BEAM
SUMMIT

Cross-Language JdbcIO Enabled By Beam Portable Schemas

Yi Hu
Google
GitHub: @Abacn

BEAM SUMMIT
NYC 2023

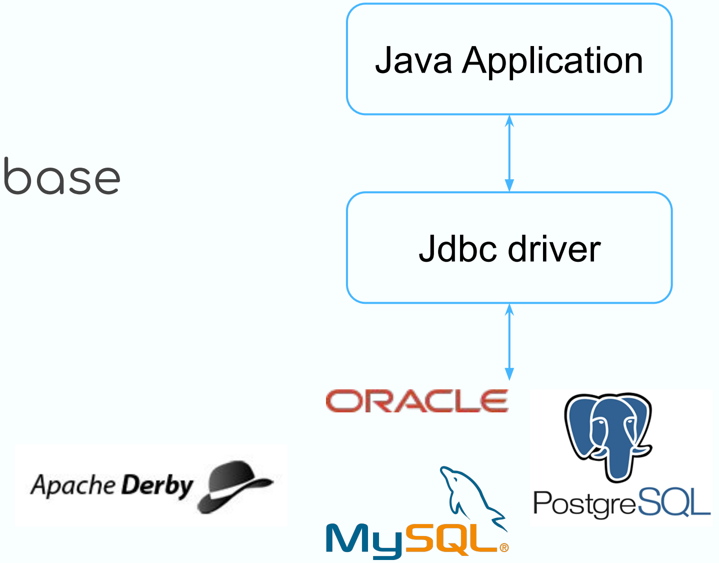
Agenda

- JdbcIO
  - Java Database Connectivity
  - Beam's JdbcIO
- Portable Beam Schemas
  - Types across language
  - Schema translation
  - Types and coders
  - Portable logical types
- Python xlang JdbcIO
  - use cases
  - run on different runners
  - deal with unsupported types

JDBC

Java Database connectivity (JDBC) is an API defining how a client may access a database for Java.

- Standardized API (java.sql.*, javax.sql.*)
- JDBC driver (class) interact with a database
- Relational databases



JdbcIO in Java SDK

One of the earliest IO connector in Beam ([BEAM-244], 0.3.0-incubating)

In Java SDK org.apache.beam.io.jdbc

- **JdbcIO.read()** read from JDBC datasource
- **JdbcIO.write()** write to JDBC
- (since v2.32) **JdbcIO.readWithPartitions()** Parallel reading from a JDBC datasource

JdbcIO as xlang connectors

In Python SDK apache_beam.io.jdbc since v2.24 (BEAM-10135, BEAM-10136)

- ReadFromJdbc
- WriteToJdbc

In Go SDK .../beam/sdks/v2/go/pkg/beam/io/xlang/jdbcio/jdbc since v2.37 (Beam-13293)

- Read
- Write

Problem: handling types and data xlang

Beam SDK prefers to follow the convention of that language

PCollection<user_type> -> PCollection[?]

	Java	Python	Go
Integers	int (4), long (8)	int (infinity precision), np.int32, np.int64, ...	int, int8, ..., int64
floating point values	float (4), double (8)	double (8), np.float128	float32, float64
Timestamp	java.time.Instant, org.joda.time.Instant	datetime.datetime	Timestamp
fixed precision numeric	java.math.BigDecimal	decimal.Decimal	decimal.Decimal

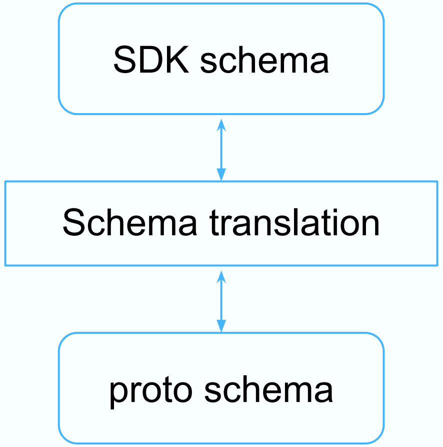
How to handle types cross language?

