Journal of Computer Applications
计算机应用,2013,33(4):1026 - 1030
ISSN 1001-9081
CODEN JYIIDU
2013-04-01
http://www.joca.cn

文章编号:1001 - 9081(2013)04 - 1026 - 05
doi:10.3724/SP.J.1087.2013.01026

面向 Web 电子产品信息分布式检索系统的设计与实现

张渊源[1],张琴燕[2*],蒋关富[3]
(1.浙江中医药大学 信息技术学院,杭州 310053; 2.浙江大学 计算中心,杭州 310058;
3.浙江大学 计算机科学与技术学院,杭州 315100)
(*通信作者电子邮箱 zqy_hellen@sina.com.cn)

摘 要:为了从这些海量信息中获取"有用的、满足用户需求的信息",提出一个基于 Hadoop 和 Lucene 技术的分布式检索系统架构处理 Web 电子产品信息检索。利用 Hadoop 的 Map 和 Reduce 实现分布式索引文件的存储,通过 Lucene 检索技术实现索引文件的访问,从而提高信息检索的效率。并且针对 Lucene_Hadoop 架构存在粗粒度检索问题,提出了一种细粒度检索方法,减少了系统建立索引的时间。实验表明基于 Hadoop 和 Lucene 的分布式检索系统在 Web 电子产品信息中具有较高的检索性能。

关键词:分布式检索系统;Web 电子产品信息;Hadoop;Lucene;细粒度检索
中图分类号:TP311.5;TP391.3 **文献标志码**:A

Design and implementation of distributed retrieval system for electronic products information

ZHUANG Yuanyuan[1], ZHANG Qinyan[2*], JIANG Guanfu[3]
(1. *College of Information Technology, Zhejiang Chinese Medical University, Hangzhou Zhejiang* 310053, *China*;
2. *Computer Center, Zhejiang University, Hangzhou Zhejiang* 310058, *China*;
3. *School of Computer Science and Technology, Zhejiang University, Hangzhou Zhejiang* 315100, *China*)

Abstract: In order to obtain the useful information that can satisfy the user requirements, this paper proposed a distributed information retrieval system based on Hadoop and Lucene handling the Web electronic products information retrieval. In order to improve the retrieval efficiency, using the Map and Reduce method of Hadoop technology implemented the storage of distributed index files and using Lucene technology implemented the file access of distributed index files. At the same time, it also proposed an improved method at fine grain retrieval level, which reduced the index building time. The experiment demonstrates that our distributed information retrieval system has a good retrieval performance for Web electronic products information.

Key words: distributed information retrieval system; Web electronic products information; Hadoop; Lucene; fine grain retrieval

0 引言

互联网上的信息量呈爆炸性的趋势增长,出现了极为丰富的数据资源,使互联网变成了一个巨大的、分布广泛的全球性信息服务中心,涉及微博、新闻、电子商务、视频、音乐、新闻、教育、健康等诸多领域[1]。其主要挑战包括如下几点:1)互联网上的大部分商业网站以非结构化或者半结构化的形式展现,其中以超文本标记语言(HyperText Markup Language, HTML)展现形式居多,缺乏语义特征,很难精确、稳定、高效地从中提取出信息。2)商业信息的海量性,必然需要一个能够高效快速地抓取商业信息的应用系统。3)随着搜索引擎的不断发展,人们对信息的检索速度、精确度等提出了更高的要求。4)商业信息的海量性,必然导致分布式存储系统的产生,即如何在分布式存储系统的基础上执行高效的分布式计算已成为一大重要挑战。因此,如何有效地集成 Web 数据,为中小型企业提供完善的市场情报分析支持,为用户提供更简便的商品信息浏览体验,具有极其重大的现实意义。

