Replacing Spark with DataFusion

Processing billions and billions of rows at speed

Bruce Ritchie

Principal Software Engineer, Veeva Systems

aka Omega359

Opinions are my own and are not the views of my employer

The Problem

Spark isn’t scaling well and costs
too much time and money

What's the concern with Spark?

- Spark is an amazing piece of technology with a few issues:
  - Random failures
  - OOM (code 137)
  - Disk full (code 100)
  - More flags than a political event
  - Issue root causes are often quite hard to diagnose
  - Very easy to generate millions of files
  - Performance isn't optimal and tuning can be a challenge
- Skewed data can result in very long run times
  - A lot of data comes in the very last day of the year for example
- Performance and costs are trending in the wrong direction
- Tests are critical to success
  - They are only good for correctness of the logic though
- Large new features that impact performance can be very expensive to test

DataFusion

Is building your own query engine the solution?

Is DataFusion a viable alternative?

- Would the improved performance be enough?
- Is the workload partitionable and vertically scalable?
- Is the required SQL functionality available?
- Is the dataframe API complete (and lazy)?
- Has udf's?
- Are the joins small enough to work in memory?
- Does the organization have the resources and skills to migrate?
  - … and maintain?

For my project the answer was yes.

POC

Rewrite it in Rust

What is being rewritten?

- Project ingests data from parquet files on S3, applies numerous transformations on it, saves it back out to S3 for further processing
- Multi-stage (prepare/ingest/snapshot) with different execution modes
- Ingest stage is being rewritten into Rust backed by DataFusion
  - Responsible for most transformations (and execution time)
  - Joins are against small reference tables
  - Easily partitionable
- Ingest stage relies on a Scala library that handles most of the transformations
  - Transformations are defined in json/yaml files and applied in file order
  - Transformations include conditional if/then/otherwise, rename, Spark expressions, type mapping, dictionary replace (with a variety of conditions such as if exists/not exists, regex, etc) and many more
  - More advanced transformations are handled in Scala and udf's

Yaml example

```yaml
- attribute: id_field_x
  nullable: true
  transformers:
    - transformer: setDefaultValue
      defaultValue: null
      type: string
    - transformer: conditional
      condition:
        when:
          expression:
            expr: >-
              upper($id_field_x$) == 'INTEG' AND regexp_match($fieldx$, '^[0-9]{4}ZZZ.*') IS NOT NULL
        then:
          expression:
            expr: substr($fieldx$, 1, 4)
        otherwise:
          setValue:
            attribute: $id_field_x$
    - transformer: dictionaryReplace
      inAttributeName: false
      inAttributeValue: true
      caseSensitive: false
      mapName: ID_X_MAPPING
      expr: $source$ == 'SOURCE_13'
    - transformer: expression
      expr: regexp_replace($id_field_x$, '[^a-zA-Z0-9]', '')
    - transformer: upper
    - transformer: conditional
      condition:
        when:
          expression:
            expr: length($id_field_x$) <= 2
        then:
          setValue:
            value: null
        otherwise:
          setValue:
            attribute: $id_field_x$
    - transformer: conditional
```

How is the POC going?

- Started initial investigation Dec 2023
- Started with migrating the Scala/Spark transformation library
- First started with Polars
  - needed lazy support which was (is?) incomplete
  - Needed more functions then were implemented
- Switched to DataFusion
  - First verified I could migrate Spark expressions and udf's
- Submitted PR's for a variety of missing functions
  - to_timestamp with formats
  - to_date with formats
  - make_date
  - upper/lower bug fix for non-ascii
  - regexp_like
  - to_char

How is the POC going?

- Initial POC of DataFusion-based library completed in late January
- Started rewrite of Spark ingest phase early February
  - First complete build finished beginning of March
  - Stage migration took the majority of time
  - Spark UDF -> DataFusion UDF was a non-trivial exercise
  - 4k loc
- Performance is exceptional
  - 15% of the cost, 7.5x faster

Next Steps

- Write/migrate tests (~10k loc)
- Update persistence code to use iceberg tables
- Handle deletes
- Cleanup/refactor code
- Update surrounding stages to work with new DataFusion ingestion stage
- Use learnings to help other internal teams with performance critical code
  - Spark -> DataFusion
  - Pandas
  - Wasm
- Continue contributing to DataFusion

Lessons Learnt

Lessons Learnt

- The Rust learning curve is real for programmers coming from a higher-level language background
  - Or maybe it's just me :D
- Programming paradigms from other languages do not always translate
- Scaling up is often a viable alternative to scaling out
- Working on an open-source project can be quite fulfilling
- It takes a lot longer to rewrite a project into a completely new language than I imagined
- DataFusion is happily more mature than I expected it to be
- Full test runs are (unfortunately) a great time to walk the dog

Thank you