ArchSummit 全球架构师峰会 2017

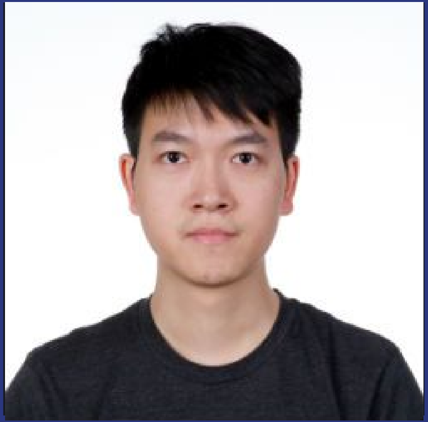
主办方 Geekbang 极客邦科技 InfoQ

Self Regulating Stream Processing in Heron

Huijun Wu

2017.12

SPEAKER INTRODUCE



Huijun Wu

Twitter, Inc.

Infrastructure, Data Platform, Real-Time Compute

TABLE OF CONTENTS 大纲

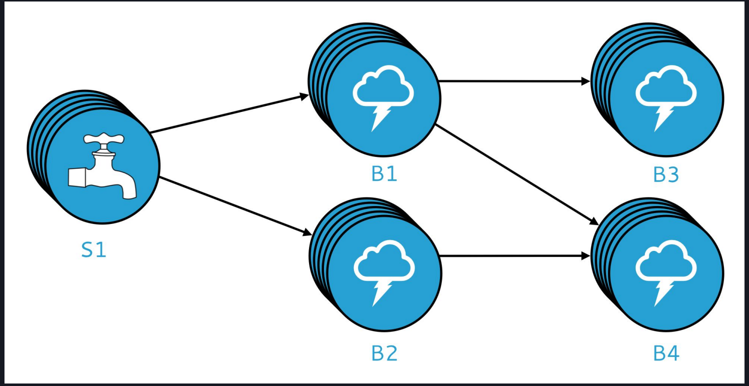
Heron Overview

Data model: user/developer perspective

What is Heron?
A real-time, distributed, fault-tolerant stream processing engine from Twitter

Topology(DAG)

- Vertex
  - Spout
  - Bolt
- Edge: Stream
  - Tuple



Compatible with Apache Storm data model

Heron Overview

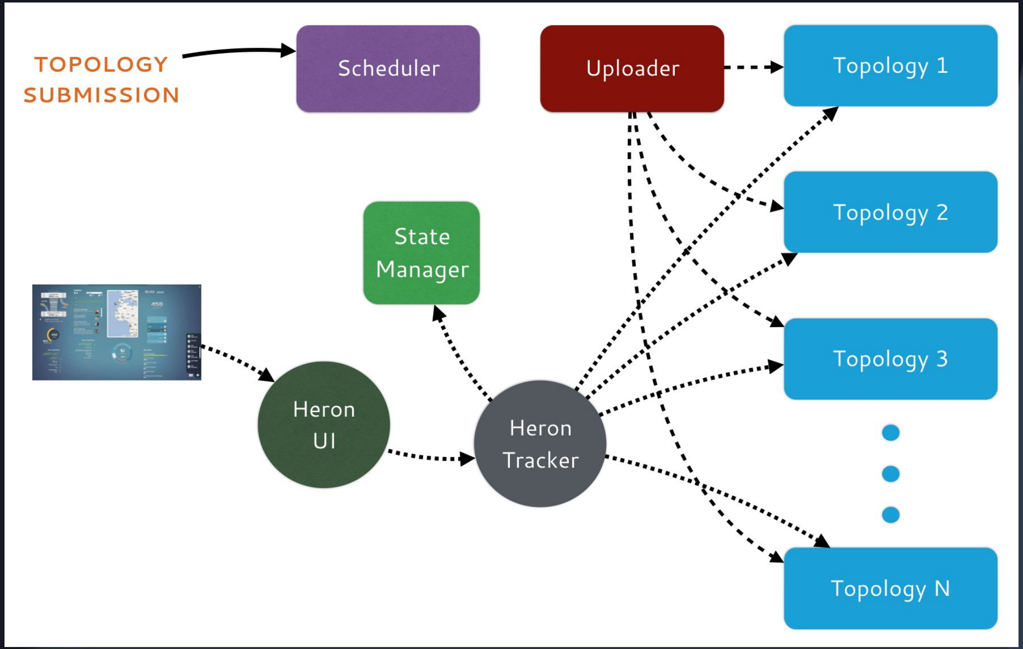
Runtime architecture: data center with multiple topologies

Topology shared services:

- Scheduler
- Uploader
- State manager: zookeeper

Tools:

- Tracker
- UI



Heron Overview

Runtime architecture: one particular topology

Shared between containers:

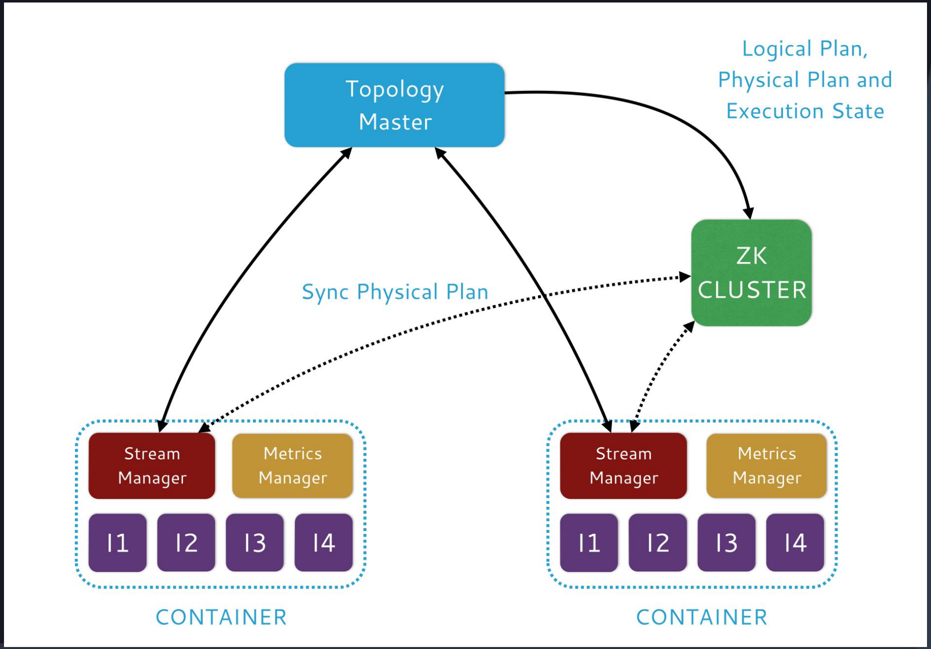
- State manager: zookeeper

Type 1: container 0

- Topology master
- Metrics cache

Type 2: container x (x>0)

