MySQL 5.7
InnoDB - What's new

赖铮 – zheng.lai@oracle.com
Principle Software Developer

ORACLE

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

- Performance
- Features
- Download & Blogs

Performance : Transactions

Transaction pool

Fixed chunks of 4MB each

Ordered on address, improves locality of reference

Improves performance of read-write transaction list scans

Reduces malloc()/free() overhead

Performance : Transactions

Transaction life cycle improvements

All transactions are considered as read-only by default

Read only transaction start/commit mutex free

No application changes required

Read views are cached

Read view recreated if a RW transaction started since the last snapshot

Reduce contention when implicit → explicit row lock conversion is done

Performance : Transactions

Transaction life cycle improvements

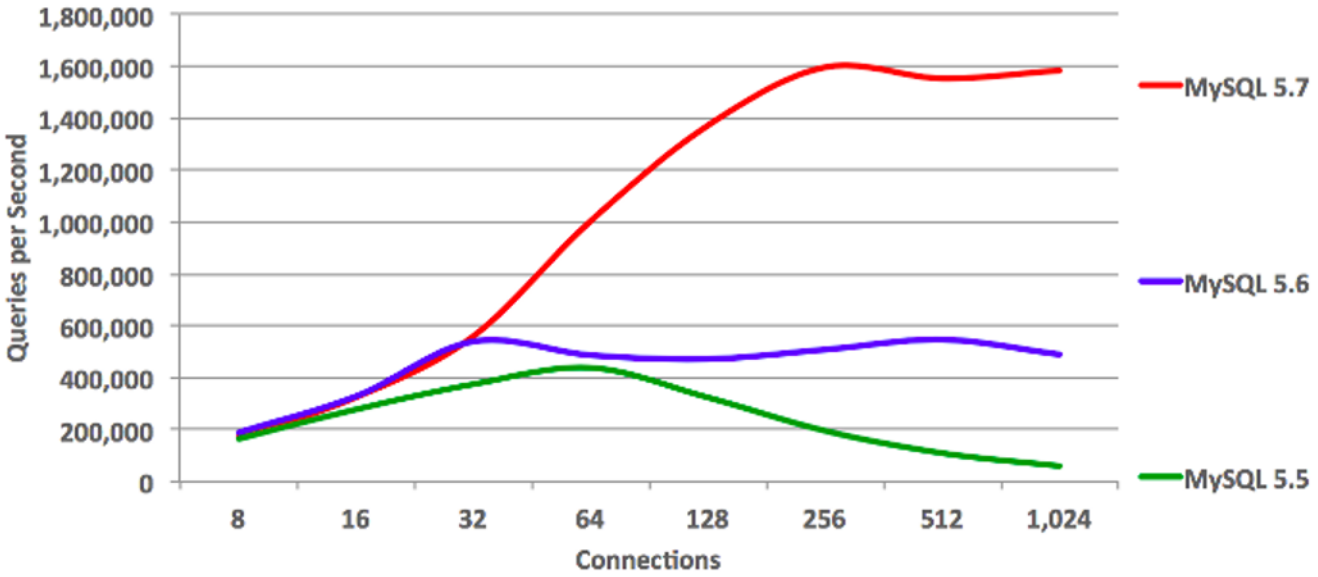
High priority transactions (Replication GCS)

Can jump the record lock queue – prioritized

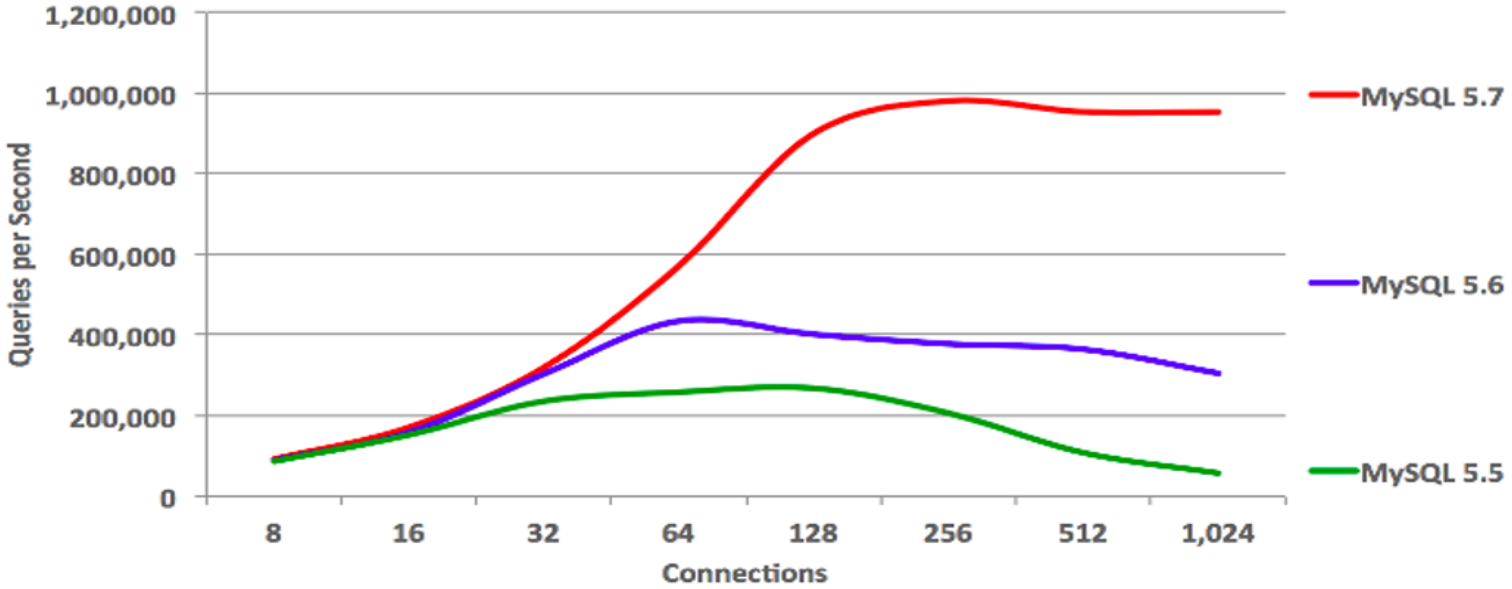
Can kill other lower priority transactions, if required

