MariaDB / MySQL Stolperfallen
und wie komme ich da wieder raus?

GNU / LinuxDay im Vorarlberg 2020, Dornbirn (Austria)

Oli Sennhauser

Senior MariaDB und MySQL Berater, FromDual GmbH

https://www.fromdual.com/presentations

Über FromDual GmbH

www.fromdual.com

Support

Beratung

remote-DBA

Schulung

Inhalt

Stolperfallen

- All – Admins und Entwickler
- Dev – Entwickler
- Adm – Admins

All: MySQL ≠ MariaDB !

- Up to version 5.5: ~100% "drop-in-replacement" (2010 – 2015)
- Version 10.0 – 10.2 / 5.6 – 5.7: only "compatible" (2013 – 2017)
- Since version 10.3 / 8.0: "different" RDBMS (but still similar) (since 2018)

- MariaDB vs. MySQL ≈ Percona Server (99.99% compatible)

- Imitations
  - Amazon Aurora – MySQL (RDS) – "fully compatible"
  - "TiDB is fully compatible with the MySQL 5.7 protocol and the common features and syntax of MySQL 5.7"
  - TDSQL (Tencent) – "Compatible with syntax elements and access API"

- Strategy
  - Is the application "certified" against my fork?
  - Decision: Support both or only one fork (which one???)?
  - Very good testing!

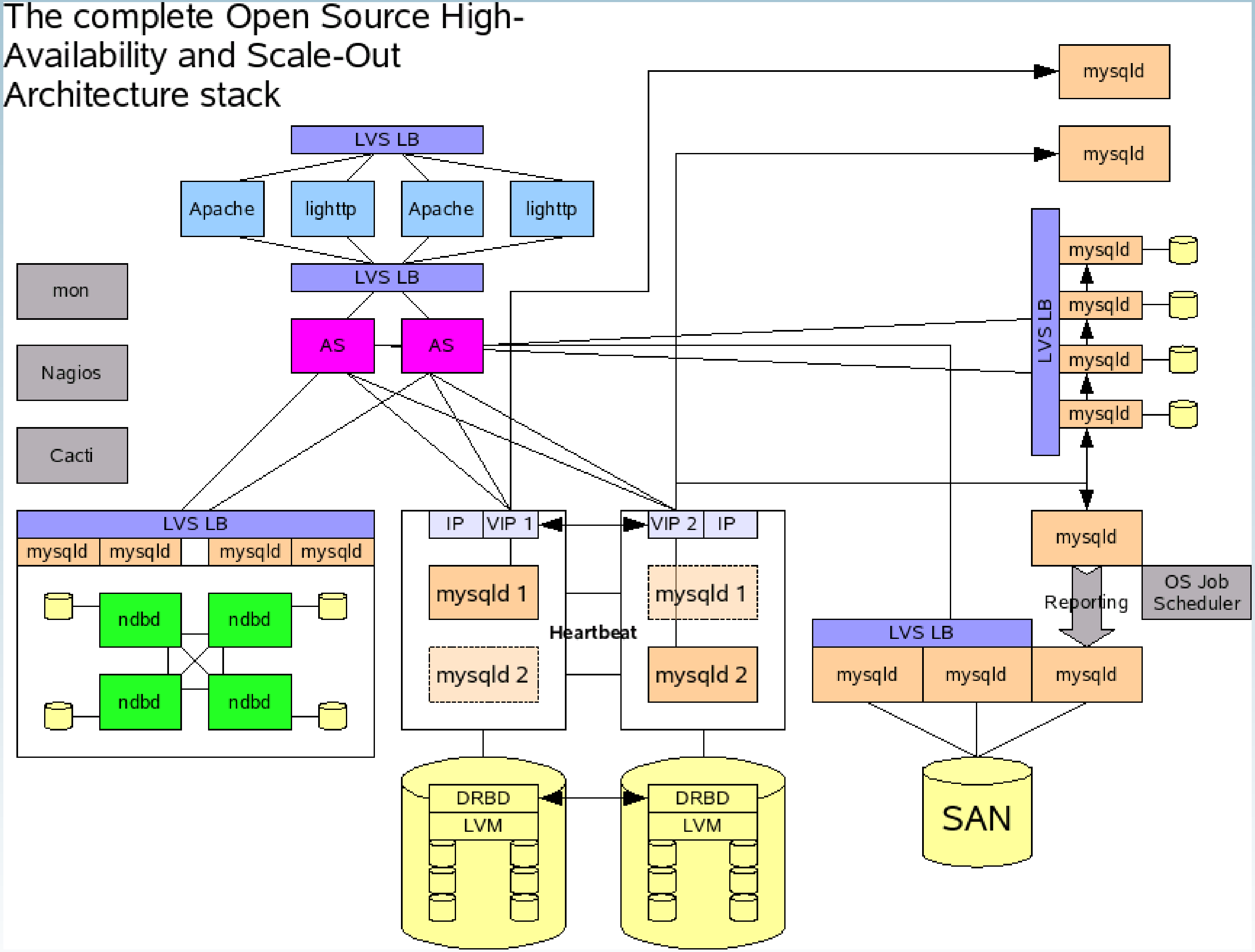
Consequences in code:

```
// MySQL/Percona 5.7 ff.
if ( (($aInstanceInfo['branch'] == 'MySQL')
   || ($aInstanceInfo['branch'] == 'Percona')
     )
  && ($aInstanceInfo['mr_version'] >= '050700') )
{

  // MySQL 5.7.0 - 5.7.99
  if ( ($aInstanceInfo['mr_version'] >= '050700')
    && ($aInstanceInfo['mr_version'] <= '050799'))
  {
    ...
  }
  // MySQL 8.0 DOES support this feature but has different tables!
  elseif ( ($aInstanceInfo['mr_version'] >= '080000') ) {
    ...
  }
} // MySQL + Percona 5.7 ff.
// MariaDB and MySQL 5.6 and older
else {

  // MariaDB 10.5 and newer
  if ( $aInstanceInfo['mr_version'] >= '100500') {
    ...
  }
  else {
    ...
  }
} // MariaDB
```

