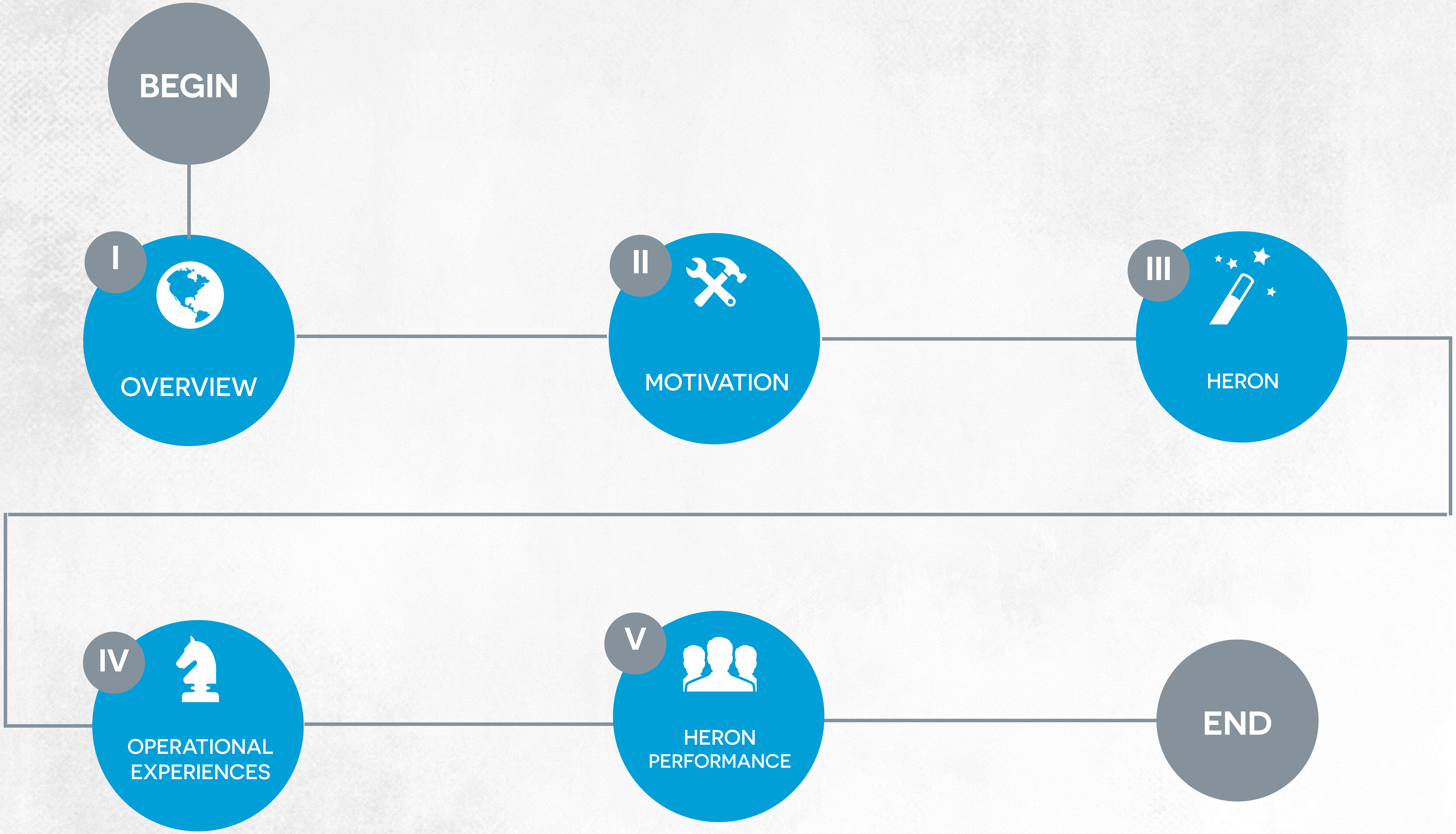
Flying Faster with Heron

KARTHIK RAMASAMY
@KARTHIKZ

#TwitterHeron

TALK OUTLINE



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!

TWITTER STORM

Streaming platform for analyzing **realtime data** as they arrive, so you can react to data **as it happens**.

GUARANTEED MESSAGE PROCESSING

HORIZONTAL SCALABILITY

ROBUST FAULT TOLERANCE

CONCISE CODE- FOCUS ON LOGIC

STORM TERMINOLOGY

TOPOLOGY

Directed acyclic graph

Vertices=computation, and edges=streams of data tuples

SPOUTS

Sources of data tuples for the topology

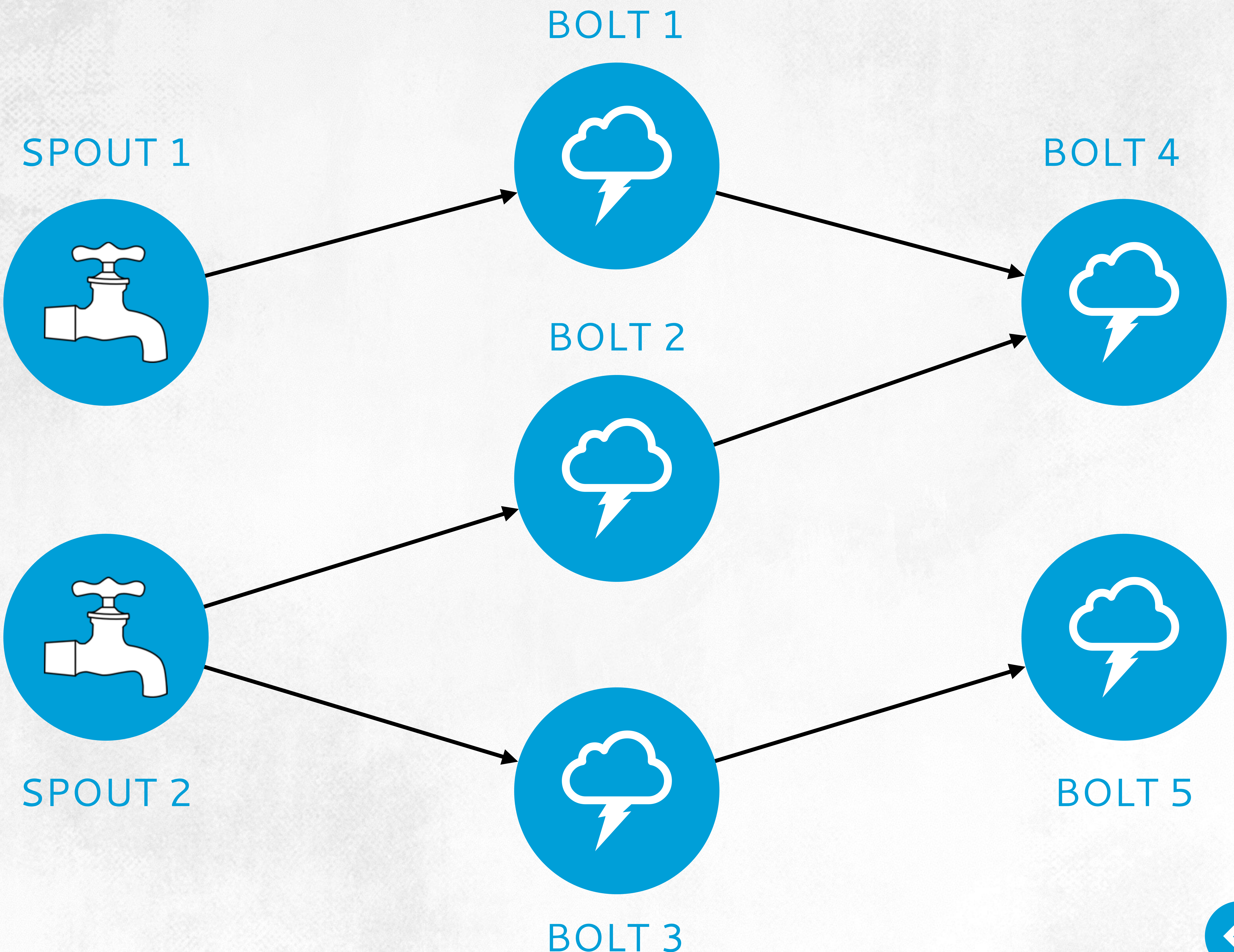
Examples - Event Bus/Kafka/Kestrel/MySQL/Postgres

BOLTS

Process incoming tuples and emit outgoing tuples

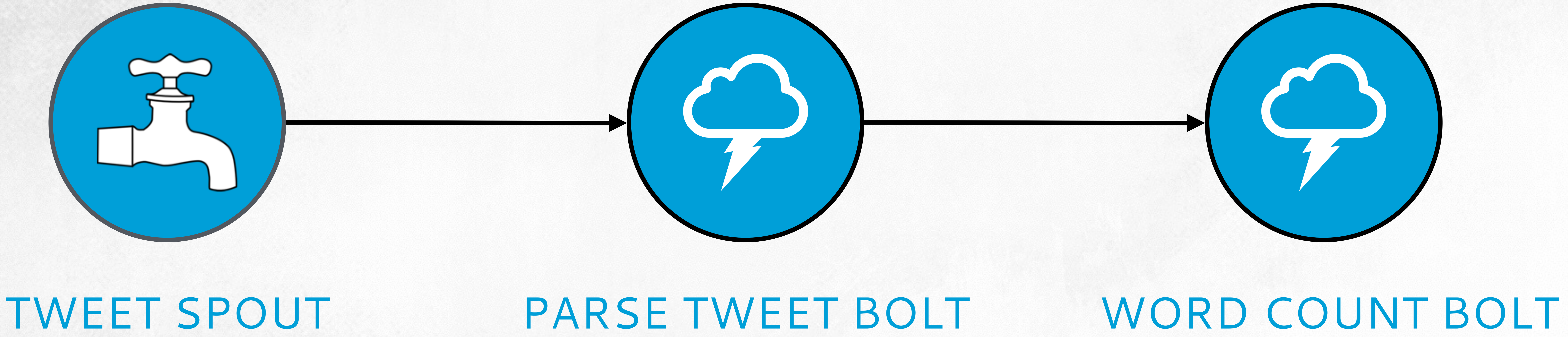
Examples - filtering/aggregation/join/arbitrary function

STORM TOPOLOGY



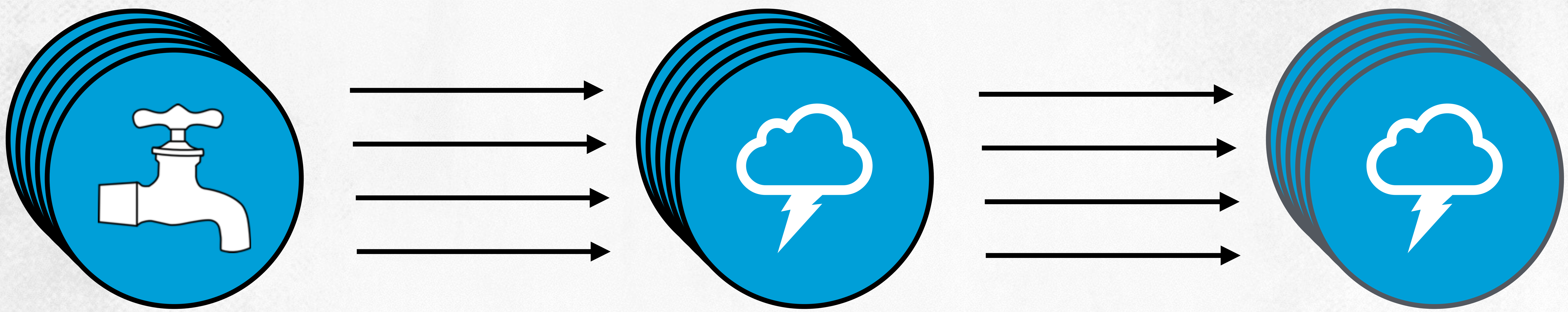
WORD COUNT TOPOLOGY

Live stream of Tweets



LOGICAL PLAN

WORD COUNT TOPOLOGY



TWEET SPOUT TASKS

PARSE TWEET BOLT TASKS

WORD COUNT BOLT TASKS

When a parse tweet bolt task emits a tuple which word count bolt task should it send to?

