

Azure SQL Database Hyperscale Assessment Query Execution Logs Extraction and Source Code Prerequisites

Contents

1. Introduction	3
2. Assessment Process	3
2.1 Databases	3
2.2 Database Volume	3
2.3 High Data Volume Tables	4
2.4 Users	5
2.5 User Tables	6
2.6 Objects	7
2.7 Total I/O Usage by Database	7
2.8 Top CPU and I/O Consuming Queries	8
2.9 Top CPU and I/O Consuming Procedures	9
2.10 Top CPU and I/O Consuming Functions	11
2.11 Data for Partitioning and Bucketing	12
2.12 Server Max & Avg CPU	13
2.13 List of Databases Having Avg_cpu > =70 %	13
2.14 Server Configuration	14
3. Source Code Assessment	15

1. Introduction

LeapLogic's Assessment profiles existing inventory, identifies workload complexity, interdependencies, and provides comprehensive actionable recommendations for migration to modern data platform.

Assessment checklist

Information	Format	Availability?
Database Object Count	File Export in text format	Choose an item.
Database Volume	File Export in text format	Choose an item.
High Data Volume Tables	File Export in text format	Choose an item.
Total Number of Users	File Export in text format	Choose an item.
Total Number of User Tables	File Export in text format	Choose an item.
Objects	File Export in text format	Choose an item.
Total I/O Usage by Database	File Export in text format	Choose an item.
Top CPU and I/O Consuming Queries	File Export in text format	Choose an item.
Top CPU and I/O Consuming Procedures	File Export in text format	Choose an item.
Top CPU and I/O Consuming Functions	File Export in text format	Choose an item.
Data for Partitioning and Bucketing	File Export in text format	Choose an item.
Server Max & Avg CPU	File Export in text format	Choose an item.
List of Databases Having Avg_cpu >=70	File Export in text format	Choose an item.
Server Configuration	File Export in text format	Choose an item.
Source Code Assessment	File Export in text format	Choose an item.

Follow the steps in section 2, 3 to collect the required information for assessment.

2. Assessment Process

All the environment specific variables are highlighted in the document in **Green**. The start and end date should be consistent for all the queries.

2.1 Databases

This query provides the total number of databases. Execute it and save the results in CSV format.

```
select count (*) from sys.databases
```

2.2 Database Volume

This query provides the database volume in MBs. Execute it and save the results in CSV format.

```

DECLARE @Databases TABLE
(
    database_nm VARCHAR(50),
    log_write DECIMAL(8,2),
    row_size_mb DECIMAL(8,2),
    total_size_mb DECIMAL(8,2)
)
INSERT INTO @Databases
SELECT
    database_name = CAST(database_name AS VARCHAR(50))
    ,log_write_secs = max(avg_log_write_percent*cpu_limit)
    , row_size_mb = max(storage_in_megabytes)
    ,total_size_mb = max(allocated_storage_in_megabytes)
FROM sys.resource_stats
where sku='Hyperscale'
GROUP BY database_name
select * FROM @Databases

```

name	dd_size
OWConfiguration	16.00 MB
OWDiagnostics	1072.00 MB
OWQueue	16.00 MB
ep	16.00 MB
plus	16.00 MB
naster	7.38 MB
node1	16.00 MB
nsdb	46.75 MB
ef	16.00 MB
ef1	16.00 MB
ReportServer	80.00 MB
ReportServerTempDB	16.00 MB
amndh	40.00 MB

2.3 High Data Volume Tables

This query provides the result for tables with a high data volume. Execute it for all the databases that needs to be migrated and save the results in CSV format.

```

DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Scheme_Name VARCHAR(50),
    Row_Count BIGINT,
    Space_Total_GB DECIMAL(10,2),

