Multivariate Logistic Regression based Gradient Descent Firefly Optimized Round Robin Scheduling with Big Data

C.R. Durga Devi, Ph.D Research Scholar, Department of Computer Science, NGM College, Pollachi, India.*

E-mail: deviswe@gmail.com

Dr.R. Manicka Chezian, Associate Professor, Department of Computer Science, NGM College, Pollachi, India.

E-mail: chezian_r@yahoo.co.in

Abstract--- A task scheduling with big data is to process and complete several tasks by managing the data in an efficient way. While accessing a large volume of data over the network, task scheduling plays a major concern to minimize the risk level. Various methods are preferred for the job scheduling. But, finding the best scheduling method is a significant challenge to improve scheduling efficiency and minimizes the traffic level in big data analytics. Multivariate Logistic Regression based Gradient Descent firefly optimized Round Robin Scheduling (MLR-GDFORRS) technique is introduced for scheduling the number of task (i.e. user request) to optimal virtual machine with minimum time. MLR-GDFORRS technique also minimizes the workload across the cloud server while handling the number of tasks. Initially, the number of tasks arrives at the cloud server from different locations. After collecting the tasks, cloud manager analyzes the tasks using Multivariate Logistic Regression Analysis. Multivariate Logistic Regression is the statistical process for analyzing the tasks and to find relationship among dependent data (i.e., priority level) and one or more independent data of same cloud user request (e.g., request arrival time, file size, predicted job completion time). After that, the tasks are stored in one or more queue based on priority level with lesser memory consumption. Followed by, the tasks get scheduled using Gradient Descent firefly optimized Round Robin Scheduling algorithm. The GDFORRS efficiently finds the resource optimized virtual machine for allocating the tasks in a circular manner. The MapReduce function assigns the high priority tasks to the optimal virtual machine. This helps to minimize the workload of the cloud server.

Keywords--- Big Data, Task Scheduling, Multivariate Logistic Regression Analysis, Round Robin Scheduling Algorithm, MapReduce Function.

I. Introduction

Big data analytics is a procedure of analyzing a large number of data and valuable information**.** Big Data plays a very significant role in several applications like healthcare, automobiles, information technology and so on. Effective utilization of a resource, time becomes a demanding task in big data analytics. Big data analytics is performed by machine learning or statistical algorithms to process the data and detect significant information out of it.

In big data analytics, tasks are distributed across several virtual machines to lessen job completion time as well as traffic level. Therefore, an appropriate task scheduling is exploited for achieving a better service provisioning in big data analytics.

The several data mining models has been developed in the recent day to perform task scheduling with big data. Multivariate Logistic Regression based Gradient Descent firefly optimized Round Robin Scheduling (MLR-GDFORRS) technique is introduced in this paper.

The main contribution of the MLR-GDFORRS technique is summarized as follows,

- The contributions of MLR-GDFORRS technique are to improve task scheduling efficiency and minimize the time and memory consumption. This contribution is achieved by applying Multivariate Logistic Regression. The regression function analysis the incoming tasks with the certain scheduling parameters. Based on regression analysis, the tasks are prioritized and stored in different queues.
- Gradient Descent firefly optimized Round Robin Scheduling is applied to find the resource optimized virtual machine among the number of the virtual machines based on light intensity. The virtual machine which utilizes the minimum resource is chosen for handling high priority task. This helps to minimize the data

Received: 5 Jan 2019/Accepted: 15 Feb 2019

corruption and probability of data loss. The Map reduce function is applied to schedule the high priority task to an optimal virtual machine with minimum scheduling time. This assists to improve the scheduling efficiency.

This paper is ordered as follows. Section 2 provides an outline of related research works. Section 3 provides the description of MLR-GDFORRS technique with neat diagram. Experimental settings of MLR-GDFORRS technique and state-of-art methods are described in section 4.

Section 5 affords experimental results and discussion with certain parameters. Finally, the conclusion of the paper is presented in section 6.

II. Related Work

A new task scheduling framework called Flutter was introduced in [1] for lessen completion time and network costs of big data processing.

The framework failed to investigate the task scheduling jointly optimized in the context of wide-area big data processing with optimal virtual machines. In [2], a hybrid evolutionary algorithm was developed for defeat Task Scheduling and Data Assignment Problem (HEA-TaSDAP). It does not consider the delay related scheduling since it failed to analyze the tasks for handling the data replication problem.

An Adaptive Cost-based Task Scheduling (ACTS) was introduced in [3] to offer a data access to the virtual machines (VMs) with lesser time and cost.

The method does not solve the load-balancing issues. A fine-grained resource-aware MapReduce scheduler that partitions tasks into phases (PRISM) was designed in [4]. The MapReduce considerably minimizes the running time of data-intensive jobs. The PRISM does not improve the scalability. A new task scheduling algorithm was designed in [5] for improving the heterogeneous task scheduling. The algorithm consumes more time complexity while scheduling the heterogeneous task.

An energy-aware fair scheduling framework was introduced in [6] for big data applications. The framework failed to effectively minimize the workload of data cluster. An Automatic MapReduce Evaluation Tool was designed in [7] for allocating the Big Data Workloads. The tool does not adopt the MapReduce frameworks to maintain additional resources.

A Hadoop Distributed File System (HDFS) was introduced in [8] for processing the number of data and job scheduling to accomplish high performance in big data processing. The performance of job scheduling was not improved. A reliability-aware task scheduling algorithm was introduced in [9] depends on replication on heterogeneous computing systems.

The algorithm failed to analysis the incoming tasks for improving the scheduling efficiency. Discrete Particle Swarm Optimization Algorithm was designed in [10] to defeat multi-level job Scheduling issue in single machine under execution time uncertainty. The algorithm does not robust for scheduling problem with more uncertainty factors, such as machine interruption and delays during their operations. Multi-objective algorithm was designed in [11] to optimize MapReduce job scheduling depends on the completion time as well as the cost of cloud service models. The algorithm does not prioritize job to improve the scheduling efficiency. Dynamic multitasking workload scheduling technique [12] was presented for big data analytics in the elastic cloud. The technique does not enhance the performance of scheduling the workload.

