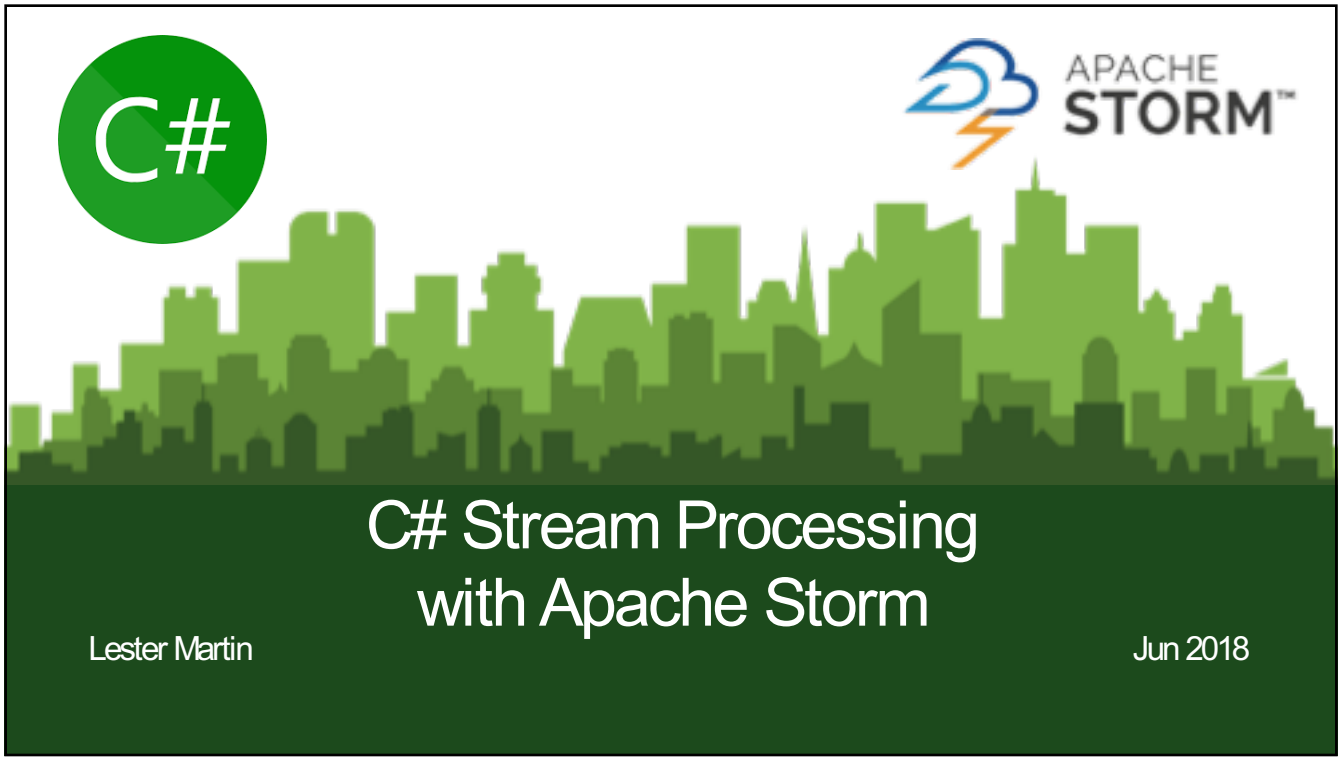


Connection before Content

Lester Martin – Hadoop/Spark/Storm Trainer & Consultant

lester.martin@gmail.com

https://about.me/lestermartin

(links to blog, twitter, github, LI, FB, etc)

Agenda

- **What is Storm?**
- **Conceptual Model**
- ***DEMO: Deploy Storm Topology***
- **Compile Time**
- ***DEMO: Develop Word Count Example***
- **Runtime**
- ***DEMO: Monitor Storm Topology***
- **Additional Features**

What is Storm?

