The Cloud Resource Forecasting Model Based HMM

Han Liu, Hai-Ning Meng, Tong Zhang, Yi-Lin Qu

Department of Computer Science and Engineering, Xi’an University of technology, Xi’an Shaanxi, China

E-mail: 1418724788@qq.com

Abstract. One of the most important goals for companies that provide cloud computing services is to maintain high availability on large computer systems. In order to accomplish such objective, it is necessary to discovery the reason of poor availability. Software aging is a main factor in cloud computing services, leading to software failures, poor performances and may result in system downtime. This paper investigates the software aging effects on the OpenStack cloud computing platform and describes a forecasting model based on Hidden Markov Models. The prediction analysis of the Hidden Markov Models on the key performance data of the system shows that the Hidden Markov Models has excellent predictive performance and is suitable for the prediction of the reliability of the cloud server system.

1. Introduction
There are two types of noises in the time series problems. One type is system noise, which drives the evolution of random processes and is the inherent nature of the system. The other type is observation noise, which has no effect on the future of the system. In order to fully describe and predict a system that evolves over time, it is necessary to model the internal nature (hidden state) of the system. The Hidden Markov Models (HMM) is a more appropriate choice. The forecasting method based on HMM is a probability domain parameter method [1], its parameter learning process is realized by the EM (Expectation Maximization) algorithm.

As the Hidden Markov Models can fully describe and predict the process of evolution over time, this paper proposes a continuous-time hidden Markov model prediction method (CT-HMM) [2]. Its main feature is that can adaptively calculate the combined weights in various ways within a certain period of time, and can make a better forecasting for non-stationary stochastic processes [3].

Nowadays, many domestic and foreign researchers have done study on server performance analysis, failure predictions and corresponding rejuvenation strategies. Researchers have never stopped studying model-based on time series forecasting.

Reference [2], the statistical learning method , which is edited by Li Hang, describes in great detail about the application scenarios of HMM, the way of solving the problem, and the recursive process of the formula , etc. HMM is mainly used in speech recognition, natural language processing, biological information, pattern recognition and other fields. However, in literature [1], [6], and [11], the analysis is performed on the complete time series observations. Literature [3] proposed a widely applicable algorithm for calculating maximum likelihood of incomplete data, which shows the theory of the likelihood and convergence of the EM algorithm. Literature [4] proposed a software aging statistical modeling method for time-series data of system attributes. The advantage of HMM is that
the estimated model can be directly applied to existing software aging and rejuvenation. Literature [7], the main research results is that studied the quantitative relationship between service availability and system parameters, and described the approximate optimal monitoring interval. Literature [8] proposed a linear regression method to analyze the occurrence of software aging, and also indicated time-series data combine with statistical learning method could be useful.

There are many other methods to monitor software aging and perform rejuvenation on cloud computing platform. Literature [9] compares aging problems appearing in different versions of the same software to detect aging problems, which clearly states software performance declining is caused by software failure, not hardware failure. Literature [15] proposed OS kernel detection technology to measure software aging and showed specific application or system-wide aging detection. Literature [12] proposed failure reports of five major open-source software projects related to cloud computing and confirmed the existence of failures related to aging. Literature [13] studied the availability of virtualized servers and proposed a method called VM-migration to enable Virtual Machine Monitor (VMM) to continue Virtual Machine (VM) execution during failures. The experiment uses a Stochastic Reward Network(SRN) for availability modeling and evaluates the model in a conventional way. The literature [14] described the aging effects on middleware that is caused by memory leaks, and developed a practical algorithm for filtering data to maintain the balance between the aging trend and the rejuvenation time to preserve system maintenance. Literature [15], in this paper on the performance parameters of computer systems, believes that the changes in computer system parameters are mostly random and depend on historical changes.

2. Hidden markov model
The HMM is a parametric random probability model. Its basic ideology is to represent a hidden state (an unobservable internal state) by a discrete random variable \( t \), and \( S \) follows a polynomial distribution. With the random transition probability matrix, the prior probability distribution of the current state depends on the previous hidden state and is unrelated with the past. This is the Markov sexual hypothesis. So, if the state at any time is known, the past, present, and future observation statistics are independent, which meet Markov condition independence.

HMM is a probabilistic model of time series. It describes the random sequence generating process of unobservable states by a Hidden Markov chain. A sequence of states generated by a Hidden Markov Chain is called a state sequence. Each state generates one observation, and the resulting random sequences of observations are called observation sequences. Each value of the sequence corresponds to a time \( t \). Let us describe the hidden Markov model in detail.

