Performance report – SupportPac MP1J
WebSphere MQ Version 8.0 for z/OS

Advanced Message Security

Version 1.0

WebSphere MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

Take Note!

Before using this report, please be sure to read the paragraphs on “disclaimers”, “warranty and liability exclusion”, “errors and omissions” and other general information paragraphs in the “Notices” section below.

First edition, September 2014.

This edition applies to WebSphere MQ for z/OS version 8.0.0 (and to all subsequent releases and modifications until otherwise indicated in new editions).

Notices

DISCLAIMERS

The performance data contained in this report were measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends on the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.
In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS

The information set forth in this report could include technical inaccuracies or typographical errors.
Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

INTENDED AUDIENCE

This report is intended for Architects, Systems Programmers, Analysts and Programmers wanting to understand the performance characteristics of **WebSphere MQ version 8.0 for z/OS**. The information is not intended as the specification of any programming interfaces that are provided by WebSphere MQ. Full descriptions of the WebSphere MQ facilities are available in the product publications. It is assumed that the reader is familiar with the concepts and operation of WebSphere MQ.

LOCAL AVAILABILITY

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES

Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS and SERVICE MARKS

The following terms, used in this publication, are trademarks or registered trademarks of the IBM Corporation in the United States or other countries or both:

- IBM®
- z/OS®
- zSeries®
- zEnterprise®
- MQSeries®
- CICS®
- DB2 for z/OS®
- IMS™
- MVS™
- FICON®
- WebSphere®

Other company, product and service names may be trademarks or service marks of others.

EXPORT REGULATIONS

You agree to comply with all applicable export and import laws and regulations.

Summary of Amendments

Date	Changes
September 2014	Initial Version

Table of Contents

Advanced Message Security

Advanced Message Security (AMS) extends the security features available in WebSphere MQ by providing end-to-end message level security through signing and encryption. It can help ensure the integrity and privacy of all messages, even when they are at rest in queues.

To enable AMS protection in version 8.0, the CSQ6SYSP macro now supports the SPLCAP parameter. This parameter determines whether the queue security policy capability is enabled. When the SPLCAP parameter is set to YES, the AMS region will be automatically started when the queue manager is started.

In version 8.0 there is a closer integration between AMS and the queue manager to such a degree that when AMS is configured via the SPLCAP parameter, the queue manager will not run without the corresponding AMS task.

AMS protection is defined at an MQ queue level using a policy via the “setmqspl” program.

This additional protection does however bring an increase to processing cost and storage usage. This document will discuss the costs and impact of AMS on messaging performance.

Testing methodology

AMS performance was measured using 5 basic configurations, as described in Appendix A.

Baseline measurements were run where AMS was not enabled and compared to measurements when:

- AMS was enabled but no policies were defined
- AMS was enabled and policies were defined for the queues in use.

For measurements using a policy, the default policy options used were:

- digital encryption algorithm of “3DES”
- digital signature algorithm of “SHA1”
- 1 recipient
- no validation (authentication) of the signature distinguished name (DN) specified.

Does AMS affect the performance of an unprotected queue?

Yes. If the queue manager has AMS enabled, then it must determine whether the MQ resource being used is protected by a policy.

Using the 5 configurations specified in Appendix A, the cost associated with the application increased by between 2.5 and 10 microseconds per API when AMS was enabled with no policies defined. For a workload with no business logic using 2KB non-persistent messages this added 10% to the transaction cost.

Due to the closer integration of AMS with the queue manager in version 8.0, this additional cost is much less than previous releases, which was typically 25 CPU microseconds per API.

Given the increase in CPU cost associated with checking for a policy, it is not advisable to enable AMS on a queue manager unless necessary.

How much difference does adding end-to-end security to a transaction make?

The simple answer is that it can make a significant amount.

If we take the most simple test configuration measured, namely a request / reply model using long running batch transactions that do little processing other than messaging, then compare the transaction rate and costs we get charts like the following:

Chart 1: Batch workload transaction Rate

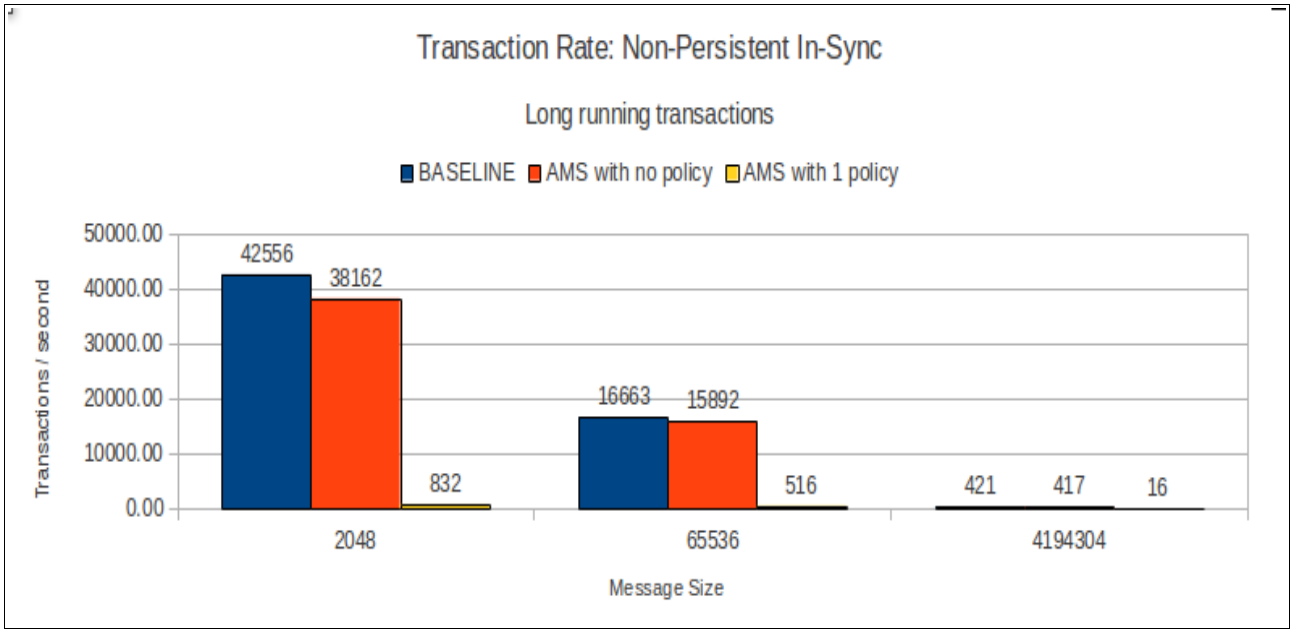
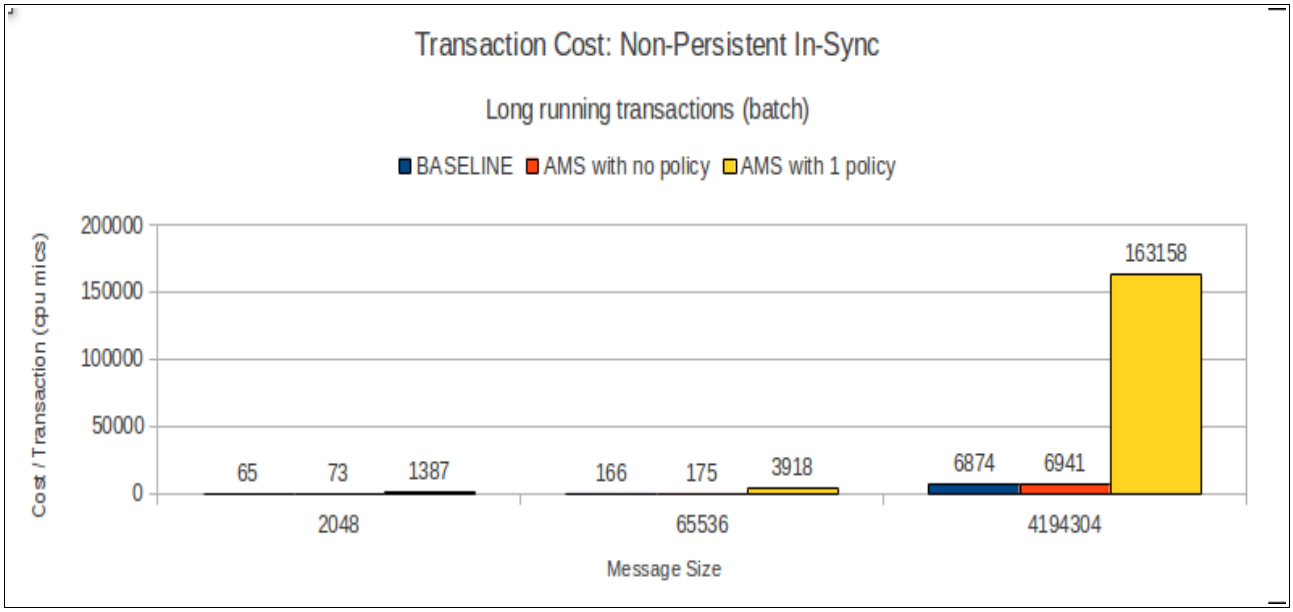


Chart 2: Batch workload transaction Cost



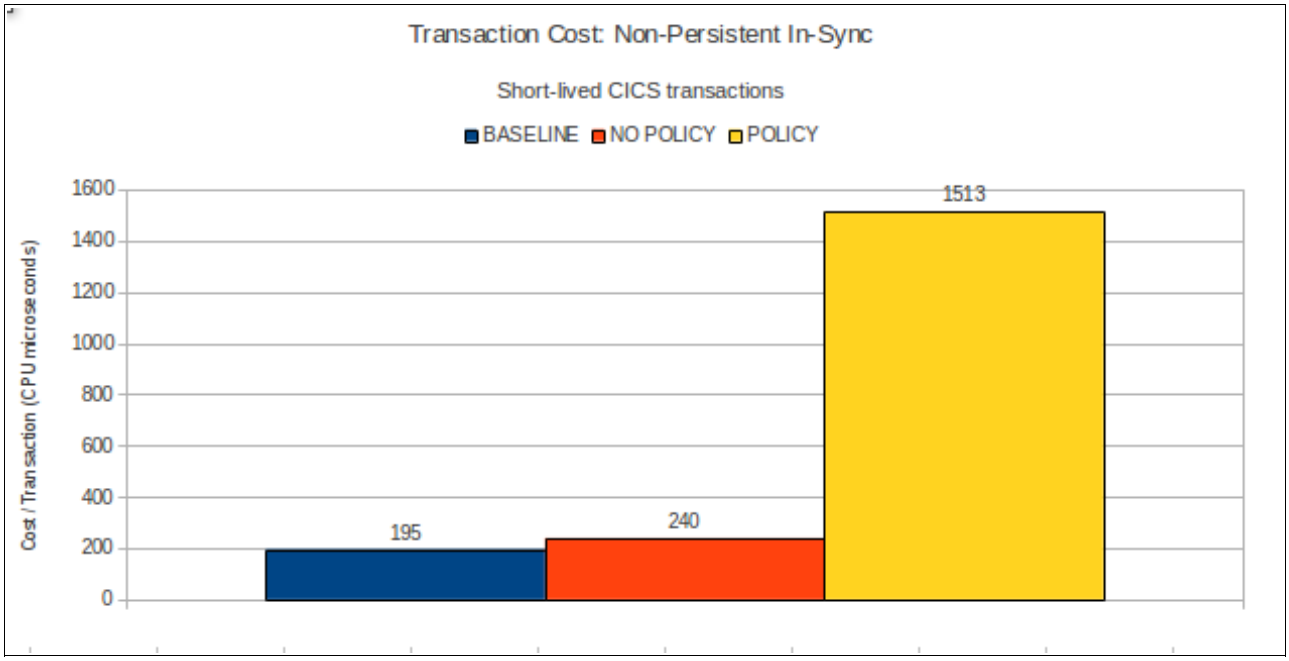
Note the decrease in transaction rate and increase in transaction cost does not necessarily correlate, as a message protected by an AMS policy will also see time spent in cryptographic processing, where the cost may be offloaded to hardware, and security checking (e.g. RACF).

