Journal of Theoretical and Applied Information Technology
31st March 2017. Vol.95. No 6
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A NOVEL METHOD FOR EVALUATION OF NOSQL DATABASES: A CASE STUDY OF CASSANDRA AND REDIS

[1]HADI HASHEMI SHAHRAKI, [2*]TAGHI JAVDANI GANDOMANI, [2]MINA ZIAEI NAFCHI
[1]Department of Computer Engineering
Shahrekord Branch, Islamic Azad University, Shahrekord, Iran
[2]Department of Computer Engineering
Boroujen Branch, Islamic Azad University, Boroujen, Iran
E-mail: [1]hadihashemi_sh@yahoo.com, [2]t_javdani@azad.ac.ir, ziaei.mina@yahoo.com

ABSTRACT

In today's complex world, due to the increasing use of web-based programs, smart phones and social networks, production rate is increasing constantly as well as volume of data. Also, companies try to provide their services with more features in order to stay ahead of their competitors. That increases the production of information by the users too. In this case, it seems that moving from rational databases to non-relational databases in a necessity. With regard to the various of use cases and their requirements, also the primary features of NoSQL databases, choosing the most appropriate NoSQL database can be big concern of developers. The right selection of NoSQL databases will avoid to wasting time, money, and energy. On this paper, we propose a general and structured method to help developers, customers, and managers to evaluate NoSQL databases in the right way and facilitate the process of decision making to select one of them.

Keywords: *Non-Relational Databases, Nosql Databases, Evaluation, Performance, Scalability*

1. INTRODUCTION

Nowadays, the volume of data is increasing constantly; hence the cost of relational databases scalability will be very expensive. In contrast, NoSQL databases are designed for appropriate horizontal scalability and implementation of the community hardware.

On the one hand, concept of "one size fit's it all" is not appropriate for current application scenarios and mainly is suitable for systems with high workload applications. [1]. Also, according to report of Digital Universe, the volume of data is expected to be double in every two years over the world. So, often systems that use RDBMS will be restricted against of the rapid data growth. In addition, since the emergence of RDBMS, most of the information systems have been built based on them [1].

Furthermore, some of information systems need to higher performance and distributed databases rather than higher reliability [2]. Existing cloud environments should to be supported by particular feature including flexible schema, fault-tolerance, simple invocations, optimum use of distributed indexes and RAM, availability, and replication of data over the multiple servers [3-4]. NoSQL databases always have many goals and advantages. One of their main goals is cost and risk reduction by using of community servers. This is the idea used by Google in BigTable.

The purpose of performance can be many factors such as throughput, run time, average operation latency, diffusion index points, being suitable for real time applications, etc. [5].

MapReduce is one of the effective techniques in enhancement of the performance [2]. However, it had been used for parallel processing by Google. Discussion, analysis, benchmark and evaluation of NoSQL databases together with comparison of them will help developers and customers to select the best choice to meet their requirements [3]. Careless in choosing databases and low attention to nature of data and environment results in to increasing re-work for revising structure and selection of database. Therefore, evaluation and analysis of databases should be done carefully by using of the best tools to save money, cost, and resources [6]. Indeed, paying enough attention during the evaluation process is crucial. On the other hand, regarding to data growth on the web, concerns about scalability, maintenance, management, and inefficient performance of database systems are increasing too. NoSQL databases have many goals and benefits. One of the main objectives is reducing costs and risks by using Commodity Server. The objective of the performance can be providing several factors such as throughput and response time, average delay of operations, the diffusion index points, the appropriateness of the database for indexing, suitability for real time systems, and suitability for low volume and accessible data [5]. MapReduce is one of the techniques that are used to increase performance of databases. MapReduce algorithm originally developed at Google for parallel processing of data [2]. Discussion, review, analyzing, evaluate and benchmark of NoSQL databases along with comparison of them will help developers and businesses to choose the best solution for their needs [3]. In case of selection of a database without considering their data types and the required working environment, a huge amount of re-works should be done to in order to select the best choice of database. Thus, evaluation and analysis of the databases by using the available tools in order to choose the best option should be considered specifically [6].

In fact, during the evaluation process, it is necessary to pay attention to the more effective items. However, due to the increasing of large volumes of data over Internet, concerns for scalability, maintenance, management, and inefficiency of database systems. These concerns are important factors influenced by the features of NoSQL databases. The previous studies indicate that there is not enough confidence in using and implementing of NoSQL databases [4,7-8]. It is why in some studies emergence of NoSQL database is called as "NoSQL movement" indicating lack of enough trust to them. Therefore, this paper tries to propose a novel method of evaluation and analysis of such databases to make them more transparent [9-11]. In order to eliminate these concerns, we need to prove performance and providing evaluation of the database. The novel aspect of this paper is providing a detailed evaluation method which is presented in Figure 21. The proposed method includes selection of the proper details (number of threads, number of records, etc.) based on the general rule of 2^i. This leads to better evaluation by considering real environments (with different workloads). Furthermore, this paper proposes a new metric named "Diffusion Index Points" as a new factor that has not been used yet.

