Distributed Learned Hash Table

Shengze Wang[1] Yi Liu[1] Xiaoxue Zhang[1,2] Liting Hu[1] Chen Qian[1]

[1] University of California, Santa Cruz [2] University of Nevada, Reno

Contacts

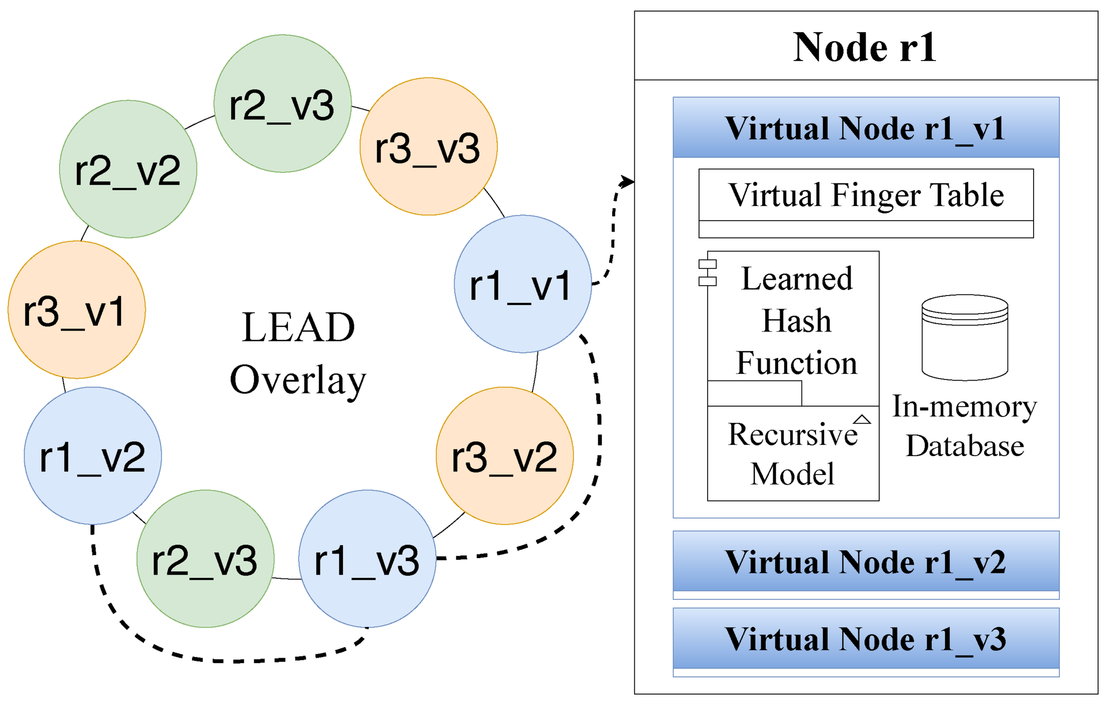
OVERVIEW

- **Distributed Hash Tables (DHTs)** are pivotal in numerous high-impact key-value applications such as CDN, DNS, IoT, DBs, and Blockchain.
- **The problem.** Traditional DHTs distribute keys randomly across the network. Hence, similar keys will be mapped to completely different storage locations. To perform a range query, all keys in the queried range need to be searched to ensure the completeness of the query. As these keys will be mapped to different locations based on the hash function, accessing these locations will result in high latency and message overhead, making range queries inefficient.

LEAD (LEArned DHT) System:

- first introduces the concept of the **Learned Hash Function** under the realm of distributed key-value database systems. We renovate traditional hash functions that map keys to random positions, allowing LEAD to maintain the inherent order of keys and enhance range query performance.
- employs mechanisms that rapidly update the overlay routing tables and maintain the learned models with a distributed model update method termed the **Federated Recursive Model (FRM)**.
- incorporates a load-balancing model using virtual nodes to allocate keys in an even manner that prevents overloading specific nodes
- designed to adapt dynamically to frequent changes in the system

LEAD DESIGN



- Physical nodes within the LEAD system are virtualized into multiple virtual nodes, each functioning as independent peers within a structured overlay network.
- LEAD utilizes consistent hashing for peer addressing.

