文章编号:1001 - 9081(2009)10 - 2647 - 05

基于 P2P 的计算资源共享与聚集平台 PCP

俞 辉
(中国石油大学 计算机与通信工程学院, 山东 东营 257061)
(huiyu@ mail. hdpu. edu. cn)

摘 要:构造了一个基于 P2P 的计算资源共享与聚集平台(PCP)。PCP 平台可以把 Internet 上空闲的异构计算资源高效地聚集起来,完成大规模的并行分布式计算。PCP 平台具有对等性、易用性、高效率等特点。该平台的对等性得益于建立在对等网络 JXTA 协议上的应用层覆盖网络及节点组织机制;易用性得益于建立在 JXTA 协议通信库上的并行编程库,通过作业管理子系统来完成并行应用程序的提交管理与监控。模拟实验结果表明,PCP 平台在处理主/从类型的并行应用问题上,吞吐量、加速比等方面有较好的性能。

关键词:分布式计算;对等网络;通信协议;并行处理
中图分类号: TP393 **文献标志码**:A

PCP: P2P-based computing resource sharing and aggregation platform

YU Hui
(*Institute of Computer and Communication Engineering, China University of Petroleum, Dongying Shandong* 257061, *China*)

Abstract: A P2P-based computing resource sharing and aggregation platform called PCP was developed and constructed in this paper. PCP is a platform harnessing the immense computational resources available in the Internet for parallel and distributed computing. PCP platform is peer-to-peer based, easy to use and efficient. These features are strongly supported by a network overlay over which peers can communicate with each other. The authors made use of a general-purpose P2P library, JXTA, supporting the common requirements of P2P software, including network overlay. Other features of the P2P library, such as Ad Hoc self-organization, discovery and grouping of peers, also support PCP efficiently. A parallel programming library was also presented in PCP by JXTA communicating library. A volunteer node can submit and monitor the parallel applications by job management subsystem to complete the whole computation. The results obtained from performance analysis show that the throughput and speedup ratio of PCP are better in master-slave style parallel applications.

Key words: distributed computing; Peer-to-Peer (P2P); communication protocol; parallel processing

0 引言

如何把 Internet 上的大规模空闲计算资源共享与聚集起来并完成一个科学问题的求解,是目前的一个很重要的研究课题。当前的一些分布式计算项目[1-4]大多是专用和集中控制的,因而存在着某些可用性和可扩展性的问题。例如网格计算模型往往聚集的是大型稀少的资源,诸如高端的超级计算机、数据存储、观测传感仪器等,相互共同安全访问与横跨多个组织,不是很适合于桌面 PC 机的计算;基于 C/S 模式的计算平台[5-6]不能够真正实现既是"资源提供者"又是"资源的使用者"的对等模式(Peer-to-Peer,P2P)计算资源共享。

本文构造了一种基于 P2P 的计算资源共享与聚集平台(P2P based Computing Platform,PCP),它能够很好地共享和聚集 Internet 上的大规模计算资源,实现真正的对等分布式计算。只要提交了一份 PCP 平台的计算作业,任何 PCP 的用户都可以使用他人的资源。PCP 采用 Java 语言的跨平台运行环境,使其更加容易聚集异构环境下的计算资源,充分利用好志愿机的空闲 CPU 周期与磁盘空间。

PCP 平台开发的关键技术是基于 JXTA[7]协议的通信库,它使计算网络中的节点形成了一个应用层覆盖网络;通过该通信库,配合 JXTA 协议应用编程函数库,组成了一个适合 PCP 平台的主/从风格的编程模型。这些关键技术使得 PCP 平台具有以下优势:1)高效地实现大规模 Internet 环境下的节点组织与通信,包括节点的自组织能力、节点的发现机制,节点工作组的形成;2)跨越防火墙和网络地址转化等问题,对于覆盖网络上的任何机器都可以直接与其他机器相连接,满足了对等计算模式的需求;3)为应用的编程模型、应用的编程接口函数提供支持,扩大应用范围。

