DEIM Forum 2010 F5-1

Multidimensional Range Query Processing in Structured P2P Overlays

Djelloul BOUKHELEF† and Hiroyuki KITAGAWA†,††

† Computer Science Department, University of Tsukuba
†† Center for Computational Sciences, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki. 〒 305–8573
E-mail: †boukhelef@kde.cs.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

Abstract The Multi-Ring Content Addressable Network (RCAN) [1] is a dynamic distributed index structure for efficient management of large multidimensional data sets over peer-to-peer systems. RCAN propose a new self-organizing overlay topology that achieves logarithmic routing performance and effective load balancing while minimizing the maintenance overhead during nodes join and departure. In this paper, we extend RCAN with an efficient support for complex queries, namely multi-attributes range queries. A range query issued by one node is successively refined by intermediate nodes until reaching the nodes involved in the query range. Recursive decomposition is based only on local information about neighboring nodes. Resulting sub-queries are forwarded in parallel to prospective target nodes. Our range query algorithm reduces the number of involved nodes, the message overhead necessary to solve a range query, as well as the total query latency which is logarithmic to number of nodes in the system (N) and involved nodes (M), namely $O(\log N + \log M)$ hops.

Key words Structured peer-to-peer, RCAN overlay, Multidimensional data, Range query processing.

1. Introduction

Peer-to-peer (P2P) is a powerful key paradigm for structuring scalable distributed systems which are decentralized and highly dynamic. Research on indexing on structured P2P networks started with the introduction of Distributed Hash Tables (DHTs). A wide range of structured P2P protocols have been proposed over the last few years, such as: Chord [2], CAN [3], P-Grid [4], to name only a few. DHT protocols support efficiently exact-match lookups. However, because the randomization does not preserve the locality of data, DHT schemes are generally not efficient to support complex queries such as range queries and nearest-neighbor searches. In situations where applications require efficient execution of such queries, more sophisticated methods that simultaneously achieve good load balance and preserve data locality are highly required.

Research on storage and management of multidimensional data in P2P systems has gained emerging intension during last years. The aim is to propose efficient techniques for indexing and processing complex queries over huge collections of multidimensional and spatial data shared among a large-scale structured P2P. Towards the efficient processing of complex and range queries, non-DHT structured P2P techniques consider adapting well-known centralized hierarchical data structures (such as search trees, tries, and skip list) to the P2P settings. These techniques do not employ hashing for data placement, and aim to address more complex problems such as multi-attribute indexing, nearest-neighbor and similarity indexing, and range search. One of the most challenging problem that face this class of structured P2P system is to avoid overloading the higher levels of the structure. Several P2P range indexes have been proposed, such as Mercury [5], BATON [6] and its variants [7], [8].

Unlike simple *point* queries, evaluation of a range query is, much more complicated. This is mainly due to the lack of global knowledge about the actual network composition, and its dynamism due to *churn*, that is nodes that join and leave the network freely. The major challenge in this settings to resole a range query is how to efficiently refine and propagate the query to all relevant nodes, while fulfilling the following properties:

- Reduce the number of visited (*non-computing*) nodes, that is nodes whose regions do not intersect with the query range, yet they may help forwarding the query message.
- Reduce the number of messages necessary to solve the range query, as well as the total time to answer the query (*latency*).
- Balance the workload among the processing nodes.

RCAN, a Multi-ring Content Addressable Network [1], is a new fully self-scaling decentralized P2P protocol with a novel topological and routing infrastructure. RCAN proposes a dynamic distributed index structure for the efficient management of large multidimensional data sets over P2P systems. Based on a new self-organizing overlay topology, RCAN is able to achieves logarithmic routing performance and effective load balancing [9] while minimizing the maintenance overhead during nodes join and departure.

In this paper we are interested in proving mechanisms for effi-

cient support of complex multidimensional queries in a structured P2P. We propose an extension of RCAN overlay to supports efficient evaluation of range queries and partial-match queries. To achive this goal, RCAN relies on an important property of range query, we called "*query locality*", which is the ability to efficiently traverse in between nodes whose regions are adjacent on the d-dimensional key space. This means that once entering the query region (starting from any boundary) the query propagation paths will never get out of the query range when forwarding the query message to other relevant nodes. The query locality property is considerably useful in reducing the number of *non-computing* nodes to be visited, and consequently the total query latency. Based on this property, the proposed range query processing algorithm fulfills all the above-mentioned properties. While most of the existing works fail to fulfill both properties (1) and (3) as they do not preserve the query locality.

