基于MapReduce的分布式ETL多维数据模型研究

宋 杰 郝文宁 陈 刚 靳大尉 赵 成
(解放军理工大学 南京 210007)

摘 要 针对MapReduce缺少对ETL上层数据模型的具体描述,提出了一种集成的基于MapReduce的分布式ETL(MapReduce Distributed ETL,简称MDETL)多维数据模型处理方法其,把对数据的处理分解成对数据属性(维和事实)的处理,解决了ETL上层具体数据模型的构建问题。用真实的数据集评估了它的性能,实验结果表明MDETL具有很好的可扩展性。

关键词 ETL,MapReduce,MDETL,维,事实
中图法分类号 TP311 **文献标识码** A

Research of Distributed ETL Dimensional Data Model Based on MapReduce

SONG Jie HAO Wen-ning CHEN Gang JIN Da-wei ZHAO Cheng
(PLA University of Science & Technology, Nanjing 210007, China)

Abstract Because MapReduce lacks support for high-level ETL specific constructs, this paper presented a parallel dimensional ETL framework based on MapReduce (MapReduce Distributed ETL—MDETL), which exhibits the data processing to the composable property(the processing of dimensions and facts), directly supports high-level ETL-specific dimensional constructs. This paper evaluated its performance on large realistic data sets. The experimental results show that MDETL achieves very good scalability.

Keywords ETL, MapReduce, MDETL, Dimensions, Facts

在数据仓库的构建过程中,ETL主要负责从不同的数据源中抽取数据,并按照用户指定的规则和需求来进行转换和清洗,最终加载到数据仓库中去[1]。随着数据量越来越大,如何对海量数据进行ETL处理面临着新的挑战。海量数据使得ETL工作变得非常耗时,但是在实际应用中,留给ETL处理数据的时间实际上是非常少的。因此,分布式的应用成为了解决问题的关键。近年来,"云计算"、MapReduce[2]等分布式计算技术被广泛地应用到数据处理分布式计算领域。

在ETL过程中,数据处理就是对数据属性的处理,而数据的属性也就是数据的维和事实,在计算过程中,这些维和事实可以被分割成很多小的集合进行分别计算,然后再将分别计算的结果合并成最终结果加载到数据仓库中去。这种"分割-合并"的思想同MapReduce程序的思想非常类似,因此两者可以有效地结合起来。

ETL工作流由于过多的ETL具体操作,如转换、清洗、过滤、汇总、加载等,使得其本质上十分复杂。编写ETL高度并行化的程序和分布式程序也都十分有难度。因而将ETL程序部署到分布式的环境中运行也非常耗时耗力,而且容易出错。而MapReduce具有很灵活的编程模式,可以部署在普通的PC机上,而且MapReduce框架自身就已经具备了通信、容错、负载平衡和调度机制,使得ETL分布式计算只需按照框架编写相关的程序,而不用考虑框架自身如何运行。所以我们只需要将MapReduce程序应用到ETL程序中去即可[3]。

MapReduce已经被应用到很多分布式的程序中,它有自己的一套方法来处理海量数据。但是,MapReduce缺少对ETL上层数据模型的具体描述和方法支持,导致ETL程序的效率十分低下。针对数据仓库数据模型的描述,如星形模式、雪花型模式、SCDs(缓慢增长维)等,本文建立了一套基于MapReduce的分布式ETL多维数据模型处理方法,使用户可以用更少的代码去构造分布式的基于MapReduce的ETL工作流程序。

1 MapReduce并发处理海量数据编程模型

MapReduce是2004年由Google提出的面向海量数据集处理的并行编程模型[4],起初主要用作互联网数据的处理,例如文档抓取、倒排索引的建立等。由于其简单而强大的数据处理接口和对大规模并行执行、容错及负载均衡等实现细节的隐藏,该技术一经推出便迅速在机器学习、数据挖掘、数据分析等领域得到广泛应用。