In this particular scenario, the costs associated with securing the messages increased the transaction cost approximately 23 times over the unsecured message regardless of the message size. It should

be noted that the messaging costs in the baseline measurement do form over 95% of the transaction cost.

In the case where the non-messaging processing costs form a larger part of the total transaction cost, for example a short-lived CICS transaction, the relative effect of protecting the message with an AMS policy is much smaller. In the following chart the effects of AMS means the transaction cost is increased 7 times over the baseline measurement.

Chart 3: CICS transaction cost



Note that the CICS transaction puts and gets 1 message, whereas the batch transaction involves 2 messages being put and gotten (the request and reply message). As a result the batch and CICS transaction costs are not directly comparable.

Does defining a policy on a queue affect the size of the message on that queue?

Yes. An additional header is added to the message. Furthermore the type of digital signature, the encryption algorithm and number of recipients can affect the size of the message.

The following table is intended as a guide for the overheads of the various configurations of message protection offered by AMS.

In each case the message is set to 2048 bytes and the sizes reported are based on accounting class(3) data.

	Digital Signature Algorithm	**Digital Encryption Algorithm**	**Authentication**	**Recipients**	**Increase in message size (bytes)**
Baseline	-	-	-	-	0
Signed	NONE	NONE	-	-	0
Signed	MD5	NONE	-	1	1155
Signed	SHA1	NONE	-	1	1153
Signed	SHA256	NONE	-	1	1173
Signed	SHA512	NONE	-	1	1206
Encryption	MD5	RC2	-	1	1432
Encryption	MD5	DES	-	1	1423
Encryption	MD5	3DES	-	1	1426
Encryption	MD5	AES128	-	1	1443
Encryption	MD5	AES128	-	1	1443
Authenticate	SHA1	3DES	1 of 1	1	1426
Authenticate	SHA1	3DES	1 of 3	3	1867
Recipients	SHA1	3DES	0	3	1867

In summary:

- Signing the message increased the message size by between approximately 1150 and 1200 bytes.
- Encrypting the message added a further 280 bytes.
- Authenticating the message did not make a difference to the size.
- Adding more recipients to the policy increased the size of the message by approximately 220 bytes per recipient.

How does version 8.0 compare with version 7.x?

The performance of AMS protected queues on WebSphere MQ version 8.0 is broadly similar to previous releases, however when closing a queue there is an additional call to the security management product e.g. RACF, which can result in increased cost from MQPUT1 or when there are few MQPUT / MQPGET requests per MQCLOSE.

Long running batch tasks using private queues using comparable policy definitions on each release showed similar performance.

Chart 4: Batch transaction rate comparison – release on release

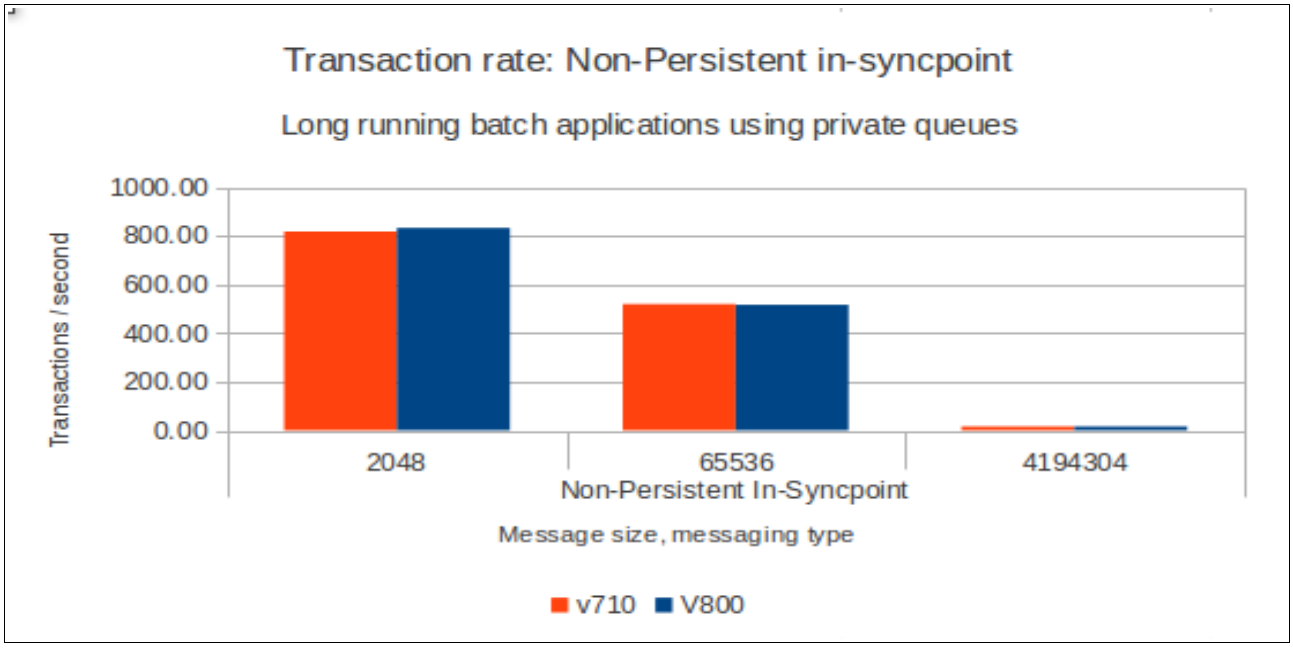
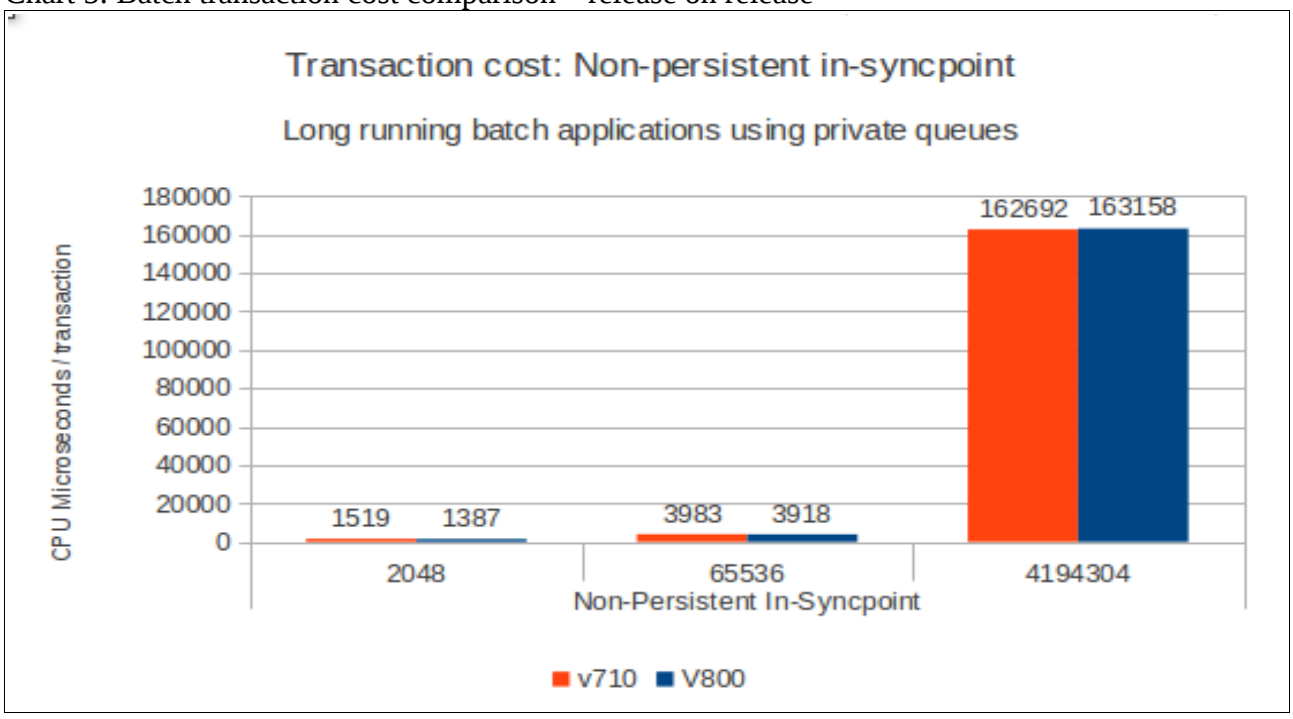
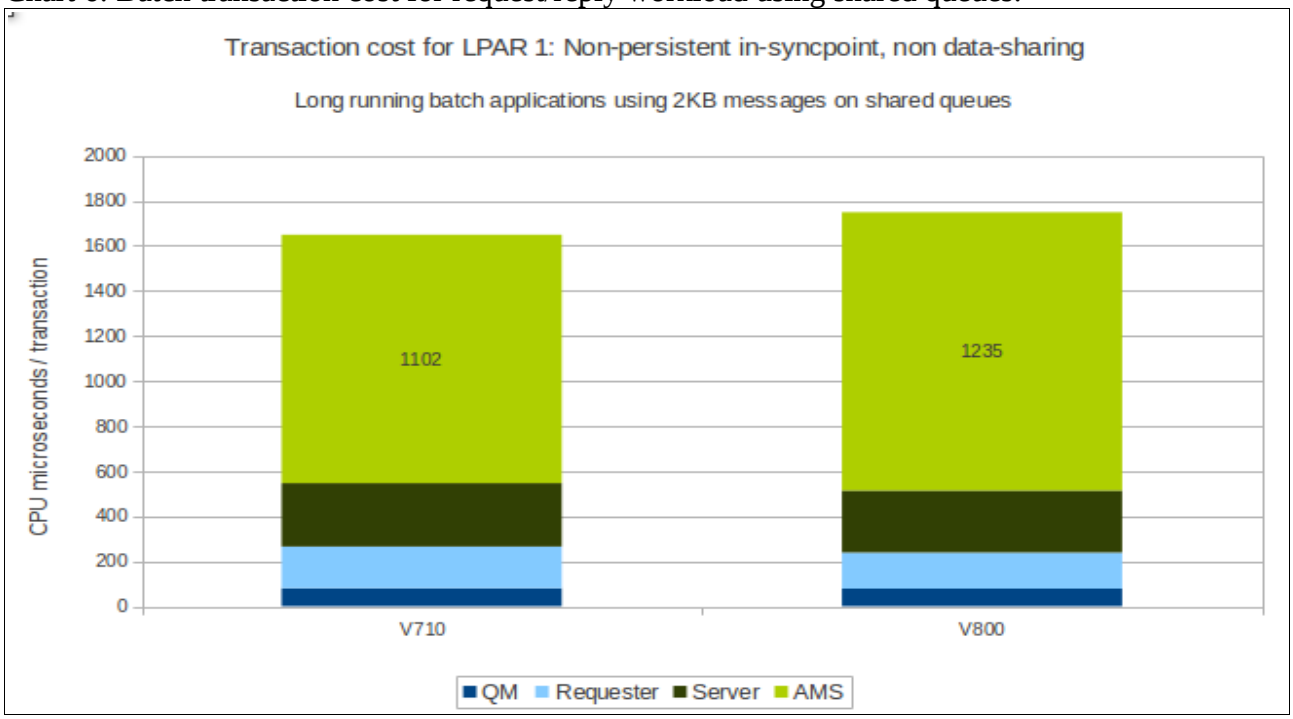


Chart 5: Batch transaction cost comparison – release on release



Long running batch tasks using shared queues with comparable policy definitions on each release show a decrease in performance which is due to the server applications using MQPUT1 for each reply message, which as previously mentioned includes an additional call to the security manager than in version 7.