The communication and time complexities of our range query processing algorithm are both logarithmic to the number of nodes that intersect with the query range. This parameter depends heavily on the query selectivity (ratio of query range to the whole space) and the distribution of nodes.

The remainder of this paper is organized as follows. Section 2. covers the related work and compares our proposal with existing methods. Section 3. briefly describes RCAN overlay and its routing topology. Section 4. presents the details of the proposed range query processing mechanism. Finally, section 5. concludes this paper and discusses future work.

2. Related work

In the literature there exist two common approaches for managing multidimensional data over structured P2P systems: either by using an order-preserving hash function over existing DHT, or by proposing distributed variants of centralized search structures.

Methods belonging to the first approach maps the multidimensional space onto a one-dimensional space using space filling curves (SFC) techniques. Then it makes use of well-known DHT protocols to manage the one-dimensional space. A main drawback of this approach which is inherited directly from the usage of SFC, is its failure to preserve the locality of data on the one-dimensional space. Moreover, the data locality of SFC will become worse with the increase of dimensionality. Another unfavorable factor, is the random hashing of partitions of the key space onto nodes in the DHT overlay.

Methods of the second approach propose distributed hierarchical indexing schemes originated from well-known centralized multidimensional indexes. A common feature of this approach is to assign special roles to some peers in the system to maintain higher-layer of the hierarchy. As a result, methods of this class present severe scalable limitations as the upper level peers might easily become bottleneck (single point of failure).

Mercury [5] attempts to support multiple-attribute range queries by using an individual hub for each attribute. Mercury organizes each attribute hub into a separate circular overlay of nodes and places data contiguously on the ring. MAAN [10] uses locality-preserving hashing to map data values onto Chord identifier space. Both MAAN and Mercury employ multiple DHTs (one for each attribute) that are mapped onto the same overlay. The separate attribute indexing scheme is not effective for processing complex queries that explicitly specify constraints among the attributes such as skyline queries [11], and range queries. Moreover, multi-attribute range query is more complicated since is has to be evaluated on one dimension at a time and then take a join operation of the results.

P-Grid [4] is a one-dimensional distributed index based on a randomized binary prefix tree (trie) which is induced by recursively bisecting the data space. Each peer is associated with one partition of the key space and maintains random connections to other peers such that prefix routing is enabled. A peer stores also some data index replicas belonging to other peers, to guarantee fault-tolerance. However, this way still has high index costs. The work in [12] proposes two algorithms of range query in P-Grid, *i.e.* min-max traversal algorithm (*sequential*) and shower algorithm (*parallel*).

BATON [6] is a one-dimensional index that uses a distributed search tree where *in-level* sideway links are employed to reduce the number of accesses to higher levels of the hierarchy and achieve routing efficiency, fault-tolerance, and load balancing. BATON* [7] extends the idea of BATON by increasing the tree fan-out to speedup the search. Work in [8] proposed the VBI-tree, a P2P network based on BATON that can adapt many data partitioning schemes. Inherited from BATON, the three methods share the following common problems: the tree structures undertake rigid node insertion procedures, where a tree restructuration may affect several nodes. This restructuring process is expensive and prevents concurrent insertions.

The works closest to ours in the multi-dimensional data indexing are MURK and SCRAP [13], SkipIndex [14], ZNet [15], Squid [16]. Belonging to the first category, these methods employ SFC techniques for dimension reduction. Multi-dimensional data is converted into single-dimensional data and then range-partitioned across peers. Existing DHT networks are then used to manage the one-dimensional space. MURK indexes multi-dimensional data partitions using the kd-tree. SkipIndex stores partition information in a binary tree. ZNet proposes a mapping of Quad-tree onto skip graph overlay using Z-curve. The evaluation of a range query in these methods rely mainly on the hierarchical overlay structure. For example, in ZNet a range query is converted to a set of zones, which is a superset of zones covered by the query range. As the query is routed, this superset is refined. To reduce unnecessary visits, the query is routed along two opposite directions, that is towards nodes which contain zones of larger Z-addresses as well as smaller Z-addresses than zones' of the current node. Apparently this solution is targeting to reduce the number of visited nodes, as well as the associated network overhead. However, such a top-down approach

may cause serious scalability issues, as the query request is always sent to higher level nodes that maintain a larger search range.

