Using PDSEs in your SYSPLEX: Best Practices

Speaker: Thomas Reed /IBM Corporation

SHARE Seattle 2015

Session: 16955

#SHAREorg

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

Agenda

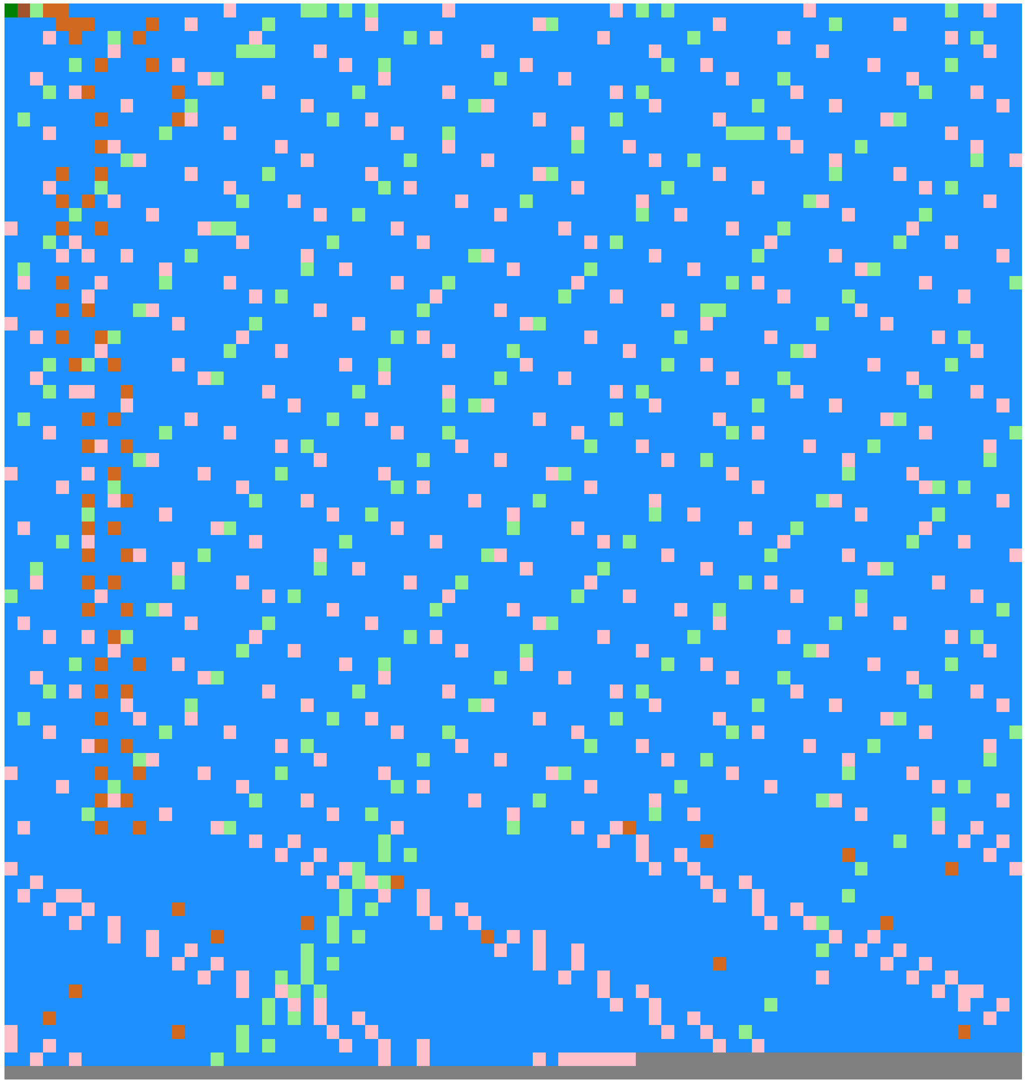
- PDSE Sharing
- PDSE Recoverability
- PDS to PDSE Conversion Considerations
- PDSE and 2.1

What is a PDSE?

- PDSE: <u>P</u>artitioned <u>D</u>ata <u>S</u>et <u>E</u>xtended
- A PDSE is a homogenous collection of directory and data pages
- PDSE server consists of one or two address spaces (SMSPDSE and SMSPDSE1)
- The SMSPDSE(1) address spaces serve client access requests for PDSE data sets
- Under the hood SMSPDSE(1) also manages PDSE serialization and buffering

What Does a PDSE Look Like?

