DOI:10.16208/j.issn1000-7024.2014.01.044

基于 Hadoop 的电子商务推荐系统的设计与实现

李文海[1,2]，许舒人[1]

（1. 中国科学院软件研究所 软件工程技术研究开发中心，北京 100190；
2. 中国科学院研究生院，北京 100190）

摘 要：为了解决大数据应用背景下大型电子商务系统所面临的信息过载问题，研究了基于 Hadoop 构建分布式电子商务推荐系统的方案。采用基于 MapReduce 模型实现的算法具有较高的伸缩性和性能，能高效地进行离线数据分析。为了克服单一推荐技术的不足，设计了融合多种互补性推荐技术的混合推荐模型。实验结果表明，基于 Hadoop 平台实现的推荐系统具有较好的伸缩性和性能。

关键词：分布式推荐系统；混合推荐；Hadoop；关联规则挖掘；协同过滤

中图法分类号：TP301 **文献标识号**：A **文章编号**：1000-7024（2014）01-0130-07

Design and implementation of recommendation system for E-commerce on Hadoop

LI Wen-hai[1,2], XU Shu-ren[1]

(1. Research and Development Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China; 2. Graduate University, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: To solve the information overload problem of large scale E-commerce systems in the big data era, a solution based on Hadoop is proposed, aiming at building a distributed recommendation system. Data analysis algorithms based on MapReduce programming model have high scalability and good performance. To overcome the limit of single recommendation technology, a hybrid model is adopted, which combines several complementary methods. Empirical studies show that the recommendation system on Hadoop has good scalability and efficiency.

Key words: distributed recommendation system; hybrid; Hadoop; association mining; collaborative filtering

0 引 言

随着用户规模的扩大和业务的发展，大型电子商务系统中的用户数和产品种类越来越多，数据规模正爆炸式地增长。然而，能够有效利用的展示区域是有限的，大量的信息资源由于成为长尾被淹没而得不到展示，信息过载问题[1]日益突出。

推荐系统[2]是一种重要的信息过滤机制，它能有效地解决信息过载问题。通过挖掘用户和信息资源之间的联系规律，它能帮助用户从网络中的大量信息里找到甚至发现他们需要的以及可能会感兴趣的信息资源。它以主动服务的方式，通过利用挖掘和分析技术，实现自动化、智能化和个性化的信息过滤服务，帮助用户发现所需。因此，它被广泛地应用于电子商务和社交网络等典型现代互联网领域。

电子商务推荐系统[3]通过模拟智能销售导购的角色，能够为电子商务网站的用户提供产品信息和相关的建议以帮助用户进行购买决策，完成购物过程。它能促进一对一的销售，可以为电子商务系统提供更加智能化和个性化的购物体验，最终提升用户满意度。电子商务的发展趋势是针对不同的用户提供个性化的购物体验，通过准确地预测用户的偏好、口味和潜在的需求，以此来提高网站的营业额，而这个目标正是通过推荐系统来实现的。

对于电子商务系统来说，推荐系统的作用主要体现在以下几方面[4]：提升购物体验，提高用户的忠诚度；增强交叉销售能力，提高整体交易量；挖掘潜在的客户资源，

收稿日期：2013-05-03；**修订日期**：2013-07-05

基金项目：国家 973 重点基础研究发展计划基金项目（2009CB320704）；国家科技支撑计划基金项目（2012BAH05F02、2012BAH09F01）

作者简介：李文海（1988-），男，江西东乡人，硕士，CCF 会员，研究方向为网络分布计算、推荐系统、数据挖掘等；许舒人（1961-），男，福建长泰人，副研究员，CCF 高级会员，研究方向为软件工程研究和大型应用系统规划、设计。E-mail: lwheagle@126.com

利用推荐激发用户的购买欲望，将用户由普通的浏览者转变为消费者。因此，可以认为电子商务推荐系统是一种集成了诸多不同种类的发现新事物的方法、技术和策略的信息过滤机制，其核心任务是通过发掘用户和商品之间的联系来向用户提供个性化的产品推荐服务，最终实现用户和系统所有者的双赢。

现代互联网应用背景下的大型电子商务推荐系统正面临着以下几个方面的挑战[5,6]：

（1）采用集中式架构的推荐系统数据处理能力有限。单机推荐算法存在数据处理规模的限制和处理效率的问题。

（2）大型电子商务系统中除了用户和产品数目庞大之外，用户和产品的自然属性也多，要对高维度的用户和产品建立准确有效的模型十分困难。

（3）商业应用需求通常复杂多变，同时对于不同的应用场景用户的关注点也不同，基于固定的模型和参数的推荐系统通常缺乏灵活性。

（4）推荐模型通常和数据特性以及应用场景有很强的相关性，这决定了推荐系统必须要综合多种互补性强的推荐技术。而现有的方案中算法和模型比较单一，难以满足主流用户的主流需求的同时兼顾用户的个性化需求。