Currently not visible to end users

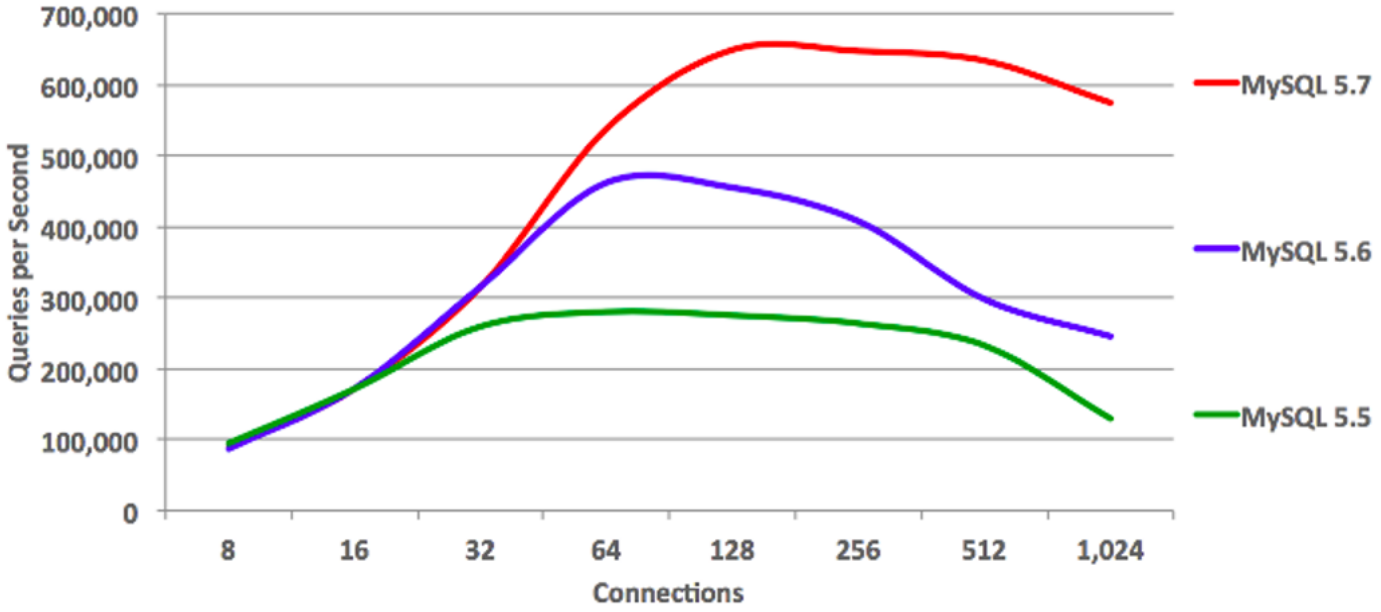
Performance : Sysbench Point Selects



Performance : Sysbench OLTP Read-Only



Performance : Sysbench OLTP Read-Write



ORACLE

Performance : Temporary Table Optimizations

DDL changes

Not stored in the data dictionary – lower mutex contention

Special shared temporary tablespace – lower IO overhead

Compressed tables done the old way, separate .ibd file

Tablespace recreated on start up

Performance : Temporary Table Optimizations

