资源重配置算法在系统可存活性中的应用*)

胡方炜　李千目　许满武
(南京大学软件新技术国家重点实验室　南京 210093)

摘　要　可存活性是用来表明系统在面对蓄意攻击、故障失效或偶发事故时仍能完成其任务的能力。如何提高系统的可存活性是当前安全研究领域的重点。目前提高系统可存活性的方法主要集中在体系结构的设计和资源重配置上。从资源重配置的角度,通过剥夺非关键服务的资源分配给资源受到破坏的关键服务,从而保障关键服务持续运行,实现系统的可存活性。从被剥夺资源的非关键服务尽可能少这一角度出发,利用近似度算法得到一个分配方案;从非关键服务对于关键服务的响应时间尽可能短这一角度考虑,采取时间优先算法得到另一个方案。在近似度算法的实现过程中,同时加入对关键服务响应时间的考虑;在时间优先算法的实现过程中,考虑服务间占有资源的近似度。然后比较两种算法产生的两个方案,选取综合性能较好的一个作为最终的资源分配方案。无论是近似度算法还是时间优先算法,实现容易、时间及空间效率较高、产生的分配方案综合性能较优。

关键词　可存活性,资源重配置,近似度算法,时间优先算法

Application of Resources Reconfiguration Algorithms in System's Survivability

HU Fang-wei　LI Qian-mu　XU Man-wu
(State Key Laboratory of Computer Software and New Technology, Nanjing University, Nanjing 210093, China)

Abstract　Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. How to increase the system's survivability is the point research in the realm of security. Presently the method to increasing the survivability is mainly concentrated on the design of system architecture or resource reconfiguration. Based on the view of resource reconfiguration, maintain the critical service running continuously to realize the system's survivability, through depriving non-critical services of their resources reconfigured to the critical service. From the point that the number of non-critical services deprived of resources is as small as possible, introduces the approximation degree algorithm(ADA) to get a scheme, and from the point that the time the non-critical services response to the critical is as short as possible, introduces time first algorithm(TFA) to get another scheme. In the procedure of ADA realization, we considered the response time of the non-critical meanwhile, also in the procedure of TFA realization, we considered the approximation degree of resources between services. Then we compared the two schemes, and chose one which is better in the comprehensive performance as the final scheme. Whether ADA or TFA, the algorithm is easy to realize, efficiency of time and space higher, and the final scheme is better in the over-all performance.

Keywords　Survivability, Resource reconfiguration, Approximation degree algorithm, Time first algorithm

1　引言

信息技术已经深入到社会生活的方方面面,给人们的生活带来方便的同时,也使得系统自身的安全受到了很大的威胁。

传统的安全技术重在防御。但是实验表明,系统很难保证不受到攻击,在受到攻击时亦不能保证其绝对安全[1]。如何在开放的复杂环境下使得系统面临攻击时仍然能够运转,即系统的可存活性,成为当前系统安全性研究的重点。

可存活性是用来表明系统在面对蓄意攻击、故障失效或偶发事故时仍能完成其任务并及时恢复整个服务的能力[2]。

系统可以提供多项服务,这些服务可以划分为两类:关键服务和非关键服务。关键服务是指当系统环境受到破坏,故障发生时,系统必须维持的功能[2]。可存活性的主要目标就是当入侵成功,甚至系统主要部件受到破坏后,系统仍然能够提供关键服务,并且能够及时恢复受损的服务。

如何分析现有系统的可存活性,以及如何提高系统的可存活性,成为当前可存活性研究领域的两大课题。

目前提高系统可存活性的方法主要集中在两个方面:体系结构的设计和资源重配置。

体系结构的设计可以有效提高系统的可存活力,使得系统面对攻击时仍能继续提供服务,但是体系结构的研究与设计花费巨大、实现困难且不易验证。

资源重配置亦可来实现系统的可存活性。只要系统面临攻击时仍然能够重新分配资源,最终保证关键服务获得其所需资源正常运行,系统便获得了可存活力。

重配置方法又可细分为两种。一种是配置冗余,即配置多样化,从多种配置中选出一种具有最大可存活性的配置。Lu Tun[3], David Wells [4]等人在此方向做了详细的研究。另一种是通过重新分配资源给关键任务来保证攻击成功后系统

*)基金项目:国家自然科学基金重大研究项目(90718021)。**胡方炜**　硕士研究生,主要研究领域为软件方法学、信息安全;**李千目**　博士后,主要研究领域为信息安全、网络性能分析;**许满武**　教授,博士生导师,主要研究领域为软件方法学、信息安全。

仍然能够完成关键的服务。Wang Jian 等人[5]提供了 ERAS 的方法，Matti A. Hiltunen 等人[6]则通过 Cactus 方法实现资源重新配置。

