Bo CHENG

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications

Hierarchical Cloud Service Workflow Scheduling Optimization Schema Using Heuristic Generic Algorithm

***Abstract**. The rapid growth of visualization technologies and Cloud computing have opened a new paradigm for utilizing the existing resource pools for on-demand and scalable computing, which enables the workflow management system to meet quality-of-service (QoS) requirements of the applications. It becomes crucial for cloud customers to choose the best Cloud services in order to minimize the running costs, and how to match and select the optimum cloud service will be a challenge problem. In this paper, we present an efficient Cloud services workflow scheduling and optimization schema using heuristic generic algorithm, and focus on the hierarchical Cloud service workflow scheduling, Cloud workflow tasks parallel split, syntax and semantic based Cloud workflow tasks matching algorithm, and multiple QoS constraints based Cloud workflow scheduling and optimization, and also presents the experiment conducted to evaluate the efficiency of our algorithm.*

***Streszczenie.** Rozwój technik wizualizacji I wykorzystanie chmury w obliczeniach komputerowych otworzyło problem jak wykorzystać zasoby danych do obliczeń skalowalnych i na życzenie. Dla klientów chmury jest bardzo ważne wybór jak najlepszego serwisu dla minimalizacji kosztów. W artykule zaprezentowano skuteczne planowanie przepływu serwisów w chmurze przy użyciu algorytmu genetycznego. (**Optymalizacja planowania przepływu hierarchicznego serwisów w chmurze przy wykorzystaniu algorytmu genetycznego**).*

Keywords: Cloud service, Workflow, Scheduling, Optimization, Heuristic generic algorithm.
Słowa kluczowe: chmura obliczeniowa, optymalizacja, algorytm genetyczny.

Introduction

Cloud computing is emerging as the new paradigm for the next-generation distributed computing. It has attracted a lot of attention in both academia and industry [1]. The rapid growth of visualization technologies and Cloud computing have opened a new paradigm for utilizing the existing resource pools for on-demand and scalable computing. The cloud computing offer a new computing model where resources such as networking infrastructures, computing capability, storage and software and can be shared as 'services' over the internet in the form of virtualized resources, which reduces the costs while increasing processing throughput and decreasing processing time for the business services [2]. In the future, the number of cloud provisioning offers is currently growing and so is the number of possibilities for running an application in the clouds, there will be many services available from the service clouds. From a business model point of view, there are four types of cloud, and that is Infrastructure as a Service (IaaS), Hardware as a Service (Haas), Platform as a Service (PaaS), and Software as a Service (SaaS). With the increasing demand for process automation in the cloud, especially for large-scale collaborative and distributed e-business and e-science applications, such as climate forecast, high-energy physics, structural biology and chemistry, require the creation of a collaborative workflow infrastructure as part of their sophisticated problem solving processes[3]. These scientific workflows usually need to process huge amount of data and computationally intensive activities to achieve a certain goals.

The Cloud workflow applications often consist a several tasks, and each task is implemented by several substitute Cloud services. QoS and service price will be necessary and play more important roles in Cloud service workflow applications [4]. In order to efficiently and cost effectively schedule the tasks and data of applications among cloud services, end user QoS-based scheduling schemas should be implemented, such as those for minimizing total execution cost and balancing the load of resources. Traditional workflow management systems should be to adapt to this new paradigm in order to leverage the benefits of Cloud services. The Cloud computing paradigm enables the workflow management system to meet the QoS requirements of the applications. The research on cloud workflow scheduling schema is becoming a significant issue not only in the area of cloud workflow systems but also general cloud computing [5]. It becomes crucial for Cloud customers to choose the best one in order to minimize their running costs, and how to match and schedule the optimum Cloud service will be a challenge problem, which is just our motivation and contribution to this research.

