z/OS Connect Enterprise Edition V2.0

Performance Summary using WOLA

Version 1.0

August 2016

Alan Hollingshead
alan_hollingshead@uk.ibm.com

z/OS Connect EE
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

1.1 Notices

This report is intended for Architects, Systems Programmers, Analysts and Programmers wanting to understand the performance characteristics of z/OS Connect EE V2.0. The information is not intended as the specification of any programming interfaces that are provided by z/OS Connect EE.

It is assumed that the reader is familiar with the concepts and operation of z/OS Connect EE V2.0.

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.

1.2 Trademarks and service marks

1.3 Terminology

CICS	-	CICS Transaction Server
Cost per transaction (ms)	-	CPU usage per transaction, in milliseconds
CP	-	Central Processor
CPU %	-	Percentage of CPU time used by transactions running on general purpose processors
EE	-	Enterprise Edition
GCP	-	General purpose Central Processor
PID	-	Product Identification Number
RMF	-	Resource Measurement Facility
SMF	-	System Management Facility
SSL	-	Secure Sockets Layer
TPS	-	Number of Transactions Per Second
TRUE	-	Task Related User Exit
TT	-	Think Time in seconds. The time between individual requests.
WOLA	-	WebSphere Optimized Local Adapters
Workload Driver	-	An application written for the purpose of these performance tests to simulate multiple simultaneous client requests
zIIP	-	IBM z Systems Integrated Information Processor

2. Overview

This report contains performance measurements for z/OS Connect EE V2.0, program number (PID) 5655-CEE, using the WOLA service provider to CICS Transaction Server. The WOLA service provider is shipped as part of z/OS Connect EE V2.0.

What is z/OS Connect EE?

IBM z/OS Connect EE V2.0 delivers RESTful APIs as a discoverable, first-class resource with Swagger 2.0 descriptions. It includes an API package artifact that encapsulates the RESTful API, together with necessary detail to invoke underlying services in the z/OS subsystems (such as CICS Transaction Server or IMS).

API Mapping

API mapping adds an abstraction layer between the API consumer and the underlying z/OS assets, allowing in-line manipulation of requests such as the mapping of HTTP headers, pass-through, redaction or defaulting of JSON fields, and rearranging the order of JSON fields and data.

Service Providers

IBM z/OS Connect EE V2.0 supports various service providers to access major z/OS subsystems such as CICS Transaction Server and IMS. This report focuses on the WOLA service provider.

3. Performance runs using the WOLA service provider

This report focuses on an end-to-end solution that uses the WOLA service provider to access CICS Transaction Server. It also uses the API mapping feature of z/OS Connect EE V2.0. The API mapping feature is not available in z/OS Connect V1.

Figure 1 shows the flow of requests and responses for the scenario:

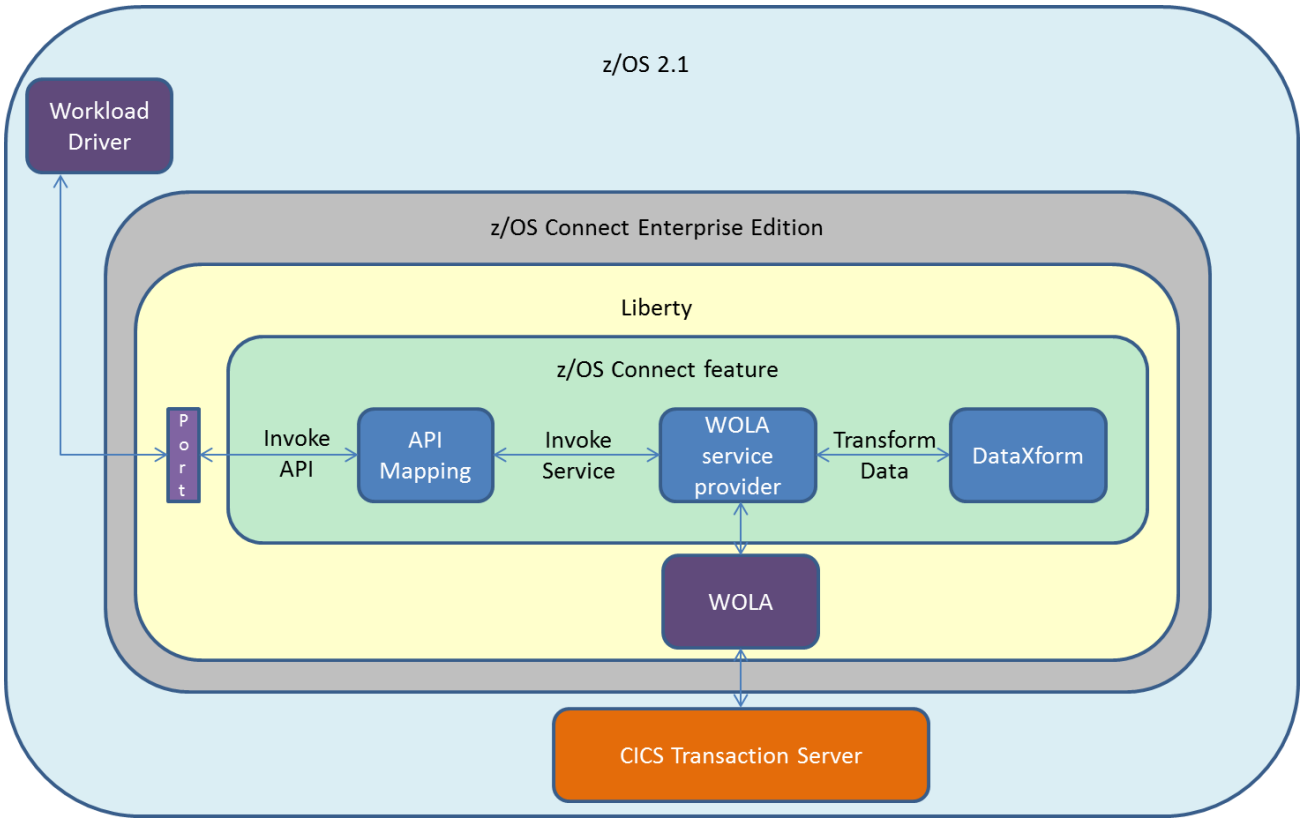


Figure 1 End-to-end solution using the WOLA service provider

3.1 Flow of work

The flow of work is as follows:

1. The workload driver sends concurrent client HTTP JSON requests to z/OS Connect EE using persistent HTTP connections. See page 7 for details of the format of these requests.
2. The JSON request calls the API feature within z/OS Connect EE.
3. The API feature performs header and field mapping.
4. The API calls the service defined within the z/OS Connect feature.
5. The service invokes the WOLA service provider, which uses the dataXform feature of z/OS Connect EE to convert the JSON to a byte array.
6. The WOLA service provider forwards the transformed data request to the WOLA component within Liberty which then invokes the WOLA TRUE (Task Related User Exit) within CICS Transaction Server.
7. The WOLA code within CICS Transaction Server links to the CICS COBOL application.
8. Depending on the content of the JSON request, the CICS COBOL application sends a 1K, 4K, 8K or 16K payload in the response.

3.2 Format of Requests

The details of each request are described below.

URL

Each HTTP request specifies a URL with the following format:

- http://<host>:<port>/<basepath>/<relative path>

  For example:
  http://myhost:2222/comm1k/mapping/CABASIC-RS2/1

- The <basepath> in the example is “comm1k”

- The <relative path> in the example is “mapping/CABASIC-RS2/1”

- The “1” at the end of the relative path is the path parameter “countIn” used by the API Mapping feature within the API Editor (see Figure 7 on page 12).

Request Fields

Each JSON request consists of two fields and is approximately 50 bytes long.

- The first field is called “count_in” whose value is passed in as a “path parameter”.
- The second field is called “count_out”. The value in this second field is used by the CICS COBOL application to determine the size of the payload to be returned by CICS.

Figure 2 shows an example of a JSON request that the CICS COBOL application interprets to generate a 1K response.

```
{"COMM1KOperation":{
    "count_in":1,"count_out":32
    }
}
```

Figure 2 JSON request used to request a 1K response

3.3 Format of Responses

The details of the responses are described below.

Response Fields

The size of the response to be sent by the CICS COBOL application is determined by the “count_out” field in the request. In the example shown in Figure 2, the “count_out” field is requesting 32 “blocks” of data. The CICS COBOL application interprets this value by generating

1. An array called “user_data” with 31 elements each consisting of 32 alphabetic values (i.e. 31 x 32 bytes = 992 bytes);
2. Five fields of various values (e.g. tranid) totalling 32 bytes.

Therefore the total response size in this example is 1K (i.e. 992 bytes + 32 bytes = 1024 bytes).

Figure 3 shows the format of the 1K response.


```
{
  "COMM1KOperationResponse": {
    "tranid": "BBO#",
    "user_data": [
      {
        "user_data": "0001-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0002-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0003-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0004-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0005-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0006-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0007-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0008-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0009-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0010-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0011-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0012-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0013-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0014-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0015-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0016-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
      {
        "user_data": "0017-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
      },
```

```
    {
      "user_data": "0018-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0019-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0020-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0021-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0022-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0023-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0024-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0025-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0026-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0027-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0028-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0029-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0030-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    },
    {
      "user_data": "0031-ABCDEFGHIJKLMNOPQRSTUVWXYZ-"
    }
  ],
  "taskid": 181,
  "rs_spare": "",
  "recv_size": 0,
  "send_size": 1024
 }
}
```


Figure 3 Example of JSON 1K response

The response data is based on the CICS COBOL copybook shown in Figure 4:

```
      *
      * Response from application CABASIC
      * 1kB of application data
           05 RECV-SIZE    PIC 9(8) COMP-4.
           05 SEND-SIZE    PIC 9(8) COMP-4.
           05 TASKID       PIC 9(8) COMP-4.
           05 TRANID       PIC X(4).
           05 RS-SPARE     PIC X(16).
           05 USER-DATA    PIC X(32) OCCURS 31 TIMES.
```

Figure 4 CICS COBOL copybook for 1K response

4K Responses

For 4K response payloads the data formats of the requests and responses follow a similar pattern to those for the 1K responses.

