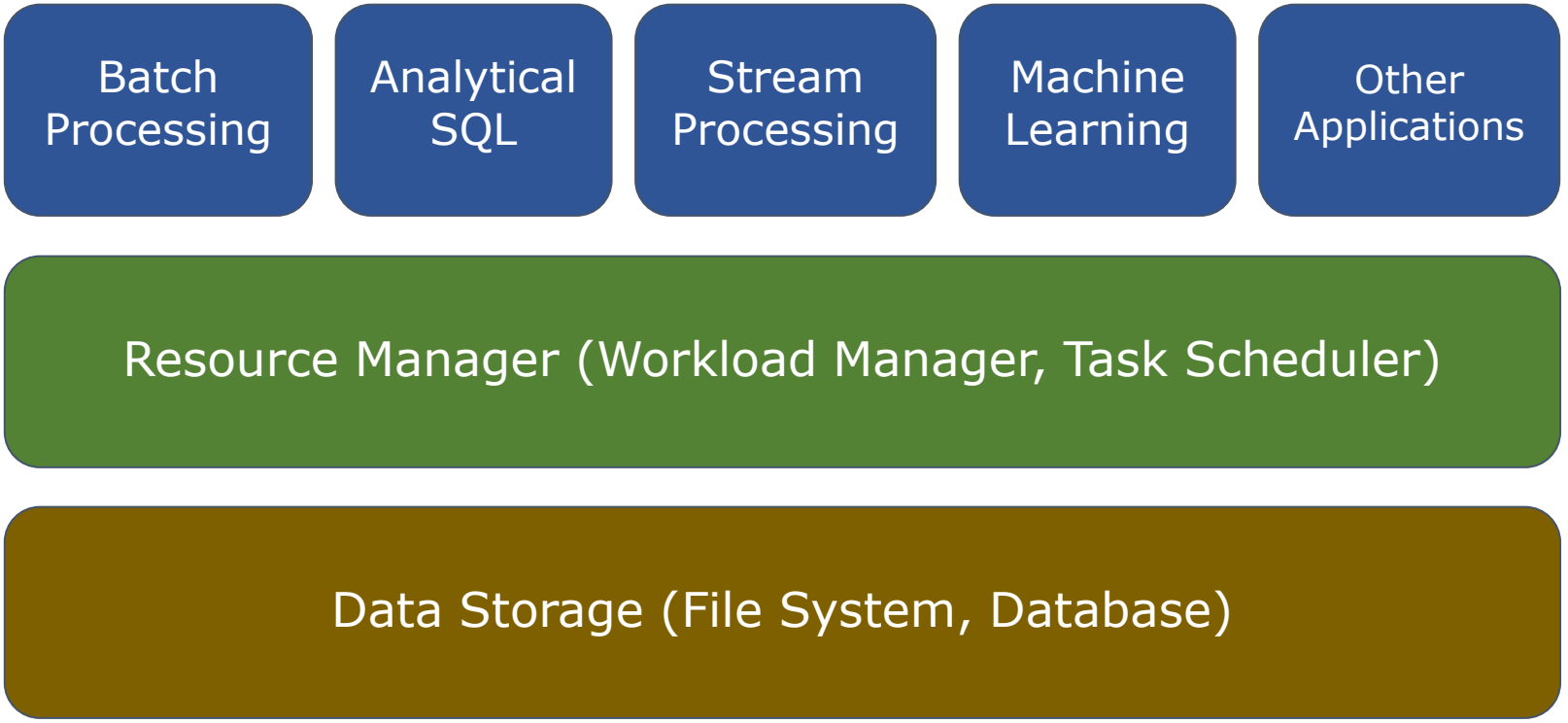
Stream Processing

Ashvin Goel

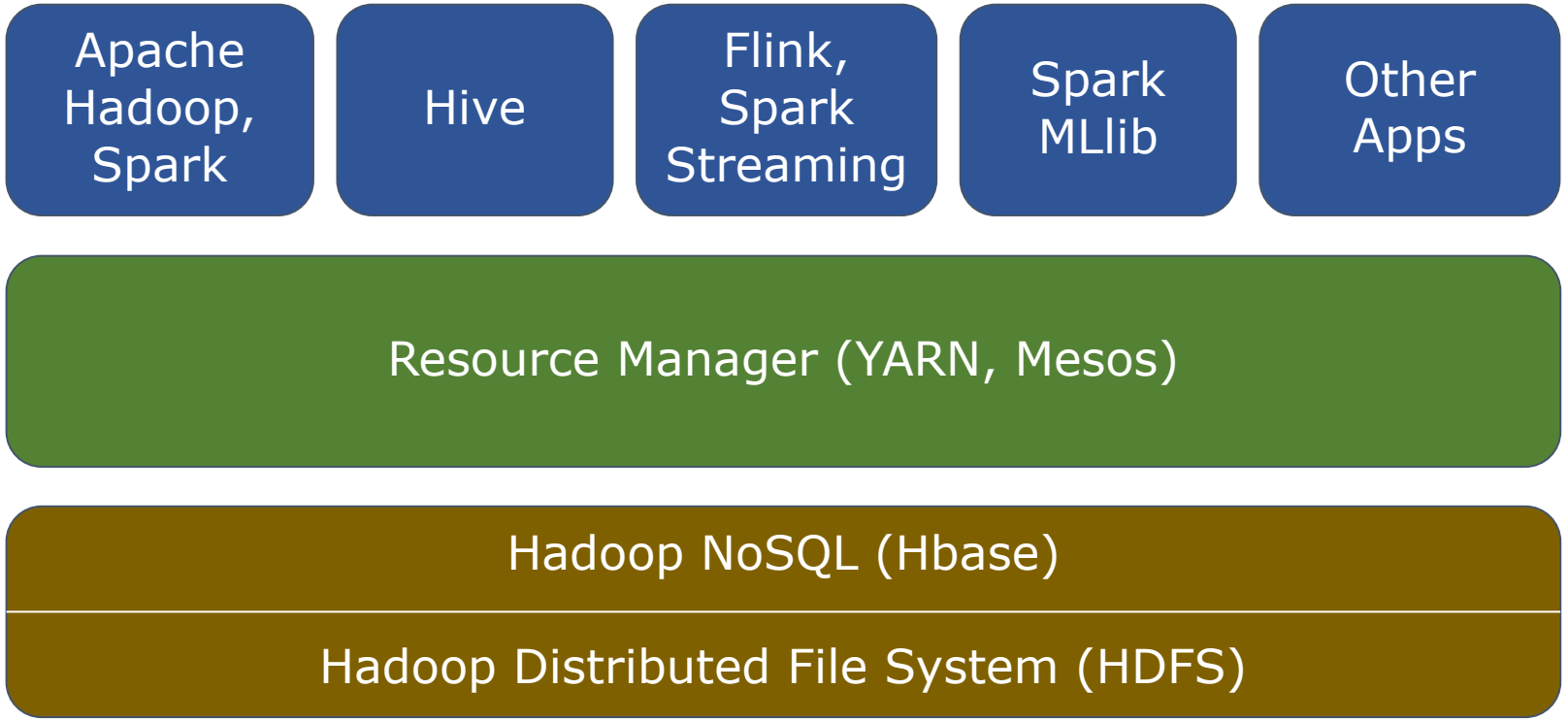
Electrical and Computer Engineering
University of Toronto

