# Why Are You Still on HFS?
# zFS V5: The Future Awaits!

*Thursday, August 13, 2015*
*01:45 PM - 02:45 PM,*

*Dolphin, Southern Hemisphere 1*

*Vivian W Morabito*

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

# Benefits of migrating from HFS to zFS

- Higher Performance
- Greater Reliability
- Expanded functionality and tuning
- zFS is IBM's strategic filesystem for z/OS
    - *All future enhancements will be made in zFS*

# zFS Performance

**V1R11:** zFS sysplex-aware fully supports shared filesystem for both admin and file operations

**V1R13:** zFS sysplex-aware support enhanced allowing clients direct access to files (“Direct I/O”).

**V2R1:** Introduction of v5 filesystems to resolve known large directory performance problems.

Sysplex client directory performance improvements for both existing v4 and new v5 filesystems

**V2R2:** zFS kernel to run in AMODE64 and can optionally run in the OMVS address space

# zFS sysplex-aware and “Direct I/O”

- **V1R11:** zFS forwards requests to the owner for sysplex-aware filesystems

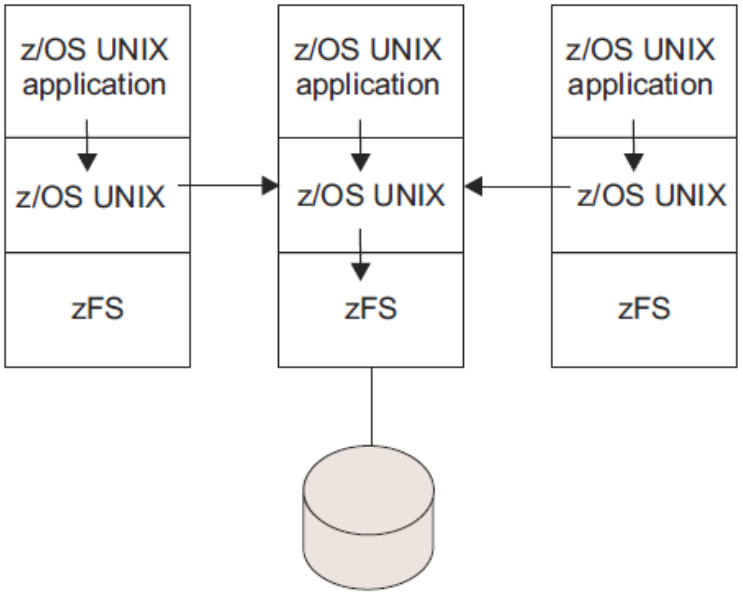
.

  - *Eliminates the need for Unix Systems to function ship requests.*

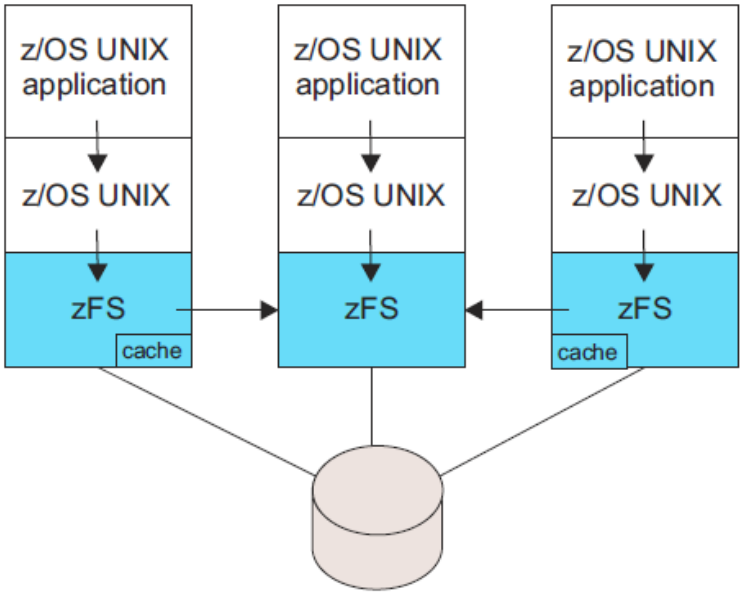
- **V1R13:** zFS further enhances its sysplex-aware support providing direct I/O from clients

