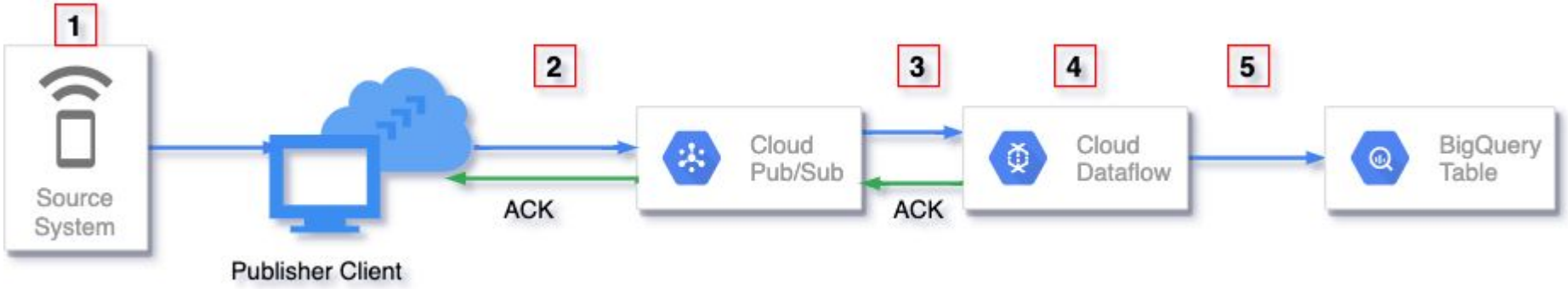
Handling duplicates in streaming pipelines using Pub/Sub and Dataflow

Zeeshan Khan
Cloud Data Engineer | Google

Google Cloud

Streaming architecture on GCP



1: Source generated duplicates

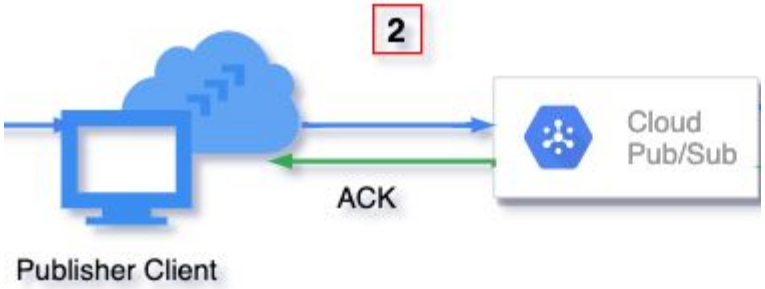
Your source system may generate duplicates because of retries, errors, network failure, etc.

PubSubMessage

```
{
  "data": string,
  "attributes": {
     string: string,
     …
   },
  "messageId": string,
  "publishTime": string,
  "orderingKey": string
}
```

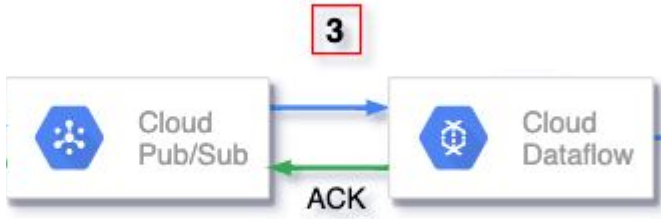
2: Publisher generated duplicates

- Messages are considered successfully published when acknowledged by the Pub/Sub service.
- Publishing may be retired if acknowledgement was not received within a deadline.
- Can produce duplicate messages with different *message_id*.



3: When reading from Pub/Sub

- Pub/Sub offers at-least once delivery
- Subscriber may receive the same message more than once.
- However duplicates have the same *message_id* and Apache Beam PubsubIO does a default deduplication.
- There is no time window for this default deduplication.

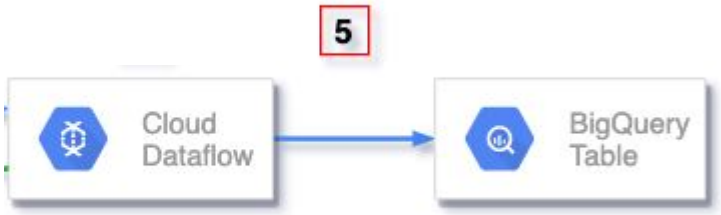


4: When processing in Dataflow

- Message can be processed more than once by workers in event of failures which may produce duplicates.
- However, Dataflow offers exactly once processing and does checkpoints and commits before moving from one stage to another.
- Such duplicates are taken care of by Dataflow, and developers don't have to worry about it.
- Common mistake : Have side effects, logging from DoFn. calls to external API

5: When writing to sink

- Each element can be retried multiple times by Dataflow workers and may produce duplicate writes.
- It is the responsibility of the sink to detect these duplicates and handle accordingly.
- Depending on the sink, duplicates may be filtered out, overwritten or appear as duplicates.



