A New Method to QoS Global Optimal Service Selection Driven by Credit

Xiaohua Guo [a], Jing Jiang [b,] * and Xuefei Li

School of information engineering, Qingdao University, Qingdao 266071, China

* Corresponding author

[a]flower_nice@126.com, [b]jj@qdu.edu.cn

Keywords: Credit level, MILP, Global optimization, Service selection.

Abstract. Aiming at the current Web service selection methods do not take into full consideration the effect of service component's dynamic on the performance of service-oriented application system, a credit-aware per-service method to service selection is proposed in our paper, which introduces credit level into service selection model and is considered as Mixed Integer Linear Programming (MILP) problem. Composite system at run time can constrain service selection according to the high-availability of service components perceived by itself and the service level requirements of users, and then dynamically bind the best of a set of services. Simulation experimental results show that the new approach can effectively lower the extra time overhead caused by repeated selection that arises from the service component's dynamic, and then enhance the QoS (Quality of Service) of composite system. Meanwhile, our method is superior to the other traditional Web service selection approaches because it improves the efficiency of CPU usage.

1. Introduction

Service-oriented complicated application systems realize its abstract business process by calling a series of existing services. In the face of a large number of services that have equal functional but different nonfunctional (QoS) properties, how to rapidly and accurately select services which can better satisfy user requirements is always one of the tough challenging tasks.

At the moment, a major trend is to design service-oriented complicated application systems for reducing its complexity as runtime self-adaptable systems, which dynamically select at run time the most appropriate of a set of services to meet user needs. Service selection policy, which has been widely investigated by a lot of researchers in recent years, is the main work of self-adaptation service selection mechanism. There are two principal methods for Web service selection: local optimization and global optimization. The former is that each task at runtime is bound with the most optimal service that implements the task; nevertheless, it can only ensure that the local QoS constrains are satisfied. While the latter can do its best to make sure the global QoS constrains for the whole execution process rather than those single tasks. The main reason for implementing global constraints is that the QoS of Web service changes relatively frequent, which results from the internal changes or heavy load. In some case, for example, the business process which works too long or some services unavailable, it is necessary to perform global optimization at the time of running business process instances for ensuring the global QoS constraints.

The QoS of self-adaptation systems in two main directions: the high-efficiency and the high-availability. The former means service performance requirements, such as the response time, the accuracy of returned results, and so on; while the latter refers to that service is always accessible across its whole life-cycle. However, no matter the local optimal method or the global optimal method, they mainly concern how to implement the high-efficiency of service, rather than the high-availability. In fact, the failure or offline of any one service component composed for self-adaptation system may eventually make the system unavailable and then it needs to select services and schedule tasks again, which can delay response time.

To solve this problem, a new method to QoS global optimization service selection driven by credit is proposed in this paper. In our method, credit record is introduced into the service selection model to

perceive service availability and then is used to drive service selection, and service selection optimal problem is modeled as MILP. Therefore, the best of a set of services are selected according to the service level requirements of users and the high-availability of service components perceived by system and then are bound with the corresponding task dynamically.

In Section 2, related works are discussed. Section 3 introduces the credit driven service selection model. Section 4 proposes the formulation of the optimization problem and Section 5 demonstrates the validity of the proposed method by a series of simulation experiments. Finally, conclusions.

2. Related Works

Milanovic et al. identify four key service-composition requirements: nonfunctional qualities, connectivity, correctness and scalability, and then compare various solutions with respect to their four requirements [1]. But QoS parameters are not extensible in their paper. Other QoS attributes can be added without effort in our method.

CHUNG-WEI HANG et al. present two distributed trust-aware service selection methods: the one on the basis of Bayesian networks and the other one by using Beta-mixture model [2]. But the local information of its running environment is needed to maintain or monitor by each consumer. That consumes large amounts of resources. Our approach, in contrast, uses credit records to evaluate the quality of service.

YIN Xian-Zhen et al. present QoS assurance mechanism based on credit to enhance the quality of application systems [3]. Their method does not consider the timeliness of credit because credit value is based on a time slot, whereas our does. The credit record values of service components in our approach are normalized in real time and then associate with the corresponding credit level, thus it can effectively solve the problem.

