Data Analytics with HPC

Data Streaming

Reusing this material

archer

epcc | THE UNIVERSITY OF EDINBURGH

Outline

- What is data stream processing
- Why use streaming
- Strategies for data stream processing
- Frameworks
- Apache Storm as an example
- Conclusions

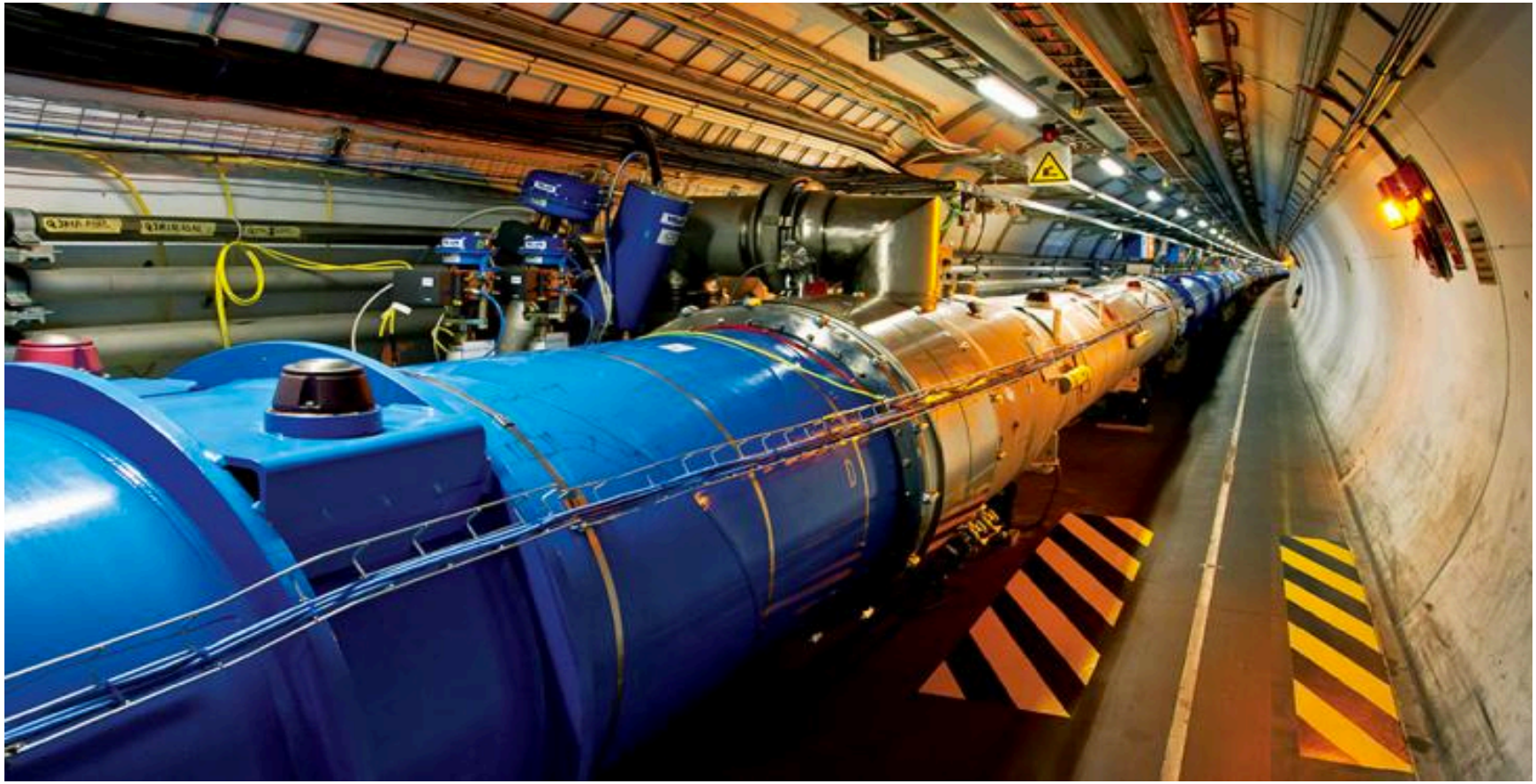
What is data streaming

- Data Streaming is a strategy for processing large datasets
- Three Vs of Big Data:
  - Volume
  - Variety
  - Velocity

Why data streaming

- Some datasets are already too large to store
- Need new strategies for handling large datasets:
  - Process as data is produced
  - Compression/stacking/aggregation
- Realtime computation of sensor data
  - Short response times

