# Research on performance testing model of databases of different structure

**Liu Xiaochuang**

Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China

lxc380@foxmail.com

**Keywords:** performance testing; relational database; NoSQL database; HBase

**Abstract:** Using our self-innovate performance testing software PTF, Performance Testing Facility, this paper comes up with a new performance testing model and executes performance testing on many databases of different structure. The databases of different structure include HBase, MySQL, Oracle, DM7 and etc. After analyzing different features of these databases according to the result, we get the same standard as in the industry, which proves the usability of our model.

## 1. Introduction

Relational database and NoSQL (Not Only SQL) database play an important role in our life and industry. Their separate and combined use provides a solid data guarantee for the server. When testing relational databases, people often use the international standard TPC-C[1], which was come up with by Transaction Processing Performance Council(TPC) and usually used to measure databases' performance by simulating transactions in a real environment. When testing NoSQL, people often use Yahoo!'s open source YCSB test suite. However, there is seldom unified model for testing SQL and NoSQL and comparing them. In this paper, a new practical testing model is proposed, and the performance testing Facility (PTF) is used to test and analyze HBase and various relational databases.

The content of this paper is mainly divided into the following parts: the first section introduces SQL, NoSQL, and related tests; the second section introduces the new test model; the third section introduces the content of the test; the fourth section is the analysis of the test results; the fifth section is the conclusion.

## 2. Overview of SQL, NOSQL database and related test

### 2.1 SQL and NoSQL database

Relational database is also called SQL database. Since the relational database model was defined in 1970, RDBMS has become the standard of the database since the 1980s, such as Oracle, MySQL, SQL Server, PostgreSQL and etc[2]. These databases store entities in the real world and their relationships through relational datasheets, which is accessed through SQL language. RDBMS supports ACID features of transactions and has a wide range of applications, such as bank account trading system, order system, educational administration system, etc. In addition, RDBMS can easily support join queries and nested queries between tables, which provides great convenience for application programming.

NoSQL database comes from Google's BigTable in 2004, and it is a sparse large table. Its data model is the key-value model [2], which is mainly used in search engine data. With the advent and rise of web2.0, technologies such as blog, micro-blog, and SNS have made everyone a producer of information, resulting in a large number of unstructured data and its processing requirements, which has brought great challenges to the backstage storage system. NoSQL makes up for the weakness of traditional database in dealing with Web 2.0 website. These features include [3]:

- Performance requirements for high concurrent read and write of databases.
- Efficient storage and access requirements for massive data.

- Requirements for high scalability and availability of databases.

Moreover, for Web 2.0, many of the main features of relational databases are often useless, such as:

- High Consistency Requirements for Database Transactions
- High real-time requirement of database reading.
- Requirements for complex SQL queries, especially multi-table Association queries.

For the comparison between SQL and NoSQL, the following table summarizes [2] [4]:

TABLE I. Comparison between SQL and NoSQL

| **Database type** | **SQL database** | **NoSQL database** |
|---|---|---|
| Application | Transactions | Search |
| Data Model | Relational Tables | Key-Value Pairs |
| Math | Set Theory | Graph Theory |
| Volume | Normal | Big |
| High Concurrency Read And Write | Normal | High |
| Extended Model | Vertical Expansion | Horizontal Expansion |
| Consistency | Strong | Weak |
| Usability | Good | Good |
| Extension Cost | High | Low |
| Programming | Easy | Hard |

### 2.2 Related Testing Research about SQL and NoSQL Database

#### 2.2.1 Relevant Testing Research on SQL Database

Transaction processing performance council-C(TPC-C) defines a transaction processing model for a "sales-order system". And transaction processing in the model includes generating order transaction, distributing order transaction, recording payment transaction, order query transaction and warehouse query transaction. This transaction processing model is universal to the performance testing of the OLTP database. Similar international standards include TPC-H standards for testing OLAP databases and TPC-W standards for testing e-commerce databases[5]. And there are also many related studies about this [1]. [6] is a model design of a test system for massive relational databases, which can test relational databases more commonly.

