CuART – A scalable Radix Tree Lookup Engine

Martin Koppehel, Thilo Pionteck
Otto-von-Guericke Universität Magdeburg

Tobias Groth, Sven Groppe
Universität zu Lübeck

Motivation

- Ever increasing database sizes and query requirements
  - provide deeper insights into larger datasets
- New database organization types
- Increase the need to quickly locate (sets of) specific entries
  - Database indexing needed
- Index Performance is one of the key factors for whole DBMS performance
  - Consulted for **almost every** query, **several** times
- Mapping from key to a memory location
- Different structures used today
  - Search Trees, Hash Tables, Prefix Trees, (Learned Indexes)
  - Used for different purposes

Background

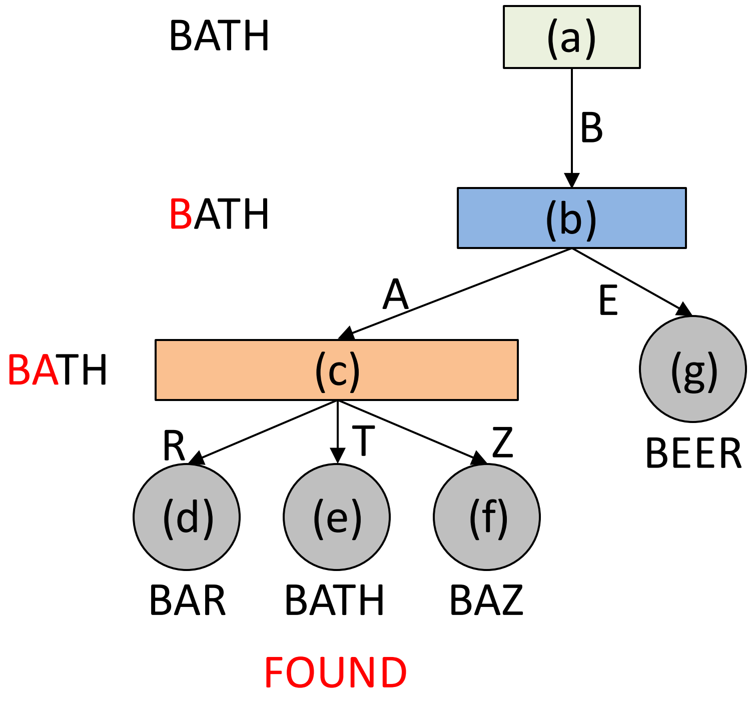
- Prefix Trees not widely used in production today
- Prefix Trees are comparably small
- Prefix Trees can serve a broad range of query types

- Very fast on CPUs for small index sizes (Caching Effects)
- Feasible for massively parallel implementation

- ART is a prefix tree with adaptive node sizes
  - Four different node sizes

- GPU implementation of ART available (GRT)
- GRT does not achieve an optimal utilization of modern GPUs
  - **Optimization time!**

