文章编号:1001 - 9081(2007)10 - 2595 - 03

迁移工作流系统中位置服务体系结构的研究与设计

秦宇锋,曾广周
(山东大学 计算机科学与技术学院,济南 250061)
(qinyf@ mail. sdu. edu. cn)

摘 要:将位置对迁移实例(MI)的服务区分为迁移实例服务和工作流服务,基于多线程设计了一种支持两类服务的体系结构,不仅有效地提高了位置服务的效率,同时增强了位置服务的可扩展性。

关键词:迁移工作流;迁移实例;位置服务;体系结构;多线程;可扩展性
中图分类号: TP302 **文献标志码**:A

Research and design of location service architecture in migrating workflow system

QIN Yu-feng, ZENG Guang-zhou
(*School of Computer Science & Technology, Shandong University, Jinan Shandong* 250061, *China*)

Abstract: The services to Migrating Instance (MI) that a location can be offered were classified into MI services and workflow services at first, and then a service architecture based on multi-thread to support these two kinds of services was proposed in this paper. It not only helps to increase the services efficiency of the location server provided, but also makes it more extensible.

Key words: migrating workflow; Migrating Instance (MI); location service; architecture; multi-thread; extensibility

0 引言

迁移工作流是近年来工作流管理研究的一个新方向,并且被解释为运行期间在工作位置上合并静态工作流说明、本地规则和策略、以及用户策略的效应[1]。

文献[2]提出了一个迁移工作流管理系统框架,其中业务过程的执行体称作迁移实例(Migrating Instance,MI),它可以携带任务说明书和当前执行结果在合适的工作位置之间连续迁移和执行,多个迁移实例可以通过迁移和协作共同完成一个跨机构跨地域的复杂工作流程;工作位置代表参与业务过程的企业或机构,并且被映射为一个由停靠站服务器(称作 MI Server)及若干工作站点组成的局域网络,约定停靠站是迁移实例的运行场所,工作站点为迁移实例提供具体的业务过程服务;迁移实例由迁移工作流管理引擎创建,因此,迁移工作流管理引擎是一个工作流组织者的映射。除创建迁移实例外,迁移工作流管理引擎还要负责工作位置的组织和管理,以及对 MI 的监控、工作流故障恢复等。

文献[3]给出了一个结合微内核技术和组件技术的 MI Server 模型。当 MI 到达时,MI Server 便为它分配一个线程,创建迁移实例环境,并通过服务代理(Service Agent)响应 MI 的服务请求;在 MI 迁出后,为其分配的线程被释放。文献[3]模型在应用中存在下述问题:

1)没有区分服务的类型,因而不能有效利用服务的差异进行设计,导致线程的整体生命周期过长。当有大量的长事务 MI 同时到达一个 MI Server 时,会造成线程资源紧张,影响系统的效率和稳定性。

2)没有考虑 MI Server 上的服务排队和调度机制,因而不支持服务优先级。

3)没有对停靠站服务器和工作流服务器进行综合设计,因而不支持服务系统的整体优化。

针对文献[3]模型存在的问题,本文设计了一种新的位置服务体系结构,不仅对服务类型进行了区分,而且综合考虑了 MI Server 和工作流服务器的一体化设计,所有实现均基于 Java 语言。

1 位置服务的分类

定义 1 迁移工作流中的服务(Migrating Workflow Service,MWS)是一个五元组(id,MI,MI Server,Spec,C),其中,id 是服务标识,MI 是服务消费者,MI Server 是服务提供者,Spec 是服务说明,C 是服务契约。

服务可以按照服务类型和 MI 是否显式地调用服务进行分类,并且通过服务契约进行约定。

根据服务类型,MI Server 提供两种类型的服务:一类是相对固定的、不针对某类 MI 的通用服务,如迁入/迁出服务、请求解析服务、通信服务、安全服务、日志服务等,本文称之为迁移实例服务。另一类是相对易变的、需要查询才能得知的、针对某个具体业务应用的服务,本文称之为工作流服务。

