# RT Intro & Kafka

**Vybornov Artyom**, avybornov@bigdatateam.org
Big Data Instructor, http://bigdatateam.org/
Head of Data Platform, Rambler Group
https://www.linkedin.com/in/artvybor/

X5

2021-04-08 / Moscow

# RT Intro & Kafka

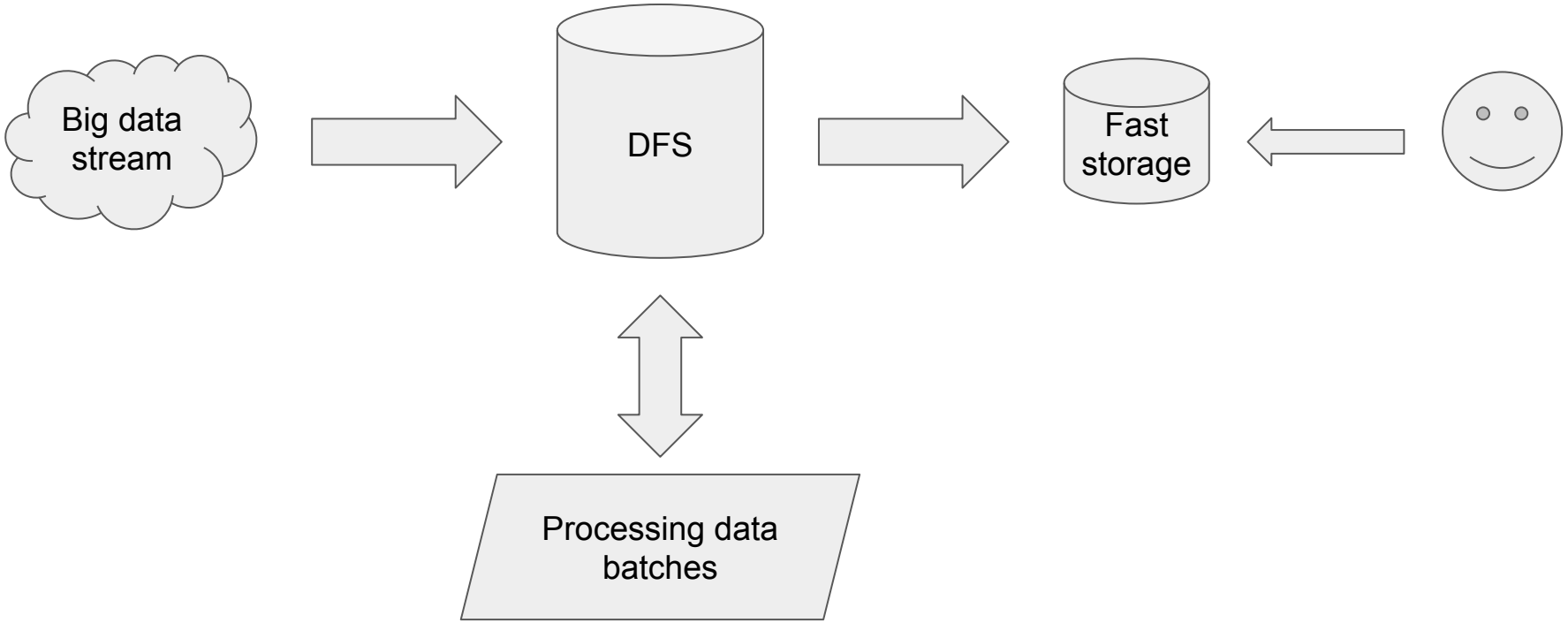
- RT Intro
  - Batch -> RT
  - Approaches to RT data processing
- Kafka
  - Internal
  - CLI

# RT Intro & Kafka

- RT Intro
  - **Batch -> RT**
  - Approaches to RT data processing
- Kafka
  - Internal
  - CLI

BIGDATA TEAM

# Batch approach

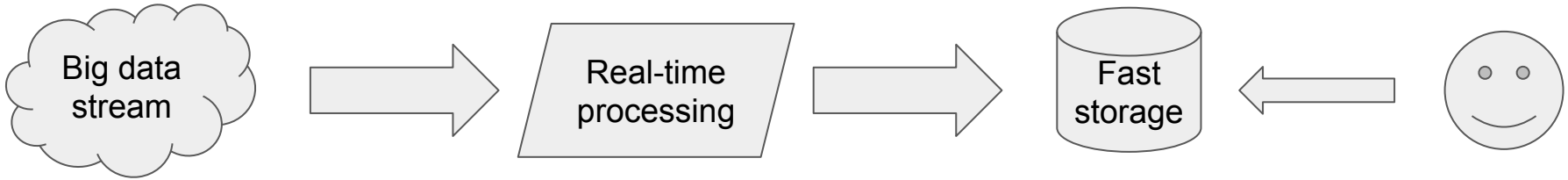


# The main cons of batch approach

- In practice a batch is a big time interval like an hour or day
- The size of batch is a minimal value for a lag
- Lag (delay) is the time between the event and its recording in the results of work
- For many purposes the following rule works: less lag => more valuable data

# Real-time big data

- Real-time big data - is a set of technologies to process big data with minimal lag
- Without DFS
- Work with a stream of data instead of a batch



BIGDATA TEAM

# Real-time big data - lag in minutes

- Build custom recommendations (Linkedin, Facebook)
  - Billions of events per day
  - Millions of events every second
  - Minutes to build fresh recommendation

BIGDATA TEAM

# Real-time big data - lag in seconds

- Programmatic advertising (Google, Facebook)
  - Billions of events per day
  - You liked the article and saw the relevant ads already on the next page
- Credit card real-time fraud detection
  - It takes a few seconds to detect a card with suspicious activity and block it

BIGDATA TEAM

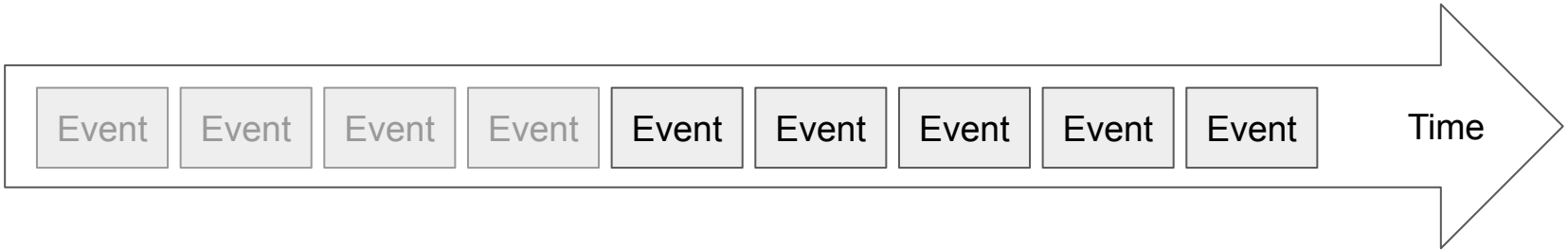
# Real-time big data - lag in ms

- Calls billing (mobile network operator)
  - Hundred millions of users
  - Less than a second to debit funds
  - Zero fault tolerance
- High-frequency trading (HFT)
  - The reaction to the change of quotations of the exchange with a delay of 10-100 milliseconds

