#SHAREorg

SHARE
Technology · Connections · Results

PDSE Performance Basics

Speaker: Thomas Reed /IBM Corporation

SHARE San Francisco

Session:12982

SHARE in San Francisco 2013

Agenda

- System and SYSPLEX Level Performance
  - PDSE Sharing
  - PDSE Features
  - PDSE and Workload
  - PDSE Caching
- Dataset Level Performance
  - PDSE Usage Considerations
  - PDSE Common Situations

Basic Basics

- PDSE: Partitioned DataSet Extended
- PDSE server consists of one or two address spaces (SMSPDSE and SMSPDSE1)
- PDSEs are essentially a collection of named Members which contain sequential data
- PDSE datasets contain both Member Data and Directory Data in 4K pages
- Can contain normal data as well as Program Object data
- IBM ships code in Program Object format

System and SYSPLEX level performance

- PDSE Sharing
- PDSE Features
- PDSE and Workload
- PDSE Caching

System and SYSPLEX level performance: PDSE Sharing Modes

Extended Sharing

Normal Sharing

System and SYSPLEX level performance: PDSE Sharing Basics

- OPEN Types:
  - INPUT: Commonly known as a READ. Used to READ one or more Members
  - OUTPUT: Commonly known as WRITE. Used to create new Members, rename Members, delete Members, or replace Members.
  - UPDATE: Update in place of a member.
- Serialization Methods:
  - GRS (Global Resource Serialization)
  - XCF (Cross-system Coupling Facility) Messaging

System and SYSPLEX level performance: PDSE Sharing Basics

- PDSE Serialization Terms
  - **Latch**: Used to serialize internal structures used to provide access to a PDSE on a single system. Either HELD or FREE
  - **Enqueue (ENQ)**: Used to serialize PDSE datasets via GRS. Either FREE, SHARED (SHR), or EXCLUSIVE (EX)
  - **Lock**: A data structure used to provide serialization while accessing or updating a PDSE.

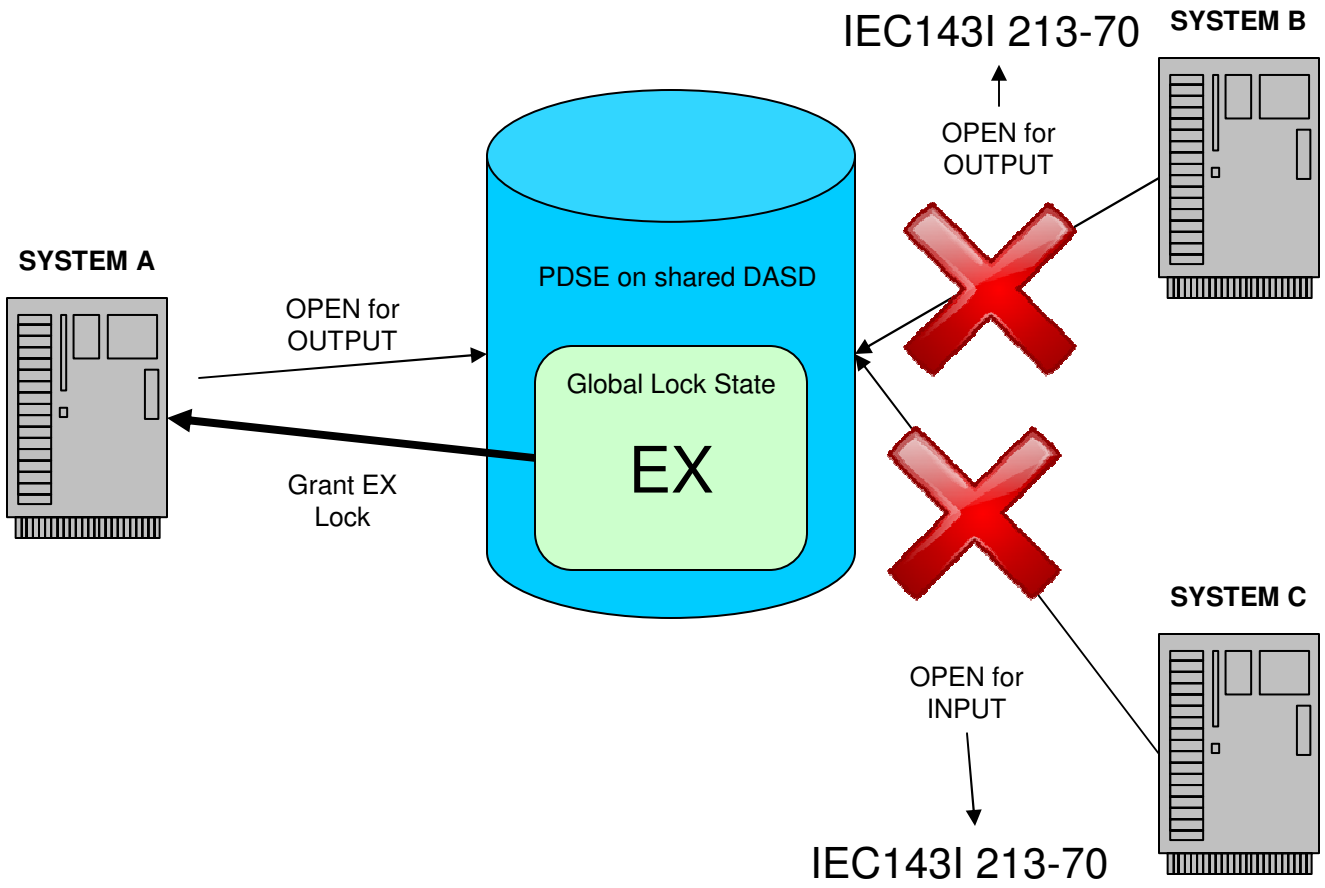
System and SYSPLEX level performance: PDSE Sharing Modes

