MySQL Query Optimization

Peter Zaitsev, CEO, Percona

March 8, 2019

SCALE 17x
Pasadena,CA

PERCONA

About Percona

Open Source Database Solutions Company

Support, Managed Services, Consulting, Training, Engineering

Focus on MySQL, MariaDB, MongoDB, PostgreSQL

Support Cloud DBaaS Variants on major clouds

Develop Database Software and Tools

Release Everything as 100% Free and Open Source

Widely Deployed Open Source Software

PERCONA Server for MySQL

5,000,000+ downloads

PERCONA Monitoring and Management

175,000+ downloads

PERCONA XtraBackup

4,500,000+ downloads

PERCONA Server for MongoDB

450,000+ downloads

PERCONA Toolkit

2,000,000+ downloads

PERCONA XtraDB Cluster

1,500,000+ downloads

About the Presentation

Cover the Basics

How MySQL Executes Queries

How To Find Queries to Optimize

How to Optimize Them

The Basics

Grand Goal

Application which Has a Great Performance

Great Performance Defined

Responds With Low Response Time

At All Times

For All Users

Response Time and Database

Database is not always at fault

Database Making your Application Slow

Dev Issues

Ops Issues

Dev Issues:

- Many Queries executed serially
- Expensive Queries
- Poorly Designed Queries
- Poorly Optimized Queries
- Saturation with Additional Load