DML changes

Special UNDO logs that are not redo logged

Undo logging required for rollback to savepoint

Changes to the temporary tablespace are not redo logged

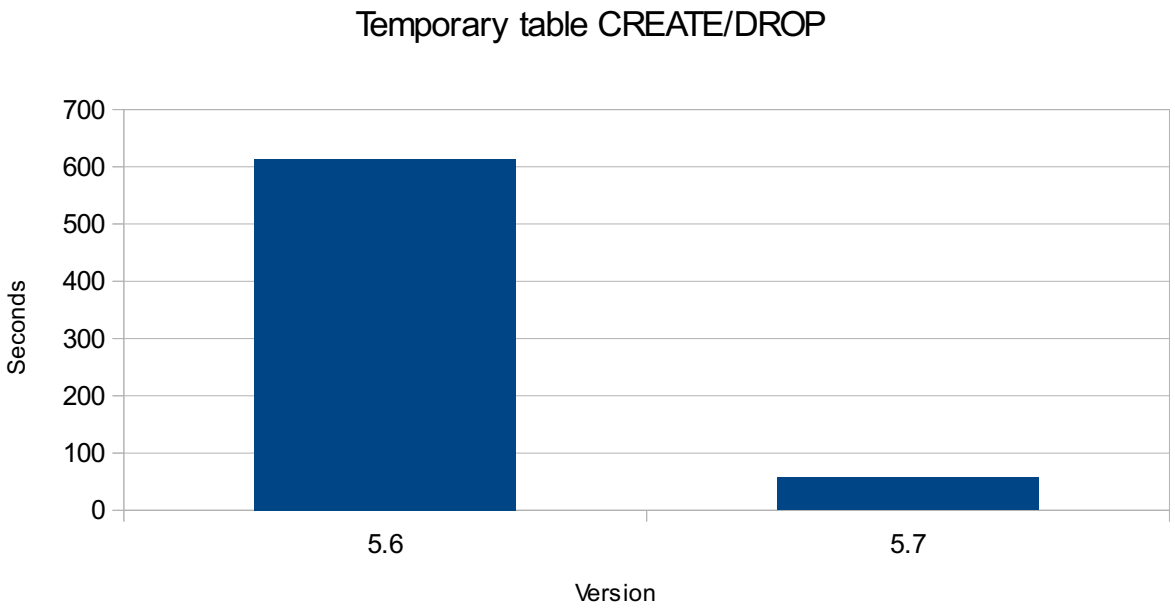
No fsyncs() on the temporary tablespace

Configuration variables

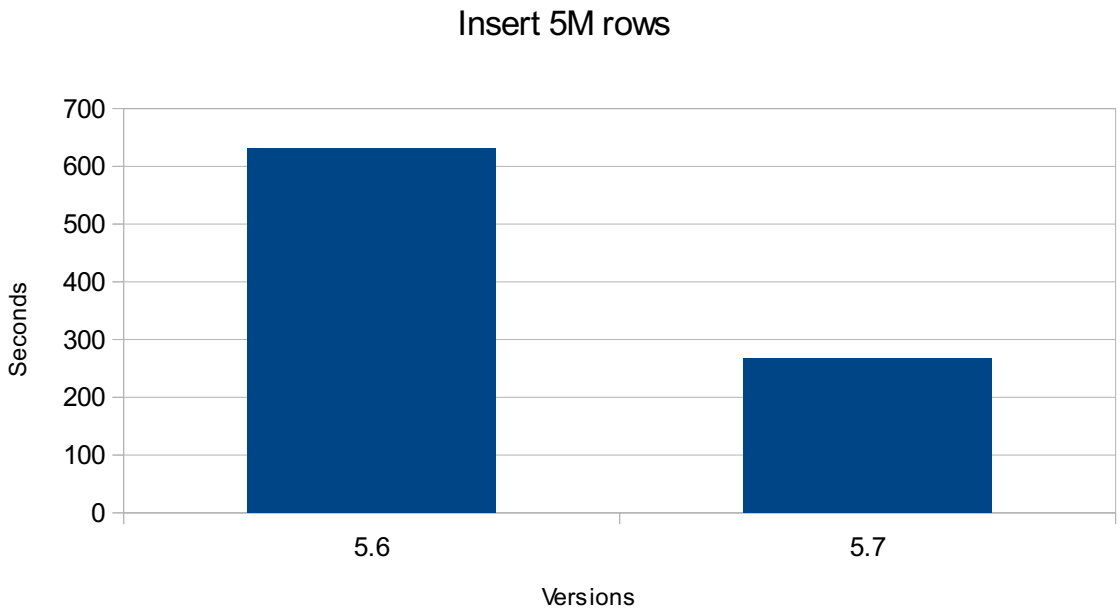
--innodb-temp-data-file-path := same format as the system tablespace

e.g., ibtmp1:12M:autoextend – default setting

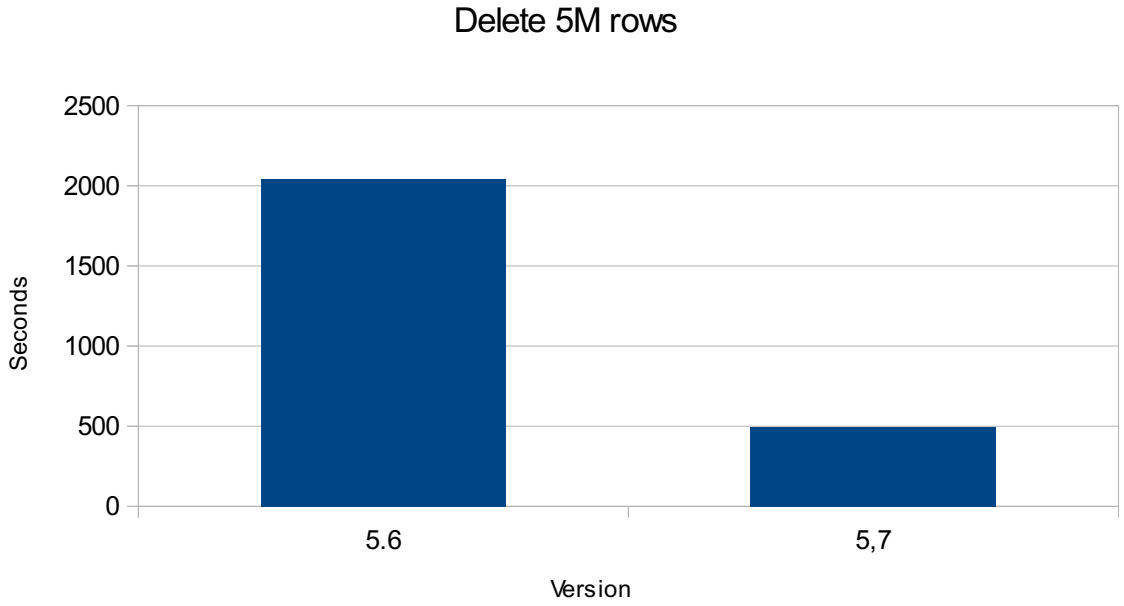
Performance : Temporary Tables Benchmarks



