CS 7172
Parallel and Distributed Computation

Spark

Kun Suo

Computer Science, Kennesaw State University

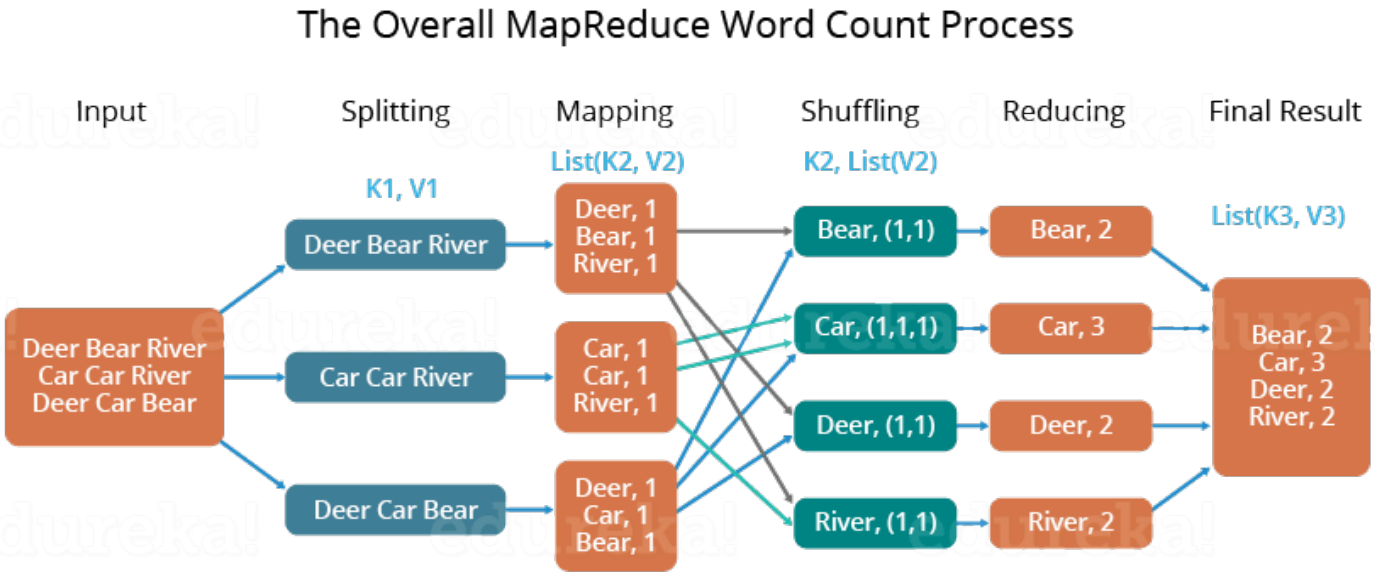
https://kevinsuo.github.io/

Outline

- Computer networks, primarily from an application perspective
- Protocol layering
- Client-server architecture
- End-to-end principle
- TCP
- Socket programming

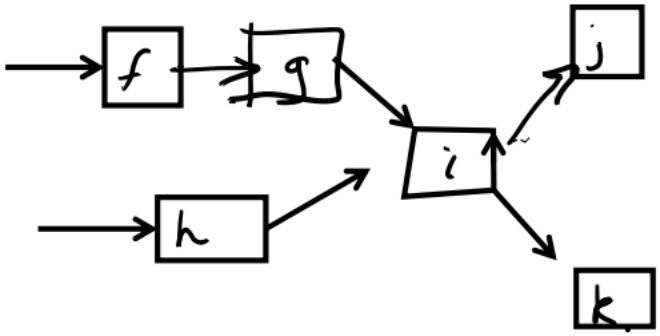
MapReduce

- MapReduce: Divide the large task into multiple small tasks, and calculate separately for these small tasks, and then combine the results of each small task to get the calculation result of the large task



MapReduce Limitations

- Static data
  - Restrictive programming model
  - No support for dataflows (not dynamic)
  - Purely batch processing (not flow)
  - Jobs can take hours to complete
  - No streaming, interactive analytics, real time requirement



