动态服务组合中资源选择的研究与实现

许 兰,朱巧明,贡正仙,朱剑非
(苏州大学 计算机科学与技术学院,江苏 苏州 215006)

摘 要:在面向服务的网格中,需要按用户请求动态地根据当前系统中服务的状况,组合出满足用户请求的服务组合。目前,当存在多个服务提供者提供相同功能的服务时,如何动态选择合适的服务却很少被论述,而且真实网格环境中的不确定性(如Cpu、网络的负载)也很少考虑。如何选择最合适的服务所在的资源,是服务组合中必须要考虑的。在分析动态服务组合的基础上,设计了一种动态选择资源的方法。并利用GT4中工厂设计模式给出其实现方式。该设计考虑了网格环境中资源的安全性和不确定性因素,比如资源经常失败,以及Cpu和网络的负载等,从而最大程度地提高服务执行的可靠性和高效性以及均衡系统负载。

关键词:网格;动态服务组合;资源选择;不确定性;工厂模式
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)12-0221-04

Design and Implementation of a Resource Selection in Dynamic Service Composition

XU Lan, ZHU Qiao-ming, GONG Zheng-xian, ZHU Jian-fei
(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)

Abstract: In the service-oriented grid, need to compose basic services dynamically that have existed in the grid to form more sophisticated services according to the user's request. In the grid, however, multiple service providers can provide the same service with different implementation. Currently, dynamic service selection among similar services from multiple providers has seldom been addressed. And uncertainty(e.g. Cpu load and network load)under a real, dynamic grid environment has never been considered. How to choose the proper resource associated with the required service is central to service composition. Based on the dynamic service composition, project a method of dynamic resource selection and provide it concrete implementation supported by factory/instance pattern in GT4. Considering the surety and uncertainty of the grid, the method can maximize the reliability and efficiency of service execution and balance the system workload.

Key words: grid; dynamic service composition; resource selection; uncertainty; factory instance

0 引 言

在服务网格中,所有的资源包括计算、通信和存储等都是以服务的形式存在。面向服务的观点简化了虚拟性,即对同一接口的不同实现的封装。虚拟性,可以在各种异构平台上实现对资源的一致访问。另外,通过虚拟可以实现将基本服务进行组合进而形成更为复杂的高级服务,而事先不必关心这些基本服务如何实现[1]。服务组合是指多个基本服务之间通过一定的协议进行协调共同完成特定的任务。设计OGSA的主要目标就是以标准方式进行服务组合并使组合的过程更加容易。网格环境中的不确定性,如服务器的负载、网络流量等都应在选择服务进行服务组合时需要考虑。所以需要一种灵活有效的服务组合方法,可根据用户的请求充分有效地利用系统已存在的服务动态地组合出满足用户需求的服务组合。

笔者分析了在服务组合中影响服务选择的不确定性因素,设计了一种在动态环境中选择资源的方法。该方法可有效地利用当前环境中服务以及所在资源的信息,选择合适的资源节点用于服务组合以及最大程度降低服务执行失败的可能性。

收稿日期:2007-03-13
基金项目:江苏省高技术研究项目(BG2005020);江苏省教育自然科学基金(04KKB320134)
作者简介:许 兰(1980-),女,安徽合肥人,硕士研究生,研究方向为计算机网络和数据库;朱巧明,教授,研究方向为计算机网络和数据库。

1 相关工作

目前服务执行路径的产生大多都是静态的,即在

实例化开始前执行路径已经确定。这种方法不能保证在按照该路径执行时资源的状态不发生变化，比如在文献[2]和[3]中分别采用的遗传和蚁群算法，它们的重点在于强调最优的全局调度。但是这里的“最优”只是在设计路径的期间，并不能保证在执行的期间也是最优的，而网格的动态性以及遗传和蚁群算法本身所需的大量计算时间更是加大了这种变化性的存在。

