Optimizer Standoff

MySQL 5.6 vs MariaDB 5.5

Peter Zaitsev, Ovais Tariq
Percona Inc
April 18, 2012

Thank you Ovais Tariq

- Ovais Did a lot of heavy lifting for this presentation
- He could not come to talk together with me because of Visa issues

Agenda

- Look at MySQL Optimizer Background
- Review History of MySQL Optimizer
- Provide Overview of MySQL optimizer in MySQL 5.6 and MariaDB 5.5
- Look into benchmarks

MySQL Optimizer Background

Anatomy of Query Execution

- Parsing
- Optimizer choses how to execute the query
- Executioner Executes chosen strategy
- In MySQL these steps are not always clearly separated

Keys to great Query Performance

- Having right execution methods
  - For Optimizer to pick
- Having smart optimizer to pick right methods
- Having accurate statistics so estimates are close to reality

How Optimizer Works

- Uses Cost model to evaluate query plans
- Optimizer Looks for Plan with Lowest Cost
- Lowest cost is not always fastest queries
  - Cost model is coarse approximation of reality
- Statistics inaccurate -> cost is not accurate too
- Optimizer looks to avoid very bad plans

Changes to MySQL Optimizer

- Changes to MySQL Optimizer are always scare
- Even small ones between major versions hurt
- There are always queries which end up worse
- Optimizer Benefit Asymmetry
  - 99% cases of queries are Better and 1% is Worse might not be good enough

MySQL Optimizer Status and History

MySQL Optimizer Status

- Rather “simple” optimizer
  - Designed to optimize query every time it is ran
  - Prepared statements are still exception
- Naïve Execution methods
- Good enough for Simple Queries

MySQL Optimizer Recent History

- MySQL 6.0 was planned for include a lot of Optimizer Improvements
- MySQL 6.0 was abandoned
- MariaDB and Oracle teams have ported code from MySQL 6.0 to MariaDB 5.3 and MySQL 5.6 respectfully
- Both Teams doing original development now

Look at the Teams

- MariaDB Optimizer team lead by Igor Babaev
  - Majority of Original MySQL Optimizer team
- MySQL 5.6 Optimizer Team
  - Mainly based in Norway
  - Coming from Sun, worked on different database technology before

Optimizers Comparison

General Notes

- MySQL 5.6.4 is not GA
  - GA version of MySQL 5.6 may include more optimizations
- MySQL 5.6 benefits from general improvements as well
  - Though we do not expect large impact as we're not testing high concurrency
- All tests done for Innodb Tables

Optimizer Features at Glance

Feature	MySQL 5.6	MariaDB 5.5
Multi Range Read (MRR) – reading rows in PK order	Yes	Yes
Key Ordered Scan (Secondary Keys)	No	Yes
Index Condition Pushdown (ICP)	Yes	Yes
Improved “Filesort” handling ORDER BY … LIMIT	Yes	No
Table Elimination Optimization	No	Yes
Index_merge sort_intersection	No	Yes
Improved choice between range and index_merge	No	Yes

Subquery Optimization

Feature	MySQL 5.6	MariaDB 5.5
Semi-join Subquery Optimizations (several strategies)	No	Yes
Derived Table Subquery Optimization	Yes	Yes
Derived Tables with Keys	Yes	Yes
Subquery Cache	No	Yes

Join Optimizations

Feature	MySQL 5.6	MariaDB 5.5
Block Nested Loop (BNL) for Outer Join	Yes	Yes
Batch Key Access (BKA) / Block Index Join	Yes	Yes
Block Nested Loop Hash (BNLH)/ Block Index Hash Join	No	Yes
Batch Key Access Hash (BKAH) Join	No	Yes

Non Performance Optimizer Features

Feature	MySQL 5.6	MariaDB 5.5
Optimizer Tracing	Yes	No
EXPLAIN for INSERT/UPDATE/DELETE	Yes	No
No Materialization of subqueries in EXPLAIN	Yes	Yes
Persistent Statistics for Innodb	Yes	No
JSON Detailed EXPLAIN	Yes	No

Benchmarks

lies damn lies and Benchmarks

About Benchmark

- Single thread run
  - Did not look how different versions scale with concurrency
- Using MySQL 5.5 as baseline
- Some benchmarks are focused looking at given optimizations
  - Might not be reporting best performance query could get
- Cold runs and Hot Runs

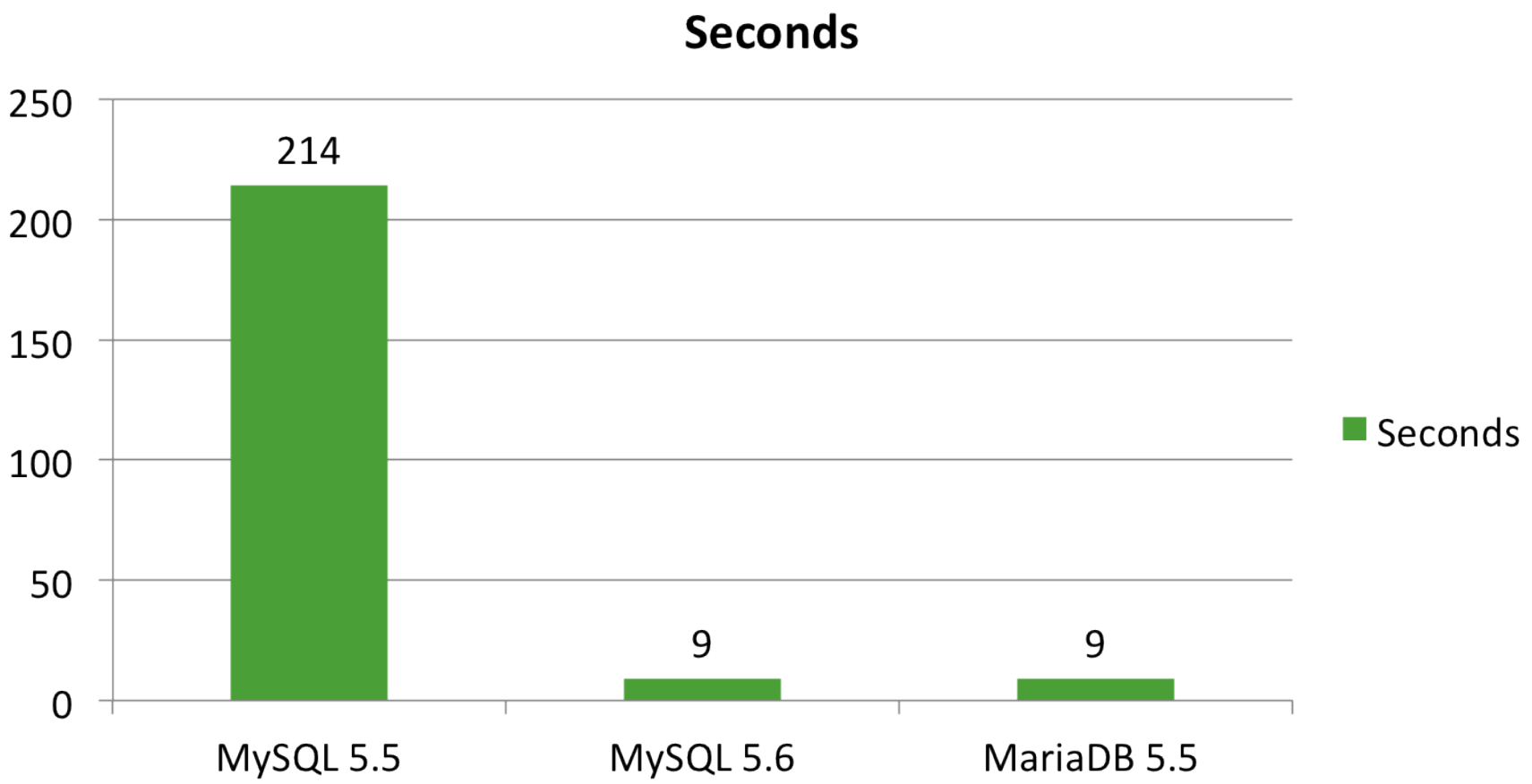
Benchmark Schema

- Use TPC-H/DBT3 schema for most benchmarks
- Use queries from this benchmarks when possible
- Added indexes and used Index hints as needed
  - Some optimizations would work great but MySQL/MariaDB would not pick them by itself