MapReduce将数据处理任务抽象为一系列的Map(映射)-Reduce(化简)操作对。Map主要完成数据的过滤操作,Reduce主要完成数据的聚集操作。输入输出数据均以〈key,value〉格式存储。由Map函数和Reduce函数定义输入键值对表、产生输出键值对表。

Map:(k1,v1)->list(k2,v2)

本文受国家自然科学基金资助项目(70971137)资助。

宋 杰(1986—),男,硕士生,主要研究方向为分布式海量数据处理、分布式ETL,E-mail:adamsongs@163.com。

Reduce:(k2,list(v2))->list(v3)

Map 函数由用户定义,定义输入键值对(k1,v1),产生中间键值对(k2,v2)。然后 MapReduce 将有相同中间键 k2 的中间值分组,并通过用户自定义的 Reduce 函数传递。接着将 k2 和 k2 对应的值表定义为键值对,将这些值合并,构造出可能更小的值表 list(v3)。

除了 Map 和 Reduce 函数接口,MapReduce 框架还提供了输入文件的读取、数据分类、合并 Map 输出、分类、文件输出等 5 种标准化的编程接口。用户可以根据需要细化或者扩展这些接口。MapReduce 框架通过在集群计算机上运行实现好的接口来实现并行计算,由每台计算机计算一块数据集。用户在使用该编程模型时,只需按照自己熟悉的语言实现 Map 函数和 Reduce 函数即可,MapReduce 框架会自动对任务进行划分以做到并行执行[5]。

2 基于 MapReduce 的分布式 ETL 多维数据模型处理

MapReduce 是一个通用的并行编程模型,但它缺少对数据仓库 ETL 数据处理上层具体数据模式如星形模式、雪花型模式、缓慢增长维等常用模式的支持描述,这就导致了在使用 MapReduce 计算模型对数据进行 ETL 处理时的效率低下。本文很好地解决了 ETL 上层具体数据模型的构建问题,提出了一系列用来处理多维数据模型的方法,这些数据可能是星形模式、雪花型模式、缓慢增长维和数据密集维等各种数据模式。图 1 说明了基于 MapReduce 的 MDETL 框架中的多维数据处理流程。

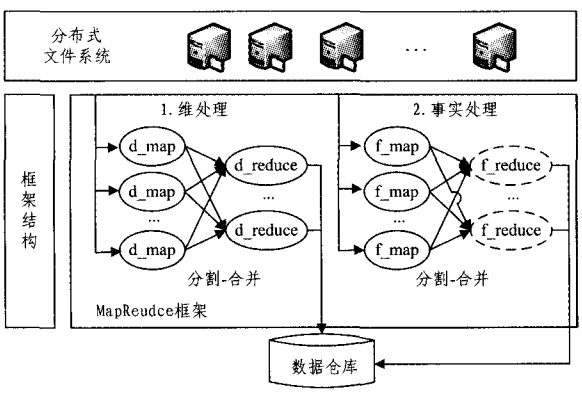

图 1 基于 MapReduce 框架的 ETL 多维数据处理流程

在 MDETL 多维数据模型处理过程中,首先其中一项 MapReduce 任务完成维处理,然后另一项 MapReduce 任务完成事实处理。一项 MapReduce 任务可以被分为很多并行的 Map 任务和 Reduce 任务来分别执行维处理和事实处理。每一项任务都由几步构成,包括从分布式文件系统中读取数据,运行 Map 函数,分类,合并 Map 输出,运行 Reduce 函数,输出结果等等。在维处理过程中,维表的输入数据可以由不同的方法来处理,例如,维数据可以被一项任务处理,也可以被所有任务处理。在事实处理过程中,事实表的数据被分成很多等大小的数据集来由并行任务执行。这包括了查询维键和将处理好的事实数据批加载到数据仓库中。在 Reduce 程序里,如果在事实数据被加载前没有聚合,那么事实数据的处理步骤就可以被省略。

2.1 数据集的分割

在一个 MapReduce 任务开始之前,从不同数据源汇集来的数据集需要先被分割成很多近似相等大小的数据集,再分布到 Map 和 Reduce 任务中,然后加工成维或事实。

