文章编号：2096-1472(2016)-06-17-04

基于Hadoop和Mahout的ASUCF算法并行化研究

曹 萍
（南京审计大学，江苏 南京 211815）

摘 要：针对高效的协同过滤推荐技术处理大数据时的计算效率问题，提出了并行计算的ASUCF算法。该算法采用Hadoop平台的MapReduce并行编程模型，改善大数据环境下高效的CF算法在单机运行时的计算性能问题。最后在实验部分，结合Mahout，实现ASUCF算法的并行化，设计不同数据集上的加速比实验，验证算法并行化后在大数据环境中具有较好的计算性能。

关键词：协同过滤；计算效率；加速比；Hadoop；Mahout
中图分类号：TP391 **文献标识码**：A

Research on Parallel ASUCF Algorithm Based on Hadoop and Mahout

CAO Ping
(*Nanjing Audit University*,*Nanjing* 211815,*China*)

Abstract: Aiming to solve the CF's (Collaborative Filtering) computing efficiency problem in big data processing,the paper proposes parallel ASUCF(Average Similarity of User-Item Collaborative Filtering) algorithm.It applies the MapReduce parallel-programming model in Hadoop platform,which improves the CF's computational efficiency in big data processing on a single PC.Combined with Mahout,the parallelization of ASUCF is achieved.The paper designs speedup experiments on different data sets.The experiment results prove that the parallel algorithm brings out better computing performance in big data processing.

Keywords:collaborative filtering;computing efficiency;speedup;Hadoop;Mahout

1 引言(Introduction)

互联网时代，网络资源纷杂，信息过载，个性化推荐成为缓解用户在网络中的信息迷茫问题的重要途径[1]。在多项目、多领域的推荐中，因不依赖用户或项目内容，具有较好通用性的协同过滤算法[2]成为较为成功的推荐技术，其改进因而也受到广泛关注。然而，改进的算法通常是以牺牲计算效率换取计算准确度的提升。随着大数据时代的来临，解决计算效率的问题也迫在眉睫。由于单机模式的计算能力有限，而分布式计算具有多资源、可扩展、高效计算等优势，所以用分布式计算实现高效的CF算法，既能提高推荐准确度，又能保证计算效率。目前主要使用云计算平台Hadoop实现算法的并行化，如文献[3—8]等是通过将算法移植至Hadoop，以得到高计算性能的算法。

本文将基于平均相似度的协同过滤推荐算法(Average Similarity of User-Item Collaborative Filtering，简称ASUCF)与开源分布式平台Hadoop结合，改写Mahout中Item-based CF分布式实现，研究ASUCF算法的并行化，探索其MapReduce过程设计，并通过在不同规模的数据集上实验，比较单节点计算与多节点计算在计算效率上的差别，证明并行化后的ASUCF算法在计算效率上的优势，更能适应大数据环境。

2 Hadoop平台及ASUCF算法并行化分析(Hadoop and analysis of ASUCF in parallel)

2.1 ASUCF算法概述

文献[9]详细描述了ASUCF算法，并通过实验证明推荐准确度的提高，在此对其简单描述，为后续的并行化分析作铺垫。ASUCF为避免矩阵预处理带来的偏差，改进的算法回归到最原始的评分矩阵，从用户行为分析、项目行为分析，引入平均相似度，将计算预测评分分解成用户角度的预测和项目角度的预测两部分，综合两部分后得到最终的用户对项目的预测评分。

用户i的项目平均相似度$\mathrm{UAS}(i)$和项目c的用户平均相似度$\mathrm{IAS}(c)$计算分别如式(1)和式(2)，I_i和C_c分别表示用户i已评分项目的集合，对项目c已评分的用户集合：

$$\mathrm{UAS}(i)=\frac{\sum_{c,w\in I_i} sim(c,w)}{C_{|I_i|}^{2}} \tag{1}$$

$$\mathrm{IAS}(c)=\frac{\sum_{i,j\in C_c} sim(i,j)}{C_{|C_c|}^{2}} \tag{2}$$

基金项目：面向服务工作流的信息系统绩效审计智能算法研究(GGSS2015-06).

