IBM

Redpaper

Keith Winnard
Gert Laumann
Jose Gilberto Biondo Jr

zEDC Compression: DFSMShsm Sample Implementation

This IBM® Redpaper™ describes a sample DFSMS implementation of the z Enterprise Data Compression (zEDC) primarily focused on DFSMShsm, but also includes other related data sets. The paper illustrates how to plan for the sample implementation and provides the results to illustrate the zEDC compression's impact on data in a controlled test environment.

The controlled test environment was used to allocate compressed and uncompressed data sets, and also included the impact of the selected data sets being migrated by DFSMShsm to Migration Level 1.

This paper includes the following sections:

- Overview
- Current usage of compression in DFSMShsm
- Planning for zEDC hardware compression
- Implementation of zEDC compression
- Sample Results

Overview

zEDC compression is a hardware compression feature introduced on the IBM EC12 and BC12 servers supporting data compression format on QSAM and BSAM data sets. zEDC compression is based on Extended format Version 2 data sets introduced in IBM z/OS® V2.1. zEDC compression is also available on the IBM z13.

The compression takes place in the I/O processor cards on the server, and typically helps reduce CPU cost compared to software-based compression techniques. Compressing data can reduce the cost of storing and managing data not only at rest, but also in transition on channels, mirroring links, or in the network.

Compression can also be achieved by software as an alternative to zEDC, but this tends to increase the amount of CPU to perform the compression.

Basic requirements for zEDC

zEDC hardware compression requires a licensed feature (FC#420) on the server side and IBM z/OS V2.1 (plus PTFs) and the zEDC Express for z/OS feature.

If support is required at a lower level of z/OS, apply the coexistence support by referring to your PSP bucket for the appropriate software maintenance.

zEDC hardware compression exploitation

zEDC hardware compression is supported on QSAM and BSAM only and limited to disk data sets. The candidates for this paper to illustrate a simple implementation of zEDC compression are BSAM and QSAM data sets, SMF logger data sets, and memory dumps written by DSS.

Current usage of compression in DFSMShsm

Before the announcement of zEDC hardware compression, DFSMShsm was already able to compress migration and backup data using its own compaction (based on Huffman frequency Encoding algorithm) on disk and tape data. Compression can be set by a parameter in DFSMShsm parmlib or activated through the SETSYS command. All DFSMShsm-managed data are compressed by using this software compression feature. A corresponding SETSYS parameter (COMPACTPERCENT) can be set for DFSMShsm to control a preset limit for the compaction percentage, which should be obtained at the individual data set level. The expected compaction percent is based on a previous compaction at backup or migration time.

Benefits of doing compression in DFSMShsm

DFSMShsm manages increasing volumes of data in the migration hierarchy. Only migration level 1 was in scope for our sample implementation. zEDC compression is available in the hardware. It is suggested to perform the compression at the earliest stage in the data set's lifecycle to reduce the amount of data in flight wherever possible. CPU consumption in DFSMShsm can be reduced and freed up for other purposes

Bandwidth to disk subsystems are relieved and there is less traffic on the channel subsystem. The migration level 1 should use less capacity on disk for data sets that were previously uncompressed. A further potential benefit is that data being mirrored to a secondary control unit are reduced by the compression ratio, using less capacity on the secondary site and transferring fewer blocks over the mirroring links.

Planning for zEDC hardware compression

Here are some primary considerations for planning to use zEDC compression.

Software requirements

To use zEDC hardware compression, the hardware and software requirements mentioned earlier must be in place. If your objective is to initially exploit zEDC using only DFSMShsm, ensure that z/OS is at V2.1 level on the systems where DFSMShsm will be running. If the compatibility PTFs are in place, systems on z/OS V1.12 and V.1.13 level are able to decode zEDC compressed data, but not write these in zEDC compressed format.

Therefore, it is suggested for all LPARs in the sysplex in scope are at the z/OS V 2.1 level or later before changing to zEDC hardware compression. The ROI on zEDC is increased the more it is used. DFSMShsm might be the first user, but likely, others will follow, like all types of non-VSAM extended format QSAM and BSAM.

Note: *Data sets for which DFSMSdss is not the data mover will not exploit zEDC compression. Standard DFSMShsm compression is used instead. The most common type data set type to which this applies is Partioned Data Sets.*

zBNA analysis tool

To be able to size an overall configuration that meets the environment's needs, the zBNA tool (IBM System z® Batch Network Analyzer) might be helpful to you. zBNA is a x86-based 'as is' tool that can analyze zEDC candidate data sets using SMF data as input and also help sizing the number of zEDC features needed.

zBNA is not targeted to analyze for DFSMShsm data in particular. You can use zBNA as a general analysis tool for all of your disk data.

