Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 9 (1): 33-49, 2015
ISSN 1819-4311 / DOI: 10.3923/jse.2015.33.49

Research on Cost-Optimal Algorithm of Multi-QoS Constraints for Task Scheduling in Hybrid-Cloud

[1,2]Liu Yanpei, [1]Li Chunlin, [1]Yang Zhiyong, [3]Chen Yuxuan and [1]Xu Lijun
[1]Department of Computer Science, Wuhan University of Technology, China
[2]Information Engineering College, Henan Institute of Science and Technology, China
[3]Department of Microelectronics and Solid-State Electronics, University of Electronics Science and Technology, China

Corresponding Author: Liu Yanpei, Department of Computer Science, Wuhan University of Technology, China

ABSTRACT

In this study, a job scheduling model and task scheduling algorithm-Cost-Optimal Algorithm of Multi-QoS Constraints for Task Scheduling in Hybrid-Cloud (CAMTH) are proposed to solve the task scheduling problem for hybrid cloud. In the job scheduling model, a hybrid cloud agent which acts as a mediator between clients and public cloud resources is presented to facilitate choosing the candidate service that has the most appropriate price and higher quality level. The proposed CAMTH is divided into three steps, namely task scheduling of private cloud, service selection and task scheduling of public clouds. In addition, CAMTH supports security and reliability under conditions of QoS constraints, in order to select the optimal price public cloud resources on the basis of the eligible conditions. In order to prove the functionality of the proposed algorithm, several experiments are carried out. Experimental result shows that CAMTH algorithm achieves a better balance with waiting time, executing time, QoS satisfaction and execution cost.

Key words: Cloud computing, agent, security, reliability, QoS satisfaction

INTRODUCTION

In recent years, cloud computing (Choi *et al*., 2014) has becomes an extremely important issue. Many existing cloud-service platforms such as Google cloud computing platform (Gonzalez *et al*., 2013), IBM blue cloud (Liu, 2014), Elastic Compute Cloud (Ko *et al*., 2014) and Microsoft cloud computing platform (Doddavula *et al*., 2013) have demonstrated their success to public and provide users a way of pay-per-use service. The cloud environment is simply seperated into private cloud and public cloud. Data center institutions should purchase, maintain, manage and operate all the hardwares and software infrastructures first before they can run an internal data center; however there are still faces some risk of demand-exceeds-supply. Some researchers had been dynamically monitoring hourly workload of the Yahoo Video which is the second largest U.S. online video sharing website in consecutive 46 days and their result shows that peak load is far greater than the average load and also peak load is temporary and unpredictable. If a private data center tries to satisfy all the constraints of workloads, the peak load will force the owners to invest more hardwares resources in private cloud. In that case, hardwares resources will be wasted most of the time. Now these unpredictable workload, peaks can be dealt with the pay-per-use of public cloud while it has not any extra resources in a private cloud and only when public cloud deals with overloaded tasks, additional fee are required while there is no need to add any extra resources to

the private cloud. Therefore, if private cloud already existed by means of building hybrid cloud can avoid the waste of deploying and operating cost. Task scheduling is one of the main challenges of the hybrid cloud.

People have done many research in this relevant field. Lee and Zomaya (2010) proposed a task scheduling algorithm in hybrid cloud environment. The algorithm allocates tasks to cloud resources only for rescheduling. However, the proposed method is to meet the QoS constraints, meanwhile maximizing the use of private cloud and minimizing execution costs of public cloud. Van den Bossche *et al*. (2011) put forward a Cost-Efficient heuristic Scheduling algorithm in the Hybrid Cloud (CESHC) and their scheduling mechanism is the most similar to ours. Their algorithm is to minimize costs under the premise of fitting the deadline constraints but it does not consider the minimum of execution time, safety and reliability of selecting public clouds. Mateescu *et al*. (2011) puts forward a hybrid cloud environments for the application of HPC and the system requests are in a single task, deadline constraints are determined by a single task instead of the entire job but in this study, consideration of the deadline constraints of the entire job occured. Mao *et al*. (2010) propose a resource automatic adjustment mechanism for allocating resources which can meet the deadline constraints, however, this method only consider resources-providing problem but it does not consider to trust QoS constraints and execution cost. Calheiros *et al*. (2012) studies the dynamic configuration technology of hybrid cloud and applies it to the cloud platform of Aneka. However, this method is only applicable to personal job and does not integrate with scheduling, therefore it is not cost-effective when multiple jobs all have deadline constraints.

By summing up the above study, it is not hard to find that the current researches on task scheduling problem in hybrid cloud mainly involve optimizing a certain target, guaranteeing the lowest execution costs, making sure that the minimum of execution time and ensuring the deadline constraint. In order to realize the optimization of task scheduling process in hybrid cloud, current researches on hybrid cloud task scheduling seldom take overall consideration of waiting time, execution time, QoS satisfaction, execution cost and so on. To overcome these limitations, this study presents a new method which considers not only the waiting time, the execution time but also the QoS satisfaction and execution cost. What is more, it also fully considers the elastic function of hybrid clouds which means users can rent or release the public cloud resources according to their need at any time.

The contributions of this study as follows: Firstly, in the job scheduling model of hybrid cloud, this study introduce the hybrid cloud agent which can support client to select the most appropriate price and choose a higher quality level of dedicated service, as intermediary between users and public cloud resources and put forwards a job scheduling model which is based on hybrid cloud agent. Secondly, it makes the deadline constraints, budget constraints, security and reliability under conditions of QoS constraints, in order to select the optimal price public cloud resources on the basis of the eligible conditions and task scheduling algorithm-Cost-Optimal Algorithm of Multi-QoS Constraints for Task Scheduling in Hybrid-Cloud (CAMTH) are proposed. Finally, the function of CAMTH algorithm is evaluated through a large number of experiments.

Figure 1 represents a job scheduling model which introduces a hybrid cloud agent in the hybrid cloud system of multi-cluster public cloud. The modes of system work are generally as follows: using meta-scheduler to receive the user's tasks, selecting appropriate hybrid cloud agency cluster according to cost budget of the task and sending the tasks to the hybrid cloud agent; Hybrid cloud agents purchase a certain amount of resources in a unified price and provides user tasks services in a particular retail price and hybrid cloud agents dynamically adjust its retail price and resources purchasing quantity according to its earnings.

