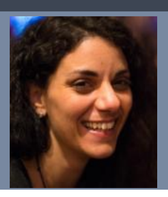
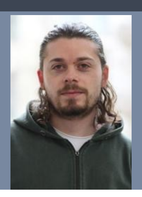
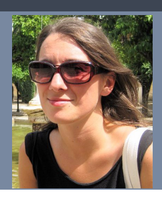
Three steps is all you need:

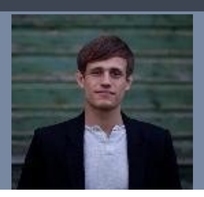
fast, accurate, automatic scaling decisions for distributed streaming dataflows


Vasiliki Kalavri


John Liagouris

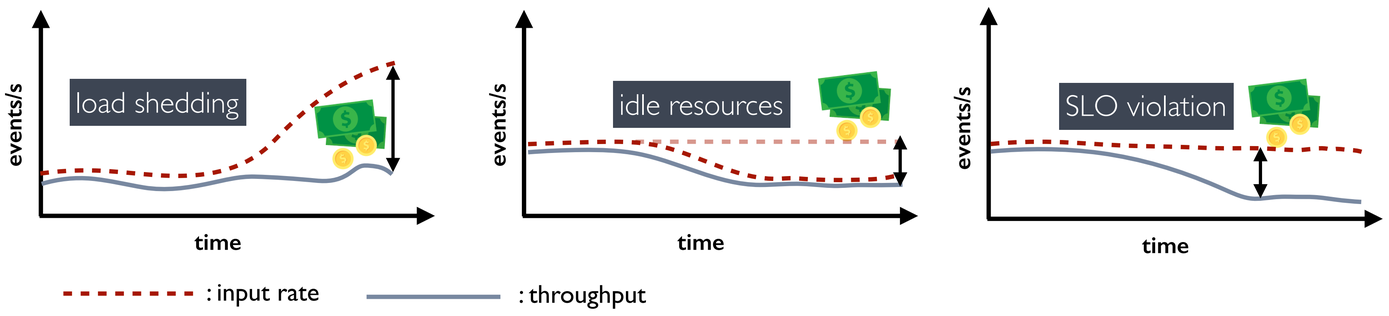

Moritz Hoffmann


Desislava Dimitrova


Matthew Forshaw


Timothy Roscoe

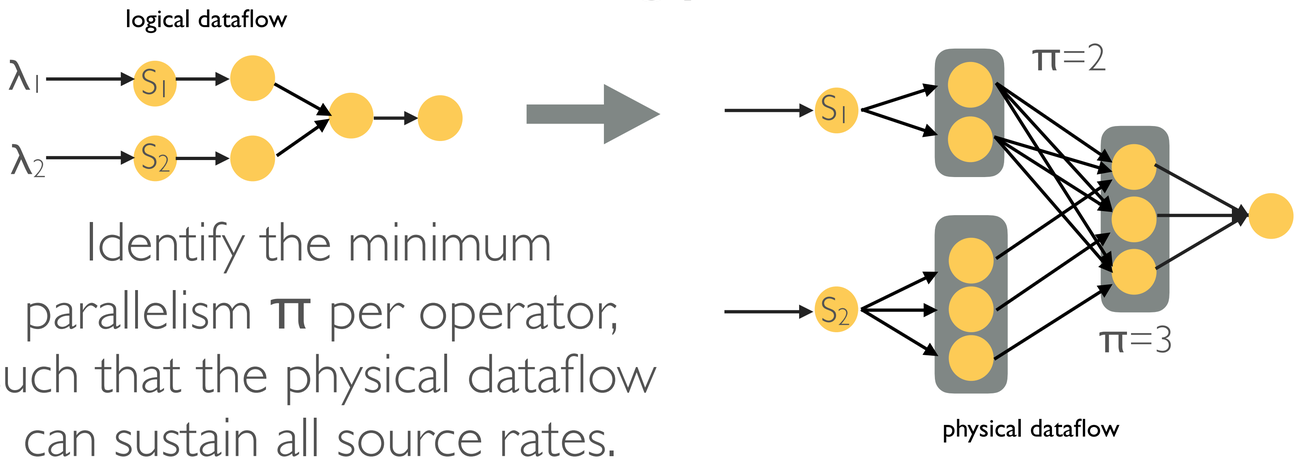
Any long-running streaming job will inevitably become over- or under- provisioned



