IBM MQ for z/OS on z17 Performance

Version 1.0 – December 2025

Tony Sharkey
Johnny Murphy

IBM MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire

Notices

DISCLAIMERS
The performance data contained in this report was 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 upon 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 *IBM MQ for z/OS 9.4 running on IBM z17.* The information is not intended as the specification of any programming interface that is provided by IBM MQ. It is assumed that the reader is familiar with the concepts and operation of IBM MQ for z/OS.

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 of their respective companies in the United States, other countries or both:

- **IBM Corporation:** IBM
- **Intel Corporation:** Intel, Xeon
- **Red Hat:** Red Hat, Red Hat Enterprise Linux

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.

Preface

In this paper, we will be looking at the improvements to our performance tests on MQ for z/OS as we moved from IBM z16 to IBM z17.

This paper is split into several parts:

Part one - Setting expectations of the hardware move.
Part two - What's new or changed on IBM z17.
Part three - Improvements from Network Express.
Part four - Replacing Storage Class Memory in the Coupling Facility.
Part five - General MQ test performance and scalability.

Part one describes what may impact the expectations of moving workload from IBM z16 to z17, and why it is not always straightforward.

Part two discusses the differences between IBM z16 and z17, particularly the CPU changes and CFCC 26, and how those changes may affect your own systems.

Part three looks at the performance benefits to components of MQ that utilize the benefits from Network Express cards rather than the more traditional OSA cards.

Part four looks at the impact of the deprecation of Storage Class Memory on IBM z17 Coupling Facilities and the alternatives when using MQ with deep shared queues.

Part five presents the results of measurements run first on z16 and then subsequently on z17. We also include scalability measurements to demonstrate how MQ performs when the number of processors is increased up to 32 on a single z/OS LPAR.

Table of Contents

1 Setting expectations of the hardware move

The IBM® z17™ (z17) offers many improvements over IBM z16 but of note are an increase in the number of processors available running at 5.5 GHz, increased L2 cache size, virtual L3 cache up to 360MB per CP and Virtual L4 cache up to 2.88GB per drawer.

Additionally, the z17 introduces new features including the following:

- Network Express, new hardware for OSA, RoCE and Coupling LR, supporting up to 25 Gb.
- zHyperLink 2.0

When trying to set the expectations of the benefits of moving to the z17 there are several good starting points:

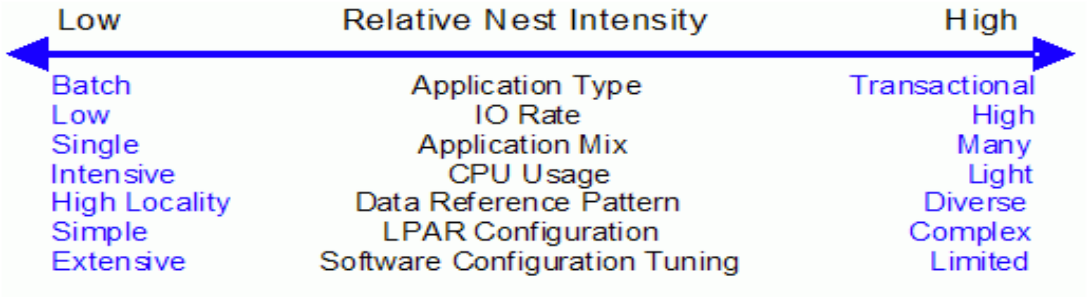
- Redbook "IBM z17 (9175) Technical Guide", in particular chapter 12 "Performance".
- Website "Large System Performance Reference (LSPR) for IBM Z".

It is important to recognize that MQ is generally a small part of your system solution and the IBM z17 technical guide offers a detailed strategy for capacity planning on your entire system.

It should also be noted that when using the LSPR data to predict how a workload might perform on the z17, the type of workload makes a difference.

The most performance sensitive area of the memory hierarchy is the activity to the memory nest, namely the distribution of activity to the shared caches and memory.

Many factors influence the performance of a workload; however, the Relative Nest Intensity (RNI) is typically the largest influencer.



Despite containing little business logic, the MQ performance workloads vary significantly in complexity and cover the entire range of Low, Average and High RNI.

Additionally, the number of processors allocated can affect the expectations – for example we have workloads that are classified as low RNI when running on 3 CPUs but average when running on 32 CPUs.

The following table shows the expected improvement on z17 over z16 for the varying workloads on the typical CPU configurations used in our performance tests:

CPUs	LOW	AVERAGE	HIGH
3	+9%	+10%	+12%
16	+9%	+11%	+13%
32	+10%	+12%	+13%

What this table suggests is that depending on the workload type and the number of CPUs allocated, we may see between 9 and 13% improvement over comparable measurements on z16.

We recognize that not everyone will migrate from z16 to z17, and indeed a more common migration path might be z15 to z17 and therefore offer the following table to indicate the expected improvements on z17 over z15 for the varying workloads on the same 3 CPU configurations detailed above:

CPUs	LOW	AVERAGE	HIGH
3	+18%	+22%	+26%
16	+20%	+24%	+28%
32	+21%	+25%	+29%

This table suggests that going directly from z15 to z17 might result in between 18 to 29% improvement in comparable measurements run on z17.

LSPR performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Tempering expectations

As discussed earlier, the expectation is that CPU intensive workloads would see 9-13% improvements over equivalent workloads on z16. This may be of the form of generally reduced CPU costs or improved response times.

Achieving this expectation should be tempered by performance of DASD, network, and cryptographic response times, which may not see the same improvement as well as many other factors.

DASD – *for example Log I/O* may benefit from the additional bandwidth offered by FICON 32 – but only if the system is bandwidth constrained. Conversely this could have a detrimental impact as more data reaching the DASD may result in the saturation of non-volatile storage (NVS, effectively cache) and being reported as Disk Fast Write Bypass (DFWBP) delays.

Our current DASD does not support zHyperLink 2.0, therefore we do not provide any performance data pertaining to that feature.

Network - In our test configuration, we are still reliant on the same 10Gb network links between z/OS LPARs and distributed partner machines. We have seen reduced latency when using Network Express in place of OSA adapters, which in our already low-latency networks can show significant increase in transaction throughput. There is also reduced transaction cost on the z/OS LPAR because of the improved processor performance.
At the time of publication, we have not configured SMC-D or SMC-R on the Network Express adapter.

At IBM z17 launch, the latest **Crypto Express** feature available remains the same as on IBM z16, but it is worth remembering that encryption and decryption of data is typically performed either by CP Assist for Cryptographic Functions (CPACF) and this nature of cryptographic work would be expected to see performance benefits akin to those suggested by LSPR comparisons, i.e. of the order 9 to 13%.

2 What's new or changed on IBM z17?

The Redbook "IBM z17 (9175) Technical Guide", provides a detailed specification of the IBM z17, but what follows offers a comparison of the changes from the IBM z16.

Machine	**IBM z16 Model A01 Max200**	**IBM z17 Model ME1 Max208**
Processor speed	5 GHz	5.5 GHz
Drawers	4	4
Processor units (PU) per drawer	4 *Dual Chip modules (DCM)*	*4 Dual Chip modules (DCM)*
Processors (cores) per PU	8	8 Plus DPU
Active cores per PU	9-11 or 10-15 for each DCM **Our SYSPLEX:** 14 for each PU, 7 per chip	9-11 or 10-15 for each DCM **Our SYSPLEX:** 14 for each PU, 7 per chip
Cores per drawer	64	64 plus 8 DPUs
Cache		
L1	256KB	256KB
L2	32MB per PU *(Effectively 4MB per core)*	36MB per PU
L3	256MB	360MB
L4	2GB	2.88GB

IBM z17 introduces the **DPU**, Data Processing Unit, which brings the network closer to the processors, and includes 32 small processor cores.

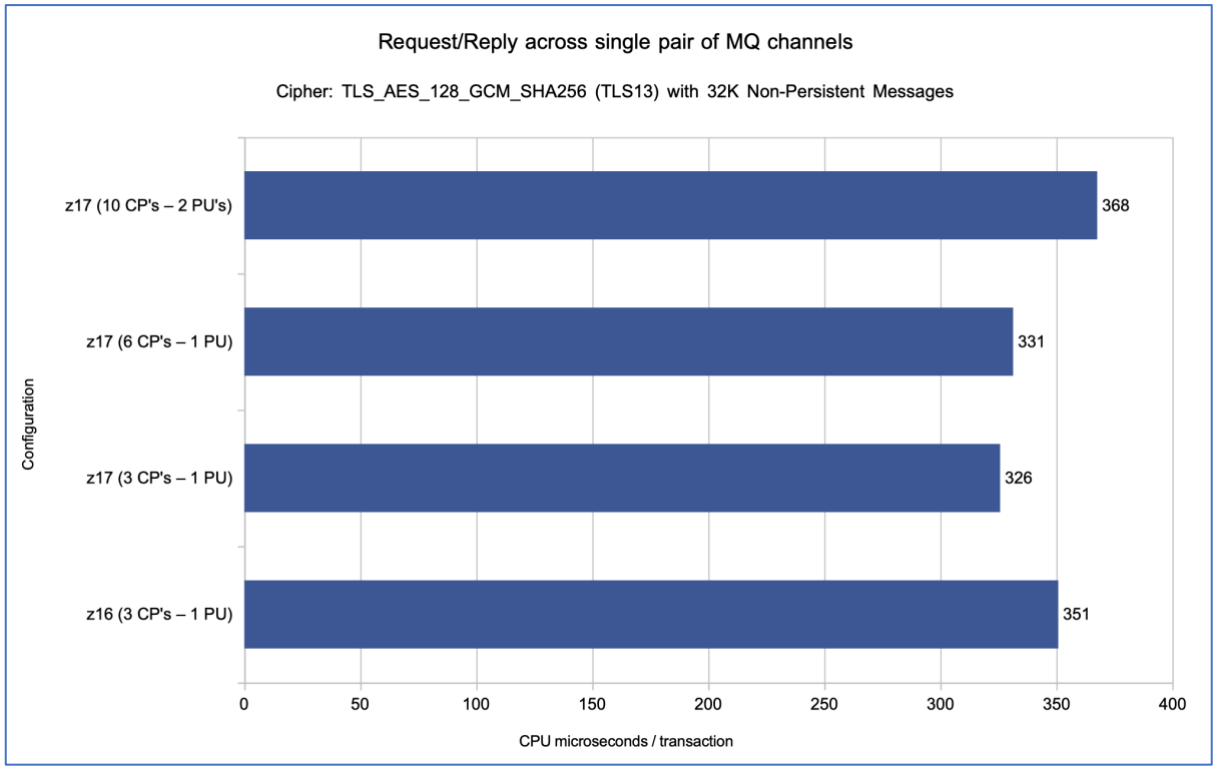
Processors

In our micro-benchmarks we have always seen an impact to transaction cost when running a workload that spans into a second and subsequent processor unit (PU), and with the reduction in cores on a PU on IBM z16 and z17, this impact can come earlier and more frequently.

On IBM z15 there were 12 cores per PU – on our systems, we were seeing up to 10 processors being allocated per PU.

On both IBM z16 and IBM z17 there are 8 cores per PU – and on our systems we are seeing a maximum of 7 processors being allocated per PU.

To demonstrate the impact of spilling over onto multiple PUs, the following chart compares the transaction cost when running a request/reply-type workload using 32KB non-persistent messages across a single pair of MQ channels protected with TLS 1.3 cipher `TLS_AES_128_GCM_SHA256`.



In the original measurements on both IBM z16 and z17, the test was configured with 3 CPUs on each LPAR. The immediate impact of the migration to z17 was a reduction in the transaction cost of 7.1%.

Configuring additional CPUs on the IBM z17, whilst remaining on a single PU, saw a small increase in transaction cost, equivalent of a 1.7% increase.

Configuring the workload such that the CPUs were spread across multiple PU's, resulted in the transaction cost increasing by 12.8% over the IBM z17 3 CPU configuration.

In these measurements, the workload did not require 10 CPUs per LPAR – and as a result, just 3 CPUs per LPAR were sufficient. Indeed, too many CPUs being allocated, such that the transaction cost increased, resulted in the throughput decreasing by 10%.

z/OS is designed to run at high CPU utilisation and for best performance, it is key to run with as few CPUs as needed without causing bottlenecks with the number of tasks waiting for CPU. The RMF CPU report can be used to indicate whether there are sufficient CPUs available and whether there is work waiting for CPUs.

In our environment, we configured micro-benchmarks with up to 32 CPUs per LPAR and compared against the equivalent IBM z16 configuration and largely saw improved performance on IBM z17.

When moving from IBM z15 to z17, the maximum number of CPUs per PU may differ and as a result LPARs with many CPUs that run workloads with larger path lengths, potentially due to far more complex processing than the MQ micro-benchmarks, may see more varied impact then in our measurements. The additional cache on IBM z17 may offer a net benefit to transaction cost.

Network Express

The Redbook ""IBM z17 (9175) Technical Guide" describes Network Express as:

"The Network Express adapter is the common IBM z17 hardware for OSA, RoCE and Coupling Express3 Long Reach (CE3-LR). It converges the legacy OSA-Express, RoCE Express and CE-LR into one hardware platform offering.

The new adapter supports all legacy functions available with OSA-OSD but uses EQDIO (Enhanced-Queued Direct I/O) architecture while OSD uses QDIO. The Network Express card only supports the EQDIO architectures and uses the OSA Hybrid (OSH) channel-path identifier for OSA-style I/O.

Network Express supports both PCIe (e.g. RoCE) and OSA functions on the same card port, which eliminates the need for some RoCE users to purchase a dedicated OSA card solely to enable RoCE I/O. Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) is a network protocol that enables direct memory transfers between two computers within the same Ethernet broadcast domain, which reduces latency and CPU load while increasing bandwidth.
These features reduce CPU usage for applications that rely on the TCP/IP stack. They also help reduce network latency with memory-to-memory transfers by using Shared Memory Communications over RDMA (SMC-R).

With SMC-R, you can transfer huge amounts of data quickly and at low latency. SMC-R is transparent to the application and requires no code changes."

