新农村信息化建设中的负载均衡问题研究

丁仁源,李 旸,马玉宝
(安徽农业大学 网络工程系,安徽 合肥 230036)

摘 要:在新农村信息一体化平台系统中,应用数据挖掘技术实现负载平衡。即根据访问的历史记录,以某种方式预测出即将到来的下一个访问请求,并按照一定的规则进行预读处理,以此平衡负载并加快访问速度。同时,运行在服务端的监控进程会定期收集农户的学习历史以及集群内各个节点的负载信息,通过对这些数据进行分析和挖掘,对服务器集群内的各种资源进行合理优化和重新配置。

关键词:数据挖掘;负载平衡;响应时间
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2009)04-0234-05

Research of Load Equalization Question in Information Construction of New Countryside

DING Ren-yuan, LI Yang, MA Yu-bao
(Department of Network Engineering of Anhui Agricultural University, Hefei 230036, China)

Abstract: In the platform for the integration of information systems in new rural areas, the application of data mining technology to achieve load balancing. That is, according to historical records, in some way to forecast the upcoming visit of a request, and in accordance with certain rules for dealing with pre-reading, to balance load and speed up access speed. At the same time, running on the service side of the monitoring process will be the regular collection of farmers and learn the history of the nodes within the cluster load of information, through the analysis of these data and mining, the server clustering within a reasonable variety of resources to optimize and re-configuration.

Key words: data mining; load balancing; response time

0 引 言

乡镇是新农村建设的关键结点,现有乡镇信息化系统基本处于纵向运作模式,如"先锋网"与"远教网"、"金农工程"、"千村百镇工程"等在乡镇一级互相分隔,软硬件资源得不到整合利用。新农村信息一体化平台的搭建,是基于"复合二层架构"实现乡镇信息化平台的横向信息整合,通过建设诸如办公管理、农业建设项目实施管理、乡镇基本建设财务管理、乡镇社区人员管理、人口土地管理等专业系统接口,融合三层信息结构软硬件资源,提高农村综合信息管理的技术水平和服务能力,探索改善农村综合信息化集成的多种途径和运营维护模式。项目示范基地设在绩溪县,项目完成后将建成具有县、乡(镇)、村三级管理与乡村二级复合结构相结合特色的八大新农村信息化示范基地,形成以服务"三农",有效进行社区管理双管齐下的核心试验区,建立相关的技术体系和功能模块,适应经济、社会和人口、资源、环境协调发展,推进新农村现代化建设的不断发展。在项目实施过程中,发现当多个农户同时上网学习常常引发远程服务器端的负载均衡问题[1],系统响应时间比较慢,大大影响农户的学习兴趣。为解决这个问题,笔者作了深入的理论探讨和实验分析。

1 请求预测

通过对访问历史数据和农户学习习惯的观察,发现如果农户对某个内容片断进行过训练,那么与之相关或者类似的一段内容极有可能被继续请求。由此认为,内容之间在逻辑上存在某种关联,从而在一定的概率下,可以预测农户下一个要访问的文件。正是基于这种猜测,文中提出了将数据挖掘[2]应用到负载平衡系统的构想。

收稿日期:2008-08-07
基金项目:安徽省自然科学研究基金项目(KJ2007B365ZC)
作者简介:丁仁源(1976-),男,硕士,讲师,研究方向为农业信息化、人工智能;李 旸,博士,副教授,硕士生导师,研究方向为计算机网络可靠性研究、农业信息化工程与应用。

从农户的学习历史记录和访问日志中,可以分析出一些句子间的关联规则,比如有这样一个规则"$S_i \Rightarrow S_j$",那么假如农户向服务器请求文件 S_i,就应考虑文件 S_j 是否需要提前准备好,因为农户很有可能即将访问该文件。在农户访问某一个文件时,将与之相关的文件预先在另外一台服务器上读入,则当前文件访问结束后,可切换到另外一台服务器,在那台服务器上,这个文件已经被预先处理过了,因此可以获得较好的响应时间。

