Memory - Monitor Memory usage

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

Contents

Scripts to monitor SQL Server memory usage in the Buffer Pool and Plan Cache, and to identify Query Memory grants and waits by session

This is a "How To" article related to memory management and the SQL Server buffer pool.

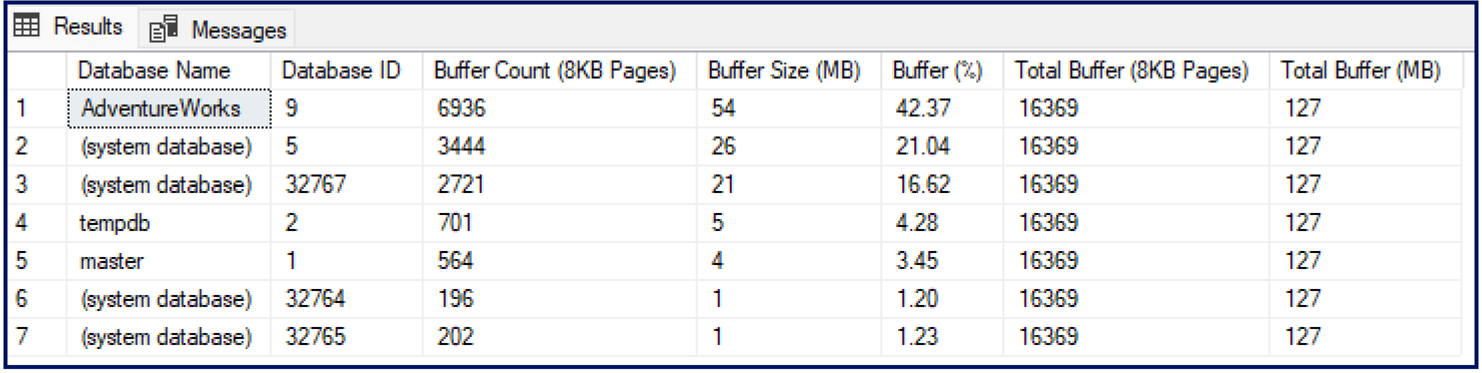
The scripts below are useful for troubleshooting out-of-memory (OOM) and other buffer pool issues. They don't cover memory tracking outside of the buffer pool.

Buffer Pool usage per database

This query provides you with a list of databases at this logical SQL server as seen from the context of the current Azure SQL Database and its SQL instance. The list contains the current database and the system databases including tempdb, but no other stand-alone databases that are hosted at the same logical SQL server. The query shows the amount of buffer pool memory each of the databases uses, and the overall percentage that is in relation to the total buffer pool.

```
WITH BufCount AS
(
    SELECT database_id, db_buffer_pages = COUNT_BIG(*)
    FROM sys.dm_os_buffer_descriptors
    WHERE database_id in (1, 2) or database_id >= 5
    GROUP BY database_id
),
TotalBuffer AS
(
    SELECT  cntr_value as totalbuffer
    FROM sys.dm_os_performance_counters
    WHERE RTRIM([object_name]) LIKE '%Buffer Manager'
    AND counter_name = 'Database pages'
)
SELECT
    [Database Name] = ISNULL(DB_NAME([database_id]), '(system database)'),
    [Database ID] = database_id,
    [Buffer Count (8KB Pages)] = db_buffer_pages,
    [Buffer Size (MB)] = db_buffer_pages / 128,
    [Buffer (%)] = CONVERT(DECIMAL(10,2), db_buffer_pages * 100.0 / tb.totalbuffer),
    [Total Buffer (8KB Pages)] = tb.totalbuffer,
    [Total Buffer (MB)] = tb.totalbuffer / 128
FROM BufCount, TotalBuffer tb
ORDER BY [Buffer Size (MB)] DESC;
```

Sample output:



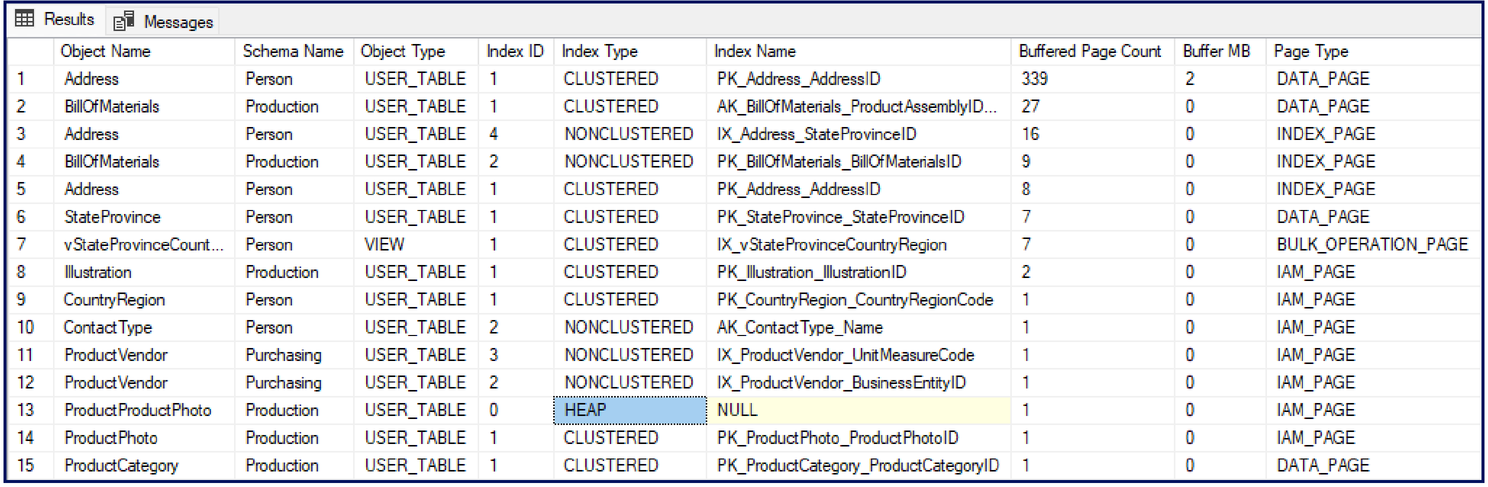
	Database Name	Database ID	Buffer Count (8KB Pages)	Buffer Size (MB)	Buffer (%)	Total Buffer (8KB Pages)	Total Buffer (MB)
1	AdventureWorks	9	6936	54	42.37	16369	127
2	(system database)	5	3444	26	21.04	16369	127
3	(system database)	32767	2721	21	16.62	16369	127
4	tempdb	2	701	5	4.28	16369	127
5	master	1	564	4	3.45	16369	127
6	(system database)	32764	196	1	1.20	16369	127
7	(system database)	32765	202	1	1.23	16369	127

Buffer Pool usage per object in the database

This query returns the Top 20 biggest buffer pool-consuming objejcts in your Azure SQL Database.

```
SELECT TOP 20
    object_name(sysobj.object_id) AS [Object Name],
    s.name as [Schema Name],
    sysobj.type_desc AS [Object Type],
    i.index_id AS [Index ID],
    i.[type_desc] AS [Index Type],
    i.[name] AS [Index Name],
    COUNT_BIG(*) AS [Buffered Page Count],
    COUNT_BIG(*) * 8192 / (1024 * 1024) AS [Buffer MB],
    bd.page_type AS [Page Type] -- ,obj.name ,obj.index_id, i.[name]
FROM sys.dm_os_buffer_descriptors AS bd
INNER JOIN sys.allocation_units au ON bd.allocation_unit_id = au.allocation_unit_id
INNER JOIN sys.partitions p ON au.container_id = p.hobt_id
LEFT JOIN sys.indexes i ON i.object_id = p.object_id AND i.index_id = p.index_id
LEFT JOIN sys.objects sysobj on i.object_id = sysobj.object_id
LEFT OUTER JOIN sys.schemas s ON sysobj.schema_id = s.schema_id
WHERE database_id = DB_ID() and sysobj.type not in ('S','IT')
GROUP BY object_name(sysobj.object_id), s.name, i.index_id, i.[name], i.[type_desc], bd.page_type, sysobj.type
ORDER BY [Buffered Page Count] DESC
```

Sample output:



	Object Name	Schema Name	Object Type	Index ID	Index Type	Index Name	Buffered Page Count	Buffer MB	Page Type
1	Address	Person	USER_TABLE	1	CLUSTERED	PK_Address_AddressID	339	2	DATA_PAGE
2	BillOfMaterials	Production	USER_TABLE	1	CLUSTERED	AK_BillOfMaterials_ProductAssemblyID...	27	0	DATA_PAGE
3	Address	Person	USER_TABLE	4	NONCLUSTERED	IX_Address_StateProvinceID	16	0	INDEX_PAGE
4	BillOfMaterials	Production	USER_TABLE	2	NONCLUSTERED	PK_BillOfMaterials_BillOfMaterialsID	9	0	INDEX_PAGE
5	Address	Person	USER_TABLE	1	CLUSTERED	PK_Address_AddressID	8	0	INDEX_PAGE
6	StateProvince	Person	USER_TABLE	1	CLUSTERED	PK_StateProvince_StateProvinceID	7	0	DATA_PAGE
7	vStateProvinceCount...	Person	VIEW	1	CLUSTERED	IX_vStateProvinceCountryRegion	7	0	BULK_OPERATION_PAGE
8	Illustration	Production	USER_TABLE	1	CLUSTERED	PK_Illustration_IllustrationID	2	0	IAM_PAGE
9	CountryRegion	Person	USER_TABLE	1	CLUSTERED	PK_CountryRegion_CountryRegionCode	1	0	IAM_PAGE
10	ContactType	Person	USER_TABLE	2	NONCLUSTERED	AK_ContactType_Name	1	0	IAM_PAGE
11	ProductVendor	Purchasing	USER_TABLE	3	NONCLUSTERED	IX_ProductVendor_UnitMeasureCode	1	0	IAM_PAGE
12	ProductVendor	Purchasing	USER_TABLE	2	NONCLUSTERED	IX_ProductVendor_BusinessEntityID	1	0	IAM_PAGE
13	ProductProductPhoto	Production	USER_TABLE	0	HEAP	NULL	1	0	IAM_PAGE
14	ProductPhoto	Production	USER_TABLE	1	CLUSTERED	PK_ProductPhoto_ProductPhotoID	1	0	IAM_PAGE
15	ProductCategory	Production	USER_TABLE	1	CLUSTERED	PK_ProductCategory_ProductCategoryID	1	0	DATA_PAGE

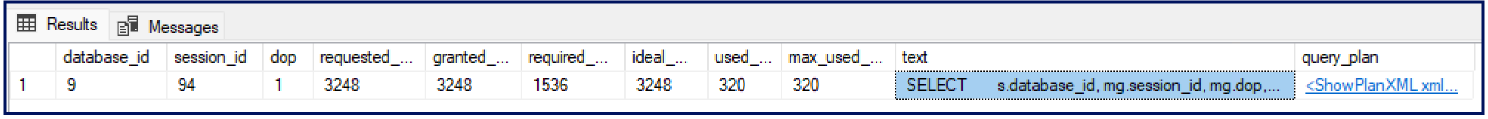
Current memory requests, grants and execution plan for each active session

This query over sys.dm_exec_query_memory_grants returns information about all queries that have requested and are waiting for a memory grant or have been given a memory grant. Queries that do not require a memory grant will not appear in this view. For example, sort and hash join operations have memory grants for query execution, while queries without an `ORDER BY` clause will not have a memory grant.

When you run this from SSMS, you can click on the query_plan to open the graphical execution plan.

```sql
-- This can be used in a script to capture information over a period of time.
SELECT
    s.database_id, mg.session_id, mg.dop,
    mg.requested_memory_kb, mg.granted_memory_kb, mg.required_memory_kb,
    mg.ideal_memory_kb, mg.used_memory_kb, mg.max_used_memory_kb,
    t.text, qp.query_plan
FROM sys.dm_exec_query_memory_grants AS mg
INNER JOIN sys.dm_exec_sessions s on s.session_id = mg.session_id
CROSS APPLY sys.dm_exec_sql_text(mg.sql_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(mg.plan_handle) AS qp
ORDER BY mg.requested_memory_kb DESC OPTION (MAXDOP 1)
```

Sample output:



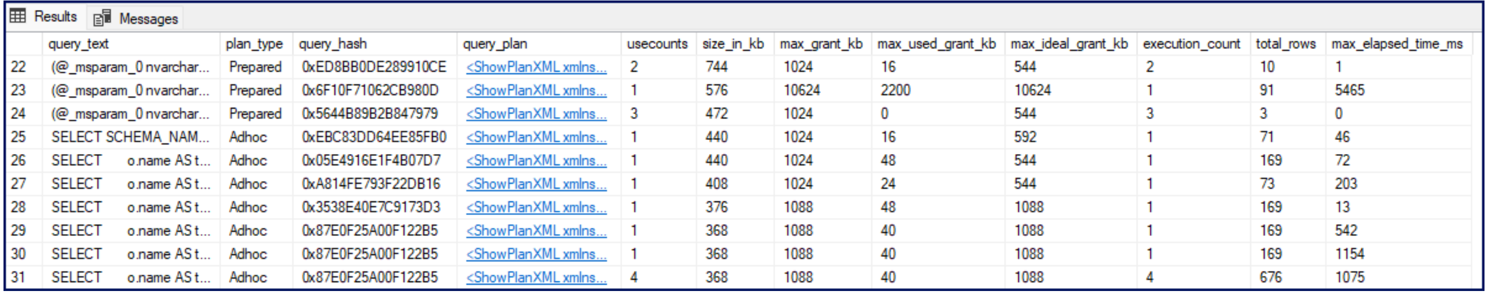
	database_id	session_id	dop	requested_...	granted_...	required_...	ideal_...	used_...	max_used_...	text	query_plan
1	9	94	1	3248	3248	1536	3248	320	320	SELECT s.database_id, mg.session_id, mg.dop,...	<ShowPlanXML xml...

Search plan cache for queries with memory grants completed

This query returns the Top 50 largest cached query plans in your Azure SQL Database. It shows the cached query text, its cached query plans, the amount of memory taken by cached plans, and the reuse count of the cached plans. See the "Public Doc Reference" section at the bottom for explanations about the "MemoryFractions" condition in the Where clause.

```sql
SELECT top 50
    t.text AS 'query_text', cp.objtype AS 'plan_type',
    qs.query_hash, qp.query_plan, cp.usecounts, cp.size_in_bytes / 1024 AS 'size_in_kb',
    qs.max_grant_kb, qs.max_used_grant_kb, qs.max_ideal_grant_kb,
    qs.execution_count, qs.total_rows, qs.max_elapsed_time / 1000 AS 'max_elapsed_time_ms'
FROM sys.dm_exec_cached_plans cp
INNER JOIN sys.dm_exec_query_stats qs ON cp.plan_handle = qs.plan_handle
CROSS APPLY sys.dm_exec_query_plan(cp.plan_handle) qp
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) t
WHERE qp.query_plan.exist('declare namespace n="http://schemas.microsoft.com/sqlserver/2004/07/showplan"; //n:
ORDER BY cp.size_in_bytes desc
OPTION (MAXDOP 1)
```

Sample output:



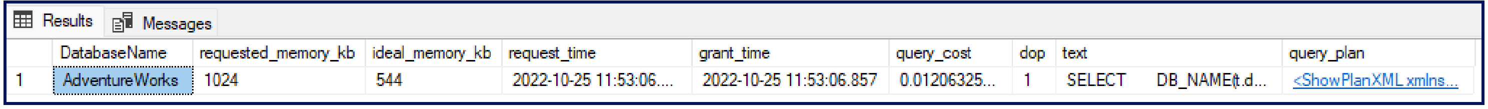
	query_text	plan_type	query_hash	query_plan	usecounts	size_in_kb	max_grant_kb	max_used_grant_kb	max_ideal_grant_kb	execution_count	total_rows	max_elapsed_time_ms
22	(@_msparam_0 nvarchar...	Prepared	0xED8BB0DE289910CE	<ShowPlanXML xmlns...	2	744	1024	16	544	2	10	1
23	(@_msparam_0 nvarchar...	Prepared	0x6F10F71062CB980D	<ShowPlanXML xmlns...	1	576	10624	2200	10624	1	91	5465
24	(@_msparam_0 nvarchar...	Prepared	0x5644B89B2B847979	<ShowPlanXML xmlns...	3	472	1024	0	544	3	3	0
25	SELECT SCHEMA_NAM...	Adhoc	0xEBC83DD64EE85FB0	<ShowPlanXML xmlns...	1	440	1024	16	592	1	71	46
26	SELECT o.name AS t...	Adhoc	0x05E4916E1F4B07D7	<ShowPlanXML xmlns...	1	440	1024	48	544	1	169	72
27	SELECT o.name AS t...	Adhoc	0xA814FE793F22DB16	<ShowPlanXML xmlns...	1	408	1024	24	544	1	73	203
28	SELECT o.name AS t...	Adhoc	0x3538E40E7C9173D3	<ShowPlanXML xmlns...	1	376	1088	48	1088	1	169	13
29	SELECT o.name AS t...	Adhoc	0x87E0F25A00F122B5	<ShowPlanXML xmlns...	1	368	1088	40	1088	1	169	542
30	SELECT o.name AS t...	Adhoc	0x87E0F25A00F122B5	<ShowPlanXML xmlns...	1	368	1088	40	1088	1	169	1154
31	SELECT o.name AS t...	Adhoc	0x87E0F25A00F122B5	<ShowPlanXML xmlns...	4	368	1088	40	1088	4	676	1075