Recent research on global optimal policy is a great promising starting point for us out on the way to service selection. Global optimal problem can be modeled as Integer Programming Problem [4], Integer Linear Programming Problem [5], Mixed Integer Linear Programming Problem [6] [7] or Multi-dimensional Multi-choice 0-1 Knapsack Problem [8] and so on. Valeria Cardellini et al. present the method to QoS driven service selection, modeling optimal problem as MILP, and choose the services based on their QoS and load capacity [7]. In these methods above mentioned, if business process which needs to run a long time or some service components unavailable, the QoS of composite system will be affected. While in our approach the scheduled service is evaluated by its credit record, which as the reference standard for next time whether to invoke the service, to avoid selecting poor service component next time and then enhance the QoS of composite system.

3. Credit-driven Service Selection Model

3.1 The Basic Definition of Credit Model and Modeling

The architecture of service-oriented complicated application system is shown in figure 1. According to the Service Level Agreements (*SLAs*) between services and broker and an appropriate utility function, the service broker's major task is to select appropriate services and combine them into application system for meeting the *SLAs* negotiated with its users.

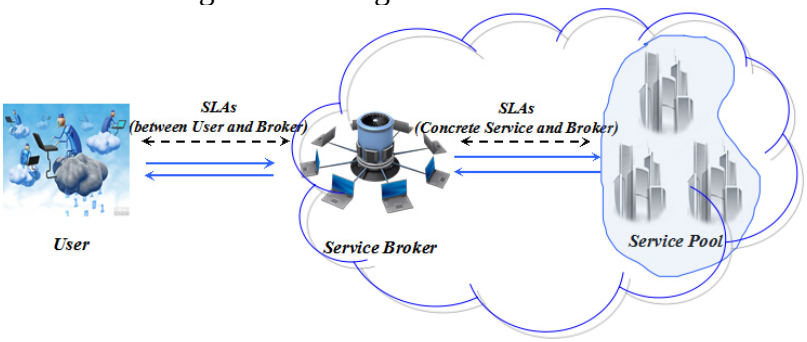

Figure 1. Service architecture of system model

The service component suddenly becomes unavailable while the system is running, which is so called service component failures caused by the uncertainty of the network environment. On the one hand, the reason is the accidental failures of system operation environment, such as operating system crash, network interruption, emergency power off and so on. The system may spend some time repairing those failures. On the other hand, the system may require scheduled outage for maintenance.

If the system mainly focuses on the cause of service component's dynamic changes, it may spend a lot of resources monitoring QoS information in real time. Thus, our new method uses credit record value to quantize the results caused by service component's dynamic and then selects the service in terms of service credit level.

Definition 1. *Task*: We denote the abstract task sets that compose the system by the $d_i \in D$, $i=0,1,...,m$, assuming all the tasks with four states: uncommitted, committed , service and completed.

Definition 2. *Service*: We denote the service component sets that implement abstract task di by S_i and similarly each service with four states: *unready*, *ready*, *service* and *reserved*. The service broker aims at selecting an appropriate service component $s_{ij} \in S_i$ $(i \in D, j \in S_i)$ for each abstract task d_i to meet the *SLAs* negotiated between service broker and its users.

Definition 3. Service *Credit Record (SCR)*: In service-oriented application systems, credit mainly contains two aspects: the service provider's credit and the service requester's credit. The former refers that provider should ensure the service can run safely and reliably over the whole service life-cycle. While the latter means the requester invokes the service in the specific format determined by provider and then pays the bills in time.

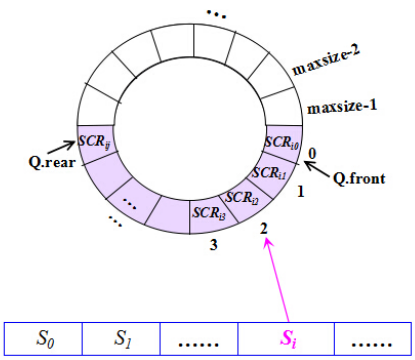

Figure 2. Storage structure of SCR

In the proposed method, the main focus is on the provider's credit. *SCR* refers to the credit history of service provider. The storage structure of *SCR* is as shown in figure 2: we denote the credit record value of sij by SCR_{ij}, which value at initialization is zero.