STREAM GROUPINGS

SHUFFLE GROUPING

Random distribution of tuples

FIELDS GROUPING

Group tuples by a field or multiple fields

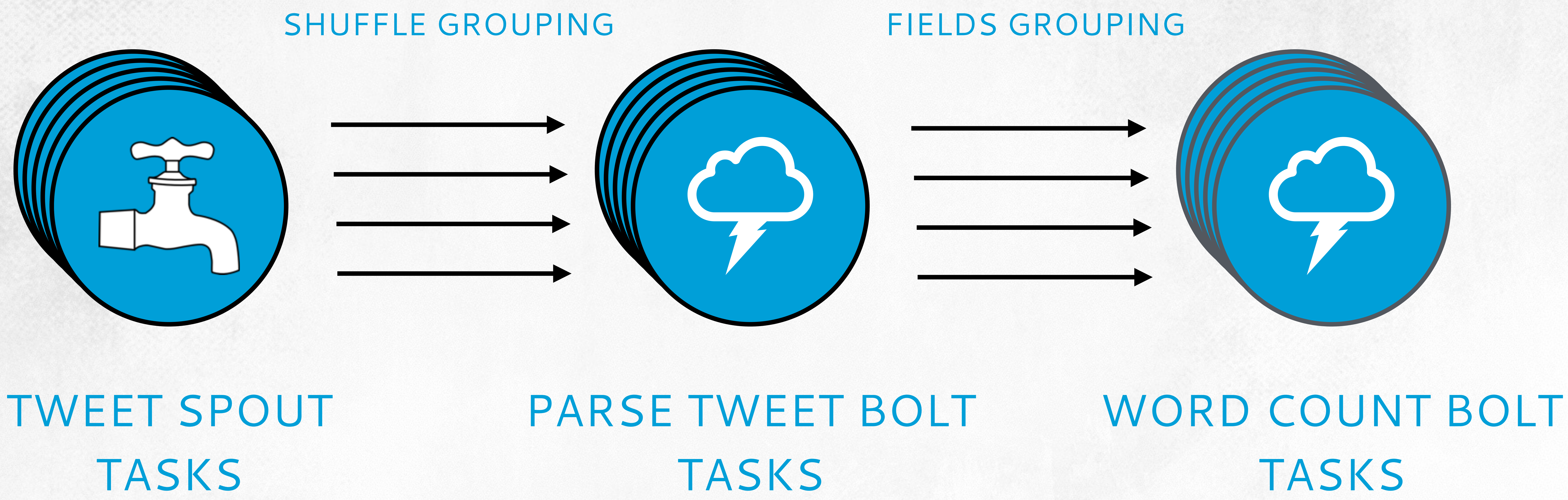
ALL GROUPING

Replicates tuples to all tasks

GLOBAL GROUPING

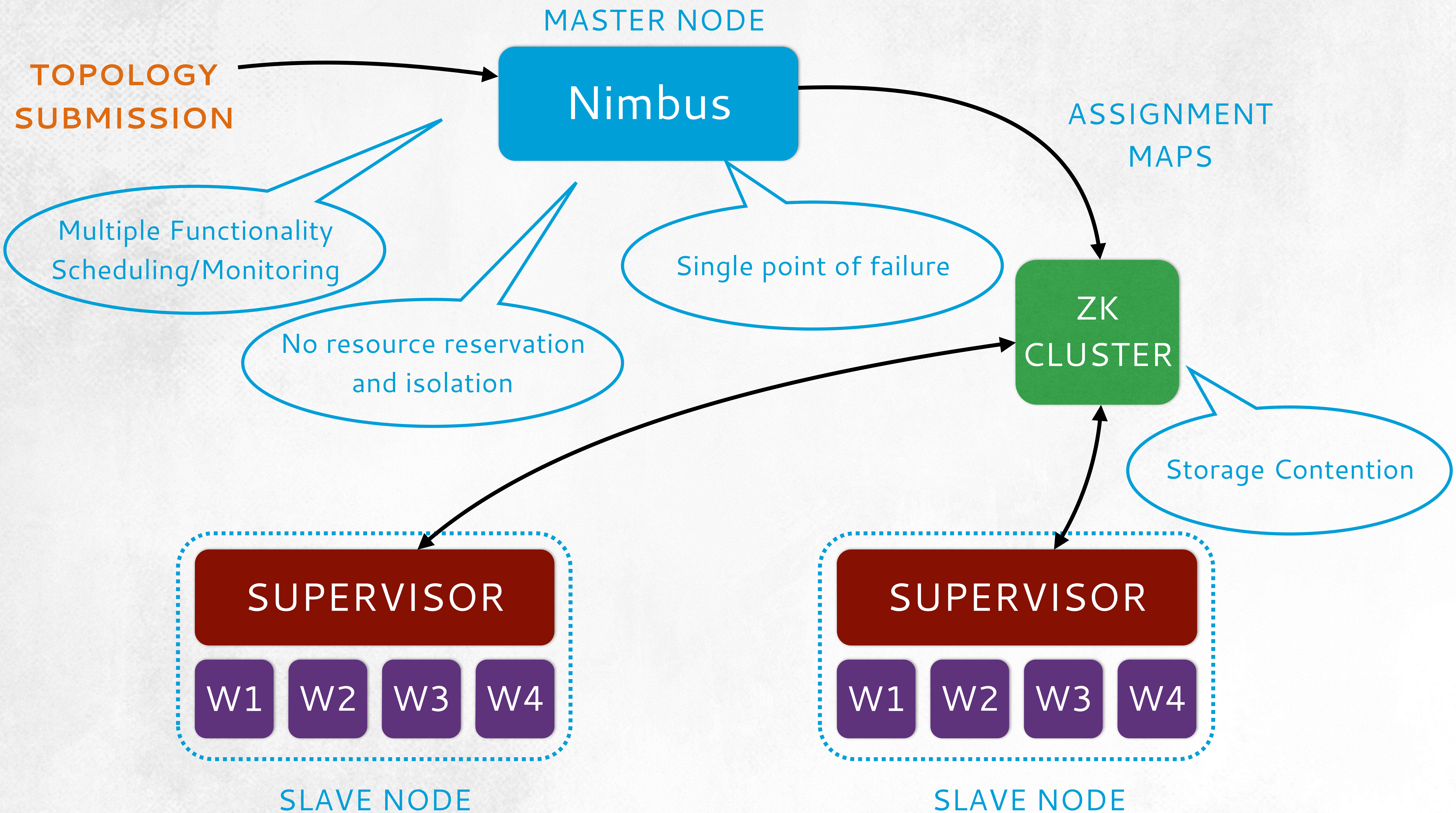
Sends the entire stream to one task

WORD COUNT TOPOLOGY

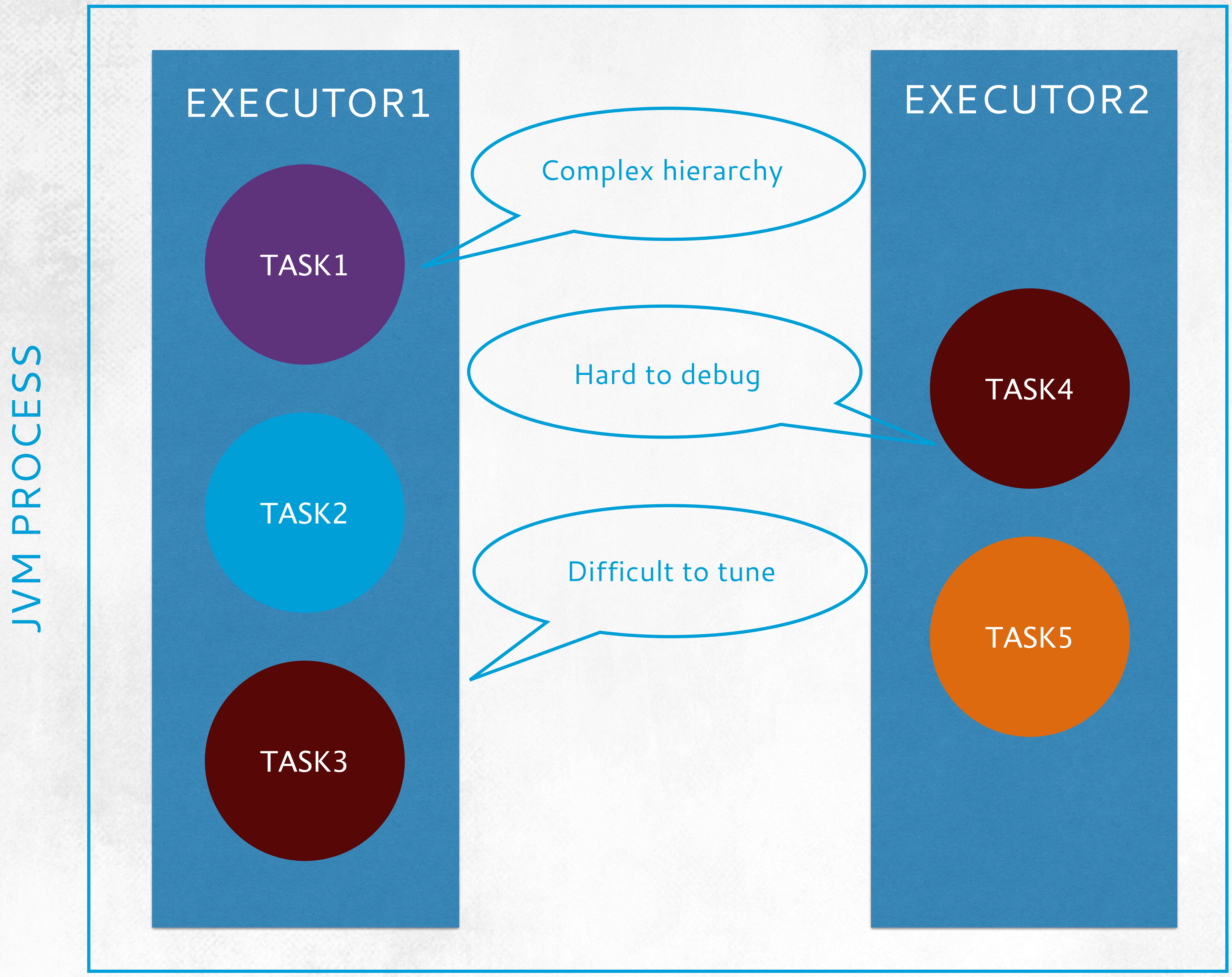


MOTIVATION

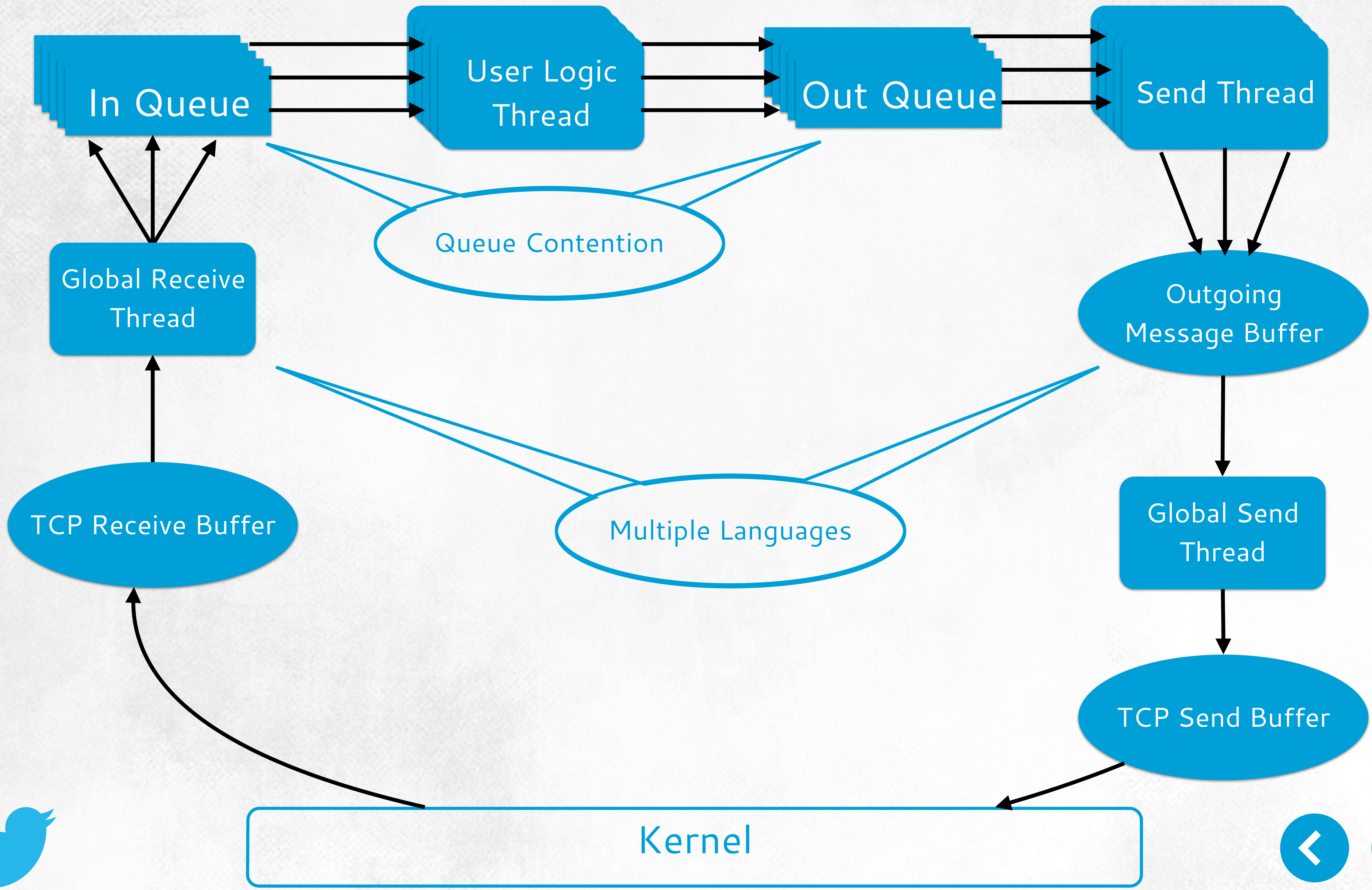