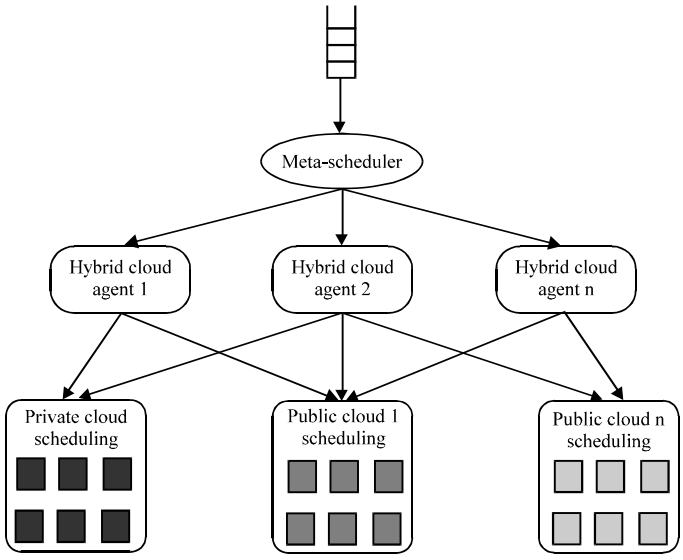

Fig. 1: Job scheduling model based on hybrid cloud agent

The advantages of introducing a hybrid cloud agent in job scheduling model are as follows. Firstly, a hybrid cloud agent can provide an efficient resource-pricing mechanism to improve the system income on the premise of meeting user's cost constraints. Secondly, hybrid cloud agent as a logical entity is equivalent to retailer in client-retailer-producer model and its autonomous transaction behavior in the resource market can help to obtain a stable resource price mechanism. At last, a hybrid cloud agent not only can serve representations for resources of different administrative domains but also can adjust the quantity of its agent resource dynamically. Therefore, a hybrid cloud agent is equivalent to a hybrid cloud resource site with dynamic adjustment ability.

In the proposed job scheduling model which is based on hybrid cloud agent, users submit jobs to the meta-scheduler and each job consists of n tasks. Multiple tasks can be processed by a resource slot which can work as a virtual machine. Then we can define job, task and resource slot as follows.

Definition 1

Job J_i: Job i is defined as J_i, in the proposed job scheduling model which is based on hybrid cloud agent, a job can be composed of n tasks and the n tasks separately refer to $T_{i,1}$, $T_{i,2}$,..., $T_{i,j}$,..., $T_{i,n}$. Among them, $T_{i,j}$ expresses the jth task in the job ith and $1 \leq j \leq n$.

Definition 2

Task $T_{i,j}$: $T_{i,j}$ is a task of the job J_j. Task is the basic unit of the user request, a resource slot can deal with one task at a time. The task can be described by a four-tuple $T_{i,j} = \{D_i, Ws_{i,j}, Ds_{i,j}, M_i\}$.

Among them, D_i represents the deadline and it expresses the deadline of the user to define the job J_i. Each job J_i has a deadline D_i which is defined by users and which is the maximum execution time of the job J_i. Each task in the job J_i ought to be fully implemented and return the results to the users before the deadline. If the job's execution time exceeds the deadline, then it would be said, a violation of the QoS constraints. The $Ws_{i,j}$ is the workload which indicates the workload size of the

task $T_{i,j}$ and which is the task $T_{i,j}$'s execution time by a standard resource slot. The $Ds_{i,j}$ is the data size and it represents the data size of the task $T_{i,j}$. It can affect the time of data transmission. The data is measured by MBs. The M_i is the execution costs which expresses the execution costs in the public cloud of job J_i.

Definition 3

Resource slots: The RS_k indicates the resource slots k and RS_k are created by private cloud or public cloud. A resource slot can be described by a septet $RS_k = \{Cp_k, Cc_k, Sc_k, Dti_k, Dto_k, Nb, Ct_k\}$.

Among them, Cp_k is computing power of resource slot k and it represents resource slot k's computing power of millions instructions per second. The Cc_k is the computing costs of public cloud resource slot k and it represents executing costs of per million instructions in the public cloud. The Sc_k is the storage costs of public cloud resource slot k and it represents storage costs of every MB in the public cloud. The Dti_k is input data costs of resource slot k and it represents the cost of inputting data to resource slot. The Dto_k is output data costs of resource slot k and it represents the cost of outputting data from resource slot. The Nb is the network bandwidth between private cloud and public cloud and it affects time of data transmission. The Ct_k is the cache tasks in the resource slot k, Ct_k contains multiple copies of task and when a job is sent to the private cloud, the tasks in the job will be automatically deployed into the private cloud resource slots.

In the proposed job scheduling model in hybrid cloud, the costs of operating and maintaining in the private cloud are too low to notice, so, this can be set as $Cc_k = 0$, $Sc_k = 0$, $Dti_k = 0$ and $Dto_k = 0$. However, the leases of public cloud resource slots are cost differently which is due to the fact that different hybrid cloud agents have different pricing strategy. If only the costs of operating and maintaining are taken into consideration using public cloud resources is always more expensive than using private ones.

MATERIALS AND METHODS

QoS constraints: Quality of Service (QoS) is a comprehensive index and it can supply different priority, users, data flow or guarantee a certain performance level of data flow for different applications (Niu *et al*., 2011). At times, QoS is used as a quality indicator and it has many alternative definitions. The most common QoS service standard is execution time, usability, security, reliability, price and reputation. This study mainly considers the execution time, price, safety and reliability of QoS. Since, this study involves selection problem of public cloud services, security and reliability are factors that are very necessary to be taken into account. Then, to clarify problem of hybrid cloud task scheduling which satisfy the multiple QoS constraints, the definition of the required QoS parameters is as follows.

Definition 4

Data transfer time Dtt: Data transmission happens when a resource slot k is without data of task T_{ij}. If necessary, this data will be transferred into the resource slot. The transmission time depends on the network bandwidth Nb. Data transfer time is defined as follows:

$$Dtt[i, j, k] = \frac{Ds_{i,j}}{Nb} \quad (1)$$

Definition 5

Estimated finish time Eft_k: The Eft_k represents estimated completion time of resource slot k. And it is decided by the size of remaining running workload in resource slot k. Based on the estimate, prediction of the completion time can be estimated if there is a new task that is used in this resource slot.

