Introduction to shared queues

Matt Leming
lemingma@uk.ibm.com

Agenda

- What are shared queues?
- SMDS
- CF Flash
- Structures – persistence and recovery
- Clients and GROUPUR

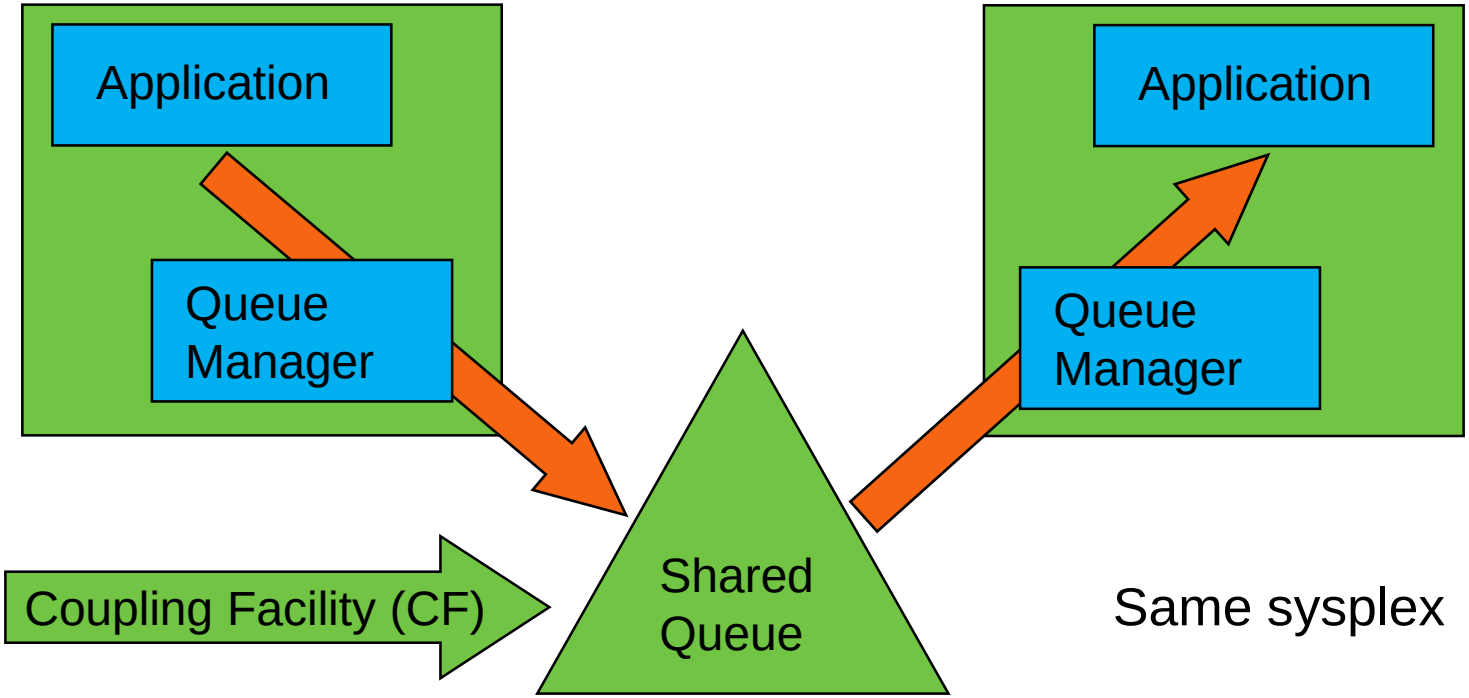
What are shared queues?

Shared queues - overview

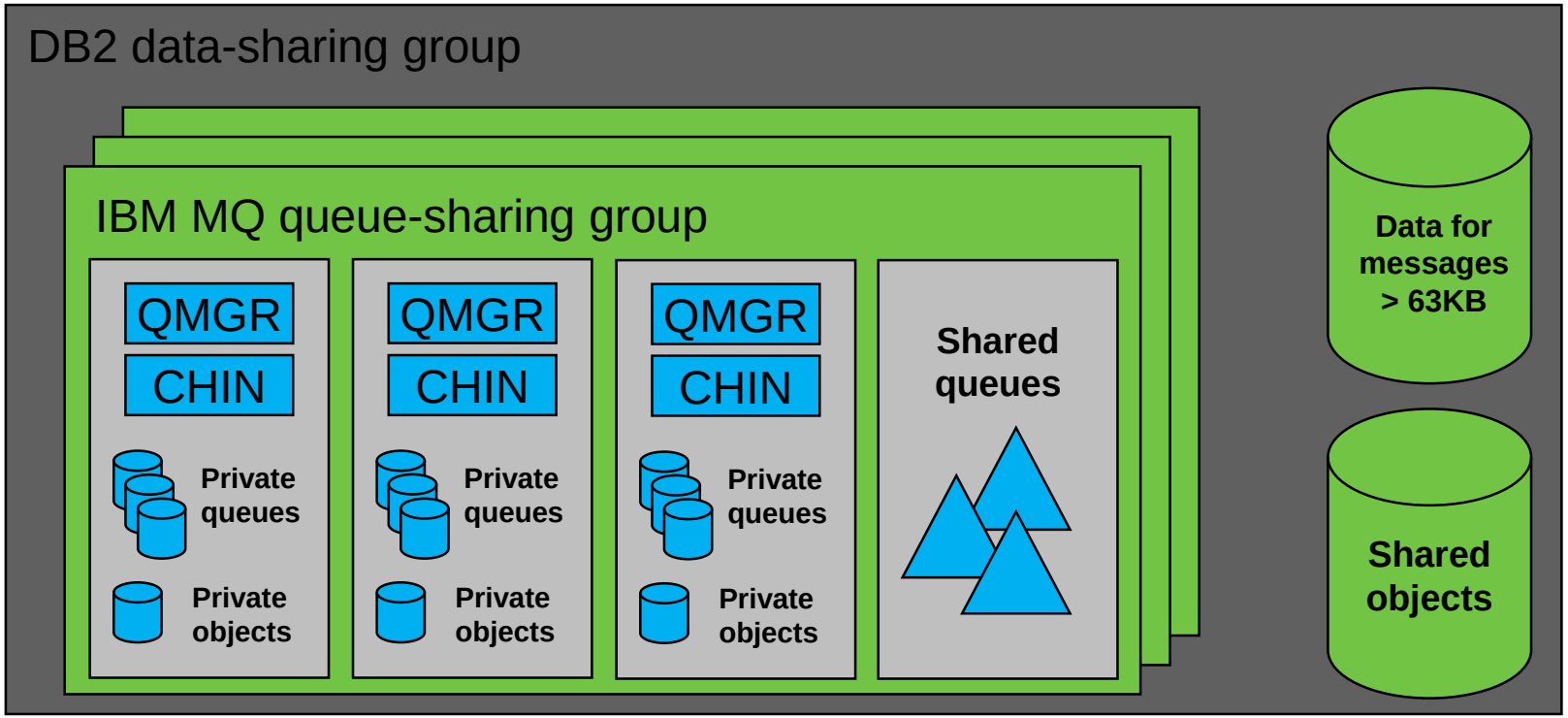
- **Function**
  - Multiple queue managers can access the same shared queue objects
  - Multiple queue managers can access the same shared queue messages

- **Benefits**
  - Availability for new messages
  - Availability for old messages
  - Pull workload balancing
  - Scalable capacity
  - Low cost messaging within a sysplex