而信息融合系统作为目前软件演化的重要技术,为应对这一挑战提供了一条可行之路。信息融合系统主要参照垂直搜索引擎的体系结构,基于 Web 数据挖掘技术,完成互联网上各类信息提取与融合。目前,在信息融合系统中,Web 信息检索技术最为复杂。文献[2]设计并实现分布式海量结构化数据存储检索系统。该系统采用列存储结构,采用集中分布式 B + Tree 索引和局部索引相结合的方法提高检索效率。在此基础上讨论复杂查询条件的任务分解机制,支持大数据的多属性检索、模糊检索以及统计分析等查询功能;然而,作者未就如何对查询结果进行规范化输出以满足用户需求进行讨论。文献[3]提出了一个基于内容及相似搜索的对等音乐文件共享系统。该系统利用了集合对等点来完成音乐文件的元数据的注册和搜索,把音乐文件的属性名 - 属性值对(Attribute-Value Pair, AV-Pair)通过音乐文件描述说明(Music File Description, MFD)来表示,使系统可以支持精确的检索;但是该方法需要花费大量时间建立目标属性名 - 属性值之间的索引关系。文献[4]提出了一种基于 Solr 的分布式实时搜索模型,分析了其实现原理。模型通过内存索引与磁盘索引相结合保证索引信息的实时展示,同时引入 CommitLog 日志保证内存索引数据容灾,并通过主从模型 Master/Slave 保证搜索服务的可用性。该方法实现具有较好的可行性,能解决目前中小企业内部的搜索引擎系统的需求。文献[5]提出了一种网格环境中基于本体的信息检索体系模型。利用 Globus 和 OGSA-DAI 工具进行计算资源和数据资源的管理,整合了闲置资源,提高了资源利用率,同时,将数据访问服务化,统一了接口访问类型;但是该方

收稿日期:2012-09-06;**修回日期**:2012-11-07。
基金项目:浙江教育厅科研项目(Y201225127);浙江省自然科学基金资助项目(LY12F02029);国家科技支撑计划项目(2011BAH16B04)。
作者简介:张渊源(1980 -),女,浙江杭州人,讲师,硕士,主要研究方向:信息系统、医疗软件中间件; 张琴燕(1976 -),女,浙江杭州人,工程师,硕士,主要研究方向:语义 Web、中间件; 蒋关富(1986 -),男,浙江杭州人,硕士研究生,主要研究方向:信息融合。

法需要进行一步研究如何动态更新本体信息,且统一接口访问类型实现起来比较复杂。

在分布式检索技术实际应用过程中:一方面随着被索引文件的增多,建立索引时间呈线性增长;另一方面在搜索引擎应用中,当索引文件量达到一定等级时,搜索引擎就遇到性能瓶颈。基于此,本文结合 Hadoop 和 Lucene 技术,实现了面向电子产品领域的分布式检索系统。同时,深入研究分布式基础架构 Hadoop 技术,提出了部分细粒度的改进,降低了系统建立索引的时间。本文实验表明基于 Hadoop 和 Lucene 的分布式检索具有较高的检索性能。

1　相关工作

国内外针对 Web 信息检索技术研究很多,比较典型的开源系统有 Solr[4,6] 企业级搜索服务器开发框架和 Nutch[7] 搜索引擎等,但是它们都存在各自的问题。在 Solr 的分布式搜索中,Master 节点容错性不足。因此它一旦停止,系统就不能为新的文档创建索引或执行复制了;此外,并不是所有的 Solr 1.3版本都是分布式感知的。其搜索、分类、调试和突出显示组件是分布感知的,而其他的不常用的组件,正在努力实现这个功能。在 Nutch 的分布式检索中,它主要面向各种类型的文本类型,针对全文进行检索,没有很好地考虑各种文档类型的结构蕴含的特殊意义;其次,它主要面向传统搜索引擎实现,并不面向垂直搜索引擎,也不能很好地根据领域性的知识来进行分类检索。

本章主要对研究中涉及的 Hadoop 和 Lucene 相关技术作简单介绍,包括基于分布式系统基础架构 Hadoop 的 Map-Reduce 和基于 Lucene 的全文检索技术。

基于分布式系统基础架构 Hadoop 的 Map-Reduce[8] 编程框架主要由作业管理器与任务管理器两部分组成。Map-Reduce 分布式计算模型如图 1 所示,其中作业管理器主要完成以用户提交的作业为单位(作业单元)的作业调度,包括作业的优先级调度以及作业的开始、处理、终止等计算调度。任务管理器具体负责执行用户定义的操作,在作业单元的基础上,进行作业的分解和归并,产生相应的任务集,包括 Map 任务与 Reduce 任务,任务是程序执行的基本单元(任务单元)。任务管理器执行过程中需要向作业管理器发送心跳信息,汇报每个任务的执行状态,帮助作业管理器收集作业执行的整体情况,为下次任务的分配提供依据。

