Complex Queries for Moving Object Databases in DHT-based Systems

Cecilia Hernández[1], M. Andrea Rodríguez[13], and Mauricio Marin[23]

[1] DIICC, University of Concepción, Chile
[2] Yahoo! Research, Santiago, Chile
[3] Center for Web Research, Chile

Abstract. Distributed moving object database servers are a feasible solution to the scalability problem of centralized database systems. In this paper we propose a distributed indexing method, using the Distributed Hash Table (DHT) paradigm, devised to efficiently support complex spatio temporal queries. We assume a setting in which there is a large number of database servers that keep track of events associated with a highly dynamic system of moving objects deployed in a spatial area. We present a technique for properly keeping the index up to date and efficiently processing range and top-k queries for moving object databases. We evaluated our system using event-driven simulators with demanding spatio temporal workloads and the results show good performance in terms of response time and network traffic.

1 Introduction

Spatio-temporal databases use index structures tailored to answering specific types of queries. Among these queries, *time-slice* and *time-interval* queries (also known as range queries) have been usually the focus of investigation. This is also true for distributed spatio-temporal databases, where previous works have solved range queries by using a global and distributed spatio-temporal index that organizes servers in terms of spatial and temporal partitions [4, 7, 10, 12]. These strategies require intensive coordination, which limits their scalability and adaptability.

This work aims at solving not only time-slice and time-interval queries, but also queries about the *current* and *historical* locations of particular objects (i.e., object-location queries), aggregation queries about the number of objects per server, and top-k queries including the top-k servers with largest number of objects and the top-k objects with largest trajectories within a spatial and temporal window. Solving different types of queries imposes particular challenges for large scale and distributed systems, since data distribution needs to accommodate different algorithms for query processing. For example, for solving queries about the location of particular objects, the classical distribution of data based on spatial and temporal data is insufficient.

In this work, we propose to handle distributed spatial-temporal data in a DHT-based network that combines a distribution of objects' trajectories (traces) and spatial partitions, which enables exact and approximated (fast) answers to object-location, range (time slice or time interval), aggregation and top-k queries. In the context of DHT-based distributed systems, such as Chord [13] and Kademlia [9], the paradigm provides hash table semantics, where a DHT interface implements a hash function that maps any given data key to a particular peer through the operations $put(key, data)$ for insertion and $get(k)$ for retrieval. In these systems, lookup is oriented to solve efficiently exact-match queries, usually in $O(\log N)$ hops, where N is the number of peers in the network. However, complex queries like range and top-k are particularly challenging in DHT systems. There have been recent works addressing range [3, 11, 14] and top-k [1] queries for spatial databases, but as far as we know, spatial-temporal queries have not been proposed using DHT systems.

In our scheme, we have a distributed *meta-index* that contains information about data distributed in several and independent spatial-temporal database servers that register all events associated with the moving objects deployed in the spatial area. The meta-index contains sparse data about the location of particular objects, the geographic extent containing objects stored in servers at different time instants, and statistical information about servers. Peers periodically poll servers to keep properly updated the distributed meta-index, which acts as an entry point to the database servers. Smart polling is esential to keep a good approximation of the real system in the index.

We have reported work on object-location queries and aggregation queries, first for a centralized meta-index in [8], and then for a distributed meta-index on a p2p network in [6]. In this paper we extend the p2p meta-index proposed in [6] to include spatio-temporal range queries and top-k object queries. To do so, we complement the data distribution in [6] and design search algorithms for these queries. In addition, we propose a fully distributed scheme to perform polling by assigning this task to a sub-set of peers. In this way peers forming the crawler (i.e., polling system) can deliver their data to any other peer and clients can submit a richer set of queries to any peer. An advantage of distributing the polling process among several peers is a better use of the collective bandwidth and, consequently, a faster update of the data stored in the meta-index.

The organization of the paper is as follows. Section 2 presents the meta-index distribution. Section 3 describes the full distributed polling strategy for updating the meta-index. Section 4 presents search algorithms for range and top-k objects queries. Evaluation and conclusion follow in sections 5 and 6, respectively.

