基于 Hadoop 的句群相似度计算①

宁可为 王 炜 李园伟 (新疆师范大学 教育科学学院 新疆 乌鲁木齐 830054)

摘 要: 介绍了 Hadoop 开源框架、Map/Reduce 编程模型以及语句相似度计算原理，利用 Hadoop 框架下的 Map/Reduce 编程模型实现了句群相似度并行计算方法。通过实验验证了该算法的稳定性和处理大量数据的可行性。

关键词: Hadoop；句群相似度；Map/Reduce

Calculation of Sentence Group Similarity Based on Hadoop

NING Ke-Wei, WANG Wei, LI Yuan-Wei

(College of Educational Science, Xinjiang Normal University, Urumqi 830054, China)

Abstract: Based on the introduction of Open Source Framework of Hadoop, encoded pattern of Map/Reduce and the calculated formula of sentence similarity, the present paper adopts the method of sentence group similarity parallel computing to the encoded pattern of Map/Reduce in terms of Hadoop. And it verifies the stability and feasibility of dealing with tremendous data.

Keywords: Hadoop; sentence group similarity; Map/Reduce

1 引言

随着互连网的飞速发展，网上信息呈爆炸式增长，任何一个用户都可能拥有海量的信息。由于晶体管电路已经逐渐接近其物理性能极限，人类再也不能期待单个 CPU 的速度每隔 18 个月就翻一倍，为我们提供越来越快的计算性能[1]。面对海量信息，基于大规模计算机集群的分布式并行编程是将来软件性能提升的主要途径。其最大优点是可以很容易通过增加计算机来扩充新的计算结点，并由此获得不可思议的海量计算能力,同时又具有相当强的容错能力，一批计算结点失效也不会影响计算的正常进行以及结果的正确性。

本文的背景是在“基于开放域的中文自动答疑系统的研究”课题研究中需要在海量知识库文件匹配出与用户所提问题的答案，而单一服务器去处理不仅影响用户体验，当用户访问量大时甚至引起服务器宕机，因而需要寻求一种简便、可行、有效的并行计算方法。

2 Hadoop概述

Hadoop[2]是一个被设计用来在由普通硬件设备组成的大型群集上执行分布式应用的框架，其框架核心包含一个分布式文件系统 HDFS(Hadoop Distributed File System)和一个 Map/Reduce 编程模型。借助于 Hadoop 程序员可以轻松地编写分布式并行程序，将其运行于计算机集群上，完成海量数据的计算。

2.1 分布式文件系统 HDFS(Hadoop Distributed File System)

Hadoop 分布式文件系统(Hadoop Distributed File System)是一个基于 master/slave 的结构的可靠存储大数据集的文件系统，其架构如图 1 所示。HDFS 由一个管理结点(Namenode)和 N 个数据结点(Datanode)组成，其每个结点均是一台普通的计算机，在 master 上运行的 Namenode 管理着整个分布式文件系统，对文件系统的操作(如建立、删除文件和

① 基金项目:全国教育科学“十一五”规划课题 (CA070275);新疆师范大学研究生科技创新项目(20091106)
收稿时间:2010-04-03;收到修改稿时间:2010-05-06

