OOM Workflow

Last updated by | Charlene Wang | Apr 10, 2022 at 11:22 PM PDT

Contents

OOM Workflow

Issue summary

This TSG, that walks through troubleshooting when,

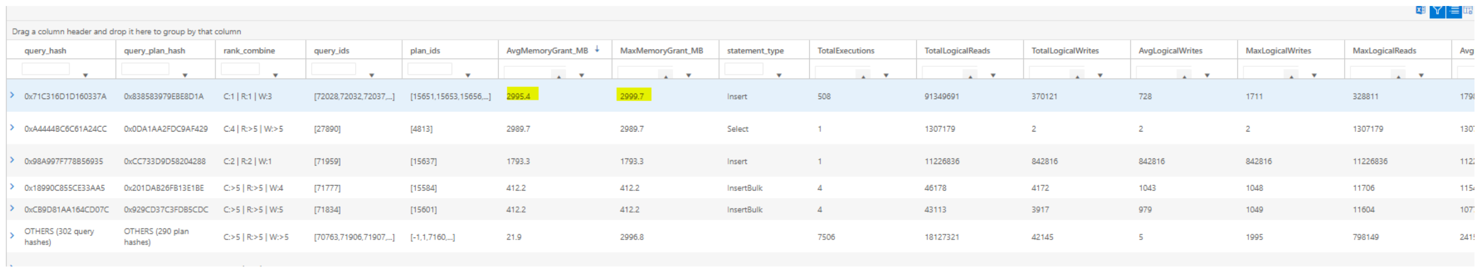
Customer reports the following errors indicating OOM in SQL DB

- Error code **701** with error message "There is insufficient system memory in resource pool '%ls' to run this query."
- Error code **802** with error message "There is insufficient memory available in the buffer pool."

Troubleshooting through ASC

If you are sure that this is a OOM issue, to identify the queries causing OOM issues you can look for memory_grants. Go to:

ASC>>SQLTroubleshooter>>Performance>>Queries>>Top 5 Queries For Each Cpu Time, Logical Reads and Logical Writes.
Look at the column's AvgMemoryGrantMB and MaxMemoryGrantMB (scroll towards right if you are not seeing them,refer below).



Drag a column header and drop it here to group by that column

query_hash	query_plan_hash	rank_combine	query_ids	plan_ids	AvgMemoryGrant_MB	MaxMemoryGrant_MB	statement_type	TotalExecutions	TotalLogicalReads	TotalLogicalWrites	AvgLogicalWrites	MaxLogicalWrites	MaxLogicalReads	Avg
0x71C316D1D160337A	0x838583979EBE8D1A	C:1 | R:1 | W:3	[72028,72032,72037,...]	[15651,15653,15656,...]	2995.4	2999.7	Insert	508	91349691	370121	728	1711	328811	179
0xA4444BC6C61A24CC	0x0DA1AA2FDC9AF429	C:4 | R:>5 | W:>5	[27890]	[4813]	2989.7	2989.7	Select	1	1307179	2	2	2	1307179	130
0x98A997F778B56935	0xCC733D9D58204288	C:2 | R:2 | W:1	[71959]	[15637]	1793.3	1793.3	Insert	1	11226836	842816	842816	842816	11226836	112
0x18990C855CE33AA5	0x201DAB26FB13E1BE	C:>5 | R:>5 | W:4	[71777]	[15584]	412.2	412.2	InsertBulk	4	46178	4172	1043	1048	11706	115
0xCB9D81AA164CD07C	0x929CD37C3FDB5CDC	C:>5 | R:>5 | W:5	[71834]	[15601]	412.2	412.2	InsertBulk	4	43113	3917	979	1049	11604	107
OTHERS (302 query hashes)	OTHERS (290 plan hashes)	C:>5 | R:>5 | W:>5	[70763,71906,71907,...]	[-1,1,7160,...]	21.9	2996.8		7506	18127321	42145	5	1995	798149	241

