Azerbaijan Journal of High Performance Computing, Vol. 2, Issue 2, 2019, pp. 158-169
https://doi.org/10.32010/26166127.2019.2.2.158.169

*Correspondence:
Ha Huy Cuong Nguyen,
University of Danang, Viet Nam,
nhhcuong@cit.udn.vn

Consistency Maintenance in Distributed Cloud Storage Systems

Ha Huy Cuong Nguyen, Tung Trong Nguyen, Trung Hai Trinh

University of Da Nang, Danang, Vietnam, nhhcuong@cit.udn.vn, tungqn@donga.edu.vn, tthai@cit.udu.vn

Abstract

In this era, several organizations are storing their data on cloud storage to meet the requirements of efficient operation like stability, scalability, and availability of services. Data replication services in cloud storage systems are there to improve performance. In this context, the requirements for ensuring data consistency became increasingly important. In this paper, we propose a virtual server solution for updating replication, efficiently resolving resources, detecting, preventing deadlocks at the data center. In this manuscript, we have shown that the proposed solution yields the results for the schema, which ensures consistent data on costs and latency. In this paper, we use the Open Stack tool, which incorporates a proposed algorithm Balancing Consistency Availability On System Physical Machine, for maintaining data consistency in Cloud Storage Systems.

Keyword: Virtual machine, best-effort, distributed environments, Cloud Storage Systems, maintaining data consistency, Cloud computing

1. Introduction

Cloud Storage Systems are widely used because of their highly efficient response to distributed systems. The increasing size of the data is the main concern of our society. To maintain the storage file efficiently, we have focused on this manuscript. To minimize the time dispersed by the user, the cloud allows easy deployment of services to develop applications as required by users based on separate physical data centers. Data replication is an effective service (response time data, high availability of data, system performance) of the system storage applications in Cloud environments. The reason is that users can go directly to the copy in data centers (Data Center - DC) nearest copy center at the other (an example is shown below) However, data centers have problems such as load balancing, fault tolerance, redundancy, or lack of system resources (memory, processing).

Moreover, there are many replicas (R1, R2. . .) at the physical data centers (Figure 1) will make it more difficult to ensure data consistency. Therefore, in recent times, solutions using virtual servers are interested in research, application. That built on the physical server platform can overcome these problems, thus improving the efficiency of the scheme to ensure data consistency. Each solution to these problems by virtual

servers can be considered as an approach to improve an efficiency consistency maintenance scheme for cloud storage systems. We use virtual server solutions to ensure data consistency by Amazon Dynamo (Vishnumurthy, & Francis, 2006, April). Dynamo's partitioning scheme relies on a variety of consistent hashing. In their scheme, the resulting range or space of a hash function is considered as a ring. Every member of the ring is a virtual node (host) where a physical node may be responsible for one or more virtual nodes. The introduction of virtual nodes, instead of using fixed physical nodes on the ring, is a choice that provides better availability and loads balancing under failures. Each data item can be assigned to a node on the ring based on its key. The hashed value of the key determines its position on the ring. Data then is assigned to the closest node on the ring clockwise.

Moreover, data is replicated on the successive K 1 nodes for a given replication factor K, avoiding virtual nodes that belong to the same physical nodes. All the nodes on Dynamo are considered equal and can compute the reference list for any given key. The reference list is the list of nodes that store a copy of data referenced by the key. Dynamo is an eventually-consistent system. Updates are asynchronously propagated to replicas. However, in this paper, we propose solutions for virtual servers to work more efficiently, thereby improving the efficiency of the schema to ensure consistent data. Specifically, with the method for predicting the distributed systems with the least completion time, which is used to make a complex commercial decision in resource allocation and scheduling. The methods of detecting deadlock using two-way search algorithms can improve the efficiency and effectiveness of resource allocation in the heterogeneous platform. In this manuscript, the authors propose a novel solution that uses virtual servers to update the replicas. Specific contributions are:

1) It is a related resource allocation solution that enables detection and handling deadlocks for virtual servers.

2) Conduct experiments and compares them to prove the effectiveness of the new proposal.

The work is organized in the following way: in part 2, we have introduced the related works and background study. In section 3, we initiated the existing models; Section 4 deals with the solutions for the distribution of heterogeneous resources in the distributor, in Section 5, we have represented the results from our evaluation. Section 6 deals with the result analysis. This section concludes with the comments and suggestions for future work.

