

文章编号:1001-9081(2009)08-2161-06

WAPM: 适合广域分布式计算的并行编程模型

付崇国¹, 徐胜超²

(1. 大连东软信息学院 网络中心, 辽宁 大连 116023; 2. 广东邮电职业技术学院 计算机系, 广州 510630)

(sine6789@sina.com; isdooropen@126.com)

摘要:早期的 MPI 与 OpenMP 等编程模型由于扩展性限制或并行粒度的差异而不适合于大规模的广域动态 Internet 环境。提出了一个用于广域网络范围内的并行编程模型(WAPM),为应用的分布式计算的编程提供了一个新的可行解决方案。WAPM 由通信库、通信协议和应用编程接口组成,并且具有通用编程、自适应并行、容错性等特点,通过选择合适的编程语言,就可形成一个广域范围内的并行程序设计环境。以分布式计算平台 P2HP 为工作平台,描述了 WAPM 分布式计算的实施过程。实验结果表明,WAPM 是一个通用的、可行的、性能较好的编程模型。

关键词:编程模型;分布式计算;并行处理;并行程序设计

中图分类号: TP393 **文献标志码:** A

WAPM: A parallel programming model in large scale Internet distributed computing environments

FU Chong-guo¹, XU Sheng-chao²

(1. Network Center, Dalian Neusoft Institute of Information, Dalian Liaoning 116023, China;

2. Department of Computer, Guangdong Vocational College of Posts and Telecom, Guangzhou Guangdong 510630, China)

Abstract: Programming models like MPI or OpenMP are not suitable for large-scale Internet environments because of scalability or parallel grain issues. In this paper, a novel parallel programming model in large-scale Internet environments called Wide Area Programming Model (WAPM), which provided a feasible way for parallel programming, was designed and implemented. WAPM includes three modules: communication library, communication protocol and application programming interface. WAPM is a good programming model, and is strongly supported by its general programming, adaptive parallelism and fault tolerance. An example application was also demonstrated with WAPM on a specific distributed computing platform. In order to testify the efficiency of WAPM, a serial of simulation experiments were done. The results obtained from performance analysis show that WAPM is a general and feasible approach for parallel programming.

Key words: programming model; distributed computing; parallel processing; parallel programming

0 引言

分布式计算平台^[1]的基本原理是利用 Internet 上广泛分布的空闲计算资源,运行计算量大的、分布式的应用。目前国际上著名计算平台包括 SETI@home^[2], XtremWeb^[3], BOINC^[4], JNGI^[5], P3^[6]等都是这种计算模式。由于它具有价格低廉、容易构建、高性能等优点,所以近年来在大规模工程和科学计算中显示出越来越重要的作用。

目前在上述分布式计算平台上运行的科学应用往往是粗粒度的任务级(程序级)并行,这种应用首先需要划分为许多能够独立运行的子任务和一个控制这些子任务的主程序,通过把这些大量的子任务分发给 Internet 环境下的志愿机进行计算;当所有的这些子任务计算完后,再由主程序对子任务的计算结果进行控制和汇总进而完成整个应用的求解。这种类型的应用在某些文献^[1]上称为 Master-Worker(主-从)风格的并行应用。如何为这种风格的应用提供一个 Internet 环境下的并行编程模型是很重要的,这是因为面对大规模动态的 Internet 环境,需要重新考虑容错、可扩展、带宽差异等一系列的新问题。在多台分布并行编程环境下,消息传递 MPI^[7]和共享变量 OpenMP^[8]是两种最基本的编程模型,在 MPI 的编

程模型中,程序的编写是在物理处理器数目已知和固定的情况下进行的,很少或没有考虑在异常的情况下节点离开或加入系统的情况。OpenMP 往往执行的是进程级或循环级的并行,而不是分布式计算平台下的任务级(程序级)粗粒度并行。这两者由于并行粒度和扩展性等问题而不适合于大规模广域和动态的 Internet 环境。另外除了能够通用编程外,一个良好的分布式计算平台的编程模型要有自适应并行和容错等特性,它不能总是依赖于平台的节点数是固定不变和局域网内的高速带宽环境。

针对这些问题,本文提出了一个用于广域网络范围内的并行编程模型 WAPM(Wide Area Programming Model),它为应用的分布式计算的编程提供了一个新的可行解决方案。以分布式计算平台 P2HP^[9]为工作环境,描述了 WAPM 的基本特点和应用的实施过程。

1 WAPM 的组成

WAPM 由通信库、通信协议、应用编程接口组成,如图 1 所示,左边是 WAPM 的模块组成,它使用通信协议等给分布式计算平台提供各种代码与数据的管理和应用编程的支持。

