Big Stream Processing 1: Intro & Apache Storm

Shiow-yang Wu (吳秀陽)

CSIE, NDHU, Taiwan, ROC

60 Seconds

Big Streaming Data

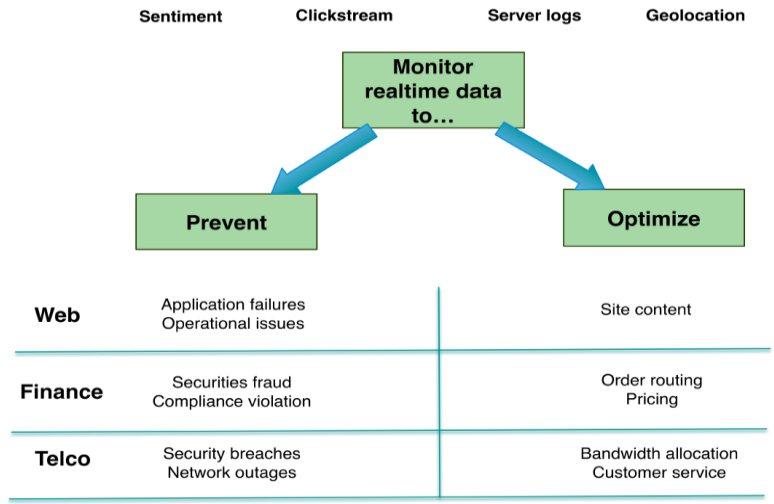
- World is getting more instrumented and connected
- Digital data from various hardware (e.g., sensors) or software flooding in the format of flowing streams
- **Examples**: financial markets, surveillance systems, manufacturing, smart cities, …
- Need to collect, process, and analyze big streams to extract valuable information, discover new insights in realtime, and to detect emerging patterns and outliers

Real-time Data Analytics

Motivation

- Many important applications must process large streams of live data and provide results in near-real-time
  - Social network trends
  - Website statistics
  - Intrusion detection systems
  - …

Patterns Driving Most Streaming Use Cases



Static vs Streaming

- In static data computation, questions are asked of static data.
- In streaming data computation, data are continuously evaluated by static questions.

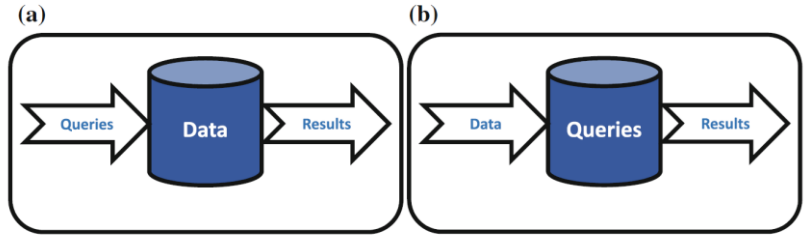


Fig. 5.1 Static data computation versus streaming data computation

Challenges

- Streaming data management
- Arbitrary and interactive exploration
- Real-time analytics
- Recency matter: alerts on recent changes
- Availability

Is Hadoop a Solution?

- Upload data to Hadoop
- Query it ! done!

Hadoop for Big Streams?

- Hadoop was designed for batch processing
- Input all data at once, process it and write a large output.

Batch vs. Real-time Processing

Batch processing:

Collect data over a period of time, input the batch into the system, process it and write the output.
Examples: billing systems.

Real-time processing:

Continually input, process and output data. Data must be processed in a small time period.
Examples: radar systems, customer services, ATMs.

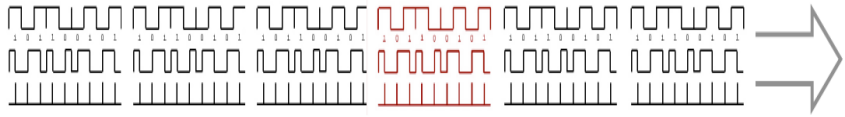
Problems of Hadoop for Big Stream Processing

- Hadoop framework requires the results of each single Map or Reduce task to be *materialized* into a local file.
- It dramatically hurts the performance of applications with realtime processing requirements.