Finally, it should be noted that, the main aim is to clarify the behavior of a database in the provided conditions and environments.

2. SELECTION OF CANDIDATE DATABASES AND OTHER FACTORS

Regarding to our goal and variety of NoSQL databases and their features, it has been tried to select two databases from two more popular, key-value and column-store categories. Thus, Cassandra and Redis were selected. For the evaluation of each Workload, we used 1, 2, and 4 threads for Machine1 and Machine2. Also, we used 1, 2, 4, 8, and 16 threads for Machine3. We can evaluate maximum processor power using the same number of test threads and processor threads [9]. We can measure maximum power processors in this direction. The other goal of this paper was targeted selection of numbers to use 2^i general rule for the numbers.

3. TEST ENVIRONMENT

This evaluation had been done on 3 machines. Machine1 had a Core i3 processor, 2GB RAM; Machine2 had a Core i5 processor, 6GB RAM, and Machine3 was an account of AmirKabir University cloud system that had 16 cores and 32GB RAM. The evaluation had been done using YCSB framework. The number of evaluation threads was 1 to 16, and the number of records was selected 2000 to 1024000. On each database we used two Workloads, WorkloadA and WorkloadF. Totally, 423 experiments have been done and the main steps of the evaluation were "identifying effective factors", "benchmarking", and "analysis and interpretation".

These machines and different variables were selected to achieve different results and closer to real environments. Variable names indicated machine number, database name, Workload, and number of threads. For instance, "M1_Cassandra_WorkloadA_with 1 thread" indicates Machine1, Cassandra DB, WorkloadA and one thread.

4. LIMITATION

One of the features NoSQL database evaluation is that so many factors should be considered. Thus, developers need to deal with many details in the evaluation process.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

With regard to the activities and internal operations of operating systems and getting different results in different time slices, it is recommended to do same evaluations in different times. To make the evaluations closer to the real conditions environments and in case of availability of laboratory facilities, evaluations can be done in different models of Sharding technique. On the other hand, the CPU Overclock might be used to evaluate the maximum CPU power for processing records.

5. RESULTS

First, a brief definition of the concepts used in the evaluation process is provided.

- *Run Time:*

Runtime is the total testing time (operations on records) based on type of workload and the number of operations.

- *Throughput:*

Throughput is the Average number of operations in a second that is estimated by the YCSB framework.

- *Average latency:*

Average latency is the average measured response time of a given database operation in the microsecond.

- *Diffusion Index Points on Run Time:*

It shows to what extent increasing number of threads influences response and execution time.

We did benchmarking based on different criteria including run time, throughput, average latency and diffusion of run time points.

5.1 Run Time

As shown in Figure 1, the run time for 1, 2, and 4 threads there is the same from 2000 to 64000 records approximately. But the start of effectiveness and different number of threads is from 128000 records. So that, 4 threads case shows the best performance and the worst performance is shown on 1 thread.

Notable point is that there is the same performance on 2 and 4 threads on 1024000 records. That means in Cassandra and WorkloadA, there is no considerable different for population of 1000000 records and number of users (threads). But, if this number of records is requested by one user in one node with same configure of Machine1, that node will run time will be doubled.

In Figure 2 WorkloadA is compared with WorkloadF in the same machine and database. WorkloadA is 1.28 times faster than WorkloadF on 512000 records and 1.46 times on 1024000 records.

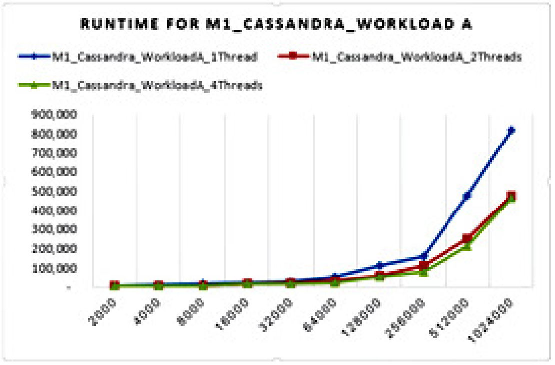


Figure 1: Run Time on Cassandra and Workloada

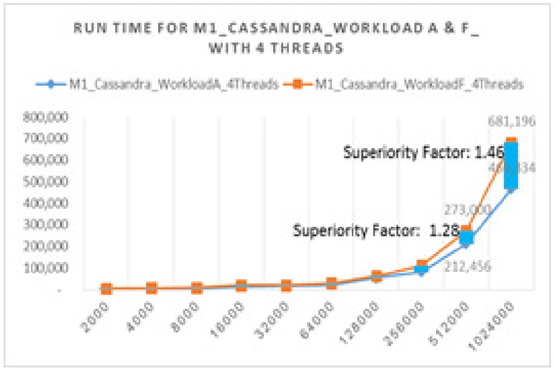


Figure 2: Comparison of Workloada and Workloadf

On Redis database the start of effectiveness is seen from 64000 records. Important note is that Machine1 can process only up to 521000 records, as shown in Figure 3. Also, this machine and WorkloadF got timeout error. This case has been shown in Figure 4.

