Snapshot Agent timeout exception

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:31 AM PST

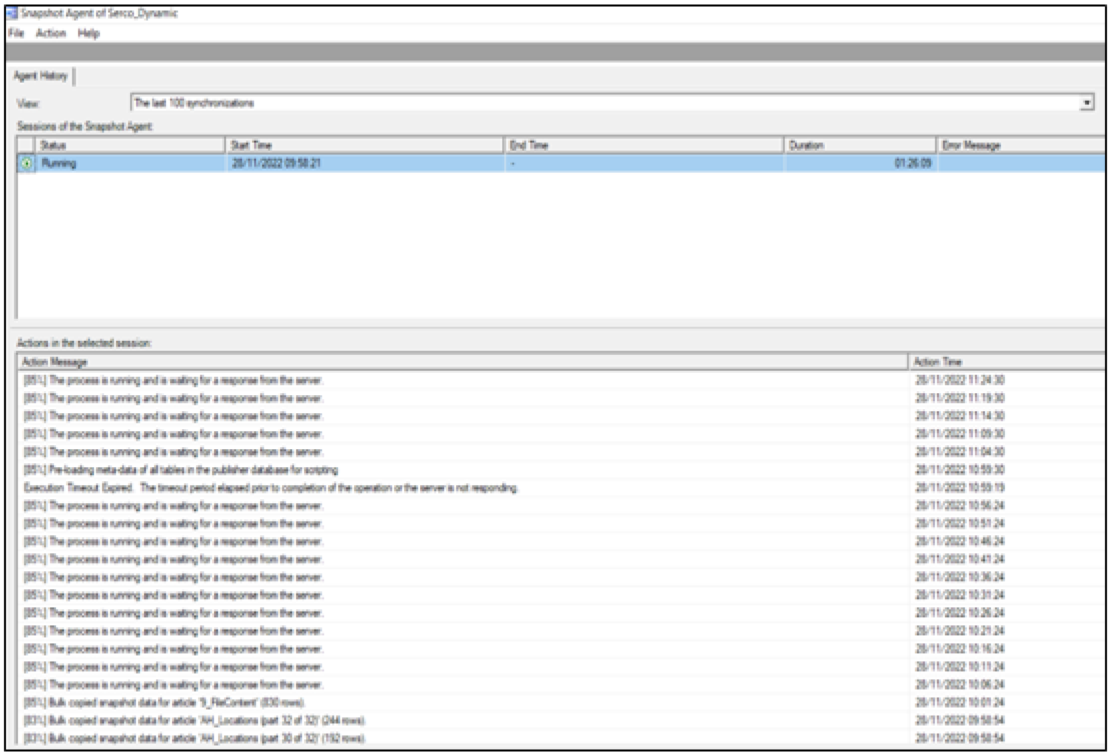
Contents

Issue

The customer is trying to configure transaction replication with the Publisher hosted on a Managed Instance. The configuration itself has succeeded, but when they run the Snapshot Agent, it is consistently timing out.

When checking the Snapshot Agent details and history in Replication Monitor, the following details are returned:

> Exception Message: Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding.
> **Pre-loading meta-data** of all tables in the publisher database for scripting