- Extended Sharing
  - PDSESHARING(EXTENDED) in IGDSMSxx
  - Enables both dataset level and member level sharing between systems
  - Utilizes both XCF and GRS for serialization
- Normal Sharing
  - PDSESHARING(NORMAL) in IGDSMSxx
  - Only supports dataset level sharing between systems
  - Utilizes only GRS for serialization

System and SYSPLEX level performance: PDSE Sharing Modes

- Performance Implications of Sharing Modes
  - NORMAL sharing:
    - PDSE datasets can be locked either Only Readers (OR) or Exclusive (EX) by the system
    - OR allows the system to perform any INPUT operation but no OUTPUT
    - EX allows the system to perform any INPUT or OUTPUT operation but is the ONLY system which may access the PDSE
  - Leads to one system being able to block all other systems in the sysplex when doing OUTPUT

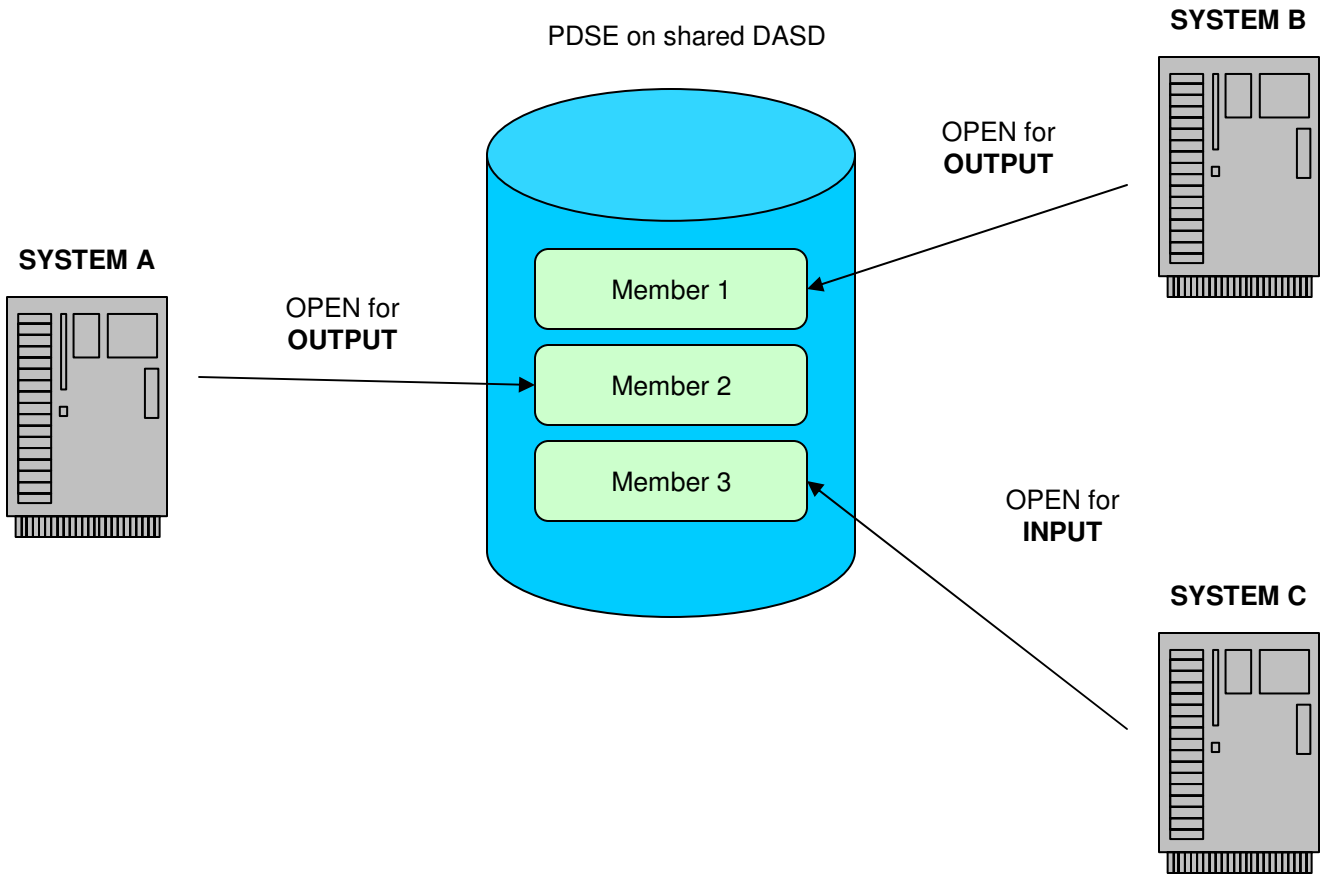
System and SYSPLEX level performance: NORMAL Mode Sharing



System and SYSPLEX level performance: PDSE Sharing Modes

- Performance Implications of Sharing Modes
  - EXTENDED sharing
    - Allows multiple systems to open the dataset for both INPUT and OUTPUT
    - Opens are only blocked in cases where an UPDATE would cause data integrity issues

System and SYSPLEX level performance: EXTENDED Mode Sharing



Sharing Modes Recap

- NORMAL Mode
  - Dataset level sharing only between systems
  - One system can block others in the GRSPLEX by holding the EX ENQ
- EXTENDED Mode
  - Member level sharing between systems
  - All systems in the SYSPLEX can access members concurrently so long as the member is not open for update

System and SYSPLEX level performance: PDSE Features

Fairness

Buffer Beyond Close (BBC)

System and SYSPLEX level performance: Fairness

- New internal feature at 1.13
- At 1.12 and below PDSE locking allows for readers to lock out writers.
  - Multiple readers waiting on the lock can acquire the lock before the writer can acquire the EX lock
  - Use of conditional ENQ means that each time the ENQ is released waiters all race for the ENQ.
  - EX lock requests need to acquire 2 enqueues versus only 1 for OR lock
- At 1.13 fairness allows writers to “get their foot in the door”
  - Change to acquire ENQ unconditionally
  - All waiters have to “get in line”

