International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT21714

Load Balancing Using SJF-MMBF and SJF-ELM Approach

S. Rekha*[1], Dr. C. Kalaiselvi[2]
*[1]Ph. D. Research Scholar, Department of Computer Science, Tirupur Kumaran College for Women, Tiruppur
Assistant Professor, Department of IT, Dr. N. G. P. Arts and Science College, Coimbatore, Tamil Nadu, India
[2] Head and Associate Professor, Department of Computer Applications, Tirupur Kumaran College for Women, Tiruppur, Tamil Nadu, India

Article Info
Volume 7, Issue 1
Page Number: 74-86
Publication Issue :
January-February-2021

ABSTRACT

This paper studies the delay-optimal virtual machine (VM) scheduling problem in cloud computing systems, which have a constant amount of infrastructure resources such as CPU, memory and storage in the resource pool. The cloud computing system provides VMs as services to users. Cloud users request various types of VMs randomly over time and the requested VM-hosting durations vary vastly. A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority or process type. Each queue has its own scheduling algorithm. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. Multi-level queue scheduling is performed via the use of the Particle Swarm Optimization algorithm (MQPSO). It checks both Shortest-Job-First (SJF) buffering and Min-Min Best Fit (MMBF) scheduling algorithms, i.e., SJF-MMBF, is proposed to determine the solutions. Another scheme that combines the SJF buffering and Extreme Learning Machine (ELM)-based scheduling algorithms, i.e., SJF- ELM, is further proposed to avoid the potential of job starvation in SJF-MMBF. In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. The simulation results also illustrate that SJF- ELM is optimal in a heavy-loaded and highly dynamic environment and it is efficient in provisioning the average job hosting rate.

Keywords: Delay-optimal virtual machine, scheduling algorithm, Shortest-Job-First, Min-Min Best Fit, Multi-level queue scheduling, VM-hosting durations and Particle Swarm Optimization.

Article History
Accepted : 10 Jan 2021
Published : 20 Jan 2021

I. INTRODUCTION

Cloud computing is a model enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. In Cloud Computing the term Cloud is

used for the service provider, which holds all types of resources for storage, computing etc[1,2].

Mainly three types of services are provided by the cloud. First is Infrastructure as a Service (IAAS), which provides cloud users the infrastructure for various purposes like the storage system and computation resources. Second is Platform as a Service (PAAS), which provides the platform to the clients so that they can make their applications on this platform. Third is Software as a Service (SAAS), which provides the software to the users; so users don't need to install the software on their own machines and they can use the software directly from the cloud[3,4].

Due to the wide range of facilities provided by the cloud computing, the Cloud Computing is becoming the need of the IT industries. The services of the Cloud are provided through the Internet. The devices that want to access the services of the Cloud should have the Internet accessing capability. Devices need to have very less memory, a very light operating system and browser. Cloud Computing provides many benefits: it results in cost savings because there is no need of initial installation of much resource; it provides scalability and flexibility, the users can increase or decrease the number of services as per requirement; maintenance cost is very less because all the resources are managed by the Cloud providers [5,6].

In cloud computing environment, scheduling tasks to the Virtual Machines (VMs) in accordance with adaptable time, which involves finding out a proper sequence in which tasks can be executed under transaction logic constraints. The job scheduling of cloud computing is a challenge. In existing work formulate the VM scheduling in such a queueing cloud computing system as a decision-making process, where the decision variable is the vector of VM configurations and the optimization objective is the delay performance in terms of average job completion time [7].

A low-complexity online scheme that combines the Shortest-Job-First (SJF) buffering and Min-Min Best Fit (MMBF) scheduling algorithms, i.e., SJF-MMBF, is proposed to determine the solutions. Another scheme that combines the SJF buffering and Reinforcement Learning (RL)-based scheduling algorithms, i.e., SJF-RL, is further proposed to avoid the potential of job starvation in SJF-MMBF. However, due to the continuing high cost of purchasing and maintaining cloud infrastructures, it is impractical to over purchase cloud infrastructures to immediately respond to all cloud users' resource requirements. And in that work depending on the single level queue doesn't provide optimal scheduling result. This problem is focused in this work[8,9].

In this proposed work , A multi-level queue scheduling algorithm is proposed for job scheduling, multi-level queue scheduling partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. Multi-level queue scheduling is performed via the use of the Particle Swarm Optimization (PSO) algorithm. It checks both Shortest-Job-First (SJF) buffering and Min-Min Best Fit (MMBF) scheduling algorithms [10].

II. LITERATURE REVIEW

Karthick et al [11] proposed a Multi Queue Scheduling (MQS) algorithm to reduce the cost of both reservation and on-demand plans using the global scheduler. Scheduling is the most important complex part in cloud computing. The ultimate aim of global scheduler is to share the resources at most

the maximum level. Researcher gives more importance to build a job scheduling algorithms that are well-suited and appropriate in Cloud computing situation. Job scheduling is one of the critical event in cloud computing because the user have to pay for services based on usage time. The proposed methodology depicts the concept of clustering the jobs based on burst time. During the time of scheduling the traditional methods such as First Come First Serve, Shortest Job First, EASY, Combinational Backfill and Improved backfill using balance spiral method are creates fragmentation.