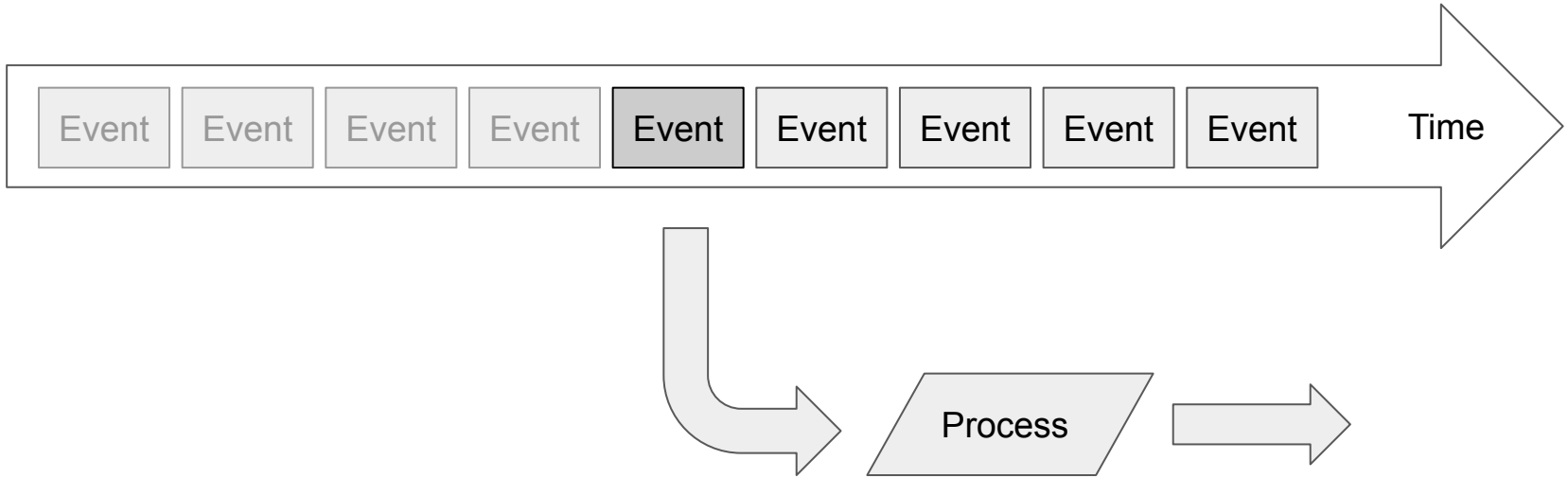
# RT Intro & Kafka

- ▶ RT Intro
  - ▷ Batch -> RT
  - ▷ **Approaches to RT data processing**
- ▶ Kafka
  - ▷ Internal
  - ▷ CLI

# Event-based approach

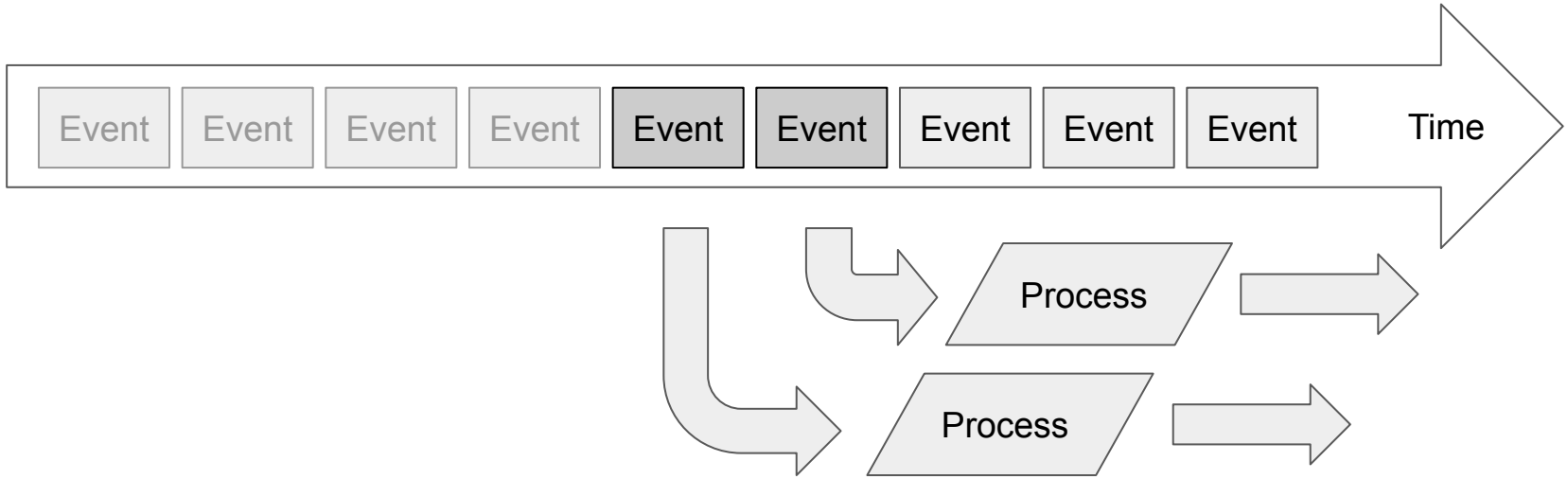


# Event-based approach




- Handling one event at a time

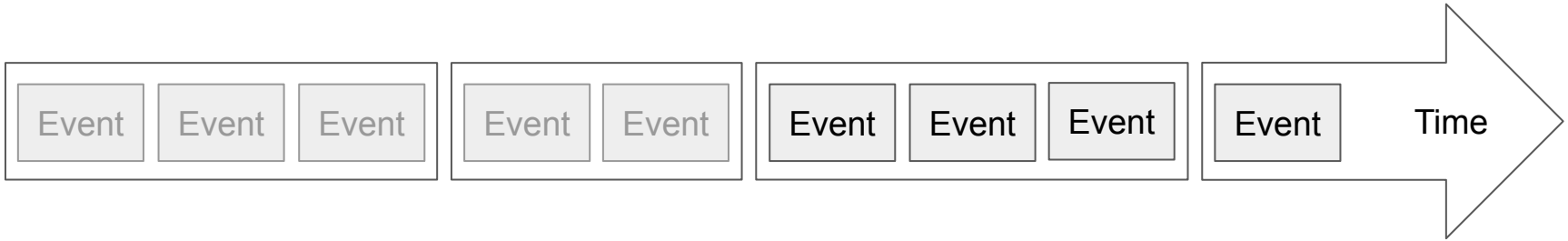
# Event-based approach




- Handling one event at a time
- Events are processed in parallel, but completely independently
- Latency ~10ms

BIGDATA TEAM

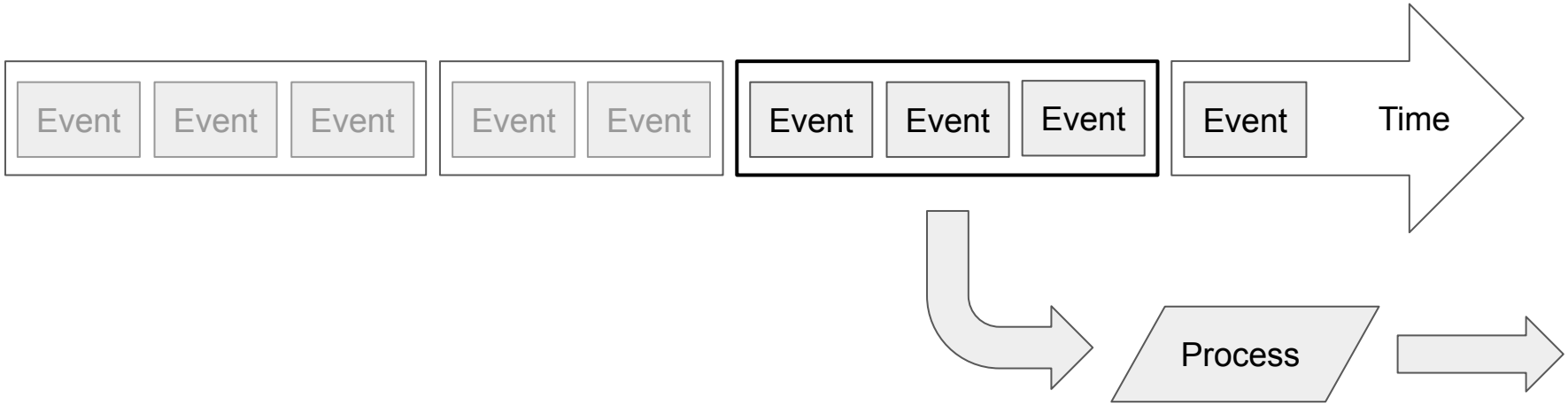
# Micro-batch approach




- Stream is cut to batch by time (for example 10 seconds batch)

