Session 18822: MQ Parallel Sysplex Exploitation, Getting the Best Availability from MQ on z/OS by using Shared Queues

Paul Kettley

PLM for Messaging on z

paulk@uk.ibm.com
Agenda

• Shared Queues

• Large messages with DB2

• SMDS

• Structures – Persistence and recovery

• Client Channels
Shared Queues
Shared Queues

Application

QMGR

Coupling Facility (CF)

Shared Queue

Application

QMGR

Same sysplex
Same QSG
Queue Sharing Groups (QSGs)

DB2 Data Sharing Group

WebSphere MQ Queue Sharing Group

Mover
QMGR
Private Queues
Private Objects

Mover
QMGR
Private Queues
Private Objects

Mover
QMGR
Private Queues
Private Objects

Shared Queues

Data for msg > 63KB

Shared Objects
CF Structures for shared-queues

- Structures for QSG 1
  - Administration structure
    - Information for unit-of-work recovery and so on
  - Application structures
    - Queue
    - Queue
    - Queue
- Structures for QSG 2
  - Administration structure
    - Information for unit-of-work recovery and so on
  - Application structures
    - Queue
    - Queue
    - Queue
Creating CF structures and shared queues

- Define a structure to z/OS (not to WebSphere MQ) by updating the CFRM policy (see System Setup Guide):
  - Structure is known to WebSphere MQ by its 12-character *str-name*.
  - Structure is known to z/OS by the 16-character name formed by:
    - \( qsg-name \| \| str-name \) (Application structures)
    - \( qsg-name \| \| CSQ_ADMIN \) (Administration structure)

- Define a shared queue using the DEFINE QLOCAL command on any queue manager in the QSG:
  - DEFINE QLOCAL(*queue-name*) QSGDISP(SHARED) CFSTRUCT(*str-name*)

- z/OS creates the structure when required (first use).
- WebSphere MQ creates the queue when required (first use).
Large messages with DB2
Large Shared Queue Messages (using DB2)

- Message 10K
- Message 100K (ptr)
- Message 100M (ptr)
- Message 1K

Shared Queue

DB2 Table CSQ.ADMIN_B_MESSAGES

- Message Data 100M
- Message Data 100K
...
SMDS
Large Shared Queue Messages (using SMDS)

1. APP MQPUT
2. QM1
3. QM2
4. QM2 SMDS

QM1 Shared Queue Message 100K (ptr)

APP MQGET

Shared Queue

Message 100K (ptr)
SMDS Performance Improvement

- Tests show comparable CPU savings making SMDS a more usable feature for managing your CF storage
- SMDS per CF structure provides better scaling than DB2 BLOB storage
Selecting which messages to offload

• Messages too large for CF entry (> 63K bytes) are always offloaded.

• Other messages may be selectively offloaded using offload rules.
  • Each structure has three offload rules, specified on the CFSTRUCT definition.
  • Each rule specifies message size in Kbytes and structure usage threshold, using two parameters:
    • OFFLDnSZ(size) and OFFLDnTH(percentage), where n = 1, 2, 3.
  • Data for new messages exceeding the specified size is offloaded (as for a large message) when structure usage exceeds the specified threshold.
  • Default rules are provided which should be useful in most cases.
  • Rules can be set to dummy values if not required.

• Without offloading data, it is possible to store 1.25M messages of 63KB on a 100GB structure

• When offloading all messages, possible to store approx 140M messages on the same structure, irrespective of message size
Typical use of offload rules

- The three offload rules have no fixed order but are typically intended to be used as follows:
  - Rule 1 is used to save space for fairly large messages by offloading them, with little performance impact, even when plenty of space left.
    - SMDS defaults: OFFLD1SZ(32K), OFFLD1TH(70)
  - Rule 2 is used as an intermediate step between rules 1 and 3, to start saving more space as the structure usage increases, in exchange for a minor performance impact.
    - SMDS defaults: OFFLD2SZ(4K), OFFLD2TH(80)
  - Rule 3 is used to maximize the remaining space when the structure is nearly full, by offloading everything possible.
    - SMDS defaults: OFFLD3SZ(0K), OFFLD3TH(90)
Storage benefits of offloading

- ~ 35000 msgs in CF
- ~ 5000 msgs in CF
- ~ 140000 offloaded msgs
- ~ 140000 offloaded msgs

70% 80% 90% 100%

> 32KB > 4KB > 0KB

1GB structure using 20KB messages

~ 35000 msgs in CF

~ 50000 offloaded msgs

~ 140000 offloaded msgs

~ 320000 msgs using offloading vs ~ 50000 without offloading
Creating a shared message data set

