Processing-in-Memory for Database Query Operators

Muhammad Attahir Jibril

Processing-in-Memory

Processing-in-Memory (PIM) is a new computing paradigm that integrates processing capabilities (cores) into memory chips. This reduces memory access latency and increases bandwidth.

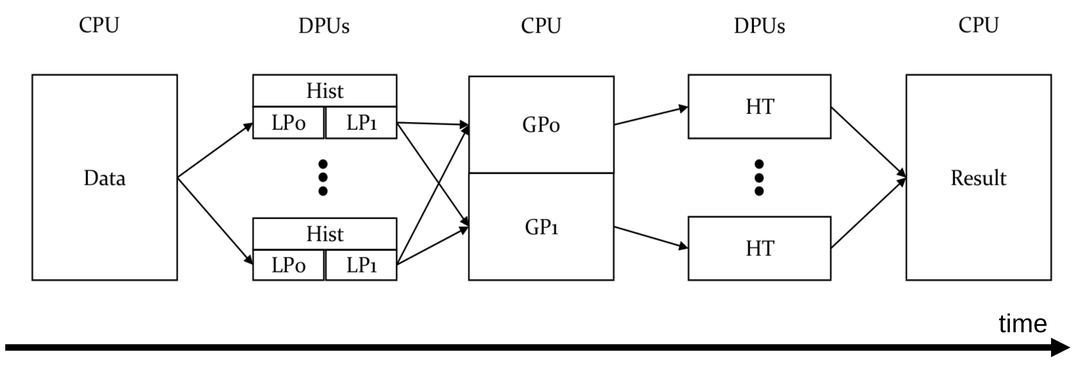
Overview

- **Use Case:** Query operators in database systems.
- **Objective:** Derive principles for designing and implementing database query operators for PIM.
- **Steps:** (1) Start with the aggregation operator to derive the design principles. (2) Validate the principles by applying them to design and implement other operators such as Sort, Join, Graph traversal etc.

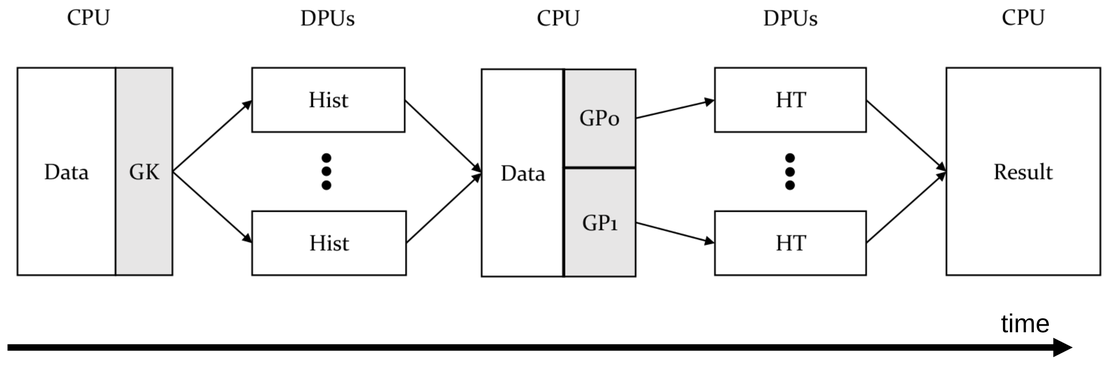
Optimizations

1. **Partitioning Phase**
   - batched partitioning
   - no. of partitions
   - histogram size
   - data cache size
   - batched transfer
2. **Aggregation Phase**
   - parallel-batched aggregation
   - hash table size
   - data cache size

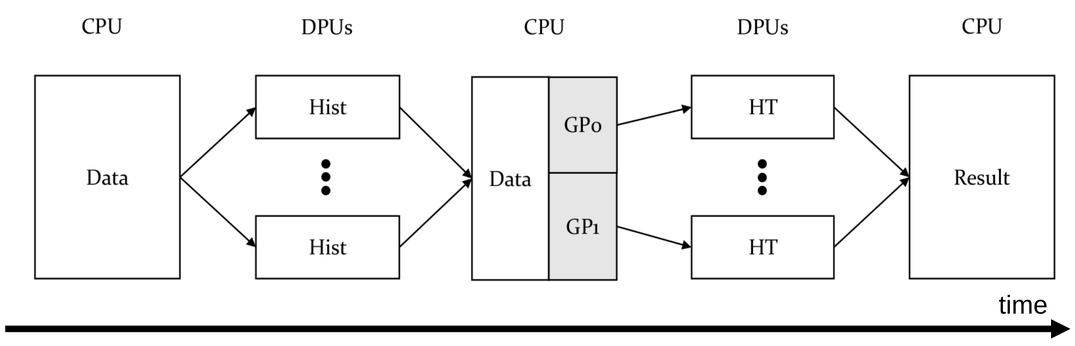
Approach



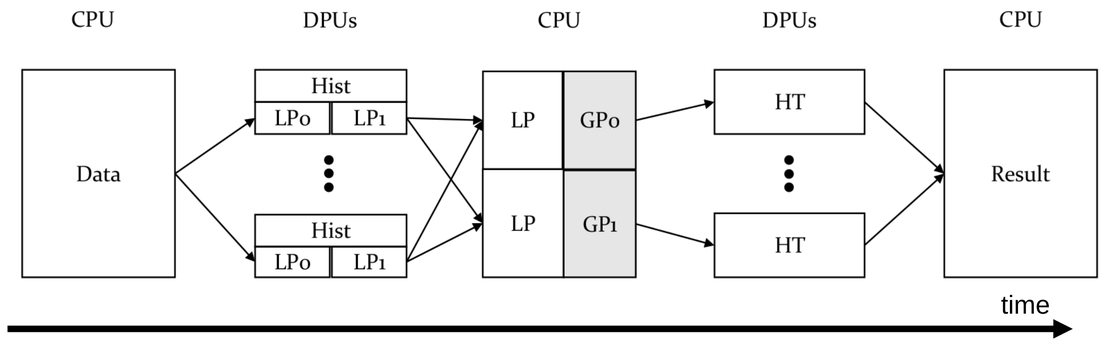
Approach A: (1) Transfer data from CPU to DPUs. (2) Compute histograms „Hist“ and build local partitions „LP“. (3) Transfer LPs to global partition „GP“ buffers on CPU. (4) Transfer GPs to DPUs. (5) Build hash tables „HT“ and compute aggregation. (6) Transfer results to CPU.



Approach B: (1) Project out grouping keys „GK“ on CPU and transfer them to DPUs. (2) Compute Hists. (3) Copy data on CPU to GP buffers. (4) Transfer GPs to DPUs. (5) Build HTs and compute aggregation. (6) Transfer results to CPU.



Approach C: (1) Transfer data from CPU to DPU. (2) Compute Hists. (3) Copy data on CPU to GP buffers. (4) Transfer GPs to DPUs. (5) Build HTs and compute aggregation. (6) Transfer results to CPU.



Approach D: (1) Transfer data from CPU to DPU. (2) Compute Hists and build LPs. (3) Transfer LPs to CPU. (4) Copy LPs on CPU to GP buffers. (4) Transfer GPs to DPUs. (5) Build HTs and compute aggregation. (6) Transfer results to CPU.