Performance Troubleshooting Azure SQL (PaaS)

Naga Sai Krishna Pamidikondala
Cloud Solution Architecture | Microsoft | Data & AI

February 20, 2020

Problem Statement

- Azure SQL Database provides tools and methods you can use to monitor usage easily, add or remove resources (such as CPU, memory, or I/O), troubleshoot potential problems, and make recommendations to improve the performance of a database.
- Automatic tuning enables a database to adapt to the workload and automatically optimize performance.
- However, some custom issues might need troubleshooting. In addition, Azure SQL does not provide Server level access to run SQL Profiler trace tool, Perfmon, Windows Event logs etc.
- This article provides some best practices and some tools can be used to troubleshoot performance problems.

Scope of the Document

This document has various Azure SQL dynamic management views (DMVs) that can be used for monitoring and troubleshooting performance problems in Azure SQL (PaaS) version of database.

Requirements / Dependencies

Running the DMV scripts provided in the document requires connectivity of Azure SQL PaaS database with SQL Server Management Studio.

Troubleshoot performance problems

To ensure that a database runs without problems, you should:

Monitor database performance to make sure that the resources assigned to the database can handle the workload. If the database is hitting resource limits, consider:

•Identifying and optimizing the top resource-consuming queries.

•Adding more resources by upgrading the service tier.

Troubleshoot performance problems to identify why a potential problem occurred and to identify the root cause of the problem. After you identify the root cause, take steps to fix the problem.

To diagnose and resolve performance problems, begin by finding out the state of each active query and the conditions that cause performance problems relevant to each workload state. To improve Azure SQL Database performance, you need to understand that each active query request from the application is in either a running state or a waiting state. As you troubleshoot a performance problem in Azure SQL Database, keep the following diagram in mind.

Running-related problems might be caused by:

· **Compilation problems**: SQL Query Optimizer might produce a suboptimal plan because of stale statistics, an incorrect estimate of the number of rows to be processed, or an inaccurate estimate of required memory. If you know the query was executed faster in the past or on another instance (either a managed instance or a SQL Server instance), compare the actual execution plans to see if they are different. Try to apply query hints or rebuild statistics or indexes to get the better plan.

Recommendation

Cognizant Cloud Services Team recommends enabling the automatic plan correction in Azure SQL Database for mitigating these problems automatically.

· **Execution problems**: If the query plan is optimal, it is probably hitting the database's resource limits, such as log write throughput. Or it might be using fragmented indexes that should be rebuilt. Execution problems can also happen when a large number of concurrent queries need the same resources. *Waiting-related* problems are usually related to execution problems, because the queries that don't execute efficiently are probably waiting for some resources.

Waiting-related problems might be caused by:

· **Blocking**: One query might hold the lock on objects in the database while others try to access the same objects. You can identify blocking queries by using DMVs or monitoring tools.

· **IO problems**: Queries might be waiting for the pages to be written to the data or log files. In this case, check
the INSTANCE_LOG_RATE_GOVERNOR, WRITE_LOG,
or PAGEIOLATCH_* wait statistics in the DMV.

· **TempDB problems**: If the workload uses temporary tables or there are TempDB spills in the plans, the queries might have a problem with TempDB throughput.

· **Memory-related problems**: If the workload does not have enough memory, the page life expectancy might drop, or the queries might get less memory than they need. In some cases, built-in intelligence in Query Optimizer will fix memory-related problems.

Database Administration / Configuration – DMVs

1. SELECT @@SERVERNAME AS [Server Name], @@VERSION AS [SQL Server and OS Version Info];

The above T-SQL script will provide the name, version information of Azure SQL database.

2. SELECT name, value, value_in_use, minimum, maximum, [description], is_dynamic, is_advanced

FROM sys.configurations WITH (NOLOCK)

ORDER BY name OPTION (RECOMPILE)

The above T-SQL script will provide the name, value and other configuration options and related information of Azure SQL database.

