A New Interaction Model for Mapping Cloud Flow to Virtual Machine

Tapalina Bhattasali

St. Xavier's College (Autonomous), Kolkata India. tapalina@sxccal.edu

Abstract. Mapping workflow instances to virtual machines in cloud domain is a challenging task. However, it must be considered during design of any cloud application to provide better resource utilization. In this paper, the problem of assigning any workflow in cloud domain to virtual machine (VM) at run-time is addressed. A mapping problem is formulated using linear assignment, where optimized solution of minimum interaction cost is evaluated during dynamic mapping. A new interaction model is designed on the basis of various interaction patterns among cloud entities. Cloud flow vector (workflow in cloud domain) is selected for successful execution of service request. As a proof of concept proposed logic is implemented in the open source modeling tool WoPeD to validate the work. No inconsistency found in mapping procedure of interaction model. Proposed interaction model can efficiently utilize the cloud-resource.

Keywords. mapping; interaction cost; virtual machine; workflow; WoPeD

I. INTRODUCTION

Rapid growth of cloud services has explored a paradigm that can utilize existing pools of resource on demand. Concept of virtualization is one of the most important issues that need to be considered to design cloud services. It is used for better resource management. Infrastructure is generally a pool of virtual machine instances (VMI).Workflow of procedural logic needs to be considered in order to get the idea about the benefits of cloud service. Complexity of design can be reduced by using simple workflow net model. Successful execution of complex workflows for real-time application scenarios considers various constraints like resource, time and priority of tasks. To simplify workflow validation procedure, more focus needs to be given to formulate a workflow net model. Major contribution of this paper is to propose a new interaction model by exploring different interaction patterns arise in various workflow instances and map it to the virtual machine instance at run-time according to minimal interaction cost. Flow vector is determined based on flow constraint, and capacity constraint. Instead of simulating the mapping procedure in any cloud specific simulator, proposed work is validated by using WF-net model [1] to simplify the task.

The rest of the paper is organized as follows. Section 2 briefly presents the literature survey. Section 3 presents different types of workflow in cloud domain. Section 4 introduces the concept of interaction model for controlling the workflow. This interaction model identifies various interaction patterns to estimate overall interaction cost. Then section 5 briefly describes the logic proposed for selecting flow vector. Overall procedure of mapping is logically represented by workflow net model. Section 6 analyzes the proposed work in terms of validation using workflow net model. Section 7 concludes this paper.

II. LITERATURE SURVEY

Researchers have been engaged in number of significant works related to virtual machine (VM) management for resource utilization [2]. Major objectives of VM management considered for performance evaluation include wastage of resource, resource utilization, reliability, total completion time, power consumption. There are several works, which focus on minimal consumption of total energy to reduce expenditure of resource. VM consolidation problem has been addressed in many papers in the literature [3].

The concept of Peer VMs Aggregation (PVA) is used to enable dynamic discovery of communication patterns and reschedule VMs based on the determined communication patterns using VM migration. In this [4], a network and a memory subsystem are modeled on CloudSim simulator. Performance is evaluated based on VMs placement, performance degradation, and network utilization of each link. Results show that it can significantly reduce the total amount of traffic in the network.

The concept of communication virtual machine (CVM)[5] presented in the context of a healthcare application, supports automatic realization of application-specific communication services through a user-centric, model-driven approach. Authors claimed that unique architectural features of CVM allow new components and features to be seamlessly added. CVM can be served as a service framework, which can be developed and incrementally improved by the collaborative efforts of the research community.

In [6], a service framework is developed to facilitate service migration and to design a cost model and decision algorithm to determine the trade-offs between service selection and migration.

