JAPS PTSP 并行任务支撑平台 并行编译

计算机科学 2000Vol 27№ 7

5-7,81

JAPS 中的并行任务支撑平台 PTSP*)

PTSP:The Parallel Task Support Platform in JAPS

李晓明 陈道蓄 谢 立 TP314

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

Abstract In parallelizing compiling system, the functions and the efficiency of the runtime support platform have a great impact on the result of the parallel execution. The thread and the RMI mechanisms in the Java language provide the distributed and parallel applications with efficient supports. In the automatic parallelizing compile system based on Java, we designed and implemented the parallel task support platform PTSP.

Keywords Parallel task, Distributed-memory, Support platform, Thread, RMI

1 概述

在现代计算技术中,分布并行处理越来越成为一种关键性的技术,这种由许多小任务合作解决大问题的方法,在过去几年发挥着越来越重要的作用,从高性能科学计算到日常事务的应用程序,都广泛接受和采纳分布并行处理.这是由于对高性能、低代价及强计算能力的需求所导致的,而大规模并行处理机(MPP)的出现及分布式计算的广泛使用则是其中两个最主要的方面。

分布式计算是解决科学计算问题的一次突破,它是由一组通过网络互连的计算机集中解决单个大的计算任务的处理方式。构成工作站网络(NOW)或工作站群集(COW),而实际上这样的连接所具备的计算资源可能已经超过了一台高性能计算机。

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式并行应用程序提供的一套工具及其运行环境。并行支撑环境的作用就在于使得开发分布式并行应用程序的工作变得相对容易,至少是在分布控制和消息传递方面可以为用户降低复杂性,为用户屏蔽分布式应用程序在网络通信和分布任务控制问题,使其对用户透明。

一般说来并行支撑环境应当至少为用户应用程序提供下述三项功能:①定义并行执行的一组子任务;②启动及终止这些子任务的执行;③在它们执行期间提供协调和交互手段。

对这些功能的调用同时也反映了并行执行的概念.它们必须以编程语言的形式出现,或者被编译器或其它语言处理器自动地插入程序中。此外,并行支撑环境最好对异构型计算也有一定的支持,这不仅是出于方便软件研究人员的考虑,也是为了提高软件的可扩充性和可移植性,同时对并行支撑环境本身的适应性也是一种提高。

随着自动并行编译研究的发展,目前已经提出很多并行任务支撑平台,其中大部分是包括在一个完整的并行编译系统中。Indiana University 的 Javar 项目[3]主要解决循环和递归方法中的数据依赖分析问题,Javar 结合了编译时分析和运行时动态确定依赖关系,重点在运行时的支撑软件,因此它的并行支撑平台还包括动态依赖分析,并且其任务也是以线程形式存在。S. Novack 等人[4]提出的并行支撑设施主要集中在 Java 的伪代码这一层次,需要结合硬件对 Pipeline 的支持,因此可移植性有限。L. V. Kale 等人[5]提出的动态支撑主要考虑如何在 NOW 环境下实现负载平衡,对单个任务的高效执行考虑不多。W. Shu[6]主要是对线程任务的并行执行提供支持,目的是在分布式存储环境下调度线程任务,但它对任务之间的通信没有提供高效的支持。R. D. Blumofe[7]等人也提出一个基于多线程的运行时支撑系统,但它主要适用于共享内存系统,其中也没有考虑通信支持设施。

2 并行任务支撑平台(PTSP)

2.1 PTSP 概述

*)本文所涉及的项目 JAPS 属于国家攀登计划 B 的课题"大规模并行编译和操作系统的某些关键技术"的一项研究内容,同时受到了"863"课题"基于 JAVA 的使用化分布并行工具包"的支持.

