Loading...

Partitioning MySQL database with high load solutions

2010-11-19 00:35

Write comment

We need to test if MySQL can be reliable with huge data sets, talking of a daily waterfall of about 30

million INSERTs and a few daily analityc big SELECTs too.
So what's the deal ? which database engine ? which platform ? which architecture ? ... and is data secured ? accessible ?
Before choosing MySQL 5.1, i turned around these three databases:

- **Oracle :**
  Known as the 'most reliable' database engine in the world, personally i don't believe in that, and i didn't choose it for its cost (Especially that :)).
- **PostgreSQL :**
  A good database engine, it succeeded where older than 5.0 versions of MySQL failed, but now, both engines are mature enough and i'm switching between them on every new projects depending on the specific needs of each project architectures.
  Switching between database engines should be a simple task when the application deploys an abstraction layer on the database.
  So why not PostgreSQL for this project ?
  1. PostgreSQL (the latest stable version to this day) still lacks on partitioning and provide very basic features compared to what MySQL do.
  2. As i know, PostgreSQL has no 'enterprise' version and no paid support, with mission critical projects, we cant rely on the community respondness to resolve issues, we need enterprise-level SLA.
- **MySQL 5.1**
  Refering to the new features list of the 5.1 version of MySQL, i felt in love with it.
  Good partitioning (and sub-partitioning), better replication engine .. and the 'enterprise' version is making us comfortable with critical-mission applications.

As for scalability, making a database respond in less than 1 second when stressed with up to 2000 INSERT/sec throughput (with TRIGGER) does not rely on System sizing and Database engine only, these things wont scale if the design is poorly done.

Let's dive into the blue:
We have a sell reporting analytic application, we have about 20 million sells per day, each selling action has a Seller and a Buyer as long as the date and the amount of the bill.
The informations are provided in flat text files, so we use Talend ETL to load the data to the database.
On the output side, we need to get a fast access to these informations:

- **(1)** Best sellers of each day having total amount of sells more than a defined limit. (Using the sum of the billing amounts of the day)
- **(2)** Occasionally fetch selling details of a Seller for a specific period.

I've deployed two tables, the first is for delivering best sellers/buyers informations **(1)**, the second is for the selling details **(2)**.
Here's the first one:

```
CREATE TABLE `transaction` (
`seller` int(11) DEFAULT NULL,
`buyer` int(11) DEFAULT NULL,
`amount` decimal(5,2) DEFAULT 0,
`sell_date` date DEFAULT NULL,
`sell_time` time DEFAULT NULL,
PRIMARY KEY(`seller`, `buyer`, `sell_date`, `time`)
) ENGINE=InnoDB
PARTITION BY RANGE ( MONTH(`sell_date`))
SUBPARTITION BY HASH ( `seller` div 1000000)
SUBPARTITIONS 10
(
PARTITION p1 VALUES LESS THAN (1) ENGINE = InnoDB,
PARTITION p2 VALUES LESS THAN (2) ENGINE = InnoDB,
```

```
PARTITION p3 VALUES LESS THAN (3) ENGINE = InnoDB,
PARTITION p4 VALUES LESS THAN (4) ENGINE = InnoDB,
PARTITION p5 VALUES LESS THAN (5) ENGINE = InnoDB,
PARTITION p6 VALUES LESS THAN (6) ENGINE = InnoDB,
PARTITION p7 VALUES LESS THAN (7) ENGINE = InnoDB,
PARTITION p8 VALUES LESS THAN (8) ENGINE = InnoDB,
PARTITION p9 VALUES LESS THAN (9) ENGINE = InnoDB,
PARTITION p10 VALUES LESS THAN (10) ENGINE = InnoDB,
PARTITION p11 VALUES LESS THAN (11) ENGINE = InnoDB,
PARTITION p12 VALUES LESS THAN MAXVALUE ENGINE = InnoDB
);
```

This table will contain the sell transactions, I've made partitioning by seller and month of the transaction in the way to facilitate queries for selling details for a specific seller and period: **(2)**.

