IBM MQ for z/OS on z16 Performance

Version 1.0 – September 2022

Tony Sharkey

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 9.3 running on IBM z16.* 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, I will be looking at the improvements to our performance tests on MQ for z/OS as we moved from IBM z15 to IBM z16.

This paper is split into several parts:
Part one - Setting expectations of the hardware move.
Part two - What's new or changed on IBM z16.
Part three - Improvements from Crypto Express 8S.
Part four - General MQ test performance and scalability.

Part one describes what may impact the expectations of moving workload from z15 to z16, and why it is not always straightforward.

Part two discusses the differences between IBM z15 and z16, particularly the CPU changes and CFCC 25, and how those changes may affect your own systems.

Part three looks at the performance benefits to components of MQ that utilize the cryptographic facilities offered on IBM Z, namely those using Cryptographic co-processor and accelerator function.

Part four presents the results of measurements run first on z15 and then subsequently on z16. 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® z16™ (z16) offers many improvements over IBM z15 but of note are an increase in the number of processors available, increased L2 cache size, virtual L3 cache up to 256MB per CP and Virtual L4 cache up to 2GB per drawer.

When trying to set the expectations of the benefits of moving to the z16 there are several good starting points:

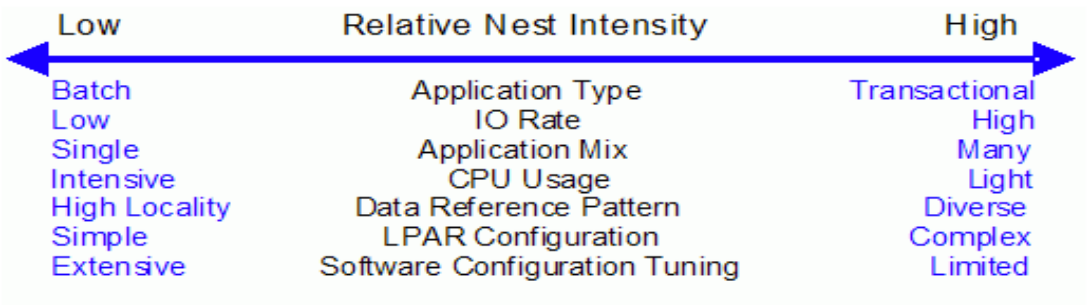
- Redbook "IBM z16 (3931) Technical Guide", in particular chapter 12 "Performance".
- Website "Large System Performance Reference (LSPR) for IBM Z".

It is important to recognize that MQ is generally a small part of your system solution and the IBM z16 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 z16, 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 CP's but average when running on 32 CP's.

The following table shows the expected improvement on z16 over z15 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%	+14%
32	+10%	+12%	+14%

What this table suggests is that depending on the workload type and the number of CPUs allocated, we may see between 9 and 14% improvement over comparable measurements on z15.

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 9-14% improvements over equivalent workloads on z15. 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.

Network - In our test configuration, we are still reliant on the same 10Gb and 1Gb network links between z/OS LPARs and distributed partner machines. As such, any improvements observed would likely be due to reduced transaction cost on the z/OS LPAR and any limitations on round-trip times may largely remain due to network performance not being affected.

Cryptographic response times will be discussed in detail in "Improvements from Crypto Express 8S" 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 14%.

2 What's new or changed on IBM z16?

The Redbook "IBM z16 (3931) Technical Guide" provides a detailed specification of the IBM z16, but what follows offers a comparison of the changes from the IBM z15.

Machine	**IBM z15 Model T01 Max190**	**IBM z16 Model A01 Max200**
Drawers	5	4
Processor units (PU) per drawer	4 *Single Chip Modules (SCM)*	4 *Dual Chip modules (DCM)*
Processors (cores) per PU	12	8
Active cores per PU	9-11 for each SCM ***Our SYSPLEX:*** *10 per PU*	9-11 or 10-15 for each DCM ***Our SYSPLEX:*** *12 for each PU, 6 per chip*
Cores per drawer	48	64
Cache		
L1	128KB	256KB
L2	4MB per core, 48MB per PU	32MB per PU *(Effectively 4MB per core)*
L3	256MB	256MB
L4	960MB / drawer	2GB

Processors

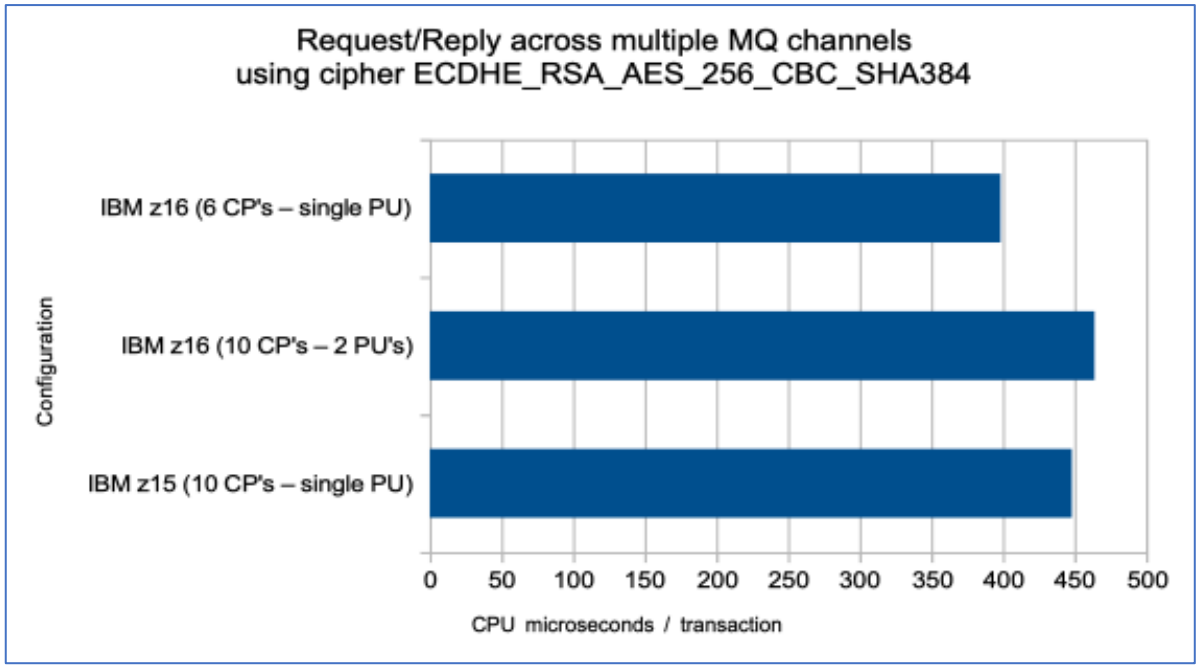
In our micro-benchmarks we have always seen an impact to transaction cost when running a workload that spans into a second and subsequent processor unit (PU), and with the reduction in cores on a PU on IBM z16, this impact can come earlier and more frequently.

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

On IBM z16 there are 8 cores per PU – and on our systems we are seeing a maximum of 6 processors being allocated per PU.

In several of our micro-benchmarks on IBM z16, we have observed the performance did not match the expectation set by the LSPR tables. This is discussed in more detail in the "General test performance and scalability" section, but a large proportion of those tests that did not perform to expectations were tests which on IBM z15 would fit onto a single PU but on IBM z16 spilled over onto multiple PU's.

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 multiple MQ channels protected with cipher `ECDHE_RSA_AES_256_CBC_SHA384`.



In the original measurements on both IBM z15 and z16, the test was configured with 10 CPUs on each LPAR. The IBM z16 transaction cost when using 10 CPU's per LPAR was 3.5% higher than on IBM z15.

By re-configuring the LPARs to fit into a single PU, i.e., 6 CPUs per LPAR, the subsequent run of the benchmark saw the cost reduced by 11% over the IBM z15 measurement and 14% over the 10 CPU measurement on IBM z16.

In these measurements, the workload did not require 10 CPUs per LPAR – and as a result, decreasing the number of CPUs allocated did not negatively affect the throughput. Indeed, the throughput increased by 15% as there was the same number of tasks involved and the CPU time spent in each task was reduced.

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 some tests with 10 CPUs per LPAR entirely for comparison purposes – for example we configured a 32KB workload that uses few MQ channels and uses a small proportion of the allocated CPUs and compare against a test that uses many MQ channels and requires all 10 CPUs per LPAR. On IBM z15 this gives a reasonable comparison, but on IBM z16 there is less value due to spanning multiple PU's.

Workloads with larger path lengths, potentially due to far more complex processing than the MQ micro-benchmarks, may see less impact when running in LPARs configured such that the CPUs span multiple PU's. In this case, the more complex workload will require more cache on both IBM z15 and IBM z16 and may see benefits as suggested by LSPR.

Coupling Facility - CFCC Level 25

CFCC level 25 is delivered on the IBM z16 (3931) and adds several new features.

From an MQ perspective the most notable, is the deprecation of DYNDISP=ON|OFF for shared engine CF images.

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.

Previously 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 LPAR. 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 LPAR 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.

In 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 25 and MQ for z/OS structures

The IBM CF Sizer for MQ tool should be used for accurate sizing of your MQ Coupling Facility structures.