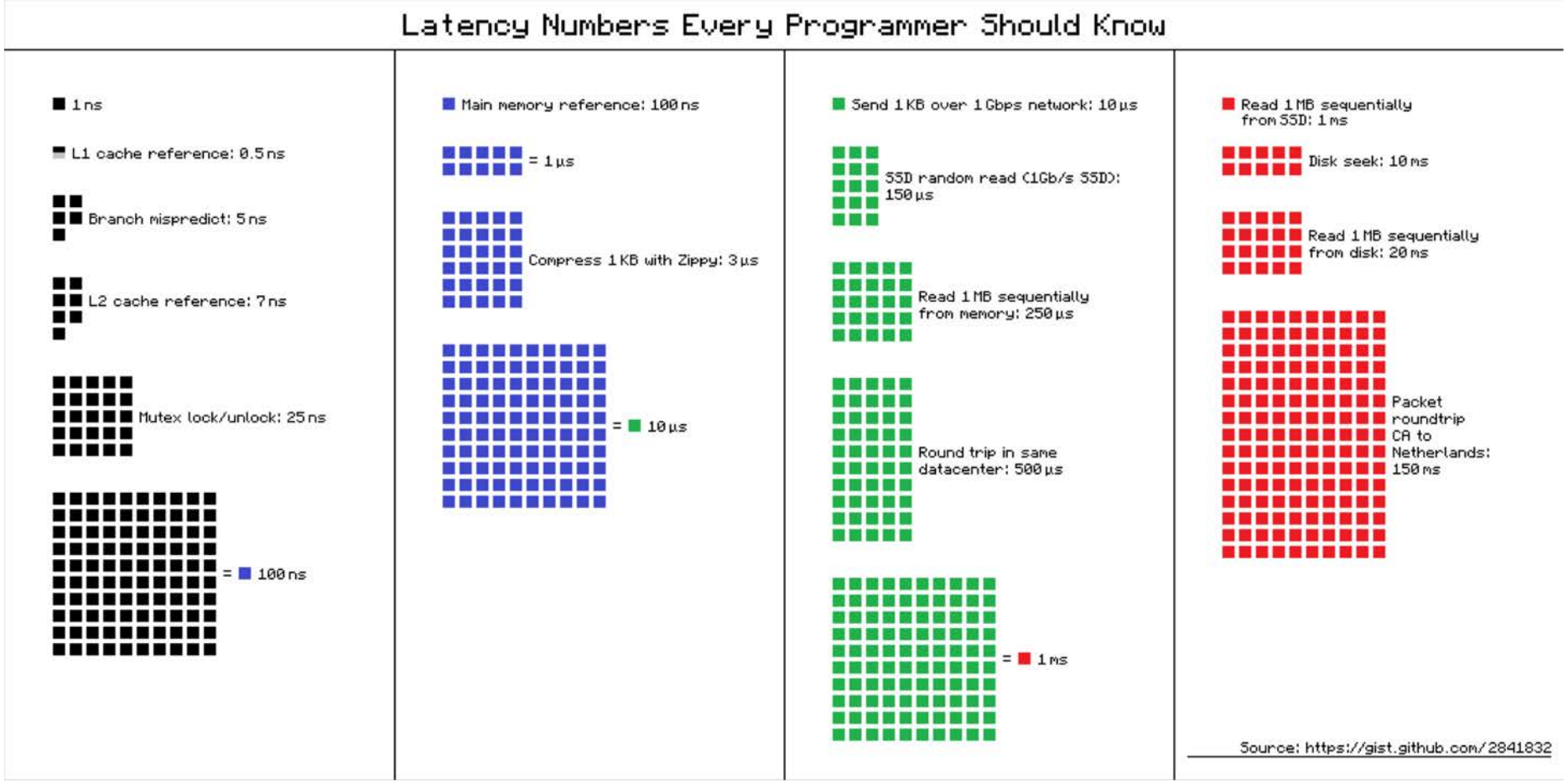
Ops Issues

Problems with System, Storage or Network

Saturation with Additional Load

Capacity Planning

Mind Network Latency



PERCONA

Query Optimization Goals

Specific User Interaction

Application As a Whole

Improving Efficiency

Assuring Scalability

Not Query Optimization Alone

General Architecture

Right Choice of Technology (Not Only MySQL)

Hardware/Instance Properties

OS and MySQL Configuration

Database Schema

How MySQL Executes Queries

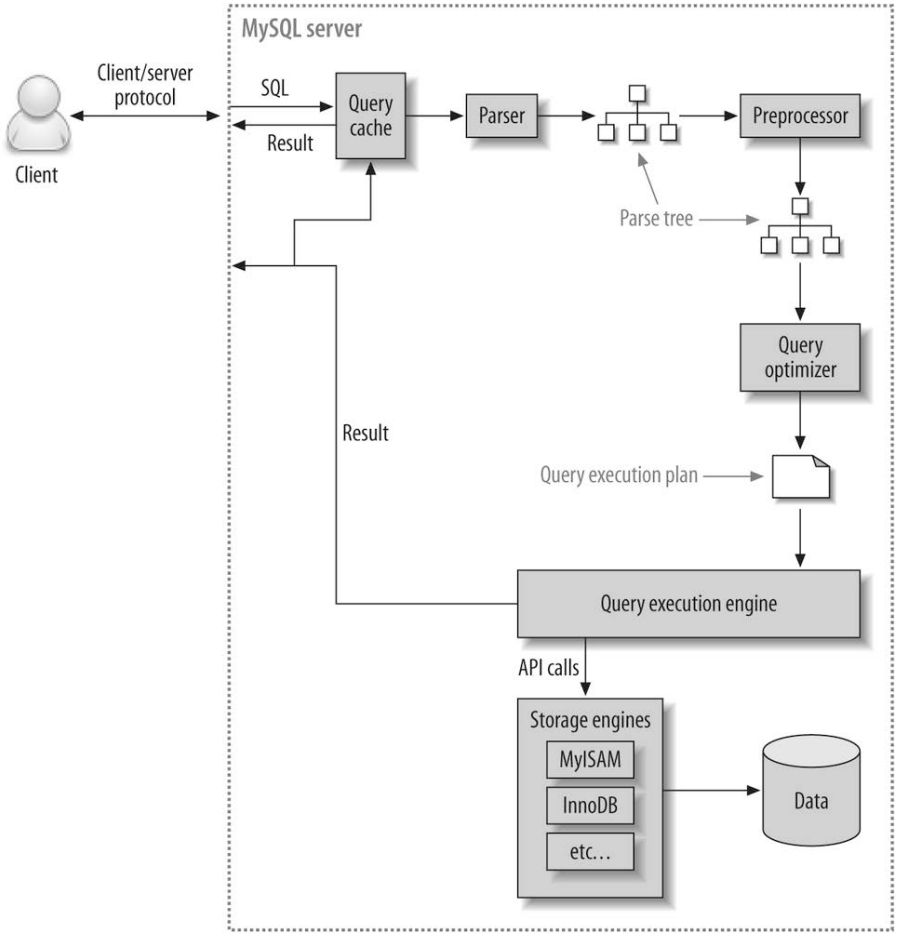
Execution Basics

Single Server

Single Thread (Using Single CPU Core)

No Intermediate Results Caching Between Query Executions

Query Execution Diagram



PERCONA

Added Complexities

UDFs (User Defined Functions)

Stored Programs

VIEWs

Use the LIMIT

Do not just stop fetching rows at the application side

MySQL Client-Server Protocol is NOT cursor based

Join Order

Permanent and “Derived” Tables are going to be “Joined in Order”

MySQL Starts from one table, finding all needed rows in it, and iterating finding matching rows from the next one