- SMDS is defined as a VSAM linear data set using IDCAMS DEFINE CLUSTER.
  - Requires **LINEAR** option.
  - Control interval size must be 4096, which is the default for linear.
  - Requires **SHAREOPTIONS(2 3)**, allowing one queue manager to write and other queue managers to read at the same time.
  - If maximum size may need to exceed 4GB, requires SMS data class which has VSAM extended addressability attribute.
  - If automatic expansion is to be supported, requires an appropriate secondary space allocation (although a default of 20% will be used if an expansion attempt fails because of no secondary allocation).
- Can optionally be pre-formatted, for example using CSQJUFMT.
  - Otherwise formatted automatically when first opened.
Creating a shared message data set

• The **DSGROUP** parameter on the **CFSTRUCT** definition specifies the group of data sets associated with the application structure.
  • It is specified as a generic data set name with a single asterisk as the point where the owning queue manager name is to be inserted.
  • It is required when the option **OFFLOAD(SMDS)** is specified.

• **CSQ4SMDS** in **SCSQPROC** provides JCL to define and format a single dataset

```jcl
DEFINE CLUSTER
  (NAME(++HLQ++.++QMGR++.++CFSTRUCT++.SMDS)  -
  MEGABYTES(++PRI++ ++SEC++) -
  LINEAR -
  DATACLAS(EXTENDED) -
  SHAREOPTIONS(2 3) )-
DATA
  (NAME(++HLQ++.++QMGR++.++CFSTRUCT++.SMDS.DATA) )
```
Access to shared message data sets

- Shared message data sets must be on shared direct access storage accessible to all queue managers within the QSG.

- Normal running:
  - Queue manager opens own data set read/write.
    - Requires **UPDATE** access to own data set.
  - Queue manager opens other data sets read-only.
    - Requires **READ** access to all other data sets.

- Media recovery processing:
  - Queue manager performing recovery opens own data set and all other data sets for read/write access.
    - Requires **UPDATE** access to all data sets.
Shared message data set capacity considerations

• Each shared message data set only contains data for large messages written via its owning queue manager.

• Message size calculation:
  • Each stored message includes standard headers (usually 352 bytes).
  • Each message is stored as one or more message blocks.
  • Each message block is stored in a range of consecutive 4K pages on the data set, with a very small header (32 bytes).
  • Approximate data set space required per large message, in bytes, is given by size of message plus header rounded up to next 4K.

• Multiply by maximum anticipated backlog of messages written via that queue manager (plus some safety margin) to estimate size needed for data set.
SMDS capacity considerations – expansion

• Data set can be automatically expanded when necessary.
  • Normally set by DSEXPAND(YES|NO) option on CFSTRUCT, which specifies default option for data set group.
  • Can also be overridden for individual data sets using DSEXPAND option on ALTER SMDS.

• Expansion attempt is automatically triggered when 90% full.
  • If no secondary allocation was specified, VSAM error message will appear, but queue manager will retry using a default secondary allocation of 20% of the existing size.
  • If expansion fails (not enough space available), queue manager sets DSEXPAND(NO) to prevent further attempts. Operator can use ALTER SMDS to set DSEXPAND(YES) again after problem is fixed.
  • If maximum extents are reached, data set cannot be expanded any further. (It could however be marked unavailable then copied to a larger data set which is then renamed back to the original name).
Structures – Persistence and Recovery
Failure and persistence

Queue manager failure

- Queue manager
- Private queues
  - Nonpersistent messages on private queues OK (kept)
- Shared queues
- Messages on shared queues OK (kept)
- Nonpersistent messages on shared queues lost (deleted)

Coupling facility failure

- Queue manager
- Private queues
  - Persistent messages on shared queues restored from log
- Shared queues
- Messages on shared queues OK (kept)
- Nonpersistent messages on shared queues lost (deleted)
Admin Structure Recovery

- Prior to V7.0.1 each queue manager would rebuild own admin structure entries
  - Particularly an issue in a DR situation.
    - Need to start all queue managers to rebuild admin structure
    - Once recovered, application structures could be recovered
- At V7.0.1 active queue managers notice if other queue managers don’t have entries, and initiate rebuild on their behalf
A failure of the Coupling Facility is most likely going to be presented to connectors as a Loss of Connectivity. Prior to V7.1, if a queue manager receives a loss of connectivity, it will terminate. In the case of a Coupling Facility failure, this would mean a QSG wide outage (unless protected by CF Duplexing).
With V7.1 the queue managers will not terminate. They will automatically attempt to re-establish access to the structures affected.

