一种基于对等网络的云资源多属性区间查询算法*

李 璞, 陈世平
(上海理工大学 光电信息与计算机工程学院, 上海 200093)

摘 要: 设计了 n 元属性组来描述云资源,并为属性组中的每个属性都划分区间。为解决云资源的多关键字高效查找问题,对不同属性的不同区间的任意组合都建立索引。针对云资源属性变动时导致索引更新时网络开销太大的缺点,提出依据索引中属性的个数对全部索引进行归类存储。仿真实验表明,在云资源的属性发生变动时,该算法在更新索引时在网络中产生的信息个数是一个常数 n,数目远远小于其他的多关键字区间查询算法,查找资源时网络开销不仅小而且稳定。

关键词: 云资源; 多属性; 区间查询; 索引; 归类
中图分类号: TP393 **文献标志码**: A **文章编号**: 1001-3695(2013)09-2831-04
doi:10.3969/j.issn.1001-3695.2013.09.069

Cloud resources multi-attribute range-query algorithm based on peer-to-peer network

LI Pu, CHEN Shi-ping
(*School of Optical-Electrical & Computer Engineering*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: This paper used attribute groups to describe cloud resources, and divided each attribute into intervals in the attribute group. To solve the multi-keyword cloud resources efficiently searching problem, this paper constructed indexing for the any combination of the different properties and different intervals. For the shortcoming of index updating while cloud resource properties change causing too much network overhead, this paper classified all indexes' storage by the number of attributes in the index. Simulation results show that, when certain attribute changes in the properties of the cloud resources, the information generated in the network by the proposed algorithm for updating the index is a constant n, which is far less than other multi-keyword range query algorithm, and network overhead is small and stable in resources searching.

Key words: cloud resources; multi-attribute; range-query; index; classify

0 引言

大规模分布式系统中的复杂查询处理是将对等计算技术运用于云计算中的重要问题, 是学术界与工业界所共同关注的研究问题。本文介绍了一种高效、通用的基于 Chord 协议的多属性区间查询处理,既支持匹配查询也支持范围查询。与现有其他技术相比, 对于任何数据元组, 该算法只需要对其编码和索引一次, 且能将查询处理的代价限制在一个很小的范围内。

云对等网络的拓扑结构特点和传统对等网络相比有很大不同:云对等网络是由云服务器(一般是小型机、x86 服务器或一些其他的专业云机器)构成的,且提供云计算的云服务器的状态较为稳定,不会频繁上下线,故云资源描述信息可以相对稳定地存储于某节点。

在云对等网络中,可用的云资源是动态变化的,每个节点存储于异地的资源描述信息需要不断地被更新,这可能造成巨大的网络通信开销。为此需要研究如何设计新的分布式多维信息存储方式,如何快速有效地更新信息,以及如何减少更新开销。本文对多维度云资源采用的是区间查询。区间查询既适合单个匹配,又适合区间匹配,还可以减少多维度云资源描述信息的组合个数,从而降低资源信息更新开销。

1 相关工作

1.1 多属性区间查询典型解决方法

方法 1 对每一属性都进行区间划分。例如对内存的划分可以是(0 GB,4 GB],(4 GB,16 GB],(16 GB,64 GB],(64 GB,∞];而对响应延迟的划分是(0 ms,10 ms],(10 ms,20 ms],(20 ms,30 ms],…。如果某个节点提供的虚拟机可支持 20 GB 的内存 ,它的响应延迟是 25 ms,那么,该节点信息将会存储在由字符串"内存(16 GB,64 GB]、延迟(20 ms,30 ms]"所映射到的责任节点上。

该方法的缺点是在多维空间里的区间组合个数可能非常多,造成巨大的更新开销。

方法 2 只选定其中一个属性用做索引。如果选定响应延迟作为索引,那么提供 20 GB 内存和 25 ms 延迟虚拟机的节点,它的信息将会存在由延迟(20 ms,30 ms]所决定的责任节点上,而用户查询只对 (0 ms,10 ms],(10 ms,20 ms],(20 ms,30 ms] 三个区间进行。

收稿日期: 2012-12-24; **修回日期**: 2013-02-15 **基金项目**: 国家自然科学基金资助项目(61170277);上海市教委科研创新重点资助项目(12zz137)

作者简介:李璞(1986-),男,江苏淮安人,硕士研究生,主要研究方向为计算机网络、云计算(534046717@qq.com);陈世平(1964-),男,浙江绍兴人,教授,博导,主要研究方向为计算机网络、分布式计算、云计算.