Planning the zEDC compression test

The initial step is to run a test on zEDC hardware compression on DFSMShsm to determine the impact on the environment. Compression rates vary from one system or environment to another. A conservative estimate is a compression ratio of 4 to 1. A test will show you the average ratio (although on limited data) and be closer to showing the potential benefit.

To test and validate the zEDC hardware compression function with focus on DFSMShsm and migration to ML1, a small test environment to activate zEDC compression is required.

Select and prepare data sets of different types (DCOLLECT, SMF, or DUMP data sets as an example). Have at least one compressed and uncompressed data set of each type.

As soon as zEDC hardware compression function is implemented in DFSMShsm, it is straightforward to enable the zEDC compression and test it. It is also a simple task to revert to your previous setup by using the hardware compaction function again.

To see the benefit of zEDC compression, create a baseline of your current non-zEDC compressed environment to use it as comparison level later on as zEDC compression is done on more data.

Planning implementation of zEDC

The implementation plan presupposes that all zEDC base requirements are in place or at least planned and installed before the implementation plan can begin.

The plan should include considerations on what to do with the DFSMShsm-managed data not supported by zEDC compression. Plan for these data sets to still be compressed by the DFSMShsm compression function, by leaving the current compaction options in the DFSMShsm parmlib. Partitioned data sets are not supported by zEDC compression, but continue to use DFSMShsm standard compaction. However, DFSMShsm will use zEDC compression through DFSMSdss as the data mover for all data set types including VSAM.

Also, be aware that DFSMSdss will be called with the option ZCOMPRESS(PREFERRED). This means that if any failure occurs, DFSMShsm might not use zEDC compression or will use standard compression.

The approach for implementing the zEDC hardware compression must be decided, The choice is either to turn on the function for all migration and backup, or to use a phased implementation.

A phased approach is recommended. The granularity of the zEDC activation parameters in DFSMShsm enables you to implement the function in up to two phases:

- Compress backup data on disk
- Compress migration data on disk

Assuming the phased approach is selected, the plan should starting out enabling zEDC compression on backup data to disk, as these data are generally limited and easily identifiable. The next step would be to test, if the backup can be read, doing a restore and validation of data. The final implementation plan should have a reasonable time gap in between the implementation steps. This allows you to enable zEDC compression of data migrated to disk followed by a test and validation of the migrated data by doing a recall of these data and seeing whether they are readable. The remaining implementation plan would have the same steps: Implement the zEDC compression and validate that the data is readable until all two phases have been successfully implemented.

Post implementation steps

After you successfully implement zEDC on your environment, you will want to see the benefits of having compressed your DFSMShsm data with this new feature. Your migration level 1 should have been reduced, even though some of your data was already compressed before you migrated to zEDC hardware compression. A typical environment has a limited number of compressed data sets on primary disk, when starting to use zEDC hardware compression.

The compression ratios are higher at the start of zEDC implementation because of the number of eligible data sets are uncompressed and, therefore, the ML1 data is expected to show a high ratio of compression. However, as the use of zEDC compression expands to the primary disks and, if effective at data set allocation, then the ML1 compression ratios will be lower because the data has already been zEDC compressed. In this situation, DFSMShsm will achieve efficiency benefits because there are lower amounts of physical data to move and manage.

The ideal time to create a baseline of your capacity is implementation time, and follow up regularly to see how your disk and tape capacity is being reduced by implementing zEDC hardware compression,

zEDC Deployment

This is the sample deployment to illustrate the implementation and the impact on a controlled test environment.

Test zEDC implementation on DFSMShsm

The first step in the zEDC hardware compression implementation on DFSMShsm is to test the feature in a controlled test environment. If you decided to do a POC, this can replace the test implementation plan. It builds on the same phases as in the test plan.

Implementation in the test environment can be accomplished by issuing the SETSYS commands to inform DFSMShsm to use zEDC compression.

