软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(10):4930−4940 [doi: 10.13328/j.cnki.jos.006663]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向异构 DHT 存储的数据与位置解耦算法*

罗超然[1], 金 鑫[1], 张 颖[2], 蔡华谦[1], 柳 熠[1], 景 翔[3], 黄 罡[1]

[1](北京大学 计算机学院, 北京 100871)

[2](北京大学 软件工程国家工程研究中心, 北京 100871)

[3](北京大学 软件与微电子学院, 北京 100871)

通信作者: 黄罡, E-mail: hg@pku.edu.cn

摘　要: 分布式哈希表 (distributed hash table, DHT) 由于其高效的数据寻址方式而被广泛应用于分布式存储. 传统 DHT 必须将数据存放在指定节点中才能实现高效的数据分布式寻址, 极大地限制了 DHT 技术的应用范围. 例如, 在异构存储网络中, 节点的存储空间、带宽、稳定性等均有较大差异, 结合数据特征和节点性能差异选择合适的数据存放节点可以很大程度上提高数据的访问效率, 而传统 DHT 数据和存储位置紧耦合的特征导致其难以应用于异构的存储网络中. 针对此问题, 提出了 vRoute 算法以实现 DHT 中数据标识与其存储位置的解耦. 通过构建基于 Bloom Filter 的分布式数据索引, vRoute 算法可以在不降低数据寻址效率的基础上允许数据存储在网络中的任意节点. 通过扩展 Kademlia 算法实现了 vRoute, 并从理论上证明了 vRoute 算法的有有效性. 最后, 模拟实验表明 vRoute 以较低的存储、网络开销实现了和传统的 DHT 算法接近的数据寻址效率.

关键词: 分布式哈希表; 对等网络; 异构分布式存储

中图法分类号: TP303

中文引用格式: 罗超然, 金鑫, 张颖, 蔡华谦, 柳熠, 景翔, 黄罡. 面向异构DHT存储的数据与位置解耦算法. 软件学报, 2023, 34(10): 4930–4940. http://www.jos.org.cn/1000-9825/6663.htm

英文引用格式: Luo CR, Jin X, Zhang Y, Cai HQ, Liu Y, Jing X, Huang G. Algorithm to Decouple Data from Their Location in DHT-based Heterogeneous P2P Storage. Ruan Jian Xue Bao/Journal of Software, 2023, 34(10): 4930–4940 (in Chinese). http://www.jos.org.cn/1000-9825/6663.htm

Algorithm to Decouple Data from Their Location in DHT-based Heterogeneous P2P Storage

LUO Chao-Ran[1], JIN Xin[1], ZHANG Ying[2], CAI Hua-Qian[1], LIU Yi[1], JING Xiang[3], HUANG Gang[1]

[1](School of Computer Science, Peking University, Beijing 100871, China)

[2](National Engineering Research Center for Software Engineering, Peking University, Beijing 100871, China)

[3](School of Software & Microelectronics, Peking University, Beijing 100871, China)

Abstract: Distributed hash table (DHT) is widely used in distributed storage because of its efficient data addressing. Nevertheless, traditional DHT-based storage has to store data in specified nodes to achieve efficient data addressing, which restricts the application scope of the DHT technology severely. Taking heterogeneous storage networks for example, the storage space, bandwidth, and stability of nodes vary greatly. Choosing appropriate data storage nodes according to data features and the performance differences among the nodes can greatly improve the data access efficiency. However, the tight coupling between data and storage location disqualifies the traditional DHT-based storage from being applied to heterogeneous storage networks. Therefore, this study proposes a vRoute algorithm to decouple the data identifier from storage location in DHT-based storage. By building a distributed data index based on Bloom Filter, the vRoute algorithm allows data to be stored in any node of the network without reducing the efficiency of data addressing. It is implemented by

* 基金项目: 国家重点研发计划 (2020YFB2104100); 国家杰出青年科学基金 (61725201); 北京高等学校卓越青年科学家项目 (BJJWZYJH01201910001004)

收稿时间: 2021-07-07; 修改时间: 2021-11-25; 采用时间: 2022-02-25; jos 在线出版时间: 2023-01-18

CNKI 网络首发时间: 2023-01-19

extending the Kademlia algorithm, and its validity is verified theoretically. Finally, the simulation experiments show that vRoute achieves a data addressing efficiency close to that of the traditional DHT algorithm with low storage and network overhead.

Key words: distributed hash table (DHT); peer to peer (P2P); heterogeneous distributed storage

基于分布式哈希表 (distributed hash table, DHT) 的 P2P (peer to peer) 存储是分布式存储的重要形式之一, 不仅在传统云存储中有广泛应用[1−3], 近几年兴起的去中心化存储系统, 如 IPFS (interplanetary file system)[4]、Storj[5]等, 底层也都采用基于 DHT 的 P2P 存储来管理数据. 在 DHT 中, 每个参与节点会被分配一个唯一的标识 *nodeID* 并以<*nodeID*, *address*> 的方式在本地路由表中记录其他部分节点的信息, 节点根据本地路由表决定消息转发的下一跳节点. 通过这种方式, DHT 能够以完全去中心化的方式高效地将消息传递至全网任意目标节点. 一般而言, 基于 DHT 的 P2P 存储可以达到对数级别的节点、数据寻址复杂度[6], 仅通过 log$N$ 次的消息路由, 即可从任意节点找到目标数据所在节点, 其中 N 为网络中节点的数量.

在基于 DHT 的 P2P 存储中, 每个节点、每份数据都会被分配一个标识, 数据存储算法需要根据数据标识和节点标识的相关性将数据存储在全网范围指定的存储节点中. 数据和存储位置紧耦合的特征使得传统 DHT 技术难以应用于异构的 P2P 存储网络, 在异构的 P2P 存储网络中不同节点的延迟、带宽、存储空间、可用性均有较大差异, 而结合节点性能的差异性选择合适的数据存放节点可以提高数据的可用性、降低数据访问延迟、提升整个存储网络的数据吞吐量[7]. 此外, 存储节点的异构性还体现在节点安全性的差异上, 数据所有者可以将数据存放在自己信任的节点上以获取更好的隐私和安全保障, 而这也是传统 DHT 技术所不能支持的.

