SSL with CICS Transaction Gateway for z/OS V9.1

Performance summary

Version 1.0
1 April 2015

Alan Hollingshead
alan_hollingshead@uk.ibm.com

CICS Transaction Gateway
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

Table of Contents

Notices....3
Trademarks and service marks....3
Overview....4
Hardware....5
Software....5
Workload....5
Configuration....5
Terminology....6
Scenario 1: Comparing three different SSL ciphers....7
Workloads....7
1.1 Cost of CICS TG CPU Per Transaction for 32K COMMAREA payloads....9
1.2 Average Response Time of CICS TG for 32K COMMAREA payloads....10
1.3 CICS TG for z/OS CPU% Usage for 32K COMMAREA payloads....11
1.4 Cost of CICS TG CPU Per Transaction for 4K COMMAREA payloads....12
1.5 CICS TG for z/OS CPU% Usage for 4K COMMAREA payloads....13
1.6 zAAP offload with CICS TG for z/OS and 32K COMMAREA payloads....14
1.7 zAAP offload with CICS TG for z/OS and 4K COMMAREA payloads....15
Scenario 2: Comparison of running with and without Hardware Cryptography....16
Workloads....16
Background information on the Hardware Cryptography configuration....16
2.1 Cost of CICS TG CPU per Transaction for 32K channel payloads....18
2.2 CPU% usage of CICS TG for 32K channel payloads....19
2.3 Cost of CICS TG CPU per Transaction for 4K channel payloads....20
2.4 CPU% usage of CICS TG for 4K channel payloads....21
Conclusions....22

Notices

This report is intended for Architects, Systems Programmers, Analysts and Programmers wanting to understand the performance characteristics of CICS Transaction Gateway for z/OS V9.1. The information is not intended as the specification of any programming interfaces that are provided by CICS Transaction Gateway for z/OS 9.1 or CICS Transaction Server for z/OS.

It is assumed that the reader is familiar with the concepts and operation of CICS Transaction Gateway for z/OS 9.1.

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.

Information contained in this report has not been submitted to any formal IBM test and is distributed “asis”. The use of this information and the implementation of any of the techniques is the responsibility of the customer. Much depends on the ability of the customer to evaluate this data and project the results to their operational environment.

The performance data contained in this report was measured in a controlled environment and results obtained in other environments may vary significantly.

Trademarks and service marks

Overview

This document contains performance measurements when using an encrypted SSL (Secure Sockets Layer) connection between Java clients and CICS Transaction Gateway for z/OS V9.1.

The report contains two scenarios.
Scenario 1 – Compares three different SSL ciphers
Scenario 2 – Compares running with and without Hardware Cryptography

Two different sized payloads are used throughout. A 32K payload, and a 4K payload. Scenario 1 uses COMMAREAs, scenario 2 uses channels.

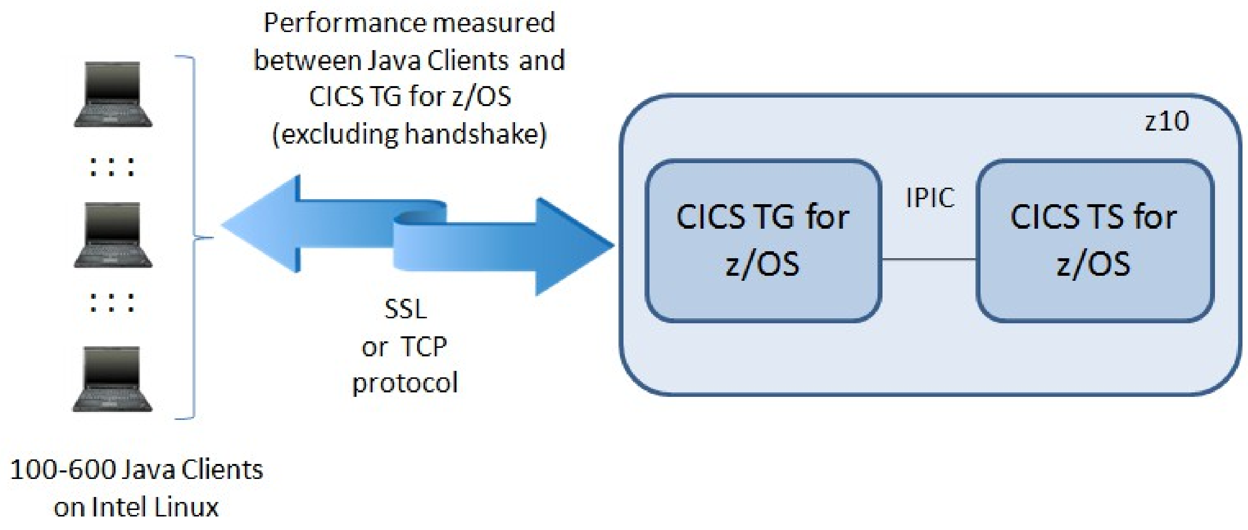

Performance measured between Java Clients and CICS TG for z/OS (excluding handshake)
z10
CICS TG for z/OS
IPIC
CICS TS for z/OS
SSL or TCP protocol
100-600 Java Clients on Intel Linux

Illustration 1: Performance Configuration

Illustration 1 shows the configuration used for both scenarios.

For inbound flows (requests) to CICS Transaction Gateway (CICS TG) , the Java clients encrypt the data before sending the data across the SSL connection. Upon receipt, the CICS TG decrypts the data. The IPIC connection between CICS TG and CICS Transaction Server (CICSTS) is not encrypted. Before sending the response back to the Java clients, CICS TG encrypts the data. Upon receipt, the Java clients then decrypt the response.

The measurements taken calculate the performance costs of using different strengths of ciphers when sending data across an SSL connection. For comparison purposes, these performance measurements are compared with Java clients connecting to CICS TG over the TCP protocol (i.e. non-SSL).

The report shows measurements for:

- CICS TG CPU cost per transaction
- CICS TG average response time
- CPU% usage of CICS TG
- zAAP offload

The Hardware Cryptography used in Scenario 2 applies to CICS TG only. See “Scenario 2: Comparison of running with and without Hardware Cryptography“ on page 16 for details of the configuration.

There are no performance measurements for the SSL handshake in this report.
Client Authentication, which is only applicable during the SSL handshake, was not enabled.

CICS TG for z/OS 9.1 was co-located on the same machine as CICS TS 5.2 on a z/OS 2.1 machine.

The measurements were taken using the following configuration:

Hardware

- IBM System z10 2097-763 model E64
- CP Assist for Cryptographic Function (CPACF)
- Crypto Express2 (CEX2) Feature
- 10GB of Central Storage (RAM)
- LPAR with 3 dedicated GCPs (for non-zAAP tests)
- LPAR with 2 dedicated GCPs and a zAAP (for zAAP tests)
- IBM System x: x3550 M3 Intel® Xeon® 5600
- OSA-Express3 10GB Ethernet SR

Software

- CICS Transaction Gateway for z/OS V9.1
- CICS Transaction Server for z/OS V5.2
- z/OS V2.1
- IBM 64-bit SDK for z/OS Java Technology Edition, Version 7.1 SR2

Workload

The workload simulation ran on an IBM System x machine running SUSE Linux Enterprise Server 11, using the CICS TG Java base classes to drive SYNCONRETURN ECI requests containing non-null payload data, thus avoiding null-stripping optimizations.

Each payload was symmetric, meaning that the size of data sent between the client and CICS Transaction Server (via the CICS Transaction Gateway), was of the same length.

The CICS Transaction Server application that received the ECI requests simply returned the same payload after altering the last byte to hex '5B'. Therefore very little work was being performed on the CICS Transaction Server in this controlled environment.

Configuration

In all tests:

- The CICS Transaction Gateway daemon address space and CICS Transaction Server region were co-located within a z/OS LPAR.

- Fast Local Sockets were used.
- TCP buffer sizes were set to 64K for both the ReceiveBufferSize and SendBufferSize.
- RMF reports were obtained through SMF to gather data about the z/OS resource usage.
- CICS TG and CICS TS were in the same STC service class.

CICS TG configuration for these measurements:

- Java7.1 SR2 64-bit
- REGION size 600M
- MEMLIMIT 12G
- Heap (Xmx) 2048M
- CTGSTART_OPTS JVM system property override: -Xcompressedrefs

The -Xcompressedrefs is recommended for users of 64-bit IBM SDK for Java to improve the effectiveness of the heap (when the maximum heap size is less than 25GB). Compressed references on 64-bit platforms decrease the size of Java objects and make more effective use of the available space. This results in less frequent garbage collection and improved memory cache utilization.

