MySQL 5.5: Storage Engine Performance Benchmark for MyISAM and InnoDB

A MySQL® Technical White Paper

January 2011

Table of Contents

Introduction

With the release of MySQL 5.5, InnoDB has become the default storage engine. MyISAM, NDB (MySQL Cluster), Memory, Archive, etc. are included as pluggable storage engines, giving users the flexibility to choose the storage engine that best fits their specific use-case.

InnoDB is the mostly widely used storage engine for Web/Web 2.0, eCommerce, Financial Systems, Telecommunications, Health Care and Retail applications built on MySQL. InnoDB provides highly efficient ACID-compliant transactional capabilities and includes unique architectural elements that deliver high performance and scalability. InnoDB is structurally designed to handle transactional applications that require crash recovery, referential integrity, high levels of user concurrency and fast response times.

While MyISAM and the other storage engines continue to be readily available, users can now create applications built on InnoDB without altering default configuration settings. All of the capabilities of InnoDB are now delivered "out of the box" with any MySQL 5.5 deployment.

The ACID properties of InnoDB are configurable, and so users can ensure ACID-compliance for those workloads demanding the highest levels of data integrity, such as ecommerce, while relaxing ACID properties where throughput is more important. Even with these relaxed properties, benefits such as crash recovery are still maintained by InnoDB, so users enjoy much higher levels of protection than they do with MyISAM.

The purpose of this whitepaper is to directly compare performance of the latest InnoDB 1.1 release included with MySQL 5.5 and MyISAM, using a benchmark that is commonly run to measure MySQL throughput.

Performance is measured for both InnoDB and MyISAM across multi-core commodity systems to provide direct comparisons in both throughput and scalability for Read / Write and Read-Only workloads. Full ACID compliance was relaxed for the InnoDB tests to provide a more comparable analysis with MyISAM. Full details are included in the "Benchmark Results" section.

The paper also discusses the most common use cases for both InnoDB and MyISAM, enabling readers to assess the best storage engine for their specific application requirements.

InnoDB 1.1 Performance and Scalability Enhancements

MySQL 5.5 introduces a re-architected InnoDB 1.1 release that includes many performance and scalability features over and above previous versions. InnoDB extends MySQL 5.5 performance and scalability by adding the following new features:

Improved Default Thread Concurrency – InnoDB now defaults to allow an unlimited number of concurrently executing threads, leveraging the processing power of multi-threaded and multi-core systems. Users can override this default behavior by setting the value of `innodb_thread_concurrency` to the limit that best works for specific applications.

Control of Background I/O Threads – Users now have two new configuration parameters for all platforms, `innodb_read_io_threads` and `innodb_write_io_threads` that allow for the setting of the number of background threads used for read and write requests. This helps users tune and scale their MySQL applications on high-end, multi-core systems

Control of Master Thread I/O Rate – Users can now configure the overall I/O capacity and bandwidth available to InnoDB for running background tasks, via the new `innodb_io_capacity` in the `my.cnf` or `my.ini` files.

Control of Using Operating System Memory Allocators – Users can now control whether InnoDB uses it own memory allocator or leverages the more efficient allocators available in the current versions of the most commonly deployed operating systems. This is easily controlled by setting the value of the new system configuration parameter `innodb_use_sys_malloc` in the MySQL 5.5 option file (`my.cnf` or `my.ini`). The default setting is 1, which instructs InnoDB to use the operating system resource.

Control of Adaptive Hash Indexing – Users can now disable the Adaptive Hash Indexing feature, which is enabled by default in the built-in InnoDB. This is useful when tuning applications or systems; too many threads blocked by RW-latch contention caused by the underlying Adaptive Hash Indexing process (as shown in the output from `SHOW ENGINE INNODB STATUS`) is a good indicator this feature can be disabled to improve performance.

Improved Performance and Scalability on Windows - MySQL has traditionally performed well on UNIX based platforms but not so much on Windows. With more developers now building and deploying applications on Windows, MySQL's footprint has expanded on Windows from the development desktop to the production datacenter. MySQL 5.5 includes Windows specific improvements that ramp up performance and scalability for systems and applications designed to service high concurrency and user loads. The key MySQL 5.5 improvements for Windows include:

- MySQL now uses native Windows synchronization primitives to implement mutexes and locking algorithms.
- MySQL now uses native Windows atomic operations vs. POSIX threads to implement and free read/write specific locks.
- MySQL now uses native Windows operating system memory allocators by default.
- MySQL on Windows I/O thread handles maximum now equals the maximum on other platforms so applications with many tables and connections get greatly reduced table opening and closing overhead.
- Legacy optimizations made on other platforms have now been ported to MySQL on Windows.

Improved Scalability via Faster Locking Algorithm – For most platforms (UNIX, Linux, Windows), InnoDB uses native atomic operations vs. POSIX threads to implement mutexes and read/write locks. These atomic operations have been optimized for the latest generation of multi-core processor designs, serving to boost InnoDB performance and scale.

