Connection time out issues after inactivity period

Asked 12 years, 4 months ago Modified 5 years, 7 months ago Viewed 19k times

5

We have an api which uses hibernate as ORM tool and we use c3p0 as the connection pool handler. We have no problems when we are under load. However, we are running out into "unable to obtain a connection" exceptions when the api has been inactive for a day or so. So, if no body uses the api over the weekend, we get connection errors on monday morning.

> Caused by: java.sql.SQLException: An attempt by a client to checkout a Connection has timed out.

We use mysql as the database. On my research, i got to know that mySQL makes connections stale after 8 hours or so. It might be possible that the connection pool is giving out a stale connection to the client and hence the connection timeout exceptions for the client.

At present, we do not have any connection testing configured in C3Po. Lets say, if I use IdleTestPeriod to test the connection before they are given to the client by the pool. Then what happens if all my connections fail the test at a point of time? Will those failed connections be removed from the pool and new active connections be generated again?

Currently, this is the c3p0 settings that we are using. Any other reasons possible for this problem?

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method="close">
        <property name="driverClass" value="${----}"/>
        <property name="jdbcUrl" value="${----}"/>
        <property name="user" value="${----}"/>
        <property name="password" value="${------}"/>
        <property name="minPoolSize" value="5"/>
        <property name="acquireIncrement" value="5" />
        <property name="maxPoolSize" value="125" />
        <property name="maxStatements" value="10" />
        <property name="maxIdleTime" value="180" />
        <property name="maxIdleTimeExcessConnections" value="30" />
        <property name="checkoutTimeout" value="3000" />
        <property name="preferredTestQuery" value="SELECT 1" />
    </bean>
```

java spring hibernate database-connection c3p0

edited Apr 28, 2019 at 9:45

Cœur
38.6k • 26 • 202 • 276

asked Jul 31, 2012 at 15:44

Npa
617 • 3 • 15 • 27

2 Answers

Sorted by: Highest score (default)

12

So you have a checkoutTimeout of 3 secs (3000 msecs) set. That's the Exception you're seeing. Clients are only permitted to wait for three seconds to checkout a Connection from the pool; if three seconds isn't enough, they see your Exception.

The question is, why are clients taking so long to get a Connection? Normally checking out a Connection is a pretty fast operation. But if all Connections are checked out, then clients have to wait for (slow) Connection acquisition from the database.

You have your pool configured to pretty aggressively cull Connections. Any number of Connections above minPoolSize=5 will be destroyed if they are idle for more than maxIdleTimeExcessConnections=30 seconds. Yet your pool is configured for large-scale bursts: maxPoolSize=125. Suppose that your app is quiet for a while, and then gets a burst of Connection requests from clients. The pool will quickly run out of Connections and start to acquire, in bursts of acquireIncrement=5. But if there are suddenly 25 clients and the pool has only 5 Connections, it's not improbable that the 25th client might time out before acquiring a Connection.

There's lots you can do. These tweaks are separable, you can mix or match as you see fit.

1. Cull idle "excess" Connections less aggressively, so that in general, your pool has some capacity to service bursts of requests. You might drop maxIdleTimeExcessConnections entirely, and let Connections slowly wither after maxIdleTime=180 seconds of disuse. (Downside? A larger resource footprint for longer during periods of inactivity.)
2. Set minPoolSize to a higher value, so that it's unlikely that the pool will see a burst of activity for which it has way too few Connections. (Downside? Larger permanent resource footprint.)
3. Drop checkoutTimeout from your config. c3p0's default is to allow clients to wait indefinitely for a Connection. (Downside? Maybe you prefer clients to quickly report a failure rather than wait for possible success.)

I don't think that the problem that you are observing has much to do with Connection testing or MySQL timeouts per se, but that doesn't mean you should not deal with those issues. I'll defer to nobeh's advice on the MySQL reconnect issue. (I'm not a big

MySQL user.) You should consider implementing Connection testing. You have a preferredTestQuery, so tests should be reasonably fast. My usual choice is to use testConnectionOnCheckin and idleConnectionTestPeriod. See http://www.mchange.com/projects/c3p0/#configuring_connection_testing

Good luck!

Share
Improve this answer
Follow

edited Mar 6, 2019 at 14:52

approxiblue
7,122 ● 16 ● 52 ● 59

answered Jul 31, 2012 at 18:04

Steve Waldman
14.1k ● 1 ● 37 ● 47

1

In the section of **High availability and clustering** in MySQL Java Connector, take a look at the properties; specifically `autoReconnect` and `autoReconnetForPools`. Use the properties in your JDBC connection URL. They have helped me before when using MySQL, Hibernate, and C3P0. Hope that this helps.

Share Improve this answer Follow

answered Jul 31, 2012 at 16:22

nobeh
10k ● 10 ● 52 ● 68

Note. Use of the autoReconnect option is not recommended because there is no safe method of reconnecting to the MySQL server without risking some corruption of the connection state or database state information. Instead, you should use a connection pool which will enable your application to connect to the MySQL server using an available connection from the pool. The autoReconnect facility is deprecated, and may be removed in a future release.
– wiggin200 Jun 16, 2020 at 12:59