DOI: 10.7631/issn.1000-2243.2014.01.0045 文章编号: 1000-2243(2014)01-0045-05

基于云计算的 Pagerank 算法的改进

郑 晶

(福建江夏学院电子信息科学学院,福建 福州 350108)

摘要: 针对 Pagerank 算法在 Web 结构挖掘中存在的需要大量迭代的问题,提出一种新的方法. 该方法通过对原始 Pagerank 值的计算公式进行改进,降低了迭代次数. 实验表明,在云计算环境下,新方法减少了网络通信和访问 HDFS 的消耗,在时间花费上优于传统的 Pagerank 算法.

关键词: 云计算; Web 结构挖掘; Pagerank; Mapreduce

中图分类号: TP311.13 **文献标识码:** A

An improved Pagerank algorithm based on cloud computing

ZHENG Jing

(College of Electronic Information Science, Jiang-xia University, Fuzhou, Fujian 350108, China)

Abstract: With the advent of the era of cloud computing, it is a new important research topic to discuss the problem of the web mining based the cloud computing. A new method is proposed to solve the large number of iterations problems in the Web structure mining for the Pagerank algorithm. Through improving the formula of the original pagerank value, it reduces the number of iterations. The experiments show that this method reduces the network traffic and the consumption of accessing HDFS in the cloud computing enviroment, and it is superior to the original Pagerank algorithm in the time consumption.

Keywords: cloud computing; Web structure mining; Pagerank; Mapreduce

0 引言

随着 Web 信息技术的发展,用户可以便捷地获取各种信息,但是也面临着如何从大量的 Web 信息中获取有用的信息的问题. 1996 年,Chen M S[1] 把数据挖掘方法引入到 Web 领域,掀起了 Web 数据挖掘研究的热潮. 1998 年,Brin 和 Page[2] 提出了 Pagerank 算法,该算法基于链接分析理论提出搜索引擎算法. 随后学者主要从以下两个方面进行深入研究: ①主题漂移,即无法区分超链接网页与当前页面的主题相关度,斯坦福大学的 Tather Haveliwala[3] 提出的主题敏感算法(topic-sensitive Pagerank, TSPR),华盛顿大学的 Matthew Richardson 和 Pedro Dominggos[4] 提出了结合链接分析和文本内容的算法(MP-Pagerank 算法); ②偏重旧网页问题,即越旧的网页的排名越靠前,宋聚平[5]、戚华春[6] 等分别提出了具有时间反馈因子的算法. 这些研究丰富了 Pagerank 算法,使其更具有实用性.

目前,Pagerank 算法存在两个方面的缺点: 一是计算网页的重要性时存在一些问题,比如主题漂移、偏重旧网页、忽视用户个性化等; 另一方面是算法的执行速度. 随着互联网的发展,网页信息成为一种海量数据,那么 Pagerank 算法的实现就需要存储海量的数据. 在云计算环境下实现此算法可以解决数据储存的困难. 本文在云计算基础架构 Hadoop 下将 Pagerank 算法与 Mapreduce[7] 框架相结合来提高该算法的运行效率,通过对原始 Pagerank 算法计算公式的改进,降低迭代次数来提高 Pagerank 值计算的时间效率.

收稿日期: 2012-09-24

通讯作者: 郑晶(1980-),讲师,E-mail: zhengjing80@qq.com

基金项目: 国家自然科学基金资助项目(30671680); 国家科技型中小企业技术创新基金资助项目(11C26213502126); 福建省教育厅科技资助项目(JA11269); 福建江夏学院青年资助项目(2011C005)

1 传统 Pagerank 算法及存在问题

1.1 Pagerank 算法简介

Pagerank 算法是利用网页之间的链接关系进行 Web 挖掘的算法，被 Google 搜索引擎采用. Pagerank 算法的基本思想是: 如果网页 i 存在一个指向网页 j 的链接，则表明 i 的所有者认为 j 比较重要，从而把 i 的一部分重要性得分赋予 j. 所以 Pagerank 算法是把一个页面的重要性均分并且传递给它所引用的网页，这样就可能使被引用的页面得到推荐. 因此网页之间的链接关系表明了此 Web 的重要性. Pagerank 算法的计算公式如下:

$$\mathrm{PR}(j) = (1-d) + d \times \sum_{(i,j)\in E}^{n} \frac{\mathrm{PR}(i)}{C(i)} \tag{1}$$

其中: E 为网络中所有网页链接边的集合; $\mathrm{PR}(j)$ 表示页面 j 的 Pagerank 值; $\mathrm{PR}(i)$ 表示链接到页面 j 的页面 i 的 Pagerank 值; $c(i)$ 表示链接到页面 j 的面页 i 所链出的网页总数; n 为互联网上所有网页的总数量; d 为阻尼系数，表示跳转到其他网页的概率，一般取为 0.85.

