Duplicate connection requests to SQL without waiting for an ACK for previous connection - XDBHost

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:23 AM PDT

Contents

Connections to XDBHost

XDBHost aka SocketDuplicator, receives incoming connection requests and sends it to SQL. XDBHost maintains a queue per SQL instance to hold incoming login requests and processes logins serially from this queue. For every login, XDBHost writes to SQL's named pipe and waits for SQL to ACK the login. Once the ACK is received, XDBHost sends the next login. If there are no bottlenecks in SQL, there will not be any issues.

Bug

We've observed that the dispatcher to which the login task is assigned is busy and has multiple workers in its runnable list for some cases. The ACK can be sent back to sqlserver only after the login worker gets access to the scheduler and processes the login workflow. This can lead to a delay in sending the ACK back to XDBHost and as a result new login requests start convoying in XDBHost's queue. If the login requests are found waiting in the queue for more than 10 seconds, they are failed and the socket dup queue is cleared.

RepairItem to fix the above behavior- Modifying the socket dup workflow on XDBHost

Post Fix - expected new workflow

1. XDBHost receives a new login.
2. XDBHost processes the login to create a new socket duplication task.
3. If there are no tasks being processed currently, this task gets processed. If there are any tasks being processed, this task is enqueued.
4. The current task initializes the datastructures used to send the login information to SQL.
5. The task then posts an Async read if not already posted.
6. Next it writes the login data to the named pipe. When writing to the named pipe the return result can be either an ERROR_SUCCESS or ERROR_IO_PENDING. • If the return value is ERROR_SUCCESS, the current task is added to the m_ConnectionContextHashTable. This is a hashtable of connections that have been written to SQL and are waiting to be ACKed. After being added to the hashmap, a new task is picked up to be processed. • If the return value is ERROR_IO_PENDING, the current task is paused and the WriteDoneHandler continues the task.

7. When an ACK is received from the SQL instance, the ReadDoneHandler is invoked. The hashmap m_ConnectionContextHashTable is searched for the connection corresponding to the ACK and the process_login_finish is raised. The socket duplication task is completed.

Since the reads from the named pipe (ACKs) arrive asynchronously and out of order, new reads are posted either

1. When the first task is processed
2. On completion of a read from the named pipe.

For this reason, when destroying the socket dup instance, we need to check for both an in-progress task and m_fIsReadOutStanding.

How good have you found this content?