  - *Reduces the overhead of using XCF communications*.

zFS read/write file system mounted with NORWSHARE




zFS read/write file system mounted with RWSHARE



# zFS for Large Directories

- Prior to V2R1, there were known performance problems with large zFS directories
    - Affects directories with more than ~10,000 entries
- In V2R1, zFS provides a new v5 directory format to resolve this problem
    - Uses extensible hash algorithm
    - Provides improved performance for large directories
    - Maintains POSIX semantics.
    - max  num. of sub-directories  is increased (4G-1)

# zFS V2R2 …. Performance keeps getting better!!

- zFS Kernel runs AMODE64
- zFS Kernel can be configured to run in the OMVS address space
- zFS will support higher system limits
  - meta_cache_size
  - vnode_cache_size
  - token_cache_size
- The logging system has been re-written in V2R2 providing I/O efficiency and more parallelism resulting in significant performance gains

# HFS / zFS performance comparison (V1R13)

- **VERIFY workload** is single-threaded test of various file I/O functions. All measurements on z990 processor
- Record sizes vary from 80, 2K, 4K, 8K, 32K, 64K,bytes
- ETR rates are reported in MB/second
- **Monoplex Results:**
- Create Sequential File
  - HFS : 7 - 13 MB/Sec (tops out with 2K record size)
  - zFS : 9 - 606 MB/Sec (Far superior for 2K through 64K record sizes. Increasing record size improves performance)
- Read Sequential File (non-cached)
  - HFS : 8 - 19 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 8 - 23 MB/Sec (20% better for 2K through 64K record sizes)

# HFS / zFS performance comparisons (V1R13)…

- VERIFY workload, Monoplex results (cont'd)
- Read Sequential File (cached)
  - HFS : 8 - 20 MB/Sec (no caching benefit)
  - zFS : 8 - 856 MB/Sec (Far superior for 2K through 64K record sizes. Increasing record size improves performance)
- Create Random File
  - HFS : 0.05 - 10 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 6 - 500 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)
- Read Random File (non-cached)
  - HFS : 0.08 - 12 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 3 - 12 MB/Sec (Only 80 bytes size shows improvement)
- Read Random File (cached)
  - HFS : 0.08 - 12 MB/Sec (no caching benefit)
  - zFS : 7 - 809 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)

# HFS / zFS performance comparisons (V1R13)…

- VERIFY workload
- Sysplex Client Results (two system sysplex)
- Create Sequential File
  - HFS : 0.2 - 11 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 8 - 580 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)
- Read Sequential File (non-cached)
  - HFS : 0.2 - 13 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 7 - 23 MB/Sec (Slightly better for all record sizes)
- Read Sequential File (cached)
  - HFS : 0.2 - 13 MB/Sec (no caching benefit)
  - zFS : 7 - 845 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)

# HFS / zFS performance comparisons (V1R13)…

- VERIFY workload, sysplex (cont.)
- Create Random File
  - HFS : 0.04 - 8 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 6 - 443 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)
- Read Random File (non-cached)
  - HFS : 0.05 - 4 MB/Sec (Increasing record size slightly improves performance)
  - zFS : 3 - 12 MB/Sec (Slightly better for all record sizes)
- Read Random File (cached)
  - HFS : 0.05 - 4 MB/Sec (no caching benefit)
  - zFS : 6 - 809 MB/Sec (Far superior for all record sizes. Increasing record size improves performance)