Terminology

CICS TG	- IBM CICS Transaction Gateway for z/OS
CICS TS	- IBM CICS Transaction Server for z/OS
Cost per transaction (ms)	- CPU usage per transaction, in milliseconds
CPACF	- Central Processor Assist for Cryptographic Function
CEX2	- Crypto Express2 Feature
CP	- Central Processor
CPU %	- Percentage of CPU time used by transactions running on general purpose processors
GCP	- General purpose Central Processor
HW Cryptography	- Hardware Cryptography
IPIC	- Internet Protocol (IP) interconnectivity
ICSF	- Integrated Cryptographic Service Facility
RMF	- Resource Measurement Facility
SMF	- System Management Facility
SSL	- Secure Sockets Layer
TPS	- Number of Transactions Per Second
TT	- Think Time in seconds. The time between individual requests.
zAAP	- IBM System z Application Assist Processor
zIIP	- IBM System z Integrated Information Processor

Scenario 1: Comparing three different SSL ciphers

This scenario compared workloads run across a network where the connection between the Java clients and the Gateway daemon was changed to measure the performance differences in using different cipher strengths over an SSL protocol. For comparison purposes the same workload was also run over a TCP (non-SSL) protocol.

The three ciphers used to encrypt and decrypt the data sent between the Java clients and CICS TG were:

1. SSL_RSA_WITH_NULL_SHA (null-cipher)
2. SSL_RSA_WITH_AES_128_CBC_SHA
3. SSL_RSA_WITH_AES_256_CBC_SHA

The certificates were stored in a Java keystore (.jks file)

Two payload sizes were used:

1. 32K COMMAREA
2. 4K COMMAREA

There are no performance measurements for the SSL handshake in this report.

Workloads

The workloads using the configuration shown in "Illustration 1: Performance Configuration" on Page 4 exhibited the following number of transactions per second (TPS):

32K Payloads

Transactions Per Second	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
TCP Protocol	197	396	594	792	986	1185
SSL_RSA_WITH_NULL_SHA	197	396	593	789	982	1170
SSL_RSA_WITH_AES_128_CBC_SHA	194	384	574	717	725	725
SSL_RSA_WITH_AES_256_CBC_SHA	194	384	570	670	670	656

For the most part, 32K payloads scaled proportionally until system CPU became constrained. This is noticeable with the workloads using stronger ciphers and larger numbers of clients and visually evident in the graph shown on "1.3 CICS TG for z/OS CPU % Usage for 32K COMMAREA payloads" on Page 11.

4K Payloads

The 4K payloads were not constrained by CPU and performed with the same TPS for all SSL ciphers and the TCP protocol:

Transactions Per Second	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
All SSL ciphers and TCP protocol	199	399	598	798	995	1195

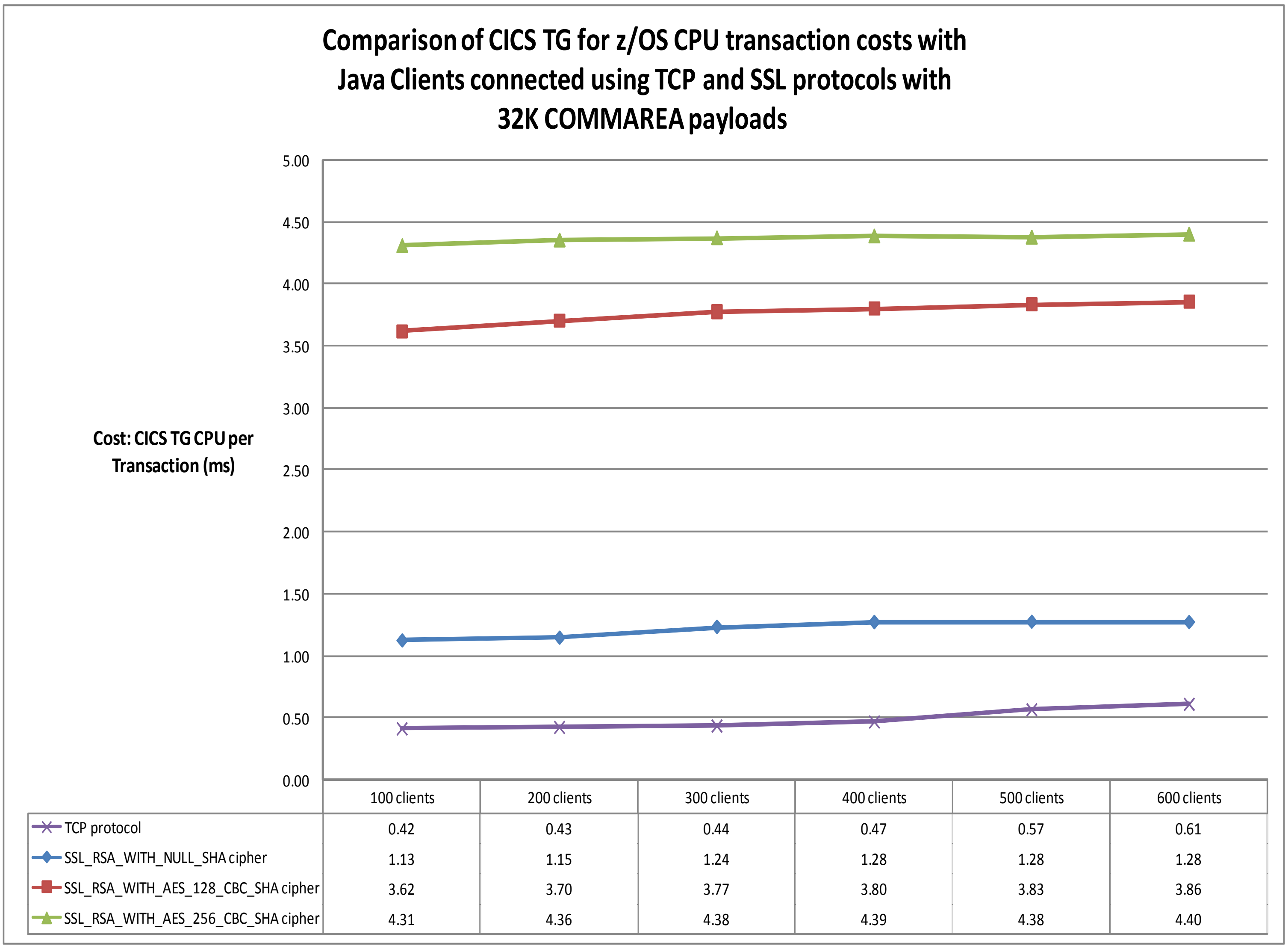
The Three Ciphers
Each cipher suite defines a key exchange algorithm, a bulk encryption algorithm, a message authentication code (MAC) algorithm, and a pseudorandom function (PRF).

The ciphers chosen for this report were carefully selected for the following reasons:

- RSA was chosen for both key agreement and key transport with a key length of 2048, as recommended in http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf
- According to http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf the US Federal Government approves the AES-128 and AES-256 encryption and decryption algorithms for 128-bit and 256-bit key lengths respectfully.
- The NULL cipher uses secure hashing to ensure consistency of delivery and so was chosen for comparison purposes with the other ciphers as well as the TCP (non-SSL) protocol.
- Block ciphers, CBC, were chosen over stream ciphers as they are more popular these days with hardware becoming cheaper.
- The SHA cryptographic hash function was chosen over MD5 because MD5 has been shown to be insecure. SHA is recommended in http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

1.1 Cost of CICS TG CPU Per Transaction for 32K COMMAREA payloads

These results measured the cost of the CICS TG CPU per transaction in milliseconds (ms) for up to 600 clients each sending and receiving a 32K COMMAREA payload.



