Performance Testing and Analysis of Redundant Arrays of Inexpensive Databases

Brandon Rogers, Joseph J. Ekstrom

Brigham Young University

Abstract

The Internet revolution has focused scaling and redundancy research on systems, such as application servers using clusters, redundant Internet connections, and other redundant hardware. Conversely, back-end services, such as databases, have largely remained undistributed, un-clustered, and housed in large, dedicated SMP machines. However, a new turn in database management recently was introduced with the proposal of a RAIDb (Redundant Array of Inexpensive Databases) standard.

Various studies have indicated that redundant disk arrays and computer node clustering have greatly improved machine performance per dollar spent. Clustering provides immense processing capability and redundancy has proved to be a valuable resource in up time and availability. Based on these notions, RAIDb utilizes increased processing power through database computer clustering, while providing increased availability through the use of redundant databases and database controllers.

This paper details the performance of the C-JDBC (Clustered JDBC), a Java-based implementation of RAIDb. Performance is measured using a variable-node RAIDb cluster against a standard database backend. Each database backend is flooded with SQL requests by a benchmark client, which keeps track of the number of requests per minute successfully served by the database engine. Results of the testing are compiled and interpreted, showing performance trends and comparisons of the database implementations.

Introduction

In 2003, Brigham Young University's School of Technology began building a laboratory for hardware and software testing and performance analysis. The lab contains 20 workstation computers, a few high-speed machines and switches, and one Itanium 64-bit computer. The purpose of this lab is to provide students and faculty with a means to perform research that can be used to characterize the performance of a system. This experimental environment is ideal for creating and performing benchmarking tests to scientifically describe the performance of these systems. This is one of two studies completed and used to christen this new lab.

Redundant Database Arrays

E-commerce and data collection services offer great value to many businesses and consumers by saving expensive time and adding the ability to skip expensive middleman-retailers. Due to this added value, important electronic systems and services push the development of high-speed super-computing, never-fail electronic storage, and high-bandwidth communications medium. Traditionally, there has been much research on scaling front tier systems (web and application servers) by utilizing clusters, redundant disk arrays, and redundant Internet lines. In spite of these incredible advances in the processing world, back-end databases have largely remained un-clustered, and housed in large, dedicated SMP (Symmetric Multi-Processor) machines. However, a turn in the tide of database management recently was introduced with the proposal of a RAIDb (Redundant Array of Inexpensive Databases) standard.

Various studies have indicated that redundant disk arrays and computer node clustering have greatly improved machine performance per spent dollar[2,3]. Clustering provides immense processing capability (the sum of the capabilities of the processors involved) and redundancy has proved to be a valuable resource in up time and availability[4]. Based on these notions, RAIDb utilizes increased processing power through database computer clustering, while providing increased availability through the use of redundant databases and database controllers[1]. As defined in September 2003 at INRIA (Institut National De Recherche En Informatique Et En Automatique) by Emmanuel Cecchet, Julie Marguerite, Willy Zwaenepoel, the RAIDb standard requires a middleware application that is independent of a specific database backend. The software solution mimics levels of performance classified by the RAID (Redundant Array of Inexpensive / Independent Disks). RAIDb is divided into three classifications, or levels: RAIDb-0, RAIDb-1, RAIDb-2[1].

RAIDb-0: full partitioning

RAIDb, level 0 (RAIDb-0), is best described as database striping, or distributing the tables in the database among backend nodes (Figure 1). RAIDb-0 is similar to common distributed database systems, such as Oracle RAC[5], PostGreSQL Replication Project, and Emic Application Clusters for MySQL[6]. Like these systems, stored data in a RAIDb-0 system is simply distributed among nodes. No replication or duplication of information is performed. Crecchet indicates, "[that] like for RAID [Redundant Array of Independent Disks] systems, the Mean Time Between Failures (MTBF) of the array is equal to the MTBF of an individual database backend, divided by the number of back-ends in the array.[1]" Simply put, the average time between failures is divided by the number of available back-ends.

