计算网格的中间件*)

The Middleware in the Computing Grid

武秀川[1,2]　鞠九滨[2]

(烟台大学计算机学院　烟台264005)[1]　(吉林大学计算机学院　长春130023)[2]

Abstract In the future, network computing will proceed within a large-scale high-performance distributed computing environment, so-called Computing Grid. It can integrate a numerous of global-wide distributed heterogeneous resources including networks, computers, peripherals, information and other kinds of resources, and supports their co-ordinate use. Computing Grid will provide users with services of resources management, access to data, login and authentication and security etc. The key to build Computing Grid environment is to develop middleware (software). Applications built on the middleware can access resources in the Grid and finish effective co-ordinate computing task very easily. In this paper, features of middleware are analyzed and compared, and some suggestions of research are also given in the paper.

Keywords Computing grid, Distributed computing, High-performance computing

因特网的出现使得数以千万计的各种类型的计算机连接组成一个集成的分布式系统,从而可以为科学计算、商业、教育及信息服务、娱乐等提供大量的应用服务。但人们一直用因特网只进行电子邮件及软件等的通信和信息检索,很少使用它进行真正意义上的计算。随着先进的网络技术特别是高速网络技术和先进的计算结构的出现,使得有可能建造一个大规模的高性能分布式计算环境——计算网格[1,2]。

计算网格的研究在计算机科学发达的西方国家仍处于实验性阶段,而在我国则刚刚起步,属于高性能网络计算的前沿课题,具有重要的理论价值和广泛的应用前景。

本文对代表性的计算网格中间件进行分类对比分析,提出有待进一步研究的若干问题。

1　计算网格与中间件

计算网格是一种硬件和软件的综合体系结构[2,3]。从硬件观点看,一个计算网格是地理上分布的异构的和动态的各种高性能计算资源,包括远端计算机、网络、存储装置、各种科学仪器、可视及虚拟现实显示设备以及个人计算机等资源的组合。从软件观点看,计算网格是一个中间件[1,2,4],它集成上述资源,使其变为用户桌面上的功能非常强大的一个独立的计算机资源,从而使用户可以不受地理边界限制,透明地、无缝地、有效地使用该资源,以解决目前仅靠本地资源不可能解决的各种复杂问题。采用计算网格,将不再仅局限于客户/服务器模式,而使一种全新的面向计算[1]的应用成为可能,并最终主导网络高性能计算,就像目前的 WWW 支持面向信息的应用开发一样。计算网格类似于电力网格,用户访问计算网格中的任何资源,就好像我们把各种电力设备插头插入到墙上的插座去使用电力网格提供的电力一样简单。上述的计算环境有时也称为超级系统或元系统,在其上进行的计算也叫超级计算或元计算[1,5,27,34]。

计算网格系统的主要功能是提供给网格用户一个透明的、分布式的、共享的、安全和容错的高性能计算环境,在此环境中的用户能够共享文件、计算资源对象、丰富的数据信息以及昂贵的仪器设备,不必由用户自己决定在何处执行自己的程序以及进行必要的程序和数据文件的拷贝等操作,完全由计算网格系统自动完成。由于系统把用户置于一个相同的虚拟环境中,因此可以更有效地实现异地、跨学科的不同用户的协同工作。另外由于在计算网格系统环境中的程序是并行运行的,并使用离线网站资源,因此可以具有更高的应用性能。与 rlogin 及 telnet 使用远程计算资源的方式不同,计算网格系统也将提供一个简单的程序设计环境和模型,最终导致用户获得更高的编程生产率。

