Indian Journal of Science and Technology, Vol 9(18), DOI: 10.17485/ijst/2016/v9i18/90697, May 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Performance Analysis of Load Balancing Algorithms in Cloud Computing Environment

V. Krishna Reddy*, K. Deva Surya, M. Sai Praveen, B. Lokesh, A. Vishal and K. Akhil

Department of CSE, K L University, Vaddeswaram, Guntur - 522502, Andhra Pradesh, India;
vkrishnareddy@kluniversity.in, suryaadsk@gmail.com, praveensai96@gmail.com,
lokeshbadri21@gmail.com, akhilkanigcharla@gmail.com, vishal.arlagadda@gmail.com

Abstract

Background/Objectives: In this paper, the performance of cloud computing system is greatly influenced by the problem of load balancing. The complexity class of the load balancing problem with respect to the complexity class belongs to the NP-system complete which involves intensely huge search space with huge number of potential solutions and also to find the optimal solution, it takes longer time. Based on these circumstances, there is no methodology to solve the problem. **Methods/Statistical Analysis:** In the cloud, we can find a near optimal solution, within a brief span of time. In this situation IT practitioners are focusing on heuristic methods. This paper proposes a multi objective load balancing in a cloud computing environment. **Findings:** This model can be applied to schedule the tasks on to the distinct data center resources. Ant Colony Optimization (ACO) algorithm is considered to know the optimal solution. Application/ **Improvements:** Experimental results show that proposed model exceeds existing models in terms of reduction in energy consumption, improving pool utilization, minimizing the number of active nodes.

Keywords: Ant Colony Optimization (ACO), Cloud Computing, FCFS Algorithm, Genetic Algorithm

1. Introduction

Cloud computing has recently received much of its attention, for Communication Technologies (ICT) services and for information delivery. Improving the use of data center resources which can operate in dynamic workload environments is the best solution for provisioning these services. Datacenters are integral components of cloud computing. In the data center normallyhundreds of virtual servers[1] run at any instance of time, hosting several tasks and the batches of task requests are received by the cloud system at an identical time. In this situation, we need to find few target servers out more powered on servers, which canaccomplish a batch of tasks that are incoming. The performance of cloud service provider is highly influenced by Load balancing which is a very important issue.

The mechanisms used in the optimization are traditional which are deterministic and rapid. They give definite answers, but they tend to stuck in local optima. The complexity class of the load balancing problem with respect to the complexity class belongs to the NP-system complete which involves intensely huge search space with huge number of potential solutions and also to find the optimal solution, it takes longer time. Based on these circumstances,thereis no methodology to solve the problem. In the cloud, we can find an immediate optimal solution[2], within a brief period of time. In this situation IT practitioners are focusing on heuristic methods.

This paper is organized as follows: Section 2 provides Design of FCFS Algorithm Section 3 provides Genetic Algorithm[3] design. Design of a Multi-Objective Model for scheduling details is provided in Section 4. Ant Colony Optimization (ACO) Algorithm design details in Section 5. Section 6 summarizes Implementation and Result Analysis and Section 7 provides Conclusions and Future Scope.

2. Design of FCFS Algorithm

FCFS is a straight forward task requesting a dynamic load balancing algorithm. The handling happens by

*Author for correspondence

picking the correct jobs based on the request. With this plan, the client ask for which starts things out to the server farm checker, and it would be executed by assigning a Virtual Machine. The implementation[4] of FCFS approach is effortlessly deal with queue implementation. At that point the 1st request from the line is uprooted and is gone to one of the Virtual Machine through the Virtual Machine Load Balancer[5]. The distribution of demand happens by two types: First, the demands can be organized in a FIFO format and besides by assigning substantial capacity hub less work and low capacity hub extra work way. Numerous capacity parameters can be thought about keeping in mind the end goal to compute the perplexing load measuring value and the present genuine load measuring value. The full algorithm as shown in the Figure 1.

The current load calculation as follows:

$A(Bi) = D1 * CCPU(Bi) + D2 * BIOS\ usage + D3 * RBW + D4 *$

Tusage 1.

$C(Bi) = A * W(Bi) + B*(W(Bi) - A(Bi))1/3$ (2)

where

A(Bi) – real current capability on resources

Bi – Total number of resources

CCPU – Rate of Tenancy w.r.t CPU

Cusage – Usage of Cache[6].

