MapReduce 在分布式搜索引擎中的应用①

吴文忠[1]，易 平[2]

[1](广东金融学院，广州 510521)
[2](凯业必达信息技术(上海)有限公司，上海 200120)

摘 要：MapReduce 是一种分布式的并行编程模式，它可以实现大型数据集的并行运算。Lucene 是 Apache 下的搜索引擎开发包，当索引文件不断增大时，Lucene 搜索便会出现瓶颈问题。通过利用 MapReduce 的思想，按城市划分策略将大量并发的搜索请求映射到对应的分布式服务器中进行 Map 操作，再结合 Lucene，从对应索引服务器中查询后利用 Reduce 操作返回最终结果。实验结果表明，这不仅解决了大数据量查询的瓶颈问题，还将系统效率提高了 66.7%。

关键词：MapReduce；Lucene；分布式搜索

Application of Distributed Search Engine Based on MapReduce

WU Wen-Zhong[1], YI Ping[2]

[1](Guangdong University of Finance, Guangdong Guangzhou, 510521 China)
[2](Careerbuilder Information Technology (Shanghai) Ltd Shanghai, 200120 China)

Abstract: MapReduce is a distributed parallelized programming model. It can implement the processing and generating large data sets. Lucene is a Search Engine API under Apache. When the index file growing, the Lucene Search performance is a bottleneck. Based on the MapReduce, this system maps the parallelized search request to the cluster server for Mapping operation. It is mapped by dividing the index file by city strategy. And then the Map Function get the search results with the lucene. The results will be returned to the user by Reduce Function. According to the experimental results, this design does not only resolve the paralleized search bottleneck, but also improves the performance by 66.7%.

Key words: MapReduce; Lucene; distributed search

1 MapReduce编程模式简介

MapReduce 是由 google 提出来的一种编程模型①，并结合文件系统 GFS②能在大规模的普通服务器分布式系统上实现大型数据集的并行运算，处理等。该模型主要由 Map 函数和 Reduce 函数组成，输入数据映射到 Map 函数处理后，将产生一个本地的中间结果 Key/value 对，而 Reduce 函数再通过 Master(主机)的调度，远程调用由 Map 函数产生的中间结果 Key/value 对，作为输入进行相应处理，再将具有相同 key 的结果合并输出最终结果，从而兼顾了分布式中数据处理的独立性及相关性。文献①中对 MapReduce 的机制，实现作了详细的阐述。而 Doug Cutting 用 java 开发的开源项目 Hadoop 也实现了 MapReduce 机制③，以及 HDFS 分布式文件系统④，促使目前国内外众多企业有机会利用 MapReduce 实现大型分布式数据处理。

作为 Doug Cutting 的另一开源项目 Lucene，提供了一组丰富的 API 供开发者实现全文检索。使用 Lucene 能实现索引文件的建立，优化到查询。但当索引文件达到增加到一定数量时，由于 Lucene 的内部机制导致大量结果数据被载入了内存，查询完后才会被丢弃。大量数据占据内存后，将会使 JVM 的频繁垃圾回收，导致系统查询性能出现严重的瓶颈。本文将利用 MapReduce 的编程模式思想结合 Lucene 查询，解

① 收稿时间:2011-03-17;收到修改稿时间:2011-04-26

决这一瓶颈问题。

2　分布式搜索系统模型

由于大容量的索引文件造成了查询的性能瓶颈，因此首先必须按照一定的策略将大索引文件划分为小索引文件分别存储于不同的服务器中。如本文以生活服务搜索为例，由于生活服务搜索有着很强的地域性，所有搜索都与城市地点有关，如：火车票，旅游，酒店等。因此本文采取按城市划分索引的策略，将原本的单个大容量索引文件，按照不同的城市划分出多个小索引文件。在处理索引文件时，先为每个城市分配一个城市代码，作为后续 MapReduce 处理时的 Key，供服务器集群进行分布式的处理。如图 1 所示，为 MapReduce 的分布式搜索引擎系统模型。

