

IBM CICS Performance Series: Web Services Performance in CICS TS V5.3

John Burgess

Ian Burnett

Martin Cocks



z Systems



IBM CICS Performance Series: Web Services Performance in CICS TS V5.3

This IBM® Redpaper™ publication describes a set of benchmarks comparing an IBM CICS® web services workload running in various configuration scenarios.

This paper starts with an IBM CICS Transaction Server (CICS TS) V5.2 benchmark running on an IBM zEnterprise® EC12 (zEC12), and shows the performance characteristics of that same workload running on a similarly configured IBM z13™ (z13). We then compare the same workload running on IBM CICS TS V5.3 and demonstrate the performance benefits of CICS TS V5.3 as compared with CICS TS V5.2 for various scenarios, including Secure Sockets Layer (SSL), Application Transparent Transport Layer Security (AT-TLS), and Hypertext Transfer Protocol (HTTP). Also covered is a comparison between CICS using SSL and CICS using AT-TLS for a configuration handling web service requests.

The intended audience for this paper is CICS systems programmers and capacity planners who have responsibility for evaluating potential benefits of migrating to a new release of CICS TS V5.3.

CICS TS configuration

For demonstration purposes, we have chosen the CICS configuration shown in Figure 1 on page 2 to use as a performance benchmark.

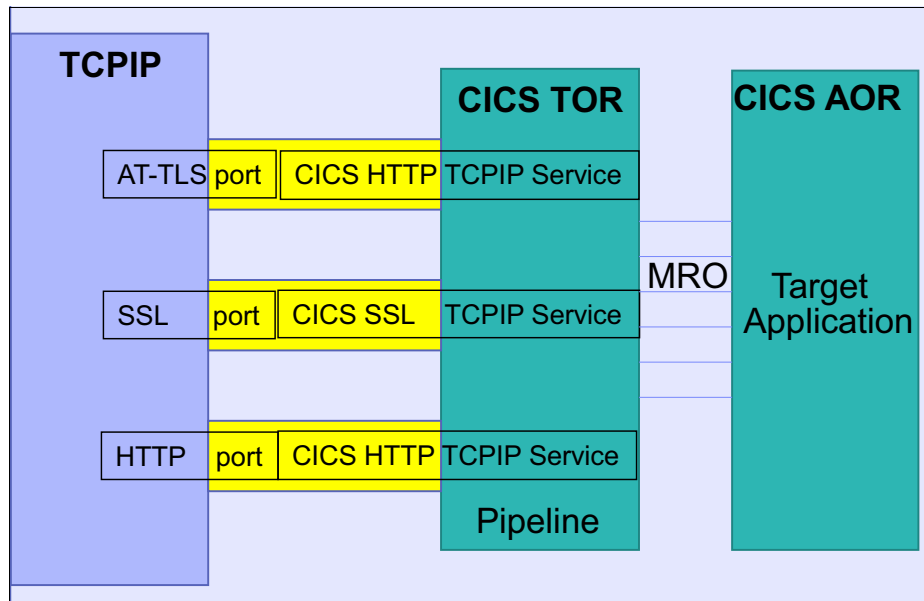


Figure 1 CICS TS configuration

CICS was set up with a CICS terminal-owning region (TOR) (an Internet Gateway-owning region or routing region), where all of the TCP/IP sessions connect using one of the three following CICS TCP/IP services:

- ▶ A CICS TCP/IP service configured for SSL sessions
- ▶ A CICS TCP/IP service configured for basic HTTP
- ▶ A CICS TCP/IP service configured for HTTP and associated with a port requiring AT-TLS

For all of these CICS TS V5.2 configurations, the web service request entered the CICS TOR and was handled by the CWXN (web attach) transaction to establish the context, and then by the CPIH (HTTP pipeline handler) transaction for the web service pipeline handling.

When using CICS TS V5.3, efficiencies in the code eliminated the need for the CWXN transaction when not using SSL. The benefit of this is described in "HTTP non-persistent session CICS TS V5.2 versus CICS TS V5.3".

The target program specified in the web service request was defined as being a remote program in a CICS application-owning region (AOR) in the same logical partition (LPAR) connected using a multi-region operation/cross-memory (MRO/XM) connection. The application program in this case has minimal business logic and only changes the received data and returns it.

For both SSL and AT-TLS, the same cipher was used: 000A - 168-bit SSL
SSL_RSA_WITH_3DES_EDE_CBC_SHA.

Hardware and software configuration

The hardware and software configuration is defined in this section.

Hardware

The hardware configuration was as follows:

- ▶ IBM zEnterprise EC12 model HA1 (machine type 2827)
- ▶ Crypto Express4 CCA Coprocessor (CEX4C)
- ▶ IBM z13 model NE1 (machine type 2964)
- ▶ Crypto Express5S CCA Coprocessor (CEX5C)
- ▶ LPAR with three dedicated CPs for system under test
- ▶ LPAR with three dedicated CPs for network driver
- ▶ Internal Coupling Facility (ICF)

Software

The software configuration was as follows:

- ▶ CICS Transaction Server V5.2 and V5.3
- ▶ IBM z/OS® V2R1
- ▶ IBM Workload Simulator for z/OS (WSim)

CICS configuration parameters

CICS configuration parameter settings were as follows:

- ▶ MN=ON
- ▶ MNPER=ON
- ▶ INTTR=ON
- ▶ SYSTR=ON
- ▶ SSLDELAY=0 (full handshake)
- ▶ SSLDELAY=600 (partial handshake)
- ▶ SEC=YES
- ▶ XTRAN=YES

For a full description of CICS configuration parameters, see *The system initialization parameter descriptions and summary* in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.3.0/com.ibm.cics.ts.sysdefinit.ion.doc/topics/dfha2_sitparm_descriptions.html

Target application program

In this benchmark, the routing region manages the web services pipeline and then executes the target application in another CICS region, connected using an MRO/XM connection. The application has little logic in it. It issues the EXEC CICS GET CONTAINER application programming interface (API) command to receive the incoming data structure (see Example 1), which has been converted from the original XML by the pipeline.

Example 1 Incoming data structure

```
01 RECEIVED-DATASTRUCTURE.
  02 JB1          PIC X(80) .
  02 JB2          PIC X(80) .
  02 JB3          PIC X(80) .
  02 JB4          PIC X(80) .
  02 JB5          PIC X(80) .
  02 JB6          PIC X(80) .
  02 JB7          PIC X(80) .
  02 JB8          PIC X(80) .
  02 JB9          PIC X(80) .
  02 JB10         PIC X(80) .
```

The application then builds a similar data structure, places it in the designated response container, and then issues the EXEC CICS RETURN API command.

The data flows back to the routing region, where the pipeline converts it back to XML and sends it back to the client.

Note: Given that the business logic in this application is insignificant, any performance improvements demonstrated in this paper are relevant to the system infrastructure only. This needs to be considered when mapping these improvements to other systems to predict CPU savings.

Configurations

This section describes the scenarios used for this web services benchmark. Each benchmark used the same CICS startup parameters and applications, but TCP/IP connectivity had various configurations, as listed in the following sections.