基于 Hadoop 平台和混合推荐策略，设计了一种面向大型电子商务系统的混合推荐系统，运用基于 MapReduce 框架的算法的伸缩性和分布式并行计算能力，使系统能对大规模的数据进行高效的分析；通过结合不同的推荐引擎来解决冷启动和数据稀疏引起的推荐效果不佳的问题。系统主要解决如下 3 个方面的问题：

（1）伸缩性。现代互联网应用中数据规模已经取代了业务逻辑的复杂性而成为主要矛盾，伸缩性是现代互联网应用的关键需求，针对持续增长的数据实现伸缩对任何数据分析系统都十分重要。采用基于 MapReduce 框架的算法提高数据分析的能力，通过 Hadoop 平台实现横向扩展，能够针对不同规模的数据实现理想伸缩。

（2）灵活性。由于商业应用需求复杂多变，允许用户通过配置模型参数，以参数化运行的方式，实现不同的推荐模型以适合不同的应用场景。

（3）多样性。非个性化的推荐能够很好地反映大众的流行趋势，而个性化的推荐能够更好地满足用户的个人兴趣和偏好，将两者进行结合能充分满足不同群体的多样化的口味。此外，采用混合推荐策略也能提高推荐结果的多样性。

1　相关技术分析

1.1　主要推荐技术的比较

常用的推荐技术有协同过滤技术、关联规则挖掘和基于知识经验的方法等，其优缺点综合比较见表 1。因此，采用可以灵活配置管理的混合推荐模型综合多种不同的推荐技术以弥补单一推荐技术的不足，可以满足灵活性和多样性的推荐需求，在满足非个性化推荐需求的同时兼顾个性化推荐需求。

表 1　主要推荐技术的比较

	优点	缺点
协同过滤技术	发现新异兴趣、不依赖于领域知识，随着时间的推移和数据量的积累效果越来越好，推荐过程的个性化、自动化程度高，能够处理复杂的非结构化对象	存在伸缩性、稀疏性和冷启动等典型的问题，推荐质量依赖于历史数据集
关联规则挖掘	能发现新异兴趣点，不依赖于领域知识	规则抽取难且耗时，个性化程度低
基于知识经验的方法	能考虑非产品属性，能体现用户需求，可弥补用户知识经验的不足	知识经验难以获得，推荐是静态的

1.2　分布式计算与存储技术

Google 发布的关于分布式基础设施的论文对业界产生了巨大的影响，其中的 MapReduce 和 GFS 等思想为分布式计算与存储提供了关键参考，Hadoop 是其开源实现[7]。

Hadoop 是一个针对大规模数据处理与分析的用于构建分布式系统的基础框架，它方便易用，用户可以在不必充分关注分布式底层细节的情况下，方便地开发分布式应用，充分利用集群进行协同计算与协同存储，实现横向扩展的目标[8]。

Hadoop 通过采用数据分布式存储、迁移代码而非迁移数据的机制，在处理大规模数据时避免了耗时的数据传输问题；利用数据适度冗余机制，允许系统从节点失效中恢复。基于 Hadoop 平台开发分布式应用，用户不用关心如何分割数据、任务如何调度和分配以及如何管理集群的拓扑结构和节点之间的通信这些和分布式系统开发相关的复杂事务，而只需要专注于应用逻辑的开发。

Hadoop 平台具有高可伸缩、低成本、高可靠、方便易用等特点[9]，其核心是 HDFS 分布式文件系统和 MapReduce 框架。前者使得在成本可控的情况下处理海量数据成为可能；后者则是一种采用分治策略、专为大规模分布式并行数据处理设计的简化编程模型，它借鉴了函数式编程的思想，将针对大规模数据的处理任务统一地抽象为 Map（映射）和 Reduce（规约）两种操作。研究表明，大多数针对大规模数据的分布式处理任务都可以通过该模型进行表达，它允许编程人员以直接调用相关编程接口的方式方便地将自己的程序运行于分布式集群系统上。这样，通过在应用程序和平台之间进行职责的重新划分，让平台框架来处理诸如底层存储、节点通信等复杂的分布式处理细节，应用程序可以专注于业务逻辑以面向更加复杂的业务需求。

由于 Hadoop 集群可以按需横向动态扩展，利用 Hadoop 平台可以突破数据规模给推荐系统带来的大数据分析

的瓶颈，满足高性能、高伸缩性计算的需求。

2 混合推荐系统的设计

本节将给出支持混合推荐模型的推荐流程和系统架构的详细设计。

2.1 推荐的流程

系统采用如图 1 所示的多阶段推荐流程。首先经过数据的抽取、转换、清洗、装载等操作，从异构、多源、含噪声的原始业务数据中提取用户信息、产品信息和偏好信息，建立用户模型和产品模型。然后利用多种不同的算法与策略构建多个相互独立的推荐引擎，通过不同的引擎产生独立的推荐结果集，再将这些推荐结果以一定的策略进行混合从而形成初始推荐集。

