CDC Transaction log growing or full

Last updated by | Ricardo Marques | Mar 24, 2023 at 4:40 AM PDT

Contents

- Issue
- Investigation / Analysis
  - Check the transaction log size and database file size
  - Check what is preventing log truncation
  - Check if the database was created using a backup
  - Check CDC progress and errors in Kusto
  - Check CDC log scan history, transaction workload, and spa...
  - Scenario "ACTIVE_TRANSACTION"
  - Capture Job Status
- Mitigation
  - Mitigation 1 - Kill the long-running transactions
  - Mitigation 2 - Try increasing the transaction log
  - Mitigation 3 - Involve the Product Group
  - Mitigation 4 - Throttle the workload or scale up
  - Mitigation 5 - CDC job is disabled
  - Mitigation 6 - Last option - Reset CDC by executing sp_rep...
  - Mitigation 7 - CDC on restored database
  - Further recommendations for avoiding this type of issue
- Internal reference

Issue

The customer is using Change Data Capture (CDC) on Managed Instance. While monitoring the MI storage allocation, they noticed that the transaction log of their CDC-enabled database is constantly growing.

Finally their applications are starting to fail with errors like these:

```
Error: 9002
The transaction log for database 'e50a08c3-df67-4238-a96f-c499393d1faf' is full
due to 'ACTIVE_TRANSACTION' and the holdup lsn is (5270698:13168:229).
```

This article will help you with both the "growing" and "full" scenarios.

Investigation / Analysis

This is very similar to the issue described in Disable CDC fails with error 22831.

Common reasons for a full transaction log include:

- Transaction Log not being truncated (active transaction, unreplicated commands, log backups not running)
- Disk volume is full (database unable to grow because the Managed Instance runs out of storage)
- Log size is set to a fixed maximum value or autogrow is disabled
- HADR replication or Availability Group synchronization that is unable to complete
- due to high workload, CDC is not able to compete.
- the database was restored containing CDC (CDC will be enabled but no CDC jobs will be present).

The following steps give you more background information on the likely cause.

Check the transaction log size and database file size

The easiest way for you to see this is through XTS view "LogNearFull-LogFull Replication MI.xts". Enter the MI name and database name into the top-left section and it will show you the current transaction log details, including current size and `log_reuse_wait_desc`.

For the customer: The output of the following queries will give you more information about the possible size limitations:

```
USE <CDC_enabled_database>;

-- Log size details:
SELECT db_name(database_id) as [Database name],
  total_log_size_in_bytes*1.0/1024/1024 as [total_log_size_in_MB],
  used_log_space_in_bytes*1.0/1024/1024 as [used_log_space_in_MB],
  used_log_space_in_percent,
  (total_log_size_in_bytes - used_log_space_in_bytes)*1.0/1024/1024 AS [free log space in MB]
FROM sys.dm_db_log_space_usage;

-- Database file size details:
select db_name() as DBName, file_id, type, type_desc, name, 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)),
  'SpacedUsedPercentAlloc' = cast(((cast(fileproperty(name, 'spaceused' )as int)/128.0)/(size/128.0) * 100) as
  'SpacedUsedPercentMax' = cast((cast(size/128.0 as numeric(15,2))/cast(max_size/128.0 as numeric(15,2)) * 100
  growth
FROM sys.database_files
```

On either resultset, the free space indicates how much time you have before the transaction log becomes full. If it is already full, the free space would be close to zero or even negative.

Note the `name` column on the 2nd resultset - you will need this name for the mitigation steps further below.

Check what is preventing log truncation

The easiest way for you to see this is through XTS view "LogNearFull-LogFull Replication MI.xts". Enter the MI name and database name into the top-left section and it will show you the current transaction log details, including current size and `log_reuse_wait_desc`.

