MySQL 5.7 in a Nutshell

Alexander Rubin
Principal Architect, Percona
December 6, 2015

About Me

My name is Alexander Rubin

- Working with MySQL for over 10 years
  - Started at MySQL AB, then Sun Microsystems,
  - then Oracle (MySQL Consulting)
  - Joined Percona 2 years ago

https://www.linkedin.com/in/alexanderrubin

Agenda: New MySQL 5.7 features

Performance and Scalability

- **Enhanced Speed**: MySQL 5.7 delivered 1,600,000 queries per second (QPS) – 3x faster than MySQL 5.6.
- **Optimized InnoDB**: New capabilities include increased performance and concurrency
- **More Robust Replication:**
  - multi-source replication
  - enhanced Global Transaction Identifiers (GTIDs)
  - improved multi-threaded slaves
- **Enhanced Optimizer:** A new dynamic cost model provides better query performance and greater user control.

Agenda: New MySQL 5.7 features

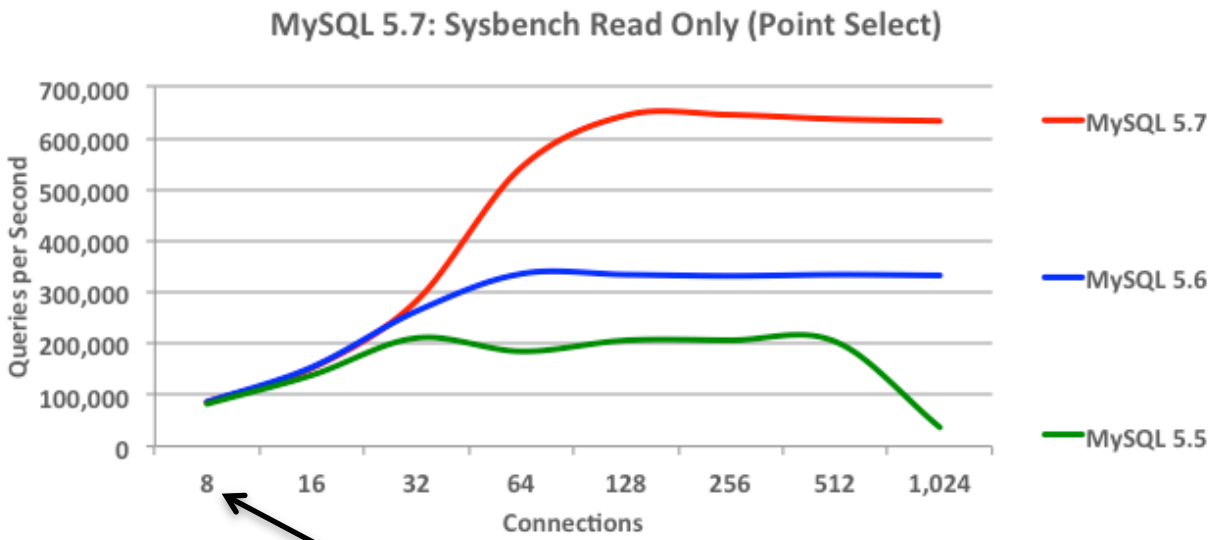
Manageability enhancements

- **JSON Data Type and Calculated Fields**: Allows for efficient and flexible storage, search, and manipulation of schema-less data.
- **Performance Schema**: Enables instrumentation for memory, transactions, stored routines, prepared statements, replication, and locks.
- **MySQL SYS Schema**: Provides helper objects that answer common performance, health, usage, and monitoring questions.
- **Improved Security**: Delivers easier and safer instance initialization, setup and management.
- **Expanded Geographic Information System (GIS) Support**: Spatial index support in InnoDB, GeoJSON, and GeoHash.

