先进计算基础设施中结点内部的任务调度和监控

Task Scheduling and Monitor of Cluster in the Advanced Computational Infrastructure

方 兴 李三立 王小鸽 黄震春 何 川

(清华大学计算机系微机组 北京100084)

Abstract Advanced Computational Infrastructure (ACI) is a high-performance computational grid, which is based on shared high-performance computational resources and it is to solve the large-scale applications. The computational resources in ACI are dynamic, distributed and heterogeneous. This article introduces the architecture of ACI and emphasizes the task scheduel and moniter of cluster in ACI. We use XML to describe the resource in system, use a node server to control the cluster and use TCP/IP to communicate which is proceeded distributed. A schedule strategy and task priority are considerd to schedule tasks. The cluster and tasks are monitored when system is running.

Keywords GRID, ACI, XML, Task schedule

一、引言

在当前的高性能计算研究中,网络并行超级计算系统(集群式计算系统,又称 cluster[1])是国内外研究的热点。网络并行超级计算可以以很低的价格提供高性能计算,能满足大多数用户的要求,特别适合在国内推广和使用。

目前对高性能应用的需求要求越来越高,仅靠一台高性能计算机已经无法完成某些超大规模应用问题。这就需要将分布在不同地区、不同体系结构的高性能计算机通过高速网络连接起来,以便满足某些大型应用问题的需要,这种技术就是计算网格技术(Computational Grids[2])。先进计算基础设施(Advanced Computational Infrastructure,以下简称 ACI),是用高速网络将处于不同地理位置、拥有不同计算资源和不同体系结构的高性能计算机及其他贵重仪器连接起来,为广大科研工作者提供一个高效、易用的高性能计算平台,并以此为依托,建立跨学科、跨地域的科学技术研究合作。

在国内外,关于计算网格的研究都受到高度重视。在美国有 NSF 支持的 NPACI[3](National Partnership for Advanced Computational Infrastructure)和 NCSA[4](National Computational Science Alliance),NASA 支持的 IPG[5](Information Processing Grid)和 DOE 支持的 ASCI DISCOM[6]四个项目。在国内有中国科学院的 NHPCE[7](National High Performance Computing Environment)等计算网格系统。

在计算网格系统中,由于存在很多不同的高性能计算机和各种设备,如何将这些异构的系统组织起来协同运作就成了计算网格系统所要解决的最首要的问题,此外网格系统还要搜集整个系统内的所有信息、对所有的资源进行统一管理、在异构操作系统之间进行文件和数据的传输、进行系统安全的保障等等。这就需要在计算网格中的应用程序和实际进行计算的高性能主机之间建立一个中间层,这个中间层能够保证系统的正常工作,它向上层的应用程序提供统一的 API (Application Programming Interface),应用程序只需调用相应的 API 便可以进行高性能计算。

ACI 是面向任务的,资源分配和系统调度都以计算任务为单位。用户使用 ACI 系统进行高性能计算的过程就是一个提交任务、执行任务、访问任务、撤消任务的过程。一个完整的计算任务包括原始的计算数据、计算函数库以及计算要求等。本文中,第二部分将介绍 ACI 系统的整体结构,后面的部分重点介绍了 ACI 系统中的 LRM 子系统。

二、ACI 系统的整体结构

ACI 系统由以下几部分构成:高性能计算结点(集群式计算系统)、结点内部资源管理系统(Local Resource Management,以下简称 LRM)、全局资源管理系统(Globle Resource Management,以下简称 GRM)、安全子系统、任务管理子系统、用户管理子系统、主机管理子系统、计费管理子系统、系统维护子系统、安全子系统、用户接口和 ACI 应用等。其体系结构如图1所示。

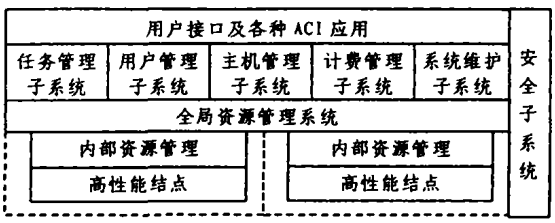


图1 ACI 体系结构

所谓资源,包括高性能结点中能够提供为高性能计算服务的硬件和软件资源,比如硬件资源有 CPU 数量、存储能力、网络带宽等;软件资源有计算函数库、数据库等。对资源的访问和管理是 ACI 系统的核心。ACI 中对资源的管理是两层结构,上层是 GRM,下层是 LRM,如图2。

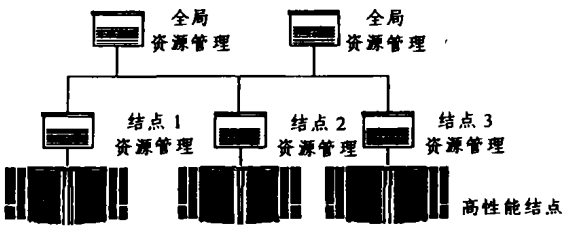


图2 ACI 资源管理示意图

方 兴 硕士生,主要研究领域为并行计算.李三立 中国工程院院士,博士生导师.王小鸽 副教授.黄震春 讲师.何 川 硕士生.

