Query Analyzer

SRE

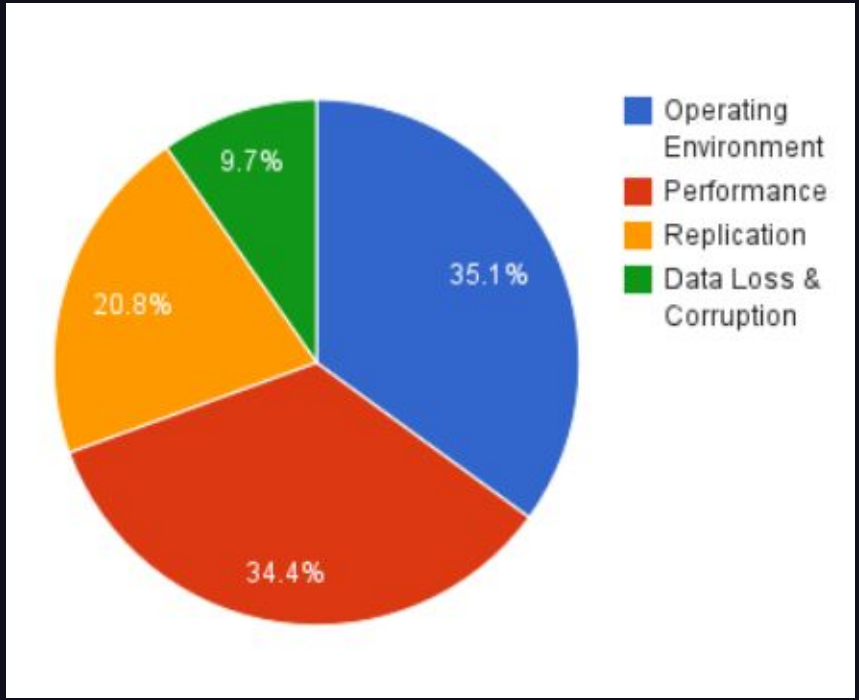
Karthik Appigatla

Basavaiah Thambara

Query Analyzer

- MySQL at LinkedIn, challenges
- Why we built Query Analyzer
- How Query Analyzer works
- How Query Analyzer helped us
- Future

Why do we need a Query Analyzer?



Categories of Database Downtime

Source: Percona White Paper - https://www.percona.com/files/white-papers/causes-of-downtime-in-mysql.pdf

Why do we need a Query Analyzer?

Item	Incidents	% of Total
SQL	20	37.8%
Schema and Indexing	8	15.1%
InnoDB	8	15.1%
Configuration	7	13.2%
Idle Transactions	4	7.6%
Other	4	7.6%
Query Cache	2	3.8%

Causes of bad performance

Source: Percona White Paper - https://www.percona.com/files/white-papers/causes-of-downtime-in-mysql.pdf

Existing approaches

- Slow Query Log
  - Adds around 35-40% overhead
  - Analyzing slow query logs is another challenge

Existing approaches

- Performance Schema
  - Requires MySQL restart to enable/disable
  - Adds around 15-20% overhead
  - Complex to analyze

Existing approaches

- Application side monitoring
  - Development effort
  - Does not give clear picture