Portable beam schemas

https://s.apache.org/beam-schemas

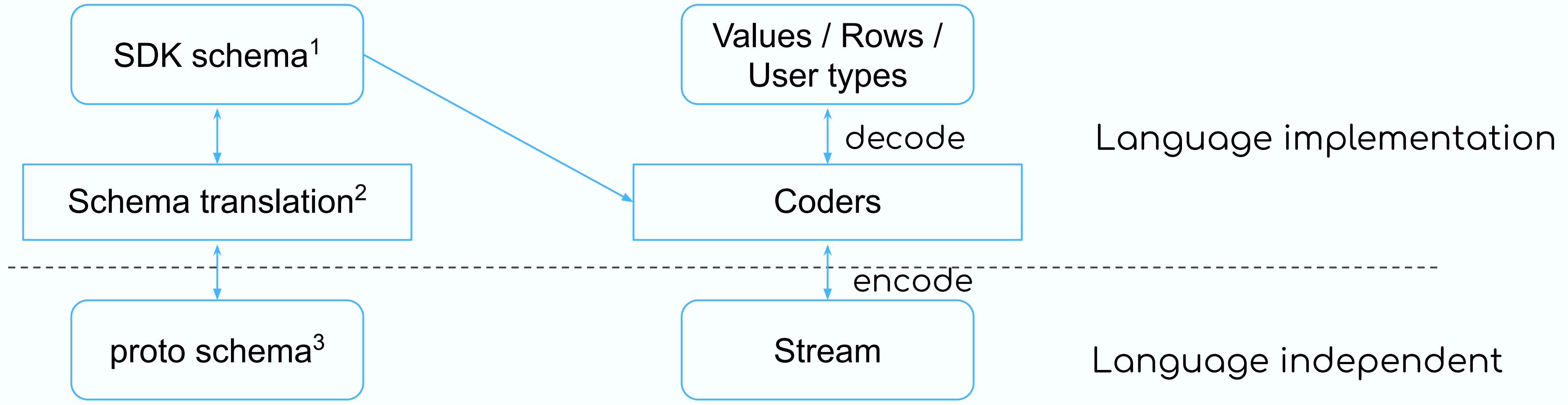
“ Beam schemas are a new and enhanced type system for Beam, making element structure explicit to support new concise transforms, relational-style optimization and execution, columnar optimization, and automatic type coercions. ”

Provide cross-language support in every Beam SDK.



Portable beam schemas

Schemas define types, coders pack/unpack values



Types and Coders: support status

Primitive types

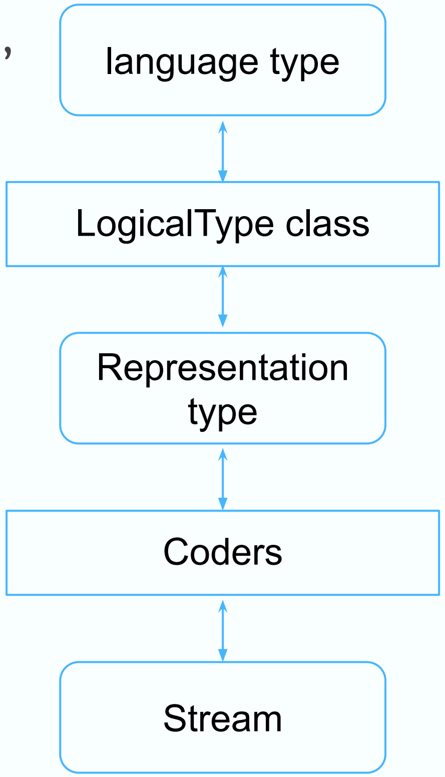
https://s.apache.org/beam-schemas

Type	BYTE	INT16	INT32	INT64	FLOAT	DOUBLE	STRING	BOOLEAN	BYTES
Java	byte	short	int	long	float	double	String	bool	byte[]
Python	np.int8	np.int16	np.int32	np.int64, int	np.float32	np.float64, float	str	bool	bytes
Go	int8	int16	int32	int64	float32	float64	string	bool	[]byte
Java	ByteCoder	BigEndianShortCoder	VarIntCoder	VarLongCoder	FloatCoder	DoubleCoder	StringUtf8Coder	BooleanCoder	ByteArrayCoder
Python	BytesCoder	BigEndianShortCoder (2.46+)	VarIntCoder	VarIntCoder	SinglePrecisionFloatCoder (2.42+)	FloatCoder	StrUtf8Coder	BooleanCoder	BytesCoder
Go	***currently (2.48) cast to int64 then encode*			coder/varint	coder/float (2.42+)	coder/double	coder/stringutf8	coder/bool	coder/bytes

Logical types in Beam

How to handle SQL types VARCHAR(size), BINARY(size), DECIMAL(size, digits), TIMESTAMP

- Non-portable logical types (Java SDK)
  - Without a URN. Identifier like "VARCHAR", "VARBINARY"
  - Not recognized cross-lang (Error beam:logical_type:javasdk:v1) (Q, #19817, #23526)
- Portable logical types
  - identifier is a URN beam:logical_type:xxx:v1
  - A uniform representation type, and a language type per SDK
  - Arguments

