MySQL Performance Tuning für Oracle-DBA's

DOAG Konferenz 2015, Nürnberg

Oli Sennhauser

Senior MySQL Consultant, FromDual GmbH

oli.sennhauser@fromdual.com

Über FromDual GmbH

www.fromdual.com

Support

Beratung

remote-DBA

Schulung

Inhalt

MySQL Performance Tuning

- Was ist Performance?
- Was kostet Performance?
- Tuning Massnahmen
- MySQL Konfiguration
- Wo schauen?
- Langsame Abfragen finden
- Optimiere das Query!
- Monitoring

Was ist Performance?

- Es gibt 2 verschiedene Ansichten:

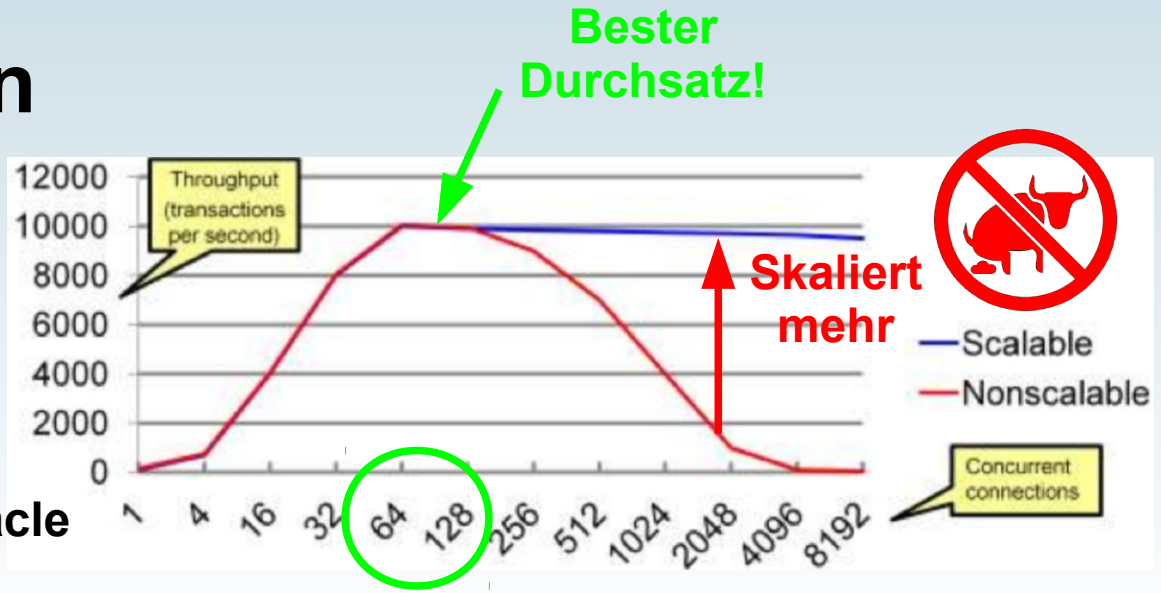
- Durchsatz
  - Wie viel Operationen kriege ich pro Zeit durch? → Anzahl
  - Abfragen pro Sekunde (DB Queries)
  - Business Operationen pro Minute (Rechnungen, Bestellungen, Einkäufe, ...)
  - Grösse: QPS, QPM, OPS

- Antwortzeit (Latenz)
  - Wie lange dauert meine Operation? → Zeit
  - Laufzeit einer SQL-Abfrage
  - Zeitdauer eines Rechnungslaufs oder eines Backups
  - Grösse: ms, s, min oder h

- Und diese beiden Ansichten sind typischerweise gegenläufig!!!

Durchsatz (throughput)

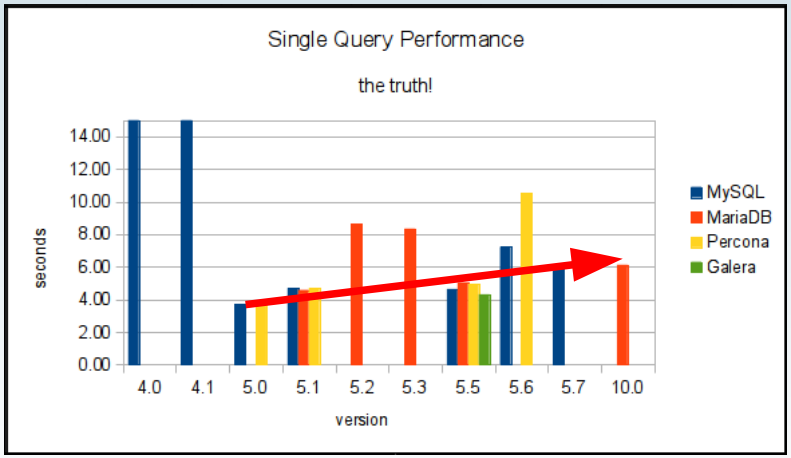
- Grad der Parallelität eines Systems
  - Concurrency, Nebenläufigkeit
  - Design und Architektur einer Applikation
- Neue DB Versionen haben tendenziell mehr Durchsatz
- Tummelplatz von Marketing!!!



