

协作过滤中的多级分布式日志存取研究

陈 琳

(西安邮电学院 计算机学院, 陕西 西安 710121)

摘 要: 协作过滤作为一种有效的个性化推荐技术受到了广泛的关注. 在实际应用中除算法本身效率外, 影响算法性能发挥的一个重要的因素就是日志数据的存储与读取速度. 针对该问题, 文中提出一种多级分布式的日志数据存取方案, 采用双哈希表的内存组织与分布式的持久存储相结合的方式缓解日志 I/O 的瓶颈问题. 实验结果表明, 该方法比直接从硬盘读取数据以及串行读取数据的性能有明显提高.

关键词: 分级存储; 协作过滤; 并行计算

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2012)01-0006-04

Multi-level Distributed Log Storage in Collaborative Filtering

CHEN Lin

(College of Computer, Xi'an University of Post & Telecommunications, Xi'an 710121, China)

Abstract: Collaborative filtering as an effective technique for personalized recommendation has attracted many attention, but the current research focuses on the efficiency of the algorithm, in the real world application the storage and access speed of log data plays an important role in the performance. One multi-level distributed data storage method is proposed in which dual hash table and distributed memory storage is combined to relax the I/O bottleneck. Experimental result shows that the performance can be improved significantly.

Key words: multi-level distributed data storage; collaborative filtering; parallel computing

1 引言

互联网的发展有力促进信息传播效率, 但资源规模的急剧膨胀导致人们获取有用知识的难度却日益增大. 个性化推荐作为缓解“信息膨胀, 认知过载”的有效途径引起人们广泛关注^[1-2], 个性化推荐可分为基于内容的推荐(Content-based)^[4-5]和协作过滤(Collaborative filtering)^[6]两种方式.

基于内容的推荐对资源进行关键特征抽取, 计算资源与用户模型的相似度, 把相似度高的资源推荐给用户. 该方法要求资源内容具有良好的结构, 易于抽取, 不适合视频、音频等难以抽取的资源.

协作过滤是一种利用用户与资源之间内在联系来预测用户偏爱的方法, 又称作社会过滤(Social

Filtering). 在协作过滤中, 通常采用评分表达用户对资源的偏爱(如 5 级制评分), 评分可以是显式获取, 也可根据下载次数、收藏情况、停留时间等因素计算获得(隐式获取). 协作过滤无需对推荐对象进行处理, 适于视频、音频等不易抽取的资源.

协作过滤有基于记忆(Memory-based)和基于模型(Model-based)两种方式^[7]. 基于记忆的算法采用统计技术获得用户的邻居, 通过邻居的评分预测当前用户的评分. 基于模型的算法利用机器学习技术如贝叶斯网络、聚类、神经网络^[8]从评分中学习个性化预测模型, 利用该模型为用户预测评分.

目前协作过滤的研究集中于算法本身. 当日志达到一定规模, 日志存取效率重要性越发重要, I/O 和 CPU 的先天失配导致性能瓶颈集中在 I/O 阶

收稿日期: 2011-05-08; 修回日期: 2011-06-15

基金项目: 国家自然科学基金项目(60825202, 60803079, 60633020, 61100166); 国家高技术研究发展项目(2008AA01Z131); 国家科技支撑计划项目(2006BAK11B02, 2006BAJ07B06); 陕西省重点学科建设专项(080204, 11JK1-035)

段,而非算法本身,因此研究一种快速高效的日志存取方法是提高算法整体性能的关键。

2 两级日志存取

为缓解算法运行速度和数据读取间的速度差距,本文提出一种两级日志存取方案:第一级是与协作过滤算法接触的双哈希表,是日志数据在内存中的存储和组织方式.第二级是分布式日志持久存储,采用的是以用户 ID 和资源 ID 逻辑切分后的双重存储,是哈希表在持久层的直接映射.如图 1 所示。

