MySQL Network and the Dell PowerEdge 2800: Capacity Sizing and Performance Tuning Guide for Transactional Applications

Enterprise Product Group (EPG)

Dell White Paper

By Todd Muirhead, Dave Jaffe and Nicolas Pujol

Dell | Enterprise Systems

April 2005

Contents

Executive Summary 3

Introduction 4

Dell PowerEdge 2800 4

MySQL Network 4

Online Store Workload 5

Testing and Sizing Guide 5

Performance Results Summary 5

Hardware and Software Installation 6

MySQL Installation and Configuration 8

DVD Store Implementation 10

The Database Schema 10

InnoDB Tables, MyISAM Tables and Indexing 11

The PHP Application 11

The Online Transaction Processing Driver Application 13

Results 15

Conclusions 17

Acknowledgements 18

Appendix A. mysqlds2 Build Script 19

Appendix B. mysqlds2 Index Creation Script 22

Section 1

Executive Summary

The MySQL® open source database has gained attention over the last few years as a low cost database option. The feature set of MySQL has increased during this time to include many of the things that enterprise class applications require including transactional integrity, foreign keys, and support for large database sizes. Utilizing the powerful Dell PowerEdge 2800, a two-processor tower server with ample internal disk capacity, to run MySQL provides a high value solution for customers.

In order to simulate how a small or medium sized business might utilize such a solution, a PowerEdge 2800 configured with only internal disks was used to run an online DVD store test database application. The results show that a server with software configured at a price of less than $9000[1] is able to handle over 1900 orders per minute, a transaction rate much higher than what many small or medium sized businesses require.

Section 2

Introduction

Usage of open source software for business critical tasks is growing rapidly within IT organizations. With the standardization of enterprise computing, formerly complex and expensive software can be acquired and commercially supported at more affordable levels than before. With over 6 million active installations, MySQL AB offers an enterprise database with comprehensive services that delivers a proven alternative to customers looking for speed and ease-of-use without compromising quality.

In this paper MySQL's ability to scale and perform on a transactional workload, referred to as Online Transaction Processing (OLTP), is demonstrated. The focus of this paper is to provide a sizing guide for MySQL running on Dell PowerEdge servers, specifically to small, medium and growing businesses looking for a high performance and reliable database solution for transactional applications.

Dell PowerEdge 2800

Dell offers a wide range of servers designed to best meet the needs of customers. The goal in selecting the hardware was to combine database performance, reliability, and affordability. The PowerEdge 2800 is an ideal platform for this purpose. The PowerEdge 2800 can be configured with as many as 10 internal disks and up to 2 processors. The combination of these features allows for a significant number of database transactions to be processed with a single server without external storage. Additionally the PowerEdge 2800 has built-in fault tolerance such as hot pluggable and redundant power supply, cooling fans, and a RAID disk controller integrated on the motherboard. For the PowerEdge 2800 configurations tested in this paper prices ranged from $6,889 to $8,782, including software costs, keeping the total server cost low.[1]

SuSE Linux Enterprise Server 9 is tested and supported on the Dell PowerEdge 2800. It is based on the Linux 2.6 kernel and is available with the Dell PowerEdge 2800 from $175 for a one processor one year subscription to $747 for a two processor three year subscription.

MySQL Network

The MySQL database is available in a community edition and a certified binaries edition. The community edition is designed with bleeding edge features, and is available on www.mysql.com and its mirror sites. The MySQL certified binaries edition has optimized software that has passed rigorous quality assurance (QA) testing. This is the software Dell and MySQL recommend for production use, and is available through MySQL Network.

MySQL Network delivers products and services such as MySQL certified and optimized binaries, a knowledge base to gain expertise, technical support, advisors, and at higher service levels other optional services such as performance tuning and intellectual property (IP) indemnifications.

MySQL Network is available from MySQL and starts at $595 per physical server.[2] This low price point is especially attractive since there is not a licensing imposed per CPU, core, memory or user limitation, giving customers the freedom to choose the right hardware for their needs.

Online Store Workload

For this paper, MySQL's capabilities for transactional applications are demonstrated. Many IT organizations need to implement online stores that process orders and want to know how far they can scale with MySQL. Dell designed an online DVD store, where customers log in to the site, browse, and then place orders. More details are provided in section 5 on the application.

