ORACLE

Improved Performance of Workloads with Oracle Database

Leverage the Oracle database Prefetch Size and EnterpriseOne Application DB Fetch Size Options to Improve Performance of your JD Edwards Workloads

October, 2022 | Version 2.0

Purpose Statement

This document provides an overview of features and enhancements included in Oracle JD Edwards EnterpriseOne Tools Release 9.2.7. It is intended solely to help you assess the business benefits of upgrading to Oracle JD Edwards EnterpriseOne Tools release 9.2.7 and configuring the database fetch size to get improved performance.

Disclaimer

This document, in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement, nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Table of Contents

List of Figures

List of Tables

Executive Summary

In today's competitive world, the pace of business is an important factor as it determines the profitability and growth for organizations. Pace of business is dependent on the efficiency and pace of operations within the organization which requires robust and performant IT systems, ERP being one of them. A performant ERP system enables faster completion of day-to-day activities and reduces turnaround times for key operations. This results in growth in business volumes, better productivity, and profitability.

JD Edwards EnterpriseOne consists of multiple components like hardware, storage, network, load balancer, web server, enterprise server, database server etc and performance of each of these components play a significant role in delivering optimal level of performance to customers. The networking layer is a key component as it interconnects all the JD Edwards components and handles all the traffic related to the workloads. Due to this foundational nature of the networking layer, it has an impact on the performance of every component, its interaction with other components and the overall end-to-end performance of JD Edwards.

JD Edwards components send and receive data across the network in multiple roundtrips, especially with the database, and this results in high network traffic. Due to this, any latency on the networking layer has a compounding impact on the overall performance of transactions and batch processes. To avoid performance issues due to network latency, JD Edwards recommends that all the components within the EnterpriseOne architecture be ideally located as closely as possible. It is recommended that in an on-premise deployment, the components be located in the same physical data center, in a cloud deployment, and the components be located in the same availability domain of a region. This topology reduces the number of network hops significantly, thereby reducing the impact of latency on overall performance.

JD Edwards implementations vary across customers with different networking topologies and infrastructure resulting in different network latencies. This results in variations in end user performance. Networks with high latencies have significant impact on the overall performance of JD Edwards as every round-trip of data to the database is impacted by the high network latency.

Each database request from a JD Edwards component to the database, fetches a pre-determined number of records from the database. The number of records fetched are based on a pre-defined default configuration setting of 10 for the JDBj JDBC driver and 100 for the JDB OCI Oracle Call Interface driver and it cannot be modified by customers. The same fetch size is used for any JD Edwards deployment irrespective of the network latency. When the number of records to be fetched for processing are large, there are multiple requests made to the database, which increases the network traffic, which in turn impacts the performance significantly in a high latency network.

Increasing the number of records fetched per database request aids in reducing the number of round-trips on the network between JD Edwards components and the database in high network latency architectures which significantly improves the performance. With tools release 9.2.7, JD Edwards introduces the ability for customers to configure their database fetch size based on their network performance and obtain optimal performance of their workload on Oracle database. This document introduces the concept of fetch size for database queries and the methodology for tuning it to achieve better performance of JD Edwards in high latency network topologies.

Overview of Database Fetch Size

The fetch size that is used internally by the JD Edwards EnterpriseOne architecture controls the number of rows fetched for every database call made from the JD Edwards EnterpriseOne layer to the database. In practice, the fetch size is restrictive when there are multiple calls or in the situation where there is high latency between the EnterpriseOne components and the Oracle database.

The large numbers of back-and-forth communications leads to overall slowness in the EnterpriseOne application and transaction response time. A solution for improving performance impacted by network latencies is to modify the fetch size exposed from the EnterpriseOne application. Doing so enables customers to take advantage of this tuning where network latency is high and there are high numbers of database calls. The most prominent area of the EnterpriseOne application where there are high numbers of calls is in the batch processing. The solution of

modifying the fetch size is available for Oracle database on-premise and all Oracle database deployment options on Oracle Cloud Infrastructure.

Network Latency

Network latency is a time delay or the duration it takes for data to travel between the sender and the receiver or between a specific user action and the response. It is important to note that network latency in an EnterpriseOne architecture in the context of this document is concerned with the time delay between the Enterprise server as well as the Oracle database and between the HTML server and the Oracle database.

Typically, the network latency on different connection types is provided below:

- **Fiber:** Typical latency would be 10 to 12 ms
- **DSL:** Typical latency would be 11 to 40 ms
- **Cable:** Typical latency would be 13 to 27 ms
- **Satellite:** Typical latency would be 594 to 612 ms

In addition to the connection type, below are examples of other factors that can affect the network latency:

- Network architecture where servers are separated by great distances or are on disparate networks
- Network devices that separate the EnterpriseOne server components (Enterprise, HTML, Oracle database) have inherent network latencies such as firewalls, load balancers, routers, switches, etc.
- To complete a user action or EnterpriseOne transaction, the application may need to perform multiple database fetches, thus the 'chattier' the EnterpriseOne process is with the database, the higher the network latency.