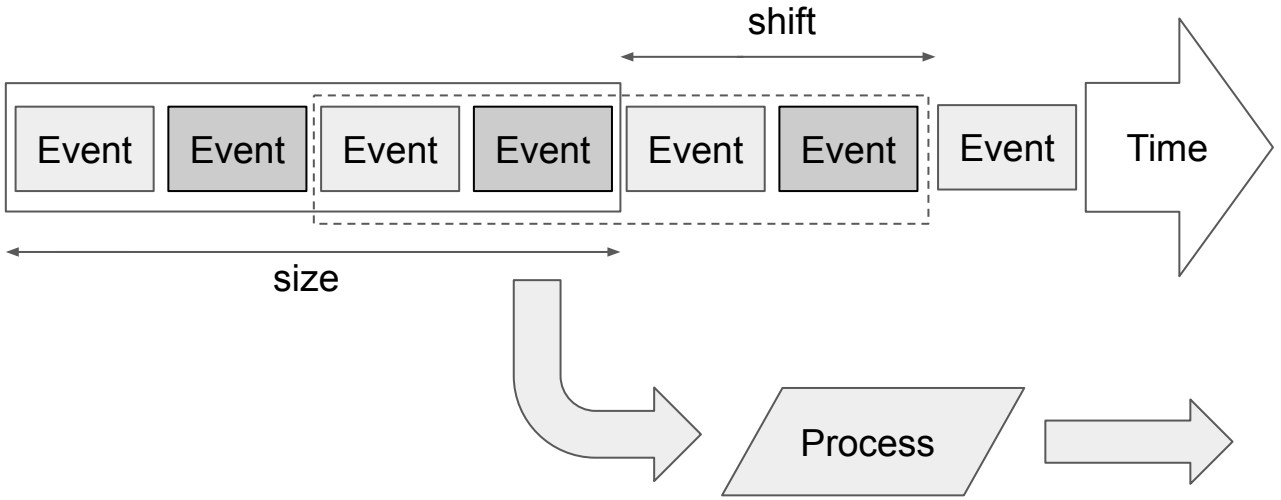
BIGDATA TEAM

# Micro-batch approach




- Stream is cut to batch by time (for example 10 seconds batch)
- Batches are processed sequentially
- Latency >> 1s

# Windowed approach



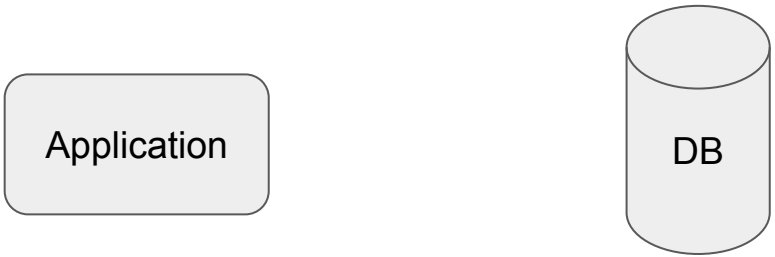

- Batch as a sliding window
- Similar to micro-batch approach

# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing

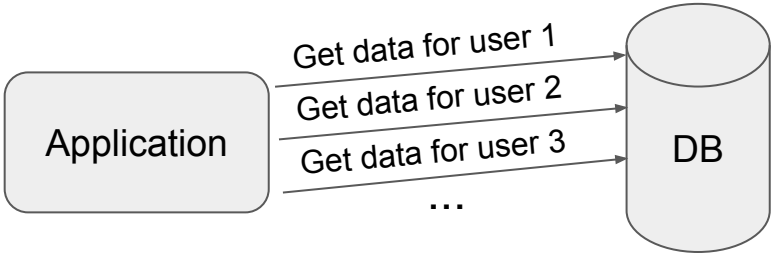
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



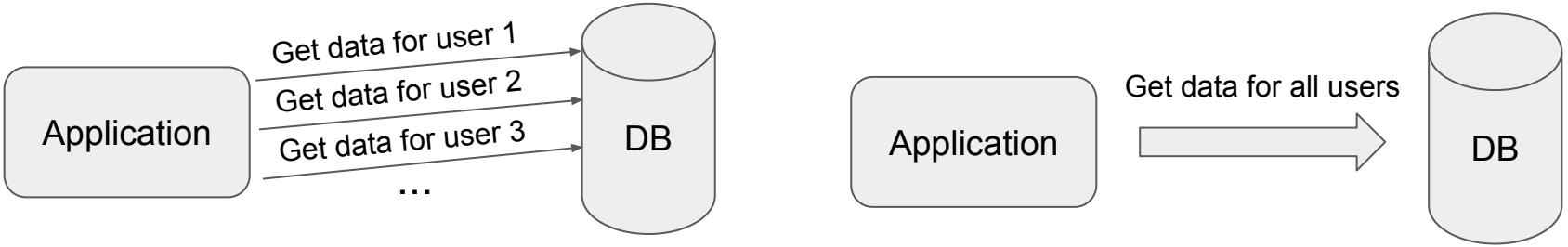
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



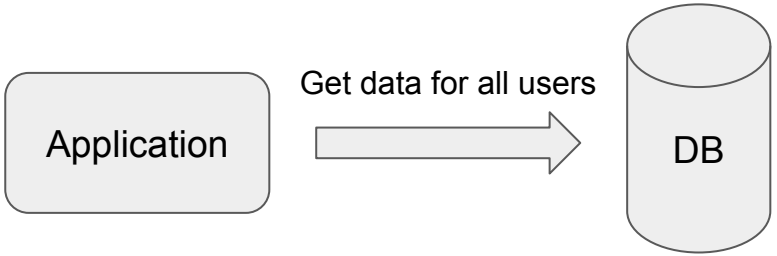
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



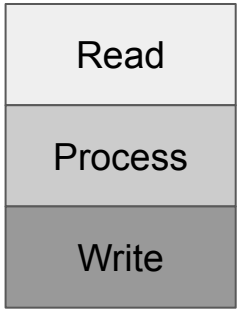
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing

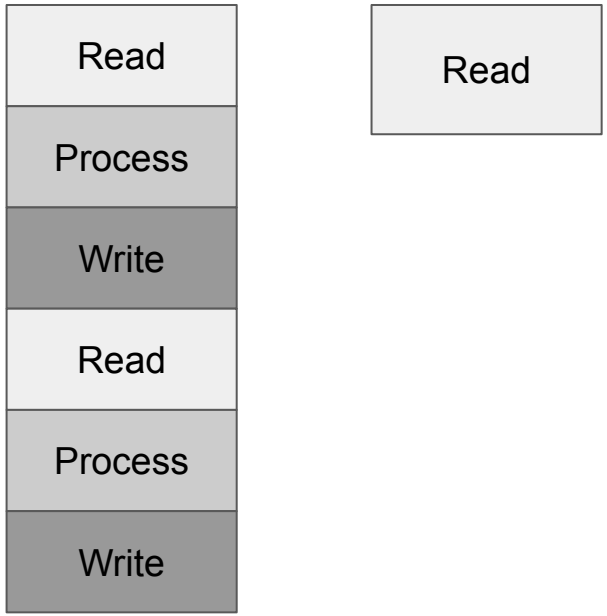


# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing

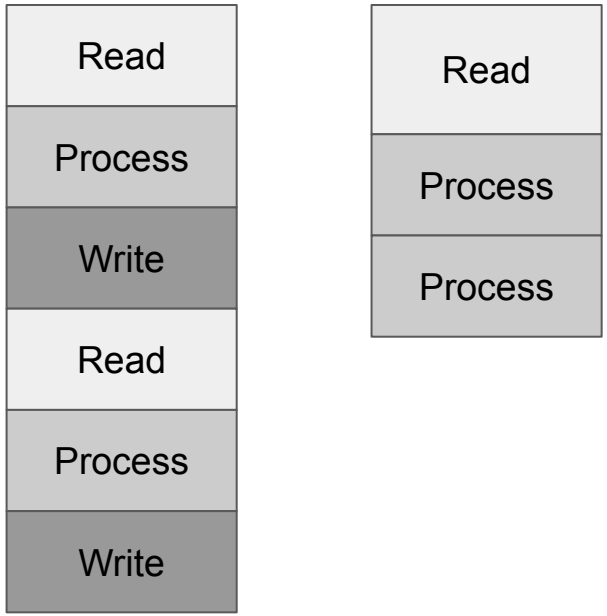
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