# HFS / zFS performance comparisons (V1R13)…

- **FSPT workload** is multi-threaded test of random 4K file I/O with 65/35 R/W ratio
- ETR and ITR improvements vary by zFS caching levels
- Monoplex Results (single owning system)
  - 0% zFS caching : ETR 10% and ITR 44% less than HFS (startup costs for zFS are slightly higher)
  - 75% zFS caching : ETR 5X and ITR 2X better than HFS
  - 100% zFS caching : ETR 128X and ITR 5X better than HFS
  - Based on this data zFS starts to outperform HFS in terms of ETR at 10% and ITR at 60% cache hits

# HFS / zFS performance comparisons (V1R13)…

- FSPT workload,
- Sysplex Client Results (two system sysplex)
  - 0% zFS caching : ETR 29% and ITR 2X better than HFS
  - 100% zFS caching : ETR 176X and ITR 22X better than HFS
  - zFS Sysplex Aware and Direct I/O always makes zFS better than HFS

# V2R1 & V2R2 Performance Workload Descriptions

- No zFS HFS comparisons available after V1R13, but the following results demonstrate the continuing zFS improvement in directory operations

- Performance results (on subsequent slides) were obtained using 3 workloads created by IBM:
  - All workloads involved many tasks processing 2000 objects in each directory, for multiple directories, on multiple file systems (see slide notes).
  - **ptestDL2** – This workload did repeated lookup (name searches) by the tasks.
  - **ptestDL** – The tasks did repeated lookup and readdir functions.
  - **ptestDU** – The tasks performed directory create/update/remove/readdir/search.

# zFS V2R1 and V2R2 Performance Gains

- Tests were run varying the sizes of the involved directories to test scale-ability.
- Results on next page show:
  - z/OS V2R1 V5 relative improvement over V1R13
  - V2R2 v5 relative improvement over V2R1 v5
- For further details, refer to presentation “zFS v5 Migration and Performance”

# zFS V2R1 and V2R2 Performance Gains

| | | Monoplex Results | | Sysplex Client Results | |
|---|---|---|---|---|---|
| Workload | Directory Sizes (names / directory) | z/OS V2R1 V5 relative improvement over V1R13 | V2R2 v5 relative improvement over V2R1 v5 | z/OS V2R1 V5 relative improvement over V1R13 | V2R2 v5 relative improvement over V2R1 v5 |
| ptestDL2 | 2,000 | EΔ = 1.197<br>IΔ = 1.197 | EΔ = 1.54<br>IΔ = 1.45 | EΔ = 1.416<br>IΔ = 1.378 | EΔ = 1.35<br>IΔ = 1.20 |
| | 20,000 | EΔ = 4.536<br>IΔ = 4.536 | EΔ = 1.60<br>IΔ = 1.51 | EΔ = 7.271<br>IΔ = 7.098 | EΔ = 1.63<br>IΔ = 1.37 |
| | 50,000 | EΔ = 10.026<br>IΔ = 10.026 | EΔ = 2.10<br>IΔ = 1.63 | EΔ = 16.668<br>IΔ = 16.297 | EΔ = 1.88<br>IΔ = 1.40 |
| ptestDL | 2,000 | EΔ = 1.167<br>IΔ = 1.167 | EΔ = 1.14<br>IΔ = 1.14 | EΔ = 1.454<br>IΔ = 1.452 | EΔ = 1.03<br>IΔ = 1.04 |
| | 20,000 | EΔ = 5.978<br>IΔ = 5.977 | EΔ = 1.14<br>IΔ = 1.14 | EΔ = 77.549<br>IΔ = 51.830 | EΔ = 1.07<br>IΔ = 1.08 |
| | 50,000 | EΔ = 9.160<br>IΔ = 9.160 | EΔ = 1.12<br>IΔ = 1.12 | EΔ = 276.67<br>IΔ = 174.81 | EΔ = 1.13<br>IΔ = 1.10 |
| ptestDU | 2,000 | EΔ = 1.167<br>IΔ = 1.167 | EΔ = 1.54<br>IΔ = 1.45 | EΔ = 1.249<br>IΔ = 1.226 | EΔ = 1.35<br>IΔ = 1.20 |
| | 20,000 | EΔ = 3.313<br>IΔ = 3.586 | EΔ = 1.60<br>IΔ = 1.51 | EΔ =8.940<br>IΔ = 6.822 | EΔ = 1.63<br>IΔ = 1.37 |
| | 50,000 | EΔ = 6.158<br>IΔ = 7.069 | EΔ = 2.10<br>IΔ = 1.63 | EΔ = 65.110<br>IΔ = 47.891 | EΔ = 1.88<br>IΔ =1.40 |