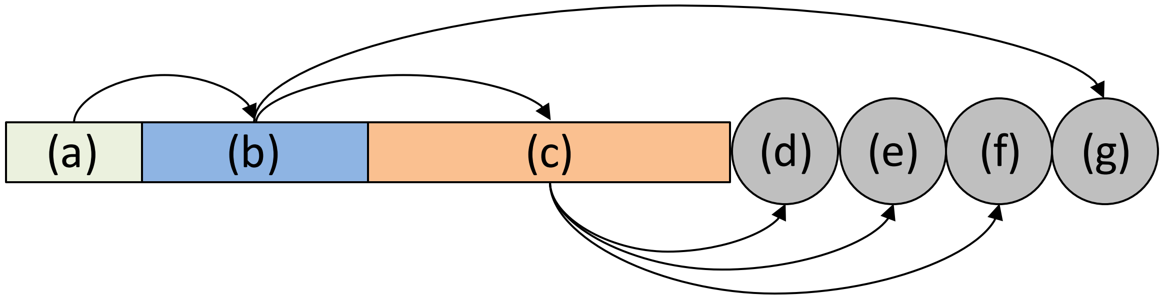
Query for BATH



GRT Problems

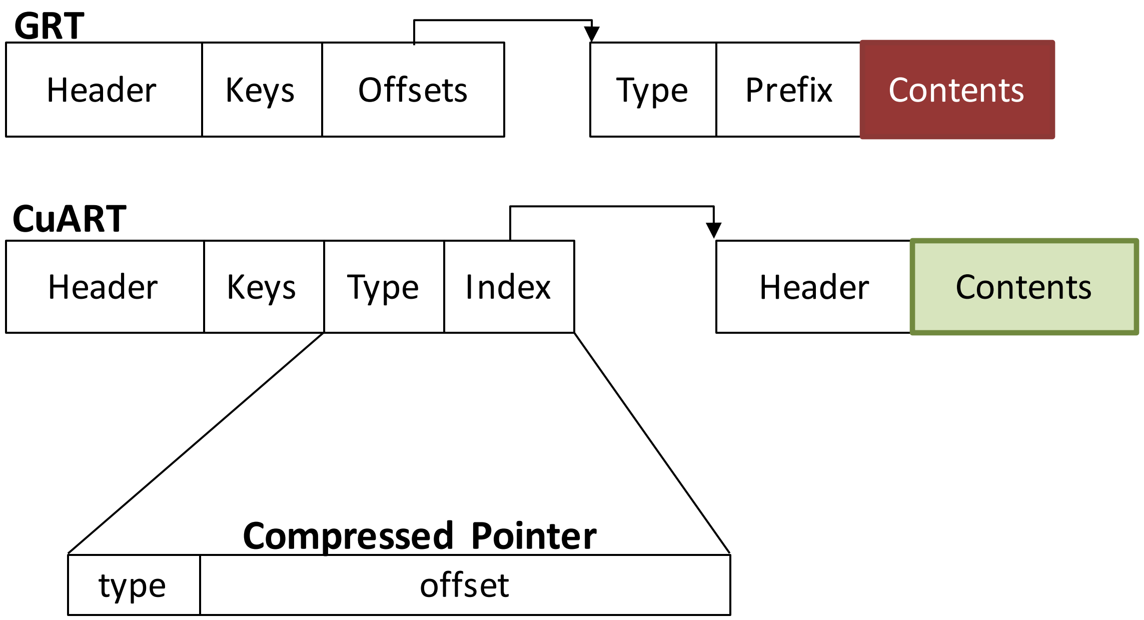
- GRT maps the nodes into a single flat buffer
- Nodes are represented by aligned union types
  - Have to read the node type first
  - Have to issue another memory transaction for the remaining node

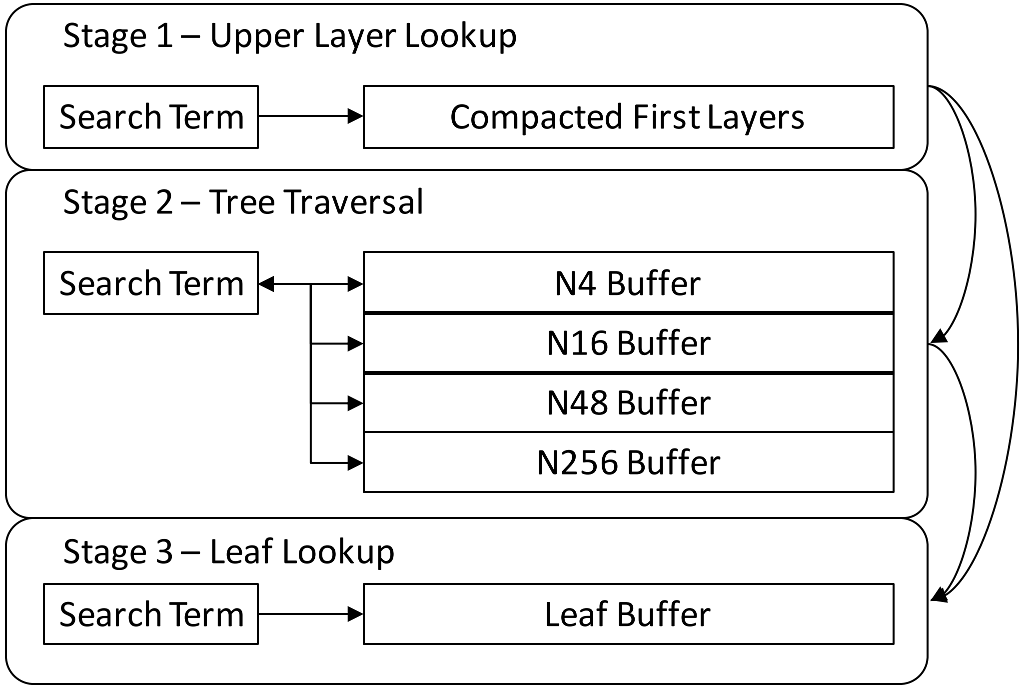
- GRT allows for arbitrarily sized strings
  - Important function, but needs slow byte-wise memory access