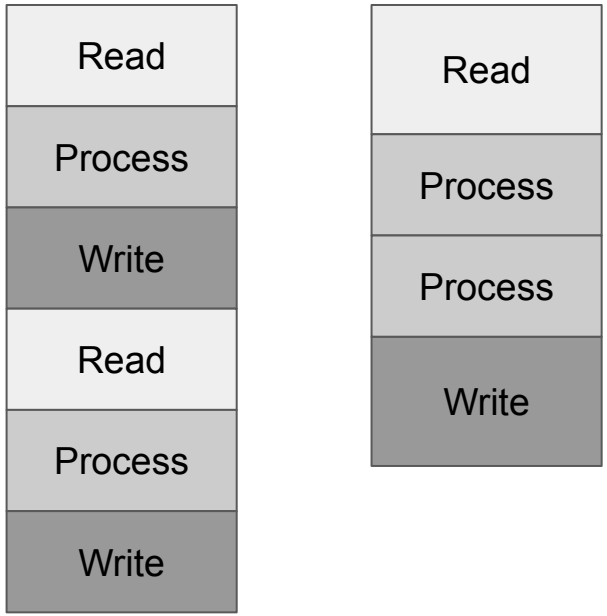
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



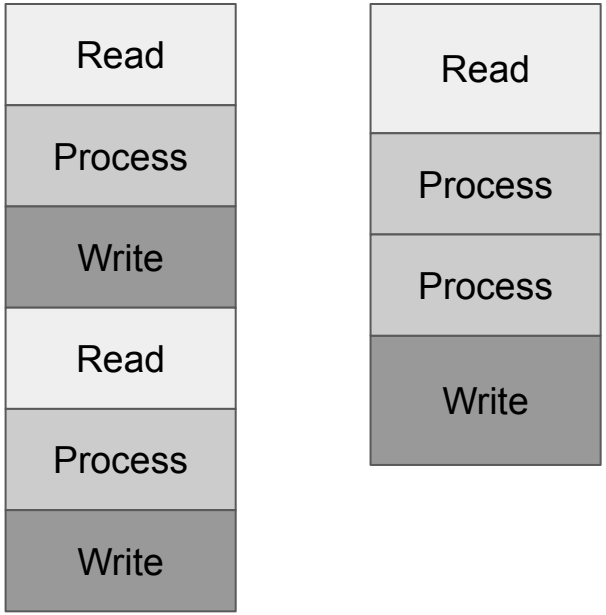
# Event-based vs micro-batch

- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing



# Event-based vs micro-batch

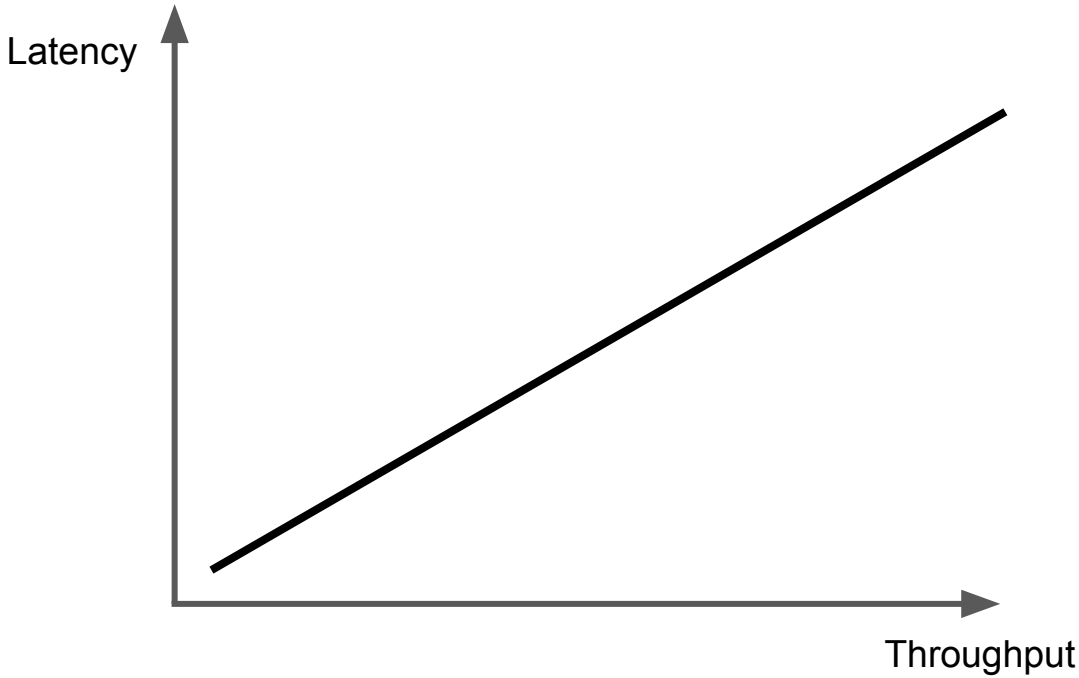
- Event-based allows you to achieve less lag
- Micro-batch allows you to save resources by reducing the common parts of each event handling/processing
- Micro-batch allows you to process more data on the same hardware



# Throughput vs latency

- In real world resources are restricted
- Big data needs a huge throughput => in most cases we a choosing micro-batch
- There is no right answer - you should choose the approach by task

# Throughput vs latency

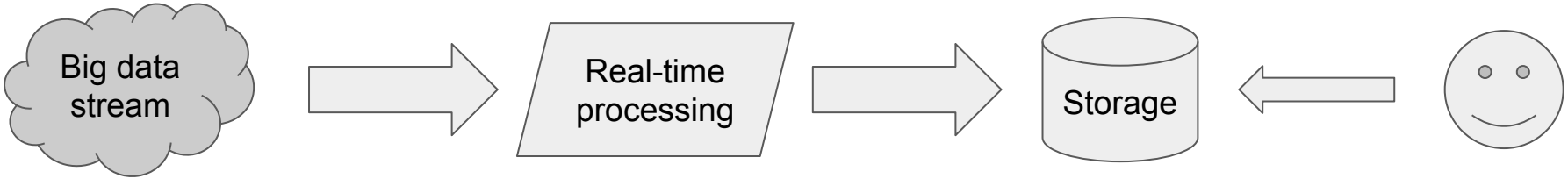


# RT Intro & Kafka

- ▶ RT Intro
  - ▷ Batch -> RT
  - ▷ Approaches to RT data processing
- ▶ **Kafka**
  - ▷ Internal
  - ▷ CLI

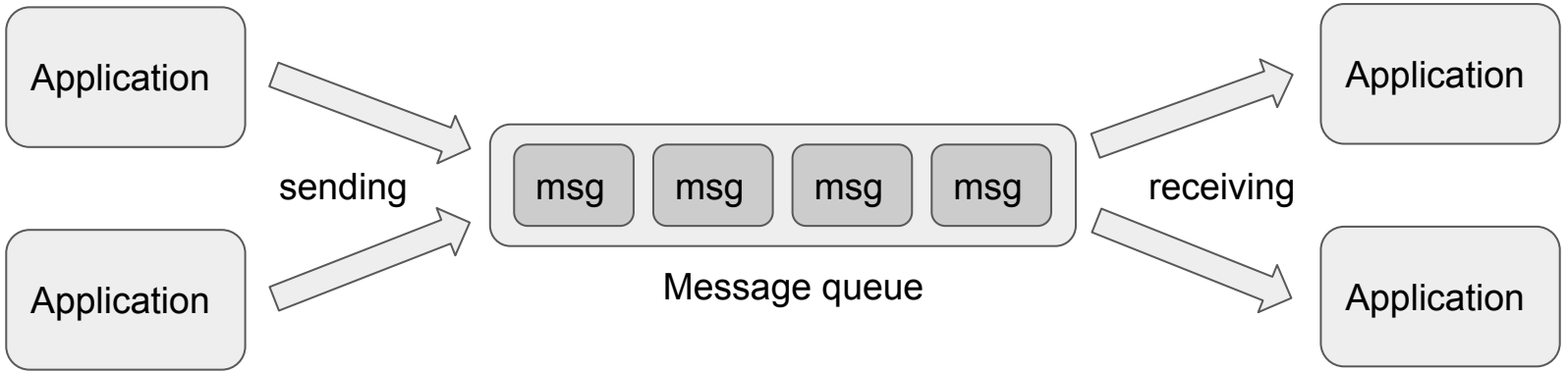
BIGDATA TEAM

# Requirements of storage for input data




- Event stream
- Big throughput (hundreds of thousands message per second)
- Small latency (less than 1s)

# Stream handling - message queue




- Message queue provides an asynchronous communications protocol between applications or between processes/threads inside a single application

