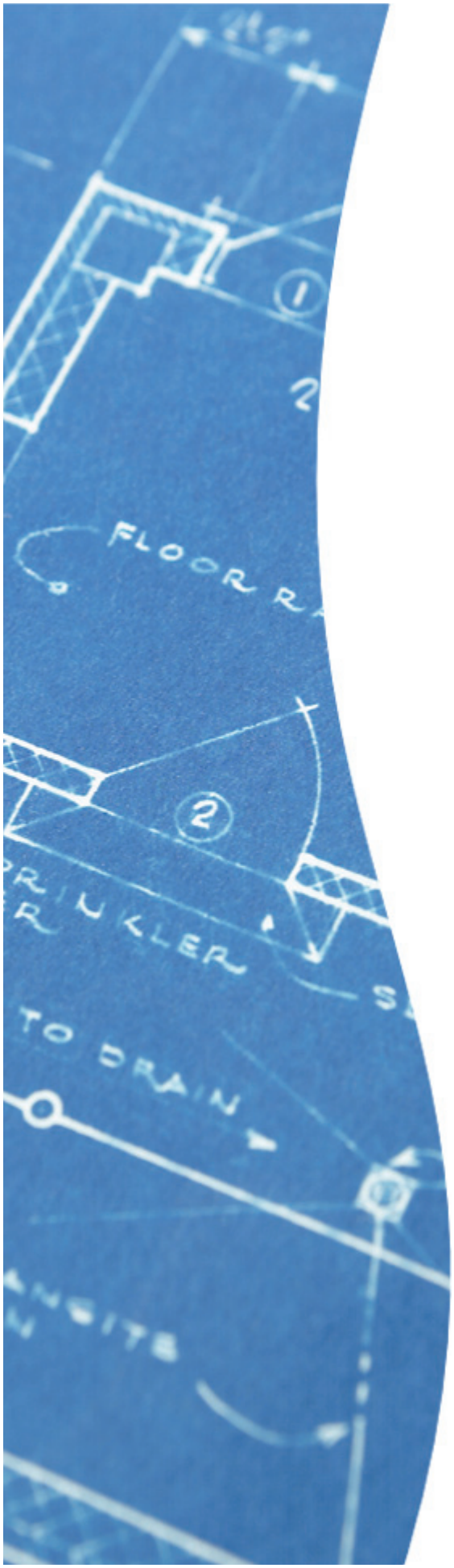

OPTIMIZING MYSQL™ SERVER ON SUN™ X64 SERVERS AND STORAGE

Luojia Chen, ISV Engineering

Sun BluePrints™ Online — February 2008

Part No 820-4498-10
Revision 1.0, 2/20/08
Edition: February 2008

Table of Contents

Chapter 1
Introduction

Today business is increasingly done on the Web, and thousands of new people, applications, businesses, and services are coming online daily. In fact, Wiki pages, mashups, social networking sites, and online stores are at the forefront of Web 2.0 technologies. As more businesses, services, and sites go online and gain in popularity, enterprises must deal with the massive increases in data generated by customer relationship management (CRM) and enterprise resource planning (ERP) applications, as well as collected community knowledge and shared information.

When information is readily available, it can help make the organization smarter and more effective at solving business challenges. As a result, applications need fast, efficient, and flexible database environments that can scale and adapt as more data enters the system. In turn, Web and e-commerce companies need easy access to an open, integrated platform—complete LAMP (Linux, Apache, MySQL, Perl) or SAMP (Solaris™ Operating System, Apache, MySQL, Perl) application stack, open source database, and high-performance servers and storage systems—capable of supporting high traffic, high scale Web sites. This Sun BluePrints™ article provides an overview of the integrated Sun™ platform, outlines the steps for optimizing MySQL™ Server on Sun x64 servers, and describes benchmark results for solutions with Sun Fire™ X4100 servers.

Sun™ Platforms and MySQL™—An Integrated Solution Stack

With the addition of MySQL to its software portfolio, Sun now offers an integrated LAMP or SAMP software stack that runs on the entire range of Sun's x64 servers and storage systems. Together, these hardware and software components give enterprises the ability to deploy on open, scalable platforms with low total cost of ownership.

