ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 3 (July - Sept. 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Genetic Algorithm with Range Selection Mechanism for Dynamic Multiservice Load Balancing in Cloud-Based Multimedia System

[I]Michael Sadgun Rao Kona, [II]K.Purushottama Rao
[I,II]Dept. of Information Technology, LBRCE, JNTUK University, Andhra Pradesh, India

Abstract
Consider a centralized hierarchical cloud-based multimedia system (CMS) which has a resource manager, cluster heads, and server clusters. Resource manager accepts clients' requests for multimedia service tasks and allocates server clusters based on the task characteristics. Cluster head distributes the task to the servers within its server cluster. An effective genetic algorithm with an immigrant scheme is proposed for load balancing that spreads the multimedia service task load on servers with the minimal cost for transmitting multimedia data between server clusters and clients, by not violating the maximal load limit of each server cluster. Dynamic multiservice scenario is considered in which each server cluster only handles a specific type of multimedia operation, and every client requests a different type of multimedia service at a different time.

Keywords
Cloud computing, load balancing, genetic algorithm and multimedia system.

I. Introduction
Cloud-based multimedia system (CMS) [3] gained momentum because there is huge number of users' demands for various multimedia computing and storage services through the Internet at the same time [4], [5].
It provides infrastructure, platforms, and software as service to large number of clients at the same time in order to store and process their multimedia application data in a distributed manner by meeting different multimedia QoS requirements through the Internet.
Cloud computing is strongly required in most multimedia applications with huge computation and for mobile devices which are power constrained.
In general, a cloud service provider offers required facilities to clients which take less cost to request, process and to compute multimedia services. So, multimedia applications are processed on powerful cloud servers with less cost because client should only pay for the utilized resources by the time.
We are considering a centralized hierarchical CMS from [3] 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.

Operation of CMS
Resource manager assigns client requested multimedia service task to different server clusters according to the characteristics of the requested tasks. Cluster head will distribute the assigned task to some server within the server cluster.
The load of each server cluster affects the performance of the whole CMS. Resource manager of the CMS is required to distribute the task load across server clusters, and hence, it is important to be able to cope with load balancing in the CMS.
Services offered by CMS are generating, editing, processing, and searching a variety of multimedia data, e.g., hypertext, images, video, audio, graphics, and so on with various requirements for the functions provided by the CMS (storage, central processing unit, and graphics processing unit clusters), e.g., the requirement for QoS of html webpage services is looser than that of video streaming services.

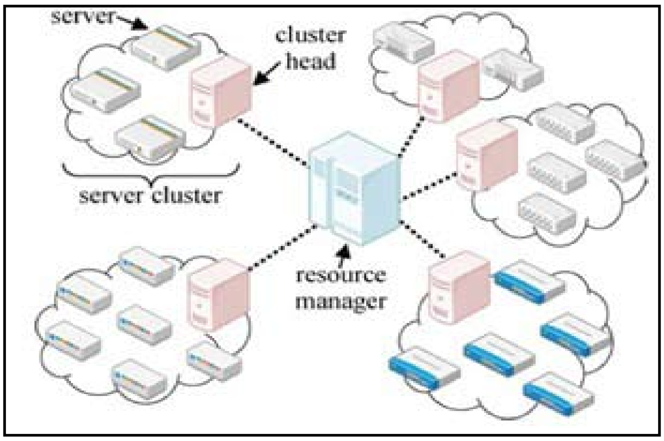

Fig. 1: Centralized Hierarchical cloud based Multimedia System [1]

We assume in CMS, each server cluster can only handle a specific type of multimedia service task, and each client requests a different type of multimedia service at different time. In this paper, we propose a genetic algorithm (GA) for the concerned dynamic load balancing problem for CMSs. In our setting of GA, elite immigrants and random immigrants are added to new population, because they are suitable for solving the problems in dynamic environments [6]. The experimental results show that to a certain extent, our approach is capable of dynamically spreading the multimedia task load evenly.

II. Problem Description
Actually this load balancing problem for the CMS is from [1], and we are extending their model with different selection method in GA.

