The Academic
International Journal of Multidisciplinary Research

An Online Peer Reviewed / Refereed Journal
Volume 1 | Issue 2 | July-September 2023
ISSN: 2583-973X
Website: www.theacademic.in

Clustering-Based Approach for Job Scheduling in Cloud Environment to Mitigate Workload

M. H. Vahitha Rahman [1], Dr. M. Vanitha [2]

[1] Ph.D Research Scholar, Department of Computer Applications, Alagappa University, Karaikudi, India,
[2] Assistant Professor, Department of Computer Applications, Alagappa University, Karaikudi, India
[1]vahithanadeer16@gmail.com, [2]vanitham@alagappauniversity.ac.in

ARTICLE DETAILS

Research Paper

Article History
Received : September 15, 2023
Accepted : September 26, 2023

Keywords :
Workload , Load Balances, ant colony optimization, optimum clustering, cloud computing.

ABSTRACT

Load balancing has a huge effect on how well a cloud computing system performs. Load balancing that works well increases cloud security and improves the user experience. In the sections that follow, we will explore a new method of load balancing in the public cloud, one that makes use of the idea of cloud subdivision. This component has a toggle switch built in so that different methods may be used depending on the situation. In this study, we focus on the critical topic of load balancing in cloud computing settings. The implement strategy improves the load balancer's reliability by facilitating early plan validation. In order to develop a task distribution plan with worldwide search capabilities, as evaluated by the performance function of computing resources, the load balancing approach, which takes its cue from ant colony behaviour, does a thorough review of all physical hosts. Load balancing in the cloud might be complicated, but there are numerous approaches to figuring it out. By comparing and contrasting the merits and drawbacks of different strategies, we may find a fresh and efficient approach to load balancing that can be used in the future.

1. INTRODUCTION

According to Khiyaita [2], the problem of load balancing in cloud data centers has attracted a substantial amount of interest in the academic community. In order to successfully achieve load balancing in cloud

data centers, it is essential to make an effective and efficient selection of the best possible physical host during task deployment. The vast bulk of work done in this field before has been devoted to finding solutions to the problem of attaining instantaneous load balancing [9] within a single iteration of the algorithm. The objective of the approach that has been suggested is to determine which employment distribution techniques are suitable for the circumstances that now exist. The excessive emphasis on the optimal load balancing approach [13] for the current deployment problem presents a limitation, which results in reduced efficiency and an unnecessary prolonging of the waiting time for consumers. In addition, establishing excellent service performance may be easily done thanks to the enormous computing resources that are available in cloud data centers. These resources often exceed the needed amount despite the real-time and demanding nature of the demands being placed on the service. As long as the overall scheme constantly tends toward a long-term ideal load balancing impact, it is not necessary to guarantee that the optimal load balancing effect is attained after each algorithm cycle in real time, as stated in the preceding premise. This is because it is sufficient to ensure that the entire scheme tends towards a long-term ideal load balancing impact.

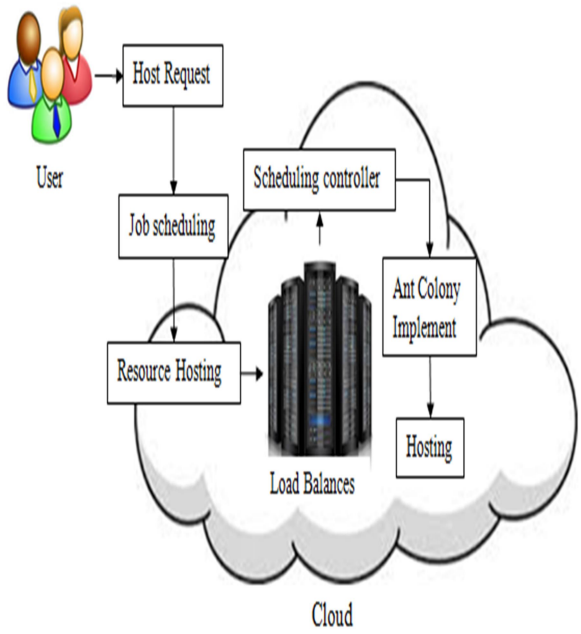
PLAN FOR A SYSTEM

The method that has been presented for estimating Ant state exemplifies a way for efficiently performing the process of load balancing that is based on optimization, while maintaining a low degree of algorithmic complexity. As a consequence of this, it provides effective management performance as well as efficient computing performance on both fronts. The following is a list of the major responsibilities that need to be met: The purpose of this work is to present the idea of process-based load adjustment in distributed computing settings, which was first conceived of by G. Nan in 2014[5]. This method is distinct from the traditional load adjusting systems that are covered in the current body of scholarly research. There are several advantages to take into consideration, including lowering the amount of superfluous computational complexity, raising the level of operational efficiency, and satisfying the performance needs of consumers. From the point of view of cloud endpoints in subterranean insect colonies, the most important thing to consider is the possible long-term advantages of improving the functions of external management and the resources that are used.

