Enhancing Availability of Marine Bigdata Repository with a New Fault Tolerance Technique

Ahmad Shukri Mohd Noor , Farizah Yunus, Rabiei Mamat, Emma A. Sirajuddin and Nur F. Mat Zin
School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu,
21030 Kuala Nerus, Terengganu, Malaysia.
ashukri@umt.edu.my

***Abstract*—System availability is one of the crucial properties of a dependable knowledge repository system in order to preserve and pull through from minor outages in a short timespan by an automated process. National Marine Bioinformatics System or NABTICS is a Marine Microbial Bigdata Repository that unites the integrated information on genomic sequence and associated metadata which projected to be a large and growing database as well as a metadata system for inputs of research analysis and solving community issues. Therefore, it is decisive to maintain the availability of the system by accurately detecting the failure in a timely manner and a prompt recovery action during the event of failure. The failure in any of NABTICS' system component can be devastating for the system causing the system is inaccessible for a period of time. In this paper, we integrated NABTICS with Cloud-based Neighbour Replication and Failure Recovery (NRFR) in order to enhance the availability of the system. We showed that the implementation resulted in better user experience with minimum system downtime as well as online database application is said to be highly available. Furthermore, NABTICS also performed better resource utilization and higher response application during runtime.**

***Index Terms*—Availability; Bigdata; Database Replication; Distributed System.**

I. Introduction

National Marine Bioinformatics System (NABTICS) is a Marine Bigdata Repository initiative by Institute of Marine Biotechnology, University Malaysia Terengganu (UMT), with the goal to serve the needs of the marine ecology research community by creating a rich, distinctive data repository and bioinformatics tools that specifically focuses on the inventory of marine organisms. Its application also includes metadata system where the datasheets would allow researchers to input results from computer-based data analysis.

A system of high availability is a system that is designed to avoid loss of service. In computing, such system is achieved by having several copies of files and databases on multiple machines. Site replication gives very high availability as it masks environmental failures, hardware failures, operator error and even some software faults [1]. In this particular technique, neighbor replication will be employed where neighboring servers or nodes maintain a replica of a primary file and database. Prior to this, an application is spread across the multiple nodes thus fulfilling the concept of distributed system.

On top of this, a monitor is also deployed to continuously check the aliveness of the nodes called Heartbeat Monitor (HBM). It constantly pings the servers and invokes recovery action to an Index Server (IS) in the event of a confirmed failure of a node via SSH command. Consequently the IS will perform a neighbor selection process to determine which replica should take over the failed node. It then invokes the selected node that contains the replica to activate Virtual IP. The user's client is then redirected to this replica without any other intervention.

An available system is desirable in critical and heavy usage setting. Providing availability to NABTICS can help researchers have convenience in doing their work and it adds to the system itself in terms of superiority, usability and consistency. As a result, NABTICS could also be a reliable system that is almost fault free.

II. Related Works

Adding availability to a system is similar to adding more reliability where an available system is guaranteed to be reliable while a reliable system may not necessarily be available. Many different works have also considered availability as an attribute of dependability. By adding the notion of recovery, system availability is branched from the concept of system reliability. According to Gokhale et al. [2], a crucial difference between reliability and availability is that reliability refers to failure-free operation during an entire interval, while availability refers to failure-free operation at a given instant of time.

A prompt and accurate failure detection with minimum time to recover is a critical factor in providing high availability in distributed system. Failure detection and recovery has become an active research area due to the needs that arise with more and more complex system. Failures are arising due to the inherently unreliable nature of distributed environment. The diverse nature of Grid Computing for example, requires the inclusion of fault tolerance capability not as a supplementary feature but rather a prerequisite [3]. As for cloud computing, it has become a critical issue due to its nature of complex multi layers and the most of current approaches, fault tolerance is exclusively handled by the provider or the customer which leads to partial or inefficient solutions [4].

A. Failure Detection

Failure detection is a process in which information about faulty nodes is collected [5]. This process involves isolation and declaration of a fault to enable proper recovery actions to start. It is a prerequisite to failure recovery in distributed systems. The properties of failure detection technique are completeness and accuracy. A third property was added by Stelling et. al [6] that is timeliness. Timeliness refers to how

fast a fault can be detected in complete and accurate manner so that corrective action can be initiated soon. The Quality of Service for failure detectors are detection time, mistake recurrence time and mistake duration. These calculations can also be used to measure availability of a system. Two implementations of failure detection are discussed that are Adaptive Affirmative Failure Detection (AAFD) and Gossip Enabled Monitoring System (GEMS) for gossip-style failure detection.

