#SHAREorg

SHARE
Technology · Connections · Results

z/OS Unix and zFS Sysplex Sharing

Scott Marcotte
IBM

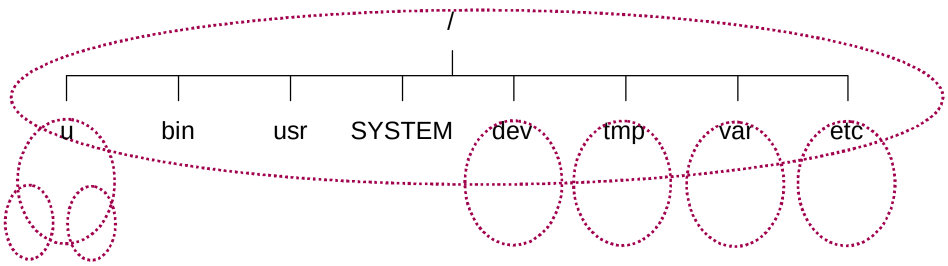
February 8, 2013 9:30AM
Yosemite B
Session Number 13023

smarcott@us.ibm.com

SHARE in San Francisco 2013

The z/OS UNIX file system

- General user view of file system tree:

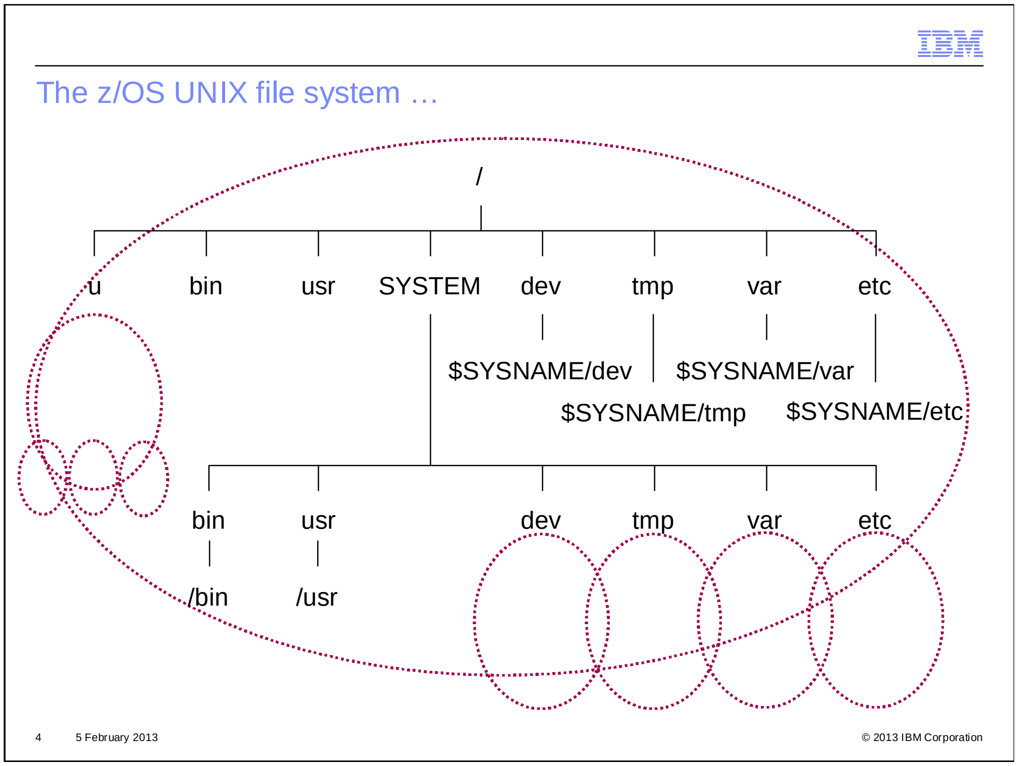


2 5 February 2013

Regardless of whether z/OS Unix Sysplex Sharing is used or not, a general user application will see the standard Unix file system tree.

