

The Teradata Scalability Story



Contributor:
Carrie Ballinger, Senior Technical
Consultant, Teradata Development
Teradata, a division of NCR

A Teradata White Paper

Teradata
a division of  NCR

The Teradata Scalability Story

table of contents	
Section	Page
Introduction	2
Scalability from the Data Warehouse Perspective . . .	3
Teradata Characteristics that Support Linear Scalability . .	4
Linear Scalability as Data Volume Grows	6
Linear Scalability as Concurrent Users are Increased	7
Examples of Scalability with Hardware Expansion	11
Conclusion	13

Introduction

“Scalability” is a widely used and often misunderstood term. This paper explains scalability as experienced within a data warehouse and points to several unique characteristics of the Teradata® Database that allow scalability to be achieved. We share examples taken from client benchmarks to illustrate and document Teradata’s inherently scalable nature.

While scalability may be considered and observed in many different contexts, including the growing complexity of requests being presented to the database, this paper limits the discussion of scalability to these three dimensions: Data volume, concurrency and processing power.

Uncertainty is Expensive

Because end users care mainly about query response time, an ongoing battle is raging in every data warehouse site. That is the battle between change and users’ desire for stable query times in the face of change.

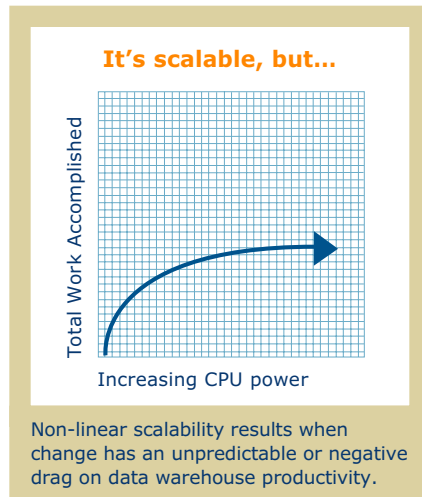
- If you add users, what will happen to query times?
- If the volume of data grows, how much longer will queries take?
- If you expand the hardware, how much relief will you experience?

Uncertainty is expensive. Being unable to predict the effect of change often leads to costly over-configurations with plenty of unused capacity. Even worse, uncertainty can lead to under-configured systems that can’t deliver on the performance required of them, often leading to expensive re-designs and iterations of the project.

A “Scalable” System May Not Be Enough

Many vendors and industry professionals apply *scalability* in a very general sense. To many, a scalable system is one that can:

- Physically accommodate more data than currently is in place
- Allow more than one user to be active at the same time
- Offer some hardware expansion, but with an unspecified effect on query times



With this loose definition, all you’re being told is that the platform under discussion is capable of some growth and expansion. While this is good, what usually isn’t explained is the impact of this growth. When scalability is not provided, growth and expansion often lead, at best case, to unpredictable query times, while at worst, to serious degradation or system failure.

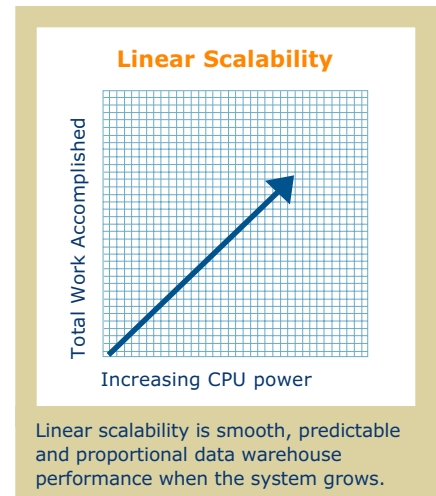
The Scalability That You Need

A more precise definition of scalability is familiar to Teradata users:

- Data growth has a predictable and knowable impact on query times
- Overall throughput is sustained while adding additional users
- Expansion of processing power leads to proportional performance increases

Linear Scalability in a data warehouse is the ability to deliver performance that is proportional to:

- Growth in the data volume
- Changes in the configuration
- Increase in the number of active clients



The Teradata Scalability Story

Scalability from the Data Warehouse Perspective

Decision support (DS) queries differ in the type of scalable performance challenges they are faced with compared with OLTP transactions. OLTP transactions access very few rows, and the work is done in a very localized area, usually involving one or a small number of parallel units. On the other hand, the work involved in a DS query is done everywhere, across all parallel units, and can absorb all available resources on the platform. Such a query reads comparatively more data, often scanning all rows of a table, and performs significantly more complex work.

Volume Growth:

OLTP: Even as the number of rows in a table increases and the data volume grows, an operational transaction tends to perform the same amount of work and usually touches a similar number of rows. Under such conditions, data volume growth has little or no effect on the response time of such a request.

DSS: However, the work required by a decision support query is directly impacted by the size of the tables being acted on. This is because the operations involved in delivering a complex answer, even if the set of rows returned is small, often require reading, sorting, aggregating and joining large amounts of data. And more data in the tables involved means more work to determine the answer.

Usage Growth:

OLTP: These transactions are localized and small, requiring comparatively small amounts of resources. Because of this, additional OLTP transactions can be added with little impact on the response time of work already in place, and with the potential to raise the total throughput of the system.

DSS: In a parallel DS system, where a single user may use almost all system resources, adding more users usually results in longer response times for all users. Since maximum throughput may already have been achieved with a low number of decision support users on the system, what you must look for when you add DS users is **no degradation of the total system throughput.**

Hardware Growth:

OLTP: In a transaction application, adding processing nodes would tend to increase system throughput, resulting in an increase in transactions completed, but may have little or no effect on individual timings. Because each transaction is localized, each one may have the same resources available even when the whole system has grown.

DSS: Because DS queries are parallelized across all available hardware, adding nodes to an MPP system can be expected to reduce the query time proportionally.

Why Data Warehouse Scalability Measurements May Appear Imprecise

Linear scalability in the database can be observed and measured. It is fact-based and scientific. The primary foundation of linear scalability is the hardware and software that is underlying the end user queries. For scalability to be observed and assessed, however, it is usually helpful to keep everything in the environment stable, including the database design, the degree of

parallelism on each node, and the software release, with only one variable, such as volume, undergoing change at a time.

