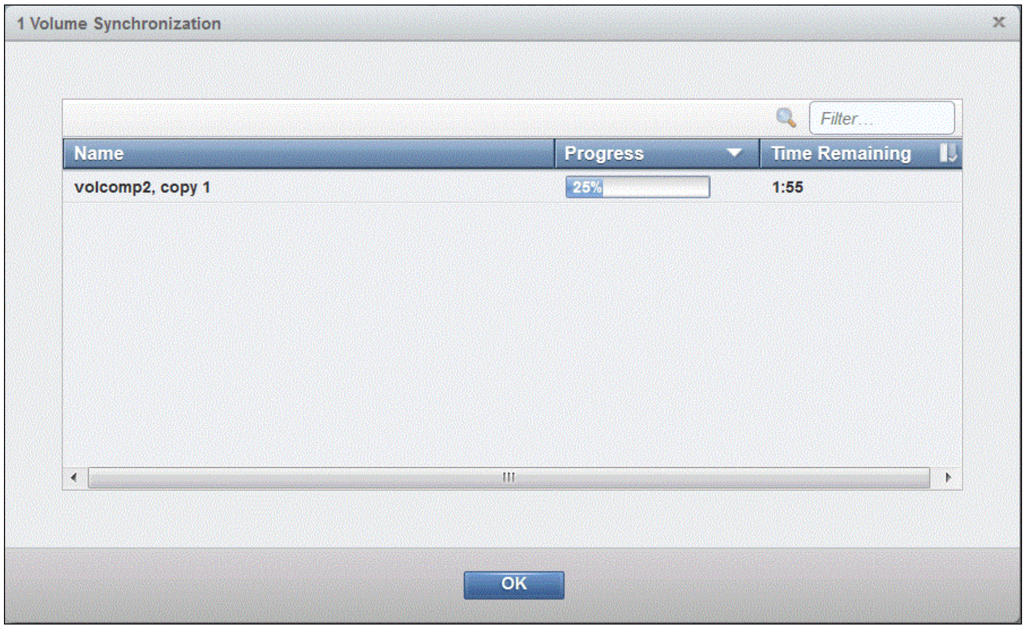


ITSO_Remote_Cluster > Volumes > Volumes
IBM SAN Volume Controller
superuser (Security Adm
Create Volumes
Actions
Filter
Showing 6 volumes | Selecting 1 vo
Running Tasks
1 min.
1 Volume Synchronization
8 min.
Recently Completed Tasks (1)
Allocated: 880.00 GiB / 4.61 TiB (19%)
Running Tasks (1)
Health Status

Name	State	Capacity	Real Capacity	Used Capacity	Pool	Host Mappings	Thin-Pr...	Compressed	Compression Savings
volcomp2	✓ Online	10.00 GiB			Tardis_XIV107_ESX	Yes	No	Yes	
Copy 0*	✓ Online	10.00 GiB	10.00 GiB	10.00 GiB	Tardis_XIV107_ESX	No	No	No	
Copy 1	✓ Online	10.00 GiB	332.80 MiB	112.91 MiB	Temp_Toaster_ESX	No	No	Yes	57.48% (152.66 MiB)
volcomp1	✓ Online	24.00 GiB	20.74 GiB	242.53 MiB	Temp_Toaster_ESX	Yes	No	Yes	57.19% (324.04 MiB)
temp	✓ Online	100.00 GiB			Tardis_XIV107_ESX	Yes	Yes	No	
Copy 0*	✓ Online	100.00 GiB	2.02 GiB	768.00 KiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	100.00 GiB	2.02 GiB	768.00 KiB	Toaster	No	Yes	No	
compressed	✓ Online	1.00 TiB	86.85 GiB	66.36 GiB	Temp_Toaster_ESX	Yes	No	Yes	93.26% (917.93 GiB)
XIV107_ESX	✓ Online	1.46 TiB			Tardis_XIV107_ESX	Yes	Yes	Yes	
Copy 0*	✓ Online	1.46 TiB	289.10 GiB	259.10 GiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	1.46 TiB	220.01 GiB	190.00 GiB	Temp_Toaster_ESX	No	No	Yes	25.53% (65.13 GiB)
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB			Tardis_XIV107_ESX	Yes	Yes	No	
Copy 0*	✓ Online	300.00 GiB	118.60 GiB	115.59 GiB	Tardis_XIV107_ESX	No	Yes	No	
Copy 1	✓ Online	300.00 GiB	121.54 GiB	115.54 GiB	Temp_Toaster_ESX	No	Yes	No	

Figure 5-19 List of Running Tasks

   b. Select the correct Volume Synchronization Task, monitor the synchronization progress, and wait until the task completes (Figure 5-20).



Name	Progress	Time Remaining
volcomp2, copy 1	25%	1:55

Figure 5-20 Synchronization progress

c. After the synchronization is complete (the task completes), select the compressed volume by right-clicking it and select **Make Primary**, as shown in Figure 5-21. This step is nondisruptive to the host.

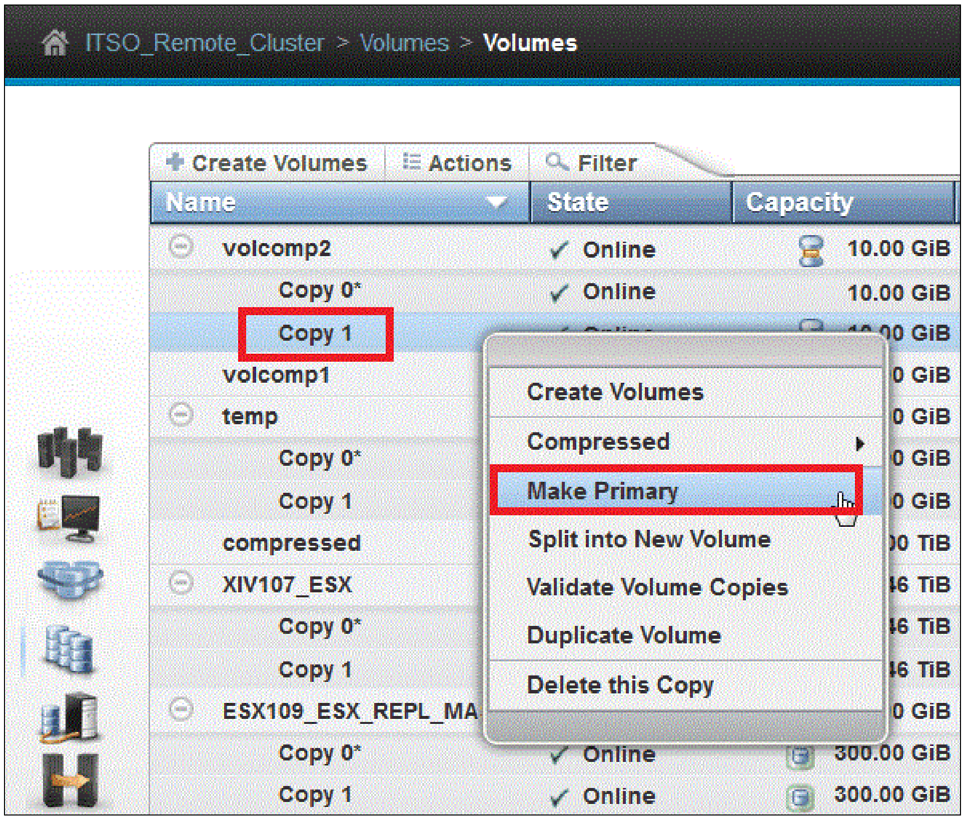
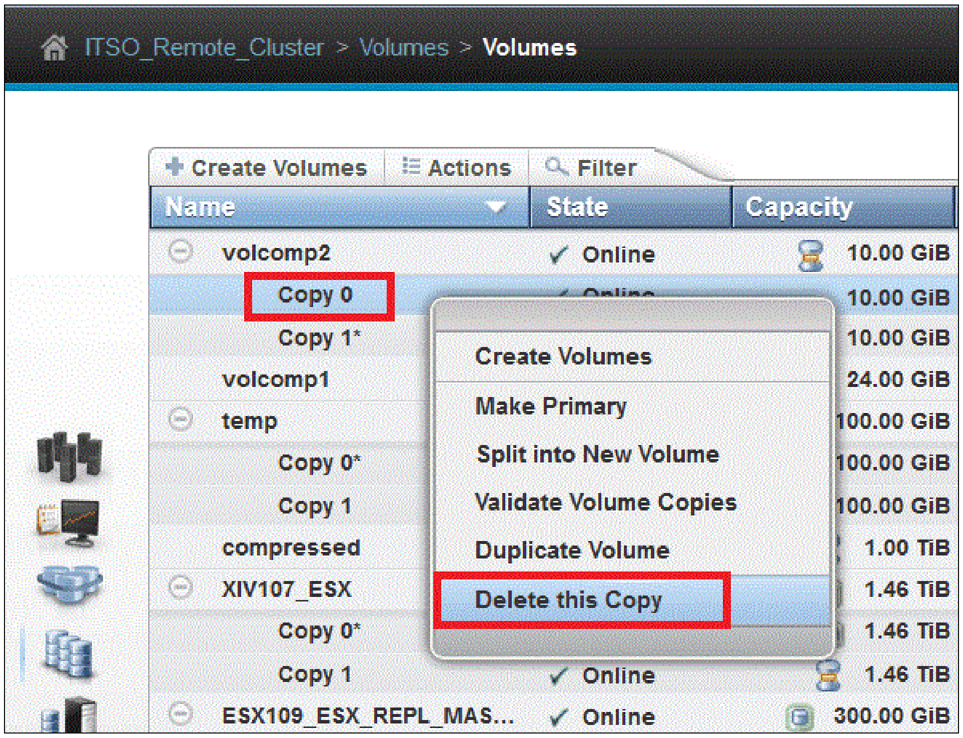


Figure 5-21 Selecting Copy 1 as Primary

d. After you have verified that the new copy is working, you can delete the original generic volume copy, as shown in Figure 5-22. The copy with the asterisk symbol (*) is the Primary.



Name	State	Capacity
volcomp2	✓ Online	10.00 GiB
Copy 0	✓ Online	10.00 GiB
Copy 1*		10.00 GiB
volcomp1		24.00 GiB
temp		100.00 GiB
Copy 0*		100.00 GiB
Copy 1		100.00 GiB
compressed		1.00 TiB
XIV107_ESX		1.46 TiB
Copy 0*		1.46 TiB
Copy 1	✓ Online	1.46 TiB
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB

Figure 5-22 Deleting the volume copy

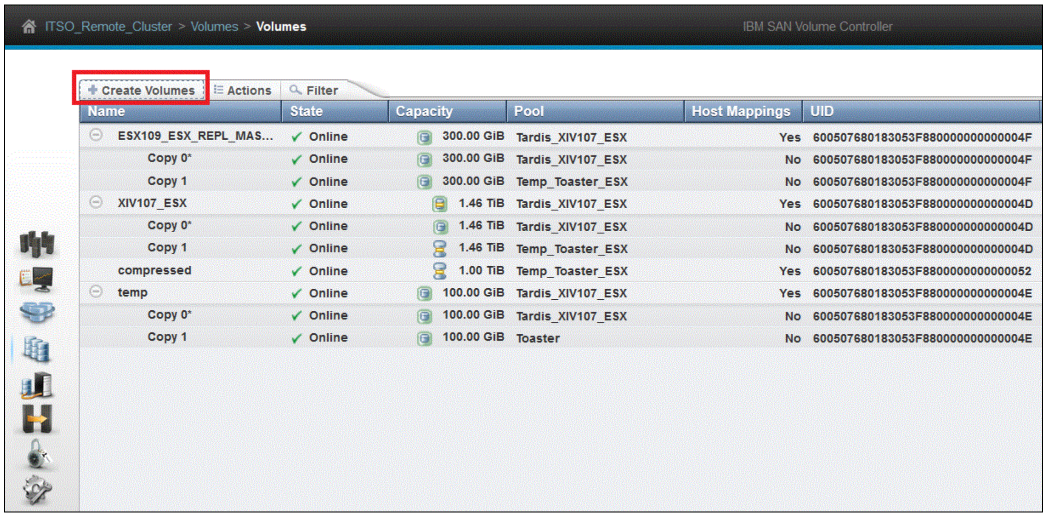
5.3.3 Creating a compressed volume mirror

There is no specific preset to create a compressed mirror, but you can use the Thin Mirror preset to create a compressed mirror.

To create a compressed mirror volume, complete the following steps:

1. Go to the Volumes panel under the SAN Volume Controller or Storwize V7000 Welcome panel, as shown in Figure 5-13 on page 52, and click **Volumes** → **Volumes**.

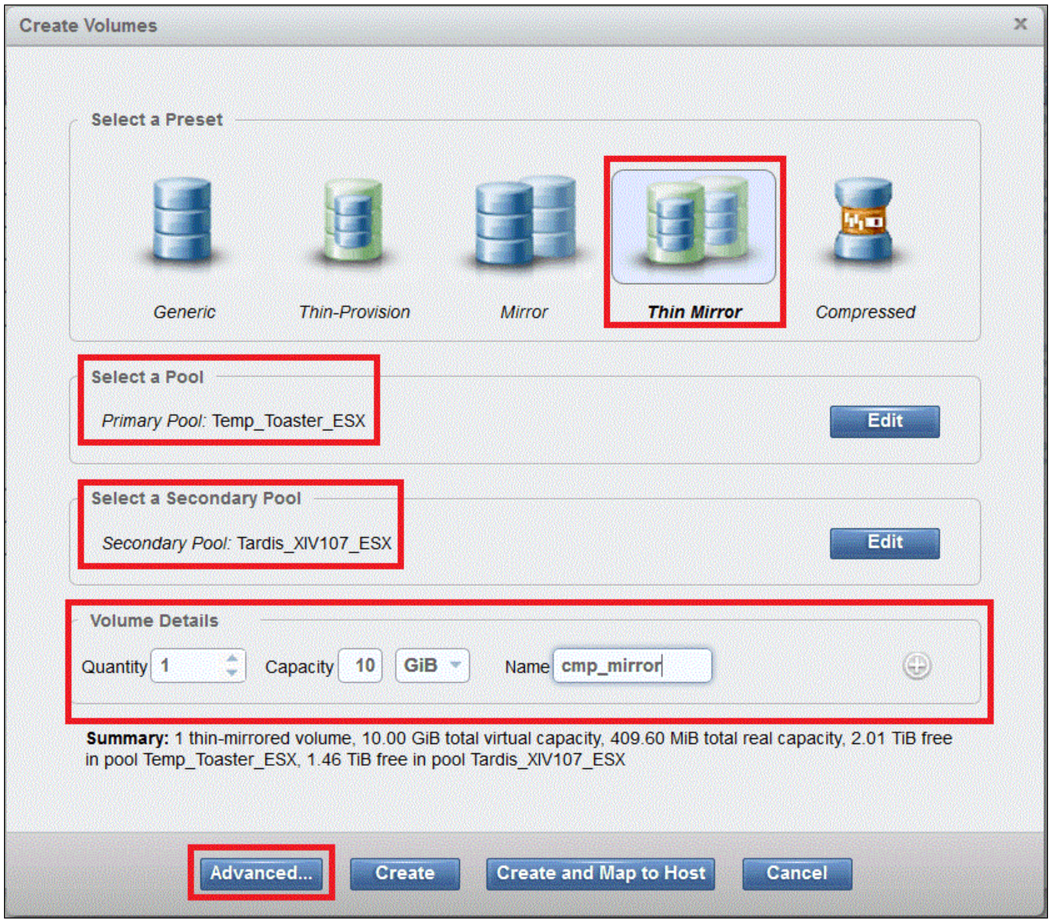
2. Click **Create Volumes**, as shown in Figure 5-23.


ITSO_Remote_Cluster > Volumes > Volumes
IBM SAN Volume Controller
Create Volumes
Actions
Filter

Name	State	Capacity	Pool	Host Mappings	UID
ESX109_ESX_REPL_MAS...	✓ Online	300.00 GiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004F
Copy 0*	✓ Online	300.00 GiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004F
Copy 1	✓ Online	300.00 GiB	Temp_Toaster_ESX	No	600507680183053F880000000000004F
XIV107_ESX	✓ Online	1.46 TiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004D
Copy 0*	✓ Online	1.46 TiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004D
Copy 1	✓ Online	1.46 TiB	Temp_Toaster_ESX	No	600507680183053F880000000000004D
compressed	✓ Online	1.00 TiB	Temp_Toaster_ESX	Yes	600507680183053F8800000000000052
temp	✓ Online	100.00 GiB	Tardis_XIV107_ESX	Yes	600507680183053F880000000000004E
Copy 0*	✓ Online	100.00 GiB	Tardis_XIV107_ESX	No	600507680183053F880000000000004E
Copy 1	✓ Online	100.00 GiB	Toaster	No	600507680183053F880000000000004E

Figure 5-23 Create volumes

3. Select **Thin Mirror** as the preset for the mirror, as shown in Figure 5-24. Select the primary pool, secondary pool, select the volume name and sizes, and finally select **Advanced**.



Primary Pool: Temp_Toaster_ESX

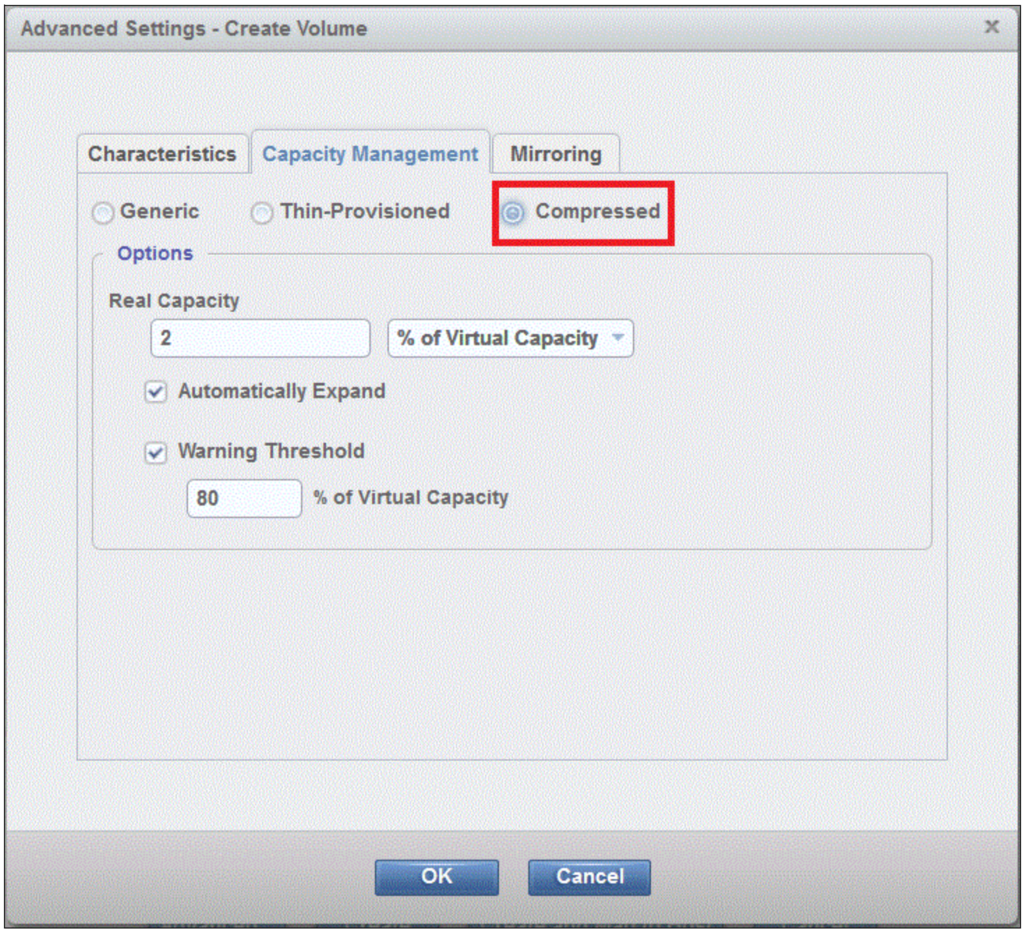
Secondary Pool: Tardis_XIV107_ESX

Quantity 1 Capacity 10 GiB Name cmp_mirror

Summary: 1 thin-mirrored volume, 10.00 GiB total virtual capacity, 409.60 MiB total real capacity, 2.01 TiB free in pool Temp_Toaster_ESX, 1.46 TiB free in pool Tardis_XIV107_ESX

Figure 5-24 Selecting pools, names, and sizes

4. From the Capacity Management tab under the Advanced settings, select **Compressed**, as shown in Figure 5-25, and then click **OK**.


Advanced Settings - Create Volume

Characteristics | Capacity Management | Mirroring

Generic [✓ Compressed] Thin-Provisioned

Options

Real Capacity: 2 % of Virtual Capacity

[✓] Automatically Expand

[✓] Warning Threshold: 80 % of Virtual Capacity

Figure 5-25 Compressed

5. Select **Create**, as shown in Figure 5-24 on page 58, and the mirrored compressed volume is created.

Tip: The preset for Thin Mirror already has the correct defaults for Real Capacity (rsize), Automatically Expand, and Warning Threshold.

5.3.4 Mapping a compressed volume to a host

Mapping is the process of controlling which hosts have access to specific volumes within a system. Volumes are LUN masked to the host's host bus adapter (HBA) worldwide port name (WWPN) by a process called *host mapping*. Mapping a volume to the host makes it accessible to the WWPNs or iSCSI qualified names (IQNs) that are configured on the host object.

Host systems are connected to the SVC and Storwize V7000 through a Fibre Channel or iSCSI connection. A volume can be created and mapped to a host, which then sees it as a LUN. The volume can be compressed or uncompressed. The compression and uncompression is run within the SVC and Storwize V7000 software stack by using IBM Random Access Compression Engine (RACE) technology. The host always sees uncompressed data.

By design, SVC and Storwize V7000 compression is transparent to a host or platform. The host data is compressed before being written to the back-end storage of the SVC and Storwize V7000 by the RACE engine. It is uncompressed after reading from the back-end storage, but before arriving at the host.

