ACM SIGMOD Programming Contest 2025

Team Kirara (Runner-up): Liming Xiang[1], Jing Feng[1], Yibo Shao[1], Yongze Yu[2], Jiaxi Hou[1]
Advisor: Hongchao Qin[1] Contact: hcqin@bit.edu.cn

[1] Beijing Institute of Technology, China; [2] Xidian University, China;

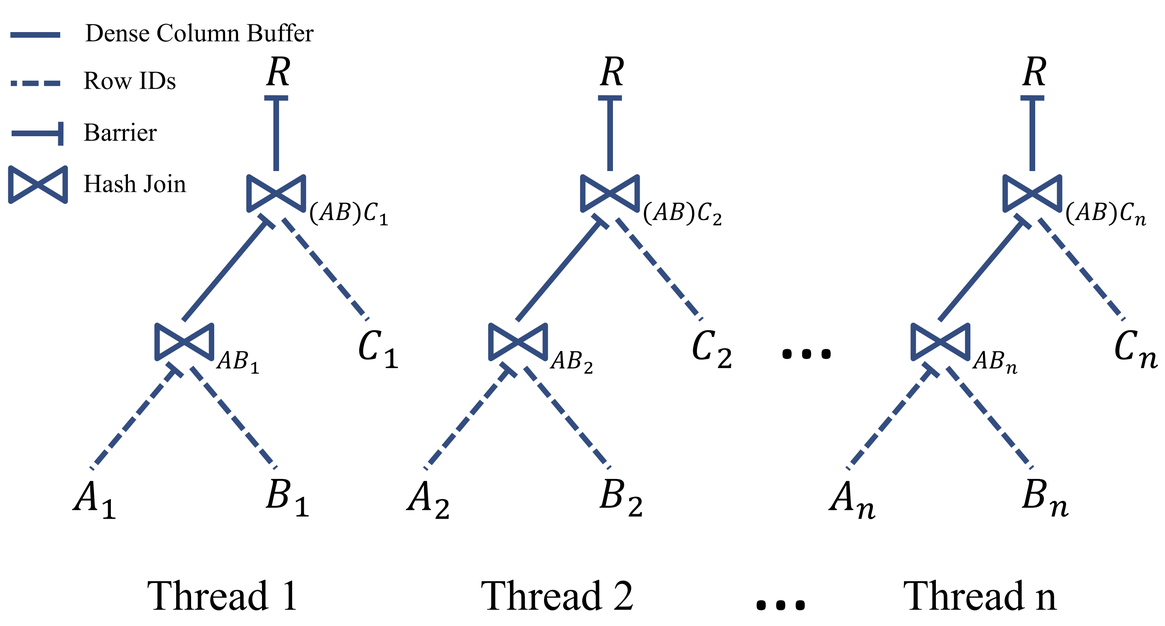
Contest Overview

- ***Task:*** Implement a high-performance, in-memory join pipeline executor across diverse hardware architectures.
- ***Dataset*:** 983 queries derived from from the Join Order Benchmark (JOB) over the IMDB dataset.
- ***Hardware:*** Evaluated on 8 diverse server architectures, including AMD, ARM, IBM, and Intel.