Two secondary components can contribute to and influence the outcome of linear performance testing: 1) Characteristics of the queries themselves, and 2) The data on which the queries are operating. When looking at results from scalability testing, the actual performance numbers may appear better than expected at some times, and worse than expected at other times. This fluctuation is common whether or not the platform can support linear scalability. This potential for inexactitude in scalability measurements comes from real-world irregularities, irregularities that lead to varying query response times. Some of these include:

- Uneven distribution of data values in the columns used for selection or joins
- Variance in the frequency of cross-entity relationships
- Uneven patterns of growth within the data
- Brute force extractions of test data
- Unequal use of input variables in the test queries
- Use of queries that favor non-linear operations

For this reason, a clearer picture of scalable potential often emerges if multiple points of comparison can be made. For example, when examining the effect of multiple users on query response time, consider not just one user against 20 users, but expand the comparison to include 30 and 40 users as well. A good rule of thumb is to benchmark with the conditions, such as users, that you expect to support today, and then increase that number and test the system again to account for growth.

The Teradata Scalability Story

Teradata Characteristics that Support Linear Scalability

The MPP platform is designed around the shared nothing model. This is useful for expanding systems because when hardware components, such as disk or memory, are shared system-wide, there is an extra tax paid, an overhead to manage and coordinate contention for these components. This overhead can place a limit on scalable performance when growth occurs or stress is placed on the system.

Because a shared nothing configuration can minimize or eliminate the interference and overhead of resource sharing, the balance of disk, interconnect traffic, memory power and processor strength can be maintained. Because the Teradata Database is designed around a shared nothing model as well, the software can scale linearly with the hardware.

What is a Shared Nothing Database?

In a hardware configuration, it's easy to imagine what a shared nothing model means: Modular blocks of processing power are incrementally added, avoiding the overhead associated with sharing components. Although less easy to visualize, database software can be designed to

follow the same approach. As in Teradata's case, it can be architected to grow at all points and use a foundation of self-contained, parallel processing units.

The Teradata Database relies on techniques such as these to enable a shared nothing model.

Teradata parallelism based on shared nothing

Parallelism is built deep into the Teradata solution, with each parallel unit, known as a virtual processor (VPROC), acting as a self-contained mini-database management system (DBMS). Each VPROC owns and manages the rows assigned to its care, including manipulating data, sorting, locking, journaling, indexing, loading and backup and recovery functions. This local autonomy eliminates extra CPUs often required by other parallel database systems when these database-specific functions must be consciously coordinated, split up, or congregated to take advantage of parallelism. Teradata's shared-nothing units of parallelism, because they reduce the system-wide effort of determining how to divide the work for most basic functionality, set a foundation for scalable data warehouse performance that is not found in most other databases.

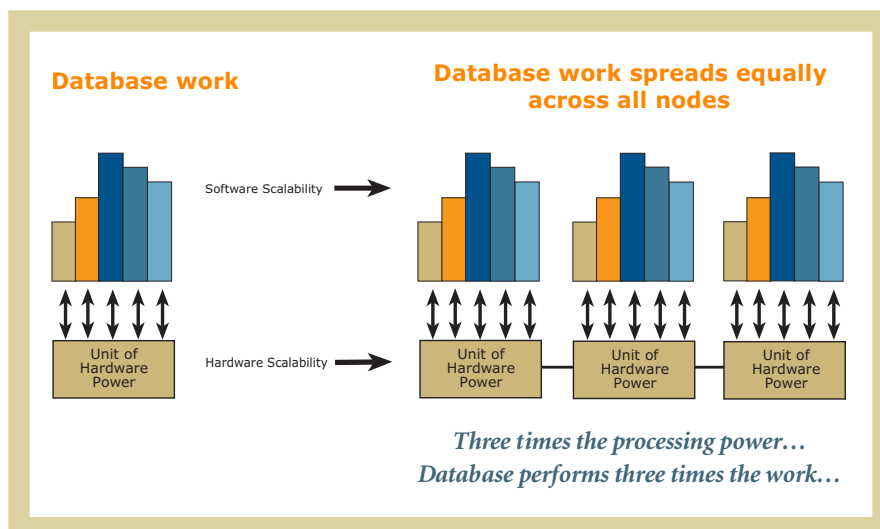
No single points of control

Just as all hardware components in a shared nothing configuration can grow, all Teradata processing points, interfaces, gateways and communication paths can increase without compromise during growth. Contrary to database systems that have added parallelism into their product after the fact, Teradata manages and processes data in parallel at the lowest level, the VPROC itself. Additional VPROCs can be easily added to a configuration for greater dispersal of data and decentralization of control, with or without additional hardware.

In addition, each node in a Teradata system can have 1 or more parsing engines, which support and manage users and external connections to the system and perform query optimization, allowing these functions to increase as needed. Databases based on non-parallel architectures often have a single optimization point that can hold up work in a busy system. Teradata's data dictionary rows are spread evenly across all VPROCs (units of parallelism) in the system, reducing contention on this important resource. Parsing engines perform a function essential to maintaining scalability: Balancing the workload across all hardware nodes and ensuring unlimited, even growth when communications to the outside world must expand.

Control over work flow from the bottom up

Too much work can flood the best of databases. Often other database systems control the amount of work active in the system at the top, by using a coordinator or a single control process. This coordinator can not only become a bottleneck, but has a universal impact on all parts of the system and can lead to a lag between the freeing up of resources at the lower levels and their immediate use.



The Teradata Scalability Story

Teradata can operate near the resource limits without exhausting any of them by applying control over the flow of work at the lowest possible level in the system. Each parallel unit monitors its own utilization of critical resources, and if any of these reaches a threshold value, further message delivery is throttled at that location, allowing work already underway to complete. With the control at the lowest level, freed up resources can be immediately put to work, and only the part of the system that is at that point in time over-worked seeks temporary relief, with the least possible impact on other components in the system.

Parallel databases performing data warehouse queries can face unexpected challenges in keeping the flow of work moving, especially among multiple users. This is because parallel functions can easily saturate the system, can become difficult to reign in and coordinate, and can produce their own severe type of contention. Traditional top-down approaches to managing the work flow often fall short, yet many of today's commercial databases, designed from a transaction processing perspective, are locked into a flow control design that puts decision support scalability at risk.