ECE1724

A Typical Big Data System



Open-Source Apache Ecosystem



Background

- Real world has physical objects
  - E.g., users, thermostat, trains, …
- These objects have state
  - E.g., user has shipping address, thermostat has on/off state, train has cargo, …
- Traditionally
  - This state was stored in database
  - Users ran queries on the state, e.g., return thermostat state
  - Worked well for decades
- But what if we want to know about state changes?
  - How often did thermostat turn on today?

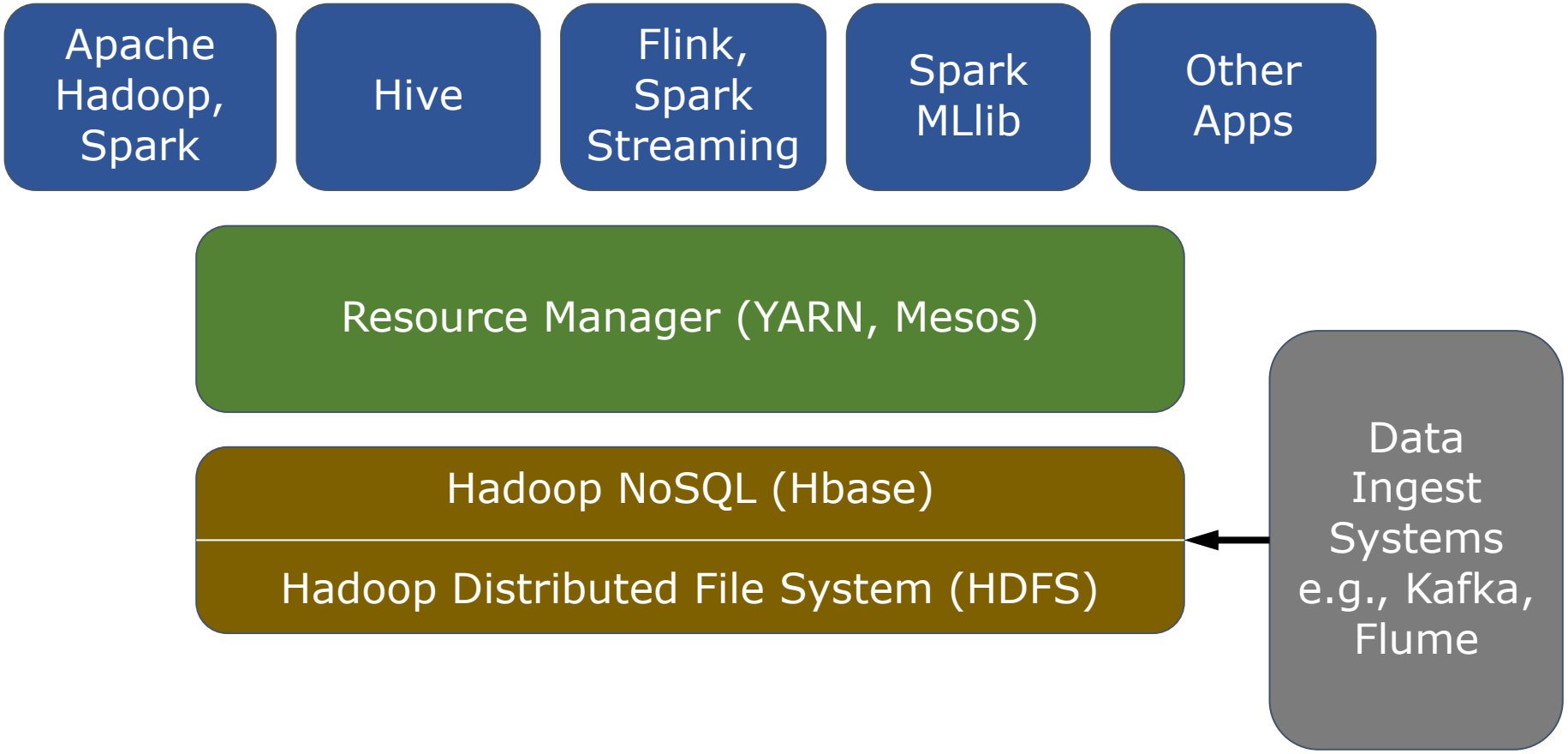
What is Stream Processing?