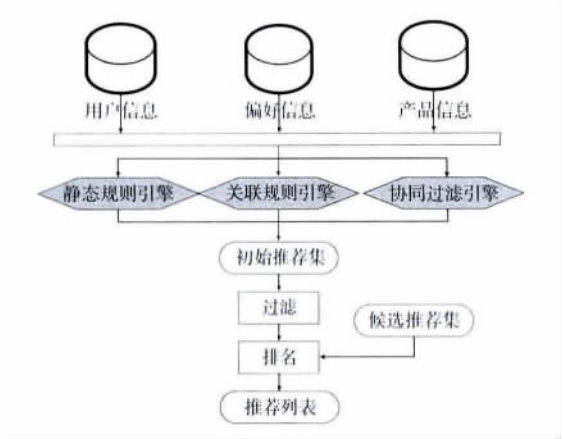

图 1 推荐系统工作流程

基于商业业务需求，有时希望将若干指定的产品推荐给用户，如通常需要在首页优先展示新加入的产品，这时需要构建一个包含此类产品的候选推荐集，在过滤阶段之后同经过筛选的初始推荐集进行合并。过滤阶段主要过滤掉以下的几类产品：

（1）用户已经购买过的产品。因为推荐系统的主要目的是帮助用户发现新产品，给用户推荐已经购买过的产品没有多大意义。然而，这也不是绝对的：对于一次消费品来说必须要被过滤，而对于可重复消费品来说可以选择不过滤。

（2）质量较差的产品。推荐系统的终极目标是提高用户体验和提升服务品质，将来自评价非常差的供应商的产品以及品质不够优良的产品推荐给用户有损用户对推荐系统的信任。

（3）具有显著的季节性特征的产品。将当前季节未供应的或者当前季节中用户不需要的产品推荐给用户也不合理。

经过过滤后的推荐结果通常可以直接展示给用户，但如果综合考虑新颖性、多样性、时间等因素对它们进行排名，则可以更好地提升用户满意度。应用协同过滤引擎，推荐结果天然地具有了排名的依据，可直接根据用户推荐向量中各个分量的取值大小进行排名。而利用静态规则引擎和关联规则引擎推荐的结果需要依据产品的其它属性。依据产品的 popularity（流行度）属性，即选取 popularity 的倒数值来为产品进行降序排列。通过将热门的畅销品降低权重，可以解决著名的哈利波特问题。经过排名阶段后的推荐结果列表便可以用富有表现力的形式展示给用户。

2.2 推荐系统的架构

基于 Hadoop 的混合电子商务推荐系统采用分层架构来实现：各层承担不同的职责，各层之间通过接口进行交互，各层之内采用模块化思想设计。这样保证了系统架构具有高内聚、低耦合和易扩展的特点。系统的分层架构如图 2 所示，自下而上依次包括以下五层：

图 2 推荐系统架构

数据访问层：数据是电子商务系统中的宝贵资产，同时也是算法和模型的处理对象。整合异构、多源、多类型的原始业务数据，并从中抽取用户和产品特征，能有效地提高推荐的准确性。同时，该层还需要负责对来自 Hadoop 集群的大规模数据分析后得到的结果进行解析。因此，该层实现了业务计算环境和集群计算环境之间的数据迁移。

数据模型层：数据模型层包括用户模型和产品模型，其中用户模型由用户基础资料、用户行为和用户信用等内容构成；产品模型包括产品元数据和反馈评价信息等。通过数据访问层对原始数据进行抽取和解析，可以获得用户属性集和产品属性集，进而生成用户特征向量和产品特征向量。

算法策略层：通用的、经典的算法通常都比较抽象，较少考虑实际的系统需求和业务需求，例如它对鲁棒性和基于应用特性或数据特性的性能调优等方面的关注较少，因此它们和具体的产品形态之间的差距较大。策略主要体现业务需求，它充当驱动器的角色，能够把业务规则封装起来提供给上层。该层包括常见的数据挖掘算法和机器学

习算法，包括相似度计算、关联规则挖掘、聚类和协同过滤等算法。算法策略层处于数据模型和推荐引擎层之间，以独立层的形式存在，这样允许整合更多数据分析算法。

推荐引擎层：该层是整个系统的核心，它将算法和策略包装成功能独立的推荐引擎，通过调用算法策略层的算法对数据模型完成相似度计算、偏好预测和关联分析等计算任务。由于电子商务系统中的推荐需求通常是多样化的，每一种推荐需求通常可以使用一种引擎来实现。设计独立的推荐引擎层允许整合更多的推荐引擎，从而实现功能上的扩展。

该层基于协同过滤技术来实现个性化推荐引擎，通过进行长尾挖掘来满足个性化推荐的需求。在非个性化推荐引擎方面，实现了静态规则推荐引擎和关联规则推荐引擎。

静态规则推荐引擎是以静态的逻辑规则的形式向用户推荐产品。与其它的引擎不同，它的规则不需要经过复杂的推荐算法计算过程，而是直接以静态的映射规则的形式来实现的。静态规则主要有 4 个方面的来源：专家经验、领域知识、商业应用需求和其它途径获得的知识与结论。这里的专家可以是一些有经验的领域资深专家，也可以是社区中的意见领袖等。由于推荐技术最根本还是一种信息过滤的机制，如果能够建立比较理想的推荐模型，同时数据质量较高且拥有大量的数据的话，推荐的效果也会比较理想，然而要获取高质量的大量数据和实现好的推荐模型都非常难，在大多数情况下，这些静态推荐规则可以在很多实际应用场合中为其它推荐技术提供有益的补充。

