Web Services组合的容错方法

唐 渊

（湖南工业大学 电气与信息工程学院，湖南 株洲 412008）

摘 要：Web服务组合的可信性是Web服务研究中的一个重要方面，容错处理则是提高服务可信性的核心技术。针对Web服务的特点提出了Web服务组合的容错基本框架，并在此框架下提出了Web服务容错方法。最后通过实验验证了方法的可行性。

关键词：Web服务；Web服务组合；容错；可信

中图分类号：TP393 **文献标志码**：A **文章编号**：1673-9833(2010)06-0053-05

A Fault-Tolerant Method of Web Services Composition

Tang Yuan

（School of Electrical and Information Engineering，Hunan University of Technology，Zhuzhou Hunan 412008，China）

Abstract：Credibility of Web services composition is an important aspect in the Web services research. Fault tolerance is the core technology to improve service reliability. Aiming at features of Web services, proposes the basic framework for fault tolerance of Web service composition, and presents its fault-tolerant method. Finally verifies the feasibility of the method through experiments.

Keywords：Web services；Web services composition；fault-tolerant；credible

0 引言

由于Web Services的平台无关性，它为Internet上的资源共享和利用提供了良好的解决方案。伴随着Web服务的广泛应用，Web服务组合在业务流程管理中起到了越来越重要的作用。正因为这样，Web服务组合的可信性成为了人们关注的焦点。容错则是提高Web服务组合可信性的一种重要的手段。

Web Services系统有自包含（self-contained）、自描述（self-describing）、模块化和松耦合等特点[1]，它的这种动态性和自适应的特性使得其可信性存在一些问题。建立容错的Web Services系统是提高Web服务可靠性和可用性的关键。当今的容错模型主要有：FTWSCM（fault tolerant web services computation model）[2]模型、PDMA-FT4WS（policy driven and multi-agent based fault tolerance for web services）[3]、基于QoS的服务容错模型[4]、eFlow[5-6]、DeW（dependable web services）[7]、Transparent fault-tolerant web service[8]、FAWS（fault tolerance for web services）[9]、FT-SOAP[10]、FTWeb（fault tolerant infrastructure for web services）[11]。但是，现有的容错模型没有根据不同类型的失败提出不同的解决方案，以获可用度更高的组合服务。

本文提出的容错方法是在服务组合时同时搜索网上备用服务作为冗余，当原有服务出现错误时，备用服务可以及时地恢复原有服务继续执行。同时根据服务失败的基本类型以及发生失败的服务所处的流程类型来确定采用何种恢复策略[12]，并且通过集成在服务中的Agent来协调容错系统的工作。

1 系统框架

本文借鉴了经典的容错反射模型，并针对Web服

收稿日期：2010-09-04

通信作者：唐 渊（1982-），男，湖南株洲人，湖南工业大学助教，硕士，主要研究方向为分布式计算，
E-mail：ttottotto@sohu.com

务组合的特点，提出了Web服务组合容错基本框架FT4WSC(fault-tolerant for web services composition)，如图1所示。

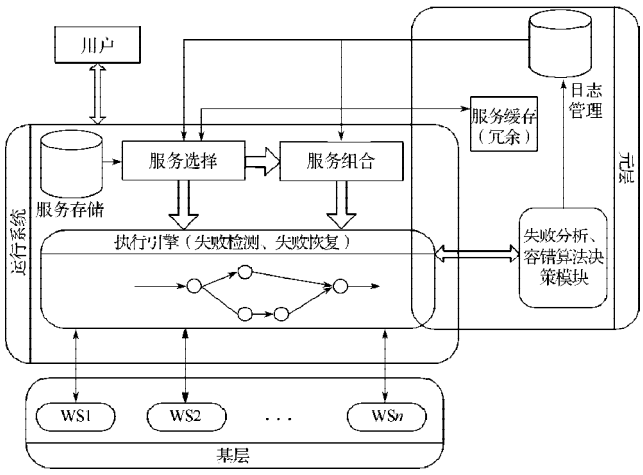

图1 FT4WSC框架图

Fig. 1 Framework of FT4WSC

以下就各个主要模块进行分别说明。

服务存储模块：UDDI作为一种Web服务的分类目录，仍是Web服务发布和发现的首选。

服务选择与组合模块：该模块的主要功能是将符合用户要求的服务从UDDI中间提取出来，并形成组合方案。同时，将冗余的Web服务形成服务缓存以供容错时使用。

执行引擎（失败检测、失败恢复）模块：该模块的主要功能是检测、记录服务运行的状态，并及时的把服务出现的情况传送给分析决策模块。更重要的是执行引擎负责记录变量的状态和变量的值，供容错时使用。