# zFS: Greater Reliability

- zFS has made improvements to reduce situations which:
  - Require re-mount of filesystem(s)
  - Require re-IPL

- zFS has a lower exposure to situations that could result in corruption of a filesystem… and in the very rare situation that a corruption does occur, zFS has a salvager to correct most corruptions.

# zFS Internal Restart (V1R13)

- There are (unusual) situations where the zFS PFS may go down on a sysplex member
- Prior to V1R13, all zFS filesystems owned on that system are moved or unmounted
- With this V1R13 support, when a such a situation occurs, zFS will internally go down, restart and internally remount any zFS file systems that were locally mounted
- zFS will not read IOEFSPRM configuration options during this internal restart.
- **zFS internal restart will clear up almost any internal zFS problem without losing the filesystem tree**
- *Note that in a single system environment if zFS goes down, you will loose the mount tree.*

# zFS filesystem corruption avoidance & correction

- Filesystem corruption is a very rare event
  - *HFS filesystems can become permanently corrupted if a system outage occurs during an HFS sync. zFS does not have this issue*.
- zFS will disable a filesystem to try to prevent a corruption
- In the rare event that a corruption occurs, zFS has a salvager that will repair most filesystem corruptions.

# zFS auto-takeover of disabled aggregates (V1R13)

- If a zFS filesystem becomes disabled it must be unmounted and then mounted again to recover
- Prior to this support this was a manual operation
- With this support, in a zFS sysplex-aware environment, zFS will attempt to automatically recover the disabled filesystem
  - *The owning system will request that another sysplex-aware system take over zFS ownership*
  - *In a single system, same-mode remounts are used to resolve disabled aggregates.*

# zFS Queries

- zFS provides query functions that provide extensive information on system utilization, files, directories, and filesystems

- V2R1: fileinfo
  - Displays detailed information about a file or directory

- V2R2: fsinfo
  - Displays detailed information about single or multiple filesystems including sysplex-wide detailed information

- These queries are not available on HFS.

# zFS system utilization queries

- zFS has extensive query commands to show system statistics.
- Data that comes from these queries allow the understanding of zFS's use of system resources and allows zFS to be tuned to optimize performance for the installation.
- Queries include:
  - Storage
  - Counters for the various caches
  - I/O by dasd or aggregate
  - PFS calls on the owner
  - Locking statistics

# fileinfo query

- **zfsadm fileinfo – displays detailed file information:**
  - **zfsadm fileinfo -path <*pathname*> [{-globalonly | -localonly | -both}]**

```
path: /home/suimghq/lfsmounts/PLEX.ZFS.FS2/DCEIMGHQ.ftestLD.p6/.
   ***   global data   ***
   fid                     7,174515      anode                   130931,2028
   length                  33546240      format                  BLOCKED
   1K blocks               21664         permissions             755
   uid,gid                 0,10          access acl              33,72
   dir model acl           33,72         file model acl          32,72
   user audit              F,F,F         auditor audit           N,N,N
   set sticky,uid,gid      0,0,0         seclabel                none
   object type             DIR           object linkcount        39686
   object genvalue         0x00000000    dir version             5
   dir name count          39686         dir data version        160308
   dir tree status         VALID         dir conversion          na
   file format bits        na,na,na      file charset id         na
   file cver               na            charspec major,minor    na
   direct blocks           0x000D91FE    0x000D91FF    0x000D9200    0x000D9201
                           0x000D9202    0x000D9203    0x000D9204    0x000D9205
   indirect blocks         0x000D9206    0x00052199
   mtime       May 28 17:09:45 2013   atime        May 28 17:09:19 2013
   ctime       May 28 17:09:45 2013   create time  May 28 15:14:48 2013
   reftime     none
```

