SANDIA REPORT

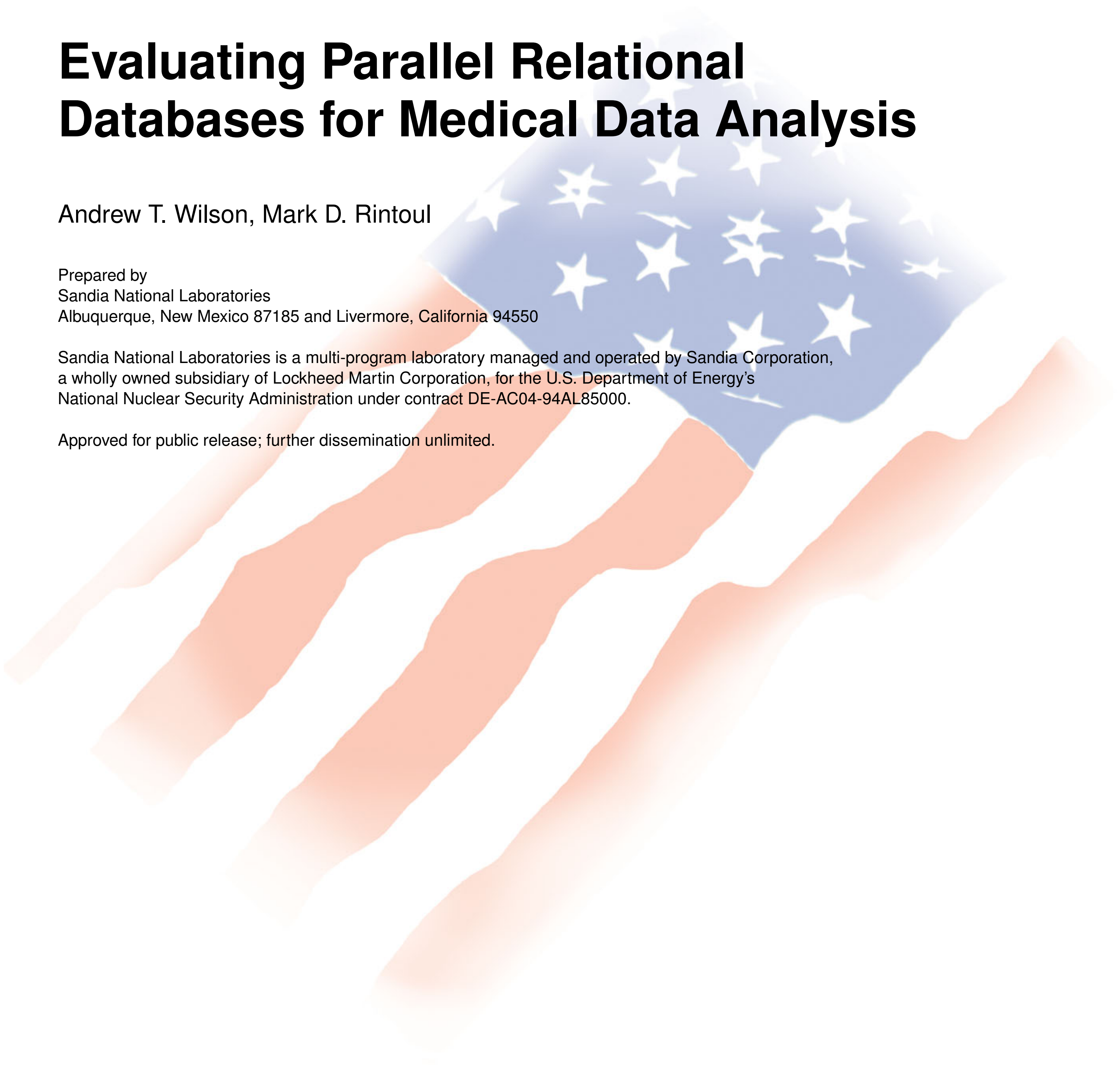
SAND2012-1881
Unlimited Release
Printed March 2012

Evaluating Parallel Relational Databases for Medical Data Analysis

Andrew T. Wilson, Mark D. Rintoul

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Sandia National Laboratories

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2012-1881
Unlimited Release
Printed March 2012

Evaluating Parallel Relational Databases for Medical Data Analysis

Andrew T. Wilson
Scalable Analysis and Visualization
Sandia National Laboratories
P.O. Box 5800, MS 1326
Albuquerque, NM 87185-1326
atwilso@sandia.gov

Mark D. Rintoul
Discrete Math and Complex Systems
Sandia National Laboratories
P.O. Box 5800, MS 1326
Albuquerque, NM 87185-1326
mdrinto@sandia.gov

Abstract

Hospitals have always generated and consumed large amounts of data concerning patients, treatment and outcomes. As computers and networks have permeated the hospital environment it has become feasible to collect and organize all of this data. This raises naturally the question of how to deal with the resulting mountain of information.

In this report we detail a proof-of-concept test using two commercially available parallel database systems to analyze a set of real, de-identified medical records. We examine database scalability as data sizes increase as well as responsiveness under load from multiple users.

Acknowledgment

We are grateful to Bill Goldman and Bryan Trujillo at Sandia National Labs for their heroic efforts in arranging for the care and feeding of the XtremeData and Netezza systems. Without their persistence in navigating a procedural maze this project might not have happened at all.

We are also grateful to Richard H. Goldstein, MD and his colleagues at RGI for granting us access to the de-identified clinical data. There is no substitute for working with real-world data sets with their quirks, inconsistencies and unexpected insights.

We acknowledge with appreciation the patient assistance rendered by technical staff at IBM/Netezza and XtremeData. Helping a customer optimize system performance without access to the system or the data is a perplexing task that they handled with consistent skill.

Finally, we thank Geno Valente at XtremeData and Joe Weeks at IBM/Netezza for the loan of the systems we tested.

Contents

List of Figures

List of Tables

Summary

With the growth of ubiquitous networking, hospitals and clinics are beginning to collect and store all patient data from admission and discharge records to the second-by-second results from heart rate monitors. The resulting volumes of information swamp the capabilities of installed databases. However, they embody an amazing wealth of information. The recent development of highly parallel database systems enables the complex analytics we need to handle tasks arising from clinical monitoring as well as medical research.

In this report we evaluate the performance of two commercially available parallel, multi-processor database systems for their performance on de-identified clinical data. The first is an IBM/Netezza Twinfin-12, a single-rack hardware-plus-software system. The second is an 8-node (half-rack) installation of dbX software from XtremeData, Inc.

We tested both single-user scalability and multi-user performance. In single-user mode we expect to see linear scalability of execution time with respect to the data size. In a multi-user scenario we want to discover how heavy a load can be tolerated while query times remain on the same order of magnitude as with just a few users.

Both systems exhibited consistently linear scaling of execution times with respect to the size of the data. In some cases, especially when comparing execution times for the 5x and 1x data sets, scaling was actually faster than linear.

In single-user mode, queries took between 4 and 58 times longer to execute on the XtremeData system than on the Netezza system. The ratio was the smallest on the simplest queries and grew larger with more complex analytics.

In multi-user mode, queries on the XtremeData system were between 4 and 40 times slower with 10 concurrent sessions than with just 1, then between 3 and 12 times slower with 75 connections than at 10. Queries on the Netezza system were between 3% and 21% slower at 10 concurrent connectiosn than at 1, then between 2x and 6.6x slower at 75 users than at 10.

Chapter 1

Introduction

1.1 Motivation

Computers and electronic sensors have been ubiquitous in US clinics and hospitals for some years now. As these devices gain network connectivity we gain the ability to collect all manner of data about each patient. On one extreme these data sets include hand-entered observations made by nurses on rounds, lab test results, and hospital admission and discharge records. On the other, higher-resolution information such as heart rate measurements taken each minute can be gathered directly from the monitors in an intensive care suite. It is not much of a stretch to envision capturing even high-resolution EKG or EEG information as a matter of routine. The state of the art in hospital care is moving inexorably toward storing every available piece of data as part of a patient's record.

This wealth of data offers the unprecedented opportunity for medical researchers and clinicians to ask analytical questions that draw upon data from entire populations of patients at once. At the risk of hyperbole, we believe that access to so much information has the potential to transform at a deep level the practice of medicine and selection of treatment.

Before we can change the world, though, there are a few more mundane problems to address. The data sets we contemplate above are several orders of magnitude larger than most clinical data collected today. This dramatic increase in scale requires an equally dramatic increase in data storage and analysis capability.

1.2 The Project

Our goal in this project is to evaluate the performance and scalability of commercially available parallel relational database systems (also known as *multiprocessor parallel (MPP)* databases) in the context of medical data. We focus specifically on so-called OLAP systems (On-Line Analytic Processing) designed to correlate, distill and summarize large amounts of data. These stand in contrast to OLTP systems (On-Line Transaction Processing) that are designed to process very large numbers of small queries with very high throughput.

In this report we evaluate the IBM/Netezza 1000 database [2] and the XtremeData dbX system

[3]. RGI Informatics, LLC worked with both vendors to arrange a short-term loan of both systems. Staff from RGI and Sandia National Labs worked together to delineate a set of technical evaluation criteria. These evaluations were then executed at Sandia. This report summarizes our findings.

1.2.1 What We Tested

We consider the following two dimensions in our evaluation.