```

```

    Space_Used_GB DECIMAL(10,2),
    Space_UnUsed_GB DECIMAL(10,2)
)

INSERT INTO @UserTable
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
t.NAME AS TableName,
s.Name AS SchemaName,
p.rows AS RowCounts,
CAST(ROUND(((SUM(a.total_pages) * 8) / 1024.00/1024.00), 2) AS NUMERIC(36, 2)) AS TotalSpaceGB,
CAST(ROUND(((SUM(a.used_pages) * 8) / 1024.00/1024.00), 2) AS NUMERIC(36, 2)) AS UsedSpaceGB,
CAST(ROUND(((SUM(a.total_pages) - SUM(a.used_pages)) * 8) / 1024.00/1024.00, 2) AS NUMERIC(36, 2))
AS UnusedSpaceGB
FROM
    sys.tables t
INNER JOIN
    sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN
    sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
INNER JOIN
    sys.allocation_units a ON p.partition_id = a.container_id
LEFT OUTER JOIN
    sys.schemas s ON t.schema_id = s.schema_id
WHERE
    t.is_ms_shipped = 0
    AND i.OBJECT_ID > 255
GROUP BY
    t.Name, s.Name, p.Rows
HAVING CAST(ROUND(((SUM(a.used_pages) * 8) / 1024.00/1024.00), 2) AS NUMERIC(36, 2)) >=10
ORDER BY    UsedSpaceGB

select * FROM @UserTable

```

This SQL will collect databases with volume equal or above 10GB. **This step must be taken for all the databases.**

Note: The filter condition can be changed for collecting lower volumes.

2.4 Users

This query provides the total number of users. Execute it for **all the databases** that need to be migrated and save the results in CSV format.

```

DECLARE @UserType TABLE
(
    database_name VARCHAR(50),
    UserType VARCHAR(50),
    Num INT
)

INSERT INTO @UserType
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
type ,count(*)
from sys.database_principals
group by type

select * FROM @UserType

```

type

Type of principal
S = SQL Server user
U = Windows user
G = Windows group
A = Application role
R = Database role
C = Certificate mapped
K = Asymmetric key mapped

2.5 User Tables

This query provides the total number of user tables. Execute it for all the databases and save the results in CSV format.

```

DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Table_Type VARCHAR(50),
    Create_Date DATETIME,
    ModifiedDate DATETIME
)

INSERT INTO @UserTable
select
database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
Name,type_desc,create_date,modify_date
from sys.tables

```

```
select * FROM @UserTable
```

2.6 Objects

This query provides all the objects in a database. Execute it for all the databases that need to be migrated and save the results in CSV format.

The results can be restricted for system tables such as SYSTEM_TABLE, INTERNAL_TABLE etc

Eg: `where type_desc not in ('SYSTEM_TABLE', 'INTERNAL_TABLE', 'SERVICE_QUEUE')`

```
DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Obj_Name VARCHAR(500),
    obj_Type VARCHAR(50),
    Create_Date DATETIME,
    ModifiedDate DATETIME
)
INSERT INTO @UserTable
select
    database_name = CAST(DB_NAME(DB_ID()) AS VARCHAR(50)),
    Name,type_desc,create_date,modify_date
from sys.objects

select * FROM @UserTable
```

2.7 Total I/O Usage by Database

This query provides the total I/O usage by databases. Execute it and save the results in CSV format.

```
SELECT Name AS Database_Name
,SUM(num_of_reads)AS Number_of_Reads
,SUM(num_of_writes)AS Number_of_Writes
,(SUM(num_of_reads) + SUM(num_of_writes)) as TotalIO
FROM sys.dm_io_virtual_file_stats(NULL,NULL) I
INNER JOIN sys.databases D ON I.database_id = d.database_id
GROUP BY name
ORDER BY TotalIO desc
```

