MySQL Performance Optimization and Troubleshooting with PMM

Peter Zaitsev, CEO, Percona

Percona Technical Webinars

9 May 2018

Few words about Percona Monitoring and Management (PMM)

100% Free, Open Source database troubleshooting and performance optimization platform for MySQL and MongoDB

Based on Industry Leading Technology

Roll your own in and out of the Cloud

PERCONA

Monitoring and Management

Exploring Percona Monitoring and Management

You should be able to install PMM in 15 minutes or less

- http://bit.ly/InstallPMM

Would like to follow along in the demo ?

- https://pmmdemo.percona.com

In the Presentation

Practical approach to deal with some of the common MySQL Issues

PMM is not just for MySQL

- Supports MongoDB as well
- Other databases can be added via External Exporters
- This Presentation is MySQL Focused

Assumptions

You're looking to Have your MySQL Queries Run Faster

You want to troubleshoot sudden MySQL Performance Problem

You want to find way to run more efficiently (use less Resources)

How to Look at MySQL Performance

Query Based Approach

- All the users (developers) care is how quickly their queries perform

Resource Based Approach

- Queries use resources. Slow Performance often caused by resource constraints

Primary Resources

- CPU
- Disk IO
- Memory
- Network

Low Resource Usage + Poor Performance

Contention

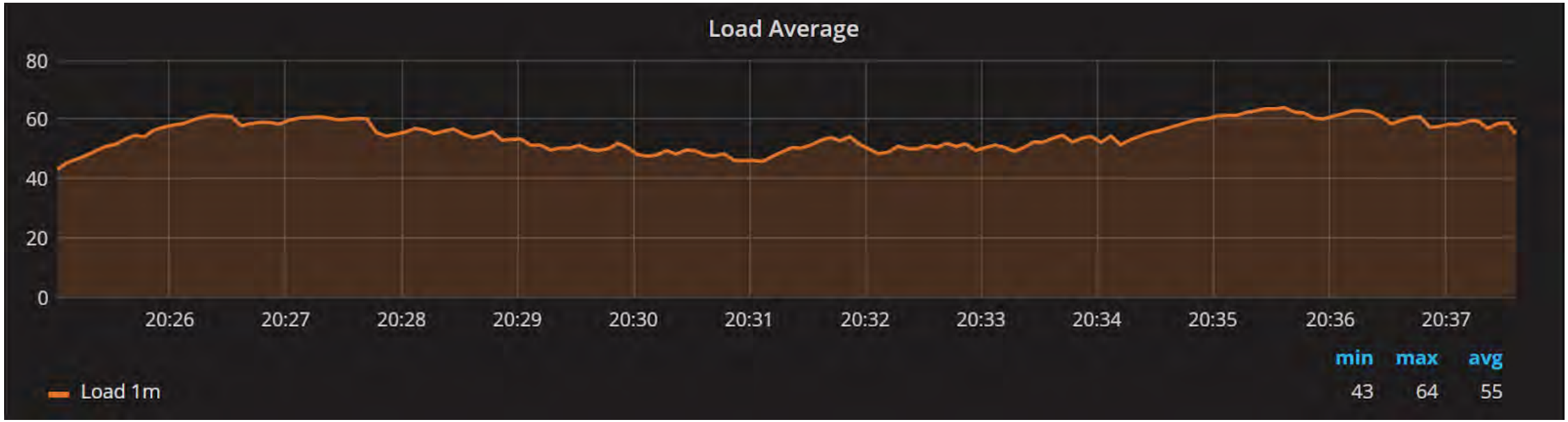
- Table Locks/Row Level Locks
- Locking/Latching in MySQL and Kernel

Mixed Resource Usage

- Single worker spending 33% on CPU
- 33% Waiting on Disk
- 33% on Network
- Will not be seen as directly constrained by any resource

Load Average

- What can you tell me about server load ?



Problems with Load Average

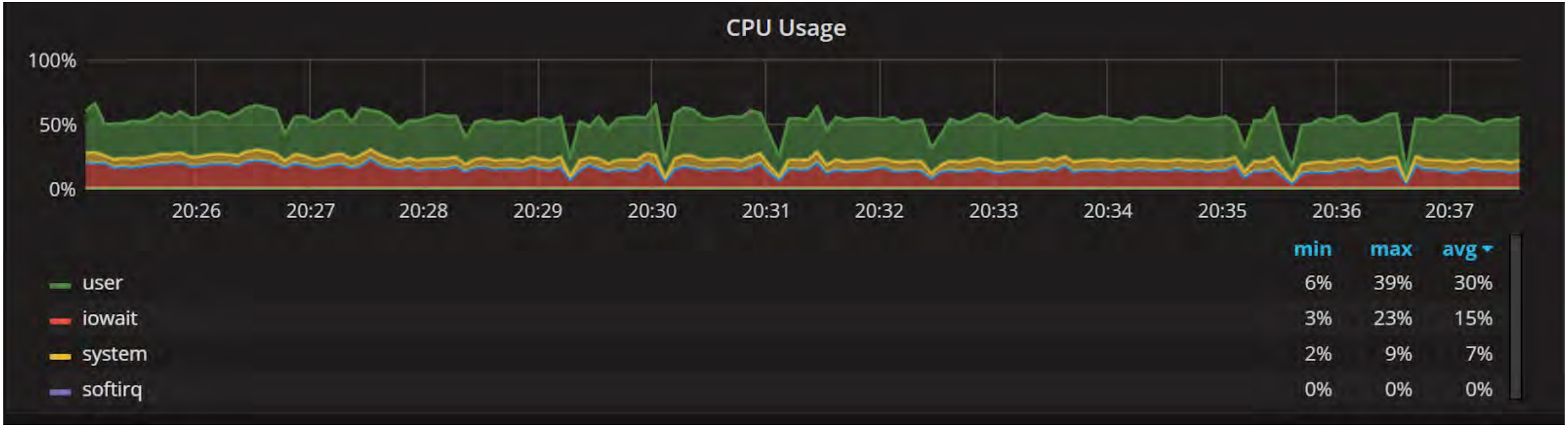
Mixes CPU and IO resource usage (on Linux)

Is not normalized for number of CPU cores available

Does not keep into account Queue Depth Needed for optimal storage performance

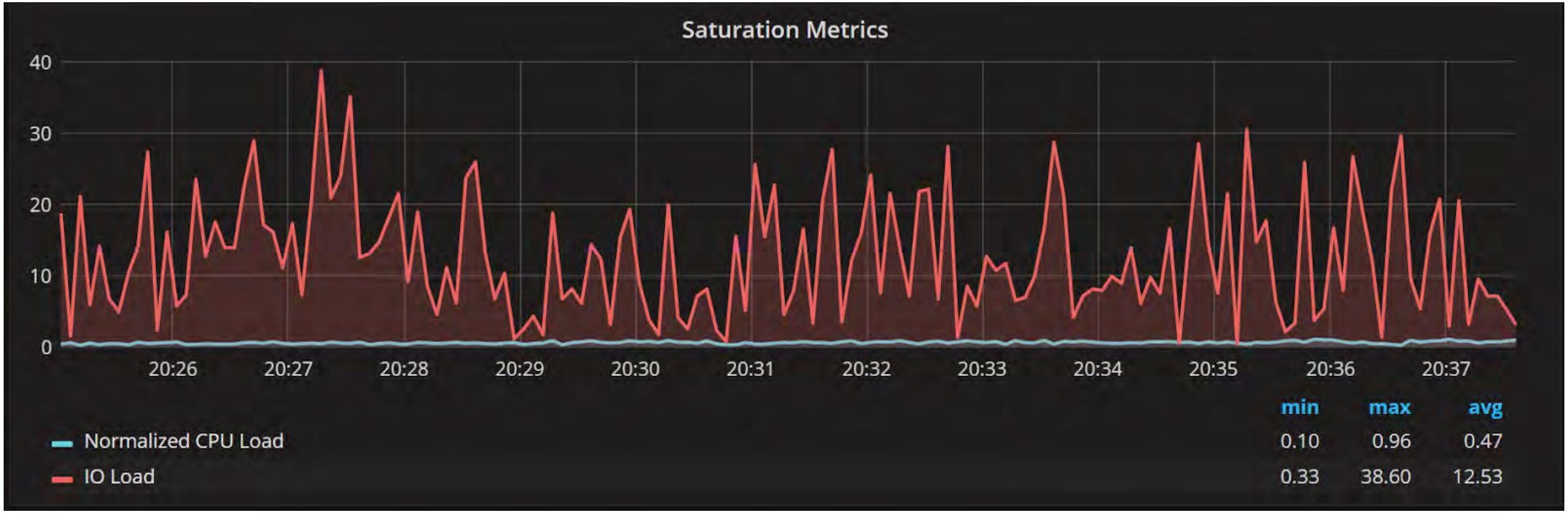
CPU Usage

- Can observe overall or per core
- Matching Load Average in the previous screen



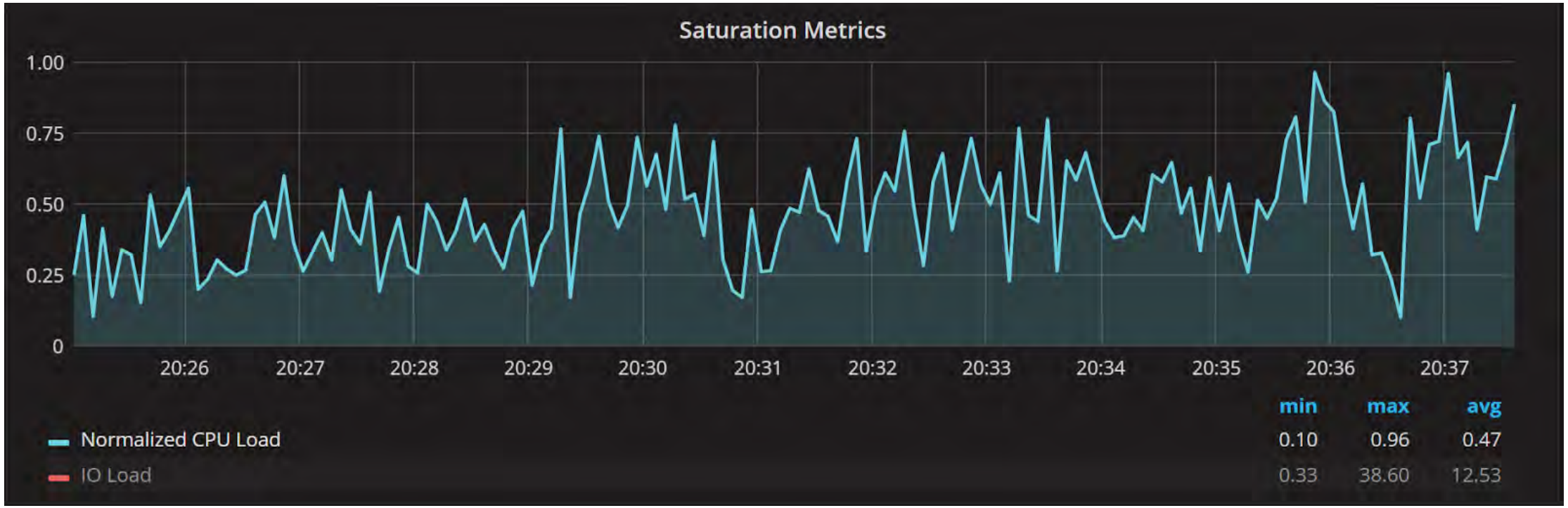
Saturation Metrics

- Good to understand where waits are happening
- IO Load is not normalized



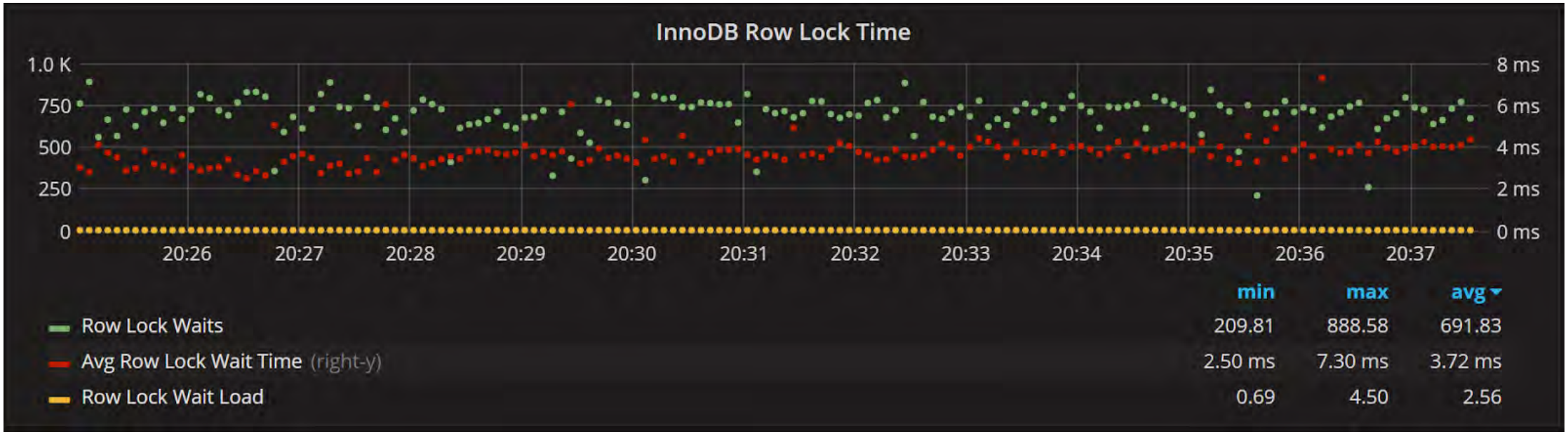
Looking at CPU Saturation Separately

- Can normalize CPU Saturation based on number of threads



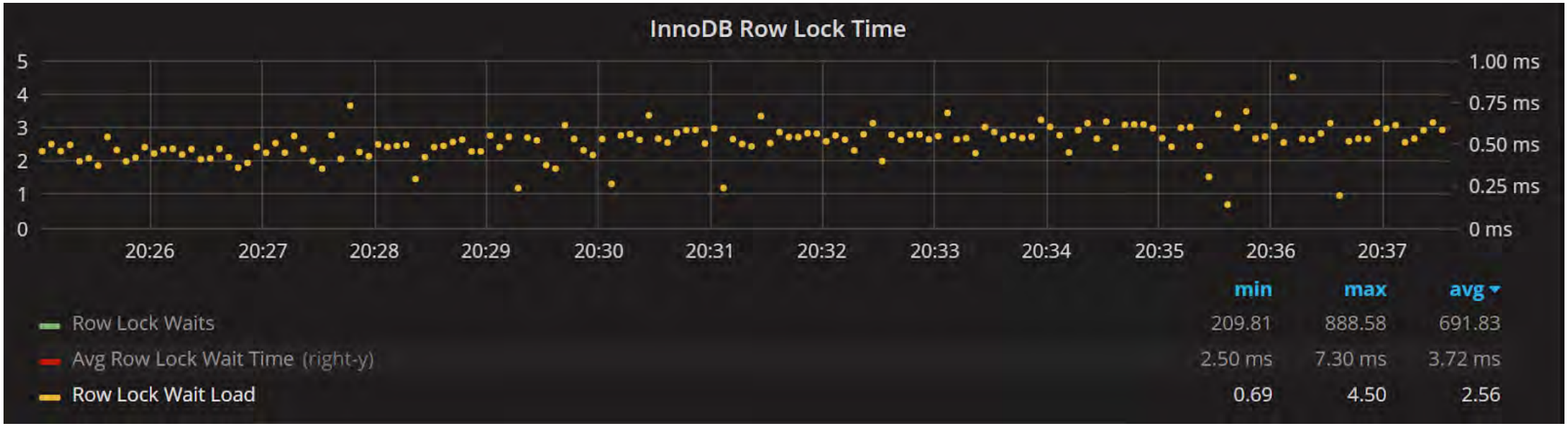
Row Locks – Logical Contention

- Row Locks are often declared by transaction semantics
- But more transactions underway also mean more locks