On our systems, we have configured both OSA and Network Express adapters, which allows us to compare the impact of z16 OSA to both z17 OSA and z17 Network Express for our MQ performance benchmarks.

Coupling Facility - CFCC Level 26

CFCC level 26 is delivered on the IBM z17 (9175) and most notably for MQ removes the "Storage Class Memory in a CF" partition feature. Additionally, when moving from IBM z15 to IBM z17, it is worth mentioning the deprecation of DYNDISP for shared engine CF images as part of CFCC 25.

Deprecation of Storage Class Memory (SCM) in a CF partition*:*

Beginning with IBM z17, a coupling facility (CF) partition can no longer use Storage Class Memory.
Note that SCM is sometimes referred to as Virtual Flash Memory (VFM).

This applies to Coupling Facilities running on IBM z17 and does not affect configurations running z/OS on z17 and an external CF on a prior generation.

For a Coupling Facility running on z17, it is advised that the CFRM policy is updated to remove references to the following Storage Class Memory attributes:

- SCMMAXSIZE()
- SCMALGORITHM(KEYPRIORITY1)

Should these attributes remain, on z/OS 3.1 at least, there may be some reduced capacity in your defined structure(s).

Instead of relying on SCM, you can use larger Coupling Facility structures in your CFRM policy or offload to either IBM MQ for z/OS' Shared Message Datasets (SMDS) or Db2.

Deprecation of DYNDISP=ON|OFF for shared engine CF images as part of CFCC 25:

DYNDISP

Coupling Facility images can run either shared or dedicated processors. Dedicated processors are recommended for best performance and production use (continuous polling model). Shared processors are recommended for test/development use where a CF image would require less than one processor's worth of capacity or for less-performance critical production usage.

Prior to IBM z16, in shared-processor mode, the CF could use several different Dynamic Dispatching (DYNDISP) models.

- DYNDISP=OFF – LPAR time-slicing completely controls the CF processor; the processor polls the entire time it is dispatched to a CF image by PR/SM. The CF image never voluntarily gives up control of the shared processor. This option provided the least efficient sharing, and worse shared-engine CF performance
- DYNDISP=ON – an optimization over pure LPAR time-slicing; the CF image sets timer interrupts to give PR/SM initiative to re-dispatch it, and the CF image voluntarily gives up control of the shared processor. This option provided more efficient sharing and better shared-engine CF performance.
- DYNDISP=THIN support to use "Thin Interrupts" has been available since zEC12/zBC12 and has been the default mode of operation for shared-engine CF images since IBM z15.

From IBM z16, DYNDISP=THIN is the only available behaviour for shared-engine CF dispatching.

Further detail on DYNDISP=THIN performance is available here.

CFCC level 26 and MQ for z/OS structures

The z/OSMF Sysplex Management service CFRM policy editor should be used for accurate sizing of your MQ Coupling Facility structures.

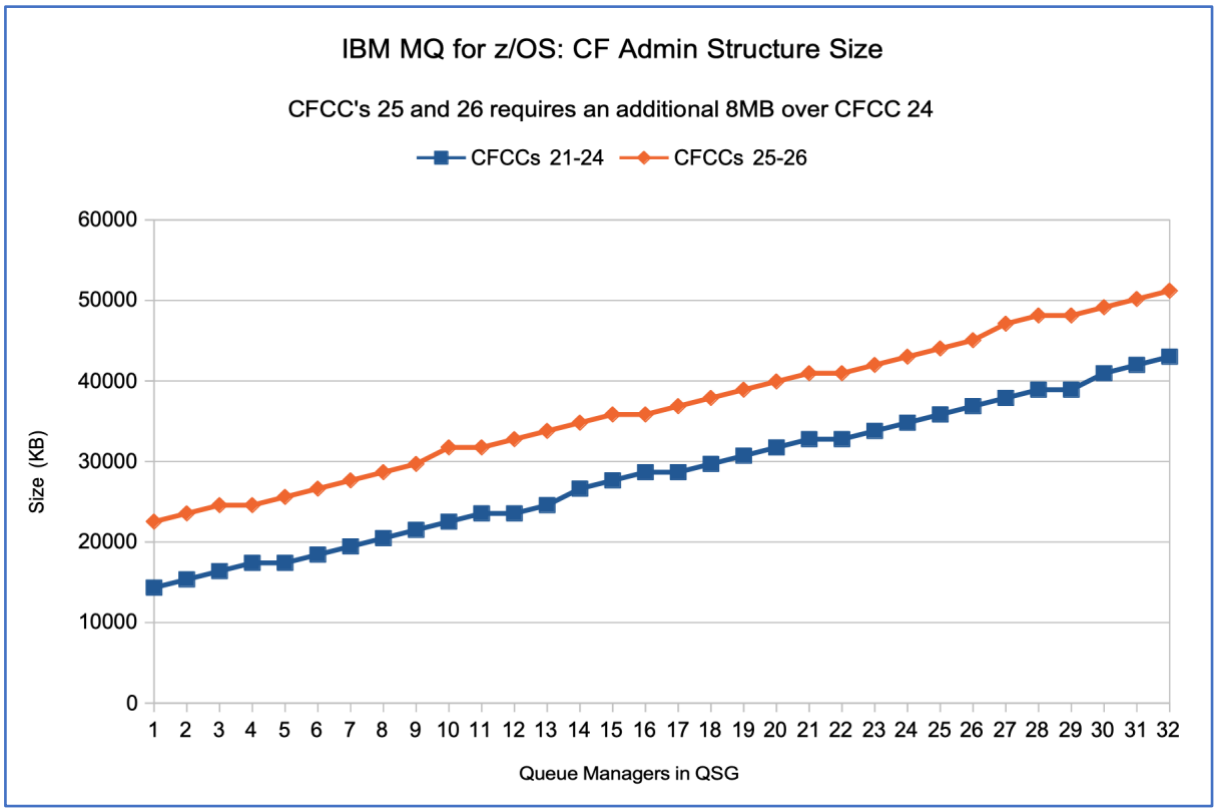
The following subsection offers an indication of the increase storage requirements for MQ CF structures that we observed moving to IBM z17.

Does CFCC level 26 affect the size of my ADMIN structure?

CFCC levels 25 and 26 required an additional 8MB to be allocated to the CSQ_ADMIN structure.

Typically, the size of the admin structure depends on the number of queue managers in your Queue Sharing Group (QSG). When migrating to CFCC level 26 from CFCC 24 or earlier, there is an additional 8MB overhead of CF storage that must be provided.

The following chart shows the required size of the admin structure by the number of queue managers in the QSG.

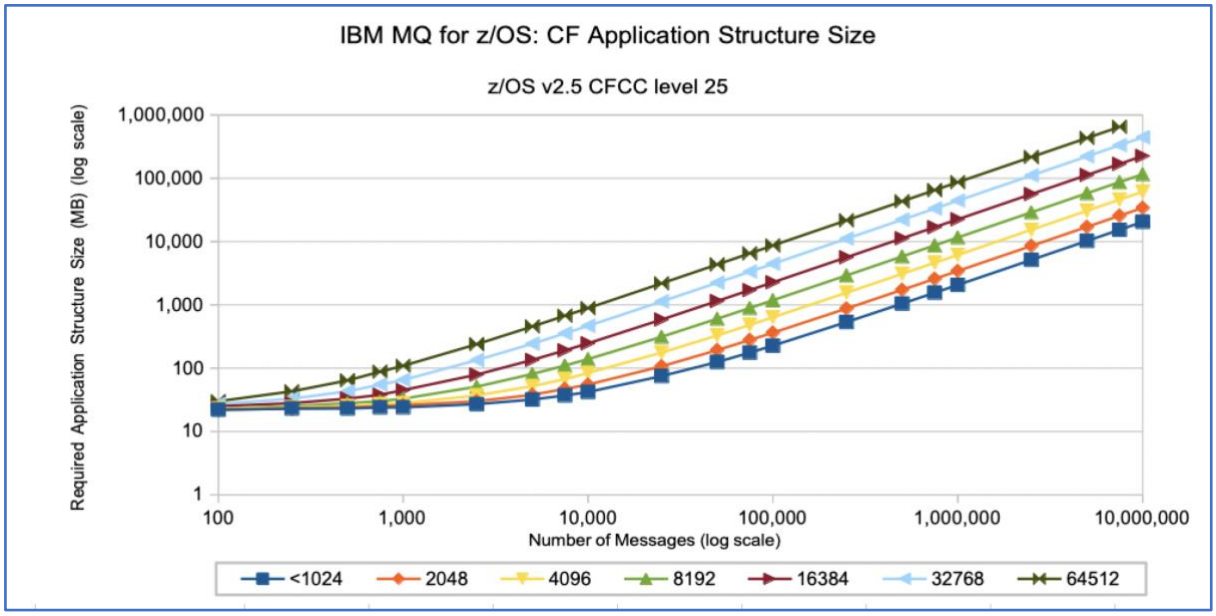


Does CFCC level 26 affect the size of my APPLICATION structures?

Yes, to a small extent. Whilst it is possible to allocate the application structures with the same sizes as with earlier CFCC levels, the number of messages able to be stored in the structure would be slightly reduced.

As with the additional storage required by the admin structure, each application structure should be increased by 8MB to ensure the structures are able to store the same number of messages as earlier CFCC levels.

The following chart offers an indication of the number of messages that can be stored for a particular CF application structure size. Note that the "message size" includes the user data and all MQ headers except for the MQMD.



The following table gives *approximate* message capacity of an IBM MQ CF application structure sized at 4GB, assuming the structure is defined in the CFRM policy with `ALLOWAUTOALT(NO)` and no messages are being offloaded due to MQ's CFLEVEL(5) offload thresholds.

Message Size (Excluding only MQMD)	**CF at CFCC level 17-26** Approximate number of messages in 4GB structure
All message sizes <= 1164	1,900,000
2,048	1,150,000
4,096	650,000
8,192	340,000
16,384	175,000
32,768	90,000
64,512	45,000

Does CFCC level 25/26 affect the size of my SYSAPPL structure?

Moving from CFCC 24 or earlier to CFCC 25 or 26 will require an increase in the size of the SYSAPPL structure.

When starting an MQ queue manager with the SYSAPPL structure configured at 20 MB, the queue manager reported:

```
14.55.43 STC48092 IXL015I STRUCTURE ALLOCATION INFORMATION FOR 213
213               STRUCTURE PERFCSQSYSAPPL, CONNECTOR NAME CSQEPERFVKW303,
213               CONNECTIVITY=DEFAULT
213               CFNAME ALLOCATION STATUS/FAILURE REASON
213               -------- ----------------------------------------
213               AACF01   INVALID STRUCTURE SIZE:             20 M
213                        INITSIZE MUST BE AT LEAST:           41 M
```

Sample JCL SCSQPROC(CSQ4CFRM) in MQ for z/OS 9.4 suggests an INITSIZE of 50,000.

3 Improvements from Network Express

Testing Environment

On IBM z16, the MQ for z/OS performance measurements over network used variations of TCP/IP on OSA, SMC-R and SMC-D.

To date, our IBM z17 environment has not been configured with SMC-R or SMC-D but there are configurations to enable use of TCP/IP on both OSA and OSH (Network Express) plus HiperSockets.

The data following in this section compares the performance of IBM z16 (OSA) with:

- IBM z17 OSA
- IBM z17 Network Express
- Additionally, some measurements are described with HiperSockets configured.

It should be emphasised that these workloads are micro-benchmarks, concentrating on applications with minimal non-MQ processing, such that MQ and network costs are the primary contributors to the overall transaction cost.

The workloads are run between 2 z/OS LPARs with very short-latency high-bandwidth networks.

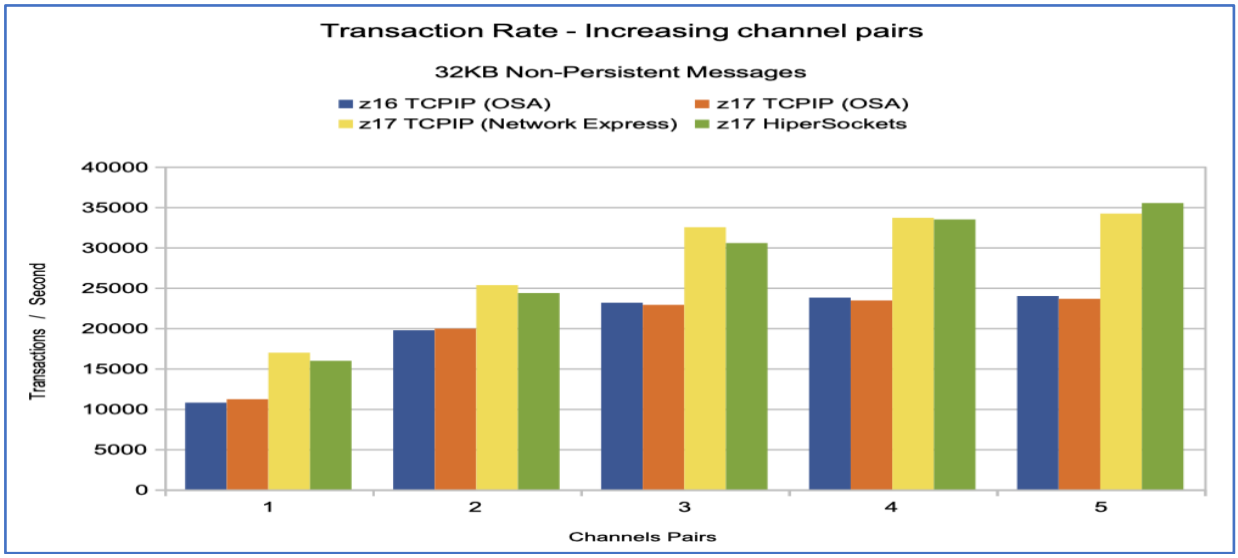
For MQ channels protected with TLS ciphers, the improvement to throughput was not as significant as for channels that were not protected. This is in part due to both sets of measurements using the same level of Cryptographic hardware, i.e. CryptoExpress 8S.

MQ Message Channel performance

When using Network Express adapters, our network-dependent MQ performance tests using MQ message channels have seen throughput increase by up to 60% compared to workloads using OSA.

