基于移动代理的网格计算

陈 佳 吴 跃
(电子科技大学计算机科学与工程学院 成都 610054)

摘 要 网格计算环境具有异构性、分布性、动态性、局域自治性等特点,针对以上特性,利用移动代理技术的网格计算方法,可以提供高效统一的应用接口,实现多编程环境下的协同应用求解,从而有效屏蔽其异构和分布等特性,动态适应其资源的变化。详细说明了基于移动代理的网格计算的体系结构和主要模块的功能,并阐述了移动代理的实现过程。

关键词 移动代理,网格计算,分布性

Grid Computing Based on Mobile Agent

CHEN Jia WU Yue
(Dept. of Computer, University of Electronic and Science Technology of China, Chengdu 610054)

Abstract To fit the features of heterogeneity ,distribution, dynamic and local autonomy in grid computing environments, the grid computing based on mobile agent technology can provide the efficient and uniform application interfaces, which can implement the cooperative application in several parallel programming environments, and mask the heterogeneity and distribution to be suitable for the dynamic varying of resources. In the paper, we detail the architecture of this grid computing and the function of its main modules, and illustrate the implementation of the mobile agent module.

Keywords Mobile agent, Grid computing, Distribution

1 引言

随着科学技术的不断发展,Internet 技术也随之经历着变革。以 Email 为主要应用的第一代 Internet 把遍布于世界各地的计算机用 TCP/IP 协议连接在一起;第二代 Internet 则通过 Web 信息浏览及电子商务应用等信息服务,实现了全球网页的连通;第三代 Internet 将"试图实现互联网上所有资源的全面连通,包括计算资源、存储资源、通信资源、软件资源、信息资源、知识资源等",这就是网格计算(Grid Computing)。网格计算是一种使用网络中的多个计算资源来解决单一问题的计算模式。当一个计算工程需要的处理资源超过本地可提供的能力时,网格计算允许该计算工程通过网络使用远程机器的 CPU 和存储资源等。据抽样调查统计显示,在今天的网络上,75%的 CPU 时间和 80%的服务器资源都是浪费的,利用网格技术就可以将这些网络资源的利用率提高到 98%~99%。

但是,网格计算环境[2,9]具有地理上分布广、计算资源动态性、非独占性、系统异构、隶属于不同组织以及网络通信延迟的不确定性等特点,直接导致了应用程序的复杂和低效,一些结点上资源很小的变化就可能对整个并行应用的性能产生很大影响,难以达到预期的效果,从而给分布式并行计算环境带来了技术困难。网格计算的中间件软件能够很好地解决这些问题,提供高效统一的应用接口,实现多种并行编程环境下的协同复杂应用的求解,充分发挥网格的计算潜力。其中,移动代理 MA(Mobile Agent)技术就是一种能很好适应此应用的中间件软件。使用软件 MA 技术设计网格计算的中间件,它屏蔽结点的异构性、资源的变化性,为用户提供基于 Web 的高性能应用求解的访问接口,目标是为用户提供协同计算服务。目前,Web 方式提供应用提交、编译和运行功能,可自主决定是自动结点选择,还是人工选择方式为应用提供计算资源。结点选择是计算服务的重点,可大大减轻用户负担,无缝地提供一个便捷的计算环境。本文首先介绍了该模型总的体系结构以及各个模块的功能,然后分析了移动代理模块的工作过程,最后对该模型进行了客观的评价并展望了未来的研究工作。

2 网格体系结构

目前,国际上以及国内关于基于移动代理的网格计算模型已逐层推出了,它们大都是针对具体的应用和特定的平台设计的。本文将介绍一种比较通用的模型,它独立于具体的应用平台,是上面提到的针对具体应用的模型的高度抽象和概括。总的来说,逻辑上它从下到上可以分成四层结构,如图 1 所示。从下至上每层可依次命名为:物理层模块、移动代理模块、用户程序包模块、用户程序模块。下面就分别对各模块进行说明。

