World Engineering & Applied Sciences Journal 7 (1): 10-17, 2016
ISSN 2079-2204

DOI: 10.5829/idosi.weasj.2016.7.1.22540

Optimized Resource Scheduling Using Classification and Regression Tree and Modified Bacterial Foraging Optimization Algorithm

[1]B.E. Anusha Bamini and [2]M.E. Sharmini Enoch

[1]Department of Computer Science and Engineering, Noorul Islam University, India
[2]Department of Electronics Communication Engineering, Noorul Islam University, India

Abstract: Cloud computing offers a great solution to distributed computing problems. Cloud provides a virtualized environment to share resources dynamically. Users are charged on a pay-per-use concept on the basis of resource used. Resource scheduling is an important and challenging issue of Cloud computing. Existing solutions to resource scheduling problems are only focus on a specific purpose, like the minimization of execution time or workload and do not use characteristics of Cloud computing for resource scheduling. A resource scheduler in Cloud computing has to satisfy cloud users with the agreed Quality of Service (QoS) and improve profits of cloud providers. In this paper, a Classification and Regression Tree (CART) workflow and a Modified Bacterial Foraging Optimization Algorithm (MBFOA) are proposed to efficiently schedule resources among the submitted jobs. To reduce the time of resource allocation and task scheduling, MBFOA is suggested. In this work CPU, RAM and Bandwidth resources are considered over the parameter time. The time for resource allocation was reduced by applying the MBFOA algorithm. Experiments were carried out in the CloudSim tool. The results show better performance.

Key words: Workflow · Scheduling · CART · MBFOA

INTRODUCTION

Cloud computing is a new paradigm for distributed computing that delivers infrastructure, platform and software as services. It is a technology specially built to manage such kind of data and provide it whenever needed. They provide services, manipulate data and share with the other servers. In order to efficiently and cost effectively schedule the tasks and data of applications onto these cloud computing environments, application schedulers have different policies that vary according to the objective functions: minimize total execution time, minimize total cost to execute, balance the load on resources used while meeting the deadline constraints of the application and so forth. In this paper, minimizing the total execution time of applications on these resources provided by Cloud service providers is focused. Ultimately, the Classification and Regression Tree as the basic workflow for dynamic distribution jobs that takes into consideration the heterogeneity of jobs and the Modified Bacterial Foraging Optimization Algorithm are proposed to increase the optimization [1, 2].

Many applications run in these environments have a complex structure and can be modeled as Directed Acyclic Graph (DAG) workflows [3]. For instance, scientific workflows and clouds [4] are considered as a new challenge on how to leverage the computing power of cloud computing on science workflow applications. Specifically, each "job" submitted comprises of a number of tasks that are mapped to the nodes of a DAG and hence their execution is subject to given precedence constraints; i.e. the execution of a downstream task cannot begin until all its precedent ones have been completed. The problem of DAG scheduling has been extensively investigated in grid computing [5]. To overcome this problem DAG is replaced with Classification and Regression Tree. CART algorithm will search for all possible resources in order to find the best resource.

Job scheduling in cloud computing is mainly focused for improving the resources utilization such as bandwidth, memory, CPU and reduce completion time. To increase the performance and to get the optimized result, the modified

Corresponding Author: B.E. Anusha Bamini, Department of Computer Science and Engineering, Noorul Islam University, India.

bacterial foraging optimization algorithm was applied. The following modifications are done in the modified BFOA:

- A single loop to include the chemotactic, reproduction and elimination-dispersal steps
- A definition of the step-size values based on the features of the problem
- A constraint-handling mechanism
- A simple communication mechanism among bacteria to allow them to move towards promising regions of the search space.

The major contributions of this work are:

- Applying Classification and Regression Tree to utilize the best resource.
- Developing MBOFA for optimized resource scheduling.
- Resource utilization over time was compared and simulated in cloudsim platform.

The paper is structured as follows: Section 2 starts with scheduling using CART. Section 3 deals with Bacterial Foraging Optimization Algorithm. Section 4 covers the Modified Bacterial Foraging Algorithm and the final section is the conclusion.