1. **Scalability With Data**: We want to see system performance vary linearly with the size of the data. When we run a set of queries on data sets A and 2A (2A is twice the size of A) we expect the queries on 2A to take twice as long as those on A.
2. **Responsiveness Under Load**: We want to discover how well each system copes with concurrent workloads from increasing numbers of users. We do not have quantitative expectations here as we do for data scalability: rather, we want a sense of how many users each system will tolerate before becoming painfully or even unusably slow.

We were fortunate enough to be able to work with real data in this project. RGI supplied 48GB of de-identified data gathered from a single hospital over a period of 9 years. This data set gave us the freedom to construct our own domain-specific benchmark queries rather than trying to adapt an existing OLAP benchmark drawn from some other application.

1.3 Organization

The rest of this report is organized as follows. In Chapter 2 we describe the two systems we evaluated. In Chapter 3 we describe briefly the properties of our sample data set and how we modified it to better test system scalability. In Chapter 4 we detail our test queries and in Chapter 5 we present our results. Appendix A contains the raw data used to create the charts in Chapter 5. We conclude in Chapter 6.

Chapter 2

Systems Evaluated

2.1 Netezza and XtremeData Systems

We compared two parallel SQL databases in this study. The first was a Twinfin-12 from Netezza Corporation [2]. [1] This system has 32TB of uncompressed disk capacity and built-in compression that typically results in 80-120TB of effective storage for user data. The Twinfin-12 is separated into a storage pool containing 96 disks and a processing pool comprising 12 processing blades (S-blades in Netezza terminology). Table 2.2 has details. Netezza systems ship as self-contained appliances combining hardware and software.

The other system we tested is an 8-node installation of dbX software supplied by XtremeData, Incorporated [3]. XtremeData focuses on a software-only product that customers can install on hardware of their own choosing. To ensure a good match between hardware and software, XtremeData supplied their product already installed on a half-rack cluster. See Table 2.1 for detailed specifications.

We note that there are many other companies that offer already or plan to offer parallel database products. Their omission from this study does not indicate any judgement of their respective merits.

2.2 System Security

Both database systems were hosted in Sandia's Computer Science Research Institute (CSRI) for the duration of testing. They were placed on a network segment without access to the Internet or to other Sandia resources. Furthermore, login access was restricted to the system consoles (located in an access-controlled machine room) and to a single workstation from which tests were conducted.

The sample data was treated as Third-Party Proprietary Information, a category of "sensitive but unclassified" information that requires strict access control based on need-to-know. As with the databases, only the staff members directly involved with testing were permitted access to the decrypted data.

[1] IBM has recently acquired Netezza. As part of this process the Twinfin has been renamed the IBM Netezza 1000.

XtremeData Hardware Specifications

Component	Specifications
Front End (x1)	2x 6-core AMD Opteron, 2.4GHz 32GB memory 16 1TB 7200 RPM disks (RAID6)
Storage Node (x8)	2x 4-core AMD Opteron, 2.6GHz 32GB memory 12 1TB 7200 RPM disks (RAID6)
Internal Network	Infiniband DDR

Table 2.1. Hardware specifications for the half-rack cluster supplied by XtremeData. While the dbX system is a software-only product, XtremeData supplied hardware as well to ensure an optimal mix of components.

Netezza Hardware Specifications

Component	Specifications
Front End (x2)	2x 4-core Intel Xeon, 2.4GHz 24GB memory
S-Blade (x12)	2x 4-core Intel Xeon, 2.4GHz 24GB memory 4 FPGAs with 2 stream engines apiece
Disks (x96)	1TB SAS, 7200 RPM Redundancy provided by database
Internal Network	Gigabit Ethernet

Table 2.2. The IBM/Netezza Analytics Appliance is sold as a self-contained unit. The S-blades are processing units that handle all database processing. The storage pool is managed by the database OS instead of via a mechanism such as RAID arrays.

2.3 Scrubbing the Data

In order to preserve the confidentiality of the de-identified clinical data we scrubbed the disks on both database systems before returning them to the vendors. We opted for a 7-pass wipe inspired by US Department of Defense Standard 5220.22 [4]. Although we are not aware of any formal technical requirements for secure erasure of de-identified medical data, a 7-pass wipe is commonly accepted as a secure method of data destruction.

Chapter 3

Test Data

RGI supplied Sandia with a set of anonymized sample data derived from records of real patients admitted to hospital between 2004 and mid-2011. The data were organized into tables covering the following subject areas:

- `sl_labs`: Lab test results
- `sl_observations`: Observations collected by nurses and doctors
- `sl_surgery` and `sl_surgery_detail`: Surgical procedures
- `sl_procedures`: Other procedures
- `sl_hospital_discharges`: Hospital admission, discharge and disposition
- `sl_device_data`: Data collected from automated monitors (heart rate, respiration, etc.)

3.1 Original Schema

We were given a simple schema for the data tables. Fields that contained a timestamp used the SQL `TIMESTAMP` or `DATETIME` data type. Fields that contained a measured quantity used the `DOUBLE PRECISION` data type. All other fields, including those containing numeric patient identifiers were of type `VARCHAR` (variable-length text). No primary or foreign key relationships were supplied.

3.2 Our Schema

Modern CPUs contain special-purpose instructions for comparing integers and floating-point (real) values. These instructions run thousands of times faster than comparisons between two strings. Since execution speed was of paramount concern in this study we chose to modify the schema with more specific data types to take advantage of this speedup.

3.2.1 Modifications to the Original

Wherever a column was labeled as a timestamp or a real-valued quantity, we kept the data type. Otherwise we tested every value in every column to identify (1) columns containing only integer data, (2) columns containing only numeric data (integer and real-valued), (3) columns containing only timestamp data and (4) columns containing no `NULL` values. We changed the data type for a column to `INTEGER`, `DOUBLE PRECISION` or `TIMESTAMP` and annotated with `NOT NULL` wherever these tests so indicated. We did not include a margin for error in these tests. Even if all rows in a column except one satisfied (e.g.) the test for integrality, that single row was enough to force the data type to remain as it was.

3.2.2 Deliberate Omissions from the Schema

We did not add any indices or primary/foreign key constraints to the database. Key constraints were unnecessary because the tests we performed do not involve the maintenance of referential integrity. Indices were omitted because of the properties of the Netezza and XtremeData systems as well as our example queries. Specifically, Netezza software does not use indices and XtremeData experts indicated that our sample queries would not benefit from indexing.

3.3 Unusual Data

This data set has a few unusual properties that challenge the system optimizations typically made for the most common business applications.

1. **One Big Table**: The Device Data table contains over 95% of all the bytes in the data set. Since all of our example queries access this table it becomes a bottleneck under load.

2. **Narrow Rows**: The device data table is also awkwardly formatted from a database optimization perspective. Each row contains a relatively small amount of data in a relatively small number of columns. This magnifies the impact of any system overhead associated with storing a single row and incurs more work during table access than if each row were more fully populated.

3. **Text Fields**: The `sl_observations` and `sl_device_data` aggregate categories of data that would often be divided into separate tables. These categories are distinguished by the contents of a `VARCHAR` field that must usually be matched somewhere in the middle to extract items of interest.

3.4 Scaling Up

The original data set takes up 48 gigabytes on disk. This is trivially small for modern parallel databases with capacities of 30-100 terabytes in a single rack. In order to provide a realistic workout as well as test the scalability of both systems we created replica data sets 5, 10, 50 and 100 times larger than the original. In the remainder of this report these are referred to as 1x (the original data), 5x, 10x, 50x and 100x. In order to simulate a larger patient population we modified the patient ID fields (`admission_id`, `patient_id`, `cis_admission_id` and `source_admission_id`) but left all other fields unchanged. The net effect is that for every patient admission in the original set, the 100x data set contains 100 patients with distinct IDs but exactly the same hospital stays and outcomes.

Chapter 4

Sample Queries

In this section we detail the queries that were used to test database performance. We explain why they were chosen for this test and discuss briefly what makes them easy or difficult in light of the data.

4.1 Basic Tests

The first set of queries contains simple "sanity check" tests meant to demonstrate that the database does not do anything egregiously inefficient when faced with large data and low cardinality. They are as follows:

1. `dd_time_bounds1`: Compute an element count and the range of dates where data was collected for each hospital location in the `sl_device_data` table.

2. `dd_time_bounds2`: The same query but with counts and date ranges computed with respect to a different field.

3. `obs_time_bounds`: Compute element count and range of dates for every area in the `observations` table.

These are easy queries. The only challenge they pose lies in the summary (an SQL `GROUP BY` operation) over hospital locations. Since we only ever see five distinct values in that column the work is difficult to partition across a parallel database system with many processing elements. Different systems may choose to handle that in different ways.

4.2 Long-Running Real Queries

The second set of tests contains a few examples of long-running queries gathered from already-installed databases at RGI. These queries take minutes or hours on RGI's systems and serve as a coarse indication of how fast parallel databases are compared to single-processor systems.

1. `ventilators`: Extract clinical observations about patients on ventilators. This query is a straightforward sub-select with a text wildcard in the `WHERE` clause.

2. `hr_bundle`: Extract heart rate data for patients with at least one "bundle" observation. This is another sub-select with a text wildcard and a `BETWEEN` constraint (i.e. heart rate must be between 120 and 300) and a nested `IN` list to identify patients of interest.

3. `duration_of_stay`: Compute the beginning and end of each patient's stay in each location in the hospital. This query uses neither wildcards nor nested `SELECT` statements but does `GROUP BY` three different fields.