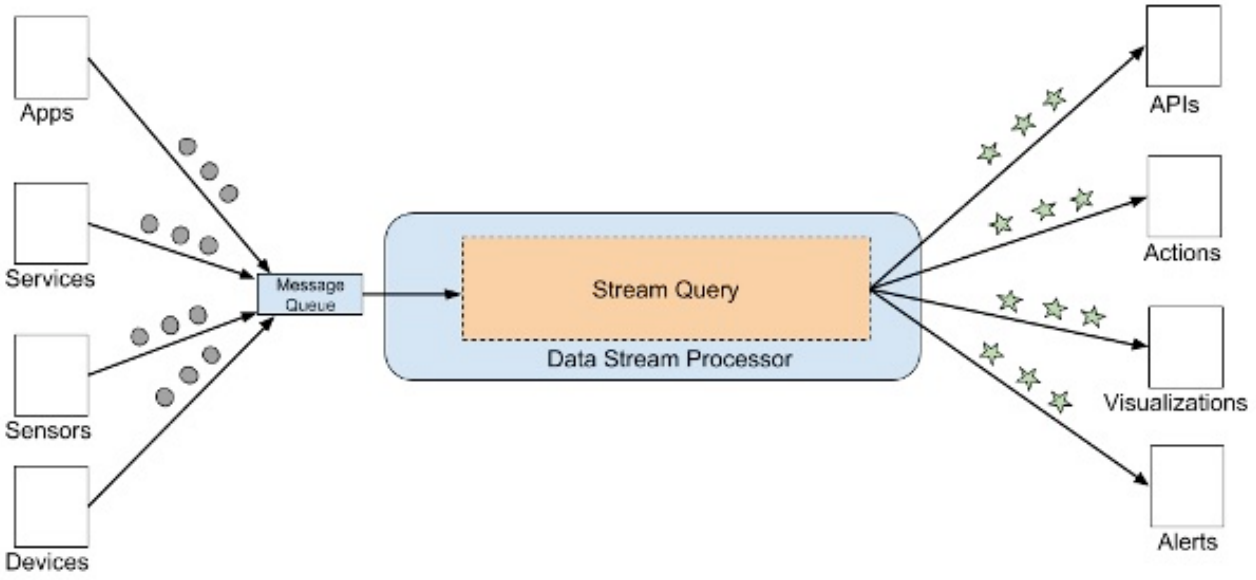
Streaming Applications

- Rise of real-time applications, such as network monitoring, sensor monitoring, AR / VR, audio and video data stream from live broadcast

- Keywords: stream, real-time

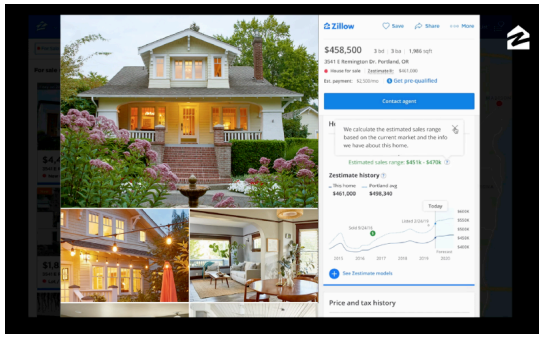
Streaming Data

- Streaming Data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes



Streaming Data Examples

- Sensors in transportation vehicles, industrial equipment, and farm machinery send data to a streaming application. The application monitors performance, detects any potential defects in advance, and places a spare part order automatically preventing equipment down time.
- A financial institution tracks changes in the stock market in real time, computes value-at-risk, and automatically rebalances portfolios based on stock price movements.
- A real-estate website tracks a subset of data from consumers' mobile devices and makes real-time property recommendations of properties to visit based on their geo-location.



Streaming Data Examples

- A solar power company has to maintain power throughput for its customers, or pay penalties. It implemented a streaming data application that monitors of all of panels in the field, and schedules service in real time, thereby minimizing the periods of low throughput from each panel and the associated penalty payouts.
- A media publisher streams billions of clickstream records from its online properties, aggregates and enriches the data with demographic information about users, and optimizes content placement on its site, delivering relevancy and better experience to its audience.
- An online gaming company collects streaming data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real-time, offers incentives and dynamic experiences to engage its players.

The Characteristics of Streaming Data

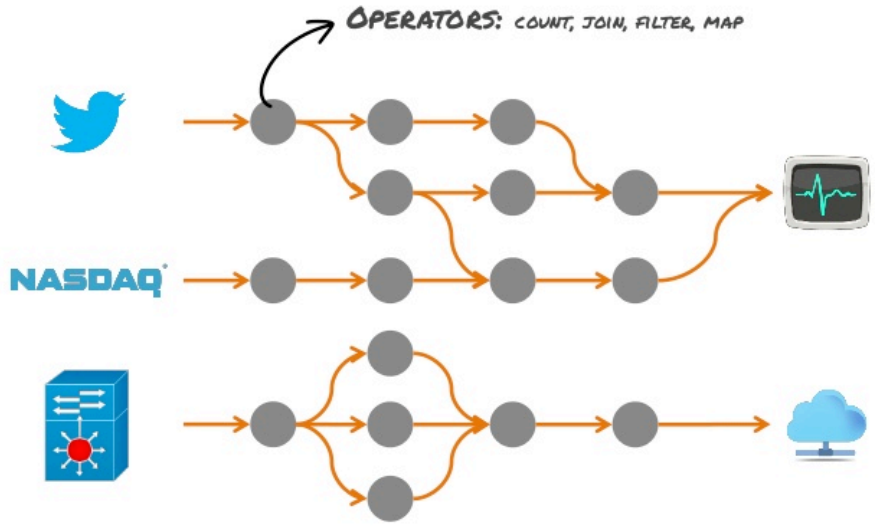
- Data arrives continuously and quickly as flowing water
- Massive data scale, data volume can reach TB level or even PB level
- High real-time requirements, the value of data will be greatly reduced over time
- Data order cannot be guaranteed. The system cannot control the order of the data elements to be processed

Comparison between Batch Processing and Stream Processing

	Batch processing	Stream processing
Data scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window, or on just the most recent data record.
Data size	Large batches of data.	Individual records or micro batches consisting of a few records.
Performance	Latencies in minutes to hours.	Requires latency in the order of seconds or milliseconds.
Analyses	Complex analytics.	Simple response functions, aggregates, and rolling metrics.
Architecture	MapReduce	Storm, Kafka, etc.