针对上述问题, 本文提出了面向基于 DHT 的异构 P2P 存储的数据和位置解耦算法 vRoute, 通过分布式的构建数据的索引, 以实现数据标识和其存储位置的解耦. 在数据索引阶段, 数据的存储节点按照 DHT 的寻址规则通过 DHT 的正向路由表向全网距离该数据标识最近的节点发送索引消息, 途径节点根据索引消息的上一跳地址构建、维护一张基于 Bloom Filter 的反向路由表. 在数据查询阶段, 节点首先沿着 DHT 的正向路由表寻址数据, 途径节点判断所查询数据的标识是否在反向路由表中已有索引, 若有则依据反向路由表继续寻址数据, 直至找到目标数据或反向路由表中再无匹配项为止. 基于此种双向寻址方式可以实现数据标识和数据存储位置的解耦, 与此同时, vRoute 算法还可以达到和传统 DHT 存储相近的对数级别的去中心化数据寻址效率, 以有效支持 DHT 在异构 P2P 网络中的应用.

本文第 1 节以 Kademlia 算法为例, 介绍传统 DHT 中的数据存储和寻址过程, 分析 DHT 中数据和其存储位置的耦合关系. 第 2 节介绍相关工作. 第 3 节介绍本文提出的 vRoute 算法. 第 4 节分析算法的有效性和理论开销. 第 5 节基于 PeerSim 模拟实验验证并评估 vRoute 算法. 第 6 节总结本文工作. 第 7 节对未来工作进行了探讨.

1 传统基于 DHT 的 P2P 存储中的数据耦合 (以 Kademlia 为例)

在基于 DHT 的 P2P 网络[8−12]中, 每个参与节点会被分配一个全网唯一的标识, 一般为定长的散列值 (hash). 节点根据自己的标识按照一定规则选择网络中符合要求的其他节点作为邻居节点, 建立连接、并将邻居节点的信息存储至本地路由表中. 在消息传递过程中, 每个节点基于 DHT 算法约定的转发规则将消息转发至邻居节点. 在 DHT 中, 给定一个目标节点标识, 从任意节点出发仅通过 log$N$ 次消息传递即可找到目标节点, 即寻址复杂度为 O(log$N$).

定义 1. P2P 网络中寻址复杂度为寻址过程中查询消息所经过的节点数量, 寻址复杂度越高, 则查询时间、流量开销越大.

以 Kademlia 算法[13]为例, 节点在加入 Kademlia 网络时会被分配一个长度为 160 b 的标识 ID_{Node}, 同时定义节点之间的距离 d 为二者标识 hash 的异或, 即:

$$d(Node_1, Node_2) = ID_1 \oplus ID_2 \tag{1}$$

每个节点本地维护一张路由表, 路由表由 160 个 K-桶 (K-bucket) 组成, 每个 K-桶中存有 K 个其他节点的路由项, 路由项以<ID_{node}, *address*>形式表示. 在发现新节点时本地节点 $Node_{\text{local}}$ 按照该节点与自己的距离将其放置在不同的 K-桶中, 其中第 i 个 K-桶中保存距离在 $[2^i, 2^{i+1})$ 范围内的节点, 若 K-桶中路由项数量已超过 K, 则替换掉最久未响应节点对应的路由项. 在收到目标标识为 ID_{target} 的节点 $Node_{\text{target}}$ 查找消息时, $Node_{\text{local}}$ 会计算自己和

$Node_{target}$ 的距离 $d(Node_{local}, Node_{target})$, 并将消息转发至对应 K-桶中距离目标最近的 α 个节点, 其中 α 为消息并行度. 根据 K-桶的构建规则, 下一跳节点 $Node_{next}$ 和 $Node_{target}$ 的距离将满足 $d(Node_{next}, Node_{target}) \leqslant d(Node_{local}, Node_{target})/2$, 从而保证在 log$N$ 跳之内消息可以从任意节点转发至目标节点. 此外还可以提高消息并行度 α 来增加节点传递消息时所选择的邻居节点数量以提高查询成功率和效率.

在基于 Kademlia 的 P2P 存储中, 数据 *data* 也会被分配一个标识 ID_{data}. 数据存放时, DHT 会根据数据 *data* 的标识 ID_{data} 选择全网与目标数据距离最近的 K 个节点, 并将 K 份数据副本分别存储于这 K 个节点中, 在数据查询时只需要找到全网 $d(data, Node)$ 最小的节点即可找到目标数据. 通过此种方式, DHT 对数据的查询请求即可视为对标识为 ID_{data} 的节点的查找, 从任意节点开始仅通过最多 log$N$ 次消息路由即可找到目标数据. 在 P2P 存储中, 节点之间的消息传递延迟是数据寻址的主要开销而 DHT 可以有效减少消息跳数, 因此基于 DHT 的 P2P 存储可以极大提高数据寻址的效率. 然而也是由于 DHT 的这种特性, 数据必须按照其标识存放在全网距离最近的节点中, 无法根据节点的存储空间、带宽和可用性的异构性来将数据存放在合适的节点上, 导致数据访问的效率往往取决于存储网络中的短板节点. 因此在基于 DHT 的异构 P2P 存储网络中, 提升数据访问效率的首要挑战就在于解耦数据标识和数据的存储位置.

2 相关工作

2.1 非结构化 P2P 的异构分布式存储

基于 DHT 的 P2P 存储采用特定算法构建节点之间的连接关系, 因而也称为结构化 P2P 存储. 与之对应的是非结构化 P2P 存储, 其节点之间的连接关系是任意的, 节点不会被分配唯一标识并可以将任何监听到的节点加入本地路由表中. 非结构化 P2P 存储对数据的存放位置也没有任何限制, 可以按照数据特征和节点特性将数据存放在合适的节点中, 因此被更多的应用于异构环境下的分布式存储[14−16]. 然而相比于结构化 P2P 存储, 非结构化 P2P 存储的主要缺陷在于数据寻址的效率. 传统非结构化 P2P 存储中的数据寻址算法, 如 random walk[17]、Gossip[18]等, 效率低且不可控, 最坏情况下寻址复杂度为 O(N), 即需要查询全网所有节点才能找到目标数据. 虽然不少现有工作针对非结构化 P2P 存储的数据寻址效率提升开展了研究[19−23], 然而这些工作一般针对特定场景优化, 且通常在最坏情况下仍然会退化为全网扫描, 在数据寻址效率上较结构化 P2P 存储仍有较大差距.