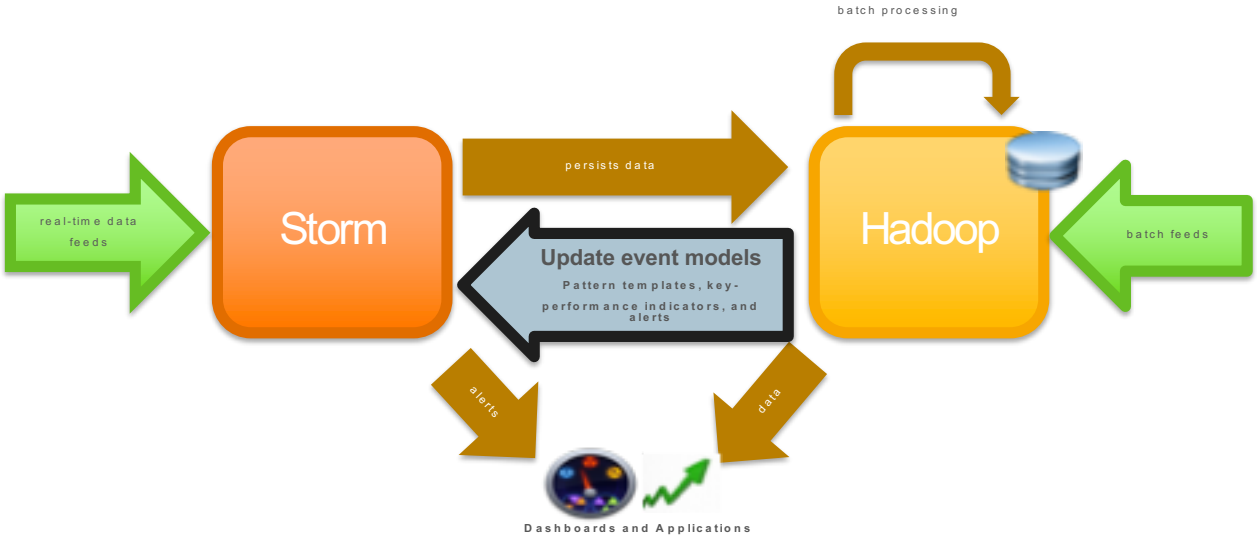
Apache Storm Brief History

- 2010 – First Streaming Framework (Backtype)
- 2011 – Acquired and Deployed at Twitter
- 2013 – Open Sourced into Apache
- Present – Large Scale Production Deployments
  - Yahoo 3500+ Nodes
  - Alibaba 1PB of Data per Day
  - Public list at http://storm.apache.org/Powered-By.html

Storm is …

- Streaming
  - Key enabler of the Lambda Architecture
- Fast
  - Clocked at 1M+ messages per second per node
- Scalable
  - Thousands of workers per cluster
- Fault Tolerant
  - Failure is expected, and embraced
- Reliable
  - Guaranteed message delivery
  - Exactly-once semantics

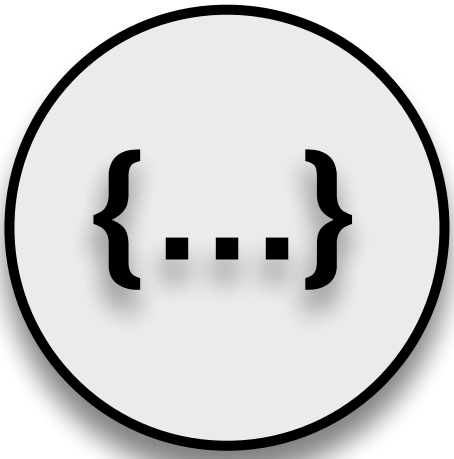
Storm in the Lambda Architecture



Page 7

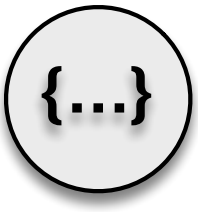
Conceptual Model

TUPLE



Tuple

- Unit of work to be processes
- Immutable ordered set of serializable values
- Fields must have assigned name