2. Related works

This section deals with the related study and background work. The problem of partitioning web- link graph for web ranking in P2P is formulated as a minimal cut-set with density balanced partitioning (Rowstron & Druschel, 2001, November). The problem is proved to be an NP-Hard by reducing to the minimum bisection problem (Li, Xie & Li, 2008; Li, Xie, & Li, (2008).). In P2P based Page Rank (Adami, Gabbrielli, Giordano, Pagano, & Portaluri, 2015, December), each computational peer contains a local web- link graph, and its PageRank is computed locally. P2P is a viable choice to address such limitations. To be able to compute the global ranking, a special node, so-called word-node, is constructed to store the link page information of the other peers.

Nassermostofi (2016). moves forward for authentication between familiar peers in P2P networking systems. A secure environment can only be achieved when peers are sure that they are communicating with the desired partner. There are two classes of P2P overlay networks: Structured and Unstructured. In structured P2P, strong consistency is provided by organizing heterogeneous nodes to an auxiliary structure on top of the overlay for updating propagation. Examples include the tree structure in SCOPE, the two-tiered structure in Ocean-Store, a hybrid of the tree, and a two-tiered structure. Nakashima, T., & Fujita, S. (2013, December) proposed an approach to building a tree by recursively selecting a representative and partitioning method. With this method, it supports only to store information of the tree structure in their subspace for all the intermediate nodes. So, the method becomes very difficult to update tree structure, and it can not be interested in the object are in the object's update dissemination tree. When we update the system applications with different problems, the process adds unnecessary overhead, which is being created while maintaining the tree from node failures. The systems cannot check error from heterogeneous nodes when increasing message request update (Pang, Wang & Zhang, 2013). The method is meant for constructing the tree structure dissemination by only involving heterogeneous nodes. In this paper, authors are interested in method maintenance and update propagation. Hence, the problem also efficiently builds the tree structure dissemination and binary tree decomposition. In this paper, in order to make it balanced and robust, the methods statistically optimized will be proposed in section 4. Vivek presented two types of bounded consistency that are provided with rumor spreading and replica chain. They have used this approach to ensure a certain probability of a new version that is being received in unstructured P2P systems. The probability is tuned by adjusting the redundancy degree in propagating an update to maintain the balance between the communications overhead with the consistency strictness. The message broadcast is an unstructured byte array that is delivered to all members of the group through the method probabilistic bounded consistency. In the previous work (Adami, Gabbrielli, Giordano, Pagano & Portaluri, 2015, December), the proposed method to be the best algorithm for constructing unstructured P2P graphs suitable for heterogeneous random selection. Here is a brief overview of other unstructured approaches. Adami, Gabbrielli, Giordano, Pagano & Portaluri. (2015, December) extends SCOPE by making both graph construction and query- resolution sensitive to node capacities. High-capacity nodes have higher degrees and are more likely to be traversed by random walks. While Swap links share these two features with GIA, Swaplinks exhibits more accurate control over the degree and probability of selection. Other examples of unstructured graph construction schemes include Araneola, an approach by Law and Siu, and Jianming Fu et al. None of these take node heterogeneity into account. The described mechanism by Jin, Ibrahim, Bell, Qi, Cao, Wu & Shi (2010) can be used as a random node selection primitive, but as was the case with the previously mentioned schemes, it does not take into account node heterogeneity.

In P2P systems, to avoid conflict relocation decisions in sequential nodes, it will contact the cached ancestors. With the method, the systems P2P in distributed systems are to retrieve the lost packets. The proposed model by Bermbach & Kuhlenkamp (2013, May) is described for consistency maintenance. Moreover, in

mentioned resource, it is extended by a consistency model for P2P applications. The used methods are a hybrid of push and pull, maintaining consistency. By Addition, they have also used to provide application tailored cache consistency, although each node can specify its consistency requirement. Proposed by Pang Wang & Zhang (2013) the model makes each node perform the strongest consistency maintenance from all its descendant nodes in the overlay replica hierarchy. The methods presented are meant for maintaining consistency at a node which cannot be reduced even. Therefore, we can see that most of the papers have not focused on the case study of strong consistency. In this paper, we have shown the solution methodology to update consistency in this above context.