For example, a request/reply workload using 32KB non-persistent messages between 2 z/OS LPARs where the data is flowed via an external switch, showed significant improvements in performance when using Network Express over using OSA on both z16 and z17.

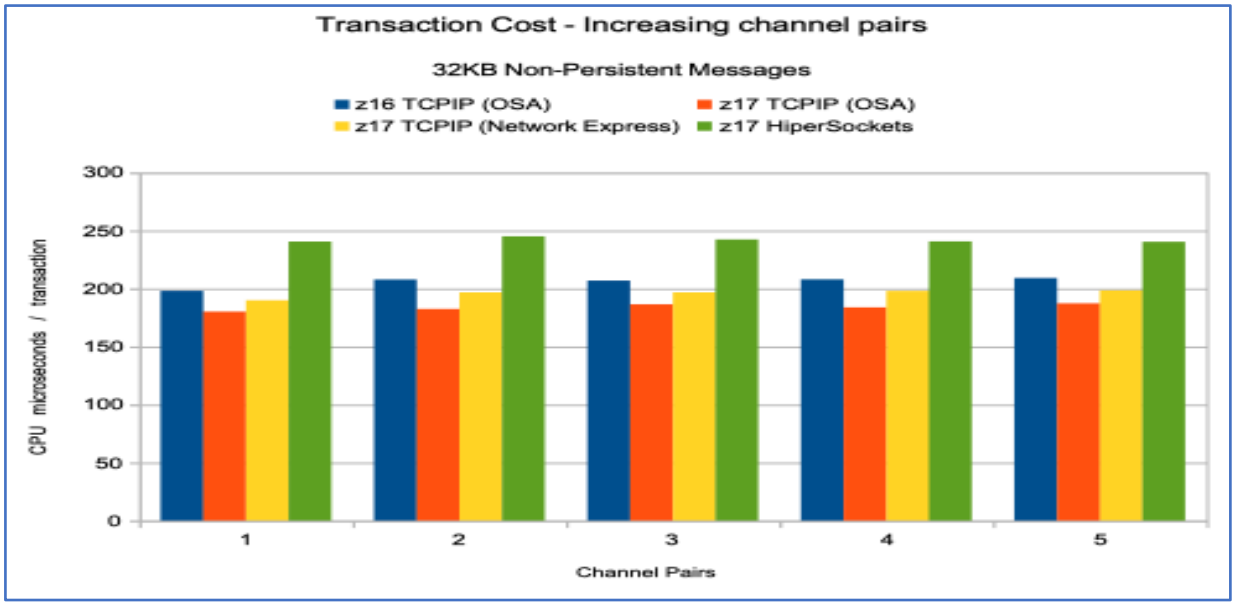
Transaction Rate:



Notes on chart:

- Transaction rates for OSA on both z16 and z17 are similar.
- Transaction rates for Network Express exceed OSA by 38-58%.
- Network Express shows similar performance to HiperSockets.

Transaction Cost:



Notes on chart:

- Transaction costs for OSA on z17 are 9-12% lower than on z16.
- Using Network Express increases the z17 transaction cost by 5-8% over the z17 OSA transaction cost but remains 5% lower than z16, i.e. there is a cost to running at a higher transaction rate.
- HiperSockets costs are approximately 30% higher the Network Express transaction costs despite achieving similar transaction rates. This additional cost is incurred in the TCP/IP address space.

Message Channel Start Rate when using Network Express

The rate and CPU cost at which channels can be started and stopped varies with the number of channels represented in SYSTEM.CHANNEL.SYNCQ.

A channel is represented in SYSTEM.CHANNEL.SYNCQ if it has ever been started. It will remain until its definition is deleted. For this reason, we recommend that redundant channel definitions be deleted.

Whilst many users do not start and stop channels with any great frequency, there may still be significant sender channel restart activity after a channel initiator failure.

The following table demonstrates the effect of the number of channels represented in the SYSTEM.CHANNEL.SYNCQ on an IBM MQ 9.4 queue manager.

Channels in .SYNCQ	**Sender Channel Start**		**Sender Channel Stop**	
	Channels per second	**9175-703 CPU milliseconds per START**	**Channels per second**	**9175-703 CPU milliseconds per STOP**
1000	840	0.20	781	0.20
2000	702	0.25	690	0.23
4000	549	0.35	637	0.30

Note: For this measurement, only the sender-side costs are reported.

Compared to IBM z16 (3931) with OSA, the channel start rate increased by up to 30% whilst channel stop rate was not significantly different.

Both channel start and stop costs were approximately 9% less on IBM z17 than IBM z16.

Message TLS-protected Channel Start rate

When determining the impact of IBM z17 to TLS-protected channels being started, there are additional considerations, including the key-size of the certificate used as well as the cipher used to protect the MQ channel.

The impact of the key-size is discussed in the blog: "MQ for z/OS: Impact of certificate key-size on TLS-protected MQ channels".

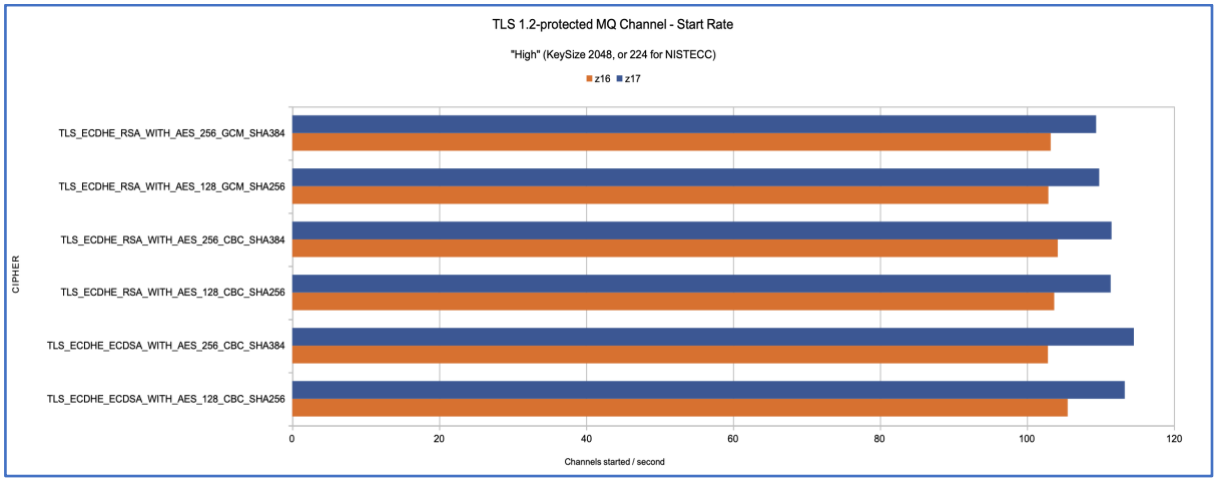
For this report, the measurements shown are when using the default key-size of 2048 bits for RSA and 192 bits for NISTECC ciphers. These key sizes correspond with the "high strength" in the above-mentioned blog.

TLS 1.2 ciphers

The following charts compare the performance of the TLS 1.2 ciphers supported by the z/OS queue manager.

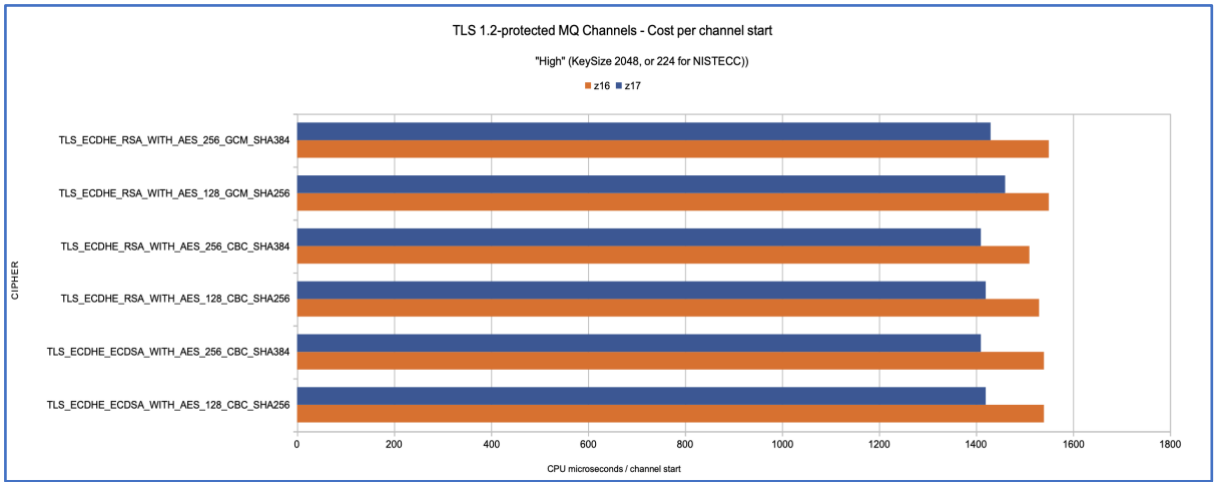
They show the rate at which our system was able to start the channels over a low-latency network – using OSA adapters on IBM z16 and Network Express adapters on IBM z17, which reduces the impact of network time on any negotiation.

TLS 1.2 Channel Start Rate



In terms of improvements to channel start rate, IBM z17 is achieving 6 to 10% more channels being started per second over the IBM z16 configuration, which uses the same generation CryptoExpress 8S adapter.

TLS 1.2 Channel Start Cost



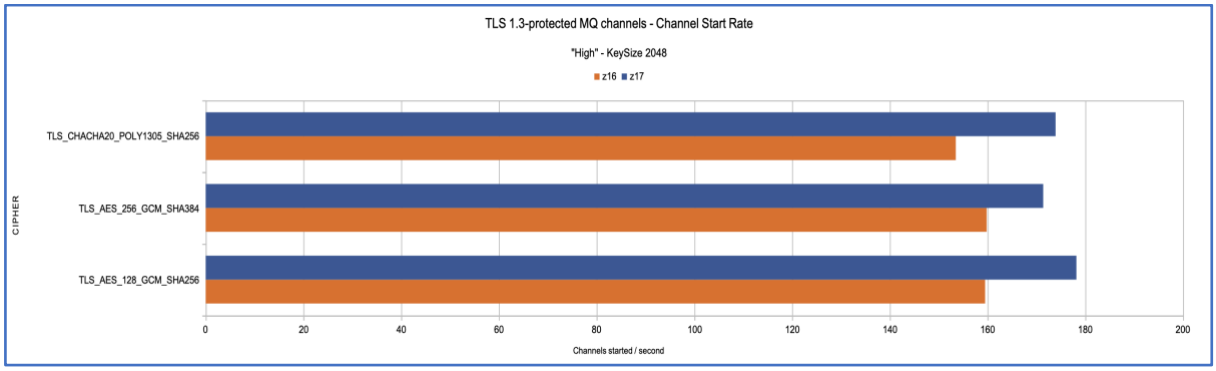
IBM z17 does demonstrate a reduced cost of channel start for TLS 1.2-protected channels ranging from 5.5 to 8.5% lower than the equivalent IBM z16 cost.

TLS 1.3 ciphers

The following charts compare the performance of the TLS 1.3 ciphers supported by the z/OS queue manager.

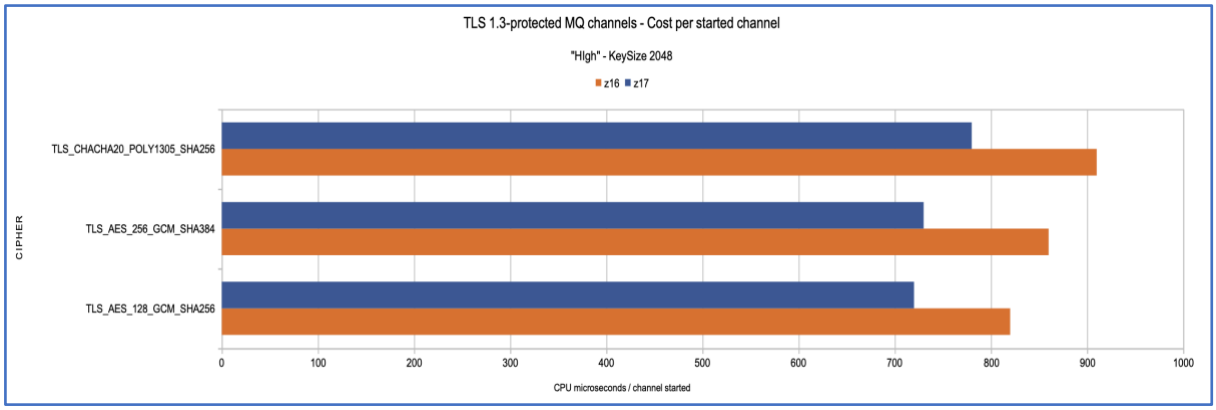
As with the TLS 1.2 ciphers, the charts show the rate at which our system was able to start the channels over a low-latency network – using OSA adapters on IBM z16 and Network Express adapters on IBM z17, which reduces the impact of network time on any negotiation.

TLS 1.3 Channel Start Rate



In terms of improvements to channel start rate, IBM z17 is achieving 7 to 13% more channels being started per second over the IBM z16 configuration, which uses the same generation CryptoExpress 8S adapter. Some of this improvement is due to processor improvements and some due to Network Express latency improvements.

TLS 1.3 Channel Start Cost



For TLS 1.3 protected channels, the IBM z17 costs are 12 to 15% lower than the equivalent measurement on IBM z16.

MQ SVRCONN Channel performance

MQ Clients putting and getting from MQ for z/OS queue managers

When using Network Express adapters, our network-dependent MQ performance client tests using SVRCONN channels have seen throughput increase by up to 50% compared to workloads using OSA.

For example:
A workload where there are 2 client applications, each running on a separate partner machine with one putting messages to a MQ for z/OS private queue and the other application getting those messages from the queue have improved the sustainable throughput rate by up to 47% on IBM z17 when using Network Express.
In this case, sustainable means that the get application can keep pace with the putting application such that the queue depth does not increase.

