IBM z15 Hardware Compression Exploitation

z/TPF MQ and z/TPF HTTP Server

Jamie Farmer

IBM **Z**

Disclaimer

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

z/TPF MQ Internal Compression

Problem Statement

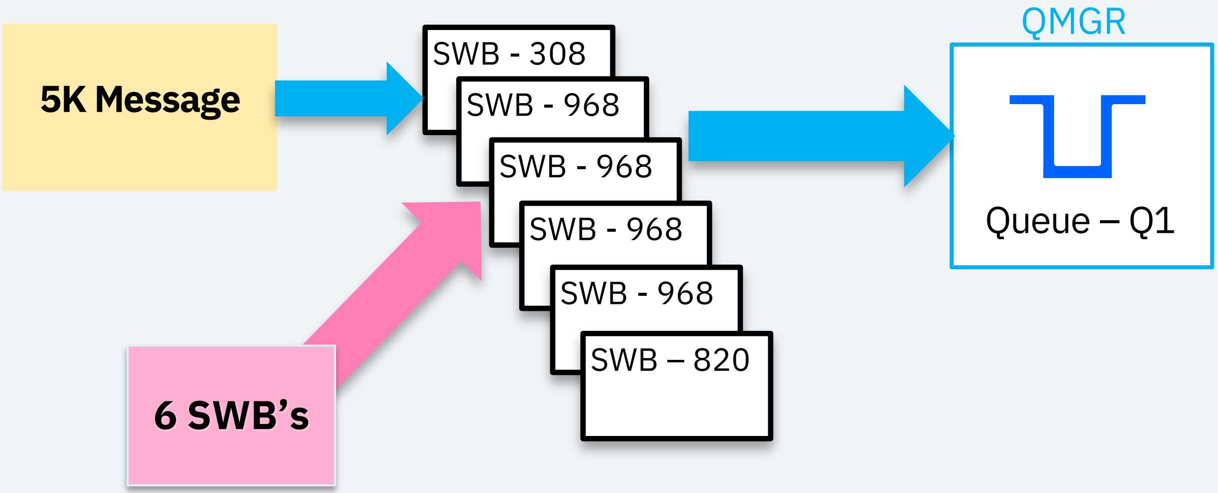
- z/TPF MQ uses 31-bit memory to save MQ messages on local queues.
  - Uses system work blocks (SWBs)
- Available space below the 2G bar can prevent future growth of the z/TPF system
  - For example, increasing the number of defined ECBs in the system
- With limited number of SWBs, more frequent I/O may result as the MQ sweeper needs to free up SWB storage
  - Small hiccups in the network or on the remote end
  - Surges of messages received on queues

As-Is: Adding a Message to a Queue

1 Application issues MQPUT to queue

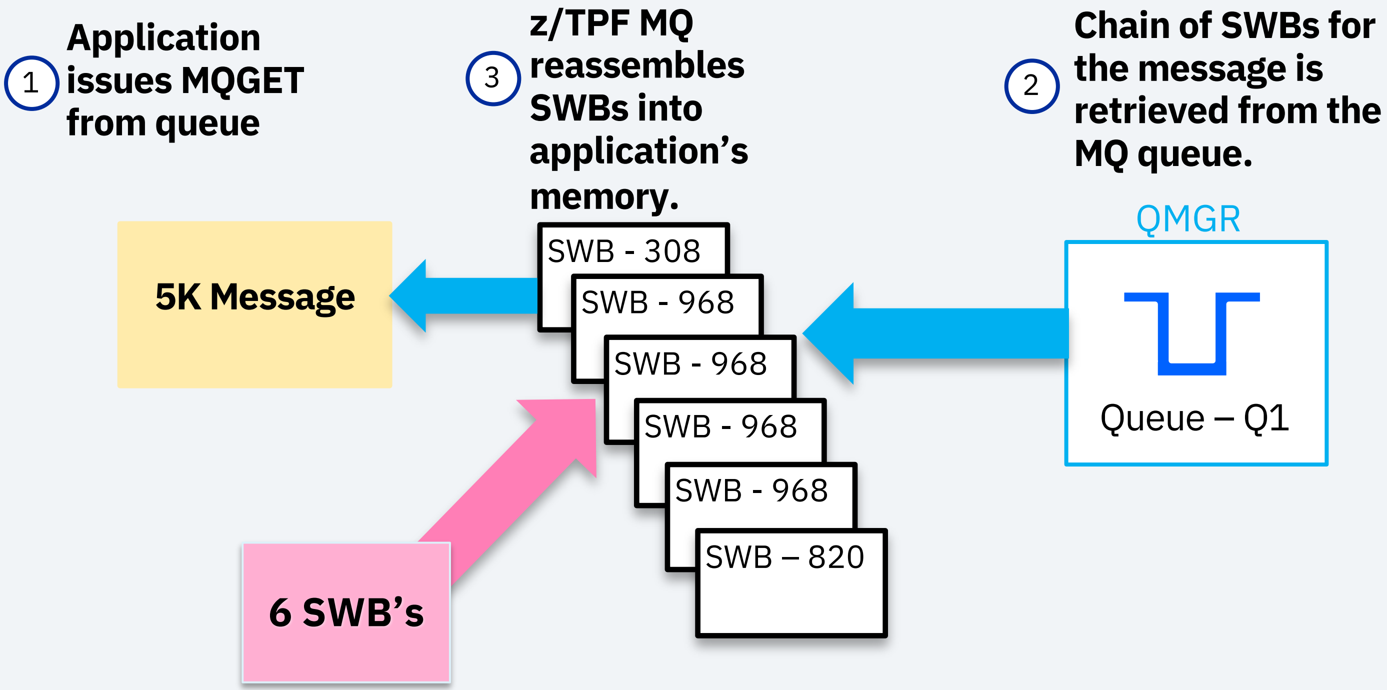
2 z/TPF MQ obtains SWBs to hold the 5K message.