The test ran both the web server and database on one single PowerEdge 2800. Customers interested in optimizing performance further could run the application and database tiers on separate servers.

Testing and Sizing Guide

Many database benchmarks in the industry stripe data over a high number of disks located in many external disk enclosures. In this test the server used only 10 internal disks, all data being mirrored and fault tolerant to potential disk failures, to reflect what many small and medium businesses use for disk configuration. Best practices for tuning MySQL were implemented and all parameter changes are fully documented in this paper. A database administrator new to MySQL should be able to reach a high level of performance in a short time by applying the same approach used in this paper.

Performance Results Summary

Taking moderate and cost effective hardware configurations, the largest configuration tested included 2 CPUs, 4GB of memory, and a total of 8 disks, all mirrored for redundancy. The top throughput achieved was 1,967 orders per minute. Assuming each order has an average price of $25, the PowerEdge 2800 running MySQL can generate an impressive annual revenue rate of $25.8B.

Because orders do not typically come in a linear fashion over a year, it is recommended to account for spikes in demand (e.g., Christmas shopping, Valentine's Day and other seasonal events). With a conservative 10% capacity utilization rate on average, the tested configuration and DVD store application is still able to handle over $2.5B in online sales per year. At this rate, the server is able to handle to a spike in demand that is up to 10x the average level. Each IT organization can take what they believe is the right percentage and scaling factor to plan for spikes.

This demonstrates the great applicability of Dell and MySQL for growing businesses looking for a tested, reliable and cost effective e-commerce solution, or transactional application.

Section 3

Hardware and Software Installation

The Dell PowerEdge 2800 server was used for testing to take advantage of its dual processing capability and its tower form factor's extra space for disk drives. In order to show a range of configurations the number of processors and the amount of memory was varied for each test run. The server was configured with one or two 3.0 GHz Intel Xeon processors with 1MB L2 cache and either 1, 2, or 4 GB of memory for the testing. Eight 146GB U320 SCSI disks were installed as internal storage and attached to the integrated PowerEdge Raid Controller 4ei. Eight is the maximum number of drives that can be installed in the PowerEdge 2800 without converting the media bay. If the media bay is converted to support disk drives, then two more disks could be installed.

The eight disks were split into two RAID sets: a two disk RAID-1 for the operating system and a six disk RAID 1+0 for the data and log files. In systems where there are more disks it is ideal to split logs onto separate physical disks from the data, but when limited to only eight it is fine to put them on the same physical disks. RAID 1+0 is more efficient for handling random writes than any other RAID type, so it gives the best performance for database. The downside to RAID 1+0 is that only half of the raw disk storage is usable. In environments where usable space is more desirable than performance, RAID 5 can be used.

Additional features of the PowerEdge 2800 were also used during testing. One of the two onboard Gigabit Ethernet adapters was used for network connectivity. Two hot-swappable power supplies were used along with redundant hot-swappable fans to make the system highly available. The optional Dell Remote Assistant Card (DRAC) 4 was also included on this server to take advantage of its web based remote control features of console redirection, power on and off, and virtual cd-rom and virtual floppy drive support.

SuSE Linux Enterprise Server 9 is tested and certified for the Dell PowerEdge 2800 and can be selected as the Operating System for the 2800 at purchase. SuSE Linux Enterprise Sever is the first commercial Linux distribution based on the Linux 2.6 kernel. SuSE Linux Enterprise Server 9 was installed on the RAID 1 mirror disk set. A basic install with the addition of Apache2, PHP4, Samba, Telnet, and FTP was completed. Apache2 and PHP4 were installed to support the test DVD store web application. Samba, Telnet, and FTP were added to support easy remote access to the system.

Apache2 is the web server that was used to provide the web application for the DVD store. Apache2 allows the PHP4 module to be loaded and execute the PHP4 web pages when accessed by a remote user. A small number of parameters were changed in the /etc/apache2/server-tuning.conf file to ensure that the webserver was not the limiting factor. The parameters ServerLimit and

MaxClients were set to 250 to have enough processes available to support all incoming requests. MaxKeepAliveRequests was set to 0 to enable an unlimited number of requests during a persistent connection. The default location for html files, and other files to be available via http from the webserver, was /srv/www/htdocs. All of the DVD store's .php4 files were located in that location.