根据 MI 是否显式地调用服务,可以把服务分为非透明服务和透明服务:前者存在于服务列表中,需要 MI 显式地提出服务请求,例如 迁出/迁入服务、通信服务、工作流服务等;后者是 MI Server 在 MI 不察觉情况下提供的服务,MI 无法直接调用,例如安全认证服务、日志服务等。

定义 2 服务说明 Spec 是一个二元组 (F,R),其中,F 是服务功能的集合,R 是定义在 F 上的一个序关系,包括顺序、

收稿日期:2007 - 05 - 23;**修回日期**:2007 - 07 - 05。
基金项目:国家自然科学基金资助项目(60573169);山东省科学技术发展计划项目(031110123)。
作者简介:秦宇锋(1983 -),男,山东济南人,硕士研究生,主要研究方向:智能计算与协同技术; 曾广周(1947 -),男,山东郓城人,教授,博士生导师,主要研究方向:CSCW、智能计算理论与技术、移动计算及其应用技术。

与分叉、或分叉、与合并、或合并等。

定义 3 给定 $f_i \in F$、$f_j \in F$ 和 $f_k \in F$。如果只有当 f_i 和 f_j 都完成后，f_k 才能执行，则称 f_i 和 f_j 为同步结构，f_k 是同步点。

当采用多线程设计系统时，需要通过控制使 f_i 和 f_j 同步。

定义 4 给定 $f_i \in F$ 和 $f_j \in F$。如果 f_i 和 f_j 的运行都不依赖对方的状态，则称它们为异步结构。

当采用多线程设计系统时，异步结构中的功能部件各自占有独立的线程。

2 MI Server 的体系结构

MI Server 的体系结构如图 1 所示，主要包括：迁移实例环境 MAE、迁移实例服务组件、工作流服务组件和微内核等。

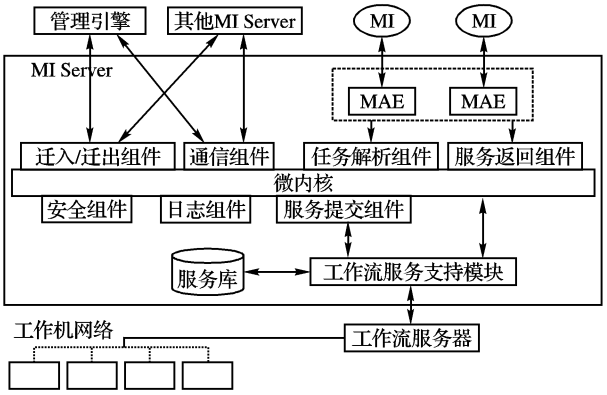

图 1 MI Server 的体系结构

2.1 迁移实例环境 MAE

MAE(Mobile Agent Environment)是 MI 访问 MI Server 的入口点。通过 MAE，迁移实例能够获取当前 MI Server 提供的服务，以及当前驻留在本 MI Server 上的其他迁移实例的信息。每个 MI 关联一个 MAE，它在 MI 通过安全认证并且被激活时创建，在 MI 迁出并得到确认后撤销。

MAE 按照服务请求类型为 MI 提供服务。如果 MI 请求非透明迁移实例服务，则 MAE 返回一个功能组件的上下文对象的引用；如果 MI 请求工作流服务，则 MAE 先将任务说明传递给任务解析组件，任务解析组件完成解析并按照一个序关系将子任务传递给服务提交组件，再对服务请求封装并传递到工作流服务支持模块。工作流服务完成后，由服务返回组件将结果返回至 MAE。

当 MI 确认所有的服务请求都已经执行完毕，并且当前位置不能满足其继续执行任务的需求时，它可以向当前 MI Server 提出迁出请求。当前 MI Server 收到迁出请求后，执行迁出服务帮助 MI 到达下一个适合工作的位置。

2.2 迁移实例服务组件

迁移实例服务组件主要包括迁入/迁出组件、通信组件、安全组件、日志组件等。它们提供了同步结构的非透明迁移实例服务。

迁入/迁出组件负责 MI 的迁移，当 MI 提出迁移申请后，当前位置上的迁出组件先与目的位置上的迁入组件协商迁移事件，成功后再启动 MI 的迁移。

通信组件负责 MI 与异地协作 MI、MI 与管理引擎、MI Server 与管理引擎、MI Server 之间的远程通信。

安全组件包括通信加密、MI 迁移加密、MI Server 对 MI 认证等功能。

日志组件记录与 MI Server 相关的各种信息。

2.3 工作流服务组件

工作流服务组件由服务库、任务解析组件、服务提交组件、服务返回组件和工作流服务支持模块 WSSM(Workflow Service Supporting Module)组成。它们提供了异步结构的非透明工作流服务。