PTSP 在分布式并行计算系统中的位置如图 1 所示，其作用在于将独立的计算机系统通过网络有机联系在一起，为 Java 应用系统构造一个虚拟的大规模的并行计算平台；它由一个 Java 执行程序 PTSPD.class 和一套编程类库 PTSCLib 构成，其中运行在 JVM 上的 PTSCD.class 是后台运行支撑，负责管理 PTSC 环境下的宿主机、任务及其相互之间交互的消息，它是为 Java 应用提供通信支撑，从而使其能够在 PTSC 这一虚拟并行平台上运行的核心；PTSCLib 则作为一个开发工具包，为程序员提供了在 PTSC 下实现任务之间相互通信的通信原语和参与 PTSC 管理的一套工具，这套工具是基于高效 Java RMI 机制的，因此对程序员来说十分直观，使用起来也十分方便，它是程序员与 PTSC 的接口，使得程序员只需使用 Lib 中对他有用的部分，而不必知道其实现细节。这种结构的设计思想是从 RPC 中吸收而来的。

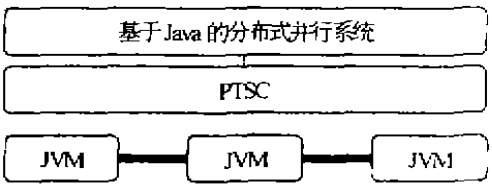

图 1　PISC 在分布式并行系统中的逻辑位置

假设在一个分布并行的计算环境下，所有的计算机系统上都启动了 PTSCD 这个进程，则用户使用 PTSC 构造出来的应用程序，就可以利用这个环境下所有的计算资源，如通过添加或删除宿主机的办法，让不同的计算机加入或退出逻辑上的并行虚拟机；在这个并行虚拟机中，所有的任务都用一个唯一的任务名进行标识，任务与任务之间的消息传送都是以任务名为标识符的，每个任务在传送消息时都不必关心对方运行在哪台计算机上，只需要以按名传送的方式发出消息就可以了；在这个并行虚拟机中，用户可以自行控制程序的分布，也可以交由 PTSC 进行调度，PTSC 将根据当前并行虚拟机中各个结点的负载程度，适当地将新任务分配到比较空闲的计算机上运行，以提高程序整体运行的效益。

综上所述，PTSC 是一个基于 Java 的并行虚拟机，它和著名的 PVM 一样，都为应用程序的并行运行提供了一个虚拟的并行计算机，使得在其上运行的应用程序可以忽略网络和通信的细节，将其看作一个规模更大的并行计算机系统。

2.2　PTSC 的结构

PTSC 是一个用 Java 语言开发的分布并行软件的支撑环境，它为基于 Java 的分布式并行程序提供了一个虚拟的并行平台，这个虚拟平台是逻辑上的概念，是通过软件来实现的，更具体地说是通过分布并行计算环境中各个结点上运行着的 Java 进程 PTSCD 相互通信连接而成的，因此，每个结点上都有一个独立的 PTSC 环境，多个结点的连接也就构成了一个逻辑上的并行平台。

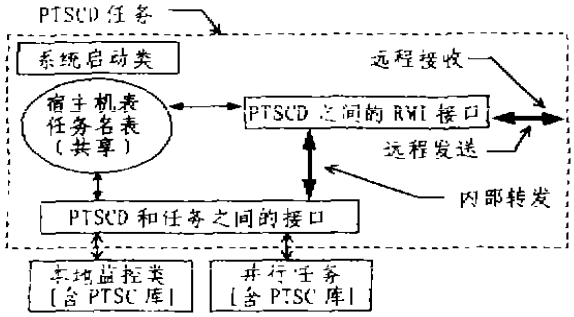

图 2　PTSP 内部框架示意图

图 2 所示的是单个 PTSCD 结点的软件模块化结构，图中每个结点由系统启动类；共享信息区；PTSCD 之间的接口，PTSCD 和任务之间的接口；PTSC 库等五个部分组成。

