MySQL 5.4 Feature Summary

Abstract

This document contains the MySQL 5.4 Feature Summary. MySQL 5.4 is based on MySQL 5.1 but includes small, high-impact changes to enhance scalability and performance in MySQL Server.

Document generated on: 2009-06-03 (revision: 1324)

Table of Contents

1. MySQL 5.4 Introduction

The Sun MySQL Performance and Scalability Project aims at identifying and solving performance and scalability issues for MySQL Server in InnoDB, Falcon, and MySQL Cluster. Currently, project investigations focus primarily on InnoDB.

This document describes the MySQL 5.4 release developed by this project. MySQL 5.4 is based on MySQL 5.1 but includes small, high-impact changes to improve MySQL Server performance, particularly with regard to scalability. These changes exploit advances in hardware and CPU design and enable better utilization of existing hardware. MySQL 5.4 currently has Beta status.

The following list summarizes areas of emphasis within MySQL 5.4. Later sections provide additional information.

- The principal emphasis is to improve scalability on multi-core CPUs. The trend in hardware development now is toward more cores rather than continued increases in CPU clock speeds, which renders “wait until CPUs get faster” a non-viable means of improving database performance. Instead, it is necessary to make better use of multiple cores to maximally exploit the processing cycles they make available. MySQL 5.4 takes advantage of features of SMP systems and tries to eliminate bottlenecks in MySQL architecture that hinder full use of multiple cores. The focus has been on InnoDB, especially locking, memory management, and thread concurrency.
- InnoDB I/O subsystem changes enable more effective use of available I/O capacity.

- Several modifications improve operation of MySQL Server on Solaris.
- There is better access to execution and performance information. Diagnostic improvements include DTrace probes, expanded `SHOW ENGINE INNODB STATUS` output, and new status variables.
- The “out of box” configuration provides better choices of default option and system variable values for MySQL operation on up to 16-way x86 servers and 64-way CMT servers with 4GB or more memory.

MySQL 5.4 incorporates work from several sources, such as community patches and modifications developed by MySQL Scalability and Performance Project members and other MySQL engineers.

The following sections describe the systems for which MySQL 5.4 is available and characteristic features of MySQL 5.4 that distinguish it from MySQL 5.1.

2. Platform Availability and Installation/Upgrade Considerations

The initial release of MySQL 5.4 is 5.4.0.

Binary MySQL 5.4.0 distributions are available as 64-bit builds for Solaris 10-x86_64, Solaris10-SPARC, and Linux x86_64-glibc23.

If you use a source distribution, the `BUILD/build_mccge.sh` script can be used to configure and build MySQL 5.4.0. Run this script with the `--help` option to see the available options.

For installation instructions, see the MySQL 5.4 Reference Manual at http://dev.mysql.com/doc/.

If you upgrade to MySQL 5.4 from an existing MySQL 5.1 installation, an issue may occur due to `InnoDB` log file configuration changes. MySQL 5.4 increases the default value of `innodb_log_files_in_group` from 2 to 3. It also increases the default value of `innodb_log_file_size` from 5MB to 128MB and the minimum value of `innodb_log_file_size` from 1MB to 32MB.

For an upgrade, it is necessary to account for these changes. There are two ways to do so:

1. Configure MySQL 5.4 to use your current configuration values. *This can be done only if your current `InnoDB` log file size is at least as large as the new minimum size of 32MB.* For example, if you use the previous default number of files (2), configured to a size of 64MB each, put these lines in your server's `my.cnf` file before starting the 5.4 server:

   ```
   [mysqld]
   innodb_log_files_in_group=2
   innodb_log_file_size=64M
   ```

2. Discard the old `InnoDB` log files and let MySQL 5.4 create new ones. This can be done regardless of your current configuration values, but it is necessary to shut down your current server cleanly so that no outstanding transaction information remains in the log. The procedure to use depends on the value of `innodb_fast_shutdown`:

   - If `innodb_fast_shutdown` is not set to 2: Stop your current MySQL server and make sure that it shuts down without errors (to ensure that there is no information for outstanding transactions in the log). Copy the old log files into a safe place in case something went wrong during the shutdown and you need them to recover the tablespace. Delete the old log files and edit `my.cnf` if necessary to specify the desired log file configuration. Then upgrade to MySQL 5.4 and start the new server. `mysqld` sees that no `InnoDB` log files exist at startup and creates new ones.
   - If `innodb_fast_shutdown` is set to 2: Set `innodb_fast_shutdown` to 1:

     ```
     mysql> SET GLOBAL innodb_fast_shutdown = 1;
     ```

     Then follow the instructions in the previous item.

In MySQL 5.4, the `innodb_file_io_threads` system variable has been removed. If you upgrade to MySQL 5.4 from an existing MySQL 5.1 installation and explicitly set `innodb_file_io_threads` at server startup, you will need to change your configuration. `innodb_file_io_threads` system has been replaced with `innodb_read_io_threads` and `innodb_write_io_threads`, which can be used instead.

3. Scalability Improvements

MySQL 5.4 modifications improve performance on SMP systems to increase scalability on multi-core systems. The changes affect `InnoDB` locking, memory management, and thread concurrency. Much of this work is based on a patch developed by Google,

about which more information can be found here:

```
http://code.google.com/p/google-mysql-tools/wiki/SmpPerformance
```

MySQL 5.4 incorporates a Google SMP patch that improves the performance of RW-locks within `InnoDB`. The patch changes the implementation of the RW-locks to use atomic instructions rather than mutexes. This patch also enables `InnoDB` memory allocation to be disabled and replaced by the normal `malloc` library, or by a different library that implements `malloc` such as `tcmalloc` on Linux or `mtalloc` on Solaris.

MySQL 5.4 also implements a lock-free method for handling `InnoDB` thread concurrency. This method is timer-based; it uses yield and sleep rather than mutexes that have scalability issues. The `innodb_thread_concurrency_timer_based` system variable determines which concurrency method to use. By default, this variable is enabled (use the lock-free method). Disabling it causes `InnoDB` to use the original mutex-based method.

The number of threads in use affects whether the lock-free timer-based method will be advantageous. For 256 threads, this method increases performance about 15%. For low thread counts (up to 2 times the number of cores), performance decreases about 1%.

The reimplementation of RW-locks and the lock-free thread concurrency method require atomic instructions. A new status variable, `Innodb_have_sync_atomic`, shows whether atomic instructions are available. If `Innodb_have_sync_atomic` is `OFF`, enabling `innodb_thread_concurrency_timer_based` has no effect. The lock-free concurrency method also requires that `innodb_thread_concurrency` be set to a value greater than 0 (the default value in 5.4).

Another new status variable, `Innodb_heap_enabled`, indicates whether the built-in `InnoDB` memory manager is used. (`ON` means it is used, `OFF` means it is not.)

4. `InnoDB` I/O Subsystem Changes

MySQL 5.4 changes to the `InnoDB` I/O subsystem enable more effective use of available I/O capacity. The changes also provide more control over configuration of the I/O subsystem. Much of this work is based on patches developed by Google, about which more information can be found here:

```
http://code.google.com/p/google-mysql-tools/wiki/InnodbAsyncIo
http://code.google.com/p/google-mysql-tools/wiki/InnodbIoTuning
```

Background I/O Threads

`InnoDB` uses background threads to perform I/O for several kinds of activities, two of which are prefetching disk blocks and flushing dirty pages. Previously, `InnoDB` used only one thread each to perform these activities, but that can underutilize server capacity. MySQL 5.4 incorporates a Google patch that enables use of multiple background read and write threads, making it possible to read and write pages faster.