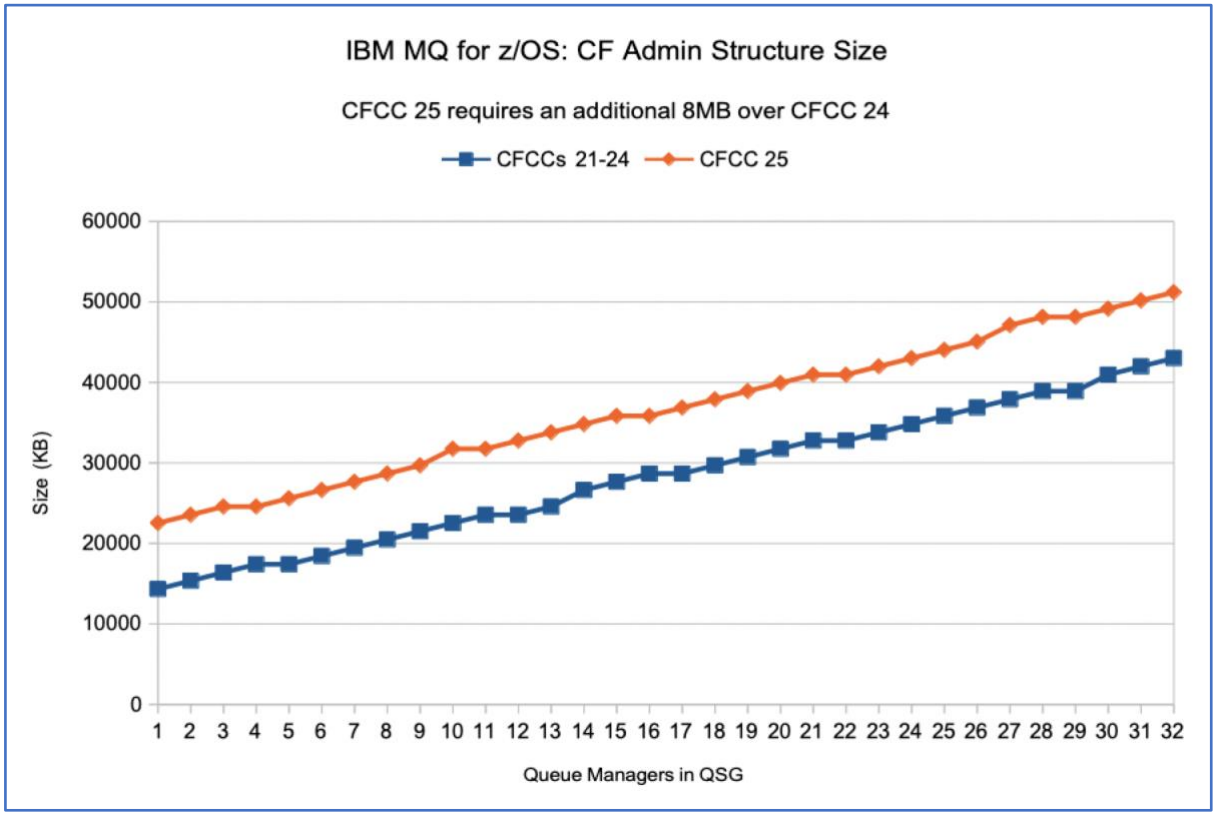
The following subsection offers an indication of the increase storage requirements for MQ CF structures that we observed moving to IBM z16.

Does CFCC level 25 affect the size of my ADMIN structure?

Yes, CFCC level 25 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 25, 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.

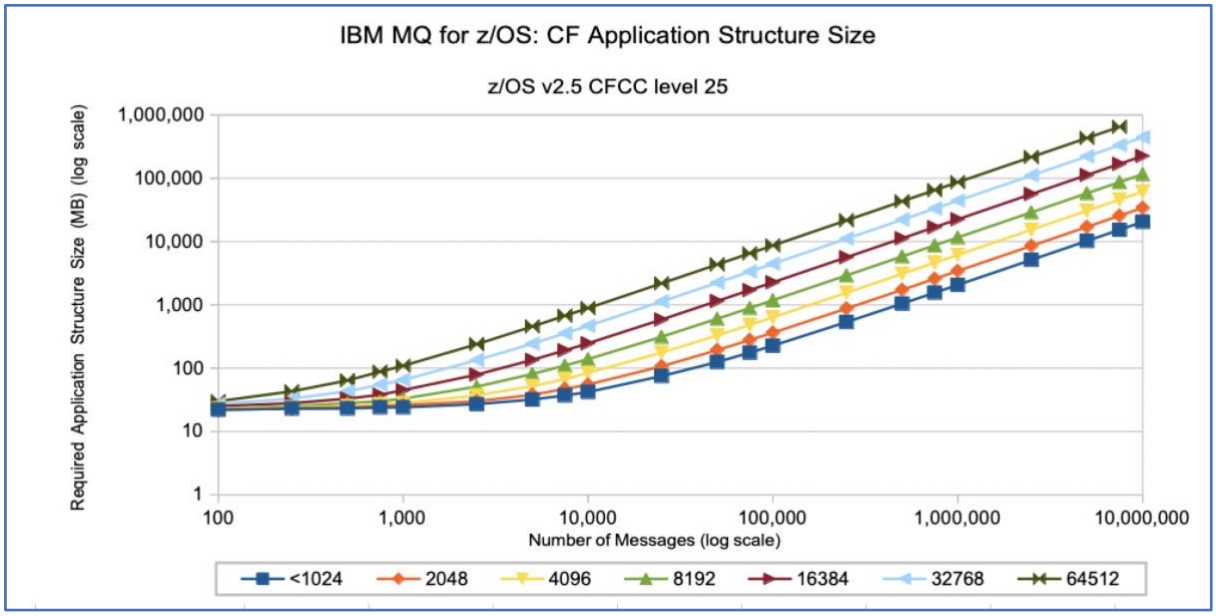


Does CFCC level 25 affect the size of my APPLICATION structures?

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 17-25** 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 affect the size of my SYSAPPL structure?

Yes.

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 9.3 suggests an INITSIZE of 20,000.

Subsequent releases of MQ will increase this value to a minimum of 50,000.

How do CF response times compare?

Early in 2022, I wrote a blog discussing CF statistics reported by MQ for z/OS. The MQ task records, as generated when enabling Accounting trace class(3), include CF request types such as *StartMon, StopMon, Move, ReadList* etc.

When comparing shared queue MQ performance benchmarks run on both IBM z15 and IBM z16, it has been observed on our system that most of the CF request types are taking slightly longer on the newer hardware and CFCC micro-code, of the order of 1 microsecond per CF request type.

This increase is despite the Coupling Facility being internal and configured with the same number and type of ICP links and dedicated processors.

Since an MQ request may consist of multiple CF request types, this can become an impact on response times made to the CF.

For example, an "MQGET-with-wait for a specific message" might make CF requests consisting of: *StartMon, StopMon, ReadList, Move* and *Delete.* An increase of 1 microsecond per CF request type could equate to an additional 5 microseconds *elapsed* time for an MQGET of this nature.

Coupling Facility - CFCC Level 24

Whilst CFCC level 24 was originally delivered on IBM z15 with driver level 41, it is worth mentioning a feature which might affect MQ for z/OS on both IBM z15 and IBM z16.

CF monopolization avoidance

With CFCC level 24 on IBM z15 onwards, z/OS will monitor in real-time the number of CF tasks that have a command assigned to them for a given structure, on a structure-by-structure basis.

When the number of CF tasks being used by any given structure exceeds a model-dependent CF threshold, and a global threshold on the number of active tasks is also exceeded, the structure will be "monopolizing" the CF, and z/OS will be informed of this monopolization.

New support in z/OS will observe the monopolization state for a structure and start to selectively queue and throttle incoming requests to the CF, on a structure-specific basis – while other requests, for other "non-monopolizing" structures and workloads, are completely unaffected.

z/OS will dynamically manage the queue of requests for the "monopolizing" structures to limit the number of active CF requests (parallelism) to them and will monitor the CF's monopolization state information to observe the structure becoming "non-monopolized" again, so that request processing can eventually revert to a non-throttled mode of operation.

The overall goal of z/OS anti-monopolization support is to protect the ability of ALL well-behaved structures and workloads that access the CF, and get their requests processed in the CF in a timely fashion, while implementing queuing and throttling mechanisms in z/OS to hold back the specific abusive workloads that are causing problems for other workloads.

This isn't new, so why is this relevant on IBM z16?

Whilst this is not IBM z16 specific, with z/OS v2r5, the z/OS MVS Setting up a Sysplex "Summary of Changes" documentation states "the default for CFMONOPAVOID function is changed to Enabled in the XCF/XES Optional Functions table of the FUNCTIONS statement".

How do I know if CF monopolization avoidance has been initiated?

There are several ways to determine if CF monopolization has been applied to a structure:

1. The system log will show message `IXL062E` when entering and `IXL063I` when exiting CF monopolizing avoidance. If the CF monopolization continues for an extended period, message `IXL064I` will be periodically logged.
2. The z/OS RMF Coupling Facility Activity Report – in particular the Coupling Facility Structure Activity section includes a count of delayed requests with the `MONOP` reason.

3 Improvements from Crypto Express 8S

As mentioned in the previous section, the IBM z16 benefits from some significant improvements in the cryptographic area – primarily in Crypto Express 8S (CEX8S).

This section details the performance improvements observed in our MQ performance tests for the following classes of tests:

- Channels protected with TLS.
- Queues protected using AMS policies.

The comparisons are between:

- IBM z15 with Crypto Express 7S (CEX7S)
- IBM z16 with Crypto Express 8S (CEX8S)

Channels protected with TLS

For MQ channels that are protected with TLS, we have seen no significant benefit for CEX8S over CEX7S response times for typical requests relating to the starting of MQ channels or the secret key re-negotiation driven by the reaching of the thresholds defined by the `SSLRKEYC` attribute.

The benefits of reduced channel start costs and reduced time to start an MQ channel that is protected by TLS, have resulted from the performance improvements to general purpose processors and CPACF.

Queues protected using AMS policies

For MQ queues protected with AMS Policies that rely on Crypto Express function, particularly AMS Privacy and AMS Integrity we have seen no significant benefit for CEX8S over CEX7S response times.