All: KISS

- "Keep It Stupid Simple"
  - Design principle by the U.S. Navy, 1960 and others (Wikipedia)
- "Most systems work best if they are kept simple rather than made complicated"
- Complicated solutions will break sooner or later!
- Caution: Software vendors, IT consultants and paper tigers have opposite goals!
- Strategy:
  - Try to make/keep it simple. Where and why do errors happen?
  - Simplicity should be a key goal in design and unnecessary complexity should be avoided
  - Reflect the requirements! Can we reduce or adapt the requirements?
  - Take your time to make a sketch (map) to get an overview!

All: non-Kiss


The complete Open Source High-Availability and Scale-Out Architecture stack
LVS LB
Apache
lighttp
Apache
lighttp
LVS LB
mon
Nagios
Cacti
AS
AS
mysqld
mysqld
LVS LB
mysqld
mysqld
mysqld
mysqld
LVS LB
mysqld
mysqld
mysqld
mysqld
ndbd
ndbd
ndbd
ndbd
IP
VIP 1
VIP 2
IP
mysqld 1
mysqld 1
Heartbeat
mysqld 2
mysqld 2
DRBD
LVM
DRBD
LVM
mysqld
Reporting
OS Job Scheduler
LVS LB
mysqld
mysqld
mysqld
SAN

Dev: MyISAM table locking

- MyISAM was default until about 2013!
  - Old and legacy
  - "Never change a running system" :-(
- Problem wit MyISAM:
  - NOT crash-safe!!! → Possibility of corrupt data!
  - Table level locks! → Does NOT scale!
  - No transactions... (business problem not DB)
- Non obvious: Also reads are affected
- Symptoms: `PROCESSLIST`:
  - `"Waiting for table level lock"`
- Strategy:
  - Think about migrating to InnoDB.

MyISAM table locking

```
Time State                        Info
6    Sending data                 select UserIndex, LastUsedShopLanguage from webshop_user2 wh
6    Waiting for table level lock update webshop_user2 set DataChange=1, LastDataChange=NOW()
6    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=317249 a
5    Waiting for table level lock update webshop_user2 set DataChange=1, LastDataChange=NOW()
5    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=1631890
5    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=695182 a
5    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=1583223
5    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=969880 a
5    Waiting for table level lock select Password, UserIndex, SubShop, MainSubShop, State, Adv
3    Waiting for table level lock select Password, UserIndex, SubShop, MainSubShop, State, Adv
3    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=1661429
3    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=654008 a
2    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=795808 a
2    Waiting for table level lock select UserIndex from webshop_user2 where UserIndex=1654024
2    Waiting for table level lock select Password, UserIndex, SubShop, MainSubShop, State, Adv
2    Waiting for table level lock select UNIX_TIMESTAMP(LastDataImport) from webshop_user2 whe
```

Adm: Big `ibdata1`

- `ibdata1` is called the System Tablespace
  - Today it contains only "system" data
- Earlier it also contained "user" data
  - Controlled by `innodb_file_per_table = {on | off}`
  - Default to on since MariaDB / MySQL 5.5
  - So no more user data in there...?
- Why `ibdata1` can still grow / is still big?
  - Historical user data???
  - Long (read) and/or big (write) transactions

    → many UNDO log segments (ROLLBACK)
  - Big InnoDB Temporary Tables
- Strategy:
  - Small transactions (anyway a good idea from DB point of view!)
  - UNDO Tablespaces (10.0 / 5.6)
  - InnoDB Temporary Tablespaces (10.2 / 5.7)
  - System Tablespace NEVER shrinks!!!
  - Otherwise only: dump / restore (TTS is too complicated!?!)