1 相关工作

在面向 Internet 的大规模计算资源共享与聚集领域,目前国内外开发了一些很重要的项目。BOINC[1]是一个现在正在实际运行的计算项目,也是目前最成熟的平台。BOINC 提供了一套 BOINC API 和 BOINC Graphic API 函数以支持任务的效果显示和数据操作,BOINC 平台上已经实现的应用包括 Folding@ Home、FightAIDS@ Home 等。Javalin[2]也是比较成功的平台,把分布式网络中的机器分为代理机、客户机和主机三种实体,连接在 Internet 或 Intranet 上的用户都可以很轻易地参与到 Javalin,Javalin 平台上已经实现的应用包括光栅跟踪、梅森素数搜索等。XtremWeb[3]是一个开放式的志愿者计算平台,它采用 Java 语言的 RMI(Remote Method Invocation)机制来划分和调度子任务,采用 P2P-RPC [10]的方式来实现,

收稿日期:2009 - 04 - 28;**修回日期**:2009 - 06 - 01。
作者简介:俞辉(1974 -),男,浙江上虞人,讲师,硕士,主要研究方向:数据挖掘、嵌入式系统。

把远程过程调用(Remote Procedure Call,RPC) API 作为它的编程模型的应用编程接口。XtremWeb 由于其性能和稳定性受限,缺乏友好的用户界面,因此它未能被广泛应用。

BOINC[1]、Javalin[2]、XtremWeb[3]和 SETI@ Home[6]都是比较早期的计算资源共享平台,最近几年提出的 Paradropper[11]和 JNGI[12]等在通信库和编程模型上有了很大的改进,计算环境的网络拓扑由 C/S 模式到层次结构,最后发展到对等网络拓扑结构,运行的应用也有所增加。

上述这些计算资源共享平台的最终目标是:高效性、友好的客户端界面、方便程序员开发、跨平台运行、容错性、负载平衡、可扩展性和安全等,但是每个平台只能获取这些目标中的一部分,而本文的 PCP 平台在高效率性、易用性与方便开发性方面都有比较好的性能。

2 JXTA:PCP 平台的通信协议

2.1 JXTA 协议

如何让在 PCP 平台的上所有的计算机可以平等地双向通信是非常重要的,因为 PCP 的目标就是能相互平等地共享计算资源。我们利用通用 JXTA 的 P2P 库,提供了一个覆盖网络,其中任何一台志愿机都可以方便地连接到其他计算机上,即使这些计算机安装有防火墙和处于内部子网中,也能实现真正的对等计算模式。

在覆盖网络上所采用的 JXTA 技术,使得 Internet 上的空闲计算机形成了所谓的对等实体(Peer),对等实体通过对等实体的 ID 号来与其他的实体进行区别。对等实体通过某种规则可以形成对等实体组,有了 JXTA 技术,PCP 平台可以高效率地管理好这些节点,包括节点的加入退出平台、节点的轮询问机制、节点的通信和节点的发现机制等。这些功能的实现都可以利用 Sun 公司 JXTA 库中的函数调用来实现。有了 JXTA 库,辅助以我们开发的 PCP 平台中的其他通信库,即可以方便管理和开发并行应用程序。

2.2 节点的组织与发现

有了该应用层的覆盖网络, PCP 平台就可以像其他的一些 P2P 软件(即时消息通信、文件共享应用、流媒体直播/点播)一样可以调用 JXTA 库里的所有的函数,来实现应用程序的分布式计算。下面给出 PCP 平台中的对等组的形成与资源发现过程。

对等组的形成过程为:Internet 上空闲的计算机按照某种规则(物理位置邻近或任务特点)形成一个对等组,对等组可以控制多台计算机和广播信息给所有计算机。PCP 平台中创建一个基于 JXTA 技术支持的对等组,我们称其为作业执行工作组, 每一个并行应用都是通过作业执行工作组提交的。在作业执行工作组中,节点根据并行应用程序的特点进行相互通信。工作组也完成划分任务与子任务、数据文件与代码文件的传输、作业描述文件的形成等功能。除了这些功能之外, JXTA 技术还可以利用 IP 组播和中继器来实现网络信息(包括志愿机的硬件信息与软件信息)的传送功能。

PCP 平台也把 JXTA 中的节点发现机制充分利用起来。JXTA 技术支持分布的服务器,让其能够发现更多可以用的计算资源,包括同网段、对等实体组、通信管道等。实现该技术的关键是对等组中的广告消息。在 PCP 平台中,用户为其自己的并行应用程序创建一个作业执行工作组后,发布自己的作业执行工作组广告消息;计算资源提供者可以根据广告消息发现该工作组,选择其是否愿意贡献计算资源给该并行应用,是否愿意加入该作业执行工作组;一旦该计算资源的提供者加入了该组,该计算资源的提供者中的每一台计算机都可以通过广告消息被发现。