计算网格系统的研究具有极重要的意义。美国自然科学基金资助的 PACI 项目的两个重要组成部分 NCSA[28] 和 NPACI[29],旨在建立一个计算网格基础设施联盟,以大力促进科学和工程技术研究的发展。而 NASA 正在构造的计算网格样机 IPG[30] 其目的是为了解决目前无法解决的科学与工程计算以及数据管理问题。欧洲计算网格论坛[31] 也在开展计算网格的研究。我国的中科院、国防科技大学及江南计算所等单位联合开展了国家高性能计算环境 NHPCE[32] 的研究,目前包括了北京、上海、西安、长沙、合肥及成都等实验性网格结点,NHPCE 的初期目标是在全国范围内建立由多个高性能计算机结点组成的计算网格系统,开发网格系统软件 GridWare 和对若干重大行业包括生物计算、气象预报、石油油藏模拟、科学数据库等具有重大生产意义的系统应用软件。NHPCE 的长期目标是提高计算网格系统的性能、可扩展性及可用性。由国家教育部支持、清华大学主持的先进计算基础设施北京上海试点工程[33] 重点项目也是一个试验性的计算网格系统,旨在为跨地域、跨学科的教学、科研合作和人才培养提供可共享的高性能的计算网格资源。

计算网格作为一个硬件集已经出现了至少十年,但从软件中间件层的观点,可把计算网格看作为一个分布式的服务及其提供者的集合[27]。计算网格所要解决的关键问题就是要建立一个中间件[4,6,26],使用户可以借助于中间件所提供的一

*)国家自然科学基金资助项目(69873018)。武秀川　博士研究生,副教授,研究领域为分布式系统和计算机网络。鞠九滨　教授,博士生导师,研究领域为分布式系统和计算机网络。

组通用的分布式服务建立自己的系统。中间件所集成的服务包括[2,5,7]因特网范围内的动态的资源管理(资源的发现、定位、分配及其调度),安全服务(认证、授权和保护),通信服务,对进程、文件、数据库、共享地址空间进行管理,命名、复制、迁移、容错等服务,以及记帐管理等功能。计算网格中间件为广域应用提供操作系统一级的服务。中间件所提供的是一个一致性的和统一的模式,即用户明确地知道如何在中间件上组织建立自己的应用程序。

本文对目前几种具有代表性的中间件系统进行分析比较,依据这些系统的构造原理的不同,将它们划分为面向对象的、基于 WWW 的、基于服务包的和基于 Agent 的四类计算网格中间件系统。

2 面向对象的中间件系统

Legion 和 Globe 是这类系统的典型代表。Legion[10]的每个组件用一个对象来表示,每个对象属于一个类,而每个类本身是一个 Legion 对象,所有这些 Legion 对象导出一组对象代理成员函数,这些函数实现关键的 Legion 服务,而 Legion 类对象导出一组另外的类代理成员函数,这些函数可以管理对象类的"实例"。Legion 通过使用三级命名体系结构识别 Legion 对象[9~11]。在最高级,对象由用户定义的称为上下文名(context name)的文本串来标识,例如表示一个处理机资源的对象可能被赋予一个上下文名,它相应于那个主机的标准 DNS,这些用户级上下文名字被环境空间目录服务映射为一个位置独立的 Legion 对象标识符 LOID。对于直接的对象到对象的通信,LOID 被绑定进程映射到对象地址 OA,在该对象地址内使用传输协议进行信息传递。实际上 OA 是一个地址语义域表,旨在封装各种形式的广播和复制通信。上述关键部分是 LOID,提供基本的系统级的命名抽象,每一个对象都被赋以一个唯一的 LOID,用于与其他的对象进行通信。

Legion 所提供的共享对象空间支持对数据文件共享的和安全的访问,而不必像 NFS 那样需要超级用户的设置。显然,共享对象空间是比共享文件功能更强大的模型,它不仅仅是共享文件,所有的实体包括程序、数据库、仪器设备等都可被命名并且在用户之间共享。一个文件是一个支持标准文件操作的文件对象。另外,文件对象接口也定义其他的特性如它的错误处理、同步性以及性能特征。由于并非所有的文件必须是相同的,这就消除了对所有文件提供 UNIX 同步语义的必要性,而可以按照每个文件来确定,甚至可以随着时间变化。

通过为所有的应用和服务定义这样一种公用的对象模型,Legion 提供更为直接的服务,如在系统级代理,调度程序能够融合进 Legion 中,就好象通常的应用进程利用标准的对象代理接口一样。

