Racoon: Rapid Contact Tracing of Moving Objects Using Smart Indexes

Rakan Alseghayer
Department of Computer Science, University of Pittsburgh
ralseghayer@cs.pitt.edu

***Abstract*—The affordable mobile sensing technologies and the rise of spatio-temporal applications granted trajectory data and data streaming high importance. This led to the need for efficient data stream and trajectory—both historical and real-time—data processing. Tremendous effort has been put in this context, especially, optimizing index structures for certain crucial operations, such as *trajectory join* and *trajectory similarity*, which are used widely in many monitoring and analytic applications. In this paper, we propose an access structure that optimizes *trajectory joins* for contact tracing and avoidance applications. Our prelim evaluation shows that our proposed spatio-temporal index potentially can enhance the performance of contact tracing applications that require trajectory join operations.**

***Index Terms*—Spatio-temporal Indexes, Access Methods, Indoors Trajectories, Data Streams.**

I. Introduction

Current database systems are optimized for certain workloads. For example, timeseries databases (e.g., InfluxDB [1]) support efficiently timeseries workload, while data stream management systems (e.g., Flink [2]), support analytics over live streams. PostGIS [3] and PG-Trajectory [4] are spatio-temporal extensions to PostgreSQL that support spatio-temporal workloads.

Even though few spatio-temporal systems exist, and many access structures were proposed [5]–[7], optimizing for both temporal and spatial analytical queries is still in need. This is due to the increased number of applications that need fast analytics for real-time streaming of trajectory data.

In the context of COVID-19 pandemic, one crucial application is *contact tracing* (e.g., [8]–[10]), that determines if two individuals have come to close distance of each other within a specified time period.

Contact tracing can be formalized as the detection of trajectory intersections in time and space. We pose that trajectory intersections can be efficiently implemented as *trajectory joins*.

The real-time nature of applications, such as contact tracing, that requires the processing of huge number of trajectories, suggests the need for efficient access methods. Motivated by that need, we propose *Racoon*, Rapid Contact Tracing of Moving Objects Using Smart Indexes, built on

- A system model that supports trajectory stream processing (§II); and
- In memory access structures that optimize trajectory joins for indoors spaces (§III).

II. System model

Our system has two main actors, the moving objects and the centralized authority. The moving objects produce data points that form indoors trajectories. Trajectories are indexed locally at the moving objects side, as well as streamed to the centralized authority, which also keeps a centralized index. The centralized authority keeps track of trajectories that are infected (or potentially infected). Regularly, those infected trajectories are sent to the moving objects side for querying whether the moving object has intersected with the infected ones. If that was the case, then the central authority asks the moving object to send its own trajectory after the contact point, and resend them to all moving objects.

The trajectories arrive at the centralized authority in the form of *micro-batches*. As defined in our previous work [11]–[14], a *micro-batch* is a group of synchronized, and same sized trajectory data points that are streamed over a set of moving objects defined by a timestamp. *Micro-batches* of trajectory data arrive in the form of *(ts, val)*, where *ts* is the timestamp, and *val* is the indoors location information (i.e., <x,y> in an indoors layout instead of <long, lat> in outdoors).

The timestamp part of the trajectory represents the temporal aspect of the data point, and helps in pre-processing the trajectory data points and synchronize the trajectories' streams.

Regarding locations, indoor floors are segmented into zones based on a 2D spatial information (i.e., <x,y> coordinates). A location point of <x,y> is mapped to a zone using hashing.

III. Spatio-temporal Moving Object Index

Our proposed spatio-temporal index for moving objects consists of two auxiliary structures (Fig. 1). The first is a *multi-level index* that serves spatial based look-ups. The second structure is an *inverted index* that speeds up temporal look-ups.

The multi-level index is based on hashing buckets and interval trees [15], [16]. The inverted index also consists of hashing buckets and interval trees. The rationale behind using interval trees is the lack of perfect precision of trajectory points in time and space. Thus, guarding the location information with an interval of time, instead of a definite timestamp, can help answering queries with higher precision. We construct intervals from timestamps, and they consist of low and high values. Those values are derived from the trajectory points' timestamps, where the timestamp = (low + high) / 2.