MDETL 提供了下面两种分割方法:

1)循环分割法。这个方法在任务之间分布成很多行,如果有 nr_map 项任务,这样行数 n 分配到的任务数即 n 取模 nr_map。这样就可以保证输入数据集均匀地分到各个任务中去。这个方法更适合于维数据被所有任务处理的时候;

2)哈希值分割法。这个方法定义了一个或者更多属性作为分割属性。在分割属性时具有相同哈希值的元组将被分到同一个任务中去。如果有 nr_map 项任务,哈希值为 hash 的元组的任务数即为 hash 取模 nr_map。这个方法更适合于具有相同哈希值属性的所有元组被一个单独任务执行的时候。

MDETL 分别提供了以上两种分割方法的 Map 函数读入方法。另外,大多数的 MapReduce 框架都提供不同的读入方法以供选择,而且允许用户自己定义其读入方法[6]。

2.2 维处理

在维处理过程中,维输入数据集被所有的 Map/Reduce 任务用配置好的处理方法进行处理。其中最基本的就是配置一个任务处理一维。例如,有 3 维数据和 3 项任务,每项任务都处理数据集的单独一维。如果只有少数的任务被调用,就不需要上面的方法,而是采用让所有的任务都处理这一个指定维的方法,这样每个任务都只处理指定维数据集的一部分。如果维之间存在依赖关系,如雪花型模式维数据,那么将采用逐级处理的方法来处理数据并实现维的并行化,这种处理方式是指按照从枝叶到根部的顺序对雪花型维进行处理(被事实表引用的维表是根部,没有外键但引用其他维表的维表是枝叶)。存在依赖性的维表(有外键关联的)在一个连续的工作中被处理。在深入的优化过程中,维可以配置到分布式的节点中存贮起来,也可以按要求加载到数据仓库中去。

2.3 事实处理

在事实处理过程中,每个 Map/Reduce 任务都处理一部分相同大小的数据集,包括读取数据、查询维表键值、转换和加载。如果一个事实是聚合的事实,Reduce 程序就用来计算所有使用聚合函数(求和、均值、计数)的元组的量值。如果不需要聚合,就可以被省略掉 Reduce 程序来进行优化。为了提高维键值的查询速度,维数据可以被全部或者部分读取到主内存中,这样批加载就可以实时地将处理好的数据从主内存中转移到数据仓库中。

2.4 MDETL 处理流程描述

算法 1 详细描述了 MDETL 在 MapReduce 框架中运行的具体处理过程。涉及到 MapReduce 的步骤在第 2—4 行和 6—7 行,这几步操作主要负责初始化、调度处理维和事实的任务、返回处理信息。第 1 行和第 5 行是没有用到 MapReduce 的步骤,主要负责准备输入数据集和在节点之间同步各个维(如果没有分布式文件系统的话)。

算法 1 基于 MapReduce 的 ETL 处理过程

1)分割数据集;

2)读取表 1 里设置好的参数并初始化;

3)读取输入数据并将数据传递给 Map 函数;

4)处理维数据并把它加载到联机/脱机的维存储中;

5)如果没有分布式文件系统,在集群计算机之间同步维数据;

6)准备事实处理(连接并将维数据存到缓存中);

7)读取输入事实处理的数据并在 Map 函数中执行转换;

8)将事实数据批加载到数据仓库中。

在 MDETL 中,所有运行参数都在一个单独的配置文件里存储,包括数据源设置、分割策略如分割的关键词、维、事实、数据密集型维、Map 任务和 Reduce 任务的数目。表 1 总结了几个关键的配置参数。这些参数给用户提供了很大的灵活性,使用户可以将任务配置得更有效率。