Dimitri Kravtchuk, MySQL Performance Architect @ Oracle

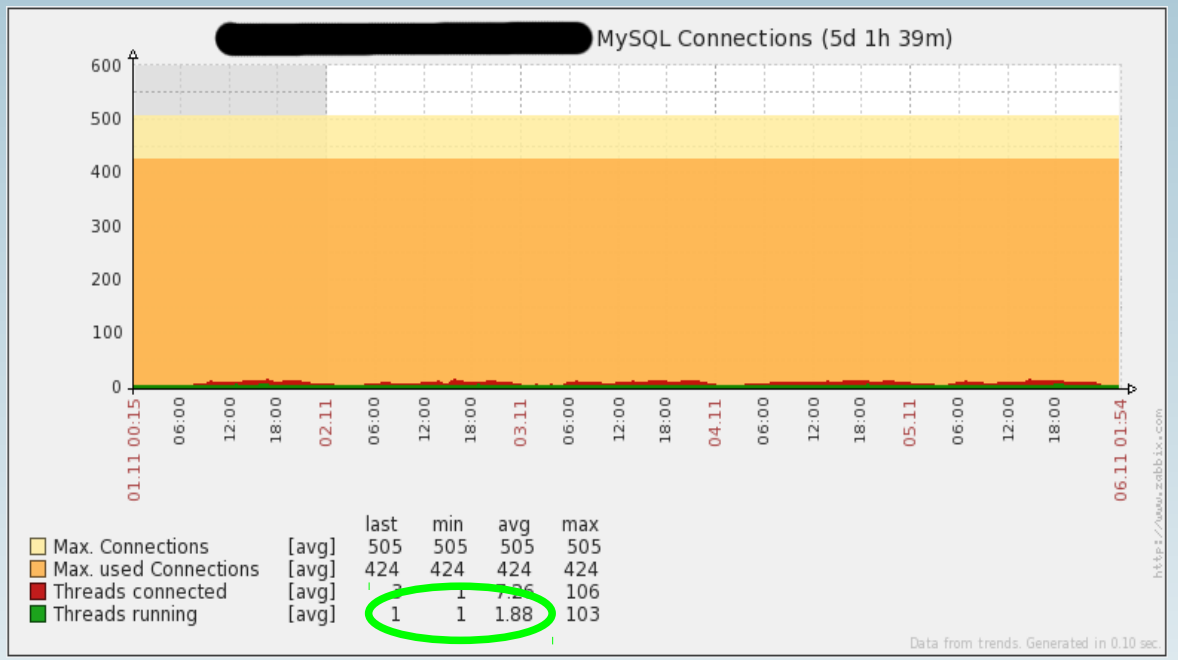
Antwortzeit (Latenz, latency)

- Hängt ab von
  - der Implementierung und
  - der Datenmenge
- Neue DB Versionen sind tendenziell langsamer!
- Durchsatz geht üblicherweise auf Kosten der Antwortzeit (mehr Codepath)
  - Wie optimal läuft mein Zeugs?
  - Tummelplatz des DBA!!!