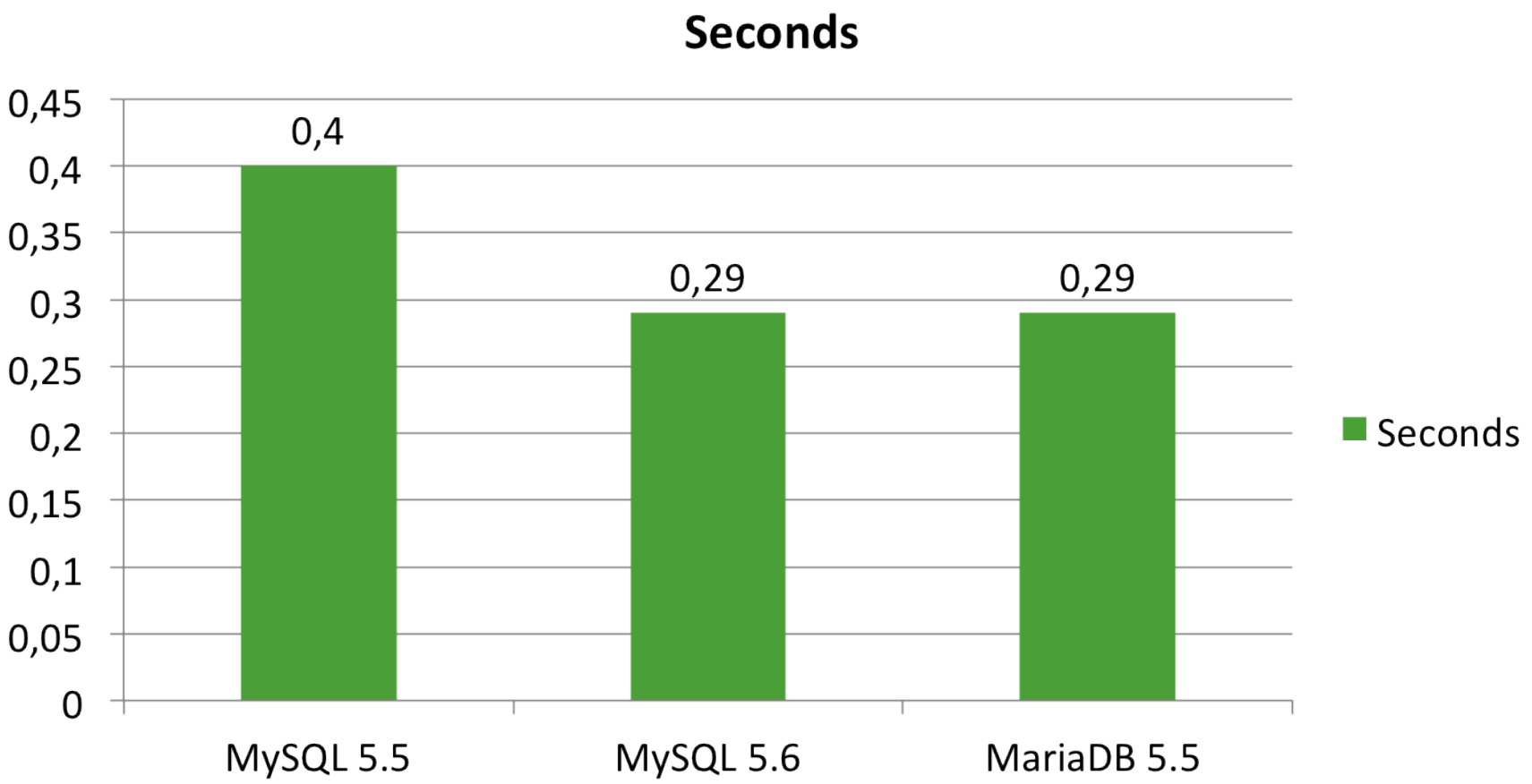
Benchmark Hardware/OS

- 2x Intel Core 6600 @ 2.4Ghz
- CentOS 6.2 (64bit)
- 8GB of Memory
- Innodb Storage Engine
- 8GB Buffer Pool
- Software RAID5, 4*5.4K drives
- XFS filesystem

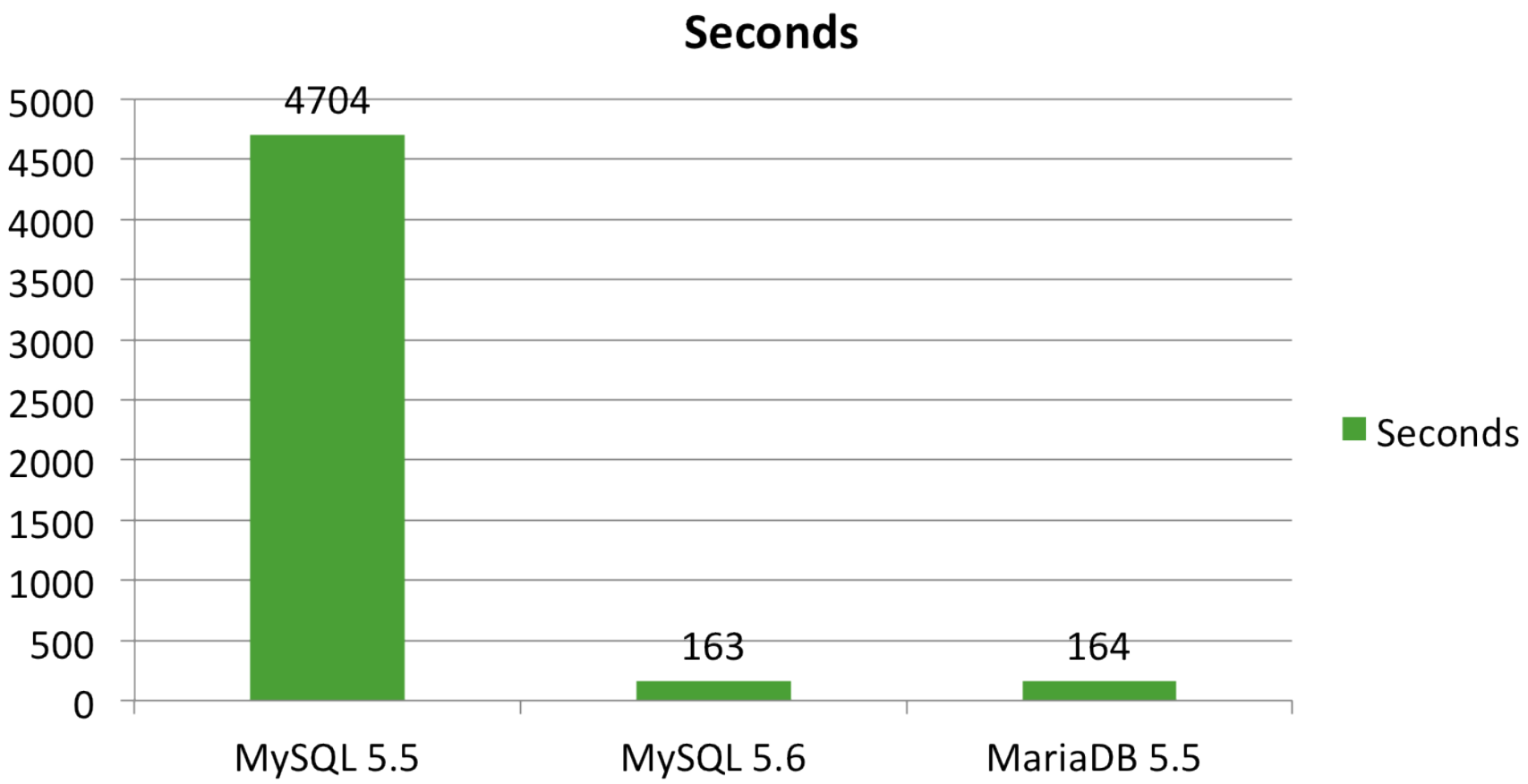
ICP, Query #19 5GB data size (Cold)



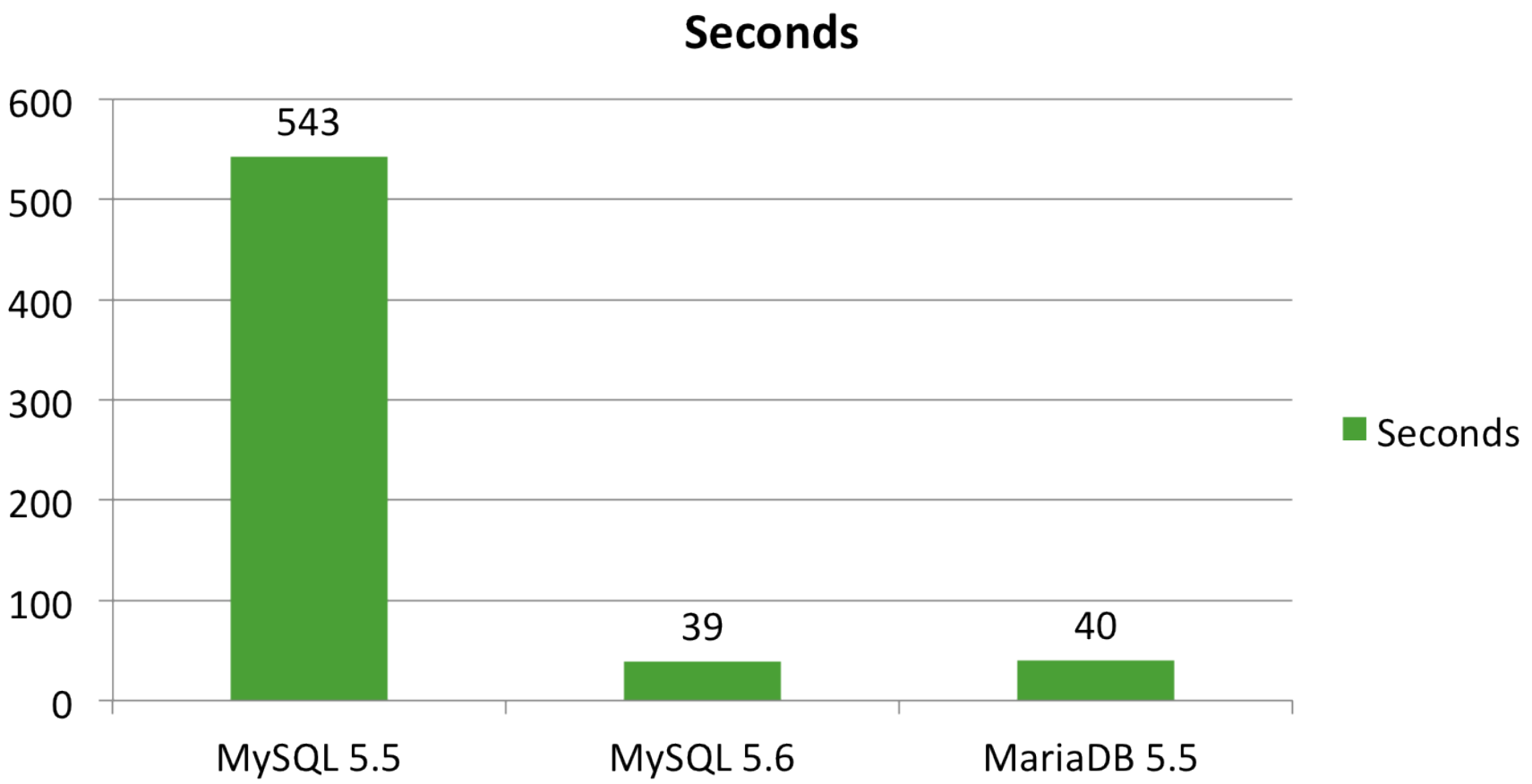
ICP, Query #19 5GB data size (Warm)