本文从服务所需资源的角度来考虑可存活性。一旦系统受到攻击，关键服务的资源受到破坏，系统能够动态调配资源，即剥夺非关键服务拥有的资源给关键服务，以保证关键服务获得所需资源后能够正常运行。待到攻击破解后，系统能够通过可存活性技术恢复非关键服务。

在将资源重配置算法引入到可存活性研究的过程中，必须考虑两个限制条件：

1)非关键服务的响应时间应尽可能短；

2)被剥夺资源的非关键服务应尽可能少。

否则，关键服务得不到及时的资源补给便会暂停服务；被剥夺资源的非关键服务太多将会影响系统的稳定性。

本文充分考虑了上述两个限制条件，采用了近似度算法和时间优先两种算法，较高效率地实现了资源重配置，从而提高了系统的可存活性。

2 问题定义

2.1 问题描述

问题可以描述为：

1)某系统有 N 种资源，设为 $X_1, X_2, \cdots, X_N$。

2)系统中有一关键服务 A，当系统受到攻击后，服务 A 的资源受到了破坏。现在服务 A 需要各种资源的数目分别为 $x_1, x_2, \cdots, x_N$，记作向量 $x=(x_1, x_2, \cdots, x_N)$；其中 $x_i \geqslant 0$，$x_i=0$ 表示服务 A 不需要 X_i 资源。

3)系统中有 m 个非关键服务 $B_1, B_2, \cdots, B_m$，每一个非关键服务 B_i 拥有各种资源数目为 $y_{i1}, y_{i2}, \cdots, y_{iN}$，记作向量 $y_i=(y_{i1}, y_{i2}, \cdots, y_{iN})$；其中 $y_{ij} \geqslant 0$，$y_{ij}=0$ 表示 B_i 没有 X_j 资源。

4)m 个非关键服务对于关键服务 A 的响应时间分别为 $t_1, t_2, \cdots, t_m$，记作向量 $t=(t_1, t_2, \cdots, t_m)$。

5)一个前提条件：

对于任一项资源 X_j，都要求“供”大于等于“求”，即关键服务 A 所需求的资源是能够通过剥夺非关键服务的资源得到满足的。

用公式表达如下：

$$\forall j \in [1, N], \sum_{i=1}^{m} y_{ij} \geqslant x_j$$

6)问题就是要选取一些非关键服务，将这些服务占有的资源重分配给关键服务 A 使用，同时要求非关键服务的响应时间尽可能地短，被剥夺资源的非关键服务尽可能地少。

2.2 模型定义

对于任意一个可行方案 $\Phi=\{d_1, d_2, \cdots, d_k\}$，其中 d_i 表示被剥夺资源的非关键服务编号，即 $B_{d_1}, B_{d_2}, \cdots, B_{d_k}$ 被剥夺资源。

定义 1 设方案 Φ 中，影响到的非关键服务(即被剥夺资源的服务)个数为 $S(\Phi)$，非关键服务的响应时间为 $T(\Phi)$，$T(\Phi)$指所有所需资源被传送到关键服务所需要的时间。

由于多个服务之间是并行的，所以从定义 1 可以得到如下结论：

1)$S(\Phi)=k$

2)$T(\Phi)=\max_{i=d_1}^{d_k} t_i$

所以此问题演化成为求一个分配方案 Φ，使得该方案满足以下 3 个条件：

1)$\min(S(\Phi))$，即影响服务尽可能少；

2)$\min(T(\Phi))$，即服务响应时间尽可能短；

3)$\forall j \in [1, N], \sum_{i=d_1}^{d_k} y_{ij} \geqslant x_j$，前提条件。

但是通过实验可以看出，一般情况下，没有能够同时满足条件 1)和条件 2)的分配方案。那么我们可以用一些启发式的方法对两个条件进行综合考虑。

定义 2 设在方案 Φ 中，影响到的非关键服务个数为 $S(\Phi)$，非关键服务响应时间为 $T(\Phi)$，则定义加权和 $p*S(\Phi)+q*T(\Phi)$为该方案的评价参数，其中 $p+q=1, 0<p<1, 0<q<1$。

由上述 3 个条件可知，方案的评价参数愈小，方案愈优。

本文中，取 $p=0.4, q=0.6$，即评价参数为

$$Acc=0.4*S(\Phi)+0.6*T(\Phi) \tag{1}$$

因为在可存活系统中，强调一个“及时”，响应时间显得更为重要，所以分配权重大于影响服务数。

3 资源重配置算法

一般情况下，很难找到一个方案既能够影响尽可能少的非关键服务，又能够使得非关键服务响应时间尽可能地短。但是，存在这样一种方案，能够满足其中一个条件，即要么影响到较少的服务，要么使得服务响应时间较短。