High latencies can make any software application sluggish and can impact the normal business processes of a customer. To combat this problem, the database fetch size for both the communications between the Enterprise server and HTML server to the Oracle database can be adjusted to assist in lowering the effect of this network latency and thus provide a better user experience with the software.

For the purposes of this discussion on database fetch size, two different JD Edwards EnterpriseOne setups with varying network latency are considered:

- Low network latency architecture with a network latency of 0.204 seconds (204 ms)
- High network latency architecture with a network latency of 1.589 seconds (1589 ms)

EnterpriseOne and Oracle Database Client Configurations

There are two database fetch size configurations for the EnterpriseOne application:

1. OCI (Oracle Call Interface) fetch are calls between the Enterprise server and Database server and the JDBj JDBC fetch size, calls between the HTML server and Database server.
2. A parameter to change the database prefetch size on the Oracle database client through the oraaccess.xml file. Each of the EnterpriseOne parameters is exposed with EnterpriseOne Tools Release 9.2.7 using Server Manager Console and through the Oracle database oraaccess.xml file.

Oracle Database Client (oraccess.xml, prefetch)

The oraaccess.xml file sets the number of prefetch rows for all queries. Setting this parameter appropriately can help reduce roundtrips to the database, thereby improving application performance. Typically, a prefetch size for small SQL queries, such as those in an interactive application, would be configured with a prefetch size of 1. Larger result sets because of SQL queries, such as those in a batch application would start the tuning configured using the default prefetch size of 50.

Figure 1 below shows configuring of the Oracle database prefetch size in an oracess.xml file

```
-->
 <oraaccess xmlns="http://xmlns.oracle.com/oci/oraaccess"
  xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
  schemaLocation="http://xmlns.oracle.com/oci/oraaccess
  http://xmlns.oracle.com/oci/oraaccess.xsd">
  <default_parameters>
    <prefetch>
      <rows>50</rows>
    </prefetch>
    <statement_cache>
      <size>100</size>
    </statement_cache>
    <result_cache>
      <max_rset_rows>100</max_rset_rows>
      <!-- 'K' or 'k' for kilobytes and 'M' or 'm' for megabytes -->
      <max_rset_size>10K</max_rset_size>
      <max_size>64M</max_size>
    </result_cache>
  </default_parameters>
</oraaccess>
```

Figure 1. Configuring Prefetch Size in oraccess.xml (Oracle database client) on an Enterprise Server

EnterpriseOne Configuration in the JDE.INI and JDBJ.INI files

Configuring these files enables users to set the maximum number of rows for a buffered fetch from the database when a SQL statement is executed, and SQL cursor is returned. Another buffered fetch retrieves the next set of maximum number of records when the previous buffered fetch is consumed from the SQL query.

Application code keeps reading from the buffered fetch cache row-by-row and will not access the database till the maximum number of buffered fetched rows are processed.

The default database fetch size is 100 for both Oracle Call Interface (Enterprise server) and JDBC (HTML server) database fetches. A configuration of 1000 for both parameter values is a good starting point for any customer evaluation of this parameter especially those EnterpriseOne applications that have heavy batch processing.

Configuration for MaxFetchNumRows and MaxFetchNumBytes in JDE.INI and statementFetchSize in JDBJ.ini

JDE.INI Configuration File
[DB SYSTEM SETTINGS]
MaxFetchNumRows=100
MaxFetchNumBytes=64000

Each of these values are evaluated and whichever is less in a query is favoured. For example, if the MaxFechNumRows is set <100 then number of bytes is adjusted. The code is modified so that it returns an array instead of a single row fetch which results in the performance gains in a high latency network.

For tuning purposes, the MaxFetchNumRows and MaxFetchNumBytes should be increased or decreased in a proportional manner. For example, if the MaxFetchNumRows is increased to 200, then the MaxFechNumBytes should also be doubled to 128000.

JDBJ.INI Configuration File
[jDBC-RUNTIME PROPERTIES]
statementFetchSize=0

This JDBJ.INI setting is new to EnterpriseOne and enables further configuration. A setting of 0 uses the default value of the Oracle database JDBC driver default; this value can change between Oracle database versions. A non-zero value overrides the default JDBC driver default if it is higher than default JDBC driver value.

The above settings can be also configured using Server Manager Console in the EnterpriseOne Server Manager Guide (E61438)
https://docs.oracle.com/en/applications/jd-edwards/cross-product/9.2/eoism/index.html

Database Fetch Size Testing

As shown in Table 1, the database fetch size testing includes a 100-user interactive load test across three of the most heavily used modules in the EnterpriseOne application with a load test of 12 batch processes from different modules. Table 2 provides a detailed description of interactive and batch processes used in testing the database fetch size.