为了便于新算法的提出，对公式(1) 用向量来表示.

令 $\mathrm{PR} = (\mathrm{PR}(1), \mathrm{PR}(2), \cdots\cdots, \mathrm{PR}(n))^{\mathrm{T}}$, $\boldsymbol{A}_{ij}^{\mathrm{T}} = \begin{cases} 0(\text{其他}) \\ 1/C(i) \ (\exists(i,j) \in E) \end{cases}$，则

$$\mathrm{PR}^{k+1} = (1-d)\boldsymbol{I} + d\boldsymbol{A}^{\mathrm{T}}\mathrm{PR}^{k} \tag{2}$$

其中: $\boldsymbol{I}$ 为单位矩阵.

根据 Pagerank 算法的计算公式来计算每个网页的 Pagerank 值，需要进行 n 次的迭代，这对于海量的网页来说需要花费巨大的时间，所以在计算 Pagerank 值的时候只要求在合理的误差之内. 本文使用 $\frac{\|P_k - P_{k-1}\|}{n} < \varepsilon$, $\varepsilon = 10^{-6}$ 来判断其 Pagerank 值达到收敛范围.

1.2 基于 Mapreduce 的 Pagerank 算法的实现

Hadoop 是一个分布式计算的开源框架，实现了 Google 云的主要技术，目前已经得到广泛的应用. Hadoop 中最核心的设计是 Mapreduce 和 HDFS(hadoop distributed file system) . HDFS 是一种高吞吐量的分布式文件系统，它能高容错、高可靠地处理和存储大量数据，与 Mapreduce 编程模型框架相结合，为应用程序提高高效的数据访问模式. Mapreduce 编程模型[8] 由 Google 在 2004 年提出，用来处理信息量大，且需要在一定时间内完成计算的数据. 由于 Mapreduce 具有函数式和矢量编程语言的共性，使其在海量数据的搜索、挖掘、分析等领域得到广泛应用. 根据 Mapreduce 的特性，本文把 Pagerank 算法与 Mpareduce 框架相结合对该算法的收敛性进行研究.

如图 1 所示，一个 Mapreduce 任务通常先被分割成若干独立的数据块，由 Map 函数并行处理; 然后框架会对 Map 函数的处理结果进行排序，并输入给 Reduce 函数; Reduce 函数组合各地的信息最终合成一个输出文件，此文件被存储在文件系统 HDFS 中.

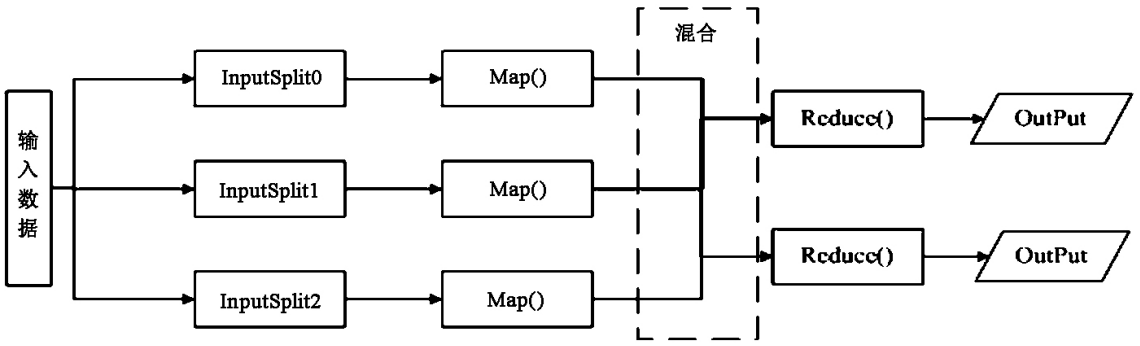

图 1 Hadoop – Mapreduce 的执行流程

Fig. 1 Implimentation of Hadoop – Mapreduce

在 Mapreduce 框架下实现 Pagerank 算法，其算法描述如下:

1) 利用 Map 函数对每条链接中的目标节点输出 < key，value > ，其中 key 代表目标节点.

$$\text{value} = \frac{\text{起始节点的 Pagerank 值}}{\text{目标节点的数量}}$$

2) 框架对 Map 函数的计算结果进行处理，归类每个 key 对应的 value，并作为 Reduce 函数的输入. 在 Reduce 函数中根据公式(1) 计算每个页面新的 Pagerank 值，并存放到文件系统 HDFS 中，用于下一次迭代.

