Introduction to Data Stream Processing

Amir H. Payberah
payberah@kth.se

2023-09-25

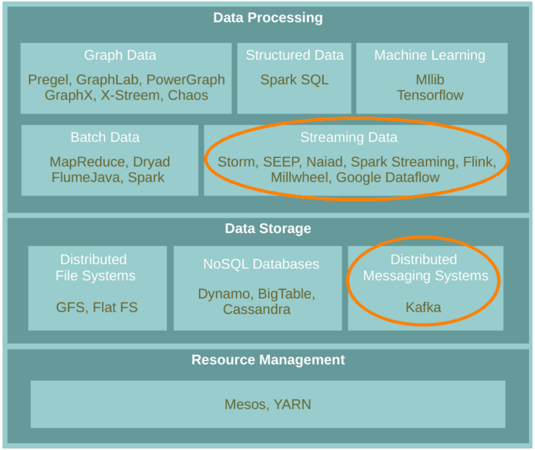
The Course Web Page

`https://id2221kth.github.io`

The Questions-Answers Page

https://tinyurl.com/hk7hzpw5

Where Are We?

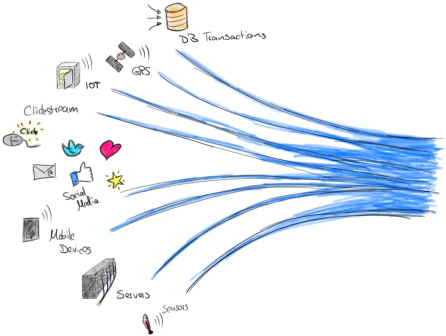

Data Processing
Graph Data
Pregel, GraphLab, PowerGraph
GraphX, X-Streem, Chaos
Structured Data
Spark SQL
Machine Learning
Mllib
Tensorflow
Batch Data
MapReduce, Dryad
FlumeJava, Spark
Streaming Data
Storm, SEEP, Naiad, Spark Streaming, Flink,
Millwheel, Google Dataflow
Data Storage
Distributed
File Systems
GFS, Flat FS
NoSQL Databases
Dynamo, BigTable,
Cassandra
Distributed
Messaging Systems
Kafka
Resource Management
Mesos, YARN

Stream Processing (1/3)

- Stream processing is the act of continuously incorporating new data to compute a result.

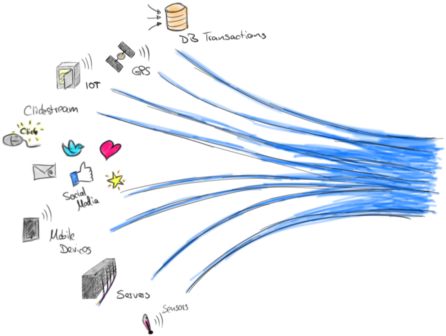
Stream Processing (2/3)

- The input data is unbounded.
    - A series of events, no predetermined beginning or end.



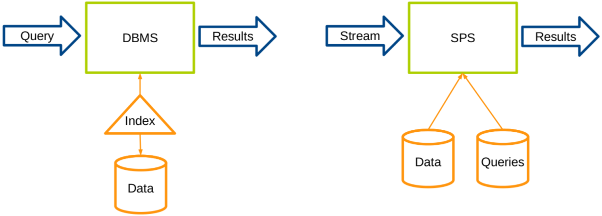
Stream Processing (2/3)

- The input data is unbounded.
  - A series of events, no predetermined beginning or end.
  - E.g., credit card transactions, clicks on a website, or sensor readings from IoT devices.



Stream Processing (3/3)

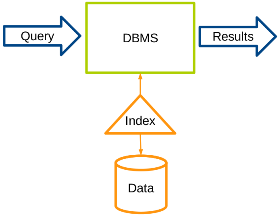
- Database Management Systems (DBMS): data-at-rest analytics
    - Store and index data before processing it.
    - Process data only when explicitly asked by the users.

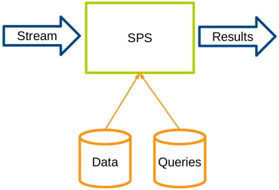


Stream Processing (3/3)

- Database Management Systems (DBMS): data-at-rest analytics
  - Store and index data before processing it.
  - Process data only when explicitly asked by the users.

- Stream Processing Systems (SPS): data-in-motion analytics
  - Processing information as it flows, without storing them persistently.



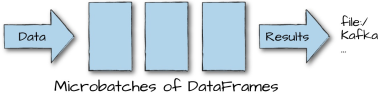


Streaming Data

- Data stream is unbound data, which is broken into a sequence of individual tuples.
- A data tuple is the atomic data item in a data stream.
- Can be structured, semi-structured, and unstructured.

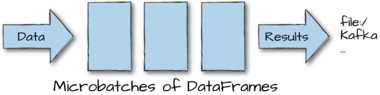
Streaming Processing Patterns

- Micro-batch systems
  - Batch engines
  - Slicing up the unbounded data into a sets of bounded data, then process each batch.

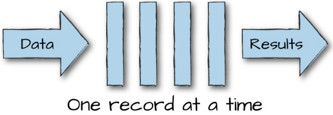


Streaming Processing Patterns

- Micro-batch systems
  - Batch engines
  - Slicing up the unbounded data into a sets of bounded data, then process each batch.



- Continuous processing-based systems
  - Each node in the system continually listens to messages from other nodes and outputs new updates to its child nodes.



Event and Processing Time

Windowing (1/2)

- Window: a buffer associated with an input port to retain previously received tuples.

