Synapse Resultset Cache Demo

Contents

[Objective 1](#_Toc61876329)

[Prerequisites 1](#_Toc61876330)

[Clear cache, buffers etc 1](#_Toc61876331)

[Execute the following (in my case on 125 million row fact table with DWU200 2](#_Toc61876332)

[Find the execution time using DMV 2](#_Toc61876333)

[Run the same query again the run find the request\_id and include whether there was a cache hit 2](#_Toc61876334)

[Now run this simulating multiple users using the SQL Query Stress tool 3](#_Toc61876335)

[Setup the Database connection 4](#_Toc61876336)

[Run the following query to get the instances and time: 5](#_Toc61876337)

[How many resultsets of aggregate queries about this size can be cached: 6](#_Toc61876338)

[Summary: 7](#_Toc61876339)

[References: 7](#_Toc61876340)

[Authors 8](#_Toc61876341)

[Reviewer 8](#_Toc61876342)

# Objective

The objective of this demo is to show that 100s to thousands of users can run queries in less time than a query would take for one user without Resultset caching!

# Prerequisites

A Synapse Dedicated Pool with Adventureworks DW

Download the SQL Query Stress tool [here](https://github.com/ErikEJ/SqlQueryStress)

# Clear cache, buffers etc

--reset performance

DBCC DROPCLEANBUFFERS

DBCC FREEPROCCACHE

DBCC DROPRESULTSETCACHE

drop view mviewtest

-- Make sure that resultset cache is turned on in the Master database using:

ALTER DATABASE dedpoolmoss SET RESULT\_SET\_CACHING ON;

# Execute the following (in my case on 125 million row fact table with DWU200

select [EnglishProductName], [SalesTerritoryRegion], [SalesTerritoryCountry],

sum([SalesAmount]) SalesAmount, sum([TotalProductCost]) [TotalProductCost], avg([DiscountAmount]) avgDiscountAmount

FROM [dbo].[FactInternetSales] s

inner join dbo.dimdate d on

d.datekey=s.orderdatekey

inner join [dbo].[DimProduct] p

on s.productkey=p.productkey

inner join [dbo].[DimCustomer] c

on c.customerkey=s.customerkey

inner join [dbo].[DimSalesTerritory] ds

on ds.salesterritorykey=s.[SalesTerritoryKey]

where [CommuteDistance] not in('0-1')

group by ds.[SalesTerritoryRegion], ds.[SalesTerritoryCountry], [EnglishProductName]

# Find the execution time using DMV

--- start test: determine what the request\_id is for the query

SELECT \* FROM sys.dm\_pdw\_exec\_requests

order by end\_time desc

SELECT total\_elapsed\_time, submit\_time, end\_time FROM sys.dm\_pdw\_exec\_requests

where request\_id='QID59358'

order by end\_time desc

In this case the elapsed time was about 33 seconds:

Graphical user interface, text, application

Description automatically generated

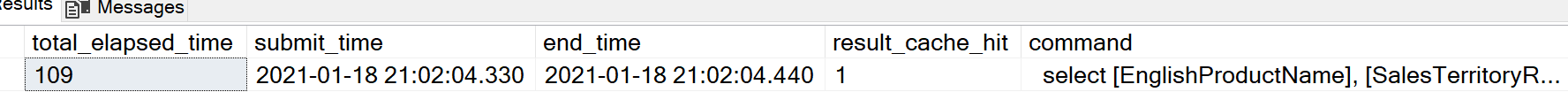
# Run the same query again the run find the request\_id and include whether there was a cache hit

SELECT total\_elapsed\_time, submit\_time, end\_time, result\_cache\_hit, command

FROM sys.dm\_pdw\_exec\_requests

where request\_id='QID59456'

order by end\_time desc

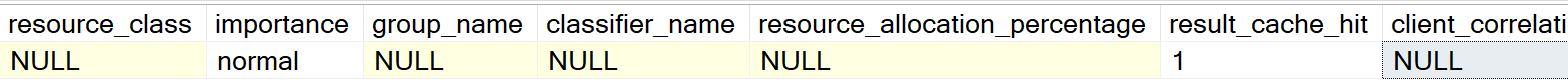


The total elapsed time is now 109 milliseconds.

That took only .0033 of the time previously, or stated another way, it’s a 3000% percentage increase.