PHP configuration required only two changes to the default /etc/php.ini file. The mysql.default_host parameter was set to the IP address of the PowerEdge 2800 because both the webserver and the database were on the same PowerEdge 2800. The mysql.default_user was set to web.

Section 4

MySQL Installation and Configuration

MySQL 4.1.9 database and client libraries were downloaded and installed on the PowerEdge 2800 server. The RPM version was downloaded so the following commands were used to install:

```
rpm -ivh MySQL-server-4.1.9-0.i386.rpm
rpm -ivh MySQL-client-4.1.9-0.i386.rpm
```

The database root user must be initialized with a password following installation by using the following command:

```
mysqladmin -u root password <password>
```

Startup the mysql database the following is used:

```
/etc/init.d/mysql start
```

Additionally the web user must be configured to have access to the DS2 database that will have all of the DVD store tables. The web user is the user that the PHP application will use to access the database. The following commands were used to first logon to mysql and then grant privileges for the web user:

```
mysql -u root –p
mysql> grant all privileges on DS2.* to web@localhost identified by '<password>’;
mysql> grant all privileges on DS2.* to web@'%' identified by '<password>';
```

The MySQL installation includes sample configuration files in the /usr/share/mysql directory for a variety of configurations. To use one of these sample files it needs to be copied to the /etc directory and renamed to my.cnf. For this testing, the my-large.cnf file was used as a starting point. It is highly recommended to use the configuration file for all parameter changes in order to maintain control over the settings of the database server. All parameters described in this paper were implemented by entries in the /etc/my.cnf file.

During initial testing a query log was enabled so that every query issued to the database was written to a file. Once development and testing of the application was completed the parameter was removed from the configuration for performance reasons. The parameter to enable the query log includes the path and filename as in the following example:

```
Log=/var/lib/mysql/mysql_query.log
```

A major factor in database performance is making sure that the database has enough memory to reduce the amount of time the database spends accessing the disk drives. The DVD store uses both MyISAM and InnoDB tables which have different parameters for controlling the amount of memory that each has available for it use. Table 1 shows the parameters that were used and the values that were assigned.

Parameter	800 MB	1.7 GB	2.4 GB
Table_cache	64M	64M	64M
Key_buffer	128M	256M	256M
Sort_buffer_size	2M	2M	2M
Read_buffer_size	2M	2M	2M
Read_rnd_buffer_size	4M	4M	4M
Myisam_sort_buffer_size	64M	64M	64M
Query_cache_size	128M	128M	128M
InnoDB_buffer_pool_size	384M	1024M	1700M
InnoDB_additional_mem_pool_size	20M	20M	20M
InnoDB_log_buffer_size	8M	8M	8M

Table 1: MySQL memory parameters used during testing for the three configurations.

In order to determine the values used for these parameters there are several things that need to be looked at. Once logged into mysql, the command "show InnoDB status" will include stats on how much of the InnoDB buffer pool memory is in use. The key_buffer size is directly related to the size of the MyISAM tables. In the case of the DVD store application the products table is the only MyISAM table used and its size is about 120 MB.

The fulltext search capability of the MyISAM table is why it was used for the PRODUCTS table. In order for the fulltext search to work, an index of type fulltext must be created against the table columns that are to be searched. The default behavior of the full text search assumes that common words and words less than 4 characters are not important and so are not indexed and cannot be searched on. In the DVD store we are searching by title and actor. In the case of the DVD store titles, all words are important regardless of length or if they are common words. MySQL has list of "STOP WORDS" that are considered common and are ignored. In order to disable the stop words and decrease the minimum length of searchable words the following two lines were added to the configuration file:

```
ft_min_word_len = 3

ft_stopword_file =
```

These settings affect the creation of the fulltext index and must be in effect when the full text index is created. If a fulltext index was created before these parameters were set, then it must be dropped and re-created in order for the new settings to work.

Section 5

DVD Store Implementation

To test MySQL database capability and performance in an Online Transaction Processing (OLTP) environment, a large database (100 GB total size), representing an online DVD store with 1 million DVD titles, 200 million customers and 120 million orders was built. Advanced database features such as transactions and referential integrity constraints were employed. In addition, functionality typical of some online stores was implemented, including reporting to the user previous purchases and recommendation of titles enjoyed by others.