Restored Group Commit – An optimization that allows InnoDB to perform low-level I/O operations (log write) once for a set of commit operations, rather than flushing and syncing separately for each commit, which can significantly improve throughput.

Improved Recovery Performance - InnoDB is known for its ability to reliably recover data after a crash. The time taken could be high if many pages had been modified but not flushed, so sometimes it was desirable to limit the log file size to force flushes even though this causes reduced performance. MySQL 5.5 includes a number of default optimizations designed to greatly speed up the scanning and applying of redo logs so the next restart is faster. You can now set the log files to their maximum possible sizes to increase the amount of write combining that can be done.

Multiple Buffer Pool Instances - Today's buffer pools are consistently sized in the multi-gigabyte range, data pages are persisted and are constantly being read and updated by different database threads. MySQL 5.5 removes the bottleneck of waiting threads when one thread is updating the

buffer pool. All the structures normally associated with the buffer pool can now be multiplied, such as its protecting mutex, the last recently used (LRU) information, and the flush list. Users can now control and tune how many buffer pool instances are used; however, for backward compatibility the default is still 1. This feature works best with combined buffer pool sizes of several gigabytes, where each buffer pool instance can be a gigabyte or more.

Multiple Rollback Segments - InnoDB can now use multiple rollback segments improving performance and scalability and greatly increasing the number of concurrent transactions that can be serviced. While previous InnoDB versions had a limit of 1,023 concurrent transactions that had modified data, MySQL 5.5 now allows for up to 128k that create undo data. This improvement reduces the contention on the single rollback segment mutex resulting in higher throughput.

Native Asynchronous I/O for Linux - MySQL 5.5 enables improved concurrency of I/O requests on Linux systems. Previous versions of InnoDB have provided "simulated asynchronous I/O" for Linux by internally arranging I/O calls as if they were asynchronous, while behind the scenes the query thread would block competing threads until the I/O request was complete. MySQL 5.5 now provides true native asynchronous I/O support for Linux and Windows based systems. This feature requires the `libaio` userspace library to be installed on Linux and comes with a configuration option `innodb_use_native_aio` that can turned off if the new default setting is not compatible with the host I/O subsystem.

Extended Change Buffering: Now with Delete and Purge Buffering - Like all databases, InnoDB uses indexed columns to make query and primary key lookups more efficient and performant. Secondary indexes, or those on columns other than the primary key, require disk writes to keep them up to date when primary key columns are inserted, deleted, or updated. Previous versions of InnoDB include an optimization that delays disk writes for secondary index maintenance when the changes are due to INSERT operations. Meaning that InnoDB would delay or buffer the changes to secondary indexes until the index contents are read into the buffer pool during normal operations, such as a requesting query. The changes can then be made quickly in memory and then flushed back to disk using the normal schedule for writing dirty blocks. When the changes in the buffer pool affect a group of sequential disk blocks, they can be flushed more efficiently than if the data were written piece by piece at the time of the change. MySQL 5.5 extends this same functionality to include writes caused by deletes (an initial delete marking operation, followed later by a purge operation that collects/purges all the deleted records). The new Delete buffering and legacy Insert buffering features are now controlled using the new `innodb_change_buffering` configuration option, which has a default of `all`.

Improved Log Sys Mutex and Separate Flush List Mutex - Operations involving the buffer pool and the flush list previously were protected by a single buffer pool mutex, which could cause contention and unnecessary delays. In MySQL 5.5 the flush list has its own mutex, reducing contention with other buffer pool operations. Previously also the log mutex was held at this critical point. A new log mutex has now been introduced that is held during this operation that decreases the use of the central log mutex. This is the new default behavior, requiring no additional configurations to be set. With the multiple buffer pool improvement each buffer pool instance has a separate flush list mutex reducing contention even further.

Improved Purge Scheduling - The InnoDB purge operation is a type of garbage collection that runs periodically. In previous versions, the purge was part of the master thread, meaning that it could block other database operations when running. In MySQL 5.5 this operation can run in its own thread, allowing for more concurrency. Users can control whether the purge operation is split into its own thread with the `innodb_purge_threads` configuration option, which can be set to 0 (the default) or 1 (for a single separate purge thread).

Improved Metadata Locking Within Transactions - In previous MySQL versions when a transaction acquired a metadata lock for a table used within a statement, it released the lock at

the end of the statement. This approach had the disadvantage that if a data definition language ("DDL") statement occurred for a table that was being used by another session in an active transaction, statements could be written to the binary log in the wrong order. MySQL 5.5 ensures transaction serialization by not permitting one session to perform a DDL statement on a table that is used in an incomplete transaction in another session. This is achieved by acquiring metadata locks on tables used within a transaction and deferring release of those locks until the transaction ends. This metadata locking approach has the implication that a table that is being used by a transaction within one session cannot be used in DDL statements by other sessions until the transaction ends. For example, if a table `t1` is in use by a transaction, another session that attempts to execute `DROP TABLE t1` will block until the transaction ends.

