Dynamic Multi-Service Load Balancing System in Cloud-Based Multimedia

Vinza V. Suthan, Chitharanjan K.

Abstract— Load balancing is a process to distributing the workload across many computers or instruction data centres to maximize throughput and minimize work load on resources. In the case of cloud computing environments there were various challenges are there in the load balancing techniques like data security, and proper distribution etc. This is an efficient dynamic load balancing algorithm for cloud workload management by which the load can be distributed not only in a balancing approach, but also it allocate the load systematically and uniformly by checking certain parameters like number of requests the server is handling currently. It balances the load on the overloaded node to under loaded node so that response time from the server will decrease and performance of the system is increased. Here to considering a centralized hierarchical cloud-based multimedia system(CMS) consisting of a resource manager, cluster heads, and server clusters, in which the resource manager assigns clients' requests for multimedia service tasks to server clusters according to the job features, and then each cluster head gives the assigning job to the servers within its server cluster. For this complicated CMS, however, it is a challenging to design an effective load balancing algorithm which spreads the multimedia service job load on servers with the minimal cost for transmitting multimedia data between server clusters and clients, while not violating the maximal load limit of each server cluster. New genetic algorithm can be minimizing the response time and minimizing the communication cost. Simulation results explained that the proposed new genetic algorithm can efficiently cope with dynamic multiservice load balancing.

Index Terms— Cloud computing, Genetic algorithm, Dynamic load balancing.

I. INTRODUCTION

With advance of technology, cloud-based multimedia system (CMS) appears because of a larger number of users' demands for different multimedia computing and storage services through the Internet at the same time. It generally not corporate infrastructure, platforms, and software to support a large number of clients simultaneously to store and process their multimedia application data in a gives manner and meet different multimedia QoS requirements through the Internet. Most multimedia applications (e.g., audio/video streaming services, etc.) require considerable computation, and are often performed on mobile devices with constrained power, so that support of cloud computing is strongly required. In general, cloud jobs providers offer the utilities based on cloud facilities to clients, so that clients do not need to take much cost to request multimedia jobs and process multimedia data as well as their calculation results. By doing so, multimedia applications are processed on powerful cloud servers, and the clients only need to pay for the utilized resources by the time.

Revised Version Manuscript Received on August 24, 2015.

Vinza V Suthan, M.Tech Student, Department of Computer Science and Engineering, SCTCE, Pappanmcode, Trivandrum, India.

Chitharanjan K, Assistant Professor, Department of Computer Science and Engineering, SCTCE, Pappanmcode, Trivandrum, India.

This paper considering a centralized hierarchical CMS composed of a resource manager and a number of server clusters, each of which is coordinated by a cluster head, and we assume the servers in different server clusters to provide different services. Such a CMS is operated as follows. Every time when the CMS receives clients' requests for multimedia service tasks, the resource manager of the CMS assigns those task requests to different server clusters according to the characteristics of the requested tasks. Subsequently, the cluster head of each server cluster distributes the assigned task to some server within the server cluster. It is not hard to observe that the load of each server cluster significantly affects the performance of the whole CMS. In general, the resource manager of the CMS ais in pursuit of fairly distributing the task load across server clusters, and hence, it is of importance and interest to be able to cope with load balancing in the CMS.

II. RELATED WORKS

Submit Load balancing for wireless networks has been studied broadly in the earlier literature, e.g., Effective load balancing for cloud-based multimedia system, Optimal resource allocation for multimedia cloud based on queuing model, Among them, the load balancing problem for CMSs in effective load balancing in cloud-based multimedia system is disturbed with distribution the multimedia service task load on servers with the minimal cost for transmitting multimedia data between server clusters and clients, while the maximal load limit of each server cluster is not despoiled. An easy concern in their setting is to suppose that all the multimedia service tasks are of the same type. In practice, however, the CMS offer services of generating, editing, processing, and searching a diversity of multimedia data, e.g., hypertext, images, video, audio, graphics, and so on [1]. Different multimedia services have various requirements for the functions provided by the CMS (storage, central processing unit, and graphics processing unit clusters), e.g., the QoS requirement of hypertext webpage services is looser than that of video streaming services. In addition, the settings in the previous works [2], [3] did not consider that load balancing should adapt to the time change.

