Comparisons between MongoDB and MS-SQL Databases on the TWC Website

Article *in* Journal of Software Engineering and Applications · April 2015

DOI: 10.11648/j.ajsea.20150402.12

3 authors, including:

Yin-Fu Huang
National Yunlin University of Science and Technology
165 PUBLICATIONS **1,503** CITATIONS

SEE PROFILE

American Journal of Software Engineering and Applications
2015; 4(2): 35-41
Published online April 28, 2015 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20150402.12
ISSN: 2327-2473 (Print); ISSN: 2327-249X (Online)

Comparisons Between MongoDB and MS-SQL Databases on the TWC Website

Chieh Ming Wu[1], Yin Fu Huang[2], John Lee[3]

[1]Dept. of Computer Science, Taiwan Water Corporation, Taichung City, Taiwan
[2]School of Computer Science & Information Engineering, National Yunlin University of Science and Technology, Douliou, Yunlin, Taiwan
[3]Dept. of Sales, Formula Chemicals Corporation, New Taipei City, Shulin District, Taiwan

Email address:
g9210822@gmail.com (C. M. Wu), huangyf@yuntech.edu.tw (Y. F. Huang), jtlinternational@hotmail.com (J. Lee)

To cite this article:
Chieh Ming Wu, Yin Fu Huang, John Lee. Comparisons between MongoDB and MS-SQL Databases on the TWC Website. *American Journal of Software Engineering and Applications*. Vol. 4, No. 3, 2015, pp. 35-41. doi: 10.11648/j.ajsea.20150402.12

Abstract: Owing to the huge amount of data in websites to be analysed, web innovative services are required to support them with high scalability and availability. The main reason of using NoSQL databases is for considering the huge amount of data and expressing large-scale distributed computations using Map-Reduce techniques. To enhance the service quality of customers and solve the problems of the huge amount of data existing in the websites such as Facebook, Google, and Twitter, the relational database technology was gradually replaced with the NoSQL database to improve the performance and expansion elasticity in recent years. In this paper, we compare both NoSQL MongoDB and MS-SQL databases, and discuss the effectiveness of the inquiry. In addition, relational database cluster systems often require larger server efficiency and capacity to be competent, but it incurs cost problems. On the other hand, using NoSQL database can easily expand the capacity without any extra costs. Through the experiments, it shows that NoSQL MongoDB is about ten times efficient for reading and writing than MS-SQL database. This verifies that the NoSQL database technology is quite a feasible option to be used in the future.

Keywords: NoSQL, MS-SQL, MongoDB, Relational Database

1. Introduction

The term "NoSQL" first made its appearance by Carlo Strozzi in the late 90s as the name of an open-source relational database and there is no relationship between and NoSQL currently in use. The usage of "NoSQL" that we recognize today traces back to a meetup on June 11, 2009 in San Francisco organized by Johan Oskarsson, a software developer based in London. They want to organize a discussion of the different ways of data storage, that can make these people interested in brainstorming together, "NoSQL" name eventually provided by Eric Evans made a name for this party [9].

NoSQL is a non-relational database management systems, it really means "Not Only SQL" and significant difference from traditional relational database management systems (RDBMS) in some ways. It is designed for distributed data stores where very large scale of data storing needs. The data object model may not require fixed schema, avoid join operations and typically scale horizontally. In other word, we can use both SQL and NoSQL database to achieve optimal results. For example, we can use NoSQL database to store huge amounts of unstructured data and store structured data using SQL database, so that can make good use of SQL syntax.

The main reason we have chosen MongoDB to do in this paper is that MongoDB is a document-oriented database and it uses JSON objects for data storing. In the Hadoop platform, regardless of the huge or small amount of data, all can be effective by management. If it's a small amount of data, then the performance of a single node has faster performance than in a multi-node cluster, and vice versa, which means that we must be manually set to one or more nodes in the cluster. MongoDB is stored in JSON format in the database, including tables are called collections and rows of JSON objects are stored as documents, so we can use MongoDB to store structured data, MongoDB also supports query operation for database, so for the relation database management system has a better alternative [7].

This paper uses MongoDB to store website message and implement the user interface. Finally, we compare the reading and writing performance between NoSQL MongoDB and MS-SQL, and found the NoSQL MongoDB is faster

(efficient) than MS-SQL in speed through the experiment data.

