New Avro Serialization for Beam SQL

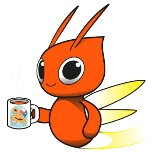
By Talat Uyarer

BEAM SUMMIT Austin, 2022

Agenda

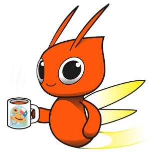
- Who am I ?
- Mission
- Benchmark Results
- Let’s Define Root Cause
- Our Solution
- Internals of Avro Row Library
- What is next ?
- Questions ?

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

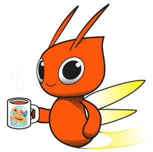
Beam SQL

How Beam SQL works

How Beam turn a String as runnable code

BEAM SUMMIT

How Beam SQL Works



Beam SQL (via Java)

```
SELECT key, SUM(value)
FROM input GROUP BY key
```

How Beam SQL Works

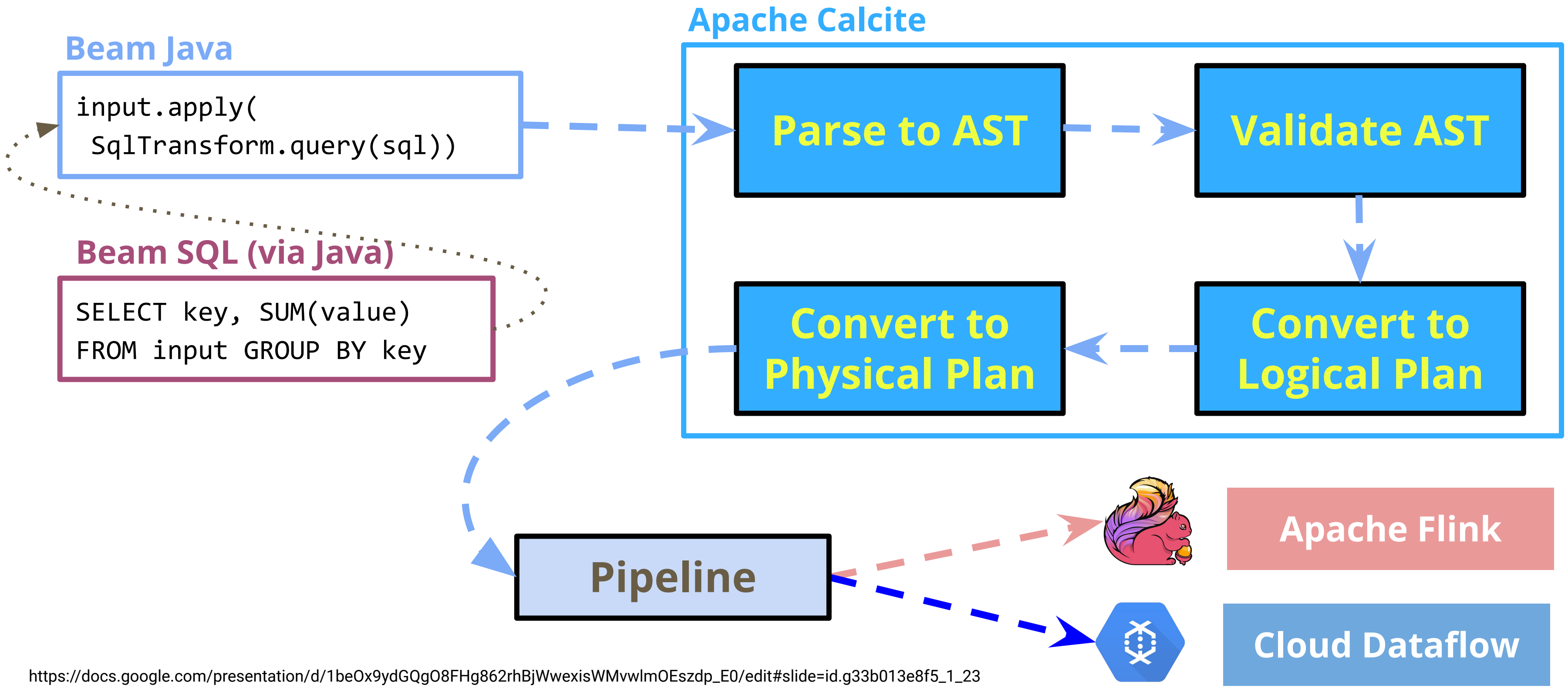
Beam Java

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

Beam SQL (via Java)

```
SELECT key, SUM(value)
FROM input GROUP BY key
```

How Beam SQL Works

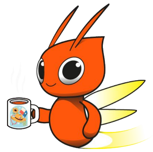


```
input.apply(
 SqlTransform.query(sql))
```

```
SELECT key, SUM(value)
FROM input GROUP BY key
```

https://docs.google.com/presentation/d/1beOx9ydGQgO8FHg862rhBjWwexisWMvwlmOEszdp_E0/edit#slide=id.g33b013e8f5_1_23

Sample Code Generation



- Generates Java code for SQL operators

```
SELECT id, convert(price), price * 10 WHERE item = “my item” ...
```

Becomes like as below. Accepts input Row format and write out in Row format.

```
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));
        }
}
```

