Item-Based 并行协同过滤推荐算法的设计与实现

燕 存,吉根林

(南京师范大学计算机科学与技术学院,江苏 南京 210023)

[摘要] 基于协同过滤的推荐已成为推荐系统中广泛采用的推荐技术. 由于应用中用户数目和商品条目的日益增长,在计算相似度和计算预测时,单机集中式计算已不能满足推荐系统实时性和可扩展性的要求. 针对这一问题,设计并实现了 Item-Based 并行协同过滤推荐算法. 该算法采用 Hadoop 的 MapReduce 与 HDFS 架构,可分为 Map 与 Reduce 两个过程. 通过在 Map 和 Reduce 节点上的并行处理可提高算法的执行效率. 实验结果表明,该算法可明显减少推荐时间,提高推荐实时性,获得良好的可扩展性.

[关键词] 推荐系统,协同过滤,Hadoop,MapReduce

[中图分类号]TP311 [文献标志码]A [文章编号]1001-4616(2014)01-0071-05

Design and Implementation of Item-Based Parallel Collaborative Filtering Algorithm

Yan Cun, Ji Genlin

(School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China)

Abstract: Collaboration filtering has been widely used in recommender system. However, with the increase of user numbers and item numbers, the computing ability of single computer can not satisfy the requirement of real-time performance and the requirement of scalability when computing the similarity and prediction. To address the issue, this paper presents an Item-Based parallel collaborative filtering recommendation algorithm. Adopting the framework of MapReduce and HDFS in Hadoop, the parallel algorithm is divided into two procedures, which are Map and Reduce. Through the computation in parallel in respective Map and Reduce nodes, the algorithm performance is improved. The experimental results show that the proposed algorithm can decrease the recommend time and has a good real-time performance and scalability.

Key words: recommendaton system, collaborative filterinig, Hadoop, MapReduce

随着计算机网络的迅猛发展,电子商务规模不断扩大,数据库中用户数量和商品数量急剧增长. 用户想要在海量的数据中发现自己所需要的信息变得异常困难. 推荐系统[1]成为解决这一问题最好的解决方案,它根据用户的兴趣推荐满足用户需求的对象,减少查找时间,实现个性化推荐服务. 目前几乎所有电子商务网站都已应用了推荐模块,如占有市场份额最多的阿里巴巴,以及全球最大的网上书店 Amazon. com[2]等. 在各种推荐算法[3-5]中,基于协同过滤的推荐无疑是最成功的个性化推荐技术[2]. 基于协同过滤的推荐技术已被应用到很多领域,主要分为基于用户的(User-Based)推荐和基于项目的(Item-Based)推荐[6]. User-Based 协同过滤推荐基于用户的历史购买记录,找出购买记录最相近的用户作为相似用户,再从购买记录中找出目标用户未购买的商品进行推荐. 在大型电子商务网站中,商品的数量相比于与日俱增的用户数量是相对稳定的,Item-based 协同过滤将计算用户之间的相似度转化为计算项目间的相似度,项目间相似度可离线计算,这将有效地减少在线计算量,可在一定程度上解决实时推荐的问题[7].

为了改善推荐算法的实时性,文献[8]在 Hadoop[9]平台上实现 User-Based 并行协同过滤推荐算法. 文献[10]在 Hadoop 平台上实现 Item-Based 并行协同过滤推荐算法,该算法中,计算用户购买相同项目对是

收稿日期:2013-03-31.

基金项目:江苏省自然科学基金重点项目(BK2011005).