本文采用近似度算法，使得选出的方案影响到的非关键服务尽可能少；采用时间优先算法，使得选出的方案中非关键服务的响应尽可能得快。

3.1 近似度算法

近似度算法在人工智能领域十分流行，在属性选择、自动生成摘要等研究中皆有应用。近似度算法主要用于计算两者之间的差距，从而得到两者间近似的程度。两者的差距越小，说明两者越近似。

本文从资源分配的角度，剥夺非关键服务的资源给关键服务。通过计算关键服务与非关键服务占有资源之间的近似程度，搜索与关键服务需要的资源最为近似的某些非关键服务，剥夺其资源分配给关键服务。通过近似度方法得到的启发式方案将影响较少数量的非关键服务。

3.1.1 近似度的定义

近似度算法种类繁多，有欧式距离法、向量空间法等等。本文需要计算关键服务与非关键服务占有的资源之间的近似程度。若关键服务 A 占有的资源用向量 x 表示，非关键服务 B_i 占有的资源用 y_i 来表示，那么就需要计算向量 x 和 y_i 之间的近似程度。鉴于两者都是向量，我们基于欧式距离法来计算向量之间的近似度。

定义 3 欧氏距离(Euclidean distance)是在 n 维空间中两个点之间的真实距离。设 n 维空间中向量 $X=(x_1, x_2, \cdots, x_n)$，向量 $Y=(y_1, y_2, \cdots, y_n)$，则这两个向量之间的欧氏距离为：

$$E_d=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{2}$$

基于欧式距离，我们定义资源相似度 dis。

定义 4 关键服务 A 需要的资源，设为向量 $x=(x_1, x_2, \cdots, x_N)$；某一非关键服务 B_i 拥有的资源，设为向量 $y_i=(y_{i1}, y_{i2}, \cdots, y_{iN})$，则 A 与 B_i 拥有资源的近似度 dis_i 为：

$$dis_i=\sum_j(x_j-y_{ij})^2 \quad j\in[1,N]\text{且 } x_j\geqslant y_{ij} \tag{3}$$

其中 dis_i 越小，说明 B_i 与 A 越相似，即两者拥有的资源越接近，则首先选取 B_i 的资源分配给 A 使用。

加入限制条件 $x_j\geqslant y_{ij}$，是从实际应用角度考虑。

若关键服务 A 的资源 $x=(2,1,2)$，非关键服务 B_1 的资源 $y_1=(2,3,2)$，B_2 的资源 $y_2=(1,0,1)$。如果不加此限制条件，根据近似度公式计算可得：

$dis_1=(2-2)^2+(1-3)^2+(2-2)^2=4$；

$dis_2=(2-1)^2+(1-0)^2+(2-1)^2=3$；

由此可见 $dis_2<dis_1$，所以选取 B_2。但是可以看出，只取 B_2 不能满足服务 A 的需求，A 还要索取 B_1 的资源。这样，此种分配方案影响到了 2 个非关键服务。

但是加入 $x_j\geqslant y_{ij}$ 这个限制条件后，重新计算：

$dis_1=(2-2)^2+(2-2)^2=0$；

$dis_2=(2-1)^2+(1-0)^2+(2-1)^2=3$；

$dis_1<dis_2$，所以选取 B_1。可以看出 B_1 直接满足 A 的需求。这样，只需要影响 1 个非关键资源就可以了。

通过加入 $x_j\geqslant y_{ij}$ 这一限制条件，使得近似度算法能够启发式地保证受影响的非关键服务尽可能地少。

3.1.2 问题求解——资源分配方案的生成

使用近似度方法解决资源分配问题的过程如下：

1) 根据公式(3)，计算非关键服务 B_i 与关键服务 A 之间拥有资源的近似度，即 x 与 y_i 的近似度 dis_i；

2) 比较近似度，选出近似度最小的一个，设为 dis_p；

3) 将 B_p 所占有的资源剥夺，分配给 A 使用，设 $d_k=p$，同时从非关键服务中删除 B_p；

4) 计算 A 得到 B_p 的资源后还需要的资源，设为向量 x'；

5) 如果 $x'=(0,\cdots,0)$，则完成返回；否则 $x=x'$，返回 1) 继续下一轮计算，寻找与 x 最为相似的服务，直到 x 为 0。

3.1.3 实例分析

例 1 若某系统描述如下：

1)拥有资源 3 种，即 $N=3$；

2)现有关键服务 A 需要的资源为 $x=(10,10,10)$；

3)现有 4 个非关键服务 B_1,B_2,B_3,B_4，其中 B_1 拥有的资源为 $y_1=(11,9,8)$，B_2 拥有的资源为 $y_2=(9,8,7)$，B_3 拥有的资源为 $y_3=(8,7,6)$，B_4 拥有的资源为 $y_4=(1,2,3)$；

