

MySQL vs MongoDB

When to Use Which Technology

Peter Zaitsev

CEO

Percona University, Berlin

May 9th, 2017



In This Presentation

**Very brief discussion on
merits of MySQL and
MongoDB**

Why MySQL and MongoDB ?

Most Popular OpenSource SQL and NoSQL Engines

327 systems in ranking, May 2017

Rank			DBMS	Database Model	Score		
May 2017	Apr 2017	May 2016			May 2017	Apr 2017	May 2016
1.	1.	1.	Oracle +	Relational DBMS	1354.31	-47.68	-107.71
2.	2.	2.	MySQL +	Relational DBMS	1340.03	-24.59	-31.80
3.	3.	3.	Microsoft SQL Server +	Relational DBMS	1213.80	+9.03	+70.98
4.	4.	↑ 5.	PostgreSQL +	Relational DBMS	365.91	+4.14	+58.30
5.	5.	↓ 4.	MongoDB +	Document store	331.58	+6.16	+11.36
6.	6.	6.	DB2 +	Relational DBMS	188.84	+2.18	+2.88
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	129.87	+1.69	-1.70
8.	8.	↓ 7.	Cassandra +	Wide column store	123.11	-3.07	-11.39
9.	9.	9.	Redis +	Key-value store	117.45	+3.09	+9.21
10.	10.	10.	SQLite	Relational DBMS	116.07	+2.27	+8.81

Why MySQL and MongoDB ?

**Two Technologies
Percona Provides
Solutions For**

Full Disclosure

**I know MySQL Much
better than MongoDB...
which will impact my
bias**

MySQL

Relational Database First and Foremost

Full SQL Support, Transactions, ACID

Designed for a Single Server first

Scale-Out as Afterthought

MongoDB

Designed for “Web Scale”

Scalability, Cloud, Multiple Machines

Replication and Sharding part of initial design

Only features which can scale

Q1: What do you know and love?

Both MySQL and MongoDB are very capable. Your experience and preference matter

Q2: Which data model fits better ?

Relational

- MySQL Obvious Choice

Document Based

- MongoDB Obvious choice
- MySQL has Document Store starting 5.7

Q3: How Data is Used

Data belongs to single application

- JSON model more expressive for application data structures
- Schema designed for specific access paths

Data shared by multiple applications

- Relational structure easier to share
- Can be more flexible in how data is accessed

Q4: Transactions

Need full Transactions

- MySQL can be better choice
- One of the main benefits of MySQL Document Store

Do not need Transactions

- MongoDB can be great choice
- Can do Atomic Document Updates

Q5: JOINS

Advanced JOINS and other SQL features

- MySQL much more powerful
- \$lookup and \$graphLookup features in MongoDB aggregation framework

Mainly simple lookups with filters/sorting

- MongoDB and MySQL both do these very well

Q6: Scale

Single Server is Good Enough

- MySQL works great
- Well optimized for Many cores; large memory; fast storage

Need Massive Scale out

- Automated sharding in MongoDB is much better
- Replication in MongoDB is easier to use
- Solutions like Vitess try to make it less painful for MySQL

Q7: Large Scale Aggregation

MongoDB

- has built in aggregation framework for parallel processing
- BI Connector and ToroDB for SQL access
- Replicate to Hadoop

MySQL

- Executes every query single threaded
- MariaDB ColumnStore (InfiniDB reborn)
- ClickHouse
- Replicate to Hadoop

MySQL and MongoDB compared

Courtesy of Alexander Rubin

From



to



MySQL

```
mysql> select * from zips limit 1\G
***** 1. row
*****
country_code: US
postal_code: 34050
place_name: FPO
admin_name1:
admin_code1: AA
admin_name2: Erie
admin_code2: 029
admin_name3:
admin_code3:
latitude: 41.03750000
longitude: -111.67890000
accuracy:
1 row in set (0.00 sec)
```

MongoDB

```
MongoDB shell version: 3.0.8
connecting to: zips
> db.zips.find().limit(1).pretty()
{
  "_id" : "01001",
  "city" : "AGAWAM",
  "loc" : [
    -72.622739,
    42.070206
  ],
  "pop" : 15338,
  "state" : "MA"
}
```



Where is my SQL?