To respond to the practical requirements mentioned above, to assume that in the CMS, each server cluster can only switch a specific type of multimedia service tasks, and each client requests a different type of multimedia services at different time. At each specific time step, such a problem can be modelled as an integer linear programming formulation, which is computationally intractable in general [4]. Usually, intractable problems are usually solved by met heuristic approaches, e.g., simulated annealing [5], genetic algorithm [6], particle swarm optimization [7], [8], etc. In CMSs, to propose a genetic algorithm (GA) for the concerned dynamic

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

load balancing problem for CMSs. In the setting of GA, elite immigrants and random immigrants are added to new population; because they are suitable for solving the problems in dynamic environments. The experimental results show that to a certain extent, CMSs approach is capable of dynamically spreading the multimedia task load evenly. some previous works on other issues of cloud computing or distributed computing have also existed, e.g., cost-optimal scheduling on clouds [9], load balancing for distributed multi-agent computing [10], communication-aware load balancing for parallel applications on clusters [11], among others. Also note that GA has been applied to dynamic load balancing in [12], but their GA was designed for distributed systems, not specific to the CMS. In addition, they did have any multi-service concern.

III. MATH

The basic idea of the GA is to imitate the evolutional behaviour of a population of chromosomes (each of which represents a candidate solution) to find the solution close to the global optimal solution by using three basic evolutional operations: selection, crossover, and mutation. The key principle of the GA is that the fittest chromosomes can survive to the next generation and in general the fittest chromosome in the final generation represents the final solution that has a good performance. We first define how a chromosome represents a candidate solution, and how the fitness of each chromosome translates the objective value of the corresponding solution. The initial step of the algorithm is to maintain a population (Generation) of chromosomes that are initialized randomly or in some way. Next, a number of the chromosomes in the population are selected as the parental pool; in which each pair of chromosomes are crossovers to produce child chromosomes. Next, parts of the original chromosomes and the child chromosomes constitute the next generation. After the mutation part as simple GA after evolving a maximal number of generations or achieving a convergent condition, the solution represented by the chromosome with the best fitness value in the last generation is outputted as the final solution.

In new genetic algorithm the modification done in the mutation part. In this stage first we collect the output from the cross over part. Then load the suitable machine and assign the job. Then initialize the job. Check the job status. In the system to used two cases .First case which machine done the job completely, then check the type and then assign the job in to that machine. In another case there is found free nodes are found in the network assign the job in that machine and the execution to be done.

Our dynamic load balancing algorithm based on the GA is given in Algorithm 1, which is explained as follows. At each time step t, Algorithm 1 iterates on t to reallocate the network load assignments to adapt the time change. At first, Line 2 constructs a complete weighted bipartite graph G_t. Subsequently, we remove infeasible cases and then apply the GA to finding a locally optimal load assignment solution. Note that feasible solutions are restricted to some constraints. First constraint to guarantees that each client only allows at most one link to be assigned. For each client j in V, the constraint enforces that of at x_{ij}^t most one server cluster *i* is 1.Second constraint enforces that the utilized capacity of each server cluster cannot exceed its capacity at the *t*-th time step. Third constraint enforces that the multimedia service type requested by each client *j* is consistent with that provided by server cluster *i*. Fourth constraint enforces that each client *j* requests the multimedia server of the QoS no more than that offered by server cluster *i*. In Algorithm 1, Line 3 removes the links in Gt violating Constraints third and fourth, while the other two constraints will be considered in the GA. Before using the GA to calculate solutions, the information of $\{l_{ij}^t\}$ and $\{w_{ij}^t\}$ is required, so Line 4 of Algorithm 1 calls Algorithm 2 to obtain those information. After that, Line 5 of Algorithm 1 calls the algorithm detailed in Algorithm 3 to compute our final load assignment solution.

Algorithm 1: Dynamic Load Balancing Algorithm.

