Prashanth Babu V V
@P7h

BIG DATA, REAL-TIME PROCESSING AND STORM

Workshop at The Fifth Elephant, 2013

https://twitter.com/P7h

https://github.com/P7h

http://P7h.org

http://about.me/Prashanth

Jeremy Kahn
@jeremyckahn

Protip: Nobody is really "qualified" to give tech talks. We're all exploring and figuring it out. Just share what you've learned.

Prerequisites for Workshop

- Laptop with any OS
- JDK v7.x installed
- Maven v3.0.5+ installed
- IDE [either Eclipse with m2eclipse plugin or IntelliJ IDEA]
- Created Twitter app for retrieving tweets
- Cloned or downloaded Storm Projects from my GitHub Account:
  - https://github.com/P7h/StormWordCount
  - https://github.com/P7h/StormTweetsWordCount

Agenda

- Big Data
- Batch vs. Real-time processing
- Intro to Storm
- Companies using Storm
- Storm Dependencies
- Storm Concepts
- Anatomy of Storm Cluster
- Live coding a use case using Storm Topology
- Storm vs. Hadoop


BIG DATA

WHAT IS BIG DATA?

VOLUME

Large amounts of data.

VELOCITY

Needs to be analyzed **quickly.**

VARIETY

Different types of structured and unstructured data.

Key questions enterprises are asking about Big Data:

How to store and protect big data?

How to backup and restore big data?

How to organize and catalog the data that you have backed up?

How to keep costs low while ensuring that all the critical data is available when you need it?

WHAT ARE THE VOLUMES OF DATA THAT WE ARE SEEING TODAY?

30 billion pieces of content were added to Facebook this past month by 600 million plus users.

Zynga processes 1 petabyte of content for players every day; a volume of data that is unmatched in the social game industry.

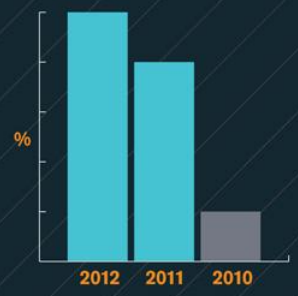
More than 2 billion videos were watched on YouTube... yesterday.

The average teenager sends **4,762 text messages** per month.

32 billion searches were performed last month... on Twitter.

Source: Gartner

Everyday business and consumer life creates 2.5 quintillion bytes of data per day.



90% of the data in the world today has been created in the last two years alone.

Source: IBM

WHAT DOES THE FUTURE LOOK LIKE?

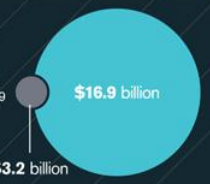
Worldwide IP traffic will **quadruple by 2015.**

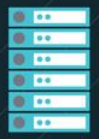
By 2015, nearly **3 billion people**

will be online, pushing the data created and shared to nearly **8 zettabytes.**

HOW IS THE MARKET FOR BIG DATA SOLUTIONS EVOLVING?

A new IDC study says the market for big technology and services will grow from $3.2 billion in 2010 to $16.9 billion in 2015. **That's a growth of 40%** CAGR.





58% of respondents expect their companies to increase spending on server backup solutions and other big data-related initiatives within the next three years.

Source: Economist Business Unit

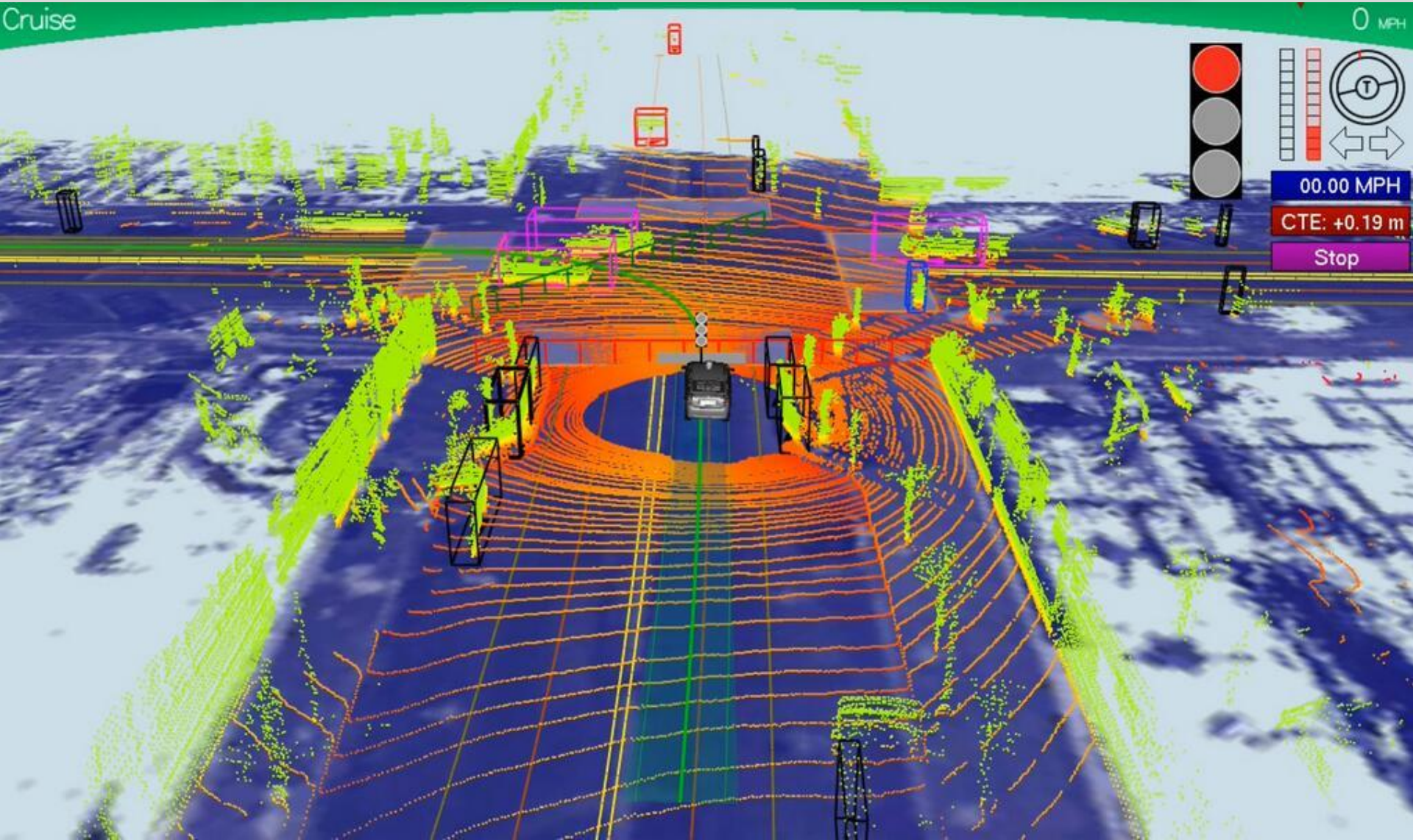
2/3rds of surveyed businesses in North America said big data will become a concern for them within the next five years.

