Research Statement

High-performance Database/Networks for AI Infrastructure

Yi Liu

University of California, Santa Cruz

Sep 24 2025

Smash: Flexible, Fast, and Resource-Efficient Placement and Lookup of Distributed Storage

Smash achieves full placement flexibility, fast storage operations, fast recovery from node dynamics, and lower DRAM cost compared to existing hash-based solutions.

Yi Liu, Shouqian Shi, Minghao Xie, Heiner Litz, and 1 more author

Architecture and Hardware

Figure 1: Our work Smash was selected by the editors to re-publish in **Communications of the ACM (CACM) 2025 as the only Research Highlight of the Oct 2025 issue.** – CACM is the most prestigious ACM journal that highlights top research work from all areas of computer science and engineering, with an impact factor 22.7. See editor's perspective on why Smash makes a distinction.

Research Overview

I am a systems researcher spanning the areas of distributed systems, computer networking, databases, and cloud/edge computing, focusing on improving the performance of distributed database systems for emerging applications such as machine learning and the Internet of Things.

In the era of massive-scale data and AI/ML computations, database engines (e.g., key-value stores, KV cache management for LLM, and vector databases) play a critical role in supporting numerous applications. Ideally, the underlying database infrastructure should be both fast to meet high user expectations and memory-efficient to minimize service costs, especially as data sizes grow and performance-hungry applications emerge in data centers. My research focuses on the design and implementation of novel data indexing schemes and algorithms aimed at optimizing distributed database performance. I believe that fast, memory-efficient database systems present unique opportunities for many data-intensive applications, ranging from cloud computing to real-time analytics, enabling them to operate more efficiently and cost-effectively in cloud services, which is crucial as the scale of data continues to grow exponentially. However, there is a growing mismatch between traditional data indexing schemes and the demands of modern data applications – comprising storage devices, network bandwidth/throughput, and data placement requirements in data centers –leading to significant challenges in efficiency and scalability. For example, in transactional systems where data locality is crucial, all data related to a single transaction could be stored within the same rack to optimize performance. However, traditional hash-based algorithms are not capable of flexibly placing data identified by different keys, as the deterministic nature of hash functions restricts the ability to place data in a way that satisfies locality requirements.

To tackle these issues, my Ph.D. research takes a deep look at exploring the potential of system performance of database systems with novel indexing schemes, and it innovates in the following themes:

- Designing a fast and communication-efficient index for key-value stores on disaggregated memory systems with RDMA RPCs.
- Achieving a fully flexible data placement and lookup in distributed storage systems via efficiently maintaining a large directory.

- Enabling scalable and low-memory table lookups for network devices with one CRC-8 hash function and learned index schemes.

My research methodology involves benchmarking to analyze bottlenecks, optimize critical components, and implement efficient systems. First, I perform comprehensive measurements to understand the performance characteristics of various database settings and break down key metrics (e.g., latency and memory costs). Second, I carefully consider how to optimize algorithms or orchestrate key modules for improving performance. Finally, I strategically implement the entire system and reassess whether the bottlenecks have been addressed. If not, I will repeat the previous steps. In the remainder of my research statement, I will detail the themes of my research mentioned above and sketch my future research plan.

Flexible Data Placement in Distributed Storage System.