1. For *t*=1, 2, 3 ...do.
2. Consider complete weighted bipartite graph G_t.
3. remove the links in G_t violating Constraints $x_{ij}^t \ \emptyset_i = x_{ij}^t \ \varphi_j^t \ , \forall_i \in$ U, j $\in V$ and $x_{ij}^t \ q_i \geq x_{ij}^t r_{ij}^t \ , \forall_i \in U, j \in V$
4. Calculate $\{l_{ij}^t\}$ and $\{w_{ij}^t\}$ by calling Algorithm 2.
5. Assign $\{x_{ij}^t\}$ by calling Algorithm 3.

In algorithm 2 computes $\{l_{ij}^t\}, \{w_{ij}^t\}$.At first, Line 1 considers each client j $\in V$ *to compute* its weight $\{w_{ij}^t\}$ with each server cluster i. Remind that we apply the distributed binning scheme to calculating the proximity. Hence line 2 and 3 calculate landmark order l_j of client j,and then ,for each available server cluster i in the set U_j that include the server clusters connected to client j, Lines 6–7 calculate the landmark distance l_j of server cluster i. In Lines 8–14, if $l_j = l_i$(i.e., client j and server cluster i belong to the same landmark bin), we actually measure the network proximity between them, and then compute their l_{ij}^t and w_{ij}^t values; otherwise, we directly let the $l_{ij}^t$and w_{ij}^tvalues be ∞.

Algorithm **2: Calculate Weights**.

1. For each client j $\in$V do
2. measure the latency from client j to each landmark
3. compute the landmark order $l_j$of client j
4. obtain the set of available server clusters U_j
5. for each i $\in U_j$
6. measure the latency from server cluster i to each landmark

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

7. Compute the landmark order l_j of server cluster i
8. If $l_i = l_j$ then
9. Measure the network proximity d_{ij}^t between server cluster i and client j
10. measure server utilization ratios U_{ikj}^t for all k $\in K_i$
11. calculate l_{ij}^t and w_{ij}^t by using equations

$$w_{ij}^t = \begin{cases} \infty, & if\ d_{ij}^t \rightarrow \infty\ or\ \emptyset_i \neq \varphi_j; \\ d_{ij}^t\, l_{ij}^t, & otherwise; \end{cases}$$

and $l_{ij}^t = \sum_{k \in K_i} U_{ikj}^t\, C_{ik}$ respectively.

12. Else
13. $w_{ij}^t = l_{ij}^t = \infty$
14. End if
15. End for
16. End for

In algorithm 2 calculated the values w_{ij}^t and l_{ij}^t and to ready to apply algorithm 3 to calculating the load assignment. This is explained as follows.

Algorithm 3. NEW GENETIC ALGORITHM (GRAPH G_t, TIME STEP t)

In new genetic algorithm the difference from the simple genetic algorithm the mutation part is different. The simple genetic algorithm we are considering the communication cost and waiting time. In new genetic algorithm considering communication cost, waiting time and cost. In the mutation part this method is applying we are considering some nodes and jobs are assigned in the nodes, after the cross over the mutation the following is happened.

J0	J1	J2	J3	J4	J5	J6	J7	J8	Jn
M1	M2	M2	M1	M2	M3	M2	M3	M4	Mn

1. Machine ⟶ load
2. Load = suitable of machine
3. Initialize 0
4. For ij=1 to N
5. If ji allocate M_k
6. Load[k]++
7. Select node = free node
8. While(true)
9. {
10. Free node suitable node other node in selected node
11. For suitable node for replace
12. Balance load ⟶ Machine
13. If suitable node is exit
14. Replace Balance load
15. initialize Machine
16. Break
17. Else
18. Add Balance load to selected node

In new GA the difference between the simple GA the change happening the after the cross over. In new GA the first we are assigning the jobs in the node and execute. Then to check the job status .which one is execute first then assigned the balance job into the other that machine. Then check the job status. Another condition is if we have free node the balance job is assigned the free node then execute. Check the job status. The new GA supported this type of load balancing. The simple GA considering the communication cost and waiting time. The new GA considering the communication cost, execution cost and waiting time.

