Distribution cleanup Performance issues

Last updated by | Ricardo Marques | Nov 15, 2022 at 3:42 AM PST

Contents

Issue

The customer is running Transactional Replication on their Managed Instance, which is hosting both Publisher and Distributor roles. Recently the DBA noticed that the Distribution database is growing constantly and rapidly, and has already reached a size of several TB. It is threatening to consume the remaining storage space of the MI soon; storage has already been set to the maximum and cannot increase further.

The `Distribution clean up: Distribution` job is running on the default schedule of 10 minutes, but each execution takes significantly longer. On checking the cleanup job history, it becomes obvious that the cleanup rate is very low, similar to this:

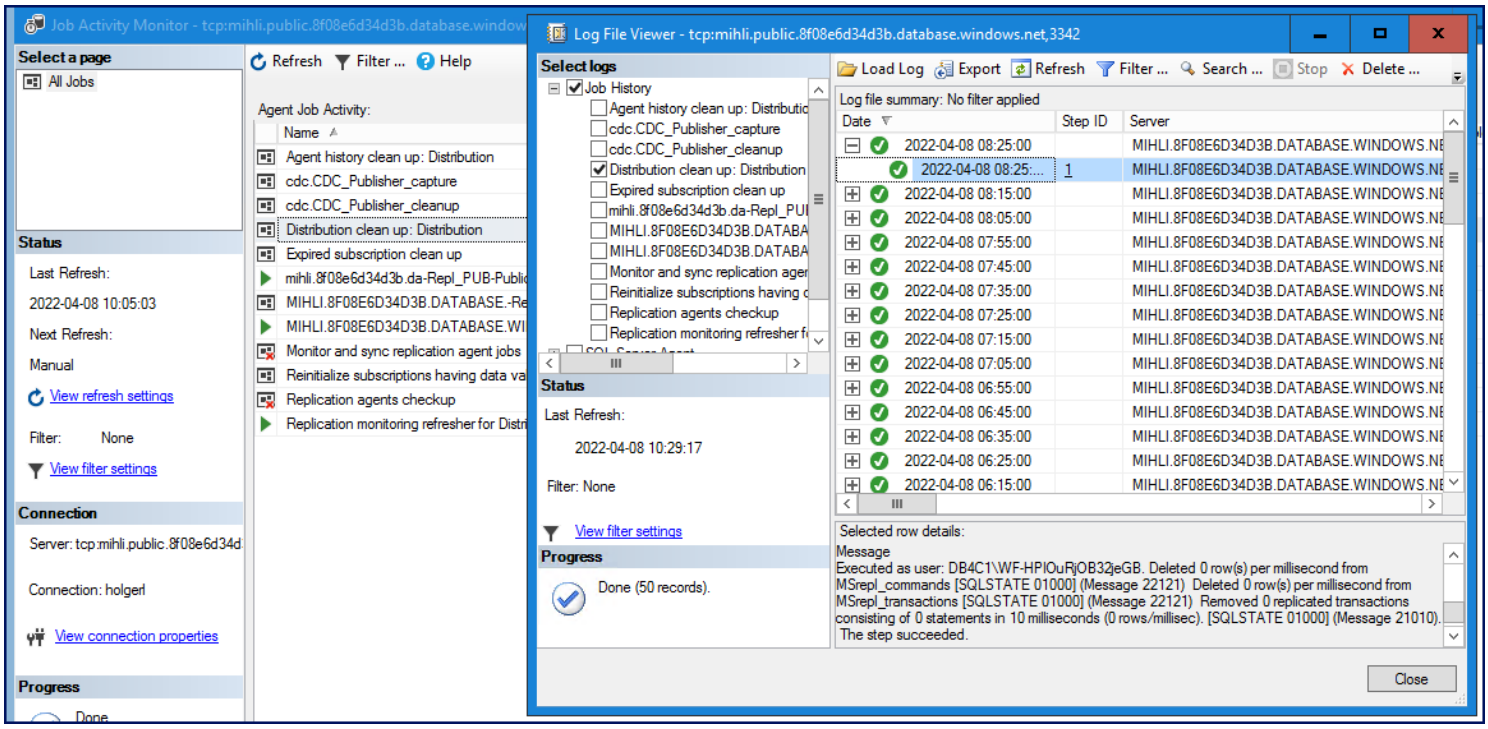
```
Executed as user: DB4C1\WF-oM3QjL6VozKSGhF.
Deleted 0 row(s) per millisecond from MSrepl_commands [SQLSTATE 01000] (Message 22121)
Deleted 0 row(s) per millisecond from MSrepl_transactions [SQLSTATE 01000] (Message 22121)
Removed 17724 replicated transactions consisting of 42008 statements in 11009050 milliseconds (0 rows/millisec
The step succeeded.

(note: 11009050 milliseconds = 11009 seconds = 183 minutes = 3 hours)
```

The customer is looking to improve the cleanup performance, otherwise the MI is going to run out of space very soon.