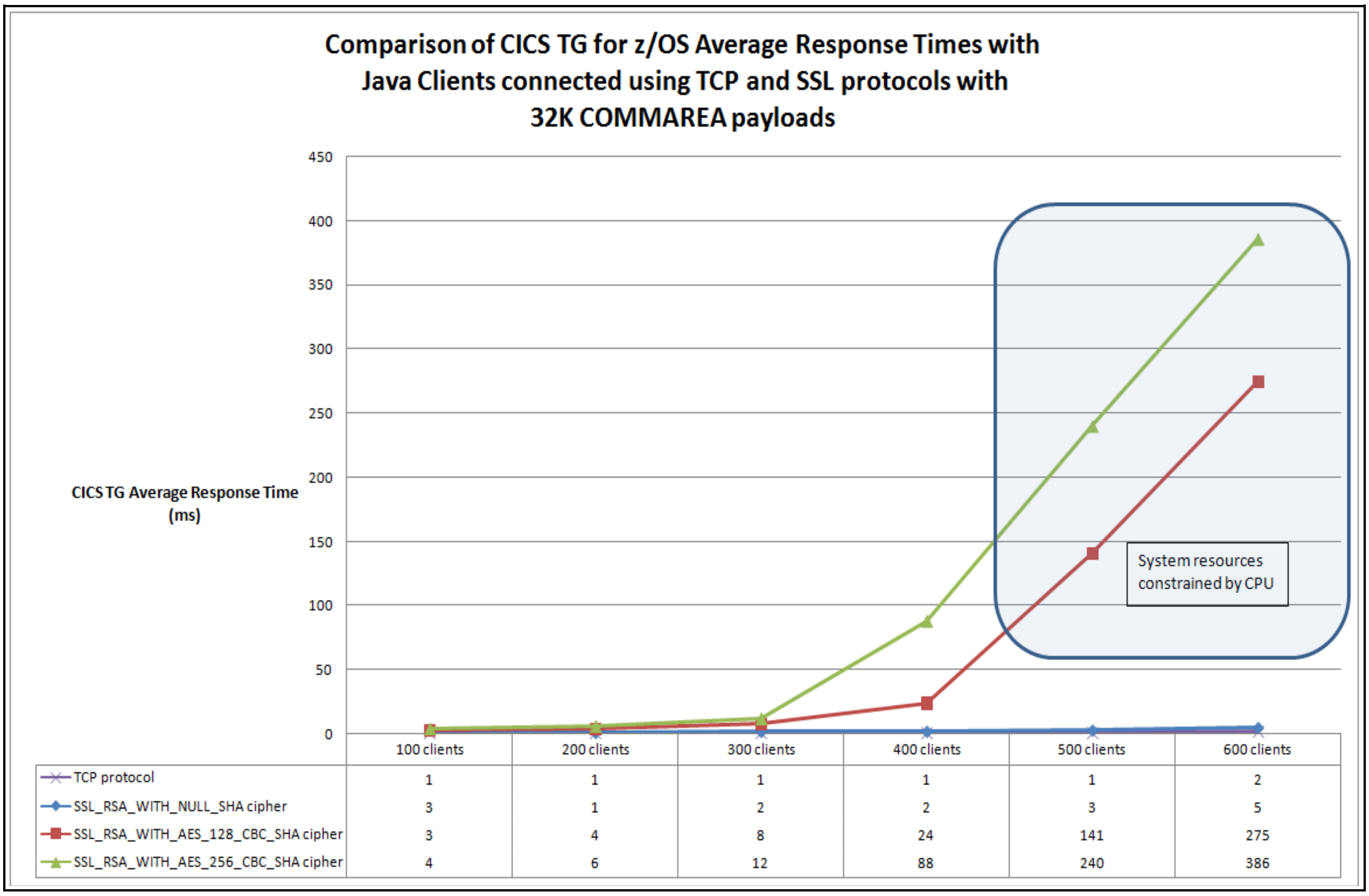
	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
× TCP protocol	0.42	0.43	0.44	0.47	0.57	0.61
◆ SSL_RSA_WITH_NULL_SHA cipher	1.13	1.15	1.24	1.28	1.28	1.28
■ SSL_RSA_WITH_AES_128_CBC_SHA cipher	3.62	3.70	3.77	3.80	3.83	3.86
▲ SSL_RSA_WITH_AES_256_CBC_SHA cipher	4.31	4.36	4.38	4.39	4.38	4.40

Observations

1. The CPU transaction cost increases with the stronger ciphers. This is to be expected.
2. The SSL_RSA_WITH_NULL_SHA cipher is slightly more costly than using a TCP (non-SSL) protocol due to secure hashing required to ensure consistency of delivery.
3. CICS TG demonstrates good CPU transaction cost scalability for 100 to 600 clients using the same cipher.

1.2 Average Response Time of CICS TG for 32K COMMAREA payloads

These results measured the CICS TG average response time using the GD_IAVRESP statistic provided with the product. The interval statistic was reset to zero between each performance run.



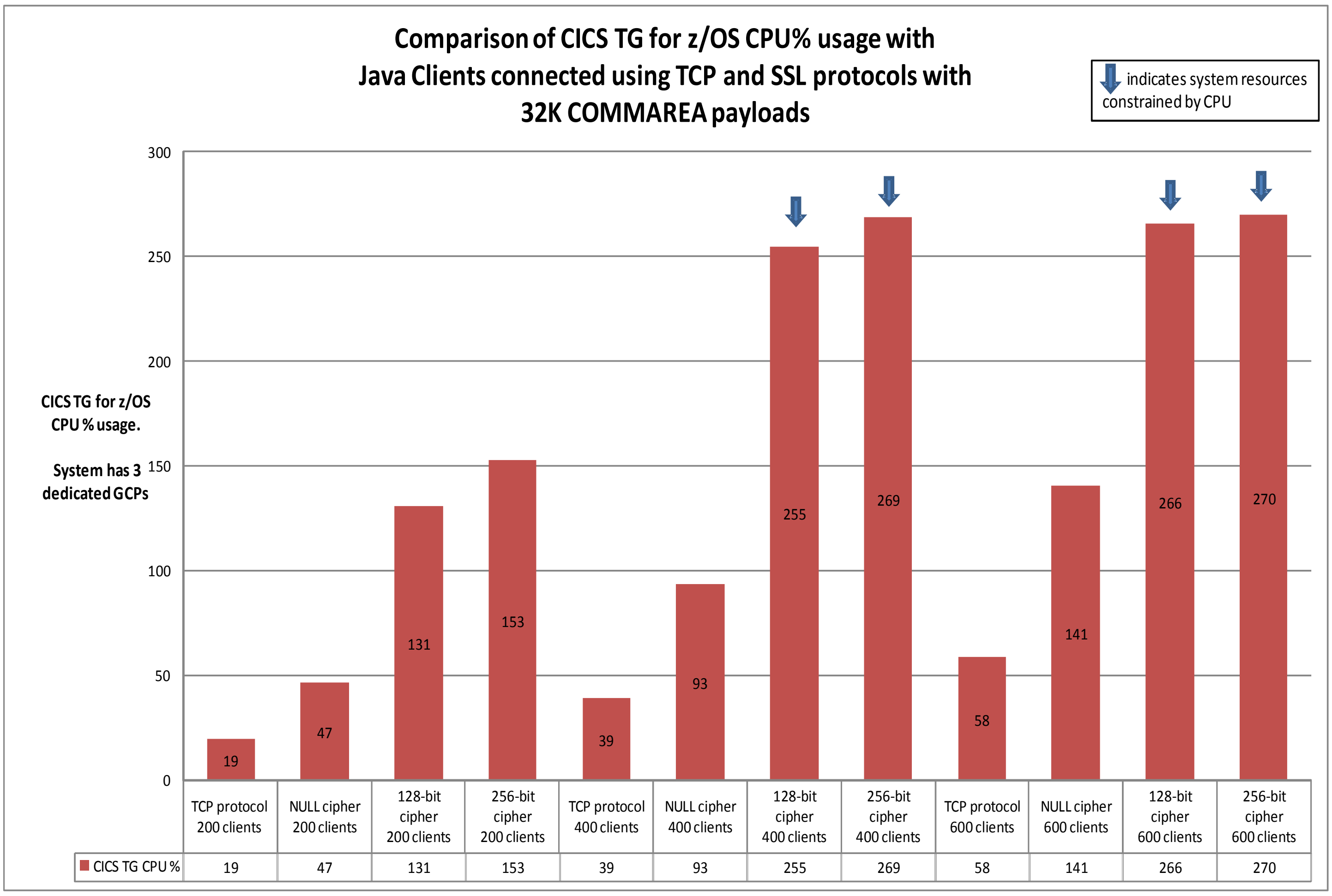
	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
× TCP protocol	1	1	1	1	1	2
◆ SSL_RSA_WITH_NULL_SHA cipher	3	1	2	2	3	5
■ SSL_RSA_WITH_AES_128_CBC_SHA cipher	3	4	8	24	141	275
▲ SSL_RSA_WITH_AES_256_CBC_SHA cipher	4	6	12	88	240	386

Observations

1. As the number of clients increased, more CPU was required to handle the requests. In turn, more internal queuing was required within the Gateway daemon. This accounts for the average response times increasing. However, when the stronger cipher keys were used with 500 and 600 clients, the system resources became constrained by CPU, and so the average response of each request took significantly longer. Despite this, the average response time scaled proportionally.
2. Customers impacted by CPU constraints are strongly recommended to consider using a zAAP/zIIP to offload work from the GCPs.
3. When the same scenario was run with 4K payloads the average response time was between 0 and 1 ms throughout as the system was not constrained by CPU.

1.3 CICS TG for z/OS CPU% Usage for 32K COMMAREA payloads

These results measured the CPU% usage of CICS TG for z/OS. Three 100% dedicated GCPs were available therefore the RMF reports can show CICS TG, CICS TS, TCPIP and other system processes using up to 300%. The results compared runs using the TCP protocol (non-SSL) with those using SSL and the different ciphers. The graph shows results for 200, 400 and 600 Java clients.



