TPC-D - The Challenges, Issues and Results

Ramesh Bhashyam
NCR Corporation

Abstract: This paper covers what we at NCR have learned about the TPC-D benchmark as we executed and published our first set of volume points for the Teradata Database. Areas where customers should read the Full Disclosure Report carefully are pointed out as well as the weaknesses in the benchmark relative to real customer applications. The key execution and optimization elements of the Teradata Database and the 5100 WorldMark platform that contribute to our published results are discussed.

The TPC-D benchmark™ has finally come out of the committee room and into the harsh light of published results. As this happens, the people who have to execute the benchmark have a chance to learn all the quirks that the committee never foresaw. They also get to deal with the reality of putting together what are sometimes very large, expensive systems with lots of parts and getting those systems to execute a long running benchmark. The rewards for doing this are many. The marketing potential is obvious for those with winning numbers. However, in the long run, the real winners will be the customers of large Data Warehouse systems as the vendors of the systems improve query optimization, execution and even reliability to meet the challenges of the benchmark.

The decision support environment is complex, widely-varying and difficult to capture in a small set of queries. In spite of this the TPC-D benchmark can be very useful for comparing the performance of database products. It is however necessary to understand a lot more about the benchmark than just the numbers. This paper is an attempt to highlight those interesting items that merit understanding.

The Benchmark

The TPC-D benchmark models a decision support environment through a collection of 17 complex queries on eight separate tables ranging in size from very small (Region) to very large (Lineitem). In order to model the changing environment in a business as it evolves from day to day it specifies two update functions, UF1 and UF2, to insert into and delete rows from the tables. It also specifies other necessary issues such as the transaction ACID properties, and database maintenance options. There are six database Scale Factors (volume points) referred to by @size -- 1 GB, 10 GB, 30GB, 100GB, 300GB, 1000GB. This size represents the size of the actual data not including indexes, temporary space or database overhead.

The Metrics

There are three primary TPC-D metrics - Power that is expressed as QppD@Size, Throughput that is expressed as QthD@size, Price Performance that is expressed as System-Price over QphD@size where QphD is the geometric mean of QppD and QthD.

The Power metric represents the single user query per hour rate for the benchmark. The QppD@Size Power metric is computed as the inverse of the geometric mean adjusted by scale factor of all the 19 run times (17 queries and two update functions). The Throughput metric represents multiple user throughput capacity for the benchmark. The QthD@size throughput metric is computed as the inverse of the arithmetic mean adjusted by scale factor of all the run times across all the streams. The benchmark allows the throughput metric to be published with just a single user i.e. the Power run could be used for computing the throughput metric also.

There are also two secondary metrics - Database Load Time and Total Data Storage/Database Size. The Database Load Time is the time taken to load and ready the data (such as create indexes) before starting the benchmark run. The

Total Data Storage/Database size is a number that is the ratio of the total disk space used in the benchmark system to the TPC-D volume point.

A Brief Comment On The Metrics

Since the Power metric is based on the geometric mean it equally rewards similar percentage reductions for long and short queries alike. This encourages percentage improvements in query times, rather than improvements in the total run time. Thus a benchmark could have a longer total run time but still win on QppD.

The poorly understood Throughput metric rewards efficient concurrent streams execution. Because it is intended to represent multiple decision support queries executing at the same time, it is more representative of how VLDB systems are used. The key issue is the number of streams and the run times for each. Fewer streams should raise questions about the ability of the DBMS to produce performance results in a production environment. If multiple streams are executed and the run times for the queries in the throughput test exceed the times in the power test by more than the number of streams then this highlights concurrency, resource management, scalability, and throughput concerns.

The price/performance metric is an important one since the QppD and QthD metrics can be increased by simply adding more hardware. This is particularly true on the MPP platforms that are typically used for the VLDB volume points.

All of the metrics are comparable only at the same volume point. It is not meaningful to extrapolate results from one volume point to another using the volume ratios. Also looking at any of these metrics in isolation provides a distorted view of the benchmarked product, as the three metrics make up an inter-dependent unit and must be considered together.

Going Behind The Metrics

The benchmark requires both an executive summary report and a full disclosure report. The executive summary includes these metrics as well as the individual query times and the system hardware description and prices. The full disclosure report goes into further detail. It is important to look at these reports in order to understand the performance of a product. The following paragraphs highlight the important points in these reports.

1. Individual Run Times. The individual query times provide insight into the efficiency of specific combinations of relational operations such as joins, aggregations and scans. In order to gauge the suitability of a product for an application one must look at these individual query times and not just the mean numbers.

2. Index Structures. The quantity, complexity and composition of indexes affect the extent to which the base tables are accessed. For example a covered index that includes all the columns in a table that are needed by a query offers a bypass to the base table access. This type of index can be effective if every query is known ahead of time. Extensive indexing, while inflating benchmark numbers has the risk of increasing data maintenance overhead, as well as disk space requirements, especially since the index size could become as large as the table itself. It could also indicate a product's inability to deal with ad-hoc queries at VLDB volume points. The full report lists all the indexes used, and the columns in the index.

