

# Everything You Wanted to Know About z/OS UNIX Sysplex File Sharing

Vivian W Morabito

Thursday March 13, 2014 1:30-2:30PM

Grand Ballroom Salon G



morabito@us.ibm.com

# Table of Contents

The Basics (bpxprm setup, file system structures)

Alternate sysplex root

Sysplex aware terminology

RWSHARE/NORWSHARE

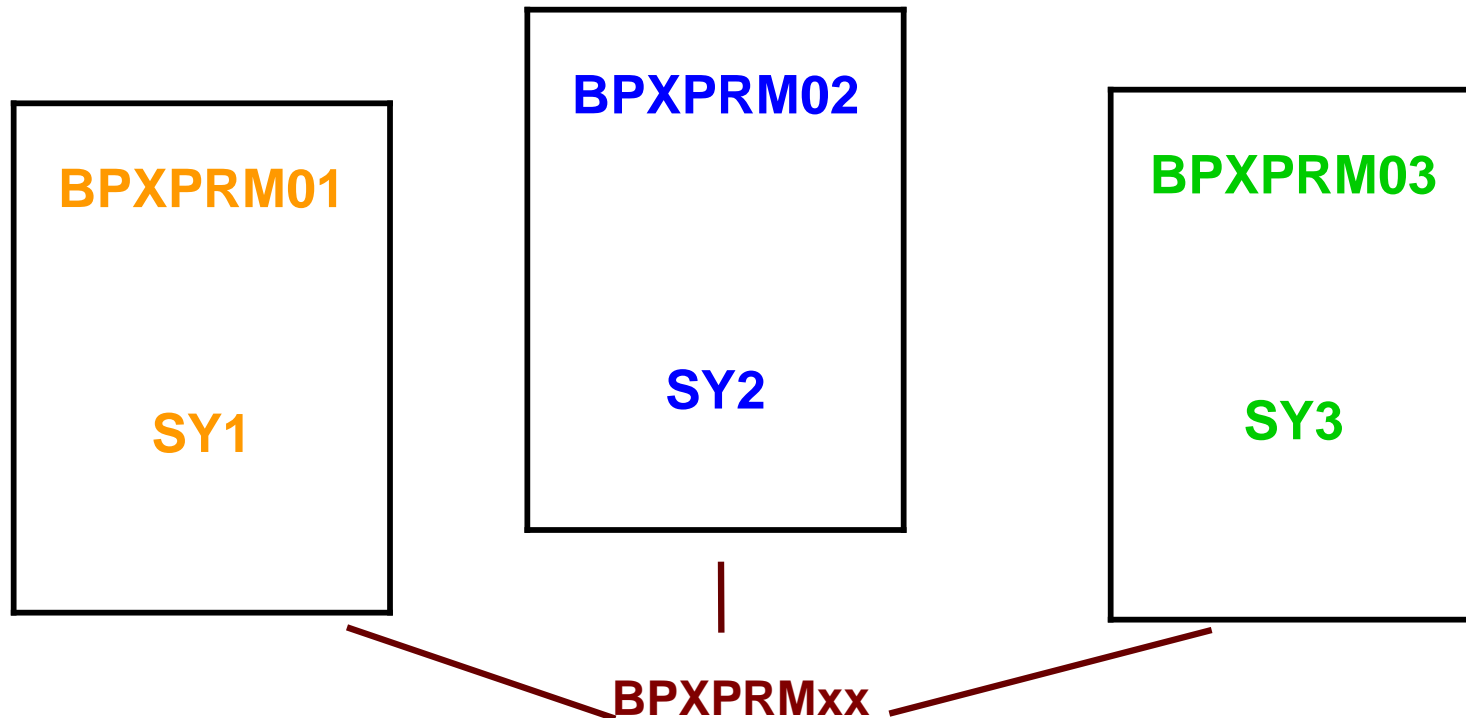
Performance numbers

zFS queries (zFS owner, sysplex aware, cache information)

# Shared Filesystem Advantages

- **Greater user mobility**
- **Flexibility with data placement**
- **Can read and write to filesystems from all systems in the shared filesystem group.**
- **Greater availability of data in the event of a system outage**
- **Common BPXPRMxx for all systems**
- **Common file system tree on all systems**

## Shared File System – Key Concepts



- All systems may share a common BPXPRMxx member that contains a ROOT statement and MOUNT statements. This is done through the use of system symbolics.
- The common BPXPRMxx member also contains the SYSPLEX(YES) statement.
- Each system may specify a BPXPRMxx member that contains system specific limits.
- All systems should have the same PFSs and any necessary subsystems required by the PFS started. For example, NFS client PFS requires the TCP/IP subsystem be started and a network connection configured.

# Overview – z/OS UNIX Sysplex File Sharing



## BPXPRMxx member parameters that support sysplex file sharing:

- **SYSPLEX(Yes)**
  - Default is No
  
- **VERSION('nnn')**
  - Allows multiple releases and service levels of the binaries to coexist and participate in a shared file system.
  - IBM recommends using 'nnn' as a symbolic name.

**Example:      VERSION(' &RLSE' )   or   VERSION(' &SYSR1' )**

- **SYSNAME(sysname) parameter on the MOUNT statements**
  - Specifies that 'sysname' be the z/OS UNIX file system owner.
  
- **AUTOMOVE parameter on the MOUNT statements**
  - Specifies z/OS UNIX action on the file system in the event of a system outage.



# Overview – z/OS UNIX Sysplex File Sharing



## Details on the automove parameter on mount statements:

- **AUTOMOVE=YES** allows the system to automatically move logical ownership of the file system as needed. AUTOMOVE=YES is the default; you can specify it as AUTOMOVE.
- **AUTOMOVE=NO** prevents ownership movement in some situations.
- **AUTOMOVE=UNMOUNT** unmounts the file system in some situations.
- **AUTOMOVE=indicator(sysname1,sysname2,...,sysnameN)** specifies a list of systems to which the ownership of file system should or should not be moved when ownership of the file system changes.
  - If indicator is specified as INCLUDE (or I), the list must provide a comma-delimited, priority-ordered list of systems to which ownership of the file system can be moved. For example, AUTOMOVE=INCLUDE(SYS1, SYS4, SYS9). You can specify an asterisk (\*) as the last (or the only) system name to indicate any active system. For example, AUTOMOVE=INCLUDE(SYS1, SYS4, \*).
  - If indicator is specified as EXCLUDE (or E), the system list must provide a comma-delimited list of systems to which the ownership of file system must not be moved. For example, AUTOMOVE=EXCLUDE(SYS3, SYS5, SYS7).



# Overview – z/OS UNIX Sysplex File Sharing



## Recommendations on the automove parameter on mount statements:

- **AUTOMOVE=YES** Should be specified (or defaulted) on most filesystems. Version and sysplex root file system should be automove.
- **AUTOMOVE=NO** Should be specified on system specific filesystems. File system will be in an “unowned” state until the owning system is re-IPLed back into the sysplex.
- **AUTOMOVE=UNMOUNT** Should be specified on system specific filesystems.
- **AUTOMOVE=indicator(sysname1,sysname2,...,sysnameN)** Useful if there are a subset of systems where you prefer the z/OS UNIX owning system.

**Note:** With zFS sysplex sharing the z/OS UNIX owning system has less significance. The AUTOMOVE syslist is used for z/OS UNIX file system ownership and not for zFS ownership. If you use RWSHARE, note that zFS ownership could move to a system not in your include list.