GRM 通过每个高性能结点上的 LRM 管理所有的子系统。GRM 并不占用计算资源，它仅仅进行资源的分配，它根据计算任务的特点以及系统内各结点的状态决定为这个任务分配哪些计算资源。LRM 存在于一个高性能结点内部，负责本地资源的管理，主要的工作是：搜集自身的状态信息；任务的启动与中止；文件的上传与下载。

一个计算任务首先通过用户接口被提交到 GRM 中，这个计算任务应该包括以下信息：选用的计算模板；提交的初始数据文件；需要的 CPU 结点个数；大致的运行时间；使用的优先级策略。

GRM 根据该任务的需求在计算资源中进行查找，如果找到满足条件的计算资源，就与相应结点上的 LRM 进行通讯，进行资源的分配。GRM 首先在资源使用表中占用该结点的计算资源，然后把计算任务及相关的信息传送给该结点，最后向其发送任务启动命令。资源的分配要在任务的调度之前进行，这样做是为了避免出现多个任务同时占用同一资源这种情况的发生。资源分配成功后，这部分资源就为这个任务预留了出来，这样便不会出现任务启动时突然发现该资源被其他任务占用的情况。

一个任务在高性能结点上启动以后，LRM 要负责对这个任务进行监控，确保这个任务的正常运行。在收到上层发来的任务中断或者任务恢复命令后，LRM 要进行相应的操作。最后当计算任务正常结束时，LRM 要将计算结果返回给 GRM，并通知 GRM 任务已经正常结束。对于在任务运行过程中出现的各种异常状况，LRM 也要进行相应的处理并向 GRM 报告。

三、LRM 的组成

LRM 主要有以下几方面的工作：

1. 执行命令：LRM 受上层 GRM 的控制，执行上层发来的各种命令，例如任务启动、中断、恢复、撤销以及下载断点和结果文件等。

2. 任务调度：根据结点内各处理机的负载情况对系统内的所有任务进行调度。

3. 垃圾数据清理：定期清除结点内各处理机上无用的目录和计算程序。

4. 任务断点结果收集：收集正在运行的任务的断点和已经完成的程序的结果文件，经过一定的处理之后，向上层返回。

5. 监控系统：主要包括监控系统中的各种资源和进程，监控正在运行的任务。

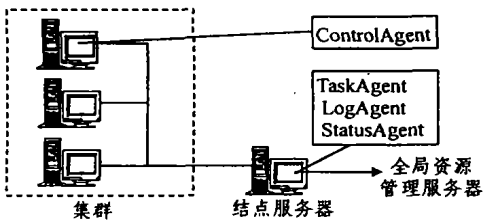


图3　LRM 的构成

LRM 由许多不同功能的服务进程构成。在结点之外有一台结点服务器，对整个结点中的所有处理机进行控制，在该服务器上主要有三个服务进程，分别为 TaskAgent、LogAgent 和 StatusAgent。TaskAgent 负责接受资源分配请求、进行资源分配并完成整个任务的调度。LogAgent 负责从结点中搜集任务断点文件以及结果文件。StatusAgent 负责从结点中的每台处理机上搜集各种状态信息，包括 CPU 占用率、内存使用情况、磁盘使用情况等。此外，在结点中的每台处理机上都运行着一个服务进程 ControlAgent，这个进程完全控制每台处理机，响应服务器发来的控制命令。

四、LRM 内部的通讯

LRM 内部所有服务进程之间的通讯都是通过 TCP/IP socket 进行的。采用 socket 通讯的优势在于这种方式较为稳定，而且适合于异构操作系统。在 ACI 中，计算结点上的操作系统可能是不同的，为了使不同的系统能够协同运作，解决系统异构性是十分重要的。

