一种网格作业管理器的研究与实现

王 勇 夏先波 万 雨 胡春明

(北京航空航天大学 计算机学院, 北京 100083)

摘 要: 随着网格技术以及应用的发展,网格作业的形式越来越呈现出复杂的流程性特征. 网格作业的定义、执行和管理需要软件系统的支持,其中作业管理器是支撑软件系统中的核心部件,它负责作业定义的解释执行,并提供运行时的监控管理功能. 在借鉴服务组合技术的基础上,提出了一种网格作业描述语言,并设计实现了支持流程性作业的作业管理器. 作业管理器由作业管理器入口、会话管理器、实例管理器、作业实例、活动实例以及活动调用管理器等主要部件组成,为网格作业的运行提供了执行引擎. 作业管理器具有分层消息处理结构,简化了处理过程的复杂性;支持的 3 种不同的会话机制,能够满足作业之间以及作业和成员服务之间会话的灵活性要求;基于事件的消息传递机制使得作业管理器更加符合面向对象的设计要求.

关 键 词: 网格; 作业; 作业管理器; 工作流

中图分类号: TP 393.09

文献标识码: A **文 章 编 号**: 1001-5965(2006)04-0484-05

Research and implement of a grid job manager

Wang Yong Xia Xianbo Wan Yu Hu Chunming

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

Abstract: With the development of grid and grid applications, more and more flow jobs have appeared. Definition, execution and management of jobs need the support of software system and the job manager is the core component. The job manager interprets the job definition and provides the monitor function. A job description language is proposed and a job manager is designed based on early work of Web service composition. The job manager is composed of invoker, conversation manager, instance manager, job instance, activity instance and activity invoker. The job manager provides an executing engine for grid jobs. The layered architecture of the job manager simplifies the message processing. Three different conversation mechanisms supported by the job manager satisfy the requirement of the conversation between jobs and between jobs and component services. Event-based message exchange mechanism makes the relation among components of the job manager clearer and meets the requirement of the object-oriented design.

Key words: grid; grid job; job manager; workflow

随着网格技术和 Web 服务技术的融合发展,在 OGSA(Open Grid Services Architecture)[1] 中服务已经成为一个核心概念,基于服务的网格体系结构[2,3] 已经初步形成,并在一系列的系统中得到了实现,如 Globus Toolkit V 3.0 以及北京航空航天大学的面向网格的应用支撑环境 WebSASE4G[2] 等. Web 服务资源框架规范 WSRF(Web Service Resource Framework)[4] 把 OGSI(Open Grid Services Infrastructure)规范[5] 的内容分为 5 个不同的、互为补充的规范,以 Web 服务

收稿日期: 2005-02-28

基金项目: 国家自然科学基金重大研究计划资助项目(90412011);教育部中国教育科研网格 ChinaGrid 公共服务平台资助项目(03-05);国家高科技(863)计划国际合作资助项目(2003AA119030)

作者简介: 王 勇(1974-),男,山东临朐人,博士生,wangy@act.buaa.edu.cn.

作为构成服务网格的基本元素,保持了同现行 Web 服务标准的兼容性.

随着网格作业越来越复杂化,作业的形式从支持单个应用的作业和批作业,到目前对流程性作业的支持已经成为构建网格的内容. 在目前主流的网格平台中,如 Globus Toolkit,尚没有提供对于流程性作业的支持. 在以服务为基本构成元素的网格[2]中,作业是对多个服务的复合,同时作业的形式也表现为一个服务. 也就是说,在流程性作业中需要定义成员服务的执行次序、执行过程中相关数据的处理、执行过程中的异常处理机制、作业实例的标识机制、成员服务在作业执行过程中扮演的角色等信息.

在 Web 服务组合研究工作的基础上,本文设计并实现了一种作业管理器,提供了对于服务网格中流程性作业的支持.

1　作业描述语言简介

传统的作业只关注单一任务的提交和执行过程,其描述语言如 RSL[6],ClassAD[7],主要描述任务执行过程中相关资源的配置情况,如执行程序的名称、相关调用参数、对任务执行的环境要求等. 网格技术的应用发展促使网格作业的形式越来越复杂,简单的作业(如单一任务的作业、批作业等)越来越不适应技术发展的需要,在以服务为基本构成元素的网格中,要求通过一次作业执行把多个服务组装到一个流程中,并把作业自身包装成一个服务发布出去供外界访问,传统的作业描述语言不能适应这种技术和需求的变化.