These queries involve a lot of processing but are structurally simple. They parallelize well since there are no complex nested sub-select statements or correlated sub-queries. We include them to permit direct comparison with legacy systems.

4.3 Ad-Hoc Queries

Near the beginning of testing, Richard H. Goldstein, MD from RGI visited Sandia for hands-on experimentation with the database platforms and the data. As part of that experimentation we devised several queries of clinical interest. We include two examples as part of our testing.

1. `hgb`: Extract the first five hemoglobin lab results for each patient during each admission. Store the results in a table where each row contains information on all 5 tests for a particular patient.

2. `hr_episodes`: Find episodes of elevated heart rate (greater than 100 beats per minute) for all patients. Compute the duration of each episode. Store details (begin time, end time, duration, patient and location ID) in a table.

These queries were chosen because they were not feasible using already-installed systems. Notably, the version of MySQL installed at RGI lacked the window analytic functions `PARTITION BY`, `LEAD`, `LAG` and `OVER` introduced in the ISO SQL:2003 standard [1]. While it is technically possible to to write these queries without those constructs, the contortions required mean that they will not execute in any reasonable amount of time.

Both Netezza and XtremeData support SQL window analytic functions. Although the dbX software lacked this functionality on initial installation, XtremeData supplied a software upgrade that added the capability.

Chapter 5

Results

In this section we describe our testing methods and results. We also list common features that we did *not* test due to the constraints of the Sandia network environment.

5.1 Testing Environment

All tests were executed using command line utilities (`nzsql` and `xdudb` respectively for the Netezza and XtremeData systems) over a remote login session. No other load was on either system apart from the tests being conducted. Furthermore, query results were not written to disk. We were only interested in the time required to execute the query and return final results to the calling program.

5.1.1 Single-User Testing

We tested the systems' scalability in the face of increasing data size by running the same queries on the original data as well as copies 5, 10, 50 and 100 times larger. These data sets are named 1x, 5x, 10x, 50x and 100x in all of the charts and tables below (Figures 5.1 through 5.5 and Tables A.1 through A.5 in Appendix A). In order to minimize the influence of system noise we executed each query five times in succession. We report the average of these times as the final result for each query.

5.1.2 Multi-User Testing

We tested database performance under multi-user load by opening many connections to the database and executing a plausible query load through each connection. We automated this using a Unix shell script [1] originally provided by Netezza system engineers and modified at Sandia to work with both systems. We reviewed the script thoroughly to ensure that it contained no Netezza-specific code and made minor changes to fix minor bugs and adapt it to XtremeData's system environment.

[1]Specifically, this script uses the `expect` utility to simulate interactive terminal sessions.

The plausible workload arises from observation of real-world query loads. By and large, queries fall into three different classes. The first class contains short-running "lookup" queries that retrieve some small information with little or no analysis. The second class contains longer-running "report" queries that require some summary and correlation but are nevertheless straightforward. The third class ("deep thought") contains analytic queries of arbitrary complexity. These typically come from business intelligence tools or individual users adept at writing SQL. Realistic workloads generally comprise a majority of "lookup" queries, a minority of "report" queries, and a very small handful of "deep thought" queries. Moreover, the size of the "deep thought" workload increases very slowly as the overall workload grows because there are relatively few users engaged in exploratory data analysis.

We simulate these three classes by dividing our test queries according to their execution time. The shortest-running test queries form the "lookup" class. The longer-running test queries compose the "report" class. The two ad-hoc queries (`hgb` and `hr_episodes`) compose the "deep thought" class.

During testing we divide our connections into three pools corresponding to the classes listed above. Each connection draws randomly a list of queries and executes them one after another with a brief pause between each submission. We time the execution of every query submitted during a two-hour testing window. The average execution time for each distinct query is reported in Figures 5.6 through 5.12. Raw data is again in Appendix A.

5.1.3 Vendor Query Optimization

Tuning and optimization of queries and data often lead to dramatic improvements in performance and scalability. Before gathering data, we invited engineers at Netezza and XtremeData to inspect anonymized versions of the schema and test queries and make suggestions on how to improve them to perform well on their respective systems. There were two constraints imposed. First, the queries had to return the same results as in their original form. Second, system-level tuning parameters were fair game but the data sets themselves could not be changed. At no time did we permit access to the real schema or to the data in any form, although we did answer questions such as "How many unique values are in column A?" or "Can this string comparison be replaced with a simpler equality test?".

Both vendors took advantage of this opportunity. Their suggestions yielded substantial improvements in query performance (up to 5x). Since this kind of optimization is standard practice in deployed environments, we used the optimized queries for all of our data collection.

5.1.4 Correction for Hardware Size

The Twinfin-12 supplied by IBM/Netezza is a full-rack system. The dbX installation from XtremeData is only half a rack. We correct for this in our reported results by halving all the execution times that we report for the XtremeData system. This is plausible if we assume perfectly linear scaling with hardware capacity and data size. Our results (see especially Figure 5.4 support this decision.

XtremeData also suggested that using the newest available hardware might yield another 2x improvement in speed. We believe that it may be more complicated than that, especially in light of the fact that most of the individual components in the XtremeData cluster (see Table 2.1) are on par with or faster than their counterparts in the Netezza system. Moreover, hardware specifications change so frequently that any number we choose would be inaccurate within weeks. We therefore decline to include a correction for newer hardware.

5.2 Single-User Scalability Results

We begin with a summary of the single-user results. On all queries in single-user mode, the Netezza system was between 4 and 58 times faster than the XtremeData system. The speedup tended to vary more between different queries than it did within a single query on different data sizes.

Figure 5.1 shows all of the query times for all data sets on both systems. In the rest of this section we break the data out more interpretable subsets.

Figure 5.2 shows the single-user execution times for the Netezza system only. We see here that despite the wide difference in execution time between separate queries, the execution time for each query rises in the same manner as data sizes increase. We will see in Figure 5.4 that this represents almost perfectly linear scalability.

Figure 5.3 shows the single-user execution times for the XtremeData system only. As with the Netezza system, query time varies widely across different queries but scales in similar fashion as the data size increases.

Figure 5.4 illustrates that both systems scale linearly with data size. Its entries compare the observed scaling behavior with a linear ideal. For example, we would expect a single query run against the 50x data set to take five times longer than the same query run against the 10x data set. We divide the 50x execution time by the 10x execution time to get an observed scaling factor, then divide that by the expected scaling factor to produce the entries in Table A.4 and Figure 5.4. Values larger than 1 indicate slower-than-linear scaling. This can arise in many ways, from resource contention to poor choice of algorithms or even differences in data movement. Values smaller than 1 indicate faster-than-linear scaling. This is typically the result of per-query bookkeeping costs being amortized over larger amounts of processing.

Finally, we ran a selected subset of our test queries on a single-processor MySQL system in production at RGI for a rough idea of the capacity of parallel databases in comparison with the current infrastructure. Figure 5.5 shows the results. The simpler queries such as `dd_time_bounds` are 70-350 times faster on a parallel database. The difference on more complex queries such as `duration_of_stay` and `ventilators` is even more dramatic. Furthermore, these are the *easy*

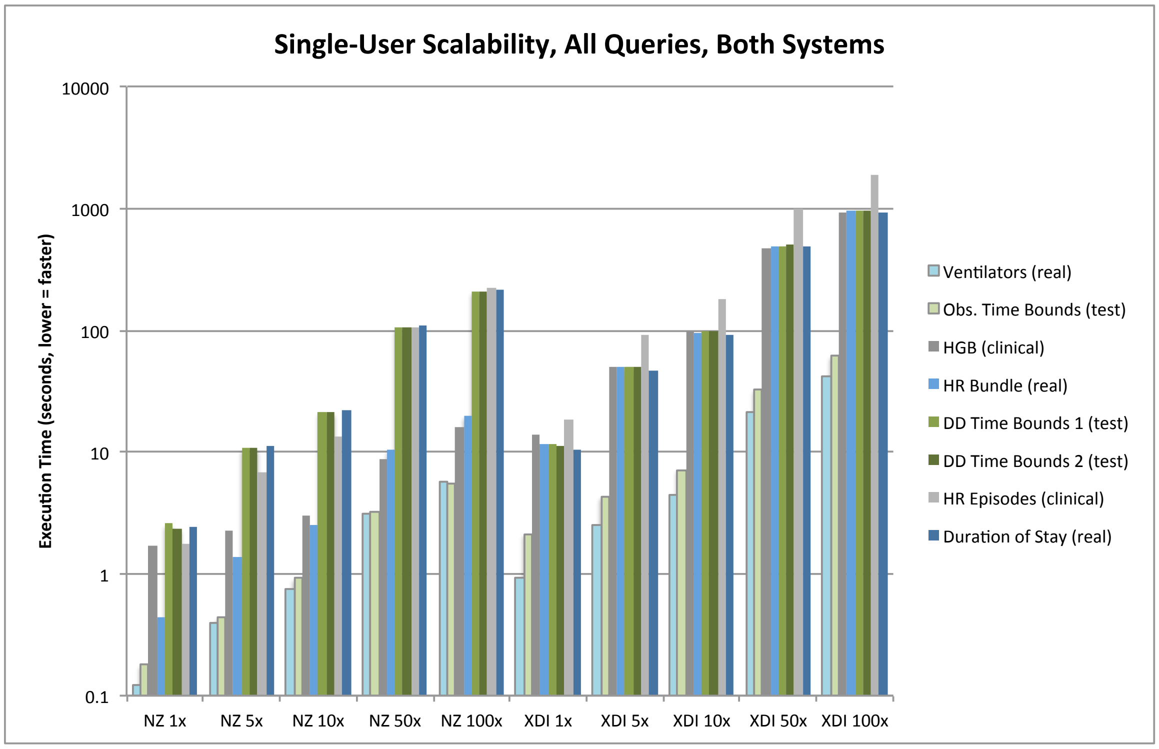


Figure 5.1. Single-user scalability test, all queries, both systems. Each query was executed on each of 5 different data sets. 5 trials of each run were averaged to minimize the effects of system noise. The Netezza system is between 4x (`dd_time_bounds 1`, all sizes) and 58x (`hgb`, 100x data set) faster than the XtremeData system on all queries.

queries. The version of MySQL installed at RGI lacks the window analytic primitives required to run the more complex `hgb` and `hr_episodes` queries. While it is possible, strictly speaking, to rewrite these queries in strict ANSI SQL:1999 the results would be so slow as to be infeasible. In effect, massively parallel databases make possible queries that are out of reach with single-system implementations.