1. To test zEDC when backing up to DASD, issue the following modify DFSMShsm command:

   ```
   F hsm,SETSYS ZCOMPRESS(DASDBACKUP(YES))
   ```

2. To test zEDC when migrating to DASD, issue the following modify DFSMShsm command:

   ```
   F hsm,SETSYS ZCOMPRESS(DASDMIGRATE(YES))
   ```

3. For validation of these functions, backup or migrate a volume without using zEDC compression to have this uncompressed backed up volume or migrated volume available to compare to the results of the same volume that was backed up or migrated with zEDC compression.

Once testing is completed and you are ready to implement zEDC compression permanently, the following SETSYS commands should be added to the ARCCMDxx parmlib member:

```
SETSYS ZCOMPRESS(DASDBACKUP(YES))
SETSYS ZCOMPRESS(DASDMIGRATE(YES))
```

Or to utilize zEDC for all functions:

```
SETSYS ZCOMPRESS(ALL)
```

The DFSMShsm environment is now zEDC compression ready.

The next step is to test zEDC compression in DFSMShsm. Use the data sets you have already prepared for this in your planning and issue the following DFSMShsm commands to go through validation steps:

- HOLD DFSMShsm auto functions (AUTOBACKUP and AUTOMIGRATE). Holding the auto functions prevents DFSMShsm from scheduling automatic activities, and therefore gives you full manual control of directing activities to your test data sets.
- Migrate an uncompressed data set to disk by issuing command: HSEND MIGRATE DSNAME ‘data set name’ ML1.
- Migrate a compressed data set to disk by issuing command: HSEND MIGRATE DSNAME ‘data set name’ ML1.
- Back up an uncompressed data set to DISK using ARCINBAK with PARM=TARGET(DASD).
- Back up a compressed data set to DISK using ARCINBAK with PARM=TARGET(DASD).
- Run a DFSMShsm DUMP of a test volume using the ZCOMPRESS(YES) keyword on the dump class backing up this volume.

- Test the restore of backups from disk, and check that the compressed data sets are readable. The same testing should be done with the DFSMShsm compressed migrate data by recalling these and validate the contents. For the DFSMShsm dump function, restore the test volume that was backed up using zEDC compression keyword and validate that the volume being restored is valid and has the same contents as before.
- To restore DFSMShsm to the previous values, issue the following modify DFSMShsm commands:

  ```
  F hsm,SETSYS ZCOMPRESS(DASDBACKUP(NO))
  F hsm,SETSYS ZCOMPRESS(DASDMIGRATE(NO))
  Or to utilize zEDC for all functions:
  F hsm,SETSYS ZCOMPRESS(NONE)
  ```

DFSMShsm is not the only potential exploiter of zEDC compression. Combined with other exploiters like SMF LOGGER and zEDC compression in general on disk, you might see a notable percentage of reduction in your disk capacity.

Collect of zEDC hardware compression test data

Now you will want to see the outcome of having zEDC compressed test data sets. The consolidated test results are included in a table later in this chapter.

Capturing the test results is done by listing the data sets you migrated. Find the disk where these are located by issuing the command shown in Example 1.

Example 1 Search for/list of data sets tested in zEDC compression

```
hlist dsname('your data set name') both terminal
```

For disk data sets, the list shows which ML1 that the disk data set is located on. Go to this disk and look up what the data set size is on disk, and view the compression result (number of tracks) directly.

Typically you will see a reduction in allocation on disk for non-compressed data when they are compressed by zEDC. For data that is already compressed, the saving, as might be expected, is smaller when using zEDC compression.

DFSMShsm zEDC compression test results

To estimate the DFSMShsm efficiency of using zEDC compression on migrated, we tested a few scenarios after the zEDC implementation. This approach tests data sets with the same volume of data so that a valid comparison can be made.

The data sets in validation scenario #1 were all DB2 data sets. They were a combination of 4 DB2 image copies, where zEDC compression was turned off in two combinations and where DB2 compression was on in one case and off in the other (data set #1 and data set # 2). Similarly, two tests were run where zEDC compression was on. In one case with DB2 compression on, and the other case with DB2 compression off (data set # 3 and data set # 4). Finally, a test was done on DB2 archives, both with DB2 compression off. The first one was with zEDC off, and the second one with zEDC on (data set # 5 and data set # 6).

In all cases, DFSMShsm had zEDC compression turned on for migration.

Table 1 shows the results on migration to level 1.