Because of the timeliness of credit, that is, with the passage of time decay, the credit record values need to be normalized to determine the credit level of each service component in the current period of time. Normalization formula is given by the following expression:

$$SCR_{nij} = \begin{cases} SCR_{ij} / SCR_{i\max}, & SCR_{ij} > 0 \\ 0, & SCR_{ij} \le 0 \end{cases} \tag{1}$$

Where SCR_{nij} and SCR_{ij} are the normalized credit record value and the actual credit record value of s_{ij}, respectively. SCR_{imax} is the maximum credit record value of the *i-th* group service components, in order to guarantee the value of SCR_{nij} ranges from 0 to 1.

Definition 4. *Credit function* (C_f): Its main mission is to adjust *SCR* (s_{ij}) according to the service completed condition in its life-cycle.

$$C_f : SCR = SCR + w_\alpha, w_\alpha \in W = \{w_1 = -1, w_2 = -0.7, w_3 = -0.4, w_4 = 0, w_5 = 1\} \tag{2}$$

where W is the sets of weighting factor corresponding to the Credit Event (*CE*)[3], which is the sets of event happened over the whole service life-cycle and the value of w_α reflects corresponding CE's impact on the service QoS. A positive historical experience has a positive value (w_α >0) and a negative historical experience has a negative value (w_α<0) and w_α=0 means that the event can be borne by the robustness of system.

Definition 5. *Credit Model (CM)*[3]: $CM=(Pr, C, R, C_f)$ is a four-tuples, where Pr is the sets of credit value of the service provider; C is the sets of credit value of the service customer ; $R=\{r1, r2, r3, ...\}$ is the sets of mapping relation from the service provider to customer, as a successful match between service providers and customers, a service relation r is built and with the end of service life r is removed from the R; The main mission of C_f is to adjust $SCR(s_{ij})$ of s_{ij} according to the service completed condition after a service relation removed from R.

Definition 6. *Service Credit Level (SCL)*: As shown in table 1, we list five levels of service provider's credit in terms of SCR_{nij}.

Table 1 Five levels of service provider's credit

SCL_{ij}	1	2	3	4	5
SCR_{nij}	$0 \le SCR_{nij} \le 0.1$	$0.1 < SCR_{nij} \le 0.3$	$0.3 < SCR_{nij} \le 0.6$	$0.6 < SCR_{nij} \le 0.85$	$0.85 < SCR_{nij} \le 1$

From table 1, mapping from normalized credit record value SCR_{nij} to service credit level SCL_{ij} can be obtained. A series of adjustment algorithms about credit record can be obtained in the paper [3] and credit mechanism has three main parts: the first is credit record database, which is used to store the credit records; the second is service relation database, which records all the service relations existing in the current system; the third is credit management module, providing service credit level information for composite system.

3.2 Credit-driven Service Selection Policy

The core of our credit-driven service selection policy is the randomized banding. Numbers of service components have a related-probability to bind abstract task d_i, but only one service component can be probabilistically selected to bind d_i at run time. The selection policy in our paper is a probabilistic set, denoted by matrix-vector C^p:

$$C^p = \left[c_0^p, \quad c_1^p, \quad \cdots, \quad c_m^p\right]^{\mathrm{T}} = \begin{bmatrix} c_{00}^p & c_{01}^p & \cdots & c_{0n}^p \\ c_{10}^p & c_{11}^p & \cdots & c_{1n}^p \\ \vdots & \vdots & \cdots & \vdots \\ c_{m0}^p & c_{m1}^p & \cdots & c_{mn}^p \end{bmatrix}, \quad c_{ij}^p \in [0,1], \quad \sum_{j=0}^{n} c_{ij}^p = 1 \tag{3}$$

Where $C^p{}_i=[C^p{}_{ij}], i \in D, j \in S_i$。 Our idea is to realize credit-driven service selection with binding probability C^p, forcing on each $C^p{}_{ij}$ an upper bound Γ_{ij}.

$$\Gamma_{ij} = SCL_{ij} / SCL_{gr} \tag{4}$$

where $SCL_{gr} = 4$ means good credit level, which service at least that level can guarantee stable long term running. When $\Gamma_{ij} \ge 1$, s_{ij} is high-availability, and unreliable otherwise, so that we can ensure that the service components selected for composite system have a relatively high credit level.