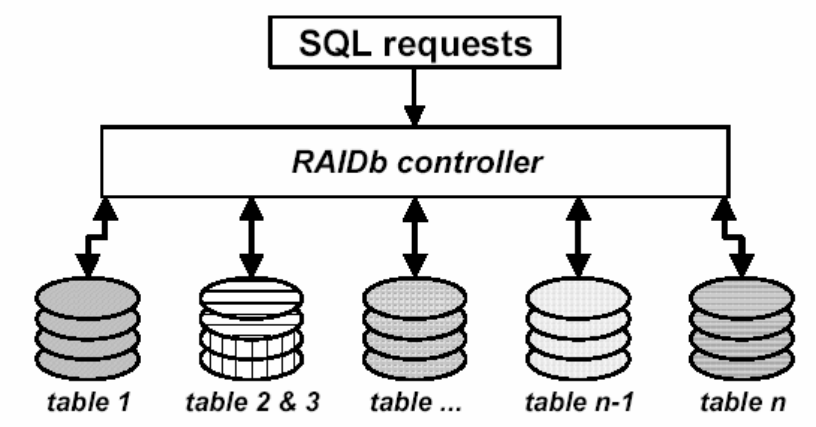

Figure 1 *– RAIDb-0 table striping.*

RAIDb-1: full replication

RAIDb, level 1 (RAIDb-1), performs data replication through database mirroring, or complete duplication of databases (Figure 2). RAIDb-1 requires that each backend node have the ability to handle the entire database. Using RAIDb-1, load balancing is also available as requests can be distributed over backend nodes.

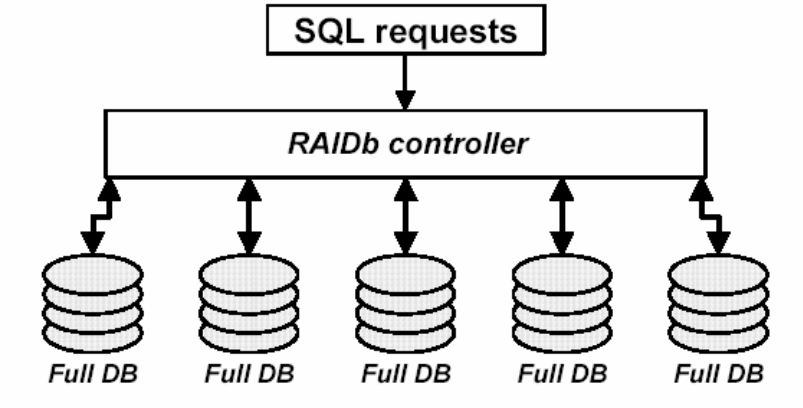

Figure 2 – RAIDb-1 – Full Database Replication.

RAIDb-2: striped redundancy

RAIDb, level 2 (RAIDb-2), features a combination of RAIDb-0 and RAIDb-1, or partial replication (Figure 3). Tables are distributed, like RAIDb-0, allowing for databases that are larger than the storage (and performance) capacity of a single node. However, using RAIDb-2, tables are also replicated, providing fault tolerance and load balanced clustering. The RAIDb-2 solution allows for continuous operation despite failed nodes.

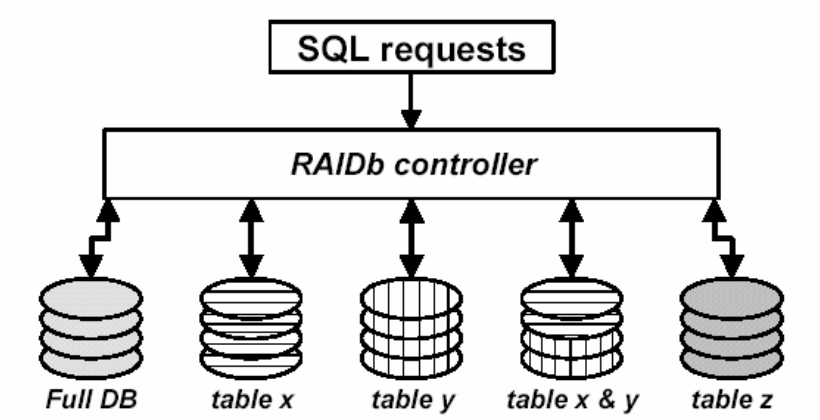

Figure 3 – RAIDb-2 – Mixed replication and data striping.