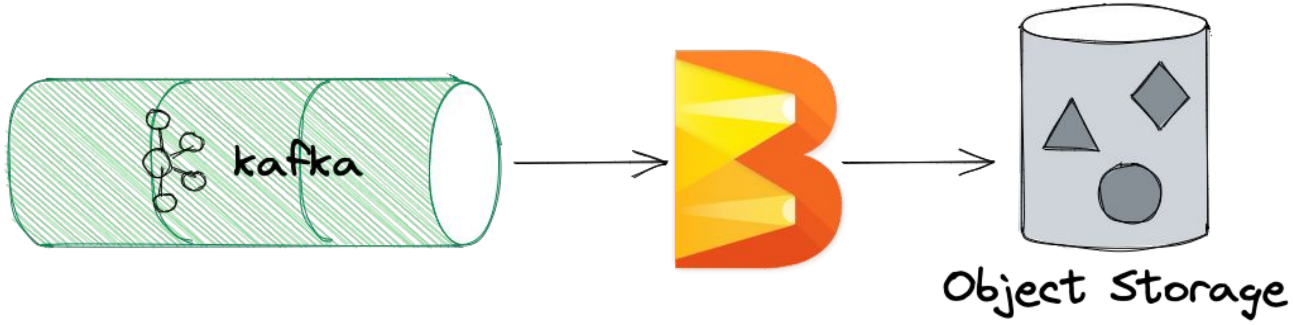
https://docs.google.com/presentation/d/1beOx9ydGQgO8FHg862rhBjWwexisWMvwlmOEszdp_E0/edit#slide=id.g33b013e8f5_2_18

Beam SQL

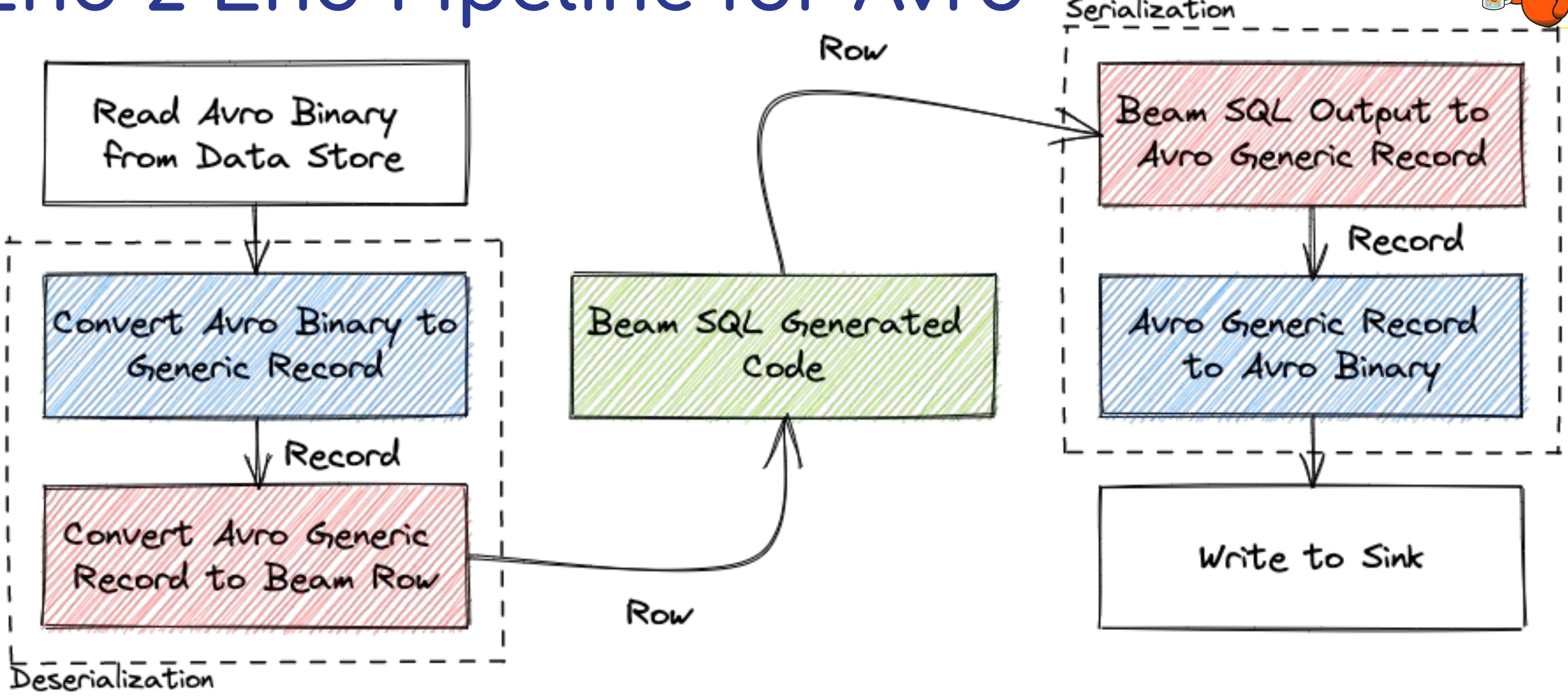
How Beam SQL Tables work

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

Sample Pipeline



End 2 End Pipeline for Avro


Read Avro Binary from Data Store
Convert Avro Binary to Generic Record
Record
Convert Avro Generic Record to Beam Row
Deserialization
Row
Beam SQL Generated Code
Row
Serialization
Beam SQL Output to Avro Generic Record
Record
Avro Generic Record to Avro Binary
Write to Sink