在文献[4]中执行路径虽然是在实例化过程中产生的，但是只考虑了服务实例所在资源的 Cpu 负载，对资源的网络等情况并未多做考虑。在文献[5]中介绍了在服务组合处理过程中引入一个概率模型来解决环境的不确定所带来的影响并考虑了用户服务组合的截止时间要求。他利用了关键路径的思想，在最初操作时利用概率模型选择在用户成本范围内最低的服务成本组合。因而也是先静态确定好路径，只是在执行的过程中，当资源发生意外超过底限时，比如资源崩溃等，才重新调用概率模型再动态选择合适的资源。但在重新选择资源时会出现死循环的现象。

在文中，服务执行路径的产生是根据服务组合方案动态选择服务实例所在资源，并考虑了系统和网络负载，使整个系统负载均衡。图 1 表示一个服务组合方案，每个节点 S_i 代表一个功能和接口相同的服务集合。选择一条服务组合的路径就是从每个服务集合中选择一个合适的服务实例，使得这个服务组合路径最优。根据集合中各个服务实例在执行过程中资源的状态来进行服务实例的选择。服务方案上每个节点，比如图 1 中的 S_2 实际上是 2 个功能和接口相同的服务。

2 动态服务组合模型

网格的目的是协调各类动态资源，其性能受到各种不确定性因素的影响，比如系统瞬时的负载、网络带宽等。服务组合方案的产生并不是文中的研究重点，将在以后的工作中进行详细阐述，此处给出一些相关概念。

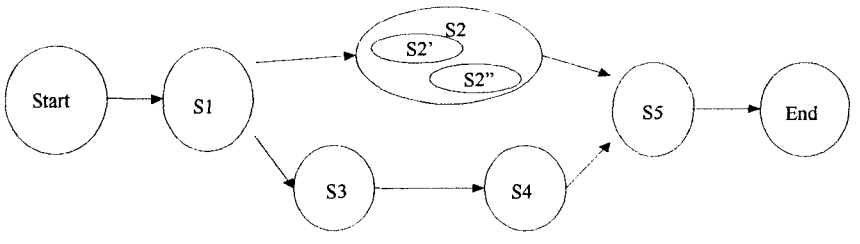

图 1 服务组合—工作流模型

服务组合方案 - 在图 1 中显示了一个根据用户请求自动组合的服务方案。S_1，…，S_5 代表了一组具有相同功能的服务集合。图中箭头表示了任务间的依赖性，比如 S_5 的执行需在 S_2 和 S_4 结束之后，并以它们的输出作为输入参数。

服务执行路径 - 是在服务组合实例化过程中，实例按照相应的路径中的顺序排列的有序集合。

如何根据服务组合方案选择满足用户要求和系统性能需要的资源是文中的研究重点。

3 动态资源选择

在动态环境下选择资源需要考虑系统实时的情况以及在资源上执行过的历史情况，因此，对于资源选择采取了类似竞价机制，从以下三方面考虑：资源的可靠性信息、资源的历史执行信息、资源的实时信息。

3.1 资源的可靠性信息

资源的可靠性(Re)是指用来从满足要求(即该资源上有所需的服务)的资源集中选择合适资源的前提条件，只有当资源的可靠性达到一定标准时该资源才会有可能成为候选资源，表示为在该资源上能成功运行某服务的几率。因此对于用户而言，Re 体现了资源安全可靠执行服务的概率。为此，采用如下公式计算 Re。

$$\mathrm{Re}(X) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{succ}_i^p \tag{1}$$

$$\mathrm{succ}_i^p = \begin{cases} 1 & \text{服务实例 } p \text{ 成功运行} \\ 0 & \text{服务实例 } p \text{ 运行失败} \end{cases} \tag{2}$$