All: Noisy Neighbours

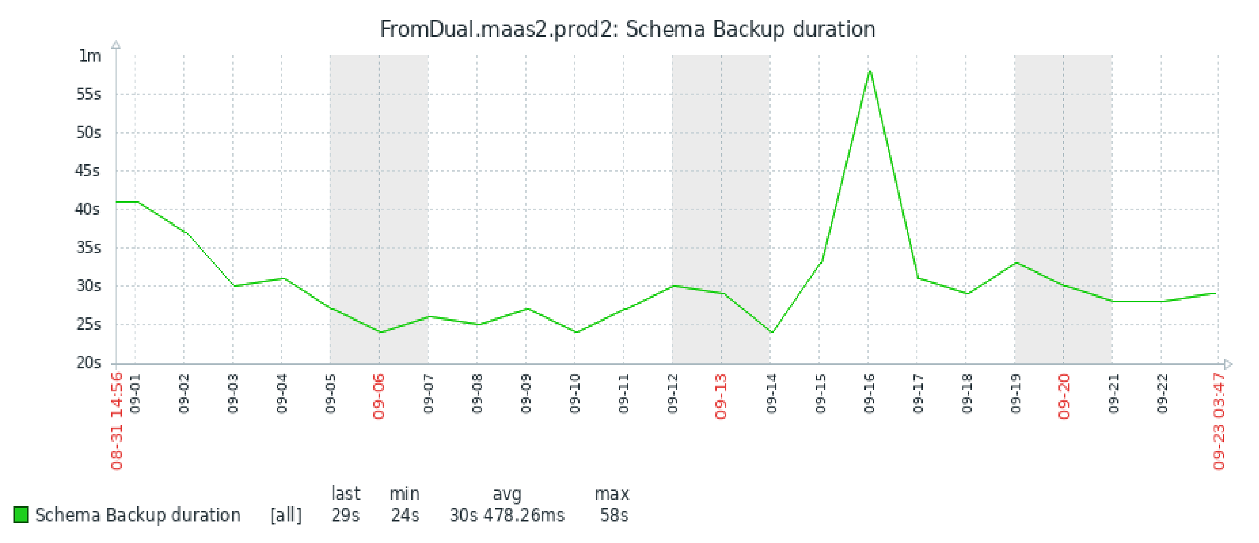
- Problem:
  - Shared and consolidated resources (SAN, network, VM, Container, many connections, ...)
  - "Others" use "our!" resources
  - → "We" become slow or even get outages...

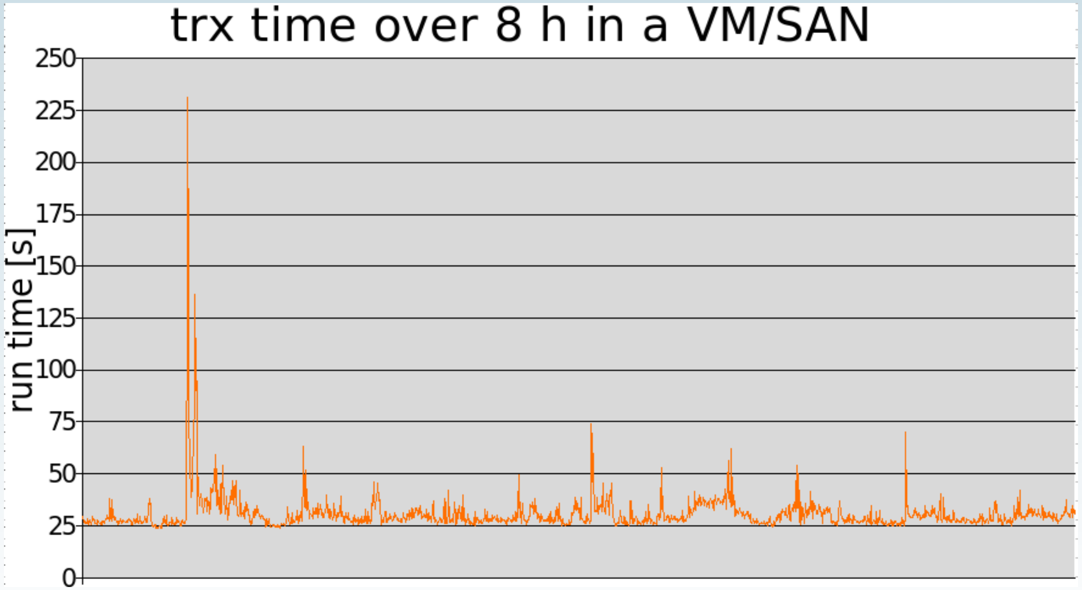
- Examples:
  - Backup times deviate
  - Cloud ping times go suddenly up
  - Unforeseen outages/freezes
  - I/O becomes slow

- Strategy:
  - Do NOT share (egoistic).
  - Monitor well!
  - Know the "big picture"!
  - Consider, that you are not alone.

