MySQL vs MongoDB

Choosing right technology for your application

Peter Zaitsev
CEO, Percona
SCALE 15x
3 March 2017, Pasadena, CA
MySQL vs MongoDB
Bigger Question

What Open Source Database Technologies I should be using for My Application?
Even Bigger Question

What Database Technologies I should be using for My Application
Increasing number of Open Source First Companies

Only use Proprietary Software when there are no other choices

Cost, Reduced Vendor Lock In are the main factors
Open Source vs Proprietary trends
New Generation Technologies – Open Source

Popularity broken down by database model, October 2016

- Wide column stores: 99.7%
- Graph DBMS: 96.9%
- Key-value stores: 95.8%
- Time Series DBMS: 90.7%
- Document Stores: 87.9%
- Search engines: 87.9%
- Relational DBMS: 63.6%
- RDF stores: 63.6%
- Native XML DBMS: 69.3%
- Object oriented DBMS: 77.3%
- MultiValue DBMS: 94.2%

© 2016, DB-Engines.com

© 2017 Percona
Polyglot Persistence

Choose Multiple Database systems

Use Their Strong Sides
Architecture Examples

- Main Operation Store + Auxiliary Services
- Micro Services with their own stores
- Data Bus/Data Flow
For Example

MySQL as main data store

Use Memcache or Redis for caching

ElasticSearch for Search

Kafka to transfer data to ElasticSearch, Hadoop
Choosing the Data Store

SQL (Relational)  NoSQL (Non Relational)
Many NoSQL Data Models

Key Value (Memcache)

Wide Column (Cassandra)

Document (MongoDB)

Graph (Neo4J)
Why MySQL vs MongoDB?

322 systems in ranking, March 2017

<table>
<thead>
<tr>
<th>Mar 2017</th>
<th>Rank</th>
<th>Feb 2017</th>
<th>Mar 2016</th>
<th>DBMS</th>
<th>Database Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1.</td>
<td>1.</td>
<td>1.</td>
<td>Oracle +</td>
<td>Relational DBMS</td>
<td>1399.50</td>
</tr>
<tr>
<td>2.</td>
<td>2.</td>
<td>2.</td>
<td>2.</td>
<td>MySQL +</td>
<td>Relational DBMS</td>
<td>1376.07</td>
</tr>
<tr>
<td>3.</td>
<td>3.</td>
<td>3.</td>
<td>3.</td>
<td>Microsoft SQL Server +</td>
<td>Relational DBMS</td>
<td>1207.49</td>
</tr>
<tr>
<td>4.</td>
<td>4.</td>
<td>5.</td>
<td>5.</td>
<td>PostgreSQL +</td>
<td>Relational DBMS</td>
<td>357.64</td>
</tr>
<tr>
<td>5.</td>
<td>5.</td>
<td>4.</td>
<td>4.</td>
<td>MongoDB +</td>
<td>Document store</td>
<td>326.93</td>
</tr>
<tr>
<td>6.</td>
<td>6.</td>
<td>6.</td>
<td>6.</td>
<td>DB2 +</td>
<td>Relational DBMS</td>
<td>184.91</td>
</tr>
<tr>
<td>7.</td>
<td>8.</td>
<td>7.</td>
<td>7.</td>
<td>Microsoft Access</td>
<td>Relational DBMS</td>
<td>132.94</td>
</tr>
<tr>
<td>8.</td>
<td>7.</td>
<td>8.</td>
<td>8.</td>
<td>Cassandra +</td>
<td>Wide column store</td>
<td>129.19</td>
</tr>
<tr>
<td>9.</td>
<td>9.</td>
<td>10.</td>
<td>10.</td>
<td>SQLite</td>
<td>Relational DBMS</td>
<td>116.19</td>
</tr>
<tr>
<td>10.</td>
<td>10.</td>
<td>9.</td>
<td>9.</td>
<td>Redis +</td>
<td>Key-value store</td>
<td>113.01</td>
</tr>
</tbody>
</table>
My MySQL vs MongoDB