Definition 6

Estimated execution time Eet: Estimated execution time is equal to the value of the workload $Ws_{i,j}$ divided by computing power Cp_k of the resource slot k plus data transfer time Dtt. It is necessary to estimate task execution time on different resource slots, allocation of appropriate resources can be occured for jobs which are with specific QoS constraints. And this study define estimated execution time Eet as follows:

$$Eet[i, j, k] = \frac{Ws_{i,j}}{Cp_k} + Dtt[i, j, k] \tag{2}$$

Definition 7

Cost function CostF: In general public cloud service providers such as Amazon elastic computing cloud and Google cloud computing platform, the charged services are computing, storage and data transmission. Therefore, public cloud's execution cost function can be calculated by workload size $Ws_{i,j}$ of computing task $T_{i,j}$ and then figure up the rent price Cc_k of resource slot, storage costs of data size $Ds_{i,j}$, storage costs Sc_k of renting the storage service, transmission costs of data size $Ds_{i,j}$, data input costs Dti_k and data output costs Dto_k. In the private cloud, any resource slot costs are set zero.

$$CostF[i, j, k] = Ws_{i,j} \times Cc_k + Ds_{i,j} \times Sc_k + Ds_{i,j} \times (Dti_k + Dto_k) \tag{3}$$

Definition 8

Deadline constraints: Given a job $J_i = \{T_{i,j}, T_{i,2}, \ldots, T_{i,n}\}$, k resource slot and the deadline D_i, if the complete time of the tasks in the last resource slot is less than or equal to D_i, then this will be called that job J_i can satisfy the deadline constraints. Since, the deadline is only related to resource slots for job computing, setting of two choice variables $a_{i,j,k}$ and b_k occured. Among them, the $a_{i,j,k}$ represents whether task $T_{i,j}$ has been assigned to resource slot k. When $a_{i,j,k}$ equals to one, it means that the task $T_{i,j}$ has been assigned to a resource slot k and when $a_{i,j,k}$ is zero, it means that the task $T_{i,j}$ hasn't. The b_k indicates whether the resource slot is being used and when $b_k = 1$, it means the resource slot k is being used and when $b_k = 0$, k isn't. The deadline constraints are defined as follows:

$$\max_{k=1\sim m}((\sum_{j=1}^{n}(Eet[i, j, k] \times a_{i,j,k}) + Eft_k) \times b_k) \leq D_i \tag{4}$$

Definition 9

Control budget constraints: Given a job such as J_i and the budget M_i is user-defined variable which means execution costs of job J_i in the public cloud. In other words, the cost of the tasks whose workload size is $Ws_{i,j}$ and data size is $Ds_{i,j}$ execution costs in the public cloud resource slots $q \in \{PuR_1, PuR_2, \ldots, PuR_m\}$ is lower than the sum of computing cost and storage costs and cost of data transmission, namely budget M_i. The budget control constraint is defined as follows:

$$\sum_{k=1}^{m}\sum_{j=1}^{n}(\text{CostF}[i, j, k]\times a_{i,j,k}) \le M_i \tag{5}$$

Definition 10

Security: Security levels (Sehgal *et al.*, 2011) are divided into hard, soft and try level and $J_{safe}(i)$ and $PuR_{safe}(j)$ separately represent the security requirements of the job J_i and the security level which is provided by public cloud resources PuR_i. When job J_i is allocated to public cloud resources PuR_i, the security function can be defined as follows:

$$\text{Sec}_{strong}(i, j) = \begin{cases} 1 & J_{sec}(i) \le PuR_{sec}(j) \\ 0 & \text{else} \end{cases} \tag{6}$$

$$\text{Sec}_{soft}(i, j) = \begin{cases} 1 & J_{sec}(i) \le PuR_{sec}(j) \\ 1 - \dfrac{J_{sec}(i) - PuR_{sec}(j)}{\text{max\_level} - PuR_{sec}(j)} & \text{else} \end{cases} \tag{7}$$

$$\text{Sec}_{try}(i, j) = \begin{cases} \dfrac{J_{sec}(i)}{\text{max\_level}} & J_{sec}(i) \le PuR_{sec}(j) \\ 1 & \text{else} \end{cases} \tag{8}$$

Definition 11

Reliable: Reliability levels are also divided into three degrees: Hard, soft and try. The reliability of public cloud resources $PuR_{rel}(i, j)$ is decided by job J_i and resource slot PuR_j. Among them, $J_{rel}(i)$ represents the reliable requirement of job J_i and $J_{rel}(i)\in(0, 1)$. When a Job J_i is assigned to run in public cloud resources PuR_i, its reliability function is defined as follows:

$$\text{Rel}_{strong}(i, j) = \begin{cases} 1 & J_{rel}(i) \le PuR_{rel}(i, j) \\ 0 & \text{else} \end{cases} \tag{9}$$

$$\text{Rel}_{soft}(i, j) = \begin{cases} 1 & J_{rel}(i) \le PuR_{rel}(i, j) \\ \dfrac{\exp(-(1 - J_{rel}(i))) - \exp(-PuR_{rel}(i, j) + 1 - J_{rel}(i))}{\exp(-(1 - J_{rel}(i))) - \exp(-1)} & \text{else} \end{cases} \tag{10}$$

$$\text{Rel}_{try}(i, j) = \begin{cases} \dfrac{1 - \exp(-PuR_{rel}(i, j))}{1 - \exp(-1)} & J_{rel}(i) \le PuR_{rel}(i, j) \\ 1 & \text{else} \end{cases} \tag{11}$$