Fig. 1. LEAD System Design

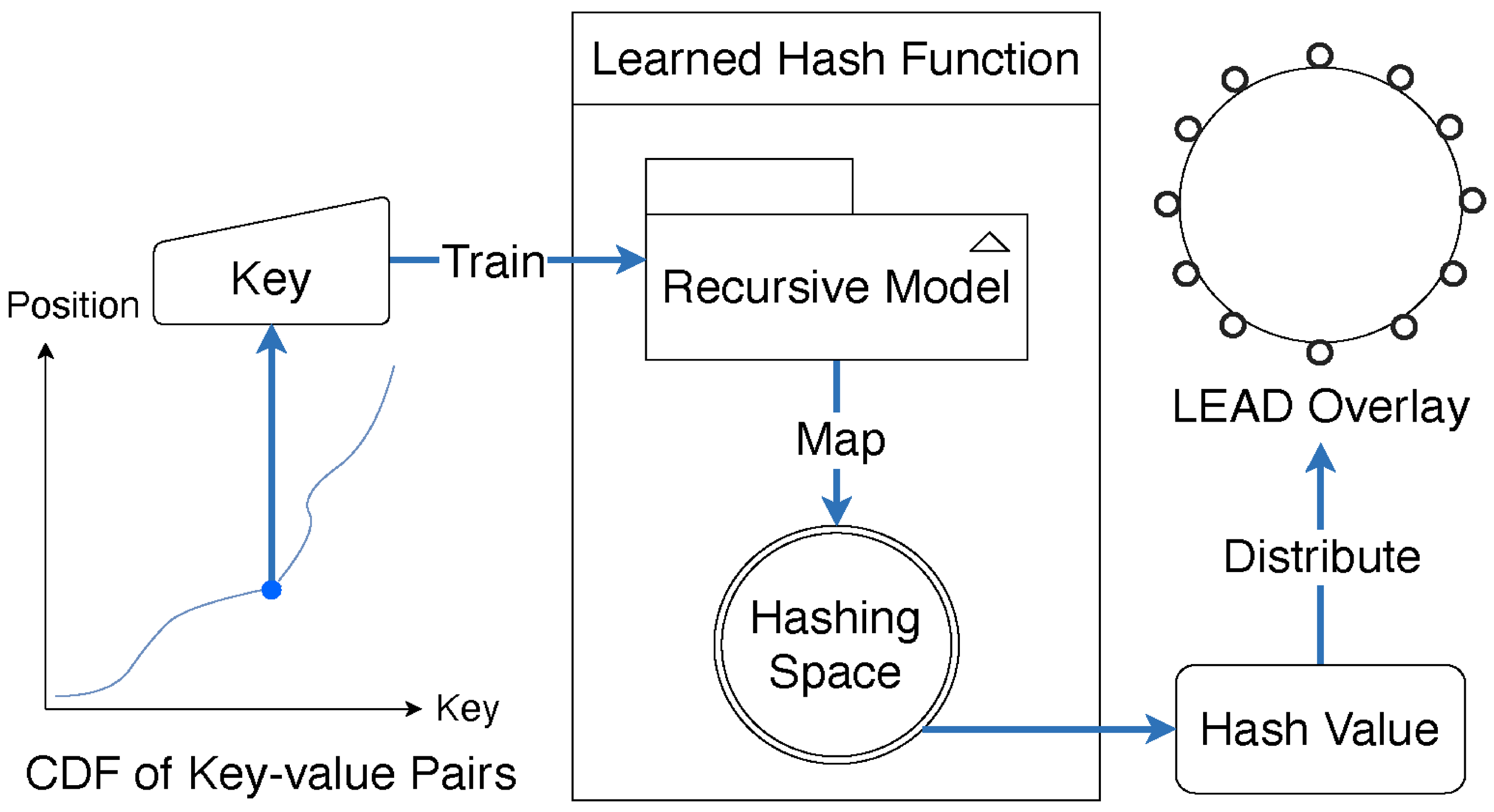


Fig. 2. Key Mapping with a Learned Hash Function

- LEAD uses the **Learned Hash Function** for key mapping. It takes a key as input and predicts its position within a hashing space as the hash value.
- Utilizing the cumulative distribution function (CDF) of keys managed on the network, the learned hash function maintains the inherent order of keys while mapping them and supports the capability for in-order data retrieval.
- LEAD employs the Recursive Model Indexes (RMI) structure to implement the learned hash function, as articulated as follows:

$$LearnedHASH(key) = \lfloor \frac{N}{H} \times f_2^{\lfloor \frac{B^a \times f_1(x)^b}{N} \rfloor}(K) \rfloor \qquad (1)$$

