Benchmarking DBMS

Herdi Ashaury

Benchmarking

What and why

Why Benchmarking?

Benchmark can help you :

observe the system’s behavior under load,

determine the system’s capacity,

learn which changes are important,

or see how your application performs with different data.

Benchmarking lets you create fictional circumstances, beyond the real conditions you can observe

Looking for Bottlenecks

Hardware : Disk, Memory, Network, etc

Operating System : File System, Memory Management, Drivers, etc

DBMS : SQL Layer, Storage engine layer

Schema Design : Indexes, Tables Structure, Data Type

Query

How the various parts of the system above interact?

Looking for Bottlenecks

Measure Quantities:

- How do they perform?
- Where is the most time spent?
- Which components are busiest?

Benchmarking Strategies

Application (Full stack) Benchmarking

Single Component Benchmarking

Full-Stack Benchmarking (+)

Benchmark all the infrastructure, web servers, applications, caching layers, databases, network, external resources, etc

Gives a better overview

Very difficult to implement

Results are complex to understand, and may be misleading

Application (Full stack) benchmarks can be hard to create and even harder to set up correctly

Design the benchmark badly = bad decisions

Sometimes, however, you don’t really want to know about the entire application.

Single Component Benchmarking (+)

Database is one component of the software

Database Benchmark useful if :

You want to compare different schemas or queries.

You want to benchmark a specific problem you see in the application.

You want to avoid a long benchmark in favor of a shorter one that gives you a faster “cycle time” for making and measuring changes.

MySQL Benchmarking (-)

The data itself and the dataset’s size both need to be realistic

Setting up a realistic benchmark can be complicated and timeconsuming, and if you can get a copy of the production dataset, count yourself lucky.

Database Benchmarks

Compare :

Different OS configuration

Different hardware

Different DB Systems

Different Parameters

Different Schema

Different Queries

Different Workload

Common Mistake Benchmarking[1]

Using a subset of the real data size, such as using only one gigabyte of data when the application will need to handle hundreds of gigabytes, or using the current dataset when you plan for the application to grow much larger.

Using incorrectly distributed data, such as uniformly distributed data when the real system's data will have "hot spots." (Randomly generated data is almost always unrealistically distributed.)

Using unrealistically distributed parameters, such as pretending that all user profiles are equally likely to be viewed.2

Using a single-user scenario for a multiuser application.

Benchmarking a distributed application on a single server.

Failing to match real user behavior, such as "think time" on a web page. Real users request a page and then read it; they don't click on links one after another without pausing.

Common Mistake Benchmarking [2]

Running identical queries in a loop. Real queries aren't identical, so they cause cache misses. Identical queries will be fully or partially cached at some level.

Failing to check for errors. If a benchmark's results don't make sense—e.g., if a slow operation suddenly completes very quickly—check for errors. You might just be benchmarking how quickly MySQL can detect a syntax error in the SQL query! Always check error logs after benchmarks, as a matter of principle.

Ignoring how the system performs when it's not warmed up, such as right after a restart. Sometimes you need to know how long it'll take your server to reach capacity after a restart, so you'll want to look specifically at the warmup period. Conversely, if you intend to study normal performance, you'll need to be aware that if you benchmark just after a restart many caches will be cold, and the benchmark results won't reflect the results you'll get under load when the caches are warmed up.

Using default server settings. There's more on optimizing server settings in later chapters.

Benchmarking too quickly. Your benchmark needs to last a while. We'll say more about this later.

Types of Benchmark

Stress Testing, System tested to its limit

Load Testing, System tested against an expected load

Endurance Testing, Measure stability on long term

Spike Testing, defines the system's behavior in such circumstances. May occur sudden change in workload

What to Measure

Throughput

Response time or latency

Stability Concurrency

Stability Scalability

Throughput

The most widely used measured quantity

Number of successful transactions per time unit (usually second or minute)

Normally it focuses only on average result

Widely used in marketing for comparisons

Important for Stress Testing

Latency

Total execution time for a transaction to complete

Min / Avg / Max response time

Important in Load Testing

Stability (Scalability)

Measures how system performs when scaling

Finds weak spots before they become a serious problem

Useful for capacity planning

Endurance Testing

Stability (Concurrency)

Measures how system performs when number of threads/connections changes

Defines if it is useful to impose an upper limit in number of threads and introduce queues

Spike Testing

Performance Testing Goals

High throughput

Low latency

Stability when faced with scalability and concurrency

Perencanaan dan Desain Benchmark

Define Pre-benchmark goals

Understand the workload to reproduce

Record Everything

Create Baselines

Define post-benchmark targets

Database Benchmarking

How

Steps for Database Benchmarking

Preparation, run benchmarks on a realistic setup and similar environment

Isolate the benchmark setup, ensure that no other application are using either database, pc, network.

Record (everything), HW/SW statistics.

Automate benchmark

Simplify the recurrent execution

Avoid human mistakes

Improved documentation

Easy to reproduce

Easy to analyze

Analysis of results

Process all the data collected

Identify which data provide useful information

Answer the questions defined in your goals

Document the result

Conclusion

Tools untuk Benchmarking

mysqlslap

MySQL Benchmark Suite (sql-bench)

Super Smack

Database Test Suite

Sysbench

MySQLSlap

Load emulator client

It executes in 3 stages :

- Create the table structre and load data
- Run Tests
- Cleanup

Output examples

```
Benchmark
Average number of seconds to run all queries: 68.692 seconds
Minimum number of seconds to run all queries: 59.301 seconds
Maximum number of seconds to run all queries: 78.084 seconds
Number of clients running queries: 10
Average number of queries per client: 1
```

Pertanyaan?

Referensi

Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko. 2012. High Performance MySQL: Optimization, Backups, and Replication (3rd. ed.). O'Reilly Media, Inc.