Beam SQL

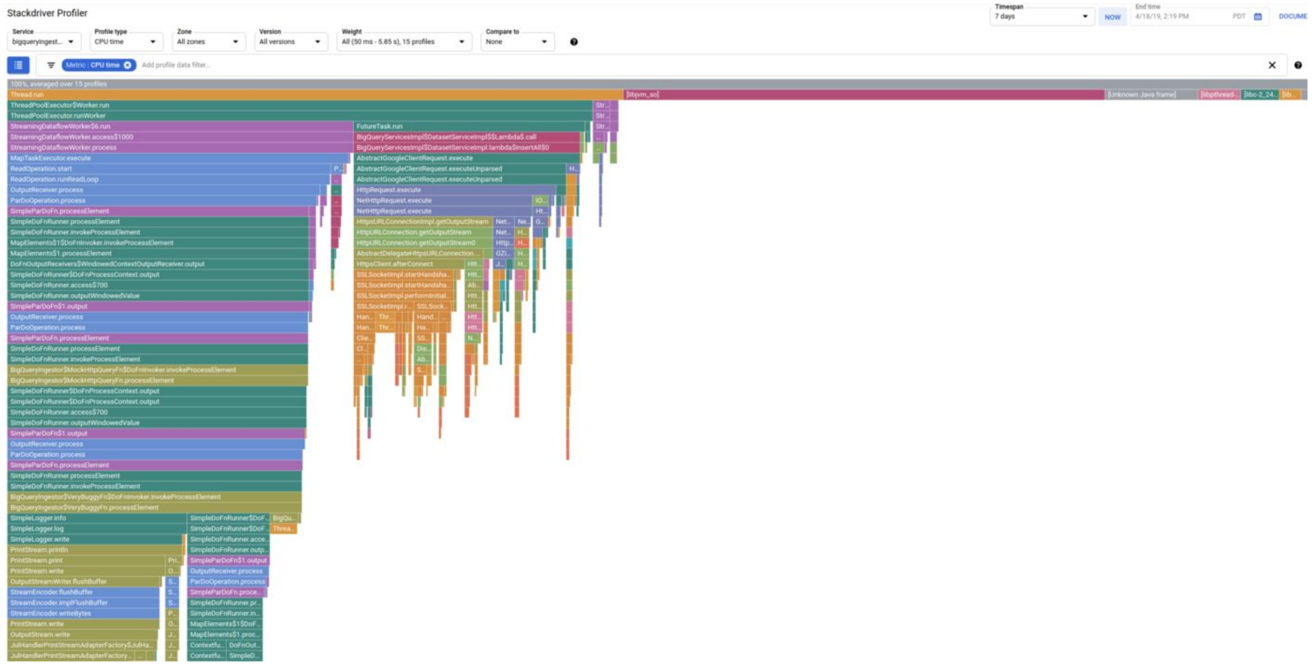
Performance Issue

We create two object for each Avro Record.

We waste our CPU cycle and memory

BEAM SUMMIT

Beam Pipeline Profiling



Stackdriver Profiler

Timespan: 7 days | NOW | End time: 4/18/19, 2:19 PM PDT | DOCUME

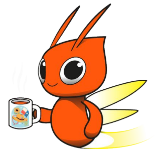
Service: bigqueryingest...
Profile type: CPU time
Zone: All zones
Version: All versions
Weight: All (50 ms - 5.85 s), 15 profiles
Compare to: None

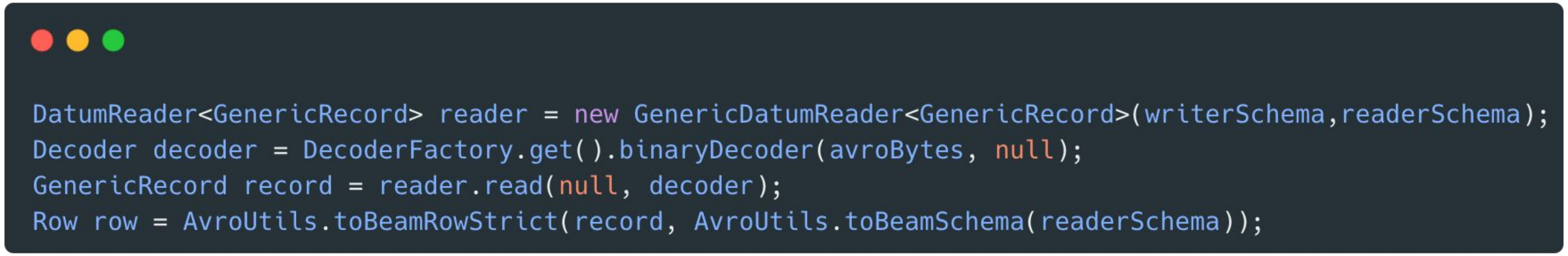
Metric : CPU time | Add profile data filter...