Stream

- Core abstraction of Storm
- Unbounded sequence of Tuples

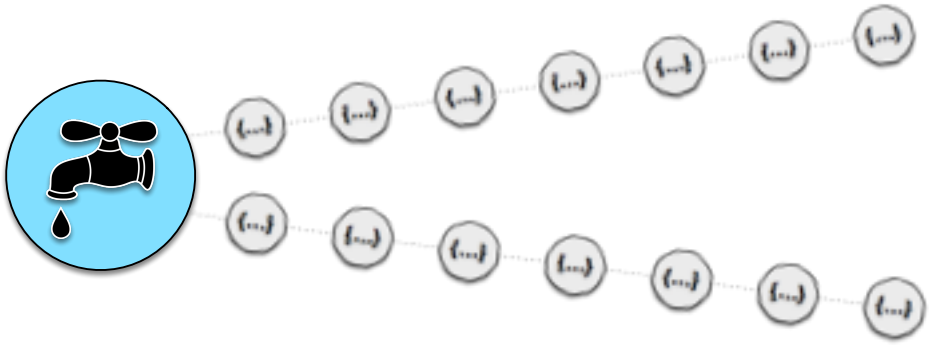
Page 11

SPOUT

Page 12

Spout

- Source of Streams
- Wrap an event source and emit Tuples



Page 13

Message Queues

Message queues are often the source of the data processed by Storm

Storm Spouts integrate with many types of message queues

Page 14

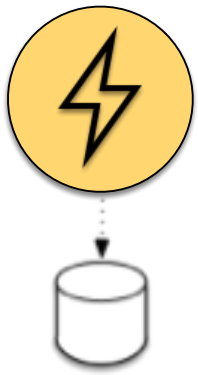
BOLT

Bolt

- Core unit of computation
- Receive Tuples and **do stuff**
- *Optionally,* emit additional Tuples

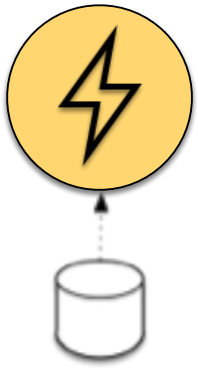
Bolt

- Write to a data store



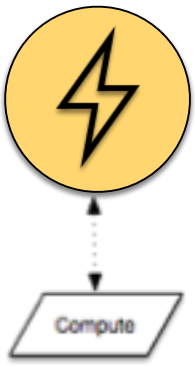
Bolt

- Read from a data store



Bolt

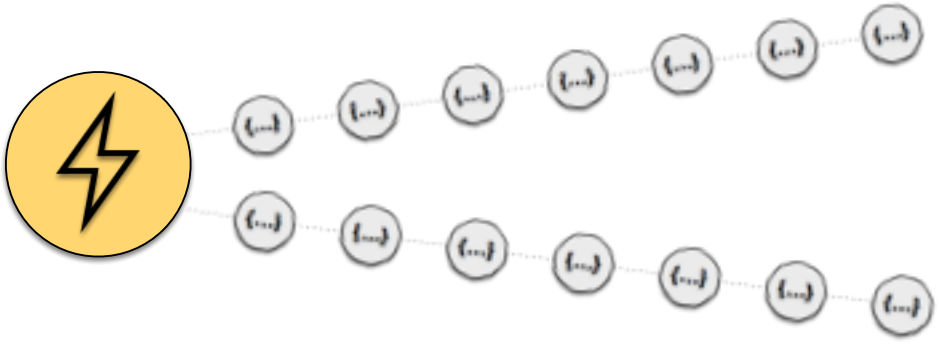
- Perform arbitrary computation



Page 19

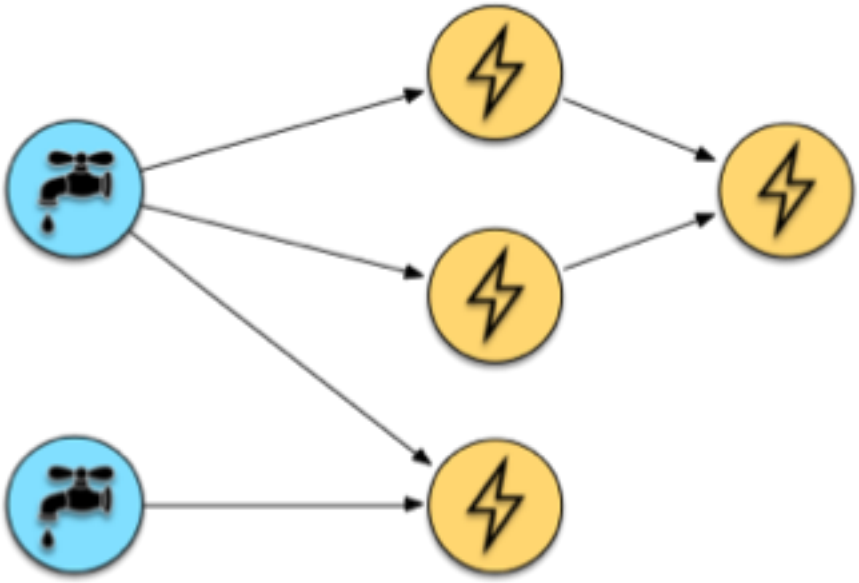
Bolt

- (Optionally) Emit additional Stream(s)



