Performance Indicators

https://www.dwhpro.com/teradata-dbql-performance-tuning-part-2-indicators/

In the first article about DBQL Performance Tuning, we talked about the different tables which are available and how DBQL logging should be set up to get the best out of these tables.

In this article, we will show you how these tables can be used to identify queries which may not be optimized. We use **performance indicators** to identify characteristics which suggest that a query is not being executed in an optimal way:

The Product Join Indicator

As its name suggests, we use the Product Join Indicator to identify unwanted product joins. In simple words, these are product joins which consume a lot more CPU than IOs.

The bad product joins (the ones we don’t want) can be identified by calculating the ratio between CPU Usage and IO Count because the CPU will be heavily used for compare operations during the join step.

The Product Join Indicator can be easily calculated from the DBC.DBQLOGTBL table:

**CASE WHEN TotalIOCount = 0 THEN 0 ELSE (AMPCPUTime \* 1000)/TotalIOCount END**

Values of 3 and above may indicate product join activity.

I just want you to keep in mind that high values could also be related to other reasons, here are some examples:

* Huge aggregation steps are done in memory. There is usually a very small amount of IOs, the operation itself is done in memory with a lot of CPU usage
* Duplicate Row Checks require a huge amount of CPU in case of many hash collisions.
* String Operations, as we often see in queries built by BI tools, usually require a lot of CPU.
* Multi Value Compression will increase the Product Join Indicator value: As more rows can be packed into one data block, fewer IOs are required to move the rows into memory. The result is a lower Product Join Indicator value.

The Unnecessary IO Indicator

This indicator is used to identify queries which consume a lot of IO but only small amount of CPU seconds.

This pattern is usually caused by large scans and can be a hint that secondary indexes are missing, or the primary index is not chosen in an optimal way.

The Unnecessary IO Indicator can be easily calculated from the DBC.DBQLOGTBL table:

**CASE WHEN AMPCPUTime = 0 THEN 0 ELSE TotalIOCount/(AMPCPUTime \* 1000) END**

Typically, values of 3 and above indicate large scans. Such queries should be analyzed for missing indexes, etc.

The CPU and IO Skew Indicator

We all know that skewed workload is one of the main performance issues on a Teradata System.

We can use these indicators to identify issues such as a bad choice of data redistribution strategy (join preparation), base tables with a skewed primary index, etc.

The CPU Skew Indicator can be easily calculated from the DBC.DBQLOGTBL table:

**CASE WHEN (AMPCPUTime / (HASHAMP()+1) =0 THEN 0 ELSE MaxAmpCPUTime/(AMPCPUTime / (hashamp()+1) END**

The IO Skew Indicator can be easily calculated from the DBC.DBQLOGTBL table:

**CASE WHEN (TotalIOCount / (HASHAMP()+1) =0 THEN 0 ELSE MaxAmpIO/(TotalIOCount / (HASHAMP()+1) END**

Queries with values above 2 should be considered for tuning.

The Impact CPU Indicator

This indicator measures the impact of skewing on the system performance

The Impact CPU Indicator can be easily calculated from the DBC.DBQLOGTBL table:

**MaxAmpCPUTime \* (HASHAMP()+1)**

The larger this value, the higher is the impact on Teradata system parallelism.

High CPU or High IO in general

These are the two base measures in performance tuning, which are used to calculate all the other indicators. Goal is always to reduce these absolute values.

The Response Time

While on its own not really a useful indicator, when combined with other indicators it can become a target for optimization. We will talk about query run times, and how they can be used for performance tuning in one of the next articles. For now, just keep in mind that run times are influenced by many different variables (workload delays, etc.) and it makes only sense to use them in combination with additional indicators.

**Whats next?** Equipped with this list of indicators, we will start to hunt down some really bad queries in the upcoming article about DBQL tables. Stay tuned!