Systematic Scheduling of Jobs in Cloud Computing Environment for Efficient Utilization of Available Resources

Pooja Kushwaha
M. Tech in Computer
Science and Engineering,
DIT University, Dehradun (India)

ABSTRACT
Cloud computing provide many services on demand to their end users, and customer can borrow those resources from CSP on only pay-per-use basis. There are many issues arises day by day in cloud computing environment. Job scheduling is one of the major issues. In scheduling we are focusing on to execute maximum no. of user's jobs by utilizing minimum no. of resources which is available in cloud computing. Also scheduling of user's jobs defines how to allocate an appropriate resource to these request come from end users to finish task in minimum time. In this research paper, we are introducing cloud computing, job scheduling and Artificial Neural Network (ANN) based task allocation model has been proposed to increase the performance of the cloud computing system and also find the optimal system cost.

Keywords
Cloud computing, Job scheduling, Min-Min, trainlm function

1. INTRODUCTION
Cloud computing are playing a vital and rising role in today's networks. Cloud computing provides pool of resources and services on demand to end users and cloud users also uses these services only pay-per-use basis. Cloud computing is an internet based computing that delivers infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). In SaaS, software application is made available by the cloud provider. In PaaS, an application development platform is provided as a service to the developer to create a web based application. In IaaS computing infrastructure is provided as a service to the requester in the form of Virtual Machine (VM) [1]. In cloud computing there are many issues introduced in their developing stage. One of them are scheduling of jobs to appropriate resources where set of jobs "n" is always greater than set of resources "m"[6].

Managing the distribution of jobs in a networked system is a complex yet crucial requirement. Real world distribution problems raise some practical considerations usually not addressed in a realistic way in more theoretical studies, where the focus is usually on the exactness of the solution rather than the suitability of the approach from the computational intractability and time complexity point of view. Artificial Neural Network (ANN) has emerged as relatively new technology which has the capability to act as universal approximation tool to provide systems which may learn from the environment and produce requisite output. The exponential growth of computational resources in computer networks necessitates the need of new and intelligent methods to meet the challenges of computationally intractable problems involving jobs allocation in multiprocessors systems, operation research, job scheduling in industries and transportation problems etc. Emergence of Artificial Neural Networks (ANNs) may provide new self-learning methods to address such problems.

2. PROBLEM FORMULATION
The parallel application should finish its execution over a Cloud Computing Environment in lesser cost than over an individual system. The parallel applications running over the Cloud Computing Environment is having *m* number of jobs which are to be allocated on *n* number of resources of the cloud computing for each task. The node of minimum cost for the jobs should be accepted for processing of it. Artificial Neural Network base job allocation model has been proposed, which give an optimal solution for the assignment of the set of "*m*" jobs of programs over the set of "*n*" cloud resources(where, $m>n$)[2] in Cloud Computing Environment. The object of this model is to increase the performance of the cloud computing environment, which will be achieved by ANN based allocation of jobs in the system.

3. PROBLEM STATEMENT
In this section, an ANN based job allocation model has been developed to find the optimal system cost so that system performance could be improve. Effectual allocation of parallel applications jobs may increase the performance of the cloud computing environment. In the suggested model the training data and the target data is required. To get this data "Min-Min" algorithm is used [3]. The data for this technique is generated randomly. Based on the results by this algorithm, the neural network of this proposed model will be trained. After the training of the network problem can be solved by using the trained net.

4. EXECUTION COST (EC)
For the task allocation *X*, the execution cost ec_{il} represents the execution of task t_i on processor P_l. Therefore, under task allocation *X*, the execution of all the tasks assigned to l^{th} processor can be computed as [4].

$$EC(X) = \sum_{i=1}^{n} \sum_{i=1}^{m} ecil\ xil$$

$$X_{il} = \begin{cases} 1, \text{ if } i^{th} \text{ task is assigned to } l^{th} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$$

5. TRAINING DATA
The Training Input Weight Matrix *TIWM (,)* and Target Output Vector *TOV* have been generated by using the "Min-Min" technique. In "Min-Min" technique proposed in [5] uses the set *U* of all unmapped jobs and the set of minimum completion time $M= \{min_{0\leq j<m} (ct(t_i, m)), \text{ for each } t_i \varepsilon U\}$.

The job having overall minimum completion time is selected from the set *M* and assigned to consequent processor. Now this job is removed from *U* and this course of action is repeated until *U* is empty i.e. all jobs are mapped.