Legion 也支持面向过程的一类高级语言的使用。它提供给程序设计者一个连接到系统的高级程序设计语言接口,该接口与支持和使用 Legion 运行库的 Legion 编译程序相连接,这样便可实现一个完整的 Legion 对象。

Legion 的最大特点是每个 Legion 对象由它的类对象定义和管理,而类对象又可以生成新的对象实例且可以激活、运行或把它与某客户机相连进行通信,因此类对象起着很重要的作用。Legion 允许用户定义和建立他们自己的类对象,并且能够决定甚至改变 Legion 支持对象的系统级的机制,这使得 Legion 用户在设计自己的应用程序时具有很大的灵活性。

Globe[4,6]是另一个面向对象的中间件平台。它的最突出特点是通过对复制的灵活支持,以实现中间件的功能。Globe 主要由一个对象模型以及一组基本的支持服务模块所组成。该对象模型称为分布式共享对象 DSO[6,12~14]。DSO 允许构造全球范围的可扩充的和可由大量的进程共享的对象,支持的服务主要包括命名和定位对象等。

一个 DSO 可以分布在若干个地址空间中,在每个地址空间,DSO 由一个本地代表所表示,进程通过 DSO 实现交互操作以及进行通信。每个本地代表由语义、通信、复制和控制四个子对象[6]所组成,它们分别实现完成 DSO、通信、一致性地复制对象和控制来自于客户进程的调用等功能。每个 DSO 提供一个或多个接口,每个接口由一组函数组成。DSO 的状态(即数据结构)可同时在多个机器上分布或复制,然而由于对象完全封装了状态及在该状态上的操作,因此进程完全不必关注这个行为。所有的实现细节包括通信协议和复制策略以及状态分布和迁移等都被隐含在它的接口后面。

DSO 使得在计算网格环境中尽可能地从逻辑上实现本地化。采用 DSO,使得 Globe 系统不再基于单个服务器、多个客户机的模式。除此之外,基于上述概念的命名服务允许用户发现、访问和共享分布式资源,解决了 CORBA、DCE 以及 DNS 中的名字解析只能在存储它的特定的、静态的位置进行的问题,进而对中间件系统的其他组件也可进一步扩充。

比较 Legion 和 Globe 可以看出它们有一些本质的不同。Legion 的对象表示资源并且是一个主动式实体,如一个对象可能是一个运行在独立的地址空间上的进程,认为对象通常都是驻留在一个地址空间上。Globe 把一个对象用复制的方法从物理上分布在多个资源上,每个对象提供一个或多个接口,而每个接口由一组方法函数组成,尽管它的对象是被动的,但多个进程可以同时访问同一个对象。两个系统不同的对象观点导致了不同的内部对象通信机制:Globe 是把对象的一部分(即本地对象)加载到调用者的地址空间,而 Legion 是采用非透明的消息传递方法。尽管有上述不同,但 Legion 和 Globe 都具有一个统一的对象模型和体系结构,并且都使用类对象高度抽象实现的细节,更为重要的是它们都具有很好的可扩充性即支持实现的灵活性,正是对象模型提供了自然的方法使它们获得这种灵活性。如在 Legion 中,考虑到并不存在一个能令所有用户都满意的安全策略,因此 Legion 没有提供一个固定的安全策略,而让用户根据自己的实际需要做出某种折衷后实现自己的安全策略或经过对象的继承使用现有的策略。出于同样的考虑,Legion 从调度到容错都提供给用户每一个机会,所以 Legion 的关键模型是完全可扩充的。而在 Globe 中用一个标准的接口把复制算法封装在一个子对象中,当在需要时可以容易地改变复制算法。由此我们可以看到面向对象的技术在中间件系统中的生命力。

3 基于 WWW 的中间件

随着因特网及一系列协议和标准的发展,WWW 得到了空前广泛的使用。虽然通过使用 Web 浏览器连接到远程服务器的最主要目的是为了获得对信息的访问,但信息并不是唯一的资源,WWW 体系结构提供了访问远程高性能软硬件资源的极大潜能,因而相继出现了一些基于 WWW 技术的计算网格中间件系统,SuperWeb 和 WebOS 是目前这一类系统的代表。