Most Popular Open Source Relational and Non Relational storage

Both Technologies started with Focus on Exceptional Ease of Use

These are 2 Technologies we focus at Percona
## Percona Software for MySQL and MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Percona Server for MySQL</td>
<td>• Percona Server for MySQL</td>
</tr>
<tr>
<td>• Percona XtraDB Cluster</td>
<td>• MongoRocksDB Storage Engine (RocksDB Based)</td>
</tr>
<tr>
<td>• Percona Xtrabackup</td>
<td>• Percona Memory Engine for MongoDB</td>
</tr>
<tr>
<td>• Percona Toolkit</td>
<td>• Percona Toolkit</td>
</tr>
<tr>
<td>• Percona Monitoring and Management</td>
<td>• Percona Monitoring and Management</td>
</tr>
<tr>
<td>• TokuDB Storage Engine</td>
<td></td>
</tr>
</tbody>
</table>

© 2017 Percona
I Personally know MySQL much better than MongoDB
NoSQL on MySQL

Starting MySQL 5.7 MySQL Supports NoSQL Interface

MongoDB Like CRUD access

With Transaction Support

Without Good Sharding Solution

Not yet commonly adopted
Top Things to Consider

- Team Preferences and Experience
- Development Process and Application Lifecycle
- Data Model
- Transactions and Consistency
- Performance
- Scalability
- Administration
- Legal
Team Preferences and Experience

Absolutely Key Question!

Unloved Technology – Recipe for Disaster

Problems may have more than one solution

Consider Original Development and Full Lifecycle
# Team Experience MySQL vs MongoDB

<table>
<thead>
<tr>
<th><strong>MySQL</strong></th>
<th><strong>MongoDB</strong></th>
</tr>
</thead>
</table>
| • Very Mature Technology  
• SQL Standard  
• Easier to move to other relational databases  
• Transactions  
• A lot of Choices | • Modern and Dynamic  
• Do not need to learn nasty SQL  
• Dynamic Schema  
• Complex Queries are recipe for complex problems  
• Easier Scalability  
• Native Integration Javascript, JSON |
Development Process and LifeCycle

- Faster Development pace or more control?
- Data Always Have Schema
- Is Data Belong to Application or used by Multiple Applications
- Application Life Time
- Application Active development time
## Development in MySQL and MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Relational Database requires more planning and control</td>
<td>• Development Pace</td>
</tr>
<tr>
<td>• Easy to use data from many applications</td>
<td>• No Need to Synchronize Schema between Application and Database</td>
</tr>
<tr>
<td>• Many applications 15+ Years</td>
<td>• Clear Path to Scalability</td>
</tr>
<tr>
<td>• Can use SQL and NoSQL Interface</td>
<td></td>
</tr>
</tbody>
</table>
Data Model

Optimal Data Model depends on the Application and Team Experience
# Data Model in MySQL vs MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Relational Data Model</td>
<td>• Document Oriented Data Model</td>
</tr>
<tr>
<td>• Easy to reflect relationships between objects</td>
<td>• Documents can embed other documents</td>
</tr>
<tr>
<td>• Change data in one place</td>
<td>• Easy to represent JSON data common in Web Applications</td>
</tr>
<tr>
<td>• Query Result - Table</td>
<td>• Complex JOINs can often be avoided</td>
</tr>
<tr>
<td></td>
<td>• Query Result – List of Documents (different structure)</td>
</tr>
</tbody>
</table>
Model Design Example

People and their Passports

People may have more than on passport

Not Everyone Has a Passport
# Relational Schema Design

```sql
mysql> select * from people;
```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stephane</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
</tr>
<tr>
<td>4</td>
<td>Cinderella</td>
</tr>
</tbody>
</table>

```sql
mysql> select * from passports;
```
<table>
<thead>
<tr>
<th>id</th>
<th>people_id</th>
<th>country</th>
<th>valid_until</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>FR</td>
<td>2020-01-01</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>US</td>
<td>2020-01-01</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>RU</td>
<td>2020-01-01</td>
</tr>
</tbody>
</table>
Easy To Answer Various Questions

- **Number of People**
  - SELECT count(*) FROM people

- **Validity Date of Persons passport**
  - SELECT valid_until from passports ps join people pl ON ps.people_id = pl.id WHERE name = 'Stephane'