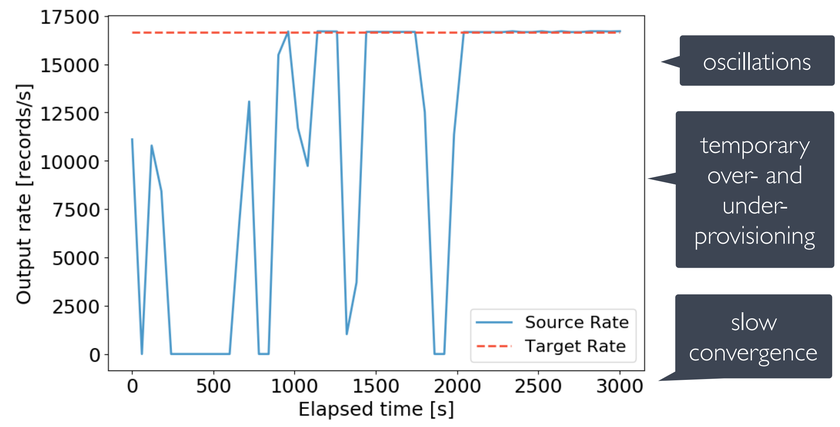
Existing scaling policies

System	Metrics	Policy	Scaling Action
Borealis	CPU, network slack, queue sizes	Rule-based	Load Shedding
StreamCloud	Average CPU, observed rates	Threshold-based	Speculative
Seep	User/system CPU time	Threshold-based	Speculative
IBM Streams	Congestion, observed rates	Threshold-based	Speculative
Spark Streaming	Pending tasks	Threshold-based	Speculative
Google Dataflow	CPU, backlog, observed rates	Heuristic	Speculative
Dhalion	Backpressure, queue sizes, observed rates	Rule-based	Speculative

The scaling problem



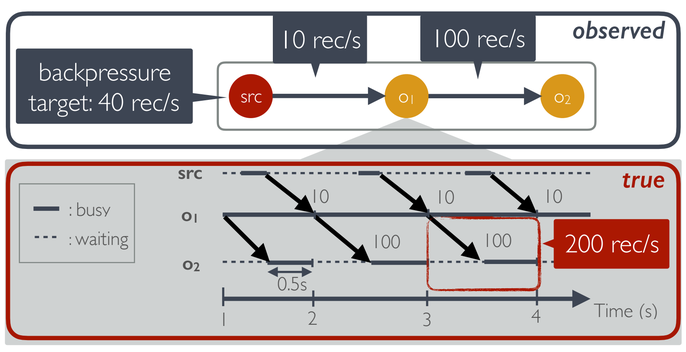
Identify the minimum parallelism **π** per operator, such that the physical dataflow can sustain all source rates.



effect of Dhalion's scaling actions in an initially under provisioned wordcount dataflow

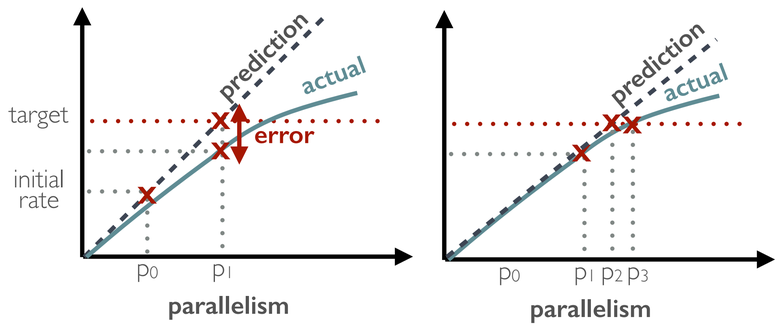
DS2: Automatic Scaling for Streaming Dataflows

collects true rates through system **instrumentation**



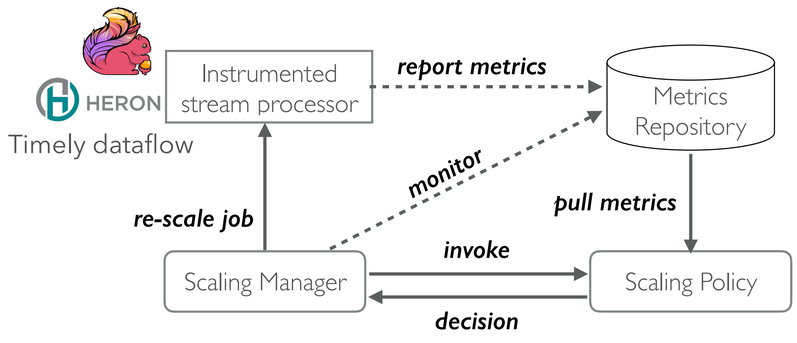
Useful time: deserialization + processing + serialization
True processing (output) rate: records processed (emitted) per unit of useful time

considers **ideal** scaling and dataflow **dependencies**



Optimal parallelism for o_i : $\frac{\text{aggregated true output rate of upstream ops}}{\text{average true processing rate of } o_i}$

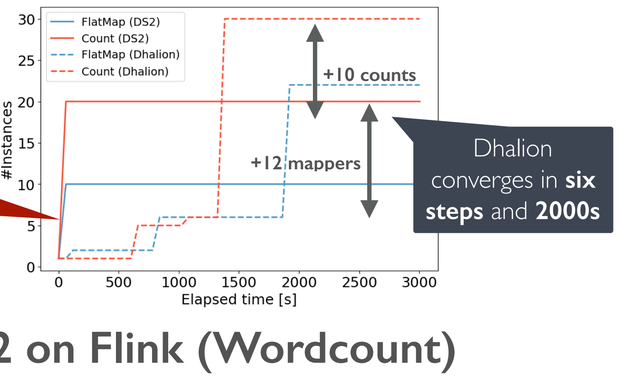
operates in an **online** and **reactive** setting



metrics per operator instance:
- **#records processed**
- **#records produced**
- **useful time** or **waiting time**