# Key features of classical message queue

- Complex schemes of message delivery
  - Reduces the throughput
- Per-message state
  - Reduces the throughput
- Stores the data in RAM
  - Not persistent storage
  - Strongly limits the amount of stored data

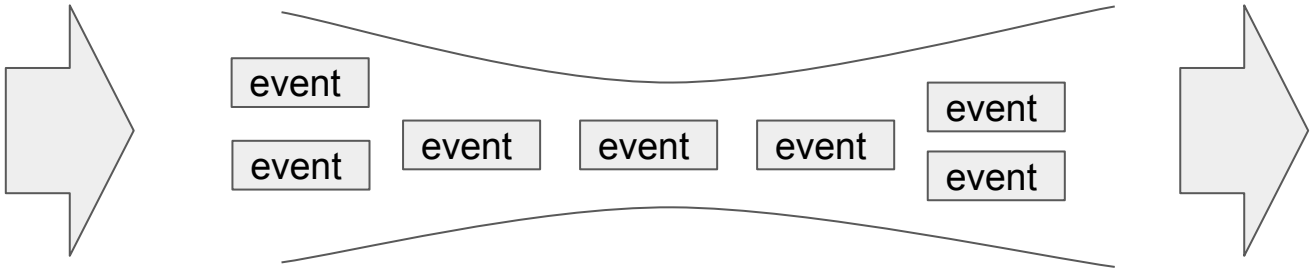
# Storage of events in big data world

- Simplify the message delivery scheme
- Per message state not tracked by queue
- High throughput
- Persistent storage for a big amount of data

# Kafka

- Kafka is a unified, high-throughput, low-latency platform for handling real-time data feeds
- Kafka is a data bus for big data
- Kafka is an input events storage for real-time processing
- Kafka is an event-based real-time processing engine (Kafka Streams)

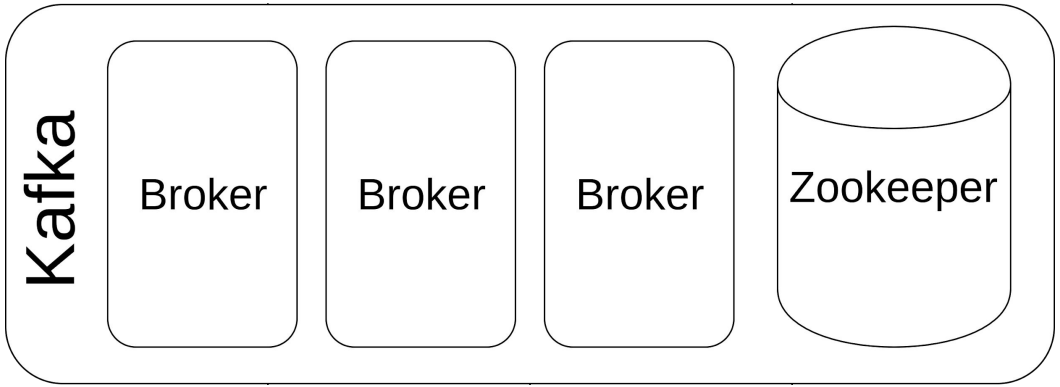
# Kafka



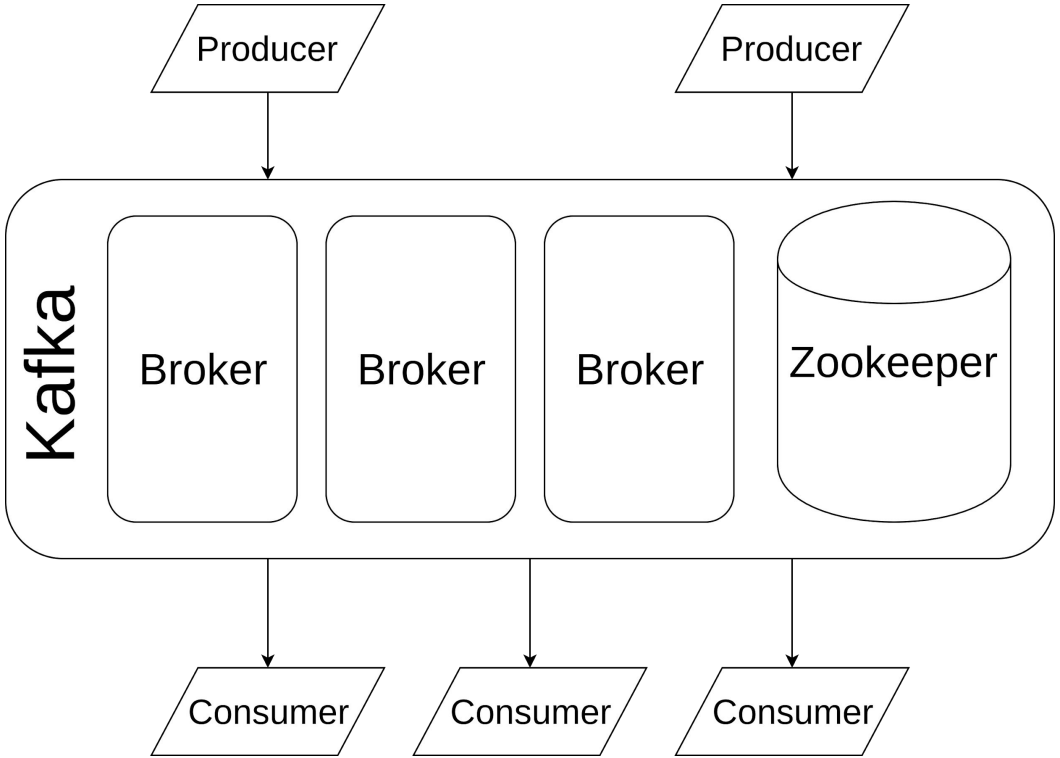
# RT Intro & Kafka

- ▶ RT Intro
  - ▷ Batch -> RT
  - ▷ Approaches to RT data processing
- ▶ Kafka
  - ▷ **Internal**
  - ▷ CLI