表 1 关键的配置参数

参数	描述
Dim_i	维表定义,i=1,…,n
$Fact_i$	事实表定义,i=1,…,n
$Dim_i(a_0,a_1,\cdots,a_n)$	定义数据源中 Dim_i 的相关属性 $a_0,a_1,\cdots,a_n$
DimScheme	维模式,联机/脱机(缺省值为联机)
nr_reduce	Reduce 任务的数量
nr_map	Map 任务的数量

3 应用测试与性能分析

3.1 测试环境

实验是在一个通过千兆网连接的 6 个节点的计算机集群中进行的,每个机器 CPU 为四核 Intel(R) Xeon W3503 2.40GHz,24G 内存,硬盘为 SATA 350GB(3GB/s,16MB 缓存,7200RPM)。所有的节点都运行在 Linux-Ubuntu-11.10-desktop-i386 内核上,安装 Hadoop0.2.2,JDK1.6 和 MDETL 数据处理程序。计算机集群上安装了 HDFS 分布式文件系统。在每个节点都安装了 MySQL5.1 数据库管理系统。集群中,一个节点为主节点,其他节点为子节点。每个子节点都运行 4 个并行的 Map/Reduce 任务,即子节点上总共运行 20 个并行任务。分布式 ETL 的集群拓扑图如图 2 所示。

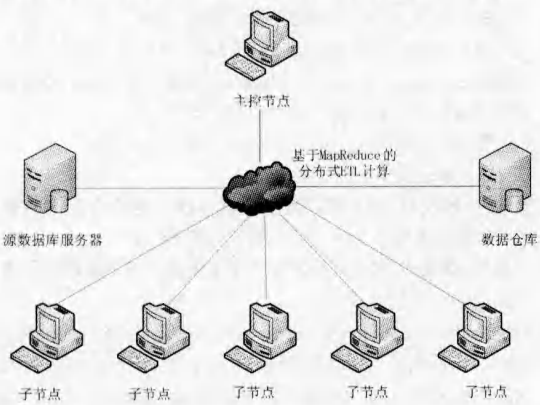

图 2 分布式 ETL 集群实验测试环境拓扑图

3.2 实验数据

实验数据来自于数据中心的一个测试实验。实验是用来自动测试数据中心 Web 页面的访问量。每个页面应用一个测试,测试结果显示出检测到的错误数目。最终这个测试的结果被存入到很多已经分割好的服务于数据源的文件里。如图 3 所示,数据用星形模式存贮到数据仓库中。星形模式比较了 1 个事实表和 3 个维表。其中,定义"页维 pagedim"为一个缓慢增长维。

实验使用数据生成器来生成测试数据。因为 MapReduce 经常处理海量的小文件而不是一个单独的大的合并文件,所以测试数据集事先被分割好,并存入很多文件中去。这些文件为处理阶段的维和事实提供输入数据来源。系统共生成了 2 个大小不一样的 pagedim 数据集:bigdim 和 smalldim。

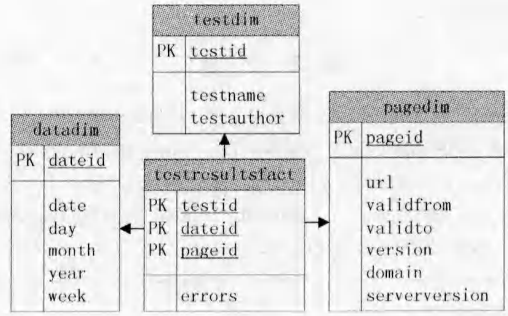

图 3 实验例子的星型模式数据

3.3 实验结果与分析

实验主要通过用不同数量的任务和不同大小的数据集的实验来评估 MDETL 的可扩展性。在维处理阶段,Map 函数负责处理数据密集型维 pagedim,Reduce 函数负责处理其他两种维 datedim 和 testdim,每个只用到单独的 Reduce 函数。在事实处理阶段,如果不需要聚集运算,就不会用到 Reduce 函数。

数据集的大小从 20GB 到 80GB 不等,通过不同任务对数据集的处理时间的消耗来测试 MDETL 的性能和可扩展性。图 4 和图 5 分别显示了 bigdim 和 smalldim 不同大小数据集用不同数量任务处理的结果。