Here's the second one (dedicated to queries fetching best sellers of each day **(1)**):

```
CREATE TABLE `daily_sell_amount_summary` (
`seller` int(11) DEFAULT NULL,
`amount_summary` decimal(5,2) DEFAULT 0,
`summary_date` date DEFAULT NULL,
PRIMARY KEY(`seller`, `summary_date`)
) ENGINE=InnoDB
PARTITION BY RANGE ( DAY(`summary_date`) )
SUBPARTITION BY HASH (   `seller` div 1000000 )
SUBPARTITIONS 10
(
PARTITION p_day_of_year_1 VALUES LESS THAN (1) ENGINE = InnoDB,
PARTITION p_day_of_year_2 VALUES LESS THAN (2) ENGINE = InnoDB,
PARTITION p_day_of_year_3 VALUES LESS THAN (3) ENGINE = InnoDB,
PARTITION p_day_of_year_4 VALUES LESS THAN (4) ENGINE = InnoDB,
PARTITION p_day_of_year_5 VALUES LESS THAN (5) ENGINE = InnoDB,
PARTITION p_day_of_year_6 VALUES LESS THAN (6) ENGINE = InnoDB,
PARTITION p_day_of_year_7 VALUES LESS THAN (7) ENGINE = InnoDB,
PARTITION p_day_of_year_8 VALUES LESS THAN (8) ENGINE = InnoDB,
PARTITION p_day_of_year_9 VALUES LESS THAN (9) ENGINE = InnoDB,
PARTITION p_day_of_year_10 VALUES LESS THAN (10) ENGINE = InnoDB,
PARTITION p_day_of_year_11 VALUES LESS THAN (11) ENGINE = InnoDB,
PARTITION p_day_of_year_12 VALUES LESS THAN (12) ENGINE = InnoDB,
PARTITION p_day_of_year_13 VALUES LESS THAN (13) ENGINE = InnoDB,
PARTITION p_day_of_year_14 VALUES LESS THAN (14) ENGINE = InnoDB,
PARTITION p_day_of_year_15 VALUES LESS THAN (15) ENGINE = InnoDB,
PARTITION p_day_of_year_16 VALUES LESS THAN (16) ENGINE = InnoDB,
PARTITION p_day_of_year_17 VALUES LESS THAN (17) ENGINE = InnoDB,
PARTITION p_day_of_year_18 VALUES LESS THAN (18) ENGINE = InnoDB,
PARTITION p_day_of_year_19 VALUES LESS THAN (19) ENGINE = InnoDB,
PARTITION p_day_of_year_20 VALUES LESS THAN (20) ENGINE = InnoDB,
PARTITION p_day_of_year_21 VALUES LESS THAN (21) ENGINE = InnoDB,
PARTITION p_day_of_year_22 VALUES LESS THAN (22) ENGINE = InnoDB,
PARTITION p_day_of_year_23 VALUES LESS THAN (23) ENGINE = InnoDB,
PARTITION p_day_of_year_24 VALUES LESS THAN (24) ENGINE = InnoDB,
PARTITION p_day_of_year_25 VALUES LESS THAN (25) ENGINE = InnoDB,
PARTITION p_day_of_year_26 VALUES LESS THAN (26) ENGINE = InnoDB,
PARTITION p_day_of_year_27 VALUES LESS THAN (27) ENGINE = InnoDB,
PARTITION p_day_of_year_28 VALUES LESS THAN (28) ENGINE = InnoDB,
PARTITION p_day_of_year_29 VALUES LESS THAN (29) ENGINE = InnoDB,
PARTITION p_day_of_year_30 VALUES LESS THAN (30) ENGINE = InnoDB,
PARTITION p_day_of_year_31 VALUES LESS THAN (31) ENGINE = InnoDB,
PARTITION p_day_of_year_rest VALUES LESS THAN MAXVALUE ENGINE = InnoDB
);
```

Having about 7 million active sellers per day, this table will contain the summary amounts of each seller, with partitioning based on the date column, we'll have each day in a single partition.

Sells older than 3 months are purged, we wont get data for more than 3 months, so a partition will contain a maximum of 3 days.

The table above is updated on each sell transaction, we use the TRIGGER below to get things done:

```
CREATE TRIGGER summarize_amount AFTER INSERT ON `transaction`
FOR EACH ROW BEGIN
INSERT INTO `daily_sell_amount_summary`(`seller`, `amount_summary`,  `summary_date`)
VALUES (NEW.`seller`, NEW.`amount`, 0, NEW.`sell_date`)
ON DUPLICATE KEY
UPDATE `amount_summary`=`amount_summary`+NEW.`amount`;
END;
```

Used hardware:

HP Itanium2 Double Core, 1.2Ghz each.
8GB RAM.
2 SCSI U320 15K Disks, 146Gb each (No RAID is set).
Linux Debian Etch

Some software tuning:

- Multi-threaded ETL (with 10 concurrent MySQL connections).
- Linux is installed on the first disk only (sdb1) with classic EXT3 filesystem.
- InnoDB is lonely hosted on the second disk (sdb2) relying on an XFS filesystem.