该方法的缺点是当进行查询时,所有延迟在这个区间内的节点,无论它的内存是何值,都会被返回。这使得处理和传输这些结果的开销很大。

方法 3 把上面两种思路结合起来。假设云节点资源信息共有 n 个属性,可以选定其中 $k(k<n)$ 个属性作为索引。所有 n 个属性的节点信息都要被存储,但存储的地点由选定的 k 个属性来决定。每个用户查询将被分解成一组 k 个区间的组合,然后对每个组合分别进行查询。如果每个查询的开销非常少,那么总的开销将不会很大。

该方法的缺点是不能够索引全部属性。

1.2 多属性区间查询研究现状

目前国内外有很多针对结构化 P2P 上多属性区间查询的研究。这些研究工作的目标都是减少解析每个多属性区间查询所需的跳数 H 和所产生的消息数 M,并且同时减少更新某个属性值所产生的消息数 U。假设 n 表示节点个数,m 表示发布属性的个数,r 表示查询区间的大小。目前关于结构化 P2P 上多属性区间查询最好的研究结果是 $H=O(\log n)$、$M=O(\log n)+O(r)$ 和 $U=O(m\log n)$。

基于结构化 P2P 实现多属性区间查询的系统从支持的属性个数划分,可分为两种:a)能支持的属性事先固定且个数比较少,具有代表性的系统有 MAAN[1]、Armada[2]、Mercury[3];b)能支持的属性无须事先固定,允许在系统运行时加入新的属性,代表性的系统有 SWORD。在 MAAN、SWORD 和 Mercury 中,更新某个资源的一个属性值所产生的平均消息复杂度为 $O(m\log n)$。其中,n 是节点个数,m 为此资源被注册的属性个数。此外,在 MAAN、SWORD 和 Mercury 中,解析每个查询所需的跳数和所产生的消息数依赖于节点个数和被查询的区间大小。

可见这些研究并不能够满足云对等网络在常数跳内高效查询的实际需要。

2 BMR:基于多属性区间的云资源存储与检索

P2P 网络对于区间查询的支持,已经被视为一项重要的开放性的研究课题。研究者们所做的一部分努力集中于尝试改善已有的基于 DHT 技术的结构,以使其支持区间查询。这些努力所要面临的最主要问题是随机的哈希函数,这是 DHT 类结构用来使不同节点间达到分布式负载平衡的方法,但不利于区间查询的实现,这是因为在对数值使用哈希函数计算后,将会使得同一区域内的数值间不再有任何的关系。本算法从另外一个思路出发来实现多属性区间查询。

2.1 云资源的定义

本算法将网络中提供云服务的节点称为云资源,下面给出云资源以及其查询的形式化描述。

定义 1 用 SHA-1 算法为每台云服务器产生一个唯一的节点编号(nodeID),该编号作为网络拓扑结构、云资源检索及索引存储的共同标志符。

定义 2 云资源是用 n 元组$(A_1,\cdots,A_i,\cdots,A_n)$来描述,其中 A_i 代表云资源描述的某一特征的属性。

定义 3 根据已定义的云资源描述规范,特征属性 A_i 描述为形如$(a_{1i},\cdots,a_{ii},\cdots,a_{mi})^{\mathrm{T}}$的 m 元组,其中,a_{ii}为特征属性 A_i 的某段区间。

定义 4 某个云节点的资源描述信息表示为$(a_{k1},\cdots,a_{ki},\cdots,a_{kn})$,其中$1\leqslant k\leqslant m$,云节点的资源描述信息存储于 nodeID$\geqslant H($classify$(a_{k1},\cdots,a_{ki},\cdots,a_{kn}))$第一个节点上。