Assume that at the beginning, the natural failure rate each public cloud resource provider is Fp_j and with the increase of service running time, the failure probability is gradually increased and reliability is gradually reduced. Assume that the job J_i's completion time is $Ct_{i,j}$ when running in the public cloud resource provider PuR_j and reliability of this service is as follows:

$$PuR_{rel}(i, j) = \exp(-Ct_{i,j}\times Fp_j) \tag{12}$$

Definition 12

Trust QoS satisfaction TQS: Job J_i is performed in public cloud resources PuR_j and w_i and w_2 separately represent the different weights of security and reliability of the QoS requirements and among them $w_1+w_2 = 1$ ($0<w_1, w_2<1$). Trust QoS satisfaction can be defined as follows:

$$TQS(i, j) = w_1 \cdot Sec(i, j)+w_2 \cdot Rel(i, j) \quad (13)$$

Dual-objective multi-dimension multi-choice Knapsack problem: Since, the problem is to choose the appropriate resource slot for the task and its main objective is to minimize execution costs of public cloud and the total execution time, therefore, the selection problem of resource can come down to double target multidimensional multiple choice Knapsack problem. To a job J_i, including n tasks and m available resource slots (resource slot $Di>Eft_k$), the Dual-Objective Multi-dimension Multi-choice Knapsack Problem (Srivastava and Bullo, 2014) can be defined as follows:

Definition 13

Dual-objective multi-dimension multi-choice Knapsack problem: Function to minimize the cost (first objective function):

$$\sum_{k=1}^{m}\sum_{j=1}^{n} CostF[i, j, k] \times a_{i,j,k}$$

Function to minimize the estimate execution time (second objective function):

$$\min_{k=1\sim m}((\sum_{j=1}^{n}(Eet[i, j, k] \times a_{i,j,k}) + Eft_k) \times b_k)$$

To make estimate execution time smaller than the deadline:

$$\max_{k=1\sim m}((\sum_{j=1}^{n}(Eet[i, j, k] \times a_{i,j,k}) + Eft_k) \times b_k) \le D_i$$

At the same time, cost function should be lower than the budget cost:

$$\sum_{k=1}^{m}\sum_{j=1}^{n}(CostF[i, j, k] \times a_{i,j,k}) \le M_i$$

among them $a_{i,j,k} \in \{0, 1\}$ is a selected variable and indicates whether the task $T_{i,j}$ is assigned to a resource slot k and $\sum_{k=1}^{m} a_{i,j,k} = 1$, $b_k \in \{0, 1\}$ which expresses whether resource slot is used and $b_k = \cup_{j=1\sim n} a_{i,j,k}$.

Optimization solution of TDO-MMKP: Optimization's main ambition is to minimize the execution costs in public cloud and total execution time. Suppose, there are N object groups and each group has l_i ($1 \le i \le N$) objects. Every object has the consumption of resources and two types of

profit optimization. This pack contains limited available resources. The TDO-MMKP is to choose an object from each group object and place in the backpack accurately which is the first kind of profit maximization and then to make the second kind of profit maximization which is always ought to make the used resources less than available ones.

Because the problem is to allocate resources for each task, scheduling table must be constructed by a hybrid cloud scheduling mechanism to meet the user's QoS constraints and maximize the use of private cloud and meanwhile minimize the cost of using a public cloud. Resource selection problem is mapped to a changing MMKP problems as follows:

- In the TDO-MMKP, each task of a job is mapped to a resource slot and each slot is mapped to an object in a group
- In the TDO-MMKP, QoS constraints of each task is mapped to the resources which is required by object
- The first profit of cost function mapped to the object must be optimized
- The second profit using the resource slot minimum estimated completion time to map to the object must be optimized
- The deadline and budget constraints are regarded as the available resources in the backpack
- Similar to MMKP, TDO-MMKP uses variables to indicate whether an object in a set is chosen
- A better first profits of solution is good. But if the two solutions have the same first profits, the one who has a better second solution is the priority choice

To find the optimal solution for n tasks scheduling with constraints in m resource slot, enumeration of all possible situations occurs by selecting an object (resource slot) from each object group (task) and then calculate the corresponding profits. This solution is only feasible in the case that both n and m relatively are small and the time complexity is $O(n_m)$. Unfortunately, public cloud's resource slot number is very large in the hybrid cloud. Therefore, it is almost impossible to find the optimal solution. So, there should be use heuristic method to solve this problem. Max-Min algorithm is used in this study and time complexity of the improved max-min algorithm is $O(n^2 \cdot m)$, thus it can greatly reduce search time for the optimal solution and realize the binocular target optimization.

CAMTH scheduling algorithm: Hybrid cloud task scheduling algorithm proposed is divided into three parts, task scheduling on private cloud, public cloud service selection and task scheduling on public clouds.

Task scheduling on private: When the job is assigned to resource slot and begins to execute, the task scheduler should allocate resource slots for each task. Based on the definition of 13, the first goal is to minimize the execution cost. Here, in this study suppose that the private cloud implementation cost is zero, so if the private cloud can handle the submitted jobs, the scheduler will focus on the second optimization goal directly to minimize the execution time.

In order to calculate the best scheduling for private cloud task, this study use the TDO-MMKP optimization solution and the improved max-min strategy to choose the biggest workload and the minimum completion time of task. Therefore, this study propose a rapid heuristic algorithm-TSPR which can calculate a good solution without spending too much time.

Pseudo-code of TSPR algorithm pseudo-code

Input: J_i and PrR_i; among them, J_i expresses the submitted job including n task and PrR_i which represents allocated private cloud resource slots and the number of the cloud resource slots is m

Output: A triple <R_i, Z_i, RT>, among them, R_i represents whether it meets the deadline constraint, Z_i is a scheduling table and RT{1...m} is a record of the shortest time of a private resource slot changing from the current to available

- Initialization algorithm Variable, RT{1...m} = 0, $Z_i = \phi$
- For each k∈PrR
- RT[k]←Eft_k
- Endfor
- All the tasks of Job J_i are expressed in descending the amount of data size $DS_{i,j}$
- For each $T_{i,j} \in J_i$ according to the order for validation
- k_min r← the first resource slot//Minimal resource slot is k_min ft
- k_min ft←∞
- For each k∈PrR
- Calculation estimate execution time Eet[i, j, k] for each task $T_{i,j}$
- If Eet[i, j, k]+RT[k]<k_min ft
- k_min ft←Eet[i, j, k]+RT[k]
- k_min r←k
- Endif
- Endfor
- Assign task $T_{i,j}$ to resources k_min r and record the mapping in the Z_i
- RT[k]←k_min ft
- If all the RT are less than deadline constraints D_i
- Return <True, Z_i, RT>
- Else
- Return <False, Z_i, RT>
- End if

Public cloud service selection strategies: If a private cloud can not meet the users' deadline constraints or budget constraints, some tasks need to be allocated to the public cloud. According to the requirements, makes the deadline, budget constraints, security and reliability as the condition of the QoS constraints. The goal of this study is to select the optimal price of public cloud based on the required security, reliability, budget constraints and deadline constraints. Here, this study propose a strategy named QoS comprehensive evaluation.