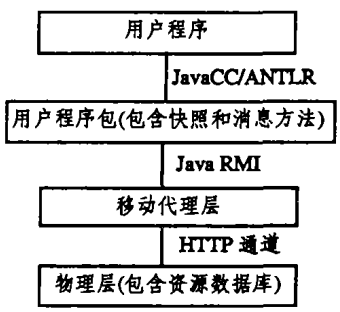

图 1 基于移动代理的网格模型

陈 佳 硕士研究生,主要研究方向:网络计算技术与应用。吴 跃 教授,博士生导师,主要研究方向:网络计算技术与应用以及数据库技术与应用。

2.1 物理层模块

物理层模块是网格系统的硬件基础，包括各种计算资源，如超级计算机、贵重仪器、可视化设备、现有应用软件等，这些计算资源通过网络设备连接起来。该模块类似于五层沙漏结构的构造层[1]，其基本功能就是控制局部的资源，向上提供访问这些资源的接口。它是网格作用域内所有连接到网络上信息包括主机名、IP 地址、体系结构类型、内存大小、主频等的集合体。本模块资源可以是一个比较复杂的系统，例如由多台计算机通过系统级网络连接形成的机群系统，在机群系统的内部，为了实现通信和管理，必须有自身的协议，该协议是内部的，与网格体系结构中资源之间的外部协议是不同的。本层资源提供的功能越丰富，则它可以支持的高级操作系统就越多。例如如果在资源层支持提前预留功能，则很容易在高层实现资源的协同调度服务，否则在高层实现此服务就会产生较大的额外开销。反之，若该层资源提供的功能较少，则网格结构的组织就可以比较简单，实现起来就相对容易一些。形成计算网格环境的过程可以这样描述：首先，初始调制器启动一个因特网组织，该组织在其数据库里维护每一个组成员的相关信息。多个因特网组织可以使用 GMDS(Globus Metacomputing Directory Service)[5]进行组织管理。然后，每个欲接入该网格系统的用户创建一个客户帐号，在 Web 服务器上运行，并在因特网组织上登记注册用户的计算机信息，包括其身份验证信息、计算资源信息、可用性标志以及负载能力等。如果通过服务器的验证后，该用户就顺利接入网格环境了。其他申请加入的用户采用同样的过程，就可以形成一个庞大的网格计算环境。

2.2 移动代理模块

Mobile Agent 技术起源于 20 世纪的分布式人工智能的研究，近年来这一技术逐渐被应用在越来越多的领域中。Mobile Agent 技术强调软件的移动性、分布性、自主性、智能性、协同性和社会性等，通常用来构建大规模的分布式软件系统。软件 Mobile Agent 是模型化的复杂软件系统的高级抽象，展示高度的动态行为[3]，是一种复杂的计算机程序，采取自治的行为，协同应用与环境交互，完成给定的目标。它与传统的 RPC(远程过程调用)相比，具有以下优点：1)减轻了网络负载；2)对网络的延时和抖动不敏感；3)可以封装专有协议，在不兼容系统之间协调和互操作；4)异步操作不需要持续的网络连接；5)容易支持服务的个性化；6)容易适应多变的网络环境。正是因为移动代理技术具有的这些优点很好地适应了网格计算环境下分布式并行计算的任务需求，所以它首当其冲地成为网格计算实现中间件的首选。

该模块是本文要讨论的重点模块，是整个系统模型设计的关键所在。首先，我们应该清楚 Mobile Agent 实质上是一个程序实体，由代码和数据共同组成，具有唯一的名字，具有智能性，可以在网络上的各结点的主机上自由移动，并且在这些主机上执行。在本系统环境下移动代理的目标是集成多种并行编程环境，在异构分布式的网格环境上进行协同计算，屏蔽应用程序的编译、运行的分布性与异构性。当一个应用程序选择一个包含多种体系结构的计算结点子集进行并行计算时，不管是 MPI 还是 PVM 并行环境，程序编译需要在每种体系结构上进行，运行时则在每个结点需要一个与体系结构相对的二进制程序副本，主结点才能驱动该应用在这个异构结点集上并行运行。需要注意的是，在实际的系统设计时，一般是在用户进程和移动代理之间插入一个经纪(Broker)进程，用来负责把用户请求转换成移动代理可以识别的请求，并将移动代理返回的结果解释成用户可见的形式，Broker 和 Mobile Agent 之间遵守既定的通信协议，Broker 在这里起到一个转换器的作用，从而有效提高移动代理的执行效率。