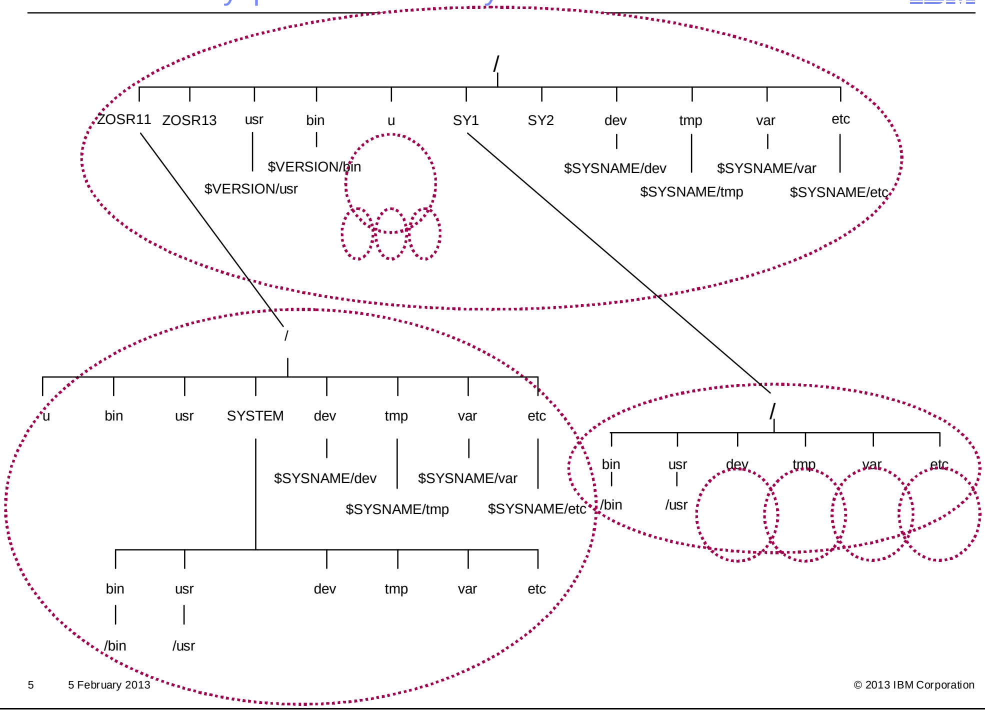
The z/OS UNIX file system …

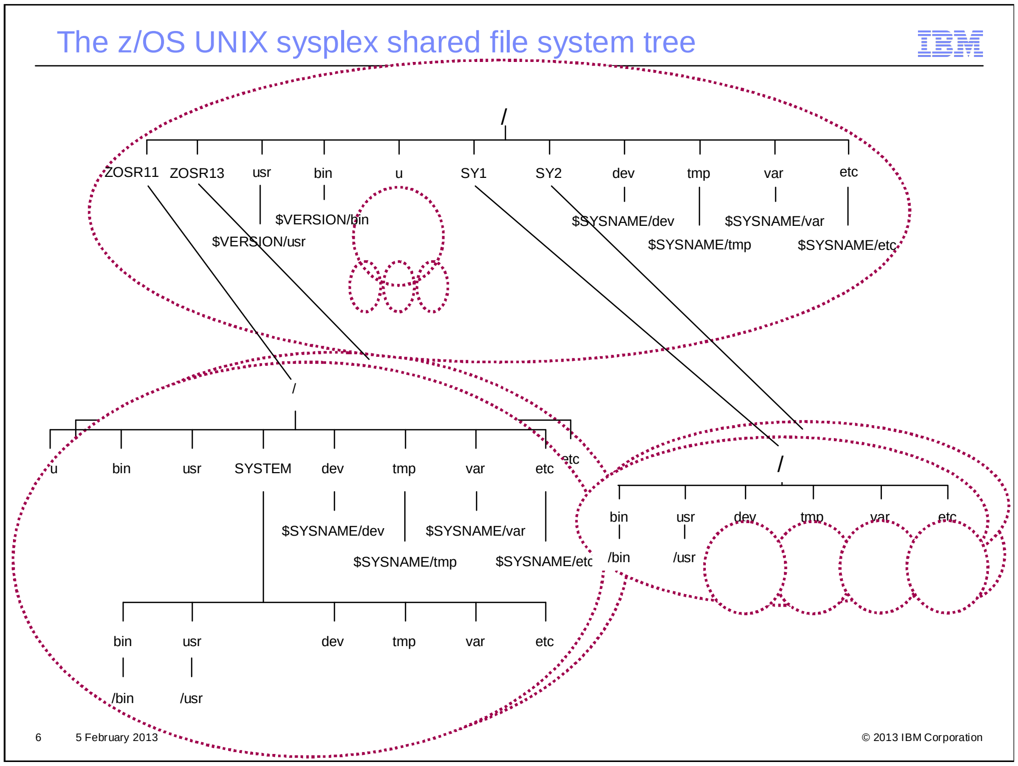
- The z/OS hierarchical file system is actually a bit more involved than the previous slide shows
- The sysplex shared file system environment needs to support multiple concurrent z/OS releases and even multiple concurrent service levels for different LPARs in a single file system hierarchy
- We also want to support system specific file systems
- The sysplex shared file system environment uses special mount points and symbolic links with system symbols to provide this
- Even a single system uses symbolic links to allow easy transition to the shared file system environment
- But, the end user view of the z/OS UNIX hierarchical file system does not change whether they are in a sysplex environment or not



If the content of the symbolic link begins with $SYSNAME and SYSPLEX is specified NO, then $SYSNAME is replaced with /SYSTEM when the symbolic link is resolved.

z/OS UNIX sysplex shared file system tree





From SY1 (running z/OS V1R11)

df /bin

Mounted on Filesystem Avail/Total Files Status

/ZOSR11 (OMVS.MNT.ZOSR11.ZD1111.ZFS) 26236/4579200 4294950685 Available

From SY2 (running z/OS V1R13)

df /bin

Mounted on Filesystem Avail/Total Files Status

/ZOSR13 (OMVS.MNT.ZOSR13.ZD1131.ZFS) 1113638/5760000 4294951449 Available

Anyone can access a different system's /etc by using a full pathname such as /SY1/etc

Anyone can access a different release of /bin by using a full pathname such as /ZOSR13/bin

The z/OS UNIX file system in a shared file system environment

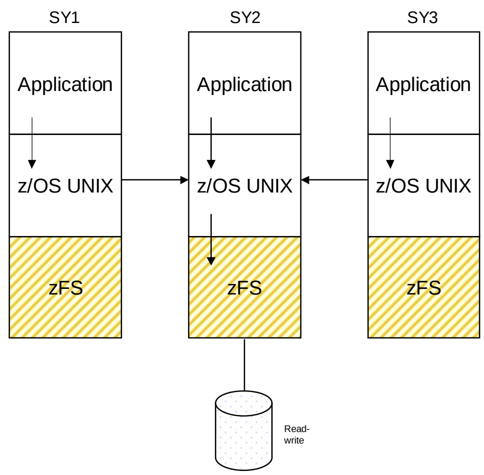
- The z/OS UNIX file system is configured by specifications in the BPXPRMxx Parmlib members
  - xx list is specified in IEASYS00 as OMVS=(xx,yy)
  - BPXPRMxx contains
    - **FILESYSTYPE** statements to initiate PFSes
    - Mount statements for root and lower file systems
      (At IPL, if a file system is already mounted, this is accepted silently)
    - **SYSPLEX(YES)** specifies shared file system environment
    - In this case a z/OS UNIX CDS (Couple Data Set) is required
    - **VERSION('ZOSR13')** specifies the value of $VERSION
      (When SYSPLEX(YES) is specified, you must specify VERSION)
  - IEASYMxx contains
    - **SYSDEF SYSNAME(SY1)** specifies the value of $SYSNAME
    - If the content of the symbolic link begins with $SYSNAME and SYSPLEX is specified NO, then $SYSNAME is replaced with /SYSTEM when the symbolic link is resolved.
  - When a file system is mounted, zFS allocates and opens the data set

You can use System Symbols in Parmlib members. For example,

In IEASYS00, you can say OMVS=(&SYSCLONE.,01)