These changes, along with optimizations made to how MySQL internally manages table locking (LOCK_open) improve performance for OLTP applications, specifically those that require frequent DDL activity. The improvements of LOCK_open also remove a number of potential stalls of the MySQL Server while performing DROP TABLE and other metadata operations.

See the Appendix section of the report for a performance comparison of InnoDB with MySQL 5.5 to the previous InnoDB plug-in provided for MySQL 5.1

You can read more about the new MySQL 5.5 and InnoDB performance and scalability enhancements, including how to enable and implement them, here:
http://dev.mysql.com/doc/refman/5.5/en/mysql-nutshell.html
http://dev.mysql.com/doc/innodb-plugin/1.1/en/index.html

Benchmark Overview

The open source Sysbench benchmark was used to compare the performance of MySQL 5.5 with the InnoDB 1.1 and MyISAM storage engines.

It is important to emphasize that performance is highly application-dependent. Users who are considering migrating existing applications from the MyISAM storage engine to InnoDB should benchmark their applications to identify any dependencies on MyISAM, and opportunities to tune the InnoDB configuration parameters to better meet performance expectations. See the "Comparing InnoDB & MyISAM" section later in the paper for more detail.

Sysbench

SysBench is a modular, cross-platform and multi-threaded benchmark tool for measuring OLTP (On-Line Transaction Processing) database performance under intensive load. It is especially useful for identifying those Operating Systems parameters needed to deliver the highest levels of performance, including:

- file I/O performance
- scheduler performance
- memory allocation and transfer speed
- POSIX threads implementation performance

Sysbench has been used in two variants:

- Sysbench RW (Read Write) test
- Sysbench RO (Read Only) test.

Both tests use a set of 10 primary key lookups together with a few variants of queries performing index scans. Sysbench RW also adds 3 update queries, 1 delete query and 1 insert query. All operations are performed on one table.

This benchmark is synthetic but it does a very good job in revealing any scalability bottlenecks by issuing more than 200.000 queries per second across a database with 1 million rows.

Users can download a copy of Sysbench for their own use from the following site:
http://sourceforge.net/projects/sysbench/

Benchmark Platform

Database Version:
MySQL 5.5.7-rc

Operating System:
Oracle Linux 5 with the Unbreakable Enterprise Kernel 2.6.32

Server Platform:
- 4 Sockets, 48 cores total, 4 x 12-core AMD Opteron 6172 “Magny-Cours” 2.1GHz CPUs. (Note: 36 cores were allocated to MySQL and the remaining 12 the Sysbench processes).
- 64 GB DDR3 RAM
- 2 x Intel X25E SSD drives

InnoDB and MyISAM Benchmark Results

The following section of the whitepaper provides an analysis of the benchmark results generated by MySQL 5.5 when configured with both the InnoDB and MyISAM storage engines.

Note that the ACID properties of InnoDB were relaxed when running the performance comparisons. The log buffer is configured to write out updates to the log file at each commit, but only flushes to disk at set time intervals. The following statement is used to configure this behavior.

```
innodb-flush-log-at-trx-commit=2
```

Using the value above, the log file is flushed once per second, essentially batching writes to disk, rather than committing transactions individually. The result is higher throughput, especially for update-intensive and I/O bound workloads.

When using a value of “2”, as above, only an operating system crash or a power outage can erase the last second of transactions. However, InnoDB's crash recovery is not affected and continues to work regardless of the value.

With a value of “0” used in the statement above, a mysqld process crash can erase the last second of transactions, though again InnoDB’s crash recovery is unaffected.

The default value of “1” is required in the statement above for full ACID-compliance.

You can learn more about InnoDB start-up options and configuration variables here:
http://dev.mysql.com/doc/refman/5.5/en/innodb-parameters.html

Sysbench Results

Both Read / Write and Read-Only Sysbench tests were run to compare the performance and scalability of InnoDB and MyISAM, with Transactions Per Second (TPS) recorded for each

storage engine as more cores were enabled in the host server. Initial runs were made with 6-cores, and then repeated as the server was scaled with 12, 18, 24, 30 and then 36-cores. In all cases, the number of concurrent database connections was set at 64.

As the data below demonstrates, InnoDB delivers very good scalability up to 30-cores. From 30 to 36-cores, performance continues to increase, though scalability is reduced.

MyISAM demonstrates almost zero scalability from 6 to 36 cores, with performance significantly lower than InnoDB.

Read-Write Results
As the graph below shows, InnoDB delivered 35x higher throughput than MyISAM, while achieving 85% - 90% scalability from 6 to 36-cores. Above 30-cores, the scalability curve starts to flatten out as the number of hot mutexes grow, but performance does still continue to increase.

Clearly table locking in MyISAM reduces throughput for Read / Write workloads.

