MySQL vs SQLite performance testing

To compare the performance of MySQL vs SQLite, a quick sample database was made based off of a previous PingER SQL database schema. Existing converted data was available based on PingER results from December, 2001.

Summary

SQLite performs at the same or better speed vs. MySQL when dealing with "single-threaded" activty, i.e. only one process reading and writing to the database. SQLite's locking must be taken into account when inserting into a table, however. Further tests on concurrent access are in order.

Testing environment

Dell desktop with single core non-hyperthreading Intel Pentium 4 2.8 ghz, 1gb RAM, local SATA storage, Red Hat Enterprise Desktop 4u5 (2.4.21), MySQL 4.1.20 running locally, SQLite 3.3.4.

The shell `time` command is used below to measure the amount of processing involved with each task. Unfortunately, it cannot capture the CPU use of the MySQL daemon processes that are handling the other end of the client-server connection. Elapsed time must be used to estimate run time. The machine being used for testing was not being used for anything else at the time the tests were run.

Table structure

MySQL table structure

```
DROP TABLE IF EXISTS "metaData";
CREATE TABLE "metaData" (
  "metaID" bigint(20) NOT NULL,
  "ip_name_src" varchar(52) NOT NULL default '',
  "ip_name_dst" varchar(52) NOT NULL default '',
  "transport" varchar(10) NOT NULL default '',
  "packetSize" smallint(6) NOT NULL default '0',
  "count" smallint(6) NOT NULL default '0',
  "packetInterval" smallint(6) default NULL,
  "deadline" smallint(6) default NULL,
  "ttl" smallint(6) default NULL,
  PRIMARY KEY  ("metaID"),
  UNIQUE ("ip_name_src","ip_name_dst","packetSize","count")
);

DROP TABLE IF EXISTS data_200112;
CREATE TABLE data_200112 (
  metaID bigint(20) NOT NULL default 0,
  minRtt float default NULL,
  meanRtt float default NULL,
  medianRtt float default NULL,
  maxRtt float default NULL,
  timestamp bigint(12) NOT NULL default 0,
  minIpd float default NULL,
  meanIpd float default NULL,
  maxIpd float default NULL,
  duplicates tinyint(1) default NULL,
  outOfOrder tinyint(1) default NULL,
  clp float default NULL,
  iqrIpd float default NULL,
  lossPercent float default NULL,
  rtts text,
  seqNums text,
  PRIMARY KEY  (metaID,timestamp),
  KEY meanRtt (meanRtt,medianRtt,lossPercent,meanIpd,clp)
);
```

SQLite table structure

Only minor changes were required for SQLite compliance. SQLite does not support referential integrity so foreign key definitions must be removed and it does not support constraint names or indexes defined in the table definition with the 'KEY' or 'INDEX' keywords.

```
DROP TABLE IF EXISTS "metaData";
CREATE TABLE "metaData" (
  "metaID" bigint(20) NOT NULL,
  "ip_name_src" varchar(52) NOT NULL default '',
  "ip_name_dst" varchar(52) NOT NULL default '',
  "transport" varchar(10) NOT NULL default '',
  "packetSize" smallint(6) NOT NULL default '0',
  "count" smallint(6) NOT NULL default '0',
  "packetInterval" smallint(6) default NULL,
  "deadline" smallint(6) default NULL,
  "ttl" smallint(6) default NULL,
  PRIMARY KEY  ("metaID")
);

DROP TABLE IF EXISTS data_200112;

CREATE TABLE "data_200112" (
  "metaID" bigint(20) NOT NULL default 0,
  "minRtt" float default NULL,
  "meanRtt" float default NULL,
  "medianRtt" float default NULL,
  "maxRtt" float default NULL,
  "timestamp" bigint(12) NOT NULL default 0,
  "minIpd" float default NULL,
  "meanIpd" float default NULL,
  "maxIpd" float default NULL,
  "duplicates" tinyint(1) default NULL,
  "outOfOrder" tinyint(1) default NULL,
  "clp" float default NULL,
  "iqrIpd" float default NULL,
  "lossPercent" float default NULL,
  "rtts" text,
  "seqNums" text,
  PRIMARY KEY  ("metaID","timestamp")
);
```

Note: some excessive quoting around field names was removed from some of the statements, but both engines appear to support DDL with or without it.

Database structure creation times

Database creation was quite fast, although MySQL reasonably involves more overhead.

MySQL DDL timing

```
jaredg@atreides > time mysql -u perftest perftest < metadata_create_mysql.sql
0.005u 0.002s 0:00.02 0.0%      0+0k 0+0io 0pf+0w

jaredg@atreides > time mysql -u perftest perftest < data_create_mysql.sql
0.008u 0.001s 0:00.01 0.0%      0+0k 0+0io 0pf+0w
```

SQLite DDL timing

```
jaredg@atreides > time sqlite3 perftest.db < metadata_create.sql
0.001u 0.002s 0:00.00 0.0%      0+0k 0+0io 0pf+0w

jaredg@atreides > time sqlite3 data.db < data_create_sqlite.sql
0.000u 0.002s 0:00.01 0.0%      0+0k 0+0io 0pf+0w
```

Data insert times

210 rows were inserted into the metaData table and 325,720 rows were inserted into the data_200112 table. The sample data consists of a month's worth of PingER results from 22 sources to between 4 and 9 destinations.