When the TSPR algorithm output <False, Z_i, RT>, the QoS comprehensive evaluation strategy ought to be executed and use QoS security and reliability assessment algorithm to find public cloud resources which can match J_i security and reliability of the job.

Pseudo-code of QoS comprehensive evaluation algorithm

Input: J_i and PuR_j, among them, J_i expresses submitted job which is include n task and PuR_j expresses public cloud resources

Output: JR, JR is the mapping between the job J_i and safety and reliability of the public cloud resources which is complying with job J_i

- Initialization algorithm variable, JR = ϕ, RT{1...m} = 0
- For each job J_i
- For each public resource PuR_j
- If public cloud resources PuR_j to meet the demand of job J_i's trust
- Computing job J_i's QoS satisfaction TrustQoS(i,j) in the public resource PuR_j
- JR←<J_i, PuR_j)
- Endif
- Endfor
- End for

Task scheduling in public cloud: Public clouds scheduling algorithms (Quarati *et al.*, 2013) are divided into two steps, the first step is task rescheduling; the second step is to minimize the rental cost of public cloud resources.

- **Task rescheduling decision:** When a private resource slot is occupied or cannot meet the deadline constraint, in order to meet the deadline constraint, some of the tasks in the job or a new incoming job may need to be deployed to public cloud but there may be more than one public cloud resources that conforms to the job's safety and reliability and tasks can be specificly chosen to fit the public cloud resources and which one of public cloud resources to be chosen is our key problem. A scheduling table needs to be generated fot the task of the submitting job and the scheduling needs to meet the deadline constrain, the minimum cost and completion time. Here, the task rescheduling technology is used to solve this problem. This study propose four steps of scheduling mechanism to accomplish this goal. Firstly, the system must decide what tasks should be arranged to a public cloud; Secondly, hybrid cloud agent must decide which public cloud resources should be arranged tasks; Thirdly, system need to judge whether public cloud can meet the deadline and budget constraints; Fourthly, if all the constraint conditions can be satisfied, the system should generate a schedule table for task of submitted job; the minimum of the execution costs involved in the last step will be described in the section 2

When a job arrives in a private cloud, the scheduler should allocate private resources for the job and calculate a scheduling table for the job using TSPR algorithm, so that we can decide whether the job can be run in a private cloud. If so, based on TSPR algorithm, the scheduler will directly send tasks and their data to the corresponding private cloud resource slot. Otherwise, the scheduler will invoke QoS comprehensive evaluation algorithm, to choose optimal price public cloud resources on the basis of satisfying the safety, reliability, budget and deadline constraints. Here, there can be multiple choices which meets the requirements of public cloud resources and then continue to invoke TRSD algorithm to decide which tasks should be arranged to public clouds and specific arrangements to which public cloud resources.

TRSD algorithm requires three arguments from TSPR algorithm results and one parameter from QoS comprehensive assessment algorithm.

Pseudo-code of TRSD algorithm

Input: Z_i, PrR_i, RT[1...m], JR; among them Z_i is job J_i scheduling table arranged in the private cloud, PrR_i represents a set of private cloud resource slot and its size is m, RT[1...m] records shortest time of a private resource slot changing from the current to available. JR is the output of the QoS comprehensive evaluation algorithm

Output: A binary $<Z'_i, TP_i>$, where Z'_i is scheduling table of job J_i in the public cloud, TP_i is the task set transferred to the public cloud

- Initialization algorithm variable, $TP_i = \phi$
- For each $k \in PrR_i$
- PR = ϕ// a set of tasks assigned to the private cloud
- PU = ϕ// a set of tasks assigned to the public cloud
- If RT[k]>D_i//task of performing on the private resource slot k is greater than the deadline
- Query task set assigned to the resources of k in Z_i
- Add tasks to PR
- Else
- Continuefor//go to line 2

- Endif
- Get PR according to the size of the Eet[i, j, k] in ascending order
- For each $T_{i,j} \in PR$
- For each JR
- In the cost of public cloud implementation calculate the task among PR
- Select one of the lowest cost from public cloud resource
- Endfor
- Move $T_{i,j}$ from PR to PU
- RT[k]←RT[k]-Eet[i, j, k]
- If RT[k]<D_i
- Breakfor//go to line 24
- Endif
- Endfor
- For each $T_{i,j} \in PU$ according to retrieve in order of increasing Eet[i, j, k]
- If RT[k]+Eet[i, j, k]≥D_i
- task $T_{i,j}$ from PU moved to PR
- RT[k]←RT[k]+Eet[i, j, k]
- Endif
- Endfor
- For each $T_{i,j} \in PU$
- Delete the mapping relationship of resource $T_{i,j}$ in Z_i
- Endfor
- $TP_i = TP_i \cup PU$
- Endfor
- return <Z_i, Tp_i>

- **Minimizing renting cost of public resource:** After assigning tasks to the public cloud resources, this study proposed that the next step of scheduling mechanism is to minimize renting cost and form a public cloud task scheduling table. The problem is scheduling tasks of the public cloud involving public cloud service model. Users can use Pay-Per-Use way to rent any type of resource slot. Resource slot charges according to the resource quality, the total time of CPU, total use of storage space and the total consumption of bandwidth

Inputting algorithm requires three arguments and parameter TP_i is from the results of TRSD algorithm and the other two parameters from scheduling mechanism are for data maintenance.