收稿日期:2009-02-16;修回日期:2009-04-09。 基金项目:国家自然科学基金资助项目(50577027)。

作者简介:付崇国(1967-),男,辽宁大连人,副教授,主要研究方向:分布式计算、计算机网络; 徐胜超(1980-),男,湖北武汉人,讲师,硕士,CCF 会员,主要研究方向:移动通信增值业务开发、并行与分布式计算、对等高性能计算。

1.1 通信库

通信库是一个支持的各种类型节点相互通信的库,在分布式计算平台中,为了协同完成一个并行应用的求解,Internet 环境下的节点(这种节点称为志愿机)往往都要扮演不同的角色,包括保存代码的后台数据服务器节点,专门负责执行任务的计算节点,还有控制应用流程的主控节点等等,如 Javalin^[10] 计算平台的节点有 Broker, Client, Host 的角色, JAVM^[11] 计算平台的节点都扮演了 Volunteer, Coordinator, Director, Client 的角色等。通信库就是为了实现各种角色节点之间交互信息服务,这些信息往往分为三类:科学计算应用程序的数据参数文件、科学计算应用程序的代码文件、平台内部各个节点的硬件信息与软件信息。为了使 WAPM 更好地使用广域 Internet 环境,通信库可以分为广域通信库、本地通信库、消息通信库几类。这一点和 Sun 公司的 JXTA^[12] 协议通信库有点类似,但是 JXTA 协议是基于对等网络的,它更适用于文件共享网络,而 WAPM 的设计是基于节点角色的,更适用于分布式科学计算中的子任务的划分与调度。

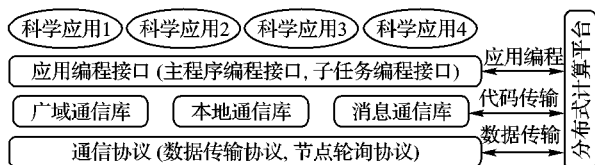


图1 WAPM 的组成结构

1.2 通信协议

通信协议是设计好的专门应用于各个角色节点通信所遵守的规则。对于代码与数据的传输,我们使用了数据传输协议,目前网络上的 FTP 协议可以实现这种数据传输。为了简化这种协议的实现,我们对 FTP 协议进行一定的改进,即在各个角色节点通信时,把所有请求封装成一个消息包,消息通信库只是提供消息请求服务和消息响应服务,当各个角色的节点需要请求数据的时候,就发送 Send 或者 Receive 这样的消息到存储代码和数据的节点,然后被请求的数据就通过 TCP 网络连接在节点和数据存储节点之间进行传输。

平台内部节点的硬件信息与软件信息的传输与监控,采用了节点的轮询协议。这个与网格计算平台比较类似,节点的硬件信息包括节点的 ID 号、IP 地址、端口号、CPU 主频、内存大小、CPU 利用率、可用磁盘空间等。节点软件信息是与任务相关的,往往包括节点正在计算的子任务数、完成的子任务数、节点的负载、是否可以接受新任务等信息。轮询协议规定分布式计算平台内有需要的节点每隔一个时间周期互相轮询状态,获取它们的状态信息,判断其是否离开或者失效,该协议也可以用于节点的信息监控,这样分布式计算平台的应用程序开发人员与用户才更加容易了解系统的状态,这种功能类似于分布式系统中的“心跳”(Heartbeat)机制。若连续几个时间周期所轮询的节点没有向管理自己的后台节点发送消息,则认为节点已经退出计算平台或者失效,那么将采用相应的容错机制保证整个计算过程不受影响。在实现轮询技术时,可以把要通信的节点信息封装成序列化对象类在网络上传输与通信。

1.3 应用编程接口

应用编程接口是一系列的函数,和 MPI 的编程模型类似,MPI 中有六个核心的函数使得 MPI 所有的通信功能可以用它的六个基本的函数调用来实现。WAPM 的应用编程接

口函数也有最核心的几个函数,是进行 Master-Worker(主-从)模式分布式计算所必须用到的,包括用于编写主程序的函数和编写子任务程序的函数。如果有更加复杂和更加专业的应用,WAPM 还可以依据分布式计算平台而开发出更多的接口函数。

主程序用于控制整个应用的执行,WAPM 中主程序编程接口包括的函数有 NewSubTask(), GetTaskStatus(), GetResult(), UpdateTaskData()。

NewSubTask(): 新申请子任务。该函数执行后,主程序就给应用申请了多个子任务,子任务的数目由程序员设置。然后分布式计算平台就接受到该申请的要求,并根据其平台的状况,把子任务分派给负载比较小的志愿机节点进行计算,这就是自适应并行的体现。

