Temporal-Perturbation aware Reliability Sensitivity Measurement for Adaptive Cloud Service Selection

Lei Wang, Qiang He, Demin Gao, Jing Wan, and Yunqiu Zhang

Abstract—Benefiting from the pay-as-you-go business model, cloud-based software applications are becoming more and more popular. A composite cloud system can be constructed by integrating existing component cloud services available over the internet as its system components. In order to fulfill the service-level agreements (SLAs), as well as users' quality of experience (QoE), a stable execution of the constructed system is desirable in the long term. To achieve this goal, system components at high risk of failing must be identified and fault-tolerated. This is extremely challenging in the dynamic cloud environment that host the component cloud services. However, existing approaches are constrained by their lack of modeling and analysis of system components' fluctuating reliability time series. To systematically address these issues, in this paper, we propose PARS, a perturbation-aware approach, for measuring the reliability sensitivity of component cloud services. It first analyzes the negative perturbations in component cloud services' historical reliability time series. Then, it calculates the reliability sensitivity of the component cloud services by analyzing how their reliability perturbations impact the reliability of the entire cloud system. Based on PARS, we propose a proactive adaptation approach for constructing and operating composite cloud systems with 1-out-of-2 N-version Programming fault-tolerance. This approach takes the reliability sensitivity of component cloud services estimated by PARS as input to assure the reliability of the cloud system. The results of experiments conducted on two widely used datasets demonstrate the effectiveness and efficiency of the proposed approaches in ensuring the reliability of composite cloud systems.

Index Terms—Cloud Service Selection, Perturbation, Proactive Adaptation, Reliability Sensitivity, Reliability Time Series

1 Introduction

To support the creation and delivery of services, standardization bodies, such as the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS), have made standardization efforts of Service-Oriented Architecture (SOA) for implementing service-oriented systems [1]. Recently, cloud computing, based on a pay-as-you-go business model, has significantly promoted the service computing techniques in real-world industrial applications. Besides SOA, the development of big data analysis technologies, including upper-level business modeling, management, and analysis, and lower-level service data management and analysis, enable us to build composite cloud systems and/or mashup-based cloud service softwares [2], [3].

A composite cloud system is constructed by integrating functionally independent system components based on Web technologies and SOAP, JSON, or HTTP communication protocols [4], [5]. As illustrated in Fig. 1, a fashion trends prediction composite cloud system can be constructed through selecting open component cloud services [6], [7], [8]. This system orchestrates 4 system components to predict fashion trends for a design company. First, *Sales data extraction* and *Review data extraction* are executed in parallel to collect sales data and consumers' review data. Second, *Data integration* merges the aforementioned data. Finally, *Trends prediction* is executed to estimate the future fashion trends. For each system component, service selection is performed to select an optimal component cloud service (mostly in the form of RESTful APIs) from a group of candidate component cloud services. For each functionality, there are usually many publicly available candidate component cloud services published online with similar (or sometimes identical) functionalities but different Quality of Service (QoS).

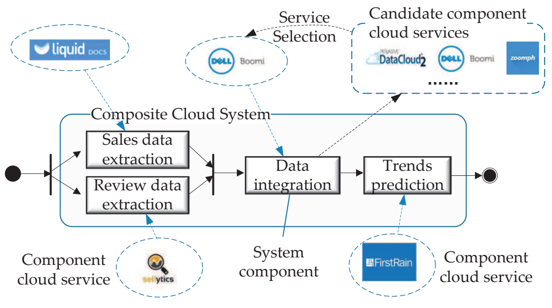


Fig. 1. A demonstrating example of composite cloud system.

To fulfill the service-level agreements (SLAs) and ensure users' quality of experience (QoE), the quality of the entire

• L. Wang, J. Wan, and Y. Zhang are with the Department of Management Science and Engineering, Nanjing Forestry University, LONGPAN Road 159, Nanjing 210037, China. E-mail: leiwangchn@163.com; 1132929104@qq.com; eitherzhangyq@163.com.
• Q. He is with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia 3122. E-mail: qhe@swin.edu.au.
• D. Gao is with the College of Information Science and Technology, Nanjing Forestry University, LONGPAN Road 159, Nanjing 210037, China. E-mail: dmgao@njfu.edu.cn.

Manuscript received xx. x, xxxx; revised xx x, xxxx.

composite cloud system must be ensured, e.g., its response time, reliability, etc. However, this is a challenging issue due to the component services' uncertain internal working status and their volatile operating environment in the cloud [9]. To guarantee the overall quality of the composite cloud system, the cloud service selection problem has attracted a lot of research attention [10], [11], [12]. Cloud service selection should prioritize the reliability of important system components in the composite cloud system for the long term.

Particular attention must be paid to system components at high risk of failing in cloud system construction and maintenance [13]. One of the fundamental issues in system construction and maintenance is that we need to evaluate the reliability sensitivity of each component cloud service [14]. Here the reliability sensitivity refers to the *Importance (Sensitivity)*, and is measured by how the increase or decrease in the reliability of an individual component influences the reliability of the entire cloud system [15].

In the cloud computing environment, the reliability of a component cloud service may be impacted by three main factors: 1) the execution status of the component cloud service, such as hardware failures, resource failures, overflow failures, CPU overload, memory exceptions, etc; 2) the communication links between callers and the service, e.g., network congestion, network failures, etc; and 3) the real-time workloads on the component cloud service. These events occur at runtime dynamically, impacting the reliability of component cloud services also dynamically. This results in a concept drifting probability distribution of each component cloud service's reliability time series [16]. In the cloud environment, the evolution of reliability time series for a component cloud service follows a continuous-time homogeneous (or 1^{st}-order) Markov chain evolution rule [9] The fluctuations in a component cloud service's reliability time series are uncertain at neighboring time points. Moreover, the fluctuations in different component cloud services' reliability time series are also different. An effective approach for reliability sensitivity measurement must be able to evaluate the impacts of uncertain reliability fluctuations of component cloud services on the entire composite cloud system.

Measuring the reliability sensitivity of component cloud services in a composite cloud system can help to find out the system components whose reliability changes have higher impacts (positive or negative) on the reliability of the entire system [17], [18], [19]. Existing long-term reliability prediction approaches are designed for *build-time* cloud service selection [20]. They are used to predict the component cloud services' average reliability in the future without analyzing their historical reliability. It is desired to ensure *runtime* system reliability through reliability sensitivity analysis. To guarantee that a composite cloud system can adapt to the dynamics in the operating environment, the reliability sensitivity of its system components reflects their real-time impacts to the reliability of the composite system. At build-time, proactive adaptation can be performed to update the cloud service composition workflow. At runtime, when the system reliability is identified to be deviating from the SLA constraint and the risky component cloud services are identified through reliability sensitivity analysis, proactive adaptation can be performed to fault-tolerate the system.

Reliability sensitivity analysis is fundamentally different from the relevant existing works on criticality measure in service-oriented systems and traditional reliability sensitivity analysis in composite systems [17], [21], [22], [23]. For example, criticality measure approaches in service-oriented systems rank the components' importance based on composite system architecture analysis [21], [22]. The component invocation relationship and frequency are analyzed to identify the important components in a composite system. The approaches for measuring the reliability sensitivity of system components in traditional computer systems, e.g., those proposed in [17], [23], calculate a component's sensitivity (importance) with parametric sensitivity analysis. The component's reliability is expressed as an input parameter of a function to evaluate the output behavior of the composite system. These approaches pinpoint the particular component that is the most influential in affecting the reliability of the composite system. However, none of the existing approaches has systematically considered the uncertain temporal perturbations when measuring the reliability sensitivity system components identification based on their historical reliability time series.

To address this above issue, this paper investigates component cloud services' evolution regularities in their up-to-date reliability time series. The key idea is to systematically measure the reliability sensitivity of each component cloud service by analyzing its negative influence on the reliability of the entire composite cloud system. Since a component cloud service's reliability follows the 1^{st}-order Markov chain rule, the goal of the reliability analysis based on historical time series is to achieve system adaptation through identifying reliable system components (those with low reliability sensitivity). As discussed above, the quality of a cloud service is impacted by a variety of factors. It is difficult, if not impossible, to analyze the impacts of these different factors on the reliability of a cloud service individually, let alone collectively. Thus, we appeal to reliability analysis based on the cloud service's historic reliability time series data, which indicates the reliability of the cloud service under different circumstances. In this way, we are able to perform reliability analysis for cloud services in a generic manner. The major contributions of this paper are summarized as follows:

- We present a reliability evaluation method for a component cloud service based on their failure probability under continuous client-side invocation tests. The reliability of a component cloud service during the evaluation period is calculated based on its failure rate and the exponential reliability equation.
- We adopt the 1^{st}-order Markov Chain rule in this paper to describe the evolution regularity of component cloud services' up-to-date reliability time series. Reliability perturbation is defined in this paper to note the changes in system components' reliability at consecutive time series points.
- We propose a perturbation-aware reliability sensitivity measurement approach (named PARS) for analyzing the cumulative negative effects of system components' reliability perturbations to the composite cloud system.

