SAP HANA Query Optimization techniques

Kumail Saifuddin Saif

SAP Technical Architect & Projects Delivery Manager
Accenture LLP, USA
kumail.saif@gmail.com

Abstract
Query performance is a key aspect for any development done in SAP HANA and it is very common to have performance issues when the data size grows in the system. SAP HANA offers effective tools and techniques for developers to make sure the query is optimized as per the hardware of SAP HANA database. Understanding the details of Query processing by different HANA engines like Calculation engine, Join engine, OLAP engine to name a few, and the role of the SQL optimizer, SQL plan cache, PlanViz, OOM dumps helps developers to write and test the code to make sure performance is optimized.

Keywords: SAP HANA, Query Performance, SQL, Performance optimization, SQL optimizer

1 Introduction:

SAP HANA offers many advancements over its predecessor ECC systems, one of the key features being in-memory processing which enables computation of a high volume of data in a very short time. However it is very important for developers to build the code of a HANA query in an effective manner and to be able to analyze the query performance with the available tools and techniques offered by SAP HANA.

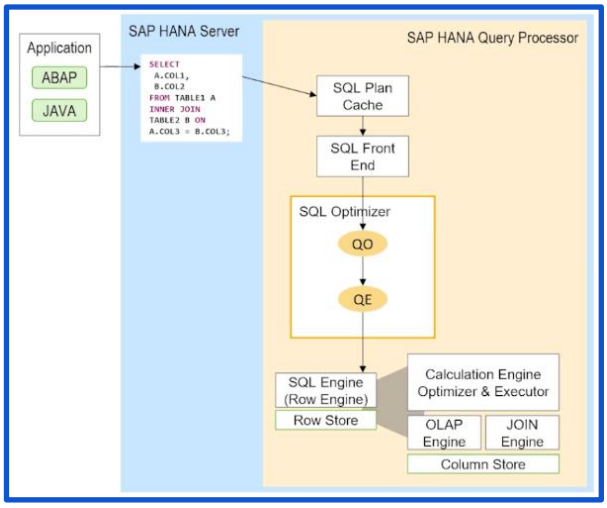
2 SQL Query Processing:

When the query comes in SAP HANA, it goes through SQL frontend, SQL optimizer and as per the generated SQL plan row store or column store. As a row store engine there is a SQL engine. As a column store, there are OLAP engine, JOIN engine and Calculation engine. These engines have their own optimizers and caches. For a query execution, the existing SQL plan is checked in the SQL plan cache as the very first step in the processing. If the plan is not found, the SQL front end checks for the syntax of the query and then sends it to the SQL optimizer. SQL optimizer generates an optimized plan for the query as per the query definition. As per the generated plan, the query is then executed in the appropriate execution engine. At the SQL optimizer stage, it goes to the query optimizer tree and query execution plan. However, in cases where the plan cache entry is found, it does not have to go through the optimizer. It just calls the stored plan and uses the plan to execute the statement. Depending on the plan, it goes to a row store engine or a column store engine.

3 SQL Plan Cache:

SQL statement is compiled to a plan before execution. Once the plan is compiled, it is efficient to use the same plan instead of compiling plans every time. Whenever execution of the statement is requested, HANA checks the SQL plan cache to see if there is a plan already compiled or not. If a plan is found, it is used, otherwise SQL is compiled and the generated plan is cached. The monitoring view **M_SQL_PLAN_CACHE**, has very useful information about plan cache such as plan cache identifiers like USER_NAME, SESSION_USER_NAME, SCHEMA_NAME, STATEMENT_STRING, and

STATEMENT_HASH. Information columns such as EXECUTION_COUNT, PREPARATION_COUNT, LAST_EXECUTION_TIMESTAMP, and LAST_PREPARATION_TIMESTAMP. This information is very useful when a performance issue needs to be analyzed. By looking up the M_SQL_PLAN_CACHE monitoring view, we can find when the plan was compiled and executed and whether the plan is still valid or not. As per the specific statement, user name, or host, a developer can search the plan cache using the wildcard. With the information from M_SQL_PLAN_CACHE, we can also search other useful information in other monitoring views, such as M_EXPENSIVE_STATEMENT or M_EXECUTED_STATMENT.



4 SQL Optimizer:

SQL optimizer is designed to make the optimal plan within a certain amount of time for the execution engines to handle the execution. In SQL optimizer, there are two optimization steps, which are rule-based optimization and Cost-based optimization. In the rule-based optimization step, the predefined, proven rules are applied in order to simplify the plan and lower the costs without conducting cost comparison. There are rules like filter pushdown, JOIN removal, or simplify GROUP BY.