综合用户、项目两方面，引入UAS、IAS，则目标用户i对目标项目c的预测评分$P_{i,c}$如式(3)所示。

$$P_{i,c}=\frac{1}{2}\left(\left(\overline{R_i}+\frac{\sum(sim(i,j)\text{-}UAS(i))\times(R_{j,c}-\overline{R_j})}{\sum|sim(i,j)\text{-}UAS(i)|}\right)+\left(\overline{R_c}+\frac{\sum(sim(c,w)\text{-}IAS(c))\times(R_{i,w}-\overline{R_w})}{\sum|sim(c,w)\text{-}IAS(c)|}\right)\right) \quad (3)$$

2.2 Hadoop简介

Hadoop起源于Apache公司的Lucene和Nutch项目[10]，是谷歌云计算理论的Java语言实现。2006年，实现分布式文件系统和MapReduce的部分从Lucene和Nutch中分离出来，成为新项目Hadoop[11]。对应GFS实现的HDFS、并行计算模型MapReduce是Hadoop中最核心的部分。HDFS是Hadoop中的文件管理系统，采用主从结构，一个集群中包括一个主控服务器，即目录节点NameNode，用于索引DataNode、负责系统内文件命名空间操作、数据块和DataNode之间的映射关系等；以及多个块服务器，即数据节点DataNode，用于数据物理存储，文件通常被划分成若干个数据块，储存在不同的DataNode中[12]。MapReduce是一种可靠、高效的并行编程模型和计算框架，借助于HDFS等分布式技术，能够处理各类PB数量级的大数据[13]，其构成部分主要有一个主控服务JobTracker，若干个从服务TaskTracker，分布式文件系统HDFS，以及客户端Client[14]，它们的主要功能分别是：

(1)JobTracker：将作业划分成若干个任务，分发给多个TaskTracker，管理任务的执行以及输出反馈。

(2)TaskTracker：完成若干个Map、Reduce任务，并向JobTracker实时反馈执行情况。

(3)HDFS：数据、相关信息的保存及管理。

(4)客户端Client：Map和Reduce程序的编写，作业的提交。

MapReduce通过分解任务、合并结果的分而治之思想实现可分解、可并行处理大数据集上的并行计算。MapReduce的任务执行过程由Map和Reduce两阶段构成，每次Map和Reduce的输入和输出均是键值对<key,value>的形式，通过对相同key键值对的若干次归类整理，调用用户自定义的Map和Reduce函数，得到最终输出结果。

2.3 ASUCF算法分析

要实现算法的并行性，首先需要分析出算法中的可并行计算部分，以及并行计算部分与串行计算之间的关系。为方便表述，设：

$$PU_{i,c}=\overline{R_i}+\frac{\sum(sim(i,j)\text{-}UAS(i))\times(R_{j,c}-\overline{R_j})}{\sum|sim(i,j)\text{-}UAS(i)|} \quad (4)$$

$$PI_{i,c}=\overline{R_c}+\frac{\sum(sim(c,w)\text{-}IAS(c))\times(R_{i,w}-\overline{R_w})}{\sum|sim(c,w)\text{-}IAS(c)|} \quad (5)$$

通过对改进算法ASUCF的分析，将每次推荐的计算分解为：UAS、IAS、$PU_{i,c}$、$PI_{i,c}$，其中$PU_{i,c}$又可分解为两两用户的相似度计算和目标预测评分的计算；$PI_{i,c}$又可分解为目标区域内两两项目的相似度计算和目标预测评分的计算。UAS、IAS之间不存在计算依赖关系，两者之间是可并行的；相似度的计算和目标预测评分计算之间存在先后顺序，即目标预测评分须依赖于相似度的计算，两者之间必须是串行关系；UAS、IAS与$PU_{i,c}$、$PI_{i,c}$存在顺序性，两两之间分别是串行计算；而$PU_{i,c}$和$PI_{i,c}$之间无依赖关系，可并行计算；预测评分计算之间也是可并行的。上述计算过程关系如图1所示。需要说明的是：UAS和IAS虽然实质上也是相似度的计算，但是由于计算空间不同，所以不与$PU_{i,c}$和$PI_{i,c}$中的相似度计算部分混合，而是作为单独的过程进行计算。