定义 5 classify$(a_{k1},\cdots,a_{ki},\cdots,a_{kn})$用来生成归类码,该归类码用于归类多维资源描述信息。在标志符决定的节点上,云资源的索引以(key,value)的形式存储,其中 key 是描述云资源的 n 元组,而 value 是云资源的集合。

2.2 基本思想与方法

由云资源的定义 2 和 3 可以推出,如果使用区间属性来为云资源建立索引,那么组合而成的索引个数是确定的,这样的话便可以根据索引的分布规律来对索引进行归类。

本算法的基本思想是:对云资源描述的每个关键字按照其值域划分区间,并使用 n 元组来描述云资源,然后对 n 元组先归类,再 hash,最后云节点的资源索引信息存储于 nodeID$\geqslant H($classify$(a_{k1},\cdots,a_{ki},\cdots,a_{kn}))$的第一个节点上。

云资源多属性区间检索可以用 query_nodes( )、classify ( )和 map( )三个函数表示。其中:classify 用于对多属性区间索引归类并产生一个归类码值;query_nodes 用来获取所有满足条件的节点;而 map 是用来选择一个最佳节点。假设给定一个查询 Q(用 n 元组表示)和目标节点 obj,云对等网络的查询可以表述为 obj = map(query_ nodes(H(classify(Q)),Q)),其中 H( )是 hash 算法。本文所要介绍的 classify( ),即如何对多属性区间索引进行归类并生成归类码,然后进行归类存储。

2.3 云资源索引的数据结构

在云对等网络中,每个云资源属性的区间个数是根据实际需要划分的,各属性区间的个数不一定相等。在实际应用中,为了便于云资源的存储与检索,可以对云资源属性的区间进行编码,具体的编码方式为:假设实际中某个属性被划分为 m 个区间,为这个属性的区间分配 $m+1$ 个码值,依次为 0、1、2、…、m,其中 0 表示这个属性未用于索引。编码后,任意属性的任意区间的组合都可以用来建立索引,有利于更加全面地检索云资源。

在云节点上存储若干(key,value)形式的索引对,把 n 元组$(a_1,\cdots,a_i,\cdots,a_n)$作为 key,nodeID 集合作为 value,这样的许多(key,value)对便构成了云资源索引。假设有一个查询,内存的区间码为 1,CPU 的区间码为 2,这样对这个内存和 CPU 的查询 Q 可以表示为 Q:(0,1,2,…,0)。查询请求节点将该查询发送到目标节点上,目标节点根据 Q 找到相应的索引,把所有满足条件的资源标志发给请求节点。

2.4 基于属性个数的云资源索引归类存储

对于 1.1 节中提到的方法 1,其优点在于查询快且比较准确,而客户端对查询结果的处理开销较小;缺点在于当资源状态发生变化时,发送消息数量庞大,也就是信息更新导致网络开销太大。之所以更新的开销太大,是因为当资源一个属性的状态发生变化,与其组合而成的全部索引都要更新。

若要是将这些索引进行归类存储的话,便可以大大地减少更新开销。本文提出了按照索引中属性的个数对索引进行归类。归类后,云节点的索引信息存储于 nodeID$\geqslant H($classify$(a_{k1},\cdots,a_{ki},\cdots,a_{kn}))$的第一个节点上。归类码的生成可以自己设计算法实现,但要保证不同个数的属性组合所产生类别的类别码不可相同。归类码实际上就是根据云资源描述 n 元组中属性区间码不为 0 的区间码个数来生成,因此只会有 n 个码值。

如图 1 所示,对 n 元组(a_1,…,a_i,…,a_n)进行归类后将产生一个归类码 K,然后对这个 K 使用 Chord 的哈希算法得到一个标志 kid,再调用 Chord 算法中的 findSuccessor(kid)找到该索引的存储节点。在目标节点上存储该索引时,把 n 元组(a_1,…,a_i,…,a_n)作为 key,把无数可以用该 n 元组描述的云节点的 nodeID 的集合作为 value,这样的许多(key,value)对便构成了存储在目标节点上的云资源索引。

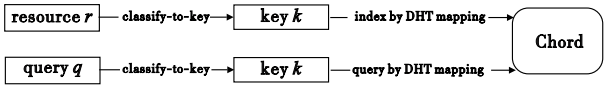

图1　资源索引创建与查询处理

云资源索引的归类存储将会大大减少信息更新所导致的巨大网络开销,具体见下节的详细分析。

2.5　云资源的查找与更新

对云资源进行查找时,首先要把查询条件转换成 n 元组表达式,然后根据这个表达式生成归类码,最后将 n 元组表达式发送给由归类码决定的目标节点。目标节点在找到符合条件的资源标志后,将其发给请求节点。

云资源的状态发生改变后,需要做两件事情:a)将旧的状态描述信息从索引中删除;b)将新的状态描述信息添加到索引中。由于本算法把索引进行了归类存储,故云资源的更新信息也可以进行归类。

当某一云节点的某个属性发生变化时,相应地要去更新索引。根据索引的归类规则,若索引只由这一个属性构成时,发往该索引所在节点的更新信息包中包含两条更新信息:一条用于删除旧索引中的云资源信息,另外一条用于添加云资源信息到其他索引中。若索引由两个属性组合而成时,因为云资源由 n 个属性描述,所以有 $n-1$ 种组合,而每种都包含两条更新信息,因此发往该类索引所在节点的更新信息数据包中的更新信息条数为 $2\times(n-1)$。

假设云资源有 n 个属性,若按照属性个数归为 n 类,当某个云资源中的某个属性发生变化时,将把更新信息发送给 n 个节点;若不进行归类,则要将更新信息发送给 $(1+C_{n-1}^{1}+C_{n-1}^{2}+\cdots+C_{n-1}^{n-1})\times 2$ 个节点。经过比较可以发现,归类后信息更新开销将大大减少。

依此类推,若云资源的 n 个属性都发生变化,归类后在网络中发送信息的条数依旧为 n;若不归类,在网络中发送信息的条数为 $(1+C_{n-1}^{1}+C_{n-1}^{2}+\cdots+C_{n-1}^{n-1})\times 2n$。由上述推理可以得出,某个云资源状态发生变化时,归类后,发送的更新消息数是非常少的。

索引归类的方法大大减少了网络中传送的消息数,同时也使得目标节点的处理开销增加、负载增加,但从整个系统性能提升的角度上来说,这点牺牲是微不足道的。

3　实验与评价

本文在 PeerSim 上用 Java 实现了一个模拟器,实验的机器配置为两个 AMD Opteron 2.2 GHz CPU、3 GB 内存、操作系统为 Linux 2.4.21。为体现本文 BMR 算法的优越性,选择基于结构化 P2P 的多属性区间查询的代表算法 SWORD 与其对比。在模拟平台上每个云资源使用 100 个属性来描述。

3.1　实验一

第一次实验　设置网络中的节点数为 512 个,依次变动某个云资源的 1、3、5、…、23 个属性,统计网络中产生的消息数目。在模拟实验中,在变动属性个数相同的情况下,随机选择属性并进行随机变动,反复 100 次,统计出消息总数取平均值。

第二次实验　设置网络中的节点数为 4 096 个,其他条件及实验次数同第一次实验。

两次实验结果数据如表 1 所示。

表 1　实验结果

更新属性个数	第一次实验		第二次实验	
	SWORD	BMR	SWORD	BMR
1	912	100	1233	100
3	2713	100	3678	100
5	4576	100	6053	100
7	6344	100	8409	100
9	8208	100	10833	100
11	9927	100	13275	100
13	11756	100	15647	100
15	13554	100	18009	100
17	15328	100	20490	100
19	17114	100	22867	100
21	18902	100	25235	100
23	20779	100	27668	100

从以上的两个实验结果对比可以得出,BMR 算法在属性变化时发送消息的数量与网络规模、发生变化属性的个数无关,而 SWORD 算法在属性发生变化时发送消息的数量与网络规模、发生变化属性的个数正相关。