The JSON request is approximately 50 bytes in length, as shown in Figure 5.

```
{"COMM4KOperation":{
    "count_in":1,"count_out":128
    }
}
```

Figure 5 JSON request used to request a 4K response

The second field, “count_out” requests 128 “blocks” of data. The CICS COBOL application interprets this value by generating

1. An array called “user_data” with 127 elements each consisting of 32 alphabetic values (i.e. 127 x 32 bytes = 4064 bytes);
2. Five fields of various values (e.g. tranid) totalling 32 bytes.

Therefore the total response size in this example is 4K (i.e. 4064 bytes + 32 bytes = 4096 bytes).

The response data is based on the CICS COBOL copybook shown in Figure 6:

```
      *
      * Response from application CABASIC
      * 4kB of application data
           05 RECV-SIZE    PIC 9(8) COMP-4.
           05 SEND-SIZE    PIC 9(8) COMP-4.
           05 TASKID       PIC 9(8) COMP-4.
           05 TRANID       PIC X(4).
           05 RS-SPARE     PIC X(16).
           05 USER-DATA    PIC X(32) OCCURS 127 TIMES.
```

Figure 6 CICS COBOL copybook for 4K response

8K and 16K Responses

For larger response sizes such as 8K and 16K, the data formats of the responses follow a similar pattern to those for the 1K and 4K responses.

- For 8K responses, the CICS COBOL application generates a 8192 byte payload consisting of
  - An array called “user_data” with 255 elements each consisting of 32 alphabetic values (i.e. 255 x 32 bytes = 8160 bytes);
  - Five fields of various values (e.g. tranid) totalling 32 bytes.
- For 16K responses, the CICS COBOL application generates a 16384 byte payload consisting of
  - An array called “user_data” with 511 elements each consisting of 32 alphabetic values (i.e. 511 x 32 bytes = 16352 bytes);
  - Five fields of various values (e.g. tranid) totalling 32 bytes.

The request size of approximately 50 bytes remains the same regardless of the different response sizes.

3.4 API Mapping

The API Editor in z/OS Connect EE V2.0 is used to perform HTTP-to-JSON mapping.

The scenarios in this performance report use a simple mapping using the path parameter “countIn” passed in the URL (see URL on page 7). The API Editor maps this value to the JSON “count_in” field, and implicitly converts it from a string to an integer, as shown in Figure 7. All other JSON fields from the HTTP request are automatically mapped without alteration.

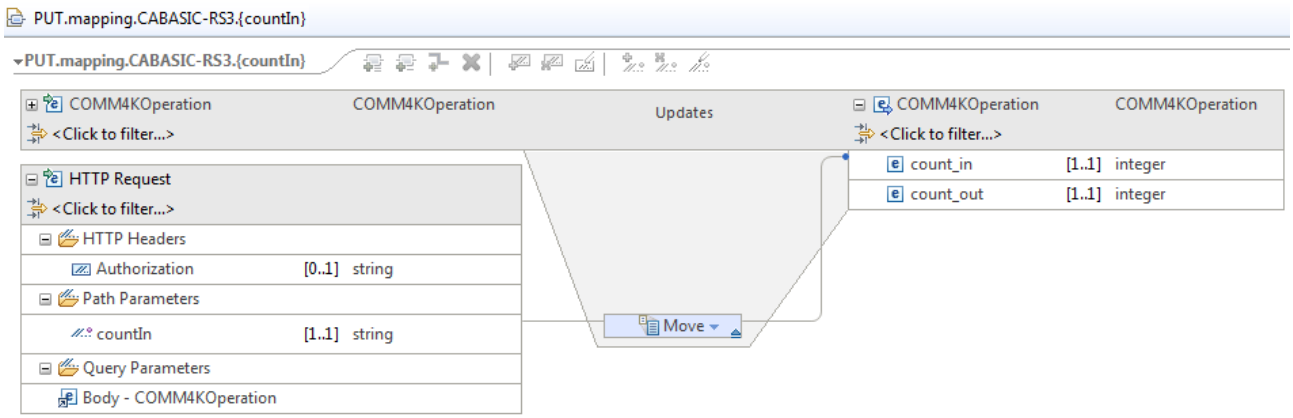


Figure 7 API Editor mapping countIn to count_in on the Request Mapping

The Response Mapping is shown in Figure 8.

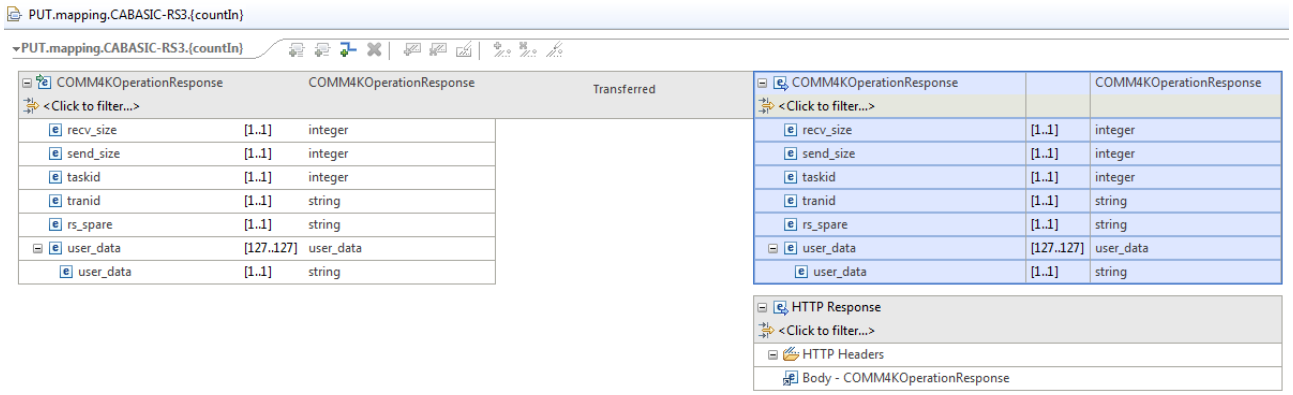


Figure 8 API Editor showing the Response Mapping

Although no mapping is required during the response flow in this scenario, the response map shows the number of array elements in the response. In the example above there are 127 user_data array elements within the 4K response.

4. Test Environment

This chapter describes the environment the performance tests were run in. It includes details of the Workload Driver, and how z/OS Connect EE V2.0 was configured.

4.1 Hardware

- IBM System z13 2964-NE1 model 7A5
- 10GB of Central Storage (RAM)
- LPAR with 5 dedicated GCPs (no zIIPs)
- OSA-Express5S 10GB Ethernet

4.2 Software

- z/OS V2.1
- z/OS Connect Enterprise Edition (EE) V2.0
  - z/OS Connect Enterprise Edition version 2.0.1.0 (20160615-1722)
    - PTF UI38699
  - Interim fixes installed
    - PI51171,PI58468,PI57546,PI54855,PI52665,PI59320
- IBM 64-bit SDK for z/OS Java Technology Edition, Version 8.0
  - Build pmz6480sr2fp10-20160108_01(SR2 FP10)
- Angel Process V3
- CICS Transaction Server V5.3

4.3 Workload Driver

The workload driver, used to simulate multiple client requests, is a Java application that runs in its own JVM on the same LPAR as z/OS Connect EE. Requests issued by the workload driver specify “localhost” to minimise potential network latency.

Thinktime

Requests are issued by the workload driver at a regular pace, using a “thinktime”. For each client this is the time the workload driver waits after a response has been received before sending the next request. All performance runs in this report use a thinktime of

- 100ms for 1K and 4K responses.
- 400ms for 8K and 16K responses.

HTTP PUT Requests

All scenarios issue HTTP PUT requests. There are no performance measurements for the HTTPS secure protocol as this is covered by Liberty performance reports, and not affected by z/OS Connect EE.

4.4 Asymmetric Payloads

Each payload was asymmetric, meaning that the size of data for the request was different to the size of data for the response. For example, a 50 byte payload request may have generated a 4K response. The size of the response was dependent upon the data within

the request.

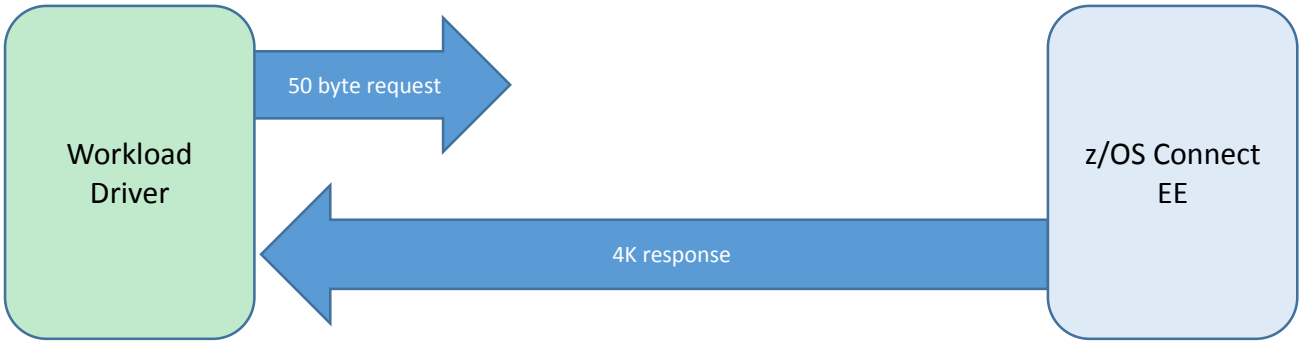


Figure 9 Example of an asymmetric payload showing a 50 byte request generating a 4K response

The structure of each request contained a simple object (see Figure 2 on page 7) whereas the structure in the response is much larger. The larger the payload the greater the number of array elements in the response. For example, a 4K response consisted of 127 array elements, whereas a 1K response consisted of just 31.

