基于 HBase 的并行 BFS 方法

强 彦 卢军佐 刘 涛 裴 博
(太原理工大学计算机科学与技术学院 太原 030024)

摘 要 NoSQL 数据库作为下一代巨型数据的存储模式,在科学计算和商业计算领域均发挥着重要作用,受到当前学术界和企业界的广泛关注。提出一种新的基于 NoSQL 数据库 HBase 的并行求取最短路径树的方法。首先利用 Watts-Strogatz 模型完成对巨型网络的数学建模,这种建模方式使得网络模型具有一定的聚类效果;其次利用 HBase 最近发布的 Coprocessor 简化和改进并行 BFS 方法,提高其计算效率。此外,还设计并实施了大量实验,得出了巨型网络的最短路径树,验证了该算法的正确性和有效性;同时对比其它路径算法,验证了该算法的高效性。

关键词 HBase,协处理器,并行广度优先算法,Mapreduce,NoSQL 数据库
中图法分类号 TP301 **文献标识码** A

HBase Based Parallel BFS Method

QIANG Yan LU Jun-zuo LIU Tao PEI Bo
(Department of Computer Science and Technology, Taiyuan University of Technology, Taiyuan 030024, China)

Abstract As the next generation of storage model of giant data, NoSQL database plays an important role both in the fields of scientific computing and commercial computing, and has gained wide attention in academia and business community. We presented a new parallel method based on HBase to gain the shortest path tree. Firstly, Watts-strogatz mode was used to complete the mathematical modeling of giant network, therefore the network would have some cluster effects. Secondly, we made a simplification and improvement to the parallel breath-first search method, in order to improve its calculation efficiency. In addition, we designed and implemented a large number of experiments. According to the experiment results, we obtained the giant network shortest path tree, and verified the correctness and validity of the algorithm. Meanwhile, Contrast to the other path algorithm, we verified the efficiency of the algorithm.

Keywords HBase, Ccoprocessor, Parallel BFS, Mapreduce, NoSQL database

网络图在现代社会可以说无处不在,如人们经常接触的互联网的超级链接结构、社交网络和交通运输网络等等,但是随着科技的发展,出现了拥有上千万甚至是上亿节点的巨型网络,对于这种网络图信息的提取将是解决相关学科的关键。以近年来发展迅速的社交网络为例,在社交网络上的节点数目可能是上亿级的,如何在如此巨大的网络中高效地提取对用户有价值的信息并将信息及时反馈给用户,将是决定社交网站是否能够吸引用户的关键问题之一,所以解决巨型网络图的信息提取问题在今天的社会变得尤为重要。

1 引言

NoSQL[1]指的是非关系型的数据库。随着互联网速度的提高和数据量的增大,传统的关系数据库在应付这种超大规模和高并发的数据时显得力不从心,暴露了很多难以克服的问题,而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。Apache 的 HBase[2] 和 Google 的 Bigtable[3] 正是这种数据的代表。

HBase 是一个分布式的、面向列的开源数据库,它是在 hadoop 上的 hdfs(Hadoop Distributed File System)[4] 分布式存储系统上建立起来的,该技术源于 Google 的 Bigtable:一个结构化数据的分布式存储系统。就像 Bigtable 利用了 Google 文件系统所提供的分布式数据存储一样,HBase 在 Hadoop 之上提供了类似于 Bigtable 的能力。HBase 是 Apache 的 Hadoop 项目的子项目,不同于一般的关系数据库,它适合于非结构化数据存储的数据库。HBase 是基于列而不是基于行的模式,且它并不支持关系数据库的结构。HBase 由于是建立在分布式系统上的,因此具有处理大数据的能力。

随着云计算技术的发展,相应的各种基于云平台的公式陆续出现,尤其是最近关于 Lin[5] 对基于 mapreduce 框架[6] 公式的研究,很大程度上开拓了我们的思路,Lin 等人对 BFS (breath-first search)广度优先算法进行改进,使其可以以 mapreduce 的分布式方式运行,这就是他们提出的并行广度算法 (parallel breath-first search),该算法可以充分利用分布式平台的计算能力求取网络中特定的点到其它所有点的最短路径

到稿日期:2012-06-11 返修日期:2012-09-15 本文受国家自然科学基金项目(61202163,61240035),山西省自然科学基金(2012011015-1),山西省科技攻关项目(20120313032-3)资助。