在本系统特定的环境下，其工作过程可以这样描述：对于那些已经通过安全认证的客户端用户，每一个移动代理代表一个客户端用户，携带该用户程序的工作请求，并智能地为该请求寻找相应可用资源，访问因特网组织数据库为该请求搜索最适合的计算机，并在其上运行之。在运行过程中，如果因为某种原因，例如任务所需资源在本结点不存在或该结点因为某些故障掉电等，使得这些计算机不可用时，Mobile Agent 将不得不从网格环境中的该结点移植到另外可用结点。此时，Mobile Agent 将被系统挂起，转入到冻结状态，并序列化为数据流[4]，然后将这些数据流封装到一个 HTTP 消息包中通过 HTTP 通道发送到目的结点，该结点上集成了一个 HTTP 服务器，此服务器预留了一页特定的 Web 页面，在这里 Servlet 将对接收到的封装了 Mobile Agent 的 HTTP 消息包进行析构和解释操作，也即是将数据流再反序列化，生成相应数据，从而重构该代理，代理从冻结状态恢复为运行状态，恢复用户进程的执行。

2.3 用户程序包模块

用户程序包实质上是一个 Java 对象，它可以通过 Mobile Agent 在远程计算机上被实例化。它主要用来监控与其相对应的用户程序的执行状态(即在程序包中加入执行快照)以及从用户进程到它的所有基于 TCP/IP 协议标准的消息通道(即在程序包中加入消息方法)。该程序包能够传送这些消息到 Mobile Agent 上，该移动代理能够把消息传送到参与同一执行任务的计算机集合中某些适合任务执行的计算机上。基于此应用目的，用户程序包必须提供状态捕获以及传递消息的方法，此方法能够在用户程序里被调用。在任务进程迁移过程中，用户程序包将负载到 Mobile Agent 上，随着 Mobile Agent 的移动而移动，当 Mobile Agent 到达目标结点后，用户程序包将根据任务进程的相应信息重构用户程序，在目标结点运行之。这就是用户进程的移植过程。需要指出的是，为了先前工作的恢复，用户程序包中的所有快照必须保存在一个固有的存储器里；为了避免磁盘溢出或者任何失败终止事件的发生，快照必须在多个客户端或组织服务器中维护。

2.4 用户程序模块

用户程序不是移动代理本身，不包含任何像 go() 或者 hop()这样的移动所需的初始化状态。因此，在基于 Mobile Agent 的网格计算的用户程序的连续计算块中必须插入诸如 For/While 的循环来表示相应状态。要达到插入状态标识的目的，需要加入预编译工具：ANTLR 和 JavaCC，前者用于 C/C++，后者用于 Java 的预处理。对于用 Java 编写的用户程序，针对该系统分布式并行计算的特性，采用基于 Java 的执行方式，即采用 Java 虚拟机(JVM)的方式运行于 JPVM 平台下，调用已经存在的 C/C++库模块和可执行代码，屏蔽结点异构特性，与各异构结点合作进行计算任务；对于用 C/C++编写的用户程序，有两种执行方式：一是调用 JNI 接口将其转换成相应的 Java 代码，按照上面的方式运行之；二是采用本地执行方式，直接地使用 PVM 和 MPI，完全编译并直接执行代码。