For this example, there are 4 configurations:

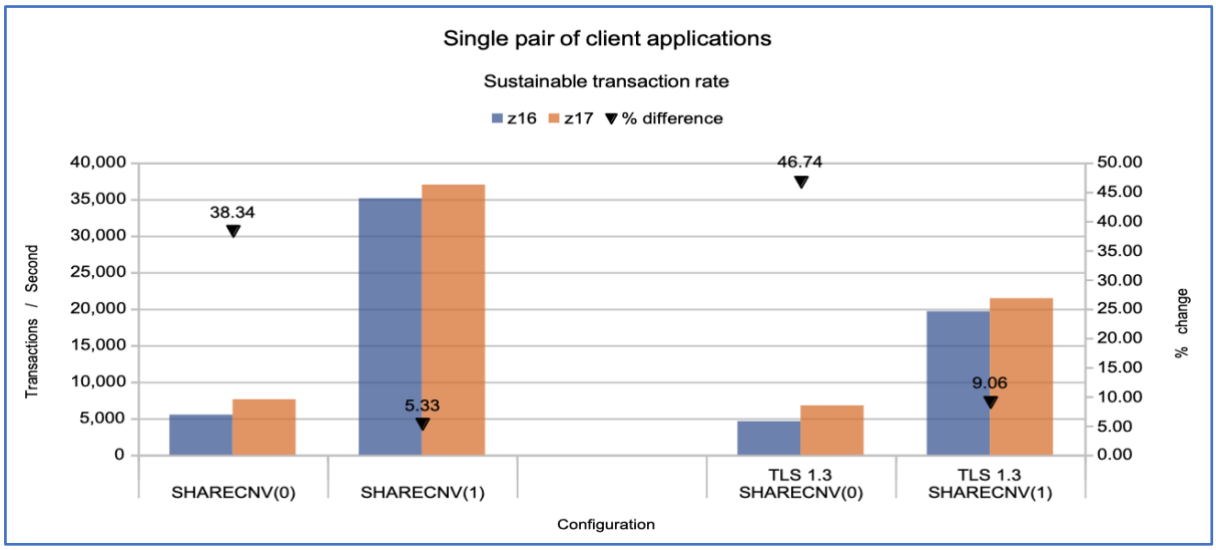
1. Clients connect using SVRCONN channels configured with SHARECNV(0). All puts and gets are synchronous.
2. Clients connect using SVRCONN channels configured with SHARECNV(1). MQ queue is configured with DEFPRESP(ASYNC) and DEFREADA(YES), for asynchronous puts and gets.
3. Clients connect using SVRCONN channels configured with SHARECNV(0) and protected with TLS 1.3 cipher TLS_AES_128_GCM_SHA256. All puts and gets are synchronous.
4. Clients connect using SVRCONN channels configured with SHARECNV(1) and protected with TLS 1.3 cipher TLS_AES_128_GCM_SHA256. MQ queue is configured with DEFPRESP(ASYNC) and DEFREADA(YES), for asynchronous puts and gets.

The configuration of these tests mean that the cost of the workload is determined from the CPU used in the MQ queue manager, channel initiator and the TCP/IP address spaces.

The putting application connects to the z/OS queue manager, opens a request queue, puts 1 million 2KB non-persistent messages, closes the request queue and disconnects.

The getting application connects to the z/OS queue manager, opens the queue, gets the next available message 1 million times, closes the queue and disconnects.

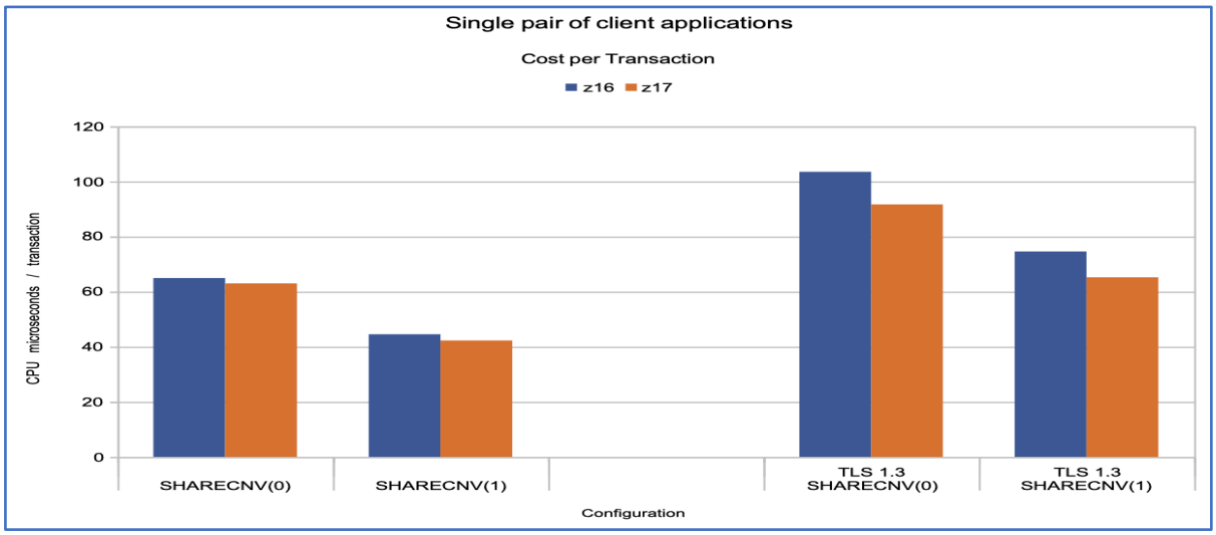
Transaction Rate:



For the tests using SHARECNV(0), there is a significant improvement in throughput regardless of whether the SVRCONN channel is protected by TLS 1.3 ciphers. For these configurations, throughput increased by up to 46.7% over z16.

For the tests using SHARECNV(1) and asynchronous puts and gets, there was a smaller improvement in transaction rate, ranging from 5 to 9%. This reduced benefit is because there are less flows between client and queue manager, therefore less impact from reduced network response times.

Transaction Cost:



With regards to the change in transaction cost when moving the workloads to z17, the non-TLS 1.3 protected channels see a smaller reduction in cost ranging from 3 to 5%.

For the TLS 1.3 protected SVRCONN channel, moving to IBM z17 saw a reduction in transaction cost of 12%, which is better than LSPR predicted.

Client Channel start rate

Many years ago, I wrote a blog "The cost of connecting to a z/OS queue manager", and part of that blog contained a table of costs for connecting and disconnecting from the z/OS queue manager.

With the IBM z17 and Network Express improvements, it seems appropriate to update that table and provide some comparison with IBM z16, since the data in the referenced blog was based on IBM z14 and z/OS 2.3!

To recap regarding MQ connections:
Connecting an MQ client to a queue manager is a relatively expensive procedure, which is why we typically recommend that your client applications remain connected for as long as is practical.

In order of cost (lowest to highest), it is more efficient to use the model 1, 2 and then 3:

1. CONNECT, OPEN, n*[PUT, GET], CLOSE, DISCONNECT
2. CONNECT, n*[OPEN, PUT, GET, CLOSE], DISCONNECT
3. n*[CONNECT, OPEN, PUT, GET, CLOSE, DISCONNECT]

The cost of the put or get is largely dependent upon the properties of the message including size, persistence, type of queue i.e. shared or local message.

The cost of the queue open and close may vary significantly if using a shared queue as a result of seeing first-open or last-close impacts as per MP16.

The original measurements and those run on IBM z16 used the TLS 1.2 cipher `TLS_RSA_AES_128_CBC_SHA256`, but the IBM z17 measurements also include TLS 1.3 cipher `TLS_AES_128_GCM_SHA256`.

Cost of connecting a client to z/OS queue manager on IBM z17:

SHARECNV	**0**	**0**	**1**	**1**
LISTENER	**Local**	**Shared**	**Local**	**Shared**
Basic	365	650	430	795
TLS 1.2 cipher	+350	+350	+350	+350
TLS 1.3 cipher	+470	+478	+472	+490
Suppressing X511 and X512	-110	-110	-110	-110

Costs shows are CPU microsecond per MQCONNX and MQDISC.

Using the table to calculate the cost of the connection:

- Using a local listener on a SVRCONN with SHARECNV(0) cost 365 microseconds.
    - Enabling a TLS 1.2 cipher added 350 microseconds for a total of 715 microseconds.
    - Enabling a TLS 1.3 cipher added 470 microseconds for a total of 835 microseconds.
    - Suppressing the X511 and X512 messages reduced the cost by 110 microseconds.

Comparing these costs to IBM z16, shows a reduction in cost of the basic MQCONNX and MQDISC of between 10 to 13% when run on IBM z17.

In addition to the reduced cost of MQCONNX and MQDISC, the time to connect and disconnect has reduced by between 8 to 12%.

Using the TLS 1.3 cipher on z17 sees a similar cost of connect and disconnect to that on z16 when using the TLS 1.2 cipher.

4 Replacing Storage Class Memory in the Coupling Facility

Storage Class Memory (SCM), also known as CF Flash, was introduced on the zEC12 as a mechanism to increase the total storage capacity for a CF structure without needing to define large amounts of real memory.

On z14, SCM was moved from Flash Express (SSD on PCIe cards) to Virtual Flash Memory (VFM), which allowed for simpler management and better performance by eliminating the I/O adapters located in the PCIe drawers.

Example uses of SCM on z/OS:

- Improved paging performance
- Improved dump capture times
- Pageable 1MB memory objects
- Coupling Facility (CF) list structures used by MQ shared queues.

The use of MQ shared queues with SCM is discussed in detail in Redbook "IBM MQ V8 Features and Enhancements" which also suggests possible use cases.

These use cases include:

1. Emergency storage, i.e. increased capacity.
2. Improved performance, i.e. being able to store full MQ messages of 63KB or less on shared queues rather than offloading the message payload to SMDS or Db2.

Over time, the cost of real storage used by the Coupling Facility has significantly reduced. This coupled with the limited use cases for SCM with MQ shared queues has lessened the value of virtual flash memory such that on z17, SCM is no longer available for use with the Coupling Facility.

If SCM attributes are still referenced in the CFRM policy where the CF is defined to z17, there may be some reduced capacity in the structures. To avoid this reduced capacity, remove the SCM attributes from the CFRM policy that is defined with TYPE(009175).

The only time Storage Class Memory would be used is for shared queues with relatively high depth, so workloads where the depths remain low will be unaffected by the deprecation of SCM.

For workloads where queues may see high depth, it is important to consider the alternative configurations in order of performance:

1. Increase CF size with additional real storage.
2. Offload to SMDS to increase the capacity of the existing CF structure.
3. Offload to Db2.

It is worth re-iterating that the coupling facility will always be used for shared queue messages.

- For messages larger than 63KB, most of the message data will be offloaded either to SMDS or Db2, but key message information will be stored in the CF.
- For messages of 63KB or smaller, the size of the CF storage and the MQ SMDS offload rules will determine when the messages are stored in their entirety in the CF and when most of the message data is offloaded to either SMDS or Db2.

Regardless of message size, and number of messages on the queues, some part of the message will be stored in the CF and therefore even if SCM is replaced with SMDS offload, some additional CF storage may be required to ensure sufficient capacity in case of emergency. Review your CF requirements using the z/OSMF Sysplex Management service CFRM policy editor.

SCM used the KEYPRIORITY1 algorithm with MQ shared queues to determine the order that messages were written to SCM (pre-staging) and the order messages were migrated back into the CF (pre-fetching).

Both pre-staging and pre-fetching were typically performed asynchronously to reduce the chance of the application being blocked whilst waiting for synchronous I/O to/from SCM.

Pre-fetching using the KEYPRIORITY1 algorithm works on the assumption that messages will be gotten in MQ message priority order. Multiple messages are pre-fetched, the number dependent upon the message size.

When processing messages out of priority order, the pre-staging and pre-fetching were unable to accurately predict which messages could be pre-staged and which could be pre-fetched next. This could result in significantly more expensive MQPUTs and MQGETs.

With a Coupling Facility on z17, particularly where the CF structure is large enough to contain the workload, these issues are avoided.

Non-Sequential gets from deep shared queue

First off, a reminder than MQ is not a database and it performs best when the queue depths are kept low.

As such, frequent use of non-sequential gets from deep shared queues should be relatively infrequent behaviour.

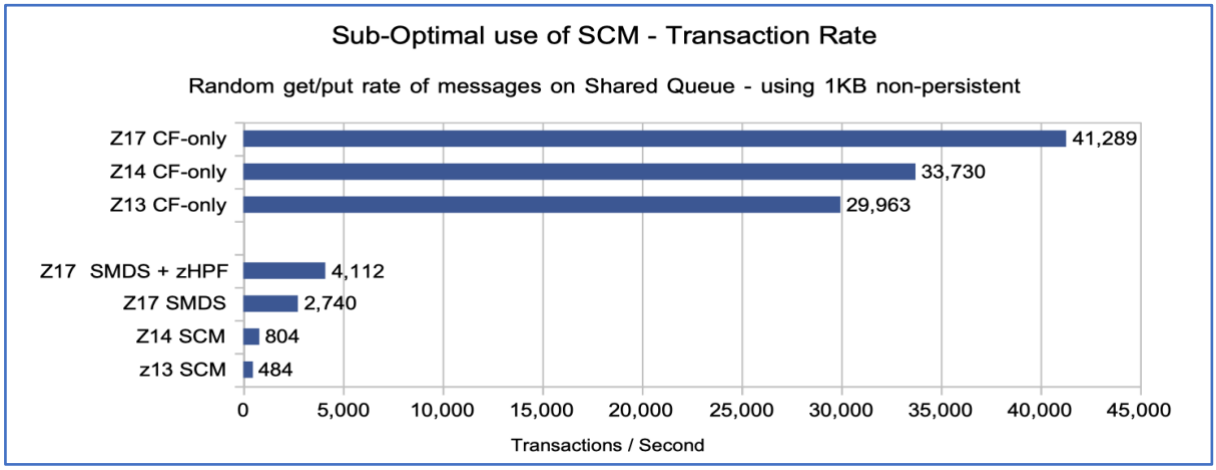
Whilst this behaviour may be infrequent, there are scenarios where it is unavoidable. As such we offer the following example of the impact to performance on MQ for z/OS.