3 作业管理子系统

PCP 平台由作业管理子系统、作业监控系统和并行编程库组成。作业监控系统是一个基于 Web 的界面,主要显示平台与应用的运行状态。PCP 平台的用户使用主机(Host)与控制机(Controller)的方式来管理并行应用的作业。

主机:主机是一个服务器端的守护进程,运行在计算资源提供者的机器上,它可以用来发现作业执行工作组,接受一个并行应用作业。

控制机:控制机是一种工具,具有计算资源的用户通过该工具进行作业的提交,并且通过主机来控制并行应用的作业分配。

主机有安全性的要求,往往通过验证数字签名才能使用。主机往往是运行在非图形用户界面(GUI)模式,启动后就决定是否接受并行应用的作业或自动地根据某种策略转移其他所提供作业的用户。一个策略是代表其所接受的作业,这种策略可能是根据并行应用作业的名称或者是并行应用作业的提交者名称。用户运行的主机也可以在图形用户界面 GUI 上决定是否接受作业 。

图 1 显示作业执行工作组与主机和控制机之间的相互关系。基对等工作组是该基工作组的所有控制器和主机创建和加入后所形成的。通过提交一份并行作业,用户利用控制机来使用别的用户的计算能力,或者通过运行主机来贡献计算能力给别的用户。图 2 给出了利用控制机来提交作业的过程,图 3 给出了通过主机执行作业的过程。

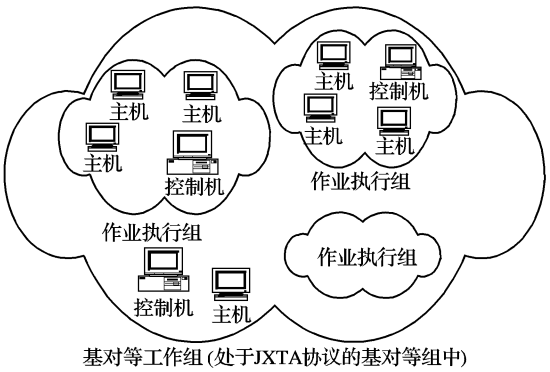

图 1 主机与控制机与作业执行组的关系

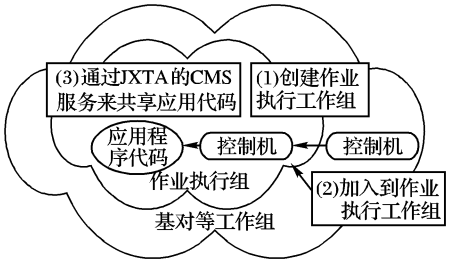

图 2 控制机提交并行作业的过程

由于 PCP 平台是采用 Sun 公司的 JXTA 协议库进行开发,并行应用程序往往也是 Sun 公司的 Java 语言开发,最终程序是通过编译与打包成 * jar 包来执行。具有计算资源的用户通过使用控制机来提交 jar 包与数据(参数文件、数据文件)。一个并行应用程序利用并行编程库来实现主机之间的通信。主机也可以自动运行一些本地的编译好的应用程序。

在 PCP 平台中,我们也采用了 JXTA 协议中的砂箱(SandBox)的实现,这种情况下本地的应用程序可以更好地支持安全的运行,并且计算资源的提供者能够更好地接受本地应用程序。

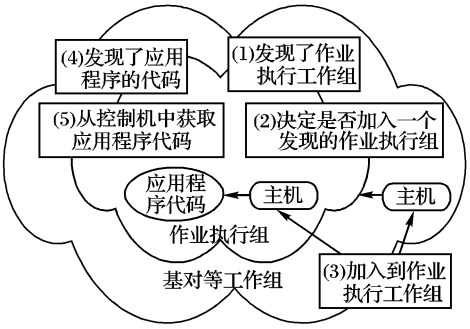

图 3　主机参与平台的过程

4　并行编程库

PCP 平台给应用程序开发员提供了基于 JXTA 的并行编程库。在设计这个并行编程库的时候充分考虑了方便与容易编程的原则。开发员不需要了解作业执行工作组通信与处理的细节,也不需要有像 MPI 或者 OpenMP 那样的并行程序设计思想。图 4 显示该并行编程库的组成结构。