Conservative Use of the Interconnect

Poor use of any component can lead quickly to non-linear scalability. Teradata relies on the BYNET™ interconnect for delivering messages, moving data, collecting results and coordinating work among VPROCs in the system. At the same time, care is taken to keep the interconnect

bandwidth usage as low as possible. VPROC-based locking and journaling keeps activities, such as locking, local to the node. Without Teradata's unique *unit of parallelism with responsibility* concept, these types of housekeeping tasks would, out of necessity, involve an extra layer of system-wide communication, messaging and data movement – overhead that impacts the interconnect and CPU usage as well.

All aggregations done by Teradata are performed on the smallest possible set of subtotals within the local VPROC before a grand total is performed. The optimizer tries to conserve BYNET traffic when query plans are developed, hence avoiding joins, whenever possible, that redistribute large amounts of data from one node to another.

When BYNET communication is required, Teradata conserves this resource by selecting and projecting data from relational tables as early in the query plan as possible. This keeps temporary spool files that may need to pass across the interconnect small. Sophisticated buffering techniques ensure that data that must cross the network is bundled efficiently.

Synchronizing Parallel Activity

When responding to an SQL request, Teradata breaks down the request into independent steps. Each step, which usually involves multiple SQL operations, is broadcast across the interconnect to the VPROCs in the system and is worked on independently and in parallel. Only after the query step is completed by all participating VPROCs will the next step be dispatched, ensuring that all parallel units in Teradata are working on the same pieces of work at the same time.

Because Teradata works in these macro units, the effort of synchronizing this work only comes at the end of each big step. All parallel units have an awareness of working as a team, which allows some work to be shared among the units. Coordination between parallel units relies on tight integration and a definition of parallelism that is built deeply into the database.

Better Than Linear Queries

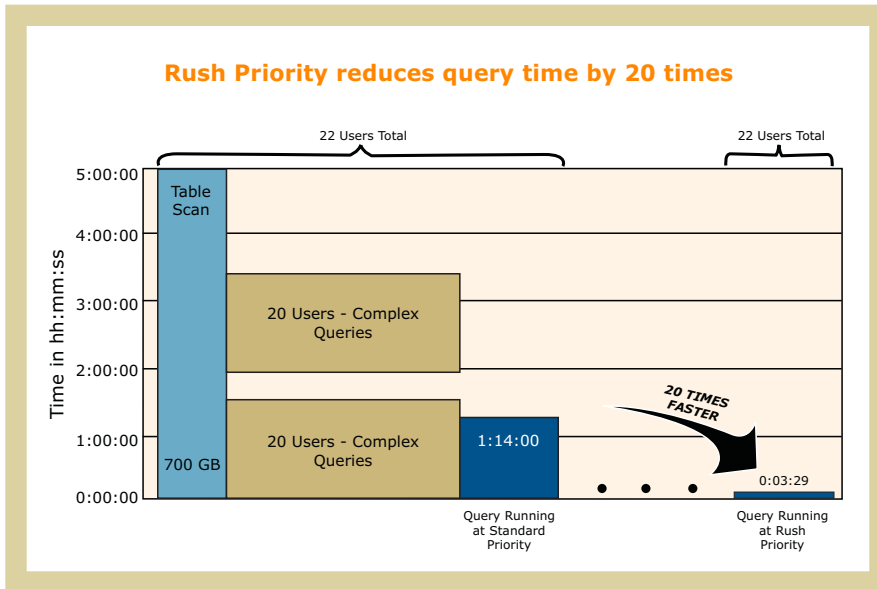
Teradata has the potential to deliver better than linear performance on some queries. Database techniques designed to support this include:

Synchronized table scan: allows multiple scans of the same table to share physical data blocks, and which has been extended to include synchronized merge joins.

Self-tuning caching mechanism: designed around the needs of decision support queries, which automatically adjusts to changes in the workload and in the tables being accessed.

Priority Scheduler: which allows the database to do different types of work at differing priorities. By applying higher priorities to critical pieces of work, the DBA can ensure fast response times even while increases in data volume or growth in the number of concurrent users are causing longer response times of other work in the system.

The Teradata Scalability Story



An example of the power of the priority scheduler, taken from a client benchmark, is illustrated above. In this test, the same query was executed against an identical workload of 21 streams: 1 table scan of a 700 GB table, and two iterations of 20 concurrent complex query streams executing against 1 TB of data. The query being tested was executed first with the standard priority, and then with a rush priority. After the priority had been raised, the query's response time dropped by a factor of 20 while running against the same heavy background work.

NOTE: The Priority Scheduler was not used to speed up query times in any of the benchmark examples illustrated in the following sections of this paper.

Linear Scalability as Data Volume Grows

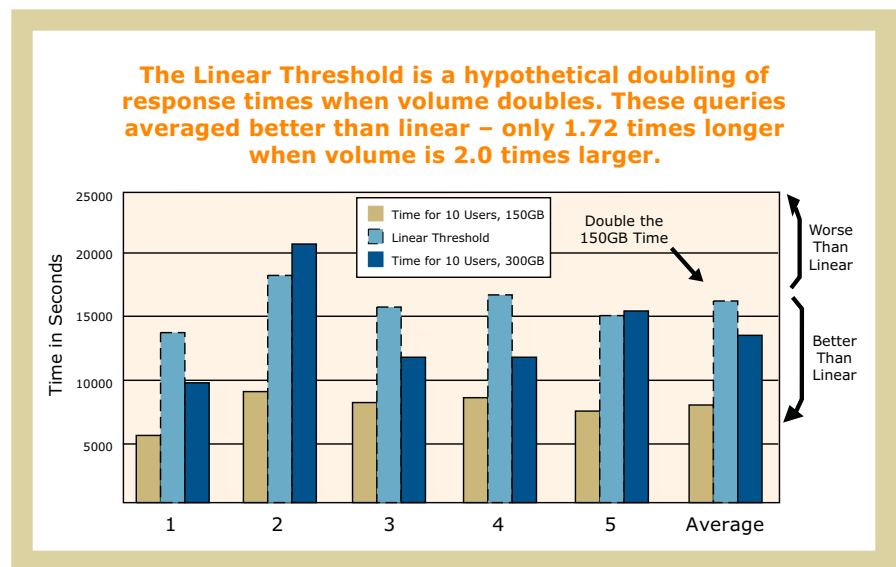
Turning to benchmark illustrations, the impact of increased data volume on query times will be considered first. In the first benchmark example, the test goal was to determine how much longer 50 queries (10 users running 5 queries each) would take as the database volume doubled. The expectation was for the 300 GB query

times, on average, to double (or less than double) the 150 GB times. This expectation was met.