© 中国科学院软件研究所 http://www.c-s-a.org.cn

文件夹)，在每一个 slave 上运行一个 Datanode。

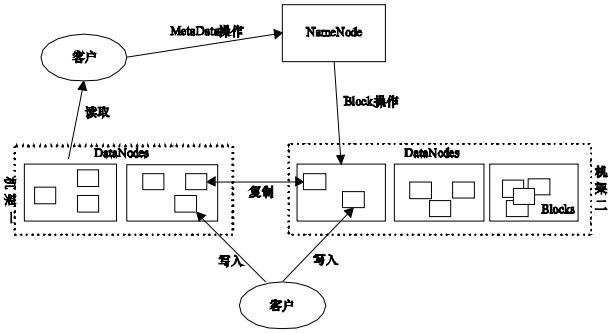

图 1　HDFS 系统架构图

HDFS 被设计成可在集群中跨机器存储海量文件的文件系统。它把每个要存储的文件分割成一个或多个数据块(Block),每个块被分配存储到 Datanode 上。其存储方式采取了副本策略，系统默认三个副本，一个放在本节点上，一个放在同一机架中的另一个节点上，还有一个副本放在另一个不同的机架中的一个节点上，以保证数据的安全性及提高系统的可靠性，可用性。例如：当客户(Client)要执行一个写文件操作时，命令不是马上就发送到 Namenode，Client 首先在本机上临时文件夹中缓存这些数据，当临时文件夹中的数据块达到了设定的 Block 的值(默认是 64M)时，Client 便会通知 Namenode，Namenode 便响应 Client 的 RPC 请求，将文件名插入文件系统层次中并且在 Datanode 中找到一块存放该数据的 Block，同时将该 Datanode 及对应的数据块信息告诉 Client，然后 Client 把这些本地临时文件夹中的数据块写入指定的数据节点。

2.2 Map/Reduce 编程模型

Map/Reduce 是一个进行大数据量计算的编程模型。它将复杂的运行于大规模集群上的并行计算过程高度地抽象到了两个函数，“Map(映射)”和“Reduce(化简)”。Map 函数是用户自定义的，它处理输入的一组(Key,Value)对，操作将产生新的(Key′,Value′)对表示的中间结果集合，而原数据保持不变。函数库将具有相同键的中间结果集通过排序聚集在一起形成(Key′,List(value′))传递给 Reduce 操作。Reduce 函数处理中间键值相关的值集合，合并这些值，最后形成一个相对较小的值集合。系统主要的数据流程如图 2 所示：

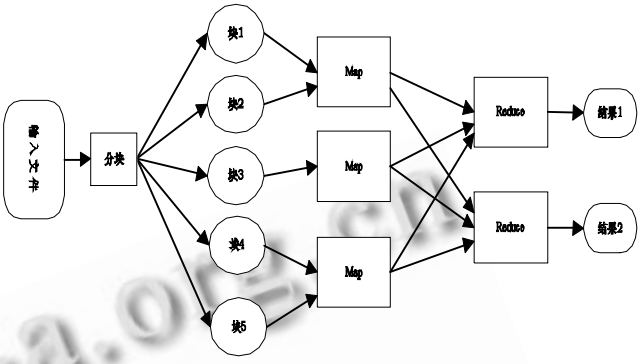

图 2　MapReduce 数据流程图

在 Map/Reduce 过程中，我们将数据并行，也就是将数据分开，而 Reduce 则把分开的数据合到了一起。Map 是一个分的过程，Reduce 则对应合的过程，这一分一合便在不知不觉中完成了计算，所以站在计算的两端来看，与我们通常熟悉的串行计算没有任何差别，由此消解其中的复杂性。

3　语句相似度计算原理

3.1 词语相似度计算

词语相似度是语句相似度计算的基础。词语间相似度计算方法有很多，本文采用中科院计算机研究所的“基于《知网》的词汇语义相似度计算方法”，来计算待比较两个语句中词语的相似度。《知网》是一个以汉语和英语的词语所代表的概念为描述对象，以揭示概念与概念之间以及概念所具有的属性之间的关系为基本内容的常识知识库[3]。在《知网》中每个词汇可表达为若干个“概念”[4]。“义原”是用于描述一个“概念”的最小意义单位[4]。在利用《知网》进行相似度计算，就是把两个词语之间的相似度归结到两个概念之间的相似度，然而所有的概念最终归结于用义原来表示，所以最终求的还是义原之间的相似度。本文只考虑对实词进行求相似度，实词义项相似度计算分成以下四部分：