DS2 in action

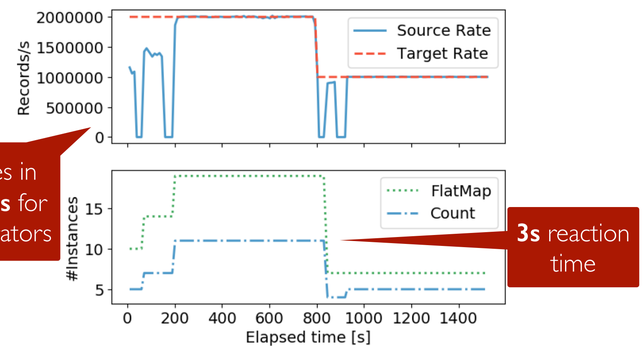
DS2 vs Dhalion on Heron (Wordcount)



DS2 converges in a **single step** and **60s**

Dhalion converges in **six steps** and **2000s**

DS2 on Flink (Wordcount)

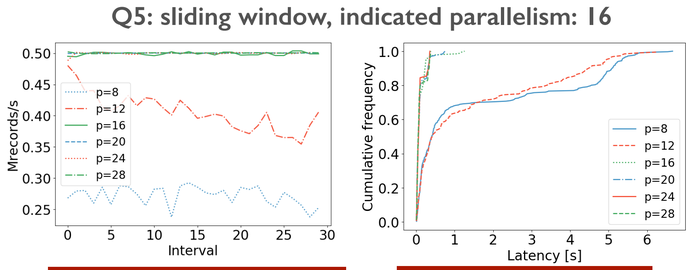


converges in **two steps** for both operators

3s reaction time

Accuracy (Nexmark)

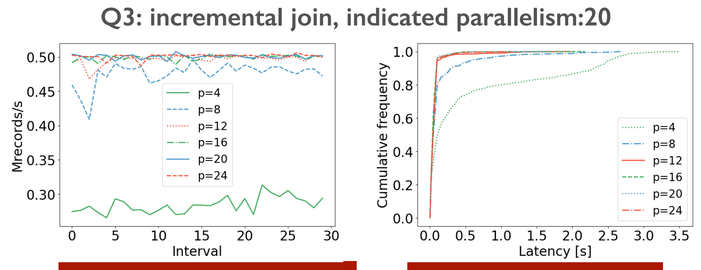
Q5: sliding window, indicated parallelism: 16



p < 16 does not meet the throughput target

p > 16 does not improve latency

Q3: incremental join, indicated parallelism:20



p < 20 does not meet the throughput target

p > 20 does not improve latency

Convergence (Nexmark)

up to **3 steps** when the initial configuration is far from optimal

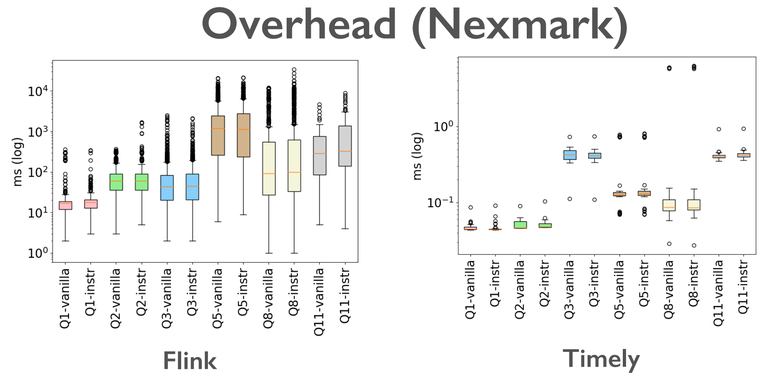
Initial configuration	Q1	Q2	Q3	Q5	Q8	Q11
8	12→**16**	11→13→**14**	16→**20**	14→15→**16**	**10**	12→22→**28**
12	**16**	**14**	18→**20**	**16**	**10**	22→**28**
16	**16**	12→**14**	**20**	**16**	8→**10**	26→**28**
20	**16**	13→**14**	**20**	14→**16**	8→**10**	**28**
24	**16**	**14**	**20**	14→**16**	8→**10**	**28**
28	**16**	**14**	**20**	13→**16**	8→**10**	**28**

parallelism

a single step for simple queries and initial configurations close to optimal

→ : **scaling action**

Overhead (Nexmark)



https://github.com/strymon-system/ds2

http://strymon.systems.ethz.ch

Systems Group, Department of Computer Science, ETH Zurich
firstname.lastname@inf.ethz.ch