The work in [17] proposes a distributed version of the MX-CIF quad-tree that supports range queries and objects with multi-dimensional extents. The leaves of the quad-tree are mapped onto a Chord overlay. In order to avoid the bottleneck of higher-layer nodes, the tree is cut-down to a certain level (f_{min}), so that no objects are stored in upper levels of the quad-tree. Also they limit the depth of the tree to (f_{max}) to avoid too much fragmentation. Each peer caches direct links to the children of the quad-tree nodes it is covering. Thus, it takes $O(\log N)$ hops to find an f_{min}-node and then a constant number of steps to reach the relevant leaves. However, the resulting structure is rather complex and need to maintain multiple sub-roots to access the structure which incurs a very high maintenance overhead.

Range query processing in the above methods consist basically of translating the query range into set of cells using the SFC, then routing each cluster of cells to relevant nodes. To reduce the routing overhead, the query is refined during the routing process using SFCs of different granularities. Besides visiting many irrelevant nodes (*non-processing*) during the query routing and propagation, a potential drawback in the previous methods, including the efficient shower algorithms [12], is that unnecessary overlay hops may be introduced in order to forward sub-queries to relevant nodes which are naturally adjacent on the multidimensional space.

SpatialP2P [18] is a recently proposed structured P2P framework, targeted to spatial data (2 dimensions only). Like our RCAN, SpatialP2P is the only pure P2P overlay that does not employ neither SFC techniques nor built using hierarchical structure. However, SpatialP2P is intrinsically different from RCAN. SpatialP2P uses a statically grid-partitioned key space, where nodes handle areas, which are either cells of the grid-partitioned space or sets of cells that do not necessarily form a rectangular region. Although this choice makes the load balancing task more flexible, however the size of metadata needed to manage complex regions may be very large especially for higher dimensions. Similarly to the above methods, SpatialP2P converts the query range into the a set of predefined cells, and route them to the relevant nodes. To reduce the routing overhead, SpatialP2P group non-resolved cells and send them to a neighboring node that is closer to all of them which in its turn will continue the query resolution.

3. Overview of RCAN framework

In this section we will briefly describe RCAN, a Multi-ring Content Addressable Network [1]. RCAN is a new self-scaling P2P protocol with a novel topological and routing infrastructure. The basic substrate of RCAN is a conventional grid-like overlay, where nodes know only about their immediate neighborhoods. The key design of RCAN is to equip each node with a few long links towards some distant nodes in the system. Long links are established in such a way that the routing path is shortened while the maintenance overhead for building and updating these links when nodes join or leave the system is very low. Distant neighbors are situated at distances inverse of powers of 2 on the coordinate space from the originating node. The set of long links from each node is partitioned into d small subsets, each of which is established along one dimension. Long links are clockwise-directed and wrap around the key space. The set of all long links in the system yields multiple independent rings along each dimension (*routing infrastructure with multiple rings*). The rings are of small size (number of nodes per ring), and their maintenance is very easy. The number of rings and their sizes self-adjust as nodes join and leave the network. In a uniformly partitioned key space, a node is member of only one ring along each dimension, *i.e.* the ring that intersects with its region. The goal of RCAN is twofold. First it aims to improve the routing performance and enhance the fault-tolerance of CAN-like overlays by building fast shortcuts between nodes on the overlay level, while minimizing their maintenance cost during frequent nodes joins and departures (*churn*). The second ultimate goal is to efficiently support semantic queries over data with multidimensional keys. Semantic queries include, for example, multi-attributes range queries, k-nearest neighbor search, etc. On the architectural point of view, RCAN is mainly designed for a large scale dynamic P2P systems.