失败分析与容错算法决策模块：该模块的主要功能是接收执行引擎监控到的服务失败信息，通过对失败的特点进行分析，从而判断失败的类型以及失败的层次。然后，该模块根据失败的分类情况触发相应的失败处理（重试调用、替代失败服务、局部重构），发出失败信息到有关的模块，用来指导流程的自恢复。

日志管理模块：该模块负责记录在流程运行的过程中，服务的失败记录。

服务缓存模块（服务冗余）：该模块为容错的关键性模块。模块的主要功能：保存组合流程中执行的实体服务所对应的冗余服务。

2 关键部件实现

2.1 执行引擎

执行引擎的主要功能是检测组合服务执行的状态，记录服务执行的接口以及对全局变量的影响，更重要的是失败检测、失败恢复。执行引擎包含：存储模块、主协调器和协调器。存储模块主要负责存储组合方案以及在组合服务执行过程中全局变量的变化；主协调器和协调器分别部署在服务器端和服务实体端，通过两者之间的协同操作来完成组合服务的运行检测及其容错。执行引擎输入是服务的组合方案，控制检测对象是实体服务。该执行引擎是分布式的体系机构，具体如图2所示。

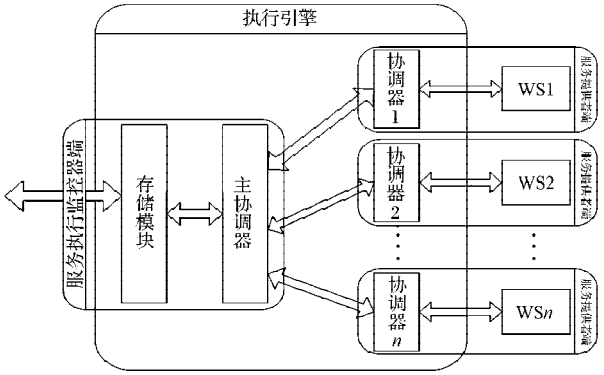

图2 执行引擎原理图

Fig. 2 Schematic of execution engine

2.2 存储模块

存储模块对Web服务组合方案进行存储。为在容错过程中更便捷地向前或向后进行搜索，本文采用了双向链表作为存储的基本结构。基本存储结构如表1所示：

表1 执行引擎存储单元结构

Table 1 Memory unit of execution engine

名称	类型	用途
WS_ID	整型	标志组合方案中的WS
Pre	指针	指向前一个服务、第一个服务为null
Next	指针	指向后一个服务、最后一个服务为null
Current_State	自定义	WS当前的状态
WS_url	字符串	WS的URL
Composition_Type	字符串	WS在方案中的流程
Type_exp	自定义	WS流程类型扩展

下面根据表1分别介绍各个组成部分。

定义1 服务当前的状态Current_State定义如下：Current_State∈{WS_really, WS_running, WS_unknown, WS_stopped,WS_finished}。

WS_really表示服务正在准备中，所有流程中的服务初始化都为此状态；WS_running表示服务正在运行，在协调器发回运行确认信息时更新；WS_unknown表示服务处于不确定状态，在主协调器和协调器之间信息出现交互问题时所处的状态；WS_stopped表示服务处于被协调器强行停止的状态；WS_finished表示服务执行完毕，在协调器发回执行完成信息以及接口数