100%, averaged over 15 profiles
Thread.run
[libjvm_so]
[Unknown Java frame]
[libpthread-...
[libc-2_24...
ThreadPoolExecutor$Worker.run
ThreadPoolExecutor.runWorker
StreamingDataflowWorker$6.run
StreamingDataflowWorker.access$1000
StreamingDataflowWorker.process
MapTaskExecutor.execute
ReadOperation.start
ReadOperation.runReadLoop
OutputReceiver.process
ParDoOperation.process
SimpleParDoFn.processElement
SimpleDoFnRunner.processElement
SimpleDoFnRunner.invokeProcessElement
MapElements$1$DoFnInvoker.invokeProcessElement
MapElements$1.processElement
DoFnOutputReceivers$WindowedContextOutputReceiver.output
SimpleDoFnRunner$DoFnProcessContext.output
SimpleDoFnRunner.access$700
SimpleDoFnRunner.outputWindowedValue
SimpleParDoFn$1.output
OutputReceiver.process
ParDoOperation.process
SimpleParDoFn.processElement
SimpleDoFnRunner.processElement
SimpleDoFnRunner.invokeProcessElement
BigQueryIngestor$MockHttpQueryFn$DoFnInvoker.invokeProcessElement
BigQueryIngestor$MockHttpQueryFn.processElement
SimpleDoFnRunner$DoFnProcessContext.output
SimpleDoFnRunner$DoFnProcessContext.output
SimpleDoFnRunner.access$700
SimpleDoFnRunner.outputWindowedValue
SimpleParDoFn$1.output
OutputReceiver.process
ParDoOperation.process
SimpleParDoFn.processElement
SimpleDoFnRunner.processElement
SimpleDoFnRunner.invokeProcessElement
BigQueryIngestor$VeryBuggyFn$DoFnInvoker.invokeProcessElement
BigQueryIngestor$VeryBuggyFn.processElement
SimpleLogger.info
SimpleLogger.log
SimpleLogger.write
PrintStream.println
PrintStream.print
PrintStream.write
OutputStreamWriter.flushBuffer
StreamEncoder.flushBuffer
StreamEncoder.implFlushBuffer
StreamEncoder.writeBytes
PrintStream.write
OutputStream.write
JulHandlerPrintStreamAdapterFactory$JulHa...
JulHandlerPrintStreamAdapterFactory...
SimpleDoFnRunner$DoF...
SimpleDoFnRunner.acce...
SimpleDoFnRunner.outp...
SimpleParDoFn$1.output
OutputReceiver.process
ParDoOperation.process
SimpleParDoFn.proce...
SimpleDoFnRunner.pr...
SimpleDoFnRunner.in...
MapElements$1$DoF...
MapElements$1.proc...
Contextfu...
DoFnOut...
SimpleD...
BigQu...
Threa...
FutureTask.run
BigQueryServicesImpl$DatasetServiceImpl$$Lambda$.call
BigQueryServicesImpl$DatasetServiceImpl.lambda$insertAll$0
AbstractGoogleClientRequest.execute
AbstractGoogleClientRequest.executeUnparsed
AbstractGoogleClientRequest.executeUnparsed
HttpRequest.execute
NetHttpRequest.execute
NetHttpRequest.execute
HttpsURLConnectionImpl.getOutputStream
HttpURLConnection.getOutputStream
HttpURLConnection.getOutputStream0
AbstractDelegateHttpsURLConnection...
HttpsClient.afterConnect
SSLSocketImpl.startHandsha...
SSLSocketImpl.startHandsha...
SSLSocketImpl.performInitial...
SSLSocketImpl.r...
SSLSock...

https://medium.com/google-cloud/profiling-dataflow-pipelines-ddbbef07761d

How we can improve this ?

