

基于 MapReduce 的 PageRank 算法的研究

陈 官, 牛秦洲

(桂林理工大学 信息科学与工程学院, 广西 桂林 541004)

摘 要: 在对 PageRank 算法进行研究的基础上, 利用 MapReduce 编程模型思想对 PageRank 算法进行改进, 设计了在云平台 Hadoop 环境下运行的基于 MapReduce 的 PageRank 分布式并行算法, 并在实验中对不同规模的 Web 图数据集进行了测试, 分析不同的 Blocksize 参数对于算法计算性能的作用以及集群节点数目对于算法运行效率的影响。

关键词: 云计算; MapReduce 模型; PageRank 算法; Hadoop; 并行计算

中图分类号: TP338.6

文献标识码: A

文章编号: 1000-7180(2012)05-0081-05

Research on Pagerank Algorithm Based on Mapreduce

CHEN Gong, NIU Qin-zhou

(College of Information Science and Engineering, Guilin University of Technology, Guilin 541004, China)

Abstract: PageRank algorithm is improved by MapReduce programming model based on research of PageRank, so the PageRank distributed parallel algorithm based on MapReduce is designed in the cloud platform Hadoop, which is used to test different web graphs datasets in experiment. At end, the impacts of different blocksize parameters to computing performance and numbers of nodes of the computer cluster to operational efficiency are analyzed.

Key words: cloud computing; MapReduce model; PageRank algorithm; Hadoop; parallel computing

1 引言

PageRank 算法是一种以父页面的重要性来决定其重要性的迭代算法, 它通过分析网络的链接结构来获得网络中的权威网页, 其技术在 Google 搜索引擎中被成功应用. PageRank 算法所处理的页面数据量较大, 采用串行算法效率较低, 算法要经过多次迭代运算, 需要大量的存储资源和计算资源. 随着计算机技术的发展, PageRank 算法通过大型的并行系统实现也成为必然. MapReduce 是 Google 提出的一种基于云计算的软件架构技术, 它采用特殊的并行编程模式, 用于海量数据集(一般大于 1TB)的并行运算^[1-4]. MapReduce 有函数式和矢量编程语言的共性, 使得这种编程模式特别适合于非结构化和结构化的海量数据的搜索、挖掘、分析与机器学习等, 用于分布式处理大规模数据集, 其计算效率极

高.

针对 PageRank 算法计算量较大的特点, 本文在 Apache 的分布式计算开源框架 Hadoop 中将 PageRank 算法和 MapReduce 相结合, 以提升 PageRank 算法的运算效率, 并对新算法的运行效率、时间开销等加以研究.

2 PageRank 与分布式并行计算

2.1 PageRank

PageRank 算法^[5-6]是由 Lawrence Page 和 Sergey Brin 提出的基于链接分析的网页排序算法, 该算法建立在随机冲浪者模型上. 其基本思想是: 网页的重要性排序是由网页间的链接关系所决定的, 算法是依靠网页间的链接结构来评价每个页面的等级和重要性, 一个网页的 PageRank 值(简称为 PR 值)不仅只考虑指向它的链接网页数, 还有指向其他

收稿日期: 2011-09-05; 修回日期: 2011-10-08

基金项目: 广西科学研究与技术开发计划基金项目(0992006-1); 广西教育厅科研立项项目(201106LX226)

网页的网页本身也非常重要. 如果将网页看作是 Web 的信息实体, 而网页间的超链接反应了实体之间的关系, 那么整个 Web 可以看作是一个有向图, 称为 Web 图. Web 图中实体的 PR 值越大, 表明其越重要. PageRank 算法简单描述如下: u 是被研究网页, v_i 是链接 u 的网页, $C(v_i)$ 是网页 v_i 向外链接的网页个数, d 是衰减因子 ($0 < d < 1$, 通常取 0.85). 因此, 网页 u 的 PR 值为:

$$PR(u) = (1-d) + d \sum_{i=1}^n \frac{PR(v_i)}{C(v_i)} \quad (1)$$

利用公式(1)可以计算 u 和 v_i 的 PR 值, 但一开始并不知道每个网页的 PR 值, 可以先假设整个 Web 图是由 u 和 v_i ($1 \leq i \leq n$) 构成的, 每个网页初始 PR 值为 $1/(n+1)$, 然后代入公式(1)算出 $PR_1(u)$ 和 $PR_1(v_i)$, 再代入公式(1)进入下一次迭代运算, 直到 $|PR_k(u) - PR_{k-1}(u)|$ 收敛于一个很小的数 ϵ , 返回 $PR_k(u)$ 值就是网页 u 的 PR 值. PageRank 算法伪代码如下:

```
PageRank(G){ //参数 G 为 Web 链接图
    k=0; PR0(u)=1/(n+1);
    for(i=1 to n) PR0(vi)=1/(n+1);
    Repeat
        k=k+1;
        PRk(u) = (1-d) + d ∑i=1n PRk-1(vi) / C(vi);
    Util |PRk(u) - PRk-1(u)| < ε;
    Return PRk(u); }
```

2.2 Hadoop

Hadoop^[7-8] 是 Apache 的一个使用 Java 的支持开发和并行处理大规模数据的分布式计算开源框架, 主要由分布式文件系统 (HDFS) 和 MapReduce 模型组成. Hadoop 为应用程序提供了一组稳定/可靠的接口和数据运动. 在 Hadoop 中实现了 MapReduce 算法, 它能够将应用程序分割成许多很小的工作单元, 每个单元可以在任何集群节点上执行或重复执行. 此外, Hadoop 所提供的 HDFS 用来在各个计算节点上存储数据, 并提供了对数据读写的高吞吐率.

2.3 HDFS

Hadoop 中的分布式文件系统 HDFS 是一种主从 (Master/Slave) 结构的体系. 一个 HDFS 集群是由一个名字节点 (NameNode) 和一组数据节点 (DataNode) 所组成. 名字节点是一个管理文件的命名空间和调节客户端访问文件的主服务器. 数据节点

负责管理它所在节点上的数据存储, 一般是一个节点上运行一个数据节点. HDFS 将文件分割成一个或多个块, 这些块存储在一组数据节点中. 名字节点负责文件命名空间的文件或目录操作, 例如打开、关闭、重命名等, 同时它还确定块与数据节点的映射. 数据节点负责来自文件系统客户的读写请求, 同时还要执行块的创建、删除以及来自名字节点的块复制指示.

2.4 MapReduce

Hadoop MapReduce 是在 Google MapReduce 的基础上实现的并行编程模型, 也是一个处理和生成超大数据集的算法模型的相关实现. MapReduce 先将输入的数据集划分为若干组数据, 由用户自定义的 Map 函数接受一组数据并将其转换为一个 $\langle \text{key}, \text{value} \rangle$ 键/值对列表, 输入域中的每个元素对应一个 $\langle \text{key}, \text{value} \rangle$ 键/值对. 框架将 Map 函数的输出进行排序处理, 再将结果输入给 Reduce 函数. Reduce 函数接收一个 key 及其相应的 value 集合, 对这些 value 进行汇总处理, 返回输入数据的 value 列表的总和. Hadoop MapReduce 在处理大数据集时, 先将其分解成若干组数据, 每组数据分别在集群中的某个节点由 Map 函数进行处理并生成键/值对, 然后这些键/值对又由大量的节点进行 Reduce 函数处理合并, 从而形成最终结果值. Hadoop MapReduce 的执行流程如图 1 所示.

Hadoop MapReduce 框架由一个 Master 服务 JobTracker 和多个运行于多个节点的 Slave 服务 TaskTracker 所组成. Master 负责调度 Job 的每一个子任务 Task 运行于 Slave 上, 并且监控它们, 如果发现失败的 Task 就重新运行它, 而这些任务分布在不同的 Slave 上. Slave 则负责直接执行每一个 Task. TaskTracker 都需要运行在 HDFS 的 DataNode 上, 而 JobTracker 则不需要, 一般情况应该把 JobTracker 部署在单独的机器上.