AMS performance for all policy types, namely Integrity, Privacy and Confidentiality have all benefitted from the performance improvements to general purpose processors and CPACF.

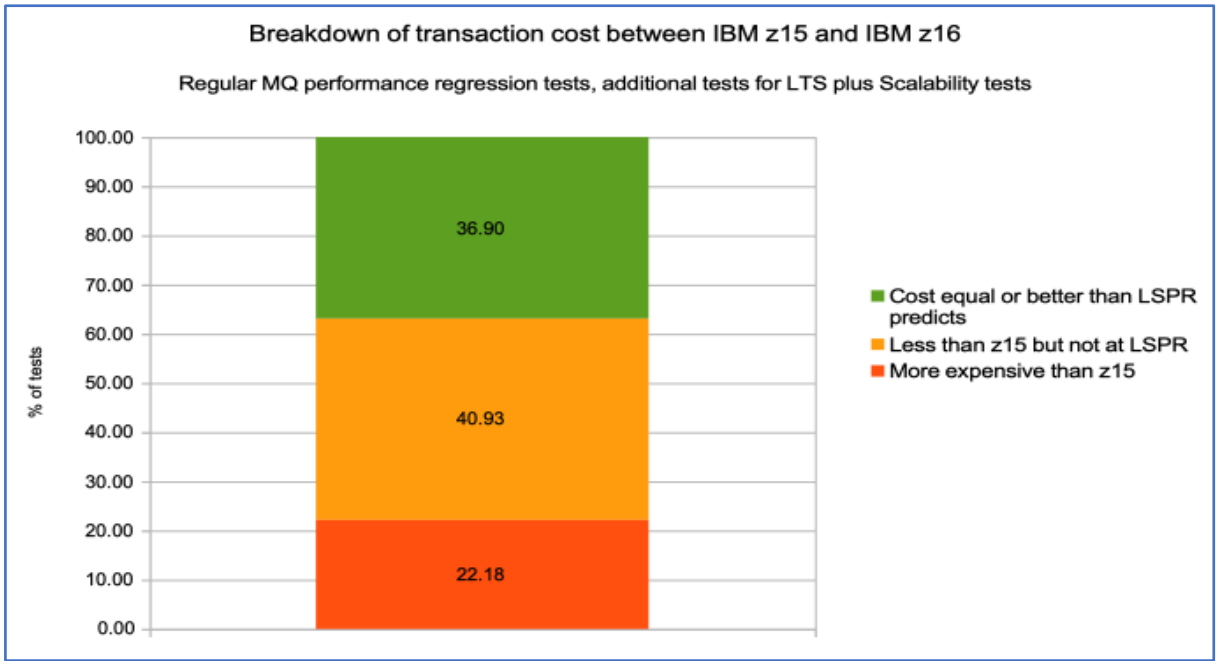
4 General test performance and scalability

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 these test results are included in the MQ for z/OS 9.3 performance report in "Appendix A – Regression".

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

For this document, we initially moved the MQ performance system en masse to IBM z16 and run all the micro-benchmarks mentioned, and these are run in the same configurations as on IBM z15. Analysis of these measurements saw a spread of results as summarised in the following chart:

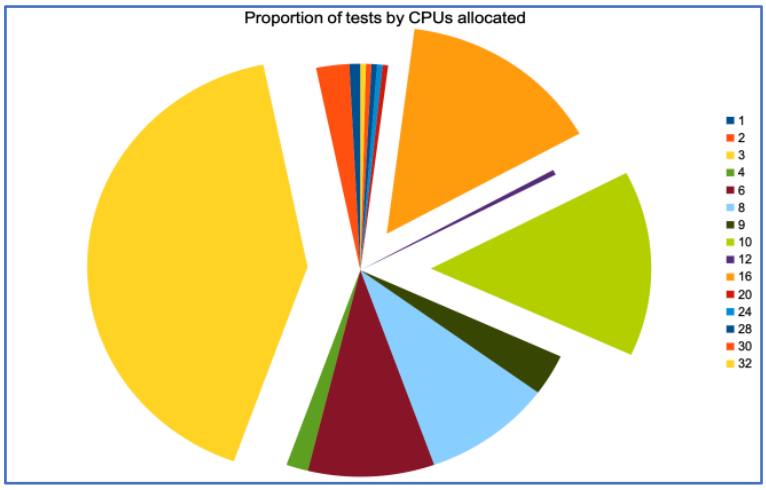


- 36.9% of micro-benchmarks have reduced cost by the expectations set by LSPR or better.
- 77.8% of micro-benchmarks have lower costs than the equivalent measurement run on IBM z15.

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

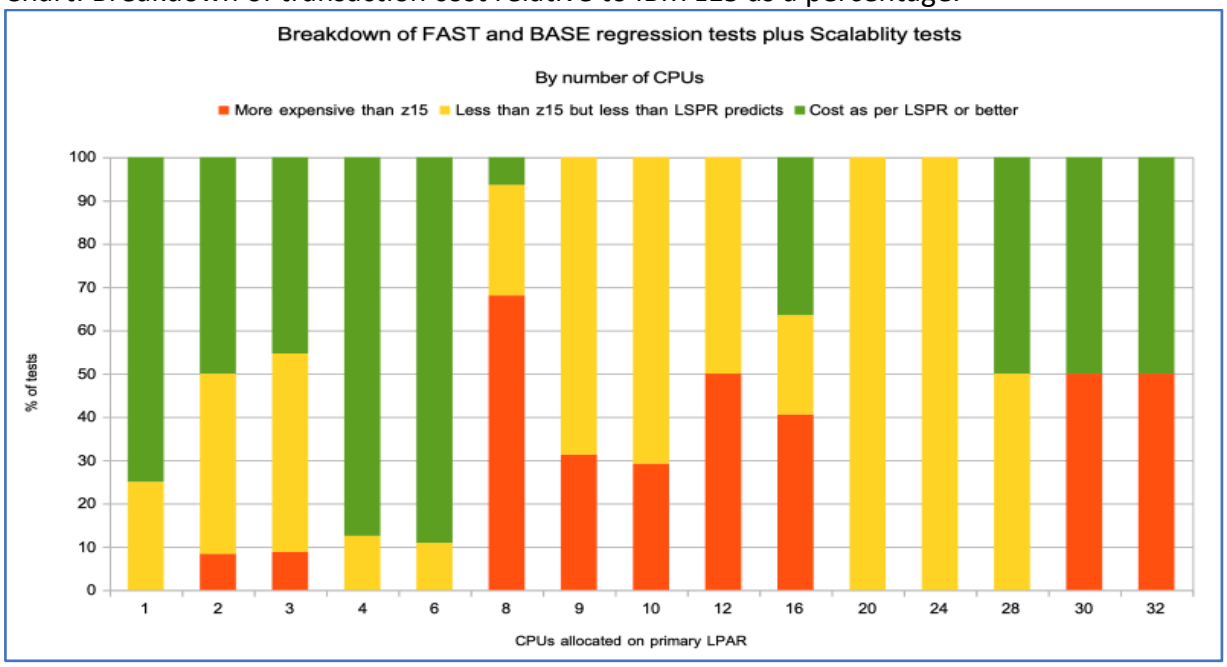
1. Proportion of tests run on a particular CPU allocation.
2. Breakdown of transaction cost relative to IBM z15 as a percentage, by CPU allocation.

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



The preceding chart shows that 70% of the tests run are run on LPARs with 3, 10 or 16 CPUs, with the remaining 12 CPU configurations accounting for just 30% of the tests run.

Chart: Breakdown of transaction cost relative to IBM z15 as a percentage.



If we concentrate on the tests showing *costs that are higher on IBM z16 than z15*, there are 3 main groupings:

- **2-3 CPUs:** Simple tests (very short pathlength, small message) are the primary areas of increased cost. In some cases where the CF response times are increased, the workload can be less "cache-friendly" and subsequently an increase in MQ API cost occurs.
- **8-16 CPUs:** This is primarily where on IBM z16 the workload spans 2 then 3 PU's, whereas IBM z15 can contain the workload on 1 or 2 PU's. Note that the 12 CPU measurements that failed are further examples of very short pathlength transactions.
- **30+ CPUs:** Failures are in 'non-persistent out-of-syncpoint' workloads, are further examples of very short pathlength transactions.

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.

- Persistent message tests demonstrated cost reductions of up to 28%.

- Channels using compression
  Hardware compression using COMPMSG(ZLIBFAST) saw a reduction in transaction cost of up to 15% and increase in throughput of up to 10%.
  Software compression using COMPMSG(ZLIBHIGH) saw a reduction in transaction cost of up to 6% and increase in throughput of up to 5%.

- *Channels protected with TLS ciphers (`SSLCIPH`)*
  TLS 1.2 ciphers - On simple request/reply workloads we saw channel throughput increase up to 5%.
  TLS 1.3 ciphers - On simple request/reply workloads we saw channel throughput increase up to 10% with a cost reduction of up to 14%.

  Performance gains are dependent on the cipher used to protect the workload and the frequency of secret key negotiation.

- Queues protected with AMS policies
  Transaction costs were up to 18% lower on IBM z16, with an increase in transaction rate of up to 46% with larger messages.

- Scalability tests achieving up to 11% higher throughput
  1. Non-persistent in-syncpoint workload achieved more than 0.6 million messages per second on a single z/OS LPAR with 28 or more CPUs, an increase of 11% over IBM z15.
  2. Non-persistent out-of-syncpoint workload achieved more than 1.3 million messages per second on single z/OS LPAR with 24 or more CPUs. This is a *decrease of nearly 20%* on the equivalent test on IBM z15.

