Poster: Distributed Learned Hash Table

Shengze Wang[1], Yi Liu[1], Xiaoxue Zhang[2], Liting Hu[1], Chen Qian[1]
[1]*University of California Santa Cruz,* [2]*University of Nevada Reno*
{shengze, yliu634, liting, cqian12}@ucsc.edu, xiaoxuez@unr.edu

***Abstract*—Distributed Hash Tables (DHTs) are pivotal in numerous high-impact key-value applications built on distributed networked systems, offering a decentralized architecture that avoids single points of failure and improves data availability. Despite their widespread utility, DHTs face substantial challenges in handling range queries, which are crucial for applications such as storage systems, decentralized databases, content distribution networks, and blockchains. To address this limitation, we present LEAD, a novel system incorporating learned models within DHT structures to significantly optimize range query performance. LEAD utilizes a recursive machine learning model to map and retrieve data across a distributed system while preserving the inherent order of data. Preliminary results indicate LEAD achieves tremendous advantages in system efficiency compared to existing range query methods in large-scale distributed systems while maintaining high scalability and resilience to network churn.**

I. Background and Motivation

Data management across decentralized computing systems supports important large-scale Internet applications such as Content Delivery Networks (CDN), Domain Name Systems (DNS), Internet of Things (IoT), edge computing, and blockchain. Distributed Hash Tables (DHT) have been widely used for decentralized data management. DHTs are distributed data structures designed to efficiently store and retrieve key-value pairs across a network of decentralized nodes. DHTs mitigate the limitations of centralized architectures by eliminating single points of failure and distributing data loads across numerous nodes, thereby enhancing data availability and network efficiency. State-of-the-art systems like InterPlanetary File System (IPFS), Cassandra, Namecoin, and Bittorrent, have exemplified the integration of DHTs in ensuring scalable and fault-tolerant data management in decentralized networked systems.

The problem. Despite their widespread adoption and inherent advantages, DHT-based systems encounter significant challenges, particularly when handling complex queries such as range queries, which are important functions in applications such as decentralized file systems and databases, ranged data lookups in edge computing and IoT, and blockchain systems. Current DHT systems are primarily optimized for single-key lookups. DHTs use a uniformly random hash function to distribute keys into random locations, hence similar keys will be mapped to completely different storage locations. This feature of DHT will introduce two major limitations for range queries. First, all keys in the queried range need to be searched to ensure the completeness of the query. Second, these keys will be mapped to different locations based on the hash function. Visiting these locations will cause a high cost of network traffic. In the literature, efforts to improve range query performance in distributed systems have led to limited solutions. The Armada system [1] uses a partition tree model within the FissionE topology. The DBST system [2] integrates binary search trees for range queries. MARQUES [3] employs space-filling curves in a multi-level overlay structure, bringing increased overhead and scalability issues. RQIOT [4] explores the idea of using order-preserving hashing to improve range query efficiency, yet such a hash function is hard to design in a decentralized fashion. These solutions cannot completely resolve the two limitations of range queries in DHT.

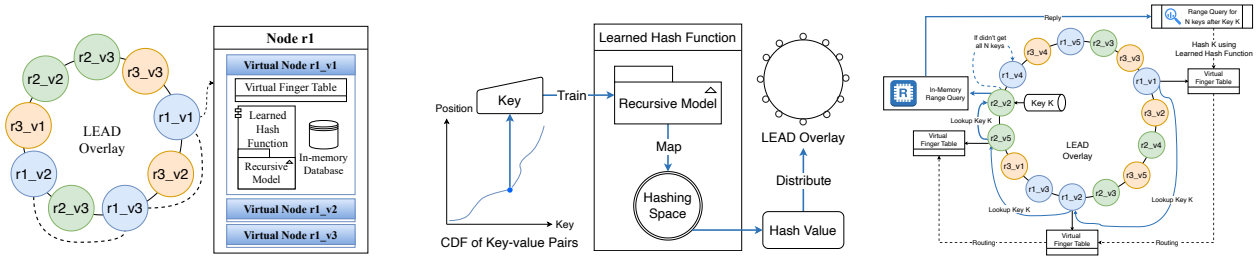
979-8-3503-5171-2/24/$31.00 ©2024 IEEE

II. LEAD Design

Our solution. To address the critical issue – enabling efficient range queries for decentralized systems – we introduce the **Distributed Learned Hash Table** with **LEAD** (LEArned DHT), a novel system that first integrates machine learning models with DHT frameworks to evidently enhance the performance of range queries evidently. Drawing on the learned indexes proposed in recent years [5], which suggests that indexes could be conceptualized as "models" that predict the position of a key within a dataset, we argue that a learned model can replace the hash function to distribute keys in decentralized networked systems. By learning the cumulative distribution function (CDF) of keys, we can **maintain the inherent order of these keys** while mapping them to a decentralized group of nodes, making similar keys to be placed in close locations. Hence the two limitations of random hash functions can be completely resolved. To minimize inference overhead and reduce the prediction error, we adapt the Recursive Model Index (RMI) structure [6] to train the learned model.

