计算机科学2004Vol.31№.9A

网络计算中的服务监控器及其在MICE中的实现

刘 颖 李 廉
(兰州大学信息科学与工程学院 兰州730000)

摘 要 本文讨论网络计算中服务监控器的设计和应用,实现对于网络计算服务过程的实时监视和管理,改善系统的运行状态,并结合现在正在开发的网络计算平台MICE(Mathematics Internet Computing Environment),实现对CSP(Computing Service Provider)的运行进行管理,直观、实时地监视CSP的执行情况,方便CSP管理员的控制和管理。本文详细阐述了CSP监控器的总体设计及其实现。

关键词 网络计算,MICE,CSP,M2,UCS,监视器,MySQL,XML

Computing Service Provider Monitor in Internet Computing and its Implementation in MICE

LIU Ying LI Lian
(School of Information Science and Technology, Lanzhou University, Lanzhou 730000)

Abstract As Grids are emerging as the next-generation computing platform, the need for Computing Service Provider Monitors that could display distributed resource conditions and help to control and steer execution of application jobs is increasing. We have responded to this requirement by developing an easy to use, lightweight Web portal called CSP Monitor in MICE(Mathematics Internet Computing Environment), which allows CSP administrators to visually see resource distribution condition and monitor, control, and steer execution of application jobs. This paper talks about its architecture, functions, design and implementation in details.

Keywords Internet computing, MICE, CSP, M2, UCS, Monitor, MySQL, XML

1 简介

网络计算利用现有的Internet连接和网络中的计算资源,将分布的计算机组织起来协同解决复杂的科学与工程计算问题。随着网络计算的普及,越来越多的人开始利用网络计算,研究人员应该提供简化操作的工具来屏蔽系统复杂性,方便计算任务的分配和管理。对网络资源和任务执行情况的监控是其中的重要内容。网络计算监控器正是针对这一目的提出的。

目前,网络计算中的监控器可以大致分为几类。

第一类监控器发现和监视整个分布式环境中资源的动态变化和使用情况。用户可以根据这些信息决定自己的程序最适合在哪台计算机上运行,管理员也可以根据这些信息调整资源的配置。这一方面的工作已经做了很多,有很多组织或个人都开发了针对自己的分布式系统的监视和信息服务(Monitoring and Information Services for Distributed Systems)工具,比如the Globus Toolkit(r) Monitoring and Discovery Service, the European Data Grid Relational Grid Monitoring Architecture (R-GMA)以及the Condor project中的Hawkeye。

另一类监控器是提供给一般用户以方便他们监视、控制自己应用程序的执行。澳大利亚墨尔本大学的G-Monitor是这一类监控器的代表。

此外,还有一类是针对计算服务提供者的监控器。计算服务提供者是网络计算中计算任务的真正执行者,涉及到任务的接受、分解、计算、转发和结果返回等方方面面,是网络计算中的关键部分之一,对它的监视和管理至关重要。针对计算服务提供者的监控器所监控的信息在某种程度上包含了上述两类监控器信息:不仅要直观、实时地显示本地可用的计算资源以及整个网络中可用的计算资源,而且还监控本地计算机上各个任务的执行状态、用户信息,以及一些配置、日志信息等。它为该计算结点的管理人员提供了简单、易用的界面,方便了管理员的控制和管理。

目前针对第三类监控器进行的研究开发并不多,本文将结合兰州大学数学网络计算环境MICE(Mathematics Internet Computing Environment)详细介绍计算服务提供者监控器的结构、功能、设计和实现。

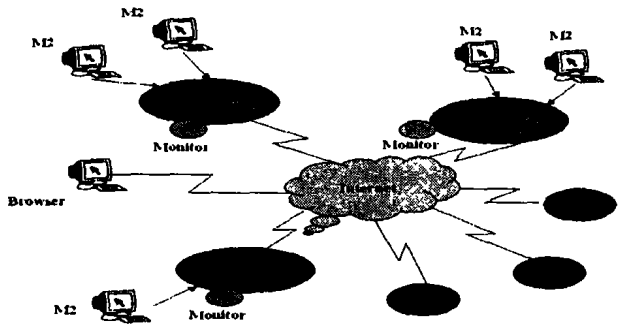
2 结构

首先我们简要介绍兰州大学的数学网络计算环境MICE(Mathematics Internet Computing Envi-

刘 颖 硕士研究生,研究方向:网络计算,分布和并行计算。李 廉 教授,博士生导师,研究方向:网络计算。

ronment)以及计算服务提供者监控器 CSP(Computing Service Provider Monitor)在这一平台中的位置和作用。

在网络计算一般模式和面向规范的程序设计思想基础上,我们提出了一种网络数学计算环境框架,并构建了一个灵活的数学网络计算应用平台 MICE。MICE 可以集成数学计算软件,它的功能和解题水平可以随着所集成软件的不断更新而升级,计算能力可以相对容易地保持其先进性和前沿性;而计算服务器之间具有自适应分布计算功能,所以也可以满足高性能计算的需求,提高了计算资源利用率。MICE 的结构如图1所示。



