The Future of PDSE: The Version 2 Format

Speaker: Thomas Reed
IBM Corporation
Session: 15083
Agenda

• The PDSE Version 2 Rationale
• Version 2 Architecture Changes
• Performance Improvements
• New Feature: PDSE Member Generations
  • What is it?
  • Generations Structure
  • Working with Generations
• Version 2 Usage and Considerations
What is a PDSE?

• PDSE: Partitioned Data Set Extended
• A PDSE is a collection of directory and data pages
• At V2R1 there are 2 dataset formats V1 and V2 PDSEs
• PDSE server consists of one or two address spaces (SMSPDSE and SMSPDSE1)
• The SMSPDSE(1) address spaces serve client access requests for PDSE datasets
• Under the hood SMSPDSE(1) also manages PDSE serialization and buffering
PDSE User Needs

• Users want to be able to better reclaim space from PDSE datasets that is allocated but unused
• Users want to reduce PDSE I/O usage
• Users want to reduce PDSE CPU usage
PDSE Version 2 Format: Rationale

- **Streamlining** of the PDSE format
- Enables multiple improvements over Version 1
  - Enhanced Partial Release
  - Consolidation of directory pages
  - Enhanced read performance
  - Reduced virtual storage utilization
The more things change... the more they stay the same

• Like Version 1, Version 2 datasets:
  • Still are homogenous collections of 4K pages
  • Still have multiple indexes
  • Are serialized identically
  • Retain the same sharing capabilities and restrictions
  • Leverage the same V2R1 IMF/BMF restructure enhancements
Streamlining the PDSE Format

- Removal of unnecessary index structures
  - Removed VDF AD mapping
  - Removes a layer of complexity from page resolution
  - Allows for faster index searches
  - Allows for finer control of partial release
Streamlining the PDSE Format

- Set commonly referenced dataset statistics as easily referenced values
  - Page, Member, and Total Member Count values are now stored in the AD root
  - No longer dynamically calculated
  - Speeds up queries
Streamlining the PDSE Format

- Variable Record PDSE efficiency enhancements
  - Removed the static RRI
  - RRI now built dynamically
  - Drastically reduces storage and CPU needs

- The Tradeoff
  - 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
Performance Benefits

• Enhancements will benefit the majority of processing based on:
  • Directory consolidation (especially VB data sets)
  • Improved space management
  • Reduced path length for almost all index operations
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
  - Reduced I/O usage
  - Reduced CPU usage
Performance Results

- Testing Configuration
  - 2 LPARs at V2R1, 7 processors each
  - SMS Parameters:
    - PDSESHARING(EXTENDED)
    - PDSE_RESTARTABLE_AS(YES)
    - PDSE_BUFFER_BEYOND_CLOSE(YES) AND PDSE1_BUFFER_BEYOND_CLOSE(YES)
    - PDSE_BMFTIME(300) AND PDSE1_BMFTIME(300)
Performance Results

• Testing Workload
  • 400 users split evenly between the LPARs
  • 30 large PDSE datasets
    • 10 with RECFM=FB, LRECL 256 and over 13,000 members
    • 10 with RECFM=VB, LRECL=133 and over 13,000 members
    • 10 with RECFM=U and about 15,000 members and 4,000 alias entries
  • TSO workload includes READ, UPDATE, IEBCOPY, CREATE, and DELETE of members
  • Comparing PDSE V1 and V2 performance at V2R1
    • Meaning both dataset types are using the IMF/BMF improvements

NOTE: Performance improvements are based on internal IBM laboratory tests. Your results will vary.
Performance Results

- Improvements between V1 and V2 PDSE datasets:
  - 11-18% Reduction in storage used
  - 9% Reduction in CPU used by SMSPDSE1
  - 2% Reduction in CPU used by TSO users

- Improvements in index heavy operations
  - Browse dataset to member list - 7% faster
  - Member delete to member list – 20% faster

NOTE: Performance improvements are based on internal IBM laboratory tests. Your results will vary.
New Feature: PDSE Member Generations

• Implemented via APAR OA42358
• Exclusive to the V2 PDSE Format
• PDSE Datasets can now retain multiple generations of members
• Applies to BOTH Data Members and Program Objects
• Retains generations up to the dataset/system limit
New Feature: PDSE Member Generations

Terminology

• Generation (GEN)
  • A prior copy of a member

• Primary Generation
  • The current member
  • Absolute and Relative 0

• Generation Numbering
  • Absolute: GEN(n), GEN(n-1), GEN(n-2)….  
  • Relative: GEN(-1), GEN(-2),….,GEN(-n)
    • n being the nth generation created
PDSE Member Generations