Chart 6: Batch transaction cost for request/reply workload using shared queues.



Moving messages between queue managers over MCA channels using comparable policy definitions on each release showed similar performance.

Chart 7: Transaction rate when moving messages between z/OS queue managers

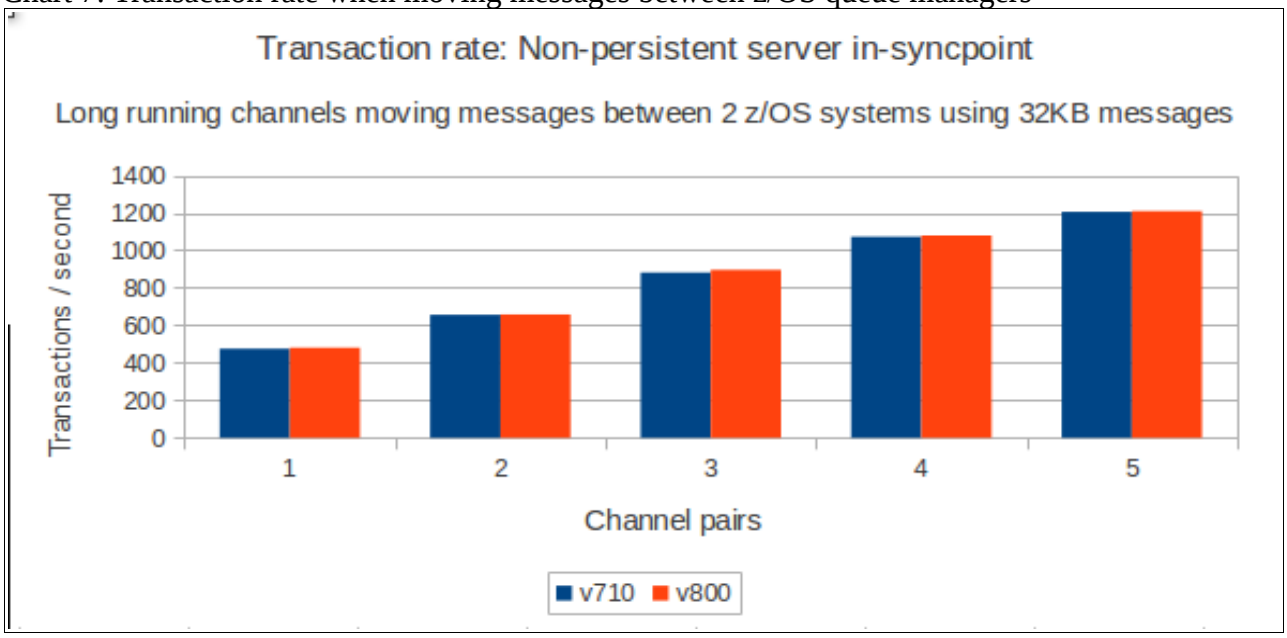
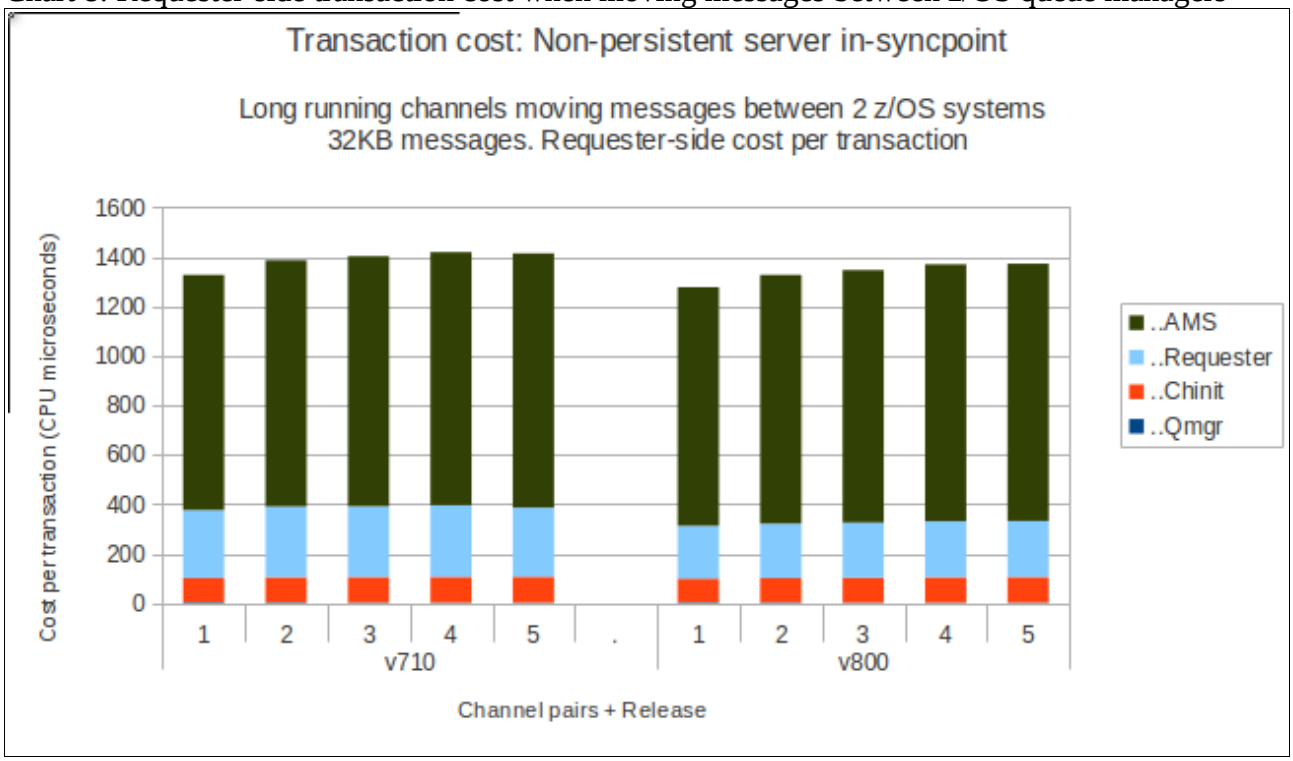
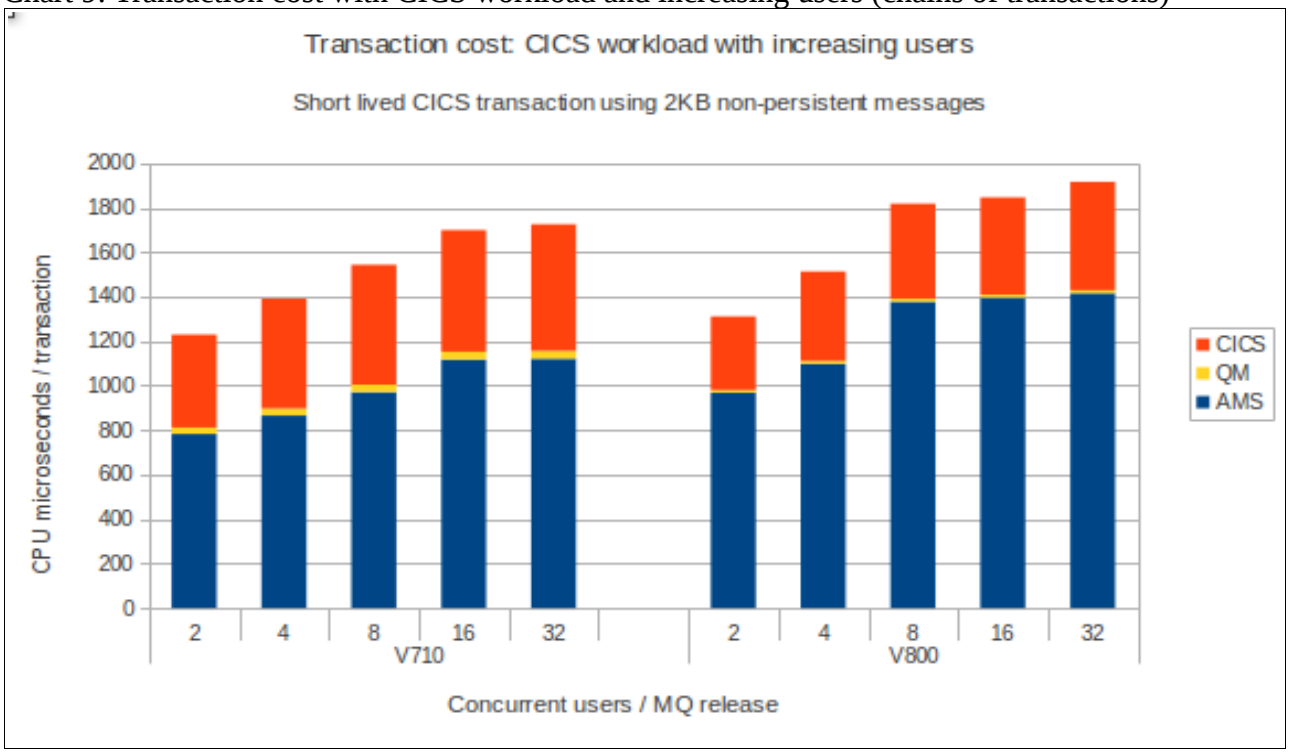


Chart 8: Requester-side transaction cost when moving messages between z/OS queue managers



For CICS workload on private queues with comparable policy definitions, the performance is slightly worse, in particular at MQCLOSE.

Chart 9: Transaction cost with CICS workload and increasing users (chains of transactions)



CICS triggering workload on private queues with comparable policy definitions showed similar performance, although again the trigger every measurement saw an increase in AMS cost in the more frequent MQCLOSE calls.