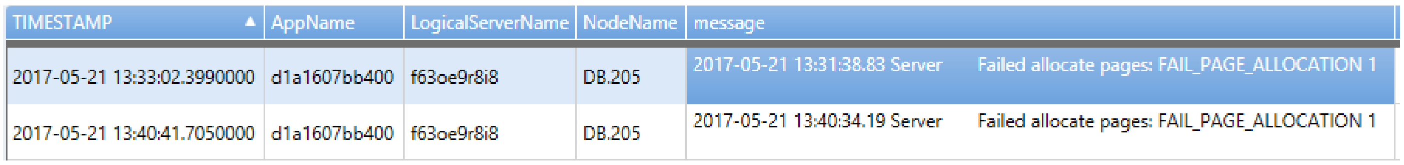
Mitigation steps

1. **Confirm OOM occurred by running the following two queries**

Query 1

```
MonSQLSystemHealth
| where TIMESTAMP > datetime(2017-05-20 01:00:00) and TIMESTAMP < datetime(2017-05-22 01:00:00)
| where AppName == "a36f774b88da"
| where message contains "Failed allocate pages: FAIL_PAGE_ALLOCATION" or message contains "Failed Virtual
Allocate Bytes: FAIL_VIRTUAL_RESERVE" or message contains "Failed to reserve pages:
FAIL_PAGE_RESERVATION" or message contains "FAIL_VIRTUAL_COMMIT"
| project TIMESTAMP, AppName, LogicalServerName, NodeName, message
```

Table 1

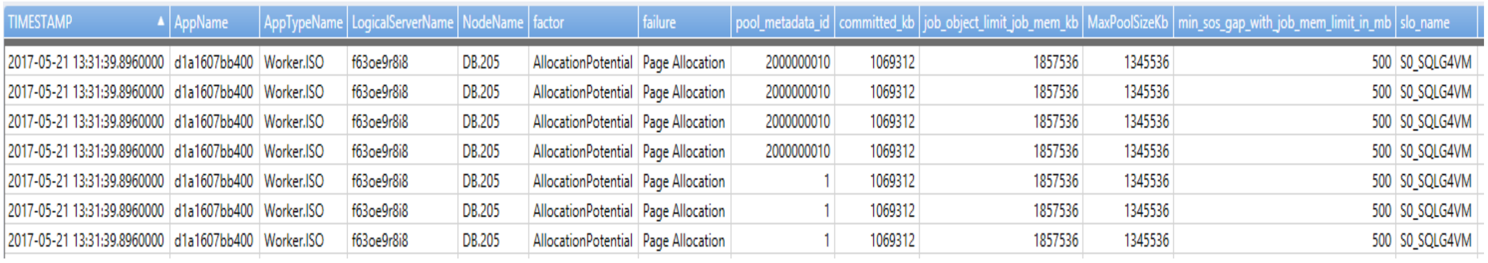


TIMESTAMP	AppName	LogicalServerName	NodeName	message
2017-05-21 13:33:02.3990000	d1a1607bb400	f63oe9r8i8	DB.205	2017-05-21 13:31:38.83 Server Failed allocate pages: FAIL_PAGE_ALLOCATION 1
2017-05-21 13:40:41.7050000	d1a1607bb400	f63oe9r8i8	DB.205	2017-05-21 13:40:34.19 Server Failed allocate pages: FAIL_PAGE_ALLOCATION 1

Query 2

```
let UserAppName="d1a1607bb400";
MonSqlMemNodeOomRingBuffer
| where TIMESTAMP > datetime(2017-05-20 01:00:00) and TIMESTAMP < datetime(2017-05-22 01:00:00)
| where AppName == UserAppName
| project TIMESTAMP, ClusterName, AppName, AppTypeName, LogicalServerName, NodeName, factor, failure,
pool_metadata_id, committed_kb, job_object_limit_job_mem_kb, is_system_physical_memory_low,
is_process_physical_memory_low, is_process_virtual_memory_low
| join kind= inner (
   MonDmDbResourceGovernance
   | where AppName == UserAppName
   | summarize by ClusterName, AppTypeName, AppName, LogicalServerName, NodeName, slo_name,
min_sos_gap_with_job_mem_limit_in_mb
) on ClusterName, AppTypeName, AppName, LogicalServerName, NodeName
| extend MaxPoolSizeKb=job_object_limit_job_mem_kb-min_sos_gap_with_job_mem_limit_in_mb*1024
| project TIMESTAMP, AppName, AppTypeName, LogicalServerName, NodeName, factor, failure,
pool_metadata_id, committed_kb, job_object_limit_job_mem_kb, MaxPoolSizeKb,
min_sos_gap_with_job_mem_limit_in_mb, slo_name
```

Table 2



TIMESTAMP	AppName	AppTypeName	LogicalServerName	NodeName	factor	failure	pool_metadata_id	committed_kb	job_object_limit_job_mem_kb	MaxPoolSizeKb	min_sos_gap_with_job_mem_limit_in_mb	slo_name
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	2000000010	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	2000000010	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	2000000010	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	2000000010	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	1	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	1	1069312	1857536	1345536	500	SO_SQLG4VM
2017-05-21 13:31:39.8960000	d1a1607bb400	Worker.ISO	f63oe9r8i8	DB.205	AllocationPotential	Page Allocation	1	1069312	1857536	1345536	500	SO_SQLG4VM

If this query does not return any rows, it implies nothing, it might just be missing telemetry. If you do get rows returned, they are referred to below.