2.2 基于 DHT 的异构分布式存储

由于基于 DHT 的 P2P 存储中数据和存储位置紧耦合的特征, 现有大部分面向异构网络的 DHT 优化工作侧重于节点之间拓扑关系的优化[24], 以及在消息路由过程中的转发选择和负载均衡[25,26], 较少有工作考虑到在选择数据存放位置时节点的异构性. Hassanzadeh-Nazarabadi 等人[7]在工作中首次提出了面向 SkipGraph 的效用和位置感知 (utility- and locality-aware) 的数据存储策略, 该工作基于节点的延迟分布、可用性、带宽和存储负载构建了效用和位置模型, 并以此决定数据备份的数量以及分别存储在哪些节点中. 然而, 该工作在存储数据时需要先确认数据的存储节点再根据节点的标识生成接近的数据标识, 这并不符合通常数据使用习惯. 首先, 存储在不同节点中相同的数据备份逻辑上是同一份数据, 却被分配了不同的标识; 其次, 大部分的应用场景中数据标识取决于数据的内容而非数据存储位置, 例如在常见的基于内容寻址的数据存储中, 一般采用数据内容的 hash 作为数据的标识以降低数据冗余度, 这种模式下数据的标识数据在存放之前就已经确定.

3 vRoute 算法设计

针对基于 DHT 的 P2P 存储中数据和存储位置紧耦合的问题, 本文提出 vRoute 算法以解耦 DHT 中的数据和其存储位置. vRoute 算法允许数据存储在 P2P 网络中任意节点的同时不降低数据的寻址效率. 虽然本文基于 Kademlia 算法实现了 vRoute 算法, 但理论上 vRoute 可以基于任何 DHT 实现. vRoute 算法核心思想如下.

(1) 基于 Kademlia 算法构建节点之间连接关系, 基于 Kademlia 的寻址本文称之为“正向寻址”, 节点本地的 Kademlia 路由表为“正向路由表”.

(2) 在数据存储阶段, 确定数据存储节点后, 存储节点根据正向路由表传递所存储数据的标识, 直至数据标识

到达全网距离最近的节点, 途径节点根据消息传入节点和数据标识构建“反向路由表”.

(3) 在数据查询阶段, 从任意节点出发按照正向路由表寻址目标数据, 当在反向路由表中遇到匹配的路由项时将消息转发给匹配节点, 开始反向寻址直至找到目标数据所在的存储节点.

由于反向路由表的存在, vRoute 算法可以查询到存储在任意节点上的数据. 此外, 正向寻址路径和反向寻址路径长度最长均为 log$N$, 因此数据寻址复杂度和传统 Kademlia 一致, 且后续实验表明, 在合理的参数配置下 vRoute 算法数据寻址效率并不低于传统的 Kademlia.

3.1 反向路由表

反向路由表是数据查询时反向寻址的依据, 反向路由表中记录了节点地址以及该节点所包含的数据标识集合, 本地节点根据反向路由表判断目标数据属于哪个集合并将消息转发至下一跳节点. 为了高效记录数据集合并快速判断目标数据是否属于该集合, 本文基于 Bloom Filter 构建反向路由表. Bloom Filter 是一种结构简单而空间复杂度较小的随机数据结构[27], 它可以精确表示集合并回答类似“元素 x 是否在集合 A 中”的判定, 其存储复杂度和判定复杂度均为常数级别, 但会有一定的假阳性概率, 即对于不属于集合的元素有一定概率判定该元素属于此集合, 在后续章节中本文将证明 Bloom Filter 的假阳性不影响 vRoute 算法数据寻址的准确率.

具体而言, 反向路由表中每个路由项为一个三元组<ID_{node}, *address*, *Set*[*BF Vector*]>, 其中 ID_{node}, *address* 分别代表节点的标识和网络地址, *Set*[*BF Vector*] 表示节点所包含的数据标识集合, 由于在指定的假阳性概率下每个 Bloom Filter 位向量 (*BF Vector*) 可包含的最大元素个数有限, 故路由项中数据标识集合以 Bloom Filter 位向量集合的方式表示以支持反向路由表的动态扩容. *BF Vector* 可以根据假阳性概率以及当前位向量的状态估算已包含的元素数量. 在向指定路由项插入新元素时, 会依次估算 *Set*[*BF Vector*] 中每个 *BF Vector* 的元素数量, 找到未达到阈值的 *BF Vector* 并插入. 当所有 *BF Vector* 所含元素数量均达到阈值时, 则会生成新的 *BF Vector* 并加入 *Set*[*BF Vector*] 集合中, 理论上单个路由项的容量可以支持无限扩容. *BF Vector* 的存储开销为常数级别, 因此反向路由表的存储开销较小, 后续算法分析和实验也将证明此结论. 此外也可以基于 Countable Bloom Filter 或 Cokoo Filter[28]构建反向路由表以支持数据的删除, 在未来工作章节将对此进行进一步讨论.

3.2 数据存储阶段构建分布式数据索引

在节点存储数据后, 存储节点生成数据索引消息并按照正向路由表向距离更近的邻居节点传递消息, 索引消息包含所存储数据的标识, 途径节点决策如下.

(1) 判断反向路由表中是否已包含索引消息上一跳节点的路由项<ID_{from}, *address*, *Set*[*BF Vector*]>, 若有则将数据标识 ID_{data} 写入 Bloom Filter 位向量 *Set*[*BF Vector*], 否则生成新的路由项并写入数据标识 ID_{data}.

(2) 判断正向路由表中是否还有距离更近的节点, 若有则继续传递索引消息.

节点在传递索引消息时的完整决策过程如算法 1 所示.

算法 1. 分布式数据索引构建算法.

输入: 当前消息的接收节点 *receiver*, 索引消息的上一跳节点 *incomNode*, 所存储的数据标识 *storedData*;
输出: 索引消息的下一跳节点 *v*.

```
1. function indexRcvSnd(Node receiver, Node incomNode, DataID storedData).
2.    if (incomNode is contained in receiver's backward routing table)
3.       find available BF Vector from Set[BF Vector] of the target entry; //找到尚未填满的 BF Vector
4.       append storedData in this BF Vector;
5.       if (all the BF Vector is full) //该路由项中所有的 BF Vector 均已填满
6.          generate new BF Vector and append storedData into BF Vector;
7.          append new BF Vector into Set[BF Vector];
8.    else
9.       create a new entry in backward routing table and;
```

10. generate new *BF Vector* and append *storedData* into *BF Vector*;
11. append new *BF Vector* into *Set*[*BF Vector*];
12. select some neighbor *v* which is closest to *storedData* from the *receiver*'s forward routing table;
13. **return** *v;* //返回索引消息的下一跳节点

3.3 数据查询阶段基于双向路由寻址数据

从任意节点发起数据查询, 查询发起节点生成正向数据查询消息并按照正向路由表向距离更近的邻居节点传递消息, 数据查询消息包含目标数据标识和查询发起节点, 途径节点决策如下.

(1) 判断当前节点是否有目标数据, 若有则将目标数据发送给查询发起节点, 否则继续执行 (2).