Biswas, et al [12] presented a multi-level queue (MLQ) task scheduling algorithm to minimize the make span for parallelizing the subtasks without violating the precedence relationships. Here, our main objective is to exploit the advantages of heuristic-based task scheduling algorithms in terms of make span, time complexity, resource utilization, system throughput and dynamic nature. Our contribution is analyzed and evaluated through experimental results.

Jaspreet Singh and Deepali Gupta [13] introduced a Smarter MQS model which effectively separate user jobs into two job queues then give more importance in formation of merging jobs pattern by merging user tasks from both queues for execution, so the technique will empower us to reduce energy consumption while naturally to some degree will reduce job completion time and the overall cost. The proposed technique will achieve a high degree of job scheduling in cloud computing environment.

Zhang and Zhou [14] proposed a cloud task scheduling framework based on a two-stage strategy to do so. It procreates VMs according to historical scheduling data, therefore saving time for tasks to wait for creating VMs. It matches tasks with their most suitable VMs dynamically, therefore saving their execution cost. Under the premise of meeting task deadlines, it minimizes the waiting time of VMs to schedule tasks, thus minimizing the cost to be paid by users who utilize VMs. The readily deployable algorithms are designed and illustrated to improve cloud task scheduling and execution results in comparison with those using traditional methods.

Sumit Arora and Sami Anand [15] proposed an efficient scheduling algorithm which will work effectively to provide better result as compared with the traditional scheduling approaches. For this Cloud Sim framework is used to simulate the proposed algorithm under various conditions and presented the better results with reduced the waiting time and processing time with optimum resource utilization and minimum overhead for the same.

Elmougy, et al[16]proposed a novel hybrid task scheduling algorithm named (SRDQ) combining Shortest-Job-First (SJF) and Round Robin (RR) schedulers considering a dynamic variable task quantum. The proposed algorithms mainly relies on two basic keys the first having a dynamic task quantum to balance waiting time between short and long tasks while the second involves splitting the ready queue into two sub-queues, Q1 for the short tasks and the other for the long ones. Assigning tasks to resources from Q1 or Q2 are done mutually two tasks from Q1 and one task from Q2. For evaluation purpose, three different datasets were utilized during the algorithm simulation conducted using Cloud Sim environment toolkit 3.0.3 against three different scheduling algorithms SJF, RR and Time Slice Priority Based RR (TSPBRR) Experimentations results and tests indicated the superiority of the proposed algorithm over the state of art in reducing waiting time, response time and partially the starvation of long tasks.

Zuo, et al [17] proposed an improved ant colony algorithm to solve this problem. Two constraint functions were used to evaluate and provide feedback regarding the performance and budget cost. These two constraint functions made the algorithm adjust the quality of the solution in a timely manner based

on feedback in order to achieve the optimal solution. Some simulation experiments were designed to evaluate this method's performance using four metrics: 1) the makespan; 2) cost; 3) deadline violation rate; and 4) resource utilization. Experimental results show that based on these four metrics, a multi-objective optimization method is better than other similar methods, especially as it increased 56.6% in the best case scenario.

Navimipour and Milani [18] proposed a new evolutionary algorithm which named CSA to schedule the tasks in Cloud computing. CSA algorithm is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior of some birds and fruit flies. The simulation results demonstrated that when the value of Pa is low, the speed and coverage of the algorithm become very high.

S. No	Author name	Method	Merits	Demerits
1.	Karthick et al [2014]	Multi Queue Scheduling.	Achieves the optimum cloud scheduling.	It consumes more time for scheduling .
2.	Biswas, et al [2017]	Heuristic-based task scheduling.	Improves the makespan, complexity and average processor utilization.	Energy consumption is very high.
3.	Jaspreet Singh et al [2014]	Smarter MQS model.	It reduces job completion time and the overall cost.	Need to improve the merging pattern. Higher energy consumption.
4.	Zhang and Zhou [2017]	Two-stage strategy.	Improves cloud task scheduling.	Time consuming nature.
5.	Sumit Arora and Sami Anand [15]	Improved scheduling.	Taking less processing time.	High average waiting time.
6.	Elmougy, et al[16]	Combining Shortest-Job-First (SJF) and Round Robin (RR).	Reducing waiting time, response time.	Need to use other task quantum calculation methodologies.
7.	Zuo, et al [2015]	Improved ant colony algorithm	Improves Resource utilization.	Time consuming nature.
8.	Navimipour and Milani [2015]	CSA	The speed and coverage of the algorithm become very high.	Higher energy consumption.

III. PROPOSED METHODOLOGY

In this section proposed model based job scheduling is described detail. In this model particle swarm optimization based multi queuing is generated. And shortest job first and min to min best fit algorithms is proposed for scheduling the jobs among multi queues. SJF-ELM is proposed to avoid job starvation.