Here's my MySQL configuration file:

```
[client]
port = 3306
socket = /tmp/mysql.sock

[mysqld]
skip-name-resolv
pid-file = /var/run/mysqld/mysqld.pid
datadir = /opt/database << This is XFS
port = 3306
socket = /tmp/mysql.sock
back_log = 50
bind-address = 0.0.0.0
max_connections = 100
max_connect_errors = 10
table_cache = 2048
max_allowed_packet = 16M
binlog_cache_size = 1M
max_heap_table_size = 64M
sort_buffer_size = 8M
join_buffer_size = 8M
thread_cache_size = 8
thread_concurrency = 8
query_cache_size = 64M
query_cache_limit = 2M
ft_min_word_len = 4
#default_table_type = InnoDB
thread_stack = 192K
transaction_isolation = REPEATABLE-READ
tmp_table_size = 64M
log-bin=mysql-bin
long_query_time = 2
log_long_format
```

```
# *** For debugging purpose, but not for the prod.
log_slow_queries

# *** Replication related settings
server-id = 1

#*** MyISAM Specific options
key_buffer_size = 2048M
read_buffer_size = 1024M
read_rnd_buffer_size = 1024M
bulk_insert_buffer_size = 1024M
myisam_sort_buffer_size = 1024M
myisam_max_sort_file_size = 40G
myisam_max_extra_sort_file_size = 40G
myisam_repair_threads = 6
myisam_recover

# *** INNODB Specific options ***
innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 6G
innodb_data_file_path = ibdata1:10M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 16
innodb_flush_log_at_trx_commit = 0
innodb_log_buffer_size = 8M
innodb_log_file_size = 256M
innodb_log_files_in_group = 3
innodb_log_group_home_dir = /opt/database << This is XFS
innodb_max_dirty_pages_pct = 90
innodb_flush_method=O_DIRECT
innodb_lock_wait_timeout = 120
innodb_file_per_table=1
innodb_log_group_home_dir = /opt/database/ << This is XFS
innodb_data_home_dir = /opt/database

[mysqldump]
quick
max_allowed_packet = 16M

[mysql]
no-auto-rehash

[isamchk]
key_buffer = 512M
sort_buffer_size = 512M
read_buffer = 8M
write_buffer = 8M

[myisamchk]
key_buffer = 512M
sort_buffer_size = 512M
read_buffer = 8M
write_buffer = 8M

[mysqlhotcopy]
interactive-timeout

[mysqld_safe]
open-files-limit = 8192
```

MySQL was compiled with these options:

```
./configure --prefix=/usr/local/mysql --enable-assembler --with-plugin-partition --
with-plugin-innobase --with-big-tables --with-mysqld-user=mysql --enable-large-files
```

Monitoring is done with Munin since it's very simple to install and has no heavy footprint on the system resources.

Results:
1. Reads:

- Query:

  SELECT seller, amount_summary FROM daily_sell_amount_summary WHERE summary_date='2008-05-10' AND amount_summary>1200;

- Table has more than 60 million rows, and 6 million just for the summary_date='2008-05-10'.

Results are from 6 seconds to 22 seconds depending on the system load (sometimes i do SELECTs while my ETL is stressing the base with triggered INSERTs.

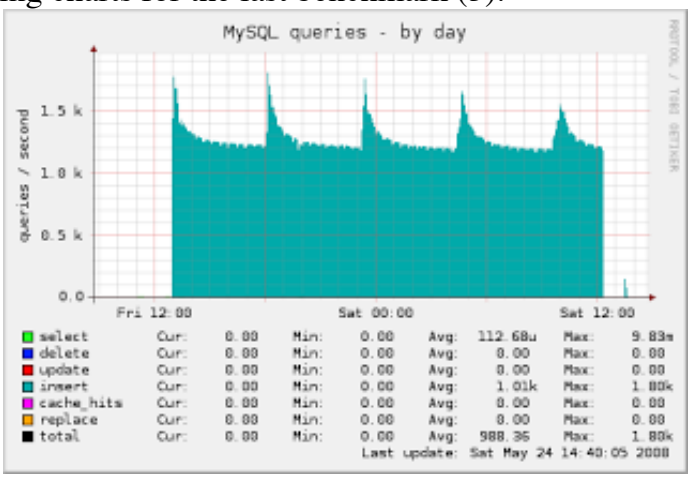
2. Writes:

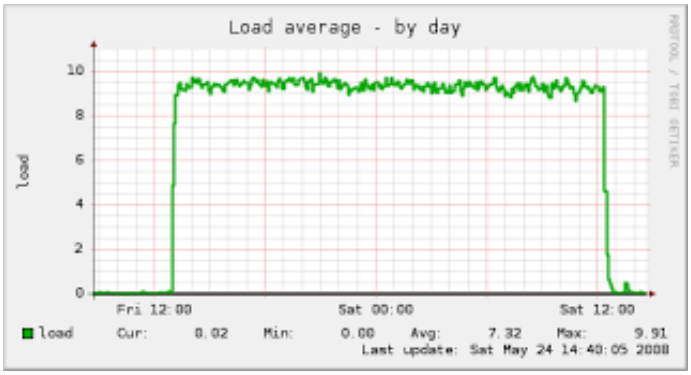
- Using multi-threaded Java ETL from Talend Open Studio.
- The ETL is ran from my Laptop (Centrino Duo 1.7Ghz, 2Gigs Memory with Ubuntu 8.04)
- Simple INSERTs in the *transaction* table, 10 concurrent threads are INSERTing.

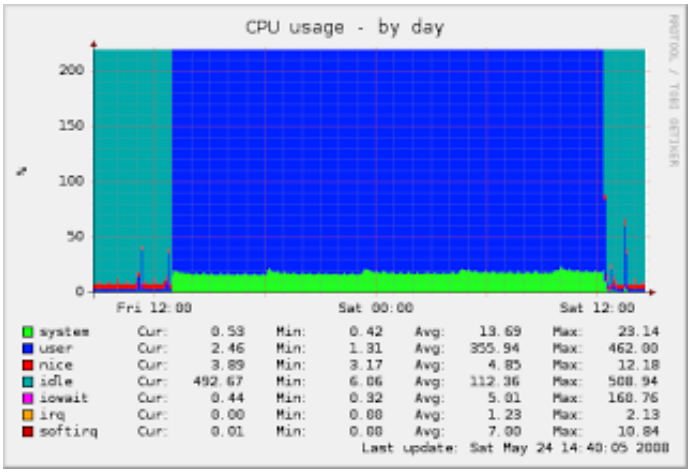
Results:

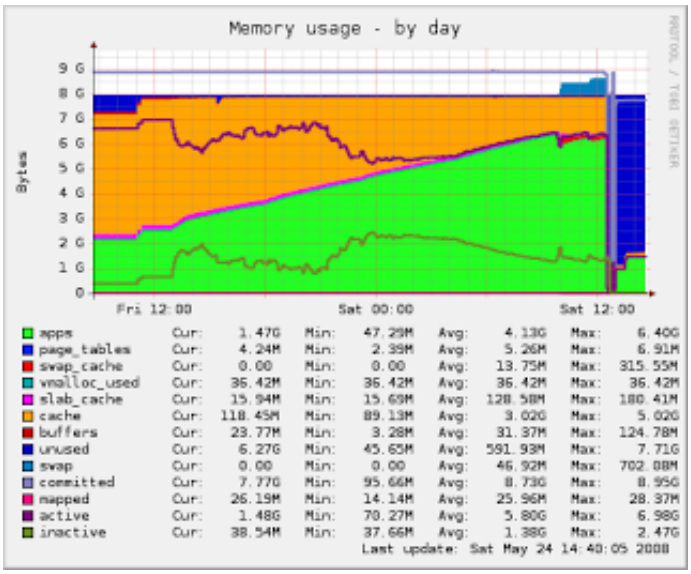
1. **MyISAM engine** with **one** ETL thread gave us about **1200 INSERT/second**.
2. **InnoDB engine** with **one** ETL thread gave us about **400 INSERT/second**.
3. **InnoDB engine** with **10** threads ETL gave us around **1400 INSERT/second**.

Here's the system monitoring charts for the last benchmark (3):









Some will ask why I've insisted on InnoDB engine when MyISAM did a higher performance, i should say that I'm sticking with it for Security and Concurrency, i think these worth the performance, especially with a mission-critical project.
InnoDB doesn't only lack with performance, hot backup is provided under paid license, no free tools to do hotbackuping, LVM snapshots are a solution for this.

You can get more power when hosting the database on a RAID 10 disk array, this will be covered in a later post, so stay tuned !

We provide MySQL consulting and engineering for robust and scalable solutions, don't hesitate to contact us if you need any of our professional services.

mysql stresstest talend partitioning

Similar posts

Réplication MySQL en Master/Master avec partage de charge
Dans cet article nous allons essayer de mettre en place une solution qui assure la redondance d'une base de données MySQL et qui permet de partager la charge entre deux nœuds.
Things to keep in mind when doing an ETL job
So what if we had a water flow of some thousands of files per hour ... ? can you estimate how much memory will the ETL eat ? that's huge ...