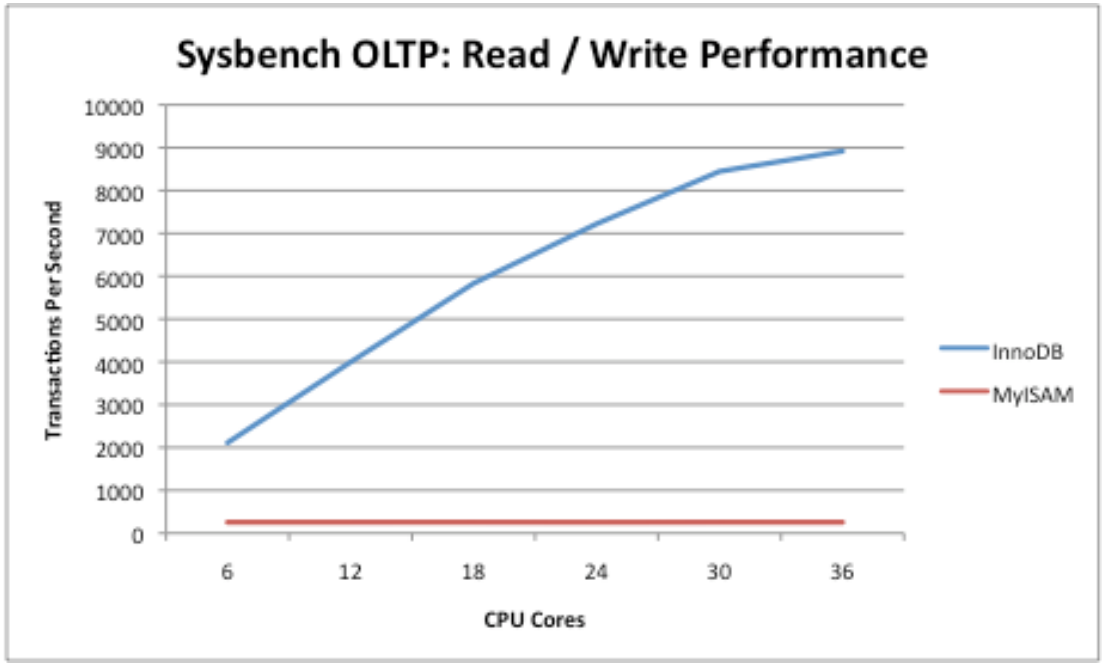

Figure 1: ***InnoDB Delivers 35x Higher Performance than MyISAM with up to 90% Scalability***

Read-Only Results
InnoDB delivered 4.6x higher throughput than MyISAM, while achieving 90% - 95% scalability from 6 to 36-cores. Above 30-cores, scalability flattens out as the server is again saturated by a number of hot mutexes.

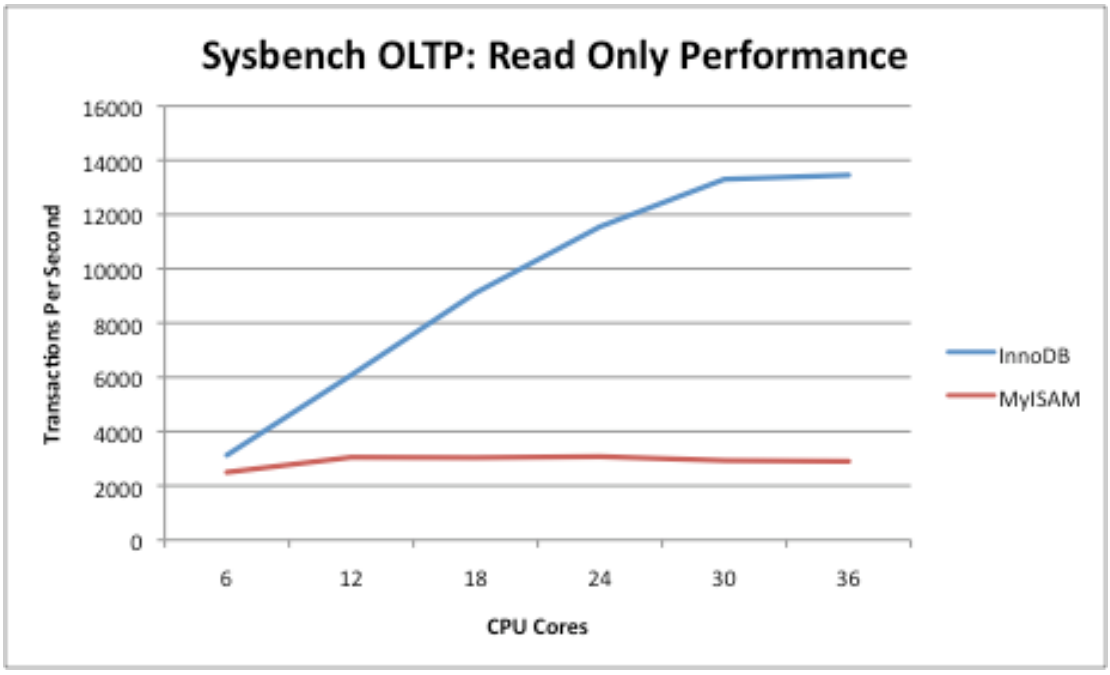

Figure 2: ***InnoDB Delivers 4.6x Higher Performance than MyISAM with up to 95% Scalability***

Comparing InnoDB & MyISAM

A major advantage of MySQL is its flexible pluggable storage engine architecture. Both InnoDB and MyISAM are examples of pluggable storage engines that can be selected for each individual table managed by the database. The use of the storage engine is transparent to both the application and the user.