 DOI : 10.18231/2454-9150.2018.0926

Beside this, workflow mapping algorithms have been proposed in several literatures [7,8,9,10]. These works focused on selecting a computing resource for every workflow task to minimize the execution time of the entire workflow or to meet the execution deadline pre-defined by the user. However, several challenges arise during mapping. Bandwidth of the link connecting two VMs allocated to execute two adjacent tasks affects the total execution time. If two tasks of workflow are mapped onto different physical servers, it results into high bandwidth consumption in data centers, leading to inefficient utilization of cloud resources. Existing mapping algorithms are no longer applicable for mapping workflow resource request in data centers. It is seen from the literature survey that basic requirement is to find a simple approach that can balance between quality of service and performance overhead on the basis of Service Level Agreement (SLA).

III. TYPES OF WORKFLOW IN CLOUD DOMAIN

Two types of workflow are considered for cloud domain - inflow and outflow.

Inflow- If any workflow moves to the direction of cloud consumer to virtual machine.

Outflow- If any workflow moves to the reverse of inflow, i.e., from virtual machine to cloud consumer.

From cloud consumer perspective, interactions within this environment are categorized into two types- intercom and intracom.

Intracom- If interaction is bounded within the same data center

Intercom- If interaction is extended to different data centers.

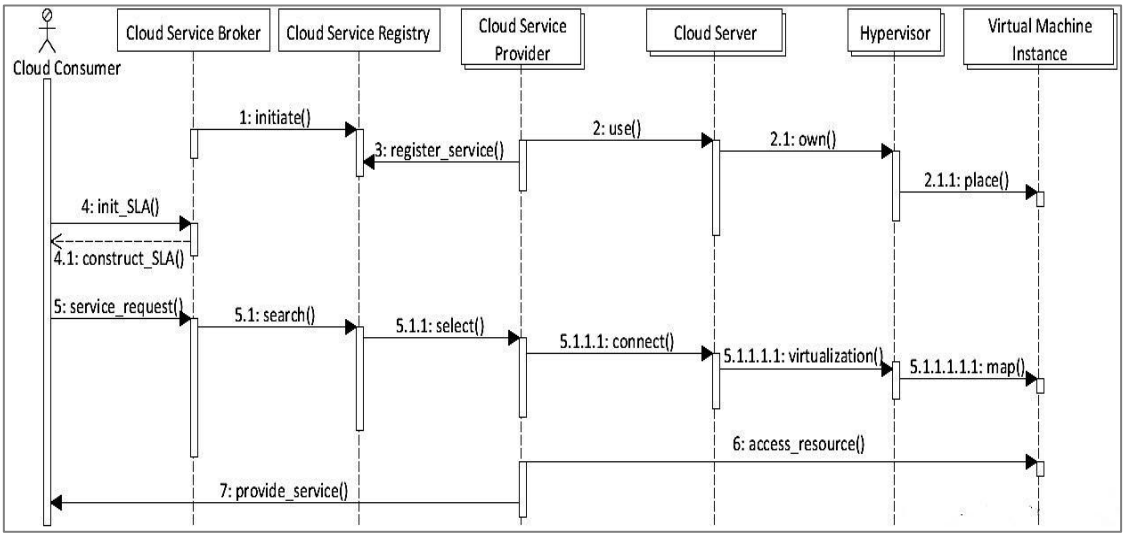

Fig.1 Sequence of Procedural Work Flow among Cloud Entities

Cloud service broker interacts with cloud service provider based on availability of service in registry and requests for services coming from cloud consumer. Cloud consumer interacts with cloud service broker (using init_SLA(), construct_SLA(), service_request())which in turn interacts with cloud service registry (using initiate(), search()) and cloud service providers (using select()). Cloud service provider interacts with cloud service registry using register_service(). Cloud service provider interacts with cloud server using use(), and connect(). Cloud server interacts with hypervisor using own(), and virtualization(), whereas hypervisor interacts with virtual machine instance dynamically using place() and map(). Cloud service provider can interacts with VMI using access_resource() through cloud server and hypervisor. Cloud service provider interacts with cloud consumer through cloud service broker using provide_service().