网格计算与 Web 服务技术日益融合的趋势使得不愿违背现有 Web 服务标准,基于商业过程执行语言(BPEL4WS)[8]进行扩展,建立了网格作业描述语言 GJDL(Grid Job Description Language),同传统的网格作业描述语言相比,GJDL 在网格作业中引进工作流机制,使得作业描述语言可以描述网格中复杂的业务流程,进而支持流程性作业;GJDL 完全面向服务构造,符合网格技术和 Web 服务技术日益相融合的趋势,作业是由组合多个服务构成的,同时作业本身也发布为一个服务供外界访问.

BPEL4WS 虽然是服务组合方面的占据主导地位的描述语言,但是它本身也存在一些缺点,例如在成员服务的绑定方式方面存在局限、过于简单的生命周期管理功能、不支持对等交互以及语言本身过于冗余复杂等. 最重要的是,由于网格环境和应用的特点以及网格技术的最新进展,BPEL4WS 不能直接应用在网格作业的构造中,例如不支持最新的网格技术规范 WSRF,对网格中的大计算量作业和大数据量交互作业没有提供支持. 在 BPEL4WS 的基础上针对网格环境和网格应用的特点进行了裁减和扩充,建立了 GJDL,GJDL 同 BPEL4WS 相比具有以下的特点:

1) 支持最新的网格技术规范 WSRF,对 WSRF 规范的支持包括 3 个方面:①支持对于有状态资源的访问,同时作业本身可以发布为一个带有资源的服务供外界访问;②在对成员服务的调用过程中根据 WS-Lifetime 规范的内容加入对于成员服务的生命周期管理功能;③把成员服务的资源属性作为对服务进行调度管理的依据.

2) 支持大计算量类型的作业. 在网格的大运算量的环境中,一个步骤通常需要多个成员服务来共同完成,在 GJDL 引入一种新的类型的活动,称为 pFlow,该活动可以由多个完成相同计算任务的成员服务共同完成.

3) 支持大数据量类型的作业. 在成员服务之间存在大数据量交互的情况下,完成作业管理功能的作业管理器可能充当作业执行过程中的瓶颈,GJDL 利用 WSRF 提供的通知机制使得成员服务彼此之间能够直接进行通讯,从而避免了作业管理器这个逻辑上的瓶颈.

4) 更多的成员服务绑定方式. BPEL4WS 只提供了一种成员服务绑定方式,本文对 BPEL4WS 的成员绑定方式进行了扩充,提出了 4 种不同的绑定方式:本地方式、静态方式、注册中心方式以及动态绑定方式.

5) 更为简洁有效的编程结构. 在 BPEL4WS 中存在过于冗余复杂的语言元素和编程结构,针对网格环境的特点,对 BPEL4WS 进行了裁减,使得 GJDL 本身更为简洁,但又能够满足网格应用的需求. 在 GJDL 中保留了 Invoke, Receive, Assign, Reply 4 种核心功能的原子活动和最为核心的 4 种类型的结构化活动 Sequence, Flow, While 和 Switch.

2　作业管理器的设计

2.1　功能分析

网格作业管理器的主要功能是按照作业的定义,对来自客户端的请求进行处理,其核心功能包括以下 4 个方面:

1) 会话管理功能. 作业的执行过程可能需要涉及到多个作业实例之间的交互,为了保证实例之间的请求和异步响应消息能够正确地递交,需

要提供作业实例级别的会话管理功能.

2）作业实例的调度管理功能. 出于对作业管理器载荷和作业实例管理方面的考虑,以及为了对外界的请求进行分级处理,需要对作业实例和外界的请求进行调度,对作业实例提供生命期管理功能.

3）作业实例的解释执行功能. 作业实例的解释执行功能包括实例中活动的执行顺序的控制、相关数据的处理、控制条件的求值、原子功能活动的执行以及执行过程中的异常处理等.

4）对外界服务实现的调用管理功能. 主要包括调用前对于服务实现的绑定功能,以及请求消息的预处理功能,如请求消息的装配、解密等处理功能.

此外还包括一些附加功能,例如定义文件的解析功能主要完成的是把作业定义文件转变为静态模型表示的过程;编译功能主要完成的是生成新的作业实例的过程,它把静态的模型表示转变为运行时的实例;作业管理器的配置功能完成了对作业管理器一些参数的配置;SOAP 消息管理功能可以分解为对于简单对象访问协议 SOAP 消息的装配和解析. 还有日志管理等一些常见的功能.