Durchschnittlicher DB-Nutzer

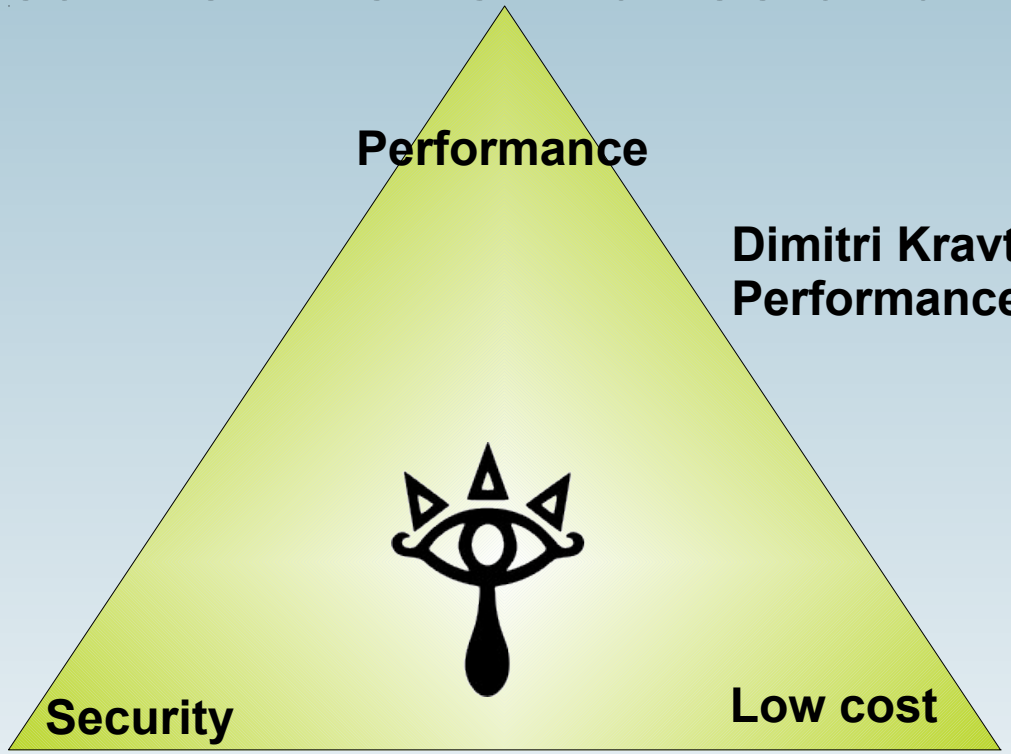
- Wo liegen nun der durchschnittliche DB-Nutzer?
- Wir sind nicht
  - Facebook,
  - Twitter,
  - LinkedIn...?



- `SHOW GLOBAL STATUS LIKE 'threads_running';`
- d.h. die Softwarehersteller arbeiten gegen uns!

Was „kostet“ Performance?

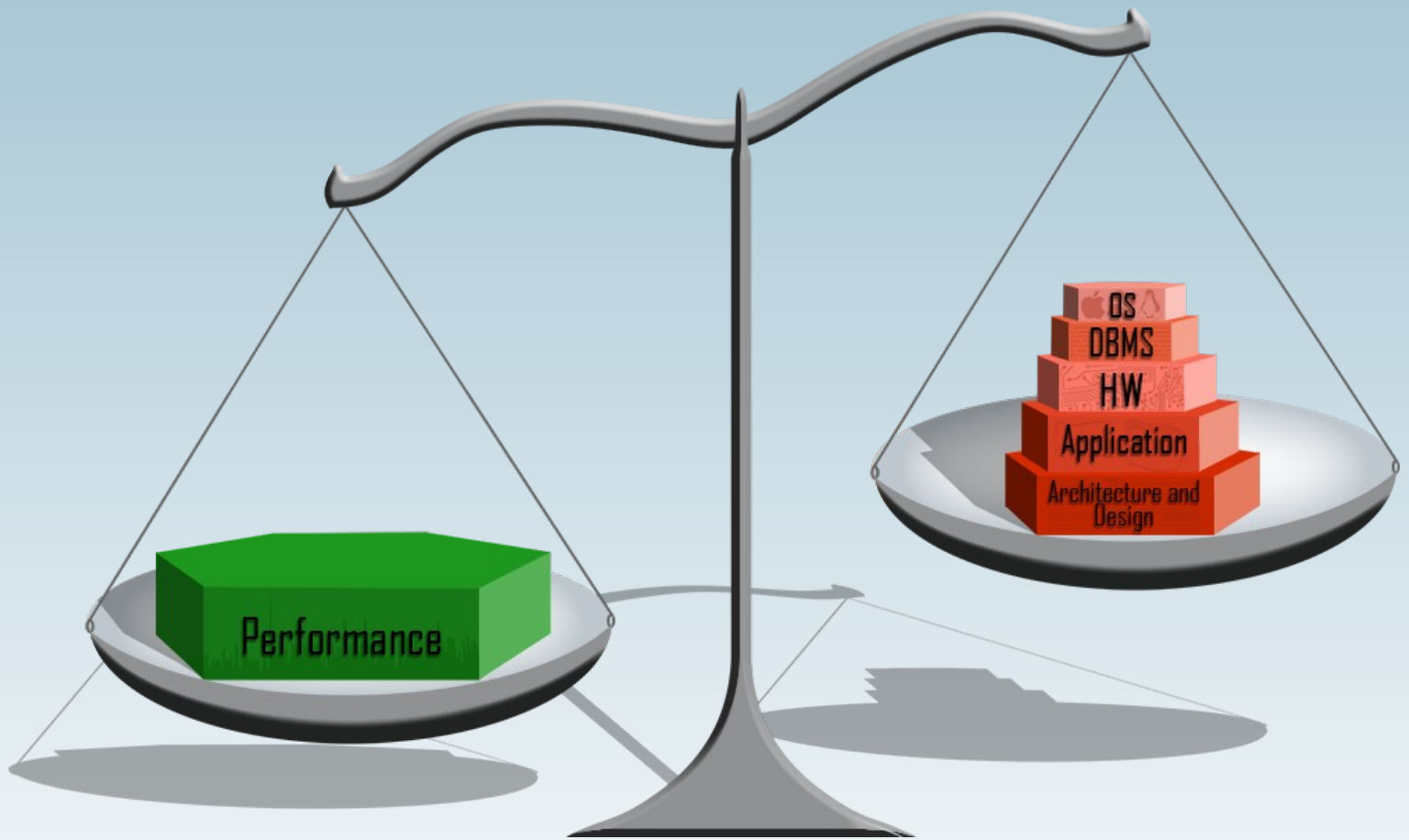
- **Was kostet mich Performance und Tuning?**