Chart 10: Transaction cost with CICS workload using TRIGGER(FIRST)

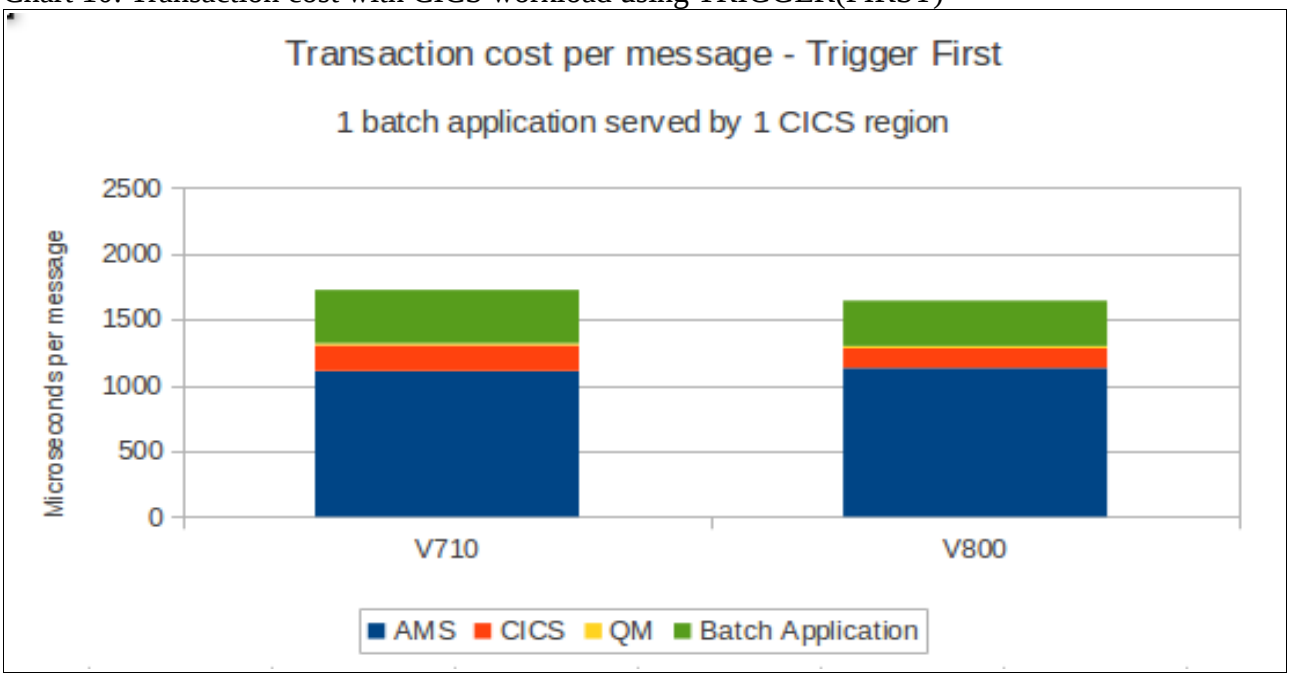
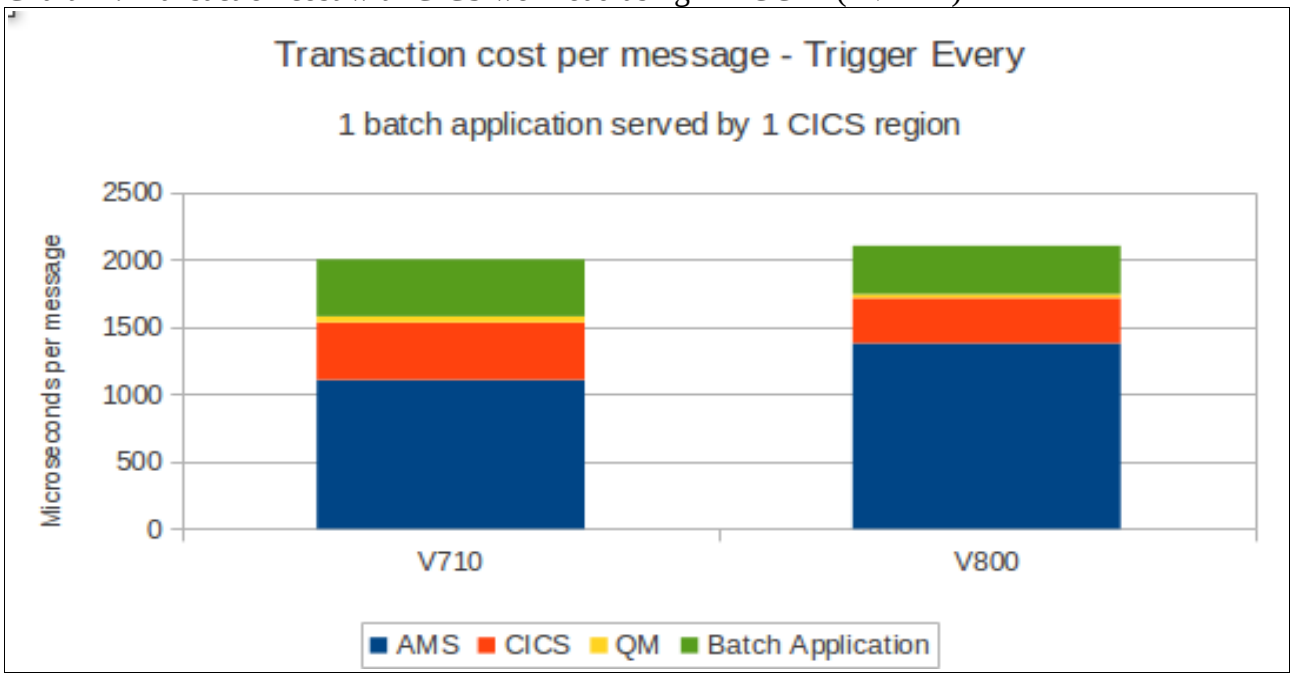


Chart 11: Transaction cost with CICS workload using TRIGGER(EVERY)



Which address spaces are charged for in the use of AMS?

As AMS runs on a TCB, the costs are incurred by the application and the AMS address space. By the time the message reaches the domain of the queue manager, it has been processed by AMS, which may have encrypted, signed the message and increased the message size.

Once the message is under the control of the queue manager, it is treated the same as an unprotected message, albeit slightly larger.

Does the type of digital signature algorithm chosen affect cost?

When a policy dictates that a message is signed, there is a significant increase in the cost of the message. In the case of a long running request/reply workload running batch applications with minimal non-MQ related processing, the transaction cost increased approximately 15 times.

Varying the signing type between MD5, SHA1, SHA256 and SHA512 saw the largest increase with MD5, particularly with larger messages, as demonstrated in the following charts.

Chart 12: Achieved transaction rate whilst varying digital signature algorithm

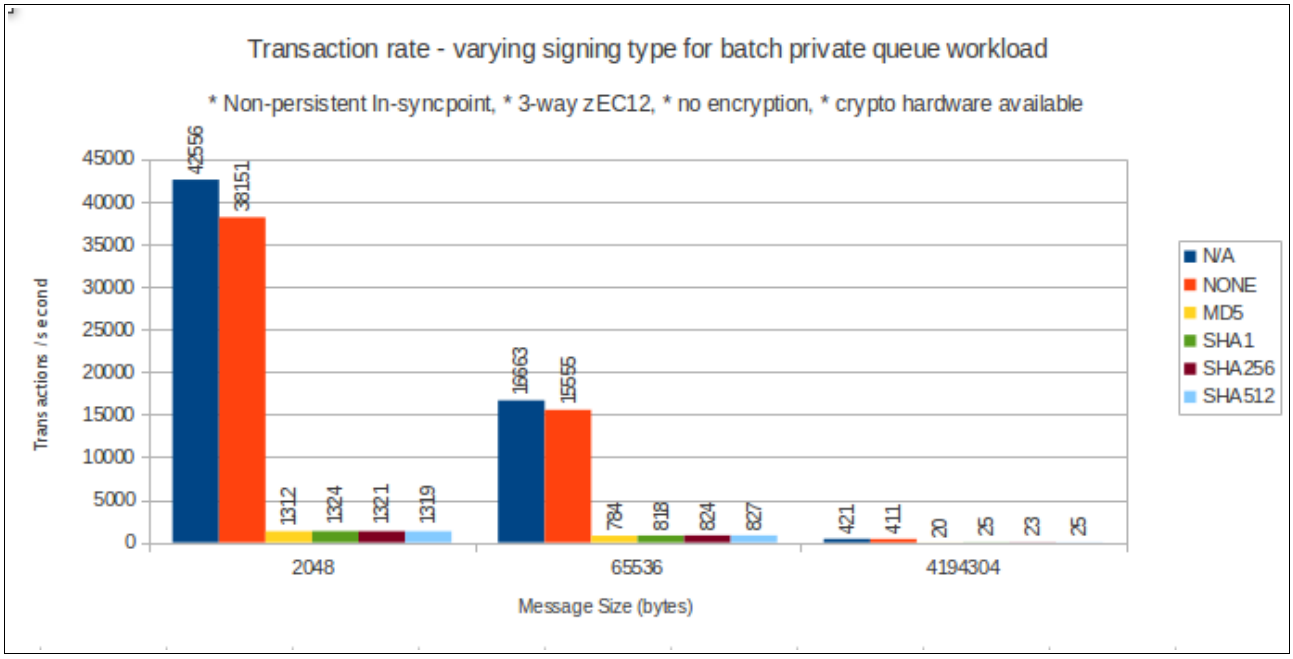


Chart 13: Transaction cost of 2KB message whilst varying digital signature algorithm

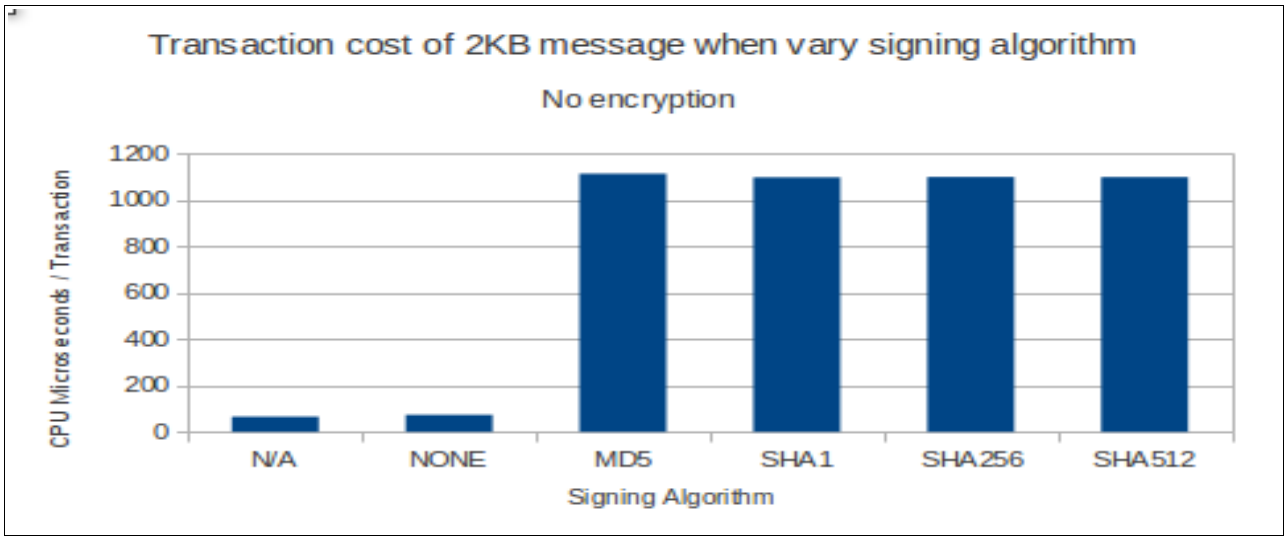


Chart 14: Transaction cost of 64KB message whilst varying digital signature algorithm

