并行支撑环境 JavaPVM 的研究及实现*)

Research and Implementation of Parallel-Supported Environment JavaPVM

胡 宁 周笑波 杜 鹏 谢 立

(计算机软件新技术国家重点实验室 南京大学计算机科学与技术系 南京 210093)

Abstract Parallel-supported environment plays a very important role in distributed/parallel system. We design and implement a parallel-supported environment JavaPVM on a network which consists of heterogenous platform and heterogenous OS system connected by ATM. And it is applied in CSCW and Java language auto-complier.

Keywords JavaPVM, Parallel-supported environment, Java, PVM

1 引言

随着 ATM 等高速网络技术的发展和计算机硬件价格的下降,由小型机、工作站、微机等组成的分布、互连、协同操作的分布并行系统日益得到广泛的使用。现有的并行支撑环境[1]如 p4、Express、PVM 和 Linda 等,虽然为在分布式系统上编制并行程序提供了良好的编程环境,但是还是难以完全满足这种分布并行处理和异构型网络计算的需求。

随着 Internet 的兴起而流行的 Java 语言[3]由于它在平台无关性和安全性方面的优点得到了广泛的应用,但它在设计网络应用程序方面,只提供了基于 TCP/IP 的面向对象的接口,在实际的分布式程序设计工作中,程序员还需要涉及网络通信的许多细节并谨慎地设计和构造自己的程序接口。另外,每个分布式应用在消息传递方面都可能有其独特之处,如果每开发一个系统都要开发一套通信接口,势必造成相互之间在结构、接口等方面的不一致性,从而不利于软件的兼容性和互操作性,这既与 Java 的初衷违背,又不适合用户的需求。目前较为流行的 PVM[2]则以其支持多种体系结构和提供统一的接口著称。若将 PVM 支持多种体系结构的特点与 Java 语言的平台无关性和安全性相结合,则能较好地满足基于 Internet 的网络计算的需求。

目前国外相关工作主要是提供 Java 语言到 PVM 原有的 C 程序库的接口[8],使得用户可用 Java 程序调用原有 PVM 的函数,但其使用的平台必须同时支持 Java 和 PVM。我们从 93 年即用 PVM 作为我们分布并行系统的支撑环境,并在其上开发了智能操作系统 KZ3/KZ4。本文中,我们改进了原有 PVM 的多目通信机制,在一个由 ATM 高速网连接的异构型网络上设计并实现了一个并行支撑环境 JavaPVM,并应用于 CSCW 和 Java 语言的自动并行编译,取得较好的效果。

2 并行支撑环境 JavaPVM

并行支撑环境是以计算机互连网络为物理基础,为用户编写及运行分布并行程序提供的一套工具及其运行环境。并行支撑环境的作用在于使得分布并行应用的开发变得相对容易,特别在分布控制和消息传递方面为用户降低了复杂性,对用户屏蔽分布并行应用在网络通信和任务分布的控制问题。

并行支撑环境 PVM 是以 Unix 为基础,支持多种体系结构的计算机系统,它将一组计算机看作单个的并行虚拟机,在包含不同体系结构的计算机网络上透明地处理消息传递、数据转换和任务调度工作,具有与计算机体系结构无关和系统简单高效等优点。

*)本研究得到 863 高科技项目基金资助。胡 宁 硕士生,主要研究领域为分布式操作系统。周笑波 博士生,主要研究领域为分布式操作系统。杜 鹏 硕士生,研究领域为分布式计算,CSCW。谢 立 教授、博导,主要研究领域为分布式计算和并行处理。

JavaPVM 是我们开发并已通过鉴定的 863 项目“实用化的分布式软件开发工具包”的核心(如图 1 所示),它集成网络程序语言 Java 和分布式并行处理系统 PVM 这两项关键技术,将不同体系结构和操作系统的计算机通过网络有机地联系在一起,为 Java 并行应用构造了一个大规模的并行计算虚拟平台,并为分布式多媒体软件开发工具、自动并行编译工具和协同软件开发工具提供底层支持。