2　应用数据挖掘技术实现负载均衡

利用数据库可以记录每一个农户访问的详细情况,这些信息在每个访问之后都会自动上载到服务器。所有这些信息被保留在一张访问的历史(Learning History)数据库表中。

从目前的使用情况和得到的结果来看,可以统计和分析出比较简单的数据。比如,农户登录的次数,累计和分段在线时间,每个句子被训练过的次数等等。现在,需要从这个不断增长的表中进一步挖掘出这样的信息,那就是:农户训练过某个句子之后,接下来的继续训练的那个句子有几种可能,分别是什么,并且要给出这个概率有多大,如图 1 所示。其中,p_{ij} 表示从状态 S_i 到状态 S_j 的转移概率,即 $p_{ij} = P(X_{t+1} = S_j \mid X_t = S_i)$。例如,在图 1 中参与者训练过 S_1 之后,重复训练 S_1 的概率是 p_{11},继续向前训练 S_2 的概率是 p_{12}。

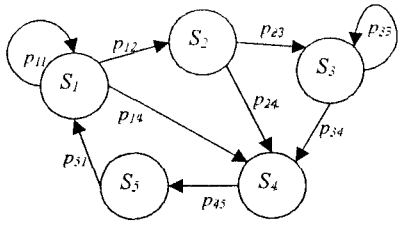

图 1　句子转移概率

将农户在某一次训练过程中的访问过的所有句子设为一个集合,用下面这个向量表示。其中,每个句子均以一个布尔值变量描述,1 代表相应的句子被训练过,0 代表没有被训练过。

$[1,1,0,0,1,1,\cdots,0,1,1]^{\mathrm{T}}$

通过分析这些向量就可以得知哪些句子会被连续访问,服务器端保存的记录非常多,考虑将这些句子按照课程和目录分组,以提高挖掘效率。

例如,训练过句子 S_i 的农户,也会训练句子 S_j,那么可用下面的关联规则来描述这个模式:

$S_i \Rightarrow S_j$[Support = 2%, Confidence = 60%]

其中,支持度为 2%,表明在所分析的农户训练历史数据中有 2% 的农户同时学习了 S_i 和 S_j,信任度为 60% 表明有 60% 的农户在训练了 S_i 之后还会紧接着训练 S_j。

如果结论与预测值相一致或趋近,那可以认为这个数据挖掘[3]的算法是可信的。理论上,可以给出最小支持度阈值和最小信任度阈值的参考值来约束上面的规则,满足这两个阈值的关联规则称为有意义的关联规则。只对达到最小支持度阈值的句子利用关联规则预存取,同时忽略掉那些未达到最小信任度阈值的规则[4]。

在应用上述算法之前,首先对 Learning History 表进行整理,为 Apriori 算法需要的数据做好准备。将该表中全部记录或者经过筛选的部分记录按照农户 ID 和训练时间进行重新排序。为了判定和收集特定参与者在某一次学习过程中所访问的全部内容,将设定一个时间间隔的门限值,低于这个门限值的学习记录被认定是属于一次学习内容的集合,否则被认定重新开始新一轮的学习。经过整理的某个农户的 Learning History 表如表 1 所示。为了说明问题,对该表进行了简化,假设总共有 6 条记录,也就是 6 次学习的历史情况。

表 1　经过预处理的访问历史表

Learning History Data Mining Table	
Learning HistoryID	SentenceID
LH_00000001	Sentence_1_1,Sentence_1_2,…
LH_00000002	Sentence_2_1,Sentence_2_2,…
……	……
LH_00000006	Sentence_6_1,Sentence_6_2,…

下面对该挖掘算法给出具体描述:

算法 1　基于 Aprori 的挖掘算法

Step 1　算法的第一遍循环,数据库中每个(数据)项均为候选 1 - 项集 C_1 中的元素。扫描一遍数据库以确定 C_1 中各元素的支持频度。

Step 2　对 C_1 中各元素通过最小支持频度进行取舍,得到 L_1。这两步进行之后会得到如表 2 所示的一组数据。

Step 3　计算 $L_{k-1} \oplus L_{k-1}$,将两个$(K-1)$ - 项集连接到一起,产生候选 C_k。

Step 4　利用前面提到的 Apriori 性质除去不可能产生频繁项集的候选项集,得到新的 C_k。

Step 5　对 C_k 中各元素通过最小支持频度进行取舍,得到 L_k。

Step 6　上面 Step 3 到 Step 5 所做的主要工作是:不断地利用频繁$(K-1)$ - 项集 L_{k-1} 产生候选 K - 项集 C_k。循环这个过程,直至无法发现新的项集而结束。

通过上面的算法,最终可得到由满足最小支持频度的候选项集所组成的频繁项集。

表 2 支持频度和项集

C_1			L_1	
项集	支持频度		项集	支持频度
Sentence_1_1	1	最小支持频度	Sentence_1_2	2
Sentence_1_2	2		Sentence_1_3	2
Sentence_1_3	2			

关联规则的生成:在从数据库中挖掘出所有的频繁项集后,就可以较为容易地获得相应的关联规则。也就是要产生满足最小支持度和最小信任度的强关联规则,可以利用项集的支持频度计算出条件概率,从而获得关联规则的信任度,如下式所示。

$$\text{Confidence}(S_i \Rightarrow S_j) = P(S_j \mid S_i) = \frac{\text{Support\_Count}(S_i \cup S_j)}{\text{Support\_Count}(S_i)}$$

这样,将会得到一组规则 $S_i \Rightarrow S_j$,其中信任度大于最小信任度阈值将被保留下来作为最终的输出。这里 S_j 可以有很多个,只选择可信度最大的那个 S_j,即对于每一个 S_i 只保留可信度最大的那个规则[5]。

3 存储策略

共享存储是新农村信息一体化平台服务器集群系统中需要解决的关键问题之一,系统对存储容量和读写速度都有较高的要求。这个系统的文件组织形式有比较突出的特点,存放的素材大多来自视频录像、科教片的片断实况。经过预处理和压缩,这些原始的媒体文件被按照句子切割开来,成为一组相对较小的完整片断。基于这个特点,在普通文件系统上设计了文件分段(File Stripping)存储和文件缓存的分布式存储策略。将媒体文件分段存储在多台服务器上,可以提高整个系统的性能,更容易达到各个存储节点间的负载均衡[6],如图 2 所示。

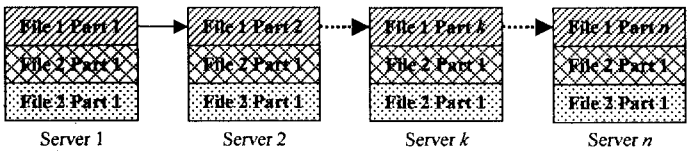

图 2 文件存储示意图

系统中某些文件的访问频繁程度很高,远远大于另外的一些。在所有的服务器上保存系统全部文件的拷贝是一个简单可行的解决方案,但显然开销较大,该方案对于一些访问概率比较小的文件来说是一种资源浪费。为此,设计了文件的 1.x 冗余存储策略。

通过上述的数据挖掘[7]结果,不难统计和分析出被频繁访问的文件。设文件每小时被访问的次数为访问频繁程度,记 File_i 的访问频繁程度为 f_i,那么如果 f_i 较大,则说明这些文件是访问"热点",可将复制到多台服务器上。根据访问的频繁程度调整文件的位置,使得每一台服务器 $\sum_{i=1}^{N} f_i$ 的都有基本相同的值。这样就不会出现一台服务器上存储的所有文件都是经常被访问的,而其他服务器上的文件访问概率都较小的情况。监控程序会定时获得每台服务器的负载,如果发现 $\sum_{i=1}^{N} f_i$ 值之间有比较悬殊的差异,将启动另外一个进程对文件分布进行调整。出于性能方面的考虑,这种调整不能实时进行。