公式(1)中，succ_i^p 表示了在指该资源上运行服务 p 是否成功。因此，$\mathrm{Re}(X)$ 体现了资源 X 历史执行的状态反映了该该资源的可靠性。$R_{\min}$ 是资源可靠性的最低标准，由系统指定，只有当资源的 $R_e \geqslant R_{\min}$ 时，该资源才有可能成为候选资源。值得注意的是 $R_{\min}$ 的设置不能过高，否则可能找不到所需的资源。因此，可以让 $R_{\min}$ 自适应地根据目前候选资源状况动态变化。

3.2 资源的历史执行信息

利用历史执行某服务实例的信息根据公式(3)得出当前该实例的估计执行时间。其中 T_{est}^i 是指第 i 次调用该资源上服务实例时估计执行时间；α_{i-1} 是指在历史情况中每次实际执行时间与估计时间之间的平均误差；β_i 是标准方差，反映了在该资源上执行服务实例的历史变动情况。由此看出，$(T_{\mathrm{est}}^{i-1} + \alpha_{i-1})$ 表示了该算法以前估计时间的情况对当前时间估计的影响，而 $(T_{\mathrm{exc}}^{i-1} + \beta_{i-1})$ 则体现了历史执行情况对当前时间估计的影响。其中 γ 是一个经验值，用于控制这两部分对目前时间估计的影响力。

$$(T_{\mathrm{est}}^i = \gamma(T_{\mathrm{est}}^{i-1} + \alpha_{i-1}) + (1 - \gamma)(T_{\mathrm{exc}}^{i-1} + \beta_{i-1})$$

(3)

$$\alpha_i = \frac{1}{n}\sum \mid T_{exc}^{i-1} - T_{est}^{i-1} \mid \tag{4}$$

$$\beta_n = \sqrt{\frac{1}{n-1}\left(\sum T_{exc}^2 - \frac{(\sum T_{exc})^2}{n}\right)} \tag{5}$$

3.3 资源的实时信息

由于网格的动态性,历史的信息只能作为参考。因此引入对实时信息的考虑,可以较好地将历史和当前信息有机地结合。

由于 Cpu 和网络的状态对于在该资源上执行任务的效率有着至关重要的影响,因此文中选择了 Cpu 和网络的信息来反映资源的实时性能。但在实际情况中经常会出现下列情况(当前节点是 X,资源 Y 和 Z 上都有满足要求的服务):

资源 Y:Cpu 负载是 0.6,而 $Net_{load(x,y)}$ 为 0.2

资源 Z:Cpu 负载是 0.3,而 $Net_{load(x,z)}$ 为 0.3

这时,该如何选择下个执行节点,哪个资源节点当前总体性能更好?为此采用公式(6)来体现资源目前的性能。

$$M_p = \rho(1 - Cpu_{load}) + (1-\rho)(1 - Net_{load(a,b)})$$
$$0 \leqslant \rho \leqslant 1 \tag{6}$$

其中 Cpu_{load} 表示 Cpu 的负载;$Net_{load(a,b)}$ 则代表了该网路的负载。ρ 代表了权重,是个经验值(在这里选择 $\rho = 0.6$)。可以看到,M_p 主要考虑了系统负载均衡,在选择资源时选择总体负载较轻的资源。这时再看一下刚刚的问题,对于资源 Y,它的整体性能 $M_p(Y) = 0.56$;对于资源 Z,$M_p(Z) = 0.7$。因此,认为资源 Z 要比资源 Y 性能更好点。可以看到由于考虑了 Cpu 和网络的负载,有利于均衡整个网格系统的负载。

3.4 资源竞价评估

采用公式(7)来表示资源的出价。Eval 体现了资源当前总体的执行性能。

$$Eval = \frac{M_p}{T_{est}} \tag{7}$$

3.5 动态服务执行路径的实现

在执行服务组合方案时,比如 $S_1 \rightarrow S_2$,当 S_1 的某个服务实例正在执行时,可以调用一个线程用于选择提供 S_2 功能的资源。判断资源的 Eval,其中 Eval 最大的即为下个执行节点。因此整个服务执行路径都是动态,也正因此,服务失败的可能性也大大降低。