Dimitri Kravtchuk, MySQL
Performance Architect @ Oracle

- **Es gibt nix um sonst im Leben! Irgendwie muss ich immer dafür "bezahlen".**

Tuning Massnahmen

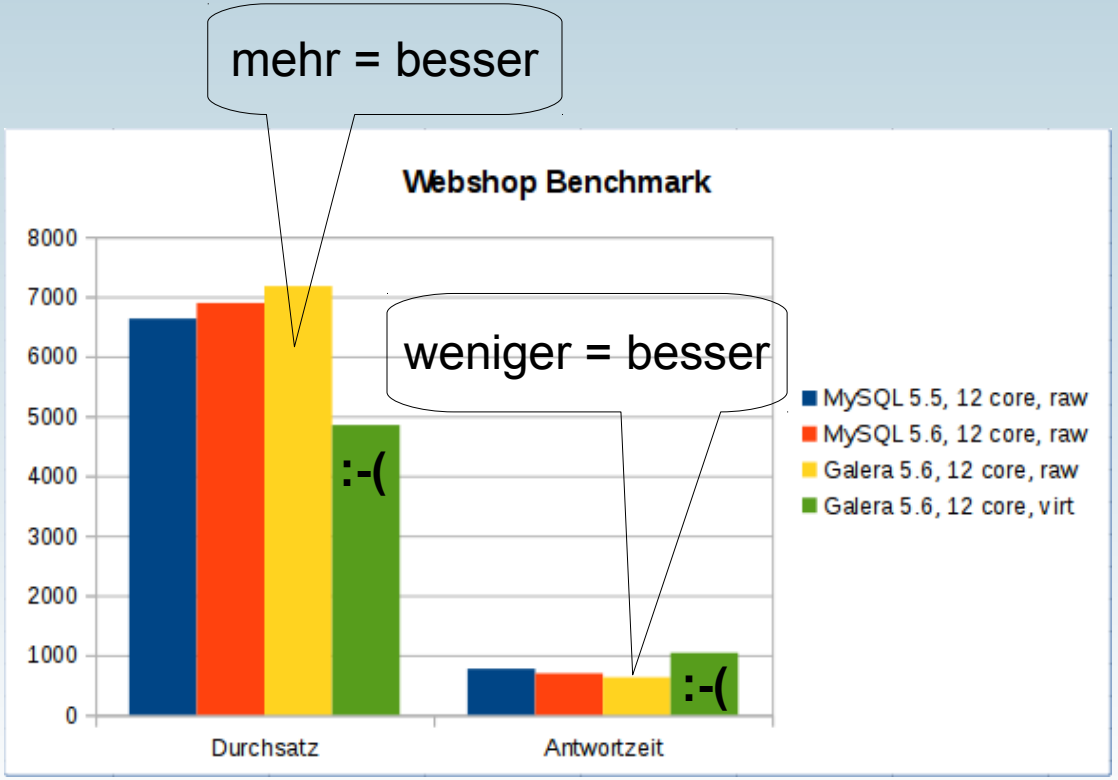


Hardware und O/S

- Tuning der Hardware
  - Kosten vs Performance
  - billig != performant (Latenz, Durchsatz)
- O/S (Linux)
  - Für die meisten Fälle OK
  - max. 20 – 50% mehr Durchsatz
  - Lohnt meist nicht!
- Virtualisierung
  - Gut für Konsolidieren von Micky Maus DB's
  - Schlecht für hohe Performance!

