zFS V5 Migration and Performance

Wednesday, August 12, 2015

01:45 PM - 02:45 PM

Dolphin, Bay

Vivian W Morabito

Agenda

- V1.5 Filesystems, extended (v5) directories
- v5 performance
- v4 to v5 Migration

Version 1.5 Fileystems

- Introduced in z/OS V2R1
- Version 1.5 filesystems can contain both v4 & new extended (v5) directories.
- Filesystems can be large! Up 16TB
  - (v1.4 filesystems max size is 4TB)
- Can only be mounted on V2R1 or later
- Optional, and are not the default in z/OS V2R1 or V2R2
- Installations should only start to use V5 filesystems when fully migrated to z/OS V2R1 or later with no plans to go back to prior releases.

Extended (v5) directories

- Support large directories with improved performance
  - use a tree directory format, providing much faster insert / search / delete performance
- Store names more efficiently than older v4 directories
- Space can be reclaimed when names are removed from v5 directories
  - v4 directories do not reclaim space until the directory is removed.
- Support large number of subdirectories (4G-1)
  - (v4 limit is 65535)

V5 Filesystem Performance

- V5 filesystems provide significant performance gains due to new tree directory structure
  - Directory searches scale well as directories increase in size
  - Directory searches & updates on the same directory can generally be performed in parallel.

Performance Workload Descriptions:

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

- External Throughput (E): # ops / unit time
  - Higher E ➔ lower average response time
- Internal Throughput (I): # ops / unit of processor time
  - Higher I ➔ each CPU operation took less time
- Tests were run with various directory sizes to assess the scalability of the performance improvement.

Performance Runs:

- On a z9 processor showing the improvement (in V2R1)
  - for V4: comparing V2R1 V4 with V1R13 V4
  - for V5: comparing V2R1 V5 with V1R13 V4
- On a z196 processor showing the V2R2 improvement over V2R1 for V5 directories.
- Comparisons are made for both monoplex & sysplex for each of the 3 workloads.

Performance runs made in V2R1 time frame (showing general improvement & gain V5 over V4)

- **ptestDL2** (dir search) Results on z9 / FICON connected DASD

	Monoplex Results			Sysplex Client Results		
Directory Sizes	R13 Operations / Second	z/OS 2.1 V4 Ratio over R13	**z/OS 2.1 V5** Ratio over R13	R13 Operations / Second I per processor	z/OS 2.1 V4 Ratio over R13	**z/OS 2.1 V5** Ratio over R13
0 Base Names (2000 names per directory)	E=307703 I=307703	EΔ 1.005 IΔ 1.005	**EΔ 1.197 IΔ 1.197**	E=232427 I=55067/proc	EΔ 0.996 IΔ 0.980	**EΔ 1.416 IΔ 1.378**
18k Base Names (20000 names per directory)	E=80840 I=80840	EΔ 0.964 IΔ 0.964	**EΔ 4.536 IΔ 4.536**	E=44532 I=10536/proc	EΔ 0.933 IΔ 0.933	**EΔ 7.271 IΔ 7.098**
48k Base Names (50000 names per directory)	E=34598 I=34598	EΔ 0.964 IΔ 0.964	**EΔ 10.026 IΔ 10.026**	E=18308 I=4333/proc	EΔ 0.918 IΔ 0.918	**EΔ 16.668 IΔ 16.297**

- V5 search performance scales almost linearly with directory size
- V5 file system monoplex performance improves **20%** for small directories, **10X** for directories with 50,000 names.
- V5 sysplex client performance improves **40%** even for small directories, **16X** for directories with 50,000 names

Performance runs made in V2R1 time frame (showing general improvement & gain V5 over V4)

