HUAWEI

Towards a *real* GPU-speedup in SQL query processing with a new heterogeneous execution framework

Adnan Agbaria | David Minor | Natan Peterfreund | Roman Talyansky | Ofer Rosenberg | Eyal Rozenberg

Heterogeneous Computing Group, Shannon Lab, Huawei Research

THIS IS A WIP

Motivation and contribution summary

- An ongoing concern of the IC&T industry is maximizing analytic DBMS performance – with GPUs showing some promise towards this end.
- Previous works have shown speedups through the use of a GPU, but it was still unclear if their solutions are competitive with more performant CPU-based DBMSes.

Contact us:

Natan Peterfreund natan.peterfreund@huawei.com

Eyal Rozenberg eyal.rozenberg@huawei.com

The contribution of this work-in-progress consists of:

1. An application-agnostic **execution environment**, (named AXE) supporting data and task parallelism over multiple GPUs and CPUs.
2. A analytic **SQL processing framework**, incorporating this engine, achieving **significant speedups** relative to a **performant** DBMS - via...
   - 2.1 Pre-processing data to enhance parallelism.
   - 2.2 Hardware-minded optimization of execution graph with rewriting rules.
   - 2.3 Breaking SQL-processing-related computational operations and constituting alternate ones, easier to optimize.

State of the art (?)

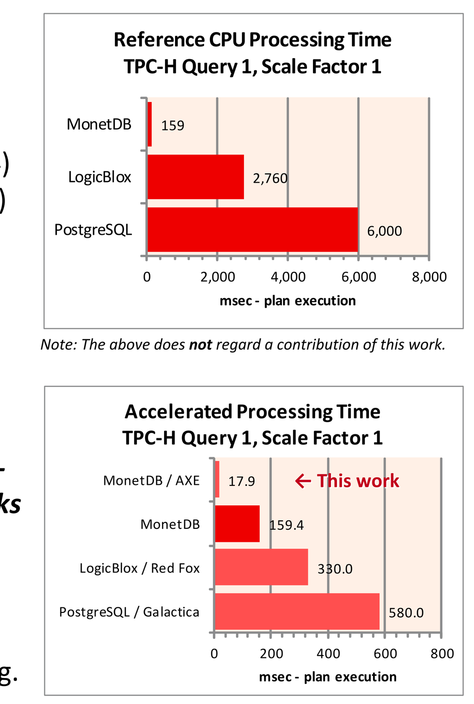
Recent work on speeding up SQL query processing, presented at this conference last year (GTC'14):

- RedFox (Wu & al., also in CGO'14)
- Galactica (Yong & al. also BDS'14)

... but the reference DBMSes used are quite slow compared to top-performing (columnar) DBMSes.

We graft our framework onto MonetDB; its unmodified CPU processing is often *faster than GPU-accelerated results in previous works (before our changing anything.)*

It seems that *we are just entering the 'ballpark'* for exploring GPGPU acceleration of SQL query processing.



CPUs used: RedFox – 2 x Xeon 2670 Galactica – 1 x Xeon 5680 This work – 2 x Xeon 2690
GPU used: RedFox – GTX Titan Galactica – Tesla K40c This work – GTX 780 Ti
While some adjustment may be necessary, the CPUs are close in performance and for the GPUs, review websites present different advantages for each card. Timing values are therefore *not normalized except for an optimistic factor 0.5 applied to the PostgreSQL CPU performance figure.* MonetDB version used: 11.15.15