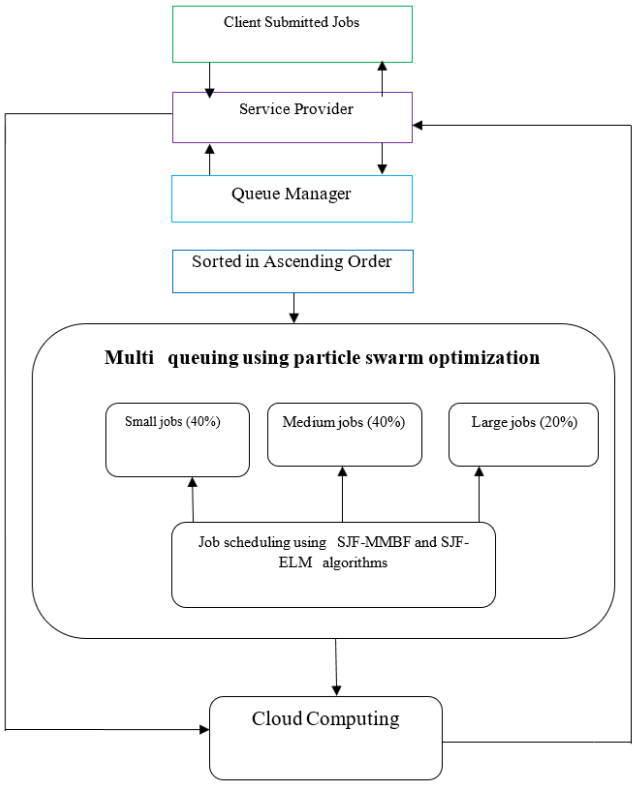

Figure 1. Overall architecture of the proposed model

In the above diagram, first jobs submitted by clients are sorted in the ascending order of the burst time. And then those jobs will be separated as Multi Queues using particle swarm optimization on the basis of burst time like small, middle and high . In multi queues preference (scheduling) to the tasks will be given by checking the SJF-MMBF and SJF-ELM algorithms for resource utilization from cloud. The queue manager plays a vital role for allocating resources to the network. It handles the utilization of all resources in the network. The queue manager checks time to time that which are presently running jobs by balancing force one of the met scheduler and its disposal. It handles the output of the tasks that are collected by the queue manager.

Different queues are made in ascending order on the basis of burst time.

1.In the small jobs queue, jobs have small burst time in which first 40% of jobs are stored.

2. In the medium jobs queue, jobs have medium burst time in which next 40% of jobs are stored.

3. In the long jobs queue, jobs have long burst time in which remaining 20% jobs are stored.

A. Multi Queuing Using Particle Swarm Optimization Algorithm

In this work tasks are queuing using particle swarm optimization algorithm

1. Particle swarm optimization:

Particle swarm optimizers (PSO) are optimization algorithms, modeled after the social behaviour of flocks of birds. PSO is a population based search process where individuals, referred to as particles, are grouped into a swarm. In this work PSO is used to separate all submitted task or jobs into multi queues. In PSO each particle in the swarm represents a candidate solution (job) to the optimization problem. In a PSO system, each particle is "flown" through the multidimensional search space, adjusting its position in search space according to own experience and that of neighbouring particles (jobs).

A particle therefore makes use of the best position encountered by itself and that of its neighbors to position itself toward an optimal solution. The effect is that particles "fly" towards a minimum, while still searching a wide area around the best solution. The performance of each particle (i.e. the "closeness" of a particle to the global optimum) is measured using a predefined fitness function which encapsulates the characteristics of the optimization problem.

Each particle (job) S maintains the following information:

X_i The current position of the particle (job);

V_i The current velocity of the particle (job);

$\mathbf{y_i}$ The personal best position of the particle(job)

The velocity and position of the particle (job) i are calculated by

$$V_{id}^{t+!}=\omega_{id}^{t+1} + C_l * r_{li} * \left(p_{id} - x_{id}^{t}\right) + c_2 * r_{2i} * \left(p_{gd} - x_{id}^{t}\right) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + V_{id}^{t+1} \quad (2)$$

Where t denotes the t th iteration in the process and d denotes the dth dimension in the search space. w is inertia weight and c_1c1 and c_2c2 are acceleration constants. r_{1i} and r_{2i} are random values uniformly distributed . p_{id} And p_{gd} represent the elements of $pbest$ and $gbest$ in the dth dimension [19,20].

The position and velocity values of each job are continuously updated to search for the suitable set of jobs until stopping criterion is met which can be a maximum number of iterations or a satisfactory fitness value(average completion time). The applied PSO algorithm is described.

PSO Algorithm

Step1 swarm (job initialization) Randomly initialize the position and velocity of each particle.

Step2particle (job) fitness (average completion time) evaluation*)*

if fitness of xi>$pbesti$

$pbesti$=xi

if fitness of $pbesti$>$gbesti$

$gbesti$=$pbesti$

Step3. Update the velocity of particle (job)i

$$V_{id}^{t+!}=\omega_{id}^{t+1} + C_l * r_{li} * \left(p_{id} - x_{id}^{t}\right) + c_2 * r_{2i} * \left(p_{gd} - x_{id}^{t}\right)$$

Update the position of particle (job) i

Step4. If stopping criterion is not met, continue Steps 2 and 3.

Step5. Return $gbest$ and its fitness values (average completion time).