2.8 Top CPU and I/O Consuming Queries

This query provides top CPU and I/O consuming queries. Execute it and save the results in CSV format.

```
DECLARE @startdt VARCHAR(10)
DECLARE @enddt VARCHAR(10)

-- Provide start and end date
SET @startdt = '2017-03-05'
SET @enddt = '2022-04-05'

select CONCAT('IDWWM', '~',
             sess.original_login_name, '~',
             Appld, '~',
             sess.client_interface_name, '~',
             convert(VARCHAR,qs.last_execution_time,120), '~',
             (total_worker_time/execution_count)/1000000.0, '~',
             (total_logical_reads+total_physical_reads+total_logical_writes)/execution_count, '~',
             0, '~',
             convert(VARCHAR,(qs.last_execution_time + qs.total_elapsed_time/86400000000),120), '~',
             convert(VARCHAR,qs.last_execution_time,120), '~',
             0, '~',
             row_number() over (ORDER BY total_worker_time/execution_count desc), '~',
             (total_worker_time/execution_count)/1000000.0, '~',
             (total_logical_reads+total_physical_reads+total_logical_writes), '~',
             tb4.ampcputime, '~',
             tb4.TotalIOCount, '~',
             qs.total_elapsed_time/1000000.0, '~',
             OBJECT_SCHEMA_NAME(st.objectid,st.dbid), '~',
             db.name, '~',
             REPLACE(CAST(text as NVARCHAR(MAX)), CHAR(10), ' '), '~',
             execution_count
            )
      from sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.plan_handle) st
INNER JOIN
(
SELECT DISTINCT
a.last_execution_time,
SUM(total_worker_time)/1000000.0 as AMPCPUTIME,
(SUM(total_logical_reads)+sum(total_physical_reads)+sum(total_logical_writes)) AS TOTALIOCOUNT
```



```

FROM
sys.dm_exec_query_stats a
WHERE
cast(a.last_execution_time as date) BETWEEN CAST(@startdt+' 00:00:00:00' as DATETIME) AND
CAST(@enddt+' 00:00:00:00' AS DATETIME)
GROUP BY a.last_execution_time
)TB4 ON TB4.last_execution_time = qs.last_execution_time
inner join sys.databases db
on ( db.database_id = st.dbid)
LEFT JOIN (
SELECT DISTINCT es.program_name as AppId, es.client_interface_name,es.original_login_name,sp.dbid
FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
on es.session_id = sp.spid
) AS Sess
ON Sess.dbid = st.dbid;

```

Change the value of start date and end date in the parameter at the beginning of the script. Save the output of above query in CSV format if you are executing it from an editor such as SSMS. If the log size is too large, then use the alternative method (see below) to generate the output.



SQL_Server_query_log.sql

Copy the attached SQL script (SQL_Server_query_log.sql) at an appropriate location. Change start and end date as required. Execute it using the below command from Windows command line and share the output file.

This method can be used to execute any SQL command through the command prompt.

```
sqlcmd -S SQL_Server_Name -U username -P password -i ./SQL_Server_query_log.sql -o output_file.log
```

2.9 Top CPU and I/O Consuming Procedures

This query provides top CPU and I/O consuming procedures.

```

DECLARE @startdt VARCHAR(10)
DECLARE @enddt VARCHAR(10)

-- Provide start and end date
SET @startdt = '2019-03-05'
SET @enddt = '2022-04-05'

select CONCAT('IDWWM', '~~',
            sess.original_login_name, '~~',

```

```

    Appld,'~~',
        sess.client_interface_name,'~~',
        convert(VARCHAR,d.last_execution_time,120),'~~',
        (d.total_elapsed_time/d.execution_count)/1000000.0,'~~',
        (total_logical_reads+total_physical_reads+total_logical_writes)/d.execution_count,'~~',
        0,'~~',
        convert(VARCHAR,(d.last_execution_time + d.total_elapsed_time/86400000000),120),'~~',
        convert(VARCHAR,d.last_execution_time,120),'~~',
        0,'~~',
        row_number() over (ORDER BY total_worker_time/d.execution_count desc),'~~',
        (total_worker_time/execution_count)/1000000.0,'~~',
        (total_logical_reads+total_physical_reads+total_logical_writes),'~~',
        tb4.ampcputime,'~~',
        tb4.TotalIOCount,'~~',
        d.total_elapsed_time/1000000.0,'~~',
        OBJECT_SCHEMA_NAME(d.object_id,d.database_id), '~~',
        db.name,'~~',
        REPLACE(CAST(text as NVARCHAR(MAX)), CHAR(10), ' '), '~~',
        execution_count
    )
FROM sys.dm_exec_procedure_stats AS d
CROSS APPLY sys.dm_exec_sql_text(d.sql_handle) st
INNER JOIN
(
    SELECT DISTINCT
    a.last_execution_time,
    SUM(total_worker_time)/1000000.0 as AMPCPUTIME,
    (SUM(total_logical_reads)+sum(total_physical_reads)+sum(total_logical_writes)) AS TOTALIOCOUNT
    FROM
    sys.dm_exec_procedure_stats AS a
    WHERE
    cast(a.last_execution_time as date) BETWEEN CAST(@startdt+' 00:00:00:00' as DATETIME) AND
    CAST(@enddt+' 00:00:00:00' AS DATETIME)
    GROUP BY a.last_execution_time
)TB4 ON TB4.last_execution_time = d.last_execution_time
inner join sys.databases db
on ( db.database_id = d.database_id)
LEFT JOIN (
    SELECT DISTINCT es.program_name as Appld, es.client_interface_name,es.original_login_name,sp.dbid
    FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
    on es.session_id = sp.spid
) AS Sess

