KEYIZHIDEYUANCHENGJISUANHUANJINGJIQIJISHU

可移植的远程计算环境及其技术

曾志勇 著

科学出版社

可移植的远程计算环境及其技术

曾志勇　著

科学出版社

北　京

内 容 简 介

面对庞大的数据资源，人们迫切需要借助高性能的计算来解决各种科学问题。由分布式计算演变而来的网格计算就是针对这一需求而发展起来。它通过协调资源的共享和协同作业解决来自动态的多机构的虚拟组织中的问题。本书详细描述了网格及网格计算的发展过程、结构体系、现有工具、运行环境等，并深入探讨了基于 Java 技术构建分布系统的方法，研究了 Java 并行虚拟环境中的协作任务调度模型及算法。

本书既可以为高等院校计算机科学专业高年级本科生以及研究生的学习提供参考，又可以为网格计算及相关领域科研人员的研究提供借鉴。

图书在版编目(CIP)数据

可移植的远程计算环境及其技术 / 曾志勇著.—北京：科学出版社，2011.8

ISBN 978-7-03-031831-2

Ⅰ.①可… Ⅱ.①曾… Ⅲ.①远程网络－研究 Ⅳ.①TP393.2

中国版本图书馆 CIP 数据核字（2011）第 135921 号

责任编辑：杨 岭 朱小刚　　封面设计：陈思思

科学出版社 出 版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

四川煤田地质制图印刷厂印刷

科学出版社发行 各地新华书店经销

2011 年 8 月第 一 版　开本：1/32（880×1230）

2011 年 8 月第一次印刷　印张：5

印数：1-1 000　字数：130 千字

定价：35.00 元

前 言

使用异构网络系统进行高性能科学计算或“通用的”事务处理在过去的几年中已被证明是可行的。但是局域网上的并行计算存在着透明性差和计算容量有限的问题，随着网络技术的不断发展，网络计算已经不再局限于局域环境中，利用广域网络中的计算机完成计算成为新的研究焦点。人们一直在讨论四通八达的计算机网络会给人类社会带来怎样的影响，许多机构都在致力于研究将地理上分布的各高性能计算机进行协同使用，即网格计算。当前已经出现了许多基于广域网的网格计算系统，其中有些研究项目得到了各国政府的资助。

现使用远程资源已经是一个毫无争议的主题，问题是在什么样的层次上和以什么样的方式利用这些分布式的资源。从我国网络基础设施的现实条件出发，我们研究了一类特殊的网格计算，即所谓的 GridRPC 计算。GridRPC 系统不仅把硬件看作是一种可以共享的资源，而且把软件也看作是可以共享的资源。对于缺乏高性能计算机和编写并行软件困难的普遍情况来说，研究怎么样远程使用并行软件库就显得尤其重要。

本书的主要研究内容和贡献如下：

（1）针对局域网上的并行计算透明性差和计算容量有限的问题，本书论述和分析了网格计算(Gridcomputing)的研究现状，并提出我们在目前的现实条件发展远程计算环境的设想。

（2）在广域网上构造计算环境，可移植性是至关重要的，因为要面对的硬件是千差万别。Java 语言及其分布式对象技术 RMI 则以良好的跨平台特性真正地实现了“编写一次即可随处运行”的设想，因此绝大多数研究者都选择 Java 语言作为开发基于广域网计算系统的基础。在分析相关工作优点与局限性的基础上，本书提出了一个使用 RMI 作为中间层、后端使用 Java 消息传递环境的远程计算解决方案：Java 并行虚拟环

境 JPVE(Java Parallei Virtual Environment)。JPVE 支持对象系列化和“动态类载入”，能够实时调整需要、易于维护、灵活、强化了计算手段。JPVE 中分了三类实体，四种应用角色，分工明确，从软件工程和资源管理的角度来说，对并行计算的普及无疑是有利的。

（3）我们实现了一个 JPVE 中的负载监测系统 JSYS。它能够实时测量构成计算环境的网络和节点机的负载状况，并将负载信息通过 API 调用的形式传递给应用程序，供动态负载平衡算法使用。JSYS 体现了动态可扩展性、异构性、一致透明性以及效率的设计思想。

（4）虽然粗粒度的 SPMD 计算可以加速程序的执行，然而考虑处理器对应用程序不同的处理性能可以进一步发掘这种能力。此外程序的响应时间与系统的负载状况有很大的关系，任务调度和数据划分在很大程度上决定并行计算的性能。在这个 JPVE 环境下，本书研究了程序的响应时间与系统的负载状况之间的关系，提出最大异构任务调度算法，它充分考虑了系统对于程序的处理特征及系统的结构和状态，在缩短计算密集型程序响应时间方面取得了较好的效果。