The host always observes the virtual size of the volume as opposed to the compressed or uncompressed size. For more information about creating the host object, mapping the volume to a host, and preparing the host to discover the new volume, see the following resources:

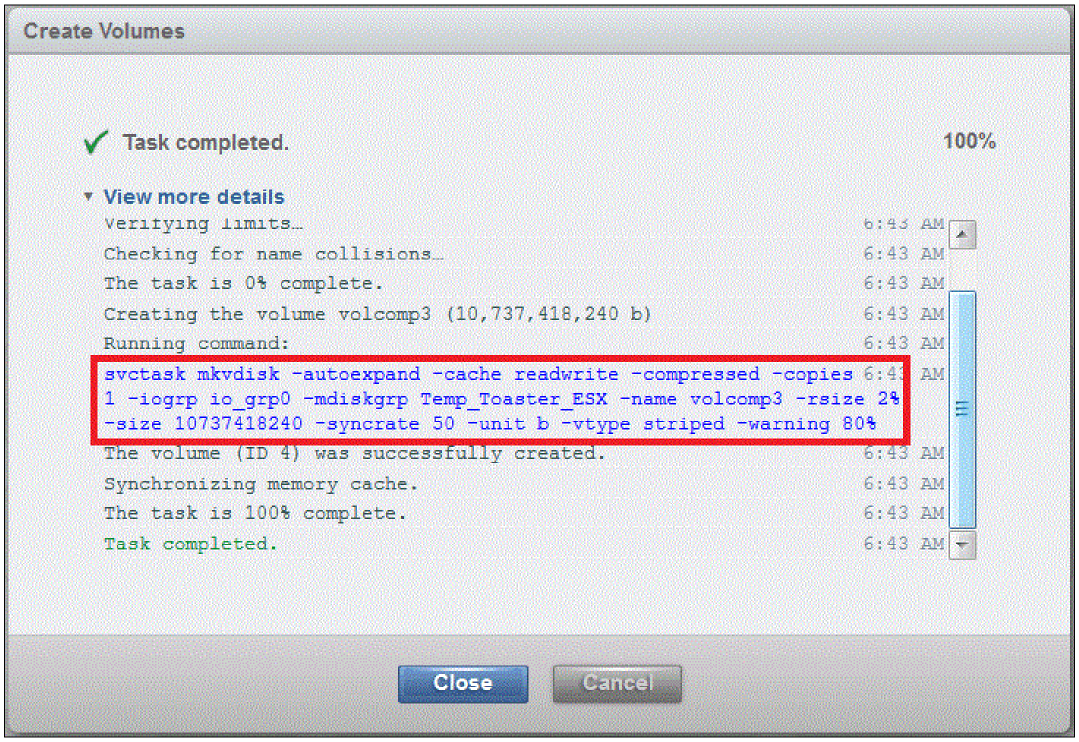
- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

5.4 Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 CLI

A detailed description of all the available CLI commands is beyond the scope of this book. This section addresses the basic commands to create (**`mkvdisk`**) and map (**`mkvdiskhostmap`**) a compressed volume to a host.

Consideration: Compressed volumes have the same characteristics as thin-provisioned volumes. You must specify that the volume is compressed, specify rsize, enable autoexpand, and specify a warning threshold. If not configured properly, the volume can go offline prematurely if the settings are incorrect. The default settings are to set rsize to 2%, enable autoexpand, and set warning to 80%.

The SVC and Storwize V7000 GUI can help you become familiar with the CLI. Examine the details that are posted by the GUI when commands are run, as shown in Figure 5-26.



```
Verifying limits…                                                      6:43 AM
Checking for name collisions…                                          6:43 AM
The task is 0% complete.                                               6:43 AM
Creating the volume volcomp3 (10,737,418,240 b)                        6:43 AM
Running command:                                                       6:43 AM
svctask mkvdisk -autoexpand -cache readwrite -compressed -copies       6:43 AM
1 -iogrp io_grp0 -mdiskgrp Temp_Toaster_ESX -name volcomp3 -rsize 2%
-size 10737418240 -syncrate 50 -unit b -vtype striped -warning 80%
The volume (ID 4) was successfully created.                            6:43 AM
Synchronizing memory cache.                                            6:43 AM
The task is 100% complete.                                             6:43 AM
Task completed.                                                        6:43 AM
```

Figure 5-26 GUI details

5.4.1 Creating a compressed volume

The **mkvdisk** command is used to create sequential, striped, or image mode volume objects. When they are mapped to a host object, these objects are seen as disk drives with which the host can perform I/O operations. There are several types or presets of volumes that can be created, including generic, thin-provision, mirror, thin mirror, and compressed. These types of volumes are described in 5.3.1, “Creating a compressed volume” on page 44.

This section concentrates only on compressed volumes, and also makes special mention of thin-provisioned volumes, which have similar characteristics.

Example 5-3 on page 63 shows how to create a compressed volume. In addition to the normal parameters, you must use the following parameters:

Attention: The following parameters are important. The **`-rsize`**, **`-autoexpand`**, **`-compressed`**, and **`-warning`** parameters must be set properly when a compressed volume is created. These parameters also safeguard against a host losing access to a LUN if the volume goes offline,

`-rsize 2%`	This parameter defines how much physical space is initially allocated to the volume (thin-provisioned or compressed volume). This parameter makes the volume thin-provisioned. Otherwise, the volume is fully allocated. Specify the `disk_size disk_size_percentage` value by using an integer, or an integer immediately followed by the percent character (%). The **`-rsize`** value can be greater than, equal to, or less than the size of the volume. Compressed volumes also have the same characteristics as thin-provisioned volumes, and therefore require this setting, which has a default of 2%.
`-autoexpand`	This parameter specifies that compressed and thin-provisioned volume copies automatically expand their real capacities by allocating new extents from their storage pool. This parameter requires that the **`-rsize`** parameter is also specified. If the **`-autoexpand`** parameter is specified, the **`-rsize`** parameter specifies a capacity that is reserved by the copy. This parameter protects the copy from going offline when its managed disk group runs out of space. It does so by having the managed disk group use this reserved space first. Without this parameter specified, the volume can go offline because it is unable to expand. Therefore, specify this parameter for compression.
`-compressed`	This parameter specifies that the volume type is compressed.
`-vtype striped`	This optional parameter specifies the virtualization type. When creating sequential or image mode volumes, you must also specify the **`-mdisk`** parameter. The default virtualization type is striped. Compressed volumes can be any type.
`-unit gb`	This optional parameter specifies the data units to use in conjunction with the capacity that is specified by the **`-size`** and **`-rsize`** parameters. The unit can be specified as `b | kb | mb | gb | tb | pb`.
`-copies 1`	This optional parameter specifies the number of copies to create. The `num_copies` value can be `1` or `2`. Setting the value to `2` creates a mirrored volume. The default value is `1`.
`-vdisk_name | vdisk_id`	This parameter is required, and specifies the name of the virtual disk that you want to map to the host, either by ID or by name. The **`lsvdisk`** or **`lssevdiskcopy`** command can be used to acquire the `vdisk_id` for the volume/vdisk to be used.

`-warning 80%`	This parameter requires that the **`-rsize`** parameter is also specified. It specifies a threshold at which a warning error log is generated for volume copies. A warning is generated when the used disk capacity on the copy exceeds the specified threshold. You can specify a `disk_size` integer, which defaults to `MBs` unless the **`-unit`** parameter is specified. Or you can specify a `disk_size%`, which is a percentage of the volume size. If **`-autoexpand`** is enabled, the default value for **`-warning`** is 80% of the volume capacity. If **`-autoexpand`** is not enabled, the default value for warning is 80% of the real capacity. To disable warnings, specify `0`. Used with compressed and thin-provisioned volumes.

Tip: When the used capacity first exceeds the warning threshold, an event is raised that indicates that additional real capacity is required. The volume goes offline unless the issue is corrected. You can manually expand the real capacity that is allocated to the volume. It is set at creation time and defaults to 80% of the virtual capacity. The real capacity can be set to an absolute value or a percentage. The alerting facilities (event, alert, or email) are asynchronous, so consider an automated response to this condition. IBM Tivoli Storage Productivity Center alerts provide a mechanism for triggering automated responses. For more information, see the following website:

http://www-03.ibm.com/systems/storage/software/center/index.html

The CLI command that is shown in Example 5-3 creates a compressed 10 GB volume. The volume belongs to the storage pool named Temp_Toaster_ESX, and is owned by the io_grp0 I/O Group. The real capacity automatically expands until the volume virtual size of 10 GB is reached.

Example 5-3 Creating a compressed volume

```
IBM_2145:ITSO_Remote_Cluster:superuser>mkvdisk -autoexpand -compressed -iogrp
io_grp0 -mdiskgrp Temp_Toaster_ESX -name volcomp3 -rsize 2% -size 10 -unit gb
-warning 80%
Virtual Disk, id [4], successfully created
IBM_2145:ITSO_Remote_Cluster:superuser>
```

5.4.2 Creating a compressed volume using addvdiskcopy

The **`addvdiskcopy`** command is used to create a mirrored copy of a volume. This command adds a copy of the chosen volume to the selected storage pool. These types of volumes are described in 5.3.1, “Creating a compressed volume” on page 44.

Remember: There are various command parameters that are important, such as **`-autoexpand`**, **`-rsize`**, and **`-warning`**. For more information about these parameters, see 5.4.1, “Creating a compressed volume” on page 61.

As mentioned previously, compressed volume copies can be freely mixed with fully allocated and regular thin-provisioned (that is, not compressed) volume copies. Volume Mirroring or Adding a Mirrored Copy is available for converting between compressed volume copies and other kinds of volumes. This process can be used to compress or migrate existing data to a compressed volume. The same process can be used to decompress an existing compressed volume.

The first step is to identify the generic volume to be compressed by running the **lsvdisk** command, as shown in Example 5-4. Example 5-4 uses a volume that is named volcomp1, which is uncompressed, and volcomp2, which is compressed.

Example 5-4 Identifying a generic volume to compress

```
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisk -delim :
id:name:IO_group_id:IO_group_name:status:mdisk_grp_id:mdisk_grp_name:capacity:type:FC_id:FC
_name:RC_id:RC_name:vdisk_UID:fc_map_count:copy_count:fast_write_state:se_copy_count:RC_cha
nge:compressed_copy_count:parent_mdisk_grp_id:parent_mdisk_grp_name
0:compressed:0:io_grp0:online:3:Temp_Toaster_ESX:1.00TB:striped:::::600507680183053F8800000
000000052:0:1:not_empty:0:no:1:3:Temp_Toaster_ESX
1:cmp_mirror:0:io_grp0:online:many:many:10.00GB:many:::::600507680183053F8800000000000057:0
:2:empty:0:no:2:many:many
2:volcomp2:0:io_grp0:online:3:Temp_Toaster_ESX:10.00GB:striped:::::600507680183053F88000000
00000054:0:1:empty:0:no:1:3:Temp_Toaster_ESX
3:volcomp1:0:io_grp0:online:3:Temp_Toaster_ESX:24.00GB:striped:::::600507680183053F88000000
00000055:0:1:empty:0:no:0:3:Temp_Toaster_ESX
4:volcomp3:0:io_grp0:online:3:Temp_Toaster_ESX:10.00GB:striped:::::600507680183053F88000000
0000005A:0:1:empty:0:no:1:3:Temp_Toaster_ESX
9:XIV107_ESX:0:io_grp0:online:many:many:1.46TB:many:::::600507680183053F880000000000004D:0:
2:empty:1:no:1:many:many
10:temp:0:io_grp0:online:many:many:100.00GB:many:::::600507680183053F880000000000004E:0:2:e
mpty:2:no:0:many:many
11:ESX109_ESX_REPL_MASTER:0:io_grp0:online:many:many:300.00GB:many:::11:rcrel0:600507680183
053F880000000000004F:0:2:empty:2:no:0:many:many
IBM_2145:ITSO_Remote_Cluster:superuser>
```

The **addvdiskcopy** command can be used to migrate or compress existing generic volumes by using the Adding a Mirrored Copy function. To see how to use the **addvdiskcopy** command to compress an existing uncompressed volume, see Example 5-5. The **addvdiskcopy** command can also be used to migrate or decompress an existing compressed volume by using the Adding a Mirrored Copy function, as shown in Example 5-6.

Example 5-5 Adding a mirror to an uncompressed volume (compressing)

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>addvdiskcopy -autoexpand -compressed -mdiskgrp 3
-rsize 2% -warning 80% 3
Vdisk [3] copy [0] successfully created
IBM_2145:ITSO_Remote_Cluster:superuser>
```

Example 5-6 Adding a mirror to a compressed volume (uncompressing)

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>addvdiskcopy -mdiskgrp 3 2
Vdisk [2] copy [0] successfully created
IBM_2145:ITSO_Remote_Cluster:superuser>
```

There are two copies for the volume after the **addvdiskcopy** command completes, as shown in Example 5-7. Run the **lsvdisk** command and observe the `(copy_count)` field.

Example 5-7 Identifying volume copies

```
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisk -delim :
id:name:IO_group_id:IO_group_name:status:mdisk_grp_id:mdisk_grp_name:capacity:type:FC_id:FC
_name:RC_id:RC_name:vdisk_UID:fc_map_count:copy_count:fast_write_state:se_copy_count:RC_cha
nge:compressed_copy_count:parent_mdisk_grp_id:parent_mdisk_grp_name
0:compressed:0:io_grp0:online:3:Temp_Toaster_ESX:1.00TB:striped:::::600507680183053F8800000
000000052:0:1:not_empty:0:no:1:3:Temp_Toaster_ESX
```

```
1:cmp_mirror:0:io_grp0:online:3:Temp_Toaster_ESX:10.00GB:striped:::::600507680183053F880000
0000000057:0:1:empty:0:no:1:3:Temp_Toaster_ESX
2:volcomp2:0:io_grp0:online:many:many:10.00GB:many:::::600507680183053F8800000000000054:0:2
:empty:0:no:1:many:many
3:volcomp1:0:io_grp0:online:many:many:24.00GB:many:::::600507680183053F8800000000000055:0:2
:not_empty:0:no:1:many:many
4:volcomp3:0:io_grp0:online:3:Temp_Toaster_ESX:10.00GB:striped:::::600507680183053F88000000
0000005A:0:1:empty:0:no:1:3:Temp_Toaster_ESX
9:XIV107_ESX:0:io_grp0:online:many:many:1.46TB:many:::::600507680183053F880000000000004D:0:
2:empty:1:no:1:many:many
10:temp:0:io_grp0:online:many:many:100.00GB:many:::::600507680183053F880000000000004E:0:2:e
mpty:2:no:0:many:many
11:ESX109_ESX_REPL_MASTER:0:io_grp0:online:many:many:300.00GB:many:::11:rcrel0:600507680183
053F880000000000004F:0:2:empty:2:no:0:many:many
IBM_2145:ITSO_Remote_Cluster:superuser>
```

Additional details can be found in Example 5-8 by running the **lsvdisk volcomp1** command. Only pertinent details are shown in the example. You can determine the following details:

- Number of copies (copy_count)
- The copy ID (copy_id)
- Whether the copy is synchronized or not (sync yes/no)
- Which copy is the primary (primary yes/no)
- Whether the copy is compressed (compressed_copy yes/no)

Example 5-8 Obtaining volume details

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisk volcomp2
id 2
name volcomp2
...
sync_rate 100
copy_count 2
...

copy_id 0
status online
sync yes
primary yes
...
used_capacity 10.00GB
real_capacity 10.00GB
free_capacity 0.00MB
overallocation 100
autoexpand
warning
grainsize
se_copy no
easy_tier on
easy_tier_status balanced
tier ssd
tier_capacity 0.00MB
tier enterprise
tier_capacity 10.00GB
tier nearline
tier_capacity 0.00MB
compressed_copy no
uncompressed_used_capacity 10.00GB
...
```

```
copy_id 1
status online
sync yes
primary no
...
used_capacity 476.41MB
real_capacity 684.80MB
free_capacity 208.39MB
overallocation 1495
autoexpand on
warning 80
grainsize
se_copy no
easy_tier on
easy_tier_status balanced
tier ssd
tier_capacity 0.00MB
tier enterprise
tier_capacity 684.80MB
tier nearline
tier_capacity 0.00MB
compressed_copy yes
...
IBM_2145:ITSO_Remote_Cluster:superuser>
```

You can monitor the copy progress by running the **`lsvdisksyncprogress`** command, as shown in Example 5-9. You also can run the **`lsvdisk volcomp2`** monitor for sync (`sync yes`).

Example 5-9 Synchronization progress

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisksyncprogress
vdisk_id vdisk_name copy_id progress estimated_completion_time
0        compressed 1       17       140914112316
2        volcomp2   1       14       140914084324
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisksyncprogress
vdisk_id vdisk_name copy_id progress estimated_completion_time
0        compressed 1       17       140914112316
2        volcomp2   1       44       140914084321
IBM_2145:ITSO_Remote_Cluster:superuser>
```

After the copies are synchronized, you can make the secondary copy the primary copy. Delete the original copy, which is a nondisruptive process to the host.

Run the **`chvdisk`** CLI command, as shown in Example 5-10, to cause the compressed copy (`copy_id 1`) to become the primary copy.

Example 5-10 Making the compressed volume the primary

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>chvdisk -primary 1 2
IBM_2145:ITSO_Remote_Cluster:superuser>
```

You can run the **lsvdisk** command to verify which copy is the primary copy, as shown in Example 5-11. Only pertinent details are shown in this example.

Example 5-11 Verifying which copy is the Primary

```
IBM_2145:ITSO_Remote_Cluster:superuser>lsvdisk volcomp2
id 2
name volcomp2
...

copy_id 0
status online
sync yes
primary yes
...

copy_id 1
status online
sync yes
primary no
...
IBM_2145:ITSO_Remote_Cluster:superuser>
```

The last step, if wanted, is to delete the original volume copy by running the **rmvdiskcopy** CLI command, as shown in Example 5-12.

Example 5-12 Deleting a volume copy

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>rmvdiskcopy -copy 0 2
IBM_2145:ITSO_Remote_Cluster:superuser>
```

One of the optional parameters that can be used with the **addvdiskcopy** command is **-syncrate rate**, which specifies the copy synchronization rate. The default and preferred setting (without specifying the parameter) is 50%. A value of zero (0) prevents synchronization. This parameter can be changed dynamically by running the **chvdisk -syncrate** command, as shown in Example 5-13, or it can be set initially by the **addvdiskcopy** command. Increasing the rate enables the synchronization to complete quicker, but adds an extra load to the system.

Example 5-13 Modifying the synchronization rate

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>chvdisk -syncrate 100 volcomp2
IBM_2145:ITSO_Remote_Cluster:superuser>
```

Table 5-1 shows the relationship between rate and data copied per second. It can also be used to determine how long it takes to migrate from generic to compressed volumes or from compressed to generic.

Table 5-1 Relationship of rate and data copied

User-specified rate attribute value	Data copied/sec
1 - 10	128 KB
11 - 20	256 KB

User-specified rate attribute value	Data copied/sec
21 - 30	512 KB
31 - 40	1 MB
41 - 50	2 MB
51 - 60	4 MB
61 - 70	8 MB
71 - 80	16 MB
81 - 90	32 MB
91 - 100	64 MB

5.4.3 Mapping a compressed volume to a host

The **mkvdiskhostmap** command is used to map a compressed volume to a host. This command creates a mapping between the volume and the specified host. This mapping essentially presents this volume to the host as though the disk was directly attached to the host. It is only after this command is run that the host can perform I/O to the volume.

When the host scans for devices that are attached to it, it discovers all of the volumes that are mapped to its FC or iSCSI ports.

This example assumes that the volume and host definition are already created. You can assign volumes to hosts that are ready for their use by running the **mkvdiskhostmap** command, as shown in Example 5-14.

Example 5-14 Running the mkvdiskhostmap command

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>mkvdiskhostmap -host 2 -scsi 2 4
Virtual Disk to Host map, id [2], successfully created
IBM_2145:ITSO_Remote_Cluster:superuser>
```

The **mkvdiskhostmap** command in Example 5-14 maps the host that is specified by the parameter **-host** (ID is 2, which is host xiv110), as shown in Example 5-15.

Example 5-15 Host name and host ID

```
IBM_2145:ITSO_Remote_Cluster:superuser>
IBM_2145:ITSO_Remote_Cluster:superuser>lshostvdiskmap 2
id name   SCSI_id vdisk_id vdisk_name vdisk_UID                        IO_group_id
IO_group_name
2  xiv110 0       3        volcomp1   600507680183053F8800000000000055 0           io_grp0
2  xiv110 1       2        volcomp2   600507680183053F8800000000000054 0           io_grp0
2  xiv110 2       4        volcomp3   600507680183053F880000000000005A 0           io_grp0
IBM_2145:ITSO_Remote_Cluster:superuser>
```

This mapping can be done by running the **lshostvdiskmap** command or the **lshost** command, as shown in Example 5-16.

Example 5-16 List all hosts

```
IBM_2145:ITSO_Remote_Cluster:superuser>lshost
id name       port_count iogrp_count status
```

```
0  XIV132     2         4          online
1  XIV137     2         4          online
2  xiv110     1         4          online
3  XIV107_esx 2         4          online
4  XIV109_esx 2         4          online
IBM_2145:ITSO_Remote_Cluster:superuser>
```

The **lshost** command also indicates which SCSI IDs are currently in use. The example uses SCSI ID 2, and the host was already using SCSI IDs 0 and 1. The last argument is the name or ID of the volume to be mapped, which you can find by running the **lsvdisk** or **lssevdiskcopy** command (Example 5-17).