Investigation / Analysis

Check cleanup job history to confirm the low cleanup rate

- In SSMS, open `Job Activity Monitor` under the `SQL Server Agent` node.
- Right-click on the `Distribution clean up: Distribution` job and select `View history`.
- Expand the most recent entries.
- Confirm that the cleanup has run recently and check the message of the job step, like this:



Check how many transactions and commands are waiting for cleanup

Do not run a "select count(*)" directly on the system table to avoid additional workload on the database; there might be billions of rows! Rather check one of the DMVs or sysindexes instead, like this:

```
select object_name(id), name, indid, dpages, rowcnt, rows, rowmodctr, pgmodctr
from distribution..sysindexes where name in ('ucMSrepl_transactions', 'ucMSrepl_commands')
```

Check how far behind the cleanup has fallen

Run the following query against the Distribution database:

```
select min(entry_time) as 'oldest transaction', max(entry_time) as 'newest transaction'
from distribution..MSRepl_transactions with (nolock)
```

The oldest transaction gives you a rough estimate on the amount of data that needs to be deleted. It will also give you a starting point for running cleanup with a custom retention - see mitigation steps below

Check the file sizes for the Distribution database

This is important to confirm that the space is indeed allocated to the data file, not the transaction log:

```
-- Database file size details - execute in Distribution database:
select db_name() as DBName, file_id, type, type_desc, state_desc,
  'SpacedUsedinMB' = cast(cast((cast(fileproperty(name, 'spaceused' )as int)/128.0) as numeric(15,2))as nvarch
  'AllocatedSizeinMB' = cast(size/128.0 as numeric(15,2)),
  'MaxSizeinMB' = cast(max_size/128.0 as numeric(15,2)),
  'FreeSpaceinMB' = CONVERT(decimal(12,2),ROUND((size-fileproperty(name,'SpaceUsed'))/128.000,2)),
  'SpacedUsedPCTAlloc' = cast(((cast(fileproperty(name, 'spaceused' )as int)/128.0)/(size/128.0) * 100) as num
  'SpacedUsedPCTMax' = cast((cast(size/128.0 as numeric(15,2))/cast(max_size/128.0 as numeric(15,2)) * 100) as
  growth
FROM sys.database_files
```

If the MI is a General Purpose instance:
Please also check ASC Distribution database storage file size (P10? P50?). Compare it to the File IO characteristics in General Purpose tier and see if increasing the file size would make any sense.

Check if the cleanup spids are blocked by another process

The session that is performing the cleanup might be blocked by another process. Likely blockers are the Log Reader Agent or a Distribution Agents. It is also possible that the cleanup session is waiting on an internal spid, like "-5".

See article Log Reader and Distribution Cleanup blocking each other for steps to check for blocking.

Mitigation

See the More Information section for technical background details. The risk of this situation has two aspects:

- You can increase the Delete batch sizes of the cleanup procedure, so that each Delete iteration removes more rows at a time
- But this would increase the risk of lock escalation and the blocking of the Log Reader Agent.

The mitigation will be a race between the progress of the Distribution cleanup vs. the Publisher database running out of transaction log space.

Update Statistics on Distribution system tables

This will be important to allow for the Delete statements to get an efficient execution plan.

```
-- Stop the cleanup agent job

-- Remove the automatically-created statistics, if they exist:
-- generate a set of commands for the drop operation:
use distribution;
select 'DROP STATISTICS [' + ss.name + '].[' + so.name + '].[' + si.name + ']'
from sysobjects so join sysindexes si on so.id=si.id join sys.schemas ss on so.uid=ss.schema_id
where si.name like '_WA_%' and so.type ='U' and so.name in ('MSsubscriptions', 'MSdistribution_history','Msrep
-- then execute the generated commands to remove the auto-stats

-- Update Statistics:
UPDATE STATISTICS MSsubscriptions WITH FULLSCAN
UPDATE STATISTICS MSdistribution_history WITH FULLSCAN
UPDATE STATISTICS Msrepl_transactions WITH FULLSCAN
UPDATE STATISTICS Msrepl_commands WITH FULLSCAN

-- Start the cleanup agent job
```

Try to remove any blocking

If you detected that the cleanup is blocked by the Log Reader or Distribution Agents, you might consider stopping these agents to let cleanup progress. If you stop the Log Reader Agent, you must closely monitor the transaction log size of the Publisher database. Start the Log Reader again if the log is 70% full.