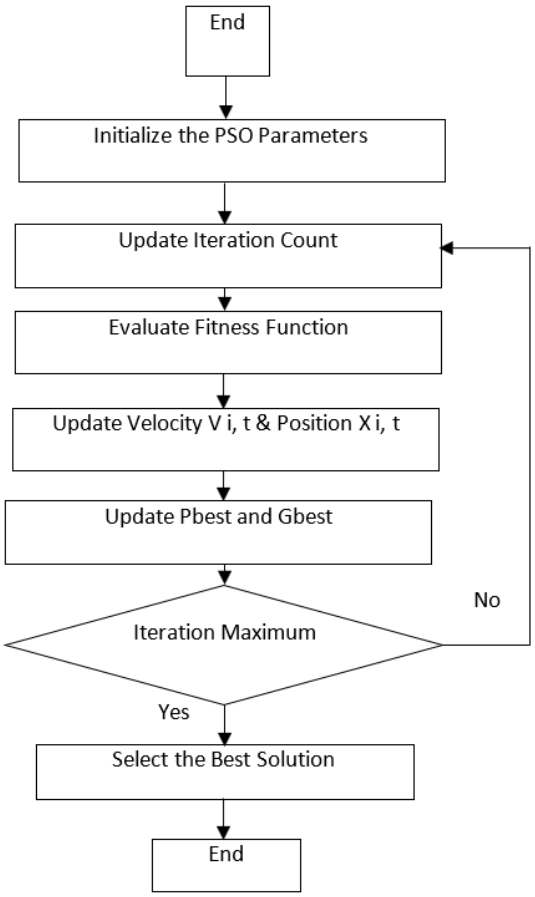

Figure: 2. Flow chart for PSO

In multi queues jobs are allocated using shortest-job-first buffering and min-min best fit scheduling algorithms for resource utilization.

1. Joint Shortest-Job-First Buffering and Min-Min Best Fit Scheduling Policy

Because of the high degree of heterogeneity and dynamism in workloads, it would be difficult and very costly to accurately model the traffic characteristics (e.g., expected arrival rate and average job size) by predicting future resource demands [21,22].

2. Min-Min Best Fit Scheduling Policy

Intuitively, in every decision epoch, if choose an action that uses the most resources among all actions that can be selected, then the average queueing delay would be shortened such that the average job completion time would also be shortened. Therefore, propose our first algorithm, called MMBF, to determine a sequential $N_{a_i^*}$ Originally, best fit was designed to schedule single-resource jobs, such as those involving memory or storage . The main idea of best fit is to find the smallest free resource among multiple segmented free resources that is large enough to satisfy a request, aiming at minimizing the amount of wasted free resources. In contrast, in MMBF, the best fit action is defined as the action that minimizes the remaining multi-resources. Then, the output scheduler always selects the best fit action in every decision epoch to minimize the long-term average job completion time. The details regarding MMBF are as follows [23,24].

Let $\Delta_k(a_t)$ denote the k^{th} normalized remaining resource under scheduling decision

$N_{at} \sqsubseteq N_{At \times V}, a_t \in \{1, \ldots, A_t\}$which is derived as

$$\triangle_K(a_t) = \frac{C_k - \sum_{v=1}^{V} N(a_t, V) R_{VK}}{C_K} \quad (3)$$

Let $\triangle\ (a_t)$ denote the minimum value of $\Delta_K(a_t)$ for k = 1,……K under action at. That is,

$$\Delta(a_t) = \min_{k \in K} \Delta_K(a_t) \quad (4)$$

Then, under the MMBF scheduling policy, the solution a_t^* at time t is the one that satisfies

$$a_t^* = \arg\min_{a_{t=1,\ldots A_t}} \Delta(a_t) \quad (5)$$

3. SJF-BASED INTRA- QUEUE BUFFERING

Since MMBF is not delay-optimal, in this section, focus on the extended problem of how to select jobs of the same type for scheduling when the number of jobs is determined, with the goal of obtaining delay-optimal solutions.

SJF is an efficient non pre-emptive scheduling scheme for achieving average job completion time optimization in a system consisting of a single resource. In SJF, a system schedules the shortest job first, then the next shortest, and so on . Since jobs requesting the same VM type require the same amount of multi-resources, it is possible to buffer them in the same queue and apply SJF to determine their queueing positions such that their scheduling priorities are determined intraqueue to improve the performance in terms of the average job completion time. Therefore, the SJF buffering policy is designed to address the problem of how to select jobs of the same type for scheduling.

Algorithm 1 SJF buffering
While a type-v job f arrives in time interval [t; t + 1), **do**

1) Find a position j' in type-v queue that satisfies
$S_v^{j'} \le s_f \le s_v^{j'+1}$ in type –v queue (6)
2) Insert job f into type-v queue in a position after j' and let

$$\begin{cases} Q_V(t+1) = Q_v(t) + 1 \\ W_v(t+1) = w_v(t) + s_f \end{cases} \quad (7)$$

End while

where S_f is the length of the new arriving job f and s_v^j is the length of the j[th] job in the type-v queue.

4. Finally, combine the SJF buffering and MMBF scheduling policies to form the first scheme, called SJF- MMBF.

1) Buffering Algorithm (SJF Buffering): All the type-v jobs that arrived in time interval [t; t + 1) are buffered in the v[th] queue according to the buffering policy, as described in Algorithm 1, for v ∈ V.
2) Scheduling Algorithm (MMBF Scheduling): In decision epoch t, do
a) Calculate the resource array N $A_t \times$ v .
b) Choose action $a_t^* \in A_t$ such that $N_{a_l^*} \subseteq N_{A_t \times V}$ is determined.
3) Scheduling Process: In time interval [t; t + 1), $N_v^p(t)$ type-v jobs continue to be served, and $\left(N(a_t^*, v)N_v^p(t)\right)$ type-v jobs are de-queued from the v[th] queue in an HOL manner and begin to be served for v ∈ V. The number of jobs waiting in the queue and the accumulative workload requirement are updated, respectively, as follows.