STORM ARCHITECTURE



STORM WORKER

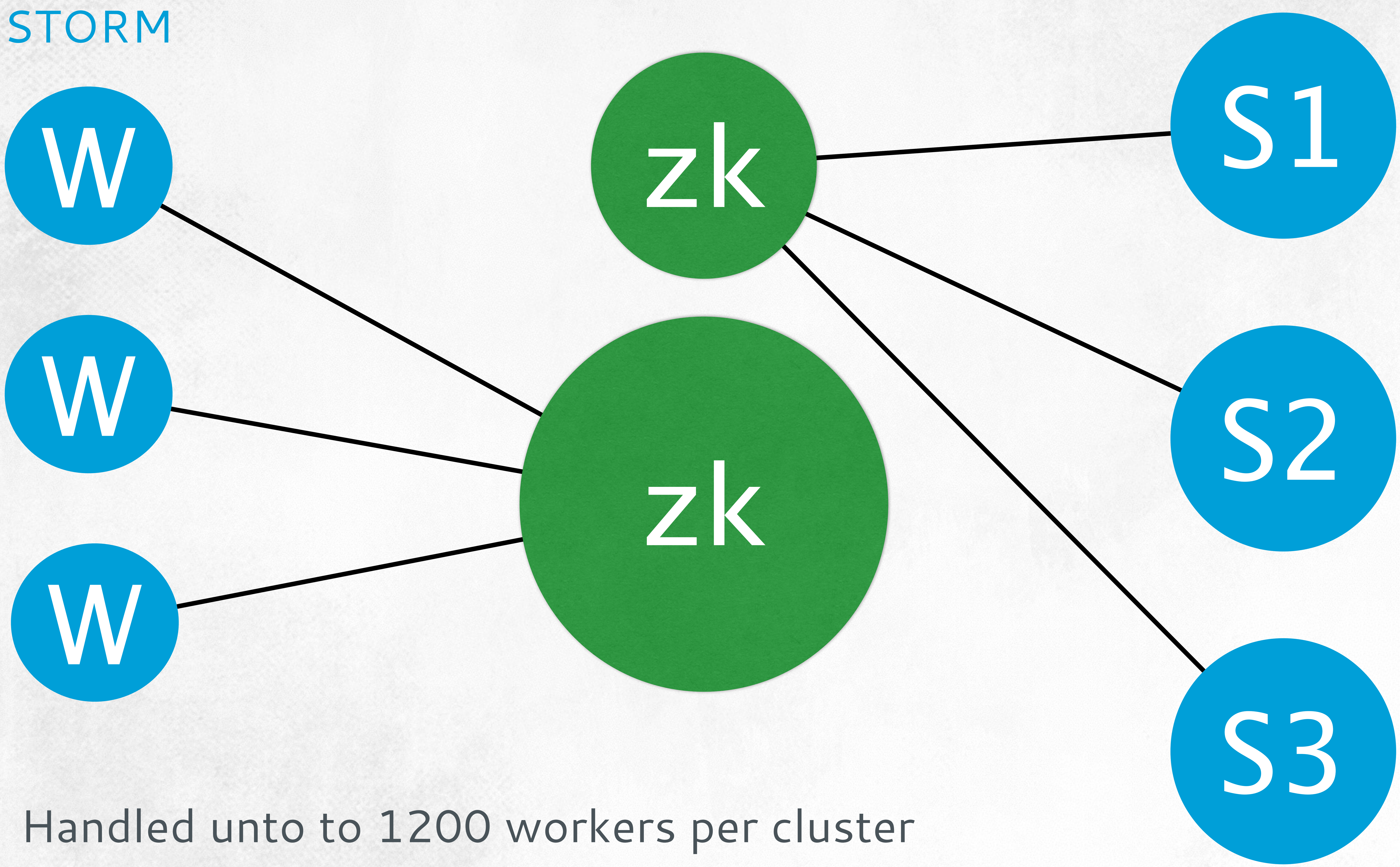


DATA FLOW IN STORM WORKERS



OVERLOADED ZOOKEEPER

Scaled up



Handled unto to 1200 workers per cluster

OVERLOADED ZOOKEEPER

Analyzing zookeeper traffic

67%

KAFKA SPOUT

Offset/partition is written every 2 secs

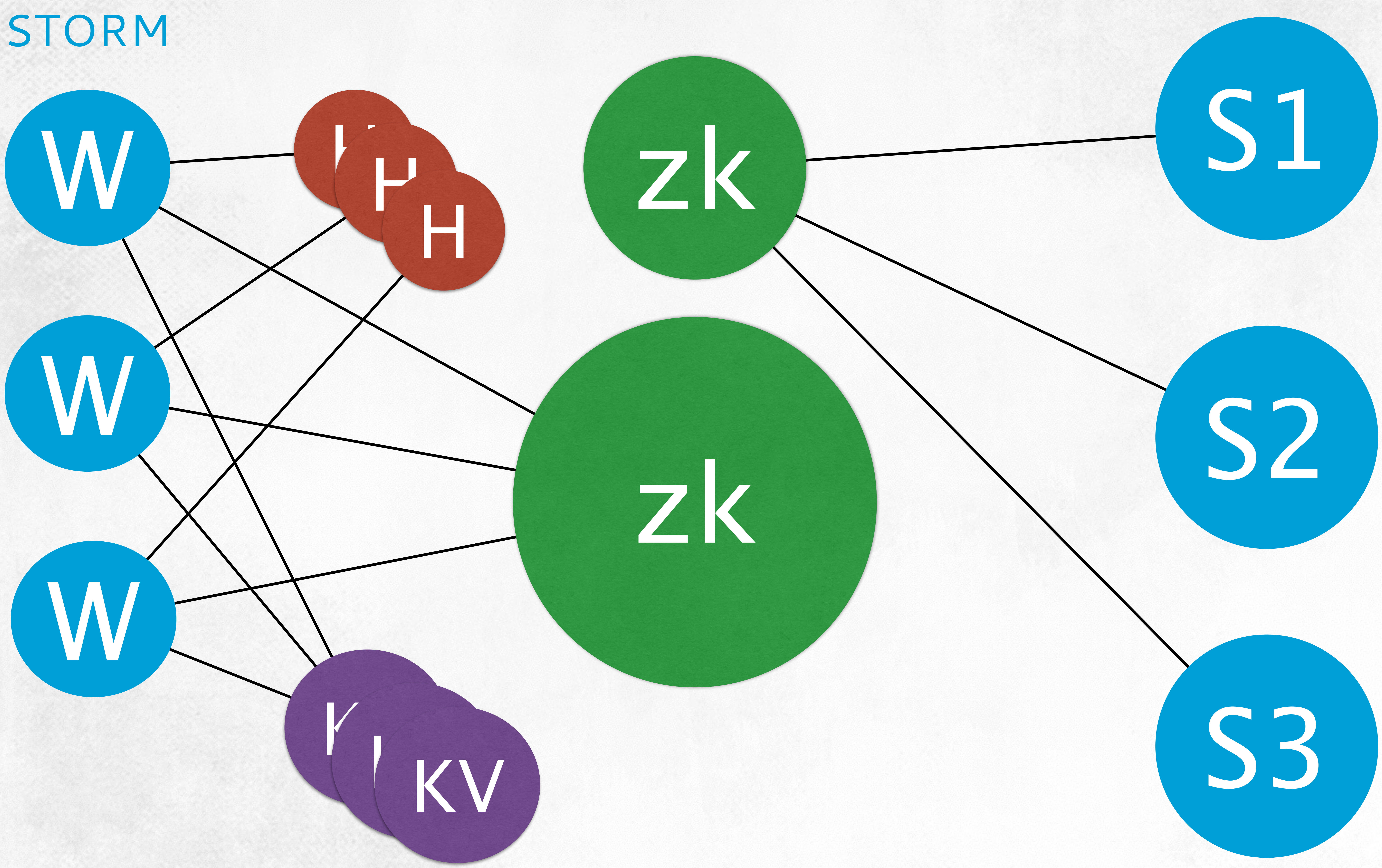
33%

STORM RUNTIME

Workers write heart beats every 3 secs