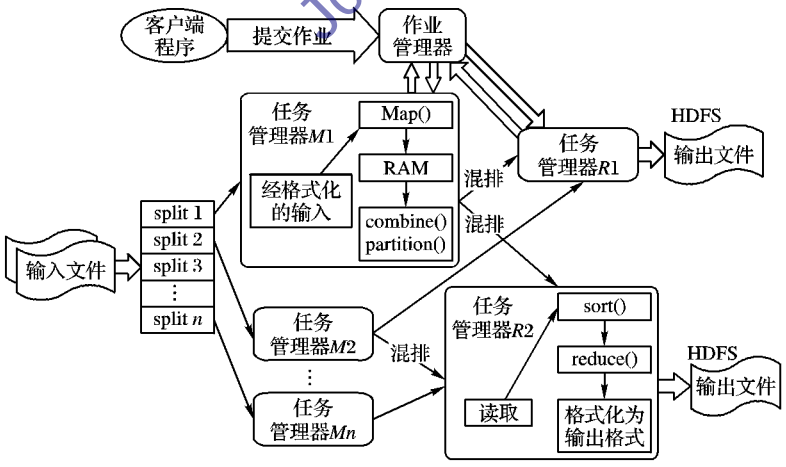

图 1　Map-Reduce 分布式计算模型

最基本的 Map-Reduce 程序至少包含 3 个部分[9]:一个 Map 函数、一个 Reduce 函数和一个入口函数(Run 函数)。在 Hadoop 架构中,分布式系统基础架构 Hadoop 的重要基础主要依赖于 Hadoop 分布式文件系统(Hadoop Distributed File System, HDFS),其主要实现分布式文件存储和防范[10]。HDFS 是一个高度容错的分布式文件系统,它能够提高数据访问的吞吐量,因此适合存储海量的大文件。

Lucene 最初是 Apache 软件基金会 Jakarta 项目组的一个子项目[11],是开源的全文检索引擎工具包。Lucene 的核心是采用倒排索引技术的全文索引引擎。该引擎通过预先设定的哈希函数为每个检索词生成唯一的哈希值并存储在文件系统中,这样查找效率就要比关系型数据库高得多。

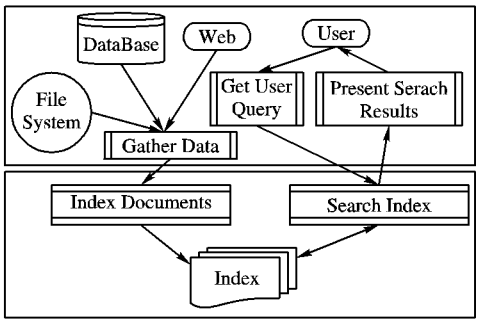

图 2　Lucene 创建索引及检索示意图

全文检索的基本步骤包括:1) 被索引的文档用 Document 对象表示;2) 索引建立器 IndexWriter 通过函数 addDocument 将文档添加到索引中,实现创建索引的过程;3) Lucene 的索引是应用反向索引;4) 当用户有请求时,Query 代表用户的查询语句;5) 索引查询器 IndexSearcher 通过函数 search 搜索 Lucene Index;6) 索引查询器 IndexSearcher 计算词条权证 term weight 和词条范围 score 并且将结果返回给用户;7) 返回给用户的文档集合用 TopDocsCollector 表示。

2　基于 Hadoop 和 Lucene 的分布式检索模型

本文基于 Hadoop 和 Lucene 的分布式检索模型称为 Lucene_Hadoop,主要包含两部分 Lucene_Hadoop_Map 和 Lucene_Hadoop_Reduce,信息存储模块主要为分布式框架提供分布式存储功能,信息检索模块主要为分布式框架提供分布式检索功能。其主要思想是通过分布式基础架构 Hadoop 提供的分布式文件读写,结合 Lucene 的索引查询技术,从不同的索引块中获取相应的搜索结果,合并结果并打分排序,返回给用户。本章将详细对这两个模型进行介绍。

2.1　Lucene_Hadoop 的 Map 端分布式检索模型

Lucene_Hadoop_Map 分布式检索模式中,分布式基础架构 Hadoop 一方面提供了分布式文件系统 HDFS 供索引等文件的存放,由命名节点 NameNode 提供文件备份等容错机制,另一方面提供了 Map-Reduce 分布式编程模型,由作业跟踪器 JobTracker 完成分布式任务的调度管理。图 3 展示了 Lucene_Hadoop_Map 分布式检索模式,主体包含三部分:1) Map 操作,利用 Lucene 完成了索引的建立;2) Reduce 操作,完成索引的归并;3) 分布式搜索。