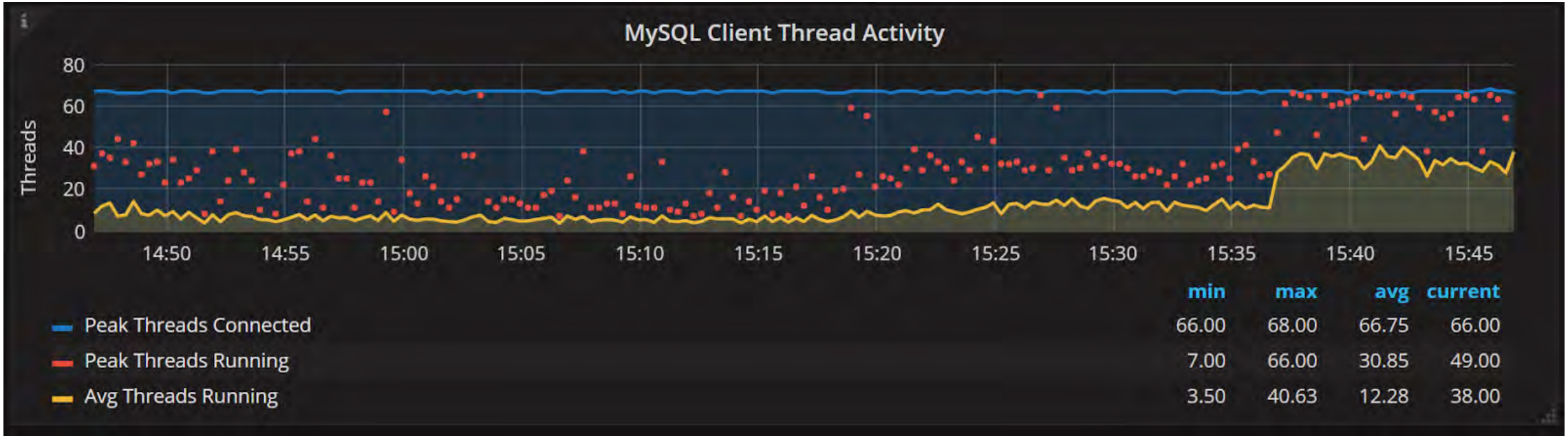
Zooming in on Row Locks Wait Load

- How many MySQL Connections are Blocked because or Row Level Lock Waits



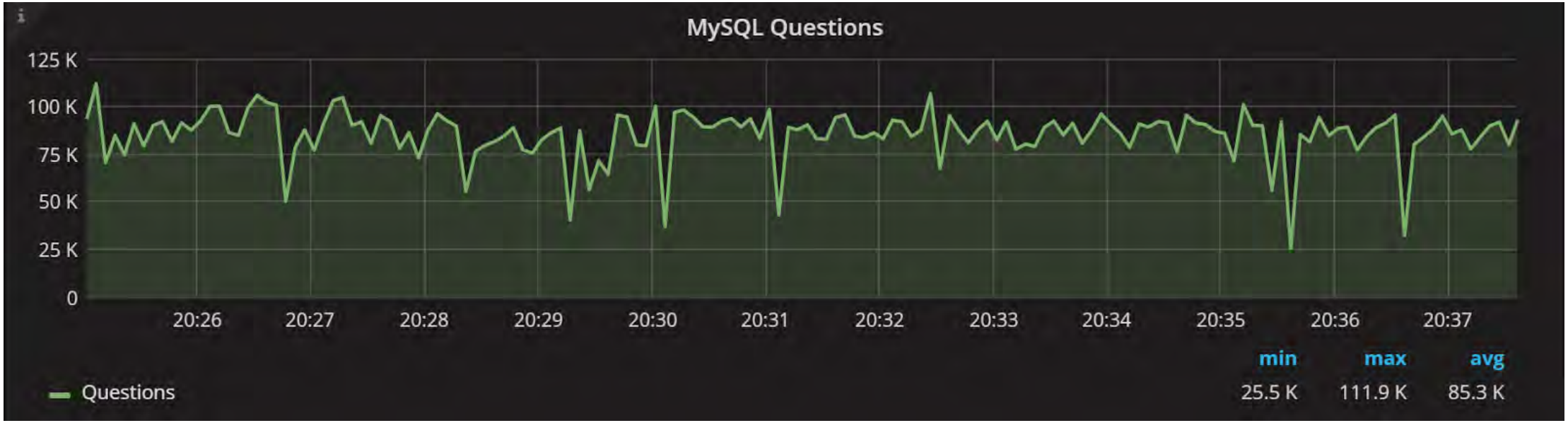
“Load at MySQL Side”

- “threads_running” - MySQL is busy handling query
- CPU ? Disk ? Row Level Locks ? Need to dig deeper



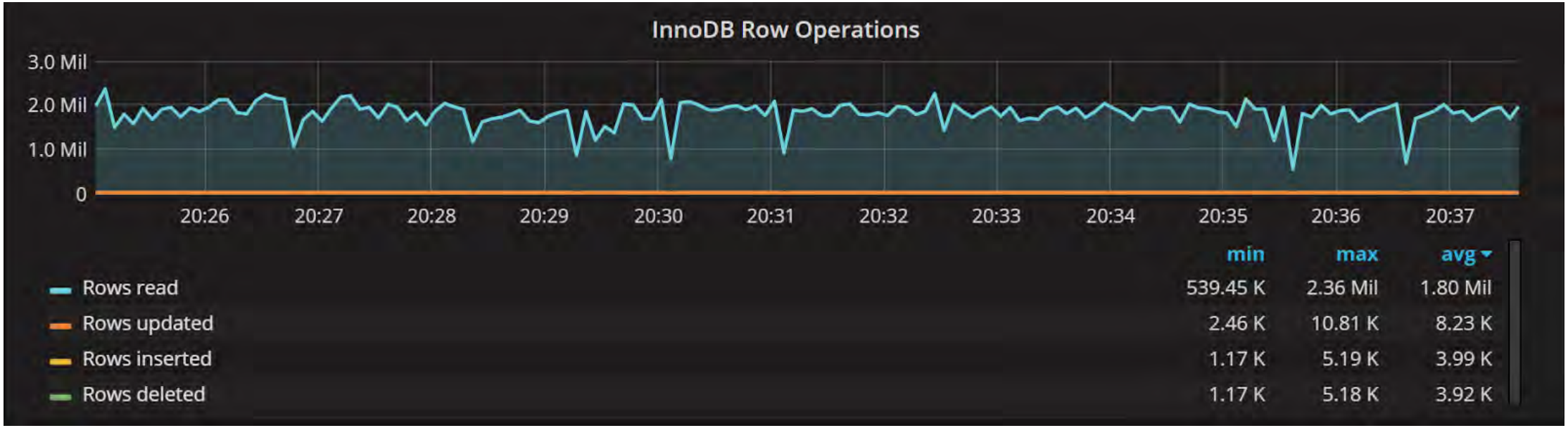
MySQL Questions – Inflow of Queries

- Are we serving more queries or less queries ?
- Any spikes or dips ?



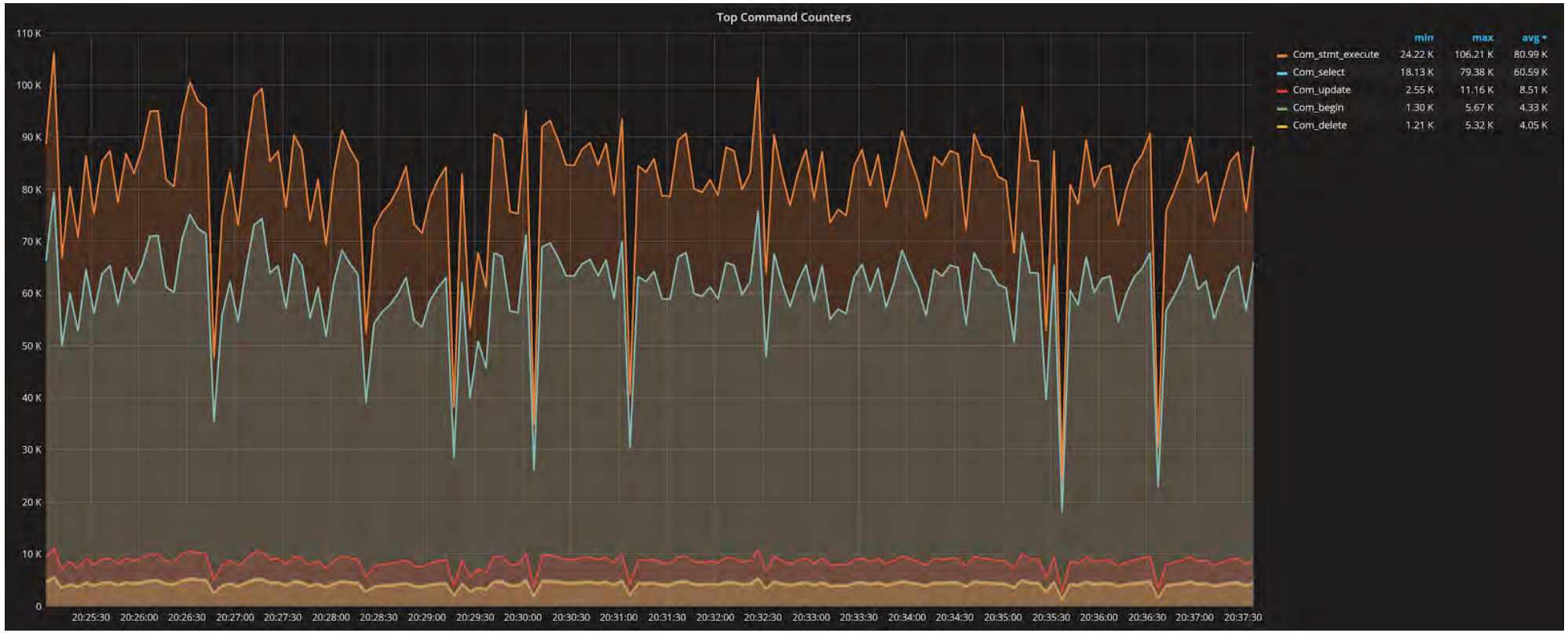
Innodb Rows – Actual Work Being Done

- Better number to think re system capacity
- Not all rows are created equal, but more equal than queries



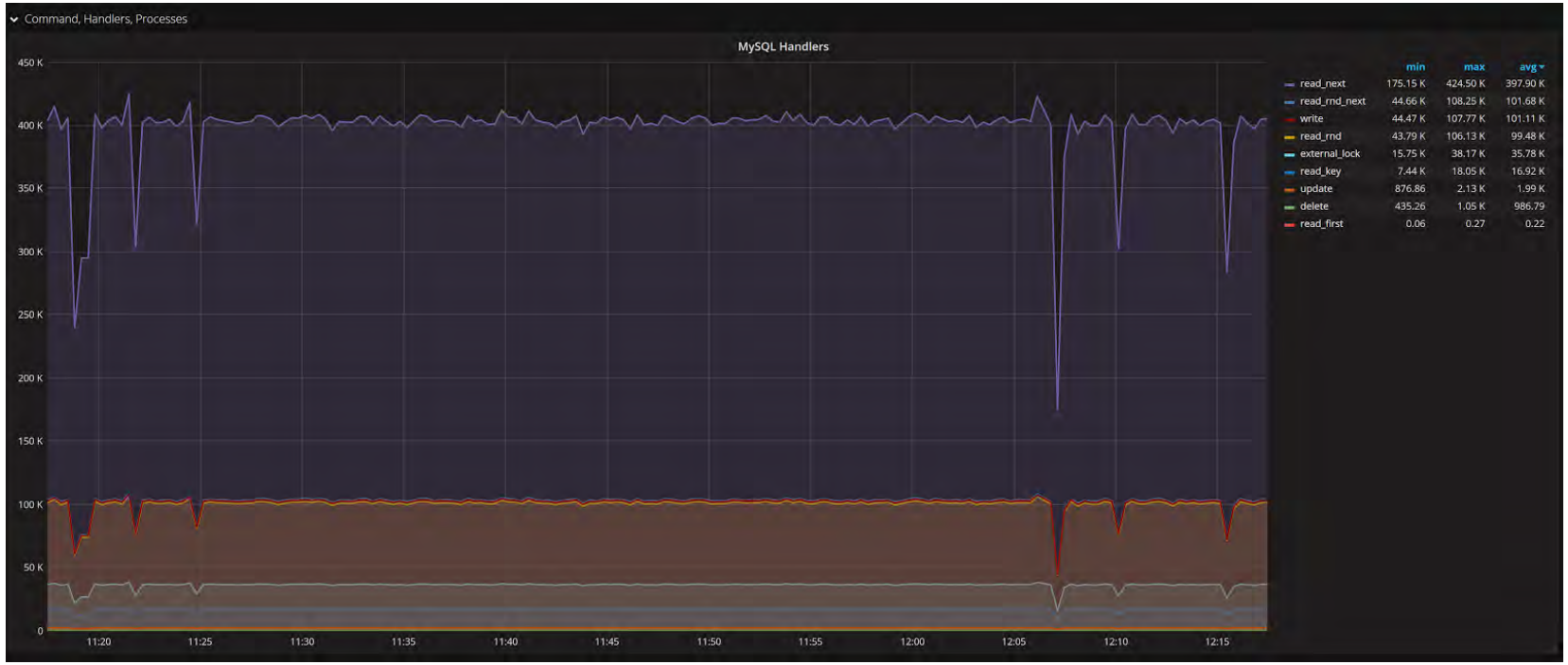
Commands – What kind of operations

- Note if prepared statements are used MySQL is “double counting”



