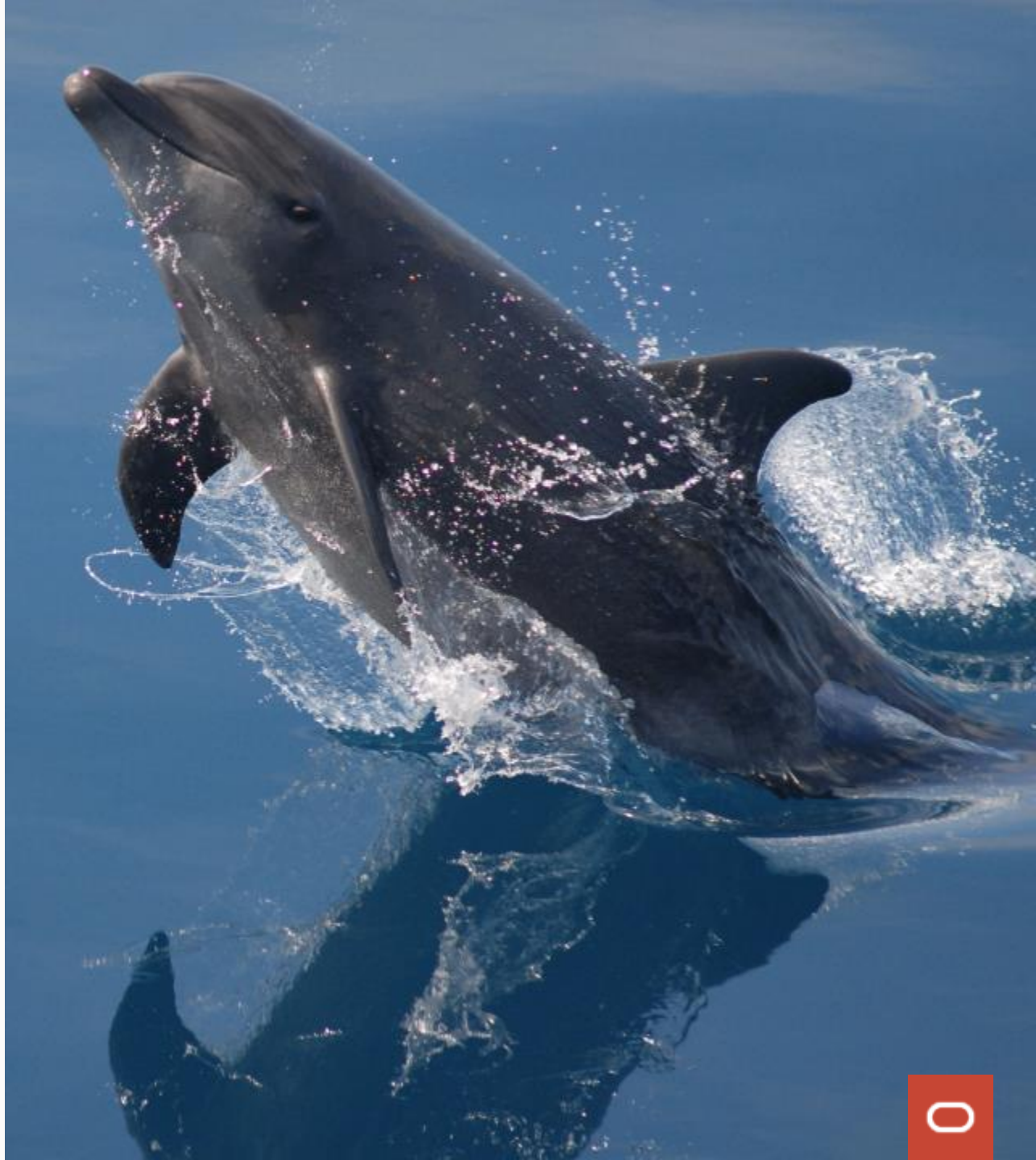
ORACLE

Monitor MySQL with Zabbix: Understanding the metrics

Vittorio Cioe
MySQL Principal Solution Engineer
vittorio.cioe@oracle.com



> whoami

- Linux and MySQL user since ≈ 2006
- Working at Oracle/MySQL since 2017
  (lot of travel => lot of fun!)
- Regularly speaking at conferences
- Previously working in the Security and Digital Transformation (API) space

- From Italy but based in Warsaw
- Love movies, travelling, cooking...

Agenda

- MySQL and how it works

Let’s have a look at key Zabbix MySQL metrics in the area of...

- Connection Handling
- Memory Buffers
- Log files
- Query execution

MySQL and how it works

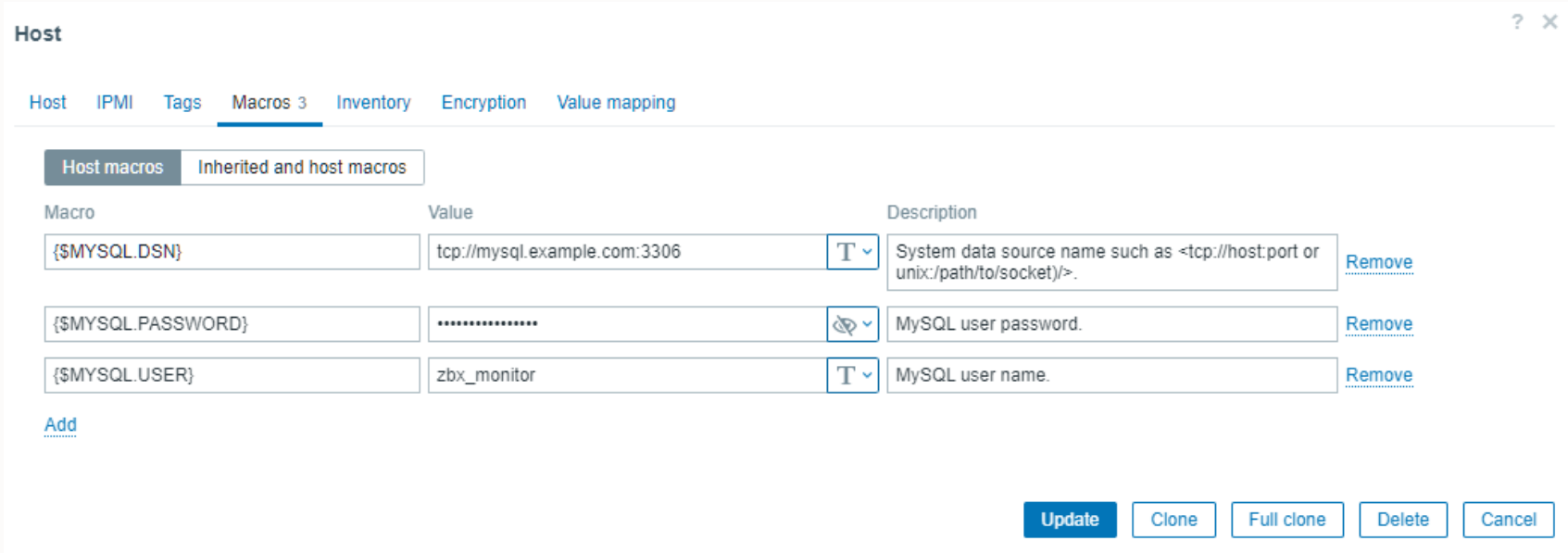
MySQL is the #1 Open Source Database

Rank			DBMS	Database Model	Score		
Sep 2022	Aug 2022	Sep 2021			Sep 2022	Aug 2022	Sep 2021
1.	1.	1.	Oracle	Relational, Multi-model	1238.25	-22.54	-33.29
2.	2.	2.	MySQL	Relational, Multi-model	1212.47	+9.61	-0.06
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	926.30	-18.66	-44.55
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.46	+2.46	+42.95
5.	5.	5.	MongoDB	Do[illegible]odel	489.64	+11.97	-6.87

Source: DB Engines Ranking

Starting monitoring MySQL with Zabbix is really easy

1. Install Zabbix agent2
2. Create a MySQL user with monitoring privileges (for example GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';)
3. Create host with the Zabbix agent interface
4. Assign the MySQL template to the host
5. Adjust user macro values to match DSN, username and password



Zabbix can get you an overview... and much more!

MySQL server: MySQL: Status

2022-10-03 14:25:19

Up (1)