In this paper, we focus on minimizing the execution time and the execution cost of workflow applications on these Cloud service resources. The rest of the paper is organized as follows: Section 2 presents hierarchical cloud workflow scheduling schema and related scheduling algorithms. In Section 3, we present an experimental evaluation of the performance with the help of an example workflow. Section 4 is the conclusions.

Hierarchical Cloud Workflow Scheduling Schema

The Cloud workflow system can coordinate multiple job submissions over cloud services. The goal of Cloud workflow scheduling schema is to make sure the proper activities are executed by the right service at the right time. We proposed a hierarchical scheduling schema in cloud workflow systems as Fig.1 shows.

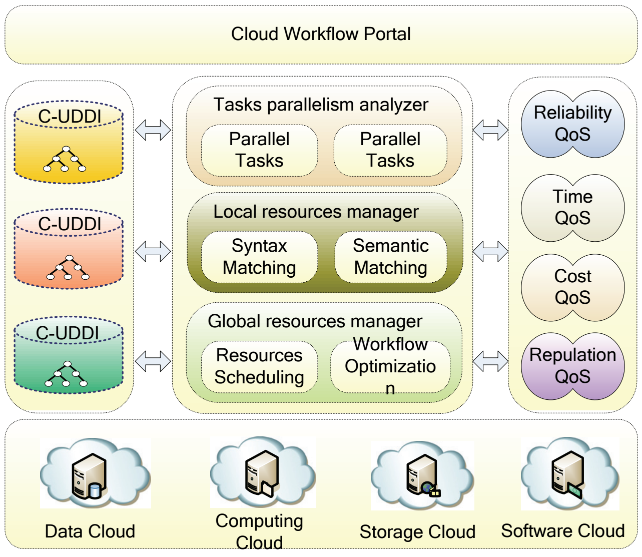

Fig.1. Cloud workflow scheduling schema

It is observed from Fig.1, the Cloud workflow scheduling schema framework is composed of Cloud service registry, task parallelism analyzer, local resource manager, global resource manager, and QoS model. The more details is described as follows.

Cloud service registry, when the cloud service provider registers the cloud service to the registry with the ontology description language, and the cloud service registry provides a semantic cloud service information management for cloud workflow based on semantic service discovery support.

Task parallelism analyzer, which is to parse the tasks and corresponding interdependent relations from the XML based cloud workflow description coming through the cloud service portal, and then to split the parallel task groups according to the maximum degree of parallelism split algorithm.

Local resources manager, which is to find the candidate cloud service group with regard to the tasks requirement in the cloud workflow with a given task with syntax and semantic matching algorithm, and performed for every tasks in the abstract cloud workflow. The local service matching proceeds by selecting the best services for every abstract task individually in the Cloud workflow.

Global resources manager, which ensures meeting global Qos constraints since it selects the optimal tasks execution path which respects global QoS constraints imposed by the customers.

Specifically, our Cloud workflow scheduling schema involves three hierarchical stages, the first stage is to split the parallel tasks, and the second stage is match the canditated services for the corresponding tasks, and the third stage is to schedule the Cloud services. Here, a cloud workflow is specified as a collection of tasks according to a combination of control flow and data flow. The cloud workflow specifications that contains the task definitions, process structures, and Qos constraints. In the first stage, the tasks can be split into parallel task groups according to the maximum degree of parallelism split algorithm. In the second stage, the global resources manager assigns workflow tasks to candidate services with syntax and semantic matching algorithm. In the third stage, we present a heuristic generic algorithm which can generate an optimized scheduling path which satisfied global QoS constraints.

Cloud Workflow Task Splitting Algorithm

A cloud workflow application can commonly be modeled as a Directed Acyclic Graph, denoted ted as a graph whose nodes are the tasks and whose edges are the precedence constraints. We denote an application workflow as a Directed Acyclic Graph represented by G=(V, E), where V ={T_1, ..., T_n} is the set of tasks, and E represents the data dependencies between these tasks, that is, $f_{j,k}$ =(T_j, T_k) $\in$ E is the data produced by T_j and consumed by T_k. Fig.2 represents a cloud workflow topology.