SQL to MongoDB Mapping Chart

<https://docs.mongodb.org/manual/reference/sql-comparison/>

MySQL

```
CREATE TABLE users (  
  id MEDIUMINT NOT NULL  
    AUTO_INCREMENT,  
  user_id Varchar(30),  
  age Number,  
  status char(1),  
  PRIMARY KEY (id)  
)
```

MongoDB

```
db.users.insert( {  
  user_id: "abc123",  
  age: 55,  
  status: "A"  
} )
```

(no schema)



Where is my *SQL*?

SQL to MongoDB Mapping Chart

<https://docs.mongodb.org/manual/reference/sql-comparison/>

MySQL

SELECT *

FROM users

WHERE status = "A"

AND age = 50

MongoDB

```
db.users.find(  
    { status: "A",  
      age: 50 }  
)
```



Where is my */etc/my.cnf*?

MySQL

`/etc/my.cnf`

MongoDB

`/etc/mongod.conf`

```
# Where and how to store data.  
storage:
```

```
  dbPath: /datawt
```

```
  journal:
```

```
    enabled: true
```

```
  engine: wiredTiger
```

```
...
```

```
/usr/bin/mongod -f
```

```
/etc/mongod.conf
```



Where are my *databases/tables?*

MySQL

Databases

```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| ...                     |
```

```
mysql> use zips
Database changed
```

Tables

```
mysql> show tables;
+-----+
| Tables_in_zips          |
+-----+
| zips                    |
+-----+
```

MongoDB

Databases

```
> show dbs;
admin    0.000GB
local    0.000GB
osm      13.528GB
test     0.000GB
zips     0.002GB
```

```
> use zips
switched to db zips
```

Collections

```
> show collections
zips
> show tables // same
zips
```



Where is my *InnoDB*?

MySQL

MyISAM

InnoDB

TokuDB

MyRocks (RocksDB)*

MongoDB

MMAPv1 memory mapped stored engine,

WiredTiger transactional, with compression, btree

Percona Memory Engine

MongoRocks (RocksDB)*



Where is my *Processlist*?

```
mysql> show processlist\G
***** 1. row
*****
      Id: 137259
     User: root
    Host: localhost
       db: geonames
 Command: Query
      Time: 0
     State: init
    Info: show processlist
 Rows_sent: 0
Rows_examined: 0
1 row in set (0.00 sec)
```

```
> db.currentOp()
{
  "inprog" : [
```

```
    {
      "desc" : "conn28",
      "threadId" : "0x19b85260",
      "connectionId" : 28,
      "opid" : 27394208,
      "active" : true,
      "secs_running" : 3,
      "microsecs_running" :
        NumberLong(3210539),
      "op" : "query",
      "ns" : "osm.points3",
      "query" : {
        "name" : "Durham"
      },
      "planSummary" : "COLLSCAN",
      "client" : "127.0.0.1:58835",
      "numYields" : 24905,
      "locks" : {
        "Global" : "r",
        "Database" : "r",
        "Collection" : "r"
      },
      "waitingForLock" : false,
```



Where are my *Grants*?

```
mysql> grant all on *.* to  
user@localhost identified by 'pass';
```

```
> use products  
db.createUser(  
  {  
    user: "accountUser",  
    pwd: "password",  
    roles: [ "readWrite", "dbAdmin"  
  ]  
}  
)
```



Where is my *Index*?