MICE结构图

图1

MICE 的整体组成主要包含:M2,CSP(Computing Service Provider)和 UCS(Universal Computing Service)。

M2是高度集成了数学文章编辑器、编译器、调试器于一体的窗口环境,即它不仅是一个编程运行环境,也是一个数学文章的书写环境,其界面同广大开发人员熟悉的 Microsoft Visual Studio 环境非常相似。

CSP 是整个 MICE 系统的核心,整个 MICE 架构中包含很多的 CSP,它们接受从 M2发送来的任务,执行并将结果再返回给 M2。任务提交和结果返回都是以 XML 文件的格式传送。

UCS 包含在 CSP 中,用于完成具体的计算任务,它包括本地计算服务资源(本地 CSP 提供)和远程计算服务资源(或者是其他 CSP 提供,或者是专门的计算服务提供者 MWS (MICE Web Service)提供)。任务被分解成子任务并计算,如果该任务在本地 CSP 的 UCS 中不能完成的话,将被发送到其他能执行该任务的 CSP 或者 MWS 上执行。

关于 MICE 的详细情况,请参见 http://mice.lzu.edu.cn 。

正是因为 CSP 的核心地位,对 CSP 的监视和控制至关重要。这一工作是通过 CSP 监控器(Monitor)来完成的。监控器是一个独立于 CSP 但是又和 CSP 密切相关的部分。CSP 管理员可以直接通过浏览器监视 CSP 的运行。本文将详细阐述监控器的功能、设计与实现。

3 功能

CSP 监控器应实现以下几部分功能:

启动/关闭 CSP:提供 CSP 当前的状态信息(是处于运行状态还是关闭状态),允许管理员通过网页上的一个按钮就可以启动/关闭 CSP。

CSP 配置:CSP 在启动的时候需要 IP、端口号以及 CSP 注册中心的相关信息。通常这些信息不会经常改变,但是监控器应该提供查看和修改这些配置信息的功能。这些配置信息被记录在一个 XML 文件中,监控器从 CSP 取得这个文件,再通过监控器的 Web 界面显示,如果管理员做了修改,监控器会将修改后 XML 文件返回给 CSP。

用户管理:M2要向 CSP 提交任务必须首先通过用户验证。用户可以通过 M2的注册界面注册。而 CSP 管理员可以在监控器中直接添加/删除/修改用户信息。

任务管理:这是监控器的重要组成部分。其中任务可以分成两部分:未完成任务(我们称为 Active Task)和已完成任务(我们称为 History Task)。对于 Active Task,CSP 管理员除了可以查看相关信息之外,还可以执行挂起/解挂/杀死操作。这两种任务信息中都包含了该任务执行的结果信息、子任务信息和 CSP 管理员对该任务操作的信息。

资源列表:CSP 上的资源可以分为三类:

CSP 函数资源:显示本地 CSP 提供的计算服务。

MWS:是 MICE 中提供专门计算服务的结点,它从 CSP 接受计算任务。MWS 列表显示网络中的 MWS 的相关信息,包括服务类别、描述、参数等。

其他 CSP:MICE 是一个数学网络计算环境,整个 MICE 架构中包含很多的 CSP,彼此之间相互合作,实现计算资源共享和并行计算。其他 CSP 列表将显示网络中其它 CSP 的相关信息,包括 IP、端口等。

CSP 管理员的登录/注销、账户管理和日志记录:由于通过监控器可以直接启动/关闭 CSP、修改 CSP 配置、控制任务的执行情况,而监控器又是一个 Web 应用程序,因此安全和管理员账户管理是监控器需要考虑的一个重要问题。每当一个用户登录到监控器的 Web 页面时,必须提供管理员用户名和密码,只有通过验证的用户才能进入监控器。监控器管理员可以查看和修改自己的账户信息。

监控器必须对管理员的每一个修改、控制操作(包括启动/关闭 CSP、修改 CSP 配置、增加/删除/更新用户信息以及挂起/解挂/杀死任务)自动记录。

这些日志只能查看不能修改，这样，一旦系统出了问题可以根据日志记录追究有关人员的责任。

4 设计

由于CSP要负责任务的接收、分解、计算、转发及结果返回，它本身已经是一个比较复杂的系统。因而监控器的一个设计宗旨就是尽量减少与CSP的联系和通信，降低CSP和监控器的设计复杂性。因此，在我们的设计方案中使用了数据库。CSP在工作的时候，将所有的工作状态和日志都记入数据库。监控器在工作时，直接从数据库中读出数据并显示。这样，CSP和监控器通过数据库作为联系的纽带之一(如图2所示)，降低了耦合度。同时由于数据库技术成熟，支持多用户的网络访问，实现了读写的并发控制，这也保证了CSP和监控器工作的正确性和效率。