3 基于 MapReduce 的 PageRank 算法

3.1 基本思想

对于 PageRank 算法, MapReduce 的处理过程可以构思如下: Hadoop 中的 Master 节点先将 HDFS 中存储网页链接关系的文件划分为多个文件块^[9], 然后再交给每个 Slave 节点, 而每个 Slave 节点上可能会进行 Map 函数处理, 也可能会进行 Reduce 函数处理. 先将划分好的网页链接关系文件的某一个文件块交给运行 Map 函数的节点, 并按照一

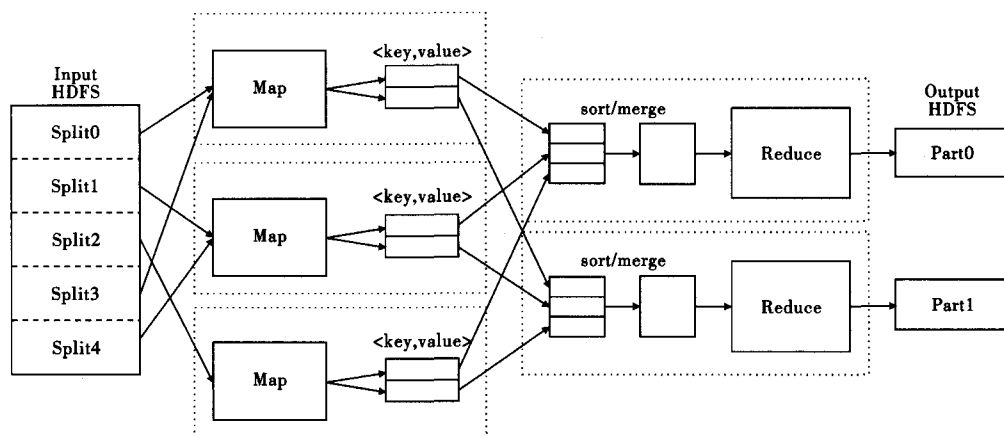


图1 Hadoop MapReduce 执行流程

定格式的<key,value>键/值对输出至 Master 节点,然后再由运行 Reduce 函数的节点读取 Map 函数输出的中间结果,进行汇总处理,形成最终结果,最后保存到 HDFS 中.在 Hadoop MapReduce 计算模式中,并行化 PageRank 算法的每次迭代都是一个 MapReduce 的计算过程,如何设计该过程也是 PageRank 并行算法的核心.

3.2 数据准备

为了实现方便,先将网页链接关系文件转换成网页链接结构文件,如表 1 所示.其中每个节点初始 PR 值为 $1-d$, d 为 0.85.

表1 网页链接结构

起始节点	初始 PageRank 值	目标节点列表
1	$1-d$	3 4 5
2	$1-d$	4 5
3	$1-d$	2
4	$1-d$	1 2
5	$1-d$	2 3

因为网页链接关系文件中包含了大量的网页节点数据,所以由网页链接关系文件转换成网页链接结构表的过程也采用并行 MapReduce 计算来处理. Map 函数生成由起始节点到目标点的<key,value>键值对,其中 key 为起始节点,value 为目标节点.将 Map 函数执行后生成中间结果<key,List(value)>作为 Reduce 函数的输入,然后,Reduce 函数把每个 key 的 value 列表输出,并为每个 key 初始化 PR 值,再输出文件到 HDFS,并作为并行 PageRank 迭代计算的输入.算法描述如下:

```
Map(key,value) {
    //key 存放起始节点,value 存放目标节点
    //从网页链接关系文件中读出每条记录
```

```
for each 目标点 in 记录 {
    起始节点->key;
    目标节点->value;
    output(key,value);
}
Reduce(key,list(value)) {
    //key 存放起始节点,list(value)为目标节点列表
    for each value in list {
        value add to sequence;
    }
    1-d-> initial_pagerank;
    (initial_pagerank,sequence)->values;
    output(key,values);
}
```

3.3 算法设计

基于 Hadoop MapReduce 的 PageRank 算法思想描述如下:

(1) PageRank 迭代阶段的 Map 函数将网页链接结构表中每一行记录里的目标节点序列中各个目标节点输出为<key(目标节点),value(起始节点 PK 值/目标节点数量)>.

(2) MapReduce 框架收集 Map 函数输出中间结果,包括 key 及其相应的 value 列表.在 Reduce 函数处理时,对于每个 key 由公式(1)计算出每个页面新的 PK 值,并将结果保存在 HDFS 中,以用于下一次迭代.

(3) 把上面生成的结果与网页链接结构表进行合并处理,并作为下一次迭代的输入,然后迭代执行 PageRank 并行算法,并统计算法收敛性,若达到收敛条件则算法结束,输出 PR 值,否则继续迭代过程.