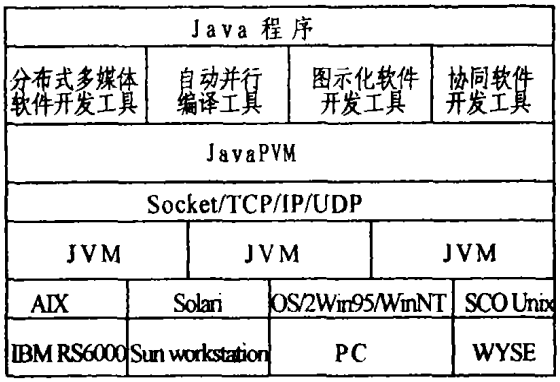

图 1 Java 语言开发工具包的总体结构及其运行环境

3 JavaPVM 的结构和功能

3.1 JavaPVM 的结构

JavaPVM 由通过分布并行计算环境中各结点上运行的 JavaPVM 监控进程 JavaPVMD 相互通信连接而成,图 2 所示的是 JavaPVM 的模型。

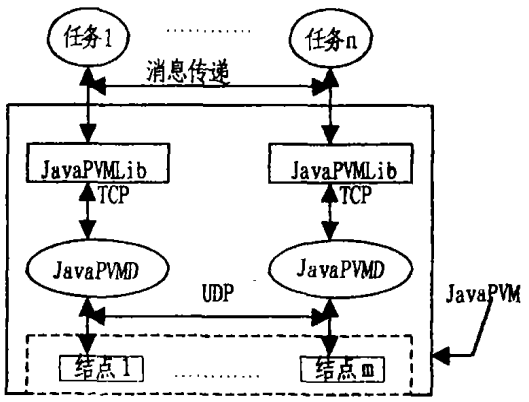

图 2 JavaPVM 的模型

3.2 JavaPVM 的功能模块

在 JavaPVM 的每个结点上都有一个独立的 JavaPVM 环境,图 3 所示的是单个 JavaPVM 结点的软件模块化结构,由以下七个部分组成:系统启动类、共享信息区、本地服务线程、远程连接类、远程守护线程、本地监控类、编程类库。

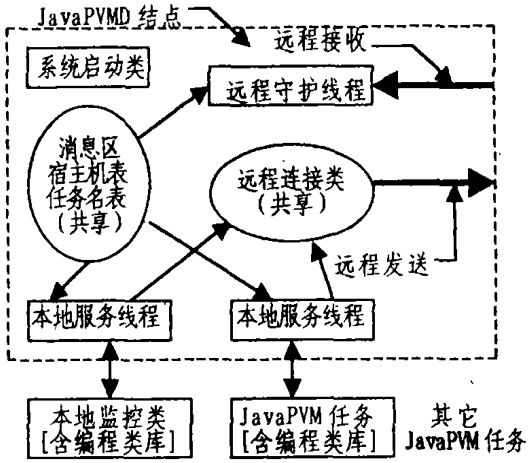

图 3 JavaPVM 结点内部框架示意图

(1)系统启动类。是 JavaPVM 系统的启动模块,对 JavaPVM 结点中所需的对象及数据进行初始化,包括:创建宿主机名表、任务名表和消息区供系统内部各个线程共享;创建 Socket 对象供编程类库使用;启动远程守护线程,监听来自远程 JavaPVM 结点的消息;在 JavaPVM 运行过程中,系统启动类始终监听来自本地新任务的连接请求,一旦接收到这种请求,就派生本地服务线程为之服务,从而为进入 JavaPVM 环境的任务服务。

(2)共享信息区。由宿主机名表、任务名表和消息区组成。

·宿主机名表 它保存着当前进入到 JavaPVM 环境中所有宿主机的名称。每当有新的宿主机进入(或退出)JavaPVM 环境,宿主机名表中就会添加(或删除)一项,并通知当前连入 JavaPVM 的所有宿主机,从而更新 JavaPVM 环境中各个结点上的宿主机表,这样就可以动态地改变并行虚拟机中的宿主机个数,因此能方便地实现 JavaPVM 环境下的宿主机配置管理。

·任务名表 它保存着当前所有 JavaPVM 任务的名字及其所在的宿主机名。每当有 Java 应用程序的进程(或线程)申请进入(或退出)JavaPVM 环境,任务名表中就会添加(或减少)一项,并将该任务的请求通知所有宿主机,以便更新各台宿主机上全局的任务名表。

·消息区 它保存着的是发送给本地 Java PVM 任务的消息。这些消息既可能是本机任务发送的,也可能是远程任务发送的。

(3)本地服务线程。在 JavaPVM 环境中,任何一个 JavaPVM 任务都存在一个本地服务线程与之对应,它们之间建立一个 TCP 连接通道。用户可以使

用 JavaPVM 中的编程类库 JavaPVMLib 方便地存取 TCP 通道。本地服务线程能够提供下列服务功能：任务加入(或退出)、发送消息、接收消息、增加(或删除)宿主机、启动(或杀死)一个 JavaPVM 任务等。