3.2　实验二

第一次实验　在模拟器上模拟 10 000 个节点的云对等网络。在此规模的网络下,依次随机生成 1 000、1 500、…、5 000 个查询来比对两种查询的平均带宽消耗(B/S),重复 100 次取平均值。

第二次实验　在模拟器上模拟 100 000 个节点的云对等网络。其他的实验内容同第一次实验。

两次实验结果的数据如表 2 所示。

表 2　实验结果

查询个数	第一次实验		第二次实验	
	SWORD	BMR	SWORD	BMR
1000	2.16	3.12	2.29	16.12
1500	2.31	8.13	2.03	11.45
2000	2.07	4.27	2.13	19.38
2500	2.22	4.48	2.21	13.54
3000	2.68	6.78	2.18	17.46
3500	2.1	7.64	2.27	16.98
4000	2.14	5.23	2.29	18.36
4500	2.03	8.95	2.33	13.78
5000	2.39	3.88	2.2	15.67

对比实验数据发现,BMR 算法每个查询的平均带宽消耗比较稳定且小于 SWORD 算法;而 SWORD 算法每个查询的平均带宽消耗与网络规模正相关,在相同的网络规模下,不同查询的带宽消耗不同,说明 SWORD 算法的带宽消耗还与查询内容相关。

4 结束语

本文首先阐述了在大规模分布式系统中处理复杂查询是将对等计算技术运用于云计算中的重要问题,然后结合云对等网络的特点,提出使用区间查询,并解释了区间查询在云计算中的优势。在比较了三种多属性区间查询的典型方法后,又分析了多属性区间查询的研究现状;接着给出了云资源的定义,简述了本算法的基本思想与方法。本算法提出用 n 元组来为云资源建立索引,并选择用各个属性各区间的组合来建立索引。针对多属性组合索引个数太多,导致云资源信息更新的网络开销过大的缺点,本算法提出依据索引中属性的个数对全部索引进行归类存储。最后叙述了在索引分类存储的对等网络中如何查找云资源,还比较了索引未归类与归类后,云资源信息更新在网络中产生的信息数量。通过对比发现,归类后云资源信息更新在网络中产生的信息数量远远小于未归类的情况。仿真实验表明,本文算法在资源更新时产生的消息数目和查找资源时产生的网络开销都远远小于其他基于对等网络的多属性区间查询算法;更重要的是本文算法在网络开销上非常稳定,满足云对等网络实际需要。

参考文献:

[1] CAI Min, FRANK M, CHEN Jin-bo, *et al*. MAAN: a multi-attribute addressable network for grid information services[C]//Proc of the 4th International Workshop on Grid Computing. 2003:3-14.

[2] LI Dong-sheng, CAO Jian-nong, LU Xi-cheng, *et al*. Delay bounded range queries in DHT-based peer-to-peer systems[C]//Proc of the 26th IEEE International Conference on Distributed Computing Systems. Washington DC:IEEE Computer Society,2006:184-191.

[3] BHARAMBE A R, AGRAWAL M, SESHAN S. Mercury: supporting scalable multi-attribute range queries[C]//Proc of Conference on Community Architectures, Protocols & Applications. New York:ACM Press,2004:353-366.

[4] 王必晴,贺鹏. H-Chord:基于层次划分的 Chord 路由模型及算法实现[J]. 计算机工程与应用,2007,43(36):141-143.

[5] 段世惠,王劲林. 基于有限范围组播的 Chord 路由算法[J]. 计算机应用,2009,29(2):514-517.

[6] OPPENHEIMER D, ALBRECHT J, PATTERSON D, *et al*. Distributed resource discovery on planetlab with sword[C]//Proc of the 1st Workshop on Real, Large Distributed Systems. New York: ACM Press,2004.

[7] KLEIS M, LUA E K, ZHOU X. Hierarchical peer-to-peer networks using lightweight super peer topologies[C]//Proc of the 10th Symposium on Computers and Communications. Washington DC: IEEE Computer Society,2005:143-148.