Join Order Is Critical For Performance

SELECT **STRAIGHT_JOIN** to force join order

Indexes

Proper Indexes are must have for Optimal Query Execution

Can improve Query Performance 1000x or more

Expensive to Maintain… so Do not Overdo

Covering Indexes to speed up data reads

Indexes are not Free

Space on Disk

Space in Memory

Extra Optimizer Load to Evaluate Them

Expensive to Maintain with Updates

Columns

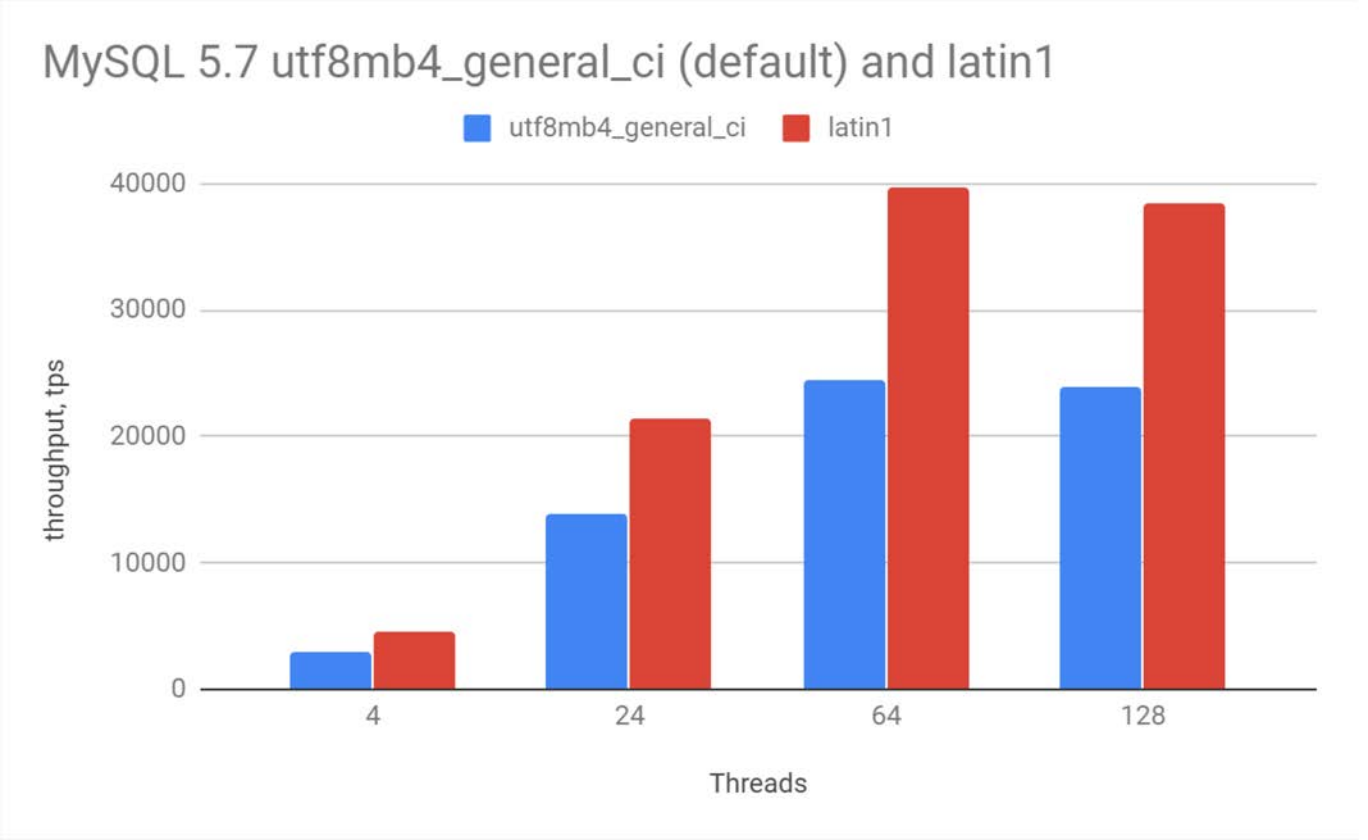
MySQL (Innodb, MyISAM, MyRocks etc) store data row by row

All columns must be read on every row access (excluding Blobs for Innodb)