- **Who Does not have A Passport ?**
  - SELECT name FROM people pl LEFT JOIN passports ps ON ps.people_id = pl.id WHERE ps.id IS NULL
MongoDB Design Example

```javascript
> db.people_embed.find().pretty()
{
  "_id": ObjectId("51f7c0048ded44d5ebb83774"),
  "name": "Stephane",
  "passport": {
    "country": "FR",
    "valid_until": ISODate("2019-12-31T23:00:00Z")
  }
}
{
  "_id": ObjectId("51f7c70e8ded44d5ebb83775"),
  "name": "John",
  "passport": {
    "country": "US",
    "valid_until": ISODate("2019-12-31T23:00:00Z")
  }
}
{
  "_id": ObjectId("51f7c71b8ded44d5ebb83776"),
  "name": "Michael",
  "passport": {
    "country": "RU",
    "valid_until": ISODate("2019-12-31T23:00:00Z")
  }
}
{
  "_id": ObjectId("51f7c7258ded44d5ebb83777"), "name": "Cinderella"
}
```
## MongoDB and MySQL Terms

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database/Schema</td>
<td>Database</td>
</tr>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
<tr>
<td>Column</td>
<td>Field</td>
</tr>
<tr>
<td>Primary Key</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Join</td>
<td>Embedded Document or Linking</td>
</tr>
</tbody>
</table>
Access Language

SQL

- Structured
- Query
- Language

CRUD

- Create
- Read
- Update
- Delete
SQL vs CRUD - Insert

SQL

```
INSERT INTO book (
    'ISBN', 'title', 'author'
) 
VALUES (
    '9780992461256',
    'Full Stack JavaScript',
    'Colin Ihrig & Adam Bretz'
);
```

CRUD

```
db.book.insert(
    { 
        ISBN: "9780992461256",
        title: "Full Stack JavaScript",
        author: "Colin Ihrig & Adam Bretz"
    }
);
```
SQL vs CRUD Update

UPDATE book
SET price = 19.99
WHERE ISBN = '9780992461256'

```
db.book.update(
    { ISBN: '9780992461256' },
    { $set: { price: 19.99 } }
);
```
SQL vs CRUD - Delete

```
DELETE FROM book
WHERE publisher_id = 'SP001';
```

```
db.book.remove({
   "publisher.name": "SitePoint"
});
```
SQL vs CRUD - Search

```sql
SELECT title FROM book
WHERE price > 10;
```

```javascript
db.book.find(
    { price: { $gt: 10 } },
    { _id: 0, title: 1 }
);
```
SQL vs CRUD - Count

```sql
SELECT COUNT(1) FROM book
WHERE publisher_id = 'SP001';
```

```javascript
db.book.count({
  "publisher.name": "SitePoint"
});
```
SQL vs CRUD - Aggregation

```
SELECT format, COUNT(1) AS `total`
FROM book
GROUP BY format;
```

```javascript
db.book.aggregate([
  {$group:
    {
      _id: "$format",
      total: { $sum: 1 }
    }
  }
]);
```
Transactions and Consistency

- Atomicity
- Consistency
- Isolation
- Durability
# Transactions in MySQL vs MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ACID Transactions of arbitrary size</td>
<td>• ACID on the document level</td>
</tr>
</tbody>
</table>

© 2017 Percona
MongoDB Reads are Dirty Reads

Reading a Large Result will Return Documents from different point in time

Same Document Can (rarely) be Returned More than Once
Performance

Very Hard to Compare Directly

Depends a lot on the Application Architecture

Many Benchmarks compare different Durability or Settings

MongoDB Scales Better so single node performance less of a focus
MySQL vs MongoDB Performance


### Small server: Linkbench, IO-bound

MyRocks: best throughput & QoS, most efficient
MongoRocks: better than WiredTiger

<table>
<thead>
<tr>
<th></th>
<th>TPS</th>
<th>Iostat r/t</th>
<th>Iostat wKB/t</th>
<th>CPU usecs/t</th>
<th>Size (GB)</th>
<th>p99 update</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoRocks+zlib</td>
<td>1087</td>
<td>1.07</td>
<td>4.42</td>
<td>24656</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>WiredTiger+zlib</td>
<td>429</td>
<td>1.24</td>
<td>17.98</td>
<td>153763</td>
<td>33</td>
<td>22</td>
</tr>
<tr>
<td>MyRocks+zlib</td>
<td>2246</td>
<td>0.67</td>
<td>1.27</td>
<td>12688</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>InnoDB</td>
<td>1860</td>
<td>0.82</td>
<td>10.62</td>
<td>7991</td>
<td>63</td>
<td>14</td>
</tr>
<tr>
<td>InnoDB+zlib</td>
<td>1855</td>
<td>0.67</td>
<td>8.60</td>
<td>10431</td>
<td>40</td>
<td>8</td>
</tr>
</tbody>
</table>
Scalability

What do I need to do to scale my Application from 100 to 100.000.000 users?

