Database Performance
What Really Matters?

Peter Zaitsev,
CEO, Percona
OSCON 2013
Portland, OR
July 25, 2013
About Me & Percona

• Founder and CEO of Percona
• Products and Services for MySQL Ecosystem
  • Support, Consulting, RemoteDBA etc
• Important Focus: Have been helping customers with Performance Problems
  • MySQL and Beyond
About the Presentation

• Share some valuable background and Theory behind Performance Optimization
• Look at Load Testing and Benchmarking
  • How to apply them in practice
  • How to look for right things
    • “What really matters”
Some background and Theory
Databases do not matter

• Users, Business does not care about databases
  • Really!
What Matters?

• Application Performance
  • As seen by your Users, Business
• Response time for their transactions with the system
What Impacts Performance?

- Changes to the transaction volume
  - More users
- Changes to transaction mix/type
  - Do not forget non-business transactions
- Data Volume
- Changes to application features
- Environment Changes
  - RAID running slow to degraded mode
Database Performance – Part of The Picture

• Database Performance – Part of the Picture
• Look at complete Application not Database alone
• Database may be the bottleneck
  • Or it might be something else
Bottleneck Solutions

• Even when Database is the Bottleneck solutions might lay elsewhere
Application Architecture

- Caching
- Queuing
- Concurrency Management
- Data Distribution (sometimes)
- Use of Data Replicas (sometimes)
- Async and Parallel Processing (sometimes)
Application Defines

• Schema and Queries
• Same Application Operation can be done using different Schema and Queries
• Some Approaches
  • Normalization/Denormalization
  • Summary Tables
  • Indexing
• Using Special Purpose Systems
  • Column Store; Full Text Search; Massively Parallel Processing; Graph etc
The Rule for Query Optimization

Do not look just optimizing the query, look at the larger picture how to optimize the whole operation.
Learning from the Industry

- Technology has a lot to learn from different industries
- Application Request processing is quite similar to manufacturing
  - Request processing takes multiple stages
  - Having input and output data
  - Utilizing Resources in the meanwhile
    - Resources on Service Level
    - Resources on Physical Level
Lessons from Manufacturing

• “The Goal” by Dr. Eliyahu M. Goldratt
• “Identify And Protect Constraint”
  • Bottleneck
• Minimize Work in Progress
Lessons from Call Center

• Queuing Theory
  • Response Time = Wait Time + Service Time
  • Requests come as “Random Arrivals”
  • The high utilization means long Wait time
  • And poor response time
The Hockey Stick

- High Utilization yields disproportional increase in response time
Throughput

• Theoretical Throughput assumes uniform arrivals
  • Or unlimited queue size
• Will not be reached; resource will stay idle some time
• Do not drive for complete resource usage
  • 100% CPU usage
  • Even at what you consider peak load
Database Performance Testing
Benchmarks

• Type of Benchmark
  • Pure Database
    • Easy to setup. Might be misleading
  • Complete System
    • Much Harder to set up. More valuable results

• Workload and Data
  • Simulated (are we getting it right?)
  • Replaying real world workload
    • How close we can emulate it?
Synthetic Benchmarks

- Most of published benchmark results are synthetic
  - Sysbench, TPC-C, LinkBench etc
  - This is what gives you comparable result
- Beware: These results are highly irrelevant
  - Your application is different
How are Benchmarks set up

• Most typical
  • Fixed concurrency “number of connections”
  • Send requests in as fast as possible
• In Real World
  • “Injection Rate” – amount users on the system
  • Concurrency is dynamic
    • More connections may be created as needed
  • “Think Time” between issuing queries
Results from Benchmarks

• Throughput is reported
  • Transactions per minute, per second
• What users care?
  • Response time, stable response time
• You can’t measure response time when you try to saturate your system fully
  • Issuing requests as fast as possible
Response Time

- Saturated Load – Higher Concurrency – Higher Response Time
Time Stability

• Those graphs correspond to about same average throughput
• Which is better?
• How would you do capacity planning?
Helpful Tools

- Sysbench has “Injection type benchmark”
  - [http://bit.ly/x4mZ0d](http://bit.ly/x4mZ0d)
- Percona Playback
  - Look at the log file, reissue queries at the intervals and threads as it happened in real run
Concurrency

- There is always optimal concurrency from "throughput" point of view
- WIP is expensive
Concurrence and Response time

• Though Experiment
  • 1CPU; 100 tasks requiring 1 sec of CPU time each
  • Run them concurrently?
  • Queue them and run one after another?
Restricting Concurrency

- Sysbench “Injection Benchmark”
- Setting the cap on amount of threads
Side Load

• Almost no one does Benchmrks/Load Testing with side load
  • Can cause serve issues
• Your Batch Jobs ?
• Backups ?
• RAID background consistency check ?
Real Performance Issues

- Persistent Problems
  - Are easy. Well caught by conventional load testing
  - Not enough resources; bad query
- Transient Problems
  - Something happens; System performs poorly for 10 seconds; few minutes
  - Very typical hard to catch
Concurrency is often the cause
Real World

• You will not catch all Real World issues even with best benchmark/load testing
• Proper monitoring is a must
• Look at the Application Response time over time
• Gather related system metrics for analyses
  • Database included but not just database
Do not focus on average

• Look at 1%, 5% of worst transactions
  • Some of your users will be impacted before the problem is there for average user

• What causes these to be slow?
  • The answer can be different
    • Disk IO; Database locks etc;
Drill Down

• Can you Drill down from your Application Transaction to the other subsystem
  • Why specific request was slow and what made it slow?
• NewRelic, AppDynamics are some of such tools available
Summary

• Take a view of the full system
• Understand your Bottleneck
• Solution might be in the different place than the problem
• Do Right benchmarks and Load Testing
• But do not expect them to catch all performance situations
Thank You!

Peter Zaitsev
pz@percona.com