In [7], an adaptive technique for failure detection AAFD was introduced. This technique incorporates pinging to ensure the liveliness of a node once it is suspected for failure thus is aptly called affirmative. This technique performs a central sampling on the heartbeat inter-arrival time to obtain the estimation for the next heartbeat arrival. If the next heartbeat did not arrive within this timeframe, the detector raises a state of suspicion and sends a ping echo request to the monitored node. The threshold for the heartbeat to arrive reflects the current state of the node CPU load and network condition.

Subramaniyan et al. [8] introduced GEMS, a gossip enabled monitoring service. Their technique uses gossip protocols to detect failures in large, distributed systems in an asynchronous manner without the limitations of reliable multicasting in group communications. GEMS is a highly scalable technique being able to detect network partitions and dynamic insertion of new nodes. Using the very simple methodology, the technique combines reachability data from a lot of different nodes to quickly determine if a node is down. Other than for failure detection, the technique also use gossip as a form of messaging making an abstract communication from application level code instead of individual module connection.

B. Failure Recovery

In distributed environment, failures can become a normalcy. A failure recovery procedure is required to restore the system to its functional state thus ensuring High Availability. Redundancy is commonly used to eliminate Single Point of Failure (SPOF) and is a notorious technique for failure recovery in distributed environment. Basically, there are three techniques used for redundancy which are checkpointing, replication and rescheduling. However, this paper focus on replication technique.

Replication is a key technique to achieve high availability in distributed and dynamic system. In replication based recovery technique, there must be multiple copies of the same object (replicas) that are running on parallel. If one replicated object fails there will be another replica that will take over it without having to take the system offline. Replication d is the primary goals in designing a dependable distributed system [9] emphasized maintaining the data on some replicas to provide reliable services. On the other hand, keeping all of the replicas updated requires extra communication and processing. Several techniques have been proposed for replicating data management with different reliability levels. They can classified into two categories; synchronous replication and asynchronous replication. The lack of global clock makes the asynchronous replication less precise but is less costly to deploy. The terminology to also consider is for active and passive replication in systems that replicate data or services. Active replication is performed by processing the same request at every replica, while passive replication involves processing each single request on a single replica and them transferring its resultant state to the other replicas.

Many classical approaches to replication are based on a primary/backup model where one device or process has unilateral control over one or more other processes or devices [10]. Two Replica Distribution technique (TRDT) proposed by Shen et al. [11], is composed of a primary and a secondary (which can become primary) nodes and is depicted in the diagram. In this technique, the nodes have identical storage capacity and all data has two replicas on different nodes and all nodes have two data replicas as presented in Figure 1. A replicator in TRDT technique is the key component that performs the replication protocol. It builds a replication link between the primary and secondary replica with a log and storage for synchronization on individual nodes.

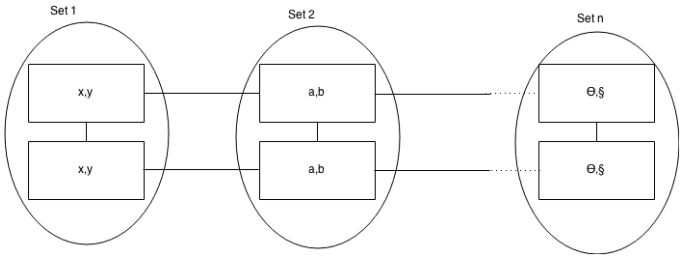

Figure 1: TRDT data replica distribution technique when *N=2n*

In Neighbor Replica Distribution Technique (NRDT), all nodes are logically designed in the form of two-dimensional *nxn* grid structure [12] as shown in Figure 2. If there are nodes in an environment where $N = n^2$, which is a set of all nodes that are logically arranged in a grid form, then it will logically be arranged in the form of nxn grid. Each node has a primary data file while its adjacent neighbor contains a replica of its primary data file.