3 移动代理的工作过程分析

MA 模块是本系统的关键和核心模块，参照图 2，其工作过程可以这样描述：1)欲登录该网格环境的用户向因特网组织的 Web 服务器发起接入请求，Web 服务器将根据用户提

(下转第 68 页)

(2) 如果有历史 P_i，使得 P_i SUB Q，则无论 P_i 是否可满足的，此次数据库访问 *query*(Q)是不必要的。否则：

(3) 如果有历史 $P_1,\cdots,P_n$，并且 Q INT $P_i(i=1,\cdots,n)$，则首先在 $P_1,\cdots,P_n$中滤除不可满足的历史，使得 $H=\{P_i|P_i$ 是可满足的，$i=1,\cdots,n,\}$，然后按照4.1节中所述的差优化算法处理 Q 和 H。

2. 设 *query*(Q)是当前查询，并且是基的，则对 Q 不进行优化，并且由于{Q}中仅可能有一个元组，则结果集中不缓存{Q}，而直接由调用程序消耗。

结论 语义缓存技术主要用来解决网络通信能力不足造成的响应时间太慢等问题。事实上，利用语义缓存进行查询求值的复杂度高于利用数据库服务器进行查询求值的复杂度，如果处理不当，造成利用语义缓存查询求值所消耗的时空代价接近甚至超过网络通信能力不足带来的时空代价，语义缓存技术显然是没有意义的。本文首先给出语义缓存查询求值的一般算法，然后指出采用一般算法得到的结果存在关系表达式过于复杂和子表达式过多的问题，最后针对这些问题从语法一级进行分析并给出了两级优化方法和实现技术。特别是在我们的优化方法中，不仅讨论了通常语义缓存只缓存与数据库内容相关的情况，而且创造性地讨论了缓存与数据库内容无关的情况，并指出如果对失败查询也进行缓存，可以进一步减少了对数据库的无效访问和简化 Where 子句中的子表达式，以较小的优化代价得到较大的优化利益。

采用语义缓存查询求值优化技术，不但可以简化语义缓存查询求值的复杂度，加快语义缓存查询求值的速度，而且减少了用户对数据库服务器的无效访问，减轻了网络通信负载，使语义缓存技术向实用化大大迈进了一步。

致谢 在此，我们向对本文的工作给予支持和建议的同行表示感谢！

参考文献

1 Ugur C, Deno P J K. A Decentralized, Peer-to-Peer Object-Replication System for Weakly Connected Environments. IEEE TRANSACTIONS ON COMPUTERS, JULY 2003,52(7)
2 Li W-S, Po O, Hsiung W-P, Candan K S, Agrawal D. Freshness-driven adaptive caching for dynamic content Web sites Data & Knowledge Engineering,2003,47:269～296
3 Ren Q, Margaret H. Dunham Semantic Caching and Query Processing. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2003,1(1)5: 192～210
4 Dar S, Franklin M, Jonsson B, Srivastava D, Tan M. Semantic data caching and replacement. In: Proc. of the 22nd VLDB Conf. Mumbai (Bombay), India, 1996. 330～341
5 Cai jun, Tao kian-Lee. On incremental cache coherency schemes in mobile computing environments [C]. In: Proc. of ICDE,1997
6 Gottlob S C, Tanca L. Logic Programming and Database, Springer-Verlag,1990
7 吴婷婷，周兴铭. 基于语义缓存的移动查询导出. 计算机学报，2002,10:1104～1110
8 吴婷婷，章文嵩，周兴铭. 断接下查询的缓存处理. 计算机学报，2003,10: 1393～1399
9 李磊，左万历，李希春. PROLOG-DBMS 系统实现中的子句间优化技术. 软件学报，1995,6(3):136～141

(上接第60页)