Each storage engine has its own specific characteristics and benefits. Beyond performance on OLTP workloads, there are many other factors that influence selection of the storage engine. The table below highlights the key feature differences between InnoDB and MyISAM.

Feature	InnoDB	MyISAM
ACID Transactions	Yes	No
Configurable ACID Properties	Yes	No
Crash Safe	Yes	No
Foreign Key Support	Yes	No
Row-Level Locking Granularity	Yes	No (Table)
MVCC	Yes	No
Geospatial Data Type Support	Yes	Yes
Geospatial (R-Tree) Indexing Support	No	Yes
B-tree Indexes	Yes	Yes
Full-text search Indexes	No	Yes
Clustered Index	Yes	No
Data Caches	Yes	No
Index Caches	Yes	Yes
Compressed Data	Yes [b]	Yes [a]
Read & Write to Compressed Tables	Yes	No (Read-Only)
Encrypted Data [c]	Yes	Yes
Replication Support [d]	Yes	Yes
Backup / Point-in-Time Recovery [d]	Yes	Yes
Query Cache Support	Yes	Yes
Update Statistics for Data Dictionary	Yes	Yes
Storage Limits (Table Size)	64TB	256TB

[a] Compressed MyISAM tables are supported only when using the compressed row format.
[b] Compressed InnoDB tables require the InnoDB Barracuda file format.
[c] Implemented in the server (via encryption functions), rather than in the storage engine.
[d] Implemented in the server, rather than in the storage engine.

Figure 3: *InnoDB to MyISAM Feature Comparison*

As both the table above and the benchmark results demonstrate, InnoDB delivers the best blend of performance, reliability and data integrity for transactional workloads, with a host of parameters that are configurable, enabling users to optimize performance and behavior for their specific workload.[1]

MyISAM remains a strategic technology for MySQL and its users, and there are some use cases that are better suited to the characteristics of MyISAM. Full-text search indexes in MyISAM are useful for many simple read-only web applications, though often users deploy MySQL and InnoDB with Sphinx or Lucene for fast text searches as an alternative to MyISAM. It is often the case that performing full text searches outside of the database will deliver higher levels of performance and scalability.

Other use cases that are potentially suitable for MyISAM include:

- Applications demanding very high raw INSERT speeds where concurrency is not a consideration. Performance will always be application dependent, so benchmarking is necessary to determine the best solution for your own environment.
- Caches or temporary tables.
- Blogs / Wikis / RSS feeds.
- Read-only tables.

[1] Configuration parameters: http://dev.mysql.com/doc/refman/5.5/en/innodb-parameters.html

It is also very easy to create read-only slaves without using MySQL replication by sharing MyISAM tables via NFS. This is not possible with InnoDB.

Considerations when Migrating from MyISAM to InnoDB

In many cases, migration can be accomplished by simply running ALTER TABLE as InnoDB is a drop-in replacement for MyISAM, but it is still important to be aware of the considerations discussed below. These are not exhaustive, but are the most common factors found among MySQL users who have migrated their tables.

Migration Process
The first consideration is the migration process itself. As mentioned above, ALTER TABLE is simple, but it will lock the table during operation. One common practice for high volume production environments is to use MySQL Replication. In this instance, changes are made to tables on the slave server only, which can then be tested before being promoted to become the new master. As a result, any disruption to the application and clients is avoided. Another option is to dump the table and then reload it.

Tuning and Functional Considerations
As part of any migration, it is important to thoroughly test your applications to ensure functionality is maintained and performance expectations are achieved. As discussed earlier, performance is always application-dependent so it is highly recommended that users benchmark their own workloads, especially when considering migrating an existing application from MyISAM to InnoDB.

Tuning the Key Buffer is one of the very few performance optimizations available for MyISAM. InnoDB is much more configurable, which can initially present more complexity, but is also much more flexible, allowing InnoDB to effectively serve a much broader range of workloads than MyISAM.

Typically users will reduce the key buffer size when moving from MyISAM to InnoDB, and will pay special attention to the innodb_buffer_pool_size parameter[2]. Unlike MyISAM, InnoDB stores both indexes and pages in memory, which provides lots of scope for performance optimization.