Note that it was a cache hit.

Also, note that it did not take any of the resource class or concurrent slots:



# Now run this simulating multiple users using the SQL Query Stress tool

Setup: File/Options need to turn off the Collect I/O and Time Statistics not used in Synapse.

Graphical user interface, text, application

Description automatically generated

## Setup the Database connection

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Paste in the query we ran previously and set the number of threads to 1 and click Go. Then set the number to 100. This will simulate 100 users running this query:

Graphical user interface, text, application

Description automatically generated

## Run the following query to get the instances and time:

Text, application

Description automatically generated

For 101 queries: it took 12435 or 123 milliseconds per query.

But these ran concurrently so you could say 100 users ran a query in a total of less than 3 seconds: (some of this is latency on the client! So it’s better than that), you can show using this query:

SELECT sum(total\_elapsed\_time) as totaltime, count(result\_cache\_hit),

(max(end\_time)-min(submit\_time)) as elaspsed,

command

FROM sys.dm\_pdw\_exec\_requests

WHERE result\_cache\_hit = 1

group by command

Graphical user interface, application

Description automatically generated

# How many resultsets of aggregate queries about this size can be cached:

--drop the cache if it exists

DBCC DROPRESULTSETCACHE

--- show the space in the cache before this query

DBCC SHOWRESULTCACHESPACEUSED

--run aggregate query (on 125million rows

select [EnglishProductName], [SalesTerritoryRegion], [SalesTerritoryCountry],

sum([SalesAmount]) SalesAmount, sum([TotalProductCost]) [TotalProductCost],

avg([DiscountAmount]) avgDiscountAmount

FROM [dbo].[FactInternetSales] s

inner join dbo.dimdate d on

d.datekey=s.orderdatekey

inner join [dbo].[DimProduct] p

on s.productkey=p.productkey

inner join [dbo].[DimCustomer] c

on c.customerkey=s.customerkey

inner join [dbo].[DimSalesTerritory] ds

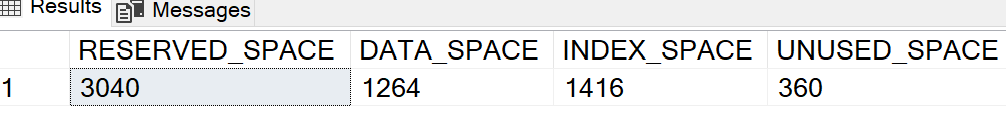
on ds.salesterritorykey=s.[SalesTerritoryKey]

where [CommuteDistance] not in('0-1')

group by ds.[SalesTerritoryRegion], ds.[SalesTerritoryCountry], [EnglishProductName]

--show space in cache used after the query

DBCC SHOWRESULTCACHESPACEUSED



Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Or about 50,000 distinct resultsets of the same size could be cached! And since it is adaptive, if it gets full, the oldest cached queries will be released.

# Summary:

The above simulation of 100 users running a query took less than 3 seconds elapsed time, rather than around 33 seconds for one user. Another way to look at it, in the same time it took for one query, we could have 1,100 users run the same query with resultset caching! (3 seconds x 11 =33 so 11 x 100 users = 1,100 users.

So when the same queries are being cached, it can be allow many users compared to paying for compute on competitors platforms!

This is when the same exact queries are being cached, but note that the size of the result set cache is huge and the amount of data that is cached is relatively small (just the results). For example, the size of the resultset cache for this one query is about:

# References:

[Performance tuning with result set caching - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/performance-tuning-result-set-caching)

[ErikEJ/SqlQueryStress: SQL query stress simulator created by Adam Machanic http://dataeducation.com/sqlquerystress-the-source-code/ (github.com)](https://github.com/ErikEJ/SqlQueryStress)

[SQL Query Stress Tool (mssqltips.com)](https://www.mssqltips.com/sqlservertip/2730/sql-query-stress-tool/#:~:text=%20SQL%20Query%20Stress%20Tool%20%201%20Step,we%20are%20going%20to%20clear%20out...%20More%20)

[DBCC SHOWRESULTCACHESPACEUSED (Transact-SQL) - SQL Server | Microsoft Docs](https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-showresultcachespaceused-transact-sql?view=azure-sqldw-latest)

# Authors

Steve Moss, Phillip Novak

# Reviewer

Saket Suman