Using Encryption and Compression on zFS Filesystems

Vivian W Morabito
morabito@us.ibm.com

Agenda

- Intro to Encryption & Compression
- zFS Encryption
- zFS Compression
- ICSF Considerations when using zFS Encryption
- zFS Encryption & Compression Performance Measurements
- (as time allows: Highlights of other zFS V2R3 new functionality)

INTRO: ENCRYPTION AND COMPRESSION

V2R3 Encryption & Compression Support for zFS

- **New** and **Existing** filesystems can be encrypted and / or compressed

- After a filesystem is fully encrypted or compressed, additional new entries will automatically be encrypted or compressed

V2R3 Encryption & Compression Support for zFS…

- Encrypting or compressing an **existing filesystem** is a long-running administrative command
- Application access is fully allowed to the filesystem during the operation.
- Progress may be monitored with FSINFO
- Console messages are output indicating encryption and compression activity

zFS V2R3: New long running admin commands…

- new configuration option **long_cmd_threads**
  - long_cmd_threads=foreground, background

  - Foreground threads: Handle overall long running operation(s)
    - default=1, must be in range 1-3

  - Background threads: Used by foreground thread to convert individual files
    - default=24, must be in range 1-64

- Long running commands require an available foreground thread to be initiated.

zFS V2R3: New long running admin commands…

- Long running commands (encyrpt, compress, shrink, online salvage) use background tasks on the zFS owning system to process every object in the filesystem.

- Try not to use long running commands during periods of high activity.

- Best to use them during “off shift” time periods.

Encryption / Compression

- zFS Encryption & Compression are both **V2R3 ONLY**

- **Wait until you are fully on V2R3 on all systems (with no intent to go back) before using zFS encryption and compression services!**

Encryption / Compression…

- Consider pairing encryption with compression.
    - If the compression is done first the amount of data to be encrypted is smaller which may improve performance.
- You cannot encrypt or decrypt an aggregate that is partially compressed or partially decompressed.
- You cannot compress or decompress an aggregate that is partially encrypted or partially decrypted.

ENCRYPTION

zFS Encryption

- Utilizes DFSMS Encryption for VSAM datasets
  - uses ICSF* to manage cryptographic keys

- zFS encrypts:
  - file contents
  - security information
  - ACLs (Access Control Lists)
  - symbolic link contents

* ICSF: z/OS Integrated Cryptographic Service Facility (ICSF)

zFS Encryption ...

- ICSF must be active

- Only V1.5 filesystems may be encrypted

- Filesystem must be mounted R/W

- New configuration option: **EDC_BUFFER_POOL**
  - Specifies size of real storage reserved for encryption & compression I/O
  - Default: 32 M for zFS PFS, setting may range from 1 M to 1 G

- edcfixed - IBM Recommended new option for user_cache_size parameter.
  - Allows zFS to permanently fix the user cache file to eliminate data movement from/to auxiliary buffer pool

    •user_cache_size=64M,edcfixed

Encryption – Key Label

- Encryption requires a **key label**
- encryption **key labels** identify **encryption keys** to be used to encrypt selected data sets.
- The specified key label and encryption key must exist in the ICSF key repository (CKDS)

Encryption – Key Label …

- Once a key label is associated with a filesystem, it **cannot be changed or removed**
    - *Even if filesystem has not yet been encrypted*

- The assignment of a key label does not encrypt a filesystem… it makes it eligible for encryption

Encryption – Key Label …

- Once a filesystem has a key label, it can never be owned by a member at a level lower than V2R3.
- In a mixed release sysplex with V2R3, the catchup mounts on any V2R1 or V2R2 members for a filesystem mounted RWSHARE with a key label will fail (even if it is not encrypted), resulting in z/OS UNIX function shipping to down level members
- A filesystem that has a key label can never be mounted on a sysplex that does not have a V2R3 member.
- **Do not assign key labels or encrypt zFS filesystems until you are fully at V2R3 with no intention to go back!!**

Creating a NEW encrypted filesystem

1. **Define filesystem and Assign a key label**
    - Use the keylabel option of define to assign key label
    - Use the dataclass option of define, selecting an SMS dataclass that has a keylabel
2. **Format** for encryption

Data will automatically be encrypted on write

New Encrypted filesystem: DEFINE

IDCAMS   DEFINE CLUSTER   (from JCL)

