A Lock-Free Cache-Aware Pipeline for High Throughput Joins

SIGMOD 2025 Programming Contest

Yannis Xiros, Zisis Vakras, Alexandros Kostas - Team O.P.T.

Department of Informatics & Telecommunications, University of Athens, Greece

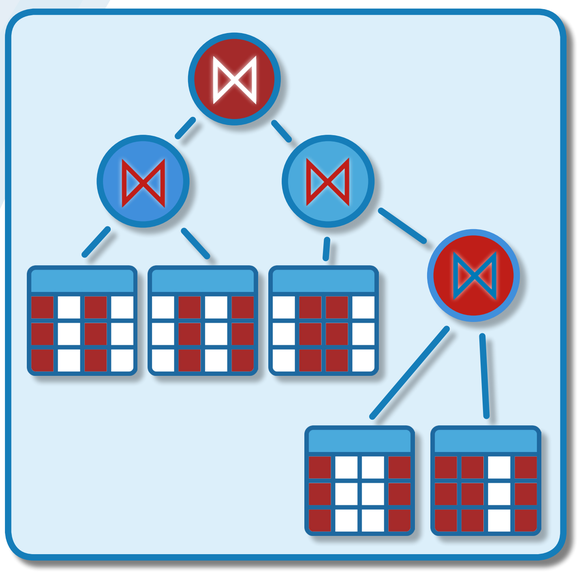
MaDgIK Lab: www.madgik.di.uoa.gr

Summary

- **A cache-efficient, multi-threaded, lock-free join pipeline implementation utilizing a memory-efficient hash table optimized for joins.**
- **630x speedup over the reference solution!**

Task Overview

- Develop an efficient, in-memory join pipeline executor delivering high performance across a wide range of hardware architectures and configurations.
- Evaluated on anonymized data and queries sampled from the J.O.B. (IMDB Dataset).
- Input Data provided **in memory** as column-store, paginated tables.
- Optimized Join Plan provided by PostgreSQL.
- Joins only on Integers, with Doubles and Strings also appearing in the results.

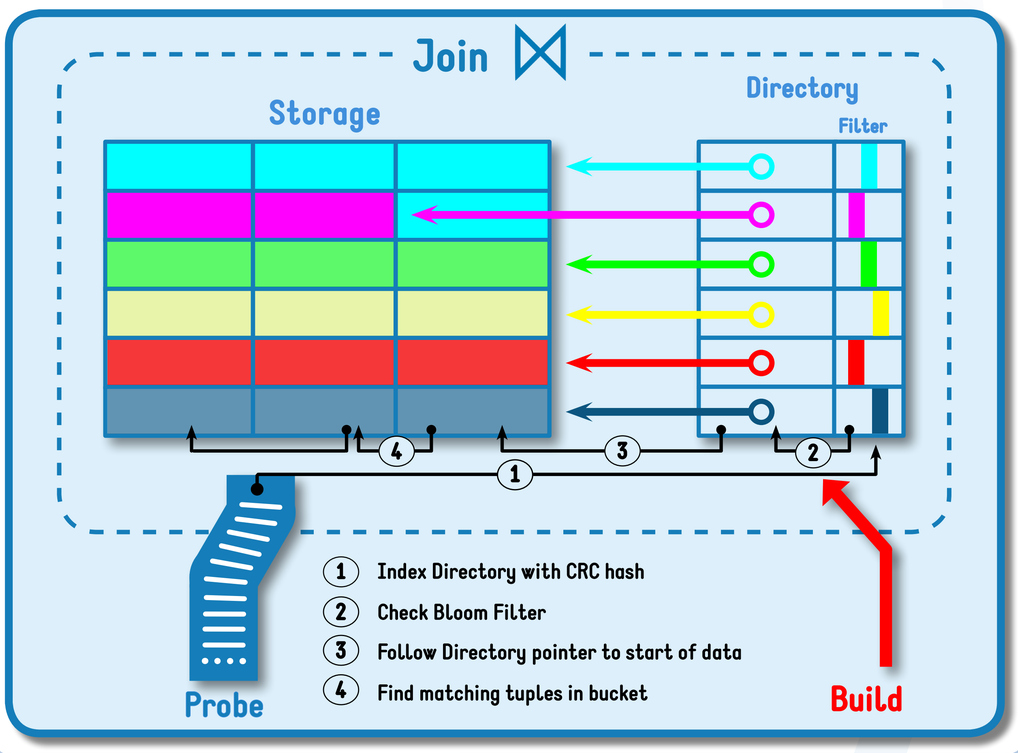


1. Loading and Preprocessing

- Classify columns as join fields, output-only attributes or unused.
- Vast majority of pagenated columns have no null values and can be accessed in place in constant time with a single division.
- Serial access can eliminate the division from the hot path.
- Copy remaining useful columns in contiguous arrays to enable random access.
- **Lazily** copy Strings only when needed in the result.
- **Lazy-load** output-only columns in the background to start join execution early.
- 99+% of columns are never loaded or get processed in the background, approximating a **zero-copy** solution!

