基于 Map/Reduce 的索引数据云存储模型研究

陆小丽 [1,2], 何加铭 [1,2*]

（1.宁波大学 通信技术研究所, 浙江 宁波 315211; 2 浙江省移动网应用技术重点实验室, 浙江 宁波 315211）

摘要: 针对目前搜索引擎引擎系统存在的数据量庞大、访问用户高并发性和搜索延迟性的特点, 提出了基于云存储的文档索引分类存储模型, 并在索引数据分类存储算法实现过程中, 采用基于 Map/ Reduce 编程模型的二次索引词权重计算, 以降低分类过程中的模糊粒度. 通过实验验证基于该存储模型的算法不仅可以提高海量数据索引库的数据处理效率, 而且在一定程度上降低了检索系统查询延迟, 提高了搜索效率.

关键词: 搜索引擎; 权重; Map/Reduce; 索引

中图分类号: TP393 文献标识码: A 文章编号: 1001-5132（2011）03-0029-05

近年来基于互联网的应用越来越丰富, 特别是搜索引擎已经成为人们访问网络资源的主要途径. 但是网络资源海量数据的膨胀和客户端并发查询的增加, 给搜索引擎系统有限的物理存储、CPU 周期、内存容量和网络带来了巨大瓶颈. 目前, 分布式存储系统是解决海量数据存储问题的主流技术, 其主要原理是将数据进行分块处理, 数据块分散存储在多台独立的设备上. 系统利用多台存储服务器分担存储负荷, 具有良好的可扩展性[1-3]. 目前主要的分布式存储系统有基于 P2P 的分布式存储、基于神经网络的分布式存储、基于 DHT 的分布式存储以及基于云计算的分布式存储[4].

笔者在深入研究分布式存储原理和云计算关键技术——Map/Reduce 编程模型基础上, 对文献[4]基于云计算的云状全分布式网络结构(CFDCSS)进行分析, 并提出新的基于云的索引数据存储模型, 在索引分类方案上采用了二层 Map/Reduce 模型, 对索引词权重进行二次计算, 建立权重索引表, 以权重阈值和索引词权重为分类标准, 对索引数据进行分类. 最后索引数据以反向索引表形式存储于主服务器集群中. 最后通过实验证明: 在基于该分类标准的文档分类具有较高的准确度, 并且在一定程度上提高了检索系统的搜索效率.

1 基于云计算的分布式数据存储模型

1.1 分布式文件系统

分布式文件系统(GFS)的组成结构是由唯一的 1 个主节点和多个数据节点组成, 主节点和各数据节点的通信如图 1 所示. 目录节点(Namenode)是服务器集群中的主节点, 管理子节点数据块和数据节点的映射关系, 是用户访问文件的入口. 客户端需要访问主节点来获取某个文件的所有数据块保

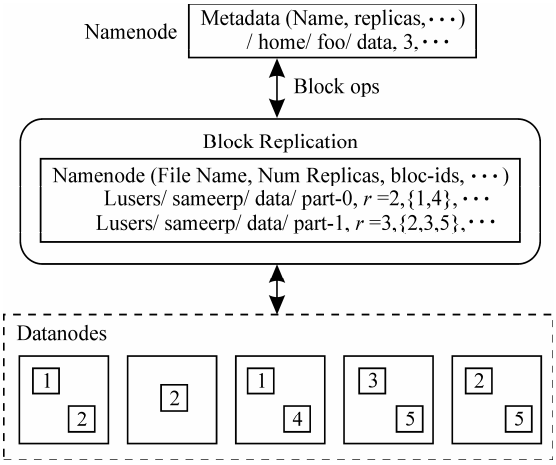

图 1 分布式文件系统

收稿日期: 2011−04−13. 宁波大学学报（理工版）网址: http://3xb.nbu.edu.cn

基金项目: 国家科技重大专项（2011ZX0302-004-02）; 国家重大专项核高基项目（2009ZX01039-001-002-004）; 科技部公共服务平台基金（9C26243314159）; 浙江省科技厅项目（2009C31107）; 宁波大学科研基金（B00241104900）.

第一作者: 陆小丽（1986－）, 女, 浙江杭州人, 在读硕士研究生, 主要研究方向: 数字无线通信. E-mail: luxiaoli234@163.com

*通讯作者: 何加铭（1949－）, 男, 浙江杭州人, 博士/教授, 主要研究方向: 数字无线通信. E-mail: hejiaming@nbu.edu.cn

存的数据节点地址. 数据节点(Datanode)是服务器集群中的一个具体成员, 主要负责数据块的存储; 数据节点根据目录节点的命令创建、删除数据块及其冗余副本[5-6].

分布式存储是为大文件的可靠存储而设计[7]. 文件的存储过程描述为: 1 个大文件被切分为一连串数据块, 除最后 1 个数据块外其他的数据块都是固定大小, 在对数据块进行编号后, 分配给子节点进行存储. 每个数据都会使用副本策略冗余存储, 所以当前数据存储块发生异常, 也不会影响整个文件的数据丢失.

1.2 基于云的数据存储模型

参考 HDFS 和 GFS 文件存储策略以及文献[4]中对基于 DHT 的云状全分布式网络结构(CFDCSS)进行分析, 笔者搭建了新的基于云的数据存储模型及存储搜索引擎的索引数据库(图 2). 位于该模型顶层的是主服务器集群, 负责控制各数据节点的通信和协调管理, 同时接收用户应用请求, 并根据应用请求进行类型分类和控制负载平衡. 各存储应用节点集群相当于存储器部分, 由庞大的磁盘阵列系统和拥有海量数据存储能力的机群系统组成, 处理数据资源存储工作. 数量一定的从节点(负责具体操作的计算机)在集群中形成虚拟节点圈, 每个从节点可能同时存在于不同的虚拟圈中.

虚拟节点圈是机群实例通过哈希计算后的映射节点, 其大小由聚类中的文档索引数目设定[4]. 虚拟圈的存在可让小范围内的服务器间进行资源交流, 以满足数据存储要求变更状态的及时处理.

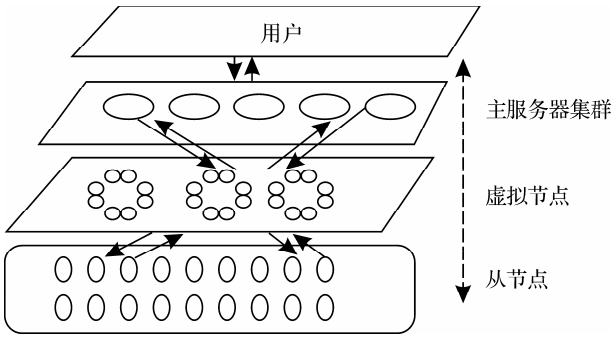

图 2 云数据存储模型

2 索引分类存储算法

2.1 算法模型

搜索引擎的海量数据处理处理和计算要求孕育了云计算的产生. 云计算主要用于解决大文件系统的存储和计算问题[7]. Map/Reduce 作为云计算的关键技术之一, 其操作代表了一大类的数据处理操作方式. Google 从搜索引擎需要选择了 Map/Reduce[8], 并将其应用于云计算系统架构; Hadoop 在模仿 Google 时也采用了 Map/Reduce.

在文献[4]中, 作者搭建了基于云计算的语义搜索引擎系统, 在存储子系统中将整个文章索引库全部集中在 1 个目录下, 没有进行分层和分类. 因此对于每次查询请求, 都需要遍历整个索引库. 这样做的优点是每次查询都不会有查漏信息, 但在查询效率上付出了代价. 特别是像移动终端查询用户, 其查询特点是查询并发性高而且具有很强的目的性, 对搜索的实时性要求也更高, 所以每次搜索遍历整个目录库的方法在实时性要求高的系统中并不适用.

为了在不降低查全率的基础上提高查询效率, 索引的存储方式是一个突破点. 笔者提出了一种基于 Map/Reduce 的索引二次分类存储算法. 在 1 次Map/Reduce结果中构建查询文章的反向索引表, 在 2 次 Map/Reduce 计算中对索引词进行权重计算分类. 图 3 即是基于 2 次 Map/Reduce 模式的文章数据处理模型.

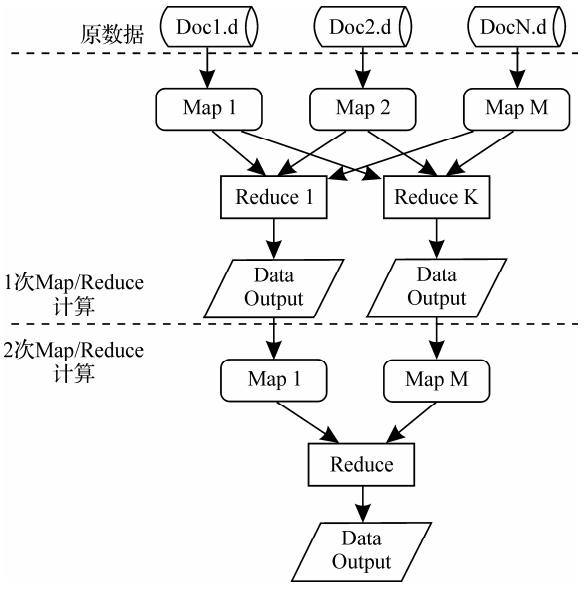

图 3 基于 2 次 Map/Reduce 的分类算法模型

Map/Reduce 是一种编程模型. 首先, Map 函数将 1 组输入(key, value)对应 1 组中间结果的(key, value)对, 然后通过 Reduce 函数把相同 key 值中的

中间结果合并化简[9-10]. 在索引数据处理中, 笔者采用 Map/Reduce 的数据处理原理设计了索引的二次分类存储. 搜索引擎对网络数据的处理过程可以描述为: 原始网页数据→文档预处理过程→生成索引表数据.

经过预处理后的文档集合 $D=\{d_1,d_2,\cdots,d_i\}$ 中的任意文档 d_i 作为 Map/Reduce 中输入的数据块, 该数据块集合为<doc-id *n*, doc *d*>. 该数据块集合第 1 次经过 Map/Reduce 的 map 函数计算特征词 *t* 在文档中的频率 *f*, 输出表示为(*t*, <*n*, *f*>). 所有的文档经过 map 函数处理后, 输出的特征词频率集合(T_i, <doc-id *I*, F_i>). 处理后的数据送往上一层 Reduce, 该阶段是数据重新合并的过程. Reduce 函数将具有相同 *t* 的<*n*, *f*>进行合并, 输出为特征词 *t* 在文档集合 *D* 中的词频记录($<n_1, f_1><n_2, f_2>\cdots$), 经过第 1 次 Map/Reduce 处理后的数据为搜索引擎的反向索引表, 表示为{term *i*, ($<n_1, f_1><n_2, f_2>\cdots$)}, term *i* 为索引词.

数据经过第 1 次 Map/Reduce 处理的结果形成新一轮(key, value). key=term *i*, value=($<n_1, f_1><n_2, f_2>\cdots$)作为输入数据再次经过 Map/Reduce, 计算特征词的权重 *w*. 在第 2 次 Reduce 处理阶段, 设置权重阈值 λ 进行降维, 处理后的结果为($<n_1, w_1><n_2, w_2>\cdots$). 根据 tf-idf 模型, 该权重可以用以下公式进行描述:

$$W_{ij}=F_{ij}\log(N/n_i),,$$

其中, *i* 为特征词向量; *j* 为任意文档; F_{ij} 为特征词 t_i 在文档 d_j 中出现的频率; *N* 为文档集合总数; n_i 为文档集合中含特征词 t_i 的文档数; 计算的结果表示特征词 t_i 在文档 d_j 中的权重.

2 次 Map/Reduce 处理过程将索引引擎预处理后的文档转变为反向索引表[11]和索引词权重表. 根据索引词的权重值和预先设置好的权重阈值对文档进行分类, 满足权重阈值条件的索引文档划分在 1 个集合中. 索引数据经过 2 次分类后, 不再是简单的反向索引表, 而是具有权重阈限的分类反向索引表.

2.2 算法实现

该分类存储算法是基于Map/Reduce编程模型, 第 1 次经过 Map/Reduce 函数的数据处理过程是将过滤后的正文文档转变为倒排索引词频表[12]. 关键算法实现的描述如下:

(1) Map 1

For all term *t* ∈ doc d do;

Frequency(*t*)= Frequency(*t*)+1; //计算索引词频数

Output (term *t*, record <doc-id *n*, Frequency (*t*)>);

(2) Reduce 1

Input term *t*, record($<n_1, f_1><n_2, f_2>\cdots$); //数据输入

Construct record *R*//新建列表

For all record<*n*, *f*>∈ record($<n_1, f_1><n_2, f_2>\cdots$);

Append (*R*,<*n*, *f*>); //合并相同索引词词频

Sort(*R*); //第 1 次分类

Output (term *t*, record *R*);

经过 1 次 Map/Reduce 数据处理后的输出数据流(term *t*, record *R*)是文章索引关键词的词频表. 为了进一步对该倒排表进行分类存储, 设计了 2 次 Map/Reduce. 其中, 在 Reduce 2 过程中, 设置了权重阈值 *Y*. 这里 *Y* 的取值是根据文章数 doc 的数量和索引量的大小而定. 接着, 可计算索引词的权重. 关键算法实现描述如下:

(3) Map 2

Construct *W* //新建权重列表

term *t* ∈doc d;

for<docid *n*, frequency(*t*)> ∈ record *R* do *h*= *h*+1;

weight(*t*, *n*)= frequency(*t*)*log(*N*/*h*); //计算索引词权重

append(term *t*, weight($<n_1, w_1><n_2, w_2>\cdots$));

(4) Reduce 2

For <doc-id *n*, weight(*t*)> ∈ weight *W*

If weight(*t*)>=*Y*; //是否满足权重阈值

Append(W-new <*n*, weight(*t*, *n*)>); //计算权重

Sort(W-new); //进行二次分类

在算法实现中, 对简单文档索引 doc1<blue car, yellow car>, doc2<red car, back car>, doc1<a red cat >进行 1 次 Map/Reduce 数据处理, 得到的词频反向索引表. 2 次 Map/Reduce 处理后, 得到分类标准表的权重反向索引表(表 1).

表 1 权重反向索引表

文档	car	yellow	cat	back	a	red
Doc1	1.168	1.584	0	0	0.584	0
Doc2	1.168	0	0	1.584	0	0.584
Doc3	0	0	1.584	0	0.584	0.584

通过 2 次 Map/Reduce 对索引词频和权重的计算，最后生成的索引数据信息以表格形式存储于分布式文件系统的服务器集群中. 当系统接收到查询请求时，将查询自然语句转换成相应的关键词在分类索引库中进行匹配. 如用户输入“car”，通过与分类索引库的索引词进行匹配，返回给用户 doc1 和 doc2 文档标志，最后到相应的网页存储节点获取原始网页信息.

3 实验仿真与分析

在实验测试中使用的实验对象来源是经过预处理的具有类别标志的文章资料库，选取其中的 600 篇文档. 该实验对象正确的分类有经济、体育、教育、自然，每种类别分别为 150 篇. 测试环境为实验室局域网，Hadoop 分布式平台(由 4 台普通计算机组成)，计算机配置为 CPU Intel 2.0 G，内存 1 G，硬盘 150 G，Linux 操作系统. 其中 1 台是 Namenode 服务器，其他 3 台为 Datanode 从数据节点服务器. 在实验过程中，将这 600 篇预处理文档在不分类前提下，通过 2 次 Map/Reduce 分类存储模型中进行分档分类，测试结果与标准分类进行比较. 同时改变权重阈值 Y 的大小，分析在不同阈值情况下的分类效果.

图 4 中的柱形索引分类准确率中的每种颜色代表一类索引文档. 从图中可以看出随着权重阈值 λ 的增加，每类索引文档的分类准确度都随之提高. 如体育类在 λ=0.5 时，分类准确率为 82%；在 λ=0.8 时，准确率提升为 85%. 但是从整体分类情况来看，部分索引分类准确度普遍较高，而存在某些类别的文档分类效果并不是很好，如自然类在 λ=0.8 时，其分类准确度才为 70%. 所以在其分类效果上，如加入语义分类等技术可能效果会更好，这也是我们后期研究的方向.

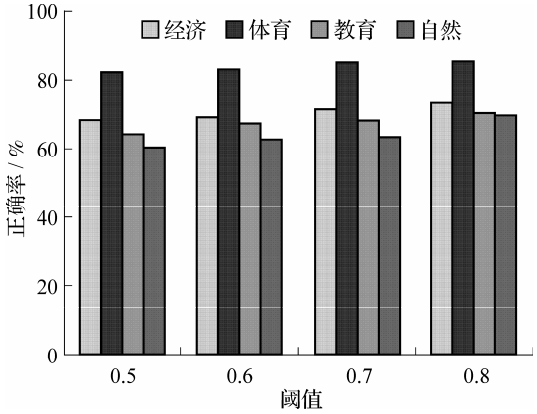

图 4 索引文档分类标准率

为了直观显示基于 Map/Reduce 的索引分类存储给搜索引擎查询系统带来的查询优势，实验还在索引分类和未分类存储状态两种的检索环境下对用户查准率进行实验仿真. 在索引分类存储检索系统中，权重阈值取值 λ=0.8. 实验对象为用户 10 次查询词，每次搜索返回给用户的检索结果取前 15 项，实验结果如图 5 所示. 从图中可以明显地看出，基于分类存储的检索环境的查询效果明显优于未查询状态. 其主要原因是在未分类状态下检索数据量很庞大，检索后返回的结果可能也会很多，由于用户一般只会对检索结果的第 1 页和第 2 页有兴趣，而用户希望得到的数据不一定在这 2 页中显示. 分类存储不仅减少了检索的计算量，而且返回的结果更加有针对性和目标性. 所以在 2 个检索环境下取每次搜索结果的前 20 项，后者更能在减少计算量的情况下提高用户满意度.