3 Chain of SWBs containing the message is added to the MQ queue.



As-Is Scenario

As-Is: Retrieving a Message from a Queue



1 **Application issues MQGET from queue**

2 **Chain of SWBs for the message is retrieved from the MQ queue.**

3 **z/TPF MQ reassembles SWBs into application's memory.**

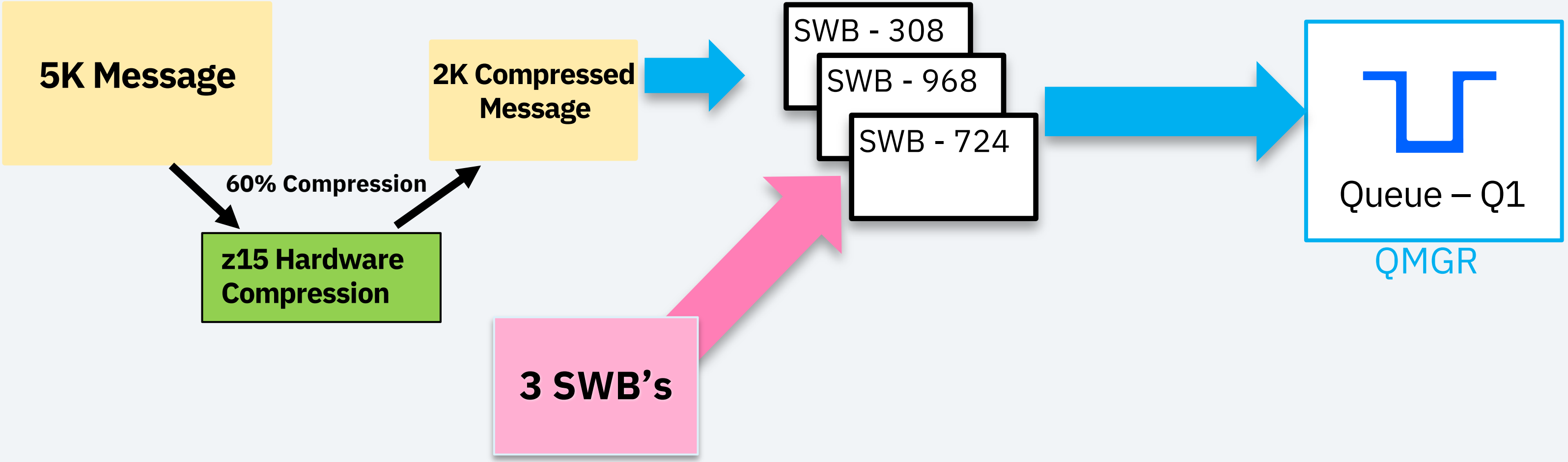
To-Be: Adding a Message to a Queue

1 Application issues MQPUT to queue

2 z15 hardware compression to compress message

3 z/TPF MQ obtains SWBs to hold the 2K message.

4 Chain of SWBs containing the message is added to the MQ queue.



5K Message

60% Compression

z15 Hardware Compression

2K Compressed Message

SWB - 308

SWB - 968

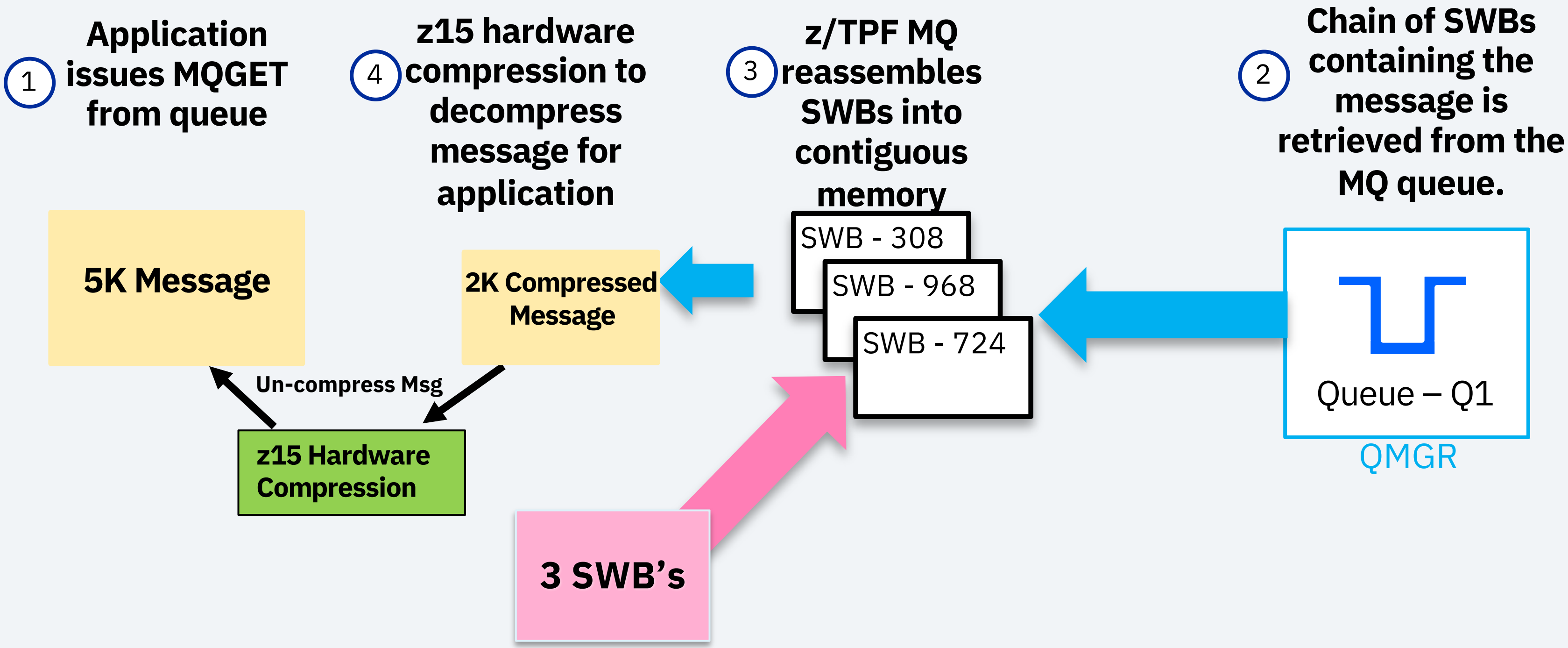
SWB - 724

3 SWB's

Queue – Q1

QMGR

To-Be: Retrieving a Message from a Queue



1 Application issues MQGET from queue

4 z15 hardware compression to decompress message for application

3 z/TPF MQ reassembles SWBs into contiguous memory

2 Chain of SWBs containing the message is retrieved from the MQ queue.

Enabling Compression on z/TPF MQ Queues

- New option on ZMQSC DEFINE QL and ZMQSC ALTER QL commands called COMPSIZE
  - Specify minimum application message size the system will use to determine if it should be compressed.
    - Messages greater than or equal to this value will be compressed when running on a processor that has Integrated Accelerator for zEDC (z15 or higher) .
    - A value of zero, means compression is disabled for z/TPF MQ queues.