Image by Free-Photos on Pixabay

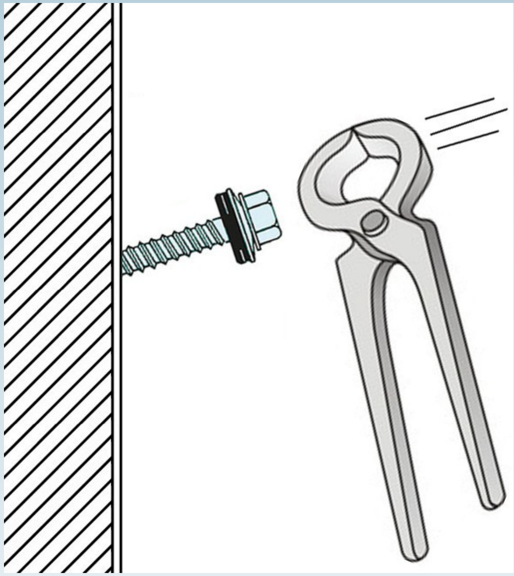
All: Noisy Neighbours





Dev: No BLOBs in a RDBMS

- One big red database vendor claimed:
  - "You can store now videos in the database!"
  - I also found it cool once... :-(
- Problem:
  - Query latency higher and
  - Database throughput lower
  - Backup size and time will increase
  - DDL operations will take ages
  - Buffer Pool (cache) is flooded
- Strategy:
  - Do NOT store BLOBs in the RDBMS but on a filer (Disk, NAS, SAN). They are made for this task.



Adm: Disk full!

- Disk full is more or less the worst you can do to your database!
  - DB will stuck
  - Worst case you get corruptions
  - Difficult to make it work again because of lack of free disk space
  - I/O systems become slower if they are filled more than 90% (SAN, SSD, ...)
- How to find who is the evil?
  - `df -h; du -hsc; ls -lSr`
- Who is the evil:
  - Mostly Binary Logs (or Relay Logs)
  - Temporary Tables (`/tmp`?)
  - DB files (`*.ibd`)
  - Other Logs (error, slow, general → `FLUSH LOGS` after `rm`!)
  - `ibdata1` not so common any more nowadays
- Strategy:
  - Monitor your disk space!
  - Try to predict how long it will take until disk is full.

Dev: Locking

- Locking is caused by the application and is NOT a database problem!!!
  - So search there and not in the database.
  - Your DBA can help you finding your problem...
- Different type of locks:
  - Explicit locks (`LOCK TABLE...`)
  - MyISAM Table Level Locks (write AND read!)
  - Metadata Locks (DDL vs. DML + `SELECT`)
  - InnoDB Row Level Locks (and special case: Deadlock)
  - Galera Cluster Lock (DDL with TOI (default))
  - Galera Cluster Conflict (looks like a deadlock but is not).
- Strategy:
  - Learn more about locking
  - Avoid: `LOCK TABLE` and MyISAM
  - Be careful with DDL operations
  - Do fast and small DML operations (smaller chunks and Query Tuning)

Adm: "Crash"

- There is an inflationary use of the term "Crash"!
- What we have seen declared as a crash:
  - Lost connection
  - Killed connection (`connection_timeout`)
  - Stalled system caused by Locks (MyISAM, Query Cache)
  - Refused connections
  - Properly shutdown database (see error log)
  - Killed DB server (Oom Killer) → can be considered as a crash (syslog)
  - You hit a Bug with Stacktrace → this is a real crash! (see error log)
- Strategy:
  - Find the real cause of the symptom
  - Check MariaDB / MySQL error log (stop sequence, start sequence, stack)
  - Check syslog (Oom killer: Who eats all your memory?)
  - Possibly InnoDB Buffer Pool or `max_connections` are too big!
  - Or you hit a bug.

Adm: Oom and Restart

```
Syslog:

...
Out of memory: Kill process 1904 (mysqld) score 39 or sacrifice child
Killed process 1904 (mysqld) total-vm:2855024kB, anon-rss:449640kB, file-rss:0kB
...

NO Shutdown sequence in MariaDB / MySQL error log (kill -9)
```

```
MariaDB / MySQL error log:

[Note] bin/mysqld (initiated by: unknown): Normal shutdown
...
[Note] bin/mysqld: Shutdown complete

--

Version: '10.0.19-MariaDB'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  MariaDB Server
/usr/sbin/mysqld (mysqld 10.0.19-MariaDB) starting as process 4469 ...
...
[Note] Server socket created on IP: '::'.
[Note] bin/mysqld: ready for connections.
```

Real Crash with Stacktrace

```
111208 14:35:05 [ERROR] mysqld got signal 11 ;

This could be because you hit a bug. It is also possible that this binary
or one of the libraries it was linked against is corrupt, improperly built,
or misconfigured. This error can also be caused by malfunctioning hardware.
We will try our best to scrape up some info that will hopefully help diagnose
the problem, but since we have already crashed, something is definitely wrong
and this may fail.

...

Thread pointer: 0x0
Attempting backtrace. You can use the following information to find out
where mysqld died. If you see no messages after this, something went
terribly wrong...
stack_bottom = (nil) thread_stack 0x48000
./bin/mysqld(my_print_stacktrace+0x2e) [0xa1195e]
./bin/mysqld(handle_segfault+0x3f6) [0x6340a6]
/lib64/libpthread.so.0 [0x3248e0eb10]
./bin/mysqld(ma_checkpoint_execute+0x42d) [0x83bead]
./bin/mysqld(ma_checkpoint_background+0x23f) [0x83c9af]
/lib64/libpthread.so.0 [0x3248e0673d]
/lib64/libc.so.6(clone+0x6d) [0x32482d40cd]
```

