Increasing Driver Performance

DataDirect Connect® Series ODBC Drivers

Introduction

One of the advantages of DataDirect Connect® Series ODBC drivers (DataDirect Connect for ODBC and DataDirect Connect64™ for ODBC) is their flexibility; they can be fine-tuned to enhance performance. The settings that enhance performance may result in reduced functionality, but often the functionality vs. performance trade-off has no impact on the application. The information in the following sections discusses how to tune the DB2, Informix, Informix Wire Protocol, Oracle, Oracle Wire Protocol, and Sybase Wire Protocol drivers for increased performance.

Tuning the Drivers

The following information explains which driver options you should set in the DataDirect Connect Series ODBC drivers to achieve maximum performance. **Condition** describes the condition under which your application operates. **Action** describes which option to set through the ODBC Driver Setup dialog on Windows or which option to set in the system information file (usually .odbc.ini) on UNIX. **Description** provides an explanation of the option and its affect on performance.

Informix (DataDirect Connect for ODBC only) and Informix Wire Protocol

**Condition:** If your application does not use threads

**Action:**

On Windows, de-select the “Application Using Threads” check box on the Advanced tab.

On UNIX, set ApplicationUsingThreads to 0 (ApplicationUsingThreads=0).

**Description:** The driver coordinates operations from different threads by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If the application does not make ODBC calls from different threads, there is no reason for the driver to coordinate operations. Disabling the “Application Using Threads” option prevents the driver from coordinating operations and improves performance for single-threaded applications.
**Condition:** If your application does not issue SQL Cancel

**Action:**
On Windows, set "Cancel Detect Interval" on the Advanced tab to 0 (None).
On UNIX, set CancelDetectInterval to 0 (CancelDetectInterval=0).

**Description:** If your application uses threads, it may also issue synchronous SQL Cancel calls. In this case, setting a value for "Cancel Detect Interval" allows long-running queries to be canceled if the application issues a SQL Cancel. The value you select determines how often (in seconds) the driver checks whether a query has been canceled using SQL Cancel. If, however, your application does not issue synchronous SQL Cancel calls, you can increase driver performance by setting "Cancel Detect Interval" to 0. In this case, the driver does not incur the overhead of periodically checking for SQL Cancel.

**Oracle Wire Protocol and Oracle Client**

**Condition:** If your application does not use threads

**Action:**
On Windows, de-select the "Application Using Threads" check box on the Advanced tab.
On UNIX, set ApplicationUsingThreads to 0 (ApplicationUsingThreads=0).

**Description:** The driver coordinates operations from different threads by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If the application does not make ODBC calls from different threads, there is no reason for the driver to coordinate operations. Disabling the "Application Using Threads" option prevents the driver from coordinating operations and improves performance for single-threaded applications.

**Condition:** If your application retrieves large result set (> 60000 bytes)

**Action:**
On Windows, set "Array Size" on the Performance tab to the approximate number of bytes of your largest result set.
On UNIX, set ArraySize to the approximate number of bytes of your largest result set.

**Description:** Values for this option can be an integer from 1 to 4,294,967,296 (4GB); the default is 60000. The value 1 is a special value that does not define the number of bytes but, instead, causes the driver to allocate space for exactly one row of data. Larger values increase throughput by reducing the number of times the driver fetches data across...
the network when retrieving multiple rows. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

**Condition:** If your application executes multiple concurrent Select statements

**Action:**

On Windows, set “Cached Cursor Limit” on the Performance tab to the number of concurrent open Select statements.

On UNIX, set CachedCursorLimit to the number of concurrent open Select statements (CachedCursorLimit=xxxxxx).

**Description:** Each concurrent open Select statement requires one Oracle Cursor Identifier. Performance is improved if this Cursor Identifier can be retrieved from a cache rather than being created for each connection. When this option is enabled and a Select statement is closed, the driver stores the Identifier in its cache rather than closing the Identifier. When an Identifier is needed, the driver takes one from its cache, if one is available, rather than creating a new one. Cached Cursor Identifiers are closed when the connection is closed. The default value is 32.

NOTE: This option is available only for the Oracle Wire Protocol driver.

---

**Condition:** If your application executes the same non-prepared Select statements multiple times

**Action:**

On Windows, set “Cached Description Limit” on the Performance tab to the approximate number of frequently executed non-prepared Select statements executed on a single connection.