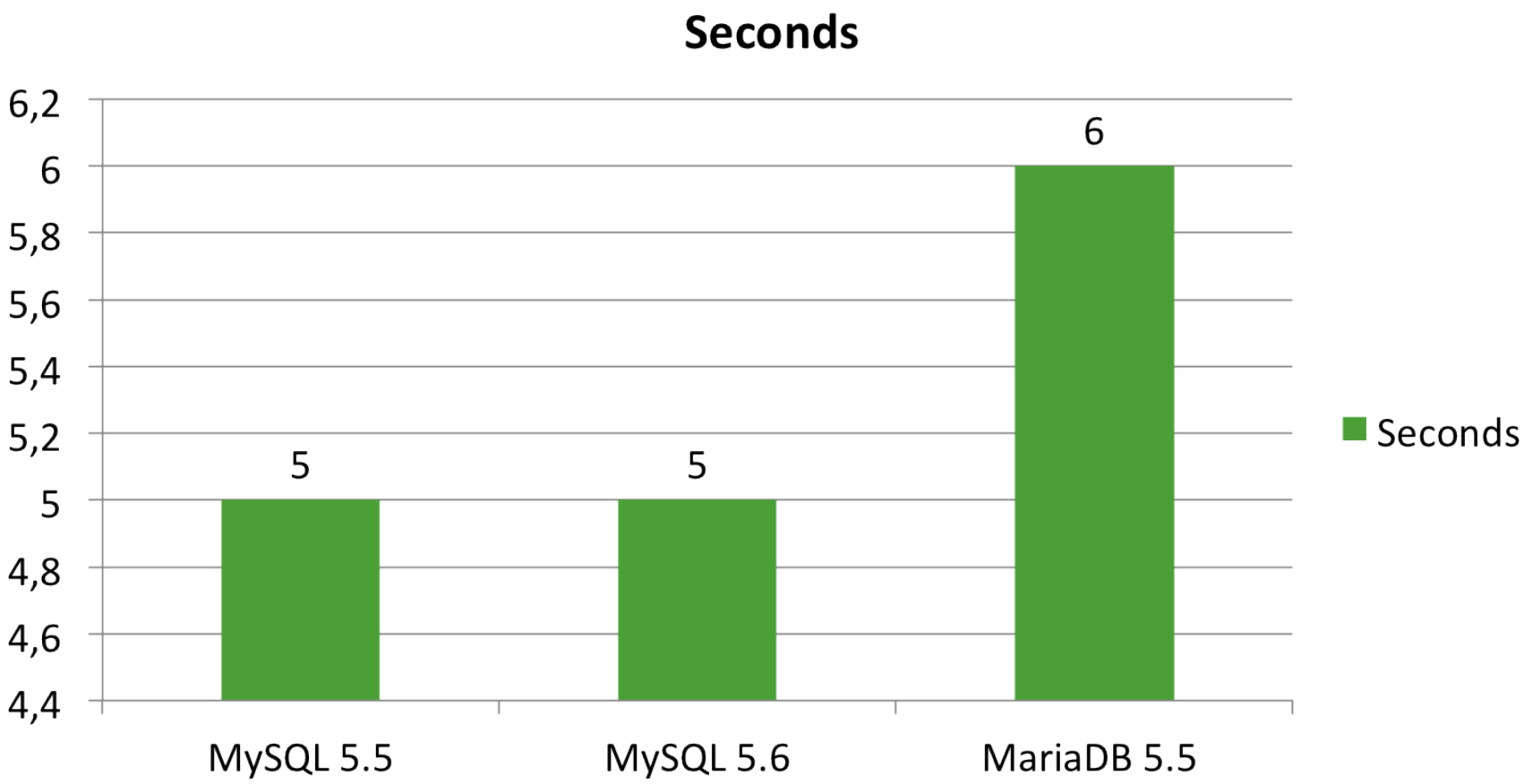
ICP, Query #19 95GB data size (Cold)



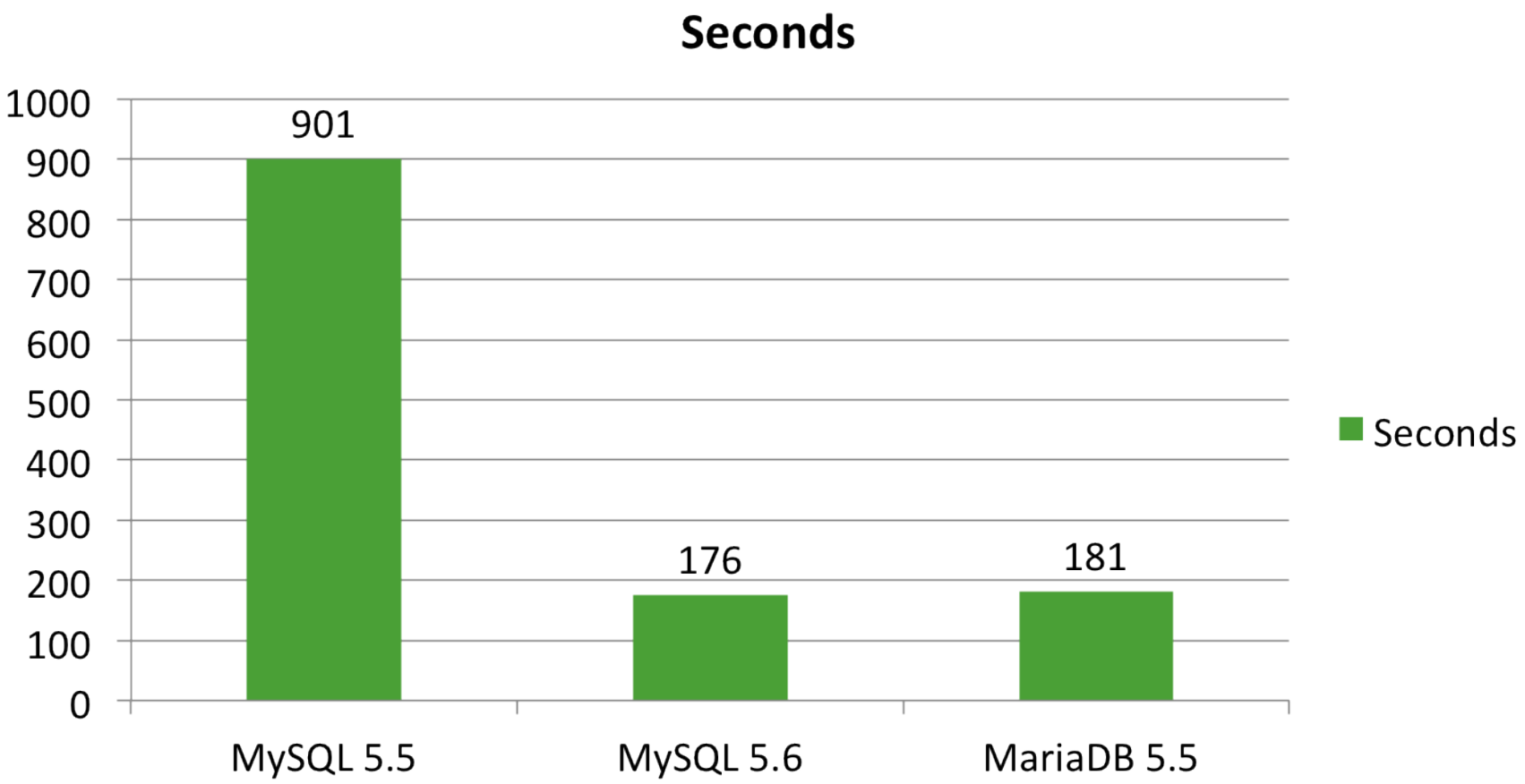
MRR, Query #10 5GB data size (Cold)