Cost-based query optimization finds candidates from the plan generated in rule-based optimization. The cost of every alternative is calculated and the cheapest one is chosen. The alternatives are found by applying various enumerators such as A_THRU_B or PRE A_BEFORE_B. We can also control enumerators by applying hints such as AGGR_THRU_JOIN, JOIN_THRU_JOIN, and JOIN_THRU_AGGR, and so on. In AGGR_THRU_JOIN, SQL optimizer tries to push down the aggregation through the JOIN so that it can reduce the data set of one of the JOIN candidates. In JOIN_THRU_JOIN it switches the order of the JOIN. PREAGGR_BEFORE_JOIN adds a pre-aggregation to one of the JOIN candidates so that the size of the JOIN can be reduced.

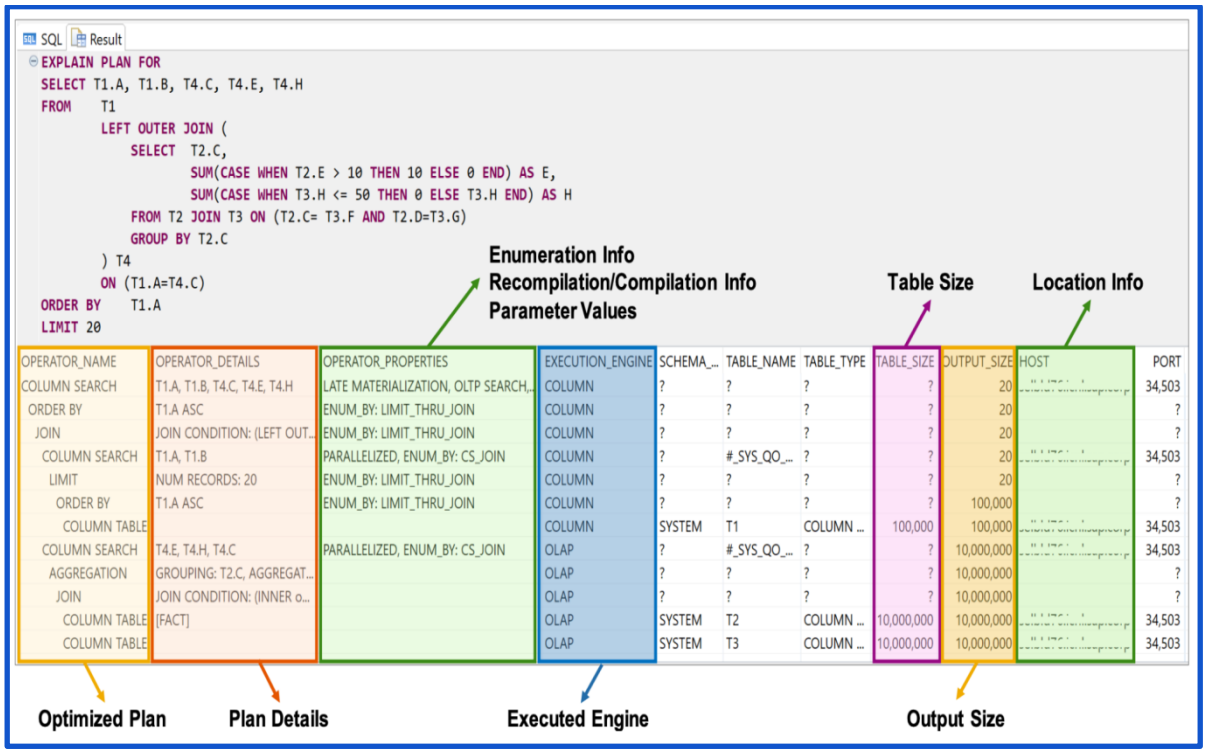
5 SQL Trace:

The SQL trace captures every single SQL statement that enters the database. It is important to find which query comes into the database layer in which order, and which part of it makes it slow. There are two ways to collect the SQL trace. The first one is through HANA studio, and the other one is using SQL commands. Since HANA 2.0 SPS04, the memory consumption can be tracked. The execution information including CPU time and memory size is collected in SQL trace.