The 10 concurrent users each executed the 5 queries in random sequences, first against a database of 150 GB, and then against a database double that size, 300 GB, using an application from the health care industry.

In the 300 GB database, a fact (detail data) table of 3 billion rows represented information from 2 years of patient transactions, making up the bulk of the database. In addition to 8 small dimension tables, there was a 7 million row patient table. No summary tables were used and only two secondary indexes were created on this large fact table. To produce the smaller volume database, the only change made was to drop half of the large patient transaction table, which made up the bulk of the data volume and was the central table in all the queries.

These benchmark queries performed complex operations favoring access by means of full table scans, and including joins of all 10 tables. Tests were executed on a 4-node Teradata system with 2 GB of memory per node. The following chart illustrates the 150 GB time, the hypothetical linear threshold, which is the 150 GB time doubled (this is the upper limit of linear scalability), and the actual reported time at 300 GB, which on an average is better than (below) the linear threshold.



The Teradata Scalability Story

	10-Users 150GB	10-Users 300GB	Calculation 300GB / 150GB	Ratio (2.0 if Linear)
1	6,775	9,750	9,750 / 6,775	1.44
2	8,901	20,277	20,277 / 8,901	2.28
3	7,797	11,901	11,901 / 7,797	1.53
4	8,357	11,265	11,265 / 8,357	1.35
5	7,500	15,240	15,240 / 7,500	2.03
Average	7,866	13,687	13,687 / 7,866	1.72

NOTE: Less than 2.0 is better than linear.

The table above shows the average response time in seconds for each of the 5 queries as executed by the 10 concurrent users, as they ran at both the 150 GB volume and the 300 GB. The column labeled **Ratio** is calculated by dividing the 300 GB time by the 150 GB time, with a ratio of 2.0 representing “perfectly” linear performance and a ratio less than 2.0 being better than linear. Better than linear performance would be established if data volume doubled, but query times were less than doubled. Some of the queries (1, 3, 4) performed faster than linear as volume increased, and others (2, 5) performed slightly slower than linear.

Overall, Teradata delivered better-than-linear performance with this benchmark application as the volume doubled. This is shown by the average ratio for all queries of 1.72, a ratio that represents less than a doubling of average query response times.

In the chart above, if queries had shown a consistent ratio that greatly exceeded the linearity ratio of 2.0, this would raise a

concern about the scalability of the platform being tested. Query times that are noticeably worse than linear indicate that the system is doing disproportionately more work or has reached a bottleneck as volume has grown. If no convincing explanation for this non-linear behavior emerges from an examination of the queries and the data, then it is advisable to increase the volume again and assess scalability from a second, higher point.

Linear Scalability as Concurrent Users are Increased

This section examines the impact on query times as users are added. All database setup, queries, execution priorities, table sizes and hardware details of the individual benchmark tests remain constant, with the number of active system users being the only variable that changes.

A decision support query executing on a Teradata system has access to almost all available resources when running stand-alone. When another user is active in the system, the first query’s execution time will increase, as that query now has to share total system resources with another user whose demands may be just as great.

For example, if the first takes 5 minutes to run as the only active query, it could take up to 10 minutes to run with one other active system user. This is true because it may have only half of the system resources available, half being used by the second query.

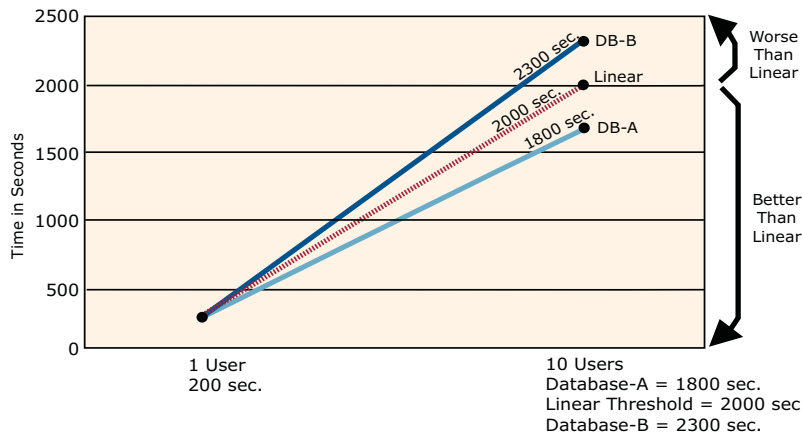
Linear scalability as users are added is established if each query’s response time lengthens proportionally to the increase in concurrency, or less than proportionally. When looking for linear scalability with concurrency test results, it’s often useful to first hypothesize what the upper limit of the query response time should be for that query’s performance to qualify as linear. Then compare that hypothetical threshold against the recorded response time when multiple users are executing.

In the fabricated example that follows (see top of next page), the single user runs a query in 200 seconds, while 10 users report an execution time of 1800, each running the same query. That 1800 seconds is less than 2000 seconds, the hypothetical time it would have taken the one user to perform his work back-to-back 10 times. If the 10 users had taken 2300 seconds, that performance would be worse than linear.

When individual query times increase proportionally or less than proportionally to the growth in the number of users, the system is said to be maintaining a constant or growing system throughput. What is often referred to as negative scale-up

The Teradata Scalability Story

A HYPOTHETICAL EXAMPLE: Database-A achieves better than linear scalability, while Database-B achieves worse than linear scalability.



The following table shows how the better-than-linear scalability was identified. All 8 queries executed in significantly less time than the linear threshold going from 1 to 10 concurrent users, with everything else in the system unchanged. While there was a 10-fold increase in the number of active users, the query response times only increased 2.4 to 5.4 times what they were with one user active. The longest-running query (Query A), for example, only runs about 2 1/2 times longer in the face of 10 times the work being placed on the system. This demonstrates very positive scale-up as users are added.