Source: Economist Business Unit

Asigra.

Bill Gross
@Bill_Gross

Google's Self-Driving Car gathers almost 1 GB per SECOND. Here's what it "sees" making a left turn: pic.twitter.com/vZCWhEeBmF



Data overload ***every*** second

Batch vs. Real-time Processing

- Batch processing
  - Gathering of data and processing as a group at one time.
- Real-time processing
  - Processing of data that takes place as the information is being entered.

Event Processing

- Simple Event Processing
  - Acting on a single event, filter in the ESP
- Event Stream Processing
  - Looking across multiple events
- Complex Event Processing
  - Looking across multiple events from multiple event streams

INTRO TO STORM

Storm

- Created by Nathan Marz @ BackType
  - Analyze tweets, links, users on Twitter
- Open sourced on 19th September, 2011
  - Eclipse Public License 1.0
  - Storm v0.5.2
  - 16k Java and 7k Clojure LOC
- Latest Updates
  - Current stable release v0.8.2 released on 11th January, 2013
  - Major core improvements planned for v0.9.0
  - Storm will be an Apache Project [soon..]

Storm

- Open source distributed real-time computation system
- Hadoop of real-time
- Fast
- Scalable
- Fault-tolerant
- Guarantees data will be processed
- Programming language agnostic
- Easy to set up and operate
- Excellent documentation

Nathan Marz
@nathanmarz

Storm is now one of the top 50 most starred projects on Github
github.com/popular/starred

Nathan Marz
@nathanmarz

Just clocked Storm 0.8.0 at 1.64 million tuples processed per second per node on an internal Twitter cluster

Storm
@stormprocessor

Storm 0.8.0 has 3x better throughput (measured 300K 100 byte tuples/node/sec on EC2 c1.xlarge machines). Still more improvements possible

Polyglotism (language agnostic) – Clojure, Java, Python, Ruby, PHP, Perl, … and yes, even JavaScript

file | 9 lines (7 sloc) | 0.207 kb

Edit Raw Blame History Delete

```python
import storm

class SplitSentenceBolt(storm.BasicBolt):
    def process(self, tup):
        words = tup.values[0].split(" ")
        for word in words:
          storm.emit([word])

SplitSentenceBolt().run()
```

https://github.com/nathanmarz/storm-starter/blob/master/multilang/resources/splitsentence.py

file | 11 lines (9 sloc) | 0.183 kb

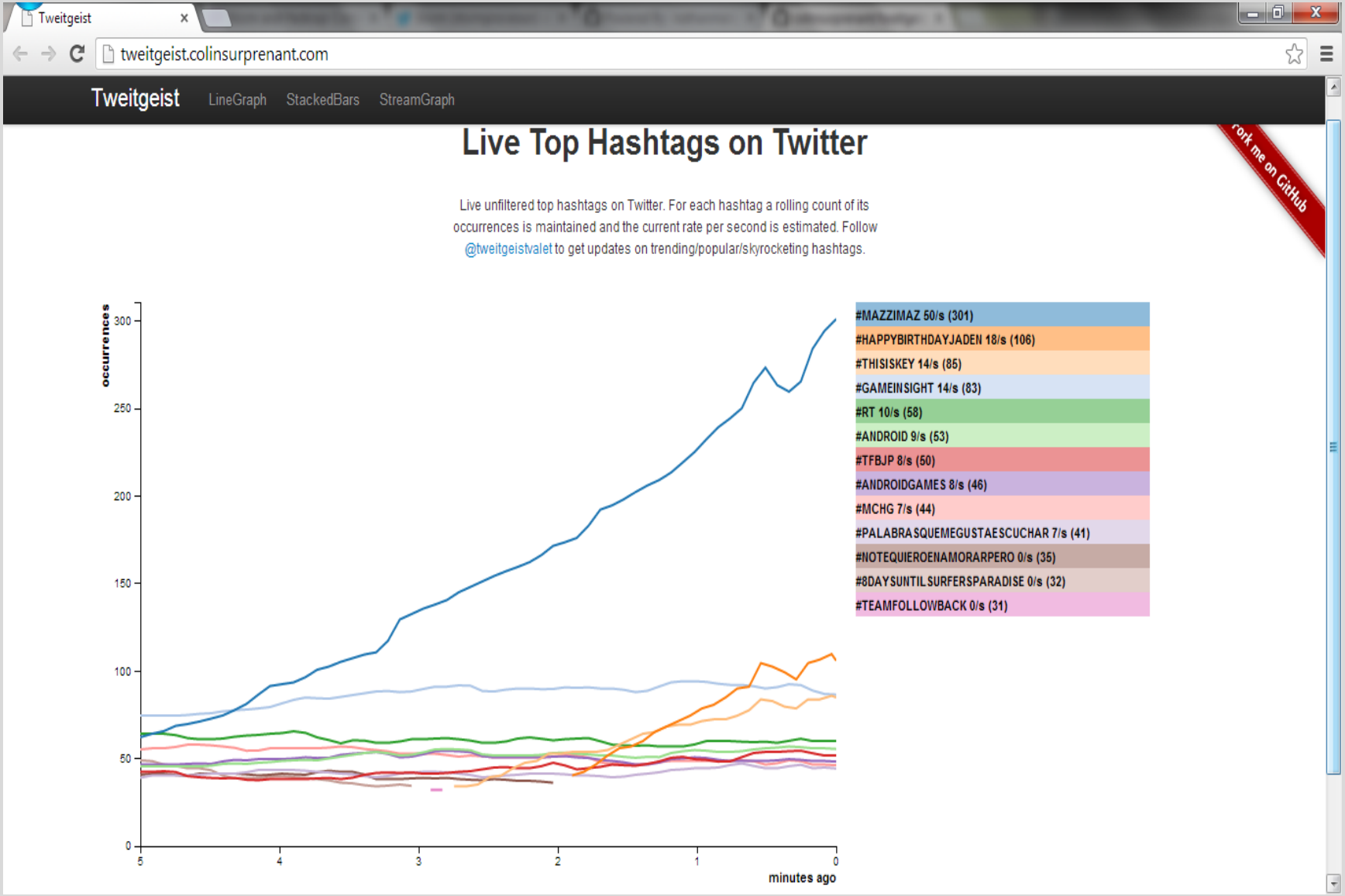
Edit Raw Blame History Delete

```ruby
require "./storm"

class SplitSentenceBolt < Storm::Bolt
  def process(tup)
    tup.values[0].split(" ").each do |word|
      emit([word])
    end
  end
end

SplitSentenceBolt.new.run
```

https://github.com/nathanmarz/storm-starter/blob/master/multilang/resources/splitsentence.rb

Use cases

- Real-time analytics
- Stream processing
- Online machine learning
- Continuous computation
- Distributed RPC
- Extract, Transform and Load (ETL)



http://tweitgeist.colinsurprenant.com/

Companies using Storm

https://github.com/nathanmarz/storm/wiki/Powered-By

- ✓ Real-time analytics
- ✓ Personalization
- ✓ Search
- ✓ Revenue optimization
- ✓ Monitoring
- ✓ Discovery
- ✓ Capacity Planning

- ✓ Content search
- ✓ Realtime analytics
- ✓ Generating custom magazine feeds
- ✓ Integrated with Elastic Search, HBase, Hadoop and HDFS

- ✓ Real-time scoring
- ✓ Moments generation pipeline
- ✓ Integrated with Kafka queues and HDFS storage

Storm-YARN enables the convergence of Big Data and low-latency processing. Empowers stream / micro-batch processing of user events, content feeds and application logs.

