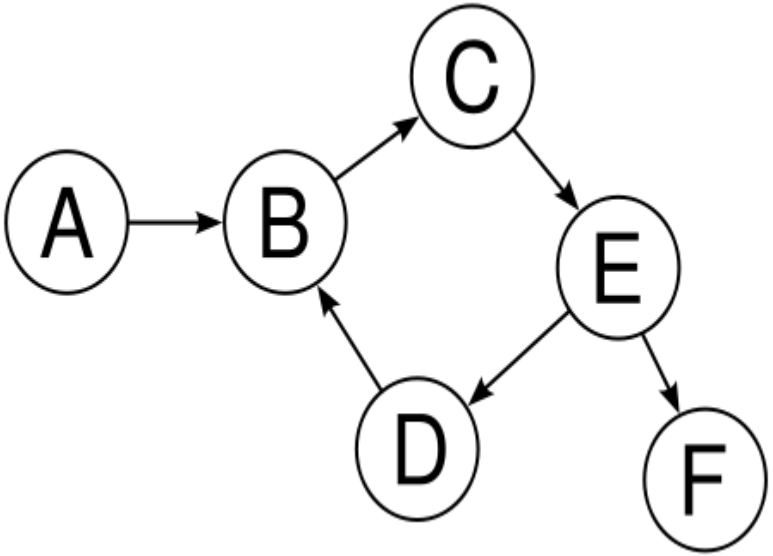
Storm@Twitter

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

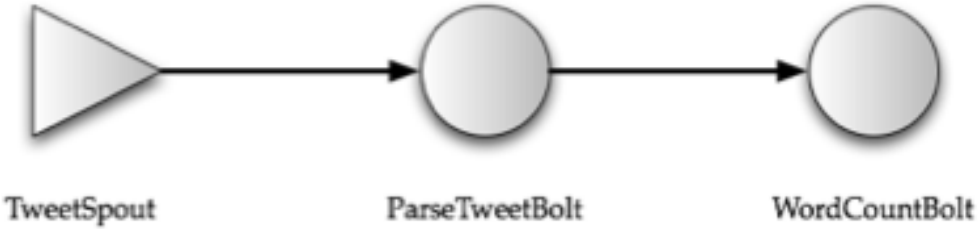
Paper Presented by Harsha Yeddanapudy

Basic Storm data processing architecture consists of *streams of tuples* flowing through *topologies*.

vertices - computation
edges - data flow



Spouts & Bolts



spouts produce tuples for the topology

bolts process incoming tuples and pass them downstream to the next bolts

Partioning Strategies

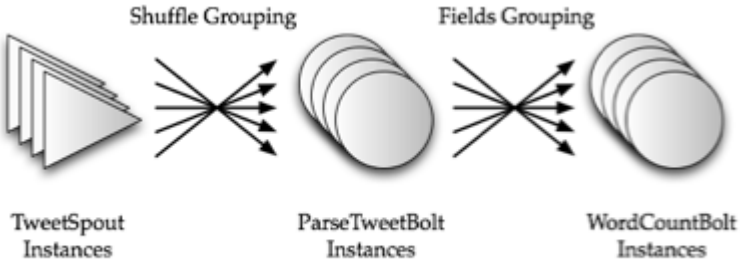
Shuffle grouping, which randomly partitions the tuples.

Fields grouping, which hashes on a subset of the tuple attributes/fields.