	Single user time	Time with 10 users active	With 10 users active, query is longer by
G	86	467	5.4 times
D	204	733	3.6 times
F	290	1304	4.5 times
C	558	1913	3.4 times
E	648	2470	3.8 times
B	1710	5107	3.0 times
H	2092	5199	2.5 times
A	2127	5171	2.4 times

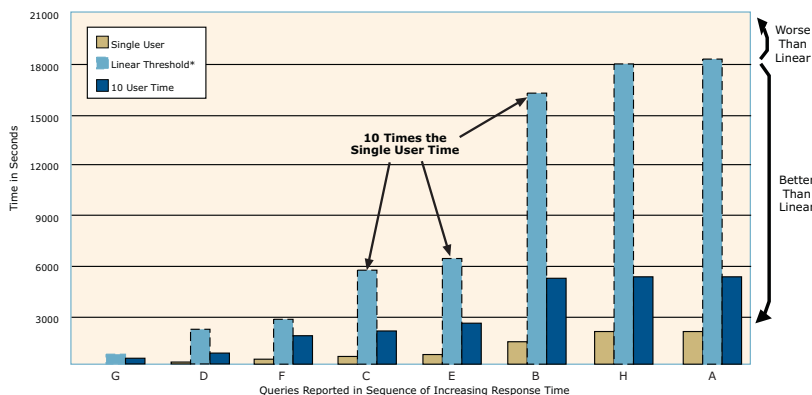
results if the system throughput degrades as users are added and the queries, overall, become disproportionately longer.

A client benchmark comparing performance with 1 against 10 users

The following example of positive multi-user scalability demonstrates how Teradata increases total system throughput with 10 active system users, compared to a single

user. Each of 8 queries (A through H) was run stand-alone by one user and then with 10 concurrent users, with each of the 10 users executing the same 8 queries in differing sequences and with different selection criteria. In all cases, the time it took Teradata to complete the 10 concurrent runs was well below 10 times the single-user execution time, or the threshold of linear scalability.

10-User Execution Times Show Better Than Linear Scalability



Time is in seconds.

The test machine for this benchmark was a 1 node server with 4 CPUs and 2 GB memory. The SQL was generated using BusinessObjects; it was run against Teradata using the BTEQ utility. The raw data measured 15 GB, and was designed using a star schema composed of one large fact table and 10 dimension tables. Data was loaded directly from MVS using Teradata's FastLoad utility. In a benchmark

The Teradata Scalability Story

report, the client commented: “This eliminates the need for an area to stage the data. It can be loaded from MVS tapes directly into the Teradata Database.”

Concurrency comparison of 1, 3, 5, and 7 streams

The next benchmark, from a telecommunications company, looks at concurrency from a slightly different perspective. In the previous example, each of the multiple users did the same number of queries as the single user. The comparison being made was between the time for one user to perform that work and the time for 10 users to perform 10 times that work.

In this benchmark, what is being compared is the time for one user to do a set of queries, and then the time for 3 users, 5 users, and 7 users to share the same number of queries (without repeating them) among themselves. In this benchmark, the total query effort remains the same for one user and for multiple users, and the focus is on total time to finish the work, rather than individual query scalability.

A special script was written that understood how and when to launch queries for the 3, 5 and 7 concurrent users, with each

Number of streams	1	3	5	7
Duration in hh:mm:ss	00:18:01	00:16:41	00:16:34	00:16:09
Duration in seconds	1081	1001	994	969
Faster than 1 stream by	- - -	7%	8%	10%

user executing only a subset of the 10. The data was sparsely indexed, and the queries were very complex, relying heavily on derived tables and left outer joins and joining from 2 to 6 tables. The benchmark used a 1-node system loaded 70 GB of user data.

At right is a chart of the total time for the set of 10 queries to execute with the varying levels of concurrency. Overall time to execute was in all cases faster with more users involved. With 7 streams active, the total time to complete the 10 queries was 10% faster than the single user time.

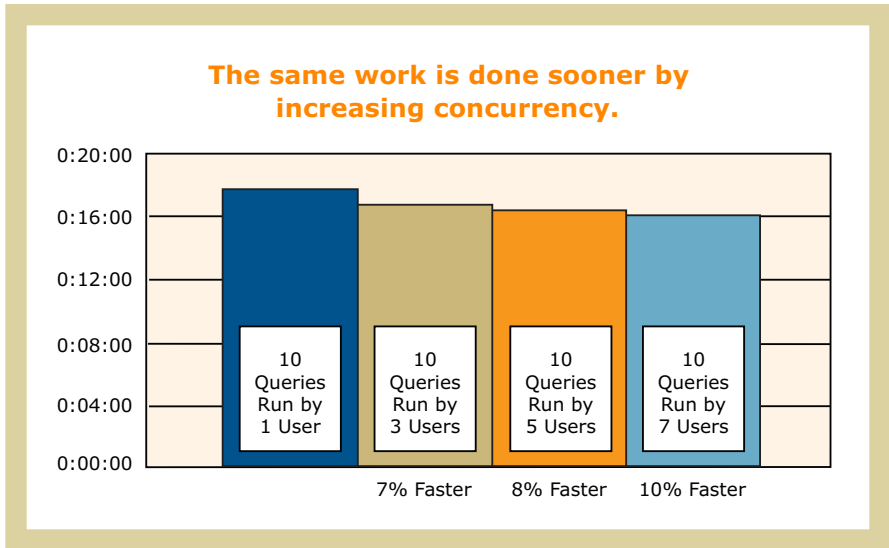
The following table shows the duration of each stream (in HH:MM:SS and in seconds) and compares each multi-user level against the single user total response time. The bottom row represents how much faster the multiple streams executed the

same work. Teradata is delivering better-than-linear performance for all the levels of concurrency testing in this benchmark.

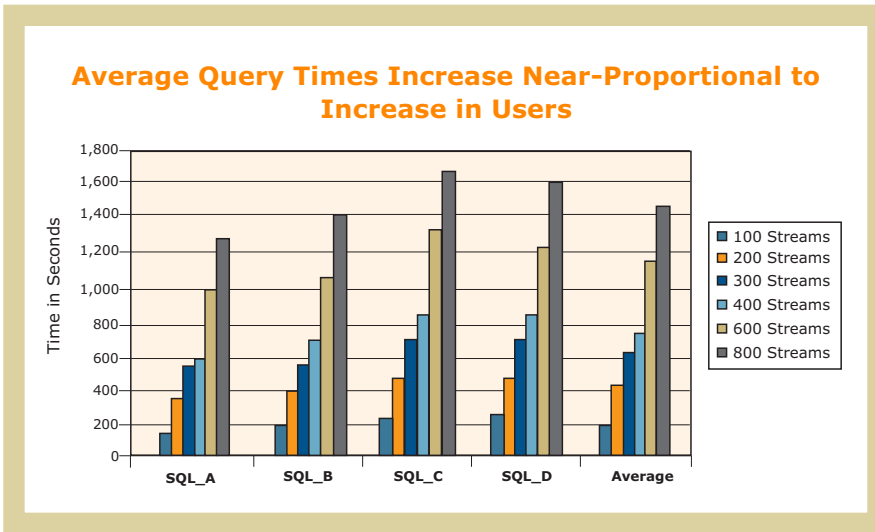
Concurrency Comparison from 100 Users to 800 Users

