



Chronos

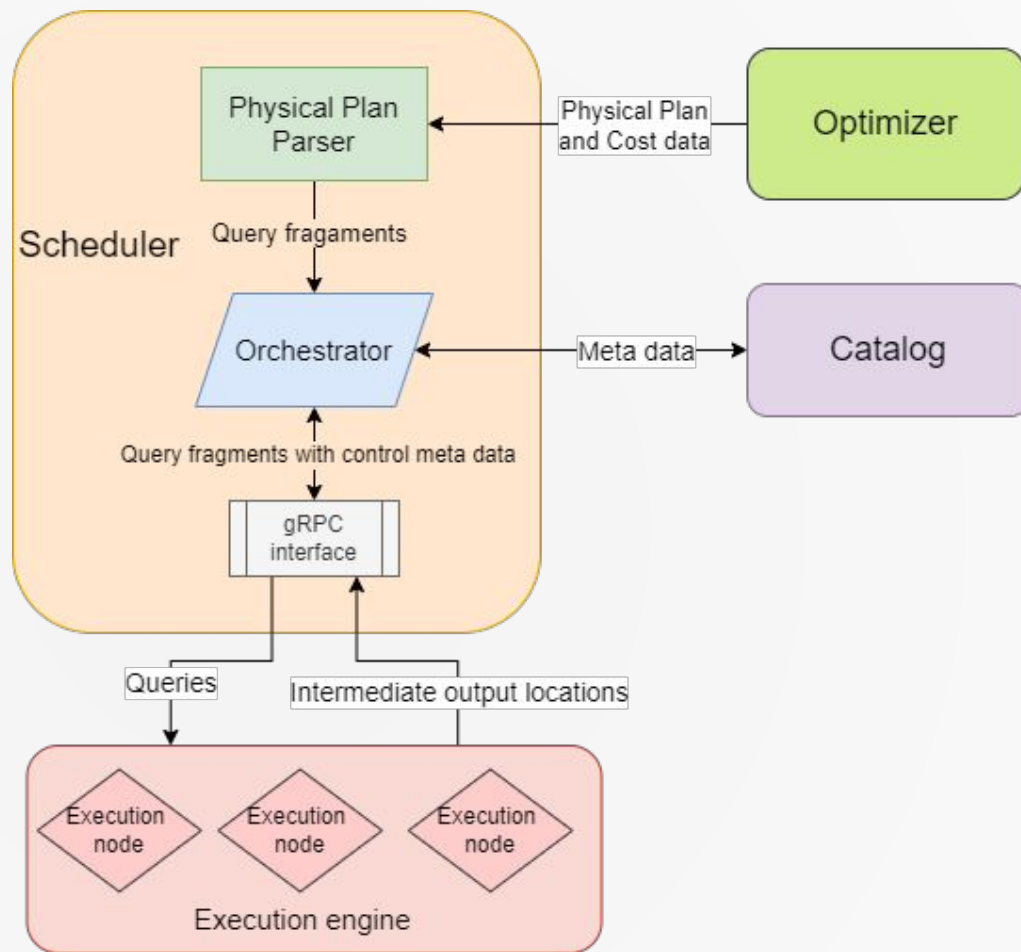
Aditya Chanana, George Li, Shivang Dalal

Outline

- Architecture and design
- Testing
- Benchmarks
- Demo
- Goals (and have we met them?)
- Future work

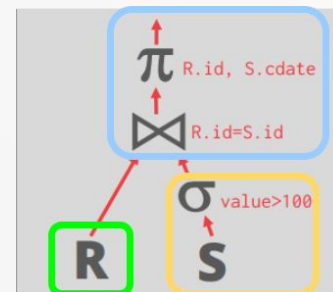
[Readme](#) for more information.

Architecture

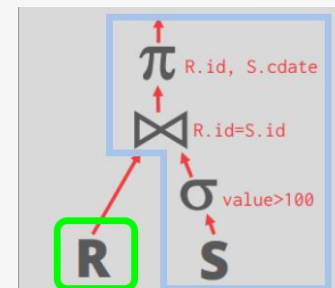


Parser

- Breaks up DataFusion execution plans into fragments
- Naive parser: Splits the plan into three fragments on every pipeline breaker
- Advanced parser: Splits the plan into two fragments and splits operators like hash join into hash probe and hash build
 - Better pipelining support
- Intermediate results are stored as partitioned Parquet files and Parquet scan nodes are inserted into parents



Naive Parser



Advanced Parser

Orchestrator

- Gives a fragment to the executor on request
- Working priority queue taking in a combination of
 - Query level priority
 - Cost of the fragment (using statistics)
 - Queuing time for the fragment
- Responsible for getting the intermediate results and updating the parent fragments. Schedule them if possible
- Takes care of aborts and updating query status

