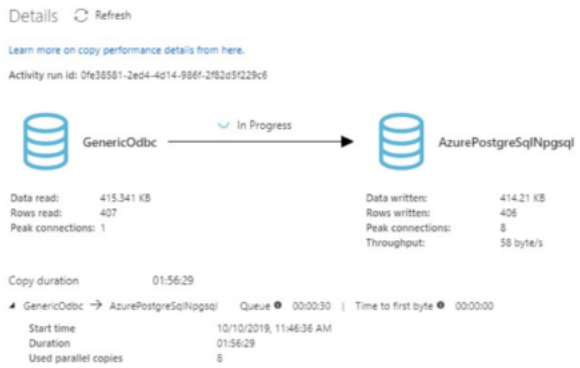
[NpgsqlConnector]Keep Alive caused self-IR hang

Last updated by | Jackie Huang | Jan 4, 2022 at 12:24 AM PST

201084- [NpgsqlConnector]Keep Alive caused self-IR hang

Monday, November 11, 2019
3:59 PM

SME	
Symptoms	Customer attached screenshot of a stalled transfer from the latest run (which should be in the log file). It shows 407 rows read and 406 written. When they checked on the sink side with select count(*) from vendor_dsos.muni_asset_staging where id_as_abs_bigint % 97 = 1 it returns 406 so it hasn't managed to write the last row (yet) . Also, running this to see when the first and last rows were written: select min(lower(transaction_time)), max(lower(transaction_time)), now() from vendor_dsos.muni_asset_staging where id_as_abs_bigint % 97 = 1 returns: min max now ---------- ---------- ---------- 10/10/2019 11:47:07 AM 10/10/2019 11:51:33 AM 10/10/2019 1:48:29 PM so it is nearly two hours since the last row was written. 
Resolution	checking the dump and found below one which may be the suspected issue. 000000ad`b79fdea8 00007ffa`cee5cc7e ntdll!ZwWaitForMultipleObjects+0x14 [minkernel\ntdll\daytona\objfre\amd64\usrstubs.asm @ 907] 01 000000ad`b79fdeb0 00007ffa`b57a3a2e KERNELBASE!WaitForMultipleObjectsEx+0xfe [minkernel\kernelbase\synch.c @ 1551] 02 000000ad`b79fe1b0 00007ffa`b57a3884 clr!WaitForMultipleObjectsEx_SO_TOLERANT+0x62 [f:\dd\ndp\clr\src\vm\threads.cpp @ 4292] 03 (Inline Function) --------`-------- clr!Thread::DoAppropriateAptStateWait+0x44 [f:\dd\ndp\clr\src\vm\threads.cpp @ 4326] 04 000000ad`b79fe210 00007ffa`b57a367d clr!Thread::DoAppropriateWaitWorker+0x1e4 [f:\dd\ndp\clr\src\vm\threads.cpp @ 4466] 05 000000ad`b79fe310 00007ffa`b59015aa clr!Thread::DoAppropriateWait+0x7d [f:\dd\ndp\clr\src\vm\threads.cpp @ 4133] 06 000000ad`b79fe390 00007ffa`b580207f clr!CLREventBase::WaitEx+0xc4 [f:\dd\ndp\clr\src\vm\synch.cpp @ 753] 07 (Inline Function) --------`-------- clr!CLREventBase::Wait+0x1f [f:\dd\ndp\clr\src\vm\synch.cpp @ 673] 08 (Inline Function) --------`-------- clr!Thread::Wait+0x1f [f:\dd\ndp\clr\src\vm\threads.cpp @ 4951] 09 000000ad`b79fe420 00007ffa`b580204c clr!Thread::Block+0x27 [f:\dd\ndp\clr\src\vm\threads.cpp @ 4908] 0a 000000ad`b79fe450 00007ffa`b5801df1 clr!SyncBlock::Wait+0x19d [f:\dd\ndp\clr\src\vm\syncblk.cpp @ 3561] 0b (Inline Function) --------`-------- clr!ObjHeader::Wait+0x29 [f:\dd\ndp\clr\src\vm\syncblk.cpp @ 2743] 0c (Inline Function) --------`-------- clr!Object::Wait+0x29 [f:\dd\ndp\clr\src\vm\object.h @ 541] 0d 000000ad`b79fe590 00007ffa`b2f1bda8 clr!ObjectNative::WaitTimeout+0xe1 [f:\dd\ndp\clr\src\classlibnative\bcltype\objectnative.cpp @ 315] 0e 000000ad`b79fe710 00007ffa`b2f1bb69 mscorlib_ni!System.Threading.SemaphoreSlim.WaitUntilCountOrTimeout(Int32, UInt32, System.Threading.CancellationToken)+0x68 [f:\dd\ndp\clr\src\BCL\system\threading\SemaphoreSlim.cs @ 469] 0f 000000ad`b79fe760 00007ffa`576eeffd mscorlib_ni!System.Threading.SemaphoreSlim.Wait(Int32, System.Threading.CancellationToken)+0x149 [f:\dd\ndp\clr\src\BCL\system\threading\SemaphoreSlim.cs @ 383] 10 000000ad`b79fe7f0 00007ffa`5772be1b Npgsql!Npgsql.NpgsqlConnector.StartUserAction(Npgsql.ConnectorState)+0x4d 11 000000ad`b79fe840 00007ffa`5772b2a4 Npgsql!Npgsql.NpgsqlCommand.Prepare()+0x11b https://github.com/npgsql/npgsql/blob/v3.1.10/src/Npgsql/NpgsqlConnector.cs Review the code, you can see it need to get SemaphoreSlim lock which is _keepAliveLock.

```
internal IDisposable StartUserAction(ConnectorState newState=ConnectorState.Executing)
{
    Contract.Ensures(State == newState);
    Contract.Ensures(IsInUserAction);

    if (!_userLock.Wait(0))
        throw new InvalidOperationException("An operation is already in progress.");

    // We now have the user lock, no user operation may be in progress but a keepalive might be

    if (IsKeepAliveEnabled)
    {
        // If keepalive happens to be in progress wait until it's done
        _keepAliveLock.Wait();

        // Disable keepalive, it will be restarted at the end of the user action
        if (KeepAlive > 0)
            _keepAliveTimer.Change(Timeout.Infinite, Timeout.Infinite);
    }
```

GitHub, Inc. [US] | https://github.com/npgsql/npgsql/blob/v3.1.10/src/Npgsql/NpgsqlConnector.cs

1 of 11 Options

```
}

/// <summary>
/// Contains the current value of the socket's ReceiveTimeout, used to determine whether
/// we need to change it when commands are received.
/// </summary>
int _currentTimeout;

/// <summary>
/// A lock that's taken while a user action is in progress, e.g. a command being executed.
/// </summary>
readonly SemaphoreSlim _userLock;

/// <summary>
/// A lock that's taken while a connection keepalive is in progress. Used to make sure
/// keepalives and user actions don't interfere with one another.
/// </summary>
SemaphoreSlim _keepAliveLock;

readonly object _keepAliveDisposeLock = new object();
```

Suggest to disable keep alive and so far working good.

https://icm.ad.msft.net/imp/v3/incidents/details/152165142/home

Tags	NpgsqlConnector

Created with Microsoft OneNote 2016.

How good have you found this content?