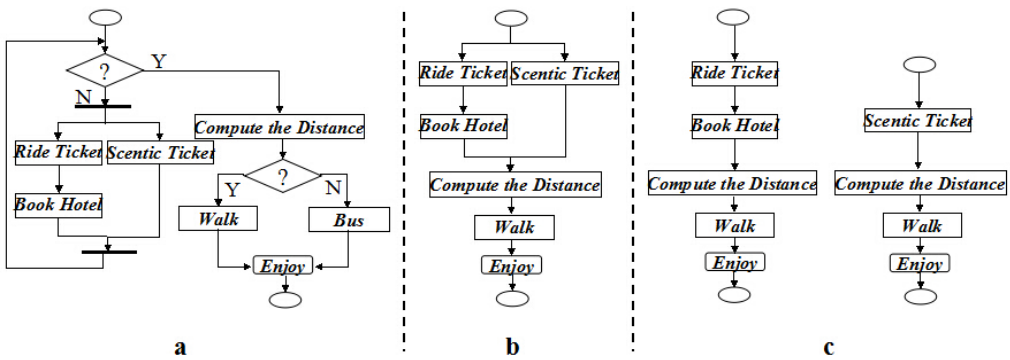

Figure 3. Workflow of a traveling website and one execution path and its subpath example

4. QoS Global Optimal Problem Driven by Credit

The goal of service broker is to determine an appropriate selection probability C^p while the *SLAs* negotiated with its users are fulfilled. In fact, computing selection probability C^p aims at selecting those services with high credit level and high quality, which is the so called "QoS Global

Optimization of Service Selection Driven by Credit". Thus, the QoS attributes of composite system are needed to compute in the process of optimization.

4.1 The Computing of QoS Parameters

The computing of QoS parameters is closely related to its workflow, which is needed to convert into execution path for simplifying the calculation process (assume that the probabilities of conditional branch and the number of times of reiterating loops are known) [6]. As shown in Figure 3, the workflow of composite system (figure 3-a) is convert into execution path (figure 3-b), assuming the loops is executed only one time. Control process of system usually has four basic structures: *order*, *parallel*, *choice* and *loops*. If execution path contains parallel structure with multi-branches, each branch is called subpath (see figure 3-c).

Thus, every execution scenario can be described by the execution path. To ensure the *SLAs* between broker and its users, each execution path should meet the QoS constrains.

Definition 7. *Service Level Aggrements (*SLAs*)*: a contract between service providers and customers to offer different qualities of service with different monetary prices (the higher service level the more expensive). Some metrics that *SLAs* may specify include [7]:①Cost: the agreed-upon cost for each invocation, ②Availability: the percentage of time that service can accessible in its life-cycle, ③Response-time: the time interval from service invocation to its completion.

In this paper, we denote by the tuple $SLA=\{Re_{max}, Av_{min}, Co_{max}\}$ the *SLA* between broker and its users, where Re_{max}, Av_{min} and Co_{max} represent the maximum value of response time, the minimum value of availability and the maximum value of cost, respectively. We add the new attribute $SCL\ (s_{ij})$, indicating credit level of s_{ij}, to extend the *SLA* model. If $SCL\ (s_{ij})\geq 4$, service component sij is high-availability; otherwise, cannot ensure normal use.

Now let's compute the QoS attributes of composite system, according to the service selection probability C^p and execution path ir_k. Here, we mainly focus on the user-perceived average QoS, which describes the quality of composite system and the QoS of worst instance, which is used as constrained boundary. Computational formulas of QoS attributes [7] are shown in table 2.

Computational formulas of average QoS values of abstract task di are shown in the second line of table 2. We denote by re_{ij}, av_{ij} and co_{ij} the response time, the availability and the cost of s_{ij}, respectively. Similarly, we denote by Re_i, Av_i and Co_i the average response time, the average availability and the average cost of d_i, respectively. Correspondingly, the QoS values of the worst instance, denoted by Re_i^w, Av_i^w and Co_i^w, the formulas are shown in the third line of table 2, where b_{ij} is a binary variable ($b_{ij}=1$ if $c^p_{ij}>0$ and 0 otherwise).