VM vs rohes-Eisen

- Benchmark von vorletzter Woche:
  - alte Hardware vs. neue Virtualisierung
  - MySQL 5.5 vs 5.6
  - Cluster vs. Single Node
  - Lasttest mit JMeter
  - Enterprise Webshop
    - Tomcat/Java
  - CPU-Bound ro-Workload

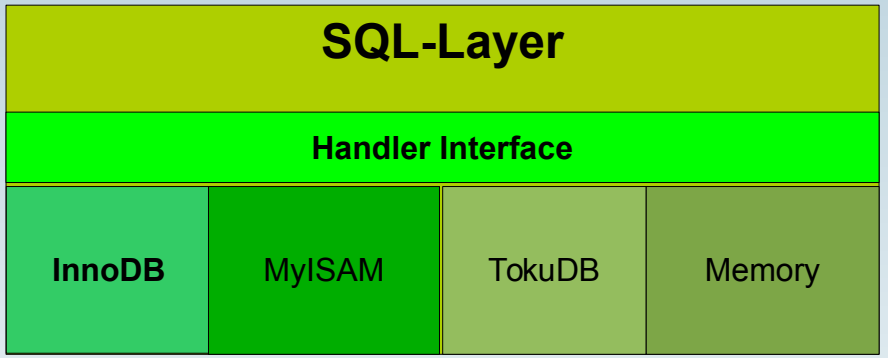


MySQL Konfiguration

- Bis hierhin war praktisch alles nicht MySQL spezifisch!

- MySQL kennt verschiedene „Storage Engines“
  - InnoDB, TokuDB, MyISAM, Memory

```
default_storage_engine = InnoDB
```



- → Heute v.a. InnoDB
  - Crash-Safe → automatisches Crash Recovery
  - Unterstützt Transaktionen
  - Erlaubt konkurrierendes Lesen und Schreiben

Wo konfiguriert man MySQL

- Konfigurationsdateien:
  - RedHat + SuSE: `/etc/my.cnf` + `/etc/my.cnf.d/*`
  - Debian + Ubuntu: `/etc/mysql/my.cnf` + `/etc/mysql/conf.d/*`
- Die Konfigurationsdatei ist unterteilt in verschiedene „Sections“:
  - `[client] [mysql] [mysqldump] [mysqld_safe] [mysqld] ...`
  - Richtige erwischen!
- Erfordert DB-Restart :-(
  - `shell> service mysql restart`
- Gutes Template unter:
  - http://fromdual.com/mysql-configuration-file-sample
- Online Konfiguration ändern:
  - `mysql> SET GLOBAL variable = wert;`
  - Persistieren in `my.cnf` nicht vergessen!
  - MySQL kennt KEINE spfiles!
- http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html

Konfiguration SQL-Layer

- Was konfiguriert man?
- Query Cache
  - Eigentlich: Result Cache
  - Nicht zu gross machen!
  - `query_cache_size = ≤128M`
  - Schlecht bei hoher Parallelität!
  - Achtung: in 5.6 default aus (`query_cache_type = on`)!
- Table Open Cache
  - Handle auf Tabellen, pro Verbindung pro Tabelle je eine
  - `table_open_cache = 384 - 2048`
- Table Definition Cache
  - Tabellendefinitionen, → Anz. Tabellen in Instanz
  - `table_defintion_cache = 512`

InnoDB Konfiguration

- InnoDB Buffer Pool
  - `innodb_buffer_pool_size` = 80% vom RAM
  - `innodb_buffer_pool_instances` = 8 - 16
- InnoDB Log Files
  - `innodb_log_file_size` = 30 Minuten Daten
  - `innodb_log_file_size` = 32 – 512M
- InnoDB Transaktionsflushing
  - `innodb_flush_log_at_trx_comit = 1` für sicher
  - `innodb_flush_log_at_trx_comit` = 0 oder 2 für schnell
- Transaktionslog und Binary Log syncing
  - `sync_binlog = 0` für schnell
  - `sync_binlog = 1` für sicher (neuer default in 5.7)

Wo schauen?