Number of Total Columns, Their Size Impacts Query Performance a Lot

Covering Indexes are great to reduce amount of data query Touches

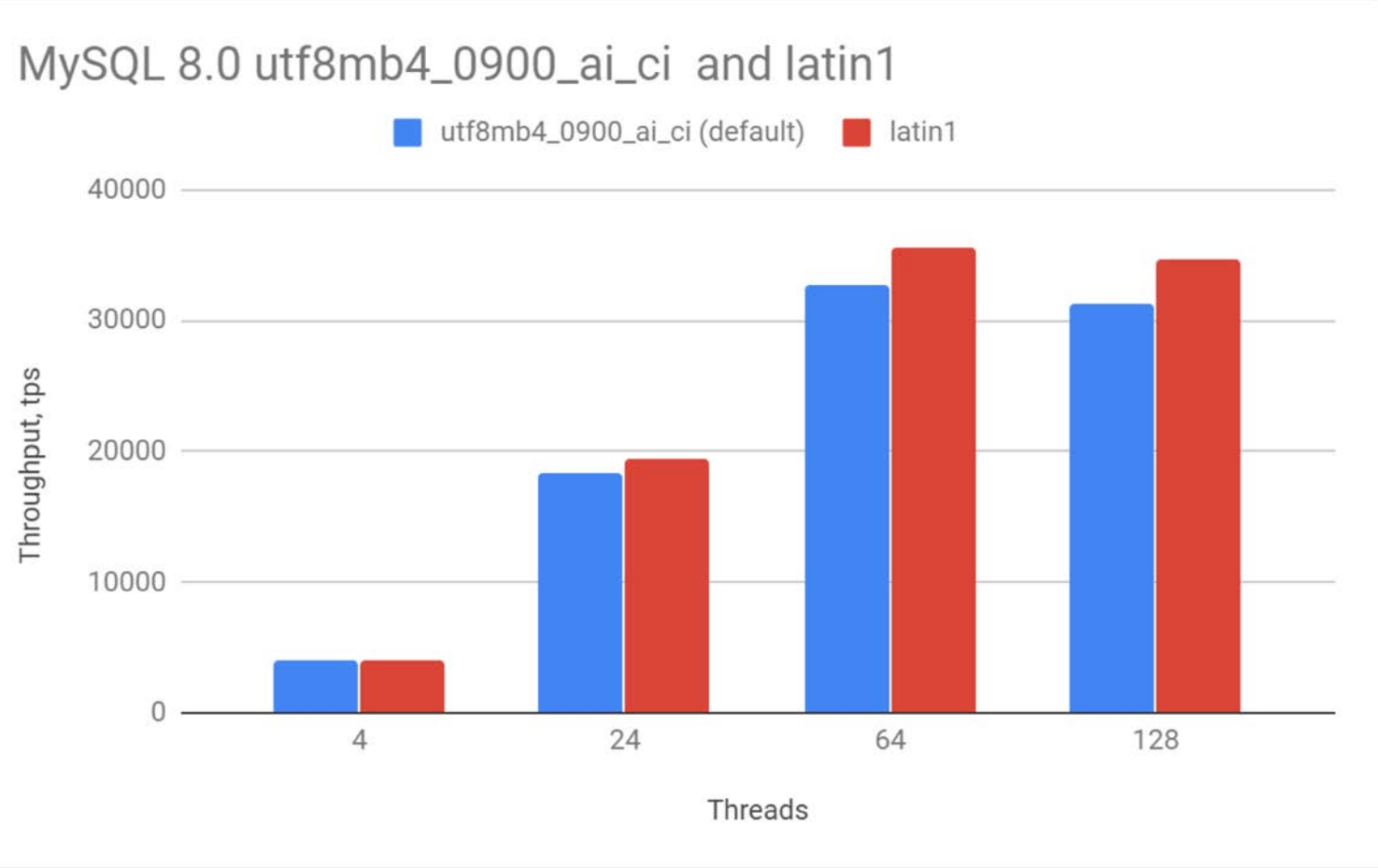
Character Sets



https://per.co.na/MySQLCharsetImpact

PERCONA

Less impact In MySQL 8



PERCONA

Grouping and Sorting

Can use Index, External Sort, Temporary File

Temporary Table can be in memory or on disk

Amount of Data you Sort, Group Matters A Lot

Too many Different Algorithms to Cover in Details

The Mysterious Optimizer

No one knows how MySQL Optimizer Really works

Designed to Choose Best Plan Based on Cost

Cost Model is just a model

Relies on Statistics which can be very wrong

Learn what MySQL Execution Can Do

Not Everything you can imagine can be done by MySQL during execution

Though it also has tricks in its sleeve you may not aware of

Are you Smarter than Optimizer ?

Use Optimizer Hints to Execute Query The way you Like

Often the plan you think is faster is not

https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html

Learn EXPLAIN

A way to understand how MySQL Expects to Execute Query

Plan May Change based on constants, time server

EXPLAIN SELECT …

EXPLAIN FORMAT=JSON SELECT …

https://dev.mysql.com/doc/refman/8.0/en/using-explain.html

Explain Example

```
mysql> explain select max(season_nr) from title group by production_year;
+----+-------------+-------+-------+---------------+-----------------+---------+------+------+----------------------------+
| id | select_type | table | type  | possible_keys | key             | key_len | ref  | rows | Extra                      |
+----+-------------+-------+-------+---------------+-----------------+---------+------+------+----------------------------+
|  1 | SIMPLE      | title | range | NULL          | production_year | 5       | NULL |  201 | Using index for group-by |
+----+-------------+-------+-------+---------------+-----------------+---------+------+------+----------------------------+
1 row in set (0.01 sec)
```