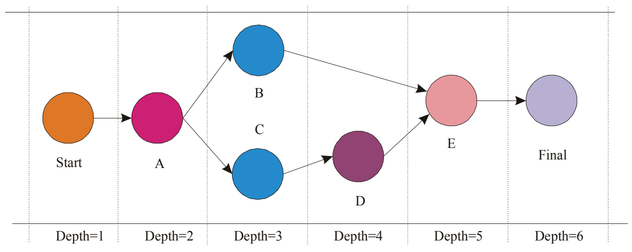

Fig.2. The parallelism split of Cloud workflow

In order to improve workflow efficiency for parallel execution of operations, it is important to focus on the depth of task in the workflow topology definition, and $P_d(T)$ denotes the direct predecessor of the task T, and $D(T)$ denotes the depth degree, if $P_d(T)=\Phi$, and then $D(T)$=1; if $P_d(T) \neq \Phi$, and then $D(T)=\max\{d(R)\}$+1, where $R \in P_d(T)$.

The parallelism split of the cloud workflow task is to form a series of tasks group with corresponding to the each job orderly. A workflow can have one or more parallel split schema to form the corresponding parallelism split set Θ, and I(Θ) denotes the number of the tasks group Θ, and σ denotes the maximum parallel split numbers, if $\sigma \in \Theta$ and exist $\forall\ \delta \in \Theta$, and $(\sigma) \leqslant (\delta)$, and we can split the multiple tasks according to the appropriate depth, and to form a largest degree of parallel split. The largest degree of the parallelism split of Fig.2 can be denoted σ ={Start, A, (B, C), D, E, Final}, and hence, the task C needs to be completed before beginning the task D, and the task E can begin after the task B and D finished.

Cloud Workflow Tasks Matching algorithm

The cloud services are describe by the extensible OWL-S ontology language [6], and the profile file is the direct related main parts, which has eight main properties, such as the serviceName, cloudServiceMetrics, textDescription, input, output, requestedBy and providedBy. Among those properties, the serviceName describe the name for the cloud services, and textDescription describe the information for the cloud service, and cloud service metrics are used to describe the QoS metrics for the cloud service. We can use the provideBy property to assign the Cloud service provider information, and also can use the requestedBy to assign the identity for the cloud service finder. The extensive top level ontology for the cloud service is shown as Fig.3.

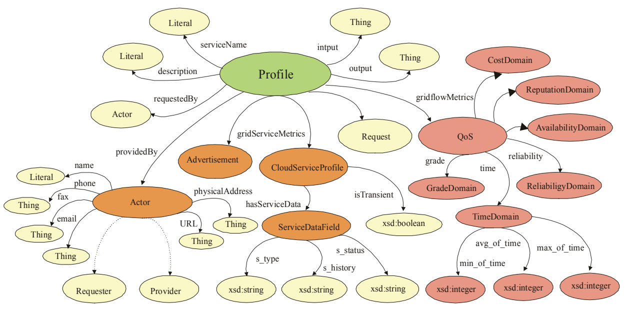

Fig.3. Top level ontology for the cloud service

Especially, the cloud service profile is a new involved class to describe the cloud service capability, which has two main properties, that is hasServiceData property and isTransient property, and the isTransient is Boolean property, and is used to represent whether is to create a transient cloud service. If the value is true, and means to create a transient cloud service instance, otherwise, to destroy a cloud service instance. The ServiceDataField class contains three main properties, and that is s_type, s_history and s_status, and means the data types, history record, and statue for the cloud service.