5.3 Multi-User Performance Results

It is difficult to make scalability predictions under multiuser conditions because of the complex strategies different systems employ to manage resource allocation. Netezza's database system, for example, has user- and group-level query priority as well as minimum and maximum system resource guarantees. The dbX system does not have either of these mechanisms but does have a per-session priority setting. There are also likely to be many system-level tuning parameters

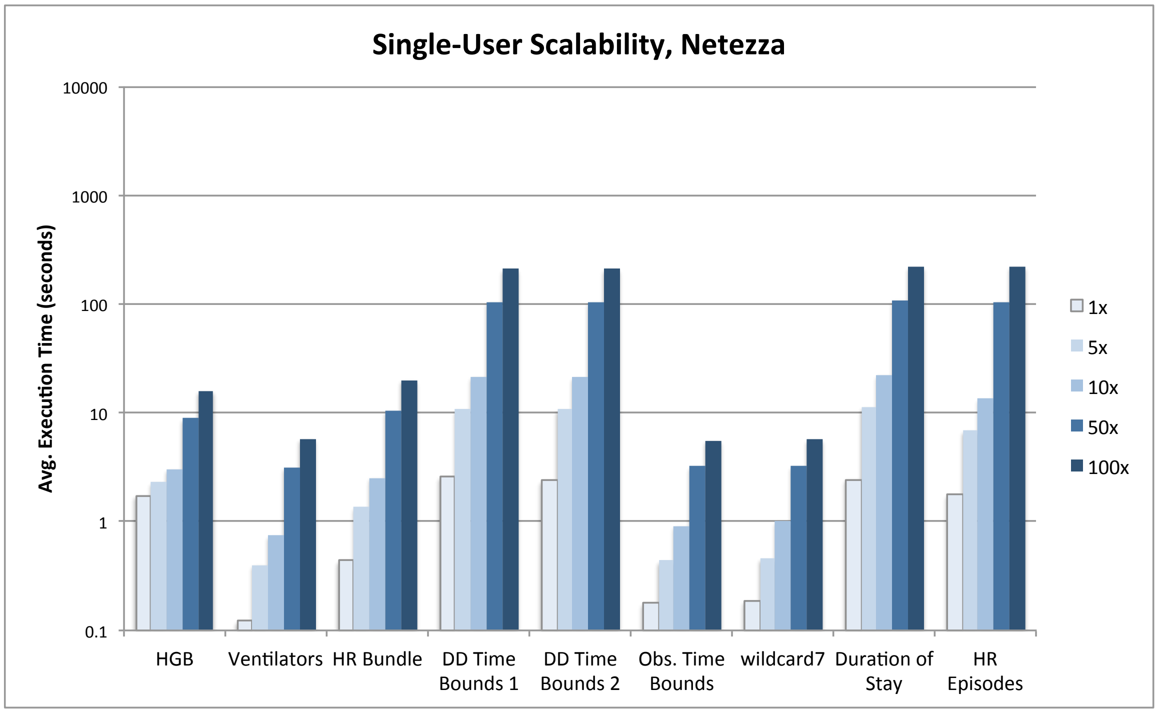


Figure 5.2. Scalability for the Netezza Twinfin. Each query was executed on 5 different data sets. 5 trials were averaged to minimize the effects of system noise.

that will further affect performance under load. Finally, there is a threshold at which a multi-user system bogs down and spends so much of its time trying to manage the workload that it is unable to make useful progress on any given query. Rather than compare the systems' performance to a specific numeric target we observe their behavior to judge how responsive they are under different levels of load. In short, we recommend treating the multiuser results cautiously. The results are quite clear but the underlying causes are not.

We begin with an overview of multiuser performance in Figure 5.6. There are two particularly notable features visible here. First, 10 concurrent users on the Twinfin see the same performance as if each user had the entire system to herself. This indicates that the system is not yet fully loaded. By comparison, the XtremeData system is running at 100% utilization at 10 users. When we reach 50 users, queries on the Netezza system take at most 10 times longer to execute than they do in single-user mode. Queries on the XtremeData system take between 10 and 50 times longer than in single-user mode. Please see Tables A.7 and A.6 for details.

In Figures 5.9 through 5.12 we compare the two systems side by side for each different load level (10, 30, 50 and 75 users). Testing beyond 100 users was not possible due to a system-level limit of 99 concurrent connections in the XtremeData dbX software. When we tried to test with 95 concurrent connections the dbX software crashed.

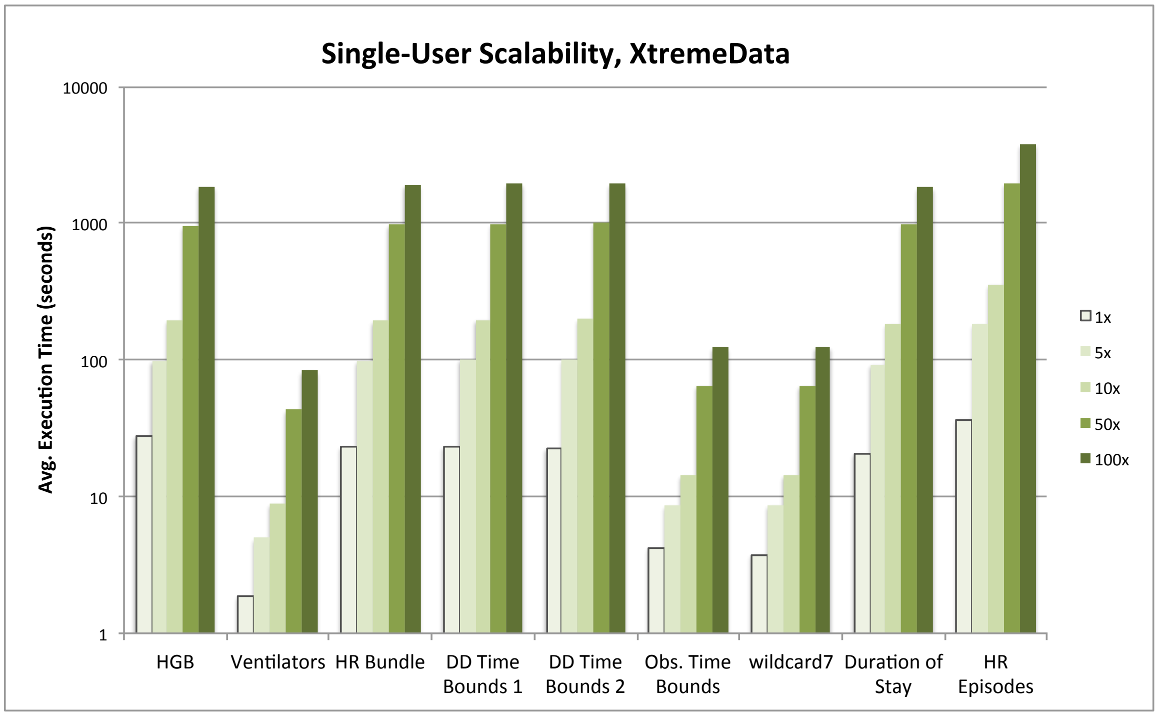


Figure 5.3. Scalability for the XtremeData dbX system. Each query was executed on 5 different data sets. 5 trials were averaged to minimize the effects of system noise.

5.4 Hardware Failover

As with any complex system, we expect that components will fail periodically in a parallel database as part of normal operation. The dbX cluster suffered several spurious disk failures (possibly caused by a faulty disk controller) and two actual disk failures. The Netezza system lost one of its 96 disks. Both systems recovered automatically from these failures with no data loss.

5.5 Things Not Tested

In our testing we excluded several aspects of database performance that are often critical in business environments. We believe wholly that these tasks are important and should be considered but the constraints of our testing environment precluded effective tests.

- *Data Ingestion Speed*: For simplicity and efficiency, we performed all of our data transformation inside the database after loading the initial 49GB of test data. This was a trivial amount to load given that parallel databases typically measure their sustained load performance in terabytes per hour. We chose to focus on query performance, which is highly specific to the users' goals, rather than spend a lot of time measuring this relatively data-agnostic feature.