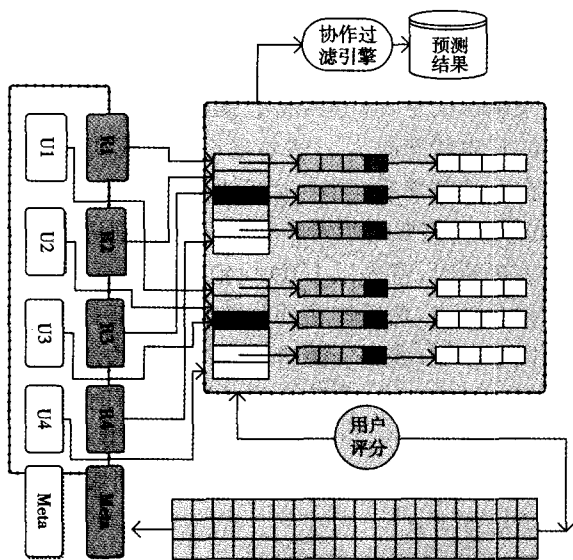


图 1 日志分级存储

前期积累的日志采用两级存储后,在系统使用过程中将不停的有新评分产生,这就需要对内存中双哈希表和持久层的评分日志进行更新.为了加快更新速度,当用户进行评分时,评分记录 $\langle \text{UID}, \text{ItemID}, \text{Rating} \rangle$ 同时送往双哈希表和评分缓冲管道.用户哈希表更新过程为:如果哈希表中已经有该用户,则找到该用户的评分链表,将评分插入在表尾,否则新建一个头结点,并把用新评分生成新链表连在头结点之后.资源哈希表的更新过程相同.发往缓冲管道数据积累到一定程度,以批处理的方式从主服务器获得存储服务器的 IP 地址,按照 IP 地址将评分发往存储服务器更新日志数据。

2.1 内存中的双哈希表式日志组织方式

虽然可以采用 SQL Server 等关系数据库的方式存储和操作数据,但持久存储设备 I/O 速度和处理器处理速度相差很大,把数据加载到内存很有必要.数据在内存中需要一种有效的组织方式,考虑到协作过滤对数据的操作主要是:① 查找用户评价过的资源;② 查找评价过某资源的用户;③ 两个用

户共同评价过的资源;④ 用户评分的平均值;⑤ 资源评分的平均值等.所以采用一种双哈希表的数据组织方式非常合适(如图 2 和图 3 所示).另外考虑内存的容量有限,所以采用对整型数据的 4 个字节进行划分,3 个高位字节存储用户 ID,最后的 1 个低位字节资源存放评分.3 个高位字节共有 24 个 bit,可以存储用户 ID 的数目有 $2^{24} = 16777216$,可以满足大多数情况下的需要.这种压缩存储的方式即可以节省一半的内存空间,又可以利用二进制位运算来加快计算速度。

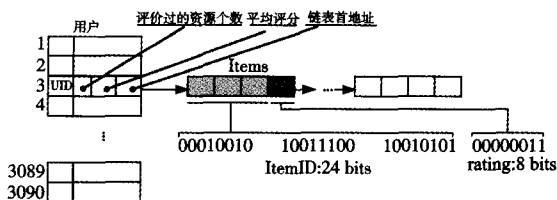


图 2 以 UserID 为索引的哈希表

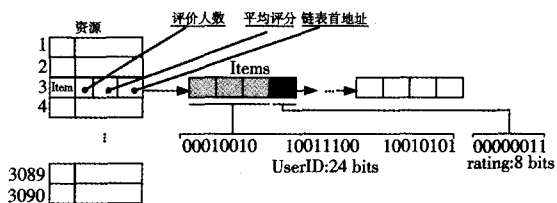


图 3 以 ItemID 为索引的哈希表

在内存中评分通过两个哈希表组织起来:一个是以 UID 为索引的哈希表(如图 2 所示),通过该哈希表可以快速找到用户评价过的资源及对这些资源的评分.哈希表的头结点包括 4 个字段:UID、评价资源的数目、平均评分和单链表的首地址,这样就可以有效地计算平均分、评价资源的数目等.另一个是以 ItemID 为索引的哈希表,可以通过资源 ID 快速查找评价过该资源的用户 ID 及其评分,并快速找到评价过该资源的人数、该资源上评分的平均值等。