- Based on PARS, we propose a proactive adaptation approach for composite cloud systems' execution quality assurance.
- We conduct extensive experiments on two widely-used public data sets to compare our approaches against 4 representative approaches.

The rest of this paper is organized as follows. We introduce the modeling approaches of reliability time times and reliability evolution in Section 2. The proposed PARS approach is described in detail in Section 3. We present the proactive adaptation approach for a composite cloud system based on PARS in Section 4. We give our implementation of experiments in Section 5. We summarize the relevant existing works in Section 6. In Section 7, we conclude the paper and lay out some important future directions.

2 Reliability Modeling

Section 2.1 introduces the formal model of reliability and a component cloud service's reliability time series. Section 2.2 presents the 1^{st}-order Markov chain evolution rule for a component cloud service's reliability time series. The major notations used in this paper are summarized in Table. 1.

TABLE 1
Major Notations Used in This Paper

Notations	Descriptions
λ	failure rate
λ_i	component cloud service i's failure rate
$\lambda_{k/n}$	failure rate of a k-out-of-n composite cloud system
Δt	the survival time for reliability evaluation
L	the invocation result
RT_{max}	the maximal response time constraint
$r_i^{\Delta t}$	reliability of component cloud service i
r_i^{TS}	component cloud service i's reliability time series
m	number of time points in a reliability time series
$r_i^{\Delta t(j)}$	jth time point's reliability value of $r_i^{\Delta t}$
$p(i,j)$	ith system component's jth perturbation
$\widehat{p}(i,j)$	normalized value of $p(i,j)$
Φ	a composite cloud system
π_j	the sensitivity value of jth perturbation
α^{m-j}	the jth perturbation's influencing factor parameter
$I_{PARS}(i)$	the sensitivity value of component cloud service i
$\widehat{r}_\Phi$	the average reliability of composite cloud system Φ
r_Φ^{SLA}	required reliability for the composite cloud system Φ
s_l	the lth most sensitive component cloud service
$OP(s_l)$	the optimal candidate component cloud service for s_l
$U(l,j)$	utility function for s_l's jth candidate
k	number of most sensitive component cloud services
SR	success rate
ORE	overall reliability enhancement

2.1 Reliability Time Series

A few metrics have been proposed for evaluating the reliability of traditional software systems. To a few, there are the Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), hazard rate (the probability of system survives till time t), PoFoD (Probability of Failure on Demand) and exponential reliability [18].

The MTTF, MTBF, and hazard rate are suitable for system failures that have a statistical regularity in time, such as errors occuring in an average of 5 seconds. In the corresponding exponential reliability calculation, the failure rate parameter (i.e., λ) is a constant number with a period of survival time.

A component cloud service (e.g., a cloud service API) is invoked on demand by users or composite cloud systems. Due to its uncertain internal working status and volatile operating environment in the cloud, the time interval between its occurrences of failures is uncertain. Thus, the above existing reliability metrics cannot be directly employed to evaluate the reliability of component cloud services.

Since we can collect the returned HTTP messages by public component cloud service invocations and there are many web testing tools (e.g., apache-JMeter[1]) for us to implement, we employ the PoFoD metric [24] in this paper to give a performance-aware reliability definition for component cloud services.

We set a fixed time period Δt as the survival time to test the failure rate of a component cloud service through continuously client-side invocation tests. The length of Δt is denoted as $len(\Delta t)$. The invocation test is performed every s seconds. We then collect the failure probability from these $n = \frac{len(\Delta t)}{s}$ times of Bernoulli trial results, i.e., with $Pr(L = 0)$ for failed invocations. Since the composite cloud systems delivered to users are usually constrained by SLAs (concerning reliability, response time, etc.), we set an acceptable maximal response time constraint RT_{max} for the received responses. This way, we have the following performance-aware failure probability evaluation result.

$$f(\Delta t) = Pr(L = 0), \tag{1}$$

where $L = 0$ means that the client-side testing invocations returned failures or did not respond within the time constraint RT_{max}.

We can obtain a rate value to estimate the ability of a component cloud service to perform its required functions within survival time of Δt with s times continuously Bernoulli trials. The rate value is expressed as the probability of $[1 - f(\Delta t)]^n$.

For a specific component cloud service within the specific time interval of Δt, the failure rate is a constant and can be expressed by failure rate λ. The failure rate for the component cloud service is

$$\lambda = 1 - [1 - f(\Delta t)]^n. \tag{2}$$

Using the exponential reliability expression, the reliability of component cloud service i within the survival time of interval Δt is

$$r_i^{\Delta t} = exp[-\lambda \times len(\Delta t)]. \tag{3}$$

Noting that we use the exponential reliability equation to calculate the reliability of a component cloud service within a specific time period, it does not mean that the reliability of a component cloud service follows an exponential distribution. The length of time span $len(\Delta t)$ is a constant value. The exponential reliability function defined in this paper does not decline with the increase in the survival time value. We give the following justifications for the definition of performance-aware exponential reliability equation. First, the definition of reliability satisfies the IEEE STD-729-1991 standard for software reliability definition [25]. The IEEE

1. https://jmeter.apache.org/

STD-729-1991 standard defines software reliability as "the ability of a system or component to perform its required functions under stated conditions for a specified period of time." In Eq. (3), λ is the probability that the system component experiences a failure within the survival time constraint with the length of $len(\Delta t)$. Let $t_1, t_2, \cdots, t_n$ represent the time points with equal intervals in Δt and $f(\Delta t)$ represent the failure probability of the system component at time t_n. According to the IEEE software reliability standard, $r_i^{\Delta t}$ is the survival rate for the system component at time t_n, i.e., the system will not fail till time t_n. Let $F(\Delta t) = 1 - r_i^{\Delta t}$, we have $f(\Delta t) = F'(\Delta t)$. Then, a differential equation can be induced: $\lambda = \frac{f(\Delta t)}{r_i^{\Delta t}} = \frac{F'(\Delta t)}{1-F(\Delta t)}$. The following solution can be obtained by solving the differential equation: $F(\Delta t) = 1 - exp[-\lambda \times len(\Delta t)]$. Hence, Eq. (3) can be induced. Second, the reliability values are normalized to the range of [0,1]. A larger value indicates higher reliability of the component cloud service.

The unstable communication links fluctuate the response time of a component cloud service over time. For example, as illustrated in Fig. 2, we use apache-JMeter to collect the response time of Baidu's map cloud service. Here, we set $RT_{max} = 1000ms$, the responses taking more than $1000ms$ are regarded as delayed responses (i.e., $L = 0$). We send an HTTP GET invocation request to Baidu map every 10 seconds. If we send 10 continuous invocation requests to evaluate the reliability of Baidu map, the survival time for the reliability evaluation period is $\frac{100}{60}$ minutes. Within the time period of $\Delta t(m-1)$, there are 4 delayed responses. Accordingly, there is $\lambda = 1 - (1 - 0.4)^{10} = 0.99395$. Then, there is $r_i^{\Delta t} = exp(-0.99395 \times \frac{100}{60}) = 0.19$.

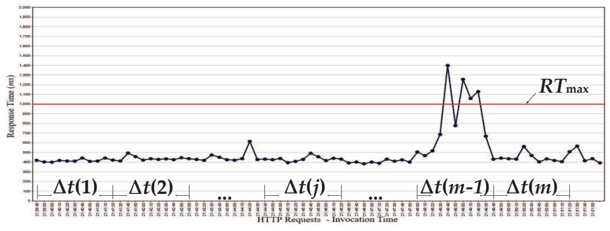


Fig. 2. Schematic view of time series.

We represent the survival time intervals as $\Delta t(1), \Delta t(2), \cdots, \Delta t(m-1), \Delta t(m)$. There is a reliability evaluation value within each time period. We have component cloud service i's reliability time series as follows:

$$r_i^{TS} = r_i^{\Delta t(1)}, r_i^{\Delta t(2)}, \cdots, r_i^{\Delta t(m)}. \tag{4}$$

2.2 Markov Chain Rule

To evaluate the uncertainty in component cloud services' reliability, the evolution feature of consecutive values in a reliability time series can be modeled as a continuous-time homogeneous Markov chain [9], [26].

As illustrated in Fig. 3, for component cloud service i, the reliability evaluation within a time period can reflect its execution status. We assume that the reliability under the up-to-date execution status is $r_i^{\Delta t(j)}$. Within the next time period, the component cloud service will be transformed to the future execution status; resulting in the reliability value of $r_i^{\Delta t(j+1)}$. Many events can lead to this transformation, such as fluctuations of server and/or client-side network changes in server load, usages of CPU and memory, invocations of functions during the service execution, etc. The occurrences of these events are uncertain at runtime.

