Data Council Austin / March 25, 2024

How Arroyo Tricks DataFusion Into Stream Processing

Jackson Newhouse
co-founder, Arroyo

arroyo

What is Arroyo?

- A new(ish) Rust-based stream processing engine
- Apache 2.0/MIT Dual Licensed
- Read from sources and write to sinks (Kafka, S3, Webhooks, Redis…)
- Supports stateful computations (Event Windows, Joins, Aggregates)
- Uses Chandy–Lamport algorithm to take frequent, asynchronous checkpoints
- SQL frontend provided by DataFusion, extended for streaming semantics
- Support for Rust UDFs using RecordBatches and dynamic linking

What We Changed/Extended For DataFusion

- Custom CREATE TABLE
- Window-Based Synthetic UDFs
- Inserting event-time semantics
- LogicalPlan Rewriting
- Swappable RecordBatch Inputs
- Changes to ExecutionPlans
- Checkpointing With DataFusion
- Rust UDFs via Dynamic Linking + Arrow Arrays

arroyo

Custom CREATE TABLE Statements

```
CREATE TABLE impulse_source (
      timestamp TIMESTAMP,
      counter BIGINT UNSIGNED NOT NULL,
      subtask_index BIGINT UNSIGNED NOT NULL
    ) WITH (
      connector = 'kafka',
      topic = 'impulse',
      format = 'json',
      type = 'source',
      bootstrap_servers = 'localhost:9092',
      event_time_field = 'timestamp');
```

arroyo

Window-based Synthetic UDFs

Add window UDFs that will be converted into custom operators

```
tumble(interval ‘1 second) as window
hop(interval ‘1 minute’, interval ‘1 hour) as window
session(interval ‘5 minute’) as window
```

arroyo

Inserting event-time semantics

Arroyo tracks progress through event-time, so Arroyo operators need to be modified to pass through event-time when they emit data.

After much trial and error settled on adding a _timestamp field throughout the LogicalPlan as part of ArroyoRewriter.

Logical Plan Rewriter

Rewrite all SQL plans that require custom behavior, e.g. LogicalPlan::Join, LogicalPlan::Aggregate, LogicalPlan::TableScan, LogicalPlan::WindowFunction.

Rewrites produce a LogicalPlan::Extension(extension) where the extension also implements ArroyoExtension:

```
pub(crate) trait ArroyoExtension {

     fn node_name(&self) -> Option<NamedNode>;

     fn plan_node(&self, planner: &Planner, index: usize,
      input_schemas: Vec<ArroyoSchemaRef>) -> Result<Node>;

     fn output_schema(&self) -> ArroyoSchema;

}
```

Swappable RecordBatch Inputs

Use interior mutability so that each execute() call gets its own input.

Depending on the use case a few different structs are used

```
Arc<RwLock<Vec<RecordBatch>>>
```

```
Arc<RwLock<Option<RecordBatch>>>
```

```
Arc<RwLock<Option<UnboundedReceiver<RecordBatch>>>>
```

arroyo

Changes To ExecutionPlans

Arroyo needs to invoke execution plans cheaply, often on single RecordBatches.

Forked DataFusion so two calls to execute() run independently. Already true of most operators, just needed to remove some OnceAsync from Join implementations.

Add a reset() method to clear metrics on operators. Without this Arroyo was quickly OOM-killed.

Checkpointing Stateful Executions

Arroyo needs to be able to restore from a sudden disruption. Checkpoints are also used to rescale pipelines.

Two main approaches for stateful operators

- Checkpoint Inputs: Write data to S3 before computing against it. Easy, but potentially inefficient.
- Flush to intermediate data on checkpoint: Used by aggregates, makes use of the two phase aggregation capabilities of DataFusion

arroyo

User Rust UDFs via Dynamic Linking

```
#[derive(WrapperApi)]
struct UdfDylibInterface {
    run: unsafe extern "C" fn(
        args_ptr: *mut FfiArraySchemaPair,
        args_len: usize,
        args_capacity: usize,
    ) -> FfiArraySchemaPair,
}
```

arroyo

Arroyo’s Wishlist for DataFusion

- More flexible CREATE TABLE Statements
- Pluggable Metrics processing, including no-op processing
- Partial -> Partial AggregateExec Mode
- Better struct and union support
- Factory-style ExecutionPlans
- Fully Retractable Aggregate State
- Support for watermark-based flushing
- Support for checkpointing

EASY/SHOULD HAPPEN-MEDIUM/SOME COMPEXITY-HARD/A BAD IDEA?

Questions?

Also, come to Micah's talk on Thursday, at 1:30:

Why Streaming SQL? The Semantics and Challenges of Applying SQL to Unbounded Data

jackson@arroyo.dev

@jacksonrnewhouse (github)

linkedin.com/in/jackson-newhouse

arroyo