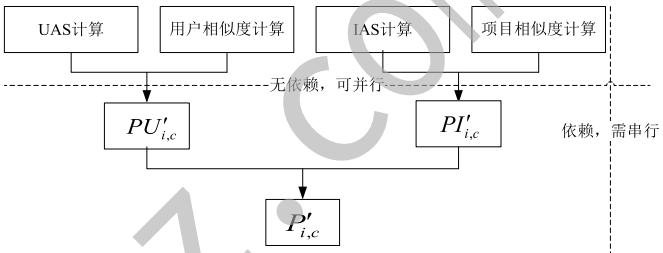

图1 算法过程分析

Fig.1 Analysis of algorithm process

3 ASUCF算法并行化计算的过程及实现(Process and implementation of parallel ASUCF)

3.1 UAS和IAS的计算

UAS的计算实际是通过对当前用户已评分项目相似度的综合衡量，得到当前用户的兴趣跨度。变换评分矩阵输入成键值对形式，过程如图2所示，共包含三个MapReduce过程，每个过程都可并行运行。后续章节中的offset代表每条记录在评分表中的偏移量。

输入：评分矩阵，当前用户id。

输出：当前用户的UAS值。

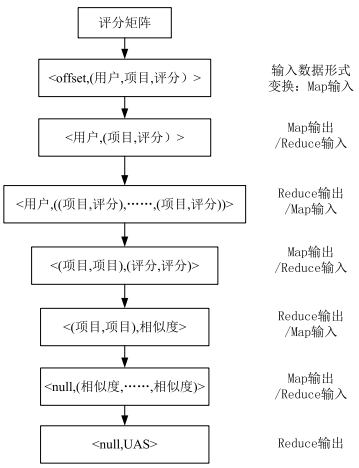

图2 UAS计算的MapReduce过程设计

Fig.2 UAS computing on MapReduce

IAS的计算实际是通过对已给出当前项目评分的用户相似度的综合衡量，得到当前项目的适用用户分布度。变换评分矩阵输入成键值对形式，过程如图3所示，共包含三个MapReduce过程，每个过程都可并行运行。

输入：评分矩阵，当前项目id。

输出：当前项目的IAS值。

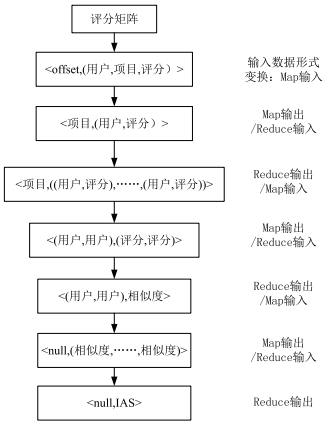

图3 IAS计算的MapReduce过程设计

Fig.3 IAS computing on MapReduce

用户相似度的并行计算过程参照文献[15]，同理得出项目相似度的并行计算过程，在此不再赘述。

3.2 预测评分及推荐的计算

综合3.1内容及用户相似度、项目相似度并行化过程设计，分析如下：

步骤一：将输入<offset,(用户,项目,评分)>经过MapReduce输出为<用户,(项目,评分)>，生成用户向量矩阵user-vector matrix；将用户向量矩阵转置为<项目,(用户,评分)>，生成项目向量矩阵item-vector matrix。

步骤二：用<(用户,用户),相似度>构成用户相似度矩阵user-similarity matrix；用<(项目,项目),相似度>构成项目相似度矩阵item-similarity matrix。

