ISSN:
Print - 2277 - 0593
Online - 2315 - 7461

Journal of Natural Science, Engineering and Technology

A SURVIVABLE DISTRIBUTED DATABASE AGAINST BYZANTINE FAILURE

***[1]D. O. ABORISADE, A. S. SODIYA[1], A. A. ODUMOSU[2], O. Y. ALOWOSILE[2] AND A. A. ADEDEJI[2]**

[1] Department of Computer Science, Federal University of Agriculture Abeokuta,, Nigeria.
[2]Department of Computer Science, Abraham Adesanya Polytechnic, Ijebu-Igbo, Ogun State, Nigeria.
***Corresponding author:** aborisadeda@funaab.edu.ng, **Tel:** +2348056535109

ABSTRACT

Distributed Database Systems have been very useful technologies in making a wide range of information available to users across the World. However, there are now growing security concerns, arising from the use of distributed systems, particularly the ones attached to critical systems. More than ever before, data in distributed databases are more susceptible to attacks, failures or accidents owing to advanced knowledge explosions in network and database technologies. The imperfection of the existing security mechanisms coupled with the heightened and growing concerns for intrusion, attack, compromise or even failure owing to Byzantine failure are also contributing factors. The importance of survivable distributed databases in the face of byzantine failure, to other emerging technologies is the motivation for this research. Furthermore, It has been observed that most of the existing works on distributed database only dwelled on maintaining data integrity and availability in the face of attack. There exist few on availability or survibability of distributed databases owing to internal factors such as internal sabotage or storage defects. In this paper, an architecture for entrenching survivability of Distributed Databases occasioned by Byzantine failures is proposed. The proposed architecture concept is based on re-creating data on failing database server based on a set threshold value.The proposed architecture is tested and found to be capable of improving probability of survivability in distributed database where it is implemented to 99.6% from 99.2%.

Keywords: Availability, Byzantine failure, Distributed Databases, Survivable and Survivability

INTRODUCTION

Adequate provision of access to information for users in their different forms has always been a great challenge to designers and providers of information repositories. One common example of such repository is the distributed database system. A distributed database is a database where every constituent database (relational and non-relational) runs on different structures (schema) of different Database Management System (DBMS). (Thandar *et al.,* 2008). It is also a database in which both the data and the DBMS are all across multiple computers. The continuous growth witnessed in Database Management System (DBMS) and Network Communication technologies are responsible for the increase in available distributed databases and Distributed Management System (DMS). Just like the Internet, distributed database technology has been useful in providing online access to different

types of information for users across the World. With the rapid improvement on traditional databases and distributed technology, the research and development of the distributed database systems, with data distributed storage and distributed processing as main features, are receiving increasing attention (Thandar *et al.,* 2008). Moreover, the growth in complementary role played by distributed database systems to the Internet in terms of its usefulness in making real time information available to users is however, being faced with certain challenges, such as problem of real time access to information owing to internal and external attacks or failures. Survivability is the capability of a system to fulfill its mission in a timely manner, in the presence of threats such as attacks or large scale natural disaster (James et al., 2010). A survivable system is a system that is capable of fulfilling its services in the face of any attack or failure (James et al., 2010). These attacks and failures are usually intended to make distributed databases fail, rendering them useless and incapable of fulfilling their expected services. Furthermore, failures of databases have been recently observed to be on the increase, owing to internal attacks or failure. Although several research efforts have been made to ensure that database systems are made to survive some of these attacks or failures, most of these research efforts give attention to survivability of general storage and database systems, with little or no attention to survivability of distributed databases (Zhang *et al.,* 2010; Prem *et al.,* 2000). Some of these researches on survivability of distributed database only succeeded in proposing virtualization ideas that has resulted in redundancy problem (Thandar *et al.,* 2008; Changqing *et al.* 2011). This was with the notion that the most common and deadly attack are usually not caused by byzantine failures. Byzantine failures are failures occasioned by internal factors like human sabotage, storage and storage system defects in a system. In addition, the growing demand of modern day information access through distributed database, coupled with its attendant challenges necessitated this research effort. In this paper, an architecture to ensure survivability of distributed database is proposed. The rest of this paper is organized as follows; Section two discusses the related work. Section 3 presents the materials and methods; Section 4 discusses results while section 5 concludes the paper.