Example:

```
ZMQSC ALTER QL-TPFQ COMPSIZE-16000
```

←

Any messages greater than or equal to 16,000 bytes will be compressed before adding to z/TPF MQ queue *TPFQ*

Setting the z/TPF MQ Default Compression Size

- New option on ZMQSC DEFINE QMGR and ZMQSC ALTER QMGR commands called COMPSIZE
  - Specify the default value to use for newly created queues created on the z/TPF system.
    - New queues will inherit this value when COMPSIZE is not explicitly specified for the queue.
  - A value of zero which is the default, means compression is disabled for new z/TPF MQ Queues.

Example:

```
ZMQSC ALTER QMGR-TPFQM COMPSIZE-16000
```

←

New queues created will inherit this COMPSIZE value of 16,000 bytes.

Which Queues can Compression Be Enabled On?

- Memory Queues
  - **ZMQSC DEF QL-MYQUEUE COMPSIZE-10000**

- Common Queues
  - **ZMQSC DEF QL-MYCOMMONQ COMMON-YES COMPSIZE-10000**

- Transmission Queues
  - **ZMQSC DEF QL-MYXMITQ USAGE-XMITQ COMPSIZE-10000**

Queue Statistics for Compression

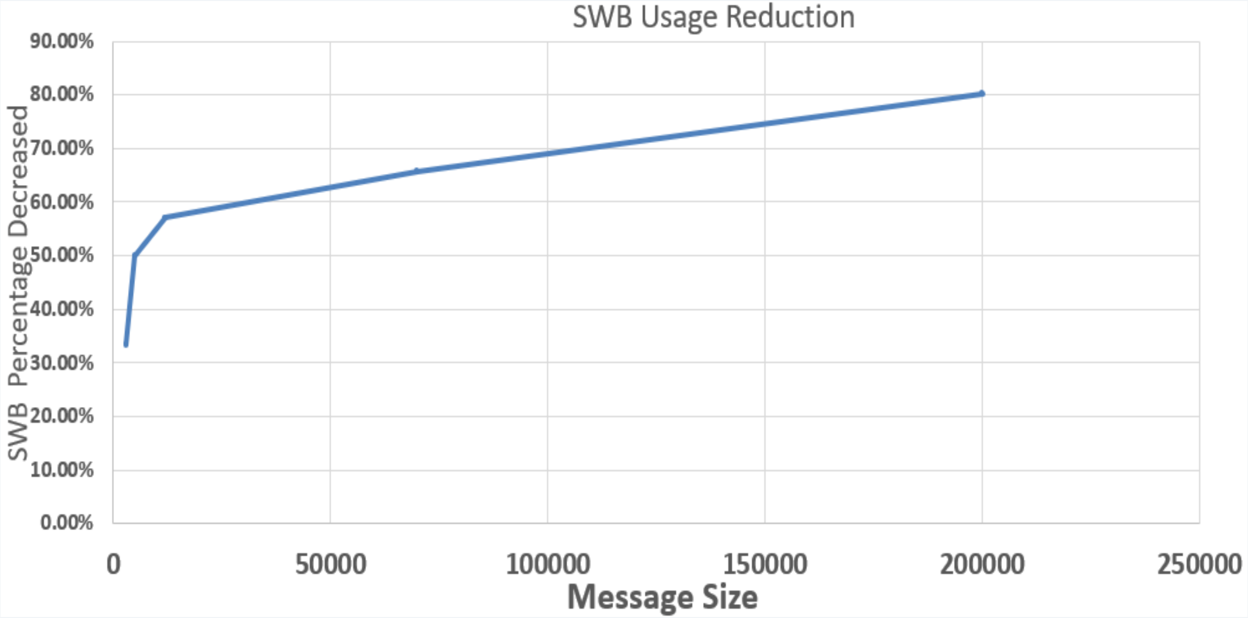
```
ZMQSC DISPLAY QL-MYQUEUE STAT
CSMP0097I 10.29.52 CPU-B SS-BSS  SSU-HPN  IS-01
MQSC0251I 10.29.52 LOCAL QUEUE STATISTICS DISPLAY: - MYQUEUE
  Current Depth        - 5
  Persistent Msgs      - 24244 (274 megabytes)
  NonPersist Msgs      - 0 (0 bytes)
  Compression Ratio    - 62 (62)
  Compression Time     - 11 (11) _
  Avg Msg Size on Q    - 11131 (11934) bytes
  Avg Orig Size Comp   - 22000 (22000) bytes
  Compressed Msgs      - 11976
  Decompressed Msgs    - 11974
  Compress per sec     - 156
  Decompress per sec   - 156
  Num of MQOPEN        - 8
  Num of MQCLOSE       - 0
  Num of SWB's in use  - 52
  Aborted Sweep Count  - 0 _
  Checkpoint High      - 0.001 seconds
  Checkpoint Last      - 0.000 seconds
  Chkpt in Progress    - NO

                         Count          Rate          RateHigh
  Get                  - 3218           315           317
  Swept Search         - 0              0             0
END OF DISPLAY+
```

MQ SWB Usage

- Number of SWBs consumed per message depends on the compression ratio and the size of the message
- Steady state where queues are not growing, the number of in-use SWBs reduced will be minimal, but as the queue grows....