The Database Schema

The MySQL DVD store database was comprised of six main tables and one other small table (see Table 2).

Table	Columns	Number of Rows
CUSTOMERS	CUSTOMERID, FIRSTNAME, LASTNAME, ADDRESS1, ADDRESS2, CITY, STATE, ZIP, COUNTRY, REGION, EMAIL, PHONE, CREDITCARDTYPE, CREDITCARD, CREDITCARDEXPIRATION, USERNAME, PASSWORD, AGE, INCOME, GENDER, PROD_ID_IDX, PROD_ID1, PROD_ID2 … PROD_ID10	200 million
ORDERS	ORDERID, ORDERDATE, CUSTOMERID, NETAMOUNT, TAX, TOTALAMOUNT	120 million
ORDERLINES	ORDERLINEID, ORDERID, PROD_ID, QUANTITY, ORDERDATE	600 million
PRODUCTS	PROD_ID, CATEGORY, TITLE, ACTOR, PRICE, SPECIAL, COMMON_PROD_ID1, COMMON_RATING1, COMMON_PROD_ID2, COMMON_RATING2, COMMON_PROD_ID3, COMMON_RATING3	1 million
INVENTORY	PROD_ID, QUAN_IN_STOCK, SALES	1 million
REORDER	PROD_ID, DATE_LOW, QUAN_LOW, DATE_REORDERED, QUAN_REORDERED, DATE_EXPECTED	variable
CATEGORIES	CATEGORY, CATEGORYNAME	16

Table 2: DVD Store Database Schema

The CUSTOMERS table was pre-populated with two hundred million customers, one hundred million US customers and one hundred million customers from the rest of the world. The ORDERS table was pre-populated with ten million orders per month for a full year. The ORDERLINES table was pre-populated with an average of 5 items per order. The PRODUCTS table contained one million DVD titles, each with a principal actor listed for search purposes. For realism titles and actor names are generated by taking combinations of real movie titles and actor names. Additionally, the CATEGORIES table contained the 16 DVD categories. .

The schema is fully documented in the database build script in Appendix A.

InnoDB Tables, MyISAM Tables and Indexing

All of the tables used in the DVD Store database were created as InnoDB tables except for the PRODUCTS table which was created as a MyISAM table. InnoDB was used specifically because of its support for transactions and foreign keys. MyISAM was used for PRODUCTS to take advantage of the full text search capability. InnoDB tables do not have the option of creating a full text index type and so can only be searched with a select statement using wild cards around the string. With a MyISAM table it was possible to create a full text index type on the ACTOR and TITLE columns and then use a SELECT statement like the one below to do a full text search for a given string:

select * from PRODUCTS where MATCH (TITLE) AGAINST ('WIND');

The select using wildcards for a title or actor string against the million row PRODUCTS table setup as an InnoDB table took approximately 3 seconds. A select for a title or actor string using the full text index and the MATCH syntax took less than a half a second against the same PRODUCTS table setup as a MyISAM table.

In order to still be able to decrement the product inventory as part of a purchase transaction, the inventory information was put into the InnoDB based INVENTORY table. This also had the added benefit of making the PRODUCTS table only accessed for browsing and keeping the writes on the INVENTORY table.

Foreign key constraints were added maintain database consistency. To prevent ORDER rows from being created without a corresponding customer from the CUSTOMER table and to prevent ORDERLINE rows without a corresponding ORDER from being created.

The database indexing scheme is documented in Appendix B.

In order to pre-populate all of the data in the DVD store database, all of the tables, indices, and foreign key constraints were created first and then the data was loaded from large text files. The indices are created as the data is loaded into the tables which was faster than creating the indices on tables after they were populated with data.

The PHP Application

The web interface to the MySQL DVD Store database was implemented through four PHP pages. The first two were used during the login phase. If the customer was a returning customer, *dslogin.php4* was used to retrieve the customer's information, in particular the CUSTOMERID. If the customer was a new customer, *dsnewcustomer.php4* was used to create a new row in the CUSTOMERS table with the user's data. Following the login phase the customer might search for a DVD by category, actor or title. This was implemented by *dsbrowse.php4*.

Finally, after the user had made his or her selections, the *dspurchase.php4* page totals up the order amount, simulates a credit card lookup, checks QUANTITY_IN_STOCK for the particular item and enters the order and its items into the ORDERS and ORDERLINES tables.