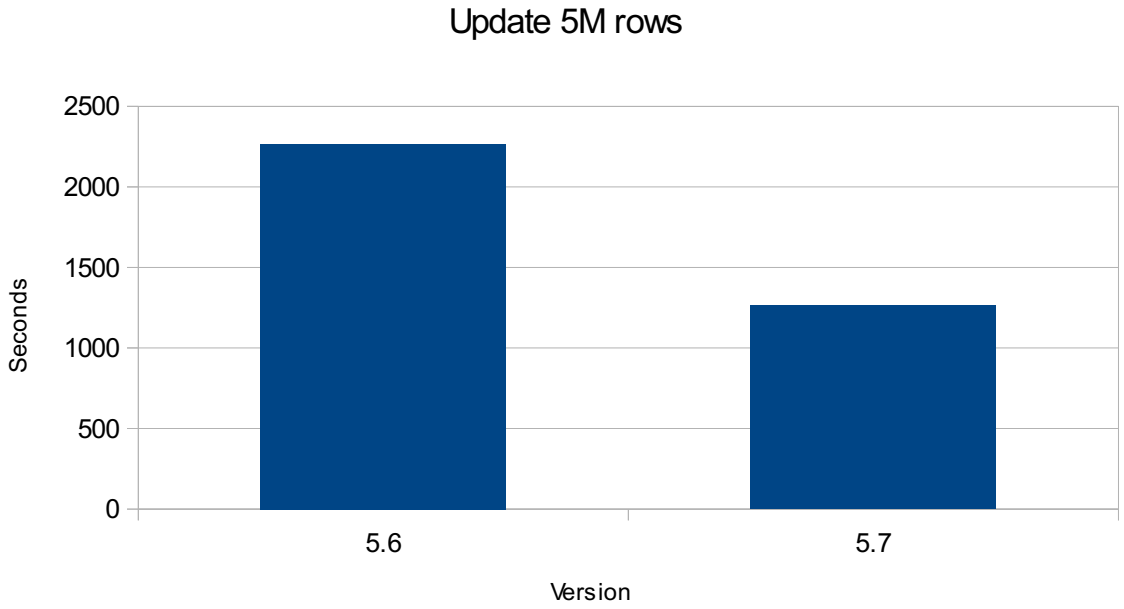
Performance : Temporary Tables Benchmarks



Performance : Temporary Tables Benchmarks



Performance : Temporary Tables Benchmarks



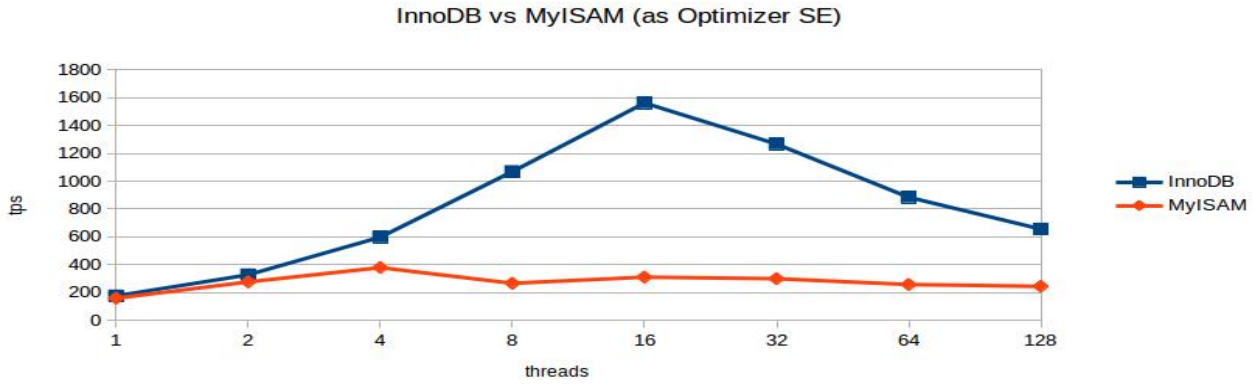
Performance : InnoDB “intrinsic” tables

Used by the optimizer

Previously used MyISAM

Better performance at high concurrency

--internal-tmp-disk-storage-engine := InnoDB | MyISAM



Performance : Buffer pool improvements

Use atomics for page reference counting

Bug#68079 - INNODB DOES NOT SCALE WELL ON 12 CORE

Fixed in 5.6 too

Faster flush list traversal

Improve flush and LRU list rescanning

Previously after flushing a page we would restart rescan from the tail

Performance : Buffer pool improvements

Multithreaded flushing

5.6 introduced a separate thread for flushing

5.7 allows multiple threads

--innodb-page-cleaners := 1..64 – default is 1

Performance : Redo log

Improved IO

Fix read on write issue – pad the log buffer before writing to disk

Optimize mutex acquire/release during log checkpoint

Improved checksum – Patch from Percona

Add version meta-data CRC-32C the only checksum on the InnoDB redo log pages --innodb-log-checksums := ON (the default)