3. Data Ratio. From the total disk usage details in the full disclosure report, a breakdown can be derived for the storage in GB used for different purposes such as the storage for the base tables, storage for indexes, storage for spools, storage for logs, unused storage, and the protection mechanism employed. These numbers can be used to predict the extent to which the configuration mirrors reality. A large amount of configured storage may reflect an inability to efficiently store large volumes of data or it may indicate an attempt to offset IO performance deficiencies.

4. Number Of Streams. The number of streams reported is only one part of the concurrency story. It is important to compare the reported single user and multi-user run times for consistency and scaleable performance. Ideally the query run time should not increase by a factor that is any more than the number of users. If large unevenness exists in a controlled publicized benchmark then the products' effectiveness for an ad-hoc environment is brought into question. If the published results use a single stream for the throughput metric, the ability of the product to perform equivalently when faced with many

simultaneous queries in a real Data Warehouse environment should be carefully examined.

5. Level Of Data Protection. TPC-D does not require data protection – only proof of data recoverability against a 100 MB test database. However, the mechanisms of data protection affect recoverability as well as the availability and integrity of the data. On-line redundancy methods such as mirroring and RAID storage are much more realistic for VLDB systems than backup/restore/roll-forward mechanisms which though much cheaper, provide an unacceptable level of availability.

6. Database Maintenance. The repeatability requirement of the TPC-D benchmark requires at least two runs. It allows two ways to proceed from one run to the next - evolve the database or reset the database. The evolve method reflects a maturing database as the changes that have occurred during a run are available at the next run reflecting the reality of a production system. The reset option allows the database to be reset to the original state after each run thus reversing the inserts and deletes that have occurred during that run. The full disclosure report states what choice was used.

7. Data Load Time. At VLDB volume points the data load time and the loading mechanisms are important pieces of information. This is an area that customers tend to overlook until they start to load a system with real data.

8. SQL Syntax. A product's ability to execute SQL-as-written in the benchmark is important in large data warehousing applications. Because applications rely on SQL generated by tools or written by non-database personnel, the SQL cannot be optimized for a particular DBMS without an experienced data base administrator reviewing and rewriting each query. This may be unfeasible or even impossible.

We believe that the benchmark could be improved in three ways:

1. The throughput information is incomplete. A system's ability to execute multiple concurrent streams is an especially important aspect for VLDB environments. At the larger volume points the benchmark should require at least two or three concurrent streams for the throughput metric.

2. The recoverability test could be strengthened. The benchmark should require the recoverability test to be performed on the full volume point database. The current test only requires this on a 100 MB sample which allows for the use of techniques that are unsuitable for VLDB such as full restore, roll-forward. The length of time for the recovery process should also be disclosed.

3. Full end-to-end load times and the associated storage should be disclosed. The load times as currently defined allow the vendors to exclude data preparation time and space used. An example of this is splitting of the data into multiple partitions prior to loading it into the DBMS.

Key Teradata DBS Features

This section discusses the key execution and optimization elements of the Teradata Database and the 5100 WorldMark platform that contribute to our published results. Note however that this is not a complete description of the 5100 platform or the Teradata DBS.

Fully Parallel DBS execution

There are multiple dimensions of parallelism in the Teradata DBS, which all contribute to the benchmark performance, especially at the higher TPC-D volume points.

The platform provides both CPU and IO parallelism. The 5100 platform is a collection of SMP nodes. Each node has its own collection of disks and communicates with other nodes over an interconnect. Increasing the number of nodes linearly increases the CPU, IO, and interconnect capacities.

The Virtual Processes (VPROCs) provide the mechanism for controlling the CPU/IO balance within a node. The VPROC is the granularity at which the data is partitioned. Therefore for IO intensive workloads (such as TPC-D on the 5100/Teradata DBS) increasing the number of VPROCs increases the number of parallel IO threads executing on a node. This improves both the node and the overall platform resource utilization.

In addition to the platform parallelism there are three major elements of software derived parallelism - data parallelism, intra-operator parallelism, and inter-operator parallelism. The data parallelism is enabled by hash-partitioning the data across all nodes and VPROCs in the system. This data partitioning trivially makes all relational operations such as table scans, index scans, projections, selections, joins, aggregations, and sorts execute in parallel across all the nodes. Each operation is performed on a partition independently of the other partitions.

The Teradata DBS provides significant intra-operator parallelism. The optimizer splits an SQL query into a small collection of high level data base operations called steps and dispatches them for execution to the nodes in the system. A step can be simple, such as "scan a table and spool the result" or complex, such as "scan tables with row qualifications, join the tables, redistribute the join result on specified columns, sort the redistributed rows and spool the result". The complex step specifies multiple relational operations which are processed in parallel by pipelining. This dynamic execution technique, in which a step spawns off other processes to perform portions of the step in parallel, is key to increasing the database parallelism. The relational-operator mix of a step is carefully chosen to avoid stalls within the pipeline.