MySQL: Status

Uptime

2022-10-03 14:46:17

01:11:54

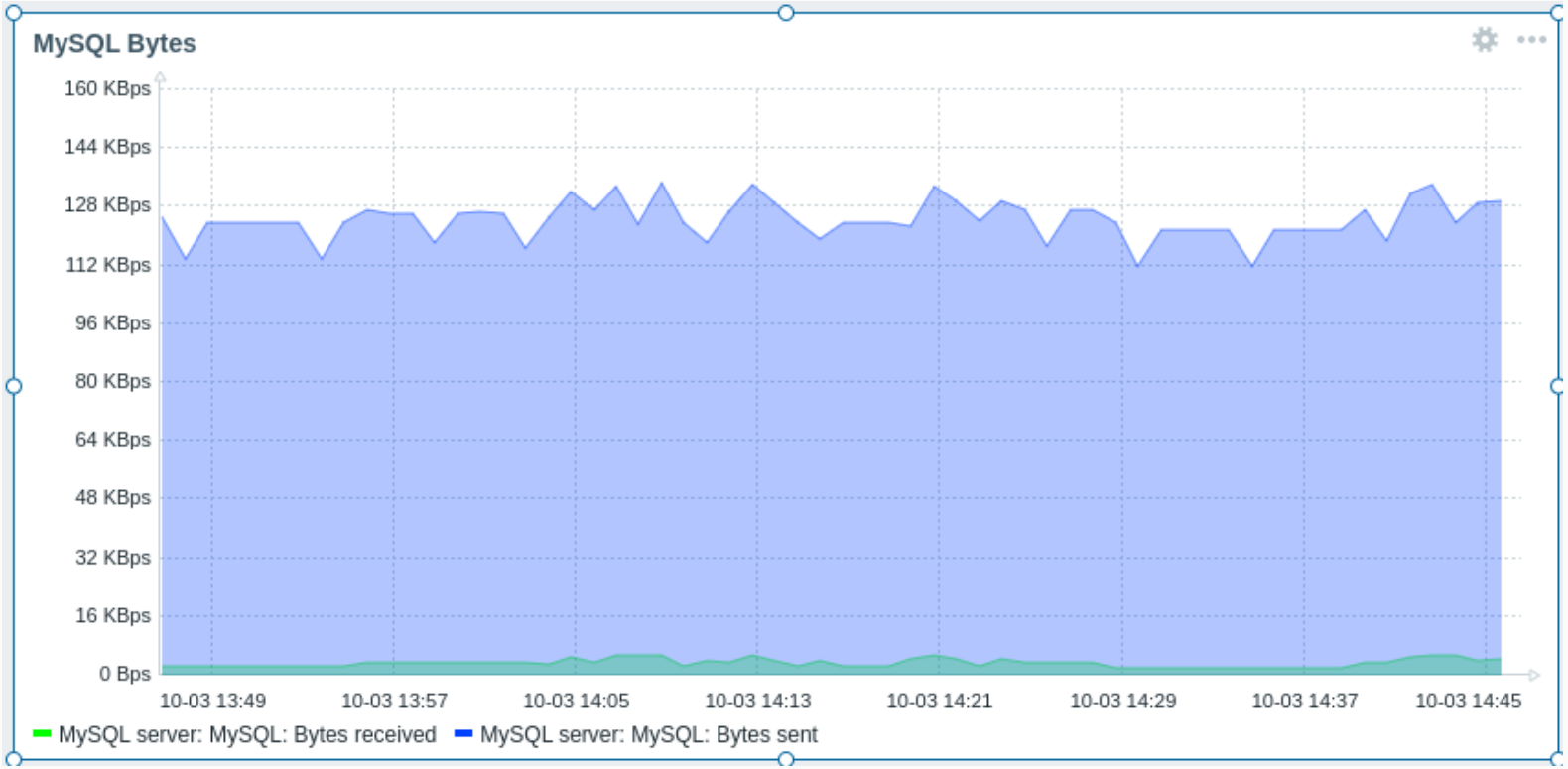
MySQL: Uptime

MySQL version

2022-10-04 07:29:20

8.0.26

MySQL: Version

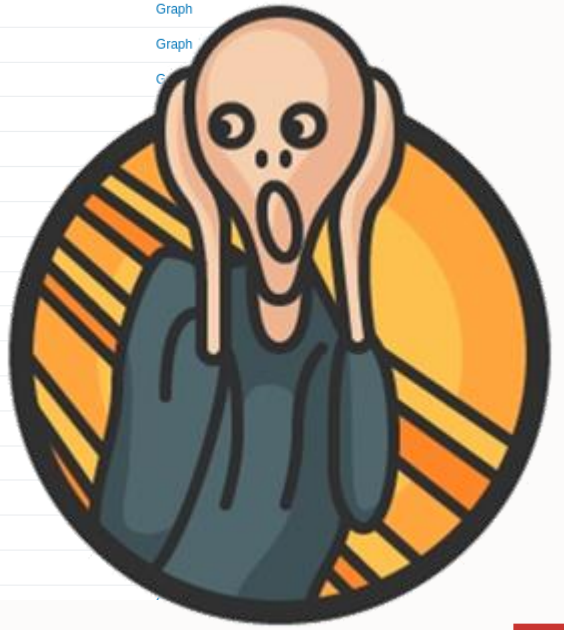


	MySQL server	MySQL: Service is down	12s
PROBLEM	MySQL server	MySQL: Buffer pool utilization is too low (less than 50% for 5m)	3m 15s
PROBLEM	MySQL server	MySQL: Service has been restarted (uptime < 10m)	8m 14s

… but MySQL provides A LOT of metrics!!

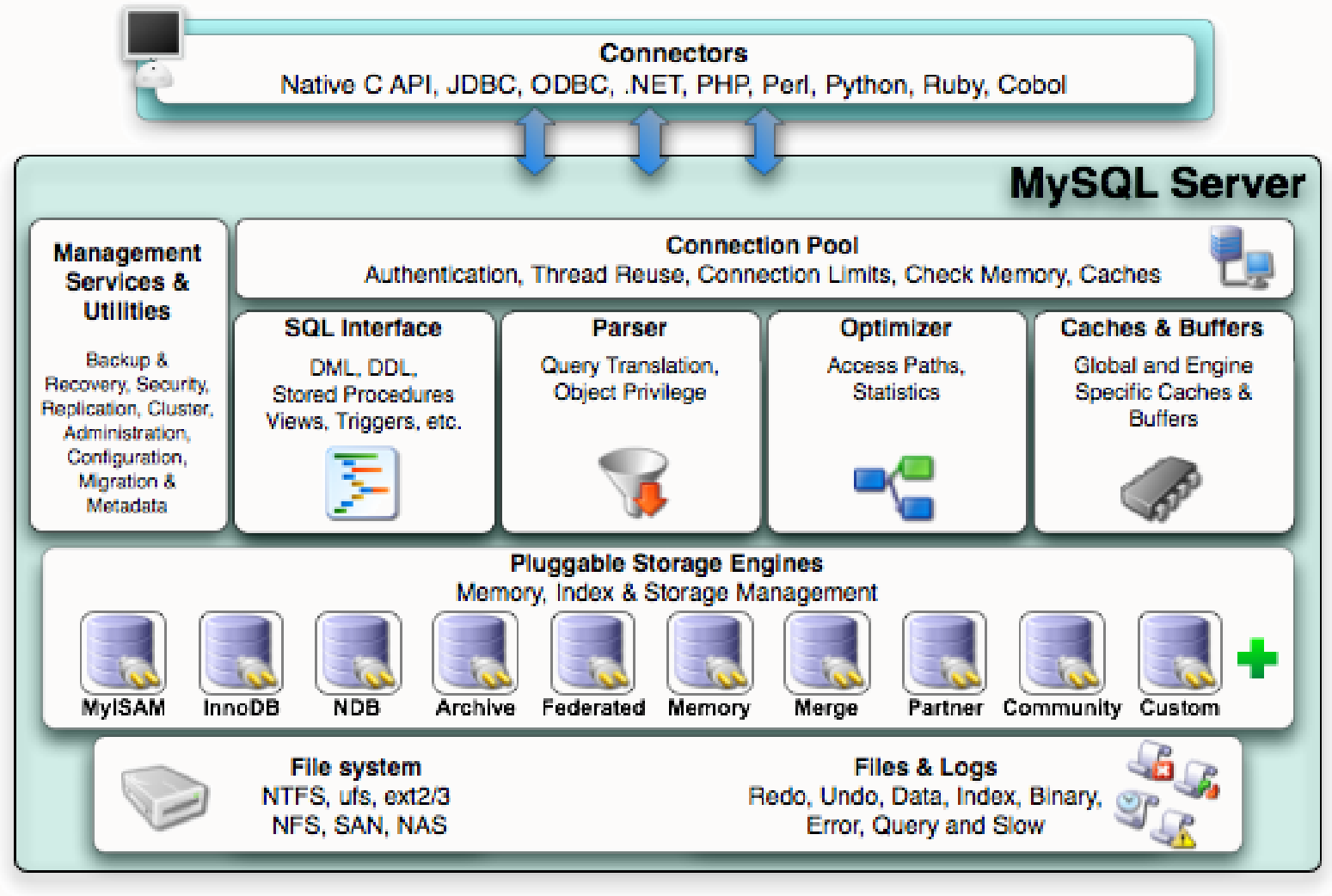
What is important to know? (SHOW STATUS provides 481 variables!!)

Host	Name ▲	Last check	Last value	Change	Tags	Info
MySQL server	MySQL: Aborted clients per second	29s	0		component: connecti…	Graph
MySQL server	MySQL: Aborted connections per second	29s	0		component: connecti…	Graph
MySQL server	MySQL: Binlog cache disk use	54m 29s	1		component: cache	Graph
MySQL server	MySQL: Buffer pool efficiency	31s	0.00005852 %	+0.00000002727 %	component: memory	Graph
MySQL server	MySQL: Buffer pool utilization	30s	87.5 %		component: memory	Graph
MySQL server	MySQL: Bytes received	29s	4.2 KBps	+105.2994 Bps	component: network	Graph
MySQL server	MySQL: Bytes sent	29s	128.14 KBps	+693.0948 Bps	component: network	Graph
MySQL server	MySQL: Command Delete per second	29s	0.1164	-0.09981	component: operations	Graph
MySQL server	MySQL: Command Insert per second	29s	1.0314	-0.1497	component: operations	Graph
MySQL server	MySQL: Command Select per second	29s	20.7935	+0.7488	component: operations	Graph
MySQL server	MySQL: Command Update per second	29s	0.3493	+0.000004244	component: operations	Graph
MySQL server	MySQL: Connection errors accept per second	29s	0		component: connecti…	Graph
MySQL server	MySQL: Connection errors internal per second	29s	0		component: connecti…	G
MySQL server	MySQL: Connection errors max connections per second	29s	0		component: connecti…	
MySQL server	MySQL: Connection errors peer address per second	29s	0		component: connecti…	
MySQL server	MySQL: Connection errors select per second	29s	0		component: connecti…	
MySQL server	MySQL: Connection errors tcpwrap per second	29s	0		component: connecti…	
MySQL server	MySQL: Connections per second	29s	0.5157	+0.01664	component: connecti…	
MySQL server	MySQL: Created tmp files on disk per second	29s	0		component: storage	
MySQL server	MySQL: Created tmp tables on disk per second	29s	0		component: storage component: tables	
MySQL server	MySQL: Created tmp tables on memory per second	29s	1.0314	-0.04989	component: memory component: tables	
MySQL server	MySQL: Get status variables				component: raw	
MySQL server	MySQL: InnoDB buffer pool pages free	29s	1024		component: innodb component: memory	
MySQL server	MySQL: InnoDB buffer pool pages total	54m 29s	8192		component: innodb component: memory	
MySQL server	MySQL: InnoDB buffer pool read requests	29s	5802462151	+714575	component: innodb component: memory	
MySQL server	MySQL: InnoDB buffer pool read requests per second	29s	11886.8202	+10.724	component: innodb component: memory	
MySQL server	MySQL: InnoDB buffer pool reads	29s	3395		component: innodb component: memory	