4 实验与测试

首先描述一下实验环境,若干数据文件随机分布在多台服务器上,这些文件在逻辑上本身没有任何关联。为了更加接近于真实的应用环境,采用 3~5M 左右的经过压缩的音频文件,在每台服务器上存放 100 个。利用 Python 语言编写多线程的服务器程序。主进程负责监听请求,如果一个请求到达,那么启动一个新的子线程进行服务,主进程继续监听。服务子线程通过读取全局的关联表和缓冲区表得到关于请求文件的信息和目前的服务器状态信息,然后按照算法 1 给出响应。

客户端可采取两种方法模拟农户的真实访问。一种是通过读取文件关联的三元组,得到下一个需要访问的文件以及访问的概率。随后,客户端通过随机过程,以 p_{ij} 的概率访问文件 f_j,以 $(1-p_{ij})$ 的概率随机访问一个文件。另外一种是通过读取农户真实的访问记录,在实验中重现,从而达到模拟农户访问的目的。

4.1 关联表数据结构

三元组列表用以描述访问当前文件之后,最可能继续访问的下一个文件及其访问的转移概率。这种关联在真实环境中将通过数据挖掘来获得和产生,换句话说,数据挖掘的目的和结果就是生成和更新这个三元组列表:

$$(f_i, f_j, p_{ij})$$

其中,f_i 代表当前访问的文件,f_j 代表最有可能访问的下一个文件,p_{ij} 代表访问文件 f_i 后接着访问文件 f_j 的概率。下面的表 3 显示了一个实例。

表 3 文件关联三元组

$File_i$	$File_j$	P_{ij}
File_SVR_01_01	File_SVR_02_01	0.8
File_SVR_01_02	File_SVR_02_02	0.7
File_SVR_01_03	File_SVR_02_03	0.6
…	…	…

4.2 缓冲区池数据结构

为了更加有效地管理缓冲区,就需要知道哪些文件已经读入了缓冲区,这些文件所占的空间大小以及

是否正在被使用等基本信息。定义了如下的数据结构用来描述缓冲区的使用情况。

算法 2 缓冲区类定义

```
// START
class BufferPool:
    def_init_(self):
    self.lock = threading.Lock()# 互斥信号量
    self.pool = []
# 缓冲池
    def addToBuffer(self,file): # 读入缓冲内容
    self.lock.acquire()
# 临界区开始
    ......
bufferBlock = {'name': filename,'size': size,'bytes': bytes,'reference': 0}
# 文件名,缓冲区大小,文件内容,引用数
    ......
    self.lock.release()
# 临界区结束
    def readFromBuffer(filename):
# 取出缓冲内容
    def isInBuffer(filename):
# 判断文件是否存在于缓冲区
// END
```

每个文件读入缓冲区之后,就为之新建一个 Buffer Block,按照上面的方法来存入基本信息并附加到缓冲池上。当文件被访问时,通过遍历整个序列即可获知该文件是否已经读入缓冲区,另外还需要修改文件的引用数目,记录有多少个请求正在访问。上面的修改和维护需要进程间的互斥,为了控制并发,必须引入信号量机制,使得各个修改这个链表的操作保持同步。

4.3 实验结果分析

本地试验,在这一组试验中,客户端随机请求若干个位于本地磁盘上的文件,客户端记录从请求发出到文件接收完毕的时间间隔。试验共进行两次,第一次不采取任何策略,第二次试验是在部署了上述策略的服务器上进行的。客户端分别模拟了 10 个线程、20 个线程、30 个线程、40 个线程和 50 线程并发时的情况。图 3 中的横坐标表示并发线程的数目,纵坐标表示线程得到响应的平均时间。位于上方的曲线是第一次试验的结果,下方的曲线是第二次试验的结果。

从数据中看出,未经过预处理的访问请求,随并发度的提高,性能下降十分明显;而经过预测的访问请求,性能得到明显改善。在具有 512M 内存的服务器中,在忽略网络瓶颈的情况下,同时可支持 20 个以上的并发请求,这个数值主要有内存大小来限制。

