New Avro Serialization for Beam SQL

By Talat Uyarer

Austin, 2022
Agenda

● Who am I?
● Mission
● Benchmark Results
● Let’s Define Root Cause
● Our Solution
● Internals of Avro Row Library
● What is next?
● Questions?
Who am I

- Living in San Francisco Bay Area since 2015
- Sr Principal Engineer at Palo Alto Networks (Cortex Data Lake Team)
- Software developer for 15+ years
- Proud Member of Apache Software Foundation
- Passionate about open-source big data projects
- Apache Beam user since early 2019
Mission

Stream processing jobs have very high latency when we use BeamSQL.

How can we improve latency while using BeamSQL?
Let’s Define Root Cause
Beam SQL

How Beam SQL works

How Beam turn a String as runnable code
How Beam SQL Works

Beam SQL (via Java)

SELECT key, SUM(value)
FROM input GROUP BY key
How Beam SQL Works

**Beam Java**

```java
input.apply(
  SqlTransform.query(sql))
```

**Beam SQL (via Java)**

```sql
SELECT key, SUM(value)
FROM input GROUP BY key
```
How Beam SQL Works

**Beam Java**
```
input.apply(
  SqlTransform.query(sql))
```

**Beam SQL (via Java)**
```
SELECT key, SUM(value)
FROM input GROUP BY key
```

**Apache Calcite**
- Parse to AST
- Validate AST
- Convert to Logical Plan
- Convert to Physical Plan

**Pipeline**

**Apache Flink**

**Cloud Dataflow**

[https://docs.google.com/presentation/d/1beOx9ydGQgO8FHg862rhBjWwexisWMvwI0E0EszdpeE0/edit#slide=id.g33b013e8f5_1_23]
Sample Code Generation

- Generates Java code for SQL operators

  ```java
  SELECT id, convert(price), price * 10 WHERE item = "my item" ...
  
  Becomes like as below. Accepts input Row format and write out in Row format.
  ```

  ```java
doFn(Context c, Row r) {
    if ("my item".equals(r.get(2))) {
      int price = r.get(1);
      c.output(new Row(r.get(0),
                   MyUdf.convert(price), price * 10));
    }
  }
  ```

  [Link to presentation slide](https://docs.google.com/presentation/d/1beOx9ydGQgO8FHg862rh8jWwexisWMvwlm0Eszdp_E0/edit#slide=id.g33b013e8f5_2_18)
Beam SQL

How Beam SQL Tables work

If Beam SQL needs Row, How Tables can run SQL?
Sample Pipeline

kafka -> BEAM -> Object Storage
End 2 End Pipeline for Avro

1. Read Avro Binary from Data Store
2. Convert Avro Binary to Generic Record
3. Convert Avro Generic Record to Beam Row
4. Beam SQL Generated Code
5. Beam SQL Output to Avro Generic Record
6. Avro Generic Record to Avro Binary
7. Write to Sink
Beam SQL

Performance Issue

We create two objects for each Avro Record.

We waste our CPU cycle and memory.
Beam Pipeline Profiling

https://medium.com/google-cloud/profiling-dataflow-pipelines-ddbef07761d
How we can improve this?

Java code snippet:

```java
DatumReader<GenericRecord> reader = new GenericDatumReader<GenericRecord>(writerSchema, readerSchema);
Decoder decoder = DecoderFactory.get().binaryDecoder(avroBytes, null);
GenericRecord record = reader.read(null, decoder);
Row row = AvroUtils.toBeamRowStrict(record, AvroUtils.toBeamSchema(readerSchema));
```
Our Solution

Austin, 2022
Our Solution - Initial Approach

- Develop a Hand Crafted a De/Serializer
- Pros
  - Easily verify performance improvement
- Cons
  - Update code for every schema changes
  - Not good for production
Our Solution - Code Generation

- Did a research little bit.
- Found RTBHouse's Avro Code Generator
- They handle every seamlessly
- Reimplement Same idea for Avro Row

https://techblog.rtbhouse.com/2017/04/18/fast-avro/
Usage of AvroRow Library

```java
public class AvroBytesToRowConverter extends DoFn<byte[], Row> {

    private SerDesRegistry registry;

    @Setup
    public void setup() {
        // Create Registry in Setup
        registry = SerDesRegistry.getDefaultInstance();
    }

    @ProcessElement
    public void processElement(ProcessContext c) {
        // Get avro byte record
        byte[] record = c.element().getValue();

        // Read schema
        Schema writerSchema = ...
        Schema readerSchema = ...

        // Create Avro decoder
        Decoder decoder = DecoderFactory.get().binaryDecoder(record, null);
        // Get Deserializer
        RowDeserializer<Row> deserializer = registry.getRowDeserializer(writerSchema, readerSchema);
        // Deserialize Avro to Row
        row = deserializer.deserialize(decoder);
    }
}
```
Benchmark Results
Based on Throughput wise

- After start using new library. We reached ~ 5x throughput with same amount of resources.
After Enable new Avro SerDes

- When we roll out on production. We see 4x less resource consumption with Dataflow Autoscaling
Internal Of Avro Row Library
Deserialization Step by Step

If de/serializer exists
Deserialization Step by Step

Return Dummy Serializer

Get Deserializer

SerDesRegistry

Generate Dummy Serializer

Put Dummy Serializer on cache

Return Dummy Serializer

Trigger Code Generation on Separate Thread
Deserialization Step by Step
Replacing Dummy Serializer

Get Deserializer

SerDesRegistry

Cache

Generate Serializer Java Code based on Schema

Compile Serializer Code and Create an Instance

Separate Thread

Replace Dummy Serializer with actual one on cache
Sample Generated Serializer
Repo Location

https://github.com/talatuyarer/beam-avro-row-serializer
What is Next?

- Publish also Serializer
- Create more document and benchmark tool
- Integrate our library with Beam master branch
- We need to improve schema evolution for BeamSQL pipelines
- Stop deserialize unused fields in sql statement
Questions?
Extra
How Avro Serialization Works

record Person {
    string userName;
    union { null, long } favouriteNumber;
    array<string> interests;
}

Avro

https://martin.kleppmann.com/2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html
Schema Evolution

- Beam SQL does not support schema changes
- This is painful if you have Select * style jobs.
- Currently only way is re-submitting stream jobs to re-generate their Beam SQL Java code with new schemas
- Luckily all events are written as Avro binary. Avro support some kind of schema evolution.
How we handle Schema Evolution

- Each job has their submitted version of Avro schema. We call Reader schema.
- Each Kafka message has Writer schema version as metadata on Kafka header.
- We convert all version of Avro events to Job’s version of Avro Generic Record and convert it to Row.
- Our schema updater check all jobs’ SQL queries if their sql has relevant fields with changes we update Job to update Beam SQL’s Java generated Code otherwise We don’t restart the job