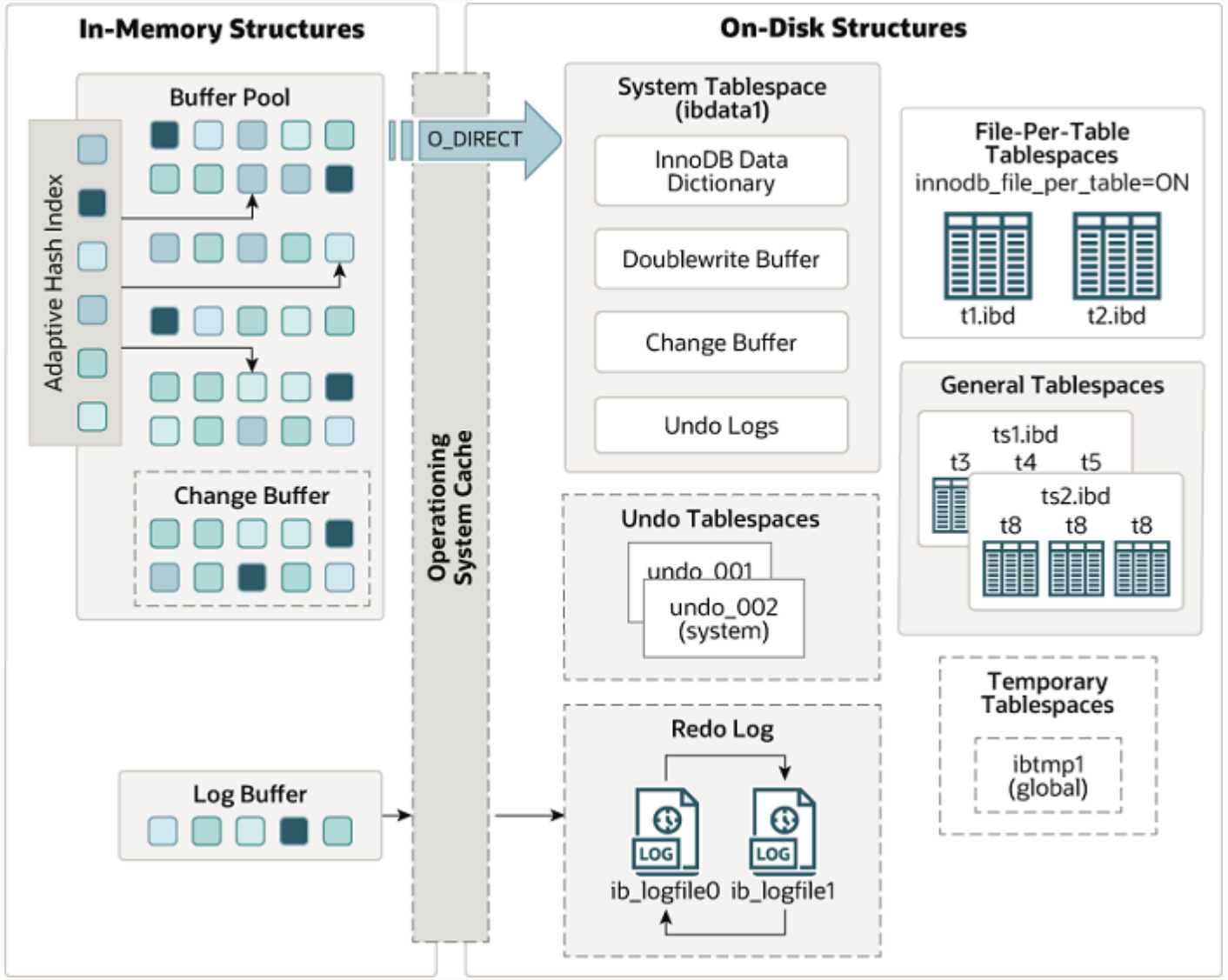


Complete list of metrics gathered by Zabbix: https://www.zabbix.com/integrations/mysql

How does MySQL work? Overall Architecture



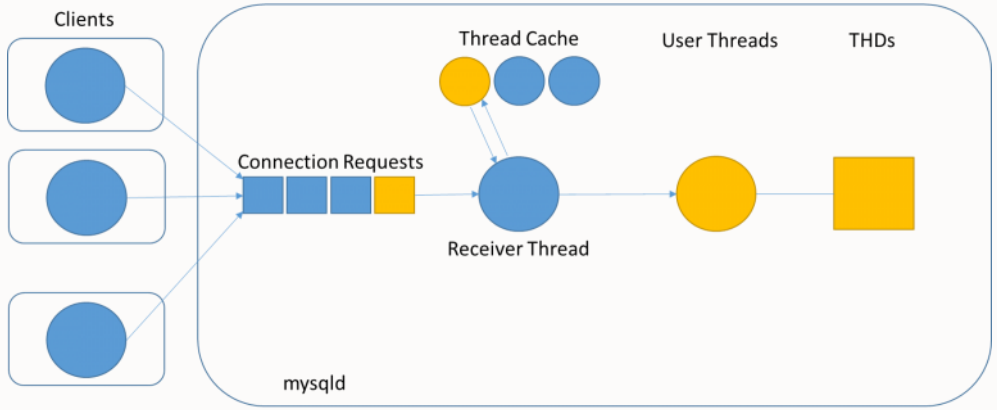
How does MySQL works? InnoDB Architecture

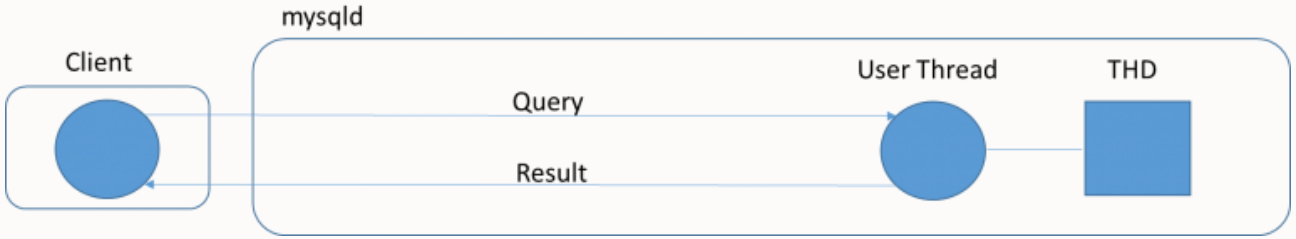


Connection Handling

How MySQL handles client connections

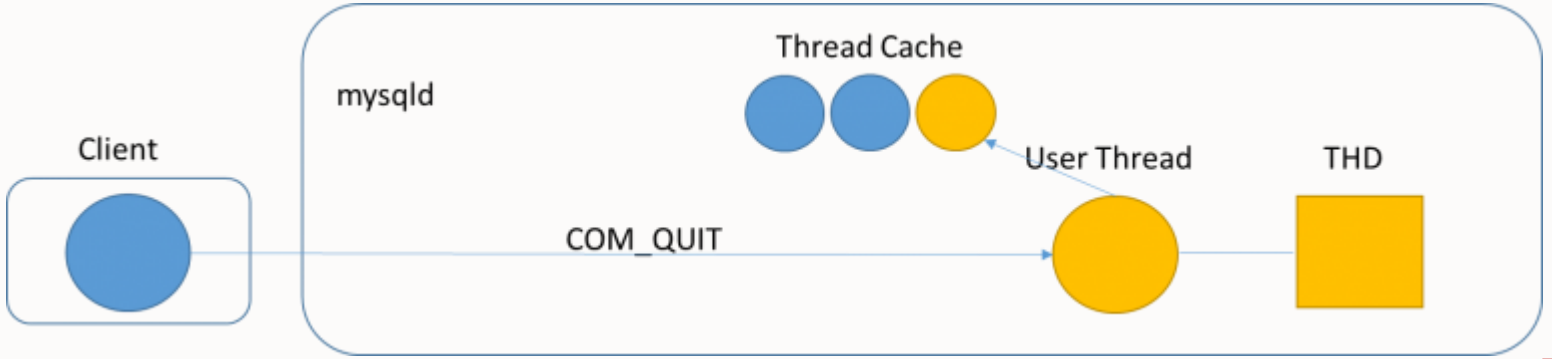
1 - Client connection establishing





2 - Client connection working

3 - Client connection quitting



Connections and Opened Files

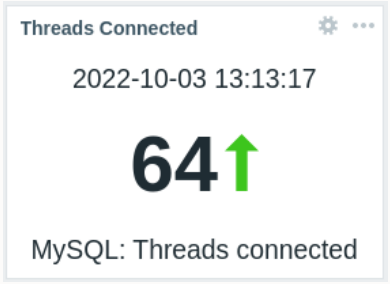
- **max_connections**
  - Maximum permitted number of simultaneous client connections
  - Be careful setting this too large as each connection requires memory

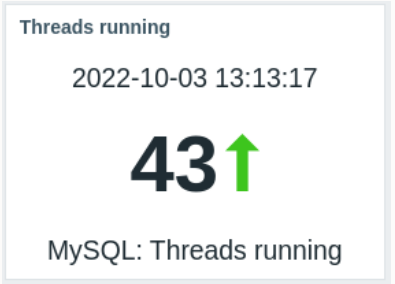
- **max_connections** affect the maximum number of files the server keeps open
  - If you increase it, you may contribute to run up against a limit imposed by your operating system on the per-process number of opened file descriptors.

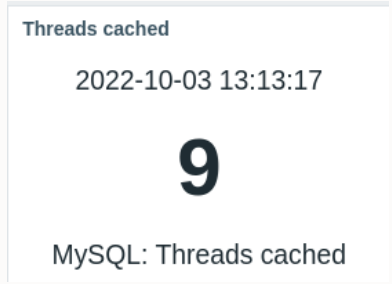
Remember to set ulimits and file descriptors in Linux servers