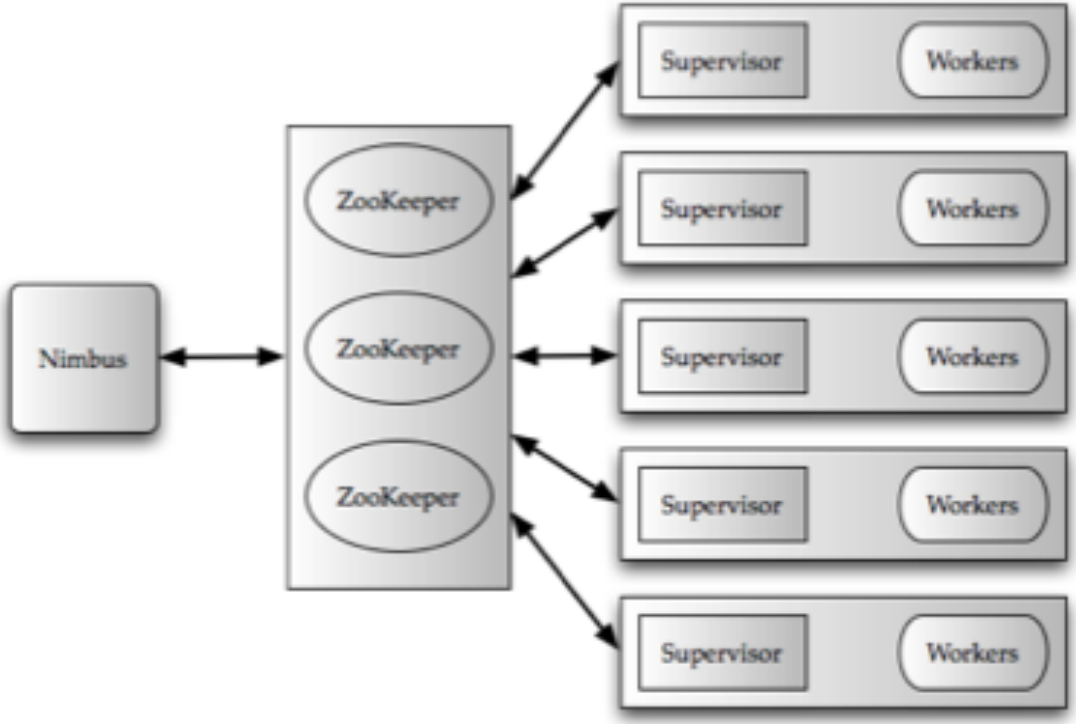
All grouping, which replicates the entire stream to all the consumer tasks.

Global grouping, which sends the entire stream to a single bolt.

Local grouping, which sends tuples to the consumer bolts in the same executor.



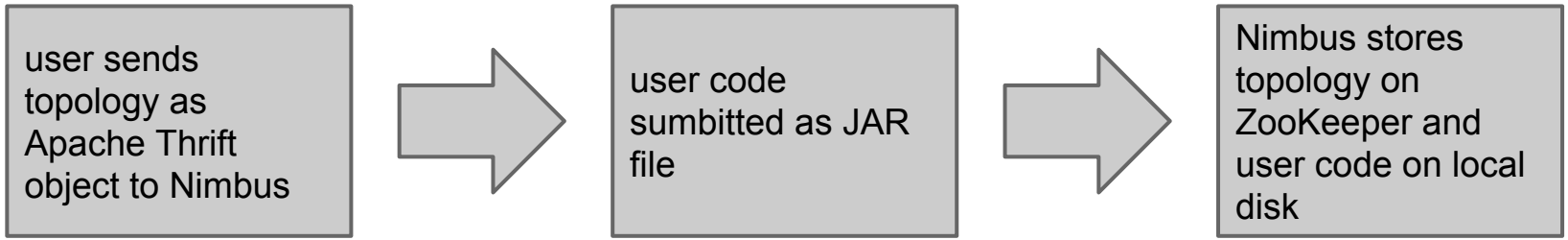
Storm Overview



Nimbus

responsible for distributing and coordinating the execution of the topology.

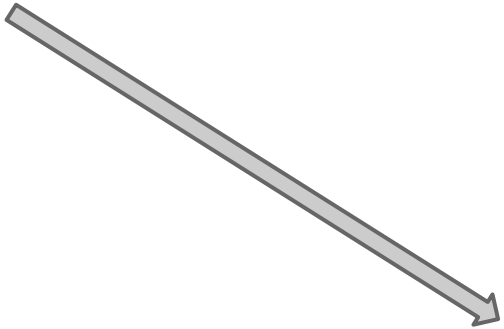
Nimbus cont.


user sends topology as Apache Thrift object to Nimbus
user code sumbitted as JAR file
Nimbus stores topology on ZooKeeper and user code on local disk