SSL full handshake

In this scenario, the web service request arrives over an SSL port. The connection is non-persistent, so when the request is complete, the connection is closed. The next request coming in needs to make a new connection and, because SSLDELAY=0 was specified in the CICS startup parameters, no SSL ID caching takes place, and a full SSL handshake needs to be executed. Full handshakes take place on the cryptographic coprocessor, and the activity can be seen in the IBM Resource Measurement Facility™ (RMF™) Monitor I report. CPU time for handshakes and encryption/decryption are charged to the S8 task control blocks (TCBs) in the CICS address space when using SSL.

SSL partial handshake

An SSL partial handshake takes place when the connection is non-persistent but the SSLDELAY parameter has been set to a value other than 0. In this case, SSLDELAY=600 was used, and so when the request coming in makes a new connection, it finds that an SSL ID for this connection has been cached by CICS. Therefore, a partial handshake can be used instead of a full handshake. Partial handshakes are performed in the software, so no activity appears on the cryptographic coprocessor.

SSL persistent connection

In this scenario, all connections are persistent, so all the requests arrive over already established connections. In this case, no handshakes other than for the first request takes place. The only requirement is encryption/decryption for the payload.

AT-TLS full handshake

AT-TLS is different from a CICS perspective, as compared with using SSL. With AT-TLS, TCP/IP handles the handshake and encryption/decryption rather than CICS. CICS accepts the request on a CICS HTTP listener rather than an SSL listener, because all of the security work has been done in the TCP/IP address space. The AT-TLS handshake has the same concept as the SSL handshake, in that no SSL ID is cached and a full handshake is required for each new connection. With AT-TLS, CPU usage moves from the CICS address space to the TCP/IP address space.

AT-TLS partial handshake

A partial handshake can be used for a new AT-TLS connection if an SSL ID has been previously cached for this connection. SSL IDs are cached by TCP/IP when using AT-TLS by specifying a nonzero GSK_V3_SIDCACHE_SIZE parameter in the AT-TLS policy. In this case, we used 1024 entries.

AT-TLS persistent connection

In this scenario, all of the connections are persistent, so all requests arrive over already established connections. In this case, no handshakes other than for the first request took place. All that is required is the encryption/decryption for the payload, which was executed by the TCP/IP address space.

HTTP non-persistent connection

In this scenario, the requests arrive over non-persistent connections. No SSL or AT-TLS security takes place. Each new request makes a new HTTP connection.

HTTP persistent connection

In this scenario, all of the requests arrive over persistent connections, so the cost of making and closing connections is eliminated. No SSL or AT-TLS security takes place.

Methodology

IBM Workload Simulator for z/OS (WSim) was used to drive transactions in a CICS region running on a different LPAR from the system under test. These transactions used the **EXEC CICS INVOKE SERVICE** API command to execute a web service in the target CICS region. The LPAR supporting the requesting application was connected to the LPAR supporting the server CICS region using a 10 Gb Open Systems Adapter (OSA) card.

The workload was allowed to reach a steady transaction rate, and then RMF and CICS statistics were used to collect performance data for the measurement period. The rate at which work is driven into CICS is varied by adjusting the WSim *user think time interval* (UTI). The UTI value represents the delay between a simulated client receiving a response, and then sending the next request into CICS. Clearly, a large think time results in a low rate of transactions in CICS. Reducing the UTI increases the rate at which work is driven into the CICS environment. Following the measurement period, the UTI was reduced so that the transaction rate increased. After another stabilization interval, data for another measurement period was collected.

This process was repeated five times so that we had a range of transaction rates for each configuration. The same set of think times were used for each configuration, resulting in the same set of transaction rates for each comparable measurement point. The data from all of these points is included in the tables in this paper.

Tools

During the benchmark sampling period, the following tools were used.

RMF Monitor I

RMF Monitor I was used to record system resource usage, including CPU, DASD, and storage. It was also used with the Workload Manager (WLM) configuration to record the CPU, transaction rates, and response times for CICS service classes and report classes. System Management Facility (SMF) records 70 - 79 are written on an interval basis and can be post-processed using the RMF utility program, ERBRMFPP.

CICS TS statistics

CICS statistics were used to monitor and report CICS resource usage including CPU, storage, file accesses, and so on.

CICS interval statistics have most of the counters reset at the start of the interval so that any resource consumption reported only relates to the measurement period being observed. Interval statistics can be activated by using the **CEMT SET STATISTICS** command. However, when you set this, the first interval can be adjusted to a shorter time so that all intervals are synchronized to the **STATEOD** parameter. For example, if you were to use the **CEMT** command to set the interval to 15 minutes at 10 past the hour, the first interval would expire in 5 minutes so that all future intervals line up on 15-minute wall clock boundaries. The values in this first report could also be associated with a much longer period, depending on the time of the last reset.

Another alternative to using interval statistics is to use **CEMT** to reset the counters and then, at the end of the measurement period, use **CEMT** to record all of the statistics. Resetting the statistics requires a change of state from ON to OFF or from OFF to ON. To make sure this occurs, the following commands (see Example 2 on page 7) are an example of resetting the statistics in the AOR in this configuration.

Example 2 CEMT command for resetting statistics

```
F CICS001,CEMT SET STAT OFF RESET
F CICS001,CEMT SET STAT ON RESET
Measurement period is between the RESET and the RECORD
F CICS001,CEMT PERFORM STAT ALL RECORD
```

Regardless of whether statistics is ON or OFF, when a **PERFORM STAT ALL RECORD** command is issued, a statistics record will be written.

CICS statistics are written as SMF 110 subtype 2 records and can be post-processed using the CICS statistics utility program, DFHSTUP, or CICS Performance Analyzer (CICS PA).

Terms used

This section describes the terms used in the results tables:

Messages per second	The arrival rate of web service requests into the CICS region. This is equivalent to the CPIH transaction count per second.
TOR CPU%	The percent of 1 CP that is being used by the CICS region to which all the clients are attached. This is as reported by RMF Monitor I under the “---APPL %---” heading for that report class. This region might also be known as a <i>routing region</i> .
AOR CPU%	The percent of 1 CP that is being used by the CICS region in which the target application program resides. This is as reported by RMF Monitor I under the “---APPL %---” heading for that report class.
TCP/IP CPU%	The percent of 1 CP that is being used by the TCP/IP address space. This is as reported by RMF Monitor I under the “---APPL %---” heading for that report class.
Response (ms)	The average CICS internal transaction response time in milliseconds as reported by RMF Monitor I in the TOR report class.
CPU per Msg (μs)	The average CPU time in microseconds calculated by adding all of the CPU used by the two CICS regions and the TCP/IP address space, then dividing by messages (msg) per second. One message represents one invocation of the target application program.
CICS web owning region CPU usage	The CPU usage for the region where the web service requests are processed by a pipeline and then routed to the target application running in the AOR. There is no application code run in this region. This is reported by RMF Monitor I for the report class for that region.

zEC12 versus z13

This section shows the results from running the SSL full handshake configuration on the zEC12 and then on the z13. The following tables show the data collected from the five different measurement points. The average shows the average number of microseconds of CPU per web service request, including both CICS regions and the TCP/IP address space.