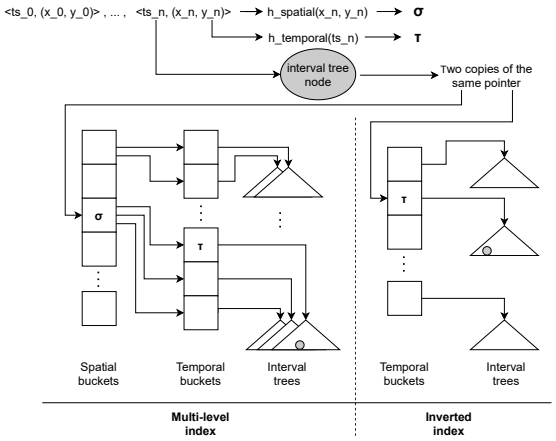

Fig. 1. Insertion of a trajectory point *n* into the *Racoon* index.

Multi-level Index In the multi-level index, the first level consists of hashing buckets based on spatial information. The second level index consists also of hashing buckets based on temporal information grouping (day, hour, etc.). Each temporal bucket in the second level index points at an interval tree. When a trajectory point arrives and an interval is formed, an interval node is created and its pointer is inserted in the corresponding interval tree that is pointed at by the multi-level index.

Inverted Index The inverted index consists of temporal buckets that are based on the same temporal information grouping (day, hour, etc.), as in the multi-level index. Each one of those buckets has a single interval tree that contains all the intervals belonging to the moving object, while they have been in the indoors environment. The nodes of the interval tree are the same ones created for the interval trees in the multi-level index described above.

Index Building Our proposed spatio-temporal index is locally built on the moving object as the data points (trajectory data) are generated by its movement. Each point is used to infer two keys, and an interval. The two keys are a *spatial* key and a *temporal* key (denoted as σ and τ respectively in Fig. 1). Then, an interval node is created, containing the interval, the spatial zone (same as the key), and the anonymized moving object ID. After creating the node, the pointer to that node is used to insert the node into the multi-level and the inverted indexes. The spatial key is used to determine the corresponding spatial bucket, and the temporal key determines the second level index bucket in order to identify the corresponding interval tree at which the interval node will be inserted.

After inserting the pointer into the multi-level index, the same temporal key is used to locate the inverted index temporal bucket. Then, the pointer to the node is used to insert the same node into the corresponding interval tree.

The size of the index depends significantly on the location sampling rate at the moving object, since each data point will be an interval. However, data points that are fixed in the same location can be translated into a single interval (single node).

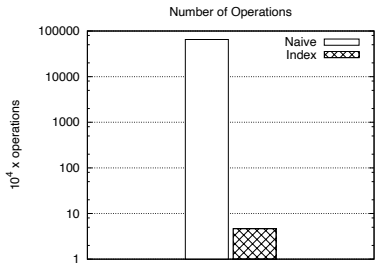

Fig. 2. Trajectory join cost in number of operations at moving objects.

IV. Moving Object Query Types

A very popular operation that is used in many spatio-temporal contexts is trajectory join. As mentioned above, contact tracing is an important application of such operation. Our proposed access structures help answering trajectory join queries efficiently in a logarithmic time.

When a trajectory is received at the moving object to be checked, of whether the trajectory has "intersected" with the moving object in time and space, the trajectory data points are used to infer spatial and temporal keys. Those keys are used to access the first and second levels of the index to reach the interval tree of question. Then the interval tree is searched for the interval that the timestamp from the query trajectory is potentially lays within. This is known as the stabbing problem [17]. If the timestamp of the trajectory point does stab an interval in the corresponding interval tree, then the answer is returned as yes, contact (or intersection) occurred, otherwise the linear scan over the query trajectory points is carried on. In case of a yes answer, the trajectory at the moving object side is streamed to the central authority for broadcasting.

V. Evaluation

In our preliminary evaluation, we compared a naive approach of doing nested loop trajectory join with our proposed index structure. We have used the [18]–[20] dataset which contains 60,949 trajectories. In our experiment, we find whether a certain trajectory has intersected with (contacted) any other trajectory in the dataset by sending the raw trajectory to all the moving objects. We measure the performance in number of operations for both approaches at each moving object (collectively). Our preliminary results show that we have an order of magnitude gain in our performance over the naive approach in number of operations (Fig. 2).