- FIFO (First In, First Out) structure
  - Mostly….
- Generations are uniquely numbered
  - They can be referenced either by their **Absolute** or **Relative** generation
  - Current member is always 0, both relative and absolute
  - Greatest number indicates the newest generation
PDSE Member Generations

- FIFO (First In, First Out) structure
  - Oldest generation is permanently deleted if it’s over the generation limit
  - Old generations generally behave just like primary members
  - Aliases are retained for previous generations*

* When STOW RECOVERG is used

Complete your session evaluations online at www.SHARE.org/Anaheim-Eval
PDSE Member Generations

Usage Considerations

• Allow extra space for each generation
• Each generation retains the entire member
• MAXGENS_LIMIT in IGDSMSxx is the System limit
• MAXGENS_LIMIT can be set dynamically
• MAXGENS_LIMIT is set at 2 billion

Complete your session evaluations online at www.SHARE.org/Anaheim-Eval
PDSE Member Generations: Working with Generations

Creating a Generation

- 2 requirements
  - (LIBRARY,2)
  - MAXGENS > 0
- New generations are automatically created on replace or delete of a member
- Update in place will not create a new generation
- Generation creation is atomic
PDSE Member Generations: Working with Generations

Reading Old Generations
- FIND macro will allow programs to connect to old generations
- Conventional READ and CHECK macros still apply
- Old generations cannot be accessed via JCL or dynamic allocation
PDSE Member Generations: Working with Generations

Deleting Old Generations
• Each generation must be deleted separately
• Deleted generations can be replaced by using STOW RG
• ISPF member delete will delete all generations
PDSE Member Generations: Working with Generations

Recovering Old Generations

• Read an old generation and then write it to either the same or a different member name
  • The old generation will become the current generation
  • Note: This method will not restore aliases

• Use the RECOVERG option for the STOW macro
  • The old generation becomes the current generation of the member of the same name
  • Note: Aliases ARE recovered by this method
PDSE Member Generations: Working with Generations

Backup Considerations

- **IEBCOPY and IDCAMS REPRO**
  - Only copy the current generation of each member
  - All old generations are lost

- **DFSMSdss**
  - Physical or Logical dump and restore retain all old generations
  - This includes HSM backup
PDSE Member Generations: DESERV Macros

FUNC=GET_G (AKA Get Generation)
• Returns information for the selected generation
• Returns the same information as GET plus the relative and absolute generation numbers
• A dummy entry is returned if the selected generation does not exist
• Does not support CONNECT
PDSE Member Generations: DESERV Macros

FUNC=GET_G
 ,AREA=(buffer_area, buffer_area_size)
 ,DCB=data_control_block
 ,NAME_LIST=(generationname,1)
 [,MF={((E,parmlist_name [,NOCHECK|COMPLETE]])|S}]
 [,RETCODE=return_code]
 [,RSNCODE=reason_code]
PDSE Member Generations: DESERV Macros

FUNC=GET_ALL_G (AKA Get All Generations)
- Returns information for the selected generation for all members
- Returns the same information as GET_ALL plus the relative and absolute generation numbers
- A dummy entry is returned if the selected generation does not exist for a member
- Does not support all the same options as GET_ALL
PDSE Member Generations: DESERV Macros

FUNC=GET_ALL_G
  ,AREA=(buffer_area, buffer_area_size)
  ,DCB=data_control_block
  ,NAME_LIST=(generationname,1)
  [,MF={{E,parmlist_name[,NOCHECK|COMPLETE]|S}}]
  [,RETCODE=return_code]
  [,RSNPCODE=reason_code]
PDSE Member Generations: STOW Macro

**DG** (Delete Generation)
- Deletes an existing generation
- Takes a member name and generation number
- Leaves a gap in the generation list
- If issued with a generation of 0, deletes the member without creating a generation
PDSE Member Generations: STOW Macro

RG (Replace Generation)
• Replaces an existing generation
• Adds a generation if replacing a gap in the generation list
PDSE Member Generations: STOW Macro

RECOVERG (Recover Generation)
• Recovers an existing generation
• Removes the selected generation from the generation list and makes it the primary member
• Creates a new generation in the replace process from the former primary member
Panels

- ISPDF now has generations support
- Enhanced member list option must be selected
PDSE Member Generations: ISPF Support