Pseudo-code of MRCPR algorithm

Input: TP_i, PRT, t_{init}; among them TP_i is tasks set of job J_i running in the public cloud, PRT is the type set of public cloud resource slot and t_{init} is initialization time of the public cloud resource slot

Output: A triple <R_i, Pu_i, Z_i>, where R_i represents whether it meets the deadline constraints, Pu_i represents a set of public cloud resource slot, Z_i represents public cloud task scheduler

- Initialization algorithm variable, $Pu_i = \phi$; $Z_i = \phi$, TotalCost = 0
- For each $T_{i,j} \in Pu_i$
- For each resource type kt in PRT//for the task to find the right resource types
- Calculating cost savings by c of performing task $V_{i,j}$ which on the resource type kt
- Calculating the saving time Eet[i, j, kt]
- Endfor
- Find the lowest cost Eet[i, j, kt]+t_{init}<D_i of resource type kt_{best}
- If kt_{best} exist
- Create an instance k whose resource type is kt_{best}

- Assign task $T_{i,j}$ to a resource k and record mapped in the Z_i
- Add k to Pu_i
- Add the running costs of $T_{i,j}$ on k to TotalCost
- Else
- Return <False, null, null>
- Endif
- Endfor
- If TotalCost≥M_i
- Return <True, Pu_i, Z_i>
- Else
- Return <False, Pu_i, Z_i>
- Endif

RESULTS

Hybrid cloud testing platform configuration: This study deploy the infrastructure in a small system as a validation master plan and use it to perform several sets of experiment. Hybrid cloud is made up of private cloud resources (Table 1) and public cloud resources (Table 2). These two tables show that with the change of capacity (Pr), the number of available resources and the cost of each resource per unit of time each cloud changes. Among them, the trust QoS satisfaction of the public cloud is set as:

$$TQS = 0.25\ (0 \le q \le 1)\ w_1 = 0.5,\ w_2 = 0.5.\ \text{If}\ 0.8 \le TSQ \le 1$$

and it is hardness level; If 0.6≤TSQ≤0.8, it is soft level; Otherwise, it is try level. Assume that a private cloud already exists and it is currently used for processing the user's job request. The private cloud is free and its operating costs are beyond consideration. In addition, resources quantity in private cloud is usually less than the amount of available resources which can be obtained from a public cloud. And each experiment repeated 1000 times.

Experiment 1

Comparison between CAMTH and FIFO: In the first experiment, compared CAMTH algorithm with FIFO scheduling algorithm. Because FIFO scheduling algorithm doesn't take into

Table 1: Private cloud resources and the corresponding processing capacity

Name	Cores	RAM (Gb)	Pr per core
A	2	2.5	1
B	2	4.0	2

Table 2: Public cloud resources, processing capacity and the cost per unit of time

Type	Cores	RAM (Gb)	Pr per core	Price	Trust QoS satisfaction
Y	1	1	1.5	2.5	0.25 q
Y	2	1	1.5	3.5	0.25 q
Z	1	2	2.0	3.0	0.25 q
Z	2	4	2.0	4.0	0.25 q
Z	3	6	2.0	5.0	0.25 q
Z	4	8	2.0	6.0	0.25 q
Z	8	16	2.0	9.0	0.25 q

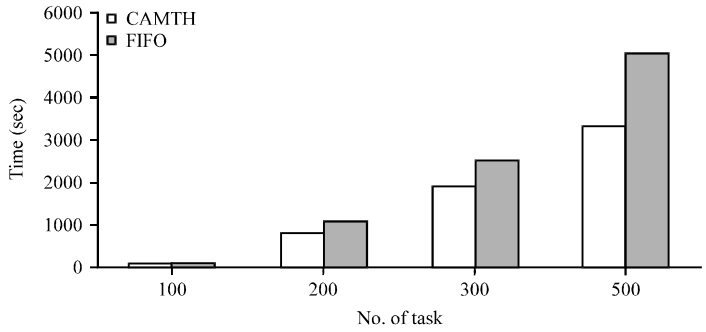

Fig. 2: Waiting time evaluation

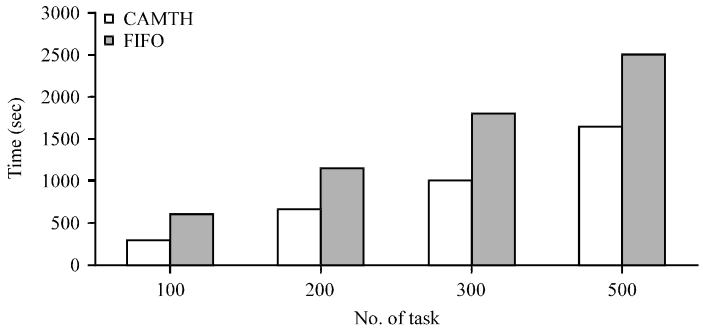

Fig. 3: Execution time evaluation

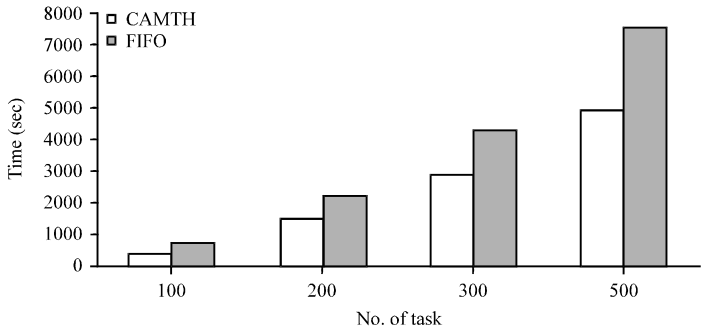

Fig. 4: Completion time evaluation

consideration the public cloud resources as an extension of the private cloud resources, then only compare the performance of task waiting time, task execution time, task completion time and the deadline constraints. With 4 sets of jobs each job has 100, 200, 300 and 500 independent tasks. In simulation, although the number of each job's task is fixed, the job number is not. Each job's deadline is set to the maximum value which indicated that there are no deadline constraints. Note that this experiment does not support public clouds. In the experiment, the task waiting time expresses the time of task from scheduling pool to task start execution and task execution time expresses the time of task from the start to finish and task completion time expresses the time of task from achieve to finish and the experimental results are as shown in Fig. 2-4.