2. Related Work

Data aggregation is one of the important functions used in the database, especially in the face of business intelligence (such as ETL, OLAP) and Data Mining applications. In relational database, aggregation is used for more in-depth analysis in visualizing data. However, it is very difficult for the memory consumption on the huge amount of data and the calculation time [9].

As in [1], the authors use the files in the NoSQL MongoDB database [3], using its MapReduce algorithm to assist in processing large amounts of data. MapReduce [2] is a very popular program mode; 2004 Google use it to manage large amounts of data. This model has two primitive functions: Map and Reduce. Map is a function, and the value of the input is a single aggregation, while the value of the output is a key-value pairs. These applications are independent of each other, so you can build efficient and safe Map tasks that are parallelized operations on each node.

MongoDB can store huge amount of website information, and these messages can be unstructured. Compared to the relational database in the practical applications, MongoDB is more flexible. Same as the relational database, MongoDB entity may have multiple databases and for each database can have multiple collections. There is a big challenge for traditional relational database in face the rapid development of internet web 2.0 technologies. In [4], the authors propose the MongoDB Auto-Sharding architecture in order to response to the rapid development of internet web 2.0 technology in the cloud environment. The Auto-Sharding main objective is that it does not require a larger or stronger machine and is able to take responsibility for split data on distribution and automatic balancing to store more data and handle more load. They propose a FODO algorithm to improve the balance of the original algorithm so you can balance between effective data server and enhance the cluster effectiveness of parallel reading and writing.

In [10], the authors propose a mechanism for automatic load balancing MongoDB, and using heat-based automatic load balancing mechanism to reduce costs. Some studies have proposed a NoSQL MongoDB allow seamless support for JDBC SQL on the MongoDB database queries, and provide a virtual architecture allowing users to query and merge information from NoSQL and RDBMS. The main approach is to convert a single SQL query syntax to the APIs on NoSQL [5]. There is research study parsing LINQ query into MongoDB collection and rewrites them for MongoDB API format. The rewritten query execution on MongoDB and the returned results are converted to in-memory data structure, and then processed by JSINQ, reached the capacity by query MongoDB with LINQ [6]. Some studies explore the MongoDB data insertion time performance, in [7], the authors point out that both in writing or reading on the job, MongoDB performance come much better than MySQL.

3. Methods

3.1. Motivation

Because of the rapid growth rate of the huge amount of data in web application, the traditional RDBMS cannot be applied for several GB of data growth, and therefore NoSQL has been used to solving RDBMS Problem when maxed above limit. There are many management information systems in Taiwan Water Corporation (TWC). Some systems will also face the problem of huge amount of data bursts. NoSQL provides a much more elastic, schemaless data model that suitable maps to an application's data organization and simplifies the interaction between the application and the database resulting in less code to write, debug, and maintain. That is why we choose the best NoSQL solutions for this job.

3.2. Background

Learn MongoDB can help us to manage the huge amounts of data from a web application; a document-oriented database. We also found the MongoDB is indeed a reliable and efficient system. MongoDB allows almost unlimited horizontal expansion. This paper uses JAVA to deal with JSON file that is language-independent configuration files. To process records in MongoDB is simpler than in RDBMS and more flexible. MongoDB is very powerful and use the document to the basic unit of database, and it is a collection of schema-free database. An independent MongoDB entity can manage multiple databases and it has a powerful JavaScript command line interface.