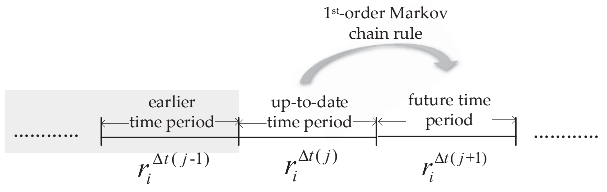


Fig. 3. Markov chain evolution rule of reliability time series.

Based on the above execution evolution feature, the reliability value during the future time period (i.e., $r_i^{\Delta t(j+1)}$) is only depend on the up-to-date reliability (i.e., $r_i^{\Delta t(j)}$) and the specific event. For the uncertainty of the events to come, the reliability value during the future time period is irrelevant with the reliability during the earlier time period (i.e., $r_i^{\Delta t(j-1)}$ and earlier reliability values). This memory-less evolution satisfies the 1^{st}-order Markov Chain rule. The time-homogeneous reliability evolution gives us more insights to analyze the single-step transition of reliability. Since single one-step transition is triggered by uncertain events, the statistics of multiple one-step ahead transitions can reflect the long-term execution evolution regulation of the component cloud services' reliability.

3 PARS Approach

In this section, we present the proposed PARS approach which analyzes how multiple one-step ahead transitions of a component cloud service's reliability influences the composite cloud system's reliability. Section 3.1 describes the perturbation function for the single-step transition component cloud service's reliability. Section 3.2 states how to aggregate the system component's reliability in a composite cloud system to calculate the composite system reliability. Section 3.3 presents our perturbation-aware approach for reliability sensitivity measurement.

3.1 Time-homogeneous Reliability Perturbations

In a composite cloud system, each system component selects and binds an optimal component cloud service. The reliability of the bound component cloud service determines the reliability of the system component. When a component cloud service is running under a specific execution status during a time period with the reliability of $r_i^{\Delta t(j)}$, let the reliability for the component cloud service during the next time interval be $r_i^{\Delta t(j+1)}$. In a reliability time series, the time-homogeneous reliability perturbation is defined as the difference in the reliability values for a single-step ahead transition.

In a reliability time series with m values, there are $m-1$ perturbations and each one can be regarded as a 1^{st}-order Markov Chain evolution. For any $j \in [1, m-1]$, the ith system component's jth perturbation can be defined as:

$$p(i,j) = r_i^{\Delta t(j+1)} - r_i^{\Delta t(j)}. \tag{5}$$

By a one-step transition, the reliability value of a component cloud service may become greater or smaller. The $p(i,j)$ may be a positive or negative values. We need to identify the system components at high risk of failing. Given a composite cloud system, if we assume the reliability of all its component cloud services except i remain unchanged, the increase in the reliability of i will increase the composite cloud system reliability. In contrast, the decrease in component cloud service i's reliability will decrease the composite cloud system's reliability. Existing works analyzed the positive and negative sensitivity of a system component in a composite system, respectively [15].

To guarantee the dependability of composite cloud systems, we need to identify the reliability perturbations of each component cloud service which have negative influences on the composite cloud system reliability. A larger negative perturbation of a component cloud service has a higher negative influence. We perform the following min-max normalization for the perturbations [27], [28].

$$\widehat{p}(i,j) = \begin{cases} \dfrac{p(i)^{max} - p(i,j)}{p(i)^{max} - p(i)^{min}} & , if\ p(i)^{max} \neq p(i)^{min} \\ 1 & , if\ p(i)^{max} = p(i)^{min}, \end{cases} \tag{6}$$

where

$$p(i)^{max} = \max_{j \in [1, m-1]} p(i,j), \tag{7}$$

and

$$p(i)^{min} = \min_{j \in [1, m-1]} p(i,j). \tag{8}$$

Let $p(\Phi, j)$ be the jth perturbation of the composite cloud system, we can obtain the normalized $\widehat{p}(\Phi, j)$ for the composite system through a similar min-max normalization process.

3.2 Reliability of Composite Cloud Systems

In a composite cloud system, when the failure rate values (i.e., λ) of the component cloud services are evaluated through the method presented in Section 2.1, we can calculate the failure rate value of the composite cloud system according to the composite system execution workflow.

Many software reliability engineering books have examined the aggregation functions for a composite system under different composition structures [19], [29], [30], [31], [32]. We present the following lessons learned for the aggregation functions for sequence, parallel, branch, loop, and k-out-of-n composite cloud system structures.

Let Φ be a composite cloud system, $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ be the failure rate values of the n system components which compose Φ. We use b_i to denote the probability that the ith branch in the composite cloud system is executed, and l_i to denote the probability that the loop is executed for i times, in which $\sum_{i=1}^{n} b_i = 1$ and $\sum_{i=0}^{n} l_i = 1$. We define λ_Φ as the failure rate value for the composite cloud system Φ within a unified survival time for all the involved component cloud services (i.e, Δt).

In practice, the composite cloud system may contain more than just one structure. We can treat the graph of the whole service execution flow as a semi-Markov process. The failure rate values for restricted branches can be calculated respectively. Hence, we obtain the measures for the failure rate value for the whole composite structure. We present the following aggregation functions for calculating λ_Φ for sequence, parallel, branch, and loop structures of composite cloud system workflows [20].

For sequence and parallel structures, the aggregation function is

$$\lambda_\Phi = 1 - \prod_{i=1}^{n} (1 - \lambda_i). \tag{9}$$

As for branch structure,

$$\lambda_\Phi = 1 - \sum\nolimits_{i=1}^{n} b_i (1 - \lambda_i). \tag{10}$$

And the loops structure can be treated by

$$\lambda_\Phi = 1 - \sum\nolimits_{i=0}^{n} l_i (1 - \lambda_1)^i. \tag{11}$$

To guarantee the execution reliability of a composite cloud system, fault tolerance techniques, such as N-version programming (or NVP) [29], [33], are usually used. Let $1 \le k \le n$, a composite cloud system fault-tolerated by NVP Φ functions as long as at least k of the n system components function properly. An NVP-based composite system can be modeled as a k-out-of-n composite system.

As for a composite cloud system, the reliability of each its component cloud service might be different. In this heterogeneous case, we cannot simply estimate the failure rate of the composite cloud system by a permutation and combination approach. The following recursive function can be used to estimate the failure rate of the k-out-of-n composite cloud system [19].

$$\begin{aligned} &\lambda_\Phi = \lambda_{k/n} = \lambda_n \times \lambda_{k/n-1} + (1 - \lambda_n) \times \lambda_{k-1/n-1}, \\ &\lambda_{0/n} = 0, \\ &\lambda_{j/i} = 1,\ when\ j > i, \end{aligned} \tag{12}$$

where $\lambda_{k/n}$ represents the failure rate of the k-out-of-n composite cloud system, λ_n represents the failure rate of the nth system component, $\lambda_{k/n-1}$ represents the system's conditional failure probability in the situation where the nth system component fails.

For example, let's consider a non-identical two-out-of-three system, we have:

$$\begin{aligned} &\lambda_{2/3} = \lambda_3 \times \lambda_{2/2} + (1 - \lambda_3) \times \lambda_{1/2}, \\ &\lambda_{2/2} = \lambda_2 \times \lambda_{2/1} + (1 - \lambda_2) \times \lambda_{1/1}, \\ &\lambda_{2/1} = 1, \\ &\lambda_{1/1} = \lambda_1 \times \lambda_{1/0} + (1 - \lambda_1) \times \lambda_{0/0} = \lambda_1, \\ &\lambda_{1/2} = \lambda_2 \times \lambda_{1/1} + (1 - \lambda_2) \times \lambda_{0/1} = \lambda_1 \lambda_2, \\ &\lambda_{2/2} = \lambda_2 \times \lambda_{2/1} + (1 - \lambda_2) \times \lambda_{1/1} = \lambda_2 + (1 - \lambda_2) \times \lambda_1 \\ &\qquad = \lambda_2 + \lambda_1 - \lambda_1 \lambda_2. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \lambda_{2/3} &= \lambda_3 \times (\lambda_2 + \lambda_1 - \lambda_1 \lambda_2) + (1 - \lambda_3) \times \lambda_1 \lambda_2 \\ &= \lambda_2 \lambda_3 + \lambda_1 \lambda_3 - \lambda_1 \lambda_2 \lambda_3 + \lambda_1 \lambda_2 - \lambda_1 \lambda_2 \lambda_3 \\ &= \lambda_1 \lambda_2 + \lambda_1 \lambda_3 + \lambda_2 \lambda_3 - 2 \lambda_1 \lambda_2 \lambda_3. \end{aligned}$$

Plugging λ_Φ into Eq. (3), we can estimate the reliability of the composite cloud system. Using Eq. (5) and (6), we can obtain the jth perturbation value $p(\Phi, j)$ for the composite cloud system Φ and the normalized jth perturbation value $\widehat{p}(\Phi, j)$, respectively.