A. System Overview
CMSs can be classified into two categories: centralized and decentralized. We are considering a centralized CMS as illustrated in Fig. 1, which consists of a resource manager and a number of server clusters each of which is coordinated by a cluster head. Different from the decentralized CMS, whenever it receives requests from client for multimedia service tasks, the resource

Vol. 2, Issue 3 (July - Sept. 2014)
ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

manager of the centralized CMS stores the global service task load information collected from server clusters, and decides the amount of client's requests assigned to each server cluster so that the load of each server cluster is distributed as balanced as possible in terms of the cost of transmitting multimedia data between server clusters and clients. The decision of assignment is based upon the characteristics of different service requests and the information collected from server clusters.

B. Problem Formulation

We are using the same CMS problem formulation as in [1], in which time is divided into different time steps. At the t^{th} time step, the CMS is modeled as a complete weighted bipartite graph $G_t = (U, V, E, \varphi, \psi^t, q, r^t, w^t)$ in which U is the set of vertices for server clusters, V is the set of vertices for clients, E is the set of edges between U and V , in which each edge $e_{ij} \in E$ represents the link between server cluster $i \in U$ and client $j \in V$. φ is a function used to represent that server cluster i can only cope with multimedia tasks of type φ_i, ψ^t is a function used to represent that client j requests the multimedia service of type ψ_j at the t^{th} time step. q is a function used to represent that server cluster i can provide the multimedia service of QoS q_i, r^t is a function used to represent that client j requests the multimedia service of QoS requirement r^t_j at the t^{th} time step and finally wt is the weight function associated with edges, in which w^t_{ij} denotes the wt value that represents the cost for transmitting multimedia data between server cluster i and client j at the t^{th} time step, which is defined as follows:

$w^t_{ij} = \infty$, if $d^t_{ij} \to \infty$ or $\varphi_i \neq \psi_j$

$w^t_{ij} = d^t_{ij} \cdot l^t_{ij}$, otherwise (1)

where

d_{ij} is the network proximity between server cluster i and client j;

l_{ij} is the traffic load of the link between server cluster i and client j that is defined as follows:

$l^t_{ij} = \sum_{k \in K} u^t_{ikj} C_{ik}$ (2)

where K_i is the servers' set in server cluster i; u^t_{ikj} is the server utilization ratio of server k in server cluster i due to client j, and C_{ik} is its capacity.

Note that the proximity d_{ij} between server cluster i and client j in (1) is required to be measured at every time step due to dynamic change of network topology. Proximity between nodes is calculated with landmark order and Binning scheme.

This paper uses the same linear programming formulation used in [1] for every t^{th} time step with constraints:

Each client only allows at most one link to be assigned (Constraint (4)), the utilized capacity of each server cluster cannot exceed its capacity at the t^{th} time step (Constraint (5)), multimedia service type requested by each client j is consistent with that provided by server cluster i (Constraint (6)) and each client j requests the multimedia server of the QoS no more than that offered by server cluster i (Constraint (7)).

Dynamic Multiservice Load Balancing In Cms (Cms-Dynmlb): $G_t = (U, V, E, \varphi, \psi^t, q, r^t, w^t)$ is the bipartite graph for CMS with m servers and n clients for each time step t=1,2,… .This paper uses Range selection method for genetic algorithm (GA) with immigrant scheme [1] for better results in solving the problem.

Algorithm 1 Dynamic Load Balancing Algorithm
for t = 1, 2, · · · do
  create complete weighted bipartite graph G_t
  remove the edges in G_t violating (6) and (7)
  calculate $\{l_{ij}\}$ load and weight $\{w^t_{ij}\}$ on server cluster i by client j by calling Algorithm2
  assign $\{x^t_{ij}\}$ by calling Algorithm 3
end for

Algorithm 2 Calculate Weights
for each client $j \in V$ do
  measure the latency from client j to each landmark
  compute the landmark order j of client j
  obtain the set of available server clusters U_j
  for each $i \in U_j$ do
    measure the latency from server cluster i to each landmark
    compute the landmark order i of server cluster i
    if $l_i = l_j$ then
measure the network proximity d^t_{ij} between server cluster i and client j
measure server utilization ratios u^t_{ikj} for all k □ K_i
      calculate l^t_{ij} and w^t_{ij} by Equations (2) and (1), respectively
    else
      $w^t_{ij} = l^t_{ij} = \infty$
    end if
  end for
end for

