

基于树结构的 MapReduce 模型

李远方, 贾时银, 邓世昆, 韩月阳

(云南大学 信息学院, 云南 昆明 650091)

摘 要: MapReduce 是 Google 开发的一种并行分布式计算模型, 已在搜索和处理海量数据领域得到了广泛的应用。此模型只适用于数据关联性弱、能够高度并行化的程序, 未能处理数据关联性强的数据(比如树形结构)。文中详细讨论了 MapReduce 的实现机制, 提出了一种基于树结构的 MapReduce 模型, 它是基于一种聚类聚合的反复轮询过程, 聚合时用 $\langle k_1, k_2, \dots, k_n, \text{value} \rangle$ 代替传统的 $\langle k, \text{value} \rangle$, 使模型更具有一般性。最后搭建 Hadoop 平台来处理 XML 结构的海量数据, 并对比新旧两种模型的效率。实验结果表明, 其执行速度明显比传统模型高效。

关键词: 树结构; MapReduce; XML; Hadoop

中图分类号: TP31

文献标识码: A

文章编号: 1673-629X(2011)08-0149-04

MapReduce Model Based on Tree Structure

LI Yuan-fang, JIA Shi-yin, DENG Shi-kun, HAN Yue-yang

(College of Information, Yunnan University, Kunming 650091, China)

Abstract: MapReduce is a parallel distributed computing model developed by Google, it is widely used in the area of searching and large data dealing. This model can be used to process data with weak correlation degree, but unable to deal with the data efficiently by making full use of the relationship among the data (such as a tree). It proposes a MapReduce model based on the tree structure, it is based on a process which is featured in repeated polling with clustering aggregation, use $\langle k_1, k_2, \dots, k_n, \text{value} \rangle$ rather than $\langle k, \text{value} \rangle$ as usual when aggregation, make the model more general. Experimental results show the execution speed is significantly higher than the traditional model.

Key words: tree structure; MapReduce; XML; Hadoop

0 引言

随着数据的猛增, 如何能在海量数据里快速搜索和处理数据是亟需解决的问题。因此, 设计数据密集和计算密集的高效算法模型就尤为重要。MapReduce 框架^[1]模型就是基于此目的而提出, 使得海量数据的搜索和处理变得简易。MapReduce 是云计算的关键技术, 将要执行的问题拆解成 Map 和 Reduce 的方式来执行, 以达到分散运算的效果。例如要搜寻哪些网页中有「Paper」这个字, 可以先用 Map 程序来计算出所有网页中每一个字的位置, 再使用 Reduce 程序在每一个字的清单中, 筛选出「Paper」这个字所对应的网页网址。现今, MapReduce 模型在统计文本、批量图像处理^[2]、多核 CPU^[3]、关联数据查询、并行科学计算^[4]等

领域得到了广泛应用。在传统模型框架中, 用户指定一个 Map 函数来处理 key/value 对, 并且产生相应的一系列中间 key/value 对, 然后使用 Reduce 函数来合并所有具有相同 key 值的中间键值对^[1], 并将结果作为下一轮的 MapReduce 模型的输入。MapReduce 函数形式实现的程序可以自动分布到一个由普通机器组成的超大集群上并发执行^[1]。系统在运行时间内会解决输入数据的分布细节, 跨越机器集群的程序执行调度, 处理机器的临时瘫痪问题, 并且处理机器之间的通讯请求^[4]。这样的解决方案使得程序员可以不需要有什么并发处理或者分布式程序经验, 就可以处理超大的分布式系统的资源。

然而此种模型存在着缺陷, 对于关联性强的数据表现出无能为力或处理效率低下。文中为了解决这个问题, 提出了一种基于树结构的 MapReduce 模型, 此种模型将有关联的数据形式化地描述为树结构, 并定义必要的函数式关联。在 Reduce 阶段, 用多关键字排序 $\langle k_1, k_2, \dots, k_n, \text{value} \rangle$ 代替传统的单关键字排序 $\langle k, \text{value} \rangle$ 。因此模型充分考虑了数据之间的关联, 所以健壮性更强。

收稿日期: 2011-01-24; 修回日期: 2011-05-07

基金项目: 云南省自然科学基金(2007F174M); 云南大学研究生科研课题资助项目(yunny200928)

