Objectstore Lock Manager

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST

Contents

Issue

Lock manager object store (OBJECTSTORE_LOCK_MANAGER) is a type of SQL server cache that allocates locks and lock owners. It also caches unused lock objects and lock owner objects for improving the performance. For each Azure SQL Database instance, there is a soft global cache cap with respect to the overall SQL instance memory. Within these boundaries, we allow the lock manager object store to grow as large as 60% of the overall SQL instance memory.

The customer workload drives the growth of the lock manager object store. If the customer's queries require a large amount of lock objects and lock owner objects, it can cause the lock manager object store to grow up to 60% of the overall SQL instance memory, taking up the largest part of the SQL server cache and thus limiting the size of the other cache types.

When the overall cache memory usage reaches the global cache cap, all caches receive pressure notifications and trigger shrinking algorithms. This will cause the caches to shrink, including the buffer pool and the plan cache. This can then cause various issues with the customer databases like:

- Queries performing significantly slower, using more CPU and data I/O due to the memory pressure.
- The CPU will possibly spike to 100%, thus also impacting the connectivity and instance health.
- The performance degradation then causing blocking with high wait times and high number of locks.
- Errors related to locks, mainly error 1204 "The instance of the SQL Server Database Engine cannot obtain a LOCK resource at this time. Rerun your statement when there are fewer active users."
- Unavailability of the database, reporting error 40613 with state 127 and 129, outage reason given as "LockManagerOOM".
- The portal's Resource Health blade reporting transient login errors and short unavailability.
- Delays or failures when trying to scale the databases.
- Out-of-Memory (OOM) errors.

Investigation / Analysis

The OBJECTSTORE_LOCK_MANAGER memory clerk is not related to memory grants, rather it is expected to grow when queries claim many locks. This can happen because of complex queries, a large amount of locked data, large transactions, and/or disabled lock escalation.

ASC - Performance and memory

Always first check ASC to confirm that the customer is affected by a large lock manager object store.

Run the ASC troubleshooter and look at the Performance insights tab for any warnings, as well as the Memory section for high values related to OBJECTSTORE_LOCK_MANAGER. Also check the Connectivity page for any login outages.

If the ASC results remain inconclusive, then run the following DMV and Kusto queries to get further details.