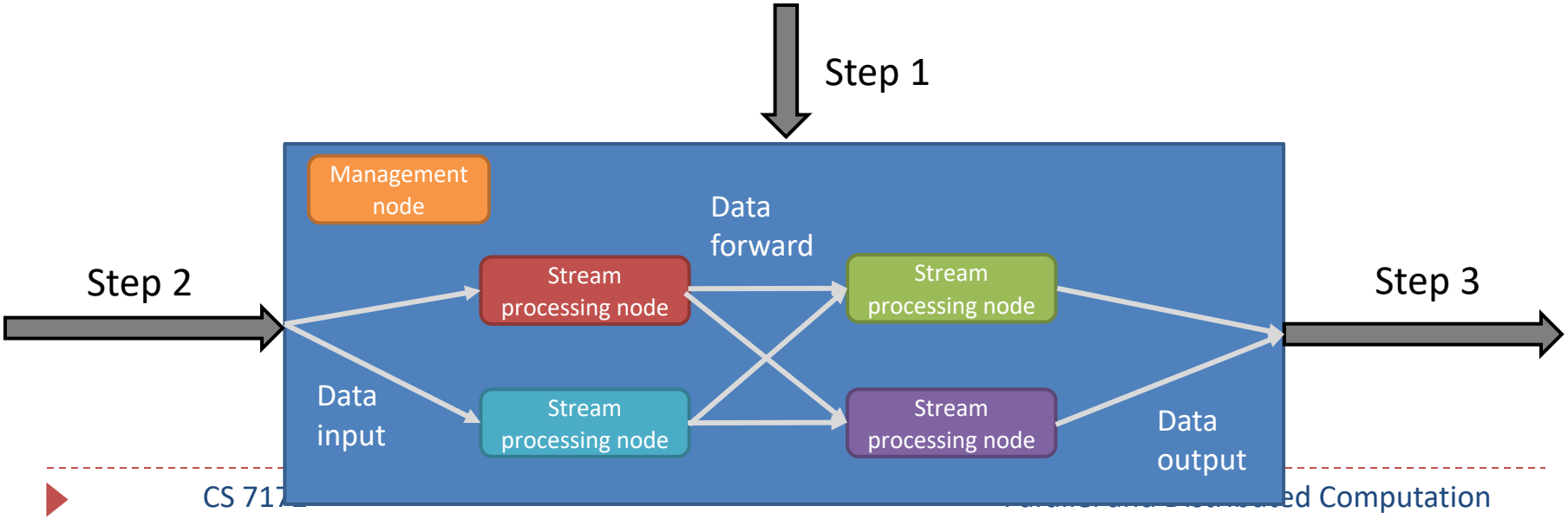
How Stream Computing Works

- Stream computing focuses on real-time. Once the data is generated, it will be processed immediately. When the data is processed, it will be serialized and stored in the cache, and then immediately transmitted to the next node through the network, and the next node will continue processing.



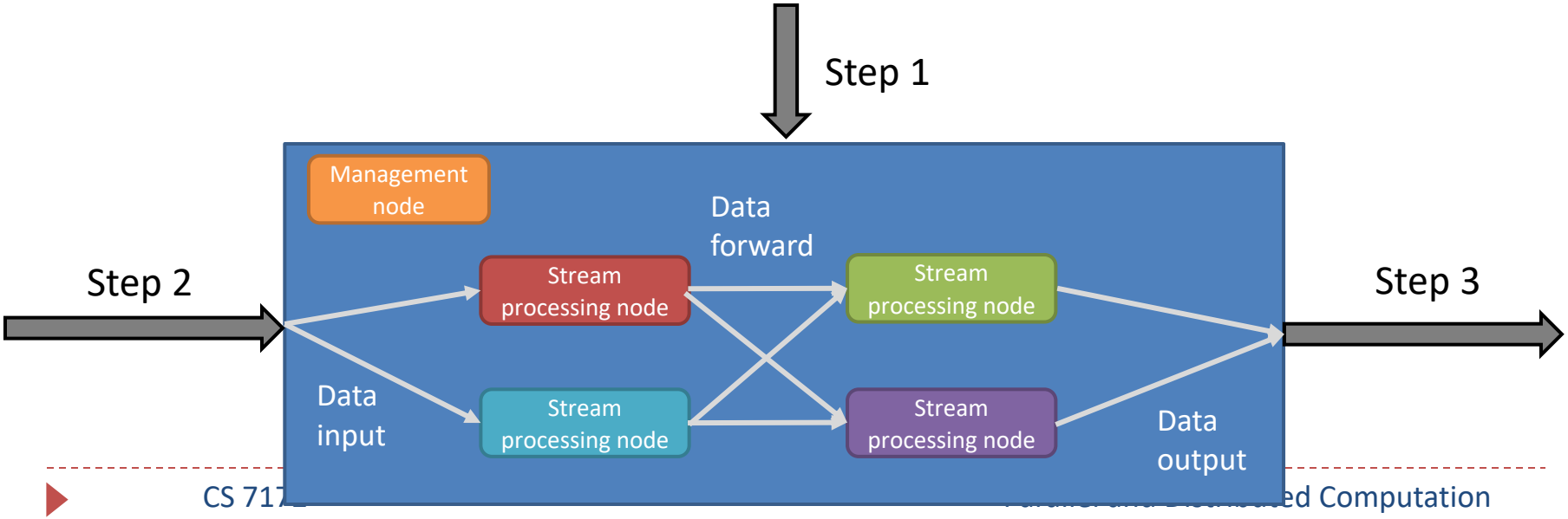
How Stream Computing Works

- Stream computing for data processing, generally includes 3 steps:
  1. Submit a streaming computing job
  2. Load streaming data for stream computing
  3. Continuously output the processed results