- Scalability within single system
- Scalability for multiple systems
- Scaling reads, writes, data size
# Scalability MySQL and MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scales well with single system size</td>
<td>Scalability was focused upon from the early versions</td>
</tr>
<tr>
<td>Can be scaled to Facebook Scale</td>
<td>Built in Sharding</td>
</tr>
<tr>
<td>Easy to scale for medium size applications</td>
<td>Sharding for Scalability, Replication is for HA</td>
</tr>
<tr>
<td>Scaling Reads with Replication</td>
<td></td>
</tr>
<tr>
<td>Sharding is Painful</td>
<td></td>
</tr>
</tbody>
</table>
Database Operations

What Developers do not like thinking about

- Automation
- Backups
- Version Updates
- Monitoring
- Disaster Recovery
- Performance Management
## Operations MySQL and MongoDB

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Flexibility</td>
<td>• Simplicity</td>
</tr>
<tr>
<td>• Many ways to solve problems</td>
<td>• Virtually Zero Administration</td>
</tr>
<tr>
<td>• Many Open Source Tools available</td>
<td>• Fewer Open Source Operation Tools</td>
</tr>
<tr>
<td>• Out of the box Manual Failover in Replication</td>
<td>• Strong Connection to MongoDB Ops Manager</td>
</tr>
</tbody>
</table>
License

MySQL

- GPL or Proprietary

MongoDB

- AGPL or Proprietary
Solved MongoDB Problems

- Poor concurrent write performance with MMAPv1
  - Fixed by WiredTiger and MongoRocks
- Poor disk space usage with MMAPv1
  - Fixed by WiredTiger and MongoRocks
- No way to validate documents
  - Document Validation since MongoDB 3.2
- No JOIN
  - Limited support in aggregation framework starting MongoDB 3.2
Typical MySQL Example

Online Ecommerce

- Full Feature Transactions are important
- Relational Model is a good fit
- Data size is typically not requiring sharding
- Long Term Application development
- Same data used by many applications directly
Typical MongoDB Example

Online Game Backend

• Extreme scaling might be needed if game becomes popular
• Data is used by one application
• Data is complicated and does not fit relational model well
• Document level consistency is often enough
• Low pace of development after “launch”
Extra Material

Other things worth sharing
Just Released!

Introducing Percona Server for MongoDB™ Product Bundle

- Percona Server for MongoDB 3.4
- Percona Monitoring and Management 1.1
- Percona Toolkit 3.0
- Optimize wider range of database workloads with greater reliability and security

Find out more
Percona Server for MongoDB 3.4

100% Free and Open Source

100% Compatible with MongoDB 3.4

Additional Memory Engine and MongoRocks

Pluggable Authentication as in MongoDB Enterprise

Auditing as in MongoDB Enterprise

Advanced Query Performance Monitoring

Hot Backups for WiredTiger
Percona Monitoring and Management

• Comprehensive Database Focused Monitoring
• 100% Open Source Roll-your-own solution
• Easy to install and use
• Supports MySQL and MongoDB
• Version 1.1 focuses on Trending and Query Analyses
• Management Features to Come
### What Queries are causing the load?

Top 10 of 57 Queries by % Grand Total Time (%GTT)