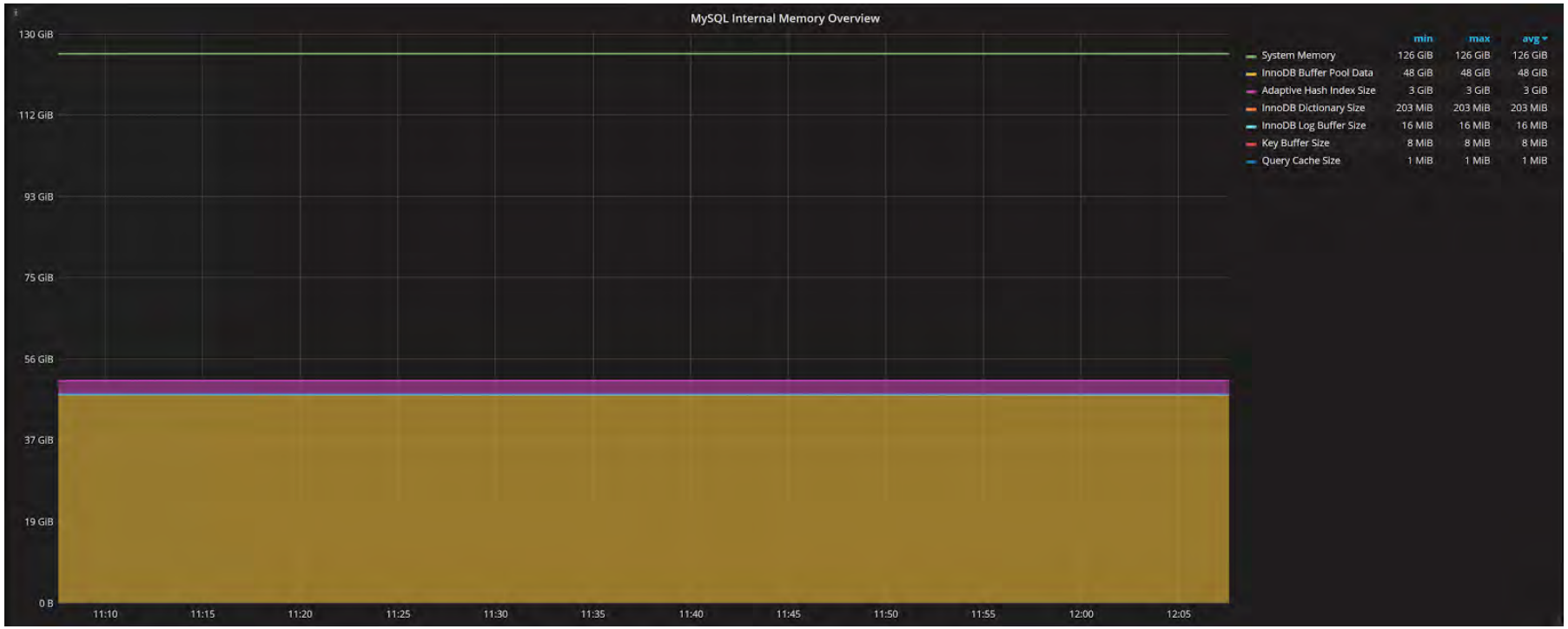
MySQL “Handlers” low lever row access

- Works for all storage engines
- Gives more details on access type
- Mixes Temporary Tables and Non-Temporary tables together



Memory usage by MySQL

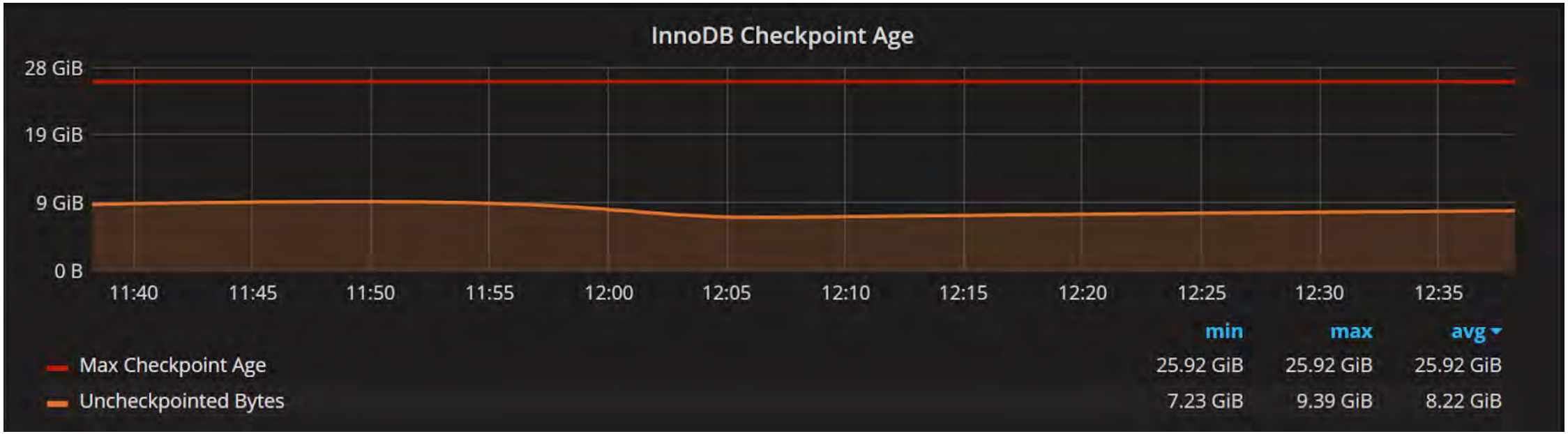
Leave some memory available for OS Cache and other needs



Innodb in Depth

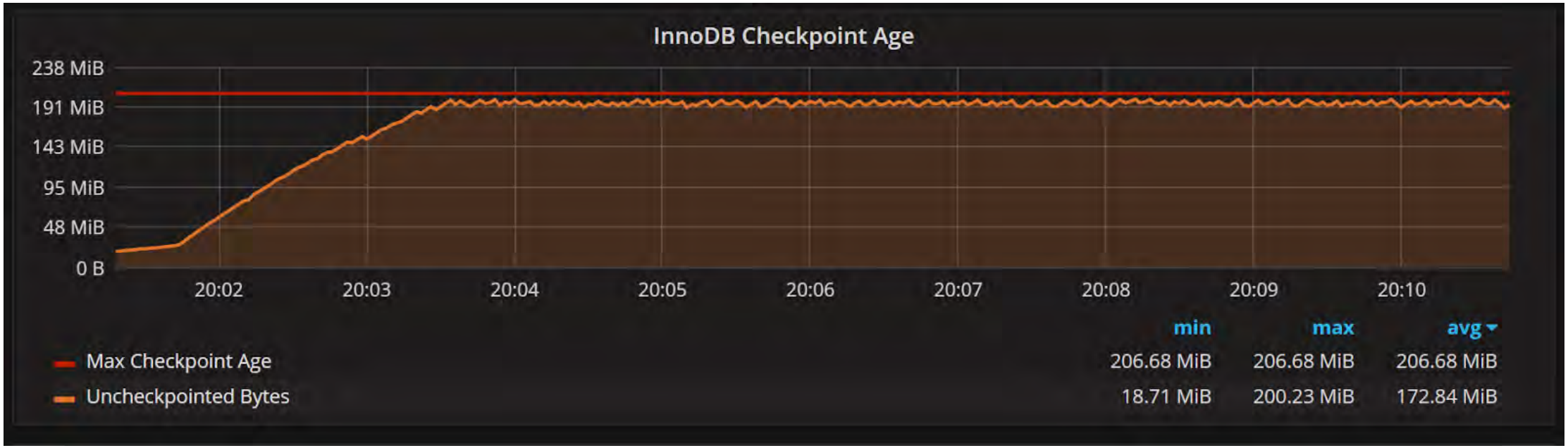
Innodb Checkpointing

- The log file size is good enough as Uncheckpointed bytes are fraction of log file size



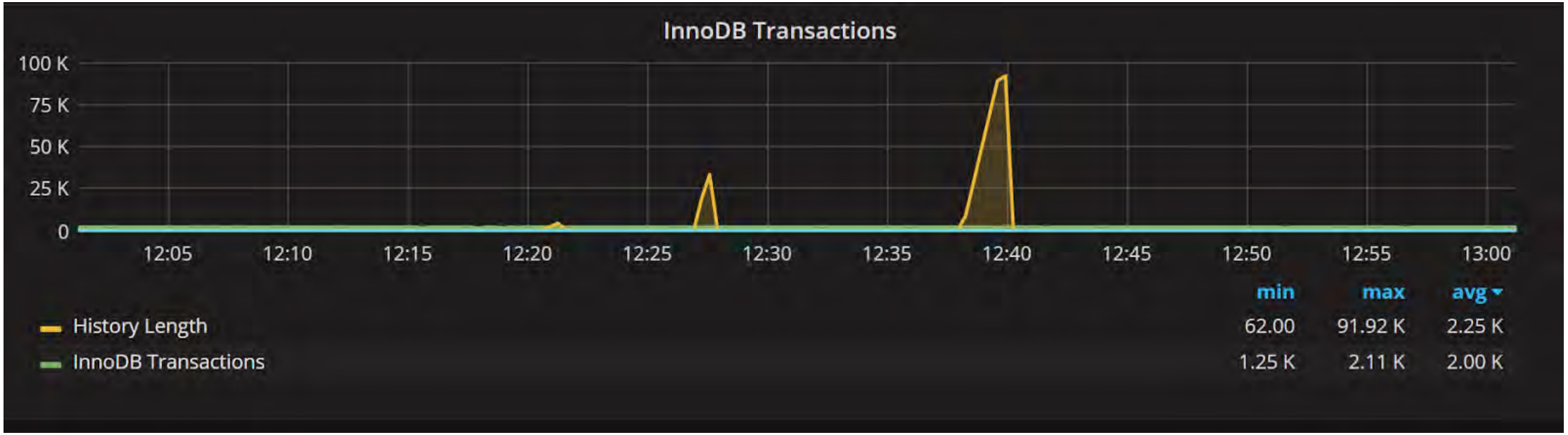
Innodb Checkpointing

- Very Close – Innodb Log File Size too small for optimal performance



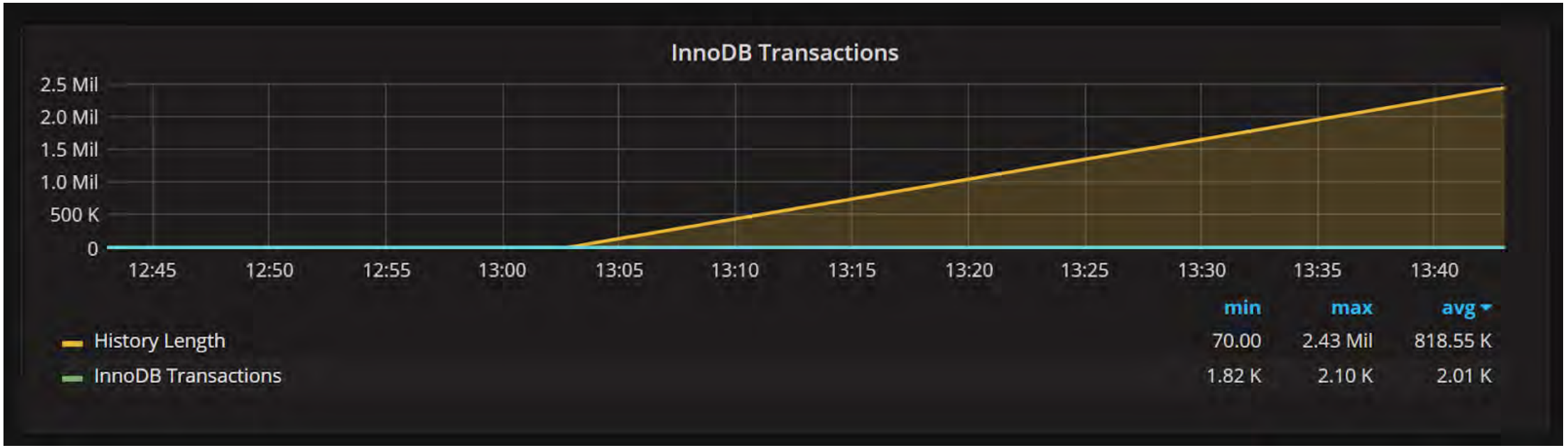
Innodb Transaction History - not yet Purged Transactions

- Short term spikes are normal if some longer transactions are ran on the system



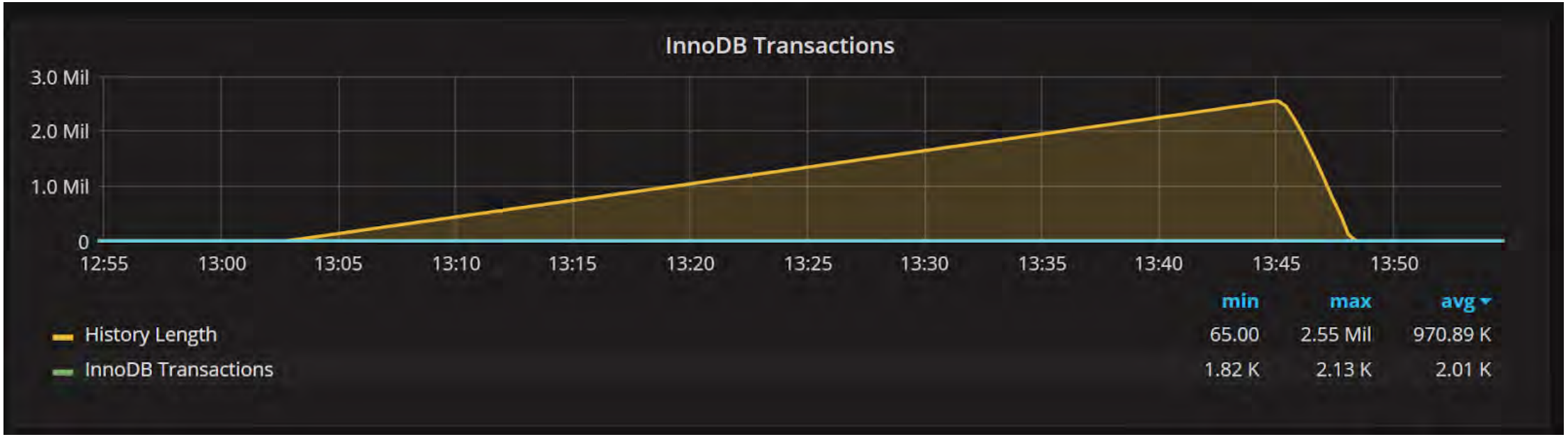
Innodb Transaction History

- Growth over long period of time without long queries in the processlist
- Often identifies orphaned transactions (left open)



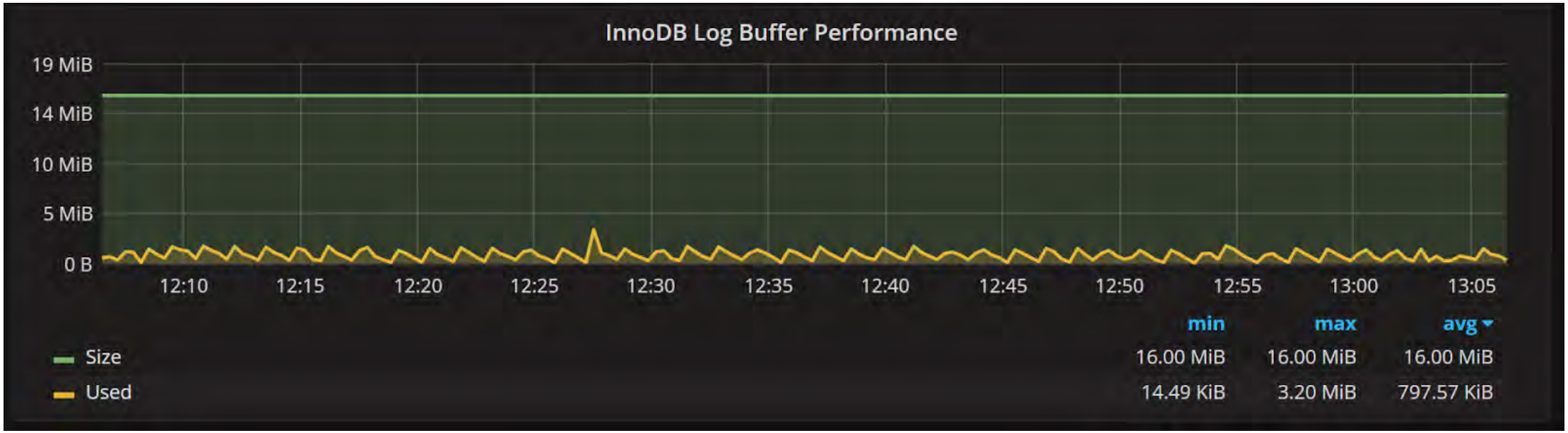
Transaction History Recovery

- If Backlog is resolved quickly it is great
- If not you may be close to the limit of purge subsystem



Is your Innodb Log Buffer Large Enough?

