Flying Faster with Heron

Karthik Ramasamy
@KARTHIKZ

#TwitterHeron
TALK OUTLINE

BEGIN

I
OVERVIEW

II
MOTIVATION

III
HERON

IV
OPERATIONAL EXPERIENCES

V
HERON PERFORMANCE

END
OVERVIEW
TWITTER IS REAL TIME

REAL TIME TRENDS
Emerging break out trends in Twitter (in the form #hashtags)

REAL TIME CONVERSATIONS
Real time sports conversations related with a topic (recent goal or touchdown)

REAL TIME RECOMMENDATIONS
Real time product recommendations based on your behavior & profile

REAL TIME SEARCH
Real time search of tweets

ANALYZING BILLIONS OF EVENTS IN REAL TIME IS A CHALLENGE!
Streaming platform for analyzing real-time data as they arrive, so you can react to data as it happens.
STORM TERMINOLOGY

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

SPOUTS
Sources of data tuples for the topology
Examples – Event Bus/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 GROUPING

- **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
MOTIVATION
**STORM ARCHITECTURE**

**Nimbus**
- Multiple Functionality
- Scheduling/Monitoring
- Single point of failure
- No resource reservation and isolation

**ZK CLUSTER**
- Storage Contention

**TOPOLOGY SUBMISSION**

**ASSIGNMENT MAPS**

**SUPERVISOR**
- W1
- W2
- W3
- W4

**SLAVE NODE**
STORM WORKER

- TASK1
- TASK2
- TASK3
- TASK4
- TASK5

Complex hierarchy
Hard to debug
Difficult to tune

JVM PROCESS

EXECUTOR1

EXECUTOR2
DATA FLOW IN STORM WORKERS

In Queue

TCP Receive Buffer

Global Receive Thread

User Logic Thread

Queue Contention

Multiple Languages

Outgoing Message Buffer

Global Send Thread

TCP Send Buffer

Kernel
OVERLOADED ZOOKEEPER

Scaled up

STORM

W

W

W

zk

zk

S1

S2

S3

Handled unto to 1200 workers per cluster
OVERLOADED ZOOKEEPER

Analyzing zookeeper traffic

KAFKA SPOUT

67%
Offset/partition is written every 2 secs

STORM RUNTIME

33%
Workers write heart beats every 3 secs
OVERLOADED ZOOKEEPER

Heartbeat daemons

STORM

W W W

I P H

zk zk zk

S1 S2 S3

KV KV KV

5000 workers per cluster
STORM - DEPLOYMENT

shared pool

storm cluster
STORM - DEPLOYMENT

- **Shared pool**
- **Isolated pools**
- **Joe’s topology**

**Storm cluster**
STORM - DEPLOYMENT

storm cluster

shared pool

isolated pools
- joe’s topology
- jane’s topology
STORM - DEPLOYMENT

- Storm cluster
- Shared pool
- Isolated pools:
  - Joe’s topology
  - Jane’s topology
  - Dave’s topology
STORM ISSUES

LACK OF BACK PRESSURE
Drops tuples unpredictably

EFFICIENCY
Serialization program consumes 75 cores at 30% CPU
Topology consumes 600 cores at 20–30% CPU

NO BATCHING
Tuple oriented system – implicit batching by 0MQ
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

TASK ISOLATION
Ease of debug ability/resource isolation/profiling

USE OF MAIN STREAM LANGUAGES
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
TOPOLOGY MASTER

Solely responsible for the entire topology

ASSIGNS ROLE

MONITORING

METRICS
TOPOLOGY MASTER

PREVENT MULTIPLE TM BECOMING MASTERS

ALLOWS OTHER PROCESS TO DISCOVER TM

Logical Plan, Physical Plan and Execution State
STREAM MANAGER

Routing Engine

- ROUTES TUPLES
- BACK PRESSURE
- 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
stage by stage 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
OPERATIONAL EXPERIENCES
HERON DEPLOYMENT

Aurora Services

Heron Tracker

Heron Web

Heron VIZ

ZK CLUSTER

Topology N

Topology 3

Topology 2

Topology 1

Observability
HERON Sample Topologies
SAMPLE TOPOLOGY DASHBOARD
Large amount of data produced every day
Large cluster
Several topologies deployed
Several billion messages every day

1 stage
10 stages

3x reduction in cores and memory

STORM is decommissioned
HERON PERFORMANCE
## Heron Performance

### Settings

<table>
<thead>
<tr>
<th>COMPONENTS</th>
<th>EXPT #1</th>
<th>EXPT #2</th>
<th>EXPT #3</th>
<th>EXPT #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spout</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Bolt</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td># Heron containers</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td># Storm workers</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
</tbody>
</table>
HERON PERFORMANCE
Word count topology – Acknowledgements enabled

**Throughput**

<table>
<thead>
<tr>
<th>Spout Parallelism</th>
<th>Storm</th>
<th>Heron</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Latency**

<table>
<thead>
<tr>
<th>Spout Parallelism</th>
<th>Storm</th>
<th>Heron</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10–14x

5–15x
HERON PERFORMANCE

Word count topology – CPU usage

# cores used

Storm
Heron

Spout Parallelism

2–3x
**HERON PERFORMANCE**

Throughput and CPU usage with no acknowledgements - Word count topology

![Throughput and CPU usage comparison](chart)
HERON EXPERIMENT

RTAC topology
HERON PERFORMANCE
CPU usage – RTAC Topology

# cores used

Storm
Acknowledgements enabled

Heron

Storm
No acknowledgements

# cores used
HERON PERFORMANCE
Latency with acknowledgements enabled - RTAC Topology

- Storm
- Heron
Many modern data processing environments require fault-tolerant and distributed stream data processing systems. Storm is a real-time data processing engine at Twitter. We also present results from an empirical evaluation demonstrating the resilience of Storm at Twitter. We note that Storm presents the design and engineering choices that enable scalability, debug-ability, manageability, and efficient sharing of the underlying business model.

After examining various options, we present the design and engineering choices that enable scalability, debug-ability, manageability, and efficient sharing of Storm topologies in Twitter. We discuss the large scale deployment of Storm at Twitter, like many other organizations, relies heavily on real-time processing at scale. We compare Storm with other stream processing systems, such as Apache Heron and Spark Streaming, and discuss the advantages of Storm.

We note that Storm is a processing engine at Twitter. We also present results from an empirical evaluation demonstrating the resilience of Storm at Twitter. We compare Storm with other stream processing systems, such as Apache Heron and Spark Streaming, and discuss the advantages of Storm.
CONCLUSION

SIMPLIFIED ARCHITECTURE
Easy to debug, profile and support

HIGH PERFORMANCE
7–10x increase in throughput
5–10x improvement in latency

EFFICIENCY
3–5x decrease in resource usage
#ThankYou
FOR LISTENING
QUESTIONS AND ANSWERS

Go ahead. Ask away.