PERCONA

EXPLAIN EXTENDED

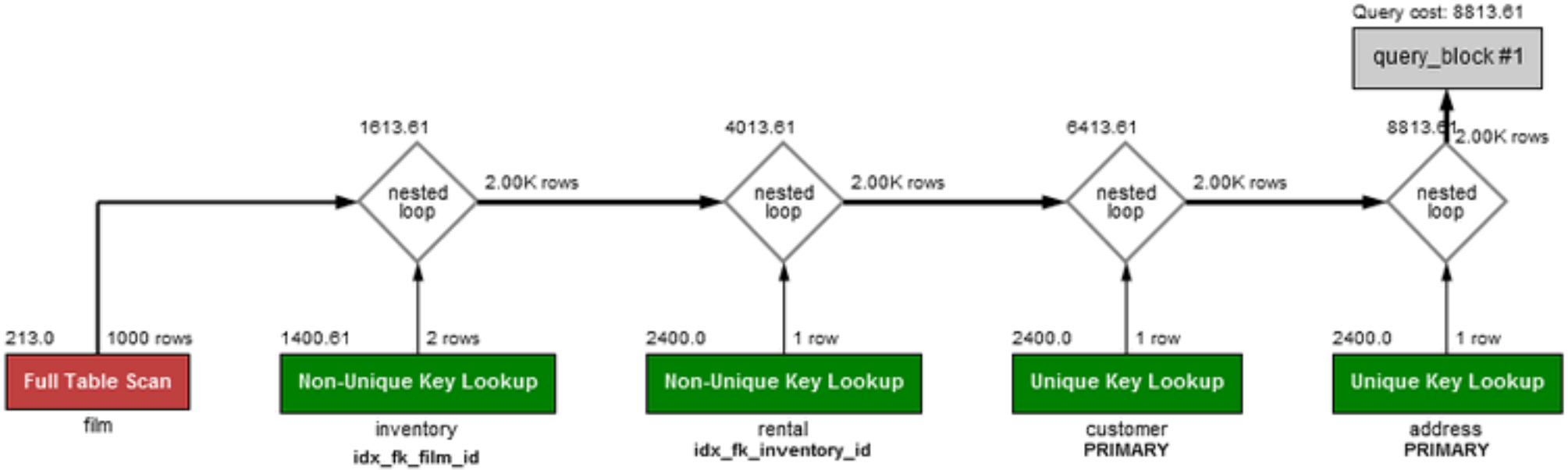
```
mysql> EXPLAIN
       SELECT t1.a, t1.a IN (SELECT t2.a FROM t2) FROM t1\G
```

```
mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1003
Message: /* select#1 */ select `test`.`t1`.`a` AS `a`,
         <in_optimizer>(`test`.`t1`.`a`,`test`.`t1`.`a` in
         ( <materialize> (/* select#2 */ select `test`.`t2`.`a`
         from `test`.`t2` where 1 having 1 ),
         <primary_index_lookup>(`test`.`t1`.`a` in
         <temporary table> on <auto_key>
         where ((`test`.`t1`.`a` = `materialized-subquery`.`a`))))) AS `t1.a
         IN (SELECT t2.a FROM t2)` from `test`.`t1`
1 row in set (0.00 sec)
```

EXPLAIN FORMAT=JSON – More Details

```
mysql> explain format=json select dept_name from departments where dept_no in (select dept_no from dep
*************************** 1. row ***************************
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "16.72"
    },
    "nested_loop": [
      {
        "table": {
          "table_name": "departments",
          <skipped>
      },
      {
        "table": {
          "table_name": "<subquery2>",
          "access_type": "eq_ref",
          "key": "<auto_key>",
          "key_length": "4",
          "ref": [
            "employees.departments.dept_no"
          ],
          "rows_examined_per_scan": 1,
          "materialized_from_subquery": {
            "using_temporary_table": true,
            "query_block": {
              "table": {
```

MySQL WorkBench Visualization



https://www.mysql.com/products/workbench/

PERCONA

Optimizer Trace

- **Advanced Optimizer Debugging if you can't figure out why given plan is chosen**

```
# Turn tracing on (it's off by default):
SET optimizer_trace="enabled=on";
SELECT ...; # your query here
SELECT * FROM INFORMATION_SCHEMA.OPTIMIZER_TRACE;
# possibly more queries...
# When done with tracing, disable it:
SET optimizer_trace="enabled=off";
```

How to Find Queries to Optimize

Where Should you Optimize your Queries ?