MySQL

```
mysql> show keys from zips\G
***** 1. row
*****

      Table: zips
    Non_unique: 0
      Key_name: PRIMARY
Seq_in_index: 1
  Column_name: id
    Collation: A
  Cardinality: 0
      Sub_part: NULL
        Packed: NULL
          Null:
    Index_type: BTREE
      Comment:
Index_comment:
***** 2. row
*****

      Table: zips
    Non_unique: 1
      Key_name: postal_code
Seq_in_index: 1
```

MongoDB

```
> db.zips.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "zips.zips"
  }
]
```



Where is my *add index*?

```
mysql> alter table zips add key  
(postal_code);  
Query OK, 0 rows affected (0.10  
sec)  
Records: 0 Duplicates: 0  
Warnings: 0
```

```
> db.zips.createIndex({ state : 1 } )  
{  
  "createdCollectionAutomatically" :  
false,  
  "numIndexesBefore" : 1,  
  "numIndexesAfter" : 2,  
  "ok" : 1  
}
```

// Index can be sorted:

```
> db.zips.createIndex({ state : -1 } )  
{  
  "createdCollectionAutomatically" :  
false,  
  "numIndexesBefore" : 2,  
  "numIndexesAfter" : 3,  
  "ok" : 1  
}
```



Where is my *Slow Query Log*?

MySQL

```
mysql> set global long_query_time = 0.1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> set global slow_query_log = 1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> show global variables like
'slow_query_log_file';
```

Variable_name	Value
slow_query_log_file	/var/lib/mysql/thor-slow.log

1 row in set (0.00 sec)

MongoDB

```
db.setProfilingLevel(level, slowms)
```

Level: 0 for no profiling, 1 for only slow operations, or 2 for all operations.

Slowms = long_query_time but in milliseconds

```
> db.setProfilingLevel(2, 100);
{ "was" : 0, "slowms" : 100, "ok" : 1 }
```

```
> db.system.profile.find( { millis : { $gt : 100 } }
).pretty()
{
  "op" : "query",
  "ns" : "zips.zips",
  "query" : {
    "city" : "DURHAM"
  },
  "ntoreturn" : 0,
```

From



to



Export from MySQL 5.7:

```
mysql> SELECT JSON_OBJECT('name', replace(name, '"', ''), 'other_tags',  
replace(other_tags, '"', ''), 'geometry', st_asgeojson(shape)) as j  
      FROM `points` INTO OUTFILE '/var/lib/mysql-files/points.json';
```

Query OK, 13660667 rows affected (4 min 1.35 sec)

From to

Load to MongoDB (parallel):

```
mongoimport --db osm --collection points -j 24 --file  
/var/lib/mysql-files/points.json
```

```
2016-04-11T22:38:10.029+0000    connected to: localhost  
2016-04-11T22:38:13.026+0000    [.....] osm.points  31.8 MB/2.2 GB (1.4%)  
2016-04-11T22:38:16.026+0000    [.....] osm.points  31.8 MB/2.2 GB (1.4%)  
2016-04-11T22:38:19.026+0000    [.....] osm.points  31.8 MB/2.2 GB (1.4%)  
...  
2016-04-11T23:12:13.447+0000    [#####] osm.points  2.2 GB/2.2 GB (100.0%)  
2016-04-11T23:12:15.614+0000    imported 13660667 documents
```

Thinking about using MongoDB ?