服务库存储 MI Server 能够提供的所有工作流服务项的信息，由 WSSM 管理和维护。服务项是一个三元组{服务名，服务缓存器名，服务线程名}，其中服务缓存器和服务线程的作用见图 2。服务库也提供外部查询功能。例如迁移工作流管理引擎的监控查询、其他位置上的 MI 迁移决策查询等。

面向 MI 的工作流服务包括如下步骤：

1)任务解析组件解析 MI 提交的任务说明，以确定 MI 在本地“做什么”。再根据解析得到的子任务的依赖关系与优先级向服务提交组件提出服务请求。

2)服务提交组件把服务请求封装并传递给 WSSM，便返回。服务请求是一个封装了{MI 的 ID，服务名称，服务参数，优先级，等待标志}的对象，称作请求对象。

3)服务返回组件在 WSSM 处侦测并提取返回结果至相应的 MI。返回结果是一个封装了{MI 的 ID，服务名称，结果}的对象，称作结果对象。

WSSM 是一类特殊的功能组件，它利用线程接收并处理 MI 提交的工作流服务请求。

2.4 微内核

微内核负责加载、管理和维护 MI Server 中的所有功能组件，并为功能调度和数据传递提供总线[3]。MI Server 启动时，微内核读取系统配置文件，并根据配置文件装载、初始化和启动迁移实例服务组件和工作流服务组件。在运行期间，当有 MI 迁入时，微内核首先驱动安全组件对其进行安全认证，然后对合法的 MI 创建运行环境 MAE，并通过 MAE 管理控制 MI 的生命周期。

3 工作流服务支持模块(WSSM)的设计

WSSM 由对象缓存器、请求调度器、服务缓存器组、线程池、动态调节器、服务加载器和连接池组成，其结构如图 2 所示。

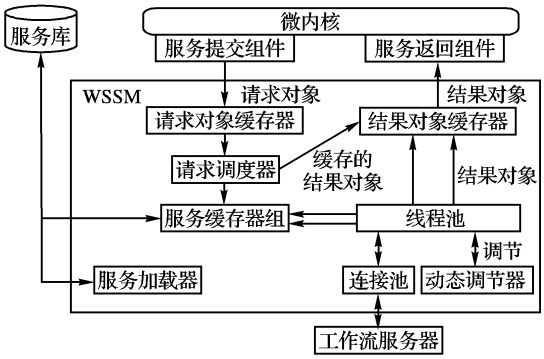

图 2 WSSM 的结构

3.1 对象缓存器

对象缓存器包括请求对象缓存器和结果对象缓存器两种。服务提交组件把请求对象放入请求对象缓存器，请求调度器把请求对象按优先级从缓存器取出并传递给服务线程。服务线程池中的服务线程把来自工作流服务器的结果对象放入结果对象缓存器，服务返回模块提取缓存区中的内容并传递给相应的 MI。

对象缓存器是一类临界资源。类似于“生产者－消费者”问题，对象生产者在缓存区满时必须等待，直到缓存区有

空间才继续存入。对象消费者在缓存区空时必须等待,直到缓存区中有对象时才能继续读取。

本文使用 Java 中的线程通信机制来同步线程对缓存区的访问,避免了多线程之间无序情况的发生。

3.2　请求调度与服务实现

请求调度器是 WSSM 的核心控制部件。它首先从请求对象缓存器中根据优先级读取请求对象,然后根据服务请求对象中的服务名称查询服务库,得到与之对应的服务缓存器名和服务线程名,再把服务请求对象存入对应的服务缓存器并激活线程池中的对应服务线程。