Additional RAIDb Levels

Two additional RAIDb levels are also detailed in the RAIDb standard, including error-checking levels for RAIDb levels 1 and 2. Error-checking RAIDb-1 (RAIDb-1ec) aims to discover and handle Byzantine failures[7] that can occur in a "highly stressed cluster of PCs[1]." Error-checked RAIDb-2 (RAIDb-2ec) aims to provide the same error checking as RAIDb-1ec to RAIDb-2 clusters.

Should RAIDb truly provide increased speed and reliability to the database realm as reported, low cost database arrays will provide industry with a viable option to high-cost database machines and clusters.

Experiment Design

C-JDBC (Clustered Java Database Connectivity) is a Java-based middle-ware implementation of RAIDb. JDBC (Java Database Connectivity) drivers are used to connect and communicate with database servers. All database operations are supported, including stored procedures, SQL (Structured Query Language), and security rules. C-JDBC is both vertically and horizontally scaleable, compliant to the RAIDb standard. C-JDBC, version 0.15, was used in all testing procedures and was downloaded from http://c-jdbc.objectweb.org/.

Hardware Configuration

The test system was configured as shown (Figure 4). The testing client has a Intel 64-bit Itanium Processor with 1 gigabyte of RAM. All database backend nodes are Pentium 4 1-GHz machines with 512 MB RAM. Backend nodes are interconnected using a Cisco Catalyst 3500XL 10/100 Mbps switch. The RAIDb Controller is also a Pentium 3 1-GHz machine, but is equipped with two 100 Mbps network cards, which are bridged, to isolate the traffic of the backend nodes and to eliminate network congestion for all SQL queries and responses. The Red Hat 9 Linux Operating System (kernel 2.4.20-8) is installed on all computer nodes. Java version 1.4.3 was used to run the RAIDb controller and testing client. All backend nodes run the MySQL database version 4.0, downloaded from http://www.mysql.org.

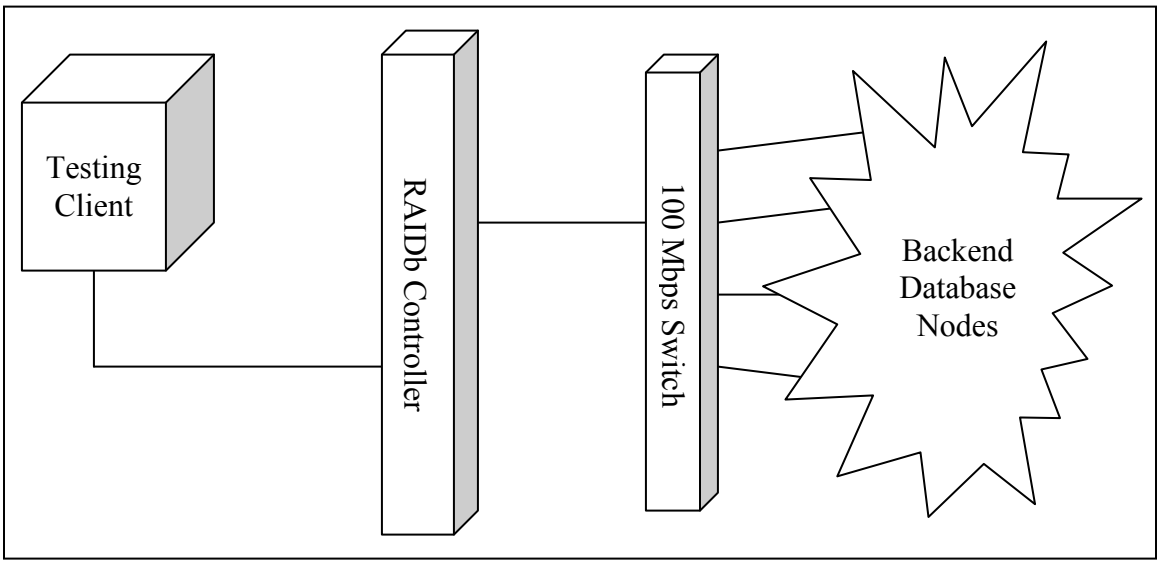

Figure 4 *– Test Environment Configuration.*

Note: The test environment was designed to test the relative change in database performance rather than guided by an attempt to achieve the highest throughput or transfer rates.