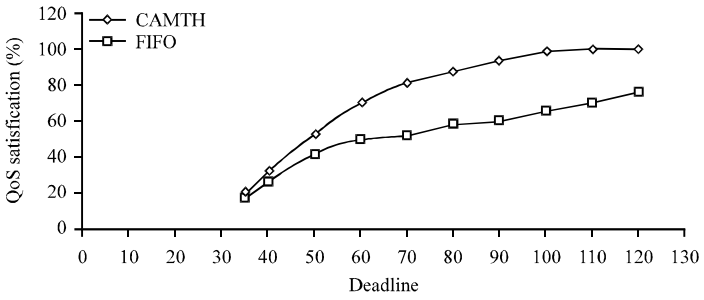

Fig. 5: Compare CAMTH's QoS satisfaction with FIFO (200 tasks)

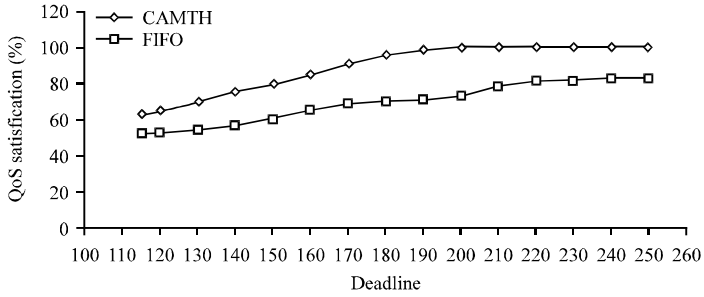

Fig. 6: Compare CAMTH's QoS satisfaction with FIFO(500 tasks)

From Fig. 2-4, it can be seen no matter the waiting time, the execution time or completion time, CAMTH algorithm is much better than the FIFO.

In QoS satisfaction evaluation, the study quantifies QoS satisfaction as complete ratio of the submitted tasks before the deadline.

Figure 5 shows the scheduling job which contains 200 tasks. Given a precise deadline such as 30 sec, the reason why CAMTH and FIFO scheduling mechanism provide a similar and low satisfaction is the fact that the available resources are limited. But if you give a loose deadline such as 80 sec, compared with FIFO, CAMTH can achieve a better QoS satisfaction.

Figure 6 shows the scheduling job which contains 500 tasks. When the deadline is 190 sec, CAMTH can achieve 100% satisfaction but in the case of deadline of 250 sec, FIFO even can't achieve 85% satisfaction.

Experiment 2

Comparison between CAMTH and greedy algorithm: In the second experiment, compare the CAMTH with another algorithm which will be called Greedy Algorithm, those selected resources still come from the private and public cloud. When the private cloud can't meet the demand of users, Greedy scheduling Algorithm chooses resources from public cloud by entirely depending on public cloud performance without regard to cost. Here, set the task execution time (viz., optimal task execution time). The algorithm can be used in private and public cloud. The experimental results are shown as in Fig. 7 and 8.

According to the result figure, the execution time of the Greedy algorithm is lower than MOCOHC algorithm's but CAMTH algorithm execution cost is far below the Greedy algorithm.

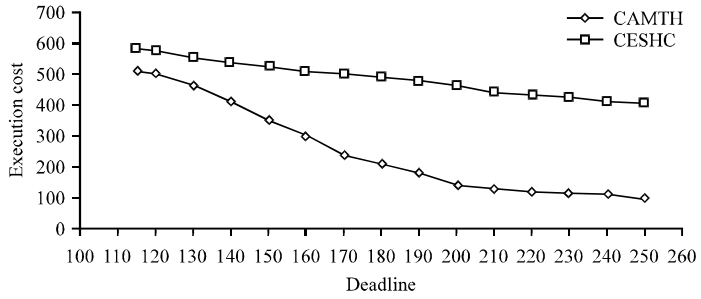

Fig. 7: Compare CAMTH's execution cost with Greedy (500 tasks)

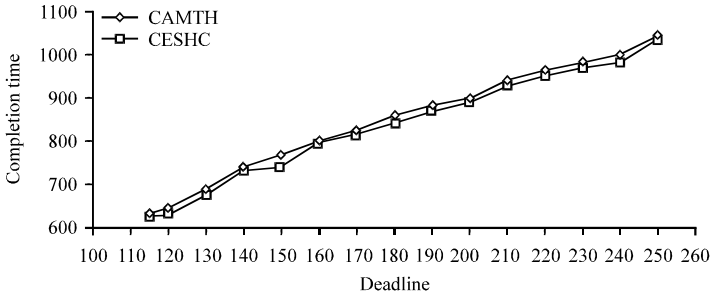

Fig. 8: Compare CAMTH's completion time with Greedy (500 tasks)

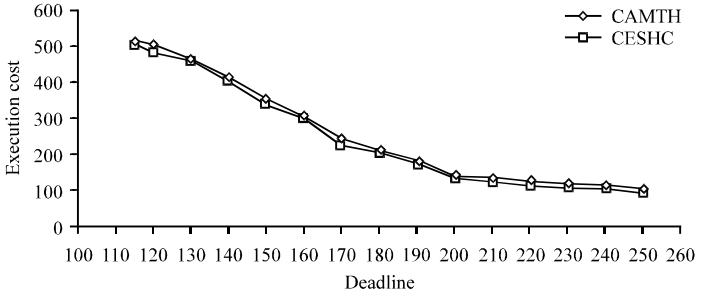

Fig. 9: Compare CAMTH's execution cost with CESHC (500 tasks)

Experiment 3

Comparison between CAMTH and CESHC algorithm: Van den Bossche *et al.* (2011) put forward a kind of efficient heuristic scheduling algorithm (CESHC) applied in a hybrid cloud whose scheduling mechanism is the most similar to our study and their algorithm is to minimize costs under the premise of the deadline constraints but it does not consider minimizing execution time and safety and reliability of public cloud selection. To compare CESHC with CAMTH, some experiments need to evaluate performance for their charges, QoS satisfaction and completion time. The experimental result are as shown in Fig. 9-11.

According to the result figure, the CESHC algorithm's execution cost is lower than MOCOHC algorithm but CESHC algorithm does not consider minimizing the execution time and safety and reliability of public cloud selection. From the experimental result, it can be seen that QoS satisfaction and completion time of the CAMTH algorithm are obviously superior to CESHC algorithm.