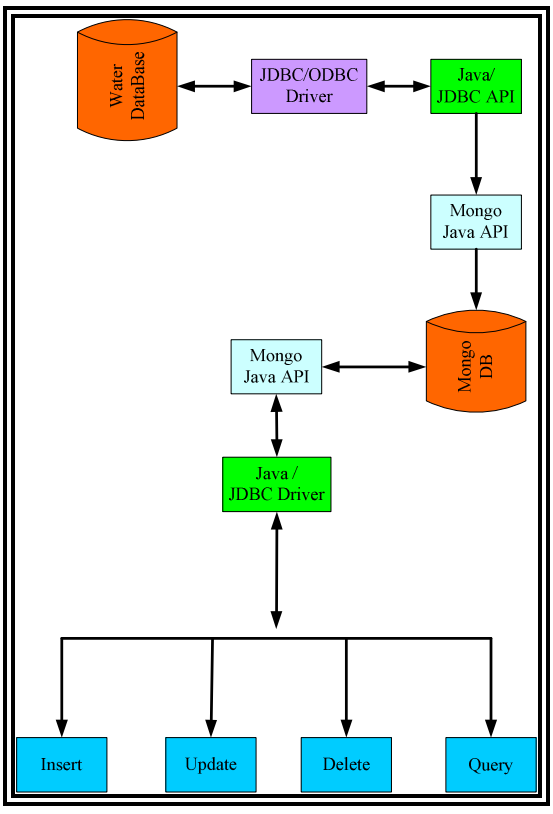


Fig. 1. *System Framework.*

This paper uses of JAVA programming language to read the related messages from the website in TWC, and insert the messages into the MongoDB database to further implement the relevant data processing functions, such as insert、query、delete and update functions. Fig.1 uses the JAVA JDBC DRIVER to capture the data from TWC website, using the JAVA DRIVER and MongoDB API to insert data into MongoDB database and implement various functions. Fig.2 shows the insert、update and delete functions by use JAVA syntax.

```
public void INSERTforMongoDB() throws UnknownHostException
{

    BasicDBObject doc=new BasicDBObject();
    doc.put("NOs", nost.getText());
    doc.put("reason", reasont.getText());
    doc.put("range", ranget.getText());
    doc.put("bulletin", buetint.getText());
    doc.put("restore", restoret.getText());
    coll.insert(doc);
    coll.setWriteConcern(WriteConcern.SAFE);
    text.append(doc+"\n");
    text.append("Data Inserted OK \n");

}
```

```
public void DELETEforMongoDB() throws UnknownHostException
{

    BasicDBObject QueryDel=new BasicDBObject();
    QueryDel.put("NOs", nost.getText());
    DBObject removeObj=coll.findAndRemove(QueryDel);
    text.append(removeObj+"\n");
    text.append("Above Data Deleted OK \n");
    mongodbCONNECTION();
}

public void UPDATEforMongoDB() throws UnknownHostException
{

    BasicDBObject QueryUPDATE=new BasicDBObject();
    QueryUPDATE.put("NOs", nost.getText());
    DBObject findObj=coll.findOne(QueryUPDATE);
    findObj.put("NOs", nost.getText());
    findObj.put("reason", reasont.getText());
    findObj.put("range", ranget.getText());
    findObj.put("bulletin", buetint.getText());
    findObj.put("restore", restoret.getText());
    coll.update(QueryUPDATE,findObj);
    text.append(findObj+"\n");
    text.append("Above Data Updated OK \n");

}
```

Fig. 2. *The insert delete update function for JAVA.*

3.3. Query

MongoDB provides find and findOne function to perform the ad hoc query in the database. We can use $lt, $lte, $gt, $gte comparison operator to do the scope of the query. It may also use $OR, $NOT. etc to enhance criteria query. Fig.3 shows use of $OR to query the water suspension number and/or reason and range of influence, and as long as one of the inputs meet the conditions, you can find the relevant information.

```
public void QUERYforMongoDB() throws UnknownHostException
{

    DBObject clause1 = new BasicDBObject("NOs", nost.getText());
    DBObject clause2 = new BasicDBObject("reason", reasont.getText());
    DBObject clause3 = new BasicDBObject("range", ranget.getText());
    text.append(clause1.toString());
    text.append(clause2.toString()+"\n");
    text.append(clause3.toString()+"\n");
    BasicDBList or = new BasicDBList();
    or.add(clause1);
    or.add(clause2);
    or.add(clause3);
    DBObject query = new BasicDBObject("$or", or);
    text.append(query.toString()+"\n");
    cur = coll.find(query);
    DBObject str1= cur.next();
    String NOS=(String) str1.get("NOs");
    String reason=(String) str1.get("reason");
    String range=(String) str1.get("range");
    String stop=(String) str1.get("stop");
    String bulletin=(String) str1.get("bulletin");
    String restore=(String) str1.get("restore");
    setFText(NOS,reason,range,stop,bulletin,restore);
}
```

Fig. 3. *Using $OR condition to query.*

3.4. Indexing

Like a book's index that allows us to become more efficient in querying instead of looking through the whole book. Indexing database is to create an entry point of a query. For example, you will often query the user name in the database, so there is necessary to build an index on the key of user name to speed up queries. The index of MongoDB and traditional relational database is almost the same (query optimization; index tuning, etc.) which requires some skills to do. This generates the command for this query as follows.

