IJCTA, 9(22), 2016, pp. 301-305
© International Science Press

Improved Load Management in Cloud Environment using MHT Algorithm

Akhilesh Kumar Bhardwaj* Dr. Rajiv Mahajan Dr. Surinder*****

Abstract : Proficient routines are desirable to make sure the effectual load supervision of contract out data on un-trusted cloud servers. This paper exhibits an overview of fundamental mechanisms of cloud computing in opening segment and illustrates consideration of load management improvement in the later one. Load management for autonomous tasks is a key issue in cloud atmosphere. In this paper, we propose a valuable algorithm named Merkle hash tree algorithm for suitable load management. Our article put forward an improvement over enhanced bee colony (EBC) algorithm meant for competent and valuable load management in cloud surroundings. The process also seeks to shorten makespan time along with the amount of task migrations. The experimental results illustrate considerable development in favor of the service quality offered to the customers.

Keywords : Cloud Computing, Load Management, MHT Algorithm, EBC Algorithm, Makespan time, Task Migration.

1. INTRODUCTION

The initiative at the back of cloud technology (CC) came into the reality after the expansion of grid, parallel and distributed computing [1]. Cloud technology takes account of reliability, virtuality, deliberate service, scalability, adaptability, enlarged intelligence, efficient automatic controlling and elevated service quality [2]. Mutual verification is the most considerable challenge in cloud. Several verification methods can be used for authenticating the customer. Few verification methods like plain password verification can be implemented with ease. Though, they are meager and prehistoric [3]. Formulate the use of a trustworthy third party auditor (TPA) can be a better way of assuring data security [4]. Cloud computing formation encloses two establishment layers: a management layer (ML) and a virtualization layer (VL) [5]. Management layer executes elements to assist the cloud operations. These elements incorporate intrusion detection, scheduler, hypervisor interface, load distribution, validation engine, external & internal API and virtual jobs element. On the other end, the Virtualization layer is enclosed by servers and platforms covering hardware which are virtualization enabled. Cloud load management and scheduling problems are considerably NP hard problems. To ensure service quality, suitable load management and scheduling is required between nodes in the scattered cloud atmosphere. A competent load management mechanism attempts to speed up the time of execution for customer desired services. It also helps in squeezes system disproportion and provides a reasonable access to the customers. Improved load management produces good QoS parameters like scalability, response time proper resource utilization and fault tolerance. Task migration time can also be improved with superior load management. The upgrading in these features surely ensures excellent service quality for the customer's applications. The active character of cloud computing circumstances desires a dynamic algorithm for resourceful load management in between the nodes.

* Research Scholar, IKG Punjab Technical University, Punjab Email: akhilesh.akhand@gmail.com
** Professor & Principal, G.C.E.T., Punjab Email: rajivmahajan08@gmail.com
*** Assistant Professor, G.T.B.C., Punjab

2. MERKLE HASH TREE ALGORITHM (MHT)

Distributed systems utilize an efficient data structure called Merkle trees to compute the hash of a log with k events such that the contents of an individual event can be verified in O (log *k*) time. The leaves are the hashes of each individual event. These hashes are incrementally combined with the hashes of their neighbors as we travel up the tree. The root hash is actively a hash of the whole database, computed using the hashes of the left and right subtrees. The true advantage of the Merkle tree is when we want to prove that the i^{th} record has not been tampered with. Suppose a customer has obtained the root hash from a trusted server and then retrieved an edge from an untrusted host. To verify the integrity of the edge, the customer needs the untrusted host to also send it the hashes so that it can reconstruct the path leading from the edge to the root hash. Hence, verifying that any given record r is in fact the i^{th} event in the log only requires transmitting and computing "i–1" hashes and not the full tree. This sequence of hashes is called an audit path. Once an event has occurred, its presence in the log must be preserved [6].

3. LITRATURE SURVEY

Enhanced bee colony algorithm has been presented in paper [7]. The authors have modified the bee colony algorithm to make the load management more powerful, more realistic. Here load management metrics like makespan time and task migration has been widely explored. In paper [8], load management based on improved throttled algorithm has been shown with superior response time as against to existing throttled and round-robin algorithms. Load management based on Ant colony algorithm has been proposed in paper [9]. The theme is developed on pheromone deposition. Most of the ants are fascinated towards the node having least load. So highest pheromone deposition crops up at that node and the system performance is improved. Response time based load management has been expressed in paper [10]. The suggested model reflects about the present responses and its variations to pick the distribution of fresh received requests. This model helps in eliminating the requirement of needless communication.