Job Scheduling Using Cart: In job scheduling the various Jobs submitted from different users are independent of each others. Each job can be sub divided into large number of tasks. DAG (Directed Acyclic Graph) is used to represent the possible dependency exists among this tasks. Large numbers of users submit their tasks at any time and various requirements.

Due to the drawback in the DAG workflow Classification and Regression Tree (CART) is applied in this work. Classification and regression trees are machine-learning methods for constructing prediction models from data and used for creating different decision rules quit effectively. The models are obtained by recursively partitioning the data space and fitting a simple prediction model within each partition. As a result, the partitioning can be represented graphically as a decision tree. Here, (1) the classification tree is used for selecting the best Virtual Machine (VM), which will exactly satisfy the user needs, (2) Regression tree is applied to decide how many resources can be provided to the user. The job scheduling is performed in two ways,

- Service-level scheduling (Task-to-Service)
- Task-level scheduling (Task-to-VM)

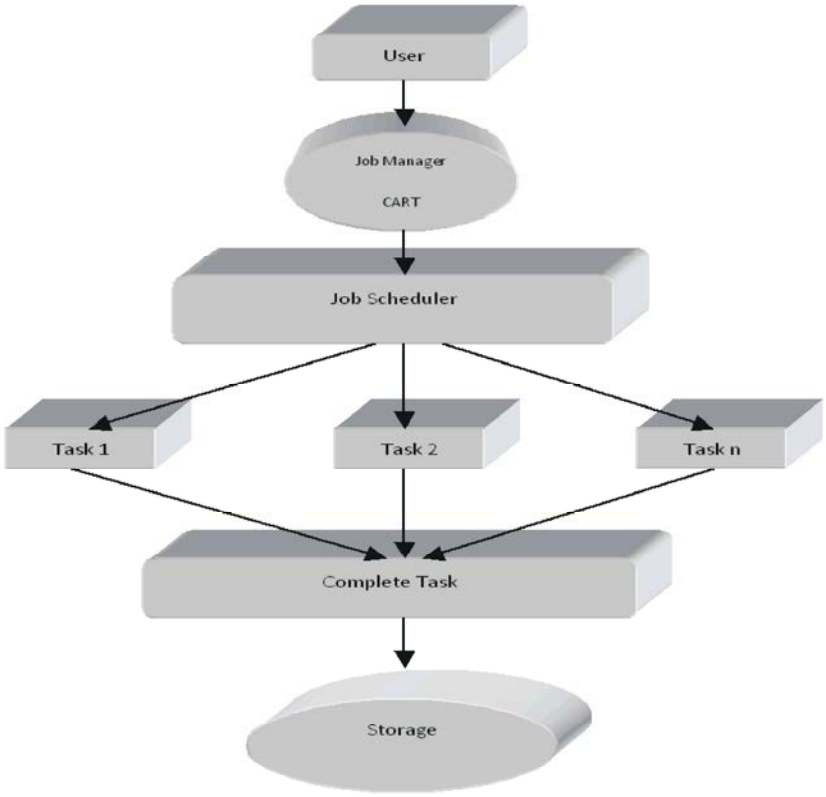

Fig. 1: Task Scheduling Architecture

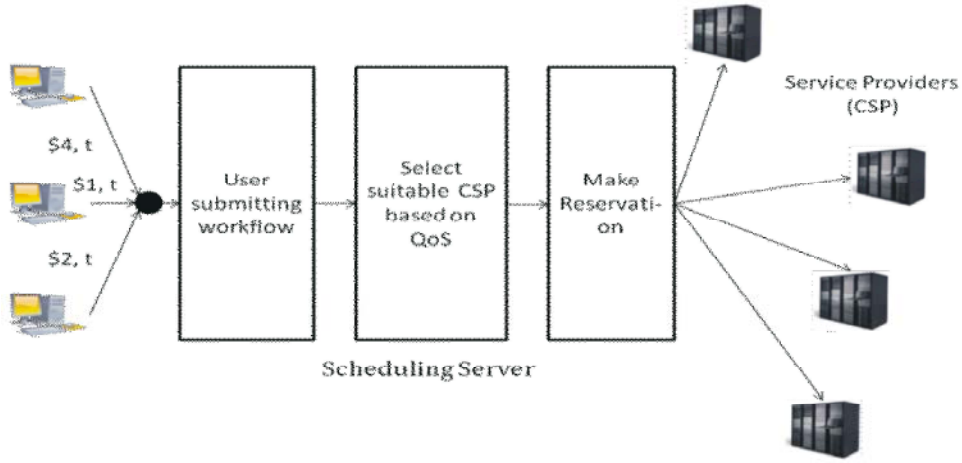

Fig. 2: Selecting Cloud Service Provider

A job with n subtasks is shown in Figure 1, where each subtask represents one unit of instructions. We suppose that only two resources are available. In Figure 1, we show a feasible schedule for one job over two resources, when each resource has the same processing power equal to one instruction per time unit. The scheduling requires 4 time units to finish a job; on the contrary, if there are two jobs in the system, we can see that two jobs can be completed within five time units. While the first set of N jobs is being processed, the second set of N jobs is gathered. Once the first group finishes execution and the second set of size N has been gathered, the algorithm proceeds in scheduling, while another set of N jobs is being gathered and so forth. Since jobs are aggregated and mapped at the same time and resource utilization increases. Conversely, the scheduling holes, i.e., the amount of time that resources are idle after scheduling [6].