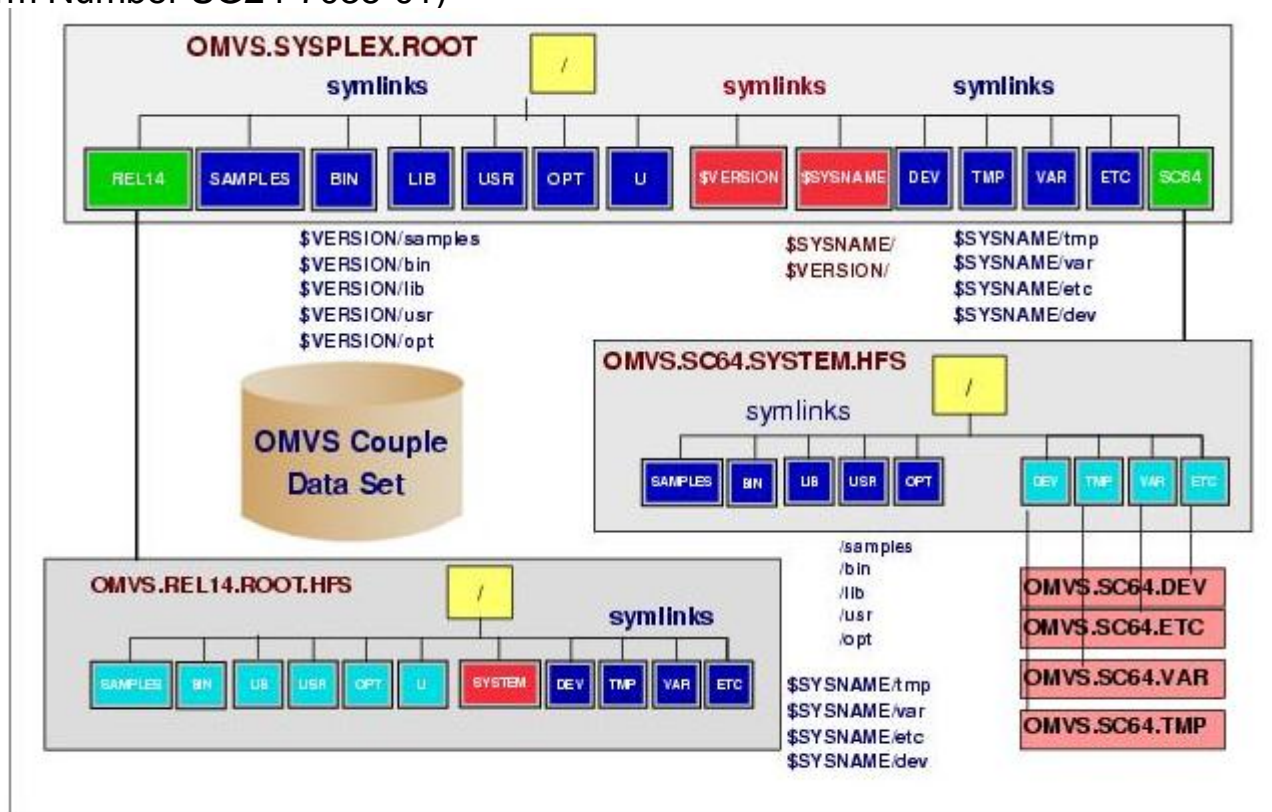
# File system structures in a shared configuration

- **Sysplex root file system**
  - The sysplex root is a file system that is used as the sysplex-wide root. Only one sysplex root is allowed for all systems participating in a shared FS environment.
- **System-specific file system**
  - Directories in the system-specific file system data set are used as mount points, specifically for /etc, /var, /tmp, and /dev.
- **Version file system**
  - You can use one version file system for each set of systems participating in shared FS and that are at the same release level.



# Hierarchical file system concepts

- Figure 6-27 All the z/OS UNIX file sharing structures used in a sysplex sharing environment. Source: Redbook: UNIX System Services z/OS Version 1 Release 7 Implementation (ISBN 073849609X - IBM Form Number SG24-7035-01)



## Shared File System – Key Concepts

- All participating systems communicate with each other using coupling facility or channel-to-channel connections.
- BPXMCDs couple data set is a complex record of participating systems and the file systems in use throughout the sharing group.

SY1 RECORD

SY2 RECORD

SY3 RECORD

SY4 RECORD

### OMVS Couple Data Set System Record

- Contains information about each system in the shared FS group.
- Changes as systems join or leave the sysplex.

# How to see the system information that is in the CDS:



MODIFY BPXOINIT,FILESYS=DISPLAY

BPXF242I 2013/02/08 08.34.28 MODIFY BPXOINIT,FILESYS=DISPLAY,GLOBAL

SYSTEM	LFS	VERSION	---	STATUS-----	RECOMMENDED ACTION
--------	-----	---------	-----	-------------	--------------------

NP5	1.	13.	0	VERIFIED	NONE
-----	----	-----	---	----------	------

NP6	1.	13.	0	VERIFIED	NONE
-----	----	-----	---	----------	------

CDS VERSION= 2 MIN LFS VERSION= 1. 13. 0

DEVICE NUMBER OF LAST MOUNT= 372

MAXIMUM MOUNT ENTRIES= 4000 MOUNT ENTRIES IN USE= 356

MAXIMUM AMTRULES= 300 AMTRULES IN USE= 8

MAXSYSTEM= 12

ALTROOT= USSZFS.ALTROOT.ZFS

## SYSTEMS PERFORMING UNMOUNT

(Since 2013/02/04 00.35.11)

NUMBER OF UNMOUNTS IN PROGRESS= 1

NP6

## ACTIVE QUEUE

UNMOUNT SHARED

# Shared File System – Key Concepts, continued

**PLEX.SYSPLEX.ROOT**

**PLEX.V1R13.VERFS**

**PLEX.V2R1.VERFS**

**PLEX.SY1.FS**

**PLEX.SY2.FS**

**PLEX.SY3.FS**

**PLEX.SY4.FS**

**USSZFS.TOTTEN.FS1**

## **OMVS Couple Data Set Mount Record**

- Used to keep hierarchy consistent across the sharing group.

- Provides information about each file system in use throughout the shared file system group.

- Changes as file systems are mounted, unmounted, moved, recovered, remounted or

**become unowned.**

## Fail-safe the sysplex root file system

- In a sysplex configuration, the *alternate sysplex root file system* is a hot standby for the sysplex root file system that is used to replace the current sysplex root file system when the sysplex root file system becomes unowned.
  - The alternate sysplex root file system is established by using the ALTROOT statement in the BPXPRMxx parmlib member during OMVS initialization or by using the SET OMVS command.

# Steps for setting up the alternate sysplex root

1. Allocate a new file system to be used as the alternate sysplex root file system.
2. On the alternate sysplex root, set up the mount points and the symbolic links. The mount points and the symbolic links must be same as the ones on the current sysplex root.
3. Specify **ALTROOT** in the BPXPRMxx parmlib member with the mount point in the root directory of the current sysplex root file system.

**Restriction:** The ALTROOT mount point must not exceed 64 characters in length.

## Example:

```
ALTROOT FILESYSTEM('USSZFS.ALTROOT.ZFS')  
      MOUNTPOINT('/mnt')
```

4. Make sure that all systems in the shared file system environment have direct access to the new file system and can locally mount it.
5. Process the **ALTROOT** statement by using the SET OMVS command or by initializing the OMVS with the updated BPXPRMxx parmlib member.

## Example:

```
SET OMVS=(xx)
```

# Steps for removing the alternate sysplex root

1. In the BPXPRMxx parmlib member, replace the ALTROOT FILESYSTEM statement with the following statement:

**ALTROOT NONE**

Because the ALTROOT NONE and ALTROOT FILESYSTEM statements are mutually exclusive, only one can be specified in the BPXPRMxx parmlib member.

**Note:** If concatenating parmlib members result in multiple ALTROOT statements, then the first parmlib member specified on the OMVS= operator command that contains the ALTROOT statement will take effect.

2. Issue a SET OMVS operator command to process the ALTROOT NONE statement.

**Example:**

SET OMVS=(XX)

# Steps for dynamically replacing the sysplex root

1. To verify that the sysplex root is locally mounted on all systems, issue:

```
Route *all, D OMVS,F,NAME=root_file_system_name
```

You should see **CLIENT=N** for each system.