Development Environment

Production Environment

Reality

Both

- You want to ensure unoptimized queries never make it to Production
- But you will have Query Performance Issues in Production anyway

Development

Can use MySQL Log (Slow Query Log) or Application Debuging

Can help not only to Optimize Slow Queries but also Eliminate Waste

Development and Production

Using Query Analyzes Tools

Percona's Open Source Solution

See it Live!

https://pmmdemo.percona.com

Top Queries

#	Query Abstract	Load			Count			Latency	
	TOTAL	21.67	100.00%		3.42k QPS	147.82m	100.00%	6.33 ms avg	
1	SELECT sbtest	6.00	27.68%		1.56k QPS	67.45m	45.63%	3.84 ms avg	
2	UPDATE sbtest	5.49	25.36%		272.61 QPS	11.78m	7.97%	20.15 ms avg	
3	COMMIT	1.99	9.18%		152.34 QPS	6.58m	4.45%	13.06 ms avg	
4	SELECT sbtest	1.17	5.41%		156.20 QPS	6.75m	4.57%	7.51 ms avg	
5	SELECT sbtest	1.11	5.13%		155.99 QPS	6.74m	4.56%	7.13 ms avg	
6	SELECT sbtest	1.07	4.95%		184.26 QPS	7.96m	5.38%	5.83 ms avg	
7	SELECT sbtest	1.06	4.90%		156.05 QPS	6.74m	4.56%	6.81 ms avg	
8	SELECT sbtest	1.06	4.88%		155.69 QPS	6.73m	4.55%	6.79 ms avg	
9	UPDATE sbtest	0.84	3.88%		154.63 QPS	6.68m	4.52%	5.43 ms avg	
10	DELETE sbtest	0.83	3.83%		153.37 QPS	6.63m	4.48%	5.41 ms avg	

PERCONA

Things to Consider

Outliers may not be causing the most load

Victims and queries causing the problem

Queries not Finished yet

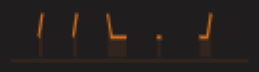
Query Profile

Metrics

Query firs

Metrics	Rate/Sec	Sum
Query Count	1.56k (per sec)	67.53m 45.63% of total
Query Time	6.01 load	3 days, 0:07:38 27.67% of total
Lock Time	0.70 (avg load)	8:21:52 23.81% of total 10.40% of query time
Innodb IO Read Wait	3.27 (avg load)	1 days, 15:17:49 39.57% of total 56.09% of query time
Innodb Read Ops	320.50(per sec)	13.85m 34.27% of total
Innodb Read Bytes	5.25 MB (per sec)	226.85 GB 34.53% of total 16.38 KB avg io size

How Efficiently Query Produces Result ?

Rows Sent	<0.01 (per sec)		57.00 <0.01 of total
Bytes Sent	0.09 (per sec)		3.93 KB <0.01 of total 69.00 Bytes bytes/row
Rows Examined	39.58k (per sec)		1.71b 25.89% of total 30.00m per row sent

Explain and Table Details

VISUAL

```
Filter with WHERE
+- Bookmark lookup
   +- Table
   |  table          sbtest1
   |  possible_keys  PRIMARY
   +- Index range scan
      key            sbtest1->PRIMARY
      possible_keys  PRIMARY
      key_len        4
      rows           49315032
```

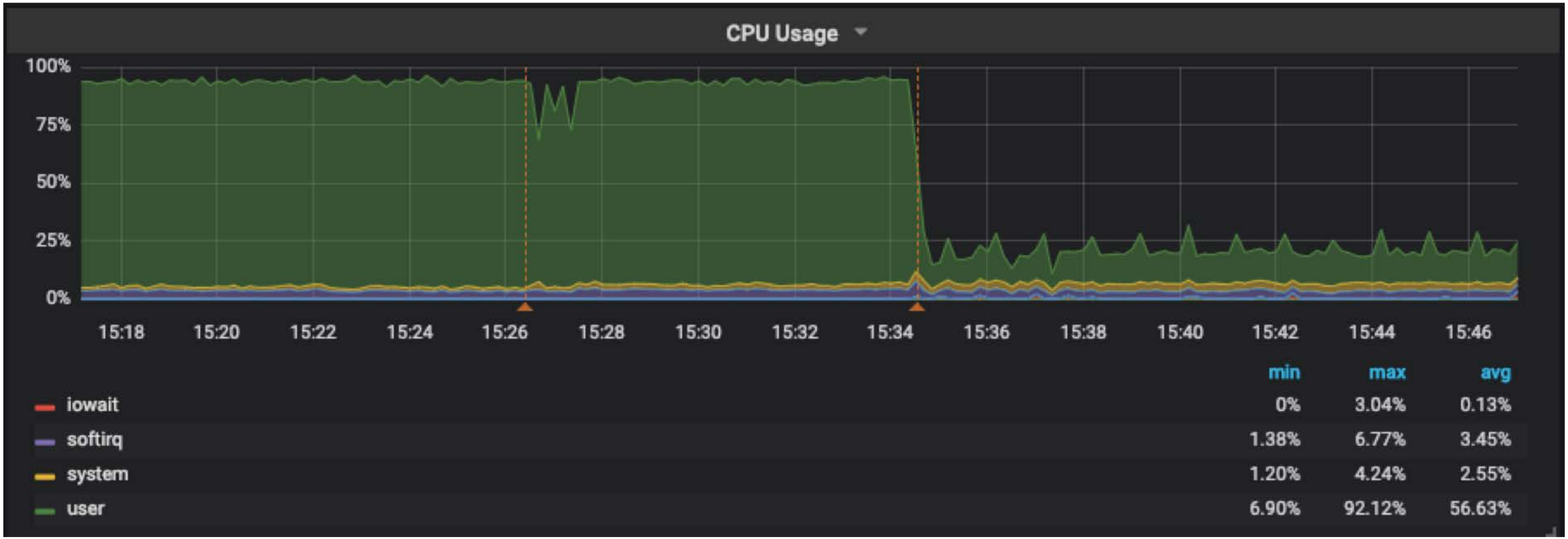
CREATE

```
CREATE TABLE `sbtest1` (
  `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=InnoDB AUTO_INCREMENT=100000001 DEFAULT CHARSET=latin1 MAX_ROWS=1000000
```