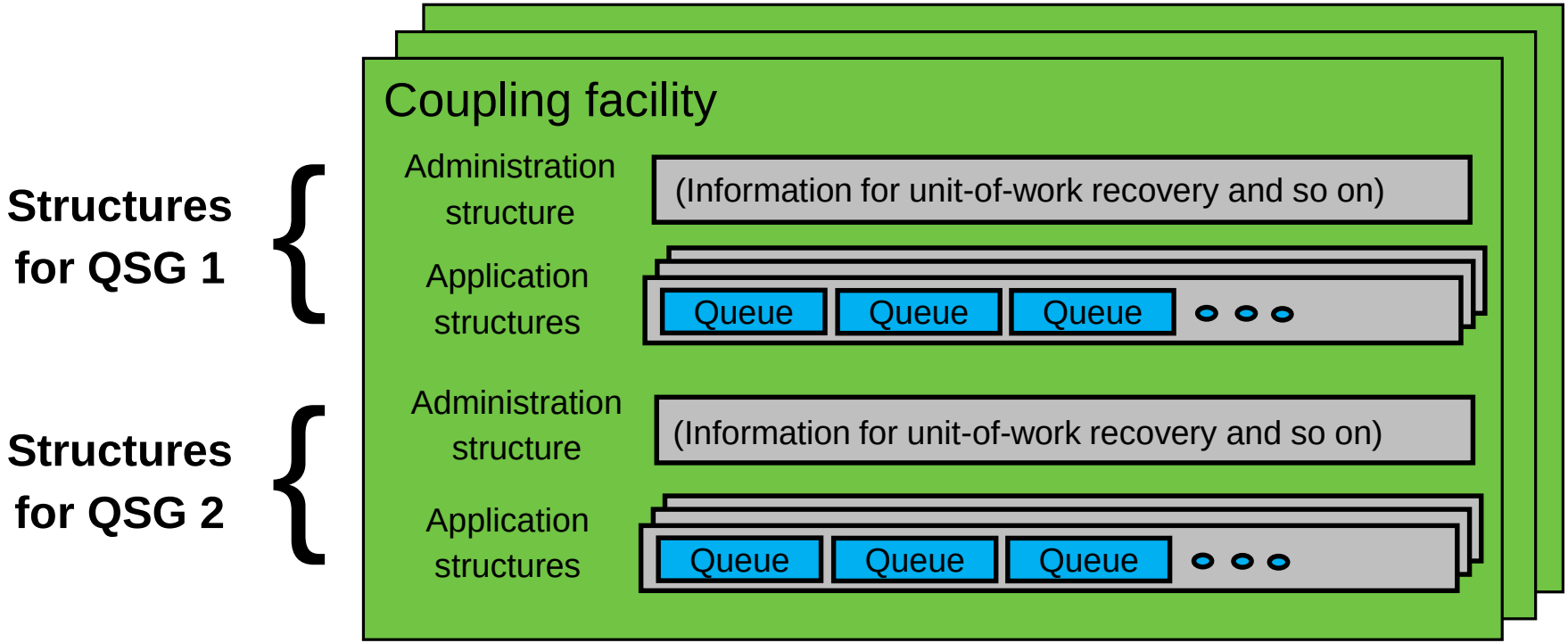
Shared queues



Queue-sharing groups (QSGs)



CF structures for shared queues

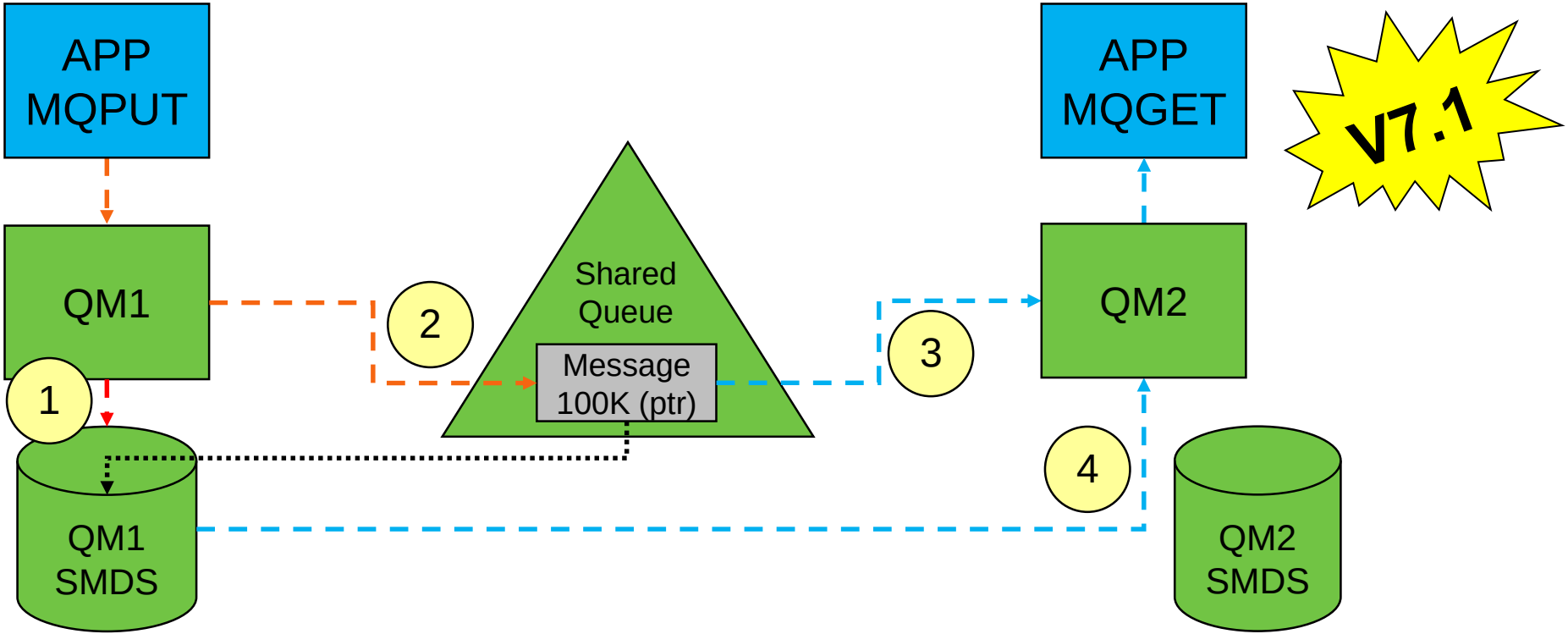


Creating CF structures and shared queues

- **Define a structure to z/OS (not to IBM MQ) by updating the CFRM policy (see the system setup guide):**
  - Structure is known to IBM MQ by its 12-character struct-name
  - Structure is known to z/OS by the 16-character name formed by:
    - `qsg-name || struct-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(struct-name)`
- **z/OS creates the structure when required (first use)**
- **IBM MQ creates the queue when required (first use)**

SMDS

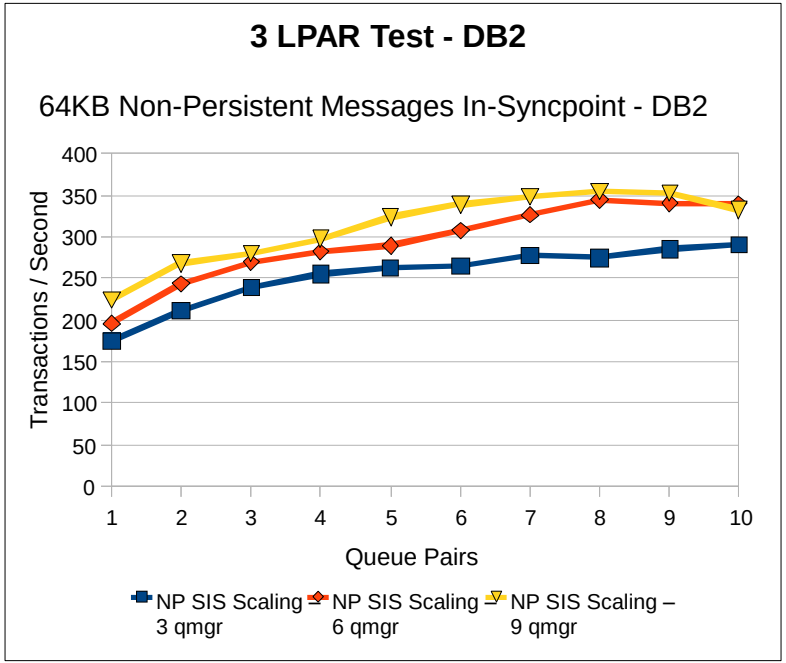
Large shared queue messages (using SMDS)