通讯联系人:吉根林,博士,教授,博士生导师,研究方向:数据挖掘技术及应用. E-mail:glji@ njnu. edu. cn

通过查找这两种项目的所有购买记录,该方法的时间复杂度为 $O(n^2m^2)$(m 代表用户数量,n 代表项目数量);同时该算法将同一用户所购买的项目发送到同一节点,计算预测时还需在每个节点上将相似性矩阵读入内存,内存消耗量大. 本文针对上述问题进行改进,提出了一种新的 Item-Based 并行协同过滤推荐算法,在 Hadoop 集群上利用 MapReduce[11] 框架和 HDFS[12] 实现 Item-Based 并行协同过滤算法,该算法计算用户购买相同项目对是将任意两两项目对作为键发送到 Reduce 节点,具有相同项目对的项目将发送到同一节点,该方法的时间复杂度为 $O(n^2m)$,缩短了计算时间;由于用户评分数据的稀疏性,将相似项目的相似性发送到同一节点,读入用户评分数据计算预测评分,计算上也降低了内存消耗.

1 Item-Based 协同过滤推荐算法

Item-Based 协同过滤是指将一个用户所购买或评级的商品匹配到相似的商品,然后将相似的商品进入推荐列表. 从计算的角度看,就是将所有用户对购买的指定商品的评价作为一个向量,每个用户对此商品的评价作为向量中对应的分量,这样通过向量值来计算物品之间的相似度,得到物品的相似物品后,根据用户历史购物记录预测当前用户对未购买物品的评价,计算得到一个按预测值排序的物品列表作为推荐.

在 Item-Based 协同过滤推荐算法中最为耗时的阶段在于计算相似度矩阵和预测用户偏好. 若用户数为 m,商品项目数为 n,在 n 个项目中找出所有项目对的时间复杂度为 $O(n^2)$,总查找空间为 m 个用户,所以计算相似度的时间复杂度为 $O(n^2m)$. 由于初始的评分矩阵是非常稀疏的,需要预测评分的项目数近似为 n,对于 m 个用户,预测的时间复杂度为 $O(nm)$. 很显然在计算相似度矩阵时,对于某一项目对计算相似度与另一对项目对计算相似度是相互独立的,因此,计算相似度矩阵是可以并行计算. 其次,预测用户偏好时,对于不同的用户计算偏好也是相互独立的,因此,预测用户偏好也可以并行计算. 但计算相似度必须在预测偏好之前完成,所以两者之间必须串行执行.

1.1 相似性度量方法

用 U 表示对项目 i 与 j 共同评分的用户集合,则项目 i 与 j 的评分相似度 $\mathrm{sim}(i,j)$ 可通过 Pearson 相关系数[13] 计算:

$$\mathrm{sim}(i,j)=\frac{\sum_{u\subset U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\subset U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\subset U}(R_{u,j}-\bar{R}_j)^2}},\tag{1}$$

式中,$R_{u,i}$,$R_{u,j}$ 分别表示用户 u 对物品 i 和 j 的评分;$\bar{R}_i$,$\bar{R}_j$ 表示物品 i,j 的平均评分.

1.2 项目的近邻选择

在近邻选择的标准中,本文通过设定一定的阈值 θ,当项目间相似度大于某一阈值 θ 时,就将此项目加入近邻集合. 这样做的意义在于,当与此项目 A 相似的项目 X 的相似度都很低,也即 A 与 X 相似度很低或不相关时,就不必将 X 加入近邻集合. 从另一个方面来说,当与 A 相似的项目很多,也不必截取前 N 个项目,而只需将相似度超过一定阈值的项全部加入候选集合,增加预测的精度. 项目 i 近邻集合 $S(i)$ 可表示为:$S(i)=\{j\mid \mathrm{sim}(i,j)>\theta, i\neq j\}$.

1.3 用户兴趣预测模型

为使预测更加准确,本文使用了一种优化的预测评分策略:将目标用户需要评分项目的平均评分作为基准值,然后寻找目标用户的目标项目的近邻集合,再利用近邻集合中的项目相似度作为权值计算目标项目的预测评分,预测用户 u 对于项目 i 的评价值的计算公式[13] 为:

$$P_{u,i}=\bar{R}_i+\frac{\sum_{j\in S(i)}\mathrm{sim}(i,j)*(R_{u,i}-\bar{R}_j)}{\sum_{j\in S(i)}\mid \mathrm{sim}(i,j)\mid},\tag{2}$$

式中,$R_{u,i}$ 表示用户 u 对物品 i 的评分,$\bar{R}_j$ 表示物品 j 的平均评分,$\mathrm{sim}(i,j)$ 表示物品 i 和 j 之间的相似度.