作者简介: 李远方(1986-), 男, 四川人, 硕士生, 主要从事云计算网络、分布式计算的研究; 邓世昆, 教授, 主要从事计算机网络、智能建筑方面的研究。

1 M/R 模型实现机制及应用

1.1 MapReduce 实现机制

当用户程序调用 MapReduce 函数, 执行过程如图 1 所示^[1]。

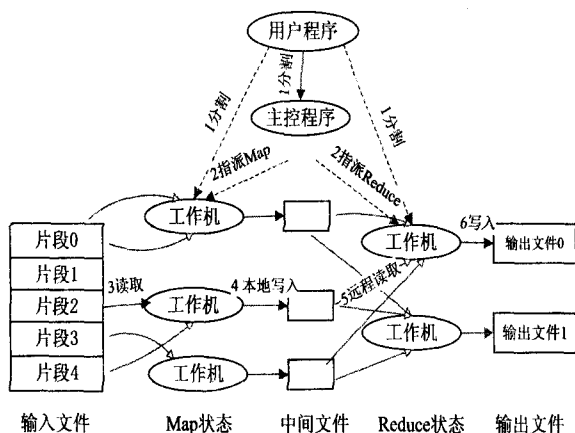


图 1 MapReduce 执行流程图

其全过程简约为两步骤:

① $\text{map}(k1, v1) \rightarrow \text{list}(k2, v2)$;

② $\text{reduce}(k2, \text{list}(v2)) \rightarrow \text{list}(v2)$ 。

以下是详细的执行流程(图中标识与下面标号一致)。

(1) 当用户在使用 MapReduce 函数库时, 首先把输入文件分成 M 块, 即 M 个任务。每块大概在 16 兆到 64 兆之间, 这个数值可由用户需要任意改变, 接着将 M 个任务分摊到各分布节点上运行。

(2) 在这些分派的任务中, 主控程序 Master 负责程序的调度与监控, 是全局的统治者。其它剩下的执行程序都是作为 Master 分派的工作机, 一共有 M 个 Map 任务和 R 个 Reduce 任务需要分派到各计算节点, Master 选择空闲的工作机来分配任务。

(3) 将一个处理任务输入到一个 Map 工作机, 它首先预处理输入的数据, 并且将关键字对传递给用户定义的 Map 函数。Map 函数产生的中间结果(关键字对)暂时缓冲到内存^[5]。

(4) 上一步的中间结果会在某个时间片内将中间结果自动写到本地盘, 这些数据通过 Partition 函数分成 R 个区。并且需要将结果的存储的具体位置定时传达给主机程序 Master, 接着主机 Master 负责把这些存储具体位置信息传送给 Reduce 主机。

(5) 当 Master 通知 Reduce 的工作机关于中间关键字对的位置时, 它调用远程程序从 Map 工作机的本地硬盘上读取缓冲的中间数据。当 Reduce 工作机读到所有计算的中间数据, 它就使用中间关键字进行排序, 相同关键字的信息就会汇总到一起。