6. FITTING NEURAL NETWORK

The feed forward neural networks are fitting networks used to fit an input-output relationship. For this purpose the parameters like size of hidden layers and training function will be used. On the basis of these parameters a fitting neural network will be returned. A feed forward network will be used here. It consists of a series of layers. The first is meant for the network input and the subsequent layer has a connection from the previous layer. The last layer is for the network's output. Feed forward network with one hidden layer and enough neurons in the hidden layers, can fit any finite input-output mapping problem.

7. TRAINING ALGORITHM

The LM (Levenberg-Marquardt) back propagation training algorithm [8] has been used to train the network in the model suggested in this chapter. This algorithm updates the weight and bias values according to the Levenberg-Marquardt optimization. In the supervised learning algorithm, Levenberg-Marquardt is the fastest back propagation algorithm. This training algorithm takes three inputs training data, validation data and test data which in turn return the trained network and training record of various values over each epoch.

MATLAB implements the above mechanism through *trainlm* function. The *trainlm* supports training with validation and test vectors. Validation vectors are used to shop training early if the performance of networks on the validation vectors fails to improve or remains the same for maximum fail epochs. Test vectors are used as further check that the network is generalizing well but do not have any effect on training.

The back propagation [7] [9] is used to compute the Jacobian JX of performance with respect to the weight and bias variables. Each variable is adjusted according to Levenberg-Marquardt,

$$jj = (jX) ; je = jX * E ; \ dX = \frac{jj + I * mu}{je}$$

Where *E* is all errors and *I* is known for identity matrix.

The adaptive value mu is enlarged or (increased) by mu_inc until the change above results in a reduced performance value. The change is then complete or (made) to the network and mu is decreased by mu_dec. Training stops when any of these conditions occurs: the maximum no. of epochs is reached, the maximum amount of time is exceeded, performance is minimized to the goal, the performance gradient falls lower or (below) min_grad, mu exceeds mu_max, and validation performance has increased more than max_fail times since the last time it decreased. After training the network, this trained and fitted network will be used to find the allocation of the desired problem.

8. PROPOSED MATHEMATICAL MODEL FOR JOB ALLOCATION

The Training Input Weight Matrix TIWM(,) and Target Output Vector TOV have been generated by using the "Min-Min" technique. In "Min-Min" technique proposed in [5] uses the set U of all unmapped jobs and the set of minimum completion time $M= \{min_{0 \le j < m}(ct\ (t_i\ ,\ m))\ for\ each\ t_i\ \varepsilon\ U\}$. The job having overall minimum completion time is selected from the set *M* and assigned to consequent processor. Now this job is removed from *U* and this course of action is repeated until *U* is empty i.e. all jobs are mapped. Transmit each row of TIWM (,) to the hidden until and calculate the output using **eq()** and output the send to the output layer units. From the output layer, output signal will be calculated by using **eq()**.Now compute the error correction factor f_k using **eq()**.Receive input signal xi and transmit to hidden unit. Once the final assignments are in hand, optimal cost of assignment is to be computed. The objective function to calculate total system cost is as follows: Total Cost = EC + CC

9. PROPOSED ALGORITHM

The algorithm consists of many steps:

Step-0: Start

Step-1: Input: *m, n, ECM (,)*

Step-2: Provide the ECM (,) to the trained and fitted neural network

Step-3: Calculate the Optimal Allocation by using the trained neural network

Step-4: Set value of TOV

Step-5: Based on TOV values determine the allocation of each job to the appropriate processor

Step-6: Calculate Execution Cost over each processor

Step-7: Calculate the Overall System Cost

Step-8: End

10. IMPLEMENTATION

The proposed model is simulated in MATLAB using following data set. The cloud computing environment consist a set of 3 cloud processor units *{P1,P2,P3}* and set of 6 jobs *{J1,J2,J3,J4,J5,J6}*. Each task on various processors, the processing cost(s) are known and stated in the matrix namely *ECM (,)*. The execution cost of each job/file at different processors is given in Table 1. The communication amongst the jobs has not been taken into consideration.

To train the network, the Training Input Weight Matrix *TIWM (,)* has been generated by using random function. The Target Output Vector *TOV* has been calculated by using "Min-Min" technique. The training data available in *TIWM (,)* and the target output available in *TOV* has been provided to the neural network. Based on this data the neural network has been trained accordingly.