- Must use new ZFS keyword instead of LINEAR
- KEYLABEL(*keylabel*) keyword   or
- DATACLASS(*keydclass)*

zfsadm define (from the shell)

- –keylabel *keylabel*   or
- -dataclass *keydclass*

IDCAMS DEFINE CLUSTER example

```
//ZDEFKEY  JOB ,'DEFINE WITH KEYLABEL',
//            MSGCLASS=H,
//            CLASS=A,
//            TIME=(1440),MSGLEVEL=(1,1)
//DEFINE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//AMSDUMP  DD SYSOUT=H
//DASD0    DD DISP=OLD,UNIT=3390,VOL=SER=POSIX1
//SYSIN    DD *
     DEFINE CLUSTER (NAME(POSIX.DEFKEY.EXAMPLE) -
     VOLUMES(POSIX1) -
     ZFS CYL(25 0) SHAREOPTIONS(3) -
     KEYLABEL(Your_Key_label))
/*
```

zfsadm define example

zfsadm define -aggr POSIX.DEFKEYL –cyl 300 10 -keylabel *Your_Key_label*

zfsadm define -aggr POSIX.DEFKEYL –cyl 300 10 -dataclass *Your_key_Dclass*

New Encrypted filesystem: FORMAT

- New keyword(s) on format: (zfsadm format, IOEFSUTL format, IOEAGFMT)
  - -encrypt | -noencrypt

- New zFS configuration variable introduced in V2R3 to set the installation default for encryption:
  - FORMAT_ENCRYPTION ON | <u>OFF</u>
  - Specifying –encrypt or –noencrypt on format will override FORMAT_ENCRYPTION setting

- If dataset does not have a key label -encrypt and FORMAT_ENCRYPTION will have no effect

zfsadm format example

```
zfsadm format -aggr POSIX.DEFKEYL –encrypt
```

Must be a version 1.5 aggregate!

In V2R3   format_aggrversion default is 5, so I did not need to specify the version in this example

IOEFSUTL format example

```
//ZFMTKEY  JOB ,'FSUTL V5 KEYL',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
//UTLFMTK  EXEC   PGM=IOEFSUTL,REGION=0M,
//  PARM=('format -aggregate POSIX.DEFKEY.EXAMPLE -encrypt')
//IOEFSPRM DD     DSN=CFCIMGKA.PARMLIB(PARMFILE),DISP=SHR
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

IOEAGFMT format example

```
//ZFMTKEY  JOB ,'AGFMT V5 KEYL',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
//AGFMTK   EXEC   PGM=IOEAGFMT,REGION=0M,
//  PARM=('-aggregate POSIX.DEFKEY.EXAMPLE -encrypt')
//IOEFSPRM DD     DSN=CFCIMGKA.PARMLIB(PARMFILE),DISP=SHR
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Encrypting an EXISTING filesystem

Assign a key label and encrypt with **zfsadm encrypt:**

zfsadm encrypt –aggr POSIX.EXIST.FILESYS –keyl *Your_Key_label*

- The existing filesystem must be mounted in read/write mode
- The filesystem does <u>not</u> need to be SMS-managed or extended format.

- IMPORTANT: before an existing filesystem is encrypted for the first time, do a full backup of the filesystem!

Encrypting an EXISTING filesystem....

- zfsadm encrypt is a long running administrative command

- There must be an available foreground long command thread in the long running thread pool (if none available, command is failed and will need to be retried)

  - *long_cmd_threads=foreground, background*
    - *Foreground threads: default=1, must be in range 1-3*
    - *Background threads: default=24, must be in range 1-64*

It is a **one-time operation** on an existing filesystem. Once a filesystem is successfully encrypted, new data written into the filesystem will always be encrypted

Querying encryption status

zfsadm fsinfo indicates filesystem status:

encrypted

in-progress

- % encrypted, or if it has been stopped
- timestamp when long running command was started & task ID

not encrypted

zfsadm fileinfo indicates if a file is encrypted or not

- For file size > 1G fileinfo will display encrypt progress

Use fsinfo to display encryption status

```
# zfsadm fsinfo   -aggr POSIX.MY.FILESYS
File System Name: POSIX.MY.FILESYS

   *** owner information ***
   Owner:               DCEIMGKA        Converttov5:           OFF,n/a
   .
   .
   .
   Status:              RW,RS,EN,NC
   .
   .
   .

Legend: RW=Read-write, RS=Mounted RWSHARE, EN=Encrypted, NC=Not compressed
```