10 Comments

1. 

Justin Swanhart
2011-05-09 01:03:21

Hi, Take a look at http://flexvie.ws. It supports summary tables which are updated from the binary log. This would remove the triggers (and potential deadlocks they create) with an lightweight binary log reader. LVM backup is great, but performance tests have shown up to 6x performance difference between when a snapshot is being read from and when no snapshot exists. Consider xtrabackup (http://www.percona.com) for a hot backup with less impact. Note: I'm the author of Flexviews and I work for Percona, but I think both of these tools would be helpful in your environment.

---

2. 

Zardosht Kasheff
2010-12-07 16:07:44

I am interested in seeing how the TokuDB storage engine can do on this workload without partitions. Based on the requirements written here, I wonder if TokuDB's fast indexed insertions and perhaps a few minor tweaks to the schema, one can achieve the same performance or better without needing partitions. Would you be interested in sharing the data so that we can try some simple experiments?

---

3. 

Dimitri
2010-11-26 10:42:31

Forgot to ask - did you try MySQL 5.5 ?? - it'll be GA before the end of the year and have way better performance comparing to 5.1 and I'm curious what will be the difference in performance on your workload.. Rgds, -Dimitri

---

4. 

Dimitri
2010-11-26 10:38:47

You are using "innodb_flush_log_at_trx_commit = 0" setting, means all your transactions are unsafe.. - is it intentional?.. Also, seems the choice of XFS was important for you, but did you run any other tests to compare XFS vs others? (ext3, ext4, etc..) Rgds, -Dimitri

---

5. 

Fourat Zouari

2010-11-25 14:23:15

I've already written a good way to do hotbackup using LVM snapshoting :here. It is a filesystem-level snapshotting.

Comments: **5/**
10
1 2 Next> ...»

Send your comment

Comment
Name
Email
Number Verification 
Type the characters you see in the picture below
Try a new code
Send comment

Tag cloud

java employment jboss jee linux ant webservice jaxws jax-ws wsdl wsi cluster soa talend mysql esb jbdev bpel wsbpel bpm

Popular Posts


Partitioning MySQL database with high load solutions

2010-10-18 02:21:56


Building a BPEL proccess with Netbeans BPEL Designer (part 1)

2010-10-25 23:58:47

Using hotbackup on Linux with LVM

2010-10-07 14:34:19

Load balancing JBoss and Apache2 / mod_jk

2010-10-23 13:03:31

Réplication MySQL en Master/Master avec partage de charge

2010-10-15 17:07:43

Popular posts: **5/**
19
1 2 3 4 Next> ...»

blog archive

- 2011 (2)
  - November (1)
    - Administrateurs systèmes Linux
  - March (1)
    - Check out SMGS v2.6 HowTos on YouTube Channel

- 2010 (17)
  - November (3)
    - A Java hello-world application on Google App Engine (part 2)
    - Partitioning MySQL database with high load solutions
    - JBoss SOA Platform: How to create a file listener ESB project
  - October (14)
    - Building a BPEL proccess with Netbeans BPEL Designer (part 1)
    - Développement d'un web service sur JBossWs (Bottom-Up)
    - Load balancing JBoss and Apache2 / mod_jk
    - A Java hello-world application on Google App Engine (part 1)
    - Things to keep in mind when doing an ETL job
    - Réplication MySQL en Master/Master avec partage de charge
    - Web Services, JAX-WS et intéropérabilité (partie 2)
    - Web Services, JAX-WS et intéropérabilité (partie 1)
    - Ingénieur qualité
    - Getting rid of "java.lang.OutOfMemoryError: PermGen space" exception on jboss
    - Symfony / PHP5 software engineer
    - Using 'Ant' the 'batch' way
    - JBoss Seam / Java EE software engineer
    - Using hotbackup on Linux with LVM

Employment

Interested in taking your passion and experience to the next level ?
Jobs at Tritux.

Services

We are providing high-quality professional services for achieving scalable enterprise-class solutions.
Take a look at our services page.

Products

We design robust, innovative, flexible and ergonomic products.
Take a look at our products overview page.

9 Rue du Niger, Bloc H, Etage 4

1002 Mont Plaisir, Tunis / Tunisia

Phone: +216 **71 90 31 40**
Fax: +216 **71 90 86 44**