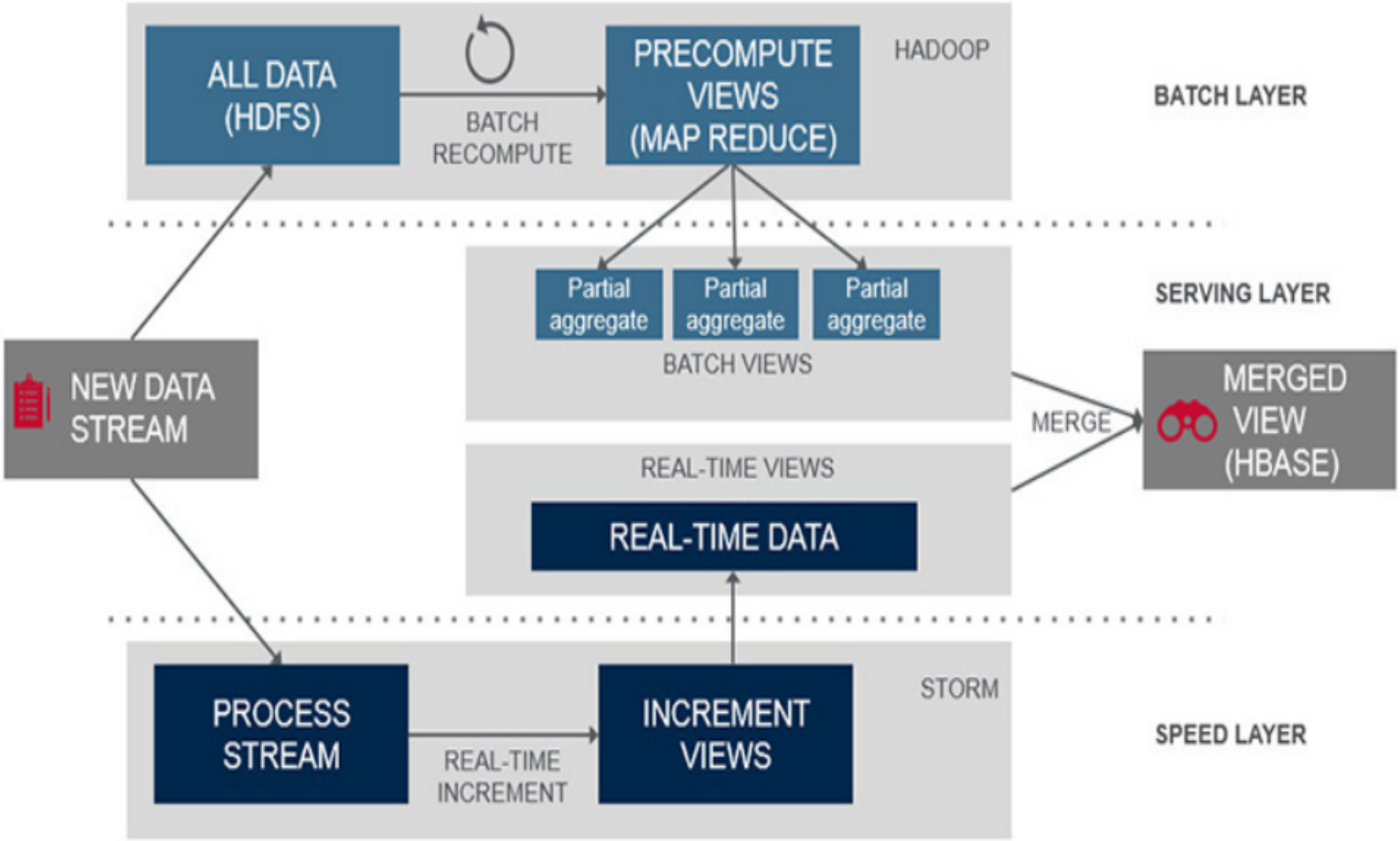
Lambda Architecture



Use Cases

	"Prevent" Use Cases	"Optimize" Use Cases
Financial Services	• Securities fraud • Operational risks & compliance violations	• Order routing • Pricing
Telecom	• Security breaches • Network outages	• Bandwidth allocation • Customer service
Retail	• Shrinkage • Stock outs	• Offers • Pricing
Manufacturing	• Preventative maintenance • Quality assurance	• Supply chain optimization • Reduced plant downtime
Transportation	• Driver monitoring • Predictive maintenance	• Routes • Pricing
Web	• Application failures • Operational issues	• Personalized content