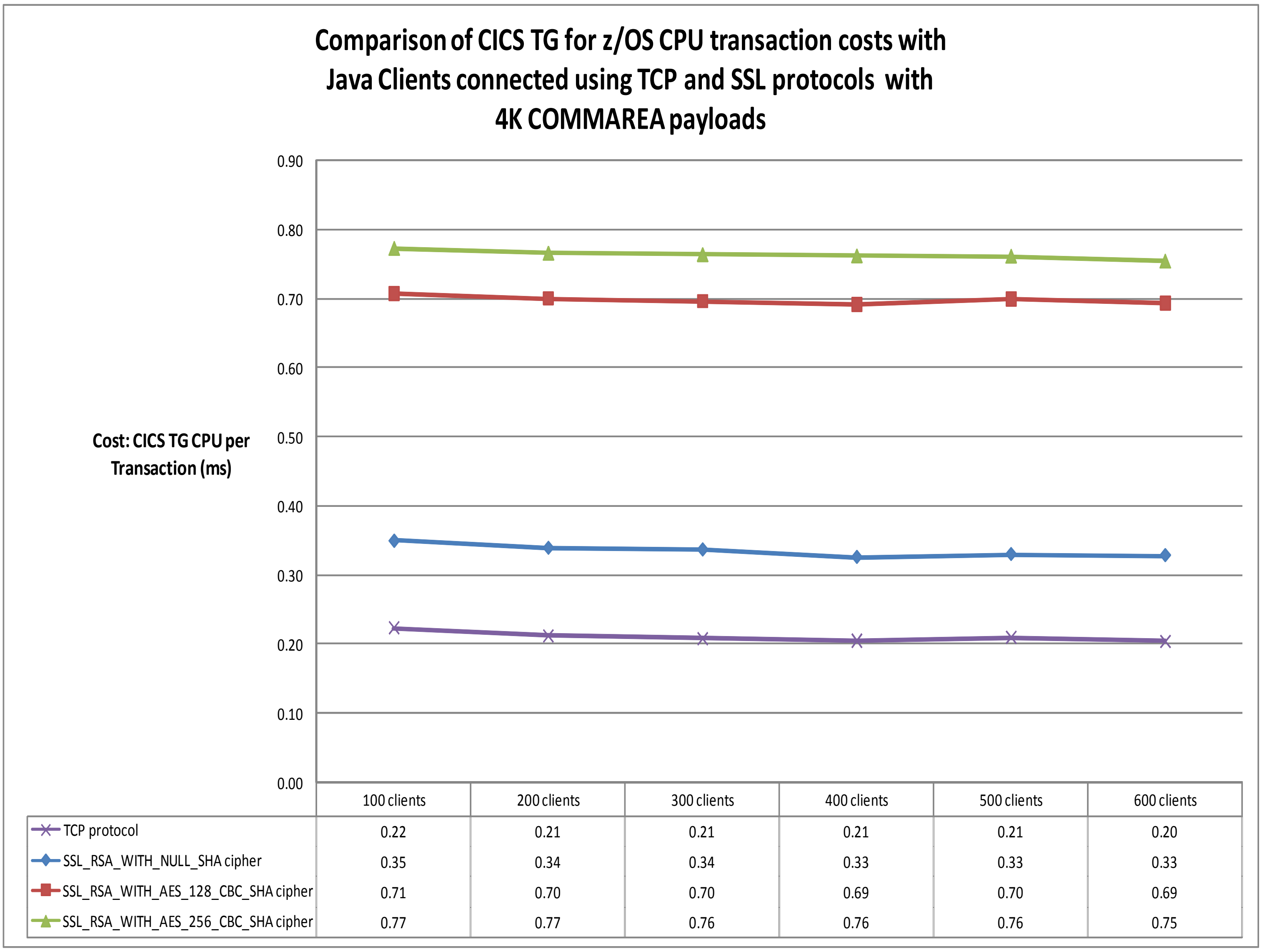
	TCP protocol 200 clients	NULL cipher 200 clients	128-bit cipher 200 clients	256-bit cipher 200 clients	TCP protocol 400 clients	NULL cipher 400 clients	128-bit cipher 400 clients	256-bit cipher 400 clients	TCP protocol 600 clients	NULL cipher 600 clients	128-bit cipher 600 clients	256-bit cipher 600 clients
🟥 CICS TG CPU %	19	47	131	153	39	93	255	269	58	141	266	270

Observations

1. CICS TG is observed to scale proportionally until the system CPU is constrained.
2. With systems that are constrained, customers are recommended to add a zAAP/zIIP or replace a GCP with a zAAP/zIIP to offload work from the GCPs. This can save on running costs and GCP charges. See “1.6 zAAP offload with CICS TG for z/OS and 32K COMMAREA payloads“ on page 14.

1.4 Cost of CICS TG CPU Per Transaction for 4K COMMAREA payloads

These results measured the cost of the CICS TG CPU per transaction in milliseconds (ms) for up to 600 clients each sending and receiving a 4K COMMAREA payload.



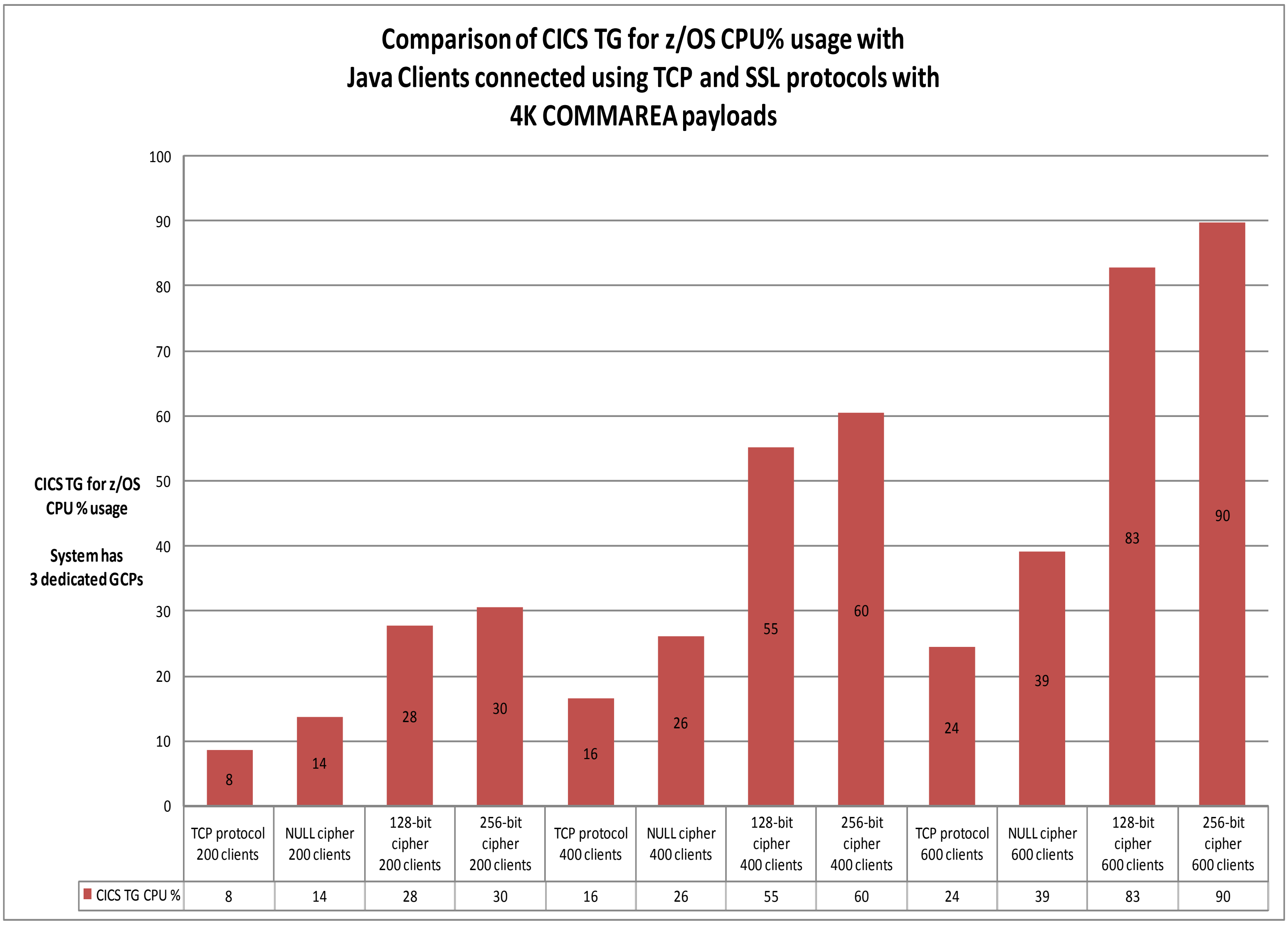
	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
✕ TCP protocol	0.22	0.21	0.21	0.21	0.21	0.20
◆ SSL_RSA_WITH_NULL_SHA cipher	0.35	0.34	0.34	0.33	0.33	0.33
■ SSL_RSA_WITH_AES_128_CBC_SHA cipher	0.71	0.70	0.70	0.69	0.70	0.69
▲ SSL_RSA_WITH_AES_256_CBC_SHA cipher	0.77	0.77	0.76	0.76	0.76	0.75

Observations

1. The CPU transaction cost increases with the stronger ciphers. This is to be expected.
2. The SSL_RSA_WITH_NULL_SHA cipher is slightly more costly than using the TCP (non-SSL) protocol due to secure hashing required to ensure consistency of delivery.
3. CICS TG demonstrates good CPU transaction cost scalability for 100 to 600 clients using the same cipher.

1.5 CICS TG for z/OS CPU% Usage for 4K COMMAREA payloads

These results measured the CPU% usage of CICS TG for z/OS. Three 100% dedicated GCPs were available therefore the RMF reports can show CICS TG using up to 300%. The results compare TCP with SSL using 200, 400 and 600 Java clients.



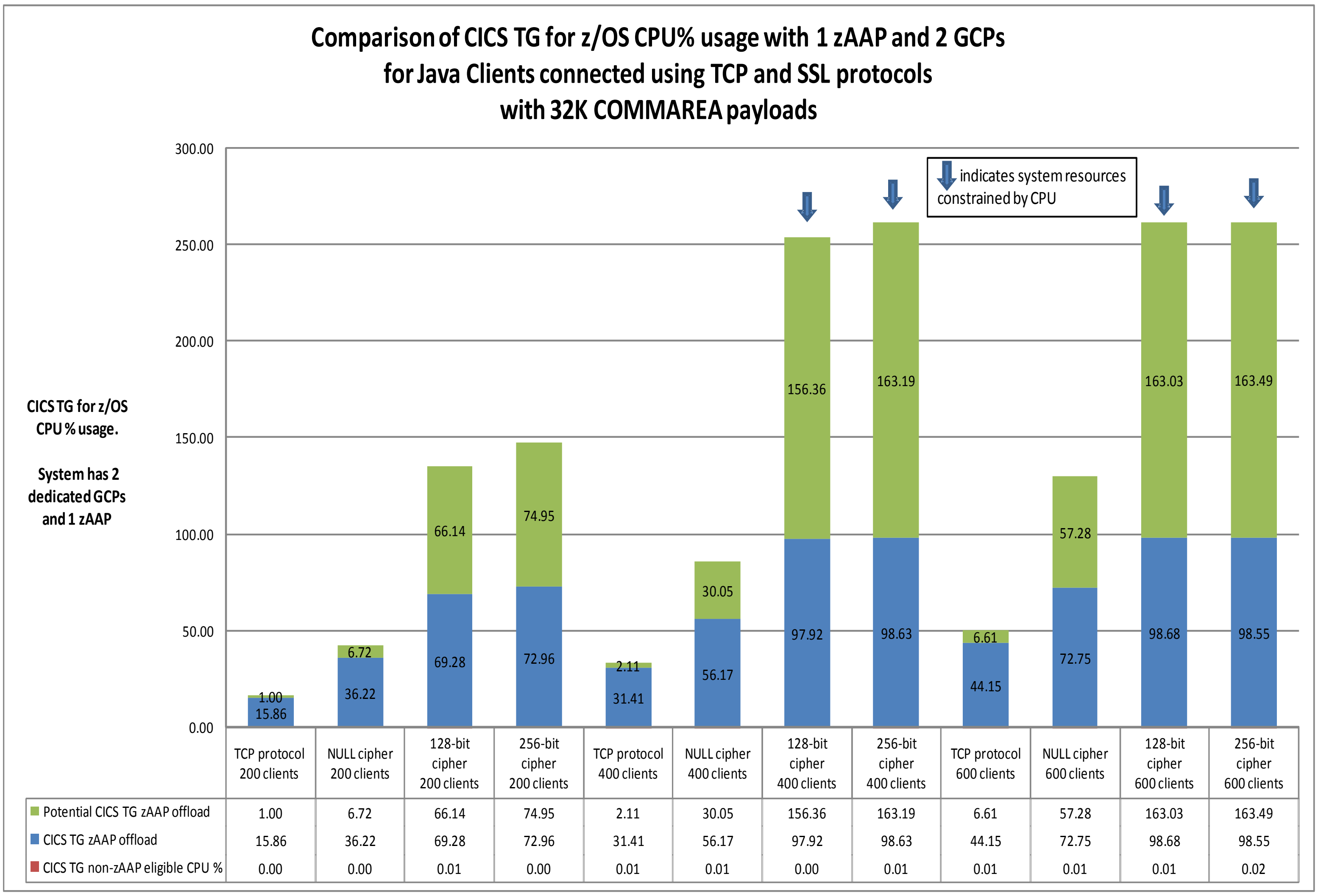
	TCP protocol 200 clients	NULL cipher 200 clients	128-bit cipher 200 clients	256-bit cipher 200 clients	TCP protocol 400 clients	NULL cipher 400 clients	128-bit cipher 400 clients	256-bit cipher 400 clients	TCP protocol 600 clients	NULL cipher 600 clients	128-bit cipher 600 clients	256-bit cipher 600 clients
🟥 CICS TG CPU %	8	14	28	30	16	26	55	60	24	39	83	90

Observations

1. CICS TG is observed to scale proportionally.
2. Although the systems for this 4K payload scenario are not constrained, customers are recommended to consider adding a zAAP/zIIP or replace a GCP with a zAAP/zIIP to offload work from the GCPs. This can save on running costs and GCP charges. See “1.7 zAAP offload with CICS TG for z/OS and 4K COMMAREA payloads“ on page 15.

1.6 zAAP offload with CICS TG for z/OS and 32K COMMAREA payloads

These results measured the CPU% usage of CICS TG for z/OS with a zAAP available to offload work from the GCPs. Two 100% dedicated GCPs were available alongside a single zAAP processor, therefore the RMF reports can show CICS TG using up to 300%. The results compared runs using the TCP protocol (non-SSL) with those using SSL and the different ciphers. The graph shows results for 200, 400 and 600 Java clients.



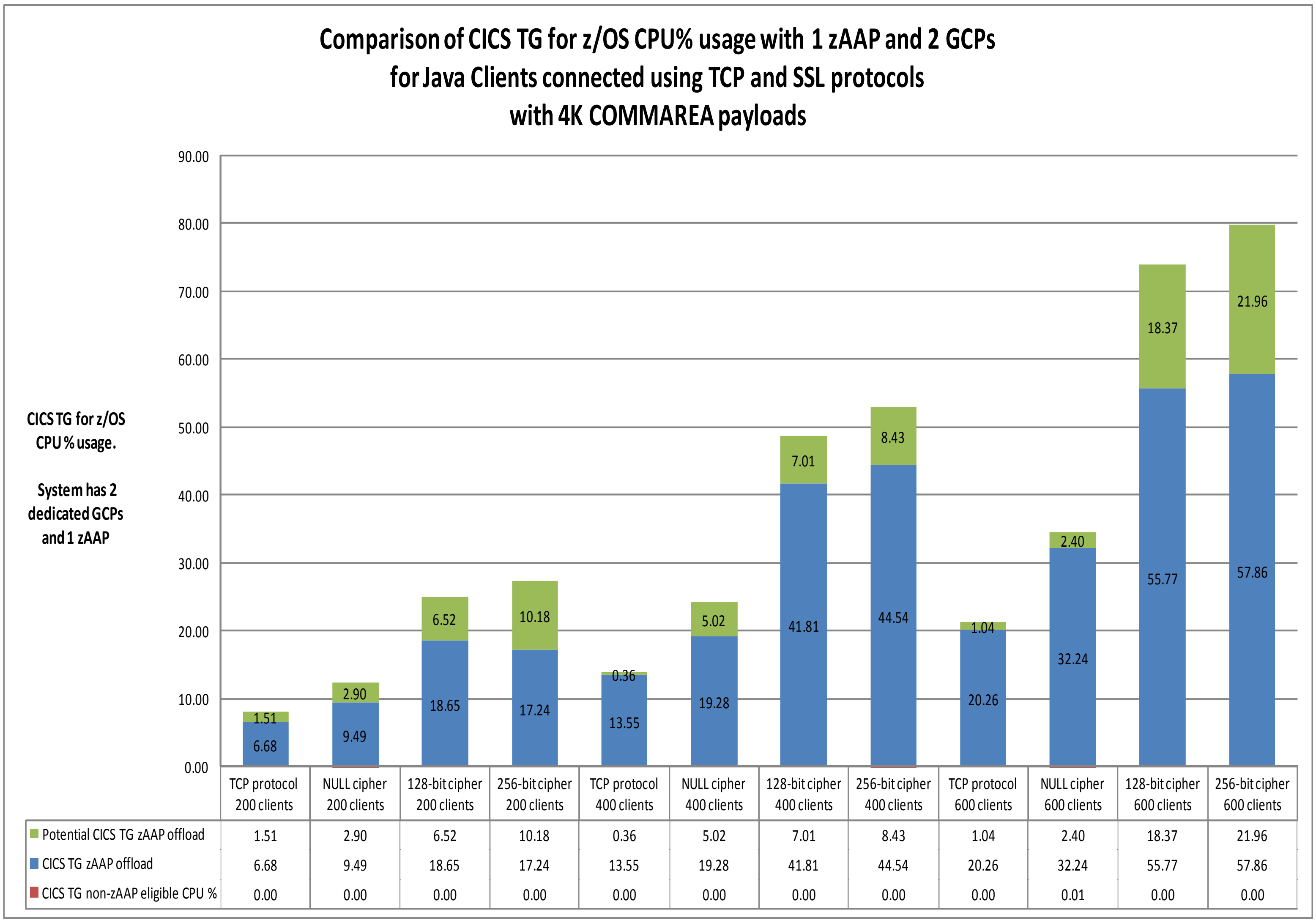
	TCP protocol 200 clients	NULL cipher 200 clients	128-bit cipher 200 clients	256-bit cipher 200 clients	TCP protocol 400 clients	NULL cipher 400 clients	128-bit cipher 400 clients	256-bit cipher 400 clients	TCP protocol 600 clients	NULL cipher 600 clients	128-bit cipher 600 clients	256-bit cipher 600 clients
🟩 Potential CICS TG zAAP offload	1.00	6.72	66.14	74.95	2.11	30.05	156.36	163.19	6.61	57.28	163.03	163.49
🟦 CICS TG zAAP offload	15.86	36.22	69.28	72.96	31.41	56.17	97.92	98.63	44.15	72.75	98.68	98.55
🟥 CICS TG non-zAAP eligible CPU %	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.02

Observations

1. The single zAAP was able to offload almost all the CPU required from the GCPs.
2. An additional zAAP would be advantageous for customers using stronger cipher keys (as indicated by the “Potential CICS TG zAAP offload” green bars) for this particular workload.
3. For the stronger ciphers with 400-600 clients, the system was constrained by CPU, as indicated by the arrows. Customers with systems showing similar constraints should consider more than one zAAP/zIIP to ease CPU contention.

1.7 zAAP offload with CICS TG for z/OS and 4K COMMAREA payloads

These results measured the CPU% usage of CICS TG for z/OS with a zAAP available to offload work from the GCPs. Two 100% dedicated GCPs were available alongside a single zAAP processor, therefore the RMF reports can show CICS TG using up to 300%. The results compared runs using the TCP protocol (non-SSL) with those using SSL and the different ciphers. The graph shows results for 200, 400 and 600 Java clients.



	TCP protocol 200 clients	NULL cipher 200 clients	128-bit cipher 200 clients	256-bit cipher 200 clients	TCP protocol 400 clients	NULL cipher 400 clients	128-bit cipher 400 clients	256-bit cipher 400 clients	TCP protocol 600 clients	NULL cipher 600 clients	128-bit cipher 600 clients	256-bit cipher 600 clients
🟩 Potential CICS TG zAAP offload	1.51	2.90	6.52	10.18	0.36	5.02	7.01	8.43	1.04	2.40	18.37	21.96
🟦 CICS TG zAAP offload	6.68	9.49	18.65	17.24	13.55	19.28	41.81	44.54	20.26	32.24	55.77	57.86
🟥 CICS TG non-zAAP eligible CPU %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00

Observations

1. The single zAAP was able to offload up to 57.86% (of 100%) CPU from the GCPs.
2. Due to the client requests running concurrently the zAAP was not available for all requests, hence why the zAAP is not 100% utilized in this scenario.
3. An additional zAAP would be advantageous for customers using stronger cipher keys (as indicated by the “Potential CICS TG zAAP offload” green bars) for this particular workload.

Scenario 2: Comparison of running with and without Hardware Cryptography

In scenario 2 a series of tests were performed to measure the value of running with hardware cryptography to offload the encryption and decryption of the payloads sent between the Java clients and CICS TG.

Two payload sizes were used:

1. 32K channel
2. 4K channel

There are no performance measurements for the SSL handshake in this report.

Workloads

The workloads using the configuration shown in Illustration 1: Performance Configuration on Page 4 exhibited the following number of transactions per second (TPS):

32K Payloads

Transactions Per Second	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
TCP Protocol	198	395	592	789	983	1176
SSL protocol with HW cryptography	198	395	588	766	795	792
SSL protocol without HW cryptography	197	394	586	756	764	751

4K Payloads

Transactions Per Second	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
TCP Protocol	200	399	599	797	998	1195
SSL protocol with HW cryptography	199	398	597	796	996	1195
SSL protocol without HW cryptography	199	397	597	797	995	1194

For smaller payloads (for example 4K) the TPS scales proportionally. For the most part, 32K payloads scaled proportionally until system CPU became constrained. This is particularly evident in “2.2 CPU% usage of CICS TG for 32K channel payloads“ on Page 19.

Background information on the Hardware Cryptography configuration

Hardware cryptography varies between different machines, and depending on how the features of hardware cryptography are configured, significant performance gains can be achieved. Newer machines than the z10, such as the zEC12 or z13, provide even greater advantages for hardware cryptography and so should be strongly considered if using SSL.

The cryptographic hardware features available on the System z10 EC include a CP (Central Processor) Assist for Cryptographic Functions (CPACF) and a Crypto Express2 (CEX2) Feature.

CPACF provides encryption accelerator functionality on a quad-core chip, which is designed to provide high-speed cryptography. The CEX2 feature combines the functions of Coprocessor mode (for secure key encrypted transactions) and Accelerator mode (for SSL acceleration) in a single feature with two PCI-X adapters. For this scenario the CEX2 feature was configured with one Coprocessor and one Accelerator.

CPACF is invoked using the Integrated Cryptographic Service Facility (ICSF). ICSF is the software on a z/OS system that serves as an interface with the hardware to direct the requests to the right CP (CPACF or CEX2).

JCECCARACFKS keystore

For this report the configuration used a JCECCARACFKS keystore to take advantage of Java Secure Sockets Extension (JSSE), RACF and ICSF Security. The certificates were stored in RACF, and the keys in ICSF. Certificates and keys were generated using RACF RACDCERT commands on TSO.

This scenario used the SSL_RSA_WITH_AES_128_CBC_SHA cipher.

The CICS TG configuration file, ctg.ini, was updated to include the following entries required for this scenario:

```
SECTION PRODUCT
: : :
esmkeyring=on
keyring=CTGV91RING
hwcrypt=on
ENDSECTION
SECTION GATEWAY
: : :
protocol@ssl.parameters=port=1234;\
: : :
ciphersuites=SSL_RSA_WITH_AES_128_CBC_SHA;
ENDSECTION
```

Details of these parameters are in the CICS Transaction Gateway for z/OS Knowledge Center: http://ibm.com/support/knowledgecenter/SSZHJ2/welcome.html

On zFS where CICS TG was running, the java.security file (located in <java-home>/lib/security) was updated with the following providers:

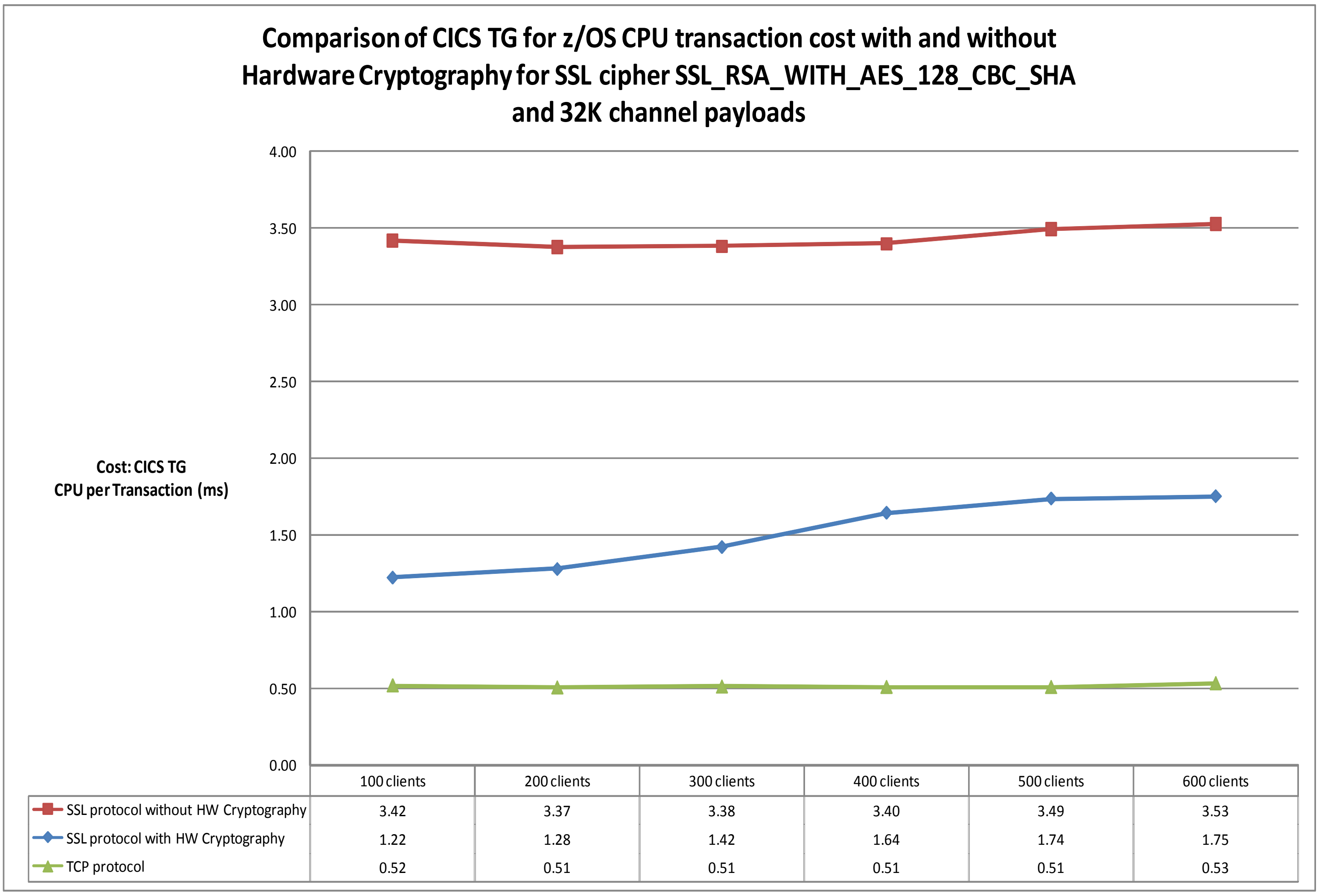
```
security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
security.provider.2=com.ibm.jsse2.IBMJSSEProvider2
```

In addition, the latest US_export_policy.jar and local_policy.jar files available from Java were used to enable full function cryptography.

2.1 Cost of CICS TG CPU per Transaction for 32K channel payloads

These results compared the CICS TG for z/OS CPU transaction cost when running with or without hardware cryptography. The Java clients connected to CICS TG over SSL using the SSL cipher SSL_RSA_WITH_AES_128_CBC_SHA. The payloads used were 32K channels.

For comparative purposes, the results shown in the graph include Java clients connected to CICS TG over the TCP (non-SSL) protocol.



	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
■ SSL protocol without HW Cryptography	3.42	3.37	3.38	3.40	3.49	3.53
◆ SSL protocol with HW Cryptography	1.22	1.28	1.42	1.64	1.74	1.75
▲ TCP protocol	0.52	0.51	0.51	0.51	0.51	0.53

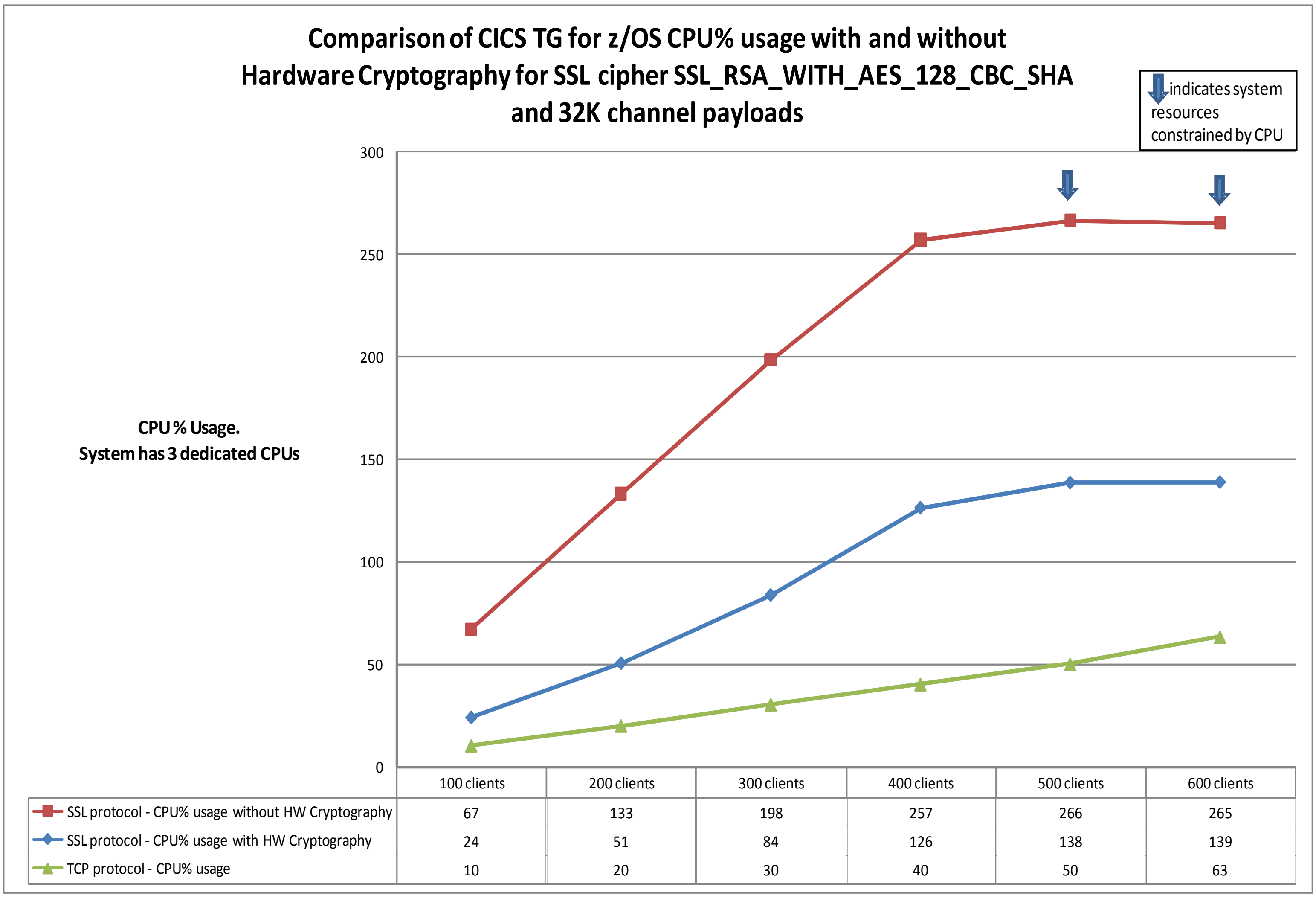
Observations

1. Using hardware cryptography to encrypt and decrypt the 32K channel payloads the GCP saving of CICS TG CPU cost per transaction was considerable for Java clients connected over SSL. This is because the majority of the workload is offloaded to the hardware cryptography CP.
2. CICS TG demonstrates good CPU transaction cost scalability for 100 to 600 clients for both with and without hardware cryptography.

2.2 CPU% usage of CICS TG for 32K channel payloads

These results measured the CICS TG CPU% usage with and without hardware cryptography enabled. The Java clients connected to CICS TG over SSL using the SSL cipher SSL_RSA_WITH_AES_128_CBC_SHA. The payloads used were 32K channels.

For comparative purposes, the results shown in the graph include Java clients connected to CICS TG over the TCP protocol.



	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
■ SSL protocol - CPU% usage without HW Cryptography	67	133	198	257	266	265
◆ SSL protocol - CPU% usage with HW Cryptography	24	51	84	126	138	139
▲ TCP protocol - CPU% usage	10	20	30	40	50	63

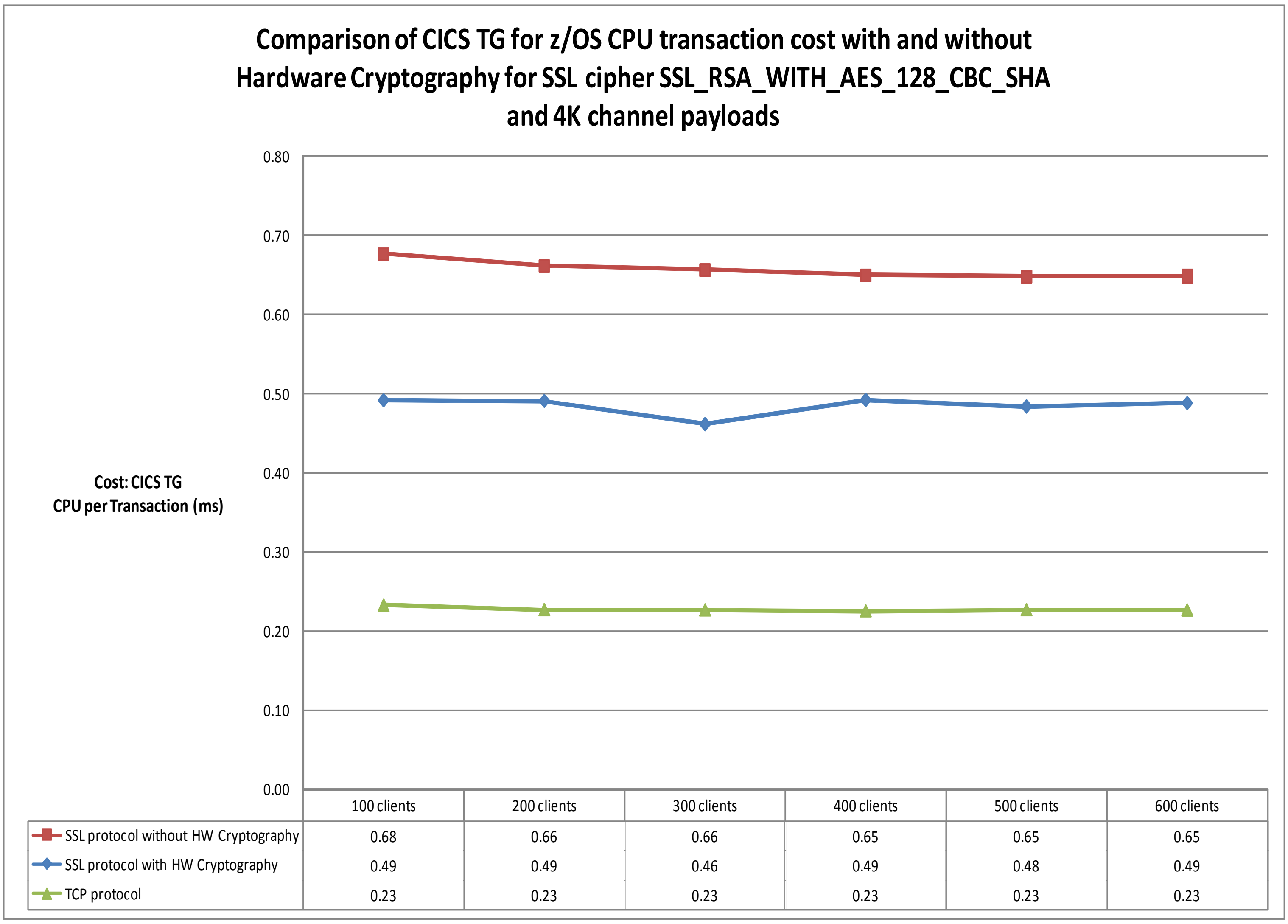
Observations

1. For this workload the CPU% usage is considerably less when using hardware cryptography for SSL requests.
2. The graph shows the CPU% usage without hardware cryptography plateaus with the higher number of clients. This is due to the system CPU resources being constrained.
3. With systems that are constrained, customers are recommended to add a zAAP/zIIP or replace a GCP with a zAAP/zIIP to offload CPU from the GCPs.

2.3 Cost of CICS TG CPU per Transaction for 4K channel payloads

These results compared the CICS TG for z/OS CPU transaction cost when running with or without hardware cryptography. The Java clients connected to CICS TG over SSL using the SSL cipher SSL_RSA_WITH_AES_128_CBC_SHA. The payloads were 4K channels.

For comparative purposes, the results shown in the graph include Java clients connected to CICS TG over the TCP (non-SSL) protocol.



	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
SSL protocol without HW Cryptography	0.68	0.66	0.66	0.65	0.65	0.65
SSL protocol with HW Cryptography	0.49	0.49	0.46	0.49	0.48	0.49
TCP protocol	0.23	0.23	0.23	0.23	0.23	0.23

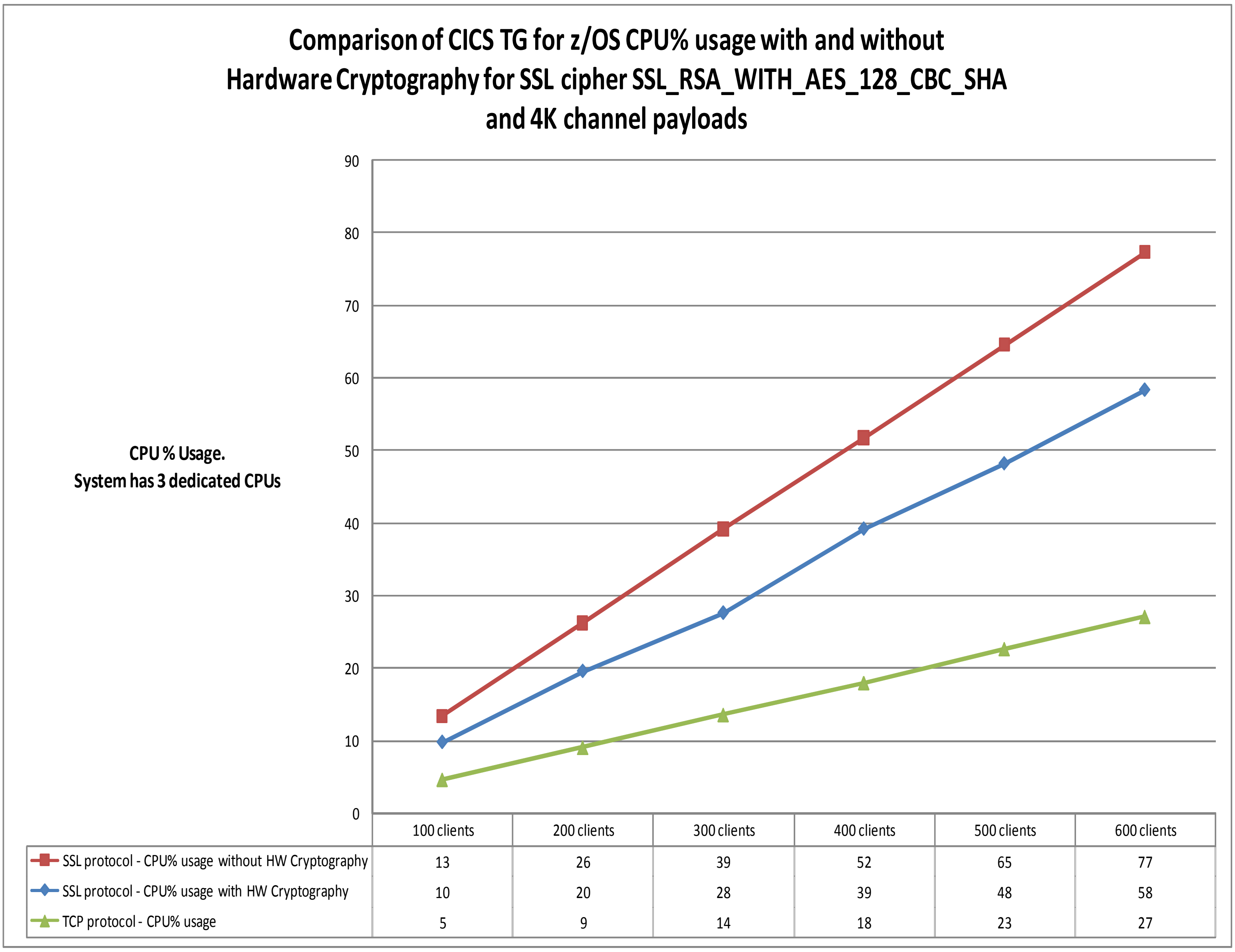
Observations

1. For smaller payloads there is still a GCP cost advantage to be gained by using a hardware cryptography CP to offload work.
2. CICS TG demonstrates good CPU transaction cost scalability for 100 to 600 clients using hardware cryptography.

2.4 CPU% usage of CICS TG for 4K channel payloads

These results measured the CICS TG CPU% usage with and without hardware cryptography enabled. The Java clients connected to CICS TG over SSL using the SSL cipher SSL_RSA_WITH_AES_128_CBC_SHA. The payloads used were 4K channels.

For comparative purposes, the results shown in the graph include Java clients connected to CICS TG over the TCP (non-SSL) protocol.



	100 clients	200 clients	300 clients	400 clients	500 clients	600 clients
🟥 SSL protocol - CPU% usage without HW Cryptography	13	26	39	52	65	77
◆ SSL protocol - CPU% usage with HW Cryptography	10	20	28	39	48	58
▲ TCP protocol - CPU% usage	5	9	14	18	23	27

Observations

1. Even for small payloads such as 4K channels, hardware cryptography is shown to save on CPU usage.

Conclusions

Customers should consider the following:

- Which SSL ciphers meet their needs. Stronger ciphers utilize more system resources than weaker ones and can result in slower response times.
- Using a zAAP or zIIP to offload work from GCPs. Less work on GCPs saves costs and reduces GCP charges. CICS Transaction Gateway is a Java application and so allows a very large proportion of the CPU usage to be offloaded from the GCPs to a zAAP or zIIP. See the examples shown in “1.6 zAAP offload with CICS TG for z/OS and 32K COMMAREA payloads“ on page 14 and “1.7 zAAP offload with CICS TG for z/OS and 4K COMMAREA payloads“ on page 15.
- Specify -Xcompressedrefs for 64-bit CICS Transaction Gateways to improve heap efficiency (see “Configuration“ on Page 5). Further information on this parameter can be found in the IBM SDK Java Technology Edition Knowledge Center.
- Using a hardware cryptography CP to save on GCP costs for:
  - Encryption and decryption of payloads
  - SSL handshaking (not covered in this report)

Notes:

1. Analysis of other payload sizes, different ciphers, or different hardware, have not been completed at this time, so there is no guarantee that equivalent observations will be seen in other configurations.
2. Due to the effects on system performance of machine hardware, levels of software configuration and payload, equivalent observations might not be seen on other systems.