3.3 Reliability Sensitivity for Component Cloud Services

To identify the system components at high risk of failing, we need to measure the reliability sensitivity of each component cloud service in the composite cloud system. The traditional reliability measures cannot be directly used, as discussed above in Section 2.1.

By considering the probability of occurrence and the Shannon decomposition rule, the Birnbaum importance index [17] for a system component can be calculated as

$$I_B(i) = \frac{Pr(L_\Phi = 0, L_i = 1)}{1-\lambda_i} - \frac{Pr(L_\Phi = 0, L_i = 0)}{\lambda_i}, \quad (13)$$

where L_Φ represents the Bernoulli trial result for the composite cloud system Φ, and L_i represents the Bernoulli trial result for component cloud service i, respectively, λ_i is the failure rate of component cloud service i.

Since it is difficult to estimate the joint probability of $Pr(L_\Phi = 0, L_i = 1)$ and $Pr(L_\Phi = 0, L_i = 0)$ for composite cloud systems. We propose PARS, a perturbation-aware approach for measuring the reliability sensitivity of component cloud services, which considers the negative perturbations and their cumulative effects.

To measure the negative influences of system component's reliability perturbation on the composite cloud system, when the composite system's reliability perturbation is a positive value, we make the normalized reliability perturbation of the composite system 0. For each $j \in [1, m-1]$,

$$\widehat{p}(\Phi, j) = \begin{cases} 0 & , if\ P(\Phi, j) > 0 \\ \widehat{p}(\Phi, j) & , if\ P(\Phi, j) \le 0. \end{cases} \quad (14)$$

The sensitivity of component cloud service i measured by each perturbation j is calculated as

$$\pi_j = \frac{\widehat{p}(\Phi, j)}{\widehat{p}(i, j)}. \quad (15)$$

There are $m-1$ times of perturbations in a reliability time series. We consider the cumulative effect and define an influencing factor parameter $\alpha \in [0, 1]$ for the perturbations. For the Markov chain evolution feature of the component cloud service's reliability time series, perturbations are mutually independent. Multiple times of perturbations in the 1^{st}-order Markov evolutionary reliability time series can reflect the cumulative effects. A nearer perturbation indicates more accurate execution state of the composite cloud system, such as service load, network status, etc. As illustrated in Fig. 4, the influencing factor parameter for the first perturbation and $(m-1)$th perturbation are α^{m-1} and α, respectively. The jth perturbation's influencing factor parameter is α^{m-j}.

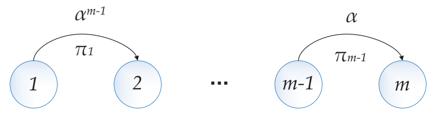


Fig. 4. Influencing factor parameter values.

The sensitivity value of component cloud service i by PARS approach is

$$I_{PARS}(i) = \frac{1}{m-1} \sum_{j=1}^{m-1} \alpha^{m-j} \pi_j. \quad (16)$$

We present the following example to further illustrate the PARS approach. Table 2 gives the evaluation results of the normalized perturbation values of a component cloud service and its composite cloud system.

TABLE 2
Normalized Perturbation Values

Examples	$\widehat{p}$	$j=1$	$j=2$	$j=3$	$j=4$
Example 1	$\widehat{p}(i,j)$	0.8	0.8	0.2	0.3
	$\widehat{p}(\Phi,j)$	0	0.6	0	0.7
Example 2	$\widehat{p}(i,j)$	0.8	0.8	0.2	0.3
	$\widehat{p}(\Phi,j)$	0.7	0.6	0.5	0.7
Example 3	$\widehat{p}(i,j)$	0.8	0.8	0.2	0.3
	$\widehat{p}(\Phi,j)$	0	0.6	0.7	0.8

If we set $\alpha = 0.8$, the sensitivity value $I_{PARS}(i)$ for component cloud service i in the above 3 examples calculated with Eq. (16) are 0.5627, 1.0523 and 1.1893, respectively. Comparing examples 1 vs. 2, we will find that the normalized p values for composite cloud system Φ in example 2 at the first and the third columns are larger than those in example 1. This indicates that component cloud service i in example 2 has a higher negative impact than that in example 1. Similar situations can be found in comparing example 1 vs. example 3, and example 2 vs. example 3.

Please note that in Eq. (16), π_j measures how much component cloud service i's jth reliability perturbation impacts the entire composite cloud system Φ's reliability, i.e., the reliability sensitivity of service i measured based on a separate perturbation. Parameter α is used to reflect the cumulative effects of continuous reliability perturbations. For the uncertain evolution regularities of component cloud service's reliability time series, Eq. (16) calculates the component cloud service's reliability sensitivity over a period of time based on parameter α. In general, for a specific application, the value of α should be a constant value. A component cloud service's reliability sensitivity changes with the length of time series and the value of α. The proper values of α and the length of time series should be determined based on an application-specific experimental analysis.

4 Proactive Adaptation Approach

In this section, we propose a proactive adaptation approach for a composite cloud system based on PARS. To guarantee the quality of a service-oriented system (e.g., response time, reliability, etc.), proactive adaptation aims at removing the risks of system execution quality declines, such as online faults, before a failure occurs. In the FP7 S-Cube project[2], a proactive adaptation architecture for service-oriented systems was proposed based on the reliability prediction mechanism [34]. We have also investigated the prediction-based proactive adaptation approach for service-oriented systems [9], [35]. In this paper, the proactive adaptation approach is fundamentally different from the previous works. In a composite cloud system, the reliability of each component cloud service dynamically changes over time. There may be seasonal fluctuations in their reliability time series. Changes,

2. https://www.s-cube-network.eu

gradual or sudden, may cause concept drifts in the distribution of their reliability time series. Traditional reliability time series prediction approaches suffer from low performance because they seldom cover these concept drifts training data in the historical data for constructing prediction models. PARS identifies system components at high risks of failing with a reliability sensitivity analysis based on up-to-date reliability time series. It can be used for adaptive cloud service selection for the proactive adaptation of the composite cloud system in the dynamic cloud environment.

4.1 The Approach

As illustrated in Fig. 5, PARS can pinpoint the potential risky (sensitive) system components in the composite cloud system. A proactive adaptation action will be triggered when the composite cloud system's reliability deviates from the SLA constraint. System reliability enhancement actions could be performed based on the reliability-sensitivity-aware proactive service selection to guarantee the quality of the composite cloud system. We present the steps taken by the proactive adaptation approach below. The proactive adaptation for a composite cloud system based on PARS is named PA-PARS and is summarized in Algorithm 1.

Step 1, risky system component identification. We use PARS to evaluate the reliability sensitivity for each component cloud service in the composite cloud system's execution plan. If a component cloud service's cumulative temporal reliability perturbations impact the reliability of composite cloud system significantly, it is considered a highly risky and a higher I_{PARS} value is assigned to the component cloud service. Such system components will risk the reliability of the composite cloud system.

Step 2, adaptation trigger. Let the average reliability of the composite cloud system be

$$\widehat{r}_{\Phi} = \frac{1}{m}\sum\nolimits_{j=1}^{m} r_{\Phi}^{\Delta t(j)}, \tag{17}$$

where $r_{\Phi}^{\Delta t(j)}$ represents the jth reliability value in the composite cloud system's nearest historical reliability time series (also known as data window time reliability time series). Let the required reliability for the composite cloud system Φ be r_{Φ}^{SLA}. In the situation when

$$\widehat{r}_{\Phi} < r_{\Phi}^{SLA}, \tag{18}$$

a proactive adaptation for the composite cloud system is triggered.