4)其中 4 个非关键服务的响应时间向量为 $t=(5,4,3,2)$。

对例 1 使用近似度算法求解。

首先判断该系统是否满足资源分配的前提条件(见问题描述中第 5 条)。可以看出，非关键服务的每一项资源总量大于关键服务所需要的该项资源量，满足前提条件。

第 1 轮计算近似度，$x=(10,10,10)$：

$dis_1=0+(10-9)^2+(10-8)^2=5$；

$dis_2=(10-9)^2+(10-8)^2+(10-7)^2=14$；

$dis_3=(10-8)^2+(10-7)^2+(10-6)^2=29$；

$dis_4=(10-1)^2+(10-2)^2+(10-3)^2=194$；

dis_1 最小，将 B_1 的资源分配给 A，此时 A 还需要的资源为(0,1,2)；同时将非关键服务编号 1 记录下来，从非关键服务中删除 B_1，即不再参与下一轮计算。

第 2 轮计算近似度，此时 A 需要的资源向量为 $x=(0,1,2)$，计算如下：

$dis_2=0$；

$dis_3=0$；

$dis_4=0$；

3 个近似度相同，按服务编号从小到大选取，所以将 B_2 的资源分配给 A，此时 A 还需要的资源为(0,0,0)，完成；将非关键服务编号 2 记录，返回。

所以分配方案为 $\Phi=\{1,2\}$，即剥夺非关键服务 B_1 和 B_2 的资源分配给关键服务 A。

分析此方案：

1)方案中影响到 2 个非关键服务 B_1 和 B_2，即 $S(\Phi)=2$；

2)B_1 和 B_2 的响应时间分别为 5、4，则 $T(\Phi)=\max(5,4)=5$；

3)此方案的评价参数为：

$Acc=0.4*S(\Phi)+0.6*T(\Phi)=0.4*2+0.6*5=3.8$

3.2 时间优先算法

在可存活系统中，强调一个“及时”，所以非关键服务的响应时间显得尤为重要。响应时间越短，系统存活的几率也就越大。如若响应太慢，关键服务可能彻底崩溃，无法挽救。

本文采用时间优先算法，使得非关键服务的响应时间尽可能地短。

3.2.1 时间优先算法实现

时间优先算法的实现过程如下：

1) 将非关键服务按照响应时间从小到大排序，排列后得到序列 $TS=\{d_1,d_2,\cdots,d_m\}$，其中 d_i 表示非关键服务编号，由此可知 $t_{d1}<t_{d2}<\cdots<t_{dm}$；

2) 按照非关键服务序列 TS，依次剥夺非关键服务 B_{d_1}，$B_{d_2},\cdots,B_{d_k}$ 的资源，直到满足关键服务所需要的资源为止。

3.2.2 实例分析

就例 1 而言，用时间优先算法获得分配方案。求解过程如下：

1)对 4 个非关键服务按照响应时间排序。

因为 $t_4<t_3<t_2<t_1$，可得序列非关键服务编号序列：$TS=\{4,3,2,1\}$。

2)按照序列 TS 选取非关键服务，依次剥夺其资源分配给关键服务。

先取编号为 4 的非关键服务，即剥夺 B_4 的资源(1,2,3)给 A，此时 A 还需资源为(9,8,7)；

然后取 B_3 的资源(8,7,6)，此时 A 还需资源为(1,1,1)；

再取 B_2 的资源(9,8,7)，此时 A 还需资源为(0,0,0)，完成。

分配方案为 $\Phi=\{4,3,2\}$，即剥夺非关键服务 B_4,B_3,B_2 的资源分配给关键服务 A。

分析此分配方案：

1)方案中影响到 3 个非关键服务 B_4,B_3,B_2，即 $S(\Phi)=3$；

2)B_4,B_3,B_2 的响应时间分别为 2、3、4，则 $T(\Phi)=\max(2,3,4)=4$；

3)此方案的评价参数为：

$Acc=0.4*S(\Phi)+0.6*T(\Phi)=0.4*3+0.6*4=3.6$

3.3 分配方案的确定

对于例 1 的问题，分别采用了不同的策略：近似度算法和时间优先算法，得出的结果不同。孰优孰劣，因评价标准不同，最终选取的方案亦不相同。

本文中采用了定义 2 中的评价参数来决定孰优孰劣，从而决定分配方案的选取。分配方案的评价参数愈小，方案愈

优，即综合考虑影响到的非关键服务数较小、响应时间较短。

例1的两个分配方案，其评价参数分别为3.8和3.6，所以选取评价参数较小的3.6作为最终分配方案，即采用时间优先算法得到的方案 $\Phi=\{4,3,2\}$，剥夺非关键服务 B_4，B_3，B_2 的资源分配给关键服务 A。