DMV - SQL memory clerks and lock blocks

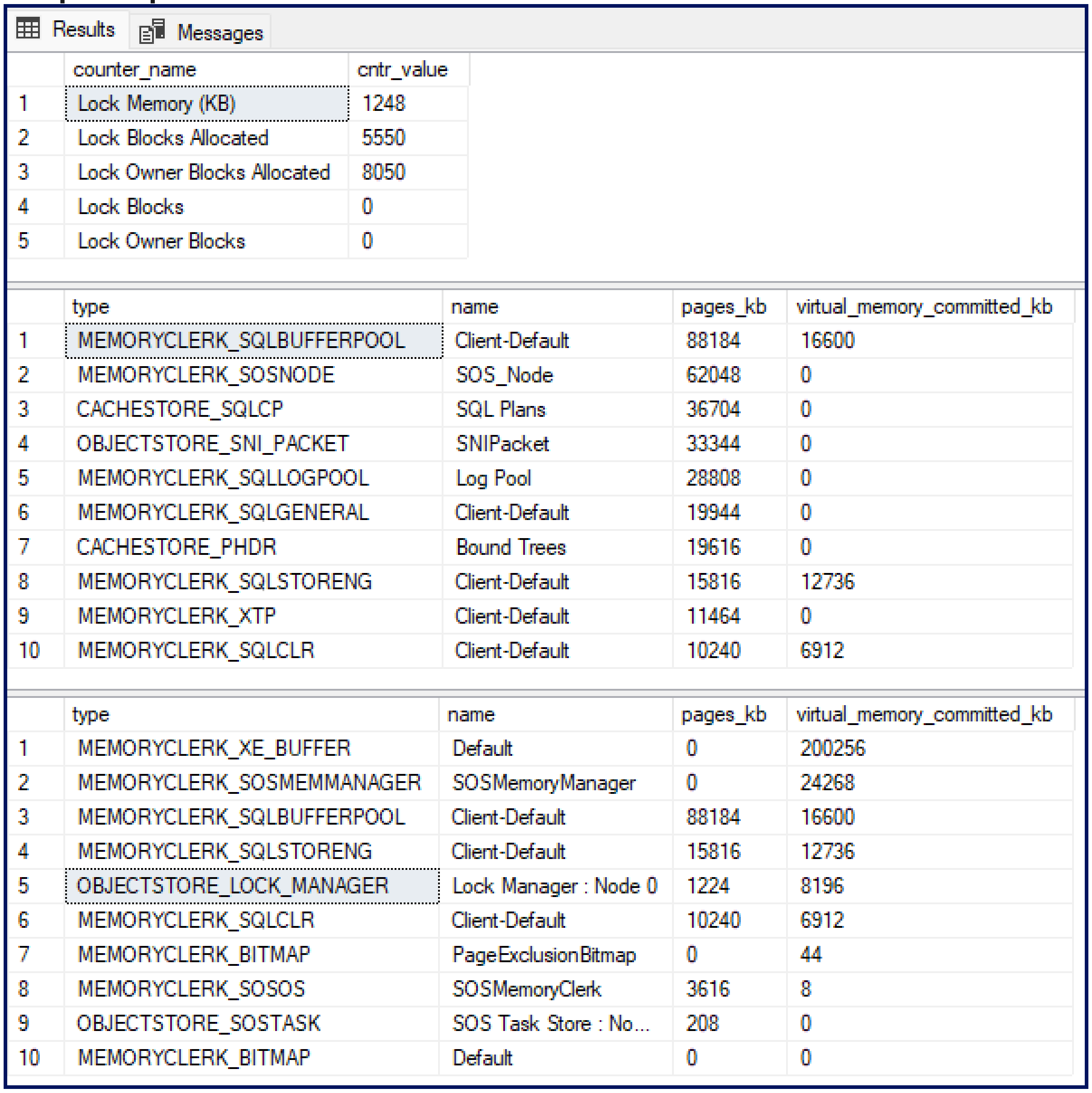
Run these DMVs to see how many lock blocks are allocated and which the largest memory clerks are.
Note that the sample output shows a normal, non-issue sitation, as I wasn't able to reproduce the issue on a test database.

```
-- total number of locks and lock owners, in-use locks and lock owners
SELECT counter_name, cntr_value
FROM sys.dm_os_performance_counters
WHERE RTRIM(object_name) like '%Memory Manager'
AND counter_name like 'Lock %';
/*
Lock Blocks Allocated - The current number of allocated lock blocks.
Lock Owner Blocks Allocated - The current number of allocated lock owner blocks.
Lock Blocks - The current number of lock blocks that are in use on the server. Refreshed periodically.
Lock Owner Blocks - The number of lock owner blocks that are currently in use on the server. Refreshed periodi
*/

-- Memory Clerks
SELECT TOP 10
    [type], [name],
    SUM(pages_kb) AS pages_kb, SUM(virtual_memory_committed_kb) AS virtual_memory_committed_kb
FROM sys.dm_os_memory_clerks
WHERE memory_node_id <> 64 -- ignore Dedicated Admin Connection (DAC) node
GROUP BY [type], [name]
ORDER BY SUM(pages_kb) DESC;

SELECT TOP 10
    [type], [name],
    SUM(pages_kb) AS pages_kb, SUM(virtual_memory_committed_kb) AS virtual_memory_committed_kb
FROM sys.dm_os_memory_clerks
WHERE memory_node_id <> 64 -- ignore Dedicated Admin Connection (DAC) node
GROUP BY [type], [name]
ORDER BY SUM(virtual_memory_committed_kb) DESC;
```

Sample output:



	counter_name	cntr_value
1	Lock Memory (KB)	1248
2	Lock Blocks Allocated	5550
3	Lock Owner Blocks Allocated	8050
4	Lock Blocks	0
5	Lock Owner Blocks	0

	type	name	pages_kb	virtual_memory_committed_kb
1	MEMORYCLERK_SQLBUFFERPOOL	Client-Default	88184	16600
2	MEMORYCLERK_SOSNODE	SOS_Node	62048	0
3	CACHESTORE_SQLCP	SQL Plans	36704	0
4	OBJECTSTORE_SNI_PACKET	SNIPacket	33344	0
5	MEMORYCLERK_SQLLOGPOOL	Log Pool	28808	0
6	MEMORYCLERK_SQLGENERAL	Client-Default	19944	0
7	CACHESTORE_PHDR	Bound Trees	19616	0
8	MEMORYCLERK_SQLSTORENG	Client-Default	15816	12736
9	MEMORYCLERK_XTP	Client-Default	11464	0
10	MEMORYCLERK_SQLCLR	Client-Default	10240	6912