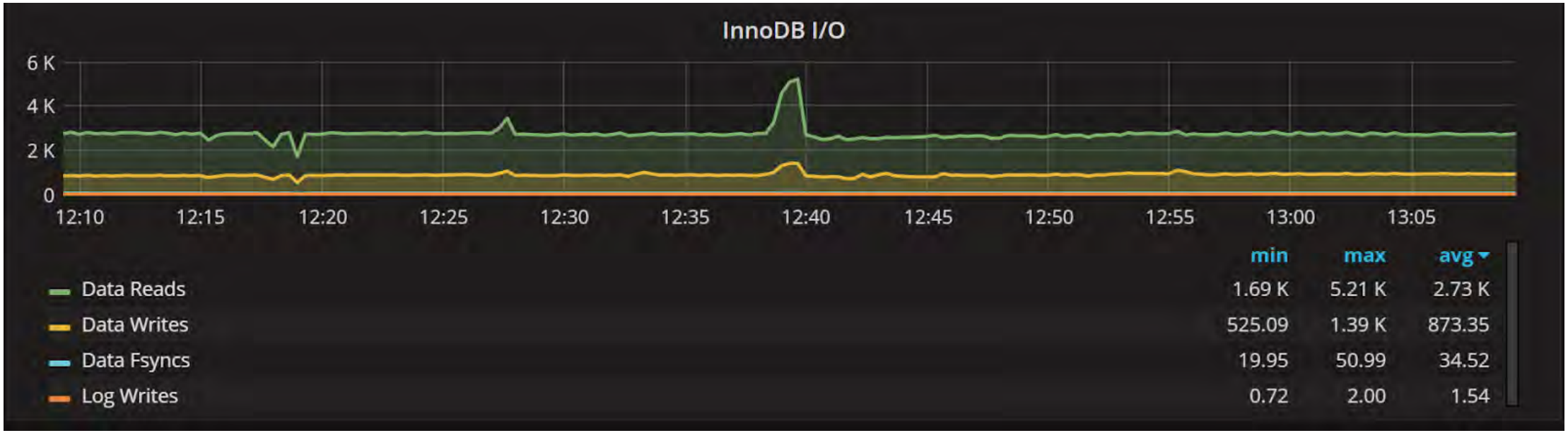
- You will be surprised to see how little log buffer space Innodb needs



Another way to look at Logging Performance

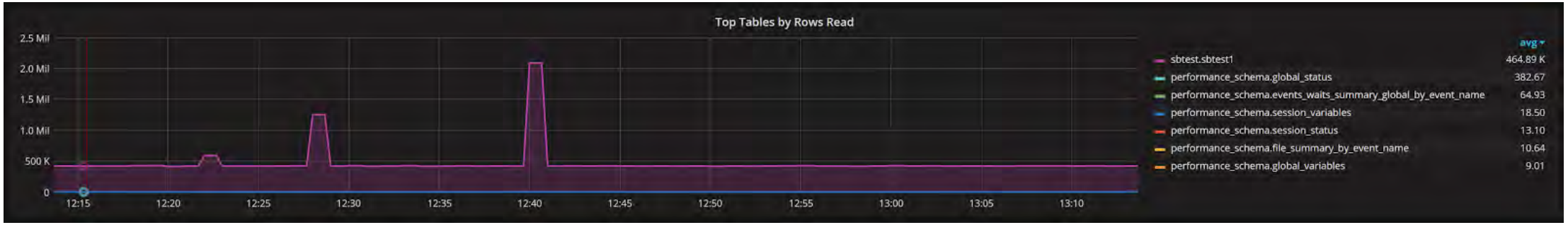
Innodb IO

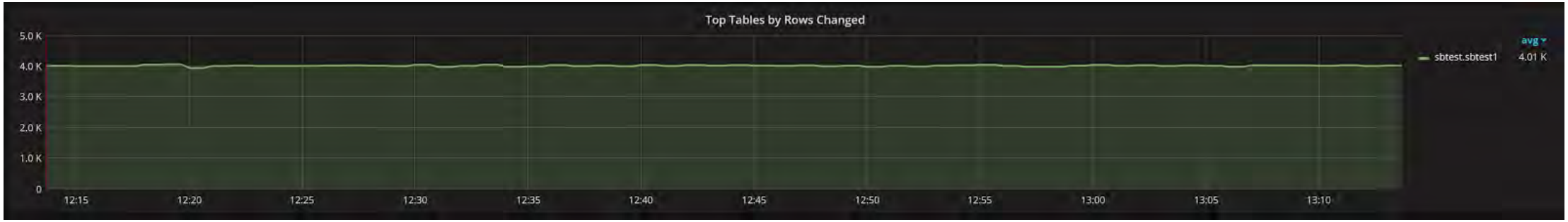
- Will often roughly match disk IO
- Allows to see the writes vs fsyncs



Hot Tables

- It is often helpful to know what tables are getting most Reads
- And Writes





Hot Tables through Performance Schema

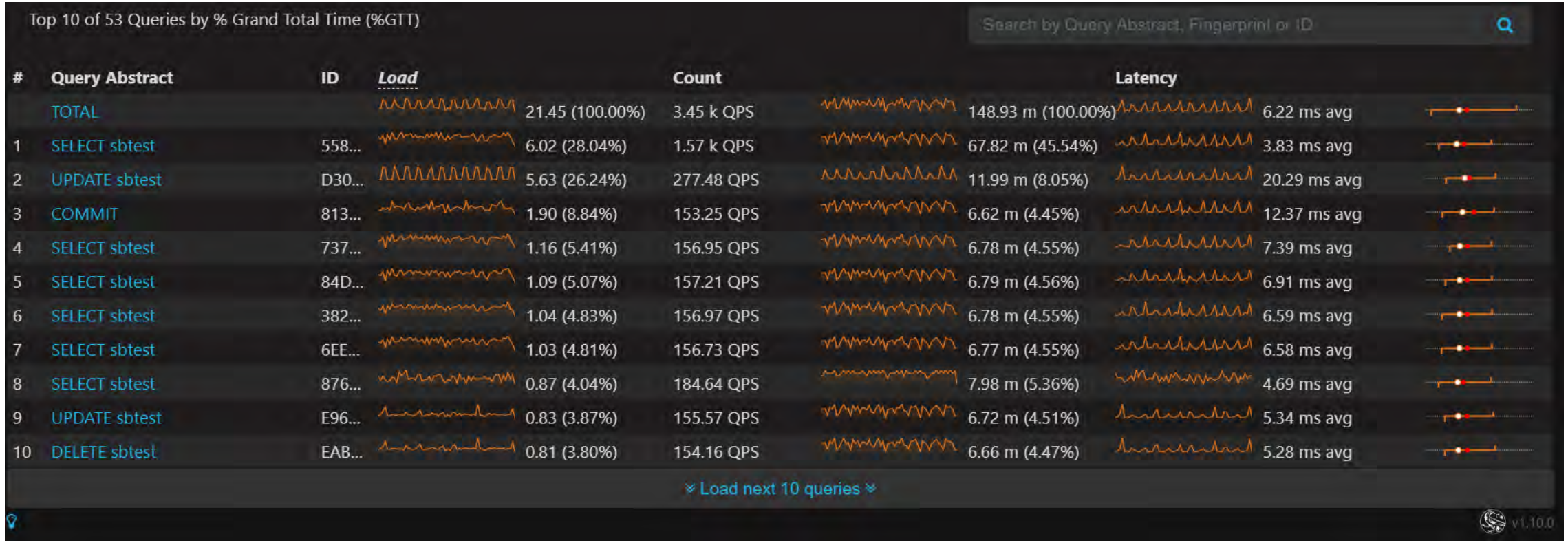
- Even more details available in Performance Schema
- Load is a better measure of actual cost than number of events

Most Active Indexes

- See through which index queries access tables

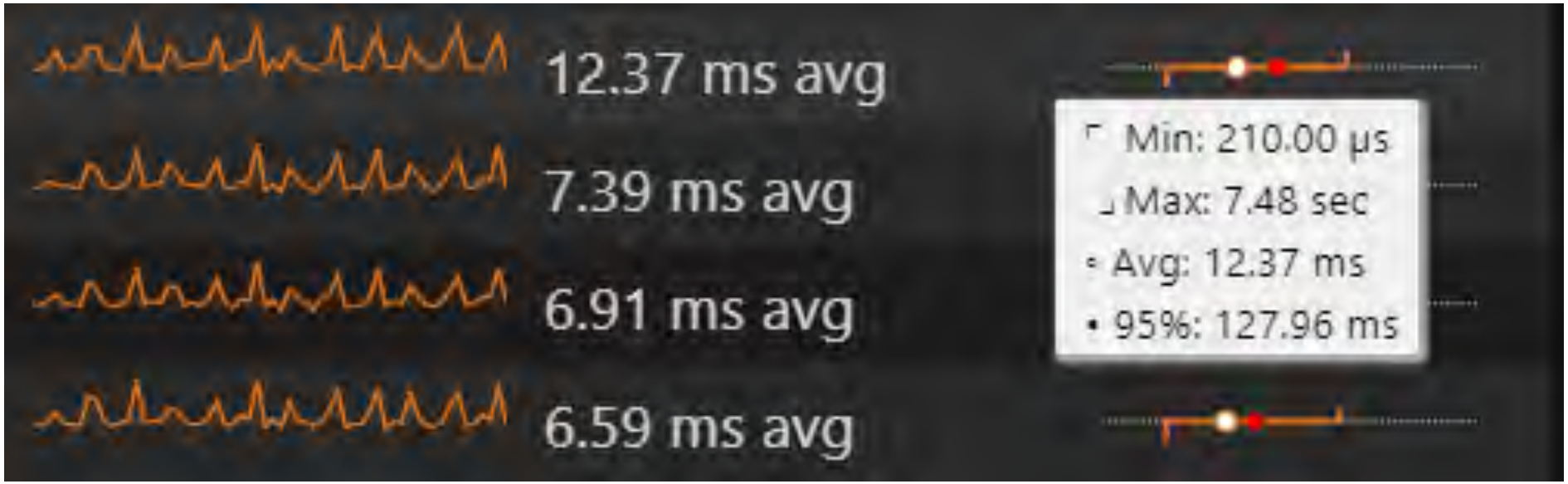
What about Queries causing the most load?

- Can examine through Query Analytics application



Latency Details Explored

- Not enough to look at Average Latency



What are Top Queries ?

Queries Sorted by their “Load”

Query ran 10 times over second each time taking 0.2 sec will be load 2

Not making a difference between queries “causing” the load or just impacted by it

Whole Server Summary #1

- Server Summary Gives a good idea what is going on query wise

Server Summary

Metrics

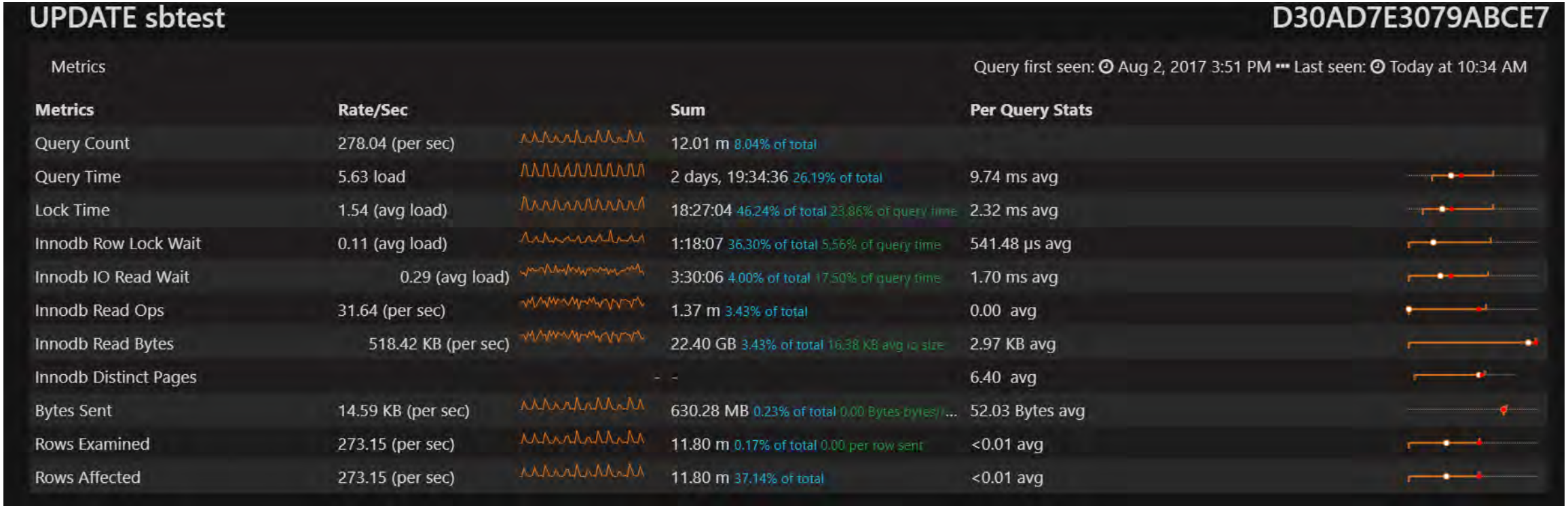
Metrics	Rate/Sec	Sum	Per Query Stats
Query Count	3.46 k (per sec)	149.43 m	
Query Time	21.50 load	10 days, 17:59:55	7.19 ms avg
Lock Time	3.33 (avg load)	1 days, 15:54:08 16.37% of query time	1.18 ms avg
Innodb Row Lock Wait	0.30 (avg load)	3:35:11 1.42% of query time	101.71 µs avg
Innodb IO Read Wait	7.29 (avg load)	3 days, 15:31:36 32.14% of query time	2.31 ms avg
Innodb Read Ops	921.56 (per sec)	39.81 m	0.09 avg
Innodb Read Bytes	15.10 MB (per sec)	652.27 GB 16.38 KB avg io size	5.00 KB avg
Innodb Distinct Pages		-	2.74 avg
Query Cache Hits	92.54 (per sec)	4.00 m 2.68% QC hit ratio	-
Rows Sent	50.95 k (per sec)	2.20 b	14.13 avg
Bytes Sent	6.35 MB (per sec)	274.26 GB 124.61 Bytes bytes/row	1.81 KB avg
Rows Examined	156.82 k (per sec)	6.77 b 3.08 per row sent	48.55 avg