3.4 算法改进——融合近似度和响应时间

在算法实现过程中，可以将近似度和时间优先融合在一起考虑，即在实现近似度算法时，加入响应时间的考虑；在实现时间优先算法时，加入近似度的度量。

3.4.1 近似度算法中考虑响应时间

首先举个例子来说明此方法。

例2 某系统拥有3种资源 X_1，X_2，X_3；关键服务 A 需要资源数量为(10,8,6)；非关键服务有4个，每个服务拥有的资源相同，均为(5,4,3)；4个服务的响应时间向量为(4,3,2,1)。

按照3.1节中描述的近似度算法，第1轮计算4个非关键服务的近似度相同，默认地按照编号从小到大排列选取，编号1排在前，所以选择1；第2轮计算剩下3个服务的近似度仍然相同，但是由于编号2排在前，所以选择2。所以得到方案 $\Phi_1=\{1,2\}$，即剥夺非关键服务 B_1，B_2 的资源分配给关键服务 A。

分析方案1：$S(\Phi_1)=2$；$T(\Phi_1)=\max(4,3)=4$，$Acc_1=0.4*2+0.6*4=3.2$。

但是，如果在近似度算法中加上对响应时间的考虑，即若几个非关键服务的近似度相同时，比较其响应时间，取响应时间短的服务。利用此方法，可得方案 $\Phi_2=\{4,3\}$，即剥夺非关键服务 B_4，B_3 的资源分配给关键服务 A。

分析方案2：$S(\Phi_2)=2$；$T(\Phi_2)=\max(1,2)=2$，$Acc_2=0.4*2+0.6*2=2$。

Φ_2 优于 Φ_1，所以加入时间因素考虑的近似度算法更具实用性。

3.4.2 时间优先算法中考虑近似度

例3 某系统拥有3种资源 X_1，X_2，X_3；关键服务 A 需要资源数量为(10,8,6)；非关键服务有3个，每个服务拥有的资源分别为(1,2,3)、(4,5,6)、(7,8,9)；3个服务的响应时间均为4。

依照3.2节中的时间优先算法，可知3个服务的响应时间相同，即得到非关键服务编号序列{1,2,3}。于是得到方案 $\Phi_1=\{1,2,3\}$，即剥夺非关键服务 B_1，B_2，B_3 的资源分配给关键服务 A。

分析方案1：$S(\Phi_1)=3$；$T(\Phi_1)=4$，$Acc_1=0.4*3+0.6*4=3.6$。

但是如果在时间优先算法中加上对近似度的考虑，即若几个非关键服务的响应时间相同时，比较其与关键服务资源的近似度，取近似度小者排在序列前端。

利用此方法，可得非关键服务编号序列为{3,2,1}。基于此可以得到分配方案 $\Phi_2=\{3,2\}$，即剥夺非关键服务 B_3，B_2 的资源分配给关键服务 A。

分析方案2：$S(\Phi_2)=2$；$T(\Phi_2)=4$，$Acc_2=0.4*2+0.6*4=3.2$。

Φ_2 优于 Φ_1，所以响应时间相同时考虑近似度的时间优先算法更优化。

3.5 时间优先算法的优化

3.5.1 时间优化算法中的问题

例1中采用时间优先算法生成的方案 $\Phi=\{4,3,2\}$，剥夺非关键服务 B_4，B_3，B_2 的资源分配给关键服务 A。但是，这并不是最优方案。

A 所需资源向量为 $x=(10,10,10)$，而 B_4，B_3，B_2 的资源分别为 $y_4=(1,2,3)$，$y_3=(8,7,6)$，$y_2=(9,8,7)$，很明显，只需要 B_3，B_2 或者 B_4，B_2 就完全可以满足 A 的需求了。

产生这个问题的原因就在于最后一个非关键服务的选择，可能会使得先前选取的某些服务没有必要。极端情况就是如果最后一个选取的服务完全能够满足关键服务的需求，那么之前选取的所有非关键服务都是多余的。

近似度算法产生的是影响非关键服务尽可能少的方案，所以不会出现多余服务的现象；而在时间优先算法中，按照响应时间的大小来选择服务，可能会有多余的服务。为了避免产生这样的情况，可以对时间优先算法得到的方案进行进一步的优化。

3.5.2 优化过程

设时间优化算法产生方案 $\Phi=\{d_1,d_2,\cdots,d_k\}$，其中 d_i 表示被剥夺资源的非关键服务编号，即 B_{d_1}，B_{d_2}，$\cdots$，B_{d_k} 被剥夺资源。关键服务 A 所需资源为向量 x，B_{d_i} 拥有的资源为 y_{d_i}。

从时间优先算法的策略可以推出，最后一个选取的，即第 d_k 个非关键服务是不可从方案 Φ 中去除的。所以优化后得到的方案 Φ' 响应时间和 Φ 方案的响应时间相同。这就要求尽可能多地删除多余的服务，使得方案中保留的服务较少，即被剥夺资源的非关键服务较少。