服务加载器监视并维护着一个目录,称之为服务目录,它映射着本地的一个文件夹。服务目录中存放若干预先设计好的 jar 文件,每个 jar 文件包括实现了 runnable 接口的 Thread 类、以 xml 语言描述的服务名称和服务缓存区名称文件。一个 jar 文件就是一个服务的映射,它对应于服务库中的一条记录。当服务目录发生变动时,服务加载器负责更新服务库。

服务缓存器组由多个服务缓存器组成,每种服务对应一个服务缓存器。服务缓存区也是一个临界资源,同种服务的不同线程需要同步地访问它。

通过请求调度算法,请求调度器最大限度地使多个服务请求同时得到处理。请求调度算法流程描述如图 3。

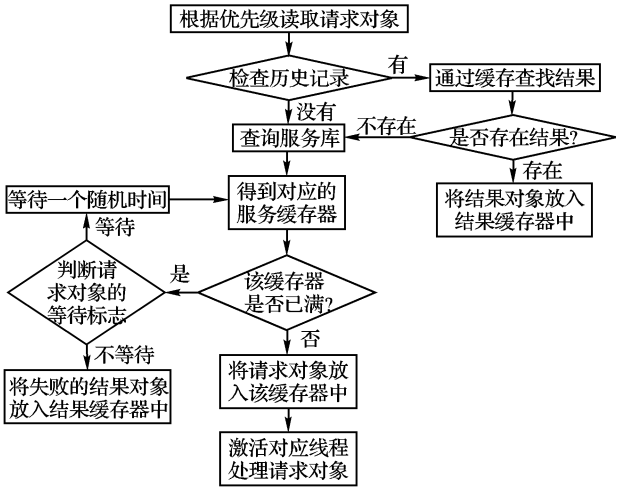

图 3　请求调度算法

3.3　线程池及其动态调节

线程池是 WSSM 的核心计算部件。系统初始化时,在线程池中预先准备好若干数量的“热”线程。当应用程序需要使用线程时,从线程池中取用一个空闲线程(如果线程池为空则等待)并启动相应的服务,服务结束后,将线程使用权归还给线程池,以备复用,这样可以避免对每一个请求都生成和删除线程的昂贵操作。

本文采用纽约奥斯威戈州立大学 Doug Lea 编写的并发实用程序开放源码库 util. concurrent 中的线程池实现[4]。

动态调节器负责监控线程池的运行强度,对池的最小热线程数进行调整,使线程池的响应速度更快,占用系统资源更少。算法的基本步骤为:

1)为线程池建立一个大小为 N 的状态队列。

2)对线程池进行一次扫描,若线程池为满载,则将状态队列尾标志为 F;若线程池为空载,则将状态队列尾标志为 E。

3)扫描线程池的状态队列,若状态队列中元素全为 F,则将对应的线程池的最小线程数 M 增加 P%;若状态队列中元素全为 E,则将对应的线程池的最小线程数减少 P%。

4)每隔时间 T 秒重复 2),3)。

P 值(0 ~ 100)过大,则可能会使线程池中的线程增大或减少得过多,这就会使线程池重复不断地由大到小,再由小到大地调节直到合适为止,造成运行效率的低下和系统资源的浪费。而 P 值过小则会对线程池的影响过小,起不到调节的作用。P 的选取在 10 ~ 20 之间为宜。

T 值代表时间间隔,当 T 过大时,则会使得调节显得过于“迟钝”;而当 T 过小时,又使得它只反映一小段时间的负载,不能以“大局”为重;同时,T 的选取也要与 N 值的大小成反比。当 N 确定时,若希望线程池至少 10 s 会自动调节一次,则 T 值为 N/10 s。

3.4　连接池

连接池是预先创建好的若干业务过程实例。设置连接池的目的是为了简化服务线程与工作流服务器之间的连接,避免频繁地创建、初始化和清除实例操作,缩短响应时间,服务线程甚至可以无需知道连接后面存在着什么样的业务过程服务。

4　业务过程组件的设计

按照文献[2]提出的迁移工作流系统框架,工作流服务器及工作机网络是为 MI 执行业务过程的主体,并且独立于 MI Server(参见图 1)。本文将开源的 JBoss[5] 作为工作流服务器,并遵守 J2EE 构架标准开发业务过程组件,其系统结构如图 4 所示。