Big *Streaming* Data Processing

- Fraud detection in bank transactions

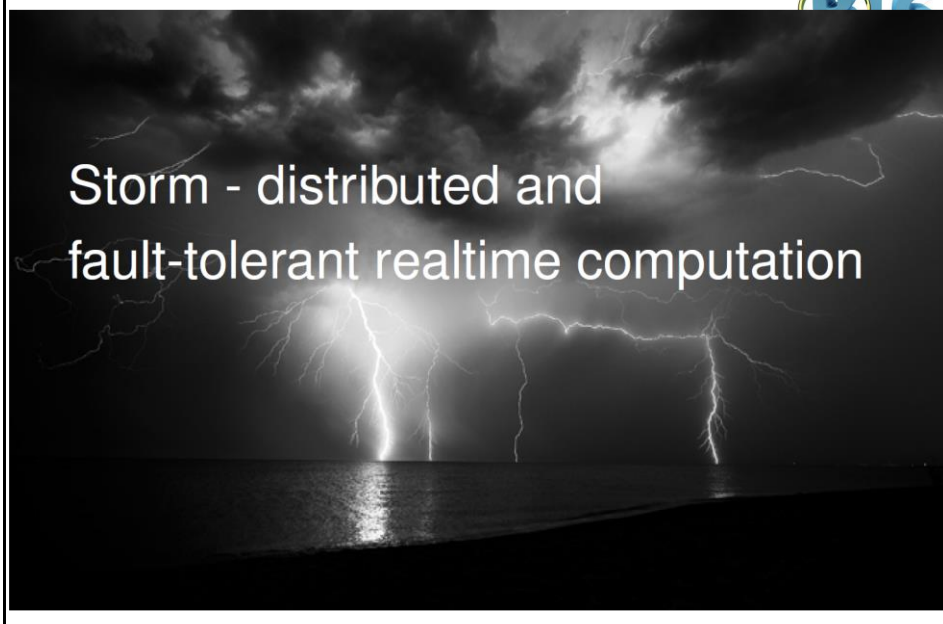
- Anomalies in sensor data



- Cat videos in tweets

How to Process Big *Streaming* Data

- Scales to hundreds of nodes
- Achieves low latency
- Efficiently recover from failures
- Integrates with batch and interactive processing



Storm Intro

- Created by Nathan Marz @ BackType
- Open sourced on 19th September, 2011
- Free and open source distributed realtime computation system
- Reliably process unbounded streams of data
- Analysis on streams of data as they come in, so you can react to data **as it happens**
- Can be easily integrated with any programming language
- Developed at Backtype and open sourced by Twitter
- Used in realtime analytics, online machine learning, continuous computation, distributed RPC, ETL(Extract-Transform-Load), …

Companies & Projects Using Storm

The Weather Channel, WebMD, Alibaba.com, FullContact, yelp, YAHOO! JAPAN, Spotify, GROUPON, Flipboard, Cerner, Aeris COMMUNICATIONS, Baidu 百度, parc A Xerox Company, 淘宝网 Taobao.com, rubicon PROJECT, rocketfuel, HolidayCheck.com, and many, others

Storm Features

- **Simple programming model**
  - Topology - Spouts – Bolts (more about this later)
- **Programming language agnostic**
  - *Clojure, Java, Ruby, Python default*
- **Fault-tolerant**
- **Horizontally scalable**
  - Ex: 1,000,000 messages per second on a 10 node cluster
- **Guaranteed message processing**
- **Fast : Uses zeromq message queue**
- **Local Mode : Easy unit testing**

Apache Storm

- Apache Storm is a free and open source distributed realtime computation system
- Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing

Key Concepts

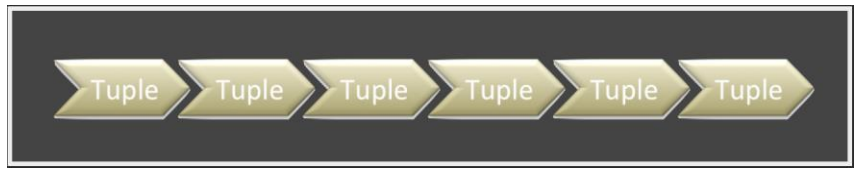
Tuples

- An ordered list of values/objects (any type)
- Values/objects must be serializable
- Tuple = list of values/objects

  [198735697, "foobar", { "ip" : "10.0.0.1" }]