强 彦(1969—),男,博士,副教授,主要研究领域为云计算、图像处理、并行计算,E-mail:dicom8@yahoo.com.cn;卢军佐(1986—),男,硕士,主要研究领域为云计算、医学图像处理;刘 涛(1988—),男,硕士,主要研究领域为云计算;裴 博(1987—),男,硕士,主要研究领域为云计算。

的距离。HBase Coprocessor[8]是很多人对 hbase-0.92的重大期待之一。它让离线分析和在线应用很好地结合在一起，另外也极大地拓展了 HBase 的应用丰富性。

Lin 等人的并行 BFS 算法只是在数据的处理阶段利用 mapreduce 框架的高效计算能力，但是该算法并没有充分利用 hadoop 中高效的分布式存储架构 hdfs。本文针对并行 BFS 算法的不足，将 HBase 这种利用分布式存储架构的数据库加入到 BFS 中，并利用 HBase 中的 Coprocessor 功能进一步提高算法的运行效率。

2 传统方法

E. W. Dijkstr 发明了 Dijkstra 方法[9]，该方法在解决传统网络图中求取特定点到图中其它点的最短路径长度——单源最短路径问题时有良好的计算效果。

```
DIJKSTRA(G,w,s)
  d[s]←0
  For all vertex v∈V do
      d[v]←∞
  Q←{V}
  While Q≠∅ do
      u←EXTRACTMIN(Q)
    For all vertex v∈u. ADJACENCYLIST do
        If d[v]>d[u]+w(u,v) then
            d[v]←d[u]+w(u,v)
```

以上所示的伪码是 Cormen 等人[10]所提出的经典格式，算法输入的数据是一种有向连接图 $G=(V,E)$，图的格式是邻接链表的形式，其中 V 代表图中的节点，E 代表边，w 表示为边的权重，如 $w(u,v)$为节点 u 和 v 之间边的权重。算法的过程：第一步初始化，将指向源节点 s 自身的最短路径值 $d[s]$ 设为 0，同时将网络中其它节点到源节点的最短路径值 $d[v]$ $(v\in V)$设为∞，集合 Q 中保存着所有节点 d 值的递增序列，初始化是将序列的第一个值作为源节节 s 的 d 值，其值为 0，其它各个节点均为∞；第二步迭代，算法取出集合 Q 中最小的节点 u 和其 $d[u]$值，接着遍历 u 的所有邻接节点 v，如果 $d[v]$大于 $d[u]+w(u,v)$，就将前者的值替换掉，否则保持不变，这样不断进行迭代，最终当集合 Q 中为空时就得到了到源点 s 的最短路径树。

3 并行广度优先最短路径算法(parallel breadth-first search,BFS)

```
Class Mapper
  Method Map(nid n,node N)
  d←N. DISTANCE
  EMIT(nid n,N)
  For all nodeid m∈N. ADJACENCYLIST do
      EMIT(nid m,d+1)
Class REDUCER
  Method Reduce(nid m,[d1],[d2]…)
      dmin←∞
        M←∅
      For all d∈counts[d1,d2…] do
        If ISNODE(d) then
            M←d
        Else if d<dmin then
          M. DISTANCE←dmin
      EMIT(nid m,node M)
```

标准的 mapreduce 版本的 BFS 算法伪码如上所示。算法的初始步骤和 Dijkstra 算法相同，首先设定指向源节点自身的最短路径值为 0，同时将其它节点的最短路径值设为∞。在 Map 阶段，mapper 类接收到所有指向源节点 s 的节点 n 后，向所有 n 的邻接点发送消息，告诉所有 n 的邻接点可以通过节点 n 到达源节点 s，且路径值为源点 s 到节点 n 的路径值加 1，同时需要在 map 过程中完成网络图结构的传送。在 reudce 过程中需要判断传递过来的 value 值是最短路径值还是节点结构，如果是节点结构就重建网络图，如果是路径值 d 需要取出它们中的最小值。接着按照这种 mapreduce 进行迭代，整个计算过程从源点开始，每一次 mapreduce 计算就向路径的下一个节点进行一次跳转，直到遍历网络图中所有的节点。