Now this trained and fitted network is being used to solve the problem taken for simulation of the proposed model. This trained network has taken this *ECM (,)* as input. The final mapping of the tasks to processors has been calculated and stored in *TOV* and show in Table 6

Table 1: Execution Cost Matrix

Processors / Jobs	P1	P2	P3
j1	2300	2400	2500
j2	2100	2700	2800
j3	2800	2800	2100
j4	2000	3000	2700
j5	2400	2300	2200
j6	2100	1900	2700

Table 2: Reduced Cost Matrix-1

Jobs	P1	P2	P3
j1	**0**	100	200
j2	0 ×	600	700
j3	700	700	**0**
j4	0 ×	1000	700
j5	200	100	0 ×
j6	200	**0**	800

Table 3: Processor assigned to job

j1	P1
j6	P2
j3	P3

Table 4: Reduced Cost Matrix-2

Jobs	P1	P2	P3
j2	**0**	0 ×	100
j4	0 ×	300	**0**
j5	100	**0**	0 ×

Table 5: Processor Assigned to job

j2	P1
j4	P3
j5	P2

Table 6: Rounded off Target Output Vector

1	1	3	3	2	2

Based on the allocation, given by the neural network which is stored in TOV, the Modified Execution Cost Matrix MECM (,) has been calculated and it has been shown in Table 7.

Table 7: Modified Execution Cost Matrix

Processors / Jobs	P1	P2	P3
J1,j2	4400	5100	5300
J3,j4	4800	5800	4800
J5,j6	4500	4200	4900

The Final mapping of the jobs/files to processors for the current allocation problem has been shown in Table 8. By implementing the proposed model the final allocation are j1, j2→P1; j3, j4→P3; j6, j5→P2.**The optimal system cost is 12600.**

Table 8: Final Mapping Status

Processor	Jobs
P1	J1,j2
P2	J6,j5
P3	J3,j4

11. RESULTS

In the proposed model a feed-forward network with the default tan-sigmoid transfer function in the hidden layer and linear transfer function in the output layer has been used. There were 15 hidden layers in the network. Then network has one output neuron, just because of there is only one target value connect with each input vector [10]. TIWM (,) is the collection of input vectors which is shown in Figure 1.

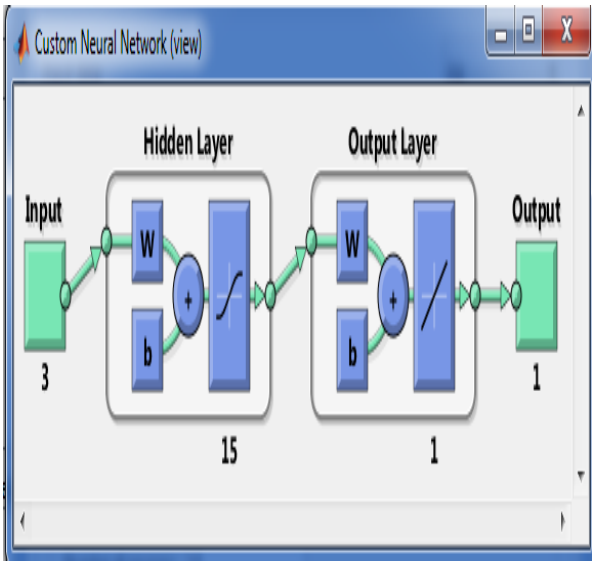

Figure 1: Task Allocation Neural Network

For training, the default Levenberg-Marquardt algorithm has been used in the network. This MATLAB application divides the input vectors and the target vectors randomly into three sets. 60% of the data is used for training, 20% of it is used to validate that network is generalizing and this section of data is also being used to stop the training before over fitting. Rest 20% is being used for the test of network generalization independently. Figure 2 shows the whole training process. This is interface that allows us to interrupt training at any point by clicking (hit the button of) Stop Training.

