Performance Tuning in Teradata

Jasmeet Singh Birgi[1], Mahesh Khaire[2], Sahil Hira[3]
[1]*Teradata Data Analyst*
[2,3]*BI Application Developer*

Abstract- **"Performance Tuning:- Increasing the System's Performance without making any hardware changes." Data is continuously increasing and most systems will respond to increased load with some degree of decreasing performance. A system's ability to accept higher load or to increase maximum load can be scaled up by inducing more nodes or by Improving hardware type, but the Question remains is it really What was required ? The most important question with the business will be if the change we are doing is really justified because adding additional hardware means additional cost to company and if Issue is not of Capacity/ Increasing Available CPU cycles that can be used adding more hardware will only increase the DWH cost for Organization. So the important thing becomes, if we are using the System resources in an Optimal way or something can be done to improve the performance without causing any additional cost to organization. Considering the Importance a lot of work has been done all over the world on DWH performance tuning, In this article we would like to share various approaches or Levels at which Performance Tuning can be done.**

I. INTRODUCTION

First a Brief on various Concepts that were shared earlier: Concept [1] various concepts that can be used to achieve better performance, Establishing and maintaining ongoing communication & training by providing support services, this helps in resolving issues before it becomes a serious problem. Help Desk & Problem Management, Network Management, Capacity Planning, Data Loading Performance, Query Management, Problem Management Process and Development, Software and Hardware should be updated, Extract Transform Load process form one effective system and should accomplish efficient, maintainable and scalable process.

Concept [2] **DWH performance is considered at these stages :** Fetch data from source system, Data processing through ETL Layer, Feeding data in to DWH, Time involved in Fetching data from DWH for reporting.

There are numerous factors that can impact Response time starting from Tool selection to Physical Design. Collecting Statistics is one strong approach that can help us solve most of the performance related issues. It basically provides optimizer with ready to use pre-compiled information on the Data and can be scheduled by DBA/ Architect to ensure smooth flow of queries on the System. This stat information is stored in System tables and format can vary depending on Database/Software type. Proposed solution has steps and they will proceed only if first condition is satisfied.

a) IF Table is Populated
b) Data is Incremental of Truncate load
c) Making Sure Data of Base Table has changes to significant amount.
d) Checking Age of stats and updating only if required.

Using the approach can help us save a significant amount of CPU getting wasted.

Additionally to above research papers, we would like to take a deep dive and focus more on various Levels at which performance tuning can be done in a DWH environment (specifically teradata):

Level 1 - Physical Design Tuning
Level 2 - Query Level Tuning
Level 3 - System or APP Level Tuning
Level 4 - Workload Management

II. PHYSICAL DESIGN TUNING

[3] Physical design tuning deals with the database design of storing business data in a normalized form. The standard design is in a third normal form, which means further breaking of Entity Relationship Diagram such as :-

- Relations become tables
- Attributes becomes columns

- Relationships become data references, For ex, Primary Key and Foreign Key

Database Design for Teradata should be implemented in as follows:-

- De-normalize where appropriate
- Partition tables where appropriate
- Group tables into databases where appropriate.
- Determine use of Segments
- Determine use of devices
- Implement referential integrity of constraints

Normalization: [3]Normalization is the technique, where non-key columns depend in the key. A fully normalized design may not provided with the best performance, Therefore it is recommended to design for the third normal form and then de-normalize it, if performance issue arises.

There are various benefits of Normalization such as:

- Searching, Sorting, Creating Index much faster way, since the tables are small and compact.
- Index Searching is also more efficiently
- There are few index per tables, therefore leads to data modification commands executed in a faster way
- There are few NULL values and less redundant data, making your database much more compact.
- Triggers executed more quickly, if you are not maintaining redundant data.
- DML anomalies are reduced.

In a nutshell, normalization is a cleaner and easier to maintain the database.

Once the database design is created, we can also use the method called de-normalization for improving performance of a specific query or application. There are various performance advantaged of De-normalization:-

- Minimizing need of Joins
- Minimizing foreign keys on tables.
- Minimizing number of Index, which help in saving space and reduces DML queries execution time.
- Pre-computing aggregate values at data modification time, rather than at select time.

There are various de-normalization techniques used:-

- Adding Redundant Columns:- helps in eliminating frequent joins.
- Adding Derived Columns:- helps minimize the use of Joins also will reduce the time needed to produce aggregate values
- Collapsing Table:- If user wants full joined data from two tables, we can simply collapse the table to improve performance by eliminating the join.

III. QUERY LEVEL TUNING

[4]SQL statements are used to retrieve data from traditional RDBMS. Therefore it is necessary to use the set of queries which uses minimal time and System resources as they will help in improving the overall performance. There are various methods which help in achieving query tuning.