>db.people.find({"username":"wjamin"})

We can create an index based on the above query of the key (shown below).

>db.people.ensureIndex({"username":1})

The index needs to be set only once per collection. If you try to establish the same index again, nothing will happen. Indexes in MongoDB make queries run faster and more efficiently. However, indexes have their cost: for every write includes insert, update, or delete will take longer for every index you add. This is because MongoDB has to update all your indexes whenever your data changes, as well as the document itself. Thus, MongoDB limits to build 64 indexes per collection. As shown in Fig.4, we have the username index, but the server have to scan all the collection, then it can find the date. Hence, it is very time consuming for a large collection. For example, the index on "username" wouldn't help much for this sort:

>db.people.find({"date":date1}).sort({"date":1, "username":1})

So we should be indexed on the date and the username.

>db.people.ensureIndex({" date":1, "username":1})

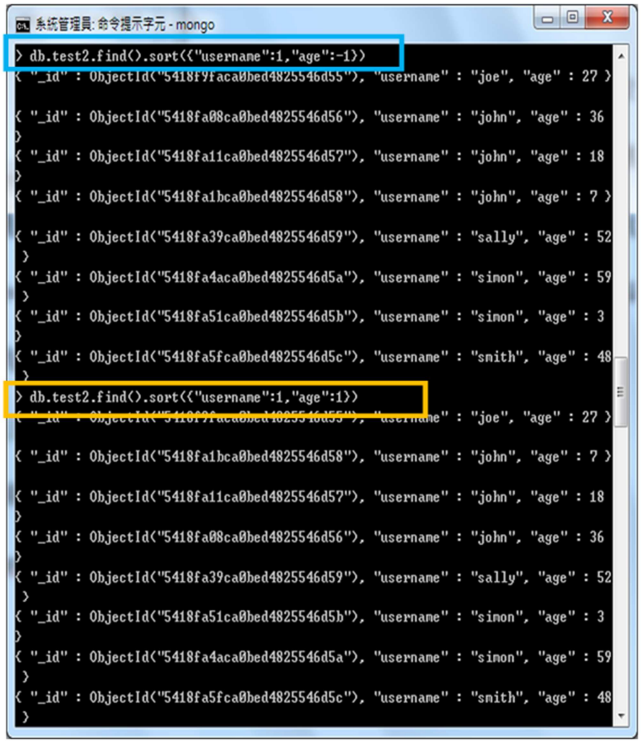


Fig. 4. *Compound query results with increasing/decreasing sorting.*

3.5. MapReduce

MongoDB provides aggregation tools in several basic query functions. These tools starts from simple task of calculating the number of documents to complex data analysis. MongoDB provide group function which allows us to perform more complex aggregation (similar to SQL's GROUP BY). In addition, the MapReduce functions are super useful in the aggregation tool and uses JavaScript as its "query language" so it can express arbitrarily complex logic. MapReduce is an aggregation method, which can be easily on multiple servers in parallel operation. The problem will be decomposition by different nodes to solve. When all the solutions return after the completion of the node, the answers will be merged into one complete answer [3].

MapReduce uses the map and reduce. With map being a kind of corresponding relation, it will correspond to the collection of each document. A little like separating into groups. The reduce will use the map list for induction, until the list of each key reduces it to a single element. This element is returned to the shuffle step until each key has a list containing a single value. In the map corresponding to use a special emit function to return values, emit function give MapReduce a key and a value. We use average water quality of purification plant in Taiwan Water Corporation counties as an example to illustrate how to use the powerful MapReduce functions.