4 资源信息收集与发布的实现

资源节点如何存储历史信息以及如何让调度节点获得这些资源的信息?文中采用了 GT4 提供的 factory/instance 模式来提供资源历史信息的收集。

4.1 GT4 的工厂模式

GT4 的工厂模式包括工厂服务、实例服务、资源和资源中心。图 2 表明了它们之间的关系。工厂服务可以在资源中心进行资源的创建和初始化工作。实例服务主要提供服务功能实体,通过它可以访问网格资源。GT4 通过工厂服务为每个用户请求创建一个资源(这里的资源是指状态信息,而 GT3 是为每个用户请求创建一个服务实例,维护资源比维护实例花费要小很多),这个资源有着唯一的标识,称为 EPR(End-PointReferences, EPR = 服务实例地址 + 资源编号)。EPR 能够提供持久、稳定的 Web 服务资源实体,能够允许同一状态随着时间的推移被重复访问。

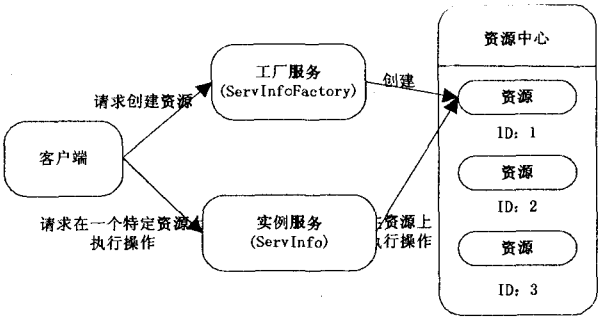

图 2　GT4 工厂模式

4.2 采用 GT4 工厂模式来收集资源的历史信息

在每个资源节点上建立一个资源服务信息收集的服务 ServInfo 以及与之相关的服务工厂 ServInfoFactory。用户通过工厂创建一个资源,然后通过 ServInfo 来设置资源属性,使资源指向一个确切的实体,比如一个文件,以提供该资源对外提供的所有服务实例的历史执行信息。

为了区分每个服务实例所对应的资源以便访问对应服务的历史信息,可以将服务对应资源的端点引用写入文件中,保存起来以便日后的设置和访问,这就要保证文件名的唯一性。为了方便起见,采用某服务实例的地址来命名,当然要去除一些非法字符。比如要查看该资源上的 MyPaper 服务历史执行信息,则可以调用文件名为 MyPaperEpr. txt 文件来查询(每个资源节点上发布服务的文件名是唯一的)。

下面具体介绍 ServInfo 服务,它属于公共服务。用户可通过 EPR 使用 ServInfo 在指定资源上执行操作,这个资源包括了该资源节点上某个服务实例的历史信息。

其资源属性(RP)包括:

RP1:名称(LastEstTime);类型(xsd:string);用来设置和访问上次 T_{est}^{i-1}。

RP2:名称(TErr);类型(xsd:string);用来设置和访问 $\sum \mid T_{exc}^{i-1} - T_{est}^{i-1} \mid$,在这里并没有直接存储 α_i,主

要是为了计算的方便。

RP3:名称(LastExcTime);类型(xsd:string);用来设置和访问 T_{exc}^{i-1}。

RP4:名称(SumSquareExc):类型(xsd:string);用来设置和访问 $\sum T_{exc}^{2}$。

RP5:名称(SumExc):类型(xsd:string);用来设置和访问 $\sum T_{exc}$。

RP6:名称(Num):类型(xsd:string);用来设置和访问该实例的执行次数 n。

PR7:名称(SuccNum):类型(xsd:string);用来表示该实例成功完成的次数。

服务 ServInfo 的接口描述(见表 1):GetTest 接口获得该服务实例的估计执行时间并且将属性 RP6 加 1。GetTestResponse 返回估计的执行时间(公式(3))。SetTestTexc 的接口用于在实例执行完后设置其各个属性并将 RP7 加 1。ExecTime 表示了实际执行时间,EstTime 表示了估计执行时间。OperationResponse 是该服务的通用响应接口。