第一独立义原描述式：将两个概念的第一位置的相似度记为 $d_1=Sim_1(S_1,S_2)$

其他独立义原描述式：语义表达式中除第一独立

© 中国科学院软件研究所 http://www.c-s-a.org.cn

义原以外的所有其他独立义原描述式的相似度记为 $d_2=Sim_2(S_1,S_2)$

关系义原描述式：语义表达式中所有的用关系义原描述式的相似度记为 $d_3=Sim_3(S_1,S_2)$

符号义原描述式：语义表达式中所有的用符号义原描述式的相似度记为 $d_4=Sim_4(S_1,S_2)$

定义 1. 设两个实词概念 G_1 和 G_2，G_1 有 4 个义原描述式:$d_{11},d_{12},d_{13},d_{14}$，$G_2$ 有 4 个义原描述式:$d_{21},d_{22},d_{23},d_{24}$，则概念 G_1 和 G_2 的相似度记为：

$$Sim(G_1,G_2)=\sum_{i=1}^{4} b_i \prod_{j=1}^{i} Sim_j(d_{1j},d_{2j}) \quad [4]$$

其中 $\beta_i(1\leqslant i\leqslant 4)$是可以调节的参数，且有：$\beta_1+\beta_2+\beta_3+\beta_4=1$，$\beta_1\geqslant\beta_2>\beta_3\geqslant\beta_4$

定义 2. 设两个汉语词语 W_1 和 W_2，W_1 有 n 个概念：$g_{11},g_{12},\cdots,g_{1n}$，$W_2$ 有 m 个概念：$g_{21},g_{22},\cdots,g2m$，则 W_1 和 W_2 的相似度为各概念的相似度之最大值，即

$$Sim(W_1,W_2)=\underset{i=1\ldots n,j=1\ldots m}{Max} Sim(g_{1i},g_{2j}) \quad [4]$$

3.2 句子相似度计算

对于两个汉语相似度计算最终是一组词语的相似度计算，首先通过分词程序将句子简化为一组关键词语，其中只包含名词、动词、形容词、限定性副词等同语义关系密切的词语，去掉会引入噪声数据的虚词，然后计算两个关键词组成的集合之间的相似度。

定义 3. 设两个语句 A 和 B，语句 A 由词语 $x_1,x_2,\cdots,x_m$ 组成，语句 B 由词语 $y_1,y_2,\cdots,y_n$ 组成，则语句 A 与 B 的相似为两组词语间的语义相似度，记为 S。

语义相似度 S 的计算如下：

(1) 构造语句 A 与 B 的“相似矩阵”M_{AB}，元素 S(x,y)意为词语 xi 同词语 y_j 间的语义相似度，矩阵中每一行表示语句 A 中的相应词语同语句 B 中所有词语的语义相似度，而每一列表示语句 B 中的相应词语同语句 A 中所有词语的语义相似度。

$$M_{AB}=\begin{pmatrix} S(x_1,y_1) & S(x_1,y_2) & \cdots & S(x_1,y_n) \\ S(x_2,y_1) & S(x_2,y_2) & \cdots & S(x_2,y_n) \\ \vdots & \vdots & \cdots & \vdots \\ S(x_m,y_1) & S(x_m,y_2) & \cdots & S(x_m,y_n) \end{pmatrix} \quad [5]$$

(2) 从矩阵 M_{AB} 中的每行选取最大值 $Max(S(x_i,y_j))$,即求得了语句 A 中的第 i 个词同语句 B 的语义相似度。对取得的最大值集合求算术平均，得出语句 A 同语句 B 的语义相似度 S_{AB}。即

$$S_{AB}=\frac{1}{n}\sum_{j=1}^{n} Max\ (S(x_i,y_j))$$

(3) 对于矩阵的非对称性，需计算语句 B 同语句 A 之间的相似度。用以上的方法，可求得语句 B 同语句 A 之间的语义相似度，即

$$S_{AB}=\frac{1}{m}\sum_{i=1}^{m} Max\ (S(x_i,y_j))$$

(4) 计算以上两个结果的算术平均值，从而得到了两个语句的相似度,即

$$Sim(A,B)=\tfrac{1}{2}(S_{AB}+S_{BA})$$

4 句群相似度并行计算处理及实现

在用 Map/Reduce 实现的算法之前，先判断被处理的语句 A(用户问句)与句群(知识库中语句)，如果被处理的语句包含有大量关键字时，在处理词语相似度计算可用并行模式来实现，否则只在句群相似度计算时用并行模式来实现，以降低网络通信量及 IO 操作次数来提高工作效率，具体处理方法如下：