Table 1 shows the performance data collected from the five different measurement points for the CICS TS V5.2 on zEC12 configuration.

Table 1 CICS TS V5.2 on zEC12

Messages per second	TOR CPU %	AOR CPU%	TCP/IP CPU %	Response (ms)	CPU per Msg (µs)
416	44.11	4.48	2.21	37	1221
500	52.97	5.37	2.64	37	1219
555	58.32	5.96	2.93	37	1210
714	74.92	7.66	3.74	37	1208
1000	103.63	10.37	4.99	37	1193

The average CPU usage per message shown in Table 1 is 1210 microseconds.

Table 2 shows the performance data collected from the five different measurement points for the CICS TS V5.2 on z13 configuration.

Table 2 CICS TS V5.2 on z13

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU %	Response (ms)	CPU per Msg (µs)
416	41.18	3.71	1.47	30	1090
500	47.97	4.45	1.75	29	1084
555	52.52	4.93	1.94	29	1070
714	68.01	6.35	2.49	29	1076
1000	93.99	8.88	3.44	28	1063

The average CPU usage per message (Table 2) is 1076 microseconds.

Figure 2 shows CPU usage for CICS and TCP/IP address spaces.

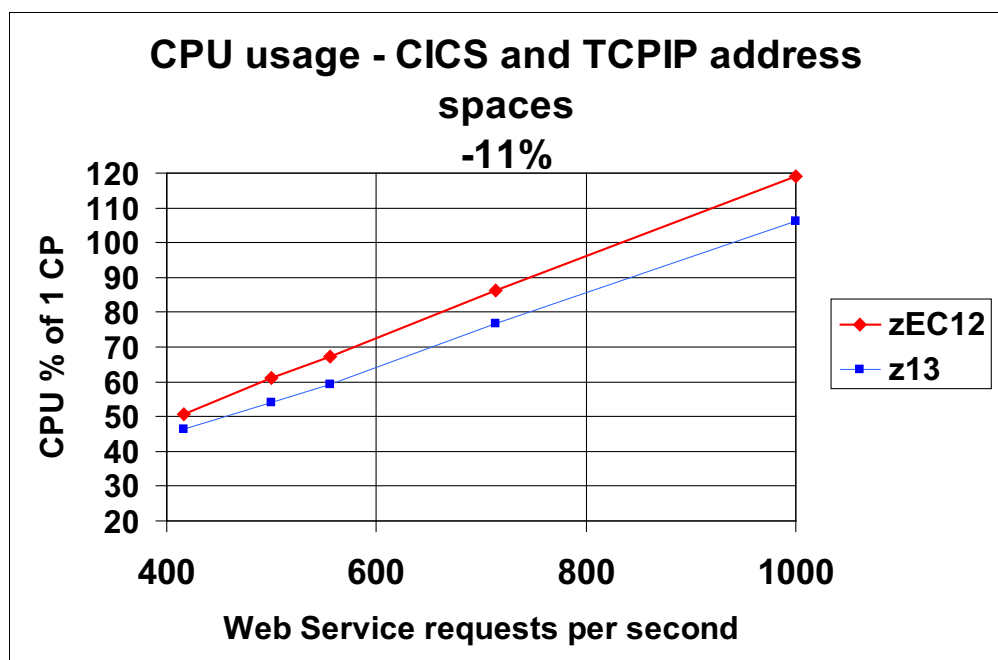


Figure 2 CPU usage for CICS and TCP/IP address spaces

Taking the average CPU used per message from the five measurement points shown in the tables (Table 1 on page 8 and Table 2 on page 8), we see 1210 microseconds on the zEC12 reduces to 1076 microseconds on the z13. Although this equates to an 11% drop in CPU/message for this particular workload, other internal IBM CICS workloads have shown up to a 30% reduction in CPU for the same hardware move.

Using the IBM Large System Performance Reference (LSPR)¹ to calculate what the reduction should be, shows it to be 10% for a workload with an average relative nest intensity (RNI).

To demonstrate the use of the LSPR for making this prediction, snippets of the relative internal throughput rate (ITR) tables from the LSPR for the relevant machines have been included.

Figure 3 shows the use of LSPR for the z13.

z13 (System z9 2094-701 = 1.00)							
Processor	#CP	PCI**	MSU***	MSUps****	Low*	Average*	High*
2964-701	1	1695	210	168	3.18	3.03	2.75
2964-702	2	3196	394	315	6.17	5.71	5.06
2964-703	3	4644	571	457	9.08	8.30	7.28

Figure 3 z13 using LSPR for making predictions

The relative ITR for processor 2964-703 for the Average RNI is 8.30. Therefore, the CPU per transaction must be (3 CPs / 8.3) = 0.361 seconds of CPU/transaction.

¹ Large Systems Performance Reference for IBM z Systems
<https://www.ibm.com/servers/resourcelink/lib03060.nsf/pages/lsprindex?OpenDocument>

Figure 4 shows zEnterprise EC12 using LSPR.

zEnterprise EC12 (System z9 2094-701 = 1.00)							
Processor	#CP	PCI**	MSU***	MSUps****	Low*	Average*	High*
2827-701	1	1,514	188	160	2.75	2.70	2.55
2827-702	2	2,853	352	299	5.34	5.10	4.69
2827-703	3	4,151	511	434	7.87	7.42	6.75

Figure 4 zEnterprise EC12 using LSPR for making predictions

Making the same calculation with the relative ITR for the 2827-703 processor in Figure 4, we get $(3 \text{ CPs} / 7.42) = 0.404$ seconds of CPU/transaction. This means that the CPU should reduce to $0.361/0.404$ (89.5%) of the original, which is exactly what is seen in the data collected during this benchmark.

In addition to a better ITR with the z13, the new Express 5 Cryptographic Coprocessors on the z13 have better performance than the Express 4, and so SSL handshaking is faster. See Figure 5.

----- CRYPTOGRAPHIC CCA COPROCESSOR ----					
			----- TOTAL -----		KEY-GEN
TYPE	ID	RATE	EXEC TIME	UTIL%	RATE
CEX4C	0	0.00	0.000	0.0	0.00
	6	1012	0.512	51.8	0.00
	7	2020	0.422	85.3	0.00
	8	0.00	0.000	0.0	0.00
----- CRYPTOGRAPHIC CCA COPROCESSOR ----					
			----- TOTAL -----		KEY-GEN
TYPE	ID	RATE	EXEC TIME	UTIL%	RATE
CEX5C	0	0.00	0.000	0.0	0.00
	6	2020	0.158	32.0	0.00
	7	0.00	0.000	0.0	0.00
	8	1012	0.240	24.2	0.00

Figure 5 Cryptographic CCA Coprocessor

The values in Figure 5 show the better execution times and the lower utilization for the same rate of handshaking between the two versions of coprocessor. Faster coprocessors contribute to the better CICS internal response times shown on the z13.