供的资源信息对其进行身份验证，通过验证的用户允许向 Web 服务器上传自己的 MPI 或者 PVM 并行应用程序包；2)上传的应用程序包通过经纪(Broker)的转换变成 MA 能够识别的请求，将源码包发送到编译移动代理结点，并请求编译；3)移动代理接收到该用户的编译请求后，则根据用户所写的 makefile 文件，通过移动代理选择一个高性能、轻负载的计算结点对用户程序进行编译，并把编译过程中遇到的错误信息回送给浏览器；4)用户根据出错信息对程序进行修改，如此重复，直到编译通过为止；5)用户向经纪发起运行请求，与编译过程的处理过程一样，经纪将其转换成移动代理可识别的请求并发送给移动代理；6)移动代理收到运行请求之后，把已编译好的可执行代码分发到通过智能抉择所选的各个结点；7)移动代理根据所选结点信息生成一个符合 MPI 或 PVM 格式的配置文件，并作为主结点据此发起计算；8)运行结束后，把运行结果回传给浏览器，等待下一个用户进程的请求。

图2 移动代理工作过程示意图

结束语 本文介绍了一种独立于具体应用业务和特定开发平台的基于移动代理的网格计算概念模型。它把分布、异构、动态变化的计算资源集中起来，为用户提供高性能并行计算平台。模型中的 Mobile Agent 就是一组智能地代替用户完成工作的程序。智能 Mobile Agent 能够发现和自动适应资源的不断变化并对应用行为做出相应的调整，结合当前计算结点的性能指标、网络带宽、用户需求等选择较优的计算结点，因此与传统的并行计算随机选择结点相比可大大提高计算的性能。并且 Mobile Agent 可在多种体系结构下编译、运行程序，故该模型可有效屏蔽系统中计算结点的异构性。该模型很好地适用于动态的网格环境，使得开发具有高性能的应用成为可能。这种结构不依赖于具体应用程序的算法，具有可移植和重用性。在以后的研究中，我们将要在移动代理系统的安全性方面做进一步的研究，必须有较好的容错方案来保证移动代理的健壮性，集中在对移动代理的合法性验证、对移动代理所携带的数据的保护以及防止某些恶意攻击及对执行环境非授权的修改等方面。

参考文献

1 都志辉，陈渝，刘鹏. 网格计算. 清华大学出版社，2002
2 Gentzsch W. Grid computing, a vender's vision. In: Proc. of the 2nd IEEE/ACM Int'l Symp on ClusterComputingandtheGrid. LosAlamitos: IEEEComputingSociety Press, 2002. 272～277
3 Cao J W, Spooner D P, Turner J D, et al. Agent-based resource management for grid computing. In: Proc. of the 2nd IEEE/ACM Int'l Symp on Cluster Computing and the Grid. Los Alamitos: IEEE Computing Society Press, 2002. 323～324
4 Fukuda M, Suzuki N, Bic L F. Introducing dynamic data structure into mobile agents. In: Proc. of the1999 Int'l Conf. on Parallel and Distributed Processing TechniquesandA-pplications - PDPTA'99, Las Vegas, NV, June 1999. 1845～1860
5 Foster I, Kesselman C. The Globus Project: A Status Report. In: Proc. IPPS/SPDP'98Heterogeneous Computing Workshop, 1998. 4～18
6 Fukuda M, Tanaka Y, Campos L M, Kobayashi S. Inter-cluster job coordination using mobile agents. In: Proc. 3rd Int'l Workshop on Active Middleware Servic-es, San Francisco, CA, IEEE CS, Aug. 2001
7 Vetter J, Schwan K. Techniques for High - Performance Computational Steering. IEEE concurrency, 1999. 63～74
8 Vetter J S, Reed D A. Real time Performance Monitoring, Adaptive Contral, and Interactive Steering of Computing Grids. International Journal of High Performance Computing Applications, 2000. 357～366
9 Fosteer I. Computer Grids. In: VECPAR2000, 2002. 3～37
10 Fukuda M, Tanaka Y, Suzuki N, Bic L F, Koba-Yashi S. A Mobile-Agent-Based PC Grid. In: Proc. 3rd Int'l Workshop on Active Middleware Services, IEEE CS, 2001