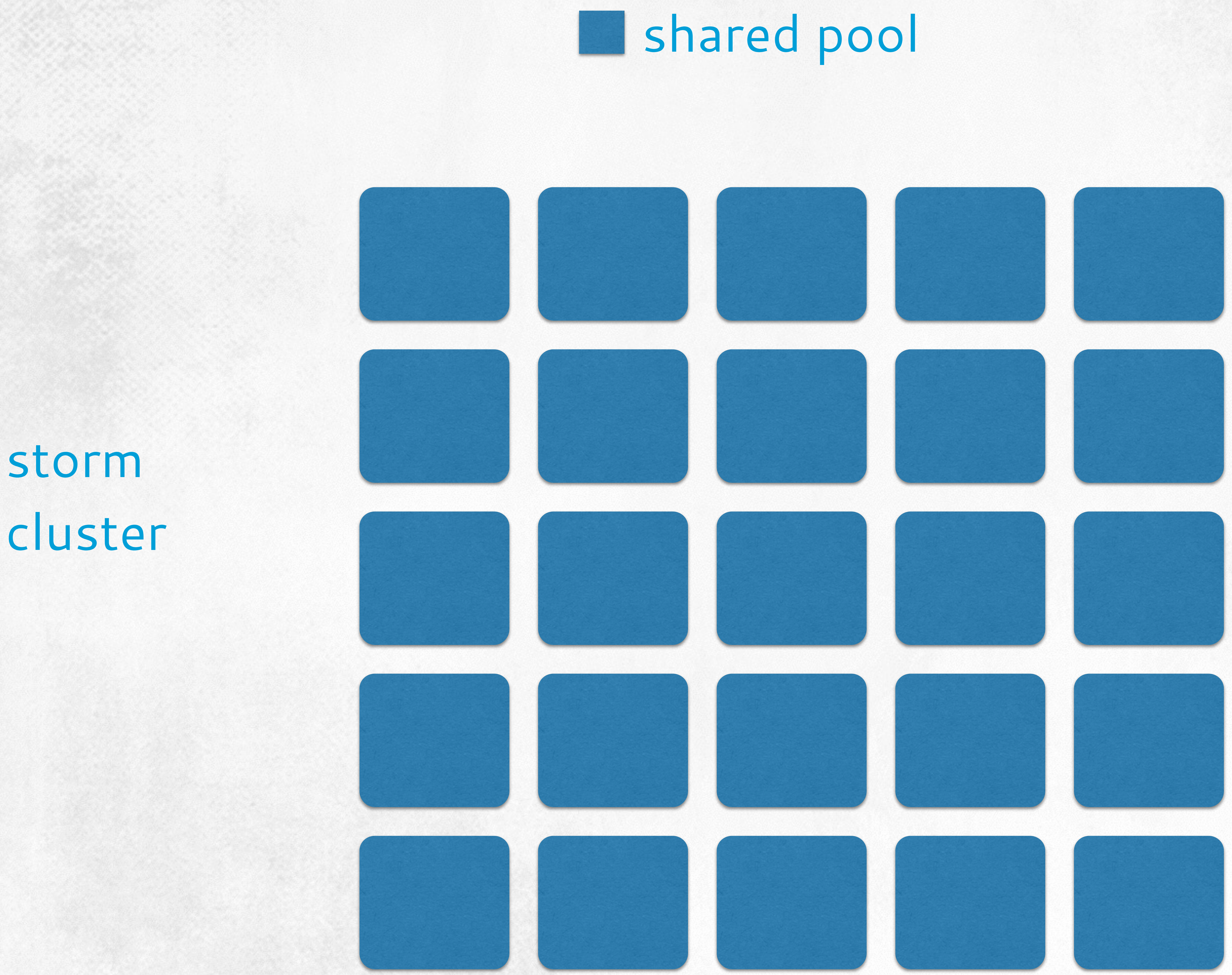
OVERLOADED ZOOKEEPER

Heart beat daemons

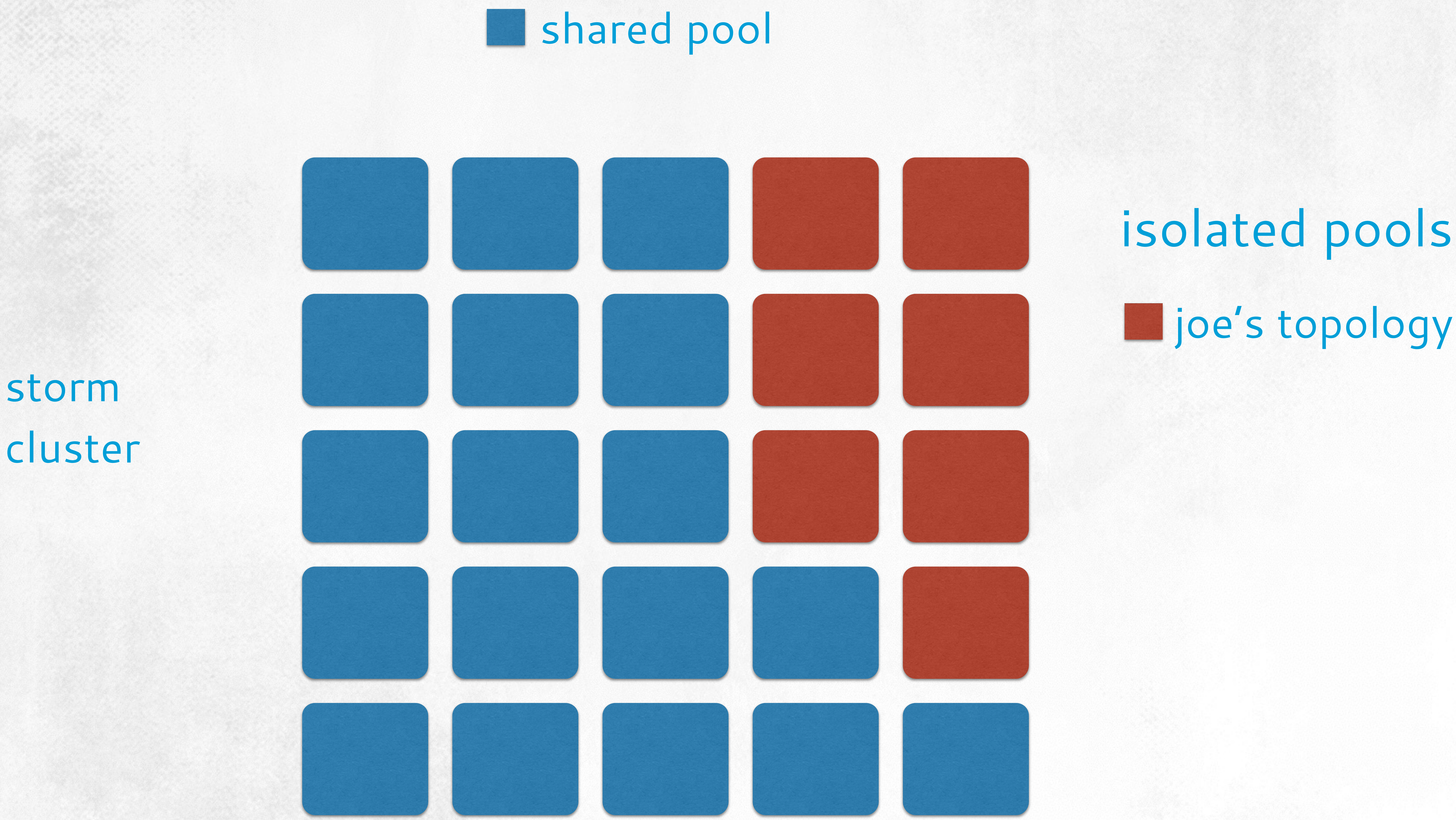


5000 workers per cluster

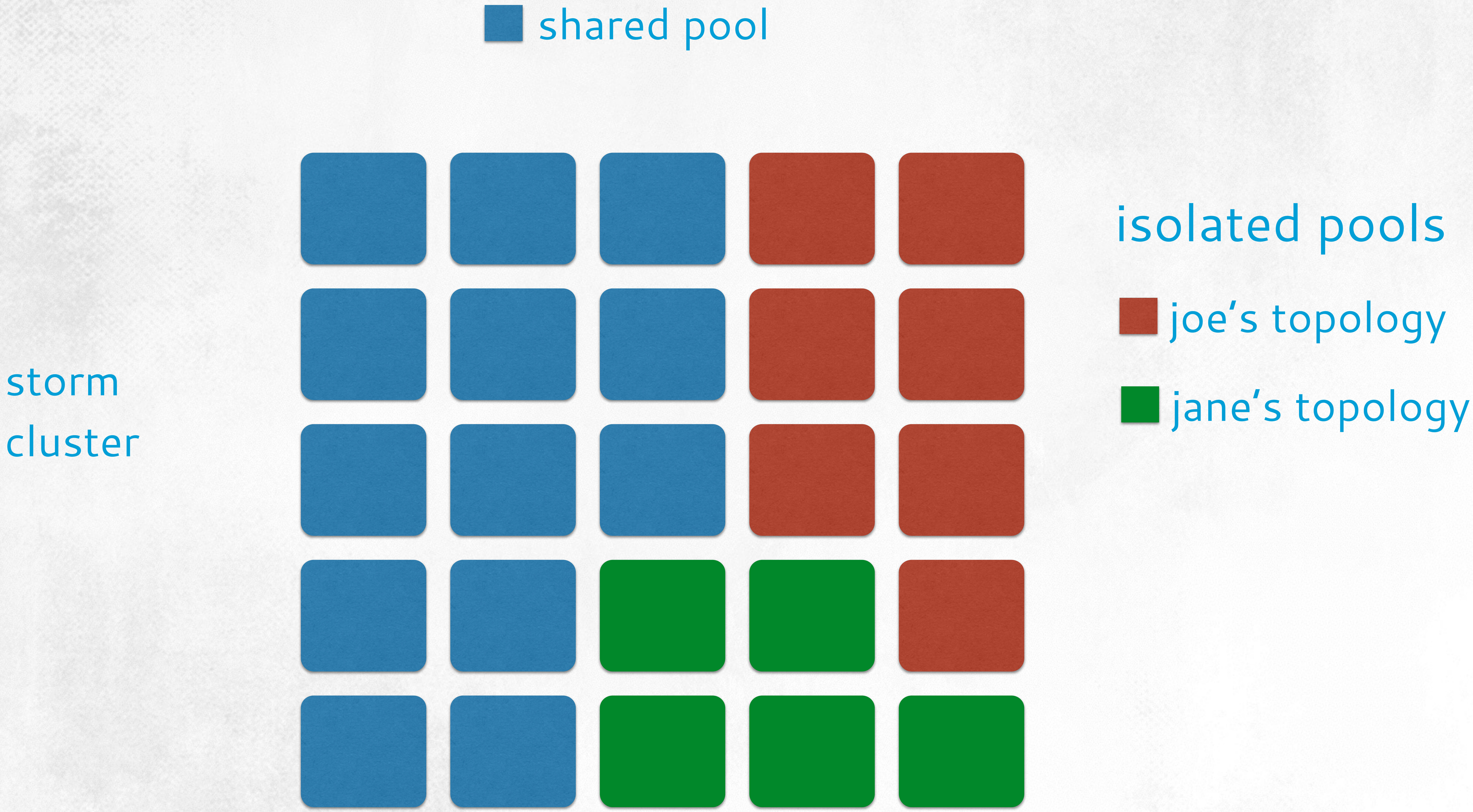
STORM – DEPLOYMENT



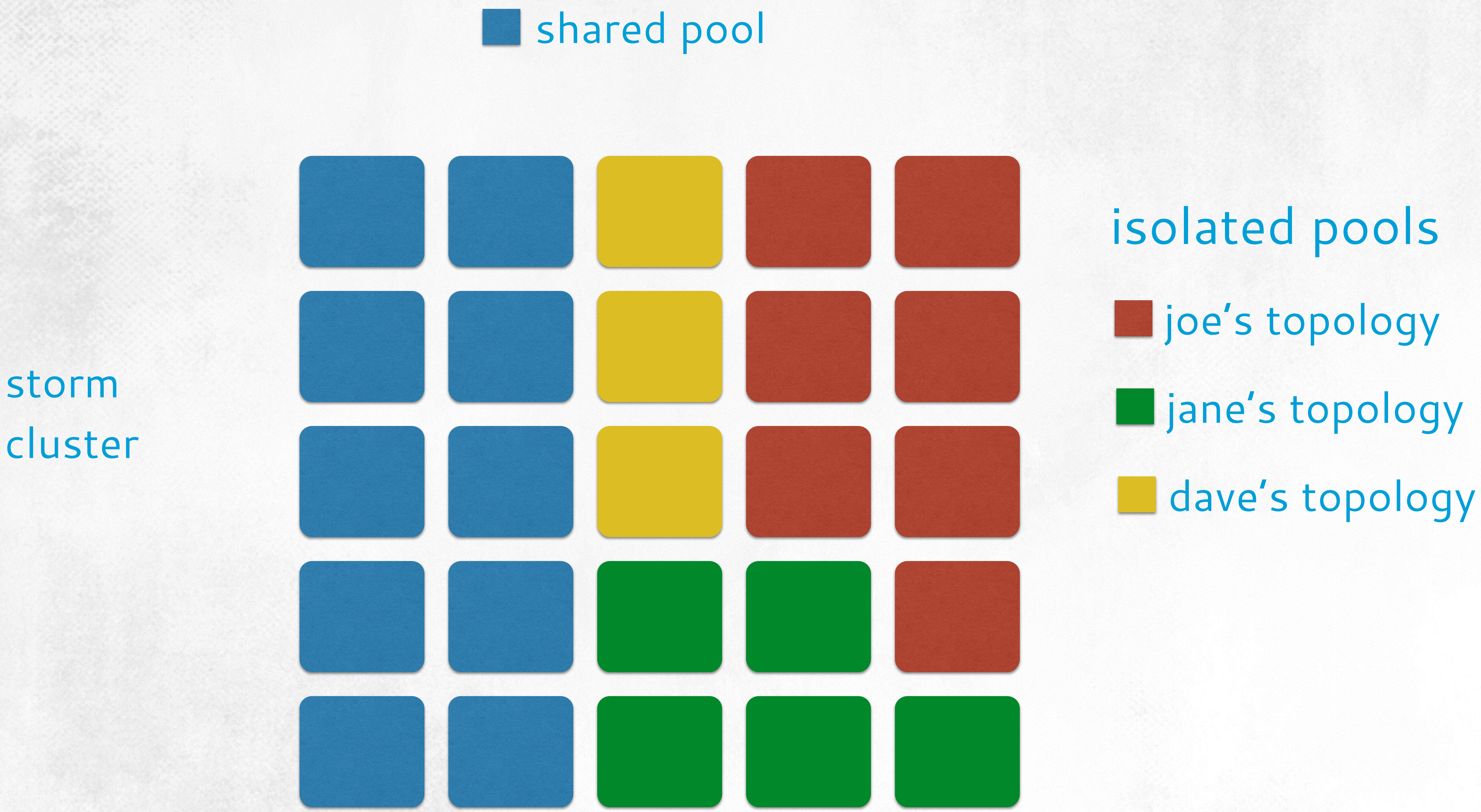
STORM – DEPLOYMENT



STORM – DEPLOYMENT



STORM – DEPLOYMENT



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

HERON DESIGN GOALS

FULLY API COMPATIBLE WITH STORM