CICS TS V5.3 performance improvements

The following areas of CICS TS V5.3 have had performance enhancements made that have contributed to the benchmark results in this section:

- ▶ General improvements
- ▶ CICS monitoring
- ▶ CICS trace facility
- ▶ MRO session management
- ▶ Default level of TLS used by CICS
- ▶ SSL
- ▶ CWXN transaction
- ▶ AT-TLS aware support

SSL full handshake CICS TS V5.2 versus CICS TS V5.3

This section compares the workload when SSL was used and there were no persistent sessions; each connection therefore required an SSL handshake. In this scenario, there was no SSL ID caching, so each connection executed an SSL full handshake.

Table 3 shows the performance data for CICS TS V5.2 with SSL full handshake.

Table 3 CICS TS V5.2 with SSL full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	40.80	3.71	1.56	32	1107
500	48.45	4.45	1.86	31	1095
556	54.38	4.95	2.09	31	1104
714	68.43	6.37	2.65	31	1084
1000	94.66	8.91	3.64	30	1092

The average CPU usage per message (Table 3) is 1094 microseconds.

Table 4 shows the performance data for CICS TS V5.3 with SSL full handshake.

Table 4 CICS TS V5.3 with SSL full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	35.61	3.49	1.50	26	975
500	42.83	4.19	1.67	27	973
556	47.45	4.66	1.99	25	973
714	60.66	5.99	2.54	25	969
1000	82.04	8.36	3.50	21	939

The average CPU usage per message (Table 4) is 965 microseconds.

Figure 6 shows the CPU used only by the CICS region that handles the TCP/IP connections and runs the web services pipeline. At the highest transaction rate taken from the tables, the reduction in CPU per transaction when using CICS TS V5.3 is approximately 12%.

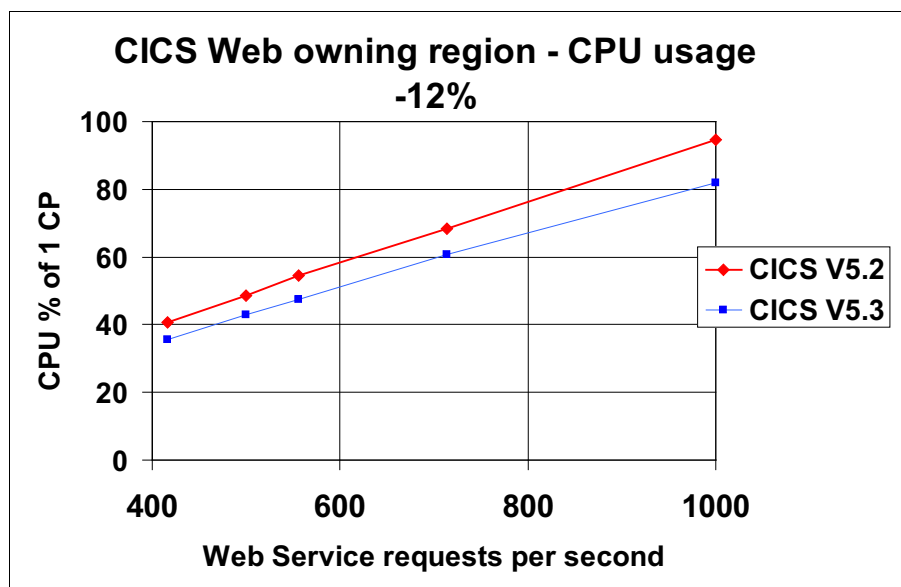


Figure 6 CICS web owning region CPU usage for SSL full handshake

Among other optimizations, in this configuration CICS TS V5.3 benefited from a reduction in the number of TCB switches needed and from the use of TLS1.2 instead of TLS1.0, both contributing to an improved response time, as shown in our results (see Table 3 on page 11 and Table 4 on page 11).

HTTP non-persistent session CICS TS V5.2 versus CICS TS V5.3

This section compares the workload when HTTP connections are used between the client and the server. There were no persistent connections, so each request made and terminated a connection.

Table 5 shows the performance data for CICS TS V5.2 with HTTP non-persistent sessions.

Table 5 CICS TS V5.2 with HTTP non-persistent sessions

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	19.72	3.41	0.64	1	571
500	23.93	4.14	0.80	1	577
556	26.58	4.60	0.89	1	583
714	34.13	5.91	1.13	1	576
1000	47.83	8.26	1.56	1	576

The average CPU usage per message shown in Table 5 is 576 microseconds.

Table 6 shows the performance data for CICS TS V5.3 with HTTP non-persistent sessions.

Table 6 CICS TS V5.3 with HTTP non-persistent sessions

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	16.72	3.30	0.68	<1	497
500	20.06	3.97	0.81	<1	496
556	22.37	4.42	0.90	<1	494
714	28.79	5.69	1.13	<1	498
1000	40.29	7.96	1.57	<1	498

The average CPU usage per message (Table 6) is 496 microseconds: a reduction of 13% when compared to CICS V5.2.

Figure 7 shows the CICS web owning region CPU usage with HTTP non-persistent sessions.

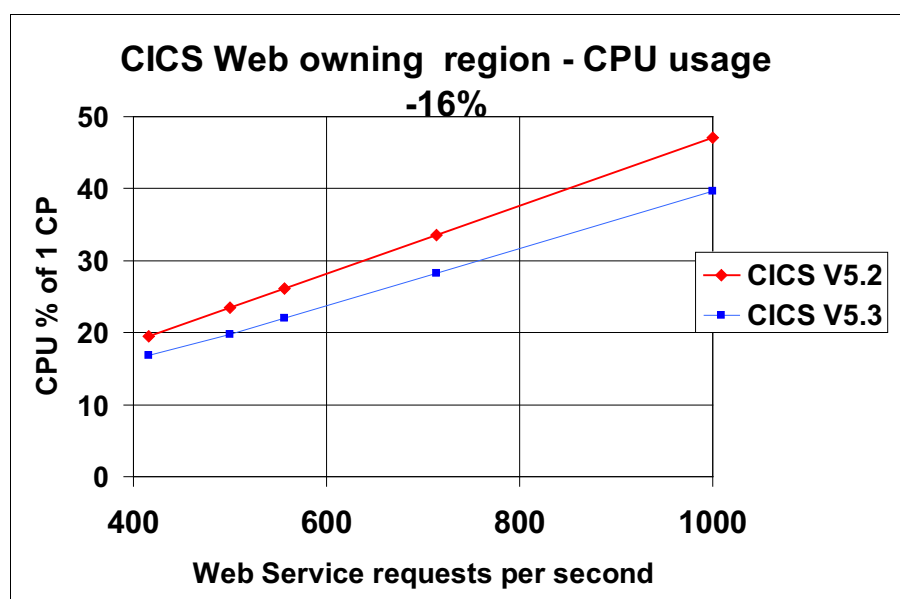


Figure 7 CICS web owning region CPU usage non-persistent sessions

In this scenario, the HTTP requests arriving in CICS fulfill the requirements for eliminating the need for the CWXN transaction. With CICS TS V5.2, there was a CWXN and a CPIH transaction needed for each web service request so that, at the highest message rate of 1000 per second, the CICS routing region was processing 2000 transactions per second. With CICS TS V5.3, the number of transactions needed was 1000 per second to achieve the same message arrival rate. Bear in mind that this might alter the average transaction profile in terms of CPU usage if you are making CPU per transaction calculations based on the total CPU used in the address space, divided by the number of executed transactions in the measurement period.

