A New High Availability Framework for NABTICS

EMMA AHMAD SIRAJUDIN
Universiti Malaysia Terengganu
School of Informatics and Applied Mathematics
21030 Gong Badak, Terengganu
MALAYSIA
gsk1713@pps.umt.edu.my

AHMAD SHUKRI MOHD NOOR
Universiti Malaysia Terengganu
School of Informatics and Applied Mathematics
21030 Gong Badak, Terengganu
MALAYSIA
ashukri@umt.edu.my

Abstract: In order to promise service continuity and reliability of data, a new environment was set up for a Marine Knowledge Repository application namely NABTICS. This new environment improves availability of the application as well as provides a stronger back up system for the data. The new implementation requires new arrangements for the servers, a new resouce management framework that employs fault tolerance as well as a data replication scheme that can deliver data consistency.

Key–Words: High Availability, Replication, Fault Tolerance

1 Introduction

As researchers experience progressive findings and new discovery in the Microbial fields, a need for a highly reliable and available software arises. The new vision not only eases the task of the researchers but also contributes to the research advancement of Microbes study and Marine Ecology. National Bioinformatics System (NABTICS) is a knowledge repository software that is aimed to help researchers to manage their research database as well as conveniently share with other researchers.

The pursuit for High Availability has been active for many decades. A plethora of techniques and methodologies have emerged with their own strength and weaknesses. Resource Monitoring System (RMS) is one technique to achieve High Availability that is most commonly used in IT landscape. As such it is often regarded as the most critical component as it oversees all the sub-systems and services performance to maintain their availability and is responsible to raise alert when a failure has occurred.

This research revolves around the development of a resource management capability that can effectively detect a failure and invoke a recovery to be performed from using a neighbor replication technique. This proposed development uses a number of system resources as well as an elegant fault detection algorithm [1]. Compared to a number of techniques, the adaptive detection technique improves the MTTF and reciprocally the MTBR where the recovery can be invoked earlier as fault detected sooner.

It is a prerequisite for the RMS itself to be highly available [**?**]. In order to provide an effective RMS, most of the solution providers have continuously improved their RMS in terms of functionality and reliability. Unfortunately, as the RMS advances, its own complexity increases proportionally. Consequently, the task of managing RMS is getting more challenging. Using the proposed framework therefore can be more apt for the specific purpose to improve NABTICS performance because this framework uses a more straight forward flow and can be easily scaled as well. Besides that having two separate services for resource monitoring can give reliability in the RMS in of itself.

Fail-over clustering is specifically aimed to increase availability as opposed to load-balancing that is more motivated to improve performance. The proposed framework therefore is a fail-over cluster where it incorporates heartbeat monitoring in a node cluster with high redundancy. As such it can also improve system performance as different modules are separated on multiple sites. It is empirically proven improve the availability for each module as their replicas are maintained on strategic places and will be activated when the primary module has failed.

2 Related Works

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 [2]. Most monitoring system

ISBN: 978-1-61804-297-2

for High Performance computing uses hierarchical monitoring and replication such as with multicast and peer to peer protocols (*Ganglia* by UC Berkeley and *Astrolabe* [3]). 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* Zenoss, Inc. *OpenNMS* (The Open-NMS 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 inhouse especially for long term as cloud service usually charges per usage.

According to Haas [4], 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.

3 Framework

3.1 Server CLuster

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-MySQL-PHP), 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.

3.2 High Availability Framework

Redundancy is vital but it is not sufficient to increase availability of a system [5]. Multiple instances need to be entangled by having a proper monitoring and indexing system so that changes can be recorded and acted upon. the high availability framework involves a heartbeat monitoring program and an indexing service to manage the resource cluster.

Firstly, the monitored application nodes have to be registered at the Index Server (IS). The nodes send periodic heart beat message to the Heartbeat Monitor (HBM) to indicate its alive status. Data receiver in HBM accepts heartbeat messages. Each monitored node has a separate data receiver assigned to it. The data receiver maintains a threshold for the next Estimated Time of Arrival (ETA) of the heart beat based on the sampling of heart beat arrival time it receives. Based on the ETA, the HBM can detect a faulty node and investigate the node by sending a ping request to it. If the node replies, HBM clears its suspicion and resumes monitoring with reset threshold. Otherwise if

ISBN: 978-1-61804-297-2

node is unreachable, HBM declares that it has failed and updates this information to IS and commands it to perform recovery.

In [1] the affirmative adaptive fault detection component gives timely detection of node failure with completeness and high accuracy. The fault detection is designed to be dynamic by deploying the ETA based on most current and older samples of the receiving time. Consequently, the detection program can adapt quickly to the changing behavior of the monitored server and network condition. Such intuitive fault measure can effectively avoid false detection and gives a fast recovery. A false detection can trigger unnecessary recovery and put dispensable load on network and server which can cause waste of resources.