2 Meta-Index Distribution

The meta-index stores partial data about the time-varying location of objects. These data include: the time-varying number of objects per server (statistical or aggregated information), the time-varying geographic extent that contains the location of objects in a server, and the coarse traces of objects' visits across

servers. Note that the coarse traces of objects in the meta-index are not "real" sparse trajectories, but lists of locations (database servers) that the objects have visited sorted by the time instant of the data collection performed by the crawler.

With the purpose of answering object-location queries and aggregation queries, we propose in [6] two strategies for distributing the meta-index data with Chord protocol and using the available bandwidth effectively. On one hand, we introduce the concept of *MSB (Most Significant Bits)* to map moving object trace data to Chord keys. The idea of the MSB is that a crawler can compound moving object data in only few *composite objects* by using the most significant bits of object ids. Each composite object contains a set of moving object ids with their respective traces. On the other hand, since the amount of aggregated data per update message is very small, we piggy it back to composite objects. This scheme allows a crawler to send fewer update messages into the peer-to-peer network avoiding overloading the network.

To support range queries (time slice and time interval queries), we also distribute information of the spatial organization of servers using a Z-curve [5]. To do so, a general square grid divides the complete geographic space. We do not impose restrictions on cells in this grid and the dynamic geographic extent associated with each server at different time instants. Thus, the geographic extent may partially cover several cells, or a cell may include the extent of several servers.

The space-filling Z-curve maps a unit line segment to a continuous curve in the unit square. In our case, the Z-code represents the *key* for a range or spatial window, which is used by the Chord protocol to distribute spatial information. The distribution of data based on the dynamic geographic extent handled by servers complements the partial trace information organized into composite objects described above. Whenever a robot (peer) of the crawler informs new updating information (i.e., server, time, number of objects, geographic extent and composite objects), in addition to distributing the composite objects, this robot also sends information to peers whose keys (Z-codes) map to cells in the geographic space overlapping the server's extent. These peers store the server, time, extent, number of objects and composite ids of an inform (Figure 1). Notice that the we do not duplicate the composite, only the composite ids, which are latter used to retrieve objects' ids or traces in approximated answers to range queries.

The meta-index distribution strategy requires the administration of different types of messages. *S-messages* (messages 1 in Figure 1) carry control data in the process of finding the corresponding peers where to store composite objects. S-messages also carry aggregated data about the number of objects per type that were found by the robots at specific servers and time instants, which are stored in all peers crossed until finding the destination peers. The idea behind this strategy is to have an overall view of the global statistics of the system in each peer [6]. *T-messages* (message 2 in Figure 1) transfer composite objects from the entry peers to the destination peers. *R-messages* (message 3 in Figure 1) carry Z-codes together with the server id, crawling time, number of objects in the server, and list of composite ids for the meta-index update.

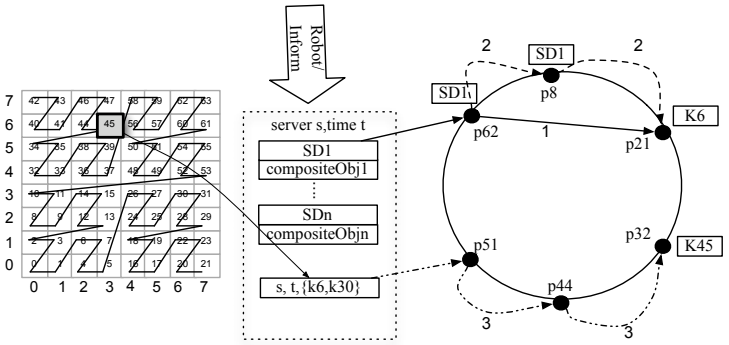


Fig. 1. Spatial and trace distribution on p2p network

3 P2P crawling (polling)

A centralized crawler, as those devised for Web search engines, is composed of a set of so-called robots that are in charge of contacting servers to download data. In the p2p context, the principle is that machines in a centralized crawler are mapped into clusters of peers that work collaboratively to poll database servers. Thus, we split the work effected by a virtual centralized crawler among several peers per database server. A set of peers plays the role of a set of robots, whereas others peers are coordinators.