From a functional perspective, your application may have dependencies on the table locks that are implemented by MyISAM and need to be identified and addressed before migration can be completed. Also, due to transactional support, InnoDB also has its own locking characteristics that should be tested for compatibility as some applications may expect table locks for certain update or delete operations.

A common operation in MyISAM that can result in performance issues when migrating to InnoDB is using the COUNT(*) operation without a WHERE clause. The code should be modified to include a WHERE clause as this will avoid a full table scan.

Other functions to consider include automatically incrementing the primary keys of multiple columns and merging tables, neither of which are supported by InnoDB.

The way in which tables are updated can also be impacted when you move from MyISAM to InnoDB. To minimize the impact of table locks, many MyISAM users divide updates into smaller operations. When moving to InnoDB, it is worth considering the batching of updates so that the affects of transactional commits are divided across multiple operations, thereby reducing overhead for each individual transaction.

File Sizes and Storage Requirements
File sizes under the InnoDB storage engine have often been larger than those generated for MyISAM, however as InnoDB can compress both data and indexes while MyISAM compresses only the indexes, this has become much less of a restraint.

[2] http://dev.mysql.com/doc/refman/5.5/en/innodb-parameters.html#sysvar_innodb_buffer_pool_size

InnoDB tables can consume more space than MyISAM as indexes contain the primary key, which is then copied to secondary indexes. To save space and increase performance, some users convert multi-column primary keys to auto-increment fields.

The transactional capability of InnoDB also increases the storage requirement as it needs to maintain undo spaces, row versions and transactional details.

On-disk fragmentation of files can also be an issue with both storage engines, but can be addressed by running the OPIMIZE TABLE command. Typically this command needs to be run less frequently with InnoDB than MyISAM.

Hardware Considerations
With the considerations above, it is important to ensure your hardware platforms are equipped with sufficient RAM and disk capacity to accommodate potentially larger file sizes. To leverage the durability and crash recovery of InnoDB, it is also important to ensure that your server platforms are equipped with the appropriate reliability mechanisms, including battery-backed caches on their RAID cards, and to use a journaling file system.

Operational Processes
Some operations tasks such as backing-up tables are different between MyISAM and InnoDB. A common utility is mysqldump, but MyISAM requires table locks whereas InnoDB does not, so the user needs to ensure they run mysqldump configured with the appropriate options[3]. Non-blocking, hot back-ups can be performed with LVM or SAN snapshots, which are also much faster than mysqldump.

MySQL Enterprise Back-Up, part of the commercial MySQL Enterprise Edition, can be used to backup both InnoDB and MyISAM tables. Hot back-up, non-blocking operations are supported for InnoDB tables by MySQL Enterprise Back-up. You can learn more at:
http://www.mysql.com/products/enterprise/backup.html

A further consideration is that MyISAM can complete LOAD FROM FILE operations faster than InnoDB. In such cases, it can make sense to create tables under the MyISAM engine first, and then alter the table to InnoDB once loaded.

Additional Resources
You can learn more about MyISAM table restrictions from the documentation here:
http://dev.mysql.com/doc/refman/5.5/en/myisam-table-problems.html

It is also worth reviewing InnoDB table limits documented here:
http://dev.mysql.com/doc/refman/5.5/en/innodb-restrictions.html

Best practices for configuring InnoDB are documented here:
http://dev.mysql.com/doc/refman/5.5/en/innodb-configuration.html

The MySQL Consulting division at Oracle can provide assistance in migration and performance projects, ensuring costs and risks are reduced and allowing you to quickly take advantages of the benefits of InnoDB.

[3] http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html

Oracle Linux

Oracle Linux with the Unbreakable Enterprise Kernel was used as the operating system for the Sysbench benchmark used in these tests.

Oracle Linux[4] is an enterprise-class Linux distribution supported by Oracle. Available under the GPL license, Oracle Linux is fully source and binary compatible with Red Hat Enterprise Linux.

Oracle Linux is certified for compliance with the Linux Standard Base (LSB), which increases the compatibility between individual Linux distributions, reducing the costs and effort involved with porting and then supporting applications across different distributions.

The Oracle Unbreakable Enterprise Kernel[5] is based on the stable 2.6.32 Linux kernel, delivering a fast, modern, reliable platform that is optimized for Oracle software and hardware. Oracle Linux combined with the Unbreakable Enterprise Kernel brings the latest Linux innovations to market delivering extreme performance, advanced scalability and reliability for enterprise applications.

The Oracle Unbreakable Linux support program[6] delivers enterprise-class support for Oracle Linux and the Oracle Unbreakable Enterprise Kernel with backports, management packs, indemnification, testing and more.

Conclusion

InnoDB is now the default storage engine for the MySQL database, replacing MyISAM. As a result, MySQL handles transactional applications that require crash recovery, referential integrity, high levels of performance and scalability with fast response times, straight out of the box.

MyISAM remains an important storage engine for specific read-intensive workloads. But as these benchmark results demonstrate, users can configure InnoDB to relax specific ACID constraints, while gaining much higher levels of protection and functionality, all with higher performance and scalability than MyISAM.