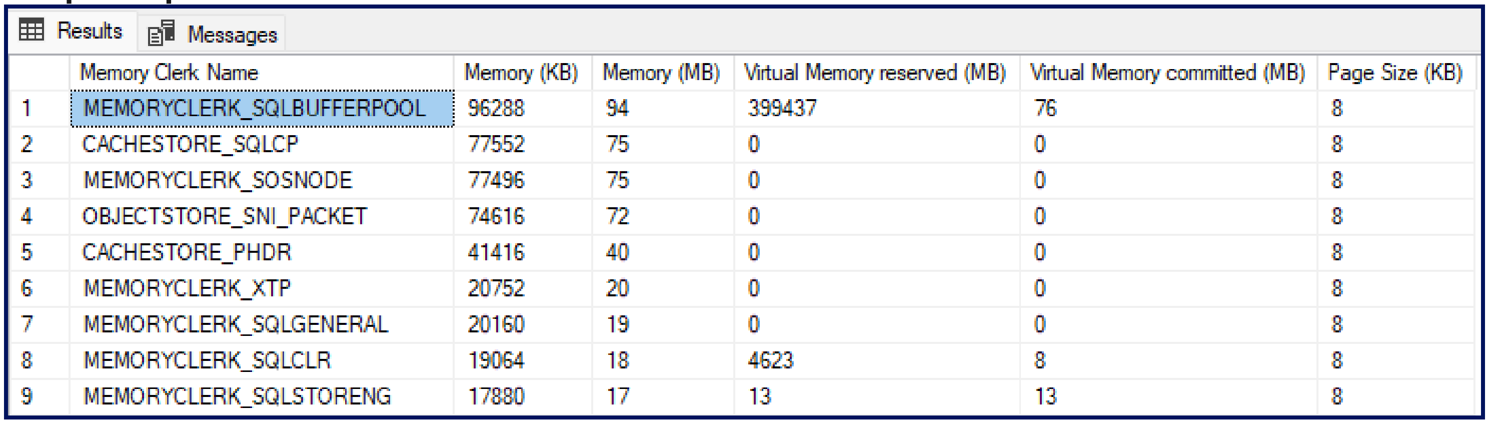
	type	name	pages_kb	virtual_memory_committed_kb
1	MEMORYCLERK_XE_BUFFER	Default	0	200256
2	MEMORYCLERK_SOSMEMMANAGER	SOSMemoryManager	0	24268
3	MEMORYCLERK_SQLBUFFERPOOL	Client-Default	88184	16600
4	MEMORYCLERK_SQLSTORENG	Client-Default	15816	12736
5	OBJECTSTORE_LOCK_MANAGER	Lock Manager : Node 0	1224	8196
6	MEMORYCLERK_SQLCLR	Client-Default	10240	6912
7	MEMORYCLERK_BITMAP	PageExclusionBitmap	0	44
8	MEMORYCLERK_SOSOS	SOSMemoryClerk	3616	8
9	OBJECTSTORE_SOSTASK	SOS Task Store : No...	208	0
10	MEMORYCLERK_BITMAP	Default	0	0

See the list of memory clerk types at sys.dm_os_memory_clerks (Transact-SQL) for more information.

Here is a variation of the `sys.dm_os_memory_clerks` query returning additional memory details:

```
-- Top clerks ordered by memory used
SELECT TOP(20)
    [type] as [Memory Clerk Name],
    SUM(pages_kb) AS [Memory (KB)],
    SUM(pages_kb) / 1024 AS [Memory (MB)],
    SUM(virtual_memory_reserved_kb) / 1024 AS [Virtual Memory reserved (MB)],
    SUM(virtual_memory_committed_kb) / 1024 AS [Virtual Memory committed (MB)],
    MIN(page_size_in_bytes) / 1024 AS [Page Size (KB)]
FROM sys.dm_os_memory_clerks
GROUP BY [type]
ORDER BY SUM(pages_kb) DESC;
```

Sample output:



Results Messages

	Memory Clerk Name	Memory (KB)	Memory (MB)	Virtual Memory reserved (MB)	Virtual Memory committed (MB)	Page Size (KB)
1	MEMORYCLERK_SQLBUFFERPOOL	96288	94	399437	76	8
2	CACHESTORE_SQLCP	77552	75	0	0	8
3	MEMORYCLERK_SOSNODE	77496	75	0	0	8
4	OBJECTSTORE_SNI_PACKET	74616	72	0	0	8
5	CACHESTORE_PHDR	41416	40	0	0	8
6	MEMORYCLERK_XTP	20752	20	0	0	8
7	MEMORYCLERK_SQLGENERAL	20160	19	0	0	8
8	MEMORYCLERK_SQLCLR	19064	18	4623	8	8
9	MEMORYCLERK_SQLSTORENG	17880	17	13	13	8