Table 2 QoS aggregation function

Aggregation Function	Re (response time)	Av (availability)	Co (cost)
Average QoS values of d_i	$Re_i = \sum_{j\in S_i} re_{ij} c_{ij}^p$	$Av_i = \sum_{j\in S_i} av_{ij} c_{ij}^p$	$Co_i = \sum_{j\in S_i} co_{ij} c_{ij}^p$
Worst QoS values of d_i	$Re_i^w = \max_{j\in S_i} re_{ij} b_{ij}$	$Av_i^w = \min_{j\in S_i} av_{ij} b_{ij}$	$Co_i^w = \max_{j\in S_i} co_{ij} b_{ij}$
Average QoS values of ir_k	$Re_k = \max_{ir_h^k \in ir_k} \sum_{d_i \in ir_h^k} Re_i$	$Av_k = \prod_{d_i \in ir_k} Av_i$	$Co_k = \sum_{d_i \in ir_k} Co_i$
Worst QoS values of ir_k	$Re_k^w = \max_{ir_h^k \in ir_k} \sum_{d_i \in ir_h^k} Re_i^w$	$Av_k^w = \prod_{d_i \in ir_k} Av_i^w$	$Co_k^w = \sum_{d_i \in ir_k} Co_i^w$

The aggregation formulas of the average QoS values of ir_k (denoted by Re_k, Av_k and Co_k) are shown in the fourth line. Similarly, the formulas of the worst QoS values of ir_k (denoted by Re_k^w, Av_k^w and Co_k^w) are shown in the fifth line, where ir_h^k represents the *h-th* subpath of ir_k.

4.2 Credit-driven QoS Global Optimization

QoS global optimal service selection problem driven by credit is to obtain an appropriate selection policy C^p while guarantees that user needs are met. Here, the optimal problem is considered as MILP and ultimately converted to find out the maximum value of objective function, which maximizes the aggregated QoS values along all the possible execution paths. Simple additive weighting technique, as a scalarization method, is used to obtain the aggregated values.

The formulas (5) are used to normalize the negative (*the former*) and the positive (*the latter*) QoS attributes, respectively. Then the normalized datas can be obtained.

$$Q_k^l\left(C^p\right)=\begin{cases}\dfrac{maxq_k^l-q_k^l(C^p)}{maxq_k^l-minq_k^l}\\ 1\end{cases},\quad Q_k^l\left(C^p\right)=\begin{cases}\dfrac{q_k^l(C^p)-minq_k^l}{maxq_k^l-minq_k^l} & ,\ maxq_k^l-minq_k^l\neq 0\\ 1 & ,\ maxq_k^l-minq_k^l=0\end{cases} \tag{5}$$

Where max q_k^l and min q_k^l represent the maximum and the minimum values of the *l-th* dimension QoS attributes along execution path ir_k, respectively. $q_k^l(p)$ shows that only the web services, which their binding probabilities are greater than 0, participate in the normalization processing.

Then, the normalized QoS attributes are calculated by using a weighted summation method.

$$sum_k=\sum_l \omega_l Q_k^l\left(C^p\right),\ \sum_l \omega_l=1,\text{且}\,\omega_l>0 \tag{6}$$

Where ω_l means the relative importance of QoS attributes according to user's QoS requirements.

Finally, the objective function can be obtained (pr_k specifies the execution probability of ir_k).

$$F(C^p)=\sum_k pr_k sum_k\left(C^p\right) \tag{7}$$

The decision variables are given by:

c^p_{ij}: its value range is from 0 to 1, indicating the probability that s_{ij} is bound to the abstract task d_i. This variable is used to drive the service selection based on credit level.

b_{ij}: a binary variable means whether s_{ij} is bound to d_i ($b_{ij}=1$ if $c^p_{ij}>0$ and 0 otherwise). It can guarantee the user needs are fulfilled.