Algorithm 3 Genetic Algorithm (graph Gt , time step t)
if t = 1 then
  generate and evaluate the initial population P_0 of size η in which each chromosome has to satisfy (4) and (5).
else
  $P^t_0 \leftarrow P^{t-1}_\tau$
end if
$i \leftarrow 0$
while $i < \tau$ or the convergent condition is not achieved
do
  select the parental pool Q^t_i from P_i^t by using Range Selection mechanism
  reproduce a new population P_i^t of size η by performing
  crossover procedure on pairs of chromosomes in Q^t_i with probability p_c
  perform mutation procedure on chromosome in P_i^t with probability p_m
  repair each infeasible chromosome in P_i^t
  evaluate P_i^t
  if $r_e > 0$ then
    a number of the best chromosome E_{i-1} , called elite, are selected from the previous generation P_{i-1}
    generate and evaluate $r_e \cdot \eta$ elite immigrants
  end if

```
   if r_r > 0 then
      generate and evaluate r_r · η random immigrants
   end if
   replace the worst chromosomes in Pit with the above
   elite and random immigrants
P^t_{i+1} ← P_i^t
i←i+1
end while
output the best found chromosome as the solution at the
t^th time step
```

Algorithm 4 Initialize
for $j = 1$ to n do
 $\arg(U_j)$ ← set of integers which store the indices of the available cluster servers U_j that are linked with client j in the reduced bipartite graph G_t (where the links violating Constraints (6) and (7) have been removed in Algorithm 1)
end for
for $p = 1$ to η do
 construct the p^{th} chromosome $c_p = \sigma_1, \cdots, \sigma_j, \cdots, \sigma_n$, in which σ_j is a number chosen arbitrarily from $\arg(U_j)$ for each $j \in \{1, \cdots, n\}$
let o be the set of the positions of 0's in c_p that do not violate our problem constraints
 for $i = 1$ to m do
  scan chromosome cp to compute the j□U $x^t_{ij}\ l^t_{ij}$ value for server cluster i
  $g_i \leftarrow \sum j \in U\ x^t_{ij}\ l^t_{ij} - \sum_{k□Ki} C_{ik}$
  let o_i be the set of o associated with cluster server i
  while $g_i > 0$ do
   find the index x such that $\sigma_x = i$ in cp , $l_{ix} > g_i$,and the gap between l_{ix} and g_i is the smallest if there is a position o_y in o that can accommodate l_{ix} and satisfy all the problem constraints then
    swap the values of o_y and σ_x
    remove o_y in o
    $g_i \leftarrow g_i - l_{ix}$
   else
    $\sigma_x \leftarrow 0$
   end if
  end while
 end for
end for

III. Our Genetic Algorithm

A. Our Algorithm

GA is based on imitating the evolutional behavior of a population of chromosomes (each 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 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 crossovered to produce child chromosomes. Next, parts of the original chromosomes and the child chromosomes constitute the next generation. 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.

At each time step t, Algorithm 1 iterates on t to reallocate the network load assignments so as to adapt to the time change. First, Line 2 constructs a complete weighted bipartite graph G_t. Line 3 removes the links in G_t violating (6) and (7). Before using our GA to calculate solutions, the information of $\{l^t_{ij}\}$ and $\{w^t_{ij}\}$ is required, so Line 4 of Algorithm 1 calls Algorithm 2 to obtain those information. After that, Line 5 of Algorithm 1 calls Algorithm 3 to compute our final load assignment solution $\{x^t_{ij}\}$.

In algorithm 2, Line 1 considers each client $j \in V$ to compute its weight w^t_{ij} with each server cluster i. We are using distributed binning scheme to calculating the proximity, so, Lines 2–3 calculate the landmark order lj of client j, and then, for each available server cluster i in the set Uj that includes the server clusters connected to client j, Lines 6 and 7 calculate the landmark distance li of server cluster i. In Lines 8–14, if lj = li (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^t_{ij} and w^t_{ij} values; otherwise, we directly let the l^t_{ij} and w^t_{ij} values be ∞.

With the values of $\{ l^t_{ij} \}$ and $\{ w^t_{ij} \}$, we are ready to apply the GA to computing the optimal load assignment in Algorithm 3. The GA runs differently based on two parameters: the input bipartite graph G_t and the input time step t. Recall that Gt may not be a complete graph any longer, because the links that violate some problem constraints have been removed in Line 3of Algorithm 1. Let η and τ denote the size of a population and the number of generations, respectively. In Lines 1–5, the initial population of η chromosomes is generated in two cases. If $t = 1$ (i.e., this is the first time to run the GA), then we randomly generate the initial population that satisfies the remaining two constraints that we did not consider yet (i.e., (4) and (5)). Otherwise (i.e., $t \neq 1$, which implies that there existed the final population P^{t-1}_τ of the $(t - 1)^{th}$ time step), Line 4 uses P^{t-1}_τ as the initial population at the t^{th} time step Subsequently, the while loop in Lines 7–23 repeats at most τ iterations, each of which produces a population P^t_{i+1} of chromosomes, i.e., the next population at the t^{th} time step. Line 8 selects a number of chromosomes from the population P^t_i as the parental pool Q^t_i. Lines 9 and 10 perform crossover and mutation operators to the population P^t_i with probabilities pc and pm, respectively. In addition, since our concerned problem considers dynamic scenarios at different time steps, we add elite immigrants and random immigrants to adapt to dynamic changes, as the immigrants are used to solve dynamic problems conventionally [6]. Lines 13–16 add elite immigrants for increasing efficiency of convergence, while Lines 17–19 add random immigrants for increasing the population diversity. Line 20 replaces the worst chromosomes in P^t_i with the elite and random immigrants. After finishing the while loop, the best found chromosome is outputted as the solution at the t^{th} time step. Note that a multimedia service task may not be able to be finished within a single time step, i.e., it takes a number of time steps to be finished. Hence, it is not necessary to maintain a specific link for the same service for some time steps because the network is packet-based.

B. Basic Elements of Our GA

To use GA in order to solve the CMS-dynMLB problem, we first define the basic elements of GA (i.e., population, chromosome, fitness function) for the problem as follows.

1) Population: A population consists of a number of chromosomes, and the number of chromosomes depends on the given initial population size.

2) Chromosome: A solution for the CMS-dynMLB problem consists of all the indicator variables $\{ x^t_{ij} | \forall i \in U, j \in V\}$.The solution for indicator variables $\{ x^t_{ij} \}$ is encoded as a sequence of decimal numbers of length n: $\{\sigma_1, \cdots, \sigma_j, \cdots, \sigma_n\}$ where $\sigma_j \in \{0, 1, 2, \cdots, m\}$ represents the link assignment of client j with the following two cases.

1) If $\sigma_j = 0$, then each $x^t_{ij} = 0$ for any $i \in U$, i.e., client j is not linked at the t^{th} time step.

2) Otherwise, $\sigma_j = k \neq 0$, meaning that $x^t_{kj} = 1$ and $x^t_{ij} = 0$ for any $i \neq k$, i.e., client j is linked to server cluster k uniquely.

3) Fitness Function: Fitness function is the measure for determining which chromosomes are better or worse. We let the objective (3) of our concerned problem as our fitness function as follows: $f(X(t)) = \lambda \cdot \sum_{i \in U} \sum_{j \in V} x^t_{ij} w^t_{ij} / \sum_{j \in V} w_{max} + (1 - \lambda) \cdot (1 - \sum_{j \in V} \sum_{i \in U} x^t_{ij} / |V|)$ where X(t) is the profile of $\{ x^t_{ij} \}$.

C. Main Components of Our GA

1) Initialization: In Lines 1–5 of Algorithm 3, we use two cases to get initial population at each time step. If t >= 1, then just take the final population at the previous time step as the initial population at the current time step. For the other case (i.e., t = 1), we randomly produce the initial population by using Algorithm 4, which is explained as follows. Lines 1–3 find the set $\arg(U_j)$ that collects the indices of the available server clusters for each client j. Next, the set $\arg(U_j)$ is used to construct a population of η chromosomes in Line 5, and then we repair each chromosome to be feasible in Lines 6–21. The idea of the repairing operation is that if (5) is violated, then we find another available server cluster to serve the violated load; otherwise, we drop the load.

2) Selection, Crossover: The range selection procedure is used. Each individual in the population is assigned a numerical rank based on their fitness, and selection is based on these ranking rather than absolute differences in fitness. The advantage of this method is that it can prevent very fit individuals to gain dominance at first at the expense of the less fit, which would reduce the genetic diversity of the population and could hamper the search for an acceptable solution [7]. One-point crossover operation is used in which two selected chromosomes are crossovered for generating two new child chromosomes in order to keep some characteristics of its parent chromosomes.