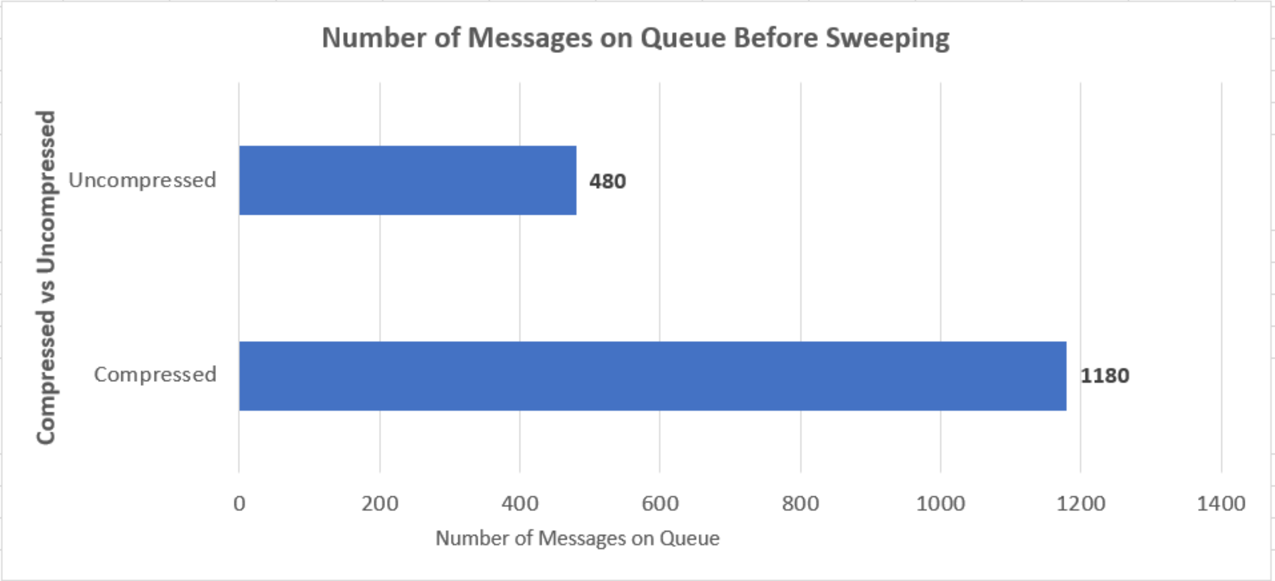
Various size XML files used as messages



Message Size	Compress Ratio	SWB Reduction
3000	53%	33%
5000	55%	50%
12000	58%	57%
70000	60%	65%
200000	78%	80%

MQ I/O Reduction - Sweeper

- Less SWBs in use reduces the likelihood of MQ sweeper activity
- Driver putting a 32K XML message onto a transmission queue once every 10ms.
  - Forced channel sender failure, measured the sweeper activity



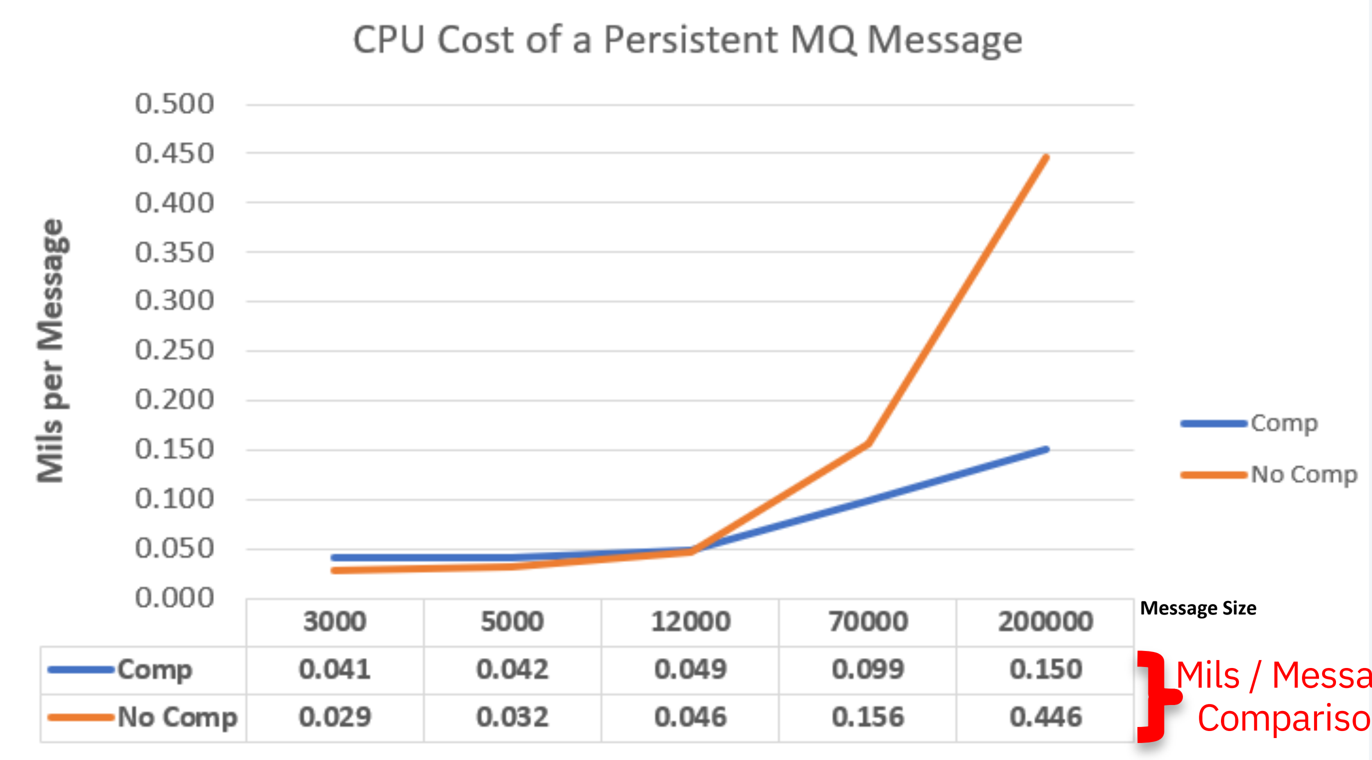
Sweeper ran every 11 seconds compressed vs every 4 seconds un-compressed

54% Reduction in the number of I/Os per message put to the queue with compression enabled.

CPU Cost of Compression for MQ Persistent Messages

- Measurement of the mils consumed per MQ persistent message

  -- Message is an MQPUT and MQGET



	3000	5000	12000	70000	200000
Comp	0.041	0.042	0.049	0.099	0.150
No Comp	0.029	0.032	0.046	0.156	0.446

Message Size

Mils / Message Comparison

41% 31% 6% 36% 66%

Message Size	Compress Ratio	SWB Reduction
3000	53%	33%
5000	55%	50%
12000	58%	57%
70000	60%	65%
200000	78%	80%

Conclusion

- z/TPF MQ Internal Compression exploits the Integrated Accelerator for zEDC (z15 or higher) to reduce the core storage required to store z/TPF MQ messages on queues.
  - Reduces 31-bit memory usage
- Can reduce the frequency of the MQ sweeper
- Can reduce the amount of I/O incurred by the MQ sweeper
- Can reduce the amount of I/O incurred during checkpointing of persistent messages
- Reduces the CPU cost of getting/putting larger messages on z/TPF MQ queues.