YAHOO! DEVELOPER NETWORK Developer Solutions APIs & Tools Community Search YDN

YDN Blog RSS Vide

Storm-YARN Released as Open Source

By Bobby Evans and Andy Feng – Tue, Jun 11, 2013 10:37 AM EDT

Recommend 40 Tweet 212

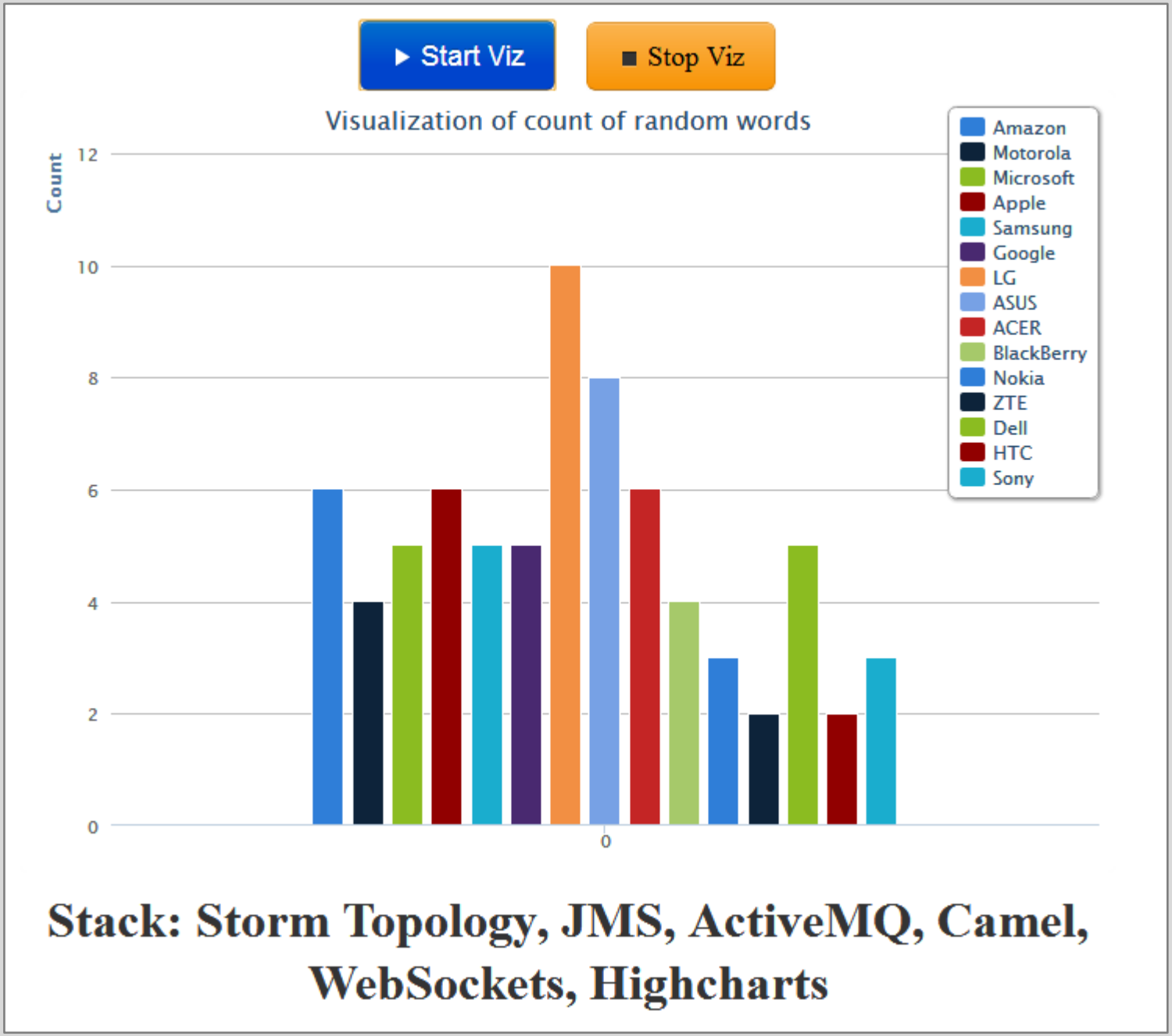
At Yahoo! we have worked on the convergence of Storm with Hadoop, as mentioned in our earlier post. We are pleased to announce that Storm-YARN has been released as open source. Storm-YARN enables Storm applications to utilize the computational resources in a Hadoop cluster along with accessing Hadoop storage resources such as HBase and HDFS.

Motivation

A QUICK DEMO

SCREENSHOT OF DEMO

https://github.com/P7h/storm-camel-example



Stack: Storm Topology, JMS, ActiveMQ, Camel, WebSockets, Highcharts

STORM DEPENDENCIES

Storm under the hood

- Clojure
  - a dialect of the Lisp programming language runs on the JVM, CLR, and JavaScript engines
- Apache Thrift
  - Cross language bridge, RPC; Framework to build services
- ØMQ
  - Asynchronous message transport layer
- Jetty
  - Embedded web server

Storm under the hood

- Apache ZooKeeper
  - Distributed system, used to store metadata
- LMAX Disruptor
  - High performance queue shared by threads
- Kryo
  - Serialization framework
- Misc.
  - SLF4J, Python, Java 5+, JZMQ, JODA, Guava

LMAX EXCHANGE

KRYO

SLF4J Python Java

STORM CONCEPTS

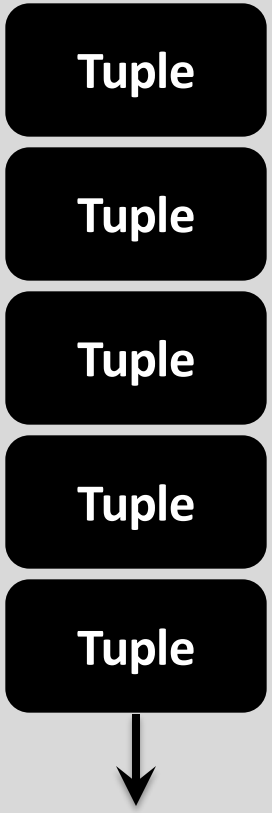
Tuples

- Main data structure in Storm.
- An ordered list of objects.
  - ("user", "Prashanth", "Babu", "Engineer", "Bangalore")
- Key-value pairs – keys are strings, values can be of **<u>any</u>** type.