3.1 Design principle

RCAN operates on a native d-dimensional Cartesian key space that warps around each dimension. The key space is subdivided into non-overlapping hyper-rectangular regions (called also *zone*). Regions' extents could be changed dynamically through split and merge operations that occur when nodes join or leave. In what follows we will describe the data and overlay structures of RCAN.

Regions

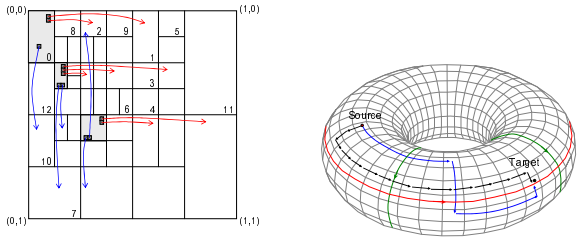
Each region r os the key space is given a globally unique identifier $(r.id)$ generated by applying a hash function to a reference point from r itself. Actually, an identifier of a region, may not bear any semantic, yet it should distinguish the region during all its life-time. In RCAN, the reference point of a region is the smallest point that may possibly belong to that region[1]. The goal it to guarantee that the reference point does not change when the region is split or merged.

Level of a region

RCAN is a decentralized self-organizing content addressable network. Multiple splits and merges may occur independently at the same time and at different locations in the key space. As consequence, regions with different extents may coexist. To keep track of the evolution of the key space, each region r is associated a positive integer $r.l$ (called *level*) that indicates the number of partitioning operations (split and merge) the region r has undertaken[2]. The next

1: In $2D$ RCAN (figure 1(a)), the reference of a region corresponds to the top-left corner of its bounding rectangle.

2: Conceptually, the level represents the depth of a region in the *virtual parti-*



(a) Long links model (b) Torus model

Fig. 1 2D RCAN overlay. (a) Multi-ring infrastructure; (b) Routing from *source* to *target* in CAN (*black arrows*) and RCAN (*blue arrows*).

dimension along which the region should be split (resp. merged) is inferred from the region's level. Specifically, the next dimension to splitting (resp. merging) r is ($r.l \mod d$) (resp. $r.l - 1 \mod d$).

Example: In figure (Fig. 1(a)), nodes 3 and 4 are sibling nodes, but 2 and 8 are not. The levels of nodes 11, 7, 10, 3, and 6 are 3, 4, 5, 6, and 7 respectively.

Split

The bounding hyper-rectangle of a region r is subdivided into two equal distinct regions along one dimension i'. The extent of r is collapsed to cover only the first half. The second half is assigned to a new sibling region s. Data items that fall into the second half are also transferred to s. After the split, the level of r is incremented by one. The level of s takes the same value as r's level.

Merge

It is the inverse of split operation. Two sibling regions r and s are combined into one big region that covers both of them. One of the two regions is extended to take the place of the new region, (let us say r). The other region (s) is deleted after transferring its responsibility to r. Finally, the level of r is decremented by one.

Assigning data items to regions

As stated above, our aim is to efficiently support semantic queries (*i.e.* range queries, nearest neighbor search, etc.) over multidimensional data. The idea is to use a *locality preserving mapping* that places data items in the key space according to their *semantic* (Spatial coordinates of data points in the $3d$ space, for example). Locality preserving mappings ensure that items with similar attribute values are assigned to the same region, or at least stored in the nearby.

CAN [3] and Chord [2] are examples of DHTs that employ hashing techniques to assign IDs to items and nodes. Unfortunately, this randomization destroys data proximity and can only support exact match queries. Instead of that, RCAN operates on a native key space where data items are mapped directly to regions according to their attributes values. As discussed above, this is very essential to efficiently support semantic queries. However, in the absence of effective load balancing techniques, this may result in a *highly* imbalanced partitioning where some regions store a large amount of data, while others are store very few items (or empty in the worst case). The situation may become arbitrarily bad if the data distribution in not uniform. To cope with this problem, RCAN proposes simple and efficient load balancing techniques that cope with load imbalance caused by changes in the network composition due to nodes joins (*static load balancing*) [9].