Example 5-17   List all volumes (vdisks)

```
IBM_2145:ITSO_Remote_Cluster:superuser>lssevdiskcopy -delim :
vdisk_id:vdisk_name:copy_id:mdisk_grp_id:mdisk_grp_name:capacity:used_capacity:real_capacit
y:free_capacity:overallocation:autoexpand:warning:grainsize:se_copy:compressed_copy:uncompr
es     sed_used_capacity:parent_mdisk_grp_id:parent_mdisk_grp_name
0:compressed:0:3:Temp_Toaster_ESX:1.00TB:66.36GB:86.85GB:20.49GB:1178:on:80::no:yes:984.29G
B:3:Temp_Toaster_ESX
0:compressed:1:3:Temp_Toaster_ESX:1.00TB:20.49GB:40.98GB:20.49GB:2498:on:80::no:yes:303.88G
B:3:Temp_Toaster_ESX
1:cmp_mirror:0:3:Temp_Toaster_ESX:10.00GB:0.16MB:220.80MB:220.64MB:4637:on:80::no:yes:0.00M
B:3:Temp_Toaster_ESX
2:volcomp2:1:6:Tardis_XIV107_ESX:10.00GB:476.28MB:684.80MB:208.52MB:1495:on:80::no:yes:1.11
GB:6:Tardis_XIV107_ESX
3:volcomp1:0:3:Temp_Toaster_ESX:24.00GB:239.28MB:731.52MB:492.24MB:3359:on:80::no:yes:569.8
1MB:3:Temp_Toaster_ESX
4:volcomp3:0:3:Temp_Toaster_ESX:10.00GB:0.16MB:220.80MB:220.64MB:4637:on:80::no:yes:0.00MB:
3:Temp_Toaster_ESX
9:XIV107_ESX:0:6:Tardis_XIV107_ESX:1.46TB:259.10GB:289.10GB:30.00GB:518:on:80:256:yes:no:25
9.10GB:6:Tardis_XIV107_ESX
9:XIV107_ESX:1:3:Temp_Toaster_ESX:1.46TB:190.00GB:220.01GB:30.00GB:681:on:80::no:yes:255.13
GB:3:Temp_Toaster_ESX
10:temp:0:6:Tardis_XIV107_ESX:100.00GB:0.75MB:2.02GB:2.02GB:4960:on:80:256:yes:no:0.75MB:6:
Tardis_XIV107_ESX
10:temp:1:2:Toaster:100.00GB:0.75MB:2.02GB:2.02GB:4960:on:80:256:yes:no:0.75MB:2:Toaster
11:ESX109_ESX_REPL_MASTER:0:6:Tardis_XIV107_ESX:300.00GB:115.59GB:118.60GB:3.01GB:252:on:80
:256:yes:no:115.59GB:6:Tardis_XIV107_ESX
11:ESX109_ESX_REPL_MASTER:1:3:Temp_Toaster_ESX:300.00GB:115.54GB:121.54GB:6.00GB:246:on:80:
256:yes:no:115.54GB:3:Temp_Toaster_ESX
IBM_2145:ITSO_Remote_Cluster:superuser>
```

For more information about the **lssevdiskcopy** command, see Chapter 7, “Compression savings reporting” on page 95.

The hosts must discover or scan for the volumes after the volume is mapped. The hosts must also be properly zoned. This information is not covered here. For more information about configuring your SVC or Storwize V7000, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

5.5 Interaction with other SAN Volume Controller and Storwize V7000 components

You must understand where the RACE technology is implemented in the SVC and Storwize V7000 software stack, which helps to determine how it applies to SVC and Storwize V7000 components. RACE technology is implemented into the SVC and Storwize V7000 thin provisioning layer, and is an organic part of the stack.

The SVC and Storwize V7000 software stack is shown in Figure 5-27.

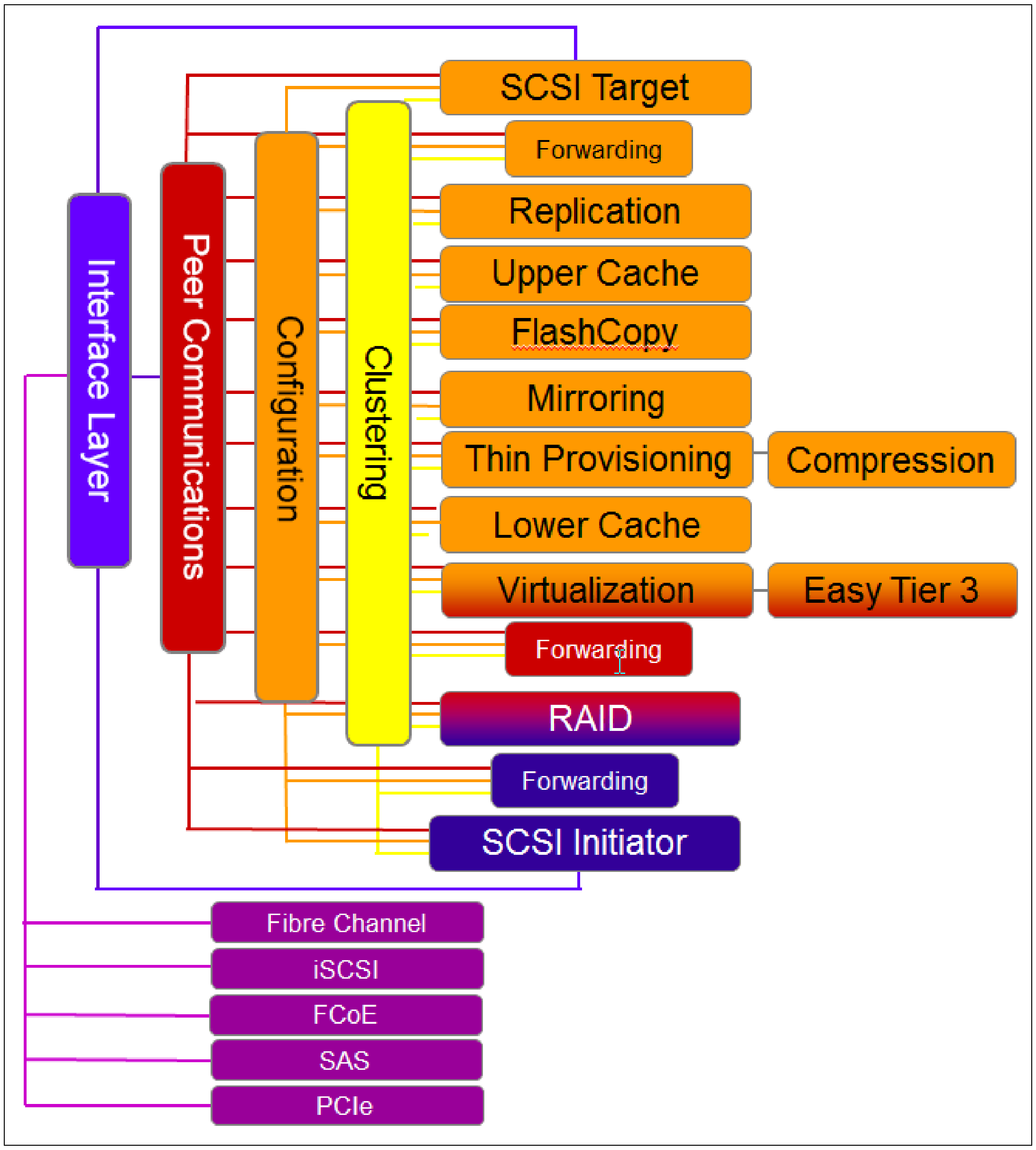


Figure 5-27 SAN Volume Controller and Storwize V7000 software stack

Compression is transparently integrated with existing system management design. All of the SVC and Storwize V7000 advanced features are supported on compressed volumes. You can create, delete, migrate, map (assign), and unmap (unassign) a compressed volume as though it were a fully allocated volume. This compression method provides nondisruptive conversion between compressed and uncompressed volumes. This conversion provides a uniform user-experience and eliminates the need for special procedures to deal with compressed volumes.

Each component is addressed individually and related to compression topics, such as copy services that operate on uncompressed data.

5.5.1 Copy Services

Copy Services functions are implemented within a single SVC or Storwize V7000, or between multiple SVCs or V7000 systems. The Copy Services layer sits above and operates independently of the function or characteristics of the underlying disk subsystems that are used to provide storage resources. Copy services functions are implemented:

- Within an SVC or Storwize V7000 System (FlashCopy and Data Migration)
- Between SVC or Storwize V7000 Systems (Metro Mirror and Global Mirror)

Within the SVC and Storwize V7000, both intracluster copy services functions (FlashCopy and Data Migration) operate at the block level. In intercluster functions (Global Mirror and Metro Mirror), they operate at the volume level.

The layers in the stack (see Figure 5-27 on page 70) for copy services (Remote Copy and FlashCopy) operate where data is uncompressed. The source and destinations (block level or volume) can be compressed or uncompressed. This configuration does not matter to copy services because it operates on uncompressed data.

5.5.2 FlashCopy

FlashCopy is used to create instant copies of virtual disks (volumes) for both the SVC and Storwize V7000. After the copy operation is completed, the target volumes contain the contents of the source volumes as they existed at a single point in time. Although the FlashCopy operation takes some time to complete, the resulting data on the target volume is presented so the copy appears to have occurred immediately.

More advanced FlashCopy functions allow operations to occur on multiple source and target volumes. FlashCopy management operations are coordinated to provide a common, single point in time for copying target volumes from their respective source volumes. This process creates a consistent copy of data that spans multiple volumes. The FlashCopy function also allows multiple target volumes to be copied from each source volume. This function can be used to create images from different points in time for each source volume.

The SVC and Storwize V7000 GUI allow for creation of a *snapshot*, *clone*, or *backup* of an existing volume.

The snapshot is not intended to be an independent copy. Rather, it is used to maintain a view of the production data at the time the snapshot was created. The snapshot holds only the data from regions of the production volume that have changed since the snapshot was created. Because the snapshot preset uses thin provisioning, only the capacity that is required for the changes is used.

The clone preset creates an exact replica of the volume, which can be changed without affecting the original volume. After the copy completes, the mapping that was created by the preset is automatically deleted. The clone has the same preset as the source volume. Therefore, if the source is compressed, the clone is also compressed.

The backup preset creates a point-in-time replica of the production data. After the copy completes, the backup view can be refreshed from the production data with minimal copying of data from the production volume to the backup volume. The backup has the same preset as the source volume. Therefore, if the source is compressed, the backup is also compressed.

5.5.3 Data migration

Migration of a compressed volume from one storage pool to another can be done by running the `migratevdisk` command or the GUI **Migrate to another pool** sequence. The source and destination storage pools must have the same extent size. The `migratevdisk` function does not allow migration from one volume type to another.

Volume mirroring can also be used to migrate a volume between storage pools. This method can be used even if the extent sizes of the two pools are not the same. Volume migration using volume mirroring can be used to migrate a volume from one volume type to another. This process can handle compressed to uncompressed, and uncompressed to compressed.

The process can be prioritized by specifying the number of threads to be used in parallel (1 - 4) while migrating. Using only one thread puts the least background load on the system.

5.5.4 iSCSI

A host system can be alternatively connected to the SVC and Storwize V7000 through a Fibre Channel connection or through an iSCSI connection.

In the simplest terms, iSCSI allows the transport of SCSI commands and data over a Internet Protocol network, based on IP routers and Ethernet switches. iSCSI is a block-level protocol that encapsulates SCSI commands into TCP/IP packets. It uses an existing IP network instead of requiring Fibre Channel HBAs and a SAN fabric infrastructure.

The iSCSI protocol is implemented at the front end of the SVC and Storwize V7000 software stack, as shown in Figure 5-27 on page 70. It can be used whether a volume is compressed or uncompressed.

5.5.5 Storage pools

A storage pool is a collection of storage capacity that provides the capacity requirements for a volume, whether it is compressed or uncompressed. A storage pool is a collection of up to 128 managed disks (MDisks) that provides the pool of storage from which volumes are provisioned. A single system can manage up to 128 storage pools. The size of these pools can be changed (expanded or shrunk) at run time by adding or removing MDisks. You do not need to take the storage pool or the volumes offline to change the size.

Each MDisk in the storage pool is divided into a number of extents. The size of the extent is selected by the administrator at the creation time of the storage pool, and cannot be changed later. The size of the extent is 16 - 8192 MB. Generally, use the same extent size for all storage pools in a system. This configuration is required for volume migration between two storage pools through the `migratevdisk` command. Volume mirroring can also be used to migrate a volume between storage pools. This method can even be used if the extent sizes of the two pools are not the same. Volume mirroring can be used to migrate a volume from one volume type to another. This capability includes compressed to uncompressed, and uncompressed to compressed.

To support the autoexpansion of thin-provisioned and compressed volumes, the storage pools from which they are allocated can have a configurable warning capacity. When the used free capacity of the group exceeds the warning capacity, a warning is logged. For example, if a warning of 80% is specified, the warning is logged when 20% of the free capacity remains.

Single-tiered storage pool

MDisks that are used in a single-tiered storage pool should have the following characteristics:

- They have the same hardware characteristics, such as the same RAID type, RAID array size, disk type, and disk revolutions per minute (RPMs).
- The disk subsystems that provide the MDisks must have similar characteristics, such as maximum input/output operations per second (IOPS), response time, cache, and throughput.
- The MDisks that are used are of the same size and are therefore MDisks that provide the same number of extents. If this configuration is not feasible, you must check the distribution of the volumes extents in that storage pool.

For more information, see *IBM System Storage SAN Volume Controller and Storwize V7000 Best Practices and Performance Guidelines*, SG24-7521.

Multitiered storage pool

A multitiered storage pool has a mixture of MDisks with more than one type of disk tier attribute. An example is a storage pool that contains a mix of generic_hdd and generic_ssd MDisks.

A multitiered storage pool therefore contains MDisks with various characteristics, as opposed to a single-tier storage pool. However, it is a preferred practice for each tier to have MDisks of the same size that provide the same number of extents.

Multitiered storage pools are used to enable the automatic migration of extents between disk tiers by using the SVC and Storwize V7000 Easy Tier function. These storage pools are described in more detail in 5.5.12, “Easy Tier” on page 79.

Compression savings can be viewed at a volume and storage pool level by using the GUI or CLI.

5.5.6 Hosts

Host systems are connected to the SVC and Storwize V7000 through a Fibre Channel or iSCSI connection. A volume can be created and mapped to a host, which sees it as a LUN. The volume can be compressed or uncompressed. The compression and decompression is run within the SVC and Storwize V7000 software stack by using RACE technology. The host always sees uncompressed data.

By design, SVC and Storwize V7000 compression is transparent to a host or platform. The host data is compressed before writing to the back-end storage of the SVC and Storwize V7000 by the RACE engine. It is decompressed after reading from the back-end storage and before arriving at the host.

The host always observes the virtual size of the volume as opposed to the compressed or uncompressed size.

5.5.7 Nodes

On the SVC and Storwize V7000, this is a hardware unit that includes the node hardware, fabric, and service interfaces. The node provides the virtualization for a set of volumes, cache, and copy services functions. The SVC and Storwize V7000 nodes are deployed in pairs, and multiple pairs make up a *clustered system*. A system can consist of 1 - 4 SVC and Storwize V7000 node pairs. Nodes are deployed in pairs that are called *I/O Groups*. Because the nodes are installed in pairs, each node provides a failover function to its partner node if there is a node failure.

One of the nodes within the system is known as the *configuration node*. It is the node that manages configuration activity for the clustered system. If this node fails, the system nominates another node to become the configuration node.

When a host server performs I/O to one of its volumes, all the I/Os for a specific volume are directed to one specific I/O Group in the system. Also, under normal conditions, the I/Os for that specific volume are always processed by the same node within the I/O Group. This node is called the *preferred node* for this specific volume. Thus, in an SVC and Storwize V7000 based environment, the I/O handling for a volume can switch between the two nodes of the I/O Group. For this reason, servers that are connected through Fibre Channel to use multipath drivers must be able to handle these failover situations.

When data is written by the host, the preferred node within the I/O Group saves the data in its cache. Before the cache returns completion to the host, the write must be mirrored to the partner node, or copied in the cache of its partner node, for availability reasons. After a copy is made of the written data, the cache returns completion to the host.

If one node of an I/O Group is missing because of a restart or a hardware failure, the remaining node empties all of its write cache. It then proceeds in an operation mode, which is called *write-through mode*. A node operating in write-through mode writes data directly to the disk subsystem before sending an I/O complete status message back to the host. Running in this mode might decrease the performance of the specific I/O Group.

A SVC node treats part of its physical memory as non-volatile. *Non-volatile* means that its contents are preserved across power losses and resets. Bitmaps for FlashCopy and Remote Mirroring relationships, the Virtualization Table, and the Write Cache are items in the non-volatile memory.

When you create a volume, it is associated with one node of an I/O Group. By default, every time that you create a volume, it is associated with the next node by using a round-robin algorithm. You can specify a preferred access node, which is the node through which you send I/O to the volume instead of using the round-robin algorithm.

For a compressed volume copy, the preferred node is responsible for all reads and writes. If the preferred node fails, the other node in the I/O Group takes over running compression for that volume copy. When the original node recovers, ownership is gracefully transferred back. If there are writes in-flight to compression when the node fails, they are redriven from the write cache on the other node.

Each node of an I/O Group runs the compression code (RACE). In a normal running scenario, Node A performs compression for volumes that have Node A as the preferred node. Similarly, Node B performs compression for volumes that have Node B as the preferred node. If Node A fails or reboots, Node B performs compression for all volumes that have Node A and Node B as the preferred node. When Node A rejoins the cluster, Node A performs compression for volumes that have Node A as the preferred node again. The processing of volumes with preferred Node A is handed to Node B, then automatically transferred back to Node A when Node A comes back online. The same logic applies if Node B fails and later rejoins the cluster.

Memory and processor cores are assigned for compression on both nodes in an I/O Group after the first compressed volume is created. These resources are not configurable. Each node has 2 GB of memory that is allocated to compression after it is enabled. In a single instance RACE scheme, each instance is allocated the 2 GB plus any extra memory that is provided by the type of node being used, such as the SVC V7000 Gen2 and SVC DH8. In a multi-instance RACE scheme, each instance is allocated 1 GB and the extra memory is partitioned by RACE.

The compression code stack is supported on five processor architectures: 4-core (Storwize V7000 Gen1/CF8), 6-core (CG8), 8-core (V7000 Gen2), 12-core (CG8), and 16-core (DH8). On 4-core processors, SVC runs one fast path core and three IBM Real-time Compression cores. On 6-core processors, SVC runs two fast path cores and four IBM Real-time Compression cores. On 8-core processors, SVC runs four fast path cores and four IBM Real-time Compression cores. On 12-core processors, SVC runs four fast path cores and eight IBM Real-time Compression cores. Finally, on 16-core processors, SVC runs eight fast path cores and eight IBM Real-time Compression cores.

Table 5-2 shows the current core allocation.

Table 5-2 Core allocation

	No compressed volumes	**At least one compressed volume**	
	System cores	System cores	Compression cores
Storwize V7000 Gen1 4-cores	4	1	3
Storwize V7000 Gen2 8-cores	8	4	4
SVC CF8	4	1	3
SVC CG8 4-cores	4	1	3
SVC CG8 6-cores	4	2	4
SVC CG8 12-cores RPQ 8S1296	4	4	8
SVC DH8 16-cores	8	8	8

Performance monitoring shows the compression processor resource usage.

Compression does not work if any other type of node is part of an I/O Group.

5.5.8 I/O Groups

Each pair of SVC and Storwize V7000 nodes is also called an *I/O Group*. A SVC or Storwize V7000 clustered system can have 1 - 4 I/O Groups. A specific *volume* is typically presented to a host server by a single I/O Group of the system.

When a host server performs I/O to one of its volumes, all the I/Os for a specific volume are directed to one specific I/O Group in the system. Also, under normal conditions, the I/Os for that specific volume are always processed by the same node within the I/O Group. This node is called the *preferred node* for this specific volume.

Both nodes of an I/O Group act as the preferred node for their own specific subset of the total number of volumes that the I/O Group presents to the host servers. A maximum of 2048 volumes per I/O Group is allowed. However, both nodes also act as failover nodes for their respective partner node within the I/O Group. Therefore, a node takes over the I/O workload from its partner node when required.

Thus, in an SVC-based environment, the I/O handling for a volume can switch between the two nodes of the I/O Group. For this reason, servers that are connected through FC must use multipath drivers to be able to handle these failover situations.

The SVC I/O Groups are connected to the SAN. This connection allows all application servers that access volumes from this I/O Group to have access to this group. Up to 256 host server objects can be defined per I/O Group. The host server objects can access volumes that are provided by this specific I/O Group.

If required, host servers can be mapped to more than one I/O Group within the SVC system. This configuration allows them to access volumes from separate I/O Groups. You can move volumes between I/O Groups to redistribute the load between the I/O Groups.

Version 6.4.0 of the SVC and Storwize V7000 software introduces a new capability to move a volume between I/O Groups to balance the workload without stopping host activity. To do this task, right-click the volume and select **Move to another I/O Group**. You can move a compressed volume if the target I/O Group does not already have the maximum number of compressed volumes. If this is the first compressed volume for the I/O Group, memory and processors are assigned for compression on both nodes in the target I/O Group.

Normally, a pair of nodes from the same I/O Group is physically located within the same rack. Starting with the code release of SVC V6.3, you can split a single system between two physical locations. Splitting the system provides protection against failures that affect an entire location, such as power failures.

There is a maximum number of compressed volumes that are supported for each I/O Group, which is 200 for all models of the SVC and Storwize V7000 Gen1 with the exception of the SAN Volume Controller DH8 and the Storwize V7000 Gen2 models, which support 512 compressed volumes per I/O Group. This maximum applies to the current release, which at the time of writing is SVC V7.4.0. Because a single SVC or Storwize V7000 cluster can have up to four I/O Groups, it can support up to 800 or 2048 compressed volumes depending on the model of the SVC or Storwize V000. The maximum number of compressed volumes per I/O Group limit ensures that compression technology works on single node if there is a node failure in the I/O Group.

Memory and processors are assigned for compression on both nodes in an I/O Group after the first compressed volume is created. This assignment does not affect other I/O Groups.

5.5.9 Clusters

A cluster system consists of 1 - 4 I/O Groups of an SVC or Storwize V7000. Normally, a pair of nodes from the same I/O Group is physically located within the same rack, and is called a cluster. Starting with SVC V6.3, you can split a single cluster or system between two physical locations. This configuration is called a *split-cluster*, and provides protection against failures that affect an entire location, such as power failures.

In simple terms, a *clustered system* is a collection of servers that provide a set of resources to a client. The key point is that the client has no knowledge of the underlying physical hardware of the system. The client is isolated and protected from changes to the physical hardware. This arrangement offers many benefits, which include, most significantly, high availability.

Resources on clustered system act as highly available versions of unclustered resources. If a node (an individual computer) in the system is unavailable or too busy to respond to a request for a resource, the request is passed to another node. The clients are unaware of the exact locations of the resources they are using.

The SVC is a collection of up to eight nodes, which are added in pairs that are known as I/O Groups. The Storwize V7000 is a collection of up to eight node canisters, which are also added in pairs that are known as I/O Groups. These nodes are managed as a set (system), and they present a single point of control to the administrator for configuration and service activity.

The node limit for an SVC and Storwize V7000 system is caused by the microcode. It is not a limit of the underlying architecture. Larger system configurations might be available in the future.

The SVC demonstrated its ability to scale during a 2008 project. The project used a 14-node cluster, which is coupled with solid-state drive (SSD) controllers. It achieved a data rate of over one million IOPS with a response time of under 1 millisecond (ms).