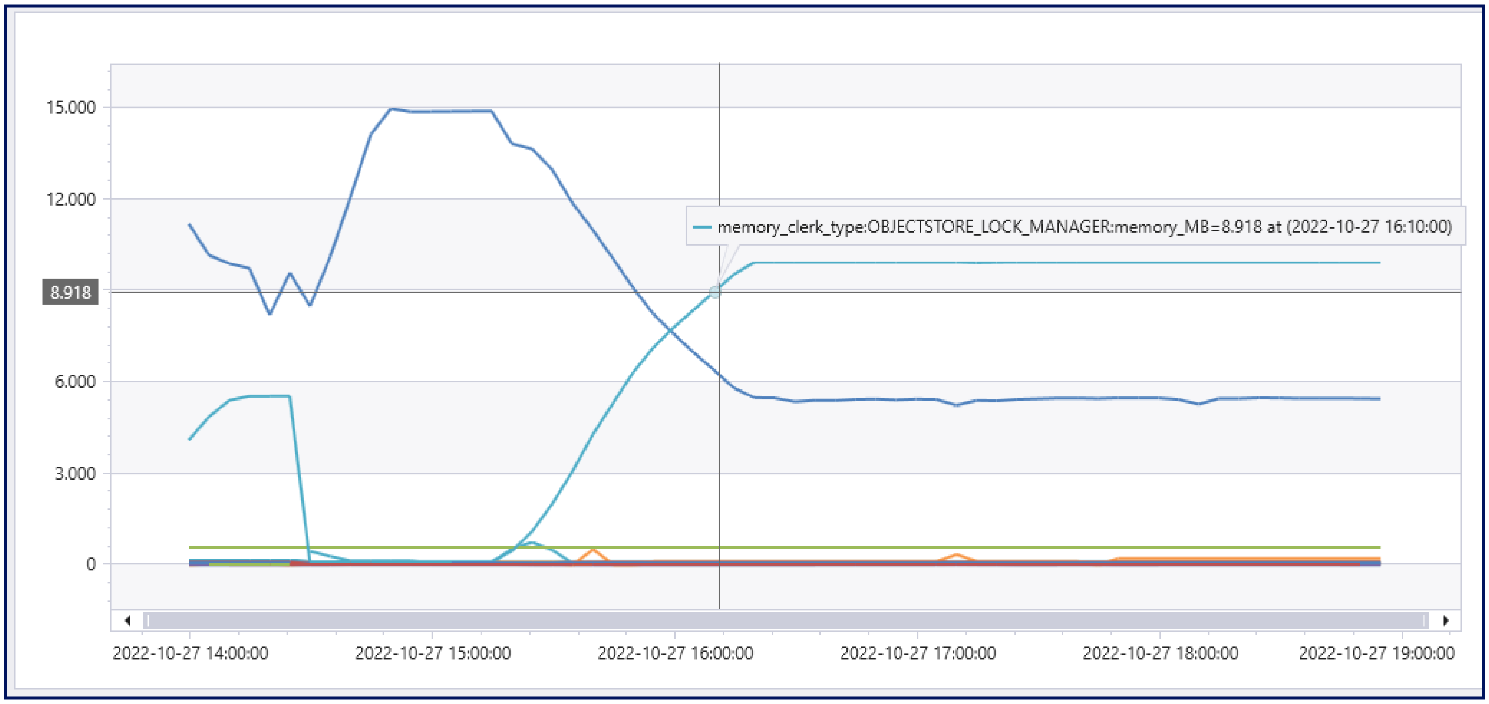
Kusto telemetry

Check the `MonSqlMemoryClerkStats` table to see the memory clerks metrics. The following query returns the same values as the "sys.dm_os_memory_clerks" DMV as above:

```
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-19 07:00:00Z);
let endTime = datetime(2022-10-19 23:00:00Z);
let timeRange = ago(7d);
MonDmDbHadrReplicaStates
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
// | where  TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where logical_database_name  =~ db
| where is_primary_replica == 1
| where AppName notcontains "b-"
| summarize min(TIMESTAMP) by AppName, NodeName
| join kind=inner
(
    MonSqlMemoryClerkStats
    | where  TIMESTAMP >= startTime
    | where  TIMESTAMP <= endTime
    // | where  TIMESTAMP >= timeRange
    | where LogicalServerName =~ srv
    | extend memory_MB = round(pages_kb/1024.0)
    | extend vm_committed_MB = round(vm_committed_kb/1024.0)
    | project TIMESTAMP, NodeName, LogicalServerName, AppName, memory_clerk_type, memory_clerk_name, memory_MB
) on AppName, NodeName
| summarize memory_MB = sum(memory_MB) + sum(vm_committed_MB) by ['memory_clerk_type'], bin(TIMESTAMP, 5min)
| render timechart
```

Sample output:

Note how the blue SQLBUFFERPOOL curve drops when the OBJECTSTORE_LOCK_MANAGER curve starts to increase



Check the `LoginOutages` table for any unplanned `LockManagerOOM` -related issues:

```
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-25 07:00:00Z);
let endTime = datetime(2022-11-19 23:00:00Z);
LoginOutages
| where outageStartTime >= startTime
| where outageEndTime <= endTime
| where logical_server_name =~ srv
| where database_name =~ db
| top 20 by TIMESTAMP desc
| project outageStartTime, outageEndTime, durationSeconds, database_name, database_type, service_level_objecti
```

Sample output:
This is from the same server/database as for the MonSqlMemoryClerkStats output above

outageStartTime	outageEndTime	durationSeconds	database_name	database_type	servic
2022-10-27 19:10:01.8387688	2022-10-27 19:10:03.7647485	1,93	databasename	SQL DB	SQLD
2022-10-27 18:15:05.9412073	2022-10-27 18:15:05.9842073	0,04	databasename	SQL DB	SQLD
2022-10-27 17:12:40.5396748	2022-10-27 17:12:54.8271609	14,29	databasename	SQL DB	SQLD
2022-10-27 14:27:50.8035523	2022-10-27 14:28:02.0985119	11,29	databasename	SQL DB	SQLD
