#SHAREorg

SHARE
Technology · Connections · Results

The Future of PDSE: New Features in z/OS 2.1

Speaker: Thomas Reed /IBM Corporation

SHARE Boston 2013

Session:14147

SHARE in Boston

Agenda

2.1 PDSE Major Topics

- IMF/BMF Code Restructure
- Easing the Member Size Limitation
- PDSE Member Generations
- Version 2 PDSE Format

What is a PDSE?

- PDSE: Partitioned DataSet 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

z/OS V2.1 PDSE Enhancements:

IMF/BMF Code Restructure

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: Solutions

- Improvements Implemented:
  - Shortened code paths
  - Enhance buffer caching efficiency
  - Enhance buffered page lookup
  - Enhance update page re-use

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

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

IMF/BMF Code Restructure: Performance Results

- TSO Performance Workload: V1R13 vs V2R1
  - 300 Concurrent Users accessing 30 large PDSEs
    - Mixture of RECFMs FB, VB, U
    - Each PDSE contained over 10,000 members
    - Varied sizes for each member

IMF/BMF Code Restructure: Performance Results

- Distinct improvements from V1R13 to V2R1
  - 87% CPU reduction and 87% storage reduction in the SMSPDSE address space
  - 55% CPU reduction and 43% storage reduction in the TSO User address spaces
- Significant reduction in CPU and Storage usage for index processing

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

IMF/BMF Code Restructure: Performance Results

- TSO Response Time Improvements
  - Enter browse, view member list
    - 60% improvement due to improved index processing
  - Edit existing member
    - 30% improvement
  - Edit new member
    - 30% improvement

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

IMF/BMF Code Restructure: Customer Perspective

- Enhancements require NO external changes to exploit
- Enhancements make NO changes to the existing PDSE data set structure
- Enhancements are compatible with ALL PDSE data sets
- Enhancements have NO coexistence considerations

- Improvements
  - Reduced CPU utilization
  - Reduced storage utilization for updates
  - More efficient storage management
  - Reduced I/O due to improved directory processing

z/OS V2.1 PDSE Enhancements:

Easing PDSE Member Size Limitation

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 are forced to use multiple large sequential datasets
  - 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
- **Implementation:**
  - Changes were made in BAM to increase the record limit for PDSE members if **BLOCKTOKENSIZE LARGE** is specified

Easing PDSE Member Size Limitation : How to Exploit the Enhancement with BSAM/BPAM

- Accessing Records Beyond the Current Limit:
  - A program can WRITE records beyond the current limit under the following conditions
    - DSORG=PS
      - *DCB MACRF is W (not WP) <WP: WRITE with Note/Point>*
        - *Or*
      - *DCB MACRF is WP and DCBE BLOCKTOKENSIZE=LARGE**
    - DSORG=PO
      - *DCBE BLOCKTOKENSIZE=LARGE**
  - (*) if BLOCKTOKENSIZE=LARGE is not specified, a WRITE beyond 15,728,639 will result in an ABEND S002-A8
    - NOTE: BLOCKTOKENSIZE=LARGE indicates a program supports a 32-bit Note token

Easing PDSE Member Size Limitation : How to Exploit the Enhancement with BSAM/BPAM cont.

- A program can READ records beyond the current limit in all cases
  - Works regardless of DSORG or MACRF
- A program can NOTE records beyond the current limit under the following conditions:
  - DCBE BLOCKTOKENSIZE=LARGE*
- (*) if BLOCKTOKENSIZE=LARGE is not specified, a NOTE beyond 15,728,639 will result in an ABEND S002-A8

Easing PDSE Member Size Limitation : How to Exploit the Enhancement with QSAM.

- Any QSAM program can get/put records beyond the current limit in all cases

Easing PDSE Member Size Limitation : Coexistence with V1R12 and V1R13

- Issuing a WRITE/PUT beyond record 15,728,639 will result in the existing ABEND S002-A8
- Issuing a READ/GET beyond record 15,728,639 will be allowed

Easing PDSE Member Size Limitation : Coexistence with V1R12 and V1R13 cont.