All: Tables w/o PK

- Tables without a Primary Key are
  - Bad design in a relational model
  - Possibly worse latency and throughput
  - Causes Slave lagging on `UPDATE` and `DELETE`
  - Prevents use of Galera Cluster/InnoDB Cluster!
- Strategy:
  - Create always Primary Keys on every MariaDB / MySQL table (also in DWH / Star Model!)
  - `AUTO_INCREMENT` can be used

Dev: Server has gone away!

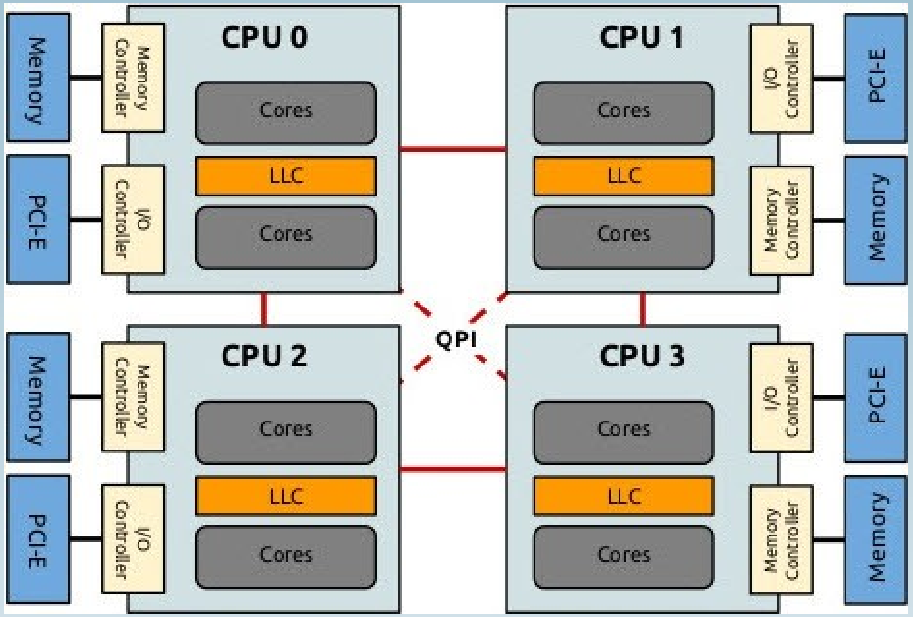
Image by FreePhotosART on Pixabay

- I was very frightened the first time!
  - Better: "Connection was terminated!"
- Symptom: Connection from client to server was terminated.
- Cause:
  - Connection was killed (by admin?)
  - Retrieved too big results (`max_allowed_packet`)
  - Server was stopped, killed or crashed
- Strategy:
  - 1. Check if admin has killed your connection (because it harmed the database)
  - 2. Check Error Log for Start/Stop or Crash messages
  - 3. Try to reproduce and possibly increase `max_allowed_packet` (typically `BLOB` or `TEXT`)

Adm: NUMA

- Problem: Big machines:

  more than 1 CPU/socket



Stolen from: blog.sqlauthority.com

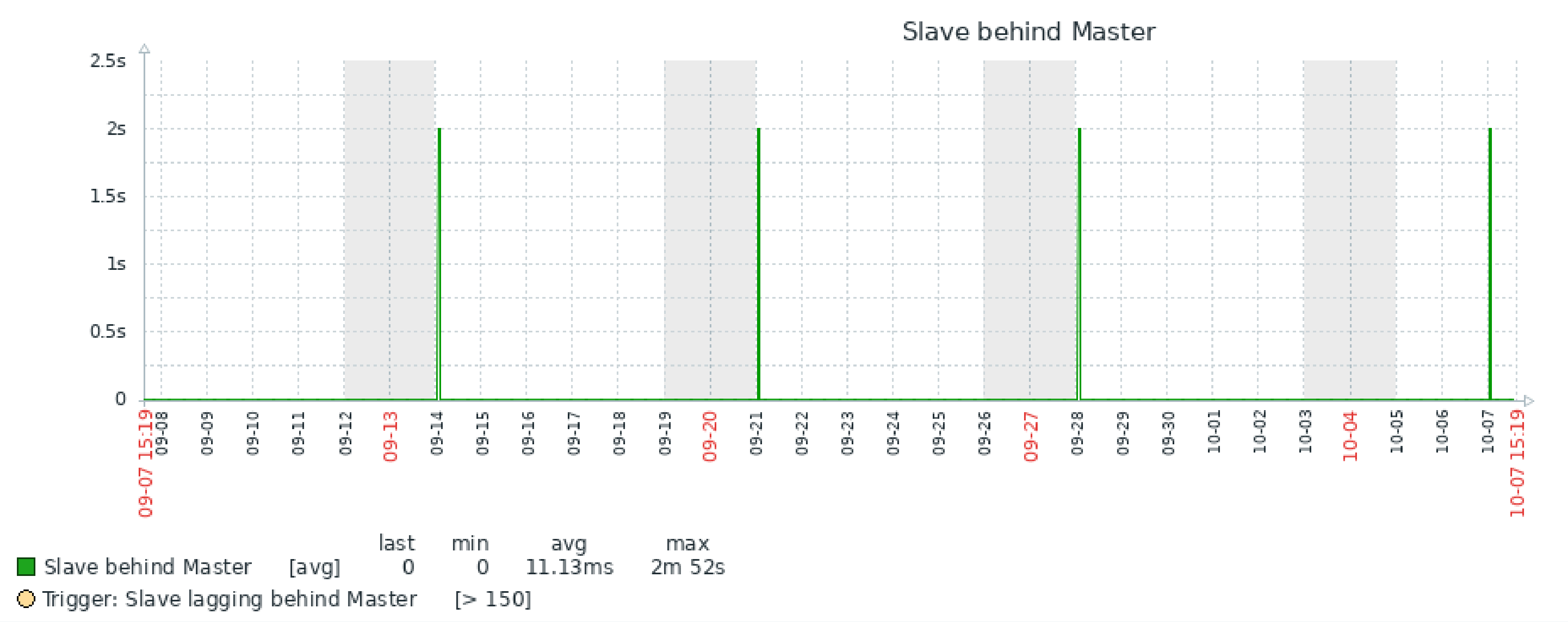
- Heavy swapping!
  - Swappiness does NOT help!
- You must tell the O/S how to behave:
- Strategy:
- Avoid such big machines! You mostly do not need them!
- `innodb_numa_interleave = ON`
- But: Not in all packages for all distributions for all versions available! :-(

All: Slave Lag

```
SQL> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
                           ...
              Slave_IO_Running: Yes
             Slave_SQL_Running: Yes
                           ...
         Seconds_Behind_Master: 999
```

- Causes:
  - Long running Query (DML or DDL)
  - Missing Primary Key (`UPDATE, DELETE`) → Long running Query
  - Weaker Slave than master (RAM, I/O, ...)?
- Strategy:
  - Do smaller transactions (not possible with DDL)
  - Upgrade MariaDB/MySQL: DDL are online or instantaneous
  - Add Primary Keys on ALL Tables
  - Configure Slave correctly and buy adequate hardware
  - Release hand break: `innodb_flush_log_at_trx_commit = 2 / sync_binlog = 0`
  - Last alternative: Parallel Replication (since MariaDB 10.0 and MySQL 5.6)