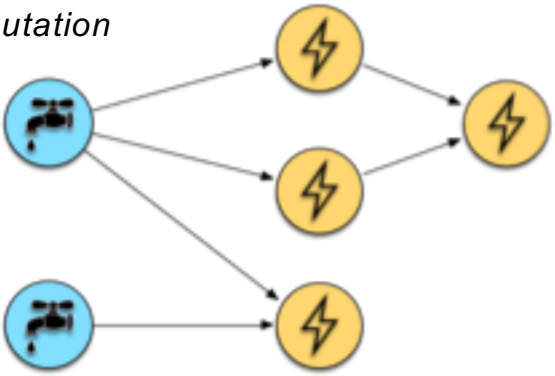
Page 20

TOPOLOGY



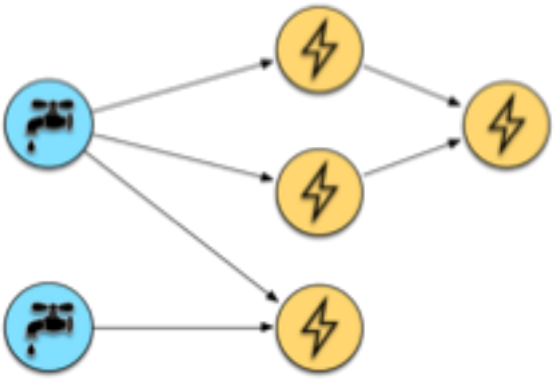
Topology

- DAG of Spouts and Bolts
- Data Flow Representation
- *Streaming Computation*



Topology

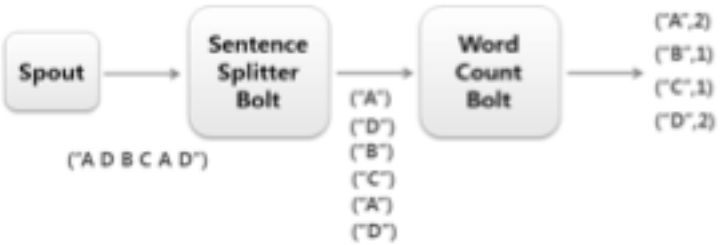
- Storm executes Spouts and Bolts as ***Tasks*** that run in parallel on multiple machines