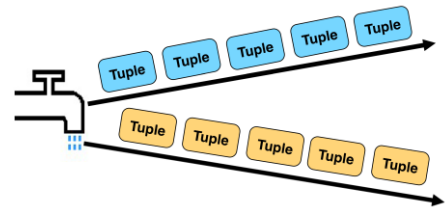
Stream

- An unbounded sequence of tuples
- Core abstraction in Storm
- Stream = sequence of tuples



Spouts

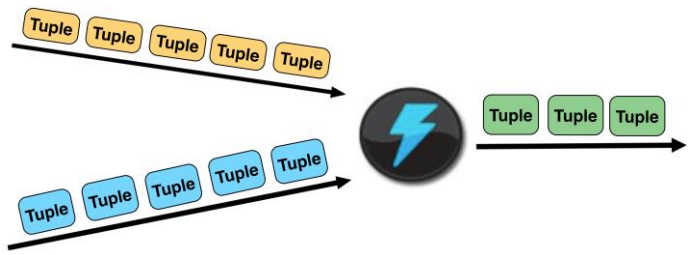
- Generate tuples from other sources
  - event data
  - log files
  - Queues
- Sources of streams



- Read input data from an external source
- Emit them as tuple streams into Storm
- Spouts can emit more than one stream
- Example: read from Twitter streaming API

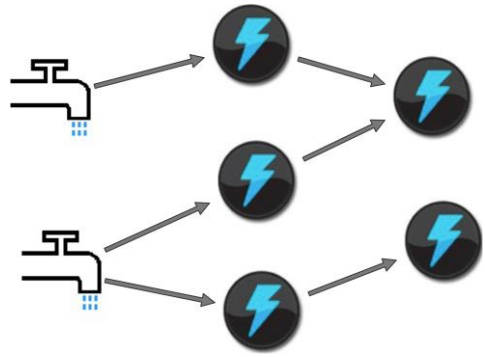
Bolts

- Process input streams and produce new streams
- Can be any functionality: filtering, functions, aggregation, joins..
- Complex transformations require multiple bolts



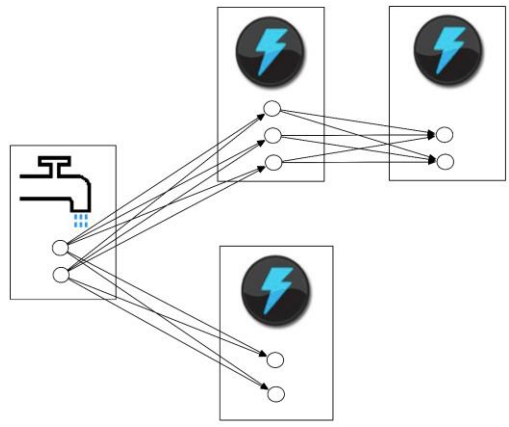
Topology

- Network of spouts and bolts
- Graph: node = spout or bolt, edge = which bolt subscribes to which stream