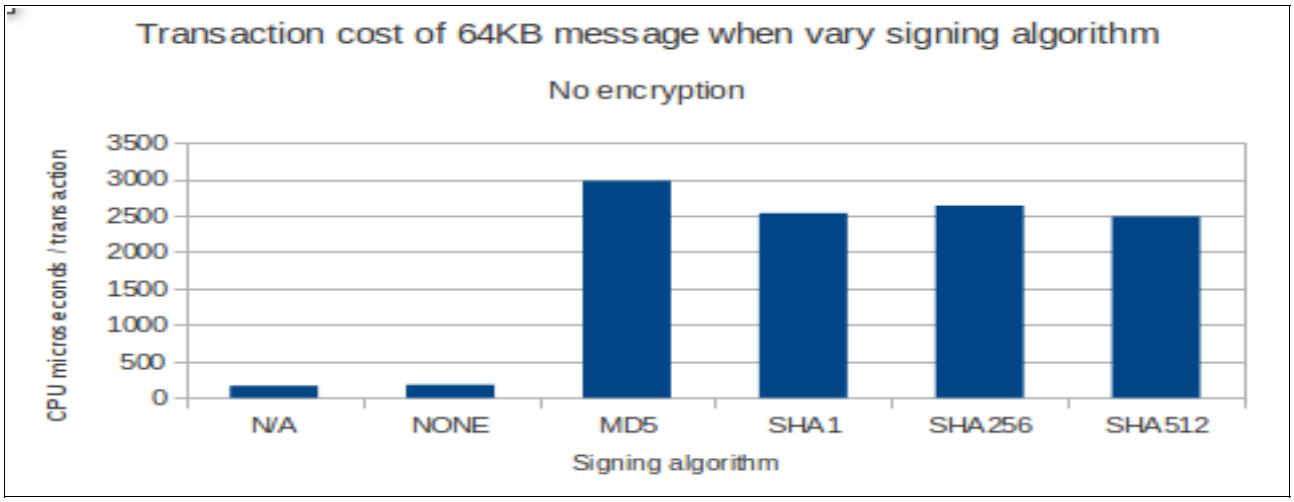
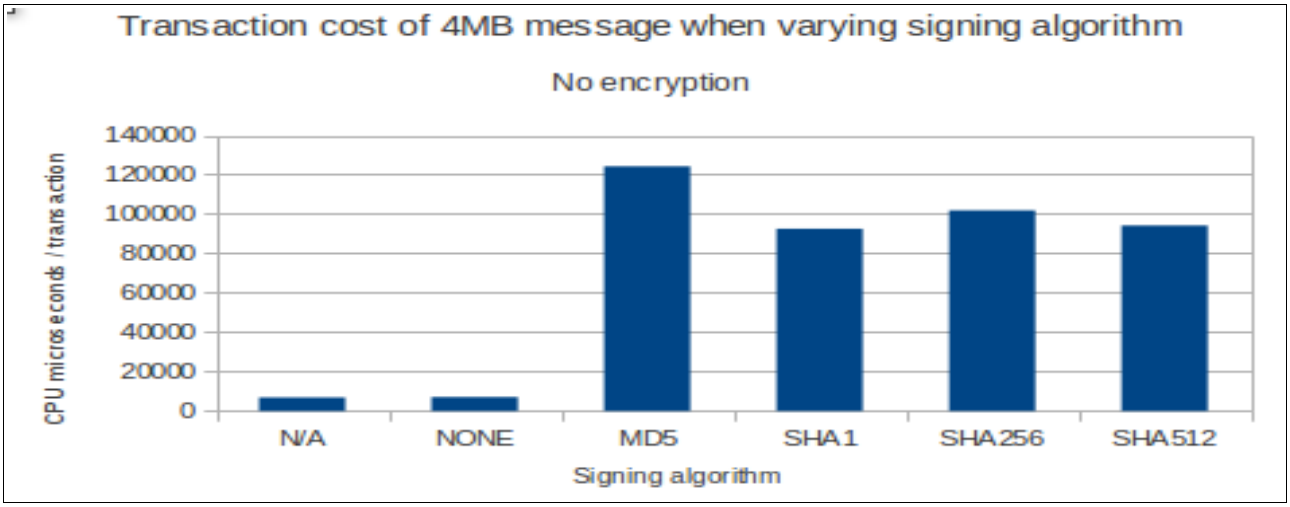
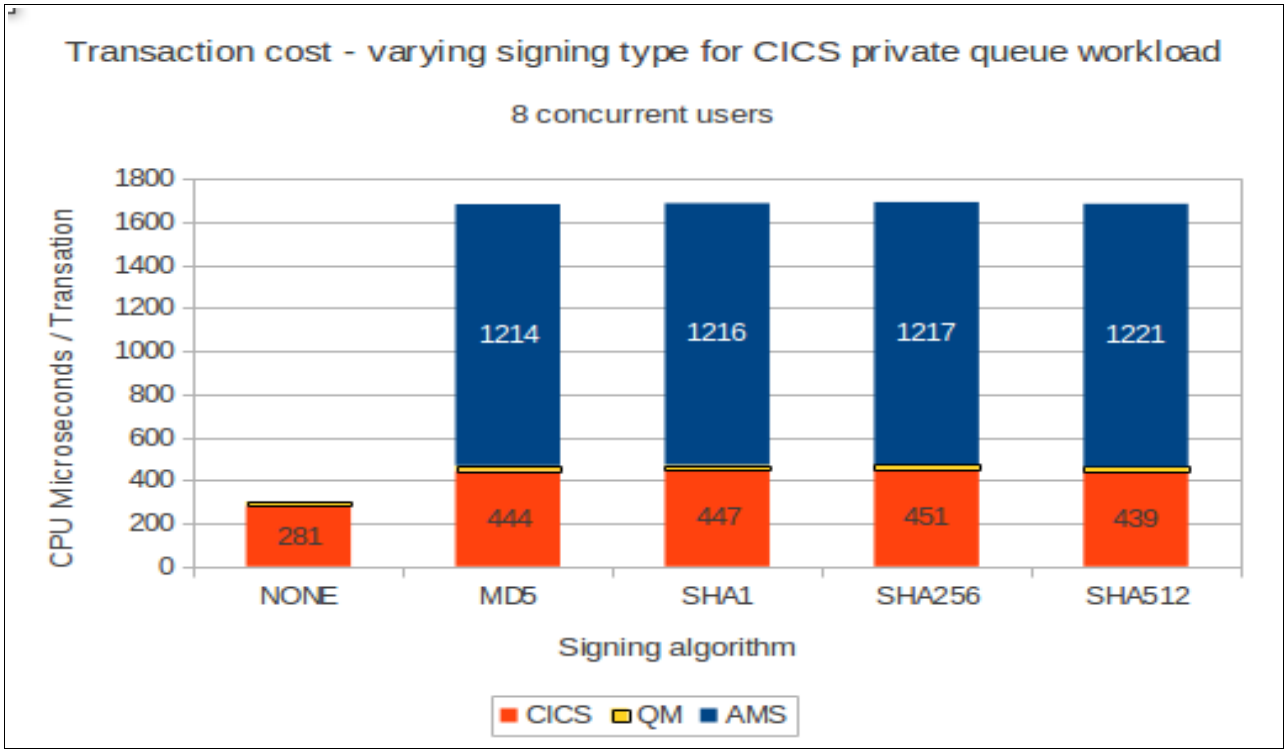


Chart 15: Transaction cost of 4MB message whilst varying digital signature algorithm



In a CICS workload environment, once the message was signed, the actual type of signing used made little difference to the transaction cost as can be seen in the following chart.

Chart 16: Transaction cost of 2KB message in CICS environment whilst varying the digital signature



In summary:

- The additional cost incurred when signing a message means that in these tests the workload was constrained for CPU.
- The digital signature algorithm applied makes relatively little difference to the achievable transaction rate – when CPU constrained
  - Signing with MD5 was more expensive than SHA* for larger messages.
- Signing a message can use the cryptographic co-processor(s) when available.
- ICSF usage and execution time for both co-processors and accelerator on Crypto Express-4S remains similar across signing types.

Does the type of encryption affect the cost?

In order to encrypt a message, it must also be signed. There is an additional cost incurred when encrypting a message on top of the costs from signing the message and this cost is similar regardless of encryption type, except for RC2.

The following table gives an indication of the increased cost of encryption compared to a message that is only signed in a batch environment.

Encryption Type	Message size 2KB	Message size 64KB	Message Size 4MB
DES 3DES AES128 AES256	1.2 times	1.3	1.4
RC2	1.5	2.2	3.2

Notes on table:

This means that a 64KB message that is signed and encrypted using AES128 uses typically 1.3 times more CPU than a 64KB message that is just signed.

A 64KB message that is encrypted using RC2 uses typically 2.2 times more CPU than a 64KB message that is only signed.

Does the number of recipients affect the cost?

When a small number of recipients are able to decrypt a message, we saw little difference in cost whether the recipient was the first or last in the list.

For example, using the policy below, we saw similar transaction costs whether user “MQSYST1” or “MQSYST4” was getting (and decrypting) the message.

```
setmqspl -m MQ01 -p QUEUE1 -s SHA1 -e 3DES
         -r "CN=MQSYST1,O=IBM,L=HURSLEY,C=UK"
         -r "CN=MQSYST2,O=IBM,L=HURSLEY,C=UK"
         -r "CN=MQSYST3,O=IBM,L=HURSLEY,C=UK"
         -r "CN=MQSYST4,O=IBM,L=HURSLEY,C=UK"
         -t 1
```

Does message authentication impact performance?

As more author distinguished names (DN) are added to the policy, there is an increase in the time spent in AMS to process these names.

On our systems, for each author DN, the time spent in MQPUT or MQGET increased by 27 microseconds for a 1KB non-persistent message.

For example, a policy is defined with 1 author DN and the AMS costs are 540 microseconds per put and get. If a further 2 more author DN's are added to the policy, the AMS cost would increase by:

```
Users * (MQPUT + MQGET)
```

i.e. 2 * (27 + 27) = 108, which gives a total AMS cost of 648 microseconds.

Tuning for AMS: Use the right sized MQGET buffer.

The AMS address space needs to allocate a buffer to receive the message, whether from the application or the queue manager, and a second buffer to store the encrypted/decrypted message.