#### 2.2.2 Relevant Testing Research on NoSQL Database

For NoSQL testing standards, TPC recently proposed the first industrial benchmark evaluation model for big data [7]. However, the industry's evaluation of NoSQL mainly uses YCSB for comparative testing of NoSQL: [8] theoretically analyses and compares various NoSQL in terms of data structure, consistency, and extension framework. [9] compares and analyses the execution time of 10 NoSQL under single node condition. [10] analyzed Cassandra, HBase, and MySQL in theory and experiment. [11] assesses MongoDB, HBase and Cassandra in different configurations (CPU core number, number of nodes, number of replicas) on Amazon's EC2. And [12] evaluates the performance of Redis, MongoDB, Couchbase, Cassandra and HBase in a distributed environment.

In this paper, a new performance testing model is proposed. Using PTF, a self-developed performance testing tool, we test relational databases and HBase under a unified standard on a single node.

## 3. The performance testing model

The purpose of performance testing of databases is to find whether the performance of databases is up to standard under different types of transactions and under different pressures. Aiming at these two requirements, the performance test model described in this paper proposes two models:

transaction control model and pressure control model.

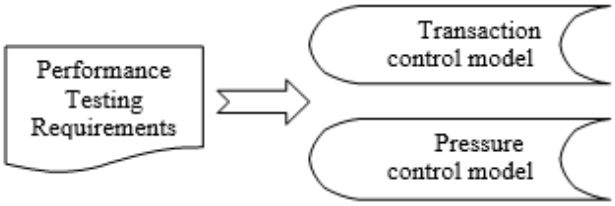


Figure 1. Get the testing model according to the testing requirement

### 3.1 Transaction Control Model

A transaction is a successful submission of a database at a time. Thr transaction control model is to control the type and distribution of transactions. We deem that transactions in a database have four types: write, read, update, and delete, and one transaction is regarded as one operation. For each operation, the distribution of data includes:sequence distribution generation, random average distribution generation, normal distribution generation, long tail distribution generation, etc. Operation and Distribution combine to produce various transaction models. This paper mainly analyses and discusses two transaction model: write Operation combine sequence Distribution, namely sequential writing, and read Operation combine random average distribution, namely random reading.

#### 3.1.1 Design of The Table Structure

Because the table models of SQL and NoSQL databases are different, it is necessary to unify the structure of tables when testing them.

For MySQL, Oracle, PostgreSQL and DM7 databases, their test tables are designed as 10 columns, with the primary key ID increasing by itself, and each column has 20 bytes of random strings. The specific table structure is shown in TABLE II.

TABLE II. Relational database table structure

| **id** | **field_0** | **field_1** | **…** | **field_9** |
|---|---|---|---|---|
| 0 | Kfd93D… | … | | |
| 1 | | | | |
| … | | | | |

The table structure of HBase is different from relational database's. Because the columns under the same column family are stored together, the column families of HBase correspond to the tables of a relational database. The specific table structure is shown in TABLE III.

TABLE III. Hbase table structure

| **rowkey** | **Cf** | | | |
|---|---|---|---|---|
| | **field_0** | **field_1** | **…** | **field_9** |
| 000 | Kfd93D… | … | | |
| 001 | | | | |
| … | | | | |

#### 3.1.2 Sequential Writing

Sequential writing insert table in the direction of increasing the primary key. For HBase, each transaction inserts a cell located by rowkey, column family and column, because a rowkey corresponds to a column family of 10 columns, so it takes 10 transactions to finish inserting one rowkey. When multithreading works, each thread inserts only some rowkeys. We design the rowkey

field inserted by each thread is the index of the current thread plus a multiple of the number of threads. For example, there are 6 execution threads in total. For the fourth thread, its index is 3. Then it inserts rowkey of 3 + N * Threads_number, where N is a natural number and Threads_number is the total number of execution threads. Because there are 6 execution threads, Threads_number is 6. In this way, in the process of insertion, the insertion order of each thread will gradually increase with the increase of N, so as to achieve the purpose of sequential insertion.

