ProxySQL Technology Day

Ghent, 03.10.2019

What do we call a “High Performance” today?

Introduction to a modern infrastructure

Vlad Fedorkov

ProxySQL Tech Day, Ghent
03.10.2019

ProxySQL Technology Day: a house keeping minute

- Keep doors open
- Grab your favorite drink and ask questions
- You can catch me, Jesmar or Stacy with any questions
- We'll have a 25 minutes break at 18:45 for snacks
  - And something else later
- Please feel free to share photos and twits with #ProxySQL hashtag
  - Also ProxySQL team is available for selfies
- A side note: talks are being recorded

ProxySQL

ProxySQL Tech Day specials:

We are giving out t-shirts for best questions.

We will also give out FREE 2-hours consulting gig for ProxySQL from the business cards.

And we've just released ProxySQL 2.0.7! Go download it for free! ☺

About me

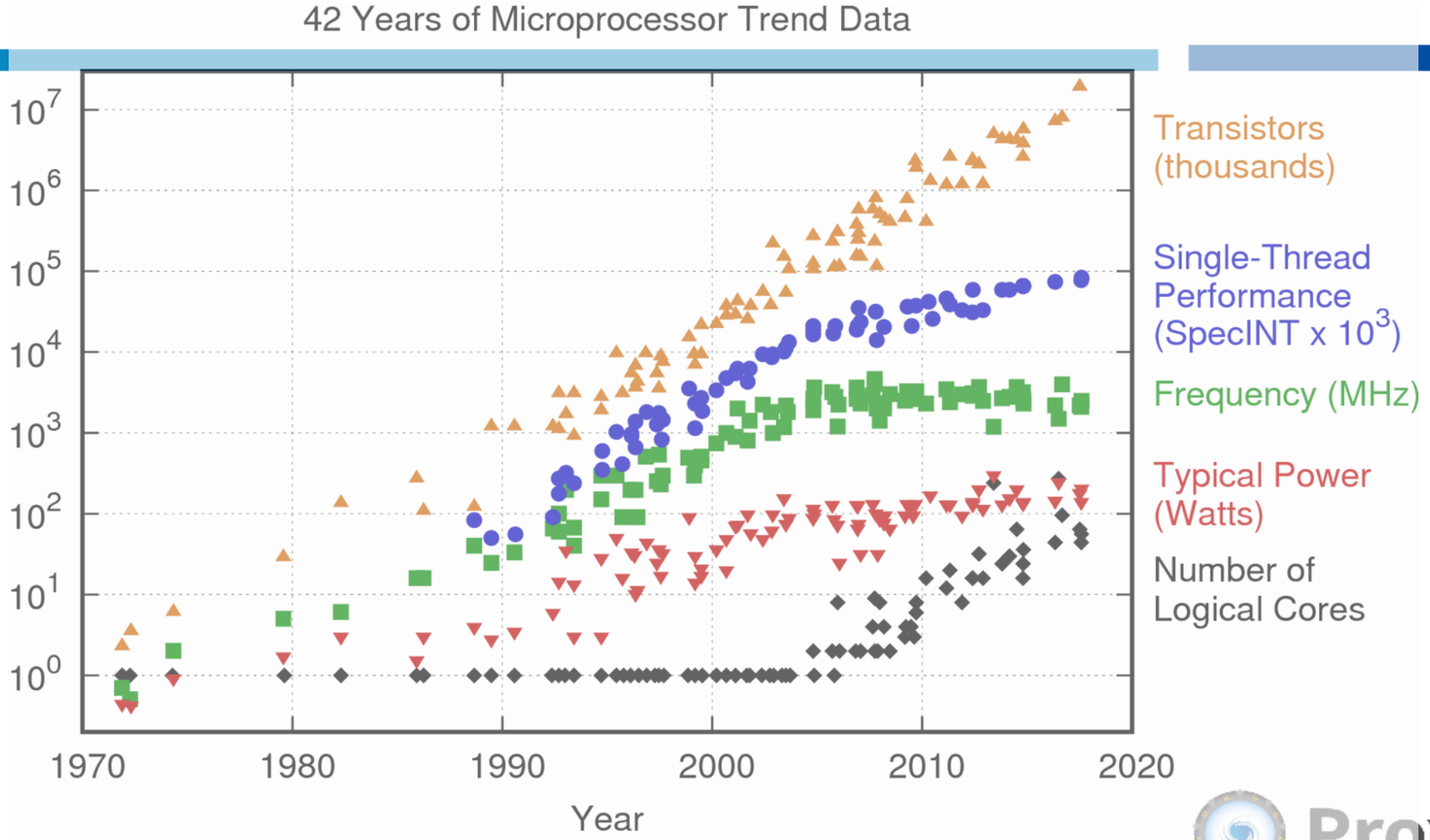
- Working with MySQL
  - Last 19+ years
  - Still like it
- Helping companies to survive growth pain and traffic spikes
  - Last 10+ years
- Working for ProxySQL for last 3 years
  - And enjoy it a lot
- Happy to learn from others and share my own experience