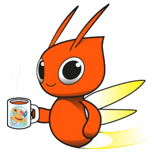




```java
DatumReader<GenericRecord> reader = new GenericDatumReader<GenericRecord>(writerSchema,readerSchema);
Decoder decoder = DecoderFactory.get().binaryDecoder(avroBytes, null);
GenericRecord record = reader.read(null, decoder);
Row row = AvroUtils.toBeamRowStrict(record, AvroUtils.toBeamSchema(readerSchema));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

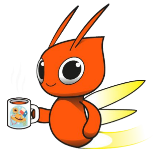
Our Solution

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

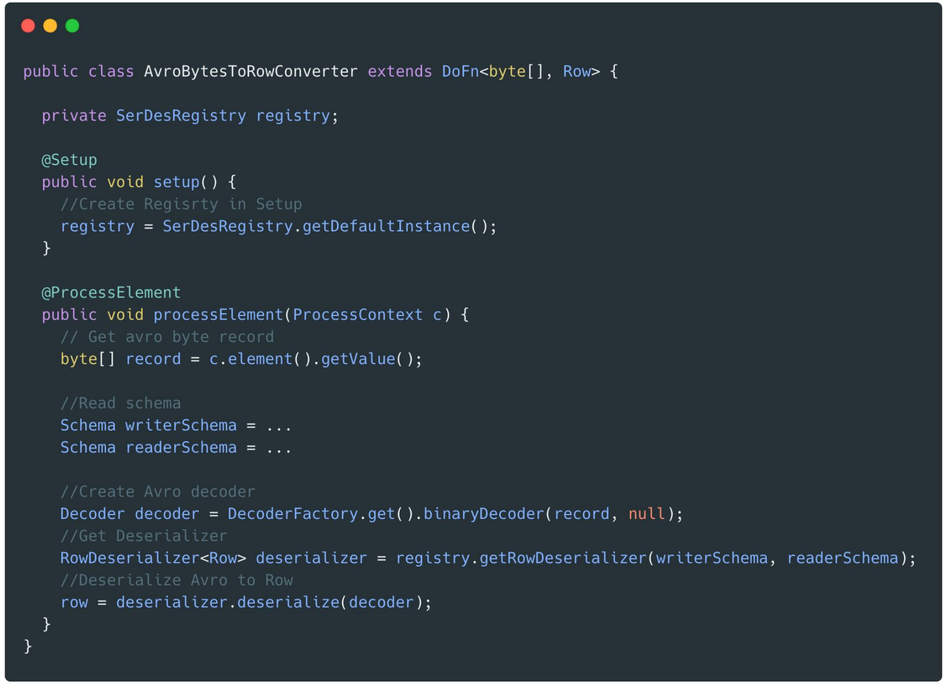
Our Solution - Code Generation



- 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/

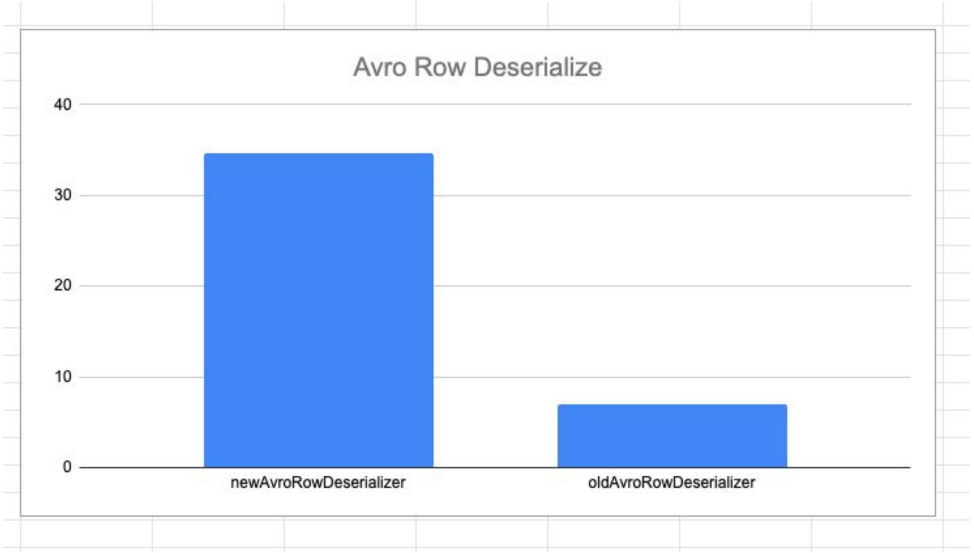
Usage of AvroRow Library