据之后服务所处的状态。各个状态之间的关系如图 3 所示。

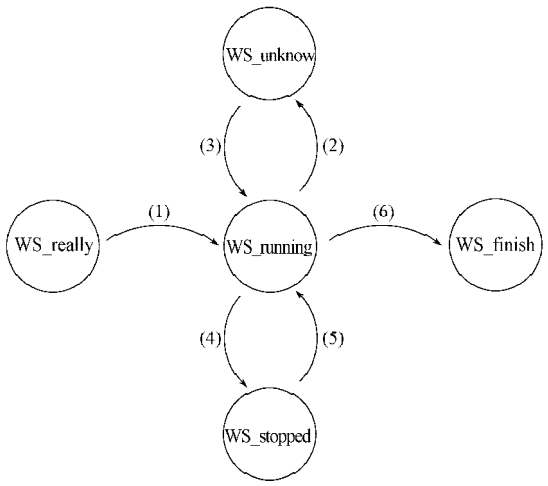

图 3　服务状态变迁图

Fig. 3　Evolution of Web services status.

图 3 中，（1）表示服务从准备状态变成运行状态。此时，协调器已经将其本地服务启动，并向主协调器发送了确认启动信息；（2）表示服务状态由运行变成不确定。此时，主协调器和协调器之间在一定时间内没有联系，无法确定 WS 处于什么样的运行状态；（3）表示服务状态由不确定变成了运行。此时，主协调器和协调器之间的联系恢复，并得到协调器发回的状态信息；（4）表示服务从运行变成了停止。此时，可能由于 WS 执行时间过长或者是发生异常导致主协调器认为 WS 已经出现失败，从而将此 WS 的状态改为停止；（5）表示服务从停止状态变成运行状态。此时，主协调器通过重试或者重新询问的方式得知 WS 运行良好；（6）表示服务正常结束。此时，WS 在正常时间内完成自身任务并通过协调器已经向主协调器发送了接口信息和全局变量信息。

定义 2　服务组合流程类型 Composition_Type ∈ {sequence, switch, while}。

定义 3　组合流程类型的扩展类型 Type_exp，它和服务组合流程类型 Composition_Type 一起标志各个不同流程，它的定义如下：

Type_exp ∈ {Null, switch_begin, switch_body_n, switch_end, while_begin, while_body, while_end}。Null 表示流程的类型为 sequence（顺序流程）；switch_begin, switch_body_n, switch_end 分别用于表示 switch 操作的开始，第 n 条路径，操作结束；while_begin, while_body, while_end 分别用于表示 while 操作的开始，循环体，操作结束。

2.3　关键部件设计类图

FT4WSC 中的多 Agent 框架设计如图 4 所示：AgentManagement 表示该 Agent 主要负责其它 Agent 的控制和管理，它可以产生并管理其它的 Agent。AgentManagement 包括 ChiefCoorAgentAD（保存主协调器的位置），CoorAgentAD（保存协调器的位置），SaveAgentAD（保存存储器的位置），ChiefCoorAgentNUM, CoorAgentNUM, SaveNUM 等 6 个变量，以及 newChiefCoorAgent(), newCoorAgent(), delChiefCoorAgent(), delCoorAgent(), delSaveAgent() 等 6 个方法。

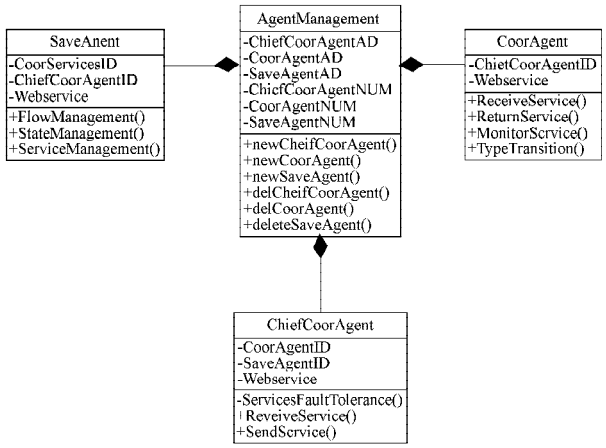

图 4　多 Agent 系统设计类图

Fig. 4　The design diagram of multi agent system

SaveAgent 表示该 Agent 主要负责存储区的管理。它包括了 CoorAgentID, CheifCoorID, Webservice 3 个变量，以及 FlowManagement(), ServiceManagement(), StateManagement() 3 个方法。