[8] SHEN H, CHEN G. Cycloid: a constant-degree and lookup-efficient P2P overlay network[C]//Proc of the 18th International Parallel and Distributed Processing Symposium. Washington DC:IEEE Computer Society,2004:1-10.

[9] GUPTA I,BIRMAN K,LNGA P,*et al*. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead[C]//Lecture Notes in Computer Science,Vol 2735. New York: ACM Press, 2003:160-169.

[10] GUPTA A,LISKOV B,RODRIGUES R. One-hop lookups for peer-to-peer overlays[C]//Proc of the 9th Conference on Hot Topics in Operating Systems. Berkeley:USENIX Association,2003:2.

[11] XU Ke, SONG Mei-na, ZHANG Xiao-qi, *et al*. A cloud computing platform based on P2P[C]// Proc of IEEE International Symposium on IT in Medicine & Education. Washington DC:IEEE Computer Society,2009:427-432.

[12] ZHAO Peng, HUANG Ting-lei, LIU Cai-xia, *et al*. Research of P2P architecture based on cloud computing[C]//Proc of International Conference on Intelligent Computing and Integrated Systems. Washington DC:IEEE Computer Society,2010:652-655.

[13] SOTIRIADIS S,BESSIS N,ANTONOPOULOS N. Using self-led critical friend topology based on P2P Chord algorithm for node localization within cloud communities[C]//Proc of International Conference on Intelligent Computing and Integrated Systems. Washington DC:IEEE Computer Society,2011:490-495.

[14] HUANG Li-can. Semantic P2P networks: future architecture of cloud computing[C]//Proc of the 2nd International Conference on Networking and Distributed Computing. Washington DC:IEEE Computer Society,2011:336-339.

(上接第 2822 页)

[3] LIU Yang, TIAN Ye, SHEN Shuo, *et al*. A compatible and equitable resolution service for IoT resource management[C]//Proc of the 6th IEEE International Conference on RFID. 2012.

[4] COHEN E, KAPLAN H. Proactive caching of DNS records: addressing a performance bottleneck[C]//Proc of Symposium on Applications and the Internet. 2001:85-94.

[5] JANG B, LEE D, CHON K, *et al*. DNS resolution with renewal using piggyback[J]. Journal of Communication and Networks, 2009,11(4):416-427.

[6] JUNG J, SIT E, BALAKRISHNAN H, *et al*. DNS performance and the effectiveness of caching[J]. Internet Measurement Workshop, 2002,10(5):589-603.

[7] BHATTI S N, ATKINSON R. Reducing DNS caching[C]//Proc of IEEE Conference on Computer Communications Workshops. 2011: 792-797.

[8] CHEN Xin, WANG Hai-ning, ZHANG Xiao-dong. Maintaining strong cache consistency for the domain name system[J]. IEEE Trans on Knowledge and Data Engineering,2007,19(8):1057-1071.

[9] YANG Qiang, ZHANG H H, LI Tian-yi. Mining Web logs for prediction models in WWW caching and prefetching[C]//Proc of the 7th ACM SIGKOD International Conference on Knowledge Discovery and Data Mining. 2001:473-478.

[10] YANG Qiang, ZHANG H H. Web-Log mining for predictive Web caching[J]. IEEE Trans on Knowledge and Data Engineering, 2003,15(4):1050-1053.

[11] BONCHI F, GIANNOTTI F, MANCO G, *et al*. Data mining for intelligent Web caching[C]//Proc of International Symposium on Information Technology. 2001:599-603.

[12] FU A Y, WU Xiao, FU Hao-huan. A statistics-based Web cache prediction model[C]//Proc of the 6th ACM-HK Postgraduate Research. 2005.

[13] CHERKASOVA L. Improving WWW proxies performance with greedy-dual-size-frequency caching policy, R.1[R]. [S.l.]:HP Laboratories,1998.

[14] HAN Jia-wei, PEI Jian, YIN Yi-wen. Mining frequent patterns without candidate generation[C]//Proc of ACM SIGMOD International Conference on Management of Data. 2000:1-12.