RELATED WORK

A number of recent literatures relating to survivability of distributed database, information and storage systems were reviewed. Some of these include James *et al.* (2010) which provided an architectural framework or resilience and survivability in communication networks and also provided a survey of the disciplines that resilience encompasses, along with significant past failures of the network infrastructure. According to James *et al.* (2010), a resilience strategy to defend against, detect and remediate challenges is presented. Arora and Gupta (2012) highlighted the common characteristics of cloud databases in the industry and discussed the various descriptions of Cloud databases. An overview of common types of databases, major challenges to develop cloud databases, and summary of the existing cloud databases was also discussed. Their research effort gave a good account of NoSQL cloud database but did not discuss anything about NoSQL cloud database security. Changqing *et al.* (2011) introduced a survivability framework for distributed systems through the use of virtualization technology and software rejuvenation methodology. They also presented a recovery model and evaluate the steady-state sys-

tem availability and survivability based on the familiar Markovian analysis through SHAPE tools. However, their software rejuvenation efforts does not prevent removal of faults but rather prevented them from manifesting themselves as unpredictable whole system failure. Chao-Rui *et al.* (2012) presented a method to measure the survivability of object-oriented software in design phase. In this, each component of the object oriented software is characterized by a composite Petri net which combines the features of State chart and Object Diagram. A fuzzy number is introduced to this net to represent uncertain elements that might affect the survivability. Survival Possibility theory was used to produce survivability measure function for each component. A survivability measure index is defined for the system. They could only prove that this index is monotonic. Jueliang *et al.* (2009) proposed a survivability evaluation model and analysis performance of Wireless Sensor Networks (WSN). They presented the model by representing the states of Wireless Sensor Networks under attack. The survivability of WSN is expressed as a continuous time Markov Chain to describe the status of real WSN's in the face of Dos attack. To achieve this, they proposed a threshold condition to trigger the transition between the states of the Wireless Sensor Network. They further presented a survivability model and evaluation of WSN under key compromise. Their study however did not include key revocation scheme and single node software rejuvenation and reconfiguration scheme for mission critical operations, based on self-healing concept. Li, Saroj and Frank (2010) proposed and developed a dependable and secured relational database management system to prevent sensitive information from being lost or stolen. The basic idea used in their work is to include a (k, n) threshold-based secret sharing scheme ($k \leq n$) to provide privacy and durability in order to prevent loss of sensitive information and also incorporated an efficient distributed database management design to enhance system performance and minimize interfered accesses contentions. They also integrated private information storage (PIS) schemes to reduce communication overhead and improve robustness of their proposed system. Zhang, Liu, and Zhong (2010) introduced an approach by which an untrusted publisher can answer queries from customers on behalf of the owner, or creator of the data with a few trusted digital signatures from the owner, the untrusted publisher can use Merkle hash tree –based techniques to provide authenticity and non-repudiation of the answer to a database query. According to Thandar *et al.* (2008) a new distributed B-tree column indexing scheme was devised. It was devised to support indexing for non-row key columns, as well as fast parallel B-tree search in large data table. They also developed a cloud database, called HSQL, based on HBase. HSQL enhances HBase with a SQL query interface, a distributed B-tree column indexing, and support for transactional data processing. Experiment results show that their B-tree index achieves good response time even with large data and large number of queries. Implementation showed that HSQL has very short response time for point query compared with HBase scan filtering. HSQL achieves an average of 6.2X speed up for range query and an average of 12.2X speed up for aggregation. However, the newly developed distributed B-tree column indexing scheme is not made generic, as it is only applicable to HBase database (Shengzhi and Peng, 2010). Thandar *et al.* (2008) presented a generic architecture of survivable storage system. The architecture is made up of three

modules namely the storage strategy, user management, and security module. The system architecture proposed is observed to be capable of access control and system self-adaptation. Yunghsiang, Soji, and Rong (2010) proposed a storage-optimal and computation efficient primitive to spread information from a single data source to a set of storage nodes which allow recovery from both crash-stop and Byzantine failures. A progressive data from retrieval scheme is employed to retrieve minimal amount of data from live storage nodes. The scheme adapts the cost of successful data retrieval to the degree of errors in the system. Sazia *et al.* (2012) presented a survivability evaluation model for real time database, by considering factors such as intrusion detection latency and a variety of parameters for real time, thereby culminating in Semi-Markov evaluation model for survival assessment. In their work, this model served as the basis for relevant quantitative criteria to be defined as the important indicators of survivability, such as integrity and availability. Three important factors like false alarm, detection rate, and the intensity of the attack are analyzed in detail using the TPC-C benchmark. A survivability information system based on service self-organization under the precondition of modules redundancy backups (Yongshi *et al,,* 2012). The service self-organization way divides the system into communications modules, services distribution modules, data storage modules according to service processing flow. Through their simulation experiment, they were able to prove that their proposed method can enhance the service survivability both of steady state and instantaneous state by setting up a Web server in the network environment. The survivability of data centers is paramount to the survivability of the whole enterprise computing system (Prem *et al.,* 2000). Prem *et al.* (2000) discussed some set of challenges they viewed as critical in building highly survivable data centers, and presented several potential solutions to tackle them.