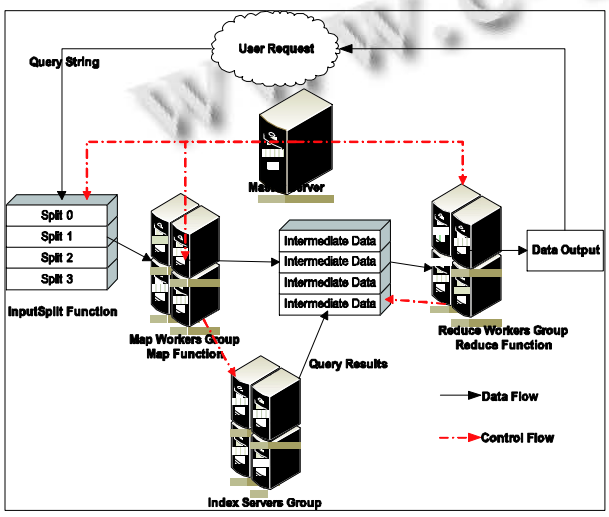

图 1　基于 MapReduce 模式的分布式搜索引擎系统模型

当大量用户同时输入搜索请求后，中央主机(Master)，触发 InputSplit 将输入数据按不同城市代码划分成若干份，以 Hashtable 的数据类型输出，再将 Hashtable 数据组按照不同的 Key 输入到相对应的工作服务器组(workers)，即 Map Function。Map Function 将请求对应的索引服务器组(Index Server)。再将返回的查询结果同样为 Hashtable 形式的中间结果集(Intermediate Data)存储为本地文件。 中央主机调度 Reduce Function 远程调用存储于 Map Workers 内的中间结果集，并按相同的 Key 将数据整合，并最终输出最终结果返回给用户。

其中三个主要的功能模块的数据类型分别为：

InputSplit(String) -> Hashtalbe(K,V)

MapFunction(K,V) -> List(Hashtalbe(K,V))

ReduceFunction(K,V) -> List(V)

3　MapReduce关键技术实现

3.1 索引文件的建立

本文利用开源的全文检索开发包 Lucene⑤实现索引的建立及查询。在创建索引时的除了保持基本的结构化的数据以外还要加入每条记录的所在城市信息代码。当利用网络爬虫⑥从互联网抓取了大量的数据后，从中提取出有效数据，由 Lucene.analysis 分词，并最终由 lucene.index 建立索引文件。具体程序实现如下（数据结构的设计）：

IndexWriter writer=new IndexWriter ("IndexPath",new StandardAnalyzer(), true);//使用 lucene.index 及 lucene.analysis 初始化写索引操作,指定存储索引目录,指定分词器.

for (int i = 0; i < title.length; i++) {

Document doc = new Document();//索引文档初始化

Field f1=new Field(fields[0],title[i],Field.Store. YES, Field.Index.TOKENIZED);//索引文档字段初始化，并决定是否分词

Field f2=new Field(fields[1],infoDesc[i],Field.Store.YES,Field.Index.TOKENIZED);

doc.add(f1);//将字段加入索引文档中

doc.add(f2);

writer.addDocument(doc);//建立索引文件

随着索引文件的不断增大到一定程度时，利用 Lucene.search 搜索时，搜索效率出现严重的瓶颈。因此只有按照不同策略部署在分布式的系统上，才能实现高效率的搜索引擎。本文结合生活搜索中很强的地域性，将数据按照不同的城市代码分别建立索引，从而将大数据量的索引文件划分，分布在不同的索引服务器中以降低单服务器的负载。目前已建立了 300 多个城市的分索引，总数据量达 600 万。

3.2 MapReduce 关键功能模块设计

本文根据上述基于 MapReduce 思想的分布式搜索系统模型，用 Java 将该系统实现。图 2 是从该系统程序中导出的 UML 框图。

由 UML 框图可知，MapReduce 的编程模型中最重要的三个功能模块分别是 InputSplit 函数，Map 函数以及 Reduce 函数由三个相对应的接口实现。

(1) InputSplit 函数：当用户请求搜索时，该请求信息将插入到用户请求队列。InputSplit 根据该队列，将不同的城市代码作为 Key，再将用户输入的请求列表作为 Value 存入 Hashtable 中。这样就完成了对大量并发请求数据的分离，其输出数据表现形式为 Hashtable<Key:AeraId,Value:ArrayList<UserQuery>>。

