Differences between HFS and zFS

Jim Showalter
IBM Corporation

August 4, 2010
Session 7523

SHARE in Boston

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- DFS
- DFSMS
- DFSMSdss
- IBM
- MVS
- RACF
- RMF
- S/390
- z/OS
- zSeries

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- File access
- Directory access
- Space in a file system
- Growing a file system
- MOUNT/automount
- Backup and Quiesce
- Creating a file system
- Sysplex sharing
- Stopping the PFS

File access

- File access commands and APIs for zFS are the same as HFS except for reason codes on failures
  - z/OS UNIX reason codes – X'0000rrrr' to X'20FFrrrr'
    Documented in z/OS UNIX System Services Messages and Codes (SA22-7807) – these are common to HFS and zFS
  - HFS specific reason codes – X'5Bxxrrrr'
    Documented in z/OS UNIX System Services Messages and Codes (SA22-7807)
  - zFS specific reason codes – X'EFxxrrrr'
    Documented in z/OS Distributed File Service Messages and Codes (SC24-5917)
  - The **bpxmtext** shell command can be used to display the meaning of zFS reason codes (as of z/OS V1R8) and z/OS UNIX reason codes

Directory access

- Directory access commands and APIs for zFS are the same as HFS except for a few non-obvious situations
  - HFS returns names in a directory in (some) alphabetical order (using opendir, readdir, closedir APIs)
    DO NOT BECOME DEPENDENT ON THIS ORDER
    - This is not POSIX behavior
    - It is not controlled by localization envars (LC_COLLATE)
    - **ls** returns sorted names but that is because –C is the default
    - The order that HFS returns names is not the same as **ls** (ls -C returns uppercase characters first; HFS returns uppercase characters last.)
    - zFS returns names unsorted – as every other UNIX file system does

Directory access …

- zFS directories can be read as a file; HFS directories return 0 bytes (using open, read, close APIs)
  - You can see this by using the **strings** command against a directory


```
# cd /zfsmnt2
# df -v .
Mounted on     Filesystem                Avail/Total    Files      Status
/zfsmnt2       (PLEX.JMS.AGGR004.LDS0004) 1000/25920     4294967269 Available
ZFS, Read/Write, Device:27, ACLS=Y
AGGRGROW
File System Owner : DCEIMGVM    Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : PLEX.JMS.AGGR004.LDS0004
# ls -a
.          A          abc        acldir     file2      go.o       test1
test4.txt  test6.dat  test8.txt
..         ab         abcd       file1      file4      linkname   test3
test5.txt  test7.txt  testdir
# strings -n 1 -t d /zfsmnt2
       4 .
      13 ..
      23 test1
      36 test3
      49 test4.txt
      66 test5.txt
      83 test6.dat
     100 test7.txt
     117 test8.txt
     134 testdir
     149 linkname
     165 A
     174 ab
     184 abc
     195 abcd
     207 acldir
     221 go.o
     233 file1
     246 file2
     259 file4
```

Directory access …

- zFS is limited to 64K subdirectories per directory
- HFS reports that it is limited to 64K
  (but may support more?)
  DO NOT CREATE MORE THAN 64K SUBDIRECTORIES IN A DIRECTORY


```
# df -v /zfsmnt3
Mounted on     Filesystem              Avail/Total    Files      Status
/zfsmnt3       (PLEX.JMS.AGGR005.LDS0005) 2522566/2604960
4294967274 Available
ZFS, Read/Write, Device:28, ACLS=Y
NBS,NONBS
File System Owner : DCEIMGVM    Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : PLEX.JMS.AGGR005.LDS0005
# getconf LINK_MAX /zfsmnt3
65535
#
# df -v /
Mounted on     Filesystem              Avail/Total    Files      Status
/              (PLEX.CFCIMGVM.ROOT)    88/1440        4294967225
Available
HFS, Read/Write, Device:1, ACLS=Y
File System Owner : DCEIMGVM    Automove=Y      Client=N
Filetag : T=off   codeset=0
# getconf LINK_MAX /
65536
#
```

Directory access …

- zFS has a performance problem with large directories
  - As you approach 100,000 entries in a zFS directory, performance begins to suffer
  - If you can,
    - spread out entries among multiple directories, or
    - try to remove older files to keep directory from getting too large, or
    - use HFS for this directory
  - There is some guidance on this in the z/OS Distributed File Service zSeries File System Administration book (SC24-5989) in Chapter 4, “Minimum and maximum file system sizes”.