3. SELECT db.[name] AS [Database Name], db.recovery_model_desc AS [Recovery Model], db.state_desc, db.containment_desc, db.log_reuse_wait_desc AS [Log Reuse Wait Description], db.[compatibility_level] AS [DB Compatibility Level], db.is_mixed_page_allocation_on, db.page_verify_option_desc AS [Page Verify Option],

db.is_auto_create_stats_on, db.is_auto_update_stats_on, db.is_auto_update_stats_async_on, db.is_parameterization_forced, db.snapshot_isolation_state_desc, db.is_read_committed_snapshot_on, db.is_auto_close_on, db.is_auto_shrink_on, db.target_recovery_time_in_seconds, db.is_cdc_enabled, db.is_memory_optimized_elevate_to_snapshot_on, db.delayed_durability_desc, db.is_auto_create_stats_incremental_on, db.is_query_store_on, db.is_sync_with_backup, db.is_temporal_history_retention_enabled, db.is_encrypted

FROM sys.databases AS db WITH (NOLOCK)

ORDER BY db. [name] OPTION (RECOMPILE);

The above T-SQL script will provide the database specific information of Azure SQL database.

The details like database recovery model, compatibility level, isolation level, auto create statistics and auto update statistics options etc.

All the above parameters are very crucial for DB Performance, configuration etc.

4. SELECT CAST(SUM(CAST(FILEPROPERTY(name, 'SpaceUsed') AS bigint) * 8192.) / 1024 / 1024 AS DECIMAL(15,2)) AS [Database Size In MB],

CAST(SUM(CAST(FILEPROPERTY(name, 'SpaceUsed') AS bigint) * 8192.) / 1024 / 1024 / 1024 AS DECIMAL(15, 2)) AS [Database Size In GB]

FROM sys.database_files WITH (NOLOCK)

WHERE [type_desc] = N'ROWS' OPTION (RECOMPILE);

The above T-SQL script will provide the database specific information that is size of each Azure SQL database in Azure SQL Server.

5. SELECT ec.client_net_address, es.[program_name], es.[host_name], es.login_name,

COUNT(ec.session_id) AS [connection count]

FROM sys.dm_exec_sessions AS es WITH (NOLOCK)

INNER JOIN sys.dm_exec_connections AS ec WITH (NOLOCK) ON es.session_id = ec.session_id

GROUP BY ec.client_net_address, es.[program_name], es.[host_name], es.login_name

ORDER BY ec.client_net_address, es.[program_name] OPTION (RECOMPILE);

6. SELECT c.session_id, c.net_transport, c.encrypt_option, c.auth_scheme, s.host_name, s.program_name, s.client_interface_name, s.login_name, s.nt_domain, s.nt_user_name, s.original_login_name, c.connect_time, s.login_time, c.client_net_Address

FROM sys.dm_exec_connections AS c JOIN sys.dm_exec_sessions AS s

ON c.session_id = s.session_id

WHERE c.session_id = @@SPID;

The above T-SQL script will provide the details of all the connections along with program / application that's is connecting, host name, login details & total number of connection count onto Azure SQL Server database.

7. Select * from sys.firewall_rules

The above T-SQL script will provide the details of all the Server level firewalls associated with Azure SQL Database.

8. Select * from sys.database_firewall_rules

The above T-SQL script will provide the details of all the database level firewalls associated with Azure SQL Database.

9. Select *

From sys.event_log

Where database_name = '< Azure SQL Database Name>'

The above T-SQL script will display the details of all the events recorded for Azure SQL database.

Note - We shall run the query against MASTER database in Azure SQL Server Instance.

10. SELECT sys.objects.name, SUM(reserved_page_count) * 8.0 / 1024

FROM sys.dm_db_partition_stats, sys.objects

```
WHERE sys.dm_db_partition_stats.object_id = sys.objects.object_id

GROUP BY sys.objects.name;

GO
```

The above T-SQL script will display size of Individual database objects in Azure SQL Database.

Size of database objects is displayed in KB.

```
11.  SELECT  link_guid , partner_server, partner_database , last_replication, replication_state_desc, role_desc, replication_lag_sec

FROM sys.dm_geo_replication_link_status;
```

The above T-SQL script will display the Geo Replication (High Availability & Disaster Recovery)

Status, partner server & database details.

```
12.  Select *

From sys.dm_database_encryption_keys
```

The above T-SQL script returns information about the encryption state of a database and its associated database encryption keys.

```
13.  SELECT *

FROM sys.dm_operation_status

WHERE major_resource_id = '< Name - Azure SQL Database >'

ORDER BY start_time DESC;
```

The above T-SQL script returns tracking all the major changes performed for Azure SQL Database.

