Practical MySQL Tuning

UKOUG Conference 2011
December 5th, Birmingham

Oli Sennhauser
Senior MySQL Consultant, FromDual
oli.sennhauser@fromdual.com
FromDual provides neutral and independent:

- Consulting for MySQL (on-site and remote)
- Remote-DBA / MySQL operations
- Support for Galera (synchronous MySQL Replication)
- Support for MySQL (Basic and Silver)
- Training for MySQL
- Consulting Partner of Open Database Alliance (ODBA.org)
- Oracle Silver Partner (OPN)
- More information at: www.fromdual.com
Our customer
Content

- FromDual Performance Balance
- Critical Resources
- Hardware / OS
- Database parameter
- Application-Tuning
- Architecture & Design
- Measuring
The FromDual way of PT

- Whom of you has or had Performance problems with MySQL?
- Whom of you has a systematic way for Performance Tuning?

→ Many roads lead to Rome!

- Starting point: Customer cries because he has a Performance Problem!
FromDual Performance Balance
Collecting facts

- How does the problem look like?
  - DB is suddenly, sometimes or since ever slow?
  - What exactly is slow?
  - New version was released?
  - Somebody has tinkered around?
  - We are short before going live and still much too slow?
- Have we historical performance data?
- Best case if:
  - You can (at will) simulate the problem
  - Predict the problem or if it happens periodic.
  - Find the pattern!
Critical resources
Hardware
OS
Critical resources

- Find the bottle-neck / the limiting resource:
  - Fortunately „only“:
    - CPU
    - Memory (RAM)
    - I/O (IOPS, throughput)
    - Network (FpS, throughput)

- But how? → find the critical resource!
Measure: CPU

- **top**

<table>
<thead>
<tr>
<th>Cpu0</th>
<th>Cpu1</th>
<th>Cpu2</th>
<th>Cpu3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1%us, 12.8%sy, 0.0%ni, 71.4%id, 1.5%wa, 0.0%hi, 7.2%si, 0.0%st</td>
<td>16.5%us, 3.4%sy, 0.0%ni, 79.4%id, 0.0%wa, 0.0%hi, 0.7%si, 0.0%st</td>
<td>99.8%us, 0.1%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.1%si, 0.0%st</td>
<td>8.5%us, 2.3%sy, 0.0%ni, 58.5%id, 28.2%wa, 2.3%hi, 0.2%si, 0.0%st</td>
</tr>
</tbody>
</table>

- **vmstat**

  ```
  # vmstat 1
  procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
  r  b  swpd free buff  cache si so  bi  bo in  cs us  sy  id  wa
  1  0  96148  56096  35936  548792  0  0  0  656  379  343  5 38  57  0
  0  0  96148  56096  35936  548792  0  0  0  260  357  5 34  61  0
  0  0  96148  56096  35936  548792  0  0  0  306  399  9 29  62  0
  3  0  96148  49192  35940  549808  0  0  1020  0  289  431  91  4  3  2
  1  0  96148  47424  35944  551572  0  0  896  0  310  378  98  2  0  0
  1  0  96148  45656  35944  553344  0  0  896  0  260  359  98  1  0  1
  2  0  96148  43948  35944  555112  0  0  896  0  280  355  97  3  0  0
  1  0  96148  42056  35952  556884  0  0  904  0  260  374  99  0  0  1
  1  0  96148  40288  35984  558672  0  0  896  3772  312  398  97  3  0  0
  1  0  96148  38520  35984  560424  0  0  896  0  259  365  97  1  0  2
  ```

- **or mpstat**

- **Which process is using CPU?**
Measure: Memory (RAM)

- **free / top:**

  ```
  #free
  total        used       free       shared      buffers       cached
  Mem:        1036016  983864    52152        0     35484   547432
  -/+ buffers/cache: 400948   635068
  swap:        4202112   96148  4105964
  ```

- **ps**

  ```
  # ps -eo user,pid,%cpu,%mem,vsz,rsz,comm --sort -vsz | \ egrep 'mysql|COMMAND'
  USER   PID %CPU %MEM VSZ   RSZ COMMAND
  mysql 1361  0.0  1.5 108368 16444 mysqld
  mysql 1210  0.0  0.1   4536  1956 bash
  mysql 1289  0.0  0.1   4060  1444 safe_mysqld
  mysql 1204  0.0  0.1   4048  1404 su
  ```
Measure: I/O

- **vmstat**

```bash
# vmstat 1
procs ---swap--- -----io----- ----cpu-----
 r  b  si  so  bi  bo  us  sy  id  wa
0  0  3  3  94 143 21 21 56  2
0  0  0  0  0  4  9  37 54  0
```

- **iostat** *(→ sysstat package)*

```bash
# iostat -x 1
avg-cpu: %user  %nice  %system  %iowait  %steal  %idle
      5.88    0.00    34.31    2.94    0.00    56.86

Device: r/s  w/s  rkB/s  wkB/s  await  svctm  %util
hda  0.00  0.00  0.00  0.00  0.00  0.00  0.00
hdc  0.00  2.94  0.00  23.53 14.67  12.00  3.53
```

- **pidstat**
Measure: Network

- **dstat**

```
# dstat

---total-cpu-usage---- -dsk/total- -net/total- ---paging--- ---system---

usr sys idl wai hig sig read writ recv send in out int cs

21 6 56 2 0 14 25k 39k 0 0 7648 880B 129 762
19 2 55 0 0 34 0 0 262B 1630B 0 0 297 374
6 2 59 0 0 33 0 0 1075B 1467B 0 0 284 372
8 3 54 5 1 29 0 0 1048B 884B 0 0 309 377
14 2 54 0 1 29 0 0 3479B 3669B 0 0 333 362
18 5 47 1 0 29 0 0 2800B 3632B 0 0 351 2257
30 69 0 0 0 1 0 0 1807B 1181B 0 0 651 243k
24 74 2 0 0 0 0 0 2380B 2183B 0 0 685 240k
```

- **watch** `-d` `-n` `1` `'ifconfig'`

- Frames per second (80k – 1.5M) / throughput (1 Gbit/s → 120 Mbyte/s)?
Summary

- **CPU**
  - Which process
  - How many cores?
    → often SQL queries

- **Memory**
  - Which process
  - Swapping?
    → Over- or Under-allocating of DB caches!