We model this problem as a metric space where dimentions are given by restrictions that peers must satisfy in order to guarantee efficient performance. A *metric space* $(\mathbb{X}, d)$ is composed of an universe of valid objects $\mathbb{X}$ and a *distance function* $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$ defined among them. The distance function determines the similarity between two given objects and satisfies triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$.

In our scheme, the collection of database servers forms a set of centers and crawler peers assign themselves to the sphere of influence (c, r_c) of the center c (server) that is closest to them in terms of the distance function. The value of r_c indicates the number of connections with peer robots the server may accept. This value depends on the relative importance and observed activity in the server with respect to all other servers. Coordinators periodically determine the values of r_c by computing the dynamic ranking of database servers along time, as we proposed for the centralized crawler in [8]. Basically, this method ranks servers considering how active objects are in their domain and what their relative importance is in terms of average number of objects. Server ranking values are between 0 and 1. The value of r_c is $\max\{m, n\}$, where m is the total number of simultaneous connections that servers can accept, and $n = \text{rank}(c) \cdot N$, with N being the average number of moving objects detected in the period.

Peers willing to become crawler robots can get information about servers from selected coordinator peers holding global information about database servers. A

peer robot gets a list of servers IDs from a coordinator peer and calculates its distance to them. Then the peer sends back to the coordinator a request to join one of the k closest servers, and the coordinator decides to accept or reject the peer as robot. The peer q asks joining one of the clusters c based on (1) the increasing order of $d(q, c)$ and (2) whether or not the number of peers assigned to the cluster is less than r_c. Notice that $d(q, c)$ does not only consider the distance between the peer q and the server c measured as the bandwidth between the two, but also a linear combination of the inverse of the sum of all distances between q and a sample of the current peers assigned to the cluster, distances measured in terms of the number of hops required by the DHT to go from q to the sample peer. The objective is to privilege peers which are more distant each other in order to distribute the communication more evenly across the network.

4 Query Processing Algorithms

We propose a two-phase algorithm for processing range and top-k queries (Algorithms 1 and 2, respectively). In the first phase, the query extent is mapped into Z-codes. Peers associated with these Z-codes are then visited to obtain tuples of the form *(server, time, list of composite ids)*. These tuples are obtained by checking the updates of each server in the peer and selecting the closest update in time for each server that overlaps the query extent.

In a second phase, we have two options. For an approximated answer, that is, answers that only use the meta-index, all composite ids selected in the first phase are grouped in a random peer to eliminate duplicates and optimize the visit to peers containing these composites. When visiting these peers, the process retrieves objects whose closest location (server) in time to the query is one of the servers that intersects the query spatial window. For a more accurate answer, we only use the meta-index to guide the search to the appropriate servers, which are selected in the first phase. So, in the second phase, we access local servers. For top-k queries about the number of objects per server, we can get good approximated answers by ranking servers in terms of the number of objects stored, as global statistical information, in each peer, or by accessing the number of objects stored with the R-messages in peers organized by the Z-curve.

5 Experimental evaluation

Our simulation environment used two event-driven simulators. One simulates the database servers and crawling generating the data in the form of the meta-index. The other simulates the Chord protocol to allocate and lookup meta-index data. We also used the network simulator NS-2 (*http://www.isi.edu/nsnam/ns*) in tandem with GT-ITM (*http://www.cc.gatech.edu/projects/gtitm*) to simulate the network topology and environment to measure the performance.