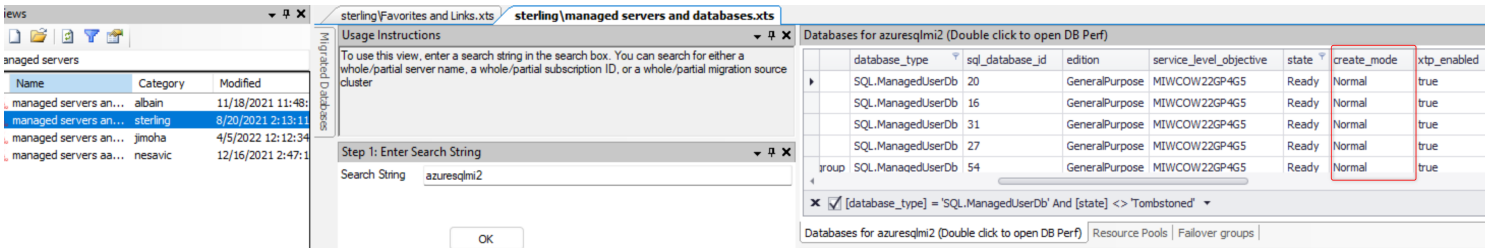
For the customer: To discover what is preventing log truncation, refer to the log_reuse_wait_desc output in sys.databases. The log reuse wait informs you what conditions or causes are preventing the transaction log from being truncated by a regular log backup.

```
SELECT [name], log_reuse_wait_desc FROM sys.databases;
```

The log_reuse_wait_desc values of `ACTIVE_TRANSACTION` and `REPLICATION` will be covered in this article. For other scenarios, please refer to articles Transaction Log full due to backups and Troubleshooting transaction log errors with Azure SQL Database and Azure SQL Managed Instance.

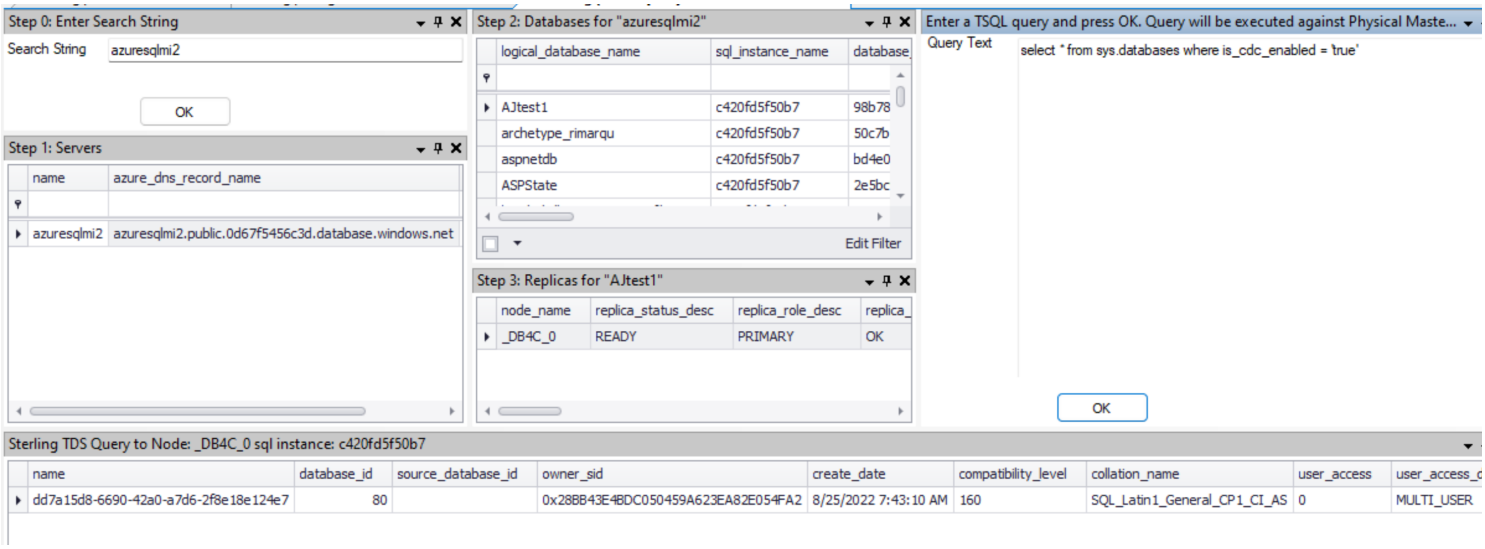
Check if the database was created using a backup

On XTS go to "Managed servers and databases.xts" view and confirm if the create mode is Restore



If yes, go to adhocquerytobackendinstance.xts view and run the query below to check if cdc is enabled

```
select * from sys.databases where is_cdc_enabled = 'true'
```



After this, on customer side run EXEC msdb..sp_help_job and ensure capture job is available. You should see an entry like below, with the category Repl-LogReader.