Directed acyclic graph

Topologies, spouts and bolts

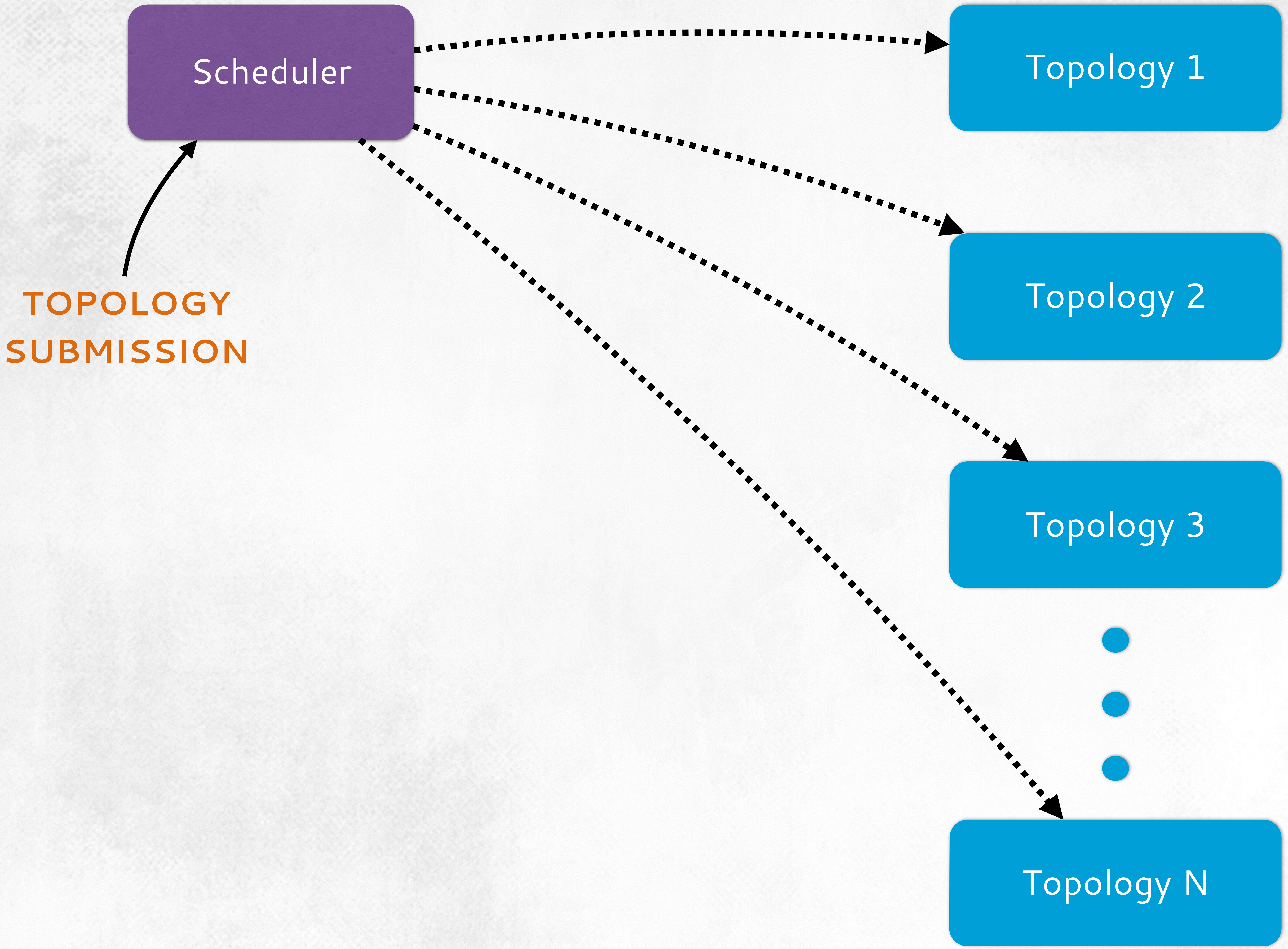
TASK ISOLATION

Ease of debug ability/resource isolation/profiling

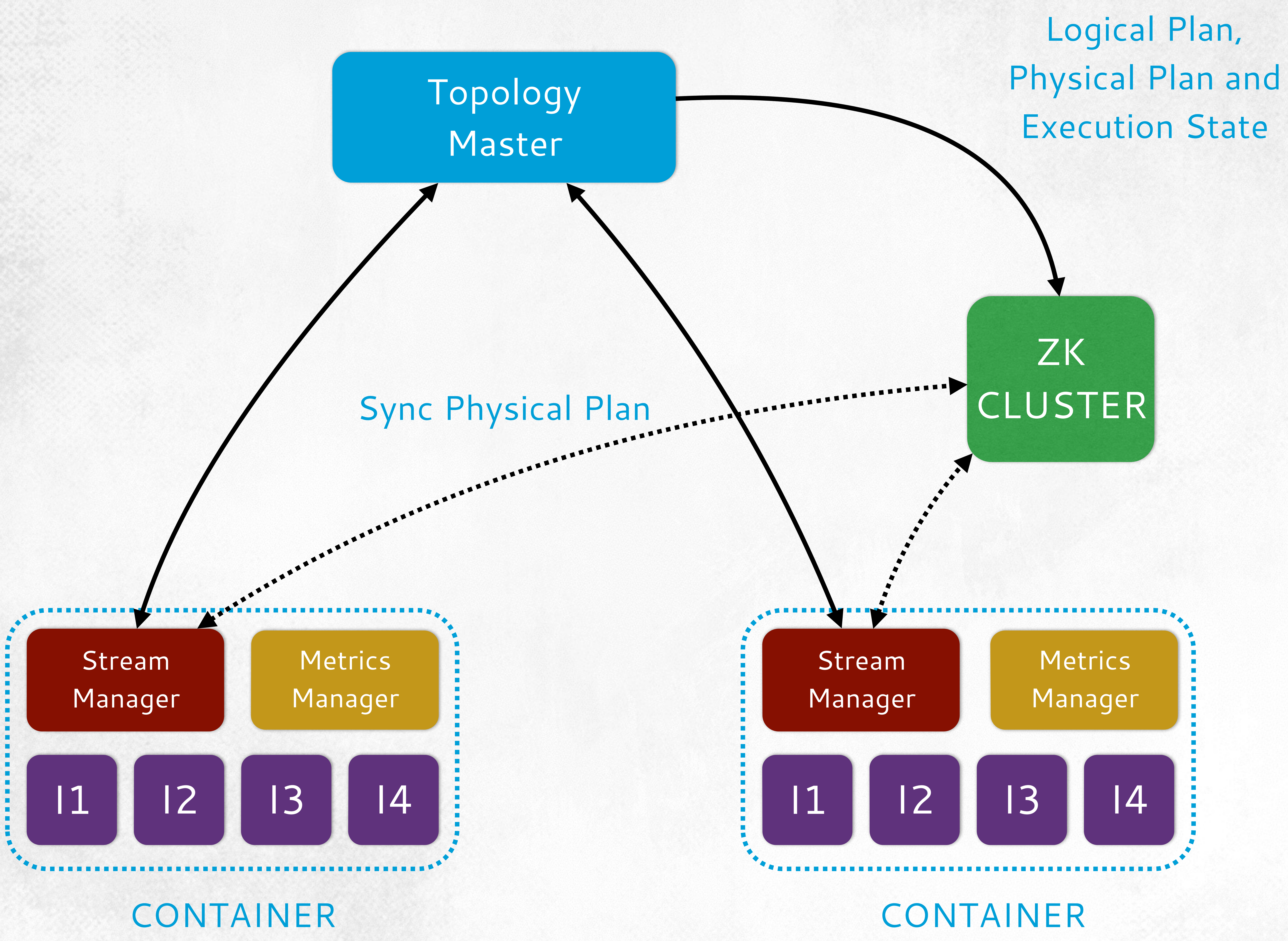
USE OF MAIN STREAM LANGUAGES

C++/JAVA/Python

HERON ARCHITECTURE



TOPOLOGY ARCHITECTURE



TOPOLOGY MASTER

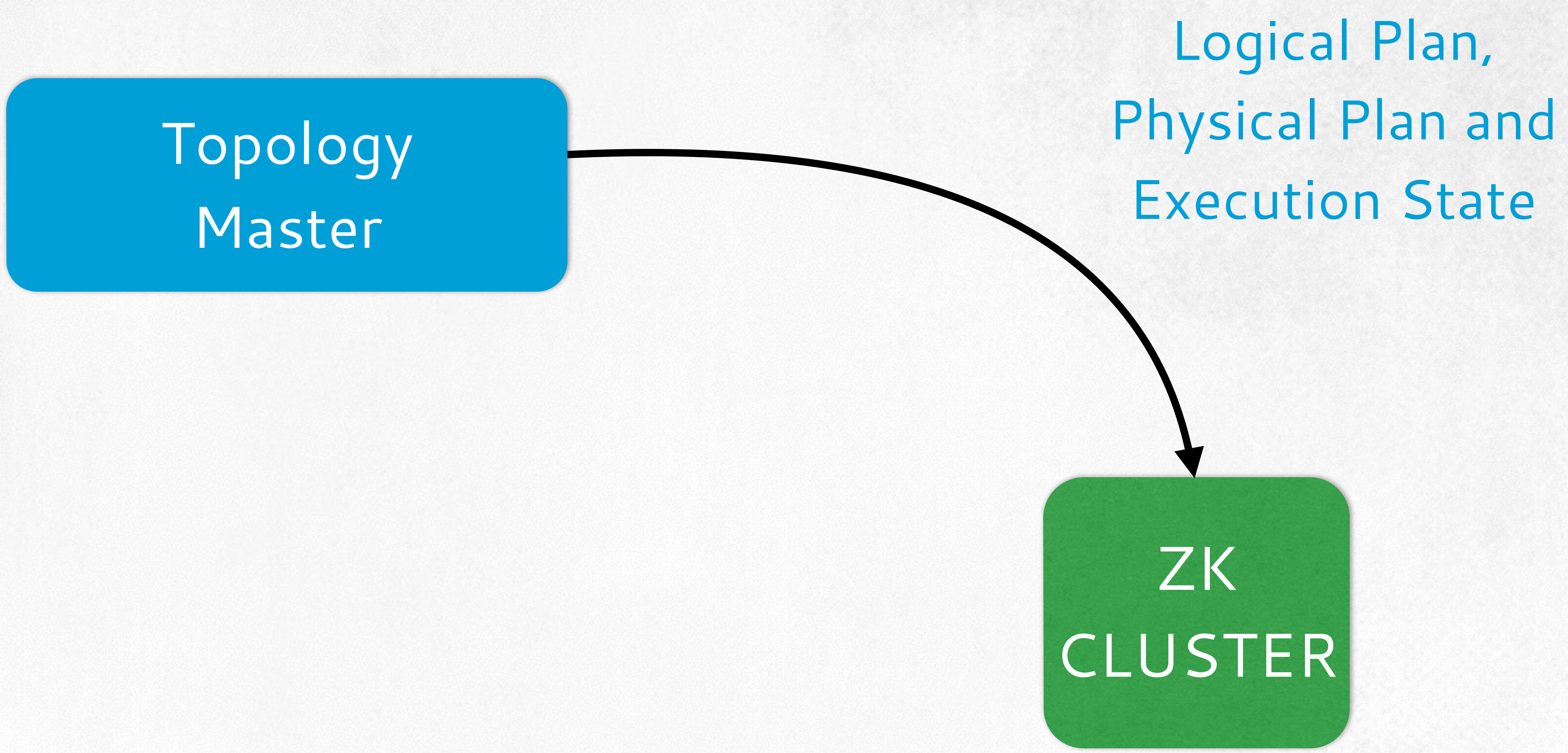
Solely responsible for the entire topology