优化过程如下：

1)计算方案 Φ 中的非关键服务拥有的资源总量，设为向量 y，则 $y=\sum_{i=1}^{k}y_{d_i}$。

2)计算多余的资源向量，设为 z，则 $z=y-x$。

定义5 n 维向量 $p=(p_1,p_2,\cdots,p_n)$ 和向量 $q=(q_1,q_2,\cdots,q_n)$，若 $\forall i\in[1,n]$，$p_i\leqslant q_i$，则 $p\leqslant q$。

3)在 Φ 中寻找资源向量小于 z(即 $y_{d_i}\leqslant z$)的非关键服务 B_{d_i}，记录其编号 d_i，得到服务序列 S。如果 S 为空，结束；否则进入4)。

4)通过公式(3)计算序列 S 中的每一个服务的资源向量与多余资源向量 z 之间的近似度 dis，找出近似度最大(即最不近似)的一个非关键服务，如为 B_p，则从方案 Φ 中删除 p。

这里使用的是近似度的逆算法，因为要求刚好相反，3.1节的近似度算法要求影响到的服务越少越好，即尽可能地少选取一些服务；而优化算法要求删除的服务越多越好，即尽可能地多选取一些服务删除。

如若 $z=(5,5,5)$，方案 Φ 中有3个可供删除，分别为 B_1，B_2，B_3，对应资源向量为 $y_1=(0,1,2)$，$y_2=(2,3,3)$，$y_3=(5,5,5)$，策略一先删除 B_1，然后删除 B_2，策略二选择删除 B_3，当然是策略一较好。通过近似度的逆算法，即4)中的计算，选取近似度较大(最不近似)的服务先删除，可以达到这一目标。

5)$z=z-y_p$。如果 $z=0$，结束；否则返回3)继续下一轮运算。

3.5.3 实例优化

例1中，对时间优先算法得到的分配方案 Φ 进行优化：

1)$y=(18,17,16)$

2)$z=y-x=(8,7,6)$

3)从中可得：$y_4 \leqslant z, y_3 \leqslant z$，即 $S=\{B_4, B_3\}$

4)计算：

$dis_4=(8-1)^2+(7-2)^2+(6-3)^2=83$；

$dis_3=(8-8)^2+(7-7)^2+(6-6)^2=0$。

$dis_4>dis_3$，所以从方案 Φ 中删除 4。

5)$z = z - y_4=(8,7,6)-(1,2,3)=(7,5,3)$

$z\neq 0$，所以返回 3)，继续下一轮。

但是没有找到比 z 小的资源向量，结束。

优化后得到的方案为 $\Phi=\{3,2\}$，即剥夺非关键服务 B_3，B_2 的资源分配给关键服务 A。

分析此分配方案：

1)方案中影响到 2 个非关键服务 B_3, B_2，即 $S(\Phi)=2$；

2)B_3, B_2 的响应时间分别为 3、4，则 $T(\Phi)=\max(3,4)=4$；

3)此方案的评价参数为：

$Acc=0.4 * S(\Phi)+0.6 * T(\Phi)=0.4 * 2+0.6 * 4=3.2$

3.6 算法分析

近似度算法可以找到一个分配方案，使得影响到的非关键服务尽可能地少；时间优先算法可以找到一个方案，使得非关键服务的响应时间尽可能地短。两种算法各在一个方向上，能够通过启发式的方法，找到一个相对最优解。

通过大量实验，分析实验结果可以发现以下规律：

1)若所有非关键服务拥有的资源种类及数量大致相同，即资源分布相对均匀的情形下，使用时间优先算法，可以得到较小的评价参数，从而得到较好性能的分配方案；

2)若非关键服务间拥有的资源种类及数量相差较大，即资源分布不均，资源相对集中于某些服务时，采用近似度算法可以得到较好的分配方案。

Wang Jian 等人在文献[5]中采用了 ERAS 的算法来实现资源分配，并举了一个例子，即例 4。本文引用其例子，用本文的方法来寻找分配方案。

例 4　系统拥有 3 种资源 X_1, X_2, X_3；关键服务 A 需要资源数量为(20,18,17)；非关键服务有 10 个，每个服务拥有的资源如表 1 所示。非关键服务响应时间向量为(2,3,3,4,5,8,11,11,15,20)。

表 1　非关键服务资源表

非关键服务	X_1	X_2	X_3
B_1	7	8	5
B_2	1	3	6
B_3	5	2	2
B_4	9	5	2
B_5	4	4	3
B_6	2	7	9
B_7	4	10	2
B_8	5	3	3
B_9	12	4	4
B_{10}	3	5	10