Although the SVC and Storwize V7000 code is based on a purpose-optimized Linux kernel, the clustered system feature is not based on Linux clustering code. The clustered system software that is used within SVC and Storwize V7000 (the event manager cluster framework) is based on the outcome of the COMmodity PArts Storage System (Compass) architecture. Compass was developed at the IBM Almaden Research Center. It is the key element that isolates the SVC and Storwize V7000 application from the underlying hardware nodes. The clustered system software makes the code portable. It provides the means to keep the single instances of the SVC and Storwize V7000 code that run on separate systems nodes in sync. Node restarts (during a code upgrade), adding new nodes, and removing old nodes from a system or node failures do not affect the SVC or Storwize V7000 availability.

It is key for all active nodes of a system to know that they are members of the system. In situations such as the split-brain scenario where single nodes lose contact with other nodes, you must have a solid mechanism to decide which nodes form the active system. Within an SVC and Storwize V7000 system, the *voting set* and a quorum disk are responsible for the integrity of the system. If nodes are added to a system, they are added to the voting set. If nodes are removed, they are also quickly removed from the voting set. Over time, the voting set, and thus the nodes in the system, can change. Therefore, the system can migrate onto a different set of nodes from the set on which it started.

The SVC clustered system implements a dynamic quorum. Following a loss of nodes, if the system can continue operation, it adjusts the quorum requirement so that further node failure can be tolerated.

The lowest Node Unique ID in a system becomes the boss node for the group of nodes. This boss node determines (from the quorum rules) whether the nodes can operate as the system. This node also presents the maximum two-cluster IP addresses on one or both of its node's Ethernet ports to allow access for system management.

There are no special considerations that are required to configure a cluster to support compression. There are some considerations when a *stretched cluster* is sized because it has some impact with compressed volumes and can consume extra processor resources to maintain mirrored copies for each distant node. The extra number of IOPS that is generated to maintain these mirrored copies between the two nodes in the I/O Group must be considered when sizing a stretched cluster solution.

5.5.10 Redundant array of independent disks

A redundant array of independent disks (RAID) combines two or more physical disk drives in an array to incorporate a RAID level for either failure protection or better performance. The most common RAID levels are 0, 1, 5, 6, and 10.

Usually, the SVC connects to external storage RAID controllers. The RAID level for these external controllers is configured at the controller level, and is independent of the SVC.

External storage is MDisks that are SCSI logical units that are presented by external storage systems that are attached to and managed by the clustered system.

Internal storage is array MDisks and drives that are held in enclosures and nodes that are part of the clustered system.

The Storwize V7000, unlike the SVC, has its own internal storage that requires RAID configuration. Both SVC and Storwize V7000 can also have SSD storage, which also requires RAID configuration. Internal SSDs are used, and are displayed as disk drives. Therefore, additional RAID protection is required. The SSDs can be internal or external. The preferred use for SSDs is to enable Easy Tier. For more information about Easy Tier, see 5.5.12, "Easy Tier" on page 79.

The RAID level does not affect compression.

5.5.11 Managed disks

A MDisk is a component of a storage pool that is managed by a clustered system. An MDisk is either part of a RAID array of internal storage or a SCSI logical unit (LU) for external storage. An MDisk is not visible to a host system on the storage area network.

The MDisks are placed into storage pools where they are divided up into a number of extents. These extents are 16 - 8182 MB, as defined by the SVC and Storwize V7000 administrator. They cannot be changed later. A volume is host-accessible storage that is provisioned out of one *storage pool*, or if it is a mirrored volume, out of two storage pools.

An MDisk has four modes:

- Array mode MDisks are constructed from drives by using the RAID function. Array MDisks are always associated with storage pools.
- Unmanaged MDisks are not being used by the system, and are not part of a storage pool. This situation occurs when an MDisk is first imported into the system and has not been assigned to a storage pool.

- Managed MDisks are assigned to a storage pool and provide extents so that volumes can use it.
- Image MDisks are assigned directly to a volume with a one-to-one mapping of extents between the MDisk and the volume. This situation is normally used when importing logical volumes that already contain data into the clustered system. It ensures that the data is preserved as it is imported into the clustered system.

5.5.12 Easy Tier

Easy Tier is a performance function that automatically migrates extents off a volume from one MDisk storage tier to another MDisk storage tier. Easy Tier monitors the host I/O activity and latency on the extents of all volumes that have the Easy Tier function turned on in a multitier storage pool. This monitoring covers a 24-hour period.

Easy Tier creates an extent migration plan based on this activity, and then dynamically moves high activity or hot extents to a higher disk tier within the storage pool. It also moves extents whose activity has dropped off or cooled from the high-tier MDisks back to a lower-tiered MDisks.

To experience the potential benefits of using Easy Tier in your environment before installing SSDs, turn on the Easy Tier function for a *single-level* storage pool. Next, turn on the Easy Tier function for the volumes within that pool. Easy Tier then starts monitoring activity on the volume extents in the pool.

For a more effective use of SSDs, place the SSD MDisks into a multitiered storage pool that is combined with HDD MDisks (generic_hdd tier). Then, turn on Easy Tier so that it automatically detects and migrates high workload extents onto the SSD MDisks.

IBM Real-time Compression Software is embedded in the SVC and Storwize V7000. Compressed volumes have a unique write pattern to the MDisks. When a host writes data to a certain offset of a compressed volume, the system compresses this data, which is then written to another offset of the underlying volume as it is represented in the storage pool. Such a change in offsets triggers unnecessary migrations of data into SSDs because repetitive writes to the same logical offset end up written in various locations instead. A new Easy Tier algorithm is required to support compression.

Starting with Version 7.1 of the SVC and Storwize V7000 software, Easy Tier supports compressed volumes. A new algorithm is implemented to monitor read operations on compressed volumes instead of reads and writes. The extents with the highest number of read operations smaller than 64 KB are migrated to SSD MDisks. As a result, frequently read areas of the volumes are serviced from SSDs. Only some of the write operations are sent to the extents stored on SSDs, speeding up those writes.

Easy Tier on non-compressed volumes is still operating as before, that is, based on read and write operations smaller than 64 KB.

The performance improvement that is achieved with Easy Tier and compression is up to three times reduction in response time.

5.5.13 Regular or generic volumes

Volumes are logical disks that are presented to the host or application servers by the SVC and Storwize V7000. The hosts cannot see the MDisks. They can see only the logical volumes that are created from combining extents from a storage pool.

There are three types of volumes: striped, sequential, and image. These types are determined by how the extents are allocated from the storage pool:

- A volume that is created in striped mode has extents that are allocated from each MDisk in the storage pool in a round-robin fashion.
- With a sequential mode volume, extents are allocated sequentially from an MDisk.
- Image mode is a one-to-one mapped extent mode volume.

Using striped mode is the best method to use for most cases. However, sequential extent allocation mode can slightly increase the sequential performance for certain workloads.

Figure 5-28 shows striped volume mode and sequential volume mode, and illustrates how the extent allocation from the storage pool differs.

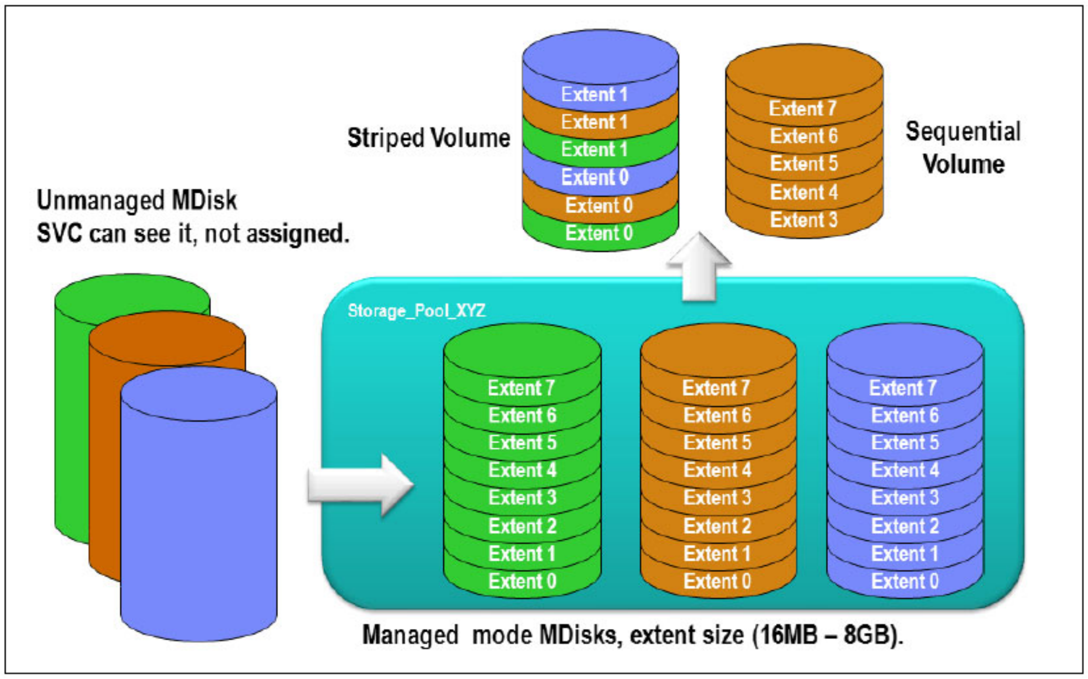


Figure 5-28 Storage pool extents

A regular or generic volume that is uncompressed can be migrated to a compressed volume by using volume mirroring.

5.5.14 Thin-provisioned volumes

A thin-provisioned volume can have a large capacity but uses only the capacity that is written by the host.

Compressed volumes are similar to thin-provisioned volumes in that they use only physical storage to store the compressed version of the volume. Both types of volumes share features such as “autoexpand”. The similarities between thin-provisioning and compression are because the RACE technology is implemented within the thin-provisioning technology in the SVC and Storwize V7000 software stack (see Figure 5-27 on page 70). Performance for both types of volumes is similar.

5.5.15 Mirrored volumes

The SVC and Storwize V7000 provides a function that is called *volume mirroring*, which enables a volume to have two physical copies. Each volume copy can belong to a different storage pool and can be on different physical storage systems, which provide a highly available solution.

When a host system issues a write to a mirrored volume, the SVC or Storwize V7000 writes the data to both copies. When a host system issues a read to a mirrored volume, the SVC or Storwize V7000 retrieves the data from the primary copy. If one of the mirrored volume copies is temporarily unavailable, the SVC and Storwize V7000 automatically uses the alternative copy. This process occurs without any outage to the host system. When the mirrored volume copy is repaired, the SVC or Storwize V7000 resynchronizes the data.

A mirrored volume can be converted into a non-mirrored volume by deleting one copy or by splitting one copy to create a new non-mirrored volume. This method can be used to convert a compressed volume to an uncompressed volume, or uncompressed to compressed.

The mirrored volume copy can be any type: image, striped, sequential, and thin-provisioned, compressed or not. The two copies can be different volume types.

Using mirrored volumes can also assist with migrating volumes between storage pools that have different extent sizes. Mirrored volumes can also provide a mechanism to migrate fully allocated volumes to thin-provisioned or compressed volumes without any host outages.

An unmirrored volume can be migrated by adding a second copy at the wanted destination, waiting for the two copies to synchronize, and removing the original copy. This operation can be stopped at any time. Again, this method is useful to convert between compressed and uncompressed volumes.

5.5.16 Compressed volumes

Version 6.4.0 of the SVC and Storwize V7000 software introduces a new capability to configure compressed volumes. Compressed volumes are a new type of thin-provisioned volumes. Compressed volumes use RACE technology to deliver IBM Real-time Compression capabilities in the SVC and Storwize V7000 product lines.

Compressed volumes are similar to thin-provisioned volumes in that they use only physical storage to store the compressed version of the volume. Both types of volumes share features such as “autoexpand”. The similarities between thin provisioning and compression are because the RACE technology is implemented within the thin-provisioning technology layer in the SVC and Storwize V7000 software stack (see Figure 5-27 on page 70).

Some of the CLI and GUI displays show a compressed size that reflects the combined savings for both compression and thin provisioning.

5.5.17 IBM Tivoli Storage Productivity Center for Replication

For more information about Tivoli Storage Productivity Center for Replication, see the following resources:

- *IBM System Storage SAN Volume Controller and Storwize V7000 Best Practices and Performance Guidelines*, SG24-7521
- *IBM Tivoli Storage Productivity Center V5.1 Technical Guide*, SG24-8053

Some related functions are already available, such as Tivoli Storage Productivity Center for Replication reports for space usage over time, which is valid both for thin-provisioned and compressed volumes. Thin-provisioned volumes are currently supported by Tivoli Storage Productivity Center for Replication. SVC and Storwize V7000 processors monitoring is already in SVC node statistics.

For more information that can be used to collect information about compressed volumes, see Chapter 7, “Compression savings reporting” on page 95.

5.5.18 IBM Storage Management Console for VMware vCenter

The IBM Storage Management Console for VMware vCenter Version 3.1.0 has support for volume data compression. You can create a volume and specify compression directly from vCenter and view information in the LUN details pane, which indicates whether the data is compressed. Versions before Version 3.1.0, such as Version 2.6.0, did not support compressed volumes directly. However, it is possible to use the console to create a fully allocated or thin-provisioned volume and then convert it to a compressed volume by using volume mirroring. For more information, see 5.3, “Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 GUI” on page 43 and 5.4, “Configuring compressed volumes using the SAN Volume Controller and Storwize V7000 CLI” on page 60.

In addition, there is no reporting of compression savings from the console itself, so you must use the SVC or V7000 for reporting instead.

6

Performance guidelines

This chapter addresses system performance when using IBM Real-time Compression in IBM Storwize V7000 and IBM SAN Volume Controller (SVC). In addition, it provides guidelines about how to measure correctly performance by using benchmarks, and describes various considerations and guidelines when configuring and using compression.

This chapter includes the following sections:

- Overview
- Understanding compression performance
- Application benchmark results
- Standard benchmark tools
- Sequential read and write I/O
- Compression hardware assignments
- Monitoring processor usage for compression
- File system alignment
- Performance guidelines summary

6.1 Overview

IBM Real-time Compression is a storage efficiency feature that is unique in the sense that it provides dramatic capacity savings while delivering best-in-class performance for enterprise application workloads. Compression runs well when the following conditions are met:

- Data is compressible (as described in 4.2.3, “Compressible data” on page 28).
- It is a real application workload (as described in 1.3, “Common use cases” on page 3 and 4.1, “Candidate data sets for compression” on page 26).
- The Storwize V7000 or SVC system has enough processor resources to sustain the entire workload (as described in 6.6, “Compression hardware assignments” on page 89).

The following sections provide more information about how IBM Real-time Compression is tuned to support real application workloads, and how well it runs in those workloads. In addition, they address the assignment of hardware resources for compression, and provide recommendations on file system alignment.

6.2 Understanding compression performance

This section addresses the basics of compression, its relationship to performance, and the effects from differing types of workloads, such as real and synthetic workloads. It also covers benchmarking and the available tools that can be used to provide workloads and measure performance.

6.2.1 Compression I/O patterns and temporal locality

IBM Real-time Compression compresses a data stream as it is written, as described in Chapter 3, “IBM Real-time Compression” on page 11. Because of temporal and then spatial locality, an incoming write stream turns into a sequential stream of contiguous physical managed disk (or arrays in Storwize V7000) logical block addresses. This process occurs even if the incoming write stream is random and made up of non-contiguous volume logical block addresses

Thus, any random small block I/O write stream is coalesced into a single chunk of data to be compressed. The compressed block is then written out to disk, which contains the sequential stream of the larger block I/Os.

In real-life applications, when this data is read back, the read stream generally follows the same random (non-contiguous volume logical block address) pattern. Thus, the compression engine reads and extracts the larger chunk of data, which results in the next few random volume I/O reads by the host. This data is read from the data that has already been extracted by extracting the first large chunk. This process therefore results in what is essentially a cache hit in the compression cache memory.

With real-world applications, there is also, in general, no such thing as truly random I/O. The reality is that an application reads and writes objects or groups of data. These groups of I/O requests form a repeatable pattern, with the same group of I/O occurring one after another, even if they are to random locations on disk. IBM has invested heavily in understanding these patterns, and IBM Real-time Compression uses this understanding to give better compression ratios and return the best performance.

Various application benchmark tools have shown that the compression performance in most cases is as good or better than thin-provisioned performance for an equivalent number of disks.

Consideration: The compression engine can reduce the number of disk I/O operations compared to non-compressed and thin-provisioned volumes. However, a workload that is truly 100% random read and write patterns results in I/O amplification rather than I/O reduction. I/O amplification is associated with extra I/O requests and is expressed as a ratio of the number of host I/O requests and the number of storage I/O requests. An example of this ratio is if every host I/O requires one directory I/O and one data I/O, where the I/O amplification is 1:2. The lower the number of storage I/O requests is, the better overall performance is with compression because you are storing more data on fewer drives, and hence you have already increased the number of I/O requests per drive. So, it is even more important to keep the number of storage I/O requests as low as possible. This I/O amplification is typically observed in generic benchmark tools and not in real application workloads.

6.2.2 Example of a real application workload

As an example of "temporal locality", imagine an email server that receives an email for user *fred*. This email contains a large 20 MB attachment.

Tip: This section addresses the I/O ordering pattern, and not the "compressibility" of the data itself.

The email server first appends this email and attachment to fred's email database. This results in a series of I/O writes to the underlying store:

- Some small database style transactional updates that detail the metadata that is associated with the email
- The text content of the email
- The attachment file in binary format

The data writes can all pass through an operating system, which potentially randomizes the logical block addresses at which each block of data is written. After the I/O are submitted in SCSI block writes to the storage system, it can also further randomize where the actual data is written to disk.

The write I/O pattern is now semi-random, but it has a temporal signature: first, the metadata, then the email text, and lastly the attachment. Within each one of these operations, the data is also semi-randomized by the file system and the disk system.

The compression engine takes these I/O requests one by one and appends them into the compression chunks, maintaining the temporal signature. It compresses the data and writes them sequentially down to disk.

Some time later, fred logs in to the email client, and requests to view the email. The reads submitted by the email server first look for the metadata, then the email contents, and then the link to the attachment. These reads have the same temporal signature given that the file system and disk system must retrieve the same data as was written.

The compression engine reads out the compressed chunks, and decompresses them. It now has more decompressed data in memory than was initially requested. The ensuing I/O requests that are submitted to the compression engine should reference the additional extracted data in memory.

The same process applies if fred then opens the attachment.

6.2.3 Benchmarking compression performance

To benchmark a compressed volume, either test your actual application workloads, or install a specific benchmark tool that truly simulates the application I/O and data patterns accurately. Available tools that can be used to benchmark compression performance include VMmark, TPC-C (`http://www.tpc.org/tpcc/`), and TPC-H (`http://www.tpc.org/tpch/`).

Consideration: Ideally, you use a copy or clone of your real application data, and a sample workload.

You can use volume mirroring within the Storwize V7000 and SVC products to clone and split off a compressed copy of real application data. You can also then use the new volume move between I/O Groups function to migrate the cloned volume to a new I/O Group.

In the course of writing this book, various performance benchmark tests were run using typical Storwize V7000 and SVC system configurations. The results are included in 6.3, “Application benchmark results” on page 86.

6.3 Application benchmark results

The following section contains the results of performance benchmark testing in this book by using typical Storwize V7000 Gen 2 and SAN Volume Controller 2145-DH8 system configurations.

6.3.1 Benchmark testing configurations

The main user benefit of compression is data reduction. The testing configurations that were used for the benchmarks in this book had the configurations that are shown in Table 6-1.

Table 6-1 Disk configurations that were used during benchmarks

Model	Configuration
Storwize V7000 Gen 2 - CSOP Tests	168x 10K RPM 600 GB SAS disks configured as 7+P RAID 5 64 x 300 GB volumes
Storwize V7000 Gen 2 - VDbench Tests	144x 10K RPM 600 GB SAS disks configured as RAID 10, 64 GB 64 GB RAM with 2 compression accelerator cards 128 x 200 GB volumes 4 FC ports per node
SAN Volume Controller DH8 - VDbench Tests	64 GB RAM with 2 compression accelerator cards 128 x 200 GB volumes 8 FC ports per node XIV Gen 3 Full System

Model	Configuration
Storwize V7000 Gen 2 - Application Tests	64 GB RAM with 2 compression accelerator cards
SAN Volume Controller DH8 - Application Tests	64 GB RAM with 2 compression accelerator cards TMS840, 64 300 GB volumes **Note:** This configuration yields the equivalent performance with RtC when using the V840.

6.3.2 Oracle online transaction benchmark

This benchmark uses Oracle to simulate an order-entry application by running a mixture of read-only and update-intensive transactions typical of online transaction processing (OLTP) environments. The benchmark incorporated five transaction types (for example, New Order, Delivery, and Payment). Throughput is measured in transactions per minute. The benchmark also reports the response time per transaction, which is broken out by transaction type.

The benchmark ran these transactions against the database:

- STOCK LEVEL: Checking the stock level
- DELIVERY: Processing a batch of 10 orders
- ORDER STATUS: Monitoring the status of orders
- PAYMENT: Processing a payment
- NEW ORDER: Entering a complete order

The benchmark measures transactions per minute (tpmC), which indicates new order transactions run per minute and provides a measure of business throughput. The benchmark also measures response time, which is the average time that a user got a response for each transaction.

Storwize V7000 Gen 2 and SVC 2145-DH8 using Oracle

Table 6-2 shows the results of the Oracle benchmarks.

Table 6-2 Storwize V7000 Gen 2 and SVC DH8 using Oracle

Transaction	Storwize V7000 Gen 2 SVC V7.4 (64 GB RAM) with 2 compression accelerator cards	SAN Volume Controller DH8 SVC V7.4 (64 GB RAM) with 2 compression accelerator cards
Stock Level	8.443	5.754
Delivery	2.699	0.719
Order Status	0.268	0.107
Payment	0.409	0.179
New Order	0.97	0.294
Average Response Time in Seconds	1.06877	0.47247
tpmC (Throughput)	30,793	37,556
Average IOPS Achieved	51,671	74,813
Number of Users	19,000	27,000

Note: *Average Response Time in Seconds* is not the statistical/mathematical average of the response time, but rather an average time that the benchmark delivers based on the weight of transactions and their rate per minute. It is also worth mentioning that the metric "Seconds*"* is an application benchmark response time that is composed of hundreds of I/Os.

6.3.3 Synthetic workloads

To demonstrate the variability between workloads that demonstrate zero and 100% temporal locality, you must modify traditional block benchmark tools to enable repeatable random workloads. This process results in best case and worst case raw block performance workloads when using compressed volumes. These tests were run with a known compressibility of data blocks by using a 1 MB pattern file that has a 65% and 80% compression ratio. These ratios are average for typical compression rates as seen by the IBM Real-time Compression Appliance product family.