Performance : Memcached Plugin

Leverages the read-only transaction optimizations

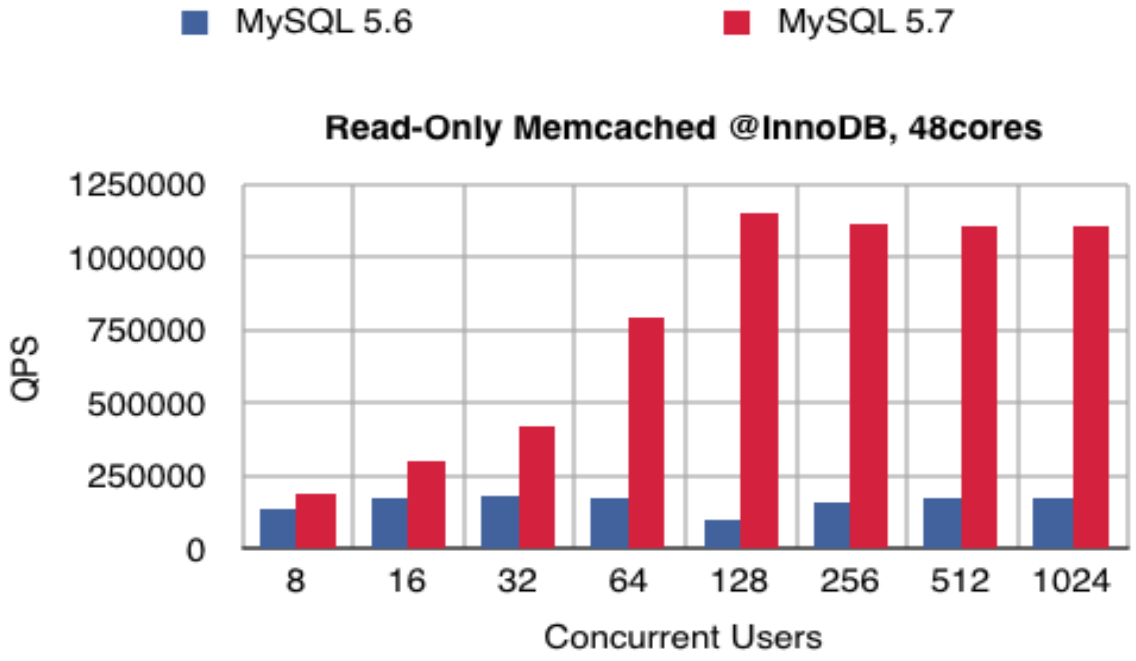
Fixed several bottlenecks in the Memcached and the plugin code

1.1 Million GET/s

Limiting factors were:

- The network
- Memcached client

Performance : Memcached Benchmarks



Performance : index->lock

Better concurrency, improved performance

Very complex fix

Previously entire index X latched during tree structure modification

B-tree internal nodes not latched before fix

New SX lock mode – compatible with S lock mode

Increases concurrency e.g., index->lock(SX), reads can proceed

Performance : DDL & Truncate

Truncate table is now atomic

Previously DROP + CREATE

ID mismatch or .ibd missing If crash after DROP but before CREATE

More schema-only ALTER TABLE supported

Rename index

VARCHAR extension

Performance : Faster DDL

Scan rows → Sort → Build table/index

Speed up the the build table/index phase only

Build phase in 5.6 it takes 2484s vs 440s in 5.7 - 1 billion rows, approx. 40G

Total time improvement ~170%

Previously build was done by doing an insert row by row

Build the index bottom up

--innodb-fill-factor := 10..100

Performance : Adaptive Hash Index (AHI)

Split the AHI

Bottleneck in read write loads

Faster drop of entries – when page is evicted from the buffer pool

--innodb-adaptive-hash-index-parts := 1..512 – default 8

Features : Partitions

Native Partitioning

Reduced memory overhead

Native partitioning is the default for InnoDB

mysql_upgrade will support metadata upgrade (no data copied)

Will allow us to add

- Foreign key support
- Full text index support

Makes it easier to plan for a parallel query infra-structure

Features : Partitions

Native Partitioning memory overhead improvement

Example Table with 8K partitions

```
CREATE TABLE `t1` (
`a` int(10) unsigned NOT NULL AUTO_INCREMENT,`b` varchar(1024) DEFAULT NULL, PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 PARTITION BY HASH (a) PARTITIONS 8192;
```

Memory overhead comparison

One open instance uses 49 % less memory (111 MB vs 218 MB)

Ten open instances take 90 % less memory (113 MB vs 1166 MB)

Features : Partitions

Import/Export support

Importing a single partition

```
# If  the table doesn't already exist, create it
mysql> CREATE TABLE partitioned_table <same as the source>;
# Discard the tablespaces for the partitions to be restored
mysql> ALTER TABLE partitioned_table DISCARD PARTITION p1,p4 TABLESPACE;
# Copy the tablespace files
$ cp /path/to/backup/db-name/partitioned_table#P#p{1,4}.{ibd,cfg} /path/to/mysql-datadir/db-name/
# Import the tablespaces
mysql> ALTER TABLE partitioned_table IMPORT PARTITION p1,p4 TABLESPACE;
```