MySQL 5.7 Performance Improvements Sysbench Becnhmark

2x Faster than MySQL 5.6
3x Faster than MySQL 5.5

645,000 QPS



Intel(R) Xeon(R) CPU E7-4860 x86_64
4 sockets x 10 cores-HT (80 CPU threads)
2.3 GHz, 512 GB RAM
Oracle Linux 6.5

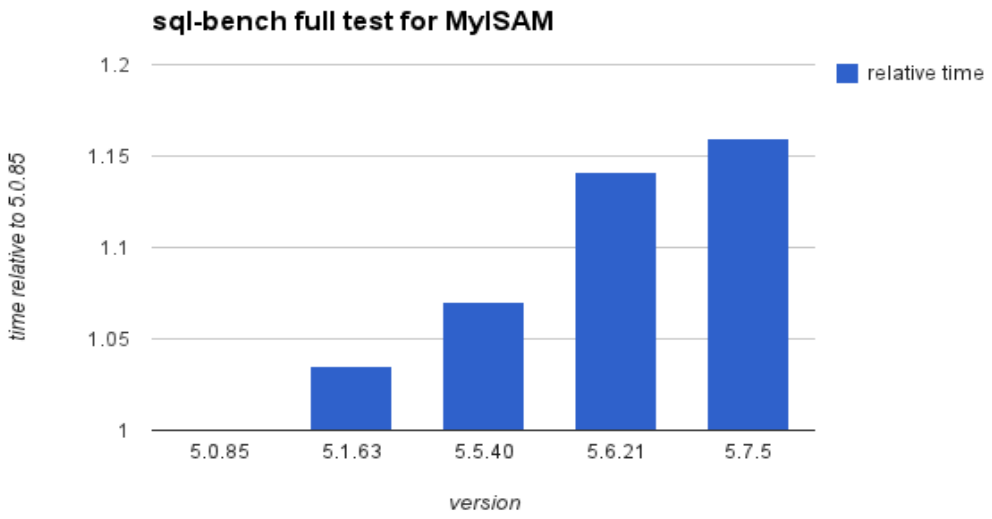
Starts with 8 Threads
What about 2-4 threads?

**Information from Oracle OpenWorld presentation by Geir Hoydalsvik*

MySQL 5.7: Single-threaded workload

- Multi-threaded workload looks great
- Single-threaded workload shows some regression

https://bugs.mysql.com/bug.php?id=68825



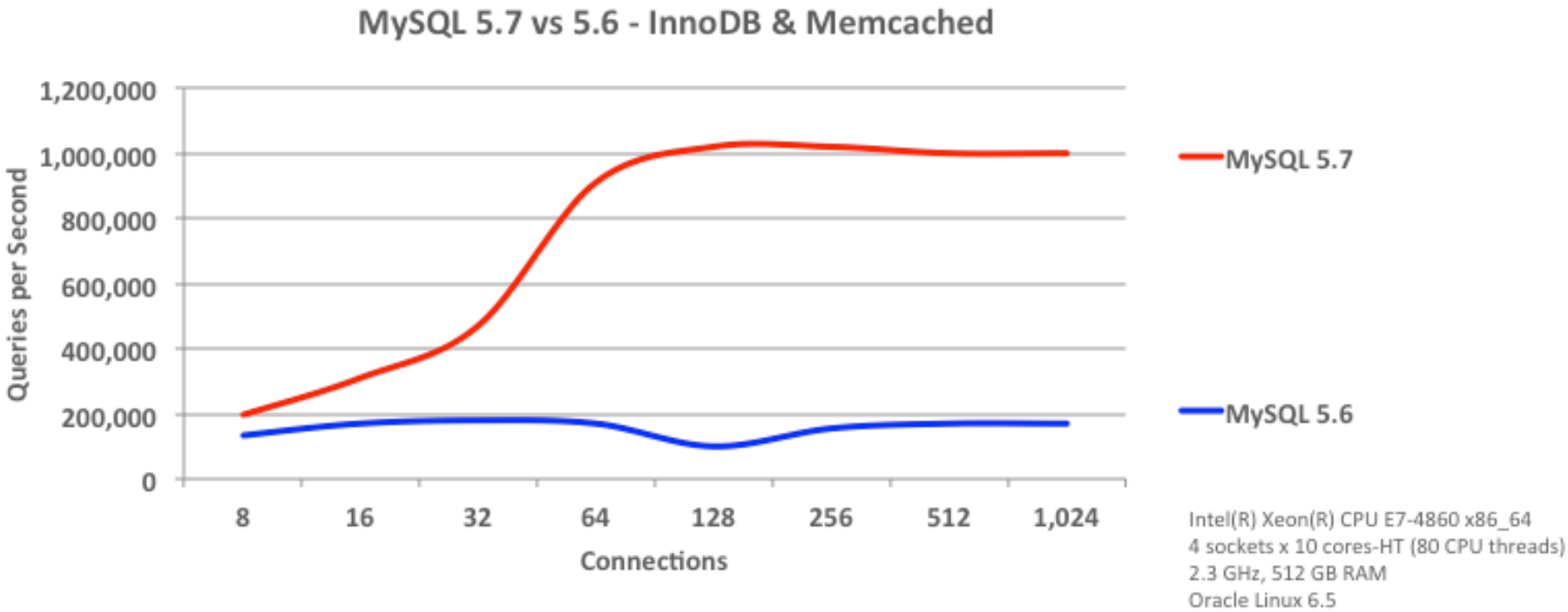
http://smalldatum.blogspot.co.uk/2014/10/single-thread-performance-in-mysql-575.html

