

一种基于场景匹配的可信构件选择方法

魏 乐, 赵秋云, 舒红平

(成都信息工程学院 计算机学院, 四川 成都 610225)

摘 要: 为了提高 Internet 环境下构件选择的有效性, 提出一种将场景匹配技术应用于构件选择的方法. 首先, 结合 Internet 环境的特点, 将构件的可信性划分为可靠性、可用性和时效性, 建立起构件可信性评估的一体化模型; 其次, 给出场景的定义和场景匹配度的计算方法; 接着, 将历史评价信息和场景匹配引入构件选择逻辑, 实现了系统环境和可信需求约束下的构件选择机制, 并给出构件选择算法; 最后, 通过仿真实验, 说明了基于场景匹配的构件选择方法的有效性和实用性.

关键词: 构件; 场景匹配; 可信评估; 构件选择

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)08-0176-04

A Trusted Component Choosing Method Based on Scene Matching

WEI Le, ZHAO Qiu-yun, SHU Hong-ping

(Department of Computer, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: In order to improve the effectiveness of component choosing under Internet, a scene matching technology is raised to be applied in component choosing. Firstly combining with characteristics of Internet environment, the component credibility is evaluated from three aspects of reliability, usability and timeliness. An integrated evaluation model is built for the component credibility. Secondly the definition of scene and the calculation method of scene matching degree are proposed. And then by introducing the history evaluation and scene matching information into the component choosing logic, the mechanism of component choosing is realized with constraints of environment and credibility demand. The component choosing algorithm is also designed. Finally through simulation experiments, it's shown that the component choosing method is valid and usable.

Key words: component; scene matching; credibility evaluation; component choosing

1 引言

基于构件的软件开发通过组装现有软件构件来建造新的软件系统, 在很大程度上提高了软件质量和开发效率. 在分布、开放、动态的 Internet 环境下, 网络节点中存放着大量的构件, 如何快速地选择合适的构件, 并保证选择的构件在环境变化的条件下保证其功能需求和非功能需求(即保证构件的可信性), 是现今 CBSD 技术研究的一个重点.

当前, 人们从多个方面对构件的可信性评估和选择方法进行了研究, 并取得了一系列的成果. 包括基于信任机制的构件选择方法^[1]、基于 Web 服务的

信任评估模型^[2]、基于模糊理论的 Web 服务信任评估模型^[3]、基于服务质量分级的构件服务质量保障方法^[4]、基 Agent 的网构软件构件模型^[5]等. 这些工作从不同的角度阐述了构件技术所涉及的问题, 但在分布式的构件库中, 如何快速、准确地选择满足功能需求和非功能需求的构件依然面临一些挑战: (1)可信是一个广泛的概念^[6], 它包括可用性、可靠性、时效性、安全性等, 如何根据需要选择可信特征集, 并建立对构件可信性的一体化评价方法, 是首要考虑的问题; (2)环境的变化会直接影响构件的可信性, 一个构件工作的历史环境信息对该构件的选择有重要的借鉴意义. 因此对构件的历史信息和当前环境进行描述, 并利用两者的联系, 做出合理的构件

选择决策,是研究的一个重点.

基于以上分析,本文将构件的可信性划分为可靠性、可用性和时效性,给出可信性的计算方法;在对场景进行了抽象的基础上,给出场景匹配度的计算方法,建立起历史信息与构件可信性之间的关联模型;通过在系统中引入场景匹配逻辑,给出基于场景匹配的可信构件选择算法和仿真实验,说明了文中方法的有效性和实用性,并总结全文.

2 相关定义

2.1 构件的可信性

构件的可信性是指构件使用者对构件所提供服务的相信程度,它是一个广泛的概念,是对若干传统技术的综合.在评估可信性时,结合构件和 Internet 环境的特点,考虑到实用性和易测量性,我们这里选择可用性、可靠性和时效性作为构件可信性的评价指标,即存在

$$CDW = f(\text{Reliability}, \text{Availability}, \text{Timeliness}) \quad (1)$$