2.3　PTSC 的功能模块

2.3.1　系统启动类　系统启动类对 PTSCD 结点中所需要的对象及其它数据进行初始化，包括：

· 创建宿主机名表和任务名表供系统内部各个线程共享，利用线程共享主进程数据的特点，在最外层定义这些数据，相当于开辟了一块共享存储区，以后 PTSC 的所用服务线程都可以对这个存储区进行读写，从而达到信息交换的目的；

· 根据 RMI 的要求启动 PTSCD 之间的接口和 PTSCD 和任务之间的接口。在 PTSC 运行过程中，系统启动类始终监听着来自任务和其它 PTSCD 的连接请求，一旦接收到这种连接请求，就会派生本地服务线程为之服务，从而为进入 PTSC 环境的任务服务。

总之，系统启动类完成了单个 PTSC 结点上的全部准备工作，并随时接受新的 PTSC 任务加入 PTSC 环境。

2.3.2　共享信息区　共享信息区在 PTSC 中的作用和地位是至关重要的，它们的效率和可靠性的高低直接影响着 PTSC 本身的效率和可靠性。PTSC 中的共享信息区由宿主机名表和任务名表组成。

(1)宿主机名表。保存着当前加入到 PTSC 环境中的所有宿主机的名称。每当有一台新的宿主机加入(或退出)PTSC 环境，宿主机名表中就会添加(或删除)一项，并通知当前进入 PTSC 中的所有宿主机，从而更新 PTSC 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和成员，因此能够方便地实现 PTSC 环境下的宿主机配置管理。

(2)任务名表，保存着当前所有 PTSC 任务的名字及其所在位置(即宿主机名)，每当有一个 Java 应用程序的进程(或线程)申请加入(或退出)PTSC 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的宿主机，以便更新各台宿主机上全局的任务名表。

这两类共享信息是 PTSC 管理每个结点及整个多机环境的关键依据，用宿主机名表存放 PTSC 环境中现有的所有宿主机信息，用任务名表存放 PTSC 环境中现有的所有任务信息，它们都相当于一种容器，存放着与 PTSC 直接相关的数据信息，我们对其采用统一的动态存储方式加以管理，通过它们的配合使用，我们可以有效地管理 PTSC 中这两类关键性的数据，实现并行虚拟机对程序员的透明性，为 Java 任务提供一个逻辑上的并行平台。

2.3.3 PTSCD 和任务间的接口 为了方便 PTSC 环境中的任务调用 PTSCD 的功能，以及为了方便建立代表自己任务的任务接口，PTSCD 为任务提供了一套基于 RMI 的调用接口。一旦有一个任务加入 PTSC 环境，PTSC 的系统启动类就会派生出一个本地服务线程与之建立 RMI 连接，此后这个 PTSC 任务所有与 PTSC 环境有关的功能调用都将通过与服务线程的信息交换完成。PTSCD 本身也是作为 PTSC 环境下的一个普通任务存在，因此其它任务调用 PTSCD 服务也使用普通任务间通信接口。PTSC 任务调用这些功能的实质，就是将调用请求的请求码附上必要的参数，作为一个消息传送给 PTSCD 任务，PTSCD 便可以对消息进行解码识别，并且采取相应的动作。

2.3.4 PTSCD 之间的管理接口 该接口响应其它 PTSCD 所能发出的任务间消息或系统控制消息，用来维护整个 PTSC 环境。和 PTSCD 任务接口一样，调用 PTSCD 管理接口也是采用 PTSC 任务间通信接口，这样就保证了 PTSC 环境下任务通信的一致性。PTSCD 管理接口提供的功能主要包括：加入宿主机；断开宿主机；维护任务列表。