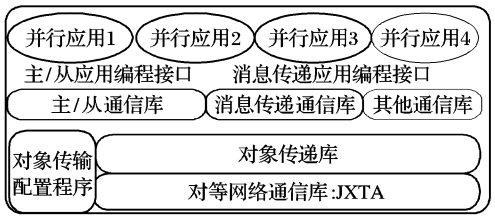

图 4　PCP 平台提供的并行编程库

4.1　对象传输库

首先提供了一个对象传递库,以支持多个并行编程模型,可以减少大量的开发工作。该对象传递库是直接使用 JXTA 协议的 API 接口实现的,隐藏了复杂的通信细节。有了对象传递库,主/从编程库与消息传递库就可以很容易地实现,因为它们依赖于对象传递库。下面的代码为利用对象传递库的 API 来实现单播与多播 Java 对象的例子程序,该例子中 Java 对象的接收实现可以采用同步(阻塞)或者异步(非阻塞)的方式。

对象传递库例程如下:

```
interface ObjectPassing {
  void send( PeerID receiver,  Serializable obj);
  void broadcast( Serializable obj);
  Envelope recv( PeerID sender);
  Envelope recv( PeerID sender,  long timeout);
                              // receive a message from anyone
  Envelope recv( ); Envelope recv( long timeout);
                              // check if a message is available
  boolean available( );
  boolean available( PeerID sender);
                          // register and unregister a message listener
  void addObjectPassingListener( ObjectPassingListener listener);
  void removeObjectPassingListener( ObjectPassingListener listener);
}
```

因为最原始的 JXTA 的已经足够灵活,可以支持所有的需要的 P2P 操作,所以一个通信过程基本上是被对等实体的 Peer ID 所确定 ,对象传递库的 API 只是 JXTA 通信协议 API 的一个很小的子集,它可以很容易构建其他的各种编程库。

4.2　消息传递库

消息传递库是在发送和接收对象时候所用到的库。在实现的时候,通信的过程不是通过 Peer ID,而是通过一个非负整数级(Rank)。这种方式与 MPI 并行程序设计中有点类似。该消息传递库中包含 Peer ID 与非负整数级的对应机制,可以使它们之间互相转化。当并行应用作业开始运行时,高级别的主机可以被分配到一个控制机,并通知在作业执行工作组中的所有主机。系统内部构成了一张高低级别表,但是该表的一致性维护也需要一定的开销。

4.3　主/从编程库

主/从编程库使得 PCP 平台支持主/从风格的并行处理。在这种情况下进行开发,必须把应用问题划分为许多独立运行的子任务与一个控制这些子任务的主控程序。所有的子任务程序执行完了,再由主控程序进行收集与汇总即可以完成整个问题的运算。下面的代码显示了主/从风格的并行应用编程的过程。

主/从编程库代码例程如下:

```
class AMaster implements Master {
  Serializable getWorkerInitData( )
    { return  < data to initialize a worker > ; }
  start( ) {
    while ( ...)
    {
      SubTask st  =  < a subtask > ;                  // submit a subtask
      submit SubTask ( st) ;
    }
  }
}
class AWorker implements Worker {
  void init( Serializable initData)
    {// initialize this worker instance }
  WorkResult process( SubTask st)
     {// process a subtask and return the result }
}
```

子任务的分发与调度是由主/从编程库所负责,而不是由应用程序负责。即使工作组已开始运行或志愿机正在处理子任务,一个志愿机也可以在任何时候加入或退出的作业执行工作组。这种节点自组织的加入和离开作业执行工作组由主/从编程库所控制,一个应用开发员并不需要去关注里面的细节。如果志愿机离开了工作组,一个尚未完成的子任务马上会交付给另一个空闲的志愿机来完成。

有时候 Internet 上有恶意的志愿者,他们加入到作业执行工作组后,也可能接收子任务但是并不去执行它。它们也有可能返回虚假与错误的子任务结果给 PCP 平台,影响到整体的计算结果。主/从编程库都可以处理这两种情况。