Database Performance Engineering – DMVs

```
SELECT avg_cpu_percent, avg_data_io_percent, avg_log_write_percent,
avg_memory_usage_percent, xtp_storage_percent,max_worker_percent,
max_session_percent, dtu_limit, avg_login_rate_percent, end_time

FROM sys.dm_db_resource_stats WITH (NOLOCK)

ORDER BY end_time DESC OPTION (RECOMPILE);
```

The above T-SQL script returns information about the latest resource utilization statistics of Azure SQL Database.

```
WITH [Waits]

AS (SELECT wait_type, wait_time_ms/ 1000.0 AS [WaitS],

        (wait_time_ms - signal_wait_time_ms) / 1000.0 AS [ResourceS],

         signal_wait_time_ms / 1000.0 AS [SignalS],

         waiting_tasks_count AS [WaitCount],

         100.0 * wait_time_ms / SUM (wait_time_ms) OVER() AS [Percentage],

        ROW_NUMBER() OVER(ORDER BY wait_time_ms DESC) AS
[RowNum]

   FROM sys.dm_db_wait_stats WITH (NOLOCK)
```

```
WHERE [wait_type] NOT IN (
    N'BROKER_EVENTHANDLER', N'BROKER_RECEIVE_WAITFOR',
N'BROKER_TASK_STOP',
        N'BROKER_TO_FLUSH', N'BROKER_TRANSMITTER',
N'CHECKPOINT_QUEUE',
    N'CHKPT', N'CLR_AUTO_EVENT', N'CLR_MANUAL_EVENT',
N'CLR_SEMAPHORE',
    N'DBMIRROR_DBM_EVENT', N'DBMIRROR_EVENTS_QUEUE',
N'DBMIRROR_WORKER_QUEUE',
        N'DBMIRRORING_CMD', N'DIRTY_PAGE_POLL',
N'DISPATCHER_QUEUE_SEMAPHORE',
    N'EXECSYNC', N'FSAGENT', N'FT_IFTS_SCHEDULER_IDLE_WAIT',
N'FT_IFTSHC_MUTEX',
    N'HADR_CLUSAPI_CALL',
N'HADR_FILESTREAM_IOMGR_IOCOMPLETION',
N'HADR_LOGCAPTURE_WAIT',
        N'HADR_NOTIFICATION_DEQUEUE',
N'HADR_TIMER_TASK', N'HADR_WORK_QUEUE',
    N'KSOURCE_WAKEUP', N'LAZYWRITER_SLEEP', N'LOGMGR_QUEUE',
        N'MEMORY_ALLOCATION_EXT',
N'ONDEMAND_TASK_QUEUE',
        N'PREEMPTIVE_HADR_LEASE_MECHANISM',
N'PREEMPTIVE_SP_SERVER_DIAGNOSTICS',
        N'PREEMPTIVE_ODBCOPS',
        N'PREEMPTIVE_OS_LIBRARYOPS',
N'PREEMPTIVE_OS_COMOPS', N'PREEMPTIVE_OS_CRYPTOPS',
```

```
                    N'PREEMPTIVE_OS_PIPEOPS',
N'PREEMPTIVE_OS_AUTHENTICATIONOPS',

                    N'PREEMPTIVE_OS_GENERICOPS',
N'PREEMPTIVE_OS_VERIFYTRUST',

                    N'PREEMPTIVE_OS_FILEOPS',
N'PREEMPTIVE_OS_DEVICEOPS', N'PREEMPTIVE_OS_QUERYREGISTRY',

                    N'PREEMPTIVE_OS_WRITEFILE',

                    N'PREEMPTIVE_XE_CALLBACKEXECUTE',
N'PREEMPTIVE_XE_DISPATCHER',

                    N'PREEMPTIVE_XE_GETTARGETSTATE',
N'PREEMPTIVE_XE_SESSIONCOMMIT',

                    N'PREEMPTIVE_XE_TARGETINIT',
N'PREEMPTIVE_XE_TARGETFINALIZE',

                    N'PREEMPTIVE_XHTTP',

      N'PWAIT_ALL_COMPONENTS_INITIALIZED',
N'PWAIT_DIRECTLOGCONSUMER_GETNEXT',

                    N'QDS_PERSIST_TASK_MAIN_LOOP_SLEEP',

                    N'QDS_ASYNC_QUEUE',