Performance of MQ persistent message benchmarks

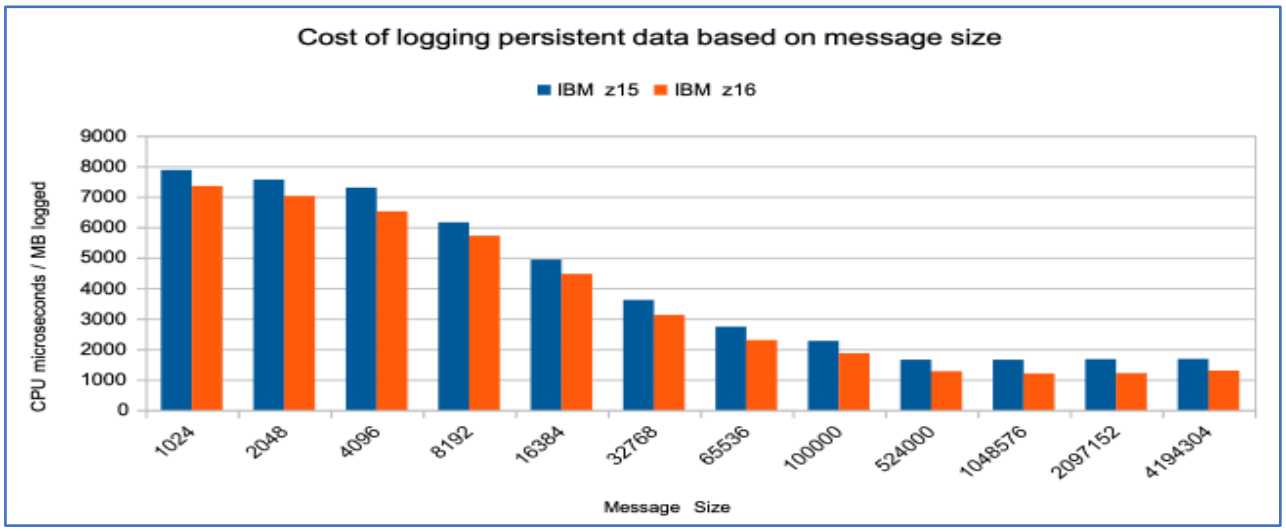
In the MQ for z/OS 9.3 performance report, the section reporting the performance of regression tests included a chart showing the "upper bounds of persistent logging rate", i.e. the maximum logging rate that MQ was able to sustain in our test environment.

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 380MB 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 z16 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.

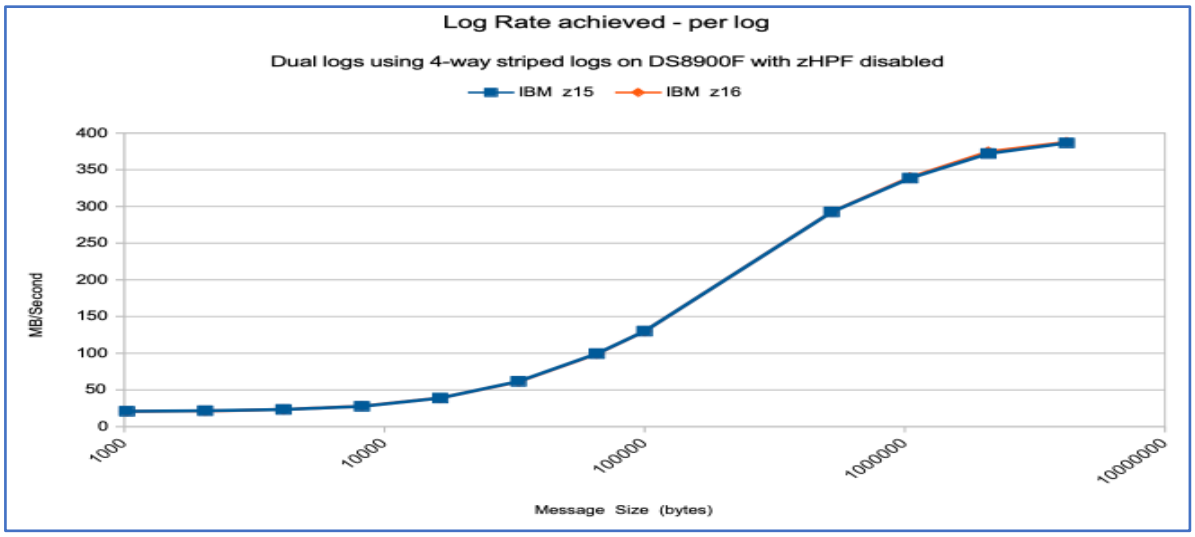
The following chart shows a comparison of the cost per MB logged for this workload.



With regards to the cost of logging the persistent messages, on IBM z16 the costs are reduced by up to 28% over the equivalent test on IBM z15 where the cost reductions can be grouped:

- Small messages (up to 16KB), the costs were reduced by up to 10%.
- Medium size messages (16KB to 100KB), costs were reduced by up to 17%
- Larger messages (up to 4MB), costs were reduced by up to 28%

The second chart shows that, as expected, the log rate achieved on IBM z16 is like that achieved on IBM z15:



Performance of MQ channels protected with TLS

When performance testing MQ channels protected with cipher specifications, we separate into 2 categories – TLS 1.2 ciphers and TLS 1.3 ciphers.

For TLS 1.2 ciphers we concentrate on the following 4 ciphers:

1. `TLS_RSA_WITH_AES_256_CBC_SHA256`
2. `TLS_RSA_WITH_AES_128_GCM_SHA256`
3. `ECDHE_RSA_AES_256_CBC_SHA384`
4. `ECDHE_ECDSA_AES_256_CBC_SHA384`

For these ciphers, we run in 2 modes: frequent secret key negotiation and no secret key re-negotiation.

The cost of the secret key negotiation is largely in the MQ channel initiator address space, but some of the cost can be offloaded to the Crypto Express hardware.

For TLS 1.3 ciphers, all 3 currently supported ciphers are benchmarked:

1. `TLS_AES_128_GCM_SHA256`
2. `TLS_AES_256_GCM_SHA384`
3. `TLS_CHACHA20_POLY1305_SHA256`

As discussed in the MQ for z/OS 9.2 performance report, TLS 1.3 ciphers have re-negotiation as part of the protocol, so the processing triggered by reaching the threshold as defined by the SSLRKEYC attribute does not perform key re-negotiation. As such, the TLS 1.3 cipher benchmarks only run using the “no secret key re-negotiation” model.

As mentioned earlier, we have found that Crypto Express 8S “CEX8S” offers no significant benefit to performance for MQ channels when compared to Crypto Express 7S.

In our measurements, the TLS prefixed ciphers were able to exploit both coprocessor and accelerator, unlike the ECDHE prefixed ciphers that were only able to use coprocessor.

TLS 1.2 ciphers

In terms of MQ channel address space cost, the `TLS_RSA` prefixed ciphers are considerably lower cost at the time of secret key negotiation.

Cost of secret key negotiation (CPU milliseconds) in MQ channel initiator address space:

Cipher	z15	z16
TLS_RSA_WITH_AES_256_CBC_SHA256	0.86	0.43
TLS_RSA_WITH_AES_128_GCM_SHA256	0.88	0.37
ECDHE_RSA_AES_256_CBC_SHA384	4.37	3.9
ECDHE_ECDSA_AES_256_CBC_SHA384	4.4	3

When running with no secret key re-negotiation outside of channel start, the 4 named ciphers deliver comparable performance at similar cost on z15.

For the purposes of this section, we compare performance results using all these named cipher specifications using non-persistent messages of 32KB. The measurements use a request/reply workload between 2 queue managers on separate LPARs, that are connected by a 10Gb low-latency link.

For simplicity and clarity, we are only showing the performance with a single pair of inbound and outbound channels.

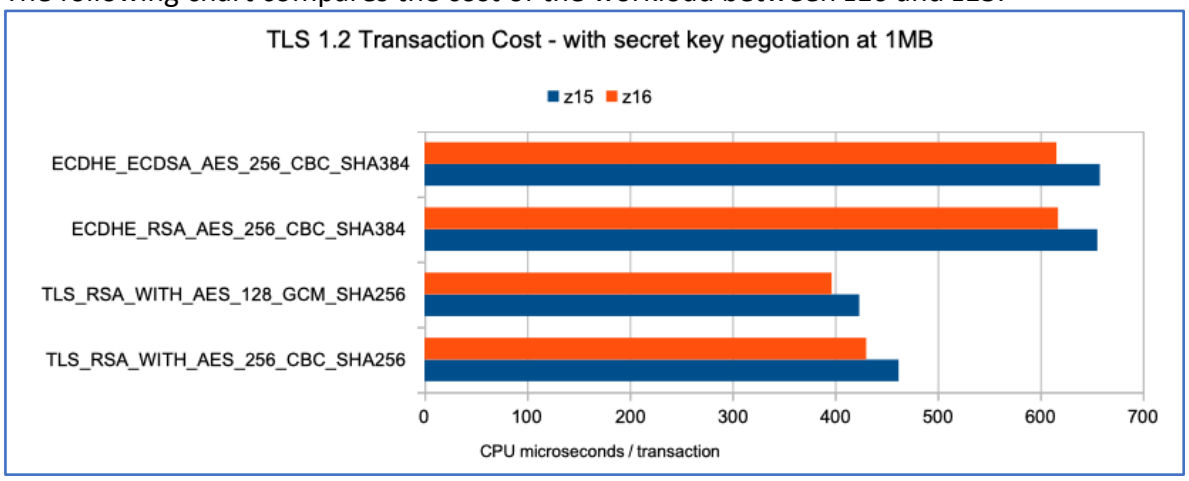
As mentioned previously, the measurements are run in 2 configurations:

1. Frequent negotiation of secret key:
    a. Negotiate the secret key every 1MB that passes over the channel, by setting `SSLRKEYC(1048576)`
    b. Demonstrates the benefits of Crypto Express features and CPACF.
2. No renegotiation of secret key:
    a. Negotiate the secret key at channel start only – with the channels remaining active for the duration of the workload.
    b. Demonstrates the benefits of CPACF only.