Map 操作包括:分布式基础架构 Hadoop 平台根据需要处理的文本的输入路径,分成多个 Map 处理。每一个 Map 中,通过调用 Lucene 的 IndexWriter 的方法生成相应的 Input HDFS Block 的索引块,同时 Map 以 < KEY, VALUE > 的格式输出,其中 KEY 是通过一致性哈希(Consistent Hashing,CH)算法生成的 ID,VALUE 是当前文档索引块的输出路径。Reduce 操作包括,收集 Map 输出的 KEY 为同一 ID 的 List < VALUE >,通过 Lucene 的合并索引文件方法,即函数 MergeIndex,合并各个不同路径的索引块,并写入到分布式文件系统 HDFS 中。

此时 Lucene_Hadoop_Map 分布式检索模式具有编写简单、生成索引快速的优点，然而建立索引时针对整个 Input HDFS Block，没有对生成的索引进行归类，提供多层次的索引，导致系统在搜索结果的时候，必须搜索所有的索引块进行全局搜索，降低了搜索的性能。为了解决以上问题，针对面向电子产品领域特点，项目进一步设计并实现了另一种分布式索引模式 Lucene_Hadoop_Reduce 模式。

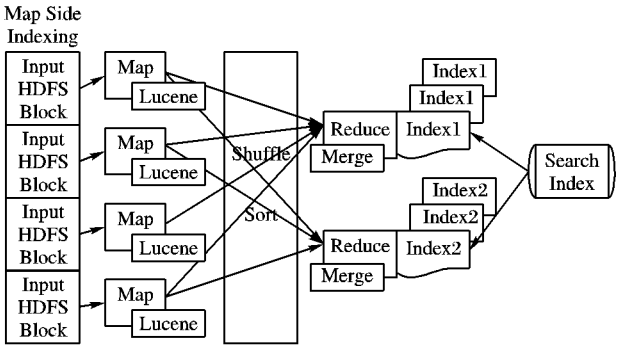

图 3 Lucene_Hadoop 的 Map 端分布式索引模型

2.2 Lucene_Hadoop 的 Reduce 端分布式检索模型

Lucene_Hadoop_Map 分布式索引模型如图 4 所示。

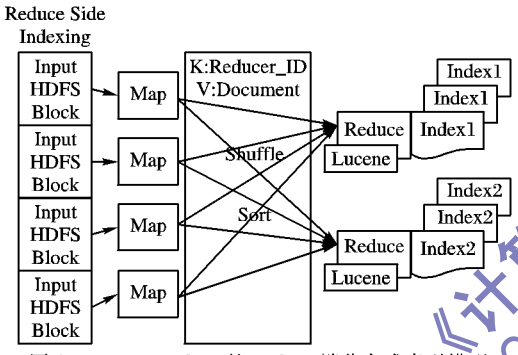

图 4 Lucene_Hadoop 的 Reduce 端分布式索引模型

Lucene_Hadoop_Map 分布式索引模式利用了 Map 端，完成了索引的建立，再通过 Reduce 端的合并操作完成了索引的合并，它只是简单地将索引分成几个部分，不能有效地进行分类，导致 SearchIndex 在搜索内容的时候只能采取全局的查询方式，降低了查询的效率。在面向电子产品领域，Input HDFS Block 中存放的是电子产品信息集合数据，电子产品信息中包含电子产品类别等信息。因此，Lucene_Hadoop_Reduce 模式考虑在 Map 端通过电子产品文本解析建模获取相关电子产品类型，根据电子产品类型分配到不同的 Reduce 进行索引建立的计算任务。相对于 Lucene_Hadoop_Map 模式建立的索引块，SearchIndex 可以通过用户界面选择电子产品类别，从相应的类别索引块进行搜索，实现了局部搜索，提高了查询的效率。

Lucene_Hadoop_Reduce 分布式索引模式主体包含两部分：1）Map 操作，完成电子产品文本信息的建模及分配；2）Reduce 操作，完成索引的建立。由于该模式面向电子产品领域，涉及到文本信息的电子产品建模，因此以下将针对 Map 设计、Reduce 设计，以及电子产品信息抽取详细展开论述。