      N'QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP',
N'REQUEST_FOR_DEADLOCK_SEARCH',

                    N'RESOURCE_GOVERNOR_IDLE',

                    N'RESOURCE_QUEUE', N'SERVER_IDLE_CHECK',
N'SLEEP_BPOOL_FLUSH', N'SLEEP_DBSTARTUP',

                    N'SLEEP_DCOMSTARTUP', N'SLEEP_MASTERDBREADY',
N'SLEEP_MASTERMDREADY',
```

```
        N'SLEEP_MASTERUPGRADED', N'SLEEP_MSDBSTARTUP',
N'SLEEP_SYSTEMTASK', N'SLEEP_TASK',

        N'SLEEP_TEMPDBSTARTUP', N'SNI_HTTP_ACCEPT',
N'SP_SERVER_DIAGNOSTICS_SLEEP',

                N'SQLTRACE_BUFFER_FLUSH',
N'SQLTRACE_INCREMENTAL_FLUSH_SLEEP', N'SQLTRACE_WAIT_ENTRIES',

                N'WAIT_FOR_RESULTS', N'WAITFOR',
N'WAITFOR_TASKSHUTDOWN', N'WAIT_XTP_HOST_WAIT',

                N'WAIT_XTP_OFFLINE_CKPT_NEW_LOG',
N'WAIT_XTP_CKPT_CLOSE', N'WAIT_XTP_RECOVERY',

                N'XE_BUFFERMGR_ALLPROCESSED_EVENT',
N'XE_DISPATCHER_JOIN',

        N'XE_DISPATCHER_WAIT', N'XE_LIVE_TARGET_TVF',
N'XE_TIMER_EVENT')

    AND waiting_tasks_count > 0)

SELECT

    MAX (W1.wait_type) AS [WaitType],

            CAST (MAX (W1.Percentage) AS DECIMAL (5,2)) AS [Wait
Percentage],

            CAST ((MAX (W1.WaitS) / MAX (W1.WaitCount)) AS DECIMAL
(16,4)) AS [AvgWait_Sec],

    CAST ((MAX (W1.ResourceS) / MAX (W1.WaitCount)) AS DECIMAL (16,4))
AS [AvgRes_Sec],

    CAST ((MAX (W1.SignalS) / MAX (W1.WaitCount)) AS DECIMAL (16,4)) AS
[AvgSig_Sec],

    CAST (MAX (W1.WaitS) AS DECIMAL (16,2)) AS [Total_Wait_Sec],
```

```
  CAST (MAX (W1.ResourceS) AS DECIMAL (16,2)) AS [Resource_Sec],

  CAST (MAX (W1.SignalS) AS DECIMAL (16,2)) AS [Signal_Sec],

  MAX (W1.WaitCount) AS [Wait Count]

FROM Waits AS W1

INNER JOIN Waits AS W2

ON W2.RowNum <= W1.RowNum

GROUP BY W1.RowNum

HAVING SUM (W2.Percentage) - MAX (W1.Percentage) < 99 -- percentage
threshold

OPTION (RECOMPILE);
```

The above T-SQL script returns information about top waits for this database since last restart or failover

```
SELECT TOP(50) LEFT(t.[text], 50) AS [Short Query Text], qs.execution_count
AS [Execution Count], qs.total_logical_reads AS [Total Logical Reads],
qs.total_logical_reads/qs.execution_count AS [Avg Logical
Reads],qs.total_worker_time AS [Total Worker
Time],qs.total_worker_time/qs.execution_count AS [Avg Worker Time],
qs.total_elapsed_time AS [Total Elapsed Time],
qs.total_elapsed_time/qs.execution_count AS [Avg Elapsed Time],
qs.creation_time AS [Creation Time],t.[text] AS [Complete Query Text],
qp.query_plan AS [Query Plan]

FROM sys.dm_exec_query_stats AS qs WITH (NOLOCK) CROSS APPLY
sys.dm_exec_sql_text(plan_handle) AS t CROSS APPLY
sys.dm_exec_query_plan(plan_handle) AS qp

WHERE t.dbid = DB_ID()
```

```
ORDER BY qs.execution_count DESC OPTION (RECOMPILE);
```