- Variablen (Einstellungen):
  - `mysql> SHOW GLOBAL VARIABLES LIKE '...';`
- Status (Messen was dabei raus kommt):
  - `mysql> SHOW GLOBAL STATUS LIKE '...';`
  - `mysql> SHOW ENGINE INNODB STATUS\G`
- Monitoring Lösungen
  - FromDual Performance Monitor für MySQL
  - MySQL Enterprise Monitor
  - etc.

Langsame Abfragen finden

- Alle DB Variablen sind richtig eingestellt...
- Performance Tuning Regel #1:
  - Tune the f... Query!
- Wie findet man diese Abfragen?
- Ohne Vorbereitung:
  - `SHOW PROCESSLIST;`
- Mit wenig Vorbereitung
  - Performance Schema
- Mit mehr Vorbereitung
  - Slow Query Log

SHOW PROCESSLIST

- Wann?
  - unvorbereitete und schnelle Intervention
  - „Es klemmt jetzt gerade!“
- Wie?
  - `mysql> SHOW [FULL] PROCESSLIST;`
  - Oder alternativ übers `INFORMATION_SCHEMA` oder neu `PERFORMANCE_SCHEMA`
- Was dann?
  - System beruhigen:

    `mysql> KILL [QUERY|CONNECTION] <connection_id>;`
  - Nachhaltig: Query tunen!

PERFORMANCE_SCHEMA


```
UPDATE setup_consumers SET enabled = 1 WHERE name = 'events_statements_history_long';

SELECT left(digest_text, 64) AS query
     , ROUND(SUM(timer_end-timer_start)/1000000000, 1) AS tot_exec_ms
     , ROUND(SUM(timer_wait)/1000000000, 1) AS tot_wait_ms
     , ROUND(SUM(lock_time)/1000000000, 1) AS tot_lock_ms
     , MIN(LEFT(DATE_SUB(NOW(), INTERVAL (isgs.VARIABLE_VALUE - TIMER_START*10e-13) second), 19)) AS first_seen
     , MAX(LEFT(DATE_SUB(NOW(), INTERVAL (isgs.VARIABLE_VALUE - TIMER_START*10e-13) second), 19)) AS last_seen
     , COUNT(*) as cnt
  FROM events_statements_history_long
  JOIN information_schema.global_status AS isgs
 WHERE isgs.variable_name = 'UPTIME'
 GROUP BY LEFT(digest_text,64)
 ORDER BY tot_exec_ms DESC
 LIMIT 10;

+----------------------------------------------------------------+---------+---------+---------+-------------+-------------+-----+
| query                                                          | exec_ms | wait_ms | lock_ms |  first_seen |   last_seen | cnt |
+----------------------------------------------------------------+---------+---------+---------+-------------+-------------+-----+
| INSERT INTO `test` SELECT ? , DATA , ? FROM `test`             | 50493.5 | 50493.5 |    26.3 | 12 16:41:35 | 12 16:42:04 |  20 |
| SELECT LEFT ( `digest_text` , ? ) , `ROUND` ( SUM ( `timer_end` | 14434.6 | 14434.6 |    25.8 | 12 16:48:44 | 12 17:07:15 |   6 |
| SELECT * FROM `test`                                           |  7483.0 |  7483.0 |     0.2 | 12 16:41:16 | 12 16:42:34 |   2 |
| SHOW ENGINE INNODB STATUS                                      |  1912.4 |  1912.4 |     0.0 | 12 16:37:19 | 12 17:07:36 | 687 |
| SHOW GLOBAL VARIABLES                                          |  1091.1 |  1091.1 |    68.8 | 12 16:37:19 | 12 17:07:36 | 687 |
| SHOW GLOBAL STATUS                                             |   638.7 |   638.7 |    40.8 | 12 16:37:19 | 12 17:07:36 | 687 |
| SELECT LEFT ( `digest_text` , ? ) , SUM ( `timer_end` - `timer_s |   356.2 |   356.2 |    42.4 | 12 16:42:38 | 12 16:45:00 |   6 |
| SELECT `digest_text` , SUM ( `timer_end` - `timer_start` ) / ? A |   325.3 |   325.3 |     0.4 | 12 16:40:44 | 12 16:42:18 |   3 |
| SELECT `DIGEST_TEXT` , ( `TIMER_END` - `TIMER_START` ) / ? AS `e |   163.2 |   163.2 |     1.0 | 12 16:37:44 | 12 16:39:22 |   9 |
| SELECT LOWER ( REPLACE ( trx_state , ?, ... ) ) AS state , COUNT |   133.9 |   133.9 |    80.2 | 12 16:37:19 | 12 17:07:36 | 687 |
+----------------------------------------------------------------+---------+---------+---------+-------------+-------------+-----+
```