Distributed storage systems, such as object stores, are widely used today to manage large-scale data in a variety of applications. In such a system, each data file consists of one or more named objects that are stored in a storage cluster. Each object is uniquely identified by a bit string, called identifier (ID), name, or key. There are two typical object placement and lookup strategies for managing objects on a massive scale. (1) Maintaining ID-location mappings in a central directory server or metadata server. Clients receive object locations by querying the server. However, the DRAM resources needed to be spent for housing the directory are significant in large-scale object storage. For instance, storing 100 billion ID-location mappings requires > 4TB DRAM, where the majority is used to store IDs. In practice, the average size of IDs is tens of bytes, such as 16 bytes. (2) Hashing-based approaches place data in storage nodes based on the hash value of the ID hID. Hashing avoids the overhead of a directory but introduces several well-known issues, such as losing the flexibility of placing objects based on application requirements. The problems include placing replicas into the same failure domain, introducing load imbalance, and forcing data re-location when nodes join or leave the system. Also, if some objects are hashed to the same storage node when they are all popular objects, the node might receive many requests and become overloaded. We revisited the above popular approaches and asked: is there any way to store each object's location in a large directory with a low memory cost?
We have identified two core services in data management: data lookup/update and data modification, including insertion and deletion. The first operation, data lookup/update, does not require altering the data's physical position, while the second involves modifying the index directory by adding or removing ID-mapping pairs. This observation led us to explore the possibility of **decoupling data lookup and modify functionalities** into distinct units – lookup units and maintenance units. The lookup unit would handle search requests using a concise data index, optimizing for memory efficiency. Meanwhile, the number of maintenance units that maintain the full directory could be smaller in distributed storage.
Developing a memory-efficient index for the lookup unit, which serves data search requests, presents a significant challenge. We observed that object IDs dominate memory consumption in large directories, as database systems commonly utilize 40-byte keys . This observation motivated us to explore Ludo hashing, a minimal perfect hashing scheme, to eliminate the need for storing keys directly in the lookup table. We applied Ludo hashing to create a memory-efficient distributed storage system that offers full flexibility in data placement in our work Smash [SIGMETRICS 23], achieving over 60% reduction in DRAM usage compared to conventional hash-based approaches.

Looking ahead, Smash offers a versatile infrastructure that can be adapted for various database systems, enabling customized data placement within data centers. This flexibility allows systems to meet specific load balancing and data locality requirements. Moreover, the principles behind Smash can be extended to metadata clusters in distributed file systems and applied to block storage on SSDs, helping mitigate wear leveling issues.

Efficient Indexing in Disaggregated Memory System.

Disaggregated memory systems achieve resource utilization efficiency and system scalability by distributing computation and memory resources into distinct pools of nodes. Remote Direct Memory Access (RDMA) is an attractive solution to support high-throughput communication between different disaggregated resource pools [HotStorage 25]. However, existing RDMA-based KV stores face a dilemma: one-sided RDMA completely bypasses computation at disaggregated memory nodes, but its communication takes multiple round trips; two-sided RDMA achieves one-round-trip communication but requires non-trivial computation for index lookups at memory nodes. This motivates us to think about whether we can improve the throughput by lowering the computation on the memory node side. We measured remote data search throughput using RDMA RPC-Dummy, where the memory node performed no computation and simply returned a dummy result. This approach allowed us to determine the maximum potential throughput when all computation is removed from the memory node. The observation that 20-40% of the achievable throughput was still available motivated us to explore ways to reduce computation time on the memory node when processing data requests.

Decoupling of Compute-intensive and Memory-intensive Part of Index. We designed and implemented an RDMA RPC-based system in our work Outback [VLDB 25], which decouples the dynamic minimal perfect hashing (DMPH) table into two components. (1) The compute-intensive component is memory-efficient and includes most computation operations of the index, which is placed onto the compute nodes. It contains all indexes that are used to compute the slot where the searched key is located. (2) The memory-intensive component contributes to the most memory cost of the index, but its computation is trivial, and it is on the memory nodes. It maintains the location information for all keys. Such a design principle of decoupling the index is ideal for disaggregated memory systems: all computation tasks on locating a searched key and the majority of computation for data insert requests are offloaded on compute nodes, while memory nodes focus on providing service for memory read and write without taking other computation tasks. Hence, this approach is particularly effective for real-world workloads dominated by data search requests.

Extending Outback to extended RDMA verbs and DPUs. We can apply Outback to another two promising approaches without using two-sided RDMA verbs. (1) Extended RDMA READ verb. We can leverage an extended one-sided RDMA indirect reading verb `RDMA_READ` (`ptr` *addr*, `size` *len*, `bool` *indirect*), where *indirect* indicates if RNIC is supposed to read back the data pointed by the *addr*. This embedded one-sided RDMA verb can free the memory node's CPU and offload the memory reading task in Outback to RNICs. The reason is that Outback can get the exact requested data address without potential data probing. (2) Performance capacity of Outback with hardware accelerators. The concept of Outback can reduce the computation burden on SmartNICs by employing one round-trip, one-sided RDMA_READ, too. For example, a SmartNIC can be placed on the memory node side, and function as an additional computation unit, and indirect data access tasks can be offloaded to it. After the compute nodes in Outback issue a one-sided RDMA to read the queried key's slot and retrieve the address from the DMPH buckets, the SmartNIC can read the memory again via the PCIe switch and obtain the queried data through an additional PCIe round trip. The computation and data search tasks offloaded to the SmartNIC can be alleviated with the assistance of DMPH for the least computation burden.