3) 把输出的结果作为框架下一次迭代的输入，直到 Pagerank 值达到所要求的收敛度.

从 Pagerank 算法在 Mapreduce 框架下运行的流程可以发现 Pagerank 算法存在的两个问题; ①迭代速度慢，因为每次的迭代都要在集群中进行通信和访问 HDFS; ②迭代次数多，只有达到预想的收敛度该算法才停止计算，这需要大量的迭代次数，因此提出提高 Pagerank 值的计算跨度来加速计算的收敛速度，减少时间花费，提高运行效率.

2　改进的 Pagerank 算法

2.1　改进算法的基本思想

针对以上问题，对 Pagerank 算法的计算公式(2) 进行递推. 这里只计算跨度为 2 的公式递推. 由公式(2) 可以得出:

$$\mathrm{PR}^{k+2} = (1-d)\boldsymbol{I} + d\boldsymbol{A}^{\mathrm{T}}\mathrm{PR}^{k+1} = (1-d)\boldsymbol{I} + d(1-d)\boldsymbol{A}^{\mathrm{T}} + (d\boldsymbol{A}^{\mathrm{T}})^2\mathrm{PR}^{k} \tag{3}$$

根据公式(3)，改进后的 Pagerank 算法的基本思想如下:

1) 利用 Mapreduce 框架得到 $\boldsymbol{A}^{\mathrm{T}}$.

2) 通过 $\boldsymbol{A}^{\mathrm{T}}$ 计算 $(\boldsymbol{A}^{\mathrm{T}})^2$.

3) 计算 Pagerank 值. 此步骤中，把公式(3) 分为两个部分. 如果 Map 函数输入的是 $\boldsymbol{A}^{\mathrm{T}}$，计算 $(1-d)\boldsymbol{I} + d(1-d)\boldsymbol{A}^{\mathrm{T}}$，那么每个目标节点 key 所对应的 value 为 $(1-d)\boldsymbol{I} + d(1-d)\boldsymbol{A}^{\mathrm{T}}$; 如果 Map 函数输入的是 $(\boldsymbol{A}^{\mathrm{T}})^2$，计算 $(d\boldsymbol{A}^{\mathrm{T}})^2\mathrm{PR}^k$，那么每个目标节点 key 所对应的 value 为 $(d\boldsymbol{A}^{\mathrm{T}})^2\mathrm{PR}^k$. 框架对 Map 函数的输出结果进行处理，是每个目标节点 key 对应的所有 value 形成一个 list，作为 reduce 函数的输入. Reduce 函数对每个目标节点 key 所对应的 value list 进行相加就可以得到新的 Pagerank 向量 PR^{k+2}.

改进的 Pagerank 算法称为 K - PR(K - step PageRank) 算法通过增大跨度的方法来减少 Pagerank 值的计算次数，使其尽快达到收敛. 其算法流程如图 2 所示:

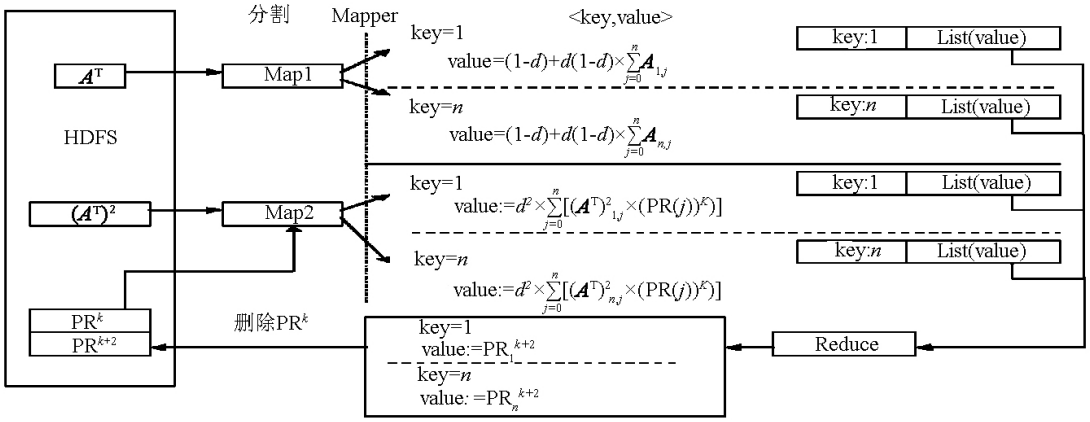

图 2　K - PR 算法的计算流程图

Fig. 2　The flowchart of K - PR algorithm

2.2　改进算法的描述

Step 1: computing $\boldsymbol{A}^{\mathrm{T}}$, $(\boldsymbol{A}^{\mathrm{T}})^2$