The purpose of this research is to propose a model of the behavior of subterranean insects. The model focuses on the most efficient distribution of tasks in order to calculate the likelihood of finding suitable physical hosts. This research suggests using the cluster-based job allocation method, which takes into

account the probabilities associated with ant colonies, in order to ascertain the ideal arrangement of prospective physical hosts. Within each iteration of the computing process, this study suggests an information structure that might be applied to the matrix to determine the final arrangement vector.

DESIGN OF COMPLEX SYSTEMS



CONSEQUENCES AND EFFECTIVENESS

The Algorithm of an Ant Colony

In this part, we elucidate the precise procedure of the Ant colony. The following information is provided:

The estimated constraint value Li has been determined for each physical host i inside the cloud data center. The empty set NPH is defined as NPH = {}. The performance constraint value of the set TR, which represents task requests, is defined as the maximum amount of resources requested in NPH. The maximum requested resource quantity, denoted as L_mreq, for TR may be determined using a specific formula. If the value of Li is greater than L_mreq, then host i will be included in the set NPH. After evaluating the constraint value Li of each physical host and comparing it with the performance

constraint value L_mreq, a new candidate set NPH = {nph_1; nph_2; ... ; nph_m, m≤m} is derived. This candidate set will be used for the subsequent clustering procedure involving physical hosts.

Input: The current value of the processing power, the current value of the available local computation , the current value of the available local memory , the received value of the task requirement , the received value of the computation requirement , and the received value of the memory requirement . Output: The final deployment solution vector .

In the first step, the null hypothesis and its complement are both empty, and the statistical parameter (S) is set to null.

Step 2: For each element tr_i belonging to the set TR, do the following actions.

Step 3: The equation for calculating R_i is given by R_i=αR_c^i+βR_mem^i. Step 4: The loop ends. Step 5: The maximum value of RR_i, where i ranges from 1 to n, is assigned to L_mreq. Step 6: For each 〖ph〗_i belonging to the set PH, perform the following:

In the seventh step of the process, the equation L_i = ∝l_c^i+βl_mem^i is used.

In the event that the value of L_i is greater than L_mreq, go to step 8.

In Step 9, the set NPH is updated by adding the element 〖ph〗_i to it, resulting in the new set NPH = NPH ∪ 〖ph〗_i. Following this, in Step 10, the conditional statement ends.

Step 11: Termination of Loop

In Step 12, the algorithm iterates over each element 〖nph〗_i in the set NPH.

The posterior probability P(B_i| A) of 〖nph〗_i is derived using the following formula:

Step 14: Terminate the loop. Step 15: Let 〖nph〗_j represent the candidate physical host with the highest value of Pj = 〖max〗_(i=1)^m' p(B_I | A)in NPH. Step 16: Update NPH' by adding 〖nph〗_j to its existing elements. Step 17: For each 〖nph〗_i belonging to the set NPH, perform the following:

In Step 18, SD(〖nph〗_i nphj) is calculated using formula (17). In Step 19, a comparison is made to see whether SD(〖nph〗_i nphj) is greater than the U_hreshold.The user's text can be rewritten to be more academic as follows: The concept of similarity

Step 20: The value of NPH' is updated to be equal to NPH' when nphi is true. Step 21: The if statement ends.

Step 22: Termination of the loop. Step 23: Iteration over each element 〖tr〗_(i) in the set TR.

In step 24, the variable S[i] represents the number of the physical host, denoted as 〖nph〗_i^', that has the highest remaining resource amount, Li, among all the physical hosts in NPH'. In step 25, the variable L_i is updated by subtracting the value of R_i from the current value of LL_i.

Step 26: Terminate the loop. Step 27: Output the value of S and conclude the program.