- Delivered with PJ46078 (December 2020)

z/TPF MQ Channel Compression

Using z/TPF MQ Channel Compression

- Use of z/TPF MQ continues to grow as an interconnectivity solution between systems within your enterprise.
  - In addition, the size of the messages being transferred over an MQ channel to or from z/TPF continues to increase.

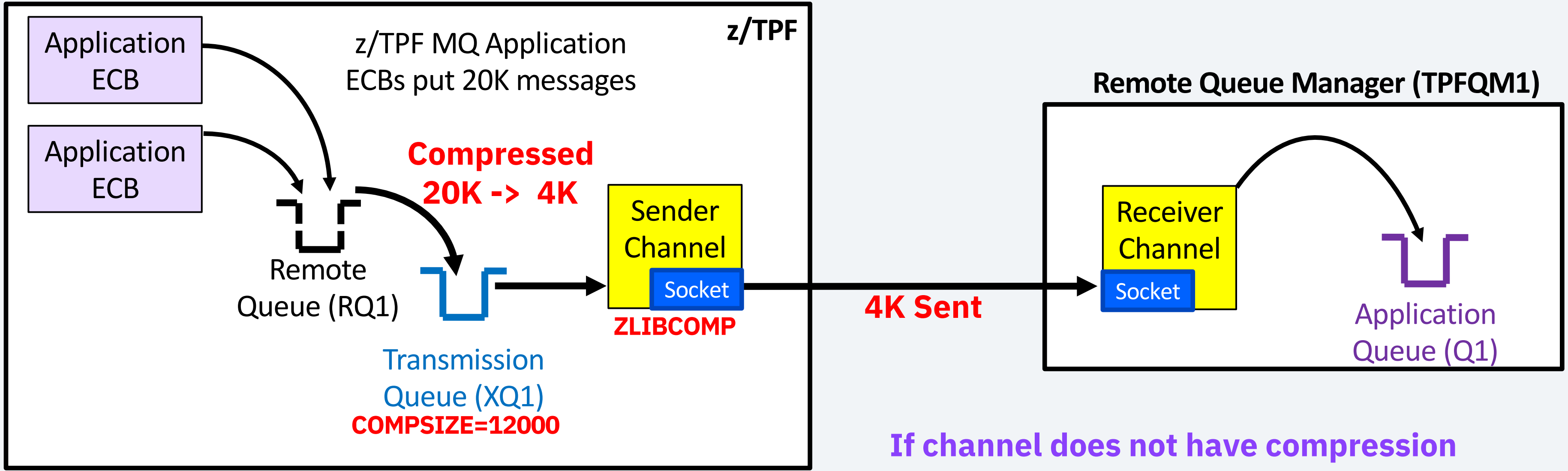
- The amount of data being sent/received by z/TPF MQ across the network can have an impact on the network and the CPU on both z/TPF and the remote end.

Enabling Compression on z/TPF MQ Channels

- New option on ZMQSC DEFINE CHL and ZMQSC ALTER CHL commands called COMPMSG
  - For a receiver channel, specify one of the following:
    - ZLIBFAST – Compression performed with speed prioritized.
    - ZLIBHIGH – Compression performed with priority on compression.
    - ANY – Supports both compression options, compression is negotiated with the sender channel.
    - NONE – Compression not supported on the channel (default).
  - For a sender channel, specify a list of supported compression options
    - ZLIBFAST, ZLIBHIGH – Supports both compression options in priority order
    - NONE – Compression not supported on the channel (default).

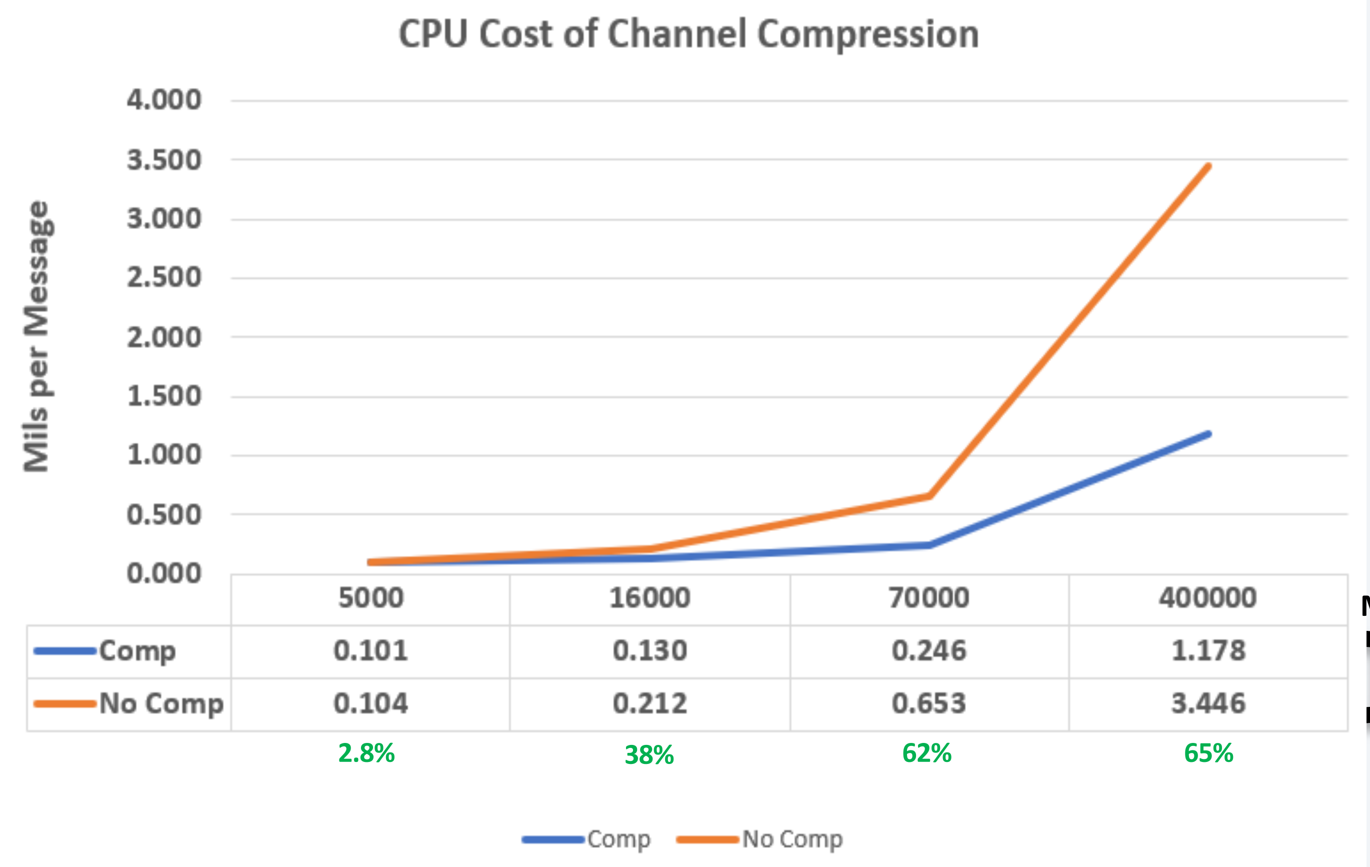
Determining the Channel Compression Message Size

- When compression is enabled for a z/TPF MQ channel, the size of the message used to determine when to compress across the network is inherited from the transmission queue (COMPSIZE) tied to the channel.