Frequent re-negotiation of secret key

When negotiating the secret key, MQ can offload a significant proportion of the processing to the Crypto Express feature. For data encryption, the encryption and decryption are processed by CPACF.

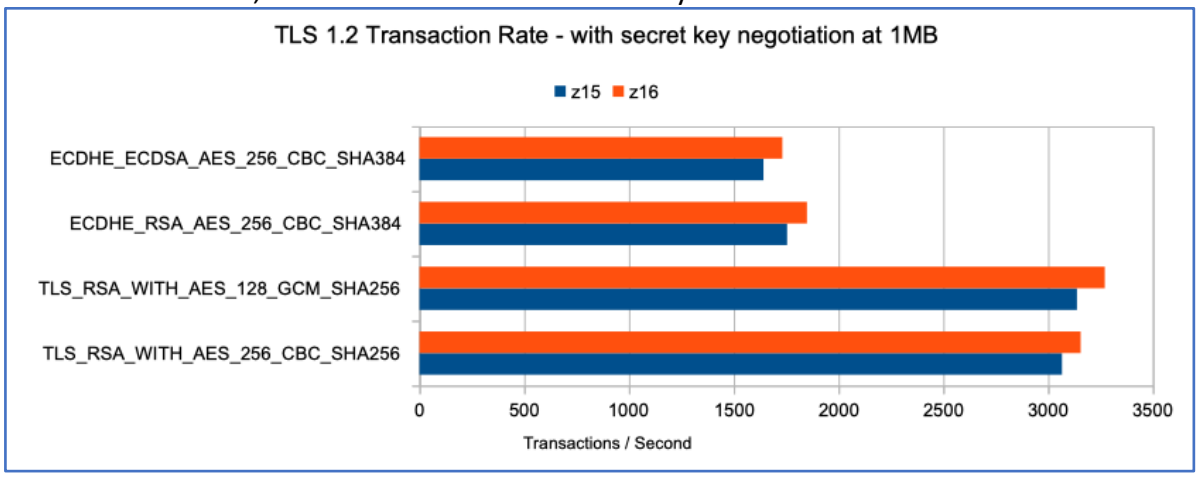
The following chart compares the cost of the workload between z16 and z15.



The transaction cost chart shows that in our measurements, that z16 reduced the cost of the workload as below:

`TLS_RSA` prefixed ciphers: 6.5% lower on z16.
`ECDHE` prefixed ciphers: 6.2% lower on z16.

In the second chart, the transaction rate achieved by the workloads is shown.



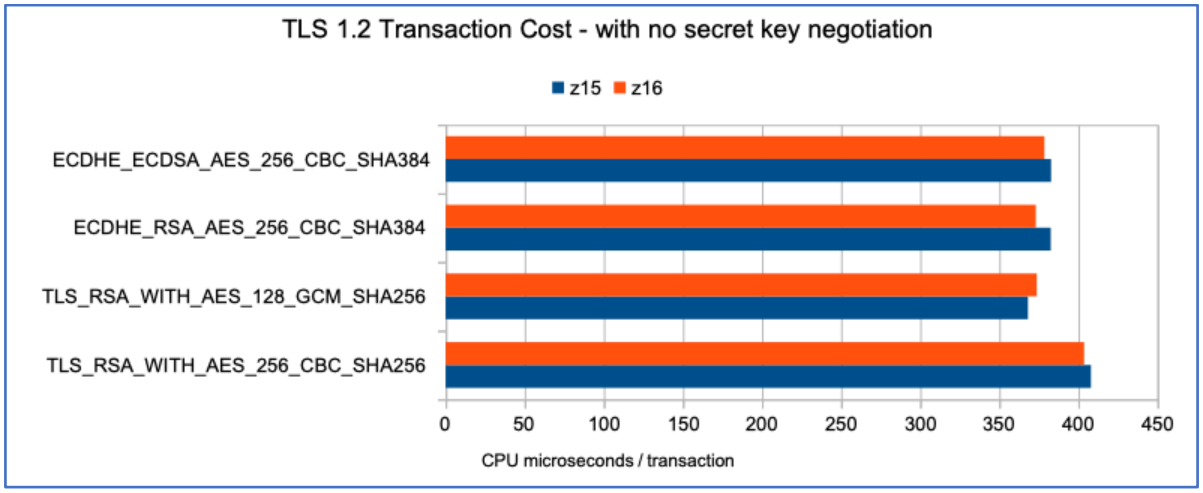
Across the workloads, we saw improvements in transaction rates over the equivalent z15 measurements, as detailed below:

`TLS_RSA` prefixed ciphers: 3% higher on z16.
`ECDHE` prefixed ciphers: 5% higher on z16.

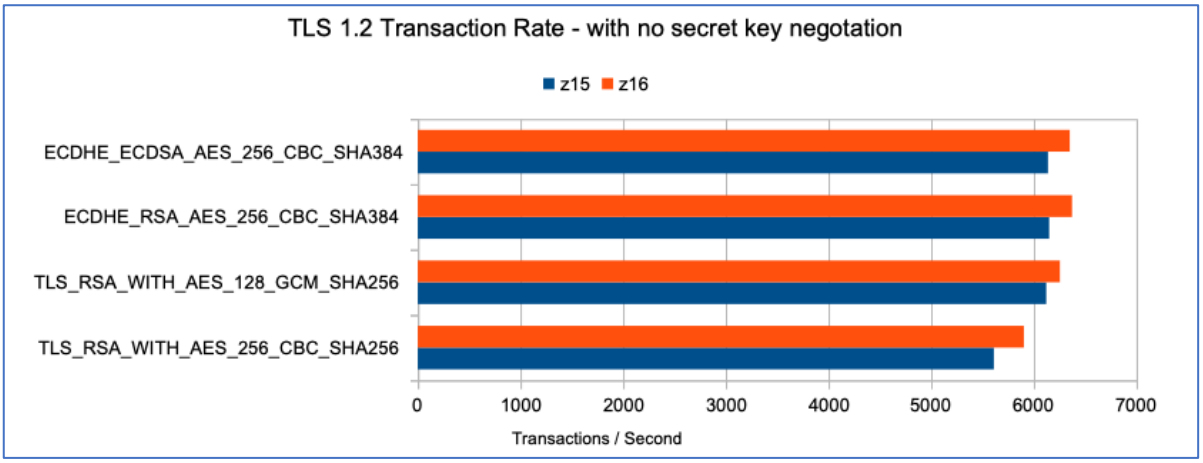
No renegotiation of secret key

By negotiating the secret key only at channel start, these measurements are aimed at demonstrating the improved performance of encryption/decryption services.

The following chart compares the cost of the workload between z15 and z16. As there is no secret key negotiation involved in the measurement, the level of Crypto Express is irrelevant for the purposes of this measurement. The chart demonstrates that IBM z16 shows similar transaction cost to that measured on IBM z15.



The second chart shows the transaction rate achieved when the secret key is negotiated at channel start. The transaction rate is up to 5% higher on z16.



TLS 1.3 ciphers

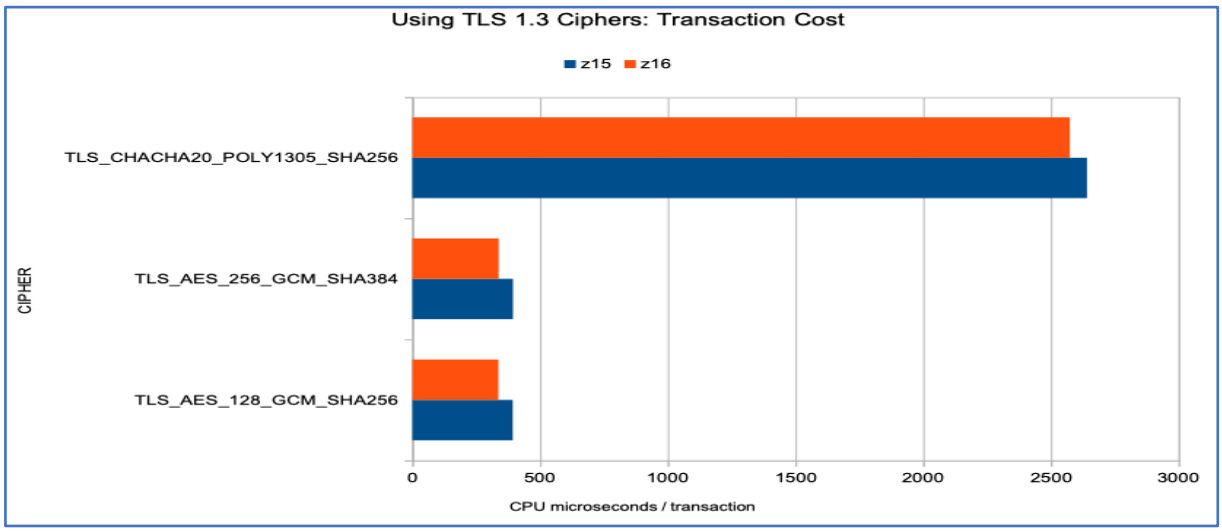
For the purposes of this section, we compare performance results using all currently supported TLS 1.3 ciphers using non-persistent messages of 32KB. The measurements use a request/reply workload between 2 queue managers on separate LPARs, that are connected by a 10Gb low-latency link.

For simplicity and clarity, we are only showing the performance with a single pair of inbound and outbound channels.