2.2.1 Lucene_Hadoop_Reduce 端的 Map 设计

Map-Reduce 分布式编程模型的作业 Job 需要从 Input HDFS Block 中读取纯文本电子产品信息数据集合，通过 InputSplit 分块分成多个 Map 程序。每一个 Map 程序以 <KEY，VALUE> 格式遍历读取相应的 InputSplit 数据块中的数据，其中 VALUE 为文本格式的电子产品信息。如算法 1 所示。Map 需要对文本格式的电子产品信息进行解析处理，把相应的电子产品信息内容加载到电子产品对象 ProductModel 中，同时采用一定的线程数并行处理，线程数需要根据硬件配置相应的调整（采用 CPU 核的数量乘以 2 加上 1 为线程数量）。然后根据产品对象模型 ProductModel 的电子产品类型，Map 把电子产品对象写到相应的中间结果。Map 程序处理完后，中间计算结果最后会被写入计算节点的本地磁盘上待相应的 Reduce 获取处理。

算法 1 解析 InputSplit 数据中的电子产品信息且转化为对象，并根据类型输出到中间结果。

输入 Texe key：产品数量；Text value：电子产品文本信息。

输出 写入中间结果。

```
Procedure void map(Text key, Text value, Context context)
  //根据硬件配置的 cpu 内核数，生成相应的线程池来并发处理
  //文本信息
  ExecutorService executor = Executors. newFixedThreadPool
    (Runtime. getRuntime. availableProcessors() * 2 + 1);
  //获得的文本内容解析加载为电子产品 Model
  ProductModel model = (ProductModel) Tools. parse(value);
  //根据电子产品 Model，获取电子产品的类型
  Type type = model. getType();
  //伪代码，获取电子产品 Model 的类型的 hash 值，保证唯一性
  String id = type. getTypeID();
  //伪代码，根据获得的 id，把电子产品发送到相应的 Reduce 类
  context. write(new Text(id), value);
End of Procedure
```

2.2.2 Lucene_Hadoop_Reduce 端的 Reduce 设计

Reduce 程序首先从多个 Map 程序所占的 TaskTracker 节点上获取中间计算结果，即分配到同一 Reduce 的电子产品对象 ProductModel。Reduce 根据电子产品对象，构建一个 Document 对象，该 Document 对象相应地包括几个 Field，该 Field 需要设置两个参数：Field. Index 和 Field. Store。其中 Field. Index 的参数可以被设置成为 TOKENIZED、UN_TOKENIZED、NO，分别表示为检索且分词、检索且不分词（代表以整个词作为索引）、不检索。Field. Store 的参数可以被设置成为 YES、NO，分别代表存储和不存储。其中 Field. Index 和 Field. Store 的 NO、NO 组合是无效的。各个组合如表 1 表示。

表 1 Field. Index 和 Field. Store 组合

Field. Index	Field. Store	组合意义
TOKENIZED	YES	被分词索引且存储，一般针对文本标题或摘要
TOKENIZED	NO	被分词索引但不存储，一般针对较长文本
UN_TOKENIZED	YES、NO	不被分词，它作为一个整体被搜索部分
NO	YES	这是不能被搜索的，它只是被搜索内容的附属物，如 URL 等
NO	NO	无效用法

根据以上组合，电子产品对象的各个字段被相应的设计为如表 2 组合。

表 2 Document 各个 Field 的设置

Document. Field	ProductModel. Filed	Setting
Name	Name	TOKENIZED, YES
Type	Type	UNTOKENIZED, YES
Site_Info	Site_Info	UNTOKENIZED, YES
Content	ProductModel	TOKENIZED, NO
Other_Info	Other_Info	UNTOKENIZED, YES
Image	Image	UNTOKENIZED, YES
Price	Price	UNTOKENIZED, YES

在 Reduce 程序处理过程中,涉及到几个问题:1) 中文分词器支持,由于 Lucene 的默认分词器 StandardAnalyzer 并不能很好地支持中文分词,所以采用了开源的 IKAnalyer[12] 分词器,并根据数据处理链模块的词典库对 IKAnalyzer 进行扩展。2) 价格 Field 问题,由于 Field 是数字类型,往往搜索的时候需要涉及到区间等操作,所以需要特别的 NumeriField 进行处理。3) 由于其他域并不会占多大空间,虽然在 Content 里面已经具有这些基本内容,但是仍然增加了这些域,方便后期查询中向用户展示。4) 索引目录问题,需要判断索引目录是否存在,如果已经存在,则需要考虑是采取删除原有索引目录的策略还是增量创建索引的策略。以下算法 2 是 Reduce 程序设计的部分伪代码:

算法 2　归并 Map 程序输出的电子产品对象,并建立索引。

输入　Texe key:一致标识;Text values:同一标识的电子产品对象列表。将索引写入 Document。

```
Procedure void reduce( Text key, Iterable < Text > values, Context
  context)
  //获取相应的 Job 的配置文件
  Configuration conf = context. getConfiguration( );
  //获取相应 Reduce 的索引路径
  String indexPath = conf. get( "index_name" + key);
  如果索引已经存在,根据策略选择更新还是重建索引
  //获取相应的写索引工具类,包括 IKAnalyzer、路径等设置
  IndexWriter writer = conf. get( "index_writer" + key);
  //遍历所有的商业文本信息
  For( Text value in values)
  //构造多个域的 Document 对象,包括 Field、NumericField 添加
  Document doc = new Document( value);
  doc. addField( );
  //把 Document 对象构建成索引
  writer. append( doc);
End of Procedure
```

3　电子产品信息抽取方法

HTML 文件由控制部分和正文部分组成。其中 HTML 正文部分即是展示给用户的文本数据内容。HTML 控制标记对大小写不敏感,同时控制标记用“ < ”开始,以“ > ”结束,由 HTML 元素及其属性两部分组成。HTML 控制标记用来控制 HTML 正文部分的展示方式,包括颜色、是否粗体、段落、表格、列表等。据统计,在面向电子产品的商业网站上,大部分的主体商品信息均以表格标签 table、有序列表 ol,无序列表 ul 展示。

以下为几个大型网站上抽样调查的部分网页的主体商品信息展现形式:国内大型交易网站淘宝,主体商品信息列表以 ol 展示;360buy 京东商城,主体商品信息列表以 ul 展示;太平洋电脑网,主体商品信息列表以 ul 展示;亚马逊卓越网,主体商品信息列表以 ul 展示。

如图 5 所示,为了应用于电子产品信息之中,本文信息抽取模块中信息处理数据流主要包括 HTML、XML 两种数据格式技术,数据流转主要包括以下几个步骤:1)根据 URL 地址,获取原始 HTML 网页内容;2)针对 HTMl 网页的内容以及结果特点,进行数据清洗,并转换成 XML 格式的数据内容;3)根据文档对象模型(Document Object Model, DOM)模型,加载 XML 数据内容到内存构建 DOM 树;4)在 DOM 树上,根据 XPath 路径获取相关内容;5)把以上相关内容转换成 XML 文档,保存到文件中。其基本数据流转图如图 5。本文基于 Anchor-Hop 模型[13],即通过一个 Anchor 点间接确定 Hop 点的方式进行定位的思想,实现了图 5 所示的信息抽取方法,称为基于 DOM 的 Anchor-Hop-T 模型。

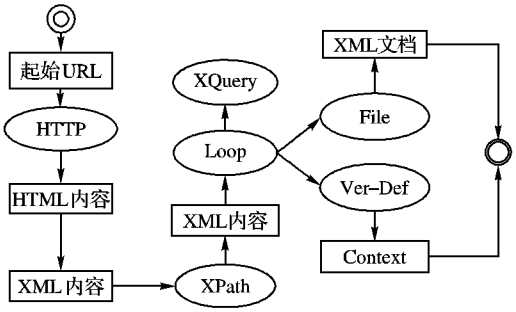

图 5　信息抽取数据流程

在淘宝、太平洋网上分别输入关键字“笔记本”,显示淘宝上有 100 个相关页面,太平洋网显示的页面数量为 165 页。淘宝上每页显示的电子产品数量是 40 个则共有 4000 个待抽取信息。太平洋网上每页显示的电子产品数量为 25 个则共有 4107 个。分别基于 DOM 的 Anchor-Hop-T 模型、基于内容或者属性的 Anchor-Hop 作为实验的类比算法进行类比,比较在召回率、查准率、抽取时间上的表现。

表 3　淘宝网抽取结果

抽取模型	抽取数	正确数	查准率/%	召回率/%	抽取时间/ms
Anchor-Hop	4127	4000	96.92	100	48053
Anchor-Hop-T	4000	4000	100	100	30681

表 4　太平洋网抽取结果

抽取模型	抽取数	正确数	查准率/%	召回率/%	抽取时间/ms
Anchor-Hop	4239	4107	96.88	100	5044
Anchor-Hop-T	4107	4107	100	100	32047

通过表 3 ~ 4 的实验结果发现:在两个模型中,Anchor-Hop 模型的查准率较低,该模型在淘宝网和太平洋网都出现了错误的情况。两个模型的召回率都非常高。两个模型的消耗时间 Anchor-Hop 模型较多。