Page 23

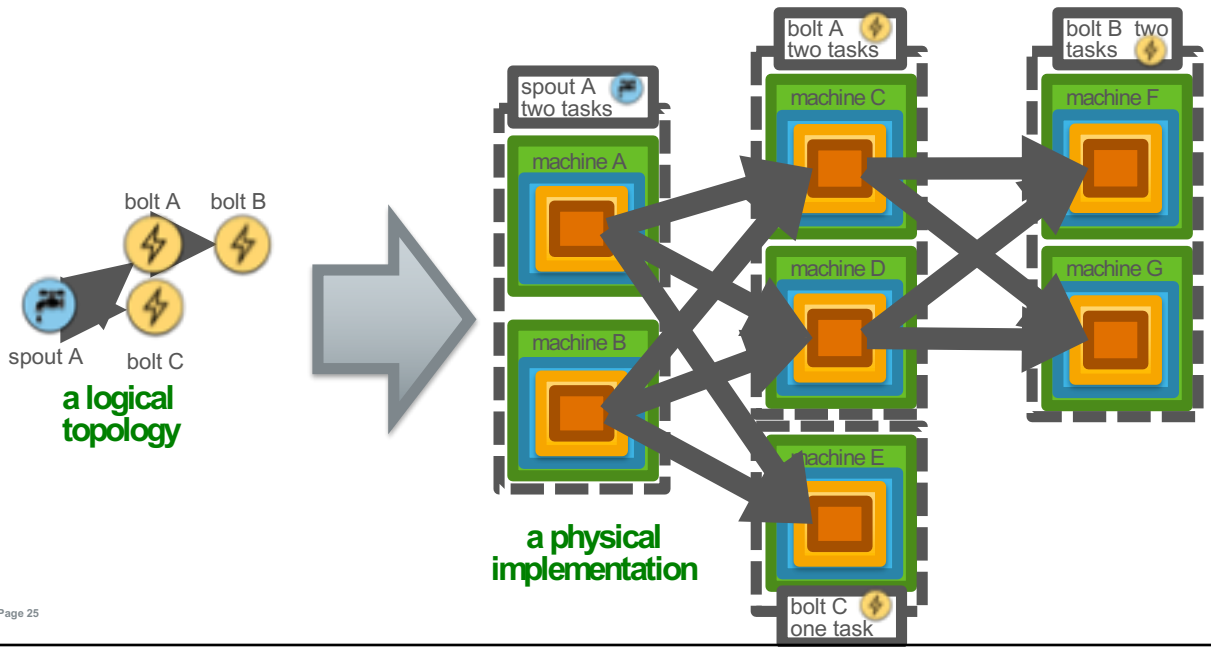
DEMO

Deploy Word Count Topology



Page 24

Parallel Execution of Topology Components



Page 25

Stream Groupings

Stream Groupings determine how Storm routes Tuples between Tasks

Grouping Type	Routing Behavior
Shuffle	Randomized round-robin (evenly distribute load to downstream Bolts)
Fields	Ensures all Tuples with the same Field value(s) are always routed to the same Task
All	Replicates Stream across all the Bolt's Tasks (use with care)
Other options	Including custom RYO grouping logic

Page 26

```
@Override
public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("sentence"));
}
```

Compile Time

Example Spout Code (1 of 2)

Name of the spout class.

Storm spout class used as a "template".

```
public class RandomSentenceSpout extends BaseRichSpout {
    SpoutOutputCollector _collector;
    Random _rand;

    @Override
    public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {
        _collector = collector;
        _rand = new Random();
    }
    @Override
    public void nextTuple() {
        Utils.sleep(100);
        String[] sentences = new String[]{ "the cow jumped over the moon", "an apple a day keeps
                the doctor away", "four score and seven years ago", "snow white and the seven dwarfs",
                "i am at two with nature" };
        String sentence = sentences[_rand.nextInt(sentences.length)];

        _collector.emit(new Values(sentence));
    }
```

Storm uses `open` to open the spout and provide it with its configuration, a context object providing information about components in the topology, and an output collector used to emit tuples.

Storm uses `nextTuple` to request the spout emit the next tuple.

The spout uses `emit` to send a tuple to one or more bolts.

Continued next page…

Example Spout Code (2 of 2)

Continued…

```
@Override
public void ack(Object id) {
}
@Override
public void fail(Object id) {
}
@Override
public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("sentence"));
}
}
```

Storm calls the spout's `ack` method to signal that a tuple has been fully processed.

Storm calls the spout's `fail` method to signal that a tuple has not been fully processed.

The `declareOutputFields` method names the fields in a tuple.

Page 29

Example Bolt Code

The `prepare` method provides the bolt with its configuration and an `OutputCollector` used to emit tuples.

The `execute` method receives a tuple from a stream and emits a new tuple. It also provides an `ack` method that can be used after successful delivery.

The `cleanup` method releases system resources when bolt is shut down.

Name of the bolt class.

Bolt class used as a "template."

```
public static class ExclamationBolt extends BaseRichBolt {
    OutputCollector _collector;

    public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
        _collector = collector;
    }

    public void execute(Tuple tuple) {
        _collector.emit(tuple, new Values(tuple.getString(0) + "!!!"));
        _collector.ack(tuple);
    }

    public void cleanup(); {
    }

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word"));
    }
}
```

Names the fields in the output tuples. More detail later.

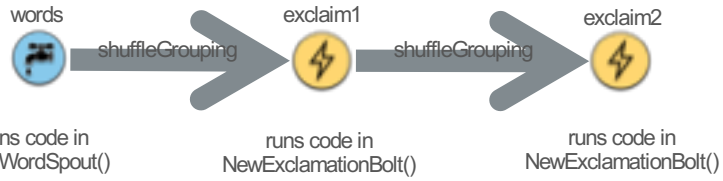
Page 30

Example Topology Code

This code…

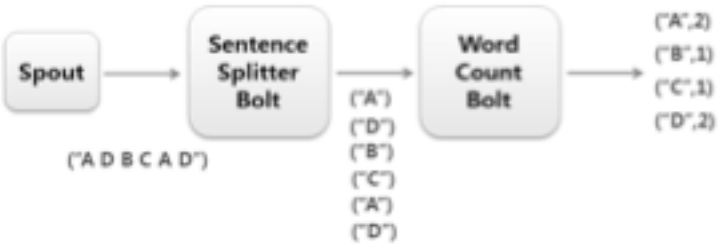
```
public static main(String[] args) throws exception {

    TopologyBuilder builder = new TopologyBuilder();
    builder.setSpout("words", new TestWordSpout());
    builder.setBolt("exclaim1", new NewExclamationBolt()).shuffleGrouping("words");
    builder.setBolt("exclaim2", new NewExclamationBolt()).shuffleGrouping("exclaim1");

    Config conf = new Config();

    StormSubmitter.submitTopology("add-exclamation", conf, builder.createTopology());
}
```

…builds this Topology.

