Int. J. Math. And Appl., 6(1–E)(2018), 877–887
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Cloud Load Prediction for Multiuser Mixed Distribution Task Length and Task Arrival Scenario

Vikash Goswami[1,*] and R. K. Shrivastava[1]

1 Department of Mathematics, S.M.S Government Science College, Jiwaji University, Gwalior, Madhya Pradesh, India.

Abstract: In the mathematics queuing theory is one of the fields which can be used for finding solutions of a wide range of practical problems. One of the recently growing application of this field is in computer network traffic management. In this paper we applied the queuing theory for the prediction of job request queue in a cloud network where the job requests are arriving through a single queue and are combination of jobs generated by a number of users with each having the different job request patterns (distribution function). The growing adaption of cloud systems, resulting in excessive loading of cloud servers. Since the cloud is designed to provide different resources as service over the internet. The cloud user requests required resources from the cloud and the cloud serves these requests by forming virtual machines and allotting it some the resources from cloud resource pool. The resources allotted to VMs depends upon the requirements of the task and guaranteed QoS of the cloud. Once the task is completed these VMs can either be dissolved to utilize its resources for another VM or can be suspended for later utilization and saving power. The creation of a VM may take several seconds while activation of a suspended VM is also required some time, which may ultimately cause sluggish response and latency in cloud response. Such situations can be avoided by predicting the upcoming task requirements. The knowledge of upcoming tasks can be used to decide whether VMs needs to keep alive, suspended or dissolved to efficiently utilize the resources, saving power and serve the requests with minimum latency. The prediction of task requirements is a difficult job especially when the task requests are generated by different users and each user may follow different distribution function. This paper presents the HMM-based prediction model for such conditions and evaluates its performance.

Keywords: Queue Prediction, Cloud Computing, HMM, Task Scheduling.

1. Introduction

The queuing theory dedicated field for finding the solution of queue management problem and has already applied to a number of practical applications like transportation scheduling, manufacturing processes, appointment management, computer network traffic management etc. The computer networking is a vast area one of them is cloud computing which is attracting lots of users towards this technology. The cloud was designed to provide different resources as service over a network (generally internet) and defined as the delivery of computing services like servers, storage, software and more over the Internet. These services can be divided into three categories such as software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). Since a number of users with different requirements may simultaneously request for resources from the cloud system. To serve these requests a number of VMs (virtual machines) are formed inside the cloud from its available resources in such a manner that it fulfills the user's requirements and guaranteed the QoS with minimum cloud resource utilization. Because the formation of VMs does not require any physical reconfiguration as it's all done using software that virtually divides or allocates the resources to form VMs. Such characteristics of VM provides the flexibility to the cloud as it can reconfigure the resources of VMs dynamically and quickly when required. The dynamic

* *E-mail: vikash.goswami088@gmail.com*

operational state of VM depends upon the current and future status of load, generally it can be kept in one of the three states alive, suspended or dissolved. The VM is kept alive if jobs are continuously arriving. It can be kept in suspended for near future utilization which can significantly reduce the response time because the suspended VM can be activated in a fraction of the time required for reformation however in such case the resources will be blocked although the power can be saved. Hence for the efficient operation of cloud it is needed to manage a tradeoff among the different states of VMs like keep alive, suspend or dissolve (In the presented scenario the arbitrary configuration of VMs is not considered instead a fixed number of predefined configurations are taken). Presently the queuing theory is gaining the applicability in the area of cloud computing where it is used to manage the clients requests arriving into the servers. However unlike other fields the cloud computing requires much complex queuing management system because of the of the complex client request characteristics, higher rate of request arrival, availability of resources, guaranteed QoS, operational cost and lower calculation time. Knowing the fact that present states of VMs must be maintained on the basis of upcoming service requests. This paper presents a queue prediction based solution for the above-mentioned problem.