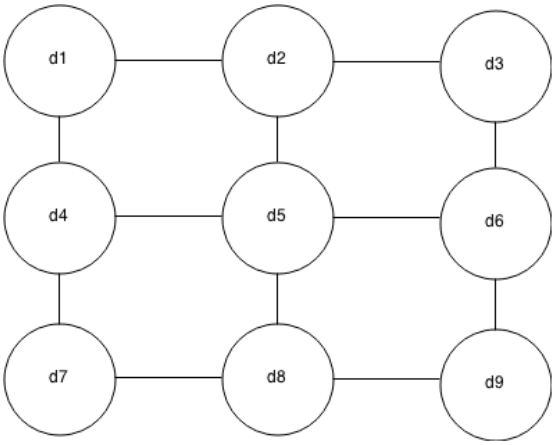

Figure 2: NRDT with 3x3 grid

C. Resource Management System for High Availability

Designing an automated high-availability resource monitoring framework with the capability to perform self-recovery has been a research interest for many years. They differentiate each other in terms of architecture, features and type of computer systems they specialize for. High Performance Computing usually requires scalability as large amount of nodes are involved such as in Grid [7]. Most monitoring system for High Performance computing uses hierarchical monitoring and replication such as with multicast and peer to peer protocols (Ganglia by UC Berkeley and Astrolabe [13]). Though redundancy is easily heightened,

these types of communications can introduce high overhead cost in both monitoring and replication which may not be worthwhile for scaled system. There are also many different solutions for resource monitoring for enterprise (Nagios (NAGIOS) Zenoss (Zennos,Inc.) OpenNMS (The OpenNMS Group, Inc.) Safekit (Evidian)). They are usually designed for more heterogeneous computing as there could be different types of resources within the enterprise. The usage can be prone to error as their design is incorporated with many functions and is usually quite complex for average users.

Cloud-based monitoring provides monitoring as a service to customers. Similar to enterprise solution, cloud-based monitoring perform centralized infrastructure/application monitoring. However the installation and maintenance are provided for which can reduce the complexity. In term of cost, it might be more expensive than hosting a monitoring service in house, especially for long term as cloud service usually charges per usage.

According to Haas [14], a high-availability stack serves one purpose: through a redundant setup of two or more nodes, ensure service availability and recover services automatically in case of a problem. Linux-HA endorses an active project and of recent Pacemaker, which is branched from Linux Heartbeat is a popular open source High Availability Stack. The stack consists of four layers: storage, cluster communications, resource management and applications. The Pacemaker's crm shell is a command interface that aims to provide simplified interface for resource management. There are many usages that involves integrating Pacemaker in Linux Heartbeat clusters.

III. FRAMEWORK

A highly available system is very much desired. There are many approaches to attain high availability that both cover hardware and software. The primary principle in these approaches is redundancy. In this research redundancy will be achieved by having additional identical components in the system namely replicas. The multiple copies are standbys for backup in case a primary application server has failed to respond. This research utilizes the distributed application architecture methodology to build a new version of NABTICS which will run on multiple servers inside a cluster. The purpose of having a distributed environment is to prepare a policy for replicated components. The replicas are placed strategically on each of the neighbors of the module's primary server.

Using the LAMP stack (Linux-Apache-MySQLPHP), a new distributed architecture of the NABTICS was designed to allow users to access information and applications through a single, consistent user environment. Rearrangement of the NABTICS involves modest restructuring of the design and software. The goal is to recreate applications that are domain specific and uses remote database as well as a local database. Applying the modularity concept in Software Engineering, the applications were restructured to be independent from each other by making separate modules.

On client-end, the application is accessed through a proxy server. This way the architecture of the distributed application delivers transparency to clients where clients only query on a single IP although the applications come from different sources. In this research distributed system, middleware is not used as on application level, the nodes are not required to communicate with each other. However, consistency is maintained through replication of database in addition to module replication in each node's neighbors.

With the administration of a resource monitor and a replication technique, server clusters are designed so that the servers in the cluster work together to protect data, keep applications and services running after failure on one of the servers, and maintain consistency of the cluster over time. The ability to handle failure allows server clusters to meet requirements for high availability. The primitive version of NABTICS is a unified system with multiple applications that run on a single site and operate a single database. In the proposed new version of NABTICS however, software modularity concept is applied where a unified system is broken down into singular independent applications. It is an important step to reduce the complexity and to also enhance scalability in this framework. Modularization is also compulsory to incorporate neighbor replication which will introduce redundancy in the system. This in return will eliminate single points of failure in the system.

Redundancy is key aspect in implementing high availability system. Each module operates on an individual server. This setting makes up a distributed architecture that comprises multiple servers running different applications for one system. All these application nodes are administered by a partnership of a monitoring and indexing services. Their function is to monitor the availability of their member nodes and enable a fail-over when a node cannot perform its duty due to a failure. This framework enables service continuity where on the user side, a minimal downtime is experienced when a site is having a problem. The novelty of this fault tolerance mechanism lies in the dynamic adaptive failure detection which will be discussed in more details.

The neighbor replication technique imposes each server to contain a copy of their immediate neighbors modules. During recovery performance, these neighbors go through a selection process to determine the best one for a fail-over based on a number of criteria. The selected node will be ordered to create a virtual IP for the failed node. As a result, the IP is kept alive and still accessible by others.

NABTICS is an application that is data centric therefore a new arrangement of database is developed to ensure data is sufficiently shared and consistent across the multiple site environment. A data replication technique is employed to manage updates and ensure retrieves of data from any site is consistent. The architectural framework of the new model includes a server cluster, a proxy server, monitoring server and an indexing server as depicted in Figure 3.

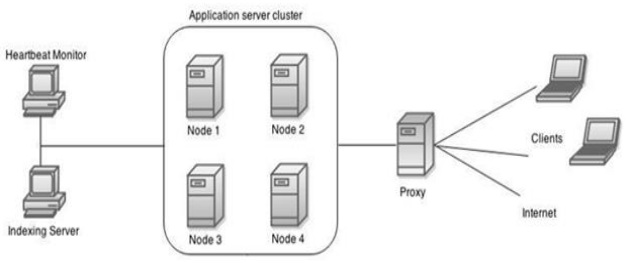

Figure 3: NABTICS in a cluster server with a resource monitor

The new model which is Neighbor Replication and Fault Recovery (NRFR) is using four instances to complement the NABTICS application. Each instance would comprise a part

of the application and support backup for other parts of the application by keeping a replica of that part. Each instance also would keep a copy of the databases. The availability of the whole system will be measured. Availability refers to the probability that a system is completely working over a period of operating time. In other words, availability is the measure of how often or how long a service or a system component is available for use [15]-[17]. The model is illustrated in Figure 2.

IV. DEPLOYMENT

Use The experiment is performed using a hypervisor called VMWare where multiple servers are run at the same time on different instances. The breakdown of the NABTICS application into distributed setting is as described in Figure 4. The NABTICS application was developed using PHP and MySQL. Apache and MySQL were installed on the Linux servers. Client can access the application cloud through host IP address and access the right files for the application in use. Figure 5 describes the event when a server fails and a recovery action has taken place where the neighbour replica is activated. A heartbeat monitor (HBM) is deployed to continuously check the aliveness of each server. An Index Service (IS) is deployed to keep records and status of all servers in this environment. In an event where the HBM detects a down server, it will reconfirm by pinging one more time. A node is said to fail if it does not respond to this pinging. HBM then will notify IS and update the status of the server. IS then performs a neighbor selection process to determine which replica is best to take to serve the client of the down server. This is done by invoking the selected neighbor to activate virtual IP. Consequently the NABTICS application process is resumed and client does only experience minimum downtime while using the application.

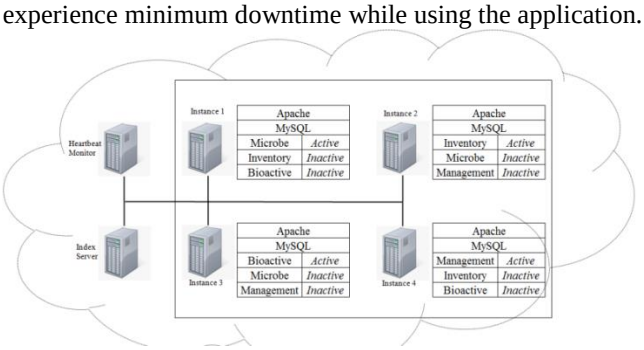

Figure 4: NABTICS model for applying Neighbor Replication and Fault Recovery NRFR

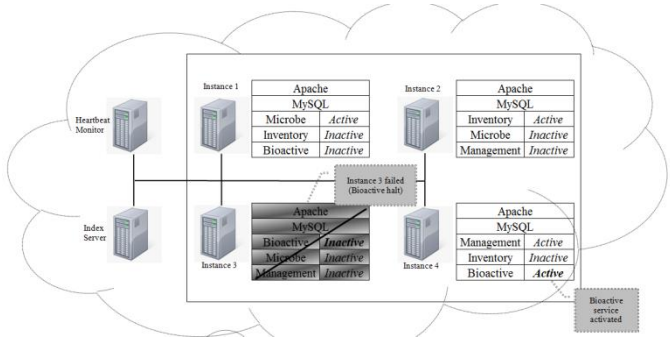

Figure 5: Instance3 is failed, its application (Bioactive) is activated in its neighbour Instance 4 within cloud using NRFR

