doi:10.3772/j.issn.1002-0470.2013.12.001

分布式结构化 P2P 网络下局部敏感哈希快速检索的负载均衡①

齐向东②* 刘大伟** *** 王劲林* **

(* 中国科学院声学所 国家网络新媒体工程技术研究中心 北京 100190)
(** 中国科学技术大学网络传播系统与控制联合实验室 合肥 230027)
(*** 中国科学院计算技术研究所烟台分所 烟台中科网络技术研究所 烟台 264670)

摘 要 研究了分布式哈希表(DHT)结构化 P2P 网络下,采用局部敏感哈希(LSH)方法进行相似检索时的负载均衡问题。考虑到 LSH 方法在高维空间下可以有效地进行 K 近邻检索,近年来 LSH 逐渐扩展到 DHT 分布式 P2P 网络下处理分布式相似检索问题,提出了一种采用虚拟节点方式管理多维度 LSH 桶空间的方法,将服从特定分布的多维 LSH 桶空间映射到 DHT 命名空间,以更好的负载均衡效果降低分布式环境下快速检索的性能损耗,优化查询效率。进而,以 Chord 结构为例,提出了基于虚拟节点的负载均衡具体算法。与其他方法相比,该方法能有效地改善节点负载均衡。通过实验验证了该方法的有效性。

关键词 负载均衡,分布式哈希表(DHT),局部敏感哈希(LSH),虚节点,分布式相似检索

0 引 言

相似检索(similarity search)问题是很多应用中的经典问题,尤其是在高维空间。随着信息的快速增长,人们在文本文档到多媒体内容范围内提出了一系列用于高维数据的相似检索方法,其中局部敏感哈希(locality sensitive hashing,LSH)[1-3]是典型的基于哈希的近似检索方法,它可以将相似的对象以很高的概率放在哈希表中同一个数据桶中。在高维空间内,LSH 的效果优于其他基于树的方法,如 KD 树等。

分布式哈希表(distributed hash table,DHT)是一种常见的结构化对等(Peer-to-Peer,P2P)网络。人们提出了一系列基于 DHT 的 P2P 覆盖层结构,如 Chord[4],CAN,Pastry 和 Tapestry。Chord 采用一致性哈希算法管理 Chord 节点,达到良好的扩展性和查询效率。然而,对象均匀放置的策略使得 Chord 无法处理相似检索这类复杂请求,后者要求相似数据放在靠近位置。近来人们提出了一些处理结构化 P2P 覆盖层网络的相似检索方法[5-7]。文献[8]将图像检索中的 Bag-of-Feature 方法和 DHT 相结合,以增强检索系统的可扩展性。文献[9]以 DHT-P2P 的分布式结构管理文本文档的特征,并结合 LSH 检索方式实现快速查询。这些模式设计了不同的索引结构,用于支持 P2P 网络中的复杂近似检索,但缺乏结合后系统本身负载均衡性能的讨论。当 Chord 假定对象是均匀分布的,整个覆盖层网络实际负载通常并不均衡。文献[10,11]提出了几种负载均衡模式,用于处理 DHT 网络中的负载均衡问题。如 Rao 在文献[10]中采用虚拟节点的方法,用于平衡节点之间的负载。本文主要考虑基于 DHT 分布式相似检索系统,将 LSH 映射到 DHT 标示空间,处理分布式相似检索。我们将讨论现有方法中的映射方法,并提出采用虚拟节点基于 Chord 结构分布式检索结构和维持算法,用于保证节点之间的负载均衡,有效处理覆盖层网络中节点的动态变化。

1 系统概览和问题阐述

首先介绍基于 DHT 的相似检索,并展示如何在分布式 P2P 覆盖层网络中建立 LSH 索引。我们的工作是对文献[5]中的负载均衡问题进行扩展,考虑了 DHT 结构的同时也考虑不同映射方法的影响,