- *Extract, Transform and Load (ETL)*: ETL jobs typically use external systems for computationally expensive work that cannot be done efficiently within the database itself. Because the Netezza and XtremeData databases lived on an isolated network segment with no access to any other Sandia resources, we were unable to test this.
- *Backup Performance*: As above, there was no backup server available for testing. Also, compared to testing queries of interest, backup performance is a relatively generic and well-studied task.

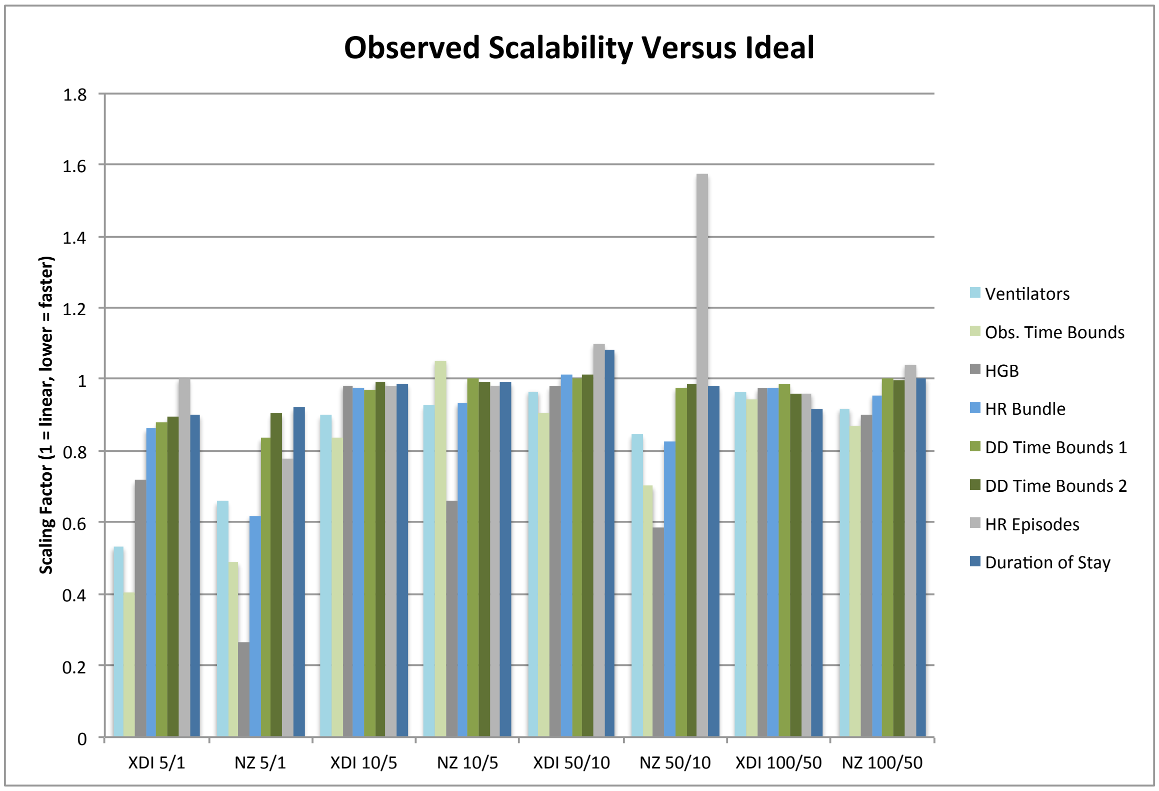


Figure 5.4. This chart shows linear scaling of performance with data size. The column labeled "XtremeData 10/5" indicates the relative slowdown on the XtremeData system on the 10x data set compared to the 5x data. A value of 1 indicates perfectly linear scaling. Values lower than 1 include sub-linear scaling – even faster than linear. We see here that both systems scale linearly with data sizes over nearly all our test cases. The outlier value for the "HR Episodes" query in the NZ 50/10 column is not a measurement error, though we do not know why it happened.

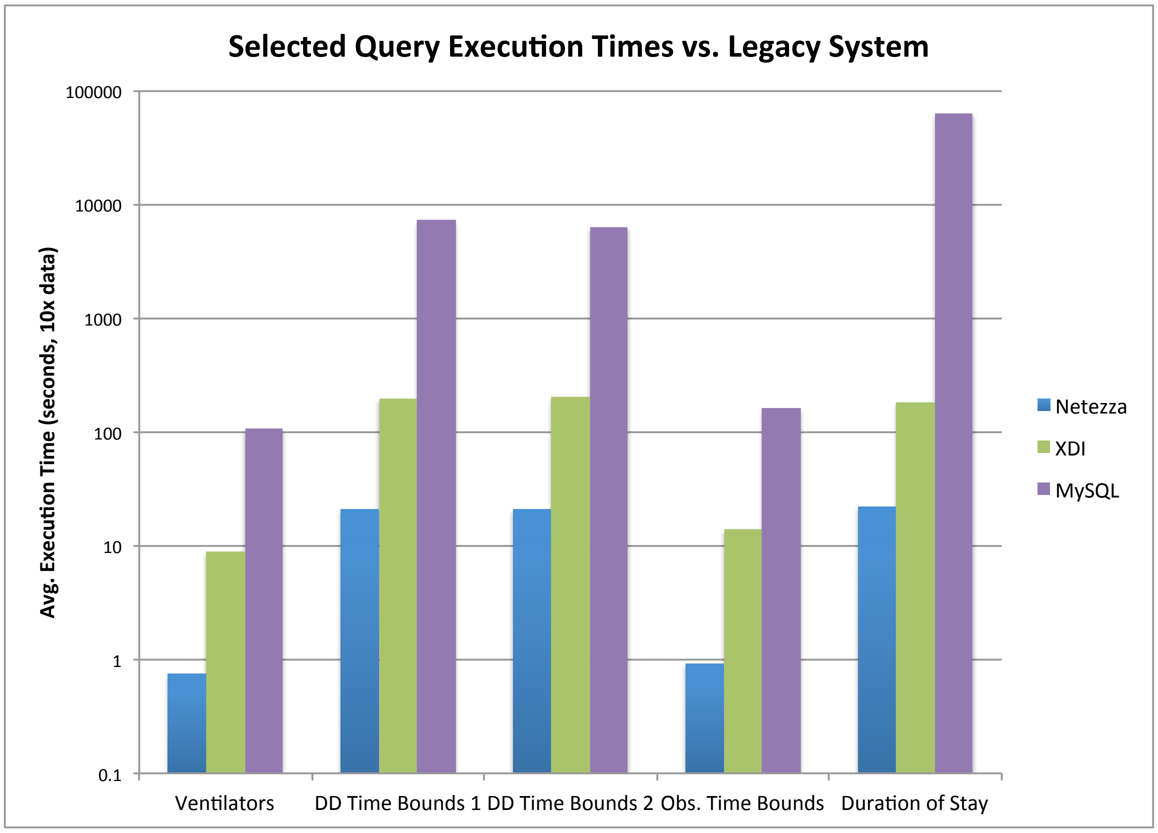


Figure 5.5. We ran the simpler test queries on a well-equipped single-processor MySQL installation for a rough comparison with parallel data warehouses. We see here that both parallel databases are dramatically faster than the single-processor installation. Remember that the vertical axis is logarithmic! The "Duration of Stay" query took over 17 hours on the MySQL system, 3 minutes on the XtremeData system and 22 seconds on the Netezza Twinfin – a speedup of 345x and 2888x respectively.

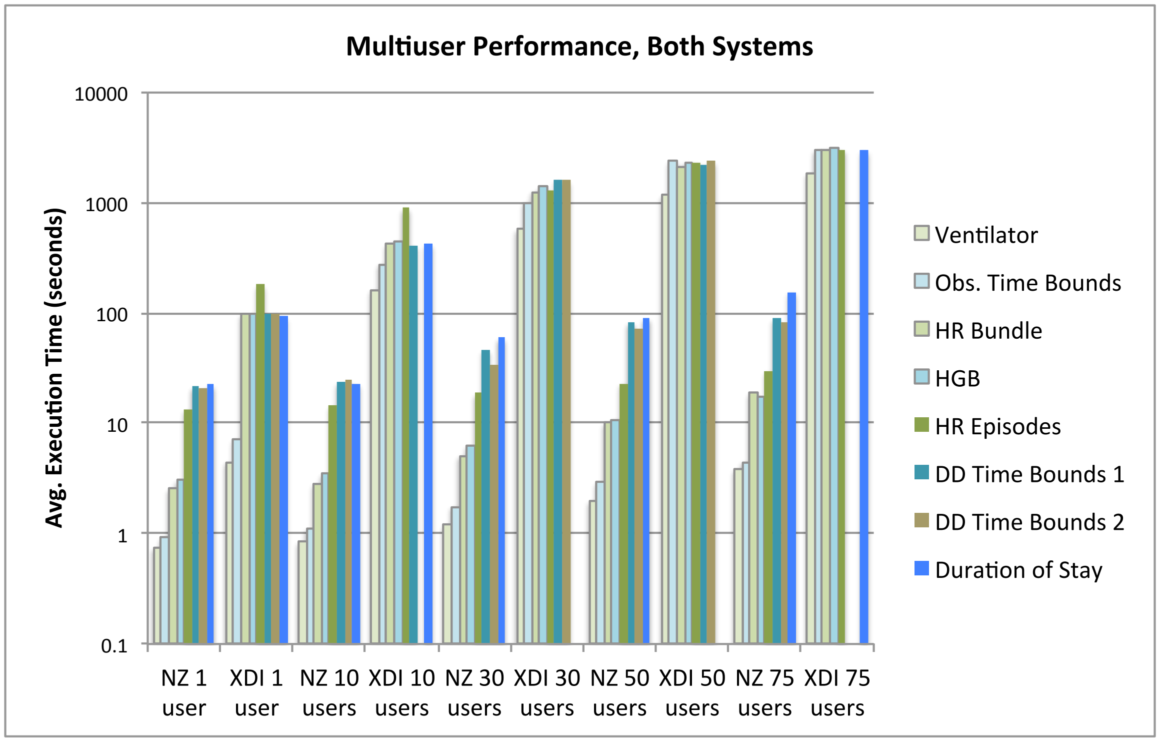


Figure 5.6. Query performance on both systems as the number of concurrent users increases from 1 to 75. Lower bars indicate better performance. Missing bars indicate queries that did not execute during the 2-hour testing period. At 50 users and beyond, most queries on the XtremeData system take about 50 minutes to execute. On the Netezza system those same queries return in about 100 seconds.