Storm

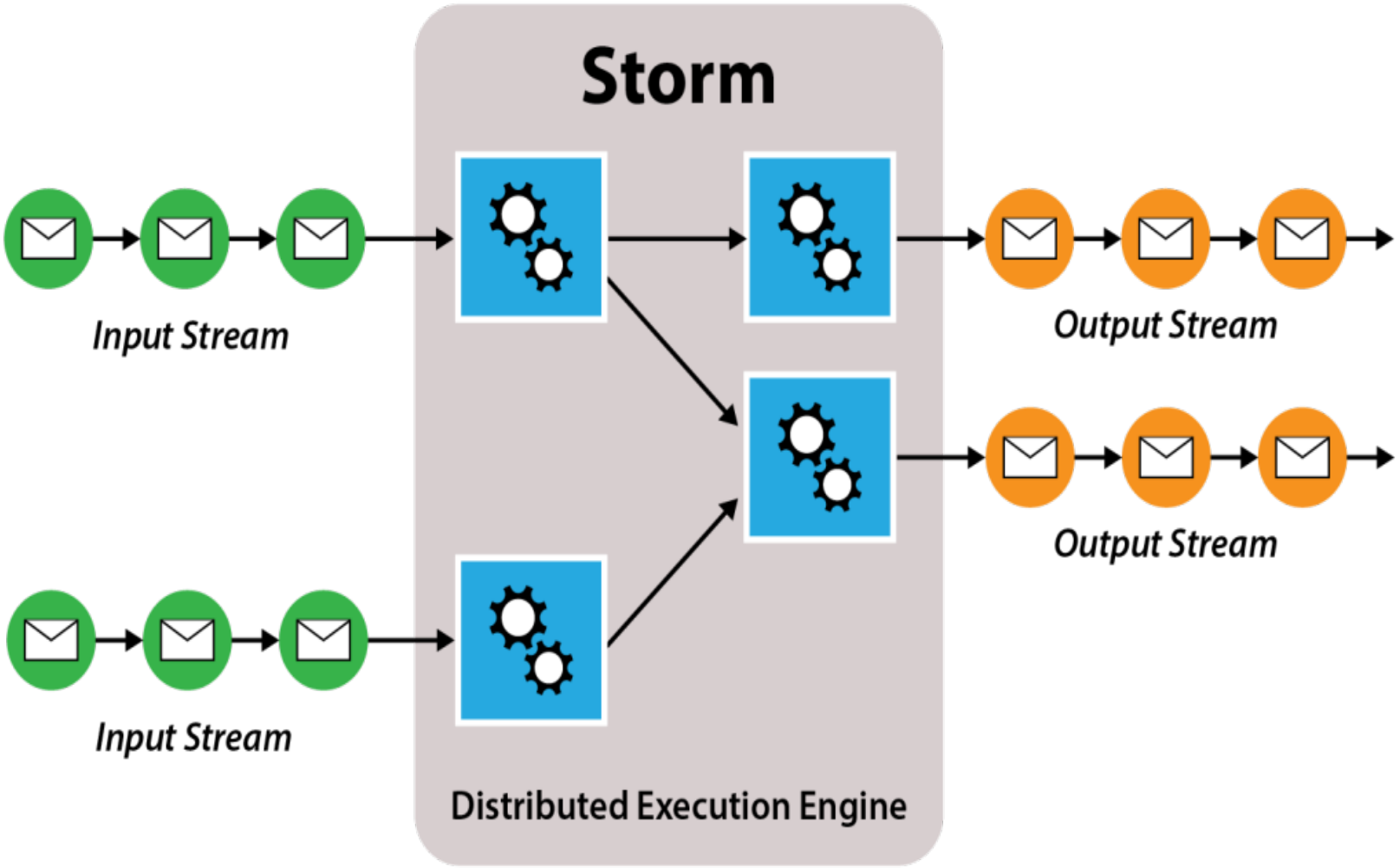
History

- Nathan Marz and team at BackType
- Acquired by Twitter
- Opensourced by Twitter
- Initial release 9-2011

