IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

An Efficient Approach to Design a Low Cost and High Performance Active-Active Clustering for Applications along with Database

D. Dashora[1]

[1]Department of Computer Engineering, Faculty of Engineering, Pacific University, Udaipur, Rajasthan, India

Corresponding Author: devesh.dashora@gmail.com, Tel.: +91-98290 57506

Available online at: www.ijcseonline.org

Accepted: 17/Jul/2018, Published: 31/July/2018

***Abstract*—** Online applications like Facebook, Google Apps, WhatsApp, Amazon, Flipkart etc. are huge companies that have a huge customer base and many of customers use their apps concurrently. It put much load over their servers and relatively more load on the application. This overload degrades the performance of the application. The more request-response to an application, the more it applies load on the server. However, a server infrastructure consists various limits to serve a total number of requests per second. Therefore, the server infrastructure and architecture of the application must be developed in such a manner that can be deployed on multiple servers. When the number of users increased, the application can serve to all users by just deploying the same application to more servers. Such kind to development architecture requires more knowledge and more experienced developers. Also, the cost of such deployment need lots of money to purchase/subscribe various third-party packages. In this paper, I am presenting a design architecture and deployment method for Active-Active application clustering that will help to develop applications, which can scale-up at any time without making any changes to application code. The application can handle any number of requests and can serve more users than expected. This architecture uses open source tools and technologies so that it is a low cost solution and provides high performance.

***Keywords*—** Cloud computing, virtualization, application clustering, distributed application, database clustering, load balancer, HAProxy, Keepalived, Memcache, Postgres database, hypervisor.

I. INTRODUCTION

Application clustering denotes that a single application can be deployed on multiple server machines and share requests of users. Especially when the application is serving requests online from the remote location. There are so many companies like Google, Facebook, WhatsApp, etc. which are using clustering in their applications. ERP applications are generally built upon some kind of clustering. Although application clustering is a very expensive and require a lot of knowledge to implement. But now there are various open source tools and technologies are available which allows creating our own application and database cluster without any cost or very low maintenance cost. Such tools are very popular and being used by lots of giant IT companies. Such tools are being described in this paper and also explain to create our own application cluster with the database.

The following consideration adopted for the architecture:

- Design a low cost and highly efficient solution for application clustering.
- Active-Active application clustering, which can serve millions of customer requests.
- Use of open source tools and technologies.
- 100% Up-time for applications to customers.

In the Active-Active clustering, multiple instances of applications and database will be distributed on multiple server machines and work in a group to serve millions of requests. It is most important that all components of the Active-Active clustered application must support clustering like a database, shared memory, message queue etc. So it needs to choose a right component to design and implement Active-Active application clustering.

The objective of this research to develop an efficient, high performance, low cost, and 100% up-time service applications which can deliver 24x7 services to users even it faces any kind of disaster. Active-Active means that the application and all of its component will have multiple numbers of instances and can deliver always running services to users without feeling them that there is any failure in the application system.

The rest of the paper is organized as follows. Section II is an illustration of related works about the proposed topic. Section III discusses the terminology and concept used in the research. Section IV describes the proposed architecture. Section V provides a description of various open source tools and technology used in the research. Section VI presents comprehensive test results. Section VI concludes the paper.

© 2018, IJCSE All Rights Reserved

II. RELATED WORK

The work done previously was only limited to clustering of database component of service applications. Most of the service applications are deployed on the single server system. And a very few applications are deployed on multiple servers but again only one server work as a primary application server and other remaining nodes works only when some disaster occurs. The shifting of services to secondary nodes are still a manual process. It brings downtime and impacts the users of that application. And when the primary server recovers, they switch the services to the primary server.
This research includes Active-Active clustering of applications along with database and other supporting components. So it reduces the downtime to zero and decreases the manual intervention as well.