5.1 : BigQuery as a sink

- Each message is provided with an insert_id when writing to BigQuery
- Deduplication guarantee depends on the insert method used to write data to BigQuery.

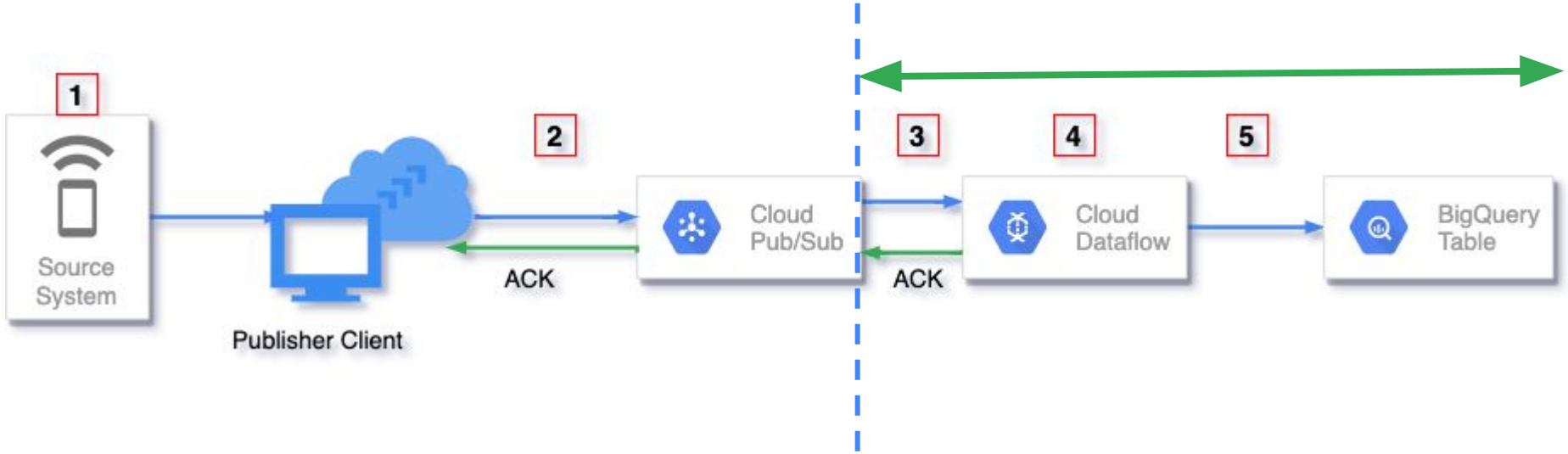
BigQuery I/O Insert method	Pipeline type	Deduplication guarantee
FILE_LOADS	Streaming or Batch	Guaranteed deduplication
STREAMING_INSERTS	Streaming	Best effort deduplication
STORAGE_WRITE_API	Streaming or Batch	Guaranteed deduplication

5.2 : File systems as sink

- Exactly once is guaranteed as any retries by Dataflow workers in event of failure will overwrite the file.
- Beam provides several I/O connectors to write files, all of which guarantees exactly once processing.

I/O Category	Apache beam I/O
File based	FileIO, AvroIO, TextIO, TFRecordIO, XmlIO, TikaIO, ParquetIO, ThirftIO
FileSystem	HadoopFileSystem, GcsFileSystem, LocalFileSystem, S3FileSystem

Streaming architecture on GCP



Deduplication options for source generated or publisher generated duplicates

- In both cases, we have duplicate messages with different message_id, which for Pub/Sub and downstream systems like Dataflow or BigQuery are two unique messages.

```
{
  "data": “test”,
  "attributes": {
      unique_id: 123#abc,
      …
    },
  "messageId": 123456,
  "publishTime": 2021-01-01 02:04:06,
  "orderingKey": ..
}
```

```
{
  "data": “test”,
  "attributes": {
      unique_id: 123#abc,
      …
    },
  "messageId": 123457,
  "publishTime": 2021-01-01 02:05:01,
  "orderingKey": ..
}
```