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

  private SerDesRegistry registry;

  @Setup
  public void setup() {
    //Create Regisrty 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

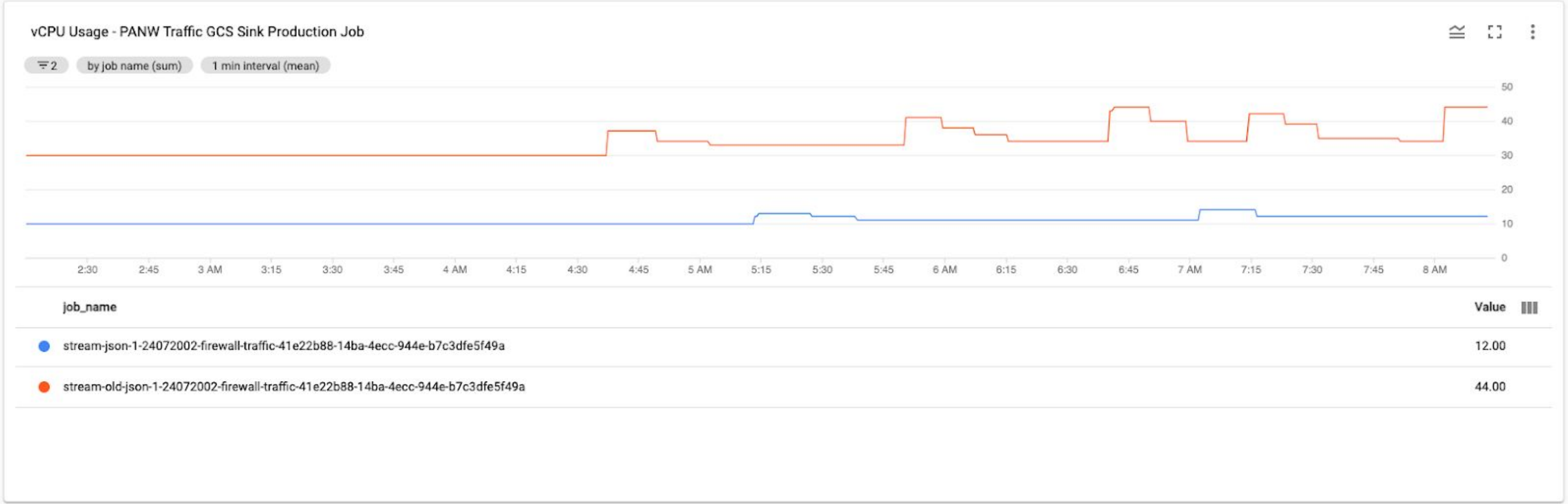
Based on Throughput wise



- After start using new library. We reached ~ 5x throughtput 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

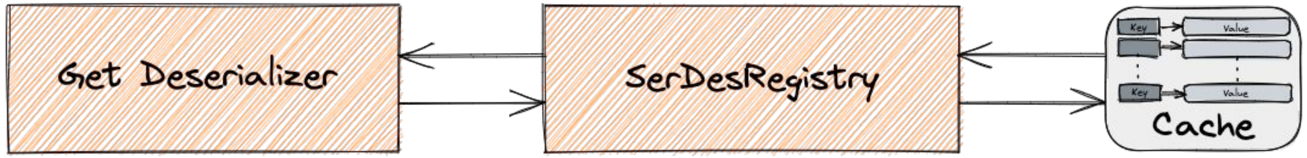


	job_name	Value
●	stream-json-1-24072002-firewall-traffic-41e22b88-14ba-4ecc-944e-b7c3dfe5f49a	12.00
●	stream-old-json-1-24072002-firewall-traffic-41e22b88-14ba-4ecc-944e-b7c3dfe5f49a	44.00

Internal Of Avro Row Library

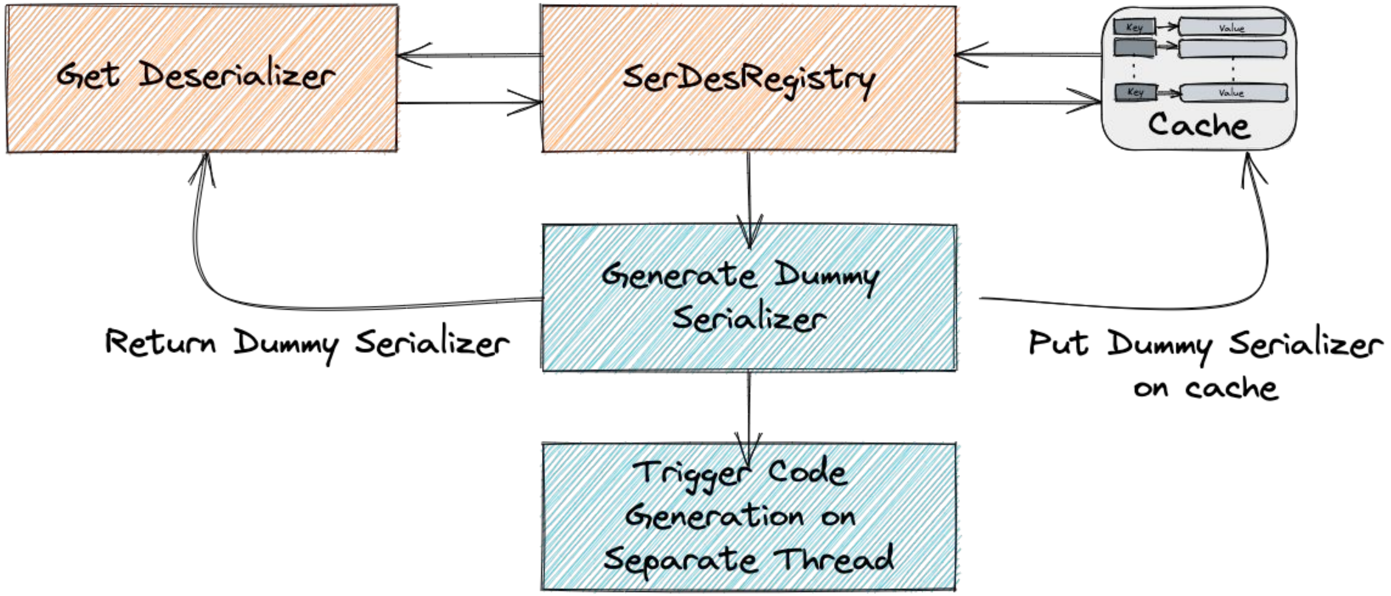