2. Allocate a new file system to be used as the new sysplex root file system.

**Rules: see notes**

3. On the new sysplex root, set up the mount points and the symbolic links. The mount points and the symbolic links must be the same as the ones on the current sysplex root.

4. On any system in the shared file system configuration, issue:

```
F OMVS,NEWROOT=new.root.file.system.name,COND=<YES | NO | FORCE>
```

**YES** Proceed conditionally.

**NO** Proceed unconditionally.

**FORCE** This option allows user to replace a failing sysplex root with the user-specified new sysplex root.

5. Update the name and type parameter (if appropriate) of the sysplex root file system in the BPXPRMxx member.



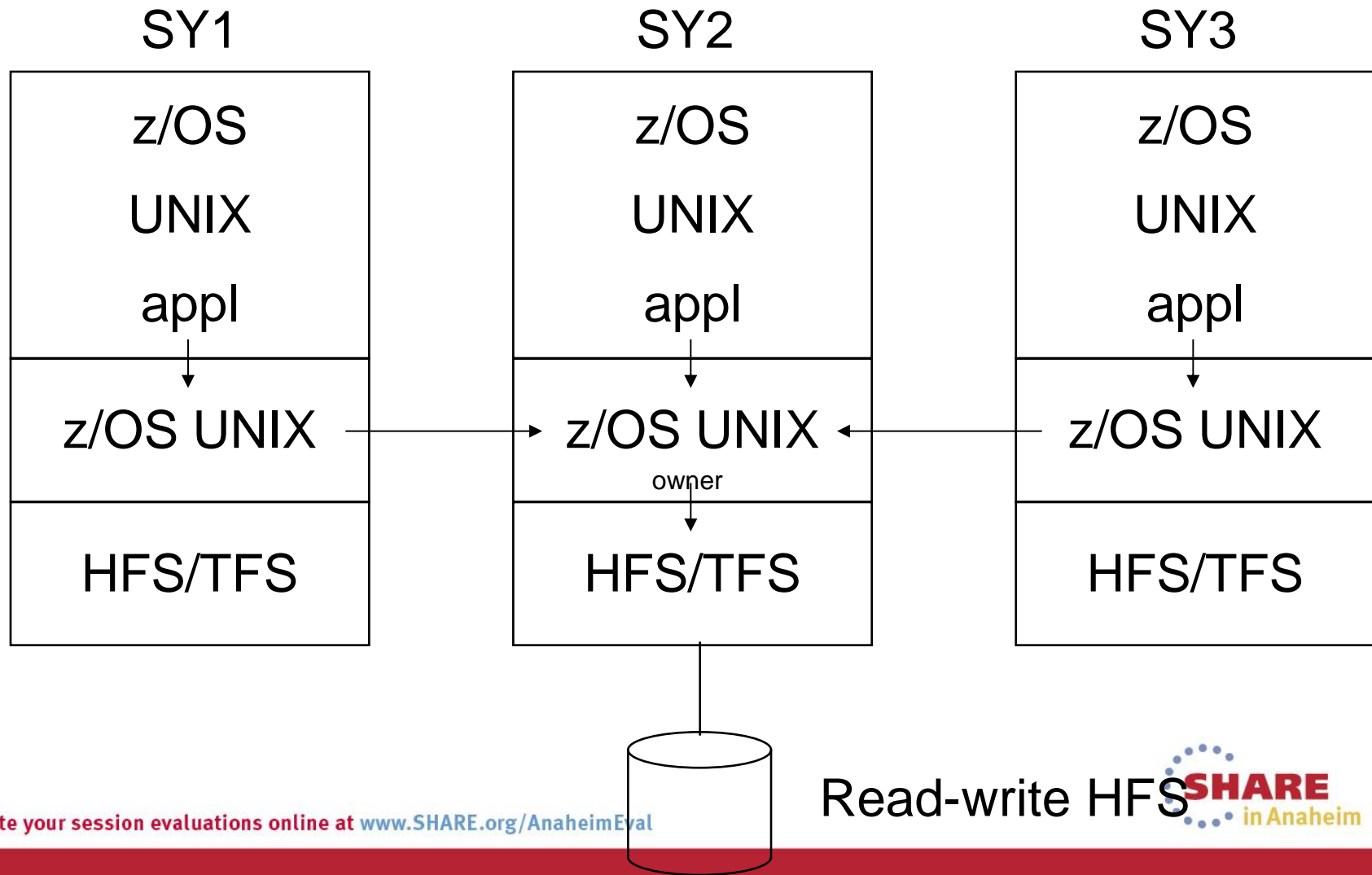
## Shared file system terminology

- **Non-Sysplex aware file system (sysplex-unaware)**
  - The file system requires it to be accessed through the remote owning system from all other systems in the shared file system configuration. There is only one connection for update at a time for a particular mount mode (read-only or read-write).
- **Function Shipping**
  - Function shipping means that a request is forwarded to the owning system and the response is returned back to the requestor through XCF communications.
- **Sysplex-aware file system**
  - The file system is locally mounted on every system and file requests are handled by the local PFS.

## Shared file system terminology (cont'd)

- **Local connection** - A mount directly to the native PFS
- **Owner** - System assigned as the mount coordinator for a particular file system, and in some cases, the system coordinating I/O for that file system.
- **Server** - An owner that has one or more function shipping client
- **Client** - A system other than the owner, which is sharing a file system through communication with the owner (function shipping)

## Non-sysplex aware file systems: z/OS UNIX does the function shipping of i/o requests



# Overview I – zFS R/W Sysplex Sharing



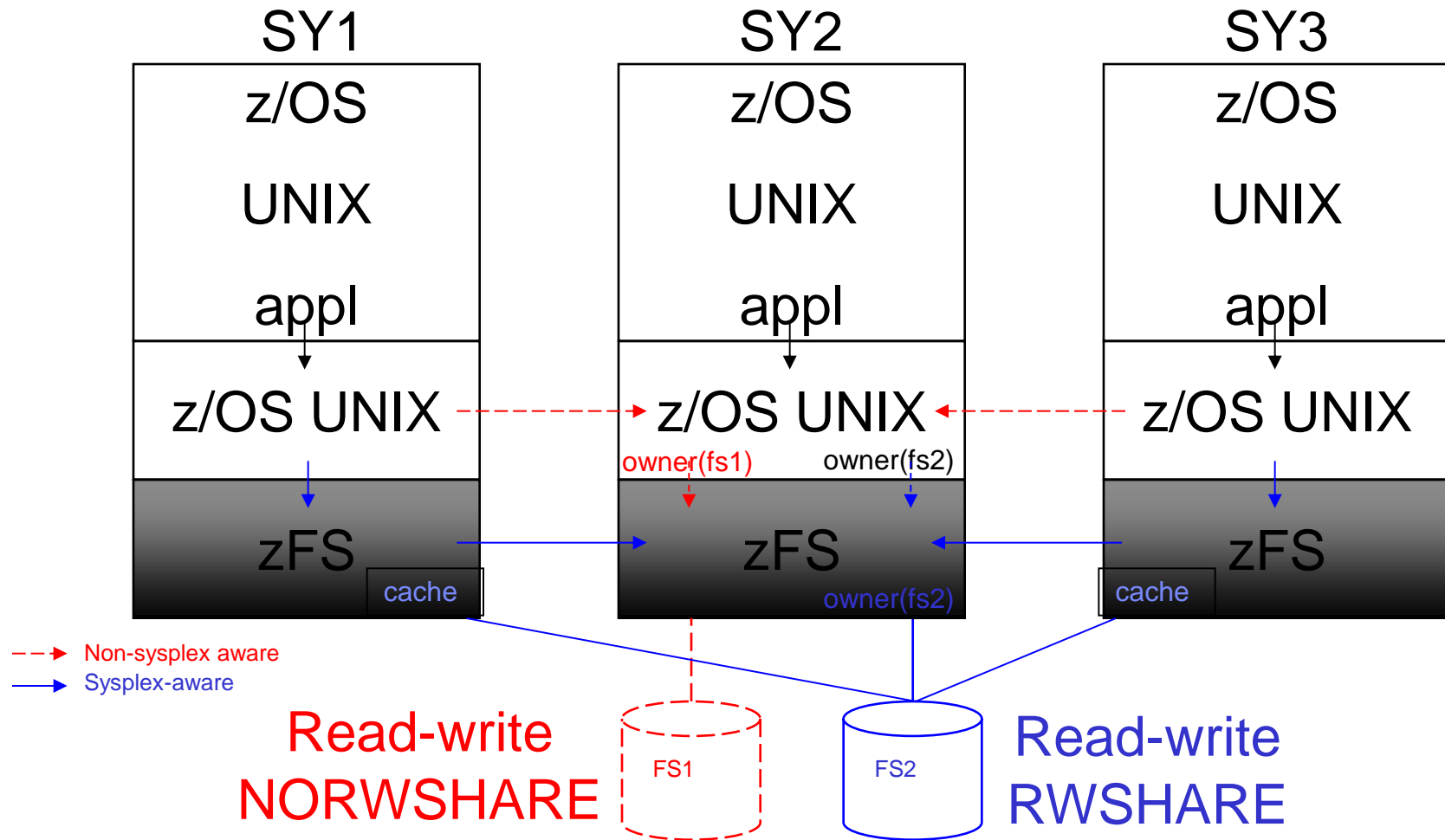
**zFS provides optional support for file systems mounted R/W in a parallel sysplex in a z/OS UNIX Shared File System Environment**

