

Performance: Applying Snapshots to the Subscriber

Last updated by | Holger Linke | Dec 21, 2022 at 4:38 AM PST

Contents

- [Issue](#)
- [Investigation / Analysis](#)
 - [Check if the Managed Instances are General Purpose or Bu...](#)
 - [Check how many threads are used to apply the snapshot](#)
- [Mitigation](#)
 - [Mitigation 1: Scale to the highest-possible SLO](#)
 - [Mitigation 2: Configure parallel snapshot threads](#)
 - [Set the publication's sync method to "native"](#)
 - [Concern 1 - has to reinitialize all subscriptions](#)
 - [Concern 2 - Snapshot generation might block published ta...](#)
 - [Configure the –MaxBCPThreads parameter of the Snapsho...](#)
 - [Mitigation 3: Initialize subscription from a backup](#)
 - [Mitigation 4: Consider Compressed snapshots](#)
- [External Pub Reference](#)

Issue

Transactional Replication with the same Managed Instance hosting the Publisher and Distributor role. The Publisher database is very large, and some of the tables contain several million rows. When the customer wants to (re-)initialize the subscriptions, the following symptoms are noticed:


- It takes a very long time to create the snapshot
- It takes a very long time to apply the snapshot to the Subscriber.

In one reported example, applying a 200 GB snapshot did not finish within 36 hours, even after scaling the Managed Instance to the largest Business Critical SLO.

Investigation / Analysis

The issue in itself can have several overlapping causes. The most likely one are that the Managed Instance I/O is undersized and throttled, and/or that the data is extracted and inserted in a single BCP thread. For large tables, the single-threaded BCP operation is the main bottleneck.

Check if the Managed Instances are General Purpose or Business Critical

Check the SLO of the Managed Instances that are part of the replication topology , and run ASC Troubleshooter against them to identify any I/O bottlenecks. If this is a General Purpose instance with just a few vCores, the instance might run into I/O latencies as described in [File IO characteristics in General Purpose tier](#) . Business Critical instances are using local SSDs instead of the GP's Azure Storage, so latency and throughput is much better on BC instances.

Check how many threads are used to apply the snapshot

(1) Check the publication's synchronization method. This can be either of 0 = "native", 1 = "character-based", 2 = "concurrent", or 3 = "concurrent character-based".

You won't see this on SSMS, so you have to:

- either script out the publication and check the `sp_addpublication ... @sync_method` parameter,
- or execute `exec sp_helppublication '<publication name>'` on the Publisher database.
The resultset of `sp_helppublication` has a column "synchronization method" which will show you the configured value.

If the sync method is set to one of the "concurrent" options, then the snapshot can only use a single thread: both for creating the snapshot and for applying it to the Subscriber.

(2) Check the snapshot folder on Azure Storage. Go into the folder of the most recent snapshot and look for the files named like "articlename_articleID.bcp". If there is only one BCP file per article, then the snapshot can only be applied single-threaded.

(3) Check the history of the Snapshot and Distribution Agents when they are creating or applying the snapshot. If you see plain messages like the following, without any indication that this is part x of y:

```
[0%] Bulk copying snapshot data for article 'articlename'
[0%] Bulk copied snapshot data for article 'articlename' (xxxxxxx rows).
```

... then you can be sure that the BCP file is created or applied single-threaded.

Mitigation

Mitigation 1: Scale to the highest-possible SLO

If budget and available storage sizes allow, scale to a high-end Business Critical instance to give the best I/O performance.

Mitigation 2: Configure parallel snapshot threads

This has two prerequisites as mentioned above: set `sync_method` to `native` and add the `-MaxBCPThreads` parameter:

Set the publication's sync method to "native"

This is a prerequisite of running the BCP operations with multiple threads. If you are using one of the "concurrent" sync methods, then one thread is used, regardless of how many threads you are configuring for BCP.

To change the sync method, you can run the following SQL statement in the Publisher database:

```
exec sp_changepublication @publication = 'Publication_Tran',
    @property = 'sync_method',
    @value = 'native',
    @force_invalidate_snapshot = 1, -- need to create a new snapshot
    @force_reinit_subscription = 1 -- need to reinitialize all subscriptions
GO
```

Concern 1 - has to reinitialize all subscriptions

If you change this option, you have to set `@force_reinit_subscription = 1`, meaning that all existing subscriptions are re-initialized. The customer may object to this, especially if they have many subscriptions and only one or few need new snapshots.