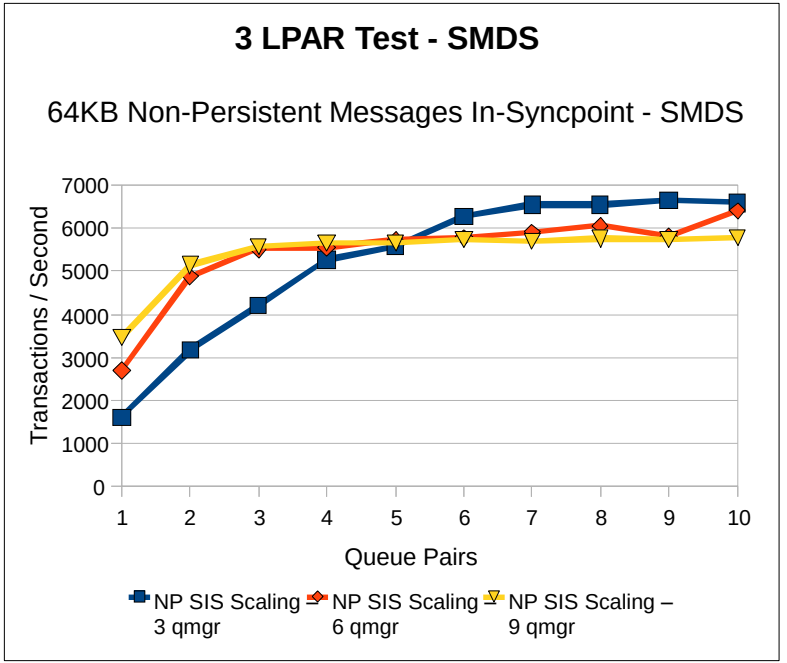


SMDS performance improvement

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





SIS: Server In Synchpoint

Selecting which messages to offload

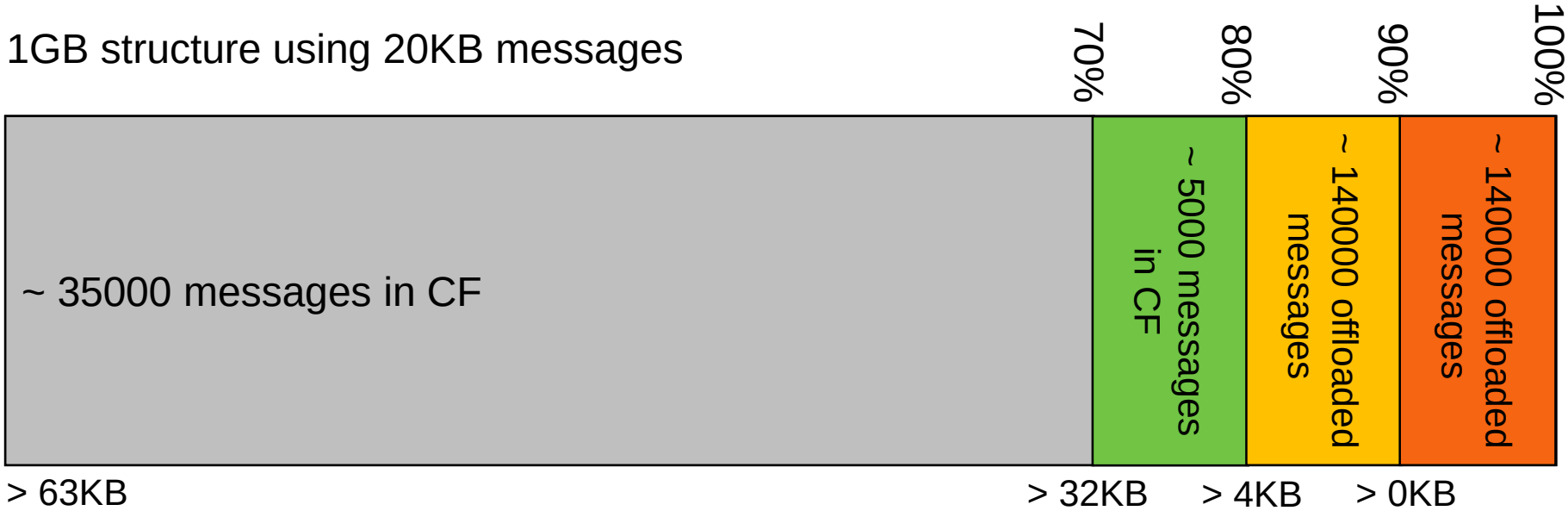
- **Messages too large for CF entry (> 63K bytes) are always offloaded**

- **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 approximately 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



~ 320000 messages using offloading vs ~ 50000 without offloading

Creating a shared message data set

- **SMDS is defined as a VSAM linear data set using `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 cont.