System Architecture

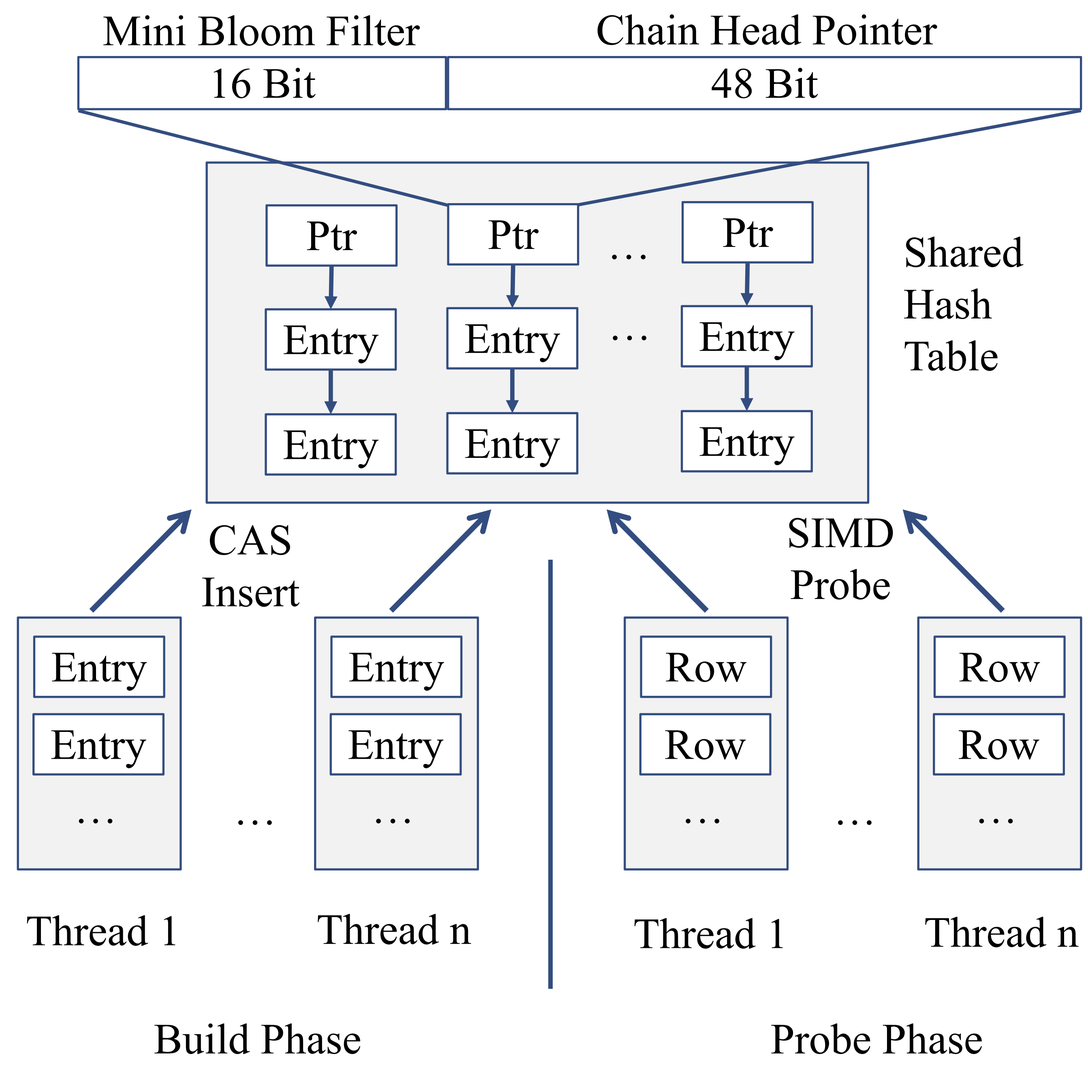
Our engine adopts a ***parallel, pull-based, vectorized execution model.***

- ***Vectorized Volcano Model:*** Operators pull columnar batches from their children, creating a demand-driven pipeline.
- ***Parallel Execution:*** A thread pool executes multiple instances of the query plan. A shared state manager coordinates atomic task claiming for operators.
- ***Delayed Materialization:*** Scan operator pass the references to original data instead of copying tuples. Materialization is deferred until required by operators like Join.



Core Optimization Techniques

- ***Concurrent Hash Join:***
  1. ***Parallel Build & Probe:*** All threads concurrently build and probe a shared hash table on partitioned data.
  2. ***Lock-Free Build:*** Entries are prepared thread-locally, then inserted via atomic Compare-and-Swap (CAS).
  3. ***Pointer-as-Bloom-Filter:*** The first 16 bits of a hash chain's head pointer serve as a mini bloom filter to accelerate key existence checks.



- ***Subtree Caching:*** Avoids re-computation by caching sub-plan results. Identical subtrees are detected via a hash fingerprint of their structure and data samples.
- ***High-Performance Memory Pool:*** A global pool serves thread-local arenas for contention-free allocation. Memory is allocated via a simple pointer bump.
- ***Adaptive Join Strategy:*** For trivial cases like a single-row build side, the engine bypasses the hash join in favor of a lightweight Nested-Loop Join.
- ***SIMD Intrinsics:*** Enables high-performance vectorization through Clang's Extended Vectors, which automatically generate optimal, platform-specific SIMD code.
- ***Dynamic Parallelism Scaling:*** The optimal degree of parallelism is decided heuristically based on the input data size, balancing throughput against coordination overhead.

Results

- Achieved ***2nd place*** with a geometric mean runtime of 7.03s across 983 queries on 8 diverse hardware platforms.
- Performance profiling revealed that the primary bottleneck is ***cache misses*** during the probe phase of our hash join, particularly for joins with large build-sides.

Benchmark Results (s)

AMD		ARM		IBM		Intel		Geo. Mean
EPYC 7F72	EPYC 7343	Ampere Altra Max	NVIDIA GH200	Power8	Power10	Xeon E7-4880	Xeon Platinum	
6.86	8.79	6.95	2.62	10.42	7.33	14.5	4.88	**7.03**

References:
Leis, Viktor, et al. "Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age." 2014.
Kersten, Timo, et al. "Everything you always wanted to know about compiled and vectorized queries but were afraid to ask." 2018.
Bandle, Maximilian, Jana Giceva, and Thomas Neumann. "To partition, or not to partition, that is the join question in a real system." 2021.