SQLQuery1.sql - az...ter (miadmin (111))*

```
EXEC msdb..sp_help_job
```

100 %

Results | Messages

	job_id	originating_server	name	enabled	description	start_step_id	category	owner
13	7D7A26E8-B038-4090-8D40-5857AD34B7F1	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	Monitor and sync replication agent jobs	1	Sync replication agent jobs and their enabled states.	1	REPL-Checkup	miadmi
14	4C7F7267-EC69-4B99-8F4E-5AF7FC78BD74	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	CommandLog Cleanup	1	Source: https://ola.hallengren.com	1	Database Maintenance	sa
15	FB378CDF-E9BF-4D28-A6C9-5B6E9C490F14	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	Agent history clean up: distribution	1	Removes replication agent history from the distrib...	1	REPL-History Cleanup	miadmi
16	98B029DD-C459-47B2-A029-5E886498E6F6	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	dfdsafds	1	No description available.	1	[Uncategorized (Local)]	miadmi
17	762CBE3F-27DE-4362-9065-5EE26CFB5408	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	SSIS Server Maintenance Job	0	Runs every day. The job removes operation record...	1	[Uncategorized (Local)]	##MS_
18	0ECCEDC6-9B46-46F7-834A-6BD9355EFBDC	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	Back Up Database - saalbawDB	1	No description available.	1	[Uncategorized (Local)]	miadmi
19	F3A9D1A5-ABD6-4137-BCE1-7785EDFC5C86	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	DatabaseIntegrityCheck - USER_DATABAS...	1	Source: https://ola.hallengren.com	1	Database Maintenance	sa
20	4209D354-982F-4605-A65A-7E1E0761C41E	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	teste_rimarqu	1	No description available.	1	[Uncategorized (Local)]	miadmi
21	2051C8A7-23DC-481D-8A2F-87935A6C9735	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	test stop job	1	No description available.	1	[Uncategorized (Local)]	miadmi
22	C9A01862-5FF7-4D19-AC06-88CB5BC29FF3	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	IndexOptimize - USER_DATABASES	1	Source: https://ola.hallengren.com	1	Database Maintenance	sa
23	DDF845D7-402C-42AA-B22A-8CF1C1D4D429	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	MoRaja_TEST18.10	1	No description available.	1	[Uncategorized (Local)]	miadmi
24	2FF16741-4C45-41D4-A6B2-99AF45E59136	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	test DbMail	1	No description available.	1	[Uncategorized (Local)]	miadmi
25	1CB3D55F-5F23-44FF-92E8-9DB86AC557CF	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	Replication agents checkup	1	Detects replication agents that are not logging hist...	1	REPL-Checkup	miadmi
26	D7ECE8F0-F0AE-4EF0-A96B-9E2E45615E40	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	tar1	1	No description available.	1	[Uncategorized (Local)]	miadmi
27	341C9483-850C-41AA-90E2-A11A6D86730D	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	TestFoGConnection	1	No description available.	1	[Uncategorized (Local)]	replicat
28	75B9F35C-637C-443F-940B-A1FB884ECE2B	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	cdc.tarasheetest_cleanup	1	CDC Cleanup Job	1	REPL-Checkup	sa
29	808CF5A5-E279-412D-A783-A66855CB8CAC	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	TestTCPNetworkConnection	1	No description available.	1	[Uncategorized (Local)]	sa
30	A3876948-8C0F-4E19-8072-A92253A97EA4	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	TestConnection	1	No description available.	1	[Uncategorized (Local)]	miadmi
31	4E118B19-AA13-4083-985C-AED4938D1A26	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	sp_purge_jobhistory	1	Source: https://ola.hallengren.com	1	Database Maintenance	sa
32	2C44AD43-F459-46B4-ADA5-B0692639B869	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	test	1	No description available.	1	[Uncategorized (Local)]	miadmi
33	F36AC418-2E25-4D43-81A6-B22E85A13656	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	DatabaseIntegrityCheck - SYSTEM_DATAB...	1	Source: https://ola.hallengren.com	1	Database Maintenance	sa
34	53480652-379C-443D-9888-B621C4BCC177	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	TestNetConnection	1	No description available.	1	[Uncategorized (Local)]	miadmi
35	DDBDB8CE-6F4E-4199-B5B6-B8F5623D9DE2	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	Distribution clean up: distribution	1	Removes replicated transactions from the distribut...	1	REPL-Distribution Cleanup	miadmi
36	999C8968-5AEA-4B4E-9C1E-C58E844976E4	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	testpowershell	1	No description available.	1	[Uncategorized (Local)]	miadmi
37	44358807-1051-46E8-8260-C8BB88678653	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	DBmailcheck	1	No description available.	1	[Uncategorized (Local)]	miadmi
38	BA4764F5-8EE6-47AF-BCA4-CB33BD516A1B	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	cdc.tarasheetest_capture	1	CDC Log Scan Job	1	REPL-LogReader	sa
39	2C8AC759-9A47-464D-81CD-CC62B43016D7	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	testagentrm	1	No description available.	1	[Uncategorized (Local)]	rimarqu
40	9D5607CD-0E96-4803-BEA7-CE4E0D1F410D	AZURESQLMI2.0D67F5456C3D.DATABASE.WINDOWS.NET	master job	1	No description available.	1	[Uncategorized (Local)]	miadmi