(4)远程连接类。为 JavaPVM 中的各个部分提供统一的远程发送接口，处理 JavaPVM 中远程发送的消息，如用户程序中 JavaPVM 任务之间的消息，或 JavaPVM 系统中各个 JavaPVM 结点之间协调远程结点消息。

(5)远程守护线程。它与远程连接类一起，完成 JavaPVM 远程发送和协调全局的功能，远程守护线程始终在某个端口上接收远程结点以 UDP 方式发送的消息，而远程连接类则专门为所有的远程通信请求服务，向远程守护线程所监听的端口发送消息。远程守护线程响应远程连接类所能发出的任务间消息和协调控制消息。

(6)本地监控类。是 JavaPVM 这个并行虚拟机的主控制台(Console)，是用户使用 JavaPVM 的终端控制程序，它的作用不仅在于为用户提供手工控制 JavaPVM 的工具，而且在对系统的跟踪方面对用户提供支持。

(7)编程类库。JavaPVMLib 是用户的编程接口，提供分布式系统中的进(线)程间通信、任务控制等功能供其调用。用户在构造应用程序时只需创建一个 JavaPVMLib 对象，就可与 JavaPVM 建立联系，成为一个 JavaPVM 任务，参与一个由多结点构成的并行虚拟环境。

4 JavaPVM 的实现

4.1 JavaPVM 实现环境

我们在由 155 兆 ATM 连接的一台 4CPU 的 WYSE Series 7000i(SCO Unix)、4 台 Sun SPARC-Station(Solaris)、6 台 IBM RS6000(AIX)和若干 PC(Win95/NT 和 os/2)的分布式环境上实现了上面所介绍的 JavaPVM。

4.2 JavaPVM 的实现

(1)共享信息区的实现。共享信息区由宿主机名表、任务名表和消息区组成，为了提高 JavaPVM 的坚定性和适应性，我们采用动态存储形式，即用队列的方式进行管理。利用对象的多态性实现管理宿主机名表、任务名表和消息存储区的类，以队列管理类为基础，实际运行时为这三个共享信息分别创建管理对象。基础管理类中不指定结点的类型和具体细节，而由运行时刻代入方法的结点类型进行动态绑定。

(2)通信的实现。在 JavaPVM 中，JavaPVM 任务与本地服务线程之间的通信是以建立 TCP 管道的方式实现的。JavaPVM 任务使用编程类库与 JavaPVMD 建立连接，JavaPVMD 在固定端口上监听到连接请求后，创建一个本地服务线程，并在其和任务之间建立 TCP 管道，双方通过管道进行通信。远程 JavaPVM 结点之间的通信则是采用无连接的 UDP 方式，以提高速度，减少开销。

(3)编程类库的实现。编程类库 JavaPVMLib 是 JavaPVM 中很关键的部分，JavaPVMLib 在保持了与 PVM 一致的风格的同时，对 PVM 的某些函数进行了改进和扩展。为了方便程序员编写网络应用程序，我们采用任务名而不是 PVM 的 Tid 来作为任务的标识，这样各任务只需根据事先约定的任务名进行通信，而不必像 PVM 那样要知道对方的 Tid。另外在 PVM 中，多目通信 MultiCast 不能保证顺序性和原子性，我们在 JavaPVMD 中加入了特殊的控制，从而保证了多目通信的顺序性和原子性。

(4)多目通信机制的实现。原有 PVM 的多目通信 MultiCast 只是简单地将信件逐一发出，不能保证顺序性和原子性。我们在设计 JavaPVM 时，对 PVM 的多目通信机制进行了改进和扩充。在 JavaPVM 建立时，由最早加入的 JavaPVMD 创建一个令牌。任务进行多目通信时，将信件按照目的任务所在的主机分为两类，所有目的任务都在本机的是本地信，有目的任务在它机是全局信，对于本地信 JavaPVMD 直接发送信件；对于全局信 JavaPVMD 根据令牌算法[1]申请令牌，获得令牌后再发送信件。发送完成以后，再将令牌交给下一个申请令牌的 JavaPVMD。这样在同一时间里只有一封全局信被发送，保证了多目通信的顺序性，而不同主机上的本地信可以同时发送，而且本地信也可以和不含本机的目的进程的全局信同时转发，具有较高的并行性。