VI. Conclusions

We have proposed a spatio-temporal access method that optimizes trajectory joins which can efficiently support contact tracing monitoring systems. In our prelim evaluation, we have found that our proposed method outperforms the naive approach by orders of magnitude. We are further investigating the usage of GPUs and compression techniques to optimize query answering and the storage of trajectories. Also, we are looking at expanding the querying capability of the system to optimize for other kinds of queries.

ACKNOWLEDGMENT

I would like to thank my advisors Prof. Panos K. Chrysanthis and Dr. Constantinos Costa for their valuable ideas and feedback. Also, thanks to the members of the ADMT (Advanced Data Management Technology) lab for their support and comments.

REFERENCES

[1] InfluxData. Influxdb. [Online]. Available: https://www.influxdata.com/

[2] A. Katsifodimos and S. Schelter, "Apache flink: Stream analytics at scale," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, 2016, pp. 193–193.

[3] PostGIS. Postgis. [Online]. Available: http://postgis.net/

[4] A. Kucuk, S. M. Hamdi, B. Aydin, M. A. Schuh, and R. A. Angryk, "Pg-trajectory: A postgresql/postgis based data model for spatiotemporal trajectories," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*, 2016, pp. 81–88.

[5] W. G. A. Ahmed R. Mahmood, Sri Punni, "Spatio-temporal access methods: a survey (2010 - 2017)," *GeoInformatica*, vol. 23, p. 1–36, 2019.

[6] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel approaches in query processing for moving object trajectories," in *Proceedings of the 26th International Conference on Very Large Data Bases*, 2000, p. 395–406.

[7] Z. He, C. Wu, G. Liu, Z. Zheng, and Y. Tian, "Decomposition tree: A spatio-temporal indexing method for movement big data," *Cluster Computing*, vol. 18, p. 1481–1492, 2015.

[8] A. Gov. Covidsafe app. [Online]. Available: https://www.health.gov.au/resources/apps-and-tools/covidsafe-app

[9] S. Gov. Tracetogether. [Online]. Available: https://www.gov.sg/article/help-speed-up-contact-tracing-with-tracetogether

[10] Y. Park, Y. Choe, O. Park, and et al., "Contact tracing during coronavirus disease outbreak, south korea, 2020," *Emerging Infectious Diseases*, vol. 26, pp. 2465–2468, 2020.

[11] R. Alseghayer, D. Petrov, P. K. Chrysanthis, M. Sharaf, and A. Labrinidis, "Dcs: A policy framework for the detection of correlated data streams," in *Real-Time Business Intelligence and Analytics*, 2019, pp. 191–210.

[12] D. Petrov, R. Alseghayer, M. Sharaf, P. K. Chrysanthis, and A. Labrinidis, "Interactive exploration of correlated time series," in *Proceedings of the ExploreDB'17*, 2017.

[13] R. Alseghayer, D. Petrov, P. K. Chrysanthis, M. Sharaf, and A. Labrinidis, "Detection of highly correlated live data streams," in *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, 2017.

[14] R. Alseghayer, D. Petrov, and P. K. Chrysanthis, "Strategies for detection of correlated data streams," in *Proceedings of the 5th International Workshop on Exploratory Search in Databases and the Web*, 2018.

[15] M. H. Overmars, *The design of dynamic data structures*, ser. Lecture notes in Computer Assisted Diagnosis. Springer, Berlin, Heidelberg, 1983, vol. 3204.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[17] J. M. Schmidt, "Interval stabbing problems in small integer ranges," in *Algorithms and Computation*, 2009, pp. 163–172.

[18] C. Costa, X. Ge, E. McEllhenney, E. Kebler, P. K. Chrysanthis, and D. Zeinalipour-Yazti, "Caprio v2.0: A context-aware unified indoor-outdoor path recommendation system," in *Proceedings of the 21th IEEE International Conference on Mobile Data Management*, 2020, pp. 230–231.

[19] C. Costa, X. Ge, and P. K. Chrysanthis, "Caprio: Graph-based integration of indoor and outdoor data for path discovery," in *Proceedings of the 45th International Conference on Very Large Data Bases*, 2019, pp. 1878–1881.

[20] ——, "Caprio: Context aware path recommendation exploiting indoor and outdoor information," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, 2019, pp. 431–436.