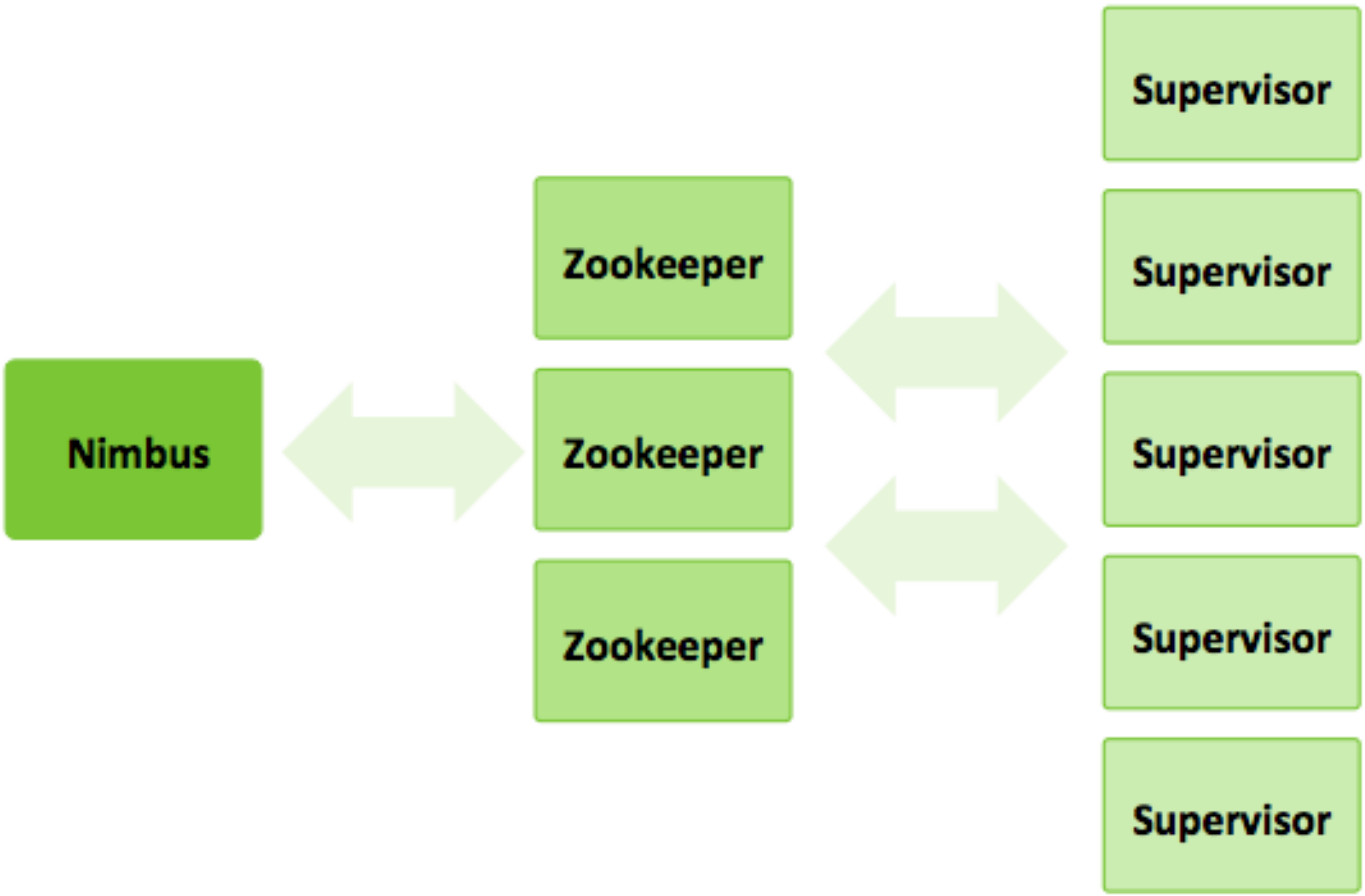
Characteristics

- **Fast** – benchmarked as processing one million 100 byte messages per second per node
- **Scalable** – with parallel calculations that run across a cluster of machines
- **Fault-tolerant** – when workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node
- **Reliable** – Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.
- **Easy to operate** – standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate

Big Picture


Storm
Input Stream
Input Stream
Output Stream
Output Stream
Distributed Execution Engine