On UNIX, set CachedDescriptionLimit to the approximate number of frequently executed non-prepared Select statements executed on a single connection (CachedDescriptionLimit=xxxxxx).

**Description:** The driver can cache descriptions of Select statements. If a description is not cached when a non-prepared Select statement is executed, the description must be retrieved from Oracle, which reduces performance. The value of this option corresponds to the number of descriptions that the driver saves for Select statements. These descriptions include the number of columns and the data type, length, and scale for each column. The matching is done by an exact-text match through the From clause. When this option is set to a value other than the default (0), applications that issue a Select statement that returns a few rows repeatedly can realize a significant performance benefit. If the statement contains a Union or a nested Select, the description is not cached.

NOTE: This option is available only for the Oracle Wire Protocol driver.
**Condition:** If your application does not use Oracle synonyms

**Action:**

On Windows, de-select the “Catalog Functions Include Synonyms” check box on the Performance tab.

On UNIX, set CatalogIncludesSynonyms to 0 (CatalogIncludesSynonyms=0).

**Description:** Standard ODBC behavior is to include synonyms in the result sets of calls to the following catalog functions: SQLProcedures, SQLStatistics, and SQLProcedureColumns. Retrieving information about synonyms is very expensive with Oracle. The “Catalog Functions Include Synonyms” option allows you to improve your application’s performance by excluding synonyms from these results sets. If your application does not need synonyms to be returned by these catalog functions, disable this option.

---

**Condition:** If your application fetches Long data from Long/LOB columns of less than 1 MB

**Action:**


On UNIX, set DefaultLongDataBuffLen to 1024 (DefaultLongDataBuffLen=1024).

**Description:** This value specifies the size of the buffer used when fetching data from Long/LOB columns. The buffer size should only be large enough to accommodate the maximum amount of data that you want to retrieve from these types of columns; otherwise, performance is reduced by transferring large amounts of data into an oversized buffer. You will need to increase the value of this option from the default of 1024 if the total size of any Long data exceeds 1 MB. The value must be in multiples of 1024 (for example, 2048).

NOTE: In the Oracle (non-Wire Protocol) driver, this option is not used if the “Optimize Long Performance” option is enabled.

---

**Condition:** If your application does not require result set information at prepare time.

**Action:**

On Windows, de-select the “Describe At Prepare” check box on the Advanced tab.

On UNIX, set DescribeAtPrepare to 0 (DescribeAtPrepare=0).

**Description:** When enabled, this option causes two round trips across the network—one for prepare and one for describe. If your application does not
require result set information at prepare time (for instance, you request information about the result set using SQLColAttribute(s), SQLDescribeCol, SQLNumResultCols, etc. before calling SQLExecute on a prepared statement), you can increase performance by disabling this option.

**Condition:** If your application does not need to support Long columns when using a static cursor

**Action:**
- On Windows, de-select the “Enable Static Cursors for Long Data” check box on the Performance tab.
- On UNIX, set EnableStaticCursorsForLong to 0 (EnableStaticCursorsForLongData=0).

**Description:** This option enables the driver to support Long columns when using a static cursor, but also reduces performance. If your application does not need to support Long columns and does not use static cursors, performance is increased by disabling this option.

**Condition:** If your application fetches Long data

**Action:**
- On Windows, select the “Optimize Long Performance” check box on the Performance tab.
- On UNIX, set OptimizeLongPerformance to 1 (OptimizeLongPerformance=1).

**Description:** When enabled, this option fetches Long data directly into the application’s buffers rather than allocating buffers and making a copy. Also, when enabled, this option decreases fetch times on Long data; however, it can cause the application to be limited to one active statement per connection.

NOTE: This option is available only with the Oracle (non-Wire Protocol) driver. If this option is enabled, the “Default Buffer Size for Long/LOB Columns” option is not used.

**Condition:** If your application does not need to return result sets from stored procedures through Ref Cursors

**Action:**
- On Windows, de-select the “Procedure Returns Results” check box on the Advanced tab.
- On UNIX, set ProcedureRetResults to 0 (ProcedureRetResults=0).

