CICS Transaction Gateway for z/OS V9.2 Performance summary

Version 1.0

Jan 2020

Aayush N Maheshwarkar

CICS Transaction Gateway

IBM India Pvt Ltd

India Software Development Labs

Bengaluru, Karnataka, India

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 V9.2. The information is not intended as the specification of any programming interfaces that are provided by CICS Transaction Gateway for z/OS 9.2 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 9.2.

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 that is contained in this report has not been submitted to any formal IBM test and is distributed “as is”. 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 that is contained in this report was measured in a controlled environment and results that are obtained in other environments may vary significantly.

Trademarks and service marks

Overview

This document contains performance measurements for CICS Transaction Gateway (CICS TG) for z/OS V9.2 used along with CICS Transaction Server (CICS TS) for z/OS V5.4.

The report looks at CPU usage for RESTFUL JSON webservice workload, including zIIP offload for Gateway daemon. It also describes the memory requirements of CICS TG.

CICS Transaction Gateway and CICS Transaction Server were collocated (same LPAR and TCPIP stack), by using IPIC connectivity. The measurements were taken by using the following configuration:

Hardware

- IBM System z: z13 2964-799 model NE1.
- LPAR with 3 dedicated GCPs.
- LPAR with 2 zIIP – where specified.
- IBM System x: Intel(R) Xeon(R) CPU X5670 @ 2.93 GHz.

Software

- z/OS version: V2R2.
- CICS TG v9.2.
- CICS TS Version: 5.4.
- IBM 64-bit SDK for z/OS Java Technology Edition, Version 8 SR5

Workload

The workload simulation ran on an IBM System x machine that is running SUSE Linux Enterprise Server 12-SP3, using the CICS TG Java base classes to drive RESTFUL JSON webservice requests containing non-null payload data, thus avoiding null-stripping optimizations.

The CICS Transaction Server applications (one for COMMAREA requests, and one for channel requests with a single container) that received the ECI requests simply returned the payload after altering the last byte to hex '5B'.

Configuration

PROTOCOL	REGION	MEMLIMIT	HEAP(Xmx)
IPIC	600M	20G	2048M

Terminology

GCP / CPU	-IBM System z General Purpose CPU
zIIP	-IBM System z INTEGRATED INFORMATION PROCESSOR
CPU %	-Percentage of CPU time that is used by transactions running on general-purpose processors
Cost per transaction (ms)	-CPU usage per transaction, in milliseconds
TPS	-Number of Transactions Per second
CICS TG	-IBM CICS Transaction Gateway for z/OS
CICS TS	-IBM CICS Transaction Server for z/OS
IPIC	-Internet Protocol (IP) interconnectivity
RMF	-Resource Measurement Facility
CC	- Channels and Containers

Scenario 1: Memory Requirements for a 64-bit Gateway daemon

CICS Transaction Gateway v9.2 on z/OS supports only 64-bit Java. Table1 describes a base suggestion about how to evaluate the necessary storage requirements. This table represents a guideline, not an accurate formula,

Storage Type	Usage
Native	3 MB per Thread
Java Heap	324 KB per IPIC Session
Java Heap	16 KB per Thread
Java Heap	12000 KB core Gateway Daemon

Table 1: Memory requirements for CICS TG

and the values can change depending on environmental factors, such as operating system release or Language Environment runtime options.

The Java heap can be modified at the Gateway daemon start by using the **-Xmx** JVM option and this is typically increased for production configurations. The default heap size that is specified by the CICS Transaction Gateway is 128 MB.

The JVM is a z/OS UNIX process that runs in its own Language Environment enclave. The JVM Heap, control blocks, and other data areas are handled by the Language Environment heap and it is governed by the Language Environment HEAP runtime option. Language Environment stack is allocated per thread with an initial size and can grow, according to the ANYHEAP runtime option values.

JVM is constructed in its own Language Environment enclave, created by the Language Environment pre-init module. The JVM is a z/OS UNIX process. The JVM Heap, control blocks, and other data areas are handled

by Language Environment heap and it is governed by the Language Environment HEAP runtime option.

The JVM Heap and default storage that is used by the Gateway daemon is all managed by Language Environment. Consider the z/OS JCL parameter **MEMLIMIT**. It sets the limit on how much virtual storage above the bar the gateway can use. If you do not set **MEMLIMIT**, the system default is 0, which means that no address space can use virtual storage above the bar. You can set an installation default **MEMLIMIT** through SMFPRM*xx* in PARMLIB or by using the IEFUSI exit.

You can also benefit from the **-Xcompressedrefs** Java option. It allows you to compress references. The JVM stores all references to objects, classes, threads, and monitors as 32-bit values. The **-Xcompressedrefs** and **-Xnocompressedrefs** command-line options enable or disable compressed references in a 64-bit JVM. Only 64-bit JVMs recognize these options.

A command-line option can be used with **-Xcompressedrefs** to allocate the heap specified with the **-Xmx** option, in a memory range of your choice. This option is **-Xgc:preferredHeapBase=<*address*>**, where **<*address*>** is the base memory address for the heap.

Larger heaps must be considered where IPIC connections are used, and the workload includes larger channel payloads. While increasing the ECI payload size, the heap size needs to be increased to avoid excessive garbage collection.

Gateway daemon MEMLIMIT with 64-bit Java

You can use following formula to calculate the MEMLIMIT setting for CICS Transaction Gateway.

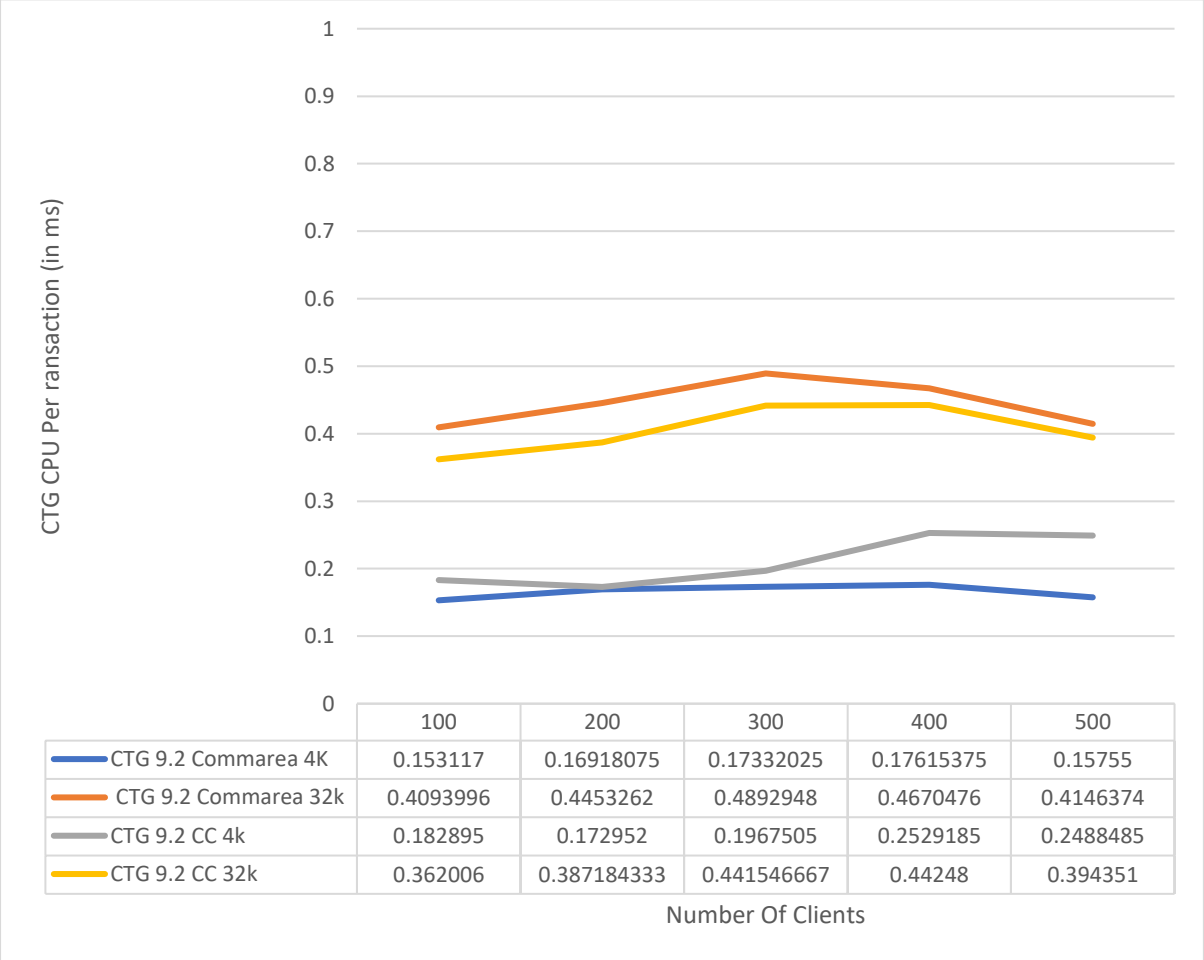
*MEMLIMIT=12 MB (core) + Heap size + (CM+WT threads) * 3 MB*