2 Item-Based 协同过滤推荐算法的并行化

MapReduce 是目前比较流行的并行编程模型,用户只需指定 Map 和 Reduce 函数的计算过程,系统将在大规模集群上自动并行计算. Google 和 Hadoop 都实现了 MapReduce.

利用 MapReduce 模型实现 Item-Based 协同过滤推荐算法的并行化,计算需要 3 个 Map 和 Reduce 过程. Map1 计算每个用户购买项目 ID 及其评分,Reduce1 计算项目评分均值. Map2 计算所有两两项目对,Reduce2 计算项目间的相似度. Map3 计算与某一项目 item_i 相似的所有其他项目,Reduce3 计算用户未评分项目的预测评分.

Map1 与 Reduce1 计算得到项目及其均值并保存于文件 av-file 中,由于计算过程比较简单,这里不赘述.

Map2 函数读入的用户评分数据文件 user-file 存放在 HDFS 中,每行数据代表一个用户的历史购买项目记录,以键值对<key,value>形式逐行读入 Map 节点. 每个键值对代表的是一条数据记录. key 是当前记录相对于输入数据文件起始点的偏移量,value 是当前记录中的项目 ID 及其评分. Map2(key,value)计算当前用户购买的所有两两项目对及其对应的评分. 函数的伪代码如下:

```
map2(key,value)  //value 是当前记录中的项目 ID 及其评分
{
1.  GenItemsandRatings(value,item,rating,len);  //从 value 中解析出项目及其评分,分别存入数组 item 和 rating 中,
    计算项目个数存入 len 中;
2.  For i=0 to len-1 do
3.    For j=i+1 to len do {
4.      item_a=Items[i];   rating_a=Ratings[i];
5.      item_b=Items[j];   rating_b=Ratings[j];
6.      key'=Combine(item_a,item_b);  //将 item_a 与 item_b 组成项目对;
7.      Value'=Combine(rating_a,rating_b);//将 rating_a 与 rating_b 组成项目对评分;
8.    Output<key',value'>;
9.    }
10.  }
```

Map2 输出后,系统将具有相同 key 的项目对发送到同一个 Reduce 节点. Reduce2(key,value)的功能是计算项目对 key 所对应的两个项目之间的相似度. 函数的伪代码如下:

```
reduce2(key,value)  //value 为项目对 key 对应的评分列表;
{
1.  AssignValue(value,ratings_i,ratings_j);  //将 value 中两个项目的评分列表分别赋予数组 ratings_i 和 ratings_j;
2.  ReadFileAveRat(av-file,items,aveRatings);  //av-file 文件存储每个项目 ID 及其评分均值,读取 av-file,将项目
    ID 存入数组 items,评分均值存入数组 aveRatings;
3.  for each val in value {
4.    ReadRatings(val,ratings_i[k],ratings_j[k]);  //从 val 中解析出项目对中两个项目评分,分别存入 ratings_i
    [k]和 ratings_j[k]中;
5.    k++;
6.    }
7.  value'=Pearson();  //由式(1)计算项目对 key 的相似度;
8.  key'=key;
9.  Output<key',value'>
10.  }
```

Reduce2 执行后,系统得到项目相似度矩阵,它将为每个用户未评价的项目进行预测评分.