If the capture job is not shown by the customer, jump to mitigation 7.

Check CDC progress and errors in Kusto

In a first step, check for any errors that might have been logged for the CDC capture job - especially if the transaction log is already full. If there is no error yet, see the 2nd Kusto query to check what phase it is currently in.

```
MonCDCTraces
| where TIMESTAMP >= datetime(2022-05-24 09:00:00Z)
| where TIMESTAMP <= datetime(2022-05-24 10:00:00Z)
| where LogicalServerName == "servername"
| where logical_database_name == "databasename"
| where event == "cdc_error"
| project TIMESTAMP, session_id, error_number, error_severity, error_state, error_message, begin_lsn, start_ls

-- sample output:
TIMESTAMP                   session_id error_number error_severity error_state error_message          begin_ls
2022-05-24 09:40:40.0404405 675332     9002         17             4           Filtered GDPR reasons. 00513EF9
2022-05-24 09:40:40.0404405 675332     18805        16             1           Filtered GDPR reasons. 00513EF9
2022-05-24 09:40:40.0404405 675332     22859        16             2           Filtered GDPR reasons. 00000000

-- session_id==675332
```

Note that the first error in the list (error 9002) is the main cause; the other errors 18805 and 22859 are only collateral messages after the log became full.

You can then use the session_id to filter on this specific capture instance execution (do not filter on session_id if the 1st query came back empty). Widen the time window to see all events related to this issue:

```
MonCDCTraces
| where TIMESTAMP >= datetime(2022-05-24 05:00:00) and TIMESTAMP <= datetime(2022-05-27 10:45:00)
| where logical_database_name == "databasename" and LogicalServerName == "servername"
| where event != "cdc_cleanup_job_status"
| where session_id == 675332
| project originalEventTimestamp, NodeName, AppName, session_id, event, tran_count, latency, error_number, dur
| union
(
MonLogReaderTraces
| where TIMESTAMP >= datetime(2022-05-24 05:00:00) and TIMESTAMP <= datetime(2022-05-27 10:45:00)
| where logical_database_name == "databasename" and LogicalServerName == "servername"
| where session_id == 675332
| project todatetime(originalEventTimestamp), session_id, phase_number, phase_state_name, repldone_phase, repl
)

-- sample output:
originalEventTimestamp      NodeName AppName      session_id event error_number phase_number phase_state_name
2022-05-24 05:07:42.8895309                       675332                        1            phase_start
2022-05-24 05:07:42.8896040                       675332                        1            phase_end
2022-05-24 05:07:42.8896158                       675332                        2            phase_start
2022-05-24 05:07:43.0330216                       675332                        2            phase_end
2022-05-24 05:07:43.0330271                       675332                        6            phase_start
2022-05-24 05:07:43.0331928                       675332                        6            phase_end
2022-05-24 05:07:43.0331960                       675332                        7            phase_start
2022-05-24 09:38:26.9885181                       675332                        7            phase_progress
2022-05-24 09:40:39.5946586 DB128C.1 c41bc79db62c 675332        cdc_error       9002
2022-05-24 09:40:39.5949149 DB128C.1 c41bc79db62c 675332        cdc_error       18805
2022-05-24 09:40:39.5950059 DB128C.1 c41bc79db62c 675332        cdc_error       22859
2022-05-24 09:40:39.5950851                       675332                        7            phase_end
2022-05-24 09:41:39.8759939                       675332
2022-05-24 09:41:39.9416196                       675332
2022-05-24 09:42:39.9896571                       675332
2022-05-24 09:42:40.0422583                       675332
(...)
```