(2) 遍历反向路由表项, 判断目标数据 *targetData* 的标识是否包含在此路由项 *entryNode* 的 *Set*[*BF Vector*] 中, 且 *entryNode* 和 *targetData* 的距离大于当前节点 *localNode* 与 *targetData* 的距离, 即: $D(entryNode, targetData) \geq D(localNode, targetData)$, 则将查询消息传递给此路由项对应的节点, 并标注查询消息传递方向为反向.

(3) 若查询消息为正向, 则从正向路由表中选择与目标数据距离更近的节点传递查询消息.

节点在数据查询阶段收到查询消息时的完整决策过程如算法 2 所示.

算法 2. 基于双向路由的数据查询算法.

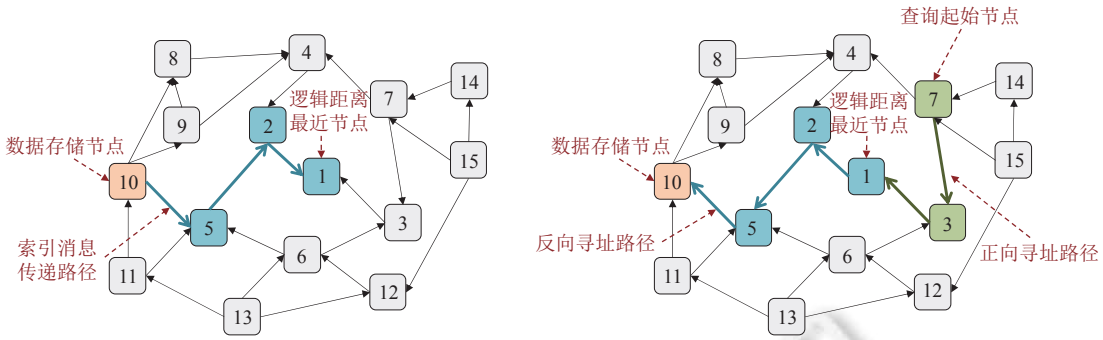
输入: 查询起始节点 *originNode*, 查询消息上一跳节点 *incomNode*, 当前消息接收节点 *receiver*, 查询目标数据标识 *targetData*, 当前消息传递方向 *direction*;
输出: 所查询的目标数据 *data*.

1. **function** *searchRcvSnd*(*Node originNode*, *Node incomNode*, *Node receiver*, *DataID targetData*, *Direction direction*).
2. **if** (*targetData* is found in local storage)
3. **return** data to *originNode*;
4. **else**
5. select neighbors *b* of which *targetData* is contained in *Set*[*BF Vector*] from *receiverNode*'s backword routing table;
6. *searchRcvSnd*(*originNode*, *receiver*, *b*, *targetData*, *backward*); //将消息反向传递给匹配节点
7. **if** (*direction* is forward)
8. select some neighbor *v* which is closest to *storedData* from the *receiver*'s forward routing table;
9. *searchRcvSnd*(*originNode*, *receiverNode*, *v*, *targetData*, *forward*); //将消息正向传递给更近节点

定理 1. 在基于 Kademlia 算法构建的 P2P 网络中, 其网络规模为 N. 假设网络中所有节点均相互连通, 对于给定数据 D, 可计算出每个节点和 D 的距离 $d(Node, D)$. 以网络中节点为点, 每个节点最多引出 α 条边指向路由表中和 D 距离最近的节点, 可得一张有向无环图 G. 其中 G 的顶点 $Node_t$ (出度为 0 的点) 为距离 $d(Node_t, D)$ 最小的点, 且从任意点出发, 最多经过 $\log N$ 个节点即可到达 $Node_t$.

证明: 由于仅保留 d 减少的边, 因此 G 中不存在环, 故 G 是有向无环图, 且由于 $Node_t$ 无更近节点, 故 $Node_t$ 为图的顶点. 其次, 对于 $Node_x$ 假设其与 D 的距离 $d(Node_x, D) \in [2^i, 2^{i+1})$, 根据 Kademlia 路由表构建规则, $Node_x$ 的路由表中第 i 个 K-桶中的邻居节点 $Node_{neighbor}$ 与 $Node_x$ 的距离同样满足 $d(Node_{neighbor}, Node_x) \in [2^i, 2^{i+1})$, 因此对于任意邻居节点 $Node_{neighbor}$, 其与数据 D 的距离一定满足 $d(Node_{neighbor}, D) < 2^{i-1}$, 即在 G 中每经过一个节点, 节点与数据 D 的距离至少减少一半. 又由于网络中所有节点均相互连通, 因此从任意点出发, 最多经过 $\log N$ 个节点即可到达顶点 $Node_t$. 定理 1 得证.

对于一个 N=15 的存储网络, 如图 1 所示. 图中数字即代表该节点与 D 的距离, 其中 $Node_1$ 为全网距离 D 最近的节点, $Node_{10}$ 和 $Node_7$ 分别为数据的存储节点和查询起始节点.



(a) 分布式索引构建 (b) 基于双向路由的数据寻址

图 1 vRoute 算法示例 (节点网络规模 N=15)

数据存储阶段如图 1(a) 所示, 假设 D 存储在距离为 10 的节点 $Node_{10}$ 中 (节点标识 ID_{10}), $Node_{10}$ 会发起数据索引消息, 消息将沿着正向路由路径 $Node_{10}$→$Node_5$→$Node_2$→$Node_1$ 在 4 跳之后传递到 $Node_1$. $Node_5$, $Node_2$, $Node_1$ 在收到索引消息时会更新自己的反向路由表增加上一跳节点路由项, 例如 $Node_5$ 的反向路由表中会包含 <ID_{10}, $address_{10}$, Set_{10}[*BF Vector*]>, 其中 ID_d 包含在 Set_{10}[*BF Vector*] 中. 索引消息经过 3 跳传递完成数据索引流程.

数据查询阶段如图 1(b) 所示, 假设查询起始节点为 $Node_7$, $Node_7$ 发起针对 D 的查询消息, 消息首先沿着正向寻址路径 $Node_7$→$Node_3$→$Node_1$ 传递至 $Node_1$, $Node_1$ 收到 D 的查询消息后发现本地反向路由表中有匹配项 $Node_2$ 随即将查询消息发送给 $Node_2$, 消息继而沿着反向寻址路径传递至 $Node_{10}$, $Node_{10}$ 将本地数据 D 直接返回给 $Node_7$, 查询消息共经过 5 跳传递至数据所在节点.

4 vRoute 算法分析

本节将从有效性和开销两个方面对 vRoute 算法进行分析, 证明 vRoute 算法从任意节点出发均能找到目标数据, 且时间、存储、网络开销均接近于传统 DHT 算法.

