
# Dhalion: Self-Regulating Stream Processing in Heron

Large Scale Data Processing and Optimisation: Session 4

**Cristian (cb2015@cam.ac.uk)**

# Content

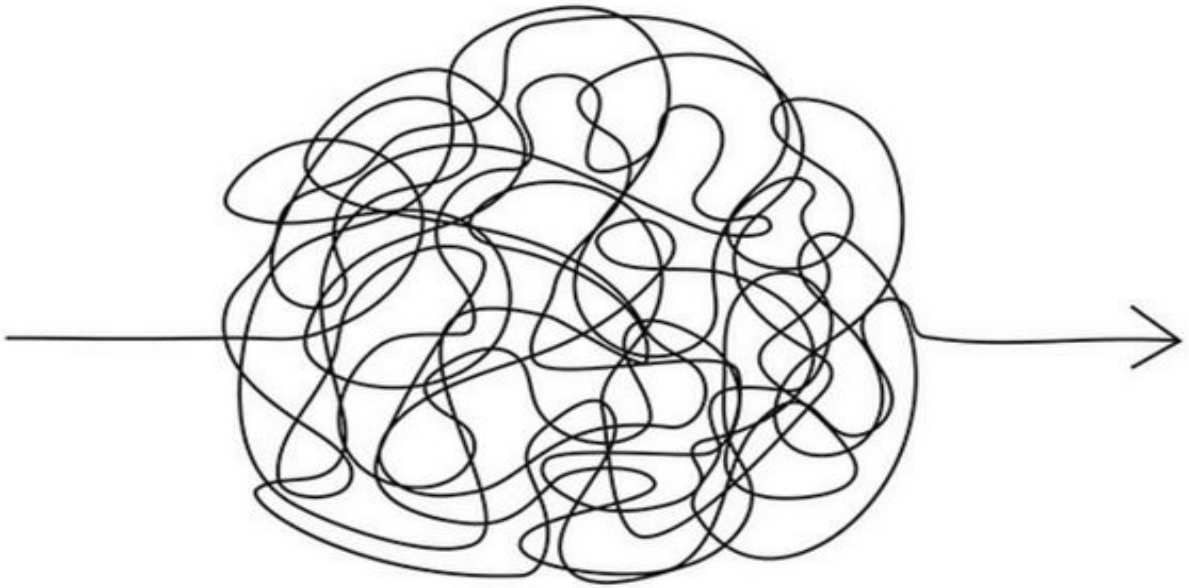
- Self-regulating Stream Processing
- Dhalion
- A concrete example: Heron
- Evaluation
- Critique and Future Work

# Distributed Stream Processing Systems



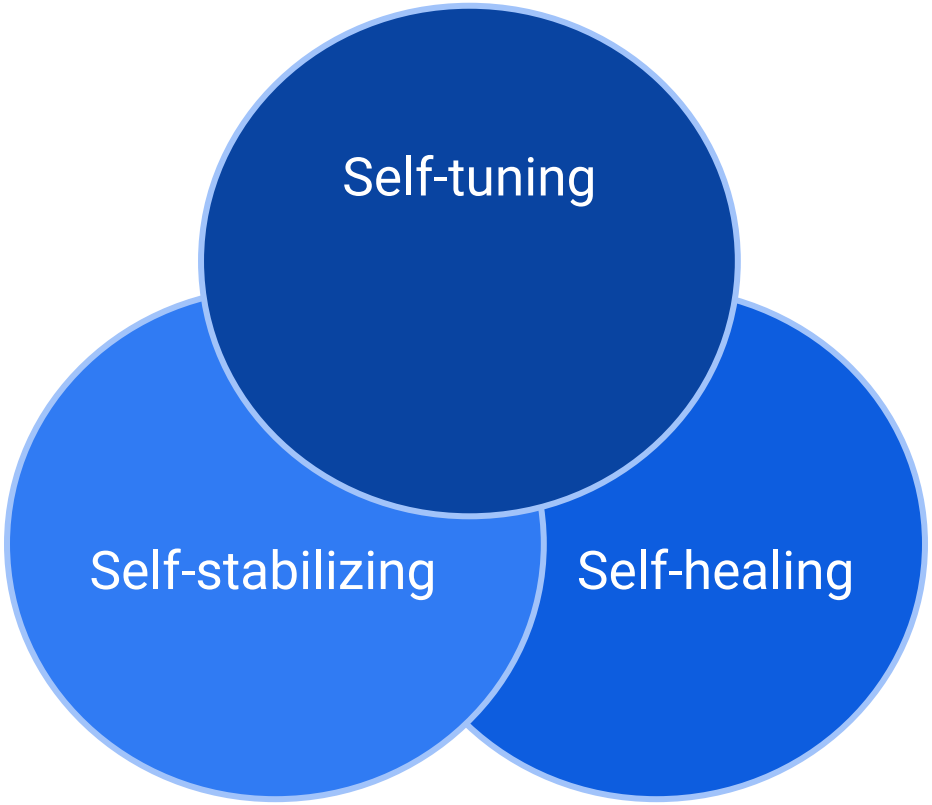

# Common problems

- They are hard to tune
- Balancing competing objectives
- Unpredictable load spikes
- Software and hardware degradation



# Self-Regulating Stream Processing

The system should automatically tune parameters to satisfy an SLO

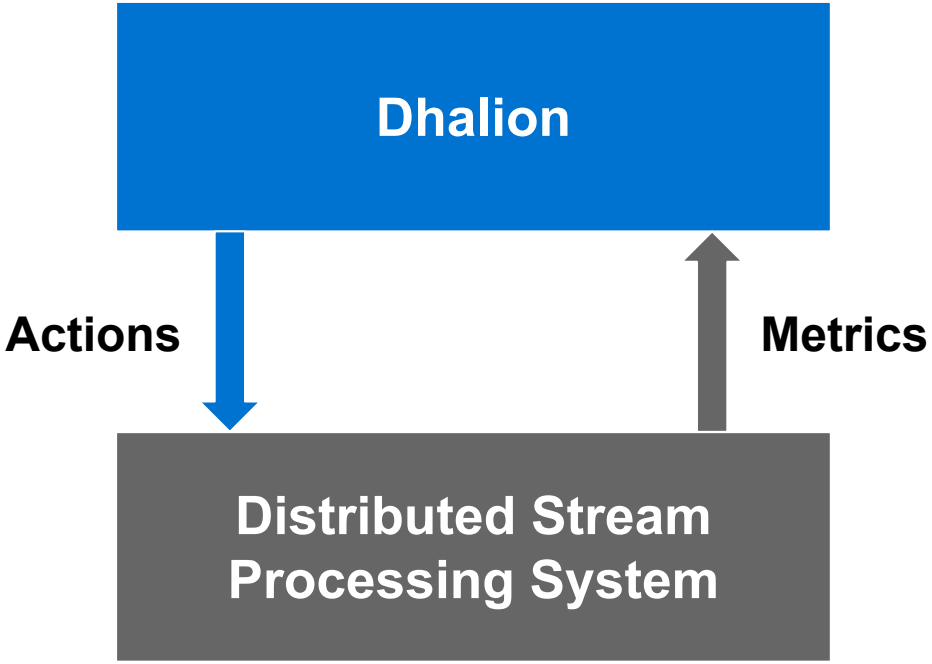



The system should adapt to changing conditions

The system should identify degrading components and recover from them

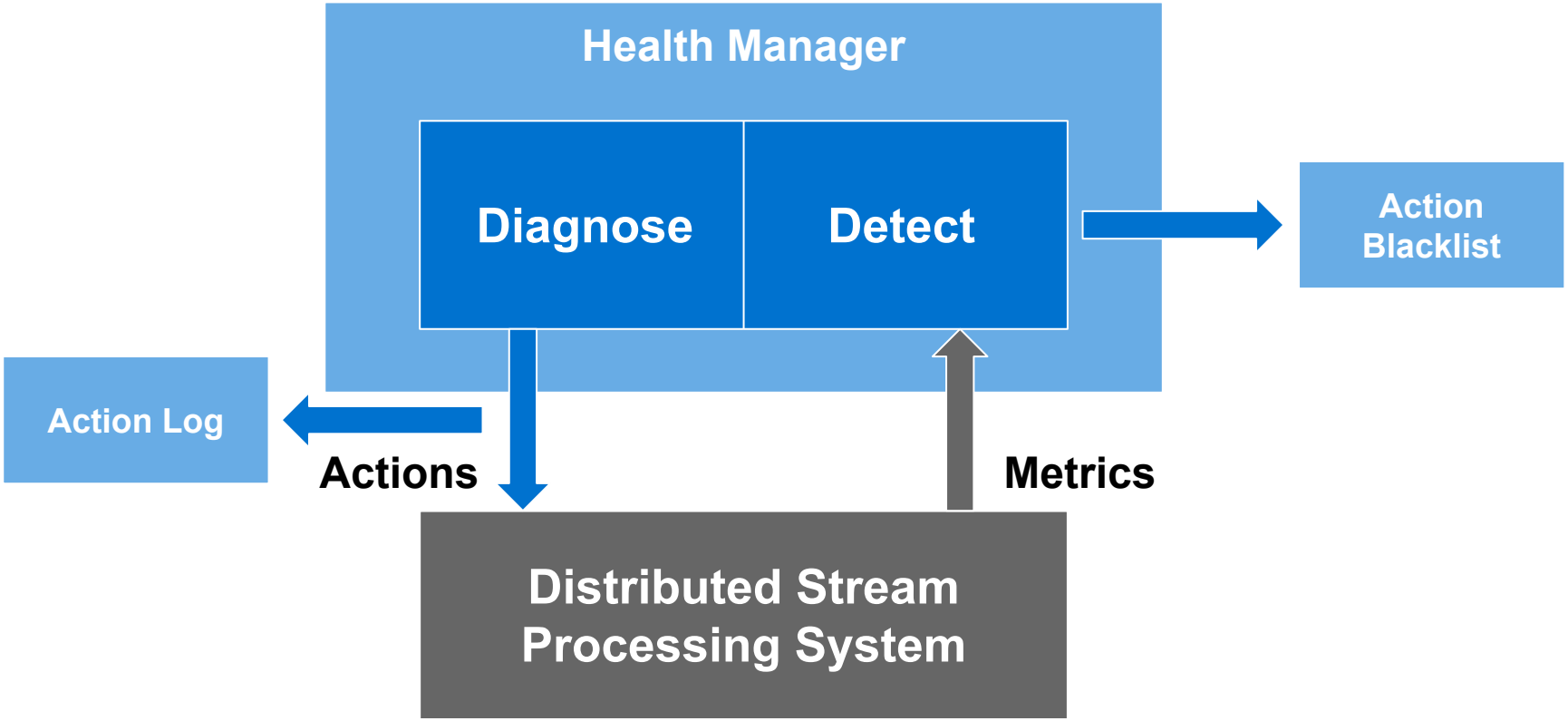
# Dhalion

Dhalion is built to make existent stream processing systems self-regulated.



# Dhalion

Dhalion periodically runs code to detect problems, diagnose them and take an appropriate action.



# Dhalion: Health Manager

A **policy** is a piece of code which optimises the system for a given objective.

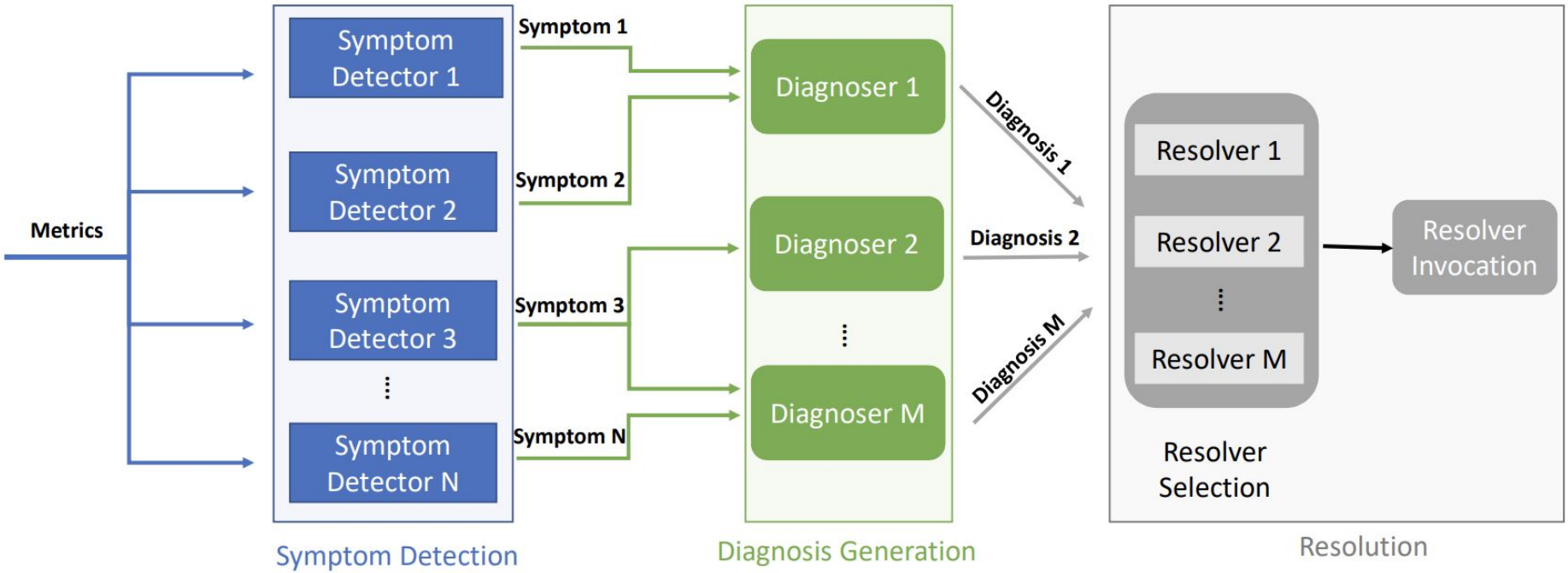



**Figure 1: Dhalion policy phases**

Image from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Dhalion Policy API

An API for custom Detectors, Diagnosers and Resolvers exists.

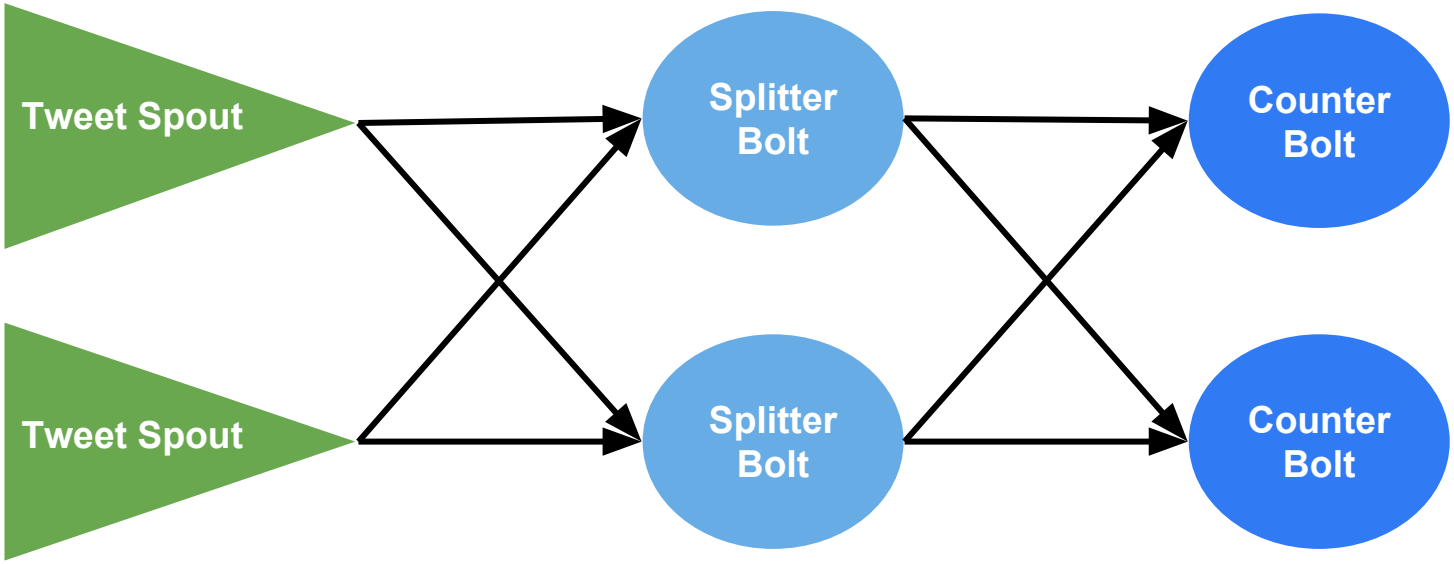
```
public interface IDetector {

  Collection<Symptom> detect(Collection<Measurement> metrics) {

      throw new UnsupportedOperationException();

}
```

# Dhalion: Action Log & Blacklist

| Action Log | Action Blacklist |
| --- | --- |
| • Record the actions taken by the Health Manager<br>• Can be used to debug and fine tune the policies<br>• The user can choose to keep the latest N logs or the logs from the past M hours | • Blacklist the actions which didn't produce the desired result for a particular diagnosis |

# Heron

Heron is essentially a directed acyclic graph (DAG) of **spouts** and **bolts**. Spouts are data sources while bolts are computations. The DAG is called a **topology**.




A key concept of Heron is **backpressure:** tell the parents in the graph to slow down when you can't keep up with the incoming data.

# Dhalion & Heron

Dhalion interacts with the Stream Processing System through the API of the system.

If Dhalion wanted to allocate more resources, it would call the corresponding API method in the Scheduler of Heron.

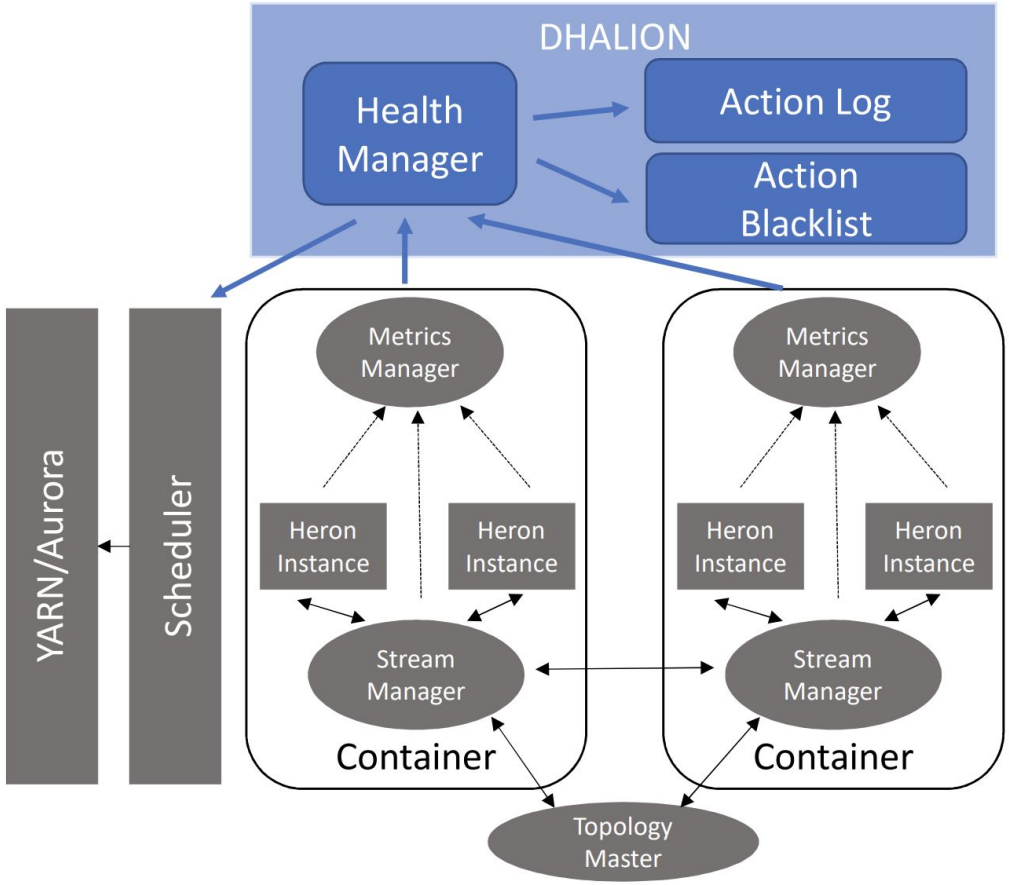



**Figure 2: Topology architecture**

Image from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Example: Dynamic Resource Provisioning

Resources must be allocated or deallocated as the load of the system changes.

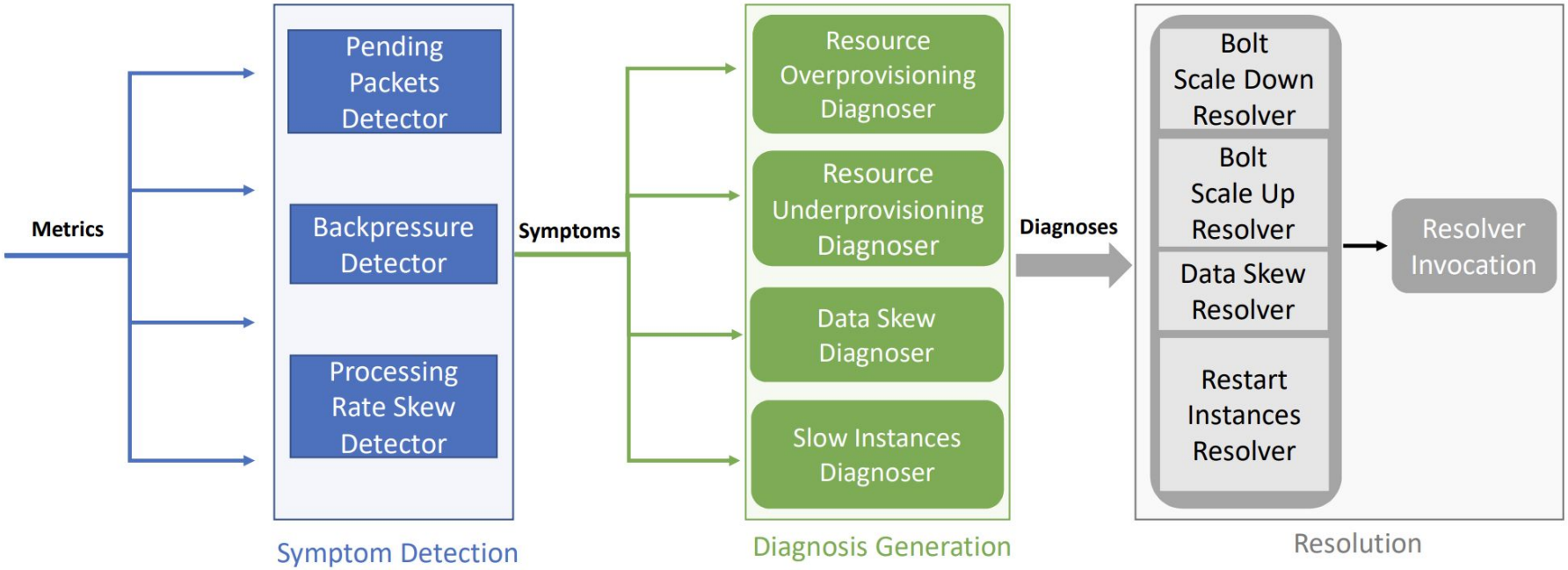



**Figure 3: Dynamic resource provisioning policy**

Image from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Evaluation: Dynamic Resource Provisioning

The data load is artificially decreased by 20% and then increased by 30% on a 3-stage topology for counting word frequencies from a stream of sentences.

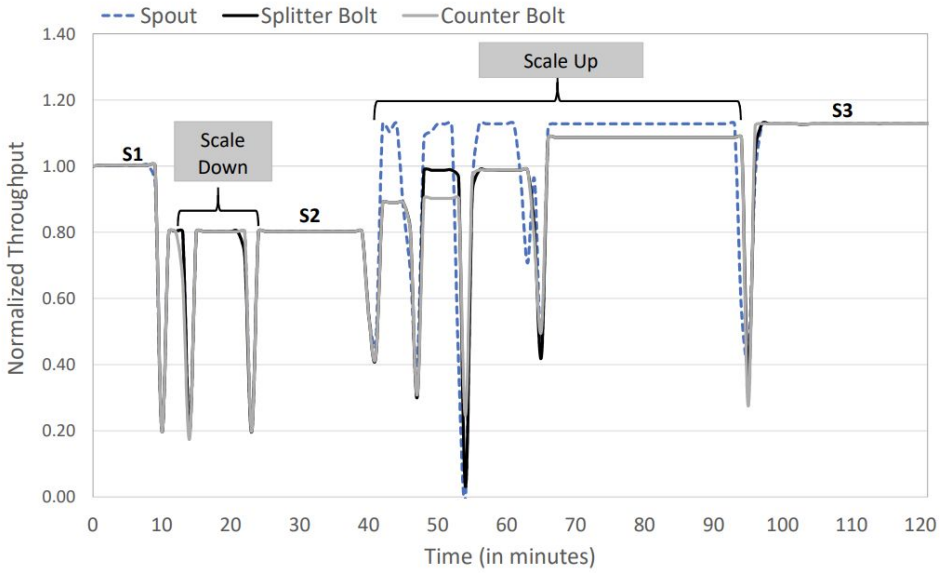



**Figure 5: Dhalion's reactions during load variations**

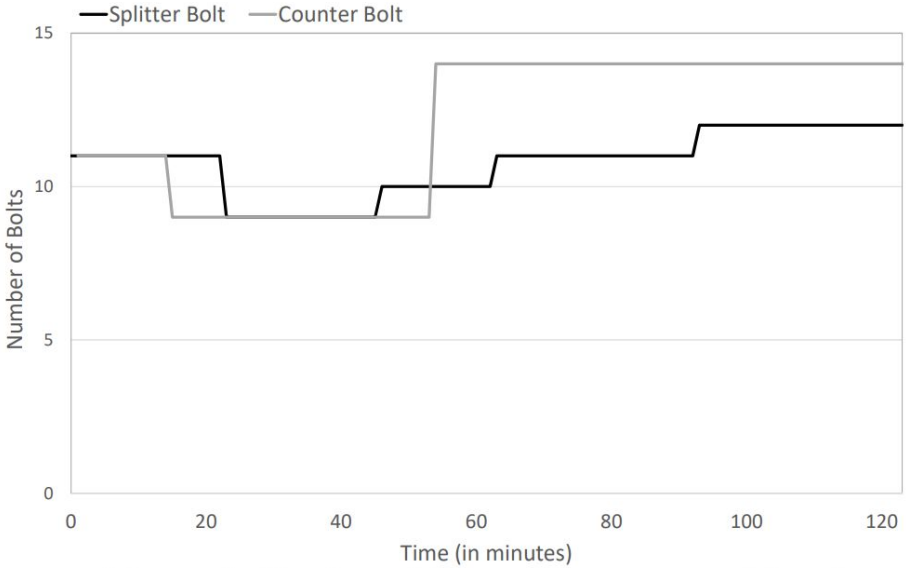



**Figure 6: Number of Heron Instances provisioned during load variations**

Images from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Evaluation: Satisfying Throughput Objective

On the same word count topology, Heron is initialised with one instance and Dhalion is set to satisfy 4 million tuples/minute at a steady state.

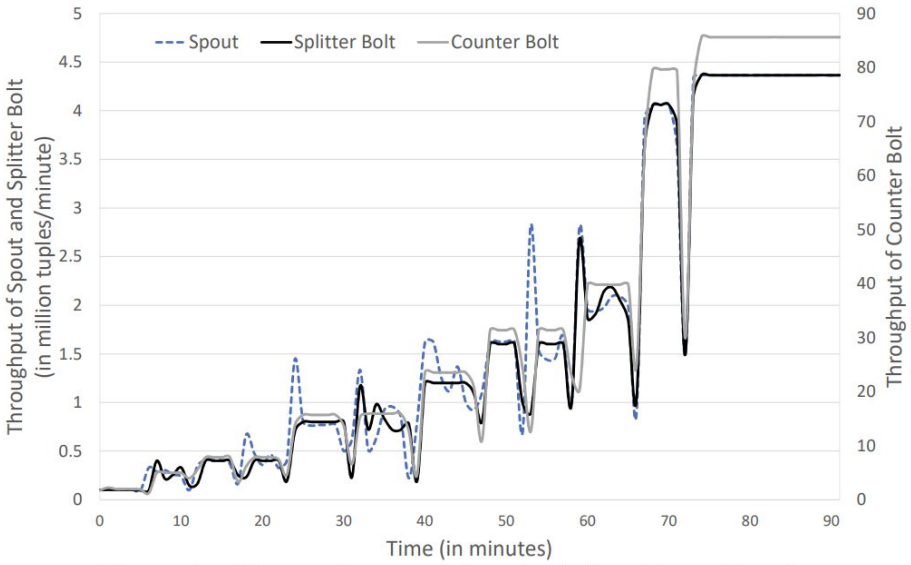



**Figure 7: Throughput achieved while attempting to satisfy a throughput SLO**

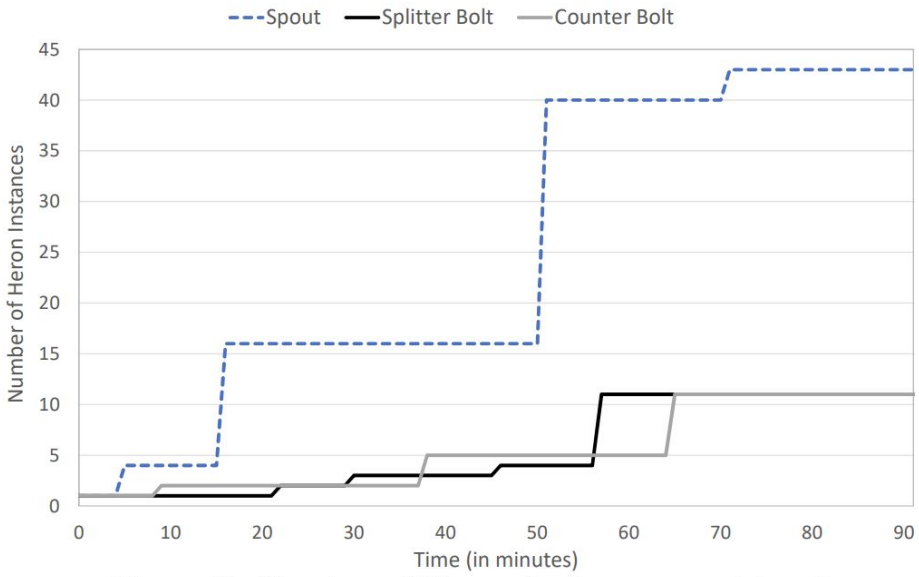



**Figure 8: Number of Heron Instances provisioned while attempting to satisfy a throughput SLO**

Images from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Evaluation: Mixed Scenario

A combination of under provisioned resources and slow instance is analysed.

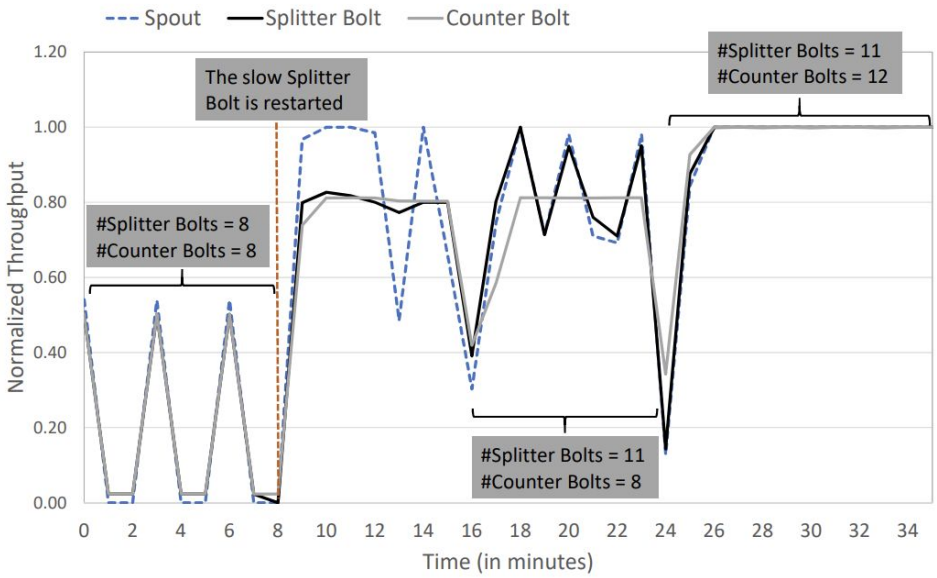



**Figure 9: Mixed scenario with underprovisioned resources and a 25% slower instance**

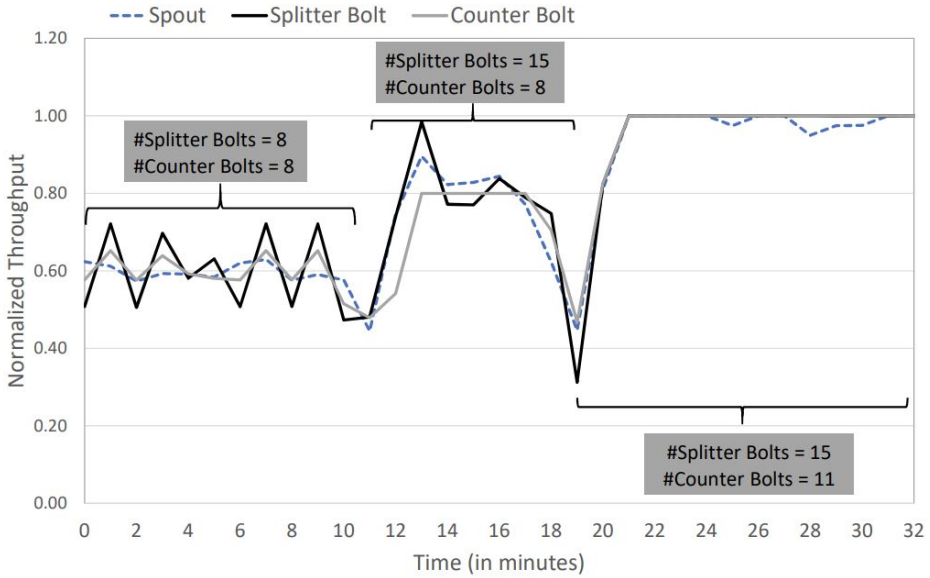



**Figure 10: Mixed scenario with underprovisioned resources and a 75% slower instance**

Images from [Dhalion: Self-regulating stream processing in Heron](#) Floratou et al., *VLDB 2017*

# Critique

| The Good | The Bad |
| --- | --- |
| • Continuously optimizes the system for many objectives.<br>• Runs alongside existing systems.<br>• Customisable and modular. | • The topology and scale used for evaluation are far from many real world scenarios.<br>• The need to tune the detectors, diagnosers and resolvers<br>• Edge cases for blacklisting<br>• How does it work in practice? |

# Future work

- All the phases of a policy can be treated as a pattern recognition problem.
- The optimisation of the system can be seen as a reinforcement learning problem
- Policies for Latency SLOs
- Reuse the execution of policy modules

# The End

Thank you!
Questions?