Streaming In Practice

KARTHIK RAMASAMY
@KARTHIKZ

#TwitterHeron
TALK OUTLINE

I. HERON OVERVIEW
II. HERON PERFORMANCE
III. HERON BACKPRESSURE
IV. HERON LOAD SHEDDING
V. CONCLUSION

BEGIN
END
HERON
OVERVIEW
STORM/HERON TERMINOLOGY

**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/HERON TOPOLOGY

SPOUT 1 → BOLT 1 → BOLT 2 → BOLT 4

SPOUT 2 → BOLT 1 → BOLT 2 → BOLT 3 → BOLT 5
WHY HERON?

- PERFORMANCE PREDICTABILITY
- IMPROVE DEVELOPER PRODUCTIVITY
- EASE OF MANAGEABILITY
HERON DESIGN DECISIONS

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

CONTAINER

Stream Manager

Metrics Manager

CONTAINER

Stream Manager

Metrics Manager

I1  I2  I3  I4

I1  I2  I3  I4

Logical Plan, Physical Plan and Execution State
HERON SAMPLE TOPOLOGIES
Heron has been in production for 2 years

Large amount of data produced every day
Large cluster
Several hundred topologies deployed
Several billion messages every day

1 stage
10 stages

3x reduction in cores and memory
HERON USE CASES

REALTIME ETL
REAL TIME BI
SPAM DETECTION
REAL TIME TRENDS
REALTIME ML
REAL TIME MEDIA
REAL TIME OPS
HERON ENVIRONMENT

- Laptop/Server
- Cluster/Aurora
- Cluster/Mesos
HERON RESOURCE USAGE
## 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

Latency

Storm
Heron

10-14x

5-15x
HERON RESOURCE USAGE

Event Spout

60-100M/min

Filter
8-12M/min

Flat-Map
40-60M/min

Aggregate Bolt

Output
25-42M/min

Aggregate
Cache 1 sec

Redis
## RESOURCE CONSUMPTION

<table>
<thead>
<tr>
<th></th>
<th>Cores Requested</th>
<th>Cores Used</th>
<th>Memory Requested (GB)</th>
<th>Memory Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>24</td>
<td>2–4</td>
<td>48</td>
<td>N/A</td>
</tr>
<tr>
<td>Heron</td>
<td>120</td>
<td>30–50</td>
<td>200</td>
<td>180</td>
</tr>
</tbody>
</table>
RESOURCE CONSUMPTION

- Spout Instances: 84%
- Bolt Instances: 9%
- Heron Overhead: 7%
RESOURCE CONSUMPTION - BREAKDOWN

- Fetching Data: 61%
- User Logic: 21%
- Heron Usage: 11%
- Writing Data: 8%
HERON BACK PRESSURE
Stragglers are the norm in a multi-tenant distributed systems
Bad machine, inadequate provisioning and hot keys

BACK PRESSURE AND STRAGGLERS

- Provides predictability
- Processes data at maximum rate
- Reduce recovery times
- Handles temporary spikes
BACK PRESSURE AND STRAGGLERS

MOST SCENARIOS BACK PRESSURE RECOVERS
Without any manual intervention

SUSTAINED BACK PRESSURE
Irrecoverable GC cycles
Bad or faulty host

SOMETIMES USER PREFER DROPPING OF DATA
Care about only latest data
LOAD SHEDDING

SAMPLING BASED APPROACHES
Down sample the incoming stream and scale up the results
Easy to reason if the sampling is uniform
Hard to achieve uniformity across distributed spouts

DROP BASED APPROACHES
Simply drop older data
Spouts takes a lag threshold and a lag adjustment value
Works well in practice
Streaming@Twitter

Maosong Fu, Sailesh Mittal, Vikas Kedigehalli, Karthik Ramasamy, Michael Barry, Andrew Jorgensen, Christopher Kellogg, Neng Lu, Bill Graham, Jingwei Wu

Twitter, Inc.

Twitter Heron: Stream Processing at Scale

Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel¹, Karthik Ramasamy, Siddarth Taneja

@sanjeevrk, @challenger_nik, @Louis_Fumaosong, @vikkyrk, @cckellogg, @saileshmittal, @peteljm, @karthikz, @staneja

Twitter, Inc., *University of Wisconsin – Madison

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

@ankitoshniwal, @staneja, @amits, @karthikz, @peteljm, @sanjeevrk, @jason_j, @krishnagade, @Louis_Fumaosong, @jakedonham, @challenger_nik, @saileshmittal, @squarecog

Twitter, Inc., *University of Wisconsin – Madison
#ThankYou
FOR LISTENING
QUESTIONS AND ANSWERS

Go ahead. Ask away.
HERON LOAD SHEDDING