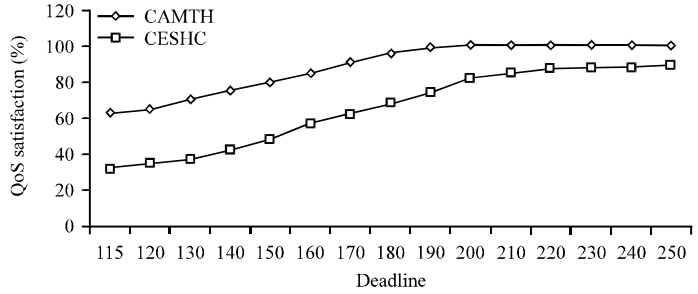

Fig. 10: Compare CAMTH's QoS satisfaction with CESHC (500 tasks)

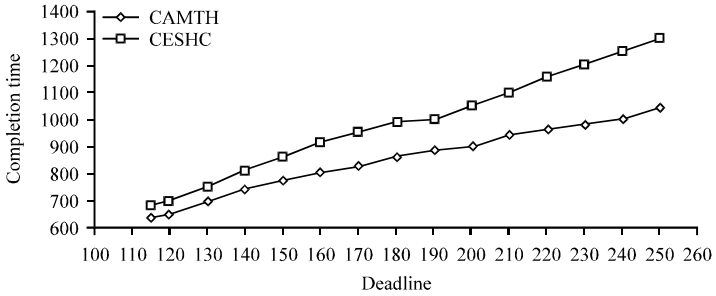

Fig. 11: Compare CAMTH's completion time with CESHC (500 tasks)

CONCLUSION

This experiment mainly studies the task scheduling problem in a hybrid cloud environment. And integratedly consider the task waiting time, execution time, QoS satisfaction, execution cost, etc and put forwards a job scheduling model which is based on hybrid cloud agent and task scheduling algorithm in hybrid cloud-CAMTH. By comparing with the classical FIFO, Greedy and CESHC which is the most similar to CAMTH, the results show that the proposed CAMTH algorithm not only can satisfy the demand of multiple QoS constraints and realize the minimum of scheduling cost in hybrid cloud but also by selecting public cloud services, ensure the task to choose the most suitable public cloud resources and improve the efficiency of the task execution.

The future study is as follows: Firstly, task scheduling algorithm of hybrid cloud proposed in this study is mainly for the independent model but it does not take into account the dependent tasks with precedence relationship and this kind of dependent task scheduling is more complex than the independent task scheduling. Therefore, how to solve this complex task scheduling in hybrid cloud environment will be the next problem to solve. Secondly, as for public cloud resources selection problem, at present, assume that the attribute values of QoS, the service providers and the users give are authentic, yet this assumption is always difficult to guarantee in practice. Therefore, in a hybrid cloud environment, to consider public cloud service selection method of the credibility of QoS of attribute values will be the next topic.

ACKNOWLEDGMENTS

The study was supported by the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation (NSF) under grants (No. 61171075), Special Fund for Fast Sharing of Science Paper in Net Era by CSTD (FSSP) No. 2013014311021, Specialized Research

Fund for the Doctoral Program of Higher Education under Grant No. 20120143110014, Program for the High-end Talents of Hubei Province and the Open Fund of the State Key Laboratory of Software Development Environment (SKLSDE-2013KF). Any opinions, findings and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

REFERENCES

Calheiros, R.N., C. Vecchiola, D. Karunamoorthy and R. Buyya, 2012. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. Future Generation Comput. Syst., 28: 861-870.

Choi, J., C. Choi, B. Ko and P. Kim, 2014. A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment. Soft Comput., 18: 1697-1703.

Doddavula, S.K., I. Agrawal and V. Saxena, 2013. Cloud Computing Solution Patterns: Application and Platform Solutions. In: Cloud Computing, Mahmood, Z. (Ed.). Springer, London, ISBN: 978-1-4471-5106-7, pp: 221-239.

Gonzalez, R., D. Hernandez, F. De la Prieta and A.B. Gil, 2013. +Cloud: An Agent-Based Cloud Computing Platform. In: Distributed Computing and Artificial Intelligence, Omatu, S., J. Neves, J.M.C. Rodriguez, J.F.P. Santana and S.R. Gonzalez (Eds.). Springer, London, ISBN: 978-3-319-00550-8, pp: 377-384.

Ko, Y., M. Jung, Y.S. Han and B. Burgstaller, 2014. A speculative parallel DFA membership test for multicore, SIMD and cloud computing environments. Int. J. Parallel Program., 42: 456-489.

Lee, Y.C. and A.Y. Zomaya, 2010. Rescheduling for reliable job completion with the support of clouds. Future Generation Comput. Syst., 26: 1192-1199.

Liu, Z., 2014. Research on computer network technology based on cloud computing. Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, July 7-10, 2013, Hangzhou, China, pp: 417-424.

Mao, M., J. Li and M. Humphrey, 2010. Cloud auto-scaling with deadline and budget constraints. Proceedings of the 11th IEEE/ACM International Conference on Grid Computing, October 25-28, 2010, Brussels, Belgium, pp: 41-48.

Mateescu, G., W. Gentzsch and C.J. Ribbens, 2011. Hybrid computing-where HPC meets grid and cloud computing. Future Generation Comput. Syst., 27: 440-453.

Niu, J.J., Z.D. Deng and C. Li, 2011. Distributed scheduling approaches in wireless sensor network. Acta Automatica Sinica, 37: 517-528.

Quarati, A., A. Clematis, A. Galizia and D. D'Agostino, 2013. Hybrid Clouds brokering: Business opportunities, QoS and energy-saving issues. Simul. Modell. Pract. Theor., 39: 121-134.

Sehgal, N.K., S. Sohoni, Y. Xiong, D. Fritz, W. Mulia and J.M. Acken, 2011. A cross section of the issues and research activities related to both information security and cloud computing. IETE Tech. Rev., 28: 279-291.

Srivastava, V. and F. Bullo, 2014. Knapsack problems with sigmoid utilities: Approximation algorithms via hybrid optimization. Eur. J. Oper. Res., 236: 488-498.

Van den Bossche, R., K. Vanmechelen and J. Broeckhove, 2011. Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science, November 29-December 1, 2011, Athens, Greece, pp: 320-327.