Deserialization Step by Step

If de/serializer exists



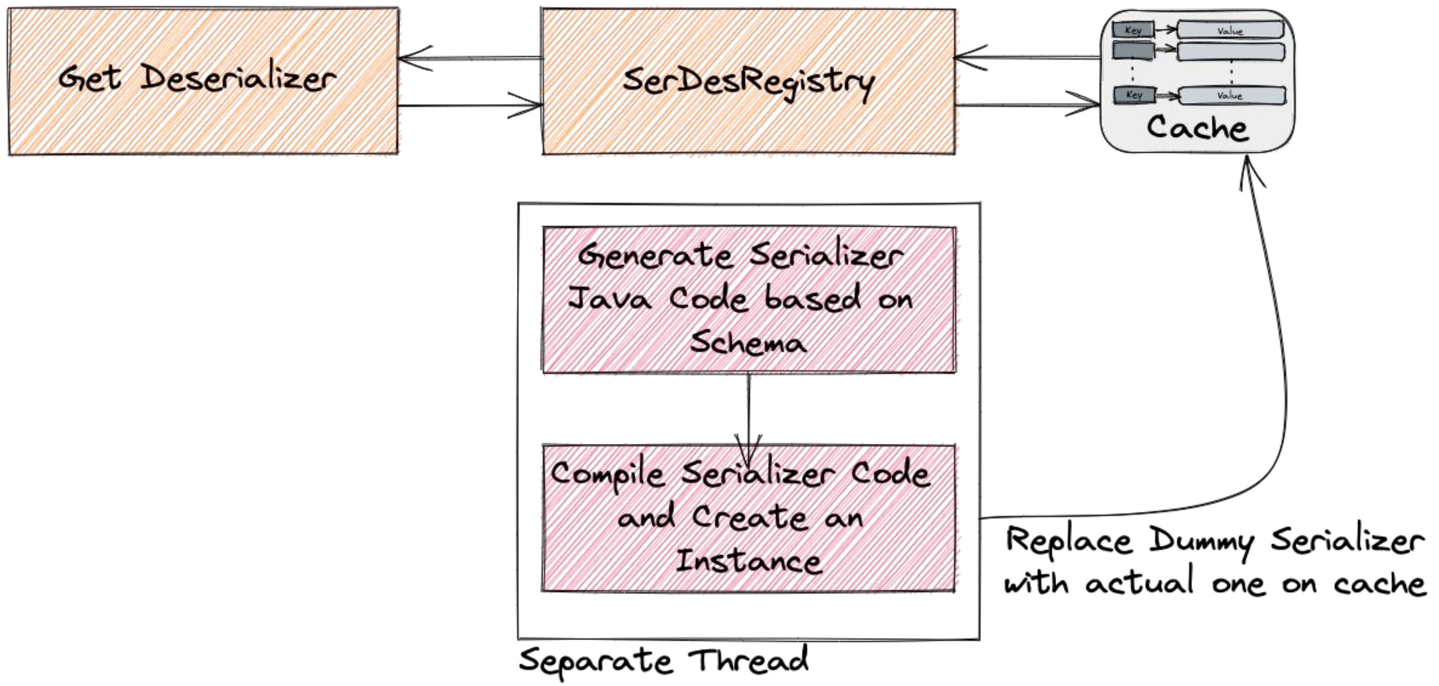
Deserialization Step by Step

Return Dummy Serializer

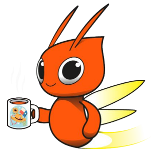

Get Deserializer
SerDesRegistry
Key
Value
Key
Value
Cache
Generate Dummy Serializer
Return Dummy Serializer
Put Dummy Serializer on cache
Trigger Code Generation on Separate Thread

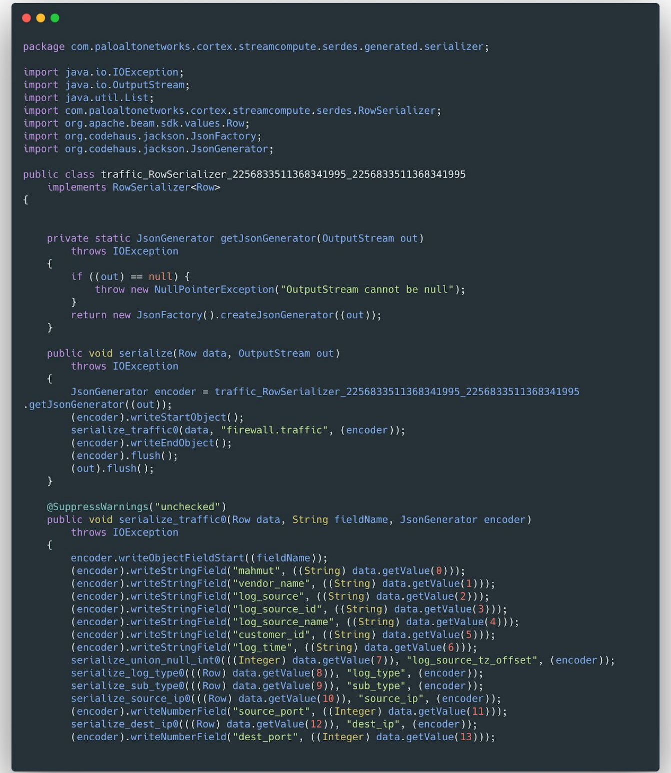
Deserialization Step by Step

Replacing Dummy Serializer


Get Deserializer
SerDesRegistry
Key
Value
Key
Value
Cache
Generate Serializer Java Code based on Schema
Compile Serializer Code and Create an Instance
Replace Dummy Serializer with actual one on cache
Separate Thread

Sample Generated Serializer