2. Related Work

This section reviews the literature most related to our work. Rich Wolski et al. [12] presented and efcient predictive scheduling methodology for power savings in private clouds. It estimates the distribution of future VM requirements and finds the quantiles to find the unwanted machines which can be shut down to save power without affecting the QoS. A cloud administrator sets a probability bound for each VM which defines the probability that a machine will be powered down when a cloud request arrives. This probability is decided on the basis required start-up delay resulting from power savings. Minh Quang Nguyen et al. [14] proposed a simple prediction model for tail latency for a large class of Fork-Join queuing networks and generalized the model for the cases where each request process through a fraction of of processing units in the whole network. Then a link between the system-level request tail latency constraints, (tail-latency SLO) is established. Secondly they proposed an empirical approach for mean latency approximation for above mentioned system based on the developed tail latency prediction model. P. Ramrez-Cobo et al. [**?**] describe a technique for Bayesian estimation related to the two-state stationary Markov arrival process (MAP2), which has been accepted as a versatile model in a number of contexts. The approach is considered for both simulated and real data sets, and the performance of the MAP2 is compared against MMPP2. Further extension on the method is provided by matrix generalization of the M/G/1 queue to estimate the queue length and virtual waiting time distributions of a stationary MAP2/G/1 queueing system that allows for dependent inter-arrival times. Gao Wanlin et al. [4] presented a sliding-window-based model combined with SS-PLS (partial least squares-semi-supervised) specifically designed to learn from smaller data sets for fast adaption. The sliding window based method is used to predict from smaller dataset, which is then combined with PLS and SS learning named as SS-PLS for further improvement. The experimental results for VM load prediction showed that the proposed technique provides significant improvements over auto regression moving average (ARMA). Zhenhuan Gong et al. [5] presented Predictive Elastic reSource Scaling (PRESS) scheme for cloud management. PRESS finds the minute dynamic patterns from application resource demands and then use it for their resource allocations adjustment. The approach contains light weight signal processing and statistical learning algorithms to predict the dynamic application resource requirements. The experimental result shows that their prediction schemes achieve better prediction accuracy than previous approaches also the elastic resource scaling effectively reduces the resource waste and the SLO (service level objective) violations. John J. Prevost et al. [6] introduce a novel framework integrating the load demand prediction and stochastic state transition models for optimal cloud resource allocation. The tradeoff is achieved between energy consumed and performance levels. The neural network and autoregressive linear prediction algorithms are used to predict the loads in cloud data center. Kranthi Manoj Nagothu et al. [7] discussed a to ultra-low power cloud computing systems methods applicable for the systems such as data centers and web hosting. Their analysis indicates that power saving up to 80% is possible when data center components are optimal allocated. The analysis done on the basis of proper arrangement of adaptive load prediction and smart task distribution systems. Sadeka Islam et al. [11] developed a prediction-based resource estimation and provisioning methods using Neural Network and Linear Regression to fulfill upcoming on-demand resource allocation in the cloud.

3. Hidden Markov Model (HMM)

3.1. Markov Models

In the Markov Model the prediction of the next state and its related observation only depends on the current state, or alternatively the state transition probabilities do not depend on the whole history of the past process [?]. This is called a first order Markov process the definition can be generalized for the i^{th} order Markov process as the probability of next state can be calculated by obtaining and taking account of the past i states. For the sequence of random variables $X = (X_I, \ldots, X_r)$ taking values in some finite set $S = s_r, \ldots, s_r$, the state space. Then the Markov Properties are [?]:

$$P(X_{(t+1)} = s_k | X_1, \ldots, X_t) = P(X_{(t+1)} = s_k | X_t), \quad (limitedhozrizon) \tag{1}$$

$$= P(X_2 = s_k | X_1), \qquad (Timeinvariant) \tag{2}$$

Because of the state transition is independent of time, we can have the following state transition matrix A:

$$a_{ij} = P(X_{(t+1)} = s_j | X_t = s_i) \tag{3}$$

a_{ij} is a probability, hence:

$$a_{ij} \geq 0, \forall i, j \sum_{j=1}^{N} a_{i,j} = 1 \tag{4}$$

Also we need to know the probability to start from a certain state, the initial state distribution:

$$\pi_i = P(X_1 = s_i), where, \sum_{i=1}^{N} \pi_i = 1 \tag{5}$$

In a visible Markov model, the states from which the observations are produced and the probabilistic functions are known so we can regard the state sequence as the output.

3.2. Hidden Markov Models

The Hidden Markov Model (HMM) extends the Markov Model for the cases where state knowledge is unavailable or in HMM, one does not know anything about what (system states) generates the observation sequence. The number of states, the transition probabilities, and from which state an observation is generated are all unknown. Hence each state of the system is liked with observation with a probabilistic function instead of deterministic function as in case of Markov Model. At time t, an observation o_t is generated by a probabilistic function $b_j(o_t)$, which is associated with state j, with the probability:

$$b_j(o_t) = P(o_t X_t = j) \tag{6}$$

3.3. Mathematical Terms in HMM

An HMM is composed of a ve-tuple: (S,K,,A,B).