SuperWeb[15]由三部分组成,即主机、客户机和代理。主机提供资源如 CPU、内存和磁盘空间等。客户机提出使用资源

的请求，代理发现并提供给客户机所需要的资源。但主机和客户机的作用并不固定，当一个机器空闲时可以作为一个主机，也可在完成某种计算而需要另外的资源时作为一个 SuperWeb 客户机。SuperWeb 系统中的关键是计算代理，它收集和管理资源，满足用户对这些资源的请求服务。一个用户（主机或客户机）通过注册或请求资源两种方法与计算代理交互作用。而注册和请求资源都只是简单的访问计算代理的不同的 Web 页面。如果一个客户机提交了多个任务，计算代理用循环等算法把任务传送到已经注册的多个主机上去。为了避免在计算代理中的争用，它并不实际加载客户任务，而只是把指向该任务请求客户机的 URL 指针送给相应的主机，主机则直接从所选择的客户机上接收任务，最后由主机直接把结果送回到客户机。显然，这种结构的关键是主机软件完全运行在用户层，仅要求注册的机器具有一个 WWW 浏览器以及运行 Java 即可，同时它明显地意味着客户机可潜在地获得多种不同的计算平台。

在 SupweWeb 的工作模式中，客户机应用并未只限制在串行计算，客户也可提交并行或分布式任务，由计算代理的调度程序把任务映射到多个处理机或是紧耦合的工作站网络或是大量的分离的资源上，依赖于用户请求。

从任一客户机把任务提交到任一主机，在因特网这样的异构环境中，客户机和主机可能具有不同的指令集、不同的操作系统、不同的字长度表示等，因此任务加载就成了计算代理所要解决的关键问题之一。在 SuperWeb 中，用 Java 的 Java-Bytecode 这样一个独立于机器的语言成功地解决了任务加载。SuperWeb 适合于非通信密集型的网格计算问题。

WebOS[17]是另一个基于 WWW 技术构建计算网格的较为成功的例子。“智能客户机”(Smart Client)在 WebOS 中起了重要的作用。它把服务器的某些功能以及状态迁移到某些空闲的客户机上以改进总的服务质量。智能客户机由两个相互协作的小程序组成。图形用户接口小程序提供服务接口并把用户请求传递给管理者小程序。对于每一个请求，管理者小程序根据某种服务信息（如服务器的硬件特性、目前的负载情况、有效性、处理器速度及网络连接速度等）选择一个最佳的服务器，以满足客户需求。根据这种负载平衡算法，结合服务器是否活跃或超时等决定服务器是否失效。一旦失效，再调用负载平衡算法选择另一个服务器。智能客户机从逻辑上把指定的服务器功能迁移到客户机上，从而动态地实现了负载平衡和透明的容错功能。这种透明性包括不必对服务器的拓扑结构作任何限制，可以根据指定的服务方式选择指定的服务器，从而比其它方案更能有效地提高整体服务性能，更加智能化。采用智能客户机还可组成一个 Rent-A-Server[17]结构，可有效地在多个服务器之间分布负载。目前所采用的方法或者由于用户过多的参与或者受到缓存一致性协议的限制，使得响应较差，或者要求每个站点购买足够的计算资源和网络带宽来满足峰值要求。用 Rent-A-Server 方法，一个超载的 HTTP 服务器能够把负载卸载到空闲的、离请求服务的客户机距离最近的服务器上去，而该服务器使用 WebFS 系统缓存（一致性的）来自于超载服务器上的数据。采用 Rent-A-Server 可以获得透明的、自动的、有效的和动态的资源补充。因特网上的任一网站仅需要足够的硬件资源处理它们的平均负载，过载时能被动态地卸载到一个或多个 Rent-A-Server 上，进而处理峰值要求。同时由于智能客户机与负载守护进程的合理使用，为应用程序的开发提供了方便的接口。