III. THE TERMINOLOGY

Active-Active clustering requires multiple hosts to create a cluster (group) for all the components of the application. For small applications or start-up business applications can use at-least two nodes or hosts for each component of the applications. For example two nodes for the application cluster, two nodes for database cluster etc. So it needs to purchase many server machines to deploy such an application. Rather, purchasing many servers, we can use a hypervisor through which we can create multiple virtual machines in a single physical host. It is also known as Virtualization technique. In the Virtualization technique, all virtual machines share the resources of host server like CPU, RAM, network etc. and also enhances the performance of server applications. Other benefits of virtualization are that we can upgrade or downgrade the resource utilized by a virtual machine at any time, even when the virtual machine is up (running) and serving request to users.
So virtualization plays a vital role to design and develop a low-cost Active-Active application cluster. There are many options available as a hypervisor for virtualization. VMware ESXi is one of the popular hypervisors which provides both free and subscription-based services. KVM, XenServer or ProxMox are the fully open source and reliable hypervisors. We can also clone a virtual machine to create an exact copy of any component of our application. VMware provides this feature as vMotion in its product named VMware vCenter. KVM enables this feature by using QEMU.

Using virtualization, we can manage a single host. It provides high-performance computing. But generally, we may have multiple servers or hosts, which may have the same or different kind of hypervisors. The servers may also be located on different physical location. And also all servers must be connected to each other in a high bandwidth network to create a cluster. So managing multiple servers with different hardware and operating system or hypervisor in a network is also a critical task. Adding new server and removing a faulty server from the cluster also requires high attention and manual work. To overcome this problem, we can create a cloud environment for the servers and network.

Cloud computing is not a new term, it is widely used and many cloud service providers are delivering ready-made high performing cloud cluster. We can hire cloud infrastructure servers, public cloud or individual virtual machines for all application components. Hiring cloud-based virtual machines from vendors are cheap when our uses of the servers are very low. But when the uses of servers get increased, the cost also get increased because the public cloud is based on pay-per-use. If we have enough budget to purchase our own servers, then we can go with a private cloud to group and manage our servers. A private cloud allows us to create our own cluster of servers or data-center where all of our physical servers can be connected to the virtual network. A virtual network is again sub-instance of our physical network. OpenStack is the most popular and open source framework available, which provides all the features and options, which are required to create our own Private cloud and data-center.

In Active-Active clustering, all request of users can be served by using a single IP address which is internally connected to the private network with multiple nodes. This enhances the security much because there is only one public IP available to users and if someone wants to attack our application network, we can block from this single IP address [14]. Generally, a cluster contains a load balancer which holds a single static IP address and can scale the cluster horizontally or vertically. This load balancer work as an entry point for our apps. All other nodes are connected to the interconnected network and work as a single computer. A cluster can be a group of ten thousand or more processors [17].

The HAProxy load balancer is an open source tool which is widely used to provide single IP to outside world. The DDoS protection is highly efficient to block hacker's attack. HAProxy load balancer provides various algorithms to redirects incoming requests to an internal range of IP addresses or domain name. It can also depend upon geo-location based or type of service request.

An application cluster can have 2 or more application nodes and all node may require a connection or connection pool to a database. So it is also required to configure the database in the cluster. There are many databases which are cluster-enabled. Database cluster can be either master-slave type or master-master type. A master-slave database cluster uses Eager replication where master-master uses Lazy replication technique [20]. A master-slave database cluster contains a master database node which can both read/write the database and one or more slave nodes which only allow reading data from the database. In a master-master or multi-master database cluster, all nodes work like master node i.e. read/write operation can be done on all nodes in the cluster.

© 2018, IJCSE All Rights Reserved

MariaDB Galera cluster is used to create multi-master cluster for MariaDB database (earlier was MySQL database). For Postgres database, pgPool-II and Bi-Directional Replication (BDR) open source tools are very popular and also being used in production. In multi-master replication concurrent queries and the transaction can be committed on all nodes asynchronously i.e. it improves the performance and can distribute the load over multiple nodes of the database [21].