All: Slave Lag



Dev: Sending data

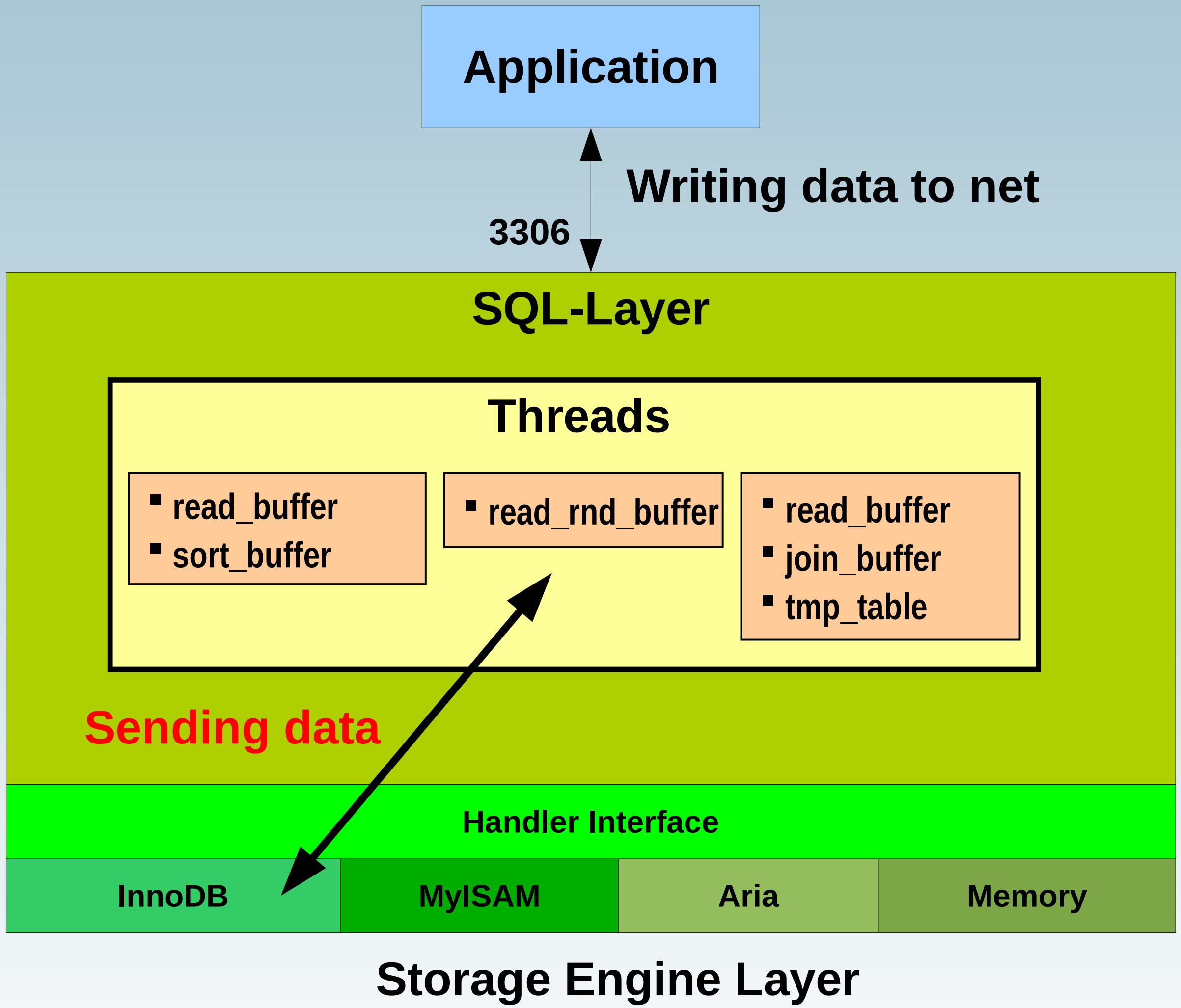
- Also interesting for Admins to find the evil!
- "`Sending data`" is NOT "`Writing to net`"

```
SQL> SHOW PROCESSLIST;
+--------+------+-----------+------+------+--------------+-------------------------------+
| Id     | User | Host      | db   | Time | State        | Info                          |
+--------+------+-----------+------+------+--------------+-------------------------------+
| 577558 | app  | 10.0.0.4  | test |   42 | Sending data | SELECT * FROM very_big_table |
+--------+------+-----------+------+------+--------------+-------------------------------+
```

- Strategies:
- → Appropriate DB Cache configuration (InnoDB Buffer Pool)

  Disk I/O?
- → Query Tuning
- → Change design or architecture
- → Or live with it (SISO principle)