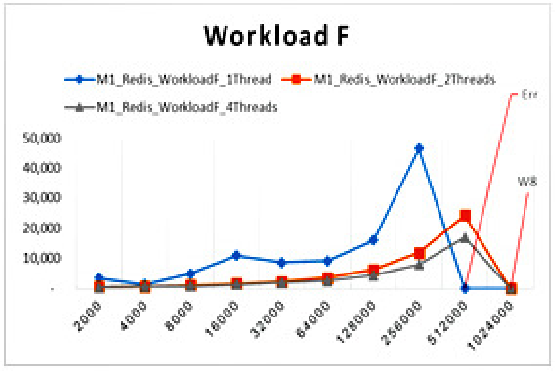


Figure 3: Run Time on Redis and Workloadf

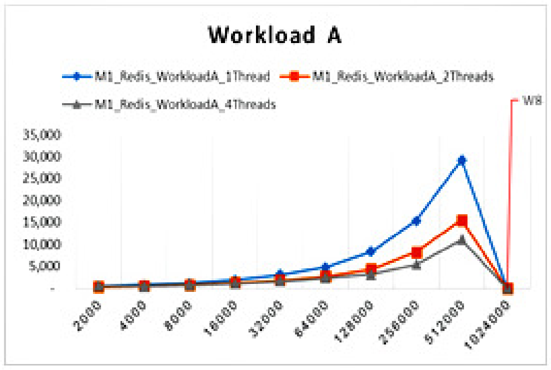
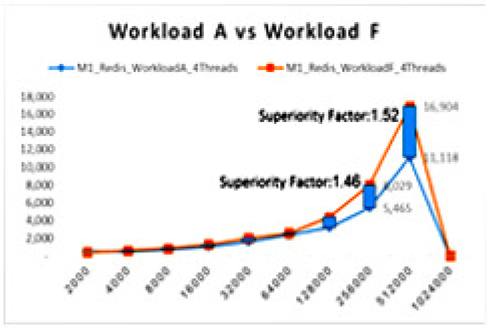
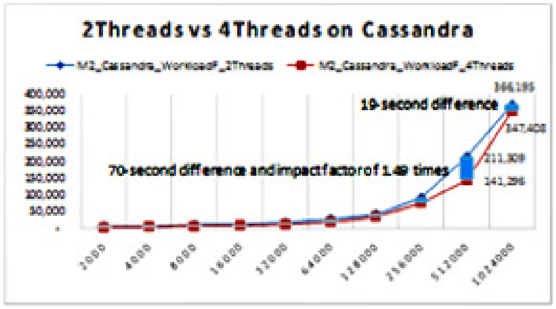


Figure 4: Run Time on Redis and Workloada

But after loading data and on 1024000 records, the system was in a Waiting state. This is mainly because of the In-Memory nature of the database and low memory (only 2GB) available in Machine1. This means that, if a node with the same power of Machine1 does not able to process 512000 records by a user. But this number of records can be processed by 2 threads as well as by 4 threads. On Machine1, two Workloads have been compared as shown in Figure 5(a). Naturally, since WorkloadF is heavier than WorkloadA, it shows weaker performance. So, superiority factor on 256000 records is 1.46 and on 512000 records is 1.52. As previously mentioned, this is because of the In-Memory nature of Redis.
Although in Cassandra and 1024000 records, there is greater run time rather than Redis. But, the test has been done completely in Redis, because of its In-Memory feature. As shown in Figure 5(b) we compared 2 threads and 4 threads on Machine2. Run time difference on 512000 records for 4 threads rather than 2 threads was 70 s. This difference reaches to 19 s by doubling number of the records. So we can conclude that in real environments with a certain difference between the numbers 2 and 4 of threads and population of 1000000 records cannot be felt. While this process cannot be seen on Redis. So that, in both of tests the different is 3 s. as shown in Figure 6, that means Redis retains its influence. That it shows the nature of Redis against increasing the number of records and the number of threads in this range in real environments. According to this test, can be said this event will remain on more records. Based on this evaluation with confidence, we can be said the impact of this factor in Redis is much weaker.