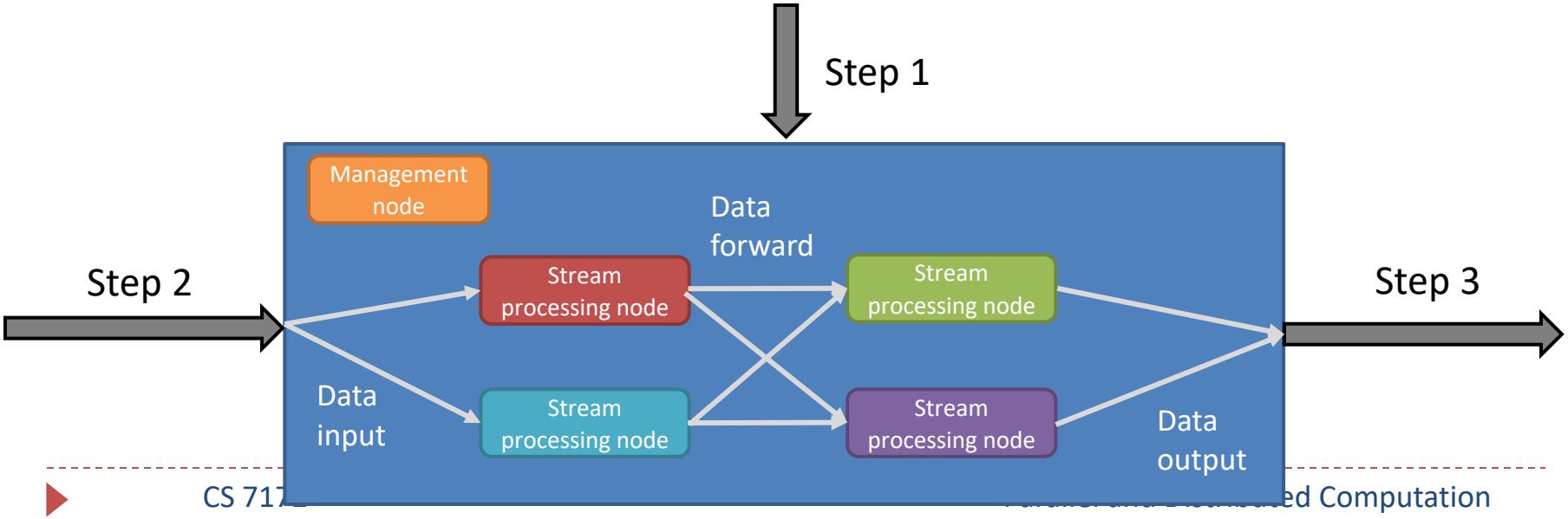
1. Submit a streaming computing job

- Streaming computing job is a ***long live service*** in memory
- The calculation logic must be ***defined in advance*** so that the stream computing knows how to process the data when received submission.
- Processing logic for streaming computing ***cannot be changed*** during execution
- Stream computing does not provide data storage services, so data that has been previously calculated ***cannot be recalculated***



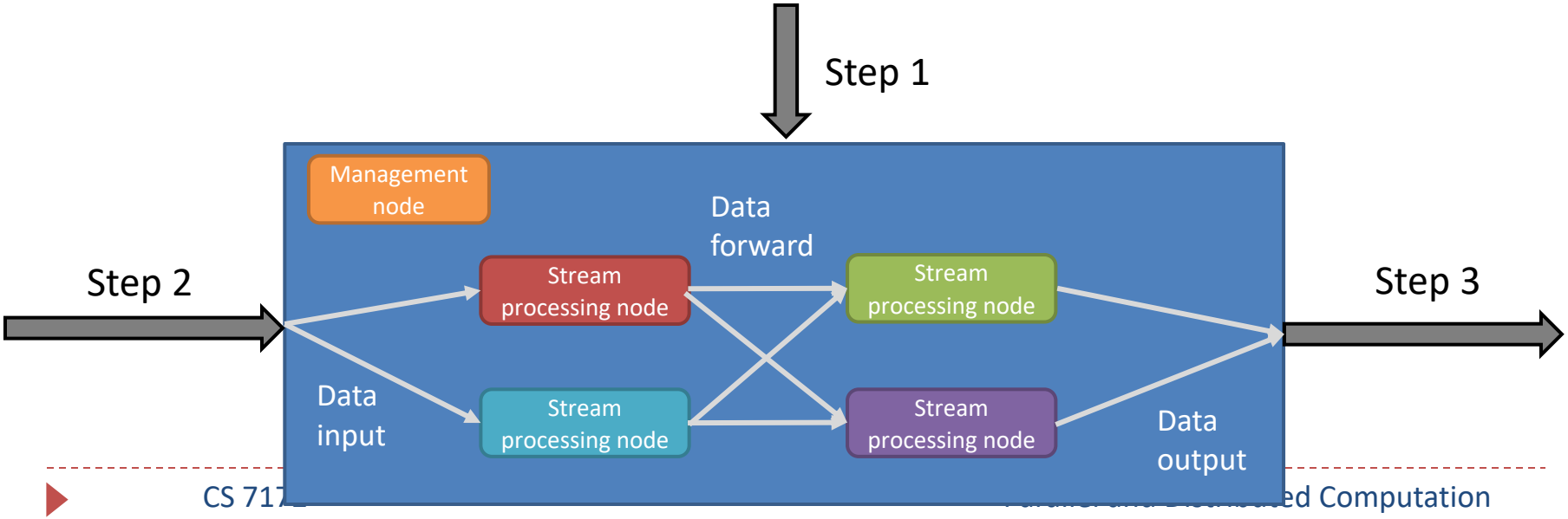
2. Load streaming data for stream computing

- Once the streaming computing job starts, it will be in waiting state for the event to trigger.
- Once the data enters the system, it will immediately execute the computing logic and output the results.
- After the data is processed at one node, it will be forwarded to subsequent nodes for further processing according to predefined rules



3. Continuously output the processed results

- After obtaining the calculation result of the small batch data, the streaming calculation job can output the corresponding result immediately, without waiting for the calculation result of the entire data.
- The data will be discarded immediately after processed