In this case you will have to explore other options first, including a re-design of the publications (e.g. moving the largest tables into separate publications).

Concern 2 - Snapshot generation might block published tables

When you use the "native" sync method (not "concurrent"), SQL Server places shared locks for the duration of the snapshot generation on all published tables. This prevents updates from being made on the published tables while the Snapshot Agent is running. The customer therefore might object to changing to "native", as they might have explicitly chosen "concurrent" to avoid blocking.

To address this concern, you might consider using snapshot isolation on the Publisher database to avoid blocking by the snapshot.

The snapshot isolation mode can be configured like this:

```
USE [master]
ALTER DATABASE your_publisher_db SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE your_publisher_db SET READ_COMMITTED_SNAPSHOT ON WITH NO_WAIT
```

(Note: this needs further testing, the author of this article hasn't tried this yet)

Configure the –MaxBCPThreads parameter of the Snapshot Agent and Distribution Agent

This parameter specifies the number of bulk copy operations that can be performed in parallel when the snapshot is created and applied. The performance benefit from using `-MaxBCPThreads` depends on the number of processors of the instance running the Snapshot and Distribution Agents. Specifying a high number for `-MaxBCPThreads` can overburden the system, because the system may spend too much time managing threads. The `MaxBCPThreads` value should not exceed the number of vCores of the Managed Instance.

`MaxBcpThreads` must have a value greater than 0 and has no hard-coded upper limit. Its default is 2 times the number of processors, up to a maximum value of 8.

When applying a snapshot that was generated for a publication using the "concurrent" snapshot option, only one thread is used, regardless of the number you specify for `MaxBcpThreads`.

The easiest way to configure `–MaxBCPThreads` is:

- In SSMS, connect to the Publisher/Distributor instance, go to SQL Server Agent, and open Job Activity Monitor
- Identify the Snapshot Agent Job and the Distribution Agent job related to the publication and subscription
- Open either job, go to the Run agent. step, and add `-MaxBCPThreads 8` at the end of the command line
- Run the Snapshot Agent job and confirm that the BCP operation is executed multi-threaded (several BCP files per article, messages in MSsnapshot_history)
- Run the Distribution Agent job and confirm the multi-threaded BCP operation; note that if the snapshot is created single-threaded, then the Distribution Agent also runs single-threaded.

Mitigation 3: Initialize subscription from a backup

Performance issues with creating and applying snapshots can be avoided by initializing the Subscriber database with a backup of the Publisher database. Initializing with a backup might be the fastest way to deliver data to the Subscriber and is convenient, because any recent backup can be used if it was taken after the publication was enabled for initialization with a backup.

The trade-off between these options is the time it takes to create the snapshot and applying it to the Subscriber, vs. the time it takes to restore the database. The snapshot might still be the better option, especially if not all tables in the database are published and the unpublished data is much larger than the published data.

Various scenarios might apply:

- [Initialize Subscription from a Backup \(avoid using Snapshot\)](#).
- [Initialize Subscription from a Backup \(on-premise to MI\)](#) - *detailed sample script*
- [Initialize Subscription from a Backup \(MI to MI PITR\)](#).
- [Initialize Subscription from a Backup \(MI COPY ONLY backup\)](#).

Mitigation 4: Consider Compressed snapshots

Using Compressed Snapshots is available for Managed Instances, but might not have the same benefit as for on-premise/IaaS SQL Server. Compressed snapshots can, in some cases, improve the performance of transferring snapshot files across the network. However, compressing the snapshot requires additional processing by the Snapshot Agent when generating the snapshot files, and by the Distribution Agent when applying the snapshot files. This may slow down snapshot generation and increase the time it takes to apply a snapshot in some cases. Additionally, compressed snapshots cannot be resumed if a network failure occurs. Consider these tradeoffs carefully when using compressed snapshots across a network.

For further details, see article [Compressed snapshots](#) .

External Pub Reference

[Enhance General Replication Performance - Snapshot Considerations](#) 

[Transactional Replication Performance Tuning and Optimization](#) 

How good have you found this content?

