Using WMQ in your CICS System

Lyn Elkins – elkinsc@us.ibm.com
IBM

Session 8288
Wednesday 02 March, 2011 1:30

Agenda

• Why are they used together?
• How are they being used together?
• Things to avoid
• Evolution not Revolution - WMQ and CICS
  • What’s new and improved?
  • What do I need to get there?
• Summary
Why are WMQ and CICS being used together?

- CICS processes an estimated 80% of ‘completed business functions’
- The cost of each transaction is often quite small
  - Rewriting custom applications is very costly
  - Reusing the existing applications is very cost effective
- WMQ provides:
  - Once and only once delivery of data
  - No need to write your own queueing mechanism
  - No sockets code
  - Quality of service can be as granular as at the message level
    - Message persistence
      - Messages can survive an outage or not
    - Messages can provide other QoS differentiation

Why are WMQ and CICS being used together? - Notes

- CICS has improved performance and lowered cost significantly over its lifespan.
- Often customized application are the core business processes – maintaining the databases of record, etc.
- Rewriting those processes have a relatively high failure rate and often cost much more to run ‘off the frame’ than originally estimated.
- We are seeing some customers port applications back to CICS
Why are WMQ and CICS being used together?

- Transactions
  - WMQ requests can fully participate in a CICS unit of work
    - Or not – if needed
- Asynchronous processing
  - Often used for batch modernization
  - Messages can build up to be processed during non-peak hours or as capacity is available
  - Trickle processing, in place of 'batch windows'
  - Persistent messages will be available for processing even after an outage
  - Pushing lower priority work out to other regions
- Pseudo-synchronous
  - Messages are async by nature
  - Often used in a ‘pseudo-synchronous’ fashion for request and reply scenarios

Why are WMQ and CICS being used together? - Notes

- Asynchronous processing
  - Advantage is that work becomes time independent
  - No requirement of requestors and processors to be available at the same time
  - Often used to push lower priority/less time critical work to other regions or environments
- Pseudo-synchronous
  - Advantage is that the same API is used for both sync and async
  - Messages and requests can expire, allowing the user app to send the request again.
  - Can be used for 'important' requests as well
    - if the request expires before a response is received, then the user can be informed that the work will be processed later
    - The request is not lost, even when the applications are not available
Why are WMQ and CICS being used together?

- **Availability**
  - Using a CICSplex and a QUEUEplex together can provide continuous availability
  - Using Messages on shared queues to initiate CICS transactions allows workload to be moved about, without disruption to the end user
  - An individual queue manager or CICS region is no longer a SPOF
- **Application and tools familiarity**
  - Standard debugging capability via CEDX
  - Common API
    - COBOL, PL/1, C, Assembler and Java
    - JMS is not supported under CICS
Messaging Patterns

Customer-facing transaction. Pseudo-synchronous does NOT mean slow!!

Could also be a Customer-facing transaction but no response is required.

Messaging Patterns - notes

- Certainly not the only styles, but these are very common
- Often there are multiple 'hops' to go thru a business process
- In this example
  - Retail sales:
    - Process A might be running on a retail point of sale machine
    - Process B is a CICS transaction to authorize a charge
  - Banking
    - Process A is an ATM
    - Process B is a CICS transaction to verify account balance before dispensing cash
How are WMQ and CICS being used together?

• Traditionally:
  • Bridging Techniques:
    • Initiating CICS transactions and programs with no changes
  • Using the MQ API in CICS programs
    • Allows for data greater than 32K to be passed into the programs
      • *No channels and containers required*
    • Provides simple ASCII-to-EBCDIC translation for string data
    • Simple and consistent API

• Newer:
  • SOAP over MQ
  • WebServices
  • New WMQ verbs – more on that later

How are WMQ and CICS being used together? Notes

• Traditionally:
  • Bridging Techniques:
    • MQ/CICS Bridge
      • *Works best for DPL enabled programs*
      • *Only using COMMAREA link (no channels and Containers)*
      • *New with WMQ V7 – MQCB function – we’ll talk about that later*
  