4.1 算法有效性分析

如前文所述, vRoute 算法基于 Bloom Filter 构建反向路由表, 使得数据寻址过程中可以快速判断下一跳地址. Bloom Filter 实现较低的存储、计算开销的代价是一定的假阳性误判概率, 但假阳性并不会影响反向寻址的正确性, 若目标数据存在则反向寻址消息一定能传递至数据所在节点.

定理 2. 在反向寻址路径上的节点可用的情况下, 反向寻址一定能找到目标数据所在节点, Bloom Filter 的假阳性不影响 vRoute 算法反向寻址的正确性, 仅影响反向寻址复杂度.

证明: Bloom Filter 对集合元素判断有一定假阳性可能, 即若目标元素在集合中则一定能判断出来, 若目标元素不在集合中则有一定概率误判. 在反向寻址过程中若目标数据在反向路由表中则一定可以将寻址消息传递给离数据存储节点更近的邻居节点, 因此 Bloom Filter 假阳性不影响正确的反向寻址路径. Bloom Filter 的假阳性会导致消息有一定概率被转发至错误节点, 因此会影响反向寻址的复杂度. 定理 2 得证.

vRoute 算法的寻址过程分为正向/反向两个阶段, 其中正向寻址和传统 Kademlia 算法一致, 寻址消息最多经过 log$N$ 跳到达全网距离最近节点随后停止, 而反向寻址也会在 log$N$ 跳内停止, 因此无论是否找到目标数据, vRoute 算法的数据查询过程最多经过 2log$N$ 跳后停止, 网络开销较低.

定理 3. 在网络规模为 N 的 vRoute 存储网络中, 在反向寻址会停止, 且会在 log$N$ 步数内停止.

证明: 假设查询的目标数据为 D, 反向寻址路径是沿着存储阶段构建的分布式索引路径反向传递消息的过程, 必然朝着 d(*Node*, D) 增加的方向因此不会循环, 一定会终止. 对于正确的反向寻址路径, 由于正向索引路径长度最长为 log$N$, 因此一定能在 log$N$ 步数内找到数据. 对于假阳性导致的错误反向寻址路径, 途径节点 $Node_{\text{local}}$ 停止继

续传递寻址消息的条件有如下 3 条.

(1) 本地无目标数据, 且反向路由表中无匹配节点 (即找到导致假阳性的数据所在节点).

(2) 反向路由表中有匹配的邻居节点, 但 $d(Node_{neighbor}, D) < d(Node_{local}, D)$.

(3) 反向路由步数已大于 log$N$ (正确路径长度小于 log$N$).

因此, 无论是否找到目标数据, 反向寻址均会在 log$N$ 跳内停止. 定理 3 得证.

综上所述, 在节点可用的情况下 vRoute 算法对于数据的寻址是有效的, 对于网络中存在数据的寻址请求 vRoute 一定可以查找到目标数据所在位置. 而对于不存在数据的寻址请求 vRoute 也能在有限的步数内停止, 网络开销较低.

4.2 算法开销分析

索引构建算法和数据寻址算法的复杂度: 如定理 1 所述, 在基于 Kademlia 算法构建的结构化 P2P 网络中, 针对某个数据可得一张有向无环图 G, vRoute 算法的索引构建流程可以看做是从任意节点向顶点 $Node_t$ 的路由过程, 其长度最长为 log$N$, 因此索引构建算法的复杂度为 O(log$N$). 数据寻址流程可以分为正向和反向两个阶段, 其中正向阶段为任意节点向顶点 $Node_t$ 的路由, 而反向阶段则是从顶点 $Node_t$ 沿着 G 中的边逆向路由的过程, 长度最长也为 log$N$, 因此 vRoute 算法的寻址复杂度为 O(log$N$). 在第 5 节中将对 vRoute 算法的索引和寻址效率进行实验评估.

假阳性对寻址效率的影响: 根据定理 2 的结论, Bloom Filter 的假阳性不会影响寻址的正确性和复杂度, 仅会增加一定的冗余消息 (同一条消息多次传递视为多条). 通过对假阳性概率的合理设置可以保证假阳性所造成冗余消息量不会带来较大的网络开销.

定理 4. 在节点规模为 N 的 Kademlia 存储网络中, 令假阳性概率为 p, K-桶容量为 k, 则假阳性导致的冗余消息量为 $R = kp \times \log N \times \sum_{i=1}^{\log N} i = kp \times \log^2 N(\log N + 1)/2$.

证明: 对于节点规模为 N 的网络, 节点标识至少包含 log$N$ 个比特, 即每个节点本地需要有 log$N$ 个 K-桶, 每个 K-桶最多有 k 个邻居节点, 从而得出每个节点正向路由表中路由项的数量最多为 $k$log$N$. 假设节点间连接关系构建随机, 正向路由表中路由关系分布均匀, 则根据对称性原则, 节点反向路由表中所包含的邻居节点数量亦为 $k$log$N$. 又有假阳性概率 p, 因此反向寻址过程中每一跳可能引发的假阳性寻址消息分叉数量期望为 kplog$N$. 由定理 3 可知, 所有反向寻址均会在 log$N$ 跳内停止, 产生在第 1 跳的假阳性消息最多经过 log$N$ 跳, 产生于第 2 跳的假阳性消息最多经过 log$N$–1 跳, 以此类推. 故假阳性导致的冗余消息量最多为:

$$R = kp \times \log N \times \sum_{i=1}^{\log N} i = kp \times \log^2 N(\log N + 1)/2 \tag{2}$$

对于节点规模为 1 000, K-桶容量为 20 的 Kademlia 存储网络, 令假阳性概率为 0.001, 则单次查询由于反向路由表的假阳性而产生的冗余消息量最多约为 10 条. 此外, 假阳性导致的冗余寻址和正确寻址路径并行, 因此不会影响查询的响应时间.