In paper [11], a resource intensity aware load management algorithm is projected. The algorithm realizes least-cost and quicker convergence to the load equilibrium state. It also curtails the probability of future load discrepancy. In paper [12], artificial bee colony algorithm based load management mechanism was proposed. Cloud throughput is optimized by emulating the behavior of honey bees.

4. PROPOSED ARCHITECTURE

The customers put forward their miscellaneous applications to the Cloud Service Provider (CSP) through a communication gateway [13]. The requests from the customers are lined up in data center of the CSP. The secondary servers are then verified for the least load with the performance level of the central processing unit for the presently executing task. The requested job is then allocated by CSP to the secondary servers having least load to process the task. Accordingly, the Customer requested job will be allotted to the available secondary server which contains least load and it is concerned to process the requested job of the customers.

The objective is alienated into several components including design and implementation of the proposed protocol to ensure suitable load management with parameters like makespan time, migration time, fault tolerance, response time and scalability. In the proposed MHTA (Merkle Hash Tree Algorithm) model, MHT supports mutually in load management and files compression. At this point, TPA backs the cloud client for assuring the dynamic data integrity. For the period of the audit, the TPA verifies that client data is left unharmed or not. Thus ,the cloud computing stage of economy is achieved properly.

Steps of working of projected system are as follows :

Step 1 : Establishment of Validation

Step 2 : Read the input file

Step 3 : Authentication

Step 4 : File Sharing

Step 5 : Encryption with Key Generation

Step 6 : MHT Block Division

Step 7 : Decompression

Step 8 : Decryption

Step 9 : Load Management by MHT

Load Management using MHT algorithm has been stepped in as: A file is divided up into k number of data blocks. Every block is hashed and these hashes of blocks are the leaves in the hash tree. Nodes in the tree are the hashes of their respective children. Concluding hash value in a solo node becomes a top hash value.

6. EXPERIMENTAL RESULTS

The proposed procedure is done in a real time development with Net Beans IDE 8.1 framework, JDK 1.8 with My SQL. In this assorted background, diverse specification based VMs are taken into consideration. Cloudlets with capricious conditions are put forward into this cloud surroundings. The amount of makespan and migrations are considered and evaluated with existing enhanced bee colony algorithm. The makespan time of the suggested system with enhanced bee colony algorithm is out in Table 1. The makespan time is graphically symbolized in Fig. 1.

Table 1. Makespan Time Comparison.

Sr. No.	*No. of cloudlets*	*Enhanced bee colony algorithm (in Seconds)*	*MHT algorithm(in Seconds)*
1.	10	43.85	35.66
2.	15	68.85	60.76
3.	20	78.85	70.23
4.	25	100.1	90
5.	30	118.85	106

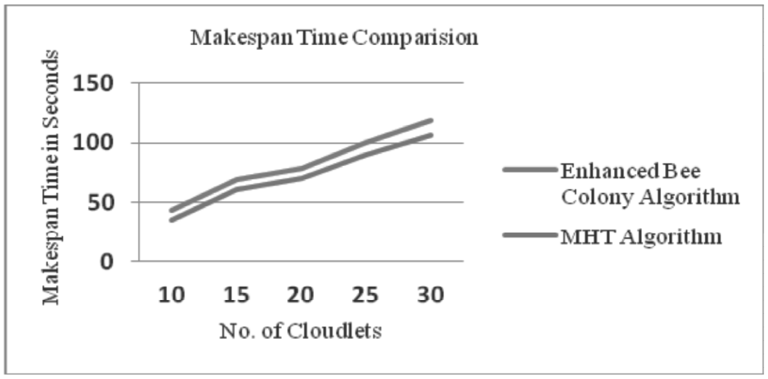

Fig. 1. Makespan Time Comparison (*a*)

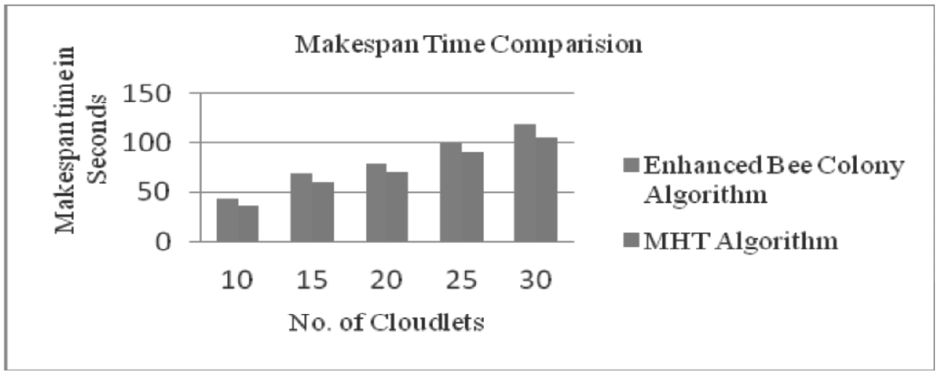

Fig. 1. Makespan Time Comparison (*b*)

