第 24 卷第 6 期　　四川理工学院学报(自然科学版)　　Vol. 24　No. 6
2011 年 12 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Dec. 2011
文章编号:1673-1549(2011)06-0686-05

云计算在素性判定中的应用

王　娜,范安东,黄　敏,李小伟

(成都理工大学管理科学学院,成都　610059)

摘　要:云计算的出现给人类带来了又一次的信息巨变,云计算的海量数据处理和虚拟资源可以帮助我们解决很多数学难题。主要利用云计算的分布式计算框架和虚拟技术研究其在素性判定中的应用,并且针对三种典型的素数判定定理提出了这些素性判定方法和云计算结合的方案,同时给出了相应的实验数据和结果分析。

关键词:云计算; PaaS; IaaS; 试除法; 拟素数; 素性制定; 梅森素数
中图分类号:TN918.1　　文献标识码:A

引　言

云计算的兴起席卷了 IT 界,带来了第三次信息浪潮。在分布式计算的基础上,把网格计算商业化,把资源像水电一样提供给用户使用,这就是云计算。云计算的高吞吐量、海量存储和大量数据处理能力使得社会各界争相使用云计算。

素数,计算数论中一颗璀璨的明珠,在信息安全和计算机中扮演着举足轻重的角色,同时,它也是一个一直困扰数学家的古老而又复杂的难题。目前最流行的素性判定方法就是 GIMPS 全球搜索素数用到的 Lucas - Lehmer primality testing。素性判定的困难之处在于海量数据计算,恰巧海量数据处理也是云计算的一个重要特征。如何利用云计算的分布式计算模式提高素性的判定,这是本文要处理的重点。本文先简单介绍云计算,然后详细分析了如何利用云计算提供的服务结合试除法的素性检测(Primality test by trial divisions)、强拟素性检测(Strong pseudoprimality test/Miller - Rabin test/)和卢卡斯 - 莱默素性检测(Lucas - Lehmer primality test)三种经典的素性判定方法进行素性判定。第三、四部分是在 Amazon 提供的云计算平台下进行实验,并且进行了相应的数据分析。最后一部分是对本文的一些思考和展望。

1　云计算

云计算是网格计算的商业化,是对基于网络的、可配置的共享计算资源池能够方便的随需访问的一种模式[1]。云计算分为三种服务模式,分别是 IaaS(基础设施即服务),PaaS(平台即服务),SaaS(软件应用即服务)。在文献[1]中给出了各种服务的相关概念,IaaS 是用户通过网络,按需获得包括计算机、存储、网络,以及其他相关的设施的一种服务; PaaS 是把端到端的分布式软件开发、测试、部署、运行环境和繁琐的应用托管当做服务提供给用户; SaaS 是使用互联网提供程序软件给远程用户的软件服务模式。本文主要使用 IaaS 和 PaaS 来完成素性的判定。首先简单介绍一下 PaaS 分布式编程模型 MapReduce。

MapReduce 是 Google 最早提出用来进行大数据量计算的简化的并行计算编程模型,是 PaaS 的核心,是云计算的一项关键技术,主要通过"Map"和"Reduce"两个函数来完成。MapReduce 过程中的输入映射化简输出都是以 key/value 对的形式来进行的。比如要把向量 $A=[a_1,a_2,a_3,\cdots,a_n]^T$ 变成 key/value 对,我们可以赋予这个向量一组 keys $\{N:1,2,3,\cdots,n\}$,使得每个 key n 对应一个 a_n[2]。

MapReduce 的过程大致如下:

(1) Input 输入:把定义好格式的标准输入文件中的

收稿日期:2011-11-30
作者简介:王　娜(1985-),女,江西吉安人,硕士生,主要从事信息安全中的数学方法方面的研究,(E-mail) nana37230425@yahoo.com.cn

数据切割成 Split 块,分配给多台计算机并行进行 Map 函数运算。

(2) Map 过程: Map 函数接受一组 Split 块中的数据,将其映射成 key/value 对。

(3) Reduce 过程: Reduce 接受 Map 函数产生的 key/value 对,进行化简,缩小 key/value 列表。

(4) Output 输出: 最后以标准输出格式输出结果。

当然,这只是简化版的过程,从过程中可以看出,用户需要做的主要工作就是指定输入输出格式,定义 Mapper 和 Reducer,指定 Map 阶段和 Reduce 阶段要做的工作。