When CICS performance monitoring is active, CICS TS V5.3 requires less DASD to support the SMF data, because half of the number of records are written. This is demonstrated in Figure 9 on page 15 in "AT-TLS full handshake CICS TS V5.2 versus CICS TS V5.3".

AT-TLS full handshake CICS TS V5.2 versus CICS TS V5.3

This section shows the comparison between CICS TS V5.2 and V5.3 when AT-TLS is used to encrypt data between the client and the server. Persistent connections were not used. The AT-TLS policy was configured with no SSL ID caching by setting GSK_V3_SIDCACHE_SIZE=0 so each web service request resulted in a new connection and a full SSL handshake. With AT-TLS, all handshaking and encryption/decryption are done in the TCP/IP address space, and not the CICS address space.

Table 7 shows the performance data for CICS TS V5.2 with AT-TLS full handshake.

Table 7 CICS TS V5.2 with AT-TLS full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	20.02	3.72	11.17	32	839
500	24.02	4.47	13.40	31	837
556	26.74	4.96	14.93	31	838
714	34.36	6.38	19.09	31	837
1000	48.18	8.90	26.87	30	839

The average CPU usage per message (Table 7) is 838 microseconds.

Table 8 shows the performance data for CICS TS V5.3 with AT-TLS full handshake.

Table 8 CICS TS V5.3 with AT-TLS full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	16.99	3.48	9.90	<1	730
500	20.42	4.18	11.94	<1	730
556	22.69	4.65	13.21	<1	729
714	29.11	5.98	16.84	<1	727
1000	40.29	8.35	23.71	<1	723

The average CPU usage per message (Table 8) is 727 microseconds: a reduction of 13% when compared to CICS V5.2.

With this AT-TLS configuration, CICS sees the incoming data as HTTP, not HTTPS, because TCP/IP handles the handshaking and encryption/decryption. This means that unlike CICS in an SSL environment, in this scenario CICS TS V5.3 can take advantage of eliminating the CWXN transaction. This gives a reduction in CPU due to fewer task dispatches, but it also reduces DASD usage for SMF records if performance monitoring is active, because the number of transactions in this routing region has been halved.

The results also show a reduction in CICS internal response times for CICS TS V5.3. In CICS TS V5.2, the SSL handshake is done by the CWXN transaction and is executed synchronously, whereas in CICS TS V5.3, the handshake is completed asynchronously in the TCP/IP address space before CICS starts processing the request. Although overall the response times seen at the client might not change significantly, the reduced response time in CICS TS V5.3 results in fewer concurrent transactions at any one time.

Figure 8 shows the CICS web owning region CPU usage with AT-TLS full handshake.

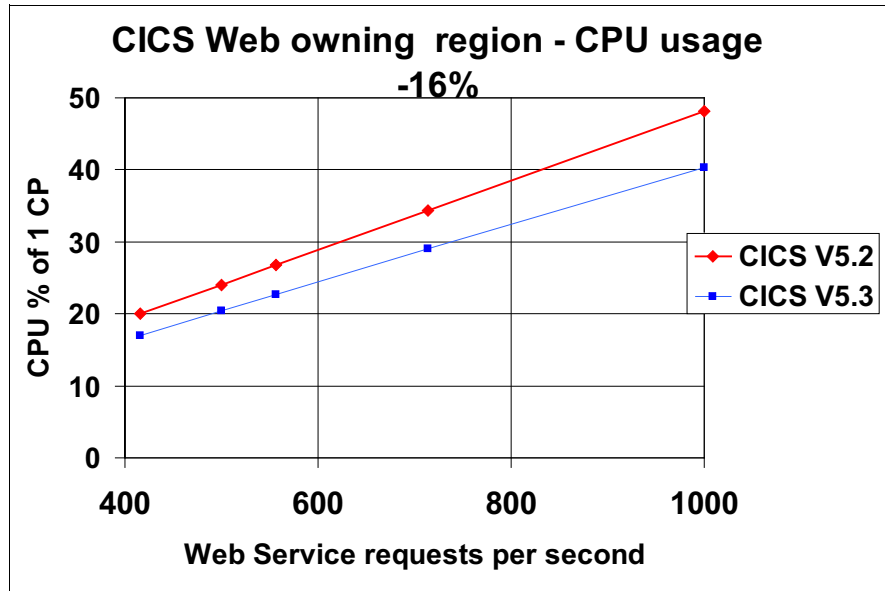


Figure 8 CICS web owning region CPU usage with AT-TLS full handshake

Figure 9 shows cylinders of DS8800 DASD devices used over time.

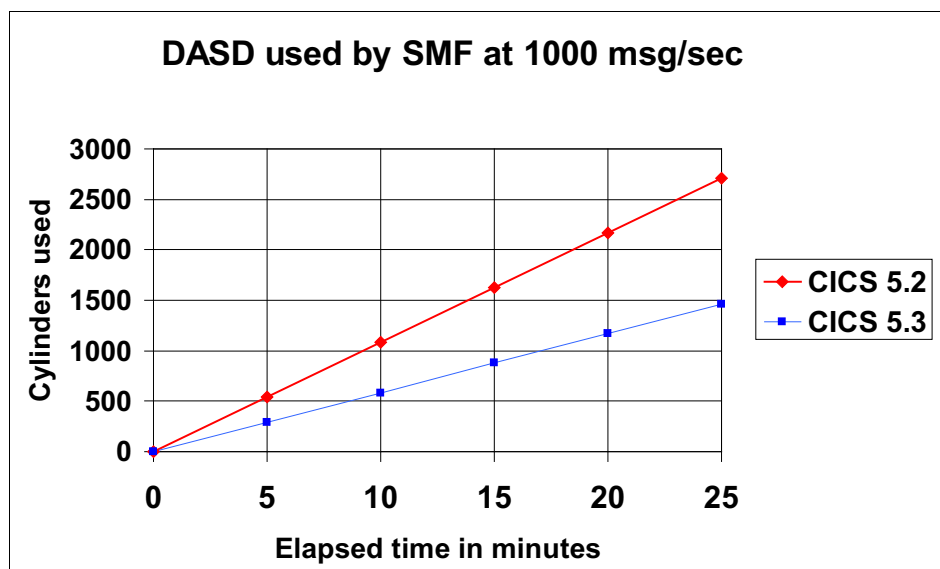


Figure 9 DASD used by SMF

CICS TS V5.3 SSL versus CICS TS V5.3 AT-TLS

This section compares the use of CICS with SSL and CICS with AT-TLS. In this scenario, no persistent connections were used, and no SSL ID caching was used in either case. Handshaking and encryption/decryption were performed in the CICS address space with SSL and in the TCP/IP address space with AT-TLS.

Table 9 shows the performance data for CICS TS V5.3 with SSL full handshake.