Figure 1-1. Sun x64 servers, storage arrays, operating systems, and MySQL provide an integrated foundation for Web and e-commerce environments

- Sun Fire x64 servers
  With a modular approach that ranges from very low cost single rackmount servers to integrated solutions, the Sun Fire x64 server family gives IT organizations the flexibility to put computing resources where they are needed most. Combining Sun's network computing expertise with the performance and power advantages of AMD Opteron™ and Intel® Xeon® processors, the Sun Fire x64 server family offers high speed and energy efficiency to database and Web 2.0 deployments on an open platform.

  All Sun Fire x64 servers can run the Solaris 10 Operating System or Linux environment and the full LAMP or SAMP software stack without modification. In addition, Sun Fire x64 servers incorporate a variety of performance and availability features that support business-critical Web deployments. For example, Sun Fire x64 servers with AMD Opteron processors utilize HyperTransport technology. Providing a scalable, direct connection between processors, I/O subsystems, and other chip sets, HyperTransport technology reduces the number of buses in the system, which in turn helps reduce transaction overhead to improve database transaction and query performance.

- Sun StorageTek™ disk arrays
  Storage is a key component of any database environment. Sun StorageTek arrays scale from media trays to workgroup, mid-range, and datacenter arrays, and network attached storage. The SMI-S compliant, Web-based Sun StorageTek Common Array Manager software helps reduce the complexity of array installation, configuration, and management. Online capacity expansion, data volume control creation, and host to volume mapping features make it easy to modify storage configurations.

  Storage systems must perform well if database environments are to meet service level agreements and other business priorities. Sun storage systems support a variety of drive sizes, capacities, and speeds. Since more drive spindles can help increase database write performance, Sun storage systems can pack high density into a small footprint, and expansion options are often available. Dual active RAID controllers and active-active configuration support available on many Sun StorageTek arrays helps enable load balancing and failover for improved performance and availability.

- A choice of operating systems
  More than ever before, businesses are looking to standardize on operating system and server hardware architectures to help reduce management costs. While all Sun Fire x64 servers come pre-loaded with the industry-leading Solaris 10 Operating System at no cost, they also can run virtually any operating system — with full support from Sun for Red Hat and SuSE versions of Linux, as well as Microsoft Windows Server and VMware.

- MySQL Server
  One of the world's most popular open source databases, MySQL Server powers many of the largest, high volume Web sites and business-critical systems. A fully integrated, transaction safe, ACID compliant database with commit, rollback, crash recovery, and row level locking capabilities, MySQL Server is a reliable foundation for e-commerce, online transaction processing (OLTP), and data warehousing solutions.

Chapter 2

Steps for Optimizing MySQL Server

Optimizing MySQL Server takes planning and understanding of the application running on the database, the service level agreements (SLAs) required, and the type of I/O needed for the application. However, MySQL Server can be used for a wide variety applications and data types, and one size does not fit all for optimization parameters and settings. Figure 2-1 and the sections that follow outline a way to evaluate the right settings for an application.