The HMM of continuous observations can be described by the following 5 parameters:

1) \( N \) is the number of states in the HMM.
2) \( M \) is the mixed Gaussian probability density function in each state.
3) \( A \) is the state transition probability matrix. \( A = \{ a_{ij} \} \) is the probability of transition from the current state \( i \) to the next state \( j \) in the hidden Markov chain, as follows:

\[
a_{ij} = P[q_{t+1} = j | q_{t} = 1]
\]

\( q_{t} \)—state at time \( t \), and \( a_{ij} \) meet the constraints: \( a_{ij} \geq 0, 1 \leq i, j \leq N \) and \( \sum a_{ij} = 1, 1 \leq i \leq N \)

4) \( B \) is the observation output probability matrix \( B = \{ b_{j}(o) \} \), \( b_{j}(o) \) is the random observation output probability function for state \( j \). The most common random observation output probability function is a linear mixture of finite Gaussian functions.

\[
b_{j}(o) = \sum_{k=1}^{M} c_{jk} G(o, \mu_{jk}, U_{jk})
\]

2
Observation vector $O$ --observation vector, $c_{jk}$ --weight of $J$-th state $k$-th mixed Gaussian function, $G$ --Gaussian distribution with mean $\mu_{jk}$ and covariance $U_{jk}$ in the $J$th state $k$th mixed function. Weight of mixed Gaussian functions $c_{jk}$ meets the constraints: $c_{jk} \geq 0, 1 \leq j \leq N, 1 \leq k \leq M$ and $\sum_{c_{jk}} = 1, 1 \leq i \leq N$

5) $\pi$ is the initial state distribution matrix $\pi = \{\pi_i\}$, and meets the following constraints:

$$b_j(o) = \sum_{k=1}^{M} c_{jk} G(o, \mu_{jk}, U_{jk})$$ (3)

For convenience, HMM can be expressed as $\lambda = (A, B, \pi)$ (4)

3. Forecasting model based on hidden Markov model

Hidden Markov model, as a statistical model, describes the process of randomly generating observation sequences from hidden Markov chains, which belongs to the generative model[4-6]. The performance of the HMM model has a great relationship with the parameters estimation. This paper uses the server system resource time series data as input values to simulate the real environment. Parameter estimation is one of the problems solved by hidden Markov models. Assuming model parameters are known, HMM will continuously modify model parameters $\lambda$ based on the observation sequence. HMM training is a crucial problem in forecasting processing. Compared with the other two problems, this is the most difficult problem. The initial probability distribution of general hidden Markov model’s can take any value, which should meet the constraint $\sum_{i=1}^{N} \pi_i = 1$.

Each $\pi_i$ is randomly generated by Matlab. The transition probability between states is also generated in this way. The concrete analysis is as follows:

![Diagram of HMM generation model](image.png)

**Figure 1. HMM generation model**

3.1. forward-backward algorithm

Given model and an observation sequence calculate the occurrence probability of the observation sequence based the model. There are three methods including the direct calculation algorithm, forward algorithm, and backward algorithm Forward-backward Algorithm.

**Definition 3.1:** (Forward probability) Given the hidden Markov model $\lambda$, define $\{q_1, q_2, ..., q_t\}$ the probability of the observation sequence at time $t$ $q_t$ and the probability of the state as the forward probability, recorded as
Forward algorithm of observation sequence probability:
Input: Hidden Markov model $\lambda$, time series data of cloud server system resources
Output: Observed sequence probability
1) Initial value
\[ \alpha_t(i) = \pi_b(o_i), i = 1, 2, ..., N \] (6)
2) Recursion 1, 2, ..., $T - 1$
\[ \alpha_{t+1}(i) = \frac{a_{ij} \alpha_t(j) b_t(o_{t+1})}{\sum_{k=1}^{N} \alpha_t(k)} \] (7)
\[ b_t(o_{t+1})i = 1, 2, ..., N \] (8)
3) End
\[ \sum_{i=1}^{N} \alpha_t(i) \] (9)

Definition 3.2: (backward algorithm) Given the hidden Markov model $\lambda$, and at time $t$ is probability $q_t$, the posterior probability that the partial observation sequence from $t+1$ to $T$ is defined as the state $\{o_{t+1}, o_{t+2}, ..., o_T\}$, which is written as:
\[ \alpha_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T, i_t = q_t | \lambda) \] (10)