5. Extreme Learning Machine algorithm

SJE-ELM is used here to optimize the long-term average job completion time g(∗). SJF-ELM scheme, the SJF discipline is used to buffer arriving jobs and the ELM-based scheduling policy is designed to determine a sequential a_t^* and N a_t^* to minimize the long-term g(∗).

Since the resource requirements of various types of VMs and the resource capability of a resource pool have been abstracted using the resource array N At ×V. The extreme learning machine (ELM) aims at avoiding time-costing iterative training process and improving the generalization performance .As a single -hidden-layer feed forward neural networks (SLFNs), the ELM structure includes input layer,

hidden layer, and output layer. Different from the traditional neural network learning algorithms (such as BP algorithm) randomly setting all the network training parameters and easily generating local optimal solution, the ELM only sets the number of hidden neurons of the network, randomizes the weights between the input layer and the hidden layer as well as the bias of the hidden neurons in the algorithm execution process, calculates the hidden layer output matrix and finally obtains the weight between the hidden layer and the output layer by using the Moore-Penrose pseudo inverse under the criterion of least-squares method[25,26,27].

Because the ELM has the simple network structure and the concise parameters computation processes, so the ELM has the advantages of fast learning speed. The original structure of ELM is expressed in Figure 3.

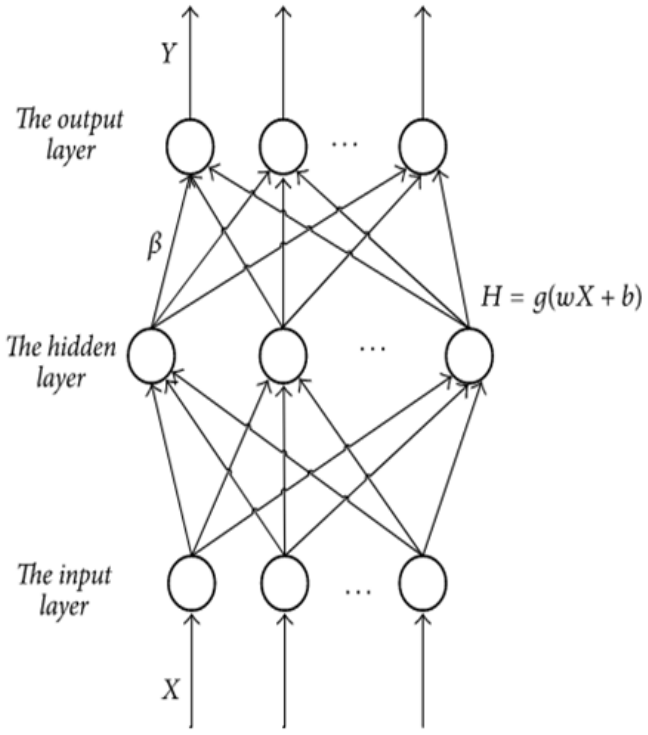

Figure 3 : The structure of the ELM

Figure 3 is the extreme learning machine network structure which includes input layer neurons, hidden layer neurons and output layer neurons. First, consider the training sample and there is an input jobs from a different queues and a desired matrix comprised of the training samples, where the matrix can be expressed as follows.

For N arbitrary jobs from multiple queue like Type 1 , Type 2 and Type 3 jobs $(X_i\ t_i) \in R^d \times R^m$ suppose that the SLFNs construct with N hidden nodes and an activation function g (x) such as the sigmoid function, can be mathematically modeled as follows:

$$\sum_{i=1}^{N} \beta_i\, g(w_i . x_j + b_i) = t_j \qquad (8)$$

where w_i is the weights vector for connecting the ith hidden nodes and the input nodes(jobs), b_i is a bias of the ith hidden nodes and the input action or job β_i is the output weights vector for connecting the ith hidden nodes and the output nodes(scheduled jobs), and $w_i . x_j$ denotes the inner product of $w_i . x_i$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_{11}x_{12}} \dots\dots \boldsymbol{x_{1Q}} \\ \boldsymbol{x_{21}x_{22}} \dots\dots \boldsymbol{x_{2Q}} \\ \vdots \\ \boldsymbol{x_{n1}x_{n2}} \dots\dots \boldsymbol{x_{nQ}} \end{bmatrix} \qquad (9)$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y_{11}y_{12}} \dots\dots \boldsymbol{y_{mQ}} \\ \boldsymbol{y_{21}y_{22}} \dots\dots \boldsymbol{y_{mQ}} \\ \vdots \\ \boldsymbol{y_{m1}y_{m2}} \dots\dots \boldsymbol{y_{mQ}} \end{bmatrix} \qquad (10)$$

Where the parameters X, Y are the dimension of an input action space matrix and output decision or scheduled matrix .

Then the ELM randomly sets the weights between the input layer and the hidden layer:

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega_{11}\omega_{12}} \dots\dots \boldsymbol{\omega_{1n}} \\ \boldsymbol{\omega_{21}\omega_{22}} \dots\dots \boldsymbol{\omega_{2n}} \\ \vdots \\ \boldsymbol{\omega_{l1}\omega_{l2}} \dots\dots \boldsymbol{\omega_{ln}} \end{bmatrix} \qquad (11)$$

Where ω represents the weights between the inputs layer neuron and the hidden layer neuron.