2.2 分布式日志持久存储

在通常情况下,所有评分日志集中写在一起,即使采用分布式存储也是通过物理切片的形式把日志进行划分(比如 100K 一块),这种物理切分方式在后来读入哈希表时,需要逐条判断评分记录该如何插入哈希表中,速度受到很大影响.因此提出一种逻辑切分的方式将日志进行了切分,具体思想是:把每个用户评价过的资源集中在一起作为一块,有几个用户将会切分为几块.把这些日志块分布存储在集群服务器上,并在主服务器记录用户 ID 和 IP 地址的映射关系.这些日志块和哈希表通过用户 ID 形成一一对应关系,当加载日志时,给出用户 ID 便可以

快速找到 IP 地址,从存储服务器读取数据直接读入内存中哈希表中,只需用用户 ID 找到链表首节点位置即可成批载入,不必一一判断.按照上述思想把评分日志按照资源 ID 进行切分并进行相同的处理过程.可以看出日志被进行了双重存储,这虽然增加了持久层物理空间的开销,但可以大大提高存取速度.现有持久层的容量很大,完全能够满足需要.另外有一个专门的 Meta 信息存放评分矩阵的元信息,包括用户数据,资源数目,以及评分数目.

3 具体实现与验证

3.1 并行上传日志

上传日志流程如图 4 所示.

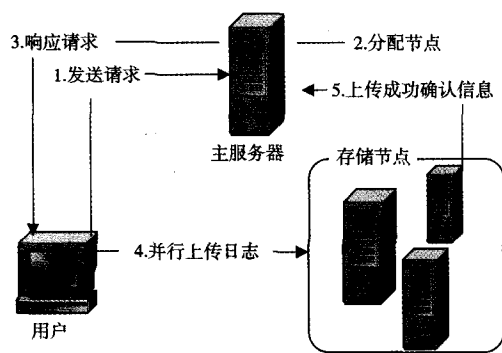


图 4 并行上传日志原理图

主服务器负责切块节点分配,存储服务器用于存放数据.按照对用户访问频率的统计,只有 20% 的用户却访问了 80% 的资源,因此不能把用户日志平均分配在存储节点上,所以主服务器首先根据存储节点负载情况,将日志数据均衡分配到各个节点,最后将日志存储信息(用户 ID,日志文件名称,对应节点 IP 等)插入数据库(状态设置为 0,表示为上传),在主服务器建立日志文件名与节点 IP 的映射表,实现从用户 ID 到节点 IP 的快速定位.把映射表发送给用户端,用户端按照映射表把日志存放在服务器上,成功过后置状态为 1,表示上传成功.客户端接收到映射表后,与服务器建立连接,开始上传日志.其主要流程如下:

算法 1 并行上传日志

Step1: 节点分配

通过 UID 和 RID 获得日志在存储节点的 IP 地址.主服务器收到日志信息后,首先根据当前 UID 和 RID 是否已经在映射表中存在,如果存在则取出存储节点的 IP,否则根据存储节点负载情况,将 UID 和 RID 日志文件均衡分配到各个节点,同时将节点 IP 插入数据库(状态设置为 0,表示为上传),

更新主服务器的文件名与节点 IP 的映射表.

Step2: 响应上传

服务器将把 IP 地址发送给客户端.

Step3: 并行上传日志

其实并非每当有日志进入是就请求服务器,而是当日志积累到一定程度或时间达到某个点时才将一批日志存入系统.所以服务端将给客户端返回一个 IP 地址列表,按照 IP 地址进行排列,为每个节点建立一个日志数据队列,将数据并行上传到对应的节点.

Step4: 上传成功确认信息

存储节点每接收到一个日志块后,向服务器发送一条确认信息,服务器将数据库中该日志块中所涉及的日志状态改为 1(表示已经上传成功).

在系统部署后,当产生新评分 $\langle \text{UID}, \text{ItemID}, \text{Rating} \rangle$ 时,需要对哈希表和持久层分别进行更新.哈希表更新时,首先以 UID 找到用户哈希表中该用户的评分链表,然后连在评分链表的尾部;以同样的方法以 ItemID 找到资源哈希表中资源的评分链表,然后把新评分连在尾部.持久层的更新时,以 UID 到主服务器查找用户所在 IP 地址,如果有即以该 IP 访问存储服务器,并把新评分上传到服务器端.更新算法如下:

算法 2 日志更新算法

一个新的评分 $\langle \text{UID}, \text{ItemID}, \text{Rating} \rangle$ 进入系统.