fsinfo shows progress for encryption in-progress

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC
File System Name: PLEX.DCEIMGNJ.BIGENC
*** owner information ***
..........
Status: RW,RS,EI,NC
...
...
Encrypt Progress: running, 23% complete started at Nov 21 14:54:40 2016 task 57F5E0
...
Legend: RW=Read-write, RS=Mounted RWSHARE, EI=Partially Encrypted
NC=Not compressed
```

fsinfo shows progress for stopped encryption

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC
File System Name: PLEX.DCEIMGNJ.BIGENC
*** owner information ***
..........
Status: RW,RS,EI,NC
...
...
Encrypt Progress: stopped, 23%
...
Legend: RW=Read-write, RS=Mounted RWSHARE, EI=Partially encrypted
NC=Not compressed
```

fileinfo: encryption status on a file

```
zfsadm fileinfo  /ict/MtPt/dir_7/file1

 path: /ict/MtPt/dir_7/file1
 ***   global data   ***
 fid                     3313,1         anode                    2380,4548
 length                  6              format                   INLINE
 .
 .
 .
 mtime       Aug  4 20:49:55 2017       atime       Aug  4 20:49:55 2017
 ctime       Aug  4 20:49:55 2017       create time Aug  4 20:49:55 2017
 reftime     none
 encrypted                              not compressed
```

Encryption of an existing aggregate

- May be cancelled (stopped):
  - Filesystem will remain partially encrypted
  - Performance will be degraded for partially encrypted filesystem… don’t let filesystem remain in this state!
  - Encryption may be resumed (using zfsadm encrypt command)

Cancel an in-progress encryption:

```
zfsadm encrypt -aggregate PLEX.DCEIMGNJ.BIGENC -cancel
IOEZ00892I Aggregate PLEX.DCEIMGNJ.BIGENC encrypt or decrypt
successfully canceled.
```

Cancelling an encryption should be a rare event

To decrypt

```
zfsadm decrypt -aggr POSIX.MY.FILESYS
IOEZ00878I Aggregate POSIX.MY.FILESYS is successfully
decrypted.
```

REMEMBER!! *Once a key label is associated with a filesystem it cannot be changed or removed, and members at releases prior to V2R3 may not own a zFS filesystem with a key label*

Encrypt / Decrypt authorization requirements:

- Superuser Authorization Required:
  - Logged in as a uid-0 user -or-
  - Have READ authority to the SUPERUSER.FILESYS.PFSCTL resource in the UNIXPRIV racf class.
- Batch formatting with encryption (IOEAGFMT / IOEFSUTL format) also requires READ access to key label

ICSF CONSIDERATIONS WHEN USING ZFS ENCRYPTION

ICSF Considerations...

ICSF performs key management for encryption.

ICSF must be operational in order for a zFS filesystem which has been assigned a key label to be mounted or opened.

zFS will wait for up to 5 minutes, issuing appropriate messages, if it detects ICSF is not up and is needed.

ICSF Considerations...

If you have BPXPRMxx mounts of encrypted zFS aggregates be sure that ICSF is started at IPL time and is operational by the time USS and zFS are started.

Start ICSF in the IPL COMMAND member as early as possible in the IPL process

S ICSF,SUB=MSTR   allows you to start ICSF without JES active.

ICSF Considerations...

The userid running the zFS started task must have READ access to any CSFKEYS profiles for aggregates that are encrypted.

ICSF Considerations...

As an alternative to permitting the userid running the zFS started task to all of the necessary security profiles, you can assign the userid running the zFS started task

- the TRUSTED attribute -or-
- the OPERATIONS attribute

Note that if zFS is running in the OMVS address space, OMVS should be given the same RACF characteristics as the zFS started task.

ICSF Considerations...

If ICSF is configured with CHECKAUTH(YES):

the userid running the zFS started task must also have at least READ access to the CSFKRR2 CSFSERV profile.