Table 6-3 shows the performance results that can be used as a generic framework for realistic upper-limit expected performance for several models of SVC and Storwize V7000 using compression. The table depicts random I/O performance that is achieved with Version 7.3 as compared to Version 7.4 of the SVC code. The workload is ideal DB type with 65% compression, random I/O, 8 K I/Os, and 80% read.

Table 6-3 Storwize V7000 Comparisons of SVC V7.3 and V7.4

	V7000 Gen1		**V7000 Gen2 64 GB RAM**		**SVC DH8 64 GB RAM**	
SVC Version	**Version 7.3**		**Version 7.3**		**Version 7.3**	
Operation	**Worst Case**	**Best Case**	**Worst Case**	**Best Case**	**Worst Case**	**Best Case**
Read Miss IOPS	2 K	44 K	46 K	149 K	54 K	242 K
Write Miss IOPS	2 K	17 K	31 K	78 K	49 K	115 K
70/30 Miss IOPS	2 K	33 K	36 K	115 K	60 K	195 K
SVC Version	**Version 7.4**		**Version 7.4**		**Version 7.4**	
Operation	**Worst Case**	**Best Case**	**Worst Case**	**Best Case**	**Worst Case**	**Best Case**
Read Miss IOPS	2 K	44 K	52 K	270 K	106 K	357 K
Write Miss IOPS	2 K	17 K	44 K	115 K	52 K	202 K
70/30 Miss IOPS	2 K	33 K	58 K	211 K	86 K	281 K

From these purely synthetic workload tests, there is a wide variability in the compressed performance. This workload is best case because everything is repeatable. In repeatable workloads, the order in which blocks were "pseudo-randomly" written is repeated when reading back in the same "pseudo-random" order.

The worst case tests are where there is no correlation between write and read I/O patterns. This is how most benchmarks work and this is why they should be avoided with IBM Real-time Compression. These tests result in much slower performance compared with a fully allocated volume. This performance loss is because the system must read and extract a large chunk of data for every host (volume) 4 KB I/O. In these cases, compression is not preferred.

6.4 Standard benchmark tools

Traditional block- and file-based benchmark tools, such as IOmeter, IOzone, dbench, and fio that generate truly random I/O patterns do not work well with IBM Real-time Compression.

These tools generate synthetic workloads that do not have any temporal locality. Data is not read back in the same (or similar) order in which it was written. It is therefore not useful to estimate what your performance will look like for an application with these tools.

Consider what data a benchmark application uses. If the data is already compressed, or is all binary zero data, the differences that are measured are artificially bad, or good, based on the compressibility of the data. The more compressible the data, the better the performance is.

Attention: Random I/O benchmark tools do not generate real application patterns, and so cannot simulate the real performance of IBM Real-time Compression.

6.5 Sequential read and write I/O

IBM Real-time Compression is optimized for application workloads that are more random in nature, and have a mixture of read and write I/O. Writing sequentially to a few target compressed volumes or to a narrow area in a single compressed volume provides lower throughput.

Similarly, sequential read streaming is governed by the decompression performance per core. This process can reduce the read MBps throughput rates compared with fully allocated volumes when large numbers of physical disks are used in a storage pool. Perform testing to ensure that backup processing can be completed within required time windows.

Review the resulting throughput when using compressed volumes for workloads that are pure file copy type of workloads, such as backup-to-disk, and backup to tape.

6.6 Compression hardware assignments

An I/O Group that is servicing at least one compressed volume dedicates certain processor and memory resources for exclusive use by the compression engine. The nodes in such an I/O Group use these resources to compress and extract data, and to maintain the compression directory information. This resource assignment is made only when an I/O Group contains compressed volumes. These resources are returned to normal when the last compressed volume in an I/O Group is deleted.

The resource assignments are made when you create the first compressed volume in an I/O Group. Take care as you deploy compressed volumes, in particular on existing I/O Groups that are serving I/O for existing non-compressed volumes. Table 6-4 details the base resources in each hardware type.

Tip: For the 4-core SAN Volume Controller 2145-CG8, treat these nodes according to the guidelines for the SAN Volume Controller 2145-CF8. To check what type of controller that you have, run `svcinfo lsnodevpd`. If the output returns Xeon 5630, it is a 4-core SAN Volume Controller 2145-CG8.

Table 6-4 Default node resources

Per node	**SAN Volume Controller CF8/CG8***	**SAN Volume Controller CG8**	**SAN Volume Controller CG8 RPQ**	**SAN Volume Controller DH8**	**Storwize V7000 Gen1**	**Storwize V7000 Gen2**
Processor Hardware	4 cores	6 cores	12 cores	16 cores	4 cores	8 cores
Cache Memory	24 GB	24 GB	32 GB	64 GB	8 GB	32 - 64 GB

Explanation: These tests were performed using the SAN Volume Controller 2145-CG8 with six cores. For test results that pertain to the SAN Volume Controller CG8 with four cores, contact your IBM representative.

When you create a compressed volume, the core assignments and resource changes in Table 6-5 are made.

Table 6-5 Compressed node resources

Per node	**SAN Volume Controller CF8/CG8***	**SAN Volume Controller CG8**	**SAN Volume Controller CG8 RPQ**	**SAN Volume Controller DH8**	**Storwize V7000 Gen1**	**Storwize V7000 Gen2**
Main I/O cores	1 core	2 cores	4 cores	8 cores	1 core	4 core
Compression cores	3 cores	4 cores	8 cores	8 cores	3 cores	4 cores
Cache memory	22 GB	22 GB	32 GB	64 GB	8 GB	32 - 64 GB
Compression memory	2 GB	2 GB	2 GB	2 GB+	2 GB	2 GB+
Compression Accelerators	0	0	0	1 - 2	0	1 - 2

Note: There are two models of the SAN Volume Controller 2145-CG8 node. One has four cores and the second has six cores. The 6-core CG8 can be upgraded to a 12-core through RPQ 8S1296.

However, there are fewer cores available to the main I/O code, that is, normal host to disk I/O processing cores. Do *not* create compressed volumes if the processor utilization (before creating compressed volumes) is consistently sustained above the levels that are shown in Table 6-6 on page 91.

Table 6-6 Maximum existing sustained processor utilization per node

Per node	SAN Volume Controller CF8/CG8*	SAN Volume Controller CG8	SAN Volume Controller CG8 RPQ	SAN Volume Controller DH8	Storwize V7000 Gen1	Storwize V7000 Gen2
Processor already close to or above:	25%	50%	No consideration	No consideration	25%	50%

If the nodes in your I/O Group are sustaining processor usage higher than the suggested maximum processor usage, add an I/O Group before creating compressed volumes. You can also add a Storwize V7000 control enclosure. You can also consider upgrading to the following hardware configurations, which are highly recommended with IBM Real-time Compression:

- Storwize V7000 Gen2 with 64 GB and 2 x Compression Accelerator Cards (per canister)
- SVC (2145-DH8) with one or two Compression Accelerator Cards (per node)

There is a maximum number of compressed volumes that are supported for each I/O Group, which at the time of writing is 200 for all models of the SVC and Storwize V7000 Gen1 except for the SAN Volume Controller 2145-DH8 and the Storwize V7000 Gen2 models, which support 512 compressed volumes per I/O Group. You can have 800 - 2048 compressed volumes for four I/O Groups, depending on the model of SVC and Storwize V7000.

If you have only a few compressed volumes, configure them on one I/O Group. Do not split them between different I/O Groups when the number of compressed volumes is small.

For larger numbers of compressed volumes, the general preference, in systems with more than one I/O Group, is to distribute compressed volumes across I/O Groups. For example, a clustered pair of Storwize V7000 Gen2 control enclosures requires 256 compressed volumes. It is better to configure 128 volumes per I/O Group, instead of 256 compressed volumes in one I/O Group. Also, ensure that the preferred nodes for each compressed volumes are evenly distributed across both nodes in the I/O Group. This is the default behavior when you create consecutive volumes.

Starting with Version 7.4, multiple instances of IBM Random Access Compression Engine (RACE) code are being introduced, which does not require any user configuration. The use of multiple instances of RACE increases performance, especially when there are two compression accelerator cards that are configured. The following hardware configurations are supported by multiple RACE instances, which are highly recommended with IBM Real-time Compression for best performance:

- Storwize V7000 Gen2 with 64 GB and 2 x Compression Accelerator Cards (per canister)
- SAN Volume Controller (2145-DH8) with one or two Compression Accelerator Cards (per node)

6.7 Monitoring processor usage for compression

When the initial compressed volume is created, certain processor and memory resources are allocated for the use of compression. For more information, see 6.2.3, “Benchmarking compression performance” on page 86. Processor resources are therefore split between system services (main I/O code), and IBM Real-time Compression and extraction of data in compressed volumes.

The Storwize V7000 and SVC hardware can handle I/O even at extreme workloads with enough processor “headroom”. In most practical scenarios, enabling compression benefits the overall performance of the system. However, when processor-intensive services are used concurrently, review the processor utilization before enabling compression. For more information, see 6.2.3, “Benchmarking compression performance” on page 86.

System services that require more processor resources include Metro and Global Mirror, FlashCopy, Easy Tier, Stretched-cluster, and thin-provisioning.

To monitor processor performance by using the GUI, click **Monitoring** → **Performance**. The processor graph is displayed, including both the System and Compression average processor utilization.

To add the compression line graph into the processor utilization graph, select **Compression%**, as shown in Figure 6-1. The system statistics can be displayed by MBps or by IOPs.

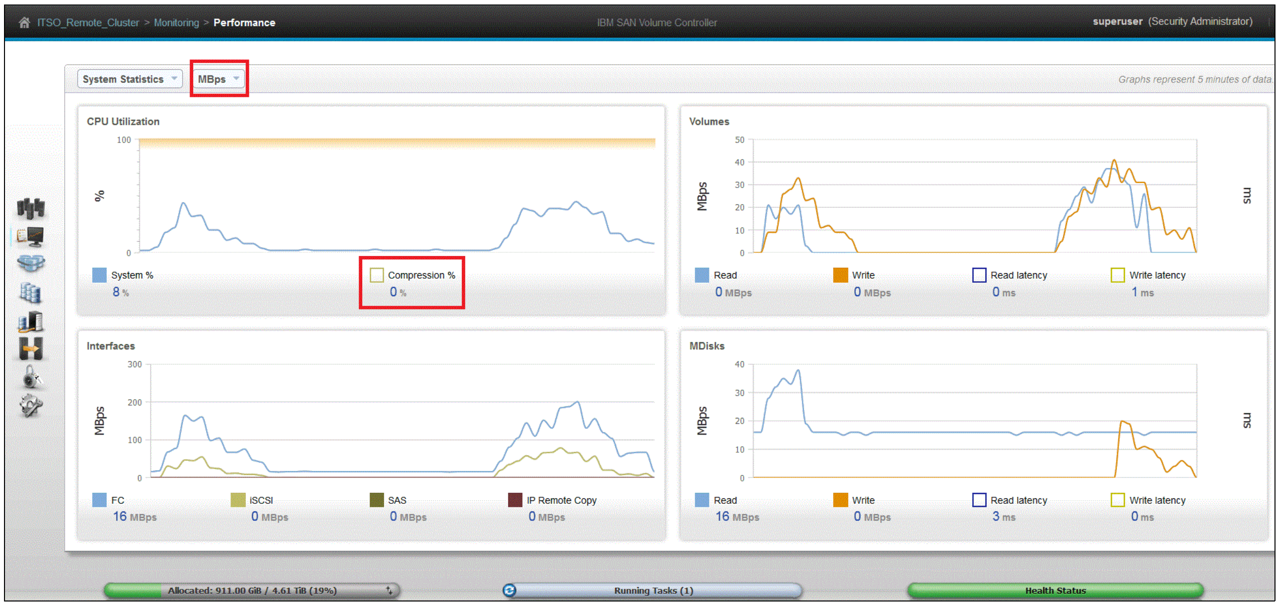


Figure 6-1 Selecting Compression%

Figure 6-2 shows the compression and the system line graphs in the processor utilization graph.

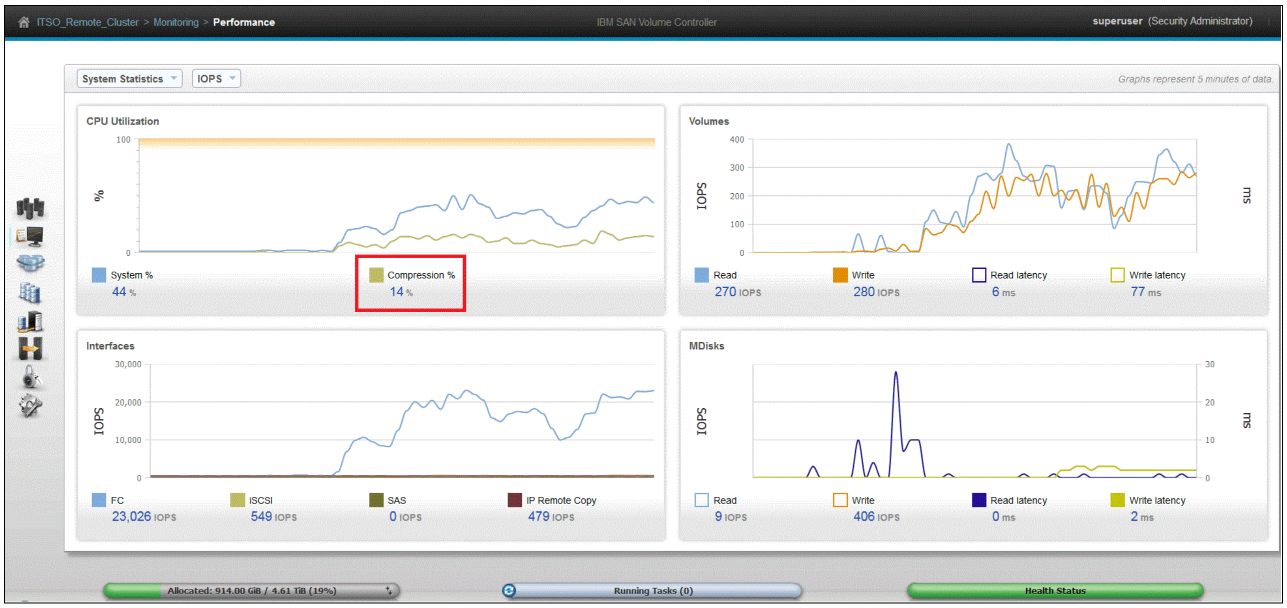


Figure 6-2 Compression and system line graphs

Example 6-1 shows how to display the processor statistics by using the CLI.

Example 6-1 Viewing processor statistics by using the CLI

```
IBM_2145:ITSO_Remote_Cluster:superuser>lsnodestats
node_id node_name stat_name         stat_current stat_peak stat_peak_time
1       node1     compression_cpu_pc 14          20        140916154358
1       node1     cpu_pc             31          46        140916154413
...
4       node2     compression_cpu_pc 8           9         140916154410
4       node2     cpu_pc             30          53        140916154415
...
IBM_2145:ITSO_Remote_Cluster:superuser>
```

6.8 File system alignment

As with other volume types, compressed volumes can be affected by suboptimal logical alignment of the host's file system as well. When a host sends read requests, the Storwize V7000 cache layer fetches data from the volume in 4 KB blocks. The fetched blocks are then cached in memory for reuse (that is, for host rereads). A host file system might write its data out of the 4 KB boundaries. In this case, any read results in the Storwize V7000 cache layer must read two adjacent blocks from the volume. Because more resources are required to fulfill the read, latency increases and an additional cache buffer is required. For more information about misaligned I/O, see *Advanced Format HDD Technology Overview*, REDP-5119.

6.9 Performance guidelines summary

In summary, you must take the following factors into account for a successful setup:

- Your data is a good candidate for compression: Validate that the application and data are compressible.
- The application is a good candidate for compression: Determine that the type of application and workload characteristics match those outlined in this chapter:
  - If possible, clone the application and workload on a suitable I/O Group and validate that performance results from using compression match your expectations.
  - Monitor existing cache hit rates on existing volumes. High cache hit rates indicate that the volume is a good candidate for compression because of locality and rereferencing data.
- The system has resources to handle compression: Verify the performance of any existing running volumes, validating that the processor node usage is under the maximums that are specified in Table 6-6 on page 91. If the system is constantly running with higher processor utilization, enabling compression can affect the other system services.

  Tip: You can add nodes and load balance volumes to increase the overall processor and memory resources of the SVC/V7000 cluster.

- Additional considerations are reviewed: Ensure that file system alignments are as specified in 6.8, “File system alignment” on page 93. Also, think about backup requirements for compressed volumes that take into account the advice in 6.5, “Sequential read and write I/O” on page 89.

7

Compression savings reporting

This chapter describes how compression savings are reported in the user interface. In addition, the chapter addresses how to interpret disk space usage while considering the different layers of storage virtualization that are involved. These layers are the host file system, real-time compression, and thin provisioning.

This chapter includes the following sections:

- Understanding capacity
- Reporting basics overview
- Reporting of compressed volumes
- Reporting of mirrored compressed volumes
- Reporting of sparse compressed volumes
- Reporting of freed data

7.1 Understanding capacity

Understanding capacity in a virtualized storage system can be a challenge. A good approach is to review the various layers that are involved in storing data, from the host to the disks:

- Data: The data itself as stored by the host OS onto the volume.
- Host file system or raw device: A file system or the raw device that stores the data.
- SCSI disk (also called LUN): The SCSI device to which the host OS writes data.
- Compressed volume: The SCSI device representation in the Storwize V7000 system.
- Storage pool: The logical storage container that holds compressed volumes.

Remember: The same storage pool can contain fully allocated, thin-provisioned, and compressed volumes concurrently. Understanding capacity in the storage pool layer in this case requires a closer review of the volumes and their types.

By design, the compression technology that is implemented in the Storwize V7000 and the SAN Volume Controller (SVC) is not apparent to the host. It compresses the client data before writing to the disk, and extracts the data as it is read by the host.

The data size looks different from a different point of view. For example, the compressed size is not reflected to the host, and can be seen only from the Storwize V7000 and SVC points of view.

7.2 Reporting basics overview

Here are the basic concepts that are relevant to the reporting of compressed data in SVC and Storwize V7000:

Virtual capacity	Virtual capacity is the volume storage capacity that is available to a host. Real capacity is the storage capacity that is allocated to a volume copy from a storage pool. In a fully allocated volume, the virtual capacity and real capacity are the same. In thin-provisioned or compressed volumes, however, the virtual capacity can be much larger than the real capacity.
Real size	Real size is the amount of space from the storage pool that is allocated to allow the volume data to be stored. A compressed volume is by default a thin-provisioned volume that allows you to allocate space on demand. When a new volume is created, there is an option to define the actual size it creates as a percentage of the original volume size. By default, it is 2%. The volume expands automatically according to the usage.
Used size	The amount of real size that is used to store data for the volume, which is sometimes called *compressed size*.
Size before compression	The size of all the data that is written to the volume calculated as though it was written without compression. This size is reflected on the host operating system.

Tip: This concept is called `uncompressed_used_size` in the CLI.

Compression ratio The ratio between the compressed size and the uncompressed size.

Consideration: Both size before compression and the compression ratio values do not calculate savings that are achieved from detecting long sequences of empty space. For more information, see 7.5, "Reporting of sparse compressed volumes" on page 113.

7.2.1 Reporting of compression in the GUI

Reporting of compressed data is available in the GUI in the following windows:

System level	Click. **Monitoring** → **System**.
Storage pool level	Click **Pools** → **Volumes by Pool** **Pools** → **MDisks by Pools**.
Volume level	Click **Volumes** → **Volumes by Pool**.

System allocation window

To view the System allocation window, click **Monitoring** → **System** by using the icon menu. The window that is shown in Figure 7-1 opens.

The System allocation window shows the allocated and physical storage. If compression is not being used, then it simply displays Allocated, which indicates the amount of capacity that is allocated to all volumes in the system. If compression is enabled, then it shows Compress Allocated, which indicates the amount of capacity that is allocated to all volume types.

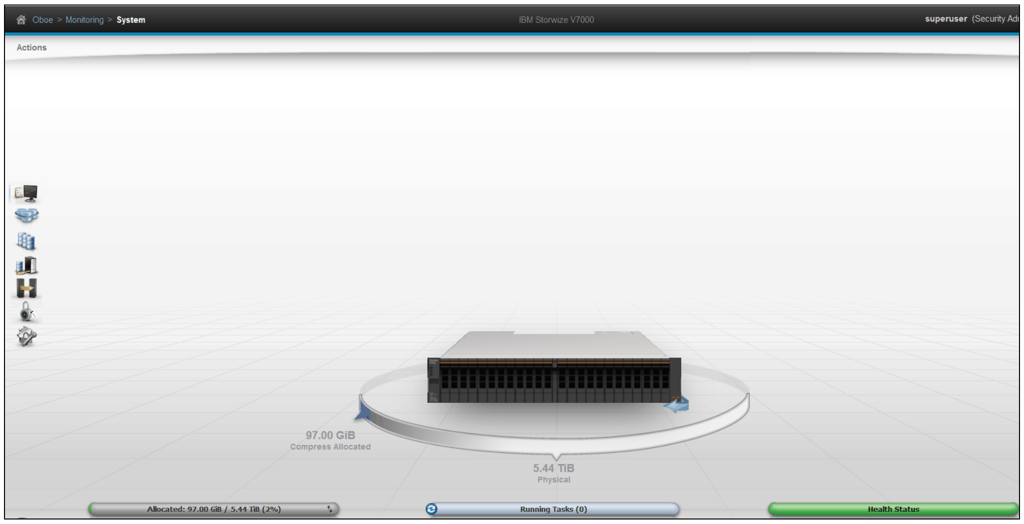


Figure 7-1 The System allocation window

Mouse over the cylinder to view efficiency-related metrics about the system, as shown in Figure 7-2.

This window shows you metrics that are related to Compression Savings, Thin-provisioning Savings, and Total Virtual Savings.