- Issuing a NOTE beyond record 15,728,639 with DCBE BLOCKTOKENSIZE=LARGE will return a valid NOTE value
- Issuing a NOTE beyond record 15,728,639 without DCBE BLOCKTOKENSIZE=LARGE will result in an ABEND S002-A4
  - WITHOUT coexistence APAR applied results are unpredictable
- Issuing a POINT beyond record 15,728,639 with DCBE BLOCKTOKENSIZE=LARGE will result in out-of-extent on the next READ/WRITE

Easing PDSE Member Size Limitation : New NOTE/POINT Token Range

- **PDSE Directory**
  - X'00000001'
- **PDSE Members (MLT)**
  - X'00000002' to X'0007FFFF'
- **PDSE Record Numbers (RLT)**
  - X'00100001' to X'7FFFFFFF'

Easing PDSE Member Size Limitation : Changes to Messages and Codes

- **IEC036I 002-A8**
  - Existing Description:
    - “Maximum number of records in a PDSE member has been exceeded. Or maximum number of blocks allowed in an extended sequential data set has been exceeded.”
  - Additional Description Added:
    - “Or a Note was issued for a PDSE data member and the record is more than 15,728,639 records into the member and BLOCKTOKENSIZE=LARGE was not specified when the PDSE was opened. NOTE could not have returned a valid value.”

z/OS V2.1 PDSE Enhancements:

PDSE Member Generations (Sneak Peak!)

PDSE Member Generations : Sneak Peak

- **New Enhancement:**
  - When creating a PDSE Version 2 Dataset the user will be able to specify the number of generations that the system will retain for replaced members.
  - This means that it will become possible to recover previous versions of replaced PDSE members

PDSE Member Generations : Old Member Generations

- Accessing old member generations:
  - New DESERV macro options will reveal old member generations and their aliases
  - ISPF and IPT will provide information about old member generations and how to recover them
  - **Note:** *Neither TSO commands nor JCL can be used to reveal old generations*
- Old generations will not be visible to current APIs

- **Support for recovering prior levels of a PDSE member is planned to be made available with a PTF for APAR OA42358 in the first quarter of 2014.****
  - Additional PTFs will tolerate and ignore old generations.

**** IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.**

PDSE Member Generations : Features

- Applies to both SMS managed and non-SMS managed PDSE datasets
- Supports both data members and program objects
- Aliases for each member generation will be retained as well

PDSE Member Generations : Usage Considerations

- Each time a member is **replaced** (not updated in place) the replaced generation and its aliases are retained
  - When the generation limit is reached the oldest generation will be deleted permenantly
- PDSE Data sets utilizing member generations will require more space
- When allocating PDSE Data sets with generations:
  - MAXGENS keyword on the DD statement or dynamic allocation or Data class
  - PARMLIB will have a system limit on the generations limit

z/OS V2.1 PDSE Enhancements:

PDSE Version 2

PDSE Version 2 : 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 : Why Do We Need a Version 2?

- **Streamlining** of the PDSE format
- Externally identical to Version 1 data sets
- Enables multiple improvements over Version 1
  - Enhanced Partial Release
  - Consolidation of directory pages
  - Enhanced read performance
  - Reduced virtual storage utilization

- Leverages V2R1 IMF/BMF restructure enhancements

PDSE Version 2 : Enhanced Partial Release

- Version 2 datasets increase the chance that partial release will be able to release space
- When might space be released?
  - When SPACE=(,,RLSE) is specified in the DD statement on OUTPUT to a V2 PDSE
  - When space management is done by DFSMShsm
  - When space management is done by DFSMSdss
  - When the PARTREL macro is used

PDSE Version 2 : Performance Benefits

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

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

PDSE Version 2 : 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)

PDSE Version 2 : 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.

PDSE Version 2 : 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.

PDSE Version 2 :
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 option for IGDSMSxx member in SYS1.PARMLIB
  - DSNTYPE=({LIBRARY|PDS|HFS},{1,2})
- Precedence:
  - DSNTYPE on JCL takes precedence over PARMLIB

PDSE Version 2 : 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

PDSE Version 2 : How to create Version 2 PDSEs cont.

- LIKE=
  - If DSNTYPE was specified on the JCL but version was not, the value for version will be obtained from the IGDSMSxx member NOT from the LIKE= data set.
  - If DSNTYPE was not specified on the JCL, then the DSNTYPE value and version will be obtained from the LIKE= data set.
  - If DSNTYPE was not specified on the JCL and LIKE= was not specified then the DSNTYPE value will be obtained from the DATA CLASS and, if necessary, the version number will be obtained from the IGDSMSxx member.
  - If the DSNTYPE is not available from any source (JCL, LIKE= or DATA CLASS) then the default values for both DSNTYPE and the version (if applicable) will be picked up from the IGDSMSxx member.

PDSE Version 2 : How to create Version 2 PDSEs cont.