Step 3, candidate component cloud service selection. We select the top-k component cloud services with the highest reliability sensitivity in the composite cloud system. Suppose that $I_{PARS}(1)$, $I_{PARS}(2)$, $\cdots$, $I_{PARS}(k)$ represent the top-k reliability sensitivity values for the component cloud services in the composite cloud system workflow. For $l \in [1, k]$, s_l represents lth identified top-k component cloud services. We calculate the reliability sensitivity for each of s_l's candidate component cloud services. For any $j \in [1, n]$, let s_{lj} represents the jth candidate component cloud service of s_l, we replace s_l by s_{lj} and remain other component cloud services unchanged in the composite cloud system. We calculate the reliability sensitivity of s_{lj} as $I_{PARS}(l, j)$. Let the price and average reliability for s_{lj} be $Price(l, j)$ and $\widehat{r}_{(l,j)}$, respectively. Since the price is not a normalized value and a lower value of price indicates a better candidate component cloud service, we perform min-max normalization for $Price(l, j)$ similar to Eq. (6) and obtain the normalized price value $Price'(l, j)$ for s_{lj}. The optimal candidate component cloud service for s_l is:

$$OP(s_l) = \arg\max_{j} U(l, j), \tag{19}$$

where

$$U(l, j) = Price'(l, j) + \widehat{r}_{(l,j)} - I_{PARS}(l, j). \tag{20}$$

Step 4, NVP-based system construction as the proactive adaptation for the composite cloud system. For each of the top-k sensitive component cloud services s_l, $OP(s_l)$ will be bound as the redundant component for s_l. We will form a 1-out-of-2 system for each risky system component.

```
Input: $r_i^{TS}$ for $i \in [1, n]$, $r_{\Phi}^{TS}$, $r_{\Phi}^{SLA}$, $\alpha$, $k$, number of
    component cloud services $n$, $Price(l, j)$, $\widehat{r}_{(l,j)}$
Output: $OP(s_1), \cdots, OP(s_k)$
 1: for each $i \in [1, n]$ do
 2:     Calculate $I_{PARS}(i)$ by solving Eq. (16);
 3: end for
 4: Calculate $\widehat{r}_{\Phi}$ by solving Eq. (17);
 5: if Eq. (18) holds then
 6:     for each $l \in [1, k]$ do
 7:         $l = \arg\max_{i} (I_{PARS}(i))$;
 8:         Delete $I_{PARS}(i)$;
 9:         for each $j$ do
10:             Calculate $Price'(l, j)$;
11:             Calculate $\widehat{r}_{(l,j)}$;
12:             Calculate $I_{PARS}(l, j)$ by solving Eq. (16);
13:         end for
14:         Determine $OP(s_l)$ by solving Eq. (19);
15:     end for
16: end if
17: return $OP(s_1), \cdots, OP(s_k)$;
```

Algorithm 1: PA-PARS.

It is worth noting that, we do not just simply replace the risky component cloud services. Instead, we continue to have the original component cloud service and form a 1-out-of-2 system. By this 1-out-of-2 system, when at least one component cloud service could return a correct result, the whole 1-out-of-2 system functions. This is a basic cost-effective consideration. Cloud services are generally rented. When the contract for the original component cloud service has not expire, abandoning the rented cloud services (although the performance is not very good) would usually cause significant resource wastes. When the contract expires, a decision can be made based on the utility function of Eq. (20). If the newly selected redundant component cloud service is still optimal, we can directly remove the original component cloud service from the composite cloud system.

4.2 Reifying Through Application

The proposed PARS and proactive adaptation approaches can be used in the design and execution maintenance stages for composite cloud systems.

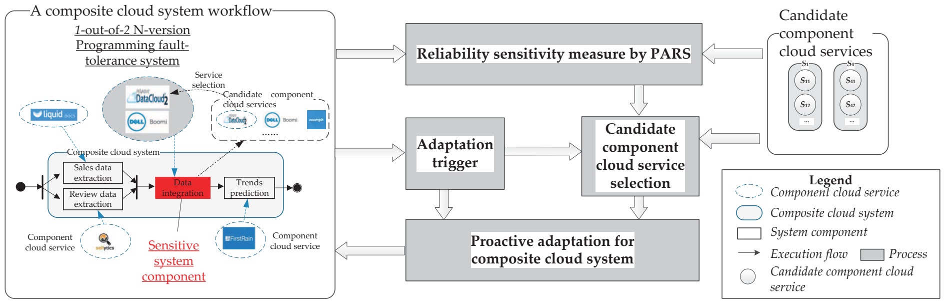

A composite cloud system workflow
1-out-of-2 N-version Programming fault-tolerance system
Service selection
Candidate component cloud services
Composite cloud system
Sales data extraction
Review data extraction
Data integration
Trends prediction
Component cloud service
Sensitive system component
Component cloud service
Reliability sensitivity measure by PARS
Adaptation trigger
Candidate component cloud service selection
Proactive adaptation for composite cloud system
Candidate component cloud services
S1
S11
S12
S4
S41
S42
Legend
Component cloud service
Composite cloud system
System component
Execution flow
Process
Candidate component cloud service

Fig. 5. The application for composite cloud system proactive adaptation.

First, in the design phase, PARS can provide important references for Web and/or Cloud service selection. As demonstrated in Fig. 1, during the construction of a composite cloud system, there are usually multiple candidate cloud services for each of the system components. By calculating the reliability sensitivity value for each candidate cloud service, we can evaluate the candidate cloud services and select the most reliable ones. In this example, Dell's Boomi API has the lowest reliability sensitivity and it will be selected as the component cloud service to perform data integration for the composite cloud system. It is worth noting that, to guarantee the long-term execution of the constructed composite cloud system, the reliability sensitivity measurement for cloud service selection should consider the long-term reliability problem [36]. In this case, the number of perturbations (i.e., the value m in Eq. (16)) in the historical reliability time series should be larger. We could also note the average perturbation from a long-term evaluation of each component cloud service. Then, we can calculate the reliability sensitivity of the component cloud service.

Second, in the dynamic cloud environment, a component cloud service's reliability time series follows the 1^{st}-order Markov Chain evaluation rule. This makes the reliability of each system component different during the construction and operation phases. During the execution of the composite cloud system, we need to calculate the real-time reliability sensitivity of each system component to be executed based on the reliability time series during the up-to-date data window time. And the proactive adaptation for the composite cloud system can be executed based on the calculated real-time reliability sensitivity and cloud service selection decision. By implementing NVP fault tolerance, the reliability of the risky component cloud services that may generate a failure during the execution is improved in advance to guarantee the stable execution of the composite cloud system.

If 1-out-of-2 fault tolerance [37] is deployed for a running component cloud service, service downtime is inevitable when service adaptation is needed at runtime. In practice, it is more cost-effective to employ PARS to analyse the reliability sensitivity of the component cloud service which is about to be executed in the workflow of a composite cloud system. If an execution risk is identified, 1-out-of-2 fault tolerance can be deployed before the execution. If a service failure occurs at runtime, trade-off decisions can be made between the impact of reliability and the adaptation cost.

5 Experimental Study

A set of experiments were conducted to investigate the effectiveness of the proposed approach. We compared our method with 4 representative approaches on two publicly available popular large-scale QoS datasets including QWS_v2 (with 2507 Web services) [38] and ICWS2012 (with 1770 Web services and at least 28 weeks' QoS data for each Web service) [39]. The experiments were implemented in Java on a PC equipped with Windows 7 x86 Enterprise Edition OS, and Intel(R) Core(TM) i7 2600 CPU, 12GB RAM, Seagate 1TB HDD.

5.1 Approaches Under Comparison

As discussed in Sections 1 and 6, existing criticality measurement approaches for service-oriented systems rely on the system architecture or QoS analysis [14], [21], [22], [23], [28], [40]. To facilitate the comparison, we adapted three representative approaches for reliability sensitivity measurement. The proactive adaptation was performed based on the results obtained by each of the criticality measurement approaches. Specifically, PARS is compared against the following four approaches:

- QoSranking: This approach identifies the critical component cloud services by the ranking of their average historical reliability [28]. We use min-max normalization to the average value of each component cloud service's reliability. Let the normalized reliability value be $\tilde{r}(i)$, the critical value for the service i would be $I_{QoSranking}(i) = 1 - \tilde{r}(i)$.
- ROCloud: Inspired by Google's PageRank algorithm, ROCloud [21], [22] calculates component cloud services' criticality values by incorporating software system structure information, system component failure rate, and the probability of the system component to cause application failures.
- Sensitivity-only: This approach only considers the cumulative perturbations for each component cloud service and set $\alpha = 1$ in Eq. (16) to measure the reliability sensitivity [14].
- Random: This approach randomly identifies the critical component cloud services in a composite cloud system. The criticality value for each component cloud service is assigned by a random value in the range of $[0, 1]$.

QoSranking is used to identify system components with low average reliability because they are likely to lead to system reliability bottleneck. To deal with the dynamic cloud environment, ROCloud and sensitivity-only approaches calculate the probability of a component causing application failures or the impact of system component's continuous perturbations on system performance.

5.2 Setup

The experiments were conducted through the following steps. First, we preprocessed to the datasets. We first randomly assign a price value in the range of [0,1] for each service in each dataset. As for the QWS dataset, we directly use the reliability values of each service. To simulate a service's reliability time series, we randomly generated a random value in the range of [0,1] for the service. Let the baseline reliability r_b for service i be $r_b(i)$, the random number be η, the upper bound for the perturbations for i is:

$$r_u(i) = [100 - r_b(i)] \times \eta + r_b(i), \tag{21}$$

and the lower bound is:

$$r_l(i) = \eta \times r_b(i). \tag{22}$$

We randomly simulate the reliability perturbations at service i's each time series point by a random value function $rand()$ (in the range of [0,1]) as

$$r_i^{\Delta t(j)} = r_l(i) + [r_u(i) - r_l(i)] \times rand(). \tag{23}$$

For the ICWS2012 dataset, 28 weeks data of 1770 real SOAP-based and RESTful Web services in QoS_{wsdl}, QoS_{wadl}, and QoS_{text} are extracted. As for each Web service, we set the maximal response time constraint $RT_{max} = 1000ms$. Using Eq. (3), 4 continuously time slots' *successability*, and *response time* parameters are used together with the RT_{max} constraint to calculate the value at a specific reliability time series point. Consequently, we obtained the values at 7 ($= 28/4$) reliability time series points for each service. Like the preprocessing of QWS dataset, we also increase the length of reliability time series by simulation. We set $r_u(i)$ and $r_l(i)$ as the maximal and the minimal values of the 7 time series points for service i, respectively. Then, Eq. (23) was used to generate the simulation reliability values for the subsequent time series points.