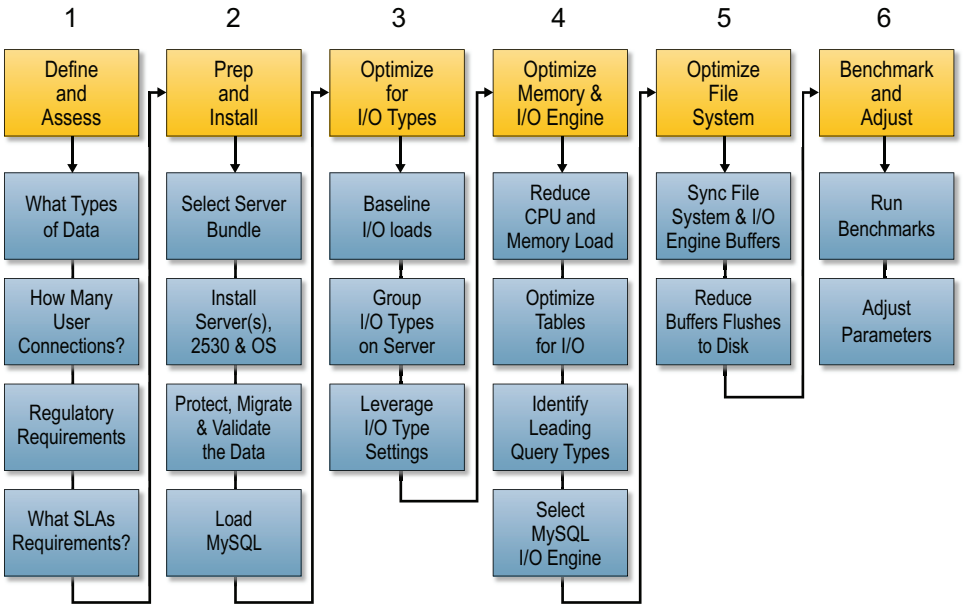

Figure 2-1. Steps for optimizing MySQL Server on Sun platforms

Step 1: Define and Assess

Understanding the data in use is the first step in optimizing the infrastructure underneath MySQL Server. In particular, the nature of the expected read/write activity and ratios is vital to establishing the settings needed to help optimize solutions. Finding the optimal buffers and caches that can help reduce excess memory and I/O traffic based on data characteristics is key.

Data Types

Knowing the data type(s) in use aid in understanding what type of pipeline to create to optimize data transfers and data movement though the server, as well as reduce memory usage, optimize read/write operations out of cache to improve performance in and out of caches. Each data type requires different pipelining through the servers and on the storage array (Table 2-1).

Table 2-1. Data types and pipelining parameters

Operation	Block Size	Mode
Write	Small	Random
Write	Large	Random
Write	Small	Sequential
Write	Large	Sequential
Read	Small	Random
Read	Large	Random
Read	Small	Sequential
Read	Large	Sequential

User Connections

Knowing the anticipated number of user connections up front enables a better understanding of:

- The overall workload on the server
- The MySQL storage engine to load
- How to set the buffers and caches
- How to distribute I/O across the SAS links to the storage array
- How to configure the array to support applications

MySQL Server runs as a single process, multithreaded application that has one master thread with highest priority to control the server. However, MySQL Server also creates a dedicated user thread running at normal priority in the thread pools for each simultaneous client request. This dedicated user thread processes the user request and sends back the result to each client once the result is ready. An additional, single user thread waits for input from the console, and a group of utility threads running at lower priority handle some background tasks.

Currently, MySQL Server exhibits limited scalability with the number of concurrent user threads handling client requests. Performance scales efficiently with each additional user thread until it reaches the peak performance point. Once the peak is reached, increasing the number of user connections can decrease MySQL Server performance due to thread concurrency contention. For applications that can tune the number of user connections, it is important to determine the optimum number of user connections for peak performance for different workloads.

Regulatory Requirements

It is important to consider regulatory requirements prior to planning the optimization effort. The answers to several questions can aid the process, including:

- Is the application subject to specific regulatory requirements?
- Is the application subject to security or access control requirements?
- Are there any business continuity needs to consider?
- Does the data need to be stored in a dedicated LUN?
- Does the data need to replicated synchronously or asynchronously?
- Does the data need to be replicated locally or remotely?

Service Level Agreement Requirements

Service level agreements help set the business continuance requirements for applications, and drive several data and storage requirements, including:

- Type of storage and data protection model to implement on the array
- Data backup procedures
- Frequency of backups and snapshots

Step 2: Preparation and Installation

Preparing and installing the Sun servers and the Sun StorageTek 2530 array starts with five basic tasks.

- *Select server platforms*
  MySQL applications under 1 TB in size that do not require highly available configurations can use a single Sun Fire X4100 M2 server and a Sun StorageTek 2530 array loaded with 73 GB disk drives. Databases over 1 TB in size that require highly available servers can use two Sun Fire X4200 M2 servers, each with two SAS host bus adapters connected to a Sun StorageTek 2530 array loaded with 146 GB disk drives.
- *Determine the physical server location*
  It is important to find the right rack space, power, and cooling to support the servers, storage systems, and any required switches or other infrastructure components.
- *Install the servers, storage systems, and Solaris Operating System*
  Install the Sun Fire X4100 M2 or Sun Fire X4200 M2 server(s), the Sun StorageTek 2530 array, and the Solaris 10 Operating System. In addition, set up the basic LUN and RAI configuration to hold new or migrated data. Detailed installation and configuration information can be found in the product guides located at *http://docs.sun.com/app/docs/prod/5b03d0ed-216d-11db-a023-080020a9ed93*.

- *Protect, migrate, and validate data*
  If the hardware is replacing an existing server, be sure to make a complete backup of the dataset to be moved. The backup can be placed on another disk array for rapid access. Next, migrate the data from the existing storage system to the Sun StorageTek 2530 array. Finally, validate that the dataset was moved properly, especially if the backup and original copies are to be erased.
- *Load MySQL*
  The next step in the optimization process is to load MySQL onto the Sun Fire X4100 M2 or Sun Fire X4200 M2 server(s) and test access to the data on the storage array.