Hadoop 是一个典型的分布式软件架构,其核心是 HDFS(Hadoop Distributed File System) 和 MapReduce。HDFS 由一个负责主控的 NameNode 和多个负责数据的存储和管理 DataNode 组成,MapReduce 则是由一个负责主控的 master(JobTracker) 和多个执行任务的 slave(TaskTracker) 组成[1]。其实,MapReduce 和 HDFS 都是运行在同一组节点上的。本文的素性判定要使用的 Amazon Elastic MapReduce 正是 Amazon 在 Amazon 的 IaaS 平台上部署 Hadoop 提供的 PaaS 服务。

2　利用云计算进行素性判定

最简单的素数判定方法当然就是试除法,试除法是一种确定性算法,虽然它的精确率是 100%,但是耗时太长,实际操作中不易进行。当下比较流行的是拟素性判定方法,著名的因特网梅森素数搜索(GIMPS) 就采用了 Lucas – Lehmer primality testing,这就是一种专门用来判定梅森素数的拟素性判定方法。

本文主要介绍试除法,强拟素数检测法,Lucas – Lehmer 梅森素数判定三种判定方法在素性判定中的使用。因为在梅森素数判定的过程中,我们设定梅森数 2^p-1 中的指数 p (p 为素数) 序列为输入文件,所以我们需要一个素数列表,这个列表就可以由试除法来产生。虽然试除法有点慢,但它是确定性算法,况且目前最大的梅森素数 $2^{43112609}-1$ 的指数也只有 8 位,加上云计算的并行运算,产生由指数构成的输入文件还是比较容易的。

2.1　试除法的素性检测

定理 1[3]　令 $n>1$。若 n 没有不超过 $\sqrt{n}$ 的素因子,则 n 为素数[3]。

这个算法虽然可行性不强,但是因为是确定性算法中比较古老的算法,并且定理易于理解,所以在计算机领域还是会用到的。这个算法需要 $O(2^{(\log n)/2})$ 比特运算量。虽然编写单机环境下用试除法判定 n 是否为素数的程序很简单,但这个算法不仅耗时长,而且对机器配置的要求也高。云计算的分布式计算技术就是利用大量的廉价计算机组成的集群来分块解决大数据处理问题。在 MapReduce 中使用试除法判定 n 是否为素数的主要思路如下:

(1) Input 输入: 输入文件就是

$$\{2,3,5,\cdots,2k+1,\cdots,N\}$$

其中 N 为小于等于 $\sqrt{n}$ 的最大奇数。Hadoop 框架会把这些数据切分成 split 块,分配给各个 slave 进行 Map 的。

(2) Map 过程: Mapper 中让 n 去除以 $2k+1$,如果除得尽就让 key 为 composite,value 值为 1; 如果除不尽就让 key 为 prime,value 值也为 1。

(3) Reduce 过程: 把 Map 过程中输出的 key/value 对中具有相同 key 的 value 合并在一起。其实这里的 key 只有两种可能,一种是 composite,另一种就是 prime。

(4) Output 输出: 输出的 key/value 对中,如果 composite 的值为 0,则 n 为素数,否则,为合数。

当然,这个算法存在改进的余地,首先,在 1) 中,输入文件可以改成 2 到 $\sqrt{n}$ 之间的素数,会减少很多次运算,我们这里主要是想看看云计算环境下和单机环境下的区别,就没做改进。其次,可以添加一些步骤,使得只要有一个 slave 出现 composite 的值为 1,就马上停止 Map 过程,直接把现有的键值对进行 Reduce,同样可以节省很多时间。

2.2　强拟素性检测

定理 2[3]　设若 n 是一个奇数,并且把 n 写成 $n=1+2^jd$ 的形式,其中 d 是奇数。我们把

$$\{b^d,b^{2d},b^{4d},b^{8d},\cdots,b^{2^{j-1}d},b^{2^jd}\} \bmod n \tag{1}$$

称作 Miller – Rabin 序列。对于任意一个满足 $1<b<n$ 的 b, Miller – Rabin 序列有如下两种形式

$$(1,1,\cdots,1,1,1,\cdots,1) \tag{2}$$

$$(?,?,\cdots,?,-1,1,\cdots,1) \tag{3}$$

满足上面定理的 n 称作是以 b 为底的强概素数。如果 n 是一个合数,则把 n 称作是强伪素数[1]。

假设 $p(x)$ 是任意一个奇整数 x 是合数的概率, $s(x)$ 是 x 为强伪素数的概率,则

$$s(x)=4^{1-k}p(x)/(1-p(x)) \tag{4}$$

可以看出当 x 趋近于无穷的时候, $s(x)$ 趋近于 0。这就是说, x 越大,强伪素数算法越好[4]。强拟素数算法明显比试除法快了很多,那我们就来数出 $[x_1,x_2]$ (其中 x_1,x_2 为整数) 区间范围内总共有多少个素数,同样的,列出 MapReduce 思想:

(1) Input 输入: $[x_1,x_2]$ 中的所有整数,Hadoop 切分数据之后发送给 slave 执行 Map。

(2) Map 过程: Mapper 里写入强拟素数的测定法

则,满足定理则让 key 为 base - b strong probable prime,value 为 1,不满足则让 key 为 composite,value 为 1。

(3) Reduce 过程:同试除法,把具有相同 key 的 value 合并在一起。其实这里的 key 也只有两种可能,一种是 composite,另一种就是 base - b strong probable prime。

(4) Output 输出:输出键值对中 base - b strong probable prime 的 value 值就是这个区间范围内素数的个数。

当然,把程序改改也可以直接输出所有的素数。虽然拟素性判定方法比试除法快,但是会存在判定错误的现象,不过,错判的概率还是比较小的。

2.3 卢卡斯 - 莱默素性检测

定理 3[5] 令 $L_1 = 4$

$$L_{n+1} = (L_n^2 - 2) \bmod (2^p - 1) \quad (5)$$

$2^p - 1$ 是素数当且仅当 $L_{p-1} = 0$[5]。

2008 年 8 月 23 号,GIMPS 用 Lucas - Lehmer primality testing 为我们找到了第 47 位素数,是目前找到的最大的梅森素数,也是至今找到的最大的素数。大部分梅森素数都是用 Lucas - Lehmer primality testing 找出来的。那如何用云计算来搜寻梅森素数呢? GIMPS 通过自愿者加入的方式进行分布式计算,从而达到寻找梅森素数的目的,云计算带来了巨大的虚拟资源库,也带来了具有超强计算能力的计算机集群,相信这些对我们寻找梅森素数可以提供一定的帮助。我们来看看如何用虚拟机和分布式计算资源寻找 $[2^{p_1} - 1, 2^{p_n} - 1]$(其中 p_1, p_n 为素数)之间的梅森素数。

当然,首先我们要进行一个简单的 MapReduce 过程把 $[p_1, p_n]$ 之间的素数列表做出来。把 $\{p_1, p_2, \cdots, p_j, \cdots, p_n\}$ 作为输入文件,Mapper 程序为用试除法判定输入的 p_j 是否为素数,如果为素数,p_j 为 key,value 为 1,否则,key 为 composite,value 为 1。Reducer 进行化简输出就可以得到 $[p_1, p_n]$ 之间的所有素数。

(1) Input 输入:用试除法找出 $[p_1, p_n]$ 之间的所有素数

$$\{p_1, p_2, \cdots, p_k, \cdots, p_n\}$$

作为此次搜寻的输入文件。

(2) Map 过程:Mapper 里面写入 Lucas - Lehmer primality testing 需要满足的条件。如果指数为 p_k 时,$L_{p_k-1} = 0$,那就让 key 为 $2^{p_k} - 1$,value 为 1,否则,让 key 值为 false,value 值为 1。

(3) Reduce 过程:把具有相同 key 的 value 合并在一起,这里要么全是 false,要么出现梅森素数。

(4) Output 输出:除了值为 flase 的 key,其他的 key 都是梅森素数。

当然,需要一提的是,在写 Mapper 函数时,可以利用 GIMPS 中给出的计算大数平方的方法 fft,可以使计算复杂度从 $O(n^2)$ 降到 $O(n\log_2 n)$[4]。

3 在 Amazon Web Services 里面实现素性判定

Amazon Web Services 提供了很多云计算服务,本文主要使用 EC2[7]、S3[8] 和 Elastic MapReduce[9],前两者属于 IaaS 服务,Elastic MapReduce 属于 PaaS 服务。值得一题的是,大数的输入问题必须先解决。小素数的判定不需要调用 Hadoop[10] 框架,在单机上就可以判断出来,如果担心自己的计算机配置太低,可以使用 EC2 调用远程虚拟机完成。数太大,用单机就很难完成了,现在研究的素数都达到了很多位数,目前找到的最大的素数如果按正常字符大小写出来的话可以长达 50 公里,一般的数据类型已经满足不了素性判定了,所以我们准备调用 gmp 软件包来解决大数的输入问题。