购物篮分析是电子商务领域中最经典的数据挖掘应用场景，关联规则推荐引擎集中关注针对交易数据库的购物篮分析。通过关联规则挖掘可以得知哪些产品经常被一同购买，从而导出基于关联规则的推荐列表。

上述 3 种推荐引擎各具特色。其中协同过滤推荐引擎实现了个性化推荐，它充分考虑用户的个人偏好。但是用户的个性化需求通常不稳定，可能会随着时间推移发生变化，仅依赖个性化引擎通常并不能得到非常令人满意的效果。此外，用户还具有关注哪些产品是热门产品、哪些产品在整体消费趋势中经常一同被购买等非个性化的需求。非个性化推荐反映群体的总体消费趋势，代表大众的喜好。结合个性化需求和非个性化需求可以更全面地满足用户的需求，实现热门资源和长尾资源之间的平衡。此外，关联规则推荐关注的是产品之间的相关性，但是只能提供非个性化的推荐；而协同过滤推荐能够实现个性化的推荐，其自动化、智能化和个性化的程度都很高，但是其也存在伸缩性、稀疏性和冷启动等问题。因此，采用具有互补意义的 3 种不同推荐引擎可以充分满足用户的各种需求，同时也能克服单个引擎的不足。最后通过混合推荐模型将 3 个不同推荐引擎的推荐结果进行组合。

应用层：位于最顶层，是用户与系统交互的接口，包含常见的配置、管理、交互和展示功能。其允许配置混合模型的参数和混合方式，保证了推荐系统的灵活性。

综上所述，设计思想可以总结为如下四点：第一，采用 Terminal/Cluster 模式的架构来满足现代互联网应用的面向海量数据分析的需要；第二，采用多阶段的推荐流程能支持混合推荐，并容易实现推荐解释的功能；第三，采用纵向分层、横向模块化、基础通用功能组件化的推荐系统架构，容易集成更多的算法、策略和引擎从而实现功能上的扩展；第四，通过应用层的配置和管理能够灵活地定义引擎方案。第一点可以实现伸缩性的目标，第二点和第三点对多样性的目标提供了支持，第四点满足了灵活性的需求。这样无论是流程还是架构都能灵活地满足大型电子商务系统多样化的推荐需求。

3　基于 Hadoop 的混合推荐系统的实现

基于 Hadoop 的混合电子商务推荐系统采用适合现代互联网应用的 T/C（Terminal/Cluster）形式架构来实现，其架构图如图 3 所示。系统包括两部分：WebTerminal 终端和 Hadoop 集群。前者是瘦客户端，只拥有少量的业务逻辑，它实现了推荐系统的配置、管理、交互和展示功能，同时能将业务系统中产生的大规模数据经过一系列的处理后作为 Hadoop 集群的输入提交到平台，以驱动基于 Hadoop 集群的大规模数据分析；后者是用于实现大规模数据分析的基础设施，它是由许多节点按照一定的拓扑结构组织而成的集群，能够提供高性能、按需、无限量的计算与存储服务。经过集群分析得到的结果再导出到 WebTerminal 终端，最终反馈并展示给用户。

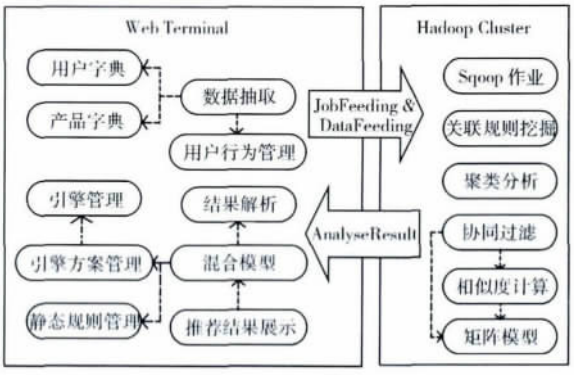

图 3　基于 Hadoop 的混合推荐系统架构

3.1　WebTerminal 端

WebTerminal 端包含数据访问层、应用层的全部功能和用户字典、产品字典、静态规则引擎和混合模型部分的实现。

原始业务数据存储在业务系统中，通常无法直接用于推荐分析，其中含有大量的脏数据、噪音数据和缺失值，为了进行有效的分析和处理，需要进行抽取、清洗、转换

和装载。数据抽取模块用于实现上述过程，从而将数据从存储系统中抽取到 Hadoop 计算平台，如图 4 所示。针对数据的异构性和多样性，定义了一个公共抽取接口 DataExtractor 来抽象不同的数据抽取逻辑实现。