Consider trying out Percona Server for MongoDB

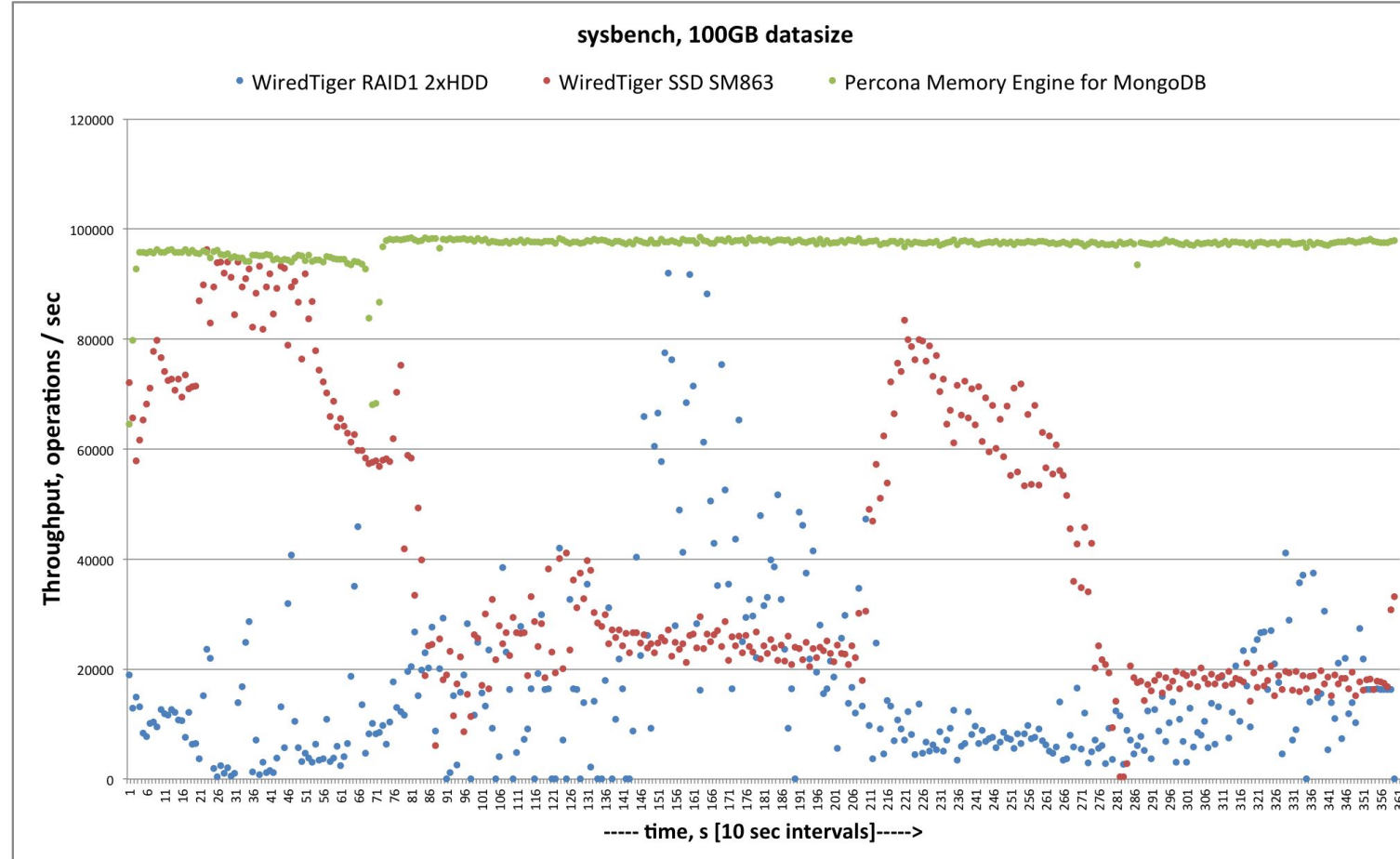


PERCONA
Server for MongoDB

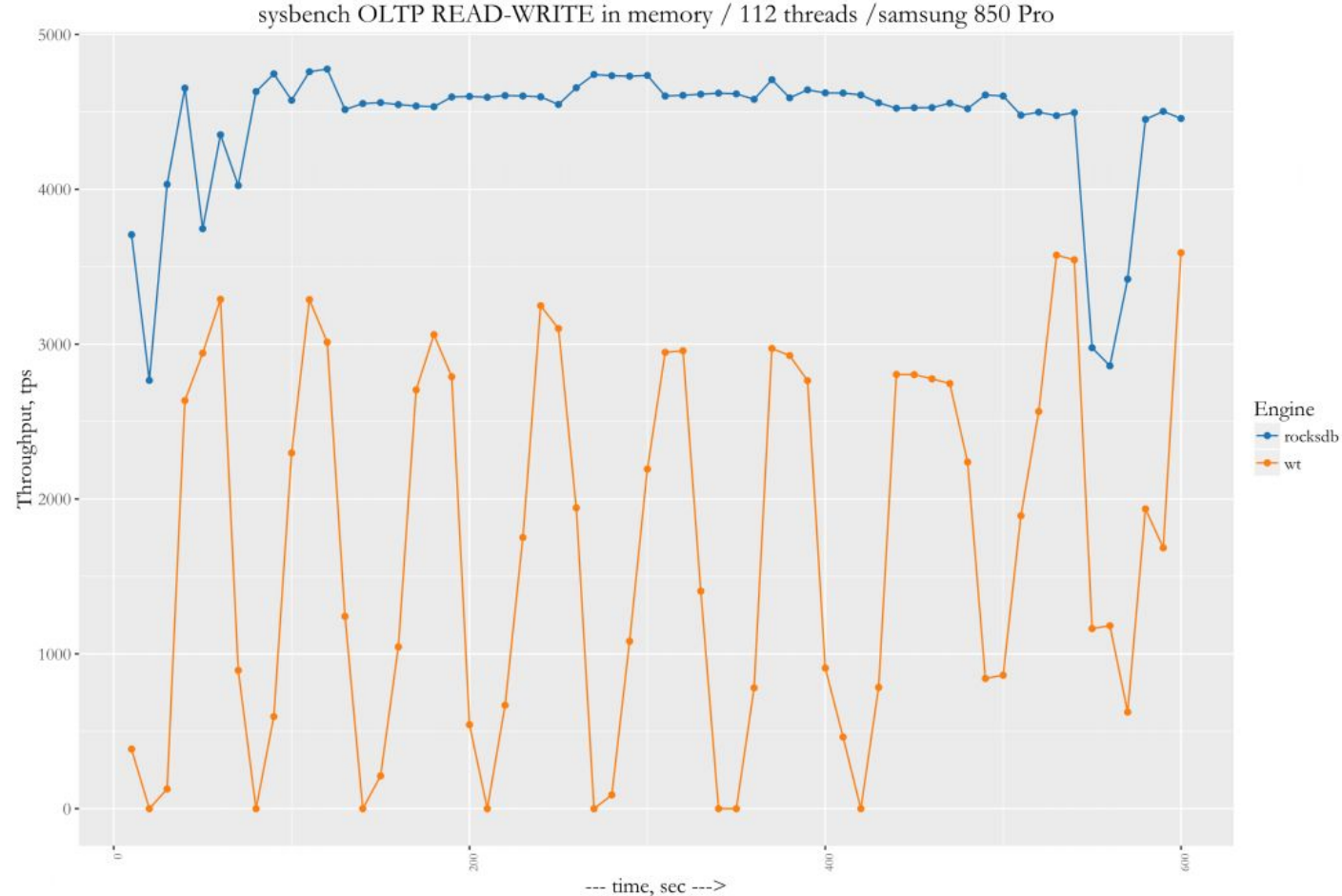
Percona Server for MongoDB 3.4

- **100% Compatible with MongoDB 3.4 Community Edition**
- **Open Source with Alternatives to many MongoDB Enterprise Features**
- **MongoRocks (RocksDB) and Percona Memory Engine**
- **New: Sensitive Data Masking**
- **New: Query Sampling**
- **New: Hot Backup for WiredTiger and MongoRocks**

Percona Memory Engine for MongoDB Benchmarks



WiredTiger vs MongoRocks – write intensive





Database Performance Matters