Feature 1637 Send out in batch and compression

- Status: Design
- Author: Jiyong Huang
- Discussion: https://github.com/lf-edge/ekuiper/issues/1637

Requirement

eKuiper sink can send data out to external systems. By default, the sink produce data for each event. But this could be a problem if the data throughput is large:

- Overhead when sending to cloud
- IO overhead when save to db/file
- Compression ratio is low

In order to save bandwidth with higher compression ratio and boost performance, we would like to introduce batch send and compression in **sink.**

Design

For batch send, we can achieve by two approaches:

1. Use window to batch data (supported now)
    a. Pros: apply for all sinks
    b. Cons:
        i. Not suitable for continuous query semantically, thus may make the SQL more complex even no window is needed
        ii. Cannot control in sink level, for example, cannot save locally in real time while publish to the cloud in batch
2. Set batch property for sink (to be implemented)
    a. Flexible

Usage

Add new properties into [sink common properties](https://ekuiper.org/docs/en/latest/guide/sinks/overview.html#common-properties).

- batchSize:
    - The upper bound of count of events to be batched together before sending out.
    - This works together with lingerInterval. If both are set, which one meets firstly will trigger a sending.
    - A small batch size will make batching less common and may reduce throughput (a batch size of zero will disable batching entirely). A very large batch size may use more memory andd reduce the sending times.
    - Default is 0, which means no batch, will send out after the linger time
- lingerInterval
    - The upper bound of time interval to wait before doing a send.
    - This works together with lingerInterval, if both are set, which one meets firstly will trigger a sending.
    - Default is 0, which means no linger. Will send out after the batchSize full. If both are 0, send out immediately.
- compressionType
    - The compression type to compress the data before sending out.
    - Optional property, should support `gzip`
    - In the future, support `gzip`, `snappy`, `lz4`, `zstd`

Use case

1. Publish to mqtt for **every 10 seconds defined by window** in **protobuf** format and compress by **gzip**

```
{
        "id": "rule1",
        "sql": "SELECT * FROM demo GROUP BY TumblingWindow(ss, 10)",
        "actions": [{
      "mqtt": {
        "server": "tcp://yourserver:1883",
        "topic": "mytopic",
        "format":"protobuf",
        "schema":"myschema.message",
        "compressionType": "gzip"
      }
        }]
}
```

2. Publish to mqtt for every 100 events or 100 seconds and compress by gzip; But publish to local mqtt for each event

```
{
        "id": "rule1",
        "sql": "SELECT * FROM demo",
        "actions": [{
      "mqtt": {
        "server": "tcp://cloud:1883",
        "topic": "remote",
        "sendSingle":"true",
        "batchSize": 100,
        "lingerInterval": 100000,
        "compressionType": "gzip"
      }
        },{
                "mqtt": {
        "server": "tcp://local:1883",
        "topic": "local",
        "sendSingle":"true"
      }
        }]
}

? Adapt to file format, currently splited by lines \n
? MQTT package size check
```

Implementation

1. compression: Just like format, feed the batch and compression properties into the transform *GenTransform(internal/topo/transform/template.go)* function. Then each sink can use context `ctx.TransformOutput(data)` to do the transformation which will have compression is property set.
2. batch: In *SinkNode(internal/topo/node/sink_node.go)*, accumulate by strategy before passing on to the sink implementation. Need to consider state saving.
3. In each sink implementations, if batch/compression is not supported, return error in the validate function. If supported, check if collect logic needs to be changed.

Source consideration (Future)

“Drinking Our Own Champagne”, the compressed data produced by sink should be abled to be received by our source. This will add new properties like:

- compressionType: decompress the compressed content
- isBatch: decode the batch events and replay in stream