2.2 体系结构

针对上述分析,设计了如图 1 所示的作业管理器. 下面讨论作业管理器体系结构中主要的部件及这些部件间的静态或动态关系.

1）作业管理器入口. 它的主要功能是充当服务容器和作业管理器中其他部件的桥梁,用来接收服务容器中经过预处理的请求消息,并阻塞本次请求过程,直到作业管理器做出响应为止.

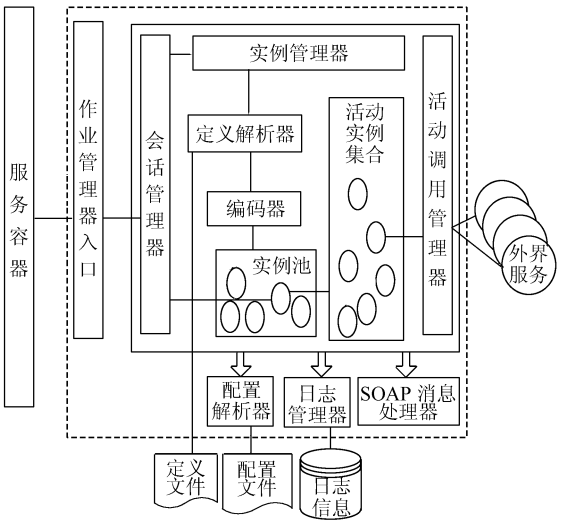

图 1 作业管理器的体系结构

2）会话管理器. 它的主要功能是实现会话管理,使得请求消息能够定位到正确的作业实例,并在响应消息和外出请求消息中标识上会话信息,使得交互另一方的作业管理器能够识别出正确的实例.

3）实例管理器. 它的主要功能是完成作业实例的调度管理,按照调度策略对请求消息进行排队,对作业实例进行生命期状态的管理,对外提供管理操作接口.

4）作业实例. 由实例管理器创建的包含实例数据和运行控制于一体的运行时对象,它自主地完成作业的解释执行功能.

5）活动实例. 它是服务实例的一部分,在实例管理器创建服务实例时一并创建,由作业实例统一管理.

6）活动调用管理器. 完成对于外界服务的调用管理功能,包括对于外界服务的绑定以及请求消息的装配和预处理.

此外,定义解析器完成对于作业定义的解析功能,它由实例管理器发起调用,生成静态的模型对象;编译器完成作业实例的生成功能,它由实例管理器发起调用,最终生成可以自主执行的服务实例和活动实例集合.

2.3 执行原理

为了更为清晰地表达作业管理器中各个部件之间的协作关系和执行原理,对作业 1 次请求的执行过程进行详细说明,如图 2 所示.

1）外界对于作业的请求以 SOAP 报文格式递交给服务容器;

2）请求消息经过服务容器的预处理(文档解析、加密、日志等)后转发给作业管理器入口;

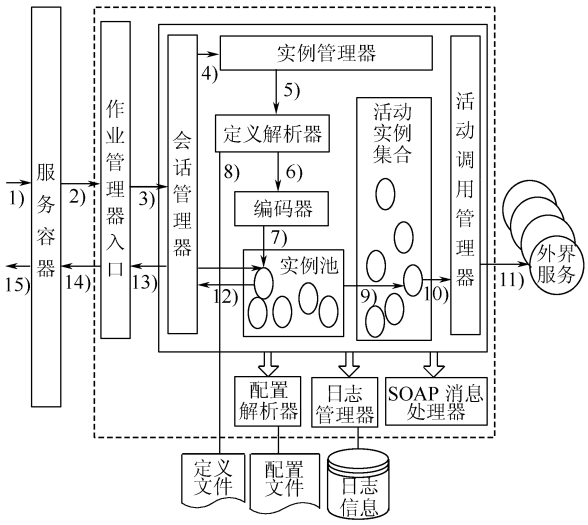

图 2 作业管理器的执行原理

3）作业管理器入口阻塞本次调用,把请求消息转交给会话管理器;

4）会话管理器分离出会话标识信息,根据会话标识信息确定正确的服务实例,如果找到正确的服务实例,则转至 8）;