The z/OS UNIX shared file system environment

- In a parallel sysplex environment, z/OS UNIX can provide access to the entire file system hierarchy for all users from all LPARs in the sysplex
- This environment is called a **shared file system environment**
- z/OS UNIX provides this support by forwarding file requests from other LPARs (SY1 or SY3) to the LPAR designated as the z/OS UNIX file system owning LPAR (SY2 in this case)
- When you mount a file system, an owning LPAR is designated and the file system is mounted and available to all LPARs in the shared file system environment
- **z/OS Unix will cache name-lookups and security attributes at non-owners for faster pathname search and proper security checking.**



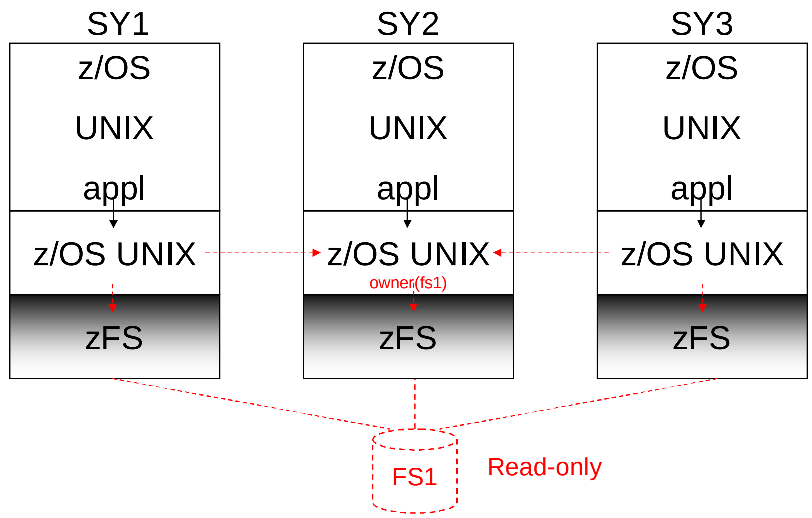
Benefits of the shared file system environment

- **System independence**
  All z/OS UNIX data can be accessed from any system in the sysplex
- **Availability**
  If a system is shutdown or if it abnormally goes down, file system ownership movement occurs automatically and file systems remain accessible from the other systems (although temporary failures may be visible to applications during abnormal shutdown)
- **Flexibility**
  General users and applications automatically access the correct (system specific and/or version/release) file systems while administrators can access any file system
- **Transparency**
  Users and applications do not need to change to run in a shared file system environment (except to possibly handle and recover from the temporary failures)

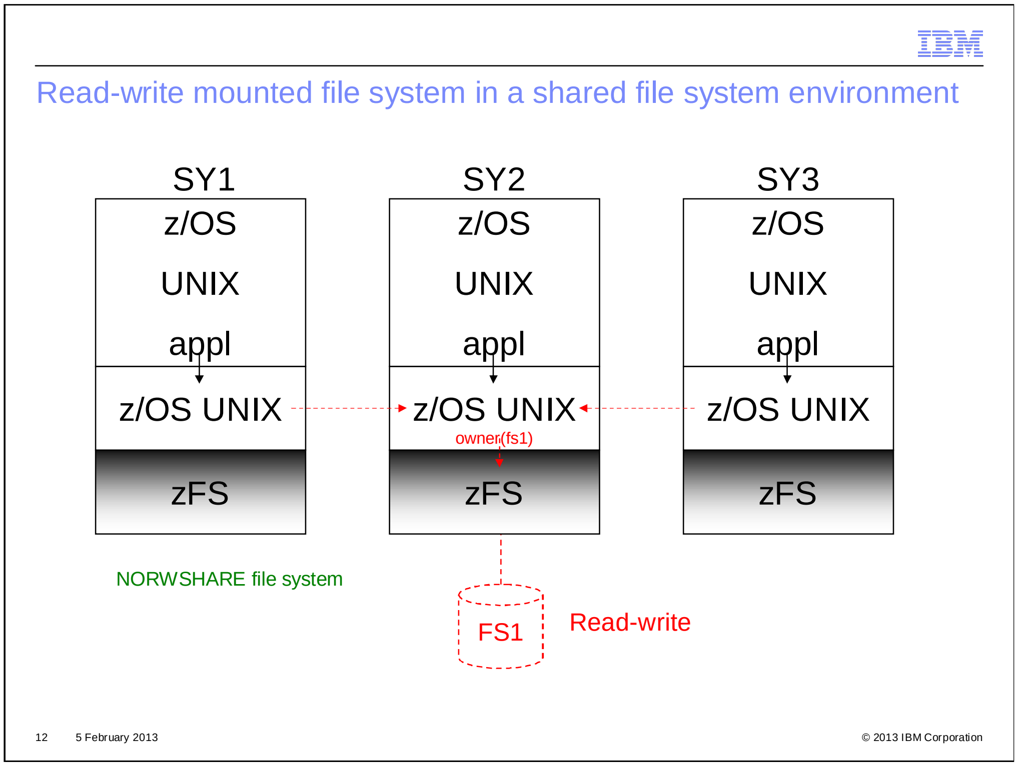
z/OS zFS RW Sysplex File Sharing

- zFS provides sysplex aware support for RW mounted file systems with GA of z/OS 11.
- zFS also provided SPE, for z/OS 11 and 12 to allow user to selectively enable zFS sysplex sharing support for RW mounted file systems on a per-file system basis.
- Slides 11-19 show details of the z/OS 12 support.
- z/OS 13 provides significant enhancements to the support, described in slides 20-25.
- z/OS 2.1 provides significant improvement to directory performance and scale-ability, described on slide 26.