A temporal task scheduling algorithm (TTSA) was designed in [13] for efficiently transmitting the incoming tasks to private cloud data center and public clouds. The algorithm failed to minimize dispatch time and execution delays in hybrid clouds. A Modified Best Fit with Capacity Based Scheduling (MBFCBS) technique was introduced in [14] to offer cloud resources to users through virtual machines. The technique does not consider a task's deadline to improve the scheduling efficiency.

In [15], a Min-Min and Max-Min algorithms were designed for efficient task scheduling in a cloud environment. The algorithm failed to lessen the execution time of tasks in cloud computing with dynamic task allocations**.** An adaptive scheduling technique was developed in [16] to increase the efficiency of Hadoop systems that process heterogeneous. MapReduce jobs. The technique does not find the best virtual machine to process the tasks.

A Hybrid Genetic algorithm and particle swarm optimization (GA-PSO) algorithm was designed in [17] to assign tasks to the resources effectively. The workflow tasks were distributed to the available VMs does not depend on the tasks size and the speed of VM.

Received: 5 Jan 2019/Accepted: 15 Feb 2019

A genetic algorithm-based job scheduling approach was designed in [18] for improving the efficiency of big data analytics. However, an accurate estimation of big data analytics was not performed. In [19], Heuristic algorithms were designed to schedule the autonomous tasks in cloud computing. The scheduling efficiency of the algorithm was not improved. Asymptotic scheduling technique was introduced in [20] for several task processing in Big Data platforms. The technique failed to use MapReduce function for minimizing the scheduling time.

The certain issues are identified from the above-said literature such as lack of task scheduling efficiency, more scheduling time and memory consumption, failed to find the optimal virtual machine and so on. In order to address the issues from the existing methods, Multivariate Logistic Regression based Gradient Descent firefly optimized Round Robin Scheduling (MLR-GDFORRS) technique is introduced in big data analytics.

III. Methodology

Multivariate Logistic Regression based Gradient Descent Firefly Optimized Round Robin Scheduling

With the rapid growth of data, the processing and analyzing a large volume of data is the significant process. While processing a large volume of data, scheduling is the important process for minimizing computation time of a job. In general, the various scheduling techniques are employed for processing tasks. In this work, machine learning technique is developed for improving the task scheduling efficiency with less complexity. Multivariate Logistic Regression based Gradient Descent firefly optimized Round Robin Scheduling technique is introduced. Scheduling is the process of assigning the user requested tasks to available resources on the basis of the regression analysis. This is done by multivariate logistic regression.

After the regression analysis, the tasks are allocated to the available virtual machines. This helps to minimize the workload and traffic while accessing the large volume of data. The scheduling process is done by applying a weighted round robin scheduling.

The weighted round-robin scheduling is improved than ordinary round-robin scheduling, when the processing capacity of virtual machines is different. The MapReduce function is also applied in MLR-GDFORRS technique to improve the scheduling efficiency. The architecture diagram of MLR-GDFORRS technique is shown in below Figure 1.

In cloud, the users' sends a request (i.e. tasks) to a cloud server. The user requested tasks are considered as jobs (i.e. file). The cloud manager analyzes the incoming tasks from the user to find the priority level for scheduling. The cloud manager uses the multivariate logistic regression to analyze the incoming user requested tasks. After that, the priority level of each task is identified and it is stored in a queue.

Then, the gradient descent firefly optimized round robin scheduling is applied to allocate the tasks to the resource optimized virtual machine.

As a result of task scheduling, the workload of cloud server and the traffic level gets minimized. The MapReduce function also minimizes the space complexity

Let us consider the number of tasks $T_1, T_2, T_3, \ldots . T_n$ (i.e. user request) and cloud manager CM. The CM analysis the incoming tasks and schedules to different virtual machines $VM_1, VM_2, VM_3, \ldots . VM_n$. Based on the above system model, the task scheduling is performed in the following sections.

Multivariate Logistic Regression Analysis

The Multivariate Logistic Regression is a machine learning method for analyzing the incoming tasks to find relationship among dependent data and one or more independent data. The MLR-GDFORRS technique performs the analysis with one or more independent data. Hence the name is called Multivariate Logistic Regression. Here the dependent data is represented as a priority level. The MLR-GDFORRS uses a different queue for storing the user request with their priority.

The independent data are a certain job request parameters are task size, request arrival time, predicted job completion time (i.e. ending time).

The regression analysis helps to identify how the dependent variable alters when any one of the independent variables gets varied. It means that if any changes in the job request parameters value, the priority level gets changed. Through regression analysis, the cloud manager schedules the tasks to the virtual machine.