MATERIALS AND METHODS

Proposed Survivability Architecture: The proposed architecture for the survivable distributed database is as shown in Figure 1. The architecture is made of three (3) major modules namely the Normal/Degrading, module, the Monitor module and the Rejuvenated module.

***Normal/Degrading*:** The normal module is basically made up of the client, the database server, and the application server. It is the module representing the regular distributed database communications between the clients, the database server and the application server. In this module, application server provides middleware security, maintenance, data access and persistence services between the distributed system and the user's applications.

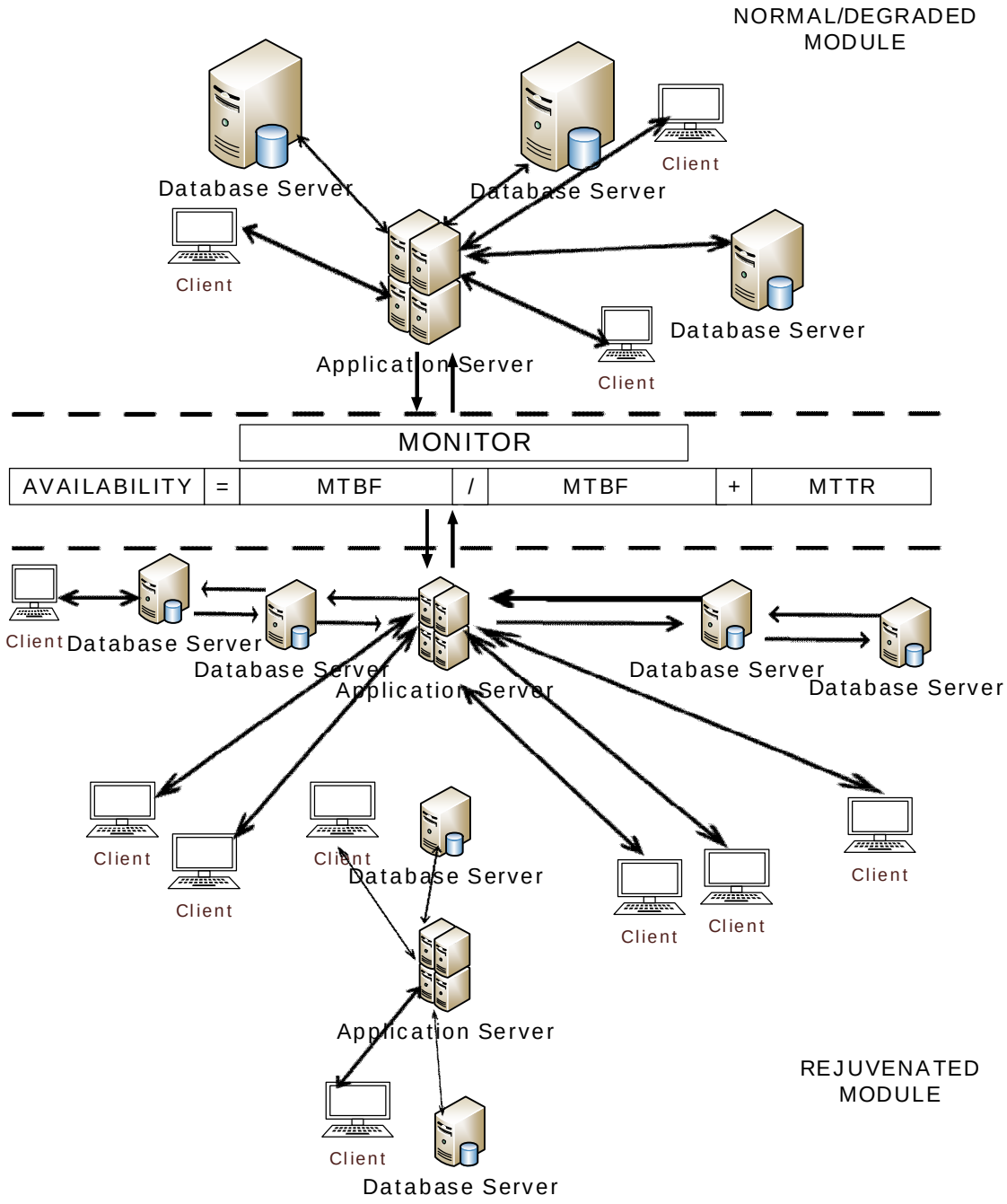

Figure 1. Proposed Survivability Architecture

Database server in this module provides database services to other computer programs or computers or users. They are dedicated computers used to run Database Management Systems (DBMS) programs. The database server program used in this work is MySQL. The client in the architecture depicts the many users that request for distributed database services. This module also depicts state of the distributed system when it is just moving from normal state to a degrading state owing to byzantine failure.

***Monitor Module*:** The monitor module is the component of the architecture where the the regular distributed systems activities in the normal/degrading module is being monitored for possible failing components of the system. This is done by constantly monitoring availability of important storage components of the storage systems such as the disks.

***Definition 1*: A**vailability is the measure of capability of a system to render its expected services at all times. It is important that the proposed model ensure unrestricted and uninterrupted access to the cloud database at all times. Availability is an important attribute of survivability. Availability is measured here with metrics such as Mean Time between Failure (MTBF) and Mean Time to Repair (MTTR). Data about the manufacturing date and expiring date of these devices are taken and used as input for MTBF and MTTR to calculate their availability in the system. The measure of availability for each of the internal devices in the distributed databases are regularly shared between the application server and the monitor.

***Rejuvenated Module*:** The third module is the rejuvenated module. It is made up of regenerated database server and application server. It is the module where the system is made stable and survivable.

Survivability Design Concept