Step 2: map( )　//把经过分割的 key/value 数据集合转化成一些列中间 key/value 集合对

{input　$\boldsymbol{A}^{\mathrm{T}}$, $(\boldsymbol{A}^{\mathrm{T}})^2$

output　< key，value >　　//key 表示每个目标节点 i; value 为其相对应的部分 Pagerank 值

if(input A^{T})
{ key = A^{T}中每一行目标节点 i
value = $(1-d)I + d(1-d)A^{\mathrm{T}}$
output(key, value)
}
if(input $(A^{\mathrm{T}})^2$)
{ key = $(A^{\mathrm{T}})^2$中每一行目标节点 i
value = $(dA^{\mathrm{T}})^2 \mathrm{PR}^k$
output(key, value)
}
}
reduce( ) //把 value 集合进行汇总处理，然后返回每个节点的 Pagerank 值
{ input < key , values > //其中 key 为 map 函数输出的目标节点 i，values 为其节点相对应的部分 pPagerank 值的 list，记为 partial Pagerank [2] list
output < key, value > //其中 key 为目标节点 i，value 为其节点跨度为 2 的新 Pagerank 值
value = 0
for 每个目标节点 in values list
value + = partial Pagerank [2]
output < key , value >
}

2.3 性能分析

通过 Pagerank 算法与改进后的 K - PR 算法的时间代价的比较可以得到，假设最终的 Pagerank 值需要 100 次的迭代，那么对于 Pagerank 算法首先需要 1 次的调用求得 A^{T}，然后需要 100 次的 Mapreduce 过程，总共需要 101 次迭代; 改进后的算法，需要三个阶段的 Mapreduce 过程，第一阶段需要 1 次，第二阶段需要 1 次，第三阶段需要 50 次，总共 52 次. 虽然改进后的 K - PR 算法每次执行的时间要长于 Pagerank 算法，因为传统的只要计算 A^{T}，而改进后的要计算 A^{T} 和 $(A^{\mathrm{T}})^2$，但是改进后的 K - PR 算法在总的执行时间上几乎节省了一半. 其次，对 Pagerank 算法与 K - PR 算法进行空间代价的比较. Pagerank 算法只要存储 A^{T} 这个矩阵，而改进后的算法需要存储 A^{T} 和 $(A^{\mathrm{T}})^2$，空间代价增加了，只是这种空间代价在云计算环境下不是一种负担. 另外，每次启动 Mapreduce 框架都将访问 HDFS，改进后的算法减少了 Mapreduce 过程，那么也减少了网络通信和访问 HDFS 的消耗，提高了 Pagerank 值计算的效率，从整体上讲，改进后的算法更适合于云计算环境.

如果要实现更少的迭代次数，可以增加 k 的取值，只是在迭代计算 Pagerank 值的第二步骤中，需要计算出关于 k 相关的邻接矩阵乘积 $(A^{\mathrm{T}})^2 \cdots (A^{\mathrm{T}})^{\mathrm{T}}$，这增加了每次迭代时的数据量、运行时间和存储空间.

3 实验结果分析

为了更好地验证改进后的算法可以节省时间，在 Hadoop 平台上进行测试. 由于实验的条件有限，实验数据采用 Stanfor 大学网络分析平台[9] 提供的 Web 图数据. 使用到的 Web 图如表 1 所示:

实验测试环境为: 两台 PC 机，内存 2 G，Linux 操作系统和 Hadoop 软件，开发环境是 MyEclipse. 平台的搭建参考文献[10]，由于受到实验条件的限制，实验只考虑了迭代一次与三次的情况，实验结果如图 3、图 4.

从图 3 可以看出，只迭代一次传统的 Pagerank 算法的时间比改进后的算法少，但是改进后的时间小于两倍的传统算法所需的时间. 因为改进后的算法在计算新的 Pagerank 时，跨度为 2，所以在整体上，改进后的算法是优于传统的算法，可以提高执行效率. 从图 4 可以看出，当迭代次数增加时，K - PR 算法与传统 Pagerank 算法之间的时间差越来越小，甚至在边数 1.5×10^6 条的条件下，已经比传统 Pagerank 算法

还少，因为改进后的 K – PR 算法更具有并行执行的优势，更适合应用在云计算环境下.