Usage of BIOS – System usage regarding Basic Input and Output.

RBW – Tenancy Rate of Network Band Width.

Di – coefficients set of dynamic parameters.

W(Bi) – Static parameters default weighed value.

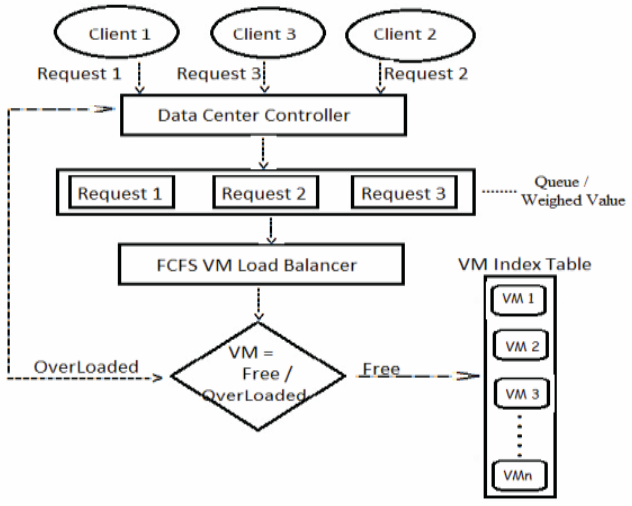

Figure 1. Flow chart of FCFS.

3. Design of a Genetic Algorithm (GA)

The batches of tasks are generated at fixed intervals, then GA based scheduler schedule them onto different resources of distributed data centers in the cloud. Each solution is scored and either accepted or rejected before considering it for the next generation. In order to solve a hard optimization problem using the paradigm of GA, Davis[7] pointed out that, the following GA components are necessary.

- A way to create some random initial population of solutions.
- An evaluation function to rate the solution in terms of their "Fitness".
- Alter the genetic structure of solutions by using operators like crossover and mutation.
- Values for GA[8] include population size, number of generations.

Step 1: Design of a Chromosome

Each resource is assigned a distinct number starting from 1 to R. Each task is assigned a unique number from 1 to V. By using the simple array data structure, we represent chromosome, the index of the array is represented using tasks and an array element is represented using resources. Allocation of tasks onto Resources is represented using each chromosome as shown in Figure 2.

The number of tasks is always equal to length of chromosomes[9]. In the chromosome each and every gene shows a resource. By showing the chromosome in this fashion, we are implicitly satisfying our first constraint, i.e. we are ensuring that on only one physical machine each task as shown 'In the Figure-2, tasks V_1, V_3 and V_v are hosted on a resource R_4.

Step 2: Initial Population

It is propagated based on the initial chromosome in which it is passed to initial population generator function. So that it can propagate new chromosomes from which it can design population of an initial generation.

Step 3: Generator Function

The primary elements to design a new chromosome are the gene values of chromosomes. The digits are arbitrarily

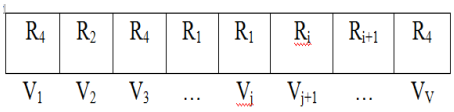

Figure 2. Design of a chromosome.

organized in the positions of the new chromosome. 100 unique and fit individuals are generated. After this initialization step, we do an iterative process. Each cycle of it is called a generation, where we:

- Choose the two parents based on roulette wheel selection.
- Create the new individuals by a crossover operator.
- Mutate some variables of the new individual with a mutation operator.
- Calculate the fitness of the individual, by taking care of fixing the constraints.
- Insert this individual in the population, and remove the individual with the lowest fitness.

Step 4: Termination Condition

For the first few runs, it is observed that the cost value of the best solution in the population is large improved. Smaller improvement was observed for the next few hundreds of generations. After a thousand generations, very small improvement was observed. Depending on the rate of improvement, GA was terminated which was observed for every successive hundred generations.