Fast Table Lookup on Network Devices.

The Forwarding Information Base (FIB) on network switches should be fast to support high network throughput and line rate. However, existing designs for network lookup functions rely on multiple independent universal hash functions, which introduces several inevitable problems, including high computation time and lack of sufficient independent hash functions on certain hardware devices. For example, Broadcom switches support RTAG7, and the Cisco Nexus 5500 Series supports CRC-8, which does not satisfy the hash independence and uniformity requirements. On the other hand, the learned model hashing (LMH) was recently proposed to use a machine learning model to replace traditional hash functions for secondary indices. The idea of LMH is to train a model that approximates the cumulative distribution function (CDF) of all keys and predicts the position of a lookup key in a sorted array. Each position of the array represents a bucket that can store the value corresponding to a lookup key. The array can then be considered a hash table where LMH replaces hash functions to calculate the position of a key.

Therefore, we planned to replace the hashing function used in network switches with LMH. However, LMH has a high memory cost (over 1.2GB for 50 Million MAC addresses), and memory resource in network devices is typically limited. We first measured the memory usage of LMH with different numbers of keys and analyzed its memory bottleneck. We found that the uneven distribution of keys required more memory for stash nodes chained in each bucket, and the time spent traversing nodes during lookups accounted for most of the latency. On the other hand, we observed that MPH allows fast key localization within each bucket while using minimal memory, but the bucket locator's process of distributing keys into buckets is the main source of latency. To address this, we propose Parrot Hashing [ICNP 24], which combines LMH and MPH to create a fast and memory-efficient solution, with LMH serving as the bucket locator, and MPH working as the slot locator to avoid traversing buckets. Furthermore, applying LMH computation on a programmable switch proposes the challenge of not supporting the multiplying of integers in the Tofino P4 pipeline. We leverage the range match table to map the keys into the bucket based on the results of LMH to realize the line-rate table lookup with a single CRC-8 function in the network switch.

Other System Research Works during My Ph.D. Study:

Improving data read throughput with an IO Aware Caching for LSM-tree-based KV store. LSM tree-based KV store is a popular database design in modern persistent storage systems due to its efficient writing with sorted keys. However, this hierarchical log structure suffers from extensive read amplification because multiple disk accesses are required when it searches for a key. Recent optimizations of LSM trees propose caching hotkeys to reduce I/Os mainly based on their access frequencies. However, our measurement shows that keys accessing differ in I/O costs, which should also be considered in the caching policy: caching key-value pairs with high I/O costs can effectively improve query latency. We designed a weighted Count-Min sketch in our SpotKV [CLOUD 24], which gives higher priority to keys lying in the higher levels, which requires more SSTable accessing. We implemented this cache admission policy and dynamic cuckoo filter to improve the data lookup throughput.
Fast user-associated contents from edge servers. We explore the opportunity of efficiently indexing user-associated contents and propose a scalable content-sharing mechanism for edge servers in our work EdgeCut [SEC 23]. We designed a compact and dynamic data structure called Ludo Locator that returns the IP address of the edge server that stores the requested user-associated content at a low memory cost.