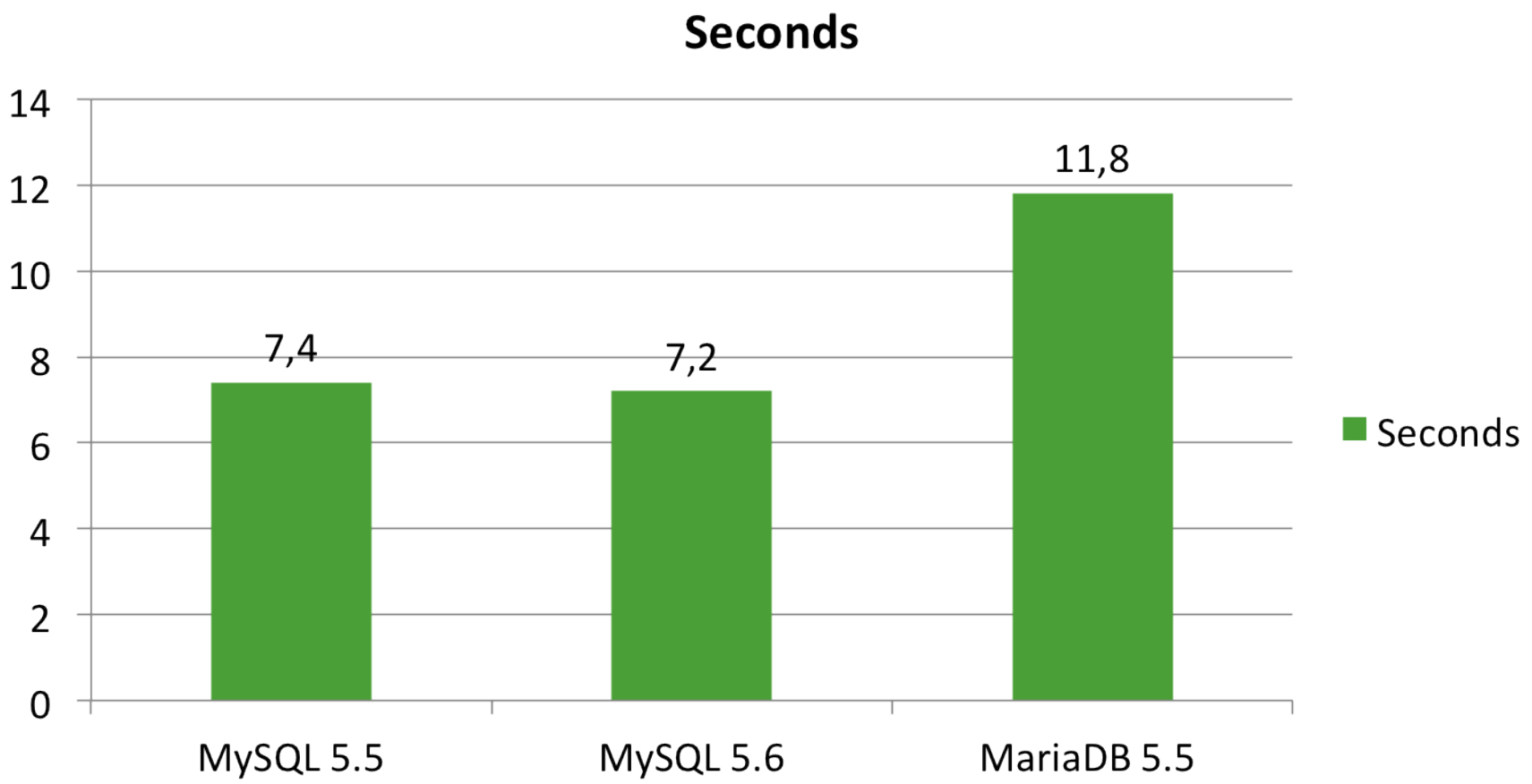
MRR, Query #10 5GB data size (Warm)



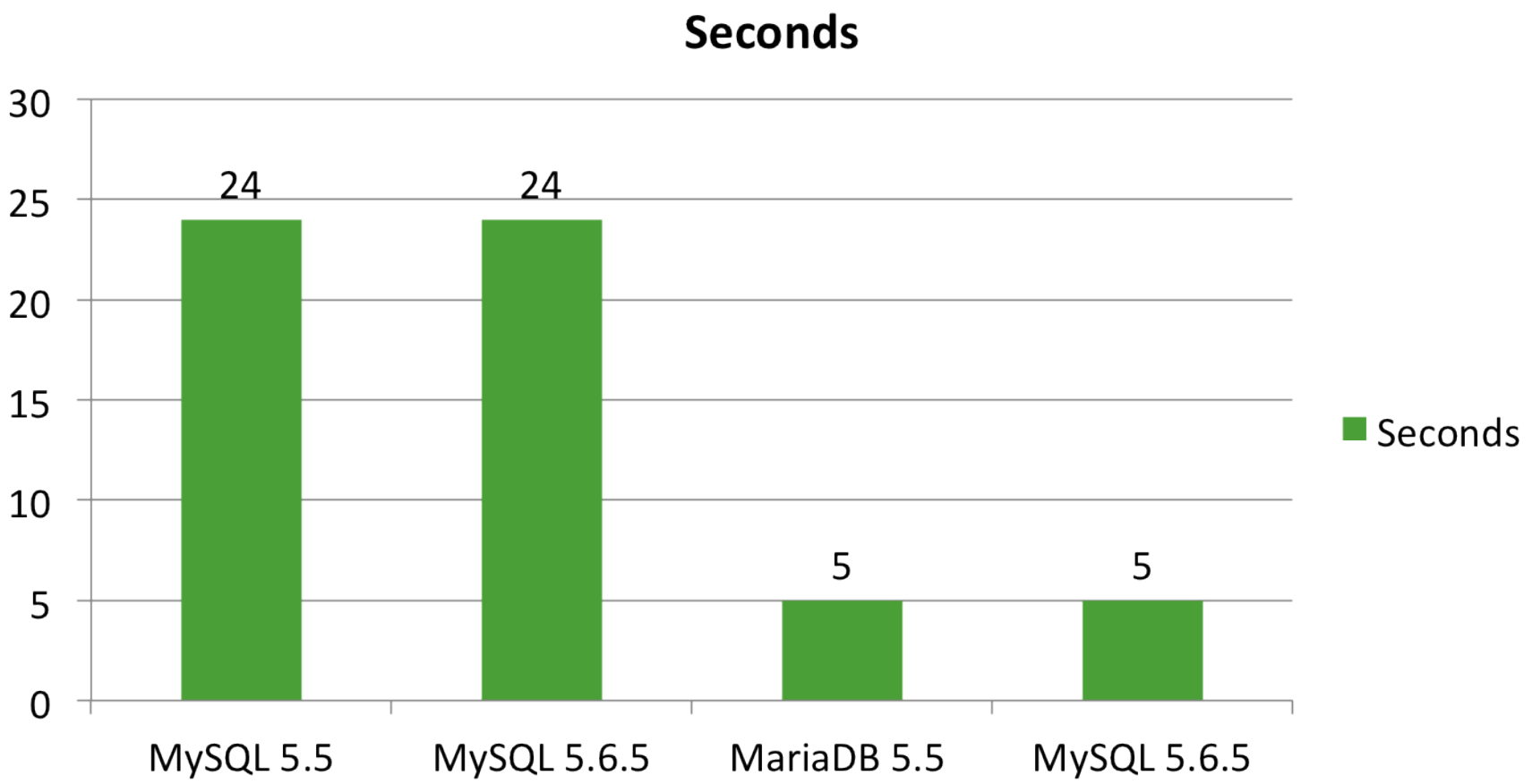
BKA, Query #3 5GB data size (Cold)



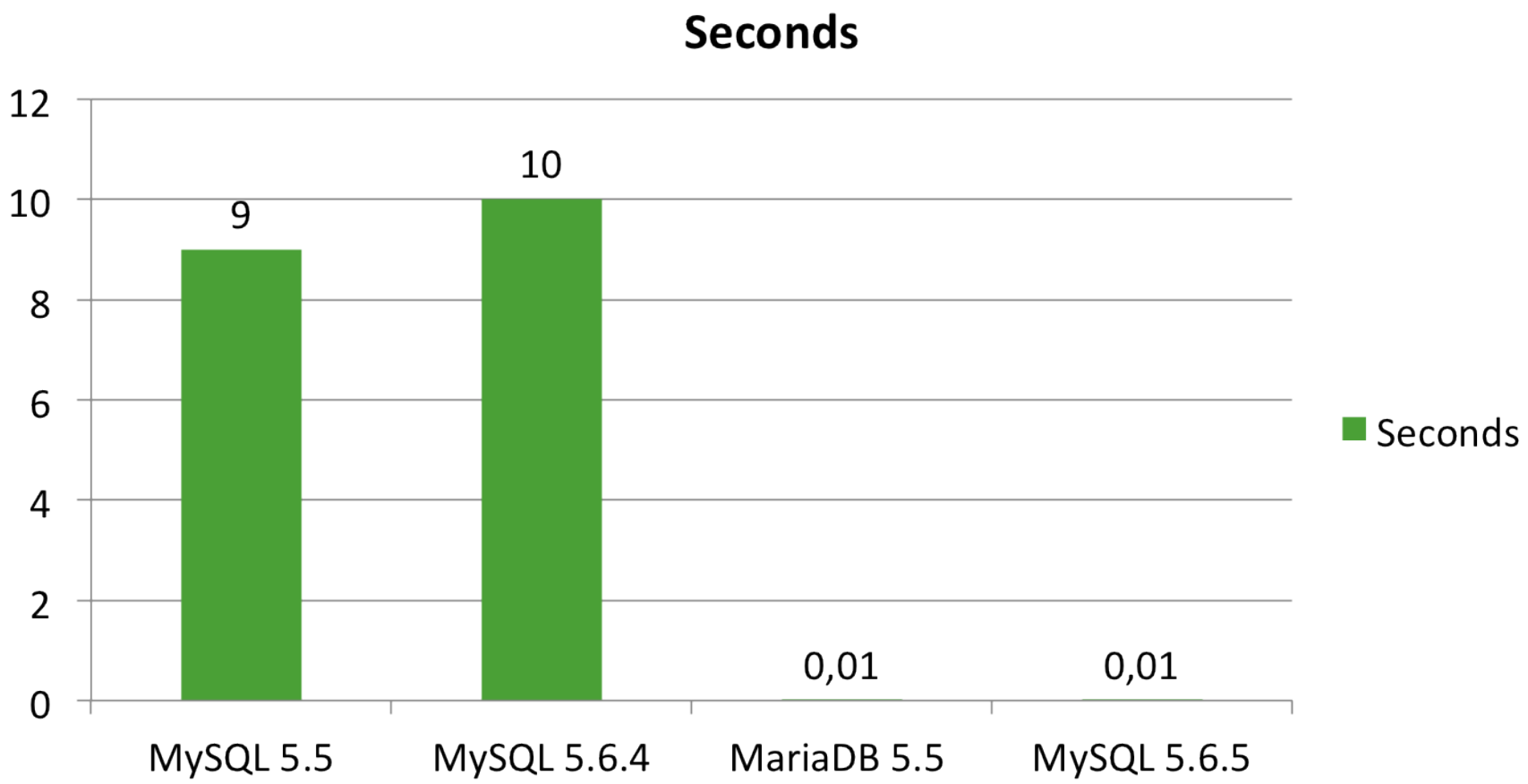
BKA, Query #3 5GB data size (Warm)



