Use dynamic management views to identify and troubleshoot query performance

Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

Dynamic Management Views provide a programmatic experience for monitoring the Azure Synapse Analytics SQL pool activity by using the Transact-SQL language. The views that are provided, not only enable you to troubleshoot and identify performance bottlenecks with the workloads working on your system, but they are also used by other services such as Azure Advisor to provide recommendations about Azure Synapse Analytics.

There are over 90 Dynamic Management Views that can queried against dedicated SQL pools to retrieve information about the following areas of the service:

- Connection information and activity
- SQL execution requests and queries
- Index and statistics information
- Resource blocking and locking activity
- Data movement service activity
- Errors

The following is an example of monitoring query execution of the Azure Synapse Analytics SQL pools. The first step involves checking the connections against the server first, before checking the query execution activity.

Monitoring connections

All logins to your data warehouse are logged to sys.dm_pdw_exec_sessions. The session_id is the primary key and is assigned sequentially for each new logon.

SQLCopy
```
-- Other Active ConnectionsSELECT * FROM sys.dm_pdw_exec_sessions
where status <> 'Closed' and session_id <> session_id();
```

Monitor query execution

All queries executed on SQL pool are logged to sys.dm_pdw_exec_requests. The request_id uniquely identifies each query and is the primary key for this DMV. The request_id is assigned sequentially for each new query and is prefixed with QID, which stands for query ID. Querying this DMV for a given session_id shows all queries for a given logon.

Step 1

The first step is to identify the query you want to investigate

```
-- Monitor active queries
SELECT *
FROM sys.dm_pdw_exec_requests
WHERE status not in ('Completed','Failed','Cancelled')
  AND session_id <> session_id()
ORDER BY submit_time DESC;

-- Find top 10 queries longest running queries
SELECT TOP 10 *
FROM sys.dm_pdw_exec_requests
ORDER BY total_elapsed_time DESC;
```

From the preceding query results, **note the Request ID** of the query that you would like to investigate.

Queries in the **Suspended** state can be queued due to a large number of active running queries. These queries also appear in the sys.dm_pdw_waits waits query with a type of UserConcurrencyResourceType. For information on concurrency limits, see Memory and concurrency limits or Resource classes for workload management. Queries can also wait for other reasons such as for object locks. If your query is waiting for a resource, see Investigating queries waiting for resources further down in this article.

To simplify the lookup of a query in the sys.dm_pdw_exec_requests table, use LABEL to assign a comment to your query, which can be looked up in the sys.dm_pdw_exec_requests view.

```
-- Query with Label
SELECT *
FROM sys.tables
OPTION (LABEL = 'My Query')
;

-- Find a query with the Label 'My Query'
-- Use brackets when querying the label column, as it it a key word
SELECT  *
FROM    sys.dm_pdw_exec_requests
WHERE   [label] = 'My Query';
```

Step 2

Use the Request ID to retrieve the querys distributed SQL (DSQL) plan from sys.dm_pdw_request_steps

```
-- Find the distributed query plan steps for a specific query.
-- Replace request_id with value from Step 1.

SELECT * FROM sys.dm_pdw_request_steps
WHERE request_id = 'QID####'
ORDER BY step_index;
```

When a DSQL plan is taking longer than expected, the cause can be a complex plan with many DSQL steps or just one step taking a long time. If the plan is many steps with several move operations, consider optimizing your table distributions to reduce data movement.

The Table distribution article explains why data must be moved to solve a query. The article also explains some distribution strategies to minimize data movement.

To investigate further details about a single step, the operation_type column of the long-running query step and note the **Step Index**:

- Proceed with Step 3 for **SQL operations**: OnOperation, RemoteOperation, ReturnOperation.
- Proceed with Step 4 for **Data Movement operations**: ShuffleMoveOperation, BroadcastMoveOperation, TrimMoveOperation, PartitionMoveOperation, MoveOperation, CopyOperation.

Step 3

Use the Request ID and the Step Index to retrieve details from sys.dm_pdw_sql_requests, which contains execution information of the query step on all of the distributed databases.

```sql
-- Find the distribution run times for a SQL step.
-- Replace request_id and step_index with values from Step 1 and 3.

SELECT * FROM sys.dm_pdw_sql_requests
WHERE request_id = 'QID####' AND step_index = 2;
```

When the query step is running, DBCC PDW_SHOWEXECUTIONPLAN can be used to retrieve the SQL Server estimated plan from the SQL Server plan cache for the step running on a particular distribution.

```sql
-- Find the SQL Server execution plan for a query running on a specific SQL pool or control node.
-- Replace distribution_id and spid with values from previous query.

DBCC PDW_SHOWEXECUTIONPLAN(1, 78);
```

Step 4

Use the Request ID and the Step Index to retrieve information about a data movement step running on each distribution from sys.dm_pdw_dms_workers.

```sql
-- Find information about all the workers completing a Data Movement Step.
-- Replace request_id and step_index with values from Step 1 and 3.

SELECT * FROM sys.dm_pdw_dms_workers
WHERE request_id = 'QID####' AND step_index = 2;
```

- Check the total_elapsed_time column to see if a particular distribution is taking longer than others for data movement.
- For the long-running distribution, check the rows_processed column to see if the number of rows being moved from that distribution is larger than others. If so, this finding might indicate skew of your underlying data. One cause for data skew is distributing on a column with many NULL values (whose rows will all land in the same distribution). Prevent slow queries by avoiding distribution on these types of columns or filtering your query to eliminate NULLs when possible.

If the query is running, you can use DBCC PDW_SHOWEXECUTIONPLAN to retrieve the SQL Server estimated plan from the SQL Server plan cache for the currently running SQL Step within a particular distribution.

```sql
-- Find the SQL Server estimated plan for a query running on a specific SQL pool Compute or control node.
-- Replace distribution_id and spid with values from previous query.

DBCC PDW_SHOWEXECUTIONPLAN(55, 238);
```

Dynamic Management Views (DMV) only contains 10,000 rows of data. On heavily utilized systems this means that data held in this table may be lost with hours, or even minutes as data is managed in a first in, first out system. As a result you can potentially lose meaningful information that can help you diagnose query performance issues on your system. In this situation, you should use the Query Store.

You can also monitor additional aspects of Azure Synapse SQL pools including:

- Monitoring waits
- Monitoring tempdb
- Monitoring memory
- Monitoring transaction log
- Monitoring PolyBase

You can view information about monitoring these areas here