- Rather than thinking in terms of state, why not think in terms of events
  - E.g., user updates shipping address, thermostat reports it has turned on, train unloads cargo, …
  - Easy to answer: “how often did thermostat turn on today?”
    - Just count the number of such events generated today
- Stream processing refers to storing and processing streams of data events

Storing Streaming Data

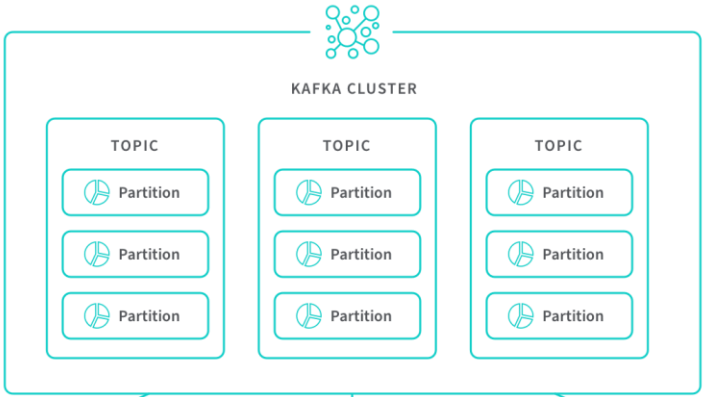
- Each event also has state
  - E.g., user's shipping address and time when it was changed
- How should this state be ingested and stored?
- Problem: Traditional databases are not designed for ingesting high volume real-time streaming data
  - Each update requires significant processing, e.g., index update
- Solution: Ingest data in a log, process it later

Data Ingestion



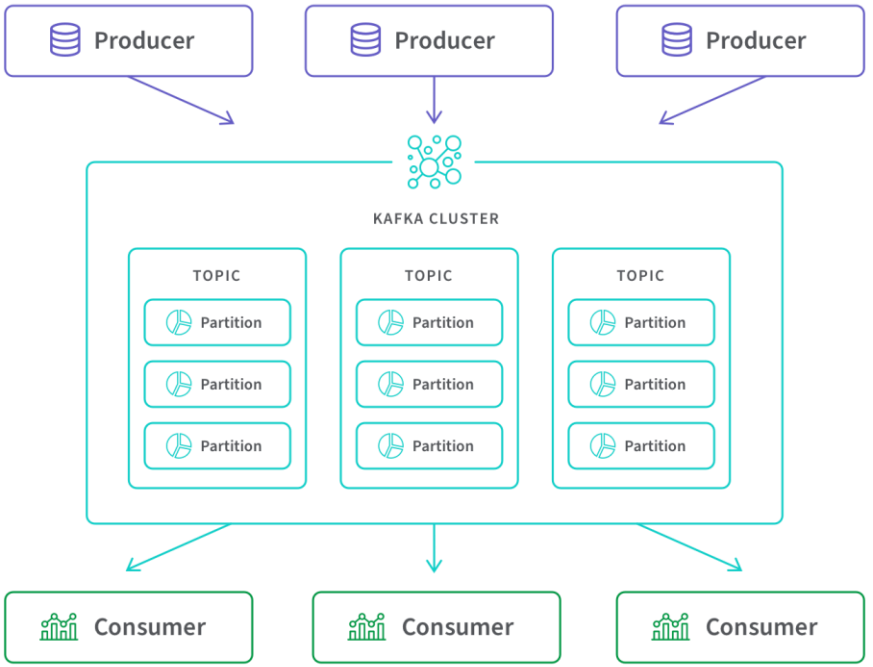
Kafka – In a Nutshell

- Kafka is a distributed, replicated logging service
  - A log is called a topic
  - A Kafka cluster stores many topics, each arbitrary size
- A topic is an ordered sequence of events
  - Partitioned across multiple nodes for scalability
  - Replicated and stored on disk for durability