The number of array elements for the different sized responses is shown in *Figure 10*.

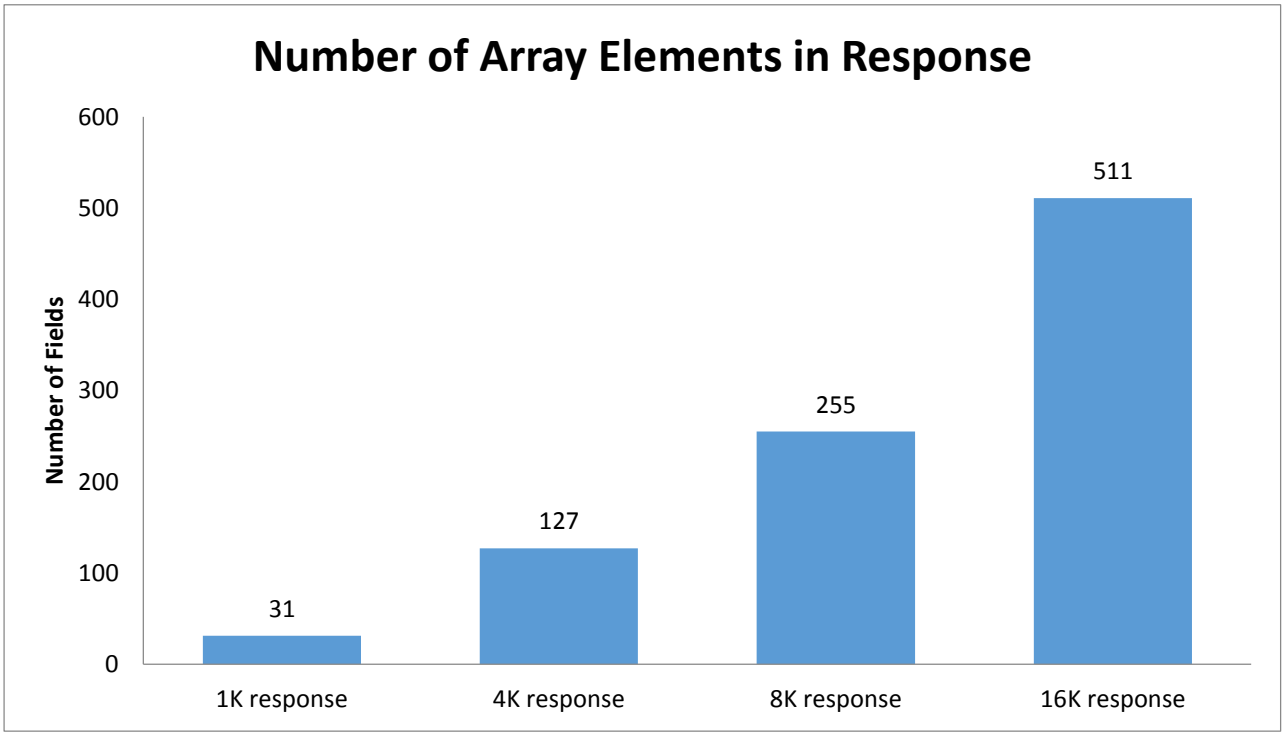


Figure 10 Number of array elements in the response

4.5 z/OS Connect EE Configuration

z/OS Connect EE was configured with:

- IBM 64-bit SDK for z/OS Java Technology Edition, Version 8.0
  - Build pmz6480sr2fp10-20160108_01(SR2 FP10)
- REGION size 0M
- MEMLIMIT 4G
- Maximum Java heap size (Xmx) 1G
- Other JVM system properties use default values (including GC mode: gencon)

The server.xml file contains definitions for:

- zosconnect-2.0 feature
- WOLA service provider
- Services used
- Data transformation

Note that the API used for these performance tests does not require any definitions in the server.xml file.

Feature Definitions

Figure 11 shows the feature definitions specified in the server.xml configuration file.

- The first feature is for z/OS Connect EE V2.0.
- The second feature is for the WOLA service provider.

```
<featureManager>
    <feature>zosconnect:zosconnect-2.0</feature>
    <feature>zosLocalAdapters-1.0</feature>
</featureManager>
```

Figure 11 Feature definitions in server.xml

WOLA Service Provider

The WOLA service provider is provided as part of z/OS Connect EE and allows calls to be made from a standalone z/OS Connect EE server to a WOLA-enabled CICS Transaction Server region.

For this report the WOLA service provider receives a request and passes it to CICS Transaction Server where a COBOL application is called. Depending on the content of the request, the COBOL application generates a response between 1K and 16K which is passed back to the WOLA server provider.

WOLA and Service Definitions

Figure 12 shows the service definitions specified in the server.xml configuration file.

- Within the zosconnect_zosConnectService elements the serviceName “comm1K” is for the 1K response scenario, “comm4K” for the 4K response scenario, and so on.
- The serviceRef for each service is wolaCABASIC which has a serviceName of

CABASIC, the CICS COBOL application used in all the scenarios.

- All the services defined use the JSON to byte array data transformation feature supplied with z/OS Connect EE.

```
<!-- Local adapters config -->
<zosLocalAdapters wolaGroup="CATMGR1" wolaName2="CATMGR2" wolaName3="CATMGR3" />

<!-- Local adapters connection factory definition -->
<connectionFactory id="wolaCF" jndiName="eis/ola">
       <properties.ola/>
</connectionFactory>

<!-- WOLA Connect service and z/OS Connect service definitions -->
<zosconnect_localAdaptersConnectService id="wolaCABASIC"
       registerName="CICSREG"
       serviceName="CABASIC"
       connectionFactoryRef="wolaCF"/>

<!-- Reference Implementation service -->
<zosconnect_zosConnectService id="comm1k_id"
       requireAuth="false"
       requireSecure="false"
       serviceName="comm1k"
       serviceRef="wolaCABASIC"
       dataXformRef="xformJSON2Byte" />

<!-- Reference Implementation service -->
<zosconnect_zosConnectService id="comm4k_id"
       requireAuth="false"
       requireSecure="false"
       serviceName="comm4k"
       serviceRef="wolaCABASIC"
       dataXformRef="xformJSON2Byte" />

<!-- Reference Implementation service -->
<zosconnect_zosConnectService id="comm8k_id"
       requireAuth="false"
       requireSecure="false"
       serviceName="comm8k"
       serviceRef="wolaCABASIC"
       dataXformRef="xformJSON2Byte" />

<!-- Reference Implementation service -->
<zosconnect_zosConnectService id="comm16k_id"
       requireAuth="false"
       requireSecure="false"
       serviceName="comm16k"
       serviceRef="wolaCABASIC"
       dataXformRef="xformJSON2Byte" />
```

Figure 12 WOLA and service definitions in server.xml

Data Transformation Definition

Figure 13 shows the data transformation definition specified in the server.xml configuration file.

- The data transformer defined within the zosconnect_ zosConnectDataXform element uses the information in the bind files to transform data from/to

different object types. For the performance tests in this report the data is transformed between JSON and byte arrays. These byte arrays map to the COBOL copybook structures described in 3.3 Format of Responses.

- The bind files were generated using the BAQLS2JS utility using COBOL copybooks to generate the artefacts.
- The location of the bindfiles and schemas is optional but must be in a directory accessible by z/OS Connect EE.

```
<!-- z/OS Connect data transformation provider -->
<zosconnect_zosConnectDataXform id="xformJSON2Byte"
       bindFileLoc="/u/ctgperf/bindfiles" bindFileSuffix=".wsbind"
       requestSchemaLoc="/u/ctgperf/schemas"
       responseSchemaLoc="/u/ctgperf/schemas"
       requestSchemaSuffix=".json"
       responseSchemaSuffix=".json">
</zosconnect_zosConnectDataXform>
```

Figure 13 Data transformation definition in server.xml

API

The API used for the performance runs is modelled using the API Editor and then deployed through the API Editor. (Alternatively, the z/OS Connect EE API Deployment Utility could have been used but would have required the .aar file to be exported and the z/OS Connect server to be restarted).

- The deployed API is automatically loaded upon first invocation, and does not need to be explicitly defined in the server.xml file.
- There are no interceptors used in these performance tests.

Angel Process

The Angel process is required to use z/OS authorized services such as WOLA. For these performance runs, version 3 of the Angel process was used.

