文章编号:1001 - 9081(2006)10 - 2294 - 03

基于移动 Agent 的 Web 服务组合执行框架设计

周远长,钟 勇,朱嘉鲁
(中国科学院 成都计算机应用研究所,四川 成都 610041)
(ychzhou04@ mails. gucas. ac. cn)

摘 要:对 Web 服务组合技术深入分析,从 Web 服务组合的控制执行层面入手,结合移动 Agent 的优势,提出一种基于移动 Agent 的 Web 服务组合执行框架。对于复杂 Web 服务组合,框架可将其分解成若干相互依赖的任务片,其中任务片对应一个线性的子 Web 服务组合,然后分配给多个移动 Agent 进行分布式执行。

关键词:Web 服务组合; 移动 Agent; 任务片
中图分类号: TP311.52;TP393.07 **文献标识码**:A

Design of Web service composition execute platform based on mobile Agent

ZHOU Yuan-chang, ZHONG Yong, ZHU Jia-lu
(*Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdou Sichuan* 610041, *China*)

Abstract: With in-depth analysis of Web service composition technique, starting from the execute aspect of Web service composition and combining the advantage of mobile Agent, an execute platform for Web service composition based on mobile Agent was proposed. As for complicated Web service composition, the platform can at first decompose it into several interdependent task pieces which correspond to one linear sub Web service composition, and then assign them to several mobile Agents to carry out the distributive execution.

Key words: Web service composition; mobile Agent; task piece

0 引言

目前对 Web 服务组合已有大量的研究,就其执行控制技术而言,主要有两种方式:1)采用集中式控制,如微软的 Biztalk 和 IBM 的 WBI-SF (WebSphere Business Integration Server Foundation),此类集中式系统在控制 Web 服务组合执行时,由系统中心控制引擎统一控制 Web 服务的执行顺序和数据传递。在 Internet 环境下,采用集中式控制方式存在很多缺点。首先无法实现负载平衡,当用户访问量过大时,其性能会明显下降;其次,由于要集中式控制 Web 服务组合执行和数据传递,必然增加网络开销和处理时间,造成服务质量下降。2)采取 Agent 技术,如文献[2]提出的一个移动 Agent 控制整个组合流程执行方式,这种方式造成移动 Agent 本身复杂化,如果再加上安全性和可靠性控制等机制,那么移动 Agent 会更加臃肿,不利于充分发挥其灵活性。

鉴于上述 Web 服务组合执行控制方式的缺点,本文提出基于移动 Agent 的 Web 服务组合流程控制执行框架。

1 移动 Agent

移动 Agent 最初设想是能够代替远程过程调用[4],它是一种特殊的 Agent,能通过计算机网络传输自身,并在远程平台重新启动执行的一类 Agent。移动 Agent 的结构包括固定部分和可变部分两部分。固定部分是每一个移动 Agent 生成时就拥有的基本功能模块,大致可分为三类:通信接口、动态加载管理和资源管理;可变部分由运行过程中,动态加载的功能函数组成。当移动 Agent 进行移动操作时,固定部分可以由目的主机自动生成,只移动可变部分即可,这种设计使得移动 Agent 更适合移动[3]。本文基于移动 Agent 设计 Web 服务组合框架,主要依托于移动 Agent 的以下几个方面的优势:

1) 移动 Agent 具有迁移能力,可以 Agent 平台之间迁移,一方面可以减轻网络上的数据流量,提高了服务响应速度,另一方面通过迁移能够克服集中是控制服务器负载过重的缺点,实现负载平衡。

2) 移动 Agent 具有交互和协调能力,使得在进行任务处理时,可以创建多个 Agent 通过彼此协调完成任务,从而提高工作效率。

3) 移动 Agent 具有自主执行能力,在分配了任务之后,可以独立的控制整个任务的完成,摆脱了集中式控制的限制。

由上述优势可知,与普通 Agent 相比移动 Agent 更灵活、更高效,更适合用于 Internet 环境下的 Web 服务组合执行,所以框架基于移动 Agent 平台设计。

2 系统框架和执行流程

系统框架运行在 Java 环境中,如图 1 所示。图中的移动 Agent 平台为 Agent 的运行提供必要的环境服务,如目录服务和消息通信服务等,另外:

1) WSAG(Web Services Agent Gateway)是由 WSAI (Web Services and Agent Integration) 提供的开放组件,它可以将 Agent 发布为 Web 服务,将 SOAP 调用转化为 Agent 间的 ACL 通信语言[7]。

收稿日期:2006 - 04 - 13

作者简介:周远长(1981 -),男,山东人,硕士研究生,主要研究方向:软件过程、软件质量; 钟勇(1966 -),男,四川人,研究员,博士生导师,博士,主要研究方向:软件过程、软件质量、现代集成制造系统(CIMS)理论; 朱嘉鲁(1982 -),男,山东人,硕士研究生,主要研究方向:分布式系统、Web 服务.