ASSIGNS ROLE

MONITORING

METRICS

TOPOLOGY MASTER



- PREVENT MULTIPLE TM BECOMING MASTERS
- ALLOWS OTHER PROCESS TO DISCOVER TM

STREAM MANAGER

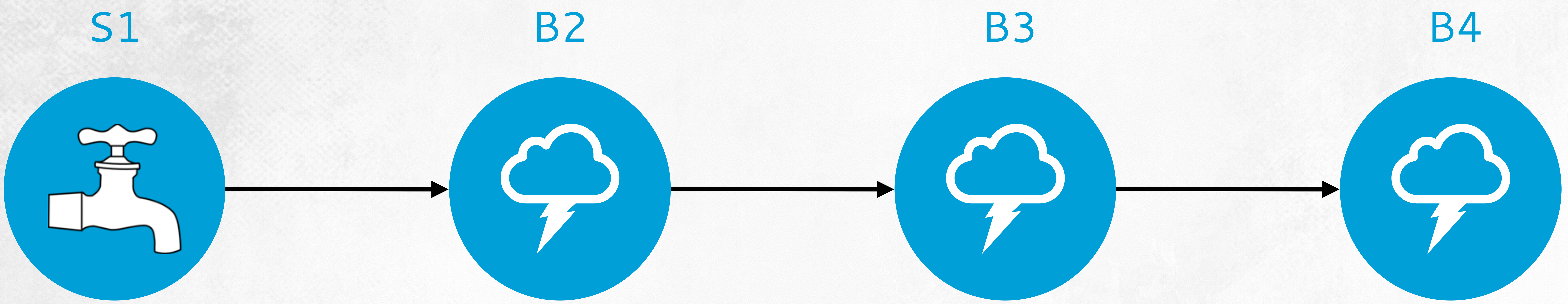
Routing Engine

ROUTES TUPLES

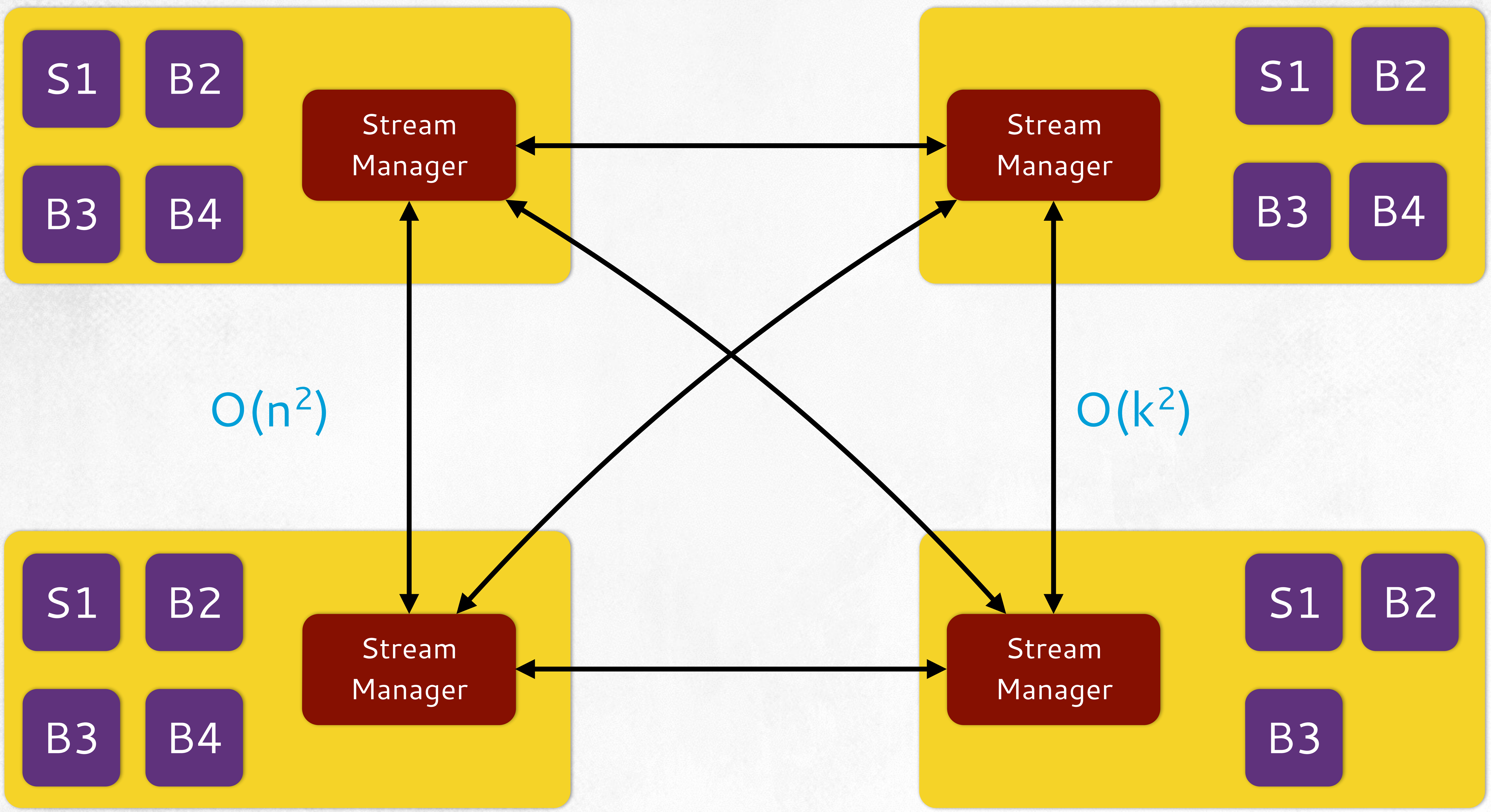
BACK PRESSURE

ACK MGMT

STREAM MANAGER

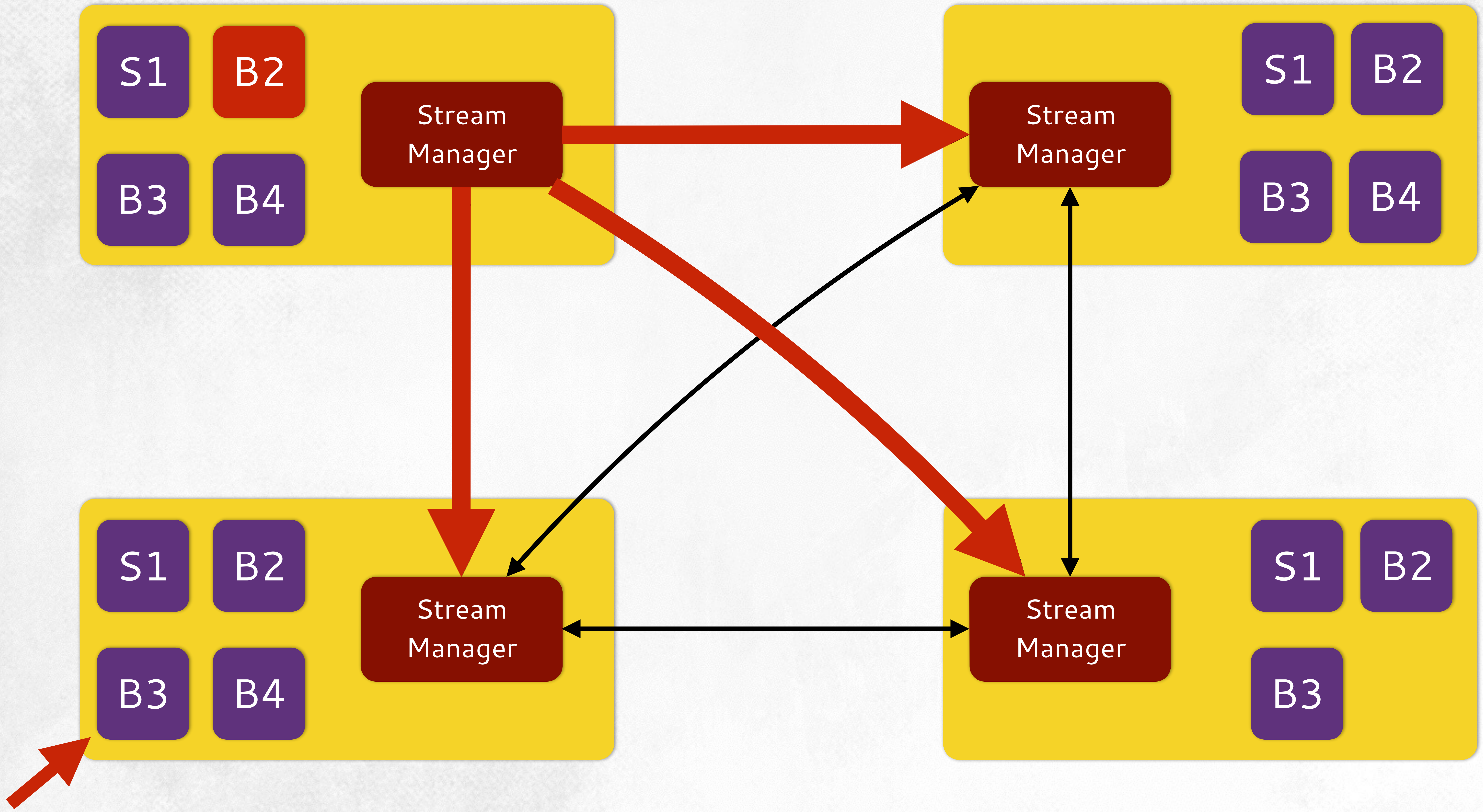