表 1 ServInfo 接口

接口名称	参数名	参数类型
GetTest	—	—
GetTestResponse	EstTime	Xsd:string
SetTexcTest	ExecTime	Xsd:string
	EstTime	Xsd:string
OperationResponse	—	—

因此,获得资源上某个服务的历史信息的过程如下:

首先,调用运行节点 Container 中的 ServInfoFactory 工厂服务创建一资源并获得其 EPR,对该资源初始化并将 EPR 写入文件。

其次,用服务名字构成该服务资源的 EPR 文件名,通过该 EPR 文件获得该服务当前的估计执行时间并设置使用该服务的次数。

最后,每次服务执行完后都会再次通过 EPR 文件设置最新一次的服务执行信息。

4.3 获得资源竞价

通过 3.2 节可获得公式(7)中的 T_{est},至于 M_p 则是可以通过 Mds 和 NWS[6]获得当前 Cpu 和网络状态的信息。因此可以得到该资源的竞价 Eval。

5 结束语

服务网格环境中的动态性和分布性,使得对于服务组合如何选择合适的资源是个极大的挑战。因为网格环境的动态性,静态选择服务实例的组合路径并不能保证该路径始终是最佳的,同时,也可能因为组合路径上服务实例所在资源的关闭等原因造成不断地重新启动服务实例或更换资源。因此,笔者提出了当存在多个资源节点都有相同服务时,根据服务的历史执行信息选择最安全、性能最好的资源来作为执行节点,并考虑了整个网格系统的均衡负载。最后给出了实现方式。下一步工作重点是考虑资源在执行服务的过程中出现异常时如何处理。

参考文献:

[1] Foster I, Kesselman C, Nick J, et al. Grid services for distributed system integration[J]. IEEE Computer, 2002, 35(6):37-46.

[2] 谷清范,吴介一,张飒兵,等. 基于遗传算法的多性能目标网格服务调度算法[J]. 信息与控制,2005(3):279-285.

[3] 张晓杰,孟庆春,曲卫芬. 基于蚁群优化算法的服务网格的作业调度[J]. 计算机工程,2006(8):216-218.

[4] Cheung W K, Liu Jiming, Tsang K H, et al. Dynamic resource selection for service composition in the grid[C]// IEEE/WIC/ACM International Conference on Web Intelligence(WI'04). [s.l.]:[s.n.], 2004:412-418.

[5] Sample N, Keyani P, Wiederhold G. Scheduling under uncertainty: planning for the ubiquitous grid[C]// Proceedings of the 5th International Conference on Coordination Models and Languages(Coord2002). [s.l.]:[s.n.],2002:300-316.

[6] Wolski R. Dynamically forecasting network performance using the network weather service[J]. Cluster Computing, 1998, 1(1):119-132.

(上接第 220 页)

二次开发提供了较好的条件。本系统也为其它钥匙形式实现一机多门的联网系统设计上位机网控软件提供了经验和借鉴内容。

参考文献:

[1] 戴 永. 基于用户可自画图形的图像特征信息卡研究[J]. 中国图像图形学报:A 辑,2003,8(10):1183-1188.

[2] 戴 永. 匙膜图像识别的"门一锁一匙"系统[J]. 湘潭大学自然科学学报,2003,25(1):17-20.

[3] 王求真. 智能像卡联网门禁系统信息采集研究[J]. 湖南科技学院学报,2006,36(5):107-110.

[4] Schmuller J. UML 基础、案例与应用[M]. 李 虎,赵龙刚,译. 北京:人民邮电出版社,2004.

[5] Rumbaugh J, Jacobson I, Booch G. The unified modeling language reference manual[M]. 北京:科学出版社, 2004.