If channel does not have compression enabled, the message is inflated back to 20K before sending across the channel.

z/TPF MQ Channel Compression Performance - SSL


CPU Cost of Channel Compression

	5000	16000	70000	400000
Comp	0.101	0.130	0.246	1.178
No Comp	0.104	0.212	0.653	3.446
	2.8%	38%	62%	65%

TLS Cipher Used :
AES256-SHA256

Message Size	Batch Size	Comp Ratio
5000	2000	65%
16000	625	74%
70000	143	79%
400000	25	80%

Message Size

} Mils / Message Comparison

Conclusion

- z/TPF MQ Network Compression exploits the Integrated Accelerator for zEDC (z15 or higher) to reduce the network bandwidth, increase the throughput, and reduce the processing time to send MQ messages across a channel.
    - For TLS enabled channels, the amount of data to be encrypted will be reduced by compressing the z/TPF MQ messages, also leading to reduced processing time for the messages.

- Scheduled to be delivered with PJ46137 – End of April 2021

What’s Next

- Using storage above the 2G bar for saving z/TPF MQ messages in memory.
  - Frees up 31-bit storage below the bar to be used for other purposes.
    - Allow you to continue to grow workload on the z/TPF system

- A new MQ queue type will be introduced to minimize the impact of existing queues on the system.

z/TPF HTTP Server Compression

Problem Statement

- Use of HTTP/REST continues to grow across the customer set.
    - In addition, the payloads and body sizes sent by the z/TPF system are getting larger and larger

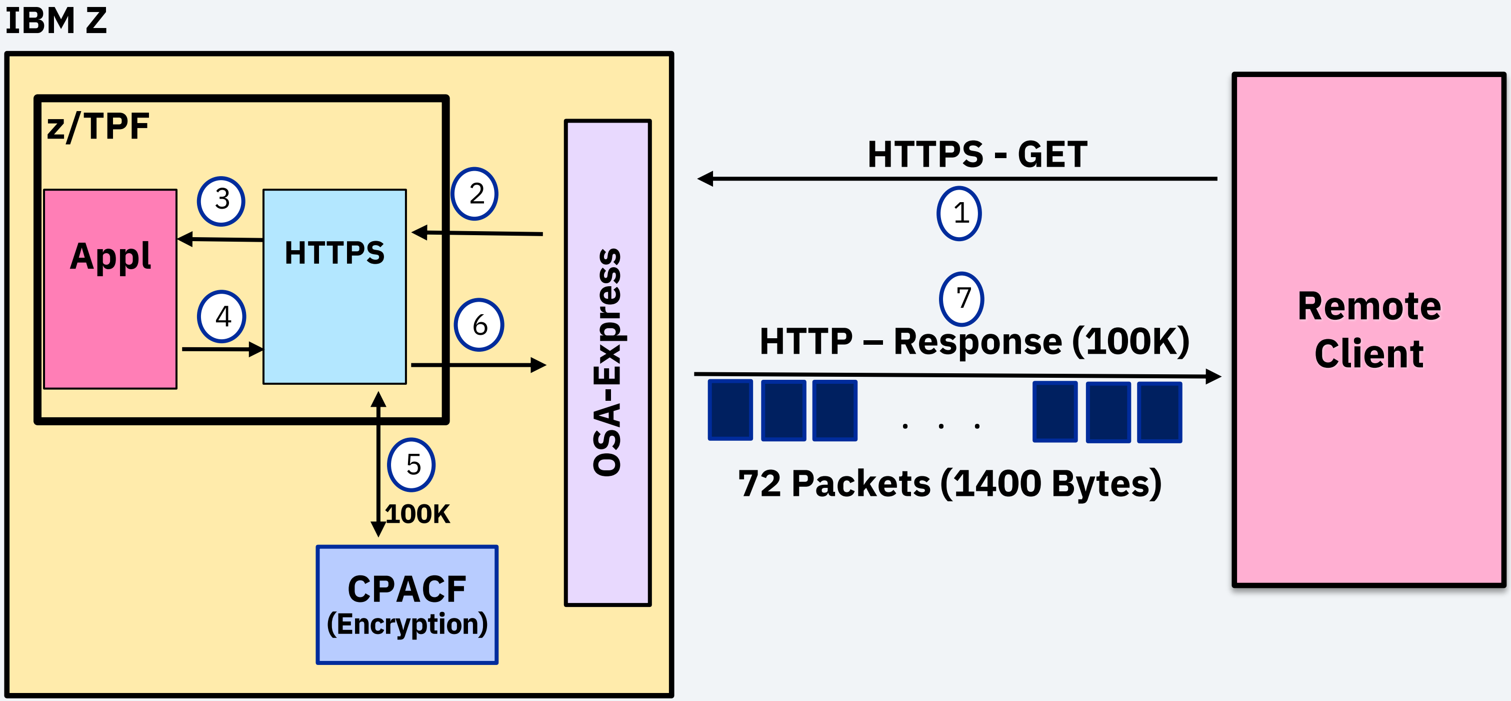
- Large amounts of connections requesting large payloads can have an impact on both the CPU and network.