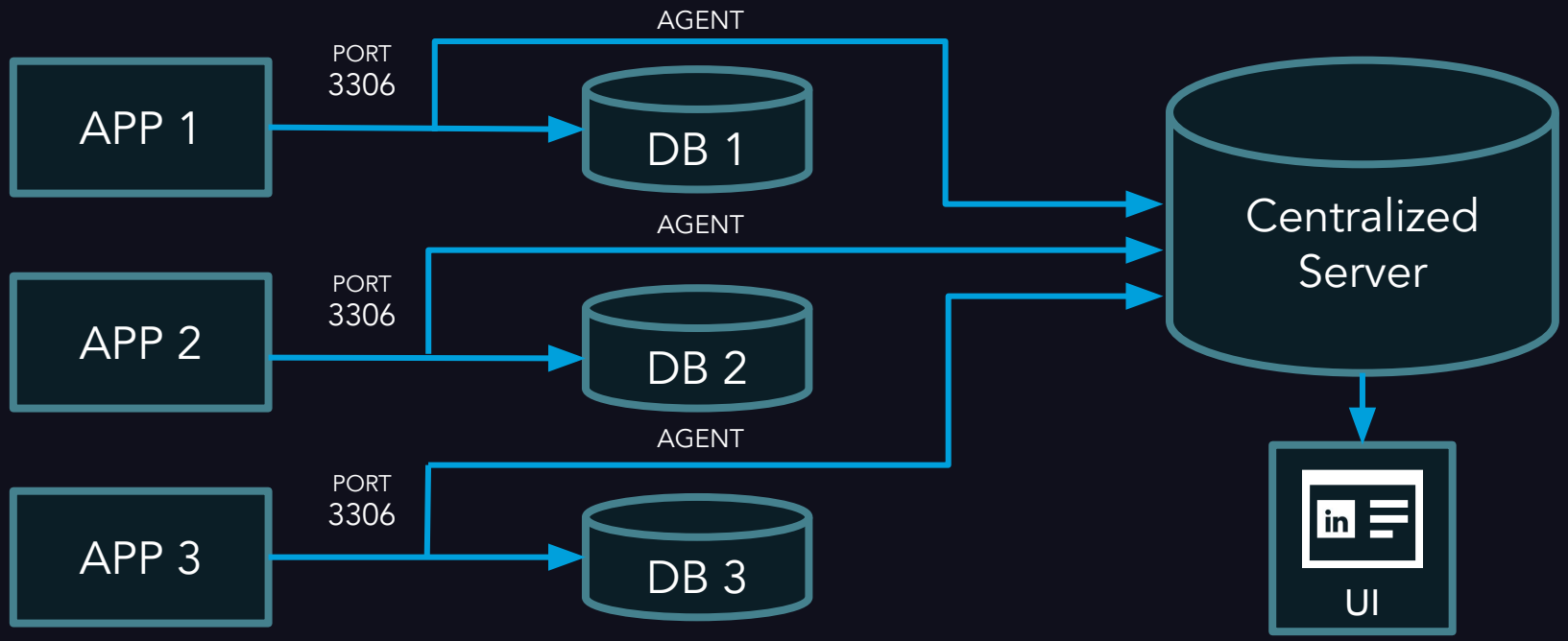
Pain Points

- Pain Points
  - Lack of consolidated view for queries
  - Lack of tools to identify problematic queries
  - Lack of Historic Query trends