ChiefCoorAgent 表示该 Agent 主要复制协调各个部件之间的工作，管理容错策略等。它包含 CoorAgentID, SaveAgentID, Webservice 3 个变量，ServicesFaultTolerance, ReceiveService(), SendService() 3 个方法。CoorAgent 表示该 Agent 主要负责检测和管理本地的 Web 服务、部分数据类型的转换。它包括 ChiefCoorAgentID, Webservice 2 个变量，ReceiveService, ReturnService, MonitorService, TypeTransition 4 个方法。

2.3.1　主协调器 Agent 设计

主协调 Agent 的主要功能：对整个流程的监控、对协调 Agent 的监控、采取合适的容错策略等。主协调 ChiefCoorAgent 的设计如图 5 所示，主要类有：ChiefCoorAgent, ChiefCoorAgentClass, ServiceFaultTolerance, MonitorCoorAgent 等。其中，ChiefCoorAgent 主要负责与其它 Agent 间进行通信，为该 Agent 的接口类；ChiefCoorAgentClass 主要负责抽象流程的实例化；ServiceFaultTolerance 主要负责容错策略实现，MonitorCoorAgent 主要负责监控协调器 Agent 状态，以

便及时发现失败；AgentInformation 主要记录容器中其它 Agent 的状态和地址，以便主协调 Agent 能够找到它们。

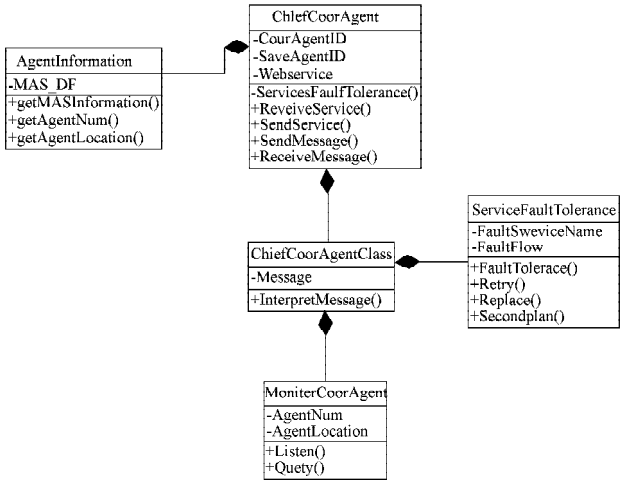

图 5 主协调 Agent 设计类图

Fig. 5 The design diagram of chief coordinating agent

2.3.2 协调器 Agent 设计

协调 Agent 的主要功能：监控本地服务执行情况，容错信息，数据类型转化处理。协调 Agent 的设计如图 6 所示，它主要的类有：CoorAgentClass 负责主协调 Agent 的信息解释；WebserviceMonitor 负责管理本地 Web 服务的执行；DataTransition 负责部分数据类型的转化。

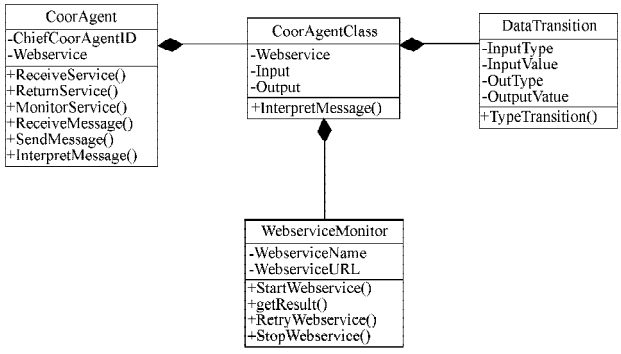

图 6 协调 Agent 设计类图

Fig. 6 The design diagram of coordinating agent

2.3.3 存储 Agent 设计

存储 Agent 的主要功能：接受来自主协调 Agent 的信息，更新流程中的服务状态，更新存储区中的流程信息，替换服务，返回给主协调 Agent 信息，WSDL 文档解释。存储 Agent 的设计如图 7 所示，

WebserviceState 负责管理服务的状态；UpdateFlow 负责更新组合服务的流程；UpdateService 负责服务替换失败的服务；WSDLInterpreter 负责解释 WSDL 文档以获得服务的 IOPE。

