

Some MQ Practicalities

Batches, Syncpoints and Triggers

H.R. McCanless



(c) Copyright IBM Corporation 2002

Any references to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



Topics

- Some Definitions
- Batches
- Syncpoints
 - ▶ Relation to Transaction Services (tx_)
 - ▶ Using mqcmit()/mqback()
 - ▶ Performance considerations when using tx_ with MQ
- Triggers and Processes in TPF



Definitions

- Unit of Work
- Batch
- Message
- Synchronization
- Syncpoint
- Transaction Services
- Triggers
- Processes



Batch

- A group of messages on a message channel.
- A minimum unit of work is a batch.
- The maximum size of the batch is declared with the message channel definition:
 - ▶ ZMQSC DEF CHL-...BATCSZ-
- The end of the batch occurs:
 - ▶ When the Transmission Queue is exhausted or
 - ▶ When the maximum batch size is reached, whichever comes first.



Why Do I Care?

- In the sender, a batch is handled by ECBs managing the transmission queue.
 - ▶ One for the current batch being transmitted
 - ▶ One to pre-fetch the next batch
 - ▶ One to delete from the queue once acknowledged.
- In the receiver, a new batch's ECB is initiated via AOR.
- The receiver channel must harden messages before acknowledging a batch.



Why Do I Care? (cont.)

- The larger the batch:
 - ▶ The fewer the administrative flows
 - ▶ The longer the hardening time for a single batch
 - May be significant for batches >1500 .
- TPF is fast.
 - ▶ It is difficult to generate traffic one message at a time and have a batch of >1 .
 - ▶ This results in each message being committed individually.



Syncpoints

- A syncpoint causes the activity of a unit of work to be hardened to the queue.
- Syncpoints in MQ are functionally similar to Transaction Services (tx_) in TPF.
- By declaring its own commit scopes (syncpoints), a program will reduce overhead.



MQ Client/Server Syncpointing

- MQ may be a transaction manager or a resource manager.
 - ▶ For TPF resident applications accessing physically local queues, TPF's transaction services must be used.
- To provide remote client applications with primitive commit scope capability TPF now provides:
 - ▶ syncpoint options in MQGMO and MQPMO
 - ▶ mqcmit() and mqback() functions



MQ Client/Server Syncpointing (cont.)

- A TPF application may issue these functions if it is a client to a physically remote manager that supports the function.
- The TPF based server under its MQ manager supports the functions for applications on physically remote clients.



MQ Client/Server Syncpointing (cont.)

- The "commit scope" begins at the beginning of a transaction.
- Options for `mqget()`, `mqput()` and `mqput1()` control whether a particular message is part of the commit scope or not.
 - ▶ `MQGMO_SYNCPOINT`
 - ▶ `MQGMO_NO_SYNCPOINT`
 - ▶ `MQPMO_SYNCPOINT`
 - ▶ `MQPMO_NO_SYNCPOINT`



MQ Client/Server Syncpointing (cont.)

- If not part of the commit scope, the activity cannot be rolled back.
- When TPF is the server, the long lived ECB issues transaction services calls to declare the commit scope.



Syncpoints and tx_

- In TPF, the logging overhead caused by the many commit scopes can be reduced substantially through the use of transaction services.
 - ▶ Multiple mqput()s are executed within a root commit scope.
 - tx_begin()
 - tx_commit()
- Beware the giant commit scope.
 - ▶ tx_ affects other activities in addition to MQ.



Getting Messages to Processes

- Two categories of assigning a message to an application
 - ▶ `mqget()` with `wait`.
 - ▶ Triggers
 - First
 - Every



Wait

- The simplest method of assigning a message to a processing program is for the program to execute an `mqget()` with `wait`.
 - ▶ The `wait` option is determined by the settings of `Options (MQGMO_WAIT)` and `WaitInterval` in the `MQGMO` structure referred to by the `mqget()` option.
 - `WaitInterval` of all binary ones (-1F) (`MQWI_UNLIMITED`) indicates an indefinite wait time



Wait (cont.)

- The ECB is suspended until a message arrives.
- The cost is in the wait time of the ECB.
- Every waiting ECB is dispatched (via event processing) when a message arrives.
 - ▶ Those not receiving a message return to waiting.



Triggers

- Triggers are associated with queues.
- TPF uses its own trigger points.
 - ▶ Non-TPF MQ uses Trigger Monitor programs and Initiation queues to activate programming due to a specified event.
- The trigger points cause the creation of an ECB to execute a process defined by the user.
 - ▶ The default process is in the user exit program CUIR.



Trigger Types

- There are two types of trigger in TPF
 - ▶ TRIGTYPE-FIRST on qdef
 - MQGET to empty queue "sets" trigger
 - Msg arrives on an empty queue creates trigger ECB
 - Executes a defined Process or the CUIR User Exit
 - ▶ TRIGTYPE-EVERY on qdef (PUT15)
 - Each message arrival results in a new ECB
 - Control is passed to the program defined in the Process associated with the queue



Processes

- ZMQSC DEF PROCESS associates a process name with a program.
- ZMQSC DEF QL associates a process name with a queue
- ZMQSC DEF QL also defines whether a trigger is to operate FIRST or EVERY.



Processes (continued)

- A Process may be associated with more than one queue's Trigger
- MQ provides the originating queue name and other information as input to the Process.
 - ▶ User information may be passed with TRIGDATA defined in ZMQSC DEF QL.
- Even on TRIGGER-EVERY, a single ECB should try to drive the queue to exhaustion.
 - ▶ Other triggered ECBs could multiplex across multiple ISs and all would operate concurrently until the queue is empty.



The Point of All This...

- A single ECB processing a queue is slow, especially if each MQ function is outside of a commit scope.
- For performance (e.g. greater than 20-50 per second):
 - ▶ Multiple ECBs should process a single queue or
 - ▶ Multiple functions should be issued within a single commit scope.



Summary

- For heavy connections, use large batches.
- Use Transaction Services functions to coordinate queue updates
- Use Triggers to avoid long lived ECBs.