但是由于允许CSP管理员直接在监控器中控制任务，比如挂起/解挂/杀死某一个任务。如果采用图2的结构：从监控器修改数据库、数据库的改变再触发CSP采取相应的动作，虽然也可行，但是却有一个严重的缺陷：很可能监控器成功修改了数据库，但是CSP却由于种种原因未能执行成功，这就会造成数据库与CSP实际的状态不一致。因此我们对图2的结构进行了修改而采用图3所示的结构。在图3中，CSP和监控器之间的联系虽然增多了，但是它们各自的功能却更加单纯和统一：CSP只负责写数据库，监控器只负责读数据库。如果CSP管理员从监控器中发出了控制操作，这一命令将直接发往CSP执行，CSP执行成功后修改数据库并给监控器发回一个执行成功信号，监控器只要刷新一次就可以看到最新的状态；如果失败则给监控器发回一个执行失败的信号，监控器也可以把相关的信息显示给CSP管理员。

图2

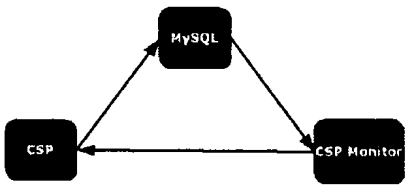


图3

我们采用的是MySQL数据库。MySQL是一个精巧的关系数据库管理系统。由于它的强大功能、灵活性、丰富的应用编程接口(API)以及精巧的系统结构，受到了广大自由软件爱好者甚至是商业软件用户的青睐，为建立基于数据库的动态网站提供了强大动力。

此外，我们在设计监控器的时候，不仅仅是想提供一个便于管理员操作和管理的图形界面，而且希望这个界面是个轻型接口(Lightweight Interface)。与一般"重型"的图形界面应用程序不同，Web程序提供了一种普遍、易用的轻型接口，而且便于CSP管理员从任何一台连上Internet的计算机上远程监控CSP，而不必一定要在CSP服务器上进行操作，也不需要专门的客户端程序。因此我们将监控器设计成Web程序，并采用基于Apache Software Foundation的Tomcat。Tomcat是一个优秀的JSP/Servlet服务器，同时也提供Web服务器功能。这样，多个CSP管理员同时通过监控器对CSP的动态、多点实时监控问题，就由Tomcat的多线程机制来控制完成，这降低了监控器设计实现的复杂性，而且也易于管理和维护。关于Tomcat，详情请见http://jakarta.apache.org/tomcat/index.html 。

MICE作为一个数学网络计算环境，需要考虑M2、CSP以及MWS位于不同操作系统计算机上的情况。为了解决这一跨平台问题，MICE所有部分的编写均采用Java语言。因此在实现CSP监控器时，我们采用的编程语言也是Java，同时使用了JSP、Applet和Servlet。

5 实现

数据库是CSP与监控器联系的主要方式，因此实现监控器时的一个重要任务就是合理设计数据库。根据监控器的主要功能，需要设计以下几种数据库表(Table)：

普通用户表；　管理员表；　管理员日志表；
任务表；　任务结果表；　子任务表；
任务操作表(记录管理员对任务的挂起/解挂/杀死操作)。

CSP与监控器的另一种联系方式是XML文件，这包括了CSP配置文件和三种资源列表。之所以采用XML文件也是源于MICE需要在网络中跨平台运行与通信，而XML很好地解决了这一问题，同时Java语言对XML也提供了很好的支持。

由于三种资源列表都是树状的层次结构，因此以树状图的形式显示效果最好。资源列表总是在动态变化的，而在网页中直接显示树状图效果并不理想，因此我们采用了Java Applet来完成这一工作。监控器在显示这些资源列表的时候，直接从CSP获得XML文件进行解析并以树状图的形式显示，不再经过数据库的转存。每次刷新时，其他CSP列表文件和MWS列表文件都从资源中心重新获取，因此能从CSP监控器中实时了解资源的分布和动态变化。

(下转第115页)