Option 1: Leverage Pub/Sub message attributes

- Set Pub/Sub message attributes when publishing
- Leverage these attributes for deduplication
- This deduplication guaranteed to work for duplicate messages that are published to Pub/Sub within 10 minutes of each other.

```
{
  "data": “test”,
  "attributes": {
      unique_id: 123#abc,
      …
    },
  "messageId": 123456,
  "publishTime": 2021-01-01 02:04:06,
  "orderingKey": string
}
```

1: Leverage Pub/Sub message attributes

```
p.apply("Read PubSub Messages",
         PubsubIO
           .fromSubscription("<PUB/SUB SUBSCRIPTION>")
           .readMessagesWithAttributes()
           .withIdAttribute("<PUB/SUB MESSAGE ATTRIBUTE KEY>"));
```

```
ReadFromPubSub(
        subscription="<PUB/SUB SUBSCRIPTION>",
        with_attributes=True,
        id_label="<PUB/SUB MESSAGE ATTRIBUTE KEY>")
```

Option 1: Leverage Pub/Sub message attributes

Cons	Pros
Need control over publishing to set message attributes. Deduplication guaranteed only if duplicate messages are published to Pub/Sub within 10 mins. This duration cannot be configured	No impact on latency No additional Dataflow processing cost

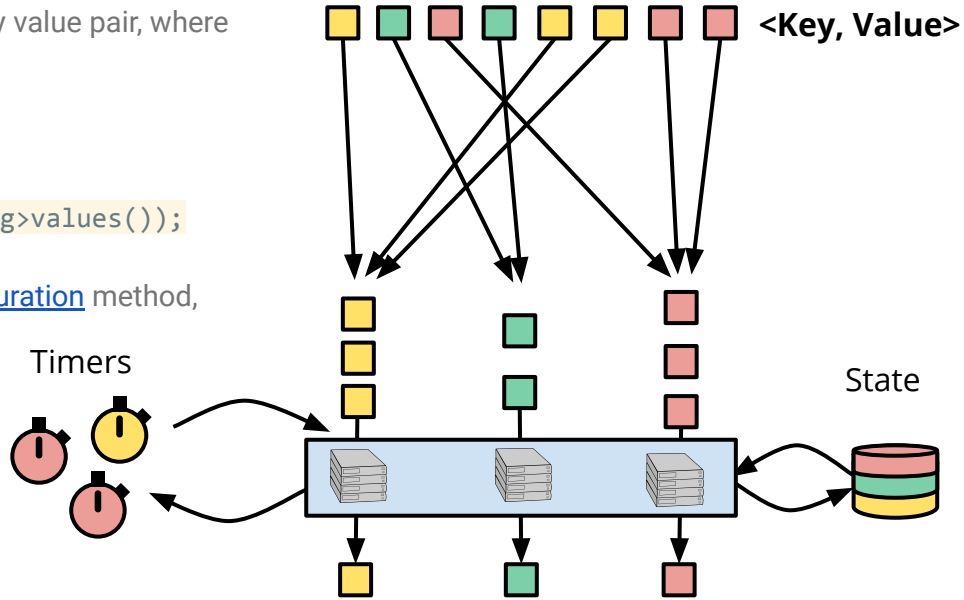
Option 2: Use Apache Beam Deduplicate PTransform

1) Deduplication can be based on the message or a key value pair, where the key could be derived from the message fields.

```
PCollection<String> words = ...;
PCollection<String> deduplicatedWords =
              words.apply(Deduplicate.<String>values());
```

2) You can configure the time duration using the withDuration method, which can be based on processing time or event time (specified using the withTimeDomain method).

Check Java documentation and Python documentation for more details on how this works.



Google Cloud

Option 2: Use Apache Beam Deduplicate PTransform

Cons	Pros
- Added Dataflow cost from reads and writes to the state stored in Streaming Engine. - Some added latency because of shuffling caused by the Stateful API.	- Full control over the deduplication window by selecting appropriate time duration. - Can use a unique message identifier for deduplication.

Option 3: Do post-processing in sink

Run scheduled batch job to do deduplication

Create materialized views

```
CREATE MATERIALIZED VIEW
     <project-id>.<my_dataset>.<deduplicated_base_table>
AS SELECT DISTINCT * FROM <base_table>
```

Option 3: Do post-processing in sink (BigQuery as an example)

Cons	Pros
- Additional cost associated with Materialized views - Restricted SQL syntax	- No impact on latency - Zero Maintenance

Questions ?