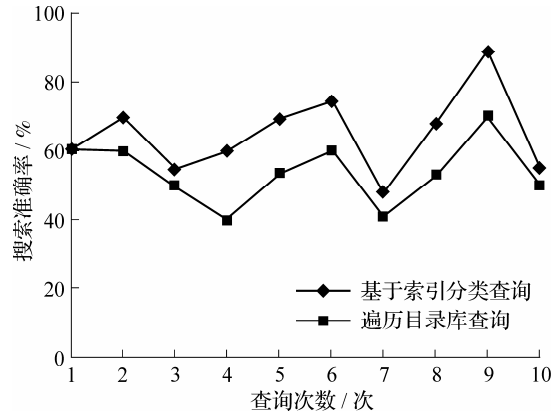

图 5 基于分类存储的检索准确率仿真图

4 结论

以云计算关键技术 Map/Reduce 为基础，设计了基于 Map/Reduce 的海量数据云存储模型和分类存储算法. 对经过预处理的正文文档进行处理，建立索引库，通过计算索引词的二次权重，对数据进

行分类, 最后分散在服务器的数据节点存储. 该模型提高了系统数据存储能力和计算速度. 但由于该方法只是基于索引在各文档的出现频率作为主要分类依据, 所以存在文档索引分类效果还存在不均匀的问题. 在试验测试中, 体育类的分类准确率很高, 而自然类的分类效果比较差. 如果要进一步提升分类整个文档库的分类准确率, 还应该对索引进行语义分析, 这也是我们需要今后对该课题继续研究的方向.

参考文献:

[1] Barroso L A, Dean J, Hölzle U. Web search for a planet: The google cluster architecture[J]. IEEE Computer Society, 2003(3):23-28.

[2] Dean J, Ghemawat S. Map/Reduce advantages over parallel databases include storage-system independence and fine-grain fault tolerance for large jobs[J]. Communications of the ACM, 2010, 53(1):72-77.

[3] Liu Hua. Words clustering based on keywords indexing from large-scale categorization corpora[J]. International Conference on Information Assurance and Security, 2009 (5):407-410.

[4] 张建梁. 基于云计算的语义搜索引擎研究[M]. 上海: 复旦大学, 2009:15-28.

[5] Richard M, McCreadie C, Macdonald C, et al. Comparing distributed indexing: To Map/Reduce or not [M]. Boston: LSDS-IR Workshop, 2009.

[6] 吴国贵, 丁振国. 基于 Map/Reduce 的分布式搜索引擎研究[J]. 知识组织与知识管理, 2007, 8(10):52-55.

[7] 蒋建洪. 主要分布式搜索引擎技术的研究[J]. 科学技术与工程, 2007(10):2419-2414.

[8] 孙瑞锋, 赵政文. 基于云计算的资源调度策略[J]. 航空计算技术, 40(3):103-105.

[9] 马云涛. 网络文件存储和共享系统的资源搜索研究和实现[D]. 南京: 东南大学, 2004:34-45.

[10] Dean J, Ghemawat S. Map/Reduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1):107-113.

[11] Gates A F, Natkovich O. Building a highlevel dataflow system on top of Map/Reduce[J]. Lyon, 2009(7):1414-1425.

[12] Yang H C, Dasdan A, Hsiao R L, et al. Simplified relational data processing on large clusters[J]. SIGMOD, 2007(6):1029-1040.

Study on Cloud Storage Model of Map/Reduce-based Index Data

LU Xiao-li[1,2], HE Jia-ming[1,2*]

(1.Institute of Communication, Ningbo University, Ningbo 315211, China; 2.Key Laboratory of Mobile Internet Application Technology of Zhejiang Province, Ningbo 315211, China)

Abstract: The main problems of current search engine system applied on intelligent terminals are limited storage capacity with massive data, high-concurrency access of users and search delay of system. Aiming to tackling these problems, this paper proposes a cloud storage model of index classification and adopts a new index storage algorithm based on Map/Reduce programming model. The algorithm calculates the secondary weight of index term in the process of index classification in order to lower the fuzzy granularity of the classification. Based on the experimental results, the proposed storage model can not only improve the mass data processing efficiency, but also to some extent reduce query delay and ameliorate the search efficiency.

Key words: search engine; index classification; data storage; Map/Reduce

（责任编辑　章践立）