CICS TRUE and Link Server

In CICS Transaction Server the TRUE was started using the command:

BBOC START_TRUE

The link server is then registered with WOLA with the following parameters:

BBOC START_SRVR RGN=CICSREG DGN=CATMGR1 NDN=CATMGR2 SVN=CATMGR3 SVC=* MNC=100 MXC=100 TXN=N SEC=N REU=Y

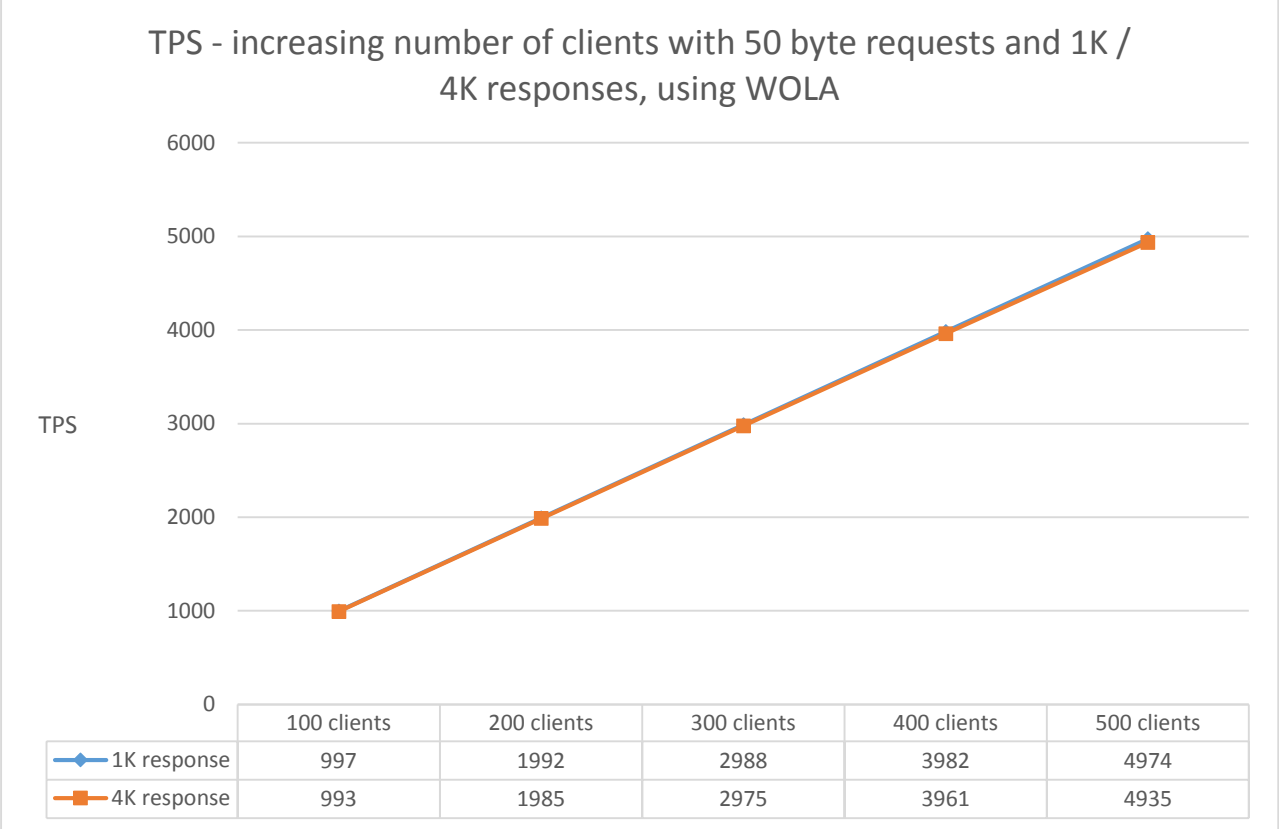
Note that the values for minimum and maximum connections (MNC and MXC respectfully) are both set to 100. As security propagation (SEC) is set to No, the reuse parameter (REU) was set to yes (Y). Setting REU to yes enables the link server to reuse the program link invocation tasks (BBO# transactions) between program invocation requests.

5. Transactions Per Second

The following results show the transactions per second (TPS) observed in the controlled environment.

5.1 TPS for 1K and 4K Responses

These performance runs used a thinktime of 100ms.



	100 clients	200 clients	300 clients	400 clients	500 clients
1K response	997	1992	2988	3982	4974
4K response	993	1985	2975	3961	4935

Figure 14 TPS for 1K and 4K responses with increasing numbers of clients

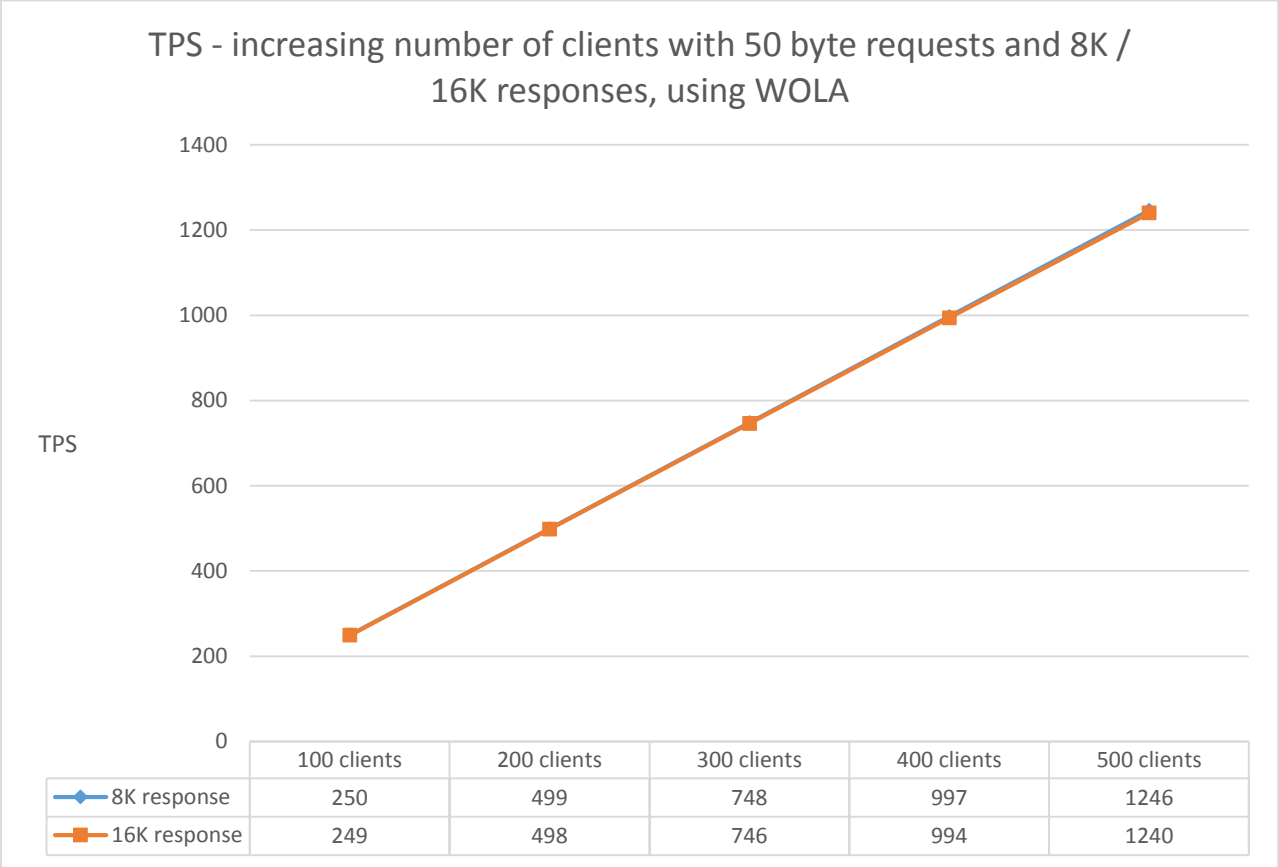
Observations

- ✓ z/OS Connect EE demonstrated good scalability.
- ✓ The results for 1K and 4K responses are almost identical (therefore the 1K line appears to be “overlaid” by the 4K line).
- ✓ Increasing the number of clients to run in parallel did not compromise the TPS.

5.2 TPS for 8K and 16K payload responses

In this environment, larger payload responses such as 8K or 16K responses exhausted the CPU available when the workload was run with a thinktime of 100ms.

By slowing down the workload to use a thinktime of 400ms, the transactions per second for 8K and 16K responses showed the following results:



	100 clients	200 clients	300 clients	400 clients	500 clients
8K response	250	499	748	997	1246
16K response	249	498	746	994	1240

Figure 15 TPS for 8K and 16K responses with increasing numbers of clients

Observations

- ✓ z/OS Connect EE demonstrated good scalability.
- ✓ The results for 8K and 16K responses are almost identical (therefore the 8K line appears to be “overlaid” by the 16K line).
- ✓ Increasing the number of clients to run in parallel did not compromise the TPS when running with a thinktime of 400ms.

6. Average Response time

Ideally the average response time should be the same whether there is one user or 500 users. Realistically, as the number of clients increase, the average response time can be expected to also increase due to resource contention in the system. Typically a maximum throughput in the system will be reached somewhere in any configuration at which point requests will begin to queue, thus increasing the average response time.

6.1 Average Response time for 1K and 4K Responses

Figure 16 shows the average response for 1K and 4K responses for different numbers of clients.