3. Resource allocation in heterogeneous distributed platforms

In computing servers, several hardware and software resources are available for being configured. Hence allocation of resources should be in an efficient manner so that we may optimally utilize the resources. Resource allocation: Over the years, the model of cloud computing has drawn the attention of researchers Cloud users around the world. Cloud computing presents a model of resource allocation other than grid or scheduling. Especially, Amazon C2 is capable of allocating smaller computing resources, rather than a few, large requirements. The emergence of heterogeneity allows clouds to compete with traditional distributed computing systems, which often include many different architectures.

No same with the distribution system traditional, we can see a system does not even include files that are too connected by a network transport. Slow delays in transactions are a limit but not guess before.

Example 1 Many replica in CLOUD STORAGE SYSTEMS

We have used the platform graph for the grid platform. We model a collection of

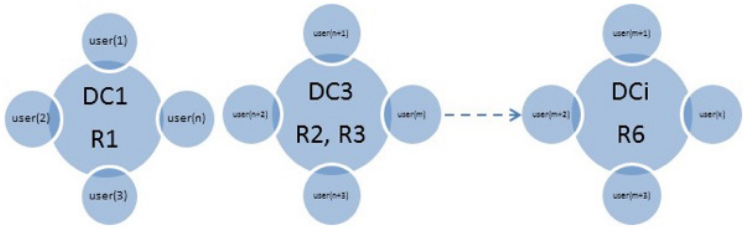

Fig. 1: Many replicas in CLOUD STORAGE SYSTEMS

heterogeneous resources as the nodes and the communication links between them as the edges of an undirected graph. See an example in Figure 2 with a simple platform on Industry 4.0. Industry 4.0 is indispensable are the following virtual machines provide services. Customers here are farmers who have the farms they may need to store data, retrieve data. A process can be in two states:

Running or blocked. In the running state (also called active state), a process has all the needed resources and is either executing or is ready for execution. In the blocked state, a process is waiting to acquire some resources.

Example 2 A requirement of data consistency

The Industrial Revolution Fourth widespread impact breakthroughs in technology in

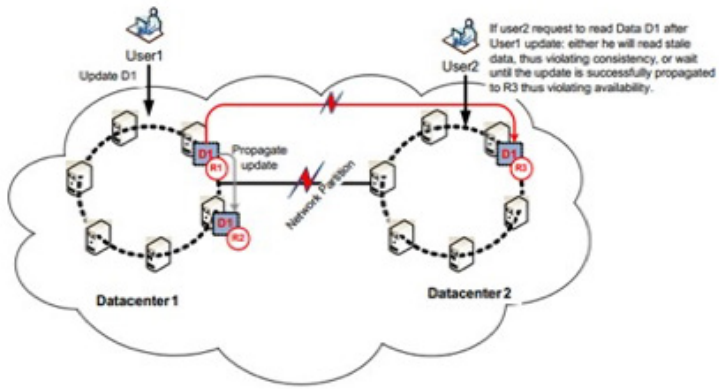

Fig. 2: A requirement of data consistency

all fields such as artificial intelligence, big data, Robotics, IoT.

The prospect of the development of education in the context of the Industrial Revolution for the fourth time, Vietnam will certainly face many difficulties in education we are still too heavy on the transmission of knowledge without looking to expand the quality and capacity of the learners; quality teaching staff, management staff has been uneven; information technology infrastructure is limited.

With the influx of new learning models and the development of science and technology, the methods of traditional education will certainly be challenging. Each student needs and different learning abilities. Advances in technology allowed the school can design individual learning pathways to suit each particular case. The educational software was put into use with the ability to adapt to the capabilities of each student and allow students with speeds matching the needs of the self.

Within the scope of the study, the author refers to the technology infrastructure solutions to solve the whole distribution of resources, avoid deadlock learners' area concurrent access to the system.