The following scenario takes a shared queue with sufficient messages such that the message may be stored in a range of locations including CF, SCM or MQ's Shared Message Data Sets (SMDS).

In each case, an application randomly gets and puts a message using a pre-determined CORRELATION ID on a queue indexed by CORRELID.

In all cases, the messages used are 1KB non-persistent.

Transaction Rate:



Unfortunately, as this is not a scenario that we regularly run, we do not have data for IBM z15 or z16.

However, there is a clear difference in performance between the CF-only configurations and those using either storage class memory (on z13 and z14) and even using SMDS on z17.

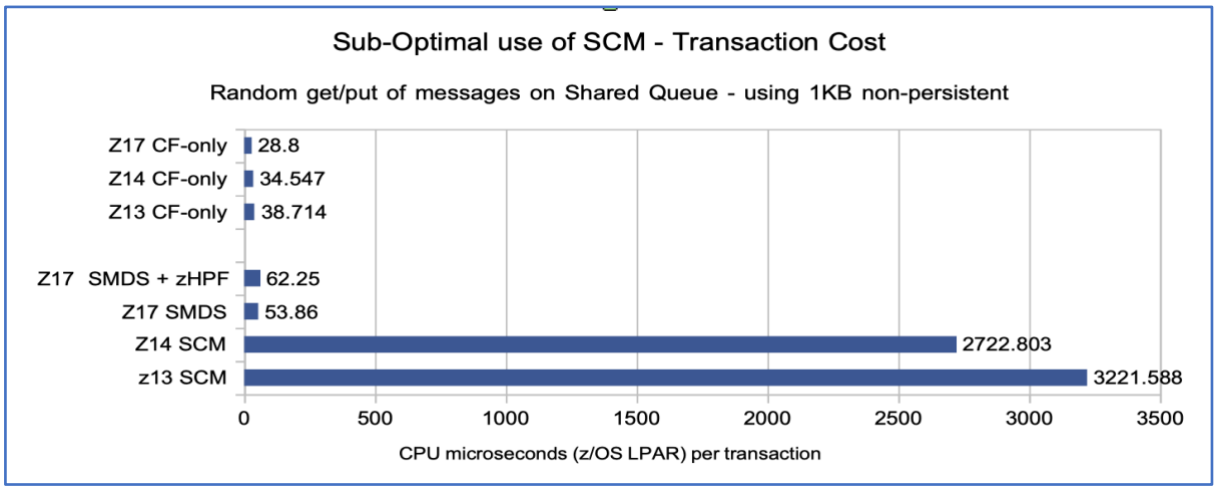
The SMDS performance is relatively low compared to the CF-only configuration as this is dependent on MQ having to perform disk I/O to read each 1KB message.
Performance using SMDS could be improved by:

- Enabling zHPF (z-high performance FICON), indeed this improved the transaction rate by 50%
- Increasing the number of buffers (DSBUFS) allocated for accessing Shared Message Data Sets. This will help the performance by keeping more of the messages in

storage, reducing disk I/O. However, a substantial increase in DSBUFs may significantly impact the amount of above bar queue manager storage.

For a 1KB message, using CF-only storage for the message set results in a consistent 10x improvement in transaction rate over any other configuration used here.

Transaction Cost:



The transaction cost of using either CF-only or SMDS in a non-sequential get from deep shared queue scenario is significantly smaller than when the MQ message data is stored in SCM.

Where is the additional cost in SMDS measurement over CF-only?
In our measurements on z17, comparing CF-only with SMDS using zHPF, the average cost of the MQGET and MQPUT transaction increase was 33.45 CPU microseconds.

This increase was attributed to 2 address spaces:

- The application accounted for 24 CPU microseconds of the increase.
- The MQ queue manager accounting for the remaining 9.45 CPU microseconds.

In the “non-sequential get from deep shared queue” scenario, the preferred performance configuration is to have sufficient CF real storage available to store the message in its entirety. Should the CF storage not be sufficient for the message set, then SMDS offload should achieve better performance than storage class memory would have on earlier generations of hardware.

Sequential gets from deep shared queue

When performing sequential gets from deep shared queues, the impact of losing the additional capacity that SCM offered becomes more relevant to the performance of the workload.

The deprecation of SCM means that a decision must be made on where the capacity for large volumes of messages should come from.

From a strictly performance perspective, storing the messages in their entirety in the Coupling Facility may give the best results, but it is essential to consider your own environment.

For example, an internal CF with fast links and sufficient processors may handle a larger coupling facility structure without a performance issue, whereas a remote coupling facility may benefit from choosing to offload most of the message payload to SMDS.

In our environment, where we have an internal CF with sufficient fast links and dedicated ICF processors, we found best performance was achieved with larger CF structures compared to smaller structures backed with SMDS offload.

Streaming Shared Queue Messages from z/OS to MQ client on Linux

To demonstrate the impact of the deprecation of Storage Class Memory on an MQ workload, particularly one using sequential gets from a deep shared queue, this section will compare 4 configurations:

1. IBM z16 configured with a 4GB structure and 1TB SCM
2. IBM z17 configured with a 4GB structure and SMDS offload, with zHPF disabled.
3. IBM z17 configured with a 4GB structure and SMDS offload, with zHPF enabled.
4. IBM z17 configured a 50GB structure.

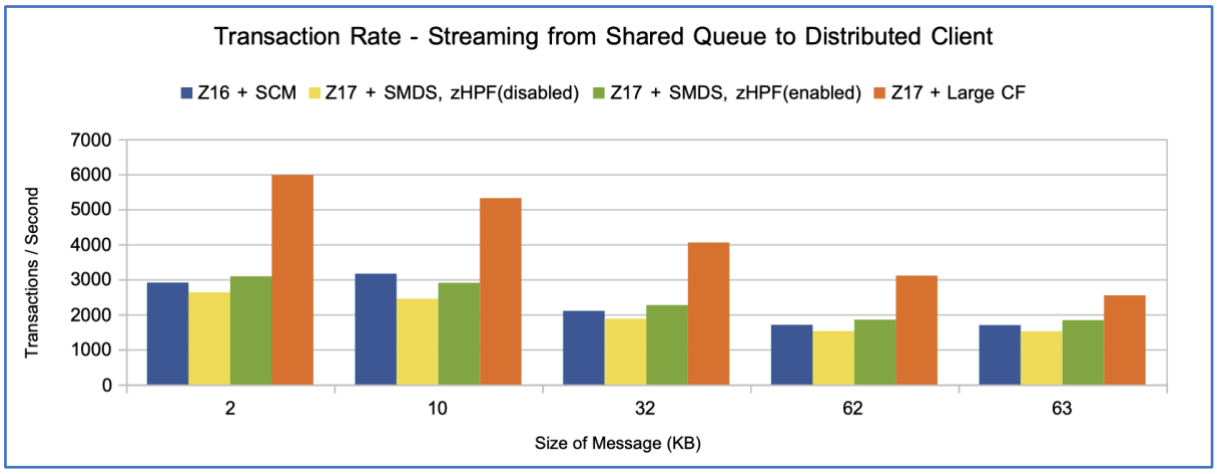
In each configuration, the workload is run using 5 message sizes namely 2KB, 10KB, 32KB, 62KB and 63KB. These message sizes are selected such that they can be hosted entirely in the Coupling Facility structure.

The IBM z17 configurations use Network Express for the network adapter but since the constraining factor is the CF response time, there was no benefit to throughput from this configuration.

The workload uses the following application model:

- On z/OS, a batch application puts persistent messages to the shared queue.
- On the Linux partner, an application connects to the z/OS queue manager and waits until there is 2GB of data on the queue before browsing the messages using pre-agreed CORRELATION IDs.
- Once the messages have been browsed in batches of 200, a second client application deletes the messages from the z/OS shared queue.

Transaction Rate:



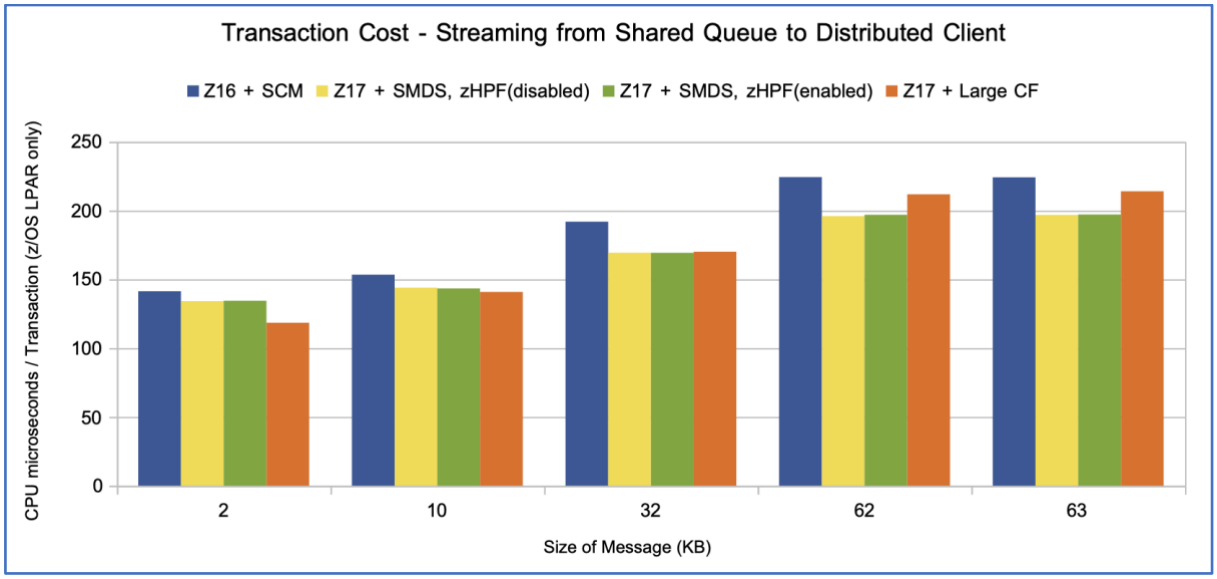
In our system configuration, moving from z16 with SCM to z17 with a large CF saw the transaction rate increase between 50 and 105%.

When comparing z16 with SCM against z17 with SMDS, provided zHPF is enabled, the performance is largely improved by 6-9% but the 10KB workload does show a drop in throughput of 8%.

Analysis of the z17 with SMDS 10KB workload shows that the workload was waiting for I/O read and writes of the SMDS' which impacted the elapsed time of the MQPUT and MQGET APIs. This caused the workload rate to be lower than the z16 measurement using SCM.

The transaction rate for non-zHPF using SMDS measurements on z17 were between 10 to 22% lower than z16. *We have been made aware of FICON performance issues on IBM z17 up to the date of publishing this report.*

Transaction Cost:



In our system configuration, moving from z16 with SCM to z17 with a large CF saw the transaction cost decrease by 5 to 16%.

Similar ranges of transaction cost reduction were observed in both z17 with SMDS configurations (5 to 13% reduction).

Comparing the performance of 62KB messages

Whilst the performance of z17 using CF has been observed to exceed the performance of the z16 configuration using SCM, the previous chart reporting the transaction cost does show that for both 62K and 63K messages, the z17 with large CF is more expensive than the z17 with SMDS configurations.

The z17 with large CF workload with 62K messages is costing 7.5% more CPU per transaction than the equivalent z17 with SMDS.

This additional CPU is observed in both the MQ MSTR and channel initiator address spaces. According to the MQ Accounting data, it is also observed in the MQPUT and MQGET APIs. Additionally, we see extended MQ commit times (elapsed) in the large CF workload.

The reason for this is that when using SMDS offload, only a small part of the MQ message is stored in the coupling facility.

For example, the MQPUT on our SMDS configuration has a CF interaction of type “new” which costs 3 CPU microseconds and is performed synchronously. Additionally, the MQPUT will cause data to be written to the shared message data set, which is largely asynchronous I/O to disk. This results in the accounting data showing the average MQPUT cost 23 microseconds, but an elapsed time of 379 microseconds.

By contrast, when using the CF to store the message in its entirety, the CF interaction of type “new” is reported as costing 21 microseconds, again synchronously. Additional MQ processing means that the accounting data shows the average MQPUT costing 49 microseconds, an increase of 26 microseconds per MQPUT over the SMDS configuration. The elapsed time for the average MQPUT is reduced from 379 microseconds to 114 microseconds.

A similar story is observed on the MQGETs.
When the get from SMDS configuration is invoked, only a small amount of data is retrieved from the CF such that the MQGET costs average 21 microseconds, with an elapsed time of 134 microseconds. Of the 21 microseconds, 8 microseconds are incurred waiting for synchronous CF responses.