COMPRESSION

zFS Compression

- Utilizes zEDC Authorized Services (introduced in z/OS v2r1)
- Compress the contents of a filesystem. (i.e., frees up space)
- Filesystem size remains the same.

zFS Compression

- zFS compresses file data only, not directories

- Compression is “advisory”

  - If the compression of a file region does not reduce space utilization, the file region is left in its uncompressed format.
  - Always eligible for compression… if file changes and space will be saved, the file is compressed

Compression…

- Any file larger than 8K can be compressed

- Average amount of disk space that is saved per file is approximately 65%

    - *Dependent on the type of data that is being compressed.*

- zfsadm fileinfo - will indicate how many blocks were saved for compression for a particular file.

zFS Compression

- Only V1.5 filesystems may be encrypted
- Filesystem must be mounted R/W

zFS Compression

- New configuration option: **EDC_BUFFER_POOL**
  - Specifies size of real storage reserved for encryption & compression I/O
  - Default: 32 M for zFS PFS, setting may range from 1 M to 1 G

- edcfixed - IBM Recommended new option for user_cache_size parameter.

  - Allows zFS to permanently fix the user cache file to eliminate data movement from/to auxiliary buffer pool

  - If specified zFS will edcfix the user cache when compression is attempted, either due to the mount of a compressed filesystem or zfsadm compress (I.e., compressng an existing filesystem)

```
user_cache_size=64M,edcfixed
```

Creating NEW compression eligible filesystems

1. **Define** filesystem as usual

2. **Format** for compression

   - New zFS configuration variable **FORMAT_COMPRESSION** = ON | <u>OFF</u>
   - May be overwritten by using **–compress** | **-nocompress** on all flavors of format
   - Must be a V5 to be compression eligible

zfsadm format example

zfsadm format -aggr POSIX.NEWFYS.COMPRESS –compress

Must be a version 1.5 aggregate!

In V2R3   format_aggrversion default is 5, so I did not need to specify the version in this example

IOEFSUTL format example

```
//ZDEFFMT  JOB ,'DEFINE AND FORMAT',
//             MSGCLASS=H,
//             CLASS=A,
//             TIME=(1440),MSGLEVEL=(1,1)
//*---------------------------------------------------
//*  DEFINE FORMAT COMPRESSION ELIGIBLE
//*---------------------------------------------------
//DEFINE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//AMSDUMP  DD SYSOUT=H
//DASD0    DD DISP=OLD,UNIT=3390,VOL=SER=POSIX1
//SYSIN    DD *
     DEFINE CLUSTER (NAME(POSIX.NEWFS.COMPRESS) -
     VOLUMES(POSIX1) -
     ZFS CYL(25 0) SHAREOPTIONS(3))
/*
//CREATE   EXEC PGM=IOEFSUTL,REGION=0M,
// PARM=('format -aggregate POSIX.NEWFS.COMPRESS -compress')
//SYSPRINT DD SYSOUT=H
//STDOUT   DD SYSOUT=H
//STDERR   DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
```

Compressing EXISTING filesystems

```
zfsadm compress –aggr  POSIX.EXIST.FILESYS
```

- The filesystem containing the data to be compressed must be mounted read / write

- IMPORTANT: before an existing filesystem is compressed for the first time, do a full backup of the filesystem!

Querying compression status

zfsadm fileinfo indicates if a file is

- compressed and space savings
- compression eligible
- not compressed

zfsadm fsinfo indicates filesystem status:

- compressed
- in-progress
  - % compressed, or if it has been stopped
  - timestamp when long running command was started & task ID
- not compressed

FILEINFO shows compression status of a file

This example show a file that had space savings due to compression

```
zfsadm fileinfo –path testmtpt/file4
```

```
path: /home/suimgju/C81500/testmtpt/file4
***   global data   ***
.
.
.
Mtime         Jan 19 12:27:57 2017     atime          Jan 19 12:27:57 2017
ctime         Jan 19 12:27:57 2017     create time    Jan 19 12:27:57 2017
reftime       none
not encrypted                          compressed 24K saved
```

FILEINFO shows compression status of a file

This example show a file that had no space saving

zfsadm fileinfo /ict/MtPt/dir_23/large_file_23

```
path: /ict/MtPt/dir_23/large_file_23
***   global data   ***
fid                     1536,1        anode                   6668,264
.
.
.
mtime         Jul  8 12:35:04 2016    atime         Jul 24 19:27:14 2017
ctime         Jul 27 15:46:02 2017    create time   Jul 27 15:46:02 2017
reftime       none
not encrypted                         compress-eligible 0K saved
```

FSINFO shows compression status of filesystem

```
# zfsadm fsinfo -aggr POSIX.MY.FILESYS
File System Name: POSIX.MY.FILESYS

   *** owner information ***
   Owner:                DCEIMGKA       Converttov5:               OFF,n/a
   .
   .
   .
   Status:               RW,RS,NE,CO
   .
   .
   .

Legend: RW=Read-write, RS=Mounted RWSHARE, NE=Not encrypted,
CO=Compressed
```

FSINFO showing compression in-progress

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC
File System Name: PLEX.DCEIMGNJ.BIGENC
*** owner information ***
.
.