- zFS manages sysplex serialization and updates
- zFS manages system recovery for the file system
- zFS still has an owner that manages metadata and administration operations
- **RWSHARE** file system – This term denotes a R/W mounted file system that is using the optional zFS sysplex sharing support.
- **NORWSHARE** file system – This term denotes a R/W mounted file system that is not using the zFS sysplex sharing support.
- We support a mix of NORWSHARE and RWSHARE mounted file systems.
- zFS moves ownership dynamically if one system has a “lions-share” of the usage to that system for optimal performance.
- **RWSHARE significantly improves client performance because:**
  - zFS caches file and directory contents at non-owners
  - zFS performs file read-ahead and write-behind for large file access.
  - zFS can directly read/write data to disk from non-owners

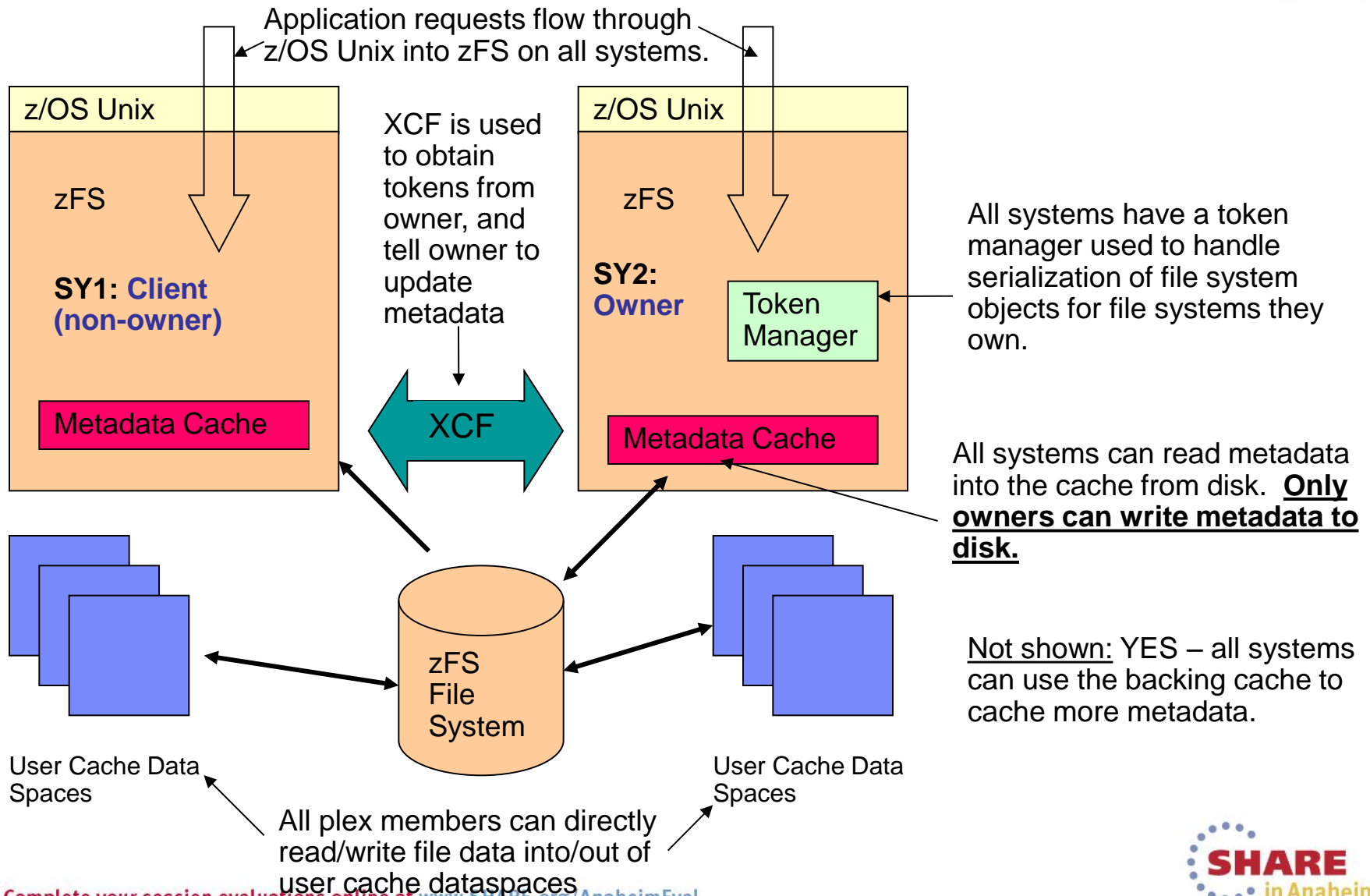
**Note:** The zFS presentation material here is at the z/OS 13 and later software levels.