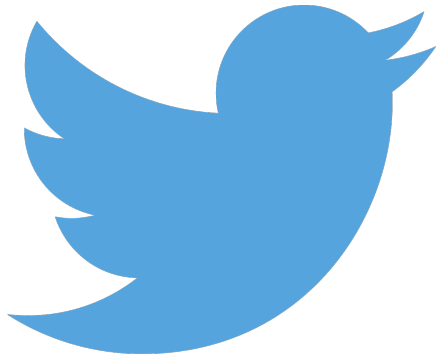
Example: LHC



Example: LHC

- 150 million sensors delivering data 40 million times per second
- 1PB of data per second (!!)
- With current systems it is impossible to store data at this rate
- Fast pre-selection of 1 in 10,000 events
- Fast processing selects 1% of the remaining for further analysis
- Produces 30PB per year

Example: Twitter



- 6,000 tweets per second
  or over 500 million per day
- Data analytics of tweets as they arrive
- Stored in MySQL databases
- Globally unique ID is assigned to each tweet
- Data hash to store the tweet in one of the databases
- Throughput can be increased by increasing the number of databases

Strategies for data stream processing

- Handling large volumes of data and/or high-velocity data streams
- No persistence – data is not stored
- Access to a sliding window on the data
- Often the goal is to extract real-time information
  - Must be scalable to handle peak loads
  - Aiming for short response times
- Data stream mining

Strategies for data stream mining

- Data mining is an example for data processing
- Many machine learning algorithms use multiple passes over the input data
- If a dataset gets too large we can't use random access and multiple scans but have to use single pass algorithms

Machine learning techniques

Online learning

- Data becomes available sequentially
- Model is updated continuously
- Data is discarded after use

Offline or batch learning

- Model is generated from the entire dataset
- Dataset is static
- Still might be too large for random access!

Sequential access

- Update incrementally
- Single step or sliding windows (mini-batch)

Random access

- Allows multiple passes over the data
- For example: k-means, recursive algorithms

Multipass algorithms

- A multipass algorithm traverses the data several times
- Sequential or random access for input data
- Only applicable when the dataset can be read many times
  - Ideally: fits into memory
- For example:
  - K-means

Single pass algorithms

- Reads input exactly once and in order
- Updates the model in each step
- The model must be represented by a small number of properties that can be updated quickly and requires limited amount of storage
  - Typically O(n) time and storage
  - Ideally O(1) storage
- For example: Naïve Bayes

Online learning

- There is a growing community building and evaluating algorithms that can be updated incrementally
- Targeting algorithms that typically require multiple passes and that would not be feasible in an online environment
- For example:
  - Clustering
  - K-means

Data Streaming frameworks

- A selection of examples:
  - Apache Storm (Twitter)
  - Apache Spark (Data Streaming)
  - Yahoo S4
  - Google TensorFlow
  - Amazon Kinesis
  - Apache Hadoop/YARN

Apache Storm

- A distributed realtime computation system
- Scalable
- Fault tolerant
- Simple APIs, multi language

Storm use cases

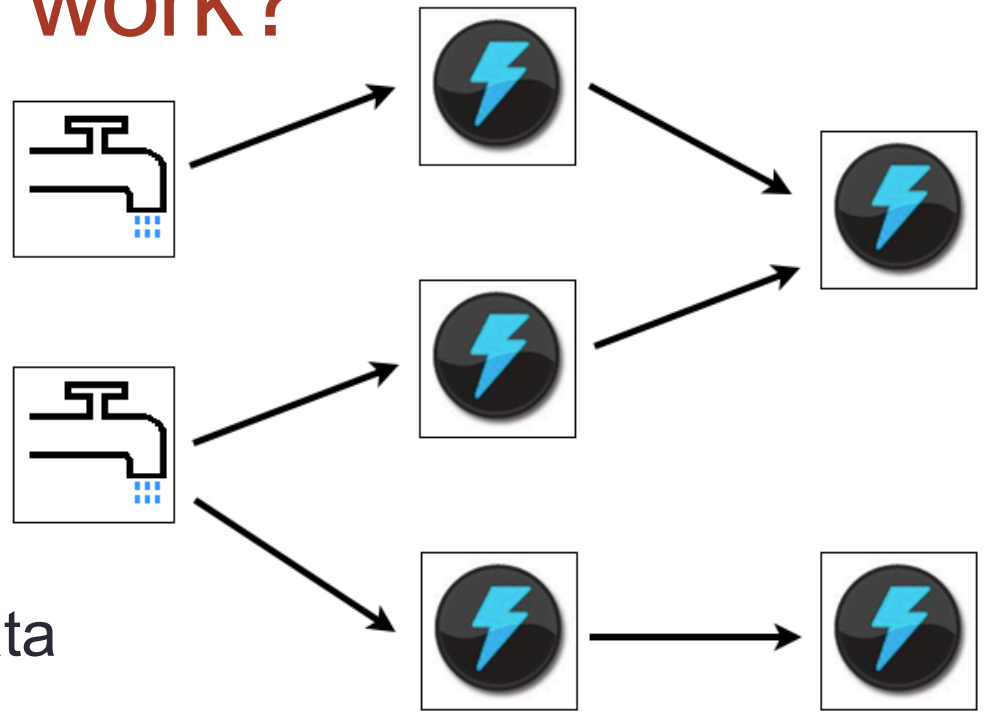
- Storm powers a wide variety of **Twitter** systems, for example realtime analytics, search, revenue optimisation.
- Storm powers a wide range of realtime features at **Spotify**, for example music recommendation, monitoring, ads targeting.
- And many others!