Resources allocation for virtual servers on the data consistency maintenance

Algorithm 1. Requests resources with best-effort

Input: $P^{j(CPU)*}$, $P^{j(RAM)*}$

from IaaS provider i;

Output: new resource $r^{CPU^{(n+1)}}$, $r^{RAM^{(n+1)}}$;

BEGIN

Operation request resource (r_i) in the critical section is

csstatei ←– trying; lrdi ←– clocki + 1; for each j ∈ R_i do

if (usedbyi[j] =0) the send request (lrdi,i) to pj end for;

senttoi [j] ←– true;

usedbyi[j] ←– R
else senttoi[j] ←–false
end if end for;
usedbyi[i] ki;
$\sum_{j=1}^{n}$ usedbyi[j] NPM);
wait(
csstatei ←– in;
Operation release resource (ri) in the critical section is
csstatei ←– out;
for each j ∈ permdelayedi do send permission(i,j) to pj end for;
R_i ← permdelayedi; permdelayedi ← ø END.

As presented above, ensuring data consistency, the virtual servers are responsible for propagating updates to replicas. In addition, the virtual server is responsible for processing any other requests from the user. Inside, the scale system is not stable, the heterogeneous user: the ability to process, bandwidth usage, the churn rate, the update rate. So, resource allocation (RAM, CPU) for the virtual server is very important. They need to be appropriate.

Moreover, solving other problems as deadlock will improve the efficiency of the virtual server in the consistency maintenance scheme. In the cloud computing model, as introduced above, the resources provided are gathered in so many complicated steps. The development of a solution to prevent deadlock needs to ensure that at least one of the following conditions cannot occur.

In the cloud computing model, as introduced above, the resources provided are gathered in so many complicated steps. The development of a solution to prevent deadlock need to ensure that at least one of the following conditions cannot occur: Resources cannot be shared, occupied, and the additional resources required, no recovery resources, existence of any cycle or not.

Virtual host distribution on the physical node at a time point. To define the distribution

Algorithm 2.Balancing Consistency Availability On System Physical Machine

Input: $P^{j(CPU)*}$, $P^{j(RAM)*}$
from IaaS provider i;
Output: new resource $r^{CPU^{(n+1)}}$, $r^{RAM^{(n+1)}}$;
BEGIN
When REQUEST(k,j) is received from pj do
clocki ← max($clock_i$,n);
prioi ← (csstatei = in) ∨ ((csstatei = trying)
∧ ((lrdi,i) ¡ (n,j)));
if (prioi) then send NOTUSED(NPM) to p_j
else if(n_i
NPM) then send NOTUSED(NPM - n_i) to p_j end if
permdelayedi ← permdelayedi ∪j

end if.
When permission(i,j) is received from p_j do
NPMi ← NPMi \ j;
When NOTUSED(x) is received from p_j do
usedbyi[j] ← usedbyi[j] -x;
if ((csstatei = trying) ∧ (usedbyi[j] = 0) ∧ (notsenttoi[j])
then send REQUEST(lrdi,i) to pj senttoi[j] ← true;
usedbyi[j] ← NPM;
end if.
END.

of all of a virtual host on the physical nodes in the time required, starting from time t and drag long d seconds is very difficult. When combine and force the best algorithm 2, we got the previous time, and the algorithm used to provide the resource in the shared environment because they do not be up the calendar and require the best.

When the nodes have been sorted, the model uses the best-effort algorithm to distribute all VM's.

The algorithm mentioned above can distribute multiple VM's on the same node. With this research, we are aiming to provide efficient distribution of resources; we have proposed the following technique in distributed environments. In this case, the algorithm tries to distribute as many VM's as possible on multiple physical nodes.

A. Resources Allocation for virtual servers on the data consistency maintenance

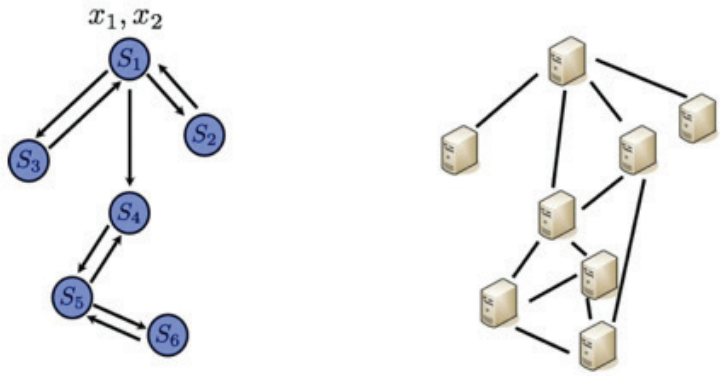

Fig. 3: Example problem instance with two nodes and one service, showing possible resource allocations

In this paper, the practical implications dictate that designers opt for best-effort availability, thus guaranteeing consistency and greedy consistency for systems that must guarantee availability. With a pragmatic way to handle the tradeoff is by balancing consistency availability tradeoff in systems.