Status: RW,RS,NE,CI
.
.
Compress Progress: running, 48% started at Nov 21 16:34:40 2016 task 57F5E0
.
Legend: RW=Read-write, RS=Mounted RWSHARE, NE=Not encrypted
CI=Partially compressed
```

FSINFO showing compression stopped

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC

File System Name: PLEX.DCEIMGNJ.BIGENC
*** owner information ***
.
.
Status: RW,RS,NE,CI
.
.
Compress Progress: stopped, 48%
.
Legend: RW=Read-write, RS=Mounted RWSHARE, NE=Not encrypted
CI=Partially compressed
```

ZFS_VERIFY_COMPRESSION_HEALTH

- New health check added for V2R3 compression support
- Checks whether all user cache pages are registered with the zEDC Express service (hardware feature) when there are compressed filesystems.
- For best performance, all systems accessing the compressed data should have zEDC Express installed.

MODIFY ZFS,QUERY & zfsadm query

- Help to determine the overall compression effectiveness and allows monitoring of the performance of the zEDC.

- Compress report is included with the ALL & LFS reports

  - # of compression / decompression calls
  - Avg call time
  - KB Input : # of kilobytes sent to zEDC card for compression or decompression
  - KB output: # of kilobytes received from zEDC card for compression or decompression

KB input and output are rounded to 8K block byte counts

Decompression

```
zfsadm decompress –aggregate name [-cancel]
```

Compress / Decompress authorization requirements:

- Superuser Authorization Required:
  - Logged in as a uid-0 user -or-
  - Have READ authority to the SUPERUSER.FILESYS.PFSCTL resource in the UNIXPRIV racf class.

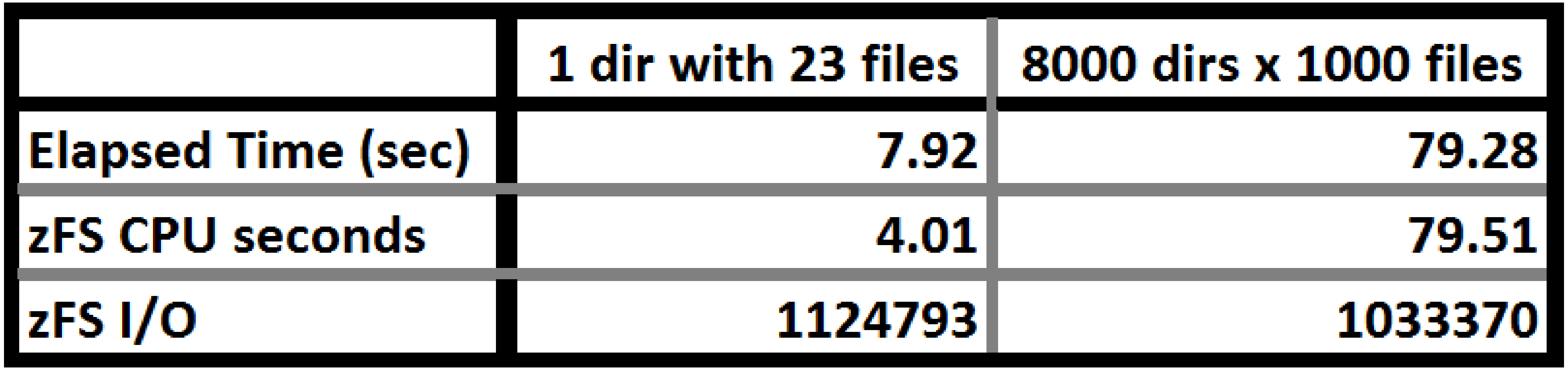
ZFS PERFORMANCE MEASUREMENTS

V2R3 Performance

- "Official" IBM Performance Measures
- Your mileage may vary
- Measurements were taken on a IBM z12 (model 2827) with 4 CPUs

zfsadm encrypt performance Measurements

directories are created each with specified # of files



	1 dir with 23 files	8000 dirs x 1000 files
Elapsed Time (sec)	7.92	79.28
zFS CPU seconds	4.01	79.51
zFS I/O	1124793	1033370

zfsadm compress Performance Measurements

directories are created each with specified # of files

running with 8GB meta cache

	1 dir with 23 files	8000 dirs x 1000 files
Elapsed Time (sec)	3.86	70.79
zFS CPU seconds	1.63	71.13
zFS I/O	633192	516868