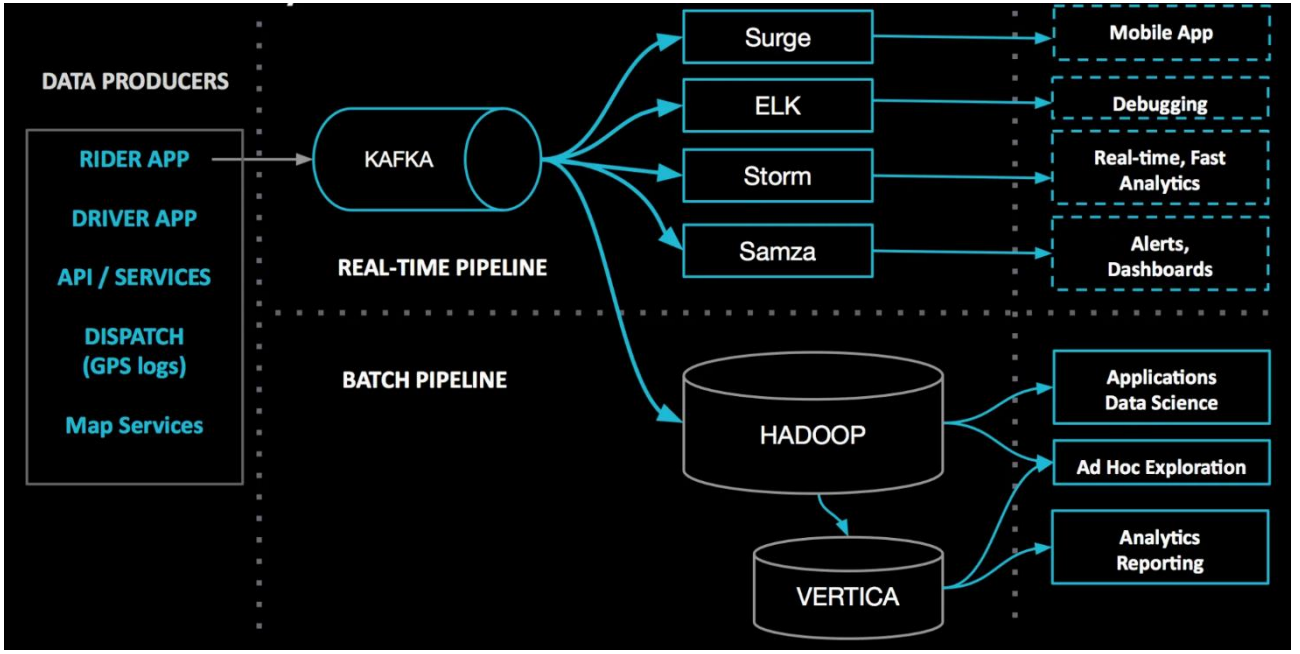
Kafka Stream Processing

- Processing in Kafka involves services (computations) that produce and consume data
  - Producers add data into topics
  - Consumers read data from topics
- E.g., a consumer service can generate data for a real-time dashboard



Stream Processing Pipelines

- Many open-source stream processing applications available today, including Storm, Flink, Samza …
  - Kakfa used for logging real-world data
- Processing pipeline at Uber

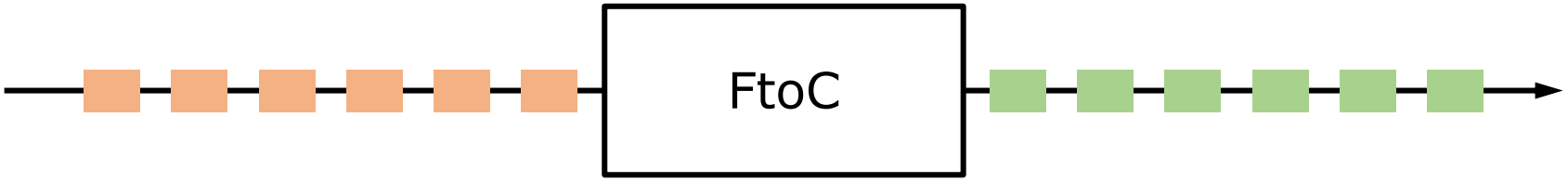


Stream Processing Applications

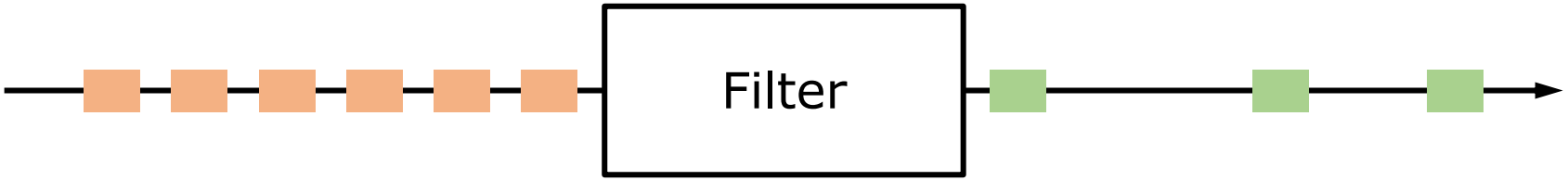
- Applications often perform database-type operations on unbounded data streams
- Operations can be stateless or stateful

Stateless Operations

- Conversion
  - E.g., convert Fahrenheit to Celcius
  - **emit** (input * 5/9) - 32

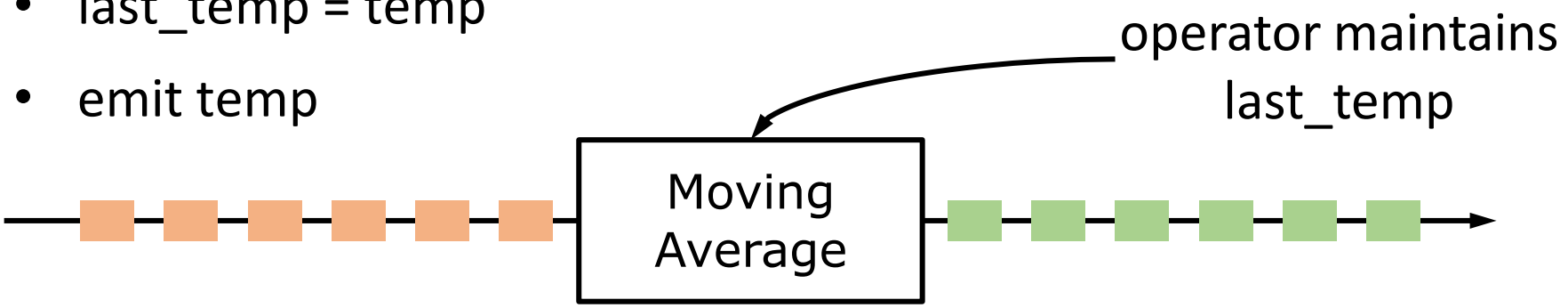


- Filter
  - if (input > threshold) **emit** input

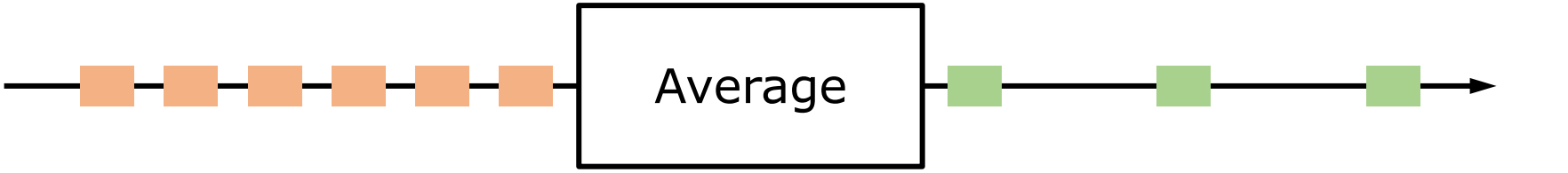


Stateful Operations

- Stateful conversion
  - temp = a * ((input * 5/9) – 32) + (1 – a) * last_temp
  - last_temp = temp
  - emit temp