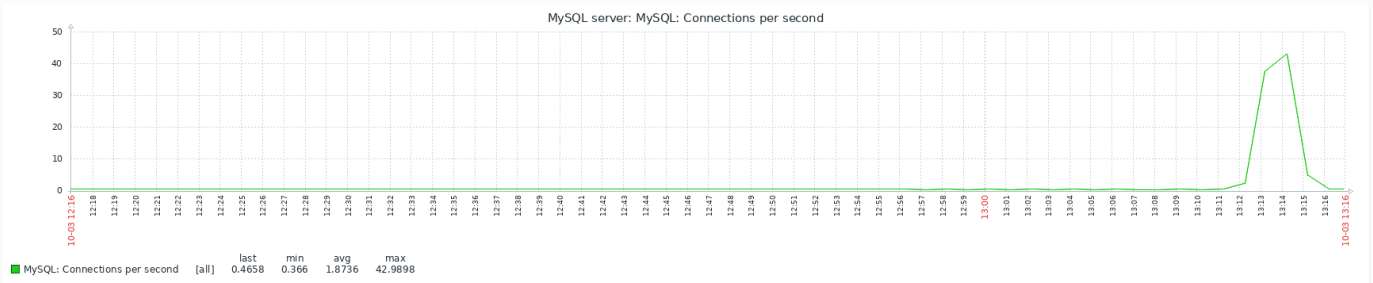
What can we monitor with Zabbix

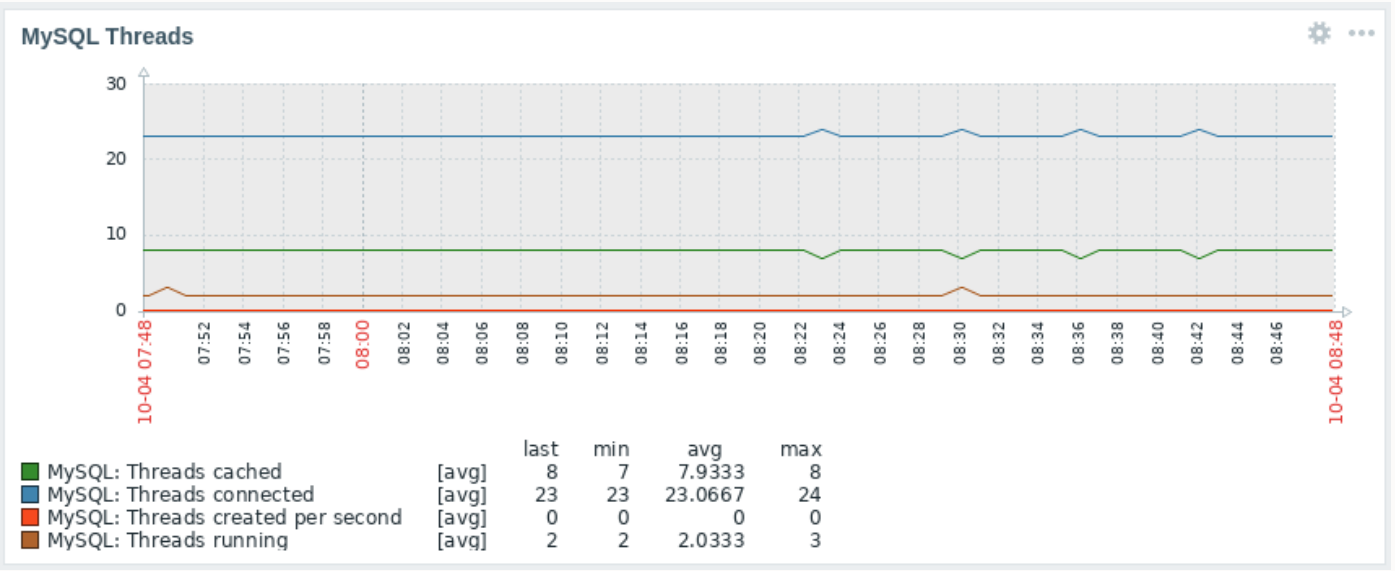
- Threads connected: based on **threads_connected**. Shows all threads including sleeping
- Threads running (not sleeping): based on **threads_running**.
- Threads cached: based on **threads_cached**. Possible values up to size of **thread_cache_size**
- Connections per second
- Overall graphic of MySQL threads **by category**







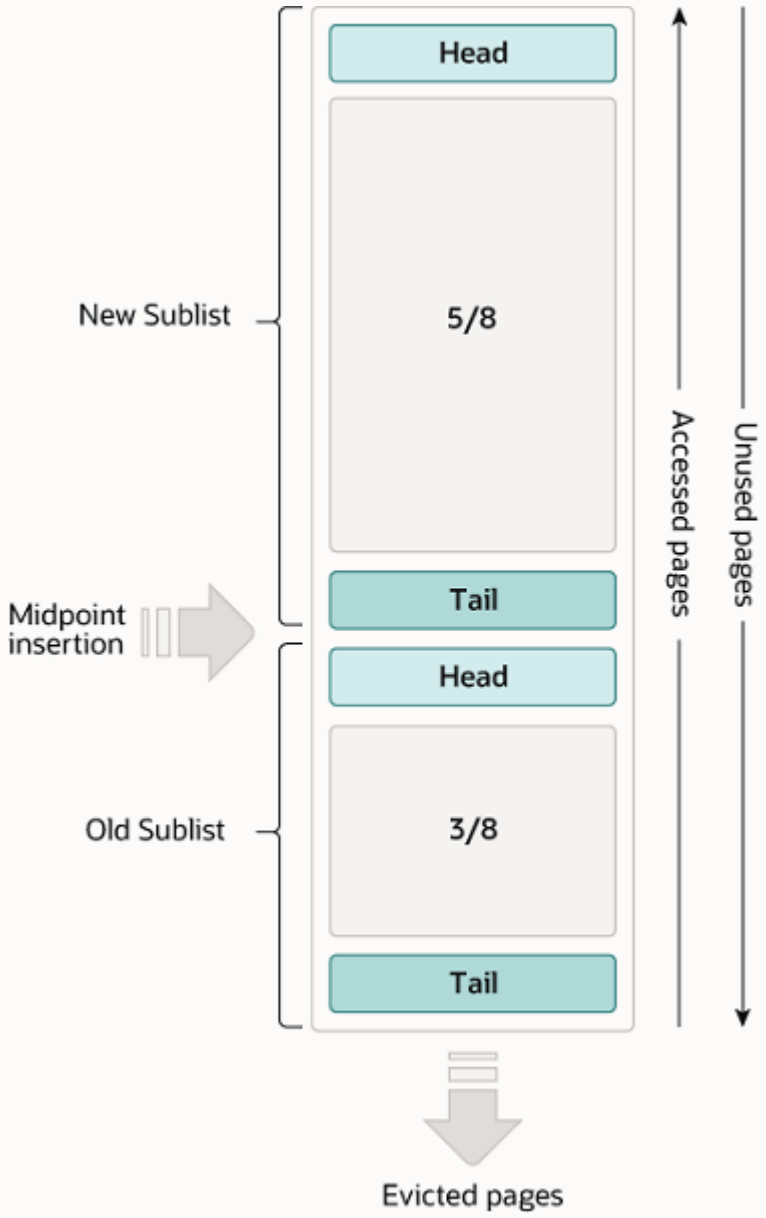




Memory Buffers

InnoDB Buffer Pool

- The buffer pool is an area in main memory where InnoDB caches table and index data as it is accessed.
- The buffer pool permits frequently used data to be accessed directly from memory, which speeds up processing.
- For efficiency of high-volume read operations, the buffer pool is divided into pages that can potentially hold multiple rows.
- For efficiency of cache management, the buffer pool is implemented as a linked list of pages; data that is rarely used is aged out of the cache using a variation of the least recently used (LRU) algorithm.



InnoDB Buffer Pool Configuration

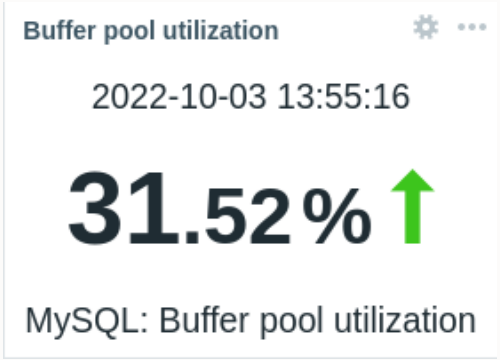
- Size of the **buffer pool**, memory area where InnoDB caches table and index data
- A larger buffer pool requires less disk I/O to access the same table data more than once
  - Since MySQL 5.7, **innodb_buffer_pool_size** can be changed dynamically
  - On dedicated servers, from 50 to 80% of physical free memory is often assigned to the buffer pool.
- **innodb_buffer_pool_instances** - rule of thumb: approximately 2G per instance (< 2G = 1 instance)
- Save the status of the buffer pool at shutdown: **innodb_buffer_pool_dump_at_shutdown**, **innodb_buffer_pool_dump_pct**
- Restore the status of the buffer pool at startup: **innodb_buffer_pool_load_now**, **SHOW STATUS LIKE 'Innodb_buffer_pool_load_status'**

InnoDB Buffer Pool – Checking the size of your working set

Verify how much the InnoDB Buffer Pool is filled with data

```
SELECT CONCAT(FORMAT(A.num * 100.0 / B.num,2),"%") BufferPoolFullPct
FROM
(
   SELECT variable_value num
   FROM performance_schema.global_status
   WHERE variable_name = 'Innodb_buffer_pool_pages_data'
) A
INNER JOIN
(
   SELECT variable_value num
   FROM performance_schema.global_status
   WHERE variable_name = 'Innodb_buffer_pool_pages_total'
) B;
```

```
+-------------------+
| BufferPoolFullPct |
+-------------------+
| 31.01%            |
+-------------------+
1 row in set (0.00 sec)
```



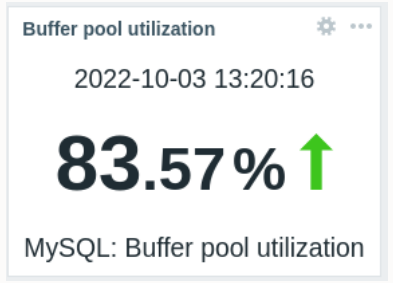
Opened tables and and Opened Files

Again... check ulimits!

- **table_open_cache**
  - Number of maximum allowed open tables for all threads
  - Each table can be open more than once
  - You can check whether you need to increase the table cache by checking the **Opened_tables** status variable
    - If the value of **Opened_tables** is large and you do not use **FLUSH TABLES** often, then you should increase the value of the **table_open_cache** variable

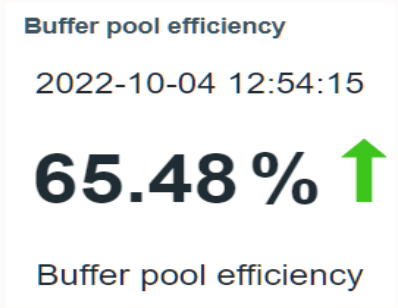
- Also table_open_cache affects the maximum number of files the server keeps open

What can we monitor with Zabbix (1/2)



