DBC/1012 MODEL 4
PERFORMANCE COMPARISON

no multiload

DBC/1012 Model 4 Performance Comparison

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The results published in this document are the fruit of many people's labor. The benchmark team came into being more from magnetism than planning yet the resulting cooperation and productivity was phenomenal.

Special thanks are in order to Vince Jones who, as product launch manager, conceived this project back in 1990. If you like this type of information, call Vince and tell him.

Next we owe a large debt of gratitude to the guys in manufacturing who in the midst of several metaphorical hurricanes supplied several million dollars worth of hardware to the benchmark team.

The folks who really burned up the pavement were, as usual, the data center engineers. Though the systems were remarkably stable, our needs were such that they had to reconfigure the systems every other day, swap boards between machines every week, and search for odds and ends of channels and systems every third day. In every case, these people caught fiery bullets in their teeth and remarked "I'm just doing my job."

There could be no success in this project without the guidance of several people in the Performance Group and Sales Support. Their depth of knowledge and "can do" willingness to help were the catalysts for success. Like the engineers in the data center, through it all their trademark was talent, long hours, and great attitudes.

Last, a very special mention is in order for Ms. Carrie Ballinger. As the principal worker and brains of the benchmark team, Ms. Ballinger's contribution to this project exceed what should ever be asked of an employee. Somewhere at the end of the fourth consecutive 16 hour day following many 10 and 12 hour days, the first results of Carrie's carefully planned tests came in and we knew we had struck a gusher. Thank you Carrie.

Many thanks to the following people. They don't come any better than this:

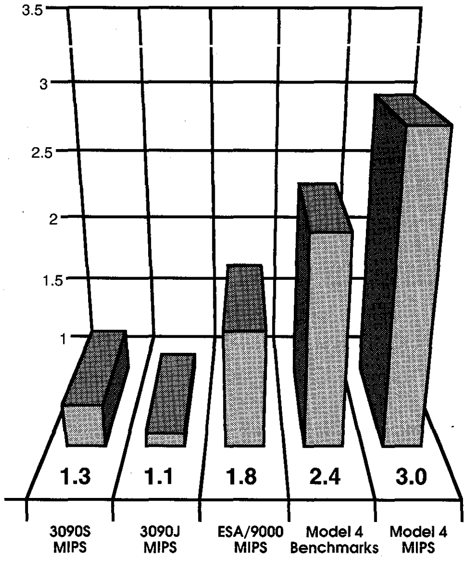
Russ Gilman, Chris Stradford, Mark McFadden, Armen Khatchatourian, Anees Narsinh, Al DiCanzio, Cindy Akiyama, Anita Richards, Bill Robertson, Ron Hu, Larry Higa, Kathleen Webber, Ken Plummer, Owen Tucker, Terry Patterson, Craig Hayes, and the entire second shift in the data center.

Dan Graham
Sr. Product Marketing Manager

1.0 EXECUTIVE OVERVIEW

INDUSTRY EXPECTATION FOR PERFORMANCE IMPROVEMENTS

IBM HARDWARE VS. MODEL 4

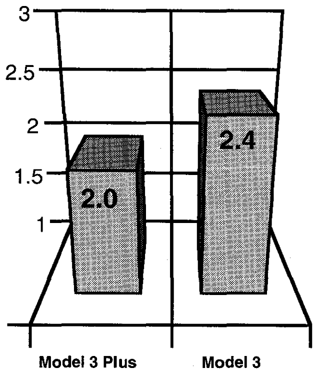


In order to determine the performance characteristics of the Model 4 DBC/1012 prior to announcement, Teradata's Marketing staff organized a series of benchmark tests which were completed in February 1991. The purpose of the testing is to provide the Teradata field organizations with general comparisons of Model 3 and Model 4 performance to assist them in their sales efforts. This document is intended to be a high level overview of the DBC/1012 Model 4 in real world simulations. Those individuals interested in the very detailed performance characteristics should refer to the results published internally by the Product Development Performance Analysis Group.

The results presented in this study should be reviewed in the context of the large scale relational database marketplace. When IBM and other vendors release new models, they provide MIPS ratings rather than actual benchmarks as performance guidelines. For example, when comparing MIPS, the 3090-S series is 1.3 times faster than the E series (30%) and the J series is 1.1 times faster than the S series (10%). While details are still incomplete, it appears the ESA/9000 might be as much as 1.8 times faster than its predecessor J series (80%). In contrast, the DBC/1012 Model 4 MIPS rating is 3.0 times faster than it's predecessor (200%), a testimony to Teradata's off-the-shelf merchant chip strategy. MIPS ratings are, however, only crude estimates of performance and do not reflect the actual improvements delivered to the end user's desktop. Consequently, Teradata is very pleased to provide the following real-world proof of Model 4 performance, especially since these results are measured at the desktop and greatly exceed the mysterious MIPS ratings offered by other vendors.

AVERAGE PERFORMANCE IMPROVEMENT

MODEL 4 OVER MODEL 3



In general, the Model 4 system is 2.0 times faster than the Model 3 Plus (a Model 3 enhanced with 256K cache & 8 megabytes of memory) and 2.4 times faster than the basic Model 3 system. These factors are applicable across a wide range of decision support queries, batch, and on-line activities. Decision support activities ranged from 1.8 times faster through 3.5 times faster than the Model 3 Plus. Batch oriented sorting, "joins", and Fastload ranged from 1.8 times faster to 2.3 times faster than the Model 3 Plus. On-line transaction processing improved by factors ranging from 1.3 to 2.0 over the Model 3 systems, depending on the OLTP activity tested. (This is within expectations since OLTP uses so little CPU and so much disk I/O.) These improvements extend the DBC/1012's already enormous lead in DSS, address the batch processing needs of our larger customers, and allows the DBC/1012 to participate in medium sized OLTP applications. Leading in two categories and competing in the last, the Model 4 is an excellent choice for nearly all large production applications.

Two customized benchmark tests were designed in order to highlight the Model 4 AMP processing speed, conservatively rated at nine MIPS, versus the Model 3 CPU, rated at three MIPS. These tests involved complex SQL functions which are common in banking, retail, and financial applications. These tests included SQL activities such as integer conversions, date computations, and roll-up summary reporting. The test results showed a staggering 3.57 and 2.7 times improvement over the Model 3 Plus system. (estimated at 4.4 and 3.3 times improvement over the basic Model 3). This leads to the obvious question that if the Model 4 CPU is rated at three times faster than a Model 3, how could it outperform the Model 3 Plus by this margin? The answer lies in the changes to the block-size management software of Release 4.1.2 which affects data blocks, spool files, and sort files. As predicted by our developers, the Model 4 can and will consume disk data at rates much higher than the Model 3 necessitating a change in the software used with the Model 4.

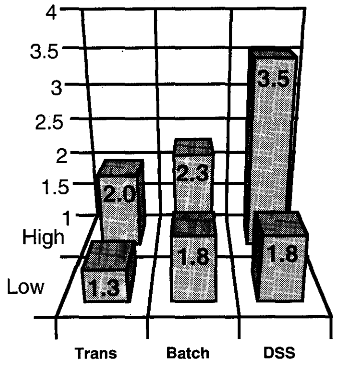
In attempting to determine differences between the 2.5 GB and 1.2 GB disk drives, several disk intensive queries and numerous on-line transaction simulations were run. It turned out to be quite difficult to create a disk bottle-necked DSS query without simulating dozens of users. Decision support queries often process many rows per I/O and are more compute intensive than might be apparent. Hence the effects of one disk drive versus another is negligible. Consequently, with only five users simulated there was little difference between 1.2 GB and 2.5 GB disks in elapsed times. With many, many users issuing concurrent DSS requests, customers can expect to see higher throughput with multiple disk units per AMP similar to the results of in the on-line transaction tests.

Numerous debit/credit OLTP Tests were conducted in order to determine the usefulness of multiple drive configurations as well as the Intelligent Peripheral Interface (IPI) mode emulation. When there are multiple Disk Storage Units (DSUs) per AMP, the IPI emulation compares the outstanding requests for disk I/O to the current sector number passing under the disk heads. IPI emulation then chooses the best transfer strategy to insure maximum throughput rather than simply handling disk I/O on a first-come-first-served basis.

In row-at-a-time functions, dual DSUs per AMP provide approximately 7.5% more throughput than a single DSU, triple DSUs provide 22% more, and quad DSUs deliver 27% more throughput. These factors held true for both 1.2 GB and 2.5 GB disks with a slight advantage in the larger 2.5 GB disks. Non-Volatile Disk Cache running with one DSU per AMP clearly wins as the price/performer in applications which update a row-at-a-time, yielding a 55% boost in throughput. It is our observation that Non-Volatile Disk Cache has the same performance characteristics on the Model 4 as on the Model 3 and should be installed wherever the application is update intensive. The above multiple-DSU factors did not occur in complex decision support since we were unable to simulate a large volume of concurrent users. Had we been able to simulate large numbers of DSS users, similar throughput improvements for multiple DSUs would prob-

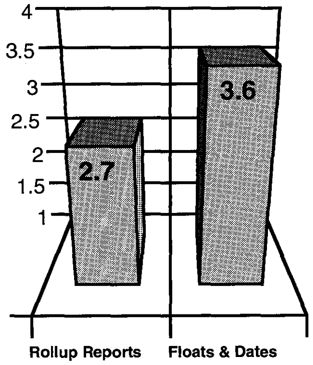
IMPROVEMENT RANGE

MODEL 4 OVER MODEL 3 SYSTEMS



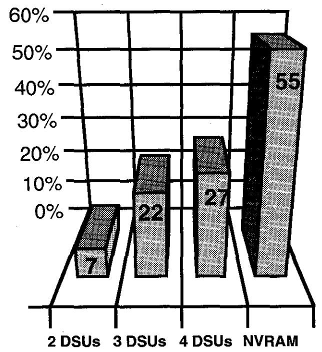
CPU INTENSIVE TESTS

MODEL 4 OVER MODEL 3 PLUS



TRANSACTIONS PER SECOND INCREASE OF MULTIPLE DISKS

1.2 GB & 2.5 GB DSU'S



ably occur over the system as a whole although no one user would see the benefit; ie more users would be getting work done but execution times would remain constant. In testing which simulated three concurrent user requests, dual 1.2 GB disks out performed a single 2.5 GB disk by 1%, a noise level result.

Fastload Phase 2 performance leaped to 2.22 times faster than the Model 3 system. Phase 1 results showed a 1.7 times improvement over Model 3. The Phase 1 results are not considered meaningful since they are heavily weighted by the speed of the particular mainframe the Fastload test is run on.

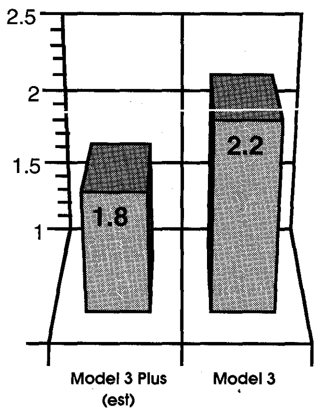
For our current DBC/1012 Model 2 customers, upgrades to a Model 4 system will be extremely rewarding . In previously published results, the Model 3 system was shown to run 2.5 to 3 times faster than the Model 2. Applying a 2.0 factor to this, conservative estimates suggest that Model 2 customers who upgrade their Model 2 AMPs to Model 4 will see performance gains of 4 to 6 times, depending on the current CPU utilization of their system. "Extremely rewarding" is clearly an understatement of the facts.