For sharing common and temporary data among the cluster nodes requires an additional in-memory database. The in-memory database can store data in RAM, which can be accessed in the cluster for fast data sharing. For example, to store the session data for a user, we can use this kind of database. Memcache and Redis Cache are the top brands in open source in-memory shared databases. Both are fast and easy-to-use. And also both have some pros and cons. So it requires analysis to conclude and choose which is better for our Active-Active application. An in-memory database is used to store temporary data only.

In Active-Active application clustering, there is more requirement which is inter-process communication system. It is required when two or more applications want to send/receive messages to perform the further task. It is not necessary for all applications. Mostly it is used when a single application uses multiple other small applications also known as threads. Such threads work independently and can produce output itself. But the output generated from one thread may be used as input for other threads. So such output message of a thread can be delivered to other thread using a messaging or inter-process messaging system. For this, we can go with Kafka which most popular nowadays and also it is open source. We can create a cluster of Kafka as well. We can also use ActiveMQ, RabbitMQ, or ZeroMQ instead of Kafka.

IV. THE ARCHITECTURE

The Active-Active architecture contains various components which must be arranged and deployed in proper manner. All components have their own role, task, characteristics and deployment procedure. Some components are required to design Active-Active Clustering and some components are optional or dependent upon the application needs. A general architecture for Active-Active clustering described as follows:

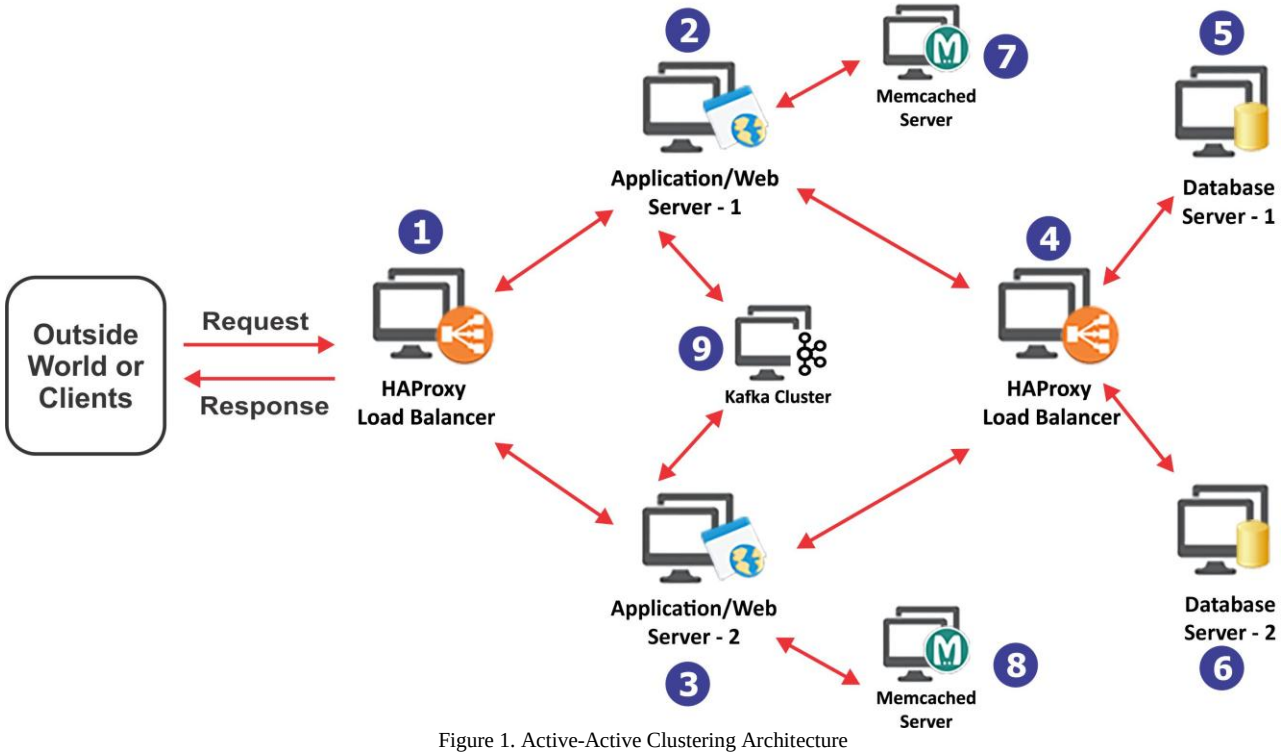

Figure 1. Active-Active Clustering Architecture

A. *Major Components*

The Active-Active application clustering architecture is shown in the Fig. 1. It has various components to fulfill the purpose of this architecture. Each component is indicated by a sequential number and described as below:

1) *Load balancer for outside world:*

All the request from users/clients will come into the system will pass through this component. It is the single entry point from where user requests can go into the application servers. This is the only single point of failure when this become faulty. To prevent the service break down, we need to deploy

© 2018, IJCSE All Rights Reserved

another secondary failover load-balancer, which will work only when this load balancer will stop working.

2) Application node1

The node contains the application instance which will actually process all the request. All remaining component will be used by application reside in this node. So it must have access to all the other components.

3) Application node2

This is the second instance of the application node1 because the Active-Active application architecture requires a minimum of two nodes of applications. We can add more nodes for the application at any piece of time or whenever the requirement get increased.

4) Load Balancer for Database:

This load balancer works as a bridge between the application and database cluster. As already described that in an Active-Active cluster environment, the access to the database must also be in a cluster. So all nodes of the database cluster will be handled by this load balancer.

5) Database server1

It is one of the nodes from the multi-master replication cluster of the database. It can read/store from/to the database. This database node can accept concurrent queries from the application server via the load balancer and execute them as a transaction. After each successful transaction, the node sends the same transaction to all other remaining nodes of the database cluster.

6) Database server2

This is the second instance of the multi-master database cluster. As we know, a cluster requires at-least two nodes to form a cluster. This node can also accept concurrent queries from front-end applications and execute them as a transaction. After successful execution of a transaction, this node will also send the same transaction on other remaining nodes of the database cluster. This will be done by all nodes very quickly.

7) Memcache server1

This node is again will be used by the application. The application can use this node to store in-memory data or cache for fast retrieval. This is used to store intermediate temporary data in the cache and can be accessed very fast.

8) Memcache server2

This is again secondary node for in-memory caching. But it is developer choice to have replication or sync with other Memcache node because some application may need the cached temporary available all the time even on failure of one node.

9) Kafka Cluster

It is for exchange information among application servers. It works as a message queue for inter-process communication. Every application server can send a piece of information to the Kafka server and afterward, all other nodes can pick that information for their use. Kafka also needs to be deployed in form of the cluster because if one node gets failed, then the messages sent by an application server can be delivered to another node even any of Kafka node gets failed.

V. TOOLS AND TECHNOLOGY

To implement this architecture, the following tools and technologies can be used.

A. Software-based load balancer

There are many software-based load balancers are available. Some open source load balancers are followed:

Table 1. Some open source and reliable software-based load balancers

S. No.	Name	Description
1	HAProxy load balancer	HAProxy is a free, reliable and widely used open source software based load balancer which manages load among two or more back-end servers. It can distribute request on round-robin, geo-location based or using various other algorithms. It makes services available all the time to users (No downtime). HAProxy regularly checks the health of all of its backend. If one of the backend servers gets down, it redirects all the requests to the remaining backend servers.
2	NGINX Proxy server and load balancer	NGINX is another choice for developers and system administrators. It also works like a HAPorxy load balancer. It can serve for HTTP, HTTPS, TCP and UDP protocols. It can also work for mail server protocols like IMAP, POP, and SMTP. It is a combination of the load balancer, reverse proxy, content cache, and web server.