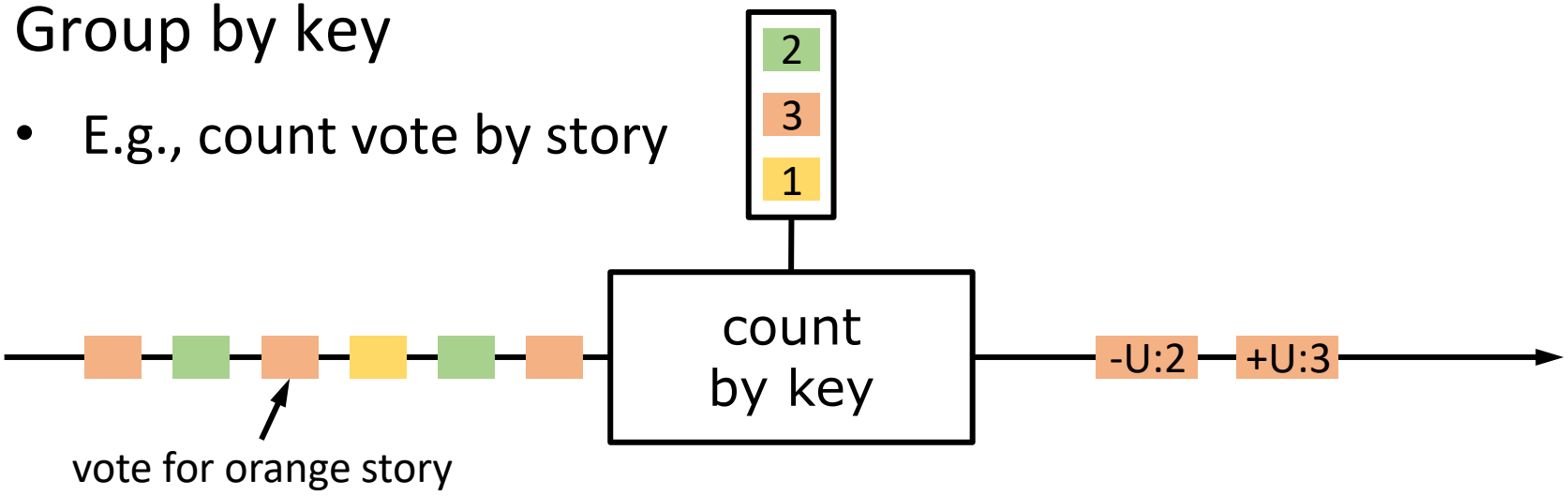


- Aggregation, e.g., average per window
  - Window can be in elements (10), time (1s)
  - Window can be disjoint (5s) or sliding (5s window every 1s)