(a)



(b)

Figure 5: Comparison of Run Time of Both Databases (A) And Comparison of 2 Thread and 4 Threads States On Workloadf and Machine2(B)

Figure 7 is a sub-display of Figure 5 depicting 128000 and 256000 records states. The run time only on 256000 records for 4threads is about 124ms greater than 2 threads case.

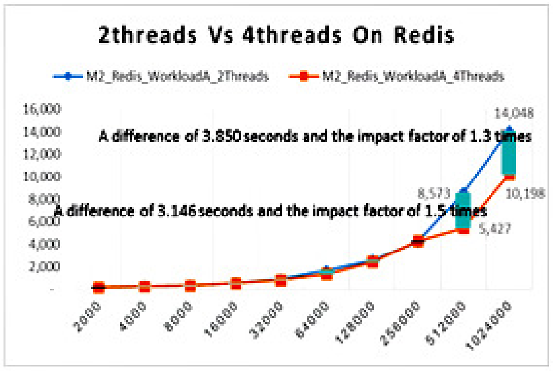


Figure 6: Comparing of 2 Threads and 4 Threads on Redis

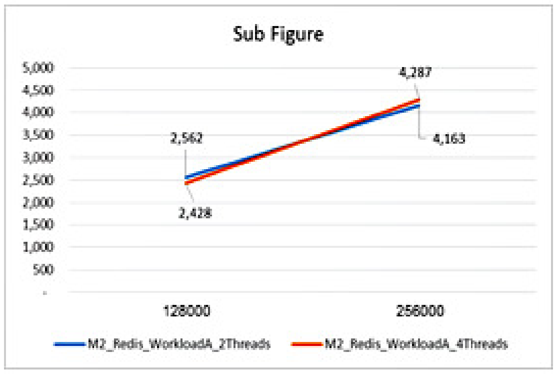


Figure 7: Run Time on Workloada and Redis

5.2 Throughput

As shown in Figure 8, in Cassandra throughput increases in Machine3 by increasing the number of threads, as expected. This event is the same as for WorkloadF. But as shown in Figure 9 and Figure 10, Redis has a threshold on 8 threads so that not only more throughputs cannot be seen, but also, number of records was decreased in most cases. The few cases where the number of 16 threads is better than 8 threads are on 64000 records that it can be seen in Figure 9. However, the difference is very low. So it can be seen that on Redis increasing of throughput is not equal with increasing of speed. In Figure 11 and Figure 12, we compared Cassandra and Redis on the best number of threads states, i.e. 8 threads and 16 threads. As can be seen Redis on WorkloadA has 2.18 times superiority rather than Cassandra, and on WorkloadF Redis has 2.30 times performance than Cassandra. This event on 8 threads has greater superiority factor. Superiority factor of the performance in WorkloadA is 3.46 and in WorkloadF are 4.26.

