APACHE DATAFUSION™

Andy Grove @ DataFusion Meetup

•••

March 25, 2024

My Interest in DataFusion

- Past
  - DataFusion started as a personal side project in 2017 with the overly ambitious goal of building a more modern version of Apache Spark
    - See https://andygrove.io/2018/01/rust-is-for-big-data/ for more context
  - The project quickly pivoted to become an in-process query engine
  - I started Ballista in 2019 to have another go at building a distributed query engine, using DataFusion as a foundation, but mostly stopped contributing at the start of 2023
- Future
  - I am excited about the DataFusion Comet project for accelerating Spark

Topics For Today

- DataFusion Comet
  - Accelerating Apache Spark with DataFusion
- DataFusion Python Bindings
  - Making DataFusion available as a foundation for Python systems
- DataFusion Ballista
  - Arrow-native Spark alternative

DataFusion Comet: Spark Native Engine

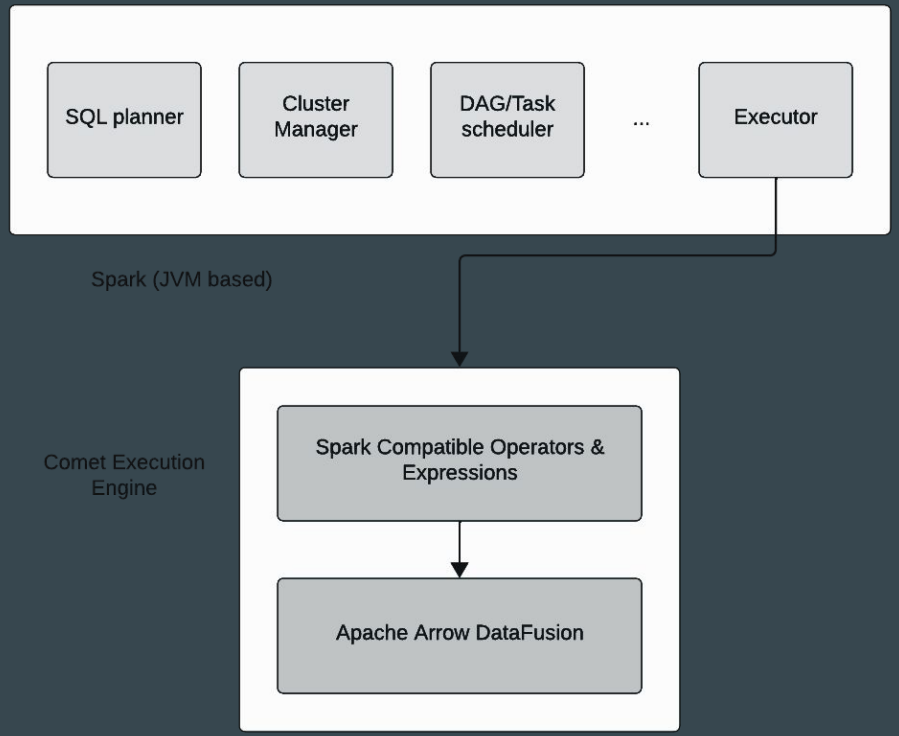
Apache Spark

- Spark is one of the most important data systems today
  - Open sourced in 2010
  - Used by 80% of the Fortune 500
  - More than 2,000 contributors
  - Mature query planner and optimizer
- The problems with Spark
  - Largely row-based volcano model, with code-gen/JIT to boost performance
  - JVM (slow startup, GC overhead, Scala-native)
  - Proprietary memory format

Apache Spark Accelerators

- Open-source (all Arrow-friendly to varying degrees)
  - Delegating to C++ code
    - Apache Gluten + Velox
    - NVIDIA Spark RAPIDS
  - Delegating to Rust code
    - Apache DataFusion Comet
    - Blaze
- Closed-source
  - Databricks Photon (C++)

DataFusion Comet Architecture



High-level work areas

- Fuzz testing to ensure compatibility with Spark
- Benchmarking against Apache Spark and Apache Gluten/Velox
- Supporting more operators, expressions, data types, etc over time
- Supporting multiple Spark versions
- Establish a release process for both Java and Rust artifacts

How will Comet help DataFusion?

- Testing at scale
  - Large potential user base
    - Spark users can test existing pipelines with Comet very easily
  - Large and diverse datasets
  - Fuzz testing will likely expose some bugs
  - Memory management / spill to disk will get tested extensively
- Drive requirements for other JVM acceleration use cases
  - Creates a new audience for DataFusion
  - Does it make sense to develop official Java bindings for DataFusion?
- Help stabilize DataFusion's public API

DataFusion Python Bindings

DataFusion Python Bindings Overview

- Features
  - Exposes DataFusion's DataFrame and SQL interfaces in Python
  - Also exposes logical query plan & optimizer
  - Substrait support
- Use Cases
  - Can be used as an general purpose library for querying data
  - Suitable for use as a SQL query planner and optimizer for other projects
    - Supports all queries in TPC-DS
    - Dask SQL switched from Apache Calcite to DataFusion Python in 2022
    - Could be used to add SQL support for others e.g. Pandas, Polars, cuDF (GPU)

DataFusion Python Bindings Overview

- Comparison to similar projects
  - Ibis is becoming a popular Python DataFrame library that can bind to multiple engines (including DataFusion), either via Python APIs or through SQL generation
  - Ibis also supports SQL -> DataFrame via sqlglot, and may support query optimization in the future
    - Collaboration potential?
  - Polars is a popular Python DataFrame library also based on Arrow. It has basic SQL support but does not support any of the official TPC-H SQL queries yet (as of version 0.19.13)