6 Explain Plan:

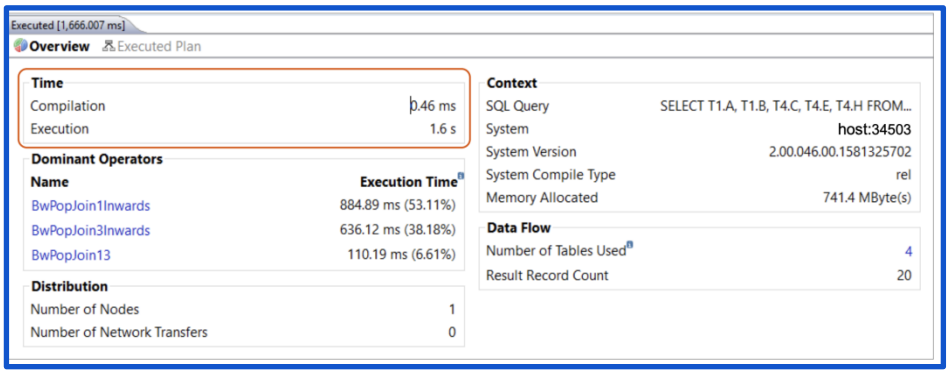
Explain plan shows us a compiled plan in tabular form without executing it. The explain plan creates physical data in a table called EXPLAIN_PLAN_TABLE, selects the data for display, and then deletes everything immediately afterwards because the information does not need to be kept. In order to capture the explain plan in HANA studio, you can select the statement and right-click and choose Explain Plan.

Operator properties contain enumeration information, recompilation information, and parameter values. Firstly, we need to search PLAN_ID for the target query in M_SQL_PLAN_CACHE. After that, we can run the simple SQL commands to see the existing explain plan, which is "EXPLAIN PLAN FOR SQL PLAN CACHE ENTRY <PLAN ID>;".



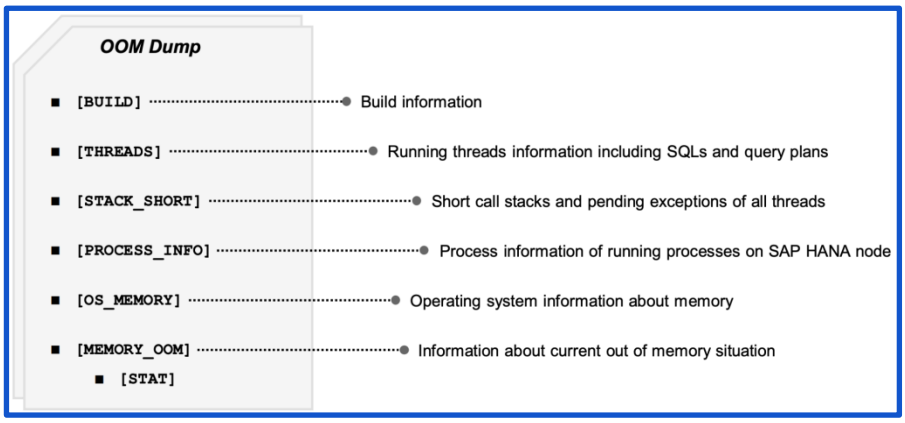
7 Visualized plan:

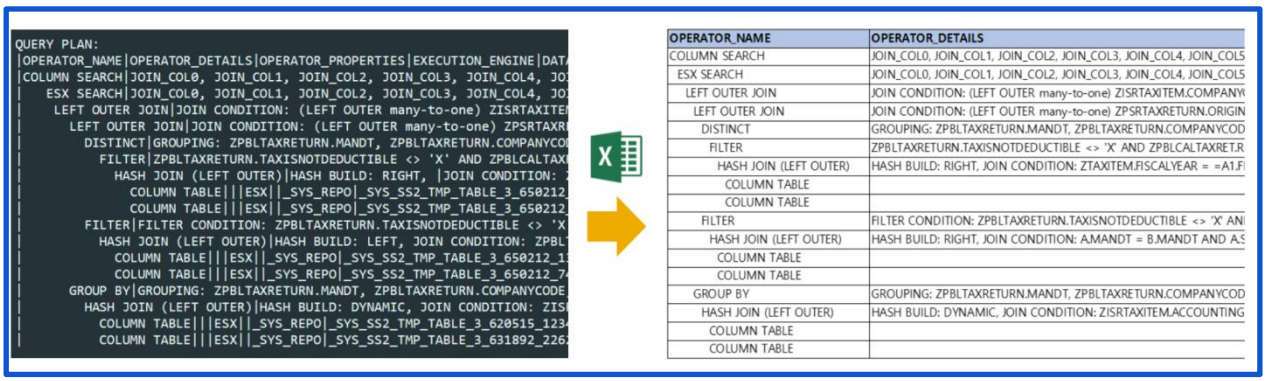
It is often called planviz. It shows us how a query executed, while EXPLAIN shows the execution plan. In order to capture a visualized plan, you need to select the query first, right-click, and choose Visualize Plan then click Execute. The execution time indicates the significance of the issue. The larger execution time indicates that the query needs to be optimized and causes performance issues. Another factor to consider is the compilation time. Most of the performance issues are due to slow execution however there are cases where the performance issue is because of the slow compilation which needs to be fixed. PlanViz also shows the dominant operators among the visualized plan. The Dominant Operators section displays the most expensive top three operators which require large execution time. The Executed Plan tab shows the graphical view of the plan. Understanding data flow is very important in terms of performance issue analysis. Using PlanViz, we can understand the logical structure of the query plan and its data flow in our analysis of the performance of the query.