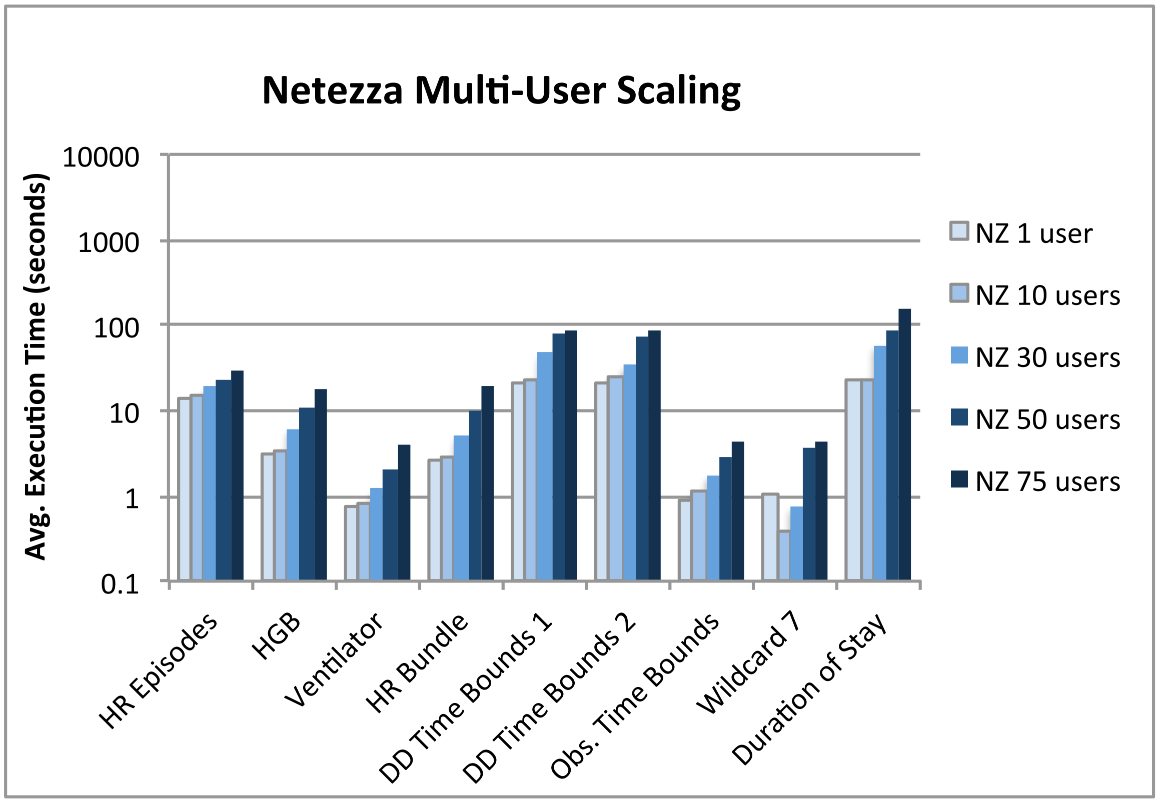


Figure 5.7. Query performance on the Netezza Twinfin as the number of concurrent users increases. Note that performance at 10 users is almost the same as when only one user is running queries. A steeper rise within a group of bars indicates a query that is more affected by higher load. Although the Netezza system software can handle several hundred concurrent connections, performance on this query load degrades sharply at around 100 users.

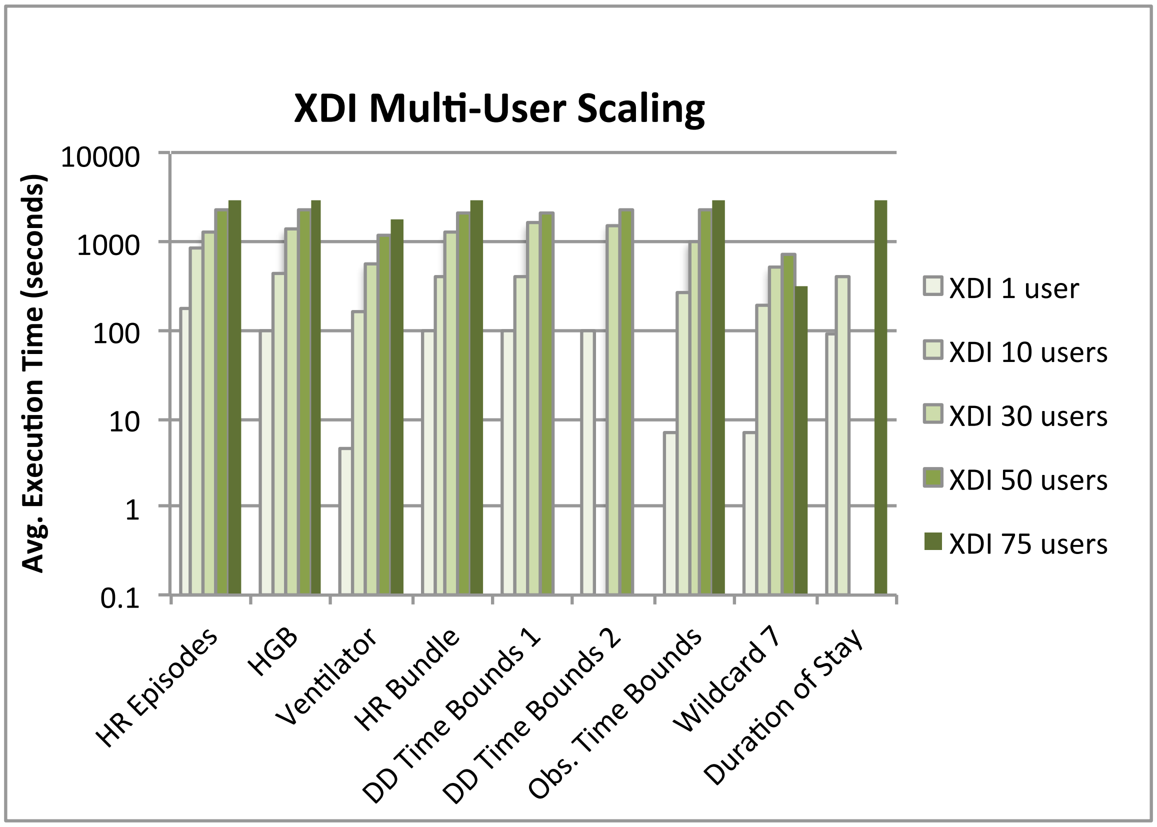


Figure 5.8. Query performance on the XtremeData dbX as the number of concurrent users increases. A steeper rise within a group of bars indicates a query that is more affected by higher load. The dbX system software has a hard limit of 99 concurrent users. The missing entries for `dd_time_bounds1` and `duration_of_stay` indicate queries that did not execute during the 2-hour testing window.

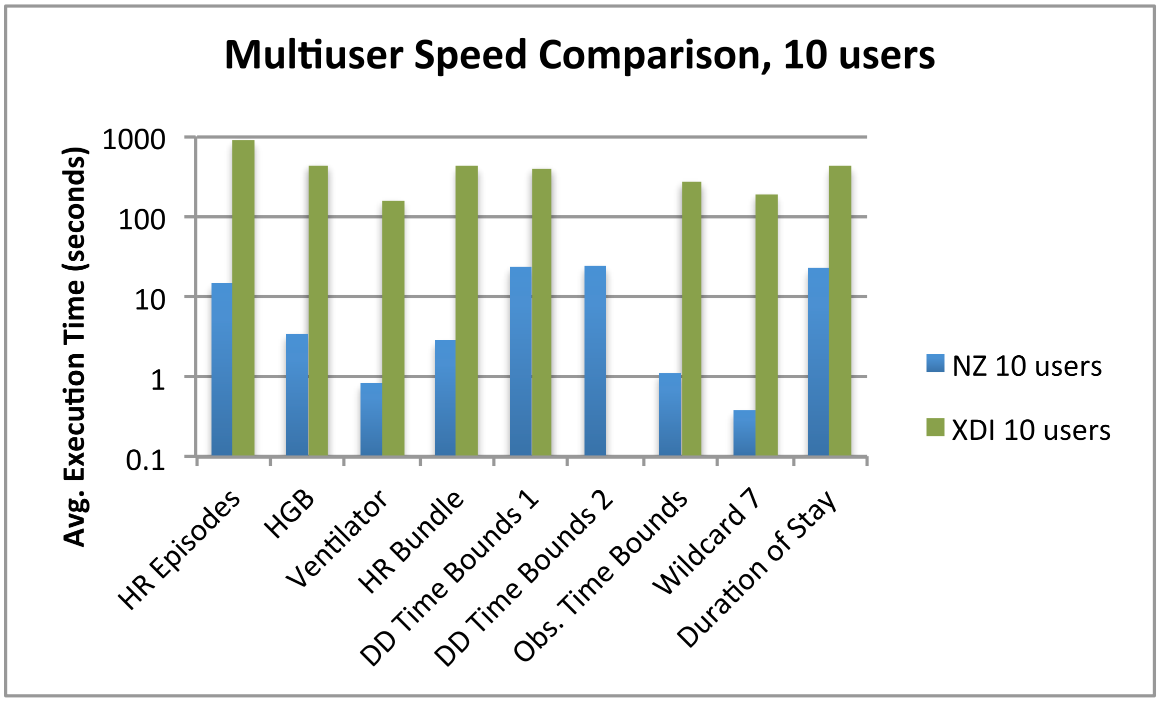


Figure 5.9. Query performance on both systems under a 10-user load. No data is available for the "DD Time Bounds 2" query on the XtremeData system because that query did not execute within the 2-hour testing window.