Lastly, we must not ignore the fact that many current customers who move to the Model 4 will also be upgrading from Release 3.2.2 to Release 4.1.2. Imbedded in the software upgrade are numerous performance enhancements, some of which are improvements of several magnitudes. For example, many of our current Release 4.1.0 users report enormous jumps in sort performance and in the DELETE ALL function. These software boosts which are accelerating existing production systems (in particular the nightly batch cycles) will be boosted even further with the addition of a Model 4 system. Hence, the Release 3.2.2 customer will get a double-dose performance boost when moving to the DBC/1012 Model 4.

Overall, the DBC/1012 Model 4 is an excellent performer with tremendous potential for customer satisfaction. Clearly, the best use of the DBC/1012 Model 4 will be in handling large volumes of data with increasingly more complex SQL and relational functions. The more complex the user request, the better. Applause is in order for the hardware designers and software developers who made this product possible. The DBC/1012 Model 4 will be a winner for Teradata but more importantly, it will make our customers winners too.

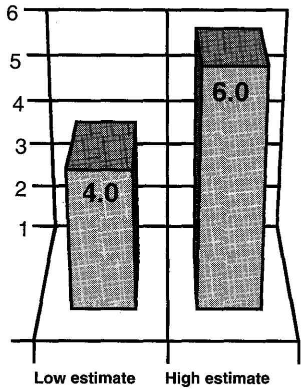
FASTLOAD IMPROVEMENT

MODEL 4 OVER MODEL 3



IMPROVEMENT RANGE

MODEL 4 OVER MODEL 2



2.0 Technical Overview of the Benchmark Tests

A primary objective of this benchmark is to provide the Teradata field organization information that will aid in configuring Model 4 systems. The enclosed results provide a strong foundation towards solving this need. Nevertheless, such information can never be more than a partial answer. This is because there are so many DBC/1012 customers and prospects and each of them uses the system differently. With that in mind, Teradata Systems Engineers are advised to refer to the DOC system Design Notes that are being developed in parallel with this document.

Nearly half the tests conducted were taken from the test suite used to produce the Release 3.2.2 versus 4.1.0 comparisons published at the end of 1990.

2.1 Test Configurations

Rel 4.1.1	Framer 60.11.120 base, 60.11.951 actual
14x24x24	Model 4 2.5 GB disks
14x24x48	Model 4 2.5 GB disks
14x24x72	Model 4 2.5 GB disks
14x24x96	Model 4 2.5 GB disks
7x24x24	Model 3 ***Plus*** 1.2 GB disks, 256k cache, 8Mb Ram
8x8x16	Model 4 1.2 GB disks
8x8x8	Model 4 2.5 GB disks

The primary test system was a 14x24x96 Model 4 machine with two VM IFPs and 12 MVS IFPs. The testing was run on pre-release 4.1.1 software, framer 60.11.120 base. All disks used were 2.5 GB disks. The mainframe in use at the time was an Amdahl 5890-300 partitioned 50/45 for VM and MVS. The high ratio of IFPs to AMPs was needed for the OLTP testing in order to isolate disk performance from IFP and host performance.

The primary Model 3 tests were conducted on a 7x24x24 machine using 1.2 GB disks, 256K cache, and 8 megabytes of memory per processor. Pre-release 4.1.1 (60.11.120) was also used on the Model 3 tests. Throughout this document, this system is referred to as the Model 3 Plus. It was not necessary to have a large number of IFPs on the Model 3 systems since it was used to test decision support queries not OLTP and disk utilization.

Two additional configurations were built in order to test the 1.2 GB disks and 2.5 GB disks. These tests were run on two Model 4 systems, one running 8x8x8 with 2.5 GB disks, the other running 8x8x16 and 1.2 GB disks.

Non-Volatile Disk Cache was used only on the 24 AMP Model 4 system. When a given benchmark test used the disk cache, the test description explains the effects.

2.2 Databases

Most of the testing was performed on database CAB which contained approximately four gigabytes of data (or 15 tape cartridges for perspective).

Easy to identify naming conventions were used in the database. For example, table T13M means there were 13 million rows and table T50K means there are 50,000 rows in the table. Column names are similarly named such that IN100 means an integer field of 100 distinct values and IN1M means one million distinct values in the column. Columns named DAB are a date-of-birth value; DAY is a randomly selected day in 1989. In nearly every case, the data rows are randomly distributed on an integer field called INSEQ which contains the row number if viewed from a sorted list. D08SEQ indicates a decimal field which also contains the consecutive row sequence number. This information is useful when reviewing the SQL in the appendix.

2.3 Measurements and Testing Method

All tests were run an average of three times in order to normalize the resulting service rates. In each case, the elapsed time reported by BTEQ following the SQL statement is used to define the system performance since this is roughly what the end user would also perceive. "Select Time" statements preceded and followed nearly every SQL statement. The BTEQ service speed matched the "select time" time in every test with consideration given for slight rounding differences. In the case of the debit/credit testing, RESUSE reports were used to cull transactions per AMP per second, pathlengths, response times, and I/Os per second per AMP.

During decision support testing, the AMP memory was emptied between each query by scanning a large table. This essentially purges all idle blocks of data from AMP memory and insured that the query response times were not affected by residual data leftover from previous tests.

2.4 Record Sizes Used in Testing

Record sizes are not included in this document for reasons of confidentiality. Consequently, these tests cannot be reproduced without contacting the Teradata Marketing organization, who will be glad to assist you in reproducing these results or in supplying table layouts.

2.5 Calculation of Estimates

Since most of the relative performance factors given compare a Model 3 Plus to a Model 4 system, mental adjustment is necessary if you are considering an upgrade from the basic Model 3 system to a Model 4. Generally, adding 256K cache and 8 megabytes of memory to the Model 3 processors results in a 15 to 25% performance increase. Hence, if a Model 4 factor is shown as two times faster than the Model 3 Plus, you should mentally convert this to 2.5 times faster than the basic Model 3. This roughly compares to tests done by Teradata's Performance Group organization showing the Model 4 running 2.6 times faster than the basic Model 3 in their GrandMix DSS tests.

In many tests, an estimated performance figure is given for the DBC/1012 model not tested. When the basic Model 3 system is being estimated, we used a multiplier of 1.25 to arrive at the estimate. When an estimate for the Model 3 Plus is being calculated, we used a multiplier of 0.8 to reduce the improvement factor. Keep in mind that additional cache memory provides differing levels of performance increase in different types of tests, hence the broad application of these estimate factors are not the most accurate method of estimating. For example, OLTP uses cache memory differently from DSS activities. When considering estimates, the Model 3 customer will never get less performance than the Model 3 Plus measurement and will occasionally receive better performance than the Model 3 estimate suggests. In general, the typical performance increase will be between the measured amount and the estimated amount. Please use the estimates as guidelines only.

2.6 The SQL Used in the Tests

In appendix A you will find the SQL used for the important queries. The SQL is sometimes useful in matching queries used by DBC/1012 users with test results found in this document. To aid in matching the SQL with the tests, most of the queries have a random sequence number attached to the SQL and the test title. For example, the Floats & Dates section immediately following contains the number 901 in parenthesis following the paragraph title to direct you to SQL item number 901 in the appendix.

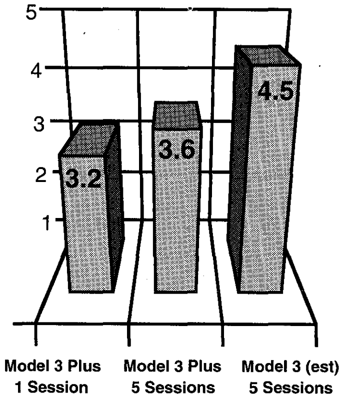
Also, the actual CREATE INDEX statements have been included for those inclined to examine the primary index and foreign keys in the tables.

3.0 CPU INTENSIVE TESTS

Two test queries were specifically designed to reveal the improved CPU speed of the DBC/1012 Model 4. The results of these queries turned out to be the biggest surprise of the entire testing process. The SQL for the tests can be referenced using the number in parenthesis following the paragraph title.

3.1 Floats & Dates (901)

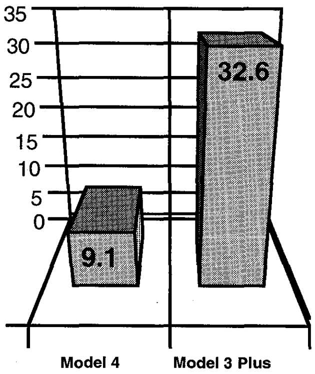
FLOATING POINT & DATES
MODEL 4 OVER MODEL 3



SQL is rarely confined to the reading and writing of data rows. Many Teradata customers invoke its 4GL like manipulation features and incur heavy CPU utilization which they interpret as disk utilization. Two of the more CPU intense functions commonly in use are integer conversion-comparisons and date calculations. These types of functions are common, everyday activities for most DBC/1012 users. For example, financial and insurance applications rely heavily on differencing two dates in payment-due calculations, amortizations, and actuarial analysis. The integer to floating point conversion is an activity frequently needed by the scientific community in applications such as weather monitoring and chemical compound analysis.

This query caused the system to scan one million rows, applying complex "where" selections. For each row selected, a comparison of an integer field to nine floating point values caused a conversion of the integer value to floating point notation. Additionally, several date fields in the row were converted in order to compute date plus or minus values for comparison.

FIVE SESSIONS FLOATS & DATES
EXECUTION SPEEDS - MINUTES



This test was run as a single session (one user doing one query) and also as five concurrent sessions. The five concurrent sessions illustrates the performance improvement possible when multiple concurrent users are running similar requests against the same tables. In such cases, there is some benefit from User-B asking for rows that User-A has recently brought into memory.

The staggering result is that the tests show a 3.57 and 3.18 times improvement over the Model 3 Plus system. How is it possible that a CPU that is three times faster can deliver 3.18 times improvement including disk I/O? Part of the answer lies in the changes to the internal block-size algorithm of Model 4 Release 4.xx software. Here we see revealed the obvious fact that the CPU is being fed more data per I/O, requiring fewer I/Os overall. This confirms the developer's choice of changing the Model 4 system to the new block-size management logic.

3.2 Roll-up Reporting – Sums & Strings (900)

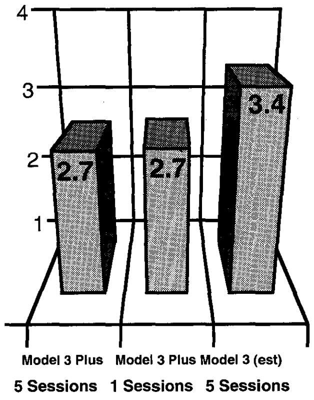
Common to financial, manufacturing, and retail applications is the use of SQL to create the "roll-up" report. This seemingly innocuous SQL process usually replaces dozens of lines of COBOL code and produces a middle-manager level report. As such, it represents a highly CPU intensive activity involving scans, sorting, and aggregations. For most organizations which manage budgets or sales history, this type of SQL query is a daily or weekly interaction with the DBC/1012. This query is a production report used by an existing DBC/1012 customer.