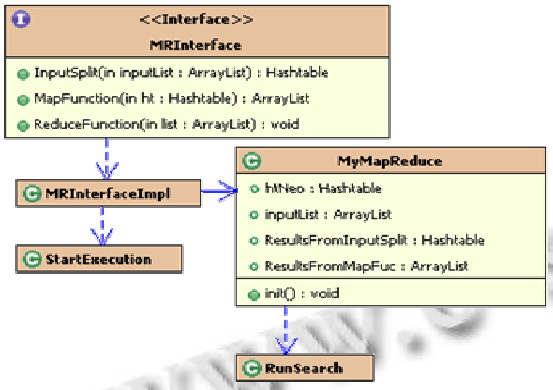

图 2 MapReduce 分布式搜索系统实现

(2) Map 函数：从 InputSplit 中得到分离的 Hashtable 后，按照不同的 Key，将请求映射到相对应的 Map 服务器进行并行运算，调用 Lucene 提供的查询接口，从对应城市的索引服务器中查询，并将返回的结果存储于 Map 服务器的本地存储器中。同时输出以每个用户请求的 SessionID 作为 Key 的 Hashtable 队列中，并提交查询结果文件的位置信息给中央主机(Master)，其输出的数据表现形式为 ArrayList<Key:SessionID,Value:Search MediateResults>。

(3) Reduce 函数：当 Reduce 从中央主机(Master)中获得存储于 Map 服务器中的查询结果的位置信息后，将远程调用该结果文件，并将这些属于同一个 Key 的结果合并，插入到系统响应队列返回给用户，其输出数据表现形式为 ArrayList(SearchResults)。

MapReduce 对搜索请求的处理关键代码如下：

```
HttpRequestList httpRequestList=new Http RequestList();
inputList=(ArrayList) httpRequestList.LoadRequestList();//载入用户请求列表
InputSplitImpl inputSplit= new InputSplitImpl();//初始化 InputSplit
System.out.println("**STEP 1 START**-> Split the Request List to sort them**");
ResultsFromInputSplit=inputSplit.InputSplit(inputList);//对请求数据进行 Split 处理
System.out.println("************STEP1COMPLETE*************");
System.out.println("**STEP 2 START**->Running **Map Function** concurrently for the different cities");
MapFunctionImpl map=new MapFunctionImpl();//初始化 Map
ResultsFromMapFuc=map.MapFunction(ResultsFromInputSplit);//对数据进行 Map 处理
System.out.println("************STEP2COMPLETE*************");
System.out.println("**STEP 3 START**->Running **Reduce Function** for collating Intermediate Results and Printing Results");
ReduceFunctionImpl reduce=newReduce FunctionImpl();//初始化 Reduce
reduce.ReduceFunction(ResultsFromMapFuc);// 通过 SeesionID 对 Map 处理后的结果进行 Reduce 处理
System.out.println("************STEP3COMPLETE*************");
```

4 分布式仿真实验结果

本实验模拟同时并发用户请求，对总共约 10G 的数据进行分布式搜索仿真实验。并与单个服务器响应请求作对比，其实验结果数据如图 3。

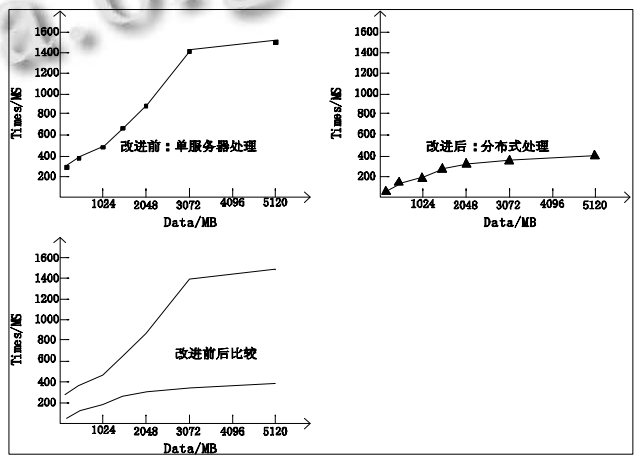

图 3 两种搜索请求处理效率比较

如图 3 可以看出，当单服务器处理搜索请求时，

（下转第 224 页）

为 RSSI 测距中一个最重要的环节就是输入参数随外部环境的变化而变化，不容易在一个点固定一个值。这也能说明误差产生的必然性和随机性。

第二次修正后 RSSI 测距如图实线所示，与第一次修正后的仿真图对比，其误差补偿效果进一步得到提高，精度进一步提高。最小二乘算法的特点说明对随机因素产生的误差具有明显的跟踪校正效果。