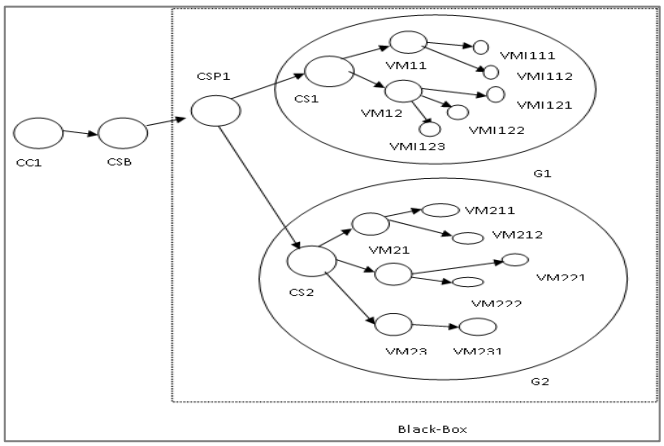

Fig. 2 Mapping Workflow to VM

In figure 2, an instance of mapping between cloud-flow and VMI is presented. A request originates from cloud consumer cc1, who contacts with CSB according to the SLA established before. CSB contacts with CSP1 according to assigned service index mentioned in CSR. CSP1 first selects group G1 (data-center) to assign tasks of workflow vector. Physical resource of CS1 partitioned into two virtual machines VM11 and VM12. VM11 can execute two VMIs

 DOI : 10.18231/2454-9150.2018.0926

namely VMI11 and VMI12. Similarly, VM12 can execute three VMIs namely VMI121, VMI122, VMI123. If tasks in workflow remain incomplete, then CSP1 connects with nearest CS2 based on its capacity and size. Mapping of group G2 (data center 2) is same as G1. Flow vector is selected from all the flow vectors that may arise from source to destination.

IV. INTERACTION MODEL

The mapping between workflow and virtual machine can be formulated as linear assignment problem. VM instances (VMI) are dynamically mapped to find proper VMs and allocate VMIs for each workflow execution. General structure of the problem formulation deals here with the question how to assign p objects of workflow instance to q other objects of VMI in the best possible way. It has objective function to determine "best possible way". The objective of solving this optimization problem is to maximize the performance and minimize the time. A new interaction model is designed for controlling workflow for efficient resource utilization. Figure 3 represents various interaction patterns among the cloud entities.

Interaction Model- A logical model M_{in} that accepts various interaction patterns from the entities in cloud domain and produces interaction matrix (I) as output. It is activated between CC (source) and VMI (sink). This model includes cloud entities, CE = {n × CC, 1× CSB, 1×CSR, m× CSP, p × CS, q × VM, k × VMI}. General format of cloud inflow vector includes **CC_n→ CSB →[CSR]→ CSP_m→ CS_p→ VM_q→ VMI_k** Cloud outflow vector includes **VMI_k→VM_q→CS_p→CSP_m→CSB→CC_n**

