MySQL 5.7新特性介绍

guangpu.feng@dmall.com
2016.05

主要内容

- 版本介绍
- 新特性
  - JSON
  - Functional index
  - GIS
  - Parallel replication
  - Statement timeout
  - Multi source replication
  - CJK parser for fulltext index
  - Performance

版本介绍

MySQL 5.7.13 (Not yet released)

MySQL 5.7.12 (2016-04-11)

MySQL 5.7.11 (2016-02-05)

MySQL 5.7.10 (2015-12-07)

} AWS RDS

MySQL 5.7.9 (2015-10-21, General Availability)

MySQL 5.7.8 (2015-08-03)

MySQL 5.7.7 (2015-04-08, Release Candidate)

JSON

- 原生支持
  - 底层BSON存储
  - insert/update时语法检查
  - 基于path expr的CRUD
  - in-place update
  - 支持doc内字段索引（functional index）

```
CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `data` json DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
mysql> INSERT INTO t1(data) VALUES(
'{"key1": "value1", "key2": "value2"}');
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+----+-------------------------------------+
| id | data                                |
+----+-------------------------------------+
|  1 | {"key1": "value1", "key2": "value2"} |
+----+-------------------------------------+
1 row in set (0.00 sec)

mysql> select data->"$.key2" from t1;
+----------------+
| data->"$.key2" |
+----------------+
| "value2"       |
+----------------+
1 row in set (0.01 sec)

mysql> select data->"$.key[0]" from t1;
+---------------+
| data->"$.key" |
+---------------+
| NULL          |
+---------------+
1 row in set (0.01 sec)
```

```
mysql> update t1 set data=json_insert(data, "$.key3", "value3");
mysql> select * from t1;
+----+-------------------------------------------------------+
| id | data                                                  |
+----+-------------------------------------------------------+
|  1 | {"key1": "value1", "key2": "value2", "key3": "value3"}|
+----+-------------------------------------------------------+
1 row in set (0.00 sec)

mysql> update t1 set data=json_replace(data, "$.key2", "VALUE");
mysql> select * from t1;
+----+-------------------------------------+
| id | data                                |
+----+-------------------------------------+
|  1 | {"key1": "value1", "key2": "VALUE"} |
+----+-------------------------------------+
1 row in set (0.00 sec)

mysql> update t1 set data=json_array_append(data, "$.key2", "extra");
mysql> select * from t1;
+----+-----------------------------------------------------------------+
| id | data                                                            |
+----+-----------------------------------------------------------------+
|  1 | {"key1": "value1", "key2": ["VALUE", "extra"], "key3": "value3"}  |
+----+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> update t1 set data=json_remove(data, “$.key3");
mysql> select * from t1;
+----+-------------------------------------------------+
| id | data                                            |
+----+-------------------------------------------------+
|  1 | {"key1": "value1", "key2": ["VALUE", "extra"]}  |
+----+-------------------------------------------------+
1 row in set (0.00 sec)
```

Functional index

- **Generated (Virtual) Columns**

  1. ALTER TABLE t2 add `sum` GENERATED ALWAYS AS ((`a` + `b`));

  2. ALTER TABLE t2 add key idx_sum(sum);

Functional index

```
CREATE TABLE `t2` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `a` int(11) DEFAULT NULL,
  `b` int(11) DEFAULT NULL,
  `sum` bigint(20) GENERATED ALWAYS AS ((`a` + `b`)) VIRTUAL,
  PRIMARY KEY (`id`),
  KEY `idx_sum` (`sum`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- **Generated columns**
  - 在insert/update时自动计算
  - 不允许更新

```
mysql> update t2 set sum=1000 where id=20;
ERROR 3105 (HY000): The value specified for generated column 'sum' in table 't2' is not allowed.
```

```
mysql> explain select sum from t2 where sum=4296620;
+----+-------------+-------+------------+------+---------------+---------+---------+-------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+---------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | t2    | NULL       | ref  | idx_sum       | idx_sum | 9       | const |    1 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+---------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

Functional index

- **Generated columns类型**
  - VIRTUAL:
    - 查询时实时计算结果，不占用存储空间
    - online add/drop
  - STORED:
    - 和普通列一样存储在表中，但不能更新
    - insert/update时计算，查询性能更好
    - add/drop会导致表重建

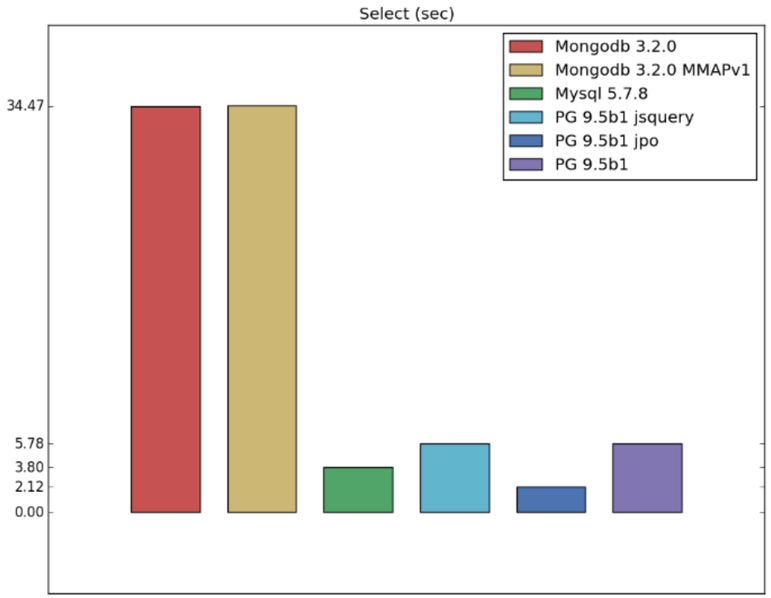
Index on JSON field

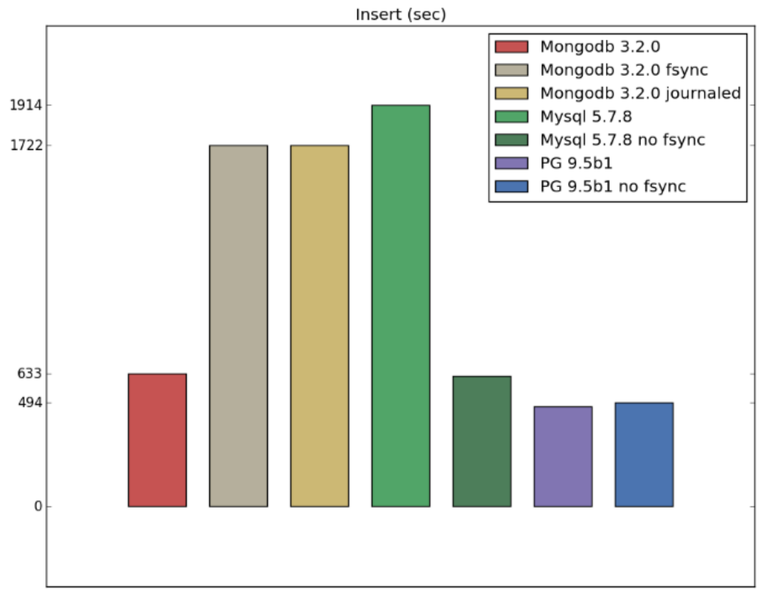
```
mysql> select * from t1;
+-------------------------------------------------------------------------------------+
| data                                                                                |
+-------------------------------------------------------------------------------------+
| {"key1": "value1", "key2": ["value2", "extra"], "key3": "value3"}                   |
| {"code": 1, "error": ["sql has syntax error", "invalid user id"], "event": "query user"} |
+-------------------------------------------------------------------------------------+

mysql> alter table t1 add sql_error varchar(255) GENERATED ALWAYS AS (JSON_SEARCH(data, "all", "%sql%", NULL, "$.error")) NULL;
mysql> alter table t1 add key idx_sql_error(sql_error);

# CREATE TABLE `t1` (
#   `id` int(11) NOT NULL AUTO_INCREMENT,
#   `data` json DEFAULT NULL,
#   `sql_error` varchar(255) GENERATED ALWAYS AS (json_search(`data`,'all','%sql%',NULL,'$.error')) VIRTUAL,
#   PRIMARY KEY (`id`),
#   KEY `idx_sql_error` (`sql_error`)
# ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

mysql> explain select * from t1 where sql_error like '"$.error[%';
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key           | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | t1    | NULL       | range | idx_sql_error | idx_sql_error | 768     | NULL |    1 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
```

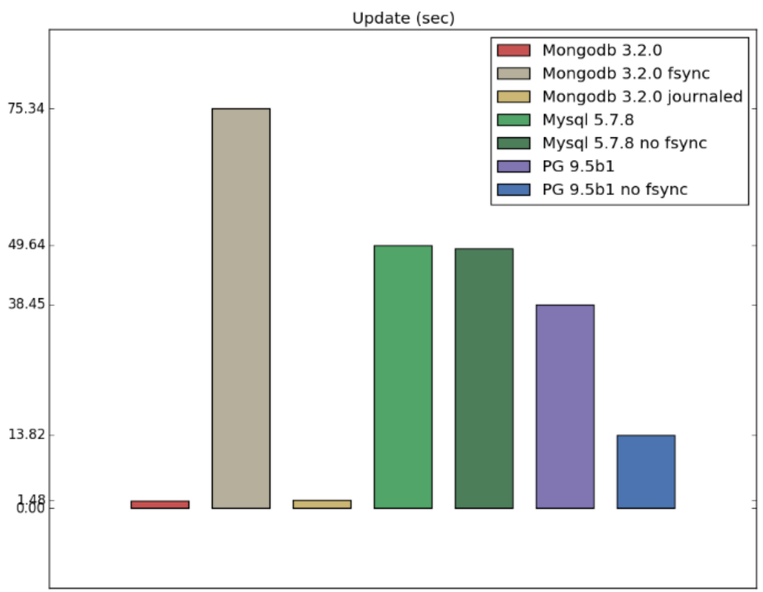
JSON benchmark

m4.xlarge: 4vCPU, 16GB Memory

Y: 完成1000,000次操作所需要的时间



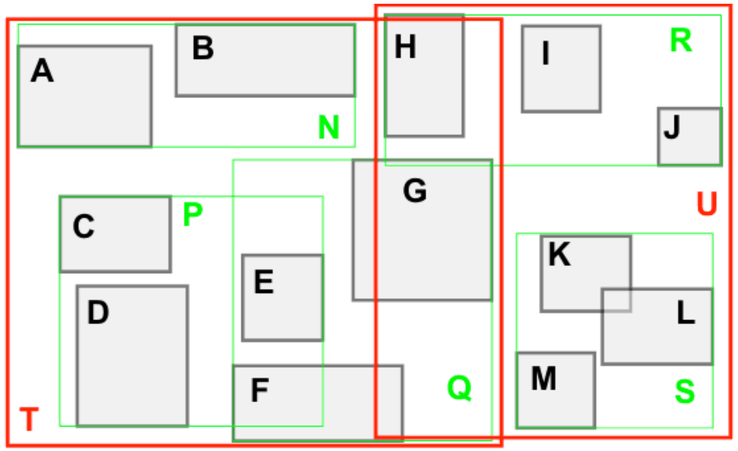


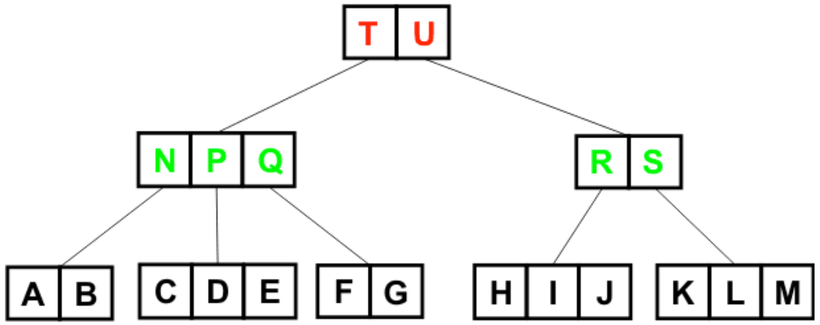


benchmark detail

Gis

- R-tree Index





Gis

- Boost.Geometry
- Data type:
  - POINT, LINESTRING, POLYGON, /MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, /GEOMETRYCOLLECTION
- Native ST_Distance_Sphere
  - ~20x faster than stored procedure
- GeoJSON
  - {“type": "Point", "coordinates": [11.11, 12.22]}
- GeoHash
  - B+tree index on generated hash values

```
CREATE TABLE `t3` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `obj` geometry NOT NULL,
  PRIMARY KEY (`id`),
  SPATIAL KEY `idx_obj` (`obj`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
1 row in set (0.00 sec)

mysql> insert into t3(obj) values(ST_GeomCollFromText('POLYGON((116.438096 40.041324,116.441827 40.039743,116.441748
40.03245,116.429164 40.032607,116.430023 40.038981,116.430205 40.040657,116.430731 40.04161,116.432298 40.04378,116.438096
40.041324))'));
…

mysql> set @g='POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,5 7,7 7,7 5,5 5))';

mysql> explain select ST_AsText(obj) from t3 where st_contains(obj, ST_GeomCollFromText(@g));
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | t3    | NULL       | range | idx_obj       | idx_obj | 34      | NULL | 25042 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+

mysql> explain select ST_AsText(obj) from t3 where st_contains(ST_GeomCollFromText(@g), obj);
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | t3    | NULL       | range | idx_obj       | idx_obj | 34      | NULL | 25042 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+

mysql> explain select ST_AsText(obj) from t3 where st_contains(obj, POINT(1,1));
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | t3    | NULL       | range | idx_obj       | idx_obj | 34      | NULL |    1 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
```

包含多边形g的区域

被包含在多边形g的区域

包含某个POINT的区域

Gis functions

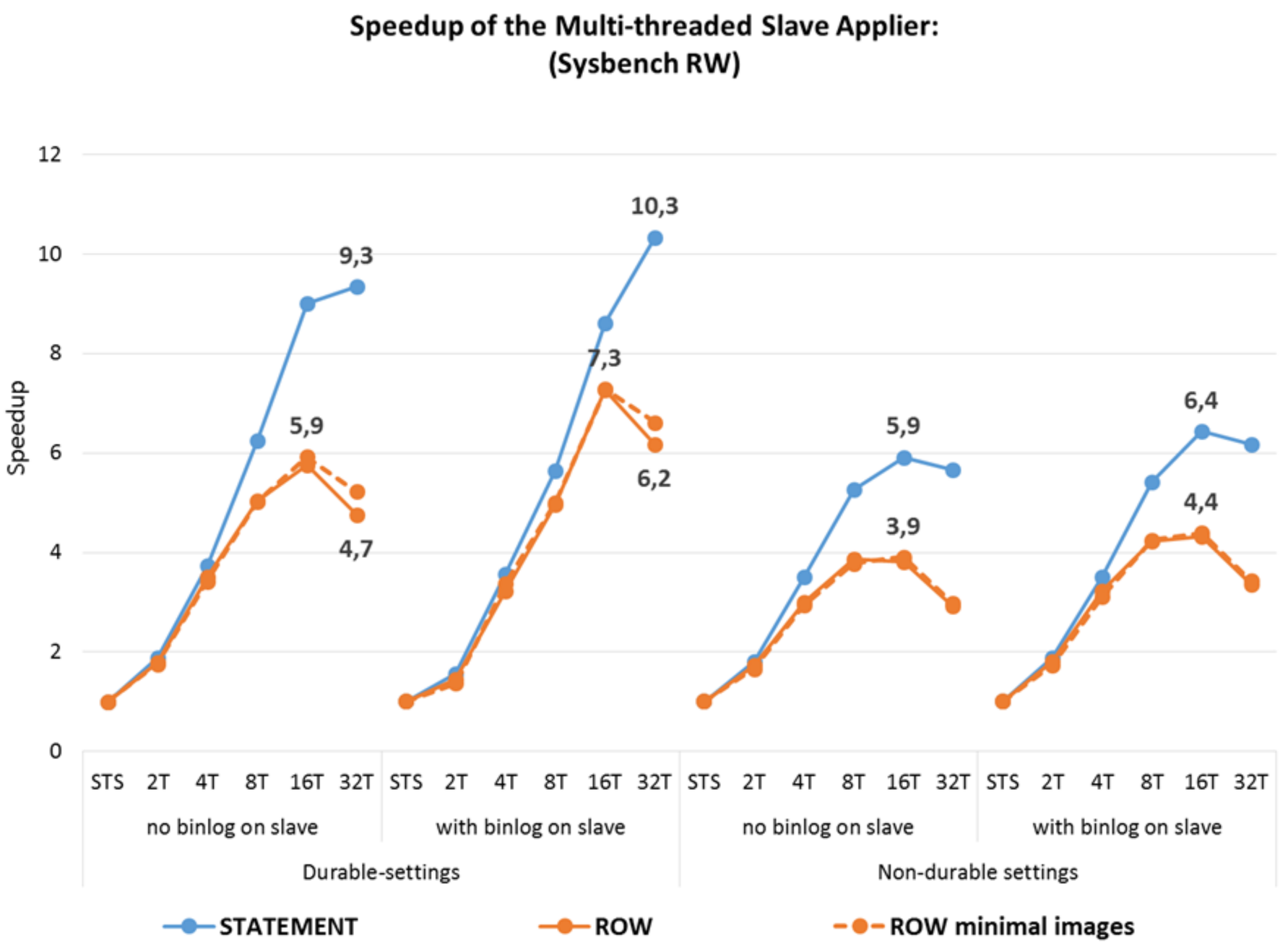
- ST_Contains/ST_Crosses/ST_Disjoint/ST_Equals/ ST_Intersects/ST_Touches/ST_Overlaps/ST_Within
- ST_Intersection/ST_Union
- more —> **MySQL 5.7 Gis Functions**

MySQL 5.7 Gis VS. PostGis

	方式	索引	维度	空间函数	坐标系转换，投影计算	事务支持
PostgreSQL + PostGis	plugin	Gist	**2D, 3D**	**非常丰富**	**Y**	Y
MySQL 5.7 Boost.Geometry	internal	R-tree	**2D** (planing)	**丰富** (planing)	**N** (planing)	Y

Parallel replication

- --slave-parallel-type=type
  - DATABASE（db级别并行复制，5.6版本引入）
  - LOGICAL_CLOCK（transaction级别并行复制）
- --slave-parallel-workers=#
- 性能数据（percona提供）
  - 4MTS ->3.5x times performance



multi-threaded-replication-performance-in-mysql-5-7

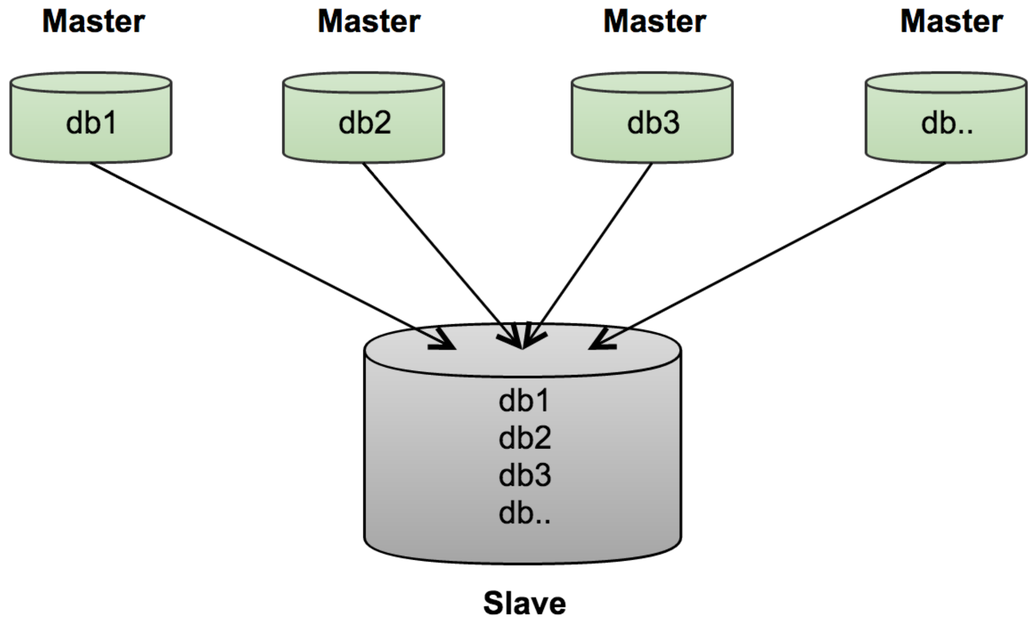
Statement timeout

- server side timeout
- --max_execution_time=#(ms)
  - SELECT statement only

```
mysql> set max_execution_time=1000;
Query OK, 0 rows affected (0.00 sec)

mysql> select count(1) from t3 a, t3 b where a.id>1 and b.id> 1;
ERROR 3024 (HY000): Query execution was interrupted, maximum statement execution time exceeded
```

Multi-source replication

- 数据汇集，方便跨库join
- 节省资源、成本



目前大数据、MyData已使用该特性（MariaDB 10.0.20）

CJK Parser

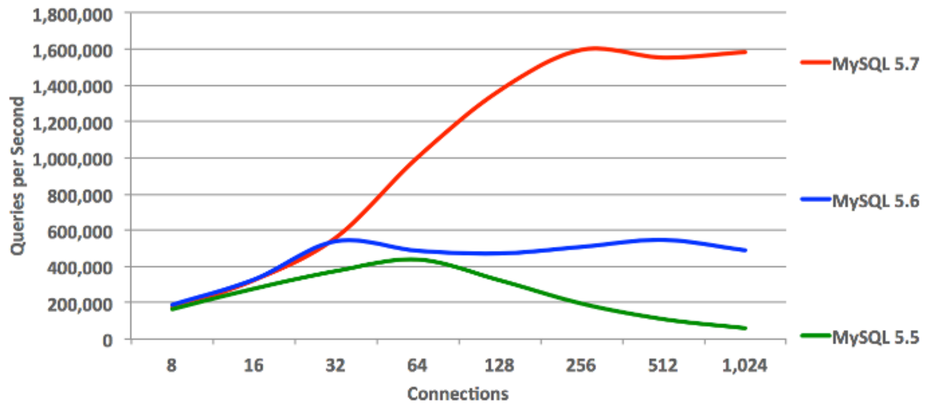
```
CREATE TABLE `t4` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `content` varchar(512) CHARACTER SET utf8mb4 DEFAULT NULL,
  PRIMARY KEY (`id`),
  FULLTEXT KEY `ft_content` (`content`) /*!50100 WITH PARSER `ngram` */
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

ngram_token_size=2(default)

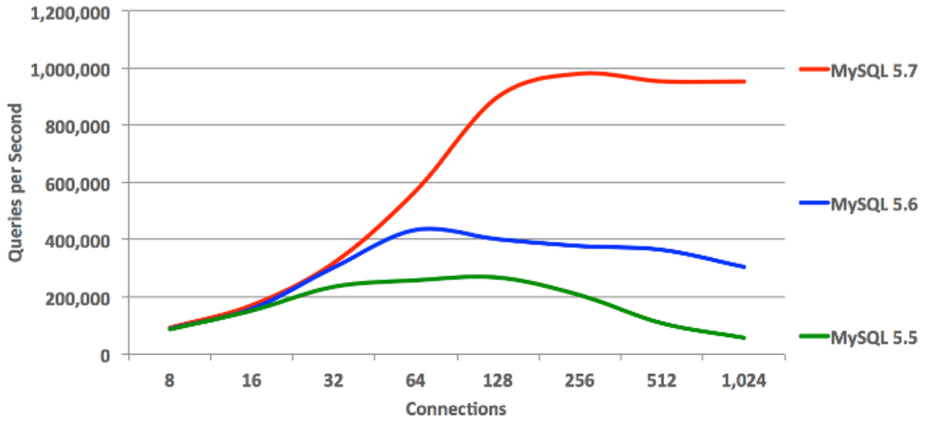
```
mysql> select * from t4;
+----+-----------------------------------------------------+
| id | content                                             |
+----+-----------------------------------------------------+
|  1 | 多点新鲜（北京）电子商务有限公司
|  2 | 网上好超市，生鲜飞速达
|  3 | 多点，让生活多点不同
+----+-----------------------------------------------------+
3 rows in set (0.00 sec)

mysql> select * from t4 where match(content) against('多点' IN NATURAL LANGUAGE MODE);
+----+-----------------------------------------------------+
| id | content                                             |
+----+-----------------------------------------------------+
|  3 | 多点，让生活多点不同
|  1 | 多点新鲜（北京）电子商务有限公司
+----+-----------------------------------------------------+
2 rows in set (0.00 sec)
```

Performance

MySQL 5.7 Sysbench Benchmark: SQL Point Selects

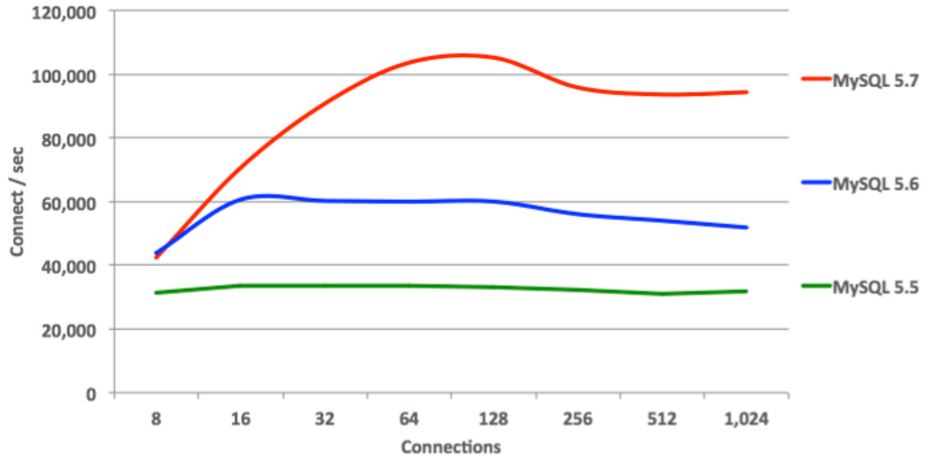


MySQL 5.7 Sysbench Benchmark: OLTP Read Only

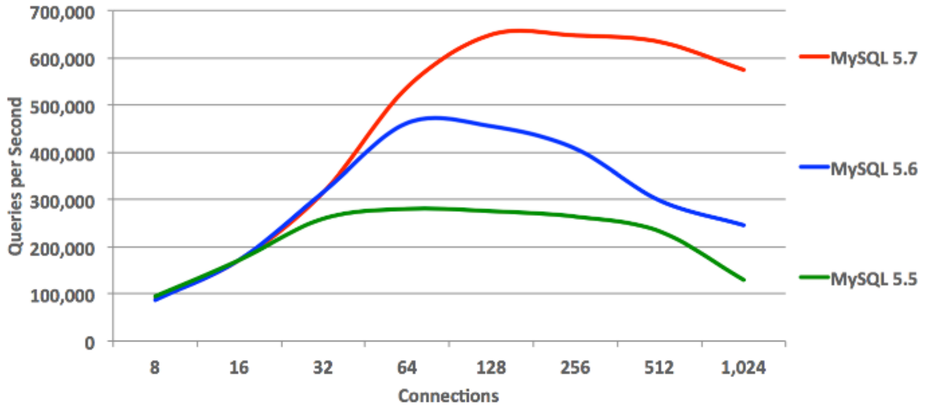


[benchmark detail]

MySQL 5.7 Sysbench Benchmark: Connection Requests



MySQL 5.7 Sysbench Benchmark: OLTP Read Write



More

- offline mode for management
- online innodb buffer pool resize
- online enlarge varchar size(metadata change only)
- online rename index(metadata change only)

…

Reference

- http://www.thecompletelistoffeatures.com
- https://dev.mysql.com/doc/refman/5.7/en/json.html
- https://dev.mysql.com/doc/refman/5.7/en/json-function-reference.html
- https://dzone.com/articles/json-support-postgresql-mysql
- http://mysqlserverteam.com/json-labs-release-effective-functional-indexes-in-innodb
- http://dev.mysql.com/doc/refman/5.7/en/alter-table-generated-columns.html
- http://mysqlhighavailability.com/multi-threaded-replication-performance-in-mysql-5-7
- http://fiercesw.com/wp-content/uploads/2016/01/Whats-New-in-MySQL-5.7-1.pdf
- http://mysqlserverteam.com/mysql-5-7-and-gis-an-example
- http://cglab.ca/~cdillaba/comp5409_project/R_Trees.html
- https://www.percona.com/blog/2016/02/03/new-gis-features-in-mysql-5-7
- https://dev.mysql.com/doc/refman/5.7/en/spatial-analysis-functions.html
- https://dev.mysql.com/doc/refman/5.7/en/spatial-function-reference.html
- http://www.slideshare.net/mattalord/my-sql-57gis
- http://www.mysql.com/why-mysql/benchmarks/