(1) Reliability 指可靠性,它表示构件在一个完整的时间间隔内正常提供服务的能力.假设在 $(0, T)$ 的时间内使用构件 Component 服务的次数为 N ,该构件正常响应请求的次数为 N_r ,则可靠性量化为

$$\text{Reliability}(\text{Component}) = N_r / N \quad (2)$$

(2) Availability 指可用性,它表示构件在某个时刻可用的概率.设构件 Component 可提供服务的时间长度为 T_u ,总的测量时间为 T ,则可用性量化为

$$\text{Availability}(\text{Component}) = T_u / T \quad (3)$$

(3) Timeliness 指时效性,它表示构件执行请求的及时性.设构件 Component 的执行时间为 t_e ,允许的最大响应时间为 t_a ,则时效性量化为

$$\text{Timeliness}(\text{Component}) = \begin{cases} 1/t_e & t_e \leq t_a \\ 0 & t_e > t_a \end{cases} \quad (4)$$

(4) CDW 表示构件的综合可信度,可根据 f 映射函数计算. f 定义为 $f = \sum_{i=1}^3 \text{Weight}_i * D_i$,其中, D_i 表示可信特征,即 $D_i \in \{\text{Reliability}, \text{Timeliness}, \text{Availability}\}$; Weight_i 表示可信特征 D_i 在构件综合可信度中所占的权重,且 $\sum_{i=1}^3 \text{Weight}_i = 1$. 实际使用中根据应用关注点,为可信特征设置合适的权重.

2.2 场景

构件工作于一定的环境下,环境的变化会直接影响到构件的可信性.因此,为了有效地对构件的可

信性进行评价,我们引入了环境的定义.

定义 1 环境是一系列变量的集合,可定义为一个四元组,即 $\text{Environment} = (\text{CPU}, \text{Memory}, \text{Bandwidth}, \text{I/O})$.

为了对环境进行准确表述,环境变量主要考虑 CPU 处理能力、剩余内存大小、网络带宽和 I/O 能力;任一时刻环境变量都有一个 Value 属性,指明该变量的值.有了环境的定义,可给出场景的定义.

定义 2 构件在 t 时刻所处的环境,称为构件的一个场景 $\text{Scene}(t) \xrightarrow{\text{define}} (\text{Environment}, t)$.

2.3 场景匹配度

构件在定义时,不但要定义构件与外部进行交互的接口,还需要指明保证其正常运行所需的资源约束.用 $\text{Constraints}(\text{Component})$ 表示构件对环境的约束,其本质上是变量 CPU、Memory、Bandwidth 和 I/O 及其约束的集合.

定义 3 构件的资源约束是一个四元组, $\text{Constraints}(\text{Component}) = (\text{CPU}, \text{Memory}, \text{Bandwidth}, \text{I/O})$. 这里每个变量具有 Threshold 和 Weight 两个属性,其中 Threshold 表示对应资源的下限阈值;Weight 表示对应资源在构件正常运行中的重要程度, $\text{Weight} \in [0, 1]$, 0 表示对该类资源无要求, 1 表示仅对该类资源有要求,且要求必须得到满足.

定义 4 设 $\text{Var}_i \in \{\text{CPU}, \text{Memory}, \text{Bandwidth}, \text{I/O}\}$; 存在时刻 t_1, t_2 的场景 $\text{Scene}(t_1)$ 和 $\text{Scene}(t_2)$, 如果满足条件 $\text{Scene}(t_1). \text{Environment}. \text{Var}_i. \text{Value} = \text{Scene}(t_2). \text{Environment}. \text{Var}_i. \text{Value}$, 则称 $\text{Scene}(t_1)$ 和 $\text{Scene}(t_2)$ 完全匹配,记为 $\text{Scene}(t_1) = \text{Scene}(t_2)$.

定义 5 设 $\text{Var}_i \in \{\text{CPU}, \text{Memory}, \text{Bandwidth}, \text{I/O}\}$; 存在 t_1 时刻的场景 $\text{Scene}(t_1)$ 和构件 Component 对环境的约束 $\text{Constraints}(\text{Component})$, 如果满足条件 $\text{Scene}(t_1). \text{Environment}. \text{Var}_i. \text{Value} \geq \text{Constraints}(\text{Component}). \text{Var}_i. \text{Threshold}$, 则称场景 $\text{Scene}(t_1)$ 满足构件 Component 对环境的约束,记为 $\text{Scene}(t_1) = \text{Constraints}(\text{Component})$.