Step 3: Optimize for I/O Types

Optimizing I/O requires baselining I/O loads, grouping I/O types, and leveraging I/O settings.

- *Baseline I/O loads* — First, baseline current workloads and I/O types on the server. Doing so helps measure optimizations during the final benchmarking phase.
- *Group I/O types* — Optimizations for small, random reads and writes are not the same as those for larger, sequential reads and writes. If multiple I/O types are in use, try to segment them on multiple servers or virtual machines.
- *Leverage I/O settings* — Grouping similar I/O types together enables the proper cache, files system settings, and MySQL storage engines to be used for optimization.

Step 4: Optimize MySQL Memory and CPU Options

The next step in the optimization process is to reduce unneeded functions that steal memory and CPU performance, including extra buffers that cause additional memory moves and consume CPU resources.

Reduce CPU and Memory Uses

One of the key advantages of open source software solutions is the ability to optimize the software for different needs. Use the `mysql> show` variables to help identify the configuration and system variables. The values for these variables can be changed in the *my.cnf* configuration file. In addition, the `mysql` startup options can be used to remove or adjust the following MyISAM memory-related variables if only Innodb tables are in the system:

- bulk_insert_buffer_size
- key_buffer_size
- key_cache_age_threshold, key_cache_block_size, key_cache_division_limit
- read_buffer_size, read_rnd_buffer_size

Optimize Tables

Make sure the tables in the database are optimized for the type of queries and sorts required by the application.

Identify Common Queries

The MySQL query cache stores the identical `SELECT` queries issued by clients to the database server. This makes it possible to locate and re-issue the same queries without repetitive hard parsing activities. MySQL also stores the query result set in the query cache, which can reduce significantly the overhead of creating complex result sets for queries from the disk or memory caches, reducing both physical and logical I/O. This can improve application performance when repetitive queries of products are being issued. Some cache guidelines include:

- If a high value for `qcache_hits` is seen compared to the total queries at runtime, or a low value for `qcache_free_memory` is seen via the `mysql>show status` command, it might be necessary to increase the value of the `query_cache_size` parameter accordingly. Otherwise, decrease the value of the `query_cache_size` parameter to save memory resources for other MySQL cache buffers.
- If `qcache_hit` is `0` during runtime, turn off the query cache by setting `query_cache_type` to `0`, and set `query_cache_size` to `0`, to reduce the overhead associated with having the query cache enabled and free memory resources.
- If the application uses many simple `SELECT` queries without repeating, enabling the query cache can impede performance by five to ten percent. However, setting the query cache can help applications with many repeated `SELECT` queries to improve performance by `200` percent or more.
- Keeping most of the database tables open can be expensive. The optimum value for the *`table_cache`* parameter is directly related to the number of tables that need to be open simultaneously in order to perform multiple-table joins. The *`table_cache`* value should be equal to no less than the number of concurrent connections times the largest number of tables involved in any one join. Typically, `1,024` is good value for applications with a couple of hundred tables. Each connection has its own entry. Check the *`Open_tables`* status variable to see if it is large compared to the *`table_cache`* setting.

Select and Configure the MySQL Storage Engine

MySQL has many variables that can be adjusted to change MySQL behavior or performance characteristics. Because MySQL includes several storage engines, including MyISAM, InnoDB, HEAP, and Berkeley DB (BDB), some variables apply to only one storage engine, and other variables are used in the SQL layer applying to all storage engines. In addition, several memory-related variables apply to all storage engines.

Table 2-2 describes some of the MySQL storage engine variables and suggested settings and guidelines.

Table 2-2. MySQL storage engine variable descriptions and setting guidelines

Variable	Description and Guidelines
join_buffer_size	The buffer used for full joins. In case where there are large joins without indexes, increase this buffer size to improve efficiency.
sort_buffer_size	The buffer used for the sort result set allocated by each thread. Use of this buffer can speed ORDER BY and GROUP BY queries.
query_cache_size	Set this variable to a nonzero value to enable query caching.
query_cache_limit	The maximum size for a cached result set. Larger result sets are not cached.
query_cache_min_res_unit	Specifies the minimum size for the memory blocks allocated by the query cache. When an application has many queries with small results, the 4 KB default block size can lead to memory fragmentation. Decreasing the block size to 2 KB or 1 KB might improve performance. For large query result sets, increasing the block size to 8 KB, 16 KB or more can speed performance.
query_cache_type	0=OFF, 1=ON

Several MySQL Innodb memory-related variables are also important.