Features : Partitions

DML

Index condition push down (ICP) – better query processing

Limited HANDLER support for partitions

```
CREATE TABLE t (a int, b int, KEY (a, b)) PARTITION BY HASH (b) PARTITIONS 2;
HANDLER t READ a = (1, 2);
HANDLER t READ a NEXT;
```

Features : Tablespace management

General Tablespaces

SQL syntax for explicit tablespace management

Replaces legacy –innodb-file-per-table usage

```
CREATE TABLESPACE Logs ADD DATAFILE 'log01.ibd';
CREATE TABLE http_req(c1 varchar) TABLESPACE=Logs ;
ALTER TABLE some_table TABLESPACE=Logs;
DROP TABLESPACE Logs; - must be empty
```

Features : Buffer Pool

Dynamic buffer pool size re-size

Done in a separate thread

--innodb-buffer-pool-chunk-size – resize done in chunk size

Example:

SET GLOBAL innodb-buffer-pool-size=402653184;

Features : UNDO Truncate

UNDO Log Space Management

Requires separate UNDO tablespaces to work

- --innodb-undo-log-truncate := on | off – default off
- --innodb-max-undo-log-size – default 1G
- --innodb-purge-rseg-truncate-frequency – default 128 - advanced

Features : Larger Page Sizes

Support for 32K and 64K Page Sizes

Larger BLOBs can be stored “on-page”

Better compression with the new transparent page compression

Features : GIS

Spatial index

Implemented as an R-Tree

Supports all MySQL geometric types

Currently only 2D supported

Supports transactions & MVCC

Uses predicate locking to avoid phantom reads

Features : GIS

R-Tree

Multi-dimension spatial data search

Queries more like:

- Find object “within”, “intersects” or “touches” another object
- MySQL geometric types
  - POINT, LINESTRING, POLYGON, MULTIPOINT,
  - MULTILINESTRING, MULTIPOLYGON, GEOMETRY

Features : Mutexes

Flexible mutexes

Mix and match mutex types in the code – build time option only

Can use futex on Linux instead of condition variables

Futex eliminates “thundering herd” problem

Not enabled by default, build with -DMUTEX_TYPE=”futex” from source

Features : Virtual Columns and Index on Virtual Columns in InnoDB

Virtual column is not stored within the InnoDB table(unless indexed)

Only virtual column's meta-data stored in the data dictionary

```
CREATE TABLE t (a INT, b INT, c INT GENERATED ALWAYS AS(a+b), PRIMARY KEY(a));
ALTER TABLE t ADD new_col INT GENERATED ALWAYS AS (a - b) VIRTUAL;
ALTER TABLEt  ADD INDEX IDX(new_col);
```

Current limitations:

- Primary Keys cannot contain any virtual columns
- Spatial and fulltext index not supported (for now)
- Cannot be used as a foreign key

Features : Transparent Data Encryption (TDE)

Two tier encryption
Master Key
- Key ring plugin provides interface to manage the Master Key
- Only the Master Key Is rotated

Tablespace key (automatically generated)
- Stored in the tablespace header
- Encrypted with the Master Key

Algorithm: AES - block encryption mode(CBC)

Features : Transparent Data Encryption (TDE)

Example:
Start the server with:

--early-plugin-load=keyring_file.so –keyring_file_data=./ring

```
CREATE TABLE t … ENCRYPTION=”Y”;
ALTER TABLE t  ENCRYPTION=”N”, ALGORITHM=COPY;
FLUSH TABLES t FOR EXPORT;
Copy t.cfg, t.cfp and t.ibd to another server
ALTER TABLE t DISCARD TABLESPACE;
ALTER TABLE t IMPORT TABLESPACE;
```

Note: Only supports COPY

Features : Full Text Search

Support for external parser

For tokenizing the document and the query

Example:

```
CREATE TABLE t1 (
  id INT AUTO_INCREMENT PRIMARY KEY,
  doc CHAR(255), FULLTEXT INDEX (doc) WITH PARSER my_parser) ENGINE=InnoDB;
ALTER TABLE articles ADD FULLTEXT INDEX (body) WITH PARSER my_parser;
CREATE FULLTEXT INDEX ft_index ON articles(body) WITH PARSER my_parser;
```

Features : Full Text Search

n-gram parser for CJK

```
CREATE TABLE articles(
        FTS_DOC_ID BIGINT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
        title VARCHAR(100),
        FULLTEXT INDEX ngram_idx(title) WITH PARSER ngram
) Engine=InnoDB CHARACTER SET utf8mb4;

ALTER TABLE articles ADD FULLTEXT INDEX ngram_idx(title) WITH PARSER ngram;
CREATE FULLTEXT INDEX ngram_idx ON articles(title) WITH PARSER ngram;
```

--ngram-token-size := 1 .. 10 (default 2)

Features : Full Text Search

MECAB parser

```
INSTALL PLUGIN mecab SONAME 'libpluginmecab.so';

SHOW STATUS LIKE 'mecab_charset';

 mysql> CREATE TABLE articles(
        FTS_DOC_ID BIGINT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
        title VARCHAR(100),
        FULLTEXT INDEX mecab_idx (title) WITH  PARSER mecab
) ENGINE=InnoDB CHARACTER SET utf8mb4;
```

