Analysis of Working Process for Cloud Computing Systems

Lanyang Ji*, Jie Ding*† and Xu Zhao‡
*School of Information Engineering, Yangzhou University, Yangzhou, 225000, China
†State Key Lab. for Novel Software Technology, Nanjing University, Nanjing, 210000, China
‡Beijing Smartchip Microelectronics Technology Company limited, Beijing, 100192, China
† Corresponding author:jieding@yzu.edu.cn

***Abstract*—Recently, cloud computing is being widely concerned by the government and the industry as a result of a significant reduction in the cost of IT and the opportunity to create new business models and resource values. To help cloud providers and consumers understand their system better and improve the quality of service, this paper studies the universal action flow extraction algorithm by using the begin action and the end action of working process for cloud systems so as to analyze it, which is based on system modeling of Petri nets.**

***Keywords*— Cloud Computing, Petri Modeling, Universal Action Flow Extraction Algorithm**

I. INTRODUCTION

Cloud computing is the outcome of traditional technologies in distributed computing, parallel computing, virtualization, .etc. It provides on-demand access to resources, improves the computational efficiency, and reduces the computational cost. Analysis of working process for these cloud computing systems helps cloud providers or consumers understand their systems, so that one can take appropriate measures to maximize the quality of service and another can enjoy the better, too. As a result, the universal action flow extraction algorithm is proposed, which is based on system modeling of Petri nets[1][2].

The rest of this paper is structured as as follows. The cloud system and its corresponding Petri model are introduced in Section II, and the action flow extraction algorithm for Petri model and modeling analysis are presented in Section III. Section IV concludes the research and acknowledges.

II. SYSTEM MODEL

As depicted in Fig.1, the system model here is exemplified by IaaS[3], as there are many IaaS providers in the market, including Ali, Microsoft and Amazon. To facilitate the understanding of the model by providers and consumers, this paper studies the following scenario modeling. A user has applied for three VMs(virtual machines) through the Iaas cloud management system and then deployed applications on the VMs to submit computing requirements[4][5]. After the deployment, the requirements need to be completed and the results are returned to the user. Based on the above, its corresponding Petri model is built up, as is shown in Fig.2.

In Fig.2, p_1 represents a user; p_2,p_3 and p_4 indicate the state that the user has applied for three VMs, deployed applications and submitted the requirements; p_{11},p_{12} and p_{13} represent the state that the three VMs have done the previous operation; p_5,p_6

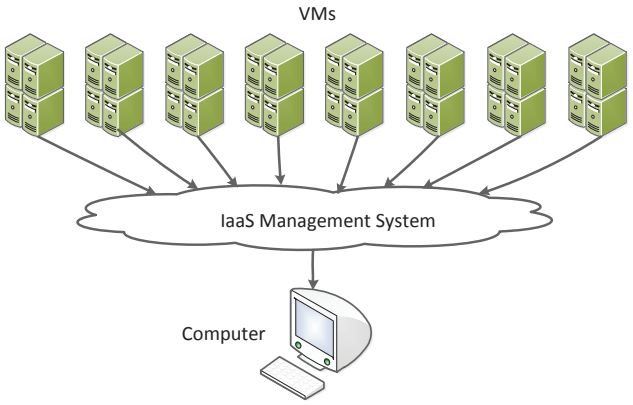

Fig. 1. IaaS Cloud System

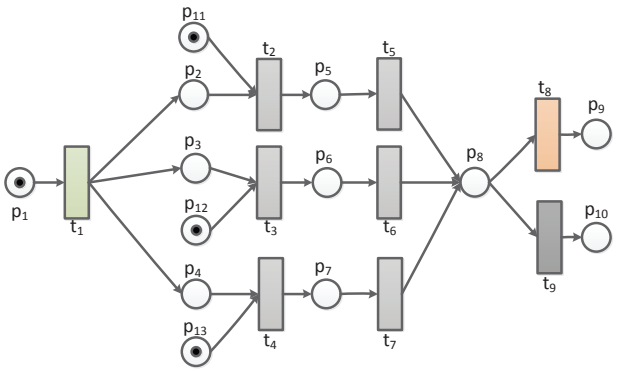

Fig. 2. Petri model

and p_7 indicate the VMs have completed the requirements; p_8 represents the user has received the results sent from the cloud management system; p_9 and p_{10} show whether the results meet expectations; From t_1 to t_9, all the actions in the whole process are represented respectively.

III. ALGORITHM AND MODEL ANALYSIS

A. Algorithm

The Petri net model can be considered as a special directed graph. Therefore, the research of this paper can be transformed into an analysis of a directed graph path. And as considering working process, more attention is paid to the action flow. So

the goal is to convert the Petri model into a directed graph with actions only. In order to be understood and handled easily, the whole algorithm is expanded around the matrix. As follows, the algorithm 1 converts the incidence matrix of the Petri model to the adjacency matrix of a directed graph which only considers the actions, and algorithm 2 extracts all paths from the starting node t_1 to the target node t_8 of a directed graph.