- 🟩 VDF
- 🟧 ND
- ⬜ NOTFMT
- 🟫 BMF
- 🟦 MEMBER
- ▭ Free
- 🟥 LOST
- 🟩 AD



PDSE Sharing

PDSE Sharing Basics

Important Terminology:

- Two sharing modes, NORMAL and EXTENDED
  - NORMAL is the default and fallback mode
  - EXTENDED is preferred in the SYSPLEX environment
- GRSPLEX Scope: A set of systems connected by only GRS
- SYSPLEX Scope: A set of systems connected by both XCF and GRS

*See the Appendix for *NEW* sharing cheat sheets

EXTENDED Sharing Mode: Basics

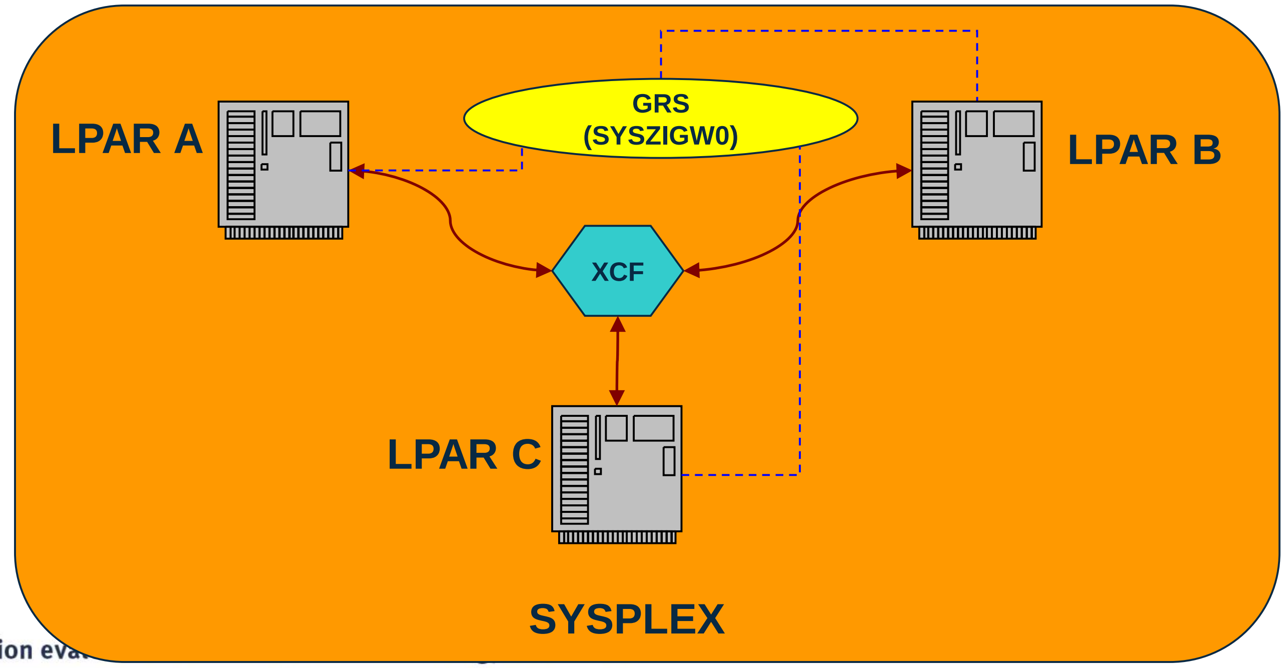
- The newest and preferred sharing mode
- Provides the ability to share at the member level <u>between</u> systems
- Can be implemented with one or both address spaces active

EXTENDED Sharing Mode: Startup

- `PDSESHARING(EXTENDED)` specified in IGDSMSxx member
- The SYSPLEX sharing mode is determined by the first PDSE address space to start within the GRSPLEX
- Mixed sharing modes are not supported
- IPL is recommended to start EXTENDED sharing
  - Starting with the ACTIVATE command is possible
  - ACTIVATE command start may cause PDSE problems
  - See Appendix for ACTIVATE command