In the regular distributed database activities (i.e. in normal/degrading module), the client or user's application program requests for a distributed database service through the application server. Database server provides database transaction services to the clients. Data about the manufacturing date and expiring date of storage devices are taken and used as input for MTBF and MTTR to calculate their availability in the system. The application server calculates the measure of availability for the storage devices in all the database servers involved in the connection, and shares with the monitor using

$$AVAILABILITY = MTBF / MTBF+MTTR \quad (1)$$

These parameters are based on the life span of the different storage devices that make up the distributed database system. The average value of all the measures of availability from all the database servers in the distributed database system is used as the threshold value for determining the system overall byzantine failure. The moment the calculated measure of availability from any of the database server or application server is less than the average value of availability from all the database servers in the system, byzantine failure is suspected, the rejuvenation mechanism is invoked. In rejuvenation mechanism, the data in all the database servers where the measure of availability is less than the average calculated measure of availability are copied to the database server with the highest measure of availability. This process continues until there is no more database server whose measure of availability is less than the calculated average measure of survivability, result-

ing into the rejuvenated module of the system. The survivability of distributed database is therefore modeled mathematically as follows. Since in regular distributed database activities, every database server is designed to exhibit capability to render all its services through their application server. According to the architecture, for every service S there exist database servers N and application servers VM's ranging from 1 to n, i.e If " service S, $ database servers N and several VMs. Clearly, a relationship exists between a service (cS) and its database servers (cN), consequently T(cS, 1)=1 for cS ³ 1. Then if cS ³ cN it is observed that the size of the range which is an integer from 1 to n-1, and the elements of the range which can be combined in C(cN, k) ways and picking an onto the range of function that is not onto from the set with cS elements to the set with cN elements which can be combined in T(cS, k) ways. Therefore, it is suffice to say that the recurrence relation between the clients requesting services and the application servers entrenching availability in the proposed architecture system is

$$T(cS,cN)=\sum_{K=1}^{n-1}C(cN,k)S(cS) \qquad (2)$$

where cS=cN and cN>1 with initial condition T(cS, 1)=1.