EXPERIMENTAL RESULTS

The evaluation of the framework may be conducted by considering criteria such as the dynamic load balancing of resource allocation [7]. (ii) This study examines the impact of load adjustments on Bayesian bunching, specifically focusing on the differences between various use assessment systems.

Consolidating these estimates requires altering the limit of the work stack and assigning resources on a part chart, both of which are processes that take place on a part chart. In the future, calculations to determine the condition of subsurface insects will be implemented. These calculations will effectively use up 90 percent of the effort for radiators.

CONCLUSION:

In the context of cloud computing, load balancing is an essential activity that must be carried out in order to make the most efficient use of available resources. Despite the fact that the integration of a dynamic load balancing algorithm into a heterogeneous cloud computing environment is where it works most effectively, there are obstacles involved in its implementation in terms of replication. In addition, the quality of mechanical assembly plays a significant part in influencing the accuracy of static and dynamic calculations. This is because of the interplay between the two types of calculations. Therefore, improved performance may be achieved by the use of dynamic load adjusting systems under either spread or progressive situations. Utilizing workflows to draw attention to the interdependencies between different jobs is one way that the development of a distributed computing system might be improved further.

REFERENCES

1) Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "VMware distributed resource management: Design, implementation and lessons learned," VMware Tech. J., vol. 1,no. 1, pp. 45–64, Mar. 2012

2) Khiyaita, M. Zbakh, H. El Bakkali, and D. El Kettani, "Load balancing cloud computing: state of art," in Network Security and Systems (JNS2), 2012 National Days of, pp. 106–109, IEEE, 2012.

3) Dupont, G. Giuliani, F. Hermenier, T. Schulze, and A. Somov, "An energy aware framework for virtual machine placement in cloud federated data centres," in Proc. 3th IEEE Int. Conf. Future Energy Syst.: Where Energy, Comput. Commun. Meet (e-Energy), 2012, pp. 1–10.

4) Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid, 2009, pp. 124–131.

5) G. Nan, Z. Mao, M. Li, Y. Zhang, S. Gjessing, H. Wang, and M. Guizani, "Distributed resource allocation in cloud-based wireless multimedia social networks," IEEE Netw. Mag., vol. 28, no. 4, pp. 74–80, Jul. 2014.

6) G. Xu, Y. Ding, J. Zhao, L. Hu, and X. Fu, "A novel artificial bee colony approach of live virtual machine migration policy using Bayes theorem," Sci. World J., vol. 2013, no. 2013, p. 369209, Sep. 2013.

7) J. M. Bahi, S. Contassot-Vivier, and R. Couturier, "Dynamic load balancing and efficient load estimators for asynchronous iterative algorithms," IEEE Trans. Parallel Distrib. Syst., vol. 16, no. 4, pp. 289–299, Apr. 2005.

8) J. Zhao, L. Hu, Y. Ding, G. Xu, and M. Hu, "A heuristic placement selection of live virtual machine migration for energy-saving in cloud computing environment," PloS One, vol. 9, no. 9, p. e108275, Sep. 2014.

9) J. Zhao, Y. Ding, G. Xu, L. Hu, Y. Dong, and X. Fu, "A location selection policy of live virtual machine migration for power saving and load balancing," Sci. World J., vol. 2013, no. 2013, p. 492615, Sep. 2013.

10) K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in Proc. IEEE Int. Perform. Comput. Commun. Conf., 2014, pp. 1–8.

11) L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, "GreenCloud: A new architecture for green data center," in Proc. 6th Int. Conf. Ind. Session Autonomic Comput. Commun. Ind. Session, Jun. 2009, pp. 29–38.

12) M. H. Willebeek-LeMair and A. P. Reeves, "Strategies for dynamic load balancing on highly parallel computers," IEEE Trans. Parallel Distrib., vol. 4, no. 9, pp. 979–993, Sep. 1993.

13) N. J. Kansal and I. Chana, "Existing load balancing techniques in cloud computing: A systematic review.," Journal of Information Systems & Communication, vol. 3, no. 1, 2012.

14) R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw.: Practice Experience, vol. 41, no. 1, pp. 23–50, 2011.

15) S. M. Lau, Q. Lu, and K. S. Leung, "Adaptive load distribution algorithms for heterogeneous distributed systems with multiple task classes," J. Parallel Distrib. Comput., vol. 66, no. 2, pp. 163–180, 2006.