If the blocking occurs on an internal session "-5', then follow the mitigation steps in article Log Reader and Distribution Cleanup blocking each other.

Scale to the largest possible instance size

General Purpose with at least 16 vCores and Business Critical Premium allow up to 16 TB of storage. The customer might consider scaling up to gain more time, if they haven't done so yet.

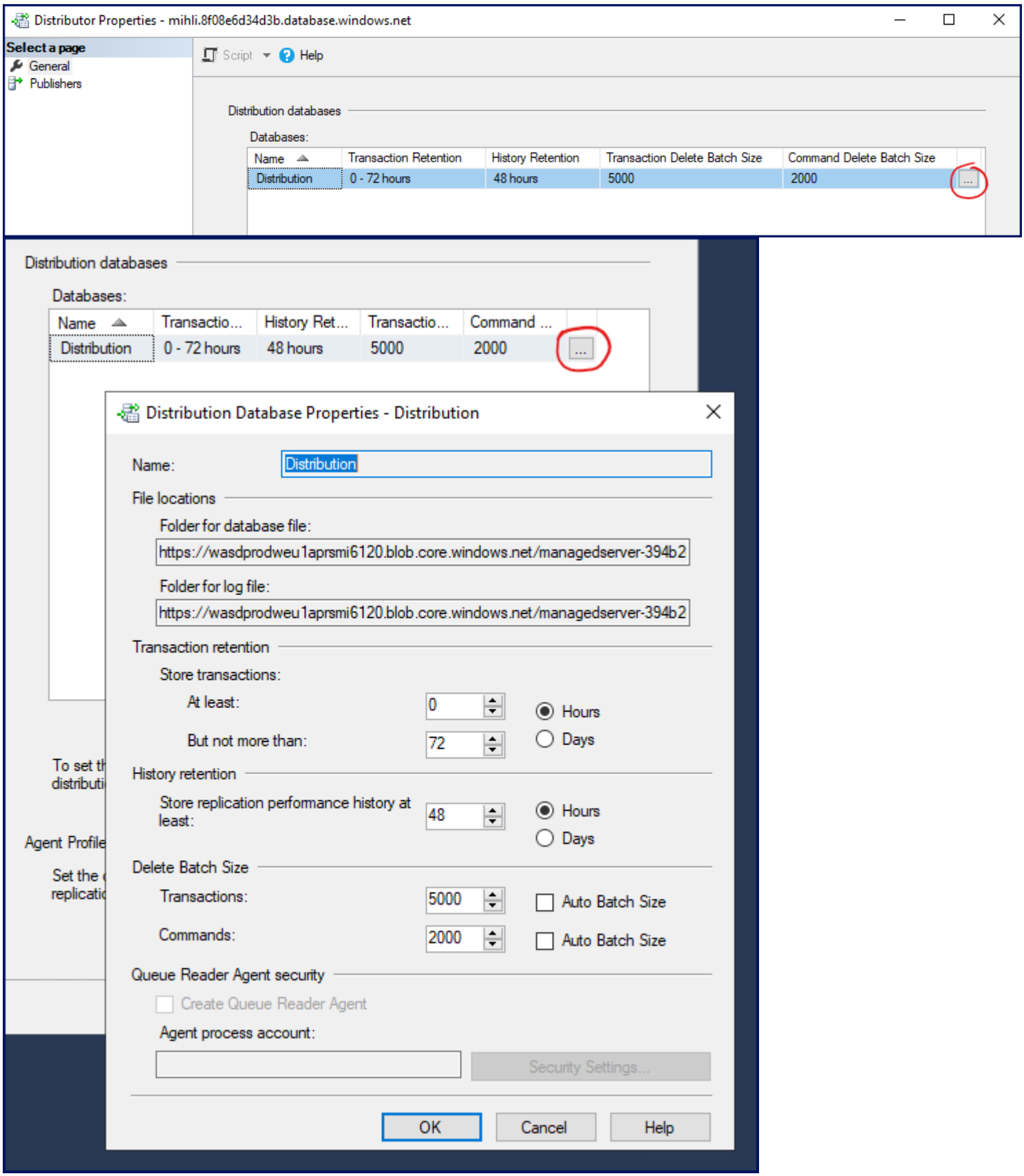
See Service tier characteristics for limitations of each service tier configuration.

Run the cleanup with increased Delete batch sizes

Set the Delete batch sizes to much higher values: for transactions to 100000 (from 5000), for commands to 50000 (from 2000).