The patch makes the number of background I/O threads configurable via system variables: `innodb_read_io_threads` controls the number of threads to use for read prefetch requests. `innodb_write_io_threads` controls the number of threads to use for writing dirty pages from the buffer cache to disk. The default for both variables is 8.

The ability to increase the number of I/O threads can benefit systems that use multiple disks for `InnoDB`. However, the type of I/O being done should be considered. On systems that use buffered writes rather than direct writes, increasing the write thread count higher than 1 might yield little benefit because writes will be quick already.

The patch also introduces the `innodb_max_merged_io` system variable that specifies the maximum number of background I/O requests that will be merged to issue a larger I/O request in a more contiguous manner. The default value is 64.

Adjustable I/O Rate

Previously, the number of input/output operations per second (IOPS) that `InnoDB` will perform was a compile-time parameter. The rate was chosen to prevent background I/O from exhausting server capacity and the compiled-in value of 100 reflected an assumption that the server can perform 100 IOPS. However, many modern systems can exceed this, so the value is low and unnecessarily restricts I/O utilization.

MySQL 5.4 incorporates a Google patch that exposes this I/O rate parameter as a system variable, `innodb_io_capacity`. This variable can be set at server startup, which enables higher values to be selected for systems capable of higher I/O rates. Having a higher I/O rate can help the server handle a higher rate of row changes because it may be able to increase dirty-page flushing, deleted-row removal, and application of changes to the insert buffer. The default value of `innodb_io_capacity` is 200. In general, you can increase the value as a function of the number of drives used for `InnoDB` I/O.

The ability to raise the I/O limit should be especially beneficial on platforms that support many IOPS. For example, systems that use multiple disks or solid-state disks for `InnoDB` are likely to benefit from the ability to control this parameter.

The rate patch also implements an `innodb_extra_dirty_writes` system variable, which is used in conjunction with `innodb_max_dirty_pages_pct`. The latter variable specifies the percentage of pages that must be dirty before `InnoDB` flushes

them. If `innodb_extra_dirty_writes` is enabled (the default), flushing of dirty pages may occur when the server is idle, even if the percentage of dirty pages has not reached the normally required percentage. This can help reduce the amount of flushing that must be done when the server is not idle and has fewer resources to spare. When `innodb_extra_dirty_writes` is disabled, extra flushing does not occur.

5. Enhanced Solaris Support

MySQL 5.4 incorporates several modifications for improved operation of MySQL Server on Solaris:

- DTrace support for execution monitoring. See Section 6, “Diagnostic and Monitoring Capabilities”.
- Atomic instructions, which are needed for the improvements to RW-locking and the lock-free thread concurrency algorithm described in Section 3, “Scalability Improvements”. Atomic instructions now are supported for Sun Studio on SPARC and x86 platforms. This extends their previous availability (supported for `gcc` 4.1 and up on all platforms).
- The Google SMP patch described in Section 3, “Scalability Improvements”, was originally intended for x86 platforms. In MySQL 5.4 it also works on SPARC platforms. Also, Solaris optimizations for the patch have been implemented.
- Large page support is enhanced for recent SPARC platforms. Standard use of large pages in MySQL attempts to use the largest size supported, up to 4MB. Under Solaris, a “super large pages” feature enables uses of pages up to 256MB. This feature is enabled by default on Solaris. It can be enabled or disabled explicitly by using the `--super-large-pages` or `--skip-super-large-pages` option.
- Inline handling for `InnoDB` and processor instruction prefetching support, previously not enabled for builds created using Sun Studio, now are supported for that build environment.

6. Diagnostic and Monitoring Capabilities

MySQL 5.4 provides improved access to execution and performance information. Diagnostic improvements include Dtrace probes, expanded `SHOW ENGINE INNODB STATUS` output, and new status variables.

DTrace Support

MySQL 5.4 includes support for DTrace, integrated from MySQL 6.0. The DTrace probes work on Solaris, Mac OS X, and FreeBSD. For information on using DTrace in MySQL, see Tracing `mysqld` Using DTrace.