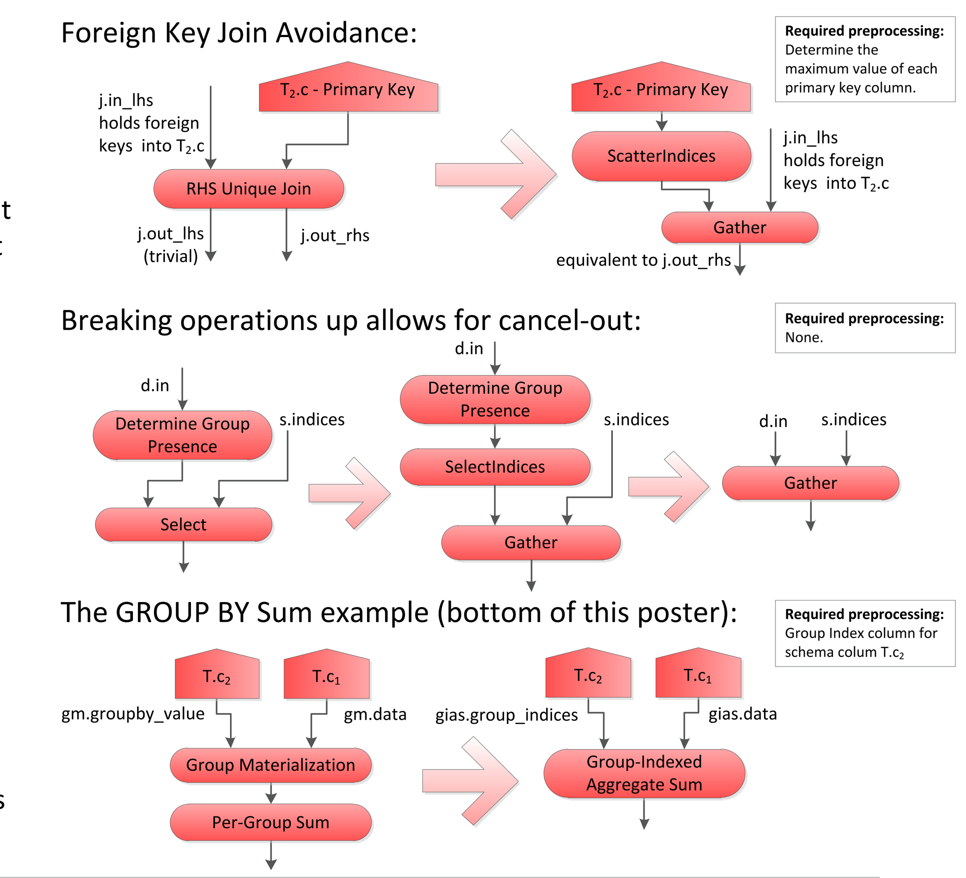
Optimization via graph rewriting

The DSL-to-execution-graph compilation component has a triple functionality:

- Compiling the host DBMS' internal execution plan representation into *graph form*.
- Transforming the graph to make it *acceptable* input for the application-agnostic execution environment (AXE).
- Effecting execution graph *optimizations*, in view of the available resources reported by AXE.

The latter two are affected through repeated application of multiple **graph rewriting rules**:

- Removing redundant nodes.
- Breaking up nodes representing complex operations into simpler, constituent operations (available to AXE)
- Re-integrating simpler computations into more complex, specifically-optimized ones.
- Replacing general-case computation with better-performing special-case
- Adjusting computation on individual input columns to utilize pre-calculated statistics.



At this time, the above is implemented only for a fragment of analytic SQL.