为了表示场景 $\text{Scene}(t_1)$ 和场景 $\text{Scene}(t_2)$ 的匹配程度,引入了场景匹配度的定义.

定义 6 设存在 t_1 时刻的场景 $\text{Scene}(t_1)$, t_2 时刻的场景 $\text{Scene}(t_2)$ 和构件 Component 对环境的约束 $\text{Constraints}(\text{Component})$; 条件 $\text{Scene}(t_1) = \text{Scene}(t_2) \vee (\text{Scene}(t_1) = \text{Constraints}(\text{Component}))$

$\wedge \text{Scene}(t_2) = \text{Constraints}(\text{Component})$) 记为 C , 则

$$\text{MatchD}(\text{Scene}(t_1), \text{Scene}(t_2)) = \begin{cases} 1, & \text{满足 } C \\ \sum_{i=1}^4 w_i * (\text{Scene}(t_1), \text{Environment}, \text{Var}_i, & (5) \\ \text{Value-Scene}(t_2), \text{Var}_i, \text{Threshold}), & \text{其他} \end{cases}$$

式中, $\text{Var}_i \in \{\text{CPU}, \text{Memory}, \text{Bandwidth}, \text{I/O}\}$; w_i 表示资源 Var_i 在构件正常运行中的重要程度, 且 $\sum_{i=1}^4 w_i = 1$; $\text{Constraints}(\text{Component}), \text{Var}_i, \text{Threshold}$ 代表以百分比的形式表示构件运行主机的剩余能力; $\text{Scene}(t_1), \text{Environment}, \text{Var}_i, \text{Value}$ 代表构件 Component 对环境的约束, 取值为阈值所占构件运行主机资源的百分比。

显然, $\text{MatchD}(\text{Scene}(t_1), \text{Scene}(t_2)) \in (-1, 1]$, 值越大, 表示匹配度越高; 除匹配度等于 1 的情况, $\text{MatchD}(\text{Scene}(t_2), \text{Scene}(t_1)) \neq \text{MatchD}(\text{Scene}(t_1), \text{Scene}(t_2))$ 。

3 可信构件选择

3.1 可信构件选择逻辑

从前面的叙述可以看出, 依赖将来运行构件实例的环境和构件库中存放的历史场景信息, 并通过两者的匹配以及构件可信度的比较, 就可以在构件库中选择一个可信度满足要求的构件。本文使用可信构件选择逻辑 CSL 来完成构件的选择。

定义 7 CSL 定义为一个四元组, 即 $\text{CSL} = (\text{Detector}, \text{Calculator}, \text{Selector}, \text{Comparator})$ 。

Detector 表示侦测器, 主要获取环境信息并传送给 Comparator ; Selector 表示查询器, 能够向各节点的构件库发送查询请求, 并接收所返回的构件信息, 将其传送给 Calculator ; Calculator 表示计算器, 能够对从 Selector 接收的信息进行计算, 比如以百分比的形式描述构件对运行环境的约束信息, 并将计算的结果送 Comparator ; Comparator 表示场景匹配器, 该部件接收来自 Detector 和 Calculator 的信息, 并按照接收的信息进行匹配度的计算, 然后结合构件的可信度, 完成构件的选择。CSL 的逻辑结构如图 1 所示。

3.2 可信构件选择算法

为了实现基于场景匹配的可信构件选择, 需要在构件库中存放构件实例运行的相关历史信息, 如构件历史场景、可信度等, 为此引入构件历史信息列表的定义。

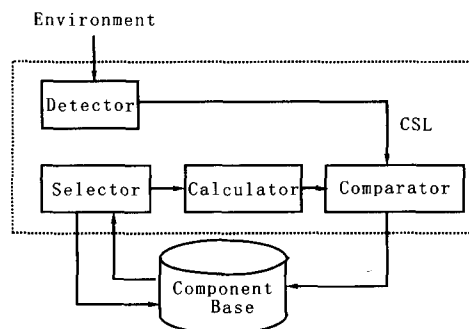


图 1 CSL 的逻辑结构

定义 8 $\text{List} = \{\text{List}_1, \text{List}_2, \dots, \text{List}_n\}$, 每一列表项 $\text{List}_i = \{\text{ID}, \text{Addr}, \text{Scene}(t), \text{CDW}, \text{Honesty}\}$, 其中 ID 是构件唯一的标识; Addr 表示构件实例运行的主机名称, 该名称具有唯一性; $\text{Scene}(t)$ 表示时刻 t 构件实例所在的场景; CDW 表示构件在 t 时刻的可信值, 该值由 Addr 计算反馈; Honesty 表示 Addr 的诚信度, 诚信度的计算见文献[7]。