4. Design of a Multi-Objective Model for Scheduling

4.1 Multi-objective Optimization

In a multi-objective optimization every solution is measured according to more than one objective function, each of which must be minimized or maximized[10]. Let Q be the number of objectives in a multi-objective problem, let S be the set of feasible solutions and let Z denote the set of all feasible solutions which contain objective value vectors. Each element of Z is a vector, referred to as objective vector formed by the values of the objective functions. Formally,

$$Z = \left\{ z_i = \frac{f_i(s)}{} \forall 1 \le i \le Q, \forall s \in S, \exists z \in Z \right\}$$

The feasible set S is a subset of a space. That feasible set is called the search space. Objective space is defined as the space from which objective values are taken.

Let us assume that all Q objectives should be minimized, without loss of generality. Thus, the goal of multi-objective optimization is to solve

$$\min \left\{ \frac{f_i(s)}{\forall} 1 \le i \le Q, s \in S \right\}$$

Where the meaning of "min", i.e., the concept of optimality in multi-objective optimization.

This section proposes a mathematical model based on Cost of Migration, Cost of down time Cost of Execution and Cost of Communication cost. The objective of the problem is minimizing the total cost.

The performance of cloud computing[11] is greatly influenced by load balancing. This problem is NP-Complete. This paper proposes a mathematical model based on migration cost, execution cost, operational cost, communication cost and down time cost. The objective of the problem is minimizing the total cost. To find out optimal solution for this problem Ant Colony Optimization algorithm is proposed.

Notation Description:

Notation	Description
R	Total Number of Resources
T	Total Number of Tasks
P_i	Maximum processing capacity of the i^{th} resource.
M_i	Maximum physical memory capacity i^{th} resource.
N_i	Maximum network bandwidth capacity of the i^{th} resource.
p_j	j^{th} task Processing capacity requirements
m_j	j^{th} task Memory capacity requirements
n_j	j^{th} task Network band width requirements
C_M	Migration cost per bytes memory footprint of a task
C_{Dj}	Cost caused by downtime during migration of task
C_{Oi}	Operational cost of the i^{th} resource.
x_{ji}	If j^{th} task is assigned on i^{th} resource then x_{ji}=1 other wise 0
X_i	If i^{th} resource is powered on then X_i=1 otherwise 0

4.2 Objective Function

To find out the optimal placement of virtual machines[4], we consider three costs.

- Cost of Migration, T_M: During migration[12], Overhead cost is obtained to migrate the memory footprint of virtual machines.

$$T_M = \sum_{i=1}^{R}\sum_{J=1}^{T} C_M * m_j * x_{ij}$$

- Cost of Downtime, T_D: During the migration of the virtual machine j, cost caused by downtime. During the execution of live migration, services are interrupted for an extremely short time. To assure the SLA with a huge priority service running on a virtual machine, the interruption of service should be minimized. To decide candidate virtual machines for live migration[13], this QoS loss can also be thought of the migration priority.

$$TD = \sum_{i=1}^{R}\sum_{j=1}^{T} C_{Dj} * x_{ji}$$

- Operational Overhead Cost, T_o: It is added to omit non-essential operation of computational resources. It comprises an electricity power, rental or purchasing cost and cooling, server administrative costs and others.

$$T_O = \sum_{i=1}^{R} C_{Oi} * x_i$$

- Cost of Execution (T_E): Execution cost based on Work load and processing capacity.

$$T_E = (Wj/ P_i) * x_{ji}$$

- Cost of communication, T_c: In a heterogeneous cloud environment tasks may need to communicate with other tasks which are hosted on other resources. In a cloud computing environment, resources are distributed and the cost of communication between resources is computed using predefined RxR cost matrix. This matrix values can be obtained based on the edge costs between resources. If two tasks are assigned to the same resource then cost of communication is zero. Otherwise the cost of communication is calculated by using Floyd–Warsh all algorithms.

4.3 Constraints

- A task is executing on only one resource at any instance.
- The task should host on the resource in which it is operating.
- Total preparing necessities of all tasks facilitated on asset should not go past the aggregate handling limit of that asset.
- Total memory prerequisites of all undertakings facilitated on asset should not go past the aggregate memory accessible on that asset.
- Total organize data transmission limit prerequisites of all errands facilitated on asset should not go past the aggregate system transfer speed limit of the asset Mathematical Model.