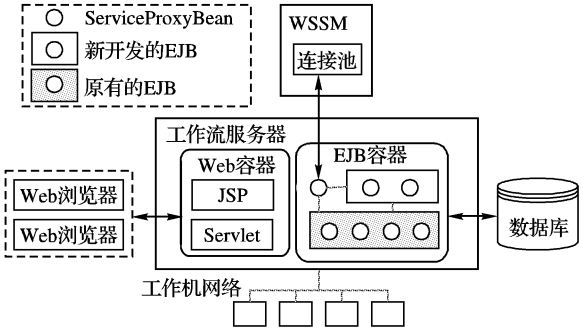

图 4　工作流服务大端业务组件的系统结构

参考 Session Facade 设计模式,并在一个企业级应用的基础上(如果存在的话),本文开发了一个称作 ServiceProxyBean 的无状态 Session Bean,为 MI Server 提供统一的调用界面,屏蔽了业务逻辑层[5] 的复杂性,并开发了若干 Entity Bean 和 Message Driven Bean, 供 ServiceProxyBean 调用。ServiceProxyBean 是底层子系统的客户端接口,它把特定于子系统的信息封装起来,并且不能在不必要的情况下公开它。

WSSM 中的连接池就是预先建立了若干 ServiceProxyBean 的客户端,线程池中的多个服务线程可以分别取得 ServiceProxyBean 客户端的一个实例。因为 ServiceProxyBean 使用了无状态 Session Bean,会话状态只会在单个方法调用中被保持,一旦方法调用结束,组件将丢弃方法调用过程中保持的状态,不保持跨越方法调用的会话状态。所以对于任何客户端调用,ServiceProxyBean 实例之间没有区别。对于后续的其他客户端方法调用,前一个调用也没有影响。

也可以在 Web 容器中开发一些 Web 界面,图形化地显示和操作 ServiceProxyBean 和其他 EJB 组件,不再赘述。

(下转第 2601 页)

具有近似线性的时间复杂度。本文 TPOM 算法和传统的 KNN 算法都采用 Java 实现,实验平台为 P4 1.8 GHz,512M 内存的 PC,TPOM 算法和 KNN 算法在离群点数目 $m = 10,20$; 近邻数 $k = 20,50,100$ 时的效率对比如图 1 所示。

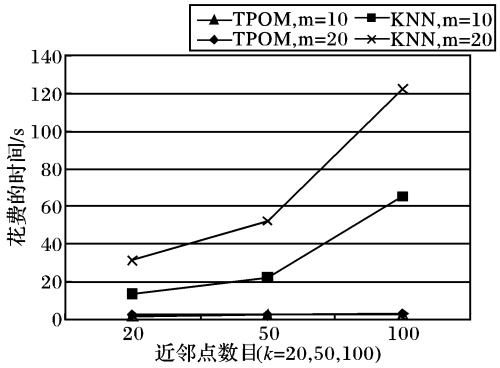

图 1　TPOM 和 KNN 的效率对比

实际应用研究发现,TPOM 算法在聚类的基础上,具有很好的可扩展性。一方面,算法可以充分利用先进的聚类技术,提高算法第一阶段的聚类效率;另一方面,采用距离的相似性度量方法,可以有效地度量高维数据之间的相似度,同时可以迅速地将数值型变量扩展到名称、顺序、比率以及混合型变量。正常情况下,TPOM 算法具有近似线性的时间复杂度,算法的时间复杂度随维数增多呈线性增长。但算法的效率有三个方面的影响因素,首先,数据集不应该是有序的,因为算法无法在有序数据集中实现近似最近邻的快速查询和正常点的有效剪枝;其次,数据必须是相对独立的,若数据相互依赖且具有大致一样的数值,则很可能处于同一个类别,近似最近邻查询需要扫描大部分数据才能达到目的;最后,算法不适于挖掘不包含离群的数据集,因为都是正常的数据无法实现正常点的有效剪枝。