Received: 5 Jan 2019/Accepted: 15 Feb 2019

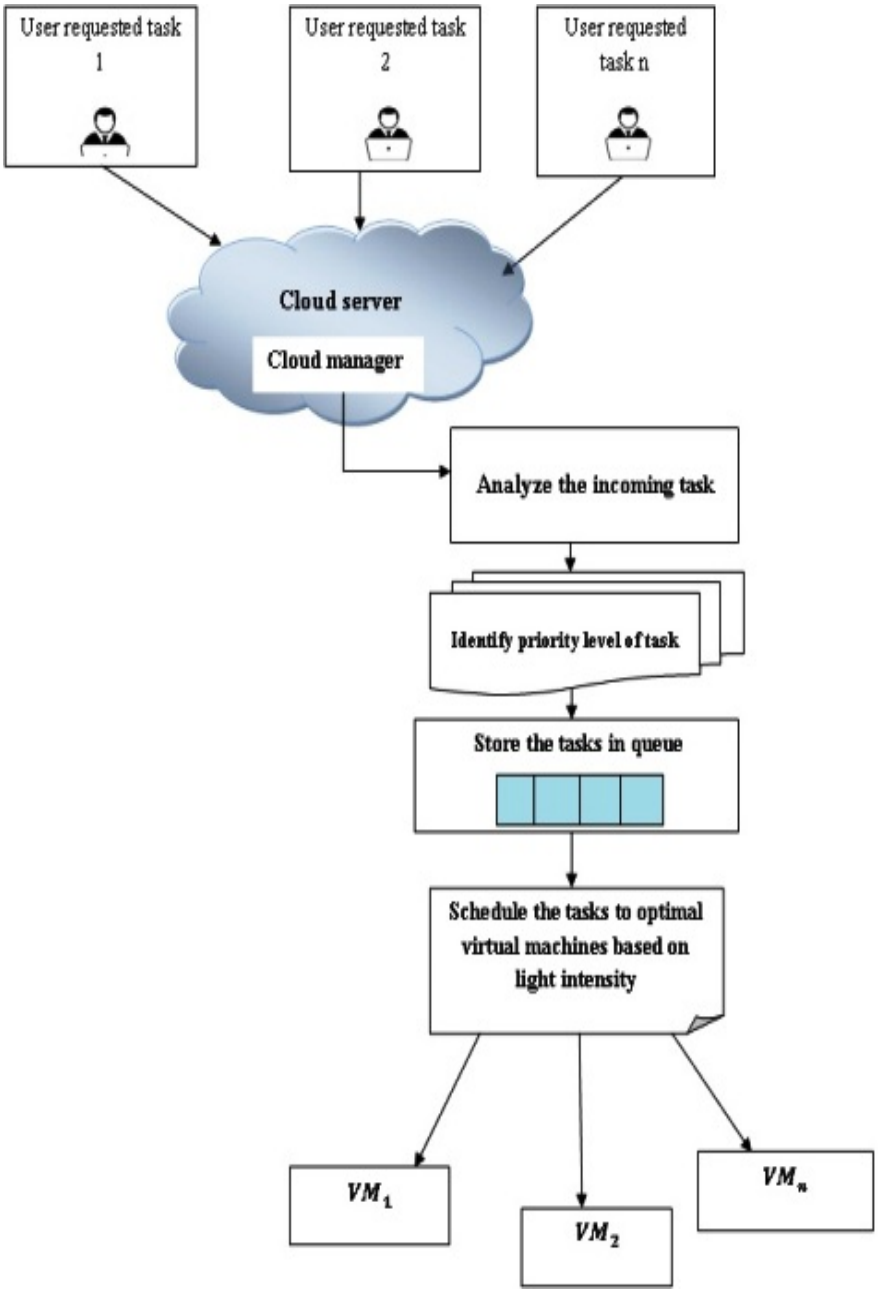

Figure 1: Architecture of MLR-GDFORRS Technique

The tasks stores in different priority level as output i.e. $\{1,2,3,\ldots.k\}$ based on, the user request priority level is determined. The regression function is mathematically expressed as follows,

$$Y = \frac{e^{\alpha_i . P_j}}{1+\sum_{i=1}^{n} e^{\alpha_i . T_i}} \quad (1)$$

From (1), Y represents an output of logistic regression, α_i denotes regression coefficients, P_j denotes a priority level of the queue, T_i denotes an incoming task.

The regression function identifies the priority level which has different possible outcomes and their predicted probability. Then the predicted probability is described as follows,

Received: 5 Jan 2019/Accepted: 15 Feb 2019

$$Y = \begin{bmatrix} p(Y=1|T_1) \\ p(Y=2|T_2) \\ . \\ . \\ . \\ p(Y=k|T_3) \end{bmatrix} = \frac{1}{1+\sum_{i=1}^{n} e^{\alpha_i \cdot T_i}} \begin{bmatrix} e^{\alpha_1 \cdot P_1} \\ e^{\alpha_2 \cdot P_2} \\ . \\ . \\ . \\ e^{\alpha_n \cdot P_k} \end{bmatrix} \quad (2)$$

From the above equation (2),p denotes a probability of output class. Based on the exponential value of tasks and their priority level, the user priority level is identified. After identifying a priority level, the tasks are stored in one or more queue$(Q1, Q2, Q3 \ldots Qn)$. $Q1$ is the highest priority with large task and the maximum time consumption. The low priority tasks are stored in the next level of the priority queue.

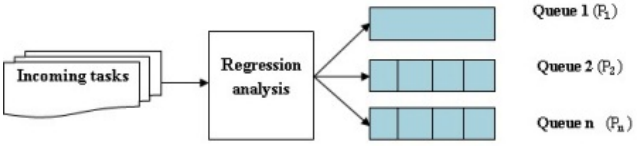

Figure 2: Multivariate Logistic Regression based Priority Level Assignments

As shown in figure 2, regression-based priority level assignments are illustrated. Based on the analysis, the highest priority tasks are assigned in the first queue. Followed by, all the requested tasks are stored in a next priority queue.

Gradient Descent Firefly Optimized Round Robin Task Scheduling Approach

After storing the number of tasks in a queue, the scheduling is performed using gradient descent firefly optimized round robin scheduling approach. This approach is the simplest method for distributing the user requests to optimal virtual machines in a circular order. In MLR-GDFORRS, an optimal virtual machine is chosen based on predicted job completion time, bandwidth utilization and memory utilization. This helps to minimize the workload across the server as well as traffic level. The scheduling is performed to handle the number of virtual machines with different processing capacities. The scheduling process is illustrated in figure 3. It illustrates a gradient descent firefly optimized round robin task scheduling with MapReduce function. The below figure clearly illustrates an efficient tasks scheduling to several virtual machines with minimum time. Let us consider the number of tasks in the different queues $Q1, Q2, Q3 \ldots Qn$. Then the high priority user tasks are scheduled first rather than the low priority tasks. The Map Reduce function assigns the higher priority tasks to the optimal virtual machine. GDFORRS approach uses the MapReduce function which is a simple and powerful programming model which has been widely used for processing the large number of tasks resulting in minimizes the traffic as well as delay.