- 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 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

```
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**

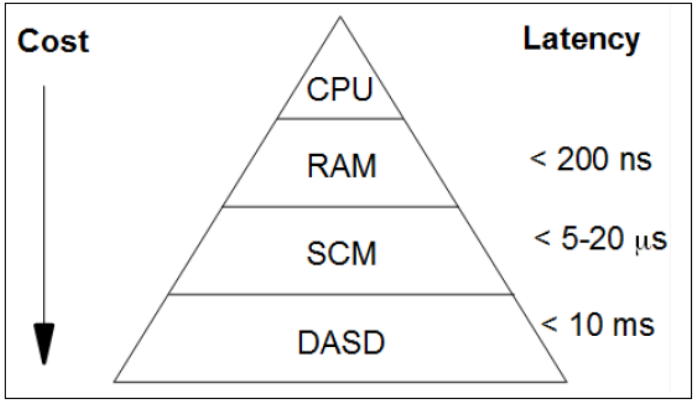
- **Normal running:**
  - Queue manager opens own data set read/write
    - Requires UPDATE access to own data set
  - Queue manager opens other data sets read-only
    - Requires READ access to all other data sets

- **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

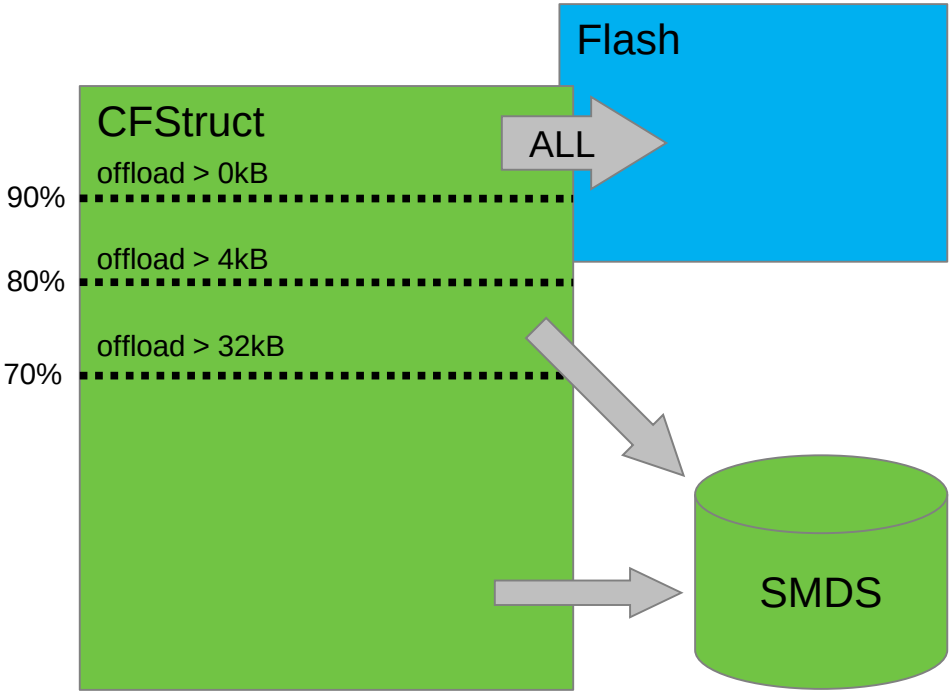
CF Flash

What is storage class memory?

- **Enterprise grade SSD, introduced originally in zEC12 hardware in Flash Express cards**
- **Faster than disk, cheaper than real storage**
- **Particularly useful when a large amount of data needs to be written fast**
- **Suitable for:**
  - Improving paging performance
  - Improving dump capture times
  - Pageable 1MB memory objects
  - CF structures (CF Flash), currently only for MQ shared queues



CF Flash: planned emergency storage

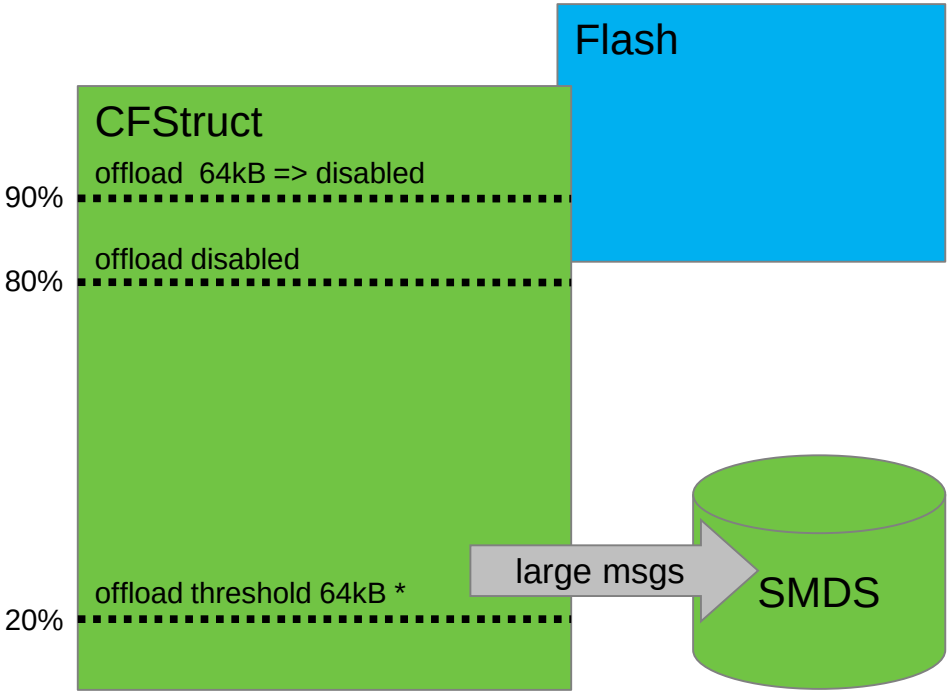


CFSTRUCT OFFLOAD rules cause progressively smaller messages to be written to SMDS as the structure starts to fill

Once 90% threshold is reached the queue manager stores the minimum amount of data per message to squeeze as many message references as possible into the remaining storage

CF Flash algorithm also starts moving the middle of the queue out to flash storage, keeping the faster 'real' storage for messages most likely to be got next