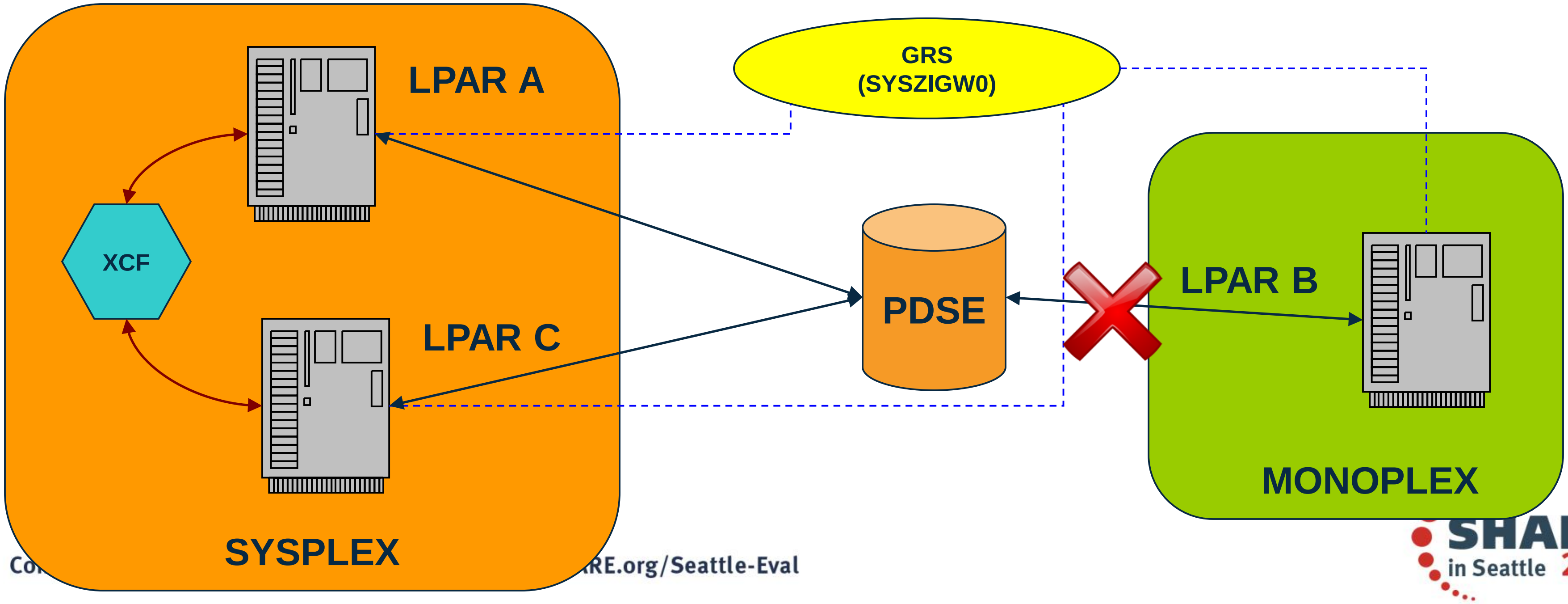
EXTENDED Sharing Mode: Sharing Requirements

- EXTENDED sharing is strictly limited to systems within the same SYSPLEX
- Participating systems must belong to the same GRSPLEX AND XCFPLEX



Improper PDSE sharing: What is it?

- Sharing a PDSE data set outside of a single XCFPLEX while running PDSE sharing EXTENDED
- Also known as sharing outside of the SYSPLEX
- Key point: PDSE sharing EXTENDED requires both GRS and XCF to mediate serialization of data sets



Improper PDSE sharing: Why is it bad?

- Improper sharing can allow for unserialized access to PDSE data sets
  - There is no warning that a data set has been accessed in an unserialized manner
  - The results are unpredictable but may include:
    - Invalid index data in-core
    - Corrupt index data on DASD
    - Corrupt member data
    - Mismatched extent information
    - Nothing at all

Improper PDSE sharing: Common Symptoms

- Corruption can cause 0F4 ABENDs
  - Corruption of the PDSE data set causes logical errors
  - Also may indicate an extent mismatch if the PDSE was moved
- Varied symptoms make improper sharing hard to diagnose
- Many symptoms can be caused by other issues

Improper PDSE Sharing: Admins Beware!