Review of the SQL of this query shows approximately 18 sum aggregations (roll-up totals), one count result, two string manipulations, seven arithmetic steps, and numerous column datatype conversions. Additionally, the sort key used contains eight fields, some of which are calculated. Overall, this is a very realistic end-user request for the DBC/1012 that would probably demolish the garden variety RDBMS software.

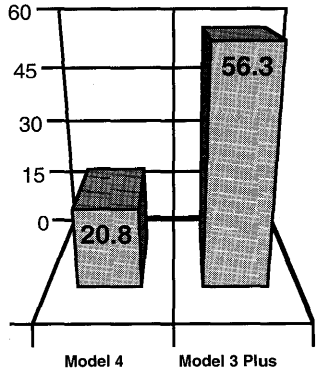
Like the Floats & Dates query, this test was run as a single session and five session query. Again, we encountered that rare excitement so much like discovering a hidden treasure. Running at 2.7 (5 sessions) and 2.45 (1 session) times faster than the Model 3 Plus, again the Model 4 CPU coupled with the changes in block-size management produced impressive response times. This type of performance in real world situations is evidence of a job well done by Teradata's development staff.

SUMS & STRINGS TEST

MODEL 4 IMPROVEMENT OVER MODEL 3

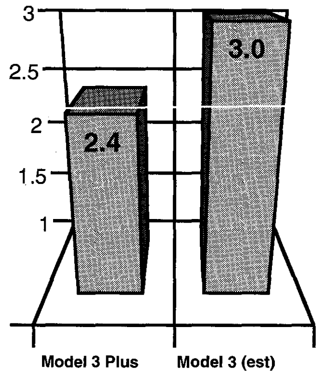


EXECUTION SPEEDS - MINUTES

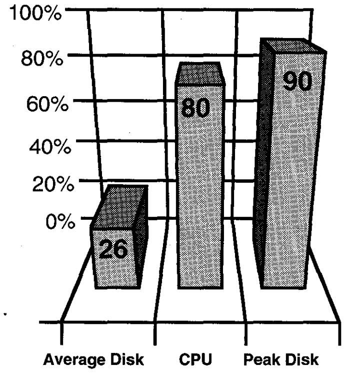


HIGH DISK & CPU UTILIZATION

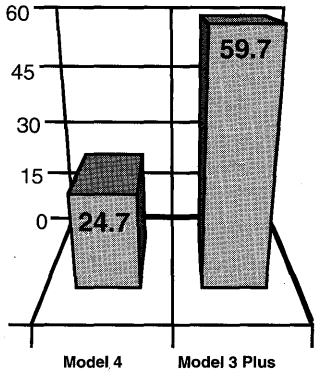
MODLEL 4 OVER MODEL 3



MODEL 4 & MODEL 3 UTILIZATIONS



EXECUTION SPEEDS - MINUTES



4.0 DISK INTENSIVE SCANS

The intent of the disk intensive tests was to reproduce the types of activity common to production environments. Foremost, we wanted to address the obvious question "How many disk drives of which type should we use?" In particular, new prospects require help selecting between two 1.2 GB disks or one 2.5 GB disk drive since they provide similar capacity. This invites the next disk related question "Which applications benefit most from having more disks per AMP?". This goal proved to be a more daunting task than one would expect.

As the benchmark proceeded, the team shifted away from queries we expected to generate high levels of disk activity such as multiple joins and row redistributions. Surprisingly, the Model 4 showed lower disk utilization in these tests than would normally be expected. We quickly devised new tests which utilized larger row sizes and 3-part multi-statement SQL requests. The multi-statement SQL approach was able to simulate multiple requesters running concurrently against different tables. The results were significantly more disk intensive.

4.1 High Utilization – Both Disk & CPU (902)

In this first multi-statement SQL disk-test, the three queries join 1 million rows to 100 rows, five million rows to fifty rows, and 13 million rows to 50,000 rows. In each of the overlapped parallel steps, the queries produce several spool files, the smaller of which are duplicated on all AMPs, sorted, and merge-joined on non-indexed columns. These types of queries are typical of a variety of production batch applications (retail, airline, banking, utilities, etc.) where the user matches customer to household, seats to airfare, product to vendors, etc.

Moving several millions of rows during these queries would have implied to the casual observer a severely high disk utilization. However, the disk utilization tended to hover between 18% and 26% utilization while CPU utilization remained steady in the mid 80's on both the Model 3 and 4. For a short three minute period, the disk utilization climbed to over 90% utilization. Still, it is surprising that the CPU and disk utilization figures for the Model 4 were almost the same as the Model 3 Plus figures. However, the disk I/Os per Amp per second on the Model 4 averaged 50% higher than on the Model 3 (for example 12 I/Os versus 7, and 33 versus 21).

Clearly, the very large blocksize used for the spool files and sort activity kept the Model 4 AMP busy such that it did not become I/O bound. Notice that the SQL join function is significantly CPU intense though we often visualize it as simply the reading and combining of disk records. The overall result showed the Model 4 system to be 2.41 times faster than the Model 3 Plus configuration. This is an excellent improvement in a supposedly disk bound activity. Clearly, the increased blocksize in the Model 4 spool-files was necessary to minimize disk I/O bottlenecks and increase throughput.

4.2 High Disk – Low CPU Utilization (903)

In the second disk intensive query, we simulated three concurrent decision support users accessing large data rows. Using a multi-statement SQL query, we invoked full-table scans of 10, 5, and 1 million rows respectively, each from different tables. By specifying a return limit of one row, the queries were forced to go through the entire process of accessing every row, then returning only the first row of each spool to the mainframe. Each query contained a simple "less-than" comparison for selecting rows.

On both the Model 3 Plus and the Model 4, disk utilization ran up to 97 to 100% and stayed there throughout the test. For the Model 3, this ran to 47 I/Os per second per AMP where the Model 4 was able to consistently exceed 52 I/Os per second per AMP. CPU Utilization on the Model 3 Plus was a consistent 35 to 38% saturation whereas the Model 4 consumed 23 to 24% of the available MIPS.

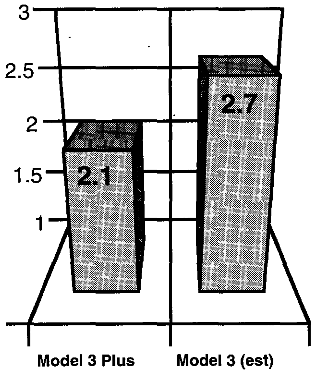
This test showed the effect of the new block-size algorithm on the Model 4 system. Having nearly eliminated the CPU as a factor, this truly disk intensive process might be expected to be as little as 1.3 to 1.5 times faster on the Model 4. But the effect of the new block-size logic on the data row, sort, and spool areas results in a performance increase of 2.13 times over the Model 3 Plus, roughly halving the response time. One conclusion we may draw from this is that similar disk constrained table scans will maintain roughly similar proportions of CPU and disk utilization across the Model 3 Plus and Model 4 DBC/1012.

4.3 1.2 GB disks & 2.5 GB disks

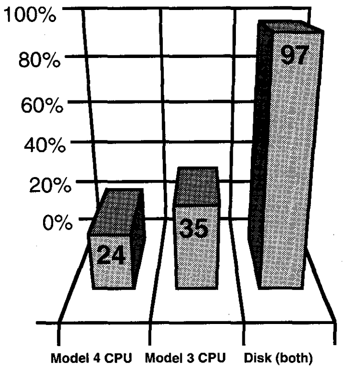
The purpose of these tests was to determine when a system should be configured with a single 2.5 GB disk versus two 1.2 GB disks. Additionally, we wanted to learn the value of having multiple spindles per AMP processor under various tests and configurations. This last requirement was all the more important now that the Intelligent Peripheral Interface (IPI) mode disk controller functions are being simulated on the Model 4 via firmware and SMD disk drives. This smarter use of the disk channel permitted by the Rotational Position Sensing (RPS) is a valuable feature in disk intensive activities that we hoped to measure. Hence, testing with one, two, three, and four 2.5 GB disk drives per AMP was done to

HIGH DISK, LOW CPU UTILIZATION

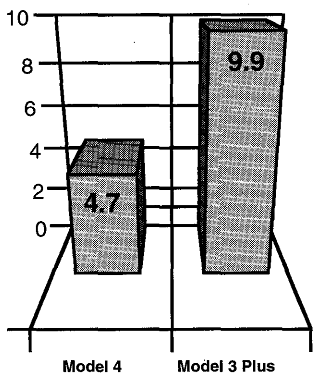
MODEL 4 IMPROVEMENTS OVER MODEL 3



AVERAGE CPU & DISK UTILIZATION



EXECUTION SPEEDS - MINUTES



determine how decision support activities are affected by the multiple spindles.

After consulting with performance experts, the OLTP debit/credit test was selected as the most likely to reveal differences in disk bound activity. The debit/credit transaction contains three updates and an insert, all of which invoke significant parallel disk activity particularly in the fallback and journalling areas. (Note that this debit/credit is not modeled after the Transaction Processing Council benchmarks A or B.) The short row-at-a-time activity in the debit/credit tests are known for consuming copious amounts of disk service with the most primitive of SQL statements.

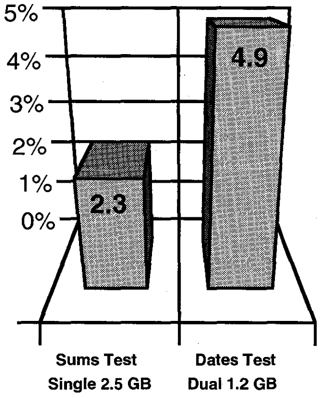
4.3.1 New Disk Drives and Decision Support Queries

Even though decision support queries may be disk intensive, they tend not to utilize more than one disk at a time per AMP. Only when multiple sessions and parallel steps are running can the single decision support request begin to exercise multiple disk drives per AMP. Hence, such requests do not generate the flurry of random disk I/O invoked by the debit/credit testing.

Earlier in this document you will find a discussion of two multi-statement SQL queries that were developed to create a disk intensive decision support process. These tests drove the disks up towards 100% use while CPU consumption was relatively low. These two tests were repeated using two Model 4 systems, both of which were running the full IPI mode emulation. The tests were conducted on an 8x8x8 with 2.5 GB disks and an 8x8x16 running 1.2 GB disks.

1.2 GB & 2.5 GB DISKS COMPARISON

CPU INTENSIVE TESTS - REVISITED



Since the queries simulated three users running full-table scans concurrently, we expected a high rate of disk I/O to favor the two spindle 1.2 GB disk configuration. This was not the case. Comparisons in the two tests showed a difference of less than one percent in the elapsed times. Generally, in decision support test of this nature, anything less than three percentage points of difference is considered noise level results. Consequently, there was no measurable difference between the dual 1.2 GB disks and single 2.5 GB disk configuration in these tests.

Two other decision support tests were run in the above configurations. These were the CPU intensive queries – Floats & Dates, Sums & Strings – discussed earlier. In the SUMs roll-up report, the 2.5 GB disk configuration out performed the 1.2 GB disks by 2.3%, still a noise level result. In the Floats & Dates test, the 2.5 GB disk out performed the dual 1.2 GB disks by 4.9%. In the latter case, it appears that the faster seek time of the 2.5 GB disk would account for a slight performance boost over the 1.2 GB disks. In each case, these tests simulated five concurrent users via the "BTEQ .repeat 5" feature.