V. PERFORMANCE EVALUATION

In this section, an analysis of the availability of the replication techniques will be presented. Availability refers to the probability a system is completely working over a period of operating time. In other words, availability is the measure of how often or how long a service or a system component is available for use as shown in Equation (1) [16].

$$\text{Availability} = \frac{\text{Operational}}{\text{Operational} + \text{Non} - \text{Operational}} \quad (1)$$

Let be the number of nodes that are operating correctly at time t, be the number of nodes that have failed at time t, and N be the number of nodes that are in operation at time t as in Equation (2) [17].

$$A(t) = \frac{N_0(t)}{N} = \frac{N_0(t)}{N_0(t) + N_j(t)} \quad (2)$$

The availability in series can be expressed as in Equation (3) [7].

$$A = A_V \times A_Z \quad (3)$$

And the availability in parallel can be expressed as equation (4).

$$A = 1 - (1 - A_V)(1 - A_Z) \quad (4)$$

If however there is a mixed environment between parallel and serial the availability A can be defined as equation (5).

$$A = (1 - (1 - A_W)(1 - A_X)) \times (1 - (1 - A_V)(1 - A_Z)) \quad (5)$$

For the performance evaluation purposes let consider a simple model of a distributed system with nine (9) nodes. Since we want to shows the significant of technique, this evaluation used nine nodes instead of using 4 nodes. Each component/site has availability as in Table 1.

Without any replication technique or single point of failure (SPOF) such system components are in series, therefore the system availability is the product of all the components' availability, as given in (4). System availability = 0.95 ×0.955 ×0.95 ×0.97 ×0.96 ×0.97 ×0.95 ×0.95 ×0.99 =0.6956.

Table 1
The nine components of interdependent servers and its availabilities

Component	Availability
Web	0.95
Application	0.955
Database	0.95
DNS	0.97
Firewall	0.96
Switch	0.97
Data Center	0.95
Applications2	0.95
Manager	0.99
Total Availability	0.6956

The TRDT availability prediction model adds a second replica to each of the servers. When a system is comprised of two redundant components, then the availability of the system

can be calculated by using parallel formula as expressed (5).

The system availability using three replication models of the same nine components have been evaluated based on Equation (3) for SFOP, Equation (4) for TRDT and Equation (5) for NRFR. Summary of the result is shown the following Table 2.

Table 2
Comparison of improvements using different replication model

Replication Model	System Availability	Improvement
SPOF	0.6956	0.000%
TRDT	0.98458	41.544%
NRFR	0.99978	43.729%

In terms of system availability score, NRFR is the most excellent followed by TRDT. However this is the the first year only. The availability and unavailability prediction over an extended period of 10 year for TRDT and NRFR. The availability prediction for second year ($A_{y2)}$ can be calculated as $A_{y2} = 1 - 2Qs$, for the third year $A_{y3} = 1 - 3Q$ and so forth.

From the Figure 6 observation, it demonstrates that, as the years goes by the availability gap is apparently larger and larger. This is especially for TRDT, the TRDT availability reduce about 4% per year or 40% for ten years. However the NRFR availability reduces about 0.12% per year or 1.2% for period of ten years. The graph plotted in figure 6 demonstrated the availability gap between TRDT and NRFR for 10 years.

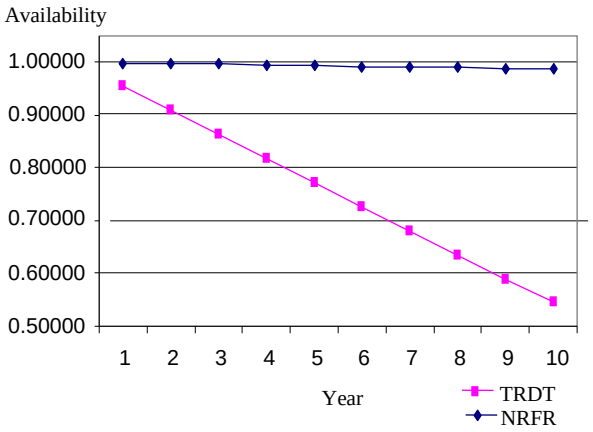

Figure 6: The availability gap between TRDT and NRFR for 10 years

VI. Conclusion and Future Works

A high availability framework often combines a resource monitoring to supervise sites, failure detection technique and recovery to gives fault tolerance to a system. A system with fault tolerance can withstand a number of failures and become more available therefore more reliable.

Therefore, to provide the repository for large number of users and data as well as to provide robust and reliable service, the NRFR Replication technique for NABTICS system need to be deployed within cloud environment. As an active community of researchers in this field, the scope of this system is dynamic and wide covering. This application provides tools for research and data management. Heavy usage of this application would also require reliability and availability to ensure continuity during task. Cloud based neighbour replica failure recovery is a dynamic technique that can ensure fault detection and recovery by keeping replicas on multiple servers. Having multiple replicas databases requires a mechanism to ensure these databases are synchronous during updates and retrieves by users.

Acknowledgment

The research was supported by Ministry of Higher Education of Malaysia (MOHE) for the grant of Fundamental Research Grant Scheme (FRGS). (Ref: FRGS/2/2014/ICT07/UMT/02/1) A New and Efficient Technique for High Dependable Marine Knowledge Repository in Cloud Environment.

References

[1] K. An, S. Shekhar, F. Caglar, A. Gokhale, and S. Sastry, "A cloud middleware for assuring performance and high availability of soft real-time applications," *Journal of Systems Architecture*, vol. 60, no. 9, pp. 757-769, 2014.

[2] S. Gokhale, J. Crigler, W. Farr, and D. Wallace, "System availability analysis considering hardware/software failure severities," in *Proc. 29th Annual IEEE/NASA Software Engineering Workshop 2005*, Greenbelt, USA, 2005, pp. 47–56.

[3] F. G. Khan, K. Qureshi, and B. Nazir, "Performance evaluation of fault tolerance techniques in grid computing system," *Computers & Electrical Engineering*, vol. 36, pp. 1110–1122, Nov 2010.

[4] T.L. Broto, and D. Hagimont, "Approaches to cloud computing fault tolerance," in *Proc. International Conference on Computer, Information and Telecommunication Systems (CITS)*, Amman, Jordan, 2012, pp. 1–6.

[5] S. S. Sathya and K. S. Babu, "Survey of fault tolerant techniques for grid," *Computer Science Review*, vol. 4, no. 2, pp. 101-120, 2010.

[6] T. Ma, J. Hillston and S. Anderson, "Evaluation of the QoS of crash-recovery failure detection categories and subject descriptors," in *Proc. of the 2007 ACM symposium on Applied Computing*, Seoul, Korea, 2007, pp. 538–542.

[7] A. S. M. Noor, *Data Neighbor Replica Affirmative Adaptive Failure Detection and Autonomous Recovery*. Dissertation for Doctor of Philosophy in Computer Science, Universiti Tun Hussein Onn Malaysia, 2012.

[8] R. Subramaniyan, P. Raman, A. D. George, and M. Radlinski, "GEMS: Gossip-enabled monitoring Service for scalable heterogeneous distributed systems," *Cluster Computing*, vol. 9, pp. 101–120, Jan. 2006.

[9] T. Amjad, M. Sher, and A. Daud, "A survey of dynamic replication strategies for improving data availability in data grids," *Future Generation Computer Systems*, vol. 28, pp. 337–349, Feb. 2012.

[10] H. H. Shen, S. M. Chen, W. M. Zheng, and S. M. Shi, "A communication model for data availability on server clusters," in *Proc. Int'l. Symposium on Distributed Computing and Application*, Wuhan, 2001, pp. 169-171.

[11] R. Mamat, M. M. Deris, and M. Jalil, "Neighbor replica distribution technique for cluster server systems," *Malaysian Journal of Computer Science*, vol. 17, pp. 11–20, 2004.

[12] D. Ford, F. I. Popovici, M. Stokely, V.-a. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation*, USENIX, 2010.

[13] H. Lin, K. Chen, and X. Yan, "Astrolabe: a Grid Operating Environment with Full-fledged Usability," in *Proc. 6th International Conference on Grid and Cooperative Computing*, Los Alamitos, USA, 2007.

[14] F. Haas, "Ahead of the pack: the pacemaker high-availability stack," *Linux Digital Journal Magazine*, pp. 98–100, April 2012.

[15] J. Gray and D. P. Siewiorek, "High availability computer systems," *IEEE Computer*, vol. 24, no. 9, pp. 39-48, 1991.

[16] A. S. M. Noor and M. M. Deris, "Fail-stop failure recovery in neighbor replica environment," *Procedia Computer Science*, vol. 19, pp. 1040, 2013.

[17] A. S. M. Noor and M. M. Deris, "Failure recovery mechanism in neighbor replica distribution architecture," in *Lecture Notes in Computer Science (LNCS)*, vol. 6377, 2010, pp. 41-48.