表 1　Web 图数据

Tab. 1　The data of the Web graph

名称	节点数/个	文件大小/M	边数/条
Notre Dame 的 Web 图	3.3×10^5	20	1.5×10^6
Stanford 大学 Web 图	2.8×10^5	31	2.3×10^6
Berkely 和 Stanford 大学 Web 图	9.1×10^5	71	5.1×10^6
Yahoot Web 图	6.9×10^5	105	7.6×10^6

实验中 K – PR 算法所占用的存储空间的大小分别为 39 M、31 M、71 M、105 M，将近是 Pagerank 算法所占用空间的 2 倍，因为 K – PR 算法要比 Pagerank 算法多存储矩阵$(A^{\mathrm{T}})^2$.

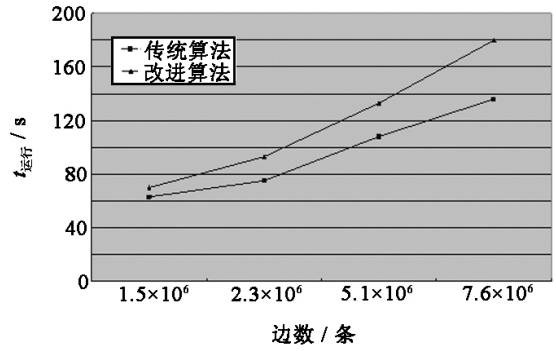

图 3　迭代一次的两种算法实验时间对比

Fig. 3　The comparison chart of two algorithms in the experimental time of one iteration

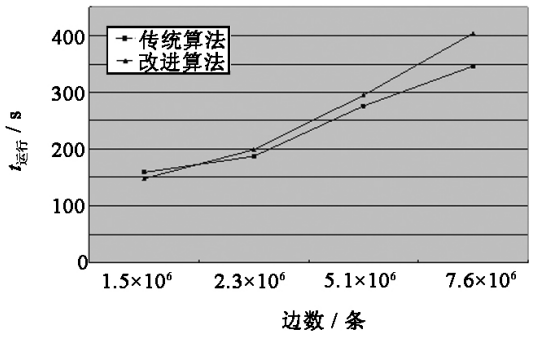

图 4　迭代三次的两种算法实验时间对比

Fig. 4　The comparison chart of two algorithms in the experimental time of three iteration

4　结语

把 Pagerank 算法和云计算环境相结合，提出一种新的更适合云计算环境的算法. 从上面的性能分析可以看出该算法是把空间换去时间的做法，提高了计算 Pagerank 值的时间，同时也减少了网络通信和访问 HDFS 的消耗，使其更具有并行执行的优势，具有一定的实用性.

参考文献:

[1] Chen M S，PARK J S，YU P S. Data mining for path traversal patterns in a Web environment [C] //Proceedings of the 16th International Conference on Distributed Computing Systems. Hong Kong: IEEE，1996: 385 – 392.

[2] Brin S，Page L. The anatomy of a large – scale hypertextual Web search engine [C] //Proceedings of the Seventh International World Wide Web Conference. Brisbane: Elsevier Science Publishers，1998: 107 – 117.

[3] Haveliwala T H. Topic – sensitive Pagerank [C] //Proceedings of the Eleventh International World Wide Web Conference. New York: ACM，2002: 517 – 526.

[4] Richardson M，Domingos P. The intelligent surfer: probabilistic combination of link and content information in Pagerank [J]. Advances in Neural Information Processing Systems，2002，14，1 441 – 1 448.

[5] 宋聚平，王永成，尹中航，等. 对网页 PageRank 算法的改进 [J]. 上海交通大学学报，2003，37(3): 397 – 400.

[6] 戚华春，黄德才，郑月锋. 具有时间反馈的 PageRank 改进算法 [J]. 浙江工业大学学报，2005，33(3): 272 – 275.

[7] 程苗. 基于云计算的 Web 数据挖掘 [J]. 计算机科学，2011，38(10A): 146 – 149.

[8] Dean J，Ghemawat S. Mapreduce: simplied data processing on large cluster [C] // Proceedings of the 6th Conference on Sympo-sium on Opearting Systems，Design and Implementation. [s. l.]: USENIX Association，2004.

[9] Stanford Universtity. Standfor network analysis platform [EB/OL]. [2002 – 05 – 08]. http: //snap. stanford. edu/data/index. html.

[10] 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.

(责任编辑: 林晓)