# fsinfo query

- New file system query command to provide more information, faster, with selection and sorting criteria
- It offers the ability to get and sort information for one or more file systems that have common names, common attributes, or that have encountered similar unexpected conditions
- Available from the shell (zfsadm), console (MODIFY) or via API call.
- Can display basic output, information know only to owner, only on local system, or full information.
- RMF enhancements will use new fsinfo api interface, providing improved performance and additional information

# FSINFO – full (brown = info not previously available)

```
File System Name: PLEX.ZFS.SMALL2

   *** owner information ***
   Owner:                DCEIMGHQ          Converttov5:              OFF,n/a
   Size:                 40320K            Free 8K Blocks:           905
   Free 1K Fragments:    7                 Log File Size:            32800K
   Bitmap Size:          8K                Anode Table Size:         80K
   File System Objects:  28                Version:                  1.5
   Overflow Pages:       0                 Overflow HighWater:       0
   Thrashing Objects:    0                 Thrashing Resolution:     0
   Token Revocations:    37                Revocation Wait Time:     85.141
   Devno:                46                Space Monitoring:         90,5
   Quiescing System:     n/a               Quiescing Job Name:       n/a
   Quiescor ASID:        n/a               File System Grow:         OFF,0
   Status:               RW,RS,GD,SE
   Audit Fid:            C3C6C3F0 F0F001F9 0000

   File System Creation Time: Nov 14 00:07:36 2014
   Time of Ownership:         Dec  9 11:10:36 2014
   Statistics Reset Time:     Dec  9 11:10:35 2014
   Quiesce Time:              n/a
   Last Grow Time:            n/a

   Connected Clients:    n/a


Legend: RW=Read-write, GD=AGGRGROW disabled, RS=Mounted RWSHARE
        SE=Space errors reported
```

# FSINFO – full (output continued)

```
*** local data from system DCEIMGHQ (owner: DCEIMGHQ) ***
  Vnodes:                410              LFS Held Vnodes:          30
  Open Objects:          2                Tokens:                   29
  User Cache 4K Pages:   13               Metadata Cache 8K Pages:  35
  Application Reads:     104355143        Avg. Read Resp. Time:     0.151
  Application Writes:    39105328         Avg. Writes Resp. Time:   0.641
  Read XCF Calls:        0                Avg. Rd XCF Resp. Time:   0.000
  Write XCF Calls:       0                Avg. Wr XCF Resp. Time:   0.000
  ENOSPC Errors:         15920695         Disk IO Errors:           0
  XCF Comm. Failures:    0                Canceled Operations:      0

  DDNAME:                SYS00008
  Mount Time:            Dec  9 11:10:35 2014

VOLSER PAV    Reads      KBytes     Writes     KBytes     Waits
Average
------ --- ---------- ---------- ---------- ---------- ---------- ----
-----
CFC000   1     821059    6997808     537165   14633376     974568
2.805
------ --- ---------- ---------- ---------- ---------- ---------- ----
-----
TOTALS         821059    6997808     537165   14633376     974568
2.805
```

# Improved quiesced filesystem displays

- df Shell command:
- Console Commands:
  - D OMVS,F
  - F ZFS,QUERY,FILESETS,QUIESCED

# Misc zFS Advantages and Improvements

- Improved FFDC
- Intelligent hang detector
- More stressful and varied testing on zFS
- Filesystem backup improvement in V2R1

# zFS is the strategic filesystem for z/OS

- No additional enhancements will be made to HFS
- zFS will continue to be enhanced in future releases!