- `innodb_buffer_pool_size`
  The `innodb_buffer_pool_size` variable is used to set the amount of memory allocated to Innodb data and the index buffer cache. MySQL does not access disks directly. It reads data into the internal buffer cache, reads and writes blocks, and flushes the changes back to the disk. If the server requests data that is available in the cache, the data is processed immediately. Otherwise, the operating system requests that the data be loaded from disk. The larger the cache size, the more disk accesses can be avoided. For example, increasing `innodb_buffer_pool_size` from 4 GB to 5 GB on Sun Fire T2000 servers with 8 GB RAM for the sysbench I/O bound workload test can improve performance by approximately 11 percent.
- `innodb_additional_mem_pool_size`
  The `innodb_additional_mem_pool_size` variable sets the amount of memory allocated to the buffer that stores the InnoDB internal data dictionary and other internal data structures. Because this parameter does not affect performance significantly, it can be set to 20 MB for sysbench OLTP testing. Note that more memory should be allocated for applications with higher table counts.

- innodb_log_buffer_size
  The `innodb_log_buffer_size` variable sets the amount of memory allocated to the buffer that store InnoDB write-ahead log entries. For large transactions, if `innodb_flush_log_at_trx_commit` is set to one before the log buffer is flushed on each transaction commit, the log can be loaded into the log buffer instead of flushing to the disk in the background to reduce disk I/O. If large log I/O is seen via the `show innodb status` output at runtime, a larger `innodb_log_buffer_size` parameter value likely is needed. Since the sysbench OLTP I/O bound workload does not perform long transactions, it is not necessary to waste memory resources by setting a higher value for the log buffer. Set the size to 8 MB.

Other MySQL Innodb variables can impact I/O performance.

- `innodb_flush_log_at_trx_commit`
  InnoDB flushes the transaction log to disk in the background approximately once per second. By default, the `innodb_flush_log_at_trx_commit` parameter variable is set to one to flush the log to the disk on each transaction commit in order to help avoid transaction loss during a MySQL, operating system, or hardware crash. For workloads running with many short transactions, disk writing can be reduced by setting the `innodb_flush_log_at_trx_commit` parameter with different values. Setting this value to zero results in no log flushing on each transaction commit. While doing so can reduce disk I/O and improve performance, transactions can be lost in the event of a MySQL crash or failure.

  Setting `innodb_flush_log_at_trx_commit` to zero during read-only tests can improve performance by 4 percent. When the value is set to two, the log flushes to the operating system cache (file system cache) instead of the disk on each transaction commit. While setting can also reduce disk I/O, performance is slightly slower than when the value is set to zero. However, transaction loss is less likely in the event of operating system or hardware failures.

- `innodb_log_file_size`
  InnoDB writes to the log files in a circular fashion. If the log file reaches the configuration limit set by the `innodb_log_file_size` parameter, the checkpoint operation is executed at once to flush the modified database pages. The `innodb_log_file_size` parameter is very important for write-intensive workloads and large data sets. While larger sizes reduce disk I/O in checkpointing and help improve performance, recovery times are also increased. The `innodb_log_file_size` parameter must be increased when large page writes are seen in the `BUFFER POOL AND MEMORY` portion of the `show innodb status` output.

Step 5: Optimize File Systems

The MySQL storage engine and I/O types in use indicate how the file system should be configured to eliminate double buffering and disk flushing.

Synchronize the File System I/O Engine Read/Write Options

File system performance has a significant impact on overall system performance, particularly when running I/O bound workloads with a database size that is larger than system memory. Strategies for configuring the file system for improved performance depend on the workload access pattern. For sequential workloads, increasing the file system cluster size to enable reading ahead or writing back more data from or to the disk can help reduce the total number of I/O operations. For random workloads, reducing the file system cluster size to match the innodb I/O size is often optimal.

Reduce Double Buffers and Flushes to Disk

Review the buffering model of the MySQL storage engine to remove double buffering. Double buffering typically occurs when the file system and I/O engine are caching data, causing extra memory moves and increased CPU cycles. Different versions of MySQL can mitigate this issue in different ways. For MySQL 5.0.42 and 5.1.18 and earlier, mount UFS with the `forcedirectio` option. For later MySQL 5.0.x and 5.1.x releases, directio can be enabled in the *my.cnf* file by setting `innodb_flush_method` to `O_DIRECT`.