使用近似度算法得到分配方案 $\Phi_1=\{1,9,6\}$，即剥夺非关键服务 B_1, B_9, B_6 的资源分配给关键服务 A。

分析方案 1：$S(\Phi_1)=3$；$T(\Phi_1)=15$，$Acc_1=0.4 * 3+0.6 * 15=10.2$。

使用时间优先算法得到方案 $\Phi_2=\{1,3,2,4,5\}$，即剥夺非关键服务 B_1, B_3, B_2, B_4, B_5 的资源分配给关键服务 A。

分析方案 2：$S(\Phi_2)=5$；$T(\Phi_2)=5$，$Acc_1=0.4 * 5+0.6 * 5=5$。

通过分析可得，方案 2 优于方案 1，所以最终采用 $\Phi_2=\{1,3,2,4,5\}$。

在文献[5]中的 ERAS 算法通过迭代寻找分配方案。对于例 4，ERAS 算法得到的最终分配方案与 Φ_2 相同，但是却需要 5 步迭代运算，时间与空间消耗较大，效率较低。

本文采用启发式的近似度算法和时间优先算法，算法简单，容易实现，而且能够获得较优的分配方案。

4 进一步工作

本文使用了近似度算法和时间优先算法，获得了两套资源分配方案，两套方案各自满足两个限制条件中的一个(即影响非关键服务的个数较少或响应时间较短)。然后通过方案评价参数的分析，选取综合性能较好的一个作为最终方案。

虽然 3.4 节在问题求解的过程中，将近似度和响应时间结合起来考虑，但是只是在近似度或者响应时间相同时才引入另外一个因素的考虑。如何将近似度和响应时间在方案求解过程中更加全面、深入地结合，即通过适当的运算和转化后，能够将两者因素综合考虑，使得最后得到的一个方案最优，是本课题下一步的研究工作。

此外，本文研究了只有一个关键服务的资源遭到破坏时需要其他服务支援的情况；如果多个关键服务的资源同时受到了破坏，其他非关键服务如何支援它们，即如何恰当地分配非关键服务的资源给多个关键服务，是本课题下一步的研究重点。

结束语　本文从资源重配置的角度实现系统的可存活性。当某个关键服务的资源受到破坏时，通过剥夺非关键服务拥有的资源分配给它，从而保证关键服务能够持续运行，实现系统的可存活性。

资源分配方案多种多样，但是本文从两个限制条件出发考虑分配方案，即要求方案中被剥夺资源的非关键服务尽可能少，非关键服务对于关键服务的响应时间尽可能短。本文利用近似度算法得到的分配方案影响非关键服务数量较少；采取时间优先算法得到的方案使得非关键服务响应时间较快。

两个算法并不是各自为政，需要结合起来考虑。在近似度算法的实现过程中，同时加入非关键服务对关键服务响应时间的考虑；在时间优先算法的实现过程中，也要考虑非关键服务与关键服务之间占有资源的近似度。

最后比较两种算法产生的两个方案，选取综合性能较好的一个作为最终的资源分配方案。

无论是近似度算法还是时间优先算法，算法都比较简单，实现起来比较容易，时间及空间效率较高，最终产生的分配方案综合性能较优。

参 考 文 献

[1] Kyamakya K. Security and Survivability of Distributed Systems: an Overview. IEEE MILCOM 2000. Los Angels, California, 2000:449-454

[2] Ellison R, Fisher D, Linger R, et al. Survivable Network Systems: an Emerging Discipline [R]. CMU/SEI-97-TR-013, 1997

[3] Lu Tun, Gu Ning. Survivability-aware Configuration Management of Service-oriented System Based on Service Dependency. Theoretical Aspects of Software Engineering, 2007:421-432

[4] Wells D, Ford S, Langworthy D, et al. Software Survivability//

(下转第 276 页)

5 模拟实验验证

为了验证基于 QoS 的组合相似度的服务选择算法的可行性和优越性，本文进行了实例数据分析，并对实验结果进行比较。

作为对比，本文还将参考文献[8,9]中的方法进行对比，它们没有计算 QoS 参数语义相似度的部分，从以下的实验结果中，我们可以看见本文的基于组合相似度的选择算法将选出更符合用户需求的服务。在本文中，我们选取必要服务参数 Q_N 中的费用、响应时间、可靠性、可用性和信誉构成 QoS 参数，其中，费用和响应时间的单位分别是 Dollar 和 ms，可靠性和可用性用百分数来表示，信誉等级用取值范围在[1,5]之间的整数表示。

表 1 定义了 2 个服务请求者的 QoS 请求，因为不同的服务请求者对每个参数的关注程度不同，因此对不同的服务参数分配的权重也不同。

表 1 服务请求者的 QoS 请求

	费用		响应时间		可靠性		可用性		信誉等级	
	期望取值	权重	期望取值	权重	期望取值	权重	期望取值	权重	期望取值	权重
R1	9.30	0.3	9.62	0.1	0.88	0.2	0.75	0.2	4	0.2
R2	9.43	0.1	10.59	0.3	0.93	0.3	0.95	0.2	3	0.1