可信构件选择算法描述如下。

步骤 1: 置可选构件集 $\text{Option} = \Phi$, 向构件库发送查询请求, 查找满足功能需求的构件加入 Option , 即 $\text{Select}(\text{Component}) \xrightarrow{C_i} \text{Option}$; 若 $\text{Option} = \Phi$, 则转步骤 10。

步骤 2: 对 Option 中的所有构件 C_i , 如果 $\text{Scene}(t_1) < \text{Constraints}(C_i)$, 则将 C_i 从 Option 中删除; 若 $\text{Option} = \Phi$, 则转步骤 10。

步骤 3: 置没有历史信息的构件集 $\text{New} = \Phi$; 对 Option 中的所有构件 C_i , 如果 $C_i, \text{List} = \text{null}$, 则将 C_i 从 Option 中删除, 并将 C_i 加入 New , 即 $C_i \rightarrow \text{New}$; 若 $\text{Option} = \Phi$, 则转步骤 7。

步骤 4: 对 Option 中的每个构件 C_j , 使用公式 $\text{CDW}(\overline{C_j}) = (\sum_{i=1}^n C_j, \text{list}_i, \text{CDW} * C_j, \text{list}_i, \text{Honesty}) / n$ 计算 C_j 的整体可信度, 若 $\text{CDW}(\overline{C_j}) < \xi$, 则将 C_j 从 Option 中删除, ξ 表示可信阈值; 若 $\text{Option} = \Phi$, 转步骤 8。

步骤 5: 对 Option 中的每个构件 C_j , 使用公式 $\text{MatchD}(\text{Scene}(t_1), \overline{C_j}) = (\sum_{i=1}^n \text{MatchD}(\text{Scene}(t_1), C_j, \text{list}_i, \text{Scene}(t))) / n$ 计算 C_j 中历史场景 $C_j, \text{list}_i, \text{Scene}(t)$ 与 $\text{Scene}(t_1)$ 的平均匹配度, 并保存运算结果。

步骤 6: 在 Option 中选取 $\text{MatchD}(\text{Scene}(t_1), C_i)$ 最大的构件 C_i , 并返回 C_i, ID, C_i 就是选择的结果, 转步骤 9。

步骤 7: 对 New 中的每个构件 C_i , 使用式(5)计算 $\text{MatchD}(\text{Scene}(t_1), C_i, \text{Constraints})$, 保存结果。

步骤 8: 在 New 中选取 $\text{MatchD}(\text{Scene}(t_1), C_i, \text{Constraints})$ 最大的构件 C_i , 并返回 C_i . ID, C_i 就是选择的结果.

步骤 9: 选择的构件 C_i 运行一段时间, 运行节点利用式(1)计算其在 t 时刻的可信度 CDW, 并将构件 ID、运行节点标识 Addr、 t 时刻的场景 Scene(t)、可信度 CDW、节点的诚信度 Honesty 等信息填入构件库.

步骤 10: 找不到需要的构件, 返回 null.

3.3 可信构件选择算法

为了评价基于场景匹配的构件选择方法对应用可信性的影响, 我们设计了仿真实验. 实验模拟了 60 个分布式环境下的构件库节点 ComponentBase1 ~ ComponentBase60, 在每个构件库中, 存储了 50 个构件. 除 ComponentBase1 和 ComponentBase15 外, 每个构件库中均存放了一个具有数据展示功能的构件 ComponentView, 以及该构件的 List 信息, 且 $\text{List} = \{\text{List}_1, \text{List}_2, \dots, \text{List}_{10}\}$; 在 ComponentBase1 和 ComponentBase15 上没有存放 ComponentView 的历史场景信息, 这表明这两个构件库中的 ComponentView 没有被使用过. 当然, 在不同的构件库中, 该构件所完成的具体功能有所不同, 大致可分为三类: (1) 简单的数据显示, 不显示颜色、图片; (2) 不但显示数据, 还对显示的数据进行格式化美化操作, 如显示颜色、图片, 控件外形美化等; (3) 对显示的数据进行美化操作, 同时可以自适应的调整界面大小, 以适应屏幕分辨率.