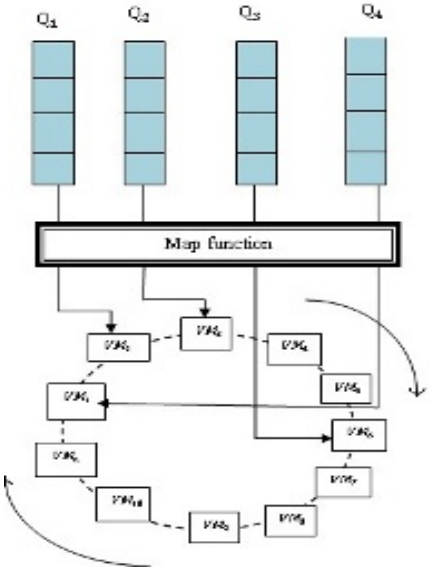

Figure 3: Gradient Descent Firefly Optimized Round Robin Task Scheduling

Received: 5 Jan 2019/Accepted: 15 Feb 2019

Mapreduce process divided into two phases namely Map phase and Reduce phase. Map phase executes tasks scheduling to the optimal virtual machine run on the input tasks in a circular way. Reduce phase executes after Map phase, and reducing the number of incoming tasks. This also helps for minimizing the workload of the server as well as traffic level.

The MapReduce takes the high priority tasks first and allocates it to the optimal virtual machine which has a higher weight. The GDFORRS identifies the virtual machine which has enough resources to process the tasks. The optimal virtual machine is detected through gradient descent firefly optimization. Let us consider the number of tasks $T_1, T_2, T_3, \ldots . T_n$ in the priority queues $Q1, Q2, Q3 \ldots Qn$. The fireflies are considered a number of virtual machines '$VM_1, VM_2, \ldots VM_n$'. Among the several virtual machines, an optimal virtual machine is identified for assigning the tasks based on the resource utilization. Here, three types of resources are considered namely predicted job completion time, bandwidth and memory. Based on this resource utilization, an optimal virtual machine is chosen. The resources for each virtual machine are calculated as follows. The predicted job completion time of the virtual machine is calculated as follows,

$$JCT = T_s - T_E \quad (3)$$

From (3),JCT denotes a predicted job completion time of the virtual machine, T_s represents a starting time to process the task and T_E denotes an ending time of that particular task. Bandwidth utilization of the virtual machine is defined as the difference between the available bandwidth and unused bandwidth. It is calculated as follows,

$$BW_{vm} = BW_a - BW_u \quad (4)$$

From (4), BW_{vm} denotes a bandwidth utilization of virtual machine, BW_a represents an available bandwidth, BW_u denotes an unused bandwidth. The memory is the other parameter in task scheduling processes. It is defined as an amount of storage space required by the virtual machine to process the user tasks. The memory utilization is calculated as follows,

$$MU_{vm} = T_s - U_s \quad (5)$$

From (5),MU_{vm} denotes a memory utilization of virtual machine and T_s denotes a total space and U_s denotes an unused space. Based on the above said parameter calculation, the cloud manager detects an optimal virtual machine.

In GDFORRS technique, the group of fireflies moves towards the other fireflies by flashing light intensity with the objective function. In general, the objective function is a function that it is desired to maximize or minimize. The objective function in GDFORRS technique is to find an optimal virtual machine with minimum resource utilization for the particular task. To discover minimum of a function, GDFORRS technique uses gradient descent approach. The objective function with the gradient descent is expressed as follows,

$$f(x) = arg\ min\{JCT, BW_{vm}, MU_{vm}\} \quad (6)$$

From (6),$f(x)$ denotes an objective function, $arg\ min$ represent a gradient descent function to select the virtual machine with minimum resource utilization for task scheduling. The initial populations of fireflies (i.e. number of virtual machines) are created as follows,

$$VM = VM_1, VM_2, \ldots VM_n \quad (7)$$

From (7),VM denotes a virtual machine in the cloud server. The light intensity of all fireflies is formulated based on the three resources as follows,

$$I\ (VM) = \{JCT, BW_{vm}, MU_{vm}\} \quad (8)$$

From (8),$I\ (VM)$ denotes a intensity of virtual machine based on predicted job completion time JCT, BW_{vm}denotes a bandwidth utilization of virtual machine, MU_{vm} represents a memory utilization of virtual machine. Based on the calculated light intensity of each virtual machine, optimal virtual machine is selected. Based on the objective function, an optimal virtual machine is selected from any pair of fireflies which is expressed as follows,

Received: 5 Jan 2019/Accepted: 15 Feb 2019

$$CM \rightarrow I(VM_i^t) + exp\,[-\beta]\big(I(VM_i^t) - I(VM_j^t)\big) + \delta_t \epsilon_t \quad (9)$$

From (9),CM denotes a cloud manager, $I(VM_i^t)$ denotes a current intensity of virtual machine, β denotes a light absorption coefficient, $I(VM_i^t)$ denotes a current intensity of VM_i and $I(VM_j^t)$ represents a current intensity of VM_j. δ_t denotes a randomization parameter between 0 and 1, ϵ_t represents a vector drawn from a Gaussian or other distribution. Based on the light intensity, the weight is assigned to each virtual machine in cloud server.

$$CM \rightarrow \omega_t\,(VM_1, VM_2, VM_3, \ldots . VM_n) \quad (10)$$

From (10),CM denotes a cloud manager, ω_t denotes a weight assigned to virtual machine in each round based on the light intensity. Based on the weight value, the virtual machines are ranked in an ascending order. The high weighted virtual machine has first rank than the less weighted virtual machine which is expressed as follows,

$$r \rightarrow VM_1, VM_2, VM_3, \ldots . VM_n \quad (11)$$

From (11),r denotes a rank assigned to each virtual machine VM. The CM finds the optimal virtual machine which has a high rank to process the first priority tasks for each round. The Map phase performs the tasks scheduling to the optimal virtual machine.

It means that the virtual machine takes minimum job completion time and utilizes the memory as well as bandwidth for processing the high priority tasks. This helps to minimize the probability of data loss during the scheduling process.

In this manner, the GDFORRS assigns jobs to the optimal virtual machines in round-robin manner. As shown in figure 3, there are 'n' virtual machines ($VM_1, VM_2, \ldots . , VM_n$), the cloud manager allocates a first request go to VM_2, then the second request go to $the\ VM_3$, the third request go to VM_4, and the fourth request go to VM_1then repeat in a round-robin manner.