(1). $S = 1, \ldots, N$ is the set of states. The state at time t is denoted s_t.

(2). $K = k_1, \ldots, k_M$ is the output observation and M is the number of observation choices.

(3). Initial state distribution $\Pi = \pi_i, i \in S.\pi_i$ is defined as:

$$\pi_i = P(s_1 = i) \tag{7}$$

(4). State transition probability distribution $A = a_i j,\ i, j \in S$.

$$a_i j = P(s_{t+1}|s_t), 1 \leq i, j \leq N \tag{8}$$

(5). Observation symbol probability distribution $B = b_j(o_t)$. The probabilistic function for each state j is:

$$b_j(o_t) = P(o_t s_t = j) \tag{9}$$

After modeling a problem as an HMM, and assuming that some set of data was generated by the HMM, we are able to calculate the probabilities of the observation sequence and the probable underlying state sequences. Also we can train the model parameters based on the observed data and get a more accurate model. Then use the trained model to predict unseen data.
To generate the HMM model for any system we need to compute three parameters

(1). Observation Sequence Computing: The probability of the observation sequences.

(2). The state sequence $(1, \ldots, N)$ that best explains the observations.

(3). The tuning of the parameters to nd the best model for given observation sequence O, and a space of possible models.

Finding the probability of an observation: Given an observation sequence $O = (o_1, \ldots, o_T)$ and an HMM $= (A, B, \Pi)$, we want to nd out the probability of the sequence $P(O|\mu)$. This process is also known as decoding. Since the observations are independent of each other, the probability of a state sequence $S = (s_1, ..., s_T)$ generating the observation sequence can be calculated as:

$$P(O|\mu) = \sum_S P(O|S, \mu)P(S|\mu) \tag{10}$$

$$= \sum_{s_1, \ldots, s_{T+1}} \pi_{s_1} \prod_{t=1}^{T} a_{s_t s_{t+1}} b_{s_t s_{t+1} o_t} \tag{11}$$

The computation is quite straightforward by summing the observation probabilities for each of the possible state sequence.
Finding the best state sequence: To find the best state sequence given a model and the observation sequence is to choose the states that are individually most likely at each time t. For each time $t, 1 \leq t \leq T + 1$, we find the following probability variable:

$$\gamma_i(t) = P(s_t = i|O, \mu) \tag{12}$$

$$= \frac{P(s_t = i, O|\mu)}{P(O|\mu)} \tag{13}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \tag{14}$$

The individually most likely state sequence S0 can be found as:

$$S' = \underset{x1 \le i \le N}{\arg\max}\, \gamma_i(t), 1 \le t \le T+1, 1 \le i \le N \tag{15}$$

This quantity maximizes the expected number of correct states.

Estimation of parameters: The last and most difficult problem about HMMs is that of the parameter estimation. Given an observation sequence, we want to nd the model parameters $\mu = (A, B, \pi)$ that best explains the observation sequence. The problem can be reformulated as nd the parameters that maximize the following probability:

$$\underset{\mu}{\arg\max}\ P(O|\mu) \tag{16}$$

There is no known analytic method to choose μ to maximize $P(O|\mu)$ but we can use a local maximization algorithm to nd the highest probability. This algorithm is called the Baum-Welch. This is a special case of the Expectation Maximization method. It works iteratively to improve the likelihood of $P(O|\mu)$. This iterative process is called the training of the model. The Baum-Welch algorithm is numerically stable with the likelihood non-decreasing of each iteration. It has linear convergence to a local optima. To work out the optimal model $\mu = (A, B, \pi)$ iteratively, we will need to dene a few intermediate variables. Dene $p_t(i, j), 1 \le t \le T, 1 \le i, j \le N$ as follows:

$$p_t(ij) = P(s_t = i, s_{t+1} = j|O, \mu) \tag{17}$$

$$= \frac{P(s_t = i, s_{t+1} = j|O, \mu)}{P(O|\mu)} \tag{18}$$

$$= \frac{\alpha_i(t)a_{ij}b_{ijo_t}\beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\beta_m(t)} \tag{19}$$

$$= \frac{\alpha_i(t)a_{ij}b_{ijo_t}\beta_j(t+1)}{\sum_{m=1}^{N}\sum_{m=1}^{N} \alpha_m(t)a_{mn}b_{mno_t}\beta_n(t+1)} \tag{20}$$