Backward algorithm of observation sequence probability:
Input: Hidden Markov model $\lambda$, time series data of cloud server system resources
Output: Observed sequence probability
1) Initial value
\[ \beta_T(i) = 1, i-1, 2, ..., N \] (11)
2) Recursion $T - 1, T - 2, ..., 1$
\[ \beta_{t+1}(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_t(j) \] (12)
\[ b_j(o_{t+1})i = 1, 2, ..., N \] (13)
3) End
\[ \sum_{i=1}^{N} \pi_t \beta_t(o_i) \beta_t(i) \] (14)

3.2 Baum-Welch algorithm
Known the observation sequences, model parameters need to be estimated. It is similar to many models mentioned before, and a set of training data sets requires to estimate the model. The Baum-Welch algorithm (EM algorithm)\(^{(3)}\) can be used to solve the parameter training problem.
Input: Time series data of cloud server system resources $O$
Output: Hidden Markov Model $\lambda$
1) Initial $n = 0, a_{ij}^{(0)}, b_j^{(0)}, \pi_i^{(0)}$ the model is
\[ \lambda^{(0)} = \left(A^{(0)}, B^{(0)}, \pi^{(0)} \right) \] (15)
2) Recursion $n=1$
\[ a_{ij}^{(n+1)} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \]
\[ b_j^{(n+1)}(k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \gamma_t(k)}{\sum_{t=1}^{T} \gamma_t(j)} \]
\[ \pi_i^{(n+1)} = \frac{\sum_{t=1}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \] (16)
\[ \gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O_t | \lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)} \quad (17) \]

\[ \xi_t(i, j) = \frac{\alpha_t(i) a_i b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_i b_j(o_{t+1}) \beta_{t+1}(j)} \quad (18) \]

3.3. forecasting algorithm

While forecasting, we have the observations before the current time \( t \), and the posterior probability of each state calculated by the above forward-backward algorithm uses all the information of the entire observation sequence, so it can be used as the weight of offline forecast, and for adaptive forecast, new forecast, weights need to be calculated\(^4\). We assume that further prediction for the current time \( t \), we need to consider the posterior probability \( P(S_{t+1} = j | y_t, \theta) \) at time \( t + 1 \).

**Input:** Hidden Markov model \( \lambda \), time series data of cloud server system resources

**Output:** forecasting sequence

1) Initial \[ \delta_t(i) = \pi_i b_i(o_t) \quad i = 1, 2, \ldots, N \] \quad (19)

\[ \psi_1(i) = 1 \quad i = 1, 2, \ldots, N \] \quad (20)

2) Recursion \( t=2,3,\ldots, T \)

\[ \delta_t(i) = \max_{L \leq j \leq N} \delta_{t-1}(j) a_j b_i(o_t) \quad i = 1, 2, \ldots, N \] \quad (21)

\[ \psi_t = \arg \max_{L \leq j \leq N} \delta_{t-1}(j) a_j \] \quad (22)

3) End \[ P = \max_{L \leq i \leq N} \delta_T(i) = \max_{L \leq i \leq N} \delta_T(i) \] \quad (23)

\[ \text{When} \quad t = T - 1, T - 2, \ldots, 1, i_t = \psi_{t+1}(i_{t+1}) \text{, the forecasting sequence is} \quad \{i_1, i_2, \ldots, i_T\}. \]

4. Case analysis

The data used in this paper comes from the instance status records of the OpenStack instance for a total of 10 hours from 21:00 on January 16, 2020 to 07:00 on January 17, 2020. It mainly monitors the instance load through the load instance. The load tool used in the instance is sys-bench, which includes: CPU load (cpu_used), free memory (memory_free), disk load (I/O wait) and other categories of system-related parameters. As shown in the table below:

| Type | Name | Description |
|------|------|-------------|
| 1    | cpu-system | Percentage of CPU consumed by system processes |
| 2    | cpu_idle | CPU idle percentage |
| 3    | cpu_user | Percentage of CPU consumed by user processes |
| 4    | memory_free | Free memory (MB) |
| 5    | memory_used | Used memory (MB) |
| 6    | load_average | System average load |
| 7    | net_peak | Network traffic |

The above parameters can affect the operation of the server system from all levels\(^7\). For instance, the percentage of cpu_system usage is too high, and the response time of the system will also slow down. Memory garbage has not been cleaned up\(^8\). The system temporarily abandoned memory clean...
up due to cpu_system, and memory garbage will increase, the butterfly effect will become more and more serious, and eventually the entire computer system will enter a paralyzed state. In order to cope with the above-mentioned problems, we conduct computer system resource consumption modelling and analysis, and at the same time predict the direction of future resource consumption, and make better use of the powerful computing, storage, and information sharing functions of computer systems.