S.NO	TESTCASE TYPE	LOAD TESTING
1	Batch	Submission of12 batch processes as listed below are categorised as query only batch processes, and batch process with SQL insert/update/delete operations. Query Only Batch Processes: • R0004P (User Defined Codes Record Types Print) • R0006P (Business Unit Report) • R00067 (Business Unit Translation Report) • R0008P (Date Patterns Report) • R0010P (Company Constants Report) • R0012P1 (AAI Report - One Line per AAI) • R0014 (Payment Terms Report) • R0018P (Tax Detail Report) • R00425 (Organization Structure Report) • R01402W (One Line per Who's Who Report) Batch processes with SQL Insert/Update/Delete Operations: • R31410 (Orders Processing) • R3483 (Material Requirement Planning (MRP) Processing) • R42565 (Print Invoices) • R43500 (Purchase Order Print)
2	Interactive	All the Interactive scripts covering the below EnterpriseOne modules were executed together as a 100 User test. • SCM – Supply Chain Management • HCM - Human Capital Management

		• SRM - Supplier Relationship Management • CRM - Customer Relationship Management • FMS - Financial Management Systems • GoToEnd Processing (P01012)

Table 1. EnterpriseOne Detail Testing Use Cases

For comparison purposes, all the test cases executed as a part of this technical brief were evaluated with both the default and tuned fetch size.

GoToEnd Processing

GoToEnd Process for Address book revisions application (P01012) is also included as a part of interactive load testing. It initiates a larger number of database requests unlike most of the usual interactive processes. Thus, it is a good performance metric to evaluate as a part of this testing. The test that performs a GoToEnd on an Address book revisions table returns 75k records as part of its processing.

Determining Tuned Database Fetch Size

Database fetch size tuning depends on the type of SQL queries that the interactive or batch job submitted to the database. Unit tests with a single batch process and GoToEnd interactive process are performed with a different range of values for each configuration parameter and evaluated by performance metrics. Performance metrics refers to the total runtime for a batch process and the server-side response time for an interactive process.

Batch Testing	oraaccess.xml Prefetch Size	JDE.INI MaxFetchNumRows	JDBJ.INI statementFetchSize	JDE.INI MaxFetchNumBytes	Runtime (minutes)
R42565	1000	100	100	64000	15.28
R42565	1250	100	100	64000	14.82
R42565	1500	100	100	64000	15.15
R42565	1750	100	100	64000	15.15
R42565	1250	750	750	480000	15.93
R42565	**1250**	**1000**	**1000**	**640000**	**13.87**
R42565	1250	1250	1250	800000	15.77
R42565	1250	1500	1500	960000	16.60

Table 2. Range Of Fetch Sizes Used For Batch Unit Testing

Table 2 above illustrates how the value of the tuned database fetch size was chosen with batch unit testing. In the case where the Oracle database oraaccess.xml file was modified, a value of 1250 was selected. This value favors batch processes and when comparing the benefit of batch versus interactive users, batch was chosen as the preferred solution because it provided the largest benefit to overall performance. Table 2 shows that batch process has an optimal runtime of 14.82 minutes with database prefetch size configuration of 1250 when compared to other sizes.

For the database fetch size on the Enterprise server, the batch process also has an optimal runtime (13.87 minutes) when fetch size in JDE.INI and JDBJ.INI was set to 1000. A value of 1000 was selected as the tuned value for testing comparison. For consistency, and since the latency between the Enterprise server and HTML server were the same (1.589 seconds latency), the JDBJ.INI StatementFetchSize of 1000 was also selected. GoToEnd process was also performed as a part of Interactive unit testing with the same set of database prefetch sizes shown in Table 2. An analysis of both the batch and interactive unit test results concludes the tuned fetch size for the testing purpose of this document.

Table 3 refers to the default and tuned database fetch size values that were used in the testing process.

TYPE	DB PREFETCH SIZE	APPLICATION FETCH SIZE		
	oraccess.xml	MaxFetchNumRows	MaxFetchNumBytes	statementFetchSize

Default	50	100	64000	0
Tuned	1250	1000	640000	1000

Table 3. Default and Optimal Fetch Sizes

Metrics

Metrics provides an indication whether the application is performing within the range of expected parameters or whether the performance is within acceptable standards. Metrics listed below are collected in this document and used for comparison when collected with default and tuned fetch sizes to frame the result analysis.

- Server-side response – JMeter utility metrics are collected during the processing of interactive user applications. JMeter collects the server HTML request and response of an application action request to the WebLogic HTML Server responsible for handling browser requests. JMeter does not collect any performance metrics that involves browser processing.
- Batch runtime – The processing time of a batch job from submission to completion (Done 'D' status). Batch load test is categorised as two types 1) Query only batch processes, and 2) Other batch processes with SQL insert/update/delete or more complex DML SQL operations. Refer to Table 1 for a detailed description of the batch processes used in the test.

EnterpriseOne Database Fetch Size Testing Results

This section discusses the outcome of the batch only, interactive only, and combined interactive and batch testing performed on EnterpriseOne with different network latency configurations. The metric that is measured for Figures 2 and 3 is the batch runtime of the EnterpriseOne process being tested.

High Network Latency Architecture - Complex Batch Profile Results

The first area of testing is on the high network latency architecture with batch. The batch that was tested included three EnterpriseOne batch processes that are in increasing complexity of DML operations (SQL insert/update/deletes) and number of rows processed and is presented in Figure 2 and are initiated as a standalone process on the Enterprise Server.