Nimbus w/ ZooKeeper & Supervisor

supervisors advertise running topologies and vacancies to Nimbus every 15 sec

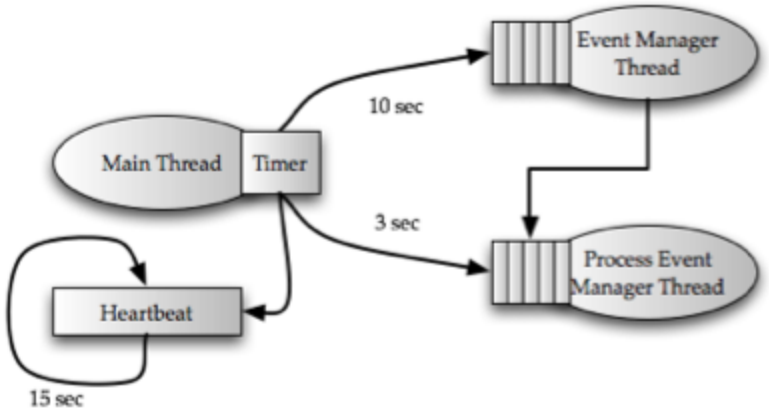
fail-fast and stateless

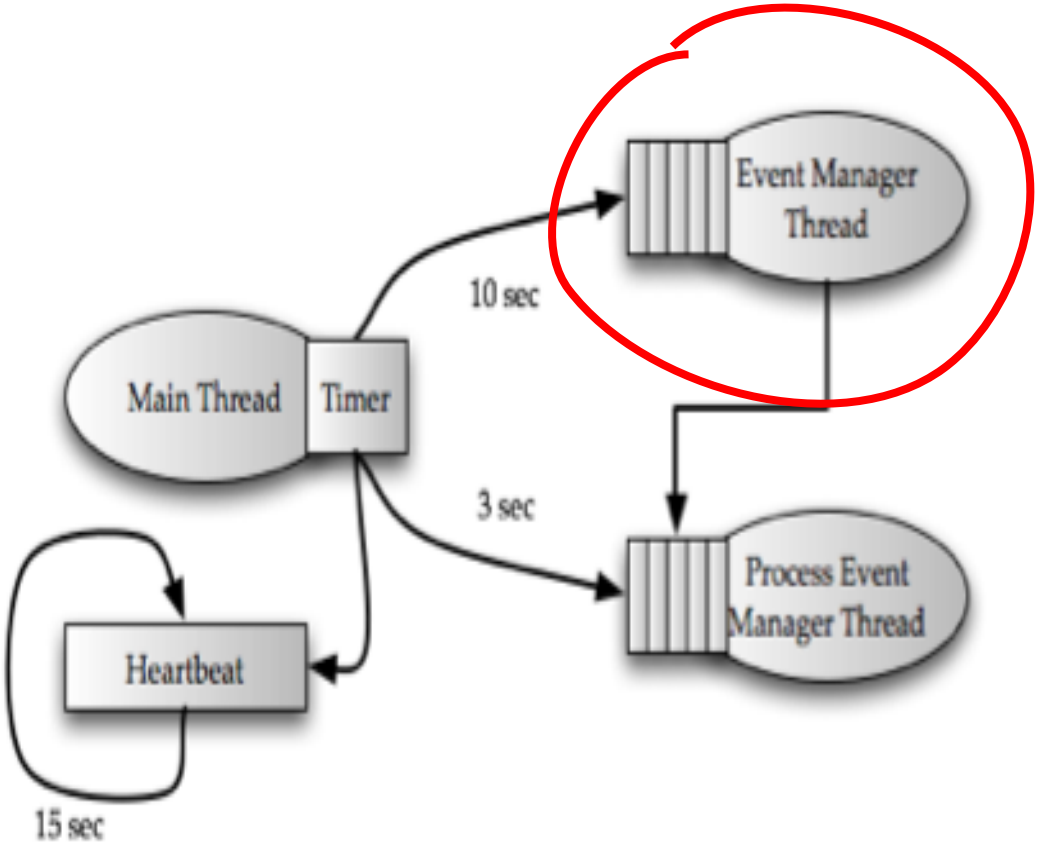


states

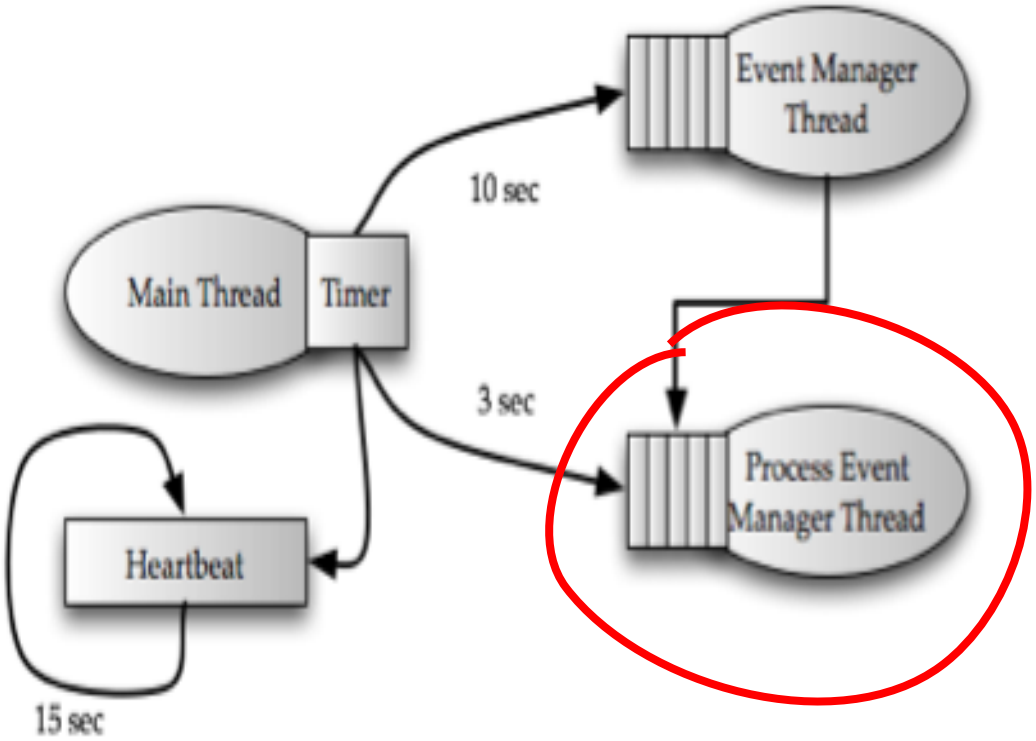
Supervisor

- runs on each storm node
- recieves assignments from nimbus and starts workers
- also monitors health of workers





- responsible for managing changes in existing assigments
- downloads JAR files and libraries for the addition of new topologies