4　结语

本文对基于距离的循环嵌套 KNN 算法进行了分析与改进,提出了二阶段近似最近邻离群挖掘算法,实现了分两个阶段完成的离群挖掘。第一阶段采用变形的 k-均值聚类方法对数据集进行大致聚类;第二阶段在聚类的基础上,采用基于循环嵌套的近似 k 最近邻算法来挖掘离群。由于聚类后,同一个类别中的数据的相似度较大,对于数据集中占大多数的正常数据,近似 k 最近邻查询只需对所在的类进行搜索而不须搜索整个数据集,因此,算法具有近似线性的时间复杂度。TPOM 算法应用在福建省海洋环境质量监测数据的分析与处理中,结果表明算法具有较好的适用性。进一步的工作主要包括相似性度量的改进、最近邻查询和剪枝效率的提高以及多源数据的离群检测。

参考文献:

[1] HODGE V, AUSTIN J. A survey of outlier detection methodologies [J]. Artificial Intelligence Review, 2004, 22(2): 85 - 126.

[2] 黄洪宇, 林甲祥, 陈崇成, 等. 离群数据挖掘综述[J]. 计算机应用研究, 2006, 23(8): 8 - 13.

[3] KNORR E, NG R. Algorithms for Mining Distance - Based Outliers in Large Datasets[A]// Proceedings of the 24th Int'l Conference on Very Large Databases. New York: ACM Press, 1998: 392 - 403.

[4] KNORR E M, NG R T. Finding intensional knowledge of distance-based outliers[C]// Proceedings of the 25th International Conference on Very Large Data Bases table of contents. San Francisco: Morgan Kaufmann Publishers Inc, 1999: 211 - 222.

[5] BARNETT V, LEWIS T. Outliers in Statistical Data[M]. 3rd ed. New York: John Wiley, 1994.

[6] KNORR E M, NG R T, TUCAKOV V. Distance-based outliers: algorithms and applications [J]. The VLDB Journal, 2000, 8(3): 237 - 253.

[7] BOLTON R J, HAND D J. Statistical fraud detection: A review[J]. Statistical Science, 2002, 17(3): 235 - 255.

[8] RAMASWAMY S, RASTOGI R, SHIM K. Efficient algorithms for mining outliers from large data sets[C]// Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2000: 427 - 438.

[9] KIM T-W, LI K-J. A distance - based packing method for high dimensional data[C]// Proceedings of the Fourteenth Australasian database conference on Database technologies. Adelaide, Australia: [s. n], 2003, 135 - 144.

[10] BAY S D, SCHWABACHER M. Mining Distance - Based Outliers in Near Linear Time with Randomization and a Simple[C]// Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington: ACM Press, 2003: 29 - 38.

[11] JIANG M F, TSENG S S, SU C M. Two-phase clustering process for outliers detection[J]. Pattern Recognition Letters, 2001, 22(6 - 7): 691 - 700.

[12] HAN J W, KAMBER M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 译. 北京: 机械工业出版社, 2001.

(上接第 2597 页)

5　结语

本文针对迁移实例服务和工作流服务的不同特点,设计了一种包括 MI Server 和工作流服务器的位置服务体系结构,其主要优点有:

1)对不同类型的服务采取了不同的处理方式,以达到功能和性能的最大优化。

2)多线程的工作流服务能同时处理多个 MI 的请求,提高了系统的并发度。

3)系统更加灵活,可以通过增加新线程实现工作流服务的动态扩展。

参考文献:

[1] CICHOCKI A, RUSINKIEWICZ M. Migrating Workflows [C]// Workflow Management Systems and Interoperability. Berlin, Heidelberg: Springer-Verlag, 1998: 339 - 355.

[2] 曾广周, 党妍. 基于移动计算范型的迁移工作流研究[J]. 计算机学报, 2003, 26(10): 1343 - 1349.

[3] 谢浩, 王晓琳, 曾广周. 面向服务的柔性迁移工作流停靠站设计[J]. 计算机应用, 2006, 12(3): 685 - 687.

[4] 纽约州立大学奥斯威戈分校网. Overview of package util.concurrent Release 1.3.4[EB/OL]. [2007 - 05 - 01]. http:// www.oswego.edu.

[5] 罗时飞. JBoss 管理与开发核心技术[M]. 3 版. 北京: 电子工业出版社, 2004.