Reduce2 的输出结果以<key,value>形式作为 Map3 的输入,key 为项目对,value 为项目对的相似度. Map3(key,value)的功能是将项目对中包含某一项目的所有项目对发送到同一个 Reduce 节点上. 函数的伪代码如下:

```
map3(key,value)  //value 为项目对的相似度;
{
1.  ReadItem(key,item1,item2);  //从 key 中解析出项目对中两个项目的 ID,分别赋予 item1 与 item2;
2.  key'=item1;
```

```
3.  value′=Combine(item2,value);  //组合 item2 和 value;
4.  Output<key′,value′>;
5.  Key′=item2;
6.  Value′=Combine(item1,value);  //组合 item1 和 value;
7.  Output<key′,value′>
}
```

Map3 执行后,系统将与项目 key 相似的项目及其之间的相似度发送到同一个 Reduce 节点,Reduce3(key,value)计算出用户未评分项目的预测评分. 函数的伪代码如下:

```
reduce3(key,value)  //value 为与项目 key 相似的项目及其之间的相似度列表;
{
1.  ReadFileAveRat(av-file,allItems,aveRating);  //av-file 文件存储每个项目 ID 及其评分均值,读取 av-file,将项目 ID 存入数组 allItems,评分均值存入数组 aveRating;
2.  ReadFileRat(user-file,userid,items,rating);  //user-file 文件存储用户 ID 及其购买项目 ID 和评分,读取 user-file,将用户 ID 存入 userid 中,项目 ID 存入数组 items 中,评分存入数组 rating;
3.  for each val in value {
4.    ReadItemSimi(val,item[k],simi[k]);  //从 val 中解析出项目 ID 和相似度,分别存入 item[k]和 simi[k]中;
5.    k++;
6.  }
7.  if 数组 items 中没有项目 key {  //若用户对项目 key 未评分,则进行评分预测;
8.    value′=Prediction();  //由式(2)计算用户 useid 对项目 key 的预测评分;
9.    key′=Combine(userid,key);  //组合 userid 和 key;
10.  }
11.  Output<key′,value′>;
12.  }
```

3　实验与结果分析

3.1　实验环境及数据集

本文实验使用的是 Movielens 站点(http://www.grouplens.org/node/73/)提供的数据集,该数据集中包含 71 567 个用户及其对 10 681 个电影的评价,其中每个用户至少对 20 部电影进行评价,评价等级为 1~5. 该数据集中的电影评分稀疏度为 0.987. 本实验由 21 台(8 核 CPU)服务器构成并行计算环境,采用 Hadoop 实现本文提出的并行协同过滤推荐算法.

3.2　实验结果

为了测试系统的可扩展性能,本文设计了两组实验. 第一组实验设定计算节点数,测试数据集大小可变. 第二组实验设定数据集大小,计算节点数可变. 由于 Item-based 协同过滤推荐中计算相似度的过程可离线计算,只有预测用户评分需实时计算,因此本实验所测试时间为预测项目评分的时间. 实验结果如图 1 和图 2 所示.

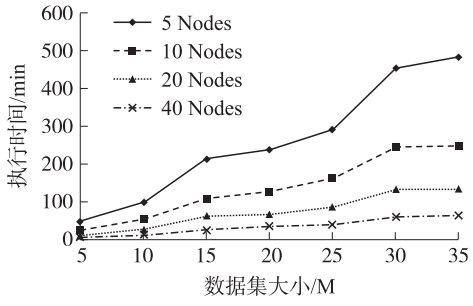

图 1　数据量与执行时间的关系

Fig. 1　Relation between datasize and running-time

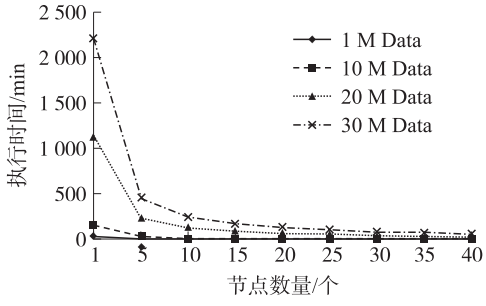

图 2　物理节点数量与执行时间的关系

Fig. 2　Relation between the number of nodes and running-time

由图 1 和图 2 可知,将 Item-based 协同过滤推荐算法并行实现后可大大减少在线预测时间,平均每个用户的预测时间是 ms 级,且在数据量不断增加的情况下通过增加计算节点数量可获得良好的可扩展性能.