We used workload data generated by a public spatio-temporal dataset generator; the Network-based Generator of Moving Objects (NGMO) [2]. The data set contained 50,000 initial moving objects, existing around 150,000 along the

Algorithm 1 Algorithm to process *range time slice or time interval* queries using the p2p meta-index

1: // Query: find all objects os that were in a region R at a time instant t or in a time interval $[t1, t2]$.
2: First Phase
3: QP, the peer where the query starts, gets all Zs that intersect R.
4: **for** z in Zs in parallel **do**
5: DHT $get(z)$ and returns to QP the list LC of composite objects associated with updates from servers that intersect the query window (time and space).
6: **end for**
7: QP processes LC (combining and eliminating duplication) and determines all Composite Objects to visit (COs)
8: Second Phase
9: **for** co in COs in parallel **do**
10: DHT $get(co)$
11: **end for**
12: QP assemblies the answer and replies to the client

Algorithm 2 Algorithm to process *Top-k time interval* queries using the p2p meta-index

1: // Query: find Top-k objects os with the longest trajectory found in a region R at a time interval $[t1, t2]$.
2: First Phase
3: QP, the peer where the query starts, gets all Zs that intersect R.
4: **for** z in Zs in parallel **do**
5: DHT $get(z)$ and returns to QP the list LC of composite objects associated with updates from servers that intersect the query window and the number of times these composite objects form part of updates during the query time interval.
6: **end for**
7: QP ranks Composite Objects to visit based on the number of times they are part of updates for the query time interval and spatial range (COs).
8: Second Phase
9: QP groups composite objects with the same score and create $coSets$
10: **while** there is a $coSet$ to visit or the difference of object trace length $df <$ threshold **do**
11: **for** co in $coSet$ in parallel **do**
12: DHT $get(co)$
13: **end for**
14: Compute df
15: **end while**
16: QP assemblies the answer and replies to the client

simulation time. We used a network topology in similar way as seen in [14]. We chose peers and clients randomly from stub nodes defined in a transit-stub network of 1168 nodes created with NS-2/GT-ITM.

A first evaluation was a comparison between results with the centralized crawler evaluated in [8] and the results obtained with the p2p crawler proposed

in this paper. The differences in speed of crawling depend on the number of peers we deploy to be robots in our simulator. We observed through experimentation that we needed about ten times more peers than central robots, point in which we start to observe idle peers. We also observed that the distributed calculation of the ranking of servers performed by the coordinator was about 10% different from that calculated by the centralized crawler. Overall we observed that the peer robots were evenly distributed across the clusters in a proportion similar to that given by the ranking of servers. Our linear combination factors for distances between peer and servers, and peer and cluster mates, was 0.6 and 0.4 respectively with distances normalized to 1. The NS-2/GT-ITM simulator delivered the following results for the average cost of peer to peer communication during polling.

Number of robot peers	160	320	640
Average Response time (ms)	0.109	0.178	0.258
Number of hops for DHT	4.915	5.058	4.223

Spatio-temporal Distribution. We evaluated the time efficiency of answering queries using the z-curve for the spatial distribution on the p2p network with respect to an ideal scenario where we distribute an R-tree that contains all objects at the query time. Note that the latter simulates a global distributed spatio-temporal indexing structure for the location of objects in every query time, which indeed it is the most time efficient structure, but a very inefficient structure from a storage-cost point of view. Figure 2(a) shows the ratio between the time cost and overhead cost of searching with the R-tree with respect to the z-curve distribution on the p2p network (good values are close to 1.0). We calculated costs in answering 100 random time-slice queries.

For answering both range and object-location queries, our proposal groups composite ids and not object ids in the spatial distribution using the Z-curve.This implies that for range queries we have to access peers where the composite ids are stored to retrieve their corresponding object ids. This disadvantage produces a clear increment in the number of messages of our proposed distribution meta-index. Our proposal, however, competes closely in time cost with the ideal scenario of the distributed R-tree.