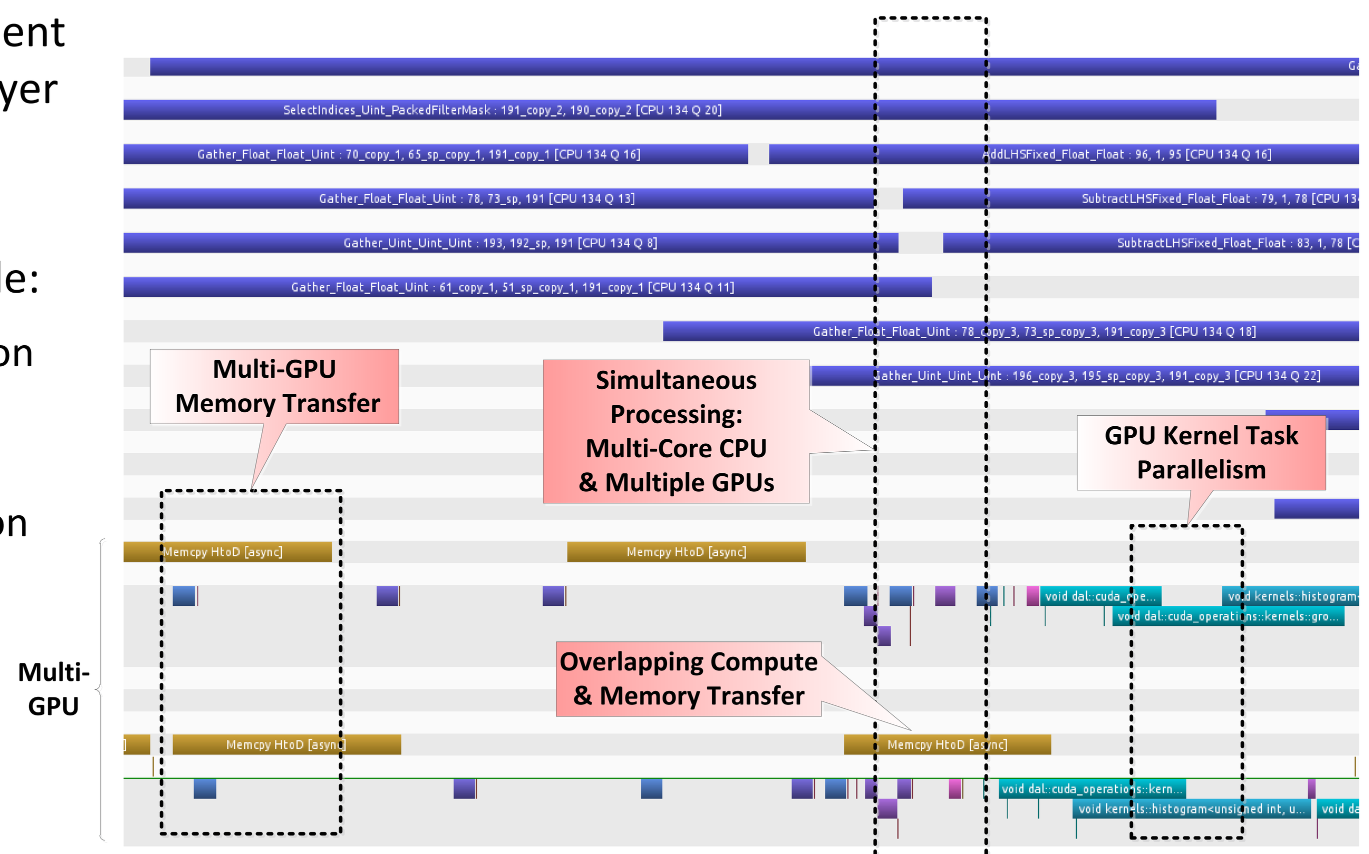
Realized plan execution

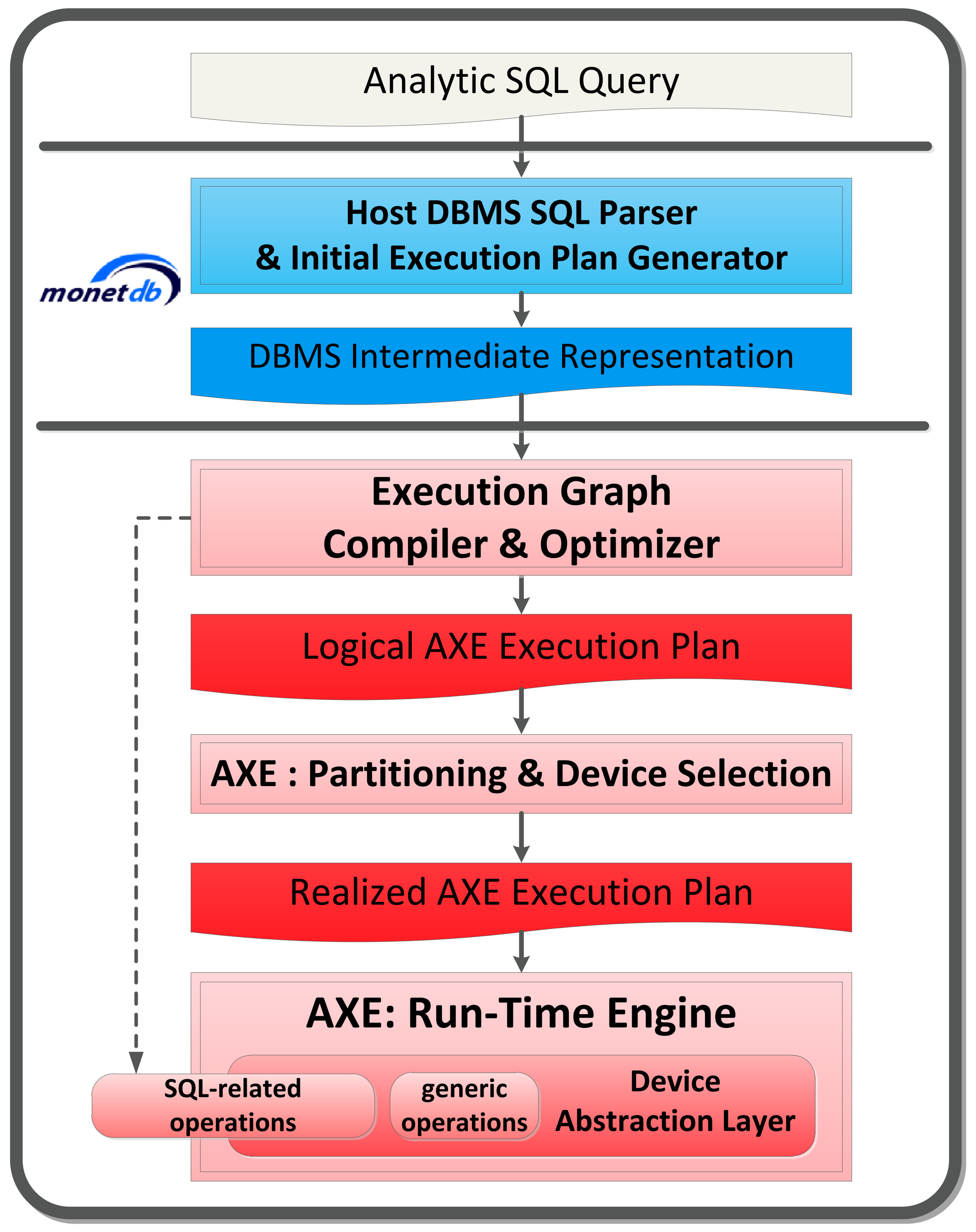
Our AXE execution environment uses its device abstraction layer to schedule execution on multiple devices.