(6) Reduce 工作机根据每一个唯一中间关键字来遍历所有的排序后的中间数据, 并且把关键字和相关

的中间结果值集合传递给用户定义的 Reduce 函数, Reduce 函数组合各地的信息最终组合成一个输出文件, 以便整体输出任务结果。

(7) 当所有 Map 和 Reduce 任务完成的时候, Master 主机将调用用户程序, 激活主程序, 调用 Reduce 工作机返回用户程序的调用位置。

以 Google 云计算 MapReduce 上的一个简单例子为例, 在 TB 数量级的文档集中统计每一个单词出现的次数。有如下的伪代码, 其中 Map 函数检查每一个单词, 并且对每一个单词增加 1 到其对应的计数器。Reduce 函数把特定单词的所有出现的次数进行合并。其伪代码如下^[6]:

```
map(String key, String value):
    // key: document name
    // value: document contents for each word w in value:
    EmitIntermediate(w, "1");
reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```

从以上的代码, 可看出 MapReduce 框架在处理并行程序时的优越性。

1.2 MapReduce 简单应用树

将传统 MapReduce 模型不做任何的修改应用树, 选用 XML 数据文件作为树结构的典型代表。其执行流程如图 2 所示: 从树的左下端开始输入数据, 经过分割过程 $//B \rightarrow //C \rightarrow //D \rightarrow //C \rightarrow //B$ ($//B$ 代表以 B 为根节点的子树), 最后执行到树的右下端。

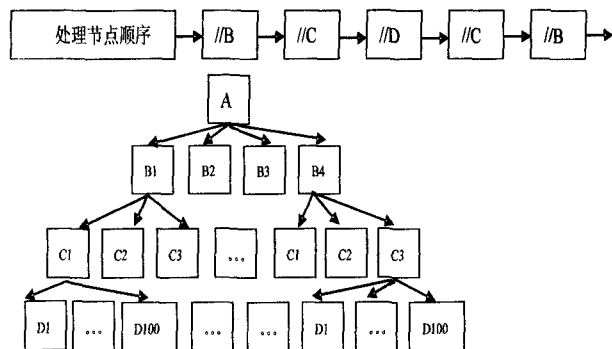


图 2 XML 数据处理流程图

将 MapReduce 传统模型应用于处理 XML 数据文件, 即 $\text{Map}: (k1, v1) \rightarrow [(k2, v2)]$ $\text{Reduce}: (k2, [v2]) \rightarrow [(k3, v3)]$ 。即将某个任务分解若干个小任务进行。此种策略只是将 MapReduce 模型简单地应

用到 XML 数据。其执行流程如图 3 所示,可知并没有真正实现并行。

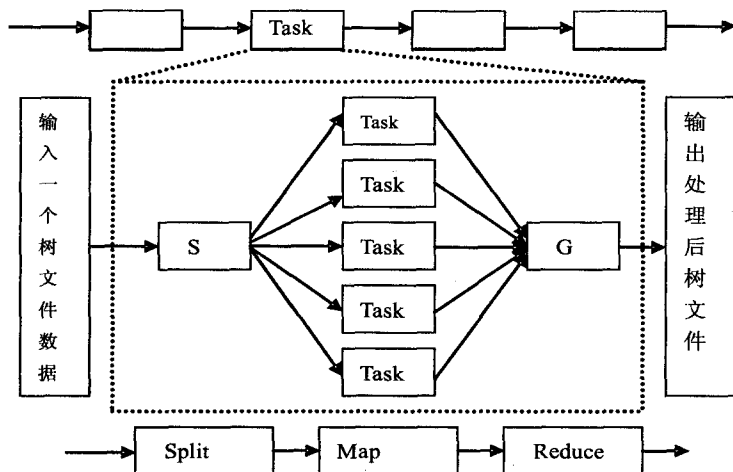


图3 传统 M/R 模型应用于 XML 数据流

2 基于树结构 M/R 模型的建立

2.1 构造树

将传统 MapReduce 模型^[6-8]机械地应用于树结构数据,并未实现并行高效的优点。其关键在于未能将数据之间的关系信息传达给 MapReduce 模型。若能将无联系和有联系的数据抽象成一般树,将相关联信息形式化为 MapReduce 加载函数,模型将达到优化,更具有普遍性。因此先按 MapReduce 编程模式构造树^[5],这是一个反复聚类聚合的过程,其具体过程如下:

(1) M-聚类:两个相互独立的集合一次 Map 将形成多条路径为 1 的集合。

(2) R-聚合:Reduce 形成一个深度为 1 的树(树林)。若 Map 建立在两个相互独立没有交集的集合上时,一次 Reduce 就产生了一个深度为 1 的树。反之,形成森林。若在 Reduce 时,若有相互关联的 value 对应有相同的 key,用 $\langle k_1, k_2, \dots, k_n, value \rangle$ 代替 $\langle k, value \rangle$ 。这样的 MapReduce 更具有普遍性。

(3) 下一轮聚类聚合,直到无法聚类为止。

此构造树抽象过程,将数据关联性达到了高度统一。将关联信息载入并行程序构件,使模型的改进成为了可能。

2.2 MapReduce 模型改进

将所有数据抽象为 CData,即可将二进制数据对象(图像)、简单数据文本(DNA 序列比对^[9,10])都抽象成 XML 数据文件。假设 xml 数据以“T[”为开始标志,以“]T”为结束标志,数据为“#d”。新的模型将是边分割数据边并行任务,并在 Reduce 阶段,用多关键