• Newer:
  • SOAP over MQ
  • WebServices
WMQ and CICS – A match made in Heaven

- Well Hursley at any rate
- As of CICS 3.2 – CICS development own the MQ interface code
  - It's threadsafe
  - Runs on L8 TCB (just like DB2)
- CICS 4.1 supports group attach
  - Like the attach to a DB2 Data Sharing group
  - CICS will attach to a queue manager in a QSG on the same LPAR
  - Selection is made randomly
- CICS PA can analyze some WMQ Performance information

WMQ and CICS – A match made in Heaven

- Some would say the match was made elsewhere
- Most customers are using the CICS delivered code
  - Performance improvements have been significant
  - No longer limited to 8 TCBs
WMQ and CICS – Things to Avoid

• Just because you can
  • Does not always mean you SHOULD!
• Long running transactions
  • Commit frequency is still important
• API issues
  • MQPUT1 in loops
  • No ‘fail if quiescing’
  • Non-expiring MQGETs
  • Poison messages

WMQ and CICS – Things to Avoid - Notes

• Long running transactions
  • Default uncommitted message count is 10,000
  • That builds up in storage
    • Even though it’s above the bar in WMQ V7, should not be abused
• API issues
  • MQPUT1 in loops
    • We estimated that at 3 MQPUT1s to the same queue in one transaction it becomes more expensive in CPU than an MQOPEN, loop of MQPUTs and MQCLOSE
  • Use backout thresholds and re-queue queues correctly
    • Avoiding poisoned message looping can save a lot
WMQ and CICS – Evolution (not revolution)

- CICS now fully supports the new WMQ V7 verbs
  - MQ PUB/SUB
    - CICS applications can publish to topic objects or topic strings
    - They can also be subscribers
  - Cooperative Browsing
    - Multiple applications can flag messages as having been 'touched'
  - Message Properties
    - Values associated with the message, but not part of the message body.
    - May be used as:
      - Selection Criteria
      - To drive application function without changing the message body
  - MQ Callback – aka the Async Consumer
    - CICS transactions can be started asynchronously based on selection criteria
    - Message data is passed in channels and containers
    - May remove the need for the CICS Bridge

WMQ and CICS – Evolution (not revolution) - notes

- MQ pub/sub is now available on 'all the major platforms' including z/OS
- Simple to use, allows further decoupling of applications
- Message properties
  - Lots of interest to provide:
    - Starting multiple transactions using MQCB based on selection criteria
    - Provide a processing 'marker' (I was here)
- Async consumer
- There are new COBOL samples with the latest WMQ and CICS PTFs
WMQ – CICS Samples using the new verbs

- CICS sample programs that illustrate async consume and pub/sub
- Asynchronous consume
  - Two message consumers and one event handler
  - CSQ4CVRG – Registers callback handlers
    - CSQ4CVEV – Event handler
    - CSQ4CVCN – Simple message consumer
    - CSQ4CVCT – Control message consumer
- Asynchronous consume and pub/sub
  - CSQ4CVPT – Two functions:
    - Publishes message to topic – consumed by CSQ4CVCN
    - Puts control messages to queue for consumption by CSQ4CVCT
- MVMP transaction allows user to control interaction

WMQ – CICS Samples using the new verbs

- These have been delivered with WMQ V7.0.1 maintenance
- ATS are also providing other single function samples available or in-flight:
  - QPUB - QPUBCBL
    - This program will publish a specified number of messages to the topic and/or topic string provided in a control message.
    - Published as a TechDoc
      - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4549
  - QSUB - QSUBCBL
    - This transaction will subscribe to a topic and/or topic string provided and read the specified number of publications.
    - In Progress
  - QDISP – QDISPCBL
    - This program will demonstrate cooperative browsing in dispatching work to other transactions.
    - In Progress
Loose Coupling with Publish/Subscribe

Loose coupling is a key factor in a true SOA environment

- My business process requests a service, that service can reside anywhere
- The services may even be asynchronous
- SOA has been around a long time for CICS users
  - I’ve been encouraging the use of CICS transactions as ‘services’ for more than 20 years
  - The names have changed
  - Now there are standards
- Pub/Sub is ideally suited for an SOA environment, because the data producer (the publisher) and the data consumer (the subscriber) can be anywhere.
Pub/Sub example