ProxySQL

Goals for DBAs 20 years ago and today

- In the year 2000 I feet myself like a super star installing MySQL 3.23.56 on the single box
- Dealing with 100k+ QPS today I don’t feel even closer
- What’s changed?

ProxySQL

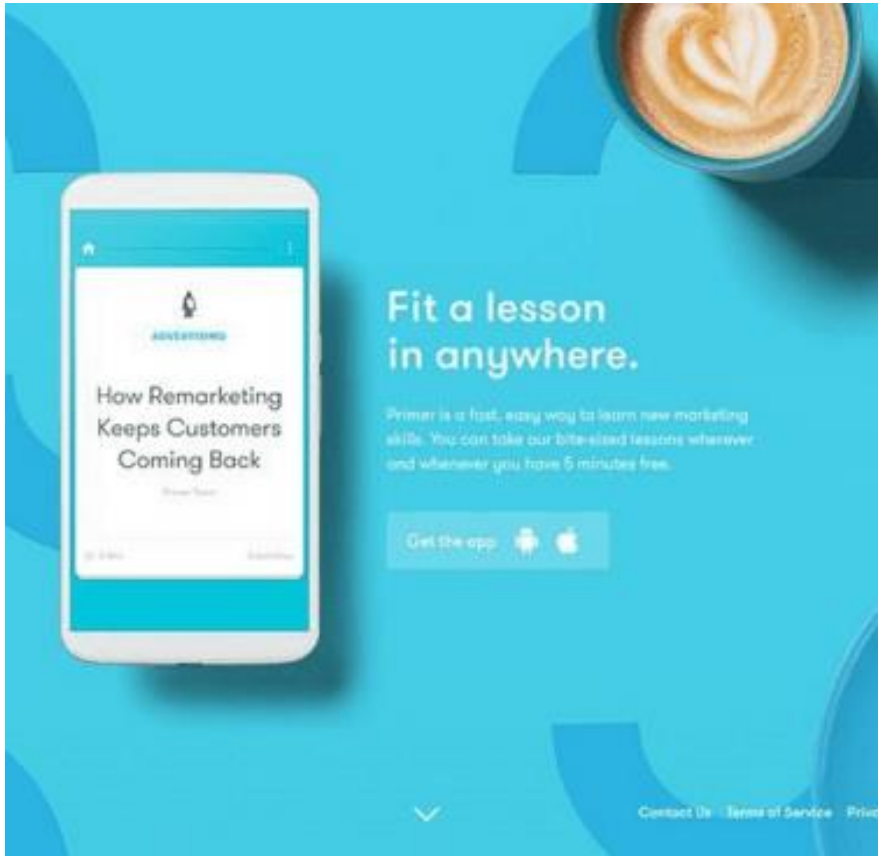

42 Years of Microprocessor Trend Data
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1970
1980
1990
2000
2010
2020
Year
Transistors (thousands)
Single-Thread Performance (SpecINT x 10^3)
Frequency (MHz)
Typical Power (Watts)
Number of Logical Cores
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