- **ptestDL** (dir search & readdir) Results on z9 / FICON Connected DASD

	Monoplex Results			Sysplex Client Results		
Directory Sizes	R13 Operations / Second	z/OS 2.1 V4 Ratio over R13	z/OS 2.1 V5 Ratio over R13	R13 Operations / Second I per processor	z/OS 2.1 V4 Ratio over R13	z/OS 2.1 V5 Ratio over R13
0 Base Names (2000 names per directory)	E=297285 I=297345	EΔ 1.002 IΔ 1.002	EΔ 1.167 IΔ 1.167	E=216433 I=50474/proc	EΔ 1.064 IΔ 1.078	EΔ 1.454 IΔ 1.452
18k Base Names (20000 names per directory)	E=33908 I=33918	EΔ 2.032 IΔ 2.032	EΔ 5.978 IΔ 5.977	E=2620 I=638/proc	EΔ 12.834 IΔ 8.840	EΔ 77.549 IΔ 51.830
48k Base Names (50000 names per directory)	E=12717 I=12720	EΔ 2.258 IΔ 2.258	EΔ 9.160 IΔ 9.160	E=384 I=97/proc	EΔ 35.490 IΔ 22.237	EΔ 276.67 IΔ 174.81

- V5 monoplex improvement: **17%** for small, **9x** for large directories
- V5 sysplex client improvement: **45%** for small, **277x** for large directories

Performance runs made in V2R1 time frame (showing general improvement & gain V5 over V4)

- **ptestDU** (dir reading & writing) Results on z9 / FICON connected DASD

	Monoplex Results			Sysplex Client Results		
Directory Sizes	R13 Operations / Second	z/OS 2.1 V4 Ratio over R13	**z/OS 2.1 V5** Ratio over R13	R13 Operations / Second I per processor	z/OS 2.1 V4 Ratio over R13	**z/OS 2.1 V5** Ratio over R13
0 Base Names (2000 names per directory)	E=109584 I=119372	EΔ 1.055 IΔ 1.022	**EΔ 1.167 IΔ 1.167**	E=65384 I=11076/proc	EΔ 1.053 IΔ 1.059	**EΔ 1.249 IΔ 1.226**
18k Base Names (20000 names per directory)	E=31688 I=31943	EΔ 1.566 IΔ 1.483	**EΔ 3.313 IΔ 3.586**	E=7441 I=1675/proc	**EΔ 3.528 IΔ 2.780**	**EΔ 8.940 IΔ 6.822**
48k Base Names (50000 names per directory)	E=12667 I=12682	EΔ 1.741 IΔ 1.751	**EΔ 6.158 IΔ 7.069**	E=830 I=205/proc	**EΔ 17.472 IΔ 12.568**	**EΔ 65.110 IΔ 47.891**

- V5 monoplex performance improves **17%** for small**,** **6x** for larger directories
- V5 sysplex client performance improves **25%** for small, **65x** for larger directories
  - *Runs in last 2 rows were hurt by small meta cache size… since these runs were made in V2R1, due to zFS storage constraints it was not possible to run with larger caches (this has been resolved in V2R2 with 64bit zFS)*

Performance runs in V2R2 time frame (showing improvements in V2R2 over V2R1 for V5 dirs)

- **ptestDL2** (dir search) Results on z196
- All measurements made running in the USS (OMVS) Address Space

	Monoplex Results		Sysplex Client Results	
Directory Sizes	z/OS V2R1 V5 Operations / Second	**z/OS 2.2 V5** Ratio over z/OS V2R1 V5	z/OS V2R1 V5 Operations / Second I per processor	**z/OS 2.2 V5** Ratio over z/OS V2R1 V5
0 Base Names (2000 names per directory)	E= 309442 I= 328321	**EΔ = 1.54** **IΔ = 1.45**	E= 219434 I= 35731	**EΔ = 1.35** **IΔ = 1.20**
18k Base Names (20000 names per directory)	E= 254871 I= 271399	**EΔ = 1.60** **IΔ = 1.51**	E= 165778 I= 28342	**EΔ = 1.63** **IΔ = 1.37**
48k Base Names (50000 names per directory)	E= 159308 I= 205294	**EΔ = 2.1** **IΔ = 1.63**	E= 123516 I= 24090	**EΔ = 1.88** **IΔ = 1.4**