Assigning regions to nodes

Conceptually, RCAN builds a d-dimensional overlay on top of an evolving set of computers (*nodes*) connected through a physical communication network. Actually, regions in RCAN are assigned to nodes using *one-to-one* mapping, which is independent of the structure and composition of the underlaying network. Each node p in the network owns one region r in the whole key space, and is responsible for the data items covered by r. The logical address of p is taken to be the identifer of its region ($p.id = r.id$). Other sophisticated mappings can also be employed in RCAN, since our definition of region is logical and independent of the physical network.

3.2 Multi-ring routing overlay

For routing purpose, each node in the system maintains routing information (*routing state*) about its neighboring nodes which consists of two types of links[3]: *short* and *long*.

Short Links

A node maintains contacts with $O(d)$ adjacent neighbors on average. Short links are maintained by exchanging heartbeat messages between neighboring nodes.

Long Links

They are at the heart of the design of RCAN. The routing state of a node in RCAN is augmented with a few unidirectional links towards nodes in the system that are at distance inverse of the power of 2 on the key space. Actually, the number of links maintained by a node along each dimension is equal to the number of times its region had split along that dimension. As result, the total number of long links at each node is always proportional to the size of its region. This property enables the implementation of self-scaling routing tables, since a node can adapt dynamically the number of long links by establishing additional links when its region splits, or dropping extra links when its region shrinks. With high probability, the total number of long links per node is tightly bounded by $O(\log N)$ [1].

Multi-rings routing infrastructure

Long links are parallel to data axes and wrap around along each dimension. Long links originating from nodes whose origins are situated on the same line form a ring. Multiple small sub-rings are hence formed along each dimension. In a regularly partitioned grid, a node is a member of one ring along each dimension (*i.e.* the ring passing by its origin). The number of rings and the number

tioning tree.

[3] When we say that a node p has a link towards a node q, this means that a direct communication channel is established between the two nodes, and through which p can send messages to q. A link is materialized by the network address, region extent, etc. of the node towards which the link points.

of nodes per ring self-scales as the network size changes. In case where regions may have different sizes, a node may become temporary member of more than d rings, because its region is large and might intersect with more than one ring along each dimension.

Example: Nodes 0, 8, 2, 9, 5, and 11 are members of the first horizontal ring. Nodes 0, 12, 10, and 7 are members of the first vertical ring. Node 11 owns a big region, it is thus member of other horizontal rings like the one passing by nodes 12, 6, 4, etc.

When a node p splits its region with a new node q along dimension i.Node q becomes the successor of p on the same ring along dimension i. On the other dimensions, q becomes a member of the ring adjacent (in the positive direction) to p's ring and takes the same position as p on its ring. If the level of p is the highest in its ring, this means that the adjacent ring where q will join does not yet exist. In this case a new ring is created and q becomes the first member in it.

RCAN's multi-rings infrastructure is a virtual model to organize long links. Rings are built using existing long links and do not incur additional building or maintenance overhead. This multi-ring routing model is highly flexible. Rings are created or removed dynamically with almost no extra cost. The number of rings and their sizes self-adjust to reflect changes in the network size or nodes distribution.

Routing mechanism

RCAN adopts a *hop-by-hop* greedy routing approach. A node uses only local routing state to decide the next routing step. In our design, a message is identified by a multidimensional key specifying the coordinates of the target point. During a routing task, a node looks-up in it routing table for a neighboring node (either immediate or distant) that is strictly closer to the target, according to a well defined metric, and forwards the message through that neighbor. If there are many neighbors at the same *expected* distance to the target, one of them is selected at random.

Intuitively, the routing task in RCAN consists of solving the routing path along one ring at each step. Rings can be used in an arbitrary order. Another good feature of RCAN's multi-ring topology is that there are many paths with almost the same expected distance between any pair of nodes. This property enables more routing flexibility and robustness against nodes and links failures.

4. Distributed query processing

RCAN provides efficient supports for exact-match queries, range queries and partial-match queries. Formally, a range query $Q(l, u)$ is a d-dimensional hypercuboid delineated by the pair $< l, u >$, where $l = (l_1, ..., l_d)$ and $u = (u_1, ..., u_d)$ define the minimum (*lower*) and maximum (*upper*) points of the query range, respectively (cf. Fig. 2). Given a range query $Q(l, u)$, the answer should contains all the data items whose attribute values are in the specific range $[l, u)$.