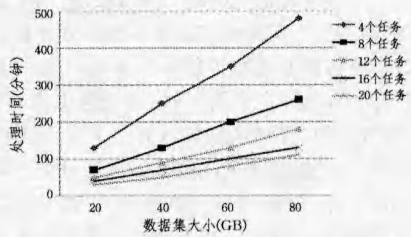

图 4 bigdim 数据集处理时间的变化曲线

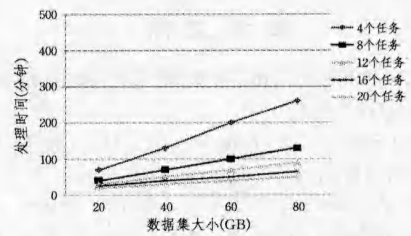

图 5 smalldim 数据集处理时间的变化曲线

从图中可以看出,随着数据集大小的变化,MDETL 的性能也跟着线性地提高。这是因为 MDETL 将数据处理拆分为维处理和事实处理,可以直接基于数据属性进行分析处理,并且其采用的计算模式是移动计算而非移动数据,所以可以将分析延迟最小化,随着数据集的加大,系统的处理效率也会得到提升。由此可知 MDETL 具有良好的性能和可扩展性。

结束语 本文基于 MapReduce 的分布式 ETL 多维数据模型的处理方法,分析了如何运用 MapReduce 计算模型分配 Map 任务和 Reduce 任务来处理不同维数据,将海量数据背景下基于 MapReduce 的分布式 ETL 数据处理分为维处理和事实处理两个阶段,并分别针对维处理和事实处理两个阶段提出了处理方法。最后实验通过大量的任务和数据集证明了 MDETL 具有良好的性能和可扩展性,为 MapReduce 对 ETL 上层数据模型的具体描述和相应的处理提供了一种可行的研

究策略方法。

参 考 文 献

[1] 徐俊,刚裴莹. 数据 ETL 研究综述[J]. 计算机科学,2011,38(4)

[2] Dean J, SGhemawat J. MapReduce: Simplified Data Processing on Large Clusters[C]// Proc. of OSDI. 2004:137-150

[3] Kovoor G, Singer J, Lujan M. Building a Java MapReduce Framework for Multi-core Architectures[C]// Proc. of MULTIPROG. 2010

[4] 王珊,王会举,等. 架构大数据:挑战、现状与展望[J]. 计算机学报,2011,10:1741-1752

[5] 李建江,崔健,等. MapReduce 并行编程模型研究综述[J]. 电子学报,2011,11:2635-2642

[6] Dean J, Ghemawat S. MapReduce: A Flexible Data Processing Tool[J]. CACM,2010,53(1):72-77

(上接第 258 页)

自适应算法。

表 4 问题转换与自适应算法结果比较

	Transformation	Adaptation
Hamming Loss	**0.201**	0.192
Accuracy	**0.711**	0.531
Precision	**0.809**	0.726
Recall	**0.669**	0.612
F-Measure	**0.695**	0.663
Subset Accuracy	**0.358**	0.201
Micro-F1	**0.676**	0.655
Macro-F1	**0.525**	0.505

结束语 为了方便 TRIZ 理论使用者进行发明创新,本文根据 40 条 TRIZ 发明原理所蕴含的冲突矛盾的相似性对原发明进行了分组,并将新生成的 20 个新的组别作为专利文件训练分类的新类别。实验比较多个不同的分类算法的分类性能,并用 8 个评估特性进行评估。从实验结果中也可以得到一些有用的信息,比如在使用 TRIZ 专利数据集时,PPT 多标签问题转换方法和 SVM 监督学习算法关联可以得到最好的效果。还有针对此数据集的问题转换方法分类效果也要优于自适应算法。选定最优的分类方法对中文专利进行面向 TRIZ 理论使用者的自动分类,以便设计者能更好地利用专利知识来辅助产品的创新和改进而服务。