- Statistics :- Statistics information can be stored in DWH, which will make sure pre availability of required data to come up with effective execution plan. With this available precompiled information optimizer /database engine will use this information rather than calculating or estimating the same at run time and will also take care in case data is skewed.
- [5][8]Primary Index Choice:- There are two types of Primary Index.
  - Unique Primary Index (UPI)
  - Non Unique Primary Index (NUPI)

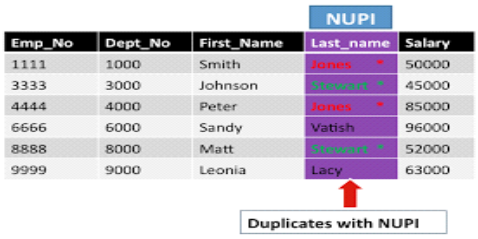
Unique Primary Index is the index which doesn't accepts duplicates values for the particular column, it also accepts NULL values once.

UPI

Emp_No	Dept_No	First_Name	Last_name	Salary
1111	1000	Smith	Jones	50000
3333	3000	Johnson	Stewart	45000
4444	4000	Peter	Jones	85000
6666	6000	Sandy	Vatish	96000
8888	8000	Matt	Stewart	52000
9999	9000	Leonia	Lacy	63000

Teradata Enforces Uniqueness on an UPI

For example in the above picture we are using Emp_No as a Unique Primary Index, therefore it won't be containing any duplicate values.



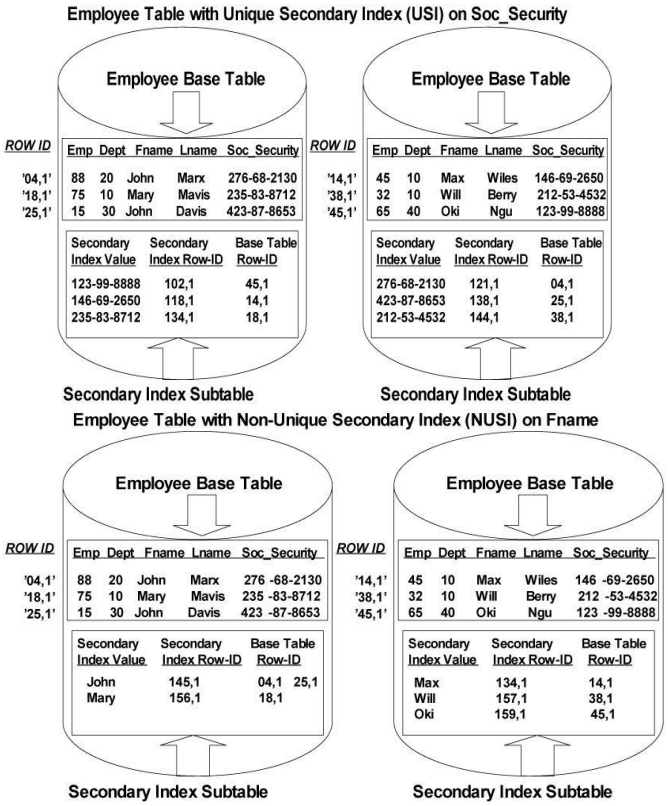
Whereas, Non unique primary Index as the name suggests can contain duplicate values and also accepts NULL values. As the example shows Last Name as the example of Non Unique Primary Index, which can accept duplicates value, As Jones comes twice in the table.

Primary Index is the Index, according to which Teradata distributes data to the AMP and while retrieving, takes data from disc to the particular AMP. Therefore to make an optimized query one should make sure to use a Primary Index column in a where condition and also in joins. Primary Index are created at the time of table creation and can't be created later on, therefore if condition demands of creation of a different primary index then one should make use if a temporary table or a volatile table which has the same structure and content like the original table but a different primary index. Therefore in a nutshell, PI is useful for

- Distribution of Data
- Retrieval of Data
- Join Operations

- [5][11]Teradata Indexing Techniques:- We can also use secondary Index for query tuning in situations where a particular columns is used again and again in a where column. Using Secondary Index increases the spool space and reduces the I/O, which provides and alternate path to the data. There can be around 32 secondary index created on the table, which can be created or dropped at the run time, or even after the table is populated. In SI a sub table is created on each AMP. Similar to Primary Index, there are two types of Secondary Index namely.

  - Unique Secondary Index (USI)
  - Non Unique Secondary Index (NUSI)



The definition of the USI and NUSI are similar to UPI and NUPI, as USI doesn't accepts duplicates, whereas NUSI can accept duplicate values. Another important aspect of Secondary Index is 2 Amp Operation and All Amp Operation. 2-AMP Operations occurs in case of USI, whereas All AMP operation occurs in case of NUSI, It is always suggested to collect stats on NUSI Index.