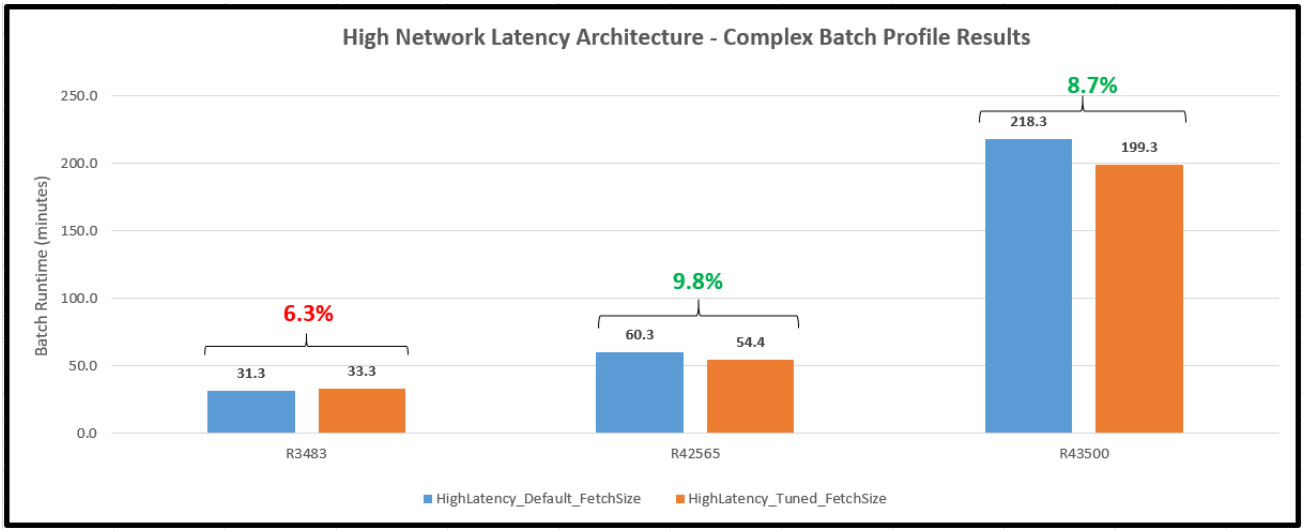

Figure 2. High Network Latency Architecture - Complex Batch Profile Results

As Figure 2 indicates, the average runtime marks performance gains of 9.8% (5.9 minutes) for batch R42565 (Print Invoices) and 8.7% (19 minutes) for batch process R43500 (Purchase Order Print). The batch process R3483 (Material Requirement Planning (MRP) Processing) shows a degradation of 6.3% (2 minutes) for the database fetch size comparison between the default and tuned configurations.

The assumption was that a tuned database fetch size should benefit the batch processes the most and in circumstances where there is high latency. Instead, it was observed that for R3483, this was not the case because a degradation was observed. The simple explanation is that the R3483 batch process does not have the complexity of DML operations in its execution to benefit from the tuned database configuration changes. Therefore, customers must be careful in implementing the database fetch size changes broadly because it can have a slight negative affect on some EnterpriseOne processes. With that stated, it is believed that the overall benefit of changing the database fetch size, in effect, outweighs any slight degradation that might occur in other processes.

Low Network Latency Architecture - Complex Batch Profile Results

A similar test was initiated in a low network latency architecture using only the three complex batch profile EnterpriseOne processes. The results are presented in Figure 3 below.