In these two algorithms, $convertToAM(T)$ represents the transformation from the incidence matrix of Petri to the adjacency matrix of a directed graph composed of action nodes. $pushStackAndMarkIn(t_b, t_{st})$ indicates putting t_b into $stack$, marking it and then assigning the top element of $stack$ to t_{st}. $EnQueue(Q_p, stack)$ represents storing $stack$ in the queue Q_p. $popStackAndMarkOut(t_{lst}), peekStack(t_{st})$ indicates deleting the top element of $stack$, mark it and assigning it to t_{lst}, then assigning the current top element to t_{st}. $h(t, \pm 1)$ represents the set of pre-places or post-places of the action t respectively.

Algorithm 1 function of convertToAM

Input: incidence matrix T, empty matrix M
Output: the matrix M, which only considers actions.

for all all t in T **do**
  if $h(t, 1)$ and $h(t_i, -1)$ have an intersection **then**
    $M(t, t_i) = 1$
  end if
end for
return M

Algorithm 2 extract all paths from a directed path

Input: T, starting or target node t_b, t_e, empty $stack$, M, last or current top node t_{lst}, t_{lst} of $stack$
Output: the queue of target paths Q_p

M = $convertToAM(T)$
$pushStackAndMarkIn(t_b, t_{st})$
for all t in M **do**
  if $M(t_{st}, t) = 1$ **then**
    if t doesn't exist in $stack$ (not marked) **then**
      if $t = t_{lst}$ **then**
        Jump out of current loop and execute the next
      end if
      $pushStackAndMarkIn(t, t_{st})$
      if $t = t_e$ **then**
        $EnQueue(Q_p, stack)$, $popStackAndMarkOut(t_{lst})$, $peekStack(t_{st})$
      else
        $t_{st} \rightarrow t$, jump to the third step and continue
      end if
    end if
  else
    if all T in M, $M(t_{st}, t) \neq 1$ and $t \neq t_e$ **then**
      $popStackAndMarkOut(t_{lst}), peekStack(t_{st})$, jump to the third step and continue
    end if
  end if
end for
return Q_p

B. Model Analysis

According to the algorithms proposed above, analysis can be done for the model displayed in Section II. For ease of understanding, simply offer the following analysis process before giving the final results.

- Put the starting action t_1 into $stack$ and mark it, find its first unmarked adjacent node t_2 and put it into $stack$ and mark it too, which is not the ending action t_8 and also not the last top element t_{lst} of $stack$. Then find the first unmarked adjacent node t_5 of t_2 and mark it, which is also not t_8 or t_{lst}. Continue the previous operation until finding t_8. Next, the first target path $t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_8$ is found.
- Take t_8 out of $stack$, assign it to t_{lst} and mark it. Find the first unmarked adjacent node t_9 of the current top element t_{st}(or t_5), which doesn't have any adjacent node and is not t_8. So continue taking the top element t_5 out of $stack$, assigning it to t_{lst} and marking it. Repeat the previous operations until find t_1. Next start to find its another adjacent node t_3...

Repeat the actions above, then three target paths can be found, just as follows.

- $t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_8$
- $t_1 \rightarrow t_3 \rightarrow t_6 \rightarrow t_8$
- $t_1 \rightarrow t_4 \rightarrow t_7 \rightarrow t_8$

Therefore, it's clear that the whole working process of the system begins with the action t_1. Then t_2, t_4, t_4 are executed respectively, and three action flows are formed. At last, performing t_8 indicates the end of the whole service.

Conclusions and Acknowledgements

The results obtained in Section III show that the algorithms can effectively analyze the working process of the cloud computing system and extract the action flows. Based on these, cloud providers and consumers can understand their systems more clearly, and cloud providers can optimize some links of the whole process, so as to provide consumers with better service quality. Additionally, The authors acknowledge the financial support by the National Natural Science Foundation of China under Grant No.61472343.

References

[1] W. Reisig, Understanding Petri nets: modeling techniques, analysis methods, case studies, Springer-Verlag Berlin Heidelberg, 2013.
[2] J. Peterson, Petri Net Theory and the Modeling of Systems, Computer Journal, 1981.
[3] J. Ding, L. Sha, and X. Chen, "Modeling and Evaluating IaaS Cloud Using Performance Evaluation Process Algebra", APCC2016, the 22nd Asia-Pacific Conference on Communications, IEEE Computer Society, 2016, pp.243-247.
[4] cloudstack, https://www.cloudstack.apache.org/.
[5] vmware, https://www.vmware.org/.