Above examples show a 2-AMP Operation and FULL AMP Operation.

[12]Another technique would be use of partitioning, which is nothing but another performance enhancement procedure. By using partitioning, we make sure that the table is not a full table scan and only particular part of the AMP is scanned, As it is done on AMP and not on the disk.

It can be done on any column and have to be defined while creating a table. As volume of data of data is increases, we move to partitioning rather than using SI, as it used SPOOL whereas SI uses PERM space. Also PPI (Partition Primary Index) is used, which is nothing but partition applied on a Primary Index column, to avoid full table scan.

AMP 1		
Order Table		
Row Hash	Order Date (PI)	Order Number
1	02-01-2003	99
5	01-01-2003	88
8	03-01-2003	95
9	01-02-2003	6
80	01-05-2003	77
87	02-04-2003	14
98	03-02-2003	17

AMP 2		
Order Table		
Row Hash	Order Date (PI)	Order Number
2	02-02-2003	44
4	01-10-2003	53
12	03-05-2003	16
42	01-06-2003	100
52	03-06-2003	35
55	02-05-2003	15
88	01-22-2003	74

If we have to find out the order number in for 1st month then it will require the full table scan, whereas in the second ex, we see that only that portion of the AMP will be scanned which has the order number from the 1st month. Therefore, it is much more efficient to use partitioning in large tables.

AMP 1		
Order Table		
Row Hash	Order Date (PI)	Order Number
5	01-01-2003	88
9	01-02-2003	6
80	01-05-2003	77
1	02-01-2003	99
87	02-04-2003	14
8	03-01-2003	95
98	03-02-2003	17

AMP 2		
Order Table		
Row Hash	Order Date (PI)	Order Number
4	01-10-2003	53
42	01-06-2003	100
88	01-22-2003	74
2	02-22-2003	44
55	02-05-2003	15
12	03-05-2003	16
52	03-06-2003	35

- Query Rewriting:- It is a process to deal with the making changes in the SQL part which can further improve the performance. It can be done in various ways. For ex:-
  - Using DISTINCT instead of GROUP BY in columns having different values.
  - Union statement should be used for breaking large SQL statement, and can be executed in parallel.
  - Left table should be greater row count than the right table in case of a join.
  - For best results, PI columns of both the tables should be in the join condition. Also those columns should have same data type and size of the data type.
  - Also various other things depending on the business logic, where we can use left join in place of an Inner Join.
- Real Time Monitoring:- Real time monitoring is nothing but making sure that how the query is running on viewpoint or PMON. It can help detect the problems which are causing skew. It can be due to bad join used missing statistics collection therefore it is necessary to have statistic collection on the table. It is of basically three types
  - Low – Done on all the PI Columns
  - Medium – Done on all the after where columns
  - High – Done on table level
- Comparison Of Resource Usage:- It a nothing but measure of resource usage, we can get result such as
  - Total CPU Usage
  - Spool Space needed
  - The LHR (Ratio between CPU and I/O usage)
  - CPU Skew
  - Skew impact on CPU

IV. SYSTEM OR APP LEVEL TUNING

[9]Application level Tuning can help companies save a lot on there system resource and hence money.

All of us have seen queries that perform unnecessary full-table scans or other operations that consume too many system resources contrary to the amount of data they process. Application tuning is a process to identify and tune target such applications for performance improvements and proactively prevent application performance problems.

STEP 1: Identifying Problematic Queries and hence area of Improvement.

Data warehouses can handle millions of queries a day. However a suspect query is one that either consumes too many system resources irrelevant of the amount of data involved or is not taking advantage of Teradata parallelism. While most DBAs are aware of the problem queries that most affect system performance, there are several ways to help prioritize what to tackle first. One way is to analyze Re Usage Data looking for the days or times of the day when the system is running close to or at 100% busy. Amp Usage data can be used to identify a particularly consumptive application or group of users. When it gets down to the real tuning analysis, though, DBQL data is the place to go.

STEP 2: Recording Similar Queries:

DBQL should be used to find specific incidents of problem queries, it can also be used to examine the frequency of a problem query. In this scenario, a DBA might notice that a marketing manager runs a problem query every Monday morning, and the same problem query is run several times a day by various users. Identifying and documenting the frequency of

problem queries offers a more comprehensive view of the queries affecting data warehouse performance and helps prioritize tuning efforts.

STEP 3: Actually Tuning the Query:

Identifying problem queries and recording instances of like queries is easy; the difficulty is analyzing and tuning a specific query. There is lot of expertise, time and attention to details required to tune such queries. There are various techniques that can help us in that:

- Using Query Explain Plan.
- DBQLSTEP table details to pin point the problematic step.

STEP 4: Translating gains to Business values

This is the most important step in System tuning . Basically this is how IT will come to know how valuable improvement is for them.