Sending data

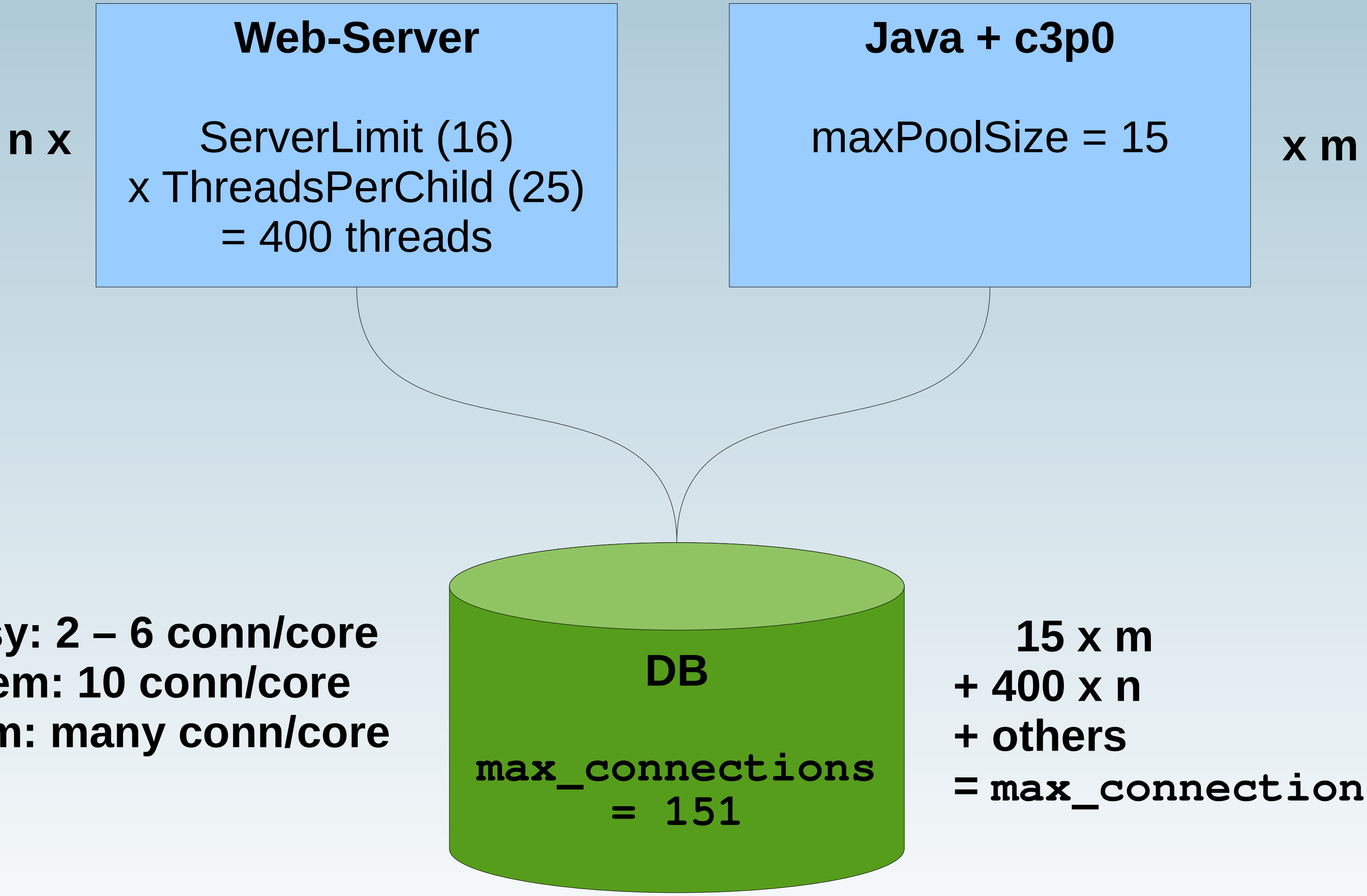


All: Too many connections

Image by Robert Allmann on Pixabay

- You reached the database limit:
  - `max_connections`
  - `max_user_connections`
  - Per Account `MAX_USER_CONNECTIONS`
- This is a fuse to protect your database!
- Cause: Locks, Long running queries, etc.
- Strategy:
  - Find the reason why you reached the limit!
  - Good starting point: `SHOW PROCESSLIST;`

All: How many connections?



n x

Web-Server

ServerLimit (16)
x ThreadsPerChild (25)
= 400 threads

Java + c3p0

maxPoolSize = 15

x m

DB

`max_connections = 151`

highly busy: 2 – 6 conn/core
busy system: 10 conn/core
idle system: many conn/core

15 x m
+ 400 x n
+ others
= `max_connections`

Dev: Simple Query Tuning

- Most seen simple errors beside missing indices:
- `WHERE col LIKE '%...%';`
  - % on the left: Index cannot be taken!
  - Avoid, change business rule or use Solr
- `WHERE UNIX_TIMESTAMP(Date) < 12454232423;`
  - Function covered column: Index cannot be taken!
  - rewrite Query: `Date < FROM_UNIXTIME(1245...)`
- `WHERE LOWER(email) = LOWER('contact@fromdual.com)`
  - MariaDB/MySQL are case INsensitive!
  - Same problem as above but possibly not solvable
  - Use indexed `GENERATED PERSISTENT` column!
- `INDEX (a, b, c) ; WHERE b = 5 AND c = 2;`
  - a filter is missing in the Query
  - Either add a filter to the query or create index on `(b,c)`

All: Galera

- Galera is cool but NOT the solution for all problems!!!
- Galera IS MariaDB / MySQL BUT behaves slightly different...
- Problems:
  - Stable Infrastructure (HW, NW, VM)
  - Operations not simple, needs drill!
  - Application must support Galera behaviour
    - → See Galera Limitations @FromDual.com
- Strategy:
  - Understand the technology!!!
  - Test application well
  - Drill your ops team, also for the bad weather scenarios.

All: Progress bar

- We rarely have one on CLI (MariaDB tries a bit)
- But how can we though see progress?
- Disk filling / file growing:
  - `watch` is your friend!
  - `df -h, ls -la, du -hsc`
- Progress on file operations:
  - Many files: `ls -la /proc/<PID>/fd/*`
  - Big file: `watch -n 1 'cat /proc/<PID>/fdinfo/<FD>'`
- Database:
  - `INSERT`: `AUTO_INCREMENT` counter or `SELECT MAX(id)` or `SELECT COUNT(*)`
  - Size of file (`*.ibd`)
  - `SHOW PROCESSLIST` → Progress (MariaDB only)
  - `SHOW GLOBAL STATUS LIKE 'Innodb_rows%ed';`
  - `INFORMATION_SCHEMA.USER_STATISTICS`
  - `PERFORMANCE_SCHEMA.status_by_thread`

Q & A

Fragen ?

Diskussion?

Wir haben Zeit für ein persönliches Gespräch...

- FromDual bietet neutral und unabhängig:
  - Beratung
  - Remote-DBA
  - Support für MariaDB, MySQL und Galera Cluster
  - Schulung

www.fromdual.com/presentations