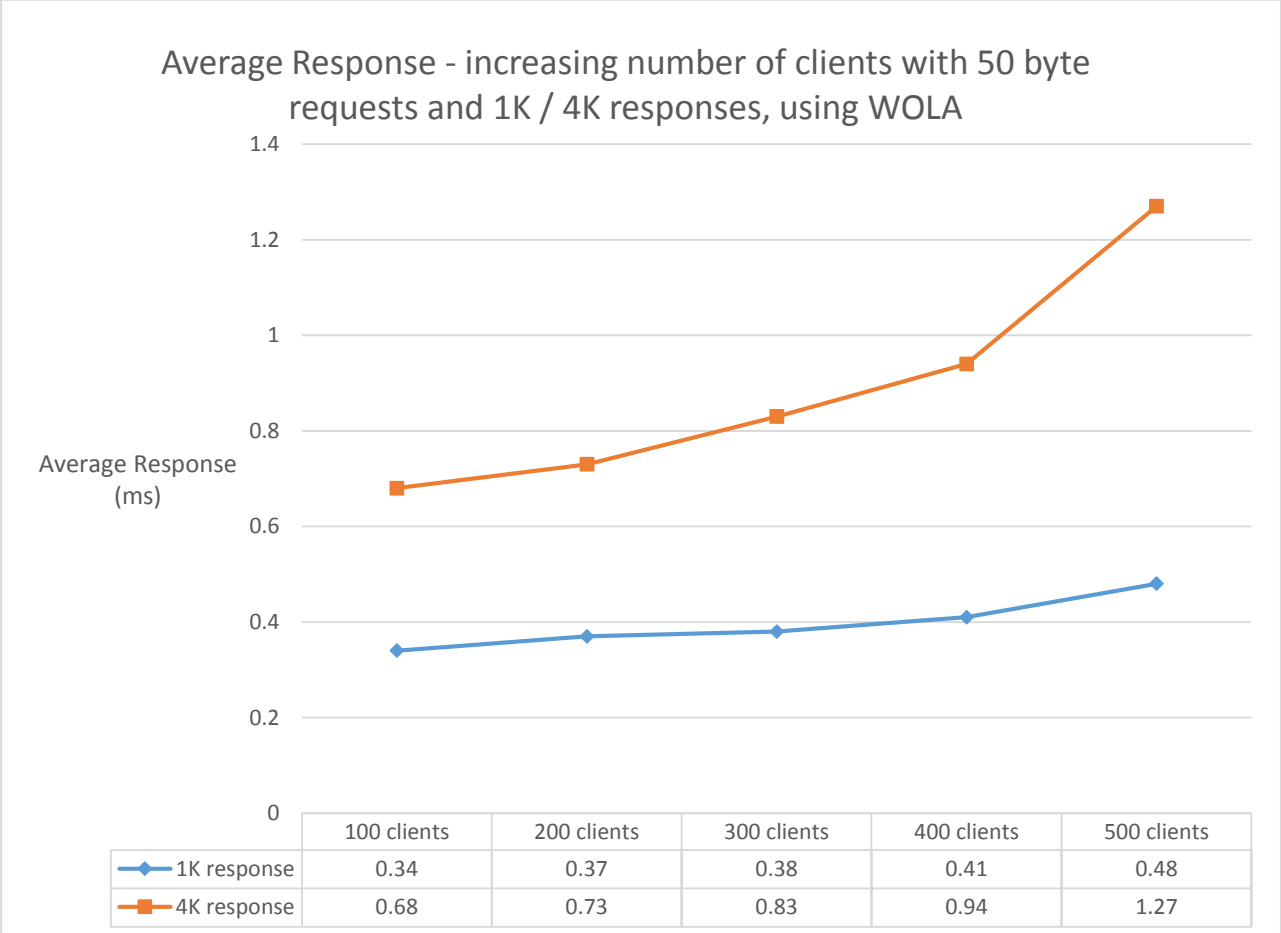


Figure 16 Average Response for 1K and 4K responses with increasing numbers of clients

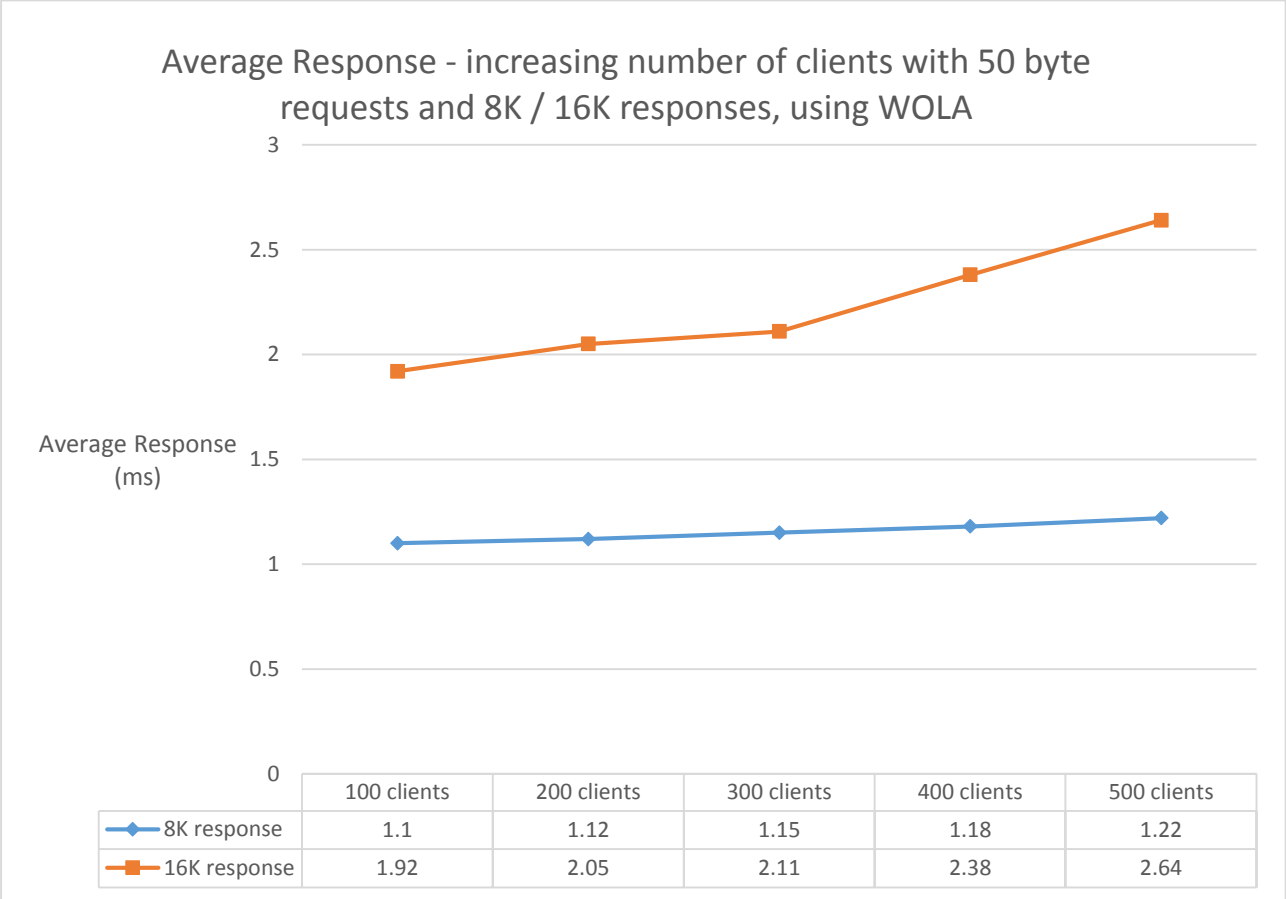
Observations

- ✓ z/OS Connect EE demonstrated acceptable scalability for average response times.
- ✓ Increasing the number of clients to run in parallel did not create unacceptable response times.

6.2 Average Response time for 8K and 16K payload responses

In this environment, larger payload responses such as 8K or 16K responses exhausted the CPU available when the workload was run with a thinktime of 100ms.

By slowing down the workload to use a thinktime of 400ms, the average response times for 8K and 16K responses showed the following results:



	100 clients	200 clients	300 clients	400 clients	500 clients
8K response	1.1	1.12	1.15	1.18	1.22
16K response	1.92	2.05	2.11	2.38	2.64

Figure 17 Average Response for 8K and 16K responses with increasing numbers of clients

Observations

- ✓ z/OS Connect EE demonstrated acceptable scalability for average response times.
- ✓ Increasing the number of clients to run in parallel did not create unacceptable response times.