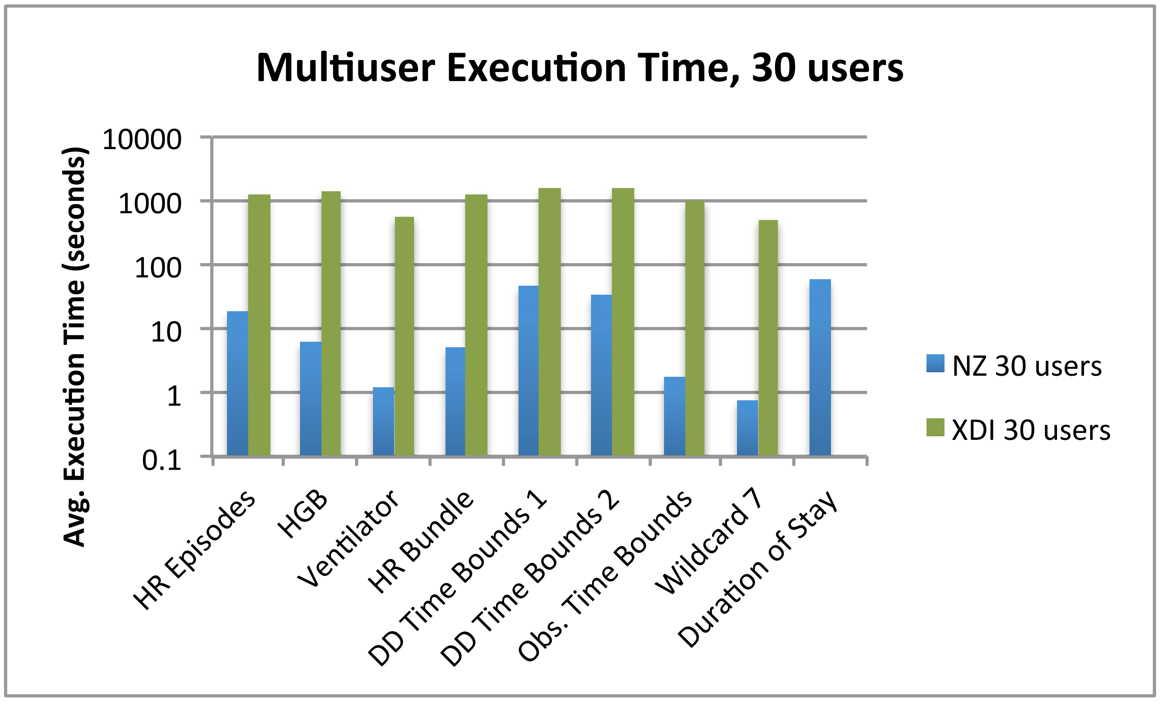


Figure 5.10. Query performance on both systems under a 30-user load. No data is available for the "Duration of Stay" query on the XtremeData system because that query was never executed during the 2-hour testing window.

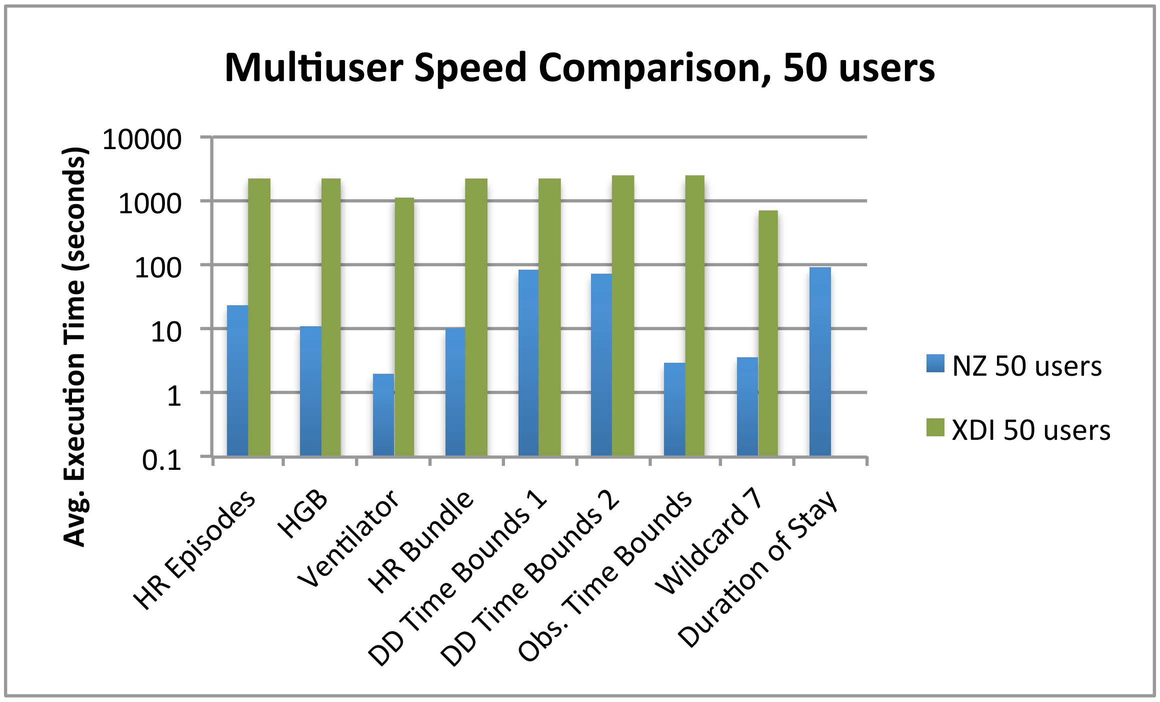


Figure 5.11. Query performance on both systems under a 50-user load. No data is available for the "Duration of Stay" query on the XtremeData system because that query was never executed during the 2-hour testing window.in the random workload.

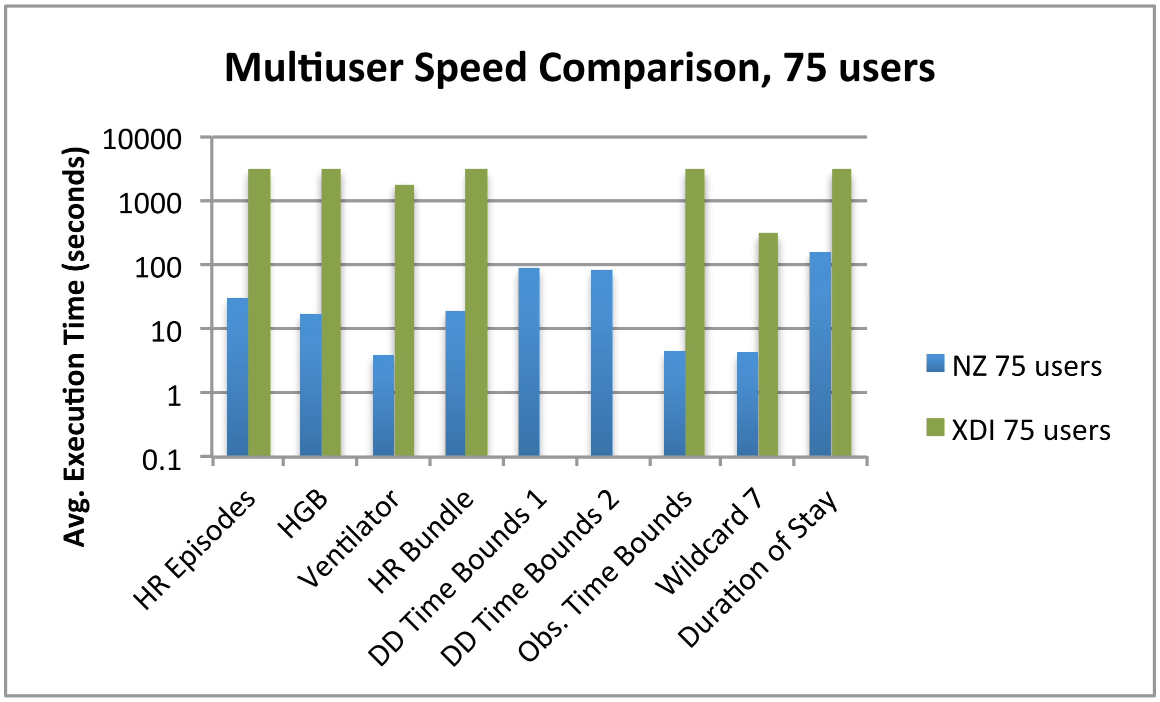


Figure 5.12. Query performance on both systems under a 30-user load. No data is available for the “DD Time Bounds 1” and “DD Time Bounds 2” queries on the XtremeData system because those queries never executed in the random workload.