步骤一和步骤二在文献[15]中已具体表述。

步骤三：矩阵相乘，公式计算。

(1)项目向量矩阵×用户相似度矩阵，根据式(4)计算$PU_{i,c}$，如图4所示。

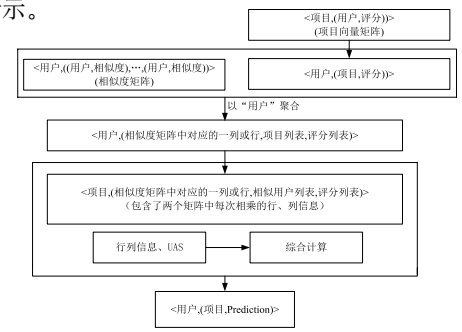

图4 $PU_{i,c}$计算的MapReduce过程设计

Fig.4 $PU_{i,c}$computing on MapReduce

(2)用户向量矩阵×项目相似度矩阵，根据式(5)计算$PI_{i,c}$，如图5所示。

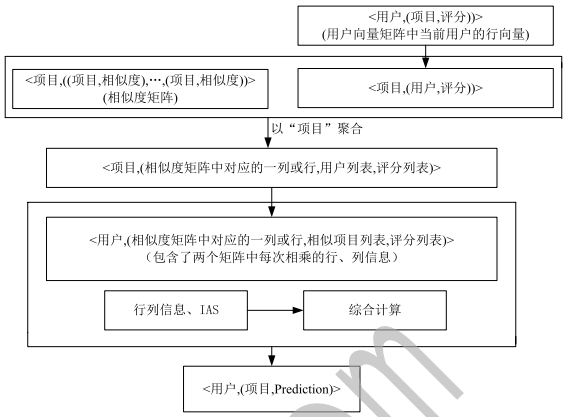

图5 $PI_{i,c}$计算的MapReduce过程设计

Fig.5 $PI_{i,c}$computing on MapReduce

关键4：根据(用户,项目)进行聚合，$P_{i,c}$、推荐结果计算如图6所示。

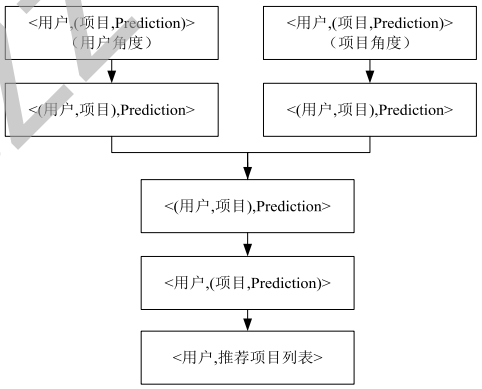

图6 计算$P_{i,c}$及推荐结果的MapReduce过程设计

Fig.6 $P_{i,c}$ and recommended results computing on MapReduce

当目标用户需要推荐时，在Map阶段输入用户对项目的预测评分，在Reduce阶段根据预测分值排序，返回TOP-N推荐集。至此，推荐完成。

在所有阶段的MapReduce过程设计没有改变算法的数学计算关系，所以对算法的计算结果没有影响，在Hadoop平台上运行与非并行模式下运行的推荐结果是一样的，但是，并行模式Hadoop下的算法，有高效的大数据集计算能力，可扩展性较高。

3.3 基于Mahout的ASUCF并行化实现

Mahout中的RecommenderJob实现了item-based CF全推荐过程的MapReduce编程，本文在此基础，改写RecommenderJob，实现ASUCF的并行化。结合上一章的分析，算法主要步骤如下[16]：

(1)改写PreparePreferenceMatrixJob，将USER_VECTORS重命名为USER_RATING_MATRIX，原Item向量RATING_MATRIX重命名为Item_RATING_MATRIX。

(2)以RowSimilarityJob的工作过程为模板，增加UserSimilarityJob，将输入改成USER_RATING_MATRIX，计算出用户的相似度。

(3)以AggregateAndRecommend的工作过程为模板，增加asucfaggregateAndRecommend，改写AggregateAndRecommend中预测评分计算：

PU(i,c)=sum(all n from N:(usersimilarity(i,n)−uas(i)*(Item_RATING_MATRIX(i,n)−Average(i))/sum(all n from N:abs(usersimilarity(i,n)−uas(i))