Tuple

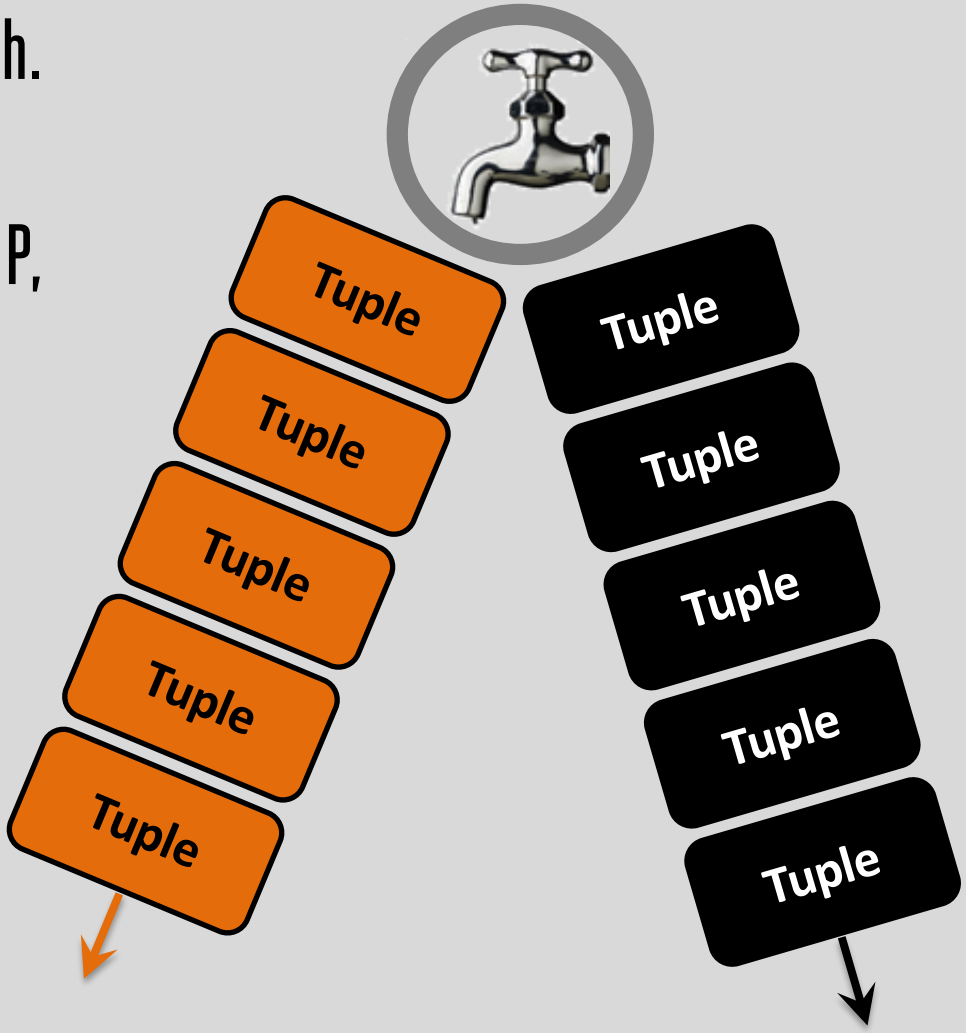
Streams

- Unbounded sequence of tuples.
- Edges in the topology.
- Defined with a schema.



Spouts

- Source of streams.
- Spouts are like sources in a graph.
- Examples are API Calls, log files, event data, queues, Kestrel, AMQP, JMS, Kafka, etc.



BaseRichSpout

backtype.storm.topology.base

Class BaseRichSpout

```
java.lang.Object
  └backtype.storm.topology.base.BaseComponent
      └backtype.storm.topology.base.BaseRichSpout
```

All Implemented Interfaces:

ISpout, IComponent, IRichSpout, java.io.Serializable

Direct Known Subclasses:

DRPCSpout, FeederSpout, MasterBatchCoordinator, SpoutTracker, TestPlannerSpout, TestWordSpout, TransactionalSpoutCoordinator

---

```
public abstract class BaseRichSpout
extends BaseComponent
implements IRichSpout
```

See Also:

Serialized Form

Constructor Summary

BaseRichSpout()

Method Summary

void	ack(java.lang.Object msgId) Storm has determined that the tuple emitted by this spout with the msgId identifier has been fully processed.
void	activate() Called when a spout has been activated out of a deactivated mode.
void	close() Called when an ISpout is going to be shutdown.
void	deactivate() Called when a spout has been deactivated.
void	fail(java.lang.Object msgId) The tuple emitted by this spout with the msgId identifier has failed to be fully processed.

Methods inherited from class backtype.storm.topology.base.BaseComponent
getComponentConfiguration

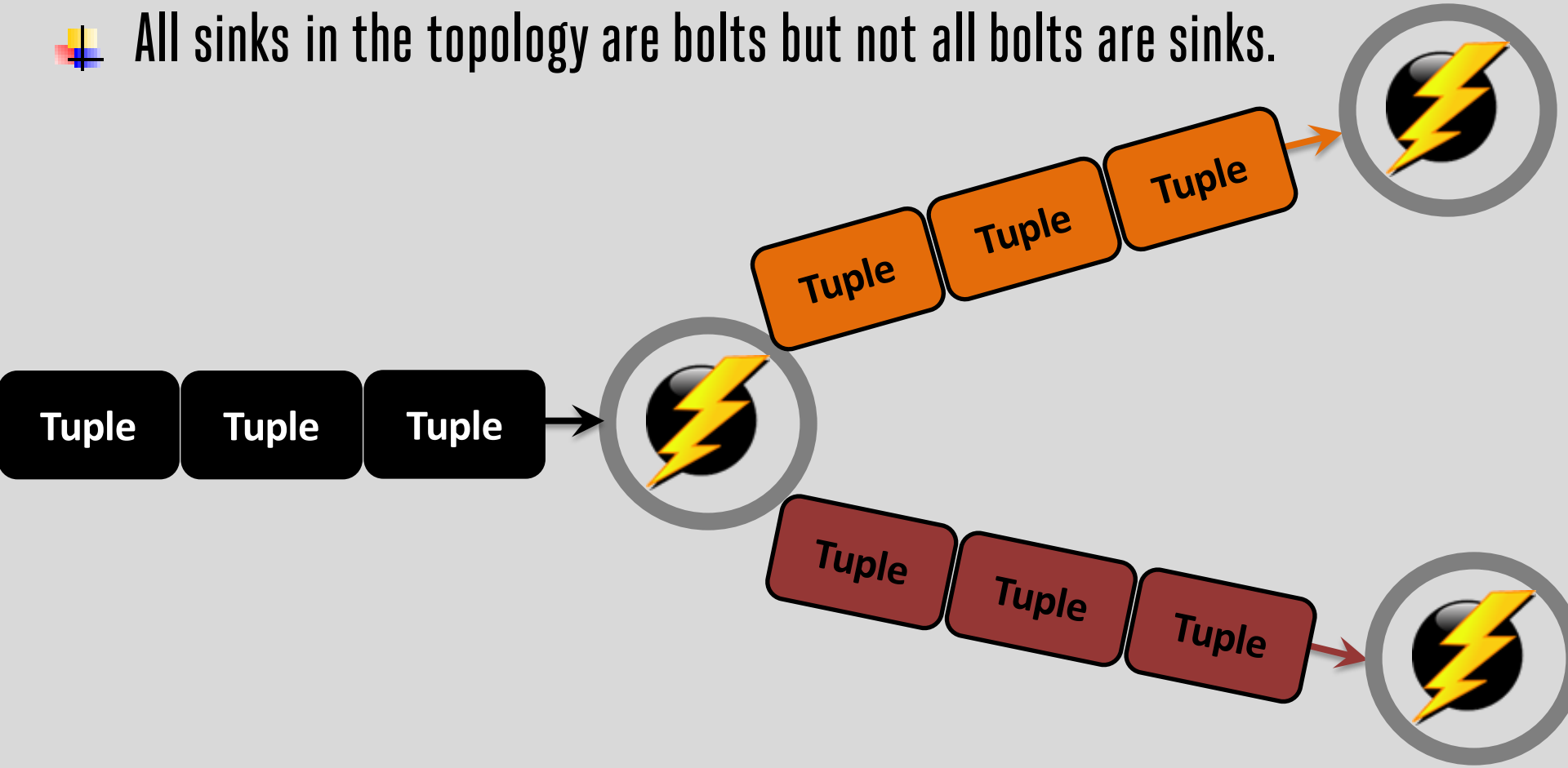
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface backtype.storm.spout.ISpout
nextTuple, open