For new applications built around MySQL 5.5, it is recommended to use the default InnoDB storage engine. For existing applications, evaluate migrating to InnoDB to gain the benefits of additional database features, performance and scalability.

Resources

Whitepaper: An Introduction to MySQL 5.5:
http://www.mysql.com/why-mysql/white-papers/mysql-wp-whatsnew-mysql-55.php

Download MySQL 5.5:
http://dev.mysql.com/downloads/mysql/#downloads

MySQL 5.5 Documentation:
http://dev.mysql.com/doc/refman/5.5/en/

[4] http://www.oracle.com/us/technologies/linux/025994.htm
[5] http://www.oracle.com/us/technologies/linux/unbreakable-enterprise-kernel-ds-173416.pdf
[6] http://www.oracle.com/us/technologies/027614.pdf

Appendix A: Sysbench Benchmark

Configuration Parameters

Server Start-Up String
MYSQLD="";if test -f /data0/bench/mysql/mysql-5.5.7-rc/bin/mysqld ;
then MYSQLD="taskset 0xFFFFFFFFF000 /data0/bench/mysql/mysql-5.5.7-rc-tp6/bin/mysqld";
MYSQLD_SUBDIR="/data0/bench/mysql/mysql-5.5.7-rc-tp6/bin";fi;if test -f
/data0/bench/mysql/mysql-5.5.7-rc-tp6/sbin/mysqld ; then MYSQLD="taskset 0xFFFFFFFFF000
/data0/bench/mysql/mysql-5.5.7-rc-tp6/sbin/mysqld";
MYSQLD_SUBDIR="/data0/bench/mysql/mysql-5.5.7-rc-tp6/sbin";fi;if test -f
/data0/bench/mysql/mysql-5.5.7-rc-tp6/libexec/mysqld ; then MYSQLD="taskset 0xFFFFFFFFF000
/data0/bench/mysql/mysql-5.5.7-rc-tp6/libexec/mysqld";
MYSQLD_SUBDIR="/data0/bench/mysql/mysql-5.5.7-rc-tp6/libexec";fi;if test "x${MYSQLD}" = "x" ;
then echo "No mysqld binary in path"; exit 1;fi; echo "Using binary from ${MYSQLD}"; LANG=""; if
test -d /data0/bench/mysql/mysql-5.5.7-rc-tp6/share/mysql ; then LANG="/data0/bench/mysql/mysql-
5.5.7-rc-tp6/share/mysql"; else LANG="/data0/bench/mysql/mysql-5.5.7-rc-tp6/share"; fi; LANG="--
lc-messages-dir=$LANG"; ulimit -c unlimited; export LD_LIBRARY_PATH=/usr/local/lib ;export
LD_PRELOAD=libjemalloc.so.1 ; $MYSQLD --no-defaults

InnoDB
--innodb_purge_threads=1
--innodb_file_format=barracuda
--innodb-buffer-pool-size=8192M
--innodb_support_xa=FALSE
--innodb_flush_method=O_DIRECT
--innodb-flush-log-at-trx-commit=2
--innodb-log-file-size=2000M
--innodb-log-buffer-size=64M
--innodb-io-capacity=200
--skip-innodb-adaptive-hash-index
--innodb-read-io-threads=8
--innodb-write-io-threads=8
--innodb_change_buffering=all
--innodb_stats_on_metadata=off
--innodb-buffer-pool-instances=12
--skip-grant-tables[7]
--max_tmp_tables=100
--query_cache_size=0
--query_cache_type=0
--max_connections=1000
--max_prepared_stmt_count=1048576
--sort_buffer_size=32768

[7] Note, for security of production workloads, it is recommended to not skip GRANT tables. Performance is unaffected by GRANT tables as these are only used when connecting to the database, and not when running queries.

Appendix B: Comparing InnoDB 1.0 and 1.1 Performance

Much of the engineering work that has gone into MySQL 5.5 and re-architecting InnoDB has been focused on providing users with "transparent" performance and scalability gains across heterogeneous platforms specifically on multi-CPU/core, hyper-threaded architectures. For quick reference, below are Oracle's internal SysBench benchmark tests comparing the performance of MySQL 5.5 with version 5.1 the on Linux platform. These benchmarks include instances of 5.1 configured with the optional InnoDB 1.0 plug-in and with the default InnoDB built-in.