# Overview II – RWSHARE and NORWSHARE File Systems

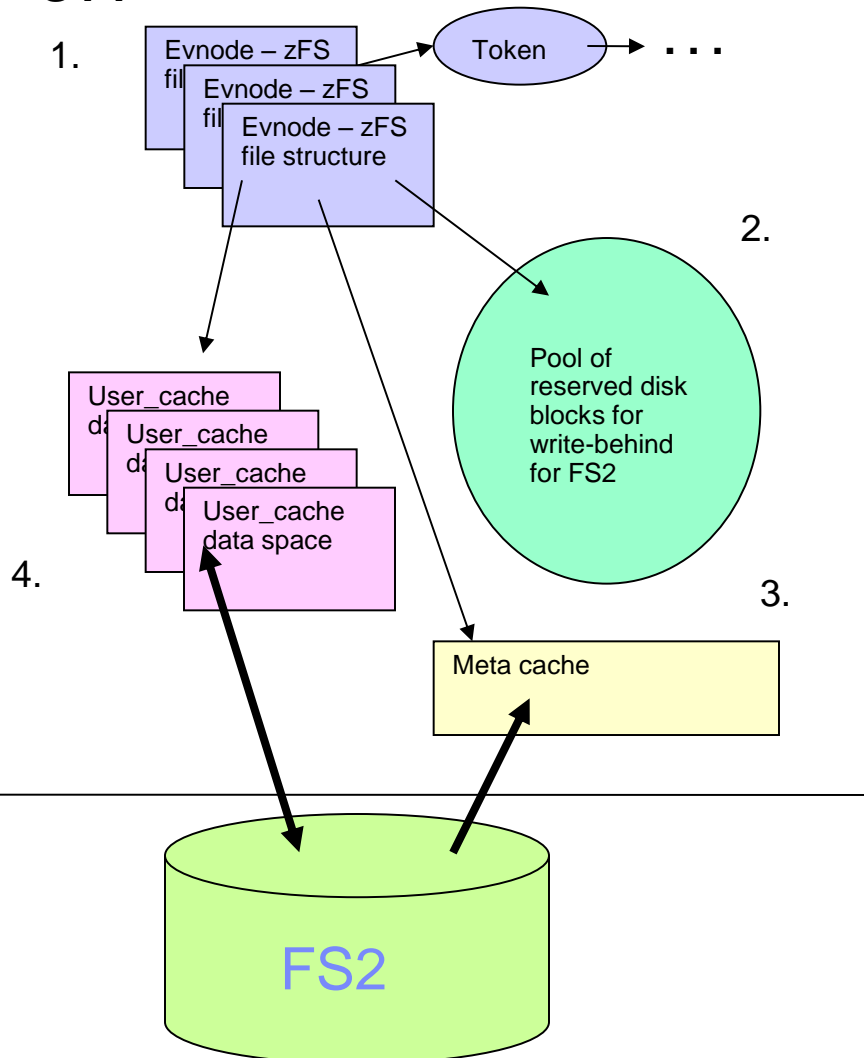


# Overview III – RWSHARE File System Design



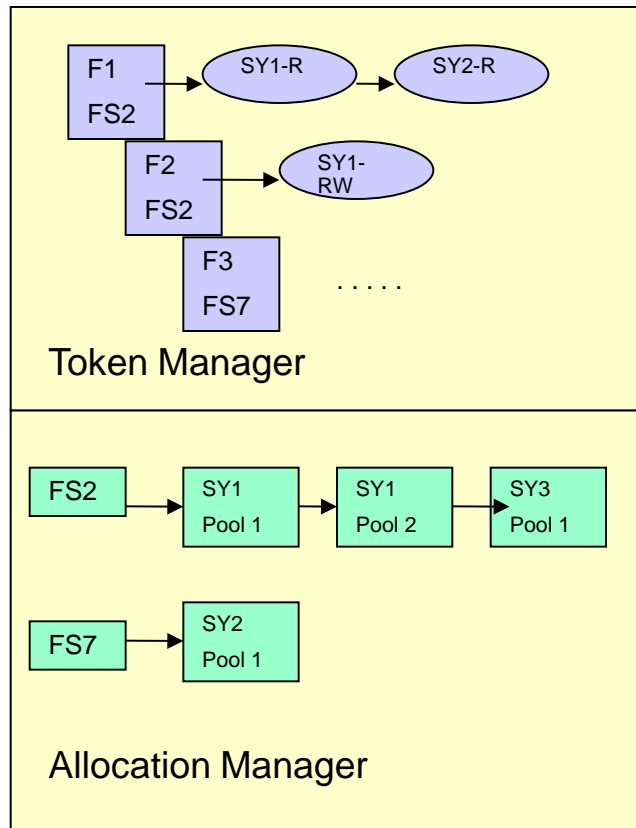
# Overview IV – zFS RWSHARE Sysplex Caching – Non Owner

## SY1



1. Clients obtain tokens from owners to allow for caching of object data via XCF communications to owner  
→ **vnode\_cache\_size** determines number of objects clients can cache.
2. If writing files, clients obtain pools of disk blocks from the owner via XCF and assign these blocks to newly written files **to allow for direct writes to disk.**  
→ Owner can always callback to reclaim unassigned blocks if disk low-on-space.
3. Clients can directly read directory contents and file metadata from disk into its metadata (and backing) cache.  
→ Will synchronize with owner if necessary.
4. Clients can directly read and write file data to/from disk from its user file cache data spaces for non-thrashing objects.

# Overview V – zFS RWSHARE Sysplex Caching - Owner



Note: Only owners directly write metadata to disk.

## Token Manager

- Owner component manages RWSHARE sysplex locking
- Tracks which systems have tokens for which files in RWSHARE file systems.
- Owner will callback to clients (revoke) if a system requests a conflicting access to an object.
- token\_cache\_size** = size of token cache managed by owners
  - Owners will garbage collect oldest tokens from clients if low on tokens in its token manager.
- Default token cache size =  $2 \times \mathbf{vnode\_cache\_size}$ .

## Allocation Manager

- Owner component for BOTH NORWSHARE and RWSHARE file systems.
- Tracks pools of blocks given to systems to support write-behind.
- If NORWSHARE, only local system has block pools
- For RWSHARE, any system could have one or more block pools.
- If low on space, allocation manager has to callback systems to get unassigned blocks back from pools.



# Overview VI – Thrash Resolution



- **What if multiple systems write same object concurrently?**
  - Prior slides show how zFS clients cache data, and perform read-ahead and write-behind after first obtaining sysplex tokens (locks) on object.
    - Excessive lock-callbacks and data sync could occur if more than one system accessing object and at least one system writing to object.
  - **zFS detects thrashing and resolves it by:**
    - Using a different protocol for that object, this protocol is much like z/OS Unix Sharing:
      - **Does not use data locks on the object**
      - **Forwards any reads/writes of the object to the server.**
      - **Still caches security information at clients for proper security checking.**
      - **Still manages a name-lookup cache for thrashing directories to reduce lookup XCF calls to owners.** Owners will callback clients if an update occurs to a name in a directory being cached by the client.
  - zFS detects when thrashing has stopped for object, and then resumes its normal protocol (aggressive caching/read-ahead/write-behind)



# RWSHARE Performance I - Workloads



- **File Workload Descriptions:**

- The file results (next slide) were obtained using 2 workloads created by IBM:
  - **FSPT** – Parallel database access simulation. Multiple tasks read/write same large file in parallel. The file is already pre-allocated on disk, no new blocks written.
  - **pctestFS** – Parallel creation/write/delete of new files to multiple file systems. This tested allocation of new files and allocation and assignment of blocks to those files along with the actual file IO to/from disk.
- Tests were run with a severely constrained user file cache (near zero hit ratio) and with a very large user file cache (near 100% hit ratio) to show the affects of each situation.
  - It is expected that most customers will achieve hit ratios from 70-95% in practice.

- **Directory Workload Descriptions:**

- The directory results (subsequent slide) were obtained using 2 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).
  - **pctestDL** – The tasks did repeated lookup and readdir functions.
  - **pctestDU** – The tasks performed directory create/update/remove/readdir/search.
- Tests were run varying the sizes of the involved directories to test scale-ability.

- **Definitions:**

- **External Throughput (E)** – Number of operations per unit of time.
  - The higher the number, the lower the average operation response time.
- **Internal Throughput (I)** – Number of operations per unit of processor time.
  - The higher the number, the less CPU time each operation took.
  - The subsequent slides show ITR per processor.