While the first two examples demonstrated linear scalability as users are added at modest levels of concurrency, this next benchmark example illustrates how Teradata scales with increasingly high numbers of users. In this benchmark, the baseline of comparison was 100 concurrent query streams, as the client already had installed a Teradata system that ran this number of users successfully. Starting at such a substantial level of concurrency, while not leaving much room for system throughput to increase, did successfully demonstrate that the growth in users results in a proportional increase in average query times.

Each user in each test executed the same four decision support queries. Every iteration of each query accessed different rows of the database, and each query joined from 3 to 6 tables. These four queries were executed by 100 concurrent users, 200, 300, 400, 600 and 800 users. The average query times for each of the four queries, as well as an average of these averages, at each of these six levels of concurrency are presented visually on the next page.



The Teradata Scalability Story



As can be seen in the final **Average** grouping on this chart, the increase in average query time, reported in seconds, as users were added is roughly proportional to the increase in users. Notice that with 100 streams, the average query time of 203 seconds compares well to the average query time with 800 streams of 1474 seconds. With eight times more users active on the system, query response time averages *less than* eight times longer. The threshold of linearity with 800 users would have been 1624 seconds, while the actual run time came in at only 1474 seconds.

The following table details the linearity story. The **Base Time** is the average query response time with 100 concurrent users (203.47 seconds). That 100-user time is compared against the **Reported Time** it took for the higher number of streams in that particular comparison to execute (on average). The **Users Increased by Factor Of** column indicates whether the number of users doubled, tripled, or as is the case in the last row, octupled. The **Average Query Time is Longer By** column divides the **Reported Time** by the 100-user **Base Time**, and shows how much longer the average query is taking with that factor of increase in users.

The information in this table can be used to assess scalability as users are increased by comparing the last two columns: The factor of user increase and the factor of query lengthening. Linear scalability is established when the average query is lengthened to the same degree (or less) that the concurrency is raised.

	Base Time	Reported Time	Users increased by factor of	Avg. query time is longer by
100 vs 200	203.47	448.40	2	2.20
100 vs 300	203.47	623.04	3	3.06
100 vs 400	203.47	761.83	4	3.74 *
100 vs 600	203.47	1157.57	6	5.69 *
100 vs 800	203.47	1474.97	8	7.25 *

* Better than linear performance is achieved when the average query lengthening factor is less than the factor of user increase.

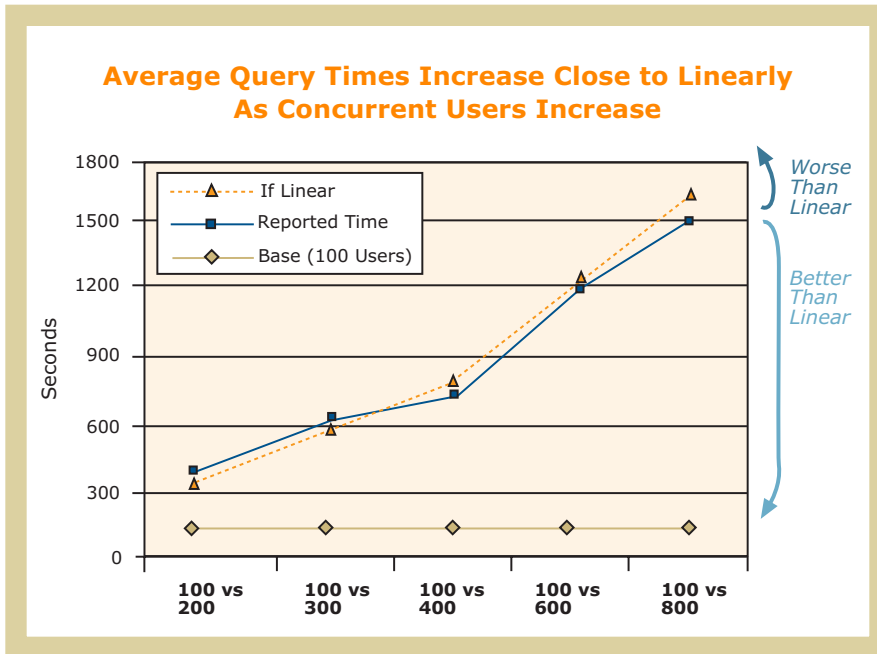
Another way of illustrating the nearness to linear scalability is to chart the average query times for each concurrency level progressively, comparing the lengthening of query times with what their rise should be if performance was perfectly linear. (See next page.)

It's interesting to note that between the 100-stream and 200-stream tests, the disk utilization became noticeably heavier in this benchmark, increasing more than the CPU usage did as concurrent users doubled. At the higher levels of concurrency, these resource utilizations continued to increase, but their rise was less dramatic than that seen going from 100 to 200 streams.

The trend in this benchmark from being slightly less than linear to greater than linear as concurrency grows points to an increasing benefit from caching and synchronized scan that is realized in this particular application above the 200-stream point. In the 400-stream and above tests, each physical I/O is providing greater value, and more work is moving through the system.

The importance of gathering multiple data points when looking for linear scalability is underscored in this example, where techniques in the Teradata Database successfully push performance forward even as stress in the system increases. Particularly useful characteristics of Teradata, such as highly effective flow control and resource sharing, as well as cache management and synchronized scan, are highlighted in this benchmark.

The Teradata Scalability Story



Scalability example expanding the hardware from 3 Nodes to 6 Nodes

In the benchmark below, the same end-user application was executed on a 3-node system and again on an equally powered 6-node system, both using Teradata. This MPP-to-MPP comparison shows close to perfect linear scalability. The time for the 31 concurrent users to complete all of the queries took half the time when the nodes were doubled, demonstrating linear performance with expansion.