Database Schema

The TPC (Transaction Processing Performance Council) web benchmark (TPC-W) was used as a foundation for the database schema as shown (Figure 5). Performance testing does not use the TPC-W benchmark, however. This allows the tests to have a representative database schema while testing the raw throughout of the database engine.

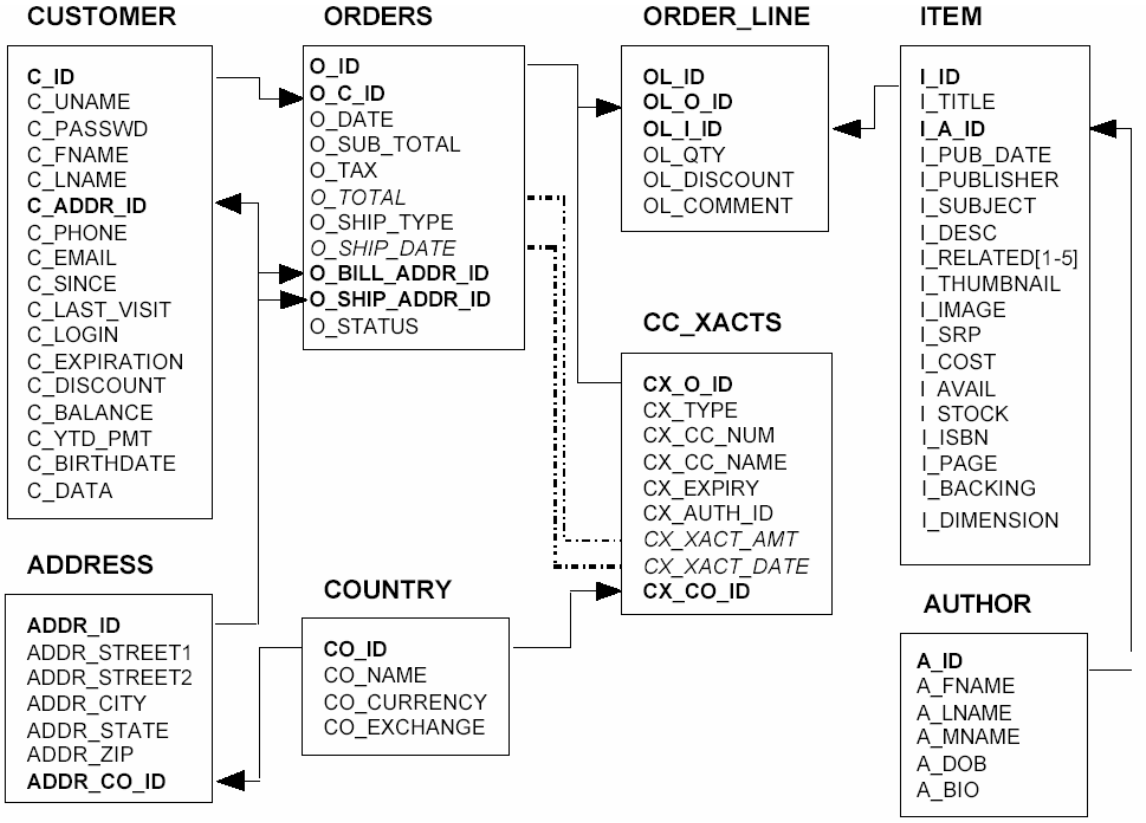

Figure 5 *– TPC-W Database Schema*

For RAIDb level 1 and 2, database tables were evenly distributed throughout the cluster. In RAIDb-1 organizations, the entire database was replicated on each node. For odd-numbered node RAIDb-2 systems, one node contained the entire database, while the remaining nodes provided equally partitioned redundancy. For even-numbered node RAIDb-2 systems, all tables were equally distributed and replicated among the available nodes.