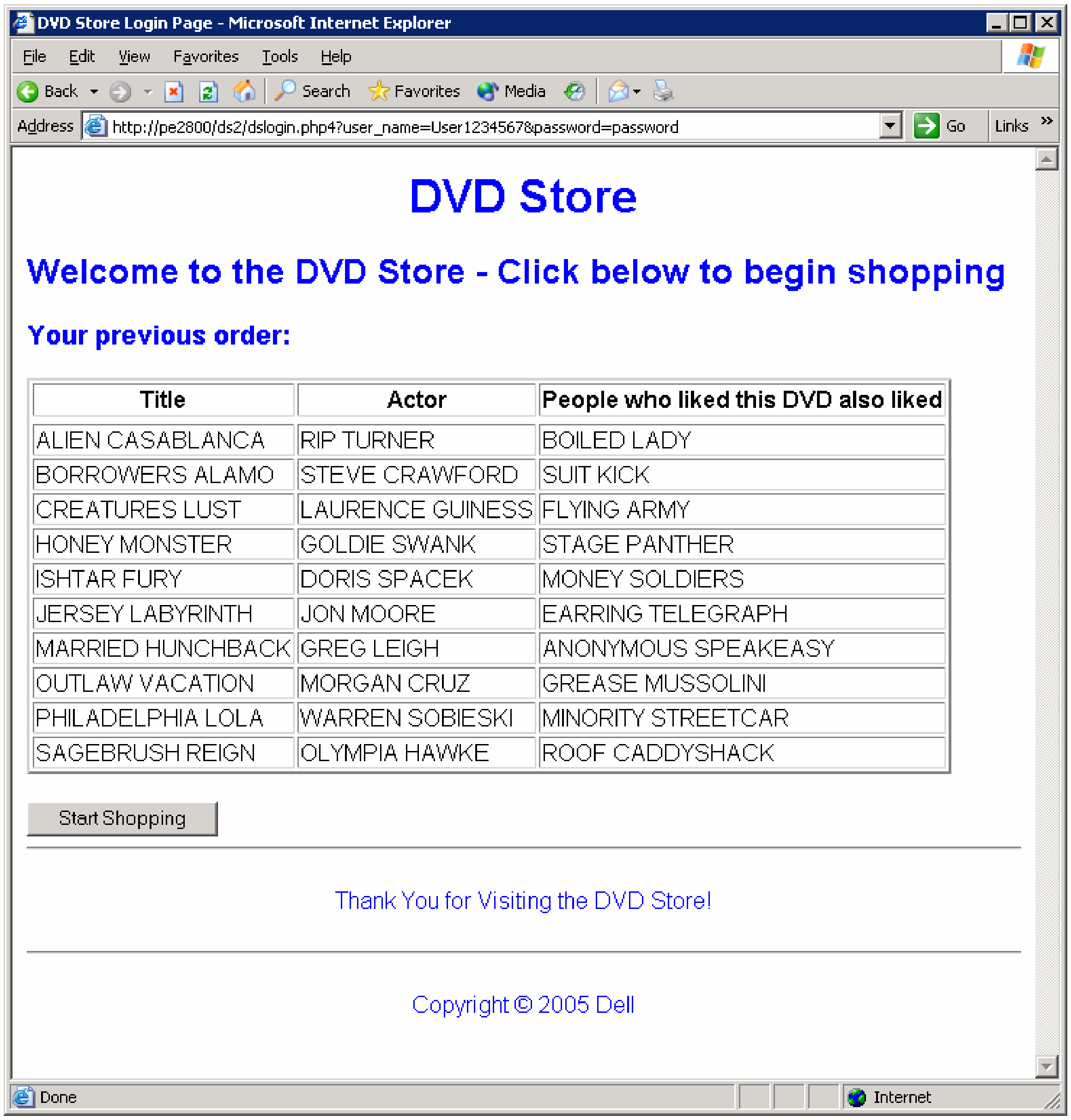
The DVD Store application has features to better model today's online stores. In the *dslogin* PHP page, for example, the user's previous order (up to ten titles) is reported, along with titles that other customers who like those titles have recommended. Searches by category return those titles in the specified category that are currently on sale. And the *dspurchase* PHP page checks the QUAN_IN_STOCK field from the INVENTORY table to see if a title is available. This is done using a database transaction, so that if there is insufficient quantity to fill the order neither the QUAN_IN_STOCK data is updated nor is a new record written to the ORDERS table. To implement that from the PHP application layer the PHP command,

```
 mysql_query('START TRANSACTION;')
```