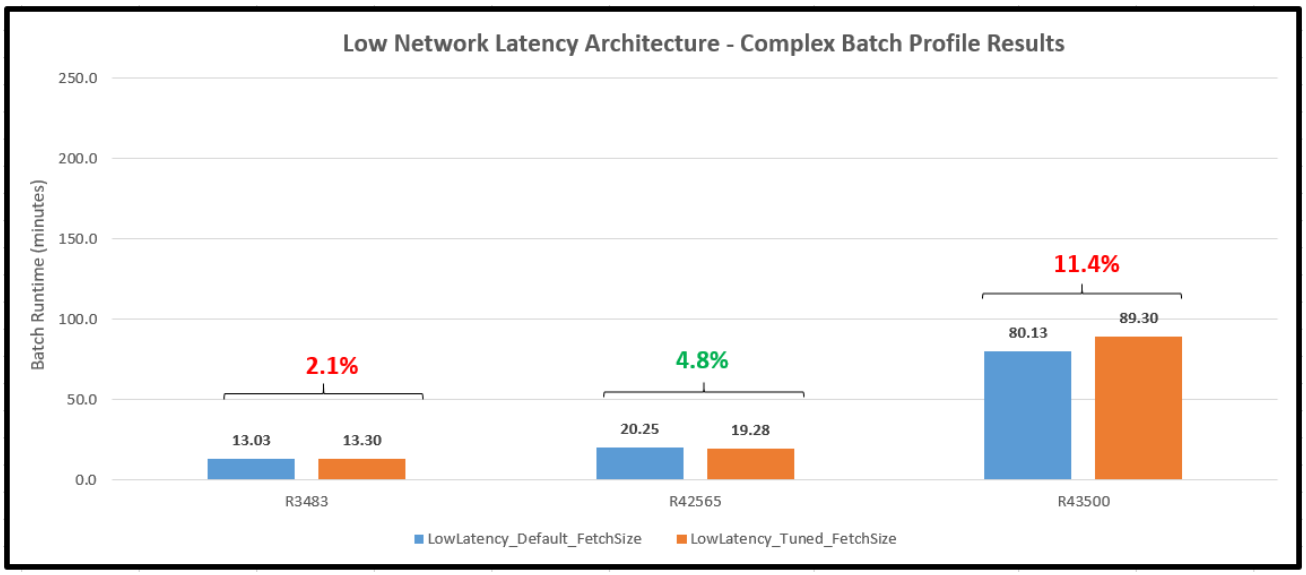

Figure 3. Low Network Latency Architecture - Complex Batch Profile Results

Figure 3 indicates gains in the batch process R42565 (Print Invoices) of 4.8% (0.97 minutes) with tuned database fetch size whereas the other batch processes observe a decrease in runtime averages.

The negative impacts in two of the three controlled EnterpriseOne batch processes illustrates a few possible conclusions. First, for low network latency architecture, it may be more prudent to leave the database fetch sizes to their default values for the EnterpriseOne JDE.INI and JDBJ.INI configurations. Secondly, a customer may want to first evaluate changing only the prefetch default value in the Oracle oraacess.xml file and then subsequently the EnterpriseOne configurations.

The guidelines that should be followed is to first tune the Oracle oraaccess.xml file, which should benefit the performance if there are any improvements to the prefetching of larger number of rows for SQL queries. This tuning is application batch specific and depends on the data result sets returned by the batch in its processing. This improvement is strictly from the Oracle database server, any benefit from the application code comes from tuning the EnterpriseOne configuration. Tuning EnterpriseOne can then be evaluated for improvements in processing using the EnterpriseOne JDE.INI and JDBJ.INI configuration variables.

Low network latency and high network latency architectures - Query Batch Profile Results

The results of batch load testing on both low network latency and high network latency architectures, where the EnterpriseOne batch process executes SQL select statements as its main database fetch size profile, are presented in Figure 4. As before, the key performance indicator for comparison is the total batch runtime.

The SQL query only EnterpriseOne batch testing consisted of initiating all 10 of the batch processes in Table 1 together and collecting the average runtime of the processes over a period of one hour. As is characteristic of SQL query only batch, these batches process in a relatively short amount of time ranging from 2-12 seconds.

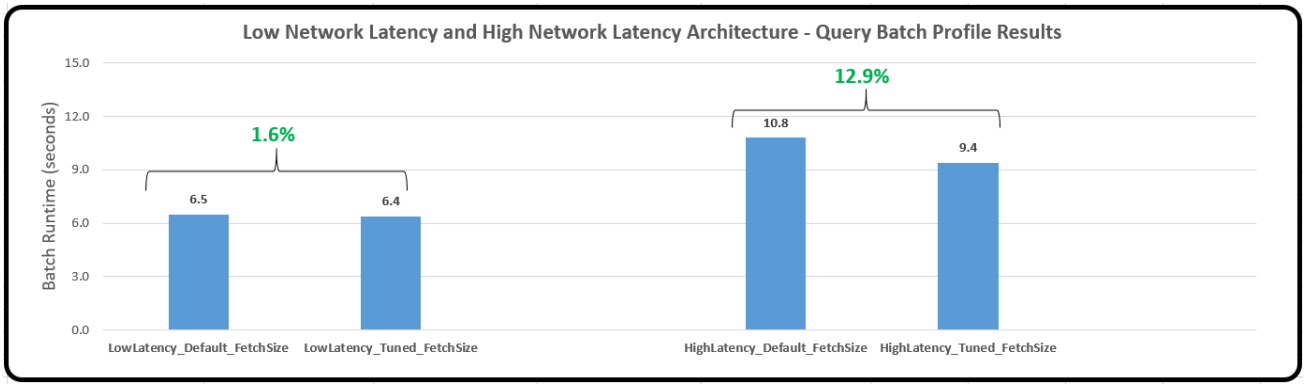

Figure 4. Low Network Latency and High Network Latency Architectures - Query Batch Profile Results

Figure 4 designates that average runtime for the query only batch processes has a benefit of 1.6% (100 ms) on the low network latency architecture. Execution of the same query only batch load on a high network latency architecture marks an improvement of 12.9% (1200 ms) with the tuned database fetch size configuration.

Results in this case of running SQL query batch profiles indicate that the performance gain illustrates a larger benefit to the high network latency architecture as compared to the low network latency architecture. This result reflects the assumption that changing the database fetch size will benefit processes with high network latency characteristics.