1. **Is non-SOS memory usage too high?**

From the result (table 2) of query 2 above, if the committed_kb (SOS memory usage) is much less than (i.e. ~50%) MaxPoolSizeKb, then it is likely that non-SOS memory usage was too high. The non-SOS memory usage includes thread-stacks memory usage, non-SQL components (fabric etc.) memory usage etc. When non-SOS memory increases, the max pool memory limits for user pool will be adjusted to lower. Run the following query to decide if the non-SOS memory usage is too high, i.e. greater than the 50% of NT Job memory limit during the incident time period. If it is true, take a DUMP before failover. We will need to debug the dump to figure out the root cause so far.

Query 3

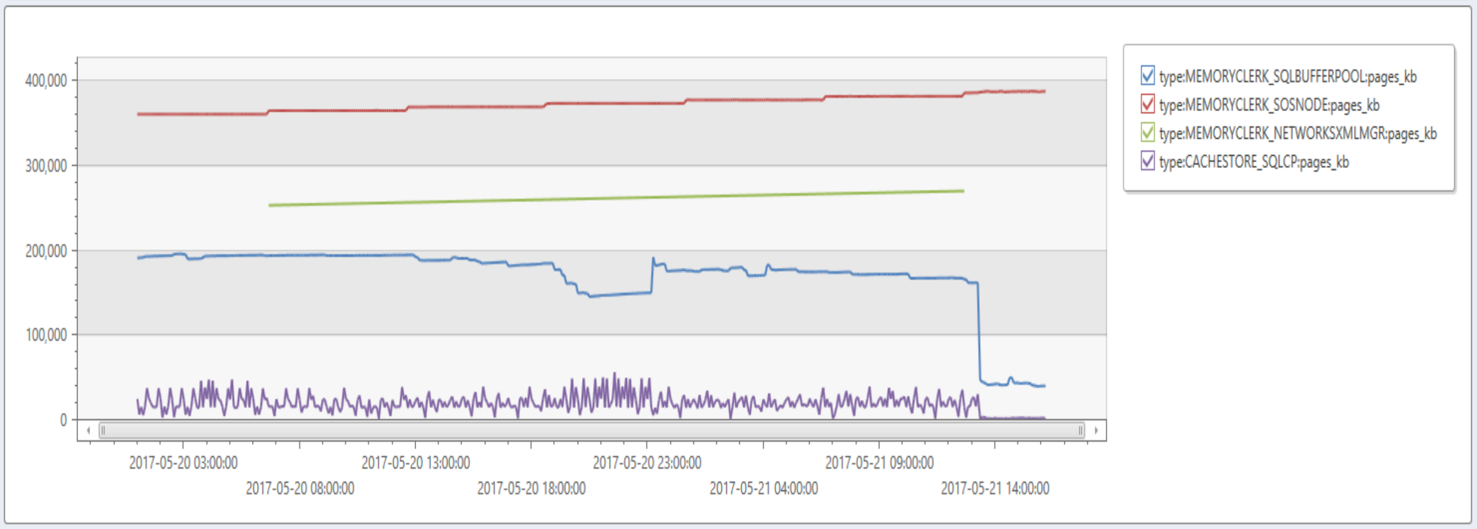
```
let startTime=datetime(2017-05-20 01:00:00);
let endTime=datetime(2017-05-22 01:00:00);
let TargetAppName="d1a1607bb400";
let userApppMemoryLoadTable = MonRgLoad
| where TIMESTAMP > startTime and TIMESTAMP < endTime
| where application_type == "Worker.ISO.Premium" or application_type == "Worker.ISO"
| where event == "instance_load" and code_package_name == "Code"
| where application_name contains TargetAppName
| parse kind = regex application_name with "fabric:/Worker.ISO.*/" UserAppName
| project TIMESTAMP, memory_load_cap , ClusterName , NodeName, UserAppName, application_type , slo_size
| summarize avg(memory_load_cap) by ClusterName, NodeName, UserAppName,
UserAppTypeName=application_type, slo_size, bin(TIMESTAMP, 30m);
let systemPoolLimitTable = MonGovernorResourcePools
| where TIMESTAMP > startTime or TIMESTAMP < endTime
| where AppTypeName == "Worker.ISO.Premium" or AppTypeName == "Worker.ISO"
| where name == "internal"
| where AppName == TargetAppName
| summarize PoolMaxMemoryMb=avg(target_memory_kb)/1024 by ClusterName, NodeName,
UserAppTypeName=AppTypeName, UserAppName=AppName, bin(TIMESTAMP, 30m), LogicalServerName;
let nonSosMemoryUsageTable = systemPoolLimitTable
| join kind= inner
(
   userApppMemoryLoadTable
)
on ClusterName, NodeName, UserAppTypeName, UserAppName
| extend NonSosMemoryMb=avg_memory_load_cap-PoolMaxMemoryMb
| where NonSosMemoryMb > avg_memory_load_cap*0.50
| project-away ClusterName1, NodeName1, UserAppTypeName1, UserAppName1, TIMESTAMP1;
nonSosMemoryUsageTable
```