is used before the inventory check and insert into the ORDERS and ORDERLINES tables are performed. Depending on the success or failure of the entire sequence (represented in the code by whether the $success variable is TRUE or FALSE) the INVENTORY, ORDERS and ORDERLINE updates are either all committed or all rolled back. The PHP code to do this is

```
  if ($success) mysql_query('COMMIT;');
    else  mysql_query('ROLLBACK;');
```

A typical DVD Store screen is shown in Figure 1. This is the page that is shown upon successful login for a returning user.


DVD Store Login Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://pe2800/ds2/dslogin.php4?user_name=User1234567&password=password
Go Links
Done
Internet

DVD Store

Welcome to the DVD Store - Click below to begin shopping

Your previous order:

Title	Actor	People who liked this DVD also liked
ALIEN CASABLANCA	RIP TURNER	BOILED LADY
BORROWERS ALAMO	STEVE CRAWFORD	SUIT KICK
CREATURES LUST	LAURENCE GUINESS	FLYING ARMY
HONEY MONSTER	GOLDIE SWANK	STAGE PANTHER
ISHTAR FURY	DORIS SPACEK	MONEY SOLDIERS
JERSEY LABYRINTH	JON MOORE	EARRING TELEGRAPH
MARRIED HUNCHBACK	GREG LEIGH	ANONYMOUS SPEAKEASY
OUTLAW VACATION	MORGAN CRUZ	GREASE MUSSOLINI
PHILADELPHIA LOLA	WARREN SOBIESKI	MINORITY STREETCAR
SAGEBRUSH REIGN	OLYMPIA HAWKE	ROOF CADDYSHACK

Start Shopping

Thank You for Visiting the DVD Store!

Copyright © 2005 Dell

Figure 1: DVD Store Screen for Returning Customer

The Online Transaction Processing Driver Application

A multi-threaded driver program was written to model an order entry or online transaction processing (OLTP) workload. Each thread of the OLTP driver application connected to the Apache web server and simulated users logging in or entering new customer data, then browsing and purchasing DVDs. Each completed sequence by a customer counted as a single order. The driver measured order rates and the average response time to complete each order. Several tunable parameters were used to control the application and are described in Table 3.

Parameter	Description	Value(s) used in test
n_threads	Number of simultaneous connections to the database	See Table 4
warmup_time	Warmup time before statistics are kept	1 min
run_time	Run time during which statistics are kept	varied
pct_returning	Percent of users that are returning users	80%
pct_new	Percent of users that are new users	20%
n_searches	Number of searches	Range: 1 - 6 Average: 3.5
n_line_items	Number of items purchased	Range: 1 – 9 Average: 5
think_time	Time between web requests for each simulated user	0.25 sec

Table 3: OLTP Driver Parameters

Section 6

Results

The testing results showed that with the PowerEdge 2800 configured only with 8 internal disks and two processors a rate of 1,976 orders per minute was achieved. The order rate was achieved at less than 90% CPU utilization to simulate the maximum level that a server should be run at in a real world scenario to leave room for spikes in usage. If an average order was $25 and the order rate of 1,976 were maintained for an entire year, the server would have processed approximately $25.8 billion. This is accomplished with a server and software solution that costs under $9,000.[1] For complete results see table below.

.

Configuration	Simultaneous Driver Threads	Orders Per Minute (larger is better)	Average Response Time (s)	CPU Utilization %	Total Hardware and Software Price
1 x 3.0 GHz/1MB L2 Cache 1 GB RAM	33	1175	.321	89	$6,889
1 x 3.0 GHz/1MB L2 Cache 2 GB RAM	34	1201	.329	89	$7.589
1 x 3.0 GHz/1MB L2 Cache 4 GB RAM	35	1215	.345	90	$8,189
2 x 3.0 GHz/1MB L2 Cache 1 GB RAM	60	1845	.570	88	$7,482
2 x 3.0 GHz/1MB L2 Cache 2 GB RAM	80	1901	1.158	90	$8,182
2 x 3.0 GHz/1MB L2 Cache 4 GB RAM	85	1967	1.217	89	$8,782