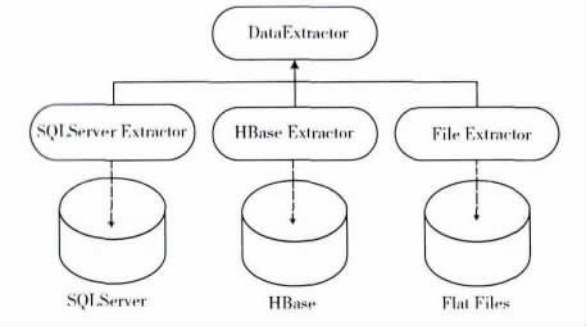

图 4 数据抽取模块

同时，利用结果解析模块可以实现将大数据分析处理的结果导回到关系数据库的功能，该模块的实现如图 5 所示。

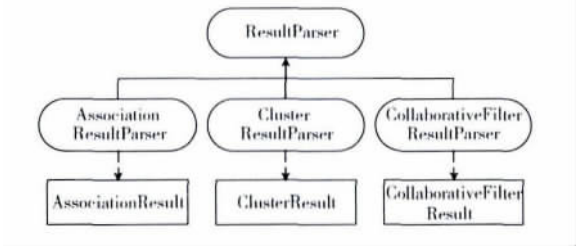

图 5 结果解析模块

Sqoop 可以通过 MapReduce 框架运行任务，实现关系型数据库和 HDFS 之间的数据导入导出[10]。这种方法简单方便，但是抽取过程不够灵活，因此数据抽取模块和结果解析模块可以作为它的有益补充，实现更细粒度、更灵活的数据迁移功能。

采用了分区混合、加权混合和随机混合 3 种不同的混合策略来实现混合推荐模型，其中利用设计模式中的策略模式来实现 3 种不同的混合策略之间的灵活切换。混合推荐模型的时序图如图 6 所示，首先是加载混合策略以及相应的配置参数，然后依次经过结果的过滤、排序、解释等过程最终为用户产生推荐列表。

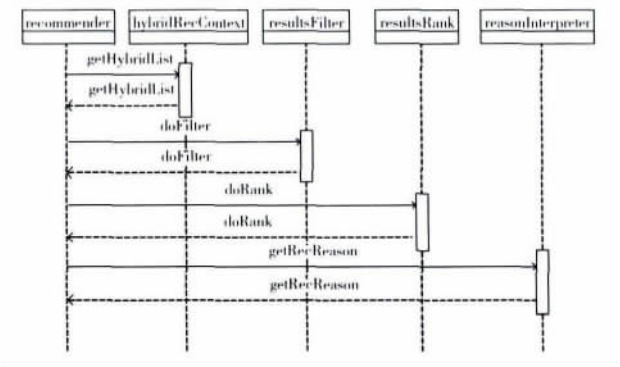

图 6 混合推荐模型时序

3.2 数据模型

推荐系统的数据资源主要包括三类：用户数据、产品数据和用户偏好数据。前两种数据分别充当用户和产品的数据字典，第三种数据是用户和产品之间的关联关系的体现。在推荐计算过程中，分别使用用户编号和产品编号来唯一索引用户与产品，利用数据字典可以获得它们的详细信息，这对产生推荐解释十分重要。在推荐计算过程中，使用产品—用户偏好矩阵来抽象用户偏好数据，该矩阵每个行向量分别对应同一种产品，每个列向量分别对应同一个用户，每个矩阵单元格分别表示一个用户对一种产品的偏好取值。

3.3 基于 Hadoop 的协同过滤推荐引擎

由于篇幅限制，这里以基于 Hadoop 的协同过滤推荐引擎为例介绍基于 Hadoop 的推荐引擎的实现。协同过滤引擎的核心包括两个部分：相似度计算组件和协同过滤算法。

协同过滤技术一般都是基于相似性来产生推荐结果，如何定义和度量实体之间的距离是其关键，采用不同的距离计算方式无论是对算法的效率还是对推荐结果的精度都有影响。系统中共实现了 6 种不同的分布式相似度计算组件：欧式距离相似度（EuclideanSimilarity）、余弦相似度（CosineSimilarity）、皮尔森相关度（PearsonSimilarity）、Jaccard 系数（JaccardSimilarity）、对数似然相似度（LoglikelihoodSimilarity）和共现相似度（CooccurrenceSimilarity）。它们都是针对向量实现的适合分布式计算环境下的相似度计算组件，是实现分布式协同过滤的基础。

采用了基于 MapReduce 的 Item-basedCF 算法来实现协同过滤，该算法包含三步计算任务：第一步，根据交易数据库计算产品—产品协同矩阵；第二步，计算用户的偏好向量。该向量是偏好矩阵的列向量，向量的每个分量对应用户对每项产品的偏好值；第三步，计算协同矩阵和偏好向量的点积，产生该用户的推荐向量，这也是为该用户输出的候选推荐列表。为了基于 MapReduce 实现以上计算任务，需要将其拆分为 6 个具体步骤，详细的作业流如图 7 所示。这些计算步骤需要使用总共 12 个 MapReduceJob 来实现，其算法架构图如图 8 所示。这一系列作业流统一地由 RecommenderJob 驱动器程序进行驱动。