Scenario 2: RESTFUL JSON Webservices for Payloads 4K and 32K - CICS TG v9.2 COMMAREA and CC Payloads

This scenario compared the CICS TG CPU cost per transaction, Transactions per second and zIIP offload for COMMAREAs and channels by using payloads of 4k and 32K for RESTFUL JSON Webservice requests.

Cost of CICS TG CPU Per Transaction:

These results that are described in Illustration 1 are measured the Cost of CICS TG CPU per transaction in milliseconds (ms) for up to 500 clients.



	100	200	300	400	500
CTG 9.2 Commarea 4K	0.153117	0.16918075	0.17332025	0.17615375	0.15755
CTG 9.2 Commarea 32k	0.4093996	0.4453262	0.4892948	0.4670476	0.4146374
CTG 9.2 CC 4k	0.182895	0.172952	0.1967505	0.2529185	0.2488485
CTG 9.2 CC 32k	0.362006	0.387184333	0.441546667	0.44248	0.394351

Illustration 1: "CPU cost per transaction"

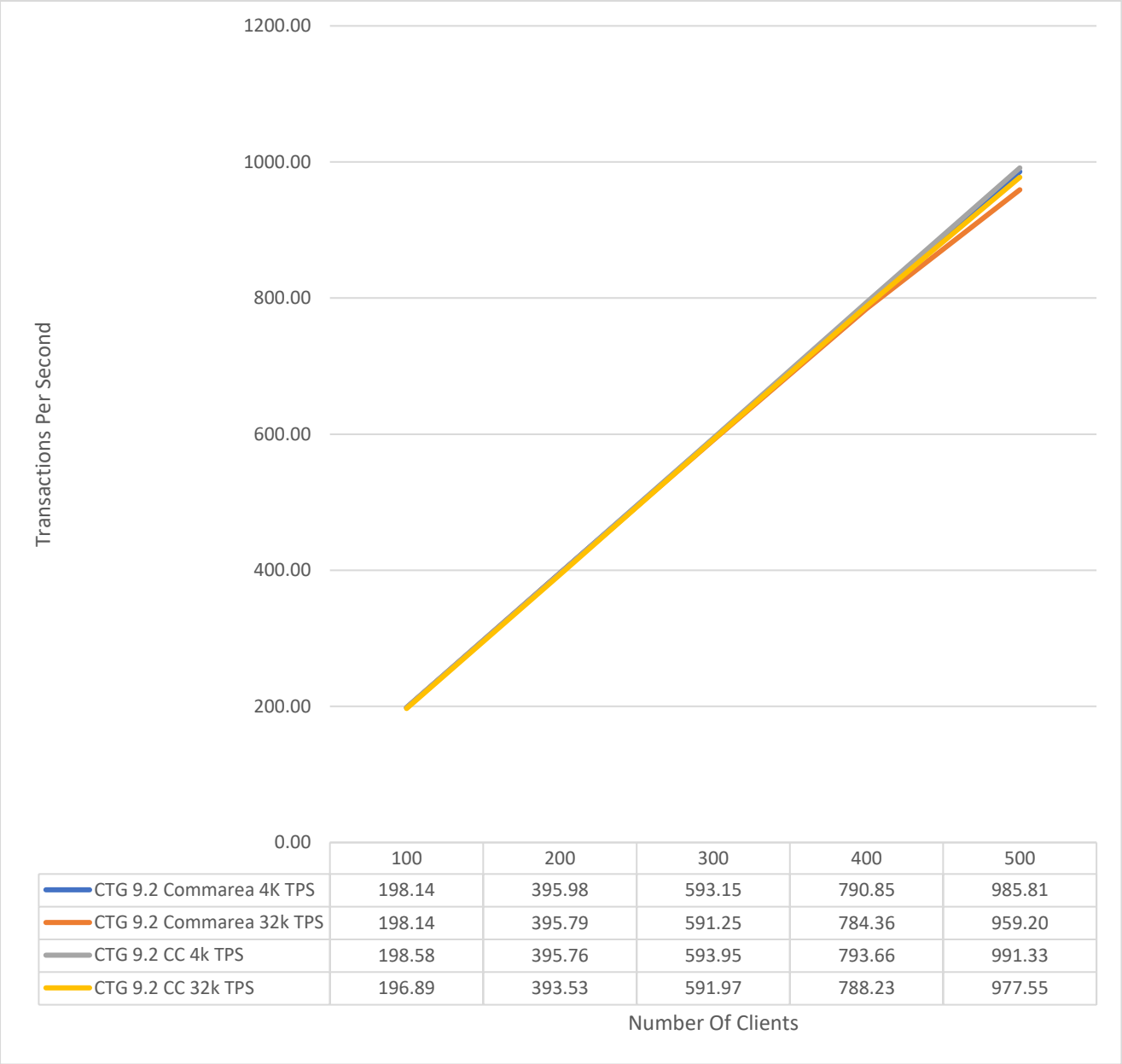
Observations:

The results in Illustration 1: "*CPU cost per transaction*" shows that the CPU

costs for channels versus COMMAREAs for small workloads up to 32K are similar. CPU cost per transaction approximately remains same for 100 to 500 clients. When writing new applications that are using the IPIC protocol into CICS TS consider by using channels for sending the payload to gain the performance benefits.

Transactions Per Second (TPS):

These results measured the CICS TG Transactions Per Second for up to 500 clients.