For relational databases, the data inserted in each row is different from that in HBase. For one row, field_0 is inserted first according to the primary key id, while the other columns are NULL at first. Then, 9 transactions are update operations for the remainding columns. When multiple threads are inserted, each thread performs the same incremental ID strategy as HBase.

### 3.1.3 Random Reading

Before random reading, 1 million rows of data are inserted to HBase and 4 relational databases. Transaction of one row is the content of all columns, a total of 10 columns.

## 3.2 Pressure control model

When testing different scenarios of a database, we need different pressure modes, including the magnitude of pressure and the change of pressure. So the pressure control model is divided into two parts: concurrent control model and delay control model.

### 3.2.1 Concurrent control model[13]

When testing, a different number of test threads will generate different connections on the database, and a different number of connections will bring different pressure requests. Concurrent control model controls concurrency pressure by the number of threads connected. As the number of threads increases, concurrency pressure increases gradually, But until a certain threshold is reached, increasing the number of threads can no longer improve performance as before. This is actually limited by server-side database's or tester's performance or network bandwidth.

When the bottleneck is at tester's performance, CPU usage or memory usage reach saturation, continuing increase thread number can not generate higher concurrent requests. The solution is to expand the number of test machines horizontally or increase the number of CPU cores and memory of test machines.

For the concurrent control model, the concurrent pressure is P, the number of testers is N, the number of threads per tester is t, when the performance bottleneck is not at tester, the formula of concurrent pressure is shown in formula (1).

$$P = N * t \quad (1)$$

### 3.2.2 Delay Control Model

Transactions are executed concurrently among many threads. However, within each thread, transactions and transactions are synchronized. After the former transaction returns the result, the latter transaction can initiate the request. The time delay from initiating a request to receiving a result is called $d_{total}$. There are other delay types: $d_{db}$ is database service delay, $d_{tester}$ is tester's scheduling delay, $d_{ping}$ is network transmission delay, is the delay of two machines' ping. The formula of $d_{total}$ is shown by formula (2).

$$d_{total} = d_{tester} + d_{ping} + d_{db} \quad (2)$$

When the number of threads increases to the threshold, the database server will queue because of the competition and scheduling between threads, so the $d_{db}$ will increase, and the database server will reach the bottleneck. This means that the pressure is too high and the server should be decompressed. In addition to the concurrent control model mentioned in section A, the way to decompress the server can also use the delay control model, which is to add artificial delay $d_{add}$ to the test machine. The concrete formula is as follows (3).

$$d_{total} = d_{tester} + d_{ping} + d_{db} + d_{add} \quad (3)$$

After adding $d_{add}$ on the test machine, although the number of concurrent threads on the server side is very high, the pressure on the server side can also decrease. In this way, the $d_{add}$ can be changed by arithmetic so as to dynamically realize increasing pressure or decompression of the server, which can achieve the purpose of pressure control.

In particular, for the incremental pressure control model, $d_{add}$ can be initially set to a larger time, such as 2s, then $d_{add}$ at regular intervals to reduce itself, such as each 1s to reduce $d_{add}$ by half. As time goes on, $d_{add}$ is close to zero, and the pressure of the database under test increases gradually, reaching the maximum at last.

## 4. Testing Content

### 4.1 Performance Indicators

TPS (transactions per second) and 95 percentile delay are used to measure the performance of the test. The higher the TPS or the lower the 95 percentile delay is, the better the database performance will get. At the same time, according to the monitoring system, other indicators, such as the CPU usage, network IO, memory usage and the number of disk IO per second of the system, are obtained to analyze the performance of the database. In the process of testing, because the network IO and memory usage have not reached the bottleneck, it is not considered as the reason of affecting performance.

### 4.2 Testing Process

For each test, the transaction control model and stress control model need to be clarified.

This paper tests five databases: HBase, MySQL, Oracle, PostgreSQL, and DM7. The selected transaction control model is sequential write and random read. The number of threads in the concurrent control model is 1, 2, 4, 8, 16, 32, 64 and 100, respectively. In order to facilitate comparison, $d_{add}$ in the delay control model is always 0. There will be a number of tests in total. The flow of a test is shown in Figure 2.