Table 9 CICS TS V5.3 with SSL full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	35.61	3.49	1.50	26	975
500	42.83	4.19	1.67	27	973
556	47.45	4.66	1.99	25	973
714	60.66	5.99	2.54	25	969
1000	82.04	8.36	3.50	21	939

The average CPU usage per message (Table 9) is 965 microseconds.

Table 10 shows the performance data for CICS TS V5.3 with AT-TLS full handshake.

Table 10 CICS TS V5.3 with AT-TLS full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	16.99	3.48	9.90	<1	730
500	20.42	4.18	11.94	<1	730
556	22.69	4.65	13.21	<1	729
714	29.11	5.98	16.84	<1	727
1000	40.29	8.35	23.71	<1	723

The average CPU usage per message (Table 10) is 727 microseconds: a reduction of 24% when compared to a CICS V5.3 SSL full handshake configuration.

Figure 10 shows the total CPU used for SSL full handshake versus AT-TLS full handshake.

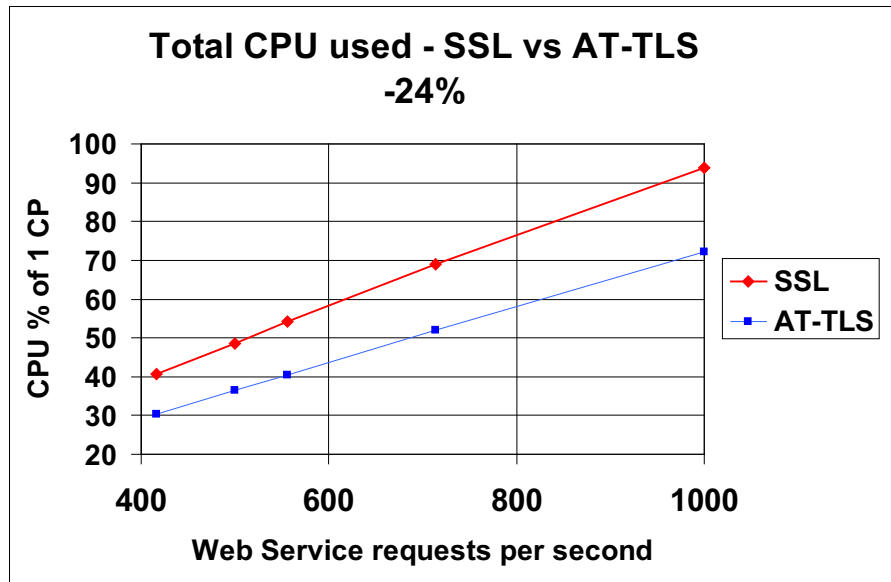


Figure 10 Total CPU used for SSL full handshake versus AT-TLS full handshake

Figure 11 shows the CPU breakdown by address space.

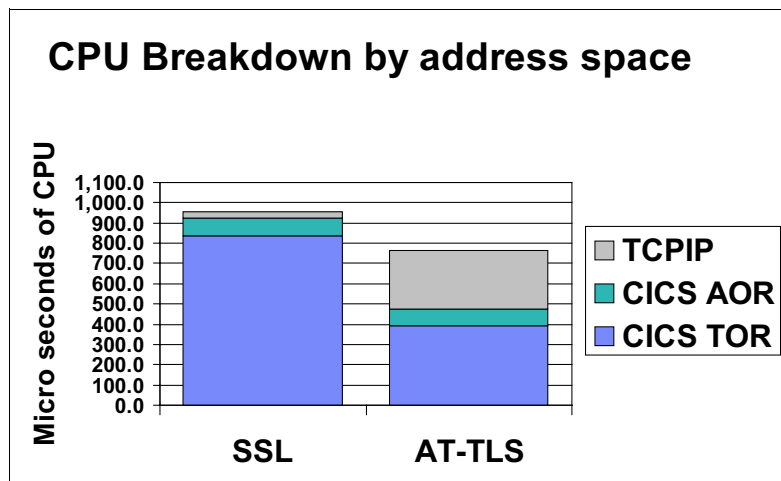


Figure 11 CPU breakdown by address space

AT-TLS in this benchmark uses about 24% less CPU than the CICS/SSL configuration. A large proportion of the CPU usage has moved from the CICS address space to the TCP/IP address space. The AT-TLS configuration also benefits from not having to run the CWXN transaction, because CICS sees the connection as HTTP and is able to optimize.

Response times in the AT-TLS configuration are improved compared with SSL. This is mainly due to the SSL handshake being moved from being done by the CWXN transaction in CICS with SSL into the TCP/IP address space with AT-TLS. Response times as seen by the client might not show a significant change. However, reduced response times in CICS reduce transaction concurrency and therefore allow for further growth in throughput.

SSL configurations

This section shows the costs associated with the three variations of SSL usage, that is:

1. Full handshake
2. Partial handshake
3. Encryption/decryption only

All are based on CICS TS V5.3.

Table 11 shows performance data for the CICS TS V5.3 SSL configuration with full handshake.

Table 11 CICS TS V5.3 with SSL full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	35.19	3.46	1.40	28	962
500	41.87	4.16	1.67	28	954
556	46.35	4.62	1.85	27	950
714	59.66	5.94	2.37	26	951
1000	83.96	8.31	3.32	25	955

The average CPU usage per message (Table 11) is 954 microseconds.

Table 12 shows performance data for the CICS TS V5.3 SSL configuration with partial handshake.

Table 12 CICS TS V5.3 with SSL partial handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	24.49	3.37	0.93	3	692
500	29.60	4.06	1.12	3	695
556	33.02	4.52	1.25	3	697
714	42.52	5.81	1.61	3	699
1000	59.57	8.12	2.24	3	699

The average CPU usage per message (Table 12) is 696 microseconds.

Table 13 shows performance data for the CICS TS V5.3 SSL configuration with persistent sessions.

Table 13 CICS TS V5.3 with SSL persistent sessions

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	15.60	3.11	0.27	1	456
500	18.77	3.74	0.32	1	456
556	20.81	4.17	0.35	<1	456
714	26.79	5.35	0.45	<1	456
1000	37.50	7.50	0.62	1	456

The average CPU usage per message (Table 13) is 456 microseconds.

Figure 12 shows the cost in microseconds of CPU associated with the various SSL configurations.

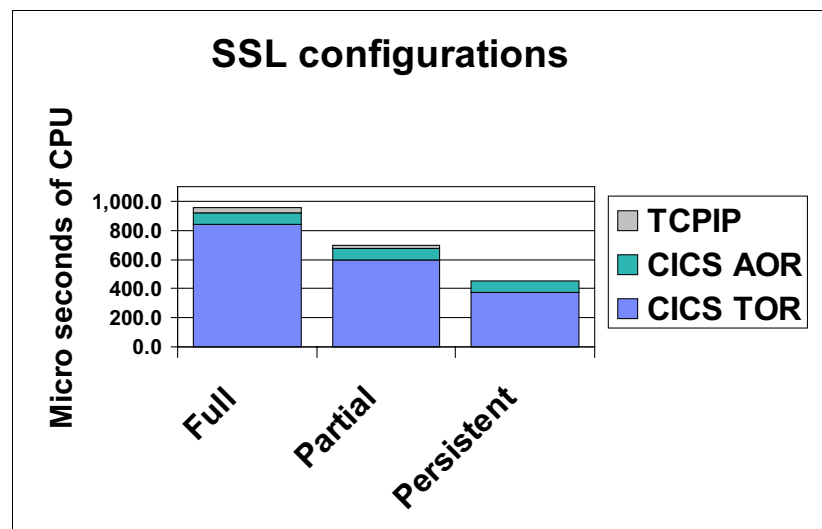


Figure 12 CPU cost per request for the SSL configurations

AT-TLS configurations

This section shows the different costs associated with the three variations of AT-TLS usage, that is:

1. Full handshake
2. Partial handshake
3. Encryption/decryption only

All are based on CICS TS V5.3.