If there is no error yet (log still growing but not full yet), you may still use this second query without the filter on session_id and look for the latest phase 7 start and progress to identify the wait resource. As you can see from the output above, phase 7 was running for more than 4 hours before it failed with error 9002. The wait_stats column shows you the reason; here it was blocked, waiting on a LCK_M_SCH_S lock. In a scenario like this, the CDC capture job was likely blocked by a long-running transactions.

Check CDC log scan history, transaction workload, and sparse replicated transactions

The "log full or growing" issue might also occur if the CDC scan job is either not running at all or is encountering a massive workload. The workload might be related to either replicated or unreplicated transactions.

You can determine this through the following `MonLogReaderTraces` Kusto query over a longer recent period. It reports the "phase_end" events and thus the regular progress of the log scan operations:

```
MonLogReaderTraces
| where TIMESTAMP > datetime(2022-09-10 09:00:00)
| where TIMESTAMP < datetime(2022-09-14 06:00:00)
| where LogicalServerName =~ "servername"
| where logical_database_name =~ "databasename"
| where event == "repl_logscan_session" and phase_number in (2, 7) and phase_state_name == "phase_end"
| project originalEventTimestamp, session_id, NodeName,phase_number, phase_state_name, max_trans, tran_count,
vlfs = toint(strcat("0x", substring(end_lsn, 0, 8)))-toint(strcat("0x", substring(start_lsn, 0, 8))),
duration = todatetime(originalEventTimestamp)-todatetime(start_time), wait_stats
| order by originalEventTimestamp asc
```

The sample output below gives hints to several conclusions - this provides some guidance on details to look for in your own scenario:

- There is regular output, so the capture job was running continuously and was not stopped. If you see nothing for the past days, then check if the CDC capture job was running at all.
- Each iteration of phase 2/7 takes about 40~45 minutes, which is very long. It indicates that the log scan operation is struggling with heavy workload.
- On each iteration, the log reader scans 500 transactions (column `max_trans`). The `tran_count` column however shows much lower values. This indicates that after scanning 500 transactions on the transaction log, only the lower `tran_count` number was identified to be replicated. It is the symptom of heavy workload but sparse replicated transactions.
- The large number in column `log_record_count` confirms the high workload on the CDC database.
- The `vlfs` column indicates the number of virtual log files that were required to scan.
- The `wait_stats` column provides information what each phase was waiting on most.