Queries that have requested memory or are waiting for memory grants

This query can be used when there are active sessions waiting on memory to be granted. These sessions will have a wait_type of RESOURCE_SEMAPHORE. You can calculate the wait time for the memory grant by subtracting request_time and grant_time.

```
SELECT
    DB_NAME(t.dbid) AS [DatabaseName] ,
    mg.requested_memory_kb ,
    mg.ideal_memory_kb ,
    mg.request_time ,
    mg.grant_time ,
    mg.query_cost ,
    mg.dop ,
    t.[text], qp.query_plan
FROM sys.dm_exec_query_memory_grants AS mg
CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(mg.plan_handle) AS qp
WHERE mg.request_time < COALESCE(grant_time, '99991231')
ORDER BY mg.requested_memory_kb DESC;
```

Sample output:



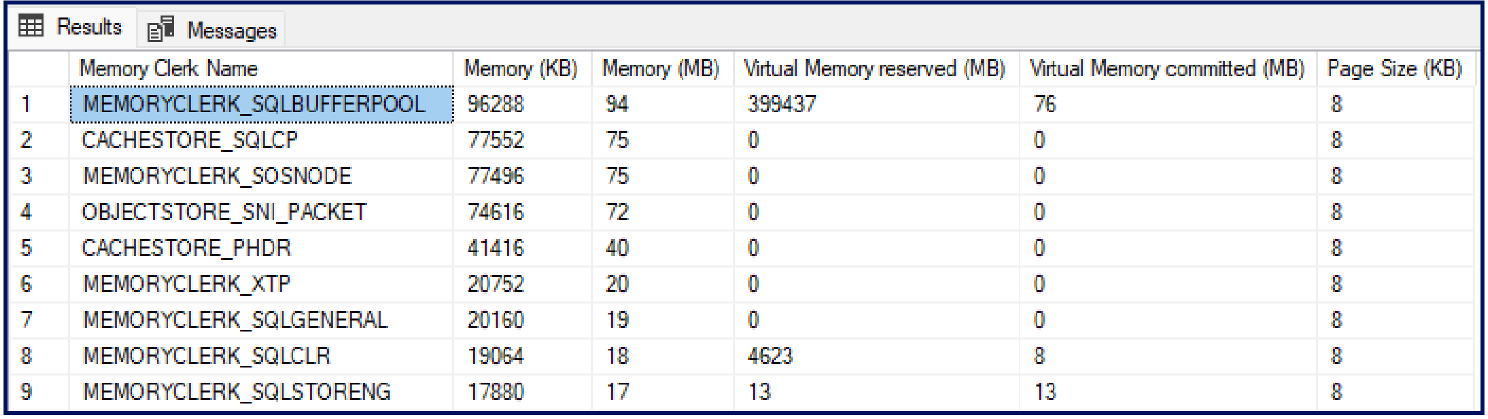
	DatabaseName	requested_memory_kb	ideal_memory_kb	request_time	grant_time	query_cost	dop	text	query_plan
1	AdventureWorks	1024	544	2022-10-25 11:53:06....	2022-10-25 11:53:06.857	0.01206325...	1	SELECT DB_NAME(t.d...	<ShowPlanXML xmlns...

Top memory clerks ordered by memory used

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

```
-- 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:



	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

Public Doc Reference

- [Understanding SQL server memory grant (2010)](#)
- [Mystery of memory fraction in Showplan XML (2010)](#)

- Old 3rd-party SQL on-premise blog article (2012).

How good have you found this content?

🙂 🙁