STREAM MANAGER



STREAM MANAGER

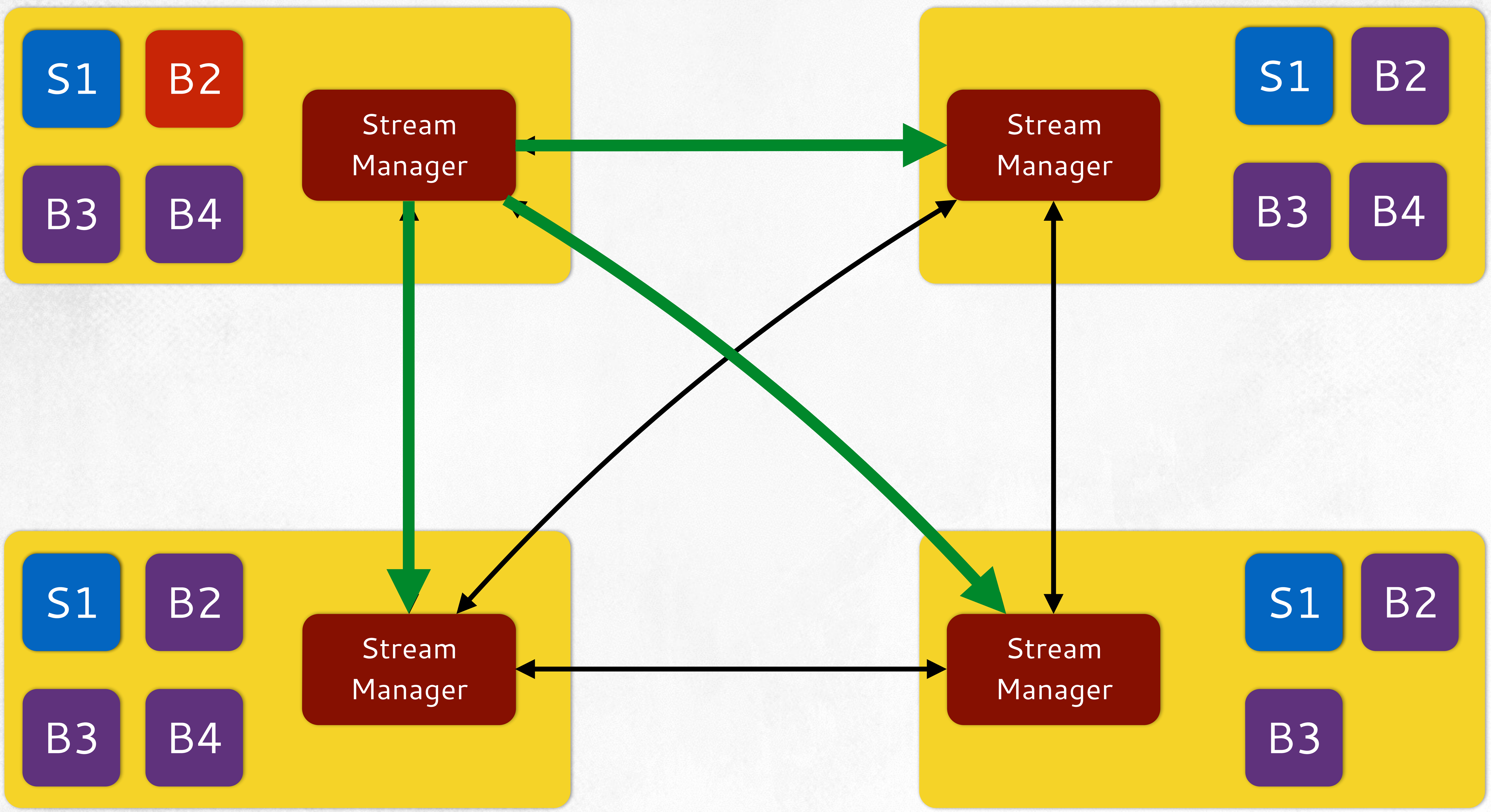
tcp back pressure



SLOWS UPSTREAM AND DOWNSTREAM INSTANCES

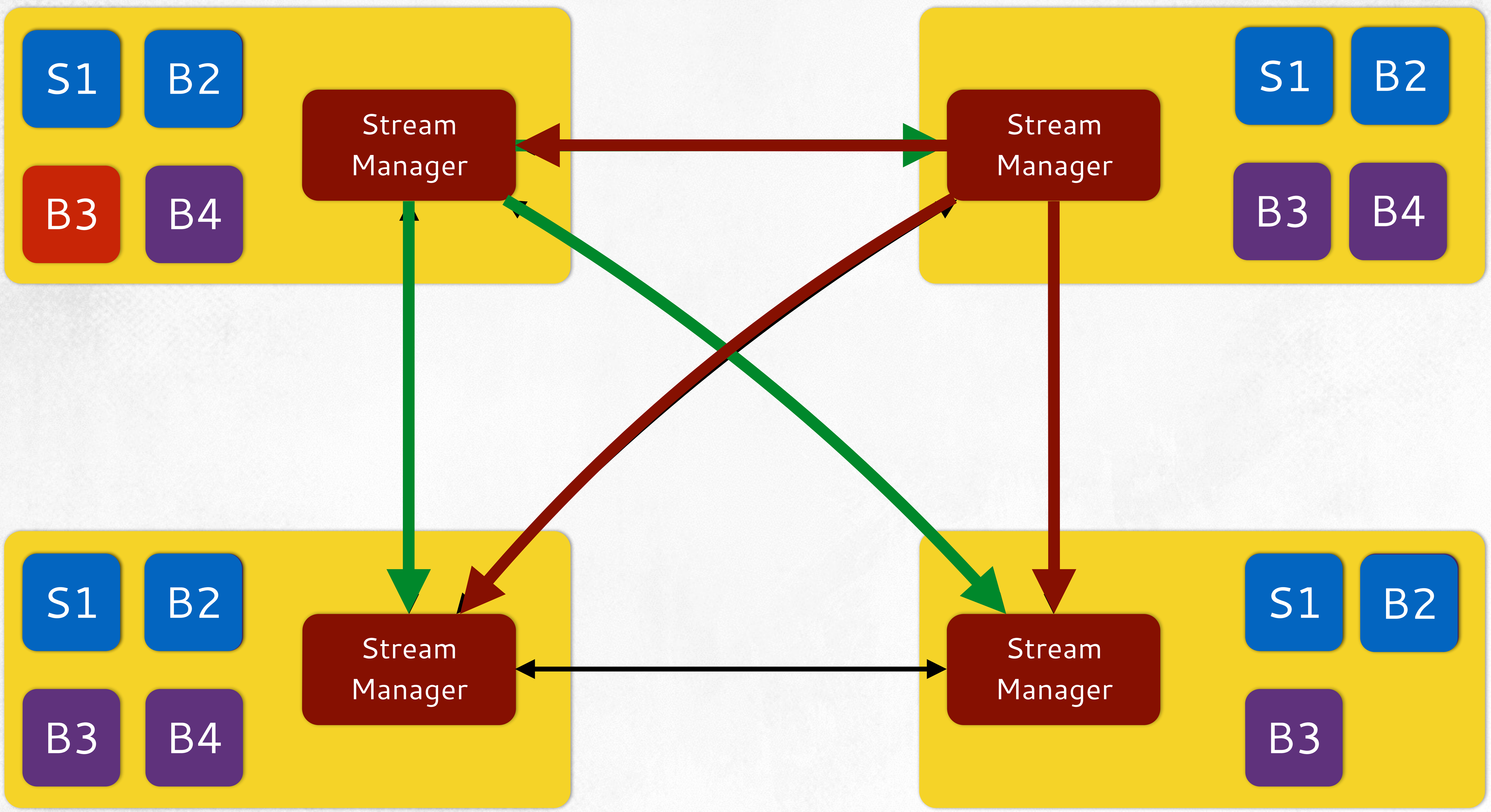
STREAM MANAGER

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

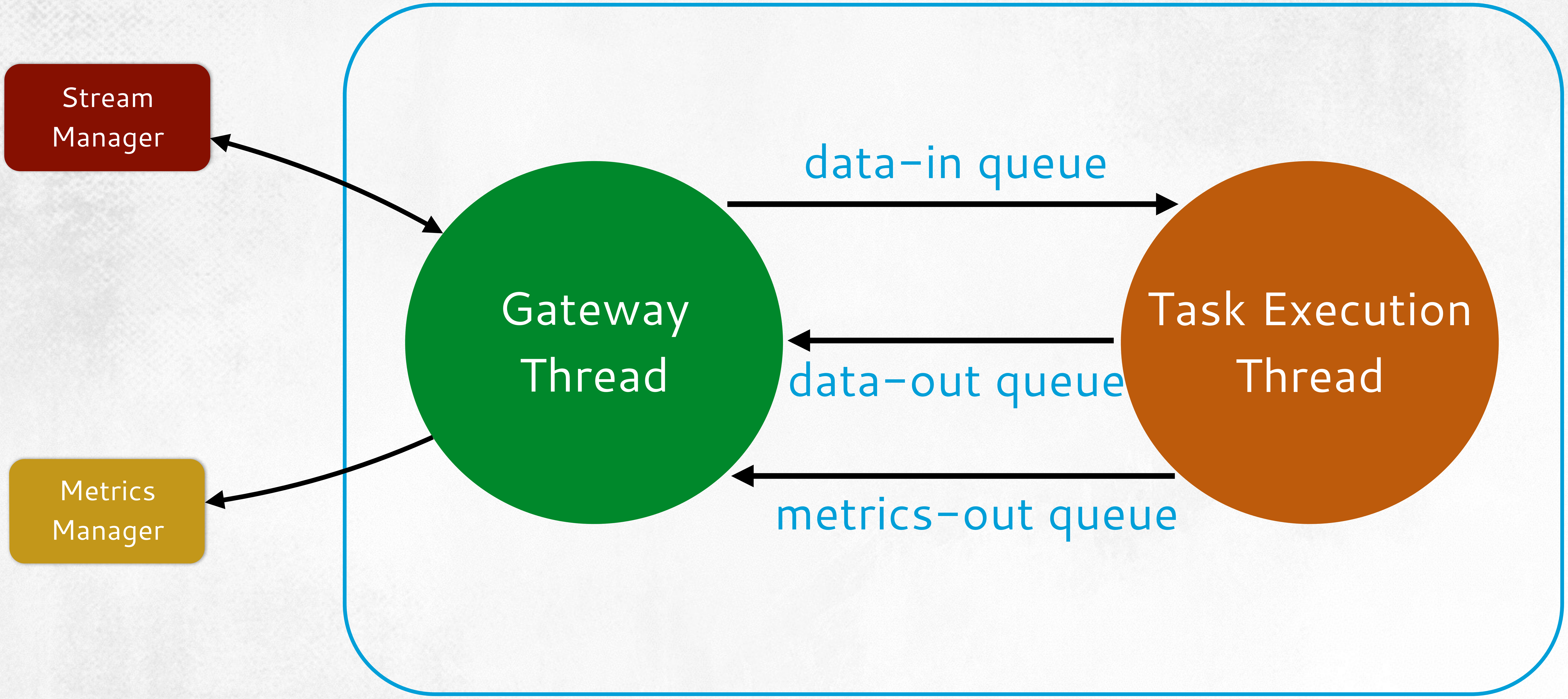
Does the real work!