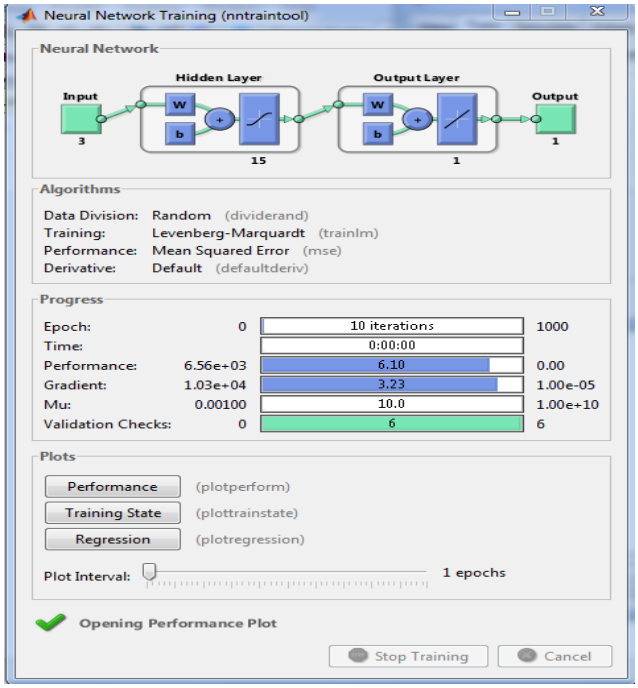

Figure 2: Neural Network Training Interface

This training process has stopped when the validation error increased, which occurred at iteration 10. If we choose the performance option in the training interface, a plot consisting the training errors, test errors, and validation

errors will appeared, which has been given in Figure 3. The result achieved using this model is reasonable because the final mean-square error is small, the characteristics of train set error and the validation set error are similar and there is no significant over fitting occurrence by epoch 4. At the fourth epoch it has the best validation performance. The training state has been shown in Figure 4.

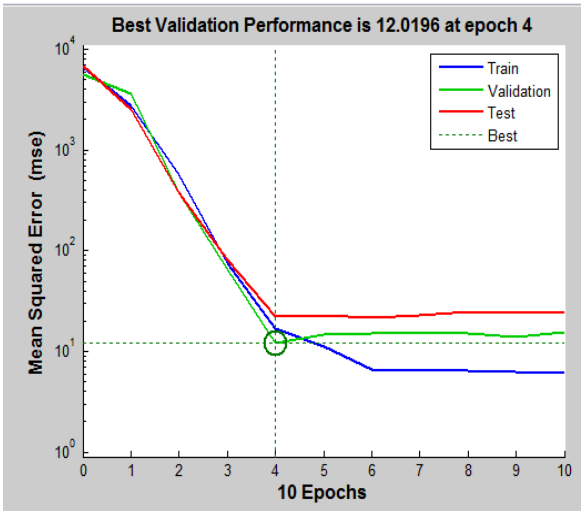

Figure 3: Neural Network TrainingPerformance

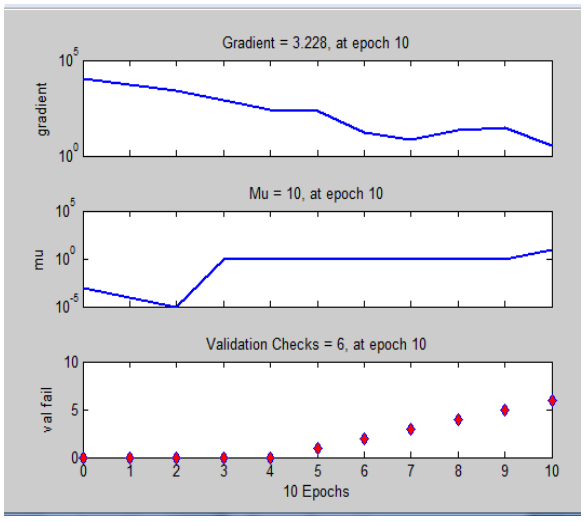

Figure 4: Neural Network Training State

To perform a linear regression between the Execution Cost of jobs over each processor and its corresponding allocation to the processor neural network training regression has been chooses from the interface and it is shown in Figure 5.The training validation and testing data is represented by the plots. It is showing the network outputs with respect to targets for training, validation, and test sets. The data must fall along a 45 degree line for a perfect fit. In this condition the network outputs are equal to the targets. In each plot, the dashed line represent the perfect result – output = targets. The best fit linear regression is represented by the solid line between the outputs and targets. The relationship between the outputs and targets is indicated by the value close to zero of R shows that there is no linear relationship between outputs and targets. In the example taken for the simulation of the model, the validation and test results shows that the R values is greater than 0.95. This indicates that the training data is of good fit.