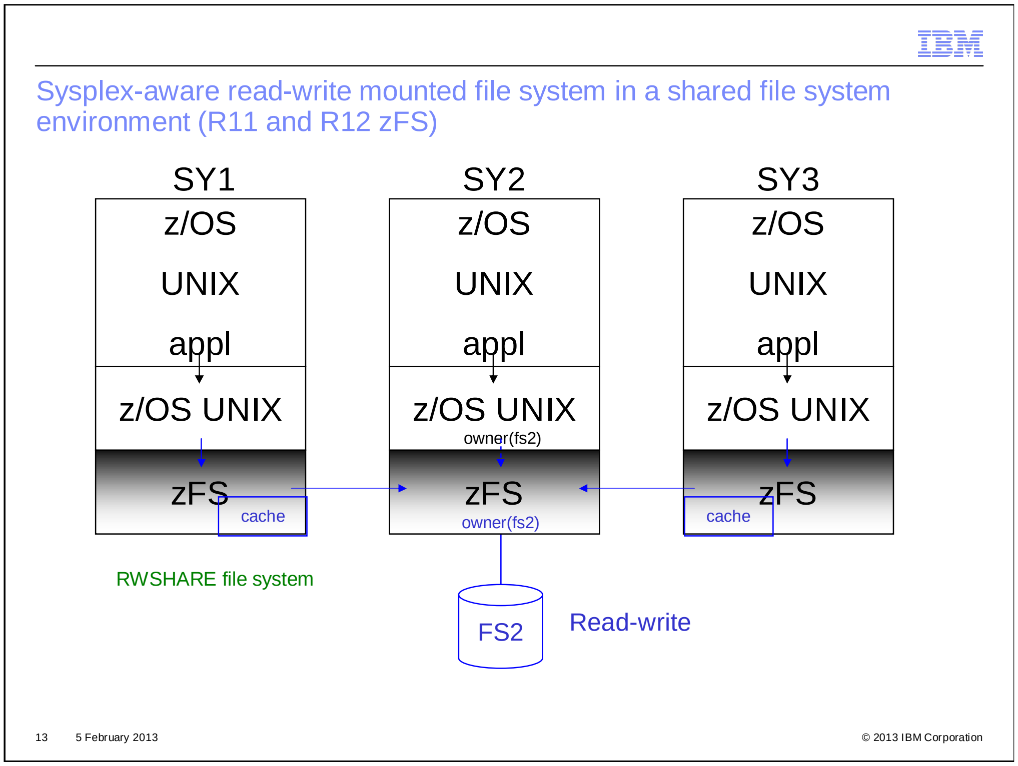
Read-only mounted file system in a shared file system environment



This shows a shared file system environment in a sysplex (z/OS UNIX BPXPRMxx specifies SYSPLEX(YES)). When a zFS file system is mounted read-only, it is locally mounted on each system. Read-only mounted file systems are always sysplex-aware. This has always been the case for shared file system environments. There is no communications between sysplex members to access a read-only mounted file system. zFS on each system will directly read from the DASD and aggressively cache file and directory contents and attributes (such as mtime, length, owner id, etc….)

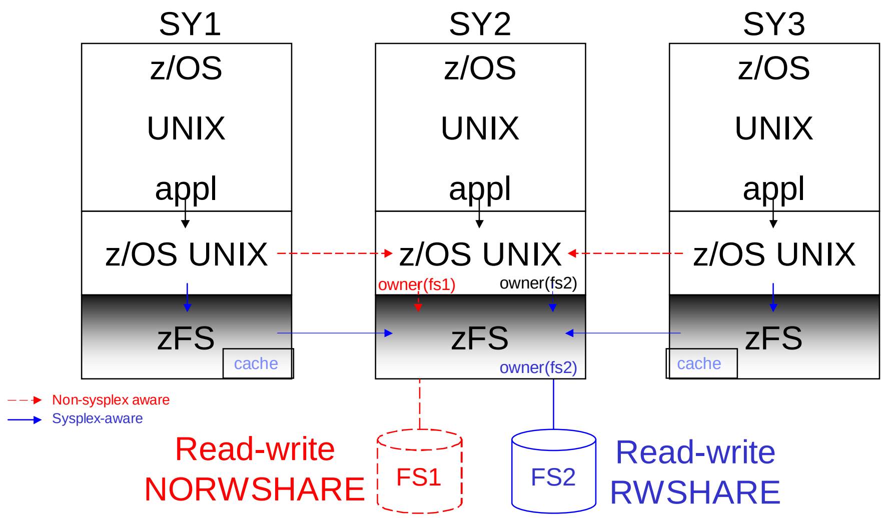


In contrast to a read-only mounted file system, this is a read-write mounted file system. One system is the owning system and the other systems are “clients”. The file system is only locally mounted on the owning system. (It is still externally mounted and available on all systems.) Applications running on the owning system (SY2) access the file system locally (without any XCF communications). But applications that run on the other systems (SY1 and SY3), access the file system through the use of z/OS UNIX function shipping (using XCF communications). That is, the request is forwarded by the z/OS UNIX on that system (SY1 or SY3) to the z/OS UNIX running on the owning system (SY2) which then program calls the local zFS. The response goes back along the same path. So, access to the file system from systems other than the owner (that is, from the client systems) involves XCF communications. This makes it important to have the z/OS UNIX owning system, be the system that is doing the most accesses. zFS on the owning system (SY2) will aggressively cache file and directory contents and attributes, but there is no caching of file or directory contents on the other sysplex members; hence, re-reads of the same file from an application on a non-owning system have to repeatedly communicate with the owner to obtain the file contents (hopefully its still cached in the zFS owner’s memory to avoid disk reads). A file system that is shared this way is referred to as a NORWSHARE file system.



Here is a picture of the new R11 zFS sysplex-aware for read-write support. When zFS runs sysplex-aware (for read-write) on all systems, a read-write mounted sysplex-aware file system is locally mounted on all systems. There is still a z/OS UNIX owning system but there is no z/OS UNIX function shipping to the owner. Rather, requests from applications on any system are sent directly to the local zFS on each system. This means it is now the responsibility of the local zFS to determine how to access the file system. One of the systems is known as the zFS owning system. This is the system where all I/O to the file system is done. zFS uses function shipping to the zFS owning system to access the file system. (If this was all that zFS did, it would be essentially the same as the z/OS UNIX function shipping as shown in the previous slide.) However, each zFS client system has a local cache where it keeps the most recently read file system information. So, in many cases (when the data is still in the cache), zFS can avoid the zFS function shipping (and the XCF communications) and satisfy the request locally in many cases. A file system accessed in this manner is known as an RWSHARE file system.