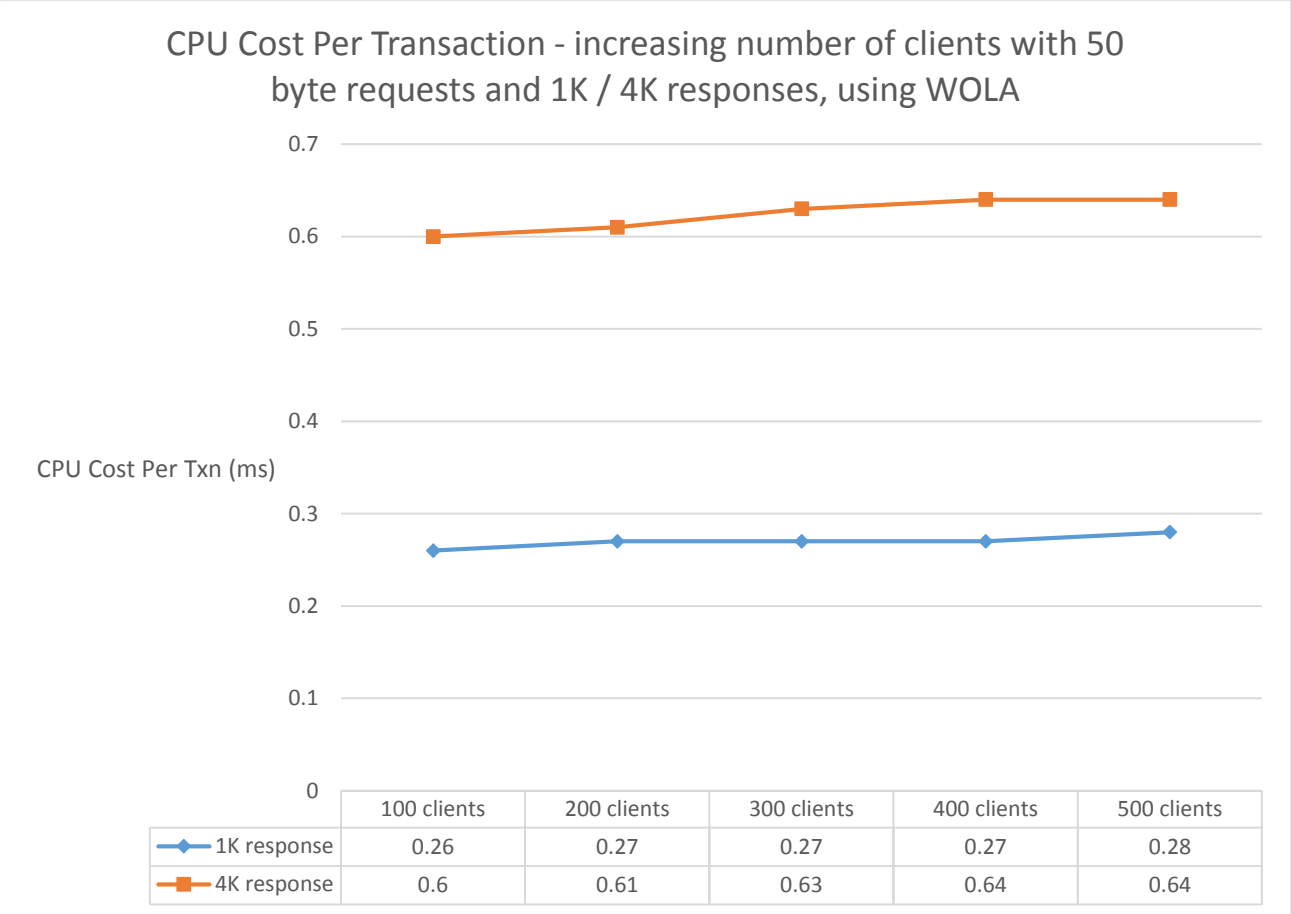
7. CPU Cost Per Transaction

The cost per transaction in CPU terms is measured in milliseconds (ms).

The calculation is made by dividing the CPU service time by the number of seconds in the SMF interval. This is then divided by the number of transactions per second (TPS) for the same SMF interval, and finally multiplied by one thousand to get the result in milliseconds.

7.1 CPU Cost Per Transaction for 1K and 4K Responses

Figure 18 shows the CPU cost per transaction for 1K and 4K responses for different numbers of clients.



	100 clients	200 clients	300 clients	400 clients	500 clients
1K response	0.26	0.27	0.27	0.27	0.28
4K response	0.6	0.61	0.63	0.64	0.64

Figure 18 CPU cost per transaction for 1K and 4K responses with increasing numbers of clients

Observations

- ✓ z/OS Connect EE demonstrated good scalability with minimal increase in the CPU Cost Per Transaction as the number of clients increased.
- ✓ Increasing the number of clients to run in parallel from 100 to 500 clients resulted in little or no change in CPU Cost Per Transaction for this scenario.

7.2 CPU Cost Per Transaction for 8K and 16K Responses

Figure 19 shows the CPU cost per transaction for 8K and 16K responses for different numbers of clients.

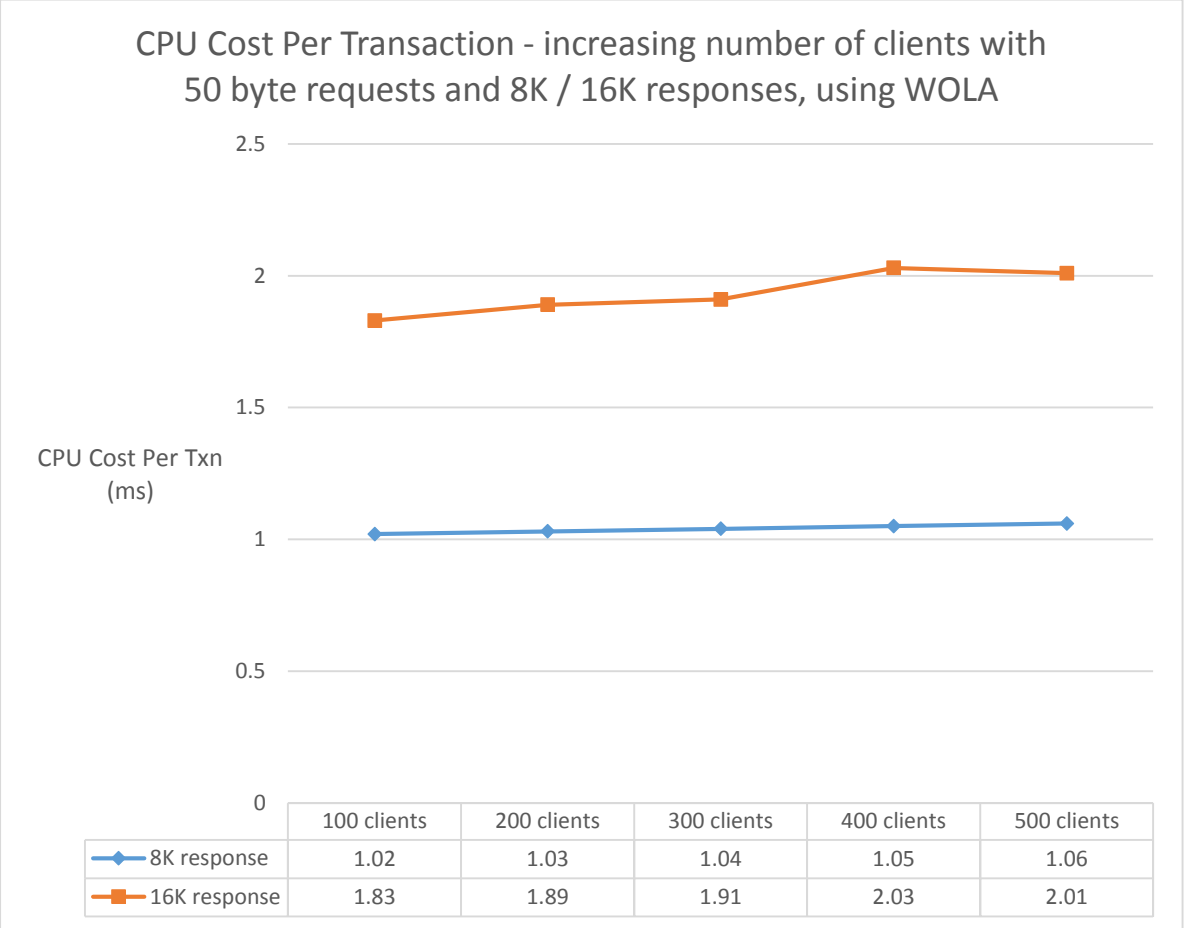


Figure 19 CPU cost per transaction for 8K and 16K responses with increasing numbers of clients

Observations

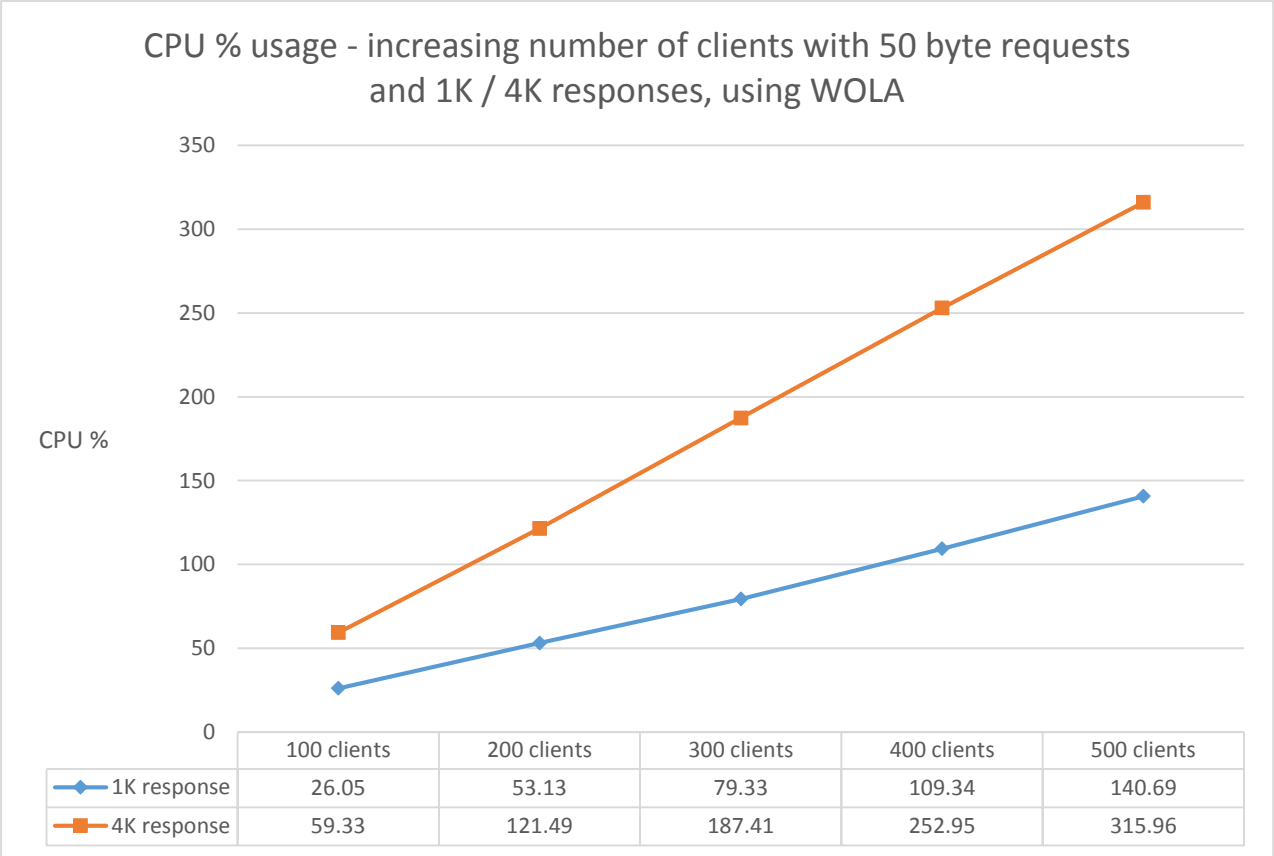
- ✓ z/OS Connect EE demonstrated good scalability with minimal increase in the CPU Cost Per Transaction as the number of clients increased.
- ✓ Increasing the number of clients to run in parallel from 100 to 500 clients resulted in minimal change in CPU Cost Per Transaction for this scenario.