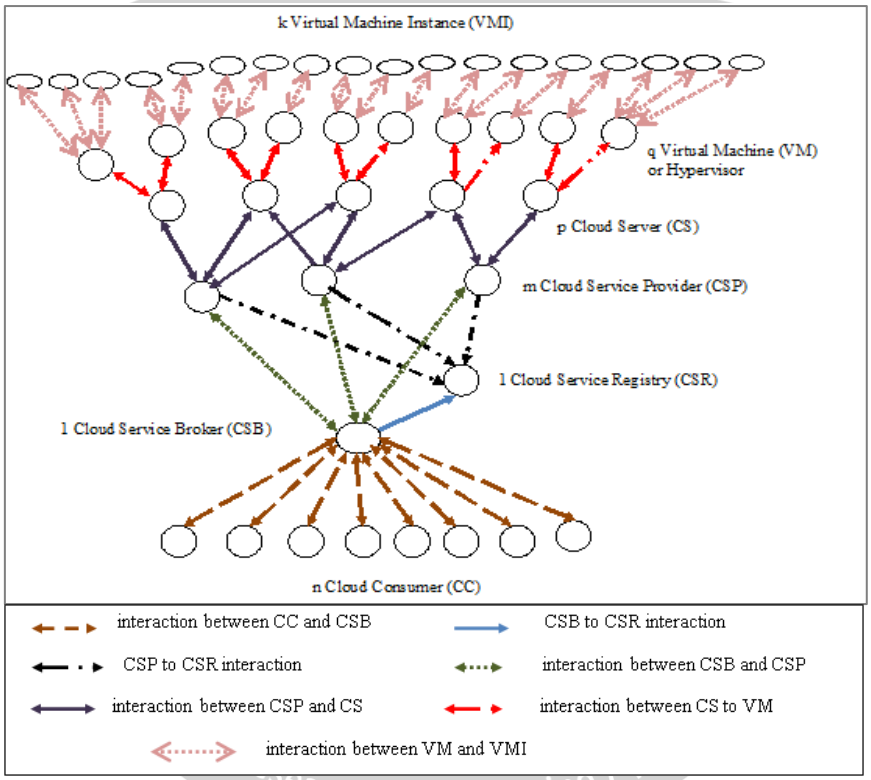

Fig.3 Interaction Model in Cloud Domain

Interaction model generates the following interaction matrix (I).

	CC	CSB	CSR	CSP	CS	VM	VMI
CC	1	n×1	n×1×1	n×1×m	n×1×m×p	n×1×m×p×q	n×1×m×p×q×k
CSB	1×n	1	1×1	1×m	1×m×p	1×m×p×q	1×m×p×q×k
CSR	0	0	1	0	0	0	0
CSP	m×1×n	m×1	m×1	1	m×p	m×p×q	m×p×q×k
CS	p×m×1×n	p×m×1	p×m×1	p×m	1	p×q	p×q×k
VM	q×p×m×1×n	q×p×m×1	q×p×m×1	q×p×m	q×p	1	q×k
VMI	k×q×p×m×1×n	k×q×p×m×1	k×q×p×m×1	k×q×p×m	k×q×p	k×q	1

Interaction overhead is calculated from interaction matrix.

overall interaction= intracom interaction + intercom interaction + inflow interaction + outflow interaction

overall interaction (OI) = {1× 1+1× m + m × p + p × q+ (q × k +k × q)/2+q × p+p × m+m × 1} +{(n × 1 + 1 × n)/2}+{n × 1+1× 1 + 1× m+ m × p + p × q +q × k}+ {1 × n+k × q+q × p+p × m + m × 1}

OI is reduced to {n×1 + 1×1+1×m+ m × p +p × q + q × k} +{k × q +q × p + p × m+ m×1+ 1×n}

 DOI : 10.18231/2454-9150.2018.0926

V. FLOW VECTOR SELECTION LOGIC

Let assume the capacity of each CS is 100%. Another assumption is that, execution requirement for workflow instance must be less than overall capacity of cloud servers. Let VMI running on VM_{ij} (where i=1,2,3,….q) in CS_j (where j=1,2,3,4,….p) is VMI_{ijk}. Major considerable parameters in this domain are – interaction cost based on type of interactions, capacity and size of CS, turnaround time for each interaction, delay between turnaround time and response time and, success rate of cloud flow assignment.

turnaround time = | timestamp when CFI arrives at CS – timestamp when CFI successfully finishes its mapping|

response time = turnaround time + delay

In ideal case, turnaround time = response time

Capacity of VM ($capa_{VM}$) is directly proportional to turnaround time of interaction (tt_{int}).

Bandwidth (BW) is inversely proportional to turnaround time of interaction, BW α 1/ tt_{int}

For each VMI_{ijk}, consider turnaround time for each CF_{ijk}, size and capacity of CS_j.