Windowing (1/2)

- Window: a buffer associated with an input port to retain previously received tuples.
- Different windowing management policies.

Windowing (1/2)

- Window: a buffer associated with an input port to retain previously received tuples.
- Different windowing management policies.
  - Count-based policy: the maximum number of tuples a window buffer can hold

Windowing (1/2)

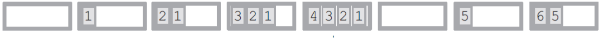
- Window: a buffer associated with an input port to retain previously received tuples.
- Different windowing management policies.
  - Count-based policy: the maximum number of tuples a window buffer can hold
  - Time-based policy: based on processing or event time period

Windowing (2/2)

- Two types of windows: tumbling and sliding

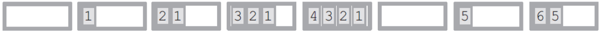
Windowing (2/2)

- Two types of windows: tumbling and sliding
- Tumbling window: supports batch operations.
  - When the buffer fills up, all the tuples are evicted.

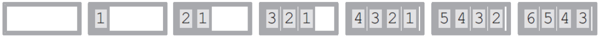


Windowing (2/2)

- Two types of windows: tumbling and sliding
- Tumbling window: supports batch operations.
  - When the buffer fills up, all the tuples are evicted.



- Sliding window: supports incremental operations.
  - When the buffer fills up, older tuples are evicted.



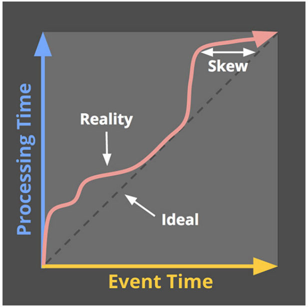
Event Time vs. Processing Time (1/2)

- Event time: the time at which events actually occurred.
  - Timestamps inserted into each record at the source.

- Prcosseing time: the time when the record is received at the streaming application.

Event Time vs. Processing Time (2/2)

- Ideally, event time and processing time should be equal.
- Skew between event time and processing time.



[https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101]

Triggering and Windowing

- Triggering determines when in processing time the results of groupings are emitted as panes.

Triggering and Windowing

- Triggering determines when in processing time the results of groupings are emitted as panes.

- Windowing determines where in event time data are grouped together for processing.

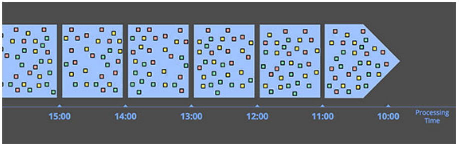
Triggering and Windowing

- Triggering determines when in processing time the results of groupings are emitted as panes.
  - Time-based or data-driven triggers

- Windowing determines where in event time data are grouped together for processing.
  - Time-based or data-driven triggers

Time-based Triggering (Processing Time)

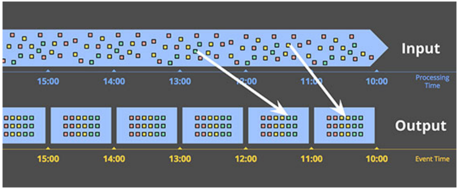
- The system buffers up incoming data into windows until some amount of processing time has passed.
- E.g., five-minute fixed windows



[https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101]

Time-based Windowing (Event Time) (1/3)

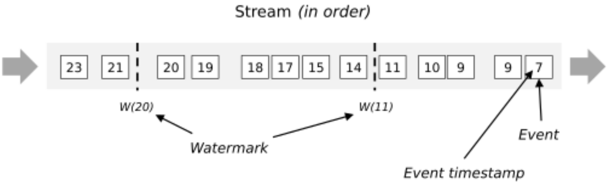
- Reflect the times at which events actually happened.
- Handling out-of-order evnets.



[https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101]

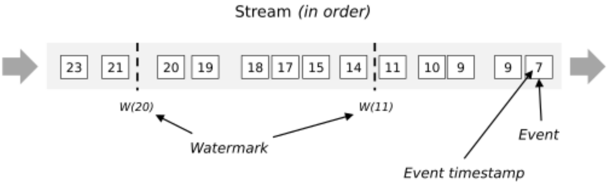
Time-based Windowing (Event Time) (2/3)

- Watermarking helps a stream processing system to deal with lateness.



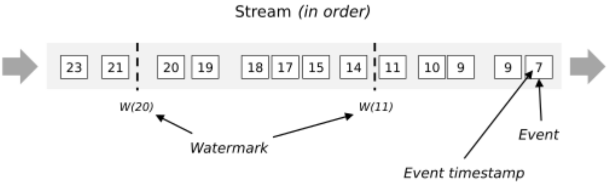
Time-based Windowing (Event Time) (2/3)

- Watermarking helps a stream processing system to deal with lateness.
- Watermarks flow as part of the data stream and carry a timestamp `t`.



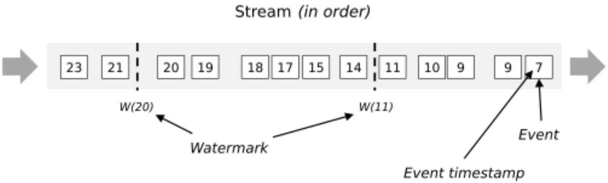
Time-based Windowing (Event Time) (2/3)

- Watermarking helps a stream processing system to deal with lateness.
- Watermarks flow as part of the data stream and carry a timestamp t.
- A watermark is a threshold to specify how long the system waits for late events.