Proposed Survivability Algorithm

Input: ManDate, ExpirDate, MTBF, MTTR
Output: calcAvail, dBserverAvail
Initialization: AppServer ¬ ManDate{ }
AppServer ¬ ExpDate{ }
AppServer ¬ MTBF{ }
AppServer ¬ MTTR{ }
Step1: ManDate ¬ ManDatevalue;
Step2: ExpDate ¬ ExpDatevalue;
Step3:MTBF¬ExpDatevalue-anDatevalue;
Step4:While(calcAvail>dBserverAvail)
calcAvail= MTBF/MTBF+MTTR
DbServerDat¬ AppServerDat
AppServerDat¬DBserverDat
If (dBserverAvail< calcAvail)
MaxAvail ¬ MindBData
Endif
Step5: End

RESULTS

For the purpose of implementing and evaluating the proposed architecture, a Java Application was developed to simulate an application server that executes and provides users access to two sets of distributed databases. The two distributed databases were designed with highly defective storage disks, but one of the two distributed databases (called the Enhanced database) also was designed its application server based on the proposed architecture design incorporated into it, while the second distributed database was designed without the proposed architecture integrated into it Java application server. The capability for the survivability of the two distributed databases was tested by making simple query call services on them from a number of clients. These transactional activities were allowed to run for a period of twelve (12) hours. Results of observations from the experiments presented in terms of measures of availability of the Distributed Database Components (DDB components) that constitute the application servers and database servers in the two distributed databases. The distributed database without the proposed scheme is called Regular DDB, while the distributed database with the proposed scheme integrated is called the Enhanced DDB as shown in Table I (Regular) and Table II (Enhanced) respectively.

Table 1. Measures of Availability in Regular DDB

DDB Components	MTBF (min)	MTTR (min)	Availability
Cooling Fan	250020	2880	0.989
Processor	175200	1440	0.992
Storage Disk	199980	2880	0.986
Power Pack	262800	2880	0.989

Table 2. Measures of Availability in Enhanced DDB

DDB Components	MTBF (min)	MTTR (min)	Availability
Cooling Fan	740060	2880	0.996
Processor	262800	1440	0.995
Storage Disk	499980	2880	0.994
Power Pack	525600	2880	0.995

DISCUSSION

Results in Table I show the measures of availability of the four (4) distributed database (DB) components in the database server for the regular distributed databases in the experiment. While cooling fan presents availability of 0.989, processor gives availability 0.992, storage disk has availability of 0.986, and power pack has availability of 0.989. This gives an average availability of 0.989 from all database servers in the regular distributed databases. Results in Table II also give the measures of availability of the four (4) distributed database (DB) components in the database server for the Enhanced distributed databases in the experiment. From cooling fan we have 0.996, from processor availability is 0.995, from storage disk we had availability of 0.994 and from power pack we got availability of 0.995. All these give an average availability of 0.995. Fig. 2 and Fig. 3 give the graph analysis of the results.