Second, we generated directed scale-free random network graphs to simulate the composite cloud systems. The widely used program package for analyzing and visualizing large networks, namely Pajek[3], was used for the simulation. Using a dataset, we randomly assigned a component cloud service to each of the nodes in the random network graph. The arcs in the graph indicates the system components' invocation relationships. Let $r_j^{\Delta t(i)}$ denote a component cloud service j's ith reliability value in the time series. The failure rate is calculated as

$$\lambda_j^i = -\ln \frac{r_j^{\Delta t(i)}}{len(\Delta t)}. \tag{24}$$

The failure rate and the system composite structure were used to calculate the composite cloud system's failure rate based on the flowQoS algorithm proposed in [41] and Eq. (9) to (12). The composite cloud system's reliability can then be calculated in a similar way as Eq. (3).

Finally, PARS and other representative approaches were used to calculate the reliability sensitivity of each component cloud service under the above simulated historical reliability time series in different datasets. The proactive adaptation algorithm PA-PARS with different reliability sensitivity measurement approaches was employed for the system components with high reliability sensitivity values. To investigate the effectiveness of PARS, we simulated each component cloud service's near-future reliability time series with the above approaches. The accuracy of the reliability sensitivity measurement and the effectiveness of the proactive adaptation approach were investigated and compared under the near future reliability time series. The operation of random assignment of the system components was repeated 200 times in each experiment and the results were averaged. During the experiments, the SLAs for composite cloud system for the system reliability is set as 60%.

5.3 Metrics

We employ the *success rate* and the *overall reliability enhancement* to evaluate the performance of the approaches.

Let $N_{\text{SLA deviation}}$ be the number of composite cloud systems, which were not fault-tolerated with NVP, that meet with SLA deviation (i.e., $\exists j \in [1, m]$, s.t. $r_\Phi^{\Delta t(j)} < r_\Phi^{SLA}$) under the historical reliability time series, $N_{\text{successfully recovered}}$ be the number of fault-tolerated composite cloud systems that successfully avoided system failures under the near-future reliability time series. The *success rate* (or SR) is defined as:

$$SR = \left({}^{N_{\text{successfully recovered}}} \big/ {}_{N_{\text{SLA deviation}}} \right) \times 100\%. \tag{25}$$

And the *overall reliability enhancement* (or ORE) is defined as:

$$ORE = \left[{}^{(\hat{r}_\Phi^{'} - \hat{r}_\Phi)} \big/ {}_{\hat{r}_\Phi} \right] \times 100\%, \tag{26}$$

where $\hat{r}_\Phi$ represents average reliability of the composite cloud system's historical reliability time series, and $\hat{r}_\Phi^{'}$ represents the composite cloud system's near-future reliability time series, which is fault-tolerated with NVP.

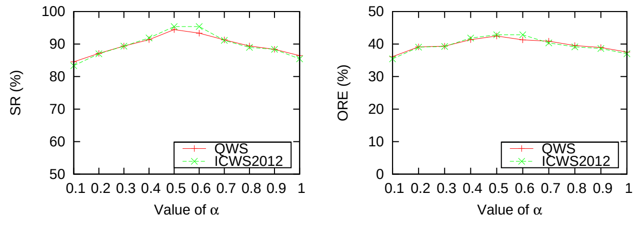
5.4 Impact of α

This experiment investigates how parameter α influences the performance of PARS. We generated a random network graph with 120 nodes and 140 arcs to simulate the workflows of composite cloud systems. The lengths of the historical and the near-future reliability time series are all set as 30. The k value for the top-k high-risk component cloud services number is set as 10. We vary the α value from 0.1 to 1, in steps of 0.1, and execute the reliability sensitivity measurement and proactive adaptation on the generated composite cloud systems. The results of *success rate* and *overall reliability enhancement* are averaged and compared.

As can be seen from Fig. 6, when the value of α is near 0.5, the success rates and the overall reliability enhancements by PARS are the highest. This indicates the effectiveness of the use of α in the reliability sensitivity measurement. It is worth noting that, the fluctuations in the reliability time series in this experiment follow a normal

3. http://vlado.fmf.uni-lj.si/pub/networks/pajek/

distribution. If PARS is used in cases where the time series follow other distributions, the optimal α value may be different, and it should be experimentally determined.



(a) SR by PARS (b) ORE by PARS

Fig. 6. Impact of the influencing factor parameter α.

5.5 Impact of k

This experiment examines how the k value of top-k most sensitive component cloud services impacts the performance of PARS. The nodes and arcs in the composite cloud systems' workflows are set as 300 and 360, respectively. The length of historical time series and near-future time series both all set as 30. We fix $\alpha = 0.5$ and vary the k value from 10 to 60, in steps of 10. The results are averaged and compared in Fig. 7.

As can be seen from the results, both SR and ORE increase with the increase in the value of k on two different datasets. PARS achieves the best performance, followed by ROCloud, QoSranking, Sensitivity-only, and Random. When $k >= 40$, the increasing trends of PARS's SR and ORE slow down. Since a larger value of k would include more redundant component cloud services and increase the total rental cost and system load. As for the composite cloud systems with higher reliability requirements, the k value should be large. Otherwise, the value can be set lower. Eventually, this should meet the requirements specified in the SLAs.

5.6 Impact of System Size

This experiment investigates how the size of composite cloud system influences the performance of PARS. We fix $\alpha = 0.5$, $k = 20$, the lengths of historical reliability time series, at 30, and vary the number of nodes in the composite cloud system workflows from 60 to 260, in steps of 40. For each workflow, we set the number of arcs as the 1.2 times as the corresponding nodes number. The SR and ORE achieved on two different datasets are shown in Fig. 8.

As can be seen from the results, PARS achieves better performance than other approaches. For all the approaches, the SR and ORE all gradually decline with the increase in the system size. The downward trend is more obvious when the node number reaches 140. In the case of large-scale composite cloud systems, we should increase the k value, so that more sensitive component cloud services could be fault-tolerated with NVP.

5.7 Impact of Time Series Length

To investigate how the length of historical reliability time series influences the performance of PARS, we varied the length of history and near future reliability time series for from 10 to 60, in steps of 10. We fixed the number of the nodes and arcs in the composite cloud system as 120 and 140, respectively, the value of k as 30, and the α as 0.5.

As can be seen in Fig. 9, PARS's ORE is obviously higher than other approaches. This suggests the effectiveness of PARS. The SR and ORE for PARS, QoSranking, ROCloud, and Sensitivity-only approaches increase with the increase in the length of historical reliability time series. When the length reaches 30 *time points*, a further extended length of the time series has little impact on SR and ORE. This indicates that a sufficient length of data window time for collecting the historical reliability time series is necessary for guaranteeing the performance of PARS.

5.8 Discussion

Now, let us discuss the findings in the experiments.

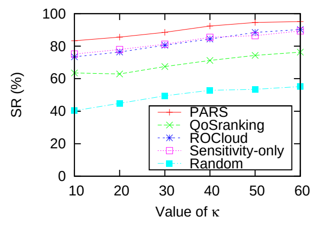
First, PARS can measure the reliability sensitivity of the component cloud services in a composite cloud system by considering the cumulative effects of temporal perturbations based on the up-to-date historical reliability time series. Service selection and proactive adaptive mechanism for composite cloud system are performed based on PARS. PARS can be used for online operation and maintenance of composite cloud systems, which are helpful to improve their system quality and users' QoE.

Second, the experimental results verify the effectiveness of PARS. In the case of increasing the length of historical reliability time series, the performance of PARS improves. PARS is more suitable for measuring short-term reliability sensitivity of component cloud services. It can be used to solve the problem of adaptive service selection for proactive execution quality assurance of composite cloud systems.

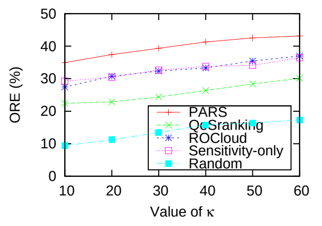
Finally, considering the concept drifting nature of component cloud services' reliability, the optimal value of influencing factor parameter α for a component cloud service may change over time. The SR of PARS is also influenced by the length of the historical reliability time series. In practice, when size of the window for reliability time series is set properly, we can select multiple samples of historical reliability time series during different time periods and adopt a clustering algorithm to classify the classes of the samples. For each class of the reliability time series, we can determine the optimal α value experimentally. When PARS is used to measure the reliability sensitivity for component cloud services, we can first determine the class of the up-to-date reliability time series, and then set the optimal α value which was identified earlier for the corresponding class. This helps to deal with the uncertain evolution problem of component cloud service's reliability time series and improve the performance of PARS.