Mix of NORWSHARE and RWSHARE File Systems (zFS APAR OA29619)



14 5 February 2013

z/OS 11 and 12 zFS allows the customer to use zFS sysplex support on a subset of their file systems. Here are two zFS read-write file systems on a sysplex running zFS sysplex-aware on file system basis on all members. One file system (FS1) is mounted NORWSHARE and the other (FS2) is mounted RWSHARE. They are both z/OS UNIX owned on SY2. The NORWSHARE file system (FS1) acts like a non-sysplex aware file system (it is only locally mounted on the z/OS UNIX owner (SY2) and requests from z/OS UNIX clients (SY1 and SY3) are function shipped to the z/OS UNIX owner (SY2) by z/OS UNIX).

The other file system (FS2) is mounted RWSHARE. It acts like a sysplex-aware file system (it is locally mounted on all systems and z/OS UNIX never function ships requests to the z/OS UNIX owner (SY2)).

When you run zFS sysplex-aware on a file system basis on all your members, the zFS Physical File System initializes as sysplex-aware but it can individually determine which file system is sysplex-aware and which is not based on the RWSHARE/NORWSHARE mount parm.

Benefits and Considerations for z/OS 11 and 12 zFS Sysplex File Sharing Support

- **Automatic Movement of zFS Owner to High Usage Clients**
  - zFS owner will move file system ownership to another member that has substantially higher application access.
- **Improved Performance for Certain Workloads:**
  - 6X improvement seen in large file read/write workloads run on a non-owner system
  - >10X or more improvement seen in cached re-reads of files and sequential file writes from applications (due to zFS caching on a sysplex client machine)
  - No improvement for directory workloads (z/OS 13 zFS improves directory workloads and **z/OS 2.1 does much better**)
- **Products That Do Not Support zFS Sysplex File Shared File Systems (RWSHARE file systems):**
  - 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, if any, 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.

15 5 February 2013

zFS will automatically move ownership to another sysplex member, if that sysplex member has a substantially higher application access rate than the current owner. This alleviates the administrator from the need to having to manually move the file system to the best-fit owner in many cases.

zFS file reads and writes are substantially faster on non-owning systems than the original z/OS Unix Shared File support, and is well suited for applications that have a high rate of repeated access to the same files from multiple plex members. Directory operations do not run any faster on a non-owner system for z/OS 11 RWSHARE file systems over NORWSHARE (directory update operations are much less frequent than file operations in most real-world environments). However, z/OS 13 zFS DOES improve directory update operation performance from non-owners over NORWSHARE.

There are certain products that do not support zFS RWSHARE file systems (zFS R/W sysplex sharing). If these products are active at your site, then you would not want to use zFS Sysplex File Sharing at all, or you would only want to use it for file systems that are not going to be accessed by these products. Thus if z/OS SMB server was running, only file systems that are not used by SMB server should be RWSHARE, and any file systems that are accessed by SMB server should be NORWSHARE.

Using zFS Sysplex File Sharing

- **For z/OS 11/12, Install zFS APAR OA29619 and co-req z/OS UNIX APAR OA29712**
  - Recommended to allow file sharing to be specified on a file system basis AND
  - Avoid potential performance problems
- **Use SYSPLEX=FILESYS Option**
  - Indicates that zFS RW sysplex file sharing is specified on a per-file system basis.
- **Specify 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. 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.
- **z/OS 11/12: Specify CLIENT_CACHE_SIZE=YYYY** if the current 128M default is not desired.
  - zFS sysplex introduces a new cache to contain cached file data for non-owning systems, the default is 128M. Heavy file use from a non-owner may benefit from a larger cache.
- **Other Important Tuning Parameters:**
  - **VNODE_CACHE_SIZE** – Determines the number of files and directories that a system (owner or non-owner) may cache in memory, default is 65536.
  - **META_CACHE_SIZE** – Determines the amount of directory data cached in memory on owners and non-owners (default 64M).

16 5 February 2013

All of the parameters described on this page are specified in the zFS kernel parameters file (IOEFSPRM) which can take advantage of parmlib search. Most of these options can also be configured with the zfsadm config command.

zFS provided an SPE (APAR OA29619) that allows zFS sysplex file sharing to be specified on a file system basis (via parameters to the MOUNT command). This APAR not only provides new functionality, but makes it safer to roll in zFS sysplex support. The z/OS 11 GA code only allowed the values OFF or ON for this setting, with the default being OFF. This was fine if zFS RW sysplex function was not desired, but could cause performance issues, and confusion if it was desired. The GA level of code only allowed the option of performing a rolling IPL specifying SYSPLEX=ON for each member. The problem was that it created a situation where some systems were using zFS sysplex and some were not for a file system. This could cause a double-transmission if the zFS owner was not the same as the z/OS Unix owner. Thus the SPE and specifying SYSPLEX=FILESYS eases transition to the use of zFS sysplex.

Along with specifying SYSPLEX=FILESYS, the customer should also specify a value for SYSPLEX_FILESYS_SHAREMODE (either RWSHARE or NORWSHARE, with the latter the default). This is the default zFS RW sysplex file sharing mode for any file system mounted RW in the sysplex that does not explicitly specify the new parameters NORWSHARE or RWSHARE. z/OS 11 zFS introduced these new parameters to allow the user to control whether a file system, that is mounted R/W mode, uses zFS RW sysplex file sharing. This allows customers using SMB server for example, to allow the file systems not used by the server to use zFS sysplex sharing.

If zFS sysplex file sharing is used, there are three tuning parameters that control the number of files and directories (VNODE_CACHE_SIZE) that can be cached in memory on a system, the amount of file data (CLIENT_CACHE_SIZE) that can be cached in memory on non-owner systems for files accessed by applications for zFS RW shared file systems, and the amount of directory data that can be cached in memory on a system (owner or non-owner). The zFS parameter USER_CACHE_SIZE controls the amount of file data that can be cached on an owner system for RW shared file systems, and for RO mounted file system data.