Step 6: Benchmark and Adjust

The last step in the optimization process is to put everything together and make final adjustments. Using the baseline established, run identical tests to measure improvement. While additional tuning can help, this process helps ensure organizations obtain optimal results from the solution.

Chapter 3

Optimizing MySQL on Sun Fire™ X4100 M2 Servers

Understanding how a system can perform under load conditions is key to ensuring desired performance characteristics can be met during peak load. Sysbench is a modular, cross-platform, multithreaded tool for evaluating a system running a database under intensive load conditions. In particular, sysbench enables database server performance to be stressed for online transaction processing workloads.

MySQL sysbench Benchmark Test Environment Configuration

In order to understand the performance characteristics of MySQL in on Sun platforms, Sun engineers tested a Sun Fire X4100 M2 server running the Solaris 10 Operating System and MySQL using the sysbench tool. Table 3-1 lists the hardware and software configuration used for the benchmark tests.

Table 3-1. Hardware and software test configuration

	Description	Configuration
Server	Sun Fire X4100 M2 server	• Four 2.8 MHz CPUs • 7,680 MB memory
Storage	Sun StorageTek 2530 array	• 1,273 GB 15K SAS drives • RAID-5 • 1 storage group/host • 2 storage controllers • PCI-X SAS host bus adapter
Software	Operating System	• Solaris 10 Operating System 6/06
	Database Software	• MySQL Database Server 5.0.45
	Test Applications	• sysbench

Sun StorageTek™ 2530 Array Configuration

The test effort used the default storage CAMS profile for configuring the volume with RAID level 5. The default 512 KB segment size was used, and read-ahead mode was enabled.

MySQL Database Server Installation

The following commands were used to install the MySQL database server.

```
shell> cd /usr/local
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> scripts/mysql_install_db
shell> chmod -R 777 *
shell> bin/mysqld_safe --user=mysql &
```

The following commands were used to install the sysbench test software.

```
shell> cd /sysbench-version
shell> ./configure
shell> make
shell> make install
```

The following commands were used to load a 100 million row test database.

```
shell> sysbench –test=oltp –oltp-table-size=100000000 –mysql-
db=sbtest100m –max-requests=0 prepare
```

MySQL Settings

The following settings in the *mysqld* file were used during the testing effort.

```
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/mysql
datadir = /usr/local/mysql/data
log-error = /data/error.txt
user=root
skip-locking
max_connections = 3000
table_cache = 1024
max_allowed_packet = 1M
sort_buffer_size = 64K
thread_cache = 8
thread_concurrency = 16
query_cache_size = 0M
query_cache_type = 0
default-storage-engine = innodb
transaction_isolation = REPEATABLE-READ
tmp_table_size = 1M
innodb_data_file_path = ibdata1:100M:autoextend
innodb_buffer_pool_size = 5500M
innodb_additional_mem_pool_size = 20M
innodb_log_file_size =1900M
innodb_log_buffer_size = 8M
innodb_flush_log_at_trx_commit =1
innodb_lock_wait_timeout = 300
innodb_max_dirty_pages_pct = 90
innodb_thread_concurrency =16
```

Benchmark Results

The sysbench tests conducted performed read-only and read/write tests with two to 256 user connections. The following sections summarize the results achieved.

32 User Read-Only Test Results

A sysbench OLTP I/O bound read-only test was performed with 32 user connections on the Sun Fire X4100 M2 server. Figure 3-1 shows the MySQL performance improvement after tuning the Innodb data and index buffer size, and tuning the UFS file system on the Solaris 10 OS with maxcontig settings and directio.