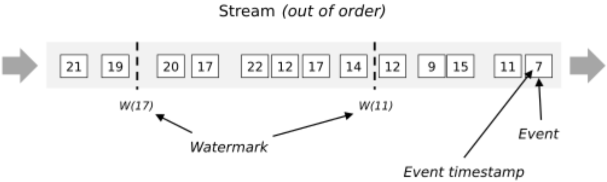
Time-based Windowing (Event Time) (2/3)

- Watermarking helps a stream processing system to deal with lateness.
- Watermarks flow as part of the data stream and carry a timestamp `t`.
- A watermark is a threshold to specify how long the system waits for late events.
- Streaming systems uses watermarks to measure progress in event time.



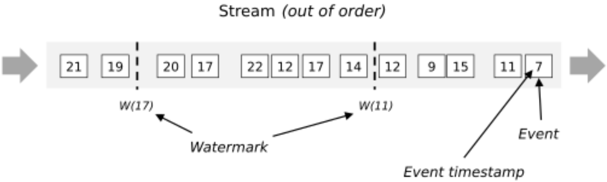
Time-based Windowing (Event Time) (3/3)

- A `W(t)` declares that event time has reached time `t` in that stream
  - There should be no more elements from the stream with a timestamp $t' \leq t$.



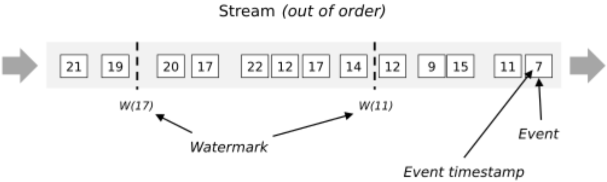
Time-based Windowing (Event Time) (3/3)

- A `W(t)` declares that event time has reached time `t` in that stream
  - There should be no more elements from the stream with a timestamp $t' \leq t$.
- It is possible that certain elements will violate the watermark condition.
  - After the $W(t)$ has occurred, more elements with timestamp $t' \leq t$ will occur.



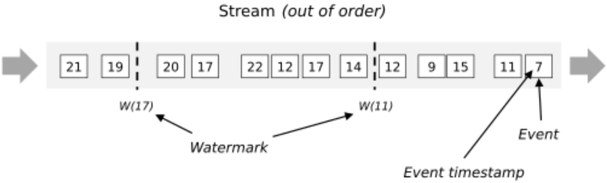
Time-based Windowing (Event Time) (3/3)

- A `W(t)` declares that event time has reached time `t` in that stream
  - There should be no more elements from the stream with a timestamp $t' \leq t$.
- It is possible that certain elements will violate the watermark condition.
  - After the `W(t)` has occurred, more elements with timestamp $t' \leq t$ will occur.
- If an arriving event lies within the watermark, it gets used to update a query.



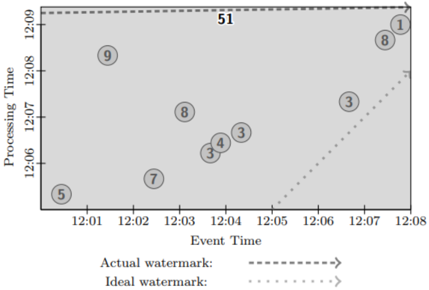
Time-based Windowing (Event Time) (3/3)

- A `W(t)` declares that event time has reached time `t` in that stream
  - There should be no more elements from the stream with a timestamp $\texttt{t}' \leq \texttt{t}$.
- It is possible that certain elements will violate the watermark condition.
  - After the `W(t)` has occurred, more elements with timestamp $\texttt{t}' \leq \texttt{t}$ will occur.
- If an arriving event lies within the watermark, it gets used to update a query.
- Streaming programs may explicitly expect some late elements.



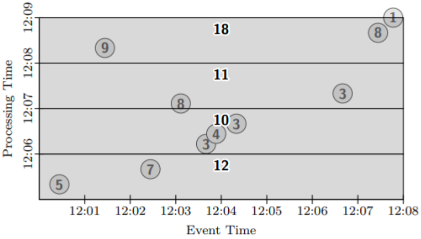
Winowing and Triggering - Example (1/3)

- Batch processing



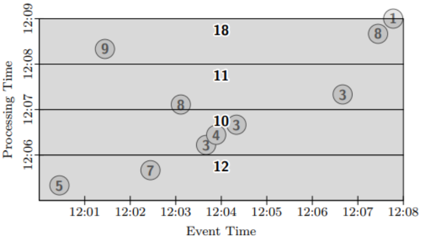
Winowing and Triggering - Example (2/3)

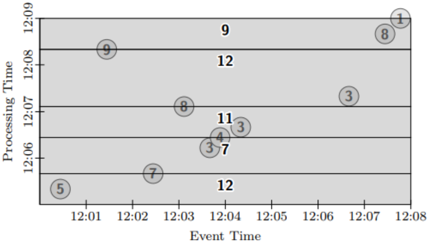
- Trigger at period (time-based triggers)



Winowing and Triggering - Example (2/3)

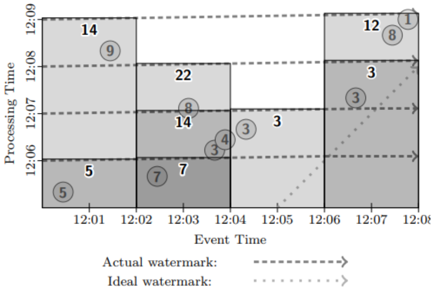
- Trigger at period (time-based triggers)
- Trigger at count (data-driven triggers)