- Problem:
  - Three major applications with different data stores
  - Customer contact information is continuously out of sync
  - Customers don’t want to update information multiple times to change the same data
- Three primary applications:
  - DDA – CICS applications with VSAM data store
  - Mortgages – CICS applications using DB2 on z/OS
  - Consumer Loan – CICS applications using DB2 on z/OS

Pub/Sub example - notes

- This is an example keenly felt by the author
  - To get my cell phone number changed at a bank I had to make several trips to cover all my account types.
- While there clearly had been an attempt to integrate some of the applications, it failed.
  - When I altered my cell phone in the DDA system it updated there
  - It overlaid my home phone number on my mortgage account
  - When I repaired the mortgage account, my entire contact information got deleted on a consumer loan account, etc.
Pub/Sub and CICS sample

• Solution:
  • Create a ‘Customer Contact Change’ topic object
  • Publish changes:
    • Alter all systems to publish contact change information OR
    • Create a single ‘contact update’ application
      • *This could be a very simple browser based app*
  • Alter the update processes to subscribe to contact change information
    • If this is currently queue driven, no application changes are required
    • ‘Administered’ Subscriptions
  • When a change is made each application will receive a copy of the update and make the appropriate change to their back-end.

Pub/Sub and CICS sample - notes

• Solution:
  • This simple solution assumes that the data for all three systems is virtually the same, which is not always the case. There are times when you must use a brokering tool to change formats for each of the systems. Again, the broker – the data transformation tool – can be the subscriber and can reformat the update request as needed.
Prior to Pub/Sub

DDA Transactions  DDA VSAM
Update Mobile Number  Update Contact Request
Customer Update Request

Update Contact Number Request

Mortgages

DB2 Databases

Consumer Loans
Update Contact Information

Pub/Sub Sample solution

DDA Transactions  DDA VSAM
MQ Subscribe
MQ Publish
MQ Subscribe
MQ Subscribe

Customer Contact Update
MQ Publish
MQ Subscribe

Mobile Number Update
Mobile Number Update

WMB Data Transformation

Mortgages

DB2 Databases
WMQ and CICS – To Evolve

- Apply the correct WMQ and CICS maintenance
  - WMQ - APARs PK97364 and PK97972
  - CICS V4.1 - APAR PK89844
- Remove the MQ INITPART from CICS
- Add the new CICS MQCONN resource definition
- Upgrade the CICS resource definitions to include the new features
  - Run DFHCSDUP with the UPGRADE USING(DFHCURDM) command

Enablement PTFs

- What are they
  - PTF to enable WMQ V7.0.x new function in CICS TS 3.2 and 4.1
- What do they enable
  - API verbs and new function support for:
    - PUBLISH/SUBSCRIBE
    - ASYNCHRONOUS CONSUME
    - MESSAGE PROPERTIES
- APARS/PTFs
  - CICS TS 3.2 – PK66866 (UK52671,UK52672,UK52673,UK52680)
  - CICS TS 4.1 – PK89844 (UK52619,UK52667,UK52668,UK52669)
- What happens if they’re not applied
  - Function can’t be exploited from CICS environment
  - Just linking with new stub doesn’t mean it will work
  - Applications will be returned MQRC_FUNCTION_NOT_SUPPORTED
    - Or the AMQC abend
Enablement PTFs - notes

- Check for the most recent updates
  - The full function was added in Dec. 2009/Jan 2010
  - If you have a fix prior to that date, you may be missing a few things

CICS WMQ Abend codes

- AMQA DFHMQCON had enabled DFHMQTRU with a global work area smaller than that needed by DFHMQTRU. This could be due to a mismatch of version level between DFHMQCON and DFHMQTRU.
- AMQB DFHMQCON had enabled DFHMQTRU with a task local work area smaller than that needed by DFHMQTRU. This could be due to a mismatch of version level between DFHMQCON and DFHMQTRU.
- AMQC Unrecognizable WMQ API call. All supported API calls are documented in the WebSphere MQ Application Programming Reference manual.
- AMQD Unrecognizable RMI API call. The CICS-MQ task related user exit (TRUE) was invoked with an unrecognizable request type.
- AMQE An attempt to EXEC CICS LOAD the data conversion service module CSQAVICM was unsuccessful.
- AMQF An internal logic error has been detected in the CICS bridge monitor.
- AMQG The CICS DPL bridge program has detected an error in a request message for this unit of work.
- AMQH The CICS bridge monitor or DPL bridge program abended due to an unexpected return code from an EXEC CICS API call.
- AMQI The CICS bridge monitor or DPL bridge program abended due to an unexpected return code from an MQ API call.
- AMQJ The CICS DPL bridge program abended before processing any messages for the unit of work.
- AMQK The CICS DPL bridge program abended during error processing.
CICS WMQ Abend codes - notes

- Can you guess which one is my favorite?