Example Usage: Executing SQL

```
from datafusion import SessionContext

ctx = SessionContext()
ctx.register_parquet('taxi', 'yellow_tripdata_2021-01.parquet')

df = ctx.sql("select passenger_count, count(*) "
             "from taxi "
             "where passenger_count is not null "
             "group by passenger_count "
             "order by passenger_count")

batches = df.collect()
```

Logical Plan

```rust
#[pyclass(name = "LogicalPlan", module = "datafusion", subclass)]
#[derive(Debug, Clone)]
pub struct PyLogicalPlan {
    pub(crate) plan: Arc<LogicalPlan>,
}
```

Logical Plan Variants

```rust
#[pymethods]
impl PyLogicalPlan {
    /// Return the specific logical operator
    pub fn to_variant(&self, py: Python) -> PyResult<PyObject> {
        Python::with_gil(|_| match self.plan.as_ref() {
            LogicalPlan::Aggregate(plan) => PyAggregate::from(plan.clone()).to_variant(py),
            LogicalPlan::Analyze(plan) => PyAnalyze::from(plan.clone()).to_variant(py),
            LogicalPlan::CrossJoin(plan) => PyCrossJoin::from(plan.clone()).to_variant(py),
            ...
```

Example: Dask SQL

- Dask is a Python library for parallel and distributed computing
- Dask SQL adds SQL support to Dask
  - Originally used Calcite for SQL parsing and planning
  - Switched to DataFusion in 2022
  - Python code creates a DataFusion logical plan and then transpiles that plan to Dask operations

Contributing

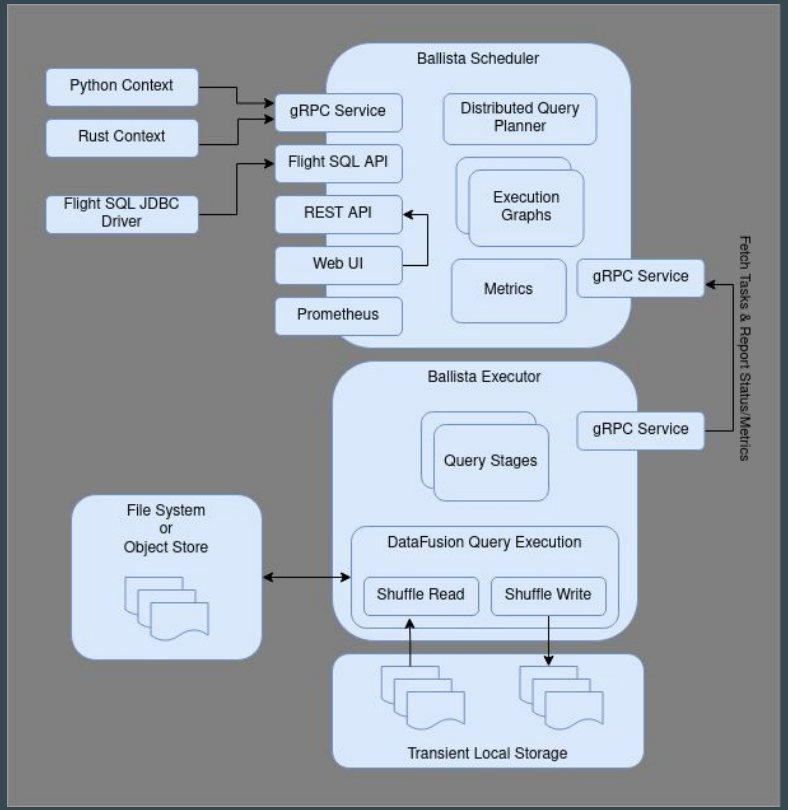
- For the project to gain traction, we need more users & maintainers
  - Evangelism is needed to promote DataFusion as a mature SQL query planner and optimizer for the Python ecosystem
  - Competing with Ibis or Polars as a DataFrame API does not seem to be the best use of time
- Areas to contribute:
  - Improve support for Ibis integration
  - Improve the documentation & examples
    - Particularly for the query planner use case
  - Blog posts
  - Testing and bug reporting

Apache DataFusion Ballista

Ballista Overview

- Distributed SQL query engine
- Inspired by Apache Spark, but with some key differences
  - Arrow-native
    - Arrow Flight SQL support
    - Arrow Flight interface between processes
    - Arrow IPC used for shuffle files
  - Implemented in Rust, but language-agnostic architecture
    - Query plans represented in protobuf format
    - Pluggable execution engine (defaults to DataFusion)
    - Could support WASM, Python UDFs with some engineering effort

Ballista Architecture



Ballista Status

- Ballista has much potential but has failed to gain much traction
- Possible reasons
  - Only a small percentage of companies have need for distributed compute?
  - Too much investment to reach feature parity with other systems, such as Apache Spark?
  - Project requires skill sets across multiple areas?
    - Devops
    - Distributed Systems
    - Scheduling
    - Query Engines
    - UI (React)
    - Python
  - Poor documentation?
  - Companies are working on forks of the project, rather than extending?

Thanks for listening!

- Contact details
  - LinkedIn: https://www.linkedin.com/in/andygrove/
  - Email: agrove@apache.org
- Social Media
  - X: @andygrove_io
  - BlueSky: https://bsky.app/profile/andygrove.bsky.social