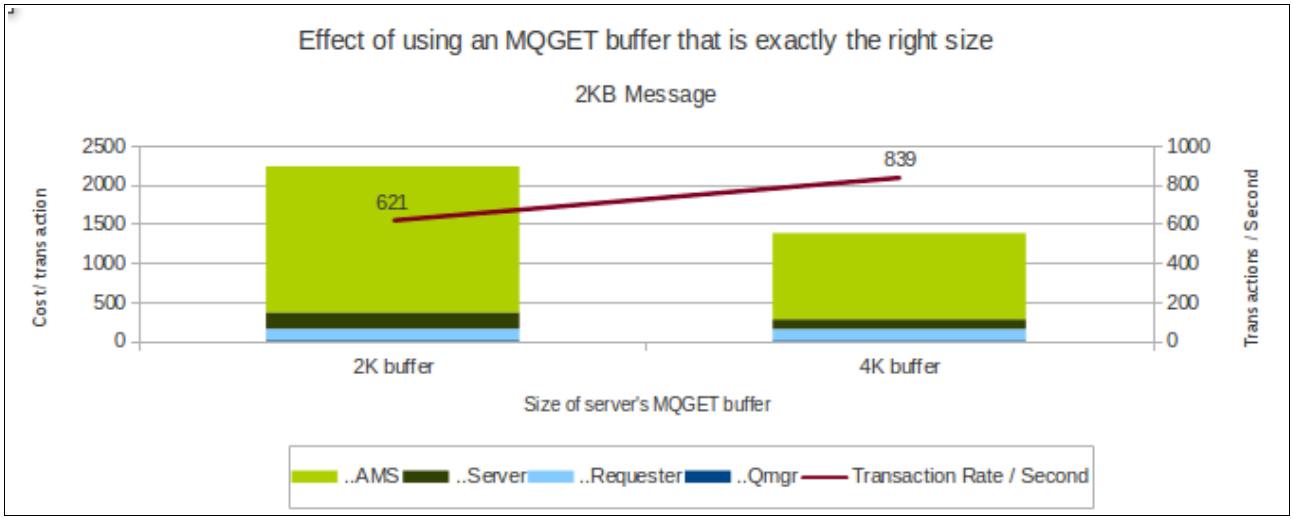
At MQPUT time, these buffers can be allocated the appropriate size since the message size is known.

When an MQGET is requested, AMS is reliant upon the value of *BufferLength* on how to size the buffers.

As mentioned in "Does defining a policy on a queue affect the size of the message on that queue?", the size of the message stored on a queue will not necessarily be the same size as the message originally put. If the getting application is expecting a fixed size message, it may specify a *BufferLength* that is large enough for the unprotected message but not large enough for the protected message.

If the *BufferLength* passed into AMS is not large enough to hold the longer encrypted message, then AMS will have to re-allocate its buffers the right size. This can have a noticeable affect on transaction rate and cost. For example consider a request / reply workload where the server task specifies an MQGET *BufferLength* of the exact size of the message and compare this with a buffer that is large enough to hold the encrypted message.

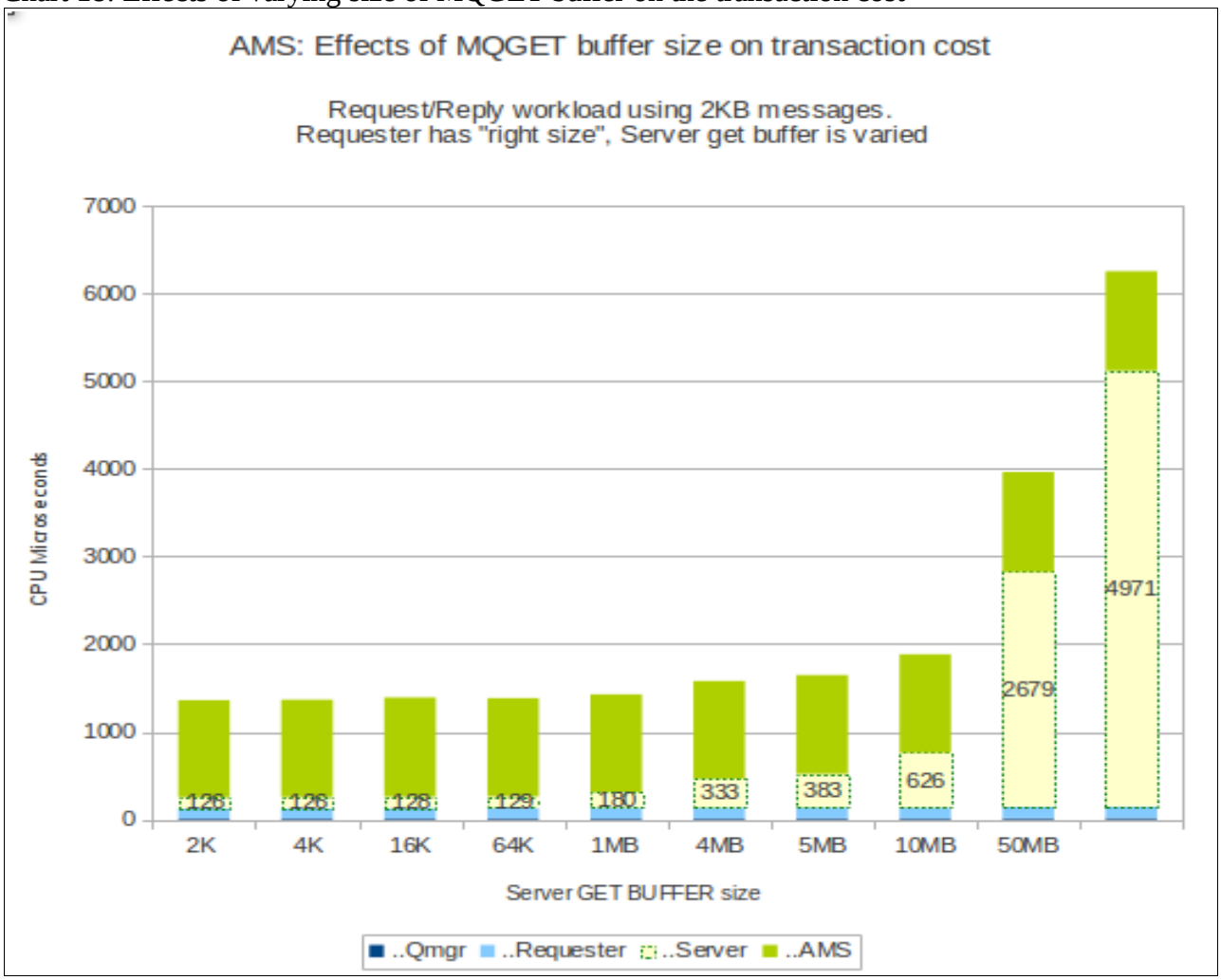
Chart 17: Effect of the specifying *bufferLength* precisely at the message size



In the above example, the application was expecting a 2KB message – however the encrypted message was actually 3KB, so the AMS region allocated its buffers at 2KB, determined they were not large enough and had to re-allocate the buffers for each MQGET.

Conversely, specifying a buffer that is much larger than is required can also impact the transaction cost significantly. The following chart shows the effect on transaction cost when using varying size buffers in the server transaction.

Chart 18: Effects of varying size of MQGET buffer on the transaction cost



Notes on chart 18:

- Using a buffer up to 64KB for a 2KB message had little effect on the transaction cost.
- Once the buffer size increased to 1MB, there is a noticeable increase in transaction cost.

In summary,

- Don't make the MQGET bufferLength just the right size for the unprotected message
- Don't make the MQGET bufferLength too large
- If the size of the message is not known, it can be more efficient to specify a small buffer and on receipt of “MQRC_TRUNCATED_MSG_FAILED”, to retry the MQGET with a larger buffer.
- In our tests where the message size was known, we found best performance to be achieved with an MQGET BufferLength to be the data length + 4KB

Tuning for AMS: Run sufficient AMS threads

Ensuring sufficient AMS threads are available to provide concurrent cryptographic functions is a key part to ensuring that messaging rate is not constrained due to waiting for a task.

The environment variables "_AMS_INIT_THREADS" and "_AMS_MAX_THREADS" are used to specify the initial number of AMS threads and the maximum number of threads that AMS will start when required.

The default settings are shown below which your site may have changed:

```
_AMS_INIT_THREADS=20
_AMS_MAX_THREADS=100
```

There is little impact of having more threads than required, however too few threads can impact the transaction rate.

Chart 19: CICS throughput when constrained for AMS threads.

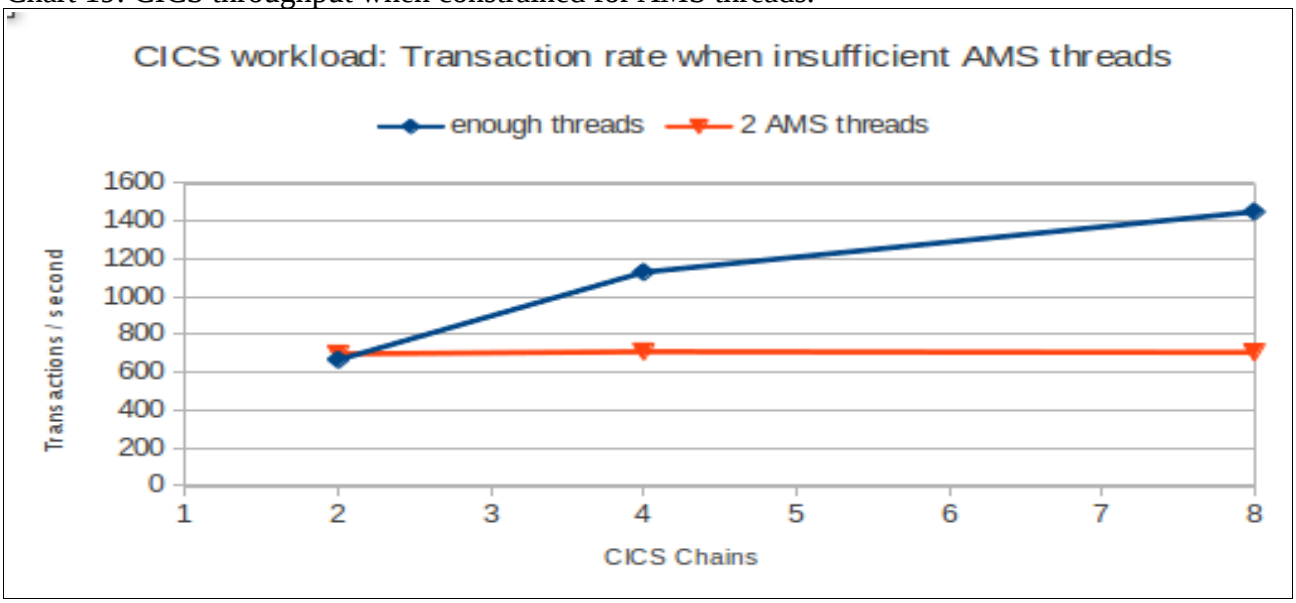


Chart 19 uses test configuration 4 as described in Appendix A.
When AMS is configured with both "_AMS_INIT_THREADS" and "_AMS_MAX_THREADS" set to 2, the workload is constrained, waiting for AMS tasks, which limits the workload rate to only 50% of the measurement with sufficient threads.

There may also be an increase in the CPU usage in the application address space and the AMS address space in the constrained case. In our measurements we saw the transaction cost increase by up to 40%.

How do you determine whether you are constrained for AMS threads?

It is better to have too many AMS threads than not enough, but in the absence of AMS instrumentation it is difficult to determine the minimum number of threads for a particular site. There are several options available -

1. Trial and error – increase the number of threads until the workload rate does not increase.
2. An instrumentation tool such as “IBM Application Performance Analyzer for z/OS” (APA) can be used to monitor the AMS address space.
   APA provides a “TCB summary” report which shows how many of the tasks have used CPU. Ideally there should be at least 1 task that has not used CPU.