$$\text{Minimize Total Cost (T)} = T_M + T_D + T_o + T_E + T_c \quad (1)$$

Subjected to Constraints

$$\sum_{i \in R} x_{ji} = 1 \quad (2)$$

$$x_{ji} \le X_i \quad (3)$$

$$\sum_{j \in P} p_j x_{ji} \le P_i \quad (4)$$

$$\sum_{j \in R} m_j x_{ji} \le M_i \quad (5)$$

$$\sum_{j \in R} n_j * x_{ji} \le N_i \quad (6)$$

$$1 \le i \le R \quad (7)$$

$$1 \le j \le T \quad (8)$$

$$X_{ji} \in \{0, 1\} \text{ and } x_{ji} \in \{0, 1\} \quad (9)$$

5. Design of a Ant Colony Optimization (ACO) Algorithm

This Section deals with the design of an Ant Colony Optimization (ACO) algorithm[14], where on the cloud computing environment this algorithm targets to allocate tasks onto available resources. The best is chosen from assignments accessible which are contrasted and the accessible asset. In a cloud domain for assessing the algorithm which is ant based algorithm, execution times from the recreation are utilized. For each assignment on each asset to speak to the handling time expected, the preparing time network is utilized; before cloud scheduling is started this is done. The objective is to find an appropriate task assignment with minimum cost.

Construction graph: The TRAP can simply be transmitted into the outline of the ACO meta-heuristic. For

instance, the setback might correspond to the construction graph TRG = (V, E) in which the set of components consists of the set of tasks and agents, that is, V = T U R. Every task, consists of costs d_{ij} is linked with all probable coupling (i, j) of tasks and agents and n couplings (i, j) of tasks and agents, correspond to at least one ant's walk on this graph

5.1 Pheromone Trails and Heuristic Information

Throughout the building of a resolution ants frequently have to capture the subsequent two points: 1. Select the task to allocate subsequently and 2. Decide the resource the task be supposed to be assigned to. Heuristic information and pheromone trail can be connected with any of the two points: it can be used to study a suitable sort for task coursework or it can be connected with the interest of handing over a task to a particular resource. In the first case, represent the interest of handing over task j straight after task i, while in the next case it represent the interest of handing over resource j to task i. In the same way, heuristic information can be connected with any of the two points. For instance, heuristic information can take task assignment[15] to tasks that can use more resources, and take the option of agents in a way that modest assignment costs are incurred and the agent desires a comparatively little quantity of its accessible resources for carrying the task out.

Solution construction: By selecting the components it can be usually performed for adding that to the partial solution from among those as explained above, which satisfy the constraints with a probability combined by the heuristic information and pheromone trails.

5.2 Importance of Proposed System

The pheromone trial is assumed only by the ant algorithm[y]. (t_{ij}) level and heuristic information (h_{ij}) value for conniving the probability matrix value. The drawback of the method is that column values in the probability matrix have the same probability value. In the computational grid situation, the algorithm was mostly proposed for generating high throughput. In this algorithm, the parameter values used are τ_0 = 0.01 and ρ = 0.3and they have used only one ant. The single ant ACO[16]algorithmconverges in a very rapid way, but it will generate bad result. Therefore a proposed MOACO algorithm may well be proposed to rise above the challenge of the presented ACO algorithm and to obtain high throughput computing and high performance computing.

First a set of jobs and a set of different Data Centers (Resources), the ant will allocate the jobs one by one with data centers until all the jobs are allocated or none of remaining jobs are allocated without exceeding its capacity. Infeasible solution can be defined here as if there is at least one task that is unallocated and then the ant stops and on the other way feasible solution refers to stopping on ant after all tasks assigned. If the both cases does not occur then the solution is known as partial solution to select the cell that is next we should visit i.e. (Data Centre, Jobs), the ant uses a distribution that is heuristic over the set of pairs that are eligible and pheromone that is weighted average.

5.3 Multi Objective Optimization using Ant Colonies

The ant colony objectives can be necessarily of three types: 1. Based on the each objective performance the solution to bring the pheromone up to date. 2. How does ant select the path locally based on the desirability and visibility? 3. The pare to front development.

5.4 Multi Objective Ant Colony Algorithms (MOACA)

Step 1: The details about data centers and tasks should be gathered.

Step 2: The value of all parameters should get initialized.

Step 3: Step 4 and step 5 for each ant should be repeated

Step 4: Select the jobs (T_i) and data centers (R_j).