Possible optimizations include:

- Overlapped I/O & computation
- Simultaneous I/O to multiple devices
- Parallel multi-device execution
- Graph node task parallelism
- Graph node data parallelism

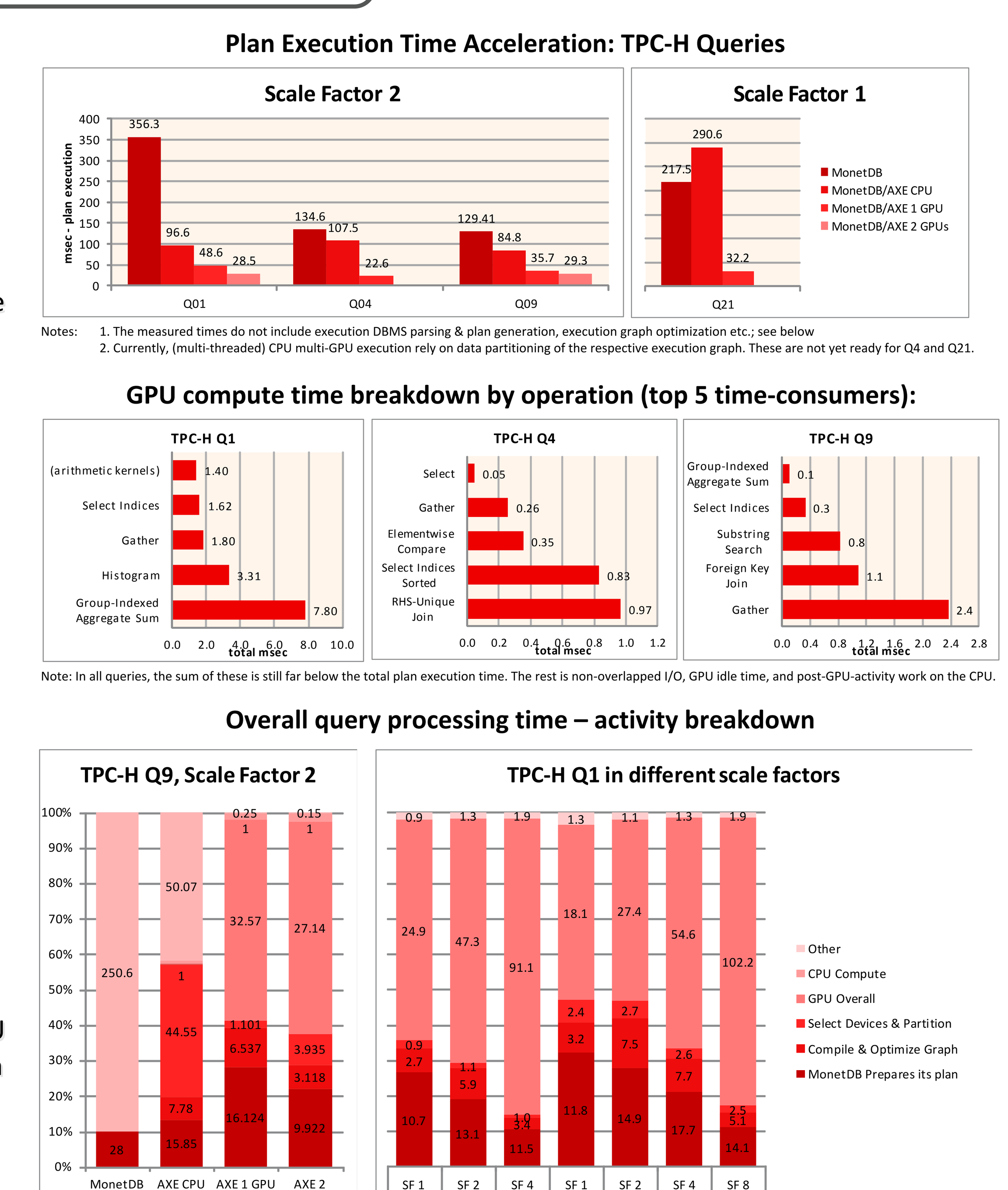
... but it's generally *not useful* to apply these *all together* to any single subgraph.