在 Amazon Web Services 上使用 MapReduce,要先用 C++ 生成 Input 文件,编写好 Mapper,Reducer,建立好安全认证文件,然后就可以在 Elastic MapReduce 创建 Job Flow 了,这里选取 Streaming 类型的 Job Flow。创建的时候要先在 Bootstrap Actions 里面设置好 Configure Daemons,Configure Hadoop 等基本配置文件。还要注意在 Extra Args 里面输入 gmp 所需要的 cacheFile,方便处理大数。当然如果 Input 文件超过 2M,就不能直接上传到 s3,我们只需要用 SSH 进入 Master Node,用 C++ 程序在 Master Node 里面生成一个文件,然后上传到 s3。考虑到经费的问题,这里申请的实例均为 m1,small 类型。

4 结果分析

4.1 试除法判定 14489963930079771 是否为素数

表 1 是试除法判定 14489963930079771 是否为素数的运行时间统计表,Slaves 数量为 1 时,直接通过本地 Linux 系统远程连接到 EC2 提供的 m1。small 虚拟 Linux 系统上进行判定,费时 2060 秒。Slaves 数量为 5 的数据是通过 Elastic MapReduce 服务,在 Amazon 搭建的 Hadoop 框架上跑 5 台虚拟机得出的。

表 1 试除法判定素数的运行时间函数

Slaves 的数量	实际计算时间(s)	网络数据传输时间(s)	总时间(s)	实际计算时间与总时间之比
5	1328	108	1436	0.924

可以得出以下结论:

(1) 不管是实际计算时间还是总时间,Slaves 数量为 5 时使用的时间比在单击上运算使用的时间缩减了很多。这是因为多台 Slaves 虚拟机分担了计算任务。

(2) 在 Hadoop 上跑实例,还应考虑到网络数据传输的时间消耗,也就是 Hadoop 框架的启动配置和文件的输入输出等等消耗的时间,所以如果任务本身耗时较小,那使用 Hadoop 执行的意义就比较小了。

(3) 一台虚拟机运行时间为 2060 秒,那 5 台并行计算应该只要花费 412 秒,可是为什么实际使用的时间为 1328 秒呢? 这里需要注意的是,分布式计算并不是简单的叠加,协同设计的复杂性告诉我们并不能这样假设。

当然判定的结果是 14489963930079771 为合数。

4.2 强拟素性检测统计 2147483647 到 2500000000 之间素数的数量

图 1 和图 2 表示的是用强拟素性检测方法统计 2147483647 到 2500000000 之间素数的情况。为了增加对比度,我们在 Hadoop 框架上分别调用了 1 台,5 台,10 台实例做 Slaves 机。图 1 是强拟素性检测统计 2147483647 到 2500000000 之间素数的数量的速度统计图,图 2 是对应的实际计算时间统计图。我们把在 1 台实例上执行任务花费的总时间与在多台实例上执行任务花费的总时间的比定义为速度[5]。

从上面的结果可以分析得出以下结论:

(1) 秉承分布式计算的优点,在一定范围内,随着 Slaves 数量的增加,运行时间会大大减少,速度自然就会增加。

(2) Slaves 的数量和速度的关系并不是呈线性,这是因为随着 Slaves 数量的增加,实际计算时间会减少,总时间中网络数据传输时间占的比重就会增大,这样就会影响到速度的线性。

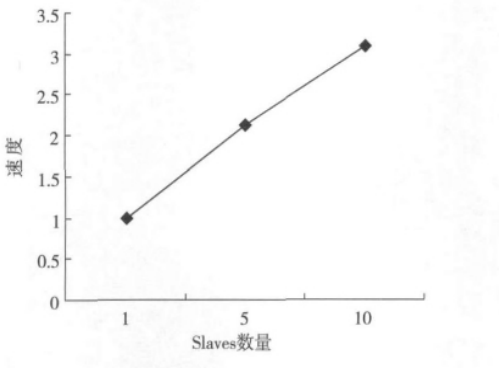

图 1　强拟素性检测的速度

4.3 Lucas – Lehmer primality testing 搜索指数在 1000 到 6000 之间的梅森素数

同样的在 Hadoop 框架上分别调用了 1 台,5 台,10 台实例。用 Lucas – Lehmer 方法得到指数在 1000 到 6000 之间的梅森素数有: $2^{1279}-1$, $2^{2203}-1$, $2^{2281}-1$, $2^{3217}-1$, $2^{4253}-1$, $2^{4423}-1$。图 3 是实验耗时统计图。