// 哈希表更新

Step1: 判断 UID 在 UserHash 中是否有头结点,若有,则将新评分插入表尾,否则,新建头结点,并把评分作为第一个元素插入链表;

Step2: 判断 ItemID 在 ItemHash 中是否有头结点,若有,则将新评分插入表尾,否则,新建头结点,并把评分作为第一个元素插入链表;

// 持久层的更新

Step1: 分别以 UID 和 ItemID 查找日志所存的 IP

Step2: 分别以 UID 和 ItemID 定位该用户在服务器中的存储地址

Step3: 将新评分写入日志存储空间

3.2 并行读取日志

并行读取的工作流程如图 5 所示.服务器接受到日志读取请求后,找到所有的存储服务器 IP 地址上的存放列表.服务器将日志存放列表信息以 XML 的格式返回给客户端.客户根据收到的日志存放列

表,为每个存储节点创建一个线程,将日志块并行载入哈希表中.整个过程可归结为:

算法3 行读取日志

Step1:发送读取请求.

Step2:查找日志存储相关信息,找到所有的存储服务器IP地址上的存放列表.

Step3:响应下载请求,将日志存放列表信息以XML的格式返回给客户端.

Step4:并行读取日志,根据收到的日志存放列表,为每个存储节点创建一个线程,将日志块并行载入哈希表中.

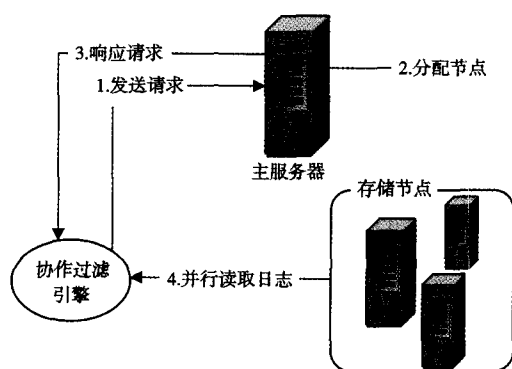


图5 并行读取日志原理图

4 结束语

针对协作过滤在存取日志时的I/O瓶颈问题,文中提出了一种分级分布式日志存储方案.在第一级的内存缓冲阶段,将日志分别以用户ID和资源ID为索引建立双重哈希表,在持久层采用分布式存储方法,把日志按照逻辑关系进行划分,分布存储在各服务器上,当把日志从服务器加载进入第一级缓冲层时不需要判断,可以直接加载.经过在10个节点组成的集群上进行试验,这种存储方式比集中式存储可以提高5-6倍左右,如果增加存储节点,读取速度也将相应提高.这为协作过滤的实效性和效率提供了保障.

参考文献:

- [1] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions [J]. IEEE transactions on knowledge and data engineering, 2005, 17(6): 734-749.
- [2] Liu D, Tsai P, Chiu P. Personalized recommendation of popular blog articles for mobile applications [J]. Information Sciences, 2011, 181(9): 1552-1572.
- [3] Zheng Y, Xie X. Learning travel recommendations from user-generated GPS traces [J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(2): 1-29.
- [4] Valle E, Avila S de, Luz Jr A D, et al. Content-based filtering for video sharing social networks [C] // In Proceedings of CoRR. Brawse-Sart, 2011.
- [5] Su J H, Yeh H H, Yu P S, et al. Music recommendation using content and context information mining [J]. Intelligent Systems, 2010, 25(1): 16-26.
- [6] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms [C] // Proceedings of the 10th international conference on World Wide Web. China; Hong kong, 2001: 285-295.
- [7] Lang K. Newsweeder: learning to filter netnews [C] // Proceedings of the Twelfth International Conference on Machine Learning. USA; California, 1995: 331-339.
- [8] Zhang T, Iyengar V S. Recommender systems using linear classifiers [J]. The Journal of Machine Learning Research, 2002(2): 334-344.

作者简介:



陈琳 女,(1977-),硕士,讲师.研究方向为数据挖掘、人工智能、个性化与兴趣感知.