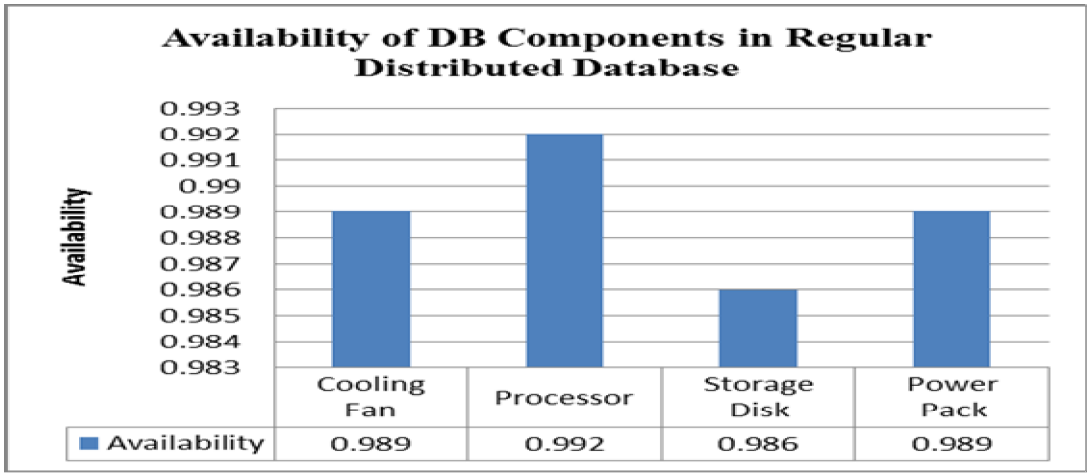

Figure 2. Graph analysis of results in Regular DB

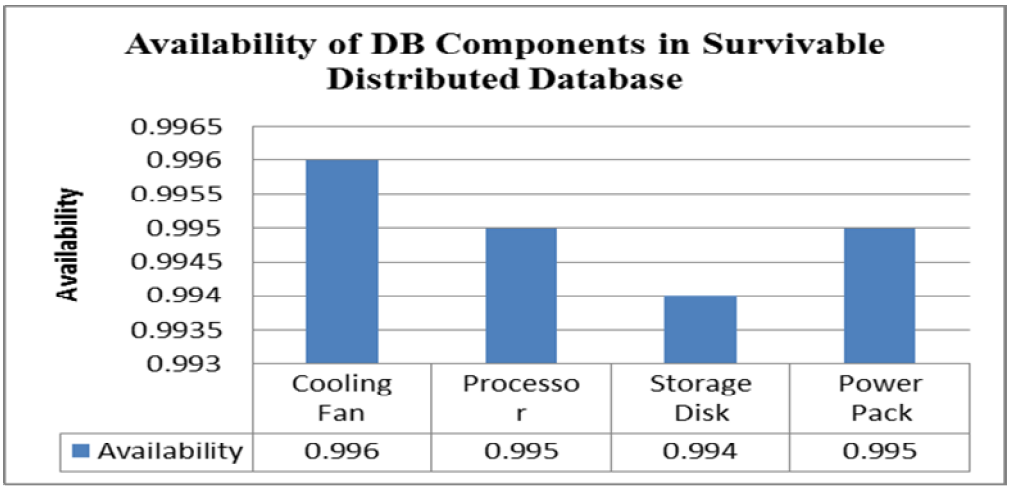

Figure 3. Graph analysis of results in Survivable DB

Results in Tables I and II further show that with the integration of the proposed scheme into a hitherto regular distributed database, its availability could increase from an average of 0.989 to 0.995. This shows that the probability of survivability is higher in distributed database in which the architectural idea proposed in this work is incorporated as further depicted in the Figs. 4 and 5.

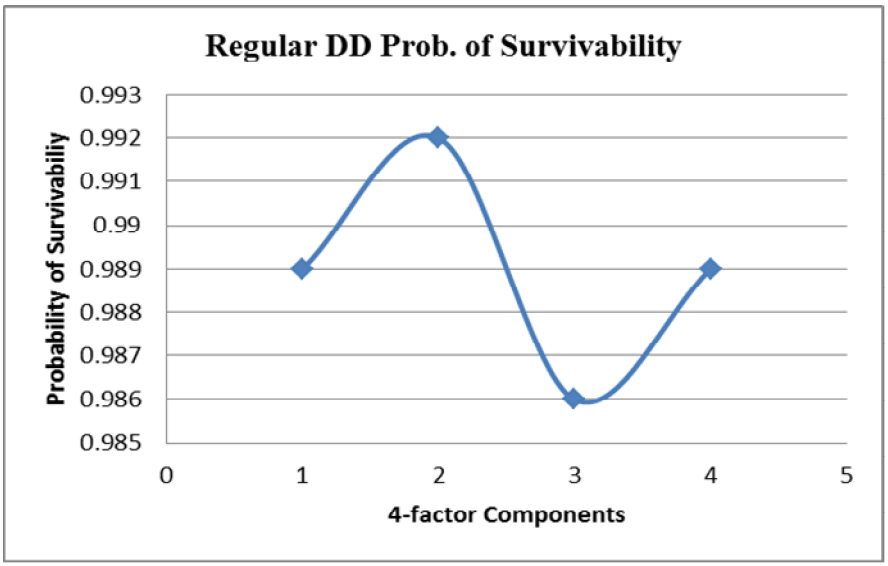

Figure 4. Probability of Survivability in Regular Distributed Database (DB)

