

Oracle® Enterprise Data Quality

Server Tuning Guide

Release 11g R1 (11.1.1.7)

E40050-02

October 2013

This document covers the server properties that can be tuned to optimize performance of the Oracle Enterprise Data Quality (EDQ) system and describes how these properties should be configured in various circumstances.

EDQ has a large number of properties that are used to configure various aspects of the system. A relatively small number of these are used to control the system's performance characteristics.

Performance tuning in EDQ is often discussed in terms of **CPU cores**. For our purposes, this refers to the number of CPUs reported by the Java Virtual Machine as returned by a call to the `Runtime.availableProcessors()` method.

1 The Properties File

The tuning controls are exposed as properties contained in the `director.properties` file. This file is found under the configuration directory. This file is found in the configuration directory: `oedq_local_home`.

2 The Tuning Properties

The available tuning properties are as follows:

<code>runtime.threads</code>	This property determines the number of threads that will be used for each batch job which is invoked. The default value of this property is zero, meaning that the system should start one thread for each CPU core that is available. You can specify an explicit number of threads by supplying a positive, non-zero integer as the value of this property. For example, if you know that you wish to start a total of four threads for each batch process, set <code>runtime.threads</code> to four.
<code>runtime.intervalth reads</code>	This property determines the number of threads that will be used by each process when running in interval mode. This will also define the number of requests that can be processed simultaneously. The default behavior is to run a single thread for each process running in interval mode.

<code>workunitexecutor.outputThreads</code>	This property determines the number of threads that will be used to write data to the results database. These threads service the queue of results and output data for the whole system, and so are shared by all the processes which are running on the system. The default value of this property is zero, meaning that the system should use one output thread for each CPU core that is available. You can specify an explicit number of output threads by supplying a positive, non-zero integer as the value of this property. For example, if you know that you wish to use a total of four threads for each batch process, set <code>workunitexecutor.outputThreads</code> to 4.
---------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3 Tuning for Batch Processing

The default tuning settings provided with EDQ are appropriate for most systems which are primarily used for batch processing. Enough threads are started when running a job to use all available cores, and if multiple jobs are started then the operating system can schedule the work for efficient sharing between the cores. Operating systems have been carefully engineered, over several decades, to handle these kind of workloads, and it is generally counterproductive to attempt to second-guess them.

4 Tuning for Real-Time Processing

If a system is being used for a significant amount of real time processing, then the tuning requirements are more complex. A system should not be used for simultaneous batch and real time processing unless the real time response is not critical. Simultaneous batch and real time processing is likely to be appropriate only for development systems, and not for production scenarios. The only batch processing that should occur on a system used for real time processing in a production environment should be any data processing that is necessary for use by the real time processes.

4.1 Batch Processing On Real-Time Systems

Ideally, batch processing will only be performed when the real time processes are stopped, such as during the scheduled maintenance window. In this case, the default setting of `runtime.threads` is appropriate. However, this is not always possible. If it is necessary to perform some data preparation whilst real time services are running, then `runtime.threads` should be set to a value which is less than the total number of cores. If this is not done, the data preparation process will place load on all the available cores when it runs. Any real time service requests which arrive during the data preparation period will then be competing with the data preparation process for CPU time. Clearly, this scenario can be handled by the operating system scheduler, but the latency on servicing real time requests would inevitably increase. Reducing the number of threads started for the batch processes means that the batch processing operations will not utilize all the cores, thereby leaving some processing capacity available for incoming real time requests.

4.2 Real-Time Thread Numbers

For most production systems the default value of one for `runtime.intervalthreads` is not appropriate. Although it is adequate for development environments, it implies that, for any given real-time service served by a process running in interval mode, all requests will be processed sequentially. If four requests for the same service arrive

simultaneously, and the average time to process a request is 100 ms, then the first message will be processed after 100 ms, the second after 200 ms and so on. In addition, all the work will be performed by a single core, meaning that on a four-core machine three of the cores would be idle, with the remaining single core doing all the work. Setting `runtime.intervalthreads` to the same as the number of available cores would allow incoming requests to be processed simultaneously, resulting in a more efficient use of resources and a much faster turnaround speed.