Testing Infrastructure

Unit tests

- High coverage unit tests for the Parser and Orchestrator

Executors

- Pull a query fragment from the scheduler, extract the query plan, and execute it using DataFusion executors
- Custom executors for Hash Probe and Hash Build
- Write out the intermediate results to partitioned parquet files

CLI

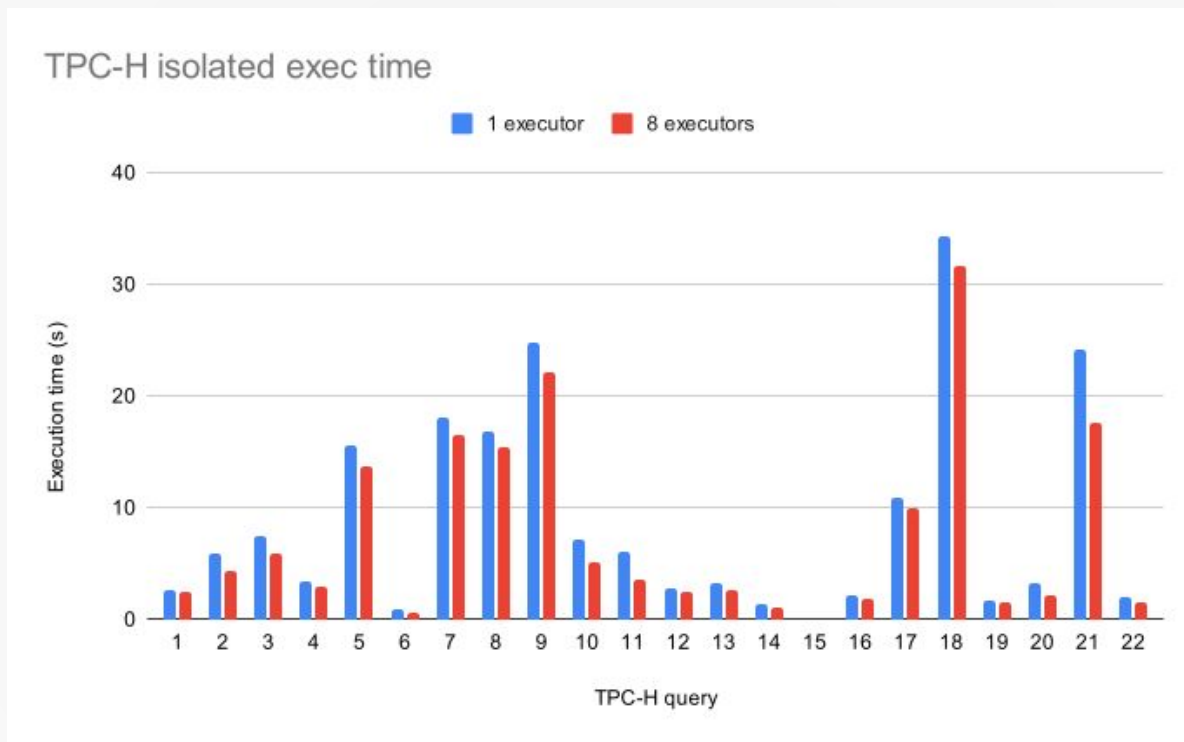
- User facing CLI which takes arbitrary SQL commands as inputs
- Uses DataFusion to convert it to parse it and convert to physical plans
- Schedules it using our scheduler API and displays the final output