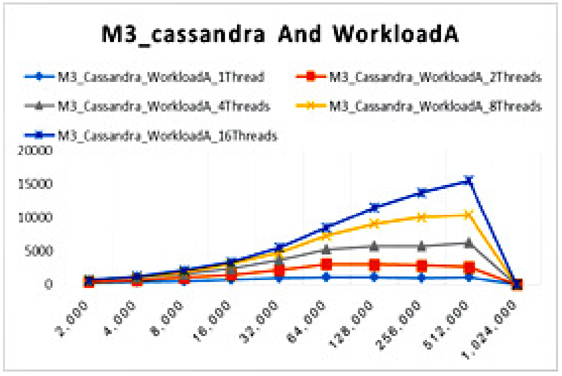


Figure 8: Throughput on Workloada and Cassandra

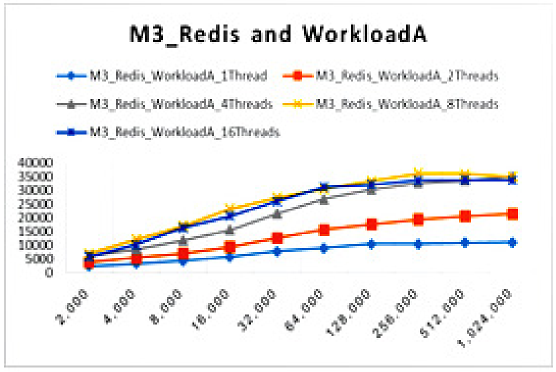


Figure 9: Throughput on Workloada and Redis

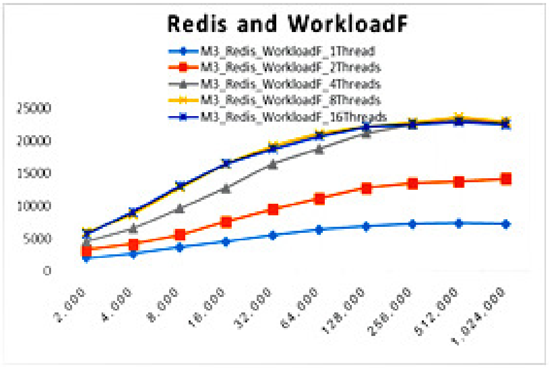


Figure 10: Throughput on Workloadf and Redis

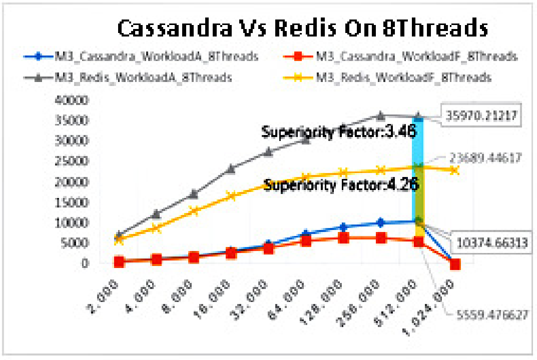


Figure 11: Comparison of Cassandra and Redis on 8 Threads

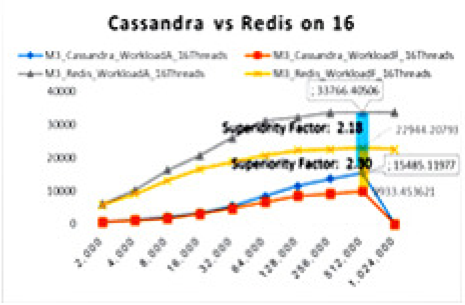


Figure 12: Comparing Cassandra and Redis On 8 Threads

5.3 Update, Write, Read Latency Average

In Figure 13, we presented operations average latency with 256000 records on different threads and WorkloadA. In this test, the average read latency has 825µs threshold on 4 threads. The number of threads starts from 1 and increases to 16 threads using the general rule 2^i.
First, the average latency is started with 1449µs on 1 thread and continues to 4 threads descending, but after that this event to 16 threads is ascending. So based on our use case statistical population, we should consider a trade-off between number of requests and number of users. The best case for such a statistical population will be 4 threads for a node like Machine3 and on 256000 records.
Same situation can be seen in Figure 14 which depicts the average latency for different thread numbers in WorkloadF. However, the best case can be seen for 2 threads. This situation can be seen for Update and Modify too.

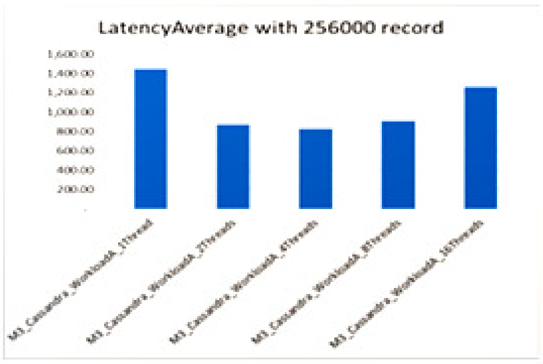


Figure13: Average Latency on Cassandra and Workloada

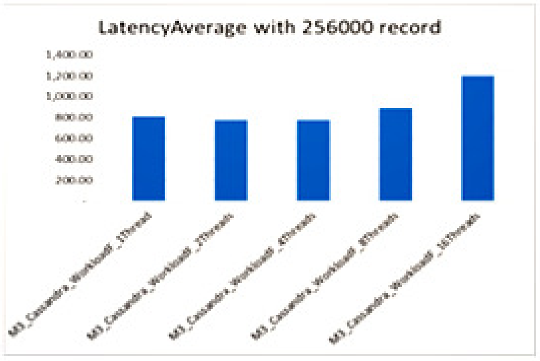


Figure14: Average Latency on Cassandra and Workloadf

While as shown in Figure 15 to 18, unexpectedly on Redis average latency is ascending and the best state is seen in 1 thread continuously.