CuART Improvements

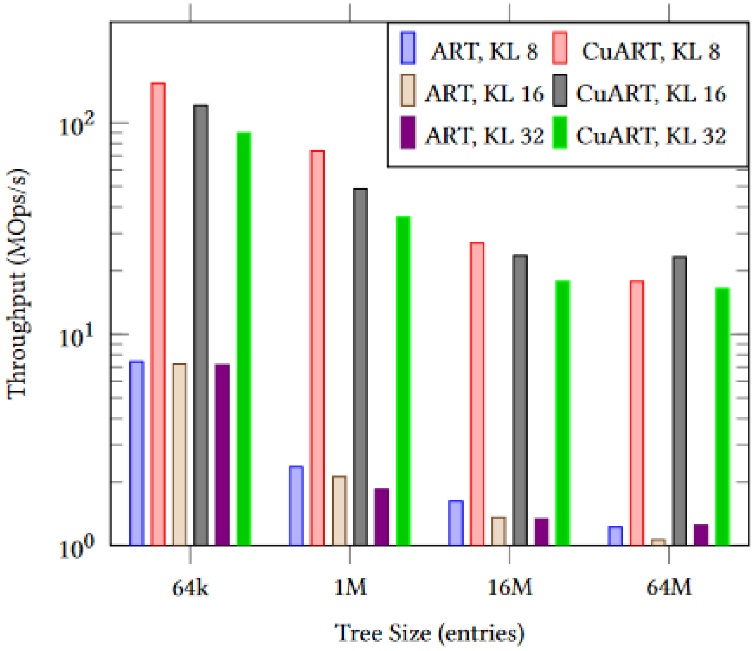
- Fold the upper three layers into one single large node (inspired by START)
- Use one buffer per node type
- Restrict leaf length to 32bytes, delegate longer keys to the CPU
- Lookahead for the next node type
  - Include the information what node comes **next** in the path already into the pointer
  - Only one memory access required