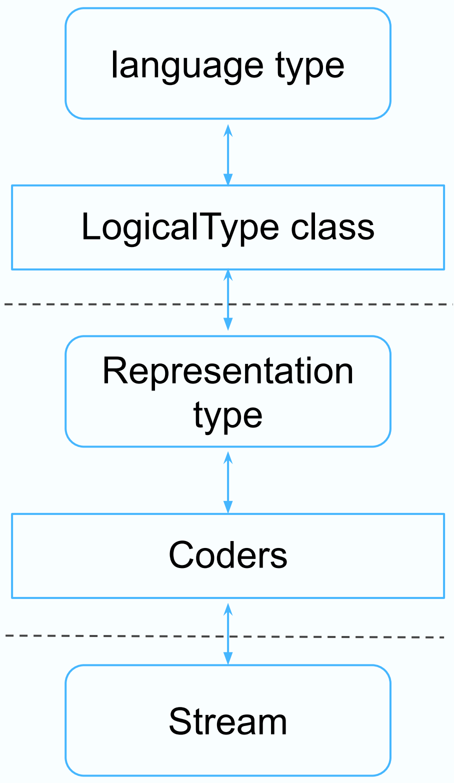


Beam portable logical types

Example:

micros_instant

Java SDK	Python SDK
java.time.Instant	beam.utils.timestamp.Timestamp
org.apache.beam.sdk.schemas.logicaltypes.MicrosInstant	beam.typehints.schemas.MicrosInstant
	Row(INT64, INT32)
	RowCoder



Standard logical types

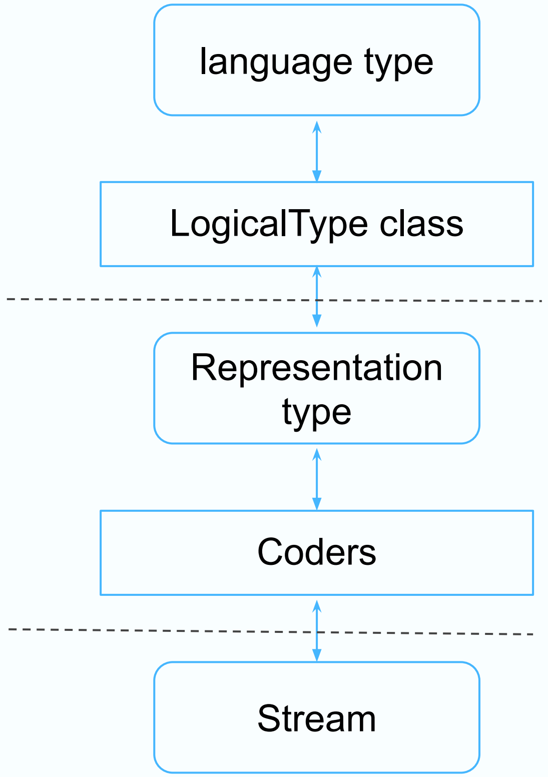
- URN defined in proto, supposed to be understood by all SDKs.
- Each SDK's LogicalType implementation defines language type and conversion rules

Portable logical types: support status

Example:

micros_instant

Type	Java/Python	Go
micros_instant	2.33	N/A as of 2.48
millis_instant	2.42	
decimal	2.43	
fixed_bytes	2.44	
var_bytes	2.44	
fixed_char	2.44	
var_char	2.44	



Standard logical types

- URN defined in proto, supposed to be understood by all SDKs.
- Each SDK's LogicalType implementation defines language type and conversion rules

Use case: xlang JdbcIO

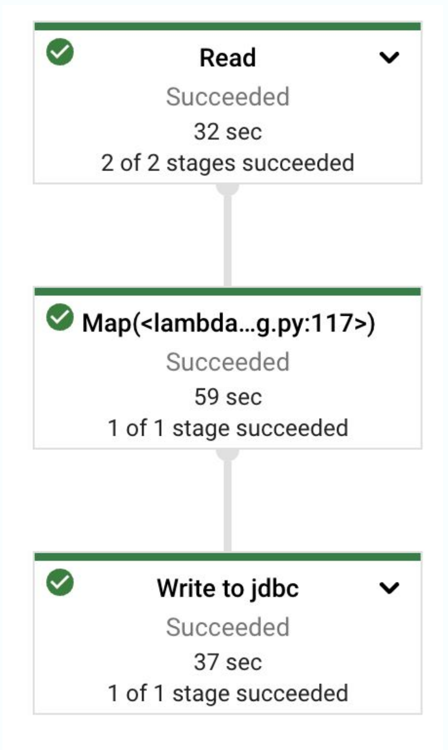
SQL schema / Python NamedTuple

```
(f_id INTEGER,
f_real FLOAT,
f_fixedchar VARCHAR(12),
f_varchar CHAR(12),
f_bin bytea,
f_timestamp Timestamp,
f_decimal DECIMAL(10,2))
```

```
JdbcWriteTestRow = typing.NamedTuple(
 "JdbcWriteTestRow",
 [
     ("f_id", int),
     ("f_real", float),
     ("f_fixedchar", str),
     ("f_varchar", str),
     ("f_bin", bytes),
     ("f_timestamp", Timestamp),
     ("f_decimal", Decimal)],)
```