GetTaskStatus(): 获取子任务的运行状态。分布式计算平台监控子任务的运行状态,子任务的状态往往包括正在运行、已完成、已经准备好等。

GetResult(): 获取子任务的计算结果。志愿机子任务计算完后,会直接把结果发送到存储数据的节点中,主程序中执行了这个函数后,将从平台的数据存储节点中获取子任务的子计算结果,分析与判断是否应该结束循环,继而为进行下一步的处理做准备。

UpdateTaskData(): 在数据存储节点中更新子任务的计算结果。这个函数是主程序执行过程中用来动态地创建子任务。

子任务编程接口用于编写子任务程序,当 Internet 环境下的一个志愿节点完成子任务后,就把计算的临时结果发送到存储代码的后台数据节点中。由于子任务之间是互相独立的,所以分布式计算平台中的志愿节点之间很少或者没有通信。目前 WAPM 中还不支持有相互交互子任务结果的函数,如何开发出功能更加强、灵活性强的子任务交互的编程接口函数是以后该领域的一个重要研究方向。

WAPM 中子任务编程接口函数包括: SetDataFile(), SetDataFileFolder(), SetTaskState(), SendResult() 等。

SetDataFile(): 指定子任务所需的数据文件的文件名。开发员在编写应用程序的时候,需要根据存储数据节点的目录结构,对子任务数据和结果进行处理。该函数执行后,可以指定数据存储节点中子任务运行所需要的数据文件的文件名。利用这些参数数据或者数据文件,子任务才可以独立的运行。

SetDataFileFolder(): 指定子任务所需的数据文件的文件目录。和 SetDataFile 函数有点类似,这个函数是指定子任务运行所需要的数据文件的文件目录。这样子任务参数数据就有文件和文件目录两种形式,方式比较灵活,可以使分布式计算平台适合多种类型的应用。

SetTaskState(): 指定子任务的状态。在子任务的执行过程中,把它自己的运行状态保存起来,这样就方便主程序控制循环,平台也可以监控各个子任务的运行状态。

SendResult(): 发送子任务结果。这个函数是 WAPM 用来实现 Master - Slave 风格的并行应用中最核心的,它把每个子任务的结果在计算完后直接发送到后台数据节点中保存起来,随时准备提供给主程序获取和更新。

2 WAPM 的实现

2.1 通用编程的实现

通过分析与研究,对于 Master-Worker(主-从)模式的并

行应用,可以根据主程序和子任务程序之间的数据独立性和数据传输方向进行分类,每一类科学问题对应一种任务关系图,包括静态任务关系图和动态任务关系图。如图2所示,箭头表示数据的传输方向。图2(a)是一个静态任务关系图,这个图在并行应用的执行过程中保持不变。图2(b~d)都是动态的任务关系图,它们在并行应用的执行过程中的不同阶段不断变化。在图2(b)中,数据可以从子任务传送到主程序;在图2(c)中,数据可以在主程序和子任务程序之间互相传输并且还支持子任务之间的递归调用;在图2(d)中,数据可以在主程序和子任务程序之间互相传输,也可以在子任务之间传输。对于目前分布式计算平台上的应用来说,密码破解类应用和 SETI@home 应用的任务关系图可以分别认为是图2(a)和图2(b)。生物信息学的基因序列比对、全球天气变暖原因分析、解方程组的迭代类的应用,它们的任务关系图可以认为是图2(c),这种应用的任务划分方式相对复杂。

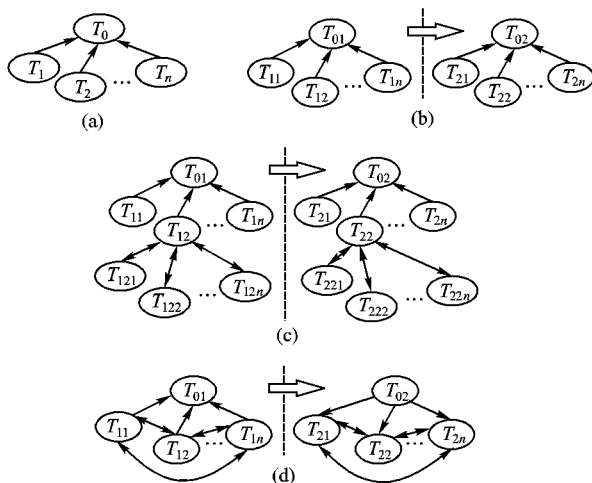


图2 任务的关系图

WAPM 编程模型为了支持上面所叙述的这些应用,在应用编程接口的设计上基本考虑到了。对于图2(a)类与图2(b)类应用,主要是要实现主控节点(Master)与从节点(Slave)之间收集与分发数据,在主程序编程接口的 GetResult(), GetTaskStatus() 中完成了主控节点(Master)到从节点收集数据的过程,而在 SendResult(), SetTaskState() 中实现了由从节点(Slave)到主控节点(Master)上提交数据的功能。但是值得注意的是,本设计中的这些函数发送和收集数据的过程利用了一个后台数据存储节点作为中转站,主控节点(Master)和从节点(Slave)的交互都是通过到后台数据节点中取或者存数据来完成,这些已经在1.3节中有详细的描述。