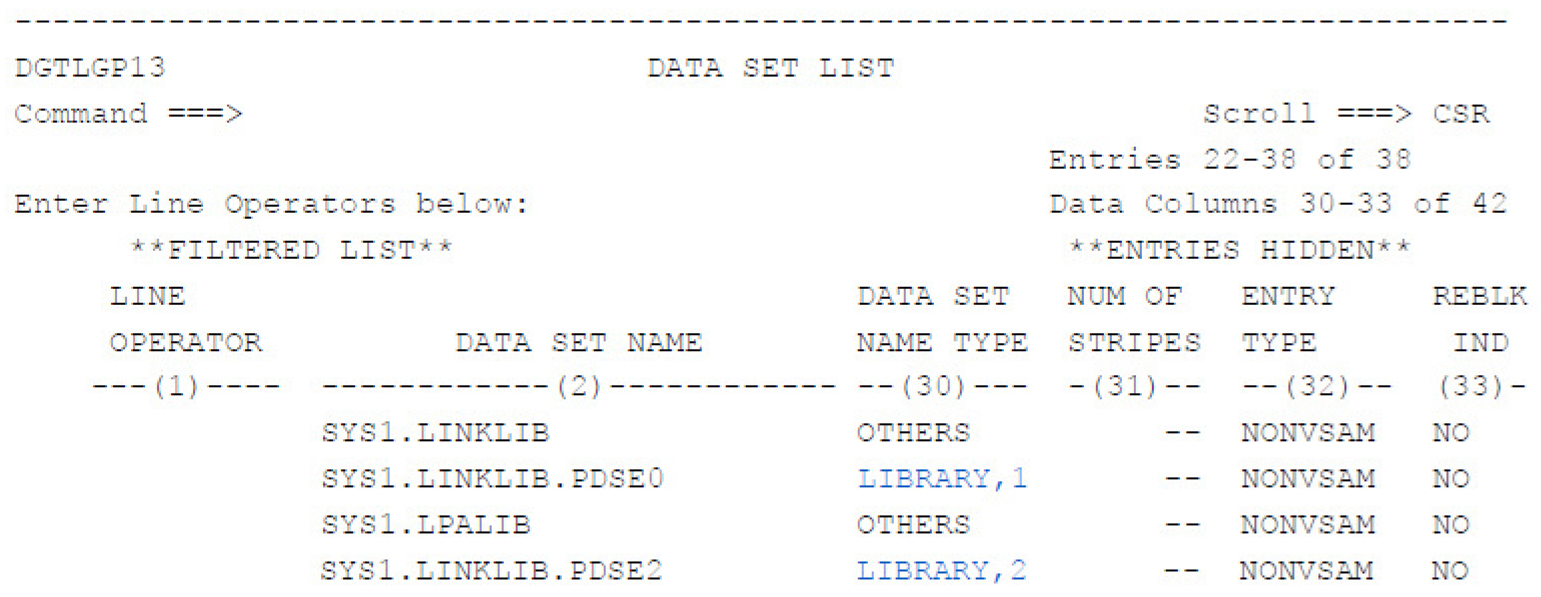
- REFDD=
  - Use the REFDD parameter to specify attributes for a new data set by copying attributes of a data set defined on an earlier DD statement in the same job.
  - One of the attributes which are copied to the new data set from the attributes specified on the referenced DD statement is DSNTYPE. DSN version will also be copied to the new data set from the referenced DD statement.

PDSE Version 2 : 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

PDSE Version 2 :
How to differentiate PDSE versions: ISMF

- Dataset List Example



```
 ------------------------------------------------------------------------------
 DGTLGP13                        DATA SET LIST
 Command ===>                                                     Scroll ===> CSR
                                                          Entries 22-38 of 38
 Enter Line Operators below:                              Data Columns 30-33 of 42
       **FILTERED LIST**                                   **ENTRIES HIDDEN**
      LINE                                           DATA SET   NUM OF   ENTRY     REBLK
      OPERATOR          DATA SET NAME                NAME TYPE  STRIPES  TYPE       IND
     ---(1)----  ------------(2)------------  --(30)---  -(31)--  --(32)--  (33)-
                 SYS1.LINKLIB                   OTHERS           --  NONVSAM   NO
                 SYS1.LINKLIB.PDSE0             LIBRARY,1        --  NONVSAM   NO
                 SYS1.LPALIB                    OTHERS           --  NONVSAM   NO
                 SYS1.LINKLIB.PDSE2             LIBRARY,2        --  NONVSAM   NO
```

- Version displayed with data set type

PDSE Version 2 : 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

PDSE Version 2 : 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

PDSE Version 2 : 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

PDSE Version 2 : Diagnostics

- Existing diagnostics updated to support PDSE Version 2 data sets
  - IEBPDSE
  - IGWFPMAN
  - IGWPIT
- Coexistence APARs are required for compatibility

Questions? Comments?

Please Fill Out the Survey!