- reads worker heartbeats and classifies them as either *valid, timed-out, not started* or *disallowed*

Workers and Executors

- executors are threads within the worker processes
- an executor can run several tasks
- a task is an instance of a spout of bolt
- tasks are strictly bound to their executors

Workers

worker receive thread: listens on TCP/IP port for incoming tuples and puts them in the appropriate in-queue

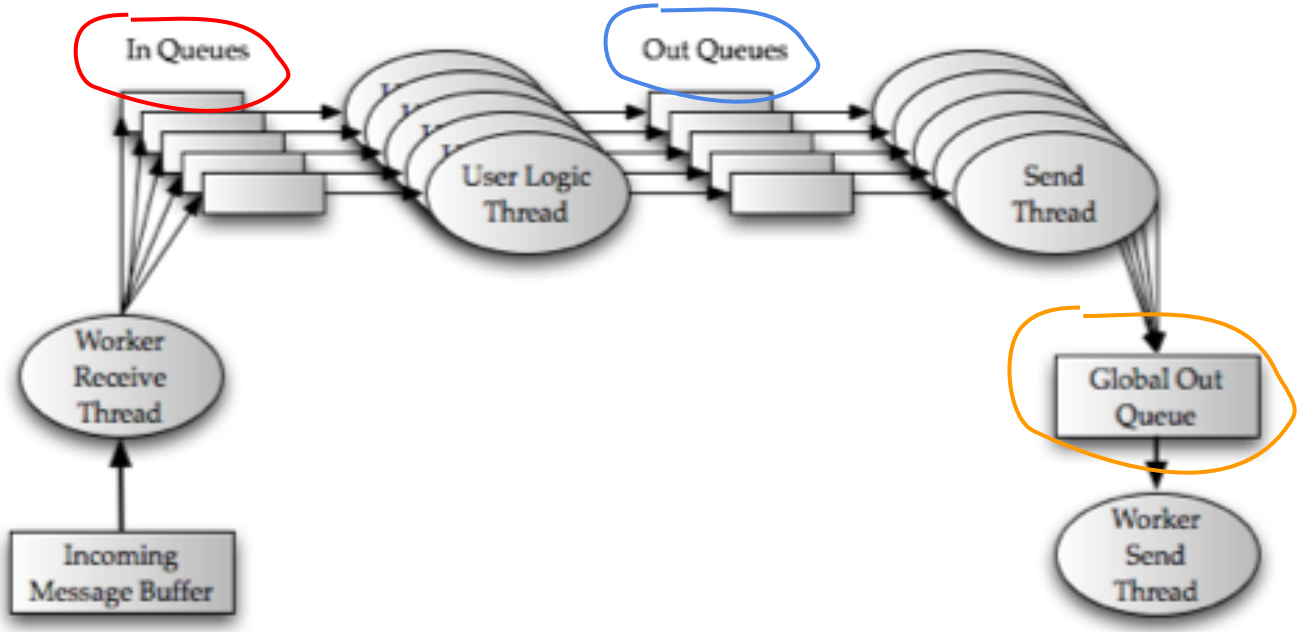
worker send thread: examines each tuple in global transfer queue, sends it to next worker downstream based on its task destination identifier