http://fromdual.com/mysql-performance-schema-hints#top_long_running_queries

Slow Query Log

- Systematischer Ansatz mit etwas Vorlauf:

```
+---------------------------------+----------+
| Variable_name                   | Value    |
+---------------------------------+----------+
| slow_query_log                  | OFF      |
| slow_query_log_file             | slow.log |
| log_queries_not_using_indexes   | OFF      |
| long_query_time                 | 0.500000 |
+---------------------------------+----------+
```

- Kann dynamisch eingeschaltet werden:
  - `SET GLOBAL slow_quey_log = 1;`
- Profile vom Slow Query Log:
  - `shell> mysqldump_slow -s t slow.log > slow.profile`
  - `shell> pt_query_digest slow.log > slow.digest`

Optimiere das Query!

- Was machen mit den langsamen Abfragen?
- Query Execution Pläne (QEP) erstellen!
  - `mysql> EXPLAIN SELECT ...`
- Interpretieren von QEP:

```
EXPLAIN SELECT * FROM emp where name = 'Oli';

+-------------+-------+------+---------------+------+---------+------+--------+
| select_type | table | type | possible_keys | key  | key_len | ref  | rows   |
+-------------+-------+------+---------------+------+---------+------+--------+
| SIMPLE      | emp   | ALL  | last          | NULL | NULL    | NULL | 261369 |
+-------------+-------+------+---------------+------+---------+------+--------+
```

Operation

Genutzter Index

Angelangte Zeilen

`EXPLAIN` Type Operationen

billig

`const`	Höchstens eine passende Zeile, wird wie Konstante behandelt
`eq_ref`	Ein Zeile pro Zeile aus vorheriger Tabelle (Primary Key- / Unique Key-Join)
`ref`	Mehrere Zeilen pro Zeile aus vorheriger Tabelle (non-Unique Key-Join, Datenmenge nimmt zu!)
`fulltext`	Der Join wird mittels FULLTEXT Index gelöst
`index_merge`	Mehrere Index-Suchen werden gemerged
`xxx_subquery`	Subqueries
`range`	Index Range Scan
`index`	Full Index Scan (IFFS)
`ALL`	Full Table Scan

teuer

Monitoring

- Fiebermessen in der DB!
- Zwei Anforderungen:
  - Alarmierung, wenn was kaputt ist
  - Trends aufzeichnen, um Entwicklungen zu sehen
- FromDual Performance Monitor (fpmmm)
- MySQL Enterprise Monitor (MEM)
- Oracle Enterprise Monitor (OEM) + MySQL Plug-In
- Munin, Cacti, MRTG, ...

MySQL leckt Speicher?