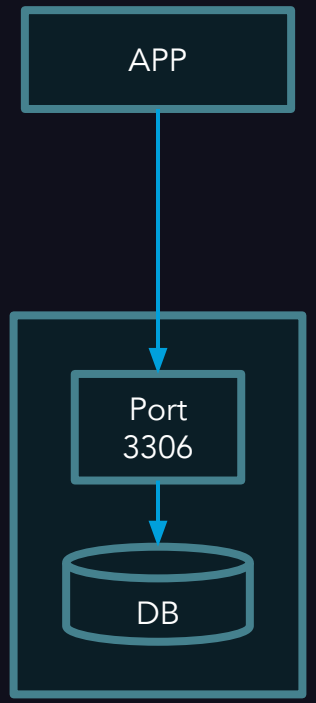
Approach

Capturing packets at network layer

Architecture

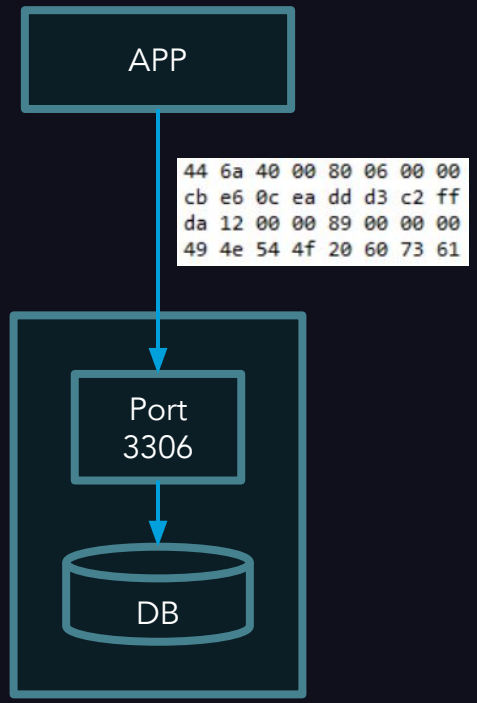


How Agent Works



Database Server

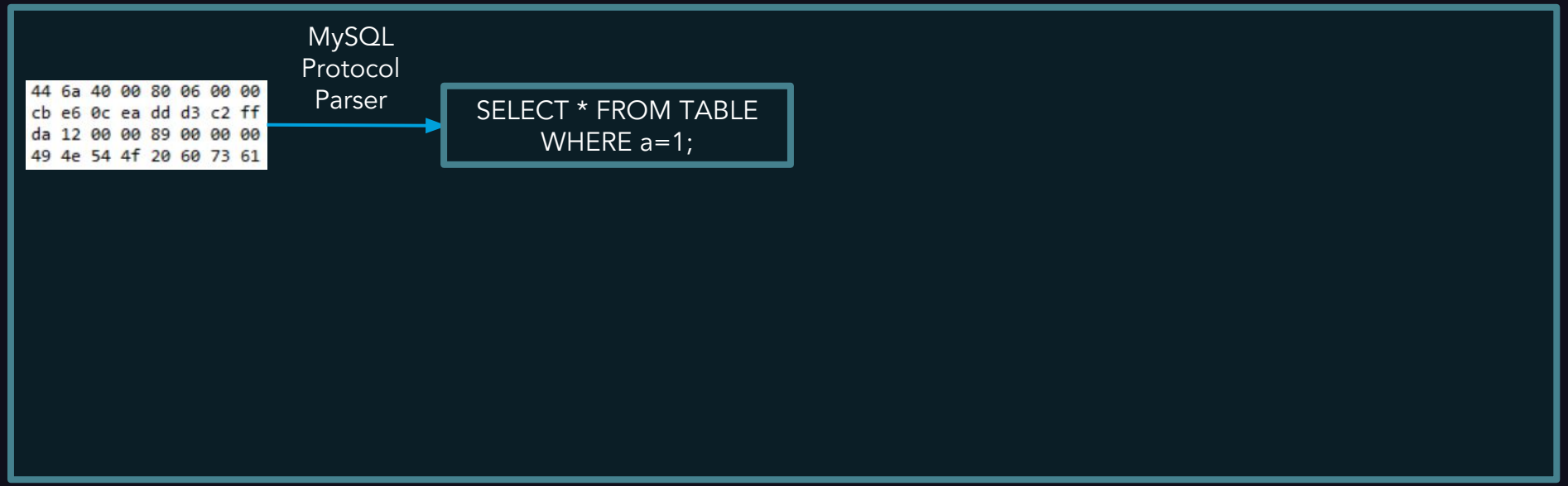
How Agent Works



Database Server

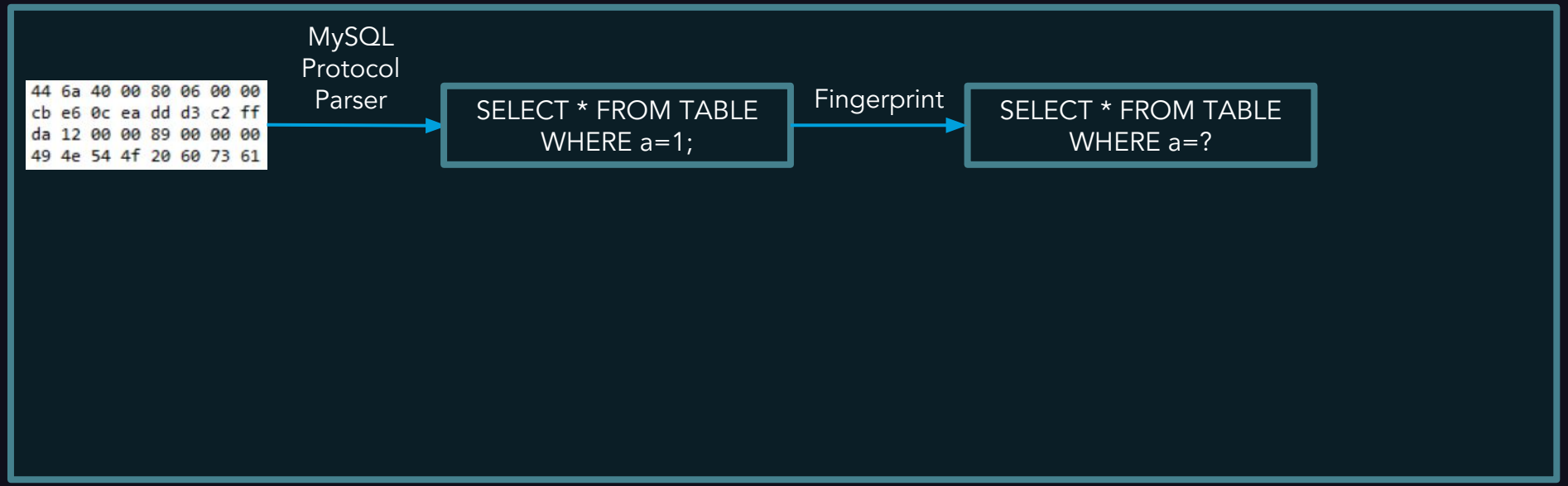
How Agent Works

How Agent Works