Then dene $_i(t)$. This is the probability of being at state i at time t, given the observation O and the model :

$$\gamma_i(t) = P(s_t = i|O, \mu) \tag{21}$$

$$= \sum_{j=1}^{N} P(s_t = i, s_{t+1} = j|O, \mu) \tag{22}$$

$$= \sum_{j=1}^{N} p_t(i, j) \tag{23}$$

The above equation holds because $_i(t)$ is the expected number of transition from state i and $p_t(i, j)$ is the expected number of transitions from state i to j. Given the above denitions we begin with an initial model μ and run the training data O through the current model to estimate the expectations of each model parameter. Then we can change the model to maximize the values of the paths that are used. By repeating this process we hope to converge on the optimal values for the model parameters. The re-estimation formulas of the model are:

$$\pi_i' = \text{probability of being at state i at time } t = 1 = {}_i(1) \tag{24}$$

$$a'_{ij} = \frac{\text{expected number of transitions from state i to j}}{\text{expected number of transitions from state i}} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)} \tag{25}$$

$$b'_{ijk} = \frac{\text{expected number of transitions from i to j with k observed}}{\text{expected number of transitions from i to j}} = \frac{\sum_{\substack{t:o_t=k,\\ i\le t\le T}} p_t(i,j)}{\sum_{t=1}^{T} p_t(i,j)} \tag{26}$$

In the following calculations, we will replace b_{ijo_t} with b_{jo_t} , i.e. each observation is generated by the probability density function associated with one state at one time step. This is also called a rst order HMM.

4. Proposed Algorithm

This section gives the detailed explanation of the proposed algorithm. The explanation follows the flow chart shown in figure 1. According to scenario a number of users are connected to a cloud and the requested task lengths as well as requests inter-arrival time for each user follows different distribution functions. Considering the conditions in the scenario the normal HMM based prediction may not work properly because some user may not generate request in each sampling window and some user may generate multiple requests under the same sampling window. Hence we proposed a dual HMM model where one HMM is used to calculate the probability of generating request by any user in the upcoming sampling window and the second HMM for calculating the request pattern that may be generated by any user in the upcoming sampling window. After that these two results are combined to estimate the exact request pattern in the upcoming sampling window.

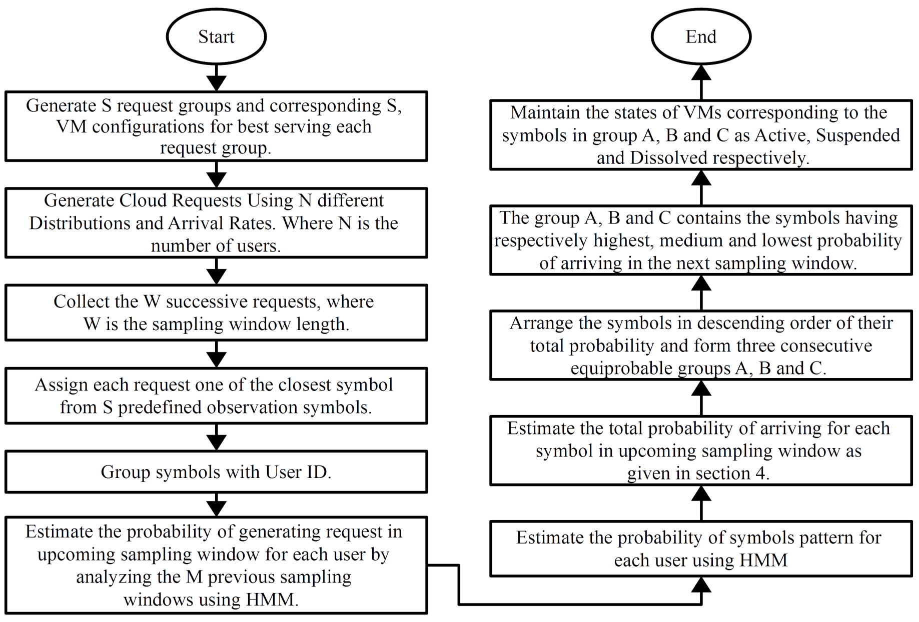
Start
Generate S request groups and corresponding S, VM configurations for best serving each request group.
Generate Cloud Requests Using N different Distributions and Arrival Rates. Where N is the number of users.
Collect the W successive requests, where W is the sampling window length.
Assign each request one of the closest symbol from S predefined observation symbols.
Group symbols with User ID.
Estimate the probability of generating request in upcoming sampling window for each user by analyzing the M previous sampling windows using HMM.
Estimate the probability of symbols pattern for each user using HMM
Estimate the total probability of arriving for each symbol in upcoming sampling window as given in section 4.
Arrange the symbols in descending order of their total probability and form three consecutive equiprobable groups A, B and C.
The group A, B and C contains the symbols having respectively highest, medium and lowest probability of arriving in the next sampling window.
Maintain the states of VMs corresponding to the symbols in group A, B and C as Active, Suspended and Dissolved respectively.
End