5 模拟实验

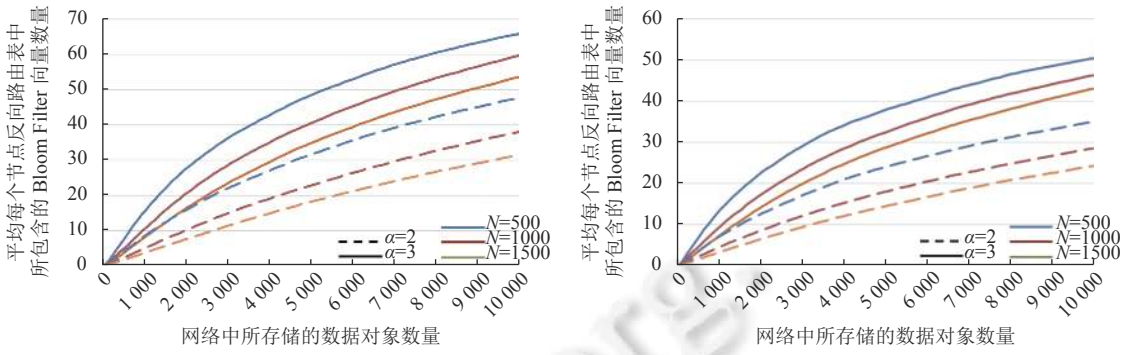
本文采用 Kademlia 算法作为结构化 P2P 网络的构建算法, 在 Kademlia 的基础上增加了分布式索引构建和反向路由表的设计以实现 vRoute 算法. 此外, 通过扩展 Peersim[29]的代码进行了 vRoute 算法的实验模拟, Peersim 是一种单机环境中的 P2P 模拟软件, 以单机环境下的对象调用替代真实环境下的网络通讯来模拟 P2P 节点之间的信息交互, 仅需较低的资源开销就可以模拟大规模的 P2P 网络, 在各种 P2P 研究中已被广泛使用. 本文基于 Peersim 进行了两部分实验: 评估 vRoute 算法本身的开销, 以及和传统 Kademlia 算法的性能对比.

5.1 vRoute 算法开销

vRoute 算法额外的存储开销主要体现在反向路由表中 Bloom Filter 所占的存储空间, 本文通过评估不同网络规模 N、数据量 M 以及消息并行度 α 的情况下平均每个节点在反向路由表中所包含的 Bloom Filter 向量数量来评估算法的存储开销. 基于文献 [30] 对 P2P 文件系统的分析, 本文模拟了常见的两种数据分布方式: 随机分布和 Zipf 分布. 实验结果如图 2 所示, 总体而言, 平均每个节点反向路由表中所包含的 Bloom Filter 向量数量随着 α 和

M 的增加而增加且最终趋于稳定. 对于 N=1000, α=3 且数据随机分布的存储网络中, M 达到 10 000 时平均每个节点所含 Bloom Filter 向量数量约为 60. 假设每个 Bloom Filter 期望所包含元素个数为 1 000, 假阳性概率 p=0.001, 则每个 Bloom Filter 向量所占用存储空间:

$$S = 1000 \times \log_2 e \times \log_2 \frac{1}{1000} \approx 14.4\ \text{Kb} = 1.8\ \text{KB} \tag{3}$$

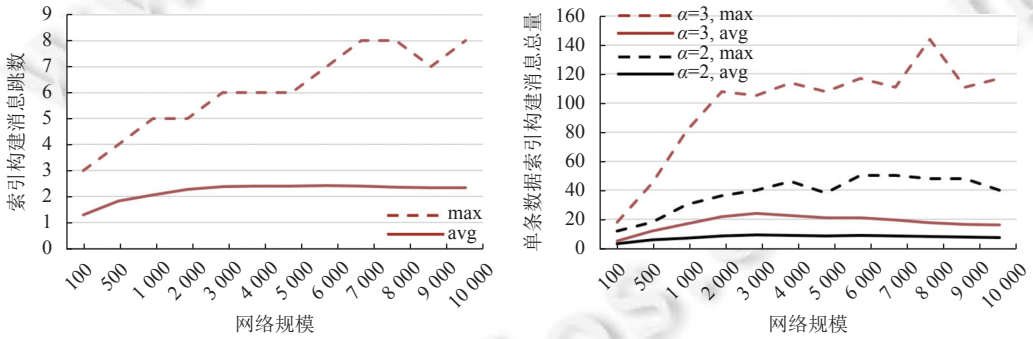


(a) 随机分布下的存储开销 (b) Zipf 分布下的存储开销

图 2 反向路由表存储开销

平均每个节点反向路由表总开销约为 108 Kb. 由于 Zipf 分布下, 数据集中在少量的节点中, 因此在相同数据量下, 反向路由表的开销会略低于随机分布, N=1000, α=3, M=10000 时, 平均每个节点的存储开销约为 81 KB.

在分布式索引构建方面, 本文评估了索引构建过程中的存储和网络开销. 由于常见 DHT 算法中数据和节点标识均基于 Hash 算法生成, 分布随机, 实验中数据和节点的标识同样遵循随机分布. 索引构建过程的主要时间开销在节点之间的消息传递过程, 因此本文评估了在不同网络规模下索引构建消息的最大跳数和平均跳数, 实验结果如图 3(a) 所示. 当节点规模为 N 时索引构建消息最大跳数不超过 log$N$, 具体而言当节点规模为 10 000 时索引消息最大跳数仅为 8, 平均跳数约为 3.



(a) 分布式索引构建消息跳数 (b) 单条数据索引构建消息总量

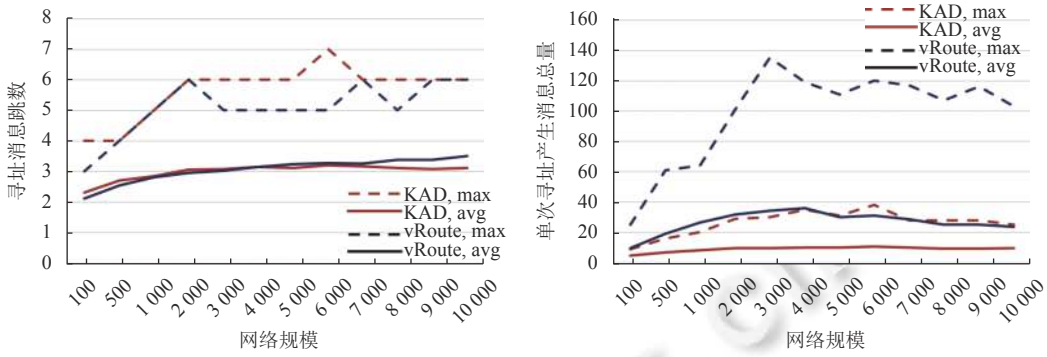
图 3 分布式索引构建过程中的时间、网络开销

索引构建的网络开销体现在索引构建消息的传递次数, 因此本文评估了在不同网络规模 N 和并行度 α 的情况下单条数据索引构建的最大和平均网络开销, 如图 3(b) 所示, 单条数据索引构建的最大消息总量基本与 N 和 α 正相关; 当 N=10000, α=2 时单次索引构建产生的消息总量最大为 40, 当 N=10000, α=3 时消息总量最大值为 120, 即单条数据的索引构建会在网络中产生 120 条消息的传递. 更高的并行度保证了更高的反向寻址路径的可用性, 但同时也增加了一定的网络开销.

