MySQL
Database Scalability

Nextcloud Conference 2016
TU Berlin

Oli Sennhauser
Senior MySQL Consultant at FromDual GmbH
oli.sennhauser@fromdual.com
About FromDual GmbH
Contents

Database scalability

- Critical Resources
- Performance and Patterns
- HA and performance
- What is evil?
Critical resources

- RAM
  - Much helps a lot
  - (all/hot) data-set should fit into `innodb_buffer_pool_size`
  - We do NOT want to read from (slow) disk
  - How big is your database???

- I/O system
  - Databases do mostly async random write and fast sync sequential write
  - Dedicated, direct attached, 15k RPM, RAID-10 or SSD
  - We do not want to share and wait for slow far away disks (I/O latency, SAN, CFS)

- CPU
  - Fast cores process slow queries faster (NO green IT!)
  - 4 cores run 4 queries at the same time
  - Does Nextcloud really have concurrency for multi-socket machines?

- Network
  - We are not aware of any limits with databases on 1 and 10 Gbit networks
Performance and Scalability?

- What does it mean?
- Performance:
  - How fast? → Latency
  - How many per time? → Throughput
- Scalability:
  - To what point?
  - To what amount of data, transactions, users, time etc.?
- Overloading
Performance patterns

• Patterns we have to recognize:
  • Read
    – Latency
    – Throughput
  • Write
    – Latency
    – Throughput

  • Random read/write
    – e.g. random row fetches
    – e.g. random writes to disk
  • Sequential read/write
    – e.g. sequential full table scans
    – e.g. log writes to disk

• Caching effects (latency, when/why is it fast)?

<table>
<thead>
<tr>
<th></th>
<th>read</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>latency</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>throughput</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>read</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>random</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
What influences Read?

- Latency
  - Indexing
  - RAM vs. Disk access
  - Complexity of Query

- Throughput
  - 1 Connection = 1 query → 1 thread = 1 core
  - 4 cores can run 4 queries at the same time
  - 4 cores can process 4 seconds query per 1 second real time
  - 10'000 queries running 1 ms each will take about 2.5 seconds on a 4 core machine (probably more)
    - → less queries or faster queries or faster cores or more cores
Variables influencing Read

- We are talking about InnoDB only, right?
- SQL Layer
  - Query Cache (bottleneck at high throughput!)
  - Table Open Cache / Table Definition Cache
- InnoDB
  - InnoDB Buffer Pool Size
  - InnoDB Buffer Pool Instances
Variables influencing Write

- We are talking about InnoDB only, right?

- SQL Layer
  - Query Cache (negatively!)
  - Table Open Cache / Table Definition Cache

- InnoDB
  - InnoDB Flush Log at Trx Commit
  - InnoDB Log File Size
  - InnoDB Buffer Pool Size
  - InnoDB I/O capacity
What influences access

- **Sequential**
  - InnoDB PK (**AUTO_INCREMENT**)
  - Covering indexes (**index(a, b, c, d, e)**)

- **Random**
  - InnoDB PK (**HASH, UUID**)
  - Non covering index ranges → random fetch from table (index on gender → 50% rows in random order from table)
HA and performance

- Cluster is for HA not for HP!!!
- M/S and M/S → asynchronous (= fast)
- Galera → synchronous (= slower)
- Sync vs. async → Galera vs. M/S
- Sharding? (Fabrics or similar or DIY)
- HA vs. KISS!!!
- Is your Software Cluster aware???
Galera and Performance

**Single Instance vs. Galera**

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>Equal*</td>
<td>Worse</td>
</tr>
<tr>
<td>Throughput</td>
<td>Bigger</td>
<td>Bigger/equal?</td>
</tr>
</tbody>
</table>

Do NOT set `innodb_flush_log_at_trx_commit = 1` in Galera!!!

* If `wsrep_sync_wait = 1` → latency is expected to be higher
What is evil?

- Missing indexes
- Bad filter (too flexible user forms)
- Bad indexes, too many indexes
- Too complex queries (frameworks)
  - `SELECT *` is mostly NOT that evil (primarily)
  - Subquery → Use `JOIN` if possible (became better in 5.6 ff.)
  - Too-many-table-joins
- Long Primary Keys (join fields)
- HASH, UUID and similar as PK
  - `AUTO_INCREMENT` is mostly good!
- `BLOB/TEXT` and other trash in hot tables
- Log files, Mouse or click tracking, monitoring data in database
Questions?

Discussion?

We have time for some face-to-face talks...

- FromDual provides neutral and independent:
  - Consulting
  - Remote-DBA
  - Support for MySQL, Galera, Percona Server and MariaDB
  - Training

[www.fromdual.com/presentations]