任务关系图是图2(c)的应用,要求能够完成子任务之间的递归调用,例如解方程组的迭代类的应用,数学模型是方程组的应用(结构静力分析计算,中尺度数值天气预报,电力系统潮流计算等)是很多的,要解这些方程组一般采用迭代法,而迭代法中循环终止的条件是其中若干个物理量满足一定的精度要求。在划分这种应用的任务的时候,不能像密码破解类应用一样简单把主程序划分成许多的子任务,而是只能做一个起始的划分,这是因为一个子任务必须获取其计算所需要的所有数据。若某一应用的计算是渐进获取数据的,在某一时刻只有部分数据,则该过程不能编程为一个子任务,而只能将这部分数据所对应的计算过程编程为一个子任务。每个子任务进行计算,每一次迭代转化后,会生成若干个新的子任

务,经过数次迭代后,任务关系图会很大。所以主程序执行过程中要动态地创建子任务,在旧的子任务计算完后,新的子任务必须不断地创建,以便来适合这个应用的运行状况。目前 WAPM 的编程接口中的 NewSubTask(), UpdateTaskData() 等就实现了这些功能。任务关系图是图2(d)的应用 WAPM 目前还暂时不能支持。

WAPM 另外的一些指定子任务的数据文件的函数 SetDataFile(), SetDataFileFolder(), 也考虑到真实世界的各种应用的不同,使用的数据文件既有文件的形式也有文件夹的形式,所以这样分布式应用程序开发人员可以使用 WAPM 进行更加多类型的并行应用编程开发。

2.2 自适应并行

分布式计算平台中的自适应并行往往指的是系统能够把子任务分发到负载最小或者最适合的志愿节点上进行计算,使得平台的效率和性能达到最佳。WAPM 中的自适应并行主要是由支持节点信息监控的通信库和分布式计算平台 P2HP 中的基于节点计算能力的任务调度算法来共同完成的。

前面已经提到,节点在通信与监控时传输的信息包括节点硬件信息和软件信息。在大规模 Internet 环境下,基于关键字的交互与通信^[13]的分布式哈希表在文件与数据共享平台中使用得很多,但是在分布式科学计算平台下往往不是很适用。节点的计算能力与 P2P(Peer-to-Peer)平台中的关键字有很大的不同,是很难用一个关键字来表示的,它是由志愿机的硬件资源和软件资源各种因素共同决定,而且在不停地变化,这样基于节点计算能力的任务调度就显得更加复杂。WAPM 中通过建立数学模型,可以把节点的计算能力利用一个包括硬件信息和软件信息的元组来描述,完成自适应并行的时候,在通信库里提供更复杂的通信原语,就可以每次把子任务调度给计算能力最强的节点^[14-15]。

2.3 容错特性

WAPM 的容错特性体现在两个方面:子任务级的容错机制与通信库里的消息容错机制。

子任务级的容错机制的好处是在不降低系统性能的情况下将容错过程简化,降低容错过程中的系统开销。该机制假设分布式计算平台中后台的数据存储节点、主控节点等都是比较稳定的节点,而 Internet 环境下的志愿机是不稳定的,可能随时会加入和退出平台,容错的处理主要针对志愿机节点。当一个志愿节点离开或者异常退出计算平台时, WAPM 灵活的广域通信库能够自动把还没有计算完的子任务结果转移到其他的志愿机节点上,从而使整个计算平台的结果不受影响。在 WAPM 编程模型中,子任务是最基本的编程单元,只要记录好子任务的 ID 号、子任务在数据存储节点里对应的数据文件、执行子任务计算的志愿机 ID 号等信息,基本可以确定该编程单元的所有信息。

分布式计算平台在调度一个子任务到志愿机节点进行计算的时候,志愿机节点将从后台数据存储节点获取该类型的子任务的描述文件和子任务 ID,并投入计算。数据存储节点将记录此类型子任务在该组数据上所分配的子任务 ID 和志愿机节点 ID,并保存在数据库中。当志愿机节点在计算子任务的过程中出现故障,无法继续工作时, WAPM 的通信库按如下方法处理:

1) 节点轮询协议发现志愿机节点失效后选择另外一个

空闲的志愿机节点;