4　细粒度优化调整及性能测试

Lucene_Hadoop 架构存在粗粒度检索的问题,在面向电子产品领域中,本文进一步提出了实用的改进设计及部分算法设计,完成了细粒度的精确检索。

4.1　InputSplit 数据块大小控制

InputSplit 数据块大小控制是指 Map-Reduce 分布式编程模型中 Map 程序处理数据块的大小的控制。在待索引的电子产品数据集合一定的情况下,如果 InputSplit 数据块较大,则可以产生较少的 Map Task 任务数,否则,会产生较多的 Map Task 任务数。Map Task 任务数较少,则不能很好地利用分布式基础架构 Hadoop 集群中服务器的计算能力,降低建索时间;而任务数过多,则会导致 Job 的过度调度,调度时间占整个建索时间的比重增大,时间并没有有效地被用于建索。分布式基础架构 Hadoop 通过提供重写方法 getSplit 控制 InputSplit 数据块的大小。算法如下所示。

算法 3　根据每个 InputSplit 需要处理的电子产品对象个数来分割 Job 的输入路径的文件。

输入　JobContext job:Job 的控制配置。

输出　List < InputSplit > 目标 InputSplit 列表。

```
Procedure List < InputSplit > getSplits( JobContext job)
  根据 job 控制配置获取待索引的输入文件 File;
  根据 job 控制配置获取每个 InputSplit 需要处理的电子产品对
```

象个数 Count;
初始化一个 List < InputSplit > list;
For(< Key, Value > value in File)) 当单位数据
初始化空 InputSplit, 如果 value 个数小于 Count, 则加入 InputSplit;
如果 value 个数等于 Count, 把当期 InputSplit 存入 list 中;
遍历结束, 直接把剩余的 InputSplit 存入 list 中;
返回目标 InputSplit 列表 list;
End of Procedure

4.2 实验结果与分析

针对 4.1 节,实验是以 2 GB 的数据分别以32 MB、64 MB、128 MB、256 MB、512 MB、1 GB、2 GB 的 InputSplit 大小进行划分,验证了 InputSplit 的大小对建立索引的影响。

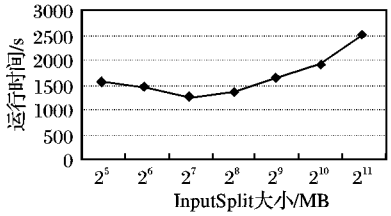

图 6 不同 InputSplit 的索引建立时间

根据图 6 所示的实验结果,可得如下结论:

1)索引建立在 2 GB 的 InputSplit 的性能最低,而且低于单机版的索引建立时间。这是因为当 InputSplit 设置为数据大小一样,则导致所有的计算将会仅被一个 Map 处理,也就是浪费了另外两台服务器的计算能力,而且由于数据的传输导致性能很低。

2)索引建立在 128 MB 的 InputSplit 的性能最高。由于 2 GB的数据划分 128 MB 为单元的 InputSplit,可以划分出 16 个单元,64 MB 的为 32 个单元,256 MB 的为 8 个单元。其中 32 个单元过多地大于 Slave 个数,8 个单元则小于 Slave 的个数,前者增加调度时间,后者浪费 Slave 计算能力,而 16 个单元则处于中间,性能最好。所以 InputSplit 的划分块数不易过多,也不能过少。

3)在 32 MB 到 128 MB 的划分过程中,其单元渐渐减小,但是仍然大于 Slave 个数,所以调度时间开始减少,性能增加。而 128 MB 到 2 GB 的划分过程,其单元数量减少,但是小于 Slave 个数,浪费的 Slave 计算能力增加,性能降低。

4.3 Map Slot 与 Reduce Slot 个数配置

合理地选择 TaskTracker 中 Task 数的大小能显著地改善索引建立的性能。增加 Task 的个数会增加系统框架的开销,但同时也会增强负载均衡并降低任务失败的开销。一个极端是 1 个 Map Slot、1 个 Reduce Slot 的情况,代表一个服务器上最多只有一个 Map 和一个 Reduce 运行,因此无法并行执行任务。另一个极端是 1 万个 Map Slot、1 万个 Reduce Slot 的情况,这会由于调度过度导致框架的开销过大而使得系统资源耗尽。一个 Tasktracker 最多可以同时运行的 Map Task 最大数目是由参数 mapred. tasktracker. map. tasks. maximum 决定的,而可以同时运行的 Reduce Task 最大数目由参数 mapred. tasktracker. reduce. tasks. maximum 决定。两个参数的设置跟计算机硬件的水平有着直接的关系,硬件计算能力强大的时候,可以设置得大一些,更好地提供并行执行能力,否则,参数设置得较小一点。一般情况下 Map Task 的参数可以设置得比 Reduce 的参数大。