Testing Client

The Java–based testing client was created to gather data about database performance by playing a series of "recorded" SQL statements. By using the same statements, and the same series of statements, change in performance reflects actual change in throughput and disassociates any change in performance with varying database queries. All SQL statements were created using TPC benchmarking tools, which were randomly assorted and arranged.

Testing Results

Twenty runs, which lasted 30 minutes each, stressed the RAIDb controller. The results of each of the runs were averaged to provide the results found below. Most RAIDb level tests showed high standard deviations among the results in the 20 runs. This is largely due to the type of query set used in the specific minute of testing. Query sets with a large proportion of database writes decreases average throughput due to the needs to broadcast and verify the write or update. Query sets with a heavy proportion of reads yielded high throughput, causing large deviations. This result is consistent with other performance analysis tests performed on C-JDBC[1].

RAIDb testing resulted in elevated throughput in the higher RAIDb levels. While no significant improvement was made using RAIDb-0, large improvement in throughput was achieved using RAIDb-1 and RAIDb-2 (Table 1). Throughput increased as nodes were added to the database cluster (Figure 6).

	1	2	3	4	5	6
Single	557.3333					
RAIDb-0		596.6667				
RAIDb-1		987.3333	1495.333	1762.667	2085.333	2308.667
RAIDb-2			1756.667	2357.333	2880.667	3344

Figure 6 *– Throughput results for RAIDb levels with 1- to 6- node clusters.*

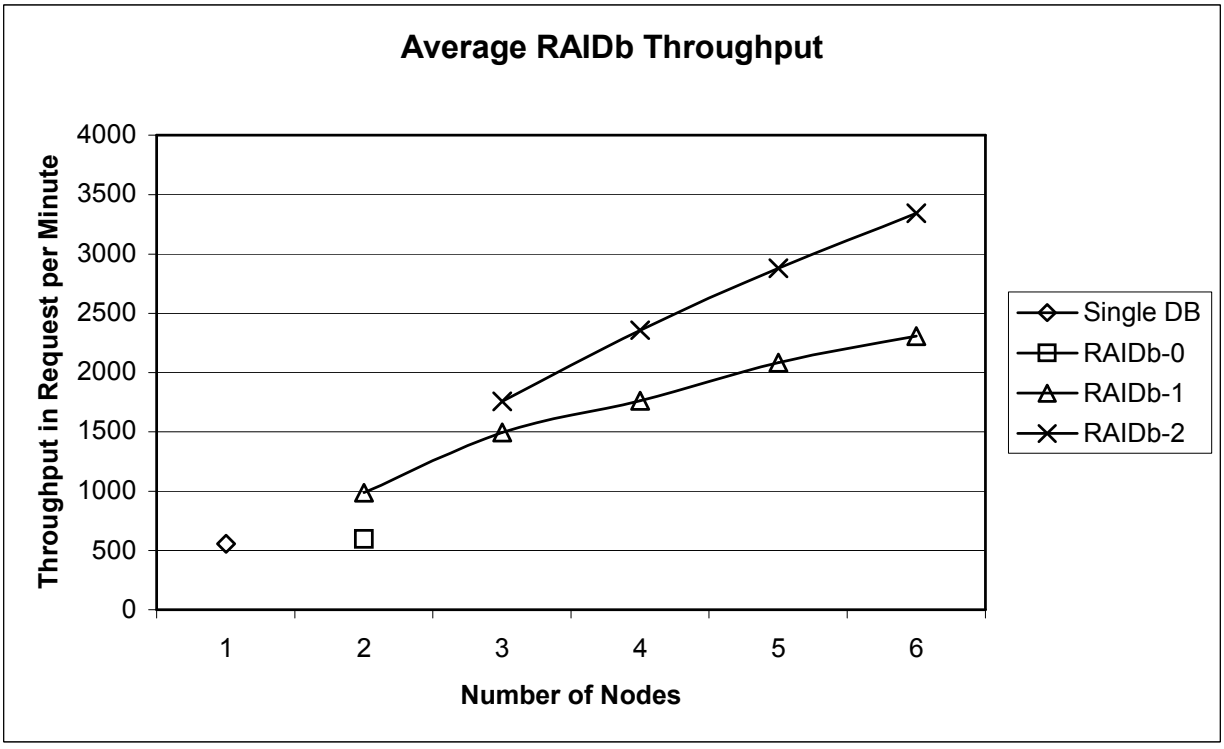

Table 1 *– Throughput results for RAIDb levels with 1- to 6-node clusters*

Standard database

The single, standard database allowed an average of 557.3333 requests per minute, or 4.64445 requests per second.