In the case of a total loss of connectivity the queue managers can automatically recover (RECOVER CFSTRUCT) the structures that were on the failed CF into an alternative CF (if available)
Total loss of connectivity - CFCONLOS(TOLERATE)

• Administration structure
  • The queue manager will not terminate and try and reconnect and rebuild its admin structure data
  • If the structure remains unavailable, some shared queue operations will be unavailable
  • Failure to connect to the admin structure during start up is not tolerated

• Application structures
  • Connection loss is partial if at least one system in the QSG still has connectivity to the CF the structure is allocated in
  • If total loss of connectivity, the structure is rebuild on an alternative CF if available
    • The structure is likely to be in a failed state and requires recovery
In the case of a partial loss of connectivity, a System Managed Rebuild will be automatically initiated by the QMGRs to rebuild the structures into a more available CF. This will mean that both persistent and non-persistent messages will be retained.
CF Loss of Connectivity Tolerance

- QMGR CFCONLOS(TERMINATE | TOLERATE)
  - Specifies whether loss of connectivity to the admin structure should be tolerated
  - Default is TERMINATE
  - Can only be altered to TOLERATE when all QSG members are at 7.1

- CFSTRUCT CFCONLOS(TERMINATE | TOLERATE | ASQMGR)
  - Specifies whether loss of connectivity to application structures should be tolerated
  - Only available at CFLEVEL(5)
  - Default is ASQMGR for new CFLEVEL(5) structures, and TERMINATE for structures altered to CFLEVEL(5)

- CFSTRUCT RECAUTO(YES | NO)
  - Specifies whether application structures should be automatically recovered
  - Only available at CFLEVEL(5)
  - Default is YES for new CFLEVEL(5) structure, and NO for structures altered to CFLEVEL(5)
CFRM Policy Considerations

• CFSTRUCT(TEST1)  
  STRUCTURE NAME(SQ27TEST1)  
  CFLEVEL(5)  
  SIZE(50000)  
  CFCONLOS(TOLERATE)  
  INITSIZE(20000)  
  RECAUTO(YES)  
  DUPLEX(ALLOWED)  
  OFFLOAD(SMDS)  
  ALLOWAUTOALT(YES)  
  PREFLIST(P5CF01,P5CF02)

• If using CFCONLOS(TOLERATE) also need to consider multiple CFs in PREFLIST
• ALLOWAUTOALT(YES) enables CF to adjust entry/element ratio, and also automatically resize structure up to SIZE value (can also adjust down to MINSIZE!!)
• MQ structures can be duplexed… this will make most types of failures transparent to MQ
Client Channels
Client Channels

- Regular client channels are stateless, so don’t use synchronization queues
  - Only benefit of using a shared channel is the shared status
  - Can cause performance issues if using shared channel
    - Needs to update DB2 status for each connect/disconnect
- Can configure a generic port to point at INDISP(QMGR) listener on each queue manager
  - Can still benefit from failover and balancing of client connections without using a shared channel, and can still use QSG name on the MQCONN
- Will not work for Extended Transactional Client (including WAS 2-Phase Commit over client conn) until at V7.0.1
2-Phase Commit Client Connections

• When setting up the connection, specify the QSG name rather than QMGR name
  • In MQConnectionFactory if using JMS under WAS, you must ensure that you are only using shared resources
  • This causes a UR with GROUP disposition to be created, rather than QMGR
  • A GROUP UR can be inquired and resolved via any member of the QSG
    • If there is a failure, the transaction manager will reconnect to the QSG and request a list of in-doubt transactions. GROUP URs will be reported back no matter what QMGR they were started on
GROUPUR – The Problem (Pre V7.0.1)

QM1

Client App connects via generic port and starts UOW

Generic Port

If TM reconnects to QM2 it only be told what is in-doubt on QM2, meaning that it will throw away any information about in-doubts on QM1

QM2

If there is a failure, TM will reconnect via generic port to inquire what transactions need resolving

TM

Client APP
Client App connects via generic port and starts UOW

xa_open string needs to be specified with the QSG name rather than QMGR name, this means a GROUPUR is created

If TM reconnects to QM2, QM2 will inquire all the indoubts that have a GROUP disposition, whatever QMGR that were running on.

If there is a failure, TM will reconnect via generic port to inquire what transactions need resolving
More Information

- WebSphere MQ for z/OS Concepts and Planning Guide
- SupportPacs MP16, MP1E, MP1F, MQ1G
  - www.ibm.com/software/integration/support/supportpacs/perfreppacs.html
- RedPaper 3636 – WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment
Any questions?
Session 17822: 
MQ Parallel Sysplex Exploitation, 
Getting the Best Availability from MQ on z/OS by using Shared Queues

Paul Kettley

PLM for Messaging on z

paulk@uk.ibm.com