# Kafka architecture

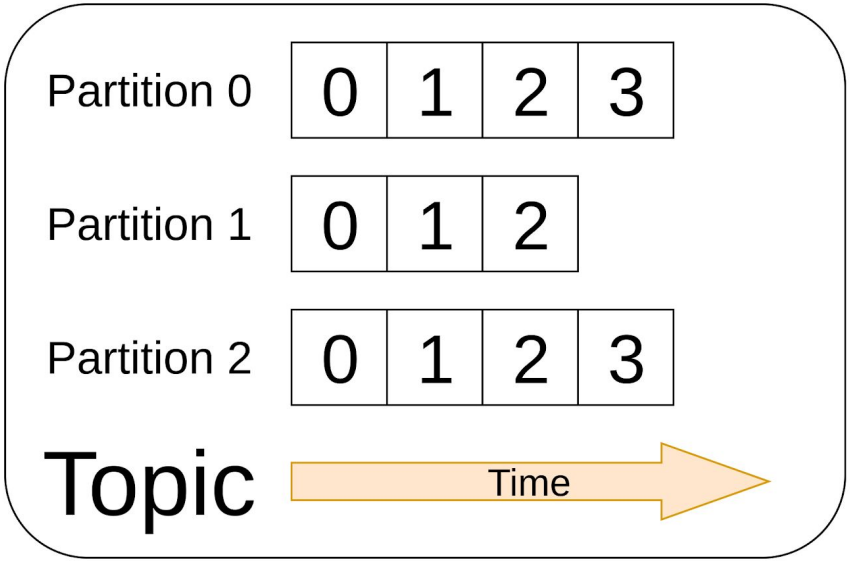


# Kafka architecture

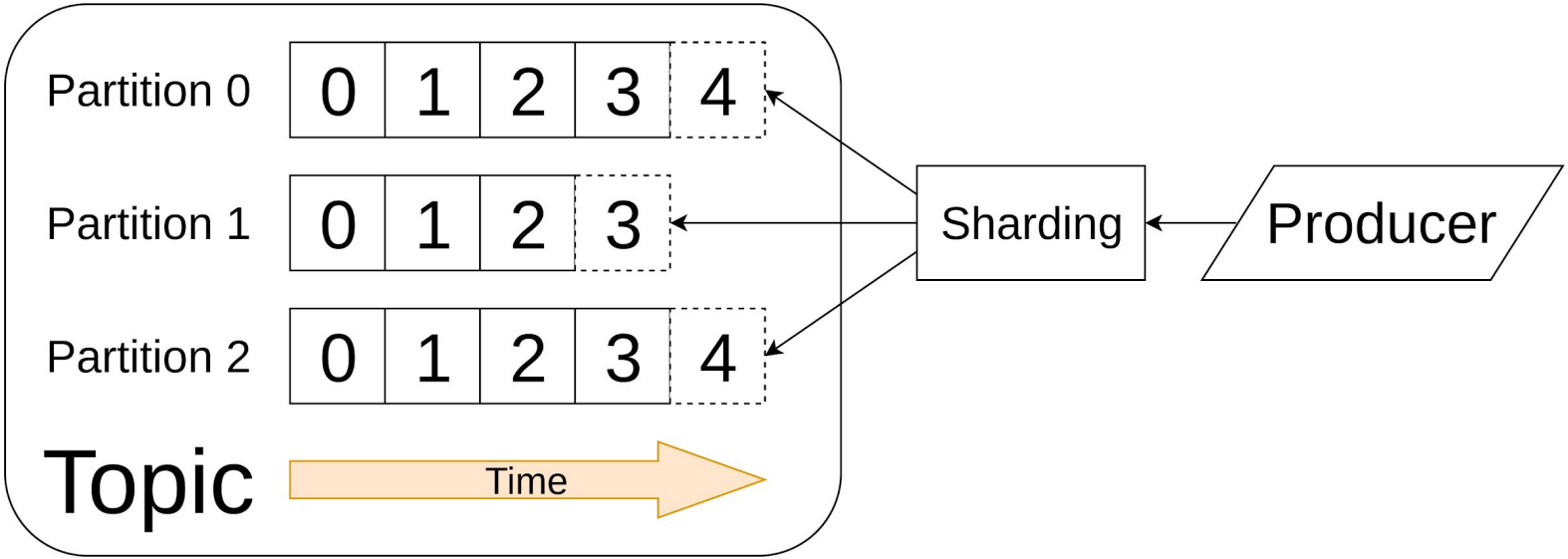


# Topic

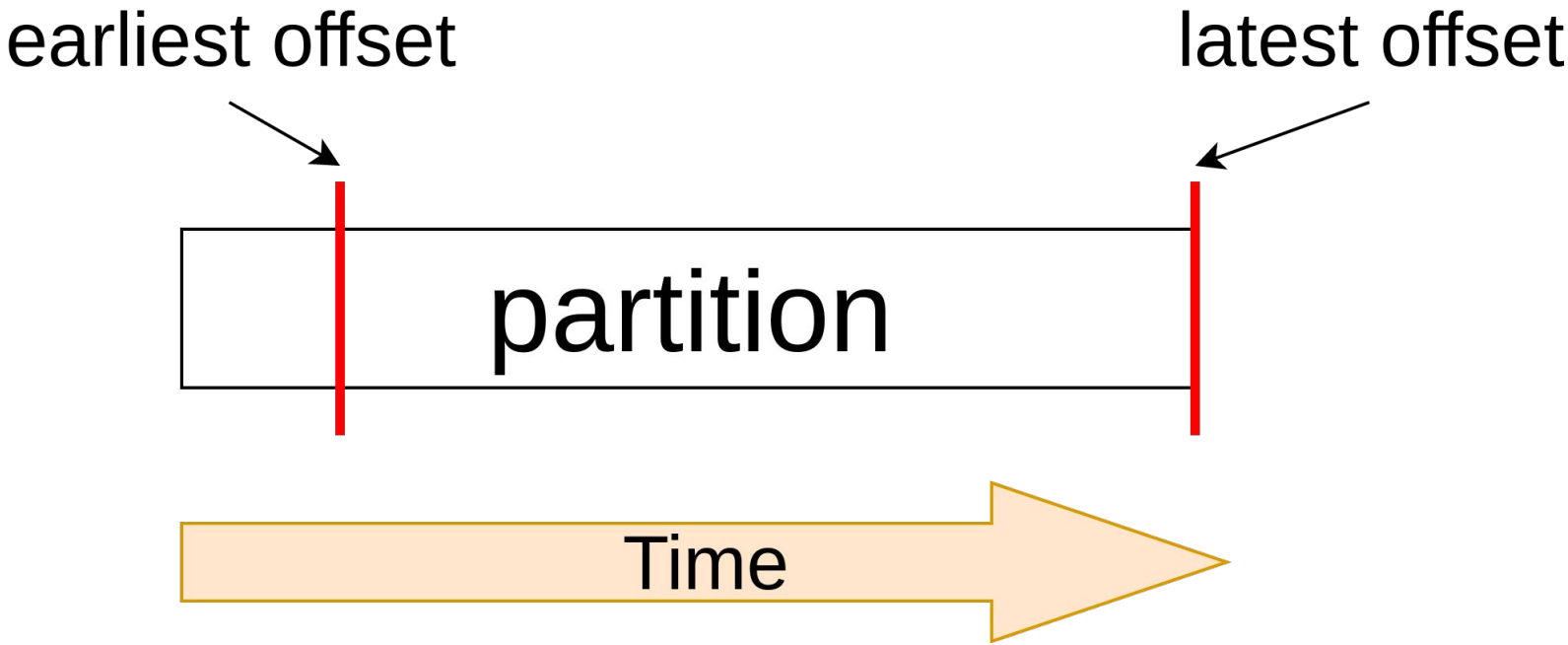
# Topic


Partition 0
0
1
2
3
Partition 1
0
1
2
Partition 2
0
1
2
3
Topic
Time

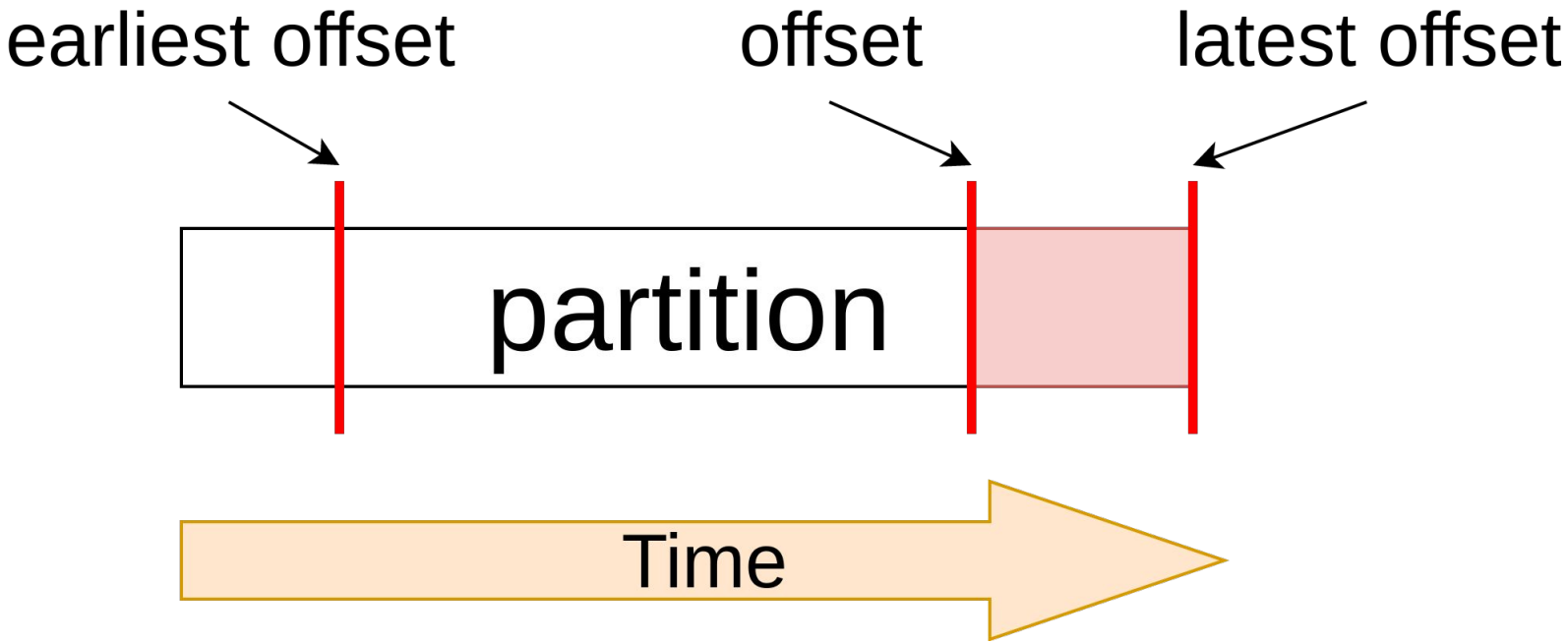
# Topic & Write


Partition 0
0
1
2
3
4
Partition 1
0
1
2
3
Partition 2
0
1
2
3
4
Topic
Time
Sharding
Producer