Value Statement

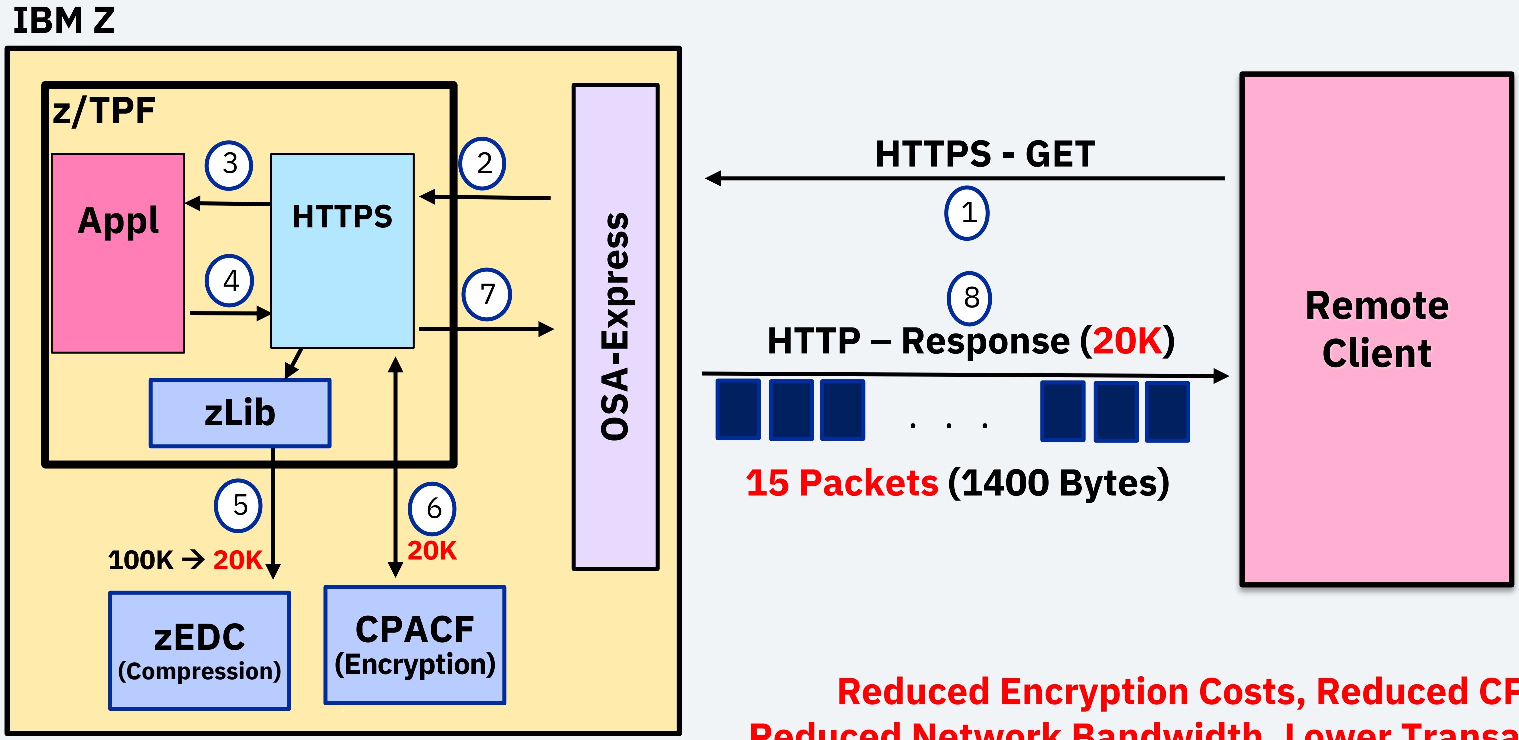
- The z/TPF HTTP server will have the ability to compress HTTP server responses when requested by the client to reduce the network bandwidth, increase the throughput, and reduce the processing time to send the responses.
  - For TLS enabled servers, the amount of data to be encrypted will be reduced by compressing the responses, also leading to reduced processing time for the responses.

As-Is Scenario

As-Is: HTTP Server Sending 100K Response



To-Be: HTTP Server Sending 100K Response



Reduced Encryption Costs, Reduced CPU Costs, Reduced Network Bandwidth, Lower Transaction Latency!

Compression on the z/TPF HTTP Server

- z/TPF HTTP Server can support either ‘deflate’ or ‘gzip’ compression
  - These compression options are available with the zlib library of z/TPF

- Only the body of an HTTP server response is compressed.
  - HTTP headers are not compressed

- New options added to the z/TPF HTTP Server Configuration File to enable and control the compression on an HTTP server instance.

Enabling Compression on the z/TPF HTTP Server

- To enable compression for an HTTP server, you specify the following options in the HTTP server configuration file in /etc/tpf_httpserver directory
  - COMP-deflate,gzip ← Specified in order of priority (used when the client doesn’t indicate a priority)
  - COMPSIZEHW-20000 ← Compress body sizes greater than or equal to 20,000 bytes when hardware is available.
  - COMPSIZESW-0 ← Do not compress body sizes when hardware is not available (don’t compress in software)

Refreshing the HTTP server configuration (ZHTPS REFRESH SERVER-*servername)* enables the compression options for all new HTTP server connections. Existing HTTP server connections are not changed.

How the HTTP Client Requests Compression

- Client’s request compression by passing an Accept-Encoding header in the HTTP request.
  - Indicates which compression options the client supports and an optional priority for each compression option

- Server responds with a Content-Encoding header indicating the compression option used in the response.

HTTP Client Request Compression Examples

- Let’s assume the COMP parameter in HTTP server configuration is:
  COMP=gzip, deflate

Client Accept-Encoding	Server Select
Accept-Encoding: deflate; q=1.0, gzip; q=0.5	Deflate
Accept-Encoding: deflate; q=1.0, gzip; q=1.0	GZIP
Accept-Encoding: deflate; q=0.5, gzip; q=0.5, identity; q=1.0	Identity = NO Compression