基于 Hadoop MapReduce 的 PageRank 算法伪代码描述如下:

```
Map(key,value) {
```

```

//key 为网页  $x$ 
//value 为  $(PR(x), links[y_1, y_2, \dots, y_n])$ 
//links 为出链链接页面
output( $x, links[y_1, y_2, \dots, y_n]$ ); //用于下次迭代
for each  $y_i$  in links {
     $Sx = PR(x)/n$ ;
    output( $y_i, Sx$ );
}
Reduce(key, value) {
    //key 为网页  $y$ 
    //value 为 list of  $[Sx]$ 和  $links[y_1, y_2, \dots, y_n]$ 
     $Px = 0$ ;
    for each  $Sx$  in list {
        if ( $Sx$  为  $x$  的 pagerank) {
             $Px += Sx$ ;
        }
        else { //用于下次迭代
            output(value);
        }
         $Px = d * Px + (1-d)$ ;
        output( $x, Px, links$ );
    }
}

```

如果每一个迭代过程分为两个 MapReduce 阶段,在第一个阶段中生成 PageRank 向量,而第二阶段将生成的 PageRank 向量与第一阶段的输入数据合并,并用于下一次迭代,这样就会需要两次访问 HDFS. 所以,可将两个阶段合并为一个阶段,在 Map 中输出 key 及其对应的 links,在 Reduce 中将该 key 新的 PageRank 与 links 一起用于下次迭代.

4 实验

4.1 实验平台

采用 3 台 Intel pentium4 2.4 GHz, 512 MB 内存 PC 机通过 10M 交换机相连搭建云环境,其中一台 PC 机作为 Master 节点,其它 PC 机为 Slave 节点. 每台 PC 机都安装 Linux Red Hat 8.0 操作系统、hadoop0.18.3、JDK1.6 和 OpenSSH 协议,开发环境采用 MyEclipse8.5 + IBM MapReduce eclipse plug-in,将 Master 配置作为 NameNode 和 JobTracker,将 Slave 配置作为 DataNode 和 TaskTracker. 实验数据集采用美国斯坦福大学社会网络研究平台提供的 Web 图数据,如表 2 所示.

表 2 实验数据集

名称	大小	节点数	边数	描述
web-Stanford	31M	28 万	231 万	Web graph of Stanford. edu
web-BerkStan	71M	90 万	511 万	Web graph of Berkeley and Stanford
web-Yahoo	105M	85 万	760 万	Web graph from Yahoo
web-Google	534M	140 万	3.7 千万	Web graph from Google

4.2 实验结果分析

(1) MapReduce 框架的并行规模对作业执行时间的影响

Hadoop 中数据分块的大小 (blocksize) 对并行运算影响很大. Map 任务的数目是由输入数据所切分的块数决定的,所以输入数据的大小以及 blocksize 决定了 Map 任务数目,而 blocksize 的大小又为 Hadoop 集群中的机器数量所影响. Hadoop 中默认 blocksize 为 64M. 在并行运算中,如果 blocksize 太小,则将增加任务数量,增加资源消耗,降低性能;如果 blocksize 太大,则任务又不能达到最大化并行处理,因此需要考虑数据处理量来决定 blocksize 的大小,并且应尽量使 Map 的任务数与集群中的节点数保持一致,从而达到最大并行化分布式任务. 实验中将 blocksize 设为 17M、36M、72M 和 144M,分别对不同的 web 图进行测试,取 3 次实验最大执行时间进行比较,结果如表 3 所示.

表 3 不同 blocksize 对执行时间的影响(单位:秒)

blocksize	17M	36M	72M	144M
maxtime				
31Mweb 图	71	82	84	86
71Mweb 图	119	120	135	136
105Mweb 图	139	144	170	197
534Mweb 图	∞	∞	1450	1237

从表 3 中可以看出,由于实验中采用了三 PC 机集群,节点比较少,因此每个节点在运行 2~4 个 Map 任务时效率较高的. 若集群节点较多时可减少 blocksize,使 Map 任务数增多;若集群节点较少时可适当增加 blocksize,但总应该保证集群中的每个节点都有 Map 任务执行,从而达到最大并行化作业的计算.