1. **Who was the top memory consumer at the time?**

Run the following query to find out who was the top memory clerks consumers at the time. For the following example, MEMORYCLERK_SOSNODE and MEMORYCLERK_NETWORKSXMLMGR were the top consumers.

Query 4

```
let oomApp="c877cf83f860";
let oomNode="DB.14";
let startTime=datetime(2019-01-07 17:00);
let endTime=datetime(2019-01-09 17:00);
let topConsumers=MonSqlMemoryClerkStats
| where TIMESTAMP > startTime and TIMESTAMP < endTime
| where AppName == oomApp and NodeName == oomNode
| where pages_kb > 50000 or vm_committed_kb > 50000
| summarize by ['memory_clerk_type'];
MonSqlMemoryClerkStats
| where TIMESTAMP > startTime and TIMESTAMP < endTime
| where AppName == oomApp and NodeName == oomNode
| join kind= inner (
  topConsumers
) on ['memory_clerk_type']
| extend type_per_node = strcat(['memory_clerk_type'], tostring(memory_node_id))
| project TIMESTAMP, type_per_node, pages_vm_committed_kb=pages_kb+vm_committed_kb
| render timechart
```



- If top memory clerk is MEMORYCLERK_SQLQERESERVATIONS, indicating some queries required a large amount of memory grants. You can use below query to find out top 5 queries with large memory grant.

```
MonWiQdsExecStats
| where TIMESTAMP > datetime(2021-05-26 22:09:38.0994729)-15m and TIMESTAMP < datetime(2021-05-26 22
| where AppName == "Appname"
| summarize sum(execution_count), sum(cpu_time), sum(max_query_memory_pages), sum(max_logical_writes
| project  query_hash, plan_id, sum_execution_count, sum_max_query_memory_pages
| order by sum_max_query_memory_pages desc
| take 5
```

If you need further help from PG, please raise an ICM to SQL DB Perf : Query Processing.

- If top memory clerk is CACHESTORE_OBJCP, then transfer to SQL DB Perf: Frontend.
- If top memory clerk is CACHESTORE_SQLCP, then transfer to SQL DB Perf: Frontend.