- Stream manager
- Heron instance
- Metrics manager

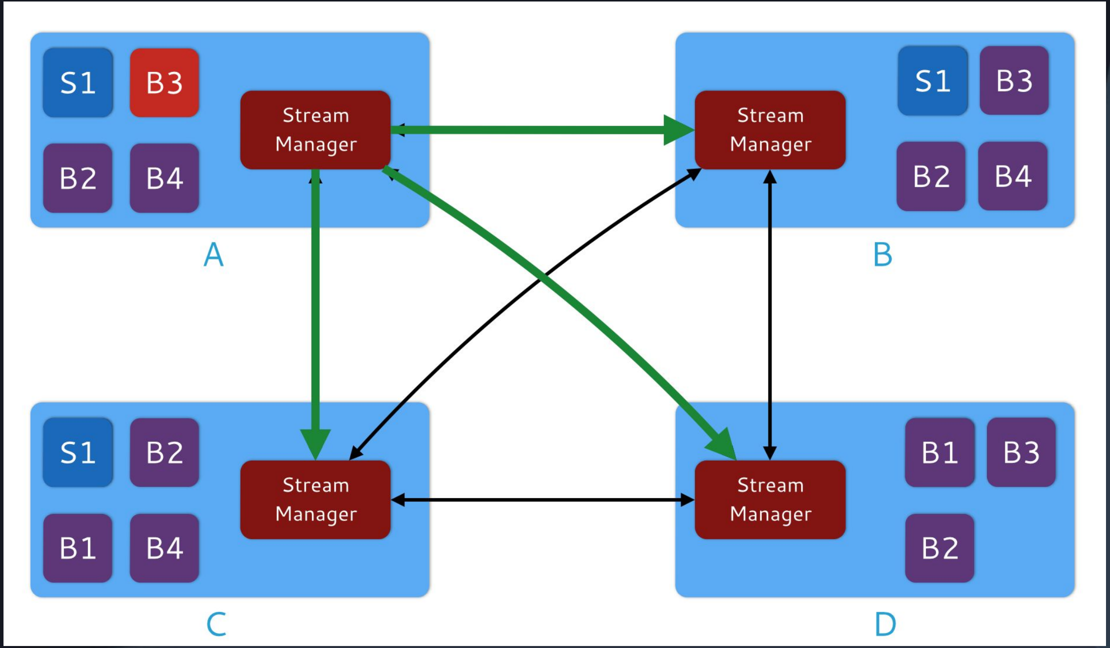


Heron Overview

Runtime rate control: backpressure

Health metrics:

- Metrics/counters
  - Backpressure
- Exceptions

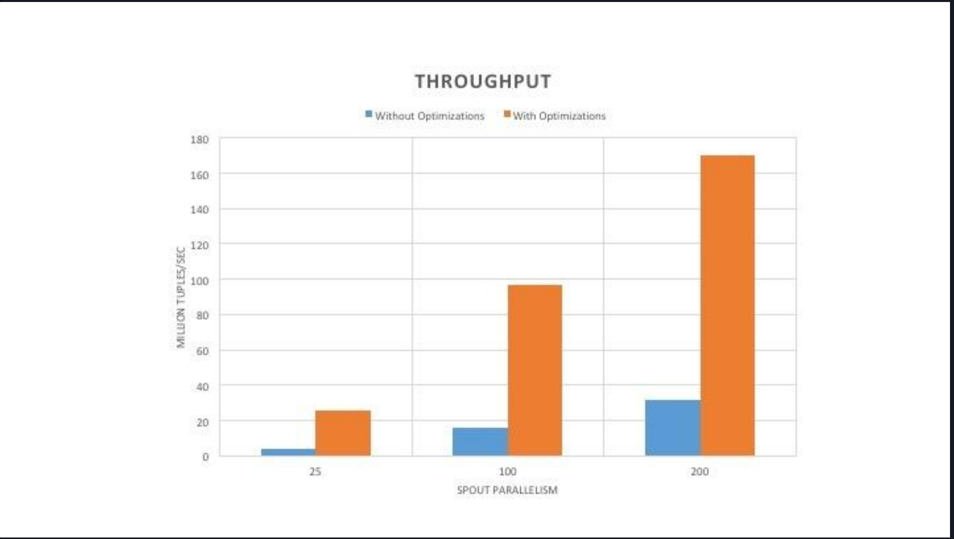


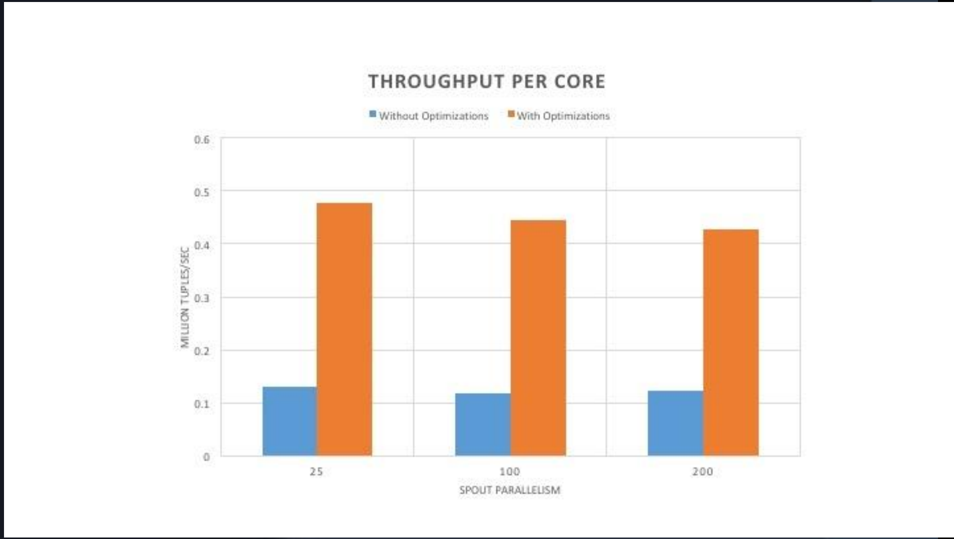
Backpressure example: B3 in the container A triggers backpressure, which is broadcasted to all Stream managers to stop local Spouts.

TABLE OF CONTENTS 大纲

Recent Improvements

Performance improvement





Recent Improvements

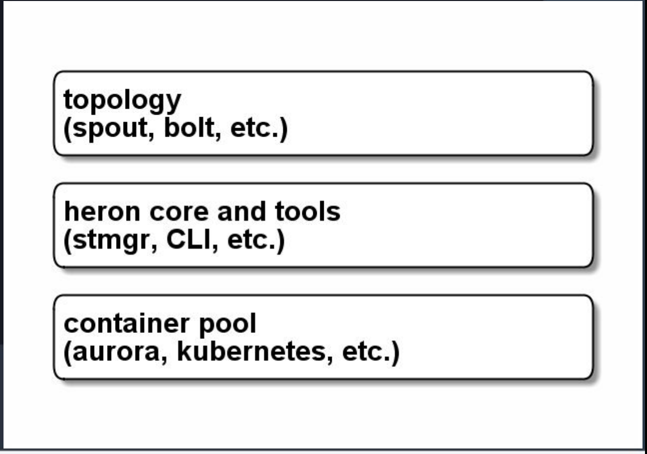
Resource managers

Service Provider Interface (SPI)

- Modular plugins
- https://github.com/twitter/heron/tree/master/heron/spi
- Scheduler implementation vs. delegation

Supported resource pools

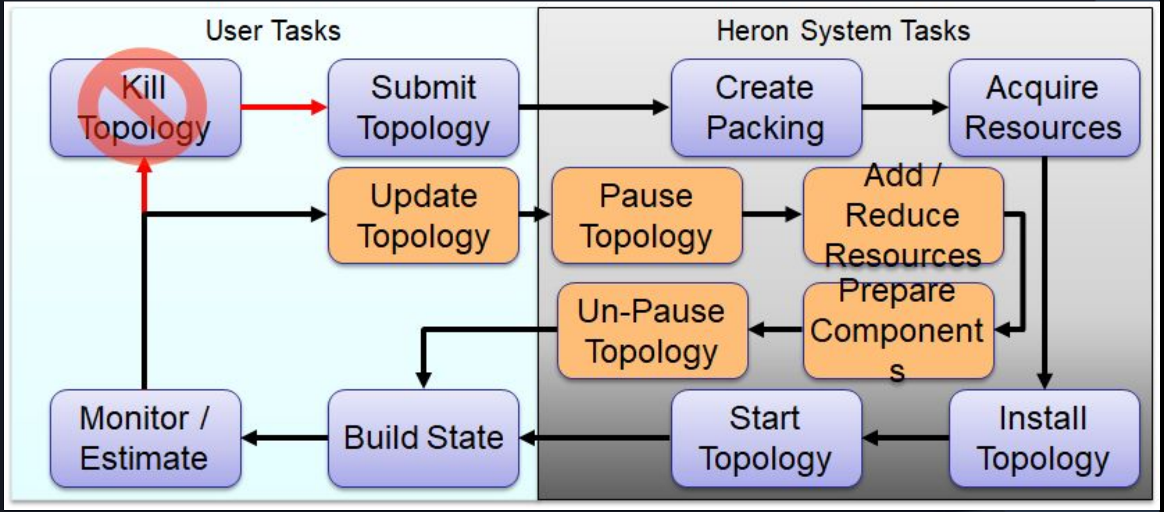
- Mesos/Aurora/Marathon
- Yarn
- Kubernetes
- Slurm
- Local



Recent Improvements

Elastic runtime scaling

- Update parallelism at runtime
- Adapt to stream traffic load
- `heron update` command
- Minimize impact to running topology
- Intelligent packing algorithm



Recent Improvements

Stateful processing: effectively once

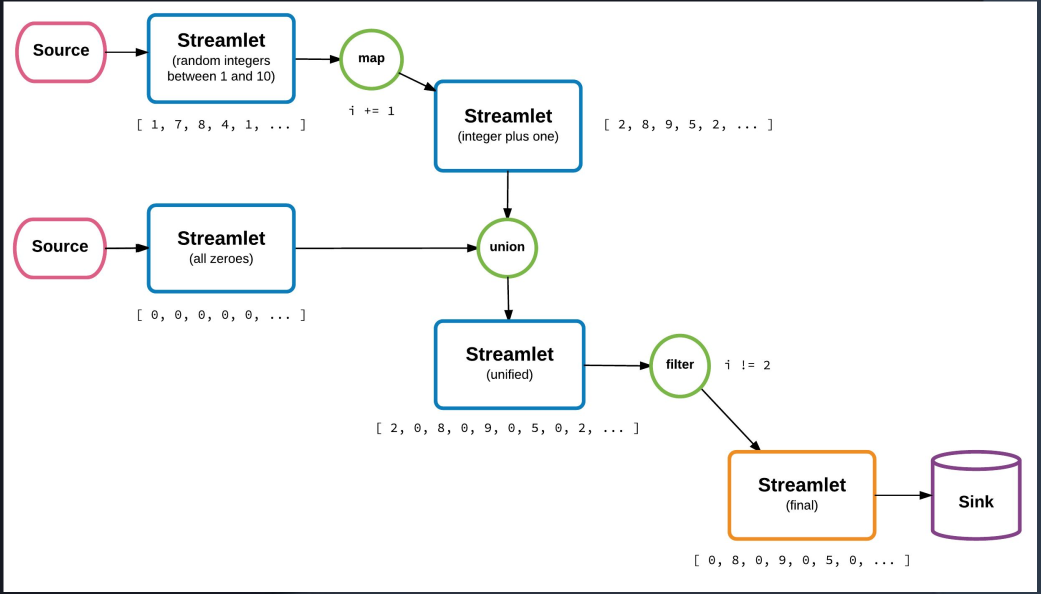
Delivery semantics:

- At most once
- At least once
- Effectively once
  - Distributed snapshot/state checkpointing
  - At-least-once event delivery plus roll back
- Exactly once

Recent Improvements

High level DSL: functional API

Domain	Original topology API	Heron Functional API
Programming style	Procedural, processing component based	Functional
Abstraction level	Low level. Developers must think in terms of "physical" spout and bolt implementation logic.	High level. Developers can write processing logic in an idiomatic fashion in the language of their choice, without needing to write and connect spouts and bolts.
Processing model	Spout and bolt logic must be created explicitly, and connecting spouts and bolts is the responsibility of the developer	Spouts and bolts are created for you automatically on the basis of the processing graph that you build



Recent Improvements

Multiple languages support

- Same data model with Java
- Python:
  - Document https://twitter.github.io/heron/docs/developers/python/topologies/
  - API https://twitter.github.io/heron/api/python/
  - Example https://github.com/twitter/heron/tree/master/examples/src/python
  - Code repositoty https://github.com/twitter/heron/tree/master/heronpy
- C++:
  - Code repository https://github.com/twitter/heron/tree/master/heron/api/src/cpp

Recent Improvements

Self regulating: health mgr/dhalion

Motivation:

➢ the manual, time-consuming and error-prone tasks of tuning various configuration knobs to achieve service level objectives (SLO) as well as the maintenance of SLOs in the face of sudden, unpredictable load variation and hardware or software performance degradation

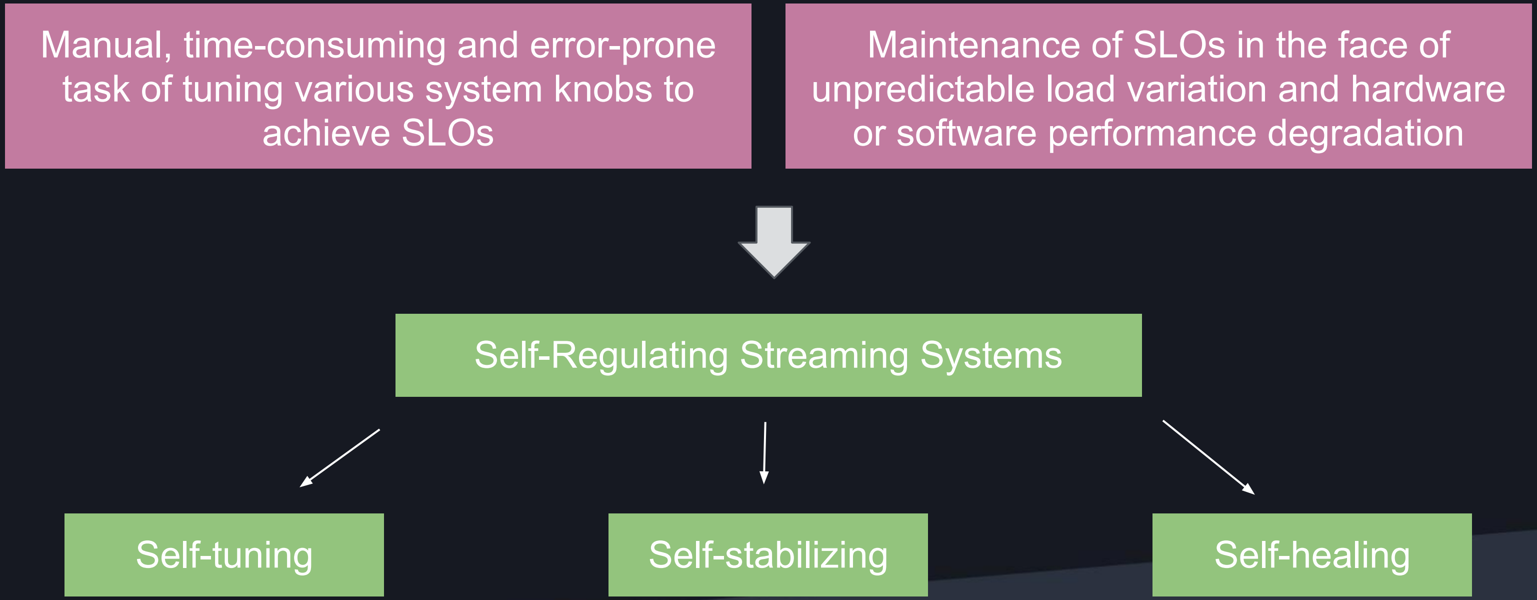
What is Dhalion:

➢ a system that provides self-regulation capabilities to underlying streaming systems

Floratou, Avrilia, et al. "Dhalion: self-regulating stream processing in heron." Proceedings of the VLDB Endowment 10.12 (2017): 1825-1836.

TABLE OF CONTENTS 大纲

Self-Regulating Streaming Systems



Manual, time-consuming and error-prone task of tuning various system knobs to achieve SLOs

Maintenance of SLOs in the face of unpredictable load variation and hardware or software performance degradation

ArchSummit 全球架构师峰会 2017
主办方 Geekbang> 极客邦科技 InfoQ

Self Regulating Challenges

Self-tuning

- Various tuning knobs
- Time consuming tuning phase
- The system should take as input as SLO and automatically configure the knobs.

Self Regulating Challenges

Self-stabilizing

- Streaming applications are long running
- Load variations are observed
- The system should react to external shocks and automatically reconfigure itself.

Self Regulating Challenges

Self-healing

- System performance can be affected by hardware or software delivering degraded quality of service
- The system should identify internal faults and attempt to recover from them.

TABLE OF
CONTENTS 大纲

Feedback Cycle

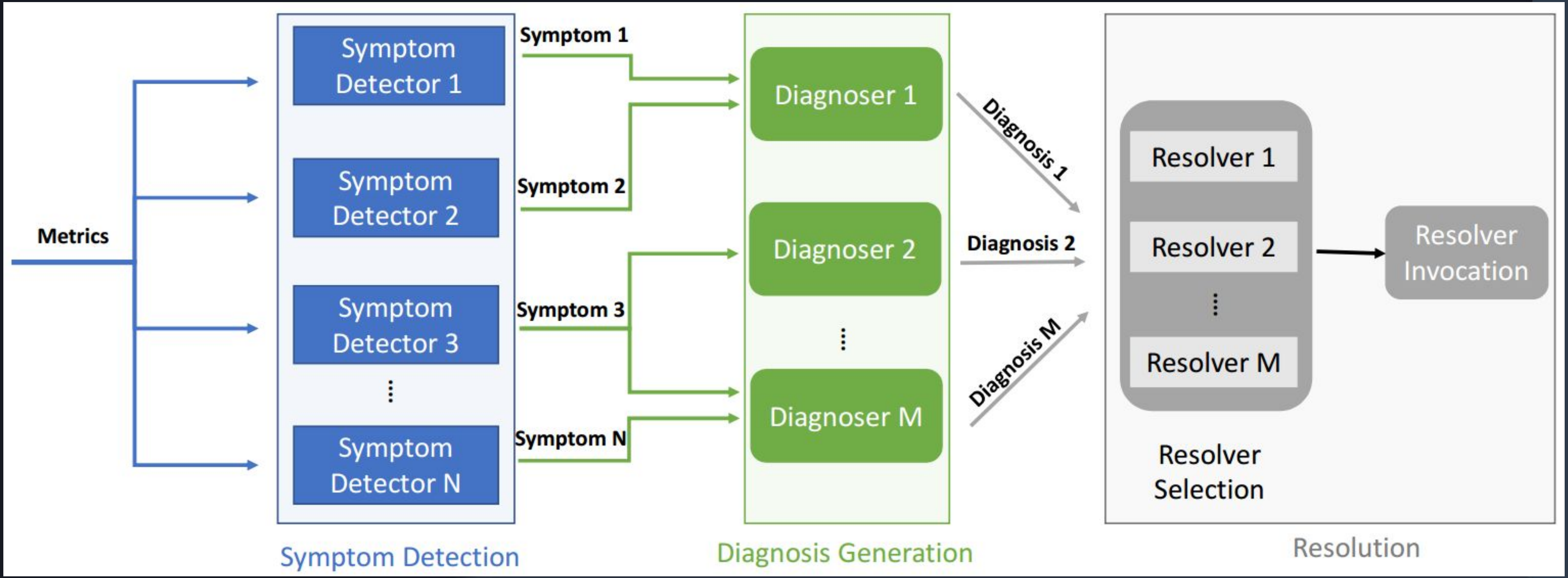
- Passive cycle
  - start backpressure(stream manager) -> cease spout(heron instance) -> stop backpressure(stream manager)

- Proactive cycle
  - [metrics -> metrics manager] -> metrics cache -> health manager -> [container/stream manager/spout/bolt -> metrics]

Dhalion terminology

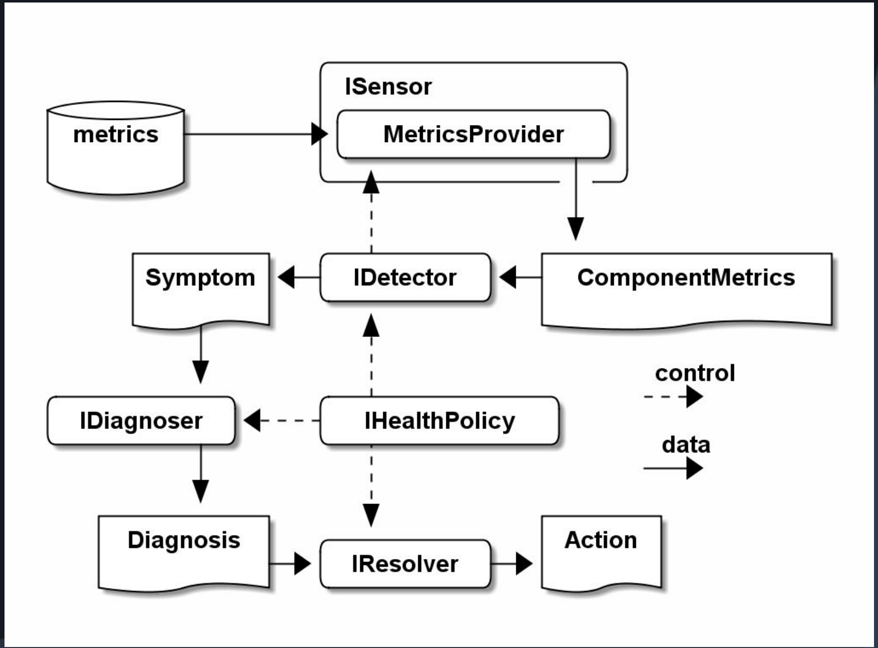
- Policy: Dhalion periodically invokes a policy which evaluates the status of the topology, identifies potential problems and takes appropriate actions to resolve them.
- Detection: Dhalion observes the system state by collecting various metrics from the underlying streaming system.
    - Symptom: Based on the metrics collected, Dhalion attempts to identify symptoms that can potentially denote that the health of the streaming application has been compromised.
- Diagnosis: After collecting various symptoms, Dhalion attempts to find one or more diagnoses that explain them.
- Resolution: Once a set of diagnoses has been found, the system evaluates them and explores the possible actions that can be taken to resolve the problem.

Dhalion policy phases

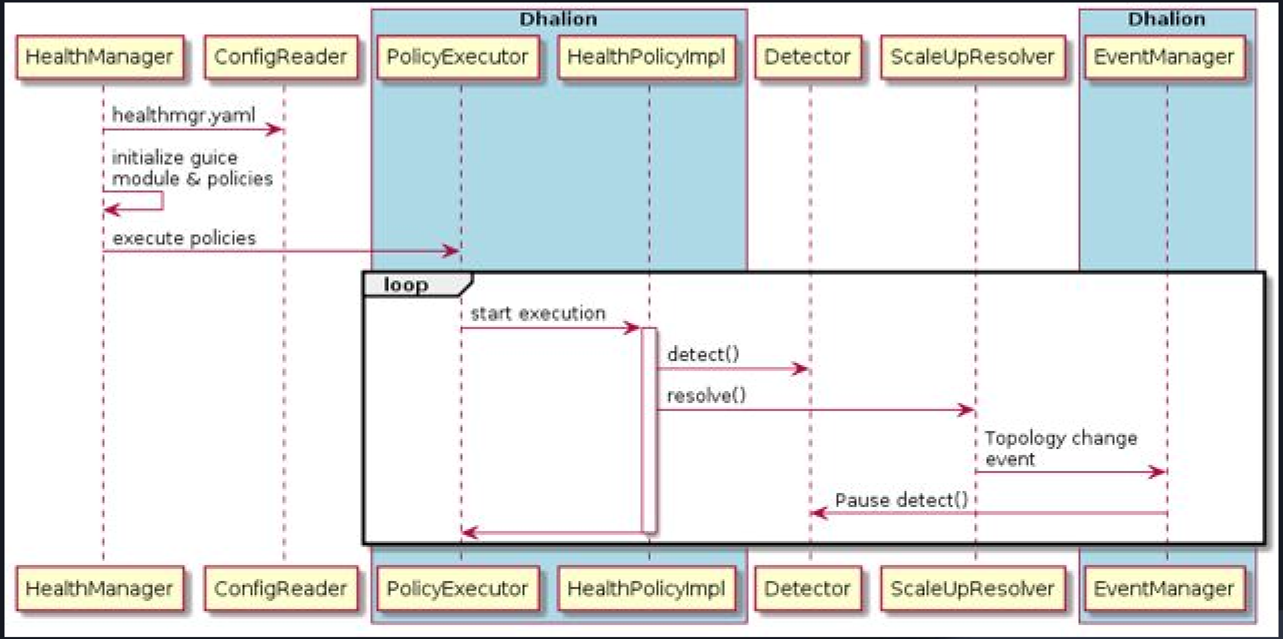


HealMgr workflow

- Data flow
  - metrics -> component metrics
  - component metrics -> symptom
  - symptom -> diagnosis
  - diagnosis -> action
- Control flow
  - policy -> detector/diagnoser/resolver
  - detector -> sensor/metrics provider



Dhalion in HealthMgr



Dhalion from Microsoft https://github.com/Microsoft/Dhalion

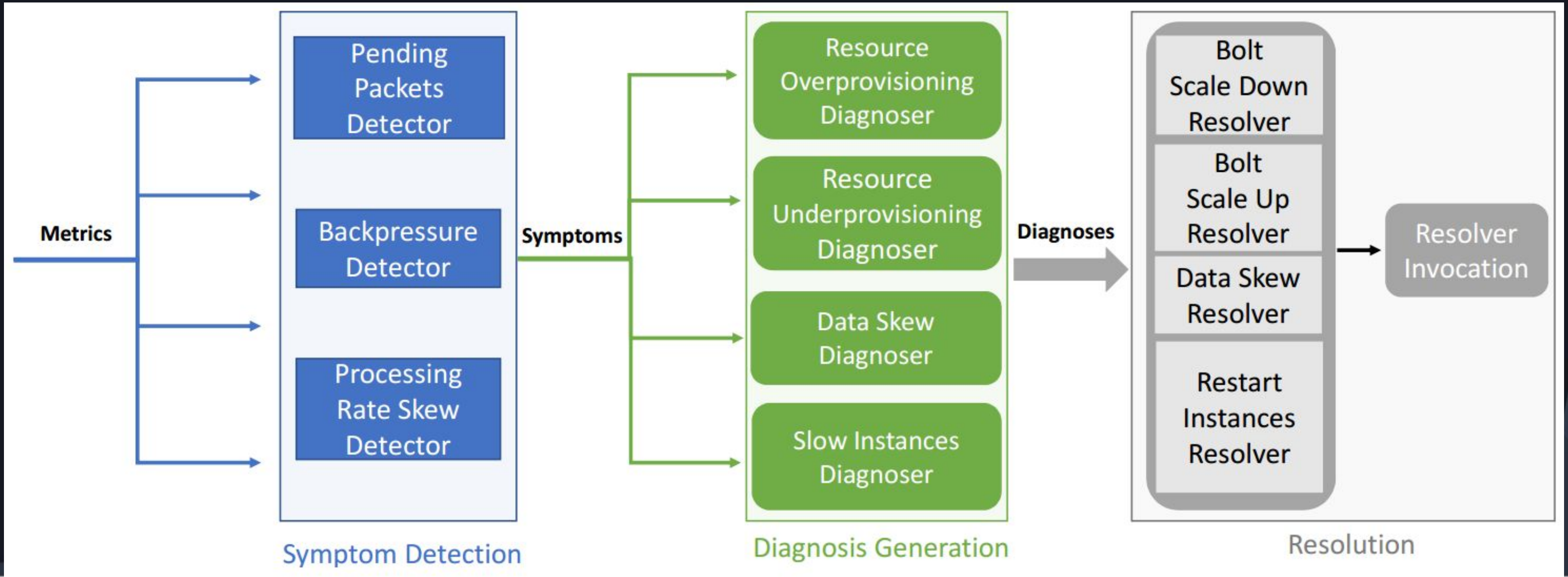
Action log and blacklist

- It is possible that a diagnosis produced by Dhalion is erroneous and thus, an incorrect action is performed that will not eventually resolve the problem.
- For this reason, after every action is performed, Dhalion evaluates whether the action was able to resolve the problem or brought the system to a healthier state.
- If an action does not produce the expected outcome then it is blacklisted and it is not repeated again.

TABLE OF CONTENTS 大纲

Dynamic Resource Provisioning

The major goal is
to scale up and down topology resources as needed
while still keeping the topology in a steady state
where backpressure is not observed.



Dynamic Resource Provisioning
Symptom detection phase

- The Pending Packets Detector focuses on the Stream Manager queue corresponding to each Heron Instance. Each Stream Manager queue temporarily stores packets that are pending for processing by the corresponding Heron Instance. This Symptom Detector examines the number of pending packets in the queues of the Heron Instances that belong to the same bolt, and denotes whether these Heron Instances have similar queue sizes or whether outliers are observed.
- The Backpressure Detector examines whether the topology experiences backpressure by evaluating the appropriate Stream Manager metrics. The existence of backpressure shows that the system is not able to achieve maximum throughput.
- The Processing Rate Skew Detector examines the number of tuples processed by each Heron Instance during the measurement period (processing rate). It then identifies whether skew in the processing rates is observed at each topology stage.

Dynamic Resource Provisioning

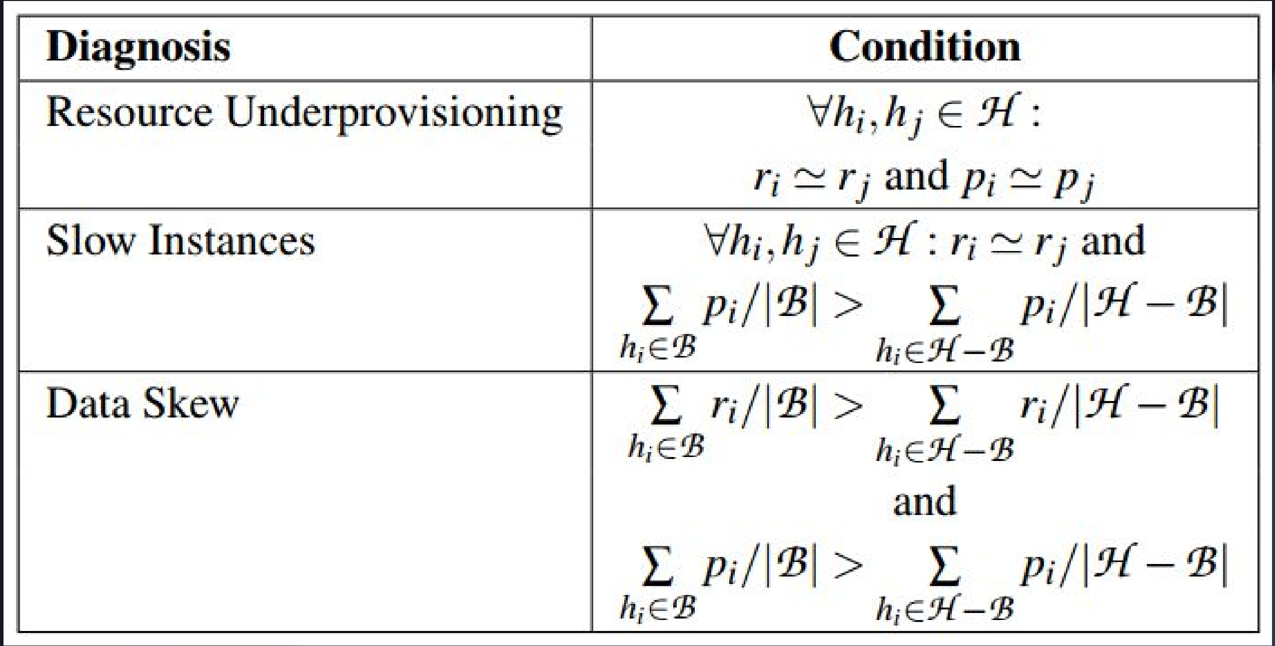
Diagnosis generation phase

h: heron instance

r: processing rate

p: pending packets

B: subset of H



Diagnosis	Condition
Resource Underprovisioning	$\forall h_i, h_j \in \mathcal{H}:$ $r_i \simeq r_j$ and $p_i \simeq p_j$
Slow Instances	$\forall h_i, h_j \in \mathcal{H}: r_i \simeq r_j$ and $\sum_{h_i \in \mathcal{B}} p_i / \lvert\mathcal{B}\rvert > \sum_{h_i \in \mathcal{H}-\mathcal{B}} p_i / \lvert\mathcal{H}-\mathcal{B}\rvert$
Data Skew	$\sum_{h_i \in \mathcal{B}} r_i / \lvert\mathcal{B}\rvert > \sum_{h_i \in \mathcal{H}-\mathcal{B}} r_i / \lvert\mathcal{H}-\mathcal{B}\rvert$ and $\sum_{h_i \in \mathcal{B}} p_i / \lvert\mathcal{B}\rvert > \sum_{h_i \in \mathcal{H}-\mathcal{B}} p_i / \lvert\mathcal{H}-\mathcal{B}\rvert$

Dynamic Resource Provisioning

Resolution phase

- Restart Instances Resolver: moves the slow Heron Instances to new containers
- Data Skew Resolver: adjusts the hash function used to distribute the data to the bolts
- Bolt Scale Up Resolver:
  - To determine the scale up factor, the Resolver computes the percentage of the total amount of time that the Heron Instances spent suspending the input data over the amount of time where backpressure was not observed. This percentage essentially denotes the portion of the input load that the Heron Instances could not handle.

Dynamic Resource Provisioning

Evaluation

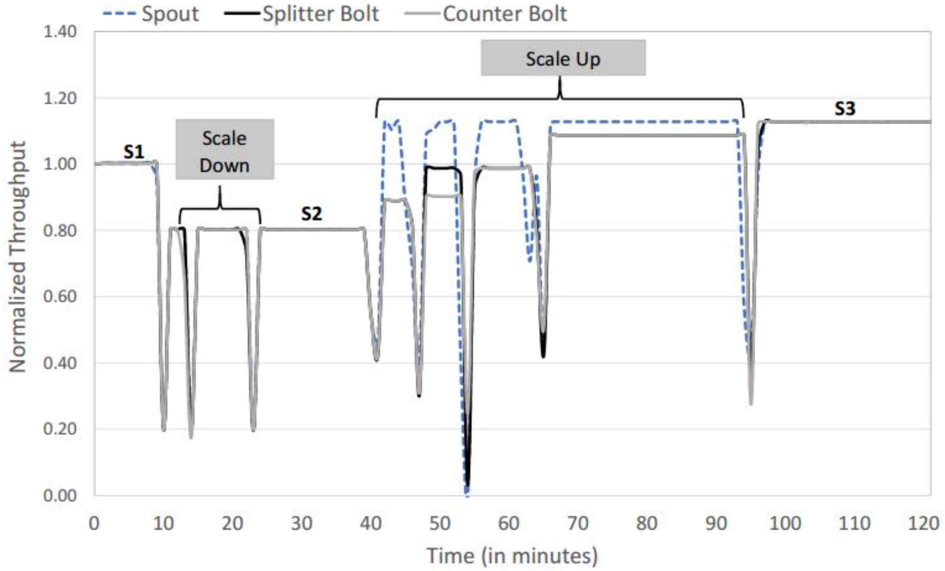


Figure 5: Dhalion's reactions during load variations

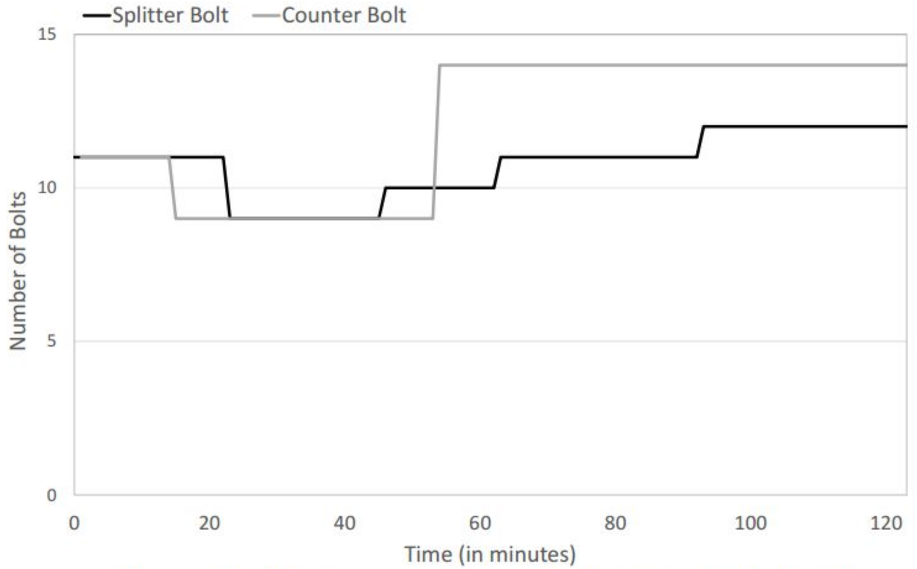


Figure 6: Number of Heron Instances provisioned during load variations

Satisfying Throughput SLOs

- Emit Count Detector: computes the total rate at which spouts emit data
- Throughput SLO Violation Diagnoser
- Spout Scale Up Resolver: increases the number of Heron Instances of the spout
  - In case the policy increases the spout parallelism, the topology might experience backpressure due to the increase of the input load. In this case, the Throughput SLO Policy employs the components used by the Dynamic Resource Provisioning Policy to automatically adjust the resources assigned to the bolts so that the topology is brought back to a healthy state.

Satisfying Throughput SLOs

Evaluation

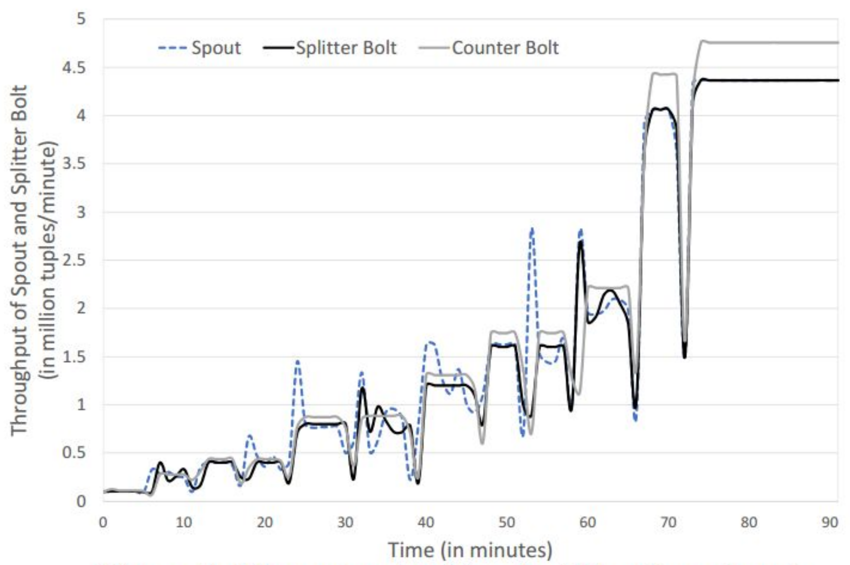


Figure 7: Throughput achieved while attempting to satisfy a throughput SLO

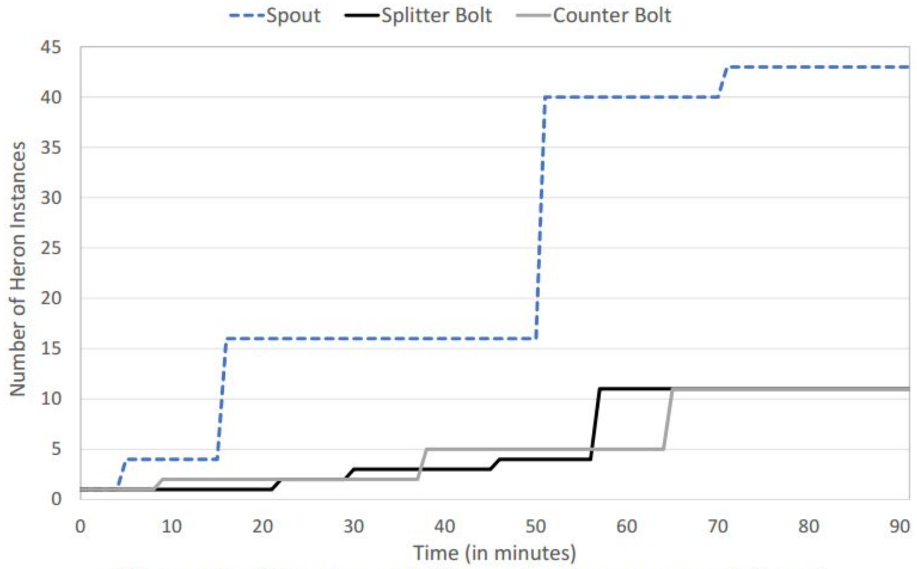


Figure 8: Number of Heron Instances provisioned while attempting to satisfy a throughput SLO

Curious to know more

- Floratou, Avrilia, et al. "Dhalion: self-regulating stream processing in heron." Proceedings of the VLDB Endowment 10.12 (2017): 1825-1836.
- Kulkarni, Sanjeev, et al. "Twitter heron: Stream processing at scale." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.
- Fu, Maosong, et al. "Twitter Heron: Towards Extensible Streaming Engines." Data Engineering (ICDE), 2017 IEEE 33rd International Conference on. IEEE, 2017.
- Fu, Maosong, et al. "Streaming@ Twitter." IEEE Data Eng. Bull. 38.4 (2015): 15-27.

Open Source

- Webpage https://twitter.github.io/heron/
- Github https://github.com/twitter/heron

THANK YOU

如有需求，欢迎至 [讲师交流会议室] 与我们的讲师进一步交流

ArchSummit
全 球 架 构 师 峰 会 2017