Using zFS Sysplex File Sharing …continued

- **Useful Commands:**
  - **F ZFS,QUERY,LEVEL** – shows the value of the SYSPLEX setting currently in use for zFS.
  - **Zfsadm lsaggr** – shows the zFS owner of a file system.
  - **Zfsadm aggrinfo –long** – indicates if zFS RW sysplex sharing is being used for the file system.
  - **F ZFS,QUERY,FILE** – indicates if zFS RW sysplex sharing is being used for the file system.
  - **df –v** – used on a system to determine if the file system on that system is using zFS sysplex file sharing.
    - The command shows Client=N if its using zFS sysplex file sharing, or if it's a RO mounted file system, and shows Client=Y if its mounted RW and not using zFS sysplex sharing.
  - **F ZFS,QUERY,SETTINGS** – shows all the current settings of the zFS kernel, including all the parameters described on the prior slide.
  - **Zfsadm config** – can dynamically tailor zFS kernel parameters, including all of the parameters on the prior slide.
    - → Except SYSPLEX which is a parameter only read at IPL time.

Robust 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.
- **Communication Failures and Timeouts**
  - Uses a timeout mechanism to prevent hang-ups in the plex 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 plex members.
  - Provides **F ZFS,NSV** command to force a consistency check between plex member's file system state and correct any problems
- **Informative Messages Provided**
  - zFS provides many lasting operator 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.

18 5 February 2013

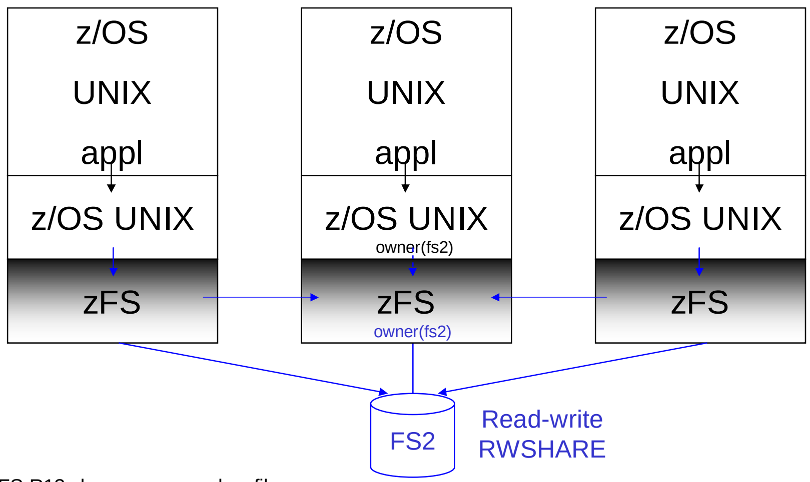
zFS handles dead system recovery much like z/OS Unix does. It will suspend applications until a new owner is found and has re-readied the file system for use and then resume application access to that file system. It will return errors to applications that had open files at the time an owner went down so they know that data might have been lost for a file. zFS is faster in finding a new owner and re-readying the file system for syplex use than z/OS Unix sysplex sharing since zFS only re-establishes connections, to open files and not every cached file at the time of dead system recovery.

zFS can handle lost transmissions and replies between sysplex members, and also prevent an application from suspending indefinitely if a plex member takes too long, or is having trouble processing a request. The file system remains available (as much as possible) and consistent in terms of cached data for the file system. zFS sysplex file system name-spaces (which records file system processing attributes and owner information) can be made consistent even in the event of lost message transmissions, replies or even requests that get hung up. zFS will automatically initiate a sysplex-wide repair and the user can always force this processing to check for consistency and repair state via the F ZFS,NSV command. It can even repair inconsistencies that were due to software errors.

Additional Notes

- **System Specific File Systems Can Use zFS RW Sysplex Sharing**
  - These 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.
- **R/O File Systems Unaffected** – The operation of file systems mounted R/O are not affected by the zFS RW sysplex file sharing introduced in z/OS 11.
- **Still Recommended that the Sysplex Root be mounted R/O**
- **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.
- **Do Not Use zFS Sysplex File Sharing Unless all Plex Members z/OS 11+** – z/OS 10 or prior systems do not have the RW zFS sysplex file sharing support, and therefore, using zFS sysplex file sharing (RWSHARE) is not recommended as double-transmission problems can occur (not to mention making things more confusing). All plex members should be at release 11 or later before attempting to use zFS sysplex file sharing.

z/OS 13 zFS Sysplex Sharing Support



zFS R13 always runs sysplex=filesys

Here is a picture of the R13 zFS sysplex-aware for read-write support. When a zFS file system is mounted sysplex-aware (for read-write), it is locally mounted on all systems. There is still a z/OS UNIX owning system but there is no z/OS UNIX function shipping to the z/OS UNIX owner. Rather, requests from applications on any system are sent directly to the local zFS on each system. This means it is now the responsibility of the local zFS to determine how to access the file system. One of the systems is known as the zFS owning system. This is the system where metadata updates to the file system are written to DASD. zFS uses direct I/O to the file system from each system for user file contents and full asynchronous write-behind with full POSIX semantics of this data. Additionally, zFS can directly read the contents of directories from disk on non-owning systems; though the owner is still contacted when changes are made, by applications to a directory.

z/OS 13 zFS Performance Benefits

- **Asyncronous write-behind on non-owners**
- **Reduced XCF transmission usage by zFS**
  - Directory and file contents read directly from disk, no longer on “wire”.
  - Makes ownership location less critical.
- **Better server (owner) scalability**
  - Significantly reduced message rate to servers, less CPU used at servers
- **Use USER_CACHE_SIZE to control non-owner caching**
  - Instead of CLIENT_CACHE_SIZE
  - Can simply specify how much data to cache on a system, no need to break it down into owner and non-owner partitions, can use minimal CLIENT_CACHE_SIZE (10M) when all systems z/OS 13 or later.
- **Eliminate directory cache and DIR_CACHE_SIZE parameter**
  - Only affected non-owner systems, used only for R/O data or NORWSHARE file systems.
  - Duplicated directory data and caused extra data movement for directory read misses.
- **Better resolution of multi-member contention on the same files and/or directories**
  - zFS dynamically switches to a protocol designed to better handle cases where multiple members are writing to the same file or directory
- **Non-owners can cache more data**
  - Non-owners can now make use of the metadata backing cache (a dataspace) used to cache more directory data. Metadata cache is limited to the storage available below the 2G bar.
    - **METABACK_CACHE_SIZE** controls the size of this cache.