For the MQGET when the message is entirely in the CF, the MQGET costs 35 microseconds but has an elapsed time of only 80 microseconds – higher cost but less waiting for disk I/O reads. The higher cost is in part due to the increased cost of CF interaction, costing 27

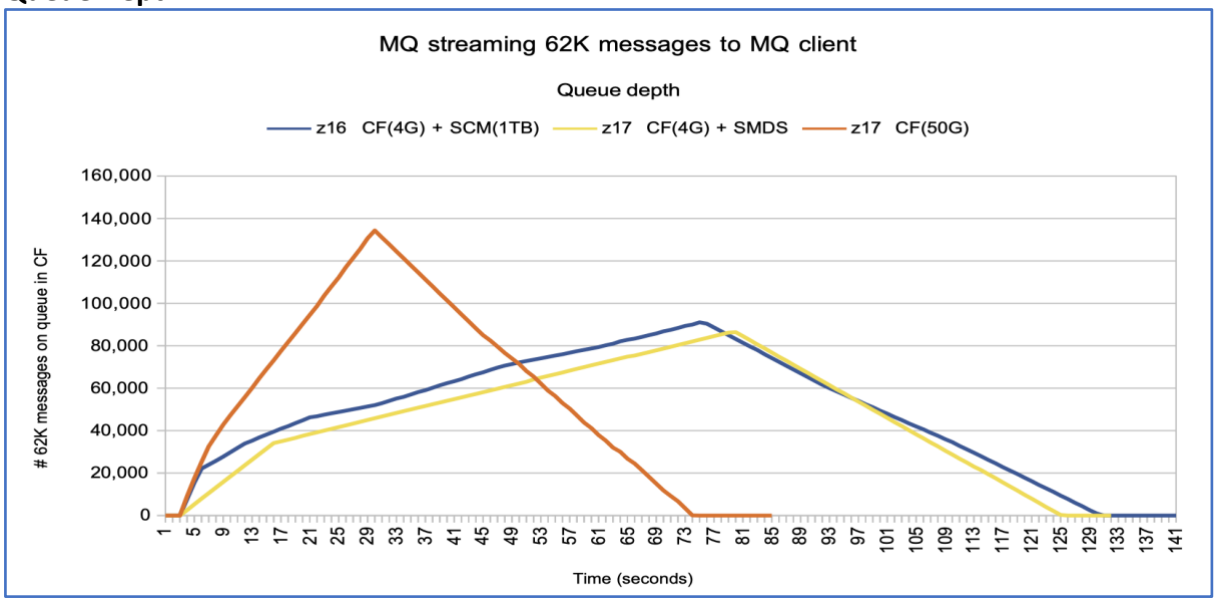
microseconds. In this instance, as the data being read is larger, some of the CF response is asynchronous.

To further add to the complexity, whilst the total data being logged remains the same between runs, because the MQPUTs and MQGETs are occurring faster with the large CF, there is more data (pages) being written the MQ log task per write request, which results in the commit elapsed time being higher for the large CF configuration.

Ultimately the desire for this workload is to complete as quickly as possible.

By plotting and comparing the depth of the deep shared queue, we can illustrate the benefit of using a large CF. Additionally the below chart shows that the z17 SMDS configuration achieves a shorter elapsed time than the z16 configuration.

Queue Depth:



Notes of Queue Depth chart:

- The z17 large CF configuration completes the work in 75 seconds, much sooner than either of the other configurations which take 125 seconds or more.
- The z17 with SMDS shows very similar performance characteristics to the z16 configuration – which is to be expected as for this message size, the z16 configuration is also using SMDS to store most of the message data.
- In all cases, the queue depth increases until all messages have been put, i.e. the local put is faster than the client-side gets.
- The steepness of the depth increase is greater for the large CF configuration – puts to CF are faster – as demonstrated by the lower elapsed time per MQPUT.
- The steepness of the depth decrease is greater for the large CF configuration – gets from CF are faster than gets from SMDS because the MQGET is not waiting for disk I/O to complete.

Performance of the large CF measurement on z17 was not impacted by using Network Express – in our measurements, we saw a similar pattern of queue depth, and performance when using the OSA adapters.

Conclusion

The choice of how to replace the removed SCM function will depend on many factors including:

1. Message sizes used – smaller messages of 32K or less will likely perform better in a large CF.
2. Disk response time – SMDS I/O will take time to complete. If message sizes are of a consistent size, some benefit may be achieved by tuning the DSBLOCK size. In our measurements, the DSBLOCK was set to the default 256K which has historically given the best performance for a range of message sizes on our system but may not be optimal if all messages were smaller.
3. CF location, links etc – for a remote or less responsive CF, offloading data to SMDS earlier may be beneficial to performance.
4. Transaction cost – in some circumstances, we observed higher costs when avoiding SMDS activity due to higher CF synchronous costs. If cost is a key metric, using SMDS may help reduce cost.
5. Whether a deep queue is expected to be a common occurrence. Should deep queue only be in emergency situations, SMDS may be an appropriate option.

..

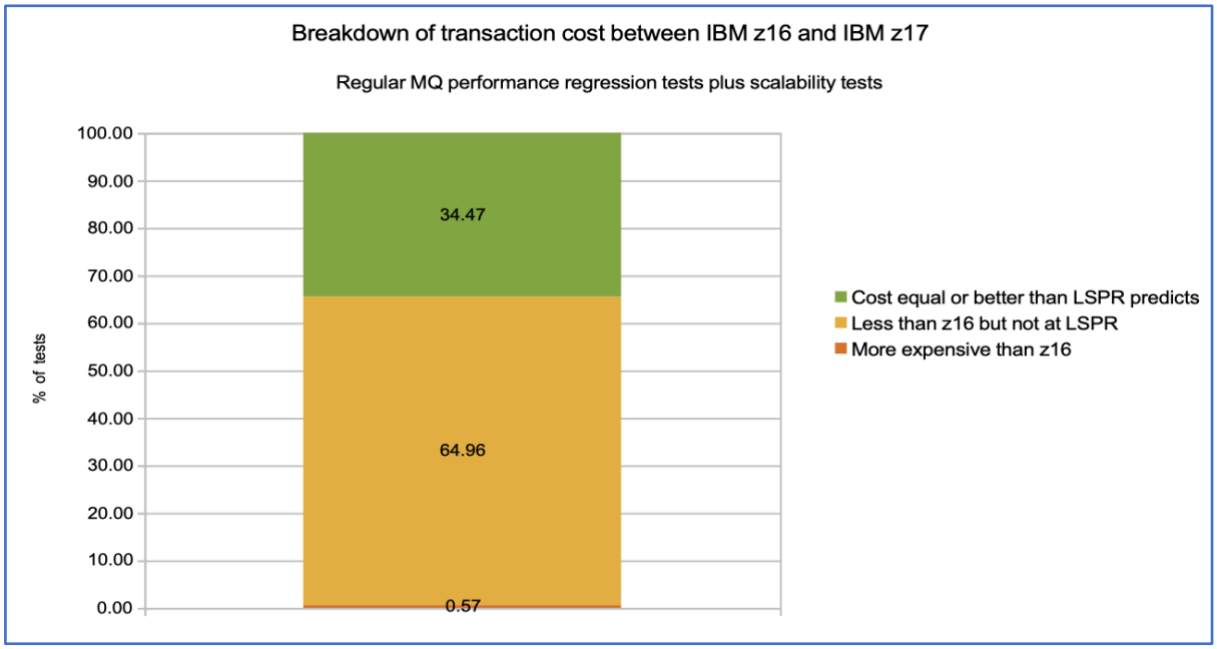
5 General test performance and scalability

MQ Performance runs a set of regression tests on a regular basis, and this provides the ability to track a range of MQ micro-benchmarks for variations due to code changes as well as system changes, perhaps due to maintenance to z/OS or other middleware products.

At an Long-Term Service (LTS) release, there are additional tests run and the performance of those is documented in the MQ for z/OS 9.4 performance report in "Appendix A – Regression".

With a new hardware release, there are additional scalability tests run which provide the ability to compare how workloads running on higher n-way LPARs may perform with regards to previous generations of hardware.

For this document, we initially moved the MQ performance system in its entirety to IBM z17 and ran all the micro-benchmarks mentioned, and these were run in the same configurations as on IBM z16. Analysis of these measurements saw a spread of results as summarised in the following chart:

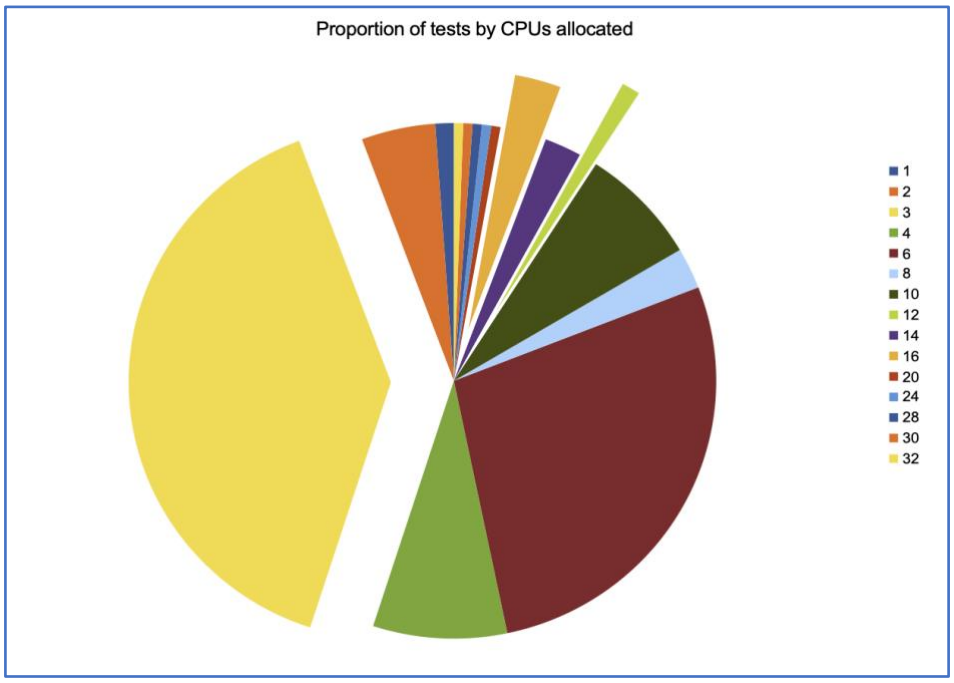


- 34.4% of micro-benchmarks have reduced cost by the expectations set by LSPR or better.
- 99.4% of micro-benchmarks have lower costs than the equivalent measurement run on IBM z16.

Taking these combined results, the tests can be categorized based on the number of CPUs, and we can generate the following charts:

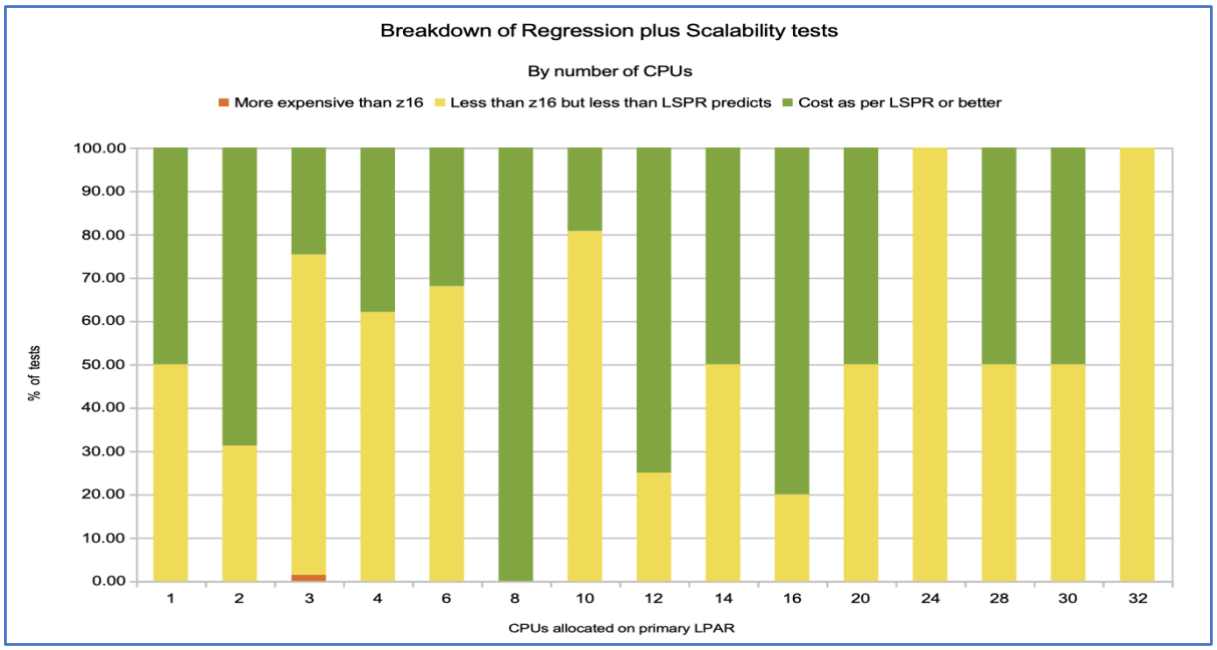
1. Proportion of tests run on a particular CPU allocation.
2. Breakdown of transaction cost relative to IBM z16 as a percentage, by CPU allocation.

Chart: Proportion of tests run on a particular CPU allocation:



The preceding chart shows that 75% of the tests run are run on LPARs with 3, 4 or 6 CPUs, with the remaining twelve CPU configurations accounting for just 25% of the tests run.

Chart: Breakdown of transaction cost relative to IBM z16 as a percentage.



If we concentrate on the tests showing *costs that are higher on IBM z17 than z16*, there is one set:

- **3 CPUs:** Message select-on-get using message properties.

Whilst there is a tendency to focus on what has changed, for example increased processor speed, Network Express, it is important to recognise that some features have not seen a significant change in performance.

For example, measurements using cryptographic or compression hardware are not always seeing an LSPR-like improvement in performance. On both IBM z16 and IBM z17, the latest available cryptographic hardware remains at CryptoExpress 8S.

That is not to suggest that our benchmarks using TLS ciphers to protect message data flowing over MQ channels do not see improvements in performance but where we do see improvements, it is more related to the benefits of Network Express and the CPU improvements, whereas on z16 where CryptoExpress 8S was introduced, there were benefits in response times when compared with CryptoExpress 7S.

General test performance

For the performance tests we typically saw performance in-line with the *lower end* of expected results from the LSPR tables, although there were some notable exceptions – excluding those tests benefitting from Network Express.

- Persistent message tests demonstrated cost reductions of up to 13.8%.

- Queues protected with AMS policies
  Transaction costs were up to 20% lower on IBM z17, with an increase in transaction rate of up to 20% with larger messages.

- Scalability tests achieving up to 10% higher throughput
  1. Non-persistent in-syncpoint workload achieved more than 0.74 million messages per second on a single z/OS LPAR with 28 or more CPUs, an increase up to 15% over IBM z16.
  2. Non-persistent out-of-syncpoint workload achieved more than 1.5 million messages per second on single z/OS LPAR with 24 or more CPUs.
     a. When the enhanced accounting and statistics data trace was disabled, i.e. trace(a) class(3), the same measurement was able to exceed 2 million messages per second.
     b. When using queue statistics, i.e. trace(s) class(5), and enabling STATQ(ON) for each of the active queues, the same measurement was able to attain 1.96 million messages per second.