Results, and a bit of analysis

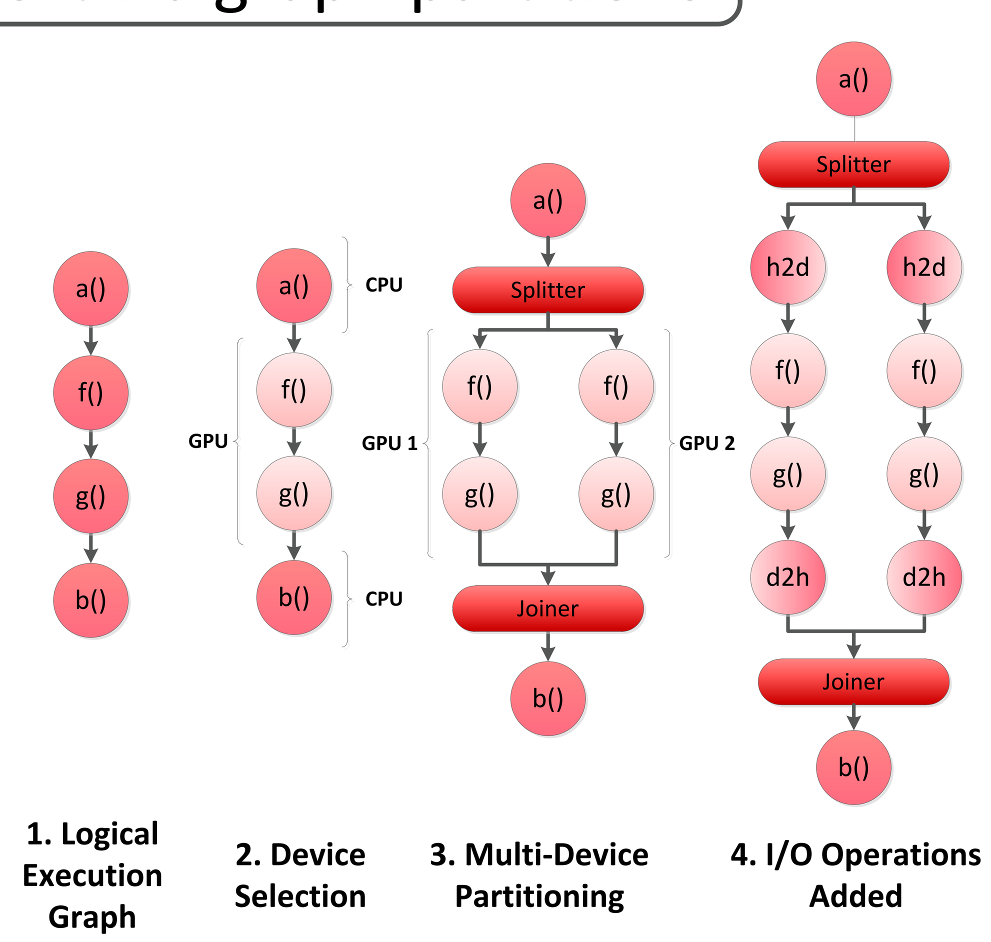
- At this time, our fragment of supported SQL only allows us to process some of the TPC-H queries (with relative efficiency). The charts present results for Q1, Q4 and Q9.
- When possible, we have timed a multithreaded CPU run with the same Logical Execution Graph, for reference (for some queries, we have still not completed necessary work to allow this).
- Our 'aggressive' graph rewriting is well reflected in the set of top GPU compute time consumers:
  - In Q1 we see the significance of the GROUP-BY Sum transformation.
  - In both Q4 and Q9 we see consumers whose use is in no way evident from the query itself (e.g. outputing the indices of bits turned on in a vector of booleans).
- It is also important to present a complete breakdown of the query processing time into the various kinds of activity other than just GPU compute and memory transfers. On the other hand, as the scale factor increases, these become less significant.