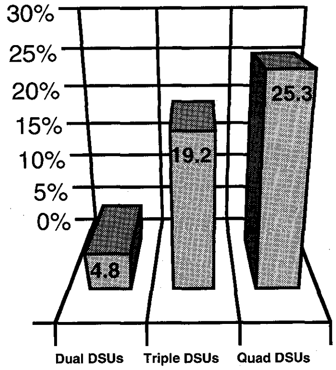
4.3.2 Multiple Drive Testing with On-line Transactions

The first thing to consider is that these tests provide insight on random I/O, disk intensive, high concurrency processes only. Activities in this genre include OLTP, BulkLoad, some Insert-Select processing, on-line complex queries of short duration, and a few (very few) decision support functions. The common denominator of these activities is the high volume, row-at-a-time processing which is not the dominating factor in such things as Fastload or SQL joins.

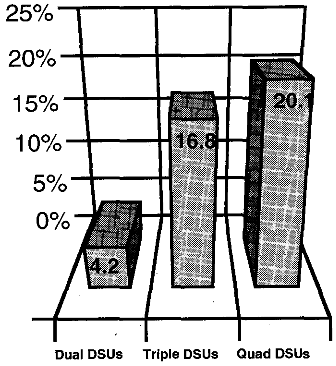
The OLTP tests were run on the same Model 4 system on average three times in each configuration. Testing was conducted on a system with 12 IFPs and 24 AMPs. The large ratio of IFPs to AMPs permitted the tests to avoid any IFP and channel saturation, thereby focusing the testing more towards disk activity. The IFPs were split across two host channels to achieve a reasonable balance of activity. Tests were run on single, dual, triple, and quad DSUs per AMP. In all cases, 2.5 GB disk drives were used. The database used fallback as well as dual before and dual after journals, clearly the most disk intensive testing possible. Extra effort was needed in the case of the two spindle configuration to level the amount of data placed on each drive. This was accomplished by repeatedly loading useless data rows into the system until the second disk filled to the same capacity as the first drive. Once this was accomplished, the real database was loaded via a restore from tape which caused the data rows to be evenly balanced across the two spindles.

ON-LINE TRANSACTION TEST – SINGLE SESSION PER AMP VERSION

TRANSACTION PER SECOND IMPROVEMENT OVER SINGLE DSU



RESPONSE TIME IMPROVEMENT OVER SINGLE DSU



The tests were heavily dependent on the number of sessions per AMP chosen during the testing. Clearly, a single session per AMP does not drive the Model 4 AMP very hard. Consequently, the tests had to be run at five sessions per AMP in order to truly illuminate the effects of IPI emulation and multiple spindles. Even at five sessions per AMP, the CPU utilization never exceeded 60%.

When one session per AMP is driven with debit/credit transactions, the following performance improvements were revealed:

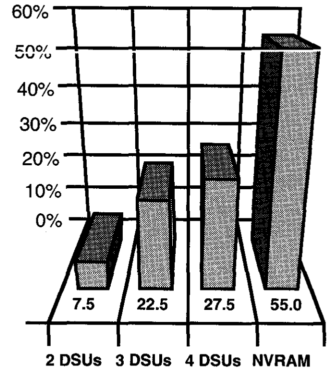
OLTP Improvements via Multiple DSUs per AMP
One Session per AMP

Compared to	**Txn AMP/sec**	**Response Time**	**I/Os per sec**
Single DSU	baseline	baseline	baseline
Dual vs. Single	+4.82%	4.20%	+5.96%
Triple vs. Single	+19.28%	+16.81%	+20.59%
Quad vs. Single	+25.30%	+20.17%	+24.88%
Triple vs. Dual	+13.79%	+13.16%	+13.81%
Quad vs. Triple	+5.05%	+4.04%	+3.56%

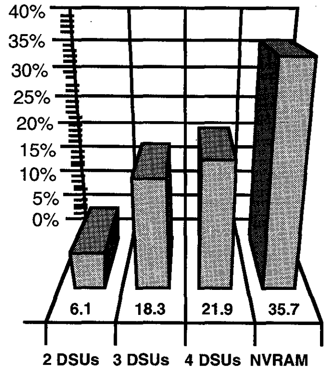
DBC/1012 Model 4 Performance Comparison

ON-LINE TRANSACTION TEST – FIVE SESSIONS PER AMP VERSION

TRANSACTION PER SECOND IMPROVEMENT OVER SINGLE DSU



RESPONSE TIME IMPROVEMENT OVER SINGLE DSU



Using the single DSU per AMP configuration as the baseline, the triple DSU configuration yields 19% more transactions per second, a 17% faster response time, and a 20% improvement in I/Os per second. Comparing the gains of the triple DSU configuration over the dual configuration, the gains are 14% more transactions, 13% faster response times, and 14% more I/Os per second.

As you can see, the rate of throughput improvement is not as high going from one to two drives as it is going from two to three drives in these tests. This is probably caused by three or four really "hot" cylinders – those containing the transient or permanent journals or possibly one of the data tables themselves. It is likely that a more linear performance leap (higher dual DSU numbers) would have occurred with more disk saturation and a higher number of sessions running.

OLTP Improvements via Multiple DSUs per AMP Five Sessions per AMP

Compared to	Txn AMP/sec	Response Time	I/Os per sec
Single DSU	baseline	baseline	baseline
Dual vs. Single	+7.50%	+6.08%	+6.20%
Triple vs. Single	+22.50%	+18.25%	+22.18%
Quad vs. Single	+27.50%	+21.90%	+25.84%
Triple vs. Dual	+13.95%	+12.95%	+15.05%
Quad vs. Triple	+4.08%	+4.46%	+3.00%
One DSU & NVRAM	+55.00%	+35.77%	-49.25%

The ability of the NVRAM to improve update performance is obvious from the five session throughput rates shown above. In these tests, the NVRAM dramatically reduces write activity by more than four to one in the fallback, dual after, dual before journal configuration. This is partially revealed in the 49% drop in I/Os per second. This produces a 55% improvement in the transaction rate which is almost twice what the four DSUs per AMP can achieve. Clearly, the NVRAM is an excellent on-line transaction and BulkLoad accelerator for applications that rely heavily on row-at-a-time database updates. On the other hand, the NVRAM option will affect only the updating side of an on-line or batch transaction. While it may seem that the NVRAM eliminates the need for multiple disk spindles, there is still a significant need to have the additional disk drives in order to overlap many read requests.

What lessons do these tests reveal concerning multiple disks per AMP? Clearly the AMPs-to-spindles ratio favored by these tests is the triple DSU per AMP configuration. This configuration provides the best use of the high CPU speed of the Model 4 while taking advantage of the overlapping seeks and rotational position sensing (RPS). This does not imply that adding a fourth DSU per AMP is of no value, only that the gain is not nearly as dramatic as the dual to triple DSU improvements. Were the Model 4 user focusing on OLTP to the exclusion of all else, the triple DSU per AMP configuration is one choice a DBC/1012 user should seriously consider. This is especially true if there is demand for additional disk storage but the number of users on the system is not increasing rapidly. Depending on the size of the DBC/1012 and budget constraints, the customer may instead choose to add additional AMPs rather than expand the number of DSUs per AMP. In some cases this is a cheaper alternative and produces the same performance results.

For example, let us assume an order entry application that has been measured running five transactions per second per AMP with two DSUs per AMP running on a 30 AMP Model 4 with 2.5 GB disks. If the customer wants to achieve a maximum throughput of 170 transactions per second, he has two choices: more DSUs or more AMPs. If an additional DSU (30) is added to each AMP, the corresponding 14% (triple vs dual) improvement will yield 0.7 additional transactions per second per AMP or 171 transactions per second. To raise the TP/s rate by adding AMPs, the customer would add four AMPs and eight DSUs. This adds four times five transactions per second yielding an overall rate of 170 transactions per second. In this example, the cost calculation shows the four AMP solution to be almost 20% cheaper. In such cases, the Systems Engineer must carefully analyze the RESUSE reports. If there is excess CPU capacity already in the system, additional spindles may be a more appropriate choice. Conversely, if the transactions use complex SQL and are compute bound, additional spindles may not help but additional AMPs will.

Considering that there was little difference between 1.2 GB and 2.5 GB disks in overall performance, this would tend towards selecting the proper number of spindles from a data volume and pricing standpoint. Budget permitting, the 2.5 GB disks are clearly the best choice in terms of price per megabyte, performance, and MTBF ratings. Nevertheless, favoring the correct number of spindles per AMP should be a primary consideration for DBC/1012 customers who expect large numbers of concurrent users on the system.

During the testing, RESUSE figures revealed that in the triple and quad configurations, there were seven to eight disk accesses queued at any given time. This means that the rotational position sensing capability of the firmware had the opportunity, on average, to select between two queued requests per DSU or between any one of the DSUs. This enabled the firmware to choose the optimum transfer strategy based which sector happened to be under the disk heads at the moment. Since we could not turn off the RPS capability, we were unable to measure the actual benefit derived.

4.4 Non-Volatile Disk Cache (NVRAM) – DSS

Nearly all tests in this document were run two more times with NVRAM attached to the 24 AMP Model 4 system. This was done to determine what performance improvements or regressions might occur. This was an especially critical test series because of the redesign of the AMP disk controllers to enable the IPI mode emulation and other special hardware improvements.

In the decision support tests, the NVRAM had almost no effect. Since these were not update tests, the NVRAM should not have affected a read only process. However, the AMP operating software maintains lists of data blocks currently in the NVRAM. Prior to reading the disk, these lists are searched in case the data block is in NVRAM and a disk read can be avoided. In our testing, this revealed a one or two percent deviation (above noise levels) from non-NVRAM testing. Some tests ran a little faster, some a little slower. Considering the benefits of NVRAM, these variations are insignificant.

5.0 General Queries

The general query suite consists of nine requests that are representative of SQL functions commonly used by Teradata customers. These involve everyday needs such as aggregations, prime index joins, table redistributions, and sorts. In the aggregation queries, the number of distinct values found in the selected columns are used as the sort key that controls the number of "roll-up" levels. In each case where aggregation (sum, max, average) occurs, the query returns a small number of summary rows called "buckets". The SQL for the tests can be referenced using the number in parenthesis following the paragraph title.

Overall, there were few surprises in these queries. They tended to contain a fairly even mix of CPU and disk utilization such that neither one is a dominating factor in response time. Performance improvements ranged from 1.8 to 2.1 times improvement averaging around 1.98 times better on the Model 4 versus the Model 3 Plus. When compared to the basic Model 3 system (64K cache & 4 megabytes of memory), the overall performance improvements average around 2.4 times faster on the Model 4 system.

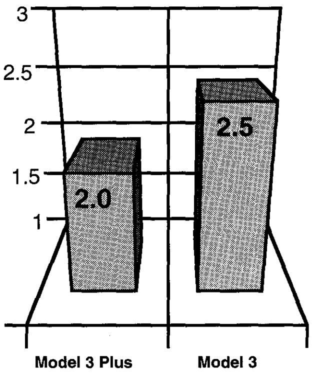
5.1 Aggregate/Sort 3 Groups, 1000 Categories (100)

This query is similar to retail applications that summarize units sold and dollars received based on store or departments within the store. Another common use would be in banking where accounts are summarized by categories of account-types. In a broad sense, this query is a category-totals report.