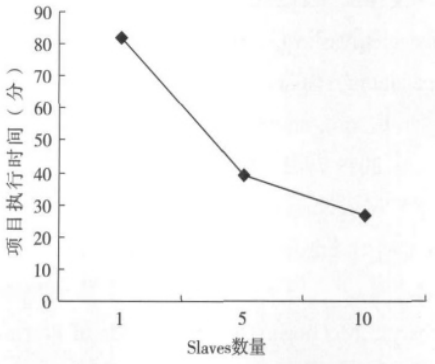

图 2　强拟素性检测的执行时间

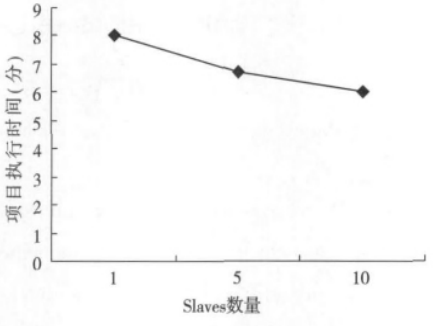

图 3　Lucas – Lehmer primality testing 项目执行时间

从上面的结果可以分析得出以下结论:

(1) 在一定范围内,随着 Slaves 数量的增加,实际计算时间的确有所缩减,但是似乎缩减得并不明显。

(2) 本文只是把指数进行分布式分配,梅森素数的判定定理其实是一个迭代过程,计算复杂性主要体现在大数的平方运算,这个问题可以借鉴 GIMPS 中的用 ffs 的处理方法来降低计算复杂度。

5　结束语

云计算还处在发展阶段,很多方面做得不够成熟,但云计算理念必然成为今后信息产业发展的趋势。海量计算一直是素性的判定的关键问题,虽然云计算对素性的判定起到了一定的推动作用,但需要改进的地方还很多。不管是在改进数学定理方面努力也好,还是在计算机算法上下功夫也好,相信有了云计算的加入,素数的判定会发展得更快。

参 考 文 献:

[1] 雷万云. 云计算技术、平台及应用案例 [M]. 北京:

清华大学出版社,2011.

[2] Somesh Srivastava, A. Ravishankar Rao, Vadim Sheinin. Accelerating statistical image reconstruction algorithms for fan-beam x-ray CT using cloud computing [J]. Proc. of SPIE, 2011, 7961(34): 1-8.

[3] Song Y. Yan. Number Theory for Computing Second Edition [M]. Germany: Springer-Verlag Berlin Heidelberg, 2002.

[4] Zachary S. McGregor-Dorsey. Methods of Primality Testing [J]. MIT Undergraduate Journal of Mathematics. 133-141.

[5] Bruno Deschamps. Sur les bonnes valeurs initiales de la suite de Lucas-Lehmer [J]. Journal of Number Theory, 2010, 130: 2658-2670.

[6] 伍 楠,吴 伟,文 梅,等. 梅森素数并行求解算法的流式实现 [J]. 计算机工程与科学,2007,29(11): 53-59.

[7] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2.

[8] Amazon Elastic Simple Storage. http://aws.amzon.com/s3.

[9] Amazon Elastic MapReduce. http://aws.amazon.com/elasticmapreduce.

[10] Apache Hadoop. http://hadoop.apache.org.

Applications of Cloud Computing in Primality Test

WANG Na, FAN An-dong, HUANG Min, LI Xiao-wei

(Management Science College, Chengdu University of Technology, Chengdu 610059, China)

Abstract: The emergence of cloud computing brought another information challenge, and large datasets processing and virtual resources can help to solve many mathematical problems. The distributed framework and Virtual Reality Technology are used to do the primality judgment. Using cloud computing, the methods and solutions are given to test the primality with three different theorem, and then the corresponding experimental datum and results are analysized.

Key words: cloud computing; PaaS IaaS; primality judgment by trial divisions; pseudo-primality; Mersenne prime