4 实验与分析

实验与分析通过基于 Hadoop 的混合电子商务推荐系统中的协同过滤引擎产生个性化推荐过程来说明系统的功能有效性和面对大规模数据的伸缩性。

4.1 实验设置

实验中采用的 Hadoop 集群由 5 个节点组成，其中包括 1 个 Master 节点，4 个 Slave 节点。每个节点的具体配置信息如下：Intel(R) Xeon CPU E5645，1G RAM，15G 硬

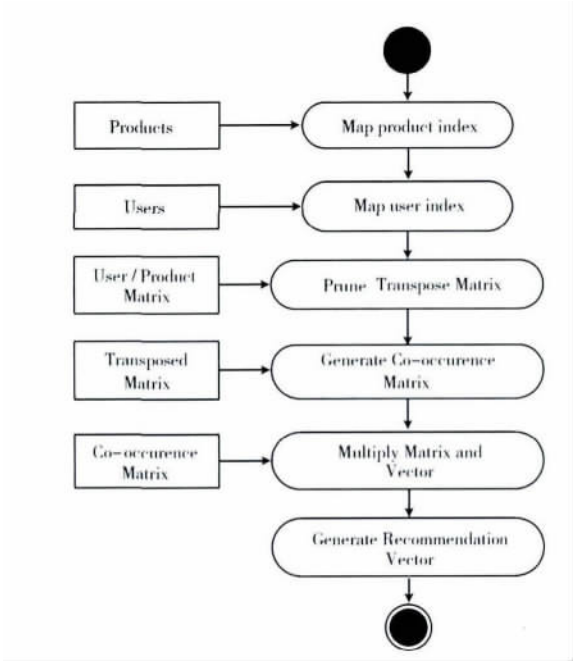

图 7　分布式协同过滤算法流程

盘，操作系统为 Cent OS 5.3，Hadoop 版本为 1.0.4，节点之间通过 100 Mbps 的网络进行连接。加速比实验中，每个 Slave 节点设置的最大 map 和 reduce 任务支持数都各为 2。此时，测试集群可以并行支持 8 个 map 任务和 8 个 reduce 任务。在此，将 map 任务数或 reduce 任务数视为计算节点的数量。通过调节每个 Slave 节点设置的最大 map 和 reduce 任务支持数来获得不同数量的计算节点。实验中采用评测协同过滤算法的经典数据集 MovieLens，其中包含 6040 位用户、3706 部电影和 1000209 条评分记录，数据的整体稀疏程度为 95.53%。

4.2　性能评估

加速比[11]（speedup）和恒等效率伸缩性[12]（isoefficiency）是评估并行系统的效率和伸缩性的两个重要的定量分析指标。前者评估固定问题规模下随着计算节点数增加运行性能的变化情况；后者使用恒等效率函数来评估评估当问题规模增大时算法的性能。

（1）加速比性能分析：Amdahl 定律是一种常用的固定规模加速比性能模型。它采用如下的公式计算加速比 S_p，$S_p = \frac{T_1}{T_p}$，其中 T_1 表示使用单个计算节点时的算法执行时间，T_p 表示使用 p 个计算节点时的并行执行时间。