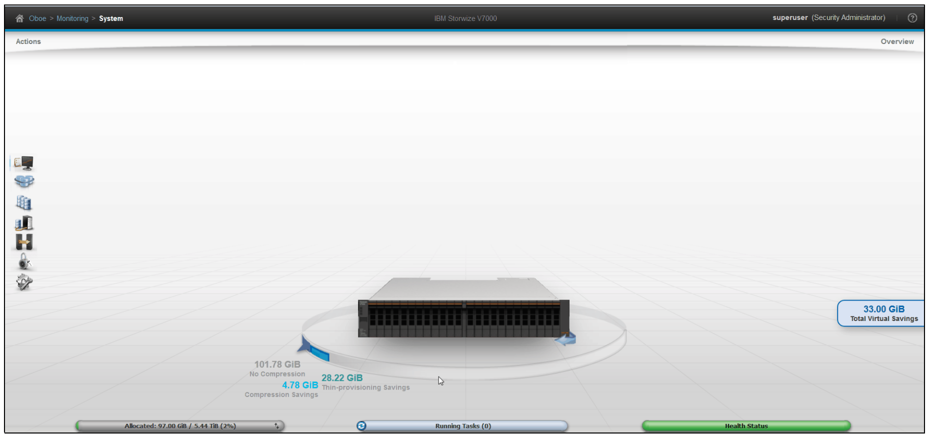


Figure 7-2 The System efficiency window

In addition, there is a bar at the lower-left side of the window that provides the same type of system storage metrics that are available through the cylinder view window, such as Allocated, Virtual, and Compression.

Storage pool level

To get information at the storage pool level, click **MDisks by Pools** in the **Pools** icon menu, as shown in Figure 7-3.

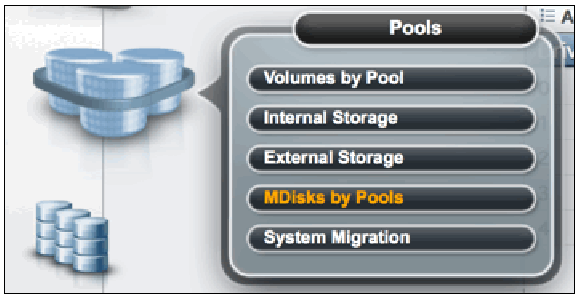
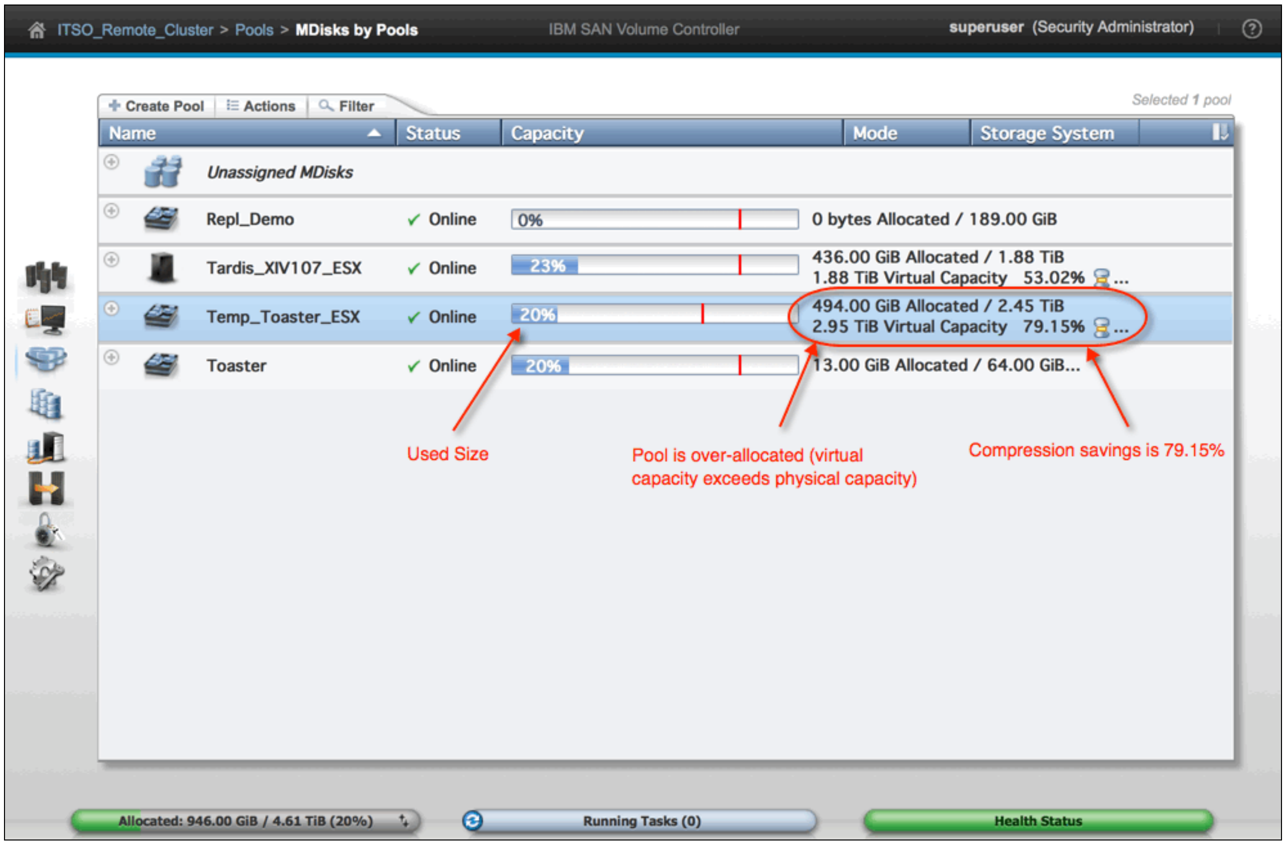


Figure 7-3 Accessing the Pools menu

The MDisks by Pools window provides the compression ratio benefits of the entire pool. Figure 7-4 on page 99 shows an example where a pool is overallocated.


Create Pool
Actions
Filter
Selected 1 pool
Used Size
Pool is over-allocated (virtual capacity exceeds physical capacity)
Compression savings is 79.15%
Running Tasks (0)
Health Status

ITSO_Remote_Cluster > Pools > **MDisks by Pools** IBM SAN Volume Controller **superuser** (Security Administrator)

Name	Status	Capacity	Mode	Storage System
Unassigned MDisks				
Repl_Demo	✓ Online	0% 0 bytes Allocated / 189.00 GiB		
Tardis_XIV107_ESX	✓ Online	23% 436.00 GiB Allocated / 1.88 TiB 1.88 TiB Virtual Capacity 53.02% ...		
Temp_Toaster_ESX	✓ Online	20% 494.00 GiB Allocated / 2.45 TiB 2.95 TiB Virtual Capacity 79.15% ...		
Toaster	✓ Online	20% 13.00 GiB Allocated / 64.00 GiB...		

Allocated: 946.00 GiB / 4.61 TiB (20%)

Figure 7-4 Overallocation example

The output has the following significant information:

- Used size: The percentages in the shaded blue part are the amount of used space from the allocated capacity. In this example, it is 494 GiB/2.45 TiB*100 = 20%.
- Compression ratio: This is the rate between the compressed size and the uncompressed size of the data. These numbers are displayed in the Volumes by Pool window. In this example, it is 79.15%. In total, 1 TB is saved in this pool.
- Virtual capacity: This is the sum of all volume sizes in this pool. In this example, it is 2.95 TiB. The virtual capacity section exists only when the total volume capacity is greater than the total capacity of the pool or the system.

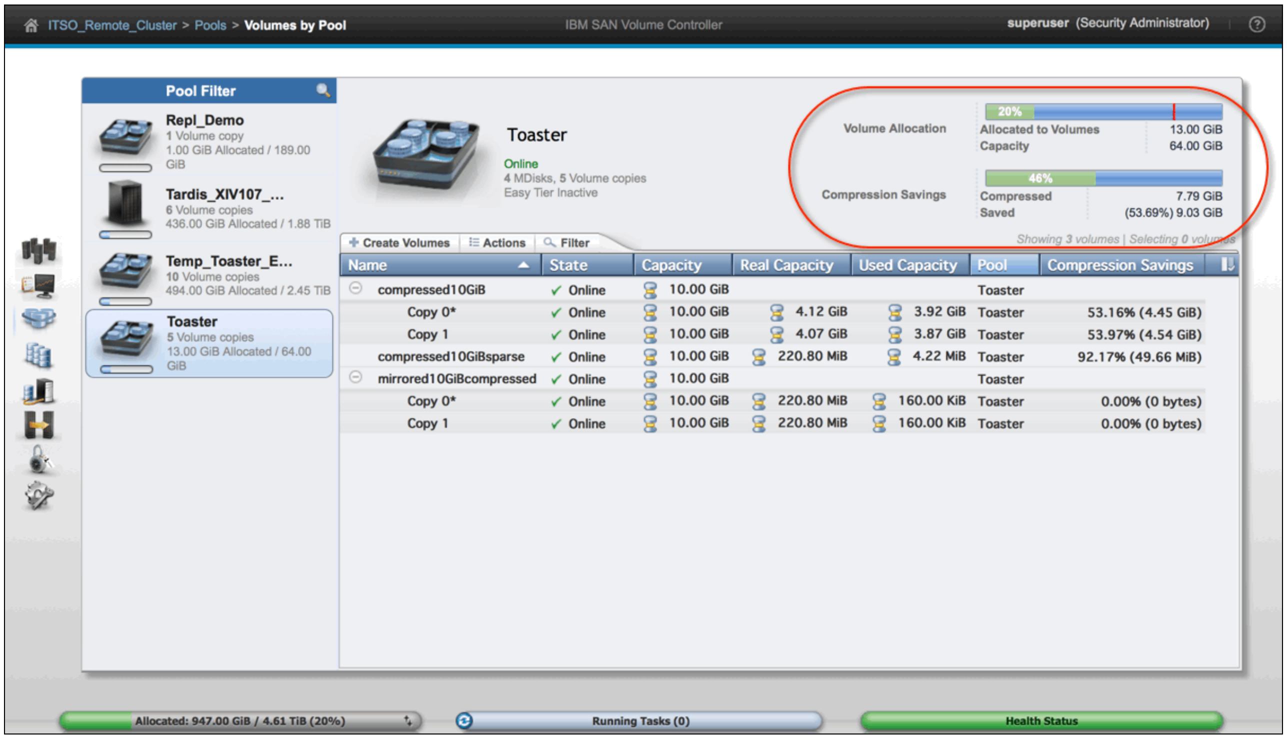
Volume level

To see a report at a volume level, click **Volumes by Pool** in the **Pools** icon menu that is shown in Figure 7-3 on page 98. As shown in Figure 7-5, the table provides the compression savings per volume in a specific pool. The display presents the space allocation and the compression savings of the entire pool. The right pane in the Volume allocation display represents the virtual capacity. This is the overallocated size of volumes in the pool.

Consideration: The reported numbers are dynamically updated as data is sent from the storage system cache to the IBM Random Access Compression Engine (RACE) component. This updating causes a slight delay (typically a few seconds) between the host writes and the updated reporting.

The Compression Savings field shows that the compressed size is 7.79 GiB. This is the amount of space the data is actually using. The saved size is 9.03 GiB, which means that in total the uncompressed size of the data is 7.79+9.03= 16.82 GiB.

The compression ratio is written below the bar and calculated from the compressed size and the uncompressed size of the data. In this example, it is (1- (7.79/16.82))*100 = 53.69%.


ITSO_Remote_Cluster > Pools > Volumes by Pool
IBM SAN Volume Controller
superuser (Security Administrator)
Pool Filter
Create Volumes
Actions
Filter
Running Tasks (0)
Health Status

Repl_Demo
1 Volume copy
1.00 GiB Allocated / 189.00 GiB

Tardis_XIV107_...
6 Volume copies
436.00 GiB Allocated / 1.88 TiB

Temp_Toaster_E...
10 Volume copies
494.00 GiB Allocated / 2.45 TiB

Toaster
5 Volume copies
13.00 GiB Allocated / 64.00 GiB

Toaster
Online
4 MDisks, 5 Volume copies
Easy Tier Inactive

Volume Allocation	Allocated to Volumes	13.00 GiB
	Capacity	64.00 GiB
Compression Savings	Compressed	7.79 GiB
	Saved	(53.69%) 9.03 GiB

Showing 3 volumes | Selecting 0 volumes

Name	State	Capacity	Real Capacity	Used Capacity	Pool	Compression Savings
compressed10GiB	✓ Online	10.00 GiB			Toaster	
Copy 0*	✓ Online	10.00 GiB	4.12 GiB	3.92 GiB	Toaster	53.16% (4.45 GiB)
Copy 1	✓ Online	10.00 GiB	4.07 GiB	3.87 GiB	Toaster	53.97% (4.54 GiB)
compressed10GiBsparse	✓ Online	10.00 GiB	220.80 MiB	4.22 MiB	Toaster	92.17% (49.66 MiB)
mirrored10GiBcompressed	✓ Online	10.00 GiB			Toaster	
Copy 0*	✓ Online	10.00 GiB	220.80 MiB	160.00 KiB	Toaster	0.00% (0 bytes)
Copy 1	✓ Online	10.00 GiB	220.80 MiB	160.00 KiB	Toaster	0.00% (0 bytes)

Allocated: 947.00 GiB / 4.61 TiB (20%)

Figure 7-5 Volume level

7.2.2 Compression reporting using the command-line interface

You can generate reports on compressed data by running the following command-line interface (CLI) commands:

System level	`lssystem`
Specific storage pool	`lsmdiskgrp [<mdisk_grp_name>]`
All storage pools	`lsmdiskgrp`
Specific volume	`lssevdiskcopy [<vdisk_name>]`

All volumes in a pool	**`lssevdiskcopy -filtervalue mdisk_grp_name=[<mdisk_grp_name>]`**
All volumes in a system	**`lssevdiskcopy`**

Example 7-1 shows the output of the **`lssystem`** command.

Example 7-1 lssystem output

```
IBM_2076:Cluster_prod:superuser>lssystem
id 0000020060C14FE2
name ITSO_Remote_Cluster
location local
partnership
total_mdisk_capacity 4.6TB
space_in_mdisk_grps 4.6TB
space_allocated_to_vdisks 922.61GB
total_free_space 3.7TB
total_vdiskcopy_capacity 4.83TB
total_used_capacity 828.01GB
total_overallocation 104
total_vdisk_capacity 2.96TB
total_allocated_extent_capacity 933.00GB
[...lines deleted for brevity...]
compression_active yes
compression_virtual_capacity 2.51TB
compression_compressed_capacity 263.79GB
compression_uncompressed_capacity 1.23TB
```

Example 7-2 shows the output of the **`lsmdiskgrp`** command for a specific volume.

Example 7-2 lsmdiskgrp output

```
IBM_2145:orchestra:superuser>lsmdiskgrp Temp_Toaster_ESX
id 3
name Temp_Toaster_ESX
status online
mdisk_count 1
vdisk_count 9
capacity 2.45TB
extent_size 1024
free_capacity 1.97TB
virtual_capacity 2.85TB
used_capacity 429.61GB
real_capacity 486.97GB
overallocation 116
warning 80
easy_tier auto
easy_tier_status balanced
tier ssd
tier_mdisk_count 0
tier_capacity 0.00MB
tier_free_capacity 0.00MB
tier enterprise
tier_mdisk_count 1
tier_capacity 2.45TB
tier_free_capacity 1.97TB
```

```
tier nearline
tier_mdisk_count 0
tier_capacity 0.00MB
tier_free_capacity 0.00MB
compression_active yes
compression_virtual_capacity 2.50TB
compression_compressed_capacity 260.08GB
compression_uncompressed_capacity 1.22TB
```

The **lsmdiskgrp** command lists all the information for each pool by using the headers that are listed in Figure 7-6. This output shows the compression-related values for each pool.

```
id
name
status
mdisk_count
vdisk_count
capacity
extent_size
free_capacity
virtual_capacity
used_capacity
real_capacity
overallocation
warning
easy_tier
easy_tier_status
[...lines deleted for brevity...]
compression_active
compression_virtual_capacity
compression_compressed_capacity
compression_uncompressed_capacity
```

Figure 7-6 Order of headers that are listed with lsmdiskgrp

For output without the headers, run **lsmdiskgrp -nohdr**, as shown in Example 7-3. (The lines in this example output wrap, but are normally all one line.)

Example 7-3 lsmdiskgrp -nohdr output

```
IBM_2145:orchestra:superuser>lsmdiskgrp -nohdr
1 Repl_Demo         online 4 0 189.00GB 1024 189.00GB 0.00MB   0.00MB   0.00MB   0   80 auto
balanced no  0.00MB  0.00MB   0.00MB 1 Repl_Demo         0 0.00MB parent
2 Toaster           online 4 1 64.00GB  1024 61.00GB  100.00GB 0.75MB   2.02GB   156 80 auto
balanced no  0.00MB  0.00MB   0.00MB 2 Toaster           0 0.00MB parent
3 Temp_Toaster_ESX  online 1 9 2.45TB   1024 1.97TB   2.85TB   429.61GB 486.97GB 116 80 auto
balanced yes 2.50TB  260.08GB 1.22TB 3 Temp_Toaster_ESX  0 0.00MB parent
6 Tardis_XIV107_ESX online 4 6 1.88TB   1024 1.45TB   1.88TB   398.40GB 433.63GB 100 80 auto
balanced yes 10.00GB 3.71GB   7.91GB 6 Tardis_XIV107_ESX 0 0.00MB parent
```

To list the information for a specific volume with the headers, run **`lssevdiskcopy [<vdisk_name>]`**. Figure 7-7 shows the headers for the compression-related values that are provided by this command for a specific volume.

```
vdisk_id
vdisk_name
copy_id
mdisk_grp_id
mdisk_grp_name
capacity
used_capacity
real_capacity
free_capacity
overallocation
autoexpand
warning
grainsize
se_copy
compressed_copy
uncompressed_used_capacity
```

Figure 7-7 Order of compression-related headers that are listed by the lssevdiskcopy command

For output without the headers, run **`lssevdiskcopy -nohdr`**, as shown in Example 7-4.

Example 7-4 lssevdiskcopy -nohdr output

```
IBM_2145:orchestra:superuser>lssevdiskcopy -nohdr compressed
0 compressed 0 3 Temp_Toaster_ESX 1.00TB 66.36GB 86.85GB 20.49GB 1178 on 80  no
yes 984.29GB 3 Temp_Toaster_ESX
```

To list all the volumes in a pool with the headers, run **`lssevdiskcopy -filtervalue mdisk_grp_name=<mdisk_grp_name>`**. Figure 7-8 shows the headers for the compression-related values that are provided by this command.

```
vdisk_id
vdisk_name
copy_id
mdisk_grp_id
mdisk_grp_name
capacity
used_capacity
real_capacity
free_capacity
overallocation
autoexpand
warning
grainsize
se_copy
compressed_copy
uncompressed_used_capacity
```

Figure 7-8 Order of compression-related headers that are listed by lssevdiskcopy for a given pool

For output without the headers, run **lssevdiskcopy -nohdr**, as shown in Example 7-5. (The lines in this example output wrap, but are normally all one line.)

Example 7-5 lssevdiskcopy -nohdr output

```
lssevdiskcopy -nohdr -filtervalue mdisk_grp_name=Temp_Toaster_ESX
0  compressed            0 3 Temp_Toaster_ESX 1.00TB   66.36GB  86.85GB  20.49GB  1178 on 80
no  yes 984.29GB 3 Temp_Toaster_ESX
1  cmp_mirror            0 3 Temp_Toaster_ESX 10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
2  volcomp2              0 3 Temp_Toaster_ESX 10.00GB  3.71GB   3.92GB   211.02MB 255  on 80
no  yes 7.91GB   3 Temp_Toaster_ESX
4  volcomp3              0 3 Temp_Toaster_ESX 10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
4  volcomp3              1 3 Temp_Toaster_ESX 10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
9  XIV107_ESX            1 3 Temp_Toaster_ESX 1.46TB   190.00GB 220.01GB 30.00GB  681  on 80
no  yes 255.13GB 3 Temp_Toaster_ESX
11 ESX109_ESX_REPL_MASTER 1 3 Temp_Toaster_ESX 300.00GB 115.54GB 121.54GB 6.00GB   246  on 80
256 yes no  115.54GB 3 Temp_Toaster_ESX
```

To list all volumes in a system with the headers, run **lssevdiskcopy** command. Figure 7-9 displays the headers for the compression-related values for all volumes in a system.

```
vdisk_id
vdisk_name
copy_id
mdisk_grp_id
mdisk_grp_name
capacity
used_capacity
real_capacity
free_capacity
overallocation
autoexpand
warning
grainsize
se_copy
compressed_copy
uncompressed_used_capacity
```

Figure 7-9 Compression-related headers for all volumes in a system through the lssevdiskcopy command

For output without the headers, run **lssevdiskcopy -nohdr**, as shown in Example 7-6. (The lines in this example output wrap, but are normally all one line.)

