

# Hyperscale db query tuning

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:34 AM PDT

---

## Contents

- [Hyperscale DB Query Tuning](#)
  - [Understand hyperscale db architecture](#)
    - [Who should consider the Hyperscale service tier](#)
  - [Performance diagnostics on compute node](#)
  - [Troubleshooting](#)
    - [Scenario 1: Overall read slow issue](#)
    - [Scenario 2: Read slow for a single query](#)
    - [Scenario 3: Write slow issue](#)
  - [Mitigation](#)
    - [Scenario 1: Overall read slow issue](#)
    - [Scenario 2: Read slow for a single query](#)
    - [Scenario 3: Write slow issue](#)
  - [Public doc](#)

## Hyperscale DB Query Tuning

### Understand hyperscale db architecture

Hyperscale db is using [distributed functions architecture](#) ☐ include compute node, page server, log service and Azure storage. Hyperscale database is not suited for all customer scenarios, make sure you review customer's workload type to identify if they can achieve better performance with hyperscale.

### Who should consider the Hyperscale service tier

The Hyperscale service tier is intended for most business workloads as it provides great flexibility and high performance with independently scalable compute and storage resources. With the ability to autoscale storage up to 100 TB, it's a great choice for customers who:


- Have large databases on-premises and want to modernize their applications by moving to the cloud
- Are already in the cloud and are limited by the maximum database size restrictions of other service tiers (1-4 TB)
- Have smaller databases, but require fast vertical and horizontal compute scaling, high performance, instant backup, and fast database restore.

The Hyperscale service tier supports a broad range of SQL Server workloads, from pure OLTP to pure analytics, but it's primarily optimized for OLTP and hybrid transaction and analytical processing (HTAP) workloads.

### Important

Elastic pools do not support the Hyperscale service tier.

## Performance diagnostics on compute node

[General performance troubleshooting](#)  works for compute node. But a query may reach different components which cause delays.

## Troubleshooting

### Scenario 1: Overall read slow issue

1. Get issue time and run ASC -> SQL troubleshooter to get Hyperscale db basic information (appName and node name)
2. Identify the Reads come from RBPEX or Page Servers

```
MonDmIoRBPEXStats
| where TIMESTAMP >= datetime(2020-08-13 03:41) and TIMESTAMP <= datetime(2020-08-13 04:00)
| where AppName =~ "aa489d3a531f"
| where NodeName =~ "DB.26"
| where counter_name in ("DataFCBBytesRead", "RbpeFCBBytesRead")
| project TIMESTAMP, NodeName, counter_name, value
| order by TIMESTAMP asc
```

counter\_name "RbpeFCBBytesRead" indicates RBPEX/Local cache read on compute node, while "DataFCBBytesRead" indicates read on page server.

3. Check ASC -> SQL Troubleshooter -> Performance -> Hyperscale to see the read latency on page server
  - o "RBIO AsyncReadRequest Client Latency" graph shows roundtrip latency from compute to page server
  - o "RBIO AsyncReadRequest Server Latency" graph shows getPage latency on page server side.

If you see client latency is high and server latency contribute most of the latency, then possible root cause is getting page slow on pages server/log service. Go to step 4 to check further.

If you see client latency is high but server latency is low, then possible root cause is network slowness when fetching large amount of data from page server.

4. Identify read slow on Xlog When query reach page server to fetch data page, the data page may not be the latest and need to apply latest transaction log from log service. In such scenario, it should read transaction log from log service and can have additional latency if accessing different log layer.

In ASC, check "Xlog Read Tier with Count" graph to see where the read on transction lands. In most scenario, "reads\_from\_broker" contribute the most. If you notice extended period of reads from lc (log cache) , lz (landing zone) or lt(long term storage), it could result in poor performance.

### Scenario 2: Read slow for a single query

In this scenario, customer experiences slowness for one specific query.

1. Get issue time and run ASC -> SQL troubleshooter to get Hyperscale db basic information (appName and node name)

## 2. Get plan\_id for a query

```
SELECT q.query_id,qp.plan_id,q.query_hash,qp.plan_id,qp.query_plan_hash,qt.query_sql_text
,convert(xml,qp.query_plan) as plan_xml
FROM sys.query_store_plan qp
INNER JOIN sys.query_store_query q ON qp.query_id = q.query_id
INNER JOIN sys.query_store_query_text qt ON q.query_text_id = qt.query_text_id
WHERE qt.query_sql_text like ('%Cars%')
```

## 3. Check the page server I/O reads

- Option1: use sys.query\_store\_runtime\_stats

```
SELECT plan_id, last_execution_time,avg_duration,max_duration,count_executions, execution_type_desc,
```



- Option 2: use actual query execution plans xml Look for ActualPageServerReadAheads and ActualPageServerReads <RunTimeInformation>

```
<RunTimeCountersPerThread Thread="8" ActualRows="90466461" ActualRowsRead="90466461"
 Batches="0" ActualEndOfScans="1" ActualExecutions="1" ActualExecutionMode="Row"
 ActualElapseddms="133645" ActualCPUs="85105" ActualScans="1" ActualLogicalReads="6032256"
 ActualPhysicalReads="0" ActualPageServerReads="0" ActualReadAheads="6027814"
 ActualPageServerReadAheads="5687297" ActualLobLogicalReads="0" ActualLobPhysicalReads="0"
 ActualLobPageServerReads="0" ActualLobReadAheads="0" ActualLobPageServerReadAheads="0" />

</RunTimeInformation>
```

### Scenario 3: Write slow issue

Primary compute replica does not write directly to page servers, it relies on log service to replay the log on page servers and secondary. Data writes to the local RBPEX(file\_id 0) of sys.dm\_io\_virtual\_file\_stats, while log writes on primary compute node is a write to log landing zone. Log writes will show as a write to local RBPEX (file\_id 2) of sys.dm\_io\_virtual\_file\_stats.

#### 1. Identify write commit latency from ASC

In ASC -> SQL Troubleshooters -> Performance -> Hyperscale, you can check "VLDB Compute Commit Latency" graph to check the average commit latency on compute node. It shows two line represents how many time 90% of commit would take and how many time 50% of commit would take.

#### 2. Check wait type for the issue query

Query with "WRITE\_LOG" wait type indicates wait on transaction log write to log landing zone.

3. Check sys.dm\_io\_virtual\_file\_stats on log file for average write latency on transaction log and if there are many pending log io in the queue.

use <hyperscal database> select \* from sys.dm\_io\_virtual\_file\_stats(db\_id(),2)

## Mitigation

### Scenario 1: Overall read slow issue

Large read on page server will cause the slowness, so the mitigation is to reduce the page server read.

- Find the top query which have most page server read
- Tune those queries using general practice like execution plan analysis, index optimization or update statistics based on situation
- Consider scale the server to higher service tier to get large RBPEX and memory on compute node

### Scenario 2: Read slow for a single query

- Tune the issue query by analyzing the execution plan
- If slowness is caused by page server read, this should only happen when it is executed the first time to pull the data into RBPEX on compute node. Consider use some warm up query

### Scenario 3: Write slow issue

- If log write is slow due to log io throttling, consider upgrade to higher service tier
- Optimize the transactions to avoid too many small transactions
- If write slow is not happening on log, but simply data write slowness, following general practice to tune the query

## Public doc

SQL Hyperscale performance troubleshooting diagnostics

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-hyperscale-performance-diagnostics> 

## How good have you found this content?



-