Horizons of Composability

Orri Erling
Software Engineer, Meta

The Present: Composable Execution is for real

- Velox accelerates Presto, Spark, Streaming and AI preproc
- A shared ABI for compute? Heterogeneous hardware?
- Language convergence, query optimization?

The Proposition, Again

- Build/maintain once for fast query everywhere
- Enable faster turnaround in building specialized solutions
- One place to optimize for new hardware
- Higher quality through more participants and more use cases
- Greater consistency of user experience between systems

Wins with Velox

- State of the art implementation of vectorized query operators
- Interactive
    - Faster and more consistent service with 1/3 of the equipment vs Java Presto
- Batch
    - 3x throughput on same cluster size
- Wins in Spark/Gluten
    - ~2.5x in TPC Benchmarks

Experience

- Industry-wide interest in composability
- Converging Presto, Spark, Streaming is possible
- Biggest divergence in built-in functions
- Proven wins in performance
- Standards, and now commoditization via open source, keep winning

Unifying?

- Back end is great, scans, joins, groupBys are the same
- Functions, everybody has their own
- Front end is harder, users are married to language/dialect
- Asks on query optimization are not as in the benchmarks

The Ongoing Inflection

- AI becomes the prime consumer, data keeps getting bigger
- Data is cheap and plentiful
- Data is 3% of world power bill
- Green data relies on efficiency gains, which rely on hardware evolution

Generations of Architecture

- General purpose CPUs have hit diminishing returns
- GPUs keep making new wins
- FPGAs could beat generic GPUs on power efficiency and cost
- If GPU is the consumer, then might as well manage data on it. Simpler data centers, fewer SKUs

Velox Futures

- Velox Wave for accelerators
- Verax for query optimization
- File formats for workload evolution

Wave

- 10+ years of data on GPU - Promising but inconclusive
- CPUs like tens of threads, GPUs need 1000x more
- Parallelism is plentiful, but it hides under abstraction
- Expose it!

Performance is locality * parallelism

- Data level = many files
- Pipeline = Many operators and many columns
- Vectorization = many rows
- Instructions = many loads
- SIMT - Many executable warps / SM

GPU vs CPU

- Best case is ratio of memory throughput, 8-15x
- Best case only possible with 100K active lanes
- Executable plan is dataflow DAG, as many steps as the longest path length
- Each step has all the compute whose prerequisites exist

Wave

- Detect suitable plans at run time
- Dataflow decomposition and fusion of operators
- Run a few splits (files) at a time, like on CPU, except every step of every file has 10K runnable SIMT lanes all the time
- So, ~100K runnable lanes across 10K Cuda cores makes for good platform utilization

Summertime Forever

- TPCH Q1
- Run the filter
  - decode the columns
- compute hash numbers and expressions
  - Group by

Getting Real

- Most workload is complex types, aggregates over arrays and maps
- It is either independent lanes all the way through or shuffled lanes to align with group by groups
- approx_percentile, set_union, map_union

Verax

- Query optimization: 500K - 1M lines, 1000 engineer years?
- Not again.
- Can we do the benchmarks in 15K lines and the real workloads in 25K?

Fundamentals, Can we get them right?

- Query graph
- It's not random rewrites of trees
- It is modeling the dependencies accurately, incl. non-inner join edges
- Spanning tree of join , allow for some nodes being included many times

Model Real Costs

- Run cost model on what you plan to execute
- Not just join cardinality but data movement too
- Want to know the future? Study history. Workloads repeat all the time
- Sampling and history from the get go

Extensibility and New Tricks

- AI, wide data need smart late materialization
- Maybe non-SQL plans, like multi-table materializations and graph analytics/training (BSP)
- Pluggability of new operators/data layouts into candidate generation
- Open for ML directed candidate generation

File Formats

- Wide
  - AI, wide data need smart late materialization
  - Maybe non-SQL plans, like multi-table materializations and graph analytics/training (BSP)
  - Pluggability of new operators/data layouts into candidate generation
  - Open for ML directed candidate generation
- Mutable
  - Privacy drives point updates to warehouse/training
  - Schema never constant
  - Semantic fidelity would like stable identifiers

File Formats (Contd.)

- Parallel
  - SIMD, GPU need formats with no data dependencies
  - Parquet/ORC encodings are a non-fit
  - Metadata is 1/3 of volume, needs random access for metadata
- Random
  - Will instance optimization/adaptivity rediscover physical design?
  - Real time, lookup-friendly
  - Sorted data enables sparse index, value-based deltas, streaming joins and aggregations?
  - Per-attribute inverted indices on demand for lookup on arbitrary attribute mix

State of Play

- Huge production wins for interactive from Velox
- Onboarding batch
- Strong concept for Wave and Verax
- File formats

Q & A