Whole Server Summary #2

Rows Affected	735.51 (per sec)	31.77 m	0.00 avg
External Sorts (Filesort)	315.26 (per sec)	13.62 m 9.11% of queries	–
Cartesian Products (Full Joins)	0.03 (per sec)	1.22 k <0.01% of queries	–
Full Table Scans	6.62 (per sec)	285.87 k 0.19% of queries	–
Queries Requiring Tmp Table In Mem...	162.94 (per sec)	7.04 m 4.71% of queries	–
Number of Tmp table in Memory	178.08 (per sec)	7.69 m 1.09 per query with tmp table	0.00 avg
Queries Requiring Tmp Table on Disk	0.42 (per sec)	18.27 k 0.01% of queries	–
Number of Tmp Tables on Disk	3.20 (per sec)	138.05 k 7.56 per query with disk tmp table	0.00 avg
Total Size of Tmp Tables	20.05 MB (per sec)	865.97 GB 122.71 KB per query	5.70 KB avg

Specific Query – Update Query

- Significant part of response time comes from row level lock waits



UPDATE sbtest

D30AD7E3079ABCE7

Metrics

Query first seen: Aug 2, 2017 3:51 PM ··· Last seen: Today at 10:34 AM

Metrics	Rate/Sec	Sum	Per Query Stats
Query Count	278.04 (per sec)	12.01 m 8.04% of total	
Query Time	5.63 load	2 days, 19:34:36 26.19% of total	9.74 ms avg
Lock Time	1.54 (avg load)	18:27:04 46.24% of total 23.86% of query time	2.32 ms avg
Innodb Row Lock Wait	0.11 (avg load)	1:18:07 36.30% of total 5.56% of query time	541.48 µs avg
Innodb IO Read Wait	0.29 (avg load)	3:30:06 4.00% of total 17.50% of query time	1.70 ms avg
Innodb Read Ops	31.64 (per sec)	1.37 m 3.43% of total	0.00 avg
Innodb Read Bytes	518.42 KB (per sec)	22.40 GB 3.43% of total 16.38 KB avg io size	2.97 KB avg
Innodb Distinct Pages		- -	6.40 avg
Bytes Sent	14.59 KB (per sec)	630.28 MB 0.23% of total 0.00 Bytes bytes/r...	52.03 Bytes avg
Rows Examined	273.15 (per sec)	11.80 m 0.17% of total 0.00 per row sent	<0.01 avg
Rows Affected	273.15 (per sec)	11.80 m 37.14% of total	<0.01 avg

Expensive SELECT Query

- Examining lots of rows per each row sent

Check Query Example

- Expensive Query not poorly optimized one

Example

```
SELECT avg(length(c))
FROM sbtest1
WHERE id
   BETWEEN 13060000
        AND 13060000+30000000
```

Explain and JSON Explain

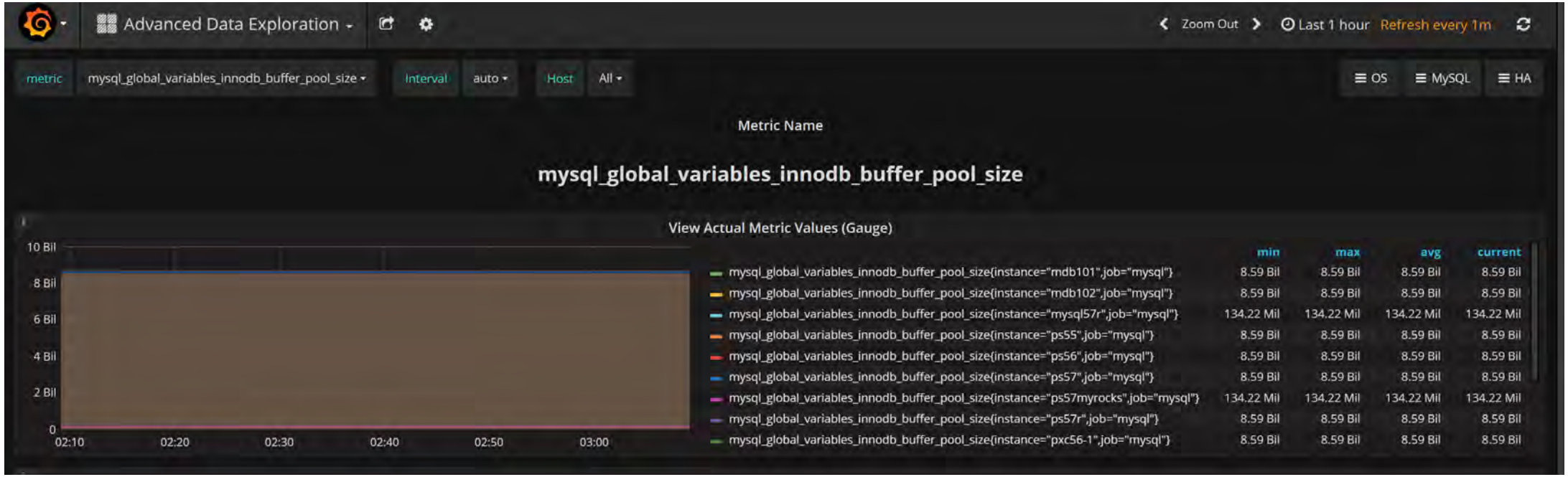
JSON

Collapse All


```
{
  "query_block": {
    "select_id": 1,
    "table": {
      "table_name": "sbtest1",
      "access_type": "range",
      "possible_keys": [
        "PRIMARY"
      ],
      "key": "PRIMARY",
      "used_key_parts": [
        "id"
      ],
      "key_length": "4",
      "rows": 37306682,
      "filtered": 100,
      "attached_condition": "(`innodb`.`sbtest1`.`id` between 13060000 and <cache>((13060000 + 30000000)))"
    }
  }
}
```

Explore Any Captured Metrics

- Standard Dashboards are only tip of the iceberg
- You can also use Prometheus directly



Lets Look at Couple of Case Studies

Impact Of Durability ?

Running sysbench with **rate=1000** to inject 1000 transactions every second

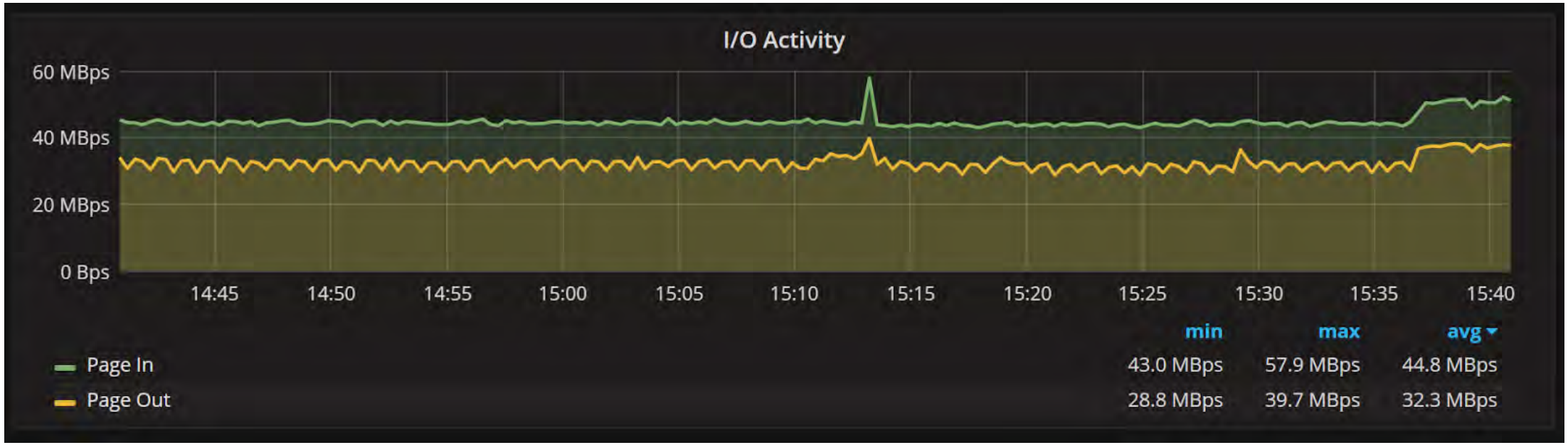
System can handle workloads with both settings

System previously running with **sync_binlog=0** and **innodb_flush_log_at_trx_commit=0**

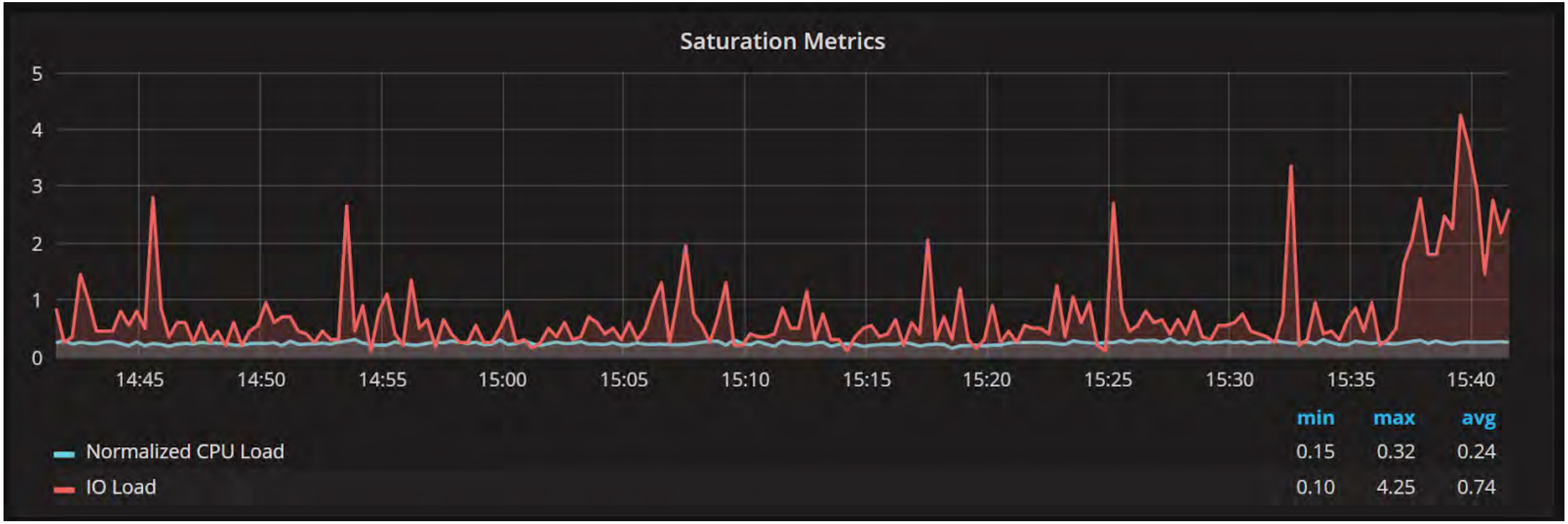
Set them to **sync_binlog=1** and **innodb_flush_log_at_trx_commit=1**