This query does an all-AMP scan of 50,000 rows, sorting them into three groups and summing the distinct variations. This may be considered a simple version of a roll-up report that users will commonly execute. Running 60% disk use to approximately 30% CPU utilization, this query is somewhat limited by disk. This resulted in a performance improvement on the Model 4 of 1.8 time faster than Model 3 Plus. One thousand rows are returned to the requester in this test.

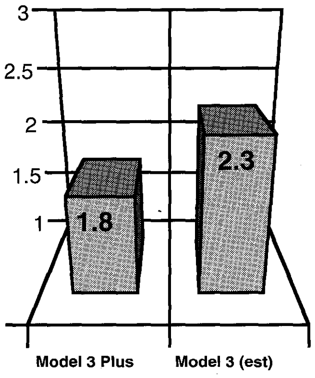
AVERAGE SPEEDUP OF GENERAL QUERIES

MODEL 4 OVER MODEL 3

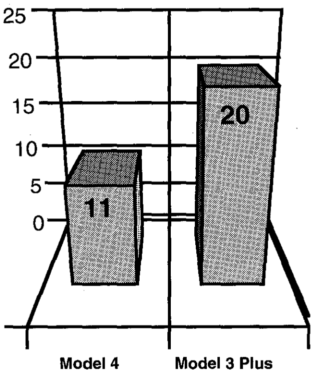


AGGREGATE & SORT - 3 GROUPS, 1000 CATEGORIES

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEEDS - SECONDS

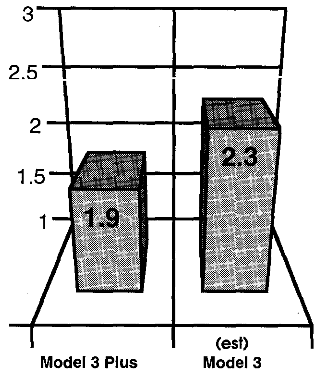


5.2 Aggregate/Sort with 4 groups, 10,000 Buckets (125)

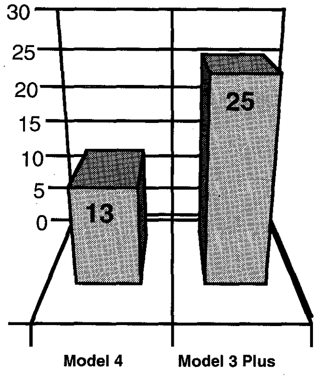
Like the previous query, this test is appropriate for any sorting of details into categories and reporting the totals. Because of the large number of category "buckets", this type of query is probably found more frequently in month-end or week-ending batch runs where a more detailed report is required. An example might be a summary of insurance claims by office and policy type, sales by product code, or perhaps service charges by rate and service location.

This query does an all-AMP scan of the same 50,000 rows as the previous query. It also sorts and aggregates all columns selected, which in this case produces four groups of roll-up results (like the prior query, the roll-up answer is the maximum value of a column). By adding the additional CPU intensive load (the additional selection & aggregate column), performance of the 10,000 row result was 1.87 times faster on the Model 4 versus the Model 3 Plus. The transfer of 9,700 rows to the host took two seconds longer than in the previous query which transferred only 1,000 rows to the host. 9,789 rows are returned to the requester in this test.

AGGREGATE & SORT –
4 GROUP, 10,000 CATEGORIES
MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES



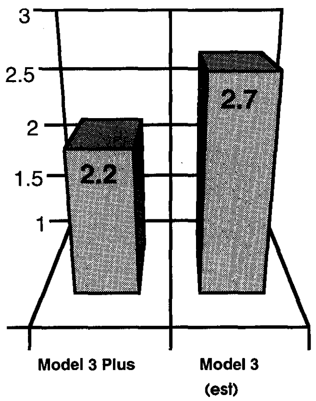
5.3 Prime Index Merge Joins (200)

An efficient and commonly used relational feature, this type of join finds a broad range of uses where a large table is matched to a medium sized table on primary indexes. One obvious use is the combining of history records to the account or product. Some examples include: Insurance – claim and account status tables; Manufacturing – product and shipping history; Communications – customer and call history.

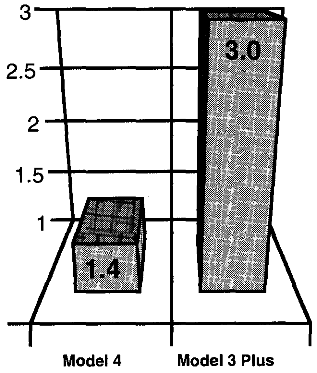
This query invokes an all-AMP scan and merge join of a one million row table. In each case, the unique primary index (UPI) is used in an equality test for the join. CPU utilization on the Model 3 Plus runs around 66% while disk utilization remains at 40%. On the Model 4 system, CPU and disk utilization runs at 53% and 37% respectively. With the higher utilization of the CPU caused by the "where" clause and the join processing, overall performance of the Model 4 system improved 2.16 times over the Model 3 Plus. 95 rows are returned to the requester.

PRIMARY INDEX MERGE JOINS - 1 MILLION ROWS

MODEL 4 IMPROVEMENT OVER MODEL 3

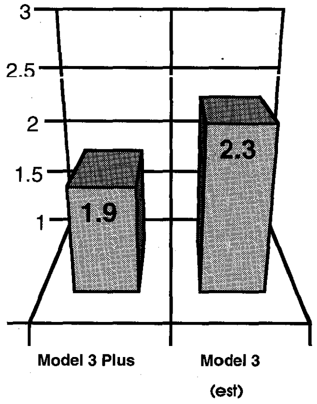


EXECUTION SPEEDS - MINUTES

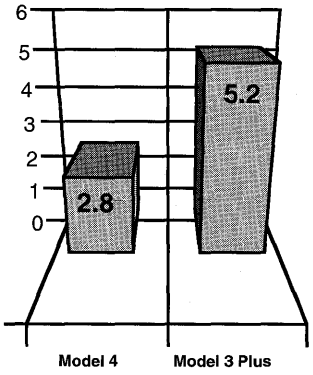


PRIMARY INDEX TO FOREIGN KEY JOIN - 1 MILLION ROWS

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTIONS SPEED - MINUTES



5.4 Prime Index to Foreign Key, redistribution of 1 million row table (202)

One of most commonly used relational joins is the foreign key variation. In this case, the rows do not share a common identifier but rather are "related" through an imbedded foreign key. This type of join connects two "peers" in the database. Hence it is used to match flights and passengers, patient and medical service, products and suppliers, or employees and departments.

This query invokes an all-AMPs scan and merge join of one million rows to the primary index of a 50,000 row table. The constraints in the "where" clause causes a row redistribution of all one million rows. This is done to place the rows on the same AMP where the primary index for the "joined-to" row exists. During this period, Model 4 Ynet utilization runs fairly high at approximately 650 I/Os per second, CPU utilization at 76%, and disk utilization at 23%. The Model 3 Plus runs the same disk use but the CPU runs over 90% and cannot drive the Ynet nearly as fast, averaging 400 I/Os per second. This query ran 1.86 times faster on the Model 4 than on the Model 3 Plus.

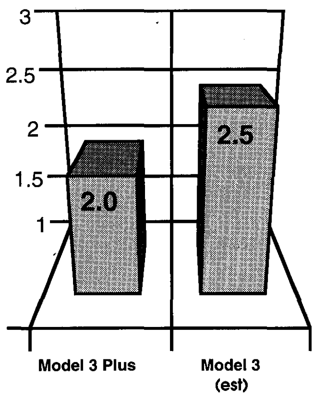
5.5 Both tables redistributed - 50% of each table Selected (214)

The following three queries represent a "join" that does not use the primary index of either table. This causes a disk and Ynet intensive redistribution of rows. Common uses of this relational capability would be matching last names across differing account types (customer information systems) or matching subscribers by household address (mail order or publishing lists).

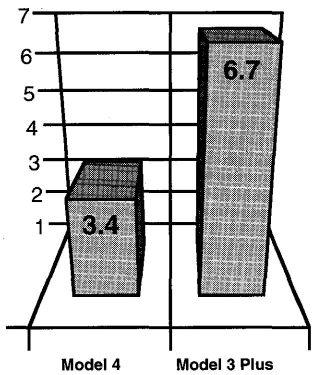
This query scans two tables, both having one million rows. Approximately 50% of each table is selected and redistributed via the Ynet. The redistribution drives the Ynet at 700 I/Os per second on the Model 4, 390 I/Os per second on the Model 3 Plus. Once redistribution completes, a merge join with constraints is performed. Owing to the higher throughput, Model 4 CPU utilization is 30% lower and disk use 12% higher than the Model 3 Plus. This results in an elapsed time 1.97 times faster than the Model 3 Plus. 299 rows are returned to the requestor in this test.

50% REDISTRIBUTION JOIN – 1 MILLION ROWS EACH TABLE

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES

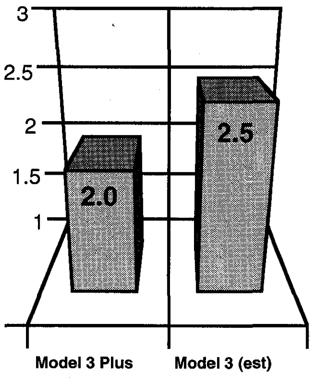


5.6 Both tables redistributed – 100% of each table Selected (214B)

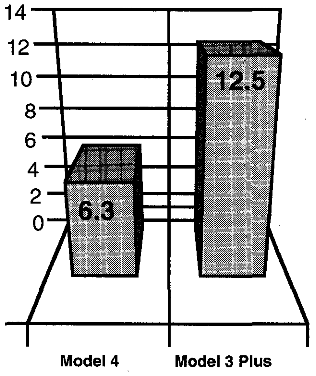
This query is identical to the previous query (214) excepting that some selection constraints have been removed. The selection criteria requires an equal join between two fields which are not the primary indexes, a fairly common use of the SQL join. The result is that both tables are 100% redistributed via the Ynet. Because of the CPU saturation, the Model 3 Plus is unable to drive the Ynet beyond 420 I/Os per second whereas the Model 4 system achieves a whopping 772 I/Os per second. This results in a Model 4 elapsed time which is 1.99 times faster than the Model 3 Plus. 1,071 rows are returned to the requestor in this test.

BOTH TABLES REDISTRIBUTED – 100% OF EACH TABLE SELECTED

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES

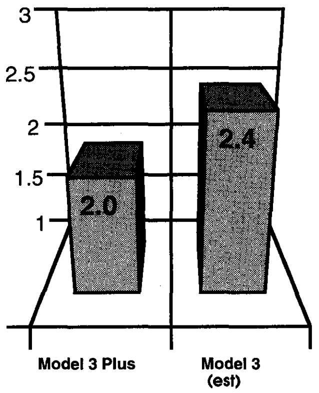


5.7 Both Tables Redistributed – 1 Million and 10 Million Rows (304)

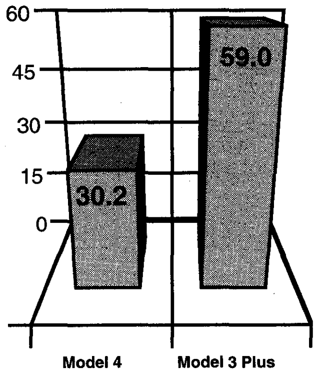
This query turns up the heat on the system. This query is identical to the prior test (214B) excepting that one of the tables contains ten million rows. This high throughput join ran about 80% CPU and 21% disk utilization, hitting the Ynet at an average of 700 I/Os per second on the Model 4. The Model 3 Plus was again limited to 400 I/Os per second on the Ynet. Disk use on the Model 3 Plus matched the Model 4 disk usage but CPU utilization stayed between 87 and 90%. This resulted in a 1.95 times improvement in elapsed time for the Model 4 system over the Model 3 Plus. 1,071 rows were returned to the requestor.