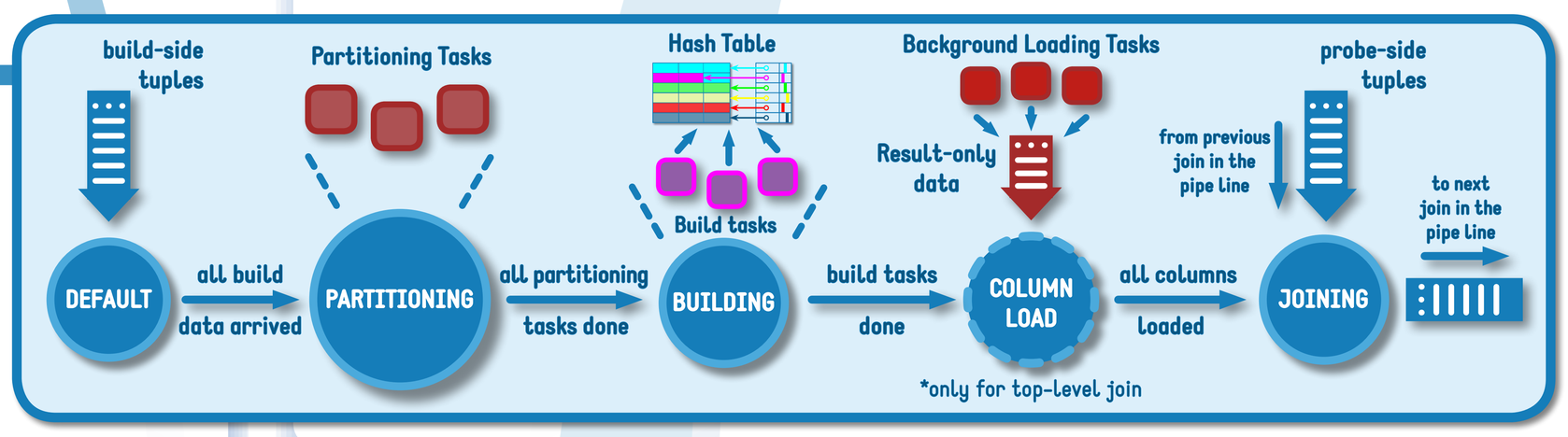
2. Unchained Hash Table

- Hash Table optimized for joins[1].
- Distinct phases for building and probing.
- Partition data among workers to parallelize building.
- All the entries of the table lie in a single large **contiguous** block, massively improving cache locality.
- Use of *CRC* hardware instruction for quick hashing with adequate collision resistance.
- **Bloom Filter** in the lower 16-bits of the directory pointer.



3. Join Data Flow & Materialization

- **Late Materialization** to eliminate unnecessary data copies and reduce cache pollution.
- Each join materializes the column needed for the next join in the pipeline to improve memory access patterns.
- Joins at the root of the tree materialize the final output.



4. Parallel Data Processing

- Priority-Based Task Queue that dynamically assigns tasks on available hardware threads.
- Prioritize tasks that enable processing of interdependent joins.
- *Morsel-driven Parallelism*[2] : break down heavier jobs into **self-contained tasks**, each responsible for independently processing a smaller chunk of the data.
- **Lock-free** chunk feeder mechanism to efficiently handle *skew*: tasks that finish early can dynamically "**steal**" pages from siblings to balance workload.
- Each task **independently** produces paginated results and deposits them in dedicated **per-worker slots** without any synchronization cost.
- Dynamic Job Batching for very small tasks to reduce overhead.
- Properly align concurrently accessed data structures to eliminate false sharing.

5. Pipeline

- Chunk and immediately forward intermediate results so the next join can also progress while the data are still hot in the cache.
- Schedule further processing of the data to the same hardware thread that produced it to optimize cache efficiency.
- **Lock-free** pipeline implemented with atomic instructions to minimize synchronization overhead and enable finer granularity tasks.
- Every completed task executes part of the pipeline control logic - appropriately advancing pipeline state and kickstarting tasks for the next stages or joins.

Acknowledgements & References

We would like to sincerely thank Stefanos Stamatis for his constant guidance, feedback and support during the entire Contest. We are also grateful to Yannis Foufoulas, Theofilos Mailis and Professor Yannis Ioannidis of the MaDgIK group for their valuable insights and discussions.

[1] Birler, Altan, et al. "Simple, Efficient, and Robust Hash Tables for Join Processing." *Proceedings of the 20th international workshop on data management on new hardware*. 2024.

[2] Leis, Viktor, et al. "Morsel-Driven Parallelism: a NUMA-Aware Query Evaluation Framework for the Many-Core Age." *Proceedings of the 2014 ACM SIGMOD international conference on Management of data.* 2014.

Want more?

HELLENIC REPUBLIC
National and Kapodistrian University of Athens

MADIK
Management of Data Information & Knowledge Group

Team O.P.T.