Future Research

The long-term goal of my research is to develop novel systems to enable fast and memory-efficient distributed database systems, drawing on recent advances in interconnection technologies (e.g., CXL/RDMA, disaggregated data center and in-network computing) and AI databases such as vector database and Retrieval-augmented generation. Specifically, I intend to leverage my knowledge in three main directions: (1) Towards efficient disaggregated systems. (2) High-performance distributed AI databases. (3) AI infrastructure on networking.
Towards efficient disaggregated systems. Disaggregation enables more flexible and efficient use of resources by decoupling memory and computation, which is crucial for modern data centers aiming to optimize resource utilization and reduce costs in industry. My research seeks to leverage recent interconnection technologies like RDMA/CXL to extend the concept of disaggregation to systems like key-value stores (e.g., LSM trees) and transactional systems. I will co-design the system from the view of network fabric and application features to make the systems easily deployable and disaggregated in future datacenters. For example, by integrating and optimizing lightweight network stacks such as DPDK and eBPF, we can create a high-performance, scalable architecture that enables more efficient communication and coordination between disaggregated components.
High performance distributed vector databases. Vector databases have emerged as a powerful tool for managing unstructured data, particularly in the realm of AI applications like large language models (LLMs). They enable efficient storage, indexing, and retrieval of high-dimensional vectors, significantly enhancing the speed and accuracy of AI-driven search and inference tasks. However, traditional VectorDBs have largely focused on search algorithm optimizations and have not fully explored the potential of distributed deployments within modern data centers from a view of a system. My future research aims to develop high-performance, distributed vector databases with focusing on efficient data partitioning, network communication, and recall performance.
AI infrastructure on networking. In my future research on networking optimization for AI infrastructure, I aim to explore new methods to improve communication efficiency in large-scale AI systems. For example, collective communication primitives using NVSHMEM and IBGDA can be leveraged to optimize token dispatch and aggregation in models like DeepSeek-R2, which distributes tokens to experts and combines results across GPUs via NVLink and RDMA. Orchestrating network and computation efficiently is a key challenge to fully exploiting these high-performance interconnects. In addition, tensor parallelism is commonly used to distribute computational workloads across multiple machines, and efficiently executing communication primitives such as all-gather and all-to-all, both within and across nodes, is critical for minimizing communication overhead in GPU clusters. Different GPUs have varying interconnect topologies and bandwidths, and understanding these differences is essential for optimizing model-specific performance. The ultimate goal of this research is to develop networking strategies that maximize throughput, minimize latency, and enable scalable execution of large language models across heterogeneous GPU clusters.

Teaching Statement

Yi Liu
University of California, Santa Cruz

I find great joy and pride in the opportunity to nurture future computer scientists and engineers, both in the classroom and through research. Throughout my PhD, I thoroughly enjoyed teaching and mentoring students, and those experiences shaped my teaching and mentoring philosophy.

Teaching Interests and Experience

Given my research experience and background, I am eligible to teach many CS undergraduate courses such as **Operating Systems, Distributed Systems, Computer Networking, Databases, Algorithms and Data Structures, Security and Privacy, System and Network Programming**, as well as lower-division CS courses such as **C/Python Programming, Discrete Math, and Probability & Statistics**. I would also enjoy giving seminars on emerging topics in computer networking and database systems for graduate courses, such as **Distributed Systems, Cloud and Edge Computing, Advanced Computer Networking, and the Internet of Things.** My teaching style was formed by serving as a teaching assistant (TA) at UC Santa Cruz. I have TA-ed for five undergraduate courses in six quarters: CSE 150 Computer Networks (2023 Spring), CSE 80N Intro to Networking (2022 Fall), CSE 107 Probability and Statistics (2022 Winter), CSE 16 Applied Discrete Mathematics (2021 Spring & Fall), and CSE 156 Network Programming (2023 Winter).

Teaching Philosophy

My teaching experiences have shaped my teaching philosophy, and I list the following teaching philosophies that I think are important for being a good professor.

1. **A good teacher has good interactions with students in class.** A good teacher fosters effective interactions with students in the classroom. Engaging with students actively helps create an environment where learning is both dynamic and collaborative. In my teaching approach, I prioritize building strong connections with my students by encouraging open communication and providing timely feedback. I make it a point to understand each student's learning style and adapt my teaching methods accordingly. During lectures and discussions, I invite questions and encourage students to share their perspectives. This not only helps clarify complex concepts but also promotes a more inclusive and participatory classroom atmosphere. In addition, I regularly incorporate formative assessments and interactive activities to make sure student understands the course materials and adjust my instruction as needed. By being approachable and responsive to students' needs, I aim to support their academic growth and foster a positive learning experience in class. Effective interactions between teachers and students are crucial for creating an engaging and supportive educational environment, and I strive to uphold these principles in all my teaching endeavors.

2. **A good teacher combines concepts and hands-on practices.** I believe in the importance of hands-on practice in computer science education, and I will let students work with real-world technologies and tangible software or hardware through various programming projects. For instance, the longest prefix match is a rule used by network switches to forward data packets to the appropriate ports. However, students in my class sometimes struggle with this concept, especially when multiple hosts share similar IP addresses. To address this, we conduct lab sessions where students build a simple network topology using a software simulator. They then observe how the network switch handles "ping" commands from different hosts. This experience helps them understand how data packets are forwarded across the network. In the future, I plan to first explain the design principles and then provide opportunities for students to apply what they've learned through practical exercises. My aim is to give students both a solid theoretical understanding and

practical skills that they can use in real-world situations. By working on projects that reflect industry challenges, students not only strengthen their grasp of fundamental concepts but also enhance their critical thinking and problem-solving skills.