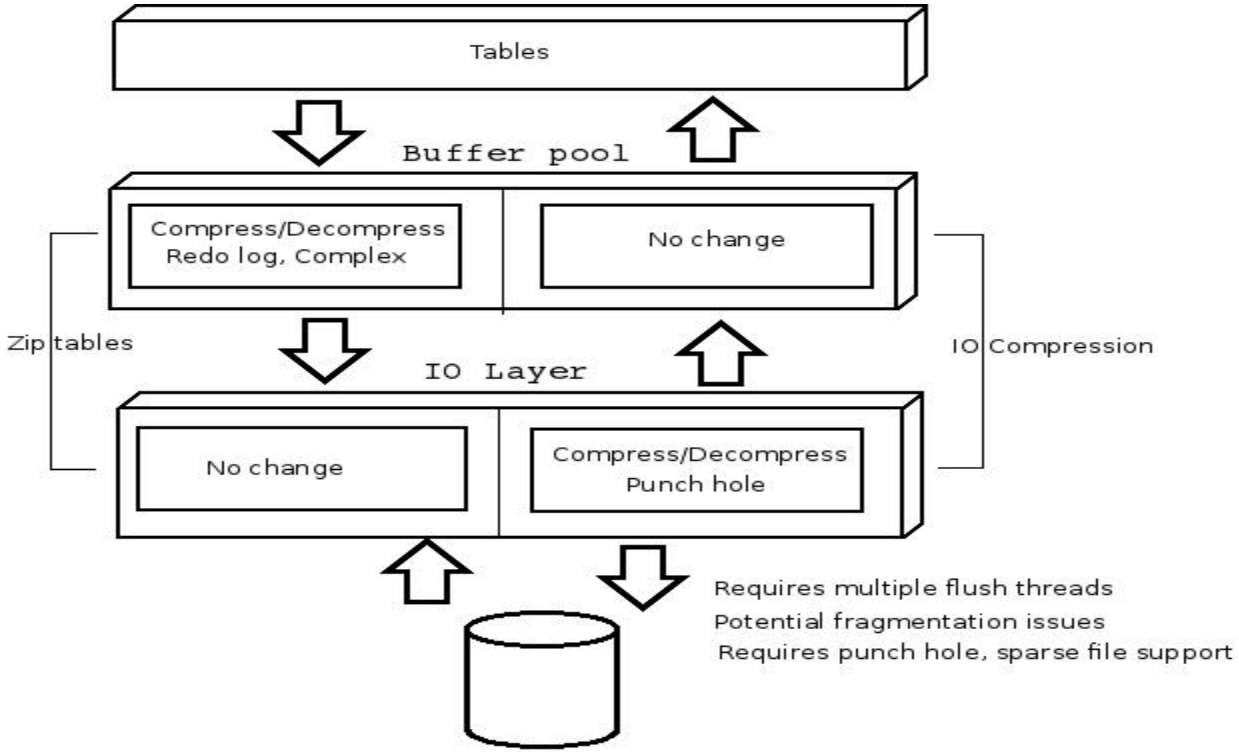
Features : Sandisk/FusionIO Atomic Writes

No new configuration variables – may change in GA
System wide setting
Disables the doublewrite buffer if the system tablespace is on NVMFS

Features : Transparent PageIO Compression

Proof of concept patch originally from FusionIO
Currently Linux/Windows only
Requires sparse file support : NVMFS, XFS, EXT4, ZFS & NTFS
Linux 2.6.39+ added PUNCH HOLE support
Can co-exist with current Zip tables
Only works on tablespaces that are not shared
Doesn't work on the system tablespace

Features : Transparent PageIO Compression



Features : Zip vs Page IO compression

Zip compression

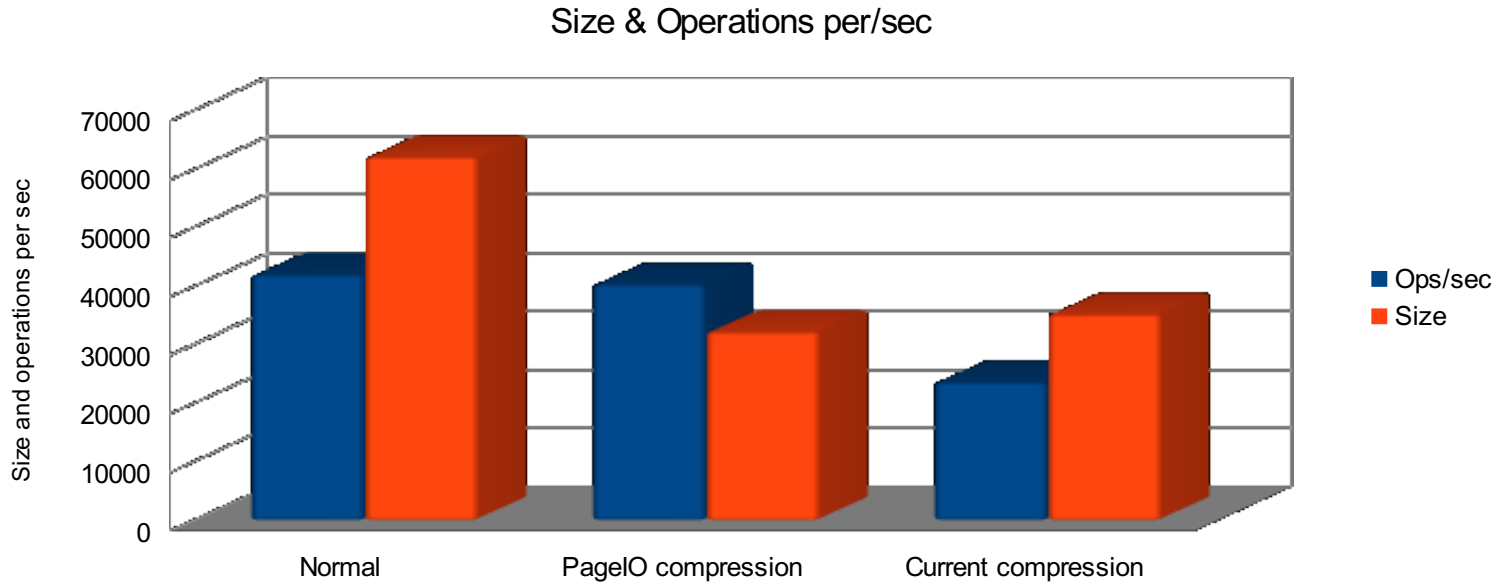
- Tested and tried, works well enough
- Complicates buffer pool code
- Special page format required
- No IO layer changes
- Algorithm supported - Zlib
- Can't compress system tablespace
- Can't compress UNDO tablespace

PageIO compression

- Requires OS/FS support
- Simple
- Works with all file types, system tablespaces
- Potential fragmentation issues
- NVMFS doesn't suffer from fragmentation
- Adds to the cost of IO
- Current algorithms are tuned to existing assumptions
- Requires multi-threaded flushing
- Easy to add new algorithms.

Features : PageIO Compression Benchmark

FusionIO – 25G BP – maxid 50 Million 64 Requesters - Linkbench



Features : Transparent PageIO Compression

New syntax

```
CREATE TABLE T(C INT) ENGINE=InnoDB, COMPRESSION=”ZLIB”;
CREATE TABLE T(C INT) ENGINE=InnoDB, COMPRESSION=”LZ4”;
ALTER TABLE T COMPRESSION=”LZ4”
ALTER TABLE T COMPRESSION=”ZLIB”;
ALTER TABLE T COMPRESSION=”NONE”;
OPTIMIZE TABLE T;
```

Features : Miscellaneous

Implement update_time for InnoDB tables
Improve select count(*) performance by using handler::records();
Improve recovery, redo log tablespace meta data changes
    No need to scan the entire directory on startup for .ibd files
Make innodb_checksum_algorithm=CRC32 the default
   The previous one was "innodb".
Default file format is now Barracuda
   Allows larger index prefixes

Features : Observability

Integrate PFS memory instrumentation with InnoDB

Memory allocated by InnoDB is accounted in PFS.

Start mysqld with --performance-schema-instrument='memory/%=on'

- memory_summary_by_account_by_event_name
- memory_summary_by_host_by_event_name
- memory_summary_by_thread_by_event_name
- memory_summary_by_user_by_event_name
- memory_summary_global_by_event_name

Features : Observability

Integrate PFS memory instrumentation with InnoDB

Example:

- SELECT event_name, current_number_of_bytes_used
FROM performance_schema.memory_summary_global_by_event_name
WHERE event_name LIKE '%innodb%' ORDER BY 2 DESC;

Features : Observability

Add InnoDB events to Performance Schema's Event Stage table

Monitor Buffer pool load and "ALTER TABLE" progress

Start mysqld with:
--performance-schema-consumer-events-stages-current='ON'
--performance-schema-consumer-events-stages-history='ON'
--performance-schema-consumer-events-stages-history-long='ON'
--performance-schema-instrument='stage/%=on'

Features : Observability

Add InnoDB events to Performance Schema's Event Stage table

Look into events_stages_current for stages names like '%innodb%' while Alter table and buffer pool load are active
The relevant PFS tables are:

- events_stages_current
- events_stages_history
- events_stages_history_long
- events_stages_summary_by_account_by_event_name
- events_stages_summary_by_host_by_event_name
- events_stages_summary_by_thread_by_event_name
- events_stages_summary_by_user_by_event_name
- events_stages_summary_global_by_event_name

Features : Observability

Better SHOW ENGINE INNODB MUTEX;

```
mysql> show engine innodb mutex;
+--------+-----------------------------------------+-----------+
| Type   | Name                                    | Status    |
+--------+-----------------------------------------+-----------+
| InnoDB | rwlock: log0log.cc:785                  | waits=2 |
| InnoDB | sum rwlock: buf0buf.cc:1379             | waits=1 |
+--------+-----------------------------------------+-----------+
```

Features : Observability

Better SHOW ENGINE INNODB MUTEX;

```
mysql> set global innodb_monitor_enable="latch";

mysql> show engine innodb mutex;
+--------+-------------------------------+-----------------------------+
| Type   | Name                          | Status                      |
+--------+-------------------------------+-----------------------------+
| InnoDB | FIL_SYSTEM                    | spins=392,waits=13,calls=14 |
| InnoDB | LOG_SYS                       | spins=30,waits=1,calls=1    |
| InnoDB | BUF_POOL                      | spins=1,waits=0,calls=1     |
| InnoDB | rwlock: dict0dict.cc:1184     | waits=2                     |
| InnoDB | rwlock: log0log.cc:785        | waits=9                     |
+--------+-------------------------------+-----------------------------+
mysql> set global innodb_monitor_disable="latch";
```

Download & Blogs

http://labs.mysql.com

http://dev.mysql.com/downloads/mysql/

http://mysqlserverteam.com/