8. CPU % usage

The CPU % usage will naturally increase as the number of clients running in parallel also increase. The CPU % includes all the GCPs (five in this environment), allowing a theoretical maximum of 500%.

8.1 CPU % usage for 1K and 4K Responses

Figure 20 shows the CPU % usage for 1K and 4K responses for different numbers of clients.



	100 clients	200 clients	300 clients	400 clients	500 clients
1K response	26.05	53.13	79.33	109.34	140.69
4K response	59.33	121.49	187.41	252.95	315.96

Figure 20 CPU% usage for increasing number of clients for 1K and 4K responses

Observations

- ✓ z/OS Connect EE demonstrated good scalability.
- ✓ The CPU % usage increased linearly as the number of clients running in parallel were increased.
- ✓ The steeper line for 4K responses is as a result of the 4K responses containing a larger number of array elements (as described in Format of Responses on page 7). As the number of array elements in a payload increases, so can the CPU processing time required to handle each of the array elements. Therefore a 4K response with 127 array elements will require more CPU processing than a 1K response with just 31 array elements.

8.2 CPU % usage for 8K and 16K Responses

Figure 21 shows the CPU % usage for 8K and 16K responses for different numbers of clients.

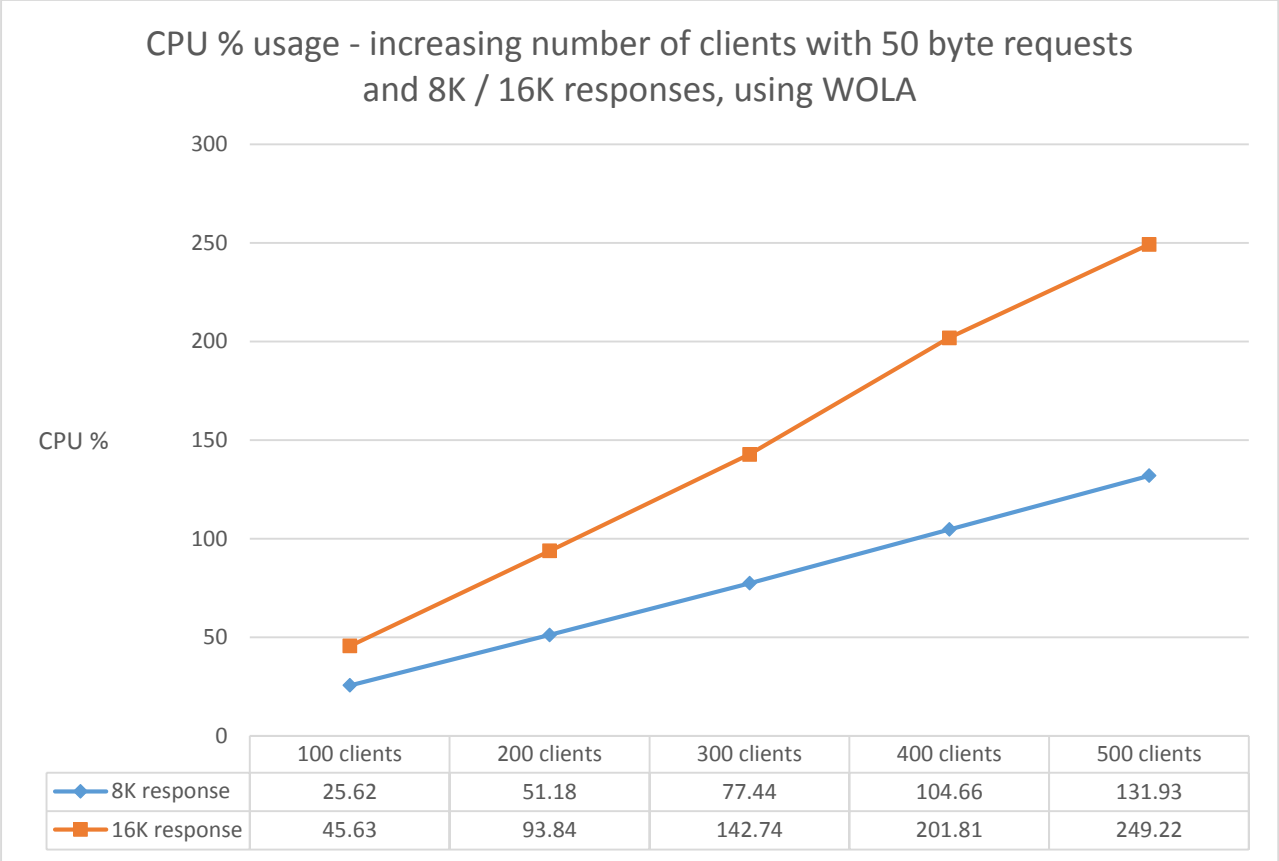


Figure 21 CPU% usage for increasing number of clients for 8K and 16K responses

Observations

- ✓ z/OS Connect EE demonstrated good scalability.
- ✓ The CPU % usage increased linearly as the number of clients running in parallel were increased.
- ✓ The steeper line for 16K responses is as a result of the 16K responses containing a larger number of fields (as described in Format of Responses on page 7). As the number of array elements in a payload increases, so can the CPU processing time required to handle each of the array elements. Therefore a 16K response with 511 array elements will require more CPU processing than an 8K response with 255 array elements.

9. zIIP Eligibility

As z/OS Connect EE V2.0 is almost entirely written in Java the zIIP eligibility was observed to understand the benefits of offloading work to one or more zIIPs.

The environment is configured with five GCPs. Although zIIPs have not been used, zIIP eligibility was captured and is shown alongside the actual GCP usage.

9.1 zIIP Eligibility for 1K responses

Figure 22 shows the GCP % usage and zIIP eligibility for 1K responses.

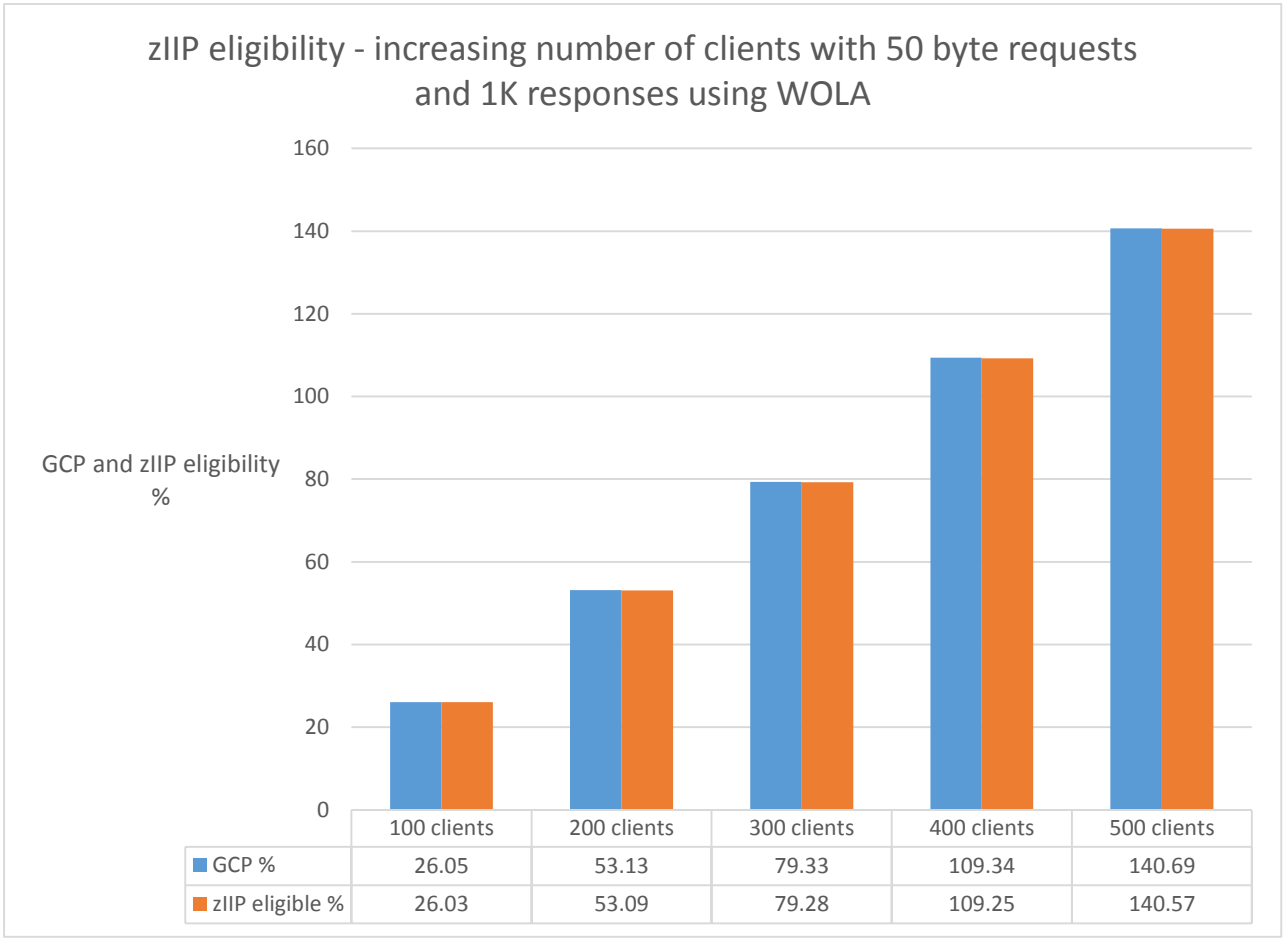


Figure 22 GCP CPU % and zIIP eligibility for increasing numbers of clients with 1K responses

Observations

- ✓ z/OS Connect EE is a Java-based product and so over 99% of the product is eligible to be offloaded to zIIP.
- ✓ The potential usage of a zIIP scaled well with the increasing numbers of clients.