Service – Level Scheduling: In this scheduling model, various metrics are calculated which are required for decision making of job scheduling which includes user preferences such as 'importance' and 'urgency', cloud resource such as task payload, time parameters, CPU states etc. Next the algorithm conducts analysis to find out parameters which are urgent and important for decision making analysis and then passes on these extracted parameters to the regression tree algorithm. Then this algorithm basically develops multiple linear discriminate functions as per regression tree algorithm. Finally the algorithm will help in taking the dynamic decision based on the current scenarios for scheduling the jobs to appropriate Cloud Service Providers (CSP). The service level scheduling allocates suitable service from CSP using Global Scheduler. It has the following three steps [7],

Step 1: Search and produce suitable service based on user requirements

Step 2: Select best service based on QoS parameters

Step 3: Make reservation to particular service

Fig. 2. shows the allocation of cloud service provider from multiple service providers. When the services are requested to the cloud, the best cloud service provider is allocated to that service based on QoS parameter. The suitable CSP is selected and it gets reserved to that particular service [8].

Task Level Scheduling: Task level scheduling is the part of resource management. The task-level scheduling is a type of dynamic scheduling which aims to optimize the Task-to-VM assignment so that the QoS constraints for each individual task will be satisfied. Therefore, in this strategy, the task-level scheduling only targets the tasks running in a specific data centre, i.e. the data centre which the cloud workflow system belongs to. Task level scheduling consists of the following three steps,

Step 1: Obtain QoS constraints for individual task

Step 2: Optimize the task to VM assignment

Step 3: Implement optimized scheduling algorithm (MBFOA)

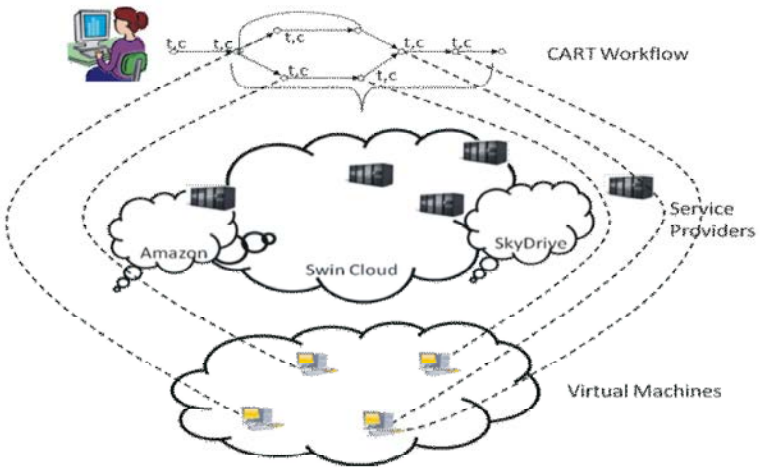

Fig. 3: Task Level Scheduling Structure

Bacterial Foraging Optimization Algorithm: The Bacterial Foraging Optimization Algorithm (BFOA) is based on some social and cooperative behavior found in nature. This task has been seen as an optimization process. The BFOA is inspired from the behavior of bacterium E. Coli in its search for food. This approach, considers three steps: (1) Chemotaxis, (2) reproduction and (3) elimination-dispersal of bacteria. BFOA has been successfully applied to solve different type of problems such as the identification of nonlinear dynamic systems. Furthermore, BFOA has been combined with other algorithms to solve multimodal optimization problems. The bacteria behave as follows [9]:

- Bacteria are randomly distributed in the map.
- They move to high-nutrient regions. Bacteria in convenient regions will be able to reproduce.
- Bacteria are located in promising regions and they are able to communicate.
- Bacteria are now located in the highest-nutrient region.
- Bacteria now disperse as to look for new nutrient regions in the map.

The generation of a random search equation is,

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(x)\Delta(i)^{\tau}}} \quad (1)$$

where, $\Delta(i)^n$ is a n-dimensional randomly generated vector with elements within the interval: [-1, 1]. Each bacterium $\theta(j,k,l)$ (where j, k and l are the chemotactic, reproduction and elimination-dispersal counters) modifies its position in Eq.2, where S(i) is the stepsize for search direction $\phi(i)$

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + S(i)\phi(i) \quad (2)$$

Modified Bacterial Foraging Optimization Algorithm: BFOA requires seven parameters. Moreover BFOA lacks a mechanism to deal with the constraints. A novel method is proposed with the following modifications to the original approach.

A single generation loop is proposed to eliminate the four nested loops controlled by the number of chemotactic, reproduction and elimination-dispersal steps combined with the population size. In this generation loop, each bacterium will perform its own chemotactic loop. A single reproduction step and a single elimination-dispersal step are performed at the end of this generation loop. In this way, the N_s parameter is eliminated as the tumble-tumble or tumble-swim step will be only limited by N_c for each bacterium. Furthermore, the elimination-dispersal step is simplified because only the worst bacterium in the population is eliminated. As a result, N_{re}, N_{ed} and P_{ed} parameters are also eliminated and just the GMAX, (number of generations) parameter is added due to this proposed modification.

The value of the step-size $S(i)$ is not defined by the user. Instead, for each decision variable i, $S(i)$ is now computed by considering its lower and upper limits, L_i and U_i by using the following formula,

$$S_{new}(i) = P * \left(\frac{\Delta \vec{x_l}}{\sqrt{n}} \right) \quad (3)$$

where, $S_{new}(i)$ is the step-size, $\Delta \vec{x_i}$ is computed as U_i - L_i, N is the number of decision variable, P is the percentage of the total step-size to be used.

A parameter-less constraint-handling technique is added to BFOA. It is based on three feasibility criteria.

- Between two feasible bacteria, any one of the best objective value is selected.
- Between a feasible and infeasible bacterium, the feasible one is selected.
- Between two infeasible bacteria, the one with the lowest sum of constraint is selected. It is calculated using the formula:

$$\sum_{i=1}^{m} \max(0, g_i(\vec{x}))$$

A simple swarming mechanism is added to the redefined chemotactic step. Half way to the end of its chemotactic loop, each bacterium, instead of determining its search direction as pointed out in Eq. (1) and (2), uses a communication mechanism to bias its search direction to the neighborhood of the best bacterium so far in the current population. This search direction is defined in Eq. (4) as:

$$\theta^i(j+1,G) = \theta^i(j,G) + \beta(\theta^B(G) - \theta^i(j,G) \quad (4)$$

where, $\theta^i(j+1,G)$ and $\theta^i(j,G)$ are new and current positions of bacterium i, $\theta^B(G)$ is the current position of best bacterium. The remaining steps in the chemotactic loop will be performed as in Eq. (5) (tumble-swim, tumble-tumble).

$$\theta^i(j+1,G) = \theta^i(j,G) + S_{new}(i)\phi(i) \quad (5)$$

Four modifications are made to the original algorithm: (1) a simplification in the design of the algorithm, (2) a definition of the step-sizes used by the algorithm in order to keep the user from the fine-tuning of these parameters, (3) an effective but simple constraint-handling mechanism and (4) a swarming mechanism to promote collaboration among bacteria [10, 11].

In the algorithm the notations used are,

S_b	-	Number of bacteria
N_c	-	Chemotactic loop limit
S_r	-	Number of bateria for reproduction
β	-	Scaling factor
R	-	Percentage of initial stepsize
GMAX	-	Number of generations

Algorithm for MBFA:

Begin
Initialize input parameters (see caption of this figure)
Create a random initial swarm of bacteria $\theta^i(j,G)$ □*i, i=1,...,*S_b
  Evaluate f ($\theta^i(j,G)$) □*i, i=1,...,*S_b
    FOR *G = 1 to GMAX* ***DO***
      FOR *i = 1 to Sb* ***DO***
        FOR *j = 1 to Nc* ***DO***
          Perform the chemotactic step (tumble-swim, tumble-tumble or swarming) for bacterium $\theta^i(j,G)$ by using Eq. 5 and 4 and the set of feasibility criteria
        End FOR
      End FOR
      Perform the reproduction step by eliminating the S_r (half) worst bacteria and duplicating the other half, based on the feasibility criteria.
      Eliminate the worst bacterium è^w(j, G) in the current population, based on the feasibility criteria.
  End FOR
End FOR

Experiments and Results: In cloud computing, CloudSim is used as a simulation tool to implement this process. In CloudSim, the jobs or tasks are submitted by users. The submitted jobs are classified into different tasks. The user tasks are classified based on the parameters such as bandwidth, memory, completion time and resources utilization etc,. In CloudSim, the tasks are added to scheduling algorithm. After scheduling, CloudSim used to bind the single user tasks with single virtual machine through Datacenter Broker and run the tasks with the help