PI(i,c)=sum(all n from mostsimilaritytoitemc:(item similarity(c,n)−ias(c)*(USER_RATING_MATRIX(i,n)−Average(c))/sum(all n from mostsimilaritytoitemc of USER_RATING_MATRIX(i):abs(itemsimilarity(i,n)−ias(i))

P(i,c)=(PU(i,c)+PI(i,c))/2

其中，PI部分的相似度计算域不同于item−based的全局搜索，为用户i已评分项目中与项目c相似的项目。需要指出的是，mahout中实现的CF算法，并没有利用平均评分来惩罚用户评分标准的差异，故在ASUCF并行计算中除引入UAS、IAS外还需加入平均评分。

3.4 实验设计与分析

实验的Hadoop平台使用六台内存2G，CPU主频3.40GHZ的PC机，搭建完全分布式环境，1台做namenode和jobtracker，另5台做datanode和tasktracker，hadoop版本为1.0.0，ubuntu版本为12.10，jdk版本为1.6，编程工具为eclipse 3.7。实验选取明尼苏达大学提供的MovieLens数据集，选取不同规模的数据集，模拟大数据环境，包括：

(1)10万条评分记录，其中用户数为943，电影数为1682。

(2)100万条评分记录，其中用户数为6040，电影数为3900。

(3)1000万条评分记录，其中用户数为71567，电影数为10681。

实验选取算法单机执行时间T1与集群执行时间Tn的比值作为直观的评估，即常用的加速比speedup。具体实验设计为：为数据集中的每个用户推荐十个项目，启动集群中的所有节点，测试随着数据集规模的增大，加速比的变化；启动集群中的部分节点，通过增加节点，观察同一个数据集上加速比的变化。实验结果如图7所示。

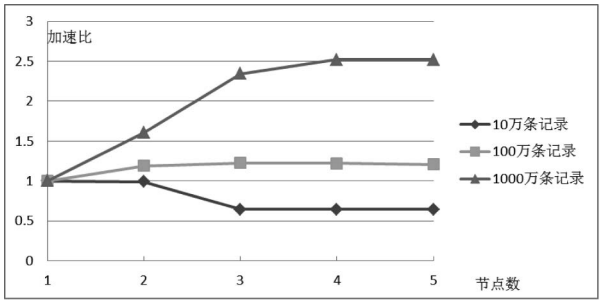

图7 加速比实验结果

Fig.7 Experiments on speedup

下面从两方面分析图7的结果：

(1)集群节点数固定，数据集规模变化

对于每个节点数情况，随着数据集规模的增大，加速比都呈现递增的趋势。当节点数目较少时，其差别不大；当节点数目较大时，差别越来越明显。

(2)数据集规模固定，集群节点数变化

对于不同的数据集，随着节点数量的增加，加速比呈现不同的变化趋势。当数据集规模较小时，加速比甚至呈现降低趋势；当数据集规模较大时，加速比呈现递增趋势；当节点数增加到一定数量时，加速比趋于稳定。

综合两方面，可以看出：只有在数据规模足够大时，集群执行的计算效率才比单机执行高，并且随着数据集的增加，集群执行的优势更突出，但当节点增加到一定数量时，计算效率也趋于稳定，这是由于节点之间存在通信、磁盘读写等开销，节点数增加，Map/Reduce所需要的系统开销也会随之增加。

4 结论(Conclusion)

本文介绍了ASUCF算法，Hadoop云平台概况，为实现高效的推荐算法，重点研究ASUCF的可并行性，分析了其在MapReduce并行编程上的过程设计，并结合Mahout中的Item−based CF分布式算法，在开源云计算平台Hadoop上实现。通过设计不同的Movielens数据集的实验，变化集群节点数目和数据集规模大小，对加速比进行评估，证明ASUCF并行算法在处理大数据时，具有较高的计算效率，解决了ASUCF算法在准确度提高的同时带来的计算效率降低的问题，实现较高准确度、较高计算效率的推荐。但是也存在不足，一方面由于实验条件的限制，搭建的集群规模有限；另一方面，是对Hadoop平台的直接应用，下一步可以结合Hadoop中任务调度等方面的性能优化，进一步提高计算能力，适应不断壮大的大数据。

参考文献(References)

[1] 李树青.个性化信息检索技术综述[J].情报理论与实践,

2009,32(5):107−113.