WebOS 实现了自己的广域文件系统 WebFS 和安全认证系统，并且为每个运行的进程形成一个虚拟机。这些虚拟机与 CRISIS 安全系统相互配合，实现安全的远程进程控制，从而使得用户可以访问远程高性能硬件及计算资源。

4 基于服务包的中间件

Globus[5,7]是目前全球功能最为完善、最具中间件特点的一个系统。Globus 是一种基于“服务包”而具有“服务总和”特性的一种体系结构。它使用一组已经存在的组件组合为一个复合的超级计算工具包。工具包定义一个计算网格所要求的服务及性能。如 MDS 组件基于 LDAP，提供对系统状态信息的分布式访问。目前实现的还有提供资源定位和进程管理(GRAM)以及通信、安全等其它服务的工具包组件。每个工具包组件由一组核心服务所组成，每个核心服务定义一个应用程序接口，对本地服务提供一个统一的接口，高级服务通过该接口使用这些核心服务实现更为复杂的全局功能。例如资源代理和协作分配程序使用由 GRAM 和 MDS 所提供的功能确定某些计算当前可获得的资源。因此当构建一个网格应用程序时，用户可从所提供的 Globus 服务包中选择合适的组件，不必全部重写应用程序。如对通信组件的调用，在局域网内通信则可用 TCP/IP 协议，在广域环境中则选用 ATM 可能更为有效。对通信组件的低级机制的选择可能影响网络的服务质量，进而对应用程序的性能有很大的影响。因此 Globus 工具包模块通过接口也可让高级工具和应用程序进行选择，从而具有极大的灵活性。这些接口提供基于规则的选择、资源特性查询以及通知机制，而建立于 Globus 工具包之上的某些高级服务和应用程序能够使用接口提供的这些机制对可获得使用的资源进行有效的配置并能适应运行期间资源的数量和服务质量的动态变化的情况。如某应用程序在某计算机上执行计算并把数据通过广域网送到远程昂贵的显示设备上。在开启时该应用决定可用的资源量和根据网络性能合适地配置其计算和通信结构，如决定在传输过程中对某些数据实施压缩，而其他数据不压缩。在运行期间，通知机制允许该应用适应网络服务质量的变化。

利用 Globus 所提供的低级工具包，可以构造高级组件以生成应用级接口，应用程序开发者可以使用这些接口灵活、方便、简单地开发自己的各类网络计算程序，仅使用一组单一的低级机制就可在多个平台上构造不同的服务，最终使得应用能够适应异构的和动态变化的计算网格环境。

目前已用 Globus 实现了两个最大的计算网格原型 I-WAY 和 GUSTO。I-WAY 连接了17个超级计算机中心、5个虚拟现实研究网站以及60多个应用组，成功地实现了分布式虚拟现实环境共享应用和分布式超级计算以及分布式广域资源管理和调度，充分显示了计算网格在大型分布式协作计算中的优越性。

但 Globus 缺乏一个通用的程序设计接口和模型，当需要为 Globus 添加一个新的功能时，首先需要为该功能设计一个接口，大大地增加了应用程序设计者的负担，很显然，当设计者试图巧妙地处理几个不同的接口和完全不同的系统时，在 Globus 中将变得比较困难。另外在用户（或机器）申请资源时，为实现资源的保护，Globus 中需进行资源申请方和提供方的相互认证，因此在计算网格这样的环境中可能需要频繁地在网络上传输双方标志身份的如证书等信息，某种程度上降低了系统的性能。