① 国家自然科学基金(60975045),国家科技支撑计划(2011BAH11B01)和中科院先导专项(XDA06030)资助项目。
② 男,1967 年生,博士;研究方向:海量数据存储、查询和索引;联系人,E-mail:qixd@intellix.com.cn
(收稿日期:2013-01-23)

这是之前的工作未曾考虑的。我们对分布式环境下相似检索系统如何提供有效的负载均衡的条件进行了讨论。

1.1 采用 DHT 和 LSH 的相似检索

这里我们简要介绍文献[5]中的方法。其基本思想是采用基于 p－稳定分布式[1]的局部敏感哈希方法。对于每个数据点 v,采用 k 个独立哈希函数,形式如式

$$h_{a,B}(v) = \lfloor \frac{a \cdot v + B}{W} \rfloor \tag{1}$$

所示,其中 a 是一个 d 维向量,向量元素从正态分布独立选取。$W \in IR$,B 是服从 $[0,W]$ 的均匀分布。

每个哈希函数将一个 d 维的数据点映射到整数集合,最终结果是一个长度为 k 的向量,如式

$$\boldsymbol{g}(v) = (h_{a_1B_1}(v),\cdots,h_{a_kB_k}(v)) \tag{2}$$

所示。为了处理分布式 P2P 网络中的相似检索,需要将哈希表映射到 DHT 覆盖层网络标示空间。图 1展示了 DHT 中通过 2 级映射后 LSH 索引。

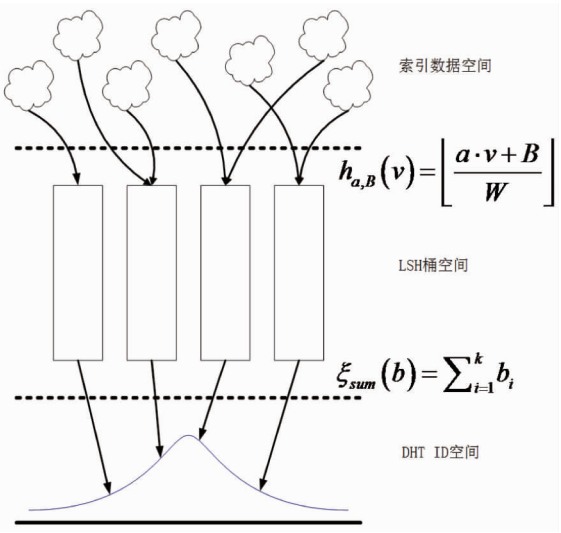

图 1 从数据空间到 DHT 命名空间的两层映射

线性映射函数 ξ 可以基于和的形式,如公式

$$\xi_{\text{sum}}(b) = \sum_{i=1}^{k} b_i \tag{3}$$

所示,其中 b 是 k 维整数向量。b 中的每个分量表示一个数据桶标签。

注意到基于和的映射函数对各个桶标签平等对待,数据桶将相似的数据指派给同一个节点。如文献[1]证明,给定所有数据点 $v_1,\cdots,v_M$,经过二级映射后,结果服从正态分布

$$N\left(\frac{k}{2},\frac{\sqrt{k\sum_i \| v_i \|_2^2}}{W}\right) \tag{4}$$

上述分布式相似检索系统使得 K 最近邻(K-nearest neighbor,KNN)搜索基于分布式 LSH 索引,不同于集中式的配置方式。同时,生成的分布式索引结构负载可预计服从正态分布,具有一定的特殊属性,从而对其在 DHT-P2P 网络的分配和处理方面,提出了新的问题。基于上述特性,我们提出了这类分布式相似检索系统的负载均衡问题。

1.2 负载均衡问题

如之前讨论,由映射函数(式(3))产生的数值服从正态分布。然而,基于 DHT 覆盖层网络的节点标示服从均匀分布。原始 Chord 协议中[5],一致性哈希用于指派对象到各个 Chord 节点上。每个节点收到几乎相同数量的对象。在我们的分布式相似检索系统内,将对象(这里是指 LSH 数据桶标签)映射到某一个节点上,不是采用一致性哈希方法,而是如图 1 所示的二级映射方式。