© 2018, IJCSE All Rights Reserved

B. Database cluster with multi-master replication

For the Active-Active cluster architecture, we have chosen the multi-master asynchronous replication for the database. It allows us to divide the load of database access among multi-nodes in the cluster. Some open source database clustering tools and technologies are as follows:

Table 2. Some open source multi-master replication based database tools

S. No.	Name	Description
1	PostgreSQL BDR	PostgreSQL BDR (Bi-directional Replication) is a product of 2Quadrent which is open source and work upon the core PostgreSQL. It provides asynchronous multi-master logical replication for PostgreSQL database. Multi-master means it can execute a query on any node of the cluster and then commit it onto the all other nodes of cluster row-by-row and asynchronously.
2	pgPool-II	The pgPool-II is mainly used to achieve connection pooling. It also has the feature of replication and load balancing. Any node in a pgPool cluster can execute queries and then commit it to other nodes.
3	MariaDB Galera cluster	If you want to use the MariaDB database, then it needs to create a cluster of MariaDB Galera cluster. It is a separate package and can be installed with existing MariaDB server. It provides synchronous replication but supports Active-Active multi-master replication. This includes a package named wsrep provider which handles the whole replication process.

C. In memory cache database

To store temporary data during for an application, we can use any in-memory caching system. It increases the performance of the application. There are many open source tools available for it. Some open source in-memory cache tools are as follows:

Table 3. Some open source in-memory caching

S. No.	Name	Description
1	Memcache	It is a key-value pair based in-memory database and belongs to the NoSQL family. It keeps data in RAM memory mapped with a unique key. It is mainly used when data is small and static. This is multi-threaded in nature so that it can be easily scaled up by occupying more thread in memory. If the server gets fails, then all data will be lost because it is volatile and doesn't support replication.
2	Redis Cache	Redis cache is also an in-memory cache database and also store each data using a unique key. It uses data-structures like a linked list, arrays, and sets. So it can be used as a shared queue (linked list), messaging queue (publish/subscribe), storing session data in a hashmap and also can also be used to store sorted data by using sorted sets.

© 2018, IJCSE All Rights Reserved

VI. RESULTS AND DISCUSSION

A. Test for BDR cluster with HAProxy

This test includes a client application which is developed using C# language. One node for HAProxy load balancer and 2 database nodes for BDR cluster. It is shown in the Fig. 2

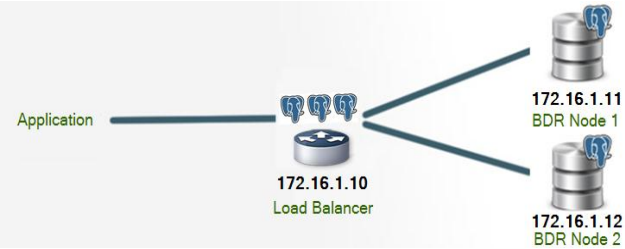

Figure 2. PostgreSQL BDR cluster of 2 nodes with a HAProxy load balancer

The client application sends high volume data to HAProxy load balancer continuously. And the load balancer sends that data queries using a round robin algorithm to two back-end database nodes of BDR cluster. Both nodes are accepting queries and execute them. After each successful execution of the query, each node sends the same query to each other and executed by them. In case of conflicts, only the last updated or inserted data will be written to the database.

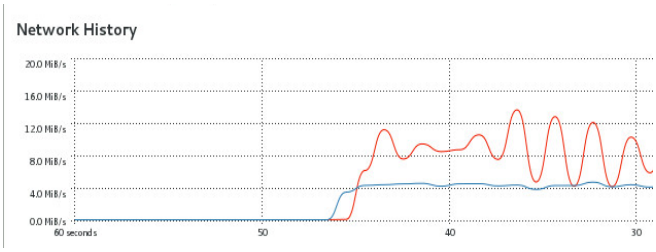

Figure 3. Network History Graph

The above Fig. 3 shows the overall processing of a BDR node. It contains two graph lines. The Red graph line shows all the incoming data queries to execute and the Blue graph line shows the outgoing queries which are being sent to the other node of the BDR cluster for replication.

B. Test for store data into different database cluster

In this test, I am going to calculate the time duration to store the same amount data to the different type of database clusters. This test includes a client application which is developed using C# language. There is one node for the HAProxy load balancer and 2 database nodes for BDR/ pgPool-II/ MariaDB Galera cluster.