Enhanced `SHOW ENGINE INNODB STATUS` Output

The output from `SHOW ENGINE INNODB STATUS` includes more information. These changes are based on patches developed by Google, about which more information can be found here:

```
http://code.google.com/p/google-mysql-tools/wiki/InnodbStatus
http://code.google.com/p/google-mysql-tools/wiki/NewShowInnodbStatus
```

A description of revisions to statement output follows.

Output includes a `BACKGROUND THREAD` section:

```
----------
BACKGROUND THREAD
----------
srv_master_thread loops: 12 1_second, 12 sleeps, 0 10_second, 2 background,
  2 flush
srv_master_thread log flush: 12 sync, 15 async
srv_wait_thread_mics 0 microseconds, 0.0 seconds
spinlock delay for 6 delay 20 rounds is 15 mics
```

The `srv_master_thread` lines shows work done by the main background thread.

The `srv_wait_thread_mics` line indicates how many microseconds threads wait to use `InnoDB` due to `innodb_thread_concurrency` limits.

The `spinlock` line shows the number of microseconds the spinloop spins before sleeping (the time spent busy-waiting).

The `SEMAPHORES` section includes a line to show the number of spinlock rounds per OS wait for a mutex:

```
----------
SEMAPHORES
----------
...
Spin rounds per wait: 0.00 mutex, 20.00 RW-shared, 0.00 RW-excl
```

Lines in the `FILE I/O` section that show information for individual I/O threads display additional information to indicate the number of pages read and written, the number of system calls for those operations, and the time in milliseconds to complete them (total and average per request).

```
--------
FILE I/O
--------
I/O thread 0 state: waiting for i/o request (insert buffer thread)
  reads 0 writes 0 requests 0 io secs 0.000000 io msecs/request 0.000000
  max_io_wait 0.000000
I/O thread 1 state: waiting for i/o request (log thread)
  reads 0 writes 3 requests 3 io secs 0.006906 io msecs/request 2.302000
  max_io_wait 6.861000
I/O thread 2 state: waiting for i/o request (read thread)
  reads 33 writes 0 requests 2 io secs 0.005925 io msecs/request 2.962500
  max_io_wait 5.895000
...
Summary of background IO slot status: 0 issued, 0 done, 0 claimed, sleep set 0
```

The `Summary of background IO slot status` line describes the state of pending IO requests. `InnoDB` provides real or simulated asynchronous I/O to perform disk I/O for many uses. Requests are put into request arrays with a fixed number of slots per array. Other threads read the requests, remove them from the array and perform the I/O. The counts indicate the number of requests in the following states:

- `issued`: The request is in the array and the I/O operation is in progress.
- `done`: The I/O operation has completed, but the thread that requested the operation has not been notified.
- `claimed`: The I/O operation has yet to be started.

New Status Variables

The `Innodb_have_sync_atomic` and `Innodb_heap_enabled` status variables provide information about availability of MySQL 5.4 SMP features; see Section 3, “Scalability Improvements”.

`Innodb_wake_ups` indicates the number of wakeups that should not have occurred. These are wakeups in a condition on a mutex when the condition is not yet met so the thread must go back to sleep.

7. Configuration Differences in MySQL 5.4 from MySQL 5.1

Several variables and options are new or changed in MySQL 5.4 to provide better “out of box” configuration of default values for MySQL operation on up to 16-way x86 servers and 64-way CMT servers with 4GB or more memory.

> **Note**
>
> The changes to `innodb_log_file_size`, `innodb_log_files_in_group`, and `innodb_file_io_threads` may cause issues if you upgrade to MySQL 5.4 from an existing MySQL installation. See Section 2, “Platform Availability and Installation/Upgrade Considerations”.