IO Bandwith

- IO Bandwidth is not significantly impacted

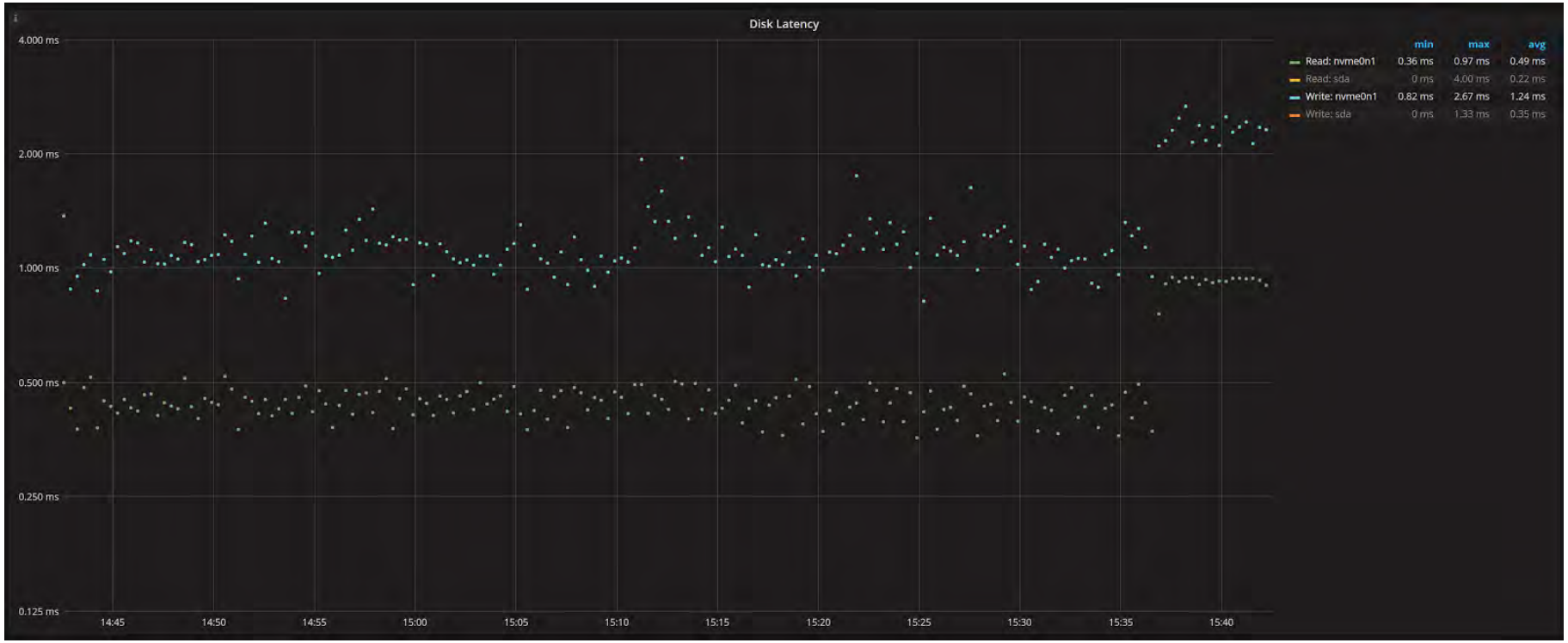


IO Saturation Jumps a Lot



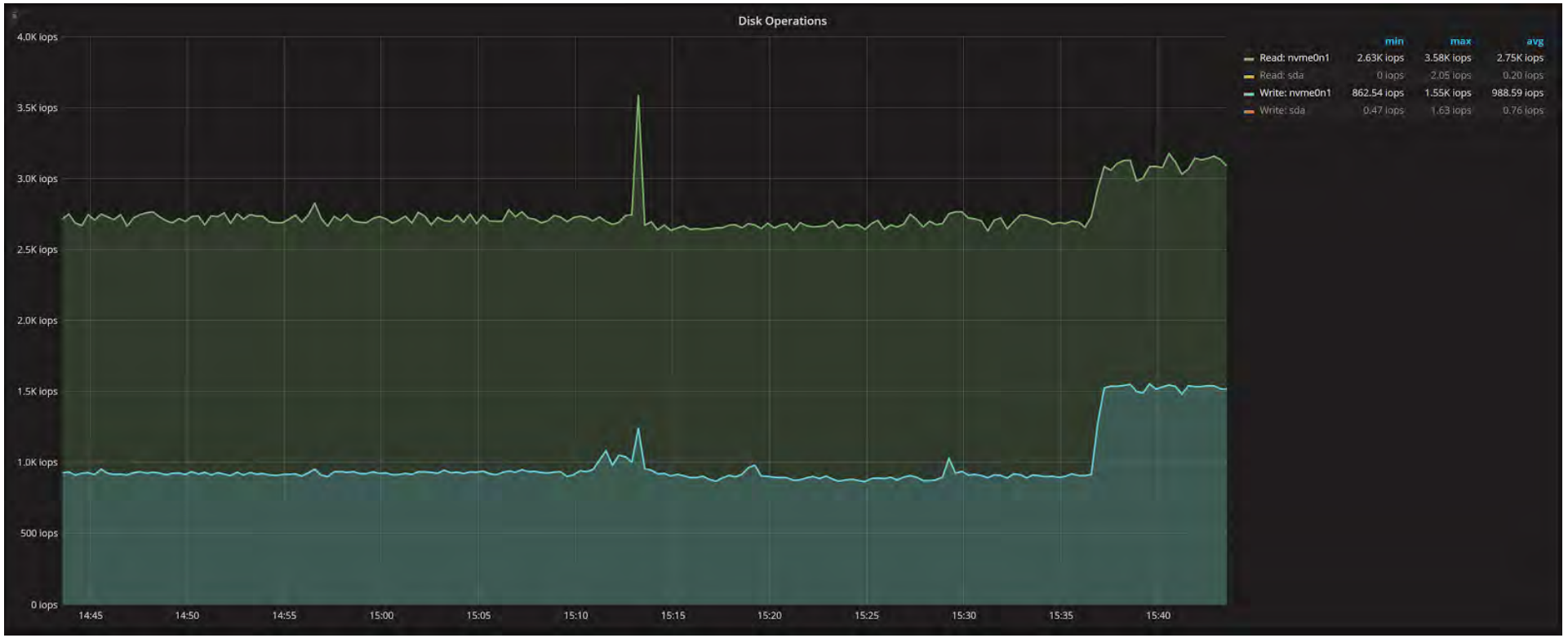
Read and Write Latencies are Impacted

- This SSD (Samsung 960 Pro) Does not like fsync() calls



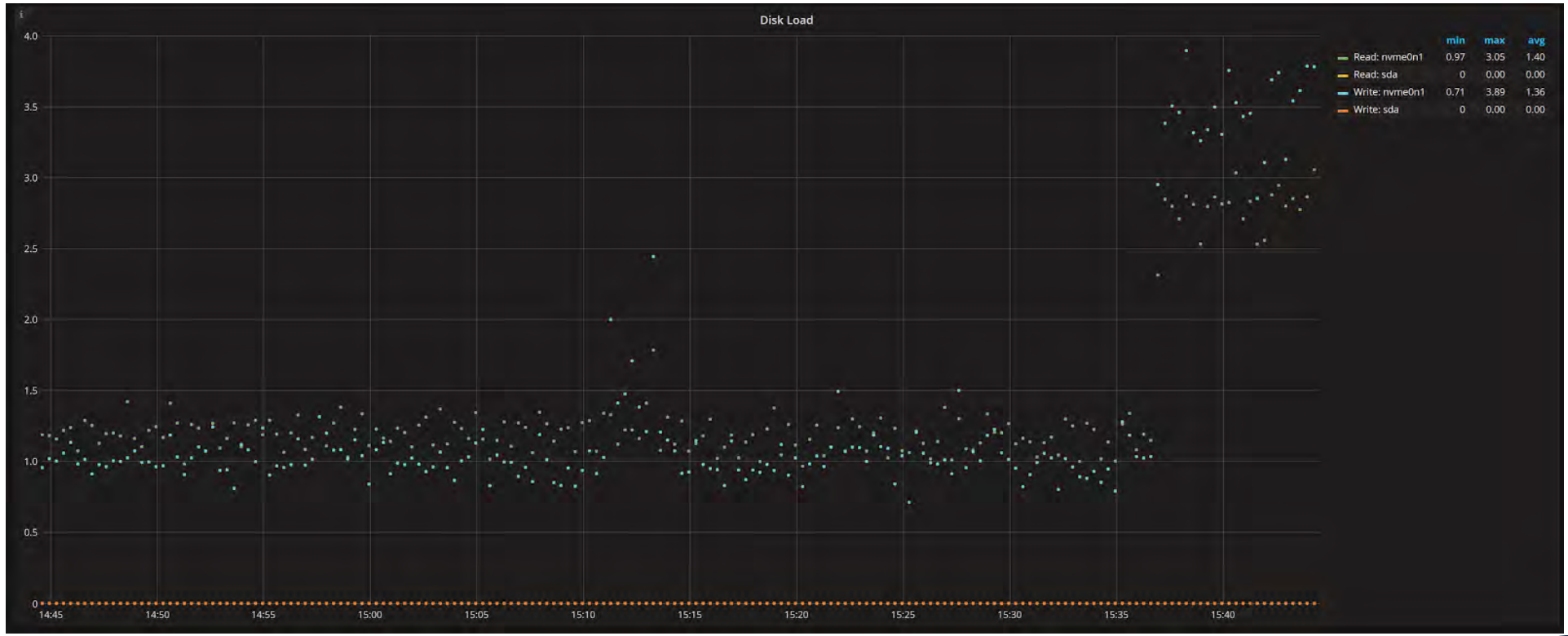
More Disk IO Operations

- Frequent Fsync() causes more writes of smaller size to storage



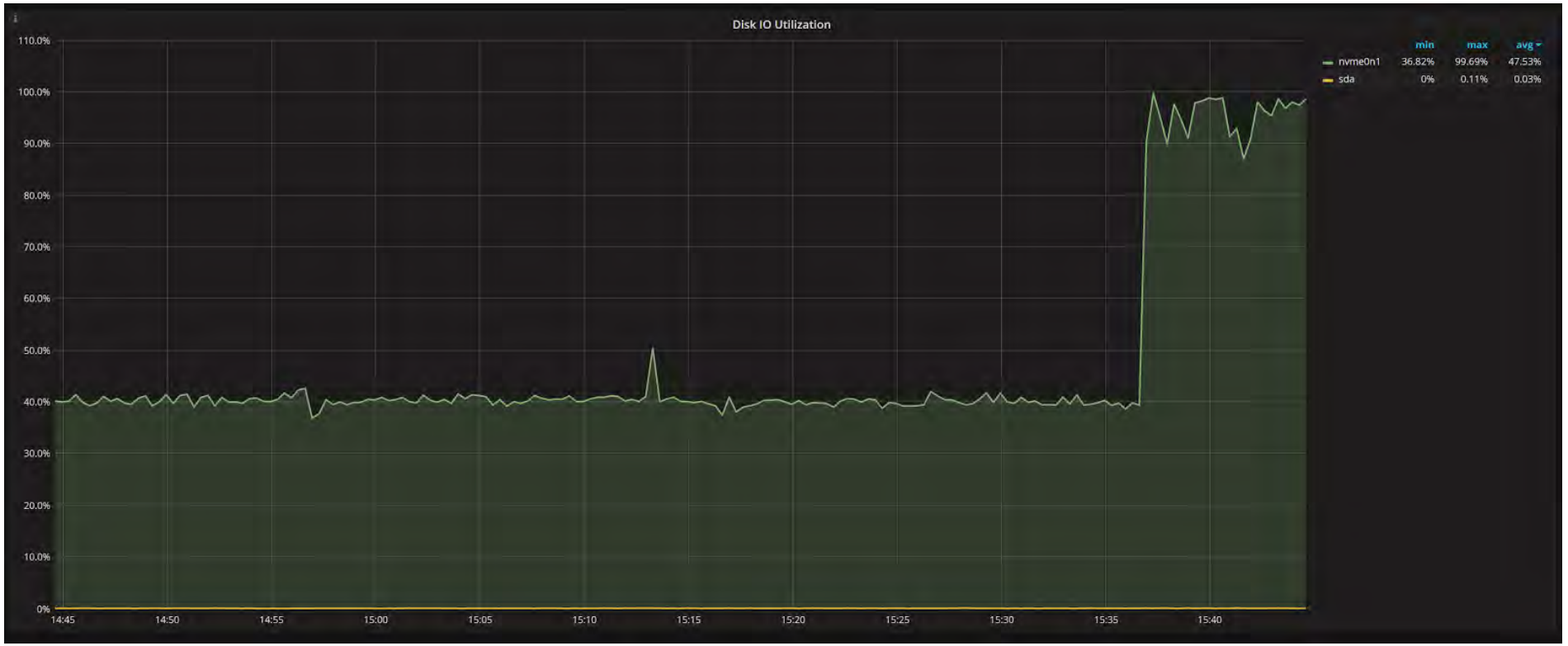
Increase In Disk IO Load

- IO Avg Latency Increase + More IOPs = Load Increase



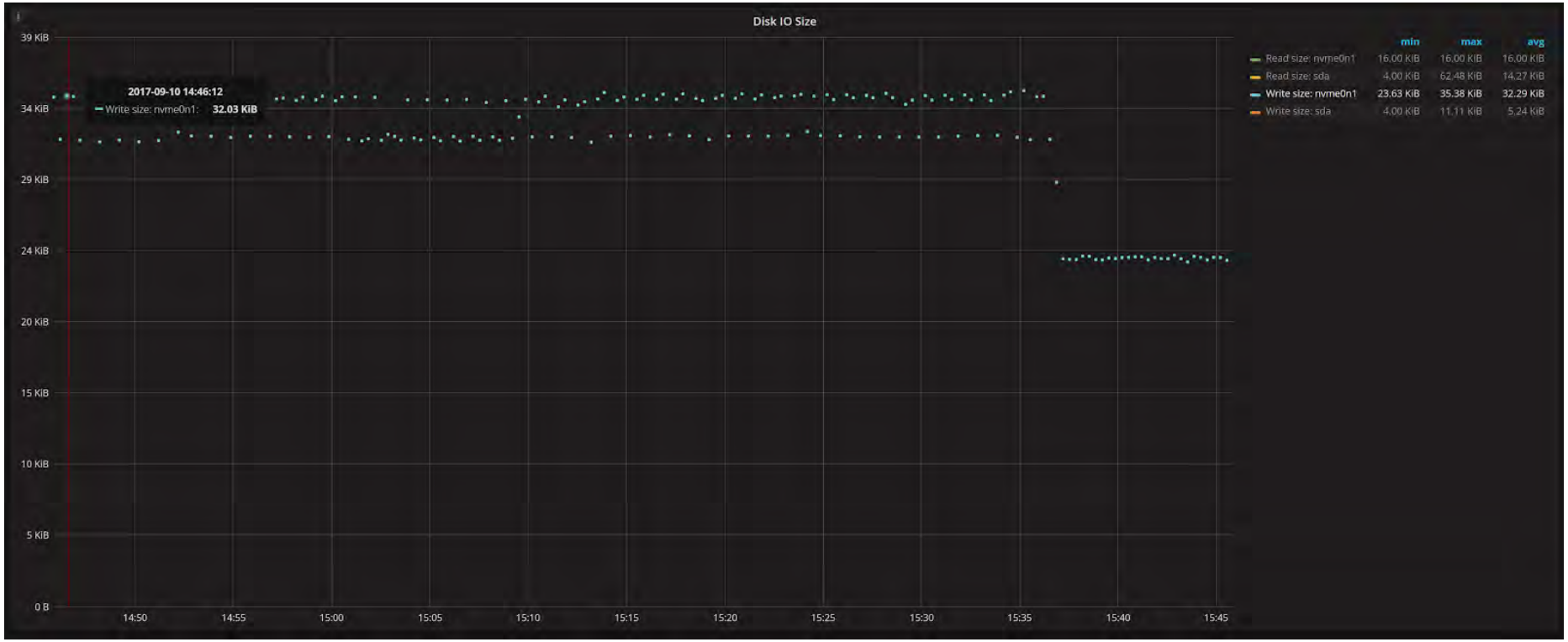
Disk IO Utilization jumps to 100%

- There is at least one disk IO Operation in flight all the time



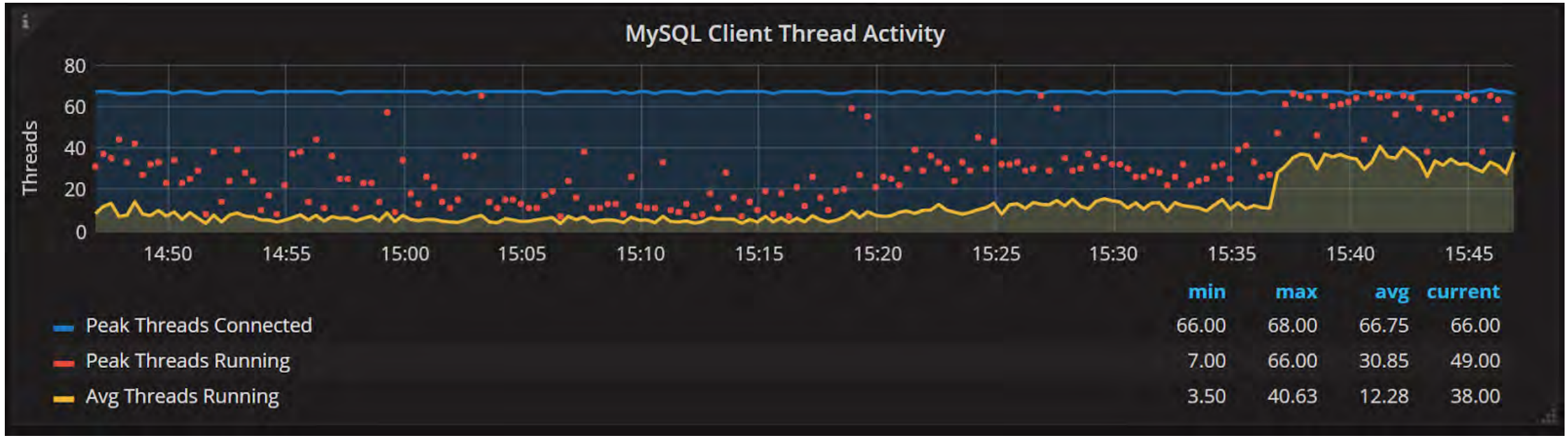
Average IO Size is down

- Large block writes to binlog and innodb transaction logs do not happen any more