Space in a file system

- HFS uses 4K blocks to store file data
- zFS uses 8K blocks but can store multiple (small) files in an 8K block
  - Files < 53 bytes are stored in the inode (with the metadata)
  - Files between 53 and 7K are stored in 1K fragments in an 8K block
  - Files > 7K are stored in 8K blocks
- Fragmented files can cause confusion about free space in zFS
  - **df** can report, for example, 20K of free space
  - but, if there are no free 8K blocks (that is, there are only free fragments), then you cannot, for example, create a 14K file
  - **zfsadm aggrinfo *aggregate_name* –long** shows detailed information including the number of free 8K blocks
- See z/OS Distributed File Service zSeries File System (SC24-5989), Chapter 4, zFS disk space allocation for more information

zfsadm aggrinfo PLEX.JMS.AGGR004.LDS0004 -long

PLEX.JMS.AGGR004.LDS0004 (R/W COMP): 500 K free out of total 12960

version 1.4

auditfid C3C6C3F0 F0F0051E 0000

55 free 8k blocks; 60 free 1K fragments

112 K log file; 24 K filesystem table

8 K bitmap file

Space in a file system …

- An HFS that is multi-volume must be SMS managed (and cataloged)
- A zFS that is > 4GB must be extended addressability in the data class definition and therefore SMS managed (A VSAM LDS is always cataloged)

Growing a file system

- Dynamic grow
  - HFS will grow when a write occurs and out of space if
    - A secondary allocation size was specified when created, and
    - there is space available on the volume(s)
  - zFS will grow when a write occurs and out of space if
    - A secondary allocation size was specified when created, and
    - there is space available on the volume(s), and
    - **aggrgrow=on** is specified in the zFS IOEFSPRM configuration file or **AGGRGROW** is specified on the MOUNT PARM

From TSO/E

MOUNT FILESYSTEM('OMVS.MNT.FS1.ZFS') TYPE(ZFS)
MODE(RDWR) MOUNTPOINT('/zfsmnt1') PARM('AGGRGROW')

From the z/OS UNIX shell

/usr/sbin/mount -t ZFS -f OMVS.MNT.FS1.ZFS -o 'AGGRGROW' /zfsmnt1

Growing a file system …

- Explicit grow with a command
  - HFS can be explicitly grown with the z/OS UNIX command **confighfs –x 10c *pathname***
  - zFS can be explicitly grown with the z/OS UNIX zFS command **zfsadm grow *aggregate_name new_total_k_bytes***
- Residing on the EAS (extended addressing space) portion of an EAV (extended address volume)
  - HFS is not eligible
  - zFS is eligible as of z/OS V1R10

zfsadm aggrinfo displays the current size of the aggregate in K-bytes

df –k also displays the current size of the aggregate in K-bytes

MOUNT/automount

- The MOUNT PARMs for HFS and zFS are different
  (the other options are the same – MOUNTPOINT, MODE, etc.)
  - HFS MOUNT PARMs (MOUNT TYPE(HFS))
    - PARM('FSFULL(*threshold,increment*)')
    - PARM('NOSPARSE')
    - PARM('NOWRITEPROTECT')
    - PARM('SYNC(*sec*)')
    - PARM('SYNCRESERVE(*nn*)')
  - zFS MOUNT PARMs (MOUNT TYPE(ZFS))
    - PARM('AGGRFULL(*threshold,increment*)')
    - PARM('AGGRGROW')
    - PARM('NBS')
    - PARM('RW')
    - PARM('RWSHARE')

HFS MOUNT PARMs described in z/OS MVS Initialization and Tuning Reference (SA22-7592)

zFS MOUNT PARMs described in z/OS Distributed File Service zFS Administration (SC24-5989)

MOUNT/automount …

- Generic file system TYPE on MOUNT
  - If you specify TYPE(HFS) and the data set is not HFS or is not found, it is treated as ZFS (and you get a zFS reason code)
  - If you specify TYPE(ZFS) and the data set is HFS, it is treated as HFS
  - In each of these cases where the TYPE did not match the actual data set type, **THE MOUNT PARMs ARE DISCARDED**
    (we don't want the mount to fail due to invalid PARMs)
  - Once you have fully migrated a file system from HFS to zFS, you should specify TYPE(ZFS) so that MOUNT PARMs are effective