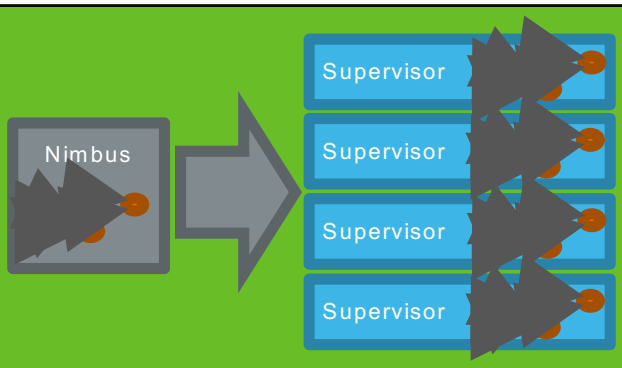


DEMO

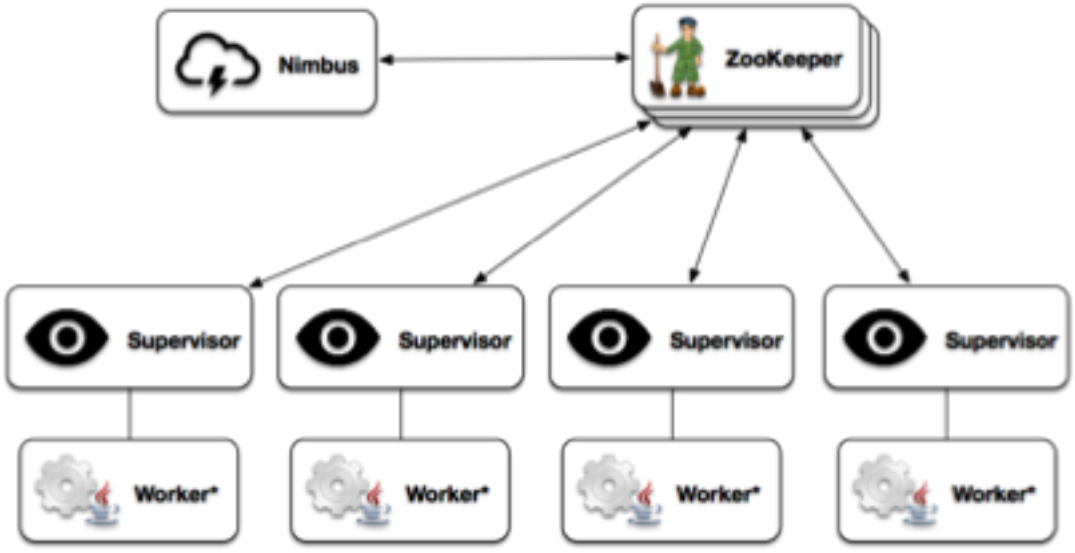
Develop Word Count Topology



Runtime

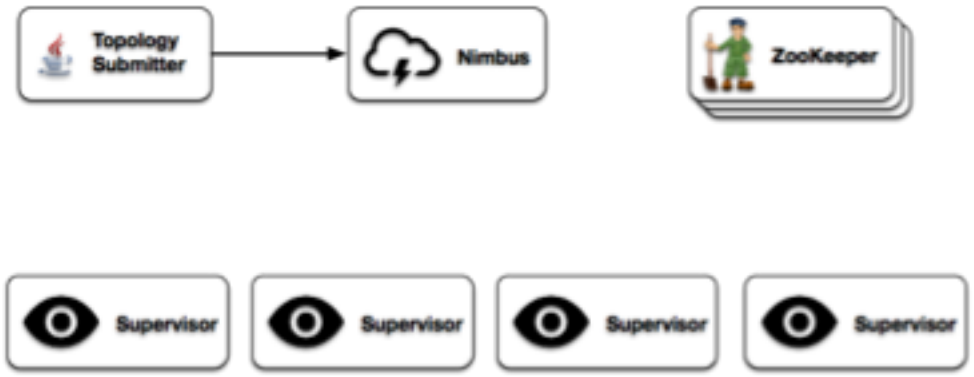


Physical View



Page 34

Topology Deployment

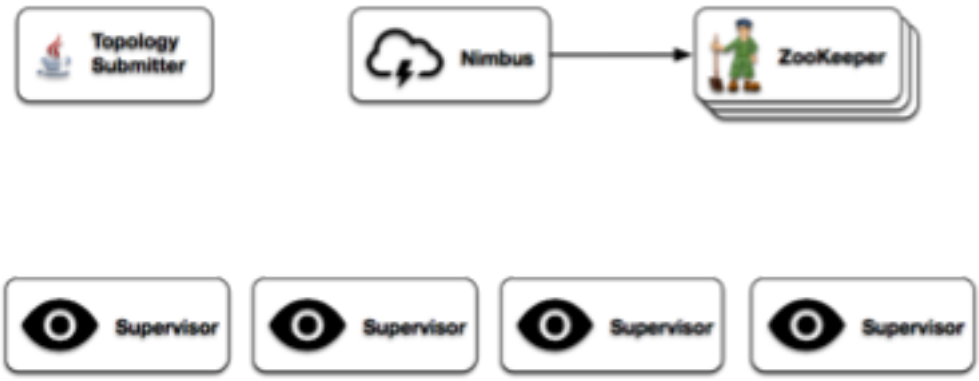


Topology Submitter uploads topology:

- topology.jar
- topology.ser
- conf.ser

Page 35

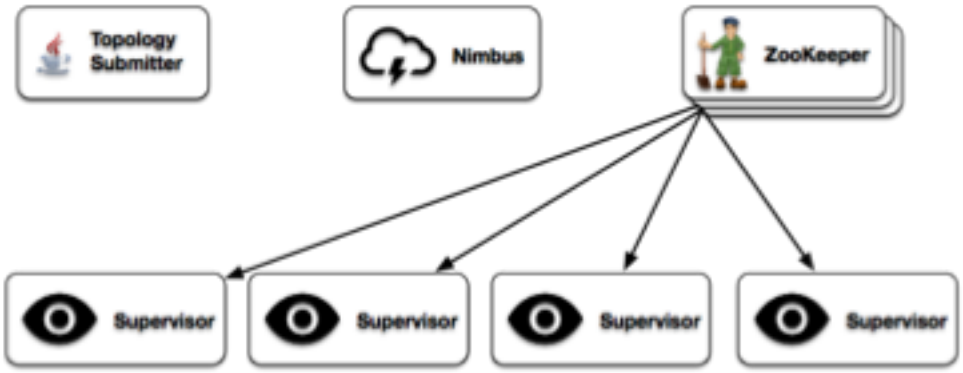
Topology Deployment



Nimbus calculates assignments and sends to Zookeeper

Page 36

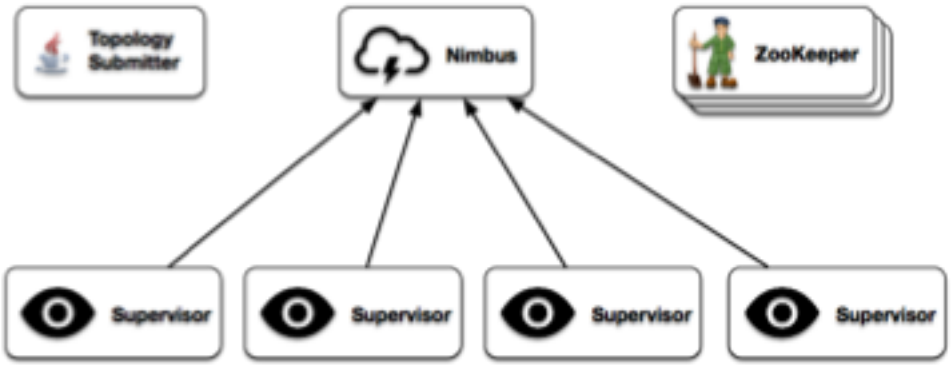
Topology Deployment



Supervisor nodes receive assignment information via Zookeeper watches

Page 37

Topology Deployment

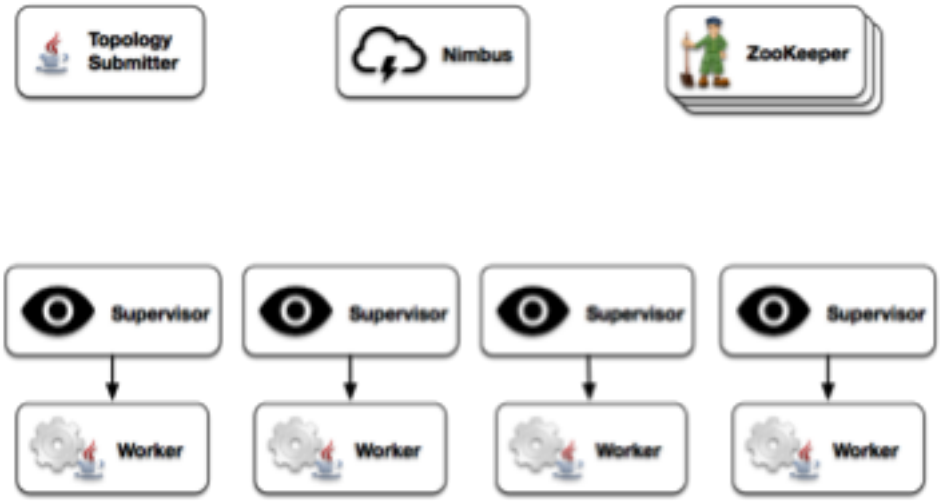


Supervisor nodes download topology from Nimbus:

- topology.jar
- topology.ser
- conf.ser

Page 38

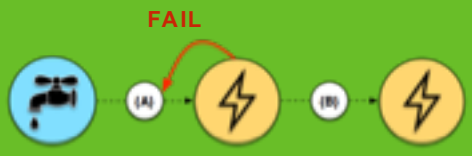
Topology Deployment



Supervisors spawn workers (JVM processes)

DEMO

Monitor Topology in Storm Cluster



Additional Features

Local Versus Distributed Storm Clusters

The topology program code submitted to Storm using `storm jar` is different when submitting to local mode versus a distributed cluster.

The `submitTopology` method is used in both cases.

- The difference is the class that contains the `submitTopology` method.

```
Config conf = new Config();
LocalCluster cluster = new LocalCluster();
LocalCluster.submitTopology("mytopology", conf, topology);
```

Instantiate a local cluster object.

Same method name, different classes.

Submit a topology to a local cluster.

Submit a topology to a distributed cluster.