5 基于 Agent 的中间件

中间件系统 Netsolve[20]和 AppleS[21]，采用一种新的观念，即在中间件中嵌入 Agent，这种中间件结构本身呈现为一种 Client-Agent-Serve[20]结构。如 Netsolve 系统结构如图1所示。图中的阴影部分即为 Netsolve 中间件，顶层的 Netsolve 客户机库与用户的应用程序相连接。用户的应用程序调用 Netsolve 的应用程序设计接口 API，通过 API 用户获得对 Grid 资源的访问。Netsolve 客户机接口支持对系统的同步调用和异步调用。Netsolve Agent 作为一个数据库通过使用网络状态服务(NWS)维持有 Netsolve 服务器的有关信息如硬件特性和所驻留的软件以及一些如网络连接性、延迟和阻塞等的动态信息。Agent 同时作为一个资源代理，对于每个 Netsolve 的用户请求，Agent 使用上述有关信息，确定一个或一组可为用户请求提供最佳服务的服务器，并且由 Agent 实现容错功能，当服务器失效时，由 Agent 对用户透明地选择另外的服务器继续完成用户提交的任务。为此可把 Agent 看作为一个 Netsolf 系统网关。Netsolve 服务器是一个守护进程，它等待客户机的请求，服务器可以运作于单个工作站或工作站机群或 MPPs 上。

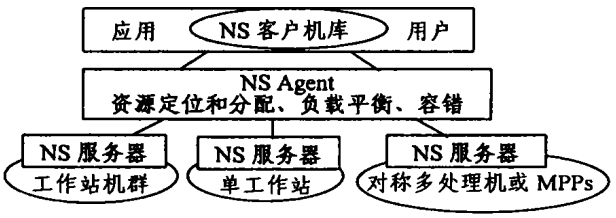

图1 Netsolve 系统结构

当用户需要完成计算任务时，向 Netsolve 客户机提出请求，该客户机与距离它最近的 Agent 通信，Agent 根据用户需求选择最合适的服务器完成计算，当任务完成后把结果直接传送给客户机，并通知 Agent 任务完成。

上述这种中间件结构是一个增强的 Client-Server 结构，或可看作为一个增强了功能的 RPC 环境[20~24]，但它是在逻辑上分离的三层结构，使得其中的每一层都可以独立于其他两部分完成自己的工作。其中的客户机仅是简单地确定用户提出的问题。在这种环境中，用户面对的是"单机"模型，而所需的并行环境的设置、数据的分布、服务器的定位及分配等问题均由 Agent 和服务器去处理。由于服务器可以工作在各种环境下，因此能够针对其具体环境实施优化，即尽可能充分地利用服务器所处的不同的各种平台特性，完全不必有用户的参与。而 Agent 作为一个资源代理所完成的任务是使得客户机仅集中于确定问题，而服务器仅集中于处理问题，双方不发生任何联系。

结束语 中间件尚有许多问题值得进一步研究：

·灵活的实现体系结构。如用户为了解决大的并行计算问题，可以非透明地由用户选择网格环境中的并行计算机。当带宽缺乏时可把数据和计算从服务器移到客户机。即中间件应具有一组进行可重用设计的协作机制。

·提供合适的程序设计模型。目前在全球网格计算中占主导地位的 Legion 和 Globus 系统可以相互补充。因为 Globus 主要提供低级服务，而 Legion 则主要提供高级程序设计模型，如果用 Globus 服务实现 Legion 的对象模型则是非常简洁的方法。另外在因特网上已经成功地构造了"对象网格"(如 CORBA 对象)和"数据网格"(如 Web)。在设计构造中间件时完全可以采用这些成熟的技术。如在 Globus 的数据访问中可以经由 CORBA 实现，同样可用 MPI 或 PVM 实现信息传递，用 DCE 实现远程过程调用，用 Condor 和 Nimrod 实现高吞吐量计算。除此之外还可以使用其他的中间件如 Globe 来形成 DSM。所有这些手段都可以简化计算网格应用程序的开发，关键是进行这些技术和中间件系统的有效的接口设计。Globus 项目组正在开发 CORBA/G(Grid)接口，就是想要达到上述目的。

·提供具有容错能力的全局目录访问协议。目前的中间件系统或者采用集中式的目录结构(如 Globus)或者是采用固定的总目录服务器的方式(如 Legion)，导致容错性能较差。

·应提供一个适应性的广域资源环境以支持适应性的资源调度策略。在计算网格这样一个高度动态变化的环境中，目前的中间件系统中信息服务功能所提供的可能是已过时的系统信息，因此对基于此信息作出的资源调度策略会产生不良的影响。