Performance of MQ persistent message benchmarks

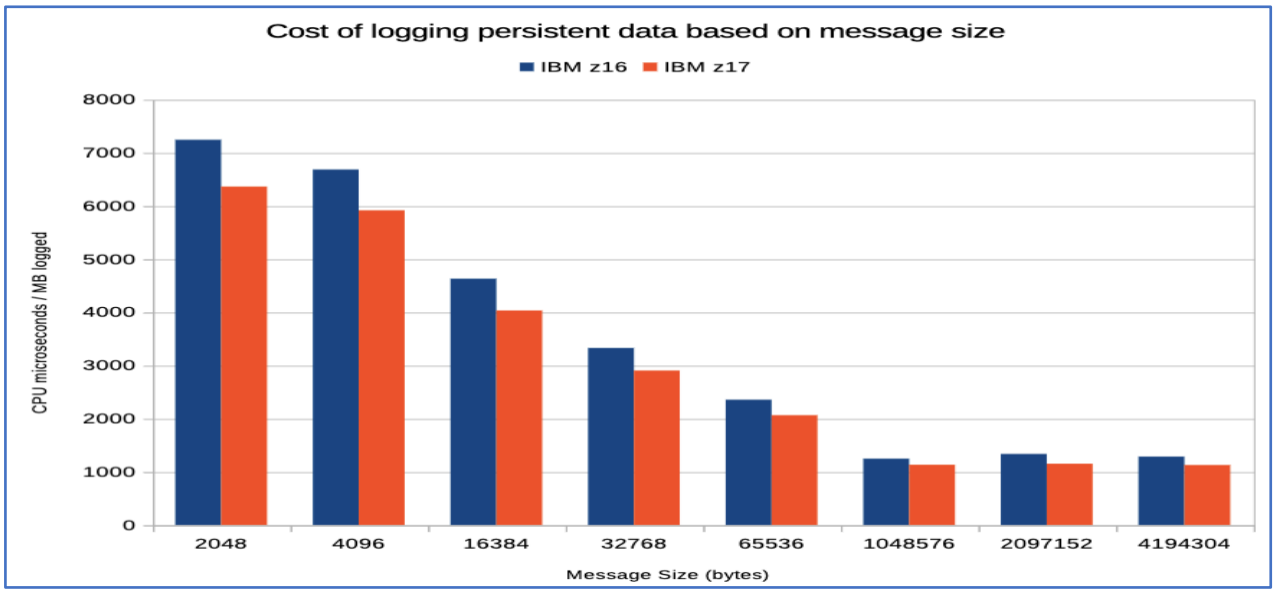
In the MQ for z/OS 9.4 performance report, the section reporting the performance of regression tests included a chart showing the "upper bounds of persistent logging rate", i.e. the maximum logging rate that MQ was able to sustain in our test environment.

The test used 3 batch tasks that each put and got messages from a common queue. One of the tasks used a 1KB persistent message, and the remaining 2 tasks vary the size of the message from 1KB to 4MB.

The chart demonstrated that in a dual log/dual archive configuration with 4 stripes on the active logs, the queue manager was able to sustain up to 447MB per second per log copy.

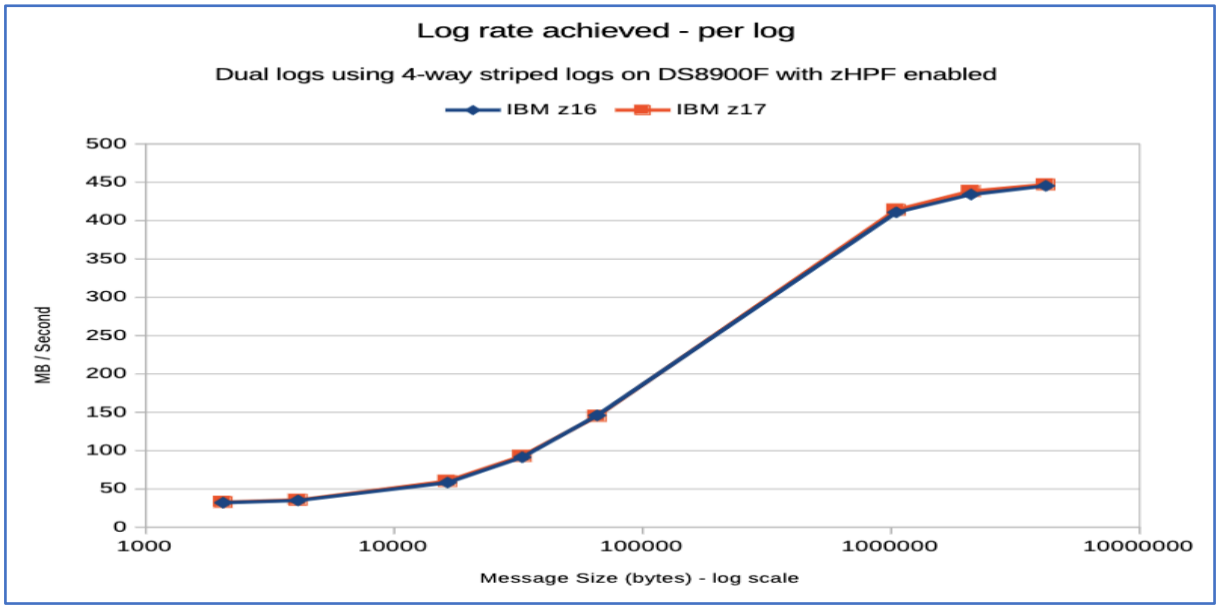
As mentioned in the "tempering expectations" section of this document, we were not expecting to see significant improvements to the log rate achieved by these benchmarks when moving to IBM z17 as the underlying disk infrastructure has not been changed. However, we would expect the transaction cost to reduce by the values suggested in the LSPR tables.

The following chart shows a comparison of the cost per MB logged for this workload.



With regards to the cost of logging the persistent messages, on IBM z17 the costs are reduced by between 9.2 to 13.8% over the equivalent test on IBM z16.

The second chart shows that, as expected, the log rate achieved on IBM z17 is like that achieved on IBM z16:



Performance of Queues protected using AMS Policies

When comparing the performance of queues protected with AMS policies we used a simple request/reply model using small (2KB), medium (64KB) and large (4MB) messages.

The policies were applied to both the request and reply-to queues where:

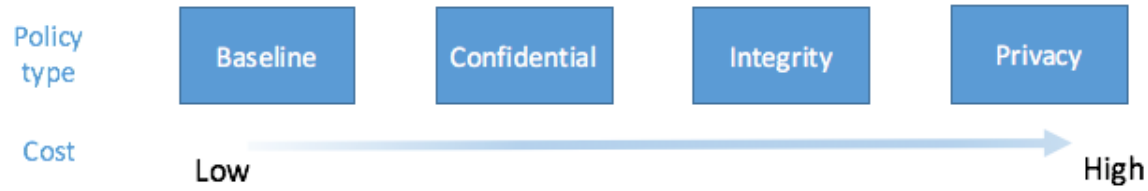
- Integrity used messages signed with SHA256.
- Privacy used message signed with SHA256 and encrypted using AES256.
- Confidentiality used messages encrypted using AES256 and the key reused 32 times.

AMS Integrity and AMS Privacy rely heavily on the response times of the Crypto Express features, whereas AMS Confidentiality will only use the Crypto Express features when the renegotiating the key.

AMS Privacy and AMS Confidentiality use CPACF to encrypt and decrypt the data.

In terms of Crypto Express usage, message size does not impact the cost protecting the message using either AMS Integrity or AMS Privacy. With regards to AMS Confidentiality, the use of Crypto Express depends on the frequency of the key reuse. If the key is reused an unlimited number of times, there will be minimal use of the Crypto Express features.

In basic terms, the costs of AMS protection can be considered thus:



The performance of our tests with queues protected by AMS policies overall improved in-line with LSPR expectations when moving from z16 to z17. Small message workloads have seen the smallest improvement in terms of both cost reduction and throughput increase, whereas larger message workloads improved at rates predicted by LSPR or higher.

The following tables indicate the change in performance between z16 and z17.

% Change in transaction cost from z16 to z17:

Message Size	Integrity	Privacy	Confidentiality
Small	-14	-14.3	-7.8
Medium	-19.5	-16.5	-8.7
Large	-20.7	-18.2	-7.1

% Change in throughput from z16 to z17:

Message Size	Integrity	Privacy	Confidentiality
Small	+3.8	+3.7	+8.3
Medium	+9.5	+7.7	+8.1
Large	+20.6	+18	+8.1

Scalability

For our scalability measurements we typically start with a non-persistent workload with the intent to be CPU limited rather than log constrained. The measurements use specific queues that are allocated to separate buffer pools and page sets for each workload to minimize any contention.

The workloads highlighted in this document are:

1. Non-persistent in-syncpoint using 2KB messages.
2. Non-persistent out-of-syncpoint using 2KB messages.
    a. Impact of accounting and statistics traces on low cost MQ workload

Basic configuration

Initially a pair of tasks are started, one requester and one server. These tasks use a pair of queues, one for the request message and one for the reply message. These queues are defined such that they use the same buffer pool and page set.

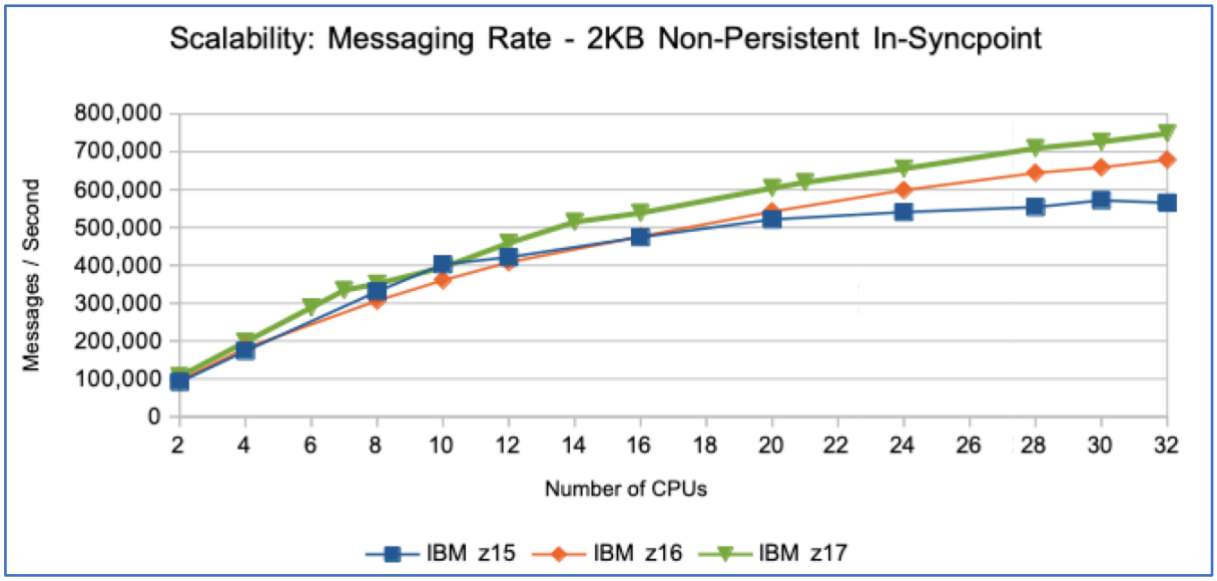
The requester puts a message and waits for a specific reply. When the requester gets that message, it generates a new request message, and this continues until the applications are requested to end.

The server waits for a request message and upon successful get, generates a reply message, and puts to the reply queue. When sync point is requested, the get and put will be performed within syncpoint.

The workload is increased with additional tasks that will use their own request and reply-to queues until there are 32 requesters, 32 servers and 32 pairs of queues. Each pair of queues is defined to a separate buffer pool and page set.

Non-persistent in-syncpoint using 2KB messages

The following chart shows the peak transaction rate achieved when the number of CPUs is increased.

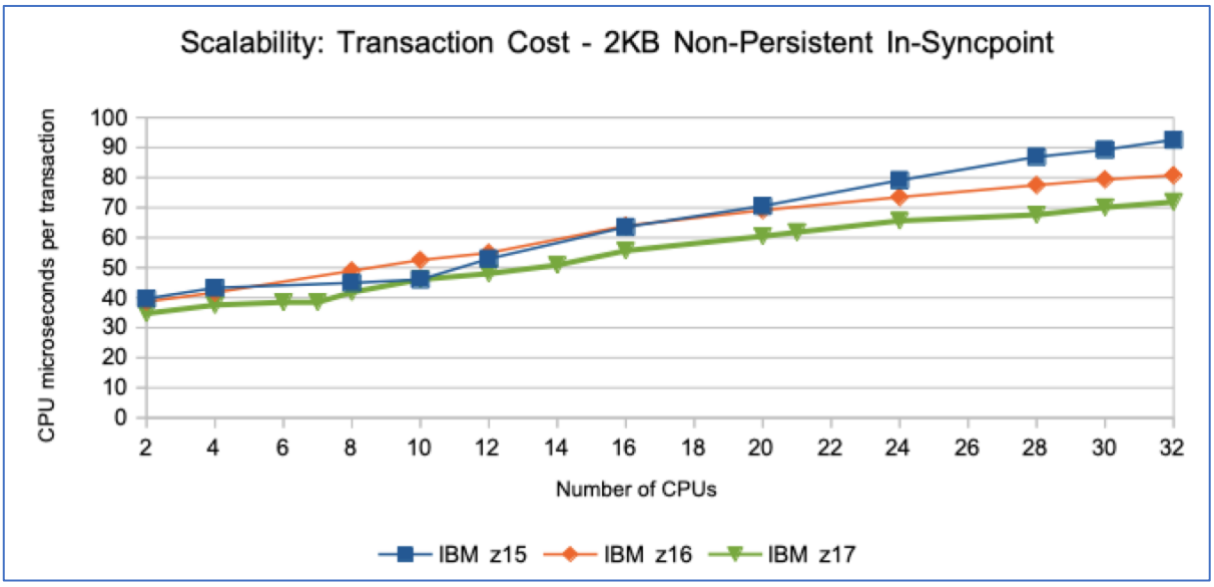


On IBM z15, the peak throughput of 571,600 messages per second, which equates to 285,800 transactions per second on a single MQ queue manager.