ProxySQL

Look has changed

1998…2000

2015…2019



ProxySQL

But not only a look has changed

1. Microsoft 904,860
2. Apple Inc. 895,670
3. Amazon.com 874,710
4. Alphabet Inc. 818,160
5. Berkshire Hathaway 493,750
6. Facebook 475,730
7. Alibaba Group 472,940
8. Tencent 440,980
9. Johnson & Johnson 372,230
10. ExxonMobil 342,170

… Ford motors 38,221

ProxySQL

Online bookstore @ year 2001

- Actual store you can walk in
- On the website
  - Catalogue
  - Item page
  - Shopping cart
  - User page (user info & orders)
- Order confirmation via the phone
- Single server: if it stuck, we'll just reboot it.

Online bookstore today (year 2019)

- Most of the sales are from website
  - Offline store may not even exist
- Payments made online
- The website
  - No one even counts how many pages are there
  - We count in services
  - Integrated backoffice
- 24/7 availability is a must
  - 5% drop of user requests considered as a major outage

ProxySQL

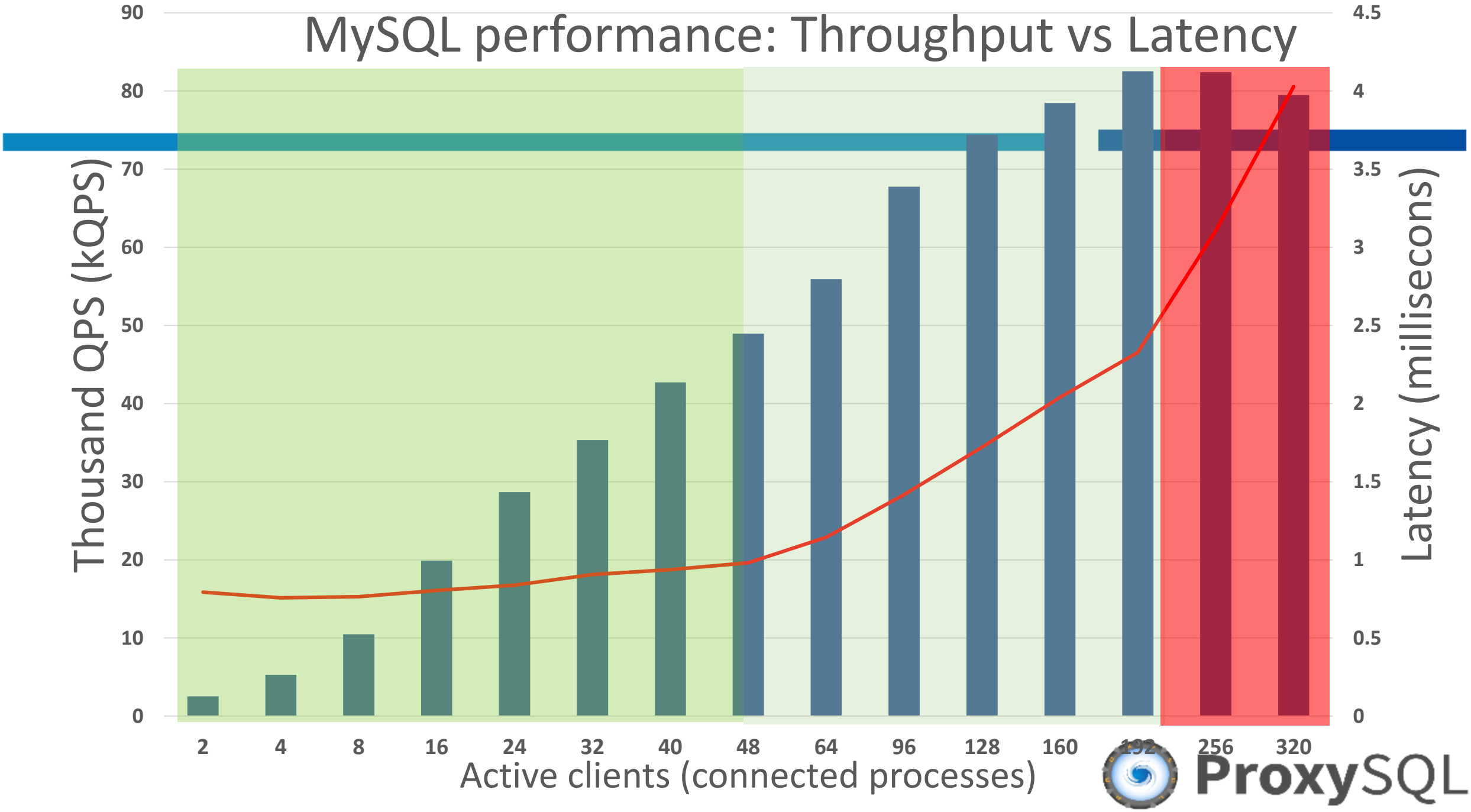
Infrastructure goals has changed

- Single box can't handle the load anymore
  - Even if it's 128 core box
- Large amount of boxes means large amount of outages
  - Both hardware and software
  - Single point of failure going to be a trouble
- "Capacity" goes beyond the "Performance" today
  - Do we have enough capacity to serve existing customers?
  - How many customers can we serve with the existing hardware?
  - How many customers can we serve with the current architecture?
- "Capacity" is how many boxes do we need to have to serve traffic with acceptable latency.

ProxySQL

Capacity in MySQL

- One thread per connection to MySQL server
- Every query is served by a single thread
- Every query is a combination of data processing (CPU) and data reading/writing (IO)