Table 4: DVD store testing results from one and two processor configurations with memory ranging from 1GB to 4GB. All server prices taken from dell.com as of March 2, 2005 and include hardware and software.[1]

Adding a processor increased orders per minute from 1215 to 1967 in the 4 GB RAM configurations, a performance increase of about 60 percent, which makes it a more effective upgrade than additional memory. It important to understand that after adding the second processor, the tests appeared to be disk limited which can be seen by the larger response times in the dual processor test configurations. Adding more disks through Dell PowerVault SCSI external storage or a Dell/EMC SAN would most likely be the best way to improve performance at this point.

Simultaneous driver connections cannot be mapped to actual users. In order to be able to drive the PowerEdge 2800 server to a 90% CPU utilization a think time of only .25 seconds between web pages was used. A real user would most likely have a much higher think time between pages which would allow a much higher number of actual users to be supported by this server than the number of driver threads that were used.

Section 7

Conclusions

Running a MySQL database on a Dell PowerEdge 2800 server with eight internal disks provides a high level of performance for a low cost. Many small and medium sized businesses should be able to use this configuration to support transactional database workloads. Although it is possible to achieve much higher levels of performance with larger servers and more disks, the MySQL database running on a Dell PowerEdge 2800 with only internal disks provides a level of performance that is more than enough for what many customers need.

[1] The Dell PowerEdge 2800 was priced at $8187 with a configuration of 2x3.0 Ghz Xeon processors, 4GB RAM, 8 146GB 10k rpm hard disks, 3 year bronze service and support and SuSE Linux Enterprise Server operating system in the small and medium business section of www.dell.com on March 2, 2005. An additional cost of $595 for MySQL Network was added to include the cost of the database software used. Total hardware and software cost was $8782. The same PowerEdge 2800 server was also tested and priced with the processors and memory indicated in Table 4.

[2] MySQL Network was available from MySQL at http://www.mysql.com/network for $595 per year per physical server as of April 1, 2005.

[3] This term does not connote an actual operating speed of 1 Gb/sec. For high speed transmission, connection to a Gigabit Ethernet server and network infrastructure is required.

Section 8

Acknowledgements

The authors would like to thank Judy Chavis for the server used in the testing and Peter Zaitsev for his assistance in tuning and configuring MySQL.