CF Flash: maximum speed



We want to keep high performance messages in the CF for most rapid access

CFSTRUCT OFFLOAD are configured with special value '64k' to turn them off. * You might choose to use one rule to alter the 'large data' threshold down from 63KB

Once 90% threshold is reached, the CF Flash algorithm starts moving the middle of the queue out to flash storage, keeping the faster 'real' storage for messages most likely to be gotten next

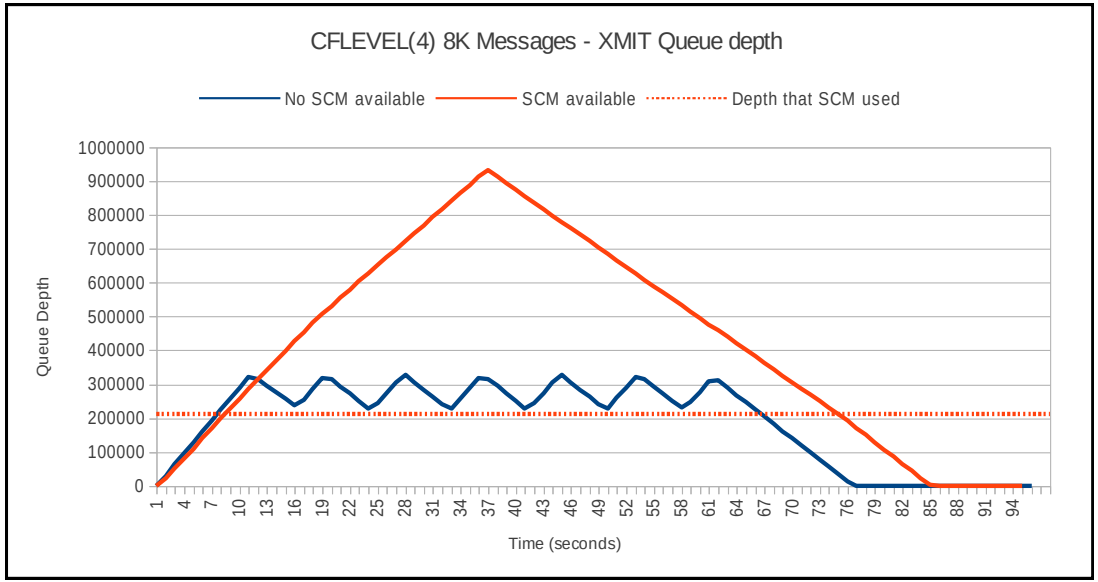
As messages are got and deleted, the CF flash algorithm attempts to pre-stage the next messages from flash into the CFSTRUCT so they are rapidly available for MQGET

In this scenario the flash storage acts like an extension to 'real' CFSTRUCT storage. However it will be consumed more rapidly since all small message data is stored in it

CF Flash: storage (using 4GB structure)

Scenario	Msg Size	Total Msgs	# in 'real'	SMDS space	# in 200 GB flash	Augmented (limit 30GB)
No SMDS No Flash	1kB	3M	3M			
	4kB	900,000	900,000			
	16kB	250,000	250,000			
SMDS No Flash	1kB	3.2M	3.2M	800MB		
	4kB	1.8M	1.8M	5GB		
	16kB	1.3M	1.3M	20GB		
“Emergency” Scenario	1kB	190M	2M	270GB	190M	30GB
	4kB	190M	600,000	850GB	190M	30GB
	16kB	190M	150,000	3TB	190M	30GB
“Speed” Scenario	1kB	150M	2M		150M	26GB
	4kB	48M	600,000		48M	8GB
	16kB	12M	150,000		12M	2GB

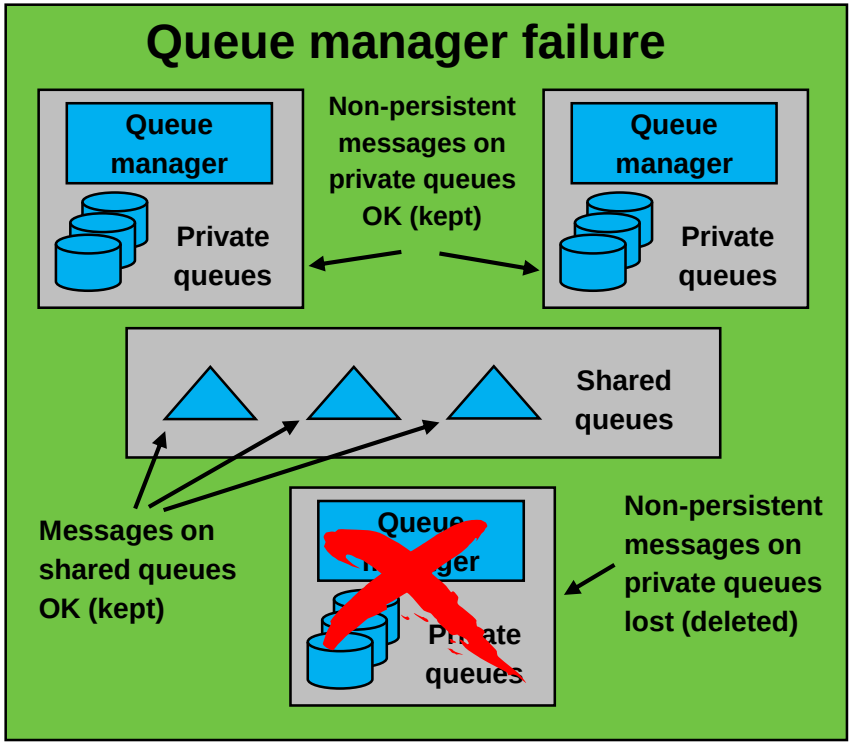
CFLEVEL(4) using 8KB messages

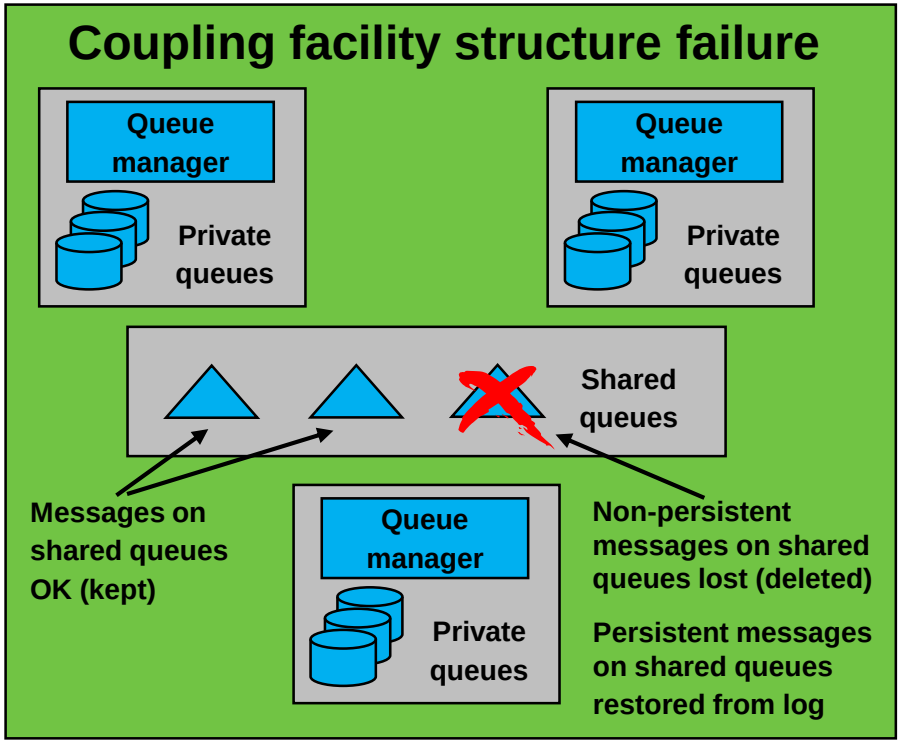


- Saw-tooth effect occurs when capture task goes into retry mode due to “storage medium full” reason code
- Even with these 5 second pauses, the non-SCM capable workload completes in 90% of the time of the SCM capable workload
- Cost of workload in MVS differs by less than 2%
- Get rate once the capture task has completed:
  - No SCM: 21100 messages/sec
    ~ 164MB/sec
  - SCM: 19000 messages / second
    ~ 148MB/sec

Structures – persistence and recovery

Failure and persistence

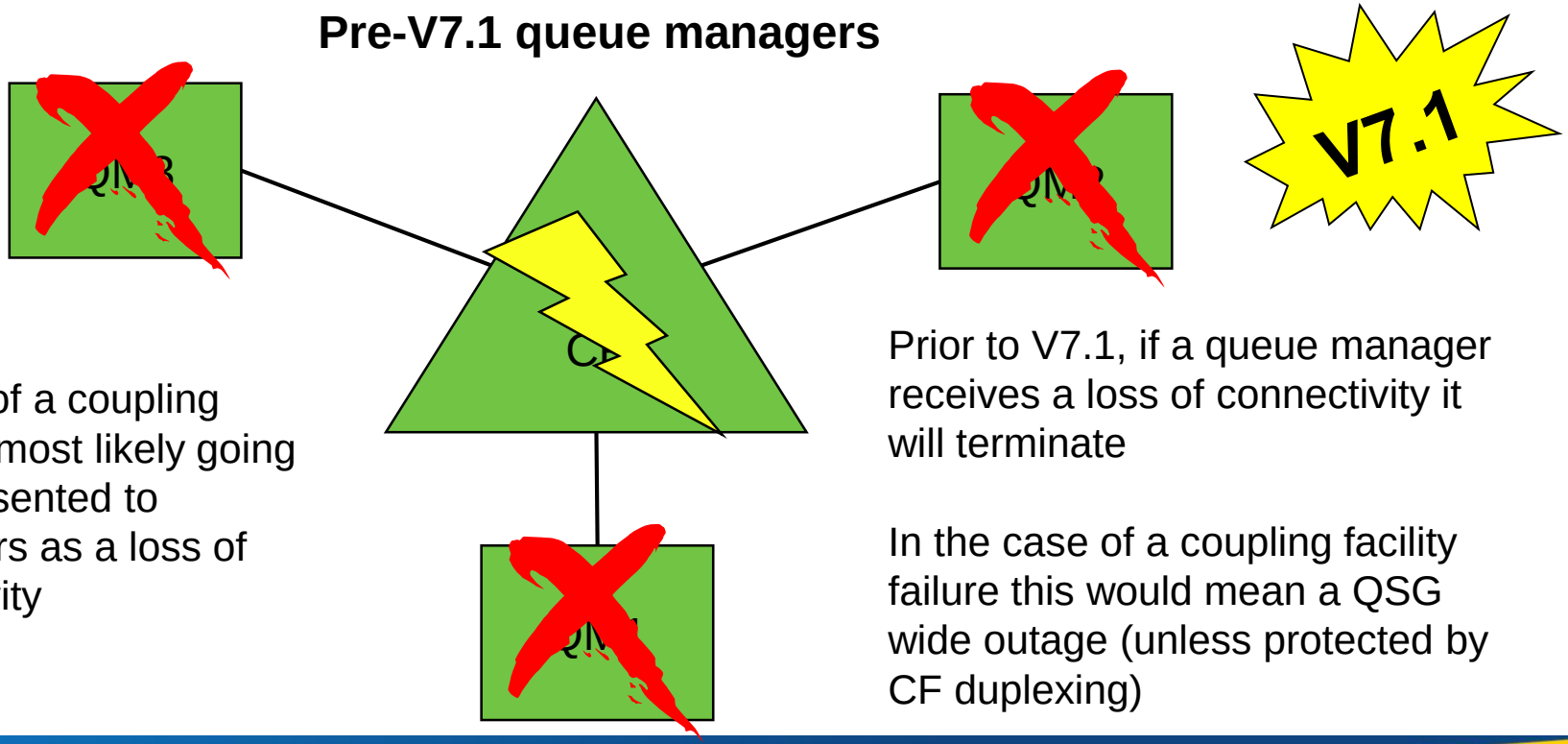




Admin structure recovery

- **Prior to V7.0.1 each queue manager would rebuild its 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
- **From V7.0.1 active queue managers notice if other queue managers don't have entries, and initiate the rebuild on their behalf**

CF loss of connectivity tolerance

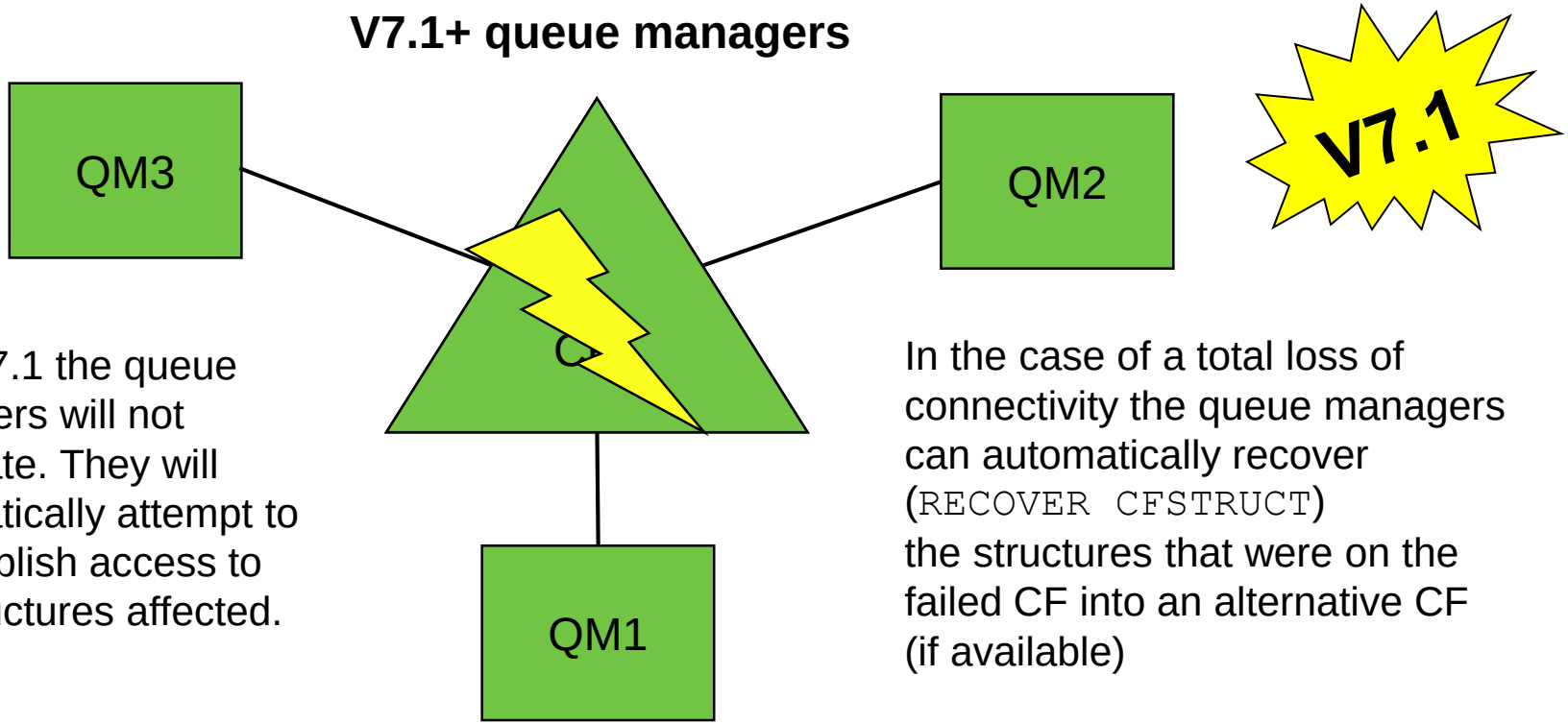


A failure of a coupling facility is most likely going to be presented to connectors as a loss of connectivity

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)

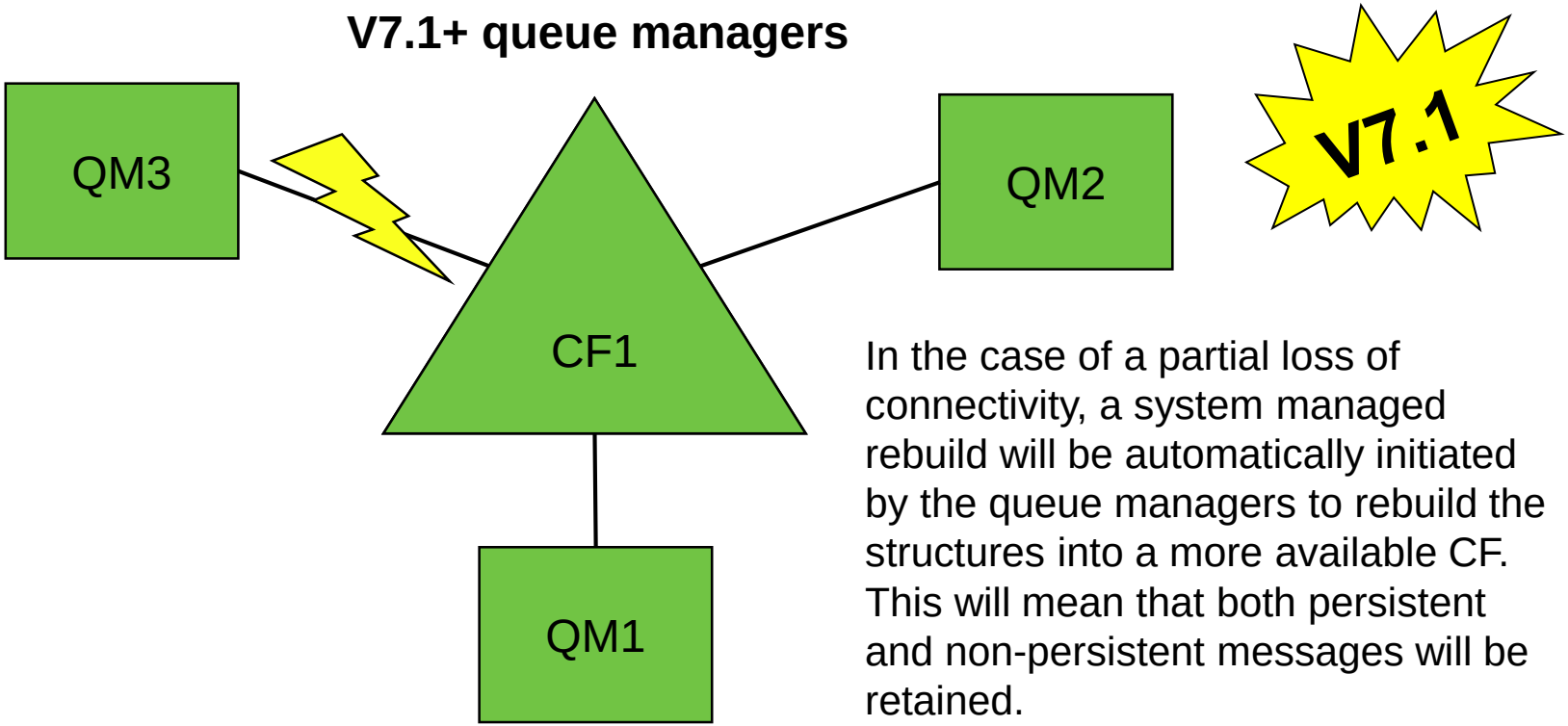
CF loss of connectivity tolerance (total)



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)

CF loss of connectivity tolerance (partial)



In the case of a partial loss of connectivity, a system managed rebuild will be automatically initiated by the queue managers 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 (or higher)

- **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

MQ Definition

```
CFSTRUCT(TEST1)
CFLEVEL(5)
CFCONLOS(TOLERATE)
RECAUTO(YES)
OFFLOAD(SMDS)
```

CFRM Definition

```
STRUCTURE NAME(SQ27TEST1)
SIZE(50000)
INITSIZE(20000)
MINSIZE(15000)
DUPLEX(DISABLED)
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`)
- IBM MQ structures can be duplexed - this makes most types of failure transparent to the queue manager

Clients and GROUPUR

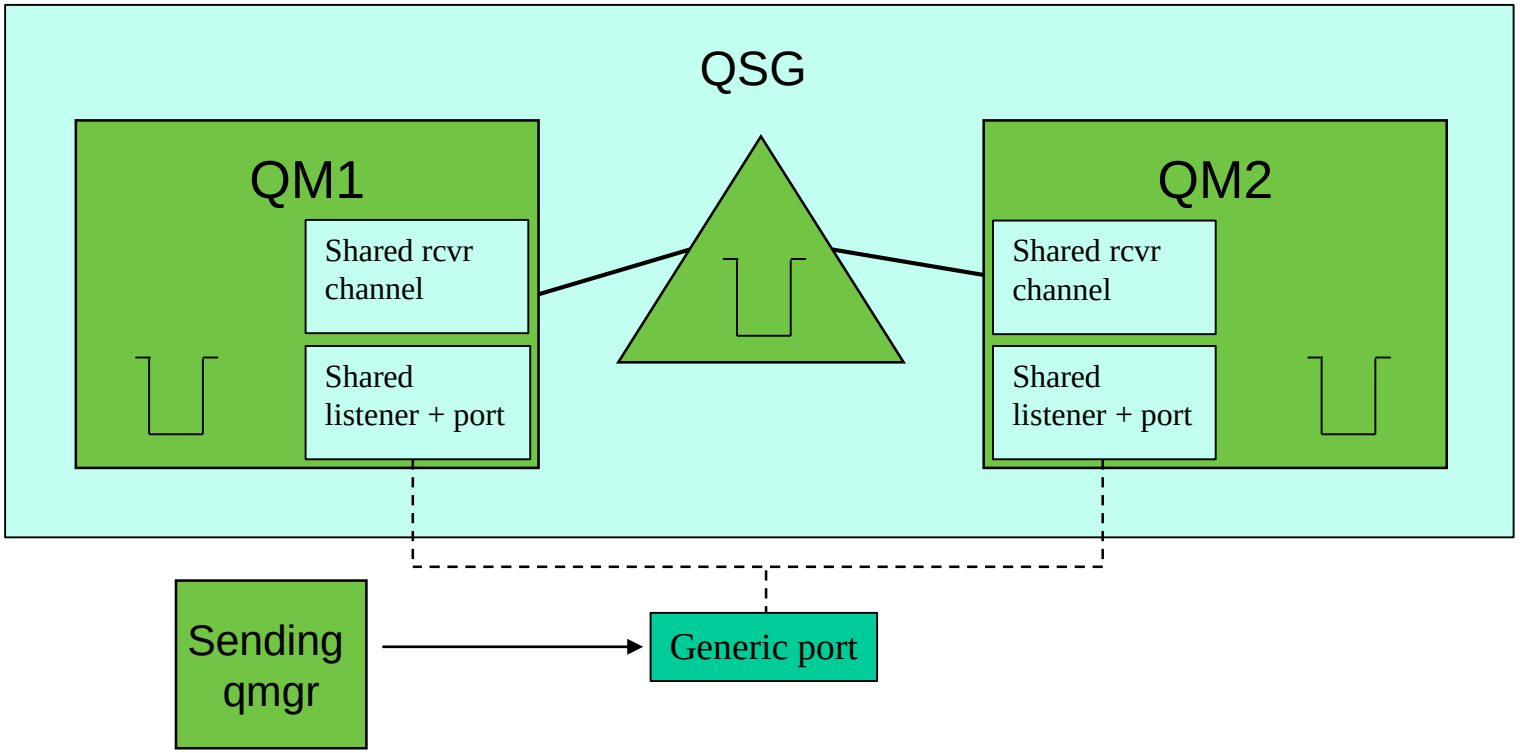
Shared channels

- **Channels can be shared in a QSG**

- **Shared sender channels**
  - **Shared transmission queue**
  - Shared synch queue
  - Some channel state held in DB2
  - Balanced around available queue managers in the QSG on channel start
  - Can restart, and recover, on any queue manager in the QSG

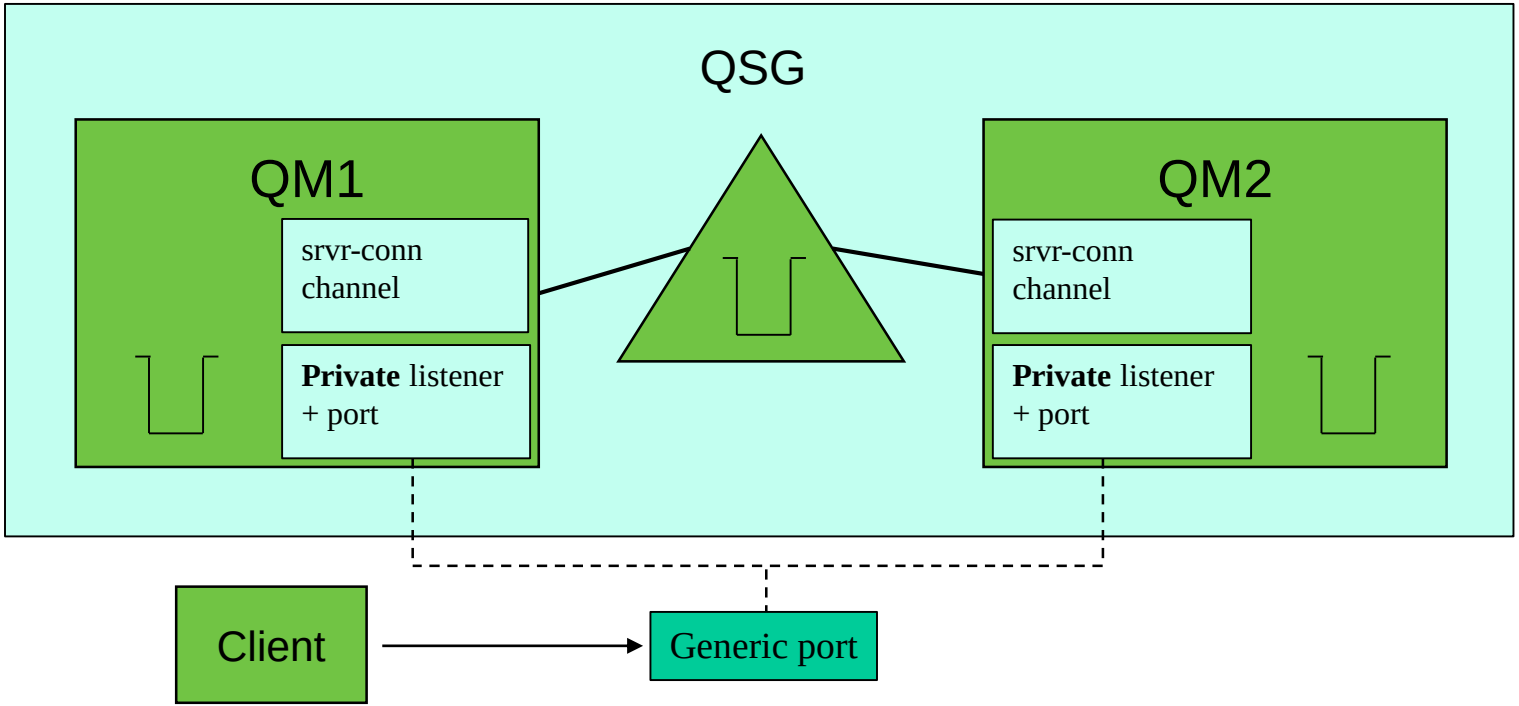
- **Receiver channels**
  - **Shared listener (and associated port)**
  - Generic port (WLM using sysplex distributor) routes to shared listener
  - Shared synch queue
  - Some channel state held in DB2
  - Can restart, and recover, on any queue manager in the QSG