SHRINK

Shrink:

- Reduces the physical size of the file system.
- Releases a specified amount of free space from the VSAM linear data set
- Filesystem must be mounted read/write
- Long running command: FSINFO shows progress

Shrink:

```
zfsadm shrink –aggregate name {–size Kbytes [-noai] | -cancel}
    [-trace file_name] [-level] [-help]
```

where *kbytes:*	*new total size of the shrunken filesystem*
-noai	*means "no active increase"*

"Active increase" is the default behavior for shrink

Shrink: Active Increase

Since applications can access the filesystem for most of the shrink processing, there may be additional blocks to be allocated.

If this allocation causes more space to be needed in the aggregate that the new total size specified in –size, zFS will actively increase the new total size.

The shrink command ends in error if the size is actively increased to the original size of the file system.

If –noai ("no active increase") is specified on the shrink call and an active increase is needed, the shrink command ends in error.

Shrink Processing

- Moves blocks that are in the portion of the dataset to be released into the portion that remains.
- Requires scanning each internal filesystem structure to determine if it owns any blocks that need to be moved.
- The bulk of the time in a shrink operation occurs during this scan
- After block movement completes, the free space is released.
    - Filesystem is quiesced briefly while free space is released

Shrink

- Use fsinfo output to estimate a reasonable –size for shrink

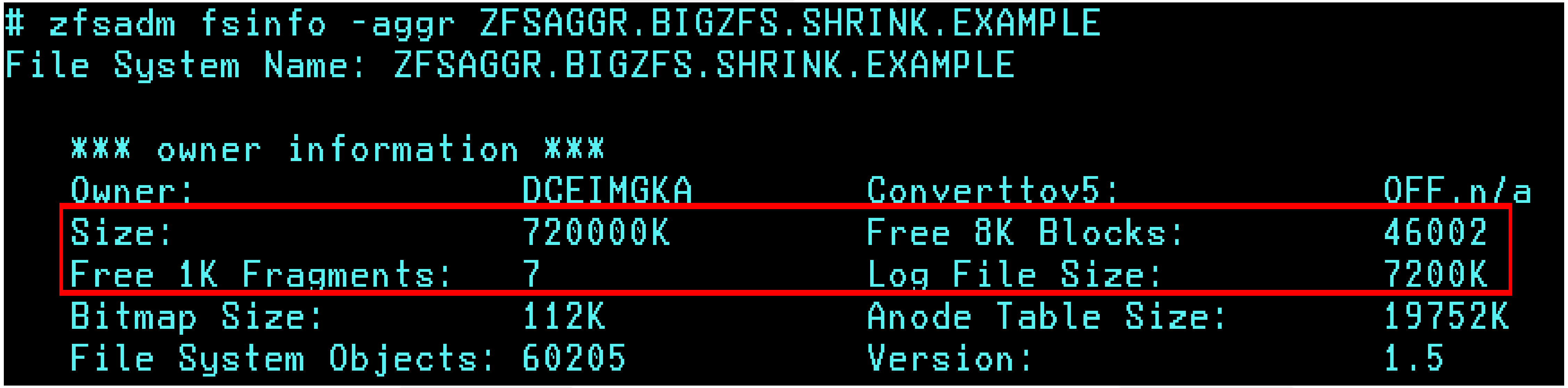
- zfsadm fsinfo will show:
  - Aggregate size (K)
  - Free size (8K blocks)
  - 1K fragments

- Don’t make -size so small that aggregate will need to grow when regular activity occurs!