3. **A good teacher knows how to progressively guide students.** A good teacher carefully plans and executes instruction to guide students through a logical progression of learning. I begin by introducing core concepts and then gradually build on them, ensuring that students master each level before moving on to more advanced material. This step-by-step approach helps students develop a solid foundation and understand complex ideas more easily. For example, in the computer networks course, I will focus on explaining the network protocol stack layer by layer and clarifying the data exchanged between adjacent layers. I will also employ various teaching techniques to support this gradual learning process, including breaking down topics into smaller segments and using real-world examples to illustrate abstract concepts. Regular evaluations and feedback allow me to adjust my approach and provide additional support where needed. By carefully managing the pace and complexity of the material, I believe I can help students gain confidence and achieve a deeper understanding of the subject.

Mentoring

During my Ph.D. studies at UC Santa Cruz, I had the privilege of assisting to mentor two junior Ph.D. students in our lab. Additionally, I have been fortunate to receive guidance from my exceptional advisors at both UC Santa Cruz and USTC. Through these experiences, I believe that a good advisor should focus on the following key goals.

1. **A good advisor supports the student's interests in research.** A good advisor supports the student's research interests by encouraging them to pursue topics they are passionate about. This approach not only motivates the student but also fosters creativity and deep engagement with the subject. For example, one of the junior Ph.D. students I helped to mentor in our lab was particularly interested in exploring a novel approach to optimize distributed hash table performance with machine learning models because he had a strong AI background before his PhD career. Although it was outside the original scope of our project, I encouraged them to explore this direction, which ultimately led to promising preliminary results and getting published as an ongoing work poster in *IEEE ICNP*.

2. **A good advisor fosters the student's innovative thinking through research guidance.** In my role as a mentor to two junior Ph.D. students, I have embraced this responsibility, providing not just technical insights but also the support needed to develop their independent research capabilities. My goal is to create an environment where students feel confident experimenting with their own ideas, taking ownership of their projects, and learning from failures as much as successes. For instance, when working on projects like designing more efficient data structures or solving computational problems, I guide my students to not only solve the immediate issue but also consider its broader implications, pushing the boundaries of their current knowledge. As a future advisor, I will focus on helping students build confidence in their own problem-solving abilities with critical thinking.

3. **A good advisor helps with the student's expression skills.** A good advisor not only guides students in research but also helps them develop strong **writing and presentation skills**. For example, writing is an essential aspect of academic success, and I believe it's crucial to start nurturing this skill early in a student's career. As a mentor, I encourage my students to approach writing and presenting as a process, one that involves clear articulation of ideas, structured argumentation, and attention to detail. Drawing from my own experiences writing academic papers, I know how challenging it can be to present complex research ideas clearly and concisely. I make it a point to provide detailed feedback on my students' drafts, not only on technical content but also on clarity, coherence, and style. I often share my own writing techniques with my labmates, such as breaking down large sections into manageable parts, revising for conciseness, and eliminating grammatical errors. Given my own preference for writing free of fancy terms or excessive embellishments, I help my students focus on precision and clarity in their writing and presenting, aiming for work that is both academically rigorous and easy to follow. Finally, I will aim to prepare my students not only for successful publication but also for effective communication of their ideas in a variety of academic and professional contexts.

References for Yi Liu

Dr. Chen Qian
Professor
Graduate Director of CSE
University of California Santa Cruz
Phone: 8314595302
Email: send.Qian.AFBFE09864@interfoliodossier.com

Dr. Heiner Litz
Kumar Malavalli Associate Professor
Director of the Center for Research in Storage Systems
University of California Santa Cruz
Phone: 6503530771
Email: send.Litz.4806618638@interfoliodossier.com

Dr. Yuanchao Xu
Assistant Professor
University of California Santa Cruz
Phone: 9198840400
Email: send.Xu.A6B6650E50@interfoliodossier.com

Dr. Minmei Wang
Assistant Professor
University of Connecticut
Phone: 8604865285
Email: minmei.wang@uconn.edu