字排序 $\langle k_1, k_2, \dots, k_n, value \rangle$ 代替传统的单关键字排序 $\langle k, value \rangle$,其并行效率易见。将 2.1 形式化描述,

即处理 XML 数据的过程 T 实际是将 X 结构数据更新成 X' 数据的过程。在 $T = (\sigma, A)$ 中, σ 是以某个节点为根的子树域, A 是处理 XML 结构的一个函数(由用户设置参数决定)。首先将 $\sigma(X)$ 分割成 n 棵非重叠的子树 $s_i (i = 1, 2, 3, \dots)$, 即变为 $\sigma(X) = \{s_1, s_2, \dots, s_n\}$ 。再将每棵子树 s_i 应用 A 变成 $s'_i, s'_i = A(s_i)$ 。最后 $X' = X[s_1 \rightarrow s'_1, s_2 \rightarrow s'_2, \dots]$ 。其具体处理过程如图 4 所示。

将 MapReduce 模型改进后应用 XML 数据流,如图 5 所示。

从图 5 可看出,新模型使 XML 数据

真正得到了并行处理,从而弥补了传统模型的缺陷。

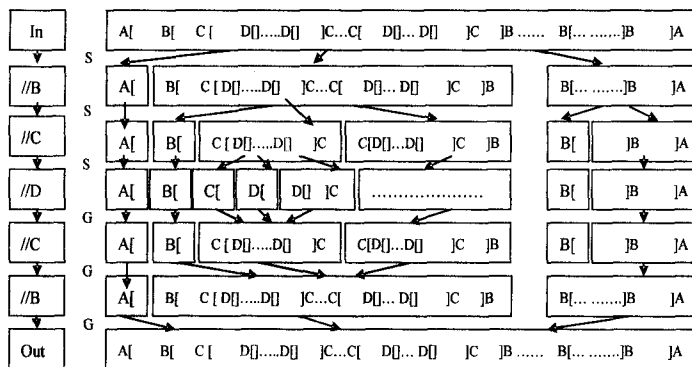


图4 XML 新的处理流程图

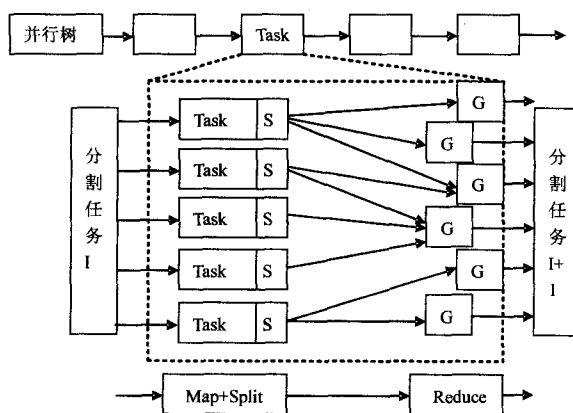


图5 改进模型应用于 XML 数据流

3 实验及结果分析

由于 Hadoop^[11,12] 是一个在集群上运行大型数据库处理应用程序的开放式源代码框架,支持 MapReduce 编程范例来创建并执行的应用程序。为了比较传统模型与改进模型之间的执行效率,特搭建 Hadoop 实验平台,整个实验平台由 6 台普通 PC 机和一个交换机组成,使用的操作系统为 redhat, java 环境为 jdk1.5-

linux 版本, Hadoop 软件版本为 Hadoop-0.21。

实验的输入数据是大批量的 XML 文件, 利用 Hadoop 加载不同的 MapReduce 模型对 XML 文件进行数据提取, 即从这些文件中查找指定的内容。通过对比连续执行、并行传统模型、并行新模型三种方法在上级节点域为 200 的情况下的执行效率, 经统计在上级节点域固定的情况下, 改进后的模型, 树节点域 20 比 10 的效率明显提高了 11% 左右。从图 6 可以看出, 随着问题的规模的增加, 改进后的模型效率得到了提高, 传统模型随着规模的增加其运行时间陡然上升, 而改进后的模型其运行时间趋于稳定。经过计算, 改进后的模型效率比传统模型提高了 13.8%。

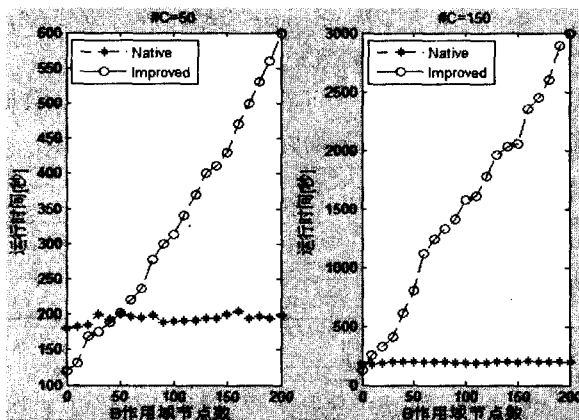


图 6 效率趋势图

4 结束语

文中为使 MapReduce 模型更具有一般性, 特提出一种新的聚类聚合模型, 它将所有数据抽象构造成树, 形式化关联信息载入 MapReduce 模型。另在 Reduce 函数阶段, 利用 $\langle k_1, k_2, \dots, k_n, \text{value} \rangle$ 代替 $\langle k, \text{value} \rangle$ 。通过实验分析发现, 改进模型的效率明显得到提高, 达到了预期的目的。

当然文中也存在很多不足, 新模型的提出, 忽略了 MapReduce 各个任务切换调度时所花费的时间, 通过

实验看到, 该部分的花销应该适当加以考虑, 建立更加完善的模型。另外, 由于实验条件限制, 并没有在高并发、大容量、高压力的环境下考虑模型的稳定性。

参考文献:

- [1] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[C]// Proceedings of the 6th Conference on Symposium on Operating Systems. Design & Implementation. [s.l.]: USENIX Association, 2004.
- [2] Catanzaro B C, Sundaram N, Keutzer K. A Map Reduce Framework for Programming Graphics Processors[C]// Workshop on Software Tools for MultiCore. [s.l.]: [s.n.], 2006.