Example 7-6 lssevdiskcopy -nohdr output

```
IBM_2145:orchestra:superuser>lssevdiskcopy -nohdr
0  compressed           0 3 Temp_Toaster_ESX  1.00TB   66.36GB  86.85GB  20.49GB  1178 on 80
no  yes 984.29GB 3 Temp_Toaster_ESX
1  cmp_mirror           0 3 Temp_Toaster_ESX  10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
2  volcomp2             0 3 Temp_Toaster_ESX  10.00GB  3.71GB   3.92GB   211.02MB 255  on 80
no  yes 7.91GB   3 Temp_Toaster_ESX
```

```
2  volcomp2             1 6 Tardis_XIV107_ESX 10.00GB  3.71GB   3.92GB   209.71MB 255  on 80
no  yes 7.91GB   6 Tardis_XIV107_ESX
4  volcomp3             0 3 Temp_Toaster_ESX  10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
4  volcomp3             1 3 Temp_Toaster_ESX  10.00GB  0.16MB   220.80MB 220.64MB 4637 on 80
no  yes 0.00MB   3 Temp_Toaster_ESX
9  XIV107_ESX            0 6 Tardis_XIV107_ESX 1.46TB   259.10GB 289.10GB 30.00GB  518  on 80
256 yes no  259.10GB 6 Tardis_XIV107_ESX
9  XIV107_ESX           1 3 Temp_Toaster_ESX  1.46TB   190.00GB 220.01GB 30.00GB  681  on 80
no  yes 255.13GB 3 Temp_Toaster_ESX
10 temp                  0 6 Tardis_XIV107_ESX 100.00GB 0.75MB   2.02GB   2.02GB   4960 on 80
256 yes no  0.75MB   6 Tardis_XIV107_ESX
10 temp                  1 2 Toaster           100.00GB 0.75MB   2.02GB   2.02GB   4960 on 80
256 yes no  0.75MB   2 Toaster
11 ESX109_ESX_REPL_MASTER 0 6 Tardis_XIV107_ESX 300.00GB 115.59GB 118.60GB 3.01GB   252  on 80
256 yes no  115.59GB 6 Tardis_XIV107_ESX
11 ESX109_ESX_REPL_MASTER 1 3 Temp_Toaster_ESX  300.00GB 115.54GB 121.54GB 6.00GB   246  on 80
256 yes no  115.54GB 3 Temp_Toaster_ESX
```

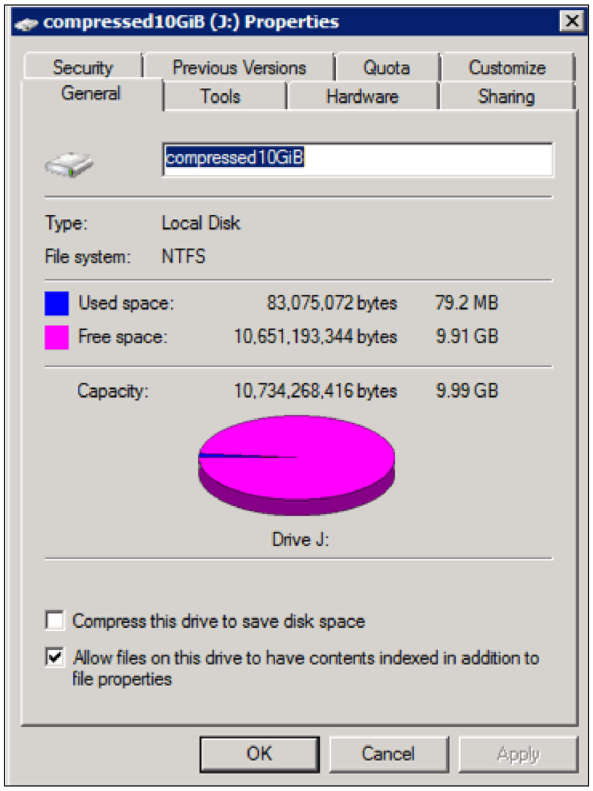
7.3 Reporting of compressed volumes

This section addresses the capacity usage as it is reflected to the host, and also from the point of view of the storage system.

Note: This example uses an SVC running software Version 7.4. The example scenario and related reporting details are valid for Storwize V7000 Gen2 and FlashSystem V840 as well.

In this example, we use a storage pool that consists of four 16 GiB MDisks, presented from an IBM XIV Gen2 to SVC, for a net pool capacity of 64 GiB. A 10 GB compressed volume is created, attached to a Windows 2008 Server, and formatted by the host.

Figure 7-10 shows the properties of the volume after formatting from the host perspective.


compressed10GiB (J:) Properties

compressed10GiB

Type:	Local Disk		
File system:	NTFS		
🟦 Used space:	83,075,072 bytes	79.2 MB	
🟪 Free space:	10,651,193,344 bytes	9.91 GB	
Capacity:	10,734,268,416 bytes	9.99 GB	

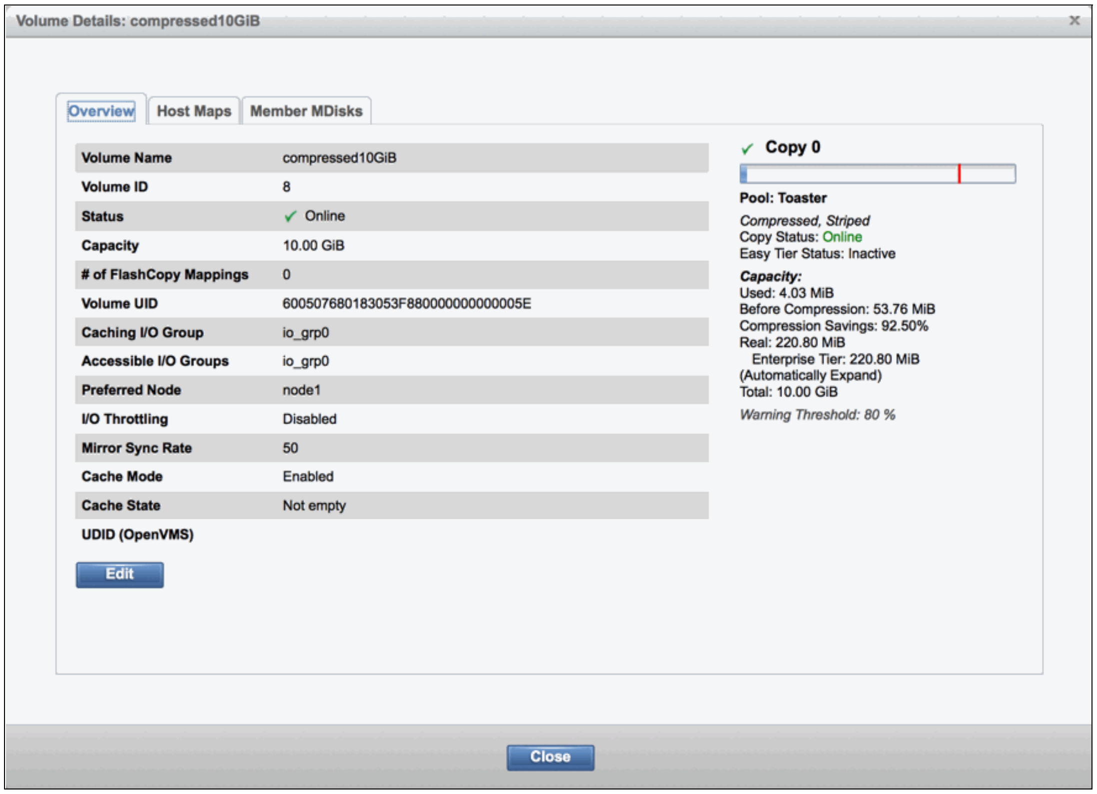
Drive J:

☐ Compress this drive to save disk space

☑ Allow files on this drive to have contents indexed in addition to file properties

Figure 7-10 Properties of the formatted and empty compressed volume from the host perspective.

Figure 7-11 on page 107 shows the properties of the formatted and empty volume from the storage system perspective.



Volume Name	compressed10GiB
Volume ID	8
Status	✓ Online
Capacity	10.00 GiB
# of FlashCopy Mappings	0
Volume UID	600507680183053F880000000000005E
Caching I/O Group	io_grp0
Accessible I/O Groups	io_grp0
Preferred Node	node1
I/O Throttling	Disabled
Mirror Sync Rate	50
Cache Mode	Enabled
Cache State	Not empty
UDID (OpenVMS)	

✓ Copy 0

Pool: Toaster

Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive

Capacity:
Used: 4.03 MiB
Before Compression: 53.76 MiB
Compression Savings: 92.50%
Real: 220.80 MiB
Enterprise Tier: 220.80 MiB
(Automatically Expand)
Total: 10.00 GiB

Warning Threshold: 80 %

Figure 7-11 Properties of the formatted, empty compressed volume from the storage system perspective

Table 7-1 shows a summary of the initial reported capacity information from the host, and the SVC point of view.

Remember: When a compressed volume is created, the compression engine immediately creates the volume's minimal metadata and header. Therefore, the used size is larger than the before compression size.

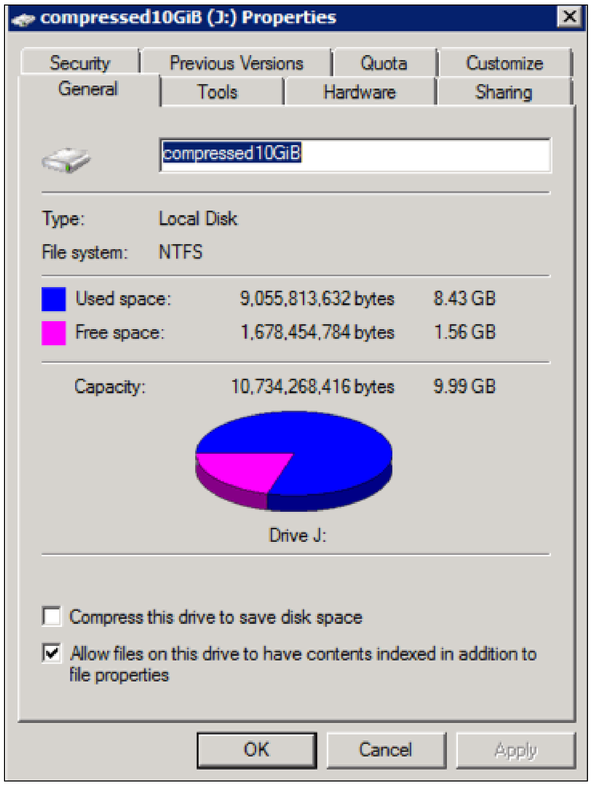
Table 7-1 Capacity report

Entity (host or storage system)	Reported capacity	Explanation
Windows 2008 used space	79.2 MB is used in the drive, out of a total capacity of 9.99 GB.	The host operating system is not aware of the actual capacity that is used by the storage system.
Storage system volume details	Real size: 220.8 MiB	220.8 MiB is the actual allocated size. This size is 2% of 100 GB, which is the volume size that is created.
Storage system volume details	Before Compression: 53.76 MiB	This is the uncompressed size of the data. This is the size that is reflected to the host. In some cases, the size is different from the host point of view and the storage system.

Entity (host or storage system)	Reported capacity	Explanation
Storage system volume details	Used size: 4.03 MiB	The 53.76 MiB of data was compressed to 4.03 MiB. This is the actual disk usage for storing this data.
Storage system volume details	Compression saving: 92.50%	According to the 'used size' and 'before compression' size, you can see that you saved 92.50% of disk space.

In this example, we now copy data to the compressed volume on the host. In this example, it is 8.43 GB of uncompressed data.

Figure 7-12 shows the data size after it is copied, as it is reported by the host. The host is not aware of the compression and presents the 'used size' as the uncompressed size of the data, in this case, 8.43 GB.



compressed10GiB

Type: Local Disk

File system: NTFS

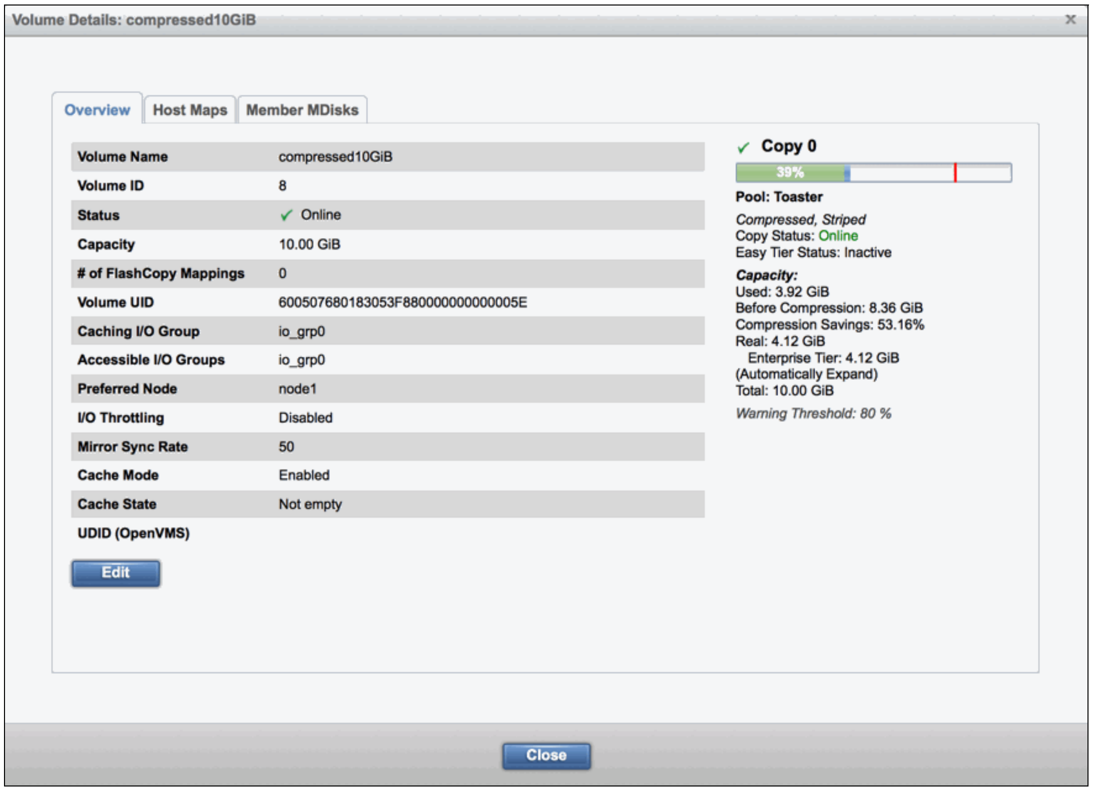
🟦	Used space:	9,055,813,632 bytes	8.43 GB
🟪	Free space:	1,678,454,784 bytes	1.56 GB
	Capacity:	10,734,268,416 bytes	9.99 GB

☐ Compress this drive to save disk space

☑ Allow files on this drive to have contents indexed in addition to file properties

Figure 7-12 Data size as reported by the host

To see the actual used size, you must look at volume details on the storage system. Figure 7-13 on page 110 shows that the data size before compression is 10.93 GB, and the used size is 4.53 GB. The compression savings are calculated from both parameters, and in this case the savings are 58.54%.



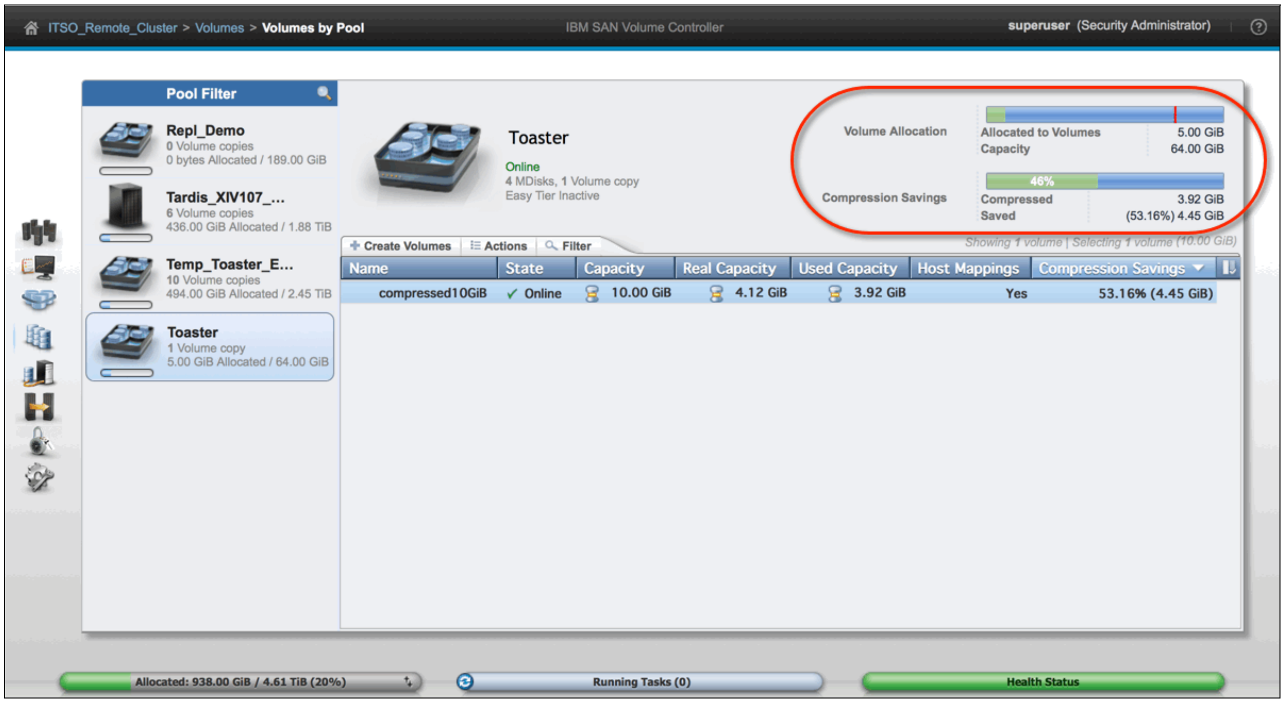
Volume Name	compressed10GiB
Volume ID	8
Status	✓ Online
Capacity	10.00 GiB
# of FlashCopy Mappings	0
Volume UID	600507680183053F880000000000005E
Caching I/O Group	io_grp0
Accessible I/O Groups	io_grp0
Preferred Node	node1
I/O Throttling	Disabled
Mirror Sync Rate	50
Cache Mode	Enabled
Cache State	Not empty
UDID (OpenVMS)	

✓ Copy 0
39%
Pool: Toaster
Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive
Capacity:
Used: 3.92 GiB
Before Compression: 8.36 GiB
Compression Savings: 53.16%
Real: 4.12 GiB
Enterprise Tier: 4.12 GiB
(Automatically Expand)
Total: 10.00 GiB
Warning Threshold: 80 %

Figure 7-13 Compressed volume details after copying data (as viewed from the storage system)

Remember: The used size can be smaller than the “before” compression size or the real size from the storage system point of view. This discrepancy can occur because thin-provisioned volumes do not allocate new storage for buffers that contain zeros.

Figure 7-14 shows the capacity details per volume. In the upper bar, you can see the volume allocation size as part of the entire pool. The lower bar shows the compressed size, uncompressed size, and the savings percentages. In this example, you can see that the allocated volumes size is 5.00 GiB, the compressed size is 3.92 GiB, and the saved space is 4.45 GiB. The total compression savings in this pool are 53.16%. In this example, because there is only one volume in the pool, the compression savings for the volume and the pool are the same.


ITSO_Remote_Cluster > Volumes > Volumes by Pool
IBM SAN Volume Controller
superuser (Security Administrator)
Pool Filter
Repl_Demo
0 Volume copies
0 bytes Allocated / 189.00 GiB
Tardis_XIV107_...
6 Volume copies
436.00 GiB Allocated / 1.88 TiB
Temp_Toaster_E...
10 Volume copies
494.00 GiB Allocated / 2.45 TiB
Toaster
1 Volume copy
5.00 GiB Allocated / 64.00 GiB
Toaster
Online
4 MDisks, 1 Volume copy
Easy Tier Inactive
Volume Allocation
Allocated to Volumes 5.00 GiB
Capacity 64.00 GiB
Compression Savings
46%
Compressed 3.92 GiB
Saved (53.16%) 4.45 GiB
Create Volumes
Actions
Filter
Showing 1 volume | Selecting 1 volume (10.00 GiB)
Allocated: 938.00 GiB / 4.61 TiB (20%)
Running Tasks (0)
Health Status

Name	State	Capacity	Real Capacity	Used Capacity	Host Mappings	Compression Savings
compressed10GiB	✓ Online	10.00 GiB	4.12 GiB	3.92 GiB	Yes	53.16% (4.45 GiB)

Figure 7-14 Volumes by pool window showing compression savings

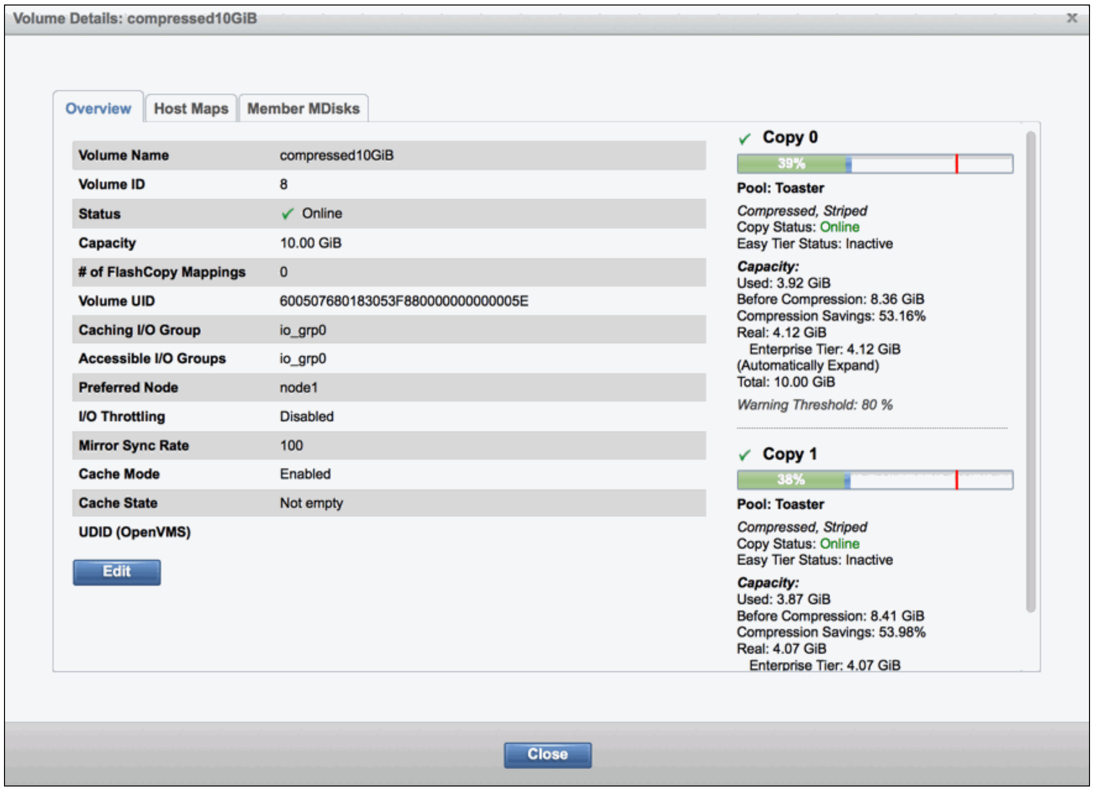
7.4 Reporting of mirrored compressed volumes

In some cases, the reporting size of different volumes that contain the same data can be different. These cases and the reasons for this behavior are addressed in this section.

7.4.1 Volume copy of an existing compressed volume

When adding a mirrored volume copy from an existing volume with data, both copies might report a different real size. On a volume copy, the data is extracted by the compression engine and compressed again when it is written to the new volume. The real size can be changed because of the sequential writes the copy does. When using sequential writes, the data is written in an optimized way to the disk and can take up a smaller size than the original copy.

Figure 7-15 on page 112 shows the copies and the different sizes that are reported, although they both contain the same data.