- V2R2 monoplex performance improves **54%** for small directories, **2X** for directories with 50,000 names.
- V2R2 sysplex client performance improves **35%** even for small directories, **88%** for directories with 50,000 names

Performance runs in V2R2 time frame (showing improvements in V2R2 over V2R1 for V5 dirs)

- **ptestDL** (dir search & readdir) Results on z196
- All measurements made running in the USS (OMVS) Address Space

	Monoplex Results		Sysplex Client Results	
Directory Sizes	z/OS V2R1 V5 Operations / Second	z/OS 2.2 V5 Ratio over z/OS V2R1 V5	z/OS V2R1 V5 Operations / Second I per processor	z/OS 2.2 V5 Ratio over z/OS V2R1 V5
0 Base Names (2000 names per directory)	E= 807211 I= 807373	EΔ = 1.14 IΔ = 1.14	E= 808759 I= 194357	EΔ = 1.03 IΔ = 1.04
18k Base Names (20000 names per directory)	E= 474689 I= 474689	EΔ = 1.14 IΔ = 1.14	E= 481902 I= 80688	EΔ = 1.07 IΔ = 1.08
48k Base Names (50000 names per directory)	E= 228514 I= 228514	EΔ = 1.12 IΔ = 1.12	E= 276036 I= 43505	EΔ = 1.13 IΔ = 1.16

- V2R2 monoplex performance improves **14%** for small directories, **12%** for directories with 50,000 names.
- V2R2 sysplex client performance improves **3%** even for small directories, **13%** for directories with 50,000 names

Performance runs in V2R2 time frame (showing improvements in V2R2 over V2R1 for V5 dirs)

- **ptestDU** (dir reading & writing) Results on z196
- All measurements made running in the USS (OMVS) Address Space

	Monoplex Results		Sysplex Client Results	
Directory Sizes	z/OS V2R1 V5 Operations / Second	**z/OS 2.2 V5** Ratio over z/OS V2R1 V5	z/OS V2R1 V5 Operations / Second I per processor	**z/OS 2.2 V5** Ratio over z/OS V2R1 V5
0 Base Names (2000 names per directory)	E= 309442 I= 328321	**EΔ = 1.54** **IΔ = 1.45**	E= 219434 I= 35731	**EΔ = 1.35** **IΔ = 1.20**
18k Base Names (20000 names per directory)	E= 254871 I= 271399	**EΔ = 1.60** **IΔ = 1.51**	E= 165778 I= 28342	**EΔ = 1.63** **IΔ = 1.37**
48k Base Names (50000 names per directory)	E= 159308 I= 205294	**EΔ = 2.1** **IΔ = 1.63**	E= 123516 I= 24090	**EΔ = 1.88** **IΔ = 1.40**

- V2R2 monoplex performance improves **54%** for small directories, **2x** for directories with 50,000 names.
- V2R2 sysplex client performance improves **35%** even for small directories, **88%** for directories with 50,000 names

So what about V5 performance at your installation?

- **Typical Customer Usage Pattern:**
  1. Directory search (lookup) most common operation, or at least one of the most frequent. *Similar to ptestDL2 workload*
  2. File Open/Read/Write/Close the next most common operations. *ptestDL workload combines reads with lookup*
  3. Directory update operations generally a much lower percentage of calls to zFS.

→**F ZFS,QUERY,KNPFS** will show you your workload characteristics in terms of what operations are most common for you.

Identifying which filesystems to convert to V1.5

- **Filesystems with large directories will benefit most**
  - As shown in the prior slides, directory performance is improved even for smaller directories (2000 names) but directories over 10,000 names will likely see non-trivial reduction in directory operation time inside zFS.
  - **ls –slk** will show your directory size on disk in kilobytes.