为了在整个相似检索系统中提供较好的负载均衡,人们针对 LSH 索引提出了不同的结构和算法。这些结构通常服从均匀分布,映射到线性均匀节点 ID 空间。事先在指定位置部署一些网关节点的方式,在系统初始时候有效。与此方法不同,我们考虑采用结构化 DHT 网络,在文献[5]中视为黑箱系统。在下一节中,我们将展示 Chord 结构中如何使用虚拟节点和维持算法保证节点之间的负载均衡。

2 结构和算法

本节中,我们描述 Chord 结构和覆盖层 DHT 的维持方法。我们的算法受文献[10]中的负载均衡技术的启发,后者强调 P2P 系统中的高度异构特性。在我们的场景中,不平衡的主要因素在于 LSH 索引负载服从正态分布,和文献[10]中的不同。然而,这种模式采用虚拟节点方式,仍可对分布式相似检索系统进行调优,从而使得系统负载均衡。

2.1 结合虚拟节点的 Chord 结构

这里介绍 Chord 节点空间之上的虚拟节点空间。如图 2 所示,每个虚拟节点像是原始 DHT 空间上的单个节点。在我们的结构中,每个 Chord 节点(物理节点)可以负责多个虚拟节点。例如,每个虚拟节点负责原始 DHT 对象描述空间中一段连续区域,在我们系统中是 LSH 桶标示区域。而现在每个 Chord 节点可以承担环状对象标示空间中的不连续部分。一般而言,虚拟节点层像是额外的映射过程。在虚拟节点空间,我们采用一致性哈希获取节点

ID,并将哈希函数产生的基于和的 key 值赋给各个节点。这些 key 是服从正态分布的 LSH 桶标签,相关的映射函数在 1.1 节介绍。

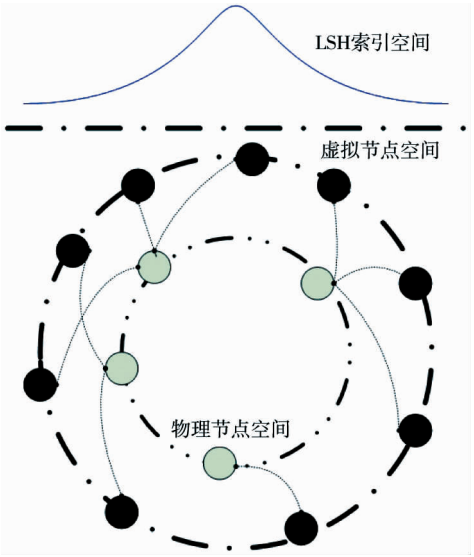

图 2　基于 Chord 的两层结构:虚拟节点和物理节点

2.2　负载均衡维持条件

我们的维持方法主要基于采用 LSH 的分布式相似检索系统。方法的优势在于,虚拟节点的负载分布是可以预测的。同时我们需要遵循另一个条件:映射过程给数据桶中相似的数据分配相同的 keyID。这里,用于相似检索的局部敏感属性不会受维持算法的影响。这些特性使得我们的算法不同于其他 DHT 网络下的负载均衡技术。我们在原始 Chord 协议上对负载进行传递,以平衡不同节点的负载。每个节点维持小部分负载和节点信息,但不会对离散化和可扩展性造成影响。因此,在动态环境下额外的维持代价是轻量级的。

2.3　负载定义和节点分类

对于每个物理 Chord 节点 n_i ,负载 L_i 表示所有虚拟节点 n_i 的和函数,其表达式为

$$L_i = \sum_{j=1}^{m_i} l_j \tag{5}$$

其中 l_i 表示虚拟节点 v_i 的负载,即存储在虚拟节点 v_i 上所有 LSH 桶标签元素之和。

节点 n_i 负责虚拟节点集合 $\{v_1,\cdots,v_{m_i}\}$,其中 m_i 表示虚拟节点数量。

如 1.1 节讨论,通过式(3)基于和的映射函数,我们可以预测输出 ξ_{sum} 的分布如式(4),估计均值为 $k/2$ 。标准正态分布有个很好的特性,被称为 68 - 99.7 规则,这里我们可以利用这一特性。因此,所有索引的负载情况可以被预测。考虑到虚拟节点数目,我们可以向各个虚拟节点分配平均负载阈值 T_{load} , T_{load} 由参数 k,W 和 $\{v_i\}$ 决定。假定所有物理节点是同构的(这里不考虑节点异构的情况),负载阈值 T_{load} 在整个 DHT 网络中是一个全局变量。