Appendix A. mysqlds2 Build Script

```
-- mysqlds2_create_db.sql: DS Database Build Script - MySQL version
-- Todd Muirhead/Dave Jaffe 3/9/05
-- Copyright Dell Inc. 2005


-- Database

DROP DATABASE IF EXISTS DS2;
CREATE DATABASE DS2;
USE DS2;


-- Tables

CREATE TABLE CUSTOMERS
  (
  CUSTOMERID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  FIRSTNAME VARCHAR(50) NOT NULL,
  LASTNAME VARCHAR(50) NOT NULL,
  ADDRESS1 VARCHAR(50) NOT NULL,
  ADDRESS2 VARCHAR(50),
  CITY VARCHAR(50) NOT NULL,
  STATE VARCHAR(50),
  ZIP INT,
  COUNTRY VARCHAR(50) NOT NULL,
  REGION TINYINT NOT NULL,
  EMAIL VARCHAR(50),
  PHONE VARCHAR(50),
  CREDITCARDTYPE INT NOT NULL,
  CREDITCARD VARCHAR(50) NOT NULL,
  CREDITCARDEXPIRATION VARCHAR(50) NOT NULL,
  USERNAME VARCHAR(50) NOT NULL,
  PASSWORD VARCHAR(50) NOT NULL,
  AGE TINYINT,
  INCOME INT,
  GENDER VARCHAR(1),
  PROD_ID_IDX INT NOT NULL,
  PROD_ID1 INT NOT NULL,
  PROD_ID2 INT NOT NULL,
  PROD_ID3 INT NOT NULL,
  PROD_ID4 INT NOT NULL,
  PROD_ID5 INT NOT NULL,
  PROD_ID6 INT NOT NULL,
  PROD_ID7 INT NOT NULL,
  PROD_ID8 INT NOT NULL,
  PROD_ID9 INT NOT NULL,
  PROD_ID10 INT NOT NULL
  )
  TYPE=InnoDB;


CREATE TABLE ORDERS
  (
  ORDERID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  ORDERDATE DATE NOT NULL,
  CUSTOMERID INT,
  NETAMOUNT NUMERIC(12,2) NOT NULL,
  TAX NUMERIC(12,2) NOT NULL,
  TOTALAMOUNT NUMERIC(12,2) NOT NULL
  )
  TYPE=InnoDB;


CREATE TABLE ORDERLINES
  (
  ORDERLINEID SMALLINT NOT NULL,
  ORDERID INT NOT NULL,
  PROD_ID INT NOT NULL,
  QUANTITY SMALLINT NOT NULL,
  ORDERDATE DATE NOT NULL
  )
  TYPE=InnoDB;


CREATE TABLE PRODUCTS
  (
  PROD_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  CATEGORY TINYINT NOT NULL,
  TITLE VARCHAR(50) NOT NULL,
  ACTOR VARCHAR(50) NOT NULL,
  PRICE NUMERIC(12,2) NOT NULL,
  SPECIAL TINYINT,
  COMMON_PROD_ID1 INT NOT NULL,
  COMMON_RATING1 INT NOT NULL,
  COMMON_PROD_ID2 INT NOT NULL,
  COMMON_RATING2 INT NOT NULL,
  COMMON_PROD_ID3 INT NOT NULL,
  COMMON_RATING3 INT NOT NULL
  )
  ;

CREATE TABLE INVENTORY
  (
  PROD_ID INT NOT NULL PRIMARY KEY,
  QUAN_IN_STOCK INT NOT NULL,
  SALES INT NOT NULL
  )
  TYPE=InnoDB;


CREATE TABLE CATEGORIES
  (
  CATEGORY TINYINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  CATEGORYNAME VARCHAR(50) NOT NULL
  )
  TYPE=InnoDB;



  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (1,'Action');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES
(2,'Animation');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES
(3,'Children');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES
(4,'Classics');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (5,'Comedy');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES
(6,'Documentary');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (7,'Drama');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (8,'Family');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (9,'Foreign');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (10,'Games');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (11,'Horror');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (12,'Music');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (13,'New');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (14,'Sci-Fi');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (15,'Sports');
  INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (16,'Travel');


CREATE TABLE REORDER
  (
  PROD_ID INT NOT NULL,
  DATE_LOW DATE NOT NULL,
  QUAN_LOW INT NOT NULL,
  DATE_REORDERED DATE,
  QUAN_REORDERED INT,
  DATE_EXPECTED DATE
  )
  TYPE=InnoDB;
```

Appendix B. mysqlds2 Index Creation Script

```
-- mysqlds2_create_ind.sql: DS Database Index Build Script - MySQL
Server version
-- Todd Muirhead/Dave Jaffe 2/25/05
-- Copyright Dell Inc. 2005
USE DS2;
CREATE UNIQUE INDEX IX_CUST_USERNAME ON CUSTOMERS
  (
  USERNAME
  );

CREATE INDEX IX_ORDER_CUSTID ON ORDERS
  (
  CUSTOMERID
  );

ALTER TABLE ORDERS
  ADD CONSTRAINT FK_CUSTOMERID FOREIGN KEY (CUSTOMERID)
  REFERENCES CUSTOMERS (CUSTOMERID)
  ON DELETE SET NULL
  ;

CREATE UNIQUE INDEX IX_ORDERLINES_ORDERID ON ORDERLINES
  (
  ORDERID, ORDERLINEID
  );

ALTER TABLE ORDERLINES
  ADD CONSTRAINT FK_ORDERID FOREIGN KEY (ORDERID)
  REFERENCES ORDERS (ORDERID)
  ON DELETE CASCADE
  ;

CREATE FULLTEXT INDEX IX_PROD_ACTOR ON PRODUCTS
  (
  ACTOR
  );

CREATE INDEX IX_PROD_CATEGORY ON PRODUCTS
  (
  CATEGORY
  );

CREATE FULLTEXT INDEX IX_PROD_TITLE ON PRODUCTS
  (
  TITLE
  );

CREATE INDEX IX_PROD_SPECIAL ON PRODUCTS
  (
  SPECIAL
  );
```