BOTH TABLES REDISTRIBUTION – 1 AND 10 MILLION ROWS (304)

MODEL 4 IMPROVEMENT OVER MODEL 3



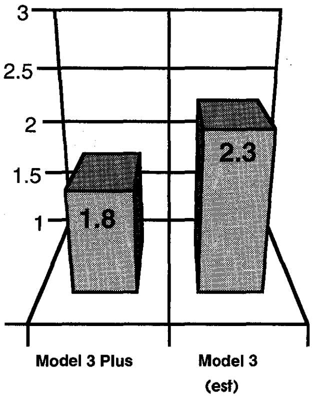
EXECUTION SPEED - MINUTES



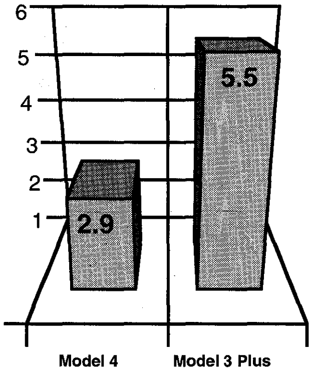
5.8 Scan-table test – 10 Million rows (310A)

SCAN 10 MILLION ROWS

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES



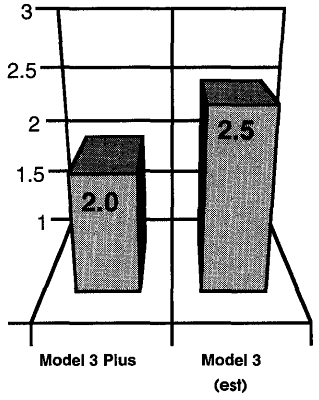
A large percentage of truly adhoc queries involve scanning large tables for any occurrence of the target data. Often, this is a spontaneous business need that demands examining tens of millions of records. Because of the infrequency of the specific request, it is often inappropriate to maintain indexes on the field with the selection constraints. Examples of this type of query might be "list the number of widgets revision H that were returned defective in January" or "what was the average age of the customers who responded to our product promotion last quarter".

This query was simply an all rows scan of ten million records. Again, we have a high disk utilization test wherein the CPU use is quite low. On the Model 4, this ran a 22/80 percent ratio of CPU to disk whereas on the Model 3 Plus a 39/63 percent ratio existed. This resulted in a 1.84 times performance improvement in elapsed time in the Model 4 versus the Model 3 Plus. 49 rows were returned to the requestor.

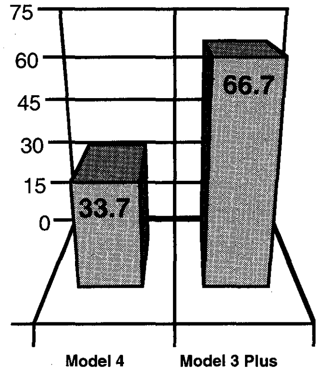
5.9 Aggregate/Sort, 3 groups, 1000 buckets – Ten million rows (313)

AGGREGATE & SORT – 10 MILLION ROWS , 3 GROUPS, 1000 CATEGORIES

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES

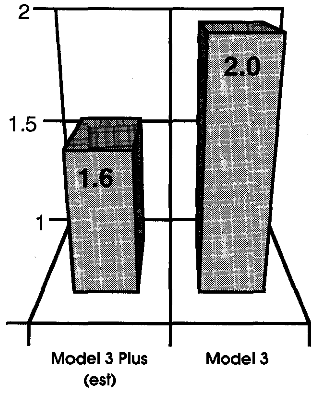


Like the queries discussed earlier, this request has numerous uses in production environments as well as in adhoc summaries. This type of query is used to group information from very large tables for easy analysis. It answers such questions as "provide a revenue summary by region, state, and city" or "summarize cargo weight by date, destination hub, and carrier".

This test is a repeat of the first test in the query suite. The key difference is that it sums ten million rows instead of 50,000. Like the first test, an all-AMPs scan is done to summarize the column INSEQ grouping the results on the first three column's values. This produces a saturation of the Model 3 Plus AMP while running a low 14% use of the disk. On the Model 4, disk use is approximately the same while CPU utilization settles at 80%. The elapsed time results show the Model 4 is 1.97 times faster than the Model 3 Plus. 1,000 rows are returned to the requestor.

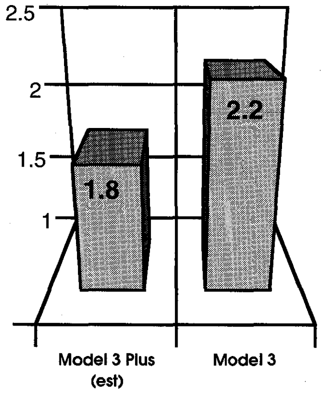
FASTLOAD PHASE 1 – 200 CHARACTER ROWS

MODEL 4 IMPROVEMENT OVER MODEL 3



FASTLOAD PHASE 2 – 200 CHARACTER ROWS

MODEL 4 IMPROVEMENT OVER MODEL 3



6.0 FASTLOAD

This section relies entirely on work done by the Teradata's Performance Group within the development organization. Further details can be obtained from the R&D Performance Group.

Testing was done comparing a Model 4 system to a Model 3 with 64K cache and 4 megabytes of memory per processor. The tests were conducted using 500,000 rows as input. One series of tests were run at 100 byte row size, the other series at a 200 byte row size.

Fastload phase 1 is heavily dependent on exclusive use of mainframe MIPS since this is the phase which reads the tape cartridge and passes rows to the Teradata Director Program (TDP). Since Teradata customers do not usually have the same host, Phase 1 performance is considered less meaningful. Also note that the host time spent in Phase 1 tends to dilute the improvements gained by the Model 4. In simpler terms, we did not expect nor did we get performance improvements in Phase 1 processing that matched other results. Consequently, Phase 1 processing improved 1.56 times for 100 byte rows and 1.96 times for 200 byte rows as measured in transactions per second per AMP.

On the other hand, Phase 2 processing is the sorting and building phase of Fastload which is purely a DBC/1012 event. Performance improvements in Phase 2 processing should, and did, roughly approximate improvements measured in other tests. For 100 byte rows, the Model 4 ran 2.08 more transactions per second per AMP than the Model 3. In the case of the 200 byte rows, Model 4 performance climbed to 2.22 times that of the Model 3.

Current customers running Level 3 software will experience even more dramatic performance boosts when moving to a Model 4 system. Because of the upgrade to Level 4 software, the customer will receive the added benefit of changes to the sort logic which affects Phase 2 of Fastload. While the two performance gains are not additive, the result for the Release 3.xx customer will still be significantly higher than the figures reported above.

Fastload performance improvements are most useful to customers who collect large volumes of data from other computers, often from outside the company. This is particularly true of point-of-sale operations, satellite weather mapping, stocks and bonds trading, and money wire transfers to name a few. The Fastload performance improvements are an enabling technology for DBC/1012 users attempting to control the ever increasing quantities of transaction data accumulated daily.

7.0 On-Line Transaction Processing

The benchmark results described in this section are primarily derived from testing done by the Product Development Performance Group and are called P-TXNs (Test 1) in this section. Testing done by the Marketing team are called M-TXNs (Test 2) in this section.

These tests are based on the old debit/credit transaction model used internally at Teradata for relative performance comparisons. These tests are not comparable to any industry standard benchmark because we use different SQL requests, database sizes, and performance constraints. These tests are not optimized to drive the mainframe nor the DBC/1012 to peak performance transaction rates. Instead, the tests are designed to show relative performance rates, a wholly different benchmark target. Consequently, any extrapolation of these numbers done to compare to industry benchmarks would be misleading and incorrect.

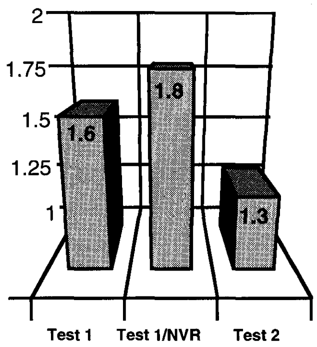
Testing done by the Product Development Performance Group (P-TXNs) used a 2x4x8 Model 3 system with 64K cache, 8 Mb memory, and two 1.2 GB disks per AMP. The Model 4 system used was a 3x4x8 with two 1.2 GB disks per AMP. Testing was performed on both configurations with and without Non-Volatile Disk cache using single after image journalling and no fallback (NNS tests). The Marketing team (M-TXNs) used an 8x8x16 Model 4 with two 1.2 GB disks per AMP and a 2x4x8 Model 3 with 64K cache, 8 Mb memory, and two 1.2 GB disks per AMP. The M-TXNs were run with fallback and no journalling excepting transient journals (FNN tests). In each case, the Model 4 system uses a higher ratio of IFPs to AMPs which is necessary to maintain a similar MIPS-to-MIPS ratio on the Model 4 system. In all cases, five session per AMP were used to drive the AMPs. While the differing configurations might suggest an unfair comparison, results are measured on a "per AMP" basis which has proven to be an accurate metric in tests of this nature.

In the P-TXNs tests, the transactions per second per AMP improved 1.61 times on the Model 4 versus the Model 3 system (61% improvement). At the same time, the response time of the Model 4 was 63% of the response time of the Model 3 transactions (37% improvement). In these tests, the Model 3 ran a 68/52% ratio of CPU-to-disk utilization whereas the Model 4 ran nearly the reverse at 53/73% CPU-to-disk utilizations. Considering the results of the multiple disk drive testing earlier in this document, the Model 4 in this test has excess capacity that could be utilized by adding additional disk drives.