Example of Streaming Systems

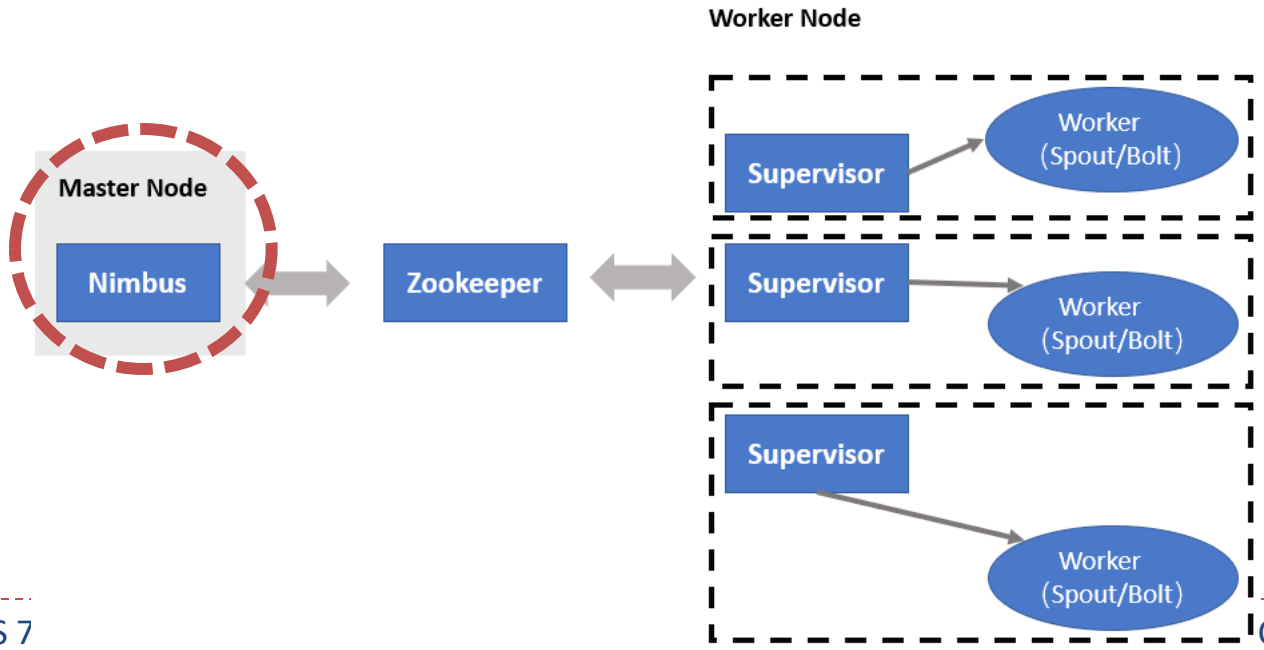
- IBM InfoSphere Streams
  - Provide real-time data analysis services for investment banks, hedge funds, government agencies, etc.