Third, the ELM assumes the weights between the hidden layer and the output layer that can be expressed as follows:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta_{11}}\boldsymbol{\beta_{12}} \ldots . \boldsymbol{\beta_{1m}} \\ \boldsymbol{\beta_{21}}\boldsymbol{\beta_{22}} \ldots . \boldsymbol{\beta_{2m}} \\ \vdots \\ \boldsymbol{\beta_{l1}}\boldsymbol{\beta_{l2}} \ldots \boldsymbol{\beta_{lm}} \end{bmatrix} \quad (12)$$

Where represents the weights between the hidden layer neuron and output (scheduling result) layer neuron.

Fourth, the ELM randomly sets the bias of the hidden layer neurons:

$$\boldsymbol{B} = [\boldsymbol{b_1}\boldsymbol{b_2} \ldots . \boldsymbol{b_n}]^{\boldsymbol{T}} \quad (13)$$

Fifth, the ELM chooses the network activation function .The output matrix job scheduling can be expressed as follows:

$$\boldsymbol{T} = \left[\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots . \boldsymbol{t_Q}\right]_{\boldsymbol{mxQ}} \quad (14)$$

Each column vector of the output matrix is as follows:

$$\boldsymbol{t_j} = \begin{bmatrix} \boldsymbol{t_{1j}} \\ \boldsymbol{t_{2j}} \\ \vdots \\ \boldsymbol{t_{mj}} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{l} \boldsymbol{\beta_{i1g}} & (\boldsymbol{\omega_i x_j} + \boldsymbol{b_i}) \\ \sum_{i=1}^{l} \boldsymbol{\beta_{i2g}} & (\boldsymbol{w_i x_j} + \boldsymbol{b_i}) \\ & \vdots \\ \sum_{i=1}^{l} \boldsymbol{\beta_{img}} & (\boldsymbol{\omega_i x_j} + \boldsymbol{b_j}) \end{bmatrix} (\boldsymbol{j} = \boldsymbol{1}, \boldsymbol{2}, \boldsymbol{3} \ldots . . , \boldsymbol{Q}). \quad (15)$$

Sixth, consider formulae (14) and (15), and can get

$$H\beta = T' \quad (16)$$

Next in step 6, (14) and (15) are computed and achieve

$$H\beta = T' \quad (16)$$

layer of hidden and T' for output transposition. Using the least square method of measuring the weight matrix values of the minimum error to achieve a unique solution [15, 16].

$$= H + \mathbf{T'}. \quad (17)$$

To improve network simplification and stabilize the output, add a regularization concept. In this criteria, neurons of hidden layer are less than training samples and represented as

$$= I + HTH - 1HTT' \quad (18)$$

In this criteria, neurons of hidden layer are greater than training samples and represented as

$$= HTI + HHT - 1T' \quad (19)$$

Feature		planes
State s	Number of jobs being served, N_v^p	N ×V
	Number of type-1 jobs waiting to be served, Q_v	(N + 1) ×V
	Workload requirement, W_v	(N + 1) ×V
Action a	VM configuration N_a at action a	N ×V

Table 1. State-action features for job scheduling

6. SJF- ELM

1) Algorithm of Buffering (Buffering in SJF): A range of [t; t + 1] are buffering time of sort of v jobs accordingly on queue as well as buffering policy. It is deployed in Algorithm SJF as SJF-ELM, v ∈ V.

2) Scheduling Algorithm (ELM Scheduling): In $\boldsymbol{t_j}$ of decision epoch, perform

$S_t \leftarrow \left(N_v^p(t)\right), Q_v(t), W_v(t)$ State is intellect.

Huge X actions are calculated feasibly and N A_s× v as array resource.

3) Scheduling Process:

a) Scheduling: jobs of $N_v^p(t)$ type 1 are being supported in queue along vϵV, as well as de-queue $\left(N\left(a_t^*, v\right) - N_v^p(t)\right)$ type-1 jobs from the 1th queue and start serving the function. queue holding plenty of waiting jobs and update the required workload in accumulation are from

$$\begin{cases} Q_v(t+1) = Q_V(T) - \left(N\left(a_t^*, v\right) - N_v^p(t)\right) \\ w_v(t+1) = W_v(t) - N\left(a_t^*, v\right) \end{cases}$$

b) Computation of time for job completion in time-averaged as follows

$$\boldsymbol{E\left[\tilde{T}(t)\right]} = \sum_{v=1}^{V} \boldsymbol{\alpha E\left(\hat{T}_v(t-1)\right) + (1-\alpha) T_v(t)}$$

Where $\boldsymbol{\alpha} \in (\mathbf{0}, \mathbf{1})$ is a weight parameter.

c) If $\boldsymbol{E\left[\widetilde{T(t)}\right] T_j} > E[\boldsymbol{T}^*]$, $\omega *$,vector of parameter are updated

d) The final number of arrived traffic T gets stored as $\{j_v(t-T+1), \ldots . j_v(t)\}$ [27].

IV. RESULT AND DISCUSSION

This section discusses the experimental results of proposed model. The model is implemented using JAVA. This model compared with the proposed multi queuing using particle swarm optimization (MPSO) based scheduling and the existing hybrid task scheduling algorithm named (SRDQ), optimal delay scheduling(ODS) are compared in terms of Throughput, Delay and cost.