**Description:** When Procedure Returns Results is enabled, the driver returns result sets from stored procedures/functions. In addition, SQLGetInfo(SQL_MULT_RESULTS_SETS) will return “Y” and
SQLGetInfo(SQL_BATCH_SUPPORT) will return SQL_BS_SELECT_PROC. If this option is enabled and you execute a stored procedure that does not return result sets, you will incur a performance penalty.

**Condition:** If any of the following apply:

- you have a batch environment with a low number of users
- your Oracle DBMS is running on a Windows server
- your Oracle server has excess processing capacity and memory available when at maximum load
- you have an application that requires maximum performance at the expense of using more Oracle server resources

**Action:**

On Windows, select Dedicated from the “Server Process Type” drop-down list on the Advanced tab.

On UNIX, set ServerType to 2 (ServerType=2).

**Description:** When using a dedicated server connection, a server process on UNIX (a thread on Windows) is created to serve only your application connection. When you disconnect, the process goes away. The socket connection is made directly between your application and this dedicated server process. This can provide tremendous performance improvements, but will use significantly more resources on UNIX servers. Because this is a thread on Oracle servers running on Windows platforms, the additional resource usage on the server is significantly less. Use this option when you have a batch environment with lower numbers of connections or if you have a performance-sensitive application that would be degraded by sharing Oracle resources with other applications.

NOTE: This option is available only for the Oracle Wire Protocol driver.

IMPORTANT: The server must be configured for shared connections (the SHARED_SERVERS initialization parameter on the server has a value greater than 0) for the driver to be able to specify the shared server process type.

**Condition:** If you are the only user requiring a call to SQLProcedures

**Action:**

On Windows, select the “Use Current Schema for SQLProcedures” check box on the Performance tab.

On UNIX, set UseCurrentSchema to 1 (UseCurrentSchema=1).

**Description:** When enabled, this option specifies that the driver return only procedures owned by the current user when executing SQLProcedures. Also, when this option is enabled, the call for SQLProcedures is optimized by
grouping queries, but only procedures owned by the current user are returned. Enabling this option is equivalent to passing the Login ID used on the connection as the SchemaName argument to the SQLProcedures call.

**Sybase Wire Protocol**

**Condition:** If your application does not use threads

**Action:**

On Windows, de-select the “Application Using Threads” check box on the Advanced tab.

On UNIX, set ApplicationUsingThreads to 0 (ApplicationUsingThreads=0).

**Description:** The driver coordinates operations from different threads by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If the application does not make ODBC calls from different threads, there is no reason for the driver to coordinate operations. Disabling the “Application Using Threads” option prevents the driver from coordinating operations and improves performance for single-threaded applications.

**Condition:** If your application fetches large result sets (> 50 rows)

**Action:**

On Windows, set “Fetch Array Size” on the Performance tab to a larger number of rows than the default value of 50.

On UNIX, set ArraySize to a larger number of rows than the default value of 50 (ArraySize=xx).

**Description:** If the Select Method connection option is set to 0 and your application fetches more than 50 rows at a time, you should set “Fetch Array Size” to the approximate number of rows being fetched. This reduces the number of round trips on the network, thereby increasing performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network. You should use "Fetch Array Size" in conjunction with "Select Method."

**NOTE:** The ideal setting for your application will vary. To calculate the ideal setting for this option, you must know the size in bytes of the rows that you are fetching and the size in bytes of your Network Packet. Then, you must calculate the number of rows that will fit in your Network Packet, leaving space for packet overhead. For example, suppose your Network Packet size is 1024 bytes and the row size is 8 bytes. Dividing 1024 by 8 equals 128; however, the ideal setting for “Fetch Array Size” is 127 **not** 128 because the number of rows times the row size must be slightly smaller than the Network Packet size.
**Condition:** If your application fetches Long data from a very large TEXT or IMAGE column, but your application displays less than 1 MB of data

**Action:**


On UNIX, set DefaultLongDataBuffLen to 1024 (DefaultLongDataBuffLen=1024).

**Description:** TEXT and IMAGE columns can contain very large amounts of data, however, your application may display only a limited amount of data. The buffer size should only be large enough to accommodate the maximum amount of data that you want to retrieve; otherwise, performance is reduced by transferring large amounts of data into an oversized buffer. If your application retrieves more than 1 MB of data, the buffer size should be increased accordingly.

