Datasource Timeouts and Support for CFQUERY Timeout Attribute

Posted At : January 16, 2006 9:29 AM | Posted By : Steven Erat
Related Categories: Java, ColdFusion, Macromedia

When considering Timeout settings and sources of problematic bottlenecks in ColdFusion applications, a widely held misconception is that all active database connections do not obey timeouts. Discussions surrounding slowness or unresponsiveness in ColdFusion applications often boil down to isolating unusual database activity as the culprit. The explanation provided is often that while the general Timeout in the ColdFusion Administrator, as well as the page level CFSETTING timeout attribute, will be enforced for all page requests except those actively waiting on a response from the database server after a SQL statement has been sent. While this is true, its sometines overlooked that the CFQUERY tag itself has a timeout attribute that is worth considering in some cases. In fact, this particular timeout is something that I've often forgotten, too.

One reason for the confusion over the CFQUERY tag timeout is that not all drivers support it. Of the databases drivers that ship with ColdFusion MX, only the Oracle and SQL Server drivers will enforce the CFQUERY attribute for timeout if it is supplied.

From the ColdFusion documentation for **CFQUERY**, the timeout attribute is described as:

CFQUERY timeout attribute

> Maximum number of seconds that each action of a query is permitted to execute before returning an error. The cumulative time may exceed this value.
>
> For JDBC statements, ColdFusion sets this attribute. For other drivers, check driver documentation.

If you've taken **thread dumps** to examine what the ColdFusion server was doing at the time of the unresponsiveness or hang, you might find some stacks in the thread dumps that appear to be in a JDBC driver in a socketRead state. These stacks correspond to web requests where the JDBC driver is waiting to receive information back from the database. Its quite normal for web requests to enter this state, so a snapshot taken by a single thread dump is almost meaningless, but if the same thread id appears in the same socketRead state over multiple consecutive thread dumps then you have reason to suspect a slow query or database problem.

In these cases where bottlenecks are present and can be attributed to database queries, then you may want to consider using the timeout attribute of CFQUERY to reduce the risk of completely backing up or hanging the server due to an accumulation of such page requests. The next step would be to consider alternatives to improve query performance such as building faster queries, using views, using cached queries via cachedwithin or a shared scope, or CFQUERYPARAM to cache statements.

So far this blog entry has referred primarily to the CFQUERY tag timeout, which will timeout prolonged, active queries after a database connection has been made. For control over connection timeouts see the CF Admin datasource detail page for the Login Timeout:

Datasource Login Timeout

> The number of seconds before ColdFusion MX times out the data source connection login attempt.

The other timeout setting in the CF Admin datasource detail page is just labeled as Timeout, with a paired setting for Interval. This timeout value is how long a connection should be idle before it is removed. So if a connection goes unused for this period of time, then the next time ColdFusion checks that connection it will be removed from the pool. The frequency that ColdFusion checks the connection idle time is set by the Interval setting. The default for the Timeout setting is 20 minutes and the Interval is 7 minutes, so every 7 minutes ColdFusion will check connections for that datasource and remove connections that have been idle for 20 minutes or more. The number of connections idle connections removed from the pool at each interval is hardcoded in ColdFusion to just five. Say you have 20 idle connections hanging around in a pool, this would require at least 28 minutes to be cleaned up.

Timeout (min)

> The number of minutes that ColdFusion MX maintains an unused connection before destroying it.

Interval (min)

> The time (in minutes) that the server waits between cycles to check for expired data source connections to close.