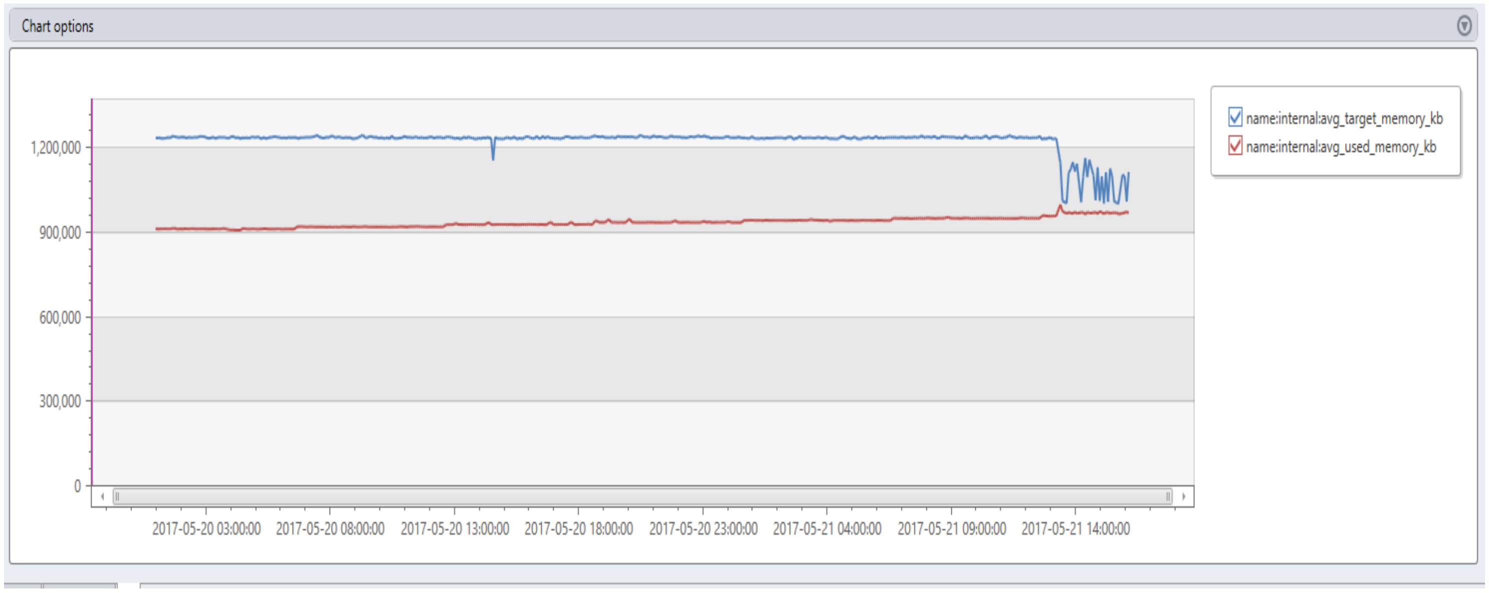
1. **Which resource pools consumed most of memory at the time?**

Run the following query to find out the resource pools that consumed most of memory. Note that internal is the system pool. If a pool name contains "SloPool" or "SloSharedPool", then it is the user pool.

- If the user pool is the top consumer and its usage is matched with top memory clerk found above, then it is clearly that the user requests caused high memory usage.Once we conclude that user queries consumed large amount memory and that caused OOM, we could suggest customer to upgrade SLO to higher edition for more memory resource. Meanwhile we can also ask QE expert to help customer to identify queries that consumed too much memory.
- Otherwise, there might potentially some bugs caused the OOMs. For known issues, please transfer to component team queue.
- Otherwise, take a dump before doing mitigations. We will need to debug dump.

Query 5

```
let oomApp="d1a1607bb400";
let oomNode="DB.205";
let startTime=datetime(2017-05-20 01:00:00);
let endTime=datetime(2017-05-22 01:00:00);
let topPoolConsumers=MonGovernorResourcePools
| where TIMESTAMP > startTime and TIMESTAMP < endTime
| where AppName == oomApp and NodeName == oomNode
| where used_memory_kb > 100000
| summarize by name;
MonGovernorResourcePools
| where TIMESTAMP > startTime and TIMESTAMP < endTime
| where AppName == oomApp and NodeName == oomNode
| join kind= inner (
  topPoolConsumers
) on name
| summarize avg(target_memory_kb), avg(used_memory_kb) by name, bin(TIMESTAMP, 5m)
| render timechart
```



Symptom

- Customer hit error code 41805 or 701
  - **701**: There is insufficient system memory in resource pool '%ls' to run this query.
  - **41805**: There is insufficient memory in the resource pool '%ls' to run this operation on memory-optimized tables. See 'http://go.microsoft.com/fwlink/?LinkID=614951' for more information.
- XEvent "**xtp_db_page_allocation_disallowed**" was fired.
- ERRORLOG contains:
  - XTP failed page allocation due to memory pressure: FAIL_PAGE_ALLOCATION
  - Memory usage tracing for all clerks, brokers, XTP consumers etc. See this attached file as an example:
    - <<errorlog_dump_on_oom.txt>>

Classification

Root cause path -

Workload performance/User-issue/error/Throttling errors/resource limit

How good have you found this content?