Executors

User Logic Thread: takes incoming tuples from in-queue, runs actual task, and places outgoing tuples in out-queue

Executor Send Thread: takes tuples from out-queue and puts them in global transfer queue

message flow inside worker



Processing Semantics

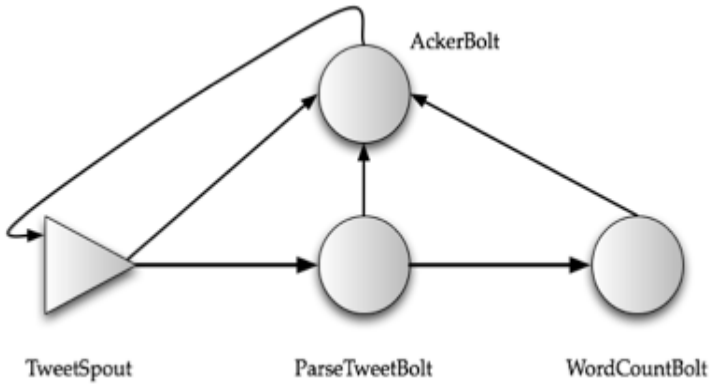
Storm provides two semantics gaurentees:

1. “at most once”
   - gaurentees that a tuple is successfully processed or failed in each stage of the topology
2. “at least once”
   - no gaurentee of tuple success or failure

At Least Once

Acker bolt is use to provide at least semantics:

- random generated 64 bit message id attached to each new tuple
- new tuples created by partioning during tasks are assigned a new message id
- backflow mechanism used to acknowledge tasks that contributed to output tuple
- retires tuple once it reaches spout that started tuple processing



XOR Implementation

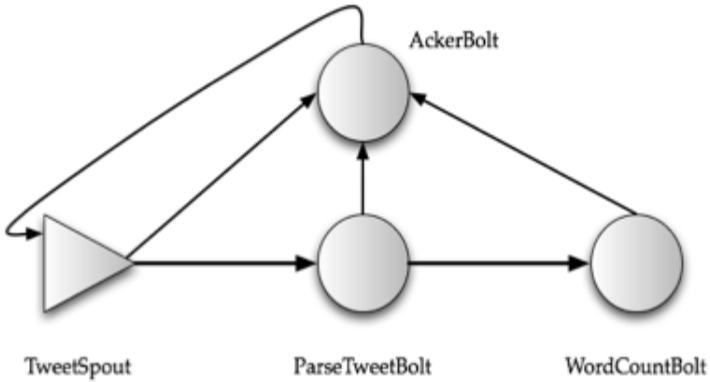
- message ids are XORed and sent to the acker along with original tuple message id and timeout parameter
- when tuple processing is complete XORed message id and original id sent to acker bolt
- acker bolt locates original tuple and get its XOR checksum, then XORed again with acked tuple id
- if XOR checksum is zero acker knows tuple has been fully processed.