[a] B referred to as the branching factor that determimines the number of "buckets" that data is divided into by the stage-one model

[b] f_i referred to as the ith stage model

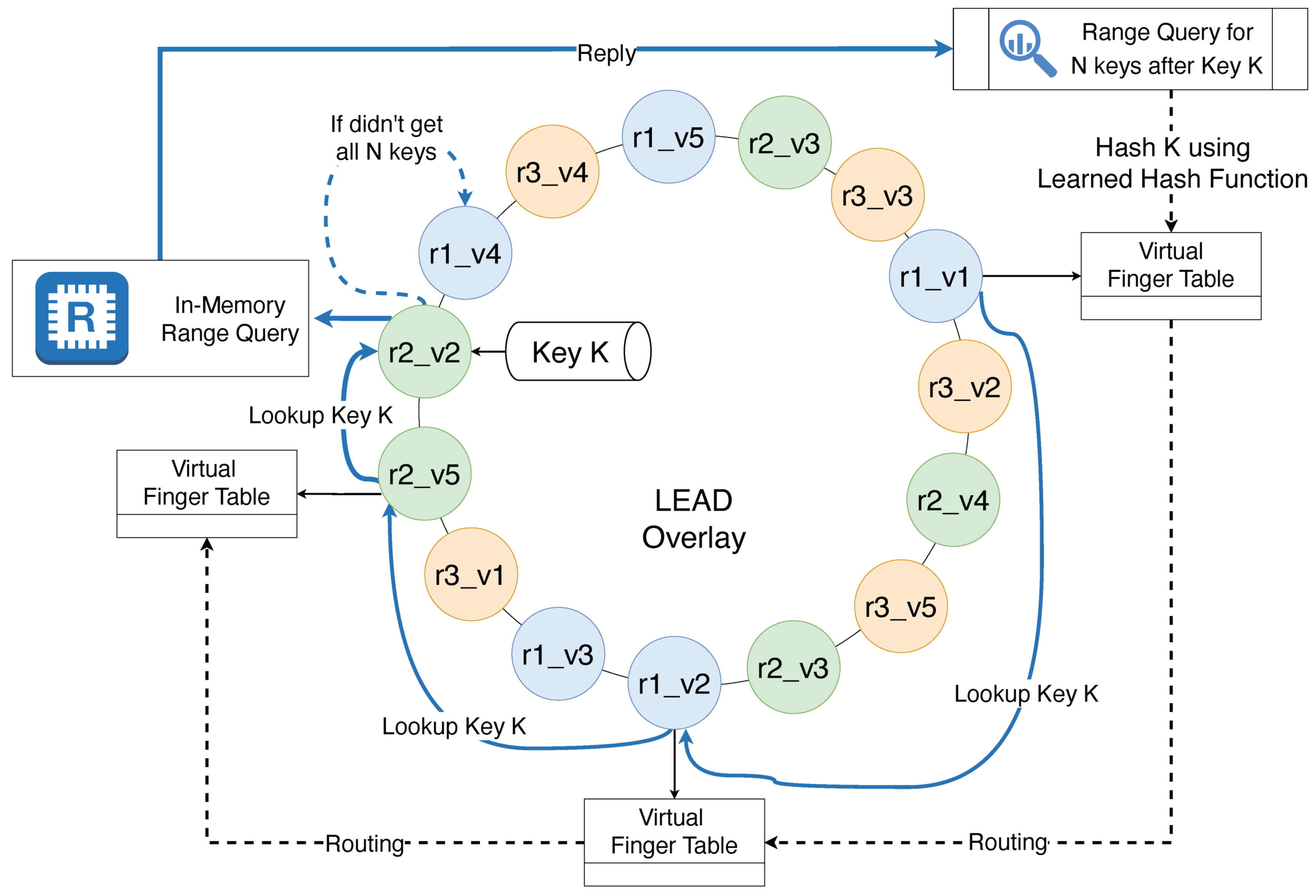


Fig. 3. Range Query in LEAD

PRELIMINARY RESULTS

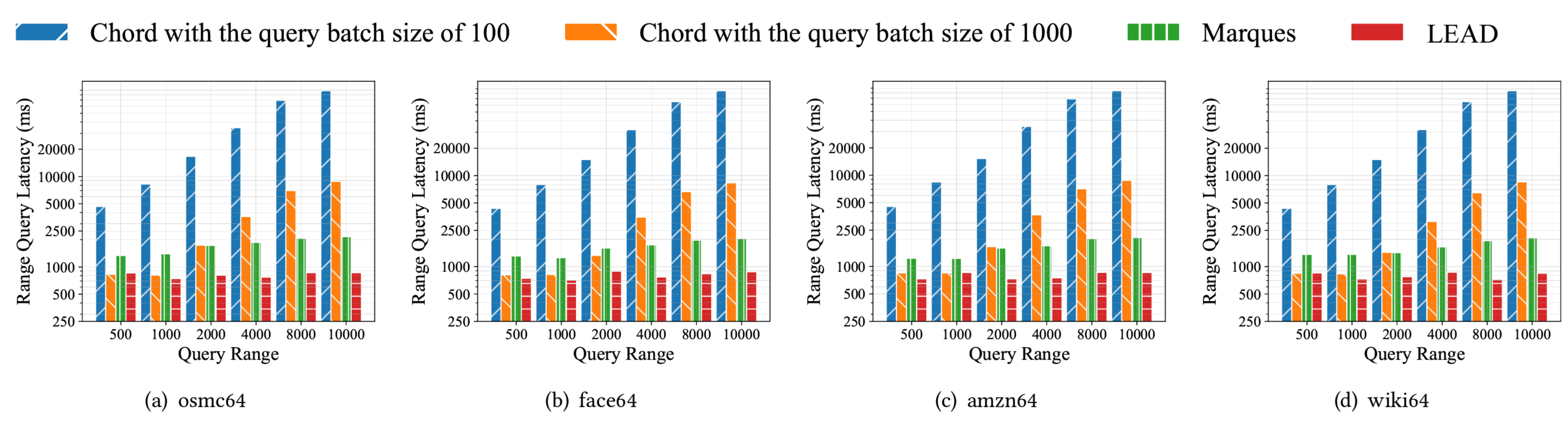


Fig. 4. Range query latency on various datasets from large-scale simulations

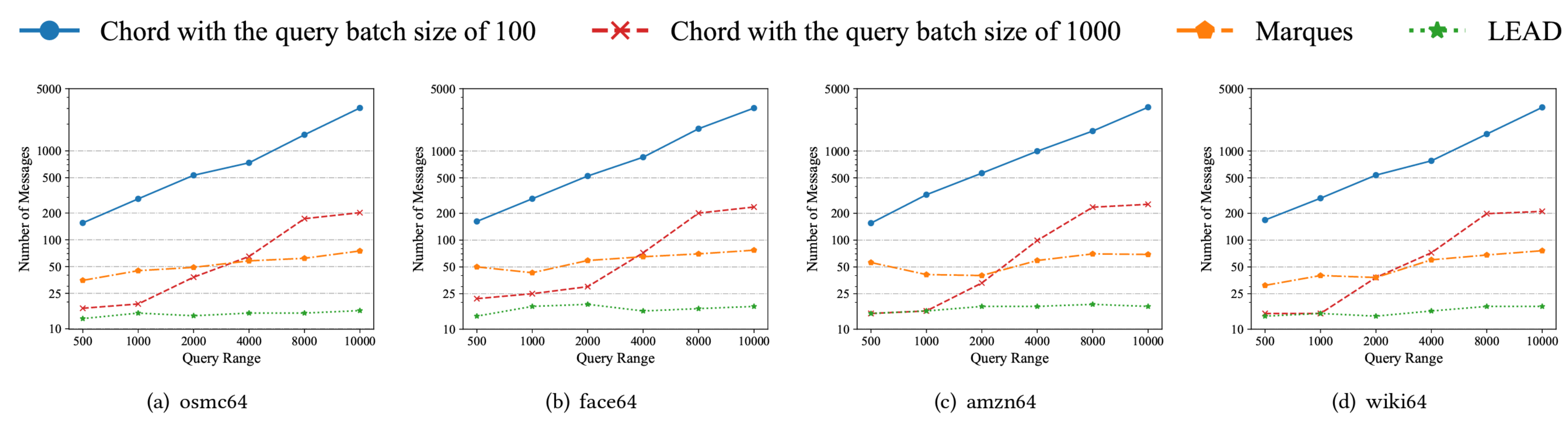


Fig. 5. Number of messages of each range query on various datasets

- LEAD achieves tremendous advantages in system efficiency compared to existing range query methods in large-scale distributed systems, **reducing query latency and message cost by 80% +**.
- LEAD exhibits remarkable scalability and robustness against system churn, providing a robust, scalable solution for efficient data retrieval in decentralized key-value systems.

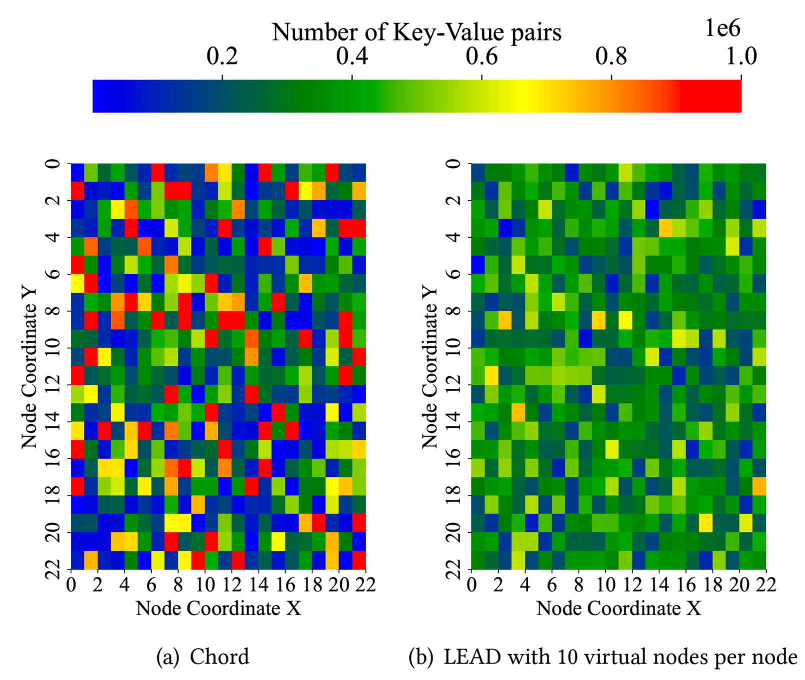