- **InnoDB Buffer Pool Utilization:**

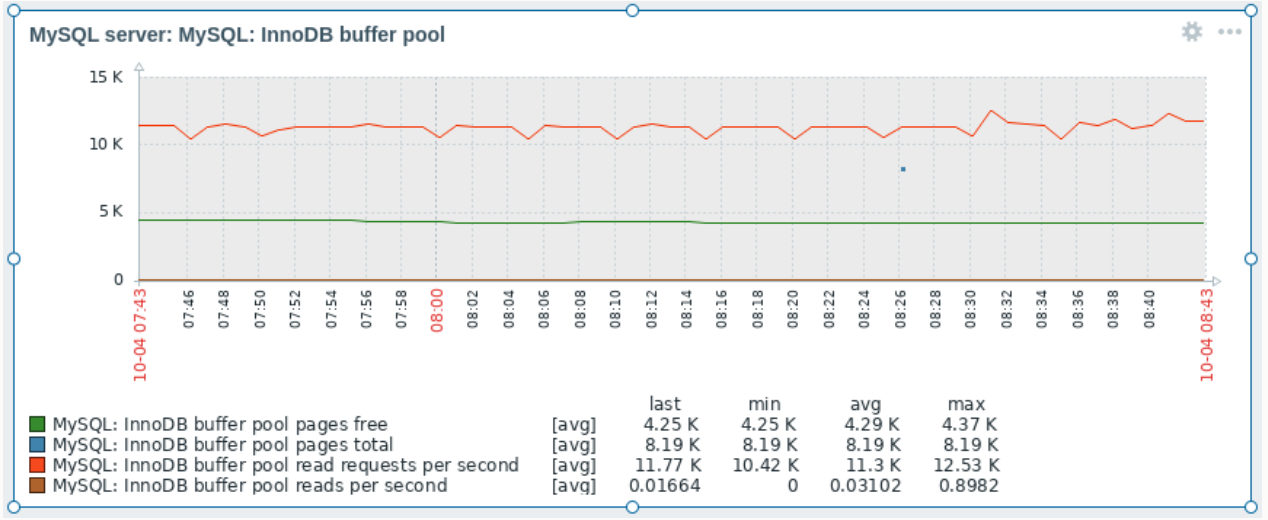
```
( last(//mysql.innodb_buffer_pool_pages_total) -
last(//mysql.innodb_buffer_pool_pages_free) ) / (
last(//mysql.innodb_buffer_pool_pages_total) + (
last(//mysql.innodb_buffer_pool_pages_total) = 0 ) ) * 100 * (
last(//mysql.innodb_buffer_pool_pages_total) > 0 )
```



- **InnoDB Buffer pool efficiency:**

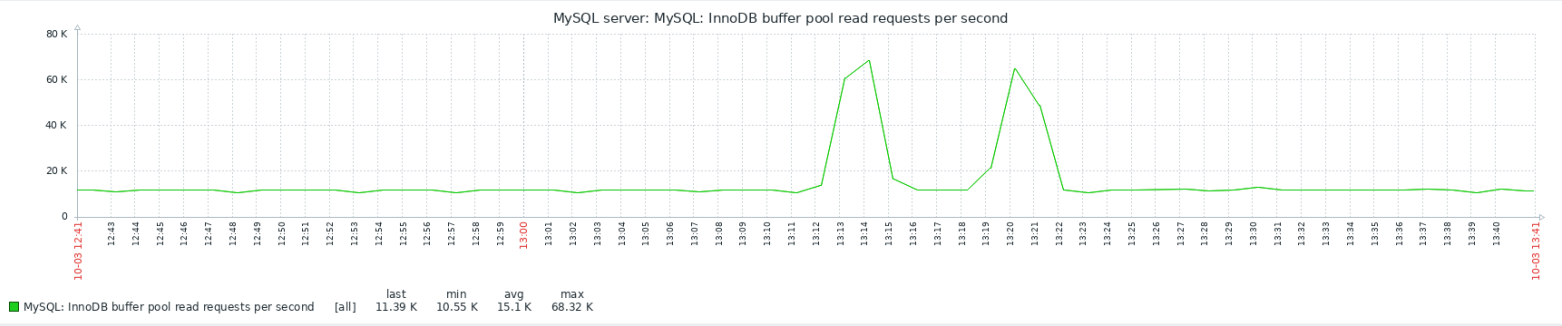
```
last(//mysql.innodb_buffer_pool_reads) / (
last(//mysql.innodb_buffer_pool_read_requests) + (
last(//mysql.innodb_buffer_pool_read_requests) = 0 ) ) * 100 * (
last(//mysql.innodb_buffer_pool_read_requests) > 0 )
```

- **InnoDB Buffer pool usage breakdown**

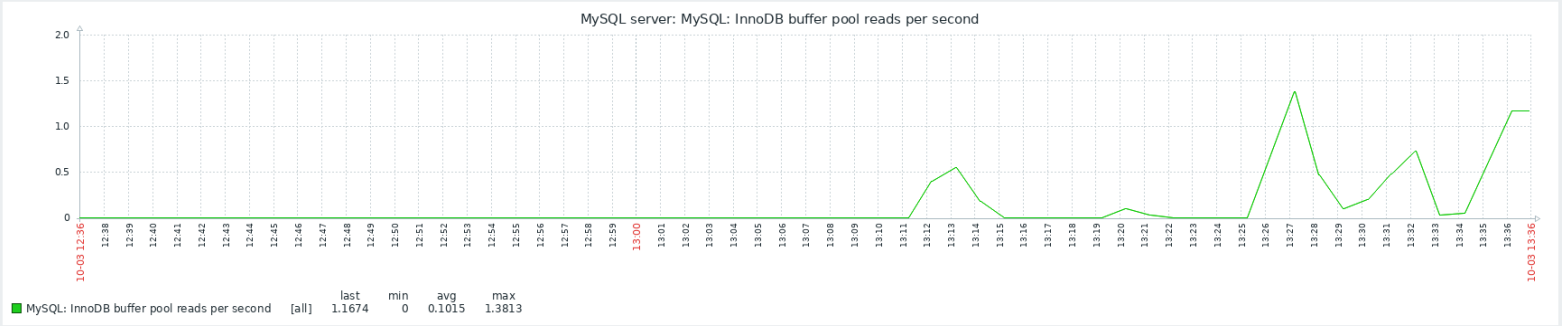


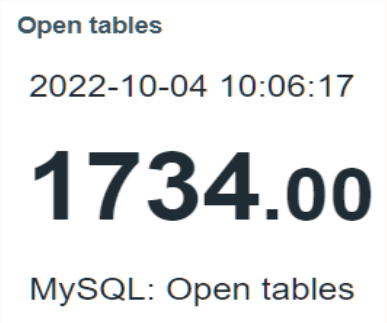
What can we monitor with Zabbix (2/2)

- InnoDB buffer pool read requests per second: **innodb_buffer_pool_read_requests**



- InnoDB buffer pool reads per second (from disk): **innodb_buffer_pool_reads**



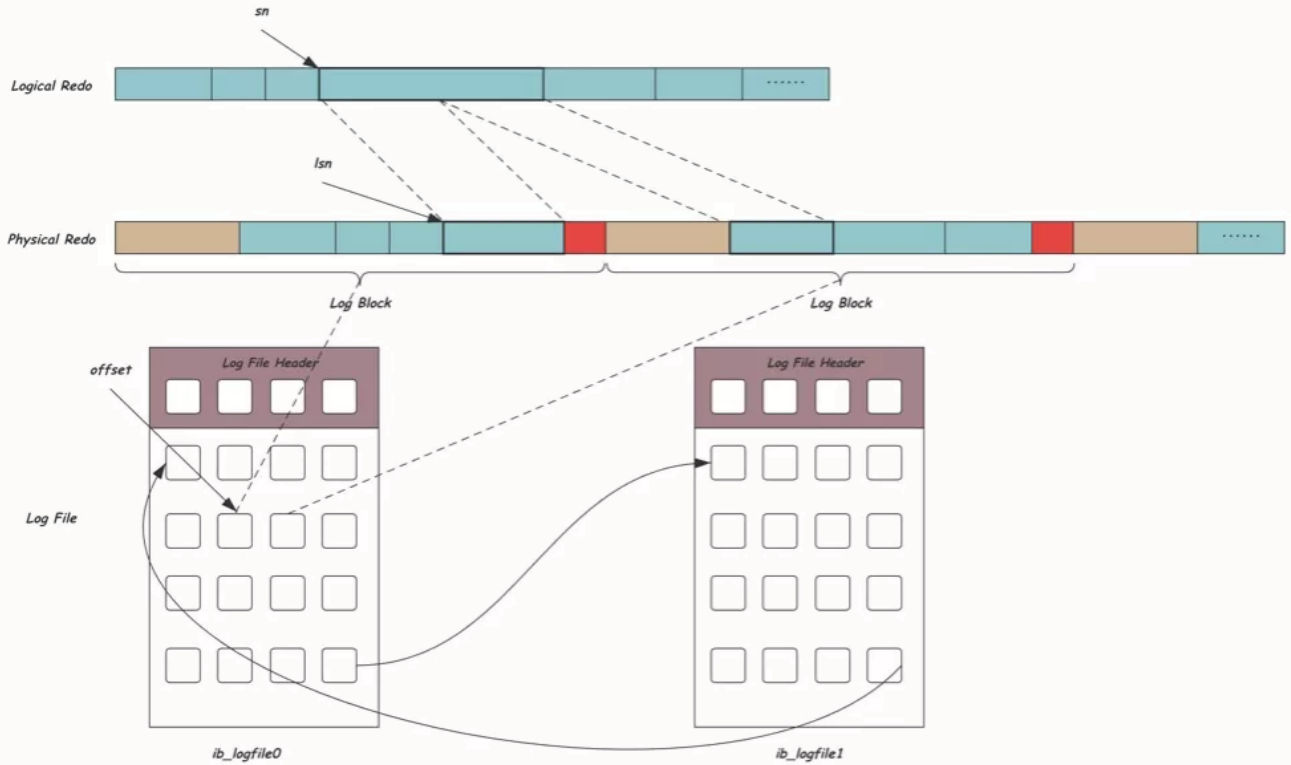


Open Tables:
depends on **opened_tables** parameter


Log files

InnoDB Redo Log

- Disk-based data structure used during crash recovery to correct data written by incomplete transactions, represented on disk by two files named ***ib_logfile0*** and ***ib_logfile1***
- Total redo log size defined by two options: **innodb_log_file_size** & **innodb_log_files_in_group**
  (Total size = innodb_log_file_size * innodb_log_files_in_group)
- Should be large enough to avoid “excessive” flushing
- With large transactions increase the size of the log buffer - **innodb_log_buffer_size**
- **Consider using dedicated volume**



An important parameter: innodb_flush_log_at_trx_commit

- Controls the balance between **strict ACID compliance** for commit operations & **higher performance**

- You can achieve **better performance** by changing the default value but then **you can lose transactions** in a crash. Possible values in order of data safety:
  - **1**: Logs are written and flushed to disk at each transaction commit
    - Is theoretically the slowest, but with fast SSD it may be around as fast as 2 and 0
  - **2**: Logs are written after each transaction commit and flushed to disk once per second
    - Transactions for which logs have not been flushed can be lost in a crash (host)
  - **0**: Logs are written and flushed to disk once per second
    - Transactions for which logs have not been flushed can be lost in a crash (host & mysqld)