- IO is relatively slow
  - so some data is cached in the memory to reduce number of IO requests
- In general capacity is always a combination of latency and throughput


MySQL performance: Throughput vs Latency
Thousand QPS (kQPS)
Latency (millisecons)
Active clients (connected processes)
0
10
20
30
40
50
60
70
80
90
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
2
4
8
16
24
32
40
48
64
96
128
160
192
256
320
ProxySQL

Where latency is coming from?

- Some steps to mention
  - Forming SQL query with ORM library
  - MySQL client API call
  - Network driver calls in OS kernel
  - Network transmission
  - Packet reception on MySQL side
  - Query parsing
  - Query execution plan
  - MySQL API calls to storage engine
  - Necessary IO requests to the disks
    - Yet another network calls in case of attached disks
  - Dataset processing on the CPU side
  - Dataset transmission to the client
  - Data post-processing on the client sides

ProxySQL

What's the problem with the latency?

- Query time is sum of latency of all components + pure execution time
- Lower latency means higher amount of served requests
- Faster response time
  - For some applications it's a key requirement
- Cost of lower latency is increasing exponentially
  - Faster hardware may be far more expensive
- So we have to scale out to keep cost manageable

Scalability

- Be able to spread traffic across nodes
- Be able to detect failures and overloads
  - add and remove nodes transparently for the application
- Be able to understand database traffic
- Be able to manage it
- All the above is not supported by “classic” MySQL
  - You have to code this logic yourself
  - Or install additional tools