The above T-SQL script returns information about the most frequently executed queries for on

The Azure SQL Database in context.

```
SELECT TOP(50) DB_NAME(t.[dbid]) AS [Database Name],
REPLACE(REPLACE(LEFT(t.[text], 50), CHAR(10),''), CHAR(13),'') AS [Short Query Text], qs.total_worker_time AS [Total Worker Time],
qs.min_worker_time AS [Min Worker Time],qs.total_worker_time/qs.execution_count AS [Avg Worker Time],
qs.max_worker_time AS [Max Worker Time], qs.min_elapsed_time AS [Min Elapsed Time], qs.total_elapsed_time/qs.execution_count AS [Avg Elapsed Time], qs.max_elapsed_time AS [Max Elapsed Time],qs.min_logical_reads AS [Min Logical Reads],qs.total_logical_reads/qs.execution_count AS [Avg Logical Reads],qs.max_logical_reads AS [Max Logical Reads],
qs.execution_count AS [Execution Count], qs.creation_time AS [Creation Time],t.[text] AS [Query Text], qp.query_plan AS [Query Plan]

FROM sys.dm_exec_query_stats AS qs WITH (NOLOCK)

CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS t

CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp

WHERE t.dbid = DB_ID()

ORDER BY qs.total_worker_time DESC OPTION (RECOMPILE);
```

The above T-SQL script returns information about the queries that makes use of most of the worker time (CPU time) in the Azure SQL Database in context. During the CPU Contention, the above query will be very handy.

```
Select *

From sys.sysprocesses

Where blocked <> 0 and SPID > 50
```

The above T-SQL script returns information about the queries that makes use of most of the worker time (CPU time) in the Azure SQL Database in context.

```
SELECT AVG (current_tasks_count) AS [Avg Task Count], AVG
(work_queue_count) AS [Avg Work Queue Count], AVG
(runnable_tasks_count) AS [Avg Runnable Task Count], AVG
(pending_disk_io_count) AS [Avg Pending DiskIO Count]

FROM sys.dm_os_schedulers WITH (NOLOCK)

WHERE scheduler_id < 255 OPTION (RECOMPILE);
```

The above T-SQL script returns information about the Tasks in Azure SQL Database.

The tasks that are running, runnable, pending for IO.

```
WITH Aggregate_IO_Statistics

AS

(SELECT DB_NAME(database_id) AS [Database Name],

CAST(SUM(num_of_bytes_read + num_of_bytes_written)/1048576 AS
DECIMAL(12, 2)) AS io_in_mb

FROM sys.dm_io_virtual_file_stats(NULL, NULL) AS [DM_IO_STATS]
```

```
WHERE database_id NOT IN (4, 5, 32767)

GROUP BY database_id)

SELECT ROW_NUMBER() OVER (ORDER BY io_in_mb DESC) AS [I/O Rank],
[Database Name],

    CAST (io_in_mb/ SUM (io_in_mb) OVER() * 100.0 AS DECIMAL(5,2)) AS
[I/O Percent],

    io_in_mb AS [Total I/O (MB)]

FROM Aggregate_IO_Statistics

ORDER BY [I/O Rank] OPTION (RECOMPILE);
```

The above T-SQL script returns information about the I/O utilization details of all Azure SQL databases present in the Azure SQL Server.

```
SELECT DB_NAME(DB_ID()) AS [Database Name], df.name AS [Logical
Name], vfs.[file_id], df.type_desc,

df.physical_name AS [Physical Name],
CAST(vfs.size_on_disk_bytes/1048576.0 AS DECIMAL(10, 2)) AS [Size on
Disk (MB)],vfs.num_of_reads, vfs.num_of_writes, vfs.io_stall_read_ms,
vfs.io_stall_write_ms,

CAST(100. * vfs.io_stall_read_ms/(vfs.io_stall_read_ms +
vfs.io_stall_write_ms) AS DECIMAL(10,1)) AS [IO Stall Reads Pct],CAST(100. *
vfs.io_stall_write_ms/(vfs.io_stall_write_ms + vfs.io_stall_read_ms) AS
DECIMAL(10,1)) AS [IO Stall Writes Pct],(vfs.num_of_reads +
vfs.num_of_writes) AS [Writes + Reads],
CAST(vfs.num_of_bytes_read/1048576.0 AS DECIMAL(10, 2)) AS [MB Read],
CAST(vfs.num_of_bytes_written/1048576.0 AS DECIMAL(10, 2)) AS [MB
Written],CAST(100. * vfs.num_of_reads/(vfs.num_of_reads +
vfs.num_of_writes) AS DECIMAL(10,1)) AS [# Reads Pct],CAST(100. *
vfs.num_of_writes/(vfs.num_of_reads + vfs.num_of_writes) AS
DECIMAL(10,1)) AS [# Write Pct],CAST(100. *
vfs.num_of_bytes_read/(vfs.num_of_bytes_read +
```

```
vfs.num_of_bytes_written) AS DECIMAL(10,1)) AS [Read Bytes
Pct],CAST(100. * vfs.num_of_bytes_written/(vfs.num_of_bytes_read +
vfs.num_of_bytes_written) AS DECIMAL(10,1)) AS [Written Bytes Pct]

FROM sys.dm_io_virtual_file_stats(DB_ID(), NULL) AS vfs

INNER JOIN sys.database_files AS df WITH (NOLOCK)

ON vfs.[file_id]= df.[file_id] OPTION (RECOMPILE);
```

The above T-SQL script returns information about the I/O statistics of each file of all the Azure SQL database in Azure SQL Server.

```
SELECT @@SERVERNAME AS [Server Name], RTRIM([object_name]) AS
[Object Name], cntr_value AS [Memory Grants
Pending]

FROM sys.dm_os_performance_counters WITH (NOLOCK)

WHERE [object_name] LIKE N'%Memory Manager%' -- Handles named
instances

AND counter_name = N'Memory Grants Pending' OPTION (RECOMPILE);
```

The above T-SQL script returns information about the number of Memory grants pending for Azure SQL database.

```
SELECT SUM (unallocated_extent_page_count) AS [free pages],

(SUM (unallocated_extent_page_count)*1.0/128) AS [free space in MB]

FROM sys.dm_db_file_space_usage;
```

The above T-SQL script is used for Determining the Amount of Free Space in TempDB

```
SELECT SUM (version_store_reserved_page_count) AS [version store pages used],

 (SUM (version_store_reserved_page_count)*1.0/128) AS [version store space in MB]

FROM sys.dm_db_file_space_usage;
```

The above T-SQL script is used for Determining the Amount Space Used by the Version Store with in TEMPDB.

```
SELECT SUM (user_object_reserved_page_count) AS [user object pages used],

 (SUM(user_object_reserved_page_count)*1.0/128) AS [user object space in MB]

FROM sys.dm_db_file_space_usage;
```

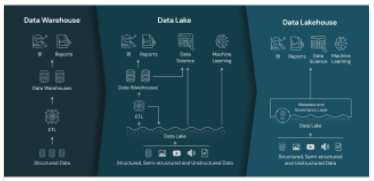
The above T-SQL script is used determining the Amount of Space Used by User Objects

Report this article

Naga Sai Krishna Pamidikondala
Cloud Solution Architecture | Microsoft | Data & AI

+ Follow

More from Naga Sai Krishna Pamidikondala



AN OVERVIEW OF THREE CLOUD DATA STORAGE PATTERNS - Data Warehous...

Naga Sai Krishna Pamidikon...

Cost Optimization case study – Azure Cosmos DB

Naga Sai Krishna Pamidikon...

SQL Server Database Migration to Azure Cloud

Naga Sai Krishna Pamidikon...

Optimizing the cost of Azure Cosmos Database - Using Azure Data Factory

Naga Sai Krishna Pamidikon...

See all 9 articles