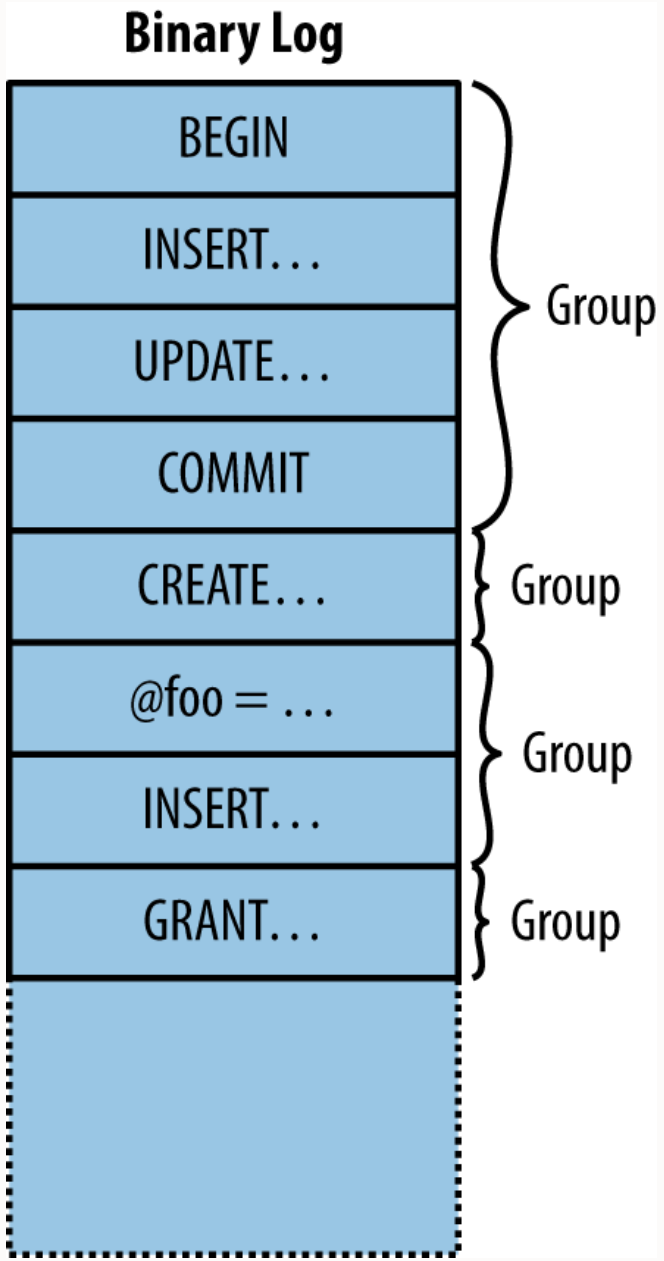
- Defaults to 1; Logs are written and flushed to disk at each transaction commit
  - Required for full ACID compliance - D in ACID
  - For this reason it is the recommended value

- Log flushing frequency is controlled by **innodb_flush_log_at_timeout**

Binary Log

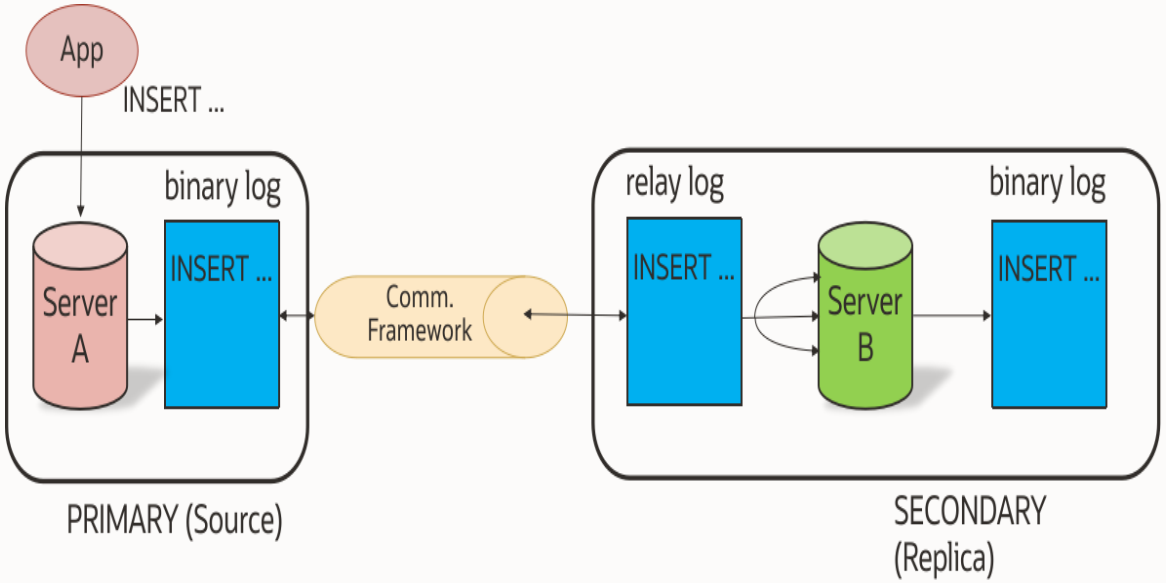
- Contains events that describe changes
- Provides data changes to be sent to Replicas
- Used for data recovery operations
- Decreases performance slightly
- Can be read with **mysqlbinlog**
- Different modes with **binlog_format**:
  STATEMENT, ROW (default in MySQL 8.0), MIXED

- By default, synched to disk before transactions are committed (control using **sync_binlog** variable)
- **Consider using dedicated volume**

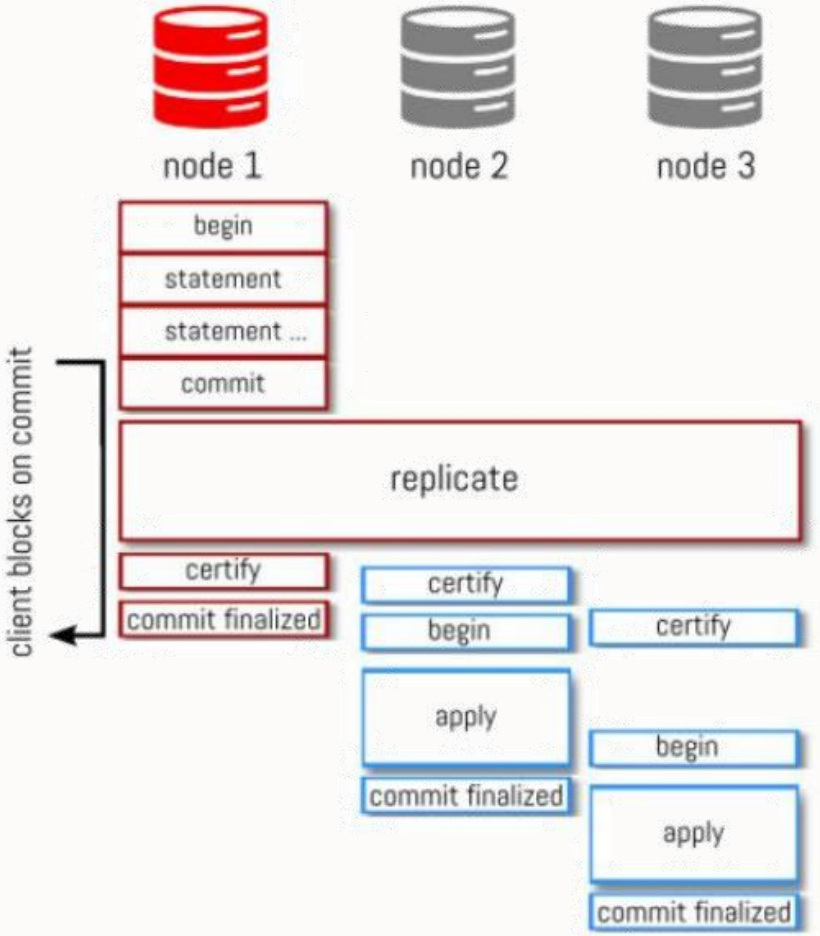


For what binlogs can be used: Replication and High Availability

Asynchronous Replication



Group Replication (InnoDB Cluster)

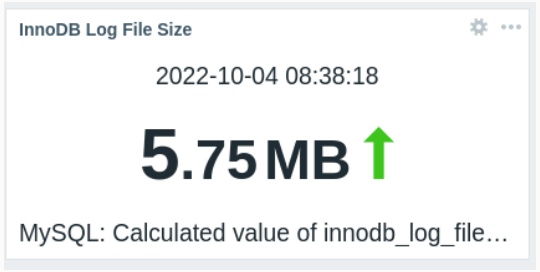


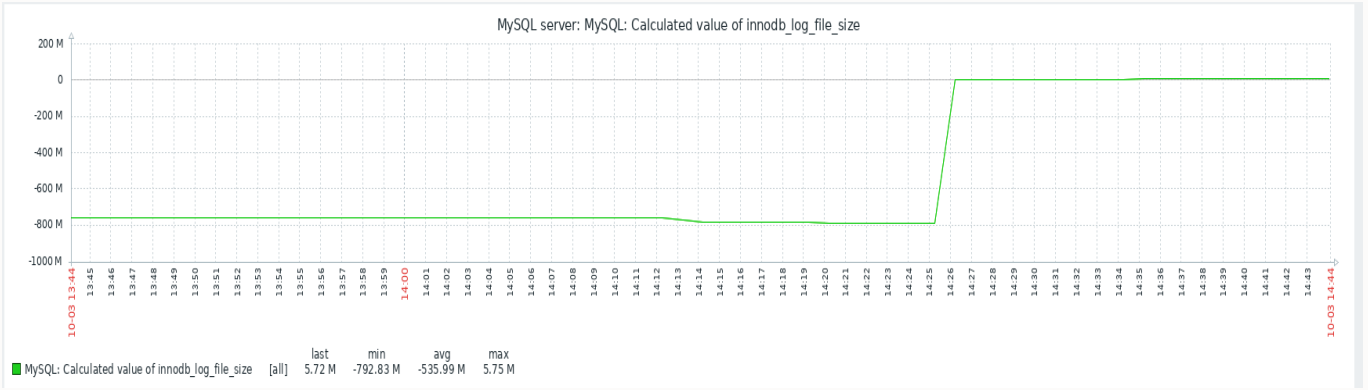
...and don't forget Point In Time Recovery (PITR)!!