[2] Liu Z B,et al.A Hybrid Collaborative Filtering Recommendation Mechanism for P2P Networks[J].Future Generation Computer Systems,2010,26(8):1409−1417.

[3] 肖强,等.Hadoop环境下的分布式协同过滤算法设计与实现[J].现代图书情报技术,2013(1):83−89.

[4] 程苗,陈华平.基于Hadoop的Web日志挖掘[J].计算机工程,2011,37(11):37−39.

[5] 张明辉.基于Hadoop的数据挖掘算法的分析与研究[D].昆明:昆明理工大学,2012.

[6] 李改,等.基于大数据集的协同过滤算法的并行化研究[J].计算机工程与设计,2012,33(6):2437−2441.

[7] 周源.基于云计算的推荐算法研究[D].成都:电子科技大学,2012.

[8] 金龑.协同过滤算法及其并行化研究[D].南京:南京大学,2012.

[9] 叶锡君,曹萍.ASUCF:基于平均相似度的协同过滤推荐算法[J].计算机工程与设计,2014,35(12):4217−4222.

[10] 陆嘉恒.Hadoop实战[M].北京:机械工业出版社,2011.

[11] Chuck Lam,James Warren.Hadoop in Action.Manning Publications,2009.

[12] 刘琨,董龙江.云数据存储与管理[J].计算机系统应用,2011,20(6):232−237.

[13] 陈全,邓倩妮.云计算及其关键技术[J].计算机应用,2009,29(9):2562−2567.

[14] Tom White.周敏奇,等,译.Hadoop:权威指南[M].北京:清华大学出版社,2011.

[15] 曹萍.基于Hadoop的协同过滤推荐并行化研究[J].计算机时代,2016(5):30−33.

[16] Sean Owen,et al.Mahout in Action[M].Manning Publications Co.,2012.

作者简介：

曹　萍(1989−)，女，硕士，助理工程师.研究领域：知识工程.

(上接第33页)

了图中画出的几个接口及其实现类外，还有一个异常处理类ServiceException，以及它的两个子类：服务已存在异常类ServiceAlreadyExistException和服务不存在异常类ServiceNotExistException来捕获在服务以及服务管理中抛出的相关异常[8,9]。

3　结论(Conclusion)

本文对综合网络管理系统各个数据层的相关接口进行了设计，使得通过统一的网络管理模块实现不同网络设备间的综合管理成为可能，为综合网管系统的进一步设计实现打下了基础。

参考文献(References)

[1] A Gupta,S Kar.The Common Object Request Broker Architecture(CORBA)and its Notification Service[J].Iete Technical Review,2015,19(1−2):31−45.

[2] J Zhai.Development and Research of Workflow Management System Based on Mobile−Agent and CORBA[J].International Journal of Hybrid Information Technology,2014,7(5):305−316.

[3] Y Yin,et al.Design and implementation of a uniform service adapter for MG[J].Endocrine−related cancer,2014,21(3):261−277.

[4] 杨志敏.电力通信网运行综合监视及关键技术[J].信息通信,2013(10):192−194.

[5] 朱彦军,王斌君,张炜.应用CORBA的光网络管理系统[J].信息网络安全,2013(12):87−89.

[6] 俞祺锟.基于CORBA的网管告警接口设计及实现[D].上海:华东理工大学,2013.

[7] 代霞,黄劲松.基于CORBA综合网络配置管理的设计与实现[J].计算机技术与发展,2008,18(2):91−93.

[8] 张斌,郭军.软件工程及应用[M].沈阳:东北大学出版社,2007.

[9] 闻晶,陈兴渝.CORBA和XML在网络资源管理系统接口中的应用[D].北京:北京邮电大学,2008.

作者简介：

陈　欣(1984−)，女，硕士，工程师.研究领域：系统分析与集成，网络管理.