4.3 I/O Heavy Real-Time Processes

If the process that is in use performs significant I/O, then there could be an argument for increasing the value of `runtime.intervalthreads` above the number of available cores. This is because the process is likely to become I/O bound, resulting in times when all the threads are waiting for disk activity to complete and when one or more cores are therefore idle. Using more active threads than there are cores means that when one thread stalls for I/O, another thread will be able to utilize the core that the thread was using.

4.4 Real-Time Tuning Example

Suppose we have a four-core Intel server which we wish to use to support four different web services. The web services in question are fairly CPU-intensive, and perform minimal amounts of I/O. Some data used by the web services must be updated on a daily basis, which includes running a data preparation process in a batch mode. We expect that the web services will not be under a steady load, but rather will receive intermittent sets of simultaneous requests. Overnight, the web services are stopped for maintenance and data preparation.

In this case, it is appropriate to leave the `runtime.threads` property set to its default value of 'one thread per CPU core'. In this case, four, since we wish the data preparation to occur in the quickest possible time. Since we know that the real time processing must be maximally responsive, and that the process is not likely to become I/O bound, we can also set the `runtime.intervalthreads` property to four, to ensure that we will process the maximum number of requests at the same time, if running in interval mode.

Note: Increasing the value of `runtime.intervalthreads` means that there will be a significant increase in the memory requirement, particularly at interval turnover.

5 JVM Tuning

JVM (Java Virtual Machine) parameters should be configured during installation; see Section 2.4.4. of the EDQ Installation Guide for further details. However, it may be necessary to tune these parameters post-installation to improve performance:

Note: All of the recommendations in this section are based on EDQ installations using the Java HotSpot Virtual Machine. Depending on the nature of the implementations, these recommendations may also apply to other JVMs.

5.1 PermGen space

If the following error message is reported in the log file, it may be necessary to increase the maximum PermGen space available:

```
java.lang.OutOfMemoryError: PermGen space
```

To do this, change the value against the `-XX:MaxPermSize` parameter on the JVM on the EDQ server. It will also be necessary to change the `-XX:ReservedCodeCacheSize` parameter proportionally. For example, if the `MaxPermSize` is doubled from 1024m to 2048m, the `ReservedCodeCacheSize` should be doubled.

5.2 Maximum heap memory

If an `OutOfMemory` error message is generated in the log file, it may be necessary to increase the maximum heap space parameter, `-Xmx`. For most use cases, a setting of 8GB is sufficient. However, large EDQ installations may require a higher max heap size, and therefore setting the `-Xmx` parameter to a value half that of the server memory is the normal recommendation.

6 Database Tuning

The most significant database tuning parameter with respect to performance tuning within EDQ is `workunitexecutor.outputThreads`. This parameter determines the number of threads, and hence the number of database connections, that will be used to write results and staged data to the database. All processes that are running on the application server share this pool of threads, so there is a risk of processing becoming I/O bound in some circumstances. If we have processes which are particularly I/O intensive relative to their CPU usage, and a database machine which is more powerful than the machine hosting the EDQ application server, it may be worth increasing the value of `workunitexecutor.outputThreads`. The additional database threads would use more connections to the database and put more load on the database.

7 Related Documents

For more information, see the following documents in the documentation set:

- *Oracle Enterprise Data Quality Installation Guide*
- *Oracle Enterprise Data Quality Architecture Guide*

See the latest version of this and all documents in the Oracle Enterprise Data Quality Documentation website at:

http://download.oracle.com/docs/cd/E48549_01/index.htm

Also, see the latest version of the EDQ Online Help, bundled with EDQ.

8 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Oracle Enterprise Data Quality Server Tuning Guide, Release 11g R1 (11.1.1.7)
E40050-02

Copyright © 2006, 2013, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