- There is no 100% safe way to circumvent EXTENDED mode's serialization requirements
- PDSE data sets cannot be serialized by third party products
  - Specifies RNL=NO
  - MIM does not serialize PDSEs
- Asking users not to update PDSEs from outside the SYSPLEX
  - Inevitably someone forgets
  - New users may not know the rules
- Reserves can cause serialization deadlocks

NORMAL Sharing Mode: Basics

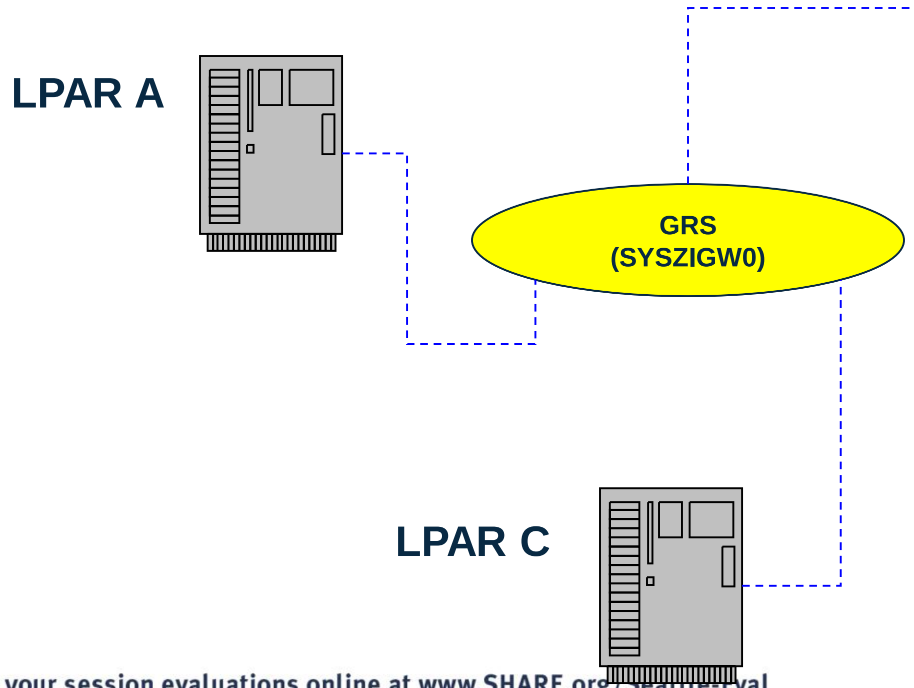
- Legacy PDSE sharing mode
- Provides the ability to share at the data set level <u>between</u> systems
- Shares at the member level on a <u>single</u> system
- Can only be implemented with the non-restartable address space (SMSPDSE)

NORMAL Sharing Mode: Startup

- `PDSESHARING(NORMAL)` specified in IGDSMSxx member
- NORMAL is the default sharing mode
- Mixed sharing modes are not supported
- To change from EXTENDED sharing to NORMAL sharing requires an IPL

NORMAL Sharing Mode: Sharing Requirements

- NORMAL sharing is not limited to systems within the same SYSPLEX
- Participating systems must belong to the same GRSPLEX



NORMAL Sharing Mode: Sharing outside the SYSPLEX correctly

- Why it works:
  - NORMAL mode sharing only utilizes GRS for serialization
  - Multiple SYSPLEXs or stand alone LPARs may share DASD within the same GRSPLEX

- Limitations:
  - Restricts inter-system sharing to the data set level
  - When a system opens the PDSE for OUTPUT it is the <u>only</u> system that can access the PDSE
  - Can decrease performance by blocking opens of the data set

PDSE Recoverability

PDSE Recoverability: SMSPDSE1

- SMSPDSE1 is the restartable PDSE address space
  - SMSPDSE1 handles local data set connections
  - SMSPDSE handles global data set connections
- SMSPDSE1 is only available to systems running PDSE sharing EXTENDED
- It is highly recommended that all customers running EXTENDED sharing take advantage of the restartable address space
- Enabled by IGDSMSxx Parameter
  - `PDSE_RESTARTABLE_AS(NO | YES)`

PDSE Recoverability: The SMSPDSE1 Restart Process

- Why perform a SMSPDSE1 restart?
  - To recover from a situation that would otherwise require an IPL
  - Recover from a PDSE latch hang situation
  - Recover from in-core corruption of a PDSE at 1.12 and below
    - 1.13 and above can use the REFRESH command
  - Recover from excessive PDSE storage usage
- What are the side effects?
  - A small amount of CSA is lost in the restart