Exact-match query (also called *point* query) is a special case of range queries where the lower and upper bounds of the query range coincide, that is $l_i = k_i = u_i$; for $i = 1...d$. The evaluation of point query consists simply of a lookup message sent to the node that covers the query point using the overlay lookup service.

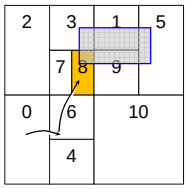


Fig. 2 2D range query $Q(< 0.47, 0.81 >; < 0.65, 0.85 >)$

The evaluation of a range query is, however, much more complicated than a point query. This is mainly due to the lack of global knowledge about the actual composition of the network, and its dynamic nature due to nodes churn. The challenge in this settings to resolve a range query is how to refine and propagate the query to relevant nodes in and efficient and deterministic way. The proposed mechanism should enable to visit all nodes which are relevant to the query, once and only once, while satisfying the following properties:

- Reduce the number of visited (*non-computing*) nodes, that's nodes whose regions do not intersect with the query range, yet they may participate in the routing process.
- Reduce the number of messages necessary to solve the query, as well as the total time to answer the query (*latency*)
- Balance the workload among the processing nodes.

If we assume that the entire key space was evenly divided into regions of the same level. The querying node (*initiator*) could easily partition the query into a set of sub-queries, and then route independently each sub-query to a relevant node. Besides relying on static partitioning of the key space, this method is obviously very inefficient when the query range spans over a large number of zones (large query selectivity). Sub-queries might also traverse through many irrelevant nodes, which causes a very high network overhead and excludes eventual intra-query optimizations.

4.1 Range query

When an initiator node receives a range query $Q(l, u)$ from its application, it routes the query to a node p whose region intersects with the query range. Besides the query range, the query message embeds also the network address or simply the coordinates on the key space of the initiator node, so that the answers can be easily routed back "directly" to the initiator node.

The query initiator happens to be any node in the system. Without loss of generality, let us assume that it is the node covering the lower bound of the query range, l. A naive method to propagate the range query is to broadcast the query message from one node to its adjacent neighbors until reaching the other boundary of the query range. This method guarantees that all the relevant nodes are visited *at least* once at a lower message cost. However, the query latency

may be high in particular for large query selectivity.

In RCAN, the query evaluation is a recursive process where the query is progressively refined and delivered to relevant nodes through neighboring nodes. Specifically, processing a range query consists of two steps: First, routing the query message to a node that is involved in the query range, then decomposing the query into smaller sized sub-queries and propagating them to their corresponding nodes.

a) Query routing

A node on the propagation path that receives the query message checks whether its local region intersects with the query range. If not, it simply routes the query message towards the node that covers the lower bound of the query range.

b) Query propagation

When the query message reaches a node n that is involved in the query range, node n determines if its own region covers the whole query range, in which case it evaluates the query over its local data and sends the answer back to the initiator node. The process of query propagation stops here as well. Otherwise, node n processes only the part of the query range that intersects with its region, and forward the remainder to its adjacent neighbors that are also relevant to the query. The propagation of the query message continue until reaching the other boundary of the query range. Since RCAN schema preserves data and region proximity when mapped into the overlay. The range query can be answered in a straightforward manner by visiting only neighboring node.

Note also an important property for range query, we called "*query locality*", which is the ability to efficiently traverse in between nodes whose regions are adjacent on the native multidimensional space. This means that once entering the query region (starting from any boundary) the query propagation pathways will not get out of this range when forwarding the query message to other relevant nodes. The *query locality* property is considerably useful to reduce the number of *non-computing* nodes to be visited, and consequently the overall query latency. Other exiting methods such as BATON, P-Grid, and ZNet do not fully satisfy the query locality property.

4.1.1 Speedup the query propagation

In case of large query range, the propagation of the query message through adjacent neighbors may result in higher latency and makes it more vulnerable to failures. In order to speedup the query propagation process, it would be advantageous to forward the query message also along the long-range contacts. Figure 3 illustrates this idea. In this simple example node 8 would forward the query message directly to node 5.