2) 计算平台向选定的志愿机节点发送重新计算的子任务 ID 和子任务描述文件;

3) 志愿机节点根据子任务描述文件,向后台数据节点发出请求,重新申请计算的子任务 ID;

4) 后台数据存储节点根据重新计算的子任务 ID,找到对应的数据分配信息,将原分配的子任务输入数据重新发送给接替工作的机器;

5) 志愿机节点向计算平台返回信息,报告已经正确恢复原来子任务的计算。

在消息容错机制中,WAPM 利用通信双方的状态同步机制进行容错,由被连接的节点探知通信状态并通过消息信道通知主动连接方,然后双方在相同状态下采取协同机制排错。

3 WAPM 分布式计算编程

本章以分布式计算平台 P2HP 为背景,描述了利用 WAPM 编程模型进行分布式应用程序开发的实施过程,证明 WAPM 是一个可行的并行程序设计编程模型。P2HP^[9]是一个基于协同服务器组的分布式计算平台,P2HP 通过 WAPM 编程模型提供的并行应用开发接口,就可以完成应用的分布式计算。由于 P2HP 计算平台采用 Java 语言开发,所以编程模型 WAPM 体现在程序员面前的表现形式是几个 jar 包,包括应用编程接口 BioSDK.jar、通信库的包 ComLib.jar、数据通信协议的包 DataCom.jar 等。某些 Jar 包可以集成到平台中而不用导入。在利用 WAPM 进行编程时,必须导入 BioSDK.jar 包,import cn.edu.hust.cgcl.biogrid.BioSDK.*。

选择最简单的并行计算问题,求从 1 加到 1 000 000 的总和作为 WAPM 编程模型的案例程序。目前我们已经实现了几种应用问题,选择这个最简单的数学问题作为例子的目的是让对并行计算不是很了解的人也很容易使用 WAPM。将问题分解为 10 个独立运行的子任务,即对以下 10 个子任务进行求和:[1, 100 000], [100 001, 200 000], ..., [900 001, 1 000 000]。

3.1 子任务的编程

考虑到文章篇幅的原因,以下都是一些基本流程的代码,没有考虑到异常情况的处理,进行比较小幅度的修改,就可以直接运行。子程序的代码如下:

```
import cn.edu.hust.cgcl.biogrid.BioSDK.*;
//将 BioSDK.jar 包含进来
public class usersubtask extends TaskRun
// TaskRun 是 BioSDK 包里的一个类
{ private Long result, first, last;
  public usersubtask (int first, int last)
  {
    FileReader FR = new FileReader(dataFile);
    //DataFile 是指定的数据文件名
    BufferedReader bfr = new BufferedReader(FR);
    String line = bfr.readLine();
    //读子程序对应的数据文件的一行
    first = Long.parseLong(line);
    line = bfr.readLine();
    last = Long.parseLong(line);
  }
  public void run() //run() 函数用于启动志愿机运行
```

```
{
  String filePath = "D: \testFile \result \";
  //指定志愿机节点的工作目录
  String fileName = this.subTaskId; //子任务计算结果的文件名
  for (int i = first; i <= last; i++)
    result += i;
  //SubTaskId(), GetResultPath() 等都是 TaskRun 类的成员函数
  String fn = filePath + fileName;
  FileWriter f = new FileWriter(fn);
  //文件对象来处理子任务的计算结果文件
  String convert = Long.toString(result);
  f.write(convert);
  f.close();
  String fileName = this.getSubTaskId(); //获取子任务的 ID 号
  String fn = this.getResultPath() + fileName;
  //获取结果文件存放的路径
  sendResult(this.getResultPath(), fileName);
  //将子计算结果发送到后台数据节点
  this.setTaskState(SubTask.SUBTASK_FINISH);
  //指定子任务的状态为完成
}
```