PDSE Recoverability: The SMSPDSE1 Restart Process

- Restart Warnings:
  - Do not route the restart command around the SYSPLEX
    - Each LPAR must complete it's restart before restarting the next
  - Depending on the number of connections that need to be quiesced and reconnected it may take a few minutes
  - Some user jobs may not be able to correctly handle the quiesce and reconnect processing and may fail

PDSE Recoverability: How to Restart SMSPDSE1

- Step 1:
  - Gather doc! At a minimum a console dump of SMSPDSE and SMSPDSE1 should be taken
- Step 2:
  - Issue the restart command
  - `V SMS,PDSE1,RESTART [,QUIESCE(duration | 15 )[,COMMONPOOLS(NEW|REUSE) ]`
  - QUIESCE option determines how long in-flight operations have to quiesce
  - COMMONPOOLS option determines whether ECSA cell pools are reused
    - Only select NEW if there was a cell pool problem

PDSE Recoverability: Phases of the SMSPDSE1 Restart

- Quiesce Phase
  - New PDSE requests are corralled
  - By default all in-flight activity has 15 seconds to complete
  - If requests do not complete within the quiesce interval the user has the choice to either wait or continue with the restart
  - Once the quiesce interval completes SMSPDSE1 stops and a new instance is started

PDSE Recoverability: Phases of the SMSPDSE1 Restart

- Reconnect Phase
  - All user connections are restored
  - There is a 15 second time limit on reconnect processing
  - If reconnect cannot be completed within 15 seconds the user can choose to retry for another 15 seconds or continue
  - Users must decide whether they can afford to lose any tasks which don't reconnect in a timely manner

PDSE Recoverability: SMSPDSE1 Restart Failure

- If necessary tasks in SMSPDSE1 have been lost the restart may hang
  - Can result from ABEND S0C1 in the wrong task
- SMSPDSE1 then must be forced down
- The user can then attempt to use the ACTIVATE command to bring SMSPDSE1 back up
- `V SMS,PDSE1,ACTIVATE`
- This is a last resort and should only be used to attempt to avert an IPL

PDSE Recoverability: z/OS 1.13 and Above PDSE Commands

- The **REFRESH** command provides a more granular way to recover from in-core corruption at 1.13 and above
- Avoids a PDSE1 restart for corruption issues
- Discards all in-core pages for the specified PDSE
- Next access will re-read all data from DASD

- `V SMS,PDSE|PDSE1,REFRESH,DSN(dsname)[,VOL(volser)]`

- **WILL NOT** resolve PDSE latch/lock issues

PDSE Recoverability: z/OS 1.13 and Above PDSE Commands

- The **CONNECTIONS** command allows the user to discover where the PDSE is in use at 1.13 and above
- Useful for PDSE in use errors
- A dump no longer has to be taken to determine what tasks are connected to a PDSE

- `D SMS,PDSE|PDSE1,CONNECTIONS,DSN(dsname),<VOLSER(volser)>`

PDSE Recoverability: The IEBPDSE Validation Tool

- IEBPDSE
  - Validates the integrity of PDSE datasets on DASD
    - Checks the indexes (AD, ND and GD)
    - Checks extent information
  - Disregards in-core index pages
  - Linklisted PDSE’s are automatically validated during NIP

PDSE Recoverability: The IEBPDSE Validation Tool

- Recent Enhancements:
  - OA40492
    - Adds AD/ND index utilization statistics
    - Tells us how sparse the indexes are overall
  - OA45786
    - Adds AD/ND/GD root page utilization statistics
    - Tells us how close we are to going into the next index level
  - OA46438
    - For corrupt PDSEs, counts mid-level empty index pages in all the indexes

PDSE Recoverability: The IEBPDSE Validation Tool

```
IGW700I PDSE Directory Validation Successful
DSN:XXXXXXX.TESTPDSE.WITHGENS
ADPages:1 IXRecords:41
ADPagesInCore:1 ADPagesRead:0
ADTreeLevels:1
NDPages:1 IXRecords:6
NDPagesInCore:1 NDPagesRead:0
NDTreeLevels:1
AD ND Tree Nodes:6
ADPercentFree:48 NDPercentFree:94
ADRootPercentFree:48 NDRootPercentFree:89
GDPages:1 IXRecords:4
GDPagesInCore:1 GDPagesRead:0
GDTreeLevels:1
GDRootPercentFree:90
Version:2
```