<table>
<thead>
<tr>
<th>#</th>
<th>Query Abstract</th>
<th>ID</th>
<th>Load</th>
<th>Count</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>7.88 (100%)</td>
<td>780.82 QPS</td>
<td>2.81 m (100%)</td>
</tr>
<tr>
<td>1</td>
<td>UPDATE sbtest</td>
<td>D30AD7E3079ABCE7</td>
<td>5.31 (67.30%)</td>
<td>258.58 QPS</td>
<td>930.90 k (33.12%)</td>
</tr>
<tr>
<td>2</td>
<td>LOCK sbtest</td>
<td>0B759DF6D01BDB8F</td>
<td>1.20 (15.21%)</td>
<td>2.54 QPS</td>
<td>9.15 k (0.33%)</td>
</tr>
<tr>
<td>3</td>
<td>SELECT sbtest</td>
<td>558CAEF5F387E929</td>
<td>0.42 (5.31%)</td>
<td>170.65 QPS</td>
<td>614.35 k (21.86%)</td>
</tr>
<tr>
<td>4</td>
<td>COMMIT</td>
<td>813031B8BC38329</td>
<td>0.23 (2.86%)</td>
<td>14.06 QPS</td>
<td>50.62 k (1.80%)</td>
</tr>
<tr>
<td>5</td>
<td>SELECT myisam.sbtest</td>
<td>C4832A98728C4424</td>
<td>0.18 (2.25%)</td>
<td>&lt;0.01 QPS</td>
<td>4.00 (0.00%)</td>
</tr>
<tr>
<td>6</td>
<td>SELECT sbtest</td>
<td>87625C47A176BEDD</td>
<td>0.08 (0.96%)</td>
<td>188.79 QPS</td>
<td>679.63 k (24.18%)</td>
</tr>
<tr>
<td>7</td>
<td>SELECT sbtest</td>
<td>643B7802D745420</td>
<td>0.07 (0.88%)</td>
<td>17.27 QPS</td>
<td>62.19 k (2.21%)</td>
</tr>
<tr>
<td>8</td>
<td>SELECT sbtest</td>
<td>9CD3EAA5A1950648</td>
<td>0.07 (0.85%)</td>
<td>16.88 QPS</td>
<td>60.75 k (2.16%)</td>
</tr>
<tr>
<td>9</td>
<td>SELECT sbtest</td>
<td>FE6FFA06B3AC9BB4</td>
<td>0.07 (0.83%)</td>
<td>17.09 QPS</td>
<td>61.53 k (2.19%)</td>
</tr>
<tr>
<td>10</td>
<td>SELECT sbtest</td>
<td>F54DBEF3D7AE474D</td>
<td>0.06 (0.75%)</td>
<td>17.33 QPS</td>
<td>62.39 k (2.22%)</td>
</tr>
</tbody>
</table>
**Why are they causing this load?**

---

**UPDATE sbtest**

---

**Selected query class:** 930.90 k Queries (258.58 QPS, 67.30%, 5.31 Load) | Total: 2.81 m Queries (780.82 QPS, 100.00%, 7.88 Load)

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Rate/Sec</th>
<th>Sum</th>
<th>Per Query Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Count</td>
<td>259.12 (per sec)</td>
<td>932.85 k 32.40% of total</td>
<td></td>
</tr>
<tr>
<td>Query Time</td>
<td>5.31 load (67.16%)</td>
<td>19101.45 sec 67.16% of total</td>
<td>5.25 ms avg</td>
</tr>
<tr>
<td>Lock Time</td>
<td>1.38 (avg load)</td>
<td>4965.60 sec 51.50% of total 15.13% of query time</td>
<td>793.79 µs avg</td>
</tr>
<tr>
<td>Innodb Row Lock Wait</td>
<td>&lt;0.01 (avg load)</td>
<td>28.14 sec 42.03% of total 1.44% of query time</td>
<td>75.79 µs avg</td>
</tr>
<tr>
<td>Innodb IO Read Wait</td>
<td>&lt;0.01 (avg load)</td>
<td>35.85 sec 2.47% of total 10.93% of query time</td>
<td>573.60 µs avg</td>
</tr>
<tr>
<td>Innodb Read Ops</td>
<td>2.48 (per sec)</td>
<td>8.91 k 1.86% of total</td>
<td>0.00 avg</td>
</tr>
<tr>
<td>Innodb Read Bytes</td>
<td>39.61 KB (per sec)</td>
<td>139.25 MB 1.86% of total 16.00 KB avg io size</td>
<td>3.50 KB avg</td>
</tr>
<tr>
<td>Innodb Distinct Pages</td>
<td>-</td>
<td>-</td>
<td>6.03 avg</td>
</tr>
<tr>
<td>Bytes Sent</td>
<td>13.18 KB (per sec)</td>
<td>46.35 MB 1.72% of total</td>
<td>52.00 Bytes avg</td>
</tr>
<tr>
<td>Rows Examined</td>
<td>258.17 (per sec)</td>
<td>929.43 k 0.64% of total 0.00 per row sent</td>
<td>0.88 avg</td>
</tr>
</tbody>
</table>
How to fix them