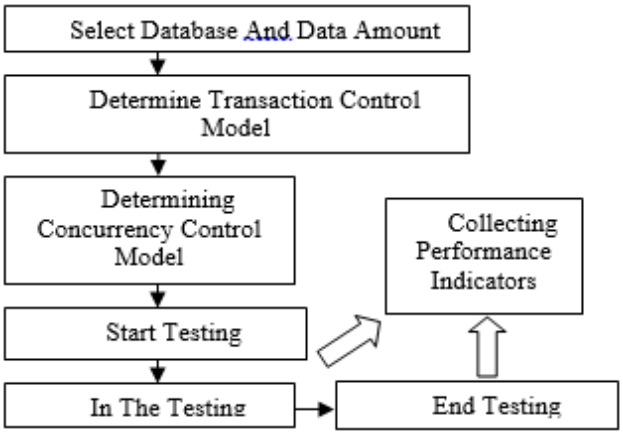


Figure 2. Testing Process

### 4.3 Testing Environment

To make all database environments the same, HBase environments are built using pseudo-distributed architecture. The following is the environment information for testing:

- CPU: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, 12 virtual cores, x86_64 architecture.
- Memory: 8GB.
- LAN: 100MB Ethernet.
- Hard Disk: HDD.
- Operating System: CentOS release 6.5 (Final).

Information of each database:

- HBase: JDK version is 1.8, Hadoop environment is 2.6.5, and HBase version is 1.2.6.

- MySQL: Version 5.6, using the InnoDB storage engine.
- Oracle: Version 11.2.
- PostgreSQL: Version 9.6.
- DM7: Version 7.

Monitor:

By installed on the server side, JMeter proxy is used to collect server-side information such as CPU usage, Memory usage, network IO size per second and Disk reads and writes per second. Then Web server receives data and dynamically displays the server status.

## 5. Analysis of test results

### 5.1 Sequential Writing Testing Analysis

Figure 3, Figure 4, and Figure 5 are sequential writing testing broken-line statistics of TPS, 95 percentile latency and server-side CPU usage for different databases under different threads. DM7 database will not connect when the number of threads is greater than 13, so only 13 threads are measured. When threads reach 64 and 32 respectively, HBase and Oracle reach the performance bottleneck, so they do not increase the number of threads.

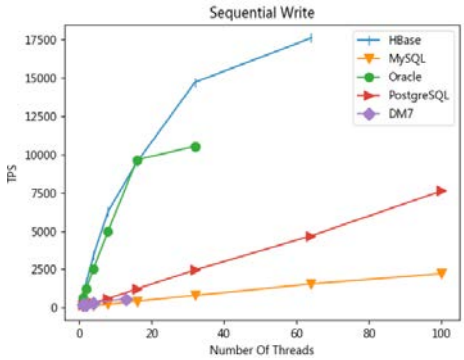


Figure 3. TPS of Sequential Writing

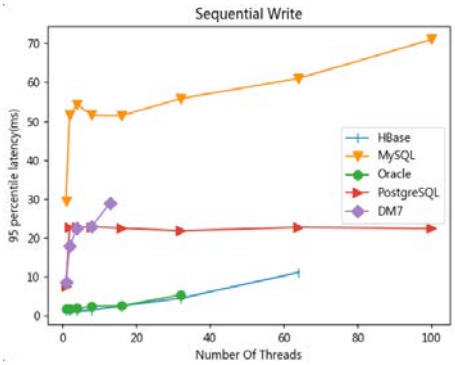


Figure 4. 95 Percentile Latency of Sequential Writing

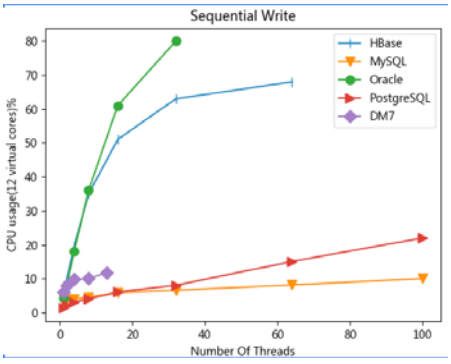


Figure 5. CPU Usage of Sequential Writing

Through analysis, we can see that HBase has the highest throughput and the lowest latency in write performance. This is because HBase uses LSM (log-structured merge-tree) data structure to store data. The insertion and update of data by HBase will be written to memstore in memory (implemented by LSM). After reaching a certain threshold (default 128MB), it will be flush into HFile, which is located on HDFS and is finally written to disk. In every testing process, there will be a process of disk IO surge and greater than 0, which is the process of writing HFile and HLog.