Methods inherited from interface backtype.storm.topology.IComponent
declareOutputFields, getComponentConfiguration

Bolts

- Process input streams and [might] produce new streams.
- Can do anything i.e. filtering, streaming joins, aggregations, read from / write to databases, APIs, run arbitrary functions, etc.
- All sinks in the topology are bolts but not all bolts are sinks.



Bolts

backtype.storm.topology.base

Class BaseRichBolt

```
java.lang.Object
  └ backtype.storm.topology.base.BaseComponent
      └ backtype.storm.topology.base.BaseRichBolt
```

All Implemented Interfaces:

IBolt, IComponent, IRichBolt, java.io.Serializable

Direct Known Subclasses:

JoinResult, ReturnResults, TestAggregatesCounter, TestGlobalCount, TestPlannerBolt

---

```
public abstract class BaseRichBolt
extends BaseComponent
implements IRichBolt
```

See Also:

Serialized Form

Constructor Summary

BaseRichBolt()

Method Summary

void	cleanup() Called when an IBolt is going to be shutdown.

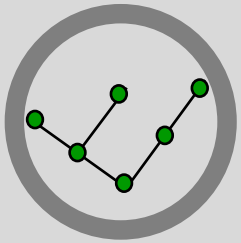
Methods inherited from class backtype.storm.topology.base.BaseComponent
getComponentConfiguration

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface backtype.storm.task.IBolt
execute, prepare

Methods inherited from interface backtype.storm.topology.IComponent
declareOutputFields, getComponentConfiguration

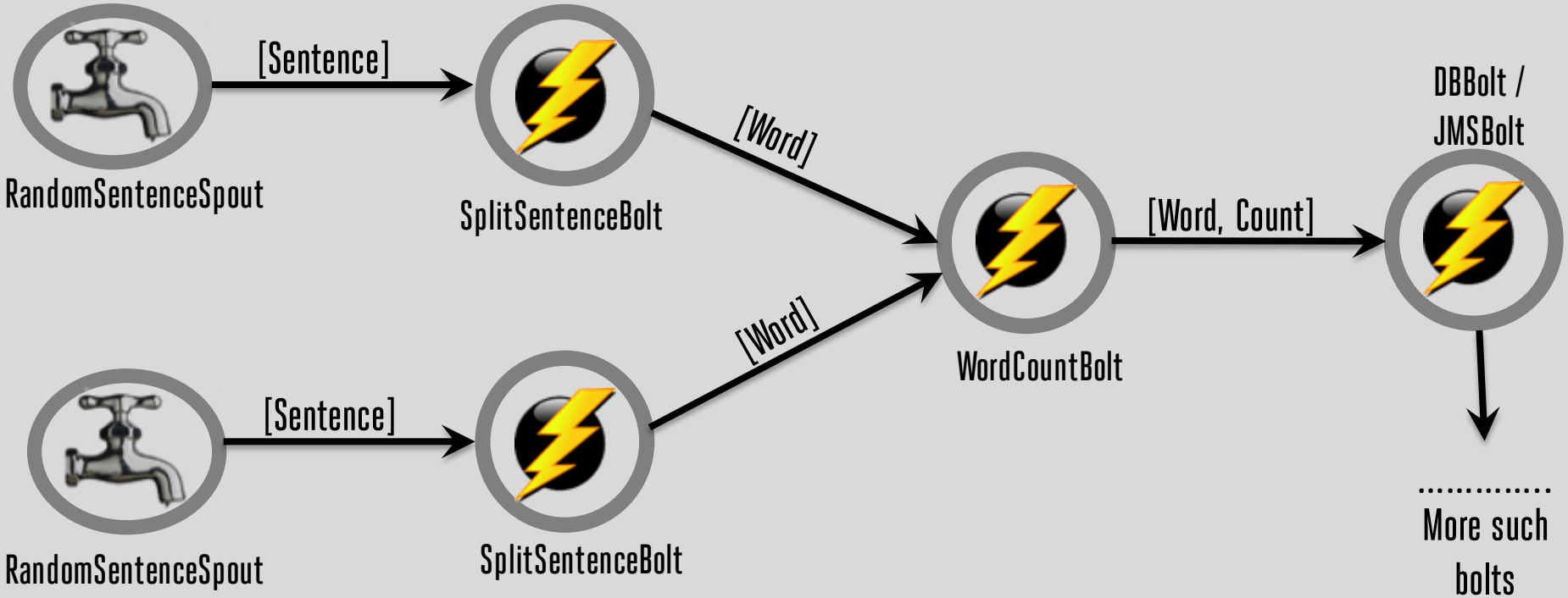
Topology



- Network of spouts and bolts.
- Can be visualized like a graph.
- Container for application logic.
- Analogous to a MapReduce job. *But runs forever.*