- The difference between the new total size and the current size of the filesystem cannot be larger than the free size

Shrink Example:

Based on zfsadm fsinfo results, reasonable shrink size is 500000 K bytes

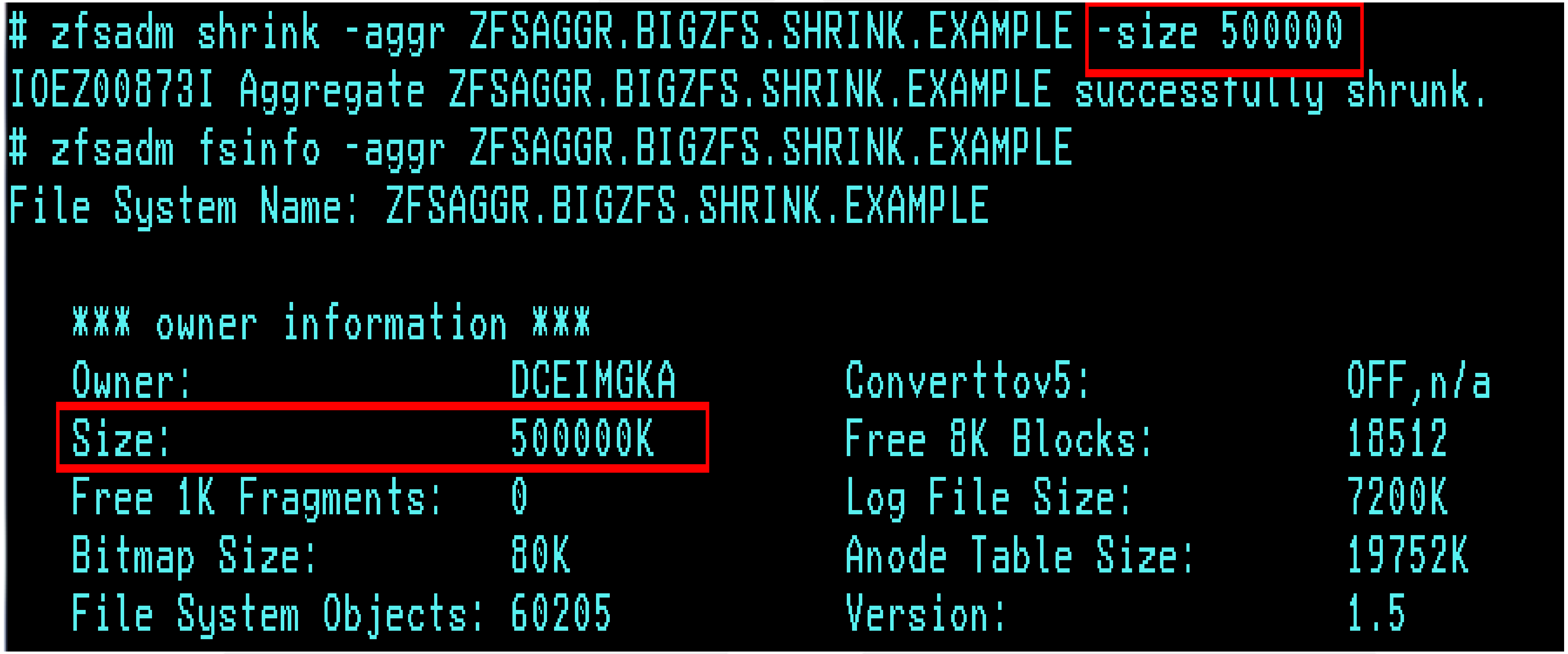


```
# zfsadm fsinfo -aggr ZFSAGGR.BIGZFS.SHRINK.EXAMPLE
File System Name: ZFSAGGR.BIGZFS.SHRINK.EXAMPLE

   *** owner information ***
   Owner:                  DCEIMGKA          Converttov5:            OFF,n/a
   Size:                   720000K           Free 8K Blocks:         46002
   Free 1K Fragments:      7                 Log File Size:          7200K
   Bitmap Size:            112K              Anode Table Size:       19752K
   File System Objects:    60205             Version:                1.5
```

zfsadm shrink -aggr ZFSAGGR.BIGZFS.SHRINK.EXAMPLE -size 500000

Shrink example (post shrink)



```
# zfsadm shrink -aggr ZFSAGGR.BIGZFS.SHRINK.EXAMPLE -size 500000
IOEZ00873I Aggregate ZFSAGGR.BIGZFS.SHRINK.EXAMPLE successfully shrunk.
# zfsadm fsinfo -aggr ZFSAGGR.BIGZFS.SHRINK.EXAMPLE
File System Name: ZFSAGGR.BIGZFS.SHRINK.EXAMPLE

   *** owner information ***
   Owner:                DCEIMGKA        Converttov5:         OFF,n/a
   Size:                 500000K         Free 8K Blocks:      18512
   Free 1K Fragments:    0               Log File Size:       7200K
   Bitmap Size:          80K             Anode Table Size:    19752K
   File System Objects:  60205           Version:             1.5
```