21 5 February 2013

z/OS 13 provides asynchronous write-behind on non-owners with full POSIX semantics, where z/OS 11 could only use write-behind in certain situations. Sysplex non-owners (clients) will directly read and write user file data to disk, eliminating transmission of that data to the owning system. The owning system still updates the metadata but these packets are small compared to the size of the file contents and reduces much overhead on owners. This makes ownership location in the plex less critical since performance on non-owners is much improved.

Configuration is slightly simpler. The directory cache used to contain directory data for RO mounted file systems, and RW mounted file systems on owners, has been removed. The prior releases of zFS duplicated directory data in both the metadata cache and the directory cache, resulting in additional CPU used to copy data between the caches and the pain of determining the size of each cache. The metadata cache is now used exclusively to determine how much directory data to cache. Additionally, non-owners now store their cached file data in the user file cache (controlled by USER_CACHE_SIZE parameter), the client cache is used only for compatibility with z/OS 11 and 12 systems. Thus the user can specify the minimum value (10M) for the CLIENT_CACHE_SIZE when all plex members are z/OS 13 and later. Future releases of zFS will remove this cache entirely.

The zFS clients (non-owners) for RWSHARE file systems can now cache additional directory data in a dataspace, controlled by the METABACK_CACHE_SIZE parameter. The metadata cache (controlled by the META_CACHE_SIZE parameter) is limited by the amount of memory zFS has available below the 2G bar that is not being used for other zFS internal structures.

z/OS 13 Performance Results (z9 machine)

- **Large File (database) Update Workload:**
  - This workload randomly updates a large file, similar to a database access.
  - 50% throughput improvement for large file read/write update workload for NORWSHARE file systems over z/OS 11 on non-owner systems.
  - 40-75% throughput improvement for RWSHARE file systems over z/OS 11 on non-owners, scaling better when run on more plex members concurrently than z/OS 11.
  - Small throughput improvement when run exclusively on owner system.
  - 8X faster on non-owners with R13 RWSHARE as opposed to R13 NORWSHARE.
- **Sequential File Creation Workload:**
  - This workload creates many files, sequentially writing data to them (a very common write pattern in the field)
  - 2X-3X throughput improvement for RWSHARE file systems over z/OS 11. Clients saw a 2X improvement, and the entire system scaled better when multiple systems ran the same workload (3X in this case).
  - Small reduction in CPU when run exclusively on owner systems (couple of pct.)
  - 16X faster on non-owners with R13 RWSHARE as opposed to R11 NORWSHARE.
- **Directory Update Workload:**
  - This workload has many processes repeatedly adding, removing, renaming and searching for files in a directory, not a typical customer environment.
  - 3X improvement for RWSHARE file systems over z/OS 11 on non-owners and a 25% improvement over NORWSHARE performance.
  - Small throughput improvement when run on owner system.
  - Small throughput improvement when run on multiple systems.
  - 25% faster on non-owners with R13 RWSHARE as opposed to R11 NORWSHARE.

22 5 February 2013

zFS performance was analyzed for file and directory workloads and compared to z/OS 11 zFS. For RWSHARE file systems, performance on non-owner systems improved up to 3X over z/OS 11. Owner systems also showed a slight improvement in throughput or a slight reduction in CPU consumption. The file workloads, when run on multiple systems scaled quite nicely, and z/OS 13 scaled better than z/OS 11 in the measured environment. For directory workloads, the non-owner performance was significantly improved, and owner saw a slight improvement too. One area of improvement for zFS is improving high directory update performance when run on multiple systems. High directory update rates, as used in the measured workload are very rare in the field, but future zFS line items are geared to improving directory performance in general to result in continue throughput improvements in future releases.

Most customer that use zFS, should see improved performance when migrating to z/OS 13 zFS.

Improved Error Handling in z/OS 13 zFS

- **Improved Critical Error Handling**
  - Severe software errors could stop zFS and z/OS Unix would restart it
  - z/OS Unix would re-mount file systems if in a shared file environment, single system environments would lose the entire tree
  - z/OS Unix restart of zFS took a very long time on large systems
  - zFS now internally restarts, internally re-mounting file systems and reducing the number of errors seen by applications, this is much faster than z/OS Unix restart
  - Single system environments no longer lose the mount tree
  - An administrator can initiate the process via **F ZFS,ABORT**
    - Obtains a dump to report to IBM and initiates internal restart.
    - Good for when something seems not right and yet cannot be corrected (a persistent error)
- **Improved File System Error Handling**
  - If a software error affects only a single file system, that file system would be disabled for access, this is not new to z/OS 13 zFS.
  - With z/OS 13 zFS, zFS will cause an internal re-mount if the file system is NORWSHARE and would initiate an owner move if RWSHARE to clear the condition automatically.
  - Will attempt this at most 3 times per file system, in case its permanently damaged.

Most software errors in zFS do not restart zFS or disable a file system. But some errors are too severe, and either require a file system to be disabled (to protect it from permanent corruption) if the error is restricted to a specific file system, or require zFS to stop if it affects the entire system.

In prior releases of zFS, a severe error, affecting system-wide zFS operation would force zFS to stop. If the system was part of a shared file system environment (a sysplex), then z/OS Unix would restart zFS and move ownership of file systems to other members and re-mount file systems on the system where zFS was restarted. The loss of zFS took time for z/OS Unix to realize, which resulted in many applications on that system receiving errors, and would take a very long time for z/OS Unix to restart zFS and re-mount file systems. With z/OS 13 zFS, zFS will internally restart itself, and internally re-mount file systems with a clean zFS memory. z/OS Unix processes are made to wait while zFS is re-starting. This resulted in far fewer applications receiving errors as zFS quickly stops user activity as soon as the severe error is found and quickly re-readies file systems and resumes user activity for a given file system as soon as its re-mounted. File systems are re-readied in priority order, based on the number of applications waiting for access to that file system.