Chapter 6

Conclusions

We have evaluated the performance of two commercially available parallel multi-processor database systems for their performance on real, de-identified clinical data. The queries used for testing are a mix of simple benchmarks used for sanity checks, slightly more complex reports drawn from a production MySQL installation, and ad-hoc clinical research queries that are not possible in the current production environment.

The hardware systems under evaluation were an IBM/Netezza Twinfin-12 and an 8-node installation of dbX from XtremeData, Inc. Netezza systems are a combined hardware/software product. XtremeData's dbX system is a software-only system.

Our test data was a set of real but de-identified records collected from a single hospital over a period of about 8 years. Since this data was trivially small (49GB) relative to the capacities of the test systems we created larger versions for scalability testing. The largest version would have occupied nearly 5 terabytes on disk before being converted into binary format.

6.1 Single-User Results

Both systems exhibited consistently linear scaling of execution times with respect to the size of the data. In some cases, especially when comparing execution times for the 5x and 1x data sets, scaling was actually faster than linear. We attribute this to the constant overhead of executing any query being amortized over a longer period of processing.

Queries took between 4 and 58 times longer to execute on the XtremeData system than on the Netezza system. The ratio was the smallest on the simplest queries and grew larger with more complex analytics.

6.2 Multi-User Results

Our goal in multi-user testing was to evaluate system usability in the presence of multiple simultaneous connections. We gathered statistics on average query response time with 1, 10, 30, 50 and

75 connections. Queries on the XtremeData system were between 4 and 40 times slower at 10 users than at 1, then between 3 and 12 times slower at 75 users than at 10. Queries on the Netezza system were between 3% and 21% slower at 10 users than at 1, then between 2x and 6.6x slower at 75 users than at 10.

6.3 Final Thoughts

Parallel analytic databases are a disruptive and enabling technology for medical research and evaluation in the presence of ubiquitous patient data. They are *disruptive* in that they obviate the need to carefully select a subset of the data for grooming and tuning prior to a limited, targeted analysis. It is not only possible but in fact *easier and faster* to ask for all the data of interest and let the system sort out the details. Parallel analytic databases are *enabling* because they make possible entirely new classes of analysis. When all the data is quickly accessible as a matter of course we can explore hypotheses and investigate hunches without having to wait days or weeks for the right data set to be made available. This in turn makes it easier to explore the “I wonder if...” questions that so often lead to discovery. The possibilities are fascinating.

References

[1] Andrew Eisenberg, Jim Melton, Krishna G. Kulkarni, Jan-Eike Michels, and Fred Zemke. Sql: 2003 has been published. *SIGMOD Record*, 33(1):119–126, 2004.

[2] IBM Corporation Software Group. The IBM Netezza data warehouse appliance architecture. 2011.

[3] XtremeData Incorporated. Xtremedata — Big Data Analytics. `http://xtremedatainc.com`, January 2012.

[4] US Government Printing Office. Department of Defense 5220.22-M National Industrial Security Program Operating Manual (NISPOM). 2006.

Appendix A

Data Tables

In this appendix we include the raw data used to create the charts in Chapter 5.

Single-User Query Timing, Netezza

	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
1x	0.12	0.18	0.44	1.73	2.57	2.37	2.45	1.76
5x	0.40	0.44	1.36	2.30	10.73	10.72	11.24	6.85
10x	0.75	0.92	2.53	3.03	21.45	21.22	22.27	13.44
50x	3.15	3.22	10.45	8.84	104.63	104.67	109.16	105.74
100x	5.78	5.61	19.98	15.89	209.16	208.74	219.10	220.27

Table A.1. This table shows the average query execution time for each of our test queries on the Netezza Twinfin-12. The leftmost column shows the size of the data set being tested compared to the original 48GB of sample data. All times are in seconds. Lower times indicate faster query execution.

Single-User Query Timing, XtremeData

	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
1x	1.86	4.25	22.94	27.67	23.15	22.49	20.69	36.54
5x	4.96	8.61	98.89	99.19	101.73	100.51	93.20	183.02
10x	8.92	14.39	193.18	195.05	197.62	199.66	183.92	358.23
50x	43.12	65.21	976.04	955.39	990.54	1009.78	993.73	1968.74
100x	83.02	123.38	1906.72	1861.03	1948.57	1934.64	1823.33	3785.46

Table A.2. This table shows the average query execution time for each of our test queries on the XtremeData dbX. The leftmost column shows the size of the data set being tested compared to the original 48GB of sample data. All times are in seconds. Lower times indicate faster query execution.

Per-Query Speed Ratios

	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
1x	7.62	11.95	26.24	7.98	4.50	4.74	4.23	10.41
5x	6.16	9.88	36.47	21.61	4.74	4.69	4.14	13.37
10x	5.98	7.85	38.14	32.19	4.61	4.70	4.13	13.33
50x	6.85	10.13	46.72	54.05	4.73	4.82	4.55	9.31
100x	7.18	11.00	47.71	58.57	4.66	4.63	4.16	8.59

Table A.3. This table shows the speed ratio between the two systems (slower time divided by faster). If both systems exhibit the same scaling behavior for corresponding queries – the desired outcome – then the numbers in each column should remain consistent. If the ratio increases as we go down a column then the Netezza system scales better than the XtremeData system on that particular query. If the ratio decreases, the opposite is true. Note that the lower execution times are more sensitive to random system noise. The 1x row should be considered less accurate than those below it.

Data	System	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
5x/1x	Netezza	0.66	0.49	0.62	0.26	0.90	0.83	0.78	0.92
	XtremeData	0.53	0.41	0.86	0.72	0.89	0.88	0.90	1.00
10x/5x	Netezza	0.93	1.05	0.93	0.66	0.99	1.00	0.99	0.98
	XtremeData	0.90	0.84	0.98	0.98	0.99	0.97	0.99	0.98
50x/10x	Netezza	0.84	0.70	0.82	0.58	0.99	0.98	0.98	1.57
	XtremeData	0.97	0.90	1.01	0.98	1.01	1.00	1.08	1.10
100x/50x	Netezza	0.92	0.87	0.96	0.90	1.00	1.00	1.00	1.04
	XtremeData	0.96	0.95	0.98	0.97	0.96	0.98	0.92	0.96

Table A.4. This table compares the observed scaling in each system to the expected (linear) ideal. Values of 1.0 indicate perfectly linear scaling. Lower values indicate better-than-linear scaling. Higher values indicate worse-than-linear scaling.

	MySQL	XtremeData	Speedup	Netezza	Speedup
Ventilators	107	8.9	12.0	0.75	143
DD Time Bounds 1	7204	198	36.4	21	343
DD Time Bounds 2	6360	200	31.8	21	303
Obs. Time Bounds	161	14	11.5	0.92	175
Duration of Stay	63540	184	345	22	2888

Table A.5. This table shows a comparison between the execution of selected queries on a production installation of MySQL at RGI's facilities and the XtremeData and Netezza systems.

Users	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
1	0.75	0.92	2.53	3.03	21.45	21.23	22.27	13.44
10	0.84	1.11	2.83	3.45	23.54	24.27	22.92	14.63
30	1.20	1.76	5.07	6.24	46.32	33.50	59.26	18.67
50	1.98	2.92	23.17	10.86	82.37	72.09	89.62	23.17
75	3.81	4.46	18.81	17.15	89.11	82.93	155.82	30.03

Table A.6. This table lists the average execution time for each query on the Netezza Twinfin as the number of concurrent users increases from 1 to 75. All times are in seconds. We consider a system usable as long query execution time stays on roughly the same order of magnitude: less than a minute for queries that should take seconds, less than an hour for queries that should take a few minutes.

Users	Ventilator	Obs. Time Bounds	HR Bundle	HGB	DD Time Bounds 1	DD Time Bounds 2	Duration of Stay	HR Episodes
1	4.46	7.20	96.59	97.52	98.81	99.83	91.96	179.12
10	161.23	275.68	420.54	446.37	409.46		420.81	884.50
30	569.38	1004.29	1252.70	1391.90	1613.70	1574.74		1273.30
50	1160.43	2363.07	2110.79	2318.24	2212.01	2355.07		2324.76
75	1825.91	2974.96	3038.14	3065.93			2951.60	3011.89

Table A.7. This table lists the average execution time for each query on the XtremeData dbX as the number of concurrent users increases from 1 to 75. All times are in seconds. We consider a system usable as long query execution time stays on roughly the same order of magnitude: less than a minute for queries that should take seconds, less than an hour for queries that should take a few minutes. Empty entries in this table indicate queries that did not have time to execute during the 2-hour testing period.

DISTRIBUTION:

1	RGI Informatics, LLC P.O. Box 51 Cornwall, NY 12418 U.S.A. (Electronic copy as well)	
1	MS 0899	RIM - Reports Management, 9532 (electronic copy),
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories