High IO Utilization

Last updated by | Peter Hewitt | Sep 27, 2022 at 6:17 AM PDT

Contents

Common solution

This is the common solution article related to high IO issues that's displayed to customers in the Azure portal when creating a support ticket.

Resolve high IO utilization in Azure SQL Database

High IO utilization issues in Azure SQL Database are commonly caused by outdated index statistics, missing or fragmented indexes, query-plan regression, poorly designed queries, increased workload, and other factors. When IO usage is high, applications may experience performance-related issues such as time-outs and increased latency.

Learn how to resolve high IO usage by using our diagnostics, reviewing the recommendations, identifying responsible queries, and viewing the optimizing application performance through batching video.

Scan and select one or more of the following headings to help resolve your issues.

High IO diagnostics

We're running checks to identify causes of high IO usage on your database. The diagnosis takes approximately 30 seconds to complete.
<Diagnostics results displayed to customer>

Recommendations to resolve high IO utilization

IO usage is split into Data IO and Log IO. Run this T-SQL query to return the Data IO and Log IO usage percentages. For IO usage above 80% review and apply the recommendations.

```
SELECT end_time, avg_data_io_percent, avg_log_write_percent
FROM sys.dm_db_resource_stats
ORDER BY end_time DESC;
```

Category	Recommendation
Statistics outdated	Update statistics to reduce IO utilization and improve performance. For more information, see How to maintain Azure SQL Indexes and Statistics. Azure Automation can be used to configure a runbook to perform scheduled index and statistics maintenance.
Add resources	Scale the database to a higher service and compute tier to acquire more allocated resources (CPU power, memory, IO throughput, and storage). Adding resources enables you to react quickly when the database reaches, or is near, the current resource limits. The database can be scaled down following performance improvements through query tuning, statistics and index maintenance, and other modifications, or otherwise remain at the higher service tier if increased application usage can't be fixed using optimization methods. **[Configure pricing tier]** (button to configure pricing tier) **Note**: Business Critical and Premium service tiers are recommended for applications requiring high transaction rate, low IO latency, and high IO throughput as they have lower IO latency compared to General Purpose, Standard and Basic service tiers.
Query optimization	Identify the top IO-consuming queries from the Query Store using Query Performance Insight. Review the queries and apply recommendations manually, or enable Automatic Tuning to automatically apply them. **[Query Performance Insight]** (button to configure QPI) **[Enable Automatic Tuning]** (button to configure Automatic Tuning) Alternatively, access the information in Query Store using SSMS or T-SQL commands. There are a number of views in SSMS available, including **Top Resource-Consuming Queries**, which identifies queries with the largest resource consumption, **Regressed Queries** for pinpointing queries that have recently regressed in performance, and **Query Wait Statistics** for analyzing the most active wait categories and contributing queries. For more information, see monitor performance using Query Store and Performance tuning sample queries. For guidance including how to identify and add missing indexes using T-SQL see Tune applications and databases for performance in Azure SQL Database.

Category	Recommendation
Index maintenance strategy	Rebuilding indexes are resource intensive (CPU utilization, memory used, and IO). However, depending on the database workload and other factors, the benefits it provides can be vitally important. To avoid unnecessary resource utilization, do not perform index maintenance indiscriminately. Consider rebuilding indexes during low resource usage times or reorganizing indexes instead of rebuilding them. See Index maintenance strategy for additional maintenance best practices.
Batching strategy	Consider grouping separate transactions into batches to consolidate IO costs. For more information, see Batching for Performance Improvements.

IO utilization metrics

The following chart shows the average percentage of data IO usage, for your database, for the past 24 hours.
<IO usage chart for the previous displayed to customer>>

Identify currently running queries with high IO utilization using T-SQL

If high IO usage is currently happening, run this T-SQL command to return current waiting tasks along with the session ids, wait types, wait durations, query texts, query execution plans, program names, and additional details.

```
SELECT 'Waiting_tasks' AS [Information], owt.session_id,
  owt.wait_duration_ms, owt.wait_type, owt.blocking_session_id,
  owt.resource_description, es.program_name, est.text,
  est.dbid, eqp.query_plan, er.database_id, es.cpu_time,
  es.memory_usage*8 AS memory_usage_KB
FROM sys.dm_os_waiting_tasks owt
  INNER JOIN sys.dm_exec_sessions es ON owt.session_id = es.session_id
  INNER JOIN sys.dm_exec_requests er ON es.session_id = er.session_id
  OUTER APPLY sys.dm_exec_sql_text (er.sql_handle) est
  OUTER APPLY sys.dm_exec_query_plan (er.plan_handle) eqp
WHERE es.is_user_process = 1
ORDER BY owt.session_id;
GO
```

The top wait types associated with IO issues are:

- `PAGEIOLATCH_` : For data file IO issues (including `PAGEIOLATCH_SH` , `PAGEIOLATCH_EX` , `PAGEIOLATCH_UP`). If the wait type name contains IO, it points to an IO issue. If there is no IO in the page latch wait name, it points to a different type of problem (for example, tempdb contention).
- `LOG_RATE_GOVERNOR` , `POOL_LOG_RATE_GOVERNOR` , and `WRITE_LOG` : For log IO issues.

Identify queries in the past with high IO utilization using T-SQL

If high IO usage occurred in the past, Query Store captures a history of queries, plans, and runtime statistics, and retains them for review. Run this T-SQL query against Query Store to view the last two hours of tracked activity for buffer-related IO (if required, modify the `DATEADD` parameters in line `rsi.start_time>=DATEADD(HOUR, -2, GETUTCDATE())` to return results for another time frame).

```
WITH Aggregated AS (
  SELECT q.query_hash, SUM(total_query_wait_time_ms) total_wait_time_ms,
    SUM(total_query_wait_time_ms / avg_query_wait_time_ms) AS total_executions,
    MIN(qt.query_sql_text) AS sampled_query_text,
    MIN(wait_category_desc) AS wait_category_desc
  FROM sys.query_store_query_text AS qt
    JOIN sys.query_store_query AS q ON qt.query_text_id=q.query_text_id
    JOIN sys.query_store_plan AS p ON q.query_id=p.query_id
    JOIN sys.query_store_wait_stats AS waits ON waits.plan_id=p.plan_id
    JOIN sys.query_store_runtime_stats_interval AS rsi
      ON rsi.runtime_stats_interval_id=waits.runtime_stats_interval_id
  WHERE wait_category_desc='Buffer IO'
    AND rsi.start_time>=DATEADD(HOUR, -2, GETUTCDATE())
  GROUP BY q.query_hash), Ordered AS (SELECT query_hash, total_executions,
    total_wait_time_ms, sampled_query_text, wait_category_desc,
    ROW_NUMBER() OVER
      (ORDER BY total_wait_time_ms DESC, query_hash ASC) AS RN FROM Aggregated)
SELECT OD.query_hash, OD.total_executions, OD.total_wait_time_ms,
  OD.sampled_query_text, OD.wait_category_desc, OD.RN
FROM Ordered AS OD
WHERE OD.RN<=15
ORDER BY total_wait_time_ms DESC;
GO
```

Optimizing application performance through batching video

This video demonstrates how to improve IO performance through batching.

Demo: Optimizing application performance through batching (3:44)

Resources

- sys.dm_db_resource_stats
  Returns CPU, I/O, and memory consumption data for a SQL database.
- sys.dm_exec_requests
  Returns information about each request that is executing in SQL Server.
- sys.dm_os_waiting_tasks
  Returns information about the wait queue of tasks that are waiting on some resource.
- Identify IO performance issues
- Detectable types of query performance bottlenecks in Azure SQL Database
- Monitoring and performance tuning in SQL Database

How good have you found this content?