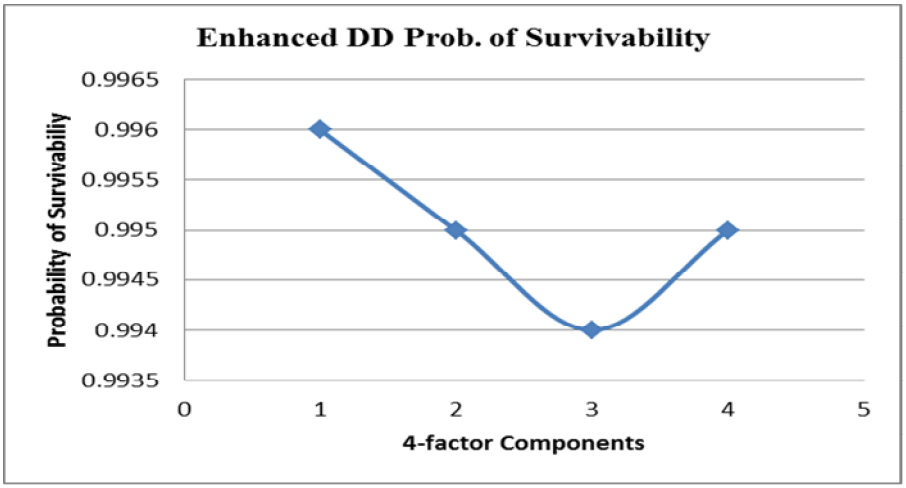

Figure 5. Probability of Survivability in Enhanced Distributed Database (DB)

CONCLUSION

The trend in the world points to the fact that distributed database and cloud computing are the technologies to support future computing, considering the volume and varieties of data that are accessible to people through computers and mobile devices. As more and more people migrate their data and infrastructures to the cloud, more reliable distributed database technology would be needed to power the cloud. This of course would come with its attendant security challenges, one of which this research effort responded to by proposing a survivable architecture in distributed databases. This study established that the present level of fault tolerance in distributed database could be improved upon.

REFERENCES

Arora, I., Gupta, A. 2012. Cloud Databases: A Paradigm Shift in Databases. *International Journal of Computer Science IJCSI* 9 (4, 3): 77-83.

Changqing, C., Heng Z., Weimin W., Gang S. 2011. A Semi-Markov Survivability Evaluation Model for Intrusion Tolerant Real-Time Database Systems. IEEE (2011).

Chao-Rui, C., Meng-Ju, H., Jan-Jan, W., Po-Yen, W., Pangfeng, L. 2012. HSQL: A Highly Scalable Cloud Database for Multi -User Query Processing. *In: 2012 IEEE Fifth International Conference on Cloud Computing*. pp. 943-944 IEEE Computer Society Washington, DC, USA 2012.

James P.G.S., David H., Egermen K. C., Abdul J., Justin P.R., Marcus S., Paul S. 2010. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. Computer Networks 54(2010) 1245-1265. Elsevier B.V. 2010.

Jueliang H., Zuohua D., Jing L., Ling Y. 2009. Measuring the Survivability of Object-Oriented Software. In: TASE '09 Proceedings of the 2009 *Third IEEE International Symposium on Theoretical Aspects of Software Engineering*. Pages 329-330. IEEE Computer Society Washington, DC, USA.

Li B., Saroj B., Frank F. 2010. Design of a Reliable Distributed Secure Database System. *In: 2010 Fifth IEEE International Conference in Networking*, Architecture, and Storage, pp 91-99. IEEE Press 2010.

Prem, D., Michael G., Chip M., Philip R., Stuart G. S. 2000. Authentic Re-Publication by Untrusted Servers: A Novel Approach to Database Survivability. Citeseer.

Sazia, P., Farookh, K.H., Jong S.P., and Dong S. K. 2012. A survivability model in Wireless Sensor Network. Computer and Mathematics with Application 2012. Elsevier Ltd.

Shengzhi, Z., Xi X., Peng, L. 2010. Challenges in Improving the Survivability of Data Centers. In: Workshop on Survivability in Cyberspace 2010.

Thandar T., Manish P., Sung-Do C., and Jong S. P. 2008. A recovery Model for Survivable Distributed Systems through the use of Virtualization. *In: Fourth International Conference on Networked Computing and Advanced Information Management. IEEE* Computer Society. 2008.

Yunghsiang S.H., Soji O., Rong Z. 2010. Survivable Distributed Storage with Progres-

sive Decoding. *In: IEEE INFOCOM 2010 mini conference proceedings*. IEEE 2010.

Yongshi Z., Jianpei Z., Lejun Z., Mo L., Jing Y., Lin G. 2010. A Survivability Information System Based on Service Self-Organization. *In: 2010 Fifth International Conference on Internet Computing for Science and Engineering*. pp 158-161. IEEE Computer Society.

Zhang W., Liu S., Zhong W. 2010. Survivable Storage Architecture. *In: Third International Symposium on Information Processing.* Pp. 95-97. IEEE Computer Society 2010.

(Manuscript received: 11th February, 2016; accepted: 15th June, 2016).