Then we process the research for the rating to provide the resource, which can be a CPU hardware. In the future, we will the progress of the virtualization as the storage drive, compatibility, and time complete when sending a contract. p, percentage of the CPU used by a requested. (Value of p can be 10%, 20%, 30%, 40%, and 100%

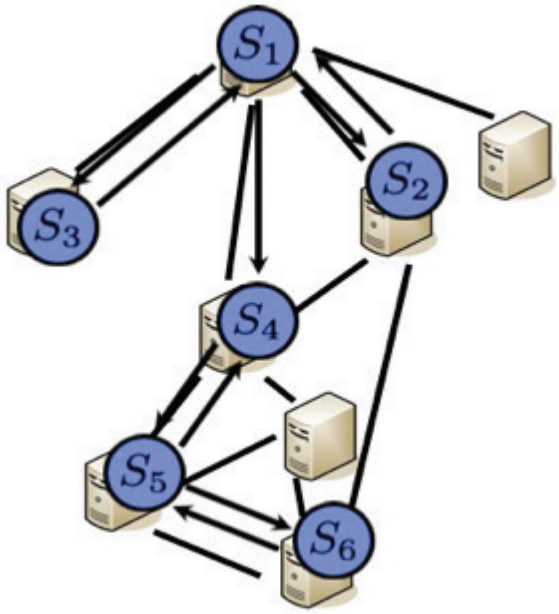

Fig. 4: Example problem instance with two nodes and one service, showing possible resource allocations.

because the percentage of CPU for using the greedy algorithm is calculated as approximately 89.20%).

VM, the number of nodes required. These are approximate as follows: small (40), medium (60), large (80). With the above two parameters, the research team determined the times to collect the results of the time when it requires to use the greedy algorithm within 1 lease contract in 1 DC and the time when the deadlock detection algorithm detected on deadlocks.

Result analysis

This section deals with result analysis and discussion. In this way, the solution includes two steps: Firstly, the Creation of a structured tree; Secondly, checking if the cycle exists in that tree. In the first step, multicast to be used in dDT algorithm construction in order to send the request to on the nodes in the systems and waiting for replies from them. After this process, we obtain a tree. In the second step, another algorithm to be used in order to detect a cycle. However, this algorithm has not been mentioned in this paper.

Through Table 1, they apply algorithm request resource with the ability to provide resources for a lease (as long as the capacity of the CPU is predetermined), we recognize that the ratio between the rent Defective and successful creation of virtual

Table 1: Comparison average time a contract ends with greedy algorithm, together with the cpu usage at local Environment

Ability	VM NOSR	VM SR	VM ID	Start Time	Start time	End time
10%	30,56	40,72	20,5	0.1	10	22.22%
20%	40,09	45,21	22,3	0.1	10	20.00%
30%	45,30	45,48	23,18	0.1	32	27.27%
40%	47,54	45,45	24,34	0.1	45	43.75%
50%	45,56	46,01	30,14	0.1	40	39.39%
60%	46,18	47,01	40,01	0.1	20	36.05%
70%	47,02	47,09	43,09	0.1	35	42.86%
80%	48,08	47,56	45,23	0.1	48	44.44%
90%	49,02	48,56	45,67	0.1	60	50.55%
100%	56,78	50,06	60,78	0.1	90	64.86%