originalEventTimestamp	session_id	NodeName	phase_number	tran_count	log_record_count	start_lsn	end_lsn	start_time	command_count	vlfs	duration	wait_stats
max_trans: 500; phase_state_name: phase_end												
2022-09-11 04:08:12.3239574	16	DB.42	7	66	341299964	00010D13:00BCEFE8:0017	00010D19:00868C50:010E	2022-09-11 03:36:20.2400000	197140	6	00:31:52.0839574	<waitstats><wait nam
2022-09-11 04:39:53.0114394	17	DB.42	2	97	314318778	00010D19:008724B0:00CC	00010D1E:01F2AD08:000C	2022-09-11 04:24:53.3600000	0	5	00:14:59.6514394	<waitstats><wait nam
2022-09-11 04:55:17.1505267	17	DB.42	7	97	314318778	00010D19:00869848:012E	00010D1E:01F2AD08:000C	2022-09-11 04:24:53.3600000	93335	5	00:30:23.7905267	<waitstats><wait nam
2022-09-11 05:24:48.9999472	18	DB.42	2	124	313250482	00010D1F:004471E8:0010	00010D24:013EDA50:01AD	2022-09-11 05:09:49.3700000	0	5	00:14:59.6299472	<waitstats><wait nam
2022-09-11 05:40:25.0435284	18	DB.42	7	124	313250482	00010D1F:00446C08:0005	00010D24:013EDA50:01AD	2022-09-11 05:09:49.3700000	95355	5	00:30:35.6735284	<waitstats><wait nam
2022-09-11 06:12:00.4542967	19	DB.42	2	122	358337961	00010D25:00329280:007B	00010D2B:004ADA78:0024	2022-09-11 05:56:39.2166666	0	6	00:15:21.2376301	<waitstats><wait nam
2022-09-11 06:26:38.6771511	19	DB.42	7	122	358337961	00010D25:00327938:0080	00010D2B:004ADA78:0024	2022-09-11 05:56:39.2166666	109225	6	00:29:59.4604845	<waitstats><wait nam
2022-09-11 06:57:08.2234033	20	DB.42	2	70	258924034	00010D2B:0061C6C8:0065	00010D2F:012E8168:038B	2022-09-11 06:41:57.7733333	0	4	00:15:10.4500700	<waitstats><wait nam
2022-09-11 07:08:30.6365779	20	DB.42	7	70	258924034	00010D2B:00604830:0098	00010D2F:012E8168:038B	2022-09-11 06:41:57.7733333	83634	4	00:26:32.8632446	<waitstats><wait nam
2022-09-11 07:36:33.9365955	21	DB.42	2	97	340984936	00010D2F:01368CF0:0216	00010D35:01393980:0063	2022-09-11 07:21:18.5033333	0	6	00:15:15.4332622	<waitstats><wait nam
2022-09-11 07:51:14.2842473	21	DB.42	7	97	340984936	00010D2F:01340A68:028A	00010D35:01393980:0063	2022-09-11 07:21:18.5033333	139092	6	00:29:55.7809140	<waitstats><wait nam
2022-09-11 08:21:44.0900796	22	DB.42	2	62	210755684	00010D35:01395580:0034	00010D3A:01CA4E30:016B	2022-09-11 08:06:27.2133333	0	5	00:15:16.8767463	<waitstats><wait nam
2022-09-11 08:35:31.2532835	22	DB.42	7	62	210755684	00010D35:01395490:0033	00010D3A:01CA4E30:016B	2022-09-11 08:06:27.2133333	101060	5	00:29:04.0399502	<waitstats><wait nam
2022-09-11 09:10:52.5251851	23	DB.42	2	42	245831667	00010D3A:01CAF9D8:00C1	00010D3D:01126948:0005	2022-09-11 08:55:26.3733333	0	3	00:15:26.1518518	<waitstats><wait nam
2022-09-11 09:24:11.9115016	23	DB.42	7	42	245831667	00010D3A:01CA4FF8:0001	00010D3D:01126948:0005	2022-09-11 08:55:26.3733333	61644	3	00:28:45.5381683	<waitstats><wait nam
2022-09-11 09:47:55.0292209	24	DB.42	2	59	250864137	00010D3D:011336F8:0008	00010D42:019E1188:0027	2022-09-11 09:32:43.8700000	0	5	00:15:11.1592209	<waitstats><wait nam
2022-09-11 10:05:03.6517658	24	DB.42	7	59	250864137	00010D3D:011289F8:000E	00010D42:019E1188:0027	2022-09-11 09:32:43.8700000	230037	5	00:32:19.7817658	<waitstats><wait nam
2022-09-11 10:36:42.3841207	25	DB.42	2	53	292339348	00010D43:000455B0:0034	00010D48:00883348:0073	2022-09-11 10:21:40.0733333	0	5	00:15:02.3107874	<waitstats><wait nam
2022-09-11 10:51:23.1055860	25	DB.42	7	53	292339348	00010D43:00044E68:0003	00010D48:00883348:0073	2022-09-11 10:21:40.0733333	156117	5	00:29:43.0322527	<waitstats><wait nam
2022-09-11 11:21:48.1204291	26	DB.42	2	36	217020753	00010D48:009AEA10:007C	00010D4D:000AA030:0041	2022-09-11 11:06:15.2100000	0	5	00:15:32.9104291	<waitstats><wait nam
2022-09-11 11:37:27.6241466	26	DB.42	7	36	217020753	00010D48:00968CB8:0011	00010D4D:000AA030:0041	2022-09-11 11:06:15.2100000	746021	5	00:31:12.4141466	<waitstats><wait nam
2022-09-11 12:20:33.4205399	1	DB.53	2	48	275915221	00010D4D:0011C668:0068	00010D51:01DB31C8:00B1	2022-09-11 12:04:54.4666666	0	4	00:15:38.9538733	<waitstats><wait nam
2022-09-11 12:26:31.2759677	1	DB.53	7	48	275915221	00010D4A:003164C8:000C	00010D51:01DB31C8:00B1	2022-09-11 12:04:54.4666666	74381	7	00:21:36.8093011	<waitstats><wait nam
2022-09-11 12:42:01.6707605	1	DB.10	2	46	199095268	00010D4D:0011C668:0068	00010D51:009411C8:003A	2022-09-11 12:26:52.2266666	0	4	00:15:09.4440939	<waitstats><wait nam
2022-09-11 13:05:27.6592416	1	DB.53	2	50	280234205	00010D4D:0011C668:0068	00010D51:01F45140:009E	2022-09-11 12:50:25.5833333	0	4	00:15:02.0759083	<waitstats><wait nam
2022-09-11 13:33:05.7184584	1	DB.37	2	48	275915221	00010D4D:0011C668:0068	00010D51:01DB31C8:00B1	2022-09-11 13:16:57.3300000	0	4	00:16:08.3884584	<waitstats><wait nam
2022-09-11 14:33:51.7608374	1	DB.37	7	48	275915221	00010D4A:003164C8:000C	00010D51:01DB31C8:00B1	2022-09-11 13:16:57.3300000	797877	7	01:16:54.4308374	<waitstats><wait nam
2022-09-12 15:45:19.2605278	1	DB.44	2	6	932608	00010DA9:014706A8:009E	00010DA9:014B88B8:008A	2022-09-12 15:45:19.1966666	0	0	00:00:00.0638612	<waitstats><wait nam