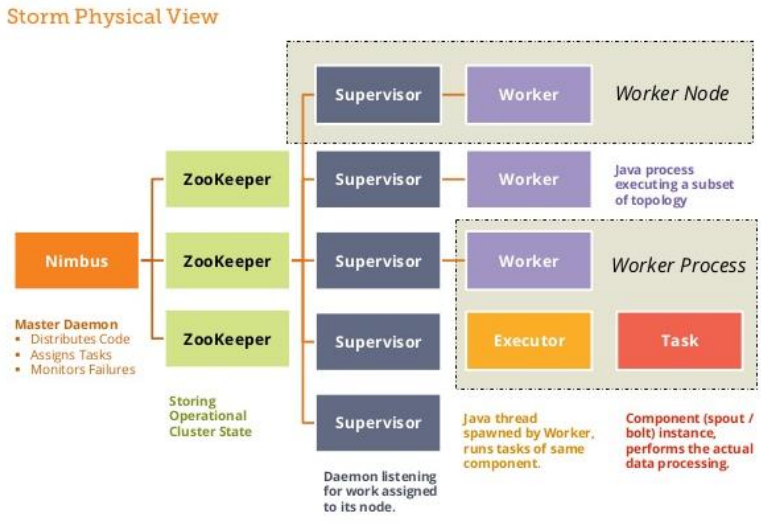


Tasks

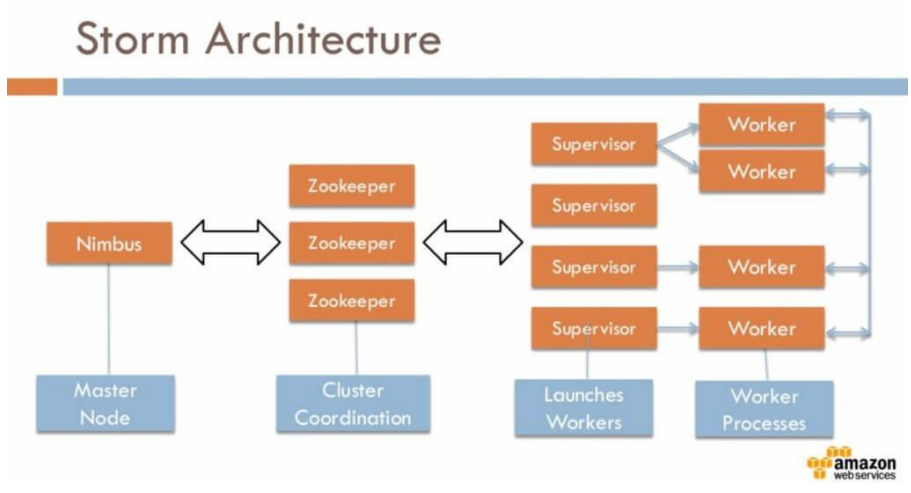
- Spouts and bolts execute as many tasks across the cluster



Storm Architecture



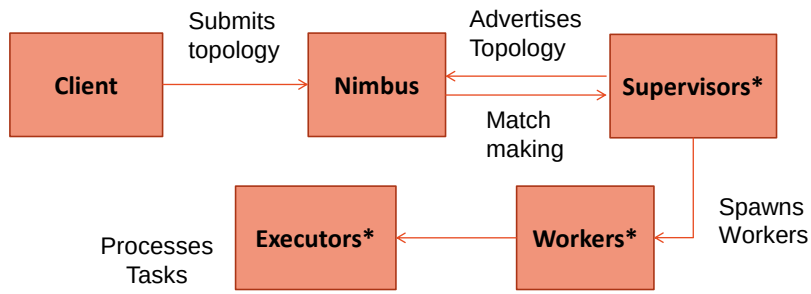
A simplified view



Storm Architecture

- **Nimbus - Master Node**
  - similar to job tracker in Hadoop
  - manages the topologies
  - distributes code across cluster
  - assigns tasks
  - monitors failures
- **Zookeeper**
  - Cluster state of Nimbus and Supervisor maintained in zookeeper
- **Supervisor – Worker Node**
  - similar to task tracker in Hadoop
  - manages workers
  - worker is a thread spawn by supervisor to do work
  - communicates with Nimbus through Zookeeper about topologies and available resources
  - runs bolts and spouts as tasks
- **Workers**
  - Listens for assigned work and executes the application.

Interaction between Storm Internals Components

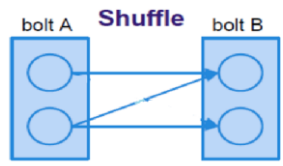


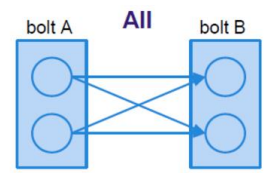
Events - Heartbeat protocol (every 15 seconds), synchronize supervisor event(every 10 seconds) and synchronize process event(every 3 seconds).

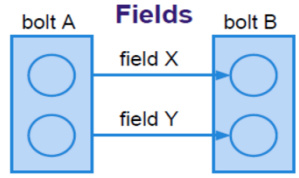
Stream Grouping

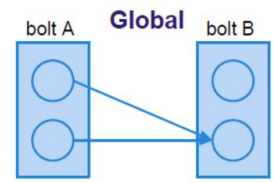
- When a tuple is emitted, which task does it go to?
- Shuffle groupings: pick a random task
- Fields groupings: consistent hashing on a subset of tuple fields
- All groupings: send to all tasks
- Global groupings: pick task with lowest id
- Direct groupings: the source decides which component will receive the tuple

Stream Grouping









Processing Semantics

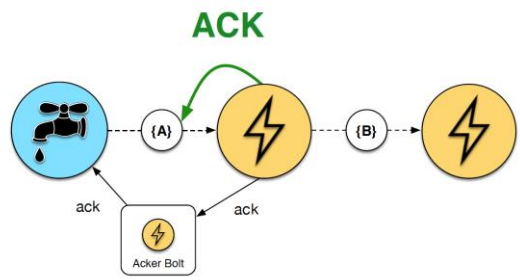
- Atleast once: Each tuple that is input to the topology will be processed atleast once.
- Atmost Once: Each tuple is processed once or dropped in case of failure.