Table 1 Scenario #1: DB2 image copy and DB2 archive migration test on zEDC

Data set #	zEDC compress	DB2 compress	Primary TRKS	Migration ML1 TRKS
#1 DB2 IC1	NO zEDC	YES	46050	22681
#2 DB2 IC	NO zEDC	NO	97485	19315
#3 DB2 IC	zEDC YES	YES	21750	22317
#4 DB2 IC	zEDC YES	NO	17670	18122
#5 DB2 AR	NO zEDC	NO	1500	375
#6 DB2 AR	zEDC YES	NO	366	367

The NOzEDC indicates that the data set was created without zEDC being active. The zEDCYES indicates that the data set was created with zEDC compression active. As can be seen from the table, zEDC compression is very effective. If zEDC compression was already done outside of DFSMShsm, the savings were not as significant. DFSMShsm used zEDC compression for the migration tasks.

A validation scenario #2 was done on more data set types (SVC dumps, DCOLLECT output data sets, and native SMF extract data set).

For each data set type, a DFSMShsm migration and backup test was done on compressed as well as on uncompressed data sets. Table 2 illustrates this scenario.

Table 2 Scenario #2 DUMP, Dcollect, and SMF dump data set migration test on zEDC

Data set scenarios	Primary TRKS	Migration ML1 TRKS
SVCdump nocompress	11032	1441
SVCdump compress	1436	1400
Dcollect DS nocompress	2655	245
Dcollect compressed	270	224
SMF dump nocompress	1900	127
SMF dump compressed	180	120

Again, the examples saw excellent compression using zEDC on the uncompressed data sets. We did more scenarios than those listed in this paper, and all showed far better compression than 4 to 1 as the announcement of zEDC indicated. But of course the compression rate depends on your data and varies from data set type to data set type.

Implementation of zEDC compression

This was the initial testing on zEDC compression on DFSMShsm data, you might have added tasks appropriate to your situation and environment. Once you are confident and more familiar with zEDC compression, you can move on with the implementation. Normally new features will be progressed across your sysplexes based on critically, starting with the least critical sysplex.

The implementation will follow the test implementation steps and sequence, but should be done in a phased manner (in 2 phases, one per function being activated), with a break within each phase that leaves time for more testing and to monitor the impact on the system over time. This break can be one or more weeks depending on individual customer quality assurance demands.

The following are suggested steps for the first implementation phase:

- Add SETSYS ZCOMPRESS(DASDBACKUP(YES)) command to DFSMShsm parmlib.
- Recycle DFSMShsm to activate the zEDC compression on disk backup data.
- Check the backup to disk and ensure that compression is active. Also, test a restore based on this backup to see that the compressed data sets are readable.

The first implementation step is now completed. The subsequent steps implement the remaining functions on DFSMShsm backup and migrate data with the planned break period between each implementation step to assess the impact.

Activation of these functions follows this same sequence. Add the ZCOMPRESS SETSYS command for the next migration/backup type, recycle DFSMShsm to have the activation happen, and run the testing according to plan for each individual function as done in the test phase.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Keith Winnard is the IBM Redbooks® Publications Project Leader for z/OS and related topics at the International Technical Support Organization, Poughkeepsie Center. He joined IT in 1977 and has worked for various clients and business partners. He is experienced in blending traditional z/OS environments and applications with web middleware and applications, and has presented on many mainframe-related topics.

Gert Laumann is an IT Specialist in Integrated Technology Delivery, Server Systems Operations/Storage Management in IBM Denmark. He has 30 years of experience in z/OS, and has worked with storage management since 1989. He is the team leader for the Scandinavianmainframe storage team. His experience is mostly with IBM products (DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm), but he has also worked with OEM software and hardware products. His focus has mainly been automation and standardization across mainframe client platforms. His focus has also been on setting up shared DASD and tape environments to mitigate data center constraints on floor space, power, and cooling. He is currently trying to achieve this efficiency across Nordic regions also by having hardware consolidated in fewer data centers.

Jose Gilberto Biondo Jr is an IT Specialist in Integrated Technology Delivery, Server Systems Operations/Storage Management in IBM Brazil. He has seven years of experience inz/OS, working with storage management since 2007. Jose works mainly with IBM storage products (DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm), but he also works with OEM software products. Jose’s areas of expertise include installing and maintaining storage products, and process automation.

Thanks to the following people for their contributions to this project:

Bob Haimowitz
Development Support Team (DST), Poughkeepsie Center

Glenn Wilcock
DFSMShsm Development Team, Tuscon

Now you can become a published author, too!

Stay connected to IBM Redbooks

Notices

This document REDP-5158-00 was created or updated on March 29, 2018.

Send us your comments 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 U.S.A.

Trademarks