Additionally, if the error resulted in disablement of a file system, prior releases of zFS required the administrator to unmount and re-mount the file system to clear the condition. zFS will now initiate this process if the file system is NORWSHARE, performing a re-mount without switching modes. If the file system is an RWSHARE file system, then another zFS member will assume ownership of the file system.

z/OS 13 zFS Migration

- **All members must specify SYSPLEX=FILESYS**
  - Note that z/OS 13 zFS assumes SYSPLEX=FILESYS operation.
- **Toleration APAR OA32925 (PTF UA55765) required on z/OS 11 and 12 sytems.**
  - And they need to have been IPLd with SYSPLEX=FILESYS operation.
  - ZOSMIGV1R13_ZFS_FILESYS health check provided to ensure these conditions are met.
- **z/OS 13 zFS might use more DASD space in certain situations**
  - A zFS file system is an array of 8K blocks
  - Prior releases of zFS stored small files in 1K fragments stored in the same 8K physical block on the dataset to save disk space.
    - Often was not useful since the older code did not aggressively place the files in the same block, but rather scattered these small files randomly in separate blocks.
  - Files and directories are no longer stored in 1K fragments
    - Though symbolic links and ACLs are aggressively packed using fragments in z/OS 13.
  - First write to a directory or file that is stored in the fragmented method converts it first to blocked.
    - Required to allow systems to directly read and write the contents, there is no means in z/OS to physically write to a 1K fragment.
  - File systems composed mostly of small files, can see increased disk space usage with z/OS 13.

z/OS 13 requires that the prior releases of zFS in the plex run SYSPLEX=FILESYS. Additionally, those releases should have the toleration APAR OA32925 applied. z/OS 13 zFS performs IO directly from multiple plex members. The smallest unit of physical IO to a zFS linear dataset is 4K (the control interval size), but zFS logically considers the file system an array of 8K blocks.

zFS also will logically break up each 8K block into eight 1K fragments. With prior releases of zFS, small files and directories might be stored in a series of contiguous fragments inside an 8K block and thus multiple files or directories stored in the same 8K block. The problem with prior releases of zFS is that they randomly distributed the files amongst the blocks, so it was often the case that multiple small files were placed in separate blocks (not really saving any disk space). With z/OS 13 zFS, and the need to be able to directly read or write the contents of a file or directory from any plex member, the use of fragments to store new file or directory data was discontinued. z/OS 13 does aggressively try to pack symbolic link contents and ACL contents into the same 8K block to conserver disk space (instead of relying on random placement of prior releases), but it only stores file and directory data in whole 8K blocks. A first-write on a z/OS 13 system to a file or directory stored in fragments inside a block, will convert that directory or file to blocked format and its thus stored in whole 8K blocks allowing for direct access from all plex members. Thus in the extreme example where all files and directories are small, and file system space is tight, zFS will use more disk space for that file system as each file is updated in z/OS 13, growing the file system as necessary. In many cases there will be little or no growth since the files or directories were already larger than 8K or the small files were not properly packed into the same block (due to the algorithms used by prior releases of zFS).

Additional z/OS 13 Notes

- **Default of AGGRGROW is now ON**
  - Instead of OFF
- **NBS is always ON**
  - NBS refers to new-block-security, and is a guarantee that in the event of a system crash, when the system restarts, no user would see garbage in files that were being written at the time of the crash.
  - NBS is always guaranteed by z/OS 13 zFS and cannot be disabled via this kernel parameter.
- **Every Customer Benefits From z/OS 13 Because:**
  - They get improved throughput and reduced CPU usage, whether they use zFS RW sysplex sharing or not.
  - They get improved availability in the face of critical errors.
  - They get simpler administration.

z/OS 2.1 Sysplex Related Benefits

- **D OMVS,F** – enhanced to properly show zFS file system quiesce status, for both NORWSHARE and RWSHARE file systems.
- Provides the option to create (or convert existing file systems) to a new format that supports very large directories.
  - Substantially improves directory performance for larger directories
    - Both single-system and sysplex.
  - Even small directories get an improvement
  - **RWSHARE sysplex client access gets the most benefit**:
    - Smaller directories see 33% throughput improvement for directory update workload over R13.
    - Directory update workloads now 3X faster than NORWSHARE run from non-owner when run with small directories (<2000 names).
    - Directory read workloads now 18X faster than NORWSHARE run from non-owner with small directories.
    - For larger directories (20,000+ names), gain is even larger.

Publications of Interest

- 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)
  zFS Concepts and zfsadm command for zFS
- z/OS Distributed File Services Messages and Codes (SC24-5917)
  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

System z Social Media Channels

- Top Facebook pages related to System z:
  - IBM System z
  - IBM Academic Initiative System z
  - IBM Master the Mainframe Contest
  - IBM Destination z
  - Millennial Mainframer
  - IBM Smarter Computing

- Top LinkedIn groups related to System z:
  - System z Advocates
  - SAP on System z
  - IBM Mainframe- Unofficial Group
  - IBM System z Events
  - Mainframe Experts Network
  - System z Linux
  - Enterprise Systems
  - Mainframe Security Gurus

- Twitter profiles related to System z:
  - IBM System z
  - IBM System z Events
  - IBM DB2 on System z
  - Millennial Mainframer
  - Destination z
  - IBM Smarter Computing

- YouTube accounts related to System z:
  - IBM System z
  - Destination z
  - IBM Smarter Computing

- Top System z blogs to check out:
  - Mainframe Insights
  - Smarter Computing
  - Millennial Mainframer
  - Mainframe & Hybrid Computing
  - The Mainframe Blog
  - Mainframe Watch Belgium
  - Mainframe Update
  - Enterprise Systems Media Blog
  - Dancing Dinosaur
  - DB2 for z/OS
  - IBM Destination z
  - DB2utor