MOUNT/automount …

- zFS is a logging file system (metadata is logged, not file data)
- Maintains file system consistency – log is replayed on next mount if file system was not cleanly unmounted – requires a R/W mount
- Problem scenario can occur
  1. R/O file system (for example, version root) needs to be updated
  2. Remount R/O to R/W
  3. Update file system
  4. Before remount back to R/O, system is re-IPLd
  5. Mount of R/O version root **fails** because log needs to be replayed
- Should always do MODIFY OMVS,SHUTDOWN before a planned system shut down
- With R9 APAR OA20615, zFS provides new IOEFSPRM option (romount_recovery=on)

The romount_recovery=on IOEFSPRM configuration option says that if the log needs to be replayed and it is a R/O mount, then zFS will temporarily mount the file system R/W, replay the log and then unmount and mount the file system R/O. The default for romount_recovery is off. You can also dynamically set this option with **zfsadm config –romount_recovery on**.

Backup and Quiesce

- DFSMSdss automatically quiesces a mounted file system on backup to ensure data integrity
  - For HFS, quiesce is a BPX1QSE call
    - df /hfsmntpoint displays Status of Quiesced
    - D OMVS,F,N=OMVS.HFS.FS1 displays Status of QUIESCED
    - D OMVS,F,E considers the HFS to be in an exception state
    - Message BPXF083I THE FOLLOWING FILE SYSTEM HAS BEEN QUIESCED FOR MORE THAN 10 MINUTES: OMVS.HFS.FS1 QUIESCING SYSTEM=DCEIMGVM JOB=SUIMGVM PID=67174418 LATCH=44.
  - For zFS, quiesce is a BPX1PCT call
    - Neither df, nor D OMVS,F show the ZFS to be quiesced
    - zfsadm aggrinfo omvs.zfs.fs1 displays status of quiesced
    - zfsadm lsaggr displays status of quiesce
    - Message IOEZ00581E There are quiesced zFS aggregates. After about 30 seconds

If another sysplex member joins, an HFS (must be R/O) will not be mounted on the joining system until the HFS is unquiesced. See z/OS UNIX System Services Programming: Assembler Callable Services Reference, Chapter 2, quiesce, Characteristics and Restrictions. A ZFS will be mounted as soon as the system joins.

Creating a file system

- An HFS is created with a DD JCL statement
  - DSNTYPE=HFS, and
  - SPACE=(40,1,1) – you must specify the number of directory blocks but the number is not used
- A zFS is created with IDCAMS/formatted with IOEAGFMT
  - DEFINE CLUSTER LINEAR
  - Specify –compat in the PARM of the IOEAGFMT step

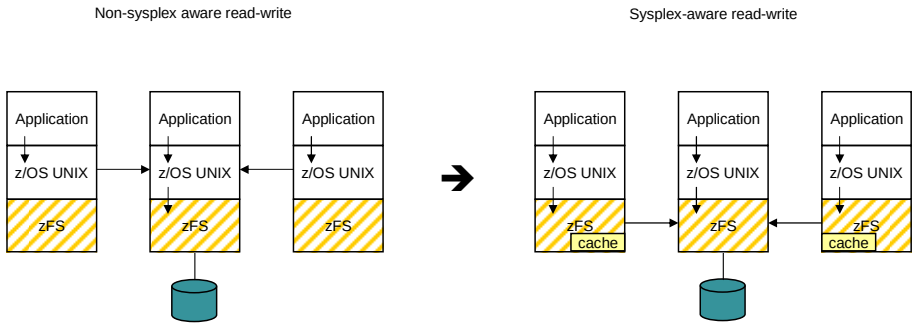

```
//USERIDA JOB
//STEP1   EXEC PGM=IEFBR14
//HFS1 DD DSN=OMVS.HFS.HOME,
//        SPACE=(CYL,(40,1,1)),
//        DSNTYPE=HFS,
//        DISP=(NEW,CATLG,DELETE),
//        STORCLAS=STANDARD

//USERIDA  JOB
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSIN    DD *
 DEFINE CLUSTER (NAME(OMVS.ZFS1) -
       VOLUMES(PRV000) -
       LINEAR -
       CYL(40 1) -
       SHAREOPTIONS(3))
/*
//STEP2    EXEC PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.ZFS1 -compat')
//SYSPRINT DD SYSOUT=H
```