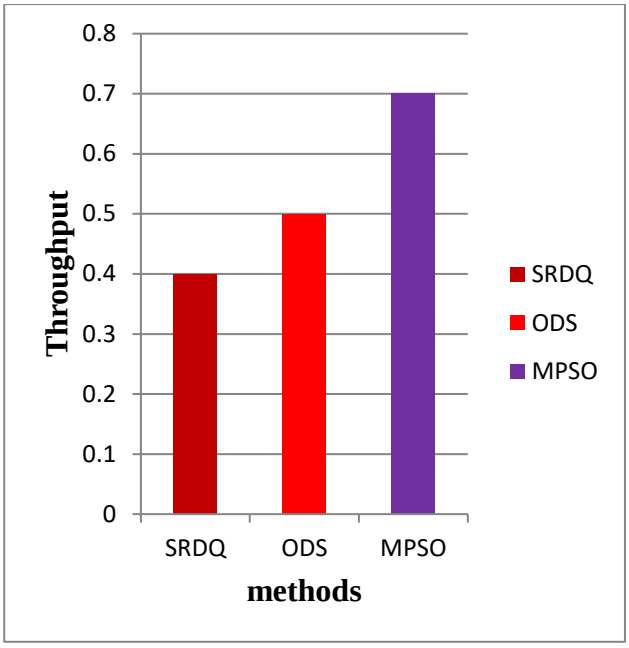

Figure 4. Throughput results vs. classification methods

Figure 4 Shows the throughput performance comparison outcomes of existing SRDQ, ODS method and proposed MPSO method. In the above graph Scheduling methods are taken as X Axis and throughput values are taken in Y Axis. From outcome, it confirmed that proposed MPSO model generated superior throughput results of 0.7 whereas existing SRDQ, ODS method gives only0.4 and 0.5 correspondingly.

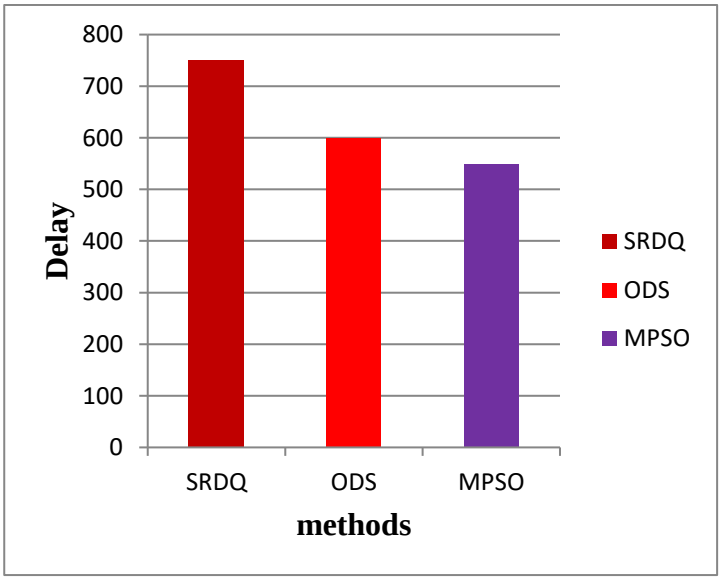

Figure 5. Delay results vs. classification methods

Job scheduling result is shown in figure 5 in terms of delay for the existing SRDQ, ODS method and

proposed MPSO method. From outcome, it confirmed that proposed MPSO model generated superior Delay results of 0.7 whereas existing SRDQ , ODS method gives only0.4 and 0.5 correspondingly.

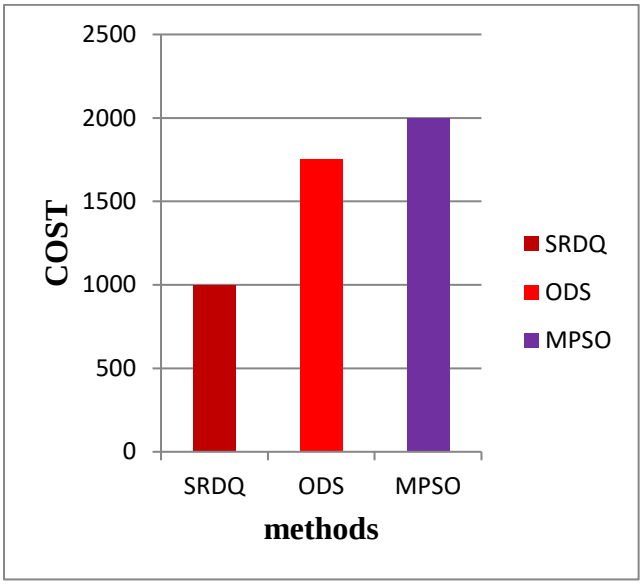

Figure 6. Cost results vs. classification methods

Overall result comparison of the proposed model based job scheduling is shown the above figure for Cost metric with the existing SRDQ, ODS methods and proposed MPSO method. From outcome, it confirmed that proposed MPSO model generated lesser cost expensive results of 0.7 for scheduling whereas existing SRDQ , ODS method gives only0.4 and 0.5 correspondingly.

V. CONCLUSION AND FUTURE WORK

A multi-level queue scheduling algorithm is proposed via the use of the Particle Swarm Optimization (PSO) algorithm in this work which partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority or process type. It checks both Shortest-Job-First (SJF) buffering and Min-Min Best Fit (MMBF) scheduling algorithms.