Table 14 shows the performance data for CICS TS V5.3 with AT-TLS full handshake.

Table 14 CICS TS V5.3 with AT-TLS full handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	16.41	3.49	12.02	<1	767
500	19.72	4.18	14.42	<1	766
556	21.90	4.65	16.09	<1	766
714	28.16	5.99	20.58	<1	766
1000	39.38	8.38	28.96	<1	767

The average CPU usage per message (Table 14) is 766 microseconds.

Table 15 shows the performance data for CICS TS V5.3 with AT-TLS partial handshake.

Table 15 CICS TS V5.3 with AT-TLS partial handshake

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	17.47	3.43	8.00	<1	694
500	21.03	4.13	9.77	<1	698
556	23.32	4.59	10.74	<1	695
714	29.97	5.91	13.63	<1	693
1000	41.92	8.27	19.21	<1	694

The average CPU usage per message (Table 15) is 694 microseconds.

Table 16 shows the performance data for CICS TS V5.3 AT-TLS persistent sessions.

Table 16 CICS TS V5.3 with AT-TLS persistent sessions

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (µs)
416	13.30	3.09	0.47	1	405
500	15.98	3.72	0.55	<1	405
556	17.79	4.13	0.62	1	405
714	22.80	5.29	0.79	1	404
1000	31.91	7.40	1.09	1	405

The average CPU usage per message (Table 16) is 405 microseconds.

Figure 13 on page 21 shows the cost in microseconds of CPU per request associated with the various AT-TLS configurations.

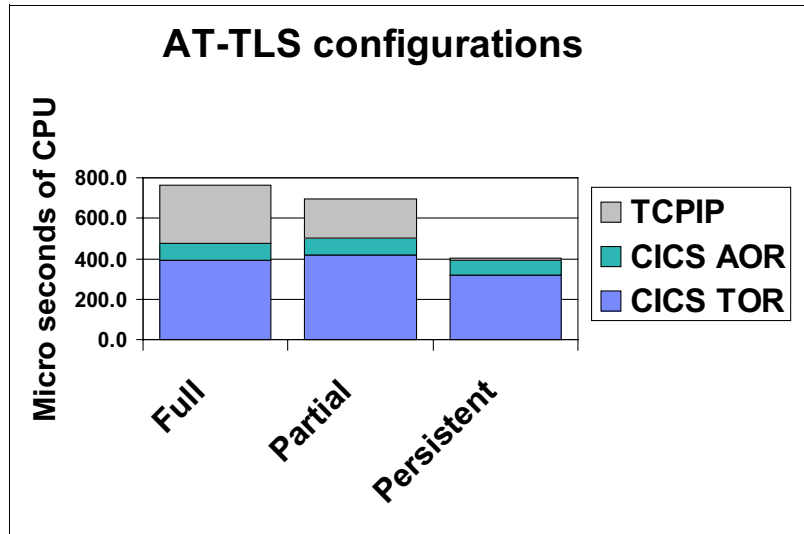


Figure 13 CPU cost per request for the AT-TLS configurations

HTTP configurations

This section shows the difference in cost between persistent and non-persistent connections.

Table 17 shows the performance data for CICS TS V5.3 with non-persistent HTTP connections.

Table 17 CICS TS V5.3 with non-persistent HTTP connections

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (μs)
416	16.85	3.33	0.61	<1	499
500	19.74	3.98	0.72	<1	488
556	21.96	4.43	0.80	<1	489
714	28.26	5.69	1.02	<1	489
1000	39.66	7.96	1.04	<1	490

The average CPU usage per message (Table 17) is 492 microseconds.

Table 18 shows the performance data for CICS TS V5.3 with persistent HTTP connections.

Table 18 CICS TS V5.3 with persistent HTTP connections

Messages per second	TOR CPU%	AOR CPU%	TCP/IP CPU%	Response (ms)	CPU per Msg (μs)
416	12.90	3.07	0.24	1	389
500	15.51	3.70	0.29	1	390
556	17.23	4.11	0.32	1	389
714	22.17	5.29	0.41	1	392
1000	31.01	7.39	0.56	1	389

The average CPU usage per message (Table 18 on page 21) is 389 microseconds.

Figure 14 shows the difference in cost in microseconds of CPU per request between persistent and non-persistent connections.

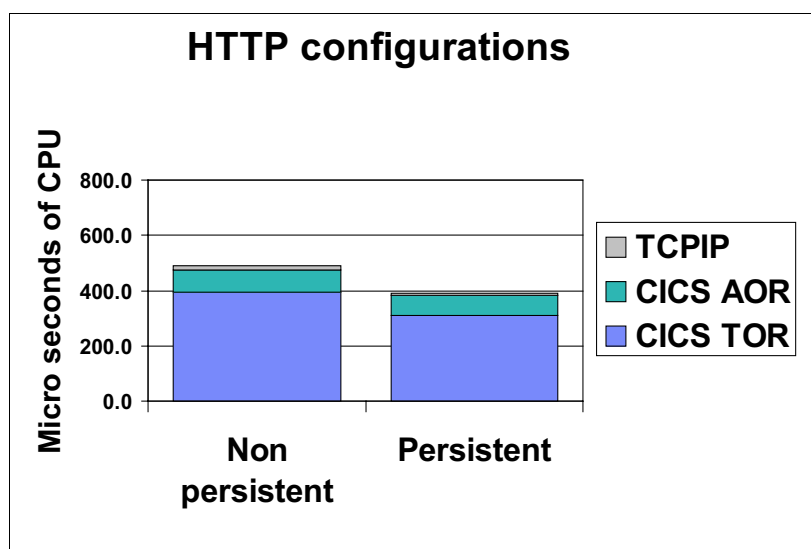


Figure 14 CPU cost per request for the HTTP configurations

Applying the results to your environment

This publication has presented reductions in CPU usage as a relative value for our workload under the conditions described. It is easily understood that changes introduced by a new release of CICS can provide a range of measured performance gains and is heavily dependent on many CICS configuration options, the supporting software and hardware environment, along with behavior of the application code.

To understand how the demonstrated performance improvements can be applied to your environment, the following list provides a guide to the functional areas that you can take advantage of in your environment to maximize the benefit of upgrading to CICS TS V5.3:

► General improvements in CICS TS V5.3

CICS TS V5.3 uses the following techniques to provide general performance improvement over earlier releases of CICS:

- Use of several new hardware instructions in IBM z9®
- Cache alignment of key CICS control blocks
- Additional tuning of internal procedures

These techniques combine to reduce the overall CPU usage of a CICS region. These improvements can provide benefits to most CICS workloads.