    →Any directory over 160K will benefit with v5

- **Convert the largest most active filesystems to V1.5 first.**

  →The **F ZFS,QUERY,FILESETS** can be used to identify your most active file systems.

v4 to v5 migration

- **Creating new V1.5 filesystems**
  - IOEFSUTL, IOEAGFMT, zfsadm format or API
- **Changing existing V1.4 filesystems to V1.5**
  - Explicitly for a mounted filesystem,
  - automatically on mount, or
  - Offline
- **Converting existing v4 directories to extended v5**
  - Explicitly one at a time,
  - automatically as they are accessed, or
  - Offline

Creating new V1.5 Filesystems

- New directories created in a V1.5 filesystem are extended v5 directories.

- IOEFSPRM parameter: format_aggrversion = 4 | 5
  - Specifies the default version to when formatting an aggregate
  - Can be overridden by options used in format invocation
  - Default is 4 (creates a v1.4 aggregate)

Creating new V1.5 Filesystems…

- IOEFSUTL and IOEAGFMT batch utilities take an optional parameter to specify the version of the filesystem to format:

  –version4 or –version5

```
//USERIDA JOB ,'Compatibility Mode',
// CLASS=A,MSGCLASS=X,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=PRV000
//SYSIN DD *
     DEFINE CLUSTER (NAME(OMVS.PRV.COMPAT.AGGR001) -
            VOLUMES(PRV000) -
            LINEAR CYL(25 0) SHAREOPTIONS(3))
/*
//CREATE EXEC PGM=IOEFSUTL,REGION=0M,
// PARM=('format -aggregate OMVS.PRV.COMPAT.AGGR001 -version5')
//SYSPRINT DD SYSOUT=H
//STDOUT DD SYSOUT=H
//STDERR DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP DD SYSOUT=H
//*
```

Creating new V1.5 Filesystems…

- zfsadm format has an optional parameter
  –version4 or –version5

```
# zfsadm define -aggr POSIX.SHARE.EXAMPLE -cyl 5 0 -vol POSIX0
IOEZ00248I VSAM linear dataset POSIX.SHARE.EXAMPLE successfully created.
# zfsadm format -aggr POSIX.SHARE.EXAMPLE -version5
IOEZ00077I HFS-compatibility aggregate POSIX.SHARE.EXAMPLE has been successfully
 created
```

Creating new V1.5 Filesystems…

- format aggregate API:

  aggptr->af_aggrversion

  – 0: default version specified by format_aggrversion
  – 4: version 1.4
  – 5: version 1.5

Changing existing V1.4 filesystems to V1.5

- **zfsadm convert –aggrversion**
  - Will change the aggregate version from V1.4 to V1.5
  - Filesystem must be mounted or attached.

```
# zfsadm convert -aggrversion  POSIX.SHARE.VIVIAN
IOEZ00810I Successfully changed aggregate POSIX.SHARE.VIVIAN to version 1.5.
```

Changing existing V1.4 filesystems to V1.5…

- To automatically change the version from V1.4 to V1.5 on mount:
  - IOEFSPRM parameter **change_aggrversion_on_mount**
    - Changes only the filesystem version, no directories are converted from v4 to extended v5.
  - IOEFSPRM parameter **converttov5=on**
  - Mount parm **CONVERTTOV5**
    - Changes the filesystem version to V1.5 and converts v4 directories to extended v5 as they are accessed.

```
# mount -t zfs -f POSIX.SHARE.VIVIAN -o CONVERTTOV5 /tmp/MtPt
```

Changing existing V1.4 filesystems to V1.5…

- There is also a **NOCONVERTTOV5** mount parameter
- Both CONVERTTOV5 and NOCONVERTTOV5 will override IOEFSPRM settings for change_aggrversion_on_mount and converttov5
  - Useful if there are a few exceptional filesystems that you do or don't want to convert.

Changing existing V1.4 filesystems to V1.5…

- Offline using the **IOEFSUTL converttov5** batch utility using the –aggrversion_only option
  - Changes from V1.4 to V1.5 only, no directories are converted from v4 to v5 with this option!