Cloud workflow matching is to assign suitable service to each task of individual workflow instances according to their functional and non-functional QoS requirements. Usually there are two matching methods for the given cloud service description and the user request information. One is syntax based text matching method, and the other is semantics based logical reasoning method. In the service discovery phase, to sort those service to form a candidate set of cloud services the degree of similarity. To increase the accuracy of the matching service, we consider the syntax and semantics method to find the corresponding cloud services in the registry. The syntax is calculated based on the service name and service descriptions with following equation.

$$sim = \frac{\lambda_a sim_{SN}(S_r.name, S_a.name) + \lambda_b sim_{SD}(S_r.des, S_a.des)}{\lambda_a + \lambda_b}$$

Where, the sim_{match} (S_r, S_a) function is used to calculate the syntax similarity between the request S_r and the service provider S_a , and which is calculated based on the cloud service name and the description, and that is the $sim_{SN}(S_r.name, S_a.name)$ function and the $sim_{SD}(S_r.des, S_a.des)$ function , and the corresponding weight value as λ_a and λ_b , especially, the λ_a and λ_b belong to the real value between 0 and 1.

Semantic-based logical reasoning method is borrowed from the field of Semantic Web methods and techniques through the use of domain ontology concepts and properties to describe attributes of the service. Here, function-based semantic similarity matching of cloud service includes the semantic similarity matching of input and output. The main rationality behind our algorithm is that an advertisement satisfies a request when the advertisement provides all the functions of the request's service. To the semantic cloud services matching application requirements, we propose a simple and effective method of calculating the semantic similarity and semantic distance.

Here, Cloud service discovery is to be achieved by matching cloud service between the requests and cloud service registry with syntax and semantic, and then calculated their similarity. Find cloud services that best match with a given workflow task request, and syntactic similarity value exceeding a given value Sim**0**, Algorithm pseudo code is shown below.

```
Procedure Match (Task T, $Sim_0$)
    List res;
    Int typeOfMatch;
    Filters ={Subsumed-by, Re-Subsumed} ;
    For $S \in CSet_{input}$ (Inputs)$\wedge$ $S \in CSet_{output}$(Output)
        {
        typeOfMatch$\leftarrow$ MIN($CSet_{input}$, $CSet_{output}$)
        if typeOfMatch$\geq$ minMode$\wedge$
          (typeOfMatch$\in$Filters$\vee$ $Sim_{match}$(T, S)$\geq$ $Sim_0$)
          {
            res= res$\cup$ {(S,typeOfMatch,$Sim_{match}$(T,S) )}
          }
        return res
```

Here, the result for the cloud service matching is indentified, and if the output request is equivalent to the output for the service provider, and the result value is 1.0, and which can decrease to 0 with different degree of similarity. Especially, the input matching is similar to the output matching process, and finally returns cloud services set to the requester.

Cloud Workflow Scheduling Algorithm

Cloud Workflow scheduling is the mapping from abstract service to the concrete Cloud service, and then to select an optimal services path to perform tasks to satisfy customer's QoS requirements. As already mentioned, QoS attributes can be either negative or positive, thus some QoS values need to be minimized whereas other values have to be maximized. To cope with this issue, the scheduling phase normalizes every QoS attribute value by transforming it into a value between 0 and 1 with respect to the formulas in reference [7]. We can also define the corresponding untility function of the cloud service instance for the tasks execution path of the cloud workflow.

$$F_{it} = \sum_{j=1}^{k} w_j \bullet q_{ij}^t \, , \; w_j \in [0,1]$$

Where w_j denotes the utility proportion for the QoS attribute j, which value is assigned by the users and applications. Here, the utility function only represents the contribution for the sub cloud service over the execution path for the cloud workflow, and not represents the overall QoS requirements, such as the total cost, total response time, and etc. Hence, it is necessary to verify the overall QoS performance of the execution path for the cloud workflow that satisfied the user requirement. Therefore, the Coud workflow scheduling optimization is an important multi-objective optimization problem, and the cloud workflow execution path optimization can be described with respect to the formulas below.