（5）在异构系统并行计算中，通信和计算的能力这两个系统的主要特征对于系统性能的发挥至关重要。但由于通信量度量的不太容易及其引入后任务调度的复杂，所以现在少有考虑通信在调度中的影响。同时，子任务的协同是协作模式并行算法的一个重要特征。本书给出了异构环境下的消息传递并行算法运行时间模型，在这个时间模型的基础上，我们提出了修正的协作任务最优化调度算法。算法不仅考虑到子任务的协同、工作站的计算能力及与此相关的负载平衡问题，而且考虑了网络的通信能力和计算中产生的通信开销，因而更加准确和更加符合实际的应用。

（6）最优化协作任务调度算法时间复杂度较高。本书接着又提出了启发式的算法 HCS，HCS 虽然比传统最优化算法多考虑了通信的问题，但由于采用了启发式算法，时间复杂度并没有因此而增加，朝着实用化的方向迈出了重要的一步。

需要指出的是，内容（4）~内容（6）讨论的数据并行程序都是基于 JPVE 环境供用户调用的并行程序，是一种会反复使用和受系统控制的应用程序 (recurring applications)，在算法方面就可能得到系统额外的信息和一些加强的假设，这是与单纯执行用户提交作业的远程计算所不同的

地方。

最后，本书给出了结论，并概述了今后进一步研究的方向。

本书能够顺利出版，我要衷心感谢我的导师陆鑫达教授所给予的指导；感谢科学出版社对编写工作的大力支持；感谢我的研究生赵文霞、陈光、彭子坤在本书编写过程中所做的大量的文献收集整理工作。由于时间仓促，水平有限，书中错误和不当之处敬请读者批评指正。

谨以此书献给我敬爱的父母、岳父母和我挚爱的妻子、儿子！

曾志勇

2011 年 5 月

目　录

第1章　绪论

1.1　引言

随着科学技术的加速发展，科学工作者要求解问题的规模不断增大，从而需要更高的计算能力即高性能计算。高性能计算的研究水平已经成为衡量一个国家高新科技水平和综合实力的重要标志。利用高性能计算技术可以对所研究的对象进行数值模拟和动态显示，从而获得实验很难甚至无法得到的结果，因此其被广泛应用于航天航空、气象气候、信息安全、生命科学、石油物探、材料工程甚至模拟宇宙演化过程等领域中。高性能计算技术的发展使得计算科学成为人类认识和改造世界的新的方法和途径。它推动了当代科学和高新技术的发展，并将逐渐影响人们的生活方式。

要实现高性能计算，通常有两个途径：设计巨型机或引入分布并行计算的概念。因为前者需开发专用的硬件和软件，所以投资巨大，应用往往局限在军事机构或超大型公司，并且受到工艺水平等众多因素的制约；而分布式并行计算往往可以充分利用现有的资源，且通用性较强，所以应用比较广泛。

所谓分布并行计算，就是分而治之，在不违背前后偏序关系的前提下尽可能让多个任务同时在多台主机上执行。随着网络传输速度的不断提高以及硬件价格的降低，分布并行计算显示出了强大的生命力，如果算法设计合理，其性能可赶上甚至超过某些巨型机。

美国国家超级计算应用中心(NCSA)主任 Larry Smarr 曾指出，当前的计算机界有一种很明显的趋势，那就是利用网络将各种不同的计算机连接起来构成一个虚拟的计算系统来求解复杂的问题[1]。而这样的一个系统必然是异构计算系统。

1.2 异构计算系统

异构计算(Heterogeneous Computing, HC)的概念源于20世纪90年代初期，指由高速网络连接的一系列计算单元（处理机）协同完成某特定任务，使系统开销最小。其中异构是相对同构而言的（同构实际是异构的一个特例），即系统中处理机计算速度、系统结构、负载以及数据传输格式和网络类型可能互不相同，这为并行程序的设计和运行带来了一些问题。

首先是数据编址的问题，比如，x86 系列所采取的小端编址(Little Endian)的方法，即在存放多于一个字节的数据时，将高位的字节放在高址，将低位的字节放在低址。而 Ultra Sparc 芯片，使用大端编址(Big Endian)的方法，将低位的字节放在高址，将高位的字节放在低址。所以如果二者之间通信发送四字节的整数时，通信环境必须进行字节位置的转换，促使被发送的消息在两台机器上的语义一致。