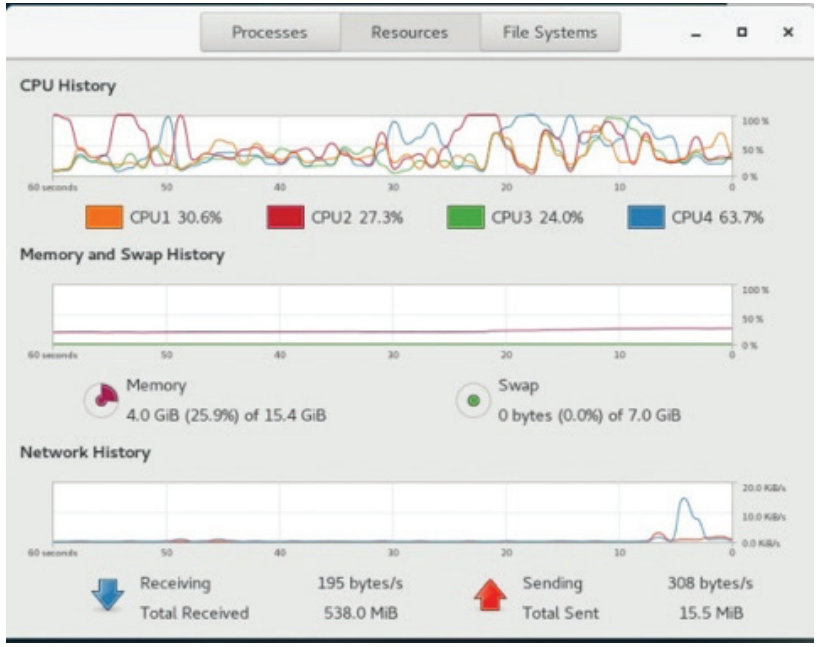

Fig. 5: Graph showing the ability of CPU to each lease contract with the best-effort algorithm.

machines is the same. With the CPU of 70%, we can see that the difference between these ratios is greater, with the failure of 88% and the success of 65, 6%.

In traditional distributed storage and cloud computing systems, the instinctive and correct way to handle replicas consistency was to ensure a strong consistency state

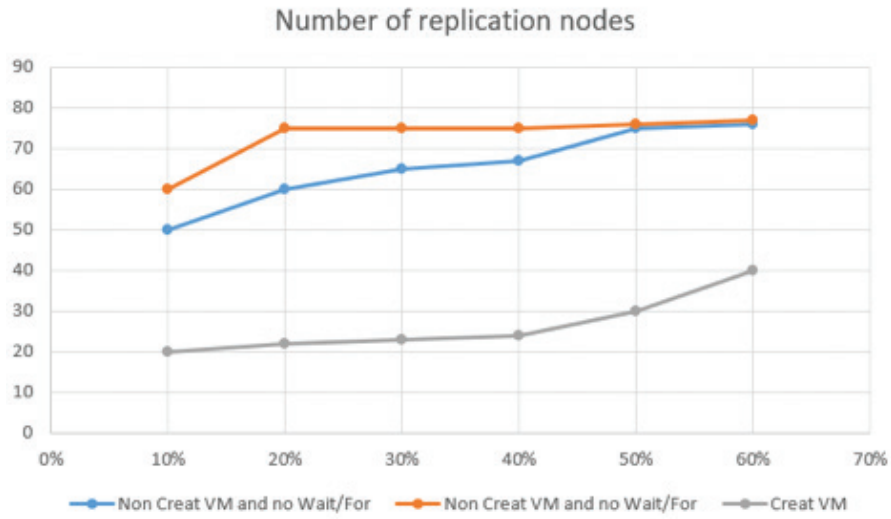

Fig. 6: Time contract graph with the ability of each CPU to provide virtual server creation resources.

Table 2: Mean attenuation limit with more experiments

Ability	VM NOSR	VM SR	VM ID	Start Time	Start time	End time
10%	50,56	60,72	20,5	0.1	5	22.22%
20%	60,09	75,21	22,3	0.1	10	20.00%
30%	65,30	75,48	23,18	0.1	12	27.27%
40%	67,54	75,45	24,34	0.1	15	43.75%
50%	75,56	76,01	30,14	0.1	20	39.39%
60%	76,18	77,01	40,01	0.1	23	36.05%
70%	77,02	77,09	43,09	0.1	25	42.86%
80%	88,08	78,56	45,23	0.1	28	44.44%
90%	89,02	78,56	45,67	0.1	30	50.55%
100%	96,78	80,06	60,78	0.1	35	64.86%

of all replicas in the systems all the time. Through Table 2, to provide resources for 10 lease contracts (given the condition that the ability of CPU is predetermined), we found that the success rate to create VM is very high, with the CPU's ability at 20%. As for CPU's ability at 40% and 60%, the success creation is also higher than that of failure creation.