3.2 主程序的编程

主程序将计算出来的部分和进行相加。在求解该问题的过程中,子程序负责子问题的求解,主程序则负责将求解出来的子问题结果进行相加汇总处理,并最终获得整个问题的答案。主程序的代码如下:

```
import cn.edu.hust.cgcl.biogrid.BioSDK.*;
public class MainProgram
{ public MainProgram()
{ }
  public static void main(String args[])
  {
    int number = 10; //划分子任务为 10 个
    long Sum = 0; //最终结果为了 Sum
    long Temp_Sum = 0; //临时结果变量
    String filePath = "D: \testFile \tempResult \";
    //子任务计算结果存放路径用于汇总
    String fileName = "";
    //取得的子任务计算结果文件名,完整路径文件
    Job job_instance = new Job(descFileName);
    //创建一个计算任务
    job_instance.monitorIp = "202.114.14.148";
    //指定后台节点 IP
    job_instance.monitorPort = 1885; //指定后台节点端口
    MainSDK MainSDK_instance = new MainSDK();
    LinkedList Linkedlist_instance = new LinkedList();
    //创建子任务列表
    MainSDK.newSubTask(number, job_instance,
      Linkedlist_instance);
    for (int i = 0; i < Linkedlist_instance.size(); i++)
    {
      SubTask tmp = (SubTask) Linkedlist_instance.get(i);
      fileName = MainSDK_instance.getResult(tmp,
        job_instance, filePath); //主程序从数据存储节点
      //获取的临时结果保存在 fileName 里
      System.out.println(" get result from dataServer succeeded!");
      //下面开始汇总计算结果:
      FileReader FR = new FileReader(fileName);
```

```
BufferedReader bfr = new BufferedReader( FR );
String line = bfr. readLine();
Temp_Sum = Long. parseLong( line );
Sum = Sum + Temp_Sum;
}
System. out. println( " The ultimate result is " + Sum );
//计算出最终结果
}
```

4 实验与结果分析

为了测试编程模型 WAPM 的性能,我们进行了一系列的实验。在实验中,把若干台 PC 机通过高速 100 Mbps 的以太网连接起来,来模拟真实的 Internet 网络环境。配合 P2HP 平台^[9],即形成了一个模拟广域范围的分布式计算环境,如图 3。各个角色节点的个数和软硬件配置如表 1。

表 1 测试环境配置

| 节点角色 | 个数 | 处理器 | 操作系统 | 内存/MB | Java 环境 |
|--------|----|-----------------------|--------------|-------|------------|
| 监控服务器组 | 1 | Intel Celeron 2.6 GHz | Windows 2000 | 512 | J2sdk1.4.2 |
| 调度服务器组 | 5 | Intel Celeron 2.6 GHz | Windows 2000 | 512 | J2sdk1.4.2 |
| 后台数据节点 | 1 | Intel Celeron 2.6 GHz | Debian Linux | 512 | J2sdk1.4.2 |
| 志愿机节点 | 40 | Intel P4 2.0 GHz | Windows XP | 512 | J2sdk1.4.2 |

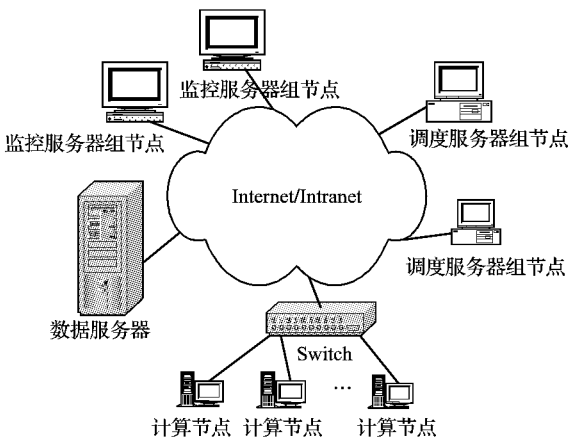


图 3 WAPM 编程模型的硬件测试环境

4.1 可行性

我们选择 RC5^[16] 密码破解算法作为测试的应用。它是一个参数变量分组算法,实际上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。在实验中所取的参数为: $w = 32, r = 12, b = 8$ (RC5_32_12_8)。在此算法中主要使用了三种运算:异或、加法和循环移位。

我们把这个并行应用的整个破解程序划分为 128 个独立的子任务进行计算,每个子任务对应一段密钥空间,根据这个空间预先设置好参数文件和数据文件,在已知明文的情况下,对密文采用逐个穷举密钥的方式来破解这个密码。

根据 WAPM 提供的主程序编程接口和子任务编程接口函数,开发员首先在后台数据节点上设置好数据文件,然后利用 NewSubTask() 函数申请 128 个子任务,P2HP 平台就把这些子任务分派到空闲的志愿机节点上进行计算。在这些子任务的执行过程中,各志愿机节点通过 SetDataFile() 函数从数据存储节点中获取其运行所需要的数据文件。子任务计算完后,通过 SendTaskState() 函数和 SendResult() 函数把自己的状态和子任务计算结果发送到数据存储节点中。主程序通过 GetTaskStatus() 函数来监控子任务的运行状态,并通过 GetResul() 函数来收集子任务的计算结果,并进行汇总处理。这样所有的计算节点都循环尝试密钥能否破解,任何一个计算节点通过密钥得出的结果如果与明文相同,整个程序就计算结束,从而破解了这个密码。