若算法具有伸缩性，则当问题规模固定时，加速比与

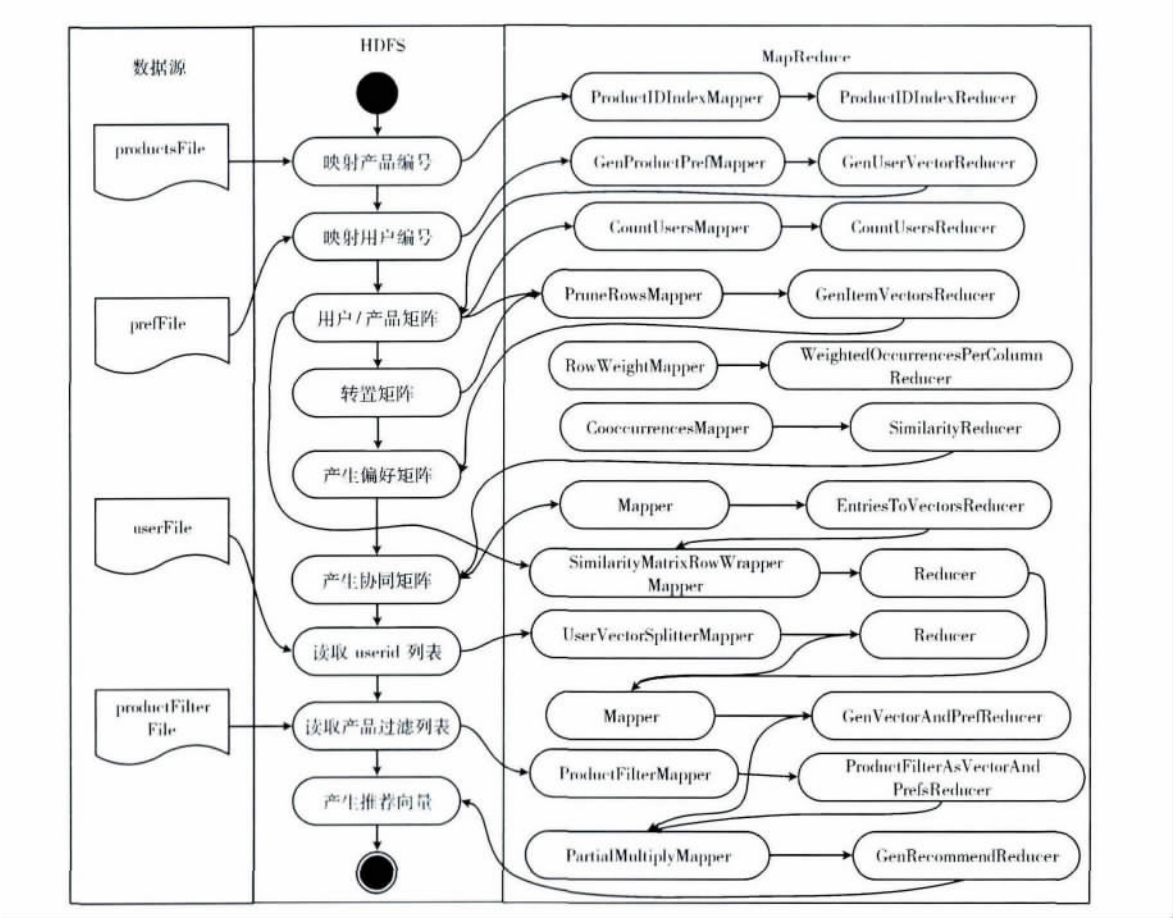

图 8　分布式协同过滤算法架构

计算节点数存在线性关系。从图 9 可看出，在 4 种不同的数据集大小（10K、100K、1M、10M）下，加速比与计算节点个数的增加近似成正比。同时，还可发现当数据集更大时可以获得更好的加速比。由此可见，基于 Hadoop 的协同过滤算法在处理更大规模的数据集的时候有更好的加速比。

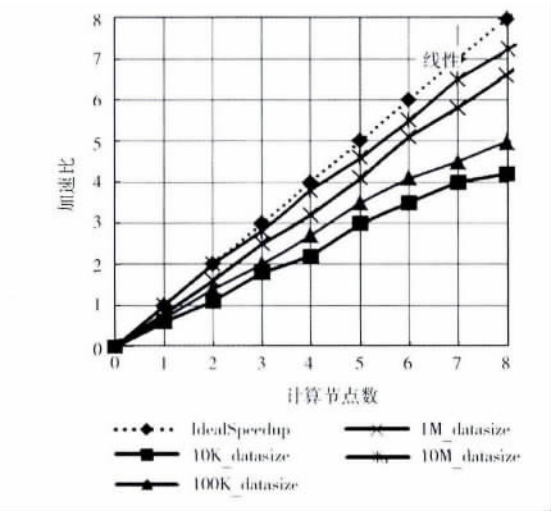

图 9 Item-basedCF 算法的加速比

（2）伸缩性分析：Isoefficiency 指标可以通过如下的公式来计算

$$E_p = \frac{S_p}{p} = \frac{T_1}{T_p \times p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + \frac{T_0}{T_1}}$$

式中：T_0——并行执行过程中的通信代价。

如果算法具有伸缩性，运行结果能保持 E_p 为常量。当 E_p 为常量时，$T_1 = f_E(P) = kP$，其中 P 表示计算节点数，因此 $f_E(P)$ 是一个线性函数。$T_1 = T(m) = lm$，其中 m 表示数据量的大小，由此可得，m 与 P 线性相关，即 $P = \frac{l}{k}m$。因此如果计算节点 P 和数据规模大小 m 线性相关，则也意味着算法具有伸缩性。如图 10 所示，当设置了 3 个不同的固定计算时间（10min、20min、30min）时，计算节点数随着数据规模增长同步线性增长。同时，还可以发现，当设置的运行时间越长，算法具有越好的伸缩性能。

综上所述，实验表明基于 Hadoop 的电子商务推荐系统在执行协同过滤时具有较好的伸缩性和性能。

5 相关工作

Gong[13] 针对协同过滤推荐技术存在的数据稀疏性问题，采用了关联规则挖掘和 User-based 协同过滤两种技术实现个性化的推荐系统，该研究表明利用不同的推荐技术可以互补各自的不足之处。

Zhao[14] 针对 User-based 协同过滤算法的伸缩性问题，

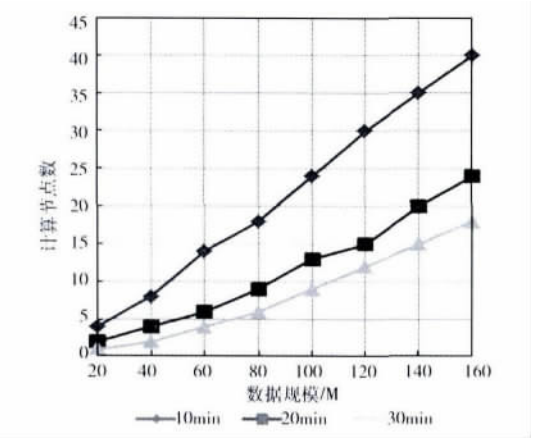

图 10 时间固定时，节点个数与数据规模的效率函数

实现了基于 Hadoop 平台的 User-based 协同过滤算法，从而实现算法的线性伸缩。

MyMediaLite[15] 作为一个轻量级多功能的推荐系统算法库，它解决了两种最常见的协同过滤场景：评分预测和基于隐式反馈的预测，提供了常见推荐算法的高效和易伸缩的版本实现。