Twitter and Storm

Twitter extracts emerging trends from the fire hose of tweets and maintains them at the local and national level. As a story begins to emerge, Twitter's trending topics algorithm identifies the topic in real time. This real-time algorithm is implemented within Storm as a continuous analysis of Twitter data.

(https://www.ibm.com/developerworks/opensource/library/os-twitterstorm/)

How does Storm work?



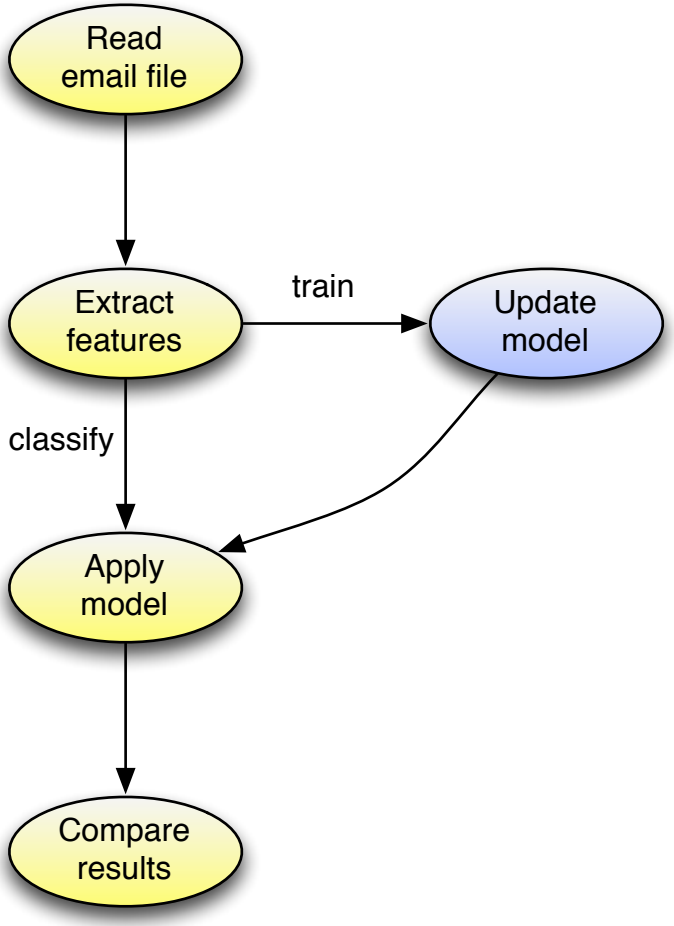
- Describe a data flow as a graph
- In Storm this is called a “topology”
- Nodes are procedures that process or produce data (“bolts” and “spouts”)
- Edges indicate how data streams between nodes

How Storm processes data

- Each node consumes a stream of data
- Data streams are streams of tuples
- When a data tuple arrives at a bolt it kicks off a processing procedure
- Spouts are data producers that initiate a data stream
- Bolts produce zero or more output tuples for each input tuple