- AMQC can occur when:
  - You try to use the new functions, but the supporting maintenance has not been applied
  - You have linked with the wrong version of the stub

What do you mean my MIPS didn’t go down?

- A common pattern has emerged
  - A promised ‘MIPS reduction of 10-20%’
  - Same Qualities of Service
  - No re-write (or a very limited one) of your business logic
  - Pulling transactional workload off z/OS and putting it on a distributed platform
- Some workloads are a very good fit for moving
- This is the tale of some that are definitely not!
What do you mean my MIPS didn’t go down? - Notes

• There have been a number of failed customer projects
• ATS has been asked for a post mortem on several
  • There are some workloads that can be easily and effectively moved, while others cannot
  • In some cases these failed solutions have been adopted, with the limitations and availability issues left un-addressed

• This information is being presented as a cautionary tale, your situation may differ from this pattern

MIPS reduction: Problem #1

• The process that was being removed was using DB2 data sharing and WMQ shared queues for extreme availability
  • This transaction had a firm 24/7 availability requirement
  • There were 6 queue managers, 6 DSG members and 24 CICS regions spread across 2 CECs
  • Transactional control was provided by CICS
• DB2 data sharing and WMQ Shared queues are only supported on z/OS
MIPS reduction: Problem #1

- To provide the same qualities of service, DB2 connect and WMQ clients were chosen to access the DSG and WMQ Shared queues
  - For both DB2 and WMQ, executing the API calls from a client is more expensive than executing those same calls locally
  - This also reduced some levels of control
- To achieve the same levels of availability:
  - Much more hardware had to be purchased than originally planned
    - 6 ‘large scale’ production UNIX machines
    - 2 QA UNIX machines
    - More software licenses, including a HACMP-like product
    - Two additional administrators were required
    - DR solution had to be devised
    - Network became more complex

MIPS reduction: Problem #2

- The typical transaction volume for a non-peak day was 5M per hour
  - Each CICS region remained constantly connected to it’s queue manager
  - Each CICS region had anywhere from 10 to 40 instances of the processing transactions available at all times
  - Each transaction processed as follows:
    - MQGET request message
    - Perform 1 DB2 query
    - Create 1 to 5 DB2 adds/updates depending on the transaction
    - MQPUT reply message
    - MQPUT audit message (if required)
    - MQPUT ‘additional attention’ message (if required)
    - SYNCPOINT
    - Loop back
MIPS reduction: Problem #2

- Without CICS the process for each request included:
  - Establish XA Unit of work
  - Connect to DB2
  - MQCONN – to gain access to the queue manager
  - MQGET request message
  - Perform 1 DB2 query
  - Create 1 to 5 DB2 adds/updates depending on the transaction
  - MQPUT reply message
  - MQPUT audit message (if required)
  - MQPUT ‘additional attention’ message (if required)
  - Request commit (whoever the TX manager was)
  - MQDIS – to disconnect from WMQ
  - Disconnect from DB2
  - Start next instance

- The additional workload from the client connections to both DB2 and WMQ increased MIPS usage by 15-20% before stress tests were run.

MIPS reduction: Problem #3

- This is certainly an z/OS centric view
- No CICS/WMQ/DB2 evaluations was done
  - In the post mortem, it was found that the customer could probably have achieved a significant MIPS reduction by:
    - Tuning the infrastructure – there were significant issues with some of the set-up that had never been addressed
    - Tuning the application code – while most of the code was quite good, there was some that could have used some work
    - Upgrading to more current releases of hardware
Summary

• Talked about why and how
• Mentioned things to avoid
• Example of where the new verbs can be useful
• Example of a failed ‘moving off platform’
• Got forth and remember that CICS and WMQ have been a winning combination for most of WMQ’s life!