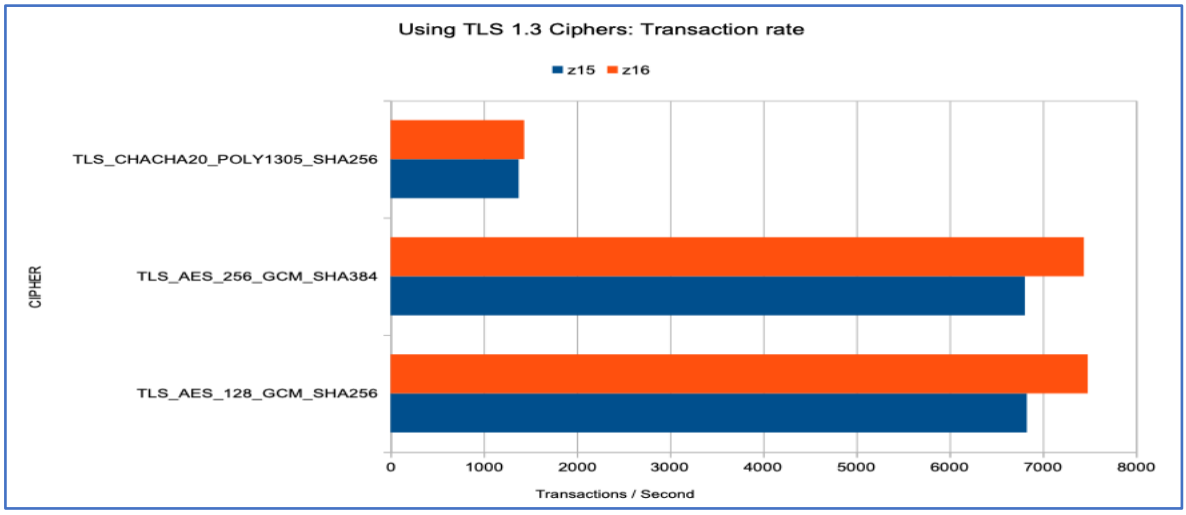
As mentioned previously, the measurements are run in a single configuration i.e.:

1. No renegotiation of secret key:
    a. Negotiate the secret key at channel start only – with the channels remaining active for the duration of the workload.
    b. Demonstrates the benefits of CPACF only.

The following chart compares the cost of the workload between z15 and z16. As there is no secret key negotiation involved in the measurement, the level of Crypto Express is irrelevant for the purposes of this measurement.
The chart demonstrates that IBM z16 shows a decrease in transaction cost of up to 14% when compared to the same transaction on IBM z15.



The second chart shows the transaction rate achieved when the secret key is negotiated at channel start. The transaction rate is up to 10% higher on IBM z16.



In the transaction rate comparison for TLS 1.3 ciphers, `TLS_AES` prefixed ciphers improved by 10%, whereas `TLS_CHACHA20` ciphers achieved a 5% improvement in throughput.

TLS channel start costs

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

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

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

Whenever an TLS-enabled channel pair[1] is started, a cryptographic handshake is performed which establishes the authenticity of the channel partners and dynamically generates a secret cryptographic encryption key. This cryptographic handshake increases both the CPU consumption and the elapsed time for the channel start.

On our IBM z16 system configured with 3 dedicated processors, we have found the additional TLS costs to be somewhat dependent of the cipher specification used. With 4000 channel pairs in SYSTEM.CHANNEL.SYNCQ:

Cipher	Channels started per second	CPU cost milliseconds / channel
TLS_RSA_WITH_AES_256_CBC_SHA256	150	1.97
ECDHE_RSA_AES_256_CBC_SHA384	95	3.43
ECDHE_ECDSA_AES_256_CBC_SHA384	88	3.39

For the TLS_RSA_WITH_AES_256_CBC_SHA256 cipher, this represents an 13% improvement over the equivalent z15 measurement.

For the ECDHE prefixed ciphers, this represents a 14% improvement over the equivalent z15 measurement.

Note that channel start costs of TLS-enabled channels are significantly higher than the costs incurred at time of secret key negotiation.

For completeness, three charts are provided:

1. Comparison of channel start rates with all TLS 1.2 and TLS 1.3 cipher specifications currently supported on MQ for z/OS.
2. Comparison of channel start costs for all TLS 1.2 and TLS 1.3 cipher specifications currently supported on MQ for z/OS.

[1] In this example, a channel pair is one CHLTYPE(SDR) and one CHLTYPE(RCVR).

3. Average execution time used in CryptoExpress8S when starting the MQ channels with TLS 1.2 and TLS 1.3 ciphers. As discussed in the "Improvements from Crypto Express 8S" section, we found the performance of the CEX8S card to be like that of CEX7S for MQ channels.

Chart: Comparison of TLS-protected channel start rates

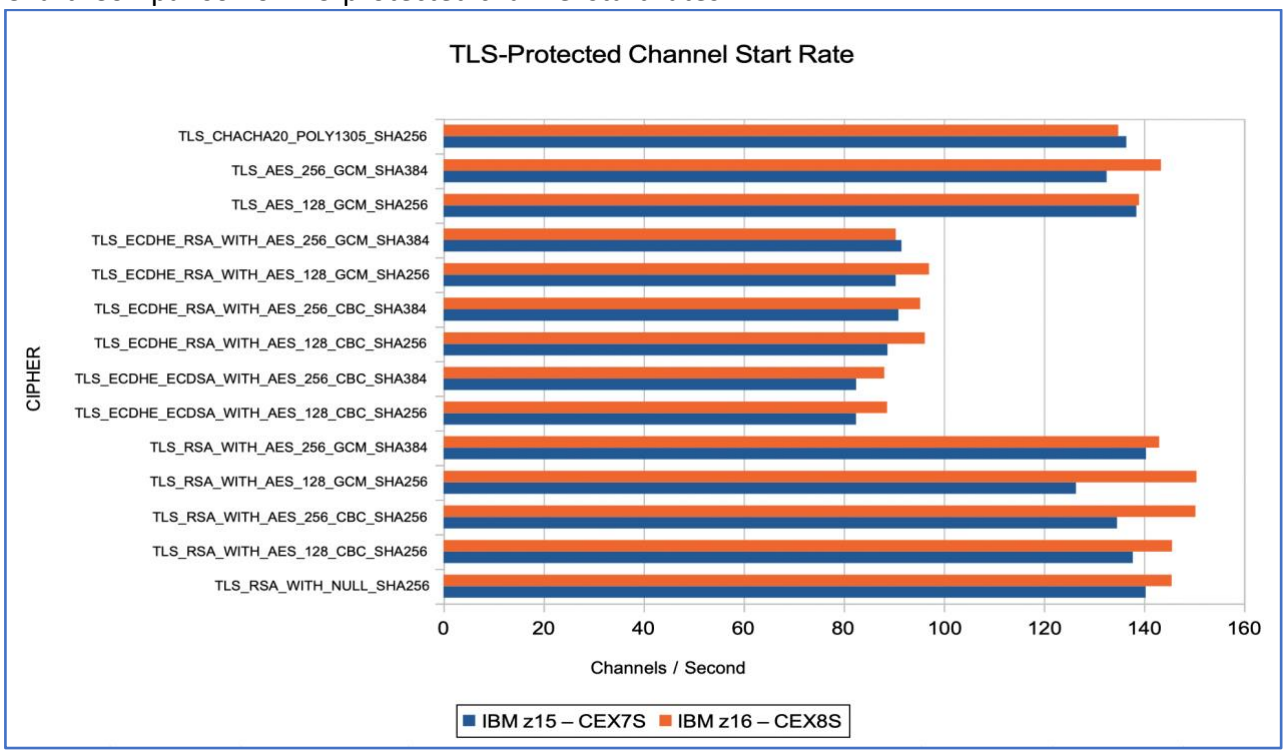


Chart: Comparison of channel start costs for all TLS 1.2 and TLS 1.3 cipher specification