The new system variables described here are all global and do not have session values. They are not dynamic (cannot be changed at runtime). To assign values different from the defaults, you must set them at server startup. Variables that have values of `ON` or `OFF` as displayed by `SHOW VARIABLES` should be enabled or disabled at server startup by assigning a value of 1 or 0.

New system variables:

- `innodb_extra_dirty_writes`: If enabled (the default), flushing of dirty buffer pages may occur when the server is idle, even if the percentage of dirty pages is less than the maximum dirty percent normally required for flushing (as indicated by `innodb_max_dirty_pages_pct`).
- `innodb_io_capacity`: The limit on the maximum number of I/O operations per second (IOPS) the server can perform. Default: 200.
- `innodb_max_merged_io`: The maximum number of background I/O requests that will be merged to issue a larger I/O request in a more contiguous manner. Default: 64.
- `innodb_read_io_threads`, `innodb_write_io_threads`: The number of background I/O threads to use for read prefetch requests and for writing dirty pages from the buffer cache to disk. Default: 8.
- `innodb_thread_concurrency_timer_based`: If enabled, use a lock-free timer-based method of handling thread concurrency. If disabled, the original mutex-based method is used. For the lock-free concurrency method to be used, two require-

ments must be satisfied:

- The `innodb_thread_concurrency` system variable must be set to a number greater than 0. The default value in MySQL 5.4 is 0, so you must change it to use the lock-free concurrency method.
- Atomic instructions must be available. That is, the `Innodb_have_sync_atomic` status variable must be `ON`.

If `innodb_thread_concurrency` is 0 or `Innodb_have_sync_atomic` is `OFF`, enabling `innodb_thread_concurrency_timer_based` has no effect.

Changes to existing system variables:

- `innodb_additional_mem_pool_size`: Default increased from 1MB to 8MB. Minimum increased from 512KB to 2MB.
- `innodb_autoextend_increment`: Default increased from 8MB to 64MB.
- `innodb_buffer_pool_size`: Default increased from 8 MB to 1GB. Minimum increased from 1MB to 64MB.
- `innodb_file_io_threads`: Removed (replaced by `innodb_read_io_threads` and `innodb_write_io_threads`).
- `innodb_log_buffer_size`: Default increased from 1MB to 16MB. Minimum increased from 256KB to 2MB.
- `innodb_log_file_size`: Default increased from 5MB to 128MB. Minimum increased from 1MB to 32MB.
- `innodb_log_files_in_group`: Default increased from 2 to 3.
- `innodb_max_dirty_pages_pct`: Default decreased from 90 to 75. Maximum decreased from 100 to 99 to never allow a completely dirty buffer pool.
- `innodb_thread_concurrency`: Default changed from 8 to 0. In effect, this changes concurrency from 8 to “infinite”.
- `table_definition_cache`: Default increased from 64 to 400.
- `table_open_cache`: Default increased from 64 to 400.

New status variables:

- `Innodb_have_sync_atomic`: Indicates whether atomic instructions are available.
- `Innodb_heap_enabled`: Indicates whether the built-in `InnoDB` memory manager is used. (`ON` means it is used, `OFF` means it is not.)
- `Innodb_wake_ups`: The number of wakeups that should not have occurred. These are wakeups in a condition on a mutex when the condition is not yet met so the thread must go back to sleep.

New option:

- `--super-large-pages`: Boolean option. Large page support is enhanced for recent SPARC platforms. Standard use of large pages in MySQL attempts to use the largest size supported, up to 4MB. Under Solaris, a “super large pages” feature enables uses of pages up to 256MB. This feature is enabled by default on Solaris. It can be enabled or disabled explicitly by using the `--super-large-pages` or `--skip-super-large-pages` option.

Changes to existing options:

- `--large-pages`: Enabled by default.

8. Acknowledgments

We wish to recognize Google Inc. for contributions to the MySQL distribution: Mark Callaghan's SMP Performance patches and other patches.