---

**Condition:** If your Sybase server is set up with a packet size other than the default of 512 bytes

**Action:**

On Windows, set “Packet Size” on the Performance tab to the number of 512-byte blocks equal to the Network Packet Size setting on your Sybase ASE Server. For example, Packet Size = 6 is equivalent to 6 * 512 = 3072 bytes.

On UNIX, set PacketSize to the number of 512-byte blocks equal to the Network Packet Size setting on your Sybase ASE Server. For example, Packet Size = 6 is equivalent to 6 * 512 = 3072 bytes (PacketSize=x).

**Description:** It is normally optimal for the client to use the maximum packet size that the server will allow. This reduces the total number of round trips required to return data to the client, thus improving performance.

To take advantage of this connection attribute, you must configure the Sybase server for a maximum network packet size greater than or equal to the value you specified for PacketSize. For example:

```
sp_configure "max network packet size", 5120
reconfigure
Restart Sybase Server
```

---

**Condition:** If your application does not execute prepared statements multiple times

**Action:**

On Windows, set “Prepare Method” on the Performance tab to 2.

On UNIX, set OptimizePrepare to 2 (OptimizePrepare=2).
**Description:** If "Prepare Method" is set to 1 and your application issues calls to SQLPrepare that contain parameter markers, or if "Prepare Method" is set to 0, then the driver creates a stored procedure on the server at prepare time. If your application executes one of these prepared statements multiple times, performance will increase because the driver created a stored procedure on the server. This is because executing a stored procedure is faster than executing a single SQL statement; however, if a prepared statement is only executed once or is never executed, performance can decrease. This is because creating a stored procedure incurs more overhead on the server than simply executing a single SQL statement. When "Prepare Method" is set to 2, the driver never creates stored procedures for prepared statement. A setting of 2 should be used if your application does not execute prepared statements multiple times.

---

**Condition:** If your application does not use multiple active statements per connection (for example, fetching more than one result set at a time)

**Action:**

On Windows, set "Select Method" on the Performance tab to 1.

On UNIX, set `SelectMethod` to 1 (`SelectMethod=1`).

**Description:** The default setting (0) of this option causes the driver to use database cursors for Select statements and allows an application to process multiple active statements per connection. An active statement is defined as a statement where all the result rows or result sets have not been fetched. This can cause high overhead on the server. If your application does not use multiple active statements, however, setting "Select Method" to 1 will increase performance of Select statements by allowing the server to return results without using a database cursor. If this option is set to 0, it should be used in conjunction with "Fetch Array Size." If this option is set to 1, "Fetch Array Size" has no effect.

---

**DB2 Wire Protocol**

**Condition:** If your application does not use threads

**Action:**

On Windows, de-select the "Application Using Threads" check box on the Advanced tab.

On UNIX, set `ApplicationUsingThreads` to 0 (`ApplicationUsingThreads=0`).

**Description:** The driver coordinates operations from different threads by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If the application does not make ODBC calls from different threads, there is no reason for the driver to coordinate operations. Disabling the "Application Using Threads" option prevents the driver from
coordinating operations and improves performance for single-threaded applications.

**Condition:** If your application makes unqualified catalog function calls

**Action:**

On Windows, select the “Use Current Schema for Catalog Functions” check box on the Advanced tab.

On UNIX, set `UseCurrentSchema` to 1 (`UseCurrentSchema=1`).

**Description:** This option speeds up applications that make unqualified catalog function calls over which the end user has no control. For example, applications such as Microsoft Access always pass "no schema" to a call to SQLTables. Setting this option can cause a dramatic improvement in speed when retrieving a table list because, normally, far fewer tables are returned.

---

We welcome your feedback! Please send any comments concerning documentation, including suggestions for other topics that you would like to see, to:

docgroup@datadirect.com
DataDirect Technologies is focused on data access, enabling software developers at both packaged software vendors and in corporate IT departments to create better applications faster. DataDirect Technologies offers the most comprehensive, proven line of data connectivity components available anywhere. Developers worldwide depend on DataDirect Technologies to connect their applications to an unparalleled range of data sources using standards-based interfaces such as ODBC, JDBC and ADO.NET, as well as cutting-edge XML query technologies. More than 250 leading independent software vendors and thousands of enterprises rely on DataDirect Technologies to simplify and streamline data connectivity. DataDirect Technologies is an operating company of Progress Software Corporation (Nasdaq: PRGS).

[www.datadirect.com](http://www.datadirect.com)