► CICS monitoring

A reduction in lock contention for the CICS monitoring subsystem has provided benefits for all workloads where customers are running with CICS performance class monitoring records enabled (MN=ON and MNPER=ON). This benefit is available with no configuration changes when upgrading to CICS TS V5.3.

As part of the CICS TS V5.3 monitoring improvements, it is now more efficient to configure the monitoring domain to emit all monitoring fields, rather than select individual fields for output. For environments using a custom monitoring control table (MCT), this improvement can be realized by removing the DFHMCT TYPE=RECORD,CLASS=PERFORM entry from the MCT.

► CICS trace facility

Improvements in the CICS trace subsystem provide benefits to customers running with a default level of trace enabled when writing to the internal trace table. The changes have contributed to the overall performance improvements demonstrated between CICS V5.2 and CICS V5.3 in this publication.

► MRO session management

CICS TS V5.3 has improved the MRO session management algorithm. In this publication, the benchmark uses MRO/XM links to connect the TOR with the AOR. This improvement can be of benefit to all workload types that use MRO sessions, with the greatest benefit realized when a high number of sessions are configured for each connection.

► Default TLS level

The default level of TLS in use for CICS TS V5.3 regions has been increased from V1.0 to V1.2. The implementation of TLS V1.2 on z/OS is more efficient than TLS V1.0. Therefore, a workload using the default level of TLS support in CICS TS V5.2 and earlier can see an improvement in CPU used per request where a handshake is required.

► SSL

When using CICS SSL support, a significant number of TCB change mode operations have been removed from the CICS infrastructure, a large proportion of which were in the SSL handshaking process. This removal of change mode operations results in a reduction in CPU consumed per request for each SSL request received by CICS, where the most impact will be seen by requests that use non-persistent connections.

► CWXN transaction

For applications that use the CICS web support (either as a web application or as a web service), optimizations have been made to remove the CWXN transaction where it is no longer necessary. If an inbound request is received by CICS, the user transaction will be directly attached without an intermediate CWXN transaction, if the following conditions are met:

- The request URI matches to an installed URIMAP
- The connection does not use SSL support
- No custom analyzer program has been specified
- Sufficient data has been received by CICS

This removal of the CWXN transaction results in an improvement in the CPU consumed per web request. There is also a secondary benefit to the elimination of this task for customers who run with CICS performance class monitoring records enabled. Without the CWXN transaction, the number of SMF 110 subtype 1 records is reduced, saving DASD space for customers who routinely collect and store monitoring data.

- ▶ AT-TLS aware support

AT-TLS provides the ability to abstract SSL certificate and algorithm configuration away from an individual CICS region and move this configuration to the network stack.

The use of AT-TLS with a CICS V5.2 or earlier has always been permitted, but with the disadvantage that SSL certificate information was not available to the invoked application. With CICS TS V5.3, an application can now extract the SSL certificate information even when the SSL connection is managed by the TCP/IP address space.

In addition to the CICS management benefits of AT-TLS, the range of scenarios that can take advantage of the CWXN task removal is increased. By using AT-TLS, each request received is presented to CICS as a non-SSL stream, meaning at least one of the four conditions for CWXN task elimination has been fulfilled, while retaining the use of an encrypted connection.

- ▶ z/OS Workload Manager (WLM) interaction

Although not highlighted in this paper, there have also been enhancements made to interactions with z/OS V2.1 WLM and V2.2 WLM. With these enhancements, transactions running in CICS TS V5.3 can be classified into Service and Report Classes, which directly measure and report their CPU consumption. CICS has been enhanced to continuously measure CPU consumption for every transaction without the need to activate CICS performance class monitoring. These measurements are aggregated by z/OS WLM to provide enhanced reporting in SMF records and RMF reports. The CPU measurements for workload that are classified in this way make it simpler and more efficient to collect the data required for mobile workload pricing.

Summary

This publication has presented a selection of configuration scenarios for a web services workload to demonstrate performance improvements introduced in the CICS TS for z/OS V5.3 release.

In all of the benchmarks described in this paper, we have demonstrated that upgrading from CICS TS V5.2 to CICS TS V5.3 can provide performance benefits with *no configuration changes*. Additionally, CICS TS V5.3 introduces several new features that potentially offer even greater savings, in terms of CPU usage, reduction in transaction concurrency allowing greater throughput, and a reduction in management overhead.

Upgrading to CICS TS V5.3 from releases earlier than CICS V5.2 has the potential to offer even greater improvements than those detailed in this paper. Refer to the following CICS TS V5.1 and V5.2 documentation to understand how cumulative changes in the CICS TS product can further improve performance for your environment:

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.1.0/com.ibm.cics.ts.whatsnew.doc/topics/dfhe4_whatsnew.html

http://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.whatsnew.doc/topics/dfhe4_whatsnew.html

This paper also describes some of the performance improvements observed by SSL workloads when upgrading hardware from an IBM zEnterprise EC12 to an IBM z13™. These performance results highlight the improvements introduced by the z13, including CP Assist for Cryptographic Function (CPACF) and use of the new Crypto Express 5S cryptographic coprocessors. Other types of workloads can provide benefits from other enhancements made in the z13.

Authors

This paper was produced by a team of specialists from around the world working with the IBM International Technical Support Organization, Hursley Center.

John Burgess is a Senior Software Engineer working in the IBM Hursley CICS performance team. He has over 20 years experience with CICS and specializes in the performance of large systems.

Ian Burnett is the CICS Transaction Server Performance Team Lead working for IBM United Kingdom. He has worked in IBM since 2001 as part of the development teams for several products including IBM WebSphere® Application Server, WebSphere MQ, and IBM CICSplex® System Manager. Since 2009, Ian has led the performance team in Hursley, with experience in a wide range of CICS performance topics. He holds a Bachelor's degree in Computer Science from the University of Warwick in the UK.

Martin Cocks is a Software Engineer in the United Kingdom. He has 13 years of experience in CICS TS, most recently technically leading the performance improvements in CICS TS V5.3. He has worked at IBM for 13 years. His areas of expertise include web services, Active-Active, CICS file control, and performance.

Thanks to the following people for their contributions to this project:

- ▶ Karen Lawrence
IBM International Technical Support Organization, Hursley Center
- ▶ Catherine Moxey
IBM Hursley Senior Technical Staff Member, CICS Performance and Optimization
- ▶ Tony Hogg
IBM Hursley z/OS systems programmer
- ▶ Will Chorlton
IBM Hursley z Systems hardware specialist

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<https://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. 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.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	Redpaper™	z/OS®
CICSplex®	Redbooks (logo)  ®	z13™
IBM®	Resource Measurement Facility™	z9®
IBM z13™	RMF™	zEnterprise®
Redbooks®	WebSphere®	

The following terms are trademarks of other companies:

Other company, product, or service names may be trademarks or service marks of others.



REDP-5322-00

ISBN 0738454966

Printed in U.S.A.

Get connected