4.2 通用性

通用性是指 WAPM 适合于多个并行应用的部署,而不是

单独地为一个应用而设计,利用主程序编程接口和子程序编程接口,用户可以进行灵活的编写程序,进行并行应用开发。WAPM 的这些编程接口是灵活和方便的,开发一个并行应用程序和串行程序一样简单,编程语言也是程序员所熟悉的 Java 或者 C++,编程的思想也不需要 MPI 或者 OpenMP 的并行思想。除了 RC5^[16] 密码破解算法外,利用 WAPM 还实现了生物多序列比对 ClustalW 算法^[17]、蛋白质折叠 HP 模型、电力系统潮流计算^[18]等。其结果见表 2。

表 2 WAPM 的通用编程测试结果

| 并行应用 | 源代码的语言 | 应用类型 | API 是否嵌入代码里 |
|----------------|--------|-------|-------------|
| RC5 密码破解 | Java | 计算密集型 | 否 |
| 多序列比对 ClustalW | C | 计算密集型 | 是 |
| 蛋白质折叠 HP 模型 | C++ | 计算密集型 | 否 |
| 电力系统潮流计算 | C++ | 计算密集型 | 否 |

4.3 加速比

多机编程环境的一个最重要目标是提高计算速度,我们采用加速比作为 WAPM 的另外一个性能指标。设在计算平台 P2HP 中,单计算节点串行完成某一应用所花费时间为 $T(1)$, n 个计算节点共同并行完成某一应用程序所花费时间为 $T(n)$, 则 n 个计算节点的加速比 $Speedup(n)$ 定义为: $Speedup(n) = T(1)/T(n)$ 。

在 n 个 ($n = 8, 16, 24, 32, 40$) 计算节点的情况下测试了 RC5^[16] 密码破解算法和生物多序列比对的 ClustalW^[17] 算法的加速比,测试结果如图 4 显示。

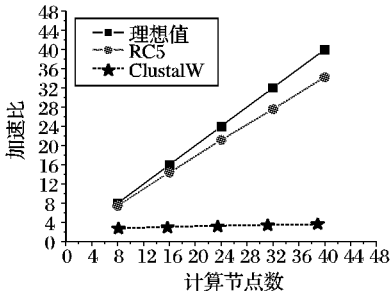


图 4 RC5 算法和 ClustalW 算法的加速比

从图 4 还可以看出,随着志愿机节点数的增加,RC5 加速比越来越大,整个程序的性能接近于理想值。主要原因是 RC5 算法这个应用是计算密集型应用,带宽需求很低,整个计算过程没有大容量的数据传输,大部分时间都花在志愿机节点的循环尝试密钥能否破解上,可以看出 WAPM 是比较适合

任务划分独立,低带宽需求,粗并行粒度的应用的。

从图 4 中可以看出,ClustalW 算法的性能提高不如 RC5 算法,加速比的增加很慢,主要原因是这个算法某个部分不容易并行处理,只处理了其中一些容易并行处理的部分。ClustalW 算法的临时比对结果有少量的大块数据传输和一定网络带宽需求,这也使它的算法性能提高不如 RC5 算法,所以带宽稍微要求高的应用对于 WAPM 来说性能提高不明显。

5 相关工作

在目前的广域分布式计算平台中,对应用的开发一部分是对某一个特定的应用进行编程,如 SETI@Home^[2]、GIMPS^[19]等,它属于专用计算平台,基本没有编程模型。或者是通过提出自己消息库在节点之间进行通信,并将应用编程接口函数嵌套在其中,进而支持多个应用的开发,平台与应用没有分离,如 Popcorn^[20]和 Javalin^[9]等。还有几种使用的是 RMI 或者 RPC 的机制。例如 XtremWeb^[3]是一个通用的分布式网络计算平台,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^[21]的方式来实现子任务的编程,把 Remote Procedure Call (RPC) API 作为它的编程模型的应用编程接口。BOINC^[4]提供一系列的由 C++ 来实现的 API 函数供客户端开发或客户端的图形界面开发,使用 HTTP 或者 FTP 协议的方式管理代码与数据。

还有的是利用对等网络环境下的通信库 JXTA^[12]进行志愿机节点间的通信,如 JNGI^[5]和 P3^[6],具有多应用支持和容错等特点,但是其编程接口函数对于程序员使用起来很复杂,也没有提出一个通用的编程模型来支持科学计算应用的开发。

在目前并行分布式计算领域,最常用的是共享内存编程模型 OpenMP^[8]和消息传递编程模型 MPI^[7],它们主要用于共享内存并行机和集群并行机。本文提出的 WAPM 提供给用户一个适合广域范围内的分布式计算的并行编程模型,其应用编程接口函数调用是基于消息通信协议机制的,而不是远程过程调用机制,适合于桌面 PC 机的分布式计算,具有价格低廉、编程方便、容易构建等特点。