Shared receiver channels



Server-conn channels, and a problem…

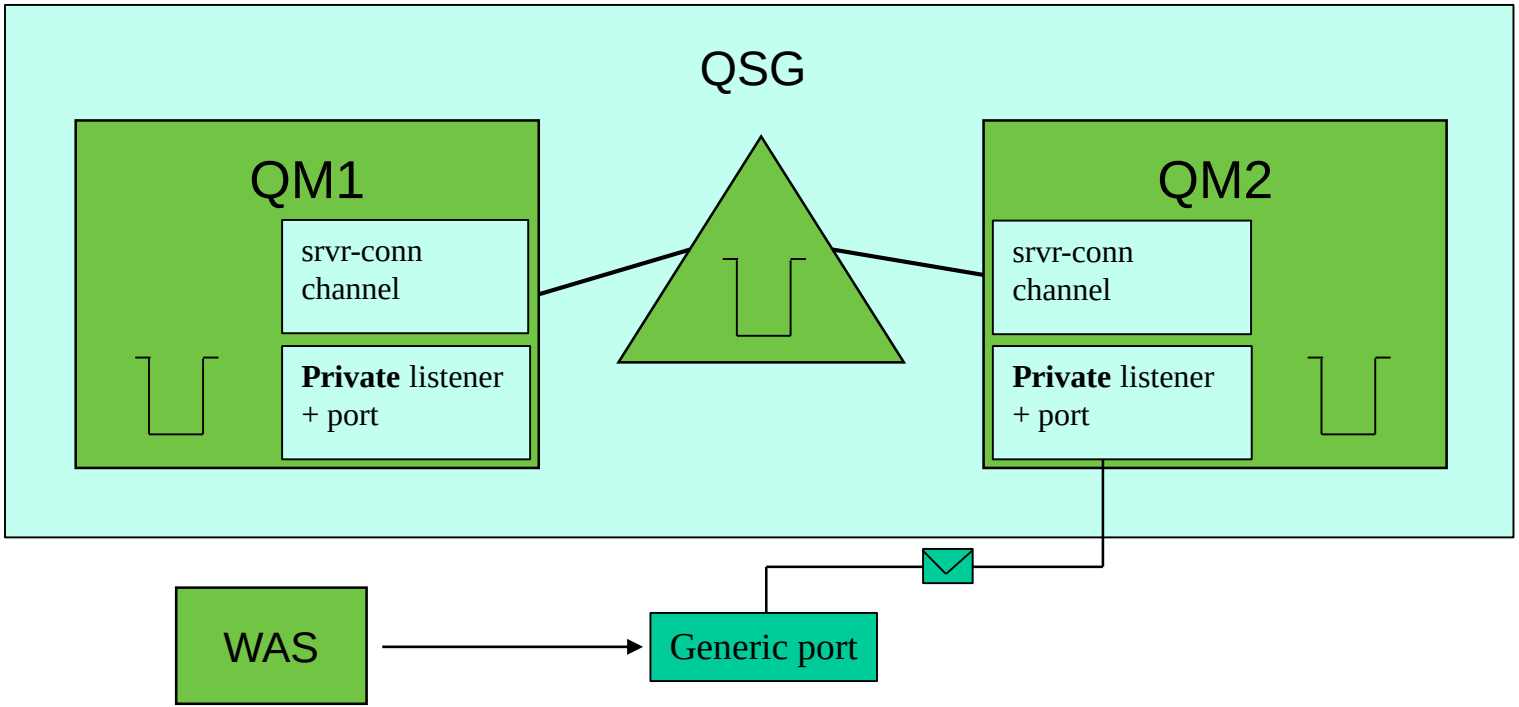
- What about server-conn channels then?



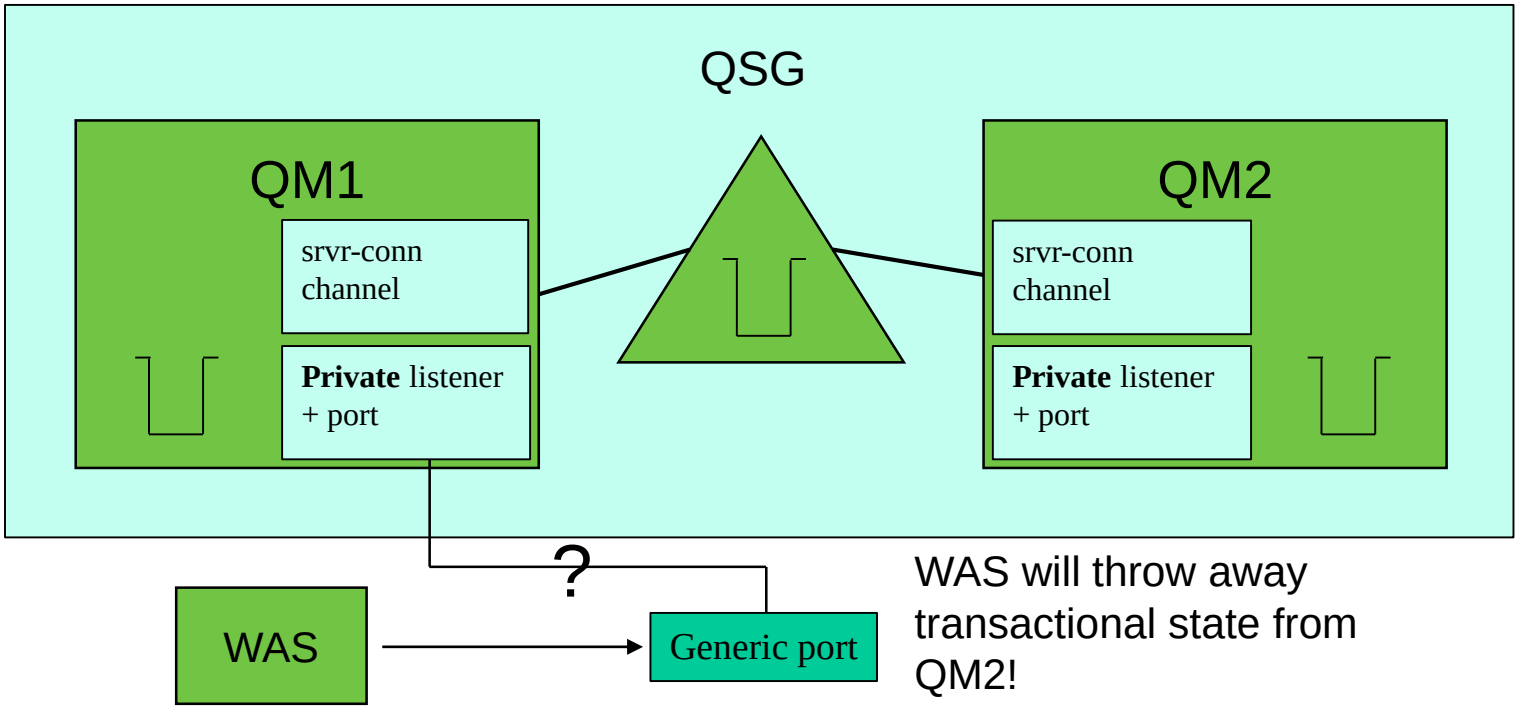
Server-conn channels, and a problem…

- **Now on to the problem…**
- **Clients can connect into any queue manager in QSG**
  - Use QSG name instead of channel name
- **Fine if transactions are coordinated by queue manager**
- **Problems if transactions are coordinated by third party (WAS)**
  - Transaction state is held in the queue manager
  - If transaction recovery is necessary (e.g. connection fails) WAS could get WLM'd to a different queue manager without the necessary state
  - Result: unatomic transaction because of the way XA works
- **IMPORTANT: This is only a problem with clients connecting in to server-conn channels. Clients using bindings based connections are fine**

Messaging



Recovery



GROUPUR: the solution!

- **MQ 7.0.1 added GROUPUR**
  - Configured on queue manager: **GROUPUR(ENABLED)**
  - Other configuration required (specifically named queue, opmode, etc)
- **Connect into QSG using QSG name**
- **Transactions associated with QSG instead of qmgr**
- **WAS can connect to any qmgr in QSG and recover, and resolve, transaction state**
- **Works for both shared, and private queues**
  - Shared queue recovery performed straight away
  - Private queue recovery deferred until owning queue manager starts up

Summary

- What are shared queues?
- SMDS
- CF Flash
- Structures – persistence and recovery
- Clients and GROUPUR

Would you like to take part in IBM MQ Design Research?

- **The IBM MQ team is currently conducting some long term research with our MQ customer base.**
- **With this survey we would like to understand:**
  - Who is interreacting with MQ and what are their responsibilities?
  - Which customers are interested in moving IBM MQ into the cloud?
  - Which customers would like to take part in future research?
- **We estimate the survey should take 4 minutes to complete.**
- **Please note: This survey is for distributed users only.**
- **If you're interested, go to ibm.biz/MQ-Customer-Survey**

Questions?