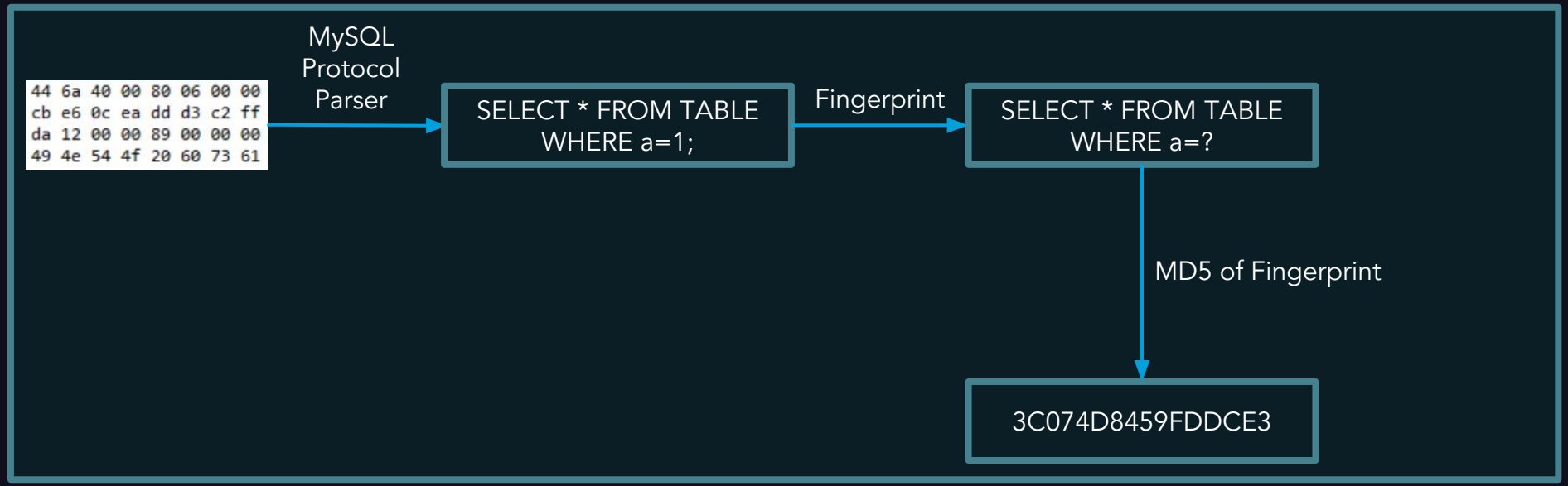
Agent

How Agent Works



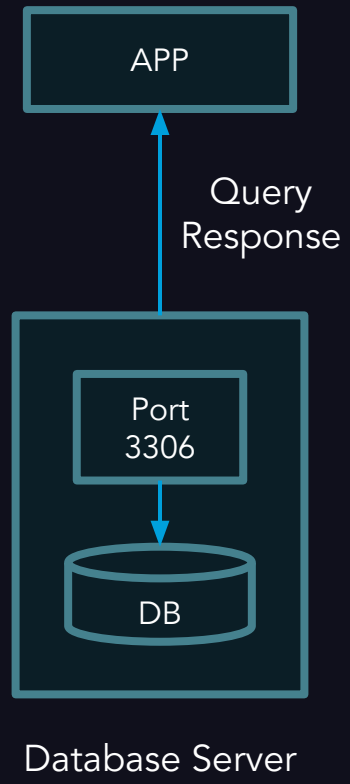
Agent

How Agent Works



Agent

How Agent Works



Anonymization Examples

Query: SELECT * FROM table WHERE value1 = 'abc’

Fingerprint: SELECT * FROM table WHERE value1 = '?'

Query: SELECT * FROM table WHERE value1 = 'abc' AND value2 = 430

Fingerprint: SELECT * FROM table WHERE value1 = '?' AND value2 = ?

Query: SELECT * FROM table WHERE VALUES IN (1,2,3)

Fingerprint: SELECT * FROM table WHERE VALUES IN (?+)

How Agent computes

Checksum (KEY)	Query Time	Count	user	db
3C074D8459FDDCE3	6ms (1ms+2ms+3ms)	3	app1	db1
B414D9DF79E10545	9s (1s+3s+4s+1s)	4	app2	db2
791C5370A1021F19	12ms (5ms+7ms)	2	app3	db3