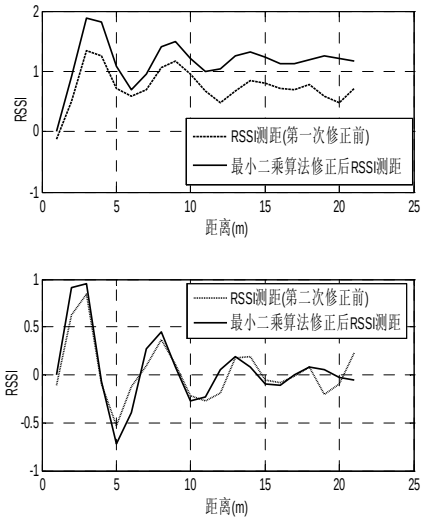

图 3　第一、二次环境参数修正前后 RSSI 测距误差对比仿真

5　结论

本文提出一种针对具体应用的无线传感器网络环境参数修正方案，通过信标节点间的相互协作，采用最小二乘拟合方法修正模型参数，取得了良好的定位性能，对环境的变化具有一定的自适应性，适用于复杂环境的传感器网络。实验结果表明，其修正后的方案明显提高了节点距离估计的精度，此修正方案可用于不同环境下基于 RSSI 定位应用中。

参考文献

1 Li D, Wong KD, Hu YH, et a1. Detection, classification, and tracking of targets. IEEE Signal Processing Magazine, 2002, 19(2):17–29.

2 陈维克,李文锋,首珩,等.基于 RSSI 的无线传感器网络加权质心定位算法.武汉理工大学学报,2006,30(2):265–268.

3 任维政,徐连明,邓中亮,等.基于 RSSI 的测距差分修正定位算法.传感技术学报,2008,21(7):1247–1250.

4 孙利民,李建中,陈渝,等.无线传感器网络.北京:清华大学出版社,2005.138.

5 金卫民,神显豪.基于 RSSI 的室外无线传感器网络自定位算法.计算机工程,2008,34(13):89–90.

6 王福豹,史龙,任丰原.无线传感器网络中的自身定位系统和算法.软件学报,2005,16(5):1148–1157.

（上接第 251 页）

索引文件在 1G-3.5G 时 Lucene 查询时间成指数增长出现瓶颈，但将海量数据按不同城市划分到分布式的环境中结合 MapReduce 编程模式后，虽然数据量小的时候 MapReduce 的分布式搜索系统优势不是很明显，当随着数据量的不断增大，系统效率提高了约 66.7%。这不但解决了 Lucene 数据量的瓶颈，而且还提高了搜索响应效率。

5　总结

虽然 Lucene 提供了从索引的建立、处理到查询的开发包，但在查询大索引数据上会出现严重的查询效率瓶颈。将单一的大索引数据文件按照不同城市代码划分到不同的分布式索引服务器上，并结合 MapReduce 的分布式编程思想将大量并发搜索请求由分布式系统处理，以解决大量搜索请求并发时的响应效率问题。实验数据表明改进后的搜索系统大大提高了查询效率。

参考文献

1 Dean R, Ghemawat A. Map Reduce: implified data processing on large cluster. SDI, 2004.

2 Ghemawat N, Gobioff H, Leung ST. The Google File system. Operating Systems Principles, 2003:29–43.

3 Doug Cutting. Scalable Computing with MapReduce. OSCON. 2005.

4 Borthankur D. The Hadoop Distributed File System: Architecture and Design. Apache Software Fundation. 2007.

5 Apache.Welcome to Lucene.http://lucene.apache.org.

6 郑力明,易平.基于 HTMLParser 信息提取的网络爬虫设计.微计算机信息,2009,25(5–3):123–125.