Another scheme that combines the SJF buffering and Extreme Learning Machine (ELM)-based scheduling algorithms, i.e., SJF- ELM, is further proposed to avoid the potential of job starvation in SJF-MMBF. In addition From the results it concluded that the proposed model gives better throughput performance. In future hybrid task scheduling algorithm can also be developed. And also the effect of precedence between tasks and load balancing will be considered.

VI. REFERENCES

[1]. Shorgin, S., Pechinkin, A., Samouylov, K., Gaidamaka, Y., Sopin, E. and Mokrov, E., 2014, October. Queuing systems with multiple queues and batch arrivals for cloud computing system performance analysis. In 2014 International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC) (pp. 1-4). IEEE.

[2]. Eisa, M., Esedimy, E.I. and Rashad, M.Z., 2014. Enhancing cloud computing scheduling based on queuing models. International Journal of Computer Applications, 85(2).

[3]. Singh, I. and Arora, A., 2015. Fuzzy Based Improved Multi Queue Job Scheduling For Cloud Computing. International Journal of Advanced Research in Computer Science, 6(5).

[4]. Singh, S. and Chana, I., 2016. A survey on resource scheduling in cloud computing: Issues and challenges. Journal of grid computing, 14(2), pp.217-264.

[5]. Nan, X., He, Y. and Guan, L., 2014. Queueing model based resource optimization for multimedia cloud. Journal of Visual Communication and Image Representation, 25(5), pp.928-942.

[6]. Sowjanya, T.S., Praveen, D., Satish, K. and Rahiman, A., 2011. The Queueing Theory in Cloud Computing to Reduce the Waiting Time. International Journal of Computer Science Engineering & Technology, 1(3).

[7]. Bhoi, U. and Ramanuj, P.N., 2013. Enhanced max-min task scheduling algorithm in cloud

computing. International Journal of Application or Innovation in Engineering and Management (IJAIEM), 2(4), pp.259-264.

[8]. LD, D.B. and Krishna, P.V., 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 13(5), pp.2292-2303.

[9]. Agarwal, D. and Jain, S., 2014. Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint arXiv:1404.2076.

[10].Salot, P., 2013. A survey of various scheduling algorithm in cloud computing environment. International Journal of Research in Engineering and Technology, 2(2), pp.131-135.

[11].Karthick, A.V., Ramaraj, E. and Subramanian, R.G., 2014, February. An efficient multi queue job scheduling for cloud computing. In 2014 World Congress on Computing and Communication Technologies (pp. 164-166). IEEE.

[12].Biswas, T., Kuila, P. and Ray, A.K., 2017, January. Multi-level queue for task scheduling in heterogeneous distributed computing system. In 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)(pp. 1-6). IEEE.

[13].Jaspreet Singh and Deepali Gupta. An Smarter Multi Queue Job Scheduling Policy for Cloud Computing. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 9 (2017) pp. 1929-1934.

[14].Zhang, P. and Zhou, M., 2017. Dynamic cloud task scheduling based on a two-stage strategy. IEEE Transactions on Automation Science and Engineering, 15(2), pp.772-783.

[15].Sumit Arora and Sami Anand.Improved Task Scheduling Algorithm in Cloud Environment. International Journal of Computer Applications (0975 – 8887). Volume 96– No.3, June 2014

[16].Elmougy, S., Sarhan, S. and Joundy, M., 2017. A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud Computing, 6(1), p.12.

[17].Zuo, L., Shu, L., Dong, S., Zhu, C. and Hara, T., 2015. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. Ieee Access, 3, pp.2687-2699.

[18].Navimipour, N.J. and Milani, F.S., 2015. Task scheduling in the cloud computing based on the cuckoo search algorithm. International Journal of Modeling and Optimization, 5(1), p.44.

[19].Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters, 85(6), pp.317-325.

[20].Chander, A., Chatterjee, A. and Siarry, P., 2011. A new social and momentum component adaptive PSO algorithm for image segmentation. Expert Systems with Applications, 38(5), pp.4998-5004.

[21].Guo, M., Guan, Q. and Ke, W., 2018. Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload. IEEE Access, 6, pp.15178-15191.

[22].Kaur, R. and Kinger, S., 2014. Analysis of job scheduling algorithms in cloud computing. International Journal of Computer Trends and Technology (IJCTT), 9(7), pp.379-386.

[23].Ru, J. and Keung, J., 2013, June. An empirical investigation on the simulation of priority and shortest-job-first scheduling for cloud-based software systems. In 2013 22nd Australian Software Engineering Conference (pp. 78-87). IEEE.

[24].Salot, P., 2013. A survey of various scheduling algorithm in cloud computing environment. International Journal of Research in Engineering and Technology, 2(2), pp.131-135.

[25].Huang, G.B., Zhu, Q.Y. and Siew, C.K., 2004. Extreme learning machine: a new learning scheme of feed forward neural networks. Neural networks, 2, pp.985-990.

[26].Li, M.B., Huang, G.B., Saratchandran, P. and Sundararajan, N., 2005. Fully complex extreme learning machine. Neurocomputing, 68, pp.306-314.

[27].Guo, M., Guan, Q. and Ke, W., 2018. Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload. IEEE Access, 6, pp.15178-15191.

Cite this article as :

S. Rekha, Dr. C. Kalaiselvi, "Load Balancing Using SJF-MMBF and SJF-ELM Approach", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 1, pp. 74-86, January-February 2021. Available at doi : https://doi.org/10.32628/CSEIT21714
Journal URL : http://ijsrcseit.com/CSEIT21714