IV. PERFORMANCE ANALYSIS

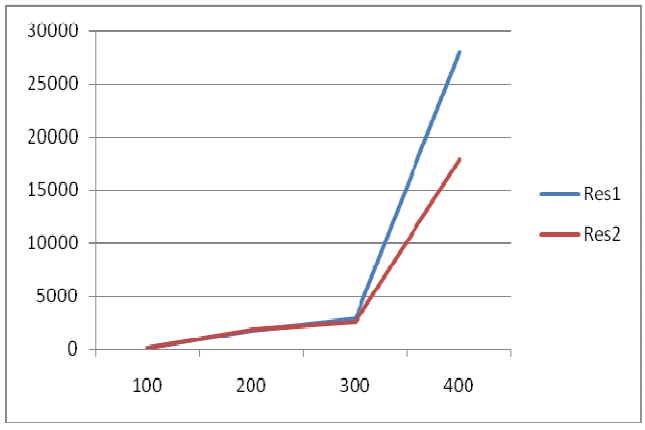
Use Consider an instance with 20 server clusters (m = 20) and 100 jobs (n = 100). Then construct the bipartite graph with check two constraints they are one is that the multimedia services type requested by each client j is consist with that provided by cluster i and enforces that each client j requests the multimedia server of the QoS no more than that offered by server cluster i.

Communication cost=Data size/Bandwidh

Response time =latency +cpu time, then calculate the average response time.

Job size	response time 1	response time 2
100	95.2487968864632	90218368766373
200	1733.13567932336	1795.46236595623
300	2830.81693828969	2520.832214124526
400	27975.4630781629	17865.59042136
500	41.489246871289	0

In the performance analysis to are considering the job size and the response time. The first we are calculating each generation's communication cost and the waiting time, execution time. Here latency means that the waiting time of each node taking the processing after the completion.



In the simulation the cloudsim is calculated the response time and the CPU time and the ideal time, cc, Exe time, waiting time.., etc."

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

V. CONCLUSION

Because A GA approach for the CMS-dynRMLB problem has been proposed and implemented. The main difference of our model from previous models is that we consider a practical multi-service dynamic scenario in which at different time step, clients can change their locations, and each server cluster only handles a specific type of multimedia tasks. The main contribution in this paper is to propose a mathematical formulation of the CMS-dynMLB problem, so that interested readers are able to investigate the problem precisely in the future. The performance of our GA approach is evaluated by detailed simulation. As a future work, we extend the behavioural characterization of proximity malware to account for strategic malware detection evasion with game theory is a challenging task..

REFERENCES

1. W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing: An emerging technology for providing multimedia services and applications," IEEE Signal Processing Magazine, vol. 28, no. 3, pp. 59–69,2011.
2. W. Hui, H. Zhao, C. Lin, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology. IEEE Press, 2011, pp. 165–168.
3. X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model," in Proceedings of 2011 IEEE 13th International Workshop on Multimedia Signal Processing (MMSP 2011).IEEE Press, 2011, pp. 1–6.
4. M. Garey and D. Johnson, Computers and Intractability - A Guide to the Theory of NP-Completeness. Freeman, San Francisco, 1979.
5. S. Kirkpatrik, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–680, 1983.
6. J. H. Holland, Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.
7. J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks. IEEE Press, 1995, p. 1942V1948.
8. Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Proceedings of IEEE International Conference on Evolutionary Computation.IEEE Press, 1998, pp. 69–73.
9. R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing. IEEE Press, 2010, pp. 228–235.
10. K.-P. Chow and Y.-K. Kwok, "On load balancing for distributed multiagent computing," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 8, pp. 787–801, 2002.
11. X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, "Communication aware load balancing for parallel applications on clusters," IEEE Transactions on Computers, vol. 59, no. 1, pp. 42–52, 2010.
12. Y. Zomaya and Y.-H. Teh, "Observations on using genetic algorithms for dynamic load-balancing," IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 9, pp. 899–911, 2001.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.