Example: Spam filtering

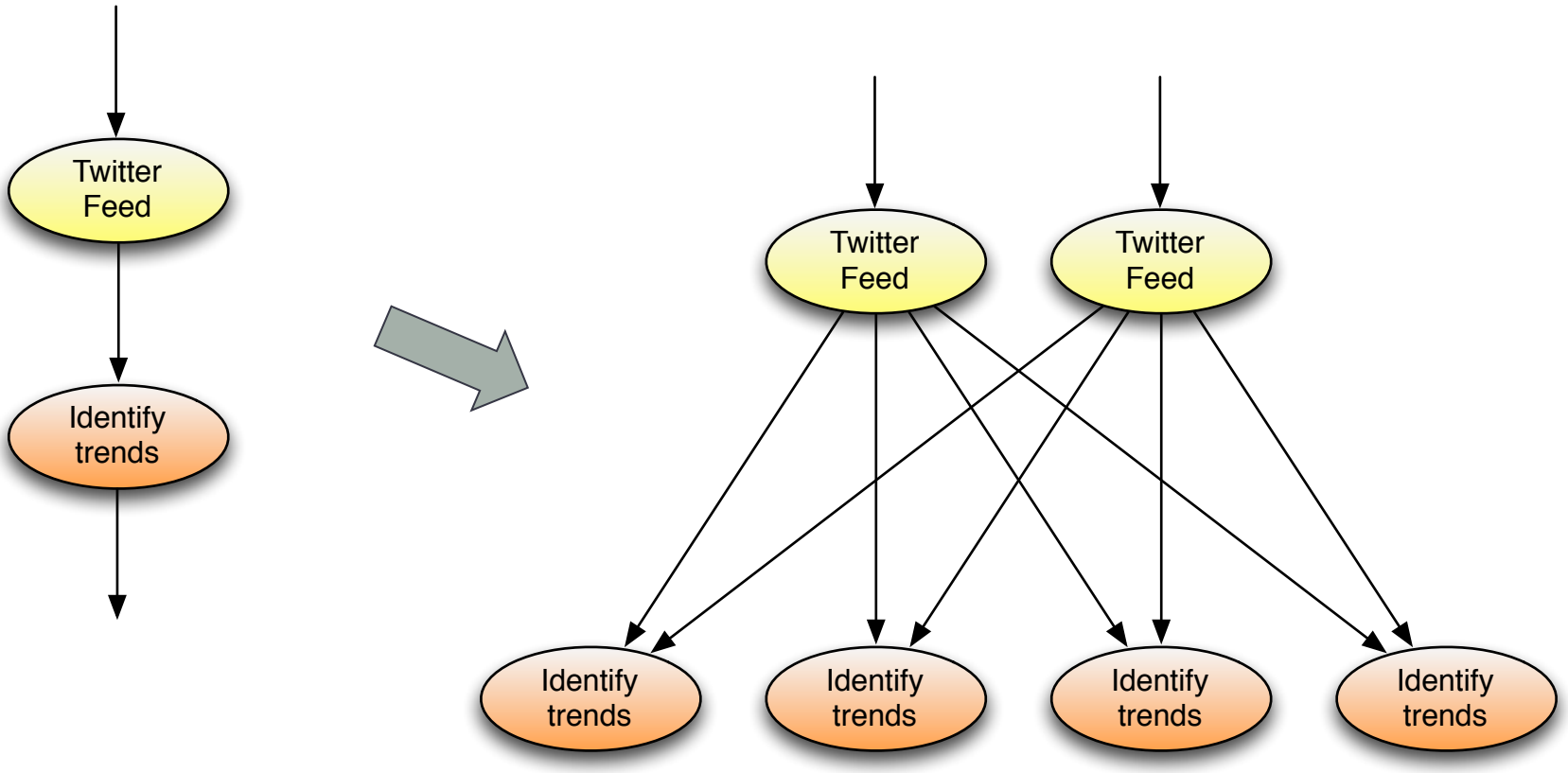
- Train model offline with an existing dataset and store
- Classify incoming emails as spam (or ham) using the stored model
- Update model when the user marks an email as spam



Scalability

- Storm uses data parallelism: Data streams are a sequence of small *blocks*
  - Twitter feed is a sequence of tweets plus metadata
  - Sensor data is a sequence of observations
- Each node executes in many instances

Data parallelisation in Storm



Parallelisation

- The system distributes each data block to a target instance, according to *grouping* types defined in the topology
  - Shuffle: random distribution, the default
  - Fields: group by certain data attributes
  - All-to-all, all-to-one
- Worker nodes wait for work: Delivery of a data block triggers the execution
- Topologies can be rebalanced to increase or decrease the number of worker processes

Fault tolerance

- Fault recovery: On failure of a worker messages are reassigned; worker are restarted if they fail
- Fault detection: data is tracked and replayed until it has been processed
- Storm daemons are designed to be stateless and fail-fast – meaning that they start up again after a failure as if nothing has happened

Conclusion

- Big data is challenging existing systems
  - Variety
  - Velocity
  - Volume
- Processing must happen at real-time as data sets are too large to be persisted
- Apache Storm is an example for a platform that offers reliable processing of unbounded streams of data