OA40492

OA45786

Converting from PDS to PDSE

Considerations When Converting: PDS to PDSE

- **PDS:**
  - Alphabetically organized linear directory consisting of member names and pointers to those members
  - Sequential areas of the data set are used for member data and not reused
- **PDSE:**
  - Collection of 4K pages. Both directory pages and member data
  - Multiple tree style directories pointing to linear data spaces for member data

Considerations When Converting: Load Modules and Program Objects

- Load modules can always be converted to Program Objects
- Program Objects may not always be able to be converted to Load Modules
  - Due to Program Object features that are unsupported by Load Modules
- Conversions are automatically done by IEBCOPY
- The conversion process will take additional processing due to Program Objects requiring a pass through the binder
- Cannot compare Load Module size to Program Object Size

Considerations When Converting: PDS to PDSE IEBCOPY

- COBOL V5’s binder output is a Program Object
- This means that it must reside in a PDSE

- Recent APAR to ease the transition:
  - OA46499: Prevents FAMS from failing an IEBCOPY on a corrupted PDS member

Considerations When Converting: PDSE's, COBOL v5, HIPERSPACE, and you!

- Normally we'd expect our LLA managed program objects to be eligible to be cached via VLF
- COBOL V5 Program Objects use deferred segments
  - This makes them ineligible for caching in VLF even if otherwise cache-worthy*
  - In these cases LLA tells PDSE to cache the Program Object
  - For PDSE to cache the Program Object HIPERSPACE caching must be enabled
  - Without HIPERSPACE caching enabled COBOL V5 Program Objects WILL NOT BE CACHED*

* Until OA45127

Considerations When Converting: PDSE’s, COBOL v5, HIPERSPACE, and you!

- **Hiperspace Caching Overview:**
  - Technote

- HIPERSPACE users at 2.1 should pick up OA46328
  - Corrects a space utilization issue

Considerations When Converting: Linklist Considerations

- Replacing a PDS in linklist versus a PDSE
  - For PDS's the following will work:

```
SETPROG LINKLIST,UNALLOCATE
P LLA
RENAME YOUR.LINKLIST.DATASET to
YOUR.LINKLIST.DATASET.OLD
RENAME YOUR.LINKLIST.DATASET.NEW to
YOUR.LINKLIST.DATASET
SETPROG LINKLIST,ALLOCATE
S LLA,SUB=MSTR
```

- This will cause 0F4 ABENDs if attempted with a PDSE

Considerations When Converting: Linklist Considerations

- The correct way to update/replace a PDSE in linklist:
  - Cannot be updated while in an active LNKLST set
- To remove a data set from an active LNKLST set:

```
LNKLST DEFINE NAME(NEWLLSET) COPYFROM(OLDLLSET)
LNKLST DELETE NAME(NEWLLSET) DSNAME(data set.to.be.removed)
LNKLST ACTIVATE NAME(NEWLLSET)
LNKLST UPDATE JOB(*)
```

- Then re-add the updated data set
- This needs to be done on all LPARs sharing the linklisted PDSE

PDSE and z/OS 2.1

PDSE and z/OS 2.1

- Enhancements, big and small
  - IMF/BMF restructure
  - PDSE member size limitation easing
- New Features
  - The PDSE V2 Format
  - PDSE V2 partial release enhancement
  - PDSE Member Generations

IMF/BMF Code Restructure: Rationale

- Current Issues/Compromises/Reasons for the restructure:
  - The PDSE index manager facility (IMF) is a heavy user of CPU time
  - Inefficient use of storage during PDSE update processing
  - Performing an IEBCOPY may result in excessive unused space in the Attribute Directory (AD)

IMF/BMF Code Restructure:
IMF Code Changes

- Majority of index processing now done inline
  - Greatly reduced code base size
  - Shortened path length for index operations
  - Most remaining IMF processing has been moved into CDM
- IMF LRU (Least Recently Used) task
  - Largely supplants BMF LRU
  - NEW index page LRU algorithm
  - Much more efficient tracking of directory page lifespans
    - Reduced CPU usage as well

IMF/BMF Code Restructure: IMF Code Changes cont.