4.3 JavaPVM 性能分析

考虑到 Java 和 C 在效率上的差异，我们对 JavaPVM 和 PVM 的通信性能进行了比较。比较的方法如下，任务 Master 向任务 Slave 发一个消息，Slave 收到后立即向 Master 发相同的消息。假定消息从 Master 到 Slave 和从 Slave 到 Master 所需时间相同，则可计算出 PVM 和 JavaPVM 发送不同大小的消息所用的时间。比较的结果是，当消息较小时，PMV 所用时间比 JavaPVM 小得多，随着消息

字节数的增加，JavaPVM 和 PVM 的差距逐渐减小。从比较的结果来看，JavaPVM 的通信性能是可以接受的。

5 JavaPVM 应用实例研究

CoPaintbrush 是一个计算机辅助协同工作系统，它采用"你见即我见(WYSIWIS)"的策略，用于多个用户通过计算机网络协同工作。对计算机辅助协同工作系统来说，通信和消息的管理是必不可少的，这些功能是所有协同应用软件得以实现和正常运行的基本前提，通信的效率和消息单元的管理质量直接影响到协同应用软件的性能。此外，协同工具的特殊性还要求环境的动态配置，允许参与协同工作的成员动态地加入或退出协同工作。

在采用 Java 语言标准的通信编程类库实现 CoPaintbrush 时，需要考虑以下这些细节问题。创建新的对象和发送消息时，都必须指定对方的机器名，这就要求程序员了解对方所在计算机的名称，这对程序员是十分苛刻的，而且参与协同工作的成员也相应地固定了，不利于软件的灵活性。同时，程序员还必须合理而谨慎地为系统的各个部分安排通信用的端口号，一旦发生错误，就会导致消息的丢失或误传。另外，Java 的通信类库不支持广播和组播，这些功能需要程序员自己实现。

我们采用 JavaPVM 可以方便地解决通信和动态配置问题。在通信方面，JavaPVM 消息发送和接收都是以任务名为依据的，只要每个用户程序都以某个唯一的名字加入 JavaPVM 环境，以后在向对方发送或从其接收消息时，都只需以它的名字为依据，这样就避免了设置端口号、获取动态协同环境、对远程消息进行错误检查等问题。在动态配置方面，JavaPVM 的任务可以随意地或退出 JavaPVM 环境，同时编程类库 JavaPVMLib 中提供了获取协同环境配置信息的方法调用。任何一个用户接口程序只需自行加入 JavaPVM，并根据动态获取的其它用户接口的有关信息，决定消息发送和接收的策略。只要在程序中约定任务命名规则，就可以识别当前 JavaPVM 下的所有协同用户接口，从而解决协同作图工具的动态配置问题。另外，JavaPVM 还提供了可靠的多目通信，以及远程任务启动和远程任务删除等远程任务控制方法。

综上所述，用 Java 语言直接开发协同作图工具 CoPaintbrush 会遇到种种不便，从而影响了系统整体设计与开发。作为一种高效的并行支撑环境，JavaPVM 为程序员提供的更高层次的通信手段及对分布式应用程序的其它支持，使他们从繁琐的通信细节中解脱出来，并使用很多有利于分布式程序设计的手段，给协同工具的开发带来很大的方便。

结束语 本文阐述了基于 Java 的并行虚拟机 JavaPVM 的设计和实现。我们将 Java 语言和 PVM 相结合，利用 Java 平台无关性的显著特点，以 PVM 的设计思想为基础，参考了它的功能和接口，构造出一个将不同硬件环境、不同操作系统的计算机联系在一起，进行分布并行计算的虚拟平台，并已在 CSCW、Java 语言自动并行编译等领域加以应用，取得了较好的效果。今后我们还要在提高通信性能等方面继续做工作。

参 考 文 献

1 孙钟秀. 分布式计算机系统. 国防工业出版社，1987

2 Geist A, et al. PVM: Parallel Virtual Machine — A User's Guide and Tutorial for Networked Parallel Computing. The MIT Press

3 Gosling J, McGiltion H. The Java Language Environment: A Whit Paper

4 The Java Language Specification Version 1.0 Beta, Sun Microsystems Computer Corp. Beta Draft of October 30, 1995

5 Geist A, et al. PVM3 User's Guide and Reference Manual. Oak Ridge National Laboratory, May 1993

6 周笑波，汲化，谢立. 基于 PVM 的分布计算的研究. 计算机学报，1997，20(6)

7 Sunderam V S, et al. The PVM Concurrent Computing System: Evolution, Experience and Trends. Parallel Computing, 1994, 20

8 Ferrari A J. JPVM: The Java Parallel Virtual Machine, June 1996

9 Thurman D A. JavaPVM: The Java to PVM Interface, December 1996