Volume Details: compressed10GiB

Volume Name	compressed10GiB
Volume ID	8
Status	✓ Online
Capacity	10.00 GiB
# of FlashCopy Mappings	0
Volume UID	600507680183053F880000000000005E
Caching I/O Group	io_grp0
Accessible I/O Groups	io_grp0
Preferred Node	node1
I/O Throttling	Disabled
Mirror Sync Rate	100
Cache Mode	Enabled
Cache State	Not empty
UDID (OpenVMS)	

✓ Copy 0
39%
Pool: Toaster
Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive
Capacity:
Used: 3.92 GiB
Before Compression: 8.36 GiB
Compression Savings: 53.16%
Real: 4.12 GiB
Enterprise Tier: 4.12 GiB
(Automatically Expand)
Total: 10.00 GiB
Warning Threshold: 80 %

✓ Copy 1
38%
Pool: Toaster
Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive
Capacity:
Used: 3.87 GiB
Before Compression: 8.41 GiB
Compression Savings: 53.98%
Real: 4.07 GiB
Enterprise Tier: 4.07 GiB

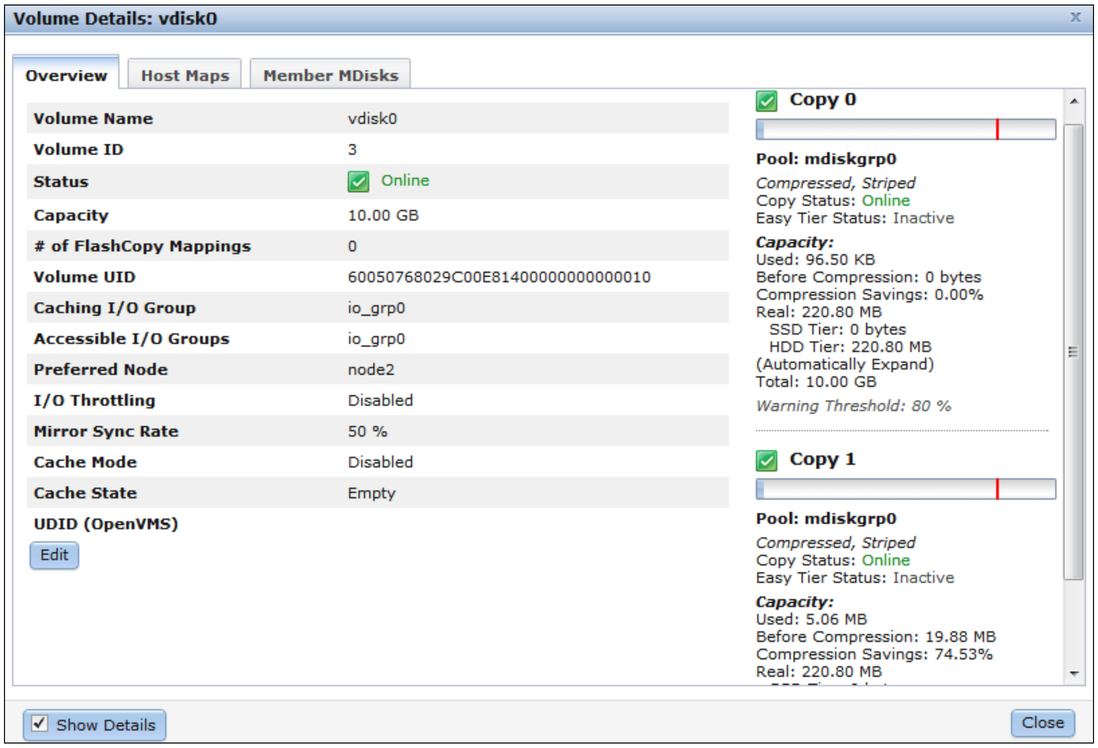
Figure 7-15 Volume details for a mirrored volume

7.4.2 Newly created mirrored compressed volumes

In the example, a new volume with Copy0 and Copy1 is added. Copy0 is basically an empty volume, and contains zeros. The compression engine has not written any data into this copy yet. When this data (zeros) is read from Copy0, it is extracted and, when synchronized to Copy1, it is compressed again. The compressed format in Copy1 consumes some metadata, which is why there is a difference between the copies.

In general, there always is some difference between compressed volume copies, mostly after the initial copy. This difference is because the actual disk space usage depends on the I/O pattern. In Copy 0, it is the original I/O to the volume, and in Copy 1, it is the 100% sequential write of the synchronization itself. After the volumes are compressed, the same I/O is performed on both copies and the results are the same.

Figure 7-16 shows that the real size and the before compression size are different.


Volume Details: vdisk0

Volume Name	vdisk0
Volume ID	3
Status	✓ Online
Capacity	10.00 GB
# of FlashCopy Mappings	0
Volume UID	60050768029C00E8140000000000010
Caching I/O Group	io_grp0
Accessible I/O Groups	io_grp0
Preferred Node	node2
I/O Throttling	Disabled
Mirror Sync Rate	50 %
Cache Mode	Disabled
Cache State	Empty
UDID (OpenVMS)	

Copy 0

Pool: mdiskgrp0
Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive
Capacity:
Used: 96.50 KB
Before Compression: 0 bytes
Compression Savings: 0.00%
Real: 220.80 MB
SSD Tier: 0 bytes
HDD Tier: 220.80 MB
(Automatically Expand)
Total: 10.00 GB
Warning Threshold: 80 %

Copy 1

Pool: mdiskgrp0
Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive
Capacity:
Used: 5.06 MB
Before Compression: 19.88 MB
Compression Savings: 74.53%
Real: 220.80 MB

Figure 7-16 Volume details vdisk0

7.5 Reporting of sparse compressed volumes

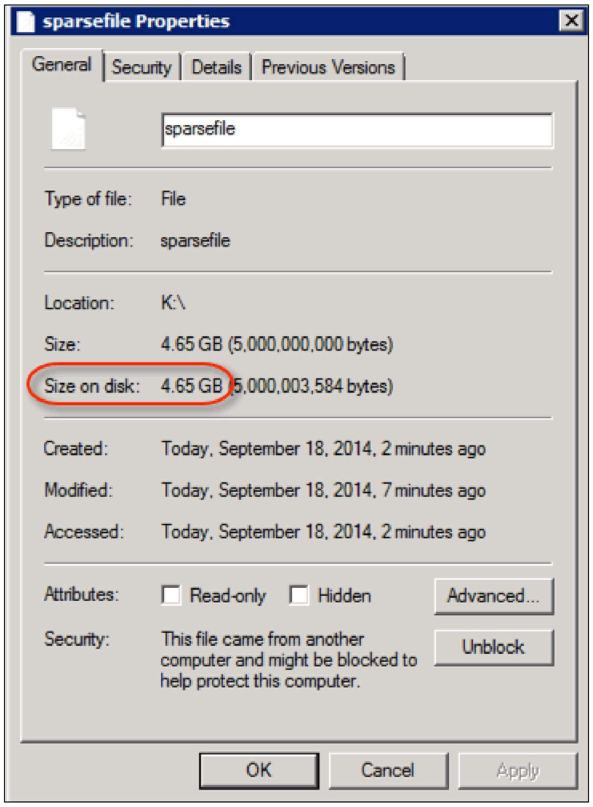
The SVC, Storwize V7000 Gen2, and FlashSystem V840 storage systems provide built-in optimization for storing large chunks of data that are empty. When a host writes data that is essentially a sequential area full of zeros to an as yet unallocated space in a thin-provisioned or compressed volume, this data is written to the disk. When a host reads an unallocated area, the system returns the same chunk of zeros. In essence, the actual storage of zeros is prevented by this mechanism.

Remember: When referring to a chunk of zeros, zero means the byte value of 0x0 (null value). It is not the ASCII value of the zero character (‘0’), which is 0x30 (or 48 in decimal).

When a host is writing zero values to an unallocated area, these writes are not accounted for by the system. As a result, the “before compression” value is lower than expected.

Explanation: The reporting “skew” occurs only if the writes are sent to unallocated areas. After the zeros are written to areas that were already written to before (overwriting previous data), the zeros are accounted for in the “before compression” value.

In the following example, a 4.65 GB file that contains large chunks of zeros is copied to a newly created file system on an empty compressed volume. Figure 7-17 shows the size of the file as viewed from the host.



sparsefile

Type of file: File

Description: sparsefile

Location: K:\

Size: 4.65 GB (5,000,000,000 bytes)

Size on disk: 4.65 GB (5,000,003,584 bytes)

Created: Today, September 18, 2014, 2 minutes ago

Modified: Today, September 18, 2014, 7 minutes ago

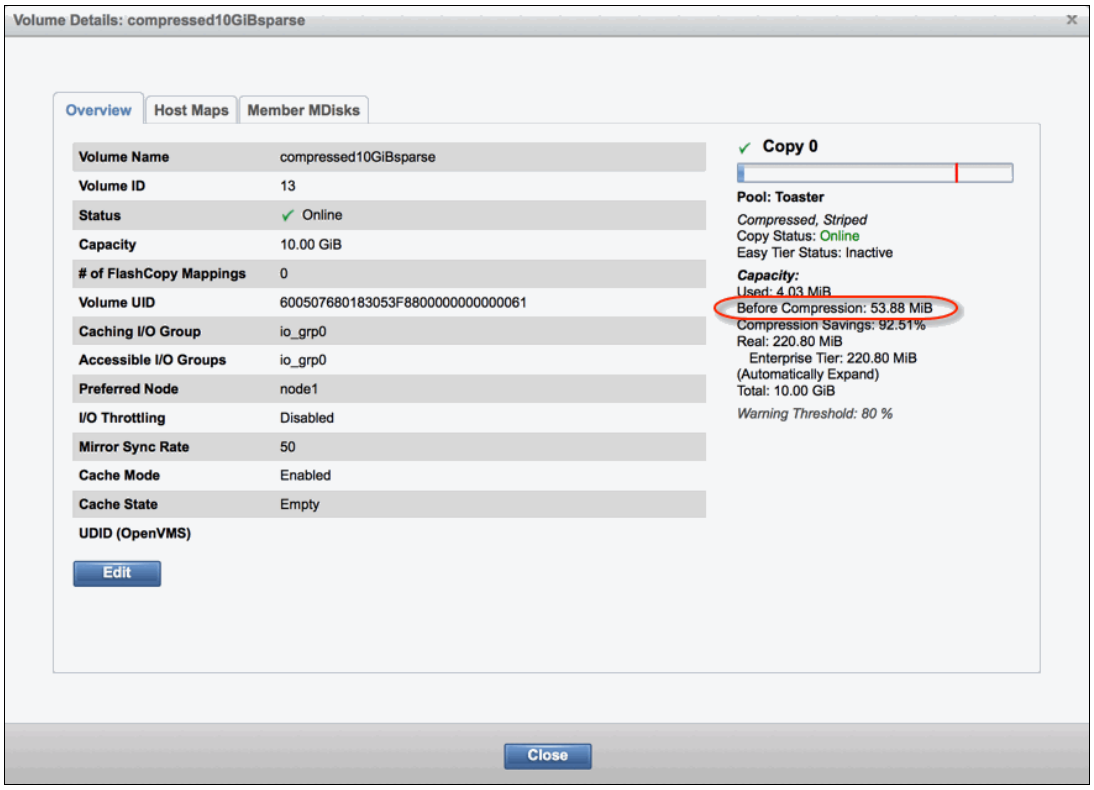
Accessed: Today, September 18, 2014, 2 minutes ago

Attributes: ☐ Read-only ☐ Hidden

Security: This file came from another computer and might be blocked to help protect this computer.

Figure 7-17 Host view of the size of a sparse file

When you view the volume properties from the storage system, the “Before Compression” size is reporting a number that is much lower than 4.65 GB. Figure 7-18 on page 115 shows that the “Before Compression” size is 53.88 MiB as viewed on the storage system.



Volume Name	compressed10GiBsparse
Volume ID	13
Status	✓ Online
Capacity	10.00 GiB
# of FlashCopy Mappings	0
Volume UID	600507680183053F8800000000000061
Caching I/O Group	io_grp0
Accessible I/O Groups	io_grp0
Preferred Node	node1
I/O Throttling	Disabled
Mirror Sync Rate	50
Cache Mode	Enabled
Cache State	Empty
UDID (OpenVMS)	

✓ **Copy 0**

Pool: Toaster

Compressed, Striped
Copy Status: Online
Easy Tier Status: Inactive

Capacity:
Used: 4.03 MiB
Before Compression: 53.88 MiB
Compression Savings: 92.51%
Real: 220.80 MiB
Enterprise Tier: 220.80 MiB
(Automatically Expand)
Total: 10.00 GiB

Warning Threshold: 80 %

Figure 7-18 Before Compression size of a volume containing a sparse file

7.6 Reporting of freed data

In some cases, the changes that are made to the data do not influence the reported size in the SVC, Storwize V7000 Gen2, or FlashSystem V840 system. This influence depends on how data was freed on the file system side, and how it is translated to actual changes to the SCSI disk itself. This section presents two possible scenarios.

7.6.1 Reporting of deleted data

When deleting data from a compressed volume, the space on the volume can still be reported as used space. It is reported this way because host file systems are freeing disk space by marking blocks as empty in the file system metadata tables. The systems are not deleting the data itself. This process translates to minor changes to the SCSI disk itself, which in turn does not free space in the compressed volume.

The compression engine does not free the space immediately. The space is freed again during the optimization process of the compression engine. The optimization cycle happens occasionally, depending on the number of writes and reads to the same disk space. Therefore, they cannot be predicted in advance or scheduled.

However, overwriting data triggers the optimization cycle. This process causes the system to reuse the block that it marked as free. This process avoids sequential writes that take more space from the disk without filling the gaps.

You can also reduce the disk space of such a volume by adding a mirrored compressed copy and deleting the original copy after the sync is complete.

7.6.2 Reporting of formatted disks

When formatting a volume, the storage system might still report on used space because of the operating system behavior of the host. Formatting file systems, such as NTFS or ext3, do not trigger complete de-allocation of space from the underlying compressed volume.

8

Reliability, availability, and serviceability

This chapter provides information about the diagnostic capabilities for the SAN Volume Controller (SVC) and Storwize V7000, and SVC Version 7.4. These capabilities are available for supporting the built-in compression software.

The content that follows is an addition to the current information that is available on SVC and Storwize V7000 RAS, monitoring, and troubleshooting capabilities. For more information, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

This chapter includes the following sections:

- Code upgrade
- Troubleshooting
- Call Home

8.1 Code upgrade

IBM Real-time Compression technology is part of the SVC software release. Updates to the IBM Random Access Compression Engine (RACE) component are provided as part of the standard SVC or Storwize V7000 software update mechanisms.

Beginning with SVC and Storwize V7000 Version 6.4, IBM Real-Time Compression is integrated into the code. For more information about how to upgrade the code of SVC or Storwize V7000, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

8.2 Troubleshooting

Events that are detected by the system are saved in an *event log*. When an entry is made in this event log, the condition is analyzed and classified to help you diagnose problems.

These events and error codes are all identical to the corresponding thin-provisioned events and error codes. However, they have a different event ID to distinguish them as compressed volume events.

For more information about working with the monitoring window, displaying events, initiating fix procedures, looking at event properties, and accessing the event log window, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

Volume compression has various error codes and informational events that are recorded in the system event log.

Consideration: The alerting facilities (event, alert, or email) are asynchronous. The events might not be received before the thin-provisioned or compressed volume goes offline. For this reason, consider an automated response to this condition. IBM Tivoli Storage Productivity Center alerts provide a mechanism for triggering automated responses. For more information, see the following website:

`http://www-03.ibm.com/systems/storage/software/center/index.html`

8.2.1 Virtual Disk Copy goes offline (error code 1865)

This error has an event ID of 060004. It is generated if the compressed Virtual Disk Copy goes offline because of insufficient space.

A compressed volume is taken offline because there is insufficient allocated real capacity available on the volume for the used space to increase further. If the compressed volume is autoexpand enabled, the storage pool it is in also has no free space.

The service action differs depending on whether the compressed volume copy is autoexpand enabled or not. Whether the disk is autoexpand enabled or not is indicated in the error event data.

Volume is autoexpand enabled

If the volume copy is autoexpand enabled, perform one or more of the following actions. After performing all of the actions that you intend to perform, mark the error as “fixed.” The volume copy then returns online.

- Determine why the storage pool free space is depleted. Any of the thin-provisioned or compressed volume copies in this storage pool might have expanded at an unexpected rate. There might be an application error. New volume copies might have been created in, or migrated to, the storage pool.
- Increase the capacity of the storage pool that is associated with the compressed volume copy by adding more MDisks to the group.
- Provide some free capacity in the storage pool by reducing the used space. Volume copies that are no longer required can be deleted, the size of volume copies can be reduced, or volume copies can be migrated to a different storage pool.
- Migrate the compressed volume copy to a storage pool that has sufficient unused capacity.
- Consider reducing the value of the storage pool warning threshold to give more time to allocate extra space.

Volume is not autoexpand enabled

If the volume copy is not autoexpand enabled, perform one or more of the following actions. In this case, the error is automatically marked as “fixed”, and the volume copy returns online when space is available.

- Determine why the compressed volume copy used space has grown at the rate that it has. There might be an application error.
- Increase the real capacity of the volume copy.
- Enable autoexpand for the compressed volume copy.
- Consider reducing the value of the compressed volume copy warning threshold to give you more time to allocate more real space.

Remember: This error also applies to thin-provisioned volumes.

8.2.2 Informational events

Informational events provide information about the status of an operation. They are recorded in the event log and, depending on the configuration, can be set to notify you through email, SNMP, or syslog.

Informational events can be either notification type I (information) or notification type W (warning). Table 8-1 provides a list of the compressed volume informational events, the notification type, and the reason for the event

Table 8-1 Informational events

Event ID	Notification type	Description
986034	I	The compressed volume copy import is successful.
986035	W	A compressed volume copy space warning has occurred.

Event ID	Notification type	Description
986036	I	A compressed volume copy repair has started.
986037	I	A compressed volume copy repair is successful.

8.3 Call Home

The SVC and Storwize V7000 can use Simple Network Management Protocol (SNMP) traps, syslog messages, and a Call Home email. These events notify you and the IBM Support Center when significant events are detected. Any combination of these notification methods can be used simultaneously.

Each event that the SVC or Storwize V7000 detects is assigned a notification type of Error, Warning, or Information. You can configure the SVC or Storwize V7000 to send each type of notification to specific recipients. For more information about Call Home for your SVC or Storwize V7000, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

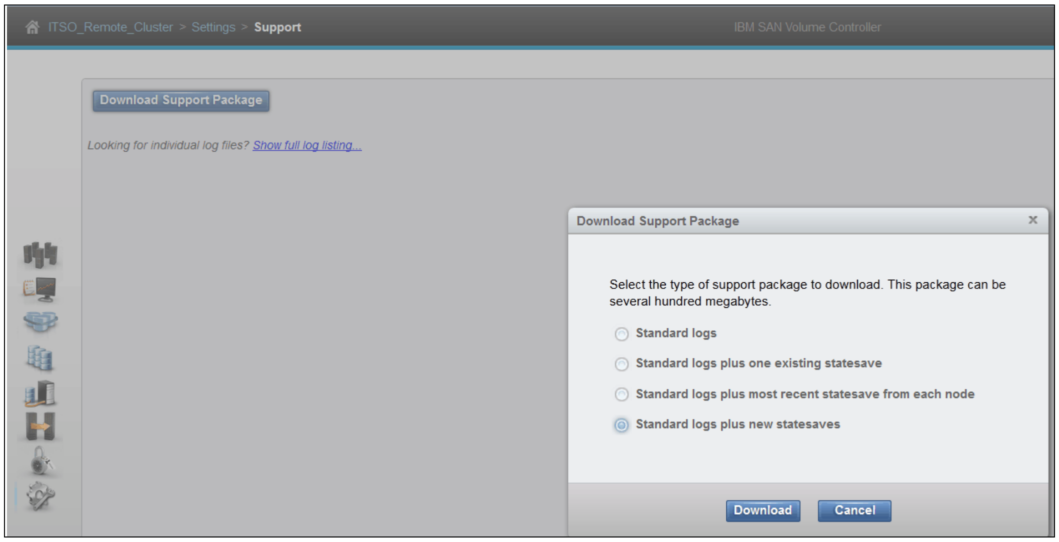
8.3.1 Collecting support information

The RACE module maintains internal diagnostic information, which is kept in memory in each of the nodes in a compression I/O Group. This information is available as part of the Support Package.

Restriction: The RACE diagnostic information is only available in a `statesave` (also called `livedump`). Standard logs do not contain this information.

To download a support package that contains the RACE diagnostic information, complete the following steps:

1. Click **Settings** → **Support** → **Download Support Package**.
2. If the system observed a failure, select **Standard logs plus most recent statesave from each node**. For all other diagnostic purposes, select **Standard logs plus new statesaves**. Figure 8-1 on page 121 shows a sample of the GUI display. For performance issues, select **Standard logs plus new statesaves**.


ITSO_Remote_Cluster > Settings > Support
IBM SAN Volume Controller
Download Support Package
Looking for individual log files? Show full log listing...
Download Support Package
Download
Cancel

Select the type of support package to download. This package can be several hundred megabytes.

- ( ) Standard logs
- ( ) Standard logs plus one existing statesave
- ( ) Standard logs plus most recent statesave from each node
- [✓ Standard logs plus new statesaves]

Figure 8-1   Support Package

3. Follow IBM support instructions to upload the diagnostic information to IBM.

8.3.2 Further information

For more information about how to download standard logs and the most recent **`statesave`** from each node in the system, see the following resources:

- *Implementing the IBM System Storage SAN Volume Controller V7.2*, SG24-7933
- *Implementing the IBM Storwize V7000 V7.2*, SG24-7938

IBM Real-time Compression in IBM SAN Volume Controller and IBM Storwize V7000

Discover how embedded compression software aids data reduction

Learn about IBM Random Access Compression Engine technology

See the compression savings that can be achieved

IBM Real-time Compression software that is embedded in IBM SAN Volume Controller (SVC) and IBM Storwize V7000 solution addresses all the requirements of primary storage data reduction, including performance, by using a purpose-built technology called real-time compression. This IBM Redpaper publication addresses the key requirements for primary storage data reduction and gives real world examples of savings that can be made by using compression.

SVC and Storwize V7000 is designed to improve storage efficiency by compressing data by as much as 80% through supported real-time compression for block storage. This process enables up to five times as much data to be stored in the same physical disk space. Unlike other approaches to compression, IBM Real-time Compression is used with active primary data, such as production databases and email systems. This configuration dramatically expands the range of candidate data that can benefit from compression. As its name implies, IBM Real-time Compression operates as data is written to disk, avoiding the need to store data that is awaiting compression.

REDP-4859-01