$$MAX \sum_{i=1}^{m} \sum_{j \in S_i} F_{ij} x_{ij}$$

Where, F_{ij} denotes the overall utility proportion for the cloud service instance j in the service candidate S_i. Now we must choose a global optimization algorithm well adapted to resolve this problem. Here, the heuristic generic algorithm [8] is applied to resolve the Cloud workflow scheduling and optimization problem, heuristic generic algorithm is a search technique often employed to find the exact or approximate solutions to optimisation and search problems. The construction of a genetic algorithm for the Cloud workflow scheduling problem contains four main steps. The first is the choice of representation of individual in the population. Then second is the design of genetic operators. The third one is the determination of the fitness function, and the last one is the determination of probabilities controlling the genetic operators.

We first need to encode the problem with a suitable genome. In our case, the genome or individual is encoded by an integer array of n elements, $X_1, X_2, ..., X_n$, and the values of X_i range from 1 to m_i, where n is the number of tasks in the workflow and m is the number of concrete cloud services for each of the task X_i.

The evolution is made by means of two operators: the crossover operator and the mutation operator. Here, the crossover operator creates new individuals from the existing ones by interchanging resources among them. We have used crossover operator to show good performance for Cloud workflow scheduling schema. The mutation operator randomly selects an abstract cloud workflow task and randomly replaces the corresponding cloud service with another one among those available.

Fitness function is used to evaluate the fitness of chromosomes to problem, and the fitness function for the cloud workflow scheduling algorithm should be created to comply with the following conditions: ① To keep the highest total utility for quality of service of cloud workflow scheduling. ② The workflow execution path for the tasks are connected from the beginning to the end node. ③ The cloud workflow scheduling execution path is effective. Here, the execution path is effective when each sub-task can be executed, followed by the each task to meet the input and output conditions, such as QoS parameters, and a task can be executed only its pre-conditions are satisfied. Especially, the following equation gives the definition of the fitness function.

$$F = w_t \times F_t + w_c \times F_c + w_r \times F_r$$

Where, w_t , w_c and w_r is the normalized weight factor for the response time, cost and reliability, and F_t , F_c and F_r represents the QoS fitness for the response time, cost and reliability respectively.

Experiment and Evaluation

Our experiments are conducted between the proposed approach and other existed approaches. The experiments are conducted on on a laptop with a Intel(R) Core(TM)2 Duo CPU, 1.58 GHz, 2 GB RAM, running Windows XP SP2. During the experiments, we aimed to compare the performances of proposed approach and Shortest Job on the Fastest Resource (SJFR) algorithm [9] and Exhaustive Search algorithm (ESA) [10], we computed the time to optimize the fitness function on the cost and time. As a case study for comparison, we considered a cloud workflow containing 12 invocations of 5 distinct abstract task services as Fig.4 shows. For each abstract workflow task, we considered a variable number of available concrete cloud services, i.e., 10, 20, 30, 40, 50, 60.

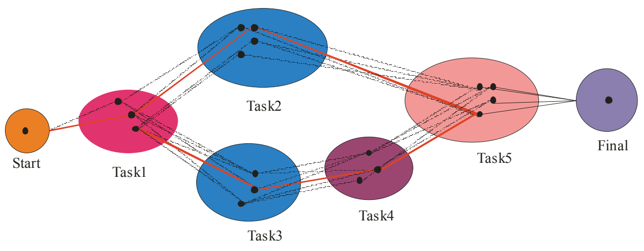

Fig.4. Optimal execution paths of cloud workflow

Especially, the code length is taken as 30, and the population size is taken as 60, and the crossover probability is taken as 0.90, and the mutation probability is taken as 0.03, also the weight parameters as w_t , w_c and w_r is taken as 0.2, 0.3 and 0.5 respectively in the proposed heuristic genetic algorithm. Both those approaches were executed 50 times, and then the average values were computed, and the standard deviation was less than 5% of the mean value. Then, the performance comparison result is shown as Fig.5.