Sysplex sharing

- Both HFS and zFS support read-write sharing from multiple systems in a shared file system environment (BPXPRMxx SYSPLEX(YES))
  - HFS read-write file systems are always non-sysplex aware (z/OS UNIX always uses function shipping to a single owning system)
  - zFS read-write file systems can be sysplex-aware or non sysplex aware
    (For sysplex-aware read-write, z/OS UNIX sends requests to the local zFS and then zFS may function ship the request if it needs to – if the data is not in the cache)

Sysplex sharing …



As of z/OS V1R11, zFS running in a shared file system environment supports sysplex-aware read-write file systems. Sysplex-aware read-write file systems can improve performance when the file system is accessed from multiple systems or when file systems require manual movement to optimize access performance. The preferred method is to specify IOEFSPRM sysplex=filesys. After all your systems are sysplex=filesys, then choose which zFS read-write file systems you want to be sysplex-aware and specify the RWSHARE MOUNT PARM. (sysplex=filesys requires R11 zFS APAR OA29619). See SHARE Session 2272 from Seattle 2010 for a full presentation on this zFS capability.

Stopping the PFS

- HFS cannot be stopped separately from z/OS UNIX
  - HFS runs in the z/OS UNIX address space
  - MODIFY OMVS,SHUTDOWN stops z/OS UNIX (and HFS)
- ZFS can be stopped
  (although you should not need to do this)
  - ZFS runs in a separate address space
  - MODIFY OMVS,STOPPFS=ZFS stops ZFS
    Restarted by replying 'R' to message BPXF032D
  - MODIFY OMVS,SHUTDOWN stops ZFS and then z/OS UNIX

The FILESYSTYPE statement in the BPXPRMxx for HFS looks like this (the PARM is optional):

```
FILESYSTYPE TYPE(HFS) ENTRYPOINT(GFUAINIT)
            PARM('SYNCDEFAULT(30)')
```

The FILESYSTYPE statement in the BPXPRMxx fpr zFS looks like this:

```
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS,'SUB=MSTR')
      PARM('PRM=(01,&SYSCLONE.)')
```

Notice the ASNAME parameter. This is what causes zFS to run in a separate address space (it must run that way).
Also, the SUB=MSTR option causes zFS to run so that it is not under control of JES. That means that you do not
need to stop ZFS in order to stop JES. This is the recommended way to run ZFS.
Finally, the PARM specifies that zFS configuration options will be found in parmlib member IOEPRM01 first and then in
parmlib member IOEPRMxx, where xx is the sysplex member's two character shorthand notation for the system name.
If you specify a IOEZPRM DD statement in your ZFS PROC, the PARM in the FILESYSTYPE is ignored.
(You will see message IOEZ00178I.)
The IOEZPRM DD is the "old" way to point to your zFS configuration options file. The FILESYSTYPE PARM
(with the IOEZPRM DD in your ZFS JCL PROC commented out) is the recommended way to point to your
ZFS configuration options. (You will see message IOEZ00374I.)

Publications

- z/OS UNIX System Services Planning (GA22-7800)
  General Administration of z/OS UNIX file systems
- z/OS UNIX Command Reference (SA22-7802)
  confighfs command for HFS
- z/OS MVS System Messages Volume 9 (IGF-IWM) (SA22-7639)
  IGWxxxt messages for HFS
- z/OS UNIX System Services Messages and Codes (SA22-7807)
  z/OS UNIX return codes, z/OS UNIX reason codes, X'5Bxxrrrr' reason codes for HFS
- z/OS Distributed File Service zSeries File System Administration (SC24-5989) – **was refreshed in April**
  zFS Concepts and zfsadm command for zFS
- z/OS Distributed File Services Messages and Codes (SC24-5917) – **was refreshed in April**
  IOEZxxxt messages and X'EFxxrrrr' reason codes for zFS
- z/OS Distributed File Service zSeries File System Implementation (SG24-6580)
  - Redbook available (updated February 2010 to include z/OS V1R11)
  - http://www.redbooks.ibm.com/abstracts/sg246580.html?Open
- z/OS Version 1 Release 8 Implementation (SG24-7265)
  - Redbook available (contains zFS updates for z/OS V1R8)
  - http://www.redbooks.ibm.com/abstracts/sg247265.html?Open
- z/OS DFSMS™ Access Method Services for Catalogs (SC26-7394)
  IDCAMS utility
- z/OS DFSMS™ Storage Administration Reference (SC26-7402)
  ADRDSSU utility for backup