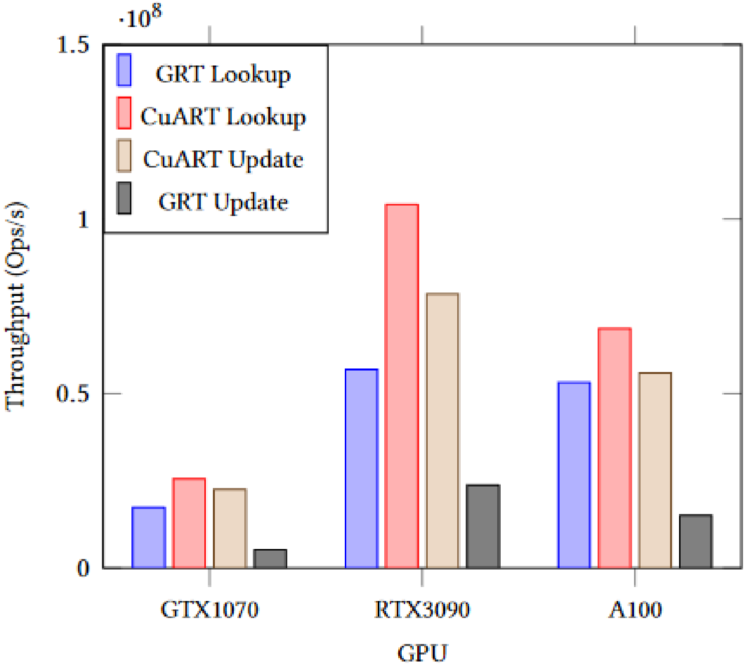


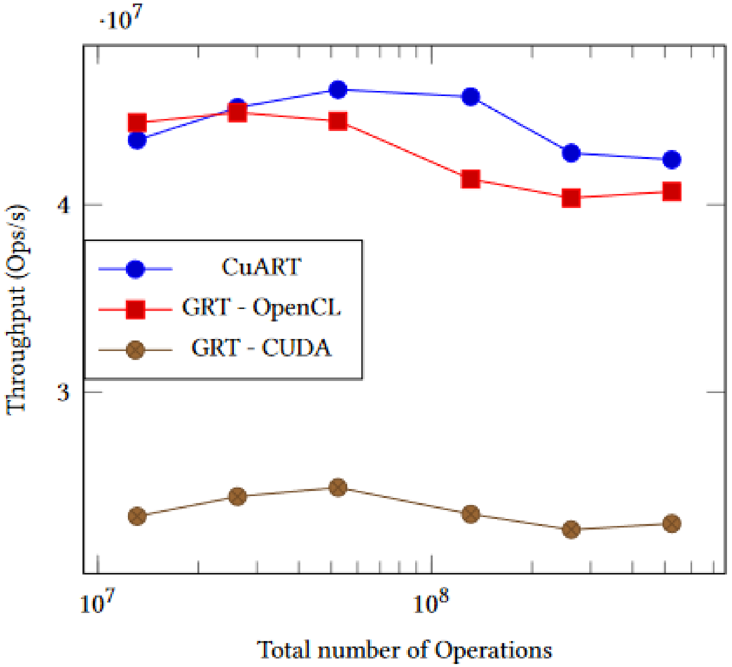


Evaluation

- 3 Test Systems: GTX 1070 (i7 8750h), RTX 3090 (Ryzen 5900), A100 (Epyc 7752)
- Implemented CuART on CPU, **same optimizations feasible**
- Works across different GPUs, variable gains
- Measurable performance gains on **real world data**

