SRing: A Structured Non DHT P2P Overlay Supporting String Range Queries

Xiaoping Sun
China Knowledge Grid Research Group
Institute of Computing Technology
Graduate School of Chinese Academy of Sciences, China
sunxp@kg.ict.ac.cn

Xue Chen
China Knowledge Grid Research Group
Institute of Computing Technology
Graduate School of Chinese Academy of Sciences, China
chenxue@kg.ict.ac.cn

ABSTRACT
This paper presents SRing, a structured non DHT P2P overlay that efficiently supports exact and range queries on multiple attribute values. In SRing, all attribute values are interpreted as strings formed by a base alphabet and are published in the lexicographical order. Two virtual rings are built: N-ring is built in a skip-list way for range partition and queries; D-ring is built in a small-world way for the construction of N-ring. A leave-and-join based load balancing method is used to balance range overload in the network with heterogeneous nodes.

Categories and Subject Descriptors
H.3.4 [**Information Storage AND Retrieval**]: Systems and Software-Distributed systems, Information networks.

General Terms
Algorithms, Design, Experimentation.

Keywords
P2P, multi-attribute, range queries, load balancing.

1. INTRODUCTION
Distributed Hash Table (DHT) overlays have high scalability of exact query routing in large-scale P2P systems [1]. However, it is difficult to implement range queries in DHT overlays. One way is to build index on DHT overlays to support range queries [4]. Another natural way is to let nodes partition the original keyword space to preserve their original order. Two key issues must be addressed, efficient query routing and balanced load distribution. This paper introduces a structured non DHT P2P overlay SRing that efficiently supports exact and range queries on multiple attribute value strings of data objects. Range overload can be quickly smoothed in heterogeneous environments without global knowledge of load distribution.

2. SYSTEM ARCHITECTURE
In SRing, data objects are identified by a set of attribute values. All attribute values are interpreted as strings consisting of characters of a base alphabet $E = \{e_1, e_2, ..., e_L\}$ ($L = |E|$). We assume that such interpretation can preserve the original order of attribute values. Many data types such as numeric values, date, and text have this property. All strings are sorted by the lexicographical order $\prec_S$ based on E. Nodes randomly draw strings of length m as their name IDs from the string space S formed by E. n nodes are sorted in increasing order of their name IDs, $nid_1 \prec_S nid_1 \prec_S ... \prec_S nid_n$, and form a ring called N-ring. Node nid_k is responsible for all strings falling in range $(nid_{k-1}, nid_k]$ in S. N-table is built to support range queries in N-ring. For building N-ring, node nid_k is also assigned a random digital string ID did_k of length m drawn from S, initially the same as its random name ID. Digital IDs are also sorted in increasing order to form a ring called D-ring. D-table is built to support query on digital IDs in D-ring.

N-ring is constructed based on common prefixes of digital IDs of nodes in a skip-list way [3] to support efficient range queries on attribute strings. Those nodes with digital IDs that share a common prefix of length l form a sub-ring at level l where nodes are sorted in increasing order of their name IDs, for l = 0, 1, 2, ..., and m (Figure 1 (a)). Level 0 is the underlying N-ring. Each node connects to its predecessor and $b = \lfloor \log L \rfloor$ successors in the each sub-ring ($L = |E|$) (Figure 1 (b)). Randomness of digital IDs ensures the efficiency of N-tables in any distribution of name IDs.

When a node joins N-ring, first it locates its digital ID in D-ring and joins D-ring. From its predecessor and successor in D-ring, the node chooses the one sharing with it longer common prefix of digital ID as the bootstrap node. Then it lookups its name ID in N-ring from that bootstrap node. Query is forwarded along the link closest to the target. When jumping into a different sub-ring, the node inserts itself into that sub-ring. When forwarding a range query, query messages are forwarded to the closest one to the target range. It can achieve $O(\log n)$ hops.