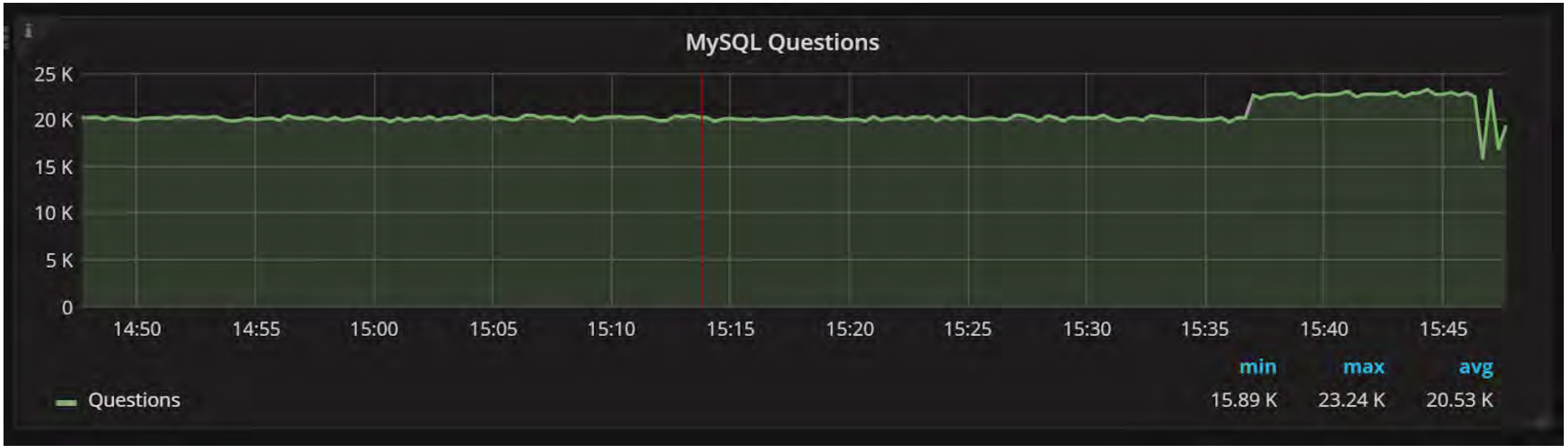
Number of Running Threads Impacted

- Need higher concurrency to be able to drive same number of queries/sec



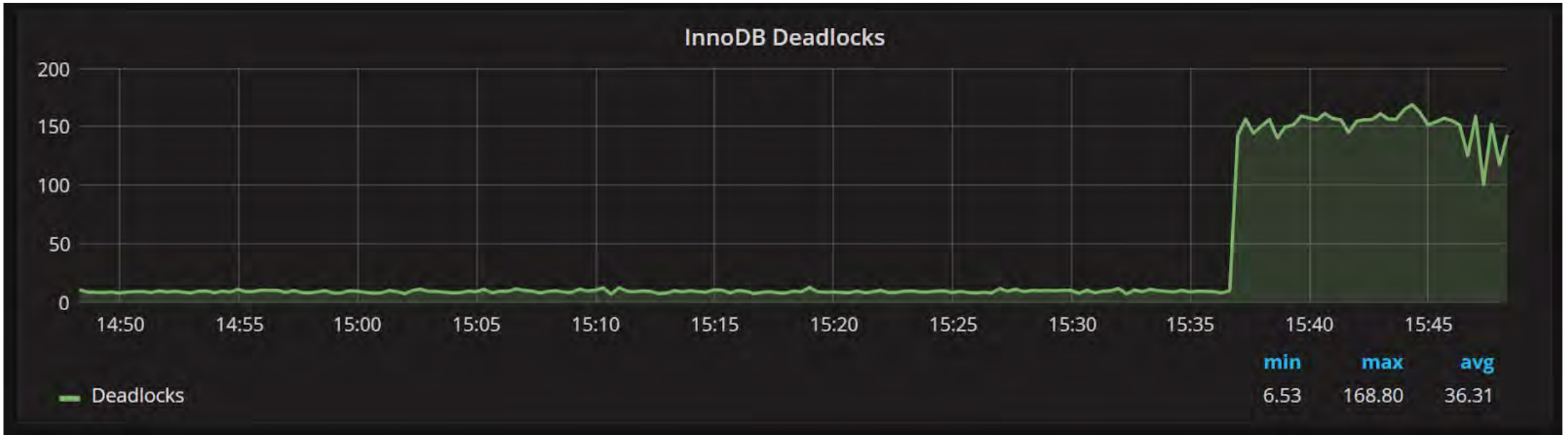
MySQL Questions

- Why does it increase with same inflow of transactions ?



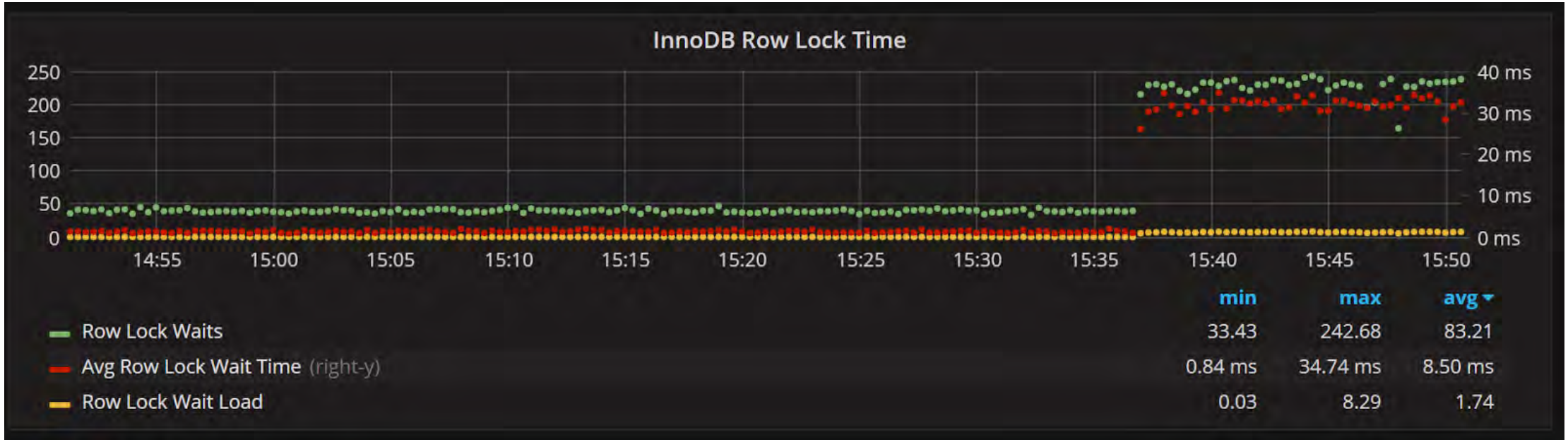
Because of Deadlocks

- Some transactions have to be retried due to deadlocks
- Your well designed system should behave the same

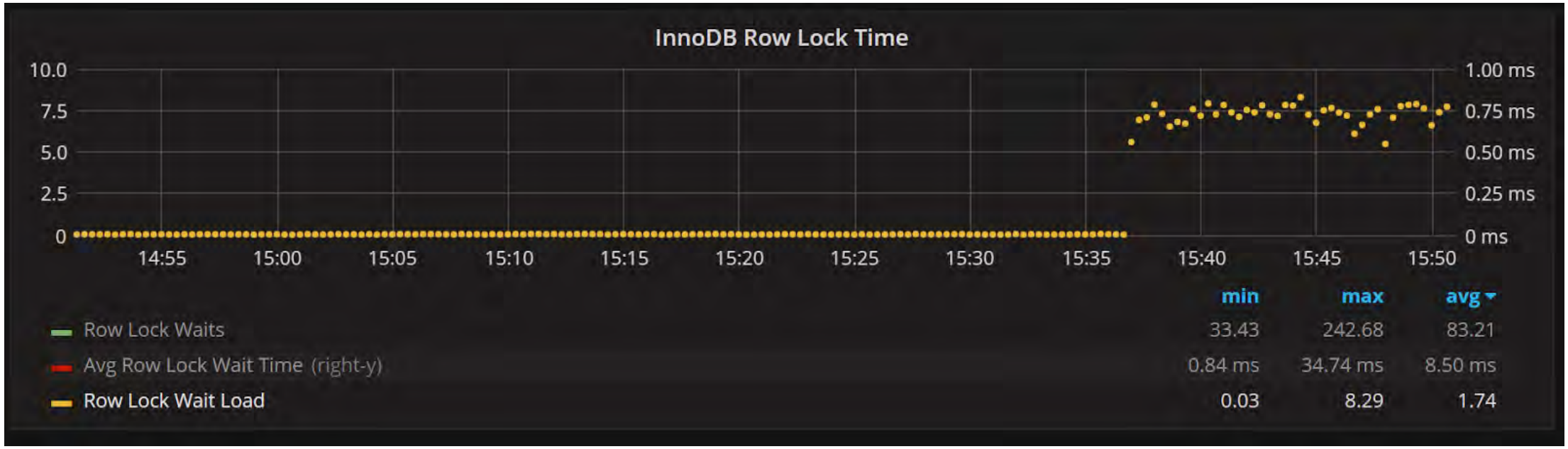


Higher Row Lock Time

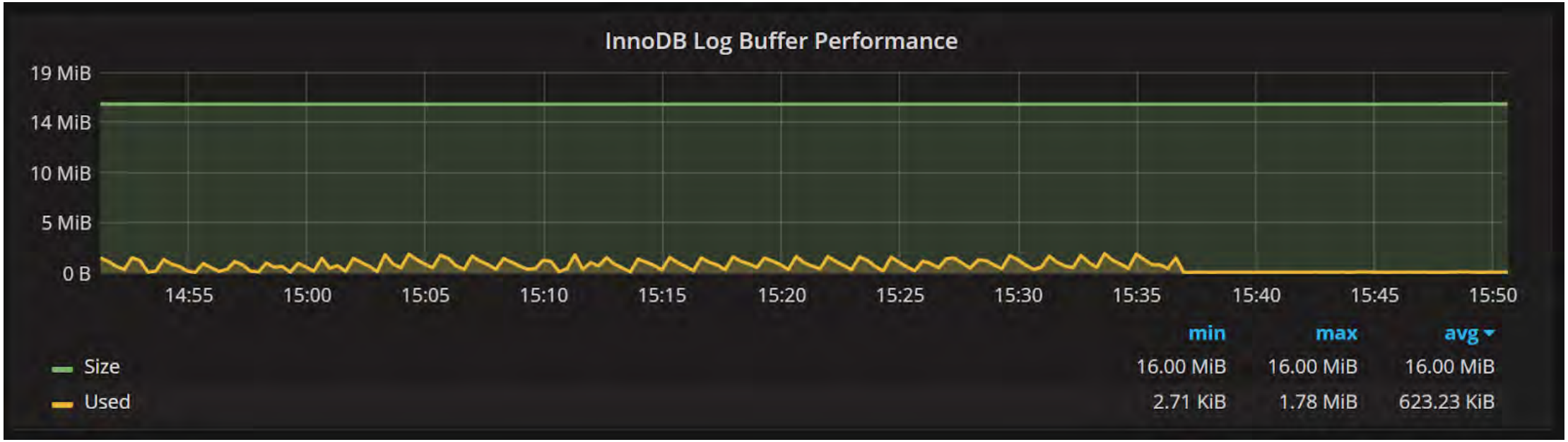
- Rows Locks can be only released after successful transaction commit
- Which now takes longer time due to number of fsync() calls



And Load Caused by Row Locks

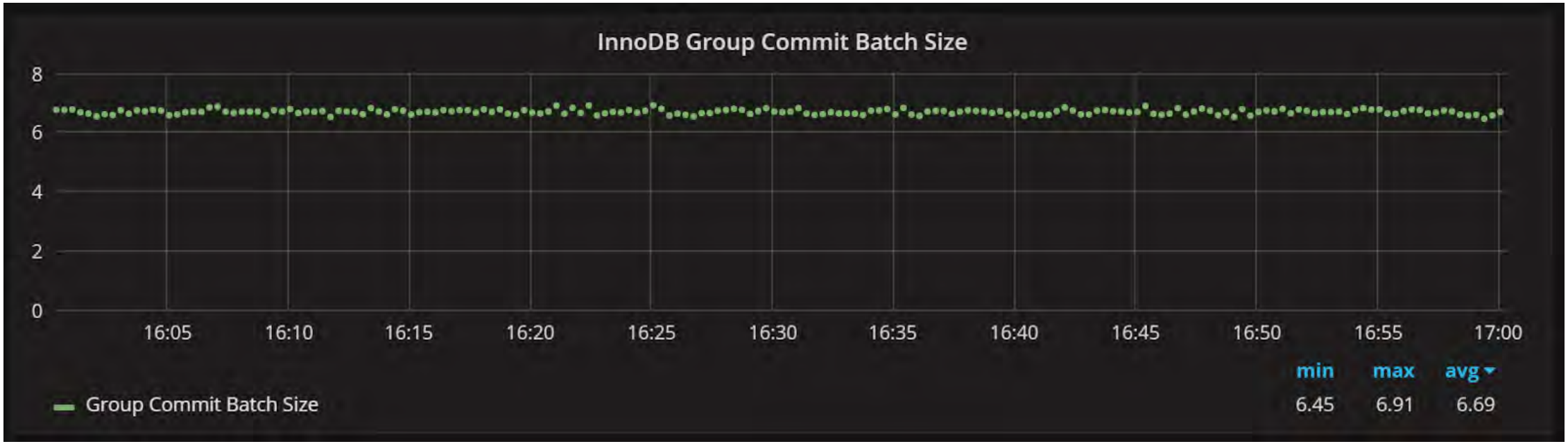


Log Buffer Used even less with durability on



Is Group Commit Working ?