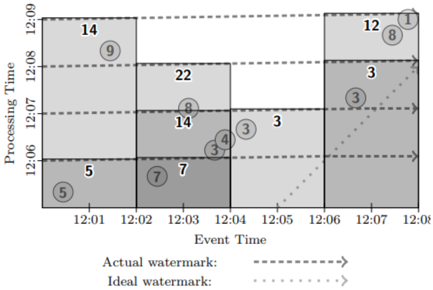
Winowing and Triggering - Example (3/3)

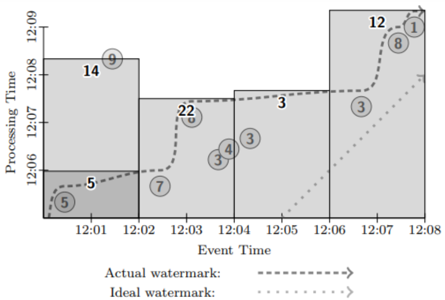
- Fixed window, trigger at period (micro-batch)



Winowing and Triggering - Example (3/3)

- Fixed window, trigger at period (micro-batch)
- Fixed window, trigger at watermark (streaming)

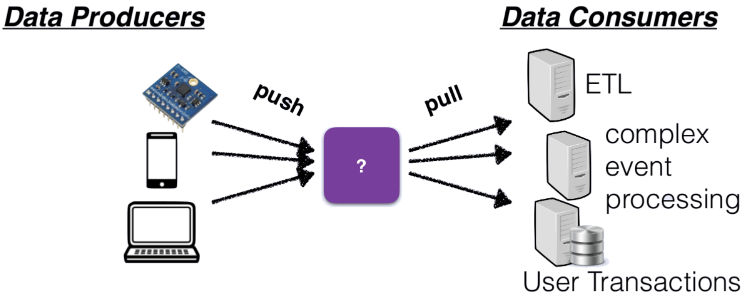




Data Stream Storage

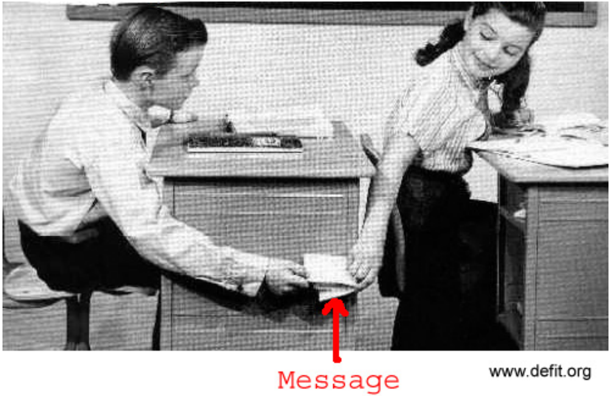
The Problem

- We need disseminate streams of events from various producers to various consumers.



Possible Solution?

- Messaging systems



What is Messaging System?

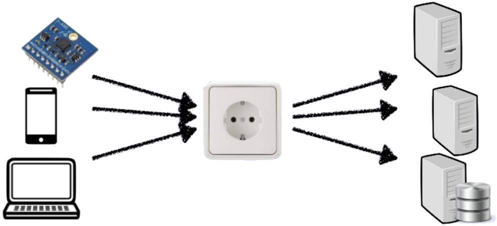
- Messaging system is an approach to notify consumers about new events.

What is Messaging System?

- Messaging system is an approach to notify consumers about new events.
- Messaging systems
  - Direct messaging
  - Message brokers

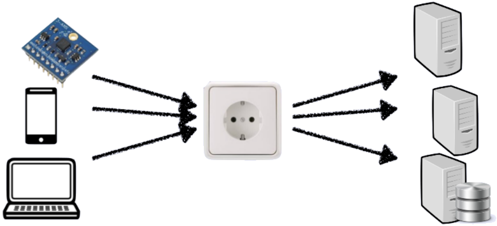
Direct Messaging (1/2)

- Necessary in latency critical applications (e.g., remote surgery).
- A producer sends a message containing the event, which is pushed to consumers.



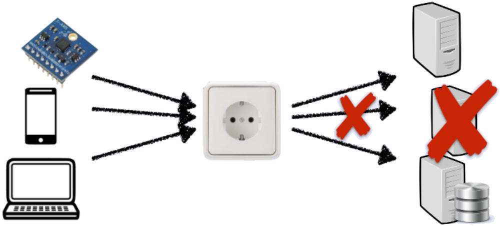
Direct Messaging (1/2)

- Necessary in latency critical applications (e.g., remote surgery).
- A producer sends a message containing the event, which is pushed to consumers.
- Both consumers and producers have to be online at the same time.



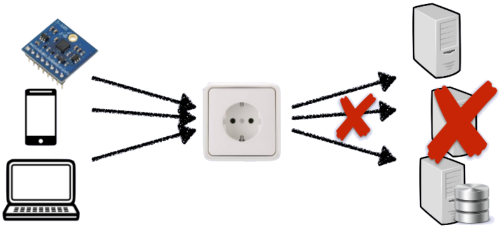
Direct Messaging (2/2)

- What happens if a consumer crashes or temporarily goes offline? (not durable)



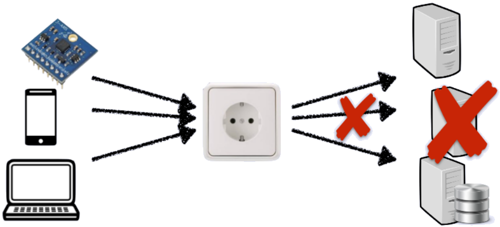
Direct Messaging (2/2)

- What happens if a consumer crashes or temporarily goes offline? (not durable)
- What happens if producers send messages faster than the consumers can process?