We compare our algorithms with algorithms greedy with detecting and eventual balancing consistency. The first tolerable status success create virtual machine rate 70

Conclusion

VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	RX/TX factor	ID	Public
4	8GB	80GB	0GB	0MB	1.0	4	Yes
2	4GB	40GB	0GB	0MB	1.0	3	Yes
1	2GB	20GB	0GB	0MB	1.0	2	Yes
1	512MB	1GB	0GB	0MB	1.0	1	Yes
8	16GB	160GB	0GB	0MB	1.0	5	Yes

Fig. 7: Graph showing lease contract completion time for each CPU capability in distributed environments, using the deadlock detection algorithm.

In the content of the article, we provide an Open Stack virtual server solution. The solution concerns the readiness criteria because it affects the cost of preparing the infrastructure. Virtualization solutions based on Open Stack have great potential to meet the needs of the users in the context of complex and sophisticated intelligent computing systems.

Using a virtual machine is an effective solution that ensures consistent data replication of Cloud Storage Systems. In this paper, the team proposed solutions to improve the performance of virtual servers in allocating system resources, algorithms to prevent, handle deadlock. Experimental results indicate that new proposals are effective. Main issues, the ability to adjust resources can positively impact each user's needs, the solution to ensure quality standards, ensure sufficient resources, storage space, support policies Use when demand increases or decreases. Standalone independence also improves on open resource standards, based on physical server resources.

We have also conducted experiments in distributed environments, some peer-to-peer distributed applications with the ability of data centers to change. Based on empirical evaluation criteria, we propose to bring some positive results.

When observing a comparative assessment between the ability to create a VM as required, or reject a request to create a VM as the other VMs can not be suspended, or stop the CPU in the Data Center.

Through this research, we found that adopting Open Stack based virtual server solutions delivers optimal performance for the distributed resources of specially adapted virtual machine systems. Provide virtual servers for high tech applications in intelligent farming.

Acknowledgment

The article was conducted with support from the scientific research project code 182.CNTT02 / HD- DHCN.

References

Jin, H., Ibrahim, S., Bell, T., Qi, L., Cao, H., Wu, S., & Shi, X. (2010). Tools and technologies for building clouds. In *Cloud Computing* (pp. 3-20). Springer, London.

Shen, H., Liu, G., & Chandler, H. (2015). Swarm intelligence based file replication and consistency maintenance in structured P2P file sharing systems. *IEEE Transactions on Computers, 64*(10), 2953-2967.

Li, Z., Xie, G., & Li, Z. (2008). Efficient and scalable consistency maintenance for heterogeneous peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems, 19*(12), 1695-1708.

Nassermostofi, F. (2016). Toward authentication between familiar Peers in P2P networking systems. In *Proceeding of the 9th GI Conference Autonomous Systems* (pp. 88-103).

Pang, X., Wang, C., & Zhang, Y. (2013). A new P2P identity authentication method based on zero-knowledge under hybrid P2P network. *TELKOMNIKA Indones. J. Electr. Eng, 11*(10), 6187-6192.

Adami, D., Gabbrielli, A., Giordano, S., Pagano, M., & Portaluri, G. (2015, December). A fuzzy logic approach for resources allocation in cloud data center. In *2015 IEEE Globecom Workshops (GC Wkshps)* (pp. 1-6). IEEE.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., & Kubiatowicz, J. D. (2004). Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications, 22*(1), 41-53.

Rowstron, A., & Druschel, P. (2001, November). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing* (pp. 329-350). Springer, Berlin, Heidelberg.

Chen, X., Ren, S., & Wang, H. (2005, March). SCOPE: Scalable consistency maintenance in structured P2P systems. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* (Vol. 3, pp. 1502-1513). IEEE.

Vishnumurthy, V., & Francis, P. (2006, April). On heterogeneous overlay construction and random node selection in unstructured p2p networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (pp. 1-12). IEEE.

Bermbach, D., & Kuhlenkamp, J. (2013, May). Consistency in distributed storage systems. In *International Conference on Networked Systems* (pp. 175-189). Springer, Berlin, Heidelberg.

Nakashima, T., & Fujita, S. (2013, December). Tree-based consistency maintenance scheme for peer-to-peer file sharing systems. In *2013 First International Symposium on Computing and Networking* (pp. 187-193). IEEE.

Submitted 25.09.2019
Accepted 16.11.2019