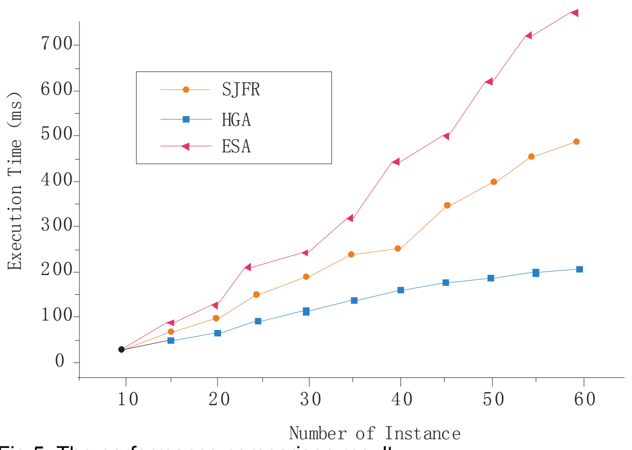

Fig.5. The performance comparison result

It is observed from Fig.5, the efficiency of the proposed heuristic generic algorithm has obvious advantages than the other two algorithms when the numbers of the tasks for candidate services group increases, while the performances of those approaches tend to be the same when the numbers for the tasks is about 10 concrete services. The experiments result also shows that the execution time for the genetic algorithm based cloud workflow tasks scheduling for cloud workflow task is less than the SJFR algorithm and ESA algorithm. To the SJFR algorithm, although the cloud workflow tasks can be shorter for the overall job execution time, but will lead to longer time for some special task execution. Finally, using the exhaustive search algorithm for all execution paths is far greater than the others among those approaches.

Conclusions

In this paper, we present an efficient Cloud services workflow scheduling and optimization schema using heuristic generic algorithm, and focus on the hierarchical Cloud service workflow scheduling, Cloud workflow tasks parallel split, syntax and semantic based Cloud workflow tasks matching algorithm, and multiple QoS constraints based Cloud workflow scheduling and optimization, and also presents the experiments conducted to evaluate the efficiency of our algorithm.

Acknowledgment

This research was supported by National Natural Science Foundation of China (Grant No. 61001118); "973" program of National Basic Research Program of China (Grant No. 2011CB302704).

REFERENCES

[1] Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25(6), 599–616.
[2] Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan , Yun Yang. A market-oriented hierarchical scheduling strategy in cloud workflow systems. The Journal of Supercomputing, 2011, 59(1), 1-38.
[3] Zhen Ye, Xiaofang Zhou, and Athman Bouguettaya. Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing. In Proceedings of Database Systems for Advanced Applications, 2011, 321–334.
[4] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, Maria Luisa Villani., An Approach for QoS-aware Service Composition based on Genetic Algorithms. In Proceedings of Genetic and Evolutionary Computation Conference,2005.
[5] Z. Wu, Z. Ni, L. Gu, X. Liu, A Revised Discrete Particle Swarm Optimisation for Cloud Workflow Scheduling. In Proceedings of 2010 International Conference on Computational Intelligence and Security, 2010,184-188.
[6] OWL-S. DAML Services, http://www.daml.org/services/owl-s/.
[7] Nebil Ben Mabrouk, Sandrine Beauche, Elena Kuznetsova, Nikolaos Georgantas, and Val‼erie Issarny. QoS-aware Service Composition in Dynamic Service Oriented Environment. Lecture Notes in Computer Science, 2009, 123-142
[8]Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Program, 2006, (3), 217–230.
[9] Abraham A, Buyya R. Nature's heuristics for scheduling jobs on computational grids. In Proceedings of 8th International Conference on Advanced Computing and Communication, 2000, 1-12.
[10]Zeng L Z, Benatalah B, Dumas M. Quality driven web services composition. In Proceedings of the 12th International Conference on World Wide Web, 2003, 411-421

***Authors**:Cheng Bo, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications; Beijing, China, E-mail: chengbo@bupt.edu.cn.*