- [3] Ranger C, Raghuraman R, Penmetsa A, et al. Evaluating MapReduce for Multi-core and Multi-processor Systems[C]// HPCA. [s.l.]: [s.n.], 2007: 13-24.
- [4] 郑启龙, 房明, 汪胜, 等. 基于 MapReduce 模型的并行科学计算[J]. 微电子学与计算机, 2009, 26(8): 21-23.
- [5] Sarje A, Aluru S. A MapReduce Style Framework for Trees[R]. [s.l.]: Department of Electrical and Computer Engineering, 2008: 17-18.
- [6] 胡或, 封俊. Hadoop 下的分布式搜索引擎[J]. 计算机系统应用, 2010, 19(7): 24-26.
- [7] 焦金涛. 基于 PageRank 的 Web 挖掘改进算法[J]. 计算机工程, 2009, 35(15): 31-32.
- [8] 史佩昌, 王怀民. 面向云计算的网络化平台研究与实现[J]. 计算机工程与科学, 2009, 31(11): 12-13.
- [9] 孙广中, 肖峰. MapReduce 模型的调度及容错机制研究[J]. 微电子学与计算机, 2007, 24(9): 37-38.
- [10] 奚建清, 游进国. 基于 MapReduce 的封闭立方体并行计算方法[J]. 华南理工大学学报, 2009, 37(1): 8-9.
- [11] Hadoop. The Apache Software Foundation[EB/OL]. 2010. <http://hadoop.apache.org/core>.
- [12] Bialecki A, Cafarella M, Cutting D, et al. Hadoop: a framework for running applications on large clusters built of commodity hardware[EB/OL]. 2005. Wiki at <http://lucene.apache.org/hadoop>.

(上接第 148 页)

- [3] 刘金岭. 基于语义的高质量中文文本聚类算法[J]. 计算机工程, 2009, 35(10): 201-205.
- [4] 刘金岭. 基于主题的中文短信文本分类研究[J]. 计算机工程, 2010, 36(4): 30-32.
- [5] 刘金岭. 基于降维的短信文本语义分类及主题提取[J]. 计算机工程与应用, 2010, 46(23): 159-161.
- [6] 丘志宏, 官雷光. 利用上下文提高文本聚类效果[J]. 中文信息学报, 2007, 21(6): 109-115.
- [7] Tombros A. Reflecting user information needs through query based summaries[R]. Glasgow: Department of Computing Science, University of Glasgow, 1997.
- [8] 曾雪强, 王明文, 陈素芬. 一种基于潜在语义结构的文本分类模型[J]. 华南理工大学学报(自然科学版), 2004, 32(z1): 99-102.
- [9] Gong Leiguang. Exploring Computational Mechanism for Contexts[J]. IEEE Computational Intelligence Bulletin, 2002, 1(1): 19-25.
- [10] 刘金岭. 基于查询词扩展的中文垃圾检索研究[J]. 计算机工程, 2011, 37(3): 151-154.
- [11] 李荣陆, 王建会, 陈晓云, 等. 使用最大熵模型进行中文文本分类[J]. 计算机研究与发展, 2005, 42(1): 94-101.
- [12] 徐长青. 中文文本分类技术研究[D]. 长春: 吉林大学, 2007.