实验中分别使用基于场景匹配的选择方法和基于随机选择的方法选择 ComponentView, 并各构造 25 次应用. 对每次应用, 通过式(1)计算构件 ComponentView 的可信度来评价选择方法的效果. 在计算场景匹配度时, CPU、Memory、Bandwidth 和 I/O 的权重分别为 0.5、0.3、0 和 0.2; 在计算综合可信度时, 可靠性、时效性和可用性的系数分别为 0.5、0.2 和 0.3. 仿真实验的结果如图 2 所示.

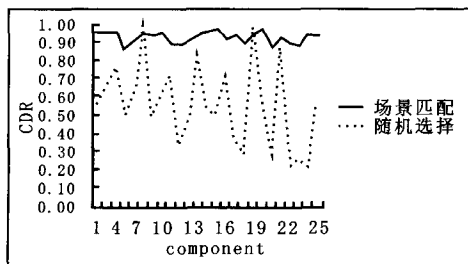


图2 仿真实验结果

从仿真实验结果可以看出, 采用基于场景匹配

的构件选择时, 虽然会出现选择的构件可信度并非最高的情况, 但其可信度取值比较平滑, 且基本保持在一个较高的水平; 采用基于随机选择的方法, 可信度的变动范围较大, 且大多数可信度较低.

4 结束语

本文从提高构件化软件服务质量的角度出发, 利用可信评价技术和场景匹配技术, 探讨了从分布的构件库中选择构件的方法. 对构件的可信度从可靠性、可用性和时效性 3 个角度衡量, 建立起构件可信度的综合评价体系. 在系统中内置 CSL, 利用环境信息和场景信息、环境信息和构件环境约束信息的匹配程度, 并依据历史评价, 来完成构件的选择. 仿真实验表明, 基于场景匹配的构件选择方法因为考虑到了历史信息和环境约束信息, 从而可以为应用提供更好的服务质量. 下一步我们的任务是将自主学习机制引入 CSL, 以进一步加强构件选择的自主性和适应性, 并建立一系列的基础开发和性能评价工具, 为实施应用做好准备.

参考文献:

- [1] 张晓梅, 张为群. 一种基于信任机制的网构软件的构件选择方法研究[J]. 计算机科学, 2010, 37(2): 161-164.
- [2] 刘彬, 张仁津. 一种基于 Web 服务的信任评估模型[J]. 微电子学与计算机, 2010, 27(8): 168-172.
- [3] 李季, 朱小勇. 基于模糊理论的 Web 服务信任评估模型[J]. 计算机工程, 2010, 36(15): 25-28.
- [4] Le Wei, Yaling Li, Tianyun Yan, et al. Model and credibility support of pervasive computing based on classified QoS[C]// The 3rd International Conference on Networks Security, Wireless Communications and Trusted Computing. Wuhan: IEEE Press, 2011: 617-621.
- [5] 常志明, 毛新军, 齐治昌. 基 Agent 的网构软件构件模型及其实现[J]. 软件学报, 2008, 19(5): 1113-1124.
- [6] 周明辉, 梅宏. 可信计算研究的初步探疑[J]. 计算机科学, 2004, 31(7): 5-8.
- [7] 杜瑞忠, 杨晓晖, 田俊峰. 利用多服务属性进行信任评估模型研究[J]. 武汉大学学报: 信息科学版, 2010, 35(5): 524-527.

作者简介:

魏乐 男, (1979-), 硕士, 讲师. 研究方向为构件技术、可信计算.

赵秋云 女, (1979-), 硕士, 讲师. 研究方向为网络计算、服务计算.

舒红平 男, (1974-), 博士, 教授. 研究方向为软件工程、数据挖掘.