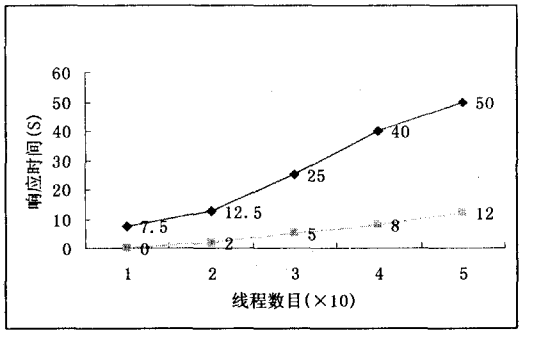

图 3　实验结果对比图

接下来的这一组试验,是在真实的网络环境中进行的,客户端按照挖掘算法[7]得到的结果进行模拟访问,并将结果与随机算法作对比试验。在这个试验中,客户端模拟了多播的机制,通过读取服务器列表,向每台服务器的相应端口发出请求。在实际应用中,这将由多播协议和多播路由器来实现。考虑到请求报文的大小相对于应答报文来说可以忽略,所以认为该替代方案不会对试验结果造成明显的影响。具体的实验步骤如算法 3 所示。

算法 3　模拟客户端访问

Step 1　客户端读取关联三元组,同时读取文件列表。

Step 2　程序随机从文件列表中选取一个文件,作为访问开始点,并向所有服务器发出请求。

Step 3　程序将读取关联列表中相应元组的 p_{ij} 值,通过随机过程,以 p_{ij} 的概率访问元组中指名的下一个文件,以$(1 - p_{ij})$的概率随机访问一个文件。

Step 4　重复上述过程,直到时间达到规定值,结束访问。

图 4 中的两条高斯曲线分别代表文中提到的策略和随机策略的响应均值和方差,横坐标代表均值,纵坐标表示概率密度。

从结果中可以看到,虚线表示的未经预处理的客户端响应时间,其均值 $\mu_1 = 2.9254$(seconds),方差 $\sigma_1 = 1.2329$。实线代表经过上述算法处理的响应时间,其均值 $\mu_2 = 2.5654$(seconds),方差 $\sigma_2 = 0.9690$。

5　结束语

通过上述实验结果和分析,可见经过预处理的服务器给出的响应时间曲线在均值和方差方面比较小。由此说明应用文中所提策略的服务器在响应速度和稳定性方面都有较大改进。

负载均衡问题是农业信息化建设中面临的主要问

题之一，在研究和分析了当前流行的多种静态和动态的均衡算法[8]之后，提出了一种基于数据挖掘思想的负载均衡方法。该方法利用对访问历史记录的数据挖掘结果，使服务器有能力预测即将到来的访问并做出预处理。文中对数据挖掘的方法和负载均衡的策略分别进行了描述和分析，并在此基础上进行测试和实验，取得了较为理想的效果。

当然，多个城市之间的数据访问和同步将会得到实施，在具体的操作过程中还需要对目前的策略进行补充和修改。

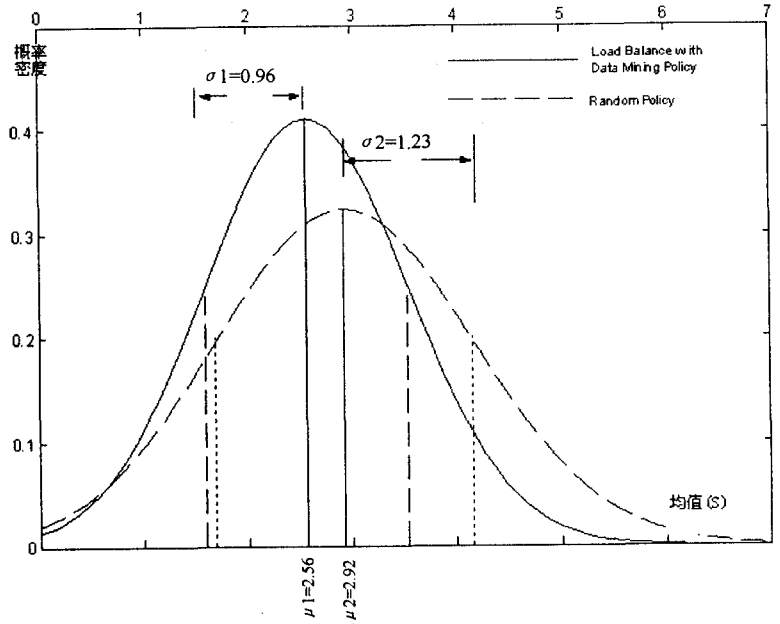

图 4 基于本地的多线程并发访问比较

参考文献：

[1] 李丙锋，祝永志，魏榕晖. 异构 Beowulf 系统负载均衡技术的研究与实现.[J] 计算机技术与发展，2008，18(7)：60－62.

