British Journal of Applied Science & Technology
7(2): 179-185, 2015, Article no.BJAST.2015.135
ISSN: 2231-0843

SCIENCEDOMAIN *international*
www.sciencedomain.org

SCIENCEDOMAIN

A Genetic Algorithm in Green Cloud Computing

Nada Al Sallami[1*] and Sarmad Al Aloussi[2]

[1]*Department of Multimedia, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan.*
[2]*Computer Information Systems Department, University of Financial and Banking Sciences, Amman, Jordan.*

Authors' contributions

This work was carried out in collaboration between both authors. Author Nada Al Sallami designed the study, performed the statistical analysis, wrote the protocol, and wrote the first draft of the manuscript and managed literature searches. Author Sarmad Al Aloussi evaluated the protocol, managed the analyse of the study and literature searches. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJAST/2015/15412
Editor(s):
(1) Wei Wu, Applied Mathematics Department, Dalian University of Technology, China.
Reviewers:
(1) Tuan Anh Nguyen, Department of Computer Engineering, Korea Aerospace University, South Korea.
(2) Tansel Dökeroğlu, Middle East Technical University, Turkey.
(3) Anonymous, Malaysia.
(4) A. Shajin Nargunam, Noorul Islam Centre for Higher Education, Noorul Islam University, India.
Complete Peer review History: http://www.sciencedomain.org/review-history.php?iid=771&id=5&aid=7955

Original Research Article

Received 24th November 2014
Accepted 15th January 2015
Published 30th January 2015

ABSTRACT

In this paper a genetic algorithm to solve the problem of load balancing in green cloud computing environment was proposed. For a specified total load on N clouds, the proposed algorithm finds an optimal load vector of length less than or equal to N. It helps in avoiding overheating by balancing the workload across all the clouds, hence reducing the amount of energy consumed. In addition, the proposed genetic algorithm may converge to a number of clouds less than or equal to N, so that a substantial reduction in energy consumption can be made by powering down servers when they are not in use.

Keywords: Green cloud computing; load balancing; genetic algorithm.

1. INTRODUCTION

The main appeal of cloud computing is that customers only use what they need, and only pay for what they actually use. Resources are available to be accessed from the cloud at any given time, and from any location via the internet. However, Internet Data Centers (IDCs) use a

*Corresponding author: E-mail: nada.alsalami@yahoo.com, nadaabbas@zuj.edu.jo;

significant and growing portion of energy, therefore, Energy-intensive IDCs are a major source of CO2 emission. Internet data centers (IDCs) have become an integral component to operate Internet services and scientific computation. Since they have been increasing in scale and complexity, they consume a growing and visible portion of energy supply [1]. In cloud computing, service requests have heterogeneous resource demands because some services may be CPU intensive whereas others are I/O-intensive. Cloud resources need to be allocated not only to satisfy Quality of Service (QoS) requirements specified by users via SLAs, but also to reduce energy usage and improve the profits of the service providers [2]. Cloud computing is highly dynamic, and hence, resource allocation problems have to be continuously addressed, as servers become available/non-available while at the same time the customer demand fluctuates. Before scheduling tasks on cloud computing, the characteristics of the cloud should be taken into account. Some of the characteristics of cloud include [3]: On-demand self-service, Ubiquitous network access, Location independent resource pooling, Rapid elasticity and Pay per use. Since cloud computing is generally characterized as an IT service (with the vendor providing and maintaining the software and hardware infrastructure), the ability of the client organization to integrate and utilize the vendor’s services determines the extent IT benefits are likely to be achieved. Organization- specific capabilities related to implementation, integration, and utilization of cloud services play a key role in deployment performance [4]. There are three IT-related capabilities-technical, managerial, and relational-characterized as a major potential source of competitive advantage [5].

Load balancing in clouds is a mechanism that distributes the excess dynamic local workload evenly across all the nodes. It is used to achieve a high user satisfaction and resource utilization ratio, making sure that no single node is overwhelmed, hence improving the overall performance of the system. Proper load balancing can help in utilizing the available resources optimally, thereby minimizing the resource consumption. It also helps in implementing failover, enabling scalability, avoiding bottlenecks and over-provisioning, reducing response time etc. Load balancing can be one such energy-saving solution in cloud computing environment. Thus load balancing is required to achieve Green computing in clouds which can be done with the help of the following two factors [4]:

1. *Reducing Energy Consumption* - Load balancing helps in avoiding overheating by balancing the workload across all the nodes of a cloud, hence reducing the amount of energy consumed.
2. *Reducing Carbon Emission* - Energy consumption and carbon emission go hand in hand. The more the energy consumed, higher is the carbon footprint. As the energy consumption is reduced with the help of Load balancing, so is the carbon emission helping in achieving Green computing.

The aim of this paper is to achieve green cloud computing by satisfy the above two factors. Although there are many works in this area (as described in section 2), however the key problems of the previous works is the applicability in real time. Our work uses an efficient fitness function to find an optimal distribution of the workload into some nodes in real time. The proposed algorithm avoids overheating by balancing the workload across all the nodes, hence reducing the amount of energy consumed. Obviously, a substantial reduction in energy consumption can be made by powering down servers when they are not in use. The proposed clouds minimization step is satisfied when the total load is low with respect to the current assigned clouds as described in section 3.1.

2. RELATED WORK

In recent years, a lot of attention has been paid to the artificial intelligence methods such as genetic algorithms by researchers because of its intelligence and inferred parallelism. Genetic algorithm has been extremely usage to solve the problem of cloud resources scheduling and has obtains perfect effects. Specifically, authors in [6] with the use of Genetic algorithm proposed a cloud scheduling approach for VM load balancing. Li et al. [7] proposed a job oriented based model for cloud resource scheduling. This model assigns jobs to the resources according to the rank of the job. Singh et al. [8], describe several job scheduling algorithms and compare between these algorithms. As it is mentioned in this paper, a good cloud job scheduling algorithm increases should schedule the resources to optimize the usage of the resources. Various scheduling algorithms are presented for resource scheduling but each one has its own restriction. In [9], Fang et al. proposed a model to deal with the job scheduling problems for a group of cloud user requests. Each datacenter has different services with various resources. This plan assumes resource provisioning as an important issue for job scheduling. The main goal of this model is reducing the average tardiness of connection requests. It presents four merged scheduling algorithms to schedule virtual machine on data centers. The mentioned model reduces the average tardiness of connection requests and the connection blocking percentage. In [10] Chen et al. proposed a Genetic algorithm based job scheduling. The fitness function is divided into three sub-fitness functions and then linear combination of these sub-fitness value is carry out for obtaining the fitness value. They use a strategy which is based on three load dimensions: CPU network throughput, disk I/O rate. To achieve a nearly optimum solution this plan applies the hybrid genetic algorithm merged with knapsack problem with multiple fitness. The author claims that the algorithm can obtain the goal of raising resources utilization efficiency and lower energy consumption. The algorithm reduces energy consumption and also the utilization of the resources. In [11], with the aid of genetic algorithm and fuzzy theory, the authors present a hybrid job scheduling approach, which considers the load balancing of the system and reduces total execution time and execution cost. They define two types of chromosomes with different QoS parameters; Then with the aid of fuzzy theory they obtain the fitness value of all chromosomes for the mentioned two types. Pushpendra et al. [12] developed a load balancing algorithm using Divisible load scheduling theorem to maximize or minimize different performance parameters (throughput, latency for example) for the clouds of different sizes (virtual topology depending on the application requirement). Sahu et al. [13] introduced a threshold based Dynamic compare and balance algorithm for cloud server optimization. It minimizes the number of host machines to be powered on, for reducing the cost of cloud services.

3. PROPOSED GENETIC ALGORITHM

Genetic Algorithm are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. It is a rapidly growing area of artificial intelligence. Genetic algorithms belong to lot of best characteristics that decides it’s a good option when someone needs to solve very complicated problems or NP hard problems. The simplicity and robustness of the algorithm has made it popular among developers. Genetic algorithm mimics the process of natural evolution based on a population of candidate solutions. In the process of evolution, a modification is performed by using genetic operators on each individual. Each chromosome represents a Load balancing result, and an evaluation fitness function is called to evaluate the offspring.

3.1 Individual Encoding

Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form so that a computer can process. Each individual is expressed as a vector of (k*) entries*, where (*k)* must be more than or equal 1 and less than or equal the current assigned clouds number (n). Each entry denotes a cloud or a cluster of clouds and stores a positive integer number more than or equal to zero, which represent the current load in that cloud or cluster of clouds. Fig. 1 shows an individual example with (k=8, n=8). The loads are distributed unevenly across the nodes. Fig. 1-b shows approximately balanced vector (obtained from the proposed algorithm). The total load is distributed near the average load. Load of size 16 is assigned to cloud number 1; load of size 23 is allocated to cloud number 2, and so on. The total load is distributed optimally across all nodes.

Cloud number	1	2	3	4	5	6	7	8	Total load
Load	0	55	0	74	0	10	1	10	150

a: individual with k=8 (unbalanced)

Cloud number	1	2	3	4	5	6	7	8	Total load
Load	16	23	17	19	19	20	19	18	150

b: individual with k=8 (balanced)

Fig. 1. Individuals encoding example

An important operation in the proposed algorithm is clouds minimization. Minimization is satisfied when the total load is low with respect to the current assigned clouds. Genetic Algorithm may converge on an optimal vector with less assigned clouds (i.e. k<n). A substantial reduction in energy consumption can be made by powering down servers when they are not in use to achieve green computing. Fig. 2 shows the clouds minimization, a vector of length equal to eight is minimized to five.

Cloud number	1	2	3	4	5	6	7	8	Total load
Load	3	5	2	19	1	8	1	1	40

a: Individual with k=8 (unbalanced)

Cloud number	1	2	3	4	5	Total load
Load	9	11	13	10	8	40

b: Individual with k=5 (balanced)

Fig. 2. Cloud minimization

3.2 Fitness Function

The fitness function is deduced from the energy consumption, the cloud's load and the average load ar_L (For Fig. 1 av_L ≈ 19). Each individual in the population of size M has fitness value assigned to it by using the following equation:

$$'\text{fitness (x)} = \sum_{i=1}^{i=k} \mathrm{ABS}(avL - L_i) \qquad (1)$$

Where: L_i is the current workload of cloud i in the current individual (x), k is the length of current individual (x), n/2 <= k <= n

Individual with small fitness value has high probability to transform to the next generation. To give rise to the fitness variation in the overall population from one generation to the next, the fitness of each individual is computed proportional to the fitness summation of all individuals in the population as follows:

$$\text{fitness(x)} = \frac{'\text{fitness(x)}}{\sum_{j=1}^{M} 'fitness(j)} \qquad (2)$$

Individual with smallest fitness value is regenerated to the next generation as it in the reproduction genetic operation.

3.3 Crossover Operator

The crossover operator uses two individuals s_1, s_2 to generate two new individuals s_1', s_2'. The parent and children individuals are typically of different size but with the same total load. For individual s_1, first randomly generates two integers *i*, *j*, where, 1<= i<= j<=z (z is the size of the smaller individual); then, copy the load in s_1 to s_2' *in* the same position. The same operation is repeated with s_2 to generate s_1' If the total load of any child is not equal the actual total load, then add or subtract the difference in load to a randomly selected point in this child, as shown in the following example (Fig. 3), where i =3, j = 5, z=5:

s_1	40	5	23	33	67	0	0	89	Total= 257

S_2	30	100	100	20	7	Total= 257

After crossover operation:

s_1'	36	5	100	20	7	0	0	89	Total= 257

s_2'	30	104	23	33	67	Total= 257

Fig. 3. Crossover operation example for total load equal to 257

3.4 Mutation Operation

The mutation is asexual operation, it operates on one individual. It randomly selects two mutation points in the selected individual from the current generation. Then sum their loads and distribute the result approximately equally between the original two nodes, as shown in Fig. 4.

s	40	5	23	33	67	0	0	89	Total= 257

s'	64	5	23	33	67	0	0	65	Total= 257

Fig. 4. Mutation operation example

3.5 The Proposed Algorithm

Algorithm: GA Cloud load Balancing (N, t, g)

<u>Input</u>: (Vector of length N contains the current load on N clouds, the total load t and the maximum generation limit g)
<u>Output</u>: Vector of length k contains the balanced workload on k clouds

Create m vectors of variable lengths k_m (N/2<= k <= N)

```
In each vector Store: random integer numbers
(the sum of these numbers must equal to t) and
av L.

Do while not (converge OR reach g limit)
        Use Eq. 2 to evaluate all individuals in
        the current generation
        If there is at least one individual with
        optimal fitness value, then converge with
        that individual.
        Otherwise
        Do
        Apply the following Genetic Operations
        to produce a new generation Contains m
        vectors:
                • Reproduction: the
                  individual with best
                  fitness value is
                  transmitted to the next
                  generation.
                • Crossover: between two
                  selected individuals
                  (based on their fitness
                  values).
                • Mutation: on a randomly
                  selected individual.
            End Do
End While
```

Instead of waiting for the GA to converge, it will be allowed to run for a fixed number of g cycles. The decision was made because solutions generated in less than g generations may not be good enough. On the other hand, running the GA for more than g generations may not be very feasible; too much time will be devoted to genetic operations. Although we limited k between two values, however its value must be selected by using heuristic algorithm as specified in the next subsection. Time complexity analysis can be used to predict the growth behavior of an algorithm and is useful for analyzing and optimizing the real time efficiency of the algorithm. There are two importance parameters affecting time complexity of GA: population size m and the maximum number of generation g. The evaluations of the fitness function also affect the time complexity. Because our fitness function is simple so that the time complexity is equal to O (mg).

4. RESULTS

First generation affects the quality of the future generations, and it is an important step in the whole algorithm. In this paper, this step is conducted by combing the random and greedy initialization methods. Each load represents either the current server's workload or the current average workload of a cluster of servers. Table 1; show some results (for the same number of clouds), the number of generations vary according to the average number of load.

Table 1. Simulation results for different workload size with k=8

Average No. of load	k	No of generation
20	8	500
50	8	1000
100	8	1060
120	8	10090
150	8	12040

Table 2; show some result (for different number of clouds), the number of generation vary according to the average number of load and the number of cloud.

Table 2. Simulation result for different cloud number

Average No. of load	k	No of generation
20	4	434
50	6	808
100	10	967
120	12	1874
150	14	1421

However, the proposed algorithm converges to wrong states as one in Fig. 5, where the length of the output vector is equal to one. To avoid such wrong cases, individual's length is set to be more than or equal to (N/2) and less than or equal to N. However, a good heuristic algorithm must be used to find the optimal value for k.

Cloud minimization is satisfied when the workload is low with respect to N, the algorithm is converges with k less than N, because the fitness values of lower length individuals is better than that of higher length individuals, as shown in Table 3. Of course, a substantial reduction in energy consumption can be made by powering down servers when they are not in use.

Table 3. Cloud minimization (N is minimized to k)

Workload	N	k
10	20	10
30	60	35
60	100	67
90	200	115
100	250	133

Cloud number	1	2	3	4	5	6	7	8	Total load
Load	12	300	500	88	7	0	3	69	979

a: individual with k=10 (unbalanced)

Cloud number	1	Total load
Load	979	979

b: individual with k=1 (balanced)

Fig. 5. Genetic algorithm converge into wrong output

5. CONCLUSION

This paper searches to find an optimal load vector of N nodes. This vector has variable length so that N may be minimized. The proposed algorithm avoids overheating by balancing the workload across all the nodes, hence reducing the amount of energy consumed. Reduction in energy consumption can be made by powering down servers when they are not in use. The existing load balancing techniques in clouds, consider various parameters like performance, response time, scalability, throughput, resource utilization, fault tolerance, migration time and associated overhead. But, for an energy-efficient load balancing metrics like energy consumption and carbon emission should also be considered which will help to achieve Green computing. Multiple objective fitness function will be used in the future to satisfy more realistic result, for example the priority factors may be consider as an objective in the fitness function. In addition k parameter must be selected by using heuristic algorithm especially when the number of servers in a cloud increases vastly.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Dung H. Phan, Junichi Suzuki, Raymond Carroll, et al. Evolutionary multiobjective optimization for green clouds GECCO 12: The proceedings of the 14th annual conference companion on Genetic and evolutionary computation. NY USA. 2012;19-26. Copyright 2012. ACM 978-1-4503-1177-9/12/07. ISBN:978-1-4503-1178-6. Doi:10.1145/2330784.2330788.
2. Jing Liu, Xing-GuoLuo, Xing-Ming Zhang, Fan Zhang, Bai-Nan Li. Job scheduling model for cloud computing based on multiobjective genetic algorithm. IJCSI International Journal of Computer Science. 2013;10(1No 3):1694-0814. ISSN (Print):1694-0784 | ISSN (Online).
3. Yogita Chawla, Mansi Bhonsle. A study on scheduling methods in cloud computing. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS). 2012;1:3. Available:www.ijettcs. org
4. Truong Vinh Truong Duy, Yukinori Sato, Yasushi Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW); 2010 .
5. Gary Garrison, Sanghyun Kim, Robin L. Wakefield. Success factors for deploying cloud computing. Communications of the ACM. 2012;55:9. DOI:10.1145/2330667.2330685.
6. Zhongni ZH, Wang R, Hai ZH, Xuejie ZH, “An approach for cloud resource scheduling based on parallel genetic algorithm”. IEEE ICCRD. 2011;2:444-447.
7. Li J, Qian W, Cong W, Ning C, Kui R, Wenjing L. “Fuzzy keyword search over encrypted data in cloud computing”. IEEE Infocom. 2010;15.
8. Singh RM, Sendhil Kumar KS, Jaisankar N. “Comparison of probabilistic optimization algorithms for resource scheduling in cloud computing environment”. International Journal of Engineering and Technology (IJET). 2013;5(2):1419-1427.
9. Fang Y, Wang F, Ge J. A task scheduling algorithm based on load balancing in cloud computing. Springer Web Information Systems and Mining. 2010;6318:271-277.
10. Chen SH, Wu J, Lu ZH. A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness. IEEE 12th ICCIT. 2012;177-184.
11. Saeed Javanmardi, Mohammad Shojafar, Danilo Amendola, Nicola Cordeschi,

Hongbo Liu, Ajith Abraham. Hybrid job scheduling algorithm for cloud computing environment. Springer Verlag Berlin Heidelberg; 2014.

12. Pushpendra Verma, Jayant Shekhar, Amit Asthana. A model for evaluating and maintaining load balancing in cloud computing. IJCSMC. 2014;3(3):501-509. ISSN:2320-088X.
13. Sahu Y, Pateriya RK, Gupta RK. Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm. CICN' 13 Proceedings of the 2013 5th International Conference on Computational Intelligence and Communication Networks, IEEE Computer Society Washington, DC, USA. 2013;527-531. ISBN:978-0-7695-5069-5. DOI:10.1109/CICN.2013.114.