Multiple device support via graph partitions

1. The AXE execution environment receives a 'Logical' Execution Graph, from an (arbitrary) application.
2. This graph is then partitioned into subgraphs; each can be assigned to a subset of the available devices (CPUs and GPUs).
3. Subgraphs are replicated for processing on multiple devices, with splitter/joiner nodes effecting partitions of the data.
4. I/O operations are added as necessary to move data between devices' memory spaces.

The ***Logical Execution Graph*** *is now a* ***Realized Execution Graph*** *ready for execution by* ***AXE's Run-Time Engine.***



Questions to ponder: - Where do splitters/joiner nodes get run? - Why do we choose certain nodes for a subgraph? - Why choose only some devices for subgraph execution?

Preprocessing ⇒ Parallelism

Pre-calculated query-inspecific column statistics:

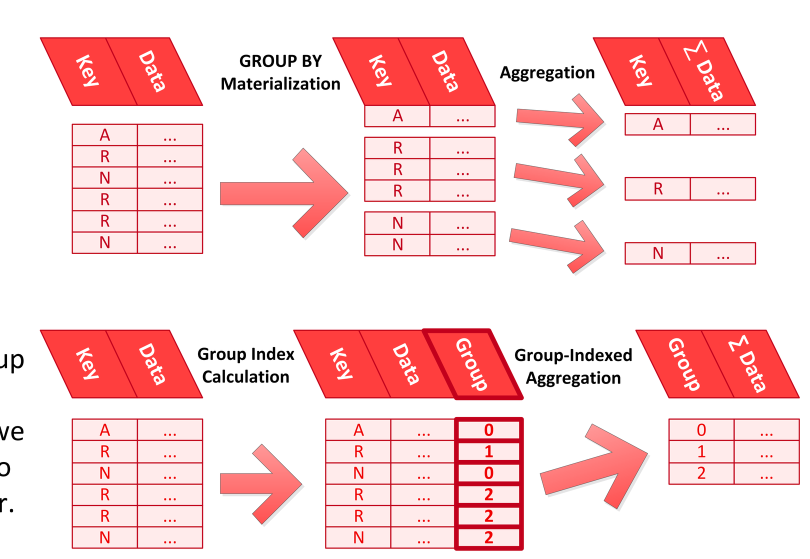
- Specifically permitted by the TPC-H benchmark.
- Can be (partially) maintained during execution.
- Allow for data regularization, and for assumptions of locality,
- which in turn facilitate parallel processing.

A common query fragment...

```
SELECT sum(Data)
FROM Table
GROUP BY Key;
```

Example: how would we execute the query fragment above?

- Typical execution, without any preprocessing of the Key column. Groups are materialized for aggregation.
- A pre-calculated Group Index for the Key column is available; we use it as an offset into the aggregates vector.



Note: Preprocessing and data regularization also benefits CPU execution of queries – but GPUs benefit more.

Some challenges for future work

We have presented results from a first iteration of development. We plan to develop it with and without relation to the use-case of SQL processing, adding several features enhancing usability as well as performance:

- Increase SQL coverage & support other DSLs.
- Support more processors types, vendors & platforms (APUs, OpenCL-based devices)
- GPU Memory management
- Automatic fusion of consecutive operations.
- Improve I/O and computation overlap.
- Fully utilize CPU vectorization capabilities to explore more optimal mixtures of execution on the CPU and on the GPU.
- et cetera.

As our framework matures, we hope to secure approval for its release as Free Software.