The data was formatted in traditional SQL style single INSERT statements (rather than the compound multiple-row INSERT format MySQL supports).

MySQL's elapsed time was actually longer than SQLite.

Sample insert data for data_200112 table

```
INSERT INTO data_200112 (metaID, minRtt, meanRtt, medianRtt, maxRtt, timestamp, minIpd, meanIpd, maxIpd,
duplicates, outOfOrder, clp, iqrIpd, lossPercent, rtts, seqNums) VALUES (390,19,21,NULL,31,1007194529,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,10,NULL,NULL);
INSERT INTO data_200112 (metaID, minRtt, meanRtt, medianRtt, maxRtt, timestamp, minIpd, meanIpd, maxIpd,
duplicates, outOfOrder, clp, iqrIpd, lossPercent, rtts, seqNums) VALUES (390,19,22,NULL,49,1007196329,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,0,NULL,NULL);
INSERT INTO data_200112 (metaID, minRtt, meanRtt, medianRtt, maxRtt, timestamp, minIpd, meanIpd, maxIpd,
duplicates, outOfOrder, clp, iqrIpd, lossPercent, rtts, seqNums) VALUES (390,19,20,NULL,21,1007198127,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,0,NULL,NULL);
INSERT INTO data_200112 (metaID, minRtt, meanRtt, medianRtt, maxRtt, timestamp, minIpd, meanIpd, maxIpd,
duplicates, outOfOrder, clp, iqrIpd, lossPercent, rtts, seqNums) VALUES (390,19,19,NULL,20,1007199920,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,0,NULL,NULL);
INSERT INTO data_200112 (metaID, minRtt, meanRtt, medianRtt, maxRtt, timestamp, minIpd, meanIpd, maxIpd,
duplicates, outOfOrder, clp, iqrIpd, lossPercent, rtts, seqNums) VALUES (390,19,20,NULL,23,1007201718,NULL,NULL,
NULL,NULL,NULL,NULL,NULL,0,NULL,NULL);
```

MySQL insert DML timing

```
jaredg@atreides > time mysql -u perftest perftest < metadata_data.sql
0.006u 0.004s 0:00.05 0.0%      0+0k 0+0io 0pf+0w

jaredg@atreides > time mysql -u perftest perftest < data_insert.sql
6.063u 4.518s 0:53.76 19.6%     0+0k 0+0io 0pf+0w
```

SQLite insert DML timing

For SQLite it was ncessary to explicitly enable an exclusive lock during the length of the insert process. This was done by starting the entire series of insert statements with `begin exclusive transaction;` and ending with `commit;`. Without that, SQLite acquires and then releases a lock on the database file using a semaphore file on the file system with each insert statement. This is very costly – a very small number of rows were inserted in a minute without exclusive locking, while the entire dataset could be inserted in under 60 seconds with it enabled.

Exclusive locking was not used during the metaData inserts.

```
jaredg@atreides > time sqlite3 perftest.db < metadata_data.sql
0.077u 0.098s 0:01.16 13.7%     0+0k 0+0io 0pf+0w

jaredg@atreides > time sh sqlite_load.sh
42.961u 2.456s 0:50.38 90.1%    0+0k 0+0io 0pf+0w
```

Contents of sqlite_load.sh - begin_txn.sql and commit.sql contain the SQL statements described above

```
cat begin_txn.sql data_insert.sql commit.sql | sqlite3 data.db
```

Query timing

Queries run

```
select m.ip_name_src, min(minRtt)
from metaData m join data_200112 d on m.metaID = d.metaID
where m.ip_name_dst like '%slac.stanford.edu'
  and m.ip_name_src not like '%stanford.edu'
  and d.minRtt is not null and d.minRtt > 0
group by m.ip_name_src
order by m.ip_name_src;

select m.ip_name_dst, min(minRtt), max(maxRtt)
from metaData m join data_200112 d on m.metaID = d.metaID
where m.ip_name_src like '%bnl.gov'
  and d.maxRtt is not null
group by m.ip_name_dst
order by m.ip_name_dst;

select m.ip_name_dst, min(minRtt), max(maxRtt), avg(meanRtt)
from metaData m join data_200112 d on m.metaID = d.metaID
where m.ip_name_src like '%hep.net'
and timestamp between 1008403200 and 1008435600
group by m.ip_name_dst
order by m.ip_name_dst;
```

MySQL timings

```
jaredg@atreides > time mysql -u perftest perftest < queries.sql
... returned data omitted ...
0.006u 0.002s 0:01.68 0.0%      0+0k 0+0io 0pf+0w
```

SQLite timings

```
jaredg@atreides > time sh sqlite_queries.sh
... returned data omitted ...
0.644u 0.063s 0:00.74 94.5%     0+0k 0+0io 0pf+0w
```