2022-10-25 18:15:02.3160952	2022-10-25 18:15:02.6652742	0,35	databasename	SQL DB	SQLD
2022-10-25 18:20:02.2975570	2022-10-25 18:20:14.6801236	12,38	databasename	SQL DB	SQLD

Cross-check the findings with any login-related error messages:

```
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-24 00:00:00Z);
let endTime = datetime(2022-10-29 23:15:00Z);
MonLogin
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
| where logical_server_name =~ srv
| where database_name =~ db
| where event =~ "process_login_finish"
| where (is_success == false and is_user_error == false) //or total_time_ms > 4000
| summarize cnt(), min(TIMESTAMP), max(TIMESTAMP) by package, error, ['state']
| sort by count_ desc
```

Sample output:
This is again from the same server/database as for the MonSqlMemoryClerkStats output above

package	error	state	count_	min_TIMESTAMP	max_TIMESTAMP
sqlserver	40613	127	113	2022-10-24 10:30:54.6124160	2022-10-27 19:10:13.4179544
sqlserver	40613	128	11	2022-10-25 13:54:50.6509864	2022-10-27 19:15:13.5017545
sqlserver	40613	129	14	2022-10-24 10:30:54.6281083	2022-10-27 18:15:11.0790871

Check `MonDmDbHadrReplicaStates` for any errors related to geo-/HADR-replication. It should come back healthy and show as "SYNCHRONIZED", as the OBJECTSTORE_LOCK_MANAGER issue does not affect node replication.

```
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-27 14:00:00Z);
let endTime = datetime(2022-10-27 19:00:00Z);
let timeRange = ago(7d);
MonDmDbHadrReplicaStates
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
// | where  TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where logical_database_name  =~ db
| where is_primary_replica == 1
| where AppName notcontains "b-"
//| where AppName == "d088e491d7ad"
| project TIMESTAMP, NodeName, is_primary_replica, synchronization_state_desc, synchronization_health_desc, da
| order by TIMESTAMP asc
| summarize min(TIMESTAMP), max(TIMESTAMP) by is_primary_replica, synchronization_state_desc, synchronization_
// should always be healthy, "SYNCHRONIZED"
```

Sample output:
This is again from the same server/database as for the MonSqlMemoryClerkStats output above

is_primary_replica	synchronization_state_desc	synchronization_health_desc	database_state_desc
1	SYNCHRONIZED	HEALTHY	ONLINE
1	SYNCHRONIZED	HEALTHY	ONLINE