- **I/O**
  - Throughput or IOPS
  - which device?
  - Random or sequential I/O
  - Read or write
    → Caches to small, tmp tables?

- **Network**
  - Errors / Drops?
  - Throughput
  - FpS
Set screws

- CPU
  - Faster cores?
  - More Cores

- Memory
  - More RAM?

- I/O
  - RAID-5 :-(
  - SAN :-(
  - RAID-10, many spindles, SSD?
  - Battery buffered I/O cache!

- Network
  - 1 Gbit

- OS
  - New 64-bit kernel
  - XFS
    - I/O Scheduler
      - noop
      - Deadline

- Virtualization :-(

www.fromdual.com
Database Tuning
MySQL Tuning

- Which Storage Engine are you using at the moment?
- Which MySQL version? (→ 5.1 and newer)
- At the moment: about 330 MySQL parameter
  → but only 8 (9) of those are primarily significant!
  → Rough-Tuning
- All other parameters only after detailed benchmarking...
  → Fine-Tuning
InnoDB Rough-Tuning

- `innodb_buffer_pool_size`
  - about 80% of RAM on a dedicated server
  - `SHOW STATUS LIKE 'Innodb_buffer_pool_pages%'`;

- `innodb_log_file_size`
  - Big = faster, but longer recovery times → 2 x 256 M

- `innodb_flush_log_at_trx_commit`
  - 0, 2 for top performance, 1 for reliability

- `sync_binlog`
  - `!= 0` → often very slow
MySQL Rough-Tuning

- **key_buffer_size**
  - about 25 – 33% of RAM on dedicated machine
  - `SHOW STATUS LIKE 'Key_blocks_%';`
- **table_open_cache**
  - Running connections x used tables → 2 – 4k is not unusable!
    See `Open[ed]_tables`.
- **table_definition_cache**
  - See `Open[ed]_table_definitions` → 512 – 4096 is not unusable!
- **query_cache_type/query_cache_size**
  - Do not make too big (<= 128 M), bad for very high concurrency!
Further aid

- How to measure?
  - `SHOW GLOBAL STATUS;`
  - `SHOW ENGINE INNODB STATUS\G`
- ca. 330 variables
- ca. 310 Status informations
- MySQL Database Health Check:
- MySQL Docu, Server Status Variables:
- That's it, so far...! :)

www.fromdual.com
Application-Tuning
Application-Tuning

- Index Tuning
  - Primary Key (InnoDB) → Length of secondary indices!
  - Remove (partial) redundant Indices
  - Remove Indices with a low cardinality!?!?
- Query Tuning
  - `SHOW PROCESSLIST;`
  - Slow Query Log (since 5.1 dynamically)
  - `log_queries_not_using_indexes = 1`
  - `mysqldumpslow -s t slow.log > slow.log.profile`
  - `EXPLAIN SELECT ...`
EXPLAIN

• Is the key to the truth
• Query Tuning: x times faster is possible!

EXPLAIN SELECT * FROM test where name = 'Oli';

<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
<th>rows</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIMPLE</td>
<td>test</td>
<td>ALL</td>
<td>last</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>261369</td>
<td></td>
</tr>
</tbody>
</table>

• EXPLAIN Output Format:

Application-Tuning

- Let it go!
  - Remove columns which are not used (or `char(0)` or fill with `NULL`!)
  - Old data → delete (archive, swap out!)
- Schema Tuning
  - `mysqldump --no-data > structure_dump.sql`
  - Right data types and right length!
    - `int(1) → 4 byte int!`
  - `utf8 → only when needed`
  - `NULL OR NOT NULL`
- Locality of the data
  - InnoDB Primary Key
  - V-Partitioning / (H-)Partitioning
Architecture & Design
MySQL Scale-Out vs Scale-Up

Scale-Up
- Costs
- MySQL Design
- Physical bottle-necks

Scale-Out
- „Relaxation of Constraints“
The MySQL Scale-Out approach

Application

Master

Slave 1

Slave 2

Slave 3

Load balancer

ro

rw

Slave Reporting

Slave Backup

...
Architectonic possibilities

- RDBMS are a slow technology!
  - Hence: Cache (hot data into a cache!)
  - MySQL Memcached Plugin / Memcached
  - HandlerSocket
  - MySQL Cluster
- Abstraction layers (ORM, Frameworks, etc.)
  - Fast development process but
  - Standard == „non optimized“ → bad for performance!
- BLOB's are bad for RDBMS
  - Put on a filer
  - Blob Streaming (PBMS)
- Sharding / distributing
  - manually
  - Spider SE
  - MySQL Replication (r/w Traffic Split)
  - MySQL Cluster
  - Synchronous Replication with Galera
Monitoring
What happens, if...?

• Customer asks: Can my system bear 30% more load?

• Chemical Engineering:

  ![Diagram](image)

• Are there differences to a DB based system?

• What do I need to answer this question?
Measure, measure, measure...

- Measure
- Simulate → Capacity planning
The circle closes...

- In ideal case: Only ONE change at a time!
Questions?

Discussion?

We have some time for face to face questions...