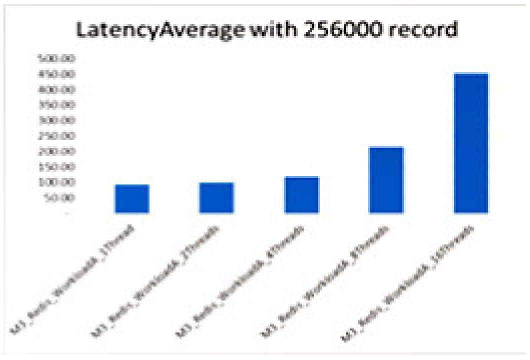


Figure 15: Average Latency on Redis and Workloada

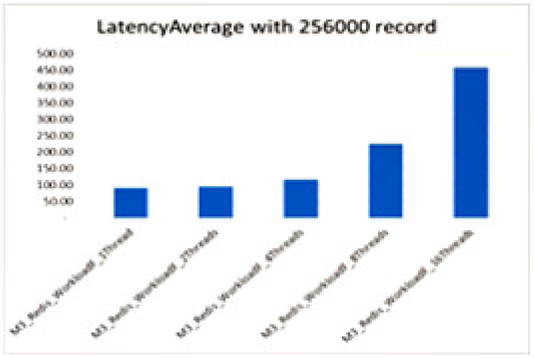


Figure 16: Average Latency on Redis and Workloadf

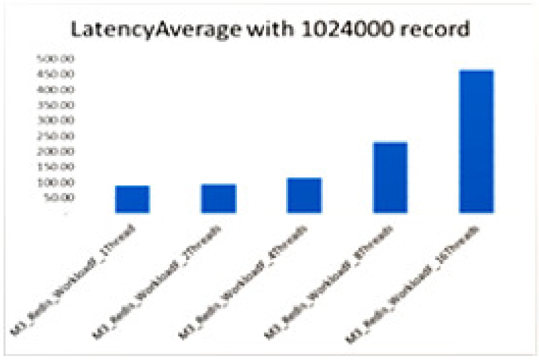


Figure 17: Average Latency on Redis and Workloada

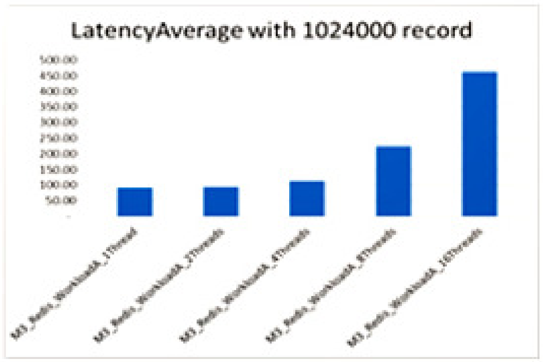
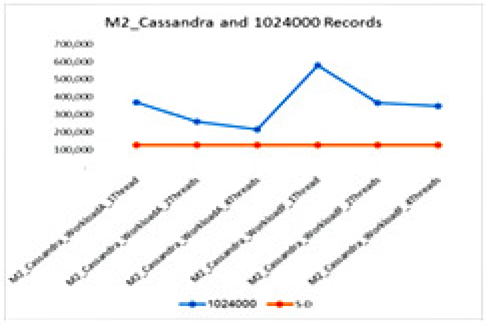


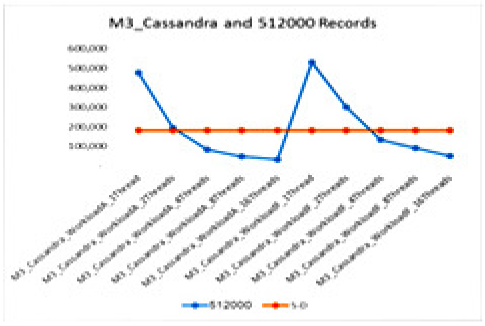
Figure 18: Average Latency on Redis and Workloadf

5.4 Diffusion Index Points On Run Time

Diffusion index indicates increase the number of threads how much will have impact on run time. The following charts display the impact of increasing number of users over the same of records. More difference between diffusion points and diffusion index indicates lower impact of increasing number of users (threads). Generally, can be said: Impact factor of increasing number of threads = the role of users in determining run time. Diffusion index on Cassandra and Machine2 and on both of Workloads is shown in Figure 19(a). Also Diffusion index on Redis and on both of Workloads is shown on Figure 19(b). In the evaluation process, we display diffusion of diffusion index on Machine3 on Figure 20. As can be seen, all diffusion index points (with the exception of one point) are on top and far from the diffusion index. Because of the more number of threads on Machine3, this event is noticeable. So that, on Cassandra only 3 points (1thread states on both of Workloads and 2 threads states on WorkloadF) are on the top of the diffusion index. That means run time of only 3 points is higher than value of diffusion index.

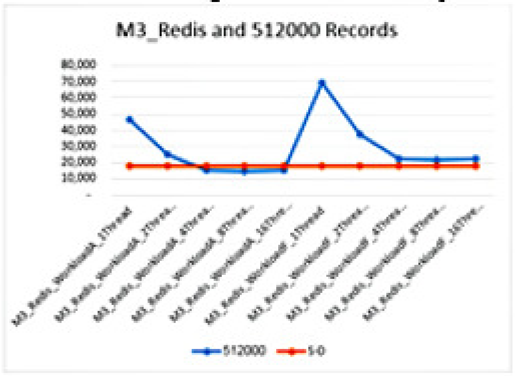


(a)