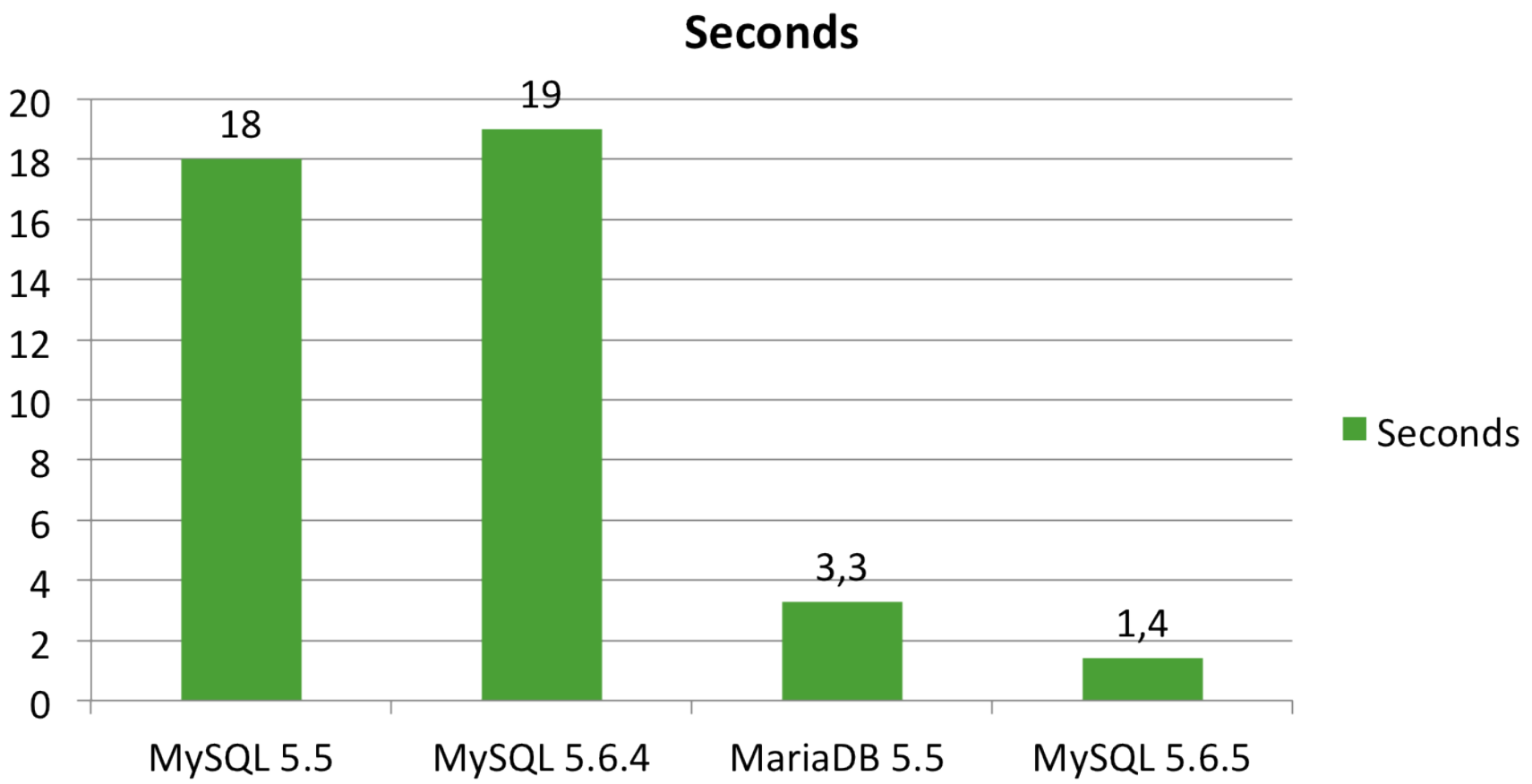
Subquery – Table Pullout Optimization (Cold)



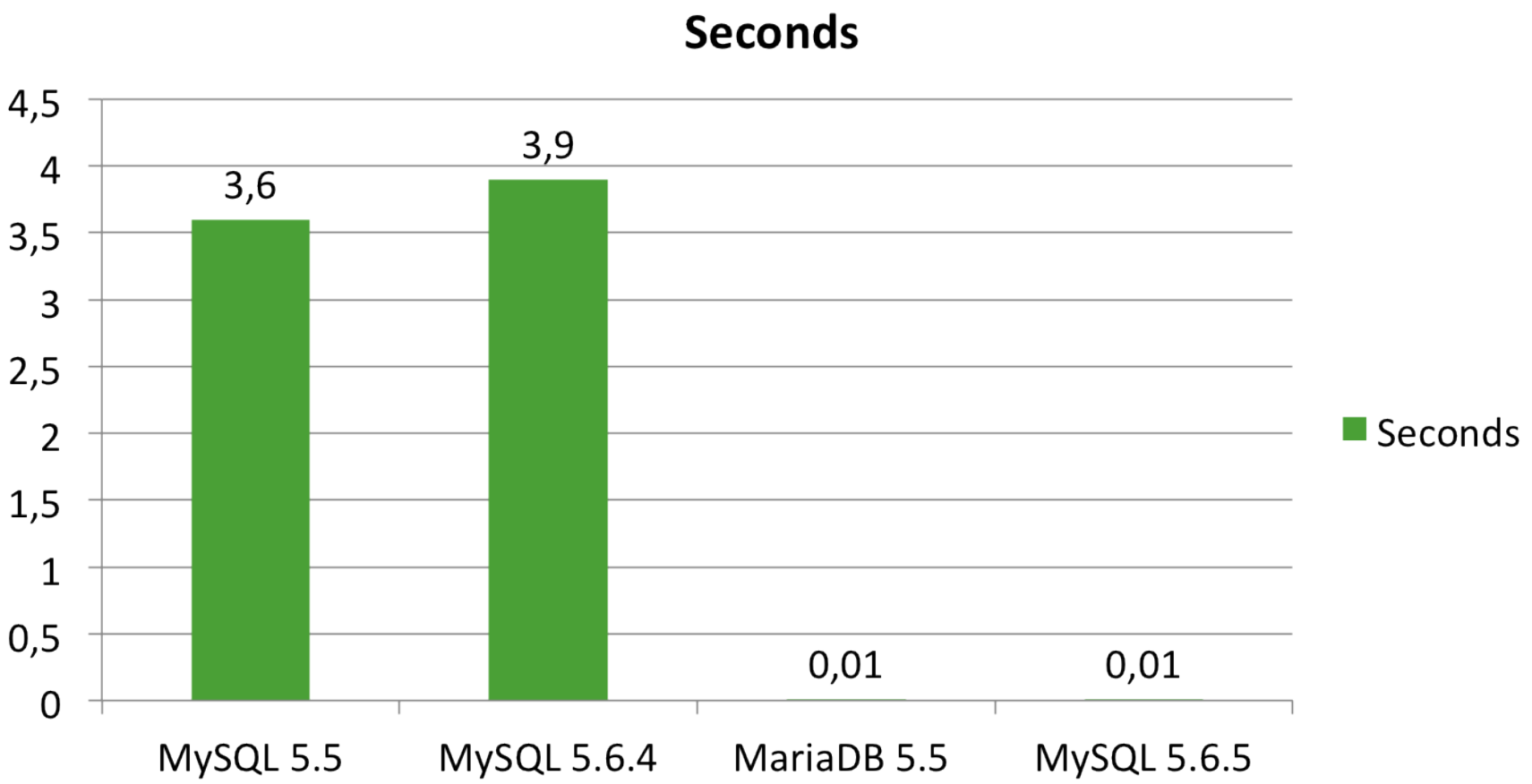
Subquery – Table Pullout Optimization (Warm)