```
Config conf = new Config();
StormSubmitter.submitTopology("mytopology", conf, topology);
```

Page 42

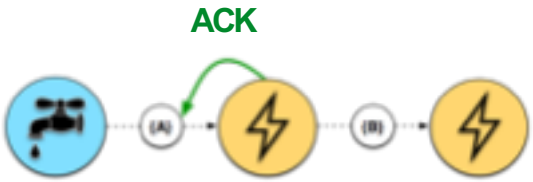
Reliable Processing

Bolts may emit Tuples ***Anchored*** to one received.
Tuple “B” is a descendant of Tuple “A”

Reliable Processing

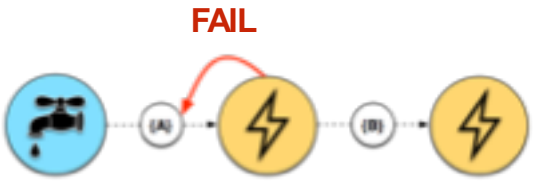
Multiple ***Anchorings*** form a Tuple tree
(bolts not shown)

Reliable Processing



Bolts can ***Acknowledge*** that a tuple has been processed successfully.

Reliable Processing



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

Reliable Processing

Any failure in the Tuple tree will trigger a replay of the original tuple

More *Stuff*

- Topology description/deployment options
  - Flux
  - Storm SQL
- Polyglot development
- Micro-batching with Trident
- Fault tolerance & deployment isolation
- Integrations
  - Messaging; Event Hubs, Kafka, Redis, Kestrel, Kinesis, MQTT, JMS
  - Databases; HBase, Hive, Druid, Cassandra, MongoDB, JDBC
  - Search Engines; Solr, Elasticsearch
  - HDFS
  - *And more!*

Questions?

Lester Martin – Hadoop/Spark/Storm Trainer & Consultant
lester.martin@gmail.com https://about.me/lestermartin

Preso & Artifacts available at https://bit.ly/2KleVkb

THANKS FOR YOUR TIME!!