传统的顺序式的存储，对于 XML 文档的检索是非常不利的。XML 文档本身是有层次结构的。因此以顺序方式存储，对查询来说，会造成不必要的 I/O 操作，这些操作对 CPU 来说，耗费是巨大的。例如上述例子的分析，如果没有其他辅助文档(DTD 或者 Schema)，为了检索一段字符数据，我们就需要遍历整个文档。如果目标在文档的末端的话，那么整个文档就需要被调进内存，这个操作量对查询来说耗费是巨大的。就算是有辅助文档的帮助，就只是在数据库中搜索我们想要得文档这一个操作，就很令人头疼。在我们的 SDML 中，每个作为文档的元素的标签头都可以被常驻内存，省去了很多 I/O 操作，这样对包含大量文档的数据库来说，会节省很多时间。并且在进行文档内检索的时候，有些我们不关心的分支子树也可以不掉入内存，节省了大量的 I/O 操作。

3.2 封锁粒度分析

·传统的 XML 数据库，由于没有层次式的存储方式，文档都是以顺序文件的形式存储的。因此，对文档的封锁，只能在整个文档的粒度上进行封锁。对于一些要求并发操作的 XML 文档，可能同时会有几个事务对同一个文档的不同元素的属性进行修改。在这种需求下，对整个文档的封锁，显然是不合乎应用的。在我们的方法中提出的局部封锁的策略，就可以很好地解决这个需求。就算在一般的需求下，当用户要求修改属性的时候，传统的 XML 数据库也只能对整个文档进行封锁，而在层次式的存储中，这就意味着要对整棵文档树进行封锁，这样就会牵扯到很多的加锁、封锁以及访问权限的问题。但是在提出了局部封锁的概念后，用户修改元素属性的时候，只需要关心当前标签的改动，无需考虑对其他祖先及子孙的影响，省去了很多的加锁操作。例如，在对上面的例子中的 book 元素的属性 borrow 进行修改的时候，若要对整棵树进行封锁的话，只对标签头进行封锁，就需要至少4次封锁，其中还不包括对兄弟节点影响所产生的封锁。而在局部封锁中，只需要对 book 这个标签头进行封锁，进一步封锁它的属性块，然后进行修改就可以了。这样就省去了大部分的封锁耗费，对于结构复杂的 XML 文档，这种封锁的优越性就更明显了。

总结 在基于 SDML 的存储策略的基础上，我们提出的适应于 XML 文档封锁的策略，可以完全适应于要求高度同步性的系统，并且可以根据不同的需求选择适应应用的封锁粒度，可以大幅度地提高系统的效率。采用这种封锁策略，可以无形中减少很多不必要的 I/O 操作，对建立索引更是有很大的帮助。

该策略对 SDML 的改进还不是很完善。例如，在这篇论文中，我们没有涉及对改进的 SDML 的存储块的插入、删除和更新操作，在以后的工作中，可以在这方面继续研究。

参 考 文 献

1 Garcia-Molina H, Ullman J D, Widom J. Database System Implementation
2 Mohan C. An Efficient Method for Performing Record Deletions and Updates Using Index Scansk
3 Mark Graves. Designing XML Databases
4 萨师煊，王珊. 数据库系统概论. 北京：高等教育出版社

(上接第86页)

监控器的另一个重要功能是启动/关闭 CSP，这是监控器的设计难点之一。我们的解决方法是：让 CSP 作为监控器的一个线程，启动 CSP 就生成一个新的 CSP 线程；关闭 CSP 也就是结束这个线程。由于监控器位于 Web 服务器中，事实上，CSP 将作为 Tomcat 的线程运行。

结束语 随着网络计算日益广泛的使用，人们迫切需要通过可视化的监控器直观、实时地监视网络中资源的动态变化和任务的执行情况，并控制和管理任务的执行。对计算服务提供节点的监视控制是其中重要的一类监控器。本文以兰州大学数学网络计算环境(Mathematics Internet Computing Environment)中 CSP(Computing Service Provider)监控器为例，详细阐述了计算服务提供节点监控器的结构、功能、设计和实现。

参 考 文 献

1 程显华等译，Mark Wutka 著. JSP 和 Servlet 程序设计使用专辑. 机械工业出版社，2002
2 詹建. 面向规范的数学网络计算环境：[硕士论文]. 2003
3 晏子译. MySQL 中文参考手册. http://www.pcsoftware.com.cn
4 李昭智等译，Ivor Horrton 等著. Java 2 编程指南. 电子工业出版社，2003
5 Placek M, Buyya R. G-Monitor: Gridbus Web portal for monitoring and steering application execution on global grids. In: Proc. of the Intl. Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2002). Conjunction with the 12th International Symposium on High Performance Distributed Computing (HPDC 2003), Seattle, USA, June 2003
6 Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid Information Services for Distributed Resource Sharing. In: Proc. 10th IEEE Intl. Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, 2001
7 MDS3. http://www.globus.org/mds/
98 DataGrid Information and Monitoring Services Architecture: Design, Requirements and Evaluation Criteria: [Technical Report]. DataGrid, 2002
10 Hawkeye. http://www.cs.wisc.edu/condor/hawkeye