Figure 1: The flowchart of proposed algorithm.

Since large number of user requests arrive at the cloud and each request may differ the others, to model such system with HMM a large number of observation symbols are required. The increase in observation symbols will definitely affect the system accuracy and complexity. Hence in the proposed algorithm we created the S requests category and the requests arriving in the cloud is replaced by the category symbol closely related to it. This limits the total number of observations to S. The cloud also have S different VM configurations for best serving each of the S requests category. The length of the sampling window W is the length of sequences of symbols used to predict the upcoming symbols sequences. Let the requests in the cloud for some sampling window length be $R = r_1^a, r_2^b, r_3^c, r_4^b, \ldots, r_W^s$, where $a, b, c, \ldots, s \in N$, $r \in S$ and r_1^a is representing the 1^st entry of request in the sampling window generated by a^th user. The N number of different users given as $U = u_1, u_2, \ldots, u_N$. Let the last L requests generated by user u_i is defined as $s_i = \{r_{-1}^i, r_{-2}^i, \ldots, r_{-L})^i\}$, and the requests by i^th user appeared in M previous sampling windows is defined as $g_i = \{b_{-1}^i, b_{-2}^i, b_{-3}^i, \ldots, b_{-M}^i\}, b0, 1$, the b is a binary variable and b_m^i states that there was at least one request in the m^th previous sampling window generated by i^th user. Now using the equation described in section 3.10 the probability of requesting a specific symbol by a specific user at the next W events and the probability of generating the request by each user in upcoming sampling windows can be calculated. Let the calculated probability of arriving of each request (symbol) in upcoming sampling window by i^th user be $P_i = \{p_1^i, p_2^i, p_3^i, \ldots, p_N^i\}$, and the probability of generating any request (symbol) in the next sampling window by i^th user be t_i. The total probability of appearing a request r_j in the upcoming sampling windows is calculated as

$$P_i^{total} = \sum_{i=1}^{N} t_i p_j * i \tag{27}$$

Hence the total probability of arriving of each request in upcoming sampling window can be given as

$$P^{total} = \{P_1^{total}, P_2^{total}, P_3^{total}, \ldots, P_S^{total}\} \tag{28}$$

Since P^{total} is non-normalized we need to convert it into normalized probability by calculating P_{all}^{total} and diving P^{total} by it as follows

$$P_{all}^{total} = \sum_{i=1}^{S} P_i^{total} \tag{29}$$

$$P_{norm}^{total} = \left\{ \frac{P_1^{total}}{P_{all}^{total}}, \frac{P_2^{total}}{P_{all}^{total}}, \frac{P_3^{total}}{P_{all}^{total}}, \ldots, \frac{P_S^{total}}{P_{all}^{total}} \right\} \tag{30}$$

$$P_{norm}^{total} = \{P_{1,norm}^{total}, P_{2,norm}^{total}, P_{3,norm}^{total}, \ldots, P_{S,norm}^{total}\} \tag{31}$$

Now rearrange the P_{norm}^{total} elements in descending order as follows

$$P_{norm,order}^{total} = \{P_{a,norm}^{total} > P_{b,norm}^{total} > P_{c,norm}^{total}, \ldots, \}, \text{ where } a, b, c, \cdots \in \tag{32}$$