The optimization problem is to find out the maximum value of the objective function (7) under the restraint conditions (8) and then the service binding probability C^p is obtained.

$$\begin{cases}\sum_{j\in S_i} c_{ij}^p=1\\ c_{ij}^p\le \Gamma_{ij}\\ c_{ij}^p\le b_{ij}\end{cases},\quad \begin{cases}re_{ij}b_{ij}\le \mathrm{Re}_i^w\\ av_{ij}b_{ij}\ge Av_i^w\\ co_{ij}b_{ij}\le Co_i^w\end{cases},\quad \begin{cases}\mathrm{Re}_k^w\le \mathrm{Re}_{\max}\\ \ln\left(Av_k^w\right)\ge \ln\left(Av_{\min}\right)\\ Co_k^w\le Co_{\max}\end{cases},\quad \forall i\in D,\forall s_{ij}\in S_i,\forall k \tag{8}$$

The restraint conditions (8) mean for each abstract task d_i at least one service can be bound, and specify the upper limit to c^p_{ij}; the QoS attribute values of the worst-case scenario are the upper limit to the selected services and ensure that the user QoS requirements are fulfilled even under the worst-case scenario.

5. The Simulation Experiment and Analysis

In order to proof the effective of service selection policy driven by credit, we conduct a series of simulation experiments. Experiment 1 simulates the execution process of binding service in proxy-based application system, and the dynamic changes of service credit record values *SCR* are mapped to the service credit level *SCL* to increase the success rate of distinguishing between good and bad services [3]. Experiment 2 demonstrates whether our method can effectively lower the rescheduling times caused by service components dynamic to reduce the CPU overhead and then improve the CPU utilization. Experiment 3 will verify the effective of our new approach.

Experiment 1 validates that whether the proposed service credit record value can real-time reflect service credit level. Thus, in the simulation system, it randomly generates 300 simple and independent service components (no internal logic) with different QoS Level *1, 2, 3, 4* and *5* that

associates with the corresponding probability of service offline *20‰, 15‰, 10‰, 5‰* and *1‰*, respectively. The arrival requester to the system is random variable and submits to Poisson distribution. After a period of time, look at all the normalized service credit record values, used to distinguish the service credit level. All the service's credit levels are 0 at initialization time.

As the system running time increases, at *t=1800s*, each service component's credit record value is adjusted according to the status of service completion, as shown in figure 4, credit mechanism can effectively sense the high-availability of service components. System is also able to identify the stable high-quality services and unstable normal services under the condition that the difference of all the service offline probabilities are not obvious. Therefore, system can select service components according to credit level to realize credit driven service selection policy.

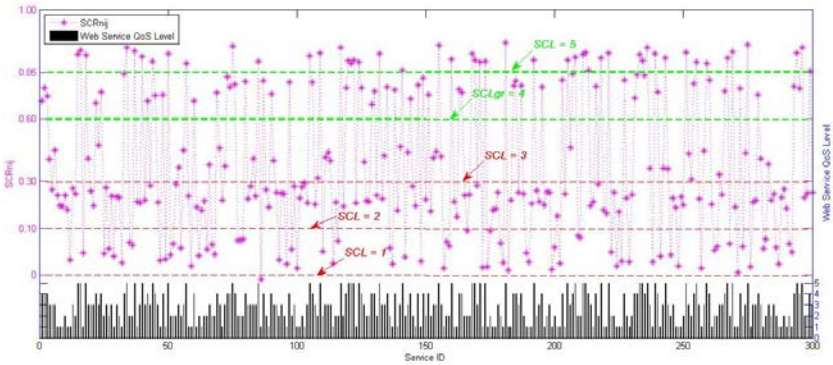

Figure 4. Comparison chart of Web Service QoS Level and SCRnij (t=1800s)

In experiment 2, the simulation system randomly generates 300 independent service components with different performance and function parameters and all the services are listed in descending order from the amount of credit level values *SCL*, and then generates 20 groups of service request tasks with different running time. The simulation system can schedule the service request task to the corresponding service, using No Service Credit Level (*NSCL*) method and our Service Credit Level (*SCL*) approach respectively, and then record the average schedule times of per group request tasks.

The result is shown in figure 5, we can see the average task schedule times with our *SCL* approach (*1.34-1.78*) is significantly lower than *NSCL* method (*1.53-2.32*).