按照上面的 mapreduce 格式进行循环迭代，直至所有节点的路径值不为∞，那么应该在什么时候迭代结束呢？这个答案是网络图的直径，或是网络图中最远两个节点之间的距离。观察上述内容我们可以发现，可以将终止条件设为所有的节点最短路径值不再为 0。通常这个过程是在 mapreduce 代码结构的外围进行的，hadoop 中提供一个名为“counters”的 API 接口可以实现相关功能。

4 对 BFS 方法的改进

4.1 HBase 数据库的建模

HBase 是一种基于 hdfs 的 Bigtable 类型的数据库，它的表结构和传统的关系型数据库是不同的，在进行基于 HBase 的单源最短路径树计算之前，首先必须创建 HBase 数据表并将文件数据导入表中。这种分布式数据库存储数据单元需要使用三元组，它们分别为行(row)、列族(column family)、列(column qualifier)，HBase 中将这个三元组作为关键值去索引存储的数据单元值。网络图的表示方式如下：

row：= 节点 id

column family：= 边的种类

column qualifier：= 邻接点的 id

value：= 边的权值

这种网络图表示方式直接，并且是一种压缩过的表示方式。由于列族中存储的是边的种类，因此这种表示方式可以存储像超链接图等其它种类的网络图。

在 BFS 算法的过程中会生成单源最短路径树，这种结构也要存储到 HBase 中以便于计算，单源最短路径树的存储方式如下：

row：= 节点 id

column family：= “pointer”

column qualifier：= 源节点 id

value：= cos(src,n)，下一跳节点集

在这里，value 中存储一个字符串，该字符串由逗号分隔的两个字符串组成，其中第一个字符串是结构中存储的节点

到源节点的路径值，第二个字符串是最短路径下一跳的节点的 id 集。

4.2 Coprocessor

由上文可知，传统的 BFS 方法需要 map 和 reduce 两个过程，且在 mapreduce 的过程中需要传递图的结构，这些都给计算构成了一些负担。为了更加优化 BFS，我们将 HBase 的 Coprocessor 引入到计算过程中，使 BFS 的计算效率更高。

Coprocessor 其实是一个类似 mapreduce 的分析组件，不过它极大简化了 mapreduce 模型，只是将请求独立地在各个 region 中并行地运行，并且提供了一套框架来让用户非常灵活地编写自定义的 Coprocessor。这就很类似于一个简单的 mapreduce，map 就是在目标 region server 上的那些 region，而 reduce 则是客户端。这个方法对于习惯于编写 mapreduce 代码的人来说非常直观，可以很容易地实现 sum、min 等接口。

利用 Coprocessor 的特点对 BFS 方法进行改进。需要将实现功能的 Coprocessor 代码移植到稳定的 HBase-0.92 版本，以便进行实验。传统的运行 BFS 的过程中 map 阶段负责传输路径值和网络结构，这使得计算过程占用了一定的时间，在 Coprocessor 中利用聚合器使得只有总路径消耗少于 HBase 表中原有值时该项才会得到更新，这个最优化步骤使得在 map 阶段完成对节点最短路径值 d 的更新，从而去除 reduce 过程，这一过程的伪码如下：

```
Method Map(nid n,node N)
If pointer ot root∈N then
For all nodeid m∈N. ADJACENCYLIST do
    EMIT(nid m,d+1)
```

该方式可以有两个优势，首先 HBase 的数据存储模式使其可以减少数据传输过程的消耗，其次这种只有 map 作业的方式可以节省大量消耗在 sort-shuffle 过程中的时间。但是这种方式会出现一次 map 过程跳跃不止一步的情况，一个 mapper 在取出数据之前该数据已经被其它 mapper 改变，这种级联现象对于 BFS 的计算有很大的影响，我们可以在 map 后立刻强制进行数据更新以强化这种级联现象，减少迭代次数。

5 指标

为了验证上文提出的算法，在 hadoop 的 mapreduce 中实现了我们的算法，且在 Watts-Strogatz 模型产生的网络图中进行了实验。

5.1 硬件指标

用来测试的云集群有 10 个节点，每个节点的配置是 2.6GHz的 cpu、8GB 的内存、1TB 的硬盘、10Gbit 的网络适配器。我们默认的配置是 64 个 mapper 和 64 个 reduce。