Storm Cluster

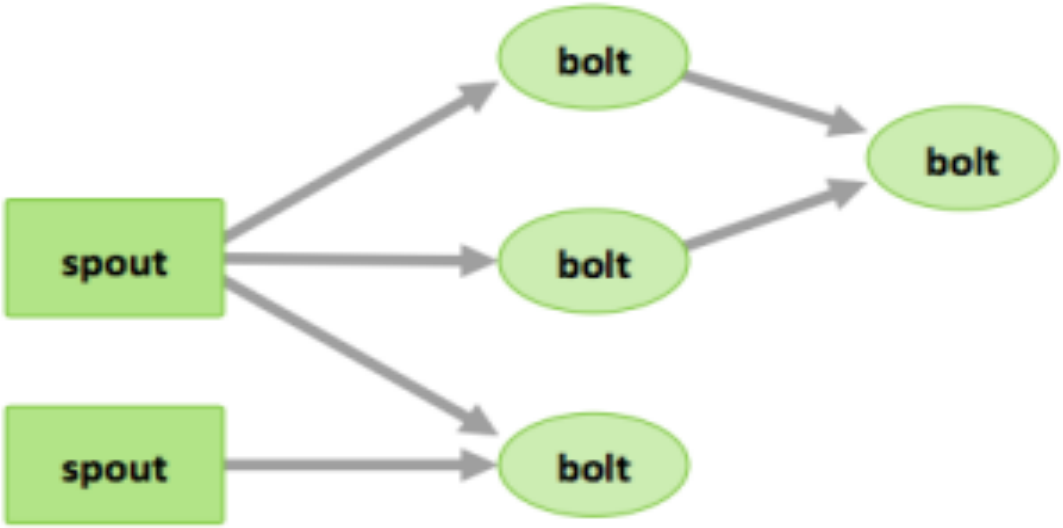


Storm Cluster Components

- **Nimbus node** (master node, similar to the Hadoop JobTracker):
  - Uploads computations for execution
  - Distributes code across the cluster
  - Launches workers across the cluster
  - Monitors computation and reallocates workers as needed
- **ZooKeeper** nodes – coordinates the Storm cluster
- **Supervisor** nodes – communicates with Nimbus through Zookeeper, starts and stops workers according to signals from Nimbus

Topology

Tuples- an ordered list of elements. For example, a “4-tuple” might be (7, 1, 3, 7)
Streams - an unbounded sequence of tuples.
Spouts -sources of streams in a computation (e.g. a Twitter API)
Bolts - process input streams and produce output streams. They can: run functions; filter, aggregate, or join data; or talk to databases.
Topologies - the overall calculation, represented visually as a network of spouts and bolts (as in the following diagram)