Next we find the two dividing points (D_1, D_2) in between 1 and S for $P_{norm,order}^{total}$ such that $1 < D_1 < D_2 < S$ and $P_{Top} > P_{Medium} > P_{Low}$. The values of P_{Top}, P_{Medium} and P_{low} are calculated as follows

$$P_{Top} = \sum_{i=1}^{D_1} P_{norm,order}^{total}(i) \tag{33}$$

$$P_{Medium} = \sum_{i=D_1+1}^{D_2} P_{norm,order}^{total}(i) \tag{34}$$

$$P_{Low} = \sum_{i=D_2+1}^{D_S} P_{norm,order}^{total}(i) \tag{35}$$

Once the above values are calculated the cloud turns on (or kept them active) the VMs whose configurations are related to requests in P_{Top}. Similarly it suspends and dissolves the VMs whose configurations are related to requests in P_{Medium} and P_{Low} respectively.

5. Simulation Results

The simulation and validation of proposed algorithm is performed using Matlab/Octave. Following configurations are used during the simulation of the algorithm.

Name of Variable	Value
Number of Cloud Users	10
Number of Symbols	10
Length of Sampling Windows	50

Table 1: Showing the simulation environment configuration

User ID	Distribution Name	Parameters
1	Beta	$\alpha = 2, \beta = 2$
2	Binomial	$n = 40, p = 0.5$
3	Cauchy	$x = 20, \gamma = 10$
4	Chi Square	$k = 1$
5	Exponential	$\lambda = 0.5$
6	F Distribution	$d_1 = 100, d_2 = 50$
7	Gamma	$k = 1, \theta = 2$
8	Weibull	$\lambda = 1, k = 5$
9	Lognormal	$\sigma = 1, \mu = 0$
10	Normal	$\sigma = 1, \mu = 0$

Table 2: List of different distribution functions used for request generation by different users.

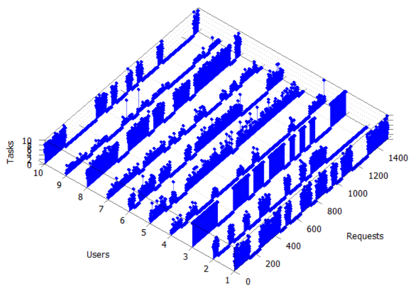

Figure 2: Plot showing the requests generated by each user

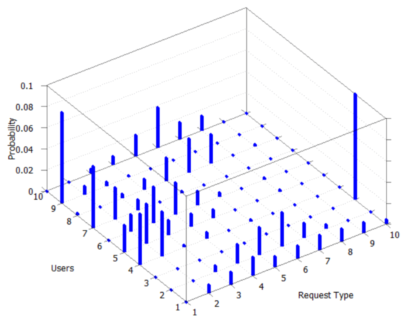

Figure 3: Plot showing the probability of generating different request symbols by each user

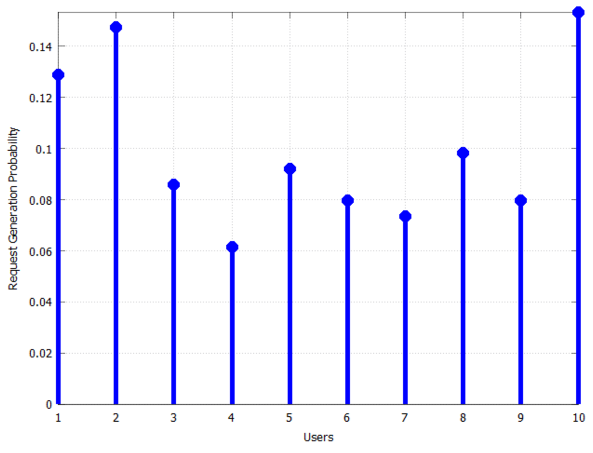

Figure 4: Plot showing the probability of generating any request symbols in upcoming sampling window by each user.

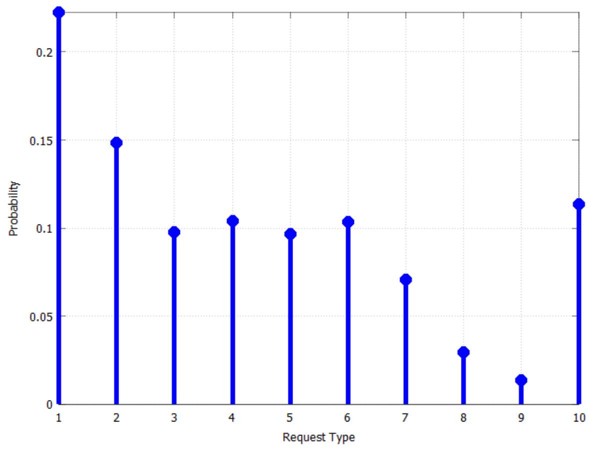

Figure 5: Plot showing total probability of arriving different request symbols in upcoming sampling window.