2）服务 Agent 组主要负责为用户提交的请求提供后台服务,其中守护 Agent 负责监听由 WSAG 传送过来的服务请求,当它收到请求后,就唤醒计划 Agent;计划 Agent 对收到的服务组合请求划分任务片,并设置相应任务片的数据依赖关系和控制依赖关系,如任务片的前置条件或后置条件等;配置 Agent 负责为任务片产生过程 Agent 组,其中包括一个结束过程 Agent,它负责将执行结果或异常返回用户端,另外配置 Agent 根据需要为它们配置行为。

3）过程 Agent 组负责 Web 服务组合控制执行,过程 Agent 根据分配到的任务片,通过迁移和彼此协调完成执行,最后返回执行后的结果或异常。

4）行为库和流程库主要起到知识库的作用,辅助框架的运行。

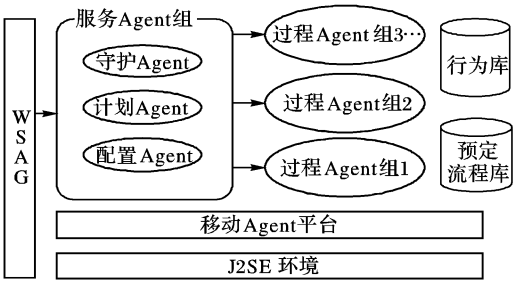

图 1　框架图

该框架的执行流程如图 2 所示。

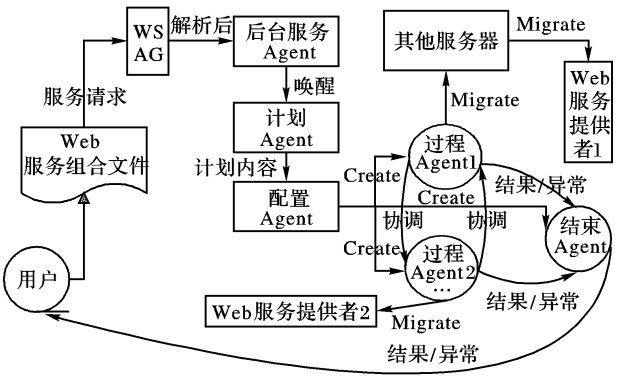

图 2　执行流程

3　框架设计和实现

3.1　示例

本文使用示例,假设有图 3 所示的 Web 服务组合流程模型(使用有向图模型描述)。

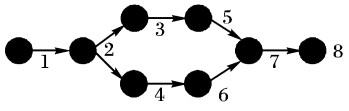

图 3　示例图

3.2　过程 Agent

过程 Agent 是任务片的执行载体,为了对过程 Agent 进行深入描述,首先做如下定义:

定义 1　活动(Activity),一个活动可以用一个四元组表示(AID,WS,INPUT,OUPUT),其中 AID 表示活动的序号,WS 表示要执行的某个 Web 服务(非空),INPUT 和 OUTPUT 表示活动的输入和输出参数。

定义 2　任务片(TaskPiece),任务片由一组严格有序的活动组成(实际上对应一个线性的子 Web 服务组合),可以表示成六元组(TID,(Activity)$^+$,INPUT,OUPUT,PRE, POST),其中 TID 是该任务片的唯一标识,INPUT 和 OUTPUT 表示活动的输入和输出,PRE 和 POST 分别表示该任务片执行前要满足的前置条件和执行完成后 Agent 要执行一些操作。

定义 3　过程 Agent(ProcessAgent),可以表示成三元组(PID, TaskPiece,TASKMAP),其中 PID 用来标识过程 Agent,TaskPiece 表示该过程 Agent 要执行的任务片,TASKMAP 表示任务片分配的全局目录,其中 TASKMAP 的每一个目录项又可表示为元组(PID,TID),这里的 PID 表示某过程 Agent 的标识,TID 表示过程 Agent 为 PID 分配得到的任务片标识,通过此目录,系统就可以根据任务片之间的依赖关系,映射到过程 Agent 之间的依赖关系。

结合本文设计的框架需求,给出过程 Agent 的几个基本行为:

1）Execution(TaskPiece),顺序执行任务片中的每一个活动。

2）Migration(host),过程 Agent 在执行活动时,通过执行这个行为,迁移到其他平台。

3）Wait(TaskPiece[ ]),过程 Agent 在执行分配的任务片之前,要等待任务片数组 TaskPiece[ ]执行完成。

4）Invoke(TaskPiece[ ]),过程 Agent 执行分配的任务片后,要通知任务片数组 TaskPiece[ ]执行。

5）Destroy( ),过程 Agent 执行分配的任务片,并将执行结果提交后,销毁自身。

6）TrowExeception( ),将过程 Agent 的执行异常发给结束过程 Agent。

3.3　计划 Agent 设计

计划 Agent 是服务 Agent 组中重要的角色,正常情况下它处在睡眠状态,当有服务请求到达时,就被守护 Agent 唤醒,然后它要对接收到的 Web 服务组合文件进行任务分片。在进行任务分片时,本文将 Web 服务组合流程抽象成有向图模型来描述,这样使得一个 Web 服务对应一个有向图的节点,同时也对应一个活动。所以下面讨论任务分片算法就可以建立在有向图的基础上。分片算法目的是将 Web 服务组合复杂流程划分若干个任务片,原则是每个任务片对应一个线性的、简单的子 Web 服务组合,且任务片之间的数据或同步依赖关系仅存在于它们包含的子 Web 服务组合的头部或尾部。详细的算法描述如下:

步骤 1　扫描有向图,根据节点 v_i 的入度 i_n 和出度 o_n,将有向图的节点划分成两个集合 $V_1 = \{v_i \mid i_n > 2 \cup o_n > 2\}$, $V_2 = \{v_i \mid i_n < 2 \cap o_n < 2\}$,并对节点设置编号 ID_i,编号原则,如果存在 $v_i \rightarrow v_j \rightarrow v_k$,则 $ID_i < ID_j < ID_k$,不允许重复。

步骤2　将集合 V_2 划分成 N 个独立的子集合 VC_i,划分子集合的原则,对于任何的 $v_i \in VC_i$,如果存在 $v_i \rightarrow v_j$,则 $v_j \in VC_i$,直到集合 V_2 划分完毕。

步骤 3　将集合 V_1 的节点尽量合并到子集合 VC_i 中,可根据情况进行:1）如果 v_i 入度小于等于 1 且出度大于 2,则将它合并到 v_i 前驱节点 v_j 所在的子集合中;2）如果 v_i 入度为大于 2 且出度小于等于 1,则将它合并到 v_i 后继节点 v_j 所在的子集合中,其他情况则将 v_i 划做一个子集合 VC_i。划分完毕后,根据节点编号,对子集合节点排序。

步骤 4　根据节点、Web 服务和活动的一一对应关系,将上面步骤得到子集合映射成活动序列即得到任务片组,对每一个任务片设置标识 TID。

步骤 5　根据文献[6]总结的 20 种工作流模式和当前 Web 服务组合模型,映射得到每个任务片的前置条件和后置

条件(实际上每个任务片的前置条件和后置条件,只与任务片的开始活动和结束活动有关)。

通过实现上述算法就实现了计划 Agent 的功能,这样对于一个给定 Web 服务组合文件,经过计划 Agent 的处理就得到了计划文件,以文章所举的示例为例,算法执行完步骤 3 得到划分后的子集为:{1、2}、{3、5}、{4、6}、{7、8},经过步骤 4 和步骤 5 的处理后生成的计划文件,可用 XML 语言表示如下:

```
<TaskPiece TID = "T1" >
    <PRE > </PRE >
    <POST >
        < TaskPiece - set > < TaskPiece > T2 </TaskPiece >
        <TaskPiece > T3 </TaskPiece > </TaskPiece - set >
    </POST >
    < INPUT > parameter </INPUT > < OUPUT > parameter
    </OUPUT >
    <Activity - set >
        <Activity AID = "A1" >
        < webservice > 1 </webservice > < INPUT > parameter
            </INPUT > < OUPUT > parameter </OUPUT >
        </Activity >
        <Activity AID = "A2" >
        < webservice > 2 </webservice > < INPUT > parameter
            </INPUT > < OUPUT > parameter </OUPUT >
        </Activity >
    </Activity - set >
</TaskPiece >
<TaskPiece TID = "T2" >
    …
<TaskPiece TID = "T3" >
    …
```

3.4 配置 Agent 设计

配置 Agent 的职责是为任务片产生并分配过程 Agent,并为每个过程 Agent 动态配置必要的行为,另外还可以产生一个结束过程 Agent,它可以监控任务的执行状况,并接收过程 Agent 执行时产生的异常或结果,返回客户端。配置 Agent 的具体行为可以描述如下:

步骤 1 根据上面产生的任务片的数量,创建相应数量的过程 Agent 和一个结束过程 Agent,为每一个过程 Agent 分配一个任务片。

步骤 2 为每一个过程 Agent 加载必要的行为,如上面提到过程 Agent 的几种基本的行为等等。

步骤 3 根据分配信息,设置每一个过程 Agent 的 TASKMAP 目录信息,同时也要记录结束过程 Agent 的必要信息。

经过上面的配置步骤后,每一个过程就得到了需要它执行的任务和行为,就可以迁移到其他平台进行执行,而无需集中式控制,从而实现 Web 服务组合流程分布式执行的目的。

3.5 守护 Agent 设计

守护 Agent 的设计,相对比较简单,它类似于操作系统的一个后台执行进程,负责监听用户的请求,当有用户请求到达时,则进行相应的处理,如唤醒计划 Agent 处理请求等等,然后,继续进入监听状态。所以守护 Agent 的具体行为可以用下面函数描述:

```
public void Listen() {
  while(1) {
    if( request! = null) {wakeup( 计划 Agent); do something else; }
  }
}
```

3.6 行为库和流程库

行为库实际上由一组行为类组成(如上面提到的几种过程 Agent 的基本行为等),为了方便配置 Agent 的使用,采用 XML 语言格式来表示行为配置文件,格式如下:

```
< BehaviorMap xmlns = "http: //test. com/behaviour" >
    < Behavior name = "example1" >
        < class name = "test. Behaviour. example1"/ >
        < description > </description >
        < StartOther name = "test" >
    </Behavior > …
</BehaviorMap >
```

< Behavior > 元素:包括三个子元素 < Class > 元素指定完整的类名,以便于动态加载; < description > 元素表示备注; < StartOther > 元素表示在 Behaviour 结束时,启动其他 Behaviour 的名字。这样配置 Agent 就可以通过此配置文件,将实现定义好的行为类动态加载给过程 Agent。

流程库是预先定义好的一些 Web 服务组合流程(存储经过计划 Agent 执行后得到的计划文件),它实现某一特定的功能,用户可以直接提交请求此类功能服务,框架从流程库中找到相应的流程执行并返回结果。这样对一些经典的通用的 Web 服务组合,就无需用户每次使用都自定义流程,方便用户使用。

4 结语

本文提出了一个基于移动 Agent 的 Web 服务组合执行框架,并详细介绍框架的设计。框架的核心部分是服务 Agent 组和过程 Agent,它成功结合了移动 Agent 的优势实现了 Web 服务组合流程的分布式执行。由于过程 Agent 执行的任务片对应一个线性的、简单的子 Web 服务组合,无需复杂引擎逻辑的控制,所以也保证了移动 Agent 的灵活性。另外,该框架还存在一些不足之处,如框架的容错能力问题、任务分片算法的优化问题,有待进一步完善,这将是下一步研究工作的重点。

参考文献:

[1] LI G, ROBERTSON D, CHEN-BURGERV Y-H. A generic multi-agent system platform for business workflows using Web services composition[A]. Intelligent Agent Technology, IEEE/WIC/ACM International Conference on[C]. 2005.301 - 307.

[2] ZHU ZQ, SANG LL, LI X. Mobile-agent-based Web service composition[J]. Lecture Notes in Computer Science. 2005, 3795: 35 - 46.

[3] 朱嘉鲁,何希琼. 基于移动 Agent 的 Web 服务集成[J]. 南京大学学报,2005,41:411 - 416.

[4] WOOLDRIDGE M. An introduction to multi - agent systems [M]. BEIJING: Publishing House of Electronics Industry, 2003.

[5] BUHLER PA, VIDAL JM, VERHAGEN H. Adaptive workflow = web services + agents[A]. Proceedings of the International Conference on Web Services[C]. CSREA Press, 2003. 131 - 137.

[6] VAN DER AALST WMP, TER HOFSTEDE AHM, KIEPUSZEWSKI B, *et al*. Workflow patterns[J]. Distributed and Parallel Databases, 2003, 14(1): 5 - 51.

[7] White stein Information Technology Group AG. Web services agent integration project[EB/OL]. http://wsai. sourceforge. net/index. html, 2003.

[8] KORHONEN J, PAJUNEN L, PUUSTJÖRVI J. Automatic composition of web service workflows using a semantic agent[A]. Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03)[C]. 2003.566 - 569.