2.3.5 PTSC 编程类库 PTSC 是分布式并行应用程序的支撑环境，支持用户使用 Java 语言书写的分布式应用程序的执行，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，因此 PTSC 需要为用户提供编程接口，其中最主要的就是 PTSC 任务与 PTSCD 之间的通信管理者(PTSCLib)类，亦即 PTSC 中用户程序的通信管理者)，任何 Java 应用程序都可以通过创建一个通信管理者对象(类 PTSCLib 的实例)加入 PTSC 环境，成为一个 PTSC 任务。它是一个简单的 Java 类库，包含调用 PTSC 功能的接口类——PTSC 方法库类(PTSCLib)，用户在构造应用程序时，只需在程序中创建一个 PTSCLib 对象，就可以与 PTSC 建立联系，并使自己成为一个 PTSC 任务，参与一个由多结点构成的并行虚拟环境。PTSCLib 编程类库中的接口类 PTSCLib 就是使用 RMI 提供了各个方法的具体实现，因此程序设计员不必了解太多的 Java 通信类库和 PTSC 细节，就可以通过 PTSCLib 提供的编程接口与 PTSC 的本地服务线程进行交互，获取它所提供的服务，从而在 PTSC 的基础上设计和开发自己的分布式应用系统。

对于任务之间发送消息而言，情况相对简单。在 PTSC 环境中，都是通过名字来获得任务的 PTSC 接口。因为只使用 RMI 接口方式传送数据，对本地任务和远程任务的通信方式相同，因此 PTSCLib 实现了通信方式的一致性。在 PTSC 中，为了简化通信的端口管理，我们采用的单一端口收发的方式，也就是说，接收者始终在某个固定端口(RMI 端口)上监听消息，而发送者也始终在某个固定端口上发送消息。每个 PTSCLib 具有自己的消息缓冲区，发送和接受都可以异步进行。所有 PTSC 环境下的通信都是分布式的，不存在单一瓶颈，因此为并行任务的执行提供了较好的支撑。它的主要功能包括：获取当前 PTSC 环境下的宿主机名称和任务名称；发送消息缓冲区的控制及使用；准备发送内容；发送消息；接收消息缓冲区的控制及使用；异步和同步接收；读取消息缓冲区内容；任务退出 PTSC 环境；向当前 PTSC 环境中添加宿主机；删除当前 PTSC 环境下的宿主机；启动新任务；杀死任务。

3 PTSP 实现

3.1 实现环境

本系统的运行硬件环境为由多台个人计算机，多台 RS6000 工作站及多台 Sun Sparc 工作站通过 EtherNet，ATM 网构成的分布式并行计算环境，采用的操作系统包括 Mircosoft Windows 95，Microsoft Windows NT 4.0，IBM OS/2 3.0，IBMAIX 4.2，Sun OS5.x。整个 PTSP 系统用 Java 语言实现，需要 JDK1.1 以上的版本。

3.2 PTSP 的实现

在 PTSP 中的 5 个主要部分分别用 5 个 Java 类实现，对于需要给出功能接口的 PTSCD 任务接口，管理接口和 PTSC 类库，按照 Java RMI 的要求给出接口描述文件。开始时 PTSC 环境中的任务是每个节点机上的 PTSCD，通过程序调用或者用户命令将各个 PTSCD 连接成互连的 PTSC 环境。当启动一个任务

(下转第 81 页)

程的方法来开发大规模、交互式的 Web 应用程序，并提出和建立了很多新的模型、开发方法和工具系统。下面，本文介绍几个有代表性的 Web 应用程序的开发方法。

4.1 WebComposition

为了解决 Web 应用程序维护困难的问题，德国 Karlsruhe 大学的 Gellersen 等人提出并建立了 WebComposition 系统：面向对象的 Web 应用程序开发方法[2]。

WebComposition 系统建立在面向对象的 WebComposition 模型的基础上。根据该模型，一个 Web 应用程序可以按层次被分解为一组不同粒度的构件。每个构件都有自己的状态和行为。构件的状态定义为构件的一系列属性，构件的行为定义为对构件状态的操作。WebComposition 模型支持构件的共享和基于原型的继承。WebComposition 系统在一个数据库 Component Store 的基础上实现了该模型的原型。WebComposition 系统为 Web 应用程序的构件提供了持久存储，支持从构件到基于文件的资源的自动转化，并在程序的整个生命周期中维持对构件的存取。