4.1. Model forecasting
We use the time series data of the disk load as the input observation set. During the HMM model training process, a total of 50 hidden states are set, and the input data is 1799 time series data to fully describe the aging process of the system. The observation set is transmitted to the initialized model, and the Baum-Welch algorithm used for parameter training. The HMM model, which is generated by training, combined with the observation sequence adopts the Viterbi algorithm to predict system aging. The algorithm can predict the change trend of system resources in the next 10 hours, and observing the trend change to determine the time threshold of aging occurrence. Within the threshold range, perform regeneration operations to effectively maintain the high reliability of the system. The parameter training algorithm is as follows:

**Algorithm 1** HMM model training:

```plaintext
Algorithm 1 HMM model training:
for i ←1 to num_Stock do
    Closing(i,:) ←Time series observations;
end for
init ← Initial state probability distribution
u ←Initial hidden state sequence
for i ←1 to num_States do
    transition(i,:) ←Initial hidden state transition probability distribution
for i ←1 to num_Stock do
    Sequence = Closing(i,:);
    for j ←1 to length do
        if j=1 then
            for k←1 to num_States do
                alpha(j,k,i) ←init(k) * prob_Gaussian(Sequence(1),u(k),sigma(k))
            end for
        else
            for k←1 to num_States do
                alpha(j,k,i) ←alpha(j,k,i) = prob_Gaussian(Sequence(j),u(k),sigma(k)) * sum(alpha(j-1,:,i).*...transition(:,k)');
            end for
        end if
    end for
    for i ←1 to num_Stock do
        Sequence = Closing(i,:);
        for j ← length to1 do
            if (j=length) then
                for k←1 to num_States do
                    beta(j,k,i) ←1.0
                end for
            else
                for k←1 to num_States do
                    beta(j,k,i) ←0.0
                end for
            end if
for q ←1 to num_States do
    beta(j,k,i) ←beta(j,k,i) + beta(j+1,q,i)*prob_Gaussian(Sequence(j+1),u(q),...,sigma(q)) * transition(k,q);
    S_beta(j,1,i) ←1.0/max(beta(j,:,i));
    beta(j,:,i) ←Backward transition probability of observed time series
end for
for i ←1 to num_Stock do
```


for j ← 1 to length do
    for k←1 to num_States do
        gamma(j,k,i) ← Initial hidden time series transition probability
    init(k) ← Hidden state initial probability distribution
    for k←1 to num_States do
        for q ← 1 to num_States do
            transition(k,p) ← Hidden state transition probability distribution
        end for
        u (k) ← Hidden state sequence
        for k←1 to num_States do
            sigma(k) ← Hidden time series transition probability
        end for
        beta(j+1,q,i)*prob_Gaussian(Sequence(j+1),u(q),...,sigma(q)) * transition(k,q);
        S_beta(j,1,i) ← 1.0/max(beta(j,:,i));
    end for
end for
for i ←1 to num_Stock do
    post(j,k,i) ← Probability of backward transition of time series observations
end for
for k←1 to num_States do
    u (k) ← Hidden state sequence
    for k←1 to num_States do
        sigma(k) ← Hidden time series transition probability
    end for
end for

The model is represented by 5 parameters \( \lambda(\mu, \sigma, \text{init}, \text{transition}, \text{alpha}) \), where init represents the initial hidden state transition probability, transition is the hidden state transition probability, and alpha describes the state's posterior transition probability. According to the model parameters obtained by the above training process, the forecasting process is as follows:

**Algorithm 2** HMM model forecasting algorithm:

for i ←1 to num_Stock do
    Closing(i,:) ← Time series observations;
    for i ←1 to num_Stock do
        for j ← 1 to length do
            for k←1 to num_States do
                post(j,k,i) ← Probability of backward transition of time series observations
            end for
        end for
        for i ←1 to num_Stock do
            for j ← 2 to length do
                for k←1 to num_States do
                    inner_Sum ← Probability of best predicted sequence
                    Predict(i,j) ← Best forecasting sequence
                    Error(i,j) ← Difference between predicted value and real value
                end for
            end for
        end for
    end for
end for
4.2. Experiment result
This section predicts the performance parameter data of the OpenStack system, and obtains the operating status trend of the system in the next 10 hours. Figure 2 and Figure 3 shows the trend prediction chart of the time series of available memory space and I/O request response time. The blue line is the actual value, and the red line is the predicted value. The available space of the swap area gradually decrease with the operation of the system. When the Swap Memory is about to run out, the system will report an error (out of memory), and even shutdown. In Figure 3, it can be seen that the overall slope of the response time of the I/O request shows a saw-tooth-like upward trend, and the system's response ability is constantly weakening. It can be seen from the above four parameters that the performance of the entire system is in a declining state.