- Assign (T_i, R_j, availability [j], availability [j]+PT_{ij}) to the output list.

Step 5: Repeat the below steps until execution of all the tasks.

- Heuristic information calculation (η_{ij})
- Current pheromone trail value calculation □□ij
- Pheromone trail matrix updating.
- Probability matrix calculation.
- As the next task T_i to be executed on the resource R_j, select the task with highest probabilities of i and j
- Remove the task T_i from the unscheduled list
- Modify the resource free time availability [j] = availability [j] + PT_{ij}

Step 6: Finding the suitable and best optimal solution by analyzing scheduling list of all the ants.

5.4.1 Calculate the Heuristic Information (η_{ij})

A heuristic[17] value, if connected with the components is said to be η_i, other wise if correlated with connections among resource i and resource j, it is said to be η_{ij}. In most of the cases η is the cost of connection of the solution under construction or adding the component.

The heuristic information η_{ij} is inversely proportional to the distance among cities I and *j*

$$\eta_{ij} = 1/cc_{ij}$$

5.5 Calculate Pheromone Trail Value (□□ij):

Pheromone trail value can be useful for knowing a suitable order for assignment of tasks.

For framing a solution of the scheduling, ants visit edges and their pheromone level changes to locally update the rule. Based on high pheromone level[10], ants choose new resource, the rule decreases the convergence. Thus, for the following ants this resource becomes less desirable, if the pheromone trail is decreased. This is achieved by Equation

$$\tau_{ij} = \rho\, T_{ij} + \Delta\tau_{ij} \text{ where } 0 < \rho < 1 \text{ and } 1 \le i, j \le N$$

Where $\Delta\tau_{ij} = (1 - \rho)/f_k$ where $0 < \rho < 1$

At each iteration the ants measure the minimized function 'f_k' for k^{th} ant.

5.6 Calculate the Probability Matrix

Using the random proportional rule, with probability P_{ij} current ant in resource i choose to go to resource j.

$$P_{ij} = \frac{T_{ij} \cdot \eta_{ij}\left(\frac{1}{CT_{ij}}\right)}{\sum T_{ij} \cdot \eta_{ij}\left(\frac{1}{CT_{ij}}\right)}$$

6. Experimental Results and Analysis

The test program was developed in C language running on Windows XP with 4GB of RAM and using an Intel Pentium 4 3.0-GHz process or run 50 times on each problem. To test the algorithm, we have used ten ant sizes *N* (10,20,30,40,50,60,70,80,90,100)during a pilot experiment. The parameters of algorithm are fixed as follows: Evaporation value of pheromone (□□□ Value of Initial pheromone deposit ($\square_0$) = 0.01, Pheromone Importance (α) = 1, Resource innate attribute importance (□) = 2, Available resources are represented in one dimensional matrix is Availability [1 …N]

The comparison was done with the proposed PSO[4] algorithm and with Genetic algorithm. But PSO achieves its effectiveness in a wide range of task with low computational cost over the genetic algorithm.

This section tells about the comparison, the experimental data, the results and result analysis. As performance measure shown in Table 1. Results of FCFS, GA and ACO based on Simulation. Total costs using three heuristics algorithms are computed: FCFS, ACO based on cost optimization, and GA selecting a resource based on minimum cost.

The total numbers of tasks that are placed 10 to 100, the processing time of each task and the memory requirement are in uniform distribution [5, 10] and [50, 100] respectively. The Data Centers are numbered from 5 to 20, the uniform distribution of total available memory is in [250, 500]. The communications among Data Centers are varied by the uniform distribution from 1 to 10 and the interactive data among of tasks are varying from 1 to 10. Simulation results demonstrate that more iteration have obtained the better solution since more solutions were generated as displayed in Table 1 and Figure 3.