Additional info: https://dev.mysql.com/doc/refman/8.0/en/point-in-time-recovery-binlog.html

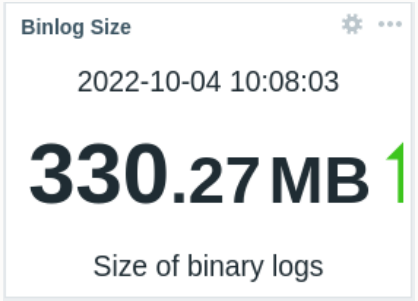
What can we monitor with Zabbix (1/2)

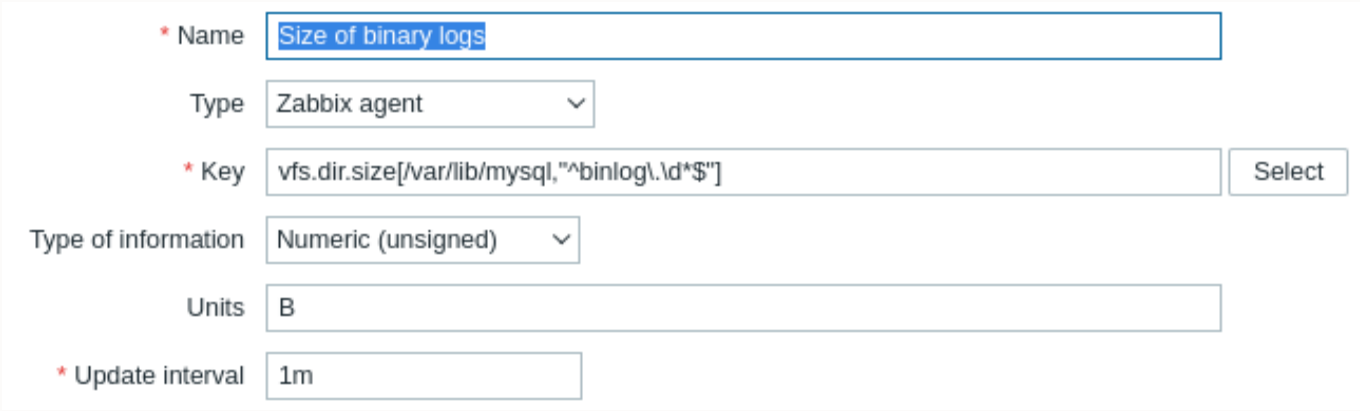
- **Calculated value of innodb_log_file_size**





- **Binary Log size** (custom item)





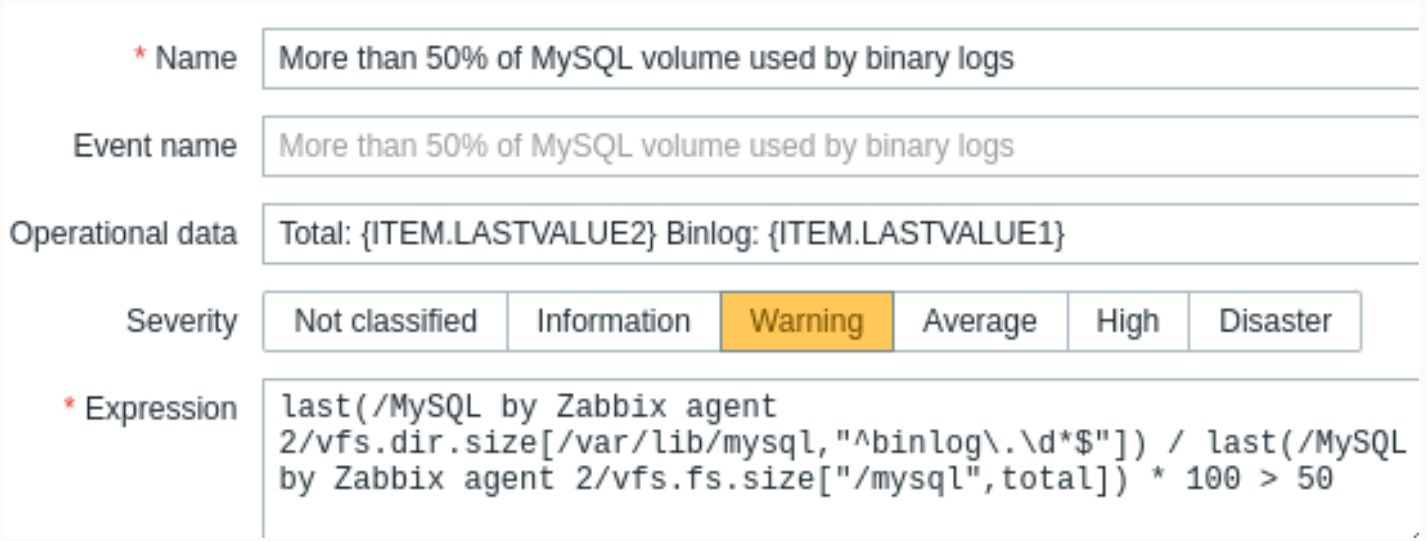
What can we monitor with Zabbix (2/2)

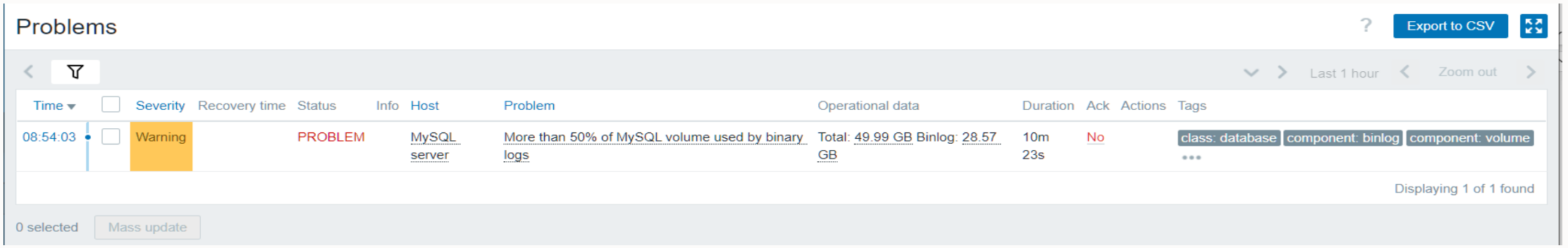
Beyond the default with a custom trigger

- Example of custom trigger for **Binary Log going over 50% of MySQL volume space**

- **Binlog disk occupation in %:**
  `(Size of binlogs / size of volume) * 100`

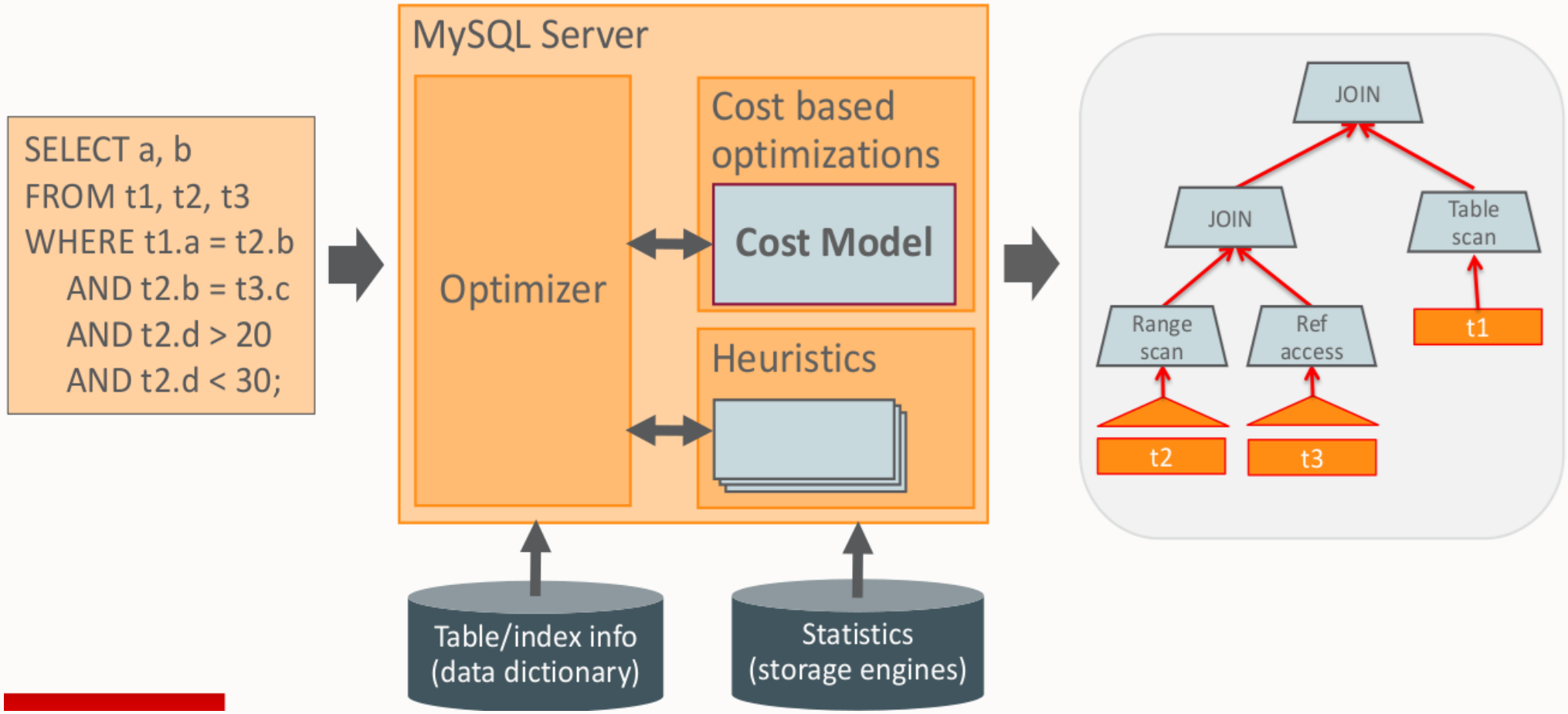
  A very usual but yet overlooked cause of issues!!