High network latency architecture - Interactive and Combined Interactive and Batch Load

The final round of tests is the interactive and the combined interactive and batch load tests performed on the high network latency architecture. Normally, the EnterpriseOne application runs the interactive user load or the batch processes in isolation. Further, a normally functioning EnterpriseOne environment has both running concurrently. Figure 5 below is the presentation of these all-encompassing test results.

When interactive user processes are combined with the batch processes, all 14 batch processes are initiated as described in Table 1. The key metric that is measured for the comparison is the effect of the load changes with regard to the JMeter server-side response time.

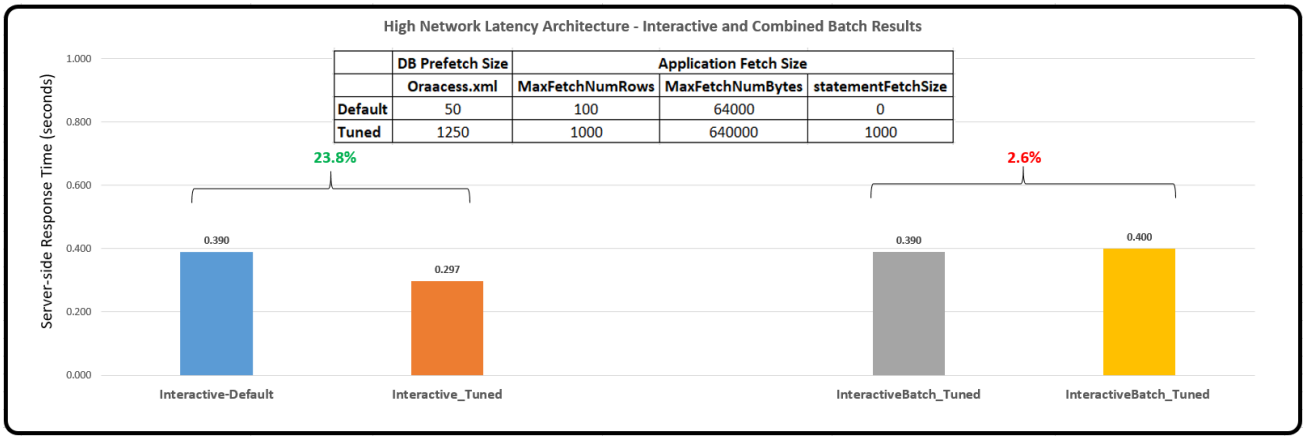

Figure 5. High Network Latency Architecture - Interactive and Combined Interactive/Batch Load Test Results

Figure 5 illustrates that in high network latency architectures, a gain of 23.8% (93 ms) is observed in the case of interactive only load with the database tuned fetch size. With an additional load of batch processes, response time marks a 2.6% (10 ms) decrease with the database tuned fetch size.

The results in Figure 5, and the analysis of the data, suggests that the database fetch size for high network latency architectures may compete with each other with regard to both interactive only and combined interactive and batch processes. The metric of a gain of 2.5% (10 ms) is not considered noticeable to an interactive user. While there is a benefit to the batch processing with the tuned database fetch size, there is no noticeable effect on the interactive user processes.

Figure 5 above further emphasized that performance in a heterogeneous environment requires special analysis and understanding. In this case, performance in a homogeneous load testing of an interactive load can have a higher benefit overall (90 ms) than a similar load where interactive SQL is introduced along with the batch load (10 ms). The analysis simply means that a customer will likely experience fluctuations in performance in a complex load scenario. Oftentimes, the benefit in a heterogeneous environment affects the performance when only one type of SQL load is initiated on the database.

GoToEnd Process

The GoToEnd processing is also added to the interactive load and evaluated for a high network latency architecture as presented in Figure 6. The metric of evaluation for the GoToEnd additional interactive processing remains the JMeter server-side response times.

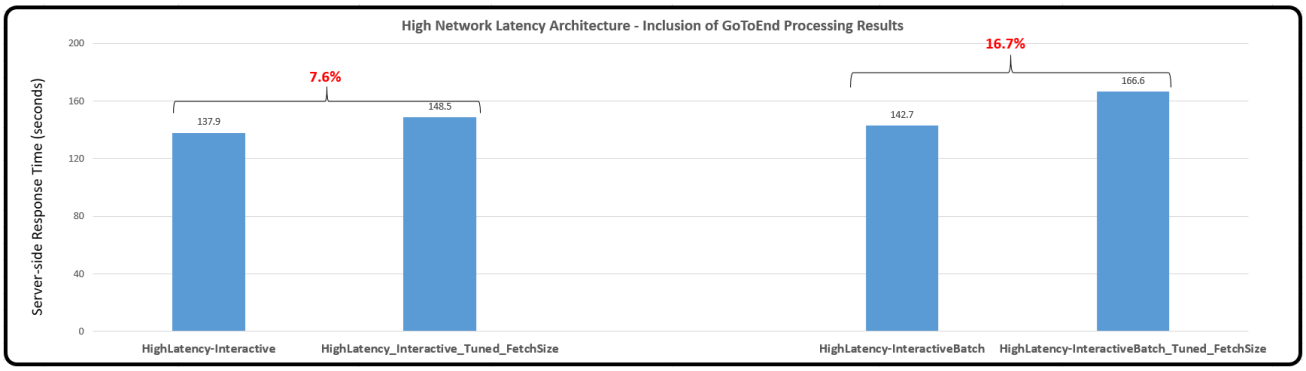

Figure 6. High Network Latency Architecture - Inclusion of GoToEnd Processing Results

Figure 6 shows a performance degradation of 7.6% (10.6 seconds) for GoToEnd process with interactive only load and 16.7% (23.9 seconds) with interactive and the complete 14 batch process load, as described in Table 1.