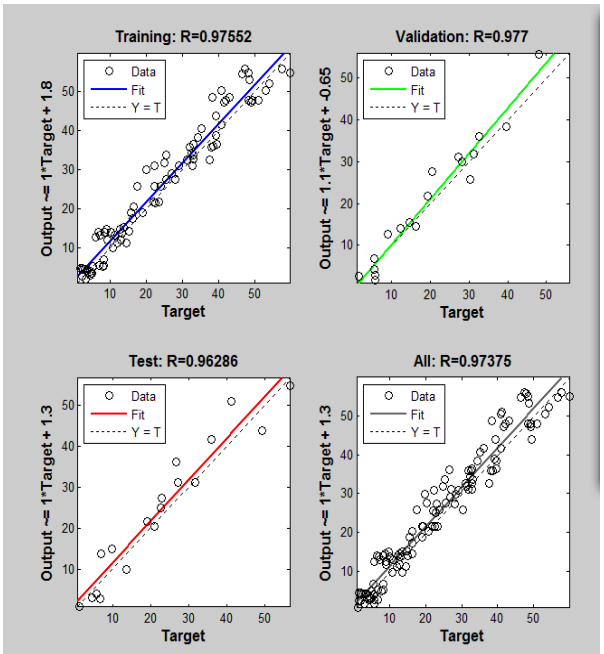

Figure 5: Neural Network Training Regression

The above analysis shows that the neural network is trained and fitted which can be used to solve the problems. The *ECM (,)* has been given to the network to find the appropriate

allocation. The results given by the network has been shown in Table 8. The trained and fitted network has assigned jobs *j1, j2→P1; j6, j5→P2; j3, j4→P3*. The optimal system cost is 12600 and the APU is 0.93053333 which has been shown in Table 9.

Table 9: Optimal Task Assignment

Processors	Job	Processor Load	PU	APU	Optimal Cost
P1	J1,j2	4400	0.9166	0.93053333	12600
P2	J6,j5	4200	0.8750		
P3	J3,j4	4800	1		

12. CONCLUSION

As we also know that scheduling is one of the major issues in cloud computing environment. In this research work we have introduced basic concepts of cloud computing, analysis of scheduling algorithm along with that include ANN based task allocation model to solve problem which is assignment of six jobs to its set of three appropriate resources. To solve problem we need TIWM and TOV. Our work is simulated in MATLAB Tool through "trainlm" function. By using all techniques and parameters we increase the performance of distributed system. We generate optimal system cost is 12600 and the APU is 0.93053333 by our proposed model.

13. REFERENCES

[1] Dr. Amit Agarwal, Saloni Jain, Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment, IJCTT-volume 9 number 7-Mar 2014.

[2] Urmani Kaushal, Avanish Kumar, Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks under Dynamic Load Sharing Scheme, International Journal of Scientific & Engineering Research, Volume 4, Issue 7, July-2013.

[3] O.M Elzeki, M.Z Reshad, M.A Elsoud, Improved Max-Min Algorithm in Cloud Computing, International Journal of Computer Application(0975-8887) Volume 50-No.12 July 2012.

[4] Urmani Kaushal, Avanish Kumar, Performance Intensification of DRTS under Static Load Sharing Scheme, International Journal of Computer Applications (0975 – 8887) Volume 71– No.16, June2013.

[5] George Amalarethinam. D.I, Vaaheedha Kfatheen .S, Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2), 2012.

[6] Mahmoud Maqableh, Huda Karajeh, Ra'ed (Moh'd Taisir) Masa'deh, Job Scheduling for Cloud Computing Using Neural Networks, Communications and Network, 2014, 6, 191-200.

[7] Bogdan M. Wilamowski, Nicholas J. Cotton, Okyay Kaynak and Günhan Dündar, Computing Gradient Vector and Jacobian Matrix in Arbitrarily Connected Neural Networks, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 55, NO. 10, OCTOBER 2008.

[8] S.Sapna, Dr.A.Tamilarasi and M.Pravin Kumar, BACKPROPAGATION LEARNING ALGORITHMBASED ON LEVENBERG MARQUARDT ALGORITHM, CS & IT-CSCP 2012.

[9] Ozgur kisi, Erdal Uncuoglu, Comparision of three-backpropagation training algorithms for two case studies, Indian Journal of Engineering & Materials Sciences, Vol. 12, October 2005.

[10] Devika Chhachhiya, Amita Sharma, Manish Gupta, Case Study on Classification of Glass using Neural Network Tool in MATLAB, International Conference on Advances in Computer Engineering & Applications (ICACEA-2014).