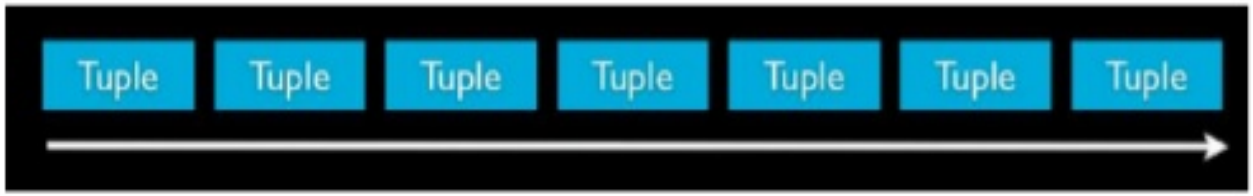


Tuple and Stream

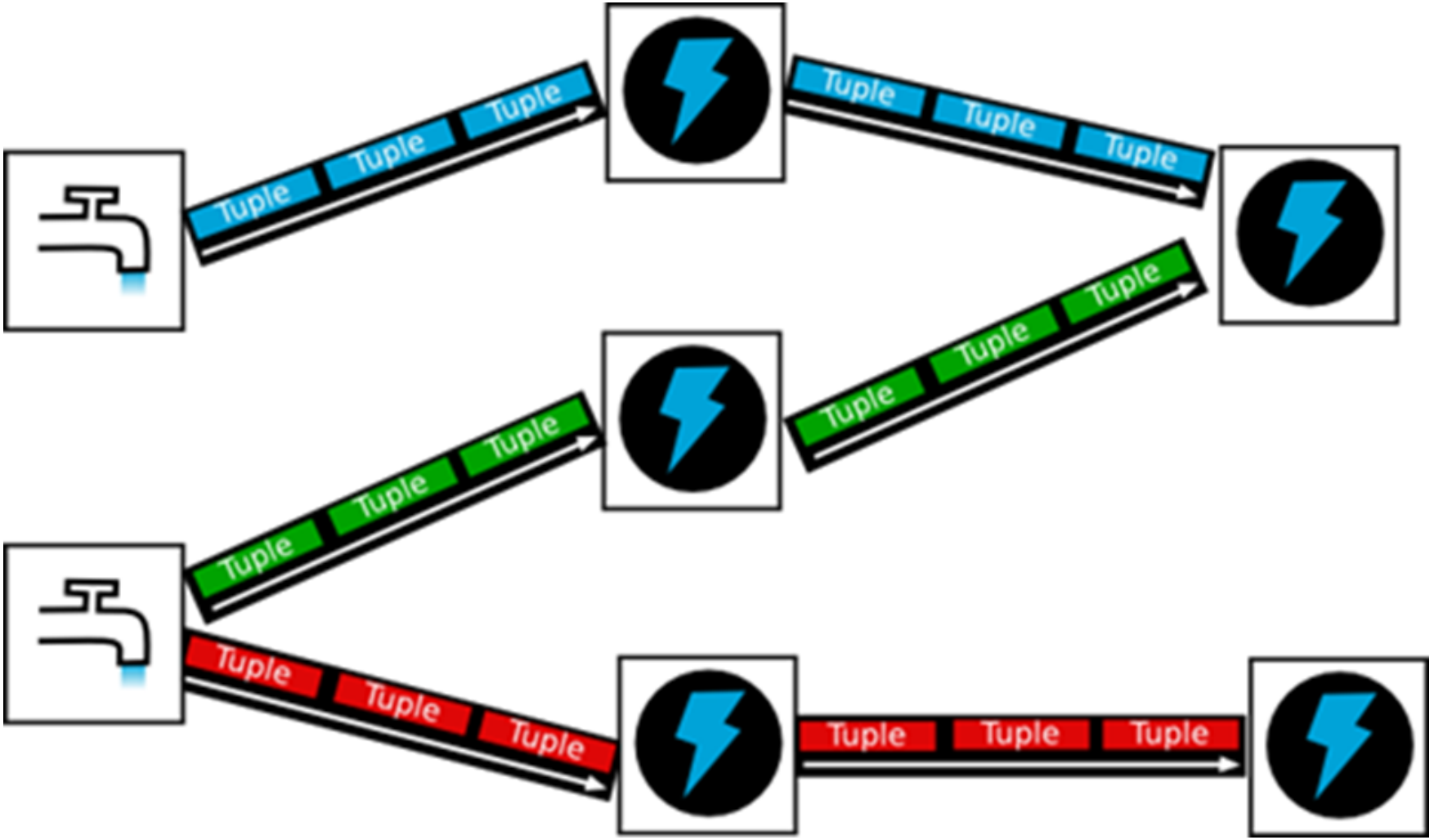
- **What is a Tuple?**
  - Fundamental data structure in Storm. Is a named list of values that can be of any data type.

```
new Values(driverId, truckId, eventTime, eventType, longitude, latitude, eventKey, correlationId);
```

- **What is a Stream?**
  - An unbounded sequences of tuples.
  - Core abstraction in Storm and are what you "process" in Storm



Inside Topology



Command Line Tool

storm jar topology-jar-path class ...

storm kill topology-name [-w wait-time-secs]

