Scalable Streaming Analytics

KARTHIK RAMASAMY

@KARTHIKZ
Overview

I. Storm Overview

II. Storm Internals

III. Operational Experiences

IV. Heron

V. END

TALK OUTLINE
WHAT IS ANALYTICS?
according to Wikipedia

DISCOVERY
Ability to identify patterns in data

COMMUNICATION
Provide insights in a meaningful way
TYPES OF ANALYTICS

varieties

CUBE ANALYTICS

PREDICTIVE ANALYTICS
DIMENSIONS OF ANALYTICS

variants

STREAMING

Ability to analyze the data immediately after it is produced

INTERACTIVE

Ability to provide results instantly when a query is posed

BATCH

Ability to provide insights after several hours/days when a query is posed
STREAMING VS INTERACTIVE

INTERACTIVE ANALYTICS

Queries

Bulkload Data

Database Server

Static Batch Results/Reports

Data Storage

STREAMING ANALYTICS

Real time alerts,
Real time analytics
Continuous visibility

Results

Data Stream Processing

Queries
WHAT IS **REAL TIME**?

**msecs or secs or mins?**

- **< 500 MS**
  - latency sensitive
  - deterministic workflows
  - fanout Tweets search for Tweets

- **> 1 SEC**
  - approximate
  - ad impressions count hash tag trends

- **> 1 HOUR**
  - high throughput
  - adhoc queries
  - monthly active users relevance for ads

**OLTP**

**REAL TIME**

**BATCH**

**Feedback**

**Complement**
STREAMING DATA FLOW

varieties
STREAMING SYSTEMS
first generation – SQL based

NIAGARA Query Engine

Stanford Stream Data Manager

Aurora Stream Processing Engine

Borealis Distributed Stream Processing Engine

Cayuga – Stateful Event Monitoring
STREAMING SYSTEMS
next generation – too many

S4 distributed stream computing platform
STORM
samza
Spark
PULSAR
STORM OVERVIEW
WHAT IS STORM?

Streaming platform for analyzing realtime data as they arrive, so you can react to data as it happens.

- Guaranteed message processing
- Horizontal scalability
- Robust fault tolerance
- Concise code - focus on logic
STORM DATA MODEL

TOPOLOGY
Directed acyclic graph
Vertices = computation, and edges = streams of data tuples

SPOUTS
Sources of data tuples for the topology
Examples – Kafka/Kestrel/MySQL/Postgres

BOLTS
Process incoming tuples and emit outgoing tuples
Examples – filtering/aggregation/join/arbitrary function
STORM TOPOLOGY

SPOUT 1 → BOLT 1 → BOLT 2 → BOLT 3 → BOLT 4 → BOLT 5

SPOUT 2 → BOLT 1 → BOLT 2 → BOLT 3 → BOLT 4 → BOLT 5
Live stream of Tweets

TWEET SPOUT → PARSE TWEET BOLT → WORD COUNT BOLT

LOGICAL PLAN
When a parse tweet bolt task emits a tuple which word count bolt task should it send to?
STREAM GROUPINGS

- **SHUFFLE GROUPING**: Random distribution of tuples
- **FIELDS GROUPING**: Group tuples by a field or multiple fields
- **ALL GROUPING**: Replicates tuples to all tasks
- **GLOBAL GROUPING**: Sends the entire stream to one task
WORD COUNT TOPOLOGY

TWEET SPOUT TASKS

SHUFFLE GROUPING

PARSE TWEET BOLT TASKS

FIELDS GROUPING

WORD COUNT BOLT TASKS
STORM INTERNALS
STORM ARCHITECTURE

Nimbus

ZK CLUSTER

SUPERVISOR
W1 W2 W3 W4

MASTER NODE

ASSIGNMENT MAPS

SYNC CODE

TOPOLOGY SUBMISSION

SLAVE NODE

SUPERVISOR
W1 W2 W3 W4
DATA FLOW IN STORM WORKERS

In Queue → User Logic Thread → Out Queue → Send Thread

Global Receive Thread → TCP Receive Buffer

Disruptor Queues

Outgoing Message Buffer → Global Send Thread → TCP Send Buffer

0mq Queues

Kernel
Large amount of data produced every day

Largest storm cluster

Several topologies deployed

Several billion messages every day

1 stage

8 stages
STORM ARCHITECTURE

Nimbus

Master Node

Slave Node

Supervisor

W1 W2 W3 W4

ZK Cluster

Assignment Maps

Storage Contention

Topology Submission

Multiple Functionality
Scheduling/Monitoring

Single point of failure
STORM WORKER

JVM PROCESS

EXECUTOR1

TASK1

TASK2

TASK3

EXECUTOR2

TASK4

TASK5

Complex hierarchy

Hard to debug

Difficult to tune

Complex hierarchy

Hard to debug

Difficult to tune
DATA FLOW IN STORM WORKERS

- In Queue
- User Logic Thread
- Out Queue
- Send Thread
- Global Receive Thread
- TCP Receive Buffer
- Outgoing Message Buffer
- Global Send Thread
- TCP Send Buffer
- Kernel
- Queue Contention
- Multiple Languages
OVERLOADED ZOOKEEPER

Scaled up

STORM

W W W

zk zk zk

S1 S2 S3

Handled unto to 1200 workers per cluster
OVERLOADED ZOOKEEPER

Analyzing zookeeper traffic

KAFKA SPOUT
Offset/partition is written every 2 secs

33%

STORM RUNTIME
Workers write heart beats every 3 secs

67%
OVERLOADED ZOOKEEPER

Heart beat daemons

STORM

5000 workers per cluster
EVOLUTION OR REVOLUTION?

fix storm or develop a new system?

**FUNDAMENTAL ISSUES—REQUIRE EXTENSIVE REWRITING**

- Several queues for moving data
- Inflexible and requires longer development cycle

**USE EXISTING OPEN SOURCE SOLUTIONS**

- Issues working at scale/lacks required performance
- Incompatible API and long migration process
HERON DESIGN GOALS

FULLY API COMPATIBLE WITH STORM

Directed acyclic graph
Topologies, spouts and bolts

USE OF WELL KNOWN LANGUAGES

No Clojure
C++/JAVA/Python
TOPOLOGY ARCHITECTURE

Topology Master

Sync Physical Plan

ZK CLUSTER

Logical Plan, Physical Plan and Execution State

Stream Manager
Metrics Manager

CONTAINER
CONTAINER

I1 I2 I3 I4

I1 I2 I3 I4

I1 I2 I3 I4

I1 I2 I3 I4
TOPOLOGY MASTER

Solely responsible for the entire topology
TOPOLOGY MASTER

PREVENT MULTIPLE TM BECOMING MASTERS

ALLOWS OTHER PROCESS TO DISCOVER TM

Topology Master

Logical Plan, Physical Plan and Execution State

ZK CLUSTER
STREAM MANAGER

Routing Engine

ROUTES TUPLES
BACKPRESSURE
ACK MGMT
STREAM MANAGER

O(n^2)  

O(k^2)
STREAM MANAGER
tcp back pressure

SLOWS UPSTREAM AND DOWNSTREAM INSTANCES
STREAM MANAGER
spout back pressure
STREAM MANAGER
back pressure advantages

PREDICTABILITY
Tuple failures are more deterministic

SELF ADJUSTS
Topology goes as fast as the slowest component
HERON INSTANCE

Does the real work!

RUNS ONE TASK

EXPOSES API

COLLECTS METRICS
HERON INSTANCE

Stream Manager

Gateway Thread

Task Execution Thread

Metrics Manager

data-in queue

data-out queue

metrics-out queue

BOUNDLED QUEUES - TRIGGERS GC IN LARGE TOPOLOGIES
Optical Nerve

GATHERS METRICS

SCRIBES

ABSTRACTED
HERON PERFORMANCE

Throughput with acknowledgements – Word count topology

- Storm
- Heron

Spout Parallelism

million tuples/min

0

100

200

500
HERON PERFORMANCE
Latency with acknowledgements enabled – Word Count Topology

- Storm
- Heron

Spout Parallelism:
- 25
- 100
- 200
- 500

Latency (ms):
- 0
- 625
- 1250
- 1875
- 2500

Graph shows the latency (in milliseconds) for different spout parallelism values when acknowledging messages. The Storm latency increases significantly with higher parallelism, while Heron's latency remains relatively constant.
HERON PERFORMANCE
CPU usage with acknowledgements enabled – Word Count Topology

- Storm
- Heron

# cores used

Spout Parallelism

0 25 100 200 500

0 625 1250 1875 2500
HERON PERFORMANCE
Throughput with no acknowledgements – Word count topology
HERON PERFORMANCE

CPU usage with no acknowledgements – Word Count Topology

- Storm
- Heron

# cores used
2500
1875
1250
625
0

Spout Parallelism
0
25
100
200
500

Graph showing CPU usage with no acknowledgements for Storm and Heron at different spout parallelism levels.
HERON PERFORMANCE

CPU usage – RTAC Topology

<table>
<thead>
<tr>
<th>Storm</th>
<th>Heron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Storm</th>
<th>Heron</th>
</tr>
</thead>
<tbody>
<tr>
<td>No acknowledgements</td>
<td></td>
</tr>
</tbody>
</table>
HERON PERFORMANCE
Latency with acknowledgements enabled – RTAC Topology

- Storm
- Heron

Latency (ms)
OPERATIONAL EXPERIENCES
OPERATIONAL EXPERIENCE

SERVICE-LESS

All topologies run under topology owner’s role

CLUSTER-LESS

Everything runs on Aurora

TENSION-LESS

No more 2am pages
DEVELOPER EXPERIENCE

- **DEBUG**
  - Faster iteration

- **TUNE**
  - Better resource utilization

- **DEPLOY**
  - Devel to prod in 5min
MIGRATION EXPERIENCE

SMALL
Couple of hours

MEDIUM
Lots of savings

LARGE
Summingbird tuning takes time
CURRENT WORK
CURRENT WORK

SERIALIZATION
- Use Java Reflection

TUNING
- Determine optimal set of parameters

ELASTIC
- Grow/Shrink based on data

CONFIGURATION
- Update topology without restarting
QUESTIONS AND ANSWERS

Go ahead. Ask away.