5.2 网络模型

在网络理论中，小世界网络也称 Watts-Strogatz 模型[11]，是一类特殊的复杂网络结构，在这种网络中大部分的节点彼此并不相连，但绝大部分节点之间经过少数几步就可到达。小世界网络最早是由邓肯 Duncan Watts 和 Steven Strogatz 在 1998 年引进的，其将高集聚系数和低平均路径长度作为特征，提出了一种新的网络模型，一般称作瓦兹-史楚盖兹模型（WS 模型），这也是最典型的小世界网络的模型。

networkx 在 2002 年 5 月产生，是用 python 语言编写的软件包，便于用户对复杂网络进行创建、操作和学习。利用 networkx 可以以标准化和非标准化的数据格式存储网络、生成多种随机网络和经典网络、分析网络结构、建立网络模型、设计新的网络算法、进行网络绘制等。networkx 中提供了 Watts-Strogatz 模型构建函数，我们可利用这个函数进行网络建模。图 1 是用该工具产生的网络图，该图有 150 个节点，从该图中可以看出它呈现一定的聚类效果。

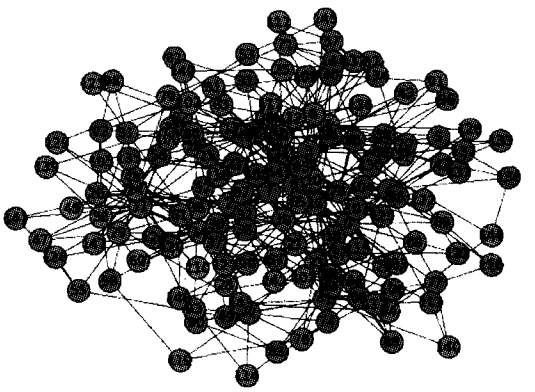

图 1 网络建模

5.3 时间复杂度

在理论上，对于每个节点度的复杂度为 O(1)的网络图，使用 E 表示网络图的边，它的时间复杂度为 $O(|E|\log|E|)$，这是由于在公式迭代的过程中每一条边都会收到一条消息，所有消息的数量为 $O(|E|)$，而后需要将数据存储到 HBase 表结构，这部分的复杂度为 $O(\log|E|)$。单个并行 BFS 传统的运行时间如图 2 红线所示，而改进后的并行 BFS 运行时间如图 2 黑线所示。

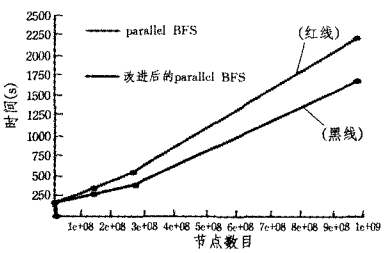

图 2 两种 BFS 算法计算效率的对比

表 1 列出了不同的节点数量的情况下，得到完整的单源最短路径树所需要迭代 BFS 的次数。从表中可以清楚地看出级联现象对于迭代次数的影响，同时如果在 map 后立刻将数据强制存入 HBase 中，这种效果将更加明显。

表 1 节点数量和各种算法迭代次数

节点数量	传统算法的迭代次数	改进后正当情况的迭代次数	Map 后立即存入数据时的迭代次数	路径长度
250	2	1	1	3
$(10)^3$	2	1	1	4
$4\times(10)^3$	3	1	1	4
$16\times(10)^3$	4	2	1	6
$65\times(10)^3$	4	2	1	5
$26\times(10)^4$	4	2	1	5
$(10)^6$	6	3	3	7
$4\times(10)^6$	5	3	3	7
$16\times(10)^6$	6	4	4	9

结束语 本文以 hadoop 平台上的并行广度算法(parallel breath-first search)为基础,提出了一种基于 HBase 的并行 BFS 方法。该方法将 HBase 数据库的表数据存储功能和 HBase 的 Coprocessor 的处理功能加入到并行 BFS 求取单源最短路径树的过程中,使得新的 BFS 方法只用 map 阶段,有效地提高了最短路径树提取的运算效率。最后利用 Watts-Strogatz 模型对网络数据建模,通过对数据运行结果的分析,证实了改进后算法的正确性和高效性。

参 考 文 献

[1] Lakshman A, Malik P. Cassandra : A Decentralized Structured Storage System[C]//SIGOPS. 2010

[2] HBase Homepage[OL]. http://hbase. apache. org

[3] Chang F, et al. Bigtable: A Distributed Storage System for Structured Data[C]//OSDI. 2006