Write

```
coders.registry.register_coder(JdbcWriteTestRow, coders.RowCoder)
with Pipeline(options=options) as p:
    input = p | SyntheticSource(...) | MapToRow(...)
    input | WriteToJdbc(
        table_name=self.table_name,
        driver_class_name=self.driver,
        jdbc_url=self.jdbc_url.replace('localhost',
'host.docker.internal'),
        username=self.username,
        password=self.password))
```


Read
Succeeded
32 sec
2 of 2 stages succeeded

Map(<lambda...g.py:117>)
Succeeded
59 sec
1 of 1 stage succeeded

Write to jdbc
Succeeded
37 sec
1 of 1 stage succeeded

Code available at https://gist.github.com/Abacn/3fa72fab4b0bbf5e3de395106ef47cfb

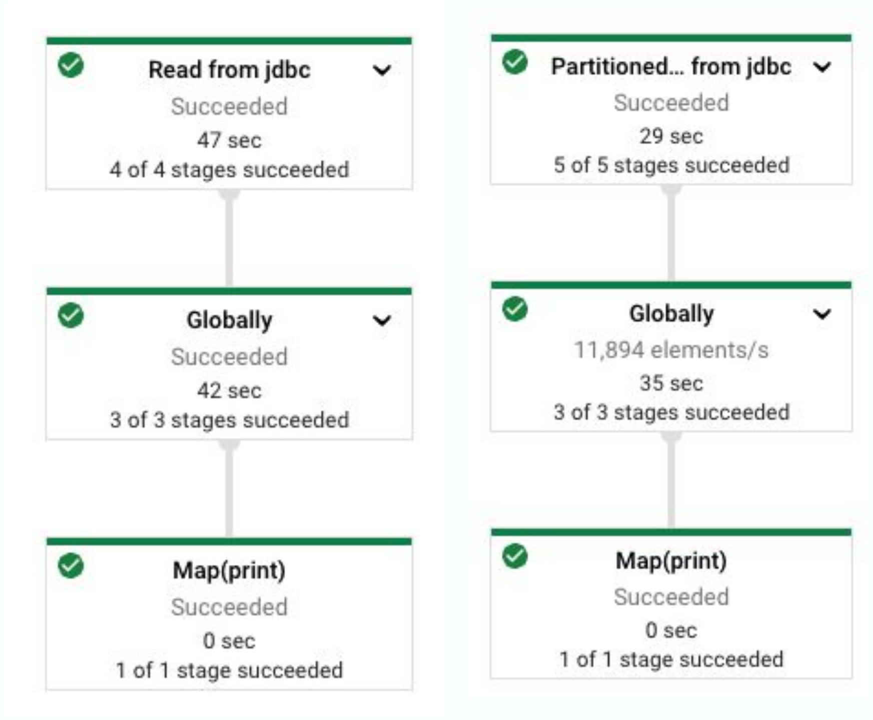
Use case: xlang JdbcIO

Read

```
LogicalType.register_logical_type(MillisInstant)
with Pipeline(options=options) as p:
    output = (p | 'Read from jdbc' >> ReadFromJdbc(
        table_name=self.table_name,
        driver_class_name=self.driver,
        jdbc_url=jdbc_url,
        username=username,
        password=password))
```

Partitioned read (since v2.46)

```
LogicalType.register_logical_type(MillisInstant)
with Pipeline(options=options) as p:
    input = (p | 'Read from jdbc' >> ReadFromJdbc(
        table_name=self.table_name,
        driver_class_name=self.driver,
        jdbc_url=jdbc_url,
        username=username,
        password=password,
        partition_column='f_id',
        partitions=100))
```