LRM 内部的通讯可以分为两种情况，一种是计算数据的传递，另一种是控制命令的发送与接收。对于计算数据，由于不需要对数据内容进行处理，而且数据量比较大，因此通过 ftp 方式进行。在 ACI 系统中使用 ncftp[8]作为数据传输的工具。对于命令的传输，为了便于命令的解释与执行，命令要有统一的格式，在 ACI 系统中命令报文采用 XML[9]格式，下面这个命令报文显示了如何在结点内部传递处理机信息：

```
〈result hostname＝“machine1” ip＝“192. 168. 1. 20” type＝“basicability”〉
〈detail name＝“cpuinfo” total＝“1600” used＝“400”/〉
〈detail name＝“memoryinfo” total＝“256M” used＝“122M”/〉
〈detail name＝“storageinfo” total＝“20000M” used＝“6780M”/〉
〈/result〉
```

在通讯中采用 XML 格式的报文有如下优势：①XML 中的内容不固定，具有很强的可扩展性；②XML 语法是一种树状结构，适合于高性能计算中大量数据信息的传输；③XML 有严格的语法，容易进行判错处理；④XML 的开发较为成熟，有很多现成的工具包可供使用，而且适合于不同的操作系统。

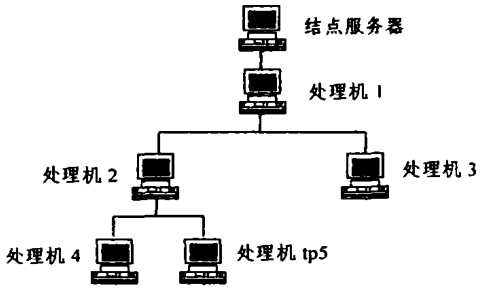


图4　分层通讯示意图

在 LRM 中，结点服务器要控制结点中的每一台处理机。这种管理是集中式的管理，当结点内的处理机数多于一定的数目时，如果每台处理机与结点服务器同时建立 socket 连接，必然会消耗服务器大量资源，甚至会造成服务器的崩溃，如果采用 UDP 广播的方式，为了确保稳定的数据传输，就需要建立一套详细而又复杂的数据校验/重传机制，因此这种方法也是不可行的。在这里采用了分层的通讯方式。所谓“分层”，也就是将传输数据的路径分层，每次某台处理机或服务器都仅与有限的几台处理机进行通讯，然后这几台处理机再与其它处理机进行通讯，通讯的路径是一个树状结构。对于一个由 n 台处理机组成的结点来说，如果每台处理机每次最多与 r 台处理机进行通讯，那么所有处理机可以被分为 $\log_r n$ 层，如果进行一次通讯的时间为 t 秒，那么经过 $t \times \log_r n$ 秒后，所有的处理机都可以接收到控制命令。图4是 r＝2时进行

分层通讯的一个示意图。在通讯过程中，命令发送方除了要发送数据信息，还要发送传输路径信息，指示下一次有哪些处理机参加通讯。

五、LRM 中任务的调度

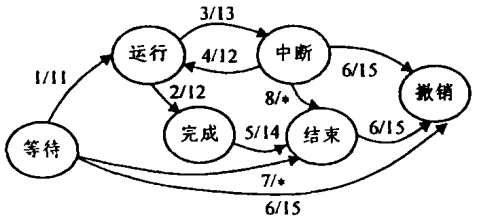
计算任务是以 MPI[10](Message-Passing Interface)并行程序的形式在高性能结点上运行的。由于结点各不相同，配置信息也有差别，为了使一个并行计算任务在结点上运行，需要以下条件：

·一台处理机作为主控处理机，控制任务的起停；

·在主控处理机上根据要求生成启动脚本文件以及中止脚本文件；

·每台处理机上都为该计算任务建立起相应的目录以及初始数据文件；

·每台参与运算的处理机必须能够正常地运行该并行计算任务，并与主控处理机保持稳定的连接。



1.启动成功 2.任务完成 3.中断成功 4.恢复成功
5.结果收集结束 6.收到撤销请求 7.启动失败
8.恢复失败 11.启动断点收集 12.启动结果收集
13.结束断点收集 14.结束结果收集 15.启动垃圾清理

图5 集群中任务状态有限状态机

LRM 进行任务调度的目的是确保任务的运行。任务调度的操作主要有以下几种：启动、中断、恢复、结束、撤销。一个任务在 LRM 中的状态有等待、执行、中断、完成、结束、撤销6种。图5给出了一个任务从等待到清除的有限状态机的描述，从图中可以清晰地看出任务在结点中的工作流程和被调度的过程。由于一个结点中同时运行的任务是有限的，而且不同任务所具有的优先级以及调度策略也不相同，因此对于每个任务的调度都将采用调度算法进行控制，调度算法将在下一节加以介绍。