Sample Topology

https://github.com/P7h/StormWordCount



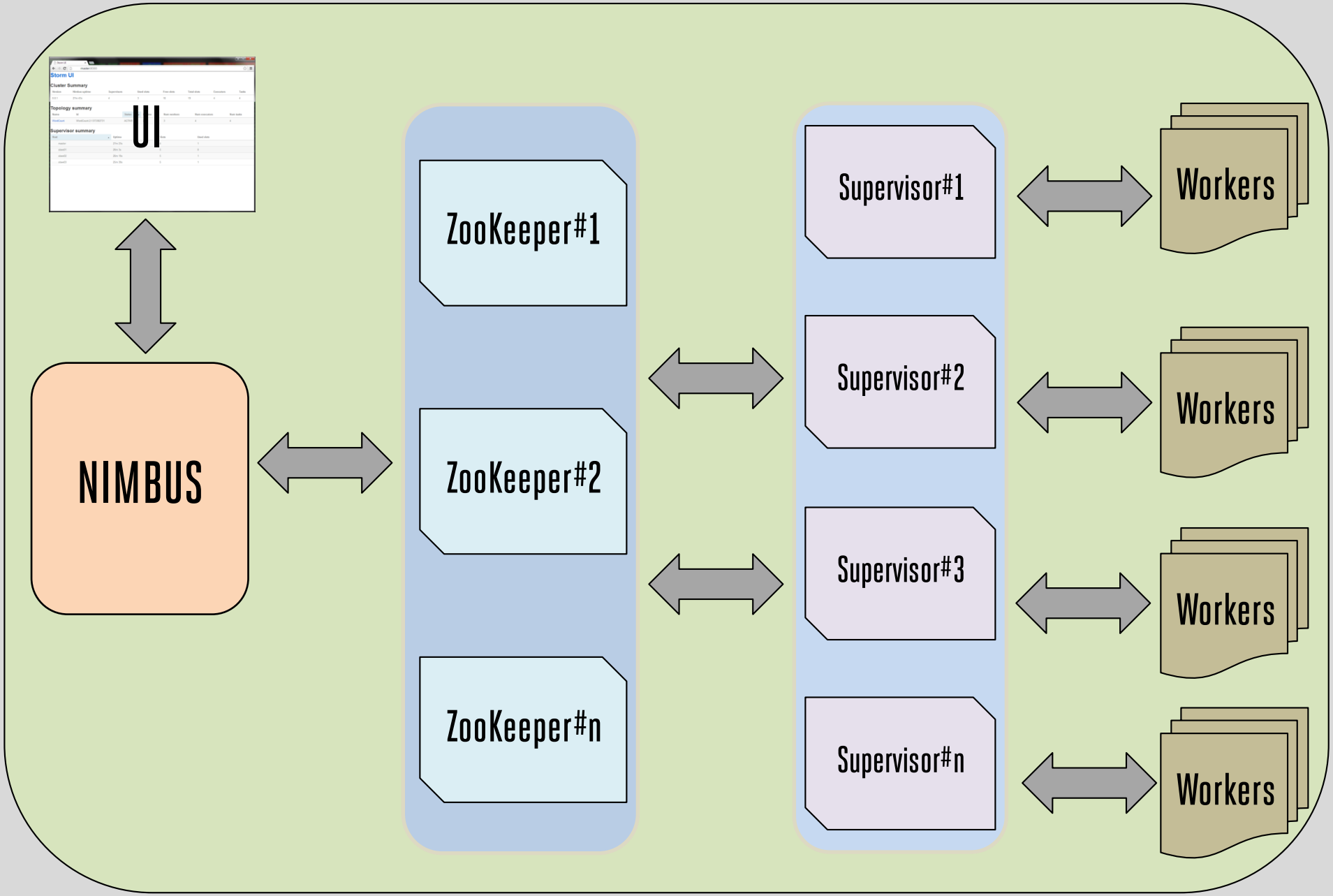
Stream Groupings

- Each Spout or Bolt might be running n instances in parallel [tasks].
- Groupings are used to decide which task in the subscribing bolt, the tuple is sent to.

Grouping	Feature
Shuffle	Random grouping
Fields	Grouped by value such that equal value results in same task
All	Replicates to all tasks
Global	Makes all tuples go to one task
None	Makes Bolt run in the same thread as the Bolt / Spout it subscribes to
Direct	Producer (task that emits) controls which Consumer will receive
Local or Shuffle	If the target bolt has one or more tasks in the same worker process, tuples will be shuffled to just those in-process tasks

ANATOMY OF A STORM CLUSTER

Storm Cluster



Storm Cluster

- Nimbus daemon is the master of this cluster.
  - Manages topologies.
  - Comparable to Hadoop JobTracker.
- Supervisor daemon spawns workers.
  - Comparable to Hadoop TaskTracker.
- Workers are spawned by supervisors.
  - One per port defined in storm.yaml configuration.

Storm Cluster [contd..]

- Task is run as a thread in workers.
- Zookeeper is a distributed system, used to store metadata.
- UI is a webapp which gives diagnostics on the cluster and topologies.
- Nimbus and Supervisor daemons are fail-fast and stateless.
  - State is stored in Zookeeper.

Storm – Modes of operation

- <u>Local mode</u>
  - Develop, test and debug topologies on your local machine.
  - Maven is used to include Storm as a dev dependency for the project.

```
mvn clean compile package && java -jar target/storm-wordcount-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Storm – Modes of operation [contd..]

- Remote [or Production] mode
  - Topologies are submitted for execution on a cluster of machines.
  - Cluster information is added in storm.yaml file.
  - More details on storm.yaml file can be found here: https://github.com/nathanmarz/storm/wiki/Setting-up-a-Storm-cluster#fill-in-mandatory-configurations-into-stormyaml

```
storm jar target/storm-wordcount-1.0-SNAPSHOT.jar
org.p7h.storm.offline.wordcount.topology.WordCountTopology WordCount
```

STORM UI

Storm UI – Cluster Summary

Storm UI

master:8080

Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.8.1	27m 43s	4	3	16	19	4	4

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
WordCount	WordCount-2-1373382731	ACTIVE	27s	3	4	4

Supervisor summary

Host	Uptime	Slots	Used slots
master	27m 23s	4	1
slave01	26m 3s	5	0
slave02	26m 19s	5	1
slave03	25m 39s	5	1

Storm UI – Topology Summary

Storm UI

master:8080/topology/WordCount-2-1373382731

Storm UI

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
WordCount	WordCount-2-1373382731	ACTIVE	50s	3	4	4

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	1000	980	0.000	0	0
3h 0m 0s	1000	980	0.000	0	0
1d 0h 0m 0s	1000	980	0.000	0	0
All time	1000	980	0.000	0	0

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Last error
randomsentencespout	1	1	40	40	0.000	0	0	

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Process latency (ms)	Acked	Failed	Last error
__acker	1	1			0			
splitsentencebolt	1	1	960	940	1.500	40	0	
wordcountbolt	1	1			0			

Hide System Stats

Storm UI – Component Summary

Storm UI

master:8080/topology/WordCount-2-1373382731/component/splitsentencebolt

Storm UI

Component summary

Id	Topology	Executors	Tasks
splitsentencebolt	WordCount	1	1

Bolt stats

Window	Emitted	Transferred	Process latency (ms)	Acked	Failed
10m 0s	1560	1540	1.250	80	0
3h 0m 0s	1560	1540	1.250	80	0
1d 0h 0m 0s	1560	1540	1.250	80	0
All time	1560	1540	1.250	80	0

Input stats (All time)

Component	Stream	Process latency (ms)	Acked	Failed
randomsentencespout	default	1.250	80	0

Output stats (All time)

Stream	Emitted	Transferred
__system	20	0
default	1540	1540

Executors

Id	Uptime	Host	Port	Emitted	Transferred	Process latency (ms)	Acked	Failed
[3-3]	1m 14s	nreapslave03	56785	1560	1540	1.250	80	0

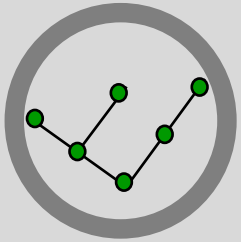
Errors

Time	Error

Hide System Stats

Code Sample – Topology