3) Mutation: Changing some genes on some chromosomes. let pm be given probability to mutate. In the mutated chromosome, a nonzero gene σ_x is chosen randomly, and then we randomly find a zero gene σ_y that can accommodate σ_x and do not violate any problem constraint. Then, we swap the values of σ_x and σ_y.

4) Repair: The modified chromosomes may become infeasible, and hence repair is required. The repairing operation is referred to Lines 6–21 in Algorithm 4, which have been used in initializing chromosomes.

5) Termination: If the difference of average fitness values between successive generations in the latest ten generations is not greater than 1% of the average of the average fitness values of these ten generations, or the maximum generations are achieved, then our GA stops. After termination, the best chromosome from the latest population is chosen, and its corresponding load assignment is outputted as the final solution.

IV. Implementation and Experimental Results

A. Implementation

CloudSim 3.0.3[2] is used for simulation with java as programming language. CloudSim has support for modeling and simulation of large scale Cloud computing environments, including data centers, on a single physical computing node. It has necessary classes in java for simulating every entity that involved in our project like Datacenter, Host, VM, Cloudlet, DatacenterBroker etc. The CloudSim toolkit supports both system and behaviour modelling of Cloud system components. It has availability of a virtualization engine that aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node.

Our simulation was tested on an Intel Core i3-2328M CPU at 2.20 GHz with 4-GB memory.

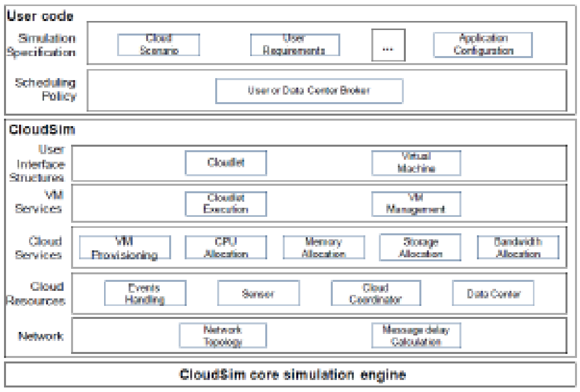

Figure 3. Layered CloudSim architecture.

B. Results

With Range selection mechanism implementation in existing genetic algorithm for balancing dynamic multimedia load in cloud based CMS there is a little betterment.

V. Conclusion

A genetic algorithm approach for optimizing the CMSdynMLB was proposed and implemented. The main difference in our model from previous models is that we considered a Range selection mechanism in genetic algorithm along with practical multiservice dynamic scenario in which at different time steps, clients can change their locations.

References

[1] *Chun-Cheng Lin, Member, IEEE, Hui-Hsin Chin, Student Member, IEEE, and Der-Jiunn Deng, Member, IEEE, "Dynamic Multiservice Load Balancing in Cloud-Based Multimedia System" in Systems Journal, IEEE (Volume:8 , Issue: 1) ,pp.225-234, March 2014.*

[2] *Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.*

[3] *W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing: An emerging technology for providing multimedia services and applications," IEEE Signal Process. Mag., vol. 28, no. 3, pp. 59–69, May 2011.*

[4] *C.-F. Lai, Y.-M. Huang, and H.-C. Chao, "DLNA-based multimedia sharing system over OSGI framework with extension to P2P network," IEEE Syst. J., vol. 4, no. 2, pp. 262–270, Jun. 2010.*

[5] W. Hui, H. Zhao, C. Lin, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in Proc. Int. Conf. Electron. Mech. Eng. Inform. Technol., 2011, pp. 165–168.
[6] H. Cheng and S. Yang, "Genetic algorithms with immigrant schemes for dynamic multicast problems in mobile ad hoc networks," Eng. ppl.Artif. Intell., vol. 23, no. 5, pp. 806–819, 2010.
[7] R.Sivaraj et al., Dr.T.Ravichandran, "A Review of selection methods in genetic algorithm" in IJEST, Vol. 3 No. 5, pp.3792-3797, May 2011.

Author's Profile

Michael Sadgun Rao Kona pursuing M.Tech Degree in Software Engineering from Lakireddy Balireddy College of Engineering, Mylavaram, AP, INDIA. His main research interest in software engineering, Cloud Computing.

K.Purushottam Rao is presently working as Assistant Professor in Dept. of Information Technology, Lakireddy Balireddy College of Engineering, Mylavaram.