CREATE TABLE `sbttest1` (  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,  `k` int(10) unsigned NOT NULL DEFAULT '0',  `c` char(120) NOT NULL DEFAULT '',  `pad` char(60) NOT NULL DEFAULT '',  PRIMARY KEY (`id`),  KEY `k_1` (`k`) ) ENGINE=MyISAM AUTO_INCREMENT=100000001 DEFAULT CHARSET=latin1

<table>
<thead>
<tr>
<th>CREATE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>Name</td>
<td>sbtest1</td>
</tr>
<tr>
<td>Engine</td>
<td>MyISAM</td>
</tr>
<tr>
<td>Version</td>
<td>10</td>
</tr>
<tr>
<td>RowFormat</td>
<td>Fixed</td>
</tr>
<tr>
<td>Rows</td>
<td>100.00 m</td>
</tr>
<tr>
<td>AvgRowLength</td>
<td>189.00 Bytes</td>
</tr>
<tr>
<td>DataLength</td>
<td>17.60 GB</td>
</tr>
<tr>
<td>MaxDataLength</td>
<td>756.00 GB</td>
</tr>
<tr>
<td>IndexLength</td>
<td>1.70 GB</td>
</tr>
</tbody>
</table>
# System Information

## MySQL Summary

<table>
<thead>
<tr>
<th>System Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Path to executable</strong></td>
</tr>
<tr>
<td><strong>Has symbols</strong></td>
</tr>
<tr>
<td><strong>User</strong></td>
</tr>
<tr>
<td><strong>Time</strong></td>
</tr>
<tr>
<td><strong>Hostname</strong></td>
</tr>
<tr>
<td><strong>Version</strong></td>
</tr>
<tr>
<td><strong>Built On</strong></td>
</tr>
<tr>
<td><strong>Started</strong></td>
</tr>
<tr>
<td><strong>Databases</strong></td>
</tr>
<tr>
<td><strong>Datadir</strong></td>
</tr>
<tr>
<td><strong>Processes</strong></td>
</tr>
<tr>
<td><strong>Replication</strong></td>
</tr>
<tr>
<td><strong>Pidfile</strong></td>
</tr>
</tbody>
</table>
What happens on OS and Hardware Level

- **I/O Activity**
  - Page In Avg: 9.3 MBps
  - Page Out Avg: 8.3 MBps

- **Memory Distribution**
  - Used Avg: 9.62 GB
  - Cached Avg: 5.45 GB
  - Free Avg: 270 MB
  - Buffers Avg: 145 MB

- **CPU Usage / Load**
  - idle Avg: 85%
  - user Avg: 9%
  - iowait Avg: 4%
  - system Avg: 2%
  - stalling Avg: 0%
  - Load 1m Avg: 2

- **Disk Latency**
  - Read Avg: 0.88 ms
  - Write Avg: 2.03 ms

- **Network Traffic**
  - Outbound Avg: 27.5 kBps
  - Inbound Avg: 3.0 kBps

- **Swap Activity**
  - Swap In Avg: 45 Bps
  - Swap Out Avg: 6 Bps

© 2017 Percona
As well as Database Level

<table>
<thead>
<tr>
<th>MySQL Uptime</th>
<th>Current QPS</th>
<th>InnoDB Buffer Pool Size</th>
<th>Buffer Pool Size of Total RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9 weeks</td>
<td>438.74</td>
<td>8 GiB</td>
<td>52%</td>
</tr>
</tbody>
</table>

### MySQL Connections
- Max Connections: 512
- Max Used Connections: 137
- Connections: 3

### MySQL Active Threads
- Threads Connected: 102
- Threads Running: 1

### MySQL Questions
- Questions: 430

### MySQL Thread Cache
- Thread Cache Size: 64
Detailed MongoDB Storage Engine Monitoring
Check out the Demo

http://pmmdemo.percona.com
April 24th-27th, 2017
Santa Clara, California

Open Source Database Conference
MySQL, MariaDB, MongoDB, PostgreSQL
Time Series, Monitoring, RocksDB and More

Use Discount Code “SCALE15” for 15% off registration

perconalive.com
BECOME A PERCONA SUPERHERO

WE’RE HIRING

CONTACT

careers@percona.com
Meet us at Booth #219
About Percona

Solutions for your success with MySQL and MongoDB

Support, Managed Services, Software

Our Software is 100% Open Source

Support Broad Ecosystem – MySQL, MariaDB, Amazon RDS

In Business for more than 10 years

More than 3000 customers, including top Internet companies and enterprises