第一种情况我们使用基于超时的再分配子任务策略,多次分配子任务。第二种情况中虚假或者错误的结果问题也由主/从编程库自动处理。主/从编程库通过轮询或者投票[8]的方式探测到虚假的计算结果,如果探测到了,主/从编程库会再次分发子任务,以便执行的结果不受到影响。

一个主程序把一个子任务分配 m 次给志愿机,并把志愿机返回的结果进行 n 次比较。如果 n 次比较都正确,主程序将把该结果作为正确的结果进行接收。对于其中 n 次的比较结果,应用开发人员可以提供代码自动执行匹配,这种类型的匹配对计算子结果是数值类型比较有用,比较适合全球分布式计算平台,因为 PCP 平台运行的应用以数学计算问题居多。

主机在主/从编程库中担任了主控机的角色。在PCP平台中,一个控制机只负责管理作业的运行。主机担任了主控机的角色,这样使得控制机能专门处理作业的提交与管理。但是有时候主机也只是作为候选的主控机。用户在连接主机的时候,可以根据条件进行选择常驻的主机。如果控制机发现多个常驻的主机,它会随机选择一个常驻主机。如果没有常驻主机,主机会随机选取作业执行工作组的任何一个机器作为常驻主机。整体来说PCP平台中的一个主机在参与一个主/从风格的并行应用的过程中,一般都是担任主控机的角色。

4.4 配置程序

与一个顺序程序相比,开发并且调试一个并行程序往往不是那么容易。这些困难包括:更长的运行时间、并行程序划分的思想、并行程序设计环境的安装与配置,所以PCP平台中就需要一个启动对象传递库的配置程序。该配置程序稳定控制并运行在一个计算机上。配置程序支持主/从风格的并行应用和消息传递风格的并行应用,因为两种类型的应用都依赖于对象传递库的API来进行通信。

有了对象传递库、消息传递库、主/从编程库与配置程序,即可以进行基于Internet的并行分布式计算的应用开发。

5 实验与性能分析

5.1 实验环境

在PCP的测试环境中,志愿机节点的个数为40,个数不多原因在于我们目前尚无法获得更大规模的计算资源。实验中,所有的机器通过100 Mbps以太网连接在一起。机器的软硬件配置如表1所示。PCP平台的构造过程主要决定于其基于JXTA协议的通信库,完成的是并行计算,所以我们在性能分析的时候以PCP平台的吞吐量与计算加速比为主要评价目标。在实验中,JXTA协议是TCP连接而不允许使用HTTP连接。

表1 PCP平台的实验环境

节点角色	CPU	JXTA协议	TCP/Mbps	节点数目	操作系统	Java环境
主机	Intel P4 2.8 GHz	2.1	100	36	Windows XP	J2sdk1.4.2
控制机	Intel P4 2.8 GHz	2.1	100	4	Windows XP	J2sdk1.4.2
主控机节点	Intel P4 2.8 GHz	2.1	100	1	Windows XP	J2sdk1.4.2

5.2 子任务的吞吐量

我们测试了一个主控机在一个固定的时间段里,对子任务分配与子任务结果的收集过程的吞吐量。并行应用的源代码是解线性方程组的雅可比迭代法,通过设置不同的数据参数文件,每个设置情况对应于一个迭代算法的子任务,形成$2^{13}=8192$个不同的子任务,8 192个子任务分配与收集完成后,主控机进行子任务结果的汇总处理。可以说该实验是把雅可比迭代法一共运算了8 192次,并把每次的结果收集到主控机的临时文件中。志愿机节点不执行任何在子任务上的后处理,而运算完后只是返回一个子结果。该实验的目的是测试PCP平台中任务执行工作组的子任务传输延迟与吞吐量。

我们选择一个固定时间$T=500$ s作为时间单位,在平台中的总节点的个数为$n(n=8、16、24、32、40)$的情况下,测试PCP平台完成的子任务个数统计,如图5所示。从结果可以看出,随着网络中的志愿机节点的增加,PCP平台完成的子任务个数也越来越多,当志愿机节点的个数达到30以上时,完成的子任务的个数增加很快,平台体现出良好的扩展性与吞吐量。