Query execution

MySQL Optimizer *(simplified)*



Analyzing Queries – EXPLAIN

Provide information about how MySQL executes statements

- The set of operations that the optimizer chooses to perform the most efficient query is called the **Query Execution Plan** or **EXPLAIN plan**
- MySQL explains how it would process the statement, including information about how tables are joined and in which order
- Returns a row of information for each table used in the statement
- Lists the tables in the output in the order that MySQL would read them while processing the statement
  - You can see where you should add indexes to tables so that the statement executes faster by using indexes to find rows
  - You can check whether the optimizer joins the tables in an optimal order
- EXPLAIN works with SELECT, DELETE, INSERT, REPLACE, and UPDATE statements

Analyzing Queries – EXPLAIN

Provide information about how MySQL executes statements

```
EXPLAIN SELECT distinct title FROM titles INNER JOIN salaries USING (emp_no) WHERE salary > 155000
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: salaries
   partitions: NULL
         type: range
possible_keys: PRIMARY,idx_salary
          key: idx_salary
      key_len: 4
          ref: NULL
         rows: 7
     filtered: 100.00
        Extra: Using where; Using index; Using temporary
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: titles
   partitions: NULL
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: employees.salaries.emp_no
         rows: 1
     filtered: 100.00
        Extra: Using index
```

Range scan

New index is used

Estimation of 7 rows examined (vs 3M)

Covering index (use only the index tree)

Former QEP

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: salaries
   partitions: NULL
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2838426
     filtered: 0.00
        Extra: Using where; Using temporary
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: titles
   partitions: NULL
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: employees.salaries.emp_no
         rows: 1
     filtered: 100.00
        Extra: Using index
```

Analyzing Queries – EXPLAIN ANALYZE

Provide information about how MySQL executes statements

- Run a statement and produces EXPLAIN output along with timing and additional, iterator-based, information about how the optimizer's expectations matched the actual execution
- The following information is provided:
  - Estimated execution cost
  - Estimated number of returned rows
  - Time to return first row
  - Time to return all rows (actual cost), in milliseconds
  - Number of rows returned by the iterator
  - Number of loops
- Can be used with SELECT statements, as well as with multi-table UPDATE and DELETE statements

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

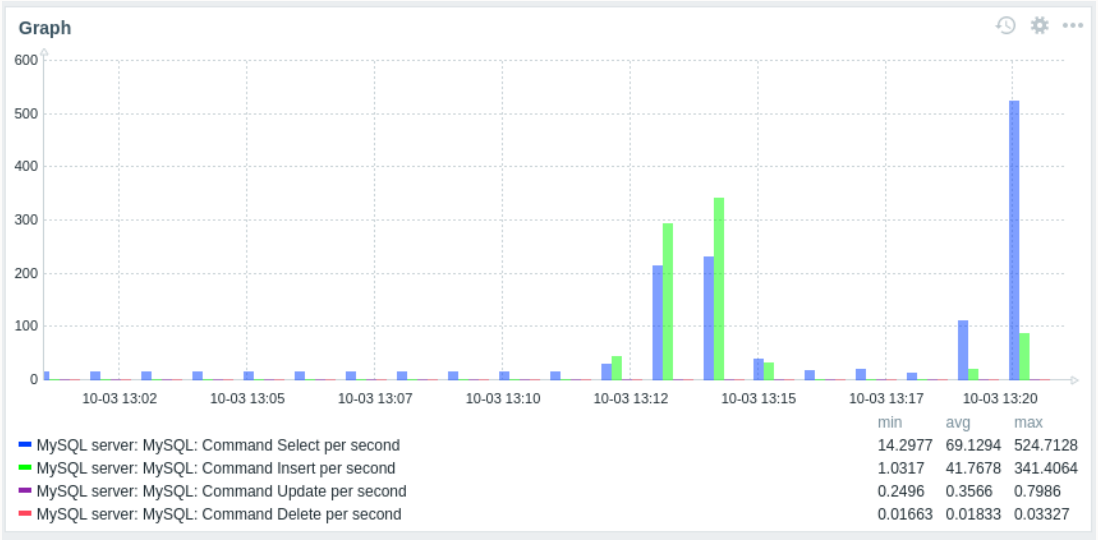
Design Questions

Starting with an efficient database design makes it easier for team members to write high-performing application code

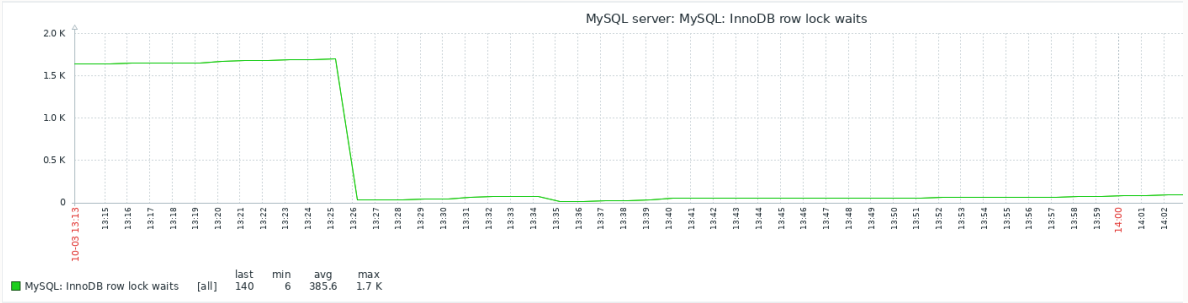
- **Do the columns have the right data types & clauses?**
- **Does each table have the appropriate columns for the type of work?**
- **Am I using the most efficient (smallest) data type possible?**
- **Do the tables have a primary key?**
- **Do I have the right indexes in the right places to make queries efficient?**

What can we monitor with Zabbix

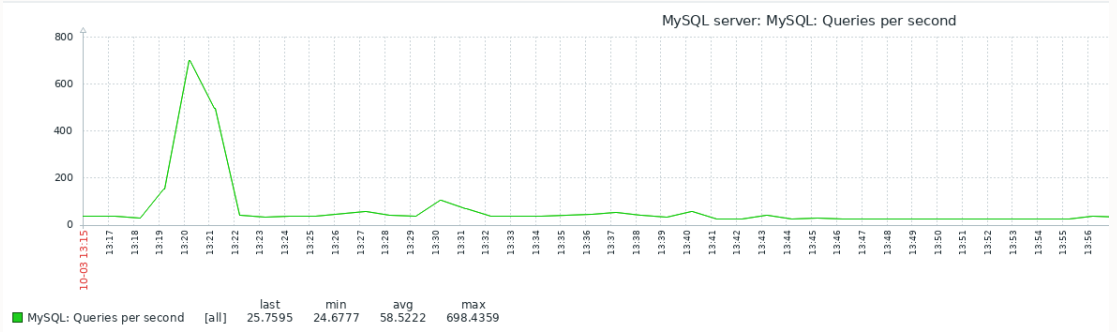
- **Commands per second:**
  - Delete per second
  - Select per second
  - Insert per second
  - Update per second



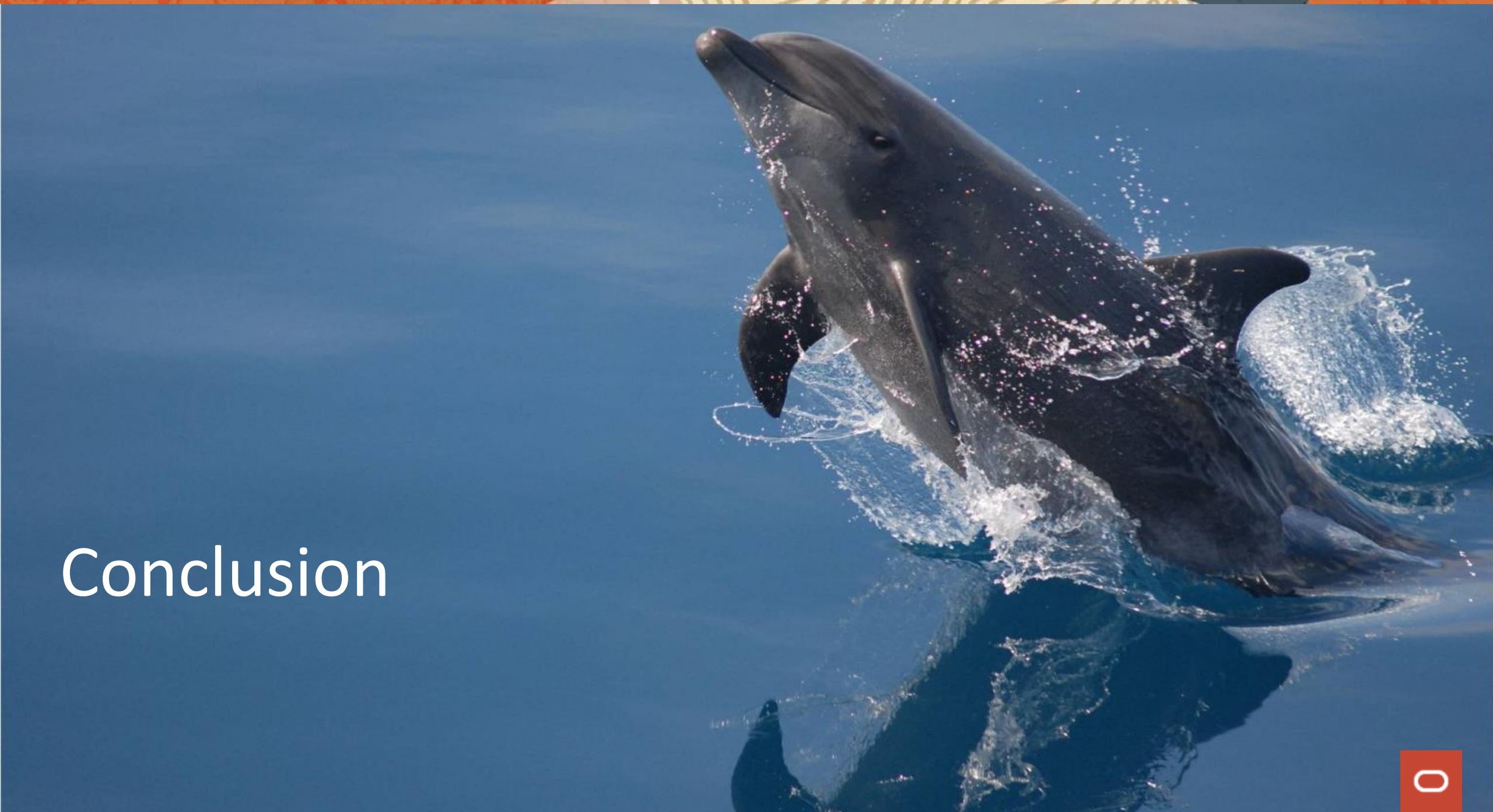
- **InnoDB row lock waits**



- **Queries per second**



Conclusion



What we just did...

- Which are the most important MySQL parameters monitored by Zabbix
- Reviewed the MySQL internal architecture
- Understood the metrics behind MySQL parameters monitored by Zabbix
- Digged in the components behind the metrics
- Checked how to control the components and optimize the metrics