Determining business value can be broken into calculations and sub-calculations. Check the impact of making a tuning change: Monthly CPU saved = Total old CPU for a month X the average improvement percent.

STEP 5: Implementing and Tracking:

Once We have finalized and implemented the solution we have to document it and track it for couple of days so confirm if it works as per plan.

Following things can be documented:

- Query optimization process
- Options found and tested
- Best option
- Options discarded, and why
- Lists of what still needs testing
- Observations and recommendations
- Anticipated savings

Application tuning focuses on returning capacity to a system by concentrating on query optimization. Through application tuning, database administrators (DBAs) look for queries wreaking havoc on the system and then target and optimize those queries to improve system performance and prevent application performance problems.

V. WORKLOAD MANAGEMENT

[10]Activities performed by the user are nothing but requests and workload is a group of similar requests and to manage those requests in a way so that all users can utilize it to the fullest is known as workload management and this forms an important part of current mixed workload environment. It is much better to keep the request of similar types in a same group so that instead of applying a setting for each request, we can apply setting for the whole group. For ex if a particular type of request is a high priority and need to be resolved in an urgent basis. Therefore instead of applying ad-hoc setting to that one, we can just simply put it in a group which handles all ad-hoc and high priority requests.

Also doing this provides us with the better overview of how system is used and by which group in a particular. We can change the group of the user anytime we want. For example, if a user who generally uses a complex queries and is given a high priority group, we can change the group if we feel, its taking more CPU or because other users are in queue for long time due to this. Also we can create different queues for different group which further helps us in making it more efficient. For example there are two types of users, one who uses complex queries and the other who uses simple select queries, therefore the one with simple queries might have to be queue for long time, as the complex one will take time. So to avoid these kind of cases, we create separate queues for different groups.

There are priority assigned to the groups according to the types of queries they execute. For examples,

- Very High Priority used for tactical queries such as short select, where multiple tables are joined. We used the concept of Join Index in this.
- High Priority used for mini batch, such as daily FASTLOAD, MLOAD.
- Medium Priority used in the strategic Complex queries, such as joins, stored procedure and macros
- Low Priority used in Batch Reports.

This whole workload management is done by priority scheduler tool. It is basically used for sharing of resources:-

- Resource partition is nothing but the groups we create to distinguish between the users and types of queries. Priority scheduler provides zero as default value, and there can be more additional resource partition. It is the resource partition, which carries weight and compared with the other resource partitions.

- The relative weight is also calculated, which is nothing but the total number of resources (in %) consumed by the total user. Relative weight can be calculated by (Individual workload/ total workload)*100
- For example, there are 3 groups of resource partition,
  - RP1 : Weight 10
  - RP2 : Weight 20
  - RP3 : Weight 30
- If total workload is 60. Now, we will calculate the relative weight of each of the resource partition.
  - The relative weight for RP1 would be (10/60)*100 = 16.66%
  - The relative weight for RP2 would be (20/60)*100 = 33.33%
  - The relative weight for RP3 would be (30/60)*100 = 50%
- The relative weight for RP1 would be 16 rather than 17 as Teradata truncates functional values, while calculates the functional weight.
- Performance Groups are defined within each additional resource partitions.
- Performance period is connection between performance group and allocation group, we can have from 1-8 performance period.
- Allocation group weight is compared with the weight of other allocation group, it can also limit the amount of CPU used by sessions under it.

VI. CONCLUSION

Increasing System Capacity is not always a solution to Improve System Performance. If we want to improve our System's performance and Utilize it to the fullest we will have to make sure we follow all the above levels continuously and keep the system tuned as being perfect in one level also does not grantee the System to be tuned.

ACKNOWLEDGMENTS

We would like to take this opportunity to thank & express our special gratitude to **Dr. Amresh Nikam(Sinhgad Institutes)** without which this work would not have been successful.

REFERENCES

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.4483&rep=rep1&type=pdf on oct 6th 2015
[2] http://research.ijcaonline.org/volume32/number1/pxc3875303.pdf on oct 6th 2015
[3] The Teradata Database-Implementation for Performance by Brian R. Marshall
[4] The Teradata Database : Introduction and SQL by Brian Marshall
[5] Teradata Database Index Essentials by Alison M Torres
[6] Teradata Basics - Teradata 14 official Certification Guide
[7] Teradata Architecture and SQL by Tom Coffing and William Coffing
[8] http://www.teradatawiki.net/2013/08/Teradata-Primary-Index.html
[9] http://apps.teradata.com//TDMO/v07n04/FactsandFun/Services/StrikeItRich.aspx
[10] http://docs.aws.amazon.com/redshift/latest/dg/cm-c-implementing-workload-management.html
[11] http://www.teradatatech.com/?p=815
[12] http://www.teradatatech.com/?p=997