```

```
ON Sess.dbid = d.database_id;
```

2.10 Top CPU and I/O Consuming Functions

This query provides top CPU and I/O consuming functions.

```
DECLARE @startdt VARCHAR(10)
DECLARE @enddt VARCHAR(10)

-- Provide start and end date
SET @startdt = '2017-03-05'
SET @enddt = '2022-04-05'

select CONCAT('IDWWM', '~',
            sess.original_login_name, '~',
            Appld, '~',
            sess.client_interface_name, '~',
            convert(VARCHAR,d.last_execution_time,120), '~',
            (d.total_elapsed_time/d.execution_count)/1000000.0, '~',
            (total_logical_reads+total_physical_reads+total_logical_writes)/d.execution_count, '~',
            0, '~',
            convert(VARCHAR,(d.last_execution_time + d.total_elapsed_time/86400000000),120), '~',
            convert(VARCHAR,d.last_execution_time,120), '~',
            0, '~',
            row_number() over (ORDER BY total_worker_time/d.execution_count desc), '~',
            (total_worker_time/execution_count)/1000000.0, '~',
            (total_logical_reads+total_physical_reads+total_logical_writes), '~',
            tb4.ampcputime, '~',
            tb4.TotalIOCount, '~',
            d.total_elapsed_time/1000000.0, '~',
            OBJECT_SCHEMA_NAME(d.object_id,d.database_id), '~',
            db.name, '~',
            REPLACE(CAST(text as NVARCHAR(MAX)), CHAR(10), ' '), '~',
            execution_count
            )
FROM sys.dm_exec_function_stats AS d
CROSS APPLY sys.dm_exec_sql_text(d.sql_handle) st
INNER JOIN
(
SELECT DISTINCT
a.last_execution_time,
SUM(total_worker_time)/1000000.0 as AMPCPUTIME,
```

```

(SUM(total_logical_reads)+sum(total_physical_reads)+sum(total_logical_writes)) AS TOTALIOCOUNT
FROM
sys.dm_exec_function_stats AS a
WHERE
cast(a.last_execution_time as date) BETWEEN CAST(@startdt+' 00:00:00:00' as DATETIME) AND
CAST(@enddt+' 00:00:00:00' AS DATETIME)
GROUP BY a.last_execution_time
)TB4 ON TB4.last_execution_time = d.last_execution_time
inner join sys.databases db
on ( db.database_id = d.database_id)
LEFT JOIN (
SELECT DISTINCT es.program_name as Appld, es.client_interface_name,es.original_login_name,sp.dbid
FROM sys.dm_exec_sessions es inner join sys.sysprocesses sp
on es.session_id = sp.spid
) AS Sess
ON Sess.dbid = d.database_id;

```

An alternative method to generate the log file for Queries/Procedures or Functions is using sqlcmd (T-SQL) CLI

2.11 Data for Partitioning and Bucketing

This query extracts table size and column details to be utilized for partition and bucket recommendation. Execute it and save the results in CSV format.

