Generating Reproducible Out-of-Order Data Streams

Philipp M. Grulich
grulich@tu-berlin.de
Jonas Traub
jonas.traub@tu-berlin.de
Asterios Katsifodimos
a.katsifodimos@tudelft.nl
Tilmann Rabl
tilmann.rabl@hpi.de
Sebastian Breß
sebastian.bress@dfki.de
Volker Markl
volker.markl@tu-berlin.de

Experiments with Out-of-Order Streams

We provide a scalable data stream generator, which introduces configurable out-of-orderness in real-world data streams. This enables reproducible and realistic experiments.

- Handling out-of-order data streams is a key feature of modern stream processing systems [1].
- Research on the support of out-of-order stream processing requires reproducible, scalable, and configurable experiments on out-of-order data streams. For example, research on efficient window aggregation [2,3,4].

The evaluation of out-of-order capabilities is hard:
- Public real-world datasets do not reflect all aspects of out-of-order streams (e.g., different delays and fractions of out-of-order tuples).
- Experiments without real-world data can lead to unrealistic results.

Architecture Overview

A general out-of-order experiment data generator:
- Introduce out-of-order data to real-world input data sets.
- Generic configuration of out-of-orderness to enable full parameter exploration.
- Reproducible generation of experimental data with configurable out-of-orderness.

Generator Configuration

```
"dataSource": {
  "file": "Spash\", 1
  "separator": ":\"\":\"\":\"\":\"\":\"\":\"\", 2
  "time": {
    "timeIndex": "$\{old\}\", 3
    "sourceTimeUnit": "ps\"\"ns\"\"ms\"\"s\"
  } 4
} 5

"experimentDataConfigurations": [ 6
  { 7
    "targetOutOfOrderFactor": [0-100], 8
    "minDelay": 0, 9
    "maxDelay": 2000, 10
    "delaySeed": $\{seed\}$ 11
  } 12
]
```

- Configurable aspects of out-of-order streams:
  - Fraction of out-of-order tuples (How many tuples are out-of-order?).
  - Minimal/maximal event delay (How late are out-of-order tuples?).
  - Out-of-order delay distribution (How are delays distributed?).

References


Adding Out-of-Order Streams to Data Streams

Source Data Stream:

Out-of-Order Stream:

Assigned Windows

15 10 5 1

Goal:
- The generation of out-of-order tuples has to be deterministic and must not change query results.

Naive Solution:
- Generating out-of-order data by adding a random delay to the event time of certain tuples → changes temporal data distribution.

Our Approach:
- Shift ingestion times of source tuples and keep original event times.

Generator Algorithm

Step 1. Preprocessing:
- Analyze the out-of-orderness of the source stream data to take this knowledge into account for data generation.

```
maxTs ← 0;
for record in sourceFile do
  if record.ts < maxTs then
    RegisterDelayedRecord(record);
  end
  maxTs ← record.ts;
end
```

Step 2. Generation of out-of-order ingestion time:
- If an in-order tuple becomes an out-of-order tuple, we add a random delay to its ingestion time (based on the configured distribution).
- Finally, we sort the data set by ingestion time.

```
for record in sourceBuffer do
  if record.ts < maxTs then
    record.ingestionTime ← record.ts;
  else
    delay ← createDelay();
    record.ingestionTime ← record.ts + delay;
  end
end
sort(recordBuffer, r → r.ingestionTime);
```

Step 3. Ingestion to the stream processing system:
- Only ingest records if ingestion time is reached.

```
for record in sourceBuffer do
  if record.ingestionTime > now then
    wait(record.ingestionTime-now);
  end
  emit(record);
end
```