On IBM z16, the peak throughput of 678,790 messages per second, or 339,395 transactions per second on a single MQ queue manager and is an *increase of 18.7%* in maximum throughput.

On IBM z17, the peak throughput of 747,970 messages per second, or 373,985 transactions per second on a single MQ queue manager and is an 10% over the equivalent IBM z16 measurement.

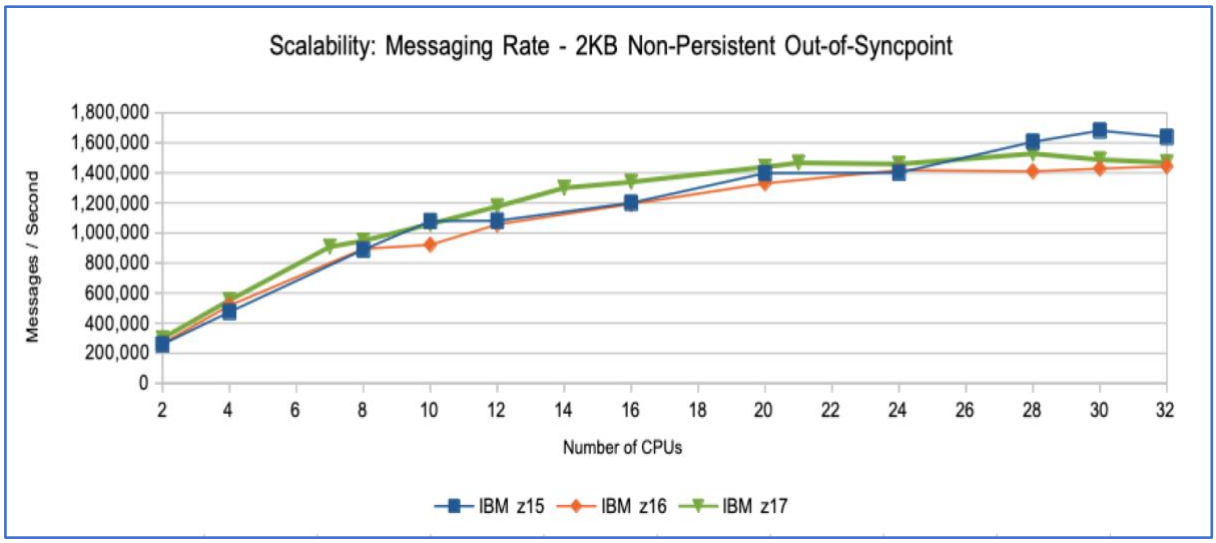
The following chart shows the cost of a transaction when the workload is running at peak throughput:



Cost per transaction on IBM z17 for the 2KB non-persistent in-syncpoint workload are generally lower than the equivalent test on IBM z16, with the reduction being between 10 and 15%.

Non-persistent out-of-syncpoint using 2KB messages

The following chart shows the peak transaction rate achieved when the number of CPUs is increased.



On IBM z15, the peak throughput of 1,681,500 messages per second, was achieved with 30 CPUs. This equates to 840,750 transactions per second on a single MQ queue manager.

On IBM z16, the peak throughput of 1,443,390 messages per second, was achieved with 32 CPUs. This equates to 721,695 transactions per second on a single MQ queue manager and is a *decrease of 14%* in maximum throughput.

On IBM z17, the peak throughput of 1,528,070 messages per second, was achieved with 28 processors. This equates to 764,035 transactions per second on a single MQ queue manager and is an increase of 5.9% on IBM z16's performance.

As a reference, this same test achieved the following results on recent generations of IBM mainframes:

Generation	Peak messaging rate/second achieved
IBM z13	1.15 million
IBM z14	1.28 million
IBM z15	1.68 million *(1.52 million at GA)*
IBM z16	1.44 million
IBM z17	1.52 million

Why the decrease in peak transaction rate?

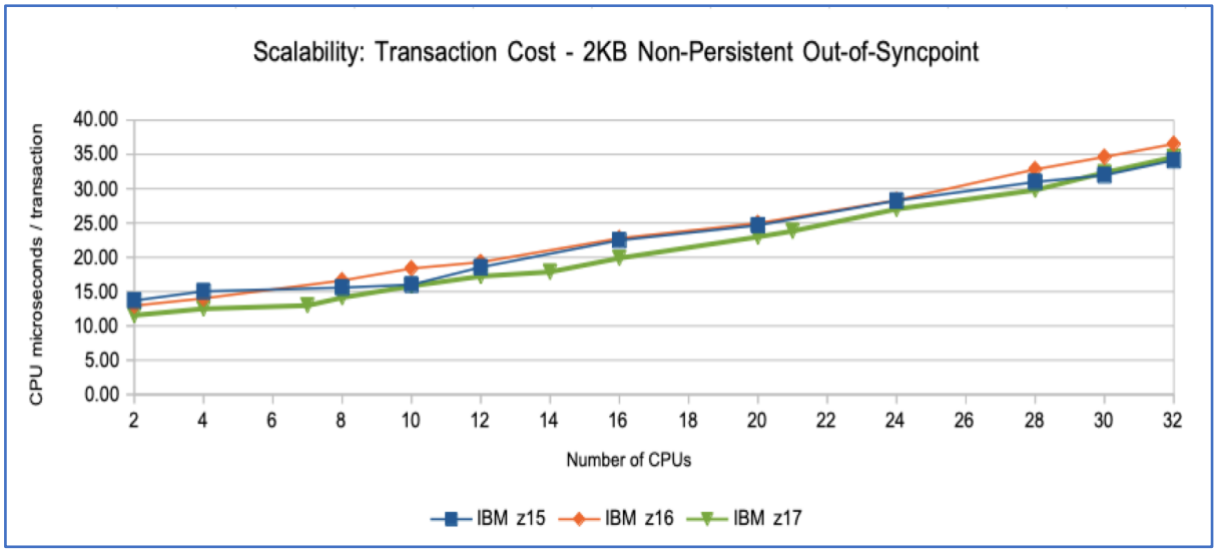
At least part of the difference in performance degradation from IBM z15 to IBM z16 and IBM z17, can be attributed to the number of cores per processor unit (PU).

This workload is a very tight loop of messaging workload, which due to the efficiency of MQ's put-to-waiting-getter, has a relatively small path length. On a low n-way system, the work can largely be kept in L1 and L2 cache, but as the number of CPUs increases, the workload must use more L3, L4 and memory which affects the performance of the workload.

Disabling Accounting trace class 3 from the IBM z16 measurement enabled a peak rate of 1.9 million messages per second, again with 32 CPUs allocated. However, it has not been determined how much impact this same action would have on the equivalent IBM z15 measurement.

Disabling accounting trace class 3 from the IBM z17 measurement enabled a peak rate of 2.0 million messages per second with 28 CPUs allocated.

The following chart shows the cost of a transaction when the workload is running at peak throughput, with class(3) accounting trace enabled on all hardware levels.



On IBM z17 up to 16 CPUs, the average cost per transaction when running at peak transaction rate is on average 12% lower than the equivalent on IBM z16.

When 20-32 CPUs are allocated, the cost per transaction on IBM z17 was determined to be 7% lower than the equivalent on IBM z16.

Impact of accounting and statistics traces on low cost MQ workload

The MP16 Capacity Planning and Tuning for IBM MQ for z/OS performance report breaks down the cost of the trace(a) class(3) "enhanced accounting and statistics data" into two phases:

1. Data gathering – 1-3 microseconds per API
2. Writing to SMF – 6-50 microseconds for the primary SMF record, depending on size, plus up to 60% of the cost of writing the primary SMF record for each continuation record, depending on how many queues the continuation record contains.

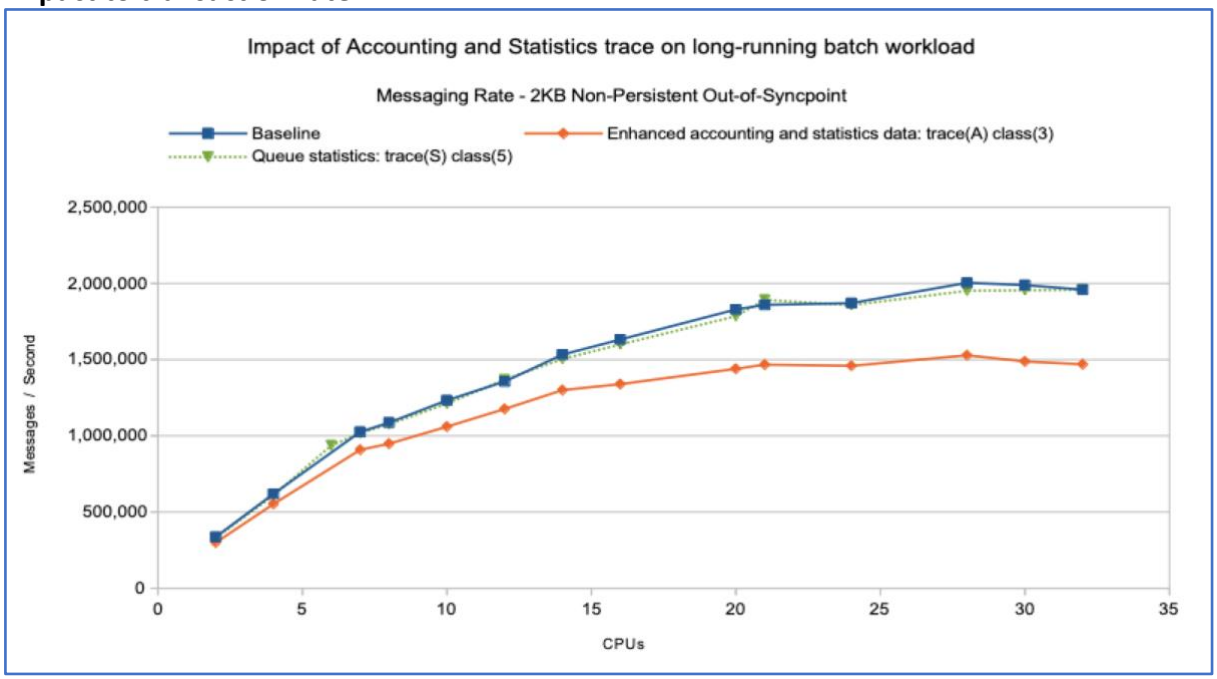
In terms of the impact to a workload, these costs may or may not be insignificant, but this section demonstrates the impact of the additional cost of collecting these statistics in a high-throughput, low-cost workload.

As the charts demonstrate, as the workload rate increases, the impact of the accounting trace grows, increasing from 10% in the 2 CPU measurement to more than 20% with 24 or more CPUs.

This impact may be more significant with short-lived transactions, such as CICS transactions, as the expensive part is the writing to SMF, which occurs at end of SMF interval and at tend transaction.

Fortunately Queue Statistics trace, i.e. trace(s) class(5), introduced in MQ for z/OS 9.3 but enhanced in MQ for z/OS 9.4, offers a lower impact approach to MQ queue monitoring such that on IBM z17 the impact of Queue Statistics trace was the equivalent of 0.08 microseconds per MQ API.

Impact to transaction rate:



As the chart demonstrates, there is an increasing impact from trace(a) class(3) as the workload scales up to 32 CPUs, such that the transaction rate is 33% lower than the rate achieved with trace disabled.

However, the queue statistics trace, where all used application queues has STATQ(ON), was able to sustain transaction rates on average within 1% of the workload with trace disabled.

Impact to transaction cost:

As with the transaction rate chart, the transaction cost when trace(a) class(3) is enabled is higher than when the measurement is run with trace disabled.
At 2 CPUs this equates to 2.2 microseconds, but at 30 CPUs this equates to 10 microseconds per transaction, or 2.5 microseconds per MQ API.

By contrast, the queue statistics measurement saw an insignificant increase in transaction cost of 0.08 microseconds per MQ API, regardless of workload or number of CPUs.

Appendix A – Test Environment

Measurements were performed using:

The IBM MQ performance sysplex ran measurements on:

- **IBM z16 (3931-7K0) – 4 CPC drawers (Max200)**
- **IBM z17 (9175-ME1) - 4 CPC drawers (Max208)**
  - **Minimum bundle level: S16**

The sysplex was configured thus:

- LPAR 1:
  - 1-32 dedicated CP plus 2 zIIP with 128 GB of real storage plus 16GB of dedicated memory.
- LPAR 2:
  - 1-10 dedicated CP plus 2 zIIP with 80 GB of real storage plus 16GB of dedicated memory.
- LPAR 3:
  - 1-3 dedicated CP with 44 GB of real storage plus 4GB of dedicated memory.
- z/OS v3r1.
- Db2 for z/OS version 13 configured for MQ using Universal Table spaces.
- IMS 15.5
- IBM CICS CTS 6.2
- MQ queue managers:
  - configured at MQ 9.4.
  - configured with dual logs and dual archives.

Coupling Facility:

- Internal Coupling Facility with 4 dedicated processors
- Coupling Facility running latest CFCC level.
- Dynamic CF dispatching off
- 3 x ICP links between z/OS LPARs and CF.

DASD:

- FICON Express 32S connected DS8900F
- 4 dedicated channel paths
- HYPERPAV enabled
- zHPF enabled unless otherwise specified.

Network:

- 10GbE network configured with minimal hops to distributed partner machines
- 1GbE network available
- OSA cards
- Network Express adapter

Applications written in a mixture of:

- C
- COBOL compiled with Enterprise COBOL for z/OS 6.4 with options ARCH(13) and OPT(1).

Appendix B – APARs applied

OA68126

UO04442
UJ97597
PH68114

Network Express
UO04064
PH66614
PH66746
PH67139
PH67370
PH67851
OA68211
OA67935