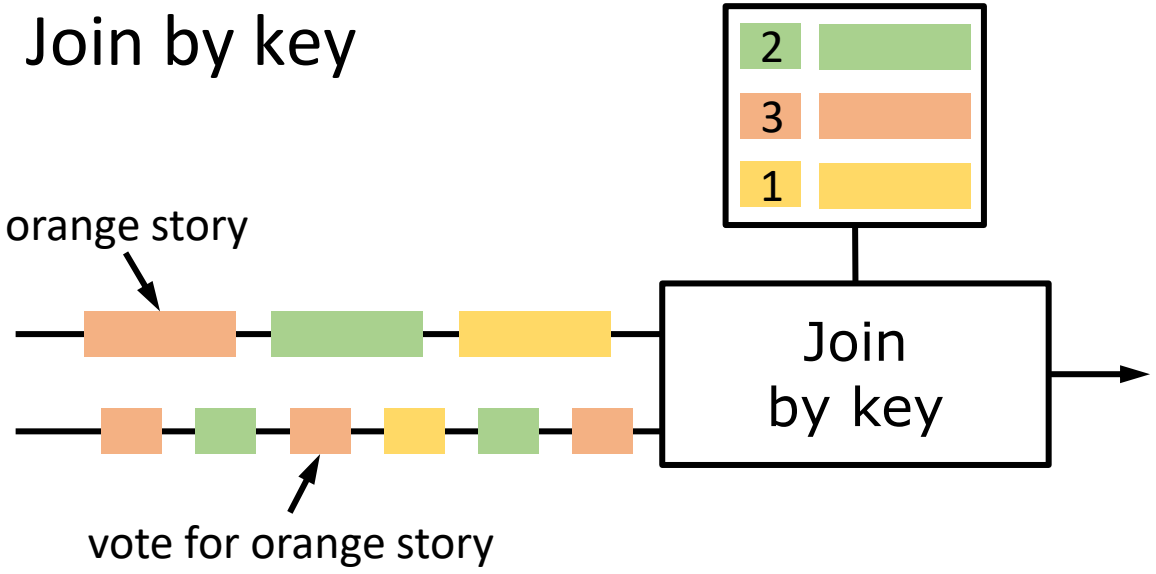


Stateful Operations By Keys

- Group by key
  - E.g., count vote by story

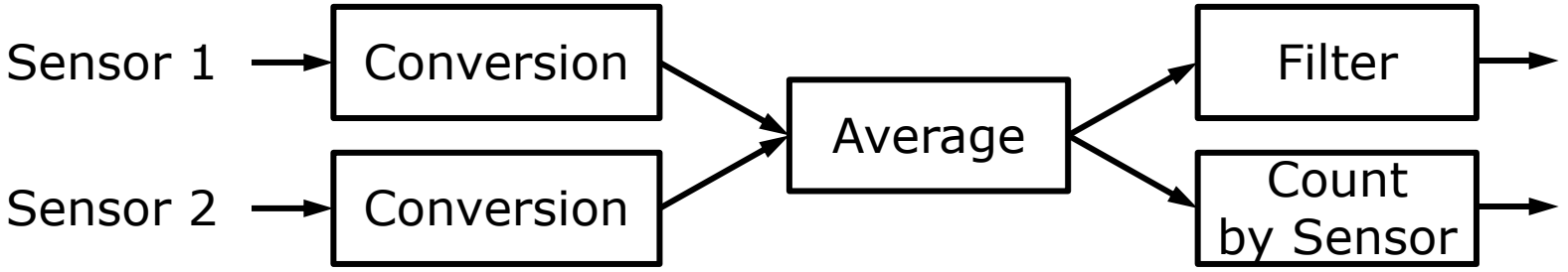


- Join by key



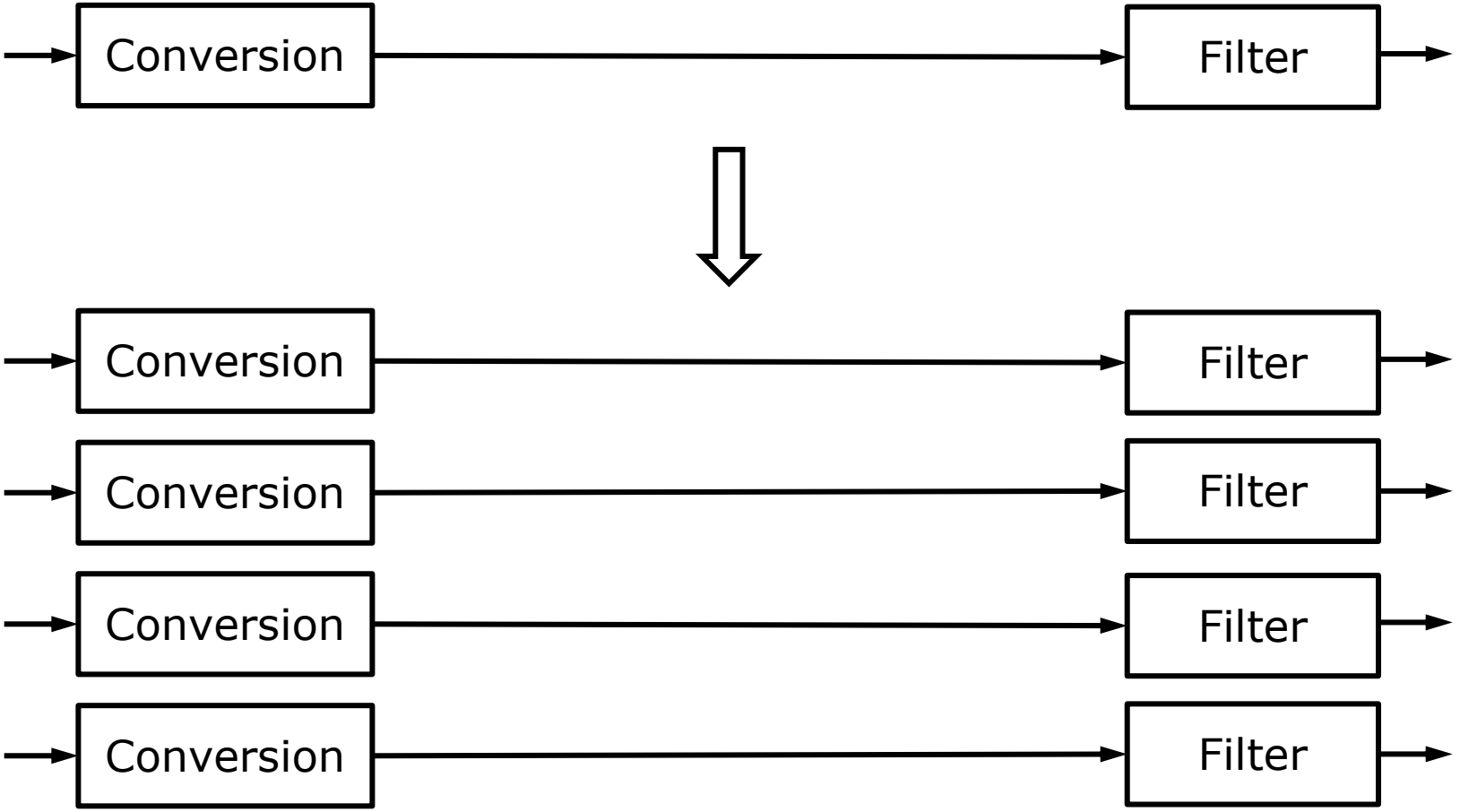
Stream Processing Dataflow

- A stream processing application consists of operations connected in a directed graph, processing data in a dataflow manner