As a result of scheduling, the reduce function lessens the number of incoming tasks in a cloud server which lessens the workload across the server as well as traffic level.

The algorithmic process of task scheduling is described as follows. As shown in below algorithm 1, Multivariate Logistic Regression based Gradient descent firefly optimized Round Robin scheduling is described. For each incoming task, the numbers of job request parameters are defined. Then the Multivariate Logistic Regression analysis is performed between the tasks and priority level.

Based on the regression analysis, the priority level of the each requested tasks are identified and it stored in the different priority queue.

Then, the Gradient descent firefly optimized Round Robin scheduling is applied to allocate the tasks to resource optimized virtual machine in a circular manner. The resource optimized virtual machine is selected using gradient descent firefly optimization. Initialize the number of virtual machines and define the objective function.

For each virtual machine, formulate the light intensity based on predicted job completion time, bandwidth utilization, memory utilization of virtual machine. After that, the weight is assigned to each virtual machine based on light intensity.

Then, the rank is assigned to each firefly based on weight value. As a result, the cloud manager identifies the optimal virtual machine and map function schedule the task to optimal virtual machine. This process minimizes the workload across the cloud server and improving load balancing in big data analytics.

Received: 5 Jan 2019/Accepted: 15 Feb 2019

Algorithm 1 Multivariate Logistic Regression based Gradient Descent Firefly Optimized Round Robin Scheduling

Input: No. of cloud user requests $T_1, T_2, T_3, \ldots . T_n$, virtual machines ($VM_1, VM_2, \ldots . , VM_n$)

Output: Improve scheduling efficiency

Begin

1. For each task T_i

2. Find the certain job request parameters

3. Perform regression analysis Y

4. Find the priority level $'P_i'$ of T_i

5. Store the T_i in a different queues

$Q1, Q2, Q3 \ldots Qn$ according to their P_i

\\ **Apply optimization technique**

6. Initialize the population of VM_i

7. **for** each highest priority T_i

8. for each virtual machine VM_i

9. Define the gradient descent objective function $f(x)$

10. Formulate light intensity $I\ (VM)$ based on predicted $JCT\ , BW_t, MU_{vm}$

11. Assign weight $'\omega_t'$ based on light intensity $I\ (VM)$

12. Rank the VM_i based on weight $'\omega_t'$

13. CM Identifies the VM_i with high rank $'r'$

14. Schedule T_i to optimal VM_i

15. end for

16. end for

17. end for

End

The above description shows that the proposed technique effectively scheduling the user requests to optimal virtual machine among the several virtual machines for minimizing the data loss. The above processes are implemented in the experimental evaluation to show the performance of proposed scheduling algorithm compared with existing methods.

IV. Experimental Evaluation

Experimental evaluation of proposed MLR-GDFORRS technique and existing methods Flutter framework [1] and HEA-TaSDAP [2] are implemented using Java language with CloudSim network simulator. The experimental evaluation is conducted using Personal Cloud Datasets. This dataset is taken from http://cloudspaces.eu/results/datasets. The dataset includes 17 attributes (i.e. columns) and it includes 66245 instances. The objective of this dataset is to perform load and transfer test. The number of 17 attributes are row id, account id, file size (i.e. task size), operation_time_start, operation_time_end, time zone, operation_id, operation type, bandwidth trace, node_ip, node_name, quoto_start, quoto_end, quoto_total (storage capacity), capped, failed and failure info. Among the 17 columns, two columns are not used such as time zone and capped. The above columns are considered for efficient task scheduling to the multiple virtual machines in the cloud. Experimental evaluation of MLR-GDFORRS technique is compared with the existing methods with different factors such as

scheduling efficiency, scheduling time, memory consumption and true positive rate. The results obtained from the experimental results are described in the next section.

Received: 5 Jan 2019/Accepted: 15 Feb 2019

V. Results and Discussion

In this section, experimental results and discussion of proposed MLR-GDFORRS technique and existing Flutter framework [1] and HEA-TaSDAP [2] are described with various performance metrics such as scheduling efficiency, scheduling time, memory consumption and true positive rate with the number of tasks (i.e. user requests). With the help of these parameters, the results are compared with the help of table and graphical representations.

Performance Analysis of Scheduling Efficiency

Scheduling Efficiency is defined as the ratio of a number of tasks that are scheduled to the virtual machine to the total number of tasks. The scheduling efficiency is calculated as follows,

$$SE = \frac{no.of\ tasks\ scheduled\ to\ VM}{n} * 100 \qquad (12)$$

From equation (12), SE denotes a scheduling efficiency and it is measured in terms of percentage (%). Here, 'n' indicates a number of user requested tasks.

Table 1: Tabulation for Scheduling Efficiency

No. of tasks	Scheduling efficiency (%)					
	No. of tasks correctly schedule	MLR-GDFORRS	No. of tasks correctly schedule	Flutter frame work	No. of tasks correctly schedule	HEA-TaSDAP
25	23	92	21	84	19	76
50	48	96	43	86	36	72
75	70	93	62	83	55	73
100	95	95	89	89	78	78
125	120	96	113	90	102	82
150	143	95	125	83	113	75
175	163	93	145	86	128	73
200	192	96	169	85	150	75
225	220	97	187	83	176	78
250	239	96	220	88	208	83

Table 1 clearly describes the performance results of scheduling efficiency versus a number of user-requested tasks with three different methods namely MLR-GDFORRS technique, Existing Flutter framework [1], and HEA-TaSDAP [2]. For the experimental consideration, the numbers of user-requested tasks are varied from 25 to 250. The above table values shows that the performance results of MLR-GDFORRS technique are improved than the existing methods.