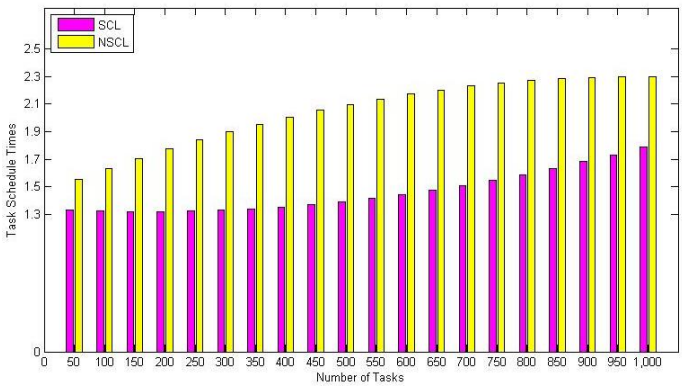

Figure 5. Comparison chart of Average Task Schedule Times with NSCL and SCL method

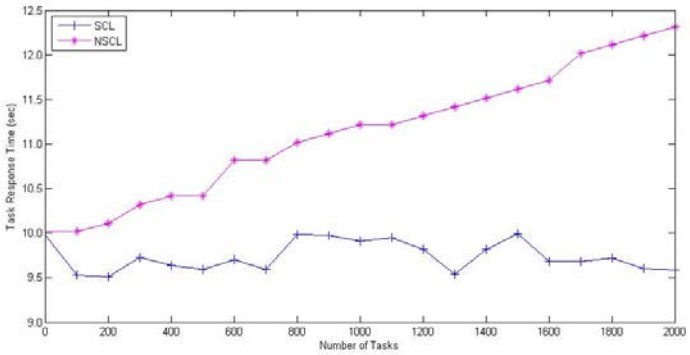

Figure 6. Comparison chart of Average Task Response Time with NSCL and SCL

In experiment 3, the simulation system randomly generates 300 independent service components with different QoS parameters, and then generates 20 groups of service request tasks with different numbers and running time. Then, the service request tasks are scheduled to the corresponding service,

Approach using NSCL method and SCL approach respectively, and then record the average response time of per group request tasks.

As the figure 6 is shown, with the rapid increase of tasks, the average task response time with *NSCL* method increases linearly, while our presented *SCL* approach is superior to the former. The reason is that our method always chooses the high-availability services in terms of service credit level, which won't usually result in rescheduling task caused by service offline at runtime, and then reduces the task response time.

6. Conclusion

Given that many service selection approaches proposed by now do not take into full consideration the effect of service component's dynamic on the quality of service-oriented application system. Therefore, to solve this problem, we present a new method to QoS global optimal service selection driven by credit, which gives out the service selection model with credit level and quantizes the service component's dynamic and then uses normalization method to map the credit record values to the service credit levels. Using the mixed integer linear programming, QoS global optimization service selection model driven by credit is built. Simulation experimental results show that the new approach can reduce the influence caused by service components dynamic and then enhance the QoS of composite system compared with *NSCL* methods. On the basis of the arguments above, we'll consider random load balancing and develop the prototype system based on QoS global optimal service selection driven by credit in the future.

References

[1]. Milanovic N, Malek M. Current solutions for web service composition [J]. Internet Computing IEEE, 2004, 8(6):51--59.

[2]. Hang C W, Singh M P. Trustworthy Service Selection and Composition.[J]. Acm Transactions on Autonomous & Adaptive Systems, 2011, 6(1):627-662.

[3]. YIN Xian-Zhen, JIANG Jing, PAN Zhen-Kuan XIA Bai-Qiang, A Credit Record Driven QoS Assurance Mechanism for SOC System[J]. In: International Seminar on Business and Information Management, 2008, 12.

[4]. Oh S C, Lee D, Kumara S, et al. A Web Service Composition Framework Using Integer Programming with Non-functional Objectives and Constraints[C]// Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services. IEEE Computer Society, 2008:347-350.

[5]. Zeng L, Benatallah B, Ngu A H H, et al. QoS-Aware Middleware for Web Services Composition [J]. IEEE Transactions on Software Engineering, 2004, 30(5):311-327.

[6]. Ardagna D, Pernici B. Adaptive Service Composition in Flexible Processes [J]. IEEE Transactions on Software Engineering, 2007, 33(6):369-384.

[7]. Cardellini V, Valerio V D, Grassi V, et al. A new approach to QoS driven service selection in service oriented architectures[C]// Proceedings of the Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System Engineering. IEEE Computer Society, 2011:102-113.

[8]. Yu T, Zhang Y, Lin K J. Efficient algorithms for Web services selection with end-to-end QoS constraints.[J]. Acm Transactions on the Web, 2007, 1(1):2007.