Fig. 6. Key-Value Pair Distribution

Conclusion

- LEAD includes the detailed design of training and updating learned models, implementing single-key and range queries, achieving load balancing, and dealing with system churns. The implementation and simulator code will be made open-sourced upon the acceptance of the paper.
- LEAD opens a completely new field for further research on integrating learned models with distributed networked systems. It is promising to explore multi-dimensional range queries with LEAD and to investigate its practical deployment of LEAD in the future.

References:

[1] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In Proceedings of the 2018 international conference on management of data (pp. 489-504).
[2] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM computer communication review, 31(4), 149-160.
[3] Ibrahim Sabek, Kapil Vaidya, Dominik Horn, Andreas Kipf, Michael Mitzenmacher, and Tim Kraska. 2022. Can Learned Models Replace Hash Functions? Proc. VLDB Endow. 16, 3 (November 2022)
[4] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. 2019. SOSD: A benchmark for learned indexes. arXiv preprint arXiv:1911.13014.
[5] D. S. Li, J. Cao, X. C. Lu, and K. C. C. Chan, "Efficient range query processing in peer-to-peer systems," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 1, pp. 78–91, 2009.
[6] A. Sen, A. S. M. S. Islam, and M. Y. S. Uddin, "Marques: Distributed multi-attribute range query solution using space filling curve on dths," in 2015 International Conference on Networking Systems and Security (NSysS)
[7] B. Djellabi, M. Younis, and M. Amad, "Effective peer-to-peer design for supporting range query in internet of things applications," Computer Communications, vol. 150, pp. 506–518, 2020.

Baskin Engineering UC SANTA CRUZ