Changing existing V1.4 filesystems to V1.5…

An aggregate is not explicitly or automatically changed from V1.4 to V1.5 if there are releases in the sysplex prior to z/OS V2R1

Converting existing v4 directories to extended v5

- zfsadm convert -path will explicitly convert the specified directory.

```
# zfsadm convert -path /tmp/MtPt/dir1
IOEZ00791I Successfully converted directory /tmp/MtPt/dir1 to version 5 format.
```

- v4 directories can be subdirectories of v5, and
- v5 directories can be subdirectories of v4

Converting existing v4 directories to extended v5…

- If the fileystem has the CONVERTTOV5 attribute set directories are automatically converted as they are accessed

  - CONVERTTOV5 can be set either by IOEFSPRM (default) or via an explicit mount parameter.

Converting existing v4 directories to extended v5…

- **IOEFSUTL converttov5**
  - Batch utility which will convert all directories contained in the filesystem.

```
//USERIDA JOB ,'Convert to version 5',
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//CONVERT EXEC PGM=IOEFSUTL,REGION=0M,
// PARM=('converttov5 -aggregate OMVS.PRV.COMPAT.AGGR001')
//SYSPRINT DD SYSOUT=H
//STDOUT DD SYSOUT=H
//STDERR DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP DD SYSOUT=H
//*
```

- **IOEFSUTL converttov4**
  - Will allow you to convert back to v4 if needed
    - *assuming not exceeding v4 subdir limit 65535 or size limit 4TB*

Converting existing v4 directories to extended v5…

- If the filesystem is not already V1.5, conversion of the first directory it contains will change its version to V1.5

- Converting a directory from v4 to v5 requires that both versions of the directory exist temporarily on disk.
  - If the aggregate becomes full during allocation of a new directory, a dynamic grow will be attempted.
  - The size of the new v5 directory will vary based on the directory contents.
  - If a system outage occurs during a directory conversion, the directory will be made consistent during log recovery processing (either the old or new directory will exist, but not both)

Version 1.5 filesystem disk space usage

- Version 1.5 filesystem pages use a prefix/postfix scheme to pack more names in a directory page
  - If the names fit a common pattern at the beginning or end, up to 4X more names can fit in a page over a V1.4 filesystem.
- v5 directories are tree structured, and a new name is placed in the tree according to the hashing algorithm.
- v4 directories can place a new name in any directory page that it fits.
- Therefore, there are some cases a v5 directory could use less space than a v4, and some cases where it could use more.
- **IBM expects that the disk space used by v5 directories will be roughly equivalent to v4 directories on average.**

IBM recommended v5 exploitation strategy

1. **Delay converting remaining HFS to zFS file systems until z/OS 2.1:**
   - Unless you have an immediate need to convert an HFS to zFS (typically due to file IO performance issues with HFS):
     - Wait until your site is at z/OS 2.1 and later before migrating the file system to zFS
       - So you can use v5 format for improved directory performance and avoid a conversion from v4 to v5 format.
2. **Do not use v5 until fully ready to commit to z/OS 2.1 for all systems.**
   - Going back to v4 via IOEFSUTL could be painful, wait until it's safe before using v5.
3. **Set format_aggrversion=5:**
   - Future file systems get created as version 5
     - And avoids having to change JCL and other programs.
4. **Set change_aggrversion_on_mount=ON**
   - Safe since it is a fast operation and ensures future directories are v5 format.
5. **Determine if CONVERTTOV5 can be globally enabled**
   - This depends on how many directories are accessed at IPL time, the number of names in each and the known zFS conversion performance results and the expected amount of delay to the system IPL.
     - User has to decide if they can tolerate the expected **one-time** IPL delay
     - And if they can, simply specify **CONVERTTOV5**=ON in **IOEFSPRM**.
6. **And if cannot globally enable, then:**
   - Determine highest usage file systems via RMF, or **F ZFS,QUERY,FILESETS** and commands shown on slide 9.
   - Also look at file systems with large directories, especially if they are high usage and then use the **CONVERTTOV5 MOUNT** parameter selectively.