- Apache Storm by Twitter
  - https://storm.apache.org/

- Puma by Facebook
- Dstream by Baidu

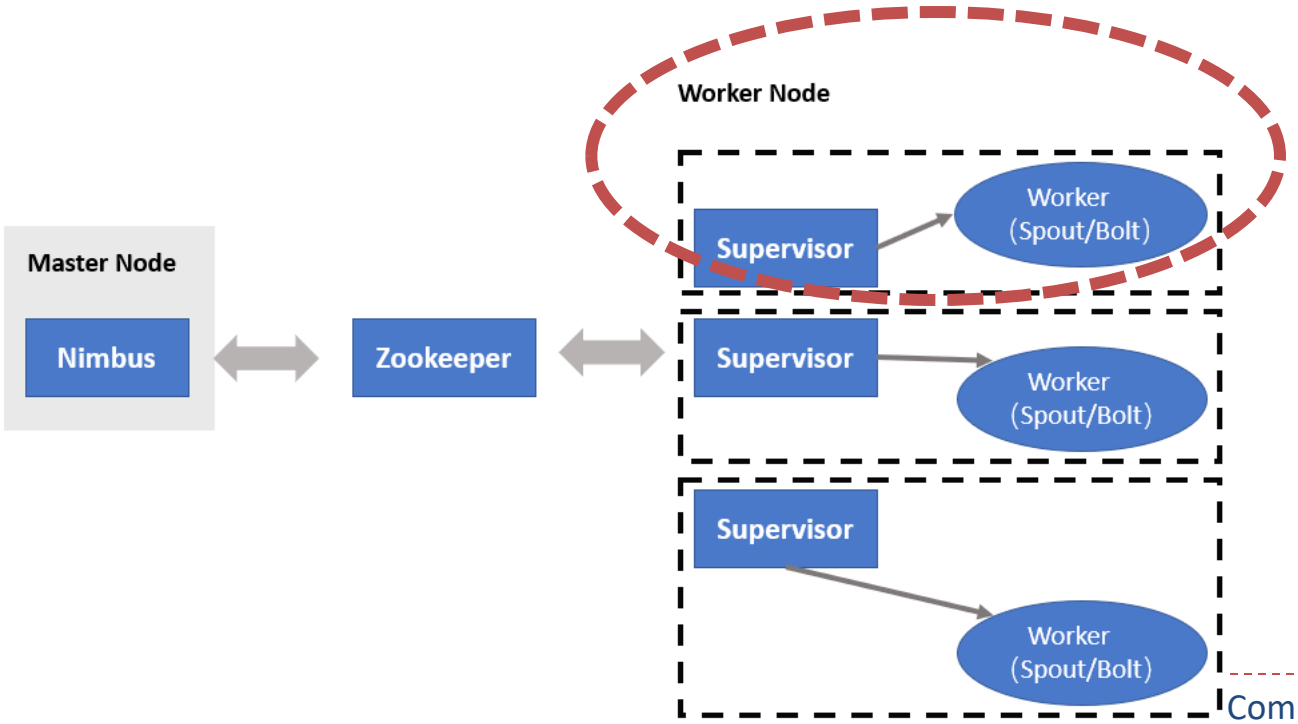
Example of Streaming Systems: Storm

- There are two kinds of nodes on the Storm cluster: Master Node and Worker Nodes.
- A process called "Nimbus" runs on the master node. Nimbus is responsible for distributing code to the cluster, assigning tasks to worker nodes, and monitoring errors and faults.



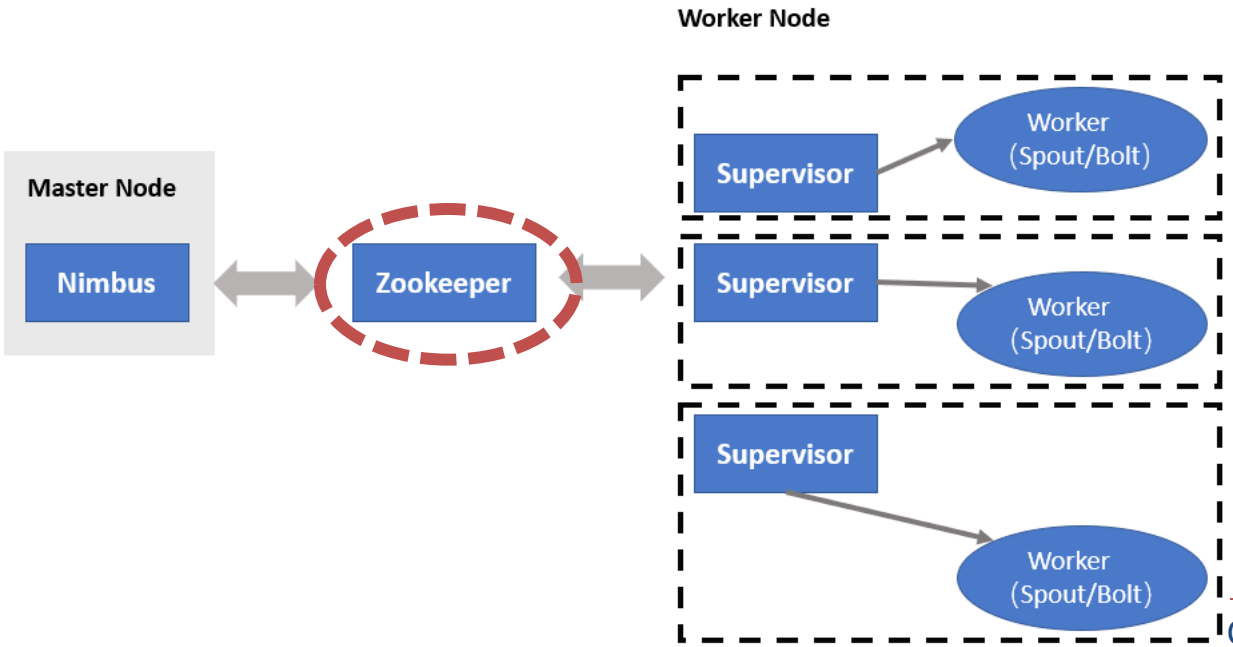
Example of Streaming Systems: Storm

- A daemon called "Supervisor" runs on each worker node.
- Supervisor is responsible for monitoring the work assigned to the node, receiving tasks assigned by Nimbus, and starting and stopping worker processes based on requirements