- Combined previously separate PDSE update and Directory page cache components
  - Created entirely new Directory Store component
    - Handles caching of directory pages as well as update processing
    - Accesses directory pages more efficiently
  - Allows for more efficient re-use of pages during index updates
  - Reduced index page splits control index growth

IMF/BMF Code Restructure: Results

- Reduced CPU utilization for the most common PDSE directory processing
  - Large PDSE datasets (>1000 Members) benefit most
  - Improved index update process
- Improved efficiency of storage utilization during PDSE update processing
- Tighter indexes resulting in fewer overall index pages for both V1 and V2 PDSEs
- IEBPDSE is enhanced
  - Recommend running IEBPDSE from the the highest level system available
  - Due to the IMF rewrite, do not expect IEBPDSE at 2.1 to return identical results to 1.13

Easing PDSE Member Size Limitation : Rationale

- **Current Limitations:**
  - PDSE members can have a maximum of 15,728,639 records
  - PDS datasets are not bound by this limit
- **Why it Matters:**
  - Current member limit can make it unfeasible to use PDSE datasets in some situations
  - Customers can’t take advantage of PDSE’s buffering and indexing benefits

- *SHARE Requirement:* **SSMVSS11010**

Easing PDSE Member Size Limitation : Solution

- **New Enhancement:**
  - Increase the PDSE member size limit
  - New limit is over 125 times larger at approximately 2,146,435,071 records
  - New limit is substantially larger than the maximum supported size of a PDS member
  - Anything that will fit in a PDS member will now fit in a PDSE member

  - Applies to **BOTH** PDSE formats, V1 & V2
  - See **BLOCKTOKENSIZE LARGE** documentation for usage details

z/OS 2.1: PDSE Version 2

- At V2R1 there are 2 data set formats V1 and V2 PDSEs
- The version 1 format is the historic PDSE format
- The version 2 format is a revision of the PDSE format
  - Brings better performance and efficiency
    - Removed unneeded structures
    - Reworked variable record member indexing
  - Reduces CPU and Storage utilization

PDSE Version 2 : Features

- Version 2 data sets use the same serialization and buffering subsystems as version 1
- The IMF/BMF performance enhancements at V2R1 apply to BOTH V1 and V2 datasets
- Version 2 datasets increase the chance that partial release will be able to release space
- Supports PDSE Member Generations

PDSE Version 2 : Performance Benefits

- Real world improvements:
  - First OPEN of large PDSEs
  - Creation of large members using variable records
  - Variable records use storage much more efficiently
  - Variable records are much faster in the vast majority of use cases
- However:
  - For PDSE’s with variable records, an OPEN followed by a ‘blind’ Point to the end of a member will be slower
    - If this is your primary use for a PDSE then consider using a V1 data set

Questions? Comments?

Please Fill Out the Survey!

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Appendix

- Cheat Sheets, Parameters, Commands and JCL

NORMAL and EXTENDED Sharing Mode: Cheat Sheet

- Normal Sharing is the Legacy PDSE Sharing mode
  - It provides the ability to share at the *dataset* level between systems.
    - It can share at a *member* level only within a single system.
  - It can only be implemented with the Non-Restartable Address Space (SMSPDSE).
    - If you wish to use the restartable Address Space you **must** use Extended Sharing.
  - Getting started with Normal Sharing
    - PDSESHARING(NORMAL) must be specified in the IGDSMSxx member.
    - In order to change from a Extended Sharing Mode to Normal Sharing Mode, you **must** IPL.
  - Normal Sharing Mode is <u>not</u> limited to systems in the same SYSPLEX.
    - However, all participating systems <u>must</u> belong to the same GRSPLEX

- Extended Sharing is the preferred method of sharing
  - It provides the ability to share at the *member* level between systems.
  - It works with either and/or both of the SMSPDSE Address Spaces active.
  - Getting started with Extended Sharing
    - PDSESHARING(EXTENDED) must be specified in the IGDSMSxx member.
    - The SYSPLEX sharing type is determined by the <u>first</u> PDSE Address Space to start.
    - IPL is recommended to start Extended Sharing.
      - ACTIVATE Command can be used, but may cause PDSE problems.
  - Extended Sharing is strictly limited to systems within the same SYSPLEX.
    - All participating systems must belong to the same GRSPLEX **AND** XCFPLEX