5.2 数据寻址的性能对比

本文与传统 Kademlia 算法在数据寻址过程中的时间和网络开销进行了对比, K-桶容量 k 及查询并行度 α 采

用了当前主流的 DHT 存储之一 IPFS 的配置: k=20, α=3, 反向路由表中假阳性概率 p=0.001, 实验进行了 500 次查询请求取最大值和平均值进行对比分析, 实验结果如图 4 所示.



(a) 数据寻址消息跳数 (b) 单次数据寻址消息总量

图 4 数据寻址过程中的时间、网络开销

整体来看 vRoute 算法数据寻址的时间开销和 Kademlia 算法相当, 与网络规模 N 成对数关系. 如图 4(a) 所示, 当 N=10000 时 vRoute 算法对单一数据对象的寻址平均经过 3.5 跳即可找到目标数据和 Kademlia 算法的 3.2 跳相当. 在网络开销方面, 由于 Kademlia 的寻址路径面向顶点呈收敛特征, 而 vRoute 算法的反向寻址是由顶点出发向叶子节点路由的过程, 呈发散特征, 因而 vRoute 算法在数据寻址方面网络开销要略高于 Kademlia 算法. 但实验结果表明 vRoute 寻址的网络开销与 N 基本成对数关系, 并不会造成过多的网络开销. 如图 4(b) 所示, 当 N=10000, 在 500 单次数据寻址模拟中, 单次数据寻址产生的消息数量最多为 103 条, 平均消息数量则为 23.66 条.

6 总 结

DHT 技术由于其极高的寻址效率被广泛应用于 P2P 存储系统中, 然而现有的 DHT 技术要求数据必须存放在算法指定的节点上才能实现高效寻址, 这种数据和存储位置的紧耦合性限制了 DHT 技术在异构 P2P 网络中的应用. 本文首先分析了传统基于 DHT 的 P2P 存储中数据耦合问题的原因, 随后提出了面向 DHT 的数据解耦算法 vRoute, 通过增加基于 Bloom Filter 的分布式索引实现数据和存储位置的解耦, 并以 Kademlia 为例从理论上分析了算法的有效性和开销, 证明 vRoute 算法在不降低数据寻址复杂度的情况下可以实现数据和存储位置的解耦. 最后, 本文通过扩展 Peersim 代码对 vRoute 算法进行了模拟实验, 实验结果表明在存储开销方面, vRoute 算法本身所带来的额外开销并不大; 在数据寻址方面, vRoute 算法的寻址效率和 Kademlia 相当, 而网络开销较 Kademlia 略虽有提升但仍然是和网络规模呈对数相关.

7 未来工作

本文提出了 vRoute 算法以解决传统 DHT 存储中数据标识和其存储位置的紧耦合关系, 在异构分布式存储、安全和隐私保护存储领域均有一定的应用潜力. 本文仅就 vRoute 算法的核心理念进行了详细介绍, 并未就异构 DHT 存储中的所有问题进行阐述, 比如: 如何支持数据的修改和删除, 以及如何应对 P2P 环境下节点频繁加入退出而导致的“搅动”现象. 首先, 在数据修改和删除方面, 本文采用 Bloom Filter 来构建反向路由表, 但 Bloom Filter 并不支持元素的删除. 通过将 Bloom Filter 替换为 Countable Bloom Filter 或者 Cokoo Filter 即可支持数据的删除, 删除数据的过程和构建分布式索引的路径相同, 只是节点在处理时将插入新元素的逻辑修改为删除新元素即可. 其次, “搅动”现象是 P2P 网络中常见的现象[31], 在传统 DHT 存储中, 节点通过周期性的更新路由表, 并给每个路由项设置一个存活时间 TTL (time to live), 保持路由表的有效性以应对“搅动”. vRoute 算法解决搅动的一个思路是在支持数据删除的基础上, 为反向路由表中每个路由项设置一个 TTL, 及时清理过期的数据项以保障反向路由表的正确性, 数据存储节点则可以通过周期性的重复构建数据索引以更新数据对应路由项的 TTL. 未来, vRoute 算法会就这两方面进行进一步的研究和优化.

References:

[1] He QL, LI ZH, Zhang X. Study on cloud storage system based on distributed storage systems. In: Proc. of the 2010 Int'l Conf. on Computational and Information Sciences. Chengdu: IEEE, 2010. 1332–1335. [doi: 10.1109/ICCIS.2010.351]

[2] Hwang K, Kulkareni S, Hu Y. Cloud security with virtualized defense and reputation-based trust management. In: Proc. of the 8th IEEE Int'l Conf. on Dependable, Autonomic and Secure Computing. Chengdu: IEEE, 2009. 717–722. [doi: 10.1109/DASC.2009.149]

[3] Xu K, Song MN, Zhang XQ, Song JD. A cloud computing platform based on P2P. In: Proc. of the 2009 IEEE Int'l Symp. on IT in Medicine & Education. Jinan: IEEE, 2009. 427–432. [doi: 10.1109/ITIME.2009.5236386]

[4] Benet J. IPFS-content addressed, versioned, P2P file system. arXiv:1407.3561, 2014.

[5] Wilkinson S, Boshevski T, Brandoff J, Buterin V. Storj: A peer-to-peer cloud storage network. 2016. https://www.doc88.com/p-9741740094823.html?r=1

[6] Chen GH, Li ZH. Peer-to-peer Network: Struture, Application and Design. Beijing: Tsinghua University Press, 2007. 53–58 (in Chinese).

[7] Hassanzadeh-Nazarabadi Y, Küpçü A, Ozkasap O. Decentralized utility- and locality-aware replication for heterogeneous DHT-based P2P cloud storage systems. IEEE Trans. on Parallel and Distributed Systems, 2020, 31(5): 1183–1193. [doi: 10.1109/TPDS.2019.2960018]

[8] Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. on Networking, 2003, 11(1): 17–32. [doi: 10.1109/TNET.2002.808407]

[9] Rowstron AIT, Druschel P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Proc. of the 2001 IFIP/ACM Int'l Conf. on Distributed Systems Platforms Heidelberg. Berlin: Springer, 2001. 329–350. [doi: 10.1007/3-540-45518-3_18]

[10] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego: Association for Computing Machinery, 2001. 161–172. [doi: 10.1145/383059.383072]

[11] Zhao BY, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD. Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications, 2004, 22(1): 41–53. [doi: 10.1109/JSAC.2003.818784]

[12] Aspnes J, Shah G. Skip graphs. ACM Trans. on Algorithms, 2007, 3(4): 37–es. [doi: 10.1145/1290672.1290674]