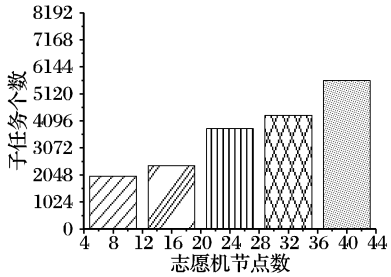

图5 子任务完成情况的吞吐量

5.3 加速比

在主/从风格的并行应用中,当并行粒度变得很大时,性能变得很容易改善,这是因为并行粒度越大,可以节省越多节点通信的开销。随着志愿机节点数目的增加,我们测试了PCP平台的加速比。

在这个实验过程中使用RC5密码算法[9]。它是一个参数变量分组算法,由三个参数确定一个加密算法。三个可变的参数是:$w/r/b$,w为以比特表示的字的尺寸,r为加密的轮次数,b为密钥的字节长度。在我们的实验中所取的参数为:$w=32, r=12, b=8$(RC5_32_12_8)。志愿机的数量从1增加到32。根据密码的密钥空间来划分子任务,设置子任务的数量是20 000,RC5密码破解算法在不同密钥空间情况下1个子任务执行时间大约为1~1.5 s。实验中每个志愿机节点都是循环尝试密钥来破解RC5密码。

图6显示了加速比随着志愿机节点数目增加的变化情况。理想的加速比是直线的,随着志愿机节点的数目增加而线性增加,但是实际的情况下都是非线性增加。原因可能是志愿机节点数大大增加后,网络通信的开销也大大增加,导致整体性能下降。

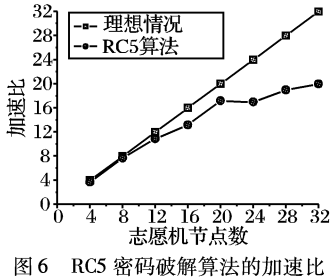

图6 RC5密码破解算法的加速比

尽管这样,在志愿机数目为32时,最高可以获得20的加速比,这个是因为RC5算法的粒度比较好或者执行的时间比较长。但是我们要注意,这种粗粒度并行、低带宽需求的应用在基于Internet的分布式计算中是很少见的,还有就是密码破解类应用属于计算密集性应用,属于易并行应用,不是所有的应用都易于并行。

6　结语

本文构造了一种基于 P2P 的计算资源共享与聚集平台，该平台的高效性得益于建立在 JXTA 协议上的节点组织机制,易用性得益于建立在 JXTA 协议上的并行编程库,通过作业管理子系统即可以把 Internet 环境下的志愿机组织起来,实现大规模计算资源共享与聚集。通过对 PCP 平台的通信延迟、吞吐量及加速比进行测试得知,该平台取得了比较好的性能。我们在开发 PCP 平台的过程中都假设主控节点是比较稳定的,所以提高平台的容错性能是本文的下一步工作。

参考文献:

[1] ANDERSON D P. BOINC: A system for public-resource computing and storage[C]// GRID 2004: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing. New York: ACM Press, 2004: 4 - 10.

[2] NEARY M O, PHIPPS A, RICHMAN S, *et al*. Javelin 2.0: Java-based parallel computing on the Internet[C]//Proceedings from the 6th International Euro-Par Conference on Parallel Processing, LNCS 1900. London: Springer-Verlag, 2000: 1231 - 1238.

[3] FEDAK G, GERMAIN C, NERI V, *et al*. XtremWeb: A generic global computing system[C]// Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. New York: ACM Press, 2001: 582 - 587.

[4] Distributed. net[EB/OL]. [2009 - 01 - 01]. http://www.distributed.net/.

[5] 章勤, 鄢娟, 金海, 等. 吴宇网络计算平台体系结构研究[J]. 计算机研究与发展, 2003, 40(12): 1725 - 1730.

[6] KORPELA E, WERTHIMER D, ANDERSON D, *et al*. SETI@home: Massively distributed computing for SETI[J]. Computing in Science and Engineering, 2001, 3(1): 78 - 83.

[7] GONG LI. JXTA: A network programming environment[J]. IEEE Internet Computing, 2001, 5(3): 88 - 95.

[8] SARMENTA L F G. Sabotage-tolerance mechanisms for volunteer computing systems[C]// CCGRID: Proceedings of the 1st International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2002: 561 - 572.

[9] HEWGILL G. RC5 and Java Toys[EB/OL]. [2009 - 02 - 02]. http://www.hewgill.com/rc5/index.html.

[10] DJILALI S. P2P-RPC: Programming scientific applications on peer-to-peer systems with remote procedure call[C]// CCGRID: Proceedings of the 3st International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2003: 406 - 413.