Other relational databases are mostly implemented by B or B+tree. From the test process, Oracle will do disk IO at intervals of a certain time, and the number of reading and writing disks per second is relatively small, generally less than five times per second. It can be seen that Oracle database is better for disk IO optimization. By properly planning and utilizing cache technology for disks, the write performance of the system is improved. Moreover, Oracle can make full use of multi-core CPU resources and achieve high concurrency and low latency for write requests. PostgreSQL is also a multi-process request processing architecture, while MySQL is a single-process multi-threaded architecture. When concurrency improves, PostgreSQL can make full use of CPU resources than MySQL, and thus achieve better performance. MySQL has the highest write latency and the lowest throughput because every write must be written to disk. When the number of threads increases to 2 or more, the latency of MySQL and PostgreSQL increases dramatically, due to competition for locks in the internal architecture. In the process of execution, we can see that the IO times per second of MySQL and PostgreSQL will be constant in a very small range, generally less than 10, while the IO numbers per second of DM7 database are very large and fluctuate greatly. The highest IO of the disk can reach 300 times per second, and generally more than 30 times IO per second. It is because DM7's disk optimization is a little worse, which greatly affects its writing speed. As can be seen from Figure 6, DM7 takes the most time to write 1 million data.

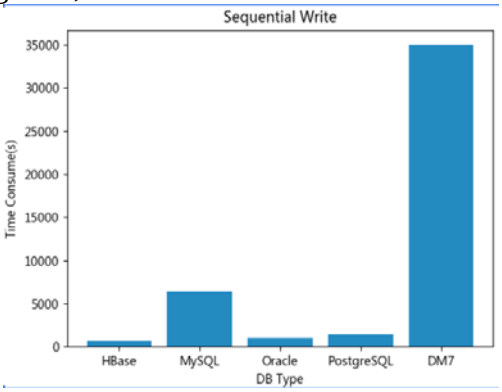


Figure 6. The Execution time of Sequential Writing of 1 Million Data

### 5.2 Random Reading Test Analysis

Figure 7, Figure 8, and Figure 9 are random reading testing broken-line statistics of TPS, 95 percentile latency and server-side CPU usage for different databases under different threads. When the amount of data is one million, a lot of data will be cached from disk to memory during testing, which greatly speeds up the performance of reading.