Table 1. Simulation results of FCFS, GA and ACO

Task resource	FCFS	GA	ACO
(10,5)	1605	1512	1445
(20,5)	2256	2065	1843
(30,10)	3671	2248	1856
(40,10)	5028	3765	1945
(50,15)	4526	4966	3800
(60,15)	6855	6140	5095
(70,20)	7523	6678	5392
(80,20)	8642	7395	6432
(90,20)	9315	8432	7254
(100,20)	10800	9744	8965

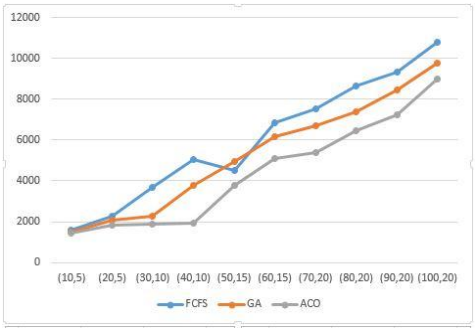

Figure 3. Experimental observations of FCFS, GA and ACO.

7. Conclusions and Future Scope

We have presented a MOACO method for solving a load balancing difficulty with multiple cost objectives. Originally, for continuous optimization problems ACO was proposed. We altered the representation of heuristic information, pheromone trail value, and the probability matrix to make an ACO applicable for load balancing, which is a leading problem. The outcomes demonstrated that the proposed ACO could yield high cost efficient solutions than a genetic algorithm. The performance measurement also revealed that the proposed MOACO algorithm outperformed the GA heuristics in terms of cost minimization.

In future research, we make an endeavor to compare our ACO to more heuristics such as Particle Swarm Optimization (PSO) with multiple objectives. Concerns relevant to Pareto optimality, such as performance measurement and solution maintenance strategy are also of interest for further study.

8. References

1. Sullivan D. The Definitive Guide to Cloud Computing. Realtime Publishers; 2010.
2. Reddy VK, Rao BT, Reddy LSS. Research Issues in cloud computing global. Journal of Computer Science and Technology. 2011; 11(11):59–64.
3. Goldberg DE. Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.; 1989.
4. Pattanaik PA, Roy S, Pattnaik PK. Performance study of some dynamic load balancing algorithms in cloud computing environment. 2nd International Conference on Signal Processing and Integrated Networks (SPIN); Noida. 2015 Feb 19-20. p. 619–24.
5. Reddy VK, Velamuri SR. A survey of various load balancing algorithms in cloud computing. JCOM. 2013; 1(1):1–6.
6. Nagaraju S, Parthiban L. SecAuthn: Provably secure multi-factor authentication for the cloud computing systems. Indian Journal of Science and Technology. 2016 Mar; 9(9):1–18.
7. Ping WJ, Zhu YL, Feng HY. A multi-load balancing method based on ant colony algorithm combined QoS in cloud computing. International Journal on Advances in Information Sciences and Service Sciences. 2012; 4(11):185–92.
8. *Davis L.* Handbook of Genetic *Algorithms*. New York: Van Nostrand Reinhold; 1991.
9. Nagamani M, Chandrasekaran E. Single objective for partial flexible open shop scheduling problem using hybrid particle swarm optimization algorithms. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–6.
10. Shyamala K, Rani TS. An Analysis on efficient resource allocation mechanisms in cloud computing. Indian Journal of Science and Technology. 2015 May; 8(9):814–21.
11. Raju IRK, Varma PS, Sundari MVR, Moses GJ. Deadline aware two stage scheduling algorithm in cloud computing. Indian Journal of Science and Technology. 2016 Jan; 9(4):1–10.
12. Jasmine RM, Nishibha GM. Public cloud secure group sharing and accessing in cloud computing. Indian Journal of Science and Technology. 2015 Jul; 8(15):1–7.
13. Hariharan T, Sundaram KM. Optimal power flow using hybrid intelligent algorithm. Indian Journal of Science and Technology. 2015 Sep; 8(23):1–9.
14. Apinanthana U, Khachitvichyanukul V. Ant colony algorithm for multi-criteria job shop scheduling to minimize makes pan, mean flow time and mean tardiness. International Journal of Management Science and Engineering Management. 2011; 6(2):116–22.
15. Job T, Kovoor BC, Jose J, Krishnan CE. Ant colony optimization module for scheduling of resource constrained construction project. Project Management Research. 2012 Aug; 1–6.
16. Kokilavani T, Amalarethinam DG. Memory constrained ant colony system for load balancing in grid computing. IJGCA. 2012 Sep; 3(3):11–20.
17. Reddy VK, Reddy LSS. Design of particle swarm optimization algorithm for multi-objective load balancing in cloud computing environment. GJMECS. 2012; 2(1):1–5.