6 Related Work

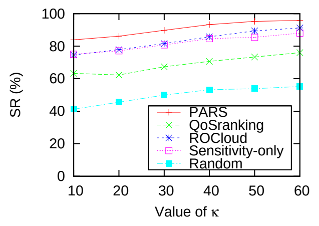
In this section, we briefly review the related works that are relevant with our proposed PARS approach, including the needs of execution quality assurance for composite cloud systems, PageRank-based system component criticality measure approaches and reliability sensitivity analysis approaches used for traditional computer systems.



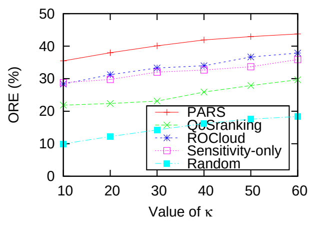
(a) QWS dataset



(b) QWS dataset

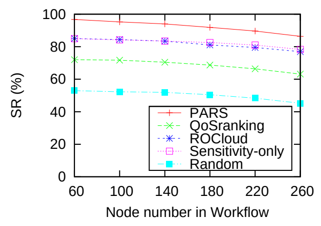


(c) ICWS2012 dataset

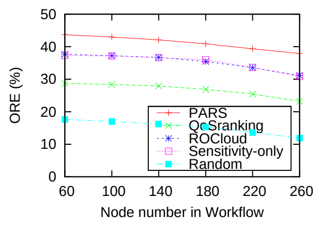


(d) ICWS2012 dataset

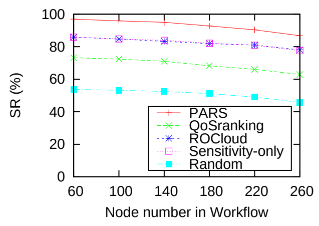
Fig. 7. Impact of the number of top sensitive component cloud services.



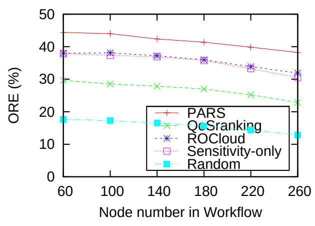
(a) QWS dataset



(b) QWS dataset

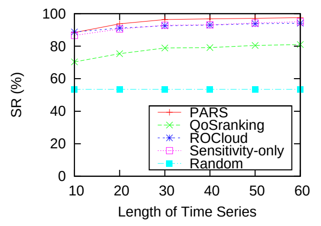


(c) ICWS2012 dataset

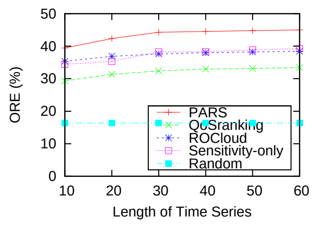


(d) ICWS2012 dataset

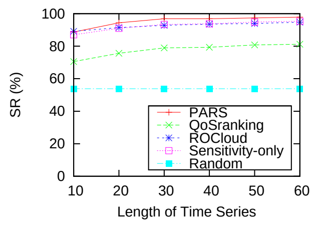
Fig. 8. Impact of the size of composite cloud systems.



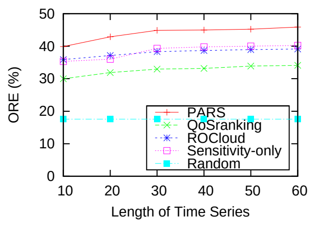
(a) QWS dataset



(b) QWS dataset



(c) ICWS2012 dataset



(d) ICWS2012 dataset

Fig. 9. Impact of the length of historical reliability time series.

Differing from the traditional component-based system, each component cloud service in a composite cloud system is usually rented from a third-party vendor. By invoking the interfaces of the component cloud services through Internet protocols, a cloud service execution engine composes different system components to create a composite cloud system. Therefore, operating in the dynamic cloud environment, the system components of a cloud system are much more loosely coupled than simple in-house software [42].

In the cloud service industry, Service Level Agreements (SLAs) management has already been included in service management suites to deal with the concept drifting probability distribution component cloud services' reliability time series, e.g., IBM Integrated Service Management/ISM, ServiceNow, and so on [43]. Research works have investigated on how to guarantee correct and continuous operations for composite cloud systems in the presence of faulty components [44], [45]. Based on DevOps, the AIOps project in Microsoft also plans to develop intelligent methods for improving service quality and customer satisfaction [46].

Traditional software engineering methodologies have obtained some insights about system reliability by analyzing software architectures. By analyzing the dependencies among system components, Bayesian networks or Palladio Component Model are employed to predict the online failure of a software by combining component failure predictions with software architectural knowledge [24], [47], [48].

To identify critical system components in a service-oriented system workflow and design an effective service fault-tolerance (FT) strategy, Z. Zheng et al. [49] proposed an architecture-based PageRank-like approach (i.e., FTCloud). By analyzing the system components' invocation relations and frequencies based on the composite system architecture information, top 20% critical system components were identified. The FT strategy was deployed on these top 20% critical components based on the known 80/20 principle. This approach was extended by integrating the system component failure rates and reliability properties in ROCloud [22].

Reliability sensitivity analysis approaches have already been used for traditional computer systems. The B-reliability importance [17] and FV-reliability importance [50], [51] measures are the representative early works. These approaches consider how much a system component's execution failures influence the whole system's reliability.

To design cost-effective cloud systems, reliability sensitivity analysis approach has been used to identify critical component cloud services in a composite cloud system. In [14], a Quality Degradation Coefficient was defined to indicate the maximum QoS degradation level. Multiple iterations of Quality Degradation Coefficient for a system component was noted and divided by their original values.

The results were then summed and averaged.

Existing PageRank-based system component criticality measurement approaches aim to identify frequently-invoked component cloud services through analyzing the historical composite cloud systems' workflow structures. Critical system components provide important references for constructing stable composite cloud systems. Faults in those system components threaten the stable execution of the composite cloud systems. Hence, such approaches are designed for determining the critical components that should be particularly maintained in the cloud. Moreover, the ROCloud approach also considers the probability of system components causing application failures when measuring component criticality. In the dynamic cloud environment, where a composite cloud system operates, it is critical to analyze the reliability sensitivity of its system components. However, this is ignored by PageRank-based approaches. To guarantee the stable operation of the composite cloud system in real-time, PARS analyzes the impacts of individual system component's reliability perturbations on the runtime reliability of the composite could system.

In this paper, we propose PARS, a temporal-perturbation aware approach for measuring the reliability sensitivity of component cloud services, which analyzes the cumulative negative impacts in their up-to-date reliability time series. The proposed approach can identify the root cause to system performance anomalies in system component level in a composite cloud system. The results can provide guidance for adaptive cloud service selection for building composite cloud systems and assuring execution quality; henceforth introducing a new way for our proactive adaptation of composite cloud systems.

7 Conclusion and Future Work

To ensure the quality of composite cloud systems in the dynamic cloud environment, this paper proposed a composite cloud system proactive adaptation approach based on reliability sensitivity assessment for component cloud services. We proposed an approach named PARS for Perturbation-aware Reliability Sensitivity measurement which considers up-to-date window for component cloud services' reliability time series. The time-homogeneous reliability perturbation and the cumulative effects of historical reliability perturbations are particularity employed by PARS for reliability sensitivity calculation. The 1-out-of-2 system NVP fault tolerance mechanism is implemented on the identified top-k most risky system components. The experimental results demonstrate the effectiveness of PARS.

We identify the following research directions for the proactive adaptation of composite cloud systems. First, for the top-k sensitive component cloud services, the online reliability prediction approach can be investigated and integrated with PARS. Second, online reliability sensitivity prediction approaches will be investigated based on PARS to improve our proactive adaptation approach for composite cloud systems. Third, different fault-tolerance and service replacement approaches will be investigated and leveraged to allow PARS to handle more sophisticated scenarios.

Acknowledgments

This work was partially supported by the Humanity and Social Science Fund of Ministry of Education of China (No. 18YJCZH170), Six talent peaks project in Jiangsu Province in 2019 (No. RJFW-029). L. Wang is the corresponding author.

References

[1] A. Bouguettaya, M. P. Singh, M. N. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, M. B. Blake, S. Dustdar, F. Leymann, and M. P. Papazoglou, "A service computing manifesto: the next 10 years," *Commun. ACM*, vol. 60, no. 4, pp. 64–72, 2017.

[2] X. Xu, Q. Z. Sheng, L. Zhang, Y. Fan, and S. Dustdar, "From big data to big service," *IEEE Computer*, vol. 48, no. 7, pp. 80–83, 2015.