storm deactivate topology-name

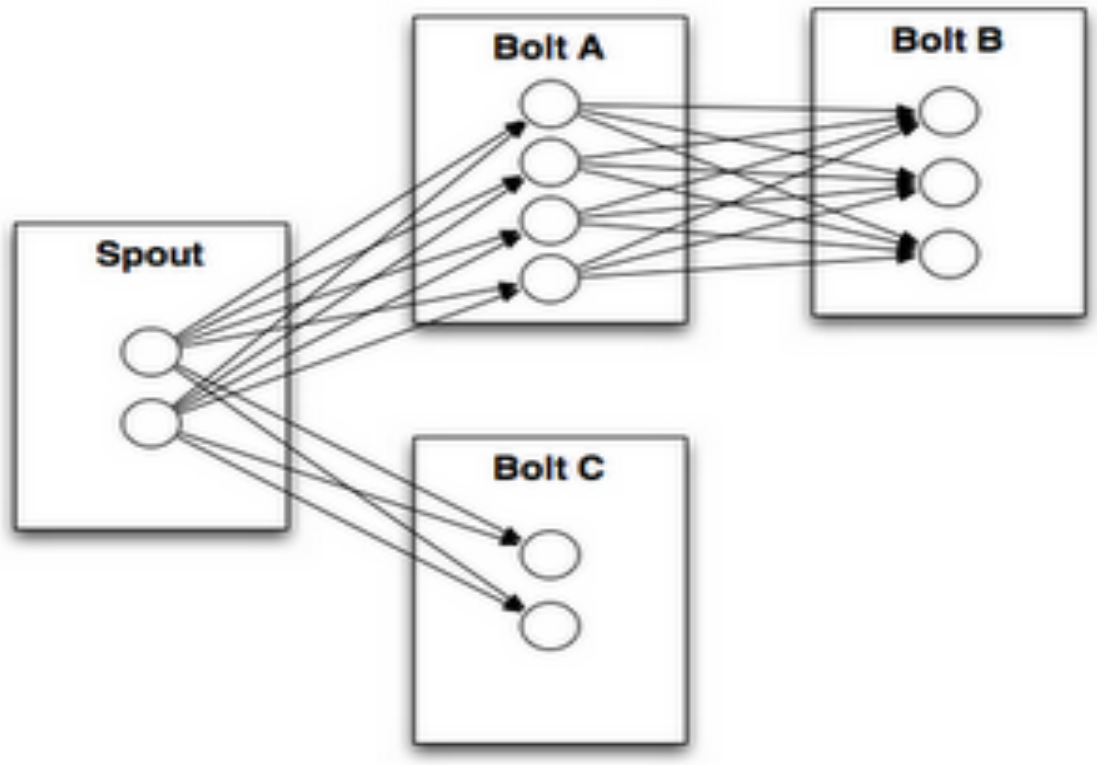
storm activate topology-name

storm rebalance topology-name [-w wait-time-secs]

Storm Demo

Storm Grouping

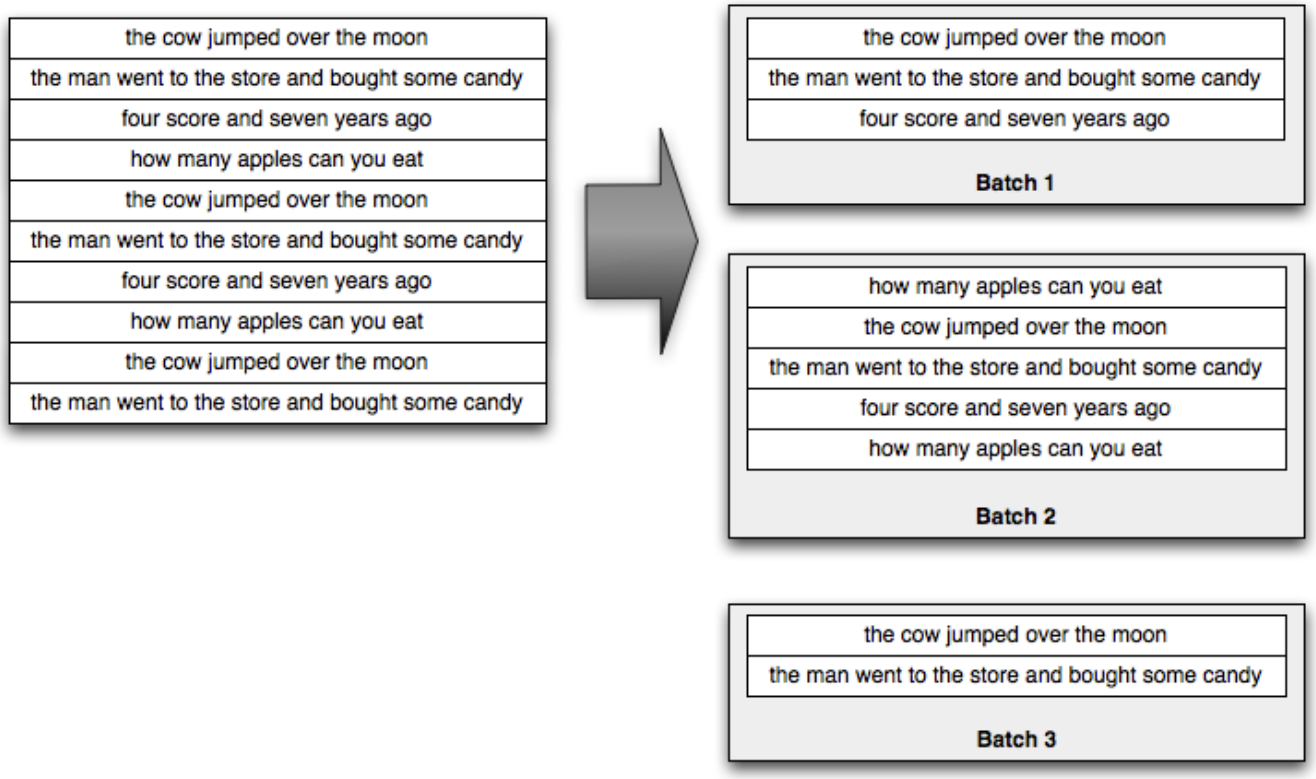
- **Shuffle grouping**
- **Fields grouping**
- **Partial Key grouping**