This is a significant finding because it shows a significant difference when a single new interactive process is introduced into the testing. GoToEnd differs in many ways in the process flow than does a batch process. In the GoToEnd processing, as data is retrieved from the database it is processed line-by-line. This form of processing negates benefits that would normally be observed with high network latency environment.

Customers must take this into consideration when deciding to implement a change to the database fetch size. Fortunately, GoToEnd processing may not be a frequent occurrence with end-user interactions with the EnterpriseOne application and these momentary performance degradation events can be discounted. This is presented as another factor in the evaluation of changing the database fetch size.

Low network latency architecture - Interactive and Combined Interactive and Batch Load

The interactive and combination of interactive and batch tests for low network latency architecture are presented here and illustrated in Figure 7. The metric for comparison for performance is the JMeter server response time.

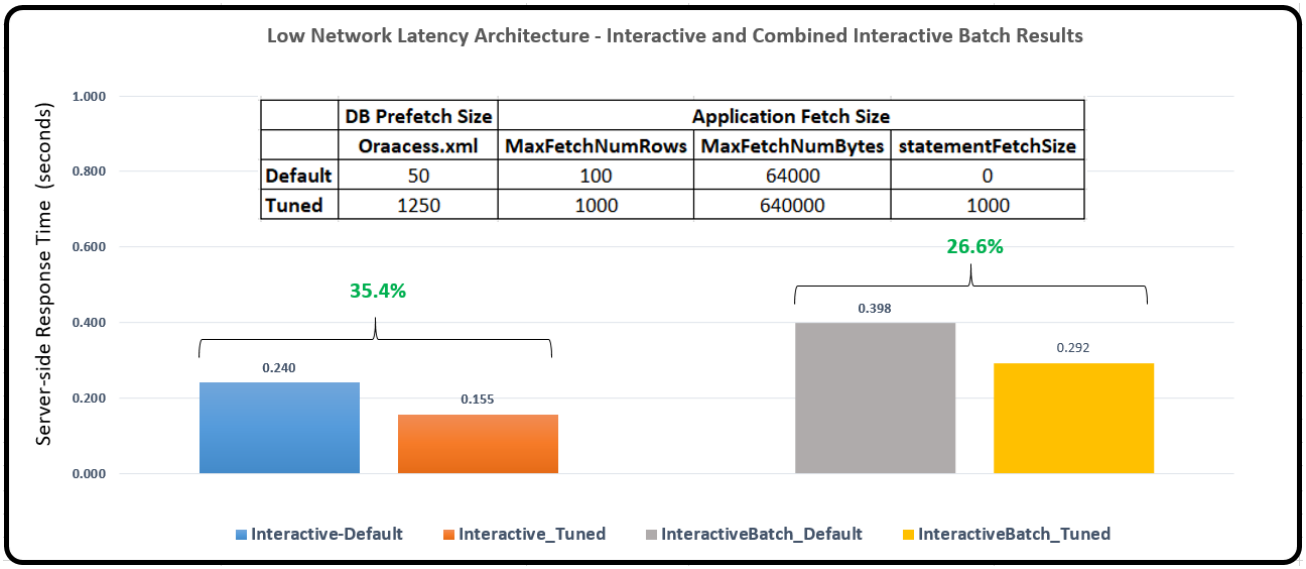

Figure 7. Comparing Average Server-side Response Time of Interactive and Batch Load Tests

Figure 7 above shows a gain of 35.4% (85 ms) in the case of interactive load only between the default and tuned database fetch size. The server-side response time increased in performance by a lesser value of 26.6% (106 ms)

when the batch load is running concurrently with the interactive processes but had increased processing time from 0.240 seconds to 0.398 seconds and from 0.155 to 0.292 seconds, respectively. Figure 7 continues to demonstrate the varying performance in a heterogeneous environment.

GoToEnd Process

GoToEnd processing is also evaluated on a low network latency architecture as illustrated in Figure 8.