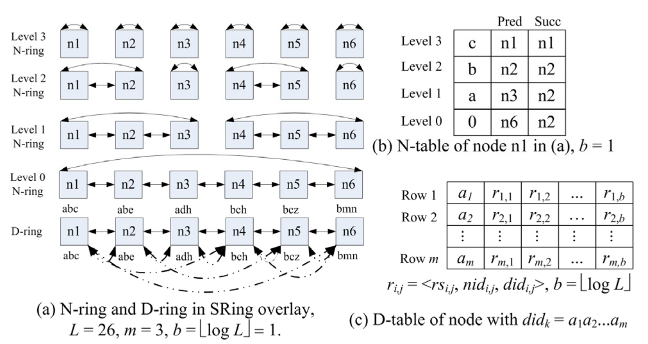


Figure 1. Architecture of SRing, N-table and D-table.

When joining D-ring, the node lookups its digital ID in D-ring and inserts into the overlay at the node currently holds its digital ID. D-table is built in a small-world way without requiring the estimate of network size. In node n_k with digital ID string $did_k = a_1a_2...a_m$ ($a_i \in E$), D-table contains m rows with each having $b = \lfloor \log_2 L \rfloor$ long links (Figure 1(c)). In the ith row, a long link $r_{i,j}$ is produced by a seed ID string $rs_{i,j} = a_1a_2,...b_ia_{i+1}...a_m$. $rs_{i,j}$ differs from did_k at the ith character. Let $v(a_i)$ denote the position of character a_i in E. Let $d = \lfloor L / 2^p \rfloor$ and p is a real number uniformly generated in range $[0, \log_2 L]$. Then, $(v(a_i) + d) \bmod L$ is the position of character b_i in E. The distance d between b_i and a_i in E follows the harmonic probability distribution in range (0, L]. In each row of D-table, we generate $b = \lfloor \log_2 L \rfloor$ seed IDs using the same harmonic probabilistic distribution. Total $m \lfloor \log_2 L \rfloor$ seed IDs are produced. It approximates Kleinberg's small-world network [2] in one-dimensional ring, without the estimate of network size.

After generating all seed IDs, n_k locates remote nodes that hold these seed IDs in D-ring and setups long links to them. In a query process in D-ring, each hop chooses the link closest to the target string for the next jump. Since many seed IDs correspond to the same node, D-table contains $O(\log n)$ distinct remote links and the routing hops is $O(\log n)$ in an overlay with n nodes. Like structured P2P overlays with ring geometries, D-table has more resilience in neighbor selection and route selection than tree geometries that support prefix-based routing [1], including the numeric routing table of SkipNet [3].

3. LOAD BALANCING IN SRING

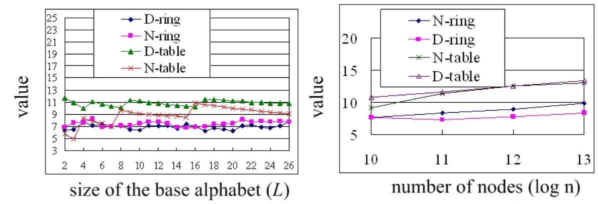
In the local storage of a node, strings are partitioned into a number of buckets, each with a predefined number of strings. The load of n_i is measured by a utilization factor $u_i = l_i / c_i$, where l_i is the number of buckets in n_i and capacity c_i is the maximum number of buckets n_i can hold. Load balancing process is periodically executed in each node to reduce the variance of u_i in the overlay.

During the load balancing, n_i concurrently sends out at most $K \ge 1$ random requesting messages that contain nid_i, did_i, l_i, c_i and a random digital ID did_s. Only when requests get responses or time out, can node n_i send more. Each requesting message is routed in D-ring to the node n_r that holds did_s and is recorded in the incoming list of n_r. n_i also keeps the load information l_{i+1} and c_{i+1} of its successor n_{i+1} in N-ring.