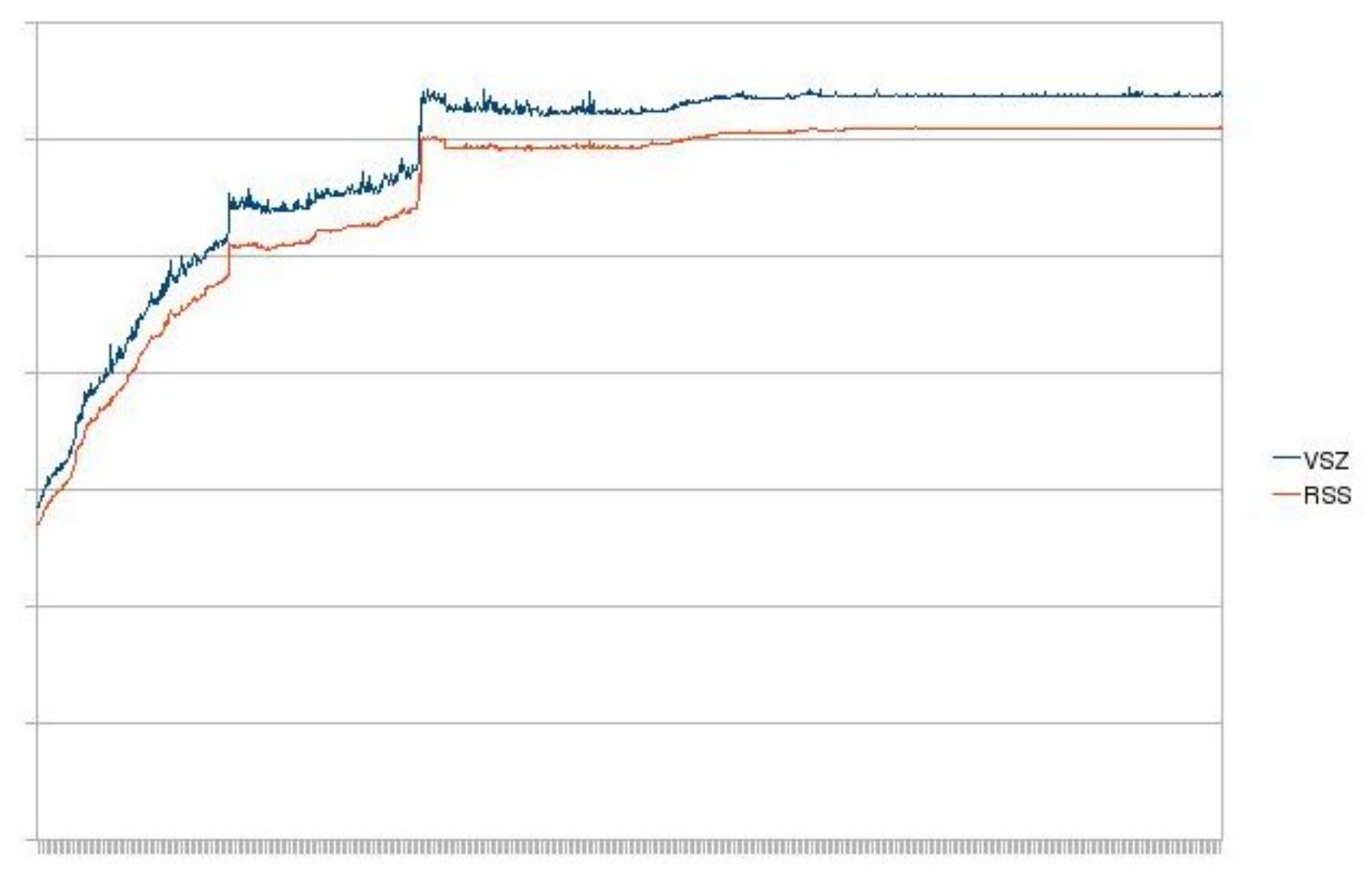
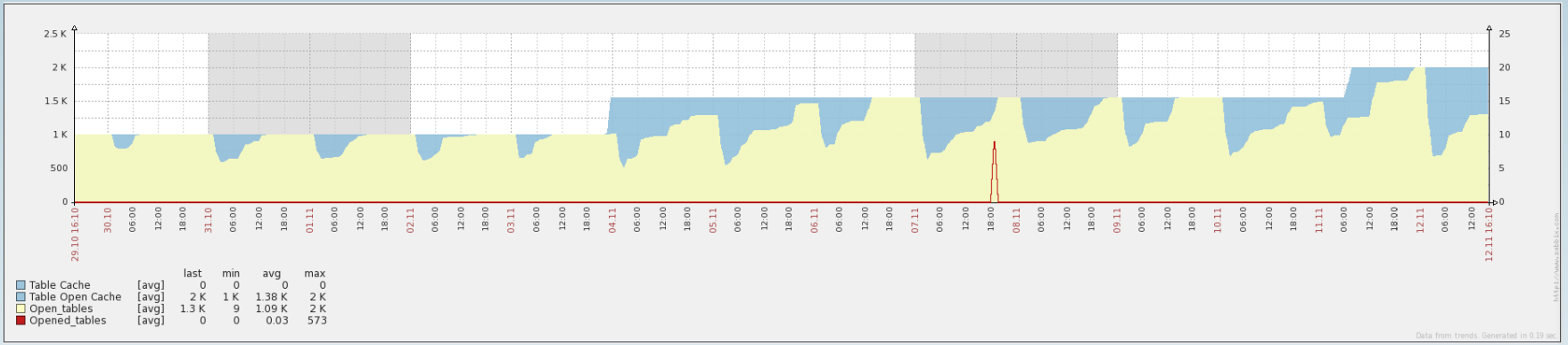


Table Open Cache gross genug?

www.fromdual.com



Schulung: MySQL Performance Tuning

ca. 6 x im Jahr (Berlin und Essen)

Stand 308

Q & A

Fragen ?

Diskussion?

Wir haben Zeit für ein persönliches Gespräch...

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

www.fromdual.com/presentations