Distribution cleanup deleting fewer rows than expected

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:37 AM PDT

Contents

Issue

The customer is running Transactional Replication on their Managed Instance, which is hosting both Publisher and Distributor roles. The `Distribution clean up: Distribution` job is running on the default schedule of every 10 minutes, but when monitoring the cleanup job history, apparently no rows are removed, like this:

```
Executed as user: DB4C1\WF-oM3QjL6VozKSGhF.
Deleted 0 row(s) per millisecond from MSrepl_commands [SQLSTATE 01000] (Message 22121)
Deleted 0 row(s) per millisecond from MSrepl_transactions [SQLSTATE 01000] (Message 22121)
Removed 0 replicated transactions consisting of 0 statements in 11000 milliseconds (0 rows/millisec). [SQLSTAT
The step succeeded.
```

This continues over several days, and because there are constant updates to the published tables, at least some metadata rows should have been removed.

Investigation / Analysis

Check the cleanup job history and its execution statistics

Run the following query at the Distributor to return the most recent execution results for the cleanup job:

```
select top 50 h.run_date, h.run_time, h.run_duration, h.run_status, h.retries_attempted, h.message
from msdb..sysjobs j
inner join msdb..sysjobhistory h on j.job_id = h.job_id
where (j.category_id = 11 or j.name like 'Distribution clean up%')
and h.step_id = 1
order by instance_id desc
```

The `message` column contains the execution results, of which you see an example in the "Issue" section above.

Understand the factors of the cleanup algorithm

The algorithm for calculating the cutoff for the cleanup is rather complicated. It is considering several factors to make sure that it keeps all the data that is still required for servicing the active subscriptions, and that it only deletes data that is no longer needed. The result is a Log Sequence Number (LSN, xact_seqno) for each Publisher database which is used as a cutoff point, and everything older than this cutoff LSN will be deleted.

The most important dependencies are the following:

1. The retention period, to have a time-based cutoff point. It is defined through the Distribution Database Properties and its min_distretention and max_distretention parameters. The default range is 0~72 hours.
2. The publication options `immediate_sync` and `allow_anonymous` and if they are enabled or disabled.
   If these options are enabled, the cleanup will always keep changes up to the upper limit of the retention period range. It prevents from cleaning up rows between the minimum and maximum retention limit (min_distretention and max_distretention).
3. The latest and most recent successful synchronization of a Distribution Agent, as logged on the `MSdistribution_history` distribution system table.
   Any subscription that hasn't synchronized for a long time, but also hasn't been marked as inactive/expired, will prevent the cleanup from removing outdated rows. This might happen if the "Expired subscription clean up" job is disabled. It might also happen if the publication is configured with a long subscription retention, or with subscriptions to never expire.
4. *There are some other dependencies, but these are mostly for on-premise environments and do not apply to Managed Instance.*

Execute the "Calculate Cutoff xact_seqno" script

Please see the script in the More Information section further below. You can run this against the Distribution database, and it will follow the same algorithm steps as the cleanup code. It will return the `xact_seqno` cutoff values for each Publisher database and subscription, and the effective cutoff `xact_seqno` for the database.

Mitigation

Check and configure retention periods

See article How to configure publication and distribution retention.

Make sure that the minimum and maximum distribution retention are set according to the business requirements (default is 0~72 hours). Also make sure that the publication retention is not set to "never expire".

Enable a daily schedule for the Expired subscription clean up

The `Expired subscription clean up` job usually runs once per day at 01:00 in the night. Please check that the job is enabled, that its schedule is set and active, and that is has run on its recent scheduled times. If it hasn't run, start the job manually from the `Job Activity Monitor`.

This should make sure that any outdated subscriptions have been removed and are no longer preventing the cleanup.

Disable the immediate_sync publication option

Discuss with the customer if the `immediate_sync` publication option can be disabled:

- If this option is enabled, it allows that a new or reinitialized subscription can receive the existing, recent snapshot.
- If it is disabled, you must create a new snapshot after creating a new subscription or after marking an existing subscription for reinit.

