Feature 2083 Cache Resend Enhancements

- Status: Design
- Author: Jiyong Huang
- Discussion: https://github.com/lf-edge/ekuiper/issues/2083

Users need to differentiate between newly incoming data and resent data to handle them differently. For instance, they may want to use the newly incoming data for instant calculations and archive the resent data. This proposal suggests several enhancements to support the distinction between newly incoming data and resent data.

Use Cases

- Sink normal and resent data to different destinations, like different MQTT topic
- Sink normal and resent data with different priority, like higher priority for normal data
- Sink normal and resent data with different format, like the most common case, to add a field to indicate the data is resent

API Design

We will add properties to sinks including common properties and sink specific properties.

Common Properties

- `resendQueue`: boolean, default `false`. If `true`, the sink will send the resent data in separate queue from the normal data. If `false`, which is the default case, the sink will send the resent data in the same queue as the normal data in time order.
- `resendPriority`: int, could be -1, 0 and 1, default is 0. If 0, the resent data will be sent with the same priority as the normal data. If 1, the resent data will be sent with higher priority than the normal data. If -1, the resent data will be sent with lower priority than the normal data.
- `resendField`: string, the field name to update and indicate that the data is resent. If not set, the data will be resent as is. If set, the data will be resent with the field set to true. The field will be added if not exist. If the field already exists, it will be updated to true.

Sink Specific Properties

Each sink may have its own properties to control the resend behavior. Name some as an example:

TODO: There are actually a lot of properties affect the send behavior like dataTemplate, dataEncoding, etc. Shall we use a send config key to configure them as a whole?

MQTT Sink

- `resendTopic`: string, default is the same as the normal topic. If set, the resent data will be sent to this topic.

REST Sink

- `resendUrl`: string, default is the same as the normal path. If set, the resent data will be sent to this path.

File sink

- `resendPath`: string, default is the same as the normal path. If set, the resent data will be sent to this path.

Implementation

Sink API

Add `collectResend` method to `SinkNode` interface:

```
type Sink interface { // Open Should be sync function for normal case. The container will run it in go func Open(ctx StreamContext) error // Configure Called during initialization. Configure the sink with the properties from rule action definition Configure(props map[string]interface{}) error // Collect Called when each row of data has transferred to this sink Collect(ctx StreamContext, data interface{}) error // Collect Called when each row of data has transferred to this sink CollectResend(ctx StreamContext, data interface{}) error Closable }
```

Compatibility: If `collectResend` is not set, use `collect` instead.

Sink Cache

If `resendQueue` is set, add a cache queue for each sink which means there will be **two** output channels . The cache queue is a FIFO queue. Otherwise, keep the default behavior that one FIFO queue for all sinks.

Sink Node

1. Switch around 2 output queues. For normal queue, call `collect` method; for cache queue, call `collectResend` method.
2. Support `resendPriority` by setting channel priorities.
3. Support `resendField` by updating the data.

Sink Implementations

Take MQTT as an example:

1. Add `CollectResend` function to collect the resent data.
2. In `CollectResend`, send the data to the resend topic if `resendTopic` is set. Otherwise, send the data to the normal topic.