```
22  public final class WordCountTopology {
23          private static final Logger LOGGER = LoggerFactory.getLogger(WordCountTopology.class);
24          private static final String TOPOLOGY_NAME = "WordCount";
25
26          public static final void main(final String[] args) {
27                  try {
28                          final Config config = new Config();
29                          config.setMessageTimeoutSecs(120);
30                          config.setDebug(false);
31
32                          final TopologyBuilder topologyBuilder = new TopologyBuilder();
33                          topologyBuilder.setSpout("randomsentencespout", new RandomSentenceSpout());
34                          topologyBuilder.setBolt("splitsentencebolt", new SplitSentenceBolt())
35                                          .shuffleGrouping("randomsentencespout");
36                          topologyBuilder.setBolt("wordcountbolt", new WordCountBolt())
37                                          .fieldsGrouping("splitsentencebolt", new Fields("word"));
38
39                          //Submit it to the cluster, or submit it locally
40                          if (null != args && 0 < args.length) {
41                                  config.setNumWorkers(3);
42                                  StormSubmitter.submitTopology(args[0], config, topologyBuilder.createTopology());
43                          } else {
44                                  config.setMaxTaskParallelism(10);
45                                  final LocalCluster localCluster = new LocalCluster();
46                                  localCluster.submitTopology(TOPOLOGY_NAME, config, topologyBuilder.createTopology());
47                                  //Run this topology for 120 seconds so that we can complete processing of decent # of tweets
48                                  Utils.sleep(120 * 1000);
49
50                                  LOGGER.info("Shutting down the cluster...");
51                                  localCluster.killTopology(TOPOLOGY_NAME);
52                                  localCluster.shutdown();
53                          }
54                  } catch (final AlreadyAliveException | InvalidTopologyException exception) {
55                          //Deliberate no op; not required actually.
56                          //exception.printStackTrace();
57                  } catch (final Exception exception) {
58                          //Deliberate no op; not required actually.
59                          //exception.printStackTrace();
60                  }
61                  LOGGER.info("\n\n\n\t\t*****Please clean your temp folder \"{}\" now!!!*****", System.getProperty("java.io.t
62          }
63  }
```



Code Sample – Spout

```java
package org.p7h.storm.offline.wordcount.spouts;

import java.util.Map;
import java.util.Random;

import backtype.storm.spout.SpoutOutputCollector;
import backtype.storm.task.TopologyContext;
import backtype.storm.topology.OutputFieldsDeclarer;
import backtype.storm.topology.base.BaseRichSpout;
import backtype.storm.tuple.Fields;
import backtype.storm.tuple.Values;
import backtype.storm.utils.Utils;

public final class RandomSentenceSpout extends BaseRichSpout {
        private static final long serialVersionUID = 25305908319060934L;
        private SpoutOutputCollector _collector;
        private Random _rand;

        @Override
        public final void open(final Map conf, final TopologyContext context, final SpoutOutputCollector collector) {
                _collector = collector;
                _rand = new Random();
        }

        @Override
        public final void nextTuple() {
                Utils.sleep(1000);
                final String sentence = SENTENCES[_rand.nextInt(SENTENCES.length)];
                _collector.emit(new Values(sentence));
        }

        @Override
        public final void ack(final Object id) {
        }

        @Override
        public final void fail(final Object id) {
        }

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


        private final static String[] SENTENCES = new String[]{
```

Code Sample – Bolt#1

```
17  public final class SplitSentenceBolt extends BaseBasicBolt {
18          private static final long serialVersionUID = 3077170245322026396L;
19
20          @Override
21          public final void execute(final Tuple input, final BasicOutputCollector collector) {
22                  //final String sentence = input.getString(0);
23                  final String sentence = (String) input.getValueByField("sentence");
24                  for (final String word : sentence.split(" ")) {
25                          collector.emit(new Values(word));
26                  }
27          }
28
29          @Override
30          public final void declareOutputFields(final OutputFieldsDeclarer outputFieldsDeclarer) {
31                  outputFieldsDeclarer.declare(new Fields("word"));
32          }
33  }
```

Code Sample – Bolt#2

```java
public final class WordCountBolt extends BaseBasicBolt {
        private static final Logger LOGGER = LoggerFactory.getLogger(WordCountBolt.class);
        private static final long serialVersionUID = -7958498892723043354L;
        final Map<String, Integer> wordCountTracketMap = new HashMap<>();
        private Stopwatch stopwatch = null;

        @Override
        public void prepare(final Map stormConf, final TopologyContext context) {
                this.stopwatch = new Stopwatch();
                this.stopwatch.start();
        }

        @Override
        public final void execute(final Tuple input, final BasicOutputCollector collector) {
                //final String word = input.getString(0);
                final String word = (String) input.getValueByField("word");
                Integer count = this.wordCountTracketMap.get(word);
                count = (count == null) ? 1 : count + 1;
                this.wordCountTracketMap.put(word, count);

                if (5 < this.stopwatch.elapsed(TimeUnit.SECONDS)) {
                        logWordCount();
                        this.stopwatch.reset();
                        this.stopwatch.start();
                }
        }

        private void logWordCount() {
                final StringBuilder wordCountLog = new StringBuilder();
                int i = 0;
                for (final String key : this.wordCountTracketMap.keySet()) {
                        if (3 < key.length()) {
                                i++;
                                if (0 != (i % 4)) {
                                        wordCountLog
                                                        .append(String.format("%15s", key))
                                                        .append(": ")
                                                        .append(String.format("%-3d", this.wordCountTracketMap.get(key)))
                                                        .append("\t");
                                } else {
                                        wordCountLog.append("\n");
                                }
                        }
                }
                LOGGER.info("\n\n{}\n{}", new Date(), wordCountLog.toString());
        }