[2] 陈国良. 并行算法实践[M]. 北京：高等教育出版社，2004.

[3] Group W. A high performance portable implementation of the message passing interface[J]. Journal of Parallel Computing, 1996, 22 (6): 789－828.

[4] 奎 因. 并行程序设计 C、MPI 与 OpenMP [M]. 北京：清华大学出版社，2005.

[5] 王 刚，王本年. 基于 FNN 与 GA 相融合的数据挖掘方法研究[J]. 计算机技术与发展，2008，18(2)：119－125.

[6] Ishigami H, Fukuda T, Shibata T, et al. Structure optimization of Fuzzy Neural Network by Genetic Algorithm[J]. Fuzzy Sets and Systems, 1995, 71(3): 257－264.

[7] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control[J]. IEEE Transactions on Systems, Man and Cybernetics, 1985, 15(1): 116－132.

[8] 李 旸，高正光，李启炎. 基于神经网络与遗传算法的数据挖掘体系结构[J]. 计算机工程，2004，30(6)：155－156.

(上接第 233 页)

量超过一定阈值，也能够检测出；同时，基于段落词频统计可以检测到将段落语句次序打乱重新组合和对段落进行扩充或压缩的情况。由于最后做出的结论有一定误差，还需要人工进一步判定，所以分别在两个窗口输出抄袭论文和待查论文疑似抄袭的段落，使用户不必再从整篇论文中查找、定位抄袭内容，从而方便用户进一步查看与判定。

对于中文学术论文的抄袭识别问题，相对于英文论文抄袭识别来说，由于需要额外考虑汉语的词切分、词法及语法特点，因此，难度较大。对于文中提出的算法和文中提到的其他算法都存在一定的误判，而且效率还需要进一步提高。另外，对于论点抄袭更是难以判定，一般需要借助于人工智能进行语意分析和判断。因此，对于论文抄袭问题还需要进一步研究，还不能完全替代人工判定。

致谢：文中的中文分词程序采用了中国科学院计算所软件研究室的汉语词法分析系统 ICTCLAS。

参考文献：

[1] 金 帛. 剽窃、抄袭他人的作品是一种严重的侵权行为——兼谈对剽窃、抄袭行为的认定[J]. 晋图学刊，2001 (4)：77－78.

[2] Mander U, Baker B S. Deducing similarities in Java sources from bytecode[C]//Usenix 1998 Annual Technical Conference. New Orleans: The Advanced Computing Systems Association, 1998: 179－190.

[3] 史彦军，滕弘飞，金 博. 抄袭论文识别研究与发展[J]. 大连理工大学学报，2005，45(1)：50－57.

[4] 鲍军鹏，沈钧毅，刘晓东，等. 自然语言文档复制检测研究综述[J]. 软件学报，2003，14(10)：1753－1760.

[5] 宋擒豹，沈钧毅. 数字商品非法复制和扩散的检测机制[J]. 计算机研究与发展，2001，38(1)：121－125.

[6] 金 博，史彦军，滕弘飞. 基于篇章结构相似度的复制检测算法[J]. 大连理工大学学报，2007，47(1)：125－130.

[7] 王小捷，常宝宝. 自然语言处理技术基础[M]. 北京：北京邮电大学出版社，2002.

[8] Zhang Hua ping. HHMM－based Chinese lexicalanalyzer ICTCLAS[C]// Second SIGHAN Workshop Affiliated with 41st ACL. Sapporo: [s. n.], 2003: 63－70.