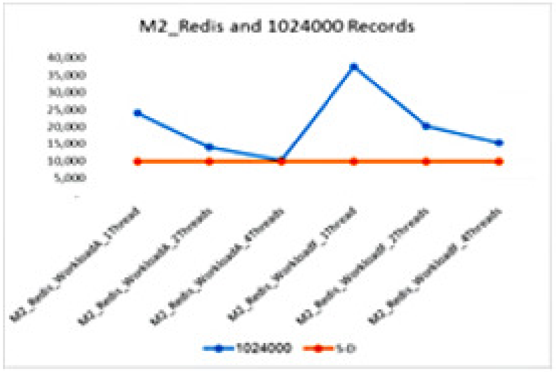


(b)

Figure 19: Diffusion Index Points on Both of Workloads and Cassandra



(a)



(b)

Figure 20: Diffusion Index Points on Both of Workloads and Redis

At the same time, run time of 7 points have the same value with diffusion index or have less than diffusion index. Due to existing of appropriate resources on this machine, the most of the distribution points can be seen around the diffusion index. This event shows high impact of the increasing number of threads on the performance. So that, Redis has lower dispersion because of being close to the diffusion points in Casandra.

6. FUTURE WORKS

According to variety of factors in databases evaluation, conducting more tests reveals more aspects so that developers would encourage experiencing more situations. The more awareness about NoSQL databases, the better evaluation process. So, more knowledge about the evaluation process leads to the better results. Because of running Operation System's tasks, it is better that evaluation process to be done in different time slices.
Furthermore, evaluation can be performed in various models of Sharding techniques to get closer to real world situation. Also, Chi-square can be used in the evaluation process. On the other hand, to evaluate the maximum power of processor to process records, processor over clock can be considered.

7. CONCLUSION

Conducted evaluation has been done in four steps including run time, average latency, throughputs, and diffusion index. Regarding the run time, start of effectiveness was from 128000 records in WorkloadA in Cassandra. So that the best performance is on 4 threads and the worst performance is on 1 thread. In throughput aspect on Cassandra and Machine1 increasing the throughput against the number of records is irregular. While on Redis this process is regular and there is the better performance of Redis on WorkloadA with 2.18-fold of superiority factor on 16threads and 3.46-fold on 8threads.Also this process and performance is correct in WorkloadA. So that, this database has 2.30-fold of superiority factor on 16threads and 4.26-fold on 8threads. In this evaluation, with WorkloadA, 4threads and Machine3 the average read latency has 825ms threshold. So, we should do trade-off between the number of requests and the number of users.
The best state on Redis is 1thread. The reason of this event is In-Memory feature on Redis. It is mainly because switching between threads reduces performance. In this evaluation, because of having less and lower resources in Machine 2, all points of diffusion index (exception one point) are above and far from diffusion index. Results on this event due to existence of appropriate resources and more number of threads in Machine3 are more significant. In Cassandra only 3 points are higher than value of diffusion index. While run time of 7 points have the same value with diffusion index or have less than diffusion index. This event reflects the high impact of increasing the number of threads in the performance in this machine.

REFRENCES:

[1] Y.L. Choi, W.S. Jeon, and S.H. Yoon, "Improving Database System Performance by Applying NoSQL", *JIPS*, Vol. 10, No. 3,2014 pp.355-364.
[2] A. Lith, and J. Mattsson, "Investigating storage solutions for large data-A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data", *Department of Computer Science and Engineering CHALMERS UNIVERSITY OF TECHNOLOGY Goteborg Sweden*, Master of Science Thesis, 2010.
[3] S.K. Gajendran, "A survey on nosql databases", *University of Illinois,* ebook. 2012
[4] K. Grolinger, W.A. Higashino, A. Tiwari, and M.A. Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores", *Journal of Cloud Computing: Advances, Systems and Applications*, Vol. 2, No. 1, 2013, pp.1.
[5] Data Engineering [in Persian]. (2016), www.bigdata.ir, last accessed June 2016.
[6] M.A. Olson, "Selecting and implementing an embedded database system. Computer", *IEEE Computer Society Press Los Alamitos*, Vol. 33, No. 9, 2000, pp.27-34.

[7] Y. Abubakar, T.S. Adeyi, and I.G. Auta, "Performance evaluation of nosql systems using ycsb in a resource austere environment", *ijais*, Vol. 7, No. 8, 2014, pp.23-27.
[8] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, "November. Security issues in nosql databases", *International Conference on Trust, Security and Privacy in Computing and Communications*, 2011, pp. 541-547.
[9] C. Mohan, "History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla", *International Conference on Extending Database Technology*, 2013, pp. 11-16.
[10] J.E. Pagán, J.S. Cuadrado, and J.G. Molina, "Morsa: A scalable approach for persiscesting and accessing large models", *International Conference on Model Driven Engineering Languages and Systems*, Vol. 6981, No. 0302-9743, 2011, pp. 77-92.
[11] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data", *arXiv,* preprint arXiv:1301.0159.