5）会话管理器没有找到正确的服务实例,并确定该请求信息是对于作业的第 1 次请求,把请求消息经过处理后转发给实例管理器;

6）实例管理器根据请求消息中携带的请求作业的类型、作业定义的地址等信息,调用定义解析器,创建静态模型对象;

7）实例管理器根据创建的静态模型对象调用编译器生成运行时作业实例,并把请求消息转交给新创建的作业实例进行处理;

8）会话管理器把请求消息转交给正确的作业实例进行处理;

9）作业实例开始自主执行,如果在执行的过程中,需要执行调用外界服务类型的活动实例,则把相应的参数传递给正确的活动实例;

10）该活动实例经过适当的数据处理,把请求参数交给活动调用管理器;

11）活动调用管理器接收来自活动实例的请求参数,根据服务实现的绑定策略绑定到合适的服务实现,把请求参数经过装配和预处理之后发给外界服务;

12）外界服务把响应消息返回给活动调用管理器;活动调用管理器把返回的响应消息经过解析和适当的处理后,交给对应的活动实例;活动实例进行适当的数据处理,并通知服务实例更新实例执行的上下文信息;作业实例执行到一定的步骤后,开始对服务的请求做出响应,把响应消息传到会话管理器;

13）会话管理器接收来自作业实例的响应消息,标记上适当的会话标识信息,并对响应消息进行适当的处理,交给作业管理器入口;

14）作业管理器入口终止阻塞过程,把响应消息交给服务容器;

15）服务容器对响应消息经过适当的后处理(如日志、加密等),返回给作业请求者. 至此,整个请求的处理过程结束.

3　作业管理器的实现机制

基于 Java 语言实现了一个作业管理器,下面给出作业管理器中会话管理器、实例管理器和活动调用管理器等主要部件的实现机制.

会话管理器提供了 3 种不同的会话机制:基于会话标识的机制、基于回调地址的会话机制和基于相关集的会话机制,并把这 3 种会话机制结合起来. ①优先使用的会话机制是基于会话标识的会话机制,这是一种自扩展的会话机制,在 SOAP 消息头部携带会话标识作为会话标识信息;②基于回调地址的会话机制,即作业实例在访问其他的服务或者作业实例时带上对方访问自己时的回调地址,以便其他的服务或者作业实例访问自己,这种方式是为了提高会话的灵活性而自扩展的一种机制;③基于相关集的会话机制,它通过 1 组作业的相关数据而唯一确定 1 个实例,作业相关集在作业定义中明确说明,这是一种标准的方式. 总的说来,上述 3 种会话机制的结合使用基本上可以满足作业之间以及作业和成员服务之间的交互场景,并在一定程度上满足会话的灵活性要求.

对作业实例的调度和管理功能是实例管理器的核心功能. 实例管理器采用先进先出(FIFO)策略对作业实例进行调度,即如果实例池中实例的数目小于配置文件中规定的作业实例的最大数目,则创建新的作业实例;否则,则把请求消息置于请求消息队列中等待实例管理器的调度. 在作业实例创建以后,对于作业实例的调度管理就是实例管理器根据不同的事件改变作业状态的过程. 作业实例的状态转换如图 3 所示,作业实例在创建以后处于就绪状态,在对于作业的请求消息或者调用服务的响应消息到达后,作业实例转入运行状态,管理员显式的挂起操作会导致作业实例进入挂起状态,显式的恢复操作会使得作业实例恢复到原来的状态(就绪状态或者运行状态),处于就绪状态和挂起状态的作业实例如果执行超时会进入终止状态,作业实例正常运行结束进入完成状态并销毁作业实例.

在作业实例和活动实例之间采用基于事件的消息传递机制. 作业实例实现为一个独立的线程,作业实例的执行就是不断地处理各种不同的事件,并按照作业定义执行其中的活动实例的过程.

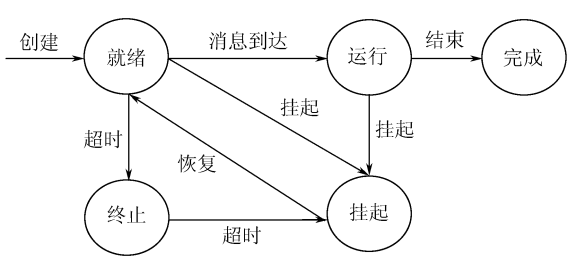

图 3　作业实例状态转换

活动实例是最小的语义单元,每一个活动通过提供一个执行方法来实现自己的执行语义. 作业实例和活动实例之间通过各种不同的事件机制来进行联系,例如接受外界请求的活动、调用外界服务实现的活动在执行结束的时候分别触发相应的事件,作业实例作为事件的接受者在接受到这些事件以后分别做出相应的处理继续作业的执行.