[3] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction." *IEEE Trans. Software Eng.*, vol. 40, no. 2, pp. 192–215, 2014.

[4] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75–81, 2005.

[5] M. P. Papazoglou and W.-J. Heuvel, "Service oriented architectures: Approaches, technologies and research issues," *The VLDB Journal*, vol. 16, no. 3, pp. 389–415, 2007.

[6] P. Wang, J. Meng, J. Chen, T. Liu, Y. Zhan, W.-T. Tsai, and Z. Jin, "Smart contract-based negotiation for adaptive qos-aware service composition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1403–1420, 2019.

[7] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J. C. Grundy, and Y. Yang, "Keyword search for building service-based systems," *IEEE Trans. Software Eng.*, vol. 43, no. 7, pp. 658–674, 2017.

[8] L. Qi, Q. He, F. Chen, W. Dou, S. Wan, X. Zhang, and X. Xu, "Finding all you need: web apis recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[9] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu, "Online reliability prediction via motifs-based dynamic bayesian networks for service-oriented systems," *IEEE Trans. Software Eng.*, vol. 43, no. 6, pp. 556–579, 2017.

[10] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134 – 150, 2014.

[11] Q. Yu and A. Bouguettaya, "Multi-attribute optimization in service selection," *World Wide Web*, vol. 15, no. 1, pp. 1–31, 2012.

[12] D. Lin, A. C. Squicciarini, V. N. Dondapati, and S. Sundareswaran, "A cloud brokerage architecture for efficient cloud service selection," *IEEE Trans. Serv. Comput.*, vol. 12, no. 1, pp. 144–157, Jan 2019.

[13] Z. Zheng, K. S. Trivedi, K. Qiu, and R. Xia, "Semi-markov models of composite web services for their performance, reliability and bottlenecks," *IEEE Trans. Services Computing*, vol. 10, no. 3, pp. 448–460, 2017.

[14] Y. Wang, Q. He, D. Ye, and Y. Yang, "Formulating criticality-based cost-effective fault tolerance strategies for multi-tenant service-based systems," *IEEE Trans. Software Eng.*, vol. 44, no. 3, pp. 291–307, 2018.

[15] P. M. Frank, *Introduction to system sensitivity theory*. Academic Press, 1978.

[16] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, p. 44, 2014.

[17] Z. BIRNBAUM, "On the importance of different components in a multicomponent system," *Multivariate Analysis-II*, pp. 581–592, 1969.

[18] M. R. Lyu, *Handbook of software reliability engineering*. IEEE computer society press and McGraw-Hill Book Company, 1996.

[19] K. S. Trivedi and A. Bobbio, *Reliability and availability engineering : modeling, analysis, and applications*, 1st ed. Cambridge University Press, 2017.

[20] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, 2010, pp. 35–44.

[21] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "Component ranking for fault-tolerant cloud applications," *IEEE Trans. Services Computing*, vol. 5, no. 4, pp. 540–550, 2012.

[22] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reliability-based design optimization for cloud migration," *IEEE Trans. Services Computing*, vol. 7, no. 2, pp. 223–236, 2014.

[23] W. Kuo and X. Zhu, *Importance measures in reliability, risk, and optimization: principles and applications*. John Wiley & Sons, 2012.

[24] F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner, "Architecture-based reliability prediction with the palladio component model," *IEEE Trans. Software Eng.*, vol. 38, no. 6, pp. 1319–1339, 2012.

[25] ANSI/IEEE, "Ieee standard glossary of software engineering terminology," *ANSI/IEEE STD-729-1991*, 1991.

[26] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[27] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[28] H. Wang, C. Yu, L. Wang, and Q. Yu, "Effective bigdata-space service selection over trust and heterogeneous qos preferences," *IEEE Trans. Serv. Comput.*, vol. 11, no. 4, pp. 644–657, 2018.

[29] M. R. Lyu, *Software fault tolerance*. John Wiley & Sons, Inc., 1995.

[30] J. D. Musa, *Software reliability engineering: more reliable software, faster and cheaper*. Tata McGraw-Hill Education, 2004.

[31] J. Knight, *Fundamentals of Dependable Computing for Software Engineers*. Chapman and Hall/CRC, 2012.

[32] S. Yamada, *Software reliability modeling: fundamentals and applications*. Springer, 2014.

[33] A. Avižienis and L. Chen, "On the implementation of n-version programming for software fault tolerance during execution," in *Proceedings of the International Computer Software and Applications Conference (COMPSAC 77)*. IEEE, 1977, pp. 149–155.

[34] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Comput. Surv.*, vol. 42, no. 3, pp. 10:1–10:42, 2010.

[35] H. Wang, L. Wang, Q. Yu, Z. Zheng, and Z. Yang, "A proactive approach based on online reliability prediction for adaptation of service-oriented systems," *J. Parallel Distributed Comput.*, vol. 114, pp. 70 – 84, 2018.

[36] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term qos-aware cloud service composition using multivariate time series analysis," *IEEE Trans. Services Computing*, vol. 9, no. 3, pp. 382–393, 2016.

[37] P. T. Popov, "Models of reliability of fault-tolerant software under cyber-attacks," in *28th IEEE International Symposium on Software Reliability Engineering, ISSRE 2017, Toulouse, France, October 23-26, 2017*, 2017, pp. 228–239.

[38] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 795–804.

[39] W. Jiang, D. Lee, and S. Hu, "Large-scale longitudinal analysis of soap-based and restful web services," in *IEEE 19th International Conference on Web Services (ICWS)*, 2012, pp. 218–225.

[40] Q. He, J. Han, Y. Yang, H. Jin, J.-G. Schneider, and S. Versteeg, "Formulating cost-effective monitoring strategies for service-based systems," *IEEE Trans. Software Eng.*, vol. 40, no. 5, pp. 461–482, 2014.

[41] Z. Zheng and M. R. Lyu, "Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints," *IEEE Trans. Computers*, vol. 64, no. 1, pp. 219–232, 2015.

[42] Z. Ye, A. Bouguettaya, and X. Zhou, "Economic model-driven cloud service composition," *ACM Trans. Internet Techn.*, vol. 14, pp. 1–19, 10 2014.

[43] H. Ludwig, K. Stamou, M. Mohamed, N. Mandagere, B. Langston, G. Alatorre, H. Nakamura, O. Anya, and A. Keller, "rsla: Monitoring slas in dynamic service environments," in *International Conference on Service-Oriented Computing*. Springer, 2015, pp. 139–153.

[44] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Systems Journal*, vol. 7, no. 2, pp. 288–297, 2013.

[45] Y. Zhang, Z. Zheng, and M. R. Lyu, "Bftcloud: A byzantine fault tolerance framework for voluntary-resource cloud computing," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 2011, pp. 444–451.

[46] Y. Dang, Q. Lin, and P. Huang, "Aiops: real-world challenges and research innovations," in *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*. IEEE Press, 2019, pp. 4–5.

[47] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske, "Hora: Architecture-aware online failure prediction," *J. Syst. Softw.*, vol. 137, pp. 669–685, 2018.

[48] T. Pitakrat, D. Okanovic, A. Van Hoorn, and L. Grunske, "An architecture-aware approach to hierarchical online failure prediction," in *12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. IEEE, 2016, pp. 60–69.

[49] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "Ftcloud: A component ranking framework for fault-tolerant cloud applications," in *IEEE 21st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2010, pp. 398–407.

[50] J. B. Fussell, "How to hand-calculate system reliability and safety characteristics," *IEEE Trans. Rel.*, vol. 24, no. 3, pp. 169–174, 1975.

[51] W. E. Vesely, "A time-dependent methodology for fault tree evaluation," *Nuclear Eng. Des.*, vol. 13, no. 2, pp. 337–360, 1970.

Lei Wang received the PhD degree in computer science from Southeast University, China. He is an associate professor with Nanjing Forestry University, China. He was an honorary visiting scholar at the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2019. His research interests mainly include service computing, software reliability engineering, and data mining. His publications have appeared in well-known journals and popular conferences.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China, in 2010. He is a senior lecturer at Swinburne. His research interests include edge computing, cloud computing, software engineering and service computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

Demin Gao received the PhD degree at the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China in 2012. He visited University of Wollongong, Australia, during 2011 to 2012, and University of Minnesota Twin Cities, Minneapolis, USA from 2016 to 2017. He is an associate professor with Nanjing Forestry University. His current research fields contain delay-tolerant sensor networks and dependable distributed computing.

Jing Wan is currently pursuing the master's degree with Nanjing Forestry University, China. Her research interests mainly include indicator system and assessment for the health of software ecosystem, software reliability engineering, and credibility evaluation and salary accounting for software crowdsourcing.

Yunqiu Zhang is currently pursuing the master's degree with Nanjing Forestry University, China. Her research interests mainly include software reliability engineering and crowdsourced service recommendation.