A service can be lost during operation due to site crash or network failure. While this loss can stop service, it can also be due to temporary glitch. Therefore sometimes recovery is not necessary. With affirmative measure unnecessary invocation can be avoided by only invoking the recovery once the faulty status of the server or network is confirmed. The affirmative step is delivered by employing ping inside the fault detection framework. Upon detecting a failure, HBM sends a ping request from monitored node to confirm its status.

After HBM has confirmed a failure, it invokes the Index Server to start recovery process. The program will look into the index to find the registered node's information and who are its neighbors. After weighting the neighbors for its server condition and replica integrity, the program will select a fail-over neighbor. It then sends message to the selected server to create a virtual IP to transfer the service to itself.

3.3 Fault Detection and Recovery

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 final architecture framework of the new model includes a server cluster, a proxy server, monitoring server and an indexing server as depicted in the following illustration.

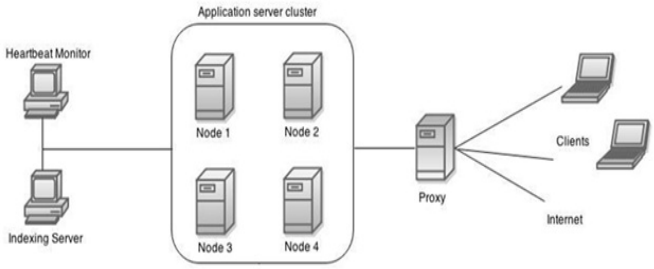

Figure 1: The framework manages and controls the member nodes of the distributed NABTICS in a cluster server.

3.4 Database Replication

After nabtics has been distributed into separate modules, singular local database is no longer relevant. Considering that there will be many issues with multiple databases and there will be high latency with single database in distributed environment, in this research the direction is to set a local database for reduced cost and faster read and a remote master database to keep consistent data.

MySQL database replication provides a way to have master-slave control on replicated databases. In

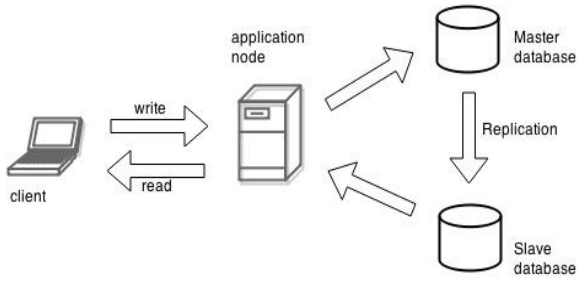

Figure 2: MySQL Master-Slave Replication

a master-slave relationship, master database will propagate all changes to its slave databases. This will ensure consistency in data as implied that only the master will hold the most correct version of the database. this will avoid many conflicts in the distributed application system.

The heartbeat monitor (HBM) is predisposed to be the most reliable server and also the most secure as the HBM is behind any communication with the outside world. Having this characteristic, HBM is chosen to host the master database. Database writes from the application node (as received by clients) is forwarded to HBM while the reads are done locally on application node. All changes on master will be propagated to the slave databases on asynchronous basis. If a slave server was behind in replication, it can update itself by checking for any updates or changes in the masters log. This arrangement ensures higher consistency since:

1. Can avoid overwritten as only one copy gets all the updates
2. All database instances are updated simultaneously regardless of its version
3. Faster reads since it is done locally

4 Conclusion

The overall view of the original system has improved slightly in availability readily as the modularity concept was applied where the number of single point of failure is reduced. Further adding the resource monitor adds robustness in high availability delivery as well as ensuring service continuity with fault tolerance. The proposed framework uses an adaptive fault tolerance technique to achieve high availability with neighbor replication and master-slave database replication. This technique is very beneficial in monitoring resource inside a dynamic distributed environment with a moderate number of nodes to increase the overall availability and performance of the system.

Acknowledgements: The research was supported by the Ministry of Science, Technology and Innovation (MOSTI) of Malaysia. (Grant No. 52074)

References:

[1] M. M. D. Ahmad Shukri Mohd Noor, "Fail-stop failure recovery in neighbor replica environment," 2013.

[2] F. P. D Ford, F Labelle, "Availability in globally distributed storage system," *Google Inc.*, 2014.

[3] H. Lin, K. Chen, and X. Yan, "Astrolabe: a grid operating environment with full-fledged usability," pp. 366–371, Aug 2007.

[4] F. Haas, "Ahead of the pack: the pacemaker high-availability stack," *Linux Digital Journal*, 2012.

[5] D. P. S. Jim Gray, "High availability computer systems," 1991.