Easy cross-language with SchemaTransforms: use your favorite Java transform in Python SDK

Ahmed Abualsaud
About me

[Images and icons representing flags, a character, and digital tools]
Agenda

- Background and problem statement
  - Why cross-language is important for the future of Beam
- What is a SchemaTransform?
  - Why is it called a SchemaTransform?
- Creating a SchemaTransform
- Running a Java expansion service
- Demo: Using Java’s wordcount transform in a Python pipeline
- Current limitations/unknowns
Problem statement

2016
Beam 0.1.0
Problem statement

- 2016: Beam 0.1.0
- 2017: Beam 0.6.0
- 2021: Beam 2.33.0
2016  Beam 0.1.0
2017  Beam 0.6.0
2021  Beam 2.33.0
2022  Beam 2.40.0 (experimental)
Problem statement

- 2016: Beam 0.1.0
- 2017: Beam 0.6.0
- 2021: Beam 2.33.0
- 2022: Beam 2.40.0 (experimental)
Enter cross-language

1. Start up expansion service
2. Create and inject transform into the pipeline
3. Run multi-language pipeline

Pipeline SDK (Python)

Cross-language transform SDK (Java)

Beam Runner

Workers

SDK Harnesses

Kafka
What is a SchemaTransform?

Ahmed Abualsaud
What is a SchemaTransform?

- SchemaTransformProvider
- Unique Identifier
- Input PCollection(s)
- Output PCollection(s)
- Configuration schema
What is a SchemaTransform? 

- configuration
- SchemaTransformProvider
- from()
- Unique Identifier
- Input PCollection(s)
- Output PCollection(s)
- Configuration schema
What is a SchemaTransform?

SchemaTransformProvider

from()

SchemaTransform

- Unique Identifier
- Input PCollection(s)
- Output PCollection(s)
- Configuration schema
What is a SchemaTransform?
What is a SchemaTransform?

- Configuration
- SchemaTransformProvider
  - from()
- SchemaTransform
  - buildTransform()
- PTransform
  - PCollection<Row>
  - PCollection<Row>
- Input PCollection(s)
- Output PCollection(s)
- Configuration schema
- Unique Identifier
Why is it called a SchemaTransform?

PCollection<Row>

PCollection<Row>

PCollection<Row>

PCollection<Row>

PCollection<Row>

PCollection<Row>

PTransform
Why is it called a SchemaTransform?

```
PCollection<Row> → Schema in1
PCollection<Row>
PCollection<Row>
PCollection<Row>

PTransform
```

PCollection<Row>
PCollection<Row>
PCollection<Row>
PCollection<Row>
Why is it called a SchemaTransform?

PCollection<Row> → Schema in1
PCollection<Row> → Schema in2

PCollection<Row>

PTransform

PCollection<Row>
PCollection<Row>
Why is it called a SchemaTransform?

PCollection<Row> → Schema in1
PCollection<Row> → Schema in2
PCollection<Row> → Schema inN

PTransform

PCollection<Row> → Schema out1
PCollection<Row> → Schema out2
PCollection<Row> → Schema outN
Creating a SchemaTransform

```java
public interface SchemaTransformProvider {
    String identifier();
}
```

- **SchemaTransformProvider**
  - from()
  - Unique Identifier
  - Input PCollection(s)
  - Output PCollection(s)
  - Configuration schema

```java
public interface SchemaTransformProvider {
    String identifier();
    Schema configurationSchema();
    SchemaTransform from(Row configuration);
    List<String> inputCollectionNames();
    List<String> outputCollectionNames();
    Optional<List<String>> dependencies(Row configuration, PipelineOptions options) {
        return Optional.empty();
    }
}
```
public interface SchemaTransform {
    PTransform<PCollectionRowTuple, PCollectionRowTuple> buildTransform();
}
public interface SchemaTransformProvider {
  SchemaTransform from(Row configuration);
}

public interface SchemaTransform {
  PTransform<PCollection<RowTuple>, PCollection<RowTuple>> buildTransform();
}
for (org.apache.beam.sdk.schemas.transforms.SchemaTransformProvider schemaTransformProvider : ServiceLoader.load(
    org.apache.beam.sdk.schemas.transforms.SchemaTransformProvider.class)) {

import com.google.auto.service.AutoService;

@AutoService(SchemaTransformProvider.class)
public class MySchemaTransformProvider implements SchemaTransformProvider {

Running a Java expansion service

**Jar containing Beam's ExpansionService**
```
java -cp beam-sdks-java-io-expansion-service-2.47.0.jar:my-project.jar org.apache.beam.sdk.expansion.service.ExpansionService 12345
```

**Jar containing our SchemaTransform(s)**

**Port the expansion service will run on**

```
Registered SchemaTransformProviders:
  my_java_transform
  beam:schematransform:org.apache.beam:kafka_read:v1
  beam:schematransform:org.apache.beam:kafka_write:v1
```
Using the SchemaTransform in a Python Pipeline

```python
with beam.Pipeline() as p:
    _ = (
        p
            | beam.Create([beam.Row(text="Hello"), beam.Row(text="World!")])
            | SchemaAwareExternalTransform(
                identifier="my_java_transform",  # SchemaTransform URN
                expansion_service="localhost:12345",  # expansion service address
                arg1="string",  # configuration parameter
                arg2=1,  # configuration parameter
                arg3=False,  # configuration parameter
                rearrange_based_on_discovery=True  # set to true if params may be out of order
            )
            | beam.ParDo(ProcessRows()))
```
Demo!

github.com/ahmedabu98/xlang-word-count
- We are restricted to using Beam Rows
  - E.g. sending Strings is okay for cross-language interface
- Not all logical types are supported yet.
  - E.g. Java Date Time → Python Timestamp. No Python Date Time equivalent
- It still takes some time/effort to create a SchemaTransform
- We don’t have performance metrics for most SchemaTransforms
QUESTIONS?

linkedin.com/in/ahmedabu98
github.com/ahmedbu98