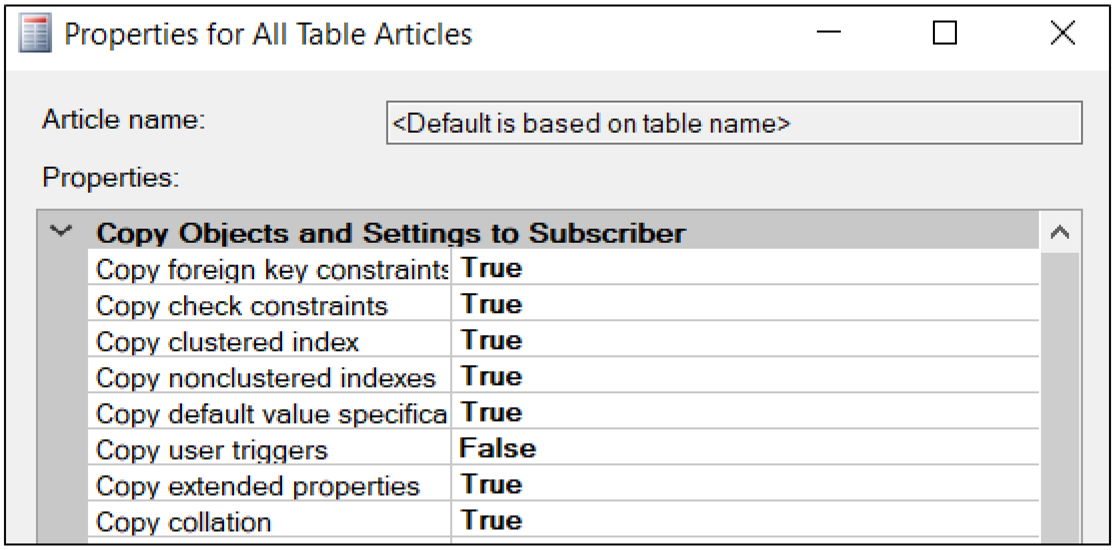
View: The last 100 synchronizations

Sessions of the Snapshot Agent:

Status	Start Time	End Time	Duration	Error Message
Running	28/11/2022 09:58:21	-	01:26:09	

Actions in the selected session:

Action Message	Action Time
[85%] The process is running and is waiting for a response from the server.	28/11/2022 11:24:30
[85%] The process is running and is waiting for a response from the server.	28/11/2022 11:19:30
[85%] The process is running and is waiting for a response from the server.	28/11/2022 11:14:30
[85%] The process is running and is waiting for a response from the server.	28/11/2022 11:09:30
[85%] The process is running and is waiting for a response from the server.	28/11/2022 11:04:30
[85%] Pre-loading meta-data of all tables in the publisher database for scripting	28/11/2022 10:59:30
Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding.	28/11/2022 10:59:19
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:56:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:51:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:46:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:41:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:36:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:31:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:26:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:21:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:16:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:11:24
[85%] The process is running and is waiting for a response from the server.	28/11/2022 10:06:24
[85%] Bulk copied snapshot data for article 'S_FileContent' (830 rows).	28/11/2022 10:01:24
[83%] Bulk copied snapshot data for article 'AH_Locations (part 32 of 32)' (244 rows).	28/11/2022 09:58:54
[83%] Bulk copied snapshot data for article 'AH_Locations (part 30 of 32)' (192 rows).	28/11/2022 09:58:54



Properties for All Table Articles

Article name: <Default is based on table name>

Properties:

Copy Objects and Settings to Subscriber	
Copy foreign key constraints	True
Copy check constraints	True
Copy clustered index	True
Copy nonclustered indexes	True
Copy default value specifica	True
Copy user triggers	False
Copy extended properties	True
Copy collation	True

The CPU consumption is normal at about 15%, with no other apparent bottlenecks on the Managed Instance. Outside of the snapshot issue, the instance and databases are performing as expected with unrelated queries succeeding.

The Publisher database contains about 60000 tables and 1.2 million Extended Properties. The exact same database and scenario on-premise completes instantly without any delay.

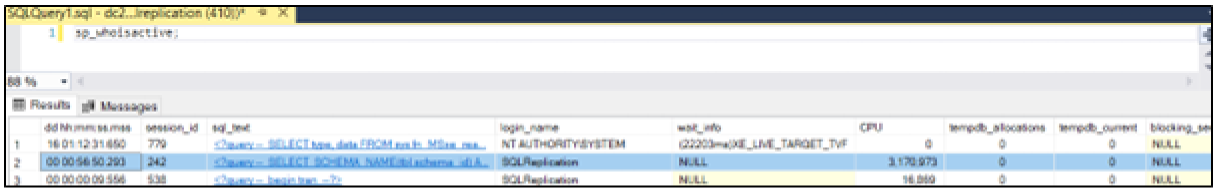
Investigation / Analysis

If possible for the customer, take the opportunity to capture the code that the long running snapshot agent job is running on the backend.