INDEXES

KeyName	Type	Unique	Packed	Column	Cardinality
PRIMARY	BTREE	Yes	No	id	98630064
k_1	BTREE	No	No	k	31560404

CPU Usage Reduction with PMM



https://per.co.na/PMMCPU

PERCONA

How to Optimize Them

Query Stats

How Many Rows does it crunch ?

How Many Rows it returns ?

How Much IO is Required ?

Is Temporary Table Required ? Temporary Sort File ?

Run EXPLAIN

Is Plan Reasonable ?

Bad Plan

Missing Indexes

Bad Optimizer Statistics

Bad Query Practices ie “WHERE col+1=10”

Expensive Queries

Queries which Naturally Require A lot of Work to Do

- SELECT AVG(value) FROM ORDERS WHERE ORDER_DATE<“2018-01-01”

Dealing with Expensive Queries

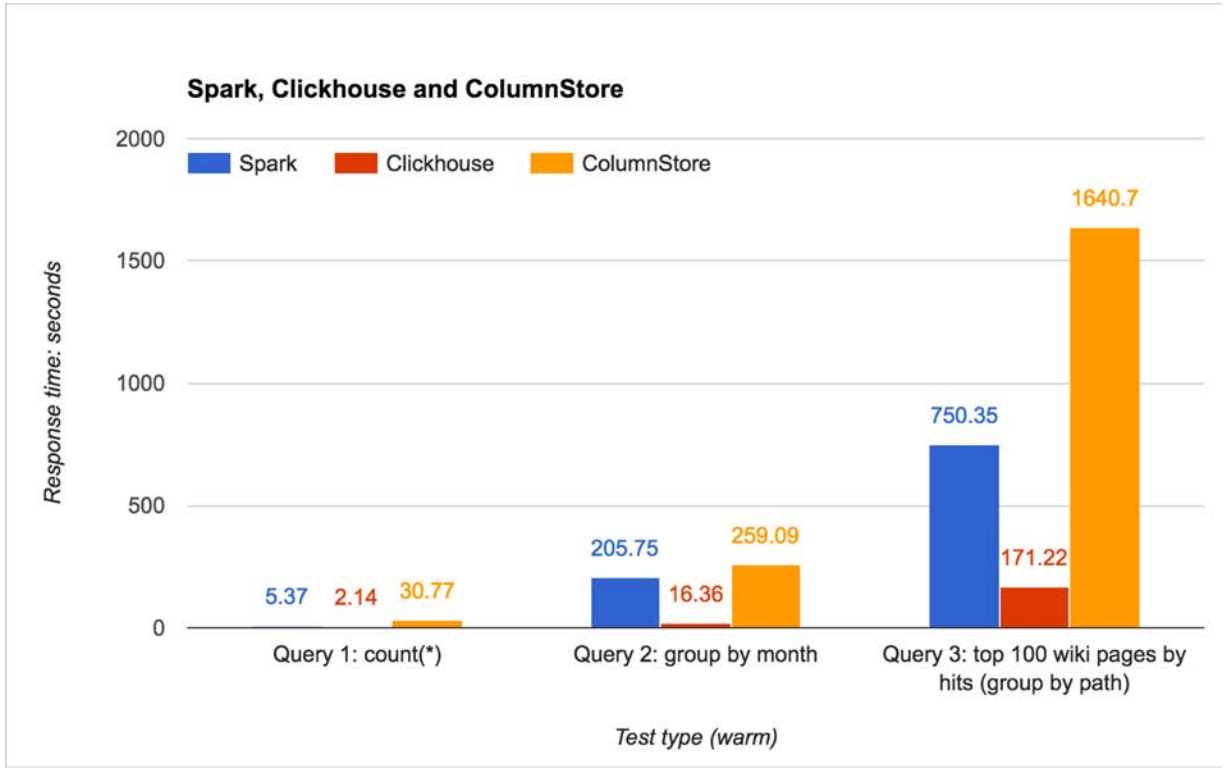
Getting Rid of Them!