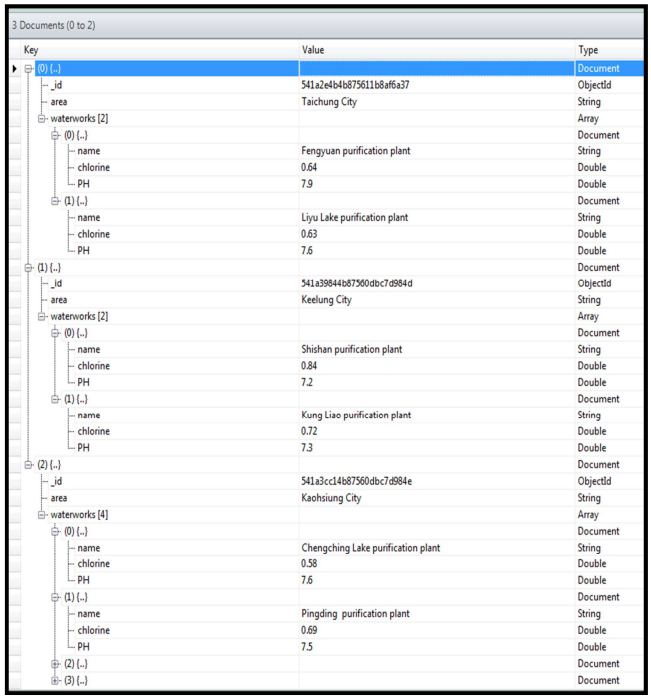


Fig. 5. *The average water quality in the JSON format.*

As shown in Fig.5, this is an average water quality data. For the convenience of description, we only input partial data and use the TAICHUNG city, KEELUNG city and KAOHSIUNG city as the key values. We want to get the average value of water quality in various PH and CHLORINE per county, and with the key values of PH and CHLORINE.

Fig. 6. *Calculate the average water quality using the MapReduce function.*

Such as Fig.6, in the Map function, that emit gives MapReduce a key like the one used by group and a value in the collection. In this case, we emit a count and some items of how many times a given key appeared in the document. The map function uses an emit function to return values that

we want to process later.

> map=function() {for (var idx=0;idx<this.waterworks.length;idx++){emit(this.area,{count: 1, chlorineV: this.waterworks[idx].chlorine,PHV:this.waterworks[idx].PH})}};

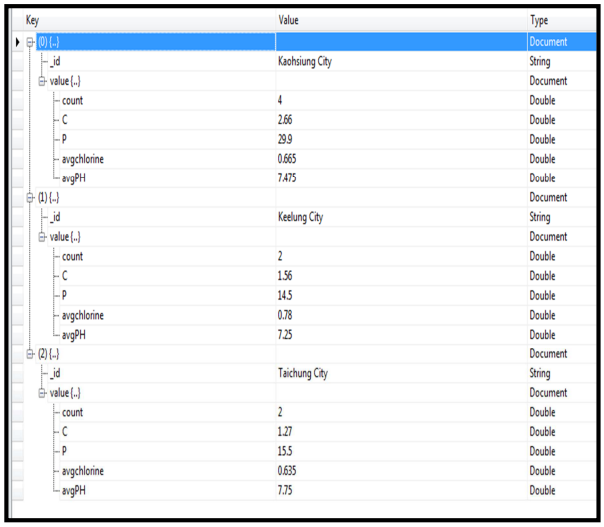
Now we have little documents that associated with a key from the collection. An array of one or more of these documents will be passed to the reduce function. The reduce function is passed two arguments:key, which is the first argument from emit, and an array of one or more documents that were emitted for that key:

>Reduce=function(key, values) {var reduced = {count:0, C:0, P:0}; values.forEach(function(val) {reduced.C += val.chlorineV; reduced.P += val.PHV; reduced.count += val.count; });return reduced;}

We use the "finalize" function to send reduce's output to calculate their average value:

>Finalize=function(key, reduced) {reduced.avgchlorine = reduced.C reduced.count;reduced.avgPH = reduced.P reduced.count;return reduced;}

The result shown in Fig.7, which is grouped by counties respectively and calculating the average value. The value of "count" is the number of water purification plant. The symbol "C" is the sum of residual chlorine effectively; "P" is the sum of PH; "avgchlorine" is the average residual chlorine effectively; and "avgPH" is the average PH in the corresponding county city.



Key	Value	Type
(0) {..}		Document
_id	Kaohsiung City	String
value {..}		Document
count	4	Double
C	2.66	Double
P	29.9	Double
avgchlorine	0.665	Double
avgPH	7.475	Double
(1) {..}		Document
_id	Keelung City	String
value {..}		Document
count	2	Double
C	1.56	Double
P	14.5	Double
avgchlorine	0.78	Double
avgPH	7.25	Double
(2) {..}		Document
_id	Taichung City	String
value {..}		Document
count	2	Double
C	1.27	Double
P	15.5	Double
avgchlorine	0.635	Double
avgPH	7.75	Double

Fig. 7. *The execution result of MapReduce Function.*

4. Experimental Results

4.1. Experimental Environment and Data Sources

In this section, we evaluate the performances on a ASUS PC with Intel® Core™2 Quad CPU 2.5GHZ and 2GB main memory running Windows 7 enterprise. All the experimental data were generated randomly and stored on a local 200GB Disk. We also install the MongoDB 2.6.3 、SQL Server 2005 Express and Eclipse IDE for Java Developers. The version is Juno Service Release 1, the data source comes from the TWC website.