You can do this through the following DMV query:

```
-- currently executing queries in the database
SELECT t.text, qp.query_plan, req.query_hash,
       req.session_id, req.request_id, req.database_id, req.start_time, req.status, req.command,
       req.blocking_session_id, req.wait_type, req.wait_time, req.last_wait_type,
       req.cpu_time, req.total_elapsed_time, req.logical_reads
FROM sys.dm_exec_requests AS req
CROSS APPLY sys.dm_exec_sql_text(req.sql_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(req.plan_handle) AS qp
```

Some customers prefer to run the public 3rd-party sp_whoisactive stored procedure instead (GNU GPLv3 license). It returns a similar output, like this:



SQLQuery1.sql - dc2...replication (410))*

```
sp_whoisactive;
```

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	CPU	tempdb_allocations	tempdb_current	blocking_se
1	16 01:12:31.650	779	<?query -- SELECT type, data FROM sys.fn_MSxe_rea...	NT AUTHORITY\SYSTEM	(22203ms)XE_LIVE_TARGET_TVF	0	0	0	NULL
2	00 00:56:50.293	242	<?query -- SELECT SCHEMA_NAME(tbl.schema_id) A...	SQLReplication	NULL	3,170,973	0	0	NULL
3	00 00:00:09.556	538	<?query -- begin tran --?>	SQLReplication	NULL	16,069	0	0	NULL

This showed that the Snapshot Agent was executing the following query in the published database:

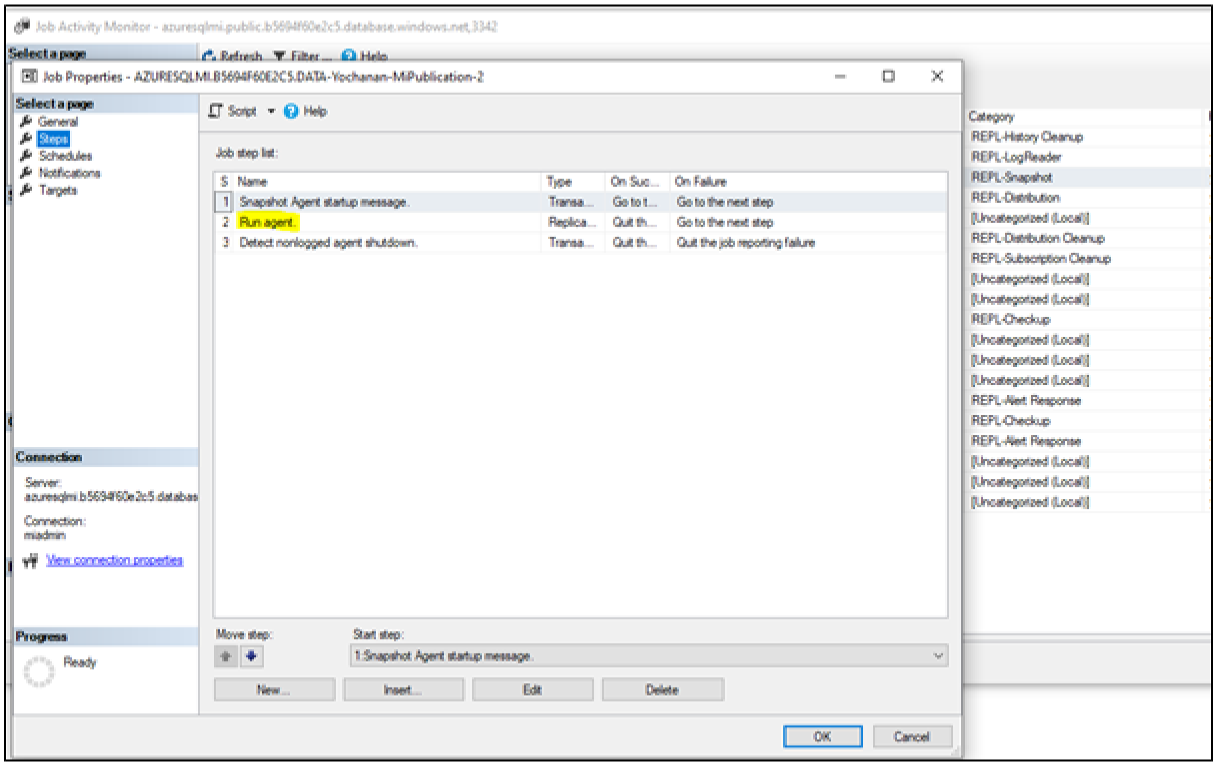
```
-- (formatted for readability)
SELECT
    SCHEMA_NAME(tbl.schema_id) AS [Table_Schema],
    tbl.name AS [Table_Name],
    i.name AS [Index_Name],
    p.name AS [Name],
    p.value AS [Value]
FROM
    sys.tables AS tbl
    INNER JOIN sys.indexes AS i ON (i.index_id > N'0' and i.is_hypothetical = N'0') AND (i.object_id=tbl.objec
    LEFT OUTER JOIN sys.key_constraints AS k ON k.parent_object_id = i.object_id AND k.unique_index_id = i.ind
INNER JOIN sys.extended_properties AS p
    ON p.major_id=CASE (i.is_primary_key + 2*i.is_unique_constraint) WHEN 0 THEN i.object_id ELSE k.object_id
    AND p.minor_id=CASE (i.is_primary_key + 2*i.is_unique_constraint) WHEN 0 THEN CAST(i.index_id AS int) ELSE
    AND p.class=CASE (i.is_primary_key + 2*i.is_unique_constraint) WHEN 0 THEN 7 ELSE 1 END
ORDER BY
    [Table_Schema] ASC,[Table_Name] ASC,[Index_Name] ASC,[Name] ASC
```