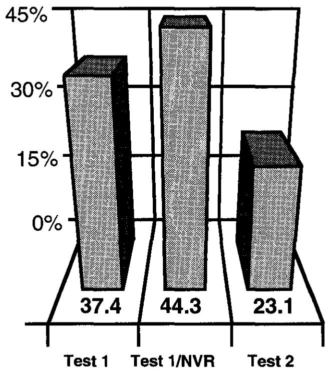
Using the P-TXN configurations, Non-Volatile Disk Cache was added to both systems This produced a 73% reduction in disk I/Os on both the Model 3 and Model 4 systems. In this configuration, the Model 4 system ran 1.77 more transactions per second per AMP while the response time

MODEL 4 IMPROVEMENT OVER MODEL 3

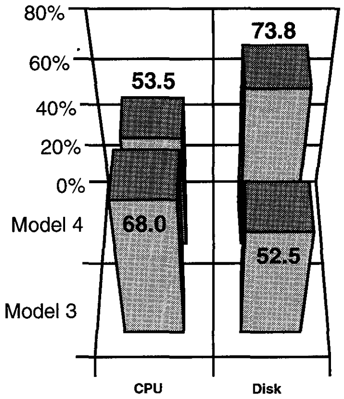
IMPROVEMENT FACTOR



RESPONSE TIME IMPROVEMENT

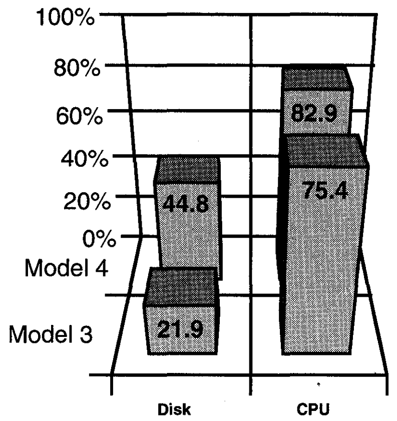


CPU & DISK UTILIZATION - TEST 1 (P-TXN)

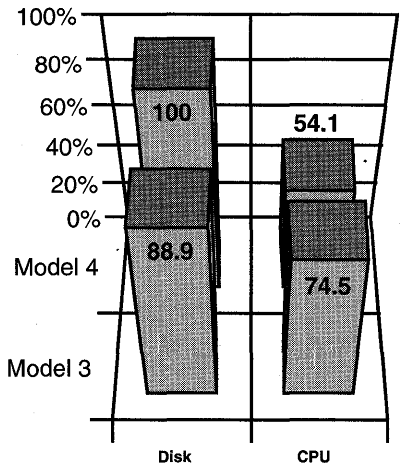


MODEL 4 IMPROVEMENT OVER MODEL 3

CPU & DISK UTILIZATION WITH NVRAM - TEST 1 (P-TXN)



CPU & DISK UTILIZATION WITH NVRAM - TEST 2 (M-TXN)



of the Model 4 was 55% of the Model 3 response times (45% improvement). As expected, the CPU-to-disk utilizations changed dramatically with the Model 3 operating at a 75/21% ratio and the Model 4 at 82/45% ratio. Clearly, the Non-Volatile Disk Cache option reduces the time spent waiting for disk I/O and delivers more of the Model 4 AMP speed to the end user.

In the M-TXNs tests, the Model 4 system produced 1.3 times more transactions per second per AMP (30% improvement) and produced a response time 76% of the Model 3 response time (24% improvement). CPU-to-disk utilizations ran 75/90% on the Model 3 and 54/100% on the Model 4. In the M-TXN configurations, the additional I/Os generated by fallback versus the single after image journalling drives up the disk I/Os dramatically. Since disk I/O becomes the major component of the transaction, the benefits of the Model 4 CPU speed are diluted. Consequently, it is necessary to add either additional disk drives or Non-Volatile Disk Cache to the system in order to optimize the Model 4 system in this testing configuration.

One other way to view these tests is to compare the P-TXNs Model 4 with Non-Volatile Disk Cache (maximum throughput) to the P-TXNs Model 3 without the disk caching (baseline throughput). Although this is a somewhat unequal comparison, it is a potential upgrade path for many DBC/1012 customers. This comparison results in the Model 4 running 2 times the number of transactions per second per AMP (100% improvement) at a response time 49% of the Model 3 baseline (51% improvement). Clearly, OLTP applications find their "best fit" configuration with the Model 4 and Non-Volatile Disk Cache.

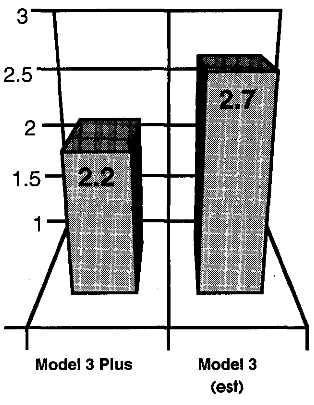
Comparing these results to the decision support and batch results, it becomes apparent that the OLTP transactions in this test are limited by disk speed. Since the SQL used is simple prime index updates, the DBC/1012 is not utilizing any relational capabilities and behaves more like an access method than a relational database manager. Similar studies show that as more journalling activity is added to the test, the performance improvements of the Model 4 are further diluted. This stands to reason since transactions which are disproportionately weighted towards disk I/O receive less benefit from the Model 4 CPU speed. In contrast, on-line complex processing which uses more of the relational features of SQL tends towards better use of the Model 4 CPU speed and larger improvements in transaction rates.

8.0 System Support Activities

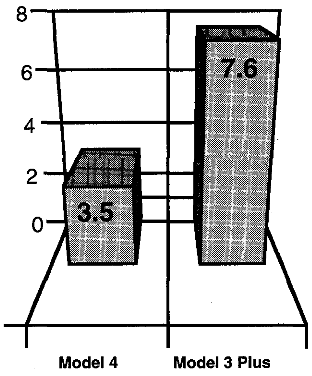
These tests reveal important performance improvements that are of great interest to the Database Administrator and application programmer. Each of these individuals tends to repeat these types of functions many times during the course of a month. Additionally, several of these functions are part of daily production batch jobs in many DBC/1012 installations. Consequently, the improved performance of the Model 4 system shown here translates directly into staff productivity as well as faster batch processing.

INSERT SELECT (71)

MODEL 4 IMPROVEMENT OVER MODEL 3



EXECUTION SPEED - MINUTES



8.1 Insert Select (71)

Insert-Select is used for many database activities and is particularly useful for creating copies or subsets of tables. Programmers often use this to create test databases by copying selected portions of a production database into a smaller version of production for quality testing of programs. This test copies one million rows from a table with fallback to a table without fallback. This function ran 2.15 times faster on the Model 4 than on the Model 3 Plus.

8.2 Create Index

The Build Index process showed an overall improvement of 2.00 times over the Model 3 Plus. Because additional processing is needed when there are many unique values in an index column, the larger tables with more distinct values showed slightly better throughput improvement than the smaller tables with few distinct values. A quick review of the execution times show that creating a non-unique secondary index on ten million rows took under six minutes while in most tests the create index ran under one minute. This would indicate that creating and dropping an index for the purpose of aiding a single batch job is a viable method for speeding up nightly processing in many instances.

In the following table the columns should be interpreted as follows:

- Test number is a randomly assigned value for tracking purposes.
- Types are UPI = unique primary index, NUSI is non-unique secondary index.
- Row count is the number of rows in the table analyzed.
- Distinct values is the number of unique values in the column analyzed.
- Model 3 Plus and Model 4 times are runtime in minutes, seconds, and thousandths.
- Speed up factor is the improvement provided by the Model 4 over Model 3 Plus.

Model 4 Improvement Over Model 3 Plus
Create Index

Test	Type	Row Count	Distinct Values	Fallback	Model 3 Plus Time mm:ss.tt	Model 4 Time mm:ss.tt	Speedup Factor
50	USI	1,000,000	1,000,000	Yes	03:38.00	02:04.41	1.75
52	NUSI	1,000,000	10	Yes	01:04.82	00:30.72	2.11
53	NUSI	1,000,000	50	Yes	00:59.23	00:29.72	1.99
54	NUSI	1,000,000	1,000	Yes	00:58.35	00:29.26	1.99
55	NUSI	1,000,000	100,000	Yes	01:09.06	00:32.94	2.10
57	NUSI *	1,000,000	1,000,000	Yes	01:29.42	00:40.90	2.19
500	NUSI	10,000,000	50	No	09:51.20	05:01.04	1.96
501	NUSI	10,000,000	100,000	No	12:50.01	05:54.65	2.17
502	NUSI	10,000,000	1,000,000	No	13:23.28	05:52.2	2.28
80	USI	1,000,000	1,000,000	No	03:32.26	02:21.18	1.50
82	NUSI	1,000,000	10	No	01:05.32	00:30.35	2.15
85	NUSI	1,000,000	100,000	No	01:08.69	00:34.20	2.01
87	NUSI	1,000,000	100,000	No	01:25.04	00:38.34	2.22
4	NUSI	5,000,000	100,000	No	06:22.94	03:43.37	1.71
5	NUSI *	5,000,000	500	No	05:16.44	02:48.00	1.88

* Multiple Column Index

8.3 Collect Statistics

Collect statistics, like most supposedly disk bound activities, reaps enormous benefits from the Model 4 CPU speed. During statistics collection, the system maintains internal lists of unique column values it has encountered. Each time a row is processed, the system matches the current column value to the internal list. When there are few distinct values in a column, the collect statistics becomes somewhat of a disk reading race. In this case, the improvement of the Model 4 CPU is truly limited by the disk speed. But in the cases where there are many unique values in a column, the Model 4 CPU zooms through the internal lists, outrunning the Model 3 Plus by a wide margin. This is most obvious in the collection of unique primary index statistics since every index value must be added to the internal list. It is also apparent in large tables where there are thousands or millions of values. This type of statistics collection is common in many applications where a value frequently repeats throughout a column such as account type, inventory location, or fare class.

In the following table the columns should be interpreted as follows:

- Test number is a randomly assigned value for tracking purposes.
- Types are UPI = unique primary index, NUSI is non-unique secondary index.
- Row count is the number of rows in the table analyzed.
- Distinct values is the number of unique values in the column analyzed.
- Model 3 Plus and Model 4 times are runtime in minutes, seconds, and thousandths.
- Speedup factor is the improvement provided by the Model 4 over Model 3 Plus.

Collect Statistics
Model 4 Improvement Over Model 3 Plus

Test	Type	Row Count	Distinct Values	Model 3 Plus Time mm:ss.tt	Model 4 Time mm:ss.tt	Speedup Factor
58	UPI	1,000,000	1,000,000	03:29.20	01:32.89	2.25
10	UPI	5,000,000	5,000,000	17:07.68	07:33.95	2.26
503	UPI	10,000,000	10,000,000	35:42.57	15:38.02	2.28
59	USI	1,000,000	1,000,000	03:04.95	01:23.37	2.22
62	NUSI	1,000,000	1,000	00:13.32	00:09.31	1.43
63	NUSI	1,000,000	100,000	03:49.57	02:18.17	1.66
504	NUSI	10,000,000	50	00:10.00	00:08.02	1.25
505	NUSI	10,000,000	100,000	11:02.33	06:49.62	1.62
506	NUSI	10,000,000	1,000,000	39:42.41	22:48.32	1.74
11	NUSI	5,000,000	100,000	09:45.23	05:41.91	1.71
12	NUSI *	5,000,000	500	00:11.22	00:08.44	1.33

* Multiple Column Index

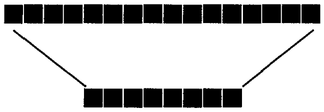
9.0 CONFIGURING MODEL 4 SYSTEMS

Let us first clearly state the ground rules for configuring Model 4 systems:

- Approach Model 4 sizing with the same diligence used with Model 3.
- Most of what applies to Model 3 applies to Model 4.
- There are no simple guidelines that by themselves will lead to the proper configuration.
- Always favor Model 4 systems for the more critical production applications.

The first general guideline in configuring Model 4 systems is that you will need approximately half as many AMPs as on a Model 3. Since the Model 4 is generally 2.4 times faster than the Model 3, this should provide approximately 1.2 times as much throughput for approximately 20% less cost (fewer AMPs and cabinets). The halved configuration will also provide better response times than the Model 3. If the customer's objective is to achieve dramatically better response times, you will need more than half the number of AMPs. Similarly, when upgrading from a Model 3 Plus to a Model 4, start by configuring the system with somewhat more than half as many AMPs. Of course, viewed another way, you can get more than twice as much work done by replacing all Model 3 AMPs with Model 4 AMPs.