The client application sends high volume data to HAProxy load balancer continuously. And the load balancer diverts all the data queries using a round robin algorithm to two back-end nodes of the database cluster. Both nodes accept queries and execute them. This process was repeated for the BDR/ pgPool-II/ MariaDB Galera cluster and total time duration for this are declared in the following table:

Table 4. Total time is taken to store data

Type of Cluster	Time duration
BDR cluster	52 mins
PgPool cluster	80 mins
MariaDB Galera cluster	105 mins

So the above test result in Table 4 shows that BDR took less time in comparison to other 2 database clusters.

C. Test for fetch data from different database cluster

In this test, I am going to calculate the time duration to fetch the same amount data to the different type of database clusters. This test also includes a client application which will make a connection to different types of database cluster and get data from them. This test includes one node for the HAProxy load balancer and 2 database nodes for BDR/ pgPool-II/ MariaDB Galera cluster.

The client application sends a single query to fetch data from the database clusters, continuously. Although it is sending only one query to the database, then the load balancer transfer it to anyone node of the database cluster. For more precise test results, I send multiple queries on at a time for a different number of data records. This process was repeated for the BDR/ pgPool-II/ MariaDB Galera cluster and total time taken for this are shown in the following table:

Table 5. Total time is taken to fetch data

Number of records	BDR/ pgPool	MariaDB
10,000 records	0.3 seconds	0.032 seconds
100,000 records	3.6 seconds	0.343 seconds
1,000,000 records	37 seconds	3.453 seconds
10,000,000 records	376 seconds	35.25 seconds

So the above test results in Table 5 shows that the BDR and pgPool both always took higher time to fetch data from the database in comparison to MariaDB Galera cluster. So choosing right database cluster is dependent upon the need and also it will depend upon the question that which type of operation will be performed mostly and how many times?

© 2018, IJCSE All Rights Reserved

According to the test results, BDR is faster to store data into the database cluster where MariaDB Galera cluster is faster while accessing data from the database. If our application store data frequently like customer data, transaction data then we should go with BDR and if our application store few amounts of data but performs various operations by fetching data from the database like data analysis, report generation etc. then we should go with the MariaDB database.

VII. Conclusion and Future Scope

A. Conclusion

The Active-Active application clustering provides a robust and reliable architecture. It enables us to create 100% up-time applications with zero downtime. It uses open source tools and technologies, so its low cost and even it fits into the budget for small applications.

In Active-Active clustering, all components are built upon the cluster topology, so it has very fewer possibilities for service downtime even in case of any sort of disaster with hardware or network malfunctioning.

HAProxy, BDR, Memcache, etc. are open source tools are well tested and widely used in production from many years. And also provides regular updates from time to time. So this fulfills the needs of an application system for always up and running. OpenStack provides cloud infrastructure. VMware and KVM both can create a virtualization environment. Memcache or Redis Cache is used to access shared data. PostgreSQL BDR cluster providing multi-master database replication with concurrent read/write to the database. Kafka is used as a Message queue for inter-process communication. And the HAProxy load balancer is doing all the neat things which are required always up and running services to the customer.

B. Future Scope

The deployment of all described open source tools involve so many manual interventions and also require skilled persons who have knowledge of Linux and clustering. So there is a scope of automatic deployment with very less manual intervention.

Deployment of a private cloud in an organization is very crucial and also requires expert and skilled persons. So private cloud deployment is another challenging job. Although we can hire or purchase from cloud service providers if we don't have enough budget and skills.

Acknowledgment

I would like to thank all university staff and faculty members of the college dept. for their guidance and kind support, which has helped me a lot to complete this research project successfully.

References

[1] R. Aluvalu, M. A. Vardhaman and J. Kantaria, *"Performance evaluation of clustering algorithms for dynamic VM allocation in cloud computing"*, In the Proceedings of the International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, India, pp. 1560, 2017.

[2] A. Awasthi and R. Gupta, *"Multiple hypervisor based Open Stack cloud and VM migration"*, In the Proceedings of the Cloud System and Big Data Engineering (Confluence), Noida, India, pp. 130, 2016.