Interaction cost is the overhead during interaction. It is calculated based on size and capacity of CS and bandwidth of communication path in cloud.

interaction cost = (size × capacity of CS)/ bandwidth involved in communication

Following parameters are determined based on optimal solution of minimum cost.

minimum turnaround time = Min{ i=1,…n, j=1,…m, k=1,….p} (individual turnaround time for interaction)

minimum interaction cost= Min (overall interaction overhead in Cloud)

It is used to determine successful execution of workflow.

```
Input: CS, VM,VMI,   FI, T_slot, group-id
Output: assignment Matrix A, status_ FI
CS⟶ array of available cloud servers
VM ⟶ array of available hypervisors
VMI ⟶ array of virtual machine instances
 FI ⟶        flow instance vector
VMI_ijk⟶ k^th VMI in j^th VM of i^th CS
tt_int(ijk) ⟶ turnaround time for interaction
intracom_cost ⟶interaction cost in intracom
intercom_cost ⟶ interaction cost in intercon
add_intra_cost ⟶ additional intracom cost
T_slot ⟶ upper bound of time slot assigned to  FI
begin
   1. CS ⟶ [CS_size, CS_capa]        // CS_capa← CS capacity
   2. VM⟶[VM_size, VM_capa]      // VM_capa
   3. VMI⟶[ VMI_size, VMI_capa]
   4. timestamp ⟵ get_currenttime
   5. CS[CS_size,CS_capa] ⟵ sort(CS_size, CS_capa)
   6. VM[VM_size, VM_capa] ⟵ sort(VM_size, VM_capa)
   7. VMI[VMI_size, VMI_capa] ⟵ sort(VMI_size, VMI_capa)
   8. for each CS_i as target,
   9. for each VM_j as target,
   10. for each VMI_k as target,
   11.  determine FI
   12.  get_intracom_cost ← min_interaction_cost ( FI,VMI_ijk,VM_ij, CS_i)
   13.  get_response_time ← min_ tt_int(ijk) + interval
   14. if minimum cost(interaction) for VMI_ijk
         a. assign (t, CFI, VMI_ijk) dynamically
         b. construct assignment matrix A
   15. if CS_i, VM_j and VMI_ijk belong to same group G_m then,    // G_m represents same
                                                                   data centre, m=1,2,..,n
         a. add_intra_cost⟵0
      else
         b. find nearest group G' with next higher size and capacity of CS
   16.  calculate add_intra_cost
   17.  add it to overall interaction cost
   18.  compare (response_time, tt_int(ijk))
   19. if response_time > tolerance_delay and intracom_cost> tolerance_overhead then
         a. timeout()
         b. reject CFI for t_slot
         c. break
    end
```

VI. 6 ANALYSIS

Here Workflow Petrinet Designer (WoPeD 3.2.0) [1] is used for workflow net modeling. It is a Java based open-source software that uses Petri Net Markup Language (PNML) to simulate and analyze workflow of Petri net models. In the proposed model, workflow is initiated by executable tasks and workflow instance (WFI) is constructed. WFI moves towards CSB, who searches service index from CSR to select CSP and determines feasible flow vector based on minimal interaction cost and maximum size and capacity of CS. Selected CSP checks capacity of selected CS where virtualization takes place on assigned VMI. CSP checks status of workflow. If it is yet to be finished, nearest CS is selected, when previous CS is overloaded. Remaining tasks in the workflow are assigned to new VMI.

Qualitative analysis of this model by WoPeD tool shows it can satisfy workflow net properties. It is well structured and can satisfy structural feasibility. This model is sound that can satisfy workflow net property, initial marking, boundedness and liveness. Coverability graph shows all the places and transitions of the model can be covered and no two vertices have same marking.