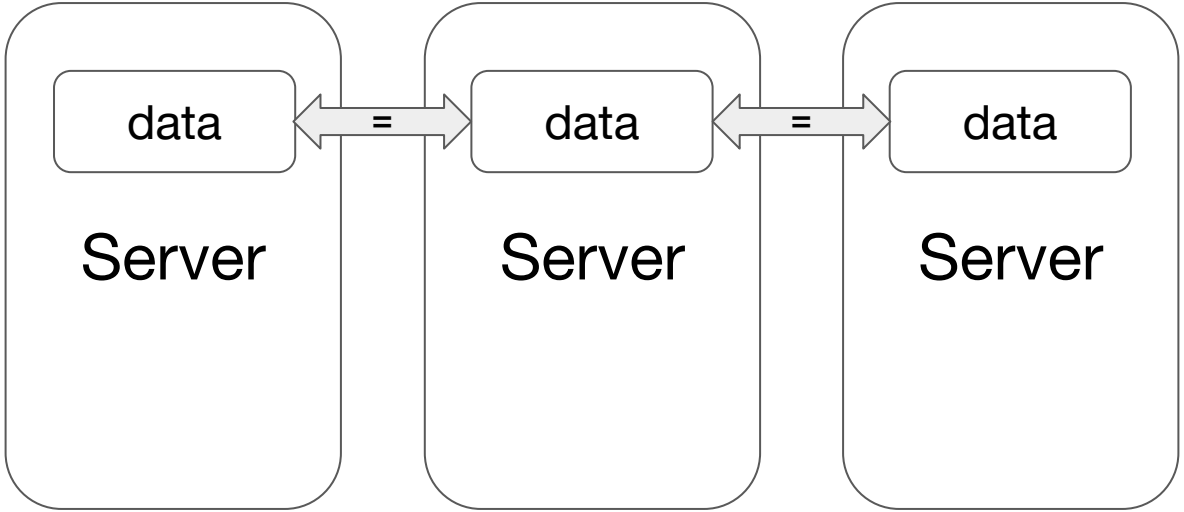
# Offsets



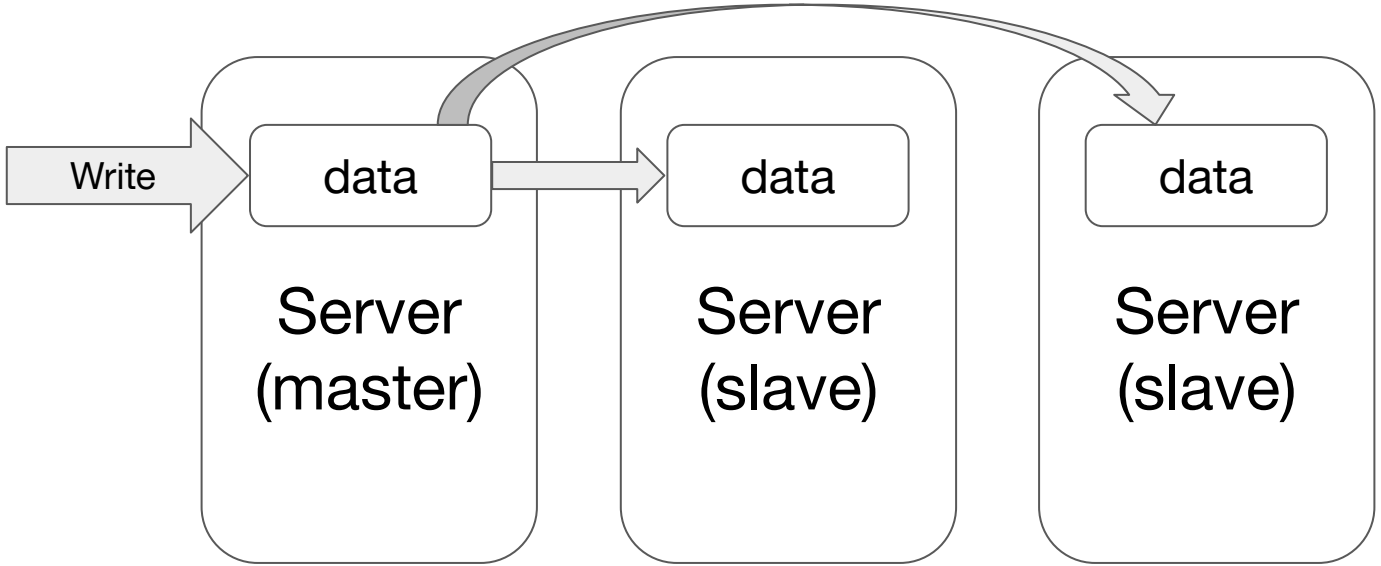
# Offsets & Read



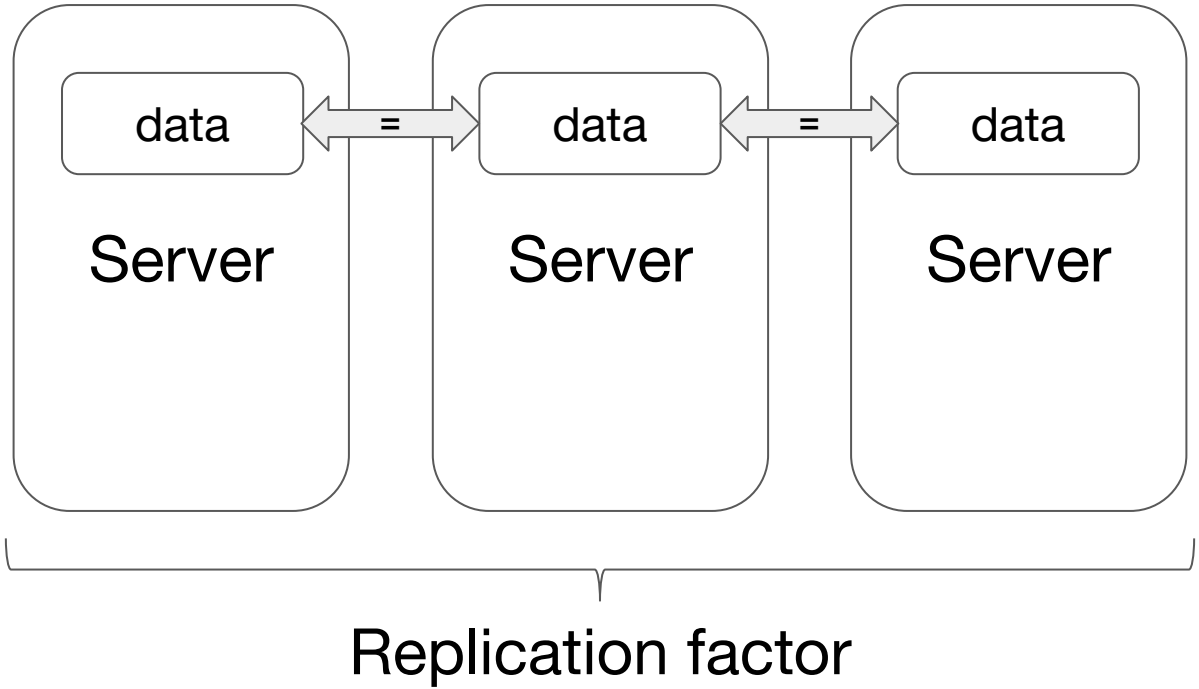
# Replication



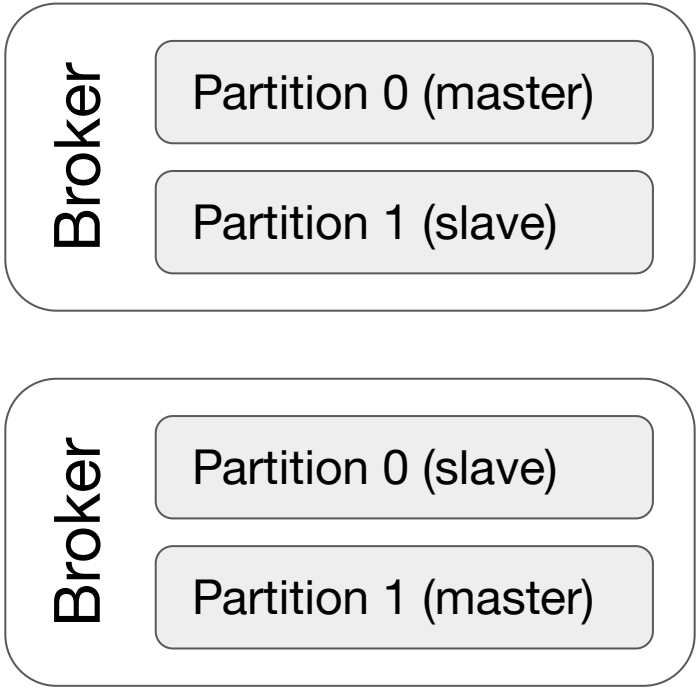
# Replication



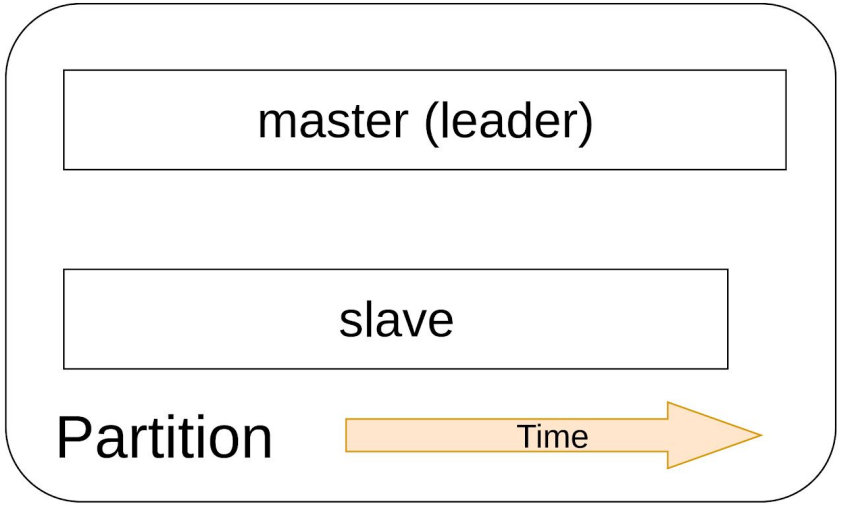
# Replication



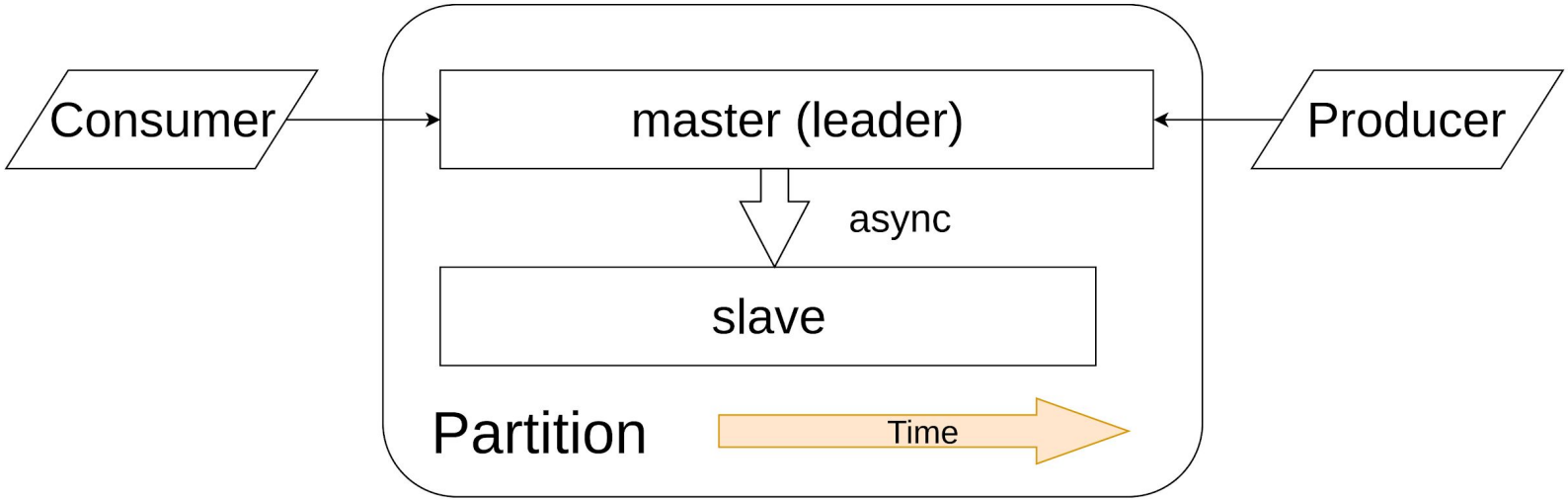
# Kafka replication



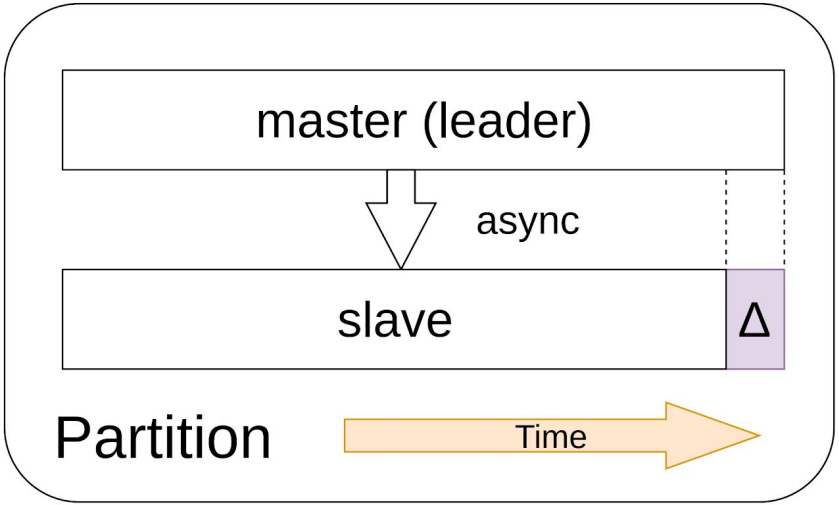
# Kafka replication



# Kafka replication



# Kafka replication delay



# RT Intro & Kafka

- ▶ RT Intro
  - ▷ Batch -> RT
  - ▷ Approaches to RT data processing
- ▶ Kafka
  - ▷ Internal
  - ▷ **CLI**

# Kafka CLI: kafka-topics

- It is the utility to create, delete, describe or change the topic
- `kafka-topics.sh --bootstrap-server $BROKERS --create --topic test_topic --partitions 3 --replication-factor 2`
- `kafka-topics.sh --bootstrap-server $BROKERS --describe --topic test_topic`
- `kafka-topics.sh --bootstrap-server $BROKERS --list`

# Kafka CLI: kafka-console-producer

- It is utility to send data from standard input and to Kafka topic
- `kafka-console-producer.sh --broker-list $BROKERS --topic test_topic`

# Kafka CLI: kafka-console-consumer

- It is utility to read data from Kafka topic
- ```
  kafka-console-consumer.sh --bootstrap-server $BROKERS
  --topic test_topic --from-beginning
  ```
  ```

# Kafka CLI: kafka-run-class

- It is entry point to run any class in the Kafka environment
- 
```
kafka-run-class.sh kafka.tools.GetOffsetShell
--broker-list $BROKERS --topic test_topic --time -1
```

# RT Intro & Kafka

- ▶ RT Intro
  - ▷ Batch -> RT
  - ▷ Approaches to RT data processing
- ▶ Kafka
  - ▷ Internal
  - ▷ CLI

# Thank you! Questions?

**Vybornov Artyom**, avybornov@bigdatateam.org
Big Data Instructor, http://bigdatateam.org/
Head of Big Data Dev Team, Rambler Group
https://www.linkedin.com/in/artvybor/