```java
package com.paloaltonetworks.cortex.streamcompute.serdes.generated.serializer;

import java.io.IOException;
import java.io.OutputStream;
import java.util.List;
import com.paloaltonetworks.cortex.streamcompute.serdes.RowSerializer;
import org.apache.beam.sdk.values.Row;
import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonGenerator;

public class traffic_RowSerializer_2256833511368341995_2256833511368341995
    implements RowSerializer<Row>
{


    private static JsonGenerator getJsonGenerator(OutputStream out)
        throws IOException
    {
        if ((out) == null) {
            throw new NullPointerException("OutputStream cannot be null");
        }
        return new JsonFactory().createJsonGenerator((out));
    }

    public void serialize(Row data, OutputStream out)
        throws IOException
    {
        JsonGenerator encoder = traffic_RowSerializer_2256833511368341995_2256833511368341995
.getJsonGenerator((out));
        (encoder).writeStartObject();
        serialize_traffic0(data, "firewall.traffic", (encoder));
        (encoder).writeEndObject();
        (encoder).flush();
        (out).flush();
    }

    @SuppressWarnings("unchecked")
    public void serialize_traffic0(Row data, String fieldName, JsonGenerator encoder)
        throws IOException
    {
        encoder.writeObjectFieldStart((fieldName));
        (encoder).writeStringField("mahmut", ((String) data.getValue(0)));
        (encoder).writeStringField("vendor_name", ((String) data.getValue(1)));
        (encoder).writeStringField("log_source", ((String) data.getValue(2)));
        (encoder).writeStringField("log_source_id", ((String) data.getValue(3)));
        (encoder).writeStringField("log_source_name", ((String) data.getValue(4)));
        (encoder).writeStringField("customer_id", ((String) data.getValue(5)));
        (encoder).writeStringField("log_time", ((String) data.getValue(6)));
        serialize_union_null_int0(((Integer) data.getValue(7)), "log_source_tz_offset", (encoder));
        serialize_log_type0(((Row) data.getValue(8)), "log_type", (encoder));
        serialize_sub_type0(((Row) data.getValue(9)), "sub_type", (encoder));
        serialize_source_ip0(((Row) data.getValue(10)), "source_ip", (encoder));
        (encoder).writeNumberField("source_port", ((Integer) data.getValue(11)));
        serialize_dest_ip0(((Row) data.getValue(12)), "dest_ip", (encoder));
        (encoder).writeNumberField("dest_port", ((Integer) data.getValue(13)));
```

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?

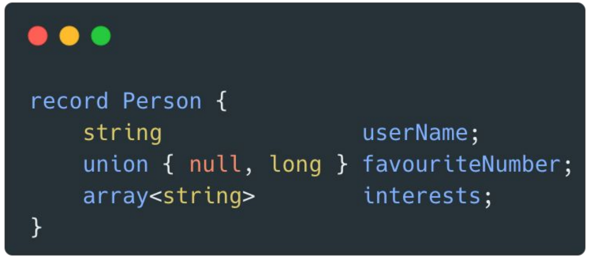
Send me an Email
talat@apache.org

Find me on Twitter
@talatuyarer

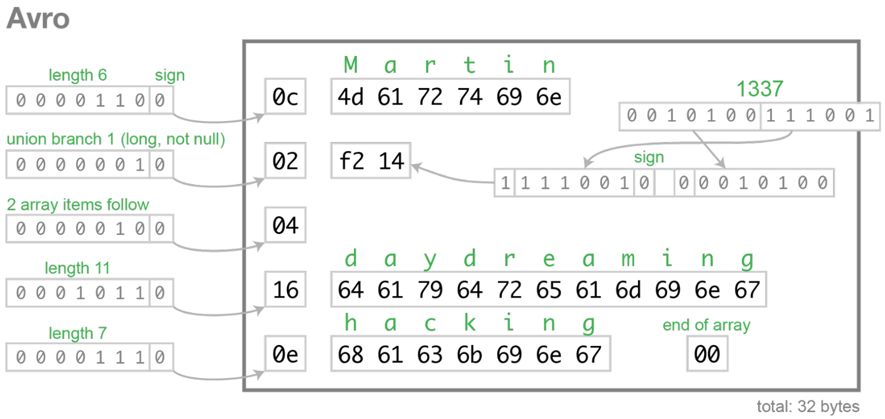
BEAM SUMMIT Austin, 2022

Extra

How Avro Serialization Works

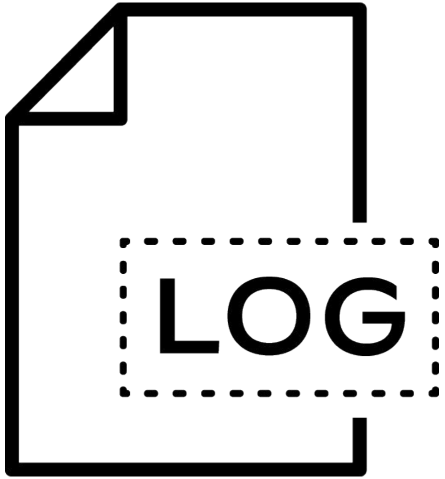


```
record Person {
    string                 userName;
    union { null, long } favouriteNumber;
    array<string>          interests;
}
```



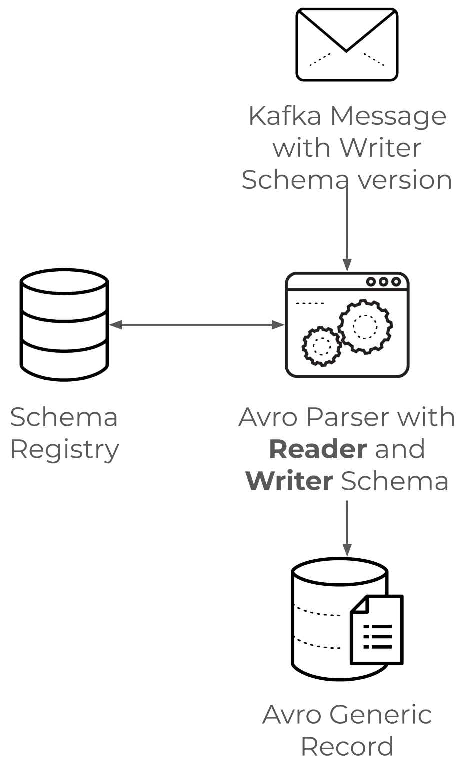
https://martin.kleppmann.com/2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html

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


Kafka Message with Writer Schema version
Schema Registry
Avro Parser with **Reader** and **Writer** Schema
Avro Generic Record

- 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