 DOI : 10.18231/2454-9150.2018.0926

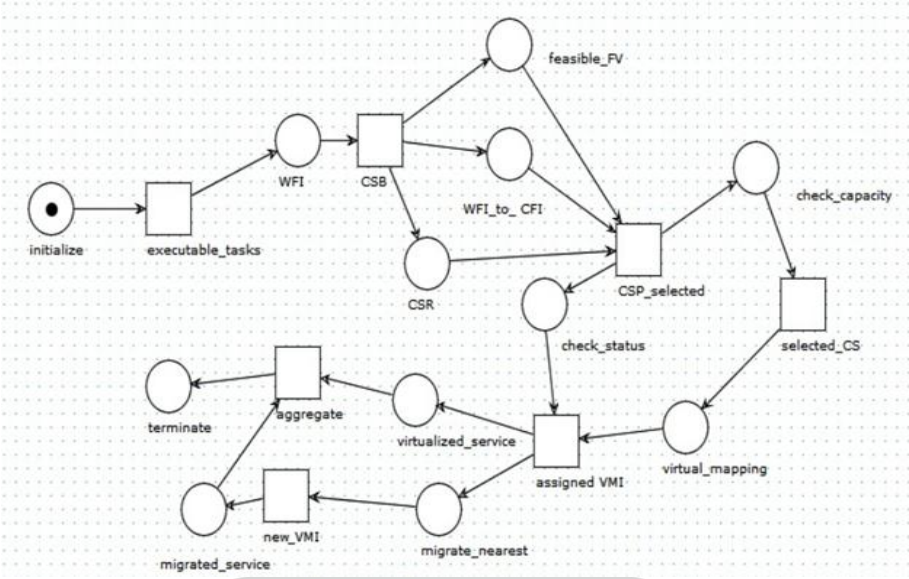

Fig.4 Validation of Interaction Flow in Cloud using WoPeD Tool

VII. CONCLUSIONS

In this paper, the problem of assigning cloud flow to virtual machine at run –time is addressed based on minimum interaction cost. Analysis through workflow net model shows proposed logical model of interaction can successfully map workflow instance to virtual machine instance. The deliverable of this paper is to design a logical model of various interaction patterns among cloud entities and to determine at least one feasible flow vector that can map workflow request to proper virtual machine instance at minimal cost. Finally, proposed logic is modeled as workflow net to ensure sound workflow net property during mapping through interaction model. Proposed interaction model will generate higher revenue for the service providers in cloud domain.

REFERENCES

[1] Workflow Petri net Designer. Available online at: http://woped.dhbw-karlsruhe.de/woped/

[2] Z. Tang, Y. Mo and K. Li, "Dynamic Forecast scheduling algorithm for Virtual Machine Placement in Cloud Computing environment", Journal of Super Computing, Springer, vol. 70, pp.1279-1296 (2014).

[3] X.Fu and C. Zhou, "Virtual Machine selection and placement for Dynamic consolidation in cloud computing environment", Frontier of Computer Science, vol. 9, pp.: 322-330(2015).

[4] Takouna, R. R. Cessa, K. Sachs, C. Meinel," Communication-Aware and Energy-Efficient Scheduling for Parallel Applications in Virtualized Data Centers". In: Proceedings of IEEE/ACM 6th International Conference on Utility and Cloud Computing, pp. 251-255 (2013).

[5] Y. Deng,S. M. Sadjadi ,P. J. Clarke, C. Zhang, V. Hristidis, R. Rangaswami , N. Prabakar , "A communication virtual machine". In: Proceedings of 30th Annual International Conference on Computer Software and Applications Conference, vol. 1, pp. 521 – 531(2006).

[6] W. Hao,I-L. Yen, B. Thuraisingham, "Dynamic Service and Data Migration in the Clouds". In: Proceedings of 33rd Annual IEEE International Computer Software and Applications Conference, pp. 134-139 (2009).

[7] Zareie, M. M. Pedram, M. Kelarestaghi, A. Kosari, "An Approach to Mapping Scientific Workflow in Cloud Computing Data Centers to Minimize Costs of Workflow Execution", International Journal of Computer Technology and Application, vol. 2 (2011).

[8] M. Rahman, S. Venugopal, and R. Buyya, "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids. In e-Science2007, pp. 35-42(2007).

[9] Y. Ahn, Y. Kim, "Auto-scaling of Virtual Resources for Scientific Workflows on Hybrid Clouds", In Science Cloud, pp. 47-52 (2014).

[10] M. Rahman, S. Venugopal, and R. Buyya, "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids. In e-Science2007, pp. 35-42(2007).

 DOI : 10.18231/2454-9150.2018.0926