获得虚拟节点 v_i 负载 l_i ,和全局负载阈值 T_{load} ,我们可以对物理节点,即系统中的 Chord 节点,根据每个 Chord 节点 i 的负载情况 L_i 划分成几个不同的等级。这里,和文献[1]相似,我们简单划分为两类,即轻负载和重负载。当满足 $L_i > T_{\text{load}}$ 时,认为节点负载较重,反之则负载较轻。

我们的目标是通过对负载从重载节点向轻载节点转移,事先各个节点的负载均衡。考虑到我们的系统用于分布式相似检索,和虚拟节点负载直接相关的是 LSH 桶标签,我们假定负载均衡算法中,所有虚拟节点的负载为最小负载单元,不会进一步分割。

2.4　数据结构

为了具体化 Chord 协议,我们定义了几种数据结构,用来记录负载和节点信息。覆盖层 Chord 环由物理节点组成,每个节点额外维持虚拟节点集合,和节点负载参数。对于 Chord 环上的虚拟节点,每个虚拟节点维持虚拟节点负载,和一个较大的后继节点列表,用于确定可分担负载的轻载节点。例如,Chord 环中一个物理节点 n, $n.L$ 表示节点 n 的负载,n. virtual_node_set 表示所负责的虚拟节点列表,虚拟节点 v 维护后继节点列表 successer_list,和负载参数 $v.l$。$v.l$ 可通过对 LSH 桶标签和元素的累加函数获得。表 1 列举了我们嵌入 Chord 中的主要数据结构。

表 1　数据结构定义

符号	定　义
. virtual_node_set	该物理节点管理的所有虚拟节点列表
. L	该物理节点负载
. successor_list	虚拟节点扩展后继节点列表,数量为 r
. l	虚拟节点负载

2.5　平衡规则

给定某一个带有高负载虚拟节点 v_k 的物理节点,其中 v_k 是重负载物理节点上负载最轻的虚拟节点。我们按照如下规则,选择一个轻负载节点转移负载。

首先,也是最重要的,所选轻载节点上的虚拟节点负责的关键字必须和节点 v_k 相近。为了测量两个虚拟节点之间距离,我们利用每个虚拟节点上的后继列表(虚拟节点也是通过 Chord 结构管理的)。当标示大的虚拟节点在标示小的虚拟节点的后继列表上时,定义两个虚拟节点相近。然后,将重负载节点上的 k 个重负载虚拟节点转移到轻负载节点上的若干个离节点 v_k 相近的虚拟节点上。需要注意的是,转移过程必须保证转移后轻负载节点负载不会过重。

为了实现分布式,每个重负载节点会周期性地扫描虚拟节点的后继节点列表,找出需要的轻负载节点。如果轻载节点满足上述条件,将在两个节点之间进行负载均衡。在运用我们的算法中,虚拟节点管理的 r 个扩展后继节点列表是我们算法的关键因素。我们可以通过增加 r 数量来提升重负载节点寻找合适的轻载节点的可能,这部分将在下一节的实验中给出说明。具体而言,每个节点 n 会执行如下算法,伪代码描述如下:

负载均衡算法主调函数

```
//called periodically
maintenance(average load Tload)
    L = update_load(n. virtual_node_set);
    if(L > = Tload)
        vk = find_vk(n. virtual_node_set);
        nlight = find_nlight(vk);
    if(find vk and vlight successfully)
        transfer(vk,nlight);
```