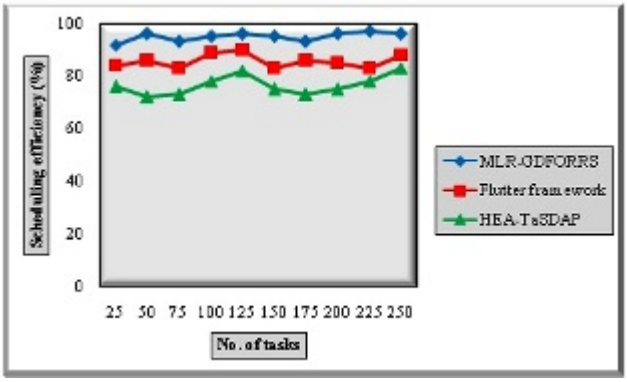

Figure 4: Performance Results of Scheduling Efficiency Versus Number of Tasks

Received: 5 Jan 2019/Accepted: 15 Feb 2019

Figure 4 depicts the performance results of scheduling efficiency with respect to a number of tasks. The numbers of user requests are considered as tasks which are taken as input for performing the scheduling in big data analytics. In the experimental consideration, the user requests are effectively analyzed for improving the performance of task scheduling. Based on the results observed, task scheduling efficiency is significantly improved using MLR-GDFORRS technique than the existing methods. This is because, the MLR-GDFORRS technique analysis the incoming tasks using multivariate logistic regression with the file size, completion time, arrival time. Based on the analysis, the priority levels of the tasks are identified. Then these tasks are stored in a queue. The high priority tasks are allocated first to virtual machines. The cloud manager identifies the virtual machine using gradient descent firefly optimized round robin scheduling approach. The optimal virtual machine is assigned based on the three scheduling parameters such as job completion time, bandwidth and memory utilization. The virtual machine with high intensity is selected and the map-reduce function assigns the high priority task to that virtual machine. This process is performed in a circular manner. As a result of scheduling, the workload and traffic level gets minimized. From the experimental results, scheduling efficiency of proposed MLR-GDFORRS technique is improved by 11% and 24% when compared to existing Flutter framework [1] and HEA-TaSDAP [2] respectively.

Performance Analysis of Scheduling Time

Scheduling time is defined as an amount of time required for scheduling the tasks to a virtual machine. The scheduling time is calculated using following mathematical formula,

$$ST = n * time\ (scheduling\ one\ task) \quad (13)$$

From (13), ST denotes a scheduling time, n denotes a total number of user requested tasks. The scheduling time is measured in terms of milliseconds (ms). The below table 2 clearly describes the task scheduling time with different methods MLR-GDFORRS technique, Flutter framework [1] and HEA-TaSDAP [2]. The below table shows that the scheduling time is varied according to the number of tasks taken for the experimental evaluation. Totally, ten different runs are carried out with three different methods. For each run, Number of tasks is varied from 25 to 250.

Table 2: Tabulation for Scheduling Time

No. of tasks	Scheduling time (ms)					
	Time to schedule one task (ms)	MLR-GDFORRS	Time to schedule one task (ms)	Flutter framework	Time to schedule one task (ms)	HEA-TaSDAP
25	0.9	23	1.2	30	1.5	38
50	0.5	25	0.7	35	0.8	40
75	0.35	26	0.45	34	0.49	37
100	0.29	29	0.36	36	0.43	43
125	0.26	33	0.34	43	0.39	49
150	0.26	39	0.3	45	0.33	50
175	0.23	40	0.26	46	0.29	51
200	0.22	44	0.258	52	0.286	57
225	0.2	45	0.24	54	0.26	59
250	0.22	55	0.25	63	0.27	68

The experimental results of the scheduling time are depicted in the following graphical representations

Received: 5 Jan 2019/Accepted: 15 Feb 2019

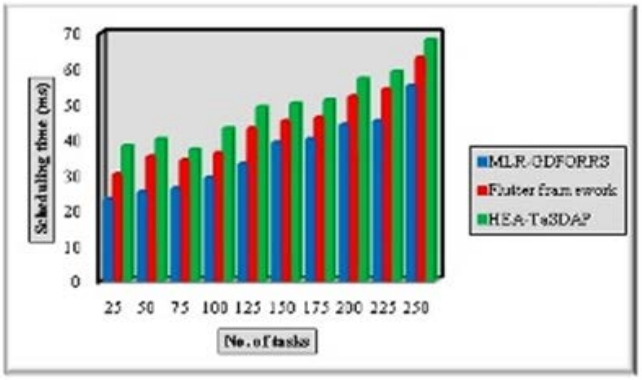

Figure 5: Performance Results of Scheduling Time Versus Number of Tasks

Performance results of scheduling time versus a number of tasks are illustrated in figure 5. The above figure clearly shows that the two-dimensional representation of scheduling time versus a number of tasks with three different methods MLR-GDFORRS technique, Flutter framework [1] and HEA-TaSDAP [2]. The performance results of three different methods are represented in three various colors. From the figure 5, the experimental result of scheduling time is minimized using MLR-GDFORRS technique when compared to existing methods. This is because, the cloud manager performs regression analysis for each incoming tasks with the scheduling parameters such as file size, arrival time and completion time. The multivariate logistic regression calculates the exponential values of each user requests and their priority level (i.e. 1, 2, 3...). Based on the analysis, the cloud manager identifies the priority level of incoming tasks. The task with high priority is processed first rather than the low priority tasks. Then the cloud manager arranges the tasks in the queue according to their priority level. Then the MapReduce function is applied to handle a large number of tasks. In GDFORRS technique, the map function assigns the high priority tasks to optimal virtual machines. This process takes minimum time. Based on observation, the experimental results of scheduling time of MLR-GDFORRS technique is minimized by 19% when compared to Flutter framework [1]. Similarly, the scheduling time of MLR-GDFORRS technique is also minimized by 28% when compared to other existing HEA-TaSDAP [2].

Performance Analysis of Memory Consumption

Memory consumption is defined as an amount of storage space required for storing the number of user-requested tasks. The formula for calculating the memory consumption is expressed as follows,

$$MC = n * \text{memory (Storing one user request)} \quad (14)$$

From (14),MC denotes memory consumption, $'n'$ denotes a number of user requested tasks. Memory consumption is measured in terms of mega bytes (MB).