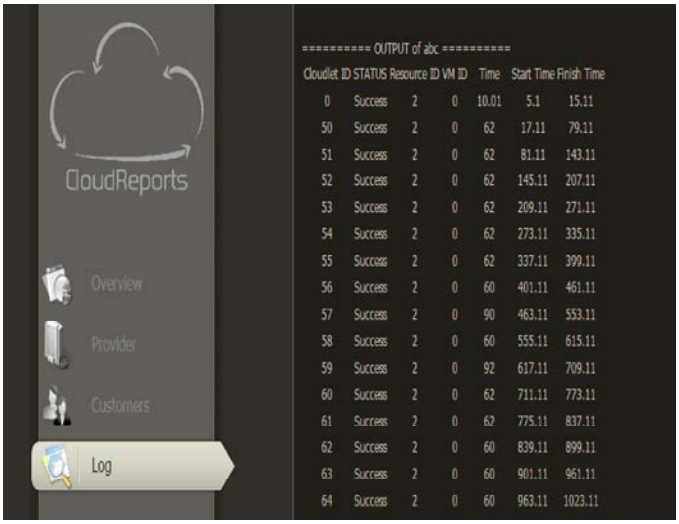

Fig. 4: Elapsed Time

Fig. 5: CPU Utilization

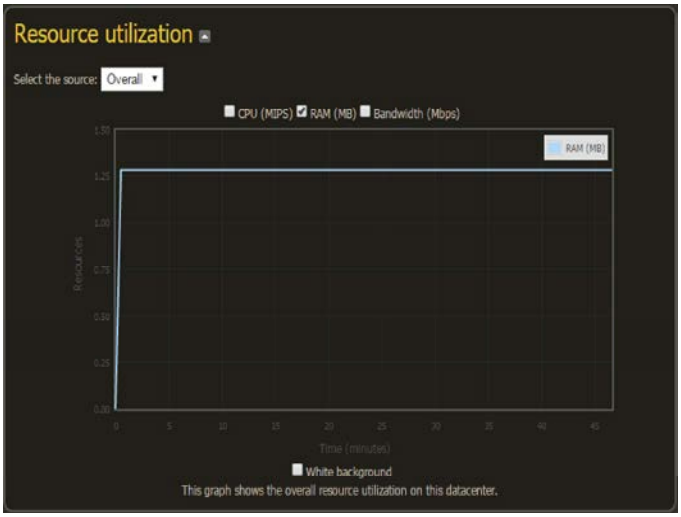

Fig. 6: RAM Utilization

Fig. 7: Overall Utilization

of virtual machine. CloudSim is implemented with the help of java. It consists of several datacenter for storing data. In a network, simulation contain server virtual machine can run in a single physical system. For binding Cloudlet (user tasks) with virtual machine CloudSim is used and allow the tasks for execution. It is used for supporting the search of reasonable virtual machine for a particular user tasks [12-15].

CONCLUSION

The proposed work shows the ability to assign specific virtual machine resources to specific tasks of a processing job, as well as the possibility to automatically allocate/deallocate virtual machines in the course of a job execution. The makespan and the cost are the major problem in this scalable computing area. In this work the CART is applied as the workflow instead of DAG to increase the allocation of resources and the MBFA algorithm was implemented using the CloudSim tool to minimize the overall time of execution and to get the optimized result.

REFERENCES

1. Xiangyu Lin, Chase Qishi Wu, 2015. End-to-End Delay Minimization for Scientific Workflows in Clouds under Budget Constraint, IEEE Transactions on Cloud Computing, 3(2).
2. Qi Zhang and Mohamed Faten Zhani, 2015. PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce, IEEE Transactions on Cloud Computing, 3(2).
3. Xua Yuming, Kenli Li and Ligang Heb, 2013. A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization, Journal of Parallel Distributed Computing, Elsevier, 73: 1306-1322.
4. Patel Swachil and Upendra Bhoi, 2013. Priority Based Job Scheduling Techniques In Cloud Computing: A Systematic Review, International Journal Of Scientific & Technology Research, 2(11).
5. Octavio Gutierrez, J. and Garcia Kwang, 2012. A Family of Heuristics for Agent-Based Elastic Cloud Bag-of-Tasks Concurrent Scheduling, Future Generation Computer Systems, Elsevier.
6. Yu, J. and R. Buyya, 2012. A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing, 3: 171-200.
7. Warneke Daniel, 2011. Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud, IEEE transactions on parallel and Distributed Systems, 22(6).
8. Juve, G. and E. Deelman, 2010. Scientific Workflows and Clouds, Computers & Operations Research, 16(3): 14-18.
9. Bajaj, R. and D. Agrawal, 2004. Improving Scheduling of Tasks in a Heterogeneous Environment, IEEE Transactions on Parallel and Distributed Systems, 15(2): 107-118.
10. Xu Yuming and Kenli Li, 2013. A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization, Journal of Parallel and Distributed Computing, pp: 1306-1322.

11. Cen Fang, Jia and Jinn Tsong, 2013. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm, Computers and Operations Research, Elesevier, pp: 3045-3055.
12. Ghanbari, Shamsollah and Mohamed Othman, 2012. A Priority based Job Scheduling Algorithm in Cloud Computing, Procedia Engineering, pp: 778-785.
13. Bessai Kahina and Samir Youcef, 2012. Bi-criteria workflow tasks allocation and scheduling in Cloud computing environments, IEEE Fifth International Conference on Cloud Computing, pp: 638-645.
14. Sheng Jun Xue and Wu Wu, 2012. Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm, TELKOMNIKA, 10(7): 1560-1566.
15. Zhang, L., Y. Chen, R. Sun, S. Jing and B. Yang, 2008. A Task Scheduling Algorithm Based on PSO for Grid Computing, International Journal of Computational Intelligence Research, 4(1).