Shrink authorization requirements:

- UPDATE (or higher) authority to the VSAM LDS
- For a filesystem defined with guaranteed space, the ability to shrink will also be controlled by the management class conditional partial release.

zfsadm chaggr

- New admin command which changes the attributes of a mounted aggregate
  - aggrfull {threshold,increment | OFF}
  - aggrgrow {ON|OFF}
  - rwshare | norwshare
    - Causes a samemode remount (if mounted read write)
- Prior to this support, to change these attributes the aggregate must be unmounted and re-mounted.

aggrfull

- Specifies the threshold and increment for reporting aggregate utilization messages to the operator.

- May be changed to:

  threshold,increment or

  OFF

aggrgrow

- Specifies whether aggregates can be dynamically extended when they become full
- chaggr may set it ON or OFF
- Aggregate must have a secondary allocation specified (when created) and there must be space on the volumes

RWSHARE and NORWSHARE

- Both are read / write mounts

- RWSHARE is “sysplex-aware”
  - Filesystem is locally mounted on every member of the sysplex
  - File requests are handled locally by zFS

- NORWSHARE is “sysplex-Unaware”
  - File operations must be “function-shipped” to OMVS owning system

zfsadm chaggr

- Also will update the zFS mount parameters in the z/OS UNIX couple dataset, preserving the changes made with chaggr if aggregate is remounted

- (note: all mount parameters are lost if an aggregate is unmounted)

Online Salvage

- Only run from a V2R3 system.
- A new long-running command which may be used by a systems programmer to verify and / or repair a damaged filesystem, while the filesystem is still mounted.
- **Damaged filesystems are extremely rare!**
- Prior to V2R3 salvage was only available as a batch tool (IOEFSUTL or IOEAGSLV), which required the filesystem be unmounted to run.
- This V2R3 addition will allow a filesystem to be repaired while still mounted in the rare event this is necessary.

Online Salvage

- Use <u>**only**</u> when the aggregate cannot be unmounted for repairs.

- **No writes are allowed to the aggregate while a salvage operation is running**

- Verification portion (longest part) of salvage may be interrupted:
  - Using the –cancel keyword
  - Unmount immediate with the –force option
  - During a shutdown

- **Once repair has begun, salvage cannot be interrupted.**

Online Salvage

- If a user wishes to “check” a file system for corruption and NOT block write activity they could use a simple command like:

```
find  .  –name  ‘*’  –print  –exec  grep  –e  “xxxx”  {}  \;
```

  - Where “xxxx” is any string NOT found in any file.
  - This would scan all directories and files and if you get no assertions you likely have a good file system.

- <u>Online salvage should only be used as a safety valve to verify and / or repair a damaged file system that cannot be unmounted.</u>

MISCELLANEOUS

Changes in defaults

format_aggrversion:	changed from 4 to 5
change_aggrversion_on_mount:	changed from OFF to ON
honor_syslist:	obsolete in V2R3 zFS V2R3 always honors the USS automove options

New configuration options

- format_perms
- format_compression
- format_encryption
- edc_buffer_pool
- edc_fixed option added to user_cache_size
- smf_recording
- long_cmd_threads

Authorization changes

- Format authorization requirement has been changed to require UPDATE authority to the VSAM data set

- To set owner, groupid, or permissions on format, also require

  UID 0 or have READ authority to SUPERUSER.FILESYS.PFSCTL

Co-Existence APAR for z/OS V2R1 & V2R2

OA51692 is the coexistence APAR that must be applied to any V2R1 and / or V2R2 members on a sysplex where V2R3 will be present.

Make APAR OA51692 active on all systems through a rolling IPL

Miscellaneous changes

- zfsadm new option: -trace *file_name*
  - Specifies the name of a where trace records from zfsadm call will be put
    - z/OS Unix file
    - An existing sequential dataset
    - A member of an existing PDS or PDSE
  - An option on ALL zfsadm commands
  - zfsadm commands that query information:
    - no longer need READ access to IOEFSPRM file (if used)
    - Installations using parmlib (IOEPRMxx) never needed any special authorization