# RWSHARE Performance II – Sysplex Client File



- Performance for RWSHARE sysplex client, z9 / FICON connected DASD

	FSPT			ptestFS		
Cache Hit Ratios	R13 <b>NORWSHARE</b> MB / Second	z/OS 13 <b>RWSHARE</b> Ratio over R13 NORW	z/OS 2.1 V4 <b>RWSHARE</b> Ratio over R13 NORW	R13 <b>NORWSHARE</b> MB / Second,	z/OS 13 <b>RWSHARE</b> Ratio over R13 NORW	z/OS 2.1 V4 <b>RWSHARE</b> Ratio over R13 NORW
<b>0% cached</b> User_cache=10M	E=16.56 I=11.51	<b>EΔ 0.993</b> <b>IΔ 2.388</b>	<b>EΔ 0.986</b> <b>IΔ 2.25</b>	E=13039 I=5603	<b>EΔ 11.069</b> <b>IΔ 7.761</b>	<b>EΔ 10.592</b> <b>IΔ 7.418</b>
<b>100% cached</b> User_cache=1280M	E=178.94 I=26.95	<b>EΔ 4.886</b> <b>IΔ 8.031</b>	<b>EΔ 4.652</b> <b>IΔ 7.646</b>	E=13125 I=5663	<b>EΔ 14.695</b> <b>IΔ 8.109</b>	<b>EΔ 13.893</b> <b>IΔ 7.616</b>

- FSPT – (2 reads per one write, so cache hit ratio very important to performance).
  - 0% Cached – ETR the same due to DASD IO bottlenecked, BUT notice that ITR still improved, this means less sysplex processor time per operation. This is due to fact that RWSHARE does almost no communication with owner system.
  - 100% Cached – ETRs and ITRs significantly improved, much better response time and much less time on sysplex processors. **Close to 5X throughput improvement.**
- ptestFS – (file writing workload, so caching hit ratio not as significant)
  - 0%/100% Cached – Both cases improve significantly as very little sysplex communication occurs between client and owner, even in low cache case. **Over 14X throughput improvement.**



# RWSHARE Performance III – Sysplex Client Directory



- Performance for RWSHARE sysplex client, z9 / FICON connected DASD

	ptestDL			ptestDU		
Directory Sizes	R13 NORWSHARE Operations / Second	z/OS 2.1 V4 RWSHARE Ratio over R13 NORW	z/OS 2.1 V5 RWSHARE Ratio over R13 NORW	R13 NORWSHARE Operations / Second	z/OS 2.1 V4 RWSHARE Ratio over R13 NORW	z/OS 2.1 V5 RWSHARE Ratio over R13 NORW
<b>0 Base Names</b> (2000 names per directory)	E=17106 I=3380	E△ 13.459 I△ 16.103	E△ 18.395 I△ 21.676	E=25895 I=5215	E△ 2.678 I△ 2.249	E△ 3.176 I△ 2.604
<b>18k Base Names</b> (20000 names per directory)	E=2306 I=521	E△ 14.582 I△ 10.825	E△ 88.110 I△ 63.468	E=3599 I=849	E△ 7.295 I△ 5.483	E△ 18.485 I△ 13.458
<b>48k Base Names</b> (50000 names per directory)	E=888 I=206	E△ 15.347 I△ 10.471	E△ 119.64 I△ 82.311	E=2548 I=559	E△ 5.691 I△ 4.608	E△ 21.209 I△ 17.562

- z/OS 2.1 greatly improved readdir (**ls** and **ls -l**) operations for sysplex client
- And z/OS 2.1 provides new v5 file system for good directory scale-ability
- Using RWSHARE and V5 file systems yields a very large performance gain in directory operations over V4 file systems, and especially V4 NORWSHARE.**
- vnode\_cache\_size=400000, meta\_cache\_size=100M, metaback\_cache\_size=2G**



# Enabling RWSHARE



- **Set IOEFSPRM/Parmlib share mode default:**
  - **SYSPLEX\_FILESYS\_SHAREMODE=XXXXX**, where **XXXXX** is either:
    - **NORWSHARE** – if you want the default RW mounted file system to NOT use zFS RW sysplex file sharing. *Note that this is default value.*
      - In this case, use the **RWSHARE MOUNT** parameter to indicate any file system you would like to exploit zFS RW sysplex sharing.
    - **RWSHARE** – if you want the default RW mounted file system to use zFS RW sysplex file sharing.
      - In this case, use the **NORWSHARE MOUNT** parameter for any file system that you would not like to use zFS RW sysplex sharing.
- **RWSHARE optimal for multi-member access to the same file system:**
  - RWSHARE does involve more zFS memory usage (tokens), and zFS is constrained below the bar.
  - Useful only for file systems accessed by multiple sysplex members.
  - **Best to selectively choose the best candidate file systems for RWSHARE usage (highest usage file systems accessed by more than one plex member at a time)**
    - → <ftp://public.dhe.ibm.com/s390/zos/tools/wjfsmon/wjfsmon.pdf> - this tool will show which R/W mounted file systems are accessed by more than one sysplex member



# RWSHARE Usage Notes



- **Products that do not support RWSHARE:**
  - z/OS SMB Server
  - Fast Response Cache Accelerator support of the IBM HTTP Server for z/OS V5.3
  - Any Product that uses Register File Interest API (unlikely there are many of these)
    - **Recommendation** → Use NORWSHARE file systems exclusively, or use RWSHARE only for file systems that are not accessed by these products if these products are used at your site.
- **System Specific File Systems**
  - These file systems should be mounted with the **AUTOMOVE UNMOUNT** or **NOAUTOMOVE**, the file system will be unmounted if the system goes down, or zFS would move ownership back to the system when it restarts due to zFS performance based aggregate movement. Should also be **NORWSHARE** since most access is from the owner.
- **Remount of R/O File System to R/W Mode**
  - Will use zFS sysplex file sharing if the default **SYSPLEX\_FILESYS\_SHAREMODE=RWSHARE** or **RWSHARE** was specified in the file system MOUNT parameter at the time of the initial R/O mount; otherwise, will not use zFS sysplex file sharing and therefore use the z/OS Unix file sharing protocol.



# RWSHARE Error Handling



- **System Outages**
  - zFS on the remaining members assume ownership of file systems owned by the down-system, much like the z/OS Unix shared file system support does. Note, however, zFS does not use the automove syslist.
- **Communication Failures and Timeouts**
  - Uses a timeout mechanism to prevent hang-ups in the sysplex if a member is having problems or is delayed somehow.
  - zFS handles the case of lost transmissions and replies, keeping the file system available (or as much as possible) and consistent.
  - Very tolerant of another zFS member taking too long to process a request, and can repair sysplex state between zFS members once the problem is resolved keeping consistency between sysplex members.
  - Provides **F ZFS,NSV** command to force a consistency check between sysplex member's file system state and corrects any problems automatically.
- **Informative Messages Provided**
  - zFS provides messages to indicate if a request is taking too long on another system, if zFS or a system takes an outage, or if its repairing sysplex state between members.
- **Severe Software Errors**
  - Global severe errors cause zFS restart, preserving mount-tree and is relatively fast.
  - File System specific severe errors temporarily disable the file system for access plex-wide, zFS will dynamically move ownership to another system to clean up file system state (**RWSHARE**) or internally remount (**NORWSHARE**) and resume user activity.





# Tuning zFS RWSHARE



- **Tune `user_cache_size/meta_cache_size/metaback_cache_size`**
  - Tuning zFS sysplex support is much the same as tuning zFS in any other situation. The primary objects to tune are the size of the caches used to hold file data (**`user_cache_size`**) and metadata (**`meta_cache_size`** and **`metaback_cache_size`**).
  - zFS is constrained below the bar, must be careful about specifying too large a `meta_cache_size` since that is below the bar. The other caches are in data spaces.
  - **Note:** The default for these caches has been changed in 2.1.
- **The `F ZFS,QUERY,STORAGE` command should be used to monitor zFS below-bar storage when adjusting all zFS cache sizes.**
- **RWSHARE specific tuning:**
  - **`vnode_cache_size`** — This variable determines how many files/directories zFS will cache in its memory.
    - It is especially important for RWSHARE sysplex clients because if the memory object does not exist for a file in memory, it has to contact the server to obtain information and a sysplex token for the file.
    - For an owner this is a smaller impact since it can simply obtain the file information from its metadata cache and it owns the token manager so there is no IO involved in most cases.
  - **`token_cache_size`** — This variable determines how many tokens are used for file systems owned by the local system. Increase if you see frequent token garbage collection.