6 结语

WAPM 是一个适合广域分布式计算的并行编程模型,它具有通用编程、自适应并行、容错性等特点,提供给用户一套简单的应用编程函数接口,包括主程序编程接口函数和子任务编程接口函数,如果配合上分布式计算平台和适当的编程语言,即形成了一个广域并行程序设计环境,为科学应用的分布式计算编程提供了一个新的可行解决方案。从实验结果分析显示, WAPM 是一个可行的、通用的、性能比较好的编程模型。下一步工作是优化通信库,并在这个通信库的基础上开发出更加灵活和方便的应用编程接口函数,进一步扩大该计算平台的应用范围。

参考文献:

- [1] SARMENTA L F G. Volunteer computing [D]. Cambridge, MA, USA: Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2001.
- [2] SETI@Home Project [EB/OL]. [2009-02-01]. <http://setiathome.ssl.berkeley.edu/>.
- [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. Washington, DC: IEEE Computer Society, 2001: 582-587.
- [4] BOINC Project [EB/OL]. [2009-02-01]. <http://boinc.berkeley.edu/>.
- [5] VERBEKE J, NADGIR N, RUETSCH G, *et al*. Framework for peer-to-peer distributed computing in a heterogeneous and decentralized environment [C]// Proceedings of third International workshop on Grid Computing (Grid 2002), LNCS 2536. London: Springer-Verlag, 2002: 1-12.
- [6] SHUDO K, TANAKA Y, SEKIGUCHI S. P3: P2P-based middleware enabling transfer and aggregation of computational resources [C]// 5th GP2PC. Washington, DC: IEEE Computer Society, 2005: 259-266.
- [7] LUSK E. MPI-2: Standards beyond the message-passing model [C]// Proceedings of 3rd Working Conference on Massively Parallel Programming Models. Washington, DC: IEEE Computer Society, 1997: 43-49.
- [8] SKILLICORN D B, TALIA D. Models and languages for parallel computation [J]. ACM Computing Surveys (CSUR), 1998, 30(2): 123-169.
- [9] 徐胜超, 金海, 章勤, 等. 基于协同服务器组的志愿者计算环境的构造 [J]. 计算机研究与发展, 2007, 44(3): 384-391.
- [10] NEARY M O, PHIPPS A, RICHMAN S, *et al*. Javelin 2.0: Java-based parallel computing on the Internet [C]// 6th International Euro-Par Conference (Euro-Par 2000), LNCS 1900. Berlin: Springer-Verlag, 2000: 1231-1238.
- [11] LAU L F, ANANDA A L, TAN G, *et al*. JAVM: Internet-based parallel computing using Java, series on scalable computing - Vol 2, Annual review of scalable computing [M]. Singapore: World Scientific, 2000: 59-74.
- [12] GONG L. JXTA: A network programming environment [J]. IEEE Internet Computing, 2001, 5(3): 88-95.
- [13] ZHAO B Y, HUANG L, STRIBLING J, *et al*. Tapestry: A resilient global-scale overlay for service deployment [J]. IEEE Journal on Selected Areas in Communications, 2004, 22(1): 41-53.
- [14] 韩宗芬, 江峰, 章勤, 等. 一种基于对等网络高性能计算的任务调度算法 [J]. 华中科技大学学报, 2007, 35(3): 31-34.
- [15] 徐胜超, 金海, 石柯. 一种用于志愿者计算的层次消息通信网络 [J]. 小型微型计算机系统, 2008, 29(6): 1036-1041.
- [16] HEWGILL G. RC5 and Java Toys [EB/OL]. [2009-02-01]. <http://www.hewgill.com/rc5/index.html>.
- [17] THOMPSON J D, HIGGINS D G, GIBSON T J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice [J]. Nucleic Acids Research, 1994, 22: 4673-4680.
- [18] 徐胜超, 方华亮, 曹芳. 志愿者计算模型在电力系统潮流计算中的运用 [J]. 计算机工程与应用, 2007, 43(9): 189-192.
- [19] Great Internet Mersenne Prime Search Project [EB/OL]. [2009-02-01]. <http://www.mersenne.org/>.
- [20] CAMIEL N, LONDON S, NISAN N, *et al*. The popcorn project: Distributed computation over the Internet in Java [C/OL]// Proceedings of the 6th International World Wide Web Conference. 1997: 7-11 [2009-02-01]. <http://www.ra.ethz.ch/cdstore/www6/Posters/721/poster.html>.
- [21] DJILALI S. P2P-RPC: Programming scientific applications on peer-to-peer systems with remote procedure call [C]// Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2003: 406-413.