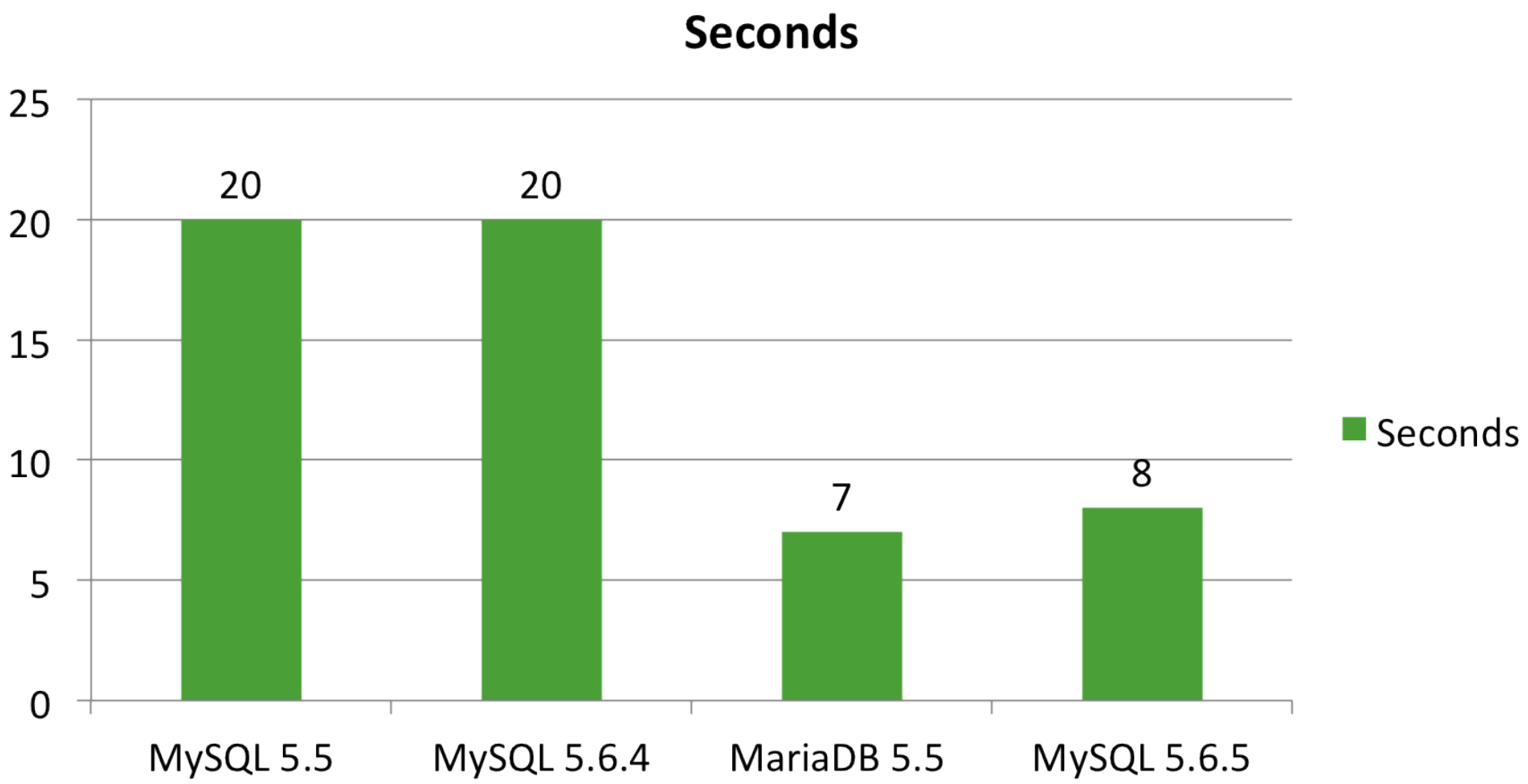
Subquery – Semi Join materialization (Cold)



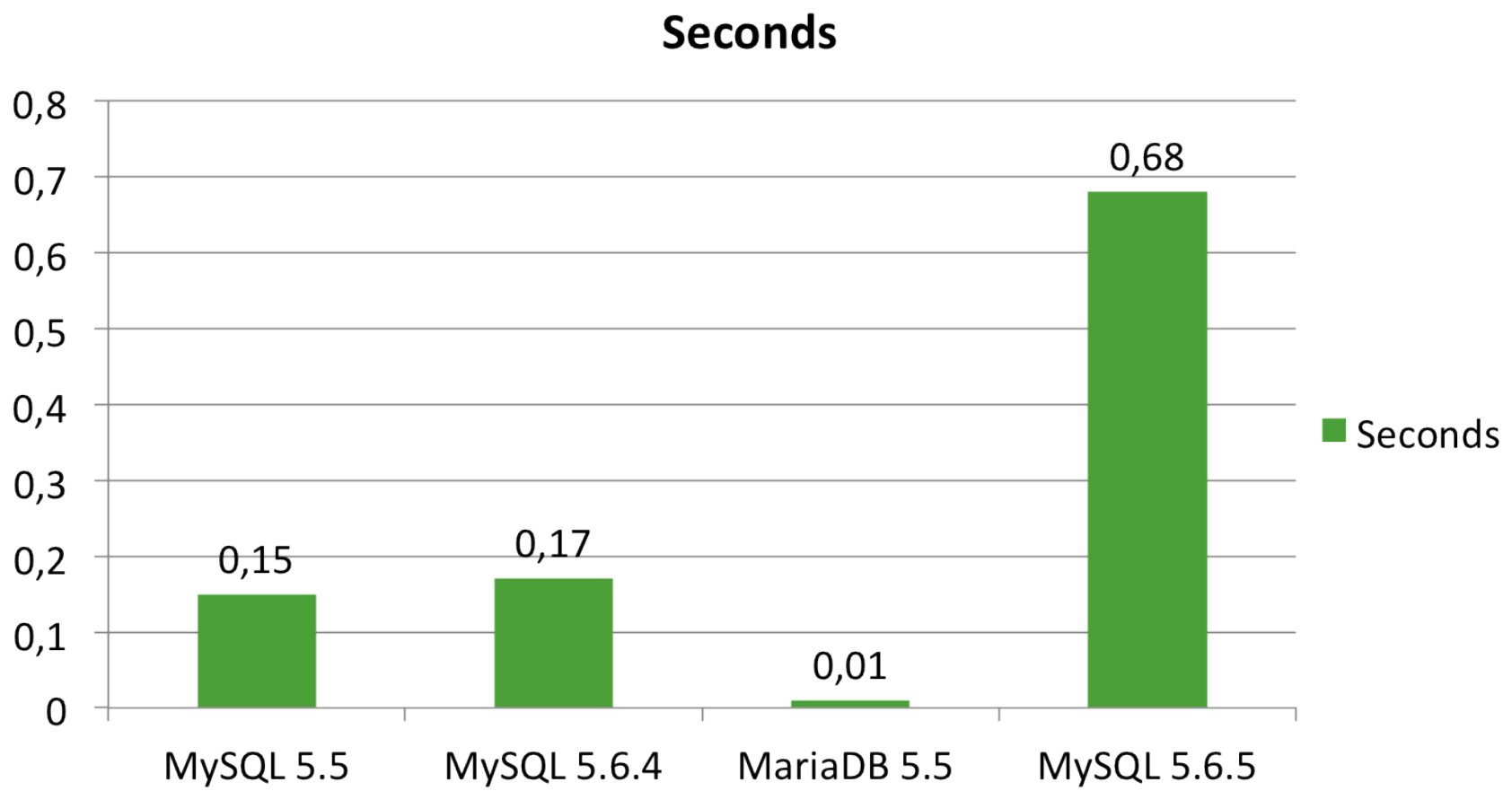
Subquery – Semi Join Materialization (Warm)



Subquery – Non Semi Join Optimization (Cold)



Subquery – Non Semi-Join Optimization (Warm)