If no new subscriptions are planned, or if reinitialisations are needed only seldomly, then the recommendation is to disable this option. If new subscriptions are added daily or several times per week, then it makes sense to keep the option enabled. The tradeoff is Distribution database size vs. the impact of creating a new snapshot.

You can use the following script to disable `immediate_sync` - to do so, you also need to disable `allow_anonymous` first:

```
-- use <publication_database>
EXEC sp_changepublication
    @publication = 'Publication_Tran',
    @property = 'allow_anonymous',
    @value = 'false'
GO
EXEC sp_changepublication
    @publication = 'Publication_Tran',
    @property = 'immediate_sync',
    @value = 'false'
GO
```

Check for failed subscriptions that have not run in a while

Ask the customer to look for any subscriptions that haven't run successfully in a while. If there are not many subscriptions, the simplest way to do so is through the Replication Monitor in SSMS.

You may also run the following query against the Distribution database to see the most recent activity datetime for each Distribution Agent:

```
select agent_id, max(a.name) as Agent_Name, max(xact_seqno) as Max_agent_xact_seqno, max(time) as Last_Activit
from distribution..MSdistribution_history h
inner join distribution..MSdistribution_agents a on h.agent_id = a.id
where a.subscriber_id > 0
group by agent_id
```

More Information

The script in the section below is more or less a copy/paste from the stored procedures that calculate the cutoff LSN for the individual Publisher databases. The inline comments are also the original comments from these procs. If you want to see the complete code for yourself, please see the final section at the bottom of this article.