·嵌入移动 Agent，可以大大地提高中间件系统的灵活性和有效性。因为除了具有 Agent 的特性外，移动 Agent 还具有移动性及其他许多优越的性能[25]，把它作为计算网格环境中一个可以迁移的程序能在网络上从一个主机移到另一个主机，从而为许多应用提供有效的选择，尤其是改进网络延迟和带宽及降低网络的脆弱性。目前已有不少成功的移动 Agent 系统，如 Telesscript、Agent TCL、Obliq、Omniware、TACOMA、ATRIL 和 Mole 等。如何把移动 Agent 嵌入到计算网格中间件结构中是值得研究的问题。移动 Object 和移动 Agent 将构成未来分布式计算的主体[16]。有人提出了未来移动式中间件的需求抽象[18]。

计算网格是互联网络深入发展的必然结果，也是分布式系统和并行系统以及机群等系统相结合的产物。有人预测计算网格必将成为21世纪全球计算机工业乃至经济发展的巨大推动力，因此将导致全球范围内的计算网格的全面和深入的研究。目前对于计算网格的研究主要侧重于其组织体系结构(硬件)的建立以及计算网格在某些专门领域的应用方面，而对其核心中间件部分中包括分布式、并行算法在内的计算网格算法以及资源调度、管理和安全保护及通信和与之密切相关的因特网门户(Internet portal)技术、支持移动对象的分布式对象系统等相关内容的研究则相对滞后。

参 考 文 献

1 Dongarra J, et al. High Performance Computing Today. To Appear in FOMMS 2000: Foundations of Molecular Modeling and Simulation Conference

2 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman, 1999

3 Bresnahan J, et al. Communication Services for Advanced Network Applications. Proceedings of the International Conference on Parallel and Distributed Processing Technique and Applications 1999, IV: 1861～1867

4 Ballintijn G C, et al. Scalable Naming in Global Middleware. In: Proc. 13th Int'l Conf. On Parallel and Distributed Computing System(PDCS-2000). Las Vegas, Aug. 2000. 624～631

5 Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit. International Journal for Supercomputer Applications,

1997,11(2): 115～128
6 Van Steen M,Homburg P,Tanenbaum A S. Globe: A Wide-Area Distributed System. IEEE Concurrency, Jan. 1999. 70～78
7 Foster I,Kesselman C. The Globus Project: A Status Report. In: Proc. of the Heterogeneous Computing Workshop. IEEE Computer Society Press, 1998. 4～18
8 Grimshaw A S,et al. Archtectural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems: [Technical Report CS-98-12]
9 Grimsshaw A S, et al. Metasystems. Communications of the ACM, Nov. 1998. 46～55
10 Grimshaw A S,et al. Legion: The next logical step toward the world-wide virtual computer. http://legion.virginia.edu/papers.html
11 White B S,et al. Grid-Based File Access: The Legion I/O Model. http://legion.virginia.edu/papers.html, 2000
12 Bakker A,et al. The Globe Distributed Network. In: Proc. 2000 USENIX Annual Conf. (FREENIX Track), San Diego, 2000. 141～152
13 Hanle C,Leiwo J,Tanenbaum A S. A Security Architecture for Distributed Shared Objects. In: Proc. 6th Annual ASCI Conf. 2000. 350～357
14 Pierre G,et al. Differentiated Strategies for Replicating Web Documents. In: 5th Intl. Web Caching and Content Delivery Workshop, Lisbon May, 2000
15 Alexanddrov A D,et al. Superweb: towards a global web-based parallel computing infrastructure. In: 11th Intl. Parallel Processing Symposium, April 1997
16 Lange D B. Mobile Objects and Mobile Agents: The Future of Distributed Computing? In: Proc. of The European Conf. on Object-Oriented Programming '98, 1998
17 Vahdat A,et al. WebOS: Operating system services for wide area applications. In: Proc. of the 7-th IEEE Intl. Symposium on High Performance Distributed Computing, Chicago Illinois, 1998. 28～31
18 Efstratiou C,et al. Architectural Requirements for The Effective Support of Adaptive Mobile Applications. Middleware 2000 Technical Program
19 Vahdat A,Eastham P,Anderson T. Turning the Web into a Computer:[Technical Report]. 1997
20 Arnold D C,Bachmann D,Dongarra J. Request Sequencing: Optimizing Communication for the Grid. In: 6th Intl. Euro-Par Conf. Munich, Germany, Sep. 2000
21 Berman F, Wolski R. The Apples Project: A Status report. In: Proc. of the 8th NEC Research Symposium, Berlin, Germany, May, 1997
22 Arnold D C,Dongarra J. The NetSolve Environment: Progressing Towards the Seamless Grid. In: 2000 Intl. Conf. on Parallel Processing (ICPP-2000), Toronto Canada, Aug. 2000.
23 Plank J,et al. Deploying Fault-tolerance and Task Migration with NetSolve. submitted to International Journal on Future Generation Computer Systems, Elsevier Pub. 1999
24 Casanova H, et al. Adaptive Scheduling for Task Farming with Grid Middleware. International Journal of Supercomputer Applications and High-Performance Computing, 1999,13(3):231～240
25 Lange D B,Oshima M. Seven Good Reasons for Mobile Agent. Communications of the ACM,1999,42(3):88～89
26 Lindahl G,et al. Metacomputing-What's in it for me? http://legion.virginia.edu/papers.html
27 Brown M D,et al. The International Grid(iGrid): Empowering Global Research Community Networking Using High Performance International Internet Service. http://www-fp.mcs.anl.gov/~foster/papers.html
28 NCSA Project: http://www.ncsa.uiuc.edu/
29 NPACI Project: http://www.npaci.edu/
30 http://www.nas.nasa.gov/IPG, 1999
31 European Grid Forum: http://www.egrid.org/
32 Chinese NHPCE Project:http://www.grid.ac.cn
33 Chinese ACI of MOE Project: http://aci.cs.tsinghua.edu.cn
34 Luthi, Hans Peter. Metacomputing, an emerging technology? Computer Physics Communications, 2000,128(1):326～332