Quality of approximated answers. We show experimental results for queries that are solved using only the data stored in the meta-index and establish comparisons with the correct results obtained by visiting all the database servers. Figure 2.b shows the percentage of correct objects in the answers to 100 random time-slice and time-interval queries, and the Pearson correlation when ranking objects by their trajectory length in 100 random time-interval queries (top-k object queries). The values 16, 32 and 64 represent the number of robots used by the crawler. For time slice and time interval, we also show the quality of results when using the meta-index to guide the search to local servers (MI-DBS), results that are almost perfect (over 99%).

Overall, the results show that there are many cases in which the data stored in the meta-index can be used to provide a very fast and good approximated

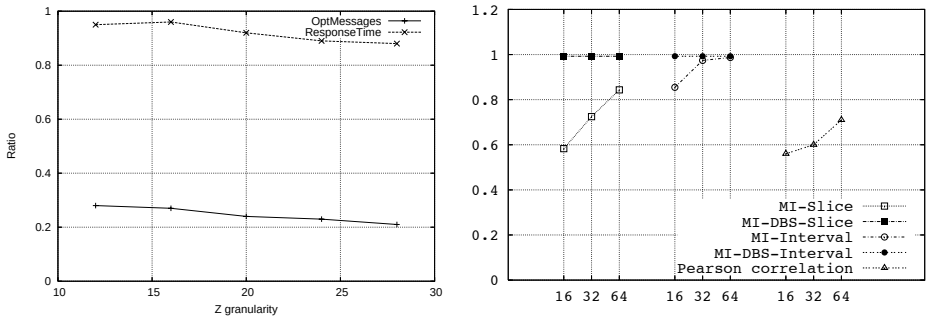


Fig. 2. Distributed meta-index: (a) the distributed meta-index versus a temporal fixed distributed R-tree, (b) quality of approximated answers

answer to classical queries for this kind of database systems. Users can use this preliminary "hint" to further investigate the system evolution or to refine their subsequent queries.

Update/Search Performance. To measure the performance of the system, we first evaluated the impact of search parameter variations on the type of queries we address in this work. Figure 3 shows the results for search parameter variations in terms of response time and communication cost. The communication cost is measured in terms of the number of messages needed per query. We computed the number of messages using the Chord Protocol. The results show that both metrics increase slightly when increasing the size of the spatial window, time interval and k, being the size of the spatial window the most sensitive. Second, we measured the communication cost to keep the meta-index up to date under different numbers of robots. Figure 4.a shows that 32 robots are enough to collect data for 150,000 moving objects. In this case, MSB was 20, since higher values for the MSB make composite too small and waste bandwidth. Third, we evaluated the elasticity of the system in terms of its ability to scale up when having to respond to concurrent queries. Figure 4.b shows that the system increases its response time with a slightly higher slope over 200 queries per second.

6 Conclusions

In this paper we have presented the design of a p2p index data structure devised to support complex queries in moving objects databases. Our main contributions are: (1) to combine, into the same index location, aggregation and range queries, and (2) to update the index based on a distributed strategy of crawling.

Area	Message	Time
0.25%	21.16	2623.6
1%	22.75	2600.8
4%	36.14	3047.1

(a)

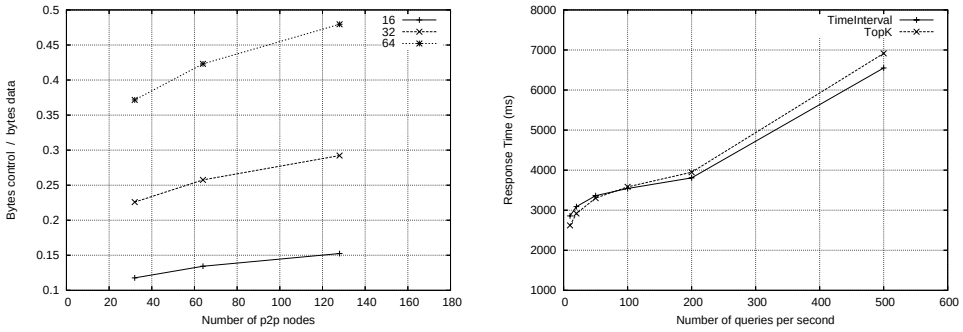
Interval	Message	Time
5%	25.22	2867.9
10%	26.15	2910.7
20%	28.04	2861.0