4.1.2 Analysis

As in the case of a simple point query, a range query message takes $O(\log N)$ steps to reach the first node that is involved in the query range. Thereafter, additional nodes will be discovered at a cost of one message each. Therefore, to answer a range query, with the range covering M nodes, the first scenario requires an amortized

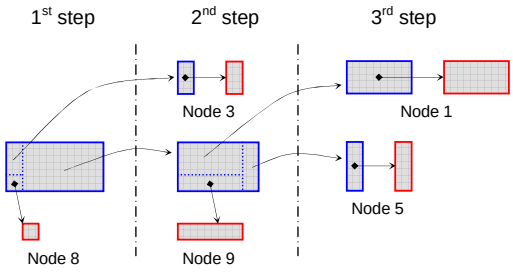


Fig. 3 Range query evaluation.

cost of $O(\log N + M^{1/d})$ hops, while the second scenario requires only $O(\log N + \log M)$ hops. While using the same number of message to solve the range query, the query latency in other systems (like BATON) is linear to the number of covered nodes (M), that is $O(\log N + M)$ hops.

4.1.3 Optimization

In order to further reduce the query latency, the query decomposition can be anticipated before reaching the first node that is involved in the query range. In this sophisticated scenario, the query range can be partitioned, by any node on the propagation path, into small sized sub-queries which will be forwarded separately to their relevant nodes. The query decomposition by intermediary nodes rely on their expected knowledge about target nodes' locations on the key space. Figure 4 illustrates an example of solving the same range query as above using this optimized schema.

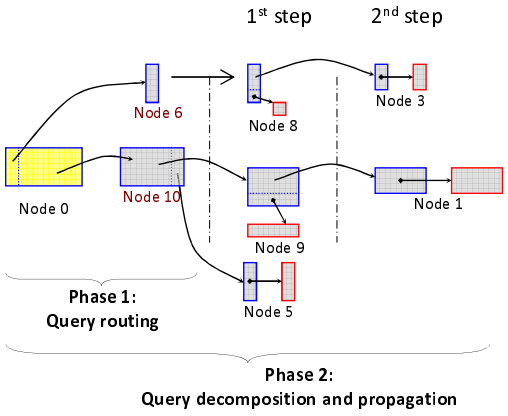


Fig. 4 Range query evaluation (Optimization).

Obviously, additional network overhead may be incurred in order to route the sub-queries through separate paths. However, this strategy allows further reduction in the query latency. To avoid visiting too many non-relevant nodes, and hence less query refinement burden during the routing phase, the query is always routed towards the neighboring nodes whose regions intersects with the query range or closer to it than other neighbors.

A trade-offs between the communication overhead and latency would be to limit the number of parallel query propagation paths that can be generated by any intermediary node to k (*outbound*). The parameter k may be application-dependent or estimated depending on the actual node workload and available bandwidth.

Larger values of k are desirable for mission-critical and high-performance systems. Smaller values are, however, more suitable for resource-limited nodes (e.g bandwidth). The value k can not exceed in any case the *out-degree* of a node which is logarithmic to the network size [1].

4.1.4 Query termination

Upon receiving the answer messages, the query initiator combines all the sub-answers to form the total answer for its query. In the meanwhile the initiator node checks whether the query range is fully covered by the combined answer. If some part remain uncovered, sub-queries corresponding to the missing parts are generated and routed towards their respective nodes. The final answer of the query will be the union of all the answers it receives.

4.2 Partial-match query processing

The above mechanism to solve a range query can be adapted in a straightforward way to support partial much queries as well. The idea is to rewrite the partial-match query into a range query. A wildcard, that is a boundary which is not specified on a dimension in the partial-match query, is replaced by the boundary of the domain on this dimension.

5. Conclusion and Future work

RCAN is a dynamic distributed indexing framework for efficient management of large multidimensional datasets over massively decentralized P2P environments. Our main focus in this paper is to efficiently support complex multidimensional queries in structured P2P. We presented an extension of RCAN protocol to supports efficient evaluation of range queries and partial-match queries. The proposed mechanisms fulfills all the desired properties, namely: (1) reduce the number of visited nodes; (2) reduce the number of messages needed to solve the range query; (3) and reduces the query latency by exploiting the *query locality* property. Most of the existing methods fail to satisfy properties (1) and (3) simultaneously as they do not preserve the *query locality*. The communication and time complexities of our range query processing algorithm are both logarithmic to the number of nodes that intersect with the query range. This parameter depends essentially on the query selectivity and the distribution of nodes.