This will delete a lot more rows per loop iteration, resulting in fewer iterations and better overall Delete performance. It will likely block the Log Reader and Distribution Agents though, and you might consider stopping these agents to give the cleanup priority.

You can change the Delete batch sizes on the "Distribution Database Properties" page, in the Delete Batch Size section:



Or you can execute the following SQL script to change the values:

```
begin transaction
select deletebatchsize_cmd, deletebatchsize_xact from msdb.dbo.MSdistributiondbs
update msdb.dbo.MSdistributiondbs set deletebatchsize_xact = 100000, deletebatchsize_cmd = 50000
select deletebatchsize_cmd, deletebatchsize_xact from msdb.dbo.MSdistributiondbs
commit
```

Run the cleanup in steps with decreasing retention

Get the oldest transaction as described in the Investigation section above. If it is, for example, 20 days behind, you can specify a retention of 18 days = 432 hours so that the Delete queries consider fewer rows. This might allow you to run an increased Delete batch size without risking lock escalation and blocking the replication agents.

The steps for this approach are:

- Stop the `Distribution clean up: Distribution` job and disable its schedule (to avoid the automatic restart)
- Determine a retention period that includes only 1 or 2 days of data for cleanup, e.g. 432 hours.
- Open an SSMS query window and execute the following stored procedure. Set both parameters to the same value:
  `EXEC dbo.sp_MSdistribution_cleanup @min_distretention = 432, @max_distretention = 432`
- When it has finished, re-run the same procedure and reduce the parameter values by a day:
  `EXEC dbo.sp_MSdistribution_cleanup @min_distretention = 408, @max_distretention = 408`
- Continue this until you have reached the default retention of 72 hours.
- Re-enable the schedule for the cleanup job and let it run normally.

Or if you want to automate it, you can easily use smaller steps like e.g. 5 hours:

```
use distribution
declare @i int = 432;
declare @final_retention int = 72;
declare @step int = 5;

while @i >= @final_retention
begin
    print @i;
    begin try
        exec distribution.dbo.sp_MSdistribution_cleanup @min_distretention = @i, @max_distretention = @i
        set @i -= @step;
    end try
    begin catch
        -- do nothing. This will skip decreasing the hours if a deadlock happens
        print N'Distribution cleanup encountered error ' + cast(ERROR_NUMBER() as varchar) + '. Iterator will
    end catch
end
```

Add data files to the Distribution database to increase I/O performance

This step is not feasible if you are already low on disk space. But it might be an option to help avoiding the issue in the future.

```
ALTER DATABASE [distribution]
ADD FILE ( NAME = N'data_1', SIZE = 16MB , MAXSIZE = 20GB , FILEGROWTH = 100MB ) TO FILEGROUP [PRIMARY]
```

Use remote distributor

According with enhancing Transactional Replication performance, it is generally advised to configure distributor on a dedicated server.

When you have more than one very busy databases, this becomes even more relevant. In some cases, having one remote distributor for each publisher will allow that you will have separate logreaders, cleanup processes and distribution agents.

More Information

The top-level call for the cleanup job is to the stored procedure `EXEC dbo.sp_MSdistribution_cleanup @min_distretention = 0, @max_distretention = 72` . This is then calling several nested stored procedures to determine the Log Sequence Number (LSN) for the retention cutoff and to perform the actual Deletes. Any row that is older than the cutoff LSN will be removed through a DELETE TOP command in a WHILE 1=1 loop.

The cleanup itself then pursues the following strategy:

- A cursor is created which is looking through MSrepl_commands for any outdated snapshot-related entries. These contain the location of the snapshot UNC folder that will be deleted.
- The commands will be deleted from MSrepl_commands in a loop with a TOP 2000 clause.
- The transactions will be deleted from MSrepl_transactions in a loop with a TOP 5000 clause, in the `sp_MSdelete_publisherdb_trans` and `sp_MSdelete_dodelete` procedure (you might see those procs if cleanup is involved in blocking).

It is the DELETE TOP 5000 and DELETE TOP 2000 loops which can make the cleanup very slow. It is usually very efficient if the Distribution database has a "normal" size. But when there is a large backlog of commands and transactions, the loop has to run for many iterations. And if the execution plan for this Delete is somehow inefficient, then this loop could run for a very long time.

The TOP clause is good to help reduce blocking and help prevent lock escalation, but the side effect is that if millions of rows needs to be deleted, then this WHILE 1=1 loop needs to be repeated more than 500 times.

Public Doc Reference

Hardware generation characteristics

Service tier characteristics

File IO characteristics in General Purpose tier

Internal Reference

Log Reader and Distribution Cleanup blocking each other

IcM 299167849
IcM 286971791
IcM 343167142

How good have you found this content?