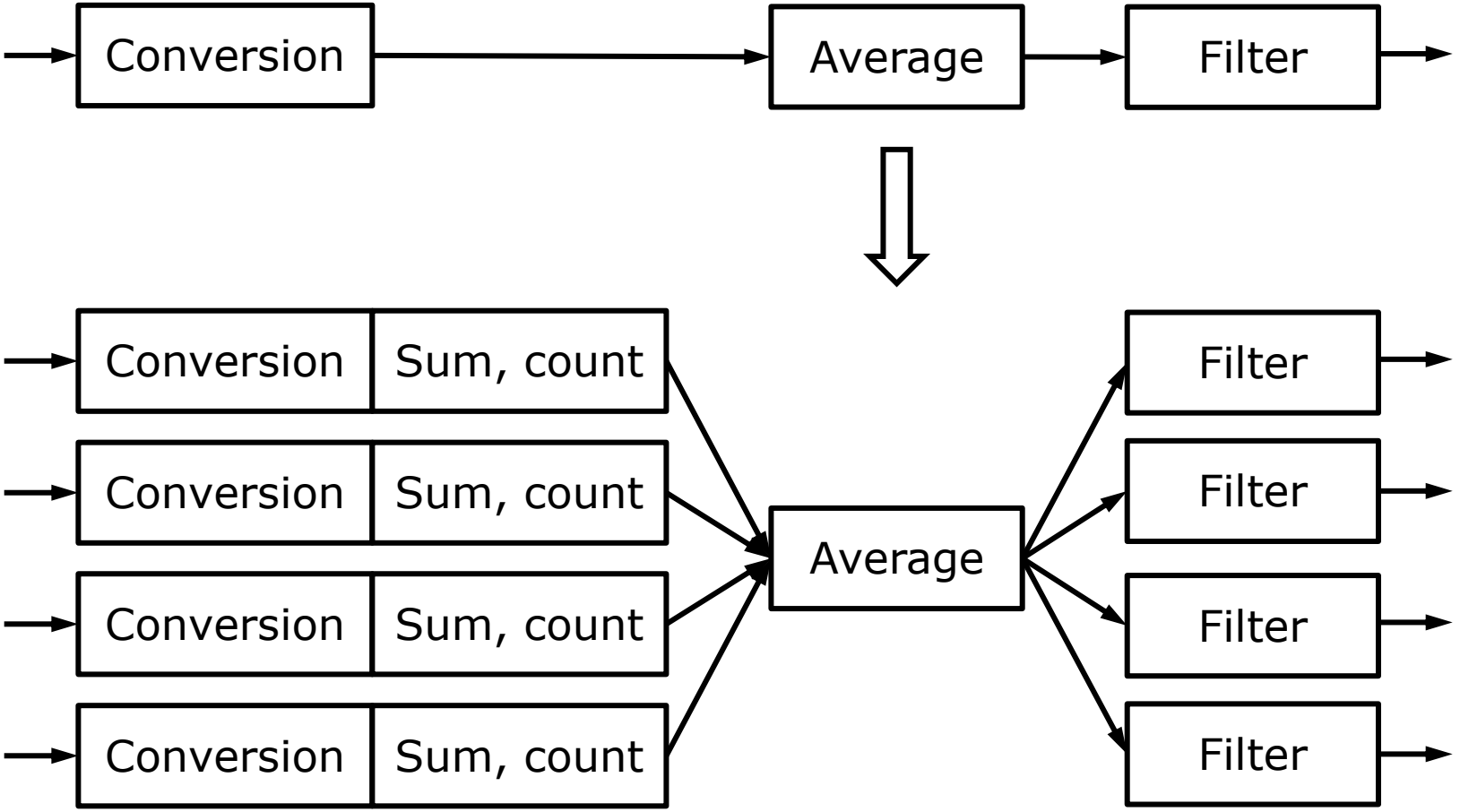
Scalable Processing – Stateless Ops

- Simple to parallelize stateless operations
  - Partition the inputs



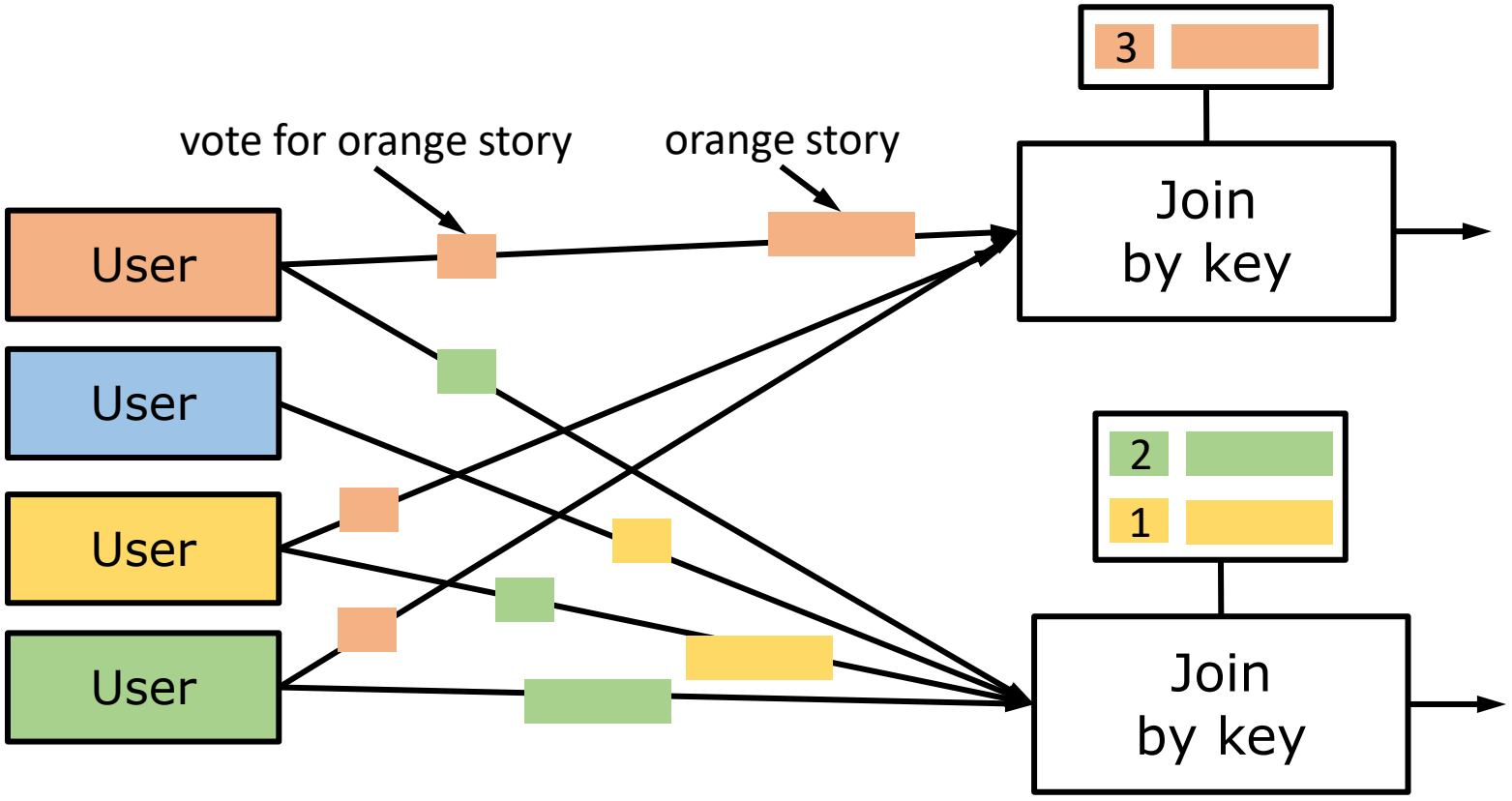
Scalable Processing – Stateful Ops

- Stateful operations complicate parallelization
  - Need to join results across parallel computations



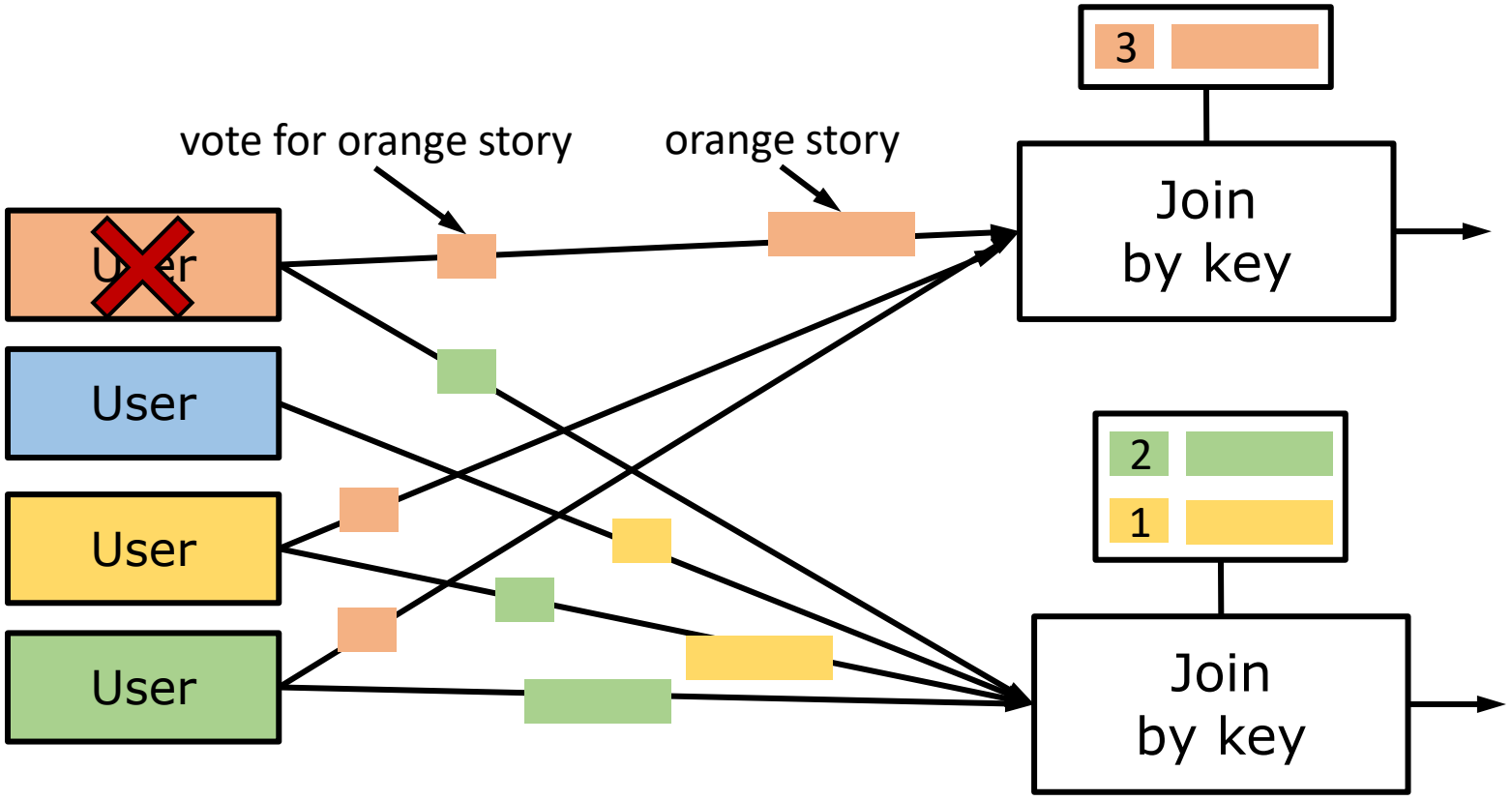
Scalable Aggregation by Keys

- Aggregation operations by keys can be parallelized by partitioning by key



Scalable Processing Complicates Fault Tolerance

- How to ensure exactly-once semantics?



Stream Processing Requirements

- Process data with low end-to-end latency
  - End-to-end latency: from when data is generated to when it is fully processed
- Handle data that arrives out-of-order
  - Real-time data may be delayed, dropped
- Exactly-once processing semantics
  - Ensure that each event is processed once by each computation (even under failures)
- Scalable storage and processing
- Reliability and fault tolerance

Today's Papers

- Millwheel
  - Describes motivation for streaming applications
  - Describes programming model for streaming applications
  - Early system providing exactly-once semantics
  - Today, part of Google Cloud Dataflow

- Noria
  - Websites often cache results obtained from streaming databases
  - How should these caches be kept up-to-date efficiently?