System and SYSPLEX level performance: Buffer Beyond Close

- IGDSMSxx Parameter
  - PDSE_BUFFER_BEYOND_CLOSE(YES|NO)
  - PDSE1_BUFFER_BEYOND_CLOSE(YES|NO)
- What does Buffer Beyond Close do?
  - When DISABLED, the last close discards all buffered pages
  - When ENABLED, buffered pages are retained until discarded by LRU processing
- Performance Implications
  - Can potentially significantly reduce I/O
  - Enhances performance in cases where PDSE is frequently opened and closed.
  - Can increase overall storage utilization in some situations

PDSE Features Recap

- Fairness
  - 1.13 and above only. Internal code change
  - Changes ENQ acquisition to Unconditional
  - Ensures writers are not locked out by readers
- Buffer Beyond Close
  - Enabled via IGDSMSxx parameter for each address space
  - Retains buffered pages after last close
  - Speeds up reopening PDSEs

System and SYSPLEX level performance: PDSE and Workload

Blocked Workload

Storage Usage

System and SYSPLEX level performance: PDSE and Workload

- PDSE and the system interact in terms of workload
- PDSE can be impacted by available CPU resources
- PDSE can impact the system through it's use of storage

System and SYSPLEX level performance: Blocked Workload

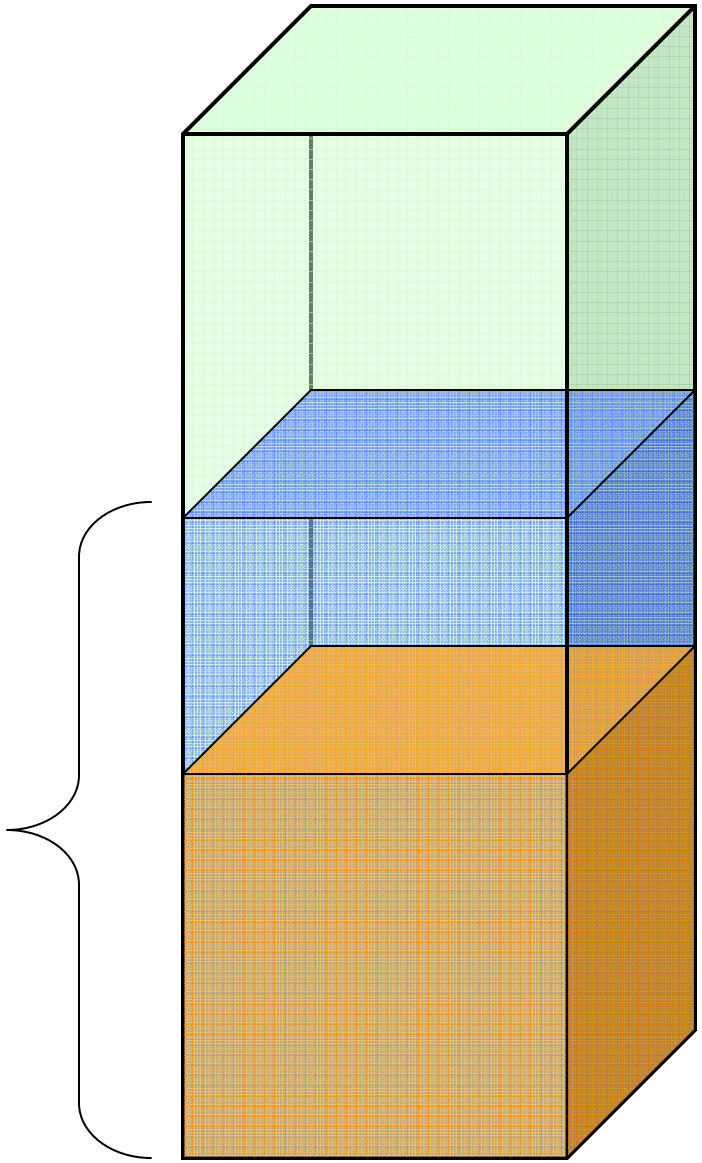
- Blocked Workload describes when lower priority tasks hold resources but do not get sufficient CPU time to complete their work and then release those resources which are in turn being waited on by higher priority tasks.
- Seen in CPU constrained environments.
- Symptoms are similar to a latch hang.
  - IGW038A messages indicating latch contention
  - Generally clears once CPU load decreases
  - May also manifest as slow obtain and release of latches depending on CPU conditions