[3] A. Babar and B. Ramsey, *"Building Secure and Scalable Private Cloud Infrastructure with Open Stack"*, In the Proceedings of the Enterprise Distributed Object Computing Workshop (EDOCW), Adelaide, SA, Australia, pp. 166, 2015.

[4] K. A. Bakar, M. H. Shaharill and M. Ahmed, *"Performance evaluation of a clustered memcache"*, In the Proceedings of the International Conference on Information and Communication Technology for the Moslem World, Jakarta, Indonesia, pp. 54, 2010.

[5] J. M. Clarence, S. Aravindh and A. B. Shreeharsha, *"Comparative Study of the New Generation, Agile, Scalable, High PerformanceNOSQL Databases"*, International Journal of Computer Applications, Vol. 48, Issue. 20, pp. 888-975, 2012.

[6] J. E. C. Cruz and I. C. A. R. Goyzueta, *"Design of a high availability system with HAProxy and domain name service for web services"*, In the Proceedings of the IEEE XXIV International Conference, Cusco, Peru, pp. 1, 2017.

[7] M. Deris, M. Rabiei, A. Noraziah And H.M. Suzuri, *"High service reliability for cluster server systems"*, In the Proceedings of the IEEE International Conference on Cluster Computing, Hong Kong, China, pp. 280, 2003.

[8] A. Dixit, A. K. Yadav and S. Kumar, *"An Efficient Architecture and Algorithm for Server Provisioning in Cloud Computing using Clustering Approach"*, In the Proceedings of the International Conference on System Modeling & Advancement in Research Trends, Moradabad, India, pp. 260, 2016.

[9] M. E. Elsaid and C. Meinel, *"Multiple Virtual Machines Live Migration Performance Modelling -- VMware vMotion Based Study"*, IEEE International Conference on Cloud Engineering (IC2E), Berlin, Germany, pp. 212, 2016.

[10] S. M. Hemam and K. W. Hidouci, *"Replicated Database Transactions Processing in Peer-To-Peer Environments"*, Journal of Networking Technology, Vol. 2, Issue. 1, pp. 63-72, 2011.

[11] G. Kecskemeti, P. Kacsuk, G. Terstyanszky, T. Kiss and T. Delaitre, *"Automatic Service Deployment Using Virtualisation"*, In the Proceedings of the Parallel, Distributed and Network-Based Processing, Toulouse, France, pp. 628, 2008.

[12] R. Khan, M. Haroon and M. S. Husain, *"Different technique of load balancing in distributed system"*, In the Proceedings of the Global Conference on Communication Technologies (GCCT), Thuckalay, India, pp. 371, 2015.

[13] C. H. Lien, Y. W. Bai, M. B. Lin and P. A. Chen, *"The saving of energy in web server clusters by utilizing dynamic server management"*, In the Proceedings of the 12th IEEE International Conference on Networks, Singapore, pp. 253, 2004.

[14] J. Liu, L. Xu, B. Gu and J. Zhang, *"A Scalable High Performance Internet Cluster Server"*, In the Proceedings of the The Fourth International Conference/Exhibition, Beijing, China, pp. 941, 2000.

[15] C. Mancaş, *"Performance improvement through virtualization"*, In the Proceedings of the RoEduNet International Conference - Networking in Education and Research (RoEduNetNER), Craiova, Romania, pp. 253, 2015.