Getting back to the CFQUERY Timeout attribute, here are some examples of some stacks that you might see in a ColdFusion thread dump where the stack shows a web request waiting on a response from the database:

Oracle (supports timeout)

```
"jrpp-749" prio=5 tid=0x010ba3b0 nid=0x24b3 runnable
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.read(SocketInputStream.java:129)
    at macromedia.jdbc.oracle.net8.OracleDataProvider.ProcessPacketStartFromSocket(Unknown Source)
    at macromedia.jdbc.oracle.net8.OracleDataProvider.receive(Unknown Source)
    at macromedia.jdbc.oracle.net8.OracleDataProvider.getArrayOfBytes(Unknown Source)
...<br/><br/>"jrpp-21" prio=5 tid=0x2b676af8 nid=0xd5c runnable [0x378ef000..0x378efdb8]
    at java.net.SocketInputStream.socketRead(Native Method)
    at java.net.SocketInputStream.read(Unknown Source)
    at macromedia.util.UtilSocketDataProvider.getArrayOfBytes(Unknown Source)
    at macromedia.util.UtilBufferedDataProvider.cacheNextBlock(Unknown Source)
    at macromedia.util.UtilBufferedDataProvider.getArrayOfBytes(Unknown Source)
    at macromedia.jdbc.oracle.OracleDepacketizingDataProvider.receive(Unknown Source)
    at macromedia.util.UtilByteArrayDataProvider.receive(Unknown Source)
    at macromedia.util.UtilByteOrderedDataReader.receive(Unknown Source)
    at macromedia.jdbc.oracle.net8.OracleNet8NSPTDAPacket.sendRequest(Unknown Source)
    at macromedia.jdbc.oracle.OracleImplStatement.fetchNext(Unknown Source)
    at macromedia.jdbc.oracle.OracleImplStatement.execute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.commonExecute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.executeInternal(Unknown Source)
    at macromedia.jdbc.base.BasePreparedStatement.execute(Unknown Source)
    at jrun.sql.JRunPreparedStatement.execute(JRunPreparedStatement.java:142)
    at coldfusion.sql.Executive.executeQuery(Unknown Source)
    at coldfusion.sql.Executive.executeQuery(Unknown Source)
    at coldfusion.sql.SqlImpl.execute(Unknown Source)
    at coldfusion.tagext.sql.QueryTag.doEndTag(Unknown Source)
...
```

SQL Server (supports timeout)

```
"jrpp-56" prio=5 tid=0x39b25e68 nid=0x564 runnable [49a1f000..49a1fdb8]
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.read(SocketInputStream.java:129)
    at macromedia.jdbc.sqlserver.SQLServerByteOrderedDataReader
          .readPacketIntoPrimaryBuffer(Unknown Source)
    at macromedia.jdbc.sqlserver.SQLServerByteOrderedDataReader.receive(Unknown Source)
    at macromedia.jdbc.sqlserver.tds.TDSRPCRequest.submitRequest(Unknown Source)
    at macromedia.jdbc.sqlserver.SQLServerImplStatement.execute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.commonExecute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.executeInternal(Unknown Source)
    at macromedia.jdbc.base.BasePreparedStatement.execute(Unknown Source)
    - locked <0x13f0e100> (a macromedia.jdbc.sqlserver.SQLServerConnection)
    at macromedia.jdbc.base.BasePreparedStatementPoolable.execute(Unknown Source)
    at coldfusion.server.j2ee.sql.JRunPreparedStatement.execute(JRunPreparedStatement.java:87)
    at coldfusion.sql.Executive.executeCall(Executive.java:814)
    at coldfusion.sql.Executive.executeCall(Executive.java:749)
    at coldfusion.sql.SqlImpl.executeCall(SqlImpl.java:320)
...<br/><br/>"jrpp-25" prio=5 tid=0x2579ac90 nid=0xc08 runnable [0x2b83e000..0x2b83fdbc]
    at java.net.SocketInputStream.socketRead(Native Method)
    at java.net.SocketInputStream.read(Unknown Source)
    at macromedia.util.UtilSocketDataProvider.getArrayOfBytes(Unknown Source)
    at macromedia.util.UtilBufferedDataProvider.cacheNextBlock(Unknown Source)
    at macromedia.util.UtilBufferedDataProvider.getArrayOfBytes(Unknown Source)
    at macromedia.jdbc.sqlserver.SQLServerDepacketizingDataProvider.signalStartOfPacket(Unknown Source)
    at macromedia.util.UtilDepacketizingDataProvider.getByte(Unknown Source)
    at macromedia.util.UtilByteOrderedDataReader.readInt8(Unknown Source)
    at macromedia.jdbc.sqlserver.tds.TDSRequest.processReply(Unknown Source)
    at macromedia.jdbc.sqlserver.SQLServerImplStatement.getNextResultType(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.commonTransitionToState(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.postImplExecute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.commonExecute(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.executeInternal(Unknown Source)
    at macromedia.jdbc.base.BaseStatement.execute(Unknown Source)
    at jrun.sql.JRunStatement.execute(JRunStatement.java:304)
    at coldfusion.sql.Executive.executeQuery(Unknown Source)
    at coldfusion.sql.Executive.executeQuery(Unknown Source)
    at coldfusion.sql.SqlImpl.execute(Unknown Source)
...
```