The above results clearly exhibit that makespan time is trim down into a noteworthy amount while using Merkle hash tree algorithm and subsequently the customers will find quick response as compared to the previous methods. If the number of task migrations is larger then it will critically influence the cloud performance and in that way reduces the service quality. A superior scheduling and load management mechanism will definitely reduce the number of task migrations. For better results, the projected scheme is also considered for the number of task migrations. The outcome is derived in the Table 2. The end result in the Table 2 illustrates that the number of task migrations is shortened while using Merkle hash tree algorithm. The number of task migration is given in the Fig.2.

Table 2. Task Migration Comparison.

Sr.No.	*No. of cloudlets*	*Enhanced bee colony algorithm*	*MHT algorithm*
1	10	2	1
2	15	3	1
3	20	7	2
4	25	11	4

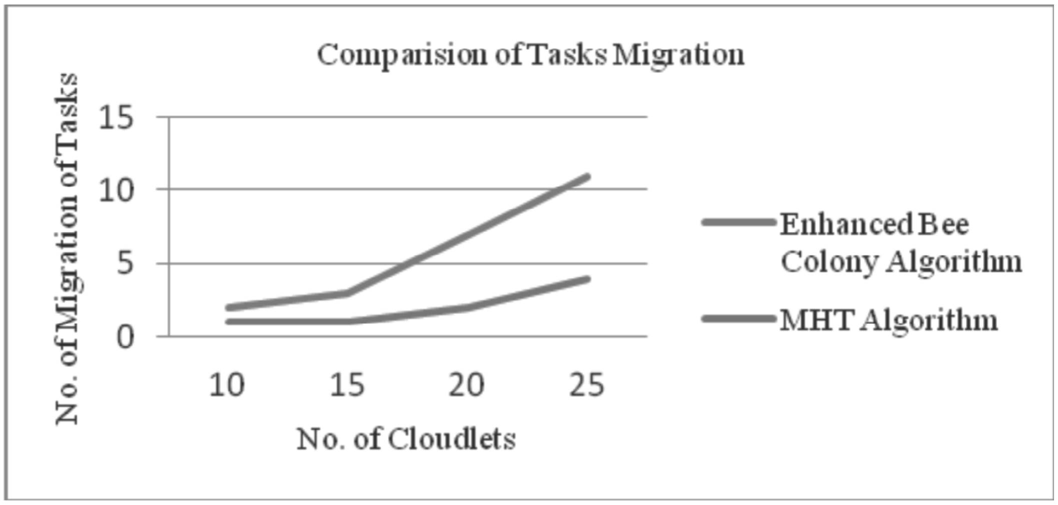

Fig. 2. Comparison of Tasks Migration (*a*)