4.2. Experimental Analysis

In this paper, we use Java language to develop SQL Server and MongoDB algorithm in the Eclipse integrated environment. Fig.8 compares the efficiency of data written between MS-SQL and MongoDB under the different operation times. Through the experiments under the indexed condition, MongoDB shows itself to be nearly 10 times faster than MS-SQL in writing ability.

In Fig.9, through experiment under the indexed condition and the different operation frequency, the ability to read data in MongoDB is nearly 10 times faster than MS-SQL.

In Fig.10 and Fig.11, we have found that under the 1,000,000 operating frequency, whether MS-SQL or MongoDB, the writing performance is far better than reading. Through the experiments, we have found the performance of writing is 3-4 times faster than reading.

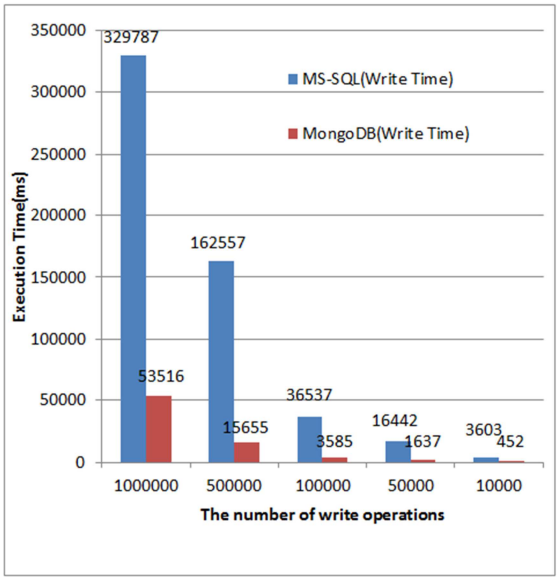


Fig. 8. *Compare the efficiency between MS-SQL and MongoDB on writing.*

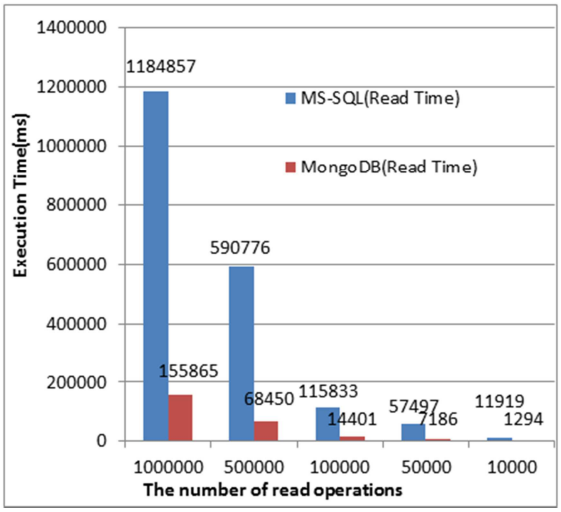


Fig. 9. *MS-SQL and MongoDB performance comparison on reading.*

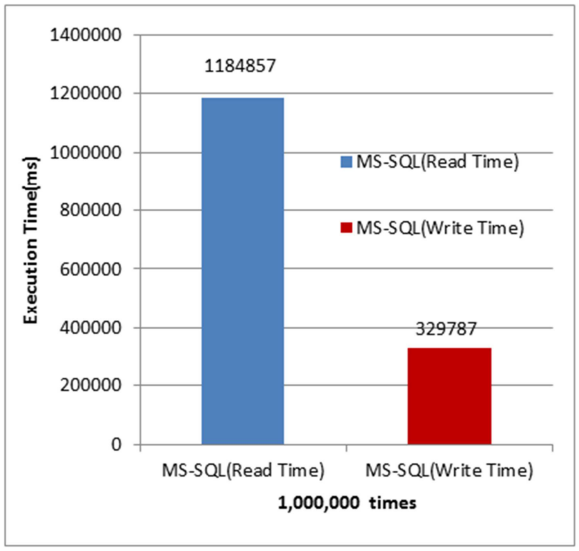