Table 3: Tabulation for Memory Consumption

No. of tasks	Memory consumption (MB)					
	Memory for storing one task (MB)	MLR-GDFORRS	Memory for storing one task (MB)	Flutter framework	Memory for storing one task (MB)	HEA-TaSDAP
25	0.9	23	1.2	30	1.4	35
50	0.6	30	0.7	35	0.9	45
75	0.45	34	0.59	44	0.65	49
100	0.41	41	0.48	48	0.54	54
125	0.36	45	0.43	54	0.51	64
150	0.32	48	0.37	56	0.42	63
175	0.28	49	0.34	60	0.39	68
200	0.27	54	0.32	64	0.35	70
225	0.26	59	0.3	68	0.33	74
250	0.25	63	0.28	70	0.31	78

Received: 5 Jan 2019/Accepted: 15 Feb 2019

The experimental results of memory consumption with respect to a number of tasks are described in table 3. The table value clearly describes the storage capacity of the user requested tasks. From the calculated results, the memory consumption of MLR-GDFORRS technique is minimized when compared to existing Flutter framework [1] and HEA-TaSDAP [2]. The performance results of memory consumption with three different methods are shown in figure 6.

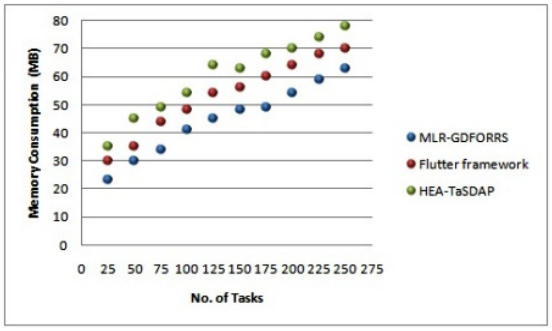

Figure 6: Performance results of memory consumption versus number of tasks

Figure 6 depicts the performance analysis of memory consumption with respect to a number of tasks. The memory consumption for storing the number of tasks is calculated with three different methods. As shown in the figure, the proposed MLR-GDFORRS technique consumes less storage space for storing the user tasks in the priority queue. Initially, the regression analysis is carried out with the number of tasks to identify the priority level. After that, the tasks are stored in the priority queue for scheduling. This process reduces the memory consumption. Moreover, by applying the map-reduce function in MLR-GDFORRS technique, the incoming tasks are mapped into a different virtual machine which utilizes the less memory space for processing user tasks. This process takes minimum storage space as well as less complexity while handling a large number of user tasks. Let us taken as 25 tasks for conducting experimental, the MLR-GDFORRS technique obtains 23MB of memory consumption whereas the existing methods namely flutter framework [1] and HEA-TaSDAP [2] consumes 30MB and 35MB respectively. Followed by, the remaining nine runs are carried out and compare the results of proposed and existing methods. The comparison results show that the MLR-GDFORRS technique significantly reduces the memory consumption by 16% and 27% than the existing Flutter framework [1] and HEA-TaSDAP [2] respectively.

Performance Analysis of True Positive Rate

The true positive rate is calculated as ratios of number of tasks are correctly scheduled to the optimal virtual machine to the total number of tasks. It is calculated using following mathematical formula,

$$TPR = \frac{no.\ of\ tasks\ correctly\ scheduled\ to\ optimal\ VM}{n} * 100 \quad (15)$$

From (15),TPR denotes a true positive rate, ‘n’ denotes a number of tasks. True positive rate is measured in terms of percentage (%).

Table 4: Tabulation for True Positive Rate

No. of tasks	True positive rate (%)					
	No. of tasks correctly schedule	MLR-GDFORRS	No. of tasks correctly schedule	Flutter framework	No. of tasks correctly scheduled	HEA-TaSDAP
25	22	88	19	76	15	60
50	46	92	40	80	32	64
75	67	89	59	79	51	68
100	93	93	83	83	73	73
125	119	95	109	87	98	78
150	141	94	121	81	103	69
175	161	92	140	80	123	70
200	190	95	163	82	142	71
225	212	94	178	79	162	72
250	229	92	215	86	202	81

Received: 5 Jan 2019/Accepted: 15 Feb 2019

Table 4 describes a true positive rate with respect to a number of tasks. The number of tasks is taken as input for scheduling processes. From the results, the proposed MLR-GDFORRS correctly schedules the user tasks to the optimal virtual machine when compared to existing methods. This result of MLR-GDFORRS technique improves the performance results of true positive rate. Experimental results of the true positive rate with three different methods are illustrated using following graphical representations.

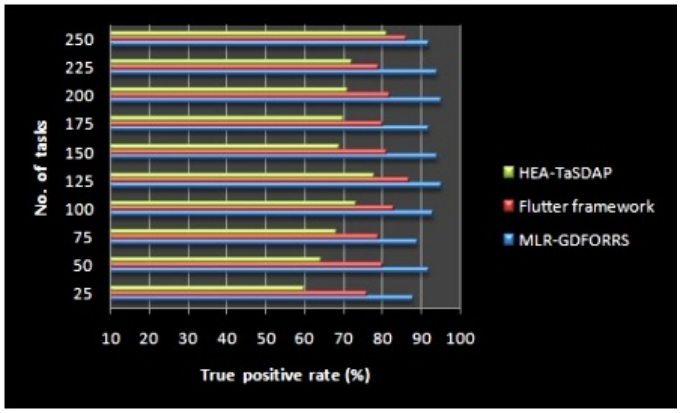

Figure 7: Performance Results of True Positive Rate Versus Number of Tasks

Figure 7 illustrates experimental results of the true positive rate with respect to a number of tasks. The above figure illustrates the proposed MLR-GDFORRS technique effectively identifies the best optimal virtual machine for assigning the incoming tasks. This is because, MLR-GDFORRS technique uses the gradient descent firefly optimized round robin scheduling approach. In this approach, the numbers of virtual machines are initialized and define the objective function. Then the light intensity of each virtual machine is calculated based on predicated job completion time, bandwidth utilization and memory utilization. After that, the light intensity of the entire firefly is analyzed and assigning the weights. Then the virtual machines are ranked based on weight value. Then the cloud manager finds an optimal virtual machine which has high rank. This virtual machine is selected as optimal one for processing the high priority task. The weighted round robin scheduling selects the best optimal virtual machine for task execution to avoid the delay as well as the probability of data loss in traffic aware of big data analytics. From the simulation results, the true positive rate of MLR-GDFORRS technique is considerably improved by 14% and 32% when compared to existing Flutter framework [1] and HEA-TaSDAP [2] respectively.

