Connection Pooling Deadlocks with User-based Security Models

Posted At : May 6, 2005 2:40 PM | Posted By : Steven Erat
Related Categories: Java, ColdFusion

This is an explanation of why someone with a ColdFusion application that uses User-based Security Models can get themselves into a connection pool deadlock, resulting in "Timeout trying to establish connection" errors even when the database sees no requests for connections coming in.

This follows up on **a previous blog entry** here that describes User-based Security Models with regard to connection pooling in general. If your're reading this and are confused, try reading the other blog entry first.

This blog entry regards thread dumps like **the one below**, so take a look at it before getting started.

User-based Security can be described as using unique username and password combinations passed through the CFQuery/CFStoredProc tag, often a requirement for databases configured for User-based Security. In this case the username/password combination often comes directly from end user logins on the application.

This regards the "Timed out trying to establish connection" errors with ColdFusion MX servers of version 6.1 U1.

I reviewed the case histories and the current stack traces recently provided from both of you, and we've arrived a hypothesis that fits well. Skip to the bottom for the conclusion, the rest is explanation. Reviewing the source code for JDBCPool we confirmed that the web threads in the thread dumps were checking the connection pool to obtain a connection once per second until the DSN Login Timeout limit is reached. Each thread in JDBCPool created a lock while it checked the pool and then released the lock at the end of the one second interval before creating/releasing the lock again. The thread dump caught the web threads all doing the same thing, checking the connection pool to see if a connection was free.

As mentioned earlier, the primary cause of "Timed out trying to establish connection" is that the JDBC driver simply didn't receive a reply from the database server to the TCP connection when first trying to create a connection to be used and then added to the pool. If the database server wasn't reachable at all then the error would be "Error establishing socket" instead.

We assume that the connection pool was full. Not being able to get a connection from the connection pool is a third cause of the "Timed out trying to establish connection" error. Since the thread was trying to get a connection from the pool and the pool was full with busy connections, the database administrators wouldn't see any new requests for connections coming in, which is exactly what they observed in earlier email threads. When you restarted the CF server, that killed the connection pool, and the DBA then saw a "stream of connections" from the server in question just after that because ColdFusion was handling requests for new connections and filling up the

connection pool again.

All the web threads in the first thread dump today were shown to be in a state where they were waiting to get a connection from the pool. Normally, this should imply that all the available connections in the pool were busy handling other queries from other web threads, but in fact there were no other web threads... just those trying to get a new connection. So if all threads are trying to get a connection and the existing connections are tied up in use with queries by web threads, where are those web threads that have active queries?

Under "normal" circumstances this would be a conundrum, but knowing that your applications have user-based query login parameters sent with each query, then it is conceivable that the application has gotten into an unexpected type of deadlock.

A ColdFusion template will hold onto a database connection until the page completes. If there are two queries having unique logins for the same datasource on the same page, the request will have two connections open by the time it reaches the end.

Recall from earlier conversations that a single datasource connection pool can be composed of connections having multiple logins. This will result in a unique username/password login behaving as if it had just a slice of the overall pool. If there are 30 user/password logins and a limit of 30 connections for the datasource, then its possible that each user may only get to use one of the 30 connections since a connection can only be reused if it has the same username/password

Template A

```
1   <cfquery
2       username="Fred"
3       password="pebbles"
4       datasource="Bedrock">
5       SELECT *
6       FROM tblNewCars
7   </cfquery>
8
9   <cfquery
10      username="Barney"
11      password="bambam"
12      datasource="Bedrock">
13      SELECT *
14      FROM tblSportsEquip
15  </cfquery>
16
```

Template B

```
1   <cfquery
2       username="Barney"
3       password="bambam"
4       datasource="Bedrock">
5       SELECT *
6       FROM tblQuarries
7   </cfquery>
8
9   <cfquery
10      username="Fred"
11      password="pebbles"
12      datasource="Bedrock">
13      SELECT *
14      FROM tblPetFood
15  </cfquery>
16
```

For example, say there are just 2 unique user/password logins, 2 cfm templates, a limit of 2 dsn connections and Simultaneous Request value of 2. Imagine template A has query for user Fred and then a query for user Barney, and template B has a query for user Barney followed by a query for user Fred, the reverse order. If both pages run at the same time, template A will get a connection for user Fred and template B will get a connection for user Barney. The dsn limit is full with 2 connections. Then template A moves on and now starts to process the second query with login credentials for user Barney, but the pool is at the limit and this query will have to wait until a free connection is available. At the same time, template B has moved on also and is now on its second query with login credentials for user Fred, but since the connection pool is full it must wait for a connection to be freed. In this case, template A's second query is waiting for template B's first query to complete, and template B is waiting for template A's second query to complete. This is a deadlock.

Each template in this case will wait for 30 seconds by default, then each one will error with "Timed out trying to establish connection".

In this deadlock situation, a thread dump would show two threads running, each one in JDBCPool trying to check out a connection, trying every second for 30 seconds, which is the default Login Timeout for a datasource. The thread dump would not show any threads actually in a query. This is how your thread dumps look.

In theory, this could be avoided by setting the dsn connection limit equal to ((Number of Simultaneous Requests) * (Number of unique login credentials per datasource per template) + (1 or 2)). So in this imaginary scenario above, this would be (2 * 2) + 1, or 5 as a value for the datasource connection limit. Having a pool of 5 would prevent the deadlock in that case.

Your situation is surely much more complicated, and estimating a good value for dsn connection pools will be important to resolve this. This hypothesis depends on the assumption that templates use at least two queries per page request with at least two user login credentials passed to the query. If you think this is likely, then this hypothesis may turn out to be accurate.

Representative Thread Dump

```
Stack 1:

Thread "jrpp-2355":
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationsometemplate.cfm:42

Thread "jrpp-2353":
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.setupCachedQuery(QueryTag.java:603)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:443)
at C:Inetpubwwwrootapplicationcustomtagssomeothertemplate.cfm:48

Thread "jrpp-2342":
```

```
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationincludessomeothertemplate.cfm:7

Thread "jrpp-2341":
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationsomeothertemplate.cfm:78

Thread "jrpp-2340":
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationsometemplate.cfm:42

Thread "jrpp-2325":
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationsometemplate.cfm:42

Thread "jrpp-2300":
```

```
Runing a CFQUERY
at java.lang.Object.wait(Native Method)
- waiting on (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.checkOut(JDBCPool.java:359)
- locked (a coldfusion.server.j2ee.sql.pool.JDBCPool)
at coldfusion.server.j2ee.sql.pool.JDBCPool.requestConnection(JDBCPool.java:755)
at coldfusion.server.j2ee.sql.pool.JDBCManager.requestConnection(JDBCManager.java:125)
at coldfusion.server.j2ee.sql.JRunDataSource.getConnection(JRunDataSource.java:138)
at coldfusion.sql.DataSrcImpl.getCachedConnection(DataSrcImpl.java:124)
at coldfusion.sql.DataSrcImpl.getConnection(DataSrcImpl.java:75)
at coldfusion.sql.SqlImpl.execute(SqlImpl.java:214)
at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:447)
at C:Inetpubwwwrootapplicationsometemplate.cfm:42
```