8 Out of Memory Dumps:

When SAP HANA requires additional memory and is not able to allocate new memory or reclaim memory, then the transaction is aborted with out-of-memory error and out-of-memory dump is generated. We can find all the running threads including SQLs and query plans under the THREAD section. STACK_SHORT shows call stacks and pending exceptions of all threads. you can check the process information under the PROCESS_INFO section. There is the OS_MEMORY section and MEMORY_OOM information as well. Also there is a monitoring view called M_OUT_OF_MEMORY_EVENTS which displays a list of the last 20 out of memory events which can be used for the analysis of multiple OOM dumps. There is a memory statement limit set by the administrator. If the statement consumes more memory then it is terminated. Top "limited composite allocator" lists the allocators consuming the most memory in the current system. We can find the root allocator's connection ID and statement ID. Using this information, we can find the problematic query. Once we find the problematic query that caused OOM, we can also see the explain plan of the query. To see the query plan in a tabular view, we can copy and paste the query plan from the OOM dump into MS Excel.





9 SQL Tuning Tips:

There are some tips which need to be kept in mind to avoid performance issues and create efficient queries.

Tip 1- Record materialization leads to longer execution time and extra memory consumption as well as more CPU consumption. It is important to use proper type casting as much as possible to avoid data materialization. Queries which use type casting of the fields trigger the materialization of records.

Tip 2- Use Partition pruning as much as possible to improve performance by ruling out irrelevant parts in a first step, restricting the amount of data. It is important to set the partitioning criteria in a way that supports the most frequent and expensive queries processed by the system.

Tip 3- In SQLScript, inlining is used to simplify complex queries by removing join operations and condensing several separate queries into a single query. Inlining is used in order to simplify complex queries. If the queries are fully inlined, then many intermediate sequences are merged into one big step. Therefore it is considered as beneficial for most procedures.

Tip 4- Group By clause is responsible for materialization. Hence it should be used carefully or with appropriate HINTS so that it does not create a suboptimal execution plan for the query.

Conclusion:

SAP HANA SQL Query performance optimization is essential to any project. Understanding the key components involved in the query processing and having a detailed understanding of the flow of the query process helps a developer write an efficient query. Also by understanding and utilizing the tools and techniques like SQL plan viz, Explain plan, SQL trace, use of plan cache, OOM dump analysis, SQL tuning tips in the SAP HANA system, a developer can troubleshoot and check query performance and make alterations in the query to make it performance efficient.

References:

1. What is SAP HANA? [Online]. Available at: https://www.ibm.com/topics/sap-hana
2. SAP HANA Installing and administering. SAP TRAINING. [Online]. Available at: https://learning.sap.com/learning-journeys/installing-and-administering-sap-hana
3. SAP HANA Troubleshooting and Performance Analysis Guide [Online]. Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM/bed8c14f9f024763b0777aa72b5436f6/7d28bc8c4e54413caf2716731494da88.html
4. SAP HANA Performance Guide for Developers[Online]. Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM/9de0171a6027400bb3b9bee385222eff/45b03e4d89634e289ce1fcee268bb100.html
5. SAP HANA SQL Reference Guide for SAP HANA Platform [Online] Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM/4fe29514fd584807ac9f2a04f6754767/20ff532c751910148657c32fe3431a9f.html?locale=en-US
6. Visualize a Graphical Query Plan [Online]. Available at: https://developers.sap.com/tutorials/dt-query-processing-part1..html
7. Memory Usage in the SAP HANA Database. SAP Help. [online]. Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM/6b94445c94ae495c83a19646e7c3fd56/bde79b28bb5710149d6eee5e75fe7f17.html
8. What is SAP HANA? Sap.com. [online]. Available at: https://www.sap.com/products/technologyplatform/hana/what-is-sap-hana.html
9. SAP HANA Studio [online]. Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM/52715f71adba4aaeb480d946c742d1f6/ade083aeded84e289d1710a1cf131499.html

10. 12142945 - FAQ: SAP HANA Hints [online]. Available at: https://userapps.support.sap.com/sap/support/knowledge/en/2142945?utm_source=chatgpt.com
11. SAP HANA Troubleshooting and Performance Analysis Guide [online]. Available at: https://sapbasisprep.wordpress.com/wp-content/uploads/2020/09/sap_hana_troubleshooting_and_performance_analysis_guide_en.pdf