The above discussion clearly shows that the MLR-GDFORRS technique improves the task scheduling efficiency and minimizes the workload across the cloud server with minimum time as well as less resource utilization in data analytics.

VI. Conclusion

An efficient technique called, Multivariate Logistic Regression based Gradient descent firefly optimized Round Robin Scheduling (MLR-GDFORRS) is developed for improving the task scheduling efficiency and minimizing the workload of the cloud server. The scheduling is carried out based on the regression analysis. The regression is used for finding the relationship between the job scheduling parameters and the priority level of tasks. From the analysis, the tasks are prioritized and it stored in different queues. Then the cloud manager finds the resource optimized virtual machine through the gradient descent firefly optimization to process the high priority tasks first among the entire virtual machine in a cloud server for improving the scheduling efficiency. Then the MapReduce function assigns the tasks to the best optimal virtual machine in a circular manner. This helps to minimize the data loss and delay while handling a large number of tasks. Experimental evaluation of MLR-GDFORRS technique and existing methods are carried out using Personal Cloud Dataset for scheduling the user requested tasks in big data analytics. The experimental result shows that the MLR-GDFORRS technique improves the task scheduling efficiency with minimum time as well as minimizes the memory consumption.

References

[1] Hu, Z., Li, B. and Luo, J. Time-and cost-efficient task scheduling across geo-distributed data centers. *IEEE Transactions on Parallel and Distributed Systems* **29** (3) (2018) 705-718.

Received: 5 Jan 2019/Accepted: 15 Feb 2019

[2] Teylo, L., De Paula, U., Frota, Y., De Oliveira, D. and Drummond, L. M. A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *Future Generation Computer Systems* **76** (2017) 1-17.

[3] Mosleh, M. A., Radhamani, G., Hazber, M. A. and Hasan, S. H. Adaptive Cost-Based Task Scheduling in Cloud Environment. *Scientific Programming*, 2016, 1-9.

[4] Zhang, Q., Zhani, M. F., Yang, Y., Boutaba, R. and Wong, B. PRISM: fine-grained resource-aware scheduling for MapReduce. *IEEE Transactions on Cloud Computing* **3** (2) (2015) 182-194.

[5] Guan, W., Yuxin, W., Hui, L. and He, G. HSIP: A Novel Task Scheduling Algorithm for Heterogeneous Computing. *Scientific Programming, Hindawi Publishing Corporation*, 2016, 1-11.

[6] Shao, Y., Li, C., Gu, J., Zhang, J. and Luo, Y. Efficient jobs scheduling approach for big data applications. *Computers & Industrial Engineering* **117** (2018) 249-261.

[7] Veiga, J., Expósito, R. R., Taboada, G. L. and Tourino, J. MREv: an automatic MapReduce Evaluation tool for Big Data workloads. *Procedia Computer Science* **51** (2015) 80-89.

[8] Usama, M., Liu, M. and Chen, M. Job schedulers for big data processing in Hadoop environment: testing real-life schedulers using benchmark programs. *Digital Communications and Networks* **3** (4) (2017) 260-273.

[9] Wang, S., Li, K., Mei, J., Xiao, G. and Li, K. A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems. *Journal of Grid Computing* **15** (1) (2017) 23-39.

[10] Joo, B. J., Shim, S. O., Chua, T. J. and Cai, T. X. Multi-level job scheduling under processing time uncertainty. *Computers & Industrial Engineering* **120** (2018) 480-487.

[11] Hashem, I. A. T., Anuar, N. B., Marjani, M., Gani, A., Sangaiah, A. K. and Sakariyah, A. K. Multi-objective scheduling of MapReduce jobs in big data processing. *Multimedia Tools and Applications* **77** (8) (2018) 9979-9994.

[12] Zhang, F., Cao, J., Tan, W., Khan, S. U., Li, K. and Zomaya, A. Y. Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud. *IEEE Transactions on Emerging Topics in Computing* **2** (3) (2014) 338-351.

[13] Yuan, H., Bi, J., Tan, W., Zhou, M., Li, B. H. and Li, J. TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE transactions on cybernetics* **47** (11) (2017) 3658-3668.

[14] Loganathan, S. and Mukherjee, S. Job scheduling with efficient resource monitoring in cloud datacenter. *The Scientific World Journal*, 2015, 1-11.

[15] Kumar, S. and Mishra, A. Application of Min-Min and Max-Min Algorithm for Task Scheduling in Cloud Environment Under Time Shared and Space Shared VM Models. *International Journal of Computing Academic Research (IJCAR)* **4** (6) (2015) 182-190.

[16] Yao, Y., Tai, J., Sheng, B. and Mi, N. LsPS: A job size-based scheduler for efficient task assignments in Hadoop. *IEEE Transactions on Cloud Computing* **3** (4) (2015) 411-424.

[17] Manasrah, A. M. and Ba Ali, H. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. *Wireless Communications and Mobile Computing*, 2018, 1-16.

[18] Lu, Q., Li, S., Zhang, W. and Zhang, L. A genetic algorithm-based job scheduling model for big data analytics. *EURASIP journal on wireless communications and networking* **2016** (1) (2016) 1-9.

[19] Madni, S. H. H., Latiff, M. S. A., Abdullahi, M. and Usman, M. J. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. *PloS one* **12** (5) (2017) 1-26.

[20] Sfrent, A. and Pop, F. Asymptotic scheduling for many task computing in big data platforms. *Information Sciences* **319** (2015) 71-91.

Received: 5 Jan 2019/Accepted: 15 Feb 2019