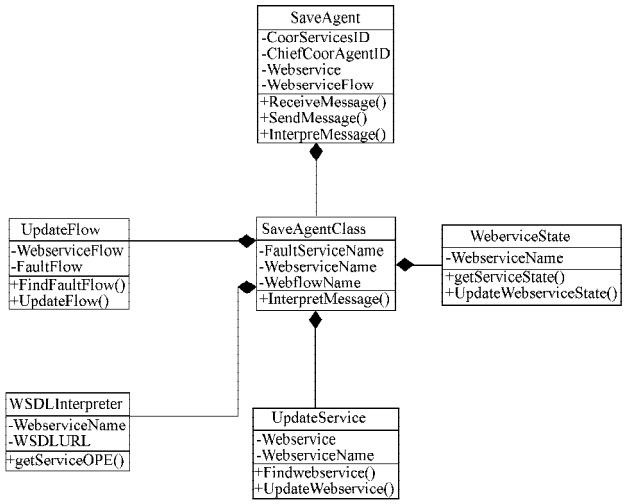

图 7 存储 Agent 设计类图

Fig. 7 The design diagram of store agent

3 组合服务容错的恢复策略

设计容错策略的目标是为了保证服务的功能得以实现，并且尽可能的减小系统开销。本文从服务组合管理者的角度出发，基于 3 种基本的容错策略：重试、替换、局部重构，提出了组合服务容错的恢复策略。

1）重试。不管对哪一级的失败，只要服务所处的状态为 unknow 时，管理者都可向协调器发送重试信息。

2）替换。服务替换的原理较简单，分别搜索各个任务的候选服务集，从中选择最佳服务作为执行该任务的组件服务。目前，服务替换策略通常采用直接加权法方法对候选服务实行打分机制[13]，本文采用直接从服务缓冲区选取最优的服务进行替换失败服务来实现替换策略。

3）局部重构。在组合结构中，Web 服务的失败会影响流程的执行，这种影响不仅仅只对失败服务本身，还可能会影响到部分流程的执行。如：在循环结构中，循环体内的服务出现失败，就会影响整个循环体。针对这种情况，本文采取了将整个循环体重构的方法以确保服务功能的实现；而对于分支结构则采取了重构整条链路来进行失败恢复。

4 系统分析

实验环境为服务器 1 台（用于部署 UDDI，负责执行引擎的运行与服务检测、恢复等），台式机 7 台（用于提供服务，模拟各类型的服务失败），对于各类失败采用何种策略进行恢复在文献[12]中已经说明，在此不累述。实验中分别对一类（0 级失败）、二类失败（1 级失败）进行了 2 000 次的测试实验，所有流程均能执行完成，由此证明了该容错方法的可行性。实验

中还测试了失败检测时间对不同类型的平均失效时间的影响。

由图 8 系统检测开销图，图 9 系统处理开销图可知，当检测时间间隔大于 4 s 时曲线基本趋于水平，这样在保证组合流程执行完成的前提下，不管是平均失效时间还是失败检测时间都是在可以接受的区间内，所以，本文认为系统容错的开销是可以接受的。

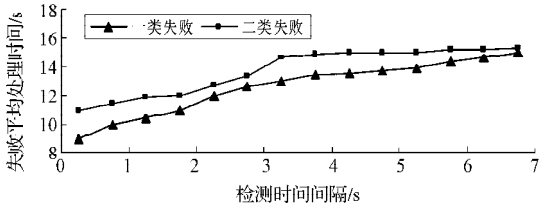

图 8　系统检测开销

Fig. 8　System testing overhead

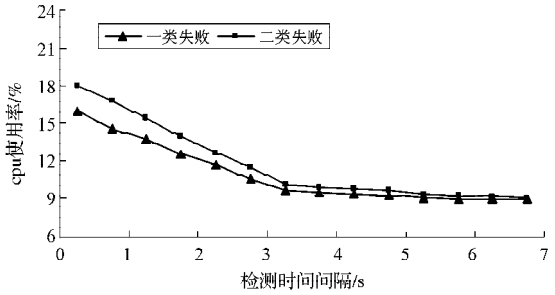

图 9　系统处理开销

Fig. 9　System processing overhead

5　结语

本文针对 Web 服务的特点，提出了服务容错基本框架 FT4WSC，在优先保证组合服务流程执行完成的前提下，提出了 Web 服务组合的容错方法，最后通过实验验证了方法的可行性。下一步工作是在系统中加入语义信息和事务管理，能够进一步对文献[12]中的第三类失败进行容错，从而提高组合服务的可靠性。