(2) MapReduce 下 PageRank 算法的时间消耗

对不同边数的 web 图以 2 机集群和 3 机集群运行 PageRank 并行算法,记录迭代阶段 MapReduce 执行时间,实验 3 次取平均值,其结果如表 4 所示.

表 4 迭代阶段 MapReduce 执行时间(单位:秒)

web 图边数	2 机集群	3 机集群
231 万	75	61
511 万	110	85
760 万	131	109
3.7 千万	1208	724

从表4可以看出,若web图的边数固定不变,则随着集群的节点数的增加,PageRank并行算法所完成的时间明显变短.其主要原因是:随着集群的节点数的增加,集群系统上的多个处理器和内存资源得到充分的利用,从而有效地提高了计算效率.因此可见,基于MapReduce的PageRank算法具有较好的可扩展性.

5 结束语

在Hadoop云计算环境下对PageRank算法进行了并行化计算的分析研究,将PageRank算法与MapReduce编程模型有效地结合起来,形成了在Hadoop云计算环境下运行的PageRank并行算法.利用Hadoop搭建云环境集群,对不同规模的Web数据集进行了测试,分析不同的Blocksize参数对于算法计算性能的影响.此外,实验还表明,集群节点越多,算法运行效率越高,有较好的扩展性.下一步的工作是对PageRank算法需大量迭代,产生较多访问HDFS和集群间通信的次数而耗费时间的问题继续研究,进一步提升基于MapReduce的PageRank算法的效率.

参考文献:

[1] Wikipedia. MapReduce[EB/OL]. [2011-10-16]. <http://zh.wikipedia.org/wiki/MapReduce>. 2011.

- [2] Jeffrey D, Sanjay G. MapReduce: simplified data processing on large clusters[C]//Proc. of OSDI'04. Boston, MA, USA, 2004:137-150.
- [3] 江务学,张璟,王志明,等. MapReduce并行编程架构模型研究[J]. 微电子学与计算机, 2010, 27(6):168-170.
- [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):13-17.
- [5] Arasu A. Pagerank computation and the structure of the web-Experiments and algorithms[D]. CA, USA: Computer Dept. of Stanford University, 2002.
- [6] Kamvar S D, Haveliwala T H. Extrapolation methods for accelerating pagerank computations[D]. CA, USA: Stanford University, 2003.
- [7] Tom White. Hadoop: the definitive guide[M]. America: O'Reilly Media, 2009:18-25.
- [8] Wikipedia. Hadoop[EB/OL]. [2011-10-15]. <http://en.wikipedia.org/wiki/Hadoop>, 2011.
- [9] 高勋. 基于云计算的Web结构挖掘算法研究[D]. 北京:北京交通大学, 2010.

作者简介:

陈 宫 男, (1976—), 硕士, 讲师. 研究方向为云计算、数据挖掘.

牛秦洲 男, (1956—), 博士, 教授. 研究方向为网络控制.

(上接第80页)

- [2] Martens A. On compatibility of web service[J]. Petri Net Newsletter, 2005(65):12-20.
- [3] Foster H, Uchitel S, Magee J, et al. Compatibility verification for web service choreography[C]//Proc of the Int'l Conf on Web Service(ICWS). San Diego: IEEE Computer Society, 2006:738-741.
- [4] Fu X, Bultan T, Su JW. Analysis of interacting BPEL web services[C]//Proc Of the Int'l World Wide Web Conf (WWW). Budapest: ACM, 2004:621-630.
- [5] 邓水光,李莹,吴健,等. Web服务行为兼容性的判定与计算[J]. 软件学报, 2007, 12(18):3001-3014.
- [6] 闫春刚,蒋昌俊,丁志军,等. 工作流网系统合理性的语言特性[J]. 应用科学学报, 2009, 29(1):61-64.
- [7] 彭纪奎,胡平. 基于Web的空间数据可视化研究[J]. 微电子学与计算机, 2010, 27(1):147-150.
- [8] Wil M P. van der Aalst, Kees M van Hee, Peter Mas-suthe, Natalia Sidorova and Jan Martijn van der Werf. Compositional Service Trees[J]. Volume 5606 of Lec-

ture Notes in Computer Science, 2009.

作者简介:



王 娟 女, (1987—), 硕士生. 研究方向为Web服务、Petri网行为理论.



方贤文 男, (1975—), 博士, 副教授. 研究方向为Petri网、Web服务、可信软件.