- Do we relay on Group Commit for our workload



Top Queries Impacted

- Commit is now the highest load contributor

Top 10 of 34 Queries by % Grand Total Time (%GTT)

Search by Query Abstract, Fingerprint or ID

#	Query Abstract	ID	Load	Count	Latency
	TOTAL		35.61 (100.00%)	22.37 k QPS	80.54 m (100.00%)	1.59 ms avg
1	COMMIT	813...	17.64 (49.55%)	981.64 QPS	3.53 m (4.39%)	17.98 ms avg
2	SELECT sbtest	558...	4.34 (12.19%)	11.46 k QPS	41.25 m (51.21%)	378.91 µs avg
3	UPDATE sbtest	E96...	3.67 (10.31%)	1.08 k QPS	3.88 m (4.82%)	3.40 ms avg
4	DELETE sbtest	EA...	3.48 (9.76%)	1.03 k QPS	3.71 m (4.60%)	3.37 ms avg
5	UPDATE sbtest	D3...	3.46 (9.70%)	1.14 k QPS	4.10 m (5.09%)	3.03 ms avg
6	SELECT sbtest	737...	0.87 (2.45%)	1.14 k QPS	4.09 m (5.08%)	766.90 µs avg
7	SELECT sbtest	84...	0.68 (1.91%)	1.14 k QPS	4.10 m (5.09%)	596.05 µs avg
8	SELECT sbtest	382...	0.61 (1.71%)	1.13 k QPS	4.08 m (5.07%)	536.26 µs avg
9	SELECT sbtest	6EE...	0.60 (1.69%)	1.14 k QPS	4.09 m (5.08%)	527.89 µs avg
10	INSERT sbtest	F12...	0.21 (0.59%)	983.33 QPS	3.54 m (4.40%)	213.69 µs avg

Load next 10 queries

Changing Buffer Pool Size

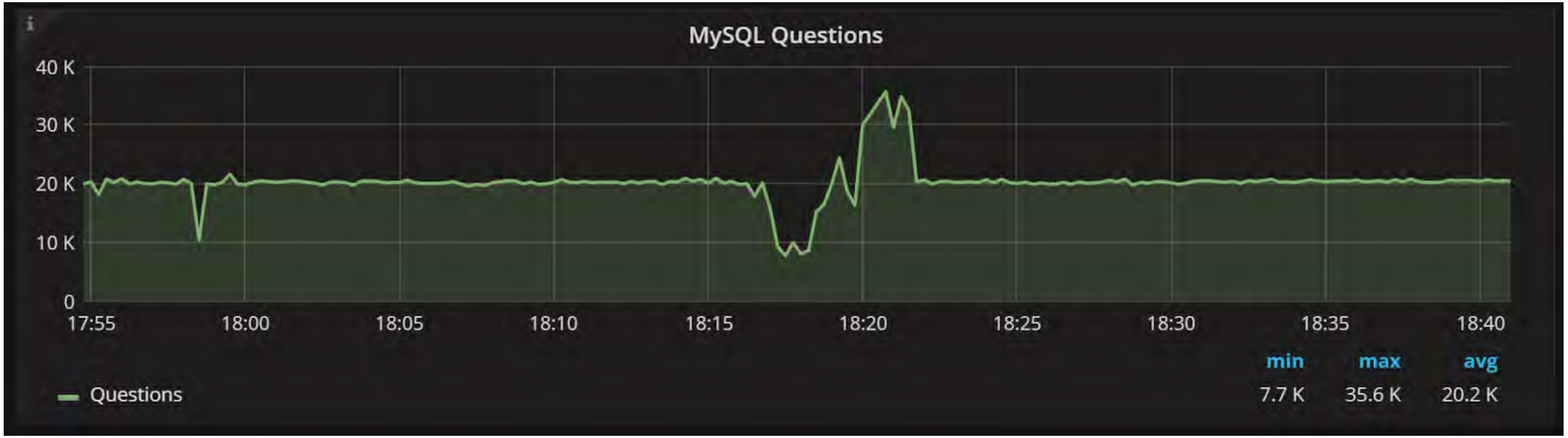
MySQL 5.7 Allows to change BP Online

- Changing buffer pool from 48GB to 4GB online

```
mysql> set global
innodb_buffer_pool_size=4096*1024*1024;
Query OK, 0 rows affected (0.00 sec)
```

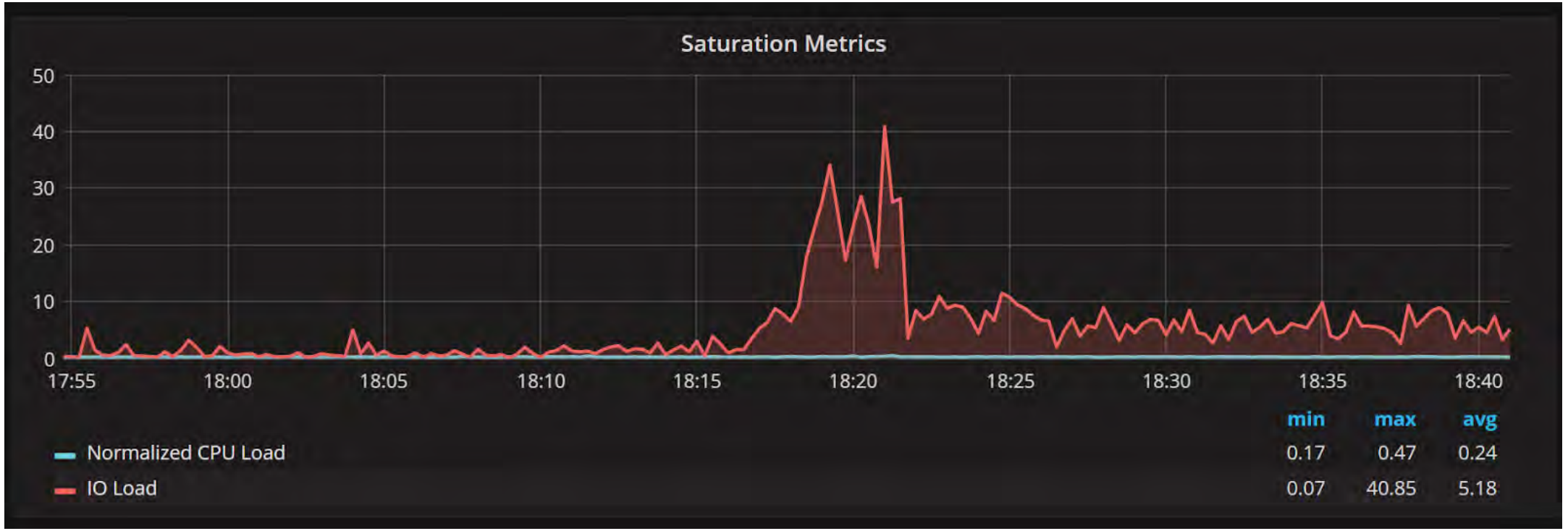
QPS Impact

- While resizing is ongoing capacity is limited – Queueing happens
- After resize completed backlog has to be worked off having higher number of queries



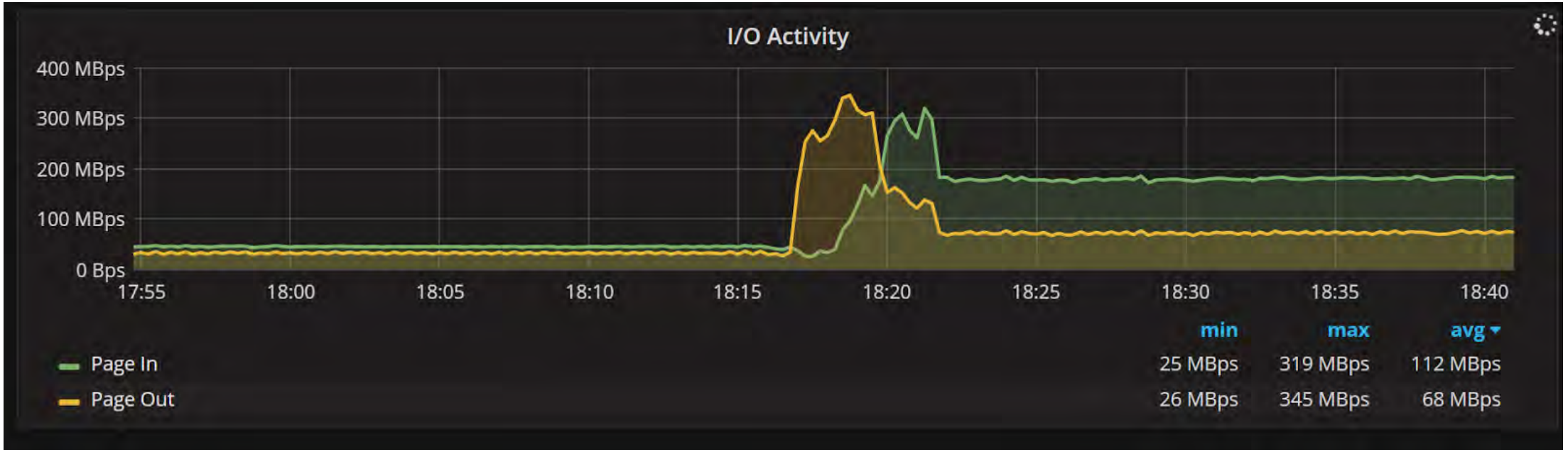
Saturation spike and when stabilizing on higher level

- Guess why the spike with lower QPS Level ?



Two IO Spikes

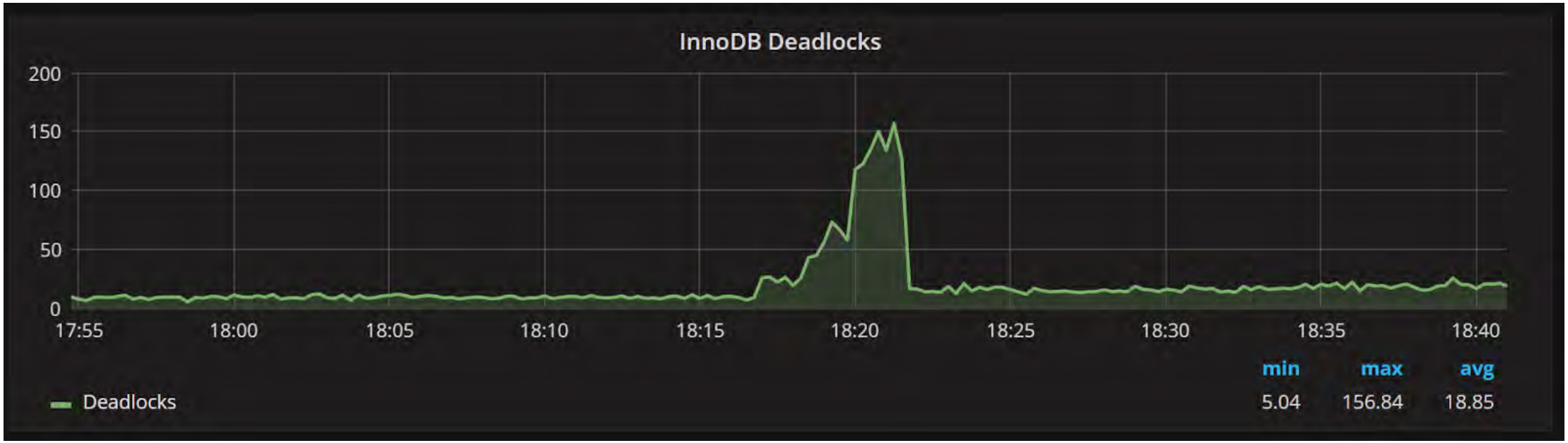
- First to Flush Dirty Pages
- Second to work off higher query rate



What is about Disk IO Latency ?

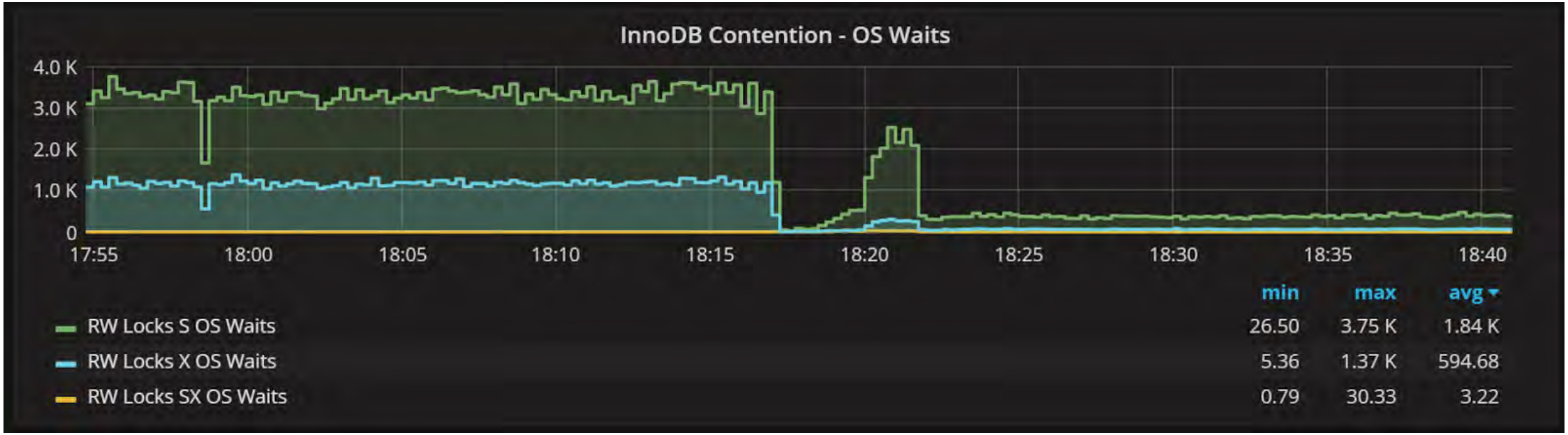
- Higher Number of IOPS does not always mean much higher latency

Longer Transactions = More Deadlocks



More IO Load Less Contention ?

- Unsure why this is the case
- Note not ALL contention is shown in those graphs



Now we see query 80% IO Bound

SELECT sbtest 558CAEF5F387E929

Metrics

Metrics	Rate/Sec	Sum	Per Query Stats
Query Count	9.75 k (per sec)	35.09 m 50.15% of total	
Query Time	5.63 load	5:37:50 38.73% of total	577.43 µs avg
Lock Time	0.23 (avg load)	0:13:39 28.03% of total 4.04% of query time	23.34 µs avg
Innodb IO Read Wait	4.48 (avg load)	4:29:03 46.96% of total 79.64% of query time	459.85 µs avg
Innodb Read Ops	4.47 k (per sec)	16.11 m 43.01% of total	0.00 avg
Innodb Read Bytes	73.31 MB (per sec)	263.91 GB 43.01% of total 16.38 KB avg io size	7.52 KB avg
Innodb Distinct Pages	-	-	3.00 avg
Rows Sent	9.75 k (per sec)	35.09 m 3.21% of total	0.98 avg
Bytes Sent	1.85 MB (per sec)	6.67 GB 4.73% of total 190.00 Bytes bytes/row	189.98 Bytes avg
Rows Examined	9.75 k (per sec)	35.09 m 1.40% of total 1.00 per row sent	0.98 avg

Summary

Can get a lot of Insights in MySQL Performance with PMM

Great tool to have when you're challenged troubleshoot MySQL

A lot of insights during benchmarking and evaluation

SAVE THE DATE

OPEN SOURCE DATABASE CONFERENCE

PERCONA LIVE EUROPE FRANKFURT

5-7 November 2018
Radisson Blu
Frankfurt, Germany

Percona to Support PostgreSQL

Thank You!