(1)负载更新子函数

```
//calculate and update node load
update_load(n. virtual_node_set)
    for each v in n. virtual_node_set
        L + = v. l;
```

(2)查找待传输负载子函数

```
//find the lightest virtual node vk to make n light
find_vk(n. virtual_node_set)
    for each v in n. virtual_node_set
        if(L-v. l < Tload)
            insert v to transfer_list;
        for each v in transfer_list
            return minimum(v);
```

(3)查找传输的目的节点子函数

```
//transfer load from heavy to light node
find_nlight(vk)
    for each v in vk. successor_list
        return the lightest node having v;
```

(4)传输负载子函数

```
transfer(vk,nlight)
    move the virtual node vk to physical node nlight;
    update the virtual nodes list information;
```

2.6 节点动态性和数据增长

注意到我们算法的基本思想是虚拟节点,DHT 覆盖层网络并没有修改。节点的动态性,如节点加入、退出、意外退出等情况可以通过底层的 Chord 协议管理。和原始的 Chord 维护模式相比,每个节点需要保存额外的负载情况和节点信息。

当我们考虑采用 LSH 的分布式相似检索,增量维持是关键因素。数据增长会在 LSH 桶中产生更多的元素,同时每个虚拟节点的负载也会相应改变。由于 LSH 是基于哈希的方法,随着数据增长,索引也必然随之改变,节点负载变化使得虚拟节点上产生大量更新过程。我们将在未来的工作中考虑适应负载平衡过程的索引模式。

3 仿真实验

我们对上述负载均衡算法进行了实验,以测试算法的有效性和性能。为了演示负载均衡的效果,我们假设节点最重负载为 l_{max},最轻负载为 l_{min},同时记录了数据包数量,以评估我们算法对系统带来的额外负担。表 2 展示了实验结果。算法采用了 10 个原始 Chord 环上的物理节点,并分配了 50 个虚拟节点。(实验过程中,虚拟节点的后继列表节点数量为 1)

表 2 负载均衡和网络通讯仿真结果

DHT	实验结果			
	总负载	l_{max}	l_{min}	通讯包个数
原 Chord 协议	745	255	17	29878
本文改进算法	745	134	56	33902

和文献[1]中采用原始 Chord 环,无负载均衡

DHT 覆盖层网络对比,我们的算法在分配总负载(包括数据对象和 LSH 桶标签)到相同数量的物理节点上,负载均衡方面效果更好。通过降低节点最高负载,提升节点最低负载,整个系统节点的节点负载更加趋于均衡,和原始 Chord 协议相比,带宽负担提升了约 13%。额外的数据包用于更新我们数据结构中的负载信息。当系统处于负载相对均衡状态时,我们可以延长算法运行的周期间隔时间,从而减轻通信负载。

为了理解本文算法的有效性,如第 3 节所述,我们对虚拟节点后继节点列表数 r 进行了调整。我们用 $l_{\max}$ 测量负载均衡程度。采用 1000 个负荷(由数据元素和 LSH 桶标签组成)放在 20 个物理节点和 100 个虚拟节点上,虚拟节点后继列表大小 r 变化范围从 1 到 10。图 3 展示了实验的结果。

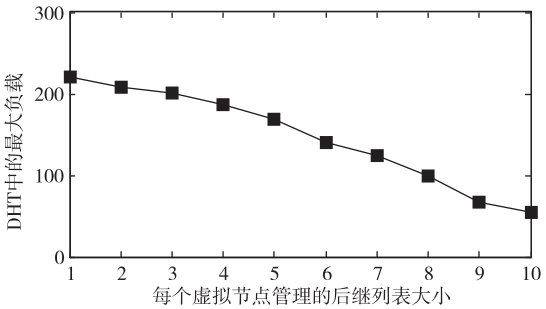

图 3　后继列表长度与最大负载关系曲线

随着 r 的增长,DHT 中的重载节点的负载降低,负载趋于均衡。这也表明 r 的增加能提高重负载节点找到合适轻载节点的概率,以便对过量负载进行转移。注意到实验过程只在某一时间段进行,如果将算法执行时间延长,将能获得更均匀的负载分布。然而,算法运行时间过长,将产生较多的通信负担,负载均衡的效果可能受到影响,因此在设计上,维持算法的时间越短越好。