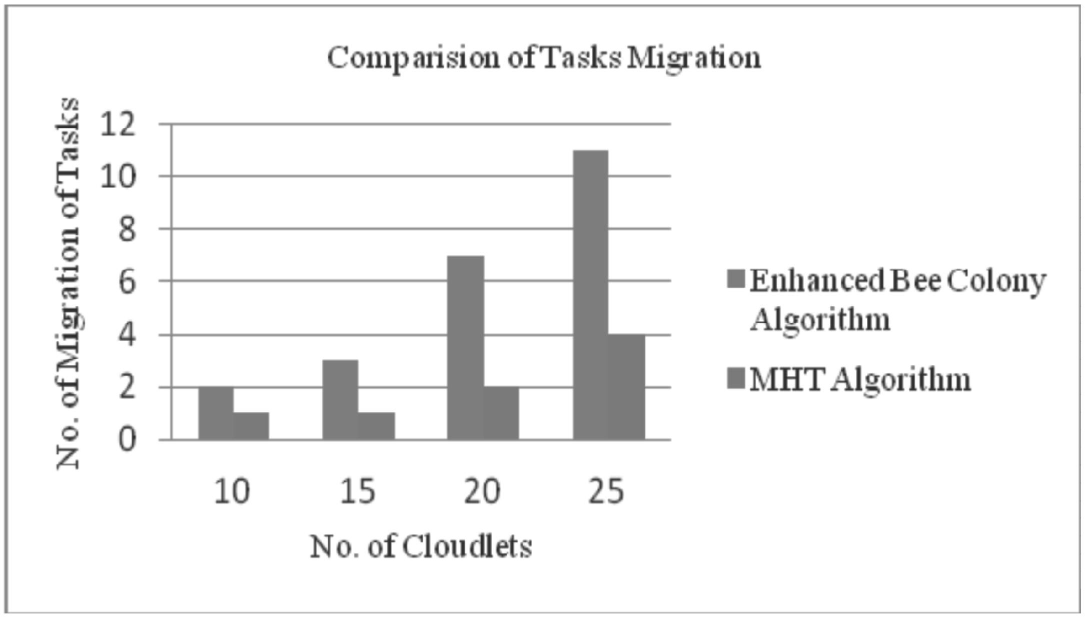

Fig. 2. Comparison of Tasks Migration (*b*)

The above experimental outcomes sum up that Merkle hash tree algorithm not only reduces the makespan time but also the number of task migrations compared to the stated existing enhanced bee colony algorithm. Therefore it facilitates efficient utilization of computational resources in the cloud scenario. In view of the fact that the algorithm diminishes the completion and reduced number of task migrations, it will surely furnish better QoS to the customers.

7. CONCLUSION

This paper suggests Merkle Hash Tree algorithm for well-organized load management in cloud conditions. The investigational outcomes verify that the projected hash based algorithm do better than the available enhanced bee colony algorithm by minimizing the makespan time and number of task migrations and so furnish better service quality to the customers. The projected work can also be used in future for different load management and scheduling metrics or in association with dissimilar algorithms for superior outcomes in diversified phase of load measurement domains.

8. REFERENCES

1. Wang En Dong WU Nan, LI Xu, *"QoS-oriented Monitoring Model of Cloud Computing Resources Availability", In the Proceeding of IEEE International Conference on Computational and Information Sciences, pp. 1537-1540, 2013.*
2. Maha Attia Hana, *"E-Government Cloud Computing Proposed Model: Egyptian E_Government Cloud Computing", In the Proceeding of IEEE International Conference on Advanced in Computing Communication and information (ICACCI), pp 847-852, 2013.*
3. Nimmy K., M. Sethumadhavan, *"Novel Mutual Authentication Protocol for Cloud Computing using Secret Sharing and Steganography", In the Proceeding of IEEE fifth International Conference on Application of Digital Information and Web Technologies (ICADIWT), pp.101-106, 2014.*
4. Shuai Han, Jianchuan Xing, *"Ensuring Data Storage Security Through a Novel Third Party Auditor Scheme in Cloud Computing", In the Proceeding of IEEE International Conference on Cloud Computing and Intelligence System (CCIS), pp. 264-268, 2011.*
5. Patrascu, V.V. Patriciu, "Logging for Cloud Computing Forensic Systems", International Journal of Computer Communication and Control, ISSN 1841-9836, 10(2):222-229, April, 2015.
6. karthikeyan bhargavan, *" merkle hash tree for distributed audit logs", 2015*
7. K.R. Ramesh Babu, Phillip Samuel, *"Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud", Innovations in Bio-Inspired Computing and Applications, Advances in Intelligent Systems and Computing 424, DOI 10.1007/978-3-319-28031-8_6, Springer, 2016.*
8. Domanal, S.G.R., Ram Mohana, G.: *Load balancing in cloud computing using modified throttled algorithm. In: IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–5, 2013*
9. Dam, S., Mandal, G., Dasgupta, K., Dutta, P.: *An ant colony based load balancing strategy in cloud computing. Springer Advanced Computing, Networking and Informatics, Vol. 28, pp. 403–413, 2014.*
10. Sharma, A., Peddoju, S.K.: *Response time based load balancing in cloud computing. In: International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1287–1293, 2014*
11. Chen, L., Shen, H., Sapra, K.: RIAL: *resource intensity aware load balancing in clouds. In: IEEE Conference on Computer Communications (INFOCOM), pp. 1294–1302 (2014)*
12. Yao, J., He, J.: *Load balancing strategy of cloud computing based on artificial bee algorithm. In: IEEE 8th International Conference on Computing Technology and Information Management (ICCM), pp. 185–189, 2012*
13. S.Shashikala, T. Karthick, *"Privacy preserving and Load Balancing for secure cloud storage", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p- ISSN: 2278-8727, volume 16 issue 1 ver. Iv, pp 102-106, 2014.*