Fig. 10. *MS-SQL performance comparison on reading writing.*

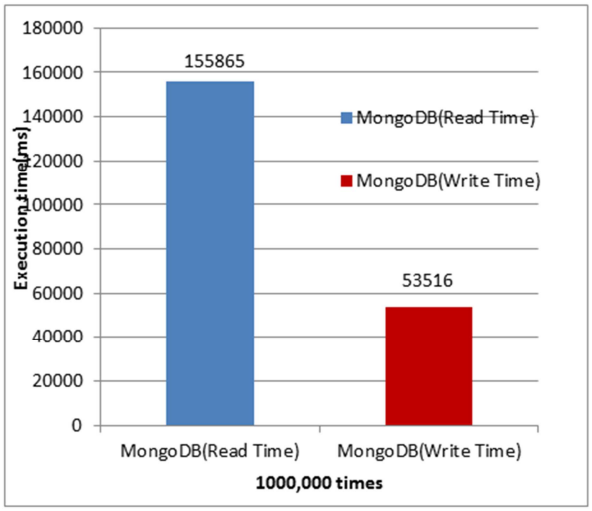


Fig. 11. *MongoDB performance comparison on reading/writing.*

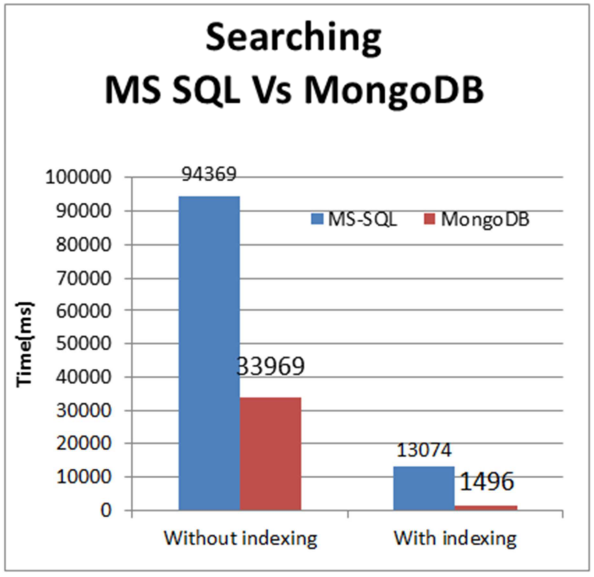


Fig. 12. *The influence of search efficiency in indexing.*

MongoDB insert or search for information is very fast. And the write performance of MongoDB or MS-SQL is better than reading efficiency. In Fig.12, under the indexed condition, the performance of searching is naturally better, especially in the MongoDB database.

In experiments using multithreading way to simultaneously read and write data to calculate their literacy effectiveness (whether it is reading or writing), Fig.13 shows that MongoDB in execution is still more efficient than MS-SQL. The experiment also found that MongoDB in a multi-threaded execution will remain stable. However, MS-SQL thread on Thread10 spends a considerable amount of time when writing, resulting in very inconsistent situations. Fig.14 effectively expresses the Thread10 exception; the maximum Y-axis set to only 100,000 in order to effectively reflect their differences.