In the chart on the next page, the actual reported 6-node time (expressed in HH:MM) is compared against what it would be if it were perfectly linear (half the 3-node time). The two-minute difference is evidence of near-perfect linearity (within 1%) as the configuration is increased.

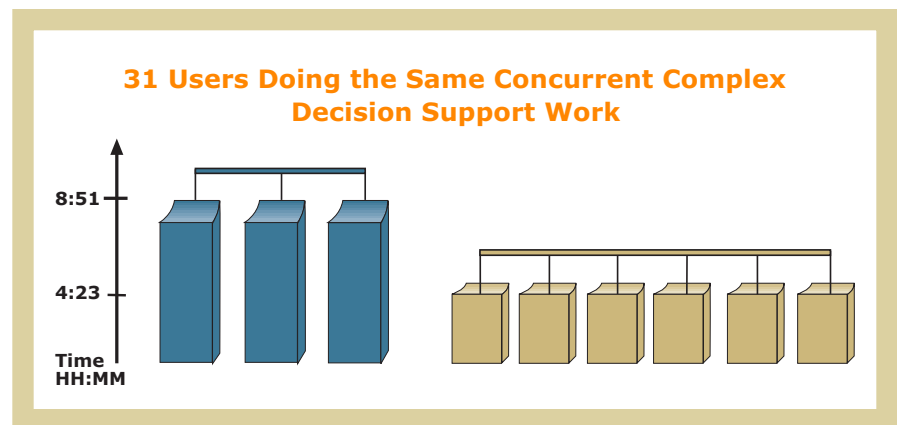
This benchmark was executed on an 8-node system, an MPP composed of eight 4-way SMP nodes. The largest table carried no secondary indexes, and was in all cases accessed using row hash match scan merge joins. The shortest reported query time was close to 100 seconds, so even though the data volume was moderate, significant database work (between 7 and 10 joins for each query) was being done.

Examples of Scalability with Hardware Expansion

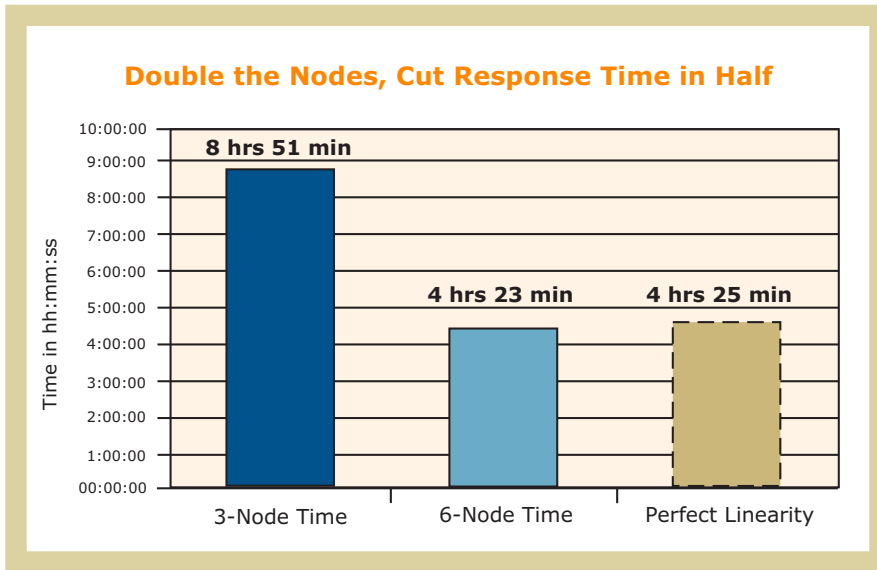
A proportional boost in performance that comes from adding nodes to an MPP system is a third dimension of linear scalability demonstrated by Teradata in

client benchmarks. In assessing this dimension of scalability, all database characteristics, data volumes and concurrency levels remain constant. As an example of this type of linearity, linear scalability is demonstrated if after doubling the number of nodes in an MPP system, the query response times are cut in half. Both the hardware and the software must be capable of linear scaling with expansion to lead to this experience.

In this benchmark, expansion is from an MPP to a larger MPP configuration. 100 GB of user data was executed against and the queries, taken from a banking application, averaged 6-way joins. The client involved in this benchmark noted the ease of setup they experienced first hand when benchmarking different configurations: “Teradata handles the partitioning of data automatically for the DBA. The DBA doesn’t have to deal with extent sizes, file



The Teradata Scalability Story



locations or pre-allocation of table spaces to support the table/index data. This is a major advantage for Teradata.”

Scalability example expanding from 6 Nodes to 8 Nodes

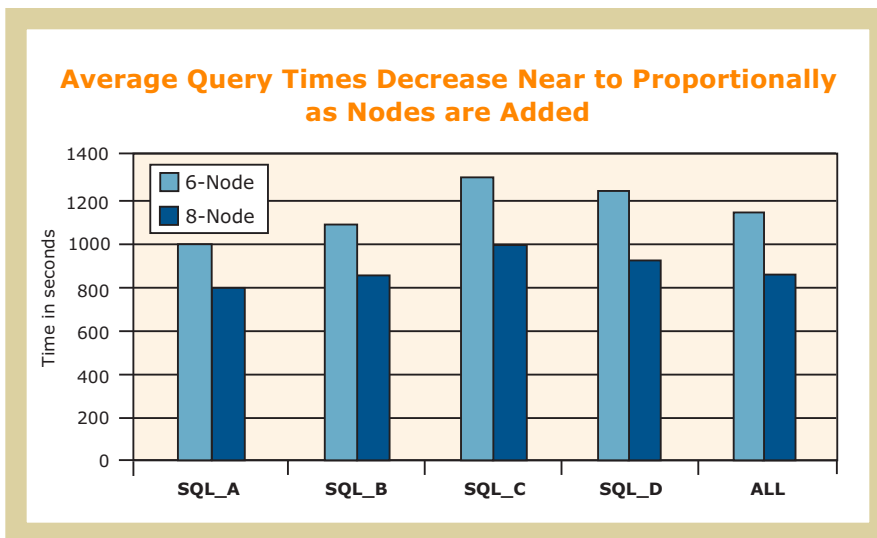
In another MPP-to-MPP benchmark, four SQL statements were executed by each of 600 concurrent users, with selection values changing with each execution. This was done first on a 6-node system, then repeated on an 8-node system. What is being compared is the average time for

each of the four queries on the 6-node system against the equivalent times on the 8-node system.