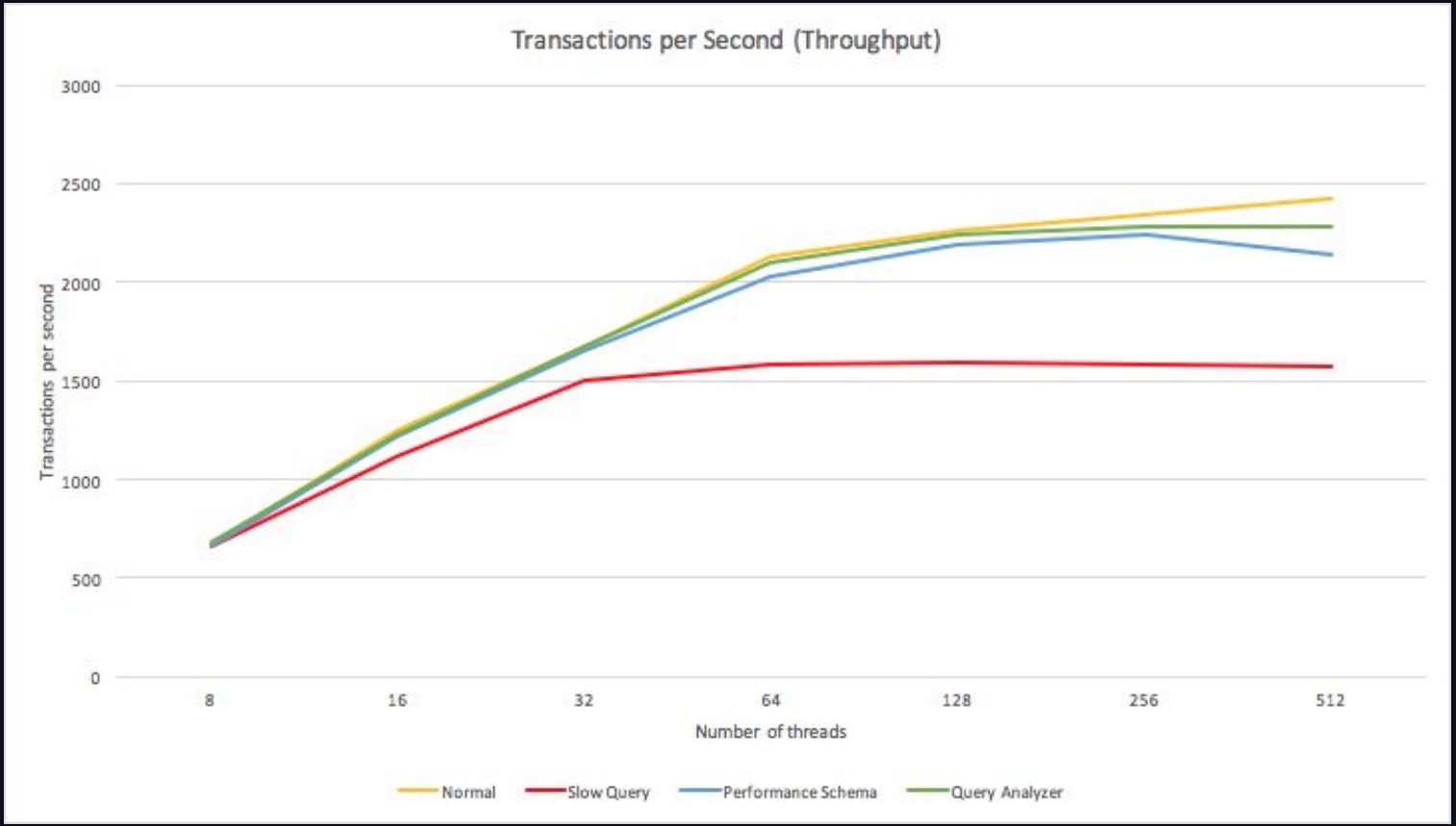
Query Metadata

Checksum	Fingerprint	First Seen	Sample at Max time	Min Time	Max Time
3C074D8459FDDCE3	SELECT * FROM T1 WHERE a>?	1 month	SELECT * FROM T1 WHERE a>0	1ms	3ms
B414D9DF79E10545	SELECT * FROM T2 WHERE b=?	1 day	SELECT * FROM T2 WHERE b=430	1s	5s
791C5370A1021F19	SELECT * FROM T3 WHERE c<?	1 hour	SELECT * FROM T3 WHERE c<3	5ms	7ms

How does it all work?

1. Captures packets on port 3306
2. Decodes packets into queries
3. Anonymizes the query
4. Measures the metrics - query time and count
5. Locally aggregates configurable time (approximately 60 secs)
6. Pushes the aggregated data to a centralized server (using SSL)

Query Analyzer Overhead



Query Analyzer Overhead

- Negligible Memory Overhead
- Negligible Network Overhead
- Negligible CPU Overhead up to 128 threads
- Around 3% CPU Overhead for > 128 threads