Sybase (does not support timeout)

```
"jrpp-79" prio=5 tid=0x858b80 nid=0xdb runnable
    at java.net.SocketInputStream.socketRead(Native Method)
    at java.net.SocketInputStream.read(SocketInputStream.java:86)
    at com.sybase.jdbc2.timedio.RawDbio.reallyRead(RawDbio.java:202)
    at com.sybase.jdbc2.timedio.Dbio.doRead(Dbio.java:243)
    at com.sybase.jdbc2.timedio.InStreamMgr.readIfOwner(InStreamMgr.java:512)
    at com.sybase.jdbc2.timedio.InStreamMgr.doRead(InStreamMgr.java:273)
    at com.sybase.jdbc2.tds.TdsProtocolContext.getChunk(TdsProtocolContext.java:561)
    at com.sybase.jdbc2.tds.PduInputFormatter.readPacket(PduInputFormatter.java:229)
    at com.sybase.jdbc2.tds.PduInputFormatter.read(PduInputFormatter.java:62)
    at com.sybase.jdbc2.tds.TdsInputStream.read(TdsInputStream.java:81)
    at com.sybase.jdbc2.tds.TdsInputStream.readUnsignedByte(TdsInputStream.java:114)
    at com.sybase.jdbc2.tds.Tds.nextResult(Tds.java:1850)
...
```

SequeLink ODBC (does not support timeout)

```
"jrpp-392" prio=5 tid=0x2C103A78 nid=0xac8 runnable
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.read(Unknown Source)
    at java.io.DataInputStream.readFully(Unknown Source)
    at com.merant.sequelink.net.TrEndPoint.recv(Unknown Source)
    at com.merant.sequelink.net.Session.channelRecv(Unknown Source)
    at com.merant.sequelink.net.Session.recv(Unknown Source)
    at com.merant.sequelink.net.Session.recvIIOPHeader(Unknown Source)
    at com.merant.sequelink.net.Session.recv(Unknown Source)
    at com.merant.sequelink.net.Session.recv(Unknown Source)
    at com.merant.sequelink.net.Session.connect(Unknown Source)
    at com.merant.sequelink.net.Session.attach(Unknown Source)
    at com.merant.sequelink.ssp.SspFactory.GetSsp(Unknown Source)
    at com.merant.sequelink.ctxt.conn.ConnectionContext.doConnect(Unknown Source)
    at com.merant.sequelink.ctxt.conn.ConnectionContext.connect(Unknown Source)
    at com.merant.sequelink.jdbc.SequeLinkConnection.<init>(Unknown Source)
    at com.merant.sequelink.jdbc.SequeLinkDriver.connect(Unknown Source)
    at macromedia.jdbc.MacromediaDriver.connect(Unknown Source)
    at jrun.sql.pool.JDBCPool.createPhysicalConnection(JDBCPool.java:599)
    at jrun.sql.pool.JDBCPool.create(JDBCPool.java:550)
    at jrun.sql.pool.JDBCPool.checkOut(JDBCPool.java:491)
...
```