Total Number of Samples	10	
Sampling Windows Length	Prediction Accuracy (%)	
	Proposed	HMM
10	78.08	68.73
20	82.93	69.34
30	80.52	77.43
40	85.49	75.53
50	86.92	73.52

Table 3: Comparison of prediction accuracy for different length of sampling window.

Sampling Window Length	50	
Total Symbols	Prediction Accuracy (%)	
	Proposed	HMM
10	86.92	73.52
15	80.21	70.45
20	71.83	64.79
25	76.94	74.68
30	68.26	57.53

Table 4: Comparison of prediction accuracy for different values of total number of symbols.

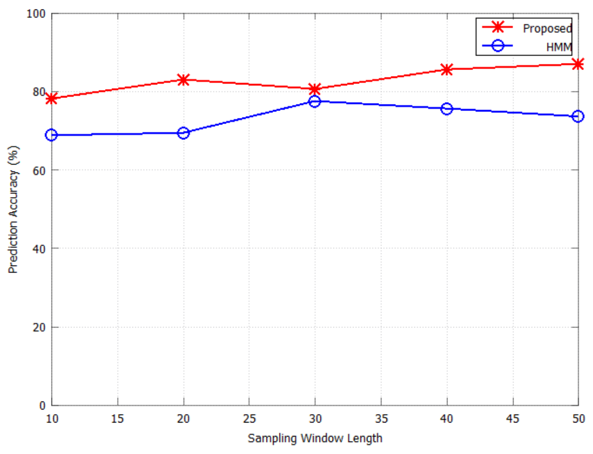

Figure 6: graph for the table 2 data (Comparison of prediction accuracy for different length of sampling window).

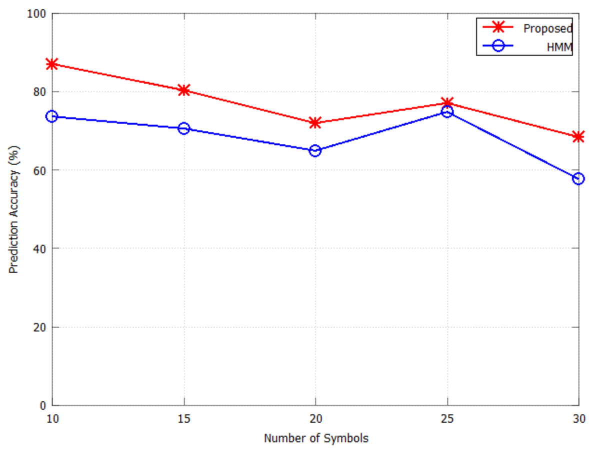

Figure 7: graph for the table 3 data Comparison of prediction accuracy for different values of total number of symbols.

Sampling Window Length				50		
No. of Symbols	Excessive Booting (%)		Excessive Power Consumption (%)		Improvements (%)	
	Proposed	HMM	Proposed	HMM	Proposed	HMM
10	7.55	12.51	6.2	10.94	35.99	41.42
15	10.12	13.42	10.56	14.04	24.78	24.92
20	15.13	18.51	11.88	14.04	18.47	15.46
25	15.27	17.66	14.39	17.83	12.19	19.1
30	16.65	17.62	13.93	18.99	5.25	25.75

Table 5: Comparison of Excessive VM Booting and Power Consumption for different number of Symbols.

Total Number of Symbols				10		
No. of Symbols	Excessive Booting (%)		Excessive Power Consumption (%)		Improvements (%)	
	Proposed	HMM	Proposed	HMM	Proposed	HMM
10	11.13	16.11	9.57	16.9	26.44	41.38
20	6.3	11.73	7.25	11.98	46.02	39.33
30	6.64	11.02	7.44	12.7	41.4	41.65
40	8.84	13.63	8.75	13.47	34.26	33.69
50	7.55	12.51	6.2	10.94	35.99	41.42

Table 6: Comparison of Excessive VM Booting and Power Consumption for different length of sampling window.