Journal of Theoretical and Applied Information Technology
31st March 2017. Vol.95. No 6
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

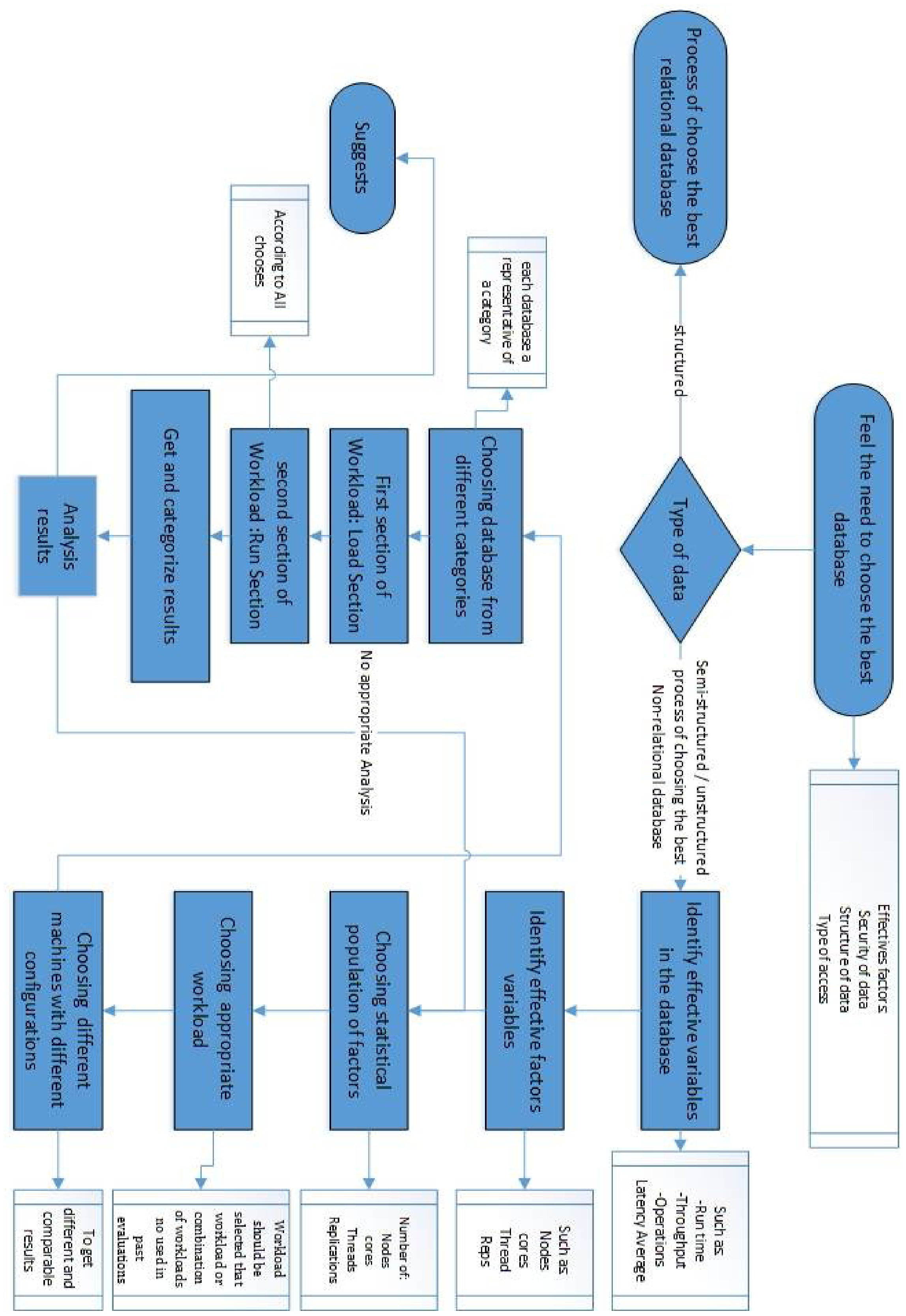

Feel the need to choose the best database
Effectives factors
Security of data
Structure of data
Type of access
Type of data
structured
Process of choose the best relational database
Semi-structured / unstructured
process of choosing the best
Non-relational database
Identify effective variables in the database
Such as
-Run time
-Throughput
-Operations
Latency Average
Identify effective factors variables
Such as
Nodes
cores
Thread
Reps
Choosing statistical population of factors
Number of:
Nodes
cores
Threads
Replications
Choosing appropriate workload
Workload should be selected that workload or combination of workloads no used in past evaluations
Choosing different machines with different configurations
To get different and comparable results
Choosing database from different categories
each database a representative of a category
First section of Workload: Load Section
second section of Workload :Run Section
According to All chooses
Get and categorize results
Analysis results
No appropriate Analysis
Suggests

Figure 21: Method Of Evaluation Of Nosql Databases