[11] 窦文, 贾焰, 王怀民, 等. 基于对端重叠网络的通用大规模计算资源共享环境的构造[J]. 计算机学报, 2004, 27(1): 21 - 31.

[12] VERBEKE J, NADGIR N, RUETSCH G, *et al*. Framework for peer-to-peer distributed computing in a heterogenuous and decentralized environment[C]// Grid 2002: Proceedings of the third International workshop on Grid Computing, LNCS 2536. Heidelberg: Springer, 2002: 1 - 12.

(上接第 2637 页)

轻载变化不大,而随着负载量的不断注入,拒绝服务依据系统的负载,将当前系统的负载控制在一个系统承载接近饱和的状态。分析原因:1)调度失败率体现了节点在轻载时系统固有的稳定性和可靠性,同时也体现了通信负载的限制使得丢包率减少;2)拒绝服务则一定程度上实现了网格对负载饱和状态的界定;3)吞吐量的提高则体现了系统良好的性能,工作的更加有序,冲突少。

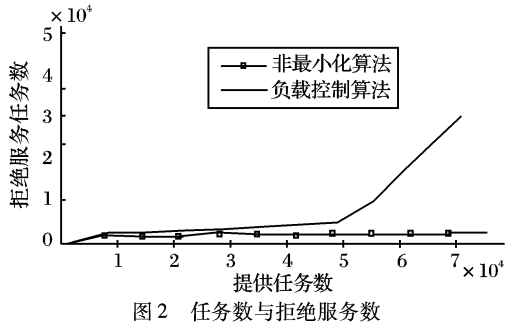

图 2　任务数与拒绝服务数

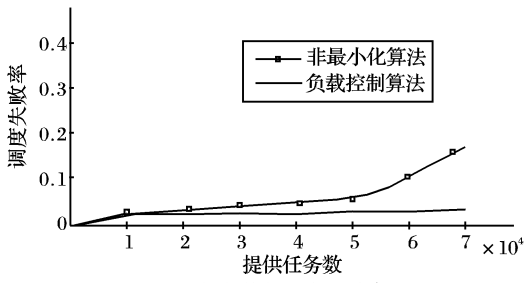

图 3　任务数与调度失败率

显然,负载控制算法通过控制节点的负载量、通信的承载量以及整个网格系统的总负载,在资源的调度和任务的总量控制上实现了主动管理,使系统更为健壮,尤其在系统进入重载时,效果更加明显。

6　结语

文章通过对网格节点负载、任务和资源间通信链路负载和整个系统负载的评价,在节点和通信上平衡负载,按照系统的吞吐能力,在系统级设计了任务量的控制,由此使得系统在负载平衡和总负载量上实现了主动控制,提高了系统的运行效率,增强了系统的健壮性。实验表明,所提出的方法在系统吞吐量、调度失败率等方面表现出良好的性能,是有效的。

参考文献:

[1] BRADLEY A, CURRANN K, PARRZ G. Resource discovery and management in computational GRID environments[J]. International Journal of Communication Systems, 2007, 19(6): 639 - 657.

[2] ALOISIO G, CAFARO M, EPICOCO I, *et al*. Resource and service discovery in the igrid information service[C]//Proceedings of International Conference on Computational Science and Applications, LNCS 3482. Berlin: Springer-Verlag, 2005: 1 - 9.

[3] MA MAN-FU, WU JIAN, LI SHU-YU, *et al*. . A grid-distance based scheduling for grid resource management[C]// Proceedings of HPC. Los Alamitos: IEEE Computer Society, 2005: 576 - 581.

[4] 刘敏, 李忠诚, 石晶林, 等. IPv6 网络中基于优先级的可用带宽测量方法[J]. 计算机研究与发展, 2004, 41(8): 1361 - 1367.

[5] MUU L D. DC optimization methods for solving minimum maximal network flow problem[EB/OL]. [2009 - 02 - 01]. http://www.mmm.muroran-it.ac.jp/~shi/muushi.pdf.

[6] BUYYA R, ABRAMSON D, GIDDY J. Nimrod/G: An architecture or a resource management and scheduling system in a global computational grid[C/OL]. [2009 - 03 - 01]. http://www.gridbus.org/papers/GridEconomy.pdf.