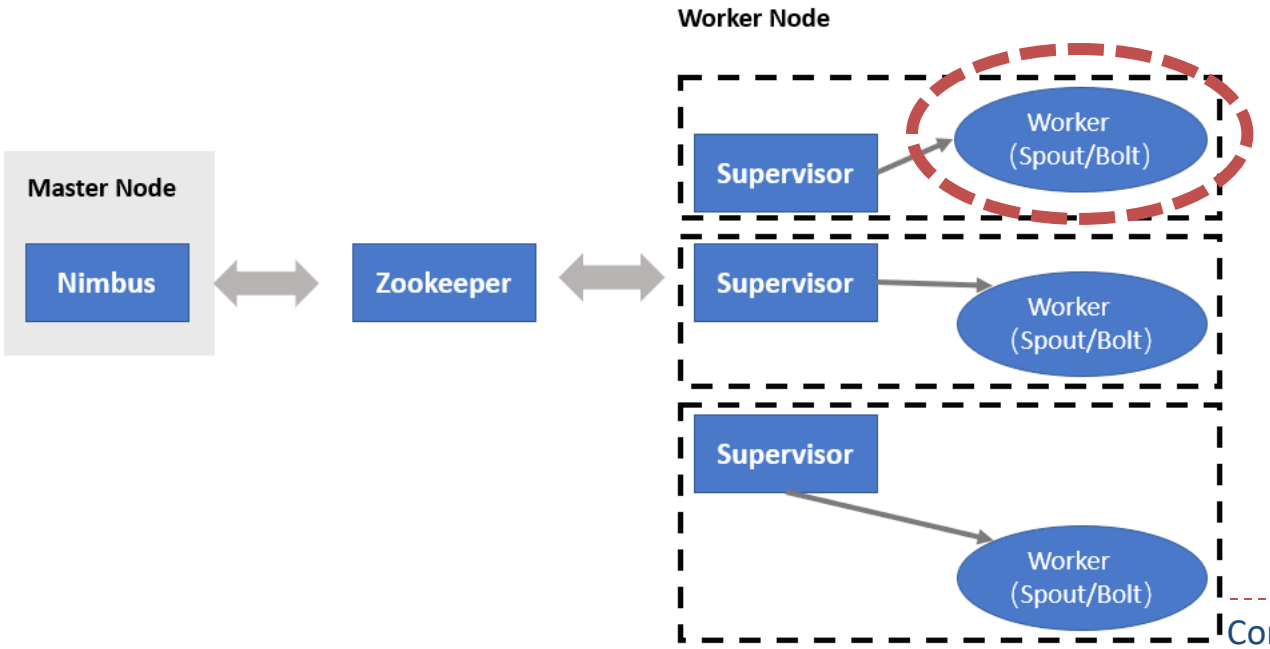
Example of Streaming Systems: Storm

- The interaction between the Master Node and the Worker Nodes is through ZooKeeper
- Nimbus sends tasks or information to the ZooKeeper cluster, and then Supervisors gets tasks from the ZooKeeper cluster and starts worker processes to execute the tasks



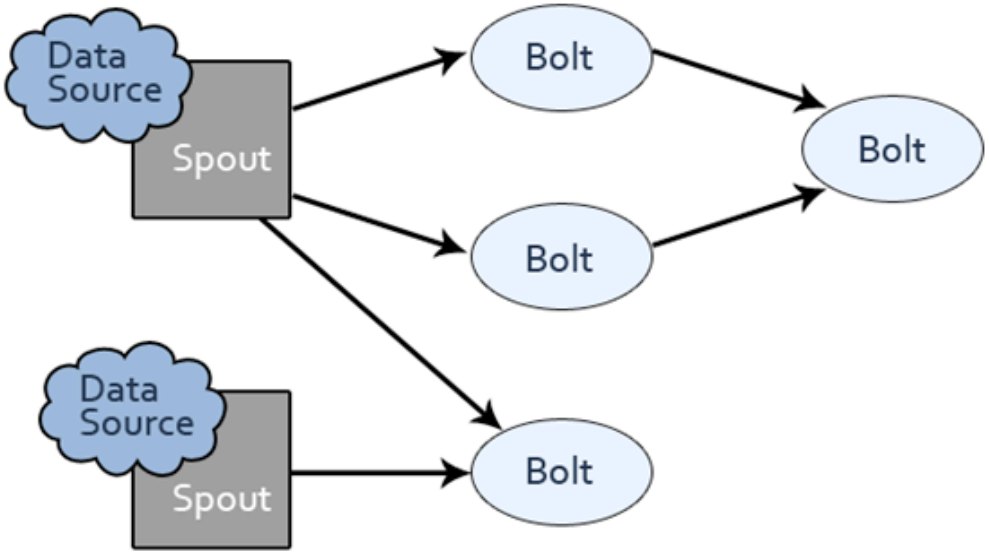
Example of Streaming Systems: Storm

- Every streaming task on Storm includes two parts:
  - Read data, and process task
  - Worker provides spout for reading data; bolt for processing tasks



Example of Streaming Systems: Storm

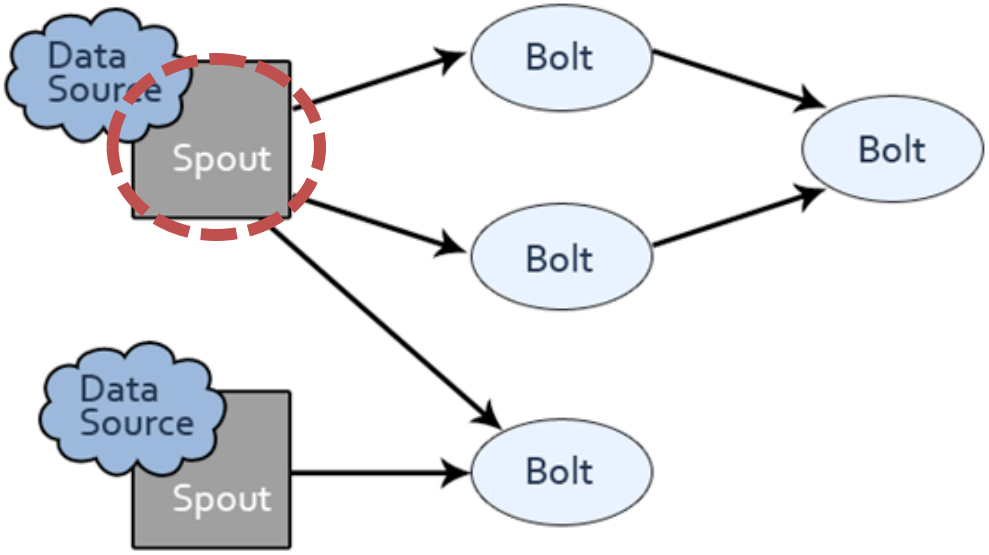
- Every streaming task on Storm includes two parts:
  - Read data, and process task
  - Worker provides spout for reading data; bolt for processing tasks
  - The ***topology*** of spouts and bolts compose one streaming task