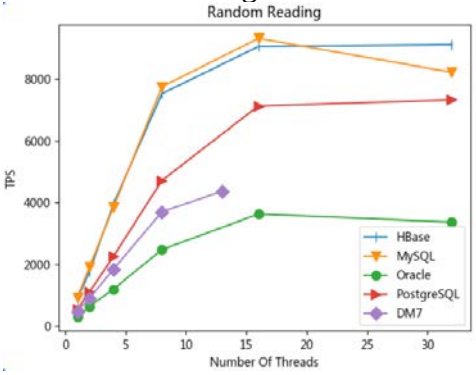


Figure 7. TPS of Random Reading

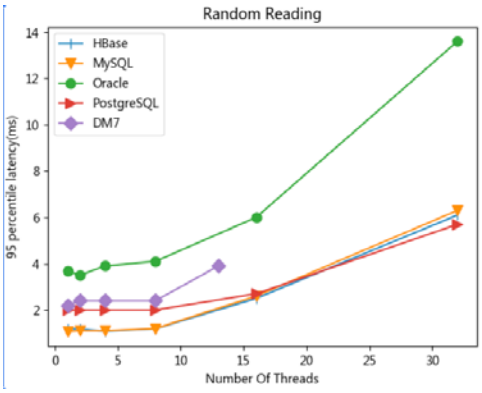


Figure 8. 95 Percentile Latency of Random Reading

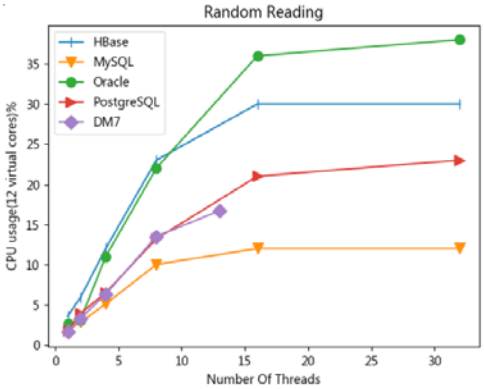


Figure 9. CPU Usage of Random Reading

From the results, we can see that HBase still performs well in reading performance, but occupies a higher CPU. MySQL's reading performance is similar to that of HBase, but its CPU consumption is low. It can be seen that although MySQL's architecture has poor writing performance, its reading performance is still good. Oracle has the worst read performance and consumes the most CPUs. When the number of threads reaches 16, all kinds of databases reach the optimum point, and the performance will not improve with the increase in the number of threads. This is because CPU will be tired of scheduling among threads, threads will be waiting for each other, and the latency will increase consequently, resulting in the throughput will be the same as former or even decline.

In summary, we find that HBase has better performance on write and read because of its LSM storage structure. Oracle has excellent write performance because of disk IO optimization and the ability to make full use of multi-core CPU. Oracle has the worst read performance when the data volume is small. For PostgreSQL and MySQL, the write performance of PostgreSQL is better than MySQL, and the read performance of MySQL is the best. DM7 database is not stable enough to establish too many connections. Because DM7's disk IO is poorly optimized, its write performance is also the worst.

## 6. Conclusion

This paper analyses the previous testing methods of SQL database and NoSQL database, and puts forward a general testing model. Using this model and PTF, we execute sequential writing and random reading performance testing of HBase and four relational databases. The results accord with the consensus of the industry, which proves the usability of the test model.

## References

[1] TONG Xiaodan. Design and implementation of the TPC-C database performance test system[D]. Harbin Institute of Technology, 2010

[2] Kepner J , Gadepally V , Hutchison D , et al. Associative Array Model of SQL, NoSQL, and

NewSQL Databases[C]// High Performance Extreme Computing Conference. IEEE, 2016.

[3] HUANG Shidong. Research of Testing Method of HBase Driven by Its Architecture and Business[D]. East China University of Science and Technology, 2013.

[4] Naheman W , Wei J . Review of NoSQL databases and performance testing on HBase[C]// International Conference on Mechatronic Sciences. IEEE, 2014.

[5] http://www.tpc.org/

[6] WU Haiping, YU Hongliang, ZHENG Weimin. General performance measures for massive databases [J]. Journal of Tsinghua University: Natural Science Edition, 2006, 46(7):1309-1312.

[7] Poess M . TPC’s Benchmark Development Model: Making the First Industry Standard Benchmark on Big Data a Success[J]. Lecture Notes in Computer Science, 2014, 8163:1-10

[8] Hecht, Robin, and Stefan Jablonski. "Nosql evaluation." International conference on cloud and service computing. IEEE, 2011.

[9] Abramova, Veronika, Jorge Bernardino, and Pedro Furtado. "Experimental evaluation of NoSQL databases." International Journal of Database Management Systems 6.3 (2014): 1.

[10] Tudorica, Bogdan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes." 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research. IEEE, 2011..

[11] Gandini, Andrea, et al. "Performance evaluation of NoSQL databases."European Workshop on Performance Engineering. Springer International Publishing, 2014

[12] Tang E, Fan Y. Performance Comparison between Five NoSQL Databases[C]// International Conference on Cloud Computing & Big Data. 2017

[13] WANG Hongman, SHI Zhihui, SHANGGUAN Linying,etc. Research on Load Test Model for 3G Online Charging System[J]. Journal of Beijing University of Posts and Telecommunications, 2009, 32(6):57-61