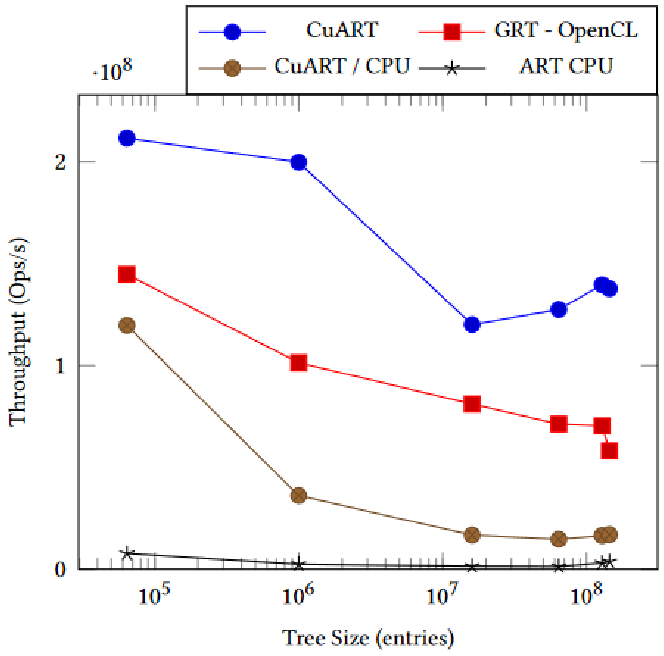


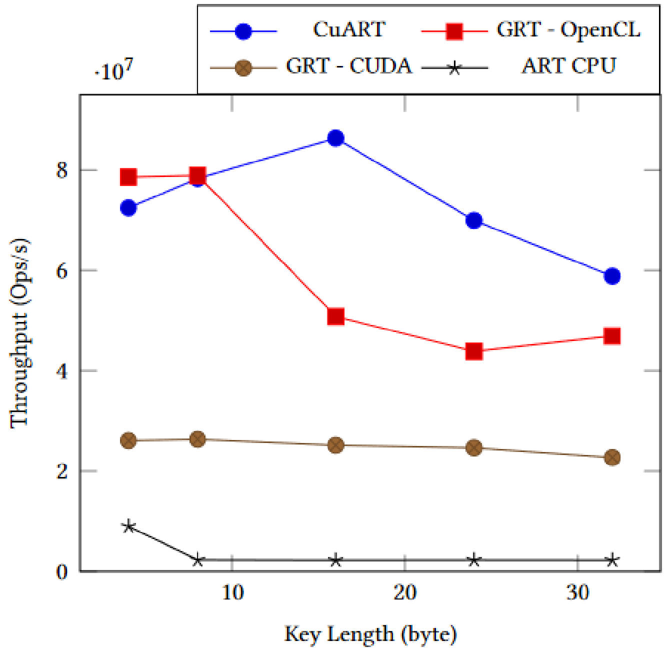


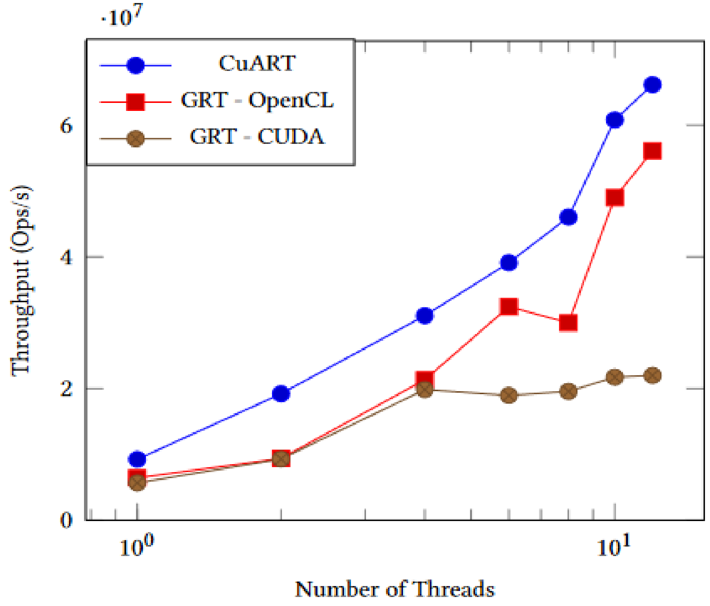


Evaluation

- Works well across a wide range of tree parameters
  - Varying tree size
  - Varying key length
- More efficient on the Host Code side