Tuning for AMS: Minimise idle getters

When an MQGET is requested against an queue protected with an AMS policy, AMS will allocate 2 buffers, one for the encrypted message retrieved from the queue and one to receive the decrypted message into.

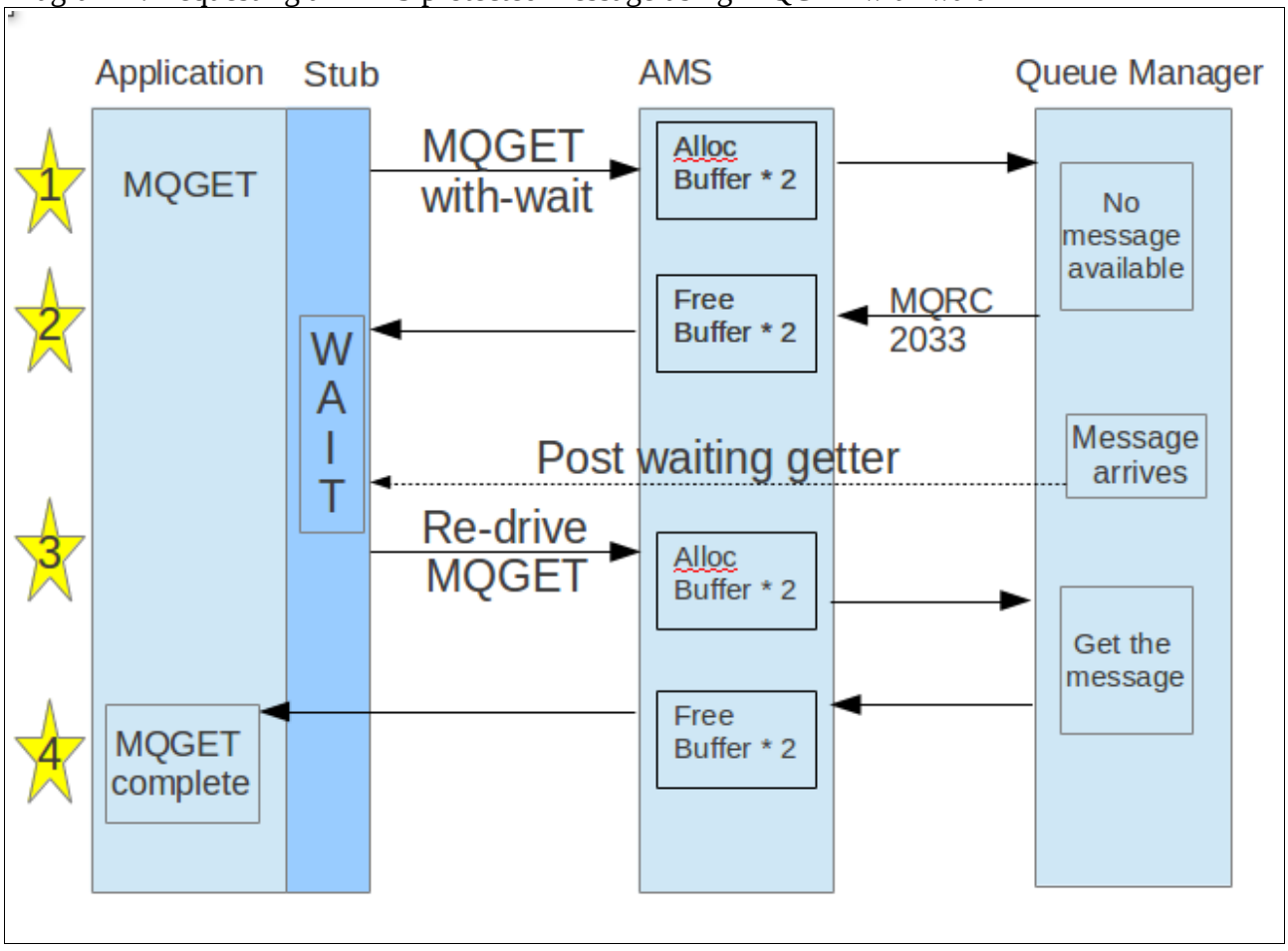
These buffers are allocated each time the MQGET is driven and released once the flow returns to the MQ stub.

When an MQGET fails due to no message being available (MQRC 2033), these buffers will have been allocated and released without being used, and the getter can be termed an “idle getter”.

The MQGET-with-wait model is slightly different and can be represented as in diagram 1. Regarding the diagram:

- Point 1 shows the MQGET being requested, causing the input and output buffers to be allocated.
- Point 2 shows the MQGET returning “no message available” to the stub, the input and output buffers being released and the stub going into a wait.
- Point 3 shows the stub being posted due to a message arriving and driving AMS to reallocate the input and output buffers
- Point 4 shows the successfully gotten message being decrypted and 2 AMS buffers being released.

Diagram 1: Requesting an AMS protected message using MQGET-with-wait



For each unsuccessful get, points 2 and 3 will occur – i.e. release and re-allocation of the buffers.

On our system an unsuccessful get from an unprotected queue took 4 microseconds whereas the same get on a protected queue using a 4KB buffer took 22 microseconds. As the use the right sized MQGET buffer section shows, when the buffer length specified is larger, the unsuccessful get time will increase.

In a shared queue environment where there are multiple tasks in get-waits, possibly spread across multiple queue managers in the QSG, all the tasks will be posted to attempt to get the message. Since only 1 task can successfully get the message and the other tasks will fail to get the message, these failed gets will cost result in more CPU usage in the associated AMS address space due to allocating and releasing storage.

Tuning for AMS: Trace

The use of the environment variable "_AMS_MSG_LEVEL" determines how much trace data is logged to the AMS task.

The default level is informational, which will show informational, warning, error and severe messages.

Increasing the level to debug or verbose is not recommended and should only be used under the guidance of IBM Service, as this can result in significant amounts of spool output and can adversely affect the performance of the AMS address space.

Tuning for AMS: Ensure there is sufficient CPU

It should be common practice to review RMF's CPU report to check that the CPU's allocated to the system are not running at 100% busy and that there is sufficient number of CPU's available to support the workloads being run.

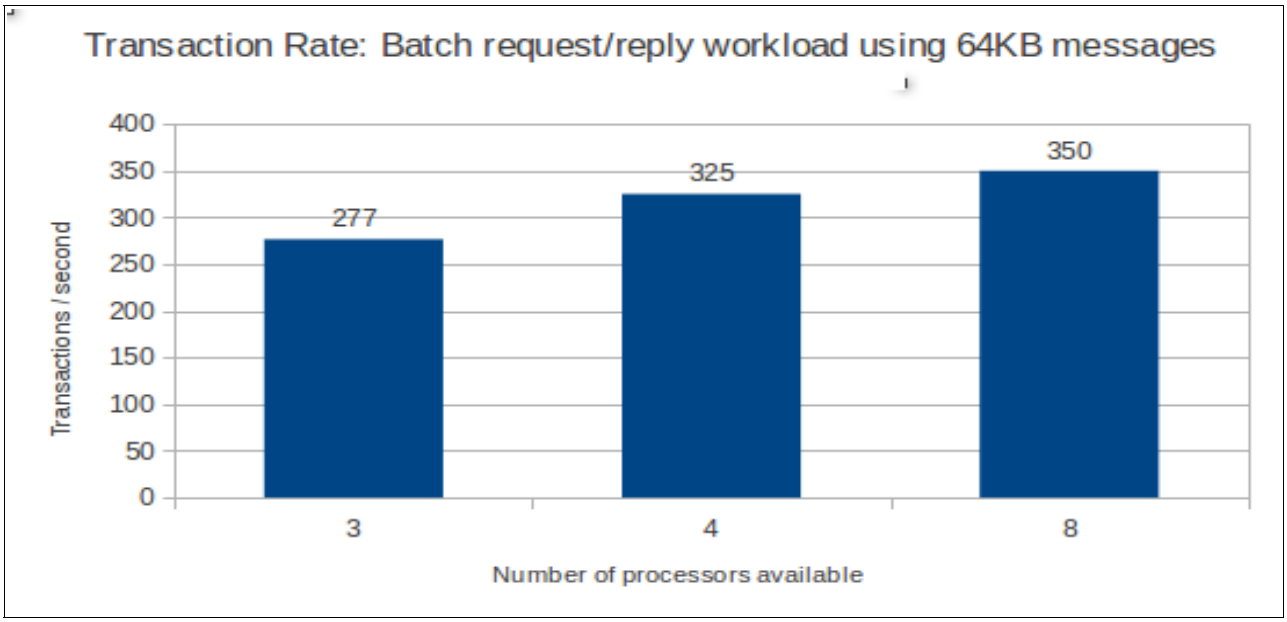
There are at least 2 considerations to take into account:

1. The RMF report shows the CPU's are 100% busy for the interval and more CPU capacity would benefit the system.
2. The RMF report shows the CPU's are less than 100% busy for the interval but due to an irregular workload there are bursts of activity which may benefit from more processors being available concurrently.

Adding AMS into a WebSphere MQ workload will increase the CPU load on your system and as such, checking for sufficient CPU capacity is important to ensure minimal impact on existing workload.

Even in an environment that is running with capacity available, consider chart 20, which uses the test configuration 1 as described in Appendix A. This test runs a request/reply workload with 5 requester tasks and 4 server tasks where the messages are 64KB. This means that there only 5 messages on the system at any time.

Chart 20: Varying number of processors for request/reply workload with 64KB messages



In Chart 20, when there were 3 processors available, the CPU was running at 94% busy but for 80% of the SMF interval, the system would have benefited from additional processors.

With 4 processors, the CPU was 83% busy but only had enough processors 60% of the time.

By increasing the number of processors to 8, the CPU usage dropped to 45% busy, there was always enough processors and the throughput increased by 26% over the 3 processor measurement.

Tuning for AMS: Use cryptographic processors where possible

When cryptographic hardware is not available, the function will be performed in the software layer. This can have a significant effect on the cost of processing a message, which can make a marked difference to the cost of processing smaller messages. We saw between 5 to 7 milliseconds per MQPUT and MQGET, regardless of message size.

The following two charts compare the transaction cost for a non-persistent request/reply workload where there is only 1 requester and 1 server task. In each case the messages are signed with MD5 and encrypted using RC2.

In this model, a transaction involves the requester putting a request message and getting a reply message, plus the server getting the request message and putting a reply message, i.e. 2 MQPUTs plus 2 MQGETs.

Chart 21: Transaction cost comparing availability of cryptographic hardware with 2KB messages.