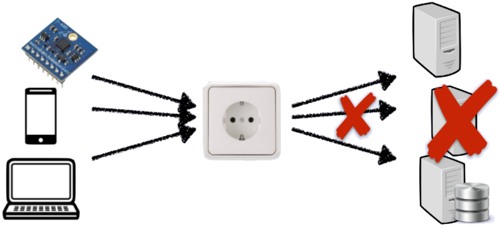
Direct Messaging (2/2)

- What happens if a consumer crashes or temporarily goes offline? (not durable)
- What happens if producers send messages faster than the consumers can process?
  - Dropping messages
  - Backpressure

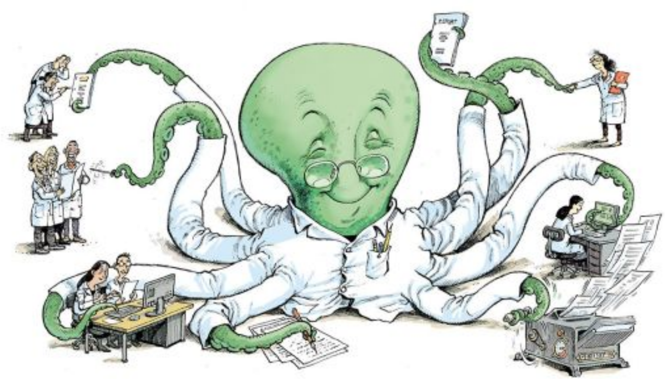


Direct Messaging (2/2)

- What happens if a consumer crashes or temporarily goes offline? (not durable)
- What happens if producers send messages faster than the consumers can process?
  - Dropping messages
  - Backpressure
- We need message brokers that can log events to process at a later time.



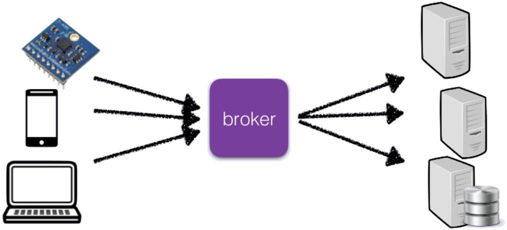
Message Broker



[https://bluesyemre.com/2018/10/16/thousands-of-scientists-publish-a-paper-every-five-days]

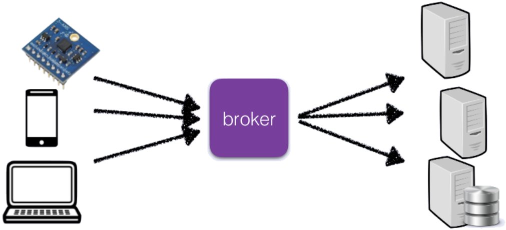
Message Broker

- A message broker decouples the producer-consumer interaction.
- It runs as a server, with producers and consumers connecting to it as clients.



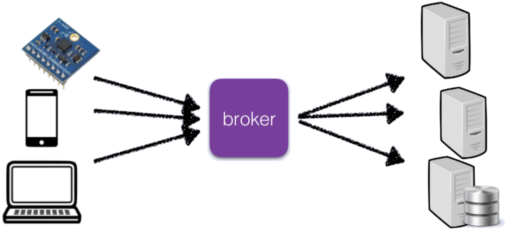
Message Broker

- A message broker decouples the producer-consumer interaction.
- It runs as a server, with producers and consumers connecting to it as clients.
- Producers write messages to the broker, and consumers receive them by reading them from the broker.



Message Broker

- A message broker decouples the producer-consumer interaction.
- It runs as a server, with producers and consumers connecting to it as clients.
- Producers write messages to the broker, and consumers receive them by reading them from the broker.
- Consumers are generally asynchronous.



Partitioned Logs

- In typical message brokers, once a message is consumed, it is deleted.

Partitioned Logs

- In typical message brokers, once a message is consumed, it is deleted.
- Log-based message brokers durably store all events in a sequential log.

Partitioned Logs

- In typical message brokers, once a message is consumed, it is deleted.
- Log-based message brokers durably store all events in a sequential log.
- A log is an append-only sequence of records on disk.

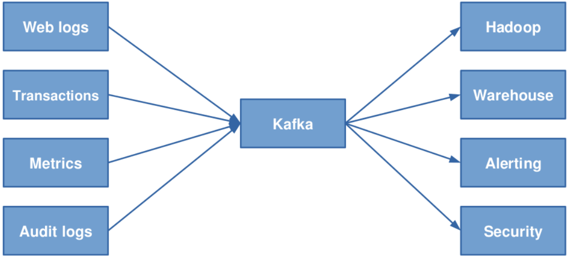
Partitioned Logs

- In typical message brokers, once a message is consumed, it is deleted.
- Log-based message brokers durably store all events in a sequential log.
- A log is an append-only sequence of records on disk.
- A producer sends a message by appending it to the end of the log.
- A consumer receives messages by reading the log sequentially.

Kafka - A Log-Based Message Broker

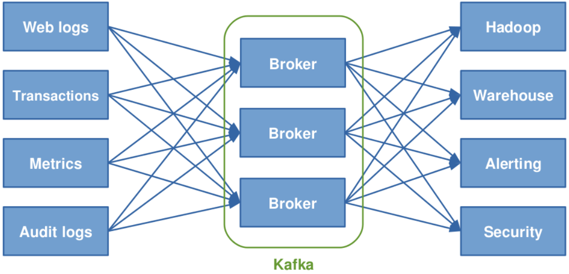
Kafka (1/5)

- Kafka is a distributed, topic oriented, partitioned, replicated commit log service.