Then, n_i draws messages from its incoming request list for making load balancing decision. For a request from n_k with l_k load and c_k capacity, the utilization gain is $g_k = u_k - u_i$. If the largest gain $g_m < 0$, n_i rejects all requestors. If $g_m > 0$, we consider the increased load in the successor n_{i+1} of n_i. Let $\delta_m = \frac{l_m}{c_m} - \frac{l_m}{c_i + c_m}$ and $\delta_{i+1} = \frac{l_{i+1} + l_i}{c_{i+1}} - \frac{l_{i+1}}{c_{i+1}}$. If $\delta_m > \delta_{i+1}$, it means that the utilization increased in n_{i+1} is smaller than the utilization decreased in n_m. Then, n_i is moved to share $m_b = l_m c_i / (c_m + c_i)$ number of buckets of n_m. If $m_b < 1$, n_i rejects all requestors and waits for the next round of load balancing. When moving, n_i transfers all buckets to n_{i+1} and quits. Then it sets its name ID to the largest string among those moved buckets from n_m and joins N-ring again. It takes $O(\log n)$ messages. Digital IDs and D-tables of all involved nodes need not to be changed.

4. EXPERIMENTS

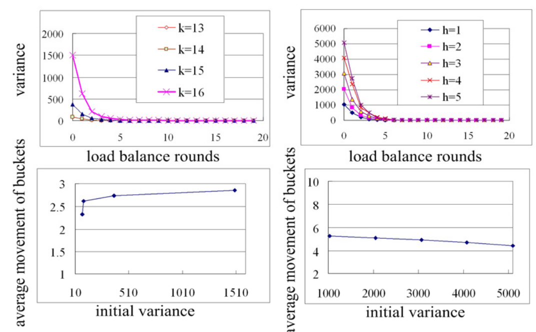
Figure 2 (a) shows that query hops is not affected by L in both N-ring and D-ring. Query hops and routing table size of N-ring and D-ring grow logarithmically with increasing n (Figure 2(b)). We test the load balancing method with Zipfian distribution of capacities (Zipf-cap) and uniform distribution (Unf-bucket) of bucket loads (Figure 3. (a)). Hotspot distributions (Hot-bucket) with all B buckets of strings located in h nodes are also tested in overlays with equal capacity (Unf-cap) (Figure 3 (b)). In each round one request is sent out. In all cases, the variance of the utilizations of nodes drops quickly. The average move times of buckets grows very slowly with the increasing initial variance.



(a) Average hops and routing table size in SRing with the base alphabet of different size, n = 1024, $b = \lfloor \log (L) \rfloor$, m = 16

(b) Average hops and routing table size in the overlay of different size, L = 26, $b = \lfloor \log(L) \rfloor$, m = 16

Figure 2. Query hops and routing table size.



(a) Zipf-cap, Unf-bucket, n = 1014, $B = 2^k$ buckets are published

(b) Unf-cap, Hot-bucket, n = 1024, and $B = h$*1024 buckets are published in h hotspots in the overlay

Figure 3. Load balancing effects and costs.

5. CONCLUSION

SRing can efficiently support range queries on multiple attribute value strings and can effectively smooth range overload in heterogeneous environments. SRing has potentials in supporting semantics-rich queries in large-scale distributed networks.

6. ACKNOWLEDGEMENTS

This work was supported by the National Basic Research Program (973 project no. 2003CB317000) and the National Science Foundation of China (No. 60503047)

7. REFERENCES

[1] K. Gummadi, R. Gummadiy, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity", In *Proceeding of SIGCOMM 2003*. pp. 381-394, 2003.

[2] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective", in *Proceeding of 32nd ACM STOC*, pp. 163-170, 2000.

[3] N. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties", In *Proceeding of USITS 2003,* pp. 113–126, Mar. 2003.

[4] H. Zhuge, X. Sun, J. Liu, E. Yao and X. Chen, "A Scalable P2P Platform for the Knowledge Grid", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No.12, pp. 1721-1736, 2005.