Figure 2. Available memory in swap

Figure 3. I/O request response time

4.3. Evaluation of model
The forecasting performance evaluation uses two methods: Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE). The results are shown in Figure 4. RMSE is more sensitive to abnormal points. If the forecasted value of a certain point is extremely abnormal, it will have a greater impact on the value of RMSE. MAPE not only considers the error between the forecasted value and the real value, but also considers the comparison between the difference and the real value. It can be seen from the map of MAPE that all values fluctuate slightly above the zero point, and the forecasting effect is very good.

![Figure 4](image)

**Figure 4.** Evaluation of model

5. Summary
Software aging have been observed in various open source cloud-oriented software systems [12], virtualization systems [13], and middleware systems [14]. In order to maintain the high availability of the system, it is recommended that the system be regenerated within a safe time when the critical resource usage reaches a threshold.

In this section, an aging modelling prediction scheme based on the HMM OpenStack cloud platform is designed and implemented. And according to the above prediction method, it can be seen that the prediction of the hidden Markov model is very suitable for the computer system, where each parameter depends on history, random nonlinearity, and single-dimensional continuous changes [15].

Acknowledgments
The completion of this thesis is very grateful for the guidance of the teacher Meng and the help of the group members. We also thank the seniors for their research experience in aging analysis and modelling, and provided substantial theoretical support for the experiment.

Reference
[1] Hassan M R and Nath B. 2005 Stock market forecasting using hidden Makov model: a new approach. Proceedings of 5th International Conference on Intelligent Systems Design and Applications, 192-6.
[2] Hang Li. 2012 Statistical learning methods. (Peking: TsingHua University Press) 256.
[3] Dempster , A.P., Liadr , N.M· and Rubin,D.B. 1977. Maximum likehood from incompelte data via EM algoithm. J.Roy.Stat.Soc.B 39 1-38.
[4] H.Okamura, Junjun Zheng, Tadashi Dohi. 2017 A Statistical Framework on Software Aging Modeling with Continuous-Time Hidden Markov Model 978-1-5386-1679-6/17 2017 IEEE DOI 10.1109/SRDS. 24.
[5] Baum, L. 1972 An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov process. *Inequalities* 3-18.

[6] Zhihang Chen. 1999 Combined prediction method based on hidden Markov model. *Signal Processing Branch of Chinese Institute of Electronics, Signal Processing Branch of Chinese Instrumentation Society, Signal and Information Processing Branch of Chinese Aviation Society. Proceedings of the Ninth National Conference on Signal Processing*. 40-3.

[7] Bing Liu. 2018 *Analysis of Application Service Availability of Virtualization System Based on Markov Regeneration Process*. Beijing Jiaotong University.

[8] B. R. Mohan and G. Ram Mohana Reddy, 2014 Software aging trend analysis of server virtualized system. *The International Conference on Information Networking 2014 (ICOIN2014), Phuket*, 260-3.

[9] F. Langner and A. Andrzejak, 2013 Detecting software aging in a cloud computing framework by comparing development versions, 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, 896-9.

[10] R. Matias, I. Beicker, B. Leitao, and P. R. M. Maciel, 2010 “Measuring software aging effects through OS kernel instrumentation,” in IEEE WoSAR, 1–6.

[11] Ran Huang. 2015 *Hidden Markov Model (HMM) -based stock price prediction analysis*. D. Qingdao University.

[12] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, 2012 “Aging-Related Bugs in Cloud Computing Software,” in IEEE WoSAR, 287–92.

[13] F. Machida, D. S. Kim, and K. S. Trivedi, 2013 Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration, *Performance Evaluation*, 70(3) 212 – 30.

[14] G. Carozza, D. Cotroneo, R. Natella, A. Pecchia, and S. Russo, 2010 Memory Leak Analysis of Mission-Critical Middleware, *Journal of Systems and Software*, 83(9) 1556 – 67.

[15] Zihao Lin. 2014 *Research on Performance Parameters and Speed of Computer System*. D. Nanjing University of Posts and Telecommunications.