4.2 W3DT

有鉴于大型 Web 应用程序开发的复杂性和困难性，M. Bichler 等人提出了 W3DT（World Wide Web Design Technique）：快速开发 Web 应用程序的方法[3]，以支持 Web 应用程序，尤其是大规模 Web 应用程序的快速开发。

W3DT 吸收了原型化开发和实体-关系模型的思想，为了快速构造一个 Web 应用程序，W3DT 提供了页面、索引、表单、菜单、链接、动态链接和图例等基本元素。W3DT Web-Designer 是基于 W3DT 的计算机辅助 Web 应用程序的开发环境，它支持 W3DT 设计阶段的图形化表示，并且能够在设计过程的每一步生成相应的 HTML 页面和 CGI 脚本。

结束语 Web 应用程序的开发是软件工程的一个新领域，使用正确有效的方法加上优良的支撑环境和 CASE 工具是开发 Web 应用程序的必然措施。本文论述了开发 Web 应用程序的特点和难点，分析了两种传统的软件开发方法，并提出了改进的模型，最后，本文介绍了 Web 应用程序开发方法和工具系统的最新进展。

参考文献

1 Kristensen A. Developing HTML Based Web Applications Issues in the Development of HTML Based Services, 1998. Available at, http: // www-uk hpl. hp. com/people/ak/doc/webe98 html

2 Gellersen H-W, et al. WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. Computer Networks and ISDN Systems, 1997, 29: 1429～1437

3 Bichler M, Nusser S. Developing Structured WWW-Sites with W3DT. WebNet 96, San Francisco, CA, 1996

（上接第7页）

时，该任务通过 PTSC 类库连接到某个 PTSCD 上，从而进入 PTSC 环境。该 PTSCD 通过管理接口将该变化通知给其它 PTSCD，实现任务名表的同步。当任务退出时，也进行类似的处理。当任务之间通过名字传送消息时，首先从某个 PTSCD 通过 PTSCD 任务接口获得目的任务的 PTSC 接口，然后直接向该接口传送数据，因此任务之间的消息传送是分布式的。

总结 PTSC 采用 Java RMI 机制，为 JAPS 中并行任务的执行提供了一个分布式的并行支撑环境。由于 PTSC 提供了对线程任务的支持，能够通过操作系统对线程的支持进一步实现单节点上的并行性，因此并行任务的执行效率有了极大的提高。PTSC 提供一个统一的接口界面给用户和 PTSC 上的任务，具有界面的一致性和简单性。在进一步的研究工作中，我们将考虑提高 PTSC 的效率和如何在 PTSC 中对任务进行优化调度。

参考文献

1 Gosling J, McGilton H. The Java Language Environment: A White Paper, 1995

2 Du Jiancheng, Chen Daoxu, Xie Li. JAPS: An Automatic Parallelizing System Based on JAVA. Science in China, 1999, 3: 279～288

3 Bik A J C, Gannon D B. Automatically Exploiting Implicit Parallelism in Java. [Technical Report TR473]. Indiana University, Jan. 1997

4 Novack S, Nicolau A. Resource-Directed Loop Pipelining. Languages and Compilers for Parallel Computing. Lecture Notes in Computer Science, Springer, 1996

5 Kale L V, et al. Load Balancing in Parallel Molecular Dynamics. In: Fifth Intl. Symposium on Solving Irregularly Structured Problems in Parallel. 1998

6 Shu W. Runtime Support for User-Level Ultra Lightweight Threads on Distributed Memory Computers. Journal of Supercomputing, 1994, 9(1/2): 91～103

7 Blumofe R D, et al. Cilk: An Efficient Multithreaded Runtime System. Journal of Parallel and Distributed Computing, 1996, 37(1)