Shrink failed due to backups

Last updated by | Radhika Shah | Sep 26, 2022 at 2:46 PM PDT

Contents

- Scenario
- Investigation/Analysis
  - Shrink taking too long
  - Shrink Failure
- Mitigation
- Internal Reference
- Public Doc Reference

Scenario

Customers report that their shrink failed or got interrupted.

Investigation/Analysis

Shrink taking too long

Shrink can take a long time since it's a single threaded operation which is very IO intensive. One way to work through shrinking in such scenarios is to run incremental shrinks.

Additionally, shrink can be slower than usual if the customer has BLOB columns like nvarchar(max) since they involve table scans. If you're looking for a failure message, you can try to shrink by 5GB and see why it failed.

Giving shrink more resources will speed things up as well but this should be used only for critical cases.

Shrink Failure

Verify the shrink potential of the database:

```
let shrinkBuffer = 1.01;
MonDmIoVirtualFileStats
| where originalEventTimestamp between ({StartDateTime}..{EndDateTime})
| where LogicalServerName == '{ServerName}'
| where db_name contains "{DBGuid}"
| where is_primary_replica == 1
| where type_desc == 'ROWS' // logs _typically_ do not need to be shrunk
| project ClusterName, file_id, AppName , NodeName , LogicalServerName, db_name , type_desc, originalEventTime
| summarize arg_max(originalEventTimestamp, *) by NodeName, LogicalServerName, db_name, type_desc, file_id
| extend
    free_percentage = round(iif(spaceused_gb == 0, 0.0 , 100 - spaceused_gb / (size_on_disk_Gb / 100)),2)
    , shrinkPotentialGb = size_on_disk_Gb - spaceused_gb
| join kind= leftouter (
   MonAnalyticsDBSnapshot
    | where TIMESTAMP between ({StartDateTime}..{EndDateTime})
    | where logical_database_id contains "{DBGuid}"
    | where  LogicalServerName == '{ServerName}'
    | project-rename db_name = logical_database_name
    | summarize arg_max(TIMESTAMP, service_level_objective, physical_database_id) by tenant_ring_name, sql_inst
) on LogicalServerName, db_name, logical_database_id
| project-away  db_name1, logical_database_id1
| extend isWorthShrinking = (spaceused_gb * shrinkBuffer + 20 < size_on_disk_Gb)
| order by shrinkPotentialGb desc
```

Check for errors on Shrink attempt:

```
MonSqlShrinkInfo
| where TIMESTAMP between ({StartDateTime}..{EndDateTime})
| where LogicalServerName == '{ServerName}'
| where logical_database_name == '{DatabaseName}'
| project TIMESTAMP, LogicalServerName, logical_database_name, database_id, file_id, sessionName, tag, status,
```

Sample output:

TIMESTAMP	LogicalServerName	logical_database_name	database_id	file_id	sessionName	tag	status	failure_reason
2022-05-28 23:28:34.6437836	[redacted]	[redacted]	6	2	sqlservr_shrink_file_mds	shrink_failure_reasons		data backup sync failure
2022-05-28 23:38:58.0901992	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_started on spid 229	
2022-05-28 23:38:58.0901992	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_foreground	
2022-05-28 23:38:58.0901992	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_pvs_cleanup_started	
2022-05-28 23:38:58.0901992	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_pvs_cleanup_completed	
2022-05-28 23:38:58.0901992	[redacted]	[redacted]	6	2	sqlservr_shrink_file_mds	shrink_failure_reasons		data backup sync failure
2022-05-29 01:26:16.9020564	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_started on spid 443	
2022-05-29 01:26:16.9020564	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_foreground	
2022-05-29 01:26:16.9020564	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_pvs_cleanup_started	
2022-05-29 01:26:16.9020564	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_pvs_cleanup_completed	
2022-05-29 01:26:16.9020564	[redacted]	[redacted]	6	2	sqlservr_shrink_file_mds	shrink_failure_reasons		log backup sync failure
2022-05-29 01:28:23.8726313	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_started on spid 350	
2022-05-29 01:28:23.8726313	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_foreground	
2022-05-29 01:28:23.8726313	[redacted]	[redacted]	6	0	sqlservr_shrink_file_mds	shrink_status	shrink_pvs_cleanup_started	

TIMESTAMP	tag	failure_reason
2022-05-28 23:28:34.6437836	shrink_failure_reasons	data backup sync failure
2022-05-28 23:38:58.0901992	shrink_failure_reasons	data backup sync failure
2022-05-29 01:26:16.9020564	shrink_failure_reasons	data backup sync failure

In this scenario, the Shrink failed due to conflicting backup operation happening on the database at the time. This is by design as outlined in this Shrink Database document.

```
You cannot shrink a database while the database is being backed up. Conversely, you cannot backup a database w
```

Both Azure SQL Database and Azure SQL Managed Instance use SQL Server technology to create transaction log backups every 5 to 10 minutes. This makes it more likely for the Shrink to have collision with auto-backups. The same restriction also applies to user-initated backups.

Mitigation

To overcome or workaround the issue, we recommend customers to attempt the shrink between the two T-log triggers and have retry logic wrapped around the Shrink command. A sample script is outlined here: Retry logic for Shrink.

Internal Reference

ICM 311049459

ICM 309975746

Public Doc Reference

Shrink Database Limitations and Restrictions

Implement retry logic for transient errors during shrink

How good have you found this content?