Possible Outputs

Acked - XOR checksum successfully goes to zero, hold dropped, tuple retired

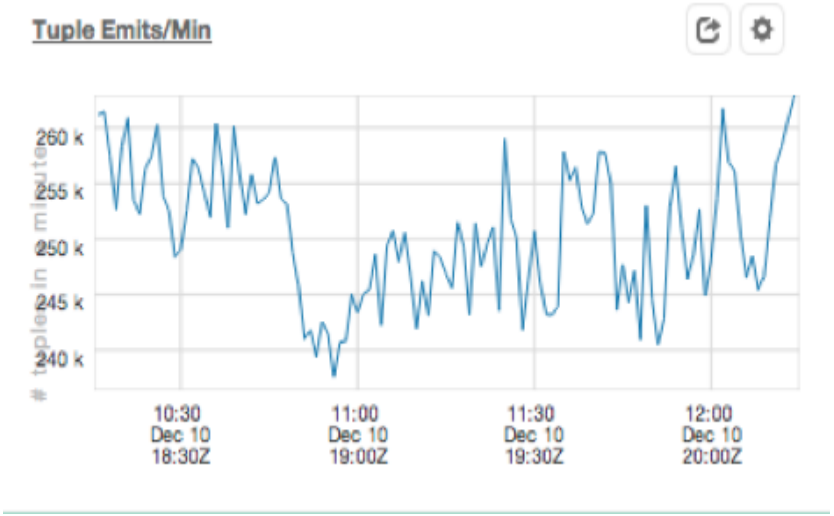
Failed - ?

Neither - Timeout parameter alerts us, restart from last spout checkpoint

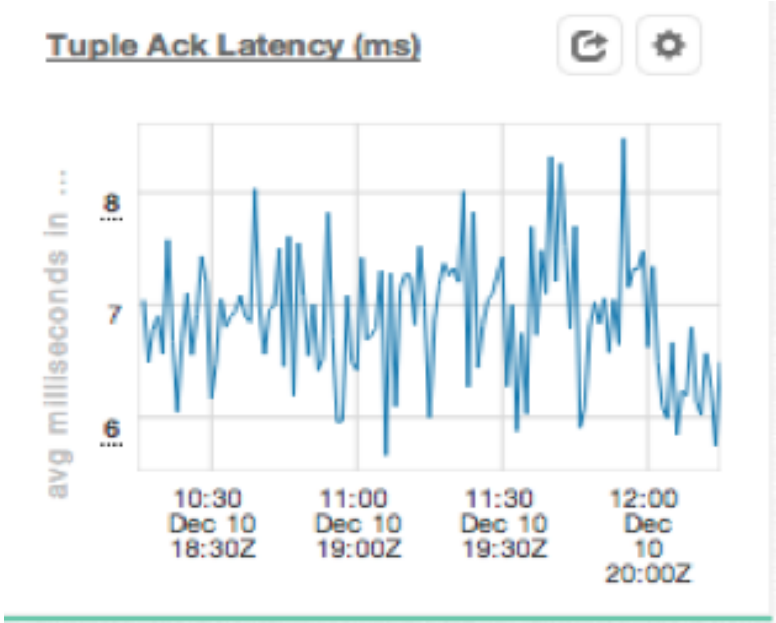


XOR Implementation cont.

Spout



Bolt



Experiment Setup

Component	# tasks
Spout	200
DistributorBolt	200
UserCountBold	300
AggregatorBolt	20

Time (relative to the start of the experiment)	# machines	# workers	Approximate #workers/machine
0 minutes	16	50	3
+15 minutes	13	50	4
+30 minutes	10	50	5
+45 minutes	7	50	7
+60 minutes	4	50	12

Results


tuples processed by topology/minute
6 m
4 m
2 m
0
14:30
Mar 14
21:30Z
14:45
Mar 14
21:45Z
15:00
Mar 14
22:00Z
15:15
Mar 14
22:15Z
15:30
Mar 14
22:30Z
15:45
Mar 14
22:45Z

Operational Stories

Overloaded Zookeeper - less writes to zookeeper, tradeoff read consistency for high availability & write performance

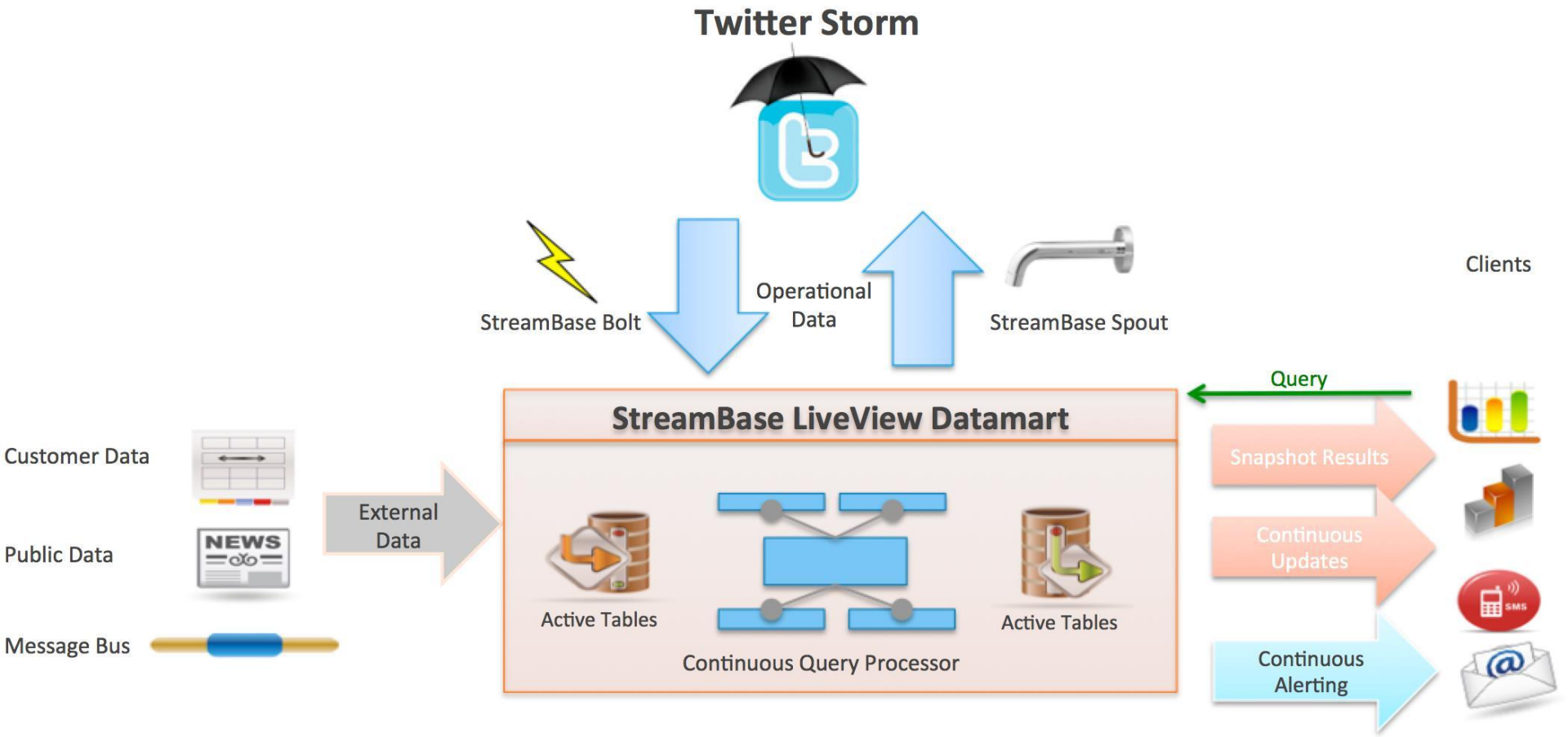
Storm Overheads - Storm does not have more overhead than equivalent Java; add extra machines for business logic and tuple serialization costs

Max Spout Tuning - Number of tuples in flight value is set dynamically by algorithm for greatest throughput

Review

Storm@Twitter is...

- Scalable
- Resilient
- Extensible
- Efficient


Twitter Storm
StreamBase Bolt
Operational Data
StreamBase Spout
Clients
Query
StreamBase LiveView Datamart
Customer Data
Public Data
Message Bus
External Data
Active Tables
Active Tables
Continuous Query Processor
Snapshot Results
Continuous Updates
Continuous Alerting
NEWS
SMS