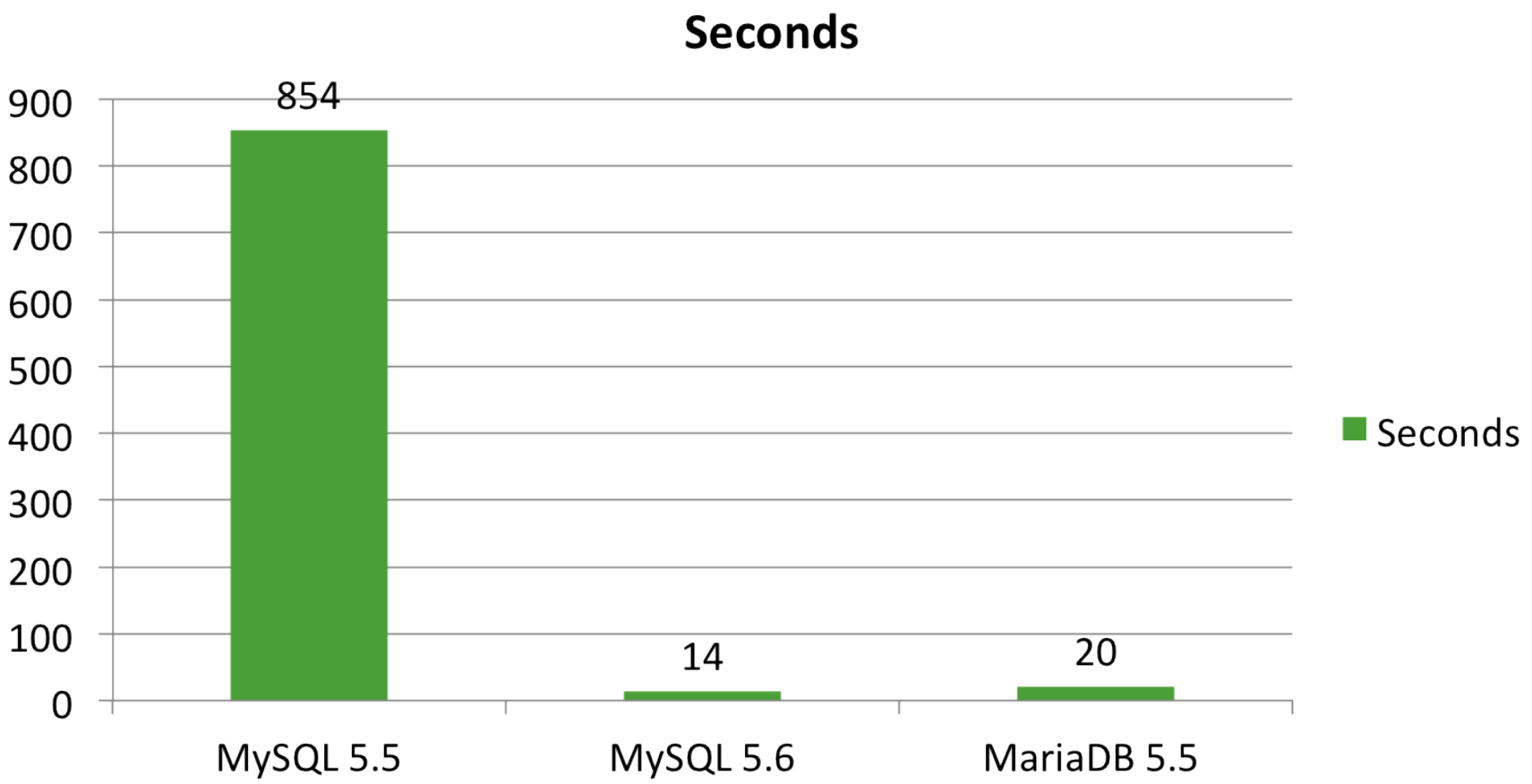


Derived Table Optimization

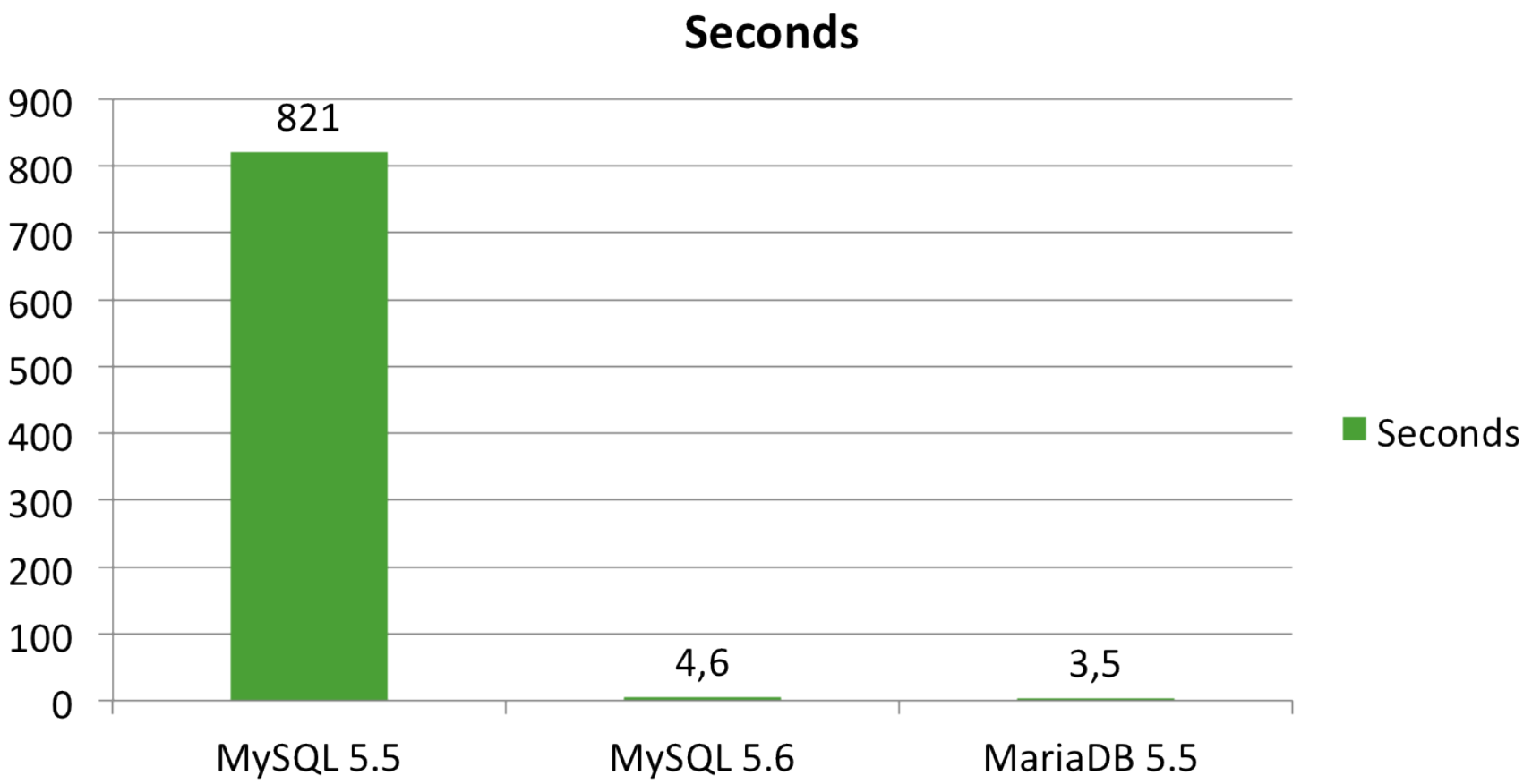
Explain on MySQL 5.5

```
mysql [localhost] {msandbox} (dbt3) > explain select s.* from supplier as s inner join
(select l_orderkey, l_shipdate, l_suppkey from lineitem where l_shipdate between '1992-06-01' and
'1992-12-31' group by l_orderkey) as l on s_suppkey=l_suppkey inner join
(select o_orderkey, o_totalprice from orders where o_orderdate between '1992-06-01' and '1992-12-
31') as o on l_orderkey=o_orderkey
where s_nationkey=24 and o_totalprice > 100000 group by s_suppkey order by o_totalprice limit 10;
+----+-------------+------------+--------+---------+-----------------------------------------------+
| id | select_type | table      | type   | rows    | Extra                                         |
+----+-------------+------------+--------+---------+-----------------------------------------------+
|  1 | PRIMARY     | <derived3> | ALL    |  132551 | Using where; Using temporary; Using filesort |
|  1 | PRIMARY     | <derived2> | ALL    |  170837 | Using where; Using join buffer               |
|  1 | PRIMARY     | s          | eq_ref |       1 | Using where                                   |
|  3 | DERIVED     | orders     | ALL    | 1492284 | Using where                                   |
|  2 | DERIVED     | lineitem   | index  | 5980955 | Using where                                   |
+----+-------------+------------+--------+---------+-----------------------------------------------+
```

Derived Table (Cold)



Derived Table (Warm)

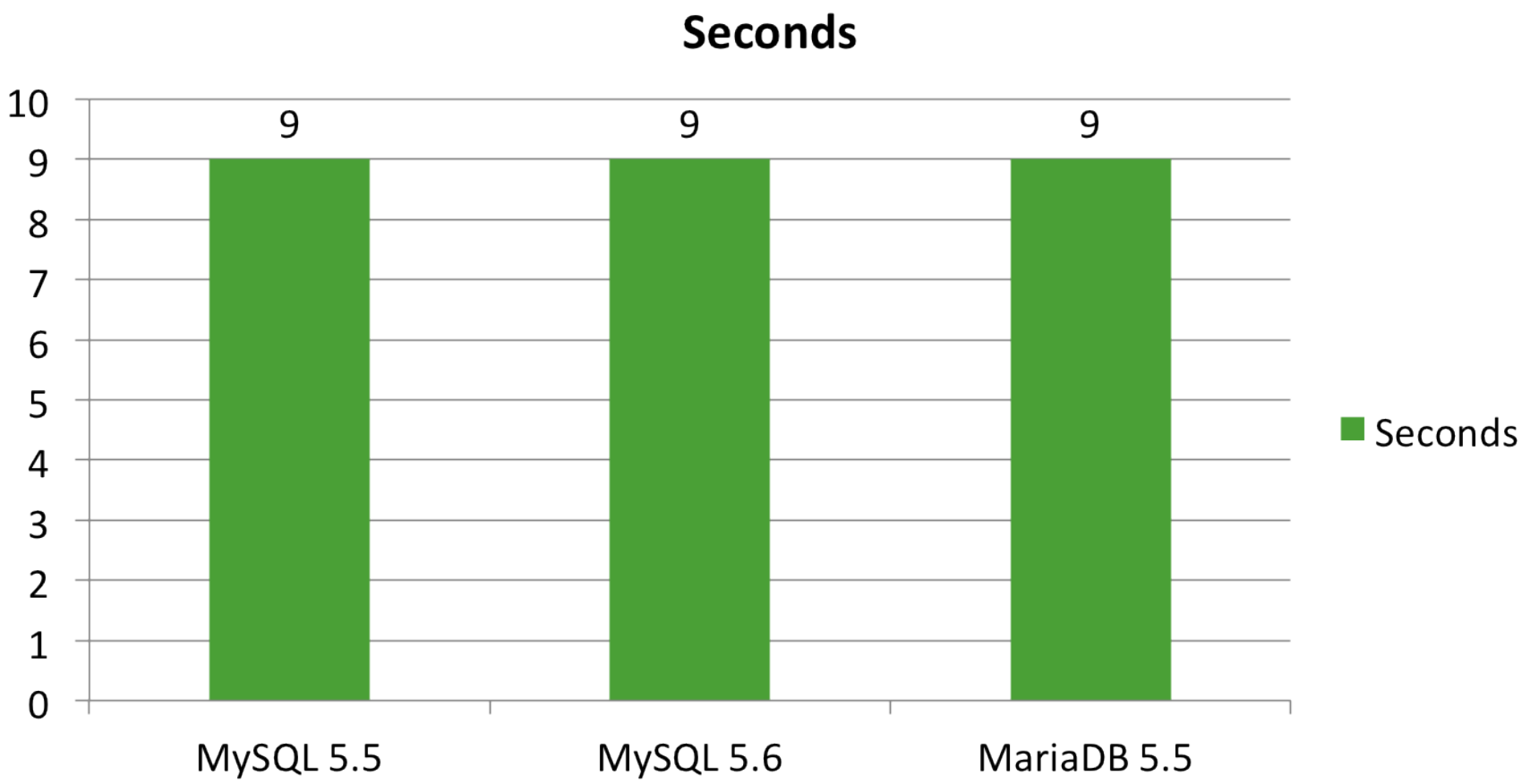


Filesort Optimization Query

Explain MySQL 5.5

```
mysql [localhost] {msandbox} (dbt3) > explain select sql_calc_found_rows * from lineitem order by l_discount desc
limit 100, 100;
+----+-------------+----------+------+---------------+------+---------+------+---------+----------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows    | Extra          |
+----+-------------+----------+------+---------------+------+---------+------+---------+----------------+
|  1 | SIMPLE      | lineitem | ALL  | NULL          | NULL | NULL    | NULL | 5987386 | Using filesort |
+----+-------------+----------+------+---------------+------+---------+------+---------+----------------+
```