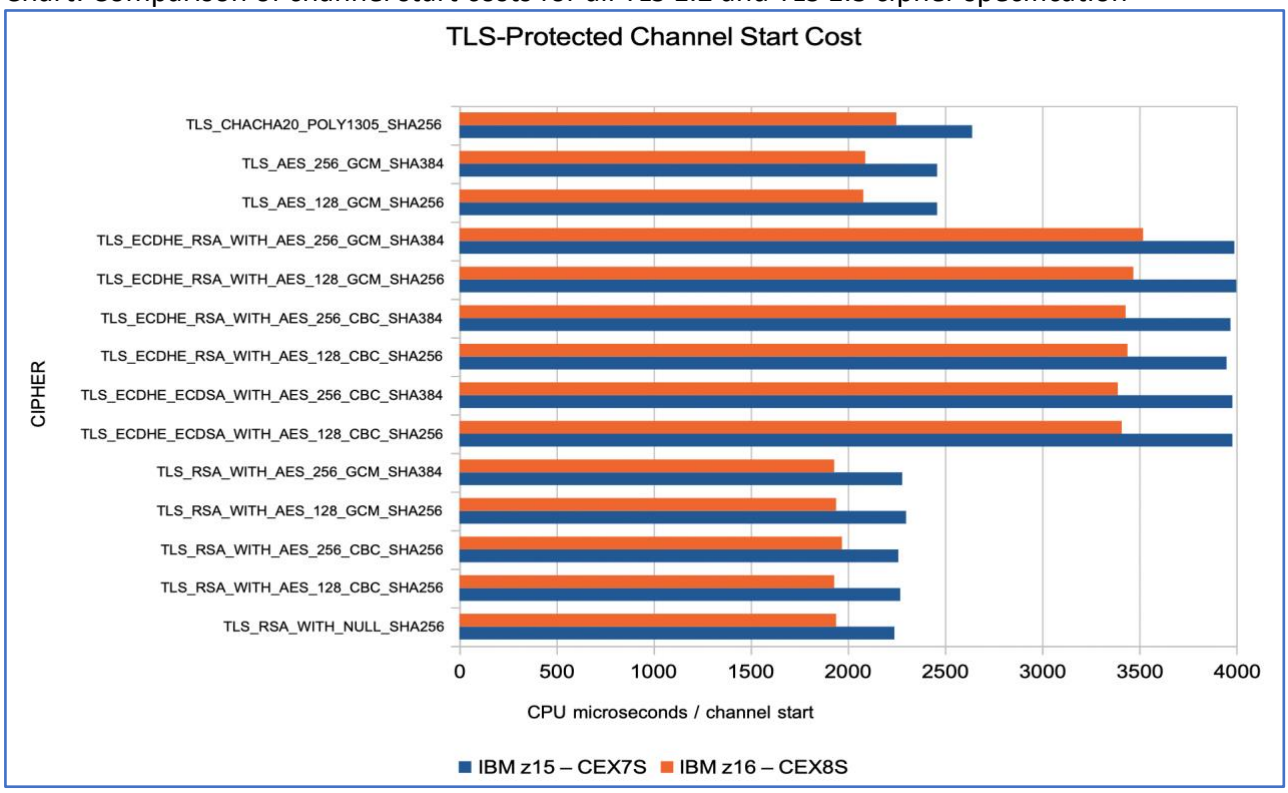
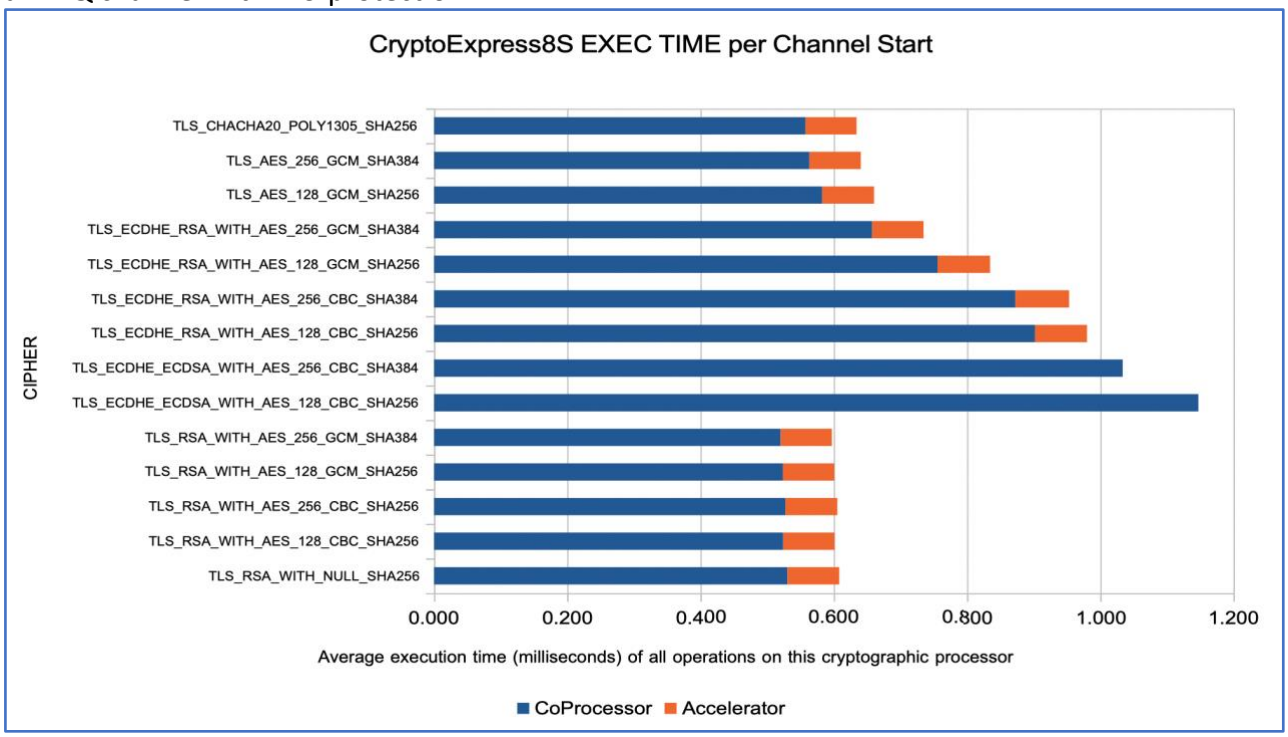


Chart: Average Execution time as reported by the RMF Cryptographic report when starting an MQ channel with TLS-protection



With regards to the average execution time, the RMF cryptographic report provides the average execution time of each call to the CEX8S. To add value to this, we have calculated

the number of calls to the CryptoExpress features as required by MQ for z/OS to start each of the ciphers.

The number of calls to the CryptoExpress features does vary, depending on the cipher used. For example, using `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384` resulted in 3 calls to the Coprocessor and no calls to the accelerator, whereas `TLS_CHACHA20_POLY1305_SHA256` had 1 call to the Coprocessor and 2 calls to the Accelerator.

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 queues where:

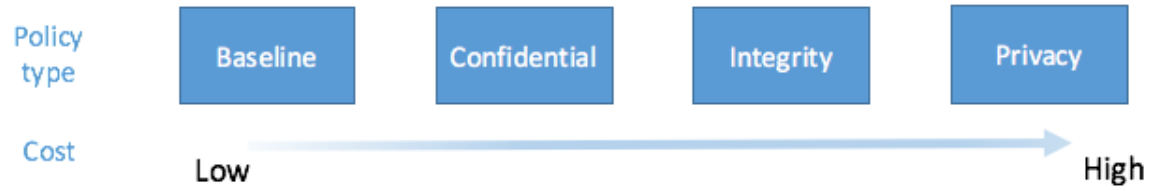
- Integrity used messages signed with SHA256.
- Privacy used message signed with SHA256 and encrypted using AES256.
- Confidentiality used messages encrypted using AES256 and the key reused 32 times.

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 z15 to z16. 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 z15 and z16.

% Change in transaction cost from z15 to z16:

Message Size	Integrity	Privacy	Confidentiality
Small	-8.5	-6.8	-4
Medium	-15.8	-14.7	-18.3
Large	-15.5	-15.7	-10.2

% Change in throughput from z15 to z16:

Message Size	Integrity	Privacy	Confidentiality
Small	+1.6	+0.6	+3.3
Medium	+8	+6.5	+20.3
Large	+44.9	+40.5	+46.9

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

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.

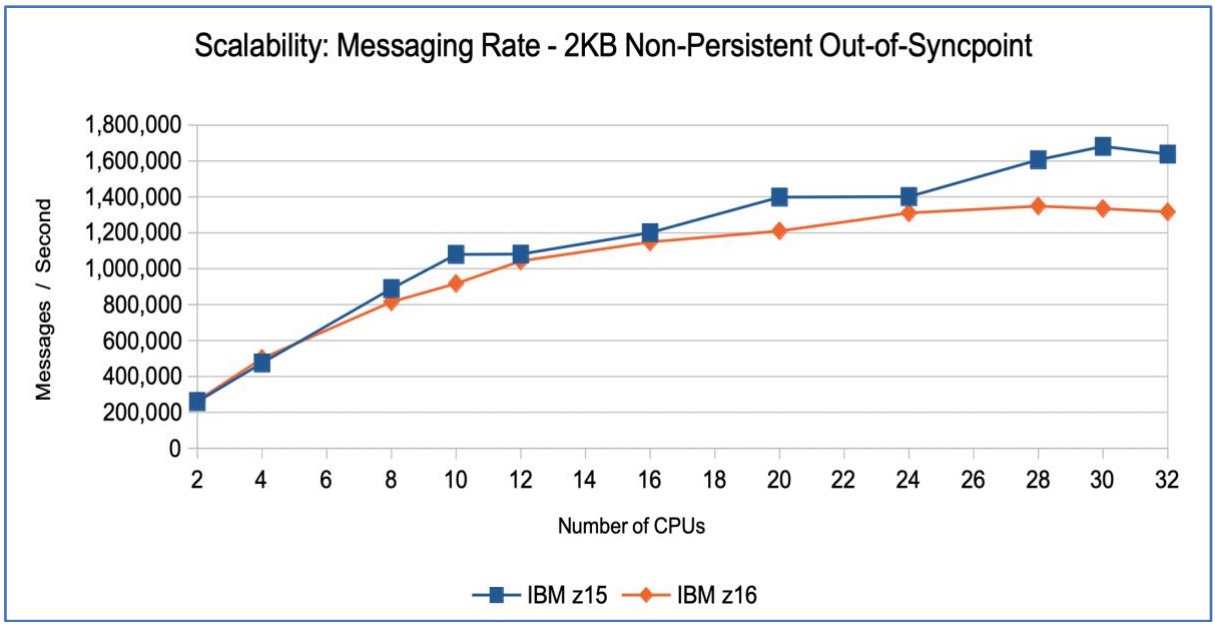
The requester puts a message and waits for a specific reply. When the requester gets that message, it generates a new request message, and this continues until the applications are requested to end.

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 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 1,638,600 messages per second, was achieved with 32 CPUs. This equates to 840,740 transactions per second on a single MQ queue manager.

On IBM z16, the peak throughput of 1,348,870 messages per second, was achieved with 28 CPUs. This equates to 674,435 transactions per second on a single MQ queue manager and is a *decrease of nearly 20%* in maximum throughput.