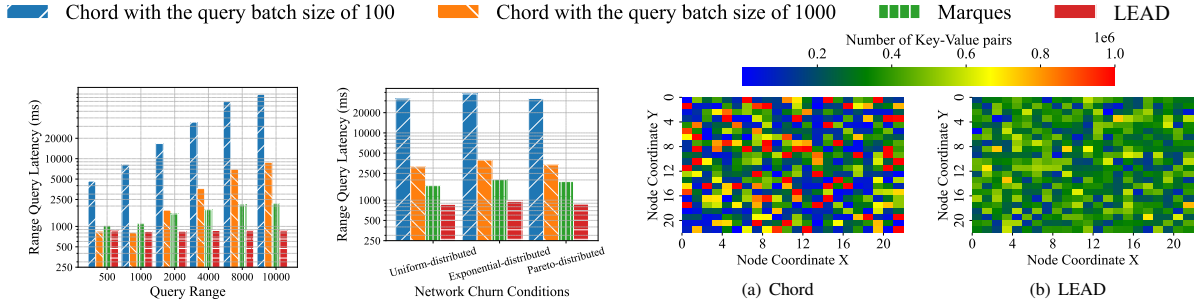
Fig. 1(a) presents the system design of LEAD. At a high level, physical nodes within the network are virtualized into multiple virtual nodes for load balancing, each functioning as independent peers within a structured overlay network. Central to each peer is the Learned Hash Function, which is utilized for efficient key mapping. This is complemented by a consistent hashing function employed specifically for peer addressing. Each peer also maintains a virtual finger table, a critical component for storing updated routing information and facilitating effective data queries. Additionally, peers are equipped with an in-memory database dedicated to the storage and rapid retrieval of key-value pairs.

LEAD introduces a novel concept within distributed networked systems: learning hash functions for key mapping, termed **Learned Hash Function**, showcased in Fig. 1(b). A



(a) System Design (b) Learned Hash Function (c) Range Query Mechanism

Fig. 1. LEAD Design



(a) Range Query Latency (b) Network Churns (c) Load Balancing

(a) Chord (b) LEAD

Fig. 2. Evaluation of LEAD on simulated testbed

learned model takes a key as input and predicts its position within a hashing space as the key's hash value. Unlike traditional hash functions, which aim to map keys to random values within a specified range, the Learned Hash Function strategically maps keys to order-preserving values in a hashing space. Utilizing the Cumulative Distribution Function (CDF) of keys managed on the network, it maintains the inherent order of these keys while mapping them to a hashing space. This preservation of key relationships enhances systems with the capability for in-order data retrieval. As shown in Fig. 1(c)), to execute a range query for a sequence of n successive key-value pairs starting from key K, the initiating peer P first applies the Learned Hash Function to hash K, yielding the hash value L_K. Using the single-key lookup mechanism adapted from Chord [7], P locates the peer S responsible for K. Once the query reaches S, S performs a local range query within its in-memory database to retrieve the sequence of key-value pairs. If S possesses all n required key-value pairs, the query is deemed resolved, and a response is sent back to P. However, if S holds only a portion of the required sequence, it forwards the remaining query to its successor. As the request moving through the network, the final peer to fulfill the range query then sends the complete set of results back to the initiating peer P.

III. Evaluation

Our simulated testbed utilizes the discrete event-driven, thread-based simulator p2psim+ [8]. We leverage four real-world datasets from the SOSD benchmark [9] and the PlanetLab Dataset from the Network Latency Datasets [10]. As shown in Fig.2, 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% to 90%+. Furthermore, LEAD exhibits remarkable scalability and robustness against system churn, providing a robust, scalable solution for efficient data retrieval in decentralized key-value systems.

Acknowledgment

The authors were partially supported by NSF Grants 2322919, 2420632, and 2426031.

References

[1] 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.

[2] S. Ahmed, A. Shome, and M. Biswas, "Dbst: A scalable peer-to-peer distributed information system supporting multi-attribute range query," in *2021 International Conference on Science & Contemporary Technologies (ICSCT)*, 2021, pp. 1–6.

[3] 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)*, 2015, pp. 1–9.

[4] 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.

[5] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 489–504.

[6] R. Marcus, E. Zhang, and T. Kraska, "Cdfshop: Exploring and optimizing learned index structures," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2020, p. 2789–2792.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM computer communication review*, vol. 31, no. 4, pp. 149–160, 2001.

[8] T. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling, "p2psim, a simulator for peer-to-peer protocols," 2003.

[9] A. Kipf, R. Marcus, A. van Renen, M. Stoian, A. Kemper, T. Kraska, and T. Neumann, "Sosd: A benchmark for learned indexes," *arXiv preprint arXiv:1911.13014*, 2019.

[10] R. Zhu, B. Liu, D. Niu, Z. Li, and H. V. Zhao, "Network latency estimation for personal devices: A matrix completion approach," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 724–737, 2017.