提出的基于 Hadoop 的电子商务推荐系统从伸缩性、灵活性和多样性出发，运用基于 MapReduce 的算法和易于配置的混合推荐模型，实现面向大型电子商务的推荐系统，突破了集中式推荐系统架构的瓶颈。

6 结束语

提出了一种基于 Hadoop 的混合电子商务推荐系统的设计与实现，为使用 Hadoop 构建可伸缩、高性能的分布式电子商务推荐系统提供了参考。系统以基于 MapReduce 的推荐算法为核心，采用易于扩展的开放式推荐流程和架构，利用混合推荐模型，允许用户在应用层灵活配置引擎方案，同时易于扩展以支持更多的算法、策略和引擎。结合 Hadoop 的分布式计算的潜力和混合推荐的优势，构建具有高伸缩性、灵活性和多样性的推荐系统，能够有效地解决大型电子商务系统中的信息过载问题，为电子商务企业在数据驱动的个性化消费时代产生持续的竞争优势。

参考文献：

[1] Information overload [EB/OL]. [2013-04-20.] http://en.wikipedia.org/wiki/Information_overload.

[2] Ricci F, Shapira B. Recommender systems handbook [M]. Springer, 2011.

[3] Pu P, Chen L, Kumar P. Evaluating product search and recommender systems for E-commerce environments [J]. Electronic Commerce Research, 2008, 8 (1-2): 1-27.

（下转第 143 页）

[5] Leroy X. Verified squared: Does critical software deserve verified tools? [J]. ACM SIGPLAN Notices. ACM, 2011, 46 (1): 1-2.

[6] Paulin-Mohring C. Introduction to the Coq proof-assistant for practical software verification [G]. LNCS 7682: Tools for Practical Software Verification. Berlin: Springer Berlin Heidelberg, 2012: 45-95.

[7] Bertot Y. A short presentation of Coq [G]. Lecture Notes in Computer Science 5170: Theorem Proving in Higher Order Logics. Berlin: Springer Berlin Heidelberg, 2008: 12-16.

[8] Le Sergent T. SCADE: A comprehensive framework for critical system and software engineering [G]. Lecture Notes in Computer Science 7083: Integrating System and Software Mode-ling. Berlin: Springer Berlin Heidelberg, 2012: 2-3.

[9] Morrisett G. Technical perspective a compiler's story [J]. Communications of the ACM, 2009, 52 (7): 106-106.

[10] Yang X, Chen Y, Eide E, et al. Finding and understanding bugs in C compilers [J]. ACM SIGPLAN Notices, 2012, 47 (6): 283-294.

[11] Klein G, Elphinstone K, Heiser G, et al. SeL4: Formal verification of an OS kernel [C] //Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. ACM, 2009: 207-220.

[12] Blazy S, Leroy X. Mechanized semantics for the Clight subset of the C language [J]. Journal of Automated Reasoning, 2009, 43 (3): 263-288.

(上接第 136 页)

[4] Ricci F, Rokach L, Shapira B. Introduction to recommender systems handbook [M]. Recommender Systems Handbook. Springer US, 2011: 1-35.

[5] Wang H F, Wu C T. A strategy-oriented operation module for recommender systems in E-commerce [J]. Computers & Operations Research, 2012, 39 (8): 1837-1849.

[6] Wei K, Huang J, Fu S. A survey of e-commerce recommender systems [C] //International Conference on Service Systems and Service Management. IEEE, 2007: 1-5.

[7] Hadoop [EB/OL]. [2013-04-20.] http://hadoop.apache.org.

[8] Chuck Lam. Hadoop in action [M]. Manning Publications, 2011: 15-16.

[9] White T. Hadoop: The definitive guide [M]. 2nd ed. O'Reilly Media. Yahoo Press, 2010: 165.

[10] Sqoop [EB/OL]. [2013-04-20.] https://github.com/cloudera/sqoop/wiki.

[11] Hill M D, Marty M R. Amdahl's law in the multicore era [J]. Computer, 2008, 41 (7): 33-38.

[12] Hanuliak P. Analytical method of performance prediction in parallel algorithms [J]. Open Cybernetics & Systemics Journal, 2012, 6: 38-47.

[13] Gong S J. Personalized recommendation system based on association rules mining and collaborative filtering [J]. Applied Mechanics and Materials, 2011, 39: 540-544.

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

[15] Gantner Z, Rendle S, Freudenthaler C, et al. MyMediaLite: A free recommender system library [C] //Proceedings of the Fifth ACM Conference on Recommender Systems. ACM, 2011: 305-308.