We are currently deploying the proposed range query processing mechanism on top of our RCAN protocol to validate it under realistic settings and investigating further optimizations of the query engine. We are also working to extend our protocol to support other non-trivial queries, such as spherical range queries, nearest neighbor searches and skyline queries.

The next step will be to study the robustness of our query evaluation mechanisms in the presence of high rate of nodes' churn by providing a *best-effort answer* for given query.

Acknowledgment

This study has been partially supported by Grant-in-Aid for Scientific Research on Priority Areas from MEXT (#21013004).

References

[1] D. Boukhelef and H. Kitagawa, "Multi-ring infrastructure for content addressable networks," Proc. of CoopIS, pp.193–211, 2008.

[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," IEEE/ACM Transactions on Networking, vol.11, no.1, pp.17–32, 2003.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. of ACM-SIGCOMM, pp.161–172, August 2001.

[4] K. Aberer, "P-grid: A self-organizing access structure for P2P information systems," Proc. of CoopIS, pp.179–194, 2001.

[5] A.R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," Proc. of ACM SIGCOMM, pp.353–366, 2004.

[6] H.V. Jagadish, B.C. Ooi, and Q.H. Vu, "Baton: A balanced tree structure for peer-to-peer networks," Proc. of VLDB, pp.661–672, 2005.

[7] H.V. Jagadish, B.C. Ooi, K.L. Tan, Q.H. Vu, and R. Zhang, "Speeding up search in peer-to-peer networks with a multi-way tree structure," Proc. of ACM SIGMOD, pp.1–12, 2006.

[8] H.V. Jagadish, B.C. Ooi, Q.H. Vu, R. Zhang, and A. Zhou, "VBI-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes," Proc. of ICDE, p.34, 2006.

[9] D. Boukhelef and H. Kitagawa, "Dynamic load balancing in RCAN content addressable network," Proc. of ICUIMC, pp.98–106, 2009.

[10] M. Cai, M.R. Frank, J. Chen, and P.A. Szekely, "Maan: A multi-attribute addressable network for grid information services," Journal of Grid Computing, vol.2, no.1, pp.3–14, 2004.

[11] S. Wang, Q.H. Vu, B.C. Ooi, A.K.H. Tung, and L. Xu, "Skyframe: a framework for skyline query processing in peer-to-peer systems," VLDB Journal, vol.18, no.1, pp.345–362, 2009.

[12] A. Datta, M. Hauswirth, R. John, R. Schmidt, and K. Aberer, "Range queries in trie-structured overlays," Proc. of IEEE P2P, pp.57–66, 2005.

[13] P. Ganesan, B. Yang, and H. Garcia-Molina, "One torus to rule them all: Multidimensional queries in P2P systems," Proc. of WebDB, pp.19–24, 2004.

[14] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, S. Shenker, and J.M. Hellerstein, "A case study in building layered dht applications," Proc. of ACM SIGCOMM, pp.97–108, 2005.

[15] Y. Shu, B.C. Ooi, K.L. Tan, and A. Zhou, "Supporting multi-dimensional range queries in peer-to-peer systems," Proc. of IEEE P2P, pp.173–180, 2005.

[16] C. Schmidt and M. Parashar, "Squid: Enabling search in dht-based systems," Journal of Parallel Distributed Computing, vol.68, no.7, pp.962–975, 2008.

[17] E. Tanin, A. Harwood, and H. Samet, "Using a distributed quadtree index in peer-to-peer networks," VLDB Journal, vol.16, no.2, pp.165–178, 2007.

[18] V. Kantere, S. Skiadopoulos, and T.K. Sellis, "Storing and indexing spatial data in p2p systems," IEEE TKDE, vol.21, no.2, pp.287–300, 2009.

[19] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Commnications Surveys and Tutorials, vol.7, no.2, pp.72–93, 2005.

[20] M. Yu, Z. Li, and L. Zhang, "Supporting multi-attribute queries in peer-to-peer data management systems," Proc. of PDCAT, pp.515–522, 2007.