Common Pitfalls of Sharing: Cheat Sheet

- Sharing a PDSE outside of the XCFPLEX
  - By sharing a PDSE outside of the XCFPLEX in Extended Sharing, you are introducing unpredictable problems.
  - Corruption can cause 0F4 ABENDs
    - Varied symptoms make improper sharing harder to diagnose
  - Improper sharing can result in <u>unserialized</u> access to datasets.
    - There is **no** warning that a dataset has been accessed in this manner.
    - Potential issues:
      - Invalid index data in-core, Corrupt dataset on DASD, corrupt member data, mismatched extent data, or even nothing at all.
  - There is no sure fire way to circumvent Extended Sharing Mode's serialization requirements.
    - PDSE datasets cannot be serialized by 3rd party products.
    - Asking users not to update PDSEs from outside SYSPLEX
      - Too hard to enforce, inevitably someone forgets, new users may no know all the rules
    - Reserves can cause serialization deadlocks.

Appendix:
Parameters, Commands and JCL

- PDSE Console Dump Parameters

```
COMM=(PDSE PROBLEM)
 JOBNAME=(*MASTER*,SMSPDSE*),
 SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT,COUPLE
 ,XESDATA),END
```

- IGDSMSxx Parameters:
  - SMSPDSE1 restartable address space:

```
PDSE_RESTARTABLE_AS(NO | YES)
```

  - PDSE Sharing Modes:

```
PDSESHARING(EXTENDED|NORMAL)
```

  - PDSE Member Generations Installation Limit

```
MAXGENS_LIMIT=n
```

Appendix:
Parameters, Commands and JCL

- PDSE Console Commands
  - SMSPDSE1 Restart Command

```
V SMS,PDSE1,RESTART
  [,QUIESCE(duration | 15 )[,COMMONPOOLS(NEW|REUSE) ]
```

  - SMSPDSE1 Activate Command

```
V SMS,PDSE1,ACTIVATE
```

  - PDSE Analysis Command

```
V SMS,PDSE(1),ANALYSIS
```

  - PDSE Freelatch Command

```
V SMS,PDSE|PDSE1,FREELATCH(<latch address>,asid,tcb)]
```

Appendix:
Parameters, Commands and JCL

- IEBPDSE JCL (1.13 and above only)

```
//VALIDATE EXEC PGM=IEBPDSE
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSLIB DD DISP=SHR,DSN=INPUT.PDSE.BAD
```

Appendix:
Parameters, Commands and JCL

- DSS PHYSICAL dump JCL

```
//DUMP      EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=*
//OUT      DD   UNIT=3390,
//              VOL=SER=xxxxxx,
//              DISP=(NEW,KEEP),
//              SPACE=(CYL,(100,100)),
//              DSN=hilev.DSSDUMP,
//              DCB=BLKSIZE=32760                          /
/SYSIN    DD   *
DUMP  PIDY(vvvvvv) -
    OUTDD(OUT)  -
     DATASET(INCLUDE(pdse.dataset.name)) -
     ALLDATA(  *  )                                       /
*
```

Appendix: SMSPDSE1 Restart Message Sequence

```
V SMS,PDSE1,RESTART
IGW036I VARY SMS,PDSE1,RESTART COMMAND ACCEPTED.
IGW057I WAITING FOR SMSPDSE1 SHUTDOWN.
IGW055I SMSPDSE1 SHUTDOWN IN PROGRESS.
IGW999I XQUIESCE Started
IGW062I SMSPDSE1 IS QUIESCING.
IGW064I SMSPDSE1 IGNORING IN-PROGRESS TASK 001B:MHLRES2B, TCB@=007DEC4 8.
*169 IGW074D SMSPDSE1 QUIESCE FAILED. RETRY? (Y/N)

 R 169,N
IEE600I REPLY TO 169 IS;N
IGW065I SMSPDSE1 QUIESCE COMPLETE.
IGW058I SMSPDSE1 SHUTDOWN COMPLETE.
IGW059I SMSPDSE1 IS BEING ACTIVATED.
IGW040I PDSE IGWLGEDC Connected
IGW040I PDSE Connecting to XCF for Signaling
IGW040I PDSE Connected to XCF for Signaling
IGW040I PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 040
 ++ INVOCATION INTERVAL:60 SECONDS
 ++ SAMPLE DURATION:15 SECONDS
IGW061I SMSPDSE1 INITIALIZATION COMPLETE.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
IGW999I XQUIESCE Stopping
IGW999I Reconnect Completed Normally
```