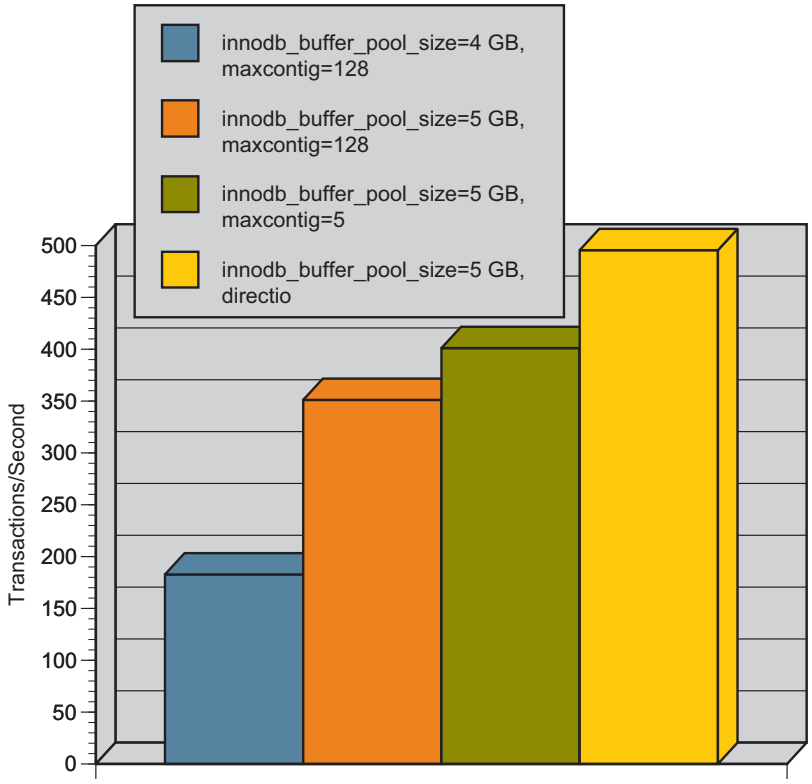

Figure 3-1. Test results for sysbench OLTP I/O bound read-only test with 32 user connections

Table 3-2 and Figure 3-2 depict the test results for a sysbench OLTP I/O bound read-only test with two to 256 user connections.

Table 3-2. Test results

Number of User Connections	Transactions/Second
2	104.73
4	251.33
8	451.56
16	555.64
32	577.96
64	563.84
128	558.66
256	519.96

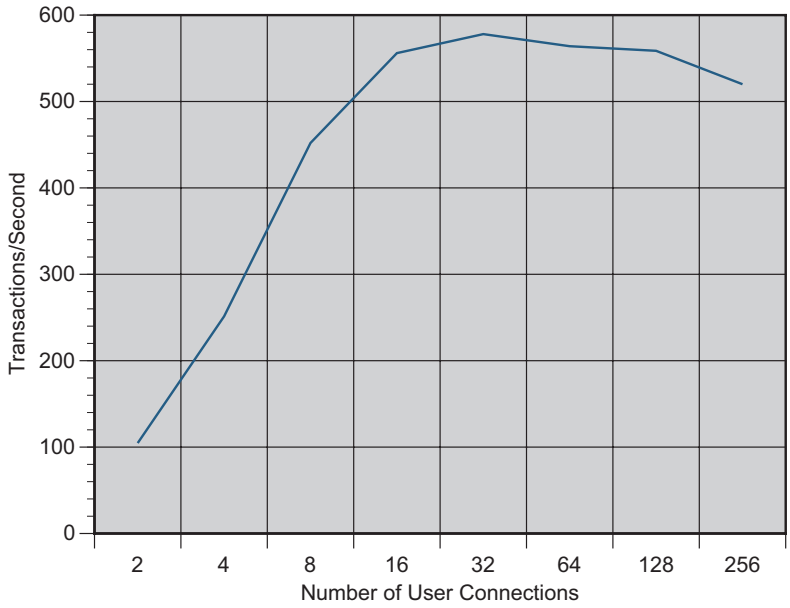

Figure 3-2. Test results for sysbench OLTP I/O bound read-only test with two 256 user connections

32 User Read/Write Test Results

A sysbench OLTP I/O bound read/write test was performed with 32 user connections on the Sun Fire X4100 M2 server. Figure 3-3 illustrates the test results.