其次，一个并行程序若要在两台不同系统结构的计算单元上运行，则要求程序所使用的通信环境支持两种系统，即通用性。

再次，由于各主机的计算能力不同，必须考虑异构任务的匹配和调度，以减少计算瓶颈、提高计算单元使用效率并提高计算性能。

尽管面临种种挑战，但异构并行计算也孕育着无穷的机会。至少它有如下若干优点：

（1） 使用现有硬件，减少计算成本；

（2） 如果能够合理匹配和调度，可能实现超线性加速比；

（3） 可以利用计算问题本身的异构性实现性能优化，即将特定类

型的运算安排到专用机上运行，提高运行速度；

（4） 虚拟机资源可以随时增减，这有助于采用最新的计算和网络技术。比如，可使用千兆以太网、FDDI(光纤分布数据接口)、HiPPI(高性能并行接口)、SONET(同步光纤网络)和 ATM(异步传输模式)等先进网络技术来提高传输速率；

（5） 程序员可以使用自己熟悉的环境进行程序开发和调试工作；

（6） 应用程序或基本操作系统中可以毫不费力地实现用户级或程序级的容错；

（7） 分布式计算可推进协同工作。

为了有效开发异构系统，不仅需要硬件的支持，更主要的是需要一个通用的网络编程环境，由此产生了 PVM（Parallel Virtual Machine）和 MPI(Message Passing Interface)。

1.3　PVM 概述

PVM 的创始人是 Emory 大学的 Vaidy Sunderam 和 Oak Ridge 国家实验室的 Al Geist，其最初目的是为当时新兴的网络异构计算开发一个研究框架。在 1991 年，PVM 公开发布版本 2.0，随后不断更新，并于 1993 年推出版本 3。版本 3 支持容错和具有更好的移动性。随后推出的版本 3.4 伴有相应的 Java 实现版本。现在已有 3.4.4 版本，使用平台也从纯 Unix 转向兼容 Windows 操作系统。随着 PVM 的成熟与稳定，虽然新版本的更新会逐渐慢下来，但是 PVM 现在仍在继续发展中。可以说，PVM 的出现使分布式计算和网格计算迈出了重要一步。

PVM 具有以下特点：

（1）通用性强，既适用于 TCP/IP 网络，又适用于 MPP 大规模并行系统；

（2）系统规模小；

（3）为所有的并行机厂商所支持；

（4）源代码开放软件，群策群力，不断发展；

（5）技术发展成熟；

（6）应用人员遍布全世界；

（7）一批标准数学软件正在移植到 PVM 平台。

PVM 的关键是虚拟机，即把整个异构计算系统看成一个大的虚拟机。PVM 提供必要的功能使任务在虚拟机上启动执行并支持任务之间的通信和同步。在 PVM 系统中任务的概念和 UNIX 中进程类似，但不完全相同。由于 PVM 提供通信和同步机制，应用程序可以写成一系列子任务以并行执行，通过发送和接受消息，多个任务可以相互协作并行地解决整个问题。

PVM 支持应用程序、机器和网络等各个层次上的异构。换句话说，PVM 允许应用程序利用适合其解的最佳的系统结构组合。如果不同的计算机具有不同的整型、浮点型数的表示形式，PVM 能够处理所有的数据类型转换。另外，PVM 允许虚拟计算机内部由多种形式的网络互联。

PVM 由两部分组成。第一部分是守护进程 PVMD。该进程必须驻留在组成并行虚拟机的每个成员机上。当一个用户欲执行 PVM 应用程序时，他/她可在任何一个主机上执行 PVMD，然后在该主机上定义自己的并行虚拟机系统构成，与此同时该虚拟机上所有成员的 pvmd 都自动相继启动。此后，用户的应用程序就可在任何一个主机下启动执行。不同的用户可以定义不同的虚拟机，彼此之间可以重叠，但它们可相互独立同时运行。

PVM 的第二个部分是系统定义的原语(函数)库。各种原语的功能包括消息传送、进程派生、进程协调同步以及虚拟机组成控制等。应用程序可以有选择地使用最合适的原语编程。同一应用程序的不同部分可以使用不同的程序设计语言(C 或 Fortran77)。从用户角度来看，PVM 软件为程序员提供了最佳的程序设计环境。

1.4　其他软件包——MPI

MPI 是消息传递界面（Message Passing Interface）的简称，它也是一种并行编程环境。为了统一互不兼容的用户界面，1992 年成立了 MPI 论坛（MPI Forum，简称 MPIF），负责制定消息传递界面的新标准，支持最佳的可移植平台。MPI 在标准化过程中吸收了欧美 40 个主要组织的 60 名代表参加，包括研制并行计算机的大多数厂商，以及来自大学、实验室与工业界的研究人员。正式的标准化过程是从 1992 年 4 月由并行计算研究中心（The Center for Research on Parallel Computation）支持召开的研讨会上开始的。在会上讨论了作为一个标准消息传递界面所具有的基本特征。当年 11 月发表了草稿（1993 年 2 月修改），1994 年发布了 MPI 的定义与试验版本 MPI1.0，1997 年发表了 MPI2.0，2008 年结合以前的版本发表了 MPI2.1。