4　结　论

本文给出了一种基于 DHT 的分布式相似检索系统,该系统采用 LSH 建立索引,并在 DHT 网络中引入虚拟节点,实现负载均衡机制。本文将服从特定分布的多维 LSH 桶空间映射到 DHT 命名空间,并对这一特定情况下负载均衡问题进行了讨论,并提出了一种基于 Chord 结构,结合虚拟节点的负载均衡算法。最后通过实验验证了算法的有效性。

参考文献

[1] Datar M, Immorlica N, Indyk P, et al. Locality- sensitive hashing scheme based on p-stable distributions. In: Proceedings of the 20th annual symposium on Computational Geometry, New York, 2004. 253-262

[2] Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science, 2006. 459-468

[3] Lv Q, Josephson W, Wang Z, et al. Multi-Probe lsh: Efficient indexing for high-dimensional similarity search. In: Proceedings of the International Conference on Very Large Databases, VLDB Endowment, USA, 2007. 950-961

[4] Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 2001, 31(4): 149-160

[5] Haghani P, Michel S, and Aberer K. Distributed similarity search in high dimensions using locality sensitive hashing. In: Proceedings of the 12th International Conference on Extending Database Technology, Advances in Database Technology, New York, USA, 2009. 744-755

[6] Bawa M, Condie T, Ganesan P. LSH forest: self-tuning indexes for similarity search. In: Proceedings of the 14th International Conference on World Wide Web, New York, USA, 2005. 651-660

[7] Tang Y, Zhou S, Xu J, et al. A lightweight multi-dimensional index for complex queries over DHTs. *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(12): 2046- 2054

[8] Zhang L L, Wang Z Y, Feng D G. Content-based image retrieval in P2P networks with bag-of-features. In: Proceedings of 2012 IEEE International Conference on Multimedia and Expo Workshops, Melbourne, Australia, 2012. 133-138

[9] Zhang Y, et al. Robust and efficient near-duplicate documents detection on P2P networks. In: Proceedings of 2012 IEEE Spring Congress on Engineering and Technology, Xian, China, 2012. 1-4

[10] Rao A, Lakshminarayanan K, Surana S, et al. Load balancing in structured P2P systems. Peer-to-peer Systems Ⅱ. Heidelberg: Springer Berlin. 2003. 68-79.

[11] Dabek F, Kaashoek M F, Karger D, et al. Wide-area Cooperative Storage with CFS. *ACM SIGOPS Operating Systems Review*, 2001, 35(5): 202-215

Load balance of the fast similarity search using locality sensitive hashing in structured P2P networks

Qi Xiangdong[*], Liu Dawei[**] [***], Wang Jinlin[*] [**]
([*] National Network & New Media Technology Research Center, Chinese Academy of Sciences, Beijing 100190)
([**] Joint Lab of Network Communication System & Control, University of Science and Technology of China, Hefei 230027)
([***] Institute of Network Technology, Yantai 264670)

Abstract

The load balancing and maintenance of the distributed similarity search using locality sensitive hashing(LSH) in structured P2P networks based on distributed hash table(DHT) were studied. Considering that LSH is proved efficient in K-Nearest Neighbor(KNN) search in high dimensions and a number of schemes are gradually used to implement LSH over DHT-based peer-to-peer systems to process the distributed similarity search, an efficient structure using virtual nodes to manage the multi-dimensional LSH bucket space in DHT peers was put forward to improve the searching efficiency and effectiveness in this scenario. Furthermore, a specific maintenance algorithm according to Chord structure was introduced. The algorithm can improve the load balancing in comparison with the state-of-the-art technique. The effectiveness of the proposed method was proved by experiment.

Key Words: load balance, distributed hash table(DHT), locality sensitive hashing(LSH), virtual node, distributed similarity search