MySQL 5.5 SysBench Benchmarks on Linux

For context, the SysBench benchmarks for Linux were performed on the following server configuration:

Intel Xeon x7460, x86_64
4 CPU x 6 Cores/CPU
2,86 GHz, 32 GB RAM
Fedora 10

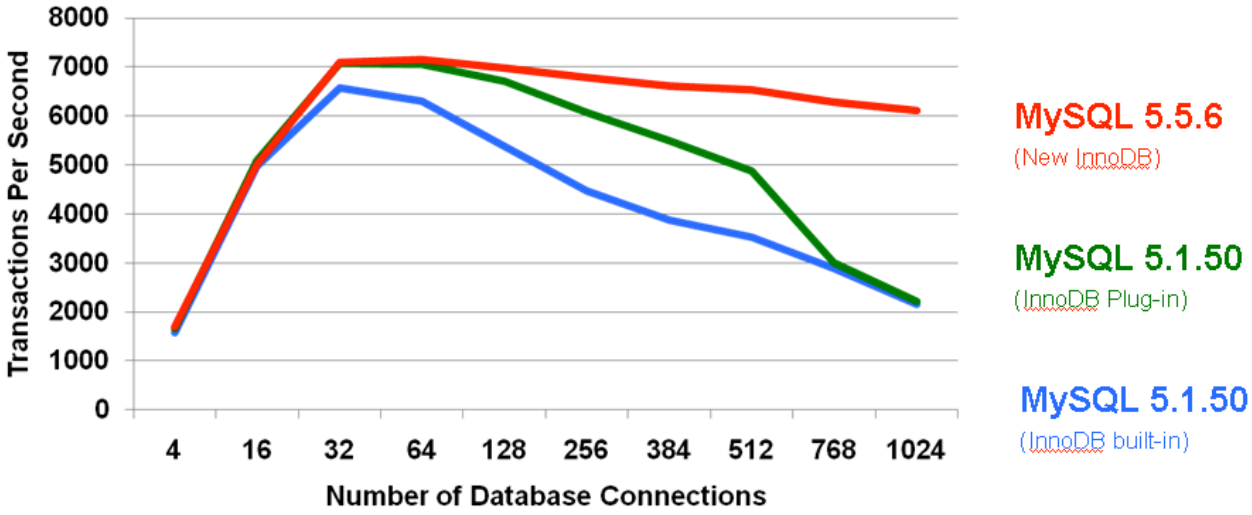

Figure 4: *SysBench Benchmarks for Linux – Read Only*

Figure 4 shows a 200% performance and scalability gain for MySQL 5.5 over 5.1 for Read Only activity on Linux at higher transaction and connection loads. New concurrency improvements allow MySQL 5.5 to sustain improved performance levels at higher transactions per second and user connection loads so applications remain responsive even when the physical server resources are saturated.

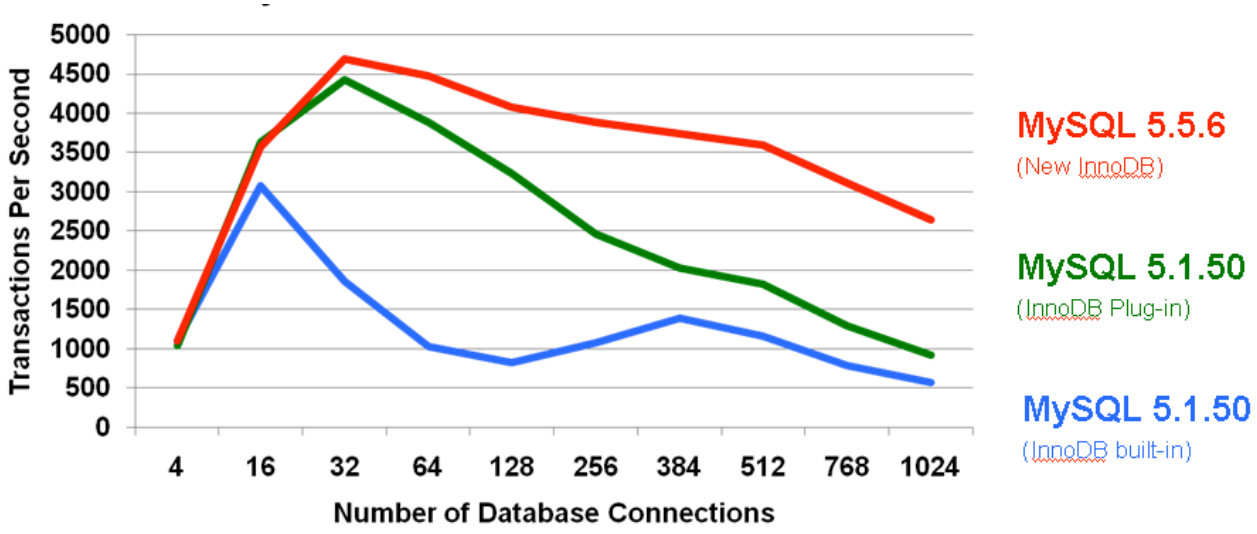

Figure 5: ***SysBench Benchmarks for Linux – Read/Write***

Figure 5 shows a 370% performance and scalability gain for MySQL 5.5 over 5.1 for Read/Write activity on Linux at higher transaction and connection loads. Again, although there is some drop-off at higher transaction and user loads, the new concurrency improvements allow MySQL 5.5 to sustain higher performance levels than previous versions of 5.1 at physical server saturation.