Allocation

- Allocates like any other PDSE
- MAXGENS must be >0
- Be sure you’re using version 2!

```
Directory blocks : 0  (Zero for sequential data set) *
Record format : FB
Record length : 80
Block size : 27200
Data set name type : LIBRARY  (LIBRARY, HFS, PDS, LARGE, Basic, *
Data set version : 2  EXTREQ, EXTREF or blank)
Num of generations : 50  (NO, OPT or blank)
Extended Attributes ______
Expiration date : _____  (YY/MM/DD, YYYY/MM/DD
Enter "/" to select option  YY.DDD, YYYY.DDD in Julian form
```
PDSE Member Generations: ISPF Support

Restrictions

• ENQUEUEing on one generation applies to all generations of that member
  • This is not a PDSE serialization restriction
  • The native API’s allow for editing of multiple generations of the same member
• ISPF Options 1 and 2 do not support a GEN parameter
• ISPF 3.1 and 3.4 do support a GEN parameter
PDSE Member Generations: ISPF Support

Editing

• Editing the current member (GEN 0) results in a new generation being created
• Editing prior generations does NOT result in a new member
• Supports referencing generations by either absolute or relative generation number
• Deleting a member in ISPF deletes all generations
  • This is an ISPF implementation feature
  • TSO DELETE pdse(member) deletes only the primary
PDSE Member Generations: ISPF Support

Editing Cont’d

- Generation creation behavior can be forced
  - SAVE NEWGEN – Creates a new generation
  - SAVE NOGEN – Does not create a new generation
- Edit will tell you which absolute generation you are working with
How to create Version 2 PDSEs

• New option for DSNTYPE keyword
  • DSNTYPE=(LIBRARY,{1,2})
    • 1 – Version 1 PDSE (Default)
    • 2 – Version 2 PDSE
    • Supported for JCL, TSO Allocate

• New options for IGDSMSxx member in SYS1.PARMLIB
  • DSNTYPE=({LIBRARY|PDS|HFS},{1,2})
  • MAXGENS_LIMIT (1 – 2bn)
• Precedence:
  • DSNTYPE on JCL takes precedence over PARMLIB
Usage Expectations

• Long Term
  • It is expected that PDSE users will specify DSNTYPE=(LIBRARY,2) in their IGDSMSxx parmlib member
  • It is expected that V2 data sets will eventually supplant V1 data sets
• The following usage considerations are applicable for mixed PDSE V1 and V2 environments
How to differentiate PDSE versions

- ISMF
  - Dataset List: Version added to data under column ‘DATA SET NAME TYPE’

- ISITMGD
  - New field added: ISMDSNVER

- SMF Type 14/15
  - New field added: SMF14DSVER
How to differentiate PDSE versions: ISMF

- Dataset List Example

<table>
<thead>
<tr>
<th>LINE</th>
<th>DATA SET NAME</th>
<th>DATA SET</th>
<th>NUM OF</th>
<th>ENTRY</th>
<th>REBLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DGTLGP13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>FILTERED LIST</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>ENTRIES HIDDEN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entry Line Operators below:

SYS1.LINKLIB
SYS1.LINKLIB.PDSE0
SYS1.LPALIB
SYS1.LINKLIB.PDSE2

Version displayed with data set type
How to differentiate PDSE versions cont.

• Note:
  • Neither IEHLIST LISTVTOC nor LISTPDS can be used to identify Version 2 PDSE data sets
  • No VTOC bit is set for Version 2 data sets

• PDSE data set versions are internally self describing
Coexistence

- Coexistence APARs:
  - OA39530
  - OA40844
  - OA41790

- Down-level systems (z/OS V1R12 and V1R13)
  - Coexistence APARs allow for access to PDSE Version 2 datasets
  - PDSE Version 2 data sets **cannot** be created below V2R1
Diagnostics

- Existing diagnostics updated to support PDSE Version 2 data sets
  - IEBPDSE
  - IGWFPMAN
  - IGWPIT

- Coexistence APARs are required for compatibility
Unsupported Releases

- Attempting to open a V2 data set on a pre-V1R12 system will result in a 0F4 ABEND
  - ABEND 0F4 RC=24 RSN=01045AF1
  - Reason Code 01045AF1 translates to: JCDM_INVALID_VDF
- PDSE Connect Processing will fail on initial page load checks
  - Prevents invalid data set information from being returned to the client
  - Prevents any processing that could break or corrupt the Version 2 PDSE from occurring
Rate this Session

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