Benchmark

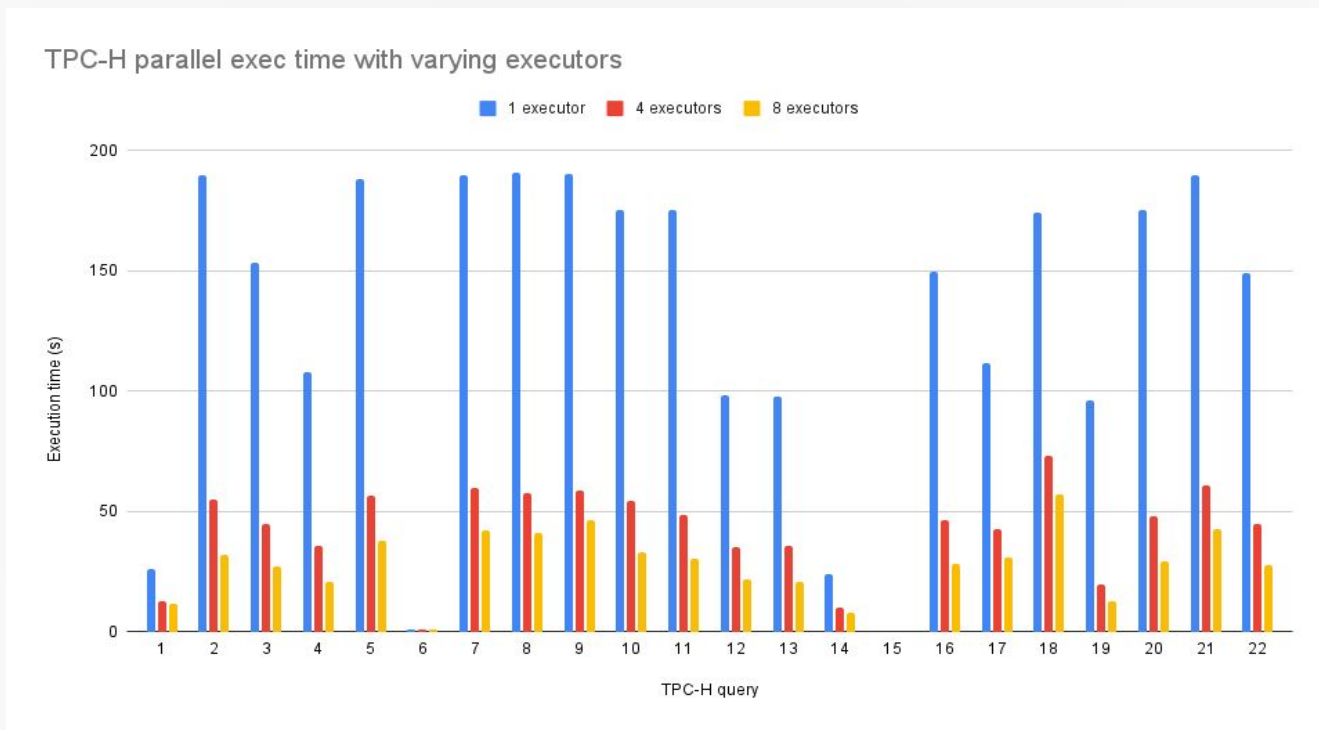
- TPC-H, SF1, All queries*
- Uses same infra as the testing one
- Varying number of executors
- Different execution modes: isolated vs parallel
- Pipelining for left-deep hash joins

