Integrating FastLanes into a JIT-Compiled GPU Query Engine

Omid Afroozeh [1] and Peter Boncz [1]

[1] Database Architectures, Centrum Wiskunde & Informatica (CWI), Netherlands
{omid,boncz}@cwi.nl

Analytical workloads on GPUs are often limited by the comparatively small on-device memory. When inputs and materialized intermediates do not fit, current engines typically fail or stream data from host memory over the slower interconnect. Compression reduces both the footprint and the transfer volume, but to realize its full potential, decoding should be fused into the query kernels. This avoids DRAM round trips, full materialization of decompressed data and kernel launch overheads. We are integrating the FastLanes file format [1] into a JIT-compiled GPU query engine that fuses each pipeline into one kernel. FastLanes's fine granular decoding enables it to be fused within the data processing kernels.

Once fused, however, decoders compete with query operators for the same on-chip resources. This raises two challenges that are invisible when decoders are benchmarked in isolation. The first challenge is register pressure. FastLanes selects an expression per column and per rowgroup from a large pool of cascadable encodings (e.g., FOR, delta, dictionary, RLE, ALP). Even a single column may thus require many different decoders. A kernel's register count and thus its occupancy are fixed by its most demanding code path. The most complex decoder therefore can penalize the entire pipeline, regardless of how much of the data actually uses it. Each decoder implementation must therefore be designed with its register footprint in mind: an optimization that speeds up one encoding (e.g., more aggressive unrolling or additional per-thread state) may raise the kernel's register count and thereby slow down all other encodings in the same pipeline.

The second challenge is where to keep the intermediates of cascaded decoding until the next decoding step consumes them, such as the bit-unpacked codes preceding a dictionary lookup. A query engine only adds to this need as pipelines typically decodes several columns at once, each with its own intermediates, alongside the state of its own operators. Neither on-chip option are fit for this: registers would only worsen the first challenge, while shared memory is contended by the same operators and also limits occupancy. We therefore stage these intermediates in global-memory buffers sized to stay resident in the GPU caches, largely avoiding DRAM traffic.

Preliminary results show that the FastLanes integration reduces PCIe transfer times and memory footprint, potentially enabling workloads that would otherwise exceed device memory. Kernel times of memory-bound queries improve, while the remaining queries incur minimal overhead.

References

[1] Azim Afroozeh and Peter Boncz. The FastLanes file format. *Proc. VLDB Endow.*, 18(11):4629–4643, 2025.