MPI 的目标是要开发一个广泛用于编写消息传递程序的标准。要求用户界面实用、可移植、高效、灵活，能广泛用于各类并行机，特别是分布式存储的并行机。近 10 年来每个计算机厂商都在开发标准平台上作了大量的工作，出现了一批可移植的消息传递环境。MPI 正是吸取了他们各自的有益经验，同时从句法与语法两方面确定核心库函数，使之能适用于更多的并行机。

MPI 的完整目标要求如下：

(1)设计一个应用程序设计接口（并不一定要实现编译器或系统调用库）。

(2)允许有效的通讯，避免内存到内存的拷贝，允许计算与通信的重叠以及进行处理器通讯，如果可能的话。

(3)允许实现异构环境下的版本。

(4)允许便利的 C 与 Fortran77 的绑定。

(5)假设一个可靠的通信接口：用户不必处理那些通信错误，这些错误应由底层通信子系统处理。

(6)定义一个接口，使之与目前的一些应用类似，如 PVM，NX，Express，P4 等，并提供扩展使之有极大的灵活性。

(7)定义一个接口，使之可以在许多厂商的平台上实现，而不必在底层通信的系统软件上进行大的改动。

(8)接口的语义应是独立于语言的。

(9)接口应设计为线程安全（Thread-safety）的。

如同 PVM，MPI 也支持点到点的发送和接收，及组的概念。MPI 特别在多点之间的集群通信上作了很多工作。MPI 中的集群通信方式包括：广播（broadcast），收集（gather），发散（scatter），同步（barrier）以及全局归约运算（reduce operation）。MPI 完全支持异步发送和接收，即非阻塞式（non-blocking）发送和接收，从而真正实现计算与通信的重叠。MPI 在各自互不干扰的通信上下文（communication context）中通信。通信上下文可以合并，分割和重组。不过，MPI 1 中不支持动态计算进程的生成，但这已在 MPI 2 中弥补。

作为一个新标准，MPI 吸收了大量消息传递环境如 Intel NX/2，Express，nCUBE's Vertex，p4 和 PARMACS，以及 Zipcode，Chimp，PVM，Chameleon 和 PICL 的优点，将之融合在一起，并在某些方面做了优化和改进。虽然 MPI 推出才几年，但在国际上已普遍使用，且用户与使用 PVM 的用户相当。

参考文献[2]详尽地讨论了 PVM 和 MPI 在特征上的差异。

1.5 存在的问题

PVM 和 MPI 代表了目前并行程序设计环境的主流，无数的并行程序是基于这两个工具而开发的，同时，也有无数的计算机系统支持这两个工具，包括各种超级计算机、UNIX 工作站群和 NT 工作站群。但是

他们使用起来并不方便，有着自身不可克服的缺点：首先，透明性和灵活性差，编写起来较为困难，表现为：

(1)用户必须了解运行环境的网络结构，包括各个运行结点机器的 IP 地址和域名。

(2)用户必须在运行程序的各个结点机上拥有账号。

(3)用户必须针对不同的系统结构对同一程序进行多次的编译，以产生相应系统结构上可以运行的代码。

(4)用户必须手动地将代码拷贝到各个参与运算的结点机器上。

(5)在 UNIX 工作站群上运行时，用户必须建立各个工作站之间的信任关系，这样就带来较大的安全隐患。

(6)缺乏负载平衡和任务调度机制，用户要对参与计算的进程的分布进行一定干预和指导。

(7)由于异构系统上代码和数据格式的不同，很难进行当系统配置改变或负载重大变化时的进程迁移，因而这种在同构系统上成功的动态负载平衡方式在这里实现起来非常困难。

其次，由于 PVM 和 MPI 由 C 或 C++实现，提供 C、C++、FORTRAN 接口，使得用 PVM 和 MPI 开发的并行程序不能在异构系统中做到无缝移植、运行，这又表现为：

(1)需要对被移植并行程序之源代码进行显式编译以生成对应于特定系统的执行代码。

(2)被移植的并行程序的源代码有可能需要做少量修改方可在特定的系统上通过编译。

(3)当全新的系统出现时，需要将 PVM 和 MPI 移植到新的系统上方可将原有的并行程序移植到新的系统上。

最后，鉴于局域网络的容量有限，Cluster 虽然提供了相当高的计算能力，但是相比与广域网络上无限待开发的计算潜力来说还是很小的。