Filesort, no SQL_CALC_FOUND_ROWS (Warm)



Filesort, WITH SQL_CALC_FOUND_ROWS (Warm)

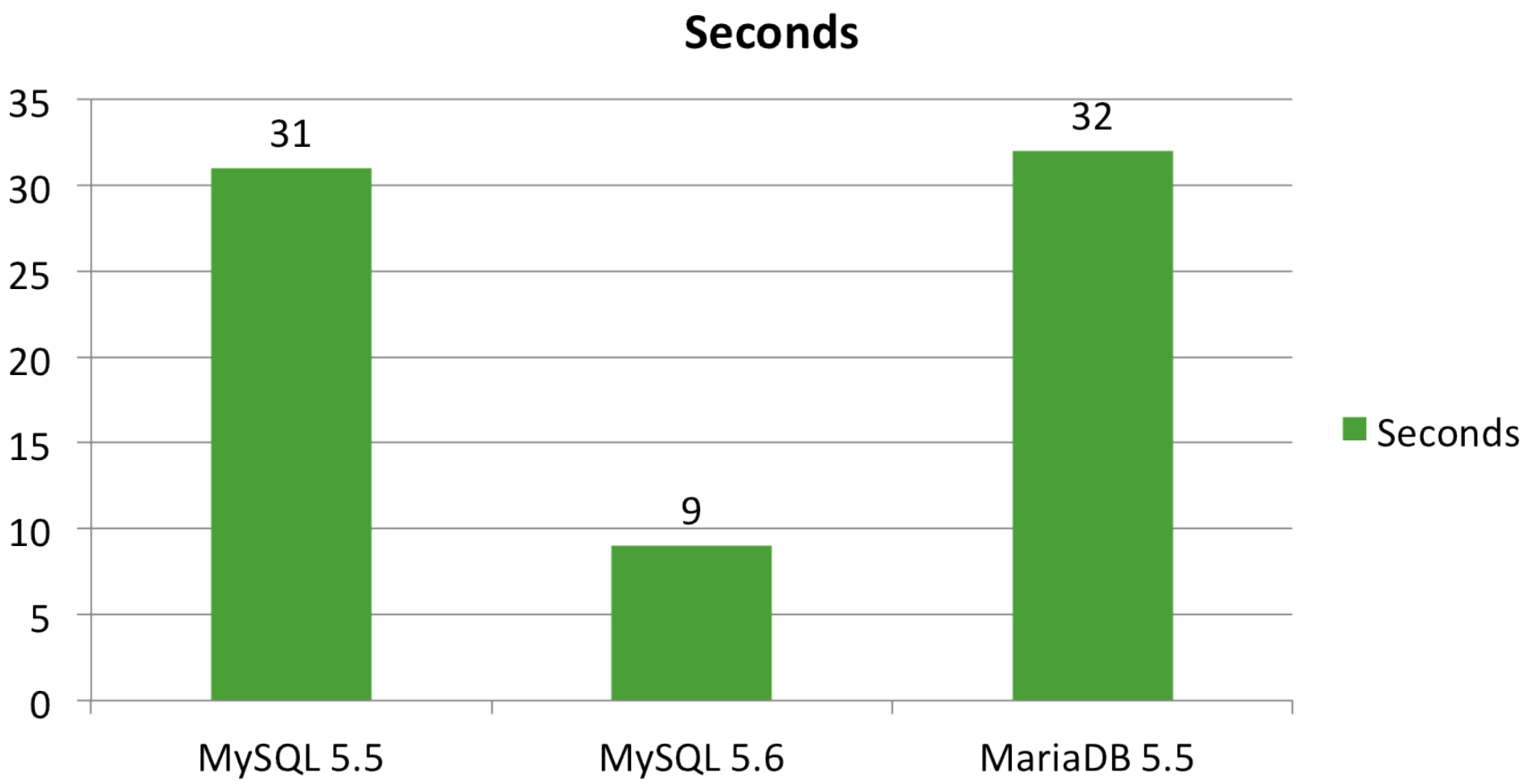


Table Elimination Example

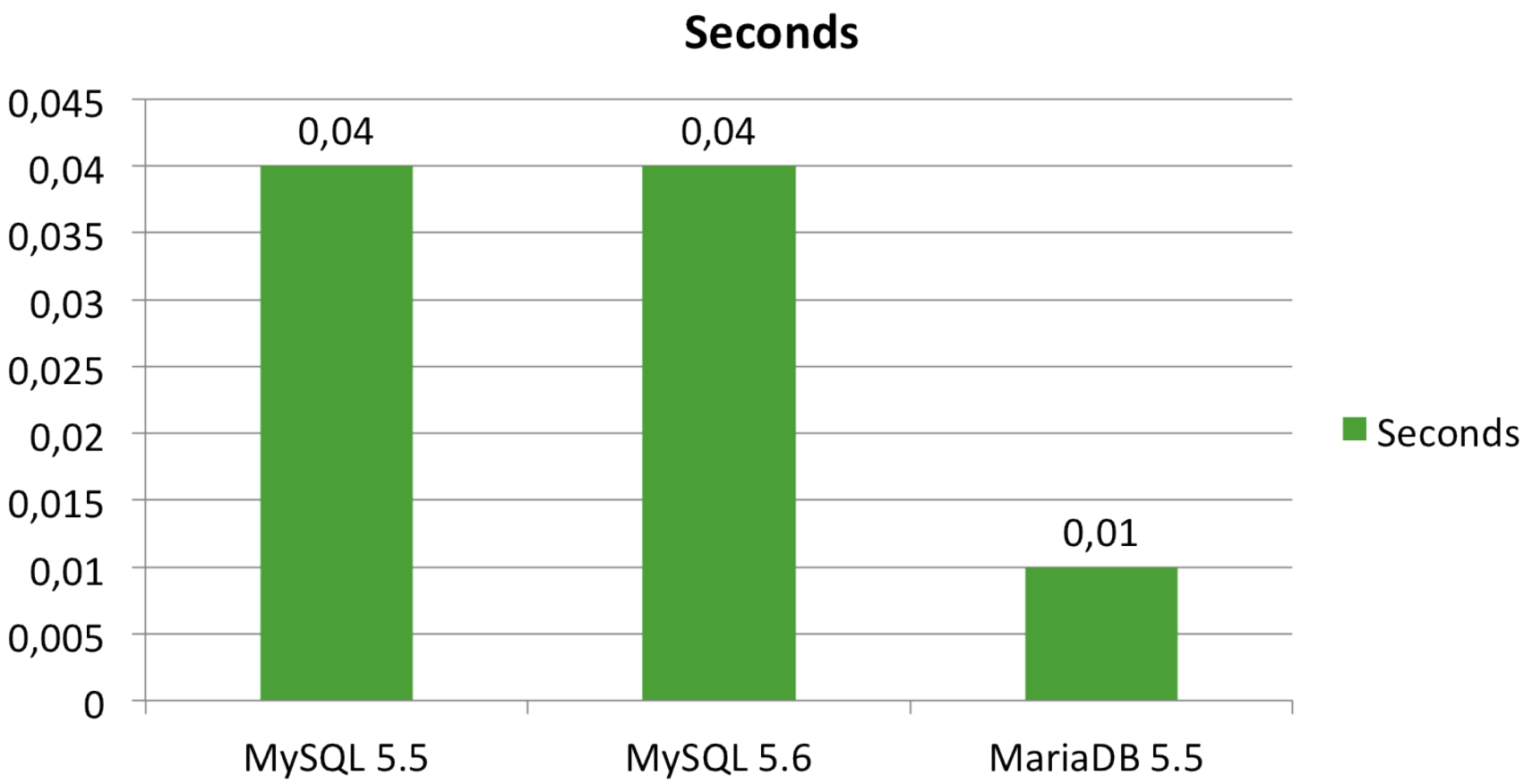
View:

```
create view v3
as
select *
from
customer left join
orders on c_custkey=o_custkey left join
lineitem on o_orderkey=l_orderkey left join
supplier on l_suppkey=s_suppkey;
```

Query:

```
select max(o_totalprice) from v3 where l_shipdate='1992-06-01';
```

Table Elimination (Warm)



Summary

- Both MySQL 5.6 and MariaDB has great optimizer improvements
- MariaDB optimizer is more advanced
- Comparable optimizations faster in MySQL 5.6
- Fine tuning where to chose which optimizer strategy is needed.

Thank You !

- pz@percona.com
- http://www.percona.com
- @percona
  - Use #perconalive
- http://www.facebook.com/Percona
- Slides will be published on the conference site