See *Distributed File Services zFS Administration* for details





# Monitoring zFS RWSHARE I

- **F ZFS,QUERY,LEVEL** – shows **SYSPLEX\_FILESYS\_SHAREMODE**

```
19.08.31 DCEIMGHQ STC00005 IOEZ00639I zFS kernel: z/OS      zFS
Version 02.01.00 Service Level 0000000 - HZFS410.
Created on Wed Mar 20 16:05:20 EDT 2013.
sysplex(filesys,rwshare) interface(4)
```

- **zfsadm lsaggr** – shows zFS owner

OMVS.ZFS.REL19.DR15	DCEDFBLD	R/O
OMVS.ZFS.REL19.DR16	DCEDFBLD	R/O
OMVS.ZFS.PROJ.DFS	DCEDFBLD	R/W

- **zfsadm aggrinfo -long** – indicates if file system is **RWSHARE** (will say **sysplex-aware**)

```
ZFSAGGR.BIGZFS.FS1 (R/W COMP): 1350847 K free out of total 2000160
version 1.4
auditfid E9C6E2F1 F0F500FE 0000
sysplex-aware
      168855 free 8k blocks;          7 free 1K fragments
      32800 K log file;              56 K filesystem table
      288 K bitmap file
```

# Monitoring zFS RWSHARE II

- **F ZFS,QUERY,FILE** – Shows local activity to file systems and also if **RWSHARE**
  - Issue on each plex member to show which systems have local activity to file system (validates if RWSHARE applicable)

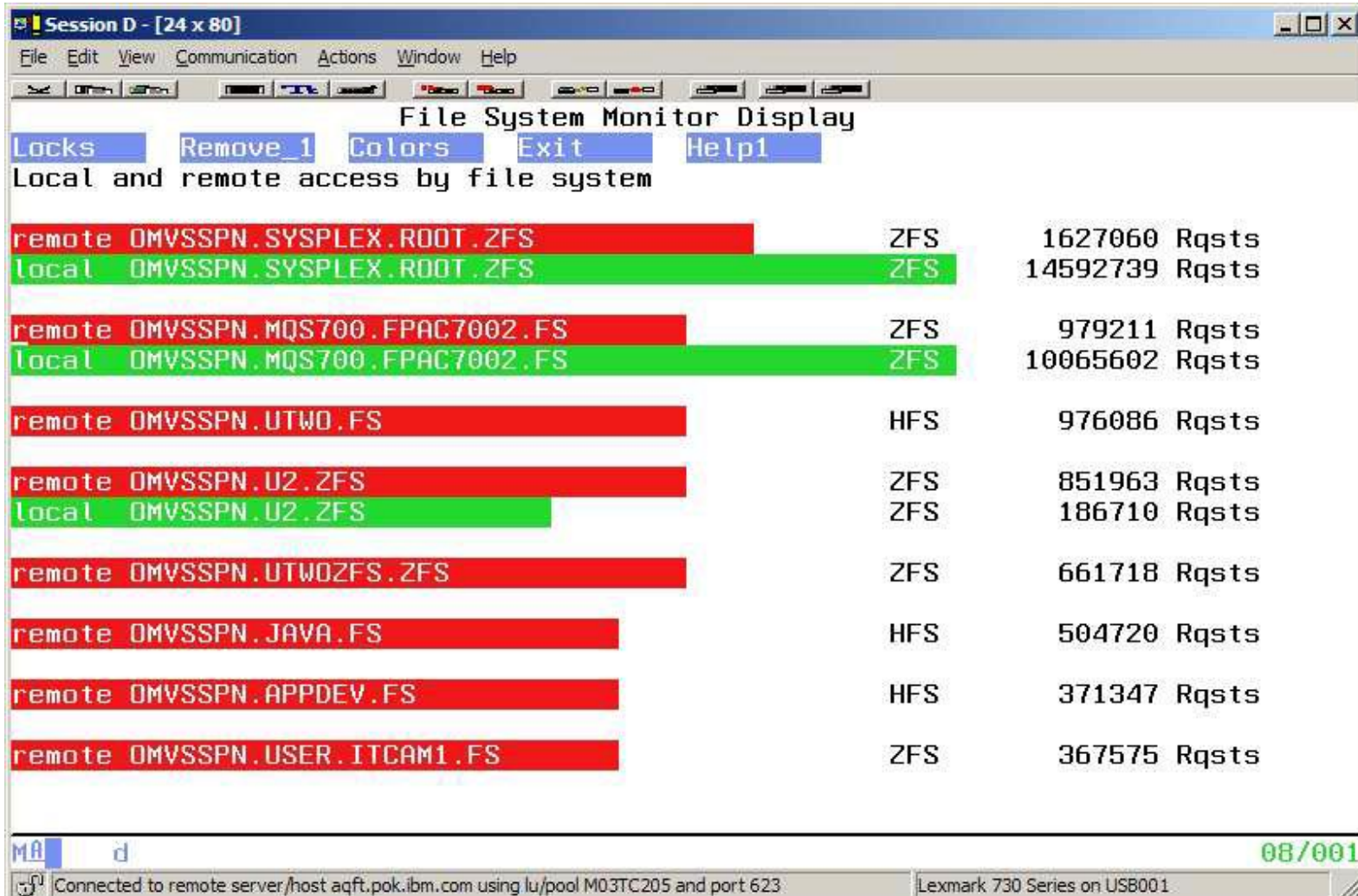
File System Name	Aggr #	Flg	Operations
ZFSAGGR.BIGZFS.FS1	1	AMSL	2198

- **df - v** – Shows file system from z/OS UNIX view from this system:
  - **Client=N** – z/OS UNIX passing requests to local zFS (**RWSHARE** or R/O file system)
  - **Client=Y** – z/OS UNIX handling sysplex, hence **NORWSHARE** function.

Mounted on	Filesystem	Avail/Total	Files	Status
/home/suimghq/lfsmounts/PLEX.ZFS.FILESYS	(ZFSAGGR.BIGZFS.FS1)			
2701694/40003				
20 4294962725	Available			
ZFS, Read/Write, Device:31, ACLS=Y				
File System Owner : DCEIMGH0		Automove=Y	Client=N	
Filetag : T=off codeset=0				
Aggregate Name : ZFSAGGR.BIGZFS.FS1				

# Monitoring zFS RWSHARE III

- **Sample wjfsmon output** – remote requests and R/W mounted = Good **RWSHARE** candidate



File System Monitor Display

Local and remote access by file system

remote	OMVSSPN.SYSPLEX.ROOT.ZFS	ZFS	1627060 Rqsts
local	OMVSSPN.SYSPLEX.ROOT.ZFS	ZFS	14592739 Rqsts
remote	OMVSSPN.MQS700.FPAC7002.FS	ZFS	979211 Rqsts
local	OMVSSPN.MQS700.FPAC7002.FS	ZFS	10065602 Rqsts
remote	OMVSSPN.UTWO.FS	HFS	976086 Rqsts
remote	OMVSSPN.U2.ZFS	ZFS	851963 Rqsts
local	OMVSSPN.U2.ZFS	ZFS	186710 Rqsts
remote	OMVSSPN.UTWOZFS.ZFS	ZFS	661718 Rqsts
remote	OMVSSPN.JAVA.FS	HFS	504720 Rqsts
remote	OMVSSPN.APPDEV.FS	HFS	371347 Rqsts
remote	OMVSSPN.USER.ITCAM1.FS	ZFS	367575 Rqsts

08/001

Connected to remote server/host aqft.pok.ibm.com using lu/pool M03TC205 and port 623 Lexmark 730 Series on USB001