RAIDb-0

RAIDb-0 showed no significant improvement over the standard database with an average throughput of 596.6667 requests per minute, or 9.9444 requests per second. Although no significant increase in throughput was recorded, CPU usage was consistently lower. The improvement ratio for this RAIDb level averages 1.0706.

RAIDb-1

RAIDb-1 shows increased performance in backend database clustering consisting of no more than 2 nodes. RAIDb-1 achieved an average throughput of 987.3333 requests per minute (16.4556 requests per second) in a 2-node system and as much as 2,308.667 requests per minute (38.4778 requests per second) in a 6-node system. The throughput improvement ratio for RAIDb-1 ranges from 1.7715 to 4.1423.

RAIDb-2

Results for RAIDb-2 indicate large throughput gains, ranging from an average of 878.3333 (14.6389 requests per second) requests per minute to 1672 requests per minute (27.8667 requests per second) in 2- to 6-node database systems, respectively. Ratio for throughput improvement ranged from 3.1519 in a 2-node cluster to less than 6.0000 in a 6-node cluster.

Conclusions and Summary

While RAIDb-0 provided little more than table distribution, higher levels of RAIDb showed significant improvement in throughput performance. RAIDb-1 systems provided increased throughput in systems needing extensive redundancy. Possible uses of RAIDb-1 would include safety and natural backups to data mining and e-commerce applications. RAIDb-2 performed well in all situations, increasing throughput efficiency by more than a factor of six.

Recommendations for Further Research

The high deviations exhibited in the RAIDb performance testing leads to more detailed experimentation and performance testing of the system. Performance testing continues on RAIDb clusters to characterize more specific performance qualifications. Future experimentation should include performance characterization of the RAIDb Controller, each backend node, and more detailed tests on the system as a whole, including database portability, recoverability, and throughput. Testing to understand any anomaly in the system should also be tested.

Bibliography

1. Cecchet, E., Marguerite, J., Zwaenepoel, W. *RAIDb: Redundant Array of Inexpensive Databases,* (September 2003), INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 4921.

2. Gibson, Garth A., Redundant disk arrays : reliable, parallel secondary storage, ACM distinguished dissertations, Cambridge, Mass. : MIT Press, 1992.

3. Nielson, Curtis R., A Descriptive Performance Model of Small, Low Cost, Diskless Beowulf Clusters, Electronic Thesis and Dissertations, Provo, UT : BYU Press, 84604.

4. Sherman, L., Choosing the Right Availability Solution: High-availability Clusters and Fault Tolerant Systems, http://whitepapers.informationweek.com/, Accessed November 22, 2003.

5. Smith, G., Pruscino, A., Oracle RAC 10g Overview, http://otn.oracle.com/products/database/clustering/pdf/TWP_RAC_Overview_10gR1_112503.pdf, Oracle Corporation, 2003.

6. Emic Application Cluster 2.0 for MySQL: High Availability Cluster with Dynamic Load Balancing, http://www.emicnetworks.com/download/pdf/eac_mysql_whitepaper.pdf, November 2003, Emic Networks, Accessed November 22, 2003.

7. Henziger, M., Google: Indexing the Web - A challenge for Supercomputers, (September 2002), Proceeding of the IEEE International Conference on Cluster Computing.

BRANDON ROGERS

Brandon Rogers is a graduate student at Brigham Young University. Brandon is currently pursuing a Master's Degree in Information Technology. Mr. Rogers has worked extensively with custom database applications and has worked for success of large database systems in international networks.

JOSEPH EKSTROM

Joseph J. Ekstrom (Ph. D. Computer Science, BYU 1992) has been Associate Professor of Information Technology at BYU since 2001. During 30 years of industrial experience he held positions from developer through senior management. His research interests include network and systems management, distributed computing, system modeling and architecture, system development, and IT curriculum and instruction.