	100	200	300	400	500
CTG 9.2 Commarea 4K TPS	198.14	395.98	593.15	790.85	985.81
CTG 9.2 Commarea 32k TPS	198.14	395.79	591.25	784.36	959.20
CTG 9.2 CC 4k TPS	198.58	395.76	593.95	793.66	991.33
CTG 9.2 CC 32k TPS	196.89	393.53	591.97	788.23	977.55

Illustration 2: "Transactions Per Second"

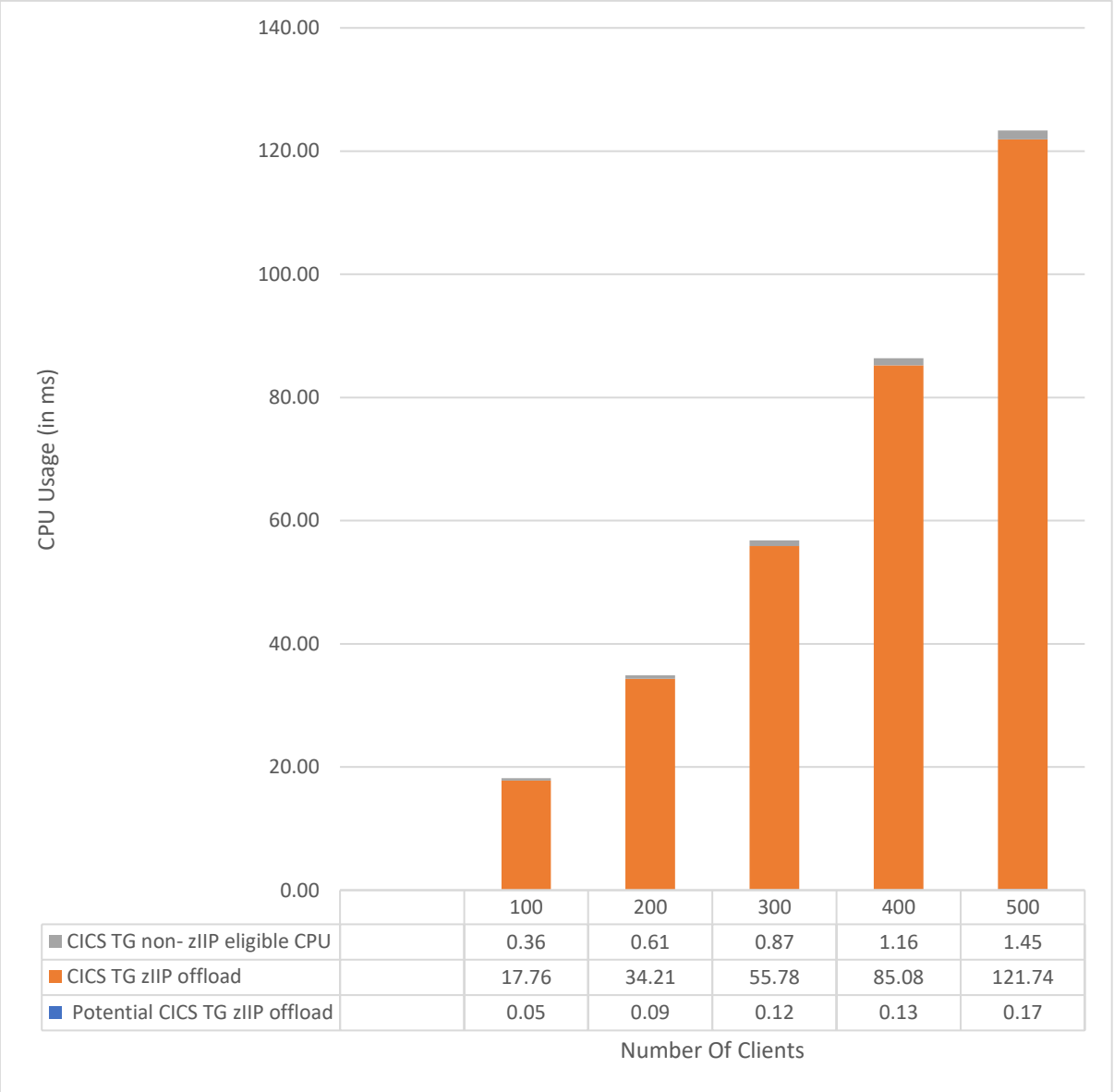
Observations:

The results in Illustration 2: “Transactions Per Second” shows that the Transaction per second for channels versus COMMAREAs for small workloads with payload up to 32K are similar. With number of clients increases, TPS increases linearly for both 4K and 32K payload for 100 to 500 clients.

zIIP offload with CICS TG v9.2 for z/OS:

These results measure the zIIP offload with increasing clients that are ranging from 100 - 500. Three 100% dedicated GCPs and two zIIP processors were used, so the RMF reports can show CPU usage above 100%.

For 4K CC:



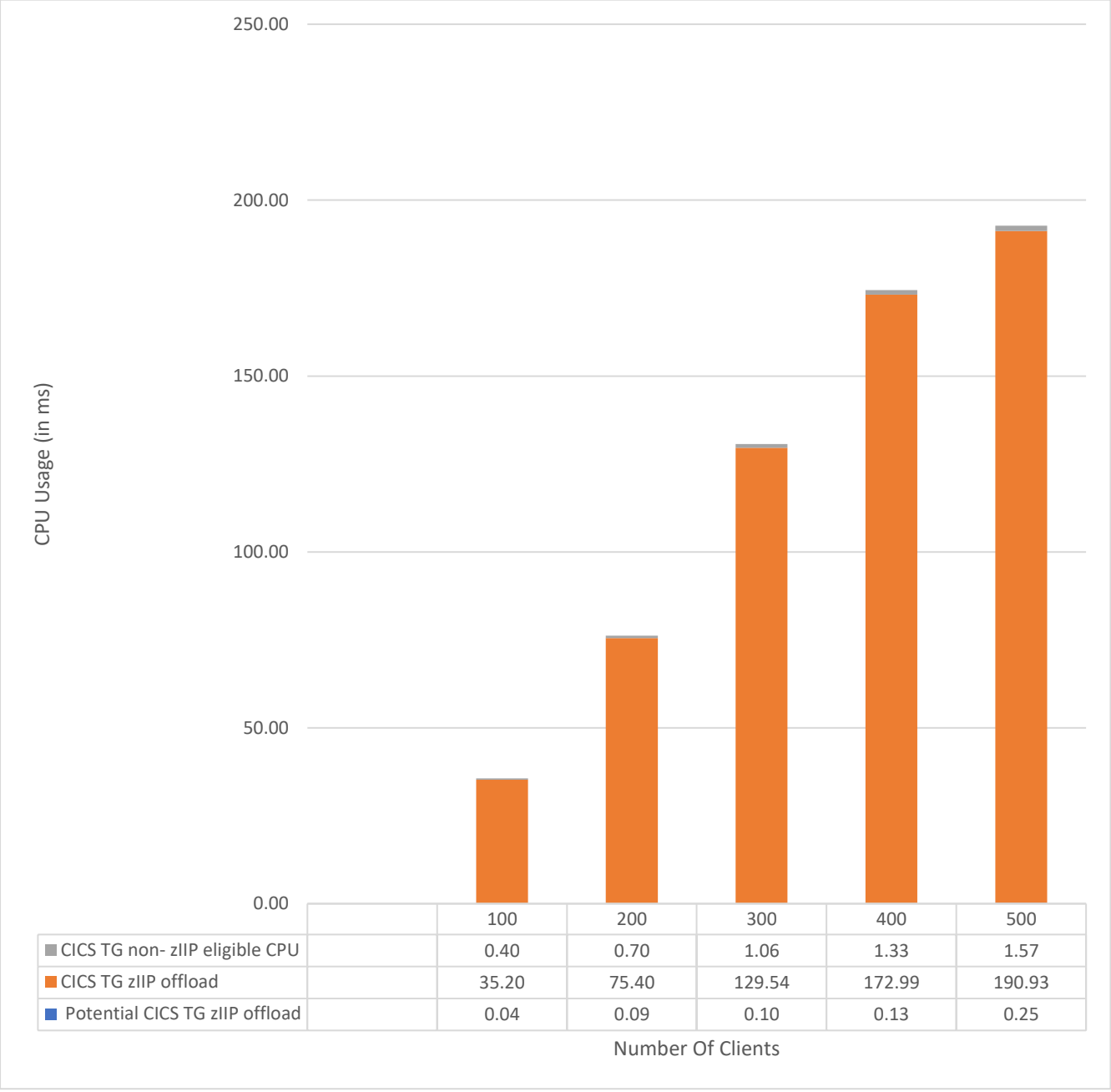
	100	200	300	400	500
CICS TG non- zIIP eligible CPU	0.36	0.61	0.87	1.16	1.45
CICS TG zIIP offload	17.76	34.21	55.78	85.08	121.74
Potential CICS TG zIIP offload	0.05	0.09	0.12	0.13	0.17

Illustration 3: “CPU usage and zIIP offload for 4K CC clients”

Observations:

The results in Illustration 3: “CPU usage and zIIP offload for 4K CC clients” shows that two zIIP processors were able to offload almost all the CPU required for transaction execution from the GCPs. The CPU usage value increases with the increase in number of clients.

For 32K CC:



	100	200	300	400	500
CICS TG non- zIIP eligible CPU	0.40	0.70	1.06	1.33	1.57
CICS TG zIIP offload	35.20	75.40	129.54	172.99	190.93
Potential CICS TG zIIP offload	0.04	0.09	0.10	0.13	0.25

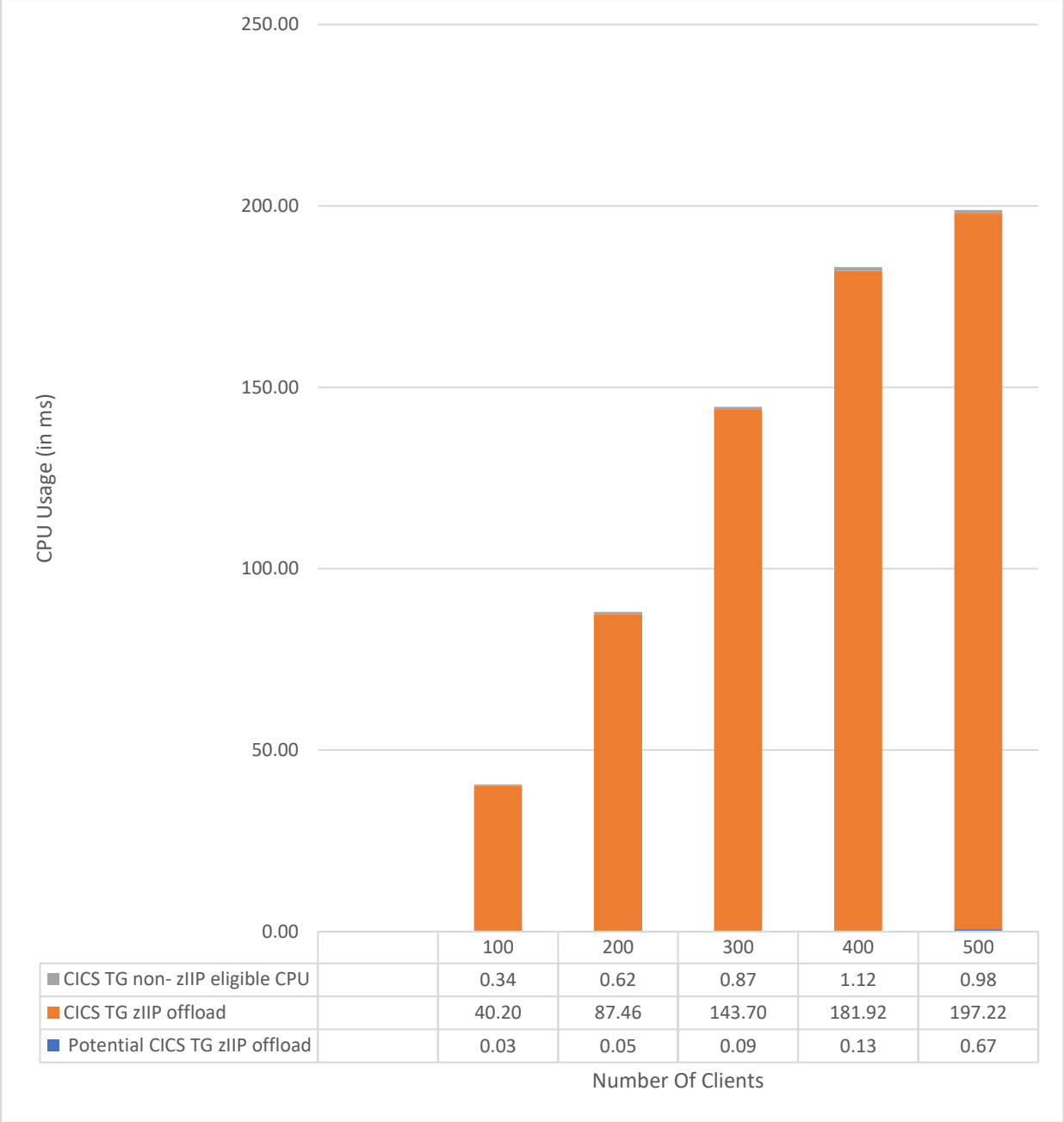
Illustration 4: “CPU usage and zIIP offload for 32K CC clients”

Observations:

The results in Illustration 4: “CPU usage and zIIP offload for 32K CC clients” shows that two zIIP processors were able to offload almost all the CPU

required for transaction execution from the GCPs even though the overall CPU usage increases with payload.

For 4K COMMAREA:



	100	200	300	400	500
CICS TG non- zIIP eligible CPU	0.34	0.62	0.87	1.12	0.98
CICS TG zIIP offload	40.20	87.46	143.70	181.92	197.22
Potential CICS TG zIIP offload	0.03	0.05	0.09	0.13	0.67

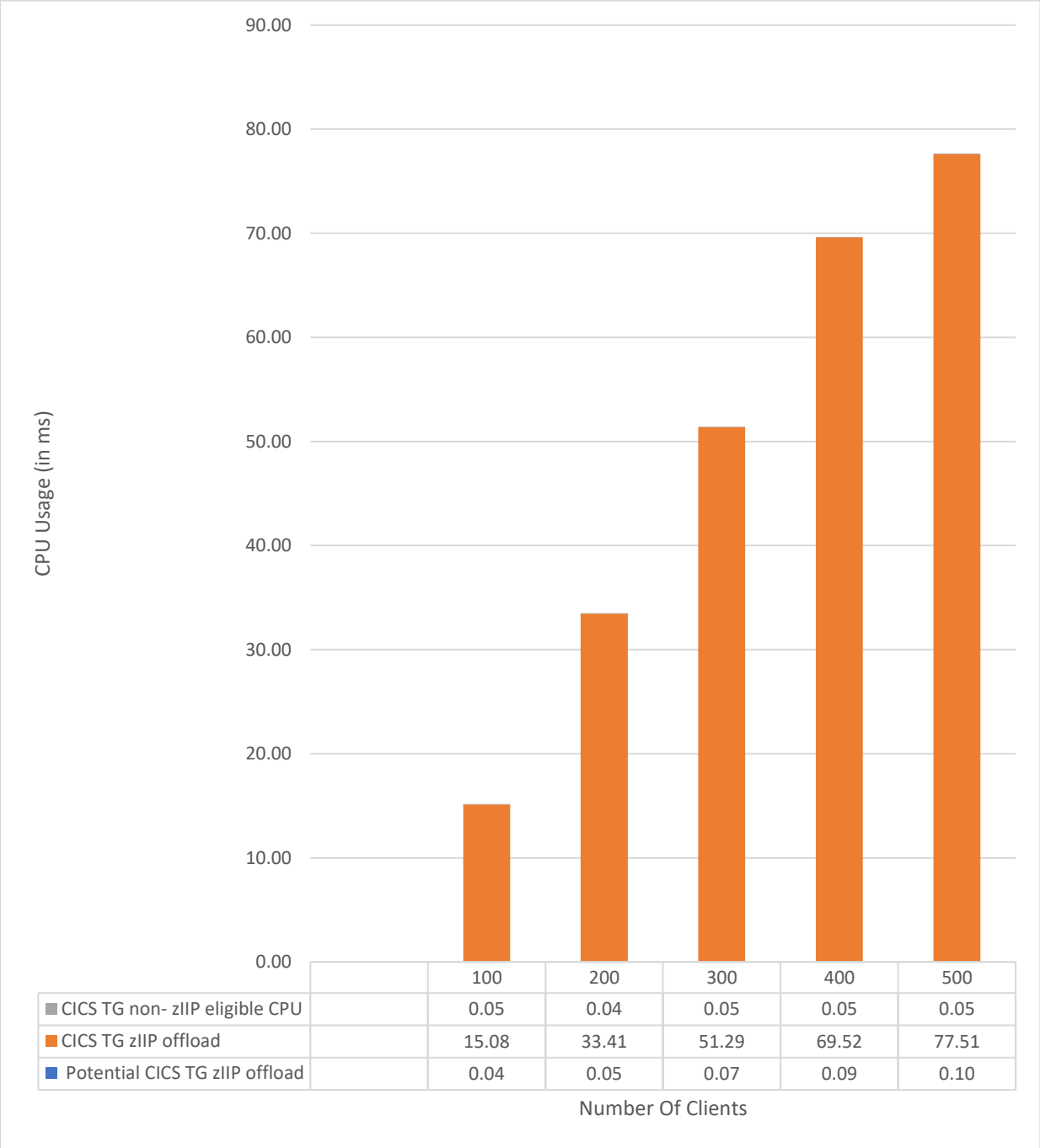
Illustration 5: "CPU usage and zIIP offload for 4K COMMAREA clients"

Observations:

The results in Illustration 5: "CPU usage and zIIP offload for 4K COMMAREA clients" shows that two zIIP processors were able to offload almost all the CPU required for transaction execution from the GCPs even though the

overall CPU usage increases with payload. The CPU usage value and zIIP offload also increases with the increase in number of clients.

For 32K COMMAREA:



	100	200	300	400	500
CICS TG non- zIIP eligible CPU	0.05	0.04	0.05	0.05	0.05
CICS TG zIIP offload	15.08	33.41	51.29	69.52	77.51
Potential CICS TG zIIP offload	0.04	0.05	0.07	0.09	0.10

Illustration 6: "CPU usage and zIIP offload for 32K COMMAREA clients"

Observations:

The results in Illustration 6: "CPU usage and zIIP offload for 32K COMMAREA clients" shows that two zIIP processors were able to offload

almost all the CPU required for transaction execution from the GCPs even though the overall CPU usage increases with payload. The non-zIIP eligible CPU values are minimal with higher COMMAREA payloads.

Scenario 3: Comparing Performance of CICS TG 9.1 with CICS TG 9.2:

This scenario compared the CICS TG CPU cost per transaction, Transactions per second and zIIP offload for RESTFUL JSON Webservice requests across CICS TG v9.2 and CICS TG v9.1.

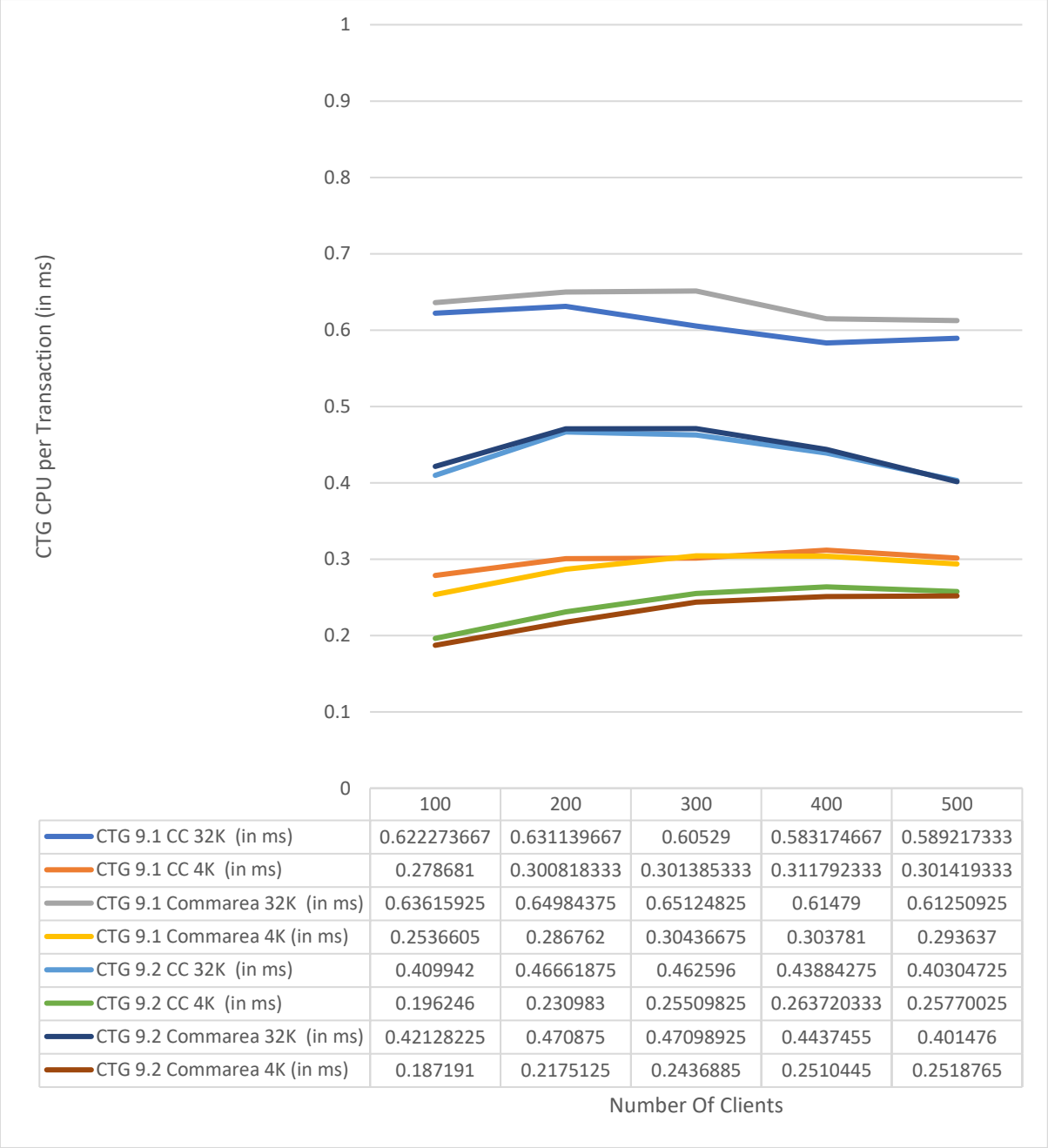
SSL_RSA_WITH_AES_256_CBC_SHA cipher used to encrypt and decrypt the data that is sent between RESTFUL JSON webservices clients and CICS TG.

Four payload sizes were used for comparison:

1. 4K CC
2. 32K CC
3. 4K COMMAREA
4. 32K COMMAREA

Cost of CICS TG CPU Per Transaction:

These results are measured the Cost of CICS TG CPU per transaction in milliseconds (ms) for different clients that are varying in the range 100 - 500 clients each using different payload options of 4K CC, 32K CC, 4K COMMAREA, and 32K COMMAREA.



	100	200	300	400	500
CTG 9.1 CC 32K (in ms)	0.622273667	0.631139667	0.60529	0.583174667	0.589217333
CTG 9.1 CC 4K (in ms)	0.278681	0.300818333	0.301385333	0.311792333	0.301419333
CTG 9.1 Commarea 32K (in ms)	0.63615925	0.64984375	0.65124825	0.61479	0.61250925
CTG 9.1 Commarea 4K (in ms)	0.2536605	0.286762	0.30436675	0.303781	0.293637
CTG 9.2 CC 32K (in ms)	0.409942	0.46661875	0.462596	0.43884275	0.40304725
CTG 9.2 CC 4K (in ms)	0.196246	0.230983	0.25509825	0.263720333	0.25770025
CTG 9.2 Commarea 32K (in ms)	0.42128225	0.470875	0.47098925	0.4437455	0.401476
CTG 9.2 Commarea 4K (in ms)	0.187191	0.2175125	0.2436885	0.2510445	0.2518765

Illustration 7: "CPU cost per transaction comparison"

Observations:

The results in Illustration 7: "*CPU cost per transaction comparison*" shows that the CPU costs for channels and COMMAREA payloads for RESTFUL JSON webservices clients with payload size off 4K and 32K. CPU cost per transaction that is measured with varying loads from 100 - 500 clients. CPU cost per transaction with CICS TG v9.2 is considerably reduced when compared with CICS TG v9.1.

Transactions Per Second (TPS):

These results measured the CICS TG Transactions Per Second for different client workload that are varying in the range 100 - 500 clients with different transaction payload like 4K CC, 32K CC, 4K COMMAREA and 32K COMMAREA.

	100	200	300	400	500
CTG 9.1 CC 32K	195.74	386.35	560.18	684.32	677.86
CTG 9.1 CC 4K	197.56	392.96	586.30	782.16	974.23
CTG 9.1 Commarea 32K	194.76	385.64	562.39	649.02	652.10
CTG 9.1 Commarea 4K	196.63	391.74	587.86	782.64	971.29
CTG 9.2 CC 32K	196.62	391.29	585.02	776.25	961.62
CTG 9.2 CC 4K	197.32	393.39	590.30	786.57	981.36
CTG 9.2 Commarea 32K	196.67	391.36	584.68	777.01	958.71
CTG 9.2 Commarea 4K	196.90	393.20	589.95	785.41	981.43

Illustration 8: "Transaction per second comparison"

Observations:

The results in Illustration 8: "*Transaction per second comparison*" shows the Transaction Per Second for channels and COMMAREA payloads for RESTFUL JSON webservices clients with payload size off 4K and 32K. Transaction per second with CICS TG v9.2 is more when compared with CICS TG v9.1 with higher number of concurrent clients workload.

zIIP offload comparison for CICS TG v9.1 and CICS TG v9.2:

These results measure the zIIP offload value and percentage for clients ranging in the range 100 - 500. Therefore, three 100% dedicated GCPs were available alongside two zIIP processors, the RMF reports can show CPU usage above 100%.

For 4K CC:

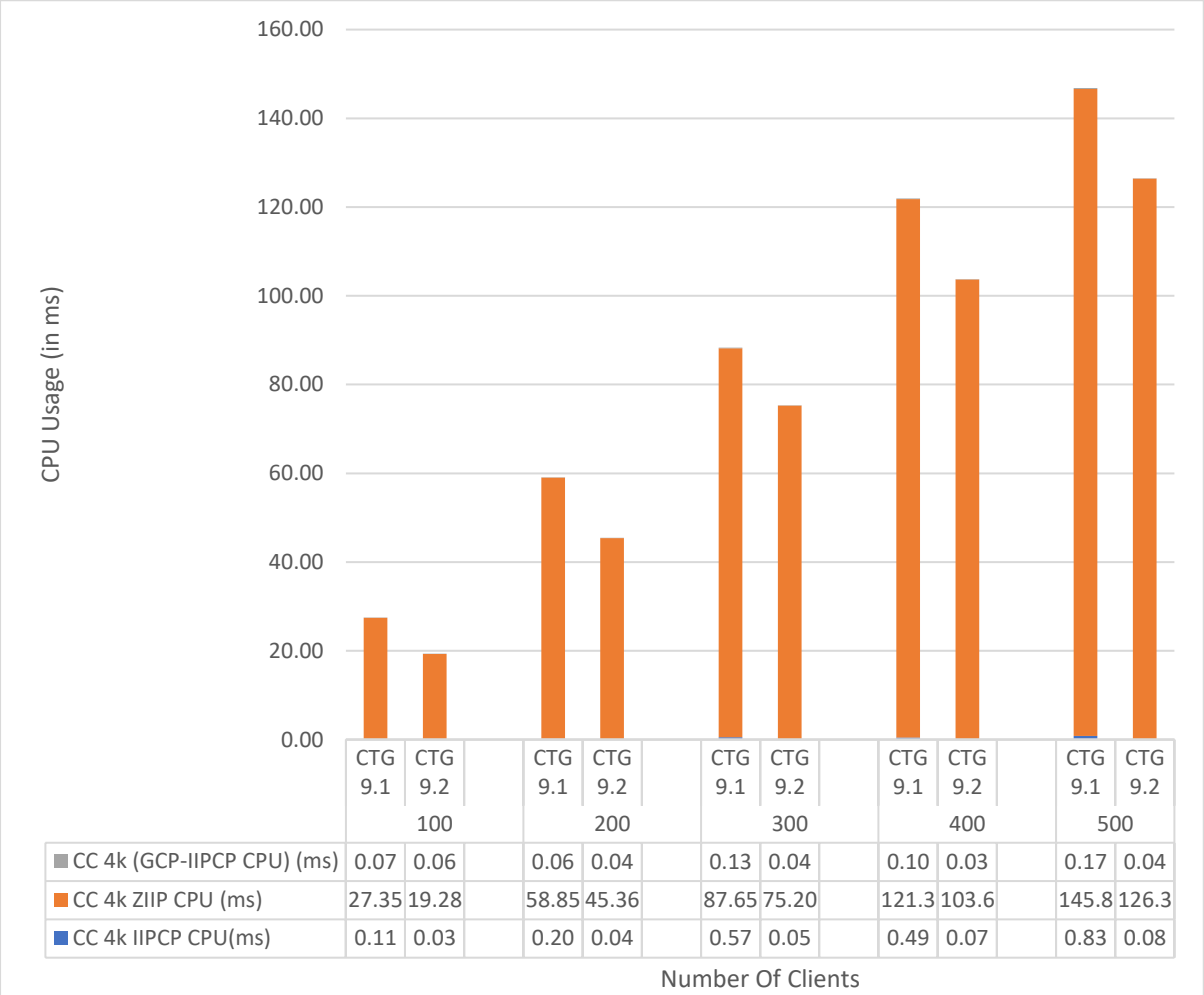


Illustration 9: "zIIP offload comparison for 4K CC"

Observations:

The results in Illustration 9: "zIIP offload comparison for 4K CC" shows the zIIP CPU offload for channels payload of 4K size for RESTFUL JSON webservices client applications with varying workload in the range 100 - 500 clients. The non zIIP eligible CPU (GCP – IIPCP) is further reduced with CICS TG v9.2. CICS TG v9.2 has more zIIP CPU eligible code when compared with CICS TG v9.1. This signifies less cost for GPU with CICS TG v9.2.

For 32K CC:

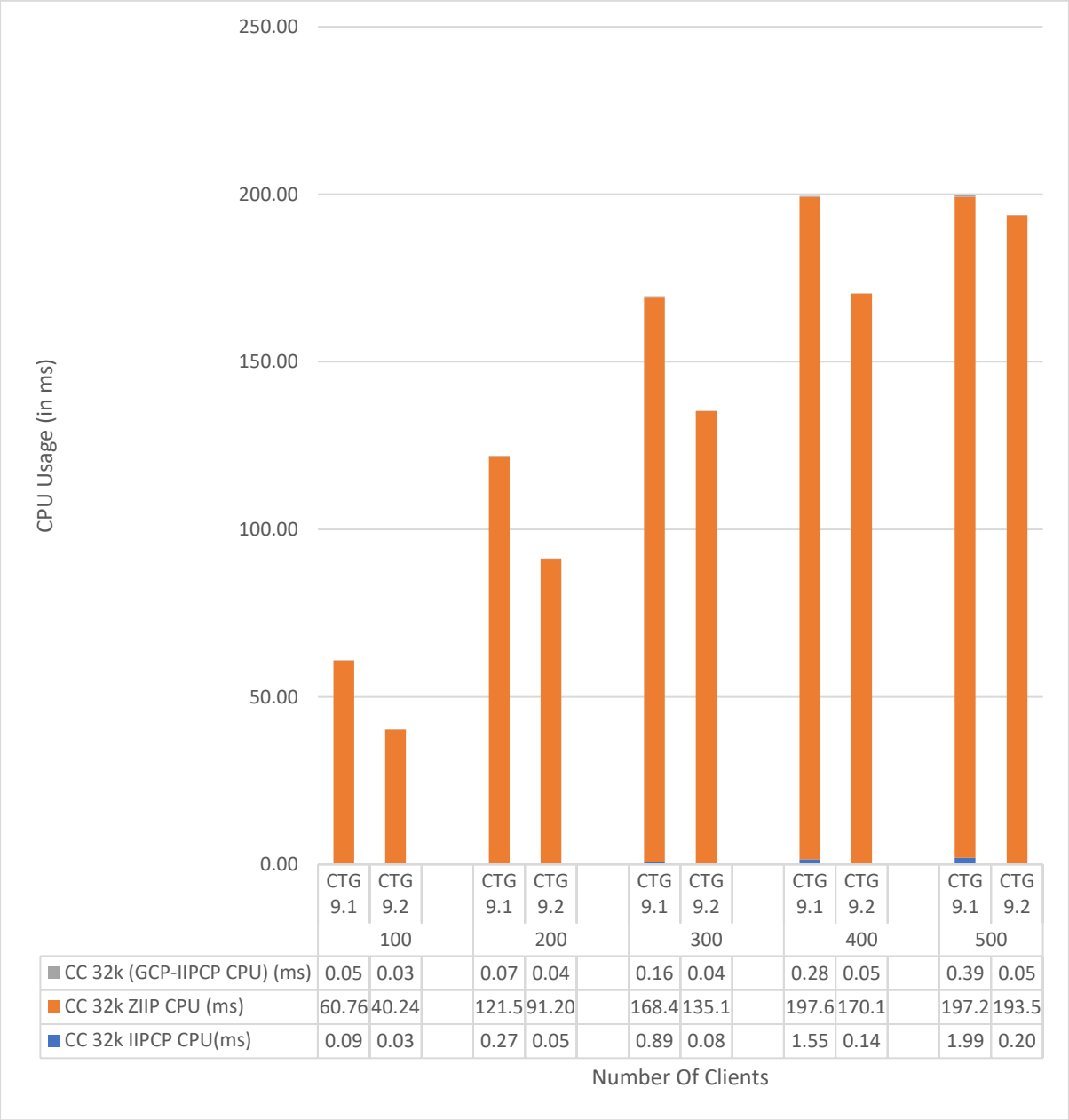


Illustration 10: "zIIP offload comparison for 32K CC"

Observations:

The results in Illustration 10: "zIIP offload comparison for 32K CC" shows the zIIP CPU offload for channels payload of 32K size for RESTFUL JSON webservices client applications with varying workload in the range 100 - 500 clients. The non zIIP eligible CPU (GCP – IIPCP) is minimal with CICS TG v9.2. The overall CPU usage is also reduced with CICS TG v9.2.

For 4K COMMAREA:

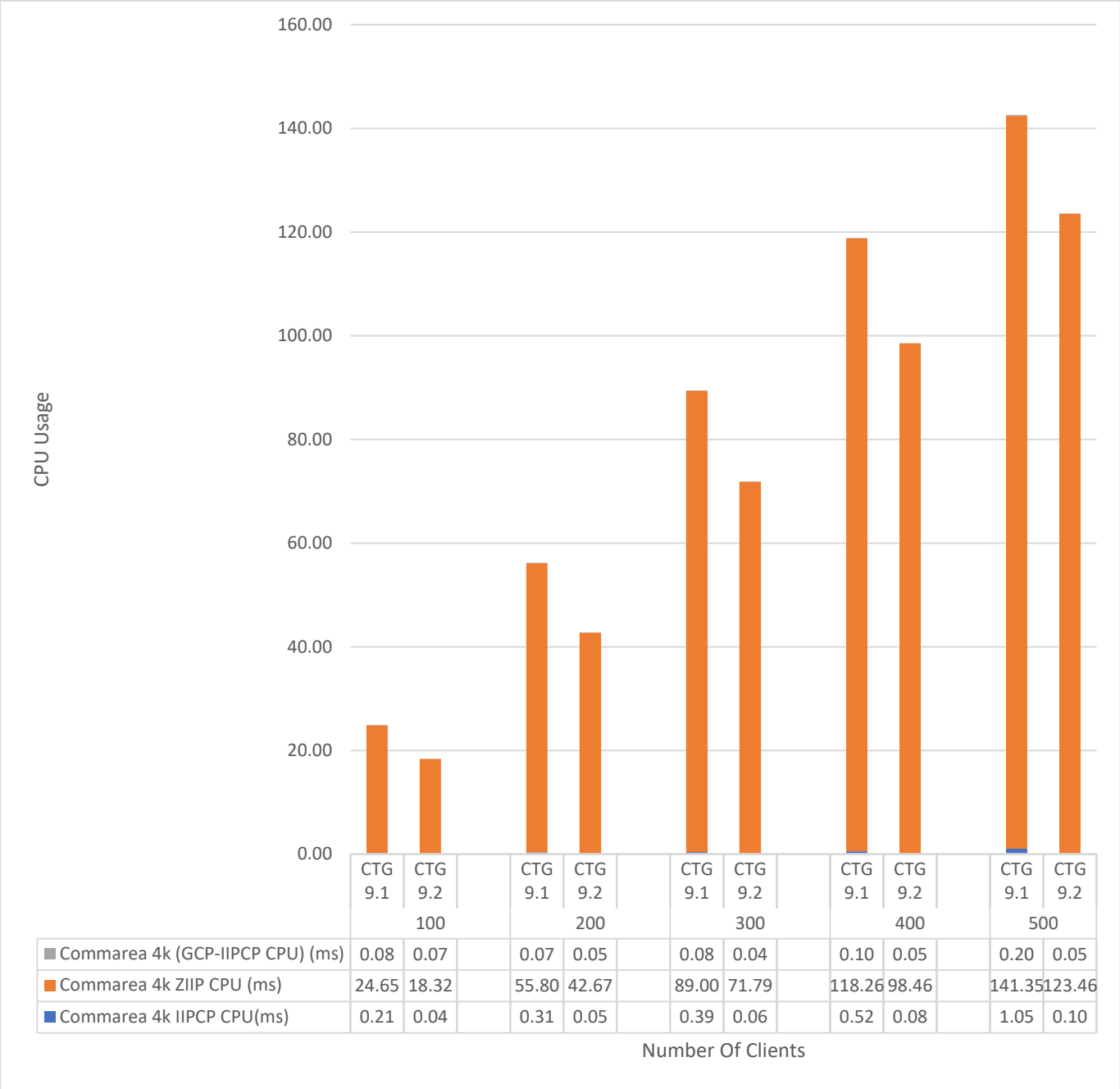
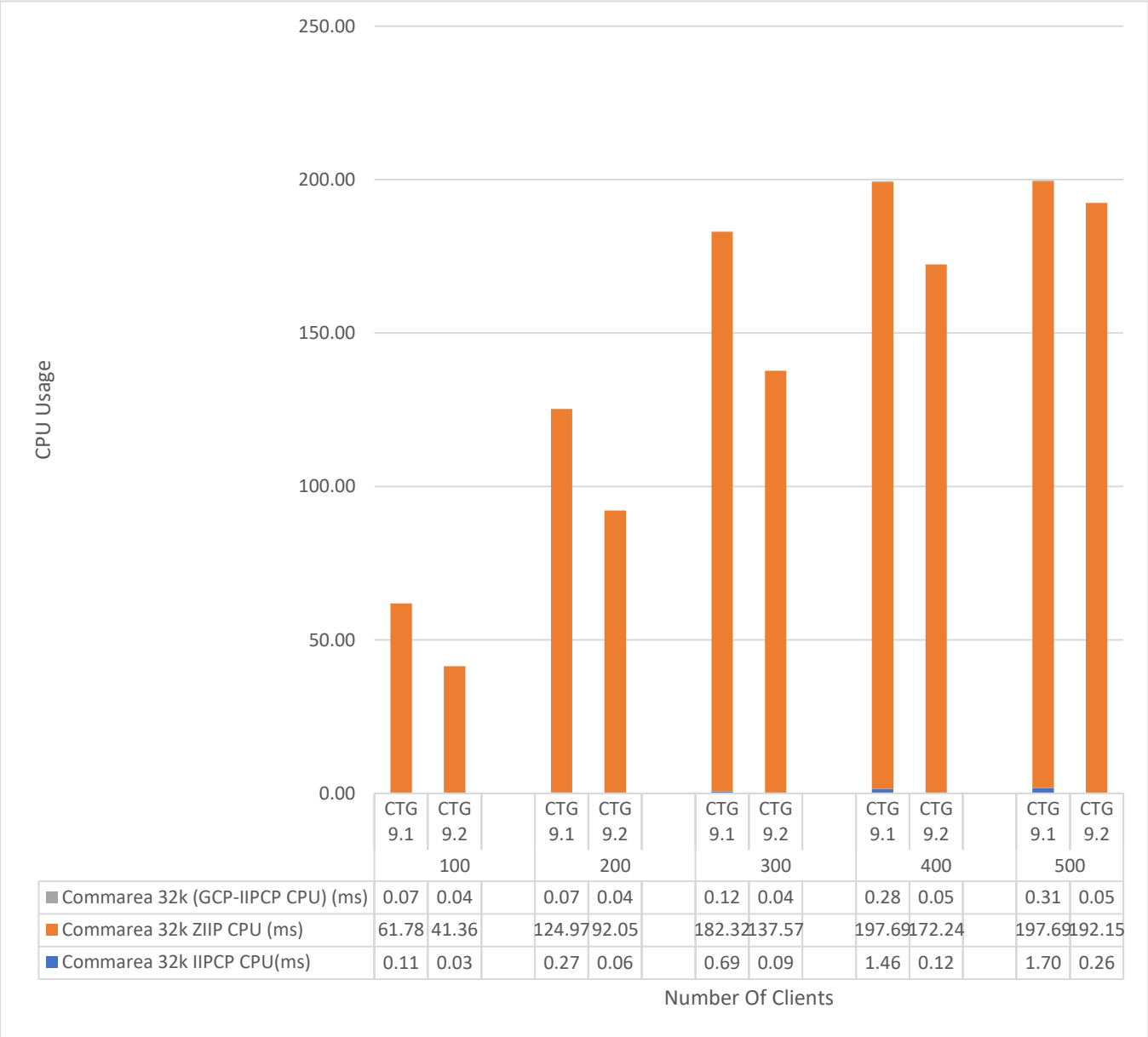


Illustration 11: "zIIP offload comparison for 4K COMMAREA"

Observations:

The results in Illustration 11: "zIIP offload comparison for 4K COMMAREA" shows the zIIP CPU offload for COMMAREA payload of 4K size for RESTFUL JSON webservices client applications with varying workload in the range 100 - 500 clients. The overall CPU usage and non zIIP eligible CPU (GCP – IIPCP) is less CICS TG v9.2 when compared with CICS TG v9.1.

For 32K COMMAREA:



	CTG 9.1	CTG 9.2	CTG 9.1	CTG 9.2	CTG 9.1	CTG 9.2	CTG 9.1	CTG 9.2	CTG 9.1	CTG 9.2
	100		200		300		400		500	
Commarea 32k (GCP-IIPCP CPU) (ms)	0.07	0.04	0.07	0.04	0.12	0.04	0.28	0.05	0.31	0.05
Commarea 32k ZIIP CPU (ms)	61.78	41.36	124.97	92.05	182.32	137.57	197.69	172.24	197.69	192.15
Commarea 32k IIPCP CPU(ms)	0.11	0.03	0.27	0.06	0.69	0.09	1.46	0.12	1.70	0.26

Illustration 11: "zIIP offload comparison for 32K COMMAREA"

Observations:

The results in Illustration 11: "zIIP offload comparison for 32K COMMAREA" shows the zIIP CPU offload for COMMAREA payload of 32K size for RESTFUL JSON webservices client applications with varying workload in the range 100 - 500 clients. The overall CPU usage and non zIIP eligible CPU (GCP – IIPCP) is less CICS TG v9.2 when compared with CICS TG v9.1.

Conclusions:

Customers should consider the following:

• When using IPIC, zIIPs can provide large benefits by offloading eligible work, thus potentially reducing the cost of running workloads.

• Channels (rather than COMMAREAs) provide greater benefits and flexibility and should be considered when writing new applications.

• The 64-bit Gateway daemon offers good scalability for both large payloads and large numbers of clients.

• When secured clients are used, CICS TG v9.2 gives higher TPS with better cost per transaction when compared with earlier versions.