Read from jdbc
Succeeded
47 sec
4 of 4 stages succeeded

Globally
Succeeded
42 sec
3 of 3 stages succeeded

Map(print)
Succeeded
0 sec
1 of 1 stage succeeded

Partitioned... from jdbc
Succeeded
29 sec
5 of 5 stages succeeded

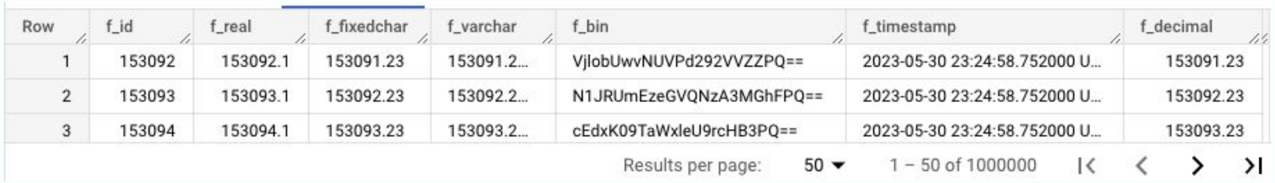
Globally
11,894 elements/s
35 sec
3 of 3 stages succeeded

Map(print)
Succeeded
0 sec
1 of 1 stage succeeded

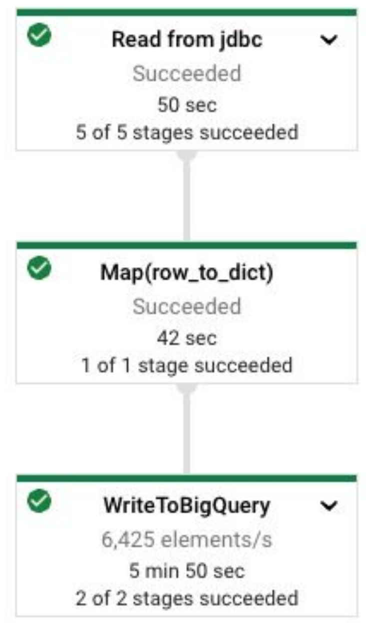
Use case: xlang JdbcIO

Read from Jdbc and write to BigQuery (storage write API, since 2.47.0)

```
LogicalType.register_logical_type(MillisInstant)
with Pipeline(options=options) as p:
    input = (p | 'Read from jdbc' >> ReadFromJdbc(...)
    _ = (input
         | beam.Map(lambda r: r.as_dict())
         | WriteToBigQuery(
           table=table,
           method=WriteToBigQuery.Method.STORAGE_WRITE_API,
           schema=SCHEMA))
```



Row	f_id	f_real	f_fixedchar	f_varchar	f_bin	f_timestamp	f_decimal
1	153092	153092.1	153091.23	153091.2...	VjlobUwvNUVPd292VVZZPQ==	2023-05-30 23:24:58.752000 U...	153091.23
2	153093	153093.1	153092.23	153092.2...	N1JRUmEzeGVQNzA3MGhFPQ==	2023-05-30 23:24:58.752000 U...	153092.23
3	153094	153094.1	153093.23	153093.2...	cEdxK09TaWxleU9rcHB3PQ==	2023-05-30 23:24:58.752000 U...	153093.23



Read from jdbc
Succeeded
50 sec
5 of 5 stages succeeded

Map(row_to_dict)
Succeeded
42 sec
1 of 1 stage succeeded

WriteToBigQuery
6,425 elements/s
5 min 50 sec
2 of 2 stages succeeded

Run everywhere

What needed to make pipeline run?

- For all runners
  - Python environment with Beam installed
  - An Java environment - needs for expansion service
- Direct runner
  - Additionally, a docker environment - run container image for other SDK
- Portable runner (Spark/Flink/...)
  - docker environment also needed to run job server

* For released beam versions, expansion service jar are automatically downloaded first-time run. Containers are also pulled automatically

What if the logical type not supported?

- Solution 1: Implement the logical type in place
- Solution 2: cast to string
    - Example: Schema with DATE and TIME field
    - Gives non-portable DATE and TIME logical type in Java SDK
      ValueError: No logical type registered for URN 'beam:logical_type:javasdk:v1'

```
rows = (p | 'Read from jdbc' >> ReadFromJdbc(
      query=f"select f_id, CAST(f_date as TEXT), CAST(f_time as TEXT), f_timestamp from {table_name}",
      table_name=table_name,
      driver_class_name=self.driver,
      jdbc_url=self.jdbc_url.replace('localhost', 'host.docker.internal'),
      username=self.username,
      password=self.password))
```

Yi Hu

QUESTIONS?

Github: @Abacn

BEAM SUMMIT

NYC 2023