Check `MonSQLSystemHealth` for information from the SQL Server errorlog. It should have some errors related to the out-of-memory condition.

```
// ERRORLOG details
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-24 00:00:00Z);
let endTime = datetime(2022-10-27 17:30:00Z);
let timeRange = ago(7d);
MonDmDbHadrReplicaStates
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
// | where  TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where logical_database_name  =~ db
| where is_primary_replica == 1
| where AppName notcontains "b-"
| distinct AppName
| join kind=inner  (
    MonSQLSystemHealth
    | where  TIMESTAMP >= startTime
    | where  TIMESTAMP <= endTime
    //| where  TIMESTAMP >= timeRange
    //| where error_id > 0
    | where message !contains "backup"
    | where message !contains("HaDr")
    | where message !contains("FSTR")
    | where message !contains("Zeroing")
    | where message !contains("Skipped as DB is not encrypted with AKV key")
    | where message !contains("All access check passed")
    | where message !contains("Starting CheckDbAKVAccess")
    | where message !contains("Backup(")
    | where message !contains("[INFO] [CKPT] HkHostDbCtxt::RegisterHkTruncationLsnChange()")
    | where message !contains("[User-provided format string filtered]")
    | where message !contains("Waiting for report fault signal")
    | where message !contains("[Filtered Args] Log was backed up")
    | where message !contains("UpdateHadronTruncationLsn")
    | where message !contains("Skipped as DEK does not exist")
    | where message !contains("Skipped as DB is not encrypted")
    | where message !contains("[VersionCleaner]")
    | where message !contains("Removing xdes id")
    | where message !contains("AutoShrinkLog: Trying to shrink log file")
    | where message !contains("[INFO] HkHostFreezeCkptTrimming()")
    | where message !contains("[INFO] createBackupContextV2()")
    | where message !contains("[INFO] HkHostBackupDeleteContext()")
    | where message != "%ls"
    | project TIMESTAMP, NodeName, AppName, error_id, message
) on AppName
| project error_id, message
```

Typical error messages to look out for:

> Error_ID=-1, "NotifyLogPoolMgr in DB: 5 shrinks logpool under memory pressure for 5613501 times with decrease percentage: 15. /"
> Error_ID=1204, "Error: 1204, Severity: 19, State: 4. The instance of the SQL Server Database Engine cannot obtain a LOCK resource at this time. Rerun your statement when there are fewer active users. Ask the database administrator to check the lock and memory configuration for this instance, or to check for long-running transactions."
> Error_ID=40613, "Error: 40613, Severity: 17, State: 127/128/129. [Filtered Args] Database '%1' on server '%2' is not currently available. Please retry the connection later."

Mitigation

Once the lock manager object store has grown large, it can't easily shrink again. The shrink algorithm, in each iteration to shrink object stores, has a relative fixed number of entries to consider, often less than 80. Because the size of lock/lock owner objects is small and other resource pools have larger entries, the shrink operation often shrinks other caches first. Hence, shrinking lock manager object stores can be very slow. Also, as long as the workload related to the OBJECTSTORE_LOCK_MANAGER growth is still active, shrinking the lock manager object store might not be able to proceed; it will rather continue to grow and maintain the pressure on the other memory clerks.

There are several options that the customer can take to reduce the impact:

1. Confirm the lock manager object store size through the queries above. Then, once not many locks are in use, attempt to clean up the lock manager object store cache and to shrink its size by running the following DBCC command: `DBCC FREESYSTEMCACHE ('Lock Manager : Node 0');`
2. Make sure that lock escalation is enabled on the tables where data is being updated. Use `select name, type_desc, lock_escalation_desc from sys.tables` to see the configured lock escalation mode, and change it with `ALTER TABLE ... SET (LOCK_ESCALATION = AUTO);` if any of the related tables are set to "disabled".
3. Avoid running concurrent DML queries (Insert/Update/Delete) when a large DML is executing. This allows for lock escalation to table/HoBT level and avoids taking fine-grained page and row locks.
4. Use the `TABLOCK` hint when executing a large DML to make sure it takes a coarse-grained lock at the table/HoBT level instead of page/row locks.
5. Break large DMLs into smaller batches and transactions to commit more frequently and release locks earlier.
6. Investigate blocking and performance in general; the head blocker could be a using a non-optimal execution plan, thus requesting more locks than a better plan would.

Internal References

- IcM 334303836
- IcM 219842452
- IcM 208239499

How good have you found this content?