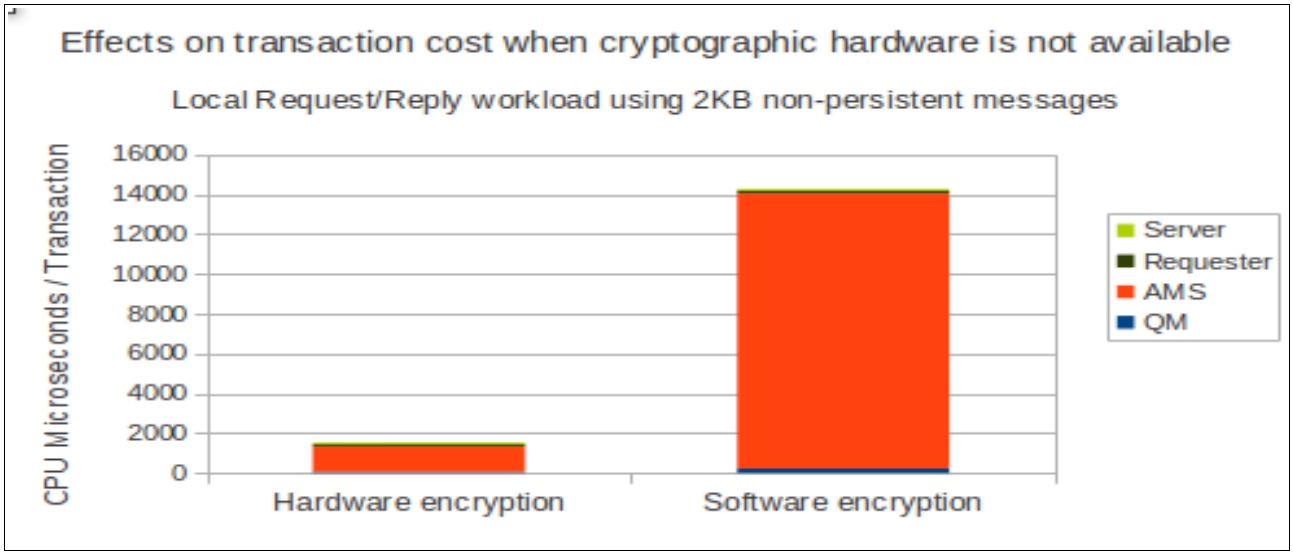
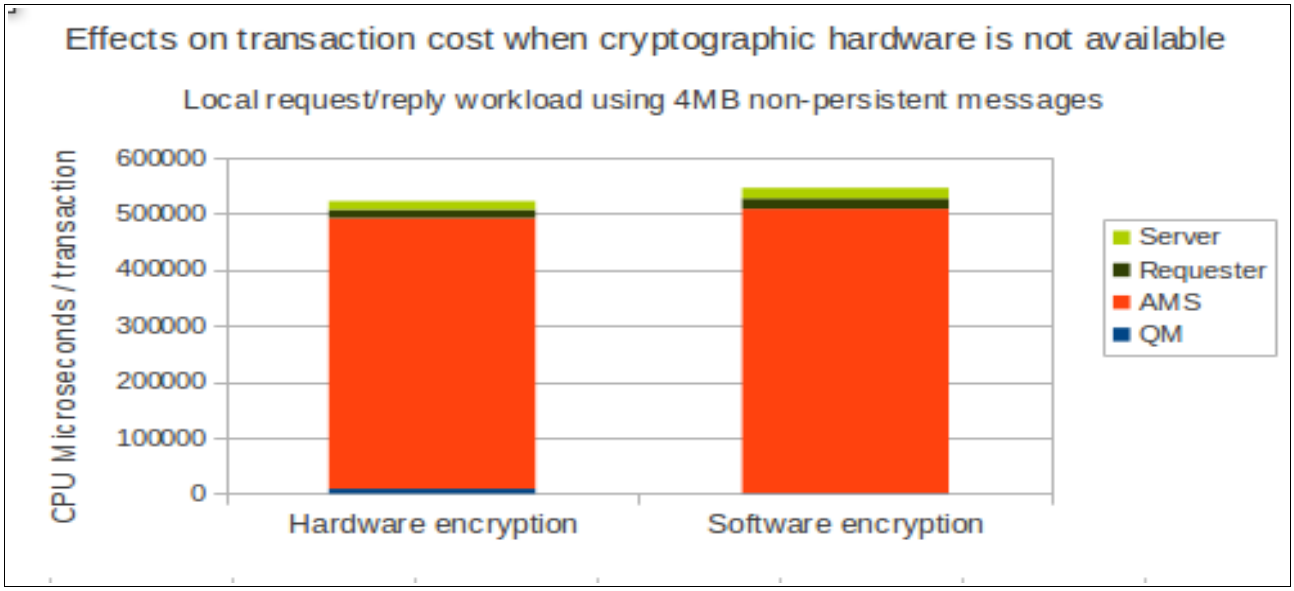


Chart 22: Transaction cost comparing availability of cryptographic hardware with 4MB messages.



Testing methodology used:
To simulate the lack of Crypto Express4 cryptographic hardware, the Integrated Cryptographic Service Facility (ICSF) address space was stopped.

This prevented the use of both the Crypto Express4 accelerator and co-processors that were typically available as well as preventing usage of the Central Processor Assist for Cryptographic Function (CPACF) which is available on every processor unit defined as a central processor.

Does put to waiting getter work with AMS?

Under certain conditions, WebSphere MQ is able to take a message directly from a putting tasks buffer and copy it to the getter tasks buffer. This can make a noticeable difference to the cost of an MQPUT and an MQGET as the message does not need to be stored in the queue managers buffers. This process is known as “put to waiting getter”.

For AMS protected queues, this option is not available.

What happens when audit records are written?

AMS is able to write audit records to SMF by setting the "_AMS_SMF_AUDIT" and "_AMS_SMF_TYPE" environment variables.

Once the SMF records are extracted, either from the SMF datasets or logstreams, they can be formatted using CSQ0USMF e.g.

```
//RUN  EXEC PGM=CSQ0USMF,REGION=0M,PARM='-SMFTYPE 180'
//STEPLIB  DD DISP=SHR,DSN=MQM.V800.INCCOM.AUTHLOAD
//SMFIN    DD DISP=SHR,DSN=&&SMFOUT
//SYSPRINT DD SYSOUT=*
```

When "_AMS_SMF_AUDIT=all" is set, there is 1 record written for each message e.g.

```
WebSphere MQ Advanced Message Security for z/OS Version 8 Release 0 Modification 0
SMF Audit Report for record type 180, QMGR=any
Generated on Thu Jul 10 11:40:51 2014

.. System=MVAA QMgr=VTS1 Policy=LIXC01 QoP=PRIVACY SigAlg=SHA1 EncAlg=3DES
   Jobname=MQCICS01 Userid=PERFTASK Operation=PUT Decision=GRANTED Audit=0 Reason=0
   Recipients='CN=PERFTASK,O=IBM,L=HURSLEY,C=UK'
   MsgId=C3E2D840E5 E3E2F14040 4040404040 40CCB5863F 6375DD44

.. System=MVAA QMgr=VTS1 Policy=LIXC01 QoP=PRIVACY SigAlg=SHA1 EncAlg=3DES
   Jobname=MQCICS01 Userid=PERFTASK Operation=GET Decision=GRANTED Audit=0 Reason=0
   Sender='CN=PERFTASK,O=IBM,L=HURSLEY,C=UK'
   Recipients='CN=PERFTASK,O=IBM,L=HURSLEY,C=UK'
   MsgId=C3E2D840E5 E3E2F14040 4040404040 40CCB5863F 6375DD44
```

In the above example a single message was put and then gotten successfully from queue LIXC01 by user "PERFTASK".

Further to each AMS audit record being written to SMF, there are approximately 3 SMF type 80 (RACF) "ACCESS" records written.

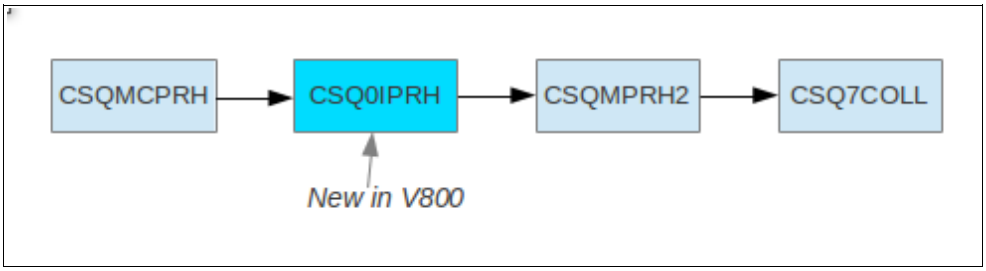
In a high volume messaging environment where SMF records are being written with high frequency, there is a possibility of the records being generated faster than SMF can write. AMS report this failure to write the SMF data in the application address space using a message similar to:

```
JOB39359 CSQ0503I @VTS1 CSQ0KMLG SMF record write failed, return code 00000028
```

If SMF records are being routinely failing to write, there may be benefit in using logstreams to store the SMF data and if the hardware supports, compressed logstreams.

Is there an impact on queue manager accounting and statistics data?

From a global trace we can see that when an MQ verb is called, the following modules are called:



Module CSQ0IPRH is where WebSphere MQ determines whether to go into AMS.

Module CSQ7COLL is where the statistics data collection will start / stop recording the time spent in the MQ API.

WebSphere MQ accounting and statistics data does not include the time spent in AMS.

Appendix A – Test configurations used with AMS

The tests run to determine the effect of AMS on a messaging workload are based on 5 basic configurations:

1. **Long running batch tasks using private queues.**
   Batch applications using non-persistent messages on private queues in a request/reply model with 2KB, 64KB and 4MB messages.

2. **Long running batch tasks using shared queues.**
   Batch applications using non-persistent messages on shared queues where there are 3 queue managers in the queue sharing group each on separate LPARs.
   Messages are 2KB or 4MB. Large messages stored on shared message data sets (SMDS).
   The servers tasks connected to each queue manager run in a MQGET-wait / MQPUT1 loop.
   Workload is non data-sharing which means that any server application regardless of the queue manager it is connected to, can process a particular request message. In the test environment, messages are processed by local servers approximately 66% of the time.

3. **Moving messages between queue managers over MCA channels**
   Moving 32KB messages between 2 z/OS queue managers with an increasing number of channels.

4. **CICS workload**
   Using a single CICS region, run short-lived transactions that perform MQPUTs and MQGETs before initiating the next transaction in a chain of transactions. Workload is increased by running multiple chains of transactions.

5. **Triggered CICS transactions**
   Messages arrive via MCA channels onto queues with either trigger every or trigger first enabled. The trigger message causes a CICS transaction to start that will MQGET of the message and will generate a reply message to be sent over an MCA channel to the originating application.

Appendix B – System configuration

MVS: zEnterprise EC12 (2827-7A1) configured thus:

LPAR 1

Between 1 and 16 dedicated CP processors

LPAR 2

Between 1 and 3 dedicated CP processors

LPAR 3

Between 1 and 10 dedicated CP processors.

Default configuration:

3 dedicated processors on each LPAR

Coupling Facility

Internal coupling facility with 4 dedicated processors.
Dynamic Dispatching switched OFF

DASD
FICON-connected DS8870
4 dedicated channel paths (shared across sysplex)
HYPERPAV enabled.

System Settings:

Each MVS image running z/OS v2r1
Coupling facility running CFCC level 19
ZHPF available, but by default is configured as disabled.
HIPERDISPATCH enabled by default.
Tests moving messages between LPARs that were on different TCP/IP subnets using a 10Gb network.

Cryptographic Hardware
Each LPAR has access to a shared Crypto Express4 Accelerator processor
Each LPAR has access to 2 Crypto Express4 CCA Co-processors

Trace Status:

- TRACE(GLOBAL) disabled.
- TRACE(S) enabled
- TRACE(A) CLASS(3) enabled

General Information

- Other IBM products used:
  - CICS V6.7 CTS4.2 with latest service applied as of May 2014.
  - WebSphere MQ v7.1.0 with latest service applied as of May 2014.