MySQL 5.7: InnoDB, NoSQL With Memcached

6x Faster than MySQL 5.6

Thank you, Facebook

1 Million QPS

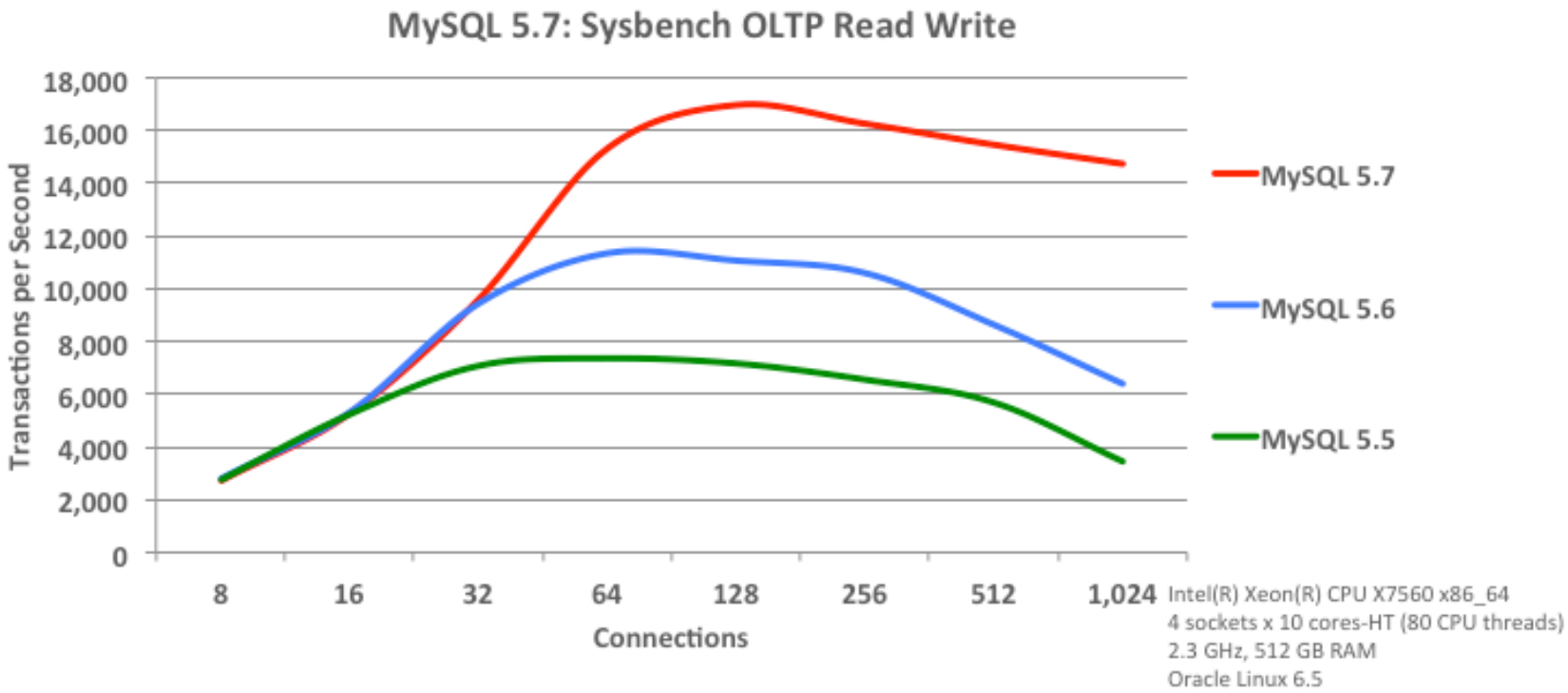


**Information from Oracle OpenWorld presentation by Geir Hoydalsvik*

Sysbench OLTP Read Write

1.5x Faster than MySQL 5.6
2.5x Faster than MySQL 5.5

17,000 TPS



**Information from Oracle OpenWorld presentation by Geir Hoydalsvik*

InnoDB vs. MyISAM in v. 5.7

Feature	MyISAM	InnoDB
Full Text Indexes	yes	Since MySQL 5.6
Portable tables (tablespaces)	yes	Since MySQL 5.6
Spatial Indexes/RTREE (GIS)	yes	*Since MySQL 5.7*
Last update for table	yes	*Since MySQL 5.7* *(http://dev.mysql.com/worklog/task/?id=6658)*
Suitable for temp tables	yes	*Since MySQL 5.7* *Also complex selects uses InnoDB ondisk temp tables*
Fast count(*)	yes	**Faster in MySQL 5.7 but does not store counter*

InnoDB Improvements Overview

- **Performance:** Buffer pool improvements
- **Performance:** Better Redo log handling and better index->lock handling
- **Performance:** DDL & Truncate improvements
- **Performance:** Temporary Table Optimizations
- **Feature**: Native Partitioning for InnoDB
- **Feature**: Dynamic buffer pool size re-size, more online alter table ops
- **Feature**: UNDO Log Space Management
- **Feature:** Transparent PageIO Compression
- **Feature:** GIS indexes
- **Miscellaneous**
  - Implement update_time for InnoDB tables
  - Improve select count(*) performance by using handler::records();
  - Improve recovery, redo log tablespace meta data changes

InnoDB: Online Operations

- Resize the InnoDB Buffer Pool online
- More Online ALTER TABLE operations
  - Enlarge VARCHAR, Rename Index
- More dynamic configuration variables
  - New variables are dynamic
  - Work to make existing variables dynamically settable

Dynamic buffer pool re-size

- May be useful in virtual enviroments where you can resize RAM online

  innodb_buffer_pool_chunk_size – resize done in chunk size

- Example:

```
mysql> SET GLOBAL
innodb_buffer_pool_size=4*1024*1024*102
4; -- 4G
```

http://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool-online-resize.html

InnoDB - Bulk Load for Create Index

- Much faster INDEX creation and bulk loads
- Performance results show
  - 2-3x performance improvement for ADD/CREATE INDEX operations
  - 2-5% improvement for standard INSERT operations

InnoDB Temporary Tables

- New separate tablespace for temporary tables
- Optimize DML operations
  - *No REDO logging, no change buffering, less locking*
- InnoDB storage engine is used for on-disk internal temporary tables
  - Complex select requiring ondisk temp tables will now use InnoDB by default (controlled by internal_tmp_disk_storage_engine variable)

http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_internal_tmp_disk_storage_engine

http://mysqlserverteam.com/mysql-5-7-innodb-intrinsic-tables/

Transparent Page Compression

- Transparent Page Level Compression
  - Happens transparently in background threads
  - For supported Linux kernels and filesystems
  - Uses sparse file and "hole punching" support
  - Reduces IO

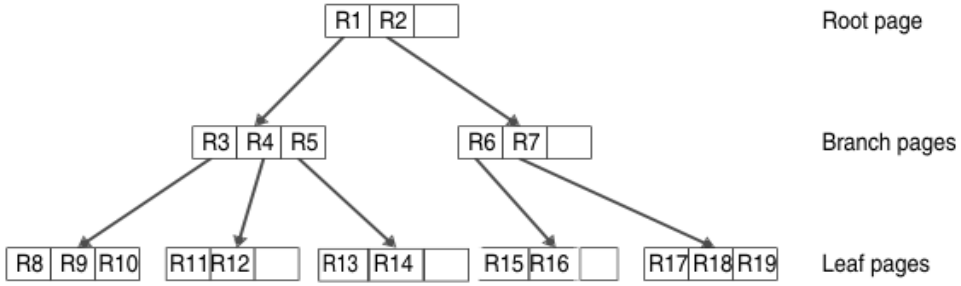
Applies to all InnoDB data, including the system tablespace and UNDO logs

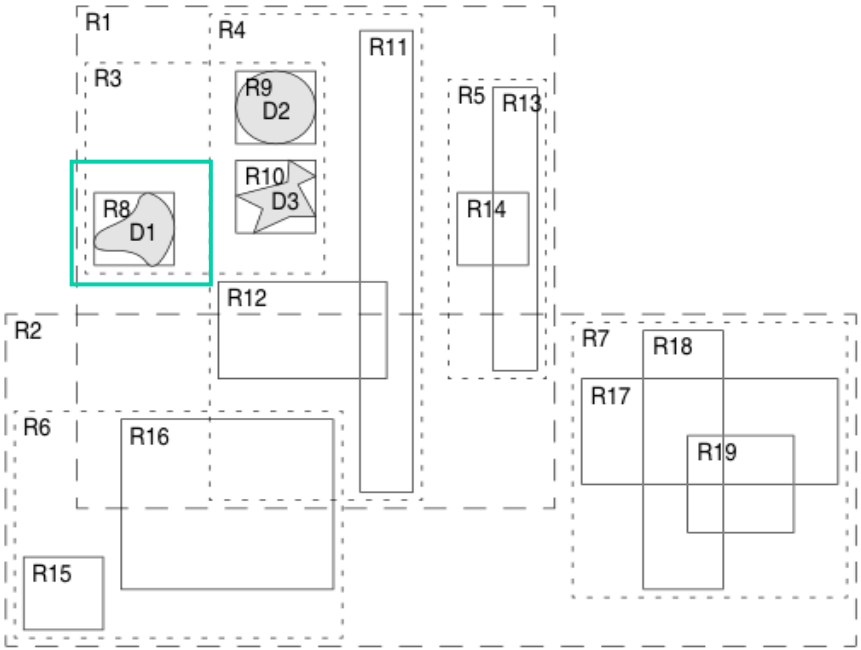
But – many file systems do not support large number of "holes" well

https://dev.mysql.com/doc/refman/5.7/en/innodb-page-compression.html

GIS - InnoDB Spatial Indexes

- Full transactional support
- Only supports 2D data for now
  - Use helper functions for distance calculations





**Graphics from Oracle OpenWorld presentation by Geir Hoydalsvik*

GIS - Additional Features

- GeoHash
  - Quick lookups for exact matches
  - Not very accurate for proximity searches
- GeoJSON
- Helper functions (**`ST_Distance_Sphere`**)

GIS - Example

```
mysql> SELECT shape into @zip_shape FROM zcta.tl_2013_us_zcta510
WHERE zcta5ce10='27701’;
Query OK, 1 row affected (0.20 sec)

mysql> SELECT name,
       ST_Distance_Sphere(shape, st_centroid(@zip_shape) ) as dist,
       ST_AsGeoJSON(shape) as GeoJSON,
       ST_GeoHash(shape, 16) as GeoHash
       FROM points
       WHERE ST_Within(shape, @zip_shape)
       and other_tags like '%"amenity"=>"cafe"%' LIMIT 1\G

*************************** 1. row ***************************
   name: Blue Coffee Cafe
   dist: 374.9045320478079
GeoJSON: {"type": "Point", "coordinates": [-78.9013567, 35.996332]}
GeoHash: dnruu8cvc4sk26qz
1 row in set (0.02 sec)
```

Generated (Virtual) Columns

```
CREATE TABLE `ontime` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `YearD` year(4) NOT NULL,
  `FlightDate` datetime DEFAULT NULL,
  `Carrier` char(2) DEFAULT NULL,
  `OriginAirportID` int(11) DEFAULT NULL,
  `OriginCityName` varchar(100) DEFAULT NULL,
  `OriginState` char(2) DEFAULT NULL,
  `DestAirportID` int(11) DEFAULT NULL,
  `DestCityName` varchar(100) DEFAULT NULL,
  `DestState` char(2) DEFAULT NULL,
...
`Flight_dayofweek` tinyint(4)
GENERATED ALWAYS AS (dayofweek(FlightDate)) VIRTUAL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;
alter table ontime add key (Flight_dayofweek);
```

```
SELECT Flight_dayofweek, count(*)
FROM ontime_sm_virtual
GROUP BY Flight_dayofweek
```

Does not store the column
But INDEX it

https://www.percona.com/blog/2015/04/29/generated-virtual-columns-in-mysql-5-7-labs/
https://dev.mysql.com/worklog/task/?id=8114

Generated (Virtual) Columns

```
mysql> EXPLAIN SELECT carrier, count(*)
       FROM ontime_sm_virtual
       WHERE Flight_dayofweek = 7 group by carrier\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: ontime_sm_virtual
   partitions: NULL
         type: ref
possible_keys: Flight_dayofweek
          key: Flight_dayofweek
      key_len: 2
          ref: const
         rows: 165409
     filtered: 100.00
        Extra: Using where; Using temporary; Using filesort
1 row in set, 1 warning (0.00 sec)
```

Using index

JSON Support

```
mysql> create table json_test (
id int primary key auto_increment,
data json
) engine=InnoDB;
Query OK, 0 rows affected (0.02 sec)
```

Same as
JSON_EXTRACT(data,"$.type")

```
mysql> select * from json_test where data->'$.type' = 'Point' limit 1;
+----+-------------------------------------------------------------+
| id | data                                                        |
+----+-------------------------------------------------------------+
|  1 | {"type": "Point", "coordinates": [-87.9101245, 41.7585879]} |
+----+-------------------------------------------------------------+
```

JSON Support: Indexes

```
mysql> explain select * from json_test where data->'$.type' = 'Point' limit 1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: json_test
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 996823
     filtered: 100.00
        Extra: Using where

mysql> alter table json_test
add data_type varchar(255) GENERATED ALWAYS AS (data->'$.type') VIRTUAL;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table json_test add key (data_type);
Query OK, 0 rows affected (2.51 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

JSON Support: Indexes

```
mysql> explain select * from json_test where data->'$.type' = 'Point' limit 1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: json_test
   partitions: NULL
         type: ref
possible_keys: data_type
          key: data_type
      key_len: 258
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
```

Replication Improvements

Multi-Source Replication
(slave can have multiple masters)

Better Multi-Threaded Slaves
(does not require multiple databases)

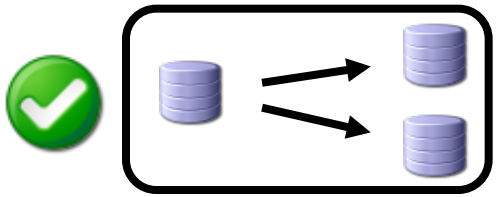
Better GTID
(online GTID deployment)

Group Replication Plugin
(virtual synchronous replication)

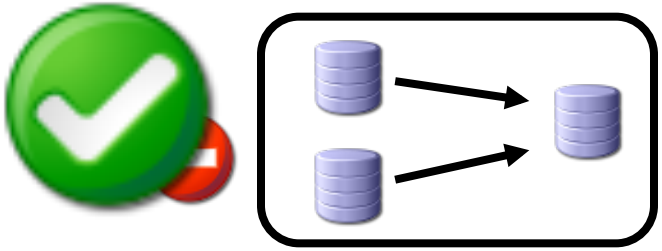
MySQL Replication Topologies

Master > Slave

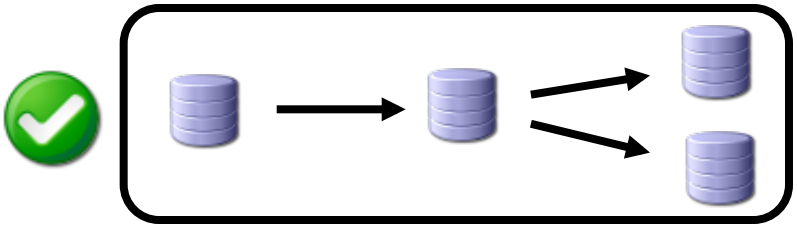
Master > Slaves



Masters > Slave (Multi-Source), **Since 5.7**



Master > Slave > Slaves



- Now 1 replication slave can have many master servers!
- Important for BI/Data Science/Ad-hoc analytics

MySQL 5.7: Optimizer Improvements

- UNION ALL queries no longer use temporary tables
- Improved optimizations for queries with IN expressions
- Improved optimizations for full-text queries
- More efficient sorting

Temporary Table for UNION ALL

```
*************************** 1. row ***************************
           table: ontime_2012
             key: covered
            ... Removed ...
           Extra: Using where; Using i
*************************** 2. row *****
           table: ontime_2012
             key: covered
            ... Removed ...
           Extra: Using where; Using i
*************************** 3. row *****
              id: NULL
     select_type: UNION RESULT
           table: <union1,2>
            type: ALL
   possible_keys: NULL
             key: NULL
         key_len: NULL
             ref: NULL
            rows: NULL
           Extra: Using temporary
```

5.6: Will create temp table (as shown)
5.7: Do not materialize in temporary tables (unless used for sorting) rows are sent directly to client
5.7: Client will receive the first row faster
5.7: Less memory and disk consumption

Optimizations for IN Expressions

- Imagine that you got list of IDs from Full Text Search solution (solr/elasticsearch/sphinx)
- Now I need to get the actual documents

```
mysql> select name, other_tags from poi_info
WHERE osm_id in (367909272, 367841688, 493001986, ...);
```

- MySQL 5.7: IN queries with row value expressions executed using range scans.

Performance Schema improvements

Memory Instrumentations

- Memory used (bytes)
- Operation counts
- Type of memory used (caches, internal buffers, etc)

Statement Instrumentations

- Stored Procedures
- Stored Functions
- Prepared Statements
- Transactions

Other Instrumentation

- Replication slave status
- MDL lock instrumentation
- User variables per thread
- Server stage tracking
- Track long running SQL

SYS Schema Included in MySQL 5.7

Get the memory usage per user with SYS schema:

```
mysql> update
performance_schema.setup_instruments
set enabled='YES', timed='YES'
where name like 'memory/%';
Query OK, 375 rows affected (0.00 sec)
Rows matched: 375  Changed: 375
Warnings: 0


mysql> select * from
sys.memory_global_total\G
*************************** 1. row
***************************
total_allocated: 90.20 MiB
1 row in set (0.01 sec)
```

```
mysql> select * from
sys.memory_by_user_by_current_bytes\G
*************************** 1. row
***************************
              user: root
current_count_used: 42
 current_allocated: 361.03 KiB
 current_avg_alloc: 8.60 KiB
 current_max_alloc: 248.04 KiB
   total_allocated: 46.34 GiB
*************************** 2. row
***************************
              user: background
current_count_used: 0
 current_allocated: 0 bytes
 current_avg_alloc: 0 bytes
 current_max_alloc: 0 bytes
   total_allocated: 14.72 KiB
2 rows in set (0.01 sec)
```

Improved MDL locking

- Removes bottlenecks around DML access to a single table
  - 10% increased throughput in OLTP_RO/ POINT_SELECT sysbench
  - Optimized for typical DML heavy workloads

http://www.chriscalender.com/troubleshooting-waiting-for-table-metadata-lock-errors-for-both-myisam-and-innodb-tables/

Security - Encryption, Passwords, Installation

- Deployment: enable secure unattended install by default
  - Random password set on install
  - Removed anonymous accounts
  - Deployment without test account, schema, demo files
- AES 256 Encryption
- Password rotation policies
  - Can be set globally, and at the user level

Explain on a Running Query

```
mysql> show processlist\G
...
     Id: 8
Command: Query
   Time: 90
  State: Sending data
   Info: select count(*), osm_id from points_new group by osm_id

mysql> explain for connection 8\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: points_new
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 11368798
     filtered: 100.00
        Extra: Using temporary; Using filesort
```

Shows query plan on connection <id>

Applicable for SELECT/INSERT/DELETE/UPDATE

New Data Dictionary, New Tablespace Management (future)

- SaaS case: 50K databases inside single instance
  - ~ 1M tables = 2M files inside MySQL datadir

- InnoDB tables replace .frm, .trg, .trn, .par files
- Ability to create 1 tablespace for multiple tables

FAQ

Q: Do you recommend upgrading to 5.7 right now?
A: As with all upgrades to a newer version it should be tested.
You can expect to see **bugs.**
Upgrade non-critical replication slaves first (i.e. reporting slaves)
Use pt-upgrade to test for any regressions
(https://www.percona.com/doc/percona-toolkit/2.2/pt-upgrade.html)

Q: When Percona Server 5.7 will be released as GA?
A: We expect to have a release around the end of January 2016

Please note that this date is approximate as there is always high risk of a large amount of bugs to be discovered following a major release.

Special Thanks

Mike Frank - Senior Product Manager, Oracle
Geir Høydalsvik - Software Development Director, Oracle
Presented “What’s New in MySQL 5.7”

Sunny Bains - Senior Engineering Manager, Oracle
Presented “MySQL 5.7: InnoDB—What’s New”

Mark Leith – Senior Software Development Manager, Oracle
Developer of SYS Schema

Chris Calender - Principal Support Engineer for MariaDB
Great blog posts explaining metadata locking
http://www.chriscalender.com/tracking-metadata-locks-mdl-in-mysql-5-7/

All Oracle MySQL Developers for the great MySQL 5.7 release!

Thank you!

Alexander Rubin

https://www.linkedin.com/in/alexanderrubin