Draft of blog post to be published on Apache site.
GitHub issue: https://github.com/apache/arrow-datafusion/issues/5201

PR against arrow-site: https://github.com/apache/arrow-site/pull/322

DataFusion Adds Support for Substrait

The Apache Arrow PMC is pleased to announce that the DataFusion project has accepted the donation of the datafusion-substrait crate, which was developed by the DataFusion community under the datafusion-contrib GitHub organization.

Substrait provides a standardized representation of query plans and expressions. In many ways, the project's goal is similar to that of the Arrow project. Arrow standardizes the memory representation of columnar data. Substrait standardizes the representation of operations on data, such as filter and query plans.

Now that DataFusion can directly run Substrait query plans, there are several exciting new integration possibilities:

- Pass serialized query plan across language boundaries, such as passing from Python to Rust or Rust to C++. For example a python based SQL frontend could pass a Substrait plan to DataFusion which is written in Rust.
- Mixing and matching query engine front-ends and back-ends based on their specific strengths. For example, using DataFusion for query planning, and Velox for execution, or Calcite for query planning and DataFusion for execution
- Easier integration for other DataFusion based projects. For example, the related Ballista project, which already provides “distributed DataFusion” execution plans, serializes query plans using a protobuf format that predates the Substrait project. By adopting Substrait, Ballista can provide distributed scheduling for query engines other than DataFusion.

Logical Plan Support

DataFusion currently supports serialization and deserialization of the following logical operators and expressions with Substrait.

DataFusion	Substrait Type	SQL	DataFusion Supported Subtypes
Projection	Relation	SELECT	
TableScan	Relation	FROM	
Filter	Relation	WHERE	
Aggregate	Relation	GROUP BY	
Sort	Relation	ORDER BY	

Join	Relation	JOIN	LEFT, RIGHT, FULL, LEFT ANTI, LEFT SEMI
Limit	Relation	LIMIT	
Distinct	Relation	DISTINCT	
SubqueryAlias	Relation	<subquery> AS <alias>	
AggregateFunction	Expression		
Alias	Expression	<column> AS <alias>	
Column	Expression		
BinaryExpr	Expression		
Between	Expression	BETWEEN	
Case	Expression	CASE ... WHEN ... END	
Literal	Expression		Int8, Int16, Int32, Int64, Boolean, Float32, Float64, Decimal128, Utf8, LargeUtf8, Binary, LargeBinary, Date32, NULL(Int8 | Int16 | Int32 | Int64 | Decimal128)

Physical Plan Support

There is also preliminary work on supporting serialization of physical plans. The tracking issue for this is TBD.

Python Bindings

Substrait support is also available from DataFusion's Python bindings. Here is an example.

```
from datafusion import SessionContext
from datafusion import substrait as ss

# Create a DataFusion context
ctx = SessionContext()

# Register table with context
ctx.register_parquet('aggregate_test_data', './testing/data/csv/aggregate_test_100.csv')

substrait_plan = ss.substrait.serde.serialize_to_plan("SELECT * FROM aggregate_test_data", ctx)
# type(substrait_plan) -> <class 'datafusion.substrait.plan'>
```

```
# Alternative serialization approaches
# type(substrait_bytes) -> <class 'list'>, at this point the bytes can be distributed to file, network, etc
safely
# where they could subsequently be deserialized on the receiving end.
substrait_bytes = ss.substrait.serde.serialize_bytes("SELECT * FROM aggregate_test_data", ctx)

# Imagine here bytes would be read from network, file, etc ... for example brevity this is omitted and variable
is simply reused
# type(substrait_plan) -> <class 'datafusion.substrait.plan'>
substrait_plan = ss.substrait.serde.deserialize_bytes(substrait_bytes)

# type(df_logical_plan) -> <class 'substrait.LogicalPlan'>
df_logical_plan = ss.substrait.consumer.from_substrait_plan(ctx, substrait_plan)

# Back to Substrait Plan just for demonstration purposes
# type(substrait_plan) -> <class 'datafusion.substrait.plan'>
substrait_plan = ss.substrait.producer.to_substrait_plan(df_logical_plan)
```

Availability

Substrait support is available in DataFusion 18.0.0 and version 0.8.0 of the Python bindings.

Get Involved

The Substrait support is at an early stage of development, and we would welcome more contributors to expand the functionality and to help with compatibility testing with other data infrastructure that supports Substrait.

If you are interested in getting involved, an excellent place to start is to read our communication and contributor guides.