[13] Maymounkov P, Mazières D. Kademlia: A peer-to-peer information system based on the xor metric. In: Proc. of the 1st Int'l Workshop on Peer-to-peer Systems. Cambridge: Springer, 2002. 53–65. [doi: 10.1007/3-540-45748-8_5]

[14] Vishnumurthy V, Francis P. On heterogeneous overlay construction and random node selection in unstructured P2P networks. In: Proc. of the 25th IEEE Int'l Conf. on Computer Communications. Barcelona: IEEE, 2006. 1–12. [doi: 10.1109/INFOCOM.2006.180]

[15] Yao ZM, Leonard D, Wang XM, Loguinov D. Modeling heterogeneous user churn and local resilience of unstructured P2P networks. In: Proc. of the 2006 IEEE Int'l Conf. on Network Protocols. Santa Barbara: IEEE, 2006. 32–41. [doi: 10.1109/ICNP.2006.320196]

[16] Venkataraman V, Yoshida K, Francis P. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In: Proc. of the 2006 IEEE Int'l Conf. on Network Protocols. Santa Barbara: IEEE, 2006. 2–11. [doi: 10.1109/ICNP.2006.320193]

[17] Jia ZQ, You JY, Rao RN, Li ML. Random walk search in unstructured P2P. Journal of Systems Engineering and Electronics, 2006, 17(3): 648–653. [doi: 10.1016/S1004-4132(06)60111-4]

[18] Zhou RF, Hwang K. Gossip-based reputation aggregation for unstructured peer-to-peer networks. In: Proc. of the 2007 IEEE Int'l Parallel and Distributed Processing Symp. Long Beach: IEEE, 2007. 1–10. [doi: 10.1109/IPDPS.2007.370285]

[19] Zhu GM, Guo DK, Jin SY. P2P probabilistic routing algorithm based on data copying and Bloom Filter. Ruan Jian Xue Bao/Journal of Software, 2011, 22(4): 773–781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3757.htm [doi: 10.3724/SP.J.1001.2011.03757]

[20] Ma WM, Meng XW, Zhang YJ. Bidirectional random walk search mechanism for unstructured P2P network. Ruan Jian Xue Bao/Journal of Software, 2012, 23(4): 894–911 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4086.htm [doi: 10.3724/SP.J.1001.2012.04086]

[21] Flanagan C, Freund SN. FastTrack: Efficient and precise dynamic race detection. ACM SIGPLAN Notices, 2009, 44(6): 121–133. [doi: 10.1145/1543135.1542490]

[22] Leibowitz N, Ripeanu M, Wierzbicki A. Deconstructing the kazaa network. In: Proc. the 3rd IEEE Workshop on Internet Applications. San Jose: IEEE, 2003. 112–120. [doi: 10.1109/WIAPP.2003.1210295]

[23] Ripeanu M. Peer-to-Peer architecture case study: Gnutella network. In: Proc. of the 1st Int'l Conf. on Peer-to-peer Computing. Linköping: IEEE, 2001. 99–100. [doi: 10.1109/P2P.2001.990433]

[24] Nguyen HS. Topology optimization for heterogeneous DHT-based multicast. In: Kecskemeti G, ed. Applying Integration Techniques and

Methods in Distributed Systems and Technologies. Hershey: IGI Global, 2019. 134–155. [doi: 10.4018/978-1-5225-8295-3.ch006]

[25] Xia L, Duan HC, Zhou X, Zhao ZF, Nie XW. Heterogeneity and load balance in structured P2P system. In: Proc. of the 2010 Int'l Conf. on Communications, Circuits and Systems (ICCCAS). Chengdu: IEEE, 2010. 245–248. [doi: 10.1109/ICCCAS.2010.5581999]

[26] Zhu YW. Load balancing in structured P2P networks. In: Shen XM, Yu H, Buford J, Akon M, eds. Handbook of Peer-to-peer Networking. Boston: Springer, 2010. 1149–1164. [doi: 10.1007/978-0-387-09751-0_41]

[27] Bloom BH. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422–426. [doi: 10.1145/362686.362692]

[28] Fan B, Andersen DG, Kaminsky M, Mitzenmacher MD. Cuckoo filter: Practically better than Bloom. In: Proc. of the 10th ACM Int'l on Conf. on Emerging Networking Experiments and Technologies. Sydney: ACM, 2014. 75–88. [doi: 10.1145/2674005.2674994]

[29] Montresor A, Jelasity M. PeerSim: A scalable P2P simulator. In: Proc. of the 9th IEEE Int'l Conf. on Peer-to-Peer Computing. Seattle: IEEE, 2009. 99–100. [doi: 10.1109/P2P.2009.5284506]

[30] Gummadi KP, Dunn RJ, Saroiu S, Gribble SD, Levy HM, Zahorjan J. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. Bolton Landing: ACM, 2003. 314–329. [doi: 10.1145/945445.945475]

[31] Liu ZY, Yuan RF, Li ZH, Li HX, Chen GH. Survive under high churn in structured P2P systems: Evaluation and strategy. In: Proc. of the 6th Int'l Conf. on Computational Science. Reading: Springer, 2006. 404–411. [doi: 10.1007/11758549_58]

附中文参考文献:

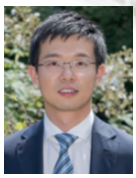
[6] 陈贵海, 李振华. 对等网络: 结构、应用与设计. 北京: 清华大学出版社, 2007. 53–58.

[19] 朱桂明, 郭得科, 金士尧. 基于副本复制和Bloom Filter的P2P概率路由算法. 软件学报, 2011, 22(4): 773–781. http://www.jos.org.cn/1000-9825/3757.htm [doi: 10.3724/SP.J.1001.2011.03757]

[20] 马文明, 孟祥武, 张玉洁. 面向非结构化P2P网络的双向随机漫步搜索机制. 软件学报, 2012, 23(4): 894–911. http://www.jos.org.cn/1000-9825/4086.htm [doi: 10.3724/SP.J.1001.2012.04086]

罗超然(1990－), 男, 博士生, CCF 学生会员, 主要研究领域为系统软件, 大数据互操作, 分布式系统, 数联网.

柳熠(1993－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为服务计算, 移动计算, 数字对象架构.



金鑫(1989－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为系统软件, 软件定义方法, 云计算.

景翔(1979－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为操作系统, 分布式计算.

张颖(1983－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为大数据互操作, 软件构件化开发.

黄罡(1975－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为系统软件, 软件自适应, 数联网.

蔡华谦(1990－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为系统软件, 分布式系统, 数联网.