# Sysplex Statistics I: F ZFS,QUERY,KNPFS - Sysplex Client Summary



## PFS Calls on Client

Operation	Count	XCF req.	Avg Time
zfs_opens	885098	0	0.020
zfs_closes	885110	0	0.010
zfs_reads	12079	0	0.157
zfs_writes	0	0	0.000
zfs_ioctls	0	0	0.000
zfs_getattr	2450523	8	0.009
zfs_setattr	313031	656	0.020
zfs_accesses	11495	0	0.018
<b>zfs_lookups</b>	<b>13764811</b>	<b>1190897</b>	<b>0.287</b>
zfs_creates	876507	876556	5.625
zfs_removes	1240556	1240621	2.117
zfs_links	157216	157216	2.567
zfs_renames	155164	155165	1.890
zfs_mkdirs	157971	157971	6.031
<b>zfs_rmdirs</b>	<b>155108</b>	<b>155109</b>	<b>2.164</b>
zfs_readdir	11322	3053	11.345
zfs_symlinks	157398	157398	4.295

zfs_readlinks	68	58	0.871
zfs_fsyzcs	0	0	0.000
zfs_truncs	0	0	0.000
zfs_lockctls	0	0	0.000
zfs_audits	33	0	0.015
zfs_inactives	2698174	0	0.020
zfs_recoveries	0	0	0.000
zfs_vgets	0	0	0.000
zfs_pfsccts	0	0	0.000
zfs_statfss	0	0	0.000
zfs_mounts	0	0	0.000
zfs_unmounts	0	0	0.000
zfs_vinacts	0	0	0.000
*TOTALS*	23931664	4094708	0.602

- Shows the number of operations that required one or more calls to the owner of the file system. Lookup requests had over 1 million XCF calls, likely to get token for a vnode not found in cache. Could make **vnode\_cache\_size** larger if memory permits to try and reduce these.
- But due to client caching, over 12 million lookup requests satisfied by client metadata/vnode cache. (Difference between Count & XCF req.)
- Directory update operations are sent synchronously to server.



# Sysplex Statistics II: F ZFS,QUERY,STKM – Token manager statistics



## Server Token Manager (STKM) Statistics

**Maximum tokens:** 200000      **Allocated tokens:** 61440  
**Tokens In Use:** 60060      **File structures:** 41259  
Token obtains: 336674      Token returns: 271510  
Token revokes: 125176      Async Grants: 64  
**Garbage Collects:** 0      TKM Establishes: 0  
**Thrashing Files:** 4      **Thrash Resolutions:** 131

### Usage Per System:

System	Tokens	Obtains	Returns	Revokes	Async Grt	Establish
DCEIMGHR	18813	161121	134907	70275	0	0
ZEROLINK	0	66055	66054	5	64	0
LOCALUSR	41247	109499	70549	54974	0	0

Shows token limit, number of allocated tokens, number of allocated file structures and number of tokens allocated to systems in sysplex.

Number of times tokens had to be collected from sysplex members due to tokens reaching limit – if high then might want to update **token\_cache\_size**

Thrashing files indicates objects using a z/OS Unix-style forwarding protocol to reduce callbacks to clients – check application usage

- Shows tokens held per-system and number of token obtains and returns since statistics last reset.
- ZEROLINK – pseudo-sysplex client used for file unlink when the file still open – used to know when file fully closed sysplex-wide to meet POSIX requirement that a file's contents are not deleted, even if its been unlinked, if processes still have file open.



# Sysplex Statistics III: F ZFS,QUERY,CTKC

SVI Calls to System PS1		
SVI Call	Count	Avg. Time
<b>GetToken</b>	<b>1286368</b>	<b>1.375</b>
GetMultTokens	0	0.000
ReturnTokens	26	0.050
ReturnFileTokens	0	0.000
FetchData	0	0.000
StoreData	540	1.566
Setattr	0	0.000
FetchDir	7140	6.291
Lookup	0	0.000
GetTokensDirSearch	0	0.000
Create	1320406	3.736
Remove	1499704	1.595
Rename	166498	1.448
Link	169176	1.549
ReadLink	0	0.000
SetACL	0	0.000
.....		
FileDebug	0	0.000
*TOTALS*	4449858	2.167

Shows requests a sysplex member sends to other sysplex members for objects in file systems owned by other members and average response time in milliseconds. Includes XCF transmission time.

Might be able to reduce GetToken calls by raising **vnode\_cache\_size** (if zFS primary storage allows it)

# Sysplex Statistics IV: F ZFS,QUERY,SVI

## SVI Calls from System PS2

SVI Call	Count	Qwait	XCF Req.	Avg. Time
GetToken	1286013	0	0	0.259
GetMultTokens	0	0	0	0.000
ReturnTokens	26	0	0	0.050
ReturnFileTokens	0	0	0	0.000
FetchData	0	0	0	0.000
StoreData	540	0	0	0.081
Setattr	0	0	0	0.000
FetchDir	7140	0	0	4.997
Lookup	0	0	0	0.000
GetTokensDirSearch	0	0	0	0.000
Create	1321096	0	0	2.371
Remove	1499689	0	177	0.645
Rename	166500	0	0	0.509
Link	169608	0	0	0.538
ReadLink	0	0	0	0.000
SetACL	0	0	0	0.000
....				
LkupInvalidate	0	0	0	0.000
FileDebug	0	0	0	0.000
*TOTALS*	4450612	0	177	1.044

Shows calls received by indicated plex member:

- Qwait non-zero when all server tasks are busy
- XCF Req. means server had to reclaim tokens from other sysplex members to process request.
- Avg. Time in milliseconds shown for server to process request.



# Publications of Interest



- **SA23-2283-00 z/OS V2R1.0 Using REXX and z/OS UNIX System Services**
- **SA23-2280-00 z/OS V2R1.0 UNIX System Services Command Reference**
- **SA23-2285-00 z/OS V2R1.0 UNIX System Services File System Interface Reference**
- **SA23-2284-00 z/OS V2R1.0 UNIX System Services Messages and Codes**
- **GA32-0884-00 z/OS V2R1.0 UNIX System Services Planning**
- **SA23-2282-00 z/OS V2R1.0 UNIX System Services Programming Tools**
- **SA23-2281-00 z/OS V2R1.0 UNIX System Services Programming: Assembler Callable Services Reference**
- **SA23-2279-00 z/OS V2R1.0 UNIX System Services User's Guide**
- **SC23-6887-00 z/OS V2R1.0 Distributed File Service zFS Administration**
- **SC23-6885-00 z/OS V2R1.0 Distributed File Service Messages and Codes**





# Trademarks and Disclaimers

- See <http://www.ibm.com/legal/copytrade.shtml> for a list of IBM trademarks.
- **The following are trademarks or registered trademarks of other companies**
  - UNIX is a registered trademark of The Open Group in the United States and other countries
  - CERT® is a registered trademark and service mark of Carnegie Mellon University.
  - ssh® is a registered trademark of SSH Communications Security Corp
  - X Window System is a trademark of X Consortium, Inc
- **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.

# Connect with IBM System z on social media!



Subscribe to the new [IBM Mainframe Weekly](#) digital newsletter to get the latest updates on the IBM Mainframe!



[System z Advocates](#) \*\*  
[IBM Mainframe- Unofficial Group](#)  
[IBM System z Events](#)  
[Mainframe Experts Network](#)  
[SHARE](#)



[IBM System z](#) \*\*  
[IBM Master the Mainframe Contest](#)  
[IBM Destination z](#)  
[SHARE Inc.](#)



[IBM System z](#) \*\*  
[IBM System z Events](#)  
[Destination z](#)  
[SHARE](#)

## System z SMEs and Executives:

Deon Newman - [@deonnewm](#)  
Steven Dickens - [@StevenDickens3](#)  
Michael Desens - [@MikeDesens](#)  
Patrick Toole - [@Pat Toole II](#)  
Kelly Ryan - [@KellykmRyan](#)  
Richard Gamblin - [@RichGx](#)

## Blogs

[IBM Mainframe Insights](#) \*\*  
[Millennial Mainframer](#)  
[#MainframeDebate](#) blog  
[SHARE blog](#)  
[IBM Destination z](#)



[IBM System z](#) \*\*  
[Destination z](#)



[IBM Mainframe50](#)

Include the hashtag [#mainframe](#) in your social media activity and [#mainframe50](#) in 50<sup>th</sup> anniversary activity