Scenario "ACTIVE_TRANSACTION"

From the initial error message, the transaction log became full due to `ACTIVE_TRANSACTION`, which indicates a long-running transaction.

You can identify long-running transactions in ASC. Run the troubleshooter on the database and check the ASC report under Performance - Blocking and Deadlocking - Top 20 Long Running Transactions.

Please also check the public article [Log truncation prevented by an active transaction] for the DMV query to find uncommitted or active transactions and their properties.

In this specific case, the customer identified that two jobs were holding long transactions: a maintenance job for index maintenance, and the CDC capture job. The CDC capture job also appeared to be intermittently blocked by the maintenance job and the CDC cleanup job.

They decided to stop the maintenance job and to kill its connections. This didn't resolve the issue though. Instead, the error message changed to a different reason but with the same `holdup lsn`:

```
The transaction log for database 'e50a08c3-df67-4238-a96f-c499393d1faf' is full
due to 'REPLICATION' and the holdup lsn is (5270698:13168:229).
```

The CDC capture job simply took over the position of the maintenance job. The capture job tries to insert rows into the "_CT" capture tables, which is an operation that is again logged in the already full transaction log.

Capture Job Status

Also there might be a case where the CDC capture job is not enabled. When CDC is enabled on a database the transaction log will retain data until it is processed by CDC.

Mitigation

You can only take a choice of actions as long as the transaction log is still in the growing phase and hasn't reached the "log full" state yet. As soon as the transaction log is full, your only chance is to increase the size of the transaction log (Mitigation 2) and hope that the available space is sufficient to clear the issue. If this is not possible, then the only option is to reset the CDC replication (Mitigation 5), which means throwing away all pending CDC changes and having to re-initialize the CDC consumers.

A possible cause is that the CDC cleanup and capture jobs are creating lock contention, as both are trying to write and delete rows to/from the same "_CT" CDC table. In addition, if a maintenance job puts a heavy load on the database or even rebuilds the index of the "_CT" table, it might contribute significantly and increase the overall impact.

If the CDC capture job is then unable to progress, it will prohibit the log backup job from truncating the transaction log because it cannot remove the unreplicated commands. Meanwhile the normal incoming workload, the maintenance job, and the cleanup job will continue to fill the transaction log with new transactions.

If the issue is related with a database that was restored with CDC enabled use Mitigation 7.

Mitigation 1 - Kill the long-running transactions

If you have identified long-running transactions, you might try to stop the associated applications or jobs. Or if you cannot relate the source of the transactions, you could try to kill the session_id/spid.

Note though that depending on the workload covered in the open transaction, the transaction might go into a state of `ROLLBACK` and not finish immediately. If the transaction was open several hours or days, it might take several hours to rollback completely.