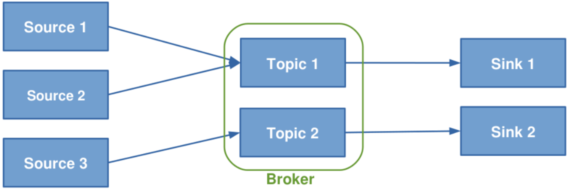
Kafka (2/5)

- Kafka is a distributed, topic oriented, partitioned, replicated commit log service.



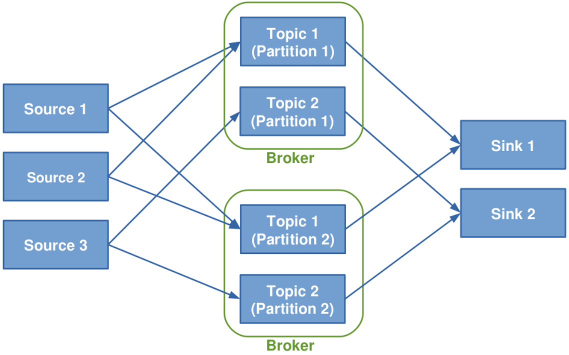
Kafka (3/5)

- Kafka is a distributed, topic oriented, partitioned, replicated commit log service.



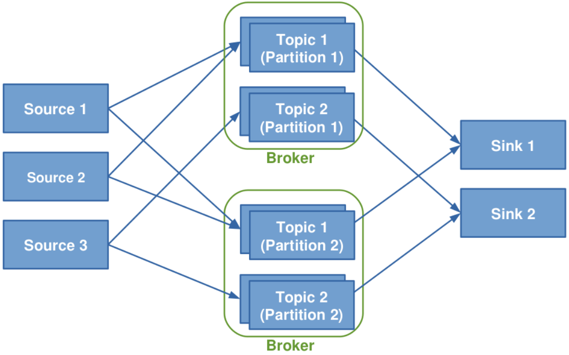
Kafka (4/5)

- Kafka is a distributed, topic oriented, partitioned, replicated commit log service.



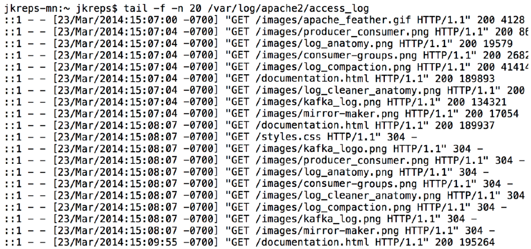
Kafka (5/5)

- Kafka is a distributed, topic oriented, partitioned, replicated commit log service.

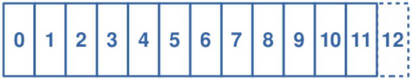


Logs, Topics and Partition (1/6)

- Kafka is about logs.

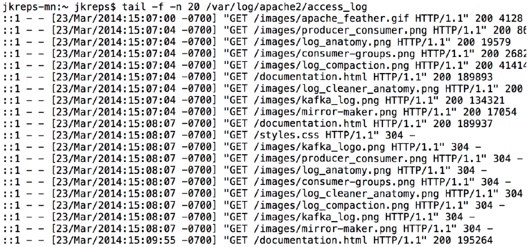


```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 86
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 2682
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 41414
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```

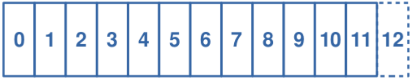


Logs, Topics and Partition (1/6)

- Kafka is about logs.
- Topics are queues: a stream of messages of a particular type

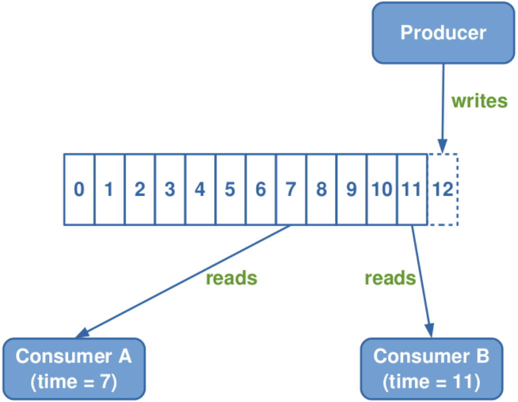


```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 86
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 268
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 4141
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```



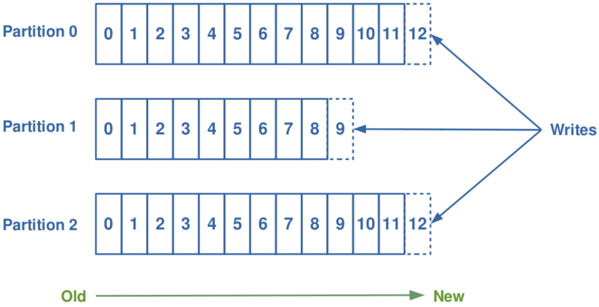
Logs, Topics and Partition (2/6)

- Each message is assigned a sequential id called an offset.



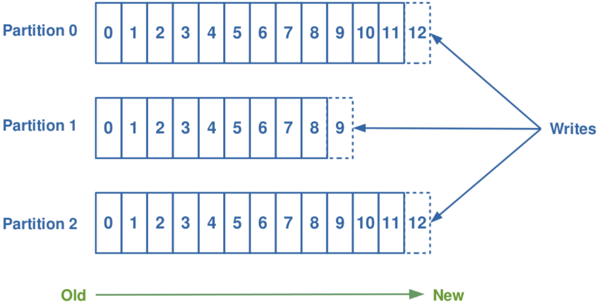
Logs, Topics and Partition (3/6)

- Topics are logical collections of partitions (the physical files).