Prediction Technique	Symbol Number									
	1	2	3	4	5	6	7	8	9	10
Exact Occurrence	10	10	7	4	5	4	1	0	1	8
Predicted by HMM	11	7	5	5	5	6	4	2	1	5
Predicted by Proposed Algo.	10	10	5	5	5	4	1	5	1	6

Table 7: Comparison of Excessive VM Booting and Power Consumption for different length of sampling window.

6. Conclusion

This paper presented a new approach with Hidden Markov Model (HMM) for cloud load prediction in the complex working conditions. The users requests are predicted for upcoming sampling (time) window length by combining the request (symbol) emission probability of each user using weighted average. The weight in weighted averaging is the normalized probability of transmission of each user in upcoming sampling window. The calculation in such way avoid the complex time consuming procedure of detecting most probable sequence generally done by Viterbi algorithm. Because in the present problem the order of symbols in which they are coming is not important instead we are interested in the probability of arriving in whole sampling window duration. The proposed algorithm is also verified by the simulation results which shows that the proposed approach improves the prediction accuracy by 10% (average) which ultimately improves the unwanted booting of VMs and power consumption by approximately 30% (average) and 40% (average) respectively than HMM.

References

[1] Xiuze Zhou, Fan Lin, Lvqing Yang, Jing Nie, Qian Tan, Wenhua Zeng and Nian Zhang, *Load balancing prediction method ofcloud storage based onanalytic hierarchy process andhybrid hierarchical genetic algorithm*, Springer Plus, 5(2016).

[2] Gabor Kecskemeti, Attila Kertesz and Zsolt Nemeth, *Cloud Workload Prediction by Means of Simulations*, Proceedings of the Computing Frontiers Conference, Siena, Italy, (2017), 279-282.

[3] Jingjing Yuan, Yuanyuan Sun and Mingyue Zhai, *Smart Grid Load Forecasting Cloud Platform Architecture: A Review*, 2016 3rd International Conference on Engineering Technology and Application, (2016).

[4] Gao Wanlin, Hu Hui, Xu Dongbo and Zhang Ganghong, *Virtual machine load prediction model for agricultural cloud video platform based on semi-supervised partial least squares*, Transactions of the Chinese Society of Agricultural Engineering, 33(2017).

[5] Zhenhuan Gong, Xiaohui Gu and John Wilkes, *PRESS: PRedictive Elastic ReSource Scaling for cloud systems*, Network and Service Management (CNSM), 2010 International Conference on 25-29, (2010).

[6] John J.Prevost, KranthiManoj Nagothu, Brian Kelley and Mo Jamshidi, *Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models*, Proc. of the 2011 6th International Conference on System of Systems Engineering, Albuquerque, New Mexico, USA, June 27-30, (2011).

[7] KranthiManoj Nagothu, Brian Kelley, Jeff Prevost and Mo Jamshidi, *Ultra Low Energy Cloud Computing Using Adaptive Load Prediction*, World Automation Congress, (2010).

[8] Zhen Xiao, Senior Member, Weijia Song and Qi Chen, *Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment*, IEEE Transaction On Parallel and Distributed System, 24(6)(2013).

[9] Eyal Zohar, Israel Cidon and Osnat (Ossi) Mokryn, *The Power of Prediction: Cloud Bandwidth and Cost Reduction*, ACM SIGCOMM Computer Communication Review October, (2011).

[10] Ya-Xiu Yu and Xin-Wei Wang, *Web Usage Mining Based on Fuzzy Clustering*, International Forum on Information Technology and Application, IEEE, (2009).

[11] Sadeka Islam, Jacky Keung, Kevin Lee and Anna Liu, *Empirical prediction models for adaptive resource provisioning in the cloud*, Future Generation Computer Systems, 28(2012), 155162.

[12] Rich Wolski and John Brevik, *QPRED: Using Quantile Predictions to Improve Power Usage for Private Clouds*, Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on 25-30 June, (2017).

[13] Minh Quang Nguyen, *Performance Analysis of Scale-Out Workloads on Parallel and Distributed Systems*, the University of Texas at Arlington Doctor of Philosophy August, (2017).

[14] Madhu Jain, Chandra Shekhar and Shalini Shukla, *Queueing Analysis of Machine Repair Problem with Controlled Rates and Working Vacation Under F-Policy*, Proceedings of the National Academy of Sciences, India Section A January, (2016).