System and SYSPLEX level performance: Blocked Workload

- Solutions:
  - Implement IGDSMSxx parameter PDSE_SYSEVENT_DONTSWAP as YES
    - Prevents tasks processing in PDSE address spaces from swapping out until they have released any resources they hold
    - Issues SYSEVENT ENQHOLD in PDSE latch processing
    - Boosts the service class of the latch holder
  - Support provided in APARs OA32587, OA36650 and OA38704
  - Allow some CPU headroom

System and SYSPLEX level performance: Storage Usage

- The PDSE storage model is based on a high water mark allocation
  - High water mark represents the highest concurrent PDSE utilization of storage
  - The high water mark is raised and additional storage allocated as necessary if the current high water mark cannot fulfill the size of the request
  - Storage is re-used if possible under the high water mark
  - Storage is marked for re-use by the LRU component
  - The high water mark does not decrease.


Free Storage
High Water Mark
Allocated Storage
Utilized Storage

System and SYSPLEX level performance: Storage Usage

- Preventing over allocation of the high water mark
  - Limit the number of concurrent opens of large PDSEs in batch.
  - Use BLDL NOCONNECT to process large PDSE member lists
  - Use STOW DISCONNECT when done with processing a member to allow storage to be reused
  - Allow LRU to work by spreading out concurrent opens

System and SYSPLEX level performance: Storage Usage

- PDSE Storage Considerations
  - LRU operation can be tuned but consumes CPU cycles
  - PDSE[1]_DIRECTORY_STORAGE parameter can be tuned but we recommend keeping the 2GB default
  - PDSE has no hard limit on the storage it can use

PDSE and Workload Recap

- Blocked Workload
  - Seen in CPU constrained environments
  - Low priority work holds resources needed by higher priority work and can't get sufficient CPU to release the resource
  - IGDSMSxx parameter PDSE_SYSEVENT_DONTSWAP(YES) can help
- Storage Usage
  - PDSE storage management works on High Water Mark allocation system
  - Limit number of concurrent opens

System and SYSPLEX level performance: PDSE Caching

VLF and LLA

Hiperspace Caching

System and SYSPLEX level performance: PDSE Caching

- PDSEs reduce I/O and caching further helps that goal
- Two types of PDSE members and two types of caching

LLA and VLF	Hiperspace
Program Objects CLISTs REXX execs	Member Pages

System and SYSPLEX level performance: PDSE Caching

- VLF and LLA considerations
  - For best performance PDSE program objects need to be bound FETCHOPT=(PACK,PRIME) (AKA move mode loading)
  - Page mode bound program objects do not fully utilize the benefits of VLF.

System and SYSPLEX level performance: PDSE Caching

- PDSE Hiperspace Caching
  - Optional extension of PDSE buffer management processing
  - Controlled/enabled by HSP_SIZE parameter
  - LRUCYCLES and LRUTIME control lifespan of cached data
  - Cached PDSEs must have an MSR (Millisecond Response Time) of 1
  - Cached data is valid only so long as the dataset is open
  - Effective for datasets which are held open for long periods and read by multiple jobs