To confirm this, ask the customer to execute the statement manually against the published database; it very likely will show the same symptom and will "never" complete unless you wait for several hours.

This issue happens due to a defect in Azure SQL code. The object prefetch step fetches and caches certain table properties in order to optimize the snapshot process. However, on user tables with a lot of extended properties, this prefetch actually results in worse performance leading to time-outs. The Snapshot Agent might get stuck and eventually time-out with the message "Prefetch object failed for Database xxxx".

Mitigation

To fix the problem, add the "-PrefetchTables 0" parameter to the code block inside the running snapshot job as follows.



Server:
azuresqlmi.b5694f60e2c5.databas

Connection:
miadmin

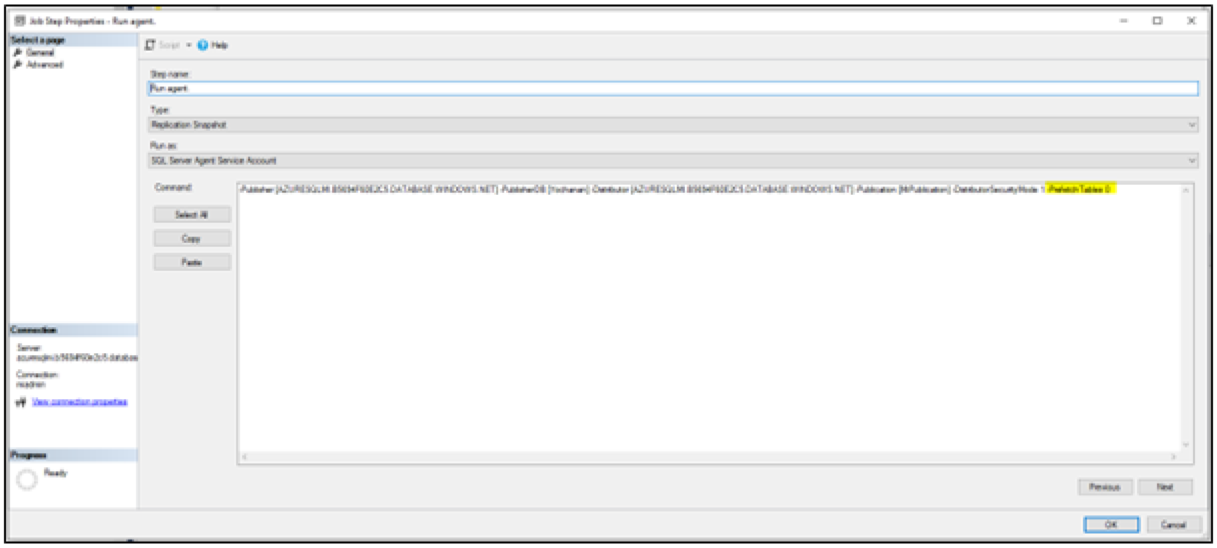
View connection properties

Job step list:

S	Name	Type	On Suc...	On Failure
1	Snapshot Agent startup message.	Transa...	Go to t...	Go to the next step
2	Run agent.	Replica...	Quit th...	Go to the next step
3	Detect nonlogged agent shutdown.	Transa...	Quit th...	Quit the job reporting failure

Start step:
1:Snapshot Agent startup message.

Category
REPL-History Cleanup
REPL-LogReader
REPL-Snapshot
REPL-Distribution
[Uncategorized (Local)]
REPL-Distribution Cleanup
REPL-Subscription Cleanup
[Uncategorized (Local)]
[Uncategorized (Local)]
REPL-Checkup
[Uncategorized (Local)]
[Uncategorized (Local)]
[Uncategorized (Local)]
REPL-Alert Response
REPL-Checkup
REPL-Alert Response
[Uncategorized (Local)]
[Uncategorized (Local)]
[Uncategorized (Local)]



Step name: Run agent

Type: Replication Snapshot

Run as: SQL Server Agent Service Account

Command: -Publisher [AZURESQLMI 85694F60E2C5.DATABASE.WINDOWS.NET] -PublisherDB [Yochanan] -Distributor [AZURESQLMI 85694F60E2C5.DATABASE.WINDOWS.NET] -Publication [MyPublication] -DistributorSecurityMode 1 -PrefetchTables 0

Edit the "Run agent" step and add the `-PrefetchTables 0` parameter to the end of the parameter command line. PrefetchTables skips the object prefetch step in the scripting operation. Without the object prefetch step, snapshot avoids its optimization and executes successfully as expected.

More Information

When we examined the long snapshot process, we realized that the part that took a long time was the INNER JOIN operation with sys.extended_properties. When we changed this operation to INNER MERGE JOIN, the query was completed very quickly. But unfortunately we don't have the chance to interfere with this during the replication snapshot process.

```
SELECT
SCHEMA_NAME(tbl.schema_id) AS [Table_Schema],
tbl.name AS [Table_Name],
i.name AS [Index_Name],
p.name AS [Name],
p.value AS [Value]
FROM
sys.tables AS tbl
INNER JOIN sys.indexes AS i ON (i.index_id > N'0' and i.is_hypothetical = N'0') AND (i.object_id=tbl.object_id
LEFT OUTER JOIN sys.key_constraints AS k ON k.parent_object_id = i.object_id AND k.unique_index_id = i.index_i
INNER MERGE JOIN sys.extended_properties AS p ON p.major_id=CASE (i.is_primary_key + 2*i.is_unique_constraint)
ORDER BY
[Table_Schema] ASC,[Table_Name] ASC,[Index_Name] ASC,[Name] ASC
```

The product group has identified a possible fix for this issue, which hasn't been deployed yet. See the reference to the work item in the internal section below.

To check column level extended properties, check:

```
SELECT
   SCHEMA_NAME(tbl.schema_id) AS SchemaName,
   tbl.name AS TableName,
   clmns.name AS ColumnName,
   p.name AS ExtendedPropertyName,
   CAST(p.value AS sql_variant) AS ExtendedPropertyValue
FROM
   sys.tables AS tbl
   INNER JOIN sys.all_columns AS clmns ON clmns.object_id=tbl.object_id
   INNER JOIN sys.extended_properties AS p ON p.major_id=tbl.object_id AND p.minor_id=clmns.column_id AND p.cl
```

Check Working with Sql serve extended properties.

Public Doc Reference

- Transactional Replication
- Snapshot Replication

Internal Reference

ServiceRequest: 2211280050001372

- Incident-352484272 Details - IcM (microsofticm.com)
- Incident-314304480 Details - IcM (microsofticm.com)
- Work Item 1909093

Root Cause Classification

/Root Cause: Azure SQL v3/Replication/Transactional Replication/Service Bug/Known Issue

How good have you found this content?