根据表 1 中定义的 QoS 向量，表 2 中提供了 5 个 Web 服务，它们的 QoS 参数值均为随机生成。

表 2 提供的 Web 服务

	费用	响应时间	可靠性	可用性	信誉等级
S1	9.32	9.23	0.95	0.93	4
S2	10.51	9.51	0.78	0.92	3
S3	9.21	10.32	0.9	0.76	5
S4	9.41	9.20	0.99	0.73	3
S5	8.99	9.89	0.93	0.96	4

首先，在本例中，因为所有的 QoS 参数都是精确匹配，所以没有需要淘汰的服务。接下来我们构造标准化匹配矩阵，通过加权操作，计算每个服务的匹配值。因为费用和响应时间是属于成本型参数，可靠性、可用性和信誉等级属于效益性参数，信誉等级属于离散型参数，根据不同的标准化算法，我们得到如表 3 所示的结果。

表 3 匹配结果

	S1	S2	S3	S4	S5	最匹配的服务
R1	0.82	0.34	0.60	0.72	0.83	S5
R2	0.83	0.44	0.39	0.72	0.71	S1

由例子我们可以看出，根据不同的服务请求，我们可以个性化地为不同的请求者推荐不同的服务。本文与文献[8,9]最大的区别是在进行 QoS 参数语义匹配的基础上再进行数值匹配，这样，如果在精确匹配时不能满足用户的需求时，可以从插入匹配或者包含匹配中选择数值匹配度达到用户要求的 Web 服务返回给用户。

结束语 本文提出了一种基于 QoS 本体的 Web 服务选择方法，我们用一个 QoS 本体以及相应的词汇来描述 QoS 参数，定义了各种质量属性和它们各自的度量方法，基于以上的工作，我们提出了一个支持 QoS 的选择机制，将 QoS 参数的语义匹配度和数值匹配度相结合，利用归一化算法对不同类型的 QoS 参数进行标准化，在此基础上提出了一个公平的、动态的选择机制。本文的方法与文献[8,9]的区别在于是在进行 QoS 参数语义匹配的基础上再进行数值匹配，这样，如果在精确匹配时不能满足用户的需求时，可以从插入匹配或者包含匹配中选择数值匹配度达到用户要求的 Web 服务，语义匹配这个过程的增加为 Web 服务发现的查全率提供了更好的保证，同时，根据本文提出的标准化算法和加权过程，根据不同的请求者的要求，可以为用户提供更合适的 Web 服务。

参 考 文 献

[1] 杨胜文，史美林. 一种支持 QoS 约束的 Web 服务发现模型[J]. 计算机学报，2005，28(4)：589-594

[2] Studer R, Benjiamins V R, Fensel D. Knowledge Engineering: Principle and Methods[J]. Data & Knowledge Engineering, 1998, 25: 161-197

[3] 李春梅，蒋运承. 具有 QoS 约束的语义 Web 服务发现的研究[J]. 计算机科学，2007，34(6)：116-121

[4] Wang X, Vitvar T, Kerrigan M, et al. A QoS-aware Selection Model for Semantic Web Services [J]. Lecture Notes in Computer Science, Nov. 2006: 390-401

[5] Maximilien E M, Singh M P. A Framework and Ontology for Dynamic Web Service Selection[J]. IEEE Internet Computing, 2004(10): 84-93

[6] Kim H M, Sengupta A, Evermann J. MOQ: Web Services Ontologies for QoS and General Quality Evaluations // European Conference on Information Systems (ECIS 2005). Regensburg, Germany, May 2005

[7] Paolucci M, Kawamura T, Payne T R, et al. Semantic Matching of Web Services Capabilities // Proceedings of the 1st International Semantic Web Conference ISWC. Sardinia, 2002: 333-347

[8] 郭得科，任彦，陈洪辉，等. 一种 QoS 有保障的 Web 服务分布式发现模型. 软件学报，2006，17(11)：2324-2334

[9] 蒋运承，史忠植. QoS 驱动的主体服务匹配[J]. 小型微型计算机系统，2005，264：687-692

(上接第 243 页)

DARPA Information Survivability Conference and Exposition. 2000, 2: 241-255

[5] Wang Jian, Wang Huiqiang, Zhao Guosheng. ERAS—an Emergency Response Algorithm for Survivability of Critical Services. Computer and Computational Sciences, 2006, 2: 97-100

[6] Hiltunen M A, Schlichting R D, Ugarte C A, et al. Survivability Through Customization and Adaptability: The Cactus Approach // DARPA Information Survivability Conference and Exposition. 2000, 1: 294-307