Note: This query provides the details for the database that we are in. You need to execute it for all the databases. Also, you need to generate the statistics before executing this query.

```

DECLARE @UserTable TABLE
(
    database_name VARCHAR(100),
    Table_Name VARCHAR(500),
    Table_Size_Bytes DECIMAL(18,2),
    Row_Num INT,
    Column_Name VARCHAR(100),
    Unique_Val_Num INT
)
INSERT INTO @UserTable
SELECT isc.table_catalog AS DATABASE_NAME,
       t.NAME           AS Table_Name,
       Cast(Round(( Sum(a.used_pages) / 128.00 ), 2) AS NUMERIC(36, 2)) * ( 2014 * 1024 ) AS TABLE_SIZE_
BYTES,

```

```

p.rows      AS NUM_ROWS,
isc.column_name AS Column_Name,
" AS num_unique_val
FROM sys.tables t
INNER JOIN sys.indexes i
    ON t.object_id = i.object_id
INNER JOIN sys.partitions p
    ON i.object_id = p.object_id
    AND i.index_id = p.index_id
INNER JOIN sys.allocation_units a
    ON p.partition_id = a.container_id
INNER JOIN sys.schemas s
    ON t.schema_id = s.schema_id
INNER JOIN information_schema.columns AS isc
    ON t.NAME = isc.table_name
GROUP BY t.NAME,
s.NAME,
p.rows,
isc.table_catalog,
isc.column_name

select * FROM @UserTable

```

2.12 Server Max & Avg CPU

This query extracts Average and Max CPU Utilization, Data IO & Memory usage. Execute it and save the results in CSV format.

```

SELECT
    AVG(avg_cpu_percent) AS 'Average CPU Utilization In Percent',
    MAX(avg_cpu_percent) AS 'Maximum CPU Utilization In Percent',
    AVG(avg_data_io_percent) AS 'Average Data IO In Percent',
    MAX(avg_data_io_percent) AS 'Maximum Data IO In Percent',
    AVG(avg_log_write_percent) AS 'Average Log Write I/O Throughput Utilization In Percent',
    MAX(avg_log_write_percent) AS 'Maximum Log Write I/O Throughput Utilization In Percent',
    AVG(avg_memory_usage_percent) AS 'Average Memory Usage In Percent',
    MAX(avg_memory_usage_percent) AS 'Maximum Memory Usage In Percent'
FROM sys.dm_db_resource_stats;

```

2.13 List of Databases Having Avg_cpu >=70 %

This query List Database name which are having CPU utilization >=70%. Execute it and save the results in CSV format.

Note: The filter condition can be changed for collecting lower AVG CPU.

```
DECLARE @s datetime;
DECLARE @e datetime;
SET @s= DateAdd(d,-7,GetUTCDate());
SET @e= GETUTCDATE();

SELECT database_name, AVG(avg_cpu_percent) AS Average_CPU_Utilization , cpu_limit As 'Number of Cores' ,GETUTCDATE() as Date_of_extraction
FROM sys.resource_stats
WHERE sku = 'Hyperscale'
and start_time BETWEEN @s AND @e
GROUP BY database_name,cpu_limit
HAVING AVG(avg_cpu_percent) >= 70;
```

2.14 Server Configuration

This query List Details Related to server instance for eg : type of Service tier,CInstance type and server name. . Execute it and save the results in CSV format.

```
select D.service_objective,D.edition,CONVERT(nvarchar(50),SERVERPROPERTY('ServerName')) as
server_name ,X.vcores
from sys.database_service_objectives D CROSS JOIN
(select max(cpu_limit) as vcores from sys.resource_stats ) X
where D.edition='Hyperscale'
```

3. Source Code Assessment

Provide the below active or in-scope SQL Server source code artifacts as applicable in the migration scope.

Sl. No.	Code Artifact	Criticality	Remarks
1	Orchestration Scripts (Control-M / AutoSys / Cron etc.)	Must	To identify interdependencies across scheduler scripts / jobs, queries, and dependent workloads
2	Procedures / Functions*	Must	To identify workload complexity, query patterns, query count etc. for effort estimation and technical debt
3	Views	Must	To identify view complexity, patterns and effort estimations
4	Shell Scripts*	Must	To identify count, dependencies, SQL queries and PL/SQL, logic (example: email, notification etc.) and effort estimations
5	DDL	Must	To identify column usage, and provide recommendation on column level lineage, and query optimization on the target system
6	DML / SQL Files*	Must	To identify count, dependencies, SQL queries and effort estimations

Note:

Limit: Assuming the orchestration script is a trigger point for every single use case execution in the existing setup. If customer is not comfortable sharing all the workloads, then share those workloads which are referred or executed through the orchestration scripts. However, in such scenarios the scope and effort estimates will be based on the given workloads.