4.4 实验结果与分析

针对 4.3 节,实验是以 2 个 Reduce Slot 不变,分别以 8 个、4 个、2 个、1 个 Map Slot 为设置检测索引建立时间,验证 Slot 的配置可以影响到索引建立的性能,且 Slot 个数要根据程序运行动态调整。

如图 7 所示的实验结果,可得如下结论:

1)索引建立在 128 MB、256 MB 以及 512 MB 的时候,都是 1 个 Map Slot 性能最高。由于数据块都不是很大,且 Slave 个数较多,所以 1 个 Map Slot 足够使用,所以过多的 Slot 影响到调度,稍微降低了性能。

2)索引建立在 1 GB 的时候,2 个 Map Slot 性能最高;在 2 GB的时候,4 个 Map Slot 性能最高。由于数据块较大,分到的 Map 程序较多,分别以 2 个、4 个 Slot 时较合适,所以此时性能最高。

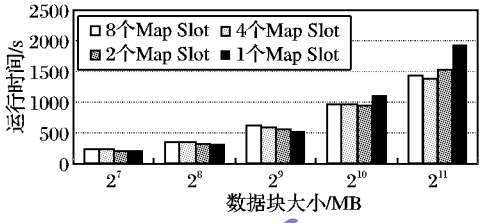

图 7 不同的 Map Slot 索引建立性能比较

5 结语

本文在 Hadoop_Lucene 基础上,提出了一种适用于 Web 电子产品信息的分布式检索系统。通过 Lucene 检索技术访问这些分布式文件,从而提高了检索效率。同时针对 Lucene_Hadoop 架构存在粗粒度检索的问题,提出了细粒度检索改进方法。下一步将研究如下两个方面的问题:1)随着手机在通信手段的重要性的增加,需要提供手机界面的信息检索支持以及条形码扫描的一系列相关功能。2)观点挖掘也是新兴的研究热点,需要在原有 Web 数据挖掘经典方法的基础上,结合自然语言处理与人工智能算法进行系统性能研究。

参考文献:

[1] YONG H K, CHOI I S. An efficient Web information extracting system[C]// Proceedings of the IEEE ISIE 2001. Pusan, Korea: IEEE ISIE, 2001: 1771 - 1774.

[2] 吴广君, 王树鹏, 陈明, 等. 海量结构化数据存储检索系统[J]. 计算机研究与发展, 2012, 49(S1): 1 - 5.

[3] 雷军环, 张光会. 一种基于内容及相似搜索的对等音乐文件共享系统[J]. 计算机应用研究, 2012, 29(4): 28 - 30.

[4] 傅巍玮, 李仁发, 刘钰峰, 等. 基于 Solr 的分布式实时搜索模型研究与实现[J]. 电信科学, 2011, 27(11): 51 - 56.

[5] 孙峥, 孙瑞志, 王剑秦. 网格环境下基于本体的信息检索体系研究[J]. 计算机工程与设计, 2009, 30(23): 5392 - 5399.

[6] Apache. SOLR[EB/OL]. [2012 - 10 - 19]. http://lucene.apache.org/solr/.

[7] 王学松. Lucene + Nutch 搜索引擎开发[M]. 北京: 人民邮电出版社, 2008.

[8] JEFFERY D, SANJAY G. Map/Reduce: simplified data processing on large clusters[C]//Proceedings of the IEEE Sixth Symposium on Operating System Design and Implementation. San Francisco, CA, USA: IEEE OSDI: 2004: 107 - 113.

[9] 苏伟兵. 个性化 Web 商务信息融合关键技术研究[D]. 杭州: 浙江大学, 2010.

[10] 黄晓云. 基于 HDFS 的云存储服务系统研究[D]. 大连: 大连海事大学, 2010.

[11] Apache. Lucene [EB/OL]. [2012 - 10 - 19]. http://lucene.apache.org/.

[12] OsChina. IKAnalyzer[EB/OL]. [2012 - 10 - 19]. http://www.oschina.net/p/ikanalyzer.

[13] 苏伟兵. 个性化 Web 商务信息融合关键技术研究[D]. 杭州: 浙江大学, 2010.