        @Override
        public final void declareOutputFields(final OutputFieldsDeclarer outputFieldsDeclarer) {
                //no-op
        }
}
```


LET'S CODE NOW!

Problem#1 – WordCount [if there are internet issues]

https://github.com/P7h/StormWordCount

- Create a Spout which feeds random sentences [you can define your own set of random sentences].
- Create a Bolt which receives sentences from the Spout and then splits them into words and forwards them to next bolt.
- Create another Bolt to count the words.

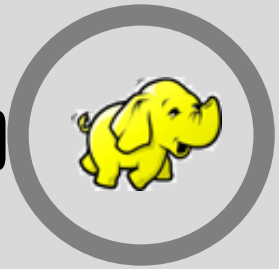
Problem#2 – Top5 retweeted tweets [if internet works fine]

https://github.com/P7h/StormTopRetweets

- Create a Spout which gets data from Twitter [please use Twitter4J and OAUTH Credentials to get tweets using Streaming API].
  - For simplicity consider only tweets which are in English.
  - Emit only the stuff which we are interested, i.e. A tweet's getRetweetedStatus().
- Create another Bolt to count the count the retweets of a particular tweet.
  - Make an in-memory Map with retweet screen name and the counter of the retweet as the value.
  - Log the counter every few seconds / minutes [should be configurable].

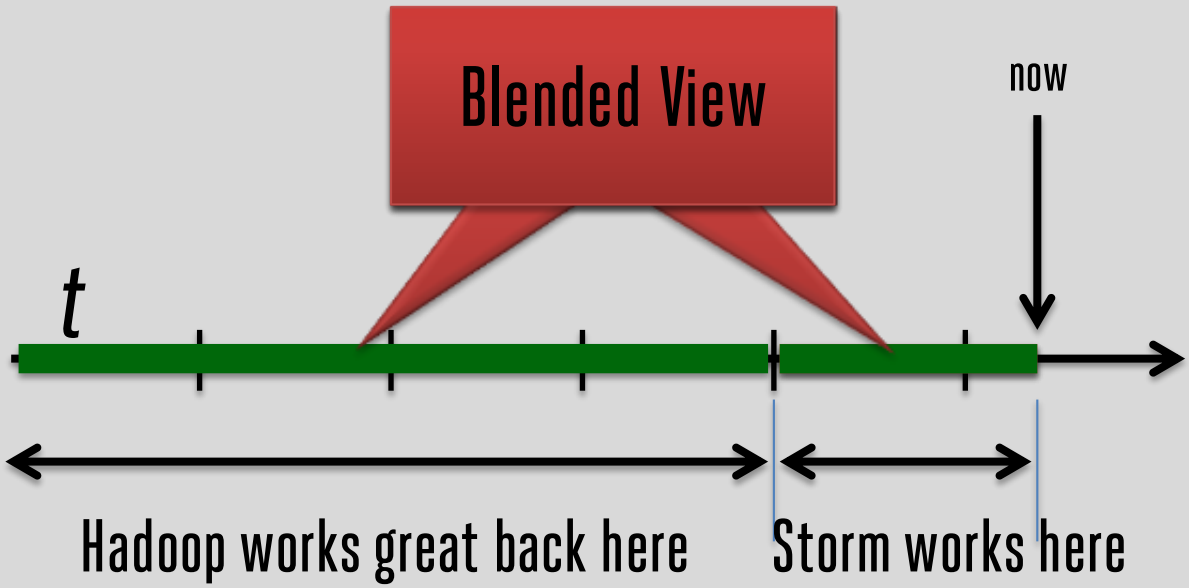
STORM VS. HADOOP

Storm vs. Hadoop

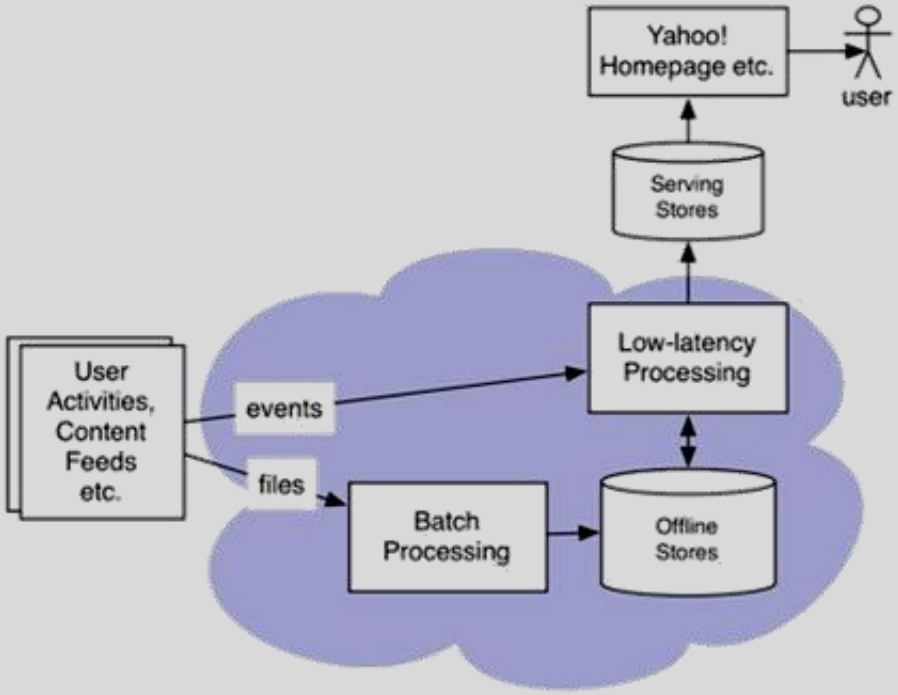


Storm	Hadoop
Real-time processing	Batch processing
Topologies run forever	Jobs run to completion
No SPOF	[Pre-YARN] NameNode is SPOF
Stateless nodes	Stateful nodes
Scalable	Scalable
Gurantees no dataloss	Guarantees no data loss
Open source	Open source

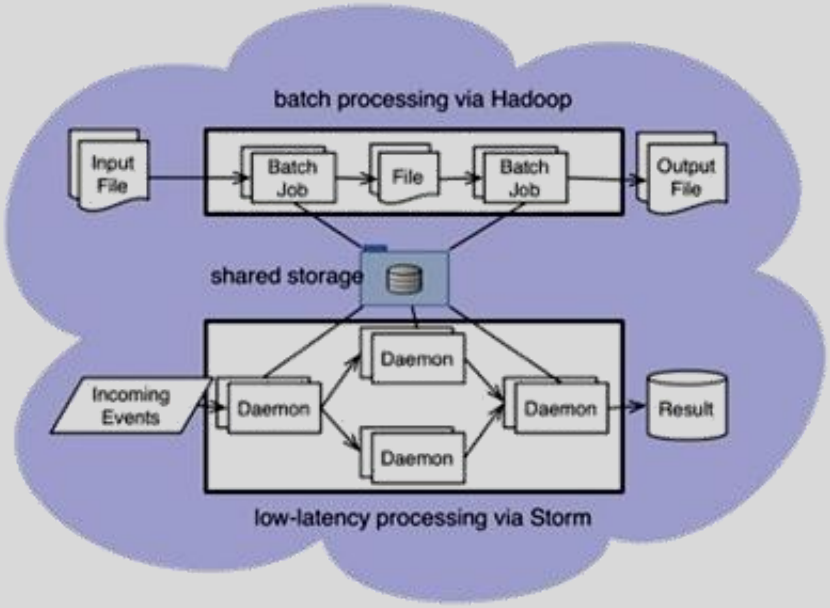
Hadoop AND Storm



Hadoop AND Storm at Yahoo



Personalization based on User Interests



Convergence of batch and low-latency processing

Advanced Topics [not covered in this session]

- ✗ Distributed RPC
- ✗ Transactional topologies
- ✗ Trident
- ✗ Unit testing
- ✗ Patterns

References

- This Slide deck [on slideshare] – http://j.mp/5thEleStorm_SS
- This Slide deck [on speakerdeck] – http://j.mp/5thEleStorm_SD
- My GitHub Account for code repos – https://github.com/P7h
- Bit.ly Bundle for Storm curated by me – http://j.mp/YrDgcs

Q & A


Prashanth Babu V V
Follow me on Twitter: @P7h
THANKS