PDSE Caching Recap

- LLA and VLF
  - Caches PDSE Program Objects
  - PDSE program objects need to be bound for move mode loading
- HIPERSPACE
  - Extends PDSE buffering for member data
  - PDSE must have MSR of 1 to be eligible
  - Cached pages are valid while the PDSE is open

Dataset Level Performance

- PDSE Usage Considerations
- Pending Deletes
- PDSE Index Fragmentation
- Very Large PDSEs
- Heavily Accessed PDSEs

Dataset Level Performance: Usage Considerations

- Strengths of PDSE datasets
  - Index structure enables fast searching
  - Flexible organization
  - Member level sharing between systems
  - Caching reduces I/O for datasets that remain open
  - Enable the use of Program Objects

Dataset Level Performance: Usage Considerations

- Weaknesses of PDSE datasets
  - Open – Read – Close use cases
  - Higher storage usage to support caching and directory
  - Pending Deletes
  - Slower copies

Dataset Level Performance: Pending Deletes

- What is a Pending Delete?
  - When a member that is in-use is deleted
  - Member is no longer accessible but data and control blocks remain
  - Flagged for deletion during Pending Delete cleanup processing

Dataset Level Performance: Pending Deletes

- How do I know if I have an issue with Pending Deletes?
  - IGW01177T OUT OF SPACE CONDITION ENCOUNTERED DURING MEMBER CREATE
  - Growth in PDSEs utilization percentage without adding additional members
  - Utilization percent takes into account both valid member data and pending deletes
  - After OA40492, new directory utilization stats have been added to the IEBPDSE tool.
    - Pending deletes will show growth in AD utilization but not ND utilization

Dataset Level Performance: Pending Deletes

- How do I eliminate pending deletes?
  - 2 methods for eliminating pending deletes
    - 1) Pending Delete Cleanup Processing
      - *Close the dataset (on all sharing LPARS)*
      - *Open the dataset for OUTPUT. This must be the FIRST open for OUTPUT to the dataset.*
    - 2) IEBCOPY
      - *Copy the PDSE to a new dataset*
      - *Pending deletes will not be copied*

Dataset Level Performance: Pending Deletes

- How do I avoid creating Pending Deletes?
  - Depends on the use case for the dataset
  - Plan to CLOSE the dataset to allow cleanup to run or IEBCOPY it
  - Identify PDSEs in which large numbers of members are deleted frequently
  - Identify PDSEs which are rarely closed
  - Use BLDL NOCONNECT to process large member lists

Dataset Level Performance: Index Fragmentation

- Member inserts and deletes will fragment the index and members over time
- Eventually this will grow the size of the index
- Growth will slow as the index splits
- After OA40492, IEBPDSE can be used to monitor the number of entries per ND page
- When entries decrease to ~60 per ND page the PDSE is likely ready to be re-org'd
- IEBCOPY should be used to re-org the PDSE

Dataset Level Performance: Very Large PDSEs

- Theoretical limitation of 522,236 Members
- Practical limits are ~250K Members
- Performance depends on the number of members open simultaneously
- STOW DISCONNECT or BLDL NOCONNECT are critical
- Can cause large spikes in storage usage
- Greatest benefit for keeping the index fresh via re-org

Dataset Level Performance: Heavily Accessed PDSEs

- Greatest risk for causing PDSE performance issues
- HIPERSPACE Caching should be strongly considered
- If shared between systems, PDSE Sharing EXTENDED should be used
- If the PDSE is rarely closed Inserts and Deletes should be limited

Dataset Level Performance Recap

- Be mindful of PDSE use cases to take best advantage of PDSEs strengths
- Pending Deletes
  - Created by the deletion of an in-use member
  - Can be eliminated by Pending Delete Cleanup Processing or through using IEBCOPY
- Index Fragmentation
  - Member and Index pages become fragmented over time
  - PDSE can be re-org'd by IEBCOPY
- Heavily Accessed or Very Large PDSEs
  - Important to identify these datasets
  - Will see the most benefit from these performance tips

Questions? Comments?

Please Fill Out the Survey!