Model3
16 AMP's Throughput = X, Cost = Y



Model 4
8 AMP's, Throughput = 1.2 X, Cost = 0.8 Y

This halving of AMPs is an appropriate approach for large set processing applications such as decision support and batch. It is inappropriate for OLTP and BulkLoad processes which are row-at-a-time activities. Any random I/O processing which is disk constrained, will need more than half as many AMPs. This is because this type of processing is more dependent on disks and channel time than on CPU time. The configured number of AMPs should be calculated using the same techniques used for Model 3 systems. Realistically, you should also maintain a similar MIPS-to_MIPS ratio between the IFPs and AMPs if your application is expected to be IFP constrained on the Model 3 system.

By halving the number of AMPs we imply a doubling of the amount of DASD space per AMP. Given that the Model 4 is typically greater than twice the speed of the Model 3 Plus, this maintains an even balance of MIPS to DASD.

With the extra CPU speed and the obvious ability to drive the Ynet harder, there appears to be no loss of parallelism resulting from halving a Model 3 Plus configuration. As with the Model 3 systems, configurations containing less than four AMPs are not recommended because of the loss of the effects of parallel processing.

As with the Model 3 systems, installations which demand high levels of concurrent activity will benefit from additional AMPs. This reduces the number of prime index SQL requests being serviced per AMP. While response time is not changed in OLTP activities by adding AMPs, the additional AMPs will be useful in sustaining a consistent response time as the user population increases.

Dual DSUs per AMP are suggested as a minimum configuration whenever possible regardless of the disk drive capacity (1.2 GB or 2.5 GB). Any installation which expects to have more than two concurrently running DSS requests will clearly benefit from the additional spindles. For those customers who perform many row-at-a-time activities such as BulkLoad and OLTP, additional DSUs per AMP allow the full benefit of RPS as well as pushing the AMP CPU to capacity. Given the results of the various tests, the Model 4 AMP will surely become disk starved in many single DSU per AMP instances. Customers with heavy concurrency loads should consider running three DSUs per AMP. Those who simply require additional DASD without a corresponding increase in users logged on should consider the quad DSU per AMP arrangement. These choices must be carefully weighed against adding additional AMPs instead of increasing the DSUs per AMP ratio. In many cases, additional AMPs can achieve the same performance objective at a lower overall cost to the customer.

Customers who expect to do large amounts of database updating should install the Non-Volatile Disk Cache (NVRAM) option. The effects of NVRAM in on-line and batch updates makes it an excellent price/performance booster. The guidelines for choosing to use NVRAM have not changed between the Model 3 and Model 4 DBC/1012 systems.

In summary, to estimate a Model 4 size based on Model 3:

- Halve the number of AMPs in CPU bound or balanced applications
- Try to begin with two DSUs per AMP
- Always weigh the throughput of multiple DSUs against additional AMPs
- Use NVRAM in any OLTP or BulkLoad intensive application

APPENDIX A - SQL USED DURING TESTING

Model 4 Tests

The answer set was exported to a dummied host file in order to suppress the effects of the mainframe I/O. Answer sets were restricted by applying cross-table Selection after the join often reducing the spool to less than 50 rows. CPU-intensive queries use a retlimit of 1. Select DATE TIME statements were included before and after each query. Between queries using the same tables, a dummy Select of 1 million rows was inserted to flush all rows from AMP memory to avoid residual rows being found in the next query test.

71 Insert/Select

```
Insert into cab.tnf
        select * from cab.t1m;
```

100Aggregate/Sort, 3 groups, 1000 buckets

```
Select ch1ae,in10,in20,max(inseq) from t50k
        group by 1,2,3 order by 1,2,3 having max(inseq) gt 2490;
```

125Aggregate/Sort with 4 groups, 10,000 buckets

```
Select ch1ae,ch1aj,in10,in20,max(inseq) from t50k
        group by 1,2,3,4 order by 1,2,3,4 having max(inseq) gt 2490;
```

200Prime Key Joins

```
Select t1m.ch15city, Tnf.ch15city from t1m,tnf
        where t1m.inseq = tnf.inseq
        and (t1m.in500k + tnf.in500K) gt 999900;
```

202Prime Key to Foreign Key, redistribution of 1 million row table

```
Select t1m.ch15city, t50k.ch15city from t1m,t50k
        where t1m.in50k = t50k.inseq
        and (t1m.in500k + t50k.in10K) gt 505000;
```

214Both tables redistributed – 50% of each table Selected

```
Select t1m.ch15city, tnf.ch15city from t1m,tnf
        where t1m.ch1sex = ‘f’
        and  tnf.in10 gt 4
        and  tnf.in1m = t1m.in1m
        and  (t1m.in500k + tnf.in500k) GT 999000;
```

214B Both tables redistributed – 100% of each table Selected

```
Select  t1m.ch15city, tnf.ch15city from t1m,tnf
    where tnf.in1m = t1m.in1m
    and  (t1m.in500k + tnf.in500k) GT 999900;
```

304 Both Tables Redistributed – 1 Million and 10 Million Rows

```
Select  t1m.ch15city, t10m.ch15city from t1m,t10m
    where t10m.in1m = t1m.in1m
    and (t10m.IN1m + t1M.IN500K) GT 1490000;
```

310A Scan-Table test – 10 Million rows

```
Select  inseq From t10M where d08seq lt 50.00;
```

313 Aggregate/Sort, 3 groups, 1000 buckets – Ten Million rows

```
Select  ch1ae,in10,in20,max(inseq) from t10m
    group by 1,2,3 order by 1,2,3 having max(inseq) GT 499995;
```

900 Rollup Report

```
Select  SSS_JD (smallint) , DCD (char(1)),
    sum(PPP_102_AM) (decimal(11,0)) , sum(PPP_102_QY)
    (integer) , sum(PURCH_102_AM) (decimal(11,0)),
    sum(PURCH_102_QY) (integer) , sum(CASH_102_AM)
    decimal(11,0)) , sum(CASH_102_QY) (integer),
    (PPP_FULL_QY + PPP_N_DU_QY) (smallint),
    (PPP_SKIP_QY + PPP_LESS_QY + PPP_MINM_QY +
    PPP_PLUS_QY), (((FC_102_AM*100 - .5) mod 1) +  .5)
            (smallint),
    AUTH_IN (char(1)) , BILL_IN (char(1)),
    index(substr(CHGOFF_DT,3,4),'01')
    (smallint) , sum(BAL_102_AM) (decimal(11,0)),
    sum(ICA_102_AM) (decimal(11,0)) , sum(ICA_102_QY)
    (integer) , sum(FC_102_AM) (decimal(11,0)),
    sum(CASH_AMT_102_AM) (decimal(11,0)),
    sum(ICA_AMT_102_AM) (decimal(11,0)) ,
    sum(index(substr(AMF_BILL_DT,3,4),('01')) *
    AMF_BILL_AM)            (decimal(11,0)) ,
    sum(LC_102_AM)  (decimal(11,0)) ,
    sum(OLAMT_102_AM)
    (decimal(11,0)) , count(*) (integer),
    sum(CC_BILL_102_AM)
    (decimal(11,0)) , sum(CHGOFF_BAL_AM)
    (decimal(11,0))
    From   ACCT_TRANS
    GROU BY 1,2,3,4,5,6,7,8;
```

901 Floats & Dates

```
Select * From t1m
    where
    (in1m       = 1.1E6
    or    in1m = 1.2E6
    or    in1m = 1.3E6
    or    in1m = 1.4E6
    or    in1m = 1.5E6
    or    in1m = 1.6E6
    or    in1m = 1.7E6
    or    in1m = 1.8E6
    or    in1m = 1.9E6
    or    in1m = 999)
    and (DATE - DAY) GT 100
    and (DAB + 15) LT (DAY + 101)
    and (DATE - DAB) GT 10;
```

902 Disk Intensive 1 - Multi-Statement SQL

```
Select  *  from t1m, t100
    where t1m.IN100 = t100.inseq
    and (t1m.inseq + t100.inseq) lt 10  ;
Select *  from t5m,t50
    where t5m.in50 = t50.inseq
    and (t5m.inseq + t50.inseq) lt 10  ;
Select *  from t13m, t50k
    where t13m.in50k = t50k.in50k
    and (t13m.inseq + t50k.inseq) lt 20  ;
```

903 Disk Intensive 2 - Multi-Statement SQL

```
Select * from t10m    where d08seq lt 50.00;
Select * from t5m     where d08seq lt 50.00;
Select * from t1m     where d08seq lt 50.00;
```

** 4 Build a NUSI on 5 Million rows 100,000 Distinct Values*

```
Create index(in100k) on T5m;
```

** 5 Two-part NUSI, 500 Distinct values*

```
Create index(in10,in50) on T5m;
```

** 50 Build a USI on a 1 million row table with fallback*

```
Create unique index(D08seq) on T1m;
```

** 52 Build a NUSI on 1 million row table, fallback, 10 distinct values*

```
Create index(ch1aj) on T1m;
```

**** 53 Build a NUSI on 1 million row table, fallback, 50 distinct values***

```
Create index(in50) on T1m;
```

**** 54 Build a NUSI on 1 million row table, fallback, 1000 distinct values***

```
Create index(in1k) on T1m;
```

**** 55 Build a NUSI on 1 million rows fallback, 100,000 distinct values***

```
Create index(in100k) on T1m;
```

**** 57 Three-part index, 100,000 distinct values***

```
Create index(in20,in50,in100) on T1m;
```

**** 80 Build a USI on a 1 million no fallback***

```
Create unique index(D08seq) on Tnf;
```

**** 82 Build a NUSI on 1 million row table, no fallback, 10 distinct values***

```
Create index(ch1aj) on Tnf;
```

**** 85 NUSI on 1 million row, no fallback, 100,000 distinct values***

```
Create index(in100k) on Tnf;
```

**** 87 Three-part index, 100,000 distinct values***

```
Create index(in20,in50,in100) on Tnf;
```

**** 500 NUSI on non fallback 10 million row table, 50 distinct values***

```
Create index(in50) on t10m;
```

**** 501 NUSI, non fallback, 10 millions rows, 100,000 values***

```
Create index(in100k) on T10m;
```

**** 502 USI on non fallback 10 million row table, 1 million values***

```
Create index(in1m) on t10m;
```

Collect Statistics

```
* 10 Collect Statistics on T5m Index(Inseq);
* 11 Collect Statistics on T5m Index(IN100K);
* 12 Collect Statistics on T5m Index(IN10,IN50);
* 58 Collect statistics on t1m index(inseq);
* 59 Collect statistics on t1m index(d08seq);
* 61 Collect statistics on t1m index(ch1aj);
* 62 Collect statistics on t1m index(in1k);
* 63 Collect statistics on t1m index(in100k);
* 64 Collect statistics on t1m index(in50);
* 88 Collect statistics on tnf index(inseq);
* 89 Collect statistics on tnf index(d08seq);
* 91 Collect statistics on tnf index(ch1aj);
* 93 Collect statistics on tnf index(in100k);
* 503 Collect statistics on t10m index(inseq);
* 504 Collect statistics on t10m index(in50);
* 505 Collect statistics on t10m index(in100k);
* 506 Collect statistics on t10m index(in1m);
```

Teradata

100 North Sepulveda Boulevard
El Segundo, California 90245
(213) 524-5000