六、任务调度算法

根据不同的任务，LRM 为每个任务设置了相应的调度策略和优先级。调度策略有三种，分别为独占式、抢占式和非抢占式。其中独占任务具有最高优先级，它是一种实时任务，要求独占本高性能结点进行计算。抢占式任务优先级次之，它可以根据任务的优先级抢占系统资源，使得优先级较低的任务让出相应的资源。非抢占式任务优先级最低，它采用先来先服务的策略进行调度，只有当系统中有多余的资源时这种任务才能运行，否则就等待。

不同于单机操作系统，LRM 要管理许多处理机，其任务调度算法必须具有以下能力：

·根据任务的不同策略以及不同优先级对任务进行调度，能够满足某些实时任务的需要；

·能够根据当前结点中所有处理机的资源使用状况动态地分配资源，尽量做到负载平衡；

·能够指定哪些处理机参与任务的计算；

·能够处理各种异常状况，比如任务启动失败或任务异常退出。

在 LRM 中，设置了三个任务队列，分别为等待队列、执行队列和中断队列。每当 LRM 收到一个新的任务请求后，就将它放入到等待队列中去等待系统调度。在以下几种情况下，调度程序进行系统调度：①系统资源空闲，而且等待队列或者中断队列中有任务等待运行；②系统超负荷运转，需要中断几个任务的运行；③有独占或者抢占式任务到来；④收到 GRM 的调度命令。

当系统需要启动一个任务时，调度程序查看等待队列和中断队列中就绪的任务，首先判断所有任务的调度策略，独占式任务能够抢占抢占式任务的资源，抢占式任务能够抢占非抢占式任务的资源，如果调度策略相同，那么再根据任务优先级进行调度，优先级高的任务能够抢占优先级低的任务的资源。独占式任务比较特殊，在它运行之前，系统中的所有任务都要中断，为该任务让出所有的资源。非独占式任务不能抢占，只能等待系统空闲后才能运行。

当系统要中断一个任务时，采用的策略正好相反，需要从运行队列中找出一个等级最低的任务加以中断，然后把这个任务放到中断队列中去。

在一个高能性能计算结点中，最重要的资源是 CPU 资源[11]，而且 CPU 的数量也是整个系统的瓶颈。LRM 对任务的调度主要考虑对 CPU 的分配。为了提高整个系统的效率，调度算法必须平衡结点内部各 CPU 的负载。LRM 通过资源服务子系统能够动态得到当前所有 CPU 的使用情况。LRM 要做的是将 CPU 按照占用率进行排队。对于那些超过额定负载的 CPU，需要通过中断任务的方法释放这些 CPU 资源，需要中断的任务由调度算法决定。如果目前有足够的 CPU 资源能够维持一个新的任务，那么便开始新任务的调度。

七、LRM 对任务和结点的监控

仅仅启动一个任务是不够的，LRM 必须不断监视所有正在运行的任务，判断该任务是否仍在运行，并且获取该任务占用的计算资源信息，如果一个任务意外中断，那么 LRM 需要重新启动这个任务，在多次启动失败以后，LRM 需要向 GRM 报告任务失败的消息。LRM 也要对结点内所有处理机的状况有清楚的了解，对结点的监控主要有以下三方面：

1)动态发现结点中的每台处理机，获得处理机的基本资源信息，包括 CPU 型号、内存大小、硬盘大小；

2)实时查询每台处理机的资源使用情况，主要为 CPU 占用率和内存占用率；

3)实时获取每台处理机的进程列表。

由于在每台处理机上都存在服务进程，因此对任务以及结点中每台处理机的监控都变得较为容易。每台处理机启动后都会自动向结点服务器发送注册报文，在处理机即将关闭时，也会向结点服务器发送注销报文。结点服务器利用这些报文动态判断当前系统中的处理机工作状况。每隔一段时间，结点服务器还会向这些已经注册的处理机发送查询报文，判断该处理机是否工作正常，并获取最新的资源信息。报文格式均为 XML 格式，实例如下：

结点服务器发送的查询 CPU 信息的报文：

```
〈query type="CPUInfo"/〉
```

从处理机返回的报文：

```
〈result type="CPUInfo"〉
〈CPUAmount value="2",load="35%"/〉
〈CPU number = "1", type = "intel386", speed = "300hz", load =
    "35%"/〉
〈CPU number = "2", type = "intel386", speed = "300hz", load =
    "40"%/〉
〈/result〉
```

（下转第23页）

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