Because the configuration expanded from 6 to 8 nodes, linear scalability would be established if query times were reduced by one fourth of their 6-node response time. The average time for each of the four SQL statements is captured in the table below (A, B, C, D). The final row represents the average time of all four SQL statement averages.

	6-Node Time	8-Node Time	Com-parison	Ratio (1.33 is Linear)
SQL_A	1003.9	800.6	1003.9 / 800.6	1.25
SQL_B	1088.1	847.6	1088.1 / 847.6	1.28
SQL_C	1307.2	1003.1	1307.2 / 1003.1	1.30
SQL_D	1231.0	928.7	1231.0 / 928.7	1.33
Avg.	1157.6	895.0	1157.6 / 895.0	1.29

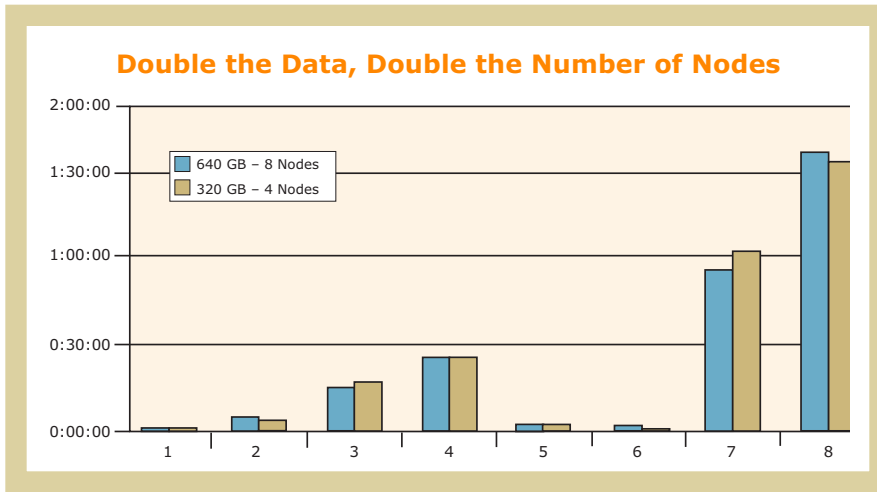
In this case, the testing has shown scalability very close to linear with the two additional nodes. Some mild fluctuation in the comparison ratios among the four queries comes from the differing characteristics of queries selected for the benchmark and their behavior under changing conditions. The detail behind this benchmark is the same as the previous example earlier in this paper of concurrency up to 800 users. In addition to testing the scalability of multiple users, this client illustrated the effect of expanding the configuration on the multi-user query time.



Doubling the Nodes and the Data

In another client benchmark (see next page), the same set of queries was executed against 320 GB on a 4-node system, and compared to the same queries running against 640 GB on an 8-node system. Each node utilized 4 CPUs with 2 GB of memory. When data volume and configuration doubled, reported query times were similar for each query, demonstrating linear scalability across two dimensions of change – hardware and volume.

The Teradata Scalability Story



NOTE: Perfect linearity is achieved if response times for each query are identical.

In this test, each of the 8 queries was executed stand-alone, both at 320 GB on a 4-node system, and then again against 640 GB on an 8-node system. Perfect linearity would've been achieved if each query executed in the same number of seconds on both configurations, with an 8-node to 4-node ratio of 1.0 being linear. Reported response times in seconds are in the table at right.

The **Ratio** column in the table, a technique used here as an aid to evaluate scalability, that ratio shows some variance across this set of queries. Note that the queries that show the greatest variation from linear performance are also the shortest queries (1, 6, for example). Further, these short queries varied in their variance, with #1 performing better on the larger configuration, and #6 performing better on the smaller. The reported times for short queries (queries that run less than a minute) are more sensitive to interference from such things as query start-up, parsing time, effort to return the answer set, and in isolation, short queries are often inconclusive.

In this benchmark the longer running queries (4, 7, 8) provide a smoother linear picture of performance and provide stability across both volume and configuration growth. The average query time across all queries is within 2% of

	320 GB 4 Nodes	640 GB 8 Nodes	Com- parison	Ratio (1.0 is Linear)
1	41	25	25 / 41	0.61
2	177	198	198 / 177	1.12
3	972	864	864 / 972	0.89
4	1,500	1,501	1501 / 1500	1.00
5	160	154	154 / 160	0.96
6	43	101	101 / 43	2.35
7	3,660	3,354	3354 / 3660	0.92
8	5,520	5,651	5651 / 5520	1.02
Avg.	12,073	11,848	11848 / 12073	0.98

perfect linearity, in the face of changes in both volume and configuration.

Conclusion

Linear scalability is the ability of a platform to provide performance that responds proportionally to change in the system. Scalability must be considered as it applies to data warehouse applications, where the numbers of data rows being processed can have a direct effect on query response time, and where increasing users directly impacts other work going on in the system. Change is here to stay, but linear scalability within the industry, unfortunately, is the exception.

The Teradata platform is unique in its ability to deliver linear scalability as users increase, as volume grows and as the MPP system expands, as evidenced by examining client benchmark results. This enables non-disruptive growth and quicker turnaround on new user requests, builds confidence in the quality of the information being provided, and offers relief from the fear of too much data. Linear scalability is the promise of a successful tomorrow. Ask for it, look for it. Demand nothing less.

Carrie Ballinger is a Senior Technical Consultant within Teradata, a division of NCR. She works in the Active Data Warehouse Center of Expertise in El Segundo, California. Carrie has specialized in Teradata Database design, performance and benchmarking since 1988. You can email her at: carrie.ballinger@ncr.com. For more information, call 1.888.627.0508 ext. 239.

Carrie would like to thank NCR's Customer Benchmarking Team in San Diego for their work on these benchmarks and for contributing the examples that tell the scalability story.

Teradata is a registered trademark and BYNET is a trademark of NCR Corporation. NCR continually improves products as new technologies and components become available. NCR, therefore, reserves the right to change specifications without prior notice. All features, functions and operations described herein may not be marketed in all parts of the world. Consult your Teradata representative for further information.

© 2001 NCR Corporation Dayton, OH U.S.A. Produced in U.S.A. All rights reserved.

www.ncr.com www.teradata.com