States of workers

Supervisor periodically checks the state of workers for managing the worker processes.

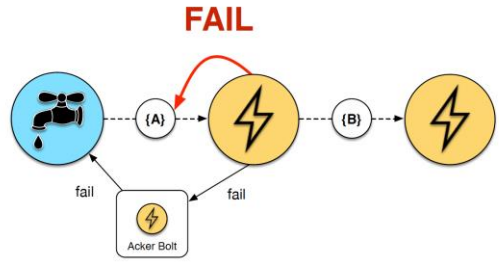
- Timed out
- Not started
- Disallowed
- Valid

Reliable Processing



Acks are delivered via a system-level bolt

Reliable Processing



Bolts can also ***Fail*** a tuple to trigger a spout to replay the original.

Topologies run forever

- Starting a topology

```
storm jar mycode.jar twitter.storm.MyTopology demo
```

- Killing a topology

```
storm kill demo
```

Hadoop vs. Storm

Hadoop

- batch processing
- runs jobs to completion
- stateful nodes
- scalable
- guarantees no data loss
- open source
- big batch processing

Storm

- real-time processing
- topologies run forever
- stateless nodes
- scalable
- guarantees no data loss
- open source
- fast, reactive, real-time processing

Storm- Pros and Cons

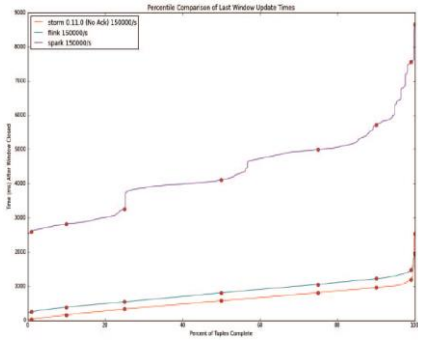
Pros

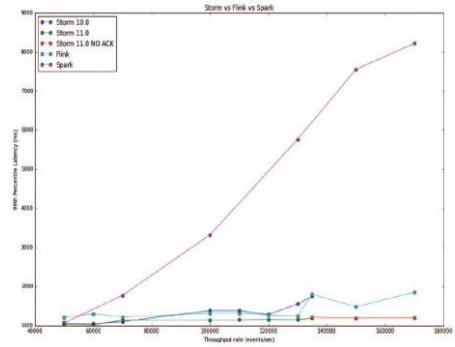
- Fault tolerance: High fault tolerance
- Latency: very less
- Processing Model: Real-time stream processing model
- Programming language dependency: any programming language
- Reliable: each tuple of data should be processed at least once
- Scalability: high scalability

Cons

- Use of native scheduler and resource management feature (Nimbus) in particular, become bottlenecks.
- Difficulties with debugging given the way the threads and data flows are split.

Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming





References

- Apache Storm Based on Topology for Real-Time Processing of Streaming Data from Social Networks, By Anatoliy Batyuk, Volodymyr Voityshyn, Presented at IEEE First International Conference on Data Stream Mining & Processing http://ieeexplore.ieee.org/document/7583573/?reload=true
- Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming, By Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Thomas Graves, Mark Holderbaugh Zhuo Liu, Kyle Nusbaum, Kishorkumar Patil, Boyang Jerry Peng and Paul Poulosky Yahoo Inc., Presented at 2016 IEEE International Parallel and Distributed Processing Symposium Workshops https://yahooeng.tumblr.com/post/135321837876/benchmarking-streaming-computation-engines-at
- INTRODUCTION TO APACHE STORM, by Tiziano De Matteis http://www.slideshare.net/tizianodem/introduction-to-apache-storm-55467258
- Using apache storm for big data, S Surshanov*, IITU, Kazakhstan http://www.cmnt.lv/upload-files/ns_24brt003_CMNT1903-802.pdf
- Storm @Twitter, Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel*, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal, Dmitriy Ryaboy, Twitter, Inc., *University of Wisconsin – Madison https://cs.brown.edu/courses/csci2270/archives/2015/papers/ss-storm.pdf