(1) 对句群 $List_i$ 进行预处理生成格式为：“序号 ‖ 句子”的句组,各语句之间用空格分隔，并存储在一个或若干个文本文件里。

(2) 将文本文件作为 Map 的输入，Map<key,value>=<序号，List[i]>，Map 函数中调用句子相似度计算函数 $Sim(S_1,S_2)$ ，把 List[i]和语句 A(由参数传入)作为形参传入函数求出语句 A 与 List[i]的相似度 Sim(A,List[i])。Map 输出<序号，Sim(A,List[i])>。

(3) Reduce<序号,Sim(A,List[i])>对接收过来的每个 Sim(A,List[i]) 做如下处理：把相似度值 Sim(A,List[i])转化成字符型和序号做连接中间用“|”分隔，形成“序号| Sim(A, List[i])”,Reduce 输出<序号，序号|Sim(A,List[i])>。

(4) 在完成(3)后可在输出目录得到第一次 Map/Reduce 后的结果文件,然后在此结果文件基础进行第二次 MapReduce 操作。

(5) 通过自定义 InputFormat 类对所有的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Sim(A,List[i])以 mapper 任务个数进行划分 N 份，生成<N，$List_j$>作为 Map 的输入。

(6) Map<N,$List_j$>将对每个列表 $List_j$作如下操作:

For(int m=0;m <$List_j$.Length;m + +)

Id=split($List_j$ [m],1) 按“|”拆分出序号

Tmp=split($List_j$ [m],2) 按“|”拆分出 Sim(A,List[i])

if(Tmp > Max) then Max= Tmp

Map 输出为<1,Id+“|”+Max>。

(7) Reduce<key,value>=<1,Id+“|”+Max>)对以 1 为键值的每个 values 值进行折分求最大值,然后存入 Max,最终求得句 A 和所有语句的相似度最高的语句的 Id 及相似度，Reduce 输出<Id ,Max>。

对于包含较多关键字的超长句 $A(x_1,x_2,x_3,\cdots,x_n)$与 $B(y_1,y_2,y_3,\cdots,y_n)$之间的相似度计算的具体处理方法如下:

(1) 将句 A 处理生成“$x_1|A,x_1|y_1$”，“$x_1|A,x_1|y_2$”,…,“$x_n|A,x_n|y_m$”词对。将句 B 预处理生成“$y_1|B,y_1|x_1$”，“$y_1|B,y_1|x_2$”,…，“$y_m|B,y_m|x_n$”词对，所有词对之间用空格分隔存入文件并作为 Map 的输入。

(2) Map<key,value>=<$x_i|A$ 或 $B,x_i|y_j$>,在 Map 函数中根据“|”折分出 x_i 与词 y_i 调用 $Sim(x_i,y_i)$求出词相似度 S,Map 输出为<key,value>=<$x_i|A$ 或 B,S>。

(3) Reduce<$x_i|A$ 或 B,S>对以 $x_i|A$ 为键值的每个 values 值取出最大值,然后存入 Max 即为从矩阵 MAB 中的每行选取最大值 $Max(S(x_i,y_j))$。Reduce 输出<$x_i|A$,Max>。

(4) 从输出文件中按“|”进行折分，分析出 x_i 属于句 A 还是句 B,把属于同一个句子中的 Max(行相似度最大值)累加算出术平均值即得到句 A 同句 B 之间的相似度，按(2)求出句 B 与句 A 的相似度，最后求两个结果的算术平均值即句 AB 相似度。

5 实验环境与结果分析

5.1 实验环境

本程序的测试环境是由一台 100M 交换器，普通双绞线及 5 台 PC 搭建的局域网,其中 PC 机配置如表 1 所示:

表 1 分布式实验环境

编号	硬件配置	操作系统	角色
1	Intel Pentium(R) Dual-core 1.86G 内存 1024MB	Linux Red-Flag 7.0	Namenode
2	Intel Pentium(R) Dual-core 2.10G 内存 2048MB	Linux Red-Flag 6.0	Datanode
3	AMD Athlon Dual-core 4600+ 内存 1024MB	Linux Red-Flag 6.0	Datanode
4	Intel Pentium(R) Dual-core 2.10G 内存 1024MB	Linux Red-Flag 6.0	Datanode
5	Genuine Intel(R) Dual-core 1.66G 内存 1024MB	Linux Red-Flag 6.0	Datanode

5.2 实验结果分析

本文通过对每条所含 10-30 个关键字的语句进行测试，其中程序的 Map 数量设置为 10,结果如表 2 所示:

表 2 语句相度计算结果

语句条数	单机	3 个节点	5 个节点
100	6.0s	36.130s	64.800s
1034	24.406s	53.455s	75.013s
31850	598.094s	567.600s	350.400s
102187	2013.531s	1745.400s	1093.800s
511002	9486.390s	7558.213s	4266.600s

在上述实验结果中可以看出,在语句数量较小或语句含关键字少的情况下，因其算法时间复杂度低，采用多节点的 Hadoop 计算速度不如采用单机模式,

© 中国科学院软件研究所 http://www.c-s-a.org.cn

节点越少，计算速度越快，这是因为在不同计算节点之间需要进行通信和大量磁盘 IO 操作会消耗一定时间。当语句量增加到 102187 条时出现拐点，采用多节点的 Hadoop 计算所消耗的时间明显低于单机模式，如图 3 所示。因此对于复杂度低及小规模的运算并不适宜采用 Hadoop，当数据达到一定量时基于 Hadoop 并行计算是一个很好的问题解决方案。

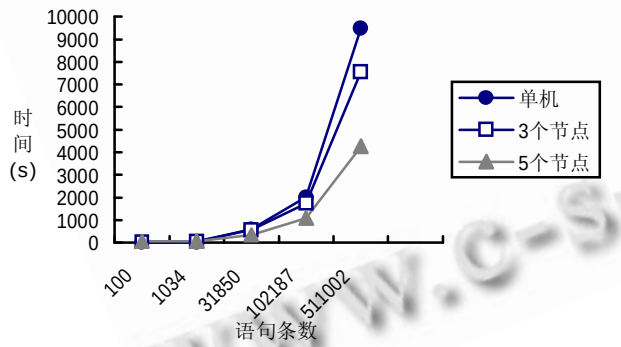

图 3 语句量与结点规模的关系

6 结束语

本文通过利用 Hadoop 框架的 Map/Reduce 编程模型实现大量语句相似度计算，作为一种分布式数据处理技术,Hadoop 它有效地利用分布式和并行技术,解决了在受限的硬件环境下处理大量数据的问题，对大规模数据集的处理和容错都给出了比较满意的解决方案。下一步的工作重点对语句相似度计算算法改进使之能更精确匹配出与原句相似的语句。

参考文献

1 曹羽中.用 Hadoop 进行分布式并行编程.[2009-05-22].http://www.ibm.com/developerworks/cn/opensource/os-cn-hadoop1/index.html#N10059.

2 Tom White.Hadoop:The Definitive Guide.1st ed,United States:O'Reilly Media,Inc., 2009:1－13.

3 张辉丽.计算机领域中文自动问答系统的研究[硕士学位论文].天津:天津大学, 2006.

4 刘群,李素建.基于《知网》的词汇语义相似度计算.第三届汉语词汇语义学研讨会. 2001:6－15.

5 佘文利.基于领域知网的中文自动答疑系统的设计.计算机系统应用 2009,18(3):4－5.

© 中国科学院软件研究所 http://www.c-s-a.org.cn