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	1.64 million *(1.52 million at GA)*
IBM z16	1.35 million

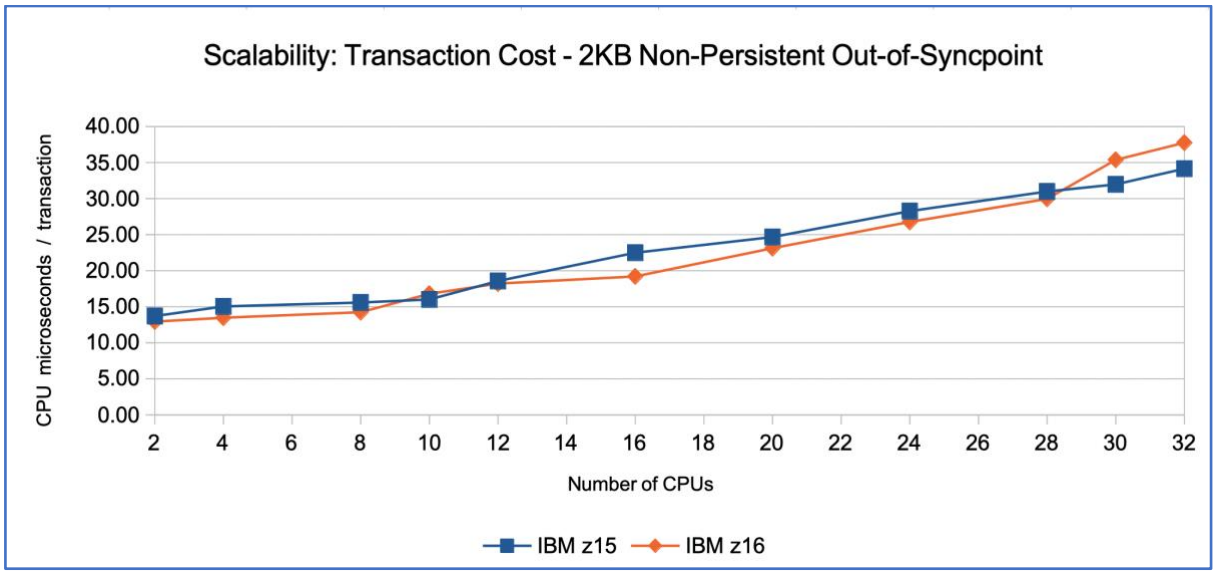
Why the decrease in peak transaction rate?

At least part of the difference in performance degradation between IBM z15 and IBM z16, 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 1.7 million messages per second, again with 28 CPUs allocated. However, it has not been determined how much impact this same action would have on the equivalent IBM z15 measurement.

The following chart shows the cost of a transaction when the workload is running at peak throughput, with class(3) accounting trace enabled on both IBM z15 and IBM z16:

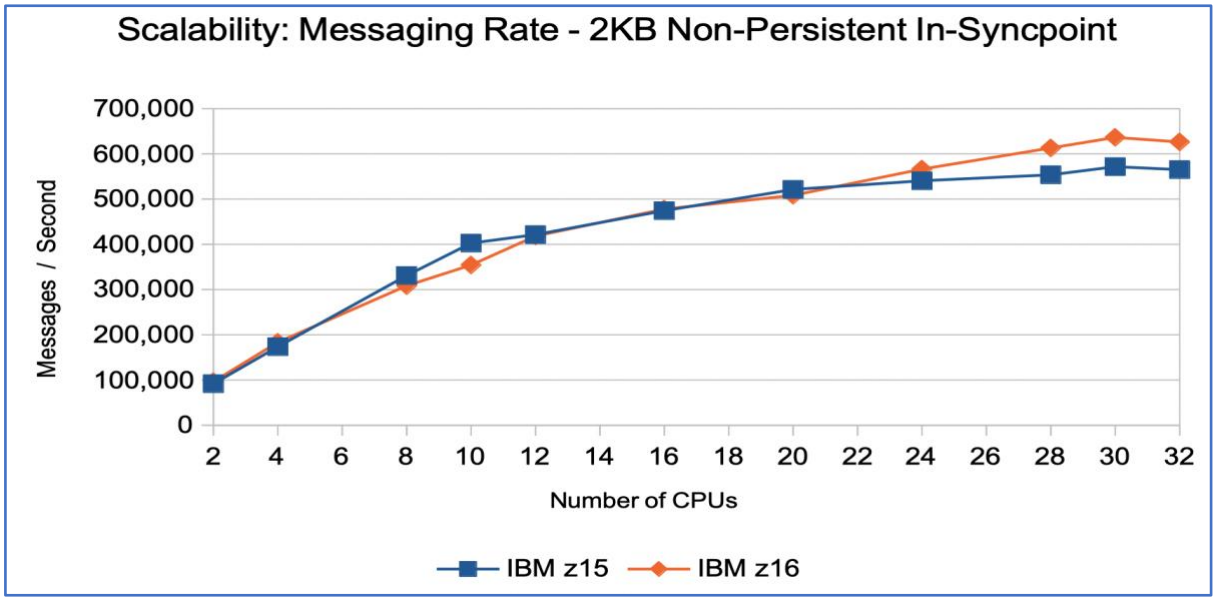


On IBM z16 up to 28 CPUs, the average cost per transaction when running at peak transaction rate is on average 5.7% lower than the equivalent on IBM z15.

When 30-32 CPUs are allocated, the cost per transaction on IBM z16 was determined to be 10% higher than the equivalent on IBM z15.

Non-persistent in-syncpoint using 2KB messages

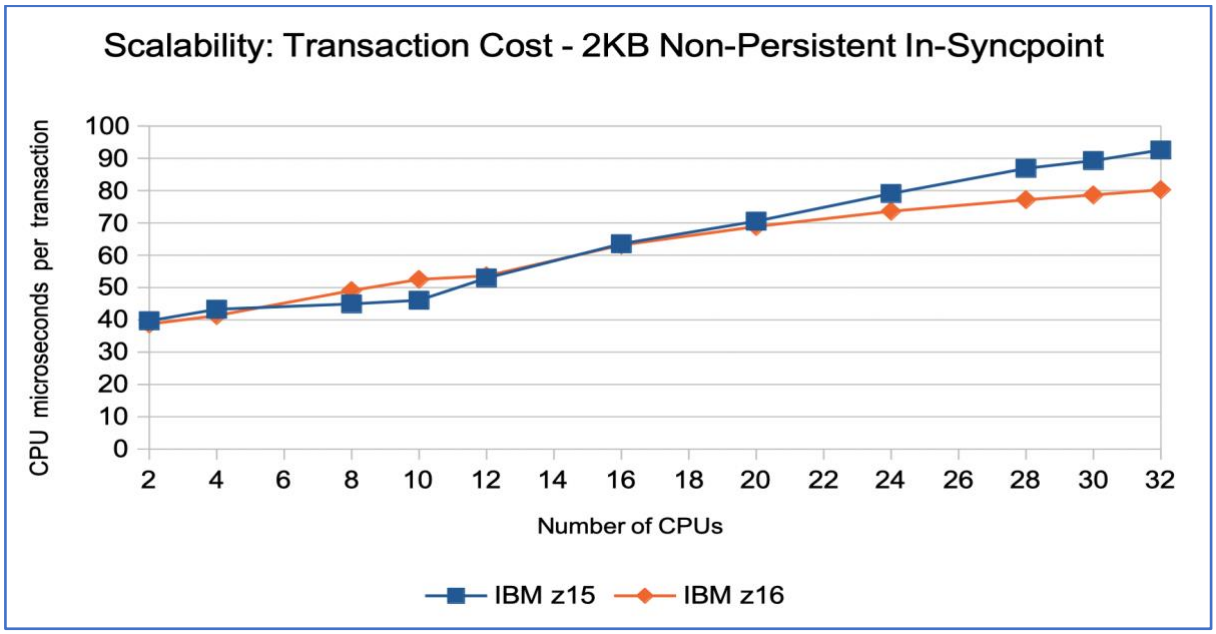
The following chart shows the peak transaction rate achieved when the number of CPUs is increased.



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 636,460 messages per second, or 318,230 transactions per second on a single MQ queue manager and is an *increase of 11.3%* in maximum throughput.

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



Cost per transaction on IBM z16 for the 2KB non-persistent in-syncpoint workload are generally lower than the equivalent test on IBM z15. At 8-10 CPUs the transaction cost is 9-14% higher on IBM z16 which can be attributed to the workload having to span multiple processor units (PU), unlike on IBM z15 which is able to allocate the CPUs on a single PU.

At the higher n-way configurations e.g., 24 processors and higher, the IBM z16 shows a decrease in transaction cost of the order of 7-13%.

Appendix A – Test Environment

Measurements were performed using:

The IBM MQ performance sysplex ran measurements on:

- **IBM z15 (8561-7J0) – 4 CPC drawers**
- **IBM z16 (3931-7K0) – 4 CPC drawers (Max200)**

The sysplex was configured thus:

- LPAR 1:
  - 1-32 dedicated CP plus 2 zIIP with 144 GB of real storage.
- LPAR 2:
  - 1-10 dedicated CP plus 2 zIIP with 48 GB of real storage.
- LPAR 3:
  - 1-3 dedicated CP with 48 GB of real storage.
- z/OS v2r5.
- Db2 for z/OS version 12 configured for MQ using Universal Table spaces.
- IMS 15.1
- IBM CICS CTS 6.2
- MQ queue managers:
  - configured at MQ 9.3.
  - 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 16S connected DS8900F
- 4 dedicated channel paths
- HYPERPAV enabled
- zHPF disabled unless otherwise specified.

Network:

- 10GbE network configured with minimal hops to distributed partner machines
- 1GbE network available

Applications written in a mixture of:

- C
- COBOL compiled with Enterprise COBOL for z/OS 6.3 with options ARCH(13) and OPT(1).