© 2018, IJCSE All Rights Reserved

[16] M. C. Mazilu, *"Database Replication"*, Database Systems Journal, Vol. 1, Issue. 2, pp. 33-38, 2010.
[17] A. Tchana, L. Broto and D. Hagimont, *"Approaches to cloud computing fault tolerance"*, In the Proceedings of the International Conference on Computer, Information and Telecommunication Systems (CITS), Amman, Jordan, pp. 1, 2012.
[18] S. A. Moiz, P. Sailaja, G. Venkataswamy and S. N. Pal, *"Database Replication: A Survey of Open Source and Commercial Tools"*, International Journal of Computer Applications, Vol. 13, Issue. 6, pp. 0975 – 8887, 2011.
[19] T. Moyo and G. Bhogal, *"Investigating Security Issues in Cloud Computing"*, In the Proceedings of the Eighth International Conference on Complex, Intelligent and Software Intensive Systems, UK, pp. 141, 2014.
[20] E. Pacitti, M. T. Özsu and C. Coulon, *"Preventive Multi-master Replication in a Cluster of Autonomous Databases"*, Euro-Par 2003 Parallel Processing, Vol. 2790, Issue. 1, pp. 318-327, 2003.
[21] E. Pacitti, C. Coulon and P. Valduriez, *"Preventive Replication in a Database Cluster"*, Distributed and Parallel Databases, Vol. 18, Issue. 3, pp. 223–251, 2005.
[22] A. B. Prasetijo, E. D. Widianto and E. T. Hidayatullah, *"Performance Comparisons of Web Server Load Balancing Algorithms on HAProxy and Heartbeat"*, In the Proceedings of the Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, pp. 393, 2016.
[23] T. Agrawal and N. Sharma, *"Efficient Load Balancing Using Restful Web Services in Cloud Computing: A Review"*, International Journal of Scientific Research in Computer Sciences and Engineering, Vol. 6, Issue. 3, pp. 67–70, 2018.
[24] N. Sharma and S. Maurya, *"A review on:VM management in Cloud & Datacenter"*, International Journal Of Advanced Studies In Computer Science And Engineering, Vol. 6 Issue. 9, pp. 12-18, 2017.
[25] R. Shingade, A. Patil, S. Suryawanshi and M. Venkatesan, *"Efficient Resource Management in Cloud Computing"*, International Journal of Engineering and Technology, Vol. 7, Issue. 6, pp. 2045-2053, 2016.
[26] P. Devi, *"Attacks on Cloud Data: A Big Security Issue"*, International Journal of Scientific Research in Network Security and Communication, Vol. 6, Issue. 2, pp. 15–18, 2018.
[27] H. Tang, R. She, C. He, and Y. Dou, *"Construction and Application of Linux Virtual Server Cluster for Scientific Computing"*, In the Proceedings of the International Conference on Network and Parallel Computing, Shanghai, China, pp. 287, 2008.
[28] A. D. Tesfamicael, V. Liu and W. Caelli, *"Design and Implementation of Unified Communications as a Service Based on the Open Stack Cloud Environment"*, In the Proceedings of the IEEE International Conference on Computational Intelligence & Communication Technology, Ghaziabad, India, pp. 117, 2015.
[29] S. Varrette, M. Guzek, V. Plugaru, X. Besseron and P. Bouvry, *"HPC Performance and Energy-Efficiency of Xen, KVM and VMware Hypervisors"*, In the Proceedings of the 25th International Symposium on Computer Architecture and High Performance Computing, Porto de Galinhas, Brazil, pp. 89, 2013.
[30] M. Xia and G. Qin, *"The research and implementation of a highly concurrent and highly available system for acquiring personal virtual assets"*, In the Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, pp. 1, 2017.
[31] C. Yan, J. Shen and Q. Peng, *"Parallel Web Prefetching On Cluster Server"*, In the Proceedings of the Electrical and Computer Engineering Canadian Conference, Saskatoon, Sask., Canada, pp. 2284, 2005.
[32] H. Yuusuf and S. Vidalis, *"On the Road to Virtualized Environment"*, In the Proceedings of the Third International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, Romania, pp. 270, 2012.

Authors Profile

Mr. Devesh Dashora (Research Scholar) is pursuing Master of Technology (M. Tech.) with specialization in Computer Science and Engineering from Department of Computer Engineering, Faculty of Engineering, Pacific University (PAHER), Udaipur (Rajasthan). His keen interest is Cloud computing, distributed programming, Big Data analytics and processing, Machine Learning, IoT and Artificial Intelligence. He always excited to learn and implement new technologies. His vision is to develop something new and different in Information Technology. He has 5 years of teaching experience and 2 years of Research Experience.

© 2018, IJCSE All Rights Reserved