为了便于扩展,活动调用管理器被设计和实现为一个独立的线程,它接受来自外出调用活动的请求参数,组装调用外界服务的请求消息,并按照绑定方式的定义绑定到具体的外界服务并发出调用请求,在接受到响应消息后触发调用结束事件,通知相应的活动实例进行后续处理. 在活动管理器中提供了多种成员服务绑定方式的支持(本地方式、静态方式、信息服务方式、动态方式). 活动管理器的一个主要功能是实现对成员服务调用的调度管理. 为保证调度过程的可靠性,采用了如下的算法来完成对某一个成员服务的调度.

1) 获得作业实例中外出调用活动的静态访问点属性,查看该属性值是否为空,如果不为 空,则转至4);

2) 如果静态访问点属性值为空,需要查询信息中心,根据外出调用活动的虚拟服务标识属性和服务选择条件属性调用信息中心提供的查询接口,获得可用服务的列表;

3) 判断可用服务的列表是否为空,如果为空,返回出错信息,并通知作业实例进行后续处理;

4) 如果不为空,按照一定的选择策略(如随机选择)选择服务进行调用;

5) 在调用的过程中判断外出调用活动的轮询属性是否为空,如果为空,说明是普通 Web 服务,则转至7);

6) 如果轮询属性不为空,说明是 WSRF 标准的服务,需要在服务的执行过程中每隔一定时间更新时间戳,通过成员服务提供的查询条件获得服务的执行状态;

7) 判断服务的调用是否出错,如果出错,则进行重新调度,转至1);

8) 如果执行的过程中没有出错,获得服务的返回结果,整个调度执行过程结束.

需要指出的是,服务容器可以选择不同的实现,分别基于 WebSASE4G 中的服务容器、ASIX 的 SOAP 引擎以及 Globus Toolkit 3.9.1 实现了与作业管理器的集成.

4 结束语

网格技术的发展促使网格作业的形式越来越复杂,简单的作业越来越不适应应用的需求,在以服务为基本构成元素的网格中,要求一个作业把多个服务组装到一个流程中,并把作业自身包装成一个服务发布出去供外界访问. 在目前主流的网格平台中,如 Globus Toolkit,尚没有提供对于流程性作业的支持.

本文首先分析了网格技术和网格中作业管理的相关背景,着重介绍了作业管理器设计方面的问题,包括作业管理器的功能分析、体系结构、执行原理以及实现机制等. 流程性的作业管理在服务网格中是一项刚刚起步的技术,无论是作业的描述模型还是作业管理的体系结构都存在欠缺,需要进一步的研究.

参考文献 (**References**)

[1] Foster I, Kesselman C, Tuecke S. Open grid services architecture roadmap [EB/OL]. http://www.gridforum.org/ogsa-wg/papers/ogsa_roadmap.0.4.pdf, 2004-06-12

[2] Hu C M, Sun H L, Huai J P. WebSASE4G: a web services-based grid architecture and its supporting environment[J]. Journal of Software, 2004, 15(7): 1064~1073

[3] Foster I, Kesselman C, Nick J, *et al*. The physiology of the grid: an open grid services architecture for distributed systems integration [EB/OL]. http://www.globus.org/ research/papers/ogsa.pdf, 2002-01-13

[4] Karl C. The WS-Resource framework [EB/OL], http://www.globus.org/wsrf/specs/ws-wsrf.pdf, 2004-09-02

[5] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations [J]. International J Supercomputer Applications, 2001, 15(3): 1~10

[6] Globus. Resource specification language [EB/OL]. http://www.globus.org/gram/rsl_spec1.html, 2004-11-07

[7] Conder. ClassAD [EB/OL]. http://www.cs.wisc.edu/condor/classad/refman/, 2004-11-01

[8] Curbera F, Goland Y, Klein J, *et al*. Business process execution language for web services [EB/OL]. http://msdn.microsoft.com/webservices/default.asp? pull=/library/en-us/dnbiz2k2/html/bpel1-0.asp, 2002-11-21