9.2 zIIP Eligibility for 4K responses

Figure 23 shows the GCP % usage and zIIP eligibility for 4K responses.

Figure 23 GCP CPU % and zIIP eligibility for increasing numbers of clients with 4K responses

Observations

- ✓ z/OS Connect EE is a Java-based product and so over 99% of the product is eligible to be offloaded to zIIP.
- ✓ The potential usage of a zIIP scaled well with the increasing numbers of clients.

9.3 zIIP Eligibility for 8K responses

Figure 24 shows the GCP % usage and zIIP eligibility for 8K responses.

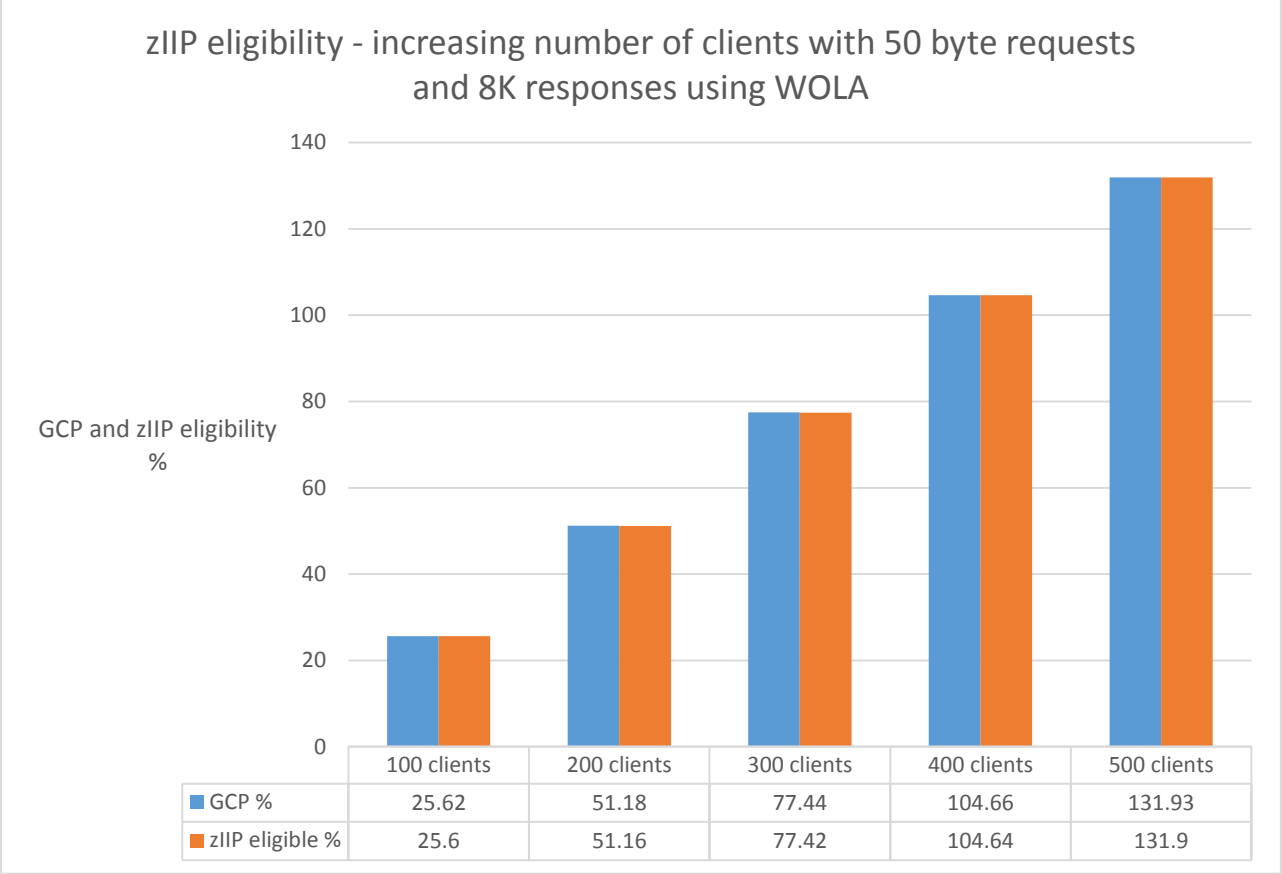


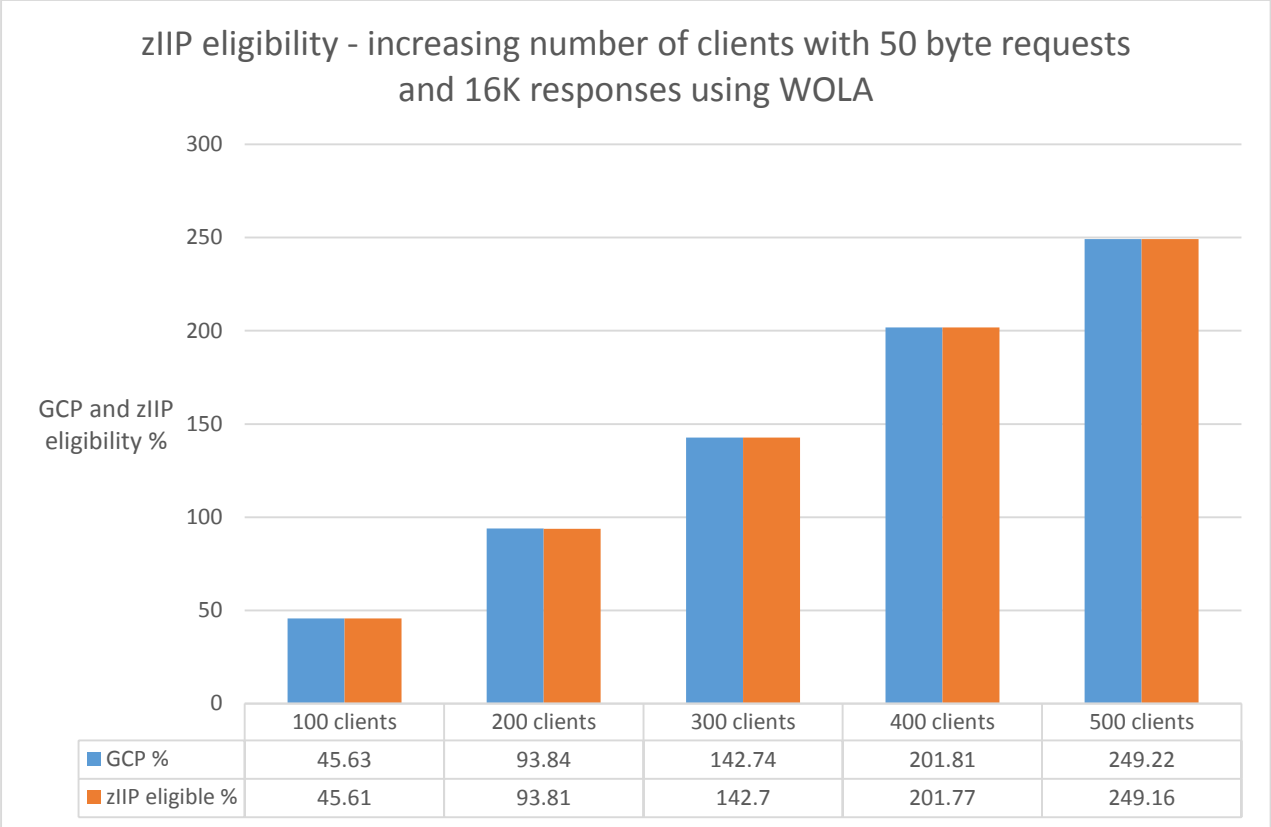
Figure 24 GCP CPU % and zIIP eligibility for increasing numbers of clients with 8K responses

Observations

- ✓ z/OS Connect EE is a Java-based product and so over 99% of the product is eligible to be offloaded to zIIP.
- ✓ The potential usage of a zIIP scaled well with the increasing numbers of clients.

9.4 zIIP Eligibility for 16K responses

Figure 25 shows the GCP % usage and zIIP eligibility for 16K responses.



	100 clients	200 clients	300 clients	400 clients	500 clients
GCP %	45.63	93.84	142.74	201.81	249.22
zIIP eligible %	45.61	93.81	142.7	201.77	249.16

Figure 25 GCP CPU % and zIIP eligibility for increasing numbers of clients with 16K responses

Observations

- ✓ z/OS Connect EE is a Java-based product and so over 99% of the product is eligible to be offloaded to zIIP.
- ✓ The potential usage of a zIIP scaled well with the increasing numbers of clients.

10. Conclusions

Customers should consider the following:

1. A single z/OS Connect EE V2.0 server managing payloads to WOLA containing large numbers of array elements that used the API mapping feature and transformed data between JSON and byte array data structures, demonstrated good scalability for
   - ✓ Transactions Per Second
   - ✓ Average Response time of requests
   - ✓ CPU Cost Per Transaction
   - ✓ CPU % Usage
   - ✓ Potential zIIP offload.
2. The WOLA service provider scaled well.
3. z/OS Connect EE V2.0 is almost entirely written in Java and so would benefit from offloading work to one or more zIIPs.

Note:

1. The payload sizes for the responses in this report are up to 16K.
2. Analysis of other payload sizes, or different hardware, have not been completed at this time, so there is no guarantee that equivalent observations will be seen in other configurations.
3. 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.