Storm Trident

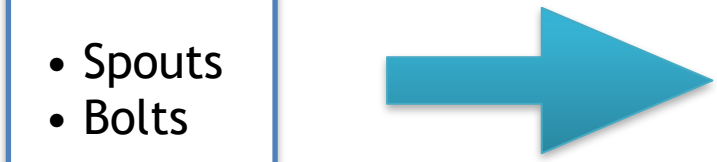
- Exactly-Once Processing
- Batch Processing
- Ordered State Updates
- Fast, Persistent Aggregation

Batched Processing

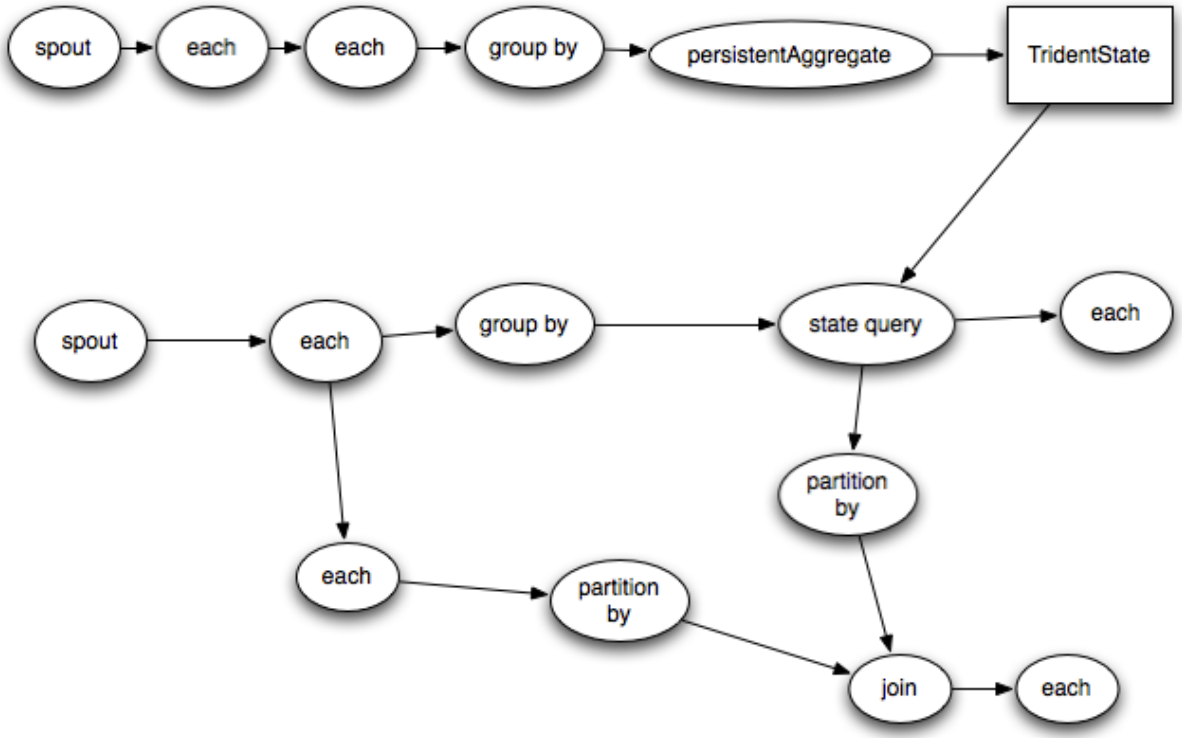


Trident Concepts

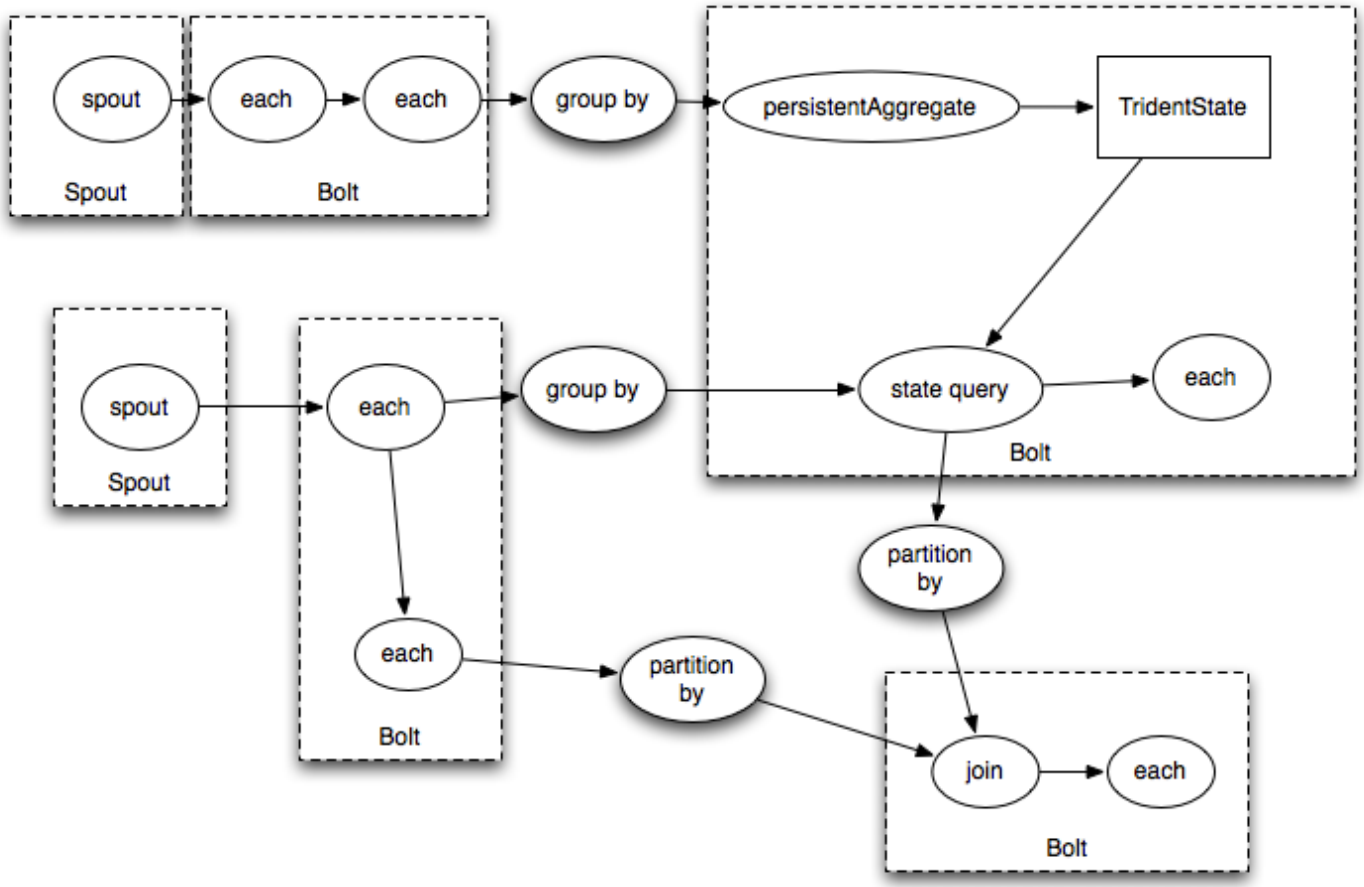
- Spouts
- Bolts



- Streams
- Operations
- States



Under the Hood



Trident Demo

Reference

- http://storm.apache.org/documentation/Documentation.html
- https://storm.apache.org/documentation/Tutorial.html
- http://nathanmarz.github.io/storm/doc-0.8.1/index.html