[4] 许春聪,黄小猛. 分布式文件系统存储介质评测与分析[J]. 软件学报,2010,33(10):1873-1880

[5] Lin J, Dyer C. Data-Intensive Text Processing with MapReduce, ser. Synthesis Lectures on Human Language Technologies[M]. Morgan & Claypool Publishers, 2010

[6] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters[J]. Commun. ACM, 2008, 51(1): 107-113

[7] 周国亮,陈红. 基于图形处理器的并行方体计算[J]. 软件学报,2010,33(10):1788-1798

[8] Dean J. Large-scale distributed systems at google: Current systems and future directions[C]//The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware. 2009

[9] Dijkstra E W. A note on two problems in connexion with graphs [C]//Numerische Mathematik. 1959: 269-271

[10] Cormen T H, Leiserson C E, Rivest R L. Introduction to Algorithms[M]. Cambridge, Massachusetts: MIT Press, 1990

[11] Kleinberg J. The small-world phenomenon: An algorithmic perspective[C] // Proceedings of the 32nd ACM Symposium on Theory of Computing. 2000: 163-170

(上接第 218 页)

5 相关工作

目前,关于软件产品线中模型同步方法的研究多是基于软件配置管理(SCM)[5]的概念,即如何管理复杂软件系统中不同版本之间的同步问题。在传统的软件配置管理工作中主要关注的是单个产品的同步管理,且版本管理工具往往在其中起着重要的作用,有些方法在传统的软件配置管理过程中被证明了是可用的。因此,有些研究主要是关注如何将原来用于管理单个产品同步的方法进行扩展以管理产品线的同步[6,7],而有些研究则将版本管理工具作为软件产品线同步管理的基础[6,8]。同时,关于软件产品线同步管理的研究多为使用"小受众"的模型来表示领域设计模型,如 xADL[8-10]。最后,关于如何使用 UML 模型提供的元素来表示软件产品线中的可变性概念,可以参考文献[2]。

结束语 为解决软件产品线中以 UML 类图为表示方法的领域设计模型与应用系统设计模型之间的同步问题,本文提出一种基于双向变换工具 GRoundTram 的设计模型同步方法 SPLSync-GRoundTram,为模型同步提供了一种新的思路。该方法将领域和应用系统设计模型均翻译成图代数语言,并在更加形式化的层次上实现自动化的模型同步。依照 SPLSync-GRoundTram 的方法概览,模型同步由 3 个阶段的活动实现。基于该方法,我们针对一个"网上书城"的设计模型进行了同步操作,以验证方法的有效性。

参 考 文 献

[1] Kang K, Cohen S, Hess J, et al. Feature-Oriented Domain Analysis(FODA) Feasibility Study [R]. Technical Report CMU/SEI-90-TR-021. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1990

[2] Gomaa H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures [M]. Boston: Addison Wesley, 2005

[3] GRoundTram Version 0. 9. 2 User Manual [OL]. http://www.biglab.org/download.html, 2012-05-20

[4] Buneman P, Fernandez M F, Suciu D. UnQL: a query language and algebra for semistructured data based on structural recursion. VLDB [J]. The VLDB Journal, 2000, 9(1): 76-110

[5] Westfechtel B, Conradi R. Software Architecture and Software Configuration Management[C]//10th International Workshop on Software Configuration Management: New Practices, New Challenges, and New Boundaries (SCM 10). Toronto, Canada, 2001: 19-26

[6] Gurp J, Prehofer C. Version Management Tools as a Basis for Integrating Product Derivation and Software Product Families [C]//Proceedings of the Workshop on Variability Management - Working with Variability Mechanisms at SPLC. 2006: 48-58

[7] Yu L, Ramaswamy S. A Configuration Management Model for Software Product Line [J]. INFOCOMP Journal of Computer Science, 2006, 5(4): 1-8

[8] Peng Xin, Shen Li-wei, Zhao Wen-yun. An Architecture-based Evolution Management Method for Software Product Line [C]// Software Engineering and Knowledge Engineering. 2009, 09: 135-140

[9] Chen P, Critchlow M, Garg A, et al. Differencing and Merging within an Evolving Product Line Architecture[C]//PFE. 2003, 03: 269-281

[10] Garg A, Critchlow M, Chen P, et al. An Environment for Managing Evolving Product Line Architectures [C]//ICSM. 2003, 03: 358-367