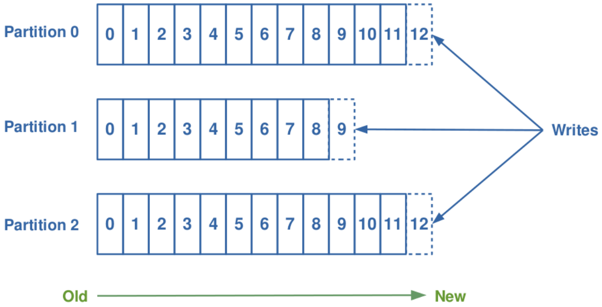
Logs, Topics and Partition (3/6)

- Topics are logical collections of partitions (the physical files).
  - Ordered
  - Append only
  - Immutable



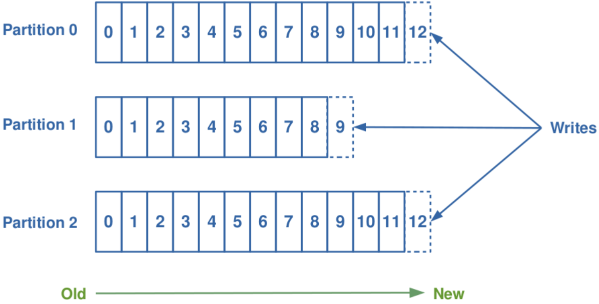
Logs, Topics and Partition (4/6)

- Ordering is only guaranteed within a partition for a topic.



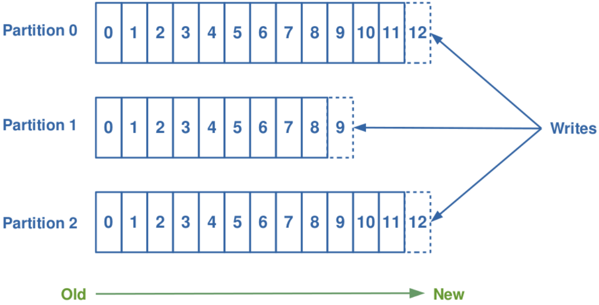
Logs, Topics and Partition (4/6)

- Ordering is only guaranteed within a partition for a topic.
- Messages sent by a producer to a particular topic partition will be appended in the order they are sent.



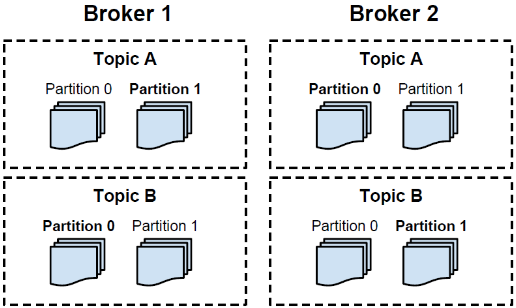
Logs, Topics and Partition (4/6)

- Ordering is only guaranteed within a partition for a topic.
- Messages sent by a producer to a particular topic partition will be appended in the order they are sent.
- A consumer instance sees messages in the order they are stored in the log.



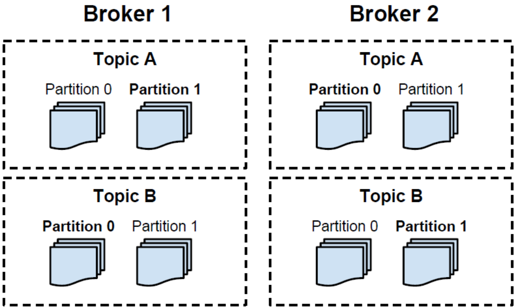
Logs, Topics and Partition (5/6)

- Partitions of a topic are replicated: fault-tolerance



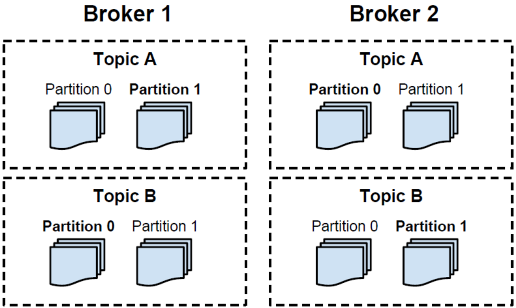
Logs, Topics and Partition (5/6)

- Partitions of a topic are replicated: fault-tolerance
- A broker contains some of the partitions for a topic.

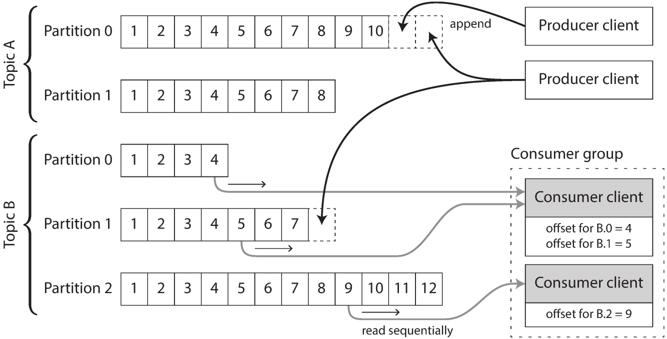


Logs, Topics and Partition (5/6)

- Partitions of a topic are replicated: fault-tolerance
- A broker contains some of the partitions for a topic.
- One broker is the leader of a partition: all writes and reads must go to the leader.