Caching Them

Pre-Generating Results

Using Systems which support such queries better

Beyond MySQL for Expensive Queries



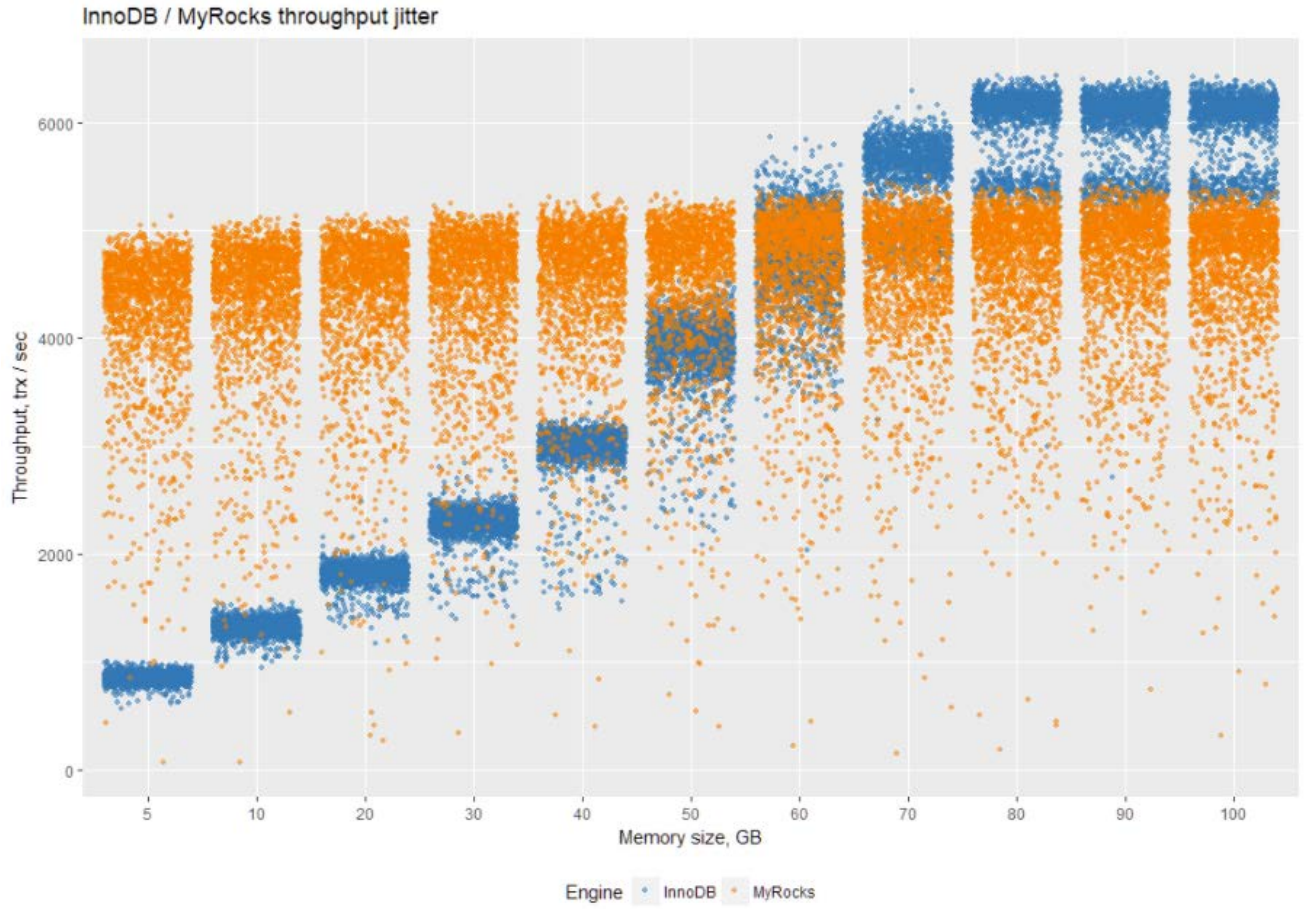
https://per.co.na/jOMbko

PERCONA

Optimizing Writes

- Less Indexes
- Data Fits in Memory
- Batching
- Partitioning
- Different Storage Engine

MyRocks – Better Performance with Large Data



PERCONA

Share your insight and join the debate!

https://per.co.na/survey

PERCONA

Join Us at Percona Live

Percona Live 2019 takes place in Austin, Texas from May 28-30, 2019 at the Hyatt Regency.

Percona Live provides an opportunity to network with peers and technology professionals. Mingle with all types of database community members: DBAs, developers, C-level executives and the latest database technology trend-setters.

SUPER SAVER TICKETS ON SALE!

https://www.percona.com/live/19/

OPEN SOURCE DATABASE CONFERENCE

PERCONA LIVE

Connect. Accelerate. Innovate.

Thank You!