(上接第15页)

小结 ACI系统作为网格计算系统的实验床,许多问题还处在探索阶段。在已经实现的系统中,网格计算的优势已经有所显现,当前已经有许多大专院校和科研部门的科研工作者在这个实验床上进行了许多有益的尝试,并且取得了令人满意的结果。在以往的高性能计算研究中,功能强大的超级计算机通常都是位于实验室中,由于其使用的复杂性以及高昂的价格,许多急需大量计算资源的问题无法在高性能计算机上加以运行,从而造成了资源的极大浪费。ACI系统将许多高性能计算机通过互联网连接起来,进行统一调度,并且为使用者提供了单一友好的用户界面,使得用户可以远程访问计算资源而无需知道该资源位于何处,大大简化了进行高性能计算的步骤并且在很大的范围内推广了高性能计算。今后,ACI系统将继续发展,主要目标是进一步提高系统效率,加快资源的访问时间,并且争取更多的高性能计算机加入到这个系统中来。

参考文献

1 Atiquzzaman, Mohammed, Srimani, Pradip K. Parallel computing on clusters of workstations. Parallel Computing,2000,26(2-3):175～177
2 Foster I,Kesselman C. Computational Grids: The Future of High-Performance Distributed Computing. Morgan Kaufmann Publishers, 1998
3 http://www.npaci.edu
4 http://www.ncsa.edu/
5 http://www.cs.man.ac.uk/ipg/
6 http://www.cs.sandia.gov/discom/
7 李伟,徐志伟,唐志敏.国家高性能计算环境的设计与实现
8 http://www.ncftp.com
9 IBM 公司的 XML4J 项目. http://www.alphaworks.ibm.com/tech/xml4j
10 Hariri S,et al. A message passing interface for parallel and distributed computing. High Performance Distributed Computing, 1993. In: Proc. the 2nd Intl. Symposium on , 1993. 84～91
11 Foster I,Roy A,Sander V,Winkler L. End-to-End Quality of Service for High-End Applications. IEEE Journal on Selected Areas in Communications Special Issue on QoS in the Internet, 1999