Application connects to wrong logical server after failover

Last updated by | Subbu Kandhaswamy | Jun 11, 2021 at 7:44 PM PDT

Contents

Issue

Application connecting to wrong Logical Server. Meaning, after performing a failover the application/client attempts to connect to old primary. This occurs 20-30 minutes after the failover was completed. There are currently two problems with Failover Group connection routing that customers frequently encounter during manual failover operations. The manifestation of both problems is that applications continue to connect to the old geo-primary or the old geo-secondary following a failover operation.

One symptom might be for the application to receive "database is read-only errors" when it tries to write to the new geo-primary database, as it is still using a connection that is connected to the old geo-primary database (which is now a secondary).

Scenario1

Failover Group connection routing is implemented using DNS. During failover, the failover group endpoints are updated to point to the appropriate new primary and new secondary servers by changing the target of the appropriate DNS entry. Today, these DNS entries are created with a TTL of 30 seconds.

This means that DNS clients cache these DNS entries for 30 seconds. As a result, updates to the DNS records do not propagate immediately; entries will be stale until all clients and intermediate nodes have refreshed their caches. So, it can take anywhere from 0 to approximately 1 minute (?) (depending on network topology) for logins to failover group endpoints to be routed to their new targets following a failover.

Workaround1

Flushing DNS caches may or may not help the problem since intermediate network nodes that service DNS requests may also cache the DNS results.

The recommended workaround for this issue is simply to wait until the DNS entries are refreshed on the client. In an upcoming deployment, the TTL on these DNS entries will be reduced to under a minute -- this will significantly reduce the impact window for this problem.

Scenario2

Some SQL client libraries use a feature called "connection pooling" which re-uses connections to the same data source rather than closing and re-opening them whenever a new database connection is needed. In particular, connection pooling is enabled in ado.net by default see here

Combining the problem described in Scenario1, connection pooling can cause newly opened connections to re-use a connection to the old database, thus preventing the application from connecting to the new primary database indefinitely. The sequence of events resulting in the problem is as follows: a. A Failover Group failover is executed, swapping the roles of all databases in the Failover Group, severing all existing connections to databases in the failover group, and swapping the DNS entries for the Failover Group endpoints to point to the new primary and secondary servers respectively. b. The client application opens a new connection to the primary failover group endpoint. Since the DNS has a 5 minute TTL, it has not yet been updated on the client machine. As a result, the new connection is incorrectly routed to the old primary server. c. Every time a new connection is opened to the same endpoint, it re-uses the connection that was established in b and so does not go to the intended server. No new DNS lookups are performed, so the client does not notice that the Failover Group endpoint target has changed.

Workarounds

There are three potential workarounds to this problem: a. Restart the application approximately 10 minutes after every failover. This will have the effect of clearing the ADO.NET connection pool. b. In the connection string, specify Pooling=False to disable connection pooling. c. Call SQLConnection.ClearAllPools or SQLConnection.ClearPool(conn) in the client application whenever a "read-only" error is encountered.

Classification

Root Cause: Azure SQL DB v2\GeoDR/FailoverGroup

How good have you found this content?