Partitioned Logs (6/6)



WESTCLOX

Go to www.menti.com, and use the code 2977 7833

- Kafka maintains feeds of messages in categories called?

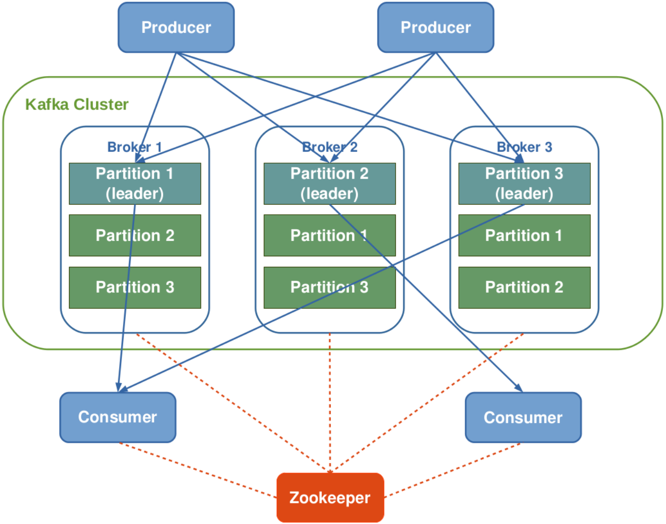
1. Chunks
2. Topic
3. Domain
4. Message

Go to www.menti.com, and use the code 1437 1825

- Kafka only provides a ___ order over messages within a partition and among partitions?

1. Partial, partial
2. Partial, total
3. Total, partial
4. Total, total

Kafka Architecture



Coordination

- Kafka uses Zookeeper for the following tasks:

Coordination

- Kafka uses Zookeeper for the following tasks:
- Detecting the addition and the removal of brokers and consumers.
- Keeping track of the consumed offset of each partition.

State in Kafka

- Brokers are sateless: no metadata for consumers-producers in brokers.

State in Kafka

- Brokers are sateless: no metadata for consumers-producers in brokers.
- Consumers are responsible for keeping track of offsets.

State in Kafka

- Brokers are sateless: no metadata for consumers-producers in brokers.
- Consumers are responsible for keeping track of offsets.
- Messages in queues expire based on pre-configured time periods (e.g., once a day).

Delivery Guarantees

- Kafka guarantees that messages from a single partition are delivered to a consumer in order.

Delivery Guarantees

- Kafka guarantees that messages from a single partition are delivered to a consumer in order.
- There is no guarantee on the ordering of messages coming from different partitions.

Delivery Guarantees

- Kafka guarantees that messages from a single partition are delivered to a consumer in order.
- There is no guarantee on the ordering of messages coming from different partitions.
- Kafka only guarantees at-least-once delivery.

Start and Work With Kafka

```
# Start the ZooKeeper
zookeeper-server-start.sh config/zookeeper.properties
```

Start and Work With Kafka

```
# Start the ZooKeeper
zookeeper-server-start.sh config/zookeeper.properties
```

```
# Start the Kafka server
kafka-server-start.sh config/server.properties
```

Start and Work With Kafka

```
# Start the ZooKeeper
zookeeper-server-start.sh config/zookeeper.properties
```

```
# Start the Kafka server
kafka-server-start.sh config/server.properties
```

```
# Create a topic, called "avg"
kafka-topics.sh --create --topic avg --bootstrap-server localhost:9092 --replication-factor 1
                        --partitions 1
```

Start and Work With Kafka

```
# Start the ZooKeeper
zookeeper-server-start.sh config/zookeeper.properties
```

```
# Start the Kafka server
kafka-server-start.sh config/server.properties
```

```
# Create a topic, called "avg"
kafka-topics.sh --create --topic avg --bootstrap-server localhost:9092 --replication-factor 1
                        --partitions 1
```

```
# Produce messages and send them to the topic "avg"
kafka-console-producer.sh --topic avg --bootstrap-server localhost:9092
```

Start and Work With Kafka

```
# Start the ZooKeeper
zookeeper-server-start.sh config/zookeeper.properties
```

```
# Start the Kafka server
kafka-server-start.sh config/server.properties
```

```
# Create a topic, called "avg"
kafka-topics.sh --create --topic avg --bootstrap-server localhost:9092 --replication-factor 1
                        --partitions 1
```

```
# Produce messages and send them to the topic "avg"
kafka-console-producer.sh --topic avg --bootstrap-server localhost:9092
```

```
# Consume the messages sent to the topic "avg"
kafka-console-consumer.sh --topic avg --from-beginning --bootstrap-server localhost:9092
```

Summary

Summary

- SPS vs. DBMS
- Data stream, unbounded data, tuples
- Event-time vs. processing time
- Windowing and triggering

Summary

- Messaging system and partitioned logs
- Decoupling producers and consumers
- Kafka: distributed, topic oriented, partitioned, replicated log service
- Logs, topcs, partition
- Kafka architecture: producer, consumer, broker, coordinator

References

- J. Kreps et al., “Kafka: A distributed messaging system for log processing”, NetDB 2011
- M. Zaharia et al., “Spark: The Definitive Guide”, O’Reilly Media, 2018 - Chapter 20
- T. Akidau et al., “The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing”, VLDB 2015.
- M. Fragkoulis et al., “A Survey on the Evolution of Stream Processing Systems”, 2020
- T. Akidau, “The world beyond batch: Streaming 101”, `https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101`

Questions?