参考文献：

[1] 柴晓路. 架构 Web Service：什么是 Web 服务?[EB/OL]. [2001-07-01]. http：//www.ibm.com/ developerworks/ cn/ webservices/ws-wsar/part2/.
Chai Xiaolu, Web Service Framework: What is Web Service? [EB/OL]. [2001-07-01]. http：//www.ibm.com/ developerworks/ cn/webservices/ws-wsar/part2/.

[2] 刘玲霞. Web 服务容错关键技术研究[D]. 长沙：国防科技大学，2006.
Liu Lingxia. The Research on Key Technologies of Fault Tolerant Web Services[D]. Changsha：National University of Defense Technology，2006.

[3] 汤景凡. 动态 Web 服务组合的关键技术研究 [D]. 杭州: 浙江大学，2005.
Tang Jinfan. Research on Key Technologies for Dynamic Web Services Composition[D]. Hangzhou： Zhejiang University，2005.

[4] 蔡美玲. 基于 QoS 的 Web 服务选择及组合服务运行时容错研究[D]. 长沙：湖南师范大学，2007.
Cai Meiling. The Research of QoS-Aware Web Services Selection and Fault-Tolerance of Runtime Service Composition [D]. Changsha：Hunan Normal University ，2007.

[5] Casati F，Ilnicki S，Jin L，et al. Adaptive and Dynamic Service Composition in eFlow[C]//Proc. Of the International Conference on Advanced Information Systems Engineering，Stockholm：Computer Science Press，2000：13-31.

[6] Casati F，Shan M C. Event-Based Interaction Management for Composite E-Services in eFlow[J]. Information Systems Frontiers，2002，4(1)：19-31.

[7] Alwagait E， Ghandeharizadeh S. DeW： A Dependable Web Services Framework[C]// Proceedings of 14th International Workshop on Research Issues on Data Engineering. USA：Computer Society Press，2004：111-118.

[8] Aghdaie N，Tamir. Y Implementation and Evaluation of Transparent Fault-Tolerant Web Service with Kernel-Level Support[C]//Proceeding of the IEEE International Conference on Computer Communications and Networks Miami. Florida：Computer Society Press，2002：63-68.

[9] Deepal Jayasinghe. FAWS for SOAP-Based Webservices-A Client-Transparent Fault Tolerant System for SOAP-Based Web Services.[EB/OL]. [2005-10-31]. http：//www-128. ibm.com/ developer- works/Webservices/library/ws-faws/.

[10] Deron L，Fang C，Chen C，et al. Fault-tolerant Web Service [C]//Proceedings of 10th Asia-Pacific Software Engineering Conference (APSWC’03). Thailand：IEEE CS Press，2003：310-319.

[11] Santos GiulianaTeixeira，lung LauCheuk， Carlos Montez. FTWeb：A Fault Tolerant Infrastructure for Web Services [C]//Proceedings of the 2005 Ninth IEEE International EDOC Enterprise Computing Conference (EDOC’05). Enschede：IEEE Computer Society Press，2005：95-105.

[12] 唐　渊，金可音，周　昆，等. Web 服务失败分类法[J]，湖南工业大学学报，2009，23(2)：73-76.
Tang Yuan，Jin keyin，Zhou kun，et al. A Fault Taxonomy about Web Services[J]. Journal of Hunan University of Technology，2009，23(2)：73-76.

[13] Yoon Seokhyun，Kim Dongjoon，Han Sangyong. WS-QDL Containing Static， Dynamic， and Statistical Factors of Web Services Quality[C]// Proceedings of the IEEE International Conference on Web Services. Florida：IEEE Computer Society Press，2004：808-809.

（责任编辑：罗立宇）