3.3　算法性能比较

为了比较本文算法与文献[10]所提出算法的时间性能,本文设计了一组实验. 实验数据集为 10 MB,分别运行在 1,5,10,15,20,25,30 个节点上. 算法执行时间比较如图 3 所示,实验结果表明本文算法性能优于文献[10]算法.

由图 3 可知,本文算法的时间性能将优于文献[10],且随着处理节点的增多,两种算法的执行时间会越来越接近. 这是因为随着处理节点的增多,每个节点的计算数据相对减少,每个节点的计算时间越来越接近,总的执行时间决定于节点之间的通信时间. 在数据量和处理节点都相同的情况下,通信时间几乎相同,所以在数据量不增加的情况下,随着节点数增多,执行时间越来越接近.

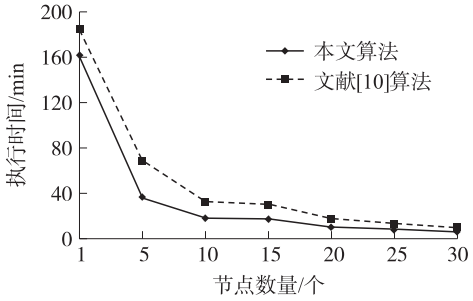

图 3　不同算法执行时间比较

Fig. 3　Running-time comparison on different algorithms

4　结语

本文研究了 Item-based 协同过滤推荐算法在 Hadoop 平台上并行实现问题,设计并实现一种新的 Item-based 并行协同过滤推荐算法,并进行算法可扩展性实验和执行时间性能比较. 实验结果表明,本文提出的并行算法在 Hadoop 集群下具有良好的可扩展性和运行效率,算法性能优于一些现有算法.

[参考文献]

[1] Schafer J B, Konstan J A, Riedl J. E-commerce recommendation applications[J]. Data Mining and Knowledge Discovery, 2001, 5(1/2): 115-153.

[2] Linden G, Smith B, York J. Amazeon. com recommendations: item-to-item collaborative filtering[J]. IEEE Internet Computing, 2003, 7(1): 76-80.

[3] O'Conner M, Herlocker J. Clustering items for collaborative filtering[C]//Proceeding of the ACM SIGIR Workshop on Recommender System. California: UC Berkeley, 1999: 121-128.

[4] 李忠俊,周启海,帅青红. 一种基于内容和协同过滤同构化整合的推荐系统模型[J]. 计算机科学,2009,36(12): 142-145.

[5] Lee J S, Jun C H, Kim S H. Mining changes in customer buying behavior for collaborative recommendations[J]. Expert System with Applications, 2005, 29(3): 700-704.

[6] Schafer J B, Frankowski D, Herlocker J, et al. Collaborative Filtering Recommender Systems[M]. Berlin Heidelberg: Springer, 2007: 291-324.

[7] Sarwer B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceeding of the 10th International Conference on World Wide Web. Hong Kong: ACM Press, 2001: 285-295.

[8] Zhao Z D, Shang M S. User-based collaborative-filtering recommendation algorithms on hadoop[C]//Third International Conference on Knowledge Discovery and Data Mining. Thailand: IEEE, 2010: 478-481.

[9] Borthakur D, Gray J, Sarma J S, et al. Apache hadoop goes realtime at Facebook[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. Athens: ACM, 2011: 1 071-1 080.

[10] Jiang J, Lu J, Zhang G, et al. Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop[C]// 2011 IEEE World Congress on Services(SERVICES). Washington: IEEE, 2011: 490-497.

[11] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[12] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. Nevada: IEEE, 2010: 1-10.

[13] Ahn H J. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem[J]. Information Sciences, 2008, 178(1): 37-51.

[责任编辑:严海琳]