Inter-operator parallelism is enabled by executing multiple "steps" of a query simultaneously. One or more processes are invoked for each step on each partition to perform the actual data base operation. Multiple steps for the same query can be executing at the same time to the extent that they are not dependent on results of previous steps.

Throughput

Throughput is critical for any decision support system. In these systems each user causes long running complex queries that consume lots of resources for long periods of time. In order to support multiple simultaneous users high throughput is very important.

The Teradata DBS is designed to maximize throughput. Maximizing throughput requires the system to utilize a large percentage of each resource. In order to operate near the resource limits without exhausting any of them and without causing the system to thrash, requires effective work flow management. The Teradata DBS system provides work flow management by monitoring the utilization of critical resources (such as CPU, memory, interconnect etc.). If any of these reach a threshold value it triggers the throttling of message delivery. This throttle controls the amount and type of work performed in the system since messages are the unit of work delivery. Message headers contain the user priority of the originating request, the amount of work that this message represents in a rough sense, and the spawn level of the message. Spawned messages are always given higher priority over new work in order to quickly complete work already in progress and free up resources. The message subsystem in conjunction with the interconnect software allows the throttling information to be passed upstream. This has the effect of controlling message generation at the origin.

Another way of increasing throughput is to maximize the value of each piece of work performed. One way that all database systems achieve this is by caching disk data blocks to avoid re-reading them. The Teradata DBS improves the effectiveness of this caching for large table scans through the use of a feature called "synchronized scans". "Synchronized scans" permits new table scans to begin at the current point of an already running scan of the same table resulting in the avoidance of IO for the subsequent scans. Since these scans may not progress at the same rate, any task is free to initiate the next data read.

The Teradata DBS also maximizes the value of each piece of work by reusing spools within the same query. The spools may be the result of complex join processing. Reusing these spools avoids redoing expensive work such as scans, row re-distributions and joins.

Query Optimizer

The cost-based optimizer is parallel aware and optimizes the execution cost across all three cost dimensions - CPU, IO, and interconnect. It picks the plan with the lowest overall cost for execution. This section highlights some of the key features that contribute to the performance.

Enhanced evaluation (EEVL). The EEVL is a feature that compiles evaluation clauses, and row building operations. These are popular operations used in row qualifications, column aggregations, and group by operations. The EEVL advantages add up when millions of rows are processed for a query.

Efficient Join Plans. The join planner first attempts to find the best join plan by using a shallow search (looking ahead up to two join steps). If the estimated cost of that plan exceeds a certain threshold (specified in number of minutes) and the estimated number of rows in one or more of the tables being joined exceeds a certain threshold (specified in millions of rows), then the join planner attempts to find a better join plan using a deeper search (looking ahead up to five join steps). The intent of this heuristic is to avoid incurring the cost of the deeper search when it is not likely to pay off. If the user specifies a particular join search depth, then the join planner attempts to find the best join plan using a single search, looking ahead the specified number of join steps.

Global Query Optimization: The optimizer merges all the query blocks and globally optimizes the entire query based on cost in a single pass. This technique is useful when there are multiple blocks in the query as in correlated sub-queries, and nested sub-queries.

This merging of query blocks enable a number of high performance features. It enables innovations over heuristic optimization techniques (such as Magic Sets and Predicate Movearound) by integrating them tightly with cost based optimization techniques.

The optimizer chooses rewrite transformations based solely on cost and not as a general heuristic that may or may not provide performance benefit. It combines join planning with rewriting. Magic sets are introduced as another constraint on table joins and the costing function costs the joins as it would any other join. This integrated approach is better because it applies Magic sets to only those portions of the query that benefit from Magic sets and applies other techniques elsewhere.

The global optimization technique also enables Predicate and Aggregation Movearound optimizations. These techniques push down or pull up aggregation operators and predicates.

The single pass global optimization also improves the optimization cost. This benefit will be pronounced for queries that have multiple nesting levels where the optimization cost is improved by not having to consider the various combinations of Magic sets that a purely heuristic approach might require.

The global optimization also results in significant complex sub-expression elimination. Redundant computations are eliminated by reusing spools. These spools could be the result of expensive join/redistribution operations.

Conclusion

The decision support area is complex and the TPC-D benchmark query suite is a good attempt at capturing this complexity. It acknowledges this in its specification of 17 queries and two update functions. Because of this complexity, it is necessary to look beyond the primary metric numbers to the details of the performance, the database and the environment in which it was executed. In fact a proper evaluation of a product will not be complete without looking at these details and evaluating the degree to which a published instance of the TPC-D benchmark applies to the customer's application and workload.

Throughput is more important than response time for decision support systems especially at VLDB volume points. We have pointed out some of the key throughput and optimizer enablers in the Teradata DBS that contribute to its superior performance.