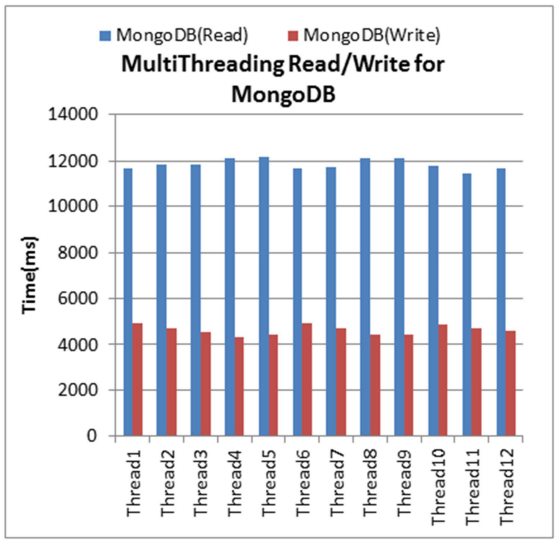


Fig. 13. *MongoDB multithreading read write performance.*

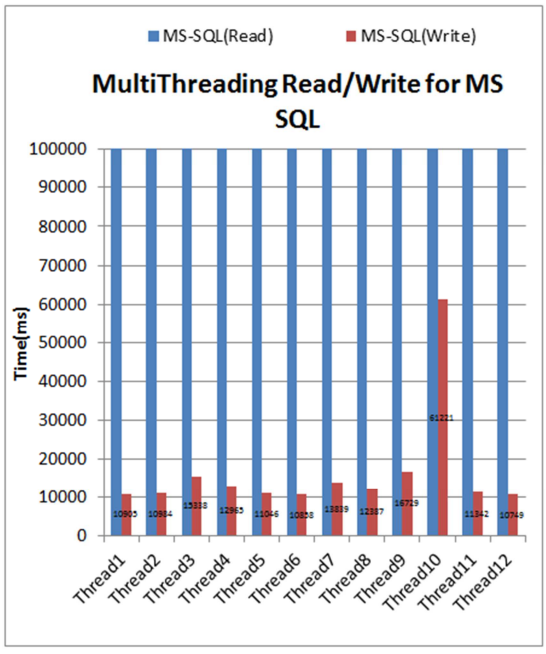


Fig. 14. *MS-SQL multithreading read write performance.*

5. Conclusions and Future Work

This paper mainly discusses the effectiveness of NoSQL; using the document-oriented NoSQL MongoDB database application to operate website message in Taiwan Water Corporation. We also use the open data of average water quality from the Taiwan Water Corporation website to illustrate MapReduce example.

We experimented various practical ways in MongoDB and use JAVA program to implement the task. In addition to the practical solution that MongoDB offers, it also has most of the internet application functions like index, replication, sharing, rich query syntax, and super elastic data model. We also compare the performance between MongoDB and MS-SQL. The results confirm the NoSQL MongoDB does have a better efficiency than MS-SQL.

Due to the popularity of the big data, the future trend for NoSQL will be based on integration. This integration will take place among the different varieties of NoSQL technologies and between SQL and NoSQL options. The convenience and the effectiveness of that integration will determine the big data applications in the enterprises (not just NoSQL technology).

References

[1] Bonnet, L.;Laurent, A.;Sala, M.;Laurent, B., REDUCE, YOU SAY: What NoSQL can do for Data Aggregation and BI in Large Repositories, 22nd International Workshop on Database and Expert Systems Applications (DEXA), pp. 483-488, 2011.

[2] J. Dean, S. Ghemawat. Mapreduce: simplified data processing on large clusters, Commun. ACM, pp.107-113, 2008.

[3] Kristina Chodorow & Michael Dirolf, MongoDB: The Definitive Guide , O'Reilly Media, 2012.

[4] Liu Yimeng; Wang Yizhi; Jin Yi, Research on The Improvement of MongoDB Auto-Sharding in Cloud Environment, 7th International Conference on Computer Science & Education (ICCSE), pp. 851- 854, 2012.

[5] Lawrence, R., Integration and Virtualization of Relational SQL and NoSQL Systems including MySQL and MongoDB, International Conference on Computational Science and Computational Intelligence, pp.285-290, 2014.

[6] Nakabasami, K.; Amagasa, T.; Kitagawa, H., "Querying MongoDB with LINQ in a Server-Side JavaScript Environment", 16th International Conference on Network-Based Information Systems (NBiS), pp.344-349, 2013.

[7] Nyati, S.S. ; Pawar, S. ; Ingle, R., Performance Evaluation of Unstructured NoSQL data over distributed framework, International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1623-1627, 2014.

[8] Okman, L. ; Gal-Oz, N. ; Gonen, Y. ; Gudes, E. ; Abramov, J., "Security Issues in NoSQL Databases", IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 541 – 547, 2011.

[9] Pramod J. Sadalage, Martin Fowler, "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", Addison-Wesley Professional, 2012.

[10] Wang Xiaolin ; Chen Haopeng ; Wang Zhenhua, "Research on Improvement of Dynamic Load Balancing in MongoDB", IEEE 11th International Conference on Dependable, Autonomic and Secure Computing (DASC), pp.124-130, 2013.