A Storm Topology

Example of Streaming Systems: Storm

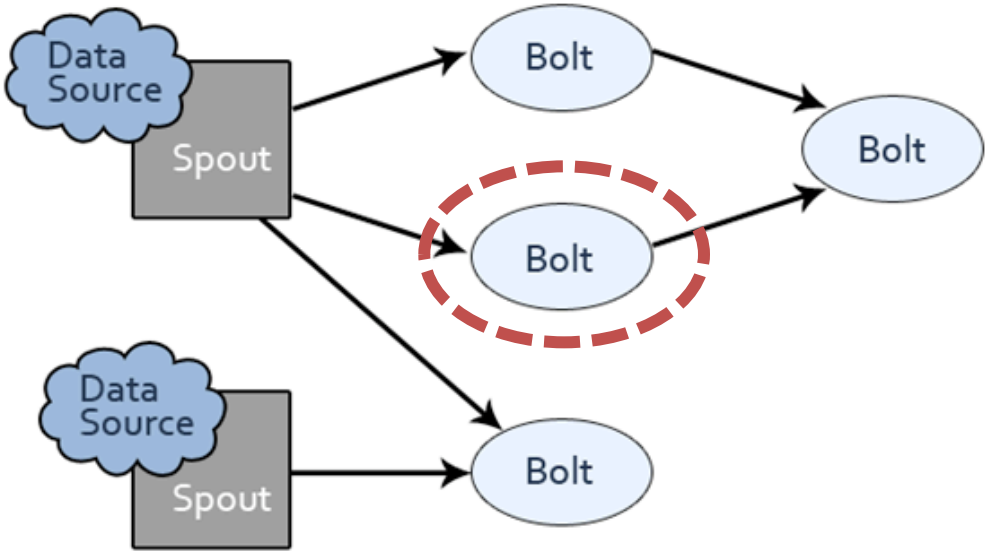
- Spout:
  - Used to receive source data.
  - Typically, Spout reads data tuples from an external data source and then sends them to the topology.



A Storm Topology

Example of Streaming Systems: Storm

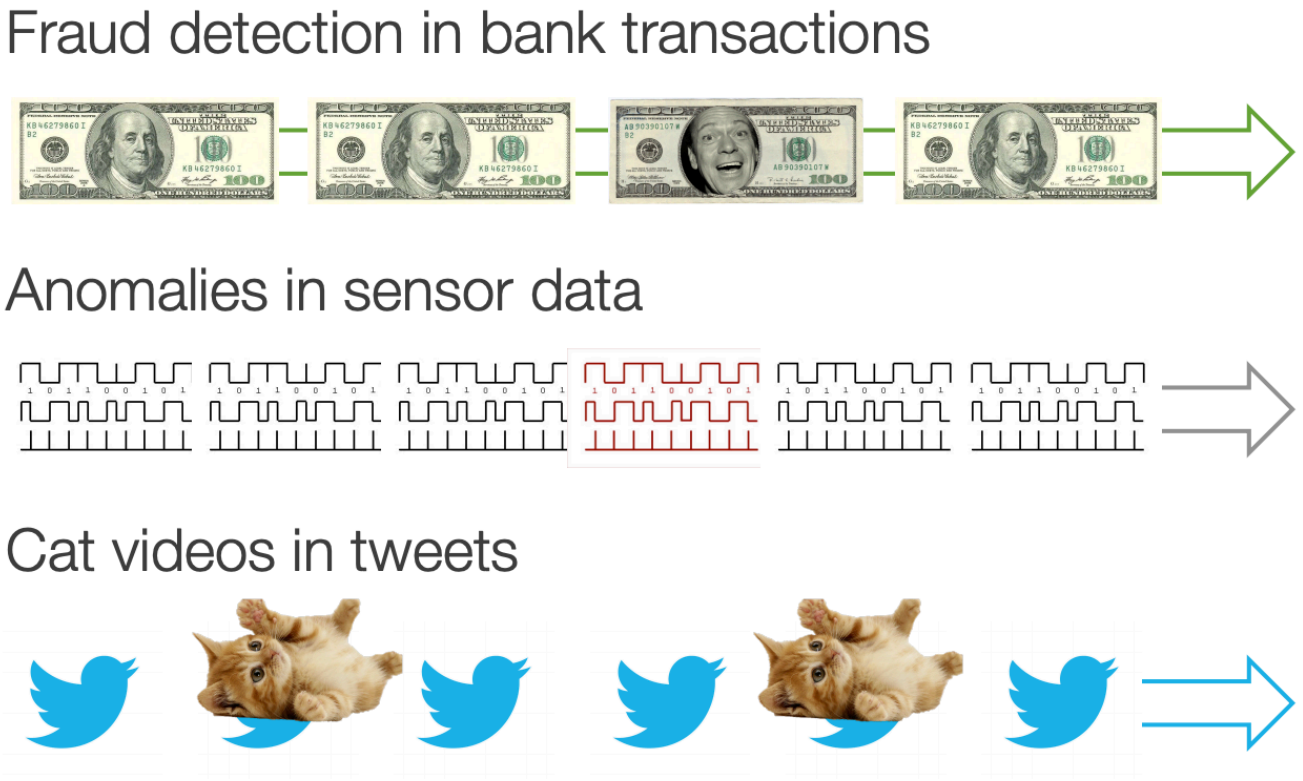
- Bolt:
  - Used to process the input data flow, such as data filtering, functions, aggregations, joins, database interactions, etc.
  - After data processing, a new stream may be output as input for the next Bolt.
  - Each Bolt often has only a single computing logic.



A Storm Topology

Example of Streaming Systems: Spark Streaming

- Spark streaming is used for big streaming data processing



Example of Streaming Systems: Spark Streaming

- Receive data streams from input sources, process them in a cluster, push out to databases dashboards

- Scalable, fault-tolerant, second-scale latencies