Script to determine the cutoff LSN for each Publisher database and each subscription

```
use distribution

set nocount on

declare @publisher_database_id int
declare @min_cutoff_time datetime
declare @max_cleanup_xact_seqno varbinary(16)
declare @retention int
declare @rptdtl int

--select history_retention, min_distretention, max_distretention from msdb.dbo.MSdistributiondbs
select @retention = max_distretention from msdb.dbo.MSdistributiondbs
--set @retention = 72  -- set value explicitly if you want to test various periods
set @min_cutoff_time = dateadd(hour, -@retention, getdate())

set @publisher_database_id = NULL  --if you want to look for specific database, change this from NULL to value
set @rptdtl = 2  --if you want more information about agents keeping the xact_seqno to a minimum, change this

declare @min_agent_sub_xact_seqno varbinary(16)
declare @max_agent_hist_xact_seqno varbinary(16)
declare @active int
declare @initiated int
declare @agent_id int
declare @min_xact_seqno varbinary(16)

--Loop through each database
if @publisher_database_id is null
begin
declare pub_db scroll cursor for
select id from MSpublisher_databases
end
else
begin
declare pub_db scroll cursor for
select id from MSpublisher_databases where id = @publisher_database_id
end

open pub_db
fetch first from pub_db into @publisher_database_id
while @@FETCH_STATUS = 0
begin

-- set @min_xact_seqno to NULL and reset it with the first prospect of min_seqno we found later
 select @min_xact_seqno = NULL
 select @active = 2
 select @initiated = 3

-- -- cursor through each agent with it's smallest sub xact seqno --
 declare #tmpAgentSubSeqno cursor local forward_only  for
         select a.id, min(s2.subscription_seqno)
         from MSsubscriptions s2 join MSdistribution_agents a on (a.id = s2.agent_id)
         where s2.status in( @active, @initiated ) and
        -- filter out subscriptions to immediate_sync publications
         not exists (select * from MSpublications p where s2.publication_id = p.publication_id and p.immediate
         and a.publisher_database_id = @publisher_database_id
         group by a.id

 open #tmpAgentSubSeqno
 fetch #tmpAgentSubSeqno into @agent_id, @min_agent_sub_xact_seqno

 if (@@fetch_status = -1) -- rowcount = 0 (no subscriptions)
 begin
    -- If we have a publication which allows for init from backup with a min_autonosync_lsn set
    --   we don't want this proc to signal cleanup of all commands
    -- Note that if we filter out immediate_sync publications here as they will already have the
    --   desired outcome.  The difference is that those with min_autonosync_lsn set have a watermark
    --   at which to begin blocking cleanup.
     if not exists (select * from dbo.MSpublications msp join MSpublisher_databases mspd
                    ON mspd.publisher_id = msp.publisher_id and mspd.publisher_db = msp.publisher_db
                    where mspd.id = @publisher_database_id and msp.immediate_sync = 1)
     begin
        select top(1) @min_xact_seqno = msp.min_autonosync_lsn
                from dbo.MSpublications msp join MSpublisher_databases mspd
                ON mspd.publisher_id = msp.publisher_id and mspd.publisher_db = msp.publisher_db
        where mspd.id = @publisher_database_id
         and msp.allow_initialize_from_backup <> 0
         and msp.min_autonosync_lsn is not null
         and msp.immediate_sync = 0
        order by msp.min_autonosync_lsn asc
     end
 end

 while (@@fetch_status <> -1)
 begin
    --always clear the local variable, next query may not return any resultset
     set @max_agent_hist_xact_seqno = NULL
    --find last history entry for current agent, if no history then the query below should leave @max_agent_xa
     select top 1 @max_agent_hist_xact_seqno = xact_seqno from MSdistribution_history
        where agent_id = @agent_id
        order by timestamp desc

     if @rptdtl = 2
     begin
        select @publisher_database_id as Pub_DB_ID, @agent_id as AgentID, @max_agent_hist_xact_seqno as Max_Ag
        @min_agent_sub_xact_seqno as Min_Agent_Sub_Xact_Seqno
     end
         --now find the last xact_seqno this agent has delivered:
         --if last history was written after initsync, use histry xact_seqno otherwise use initsync xact_seqno
         if isnull(@max_agent_hist_xact_seqno, @min_agent_sub_xact_seqno) <= @min_agent_sub_xact_seqno
         begin
           set @max_agent_hist_xact_seqno = @min_agent_sub_xact_seqno
         end
         --@min_xact_seqno was set to NULL to start with, the first time we get here, it'll gets set to a non-
         --then we graduately move to the smallest hist/sub seqno
         if ((@min_xact_seqno is null) or (@min_xact_seqno > @max_agent_hist_xact_seqno))
         begin
           set @min_xact_seqno = @max_agent_hist_xact_seqno
         end

     fetch #tmpAgentSubSeqno into @agent_id, @min_agent_sub_xact_seqno
 end
 close #tmpAgentSubSeqno
 deallocate #tmpAgentSubSeqno

/*
** Optimized query to get the maximum cleanup xact_seqno
** If the query below returns nothing, nothing can be deleted.
** Reset @max_cleanup_xact_seqno to 0.
*/

 if @min_xact_seqno is NULL and @rptdtl = 2 and @agent_id is NULL
    begin
       select @publisher_database_id as Pub_DB_ID, 'Either this is an immediate_sync publication, or the subsc
    end

 select @max_cleanup_xact_seqno = 0x00
 -- Use top 1 to avoid warning message of "Null in aggregate…" which will make
 -- sqlserver agent job having failing status
 select top 1 @max_cleanup_xact_seqno = xact_seqno
   from MSrepl_transactions with (nolock)
   where publisher_database_id = @publisher_database_id
                and (xact_seqno < @min_xact_seqno or @min_xact_seqno IS NULL)
                and entry_time <= @min_cutoff_time
                order by xact_seqno desc

 select @publisher_database_id as Pub_DB_ID, @max_cleanup_xact_seqno as Max_Cleanup_Xact_Seqno
 fetch next from pub_db into @publisher_database_id
```

```
end
close pub_db
deallocate pub_db
```

Source code for the cleanup code

The script in the section above is more or less a copy/paste from the stored procedures that calculate the cutoff LSN for the individual Publisher databases. The inline comments are also the original comments from these procs. If you want to see the code for yourself, you can easily do so by running the following commands in your Distribution database:

```
-- direct output to text or file, not to grid
use distribution
sp_helptext sp_MSdistribution_cleanup
sp_helptext sp_MSdistribution_delete
sp_helptext sp_MSdelete_publisherdb_trans
sp_helptext sp_MSdelete_dodelete
sp_helptext sp_MSmaximum_cleanup_seqno
sp_helptext sp_expired_subscription_cleanup
```

How good have you found this content?