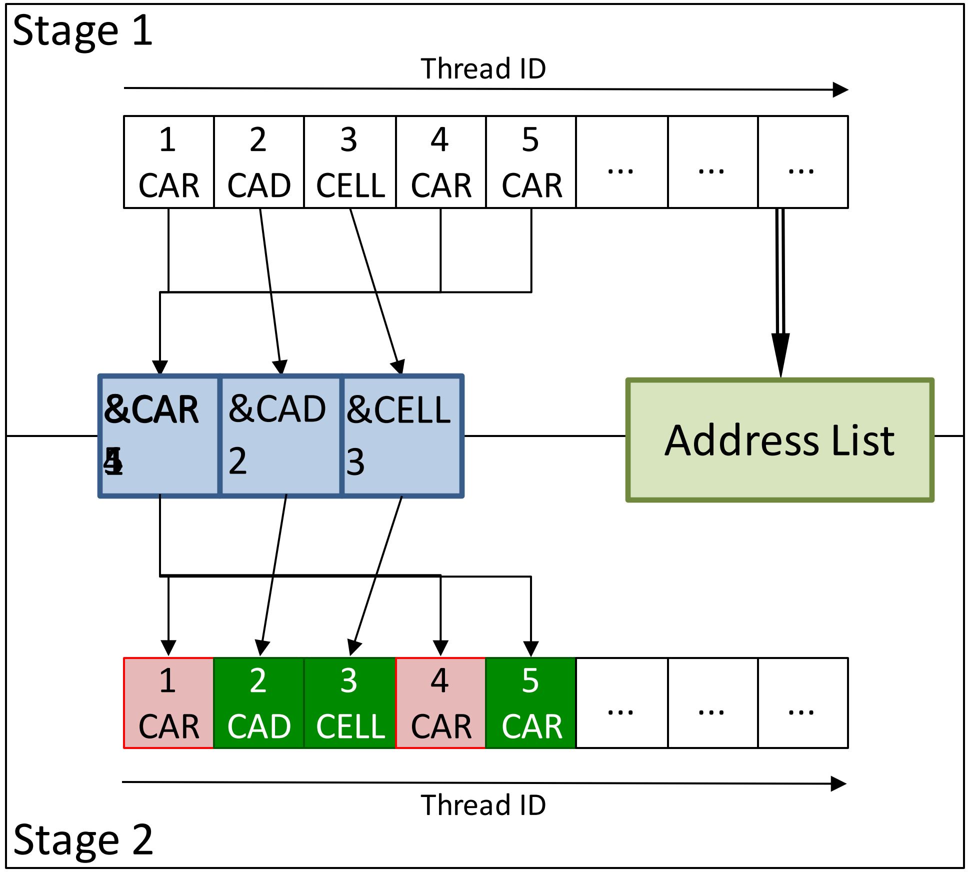


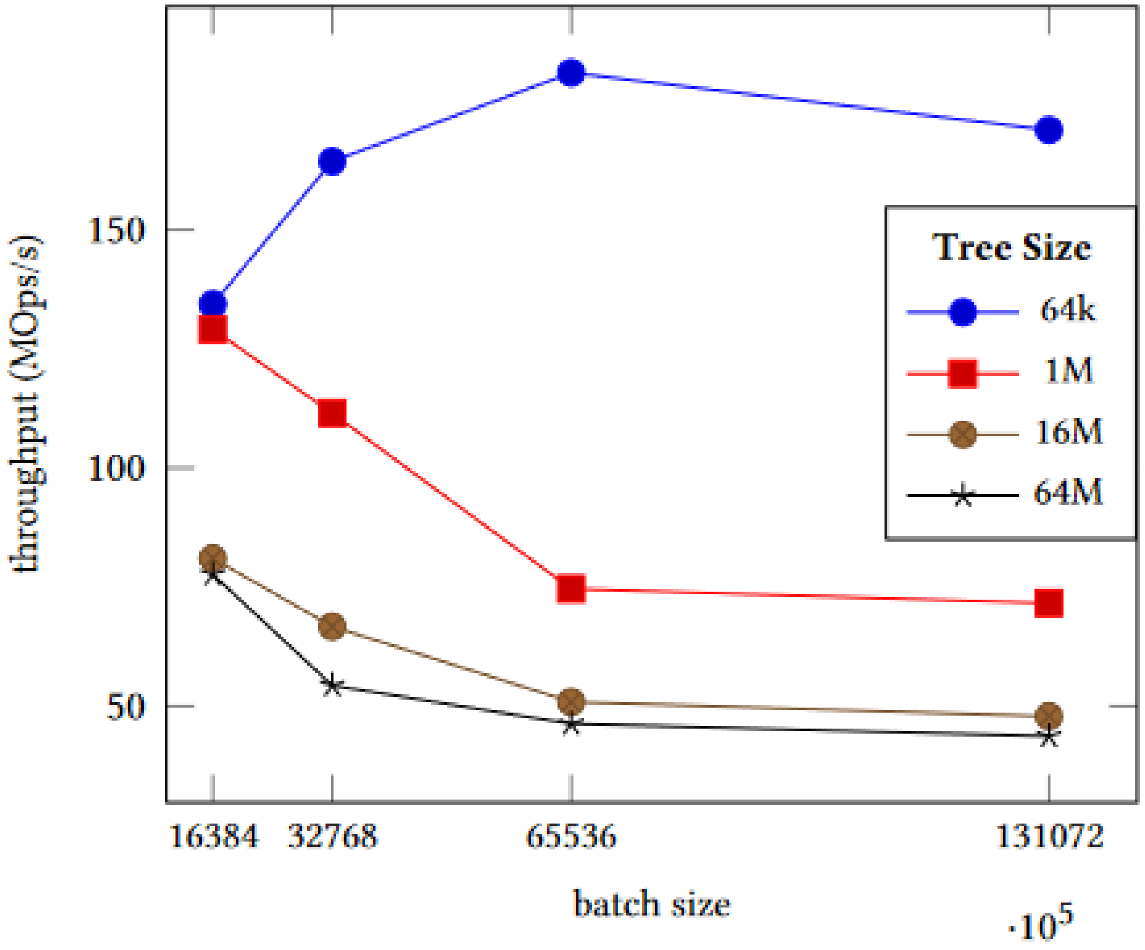




Updates

- Implemented parallel atomic updates on the GPU, using a device-side hashmap





Findings & Future Work

- Reduced number of memory transactions initiated
- Optimized updates by atomic transactions

- Outperform CPU, existing GRT implementation for point lookups
  - Over a wide range of GPUs, GDDR6X and HBM2 work well
- Same optimizations (flat buffers, type lookahead) applicable to CPU as well

- Future extensions
  - structurally modifying operations
  - Multi-GPU implementation
  - Implementation on HBM-enabled FPGAs
  - Improved handling of long keys