HTTP Server Compression Statistics

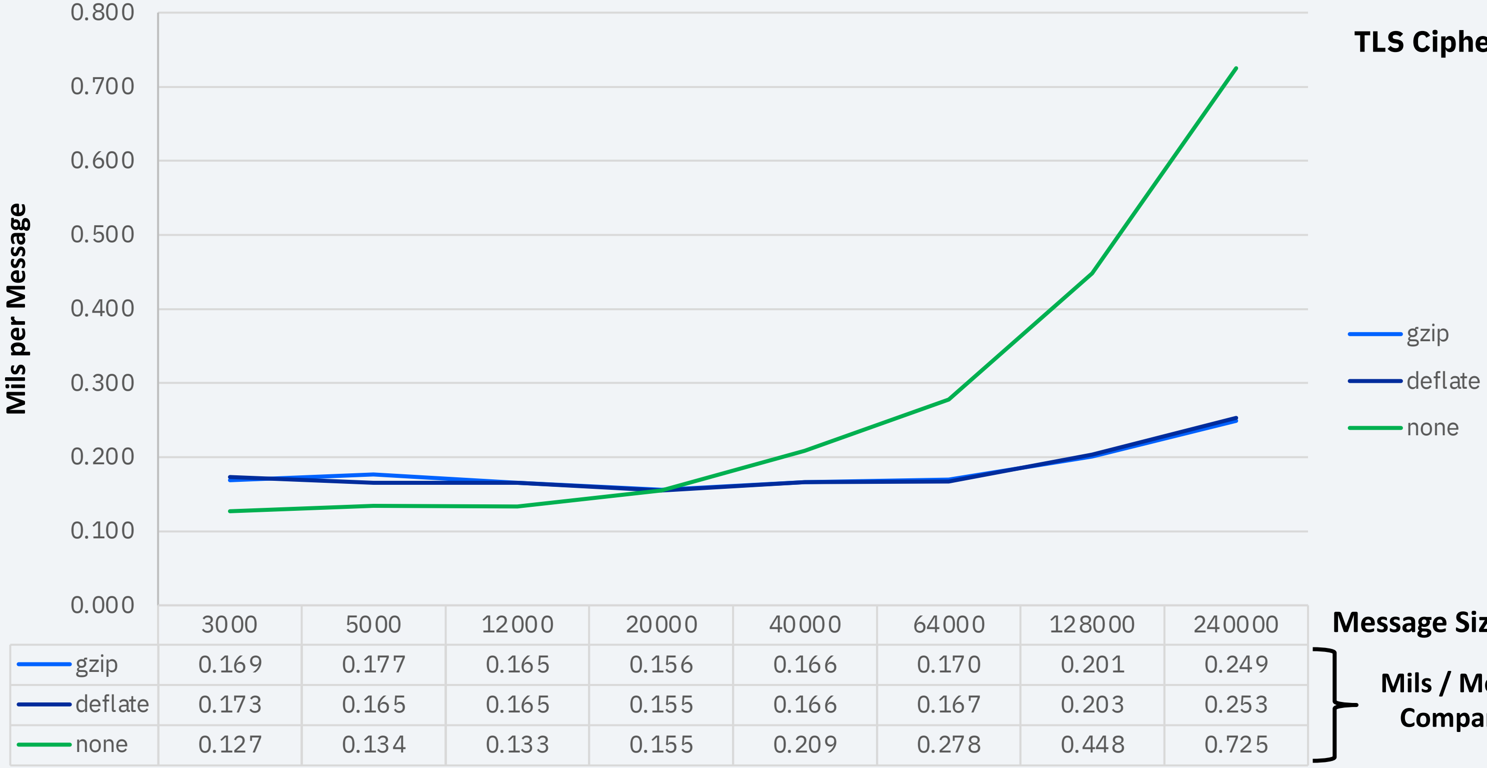
```
ZHTPS STATS S-HTTP81PORT
HTPS0011I 15.04.19 HTTP SERVER STATISTICS FOR HTTP81PORT
ACTIVE - 7                  HWMARK - 10
REFUSED - 0                 REUSED - 0
TIMED OUT - 2               URL ERRORS - 1
AVG COMPRESSION RATIO - 76.123
AVG TIME TO COMPRESS - 0.019 MILS
AVG SIZE (BEFORE COMPRESSION) - 20184
HARDWARE COMPRESSION COUNT - 7489
SOFTWARE COMPRESSION COUNT - 0
NO COMPRESSION COUNT - 349
LAST STATS RESET TOD - D962CAAA5BAAD838 (2021-03-09 10:41:22)
END OF DISPLAY+
```

A RESET option was added to the ZHTPS STATS command to clear the statistics:

ZHTPS STATS S-HTTP81PORT RESET

HTTP Server Compression Performance

CPU Cost of HTTP Hardware Compression for SSL Sessions



	3000	5000	12000	20000	40000	64000	128000	240000
gzip	0.169	0.177	0.165	0.156	0.166	0.170	0.201	0.249
deflate	0.173	0.165	0.165	0.155	0.166	0.167	0.203	0.253
none	0.127	0.134	0.133	0.155	0.209	0.278	0.448	0.725

Message Size

Mils / Message Comparison

TLS Cipher Used : AES256-SHA

Message Size	Compress Ratio
3000	64%
5000	70%
12000	79%
20000	82%
40000	84%
64000	84%
128000	85%
240000	86%

Conclusion

- z/TPF HTTP Server Compression exploits the Integrated Accelerator for zEDC (z15 or higher) to reduce network bandwidth, increase throughput, and reduce the processing time to send responses.

- For TLS enabled servers, the amount of data to be encrypted before sending can also be greatly reduced.

- Delivered with PJ46306 (April 2021)

Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

Question	Answer
Q:is using XWB's no longer the direction for MQ, to help users on z13-z14 hardware??	A: Using storage above the bar is the next project being looked at for MQ.
Q: Are costs of compression dependant on z15?	A: Correct, Internal compression needs z/15 Integrated Accelerator in order to compress messages
Q: My assumption here is the Queue needs to be empty when adding the compsize, to turn on compression?	A: Queue compresion can be enabled at any time, compression is only done when adding a message, so as messages are added after the option is enabled they will be compressed.
Q: If you have the compression parameter defined but on a z14 will that parameter just be ignored or would an error occur? Our shop has different levels of CPUs for primary/fallback.	A: Correct, on z14 without the Integrated Accelerator, the COMPSIZE parameter is ignored for compression. With PJ46137, network compression we are adding the ability to decompress messages when falling back to a CPU without hardware compression. Also, in a fallback scenario where messages are compressed on a z15 and then the system falls back to a z14 (like in a Disaster Recovery), the MQ messages compressed will be compressed in software. That was added as part of the channel compression APAR PJ46137.
Q: Were there changes to ensure that any Receiver channel can handle compression even if TPF is not currently using compression?	A: Compression is negotiated between channels, if the receiver does not support compression, the sender will not send any compressed messages. If on z/15, you would need to alter the receiver channel to allow compression, in order to receive messages.
Q: If you use Channel Compression Only (not Queue Compression) on an SSL channel, does compression take place before encryption?	A: Correct, compression is done by MQ prior to issuing the SSL api's to send the message
Q: Is the channel compression widely available on other platform such as Windows / Linux ?	A: Channel compression is wideley available on other platforms such as Windows, Linux, z/OS. In addition compression is available to MQ clients such as Java connecting to z/TPF Server connection channels (SVRCONN).

Thank you