1y | 6m | 1mon | 1week | 24h | 1h

[redacted].linkedin.com | 2017-08-22 04:05 | 2017-08-22 06:05

Submit

Number of distinct queries 574

Rank	Reviewed	Query	First Seen	Count%	QPS	Avg Time	Cumulative Time	Query Load%
1	✓	select alert.id as alert_id	2 months ago	78.38(7.647M)	5.573K	0.328ms	41m 50s	32.61
2		select * from (select @row := @row + ? as rownum	1 day ago	0.00(332)	0.24	4.675s	25m 51s	20.16
3	⚠	select postcommit_event.id as postcommit_event_id	2 months ago	0.00(24)	0.02	54.072s	21m 37s	16.86
4		select build_event.id as build_event_id	2 months ago	0.00(101)	0.07	7.606s	12m 48s	9.98
5		select precommits.id as precommits_id	2 months ago	0.07(7.205K)	5.25	60.935ms	7m 19s	5.7
6		select deployable_name	2 months ago	0.01(707)	0.52	473.148ms	5m 34s	4.34
7		select postcommit_event.id as postcommit_event_id	2 months ago	0.00(2)	0.08	55.369s	1m 50s	1.44
8		select alert.id as alert_id	2 months ago	5.11(498.3K)	363.21	0.199ms	1m 38s	1.29
9		select alert.filter as alert_filter	2 months ago	1.71(166.9K)	121.68	0.446ms	1m 14s	0.97
10		select `cfg2idx_a`.`iv2_contributions`.`name`	1 day ago	0.01(1.227K)	42.31	44.600ms	54.724s	0.71

« 1 2 3 … 58 »

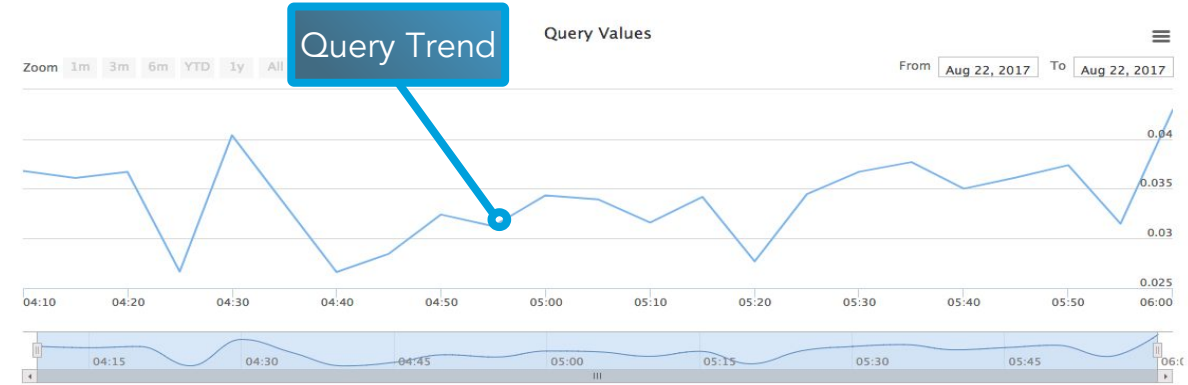
10 25 50 100

Query Info

Checksum	2C76986F52A6B910
Query	SELECT target_state.id AS [redacted]
First Seen	2 weeks ago (2017-08-07 21:31:44)
Last Seen	1 day ago (2017-08-22 06:00:01)
Max Time	52.441ms (2017-08-08 23:12:59)
Min Time	0.005ms (2017-08-08 23:12:59)

Metrics during selected time frame

Metric	Count%	Average Time	Cumulative Time	Load%	Min Time	Max Time
Query_time	0.97	33.953ms	7m 36s	29.52	26.615ms	42.949ms



JIRA:MySQL-225 - Query 2C76986F52A6B910 not using index

Needs Attention | Mark as Reviewed

Query Load Explained

	Time	Count	Load
Query 1	2 sec	100	2 * 100 = 200
Query 2	0.1 ms	10M	0.1m * 10M = 1000
Query 3	10 ms	1M	10m * 1M = 10000

	Load		Percentage Load
Query 1	200	200/11200 * 100	1.78%
Query 2	1000	1000/11200 * 100	8.93%
Query 3	10000	10000/11200 * 100	89.29%

How can you leverage this?

Data SREs

Saves time spent debugging queries

Developers

Visually check the load of their query

Security

Capture access to sensitive information

Cost Savings

Optimized hardware usage & estimation server capacity

Thank You!