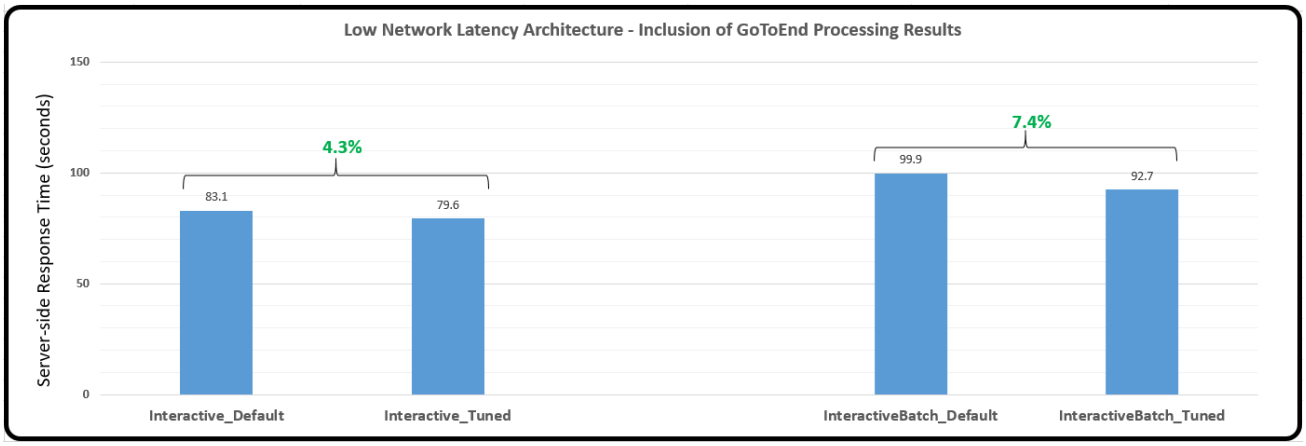

Figure 8. Low Network Latency Architecture - Inclusion of GoToEnd Processing Results

Figure 8 depicts that the JMeter average server-side response time gains 4.3% (3.5 seconds) with interactive only testing whereas it increases to 7.4% (7.2 seconds) when batch is run concurrently. As stated before, adding varying heterogeneous loads to testing can and will shift the overall performance benefit. In this case, there is an improvement of 3.5 and 7.2 seconds respectively in overall performance between the default and tuned values and between the GoToEnd homogeneous processing or initiated heterogeneously with a batch load.

Figure 8 also shows that in this case of low network latency architecture, under full loaded conditions including GoToEnd, its benefits can be observed.

Conclusion

Choosing the correct database fetch size for your organization's EnterpriseOne application architecture may take several trial runs before finding the value that best fits the business needs. There are three areas where database fetch size can be modified, and depending on the existence of network latency, one or all of these areas are tunable. This document, and its results, represent testing in a controlled EnterpriseOne environment. These results were found to be a good first value of database fetch size. A customer might have different results, but the methodology of selecting the database fetch size is the same.

As seen in the results section presented above, the change in database fetch size is most beneficial for batch processes because EnterpriseOne application batch controls the largest number of database calls with the greatest number of rows retrieved from the application. It is suggested that if there is high latency between either the Enterprise server, HTML server or both, the tuned value would be a better starting point.

That is, the database fetch size is dependent on more than one factor, namely the number of database fetches, the number of network round trips that the Enterprise server must complete for a user transaction, and whether that be interactive or batch. In customer deployments, network latencies between the EnterpriseOne server components differ, as well as the database SQL issued, and database result sets returned, which makes choosing the tuned database fetch size unique to each customer environment.

The results presented in this document also illustrate a common problem in tuning, which is that a 'one size fits all' approach is not practical. Depending on the workload, network latency being high or low, and the three varying tuning values with the Oracle oraaccess.xml, and the two areas of tuning within the configuration of the EnterpriseOne system, tuning for one area of the EnterpriseOne application may, in fact, have a small or even large effect on other areas of the application. With this in mind, it is recommended that customers tune carefully and test with isolated and full load on the system that fits the best needs of the business processes.

Appendix A: Test Configuration

The JD Edwards EnterpriseOne components that were configured in the architecture for the testing process discussed in this document are listed below:

The JD Edwards EnterpriseOne components were implemented on Oracle Cloud Infrastructure as a standard VM.

JD Edwards EnterpriseOne Enterprise Server:

- Oracle Linux Server 8.5
- Oracle Database 19c (19.14.0.0.0 and 19.15.0.0.0) client

Database Server:

- Oracle Linux Server 7.9
- Oracle Database 19c (Enterprise Edition Release 19.14.0.0.0 and 19.16.0.1.0 – Production)

HTML Server

- Oracle Linux Server 8.5
- WebLogic Server 14.1.1.0.0 ; Java JDK (1.8.0_272)

Deployment Server:

- Windows Server 2016 Standard
- VM Standard 2.4 with 4 OCPUs
- 4 OCPUs x Intel Xeon Platinum 8167M CPU @ 2.00 GHz
- 60 GB RAM

Server Manager Console:

- Oracle Linux Server 8.5

Test Controller:

- Windows Server 2016 Standard
- VM Standard 2.4 with 4 OCPUs
- 4 OCPUs x Intel Xeon Platinum 8167M CPU @ 2.00 GHz
- 60 GB RAM
- Apache JMeter 5.4.3

Software: JD Edwards EnterpriseOne Application 9.2 Update 4 with Tools Release 9.2.6.3 and Tools 9.2.7

Connect with us

Call +**1.800.ORACLENTERPRISEONE** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle