

GreenCloud for MySQL Performance Brief

Social Web Database Benchmark

*A database benchmark designed for Social
Networking Website workloads*

Social Web Database Benchmark

A database benchmark designed for a social networking website workload

Introduction

Social networking websites allow people to connect with others who share common interests, and provide a forum for interaction between them. They also allow people in these groups to keep track of what is happening in each others' lives.

The last few years have seen explosive growth not only in social network websites, but in applications which utilize data from these networks. Data stores and content on major social web applications are estimated to quadruple every 18 months, along with a doubling of queries per terabyte of data. This results in incredible pressure on web applications' database tiers, a pressure which will only increase in the years to come.

Database tiers at social networking sites see access patterns which are very different from typical OLTP and e-commerce applications. Whereas OLTP and e-commerce share a balanced read/write profile, social networking databases have very high read-to-write ratios. They also need to support hundreds of simultaneous user connections and majority of queries are point queries that return a few rows of a few hundred bytes, as compared to reporting queries which may scan entire databases and return hundreds of rows. Some database queries may also require joins or self joins for analysis of a graph, a particularly difficult to perform operation.

The most notable benchmarks for database TPC-C & TPC-W do not come close to representing the read/write mix and queries of social web database workloads. Similarly sysbench¹ a benchmark used in MySQL benchmarking does not represent a complex environment of a social web application.

SocialWebDB Benchmark

SocialWebDB Benchmark is comprised of a set of database queries that exercise the database functionalities similar to a social networking application. The benchmark measures three different loads.

1. A set of point queries on a read-only workload.
2. Point queries in a read-write (90-10) workload.
3. Queries with complex self join depicting a friend of friend analysis pattern.

¹ Sysbench <http://sysbench.sourceforge.net/>

The database is created with realistic schema and data distribution. The indexes accesses are with clustered, non-clustered and secondary keys. The columns are of various data types including timestamp and text columns.

Database Schema

Data population criterion:

1. All 'id' and 'fk' values must be at least 14 digits long.
2. All 'vc' values must be in the range of 10-32 characters with at least 15% of the records having length of 32 characters.
3. 80% of char values must be more than 512 characters long with at least 20% of the records having length of more than 1024 characters and 20% having record length of at least 4096 characters.

Real life cases:

- A table for storing blog posts with an un-optimized category field
- A table for storing guestbook comments for a Web 2.0 site with references to the object to which the comment belongs and the user who owns the comment
- A table for storing text which is retrieved based on a fk with a sort operation applied.
- Store edge data belonging to nodes (friends of a user)
- Store one-to-many relationships using a clustered key and no VARCHAR/Text fields.

G1 – Without Clustered Key

```
CREATE TABLE `g1` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `fk` bigint(20) unsigned NOT NULL default '0',  
  `vc` varchar(32) character set latin1 default NULL,  
  `d` timestamp NOT NULL default '0000-00-00 00:00:00',  
  `c` text character set utf8,  
  PRIMARY KEY (`id`),  
  KEY `kid` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

G1WCK – With Clustered Key

```
CREATE TABLE `g1wck` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `fk` bigint(20) unsigned NOT NULL default '0',  
  `vc` varchar(32) character set latin1 default NULL,  
  `d` timestamp NOT NULL default '0000-00-00 00:00:00',  
  `c` text character set utf8,  
  PRIMARY KEY (`fk`, `d`, `id`),  
  KEY `kid` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Generic parent/child objects with FOREIGN KEY (for joins)

```
CREATE TABLE `o` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `fk` bigint(20) unsigned NOT NULL default '0',  
  `d_u` timestamp NOT NULL default '0000-00-00 00:00:00',  
  `d_c` timestamp NOT NULL default '0000-00-00 00:00:00',  
  PRIMARY KEY (`id`),  
  KEY `fk` (`fk`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
CREATE TABLE `o2` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `o_id` bigint(20) unsigned NOT NULL default '0',  
  `d_c` timestamp NOT NULL default '0000-00-00 00:00:00',  
  `content` text character set utf8,  
  `keywords` text character set utf8,  
  PRIMARY KEY (`id`),  
  FOREIGN KEY `o_id` (`o_id`) REFERENCES o(id) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

CMS - Generic Content Management System (Multiple BLOBS)

```
CREATE TABLE `cms1` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `fk` bigint(20) unsigned NOT NULL default '0',  
  `d` timestamp NOT NULL default '0000-00-00 00:00:00',  
  `abstract` text character set utf8,  
  `content` text character set utf8,  
  `keywords` text character set utf8,  
  PRIMARY KEY (`id`),  
  KEY `fk` (`fk`,`d`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Edge Schema

```
CREATE TABLE `e` (  
  `id` int(11) NOT NULL default '0',  
  `fid` int(11) NOT NULL default '0',  
  PRIMARY KEY (`id`,`fid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Average relationship between id & fid is 60.

Data Distribution

| Data Distribution | CMS1 | G1WK | O | O2 | G1 | e |
|--------------------|-------|-------|-------|-------|------|------------------------|
| No. of rows | 252 M | 100 M | 120 M | 120 M | 53 M | 900 M |
| Clustered key rows | | 0.8 M | | 120 M | 53 M | 15 M (unique id) |

| | | | | | | |
|----------------------------|-------|-------|-------|-------|------|-------|
| Secondary key distribution | 2 M | | 40 M | | | |
| DB Size | 48 GB | 24 GB | 10 GB | 26 GB | 9 GB | 53 GB |

Read Only Benchmark

Queries

```

SELECT * FROM cms1 WHERE fk = ? order by d limit 10
SELECT * FROM g1wck WHERE fk = ? limit 10
SELECT * FROM cms1 WHERE id = ?
SELECT * FROM o JOIN o2 ON o.id=o2.o_id WHERE o.fk = ?

```

Client Workload

- Increase thread concurrency up to a point where thread thrashing occurs.
- Each test should start with cold start and all buffers cleared.
- The benchmark should run for fixed (800,000) number of queries.

90-10 Read/Write Benchmark

Queries

```

SELECT * FROM cms1 WHERE fk = ? order by d limit 10
SELECT * FROM g1wck WHERE fk = ? limit 10
SELECT * FROM cms1 WHERE id = ?
SELECT * FROM o JOIN o2 ON o.id=o2.o_id WHERE o.fk = ?
UPDATE g1 SET vc=?,d=?,c=? WHERE id=? limit 1
UPDATE cms1 SET abstract=?,content=?,keywords=? WHERE id=?
INSERT INTO o SET fk=?,d_c=?
DELETE from g1 where id=?

```

Client Workload

- Increase thread concurrency up to a point where thread thrashing occurs.
- Each test should start with cold start and all buffers cleared.
- The benchmark should run for fixed (4,000,000) number of queries.

Complex Join Benchmark

Queries

```
SELECT * FROM e e1 JOIN e e2 ON e1.fid=e2.id JOIN e e3 ON e2.fid=e3.id  
WHERE e1.id= ? AND e3.fid= ?
```

```
SELECT * FROM e e1 JOIN e e2 ON e1.fid=e2.id JOIN e e3 ON e2.fid=e3.id JOIN  
e e4 ON e3.fid=e4.id WHERE e1.id= ? AND e4.fid= ?
```

Client Workload

- Increase thread concurrency up to a point where thread thrashing occurs.
- Each test should start with cold start and all buffers cleared.
- The benchmark should run for fixed number of queries to get a good average measure.

Performance of Virident Green Cloud Server for MySQL and Commodity MySQL Server

System Configuration

| | Virident Q300 | Commodity Server |
|-----------------------------|---|--|
| CPU | 8-cores Opteron 2.2GHz | 8-cores Opteron 2.2GHz |
| DRAM | 32 GB | 32 GB |
| Drive Arrays | 6 x SAS 15K RAID-0 | 6 x SAS 15K RAID-0 |
| Storage Class Memory | 256GB | |
| MySQL | MySQL-for-Virident 5.0.67-r14590 | MySQL 5.0.67 |
| Storage Engine | Virident Optimized InnoDB | InnoDB |
| InnoDB Config | Buffer_Pool = 1G, Virident_Buffer_Pool = 64G | Buffer_Pool = 24G, innodb_flush_method = O_DIRECT |

Virident Server performance relative to Commodity Server