(b)

TopK	Message	Time
10	15.9	2636.3
20	25.2	2882.8
30	29.4	2888.1

(c)

Fig. 3. Effect of search parameters: (a) spatial window size (b) time interval length and (c) k (*Area* is the percentage of the total geographic region, *Interval* is the percentage of the simulation period, K is the parameter of top-k queries, $Message$ is the average number of messages per query, and $Time$ in the average time measured in milliseconds)



(a) (b)

Fig. 4. Communication overhead and elasticity of using the meta-index: (a) relative number of bytes transmitted in update operations, (b) response time for different query rates given to the system for time interval and top-k queries (MSB=20, 32 robots and 128 peers)

The experimental results show that, at least for the classical range queries based on a spatio-temporal window, our strategy can be as efficient as the well-known RTree, which we set it to work in an extremely convenient setting. To the best of our knowledge, alternative solutions to solve location queries have not been proposed so far by other authors. The use of the p2p index, as a devise to support fast approximated solutions to queries, is also promising. How close the approximated answer is to the exact solution depends on how frequently the index is updated. We have proposed a p2p crawler that performs as efficiently as a centralized crawler. We have found that with 10 peers per robot of a high-performance centralized crawler it is possible to keep the index properly updated along time.

References

1. Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Processing top-k queries in distributed hash tables. In Anne-Marie Kermarrec, Luc Bougé, and Thierry Priol, editors, *Euro-Par*, volume 4641 of *Lecture Notes in Computer Science*, pages 489–502. Springer, 2007.
2. Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
3. Yatin Chawathe, Sriram Ramabhadran, Sylvia Ratnasamy, Anthony LaMarca, Scott Shenker, and Joseph M. Hellerstein. A case study in building layered dht applications. In Roch Guérin, Ramesh Govindan, and Greg Minshall, editors, *SIGCOMM*, pages 97–108. ACM, 2005.
4. C. du Mouza and P. Rigaux. Web architectures for scalable moving object servers. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 17 – 22, New York, NY, 2002. ACM Press.
5. Christos Faloutsos and Shari Roseman. Fractals for secondary key retrieval. In *PODS*, pages 247–252. ACM Press, 1989.
6. Cecilia Hernández, M. Andrea Rodríguez, and Mauricio Marín. A p2p meta-index for spatio-temporal moving object databases. In *Database Systems for Advance Applications, DASFAA*, 2008.
7. H. Lee, J. Hwang, J. Lee, S. Park, C. Lee, and Y. Nah. Long-term location data management for distributed moving object databases. In *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 451–458. IEEE Press, 2006.
8. Mauricio Marín, Andrea Rodríguez, Tonio Fincke, and Carlos Román. Searching moving objects in a spatio-temporal distributed database servers system. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (2)*, volume 4276 of *Lecture Notes in Computer Science*, pages 1388–1401. Springer, 2006.
9. Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2002.
10. Anand Meka and Ambuj K. Singh. Dist: a distributed spatio-temporal index structure for sensor networks. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *CIKM*, pages 139–146. ACM, 2005.
11. Anirban Mondal, Yi Lifu, and Masaru Kitsuregawa. P2pr-tree: An r-tree-based spatial index for peer-to-peer environments. In Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, and Athena Vakali, editors, *EDBT Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 516–525. Springer, 2004.
12. Y. Nah, J. Lee, W.J. Lee, H. Le, M.H. Kim, and K.J. Han. Distributed scalable location data management system based on the GALIS architecture. In *Tenth IEEE International Workshop on Object-Oriented Real Time Dependable Systems*, pages 397–404. IEEE Press, 2005.
13. Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.
14. Egemen Tanin, Aaron Harwood, and Hanan Samet. Using a distributed quadtree index in peer-to-peer networks. *VLDB J.*, 16(2):165–178, 2007.