* Datafusion fails on q15

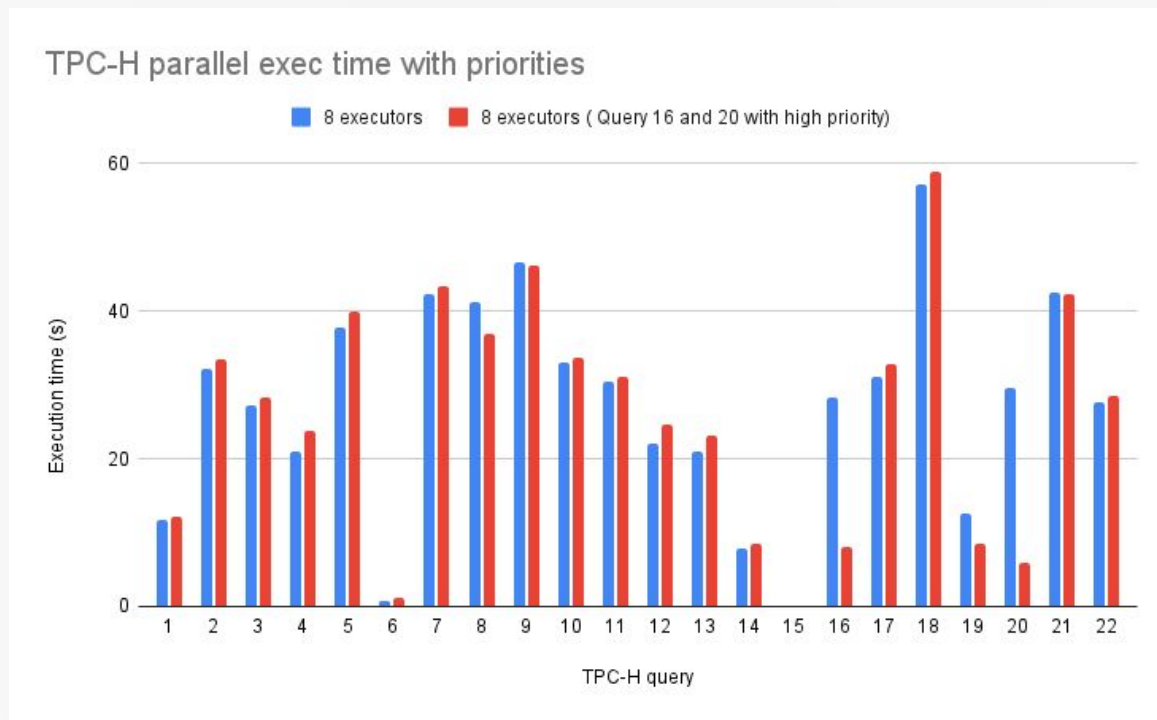
TPC-H Isolated



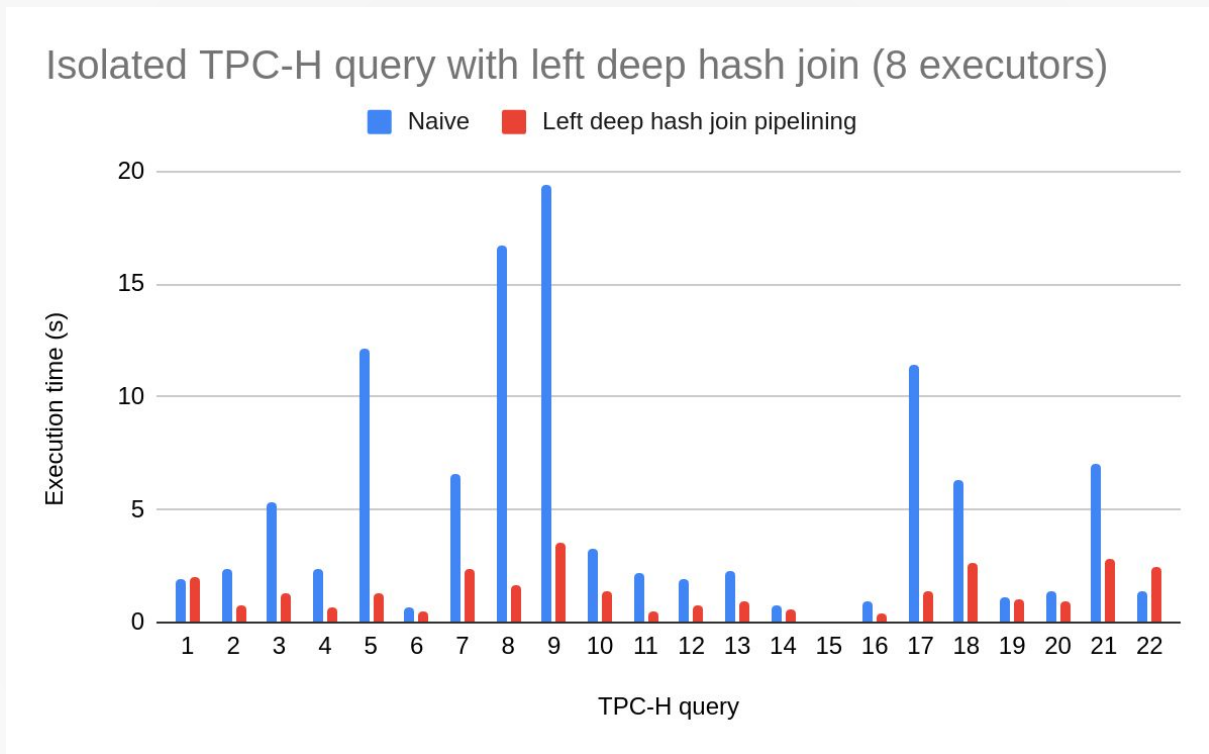
TPC-H Parallel



TPC-H Parallel with Priorities



TPC-H Isolated w/ pipelining for hash joins



Demo


Short demo of using the CLI to execute queries

Goals (and have we met them?)

- **75%** – Working scheduler that is able to interact with Executors and Optimizer ✓
- **100%** – Interleaves queries ✓
 - Achieves “fair” scheduling based on the costs of different query fragments and lineage. ✓
 - Pipeline operators ✓
- **125%** – TPC-H benchmark ✓
 - Intra operator parallelism and partition column data across execution nodes

Code quality

- Increased use of manual end-to-end testing (using CLI, TPC-H benchmarks)
 - Code coverage varies as a result
- Being agile requires using some tentative workarounds
- Write more comprehensive unit tests to prevent regression
- Good level of documentation, but lacking code examples

```
Crate chronos  source · [-]  
  
[-] A scheduler for databases.  
  
Provides inter-query and in and intra-query parallelism by splitting up queries into fragments.  
  
In general, a new query fragment is created when an ExecutionPlan has more than one child, with each child becoming its own fragment. Support is also available for pipelining hash join by splitting up the hash build phase into a separate fragment.  
  
Modules  


---

scheduler Provides APIs to schedule a DataFusion ExecutionPlan for execution.  
scheduler_interface gRPC interface to interact with the scheduler APIs. Intended for use by the query optimizer and execution engines.  
utils Util code.
```

Future Work

- Better configuration knobs
- Full support for pipelinable operators
- Better cost estimates and priority tuning
- Assign tasks to workers based on locality (morsel-esque)
- Fault tolerance
- Support in-memory intermediate results
- Spilling precomputed hash tables for hash join probing to disk/object store
- Integration with the execution engine teams' work

Thank you :)