参 考 文 献

[1] 左晶. IPC 和 USC 分类体系下专利检索的对比分析[J]. 现代情报,2007,1:130-132

[2] Meyer D. Support Vector Machines[J]. The Interface to libsvm in package e1071. e1071 Vignette,2012

[3] Chou K C, Shen H B. Predicting eukaryotic protein subcellular location by fusing optimized evidence-theoretic K-nearest neighbor classifiers[J]. Journal of Proteome Research, 2006, 5(8): 1888-1897

[4] 卢长林. 手扶电动整枝机[D]. 1992

[5] Elisseeff A, Weston J. A kernel method for multi-labelled classification[C]// Advances in neural information processing systems. 2001:681-687

[6] Godbole S, Sarawagi S. Discriminative methods for multi-labeled classification[M]// Advances in Knowledge Discovery and Data Mining. Berlin Heidelberg: Springer, 2004:22-30

[7] Crammer K, Singer Y. A family of additive online algorithms for category ranking[J]. The Journal of Machine Learning Research, 2003, 3:1025-1058

[8] Zhang M L, Zhou Z H. Multilabel neural networks with applications to functional genomics and text categorization[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(10):1338-1351

[9] Zhang M L, Zhou Z H. A k-nearest neighbor based algorithm for multi-label classification[C]// Granular Computing, 2005 IEEE International Conference on. IEEE, 2005, 2:718-721

[10] Tsoumakas G, Dimou A, Spyromitros E, et al. Correlation-based pruning of stacked binary relevance models for multi-label learning[C]// Proceeding of ECML/PKDD 2009 Workshop on Learning from Multi-Label Data. Bled, Slovenia, 2009:101-116

[11] Tsoumakas G, Katakis I, Vlahavas I. Mining multi-label data [M]. Data mining and knowledge discovery handbook. US: Springer, 2010:667-685

[12] Read J. A pruned problem transformation method for multi-label classification[C]// Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008). 2008:143-150

[13] Zhang M L, Zhou Z H. A k-nearest neighbor based algorithm for multi-label classification[C]// Granular Computing, 2005 IEEE International Conference on. IEEE, 2005, 2:718-721

[14] Gao S, Wu W, Lee C H, et al. A MFoM learning approach to robust multiclass multi-label text categorization[C]// Proceedings of the twenty-first international conference on Machine learning. ACM, 2004:42

[15] Williams T, Domb E. Reversability of the 40 Principles of Problem Solving[J]. The TRIZ Journal, May 1998

[16] Cong H, Tong L H. Grouping of TRIZ Inventive Principles to facilitate automatic patent classification[J]. Expert Systems with Applications, 2008, 34(1):788-795

[17] Pro_Techniques[OL]. http://www.iwint.com.cn/

[18] CREAX http://www.creax.com

[19] 费洪晓,康松林,朱小娟,等. 基于词频统计的中文分词的研究[J]. 计算机工程与应用,2005,41(7):67-68

[20] 王素格,魏英杰. 停用词表对中文文本情感分类的影响[J]. 情报学报,2008,27(2):175-179

[21] Erk K, Padó S. A structured vector space model for word meaning in context[C]// Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2008:897-906

[22] 任永功,杨荣杰,尹明飞,等. 基于信息增益的文本特征选择方法[J]. 计算机科学,2012,39(11):127-130

[23] Tsoumakas G, Vlahavas I. Random k-labelsets: An ensemble method for multilabel classification[C]// Machine Learning: ECML 2007. Berlin Heidelberg: Springer, 2007:406-417

[24] Yang Y, Liu X. A re-examination of text categorization methods [C]// Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999:42-49

[25] Tsoumakas G, Xioufis E S, Vilcek J, et al. MULAN: A Java Library for Multi-Label Learning[J]. Journal of Machine Learning Research, 2011, 12(7):2411-2414

[26] Hall M, Frank E, Holmes G, et al. The WEKA data mining software: an update[J]. ACM SIGKDD Explorations Newsletter, 2009, 11(1):10-18