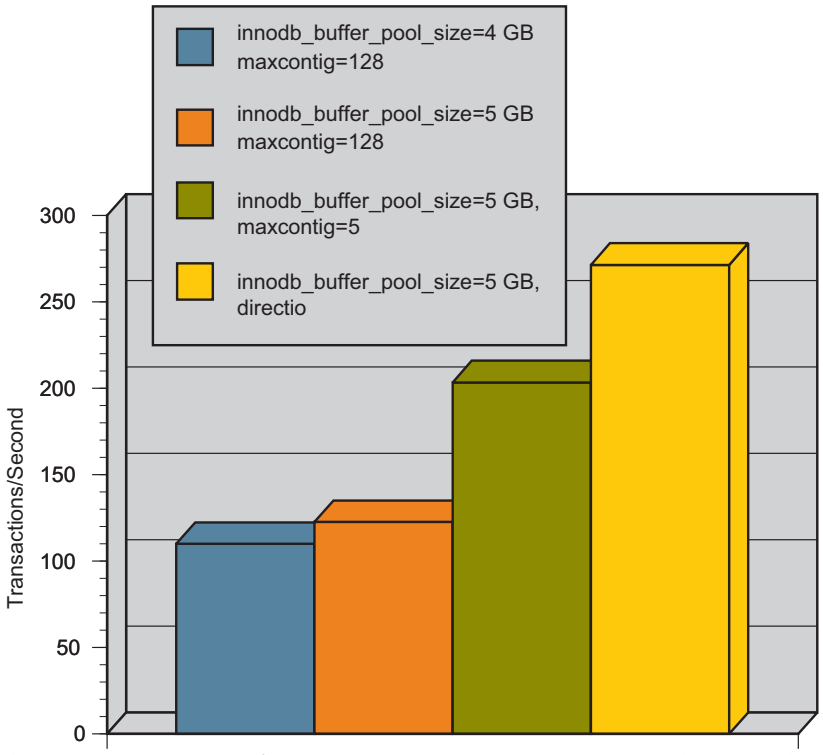

Figure 3-3. Test results for sysbench OLTP I/O bound read/write test with 32 user connections

Table 3-3 and Figure 3-4 depict the test results for sysbench OLTP I/O bound read/write test with two 256 user connections.

Table 3-3. Test results

Number of User Connections	Transactions/Second
2	146.04
4	232.05
8	291.1
16	289.12
32	288.51
64	297.32
128	295.32
256	273.91

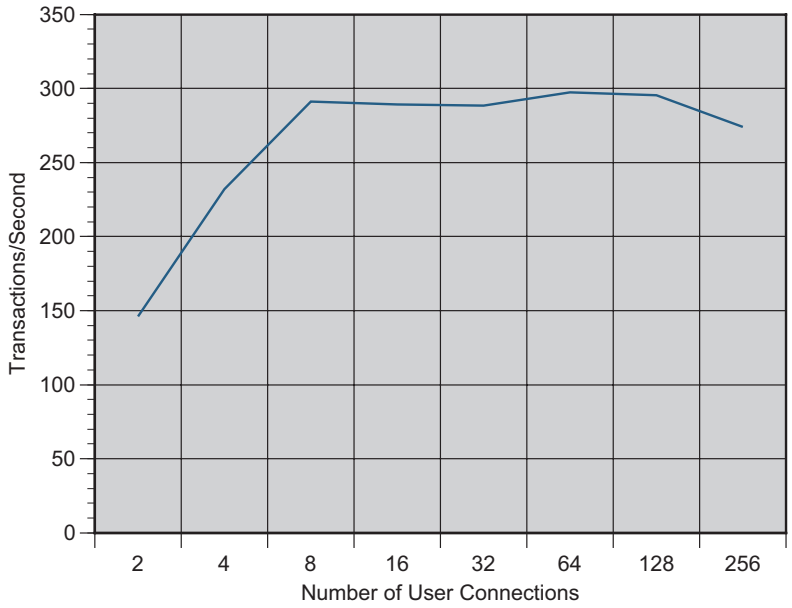

Figure 3-4. Test results for sysbench OLTP I/O bound read/write test with two 256 user connections

Chapter 4
For More Information

About the Author

Luojia Chen is a software engineer in Sun's ISV Engineering organization. Working on the open source team, Luojia specializes in MySQL adoption of key Sun technologies. Currently, she is focused on MySQL benchmarks, performance monitoring, optimization, and scalability in order to understand how to make MySQL run at peak performance on Sun platforms.

References

Table 4-1 lists Sun hardware, software, support, and developer resources.

Table 4-1. Resources

Description	URL
Solaris 10 Operating System	sun.com/solaris
Sun BluePrints Program	sun.com/blueprints
Sun Servers	sun.com/servers
Sun Storage	sun.com/storage
Sun's GNU/Linux Offerings	sun.com/software/linux
Sun to Acquire MySQL Press Kit	sun.com/aboutsun/media/presskits/2008-0116

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is

`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints Online Web site at:

`http://www.sun.com/blueprints/online.html`

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

Printed in USA 02/08