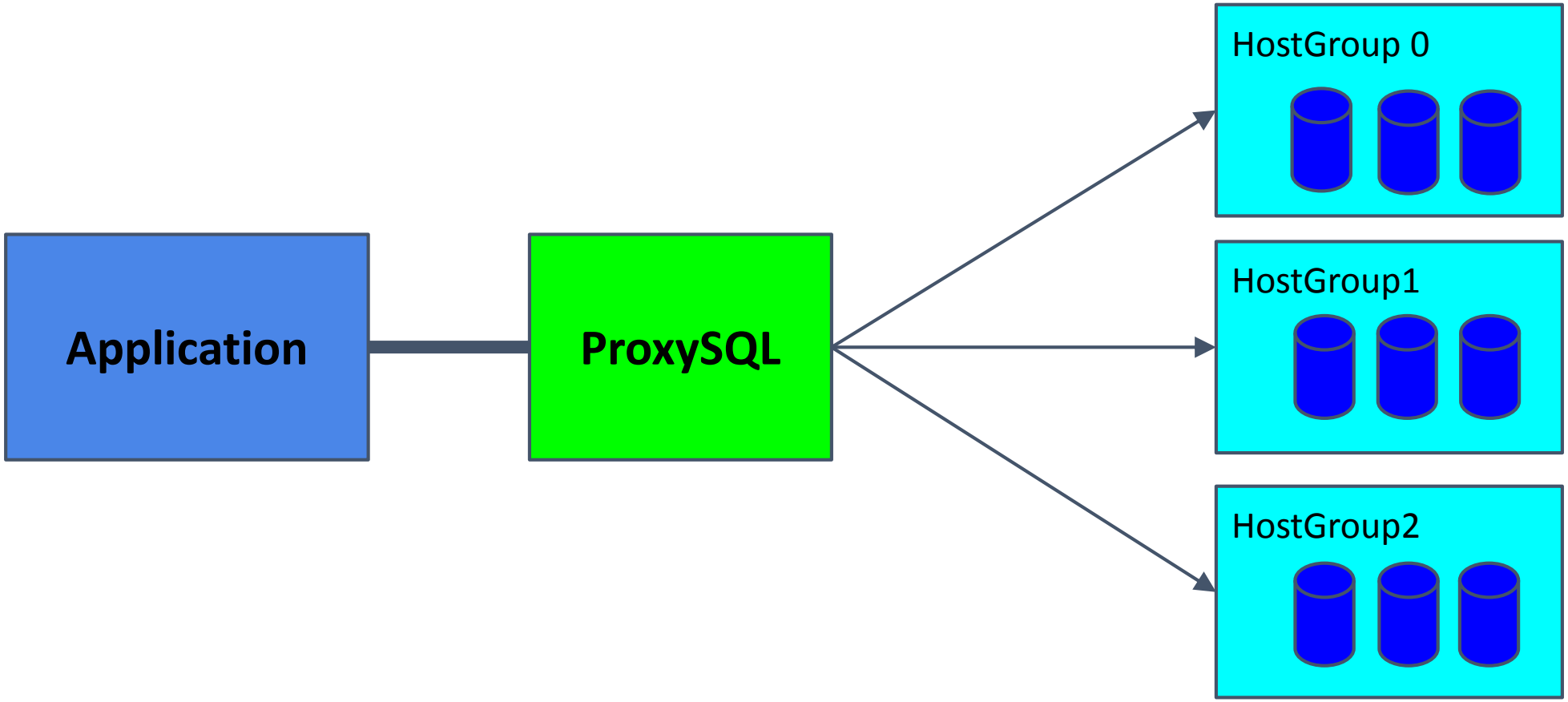
ProxySQL

Load balancers are keys for today's infrastructure

- We had a lot of it for HTTP
- It's complicated to proxy the database traffic
- Where are number of them on the market:
  HAProxy, MaxScale, MySQL Router, Vitesse, etc
- And apparently ProxySQL

ProxySQL

Load balancing for MySQL: Basic design



ProxySQL

ProxySQL

- ProxySQL understand SQL language
  - Unlike layer 4 ISO/OSI proxies working on transport level
- It knows everything about query it’s processing, the state of connection, authorization and results.
- It uses internal connection pool with connection multiplexing to re-use existing connections.
- It’s able to route queries based on various filters:
  - By user, by database (schema name) and by query itself
- You’ll know more from the next speaker ☺

Questions!

ProxySQL