Mitigation 2 - Try increasing the transaction log

First check if the Managed Instance still has storage space available (customer portal or ASC). Allocate additional space if needed and possible.

Then increase the transaction log file of the affected database. The [sys.database_files query from the Investigation section] above provides you with the file name for the `ALTER DATABASE` statement - it usually is "log" but could have been customized by the customer:

```
-- specify a SIZE that is larger than the existing size:
ALTER DATABASE [<database name>] MODIFY FILE (NAME = N'log', SIZE = 200MB )
GO
```

Let the CDC capture job continue its work until it has caught up with its pending commands. Once it has moved forward or completed, the regular log backups will truncate the transaction log.

When the situation has cleared, the customer can run `DBCC SHRINKFILE (2, <targetsize>)` to release the unneeded log file space to the Managed Instance storage.

Mitigation 3 - Involve the Product Group

If the steps above haven't helped: Open an IcM to get further assistance from the PG.

In a previous case, the PG managed to increase the log size through a mitigation step at the backend. The customer also stopped the CDC jobs and the other maintenance job on the database. It took a bit to resolve the 9002 error (likely until log backup ran) but it finally caught up.

Mitigation 4 - Throttle the workload or scale up

If the customer can control the applications, they might consider throttling the workload until the CDC capture job has caught up with the backlog. Another option is to scale up to Business Critical or one of the Premium SLOs to provide for better I/O and CPU performance on the database.

Mitigation 5 - CDC job is disabled

One of 2 options

- start the capture job so the transaction log is processed and truncated - sys.sp_cdc_add_job.
- Disable CDC. This would result in losing the change data already captured by CDC and change data currently in the log, but would also allow transaction log truncation - Enable and Disable change data capture - SQL Server

Mitigation 6 - Last option - Reset CDC by executing sp_repldone

This is the very last resort if nothing else has helped. Use the sp_repldone stored procedure to mark the "REPLICATION" log records being distributed:

```
-- Emergency command to mark all log records as distributed
EXEC sp_repldone @xactid = NULL, @xact_seqno = NULL, @numtrans = 0, @time = 0, @reset = 1
```

With this command, you are throwing away all pending CDC changes, causing data loss and gaps in the target database. You need to re-sync the source and target databases afterwards, which usually is a manual process. If the same database is used for both CDC and Transactional Replication, you also need to reinitialize all Transactional Replication subscriptions with a new snapshot. CDC and replication share the same mechanism for harvesting changes from the transaction log, and the `sp_repldone` command will reset both.

After marking those records as distributed, wait for the transaction log backup to run (usually runs every 5 to 10 minutes) and to truncate the log records.

Mitigation 7 - CDC on restored database

Try to disable CDC and re-enable if needed.

If you query EXEC sys.sp_cdc_disable_db on the Database, and it throws an error saying that the log is full, increase log size

```
use master; ALTER DATABASE [<logical_database_id>] MODIFY FILE ( NAME = N'<file_name>', SIZE = 2147483MB)
```

Then retry to disable (and eventually enable) again CDC.

Further recommendations for avoiding this type of issue

These do not resolve the immediate issue but are worth considering for the future:

- Reduce the CDC retention from its default of 3 days to a smaller value
- Run the cleanup job more often
- Exclude the "_CT" tables from the index maintenance job to avoid contention
- Reduce the number of CDC-enabled tables, if possible, to reduce the overall workload of the capture job

If it`s related with a workload issue, the customer can do one of the following:

- Increase the maximum number of transactions to process in each cycle (max_trans). The default value is 500, we recommend trying with a few thousand. Documentation
- Keep the number of Virtual log files (VLFs) as small as possible. About VLFs Transaction Log Architecture and Management Guide
- Try to persist more data together so you would have less transactions.
- Reduce the number of objects tracked by a CDC, if possible.
- Pause the CDC jobs during peak demand time, as we recommend is general CDC documentation https://docs.microsoft.com/en-us/sql/relational-databases/track-changes/about-change-data-capture-sql-server?view=sql-server-ver16#agent-jobs Customers are also advised to test the system for the high load, and monitor the latency and the log usage, as advised in the documentation: https://learn.microsoft.com/en-us/sql/relational-databases/track-changes/administer-and-monitor-change-data-capture-sql-server?view=sql-server-ver16#Monitor=

Internal reference

IcM 351618388

359712549

374620807

How good have you found this content?