RUNS ONE TASK

EXPOSES API

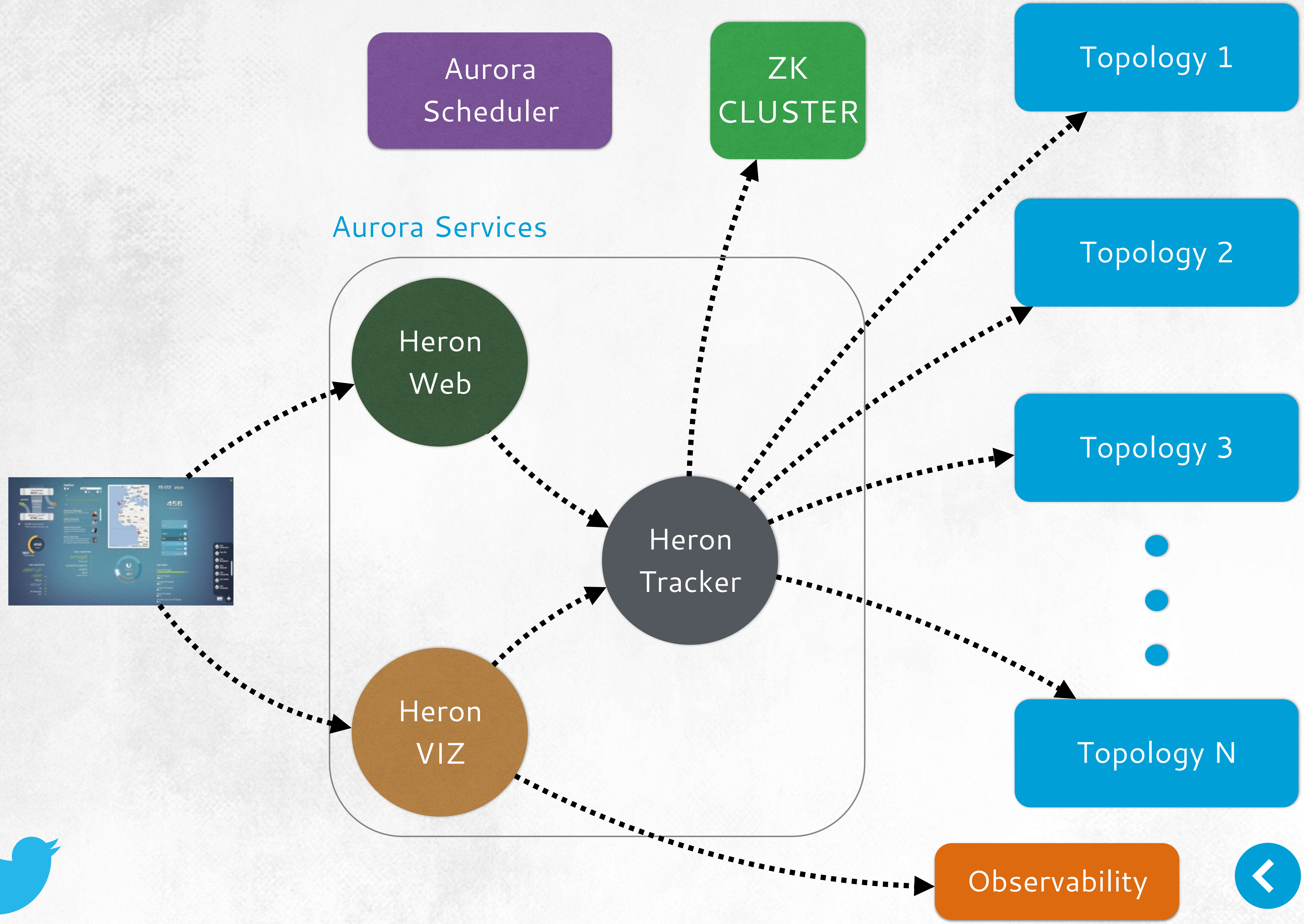
COLLECTS METRICS

HERON INSTANCE

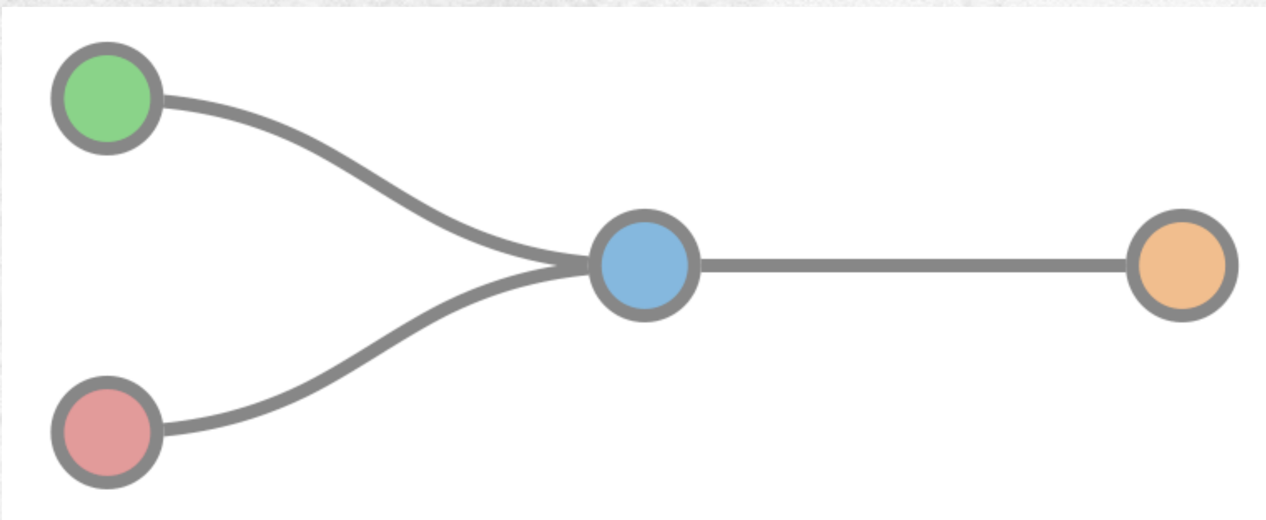


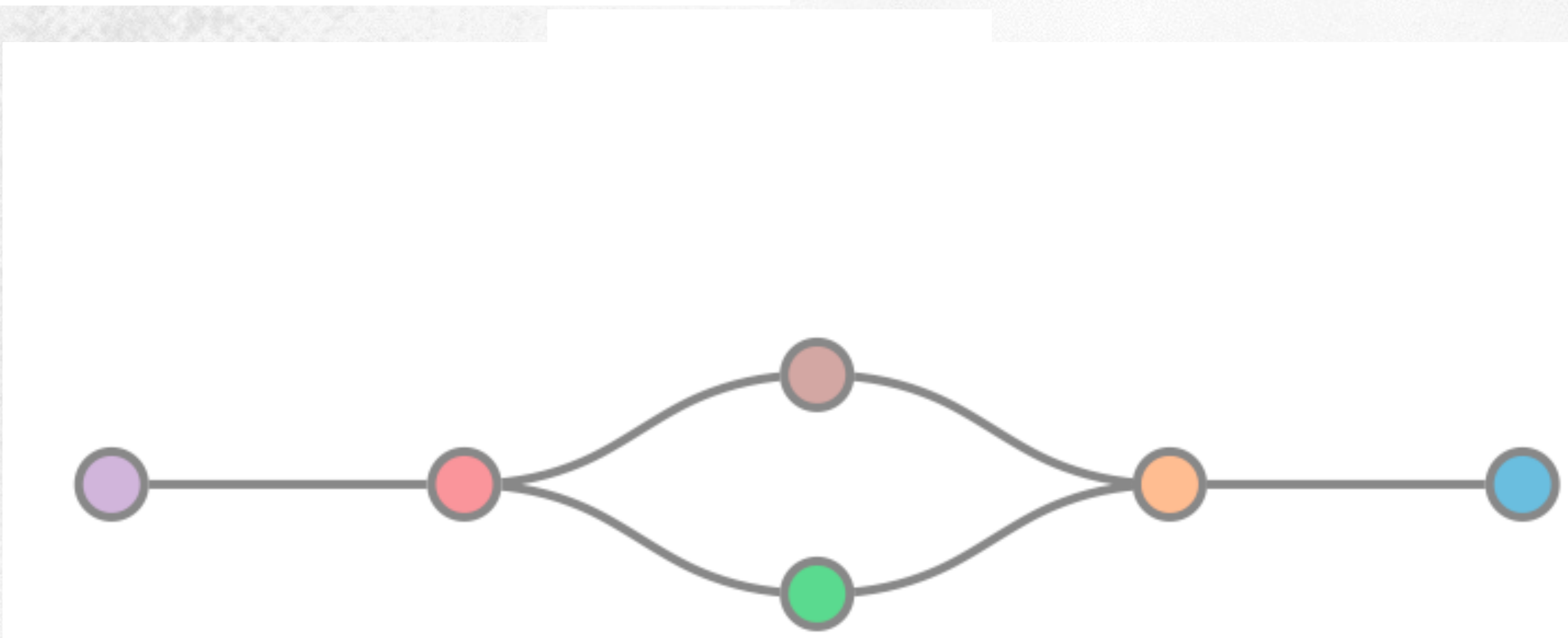
OPERATIONAL EXPERIENCES

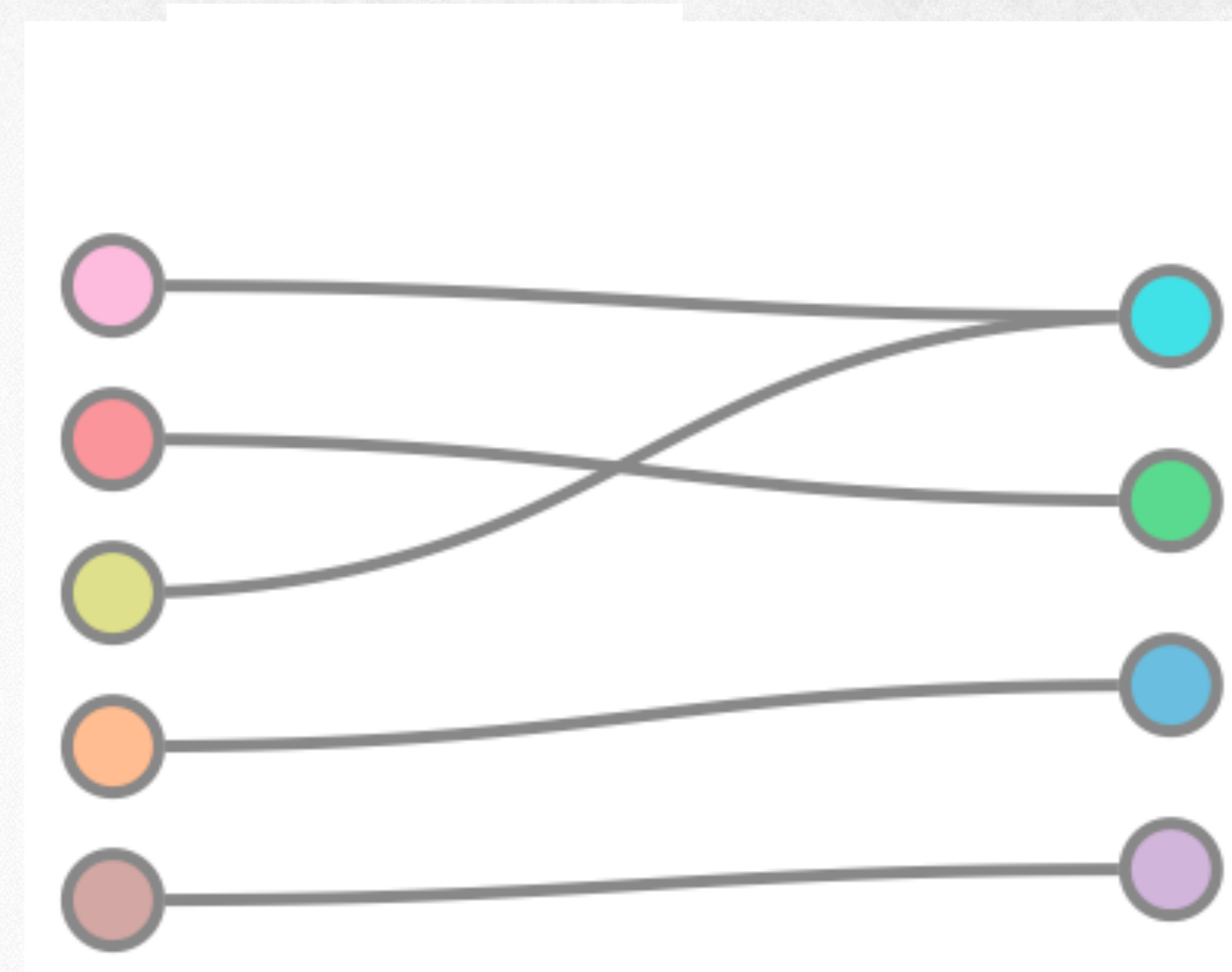
HERON DEPLOYMENT

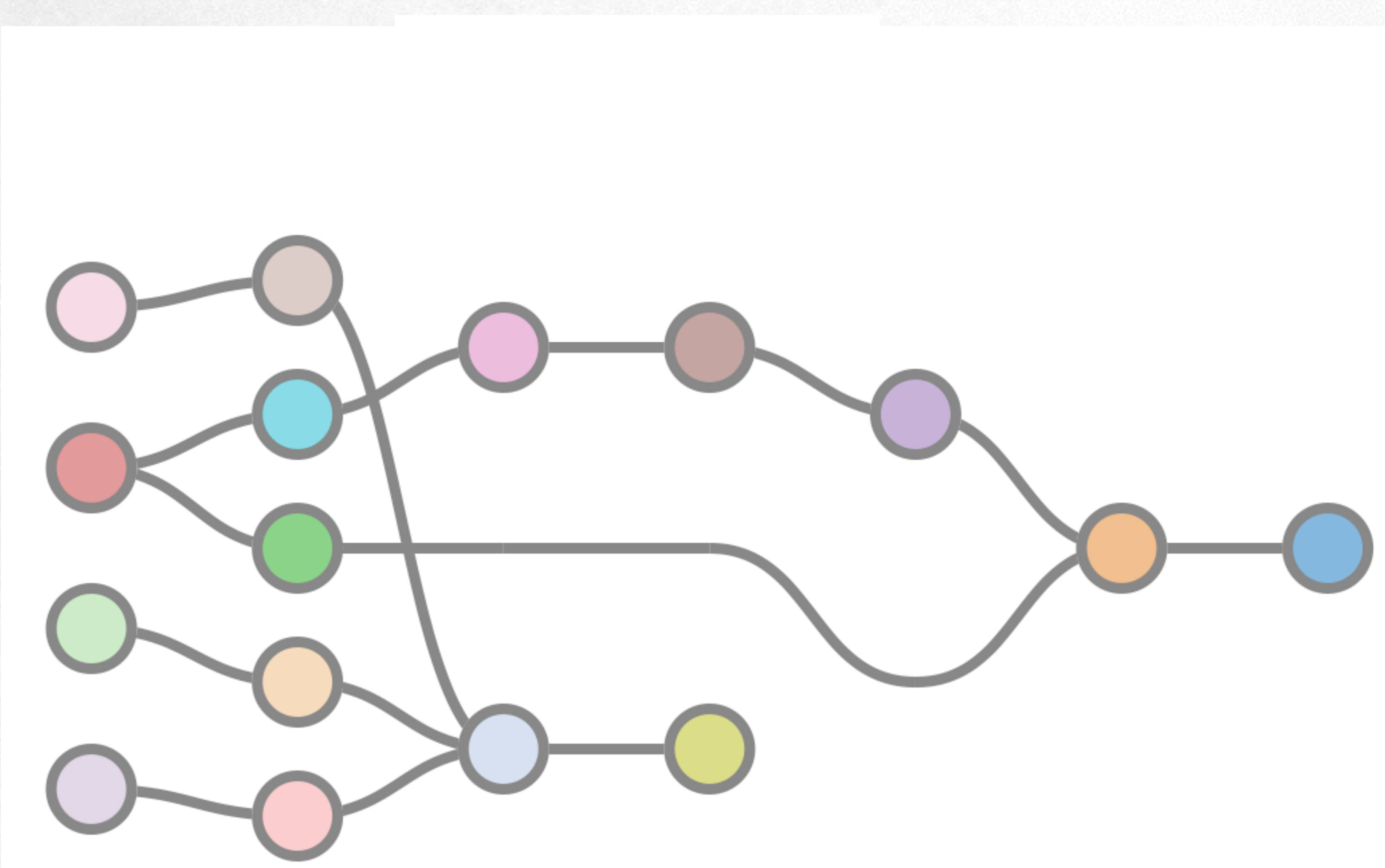


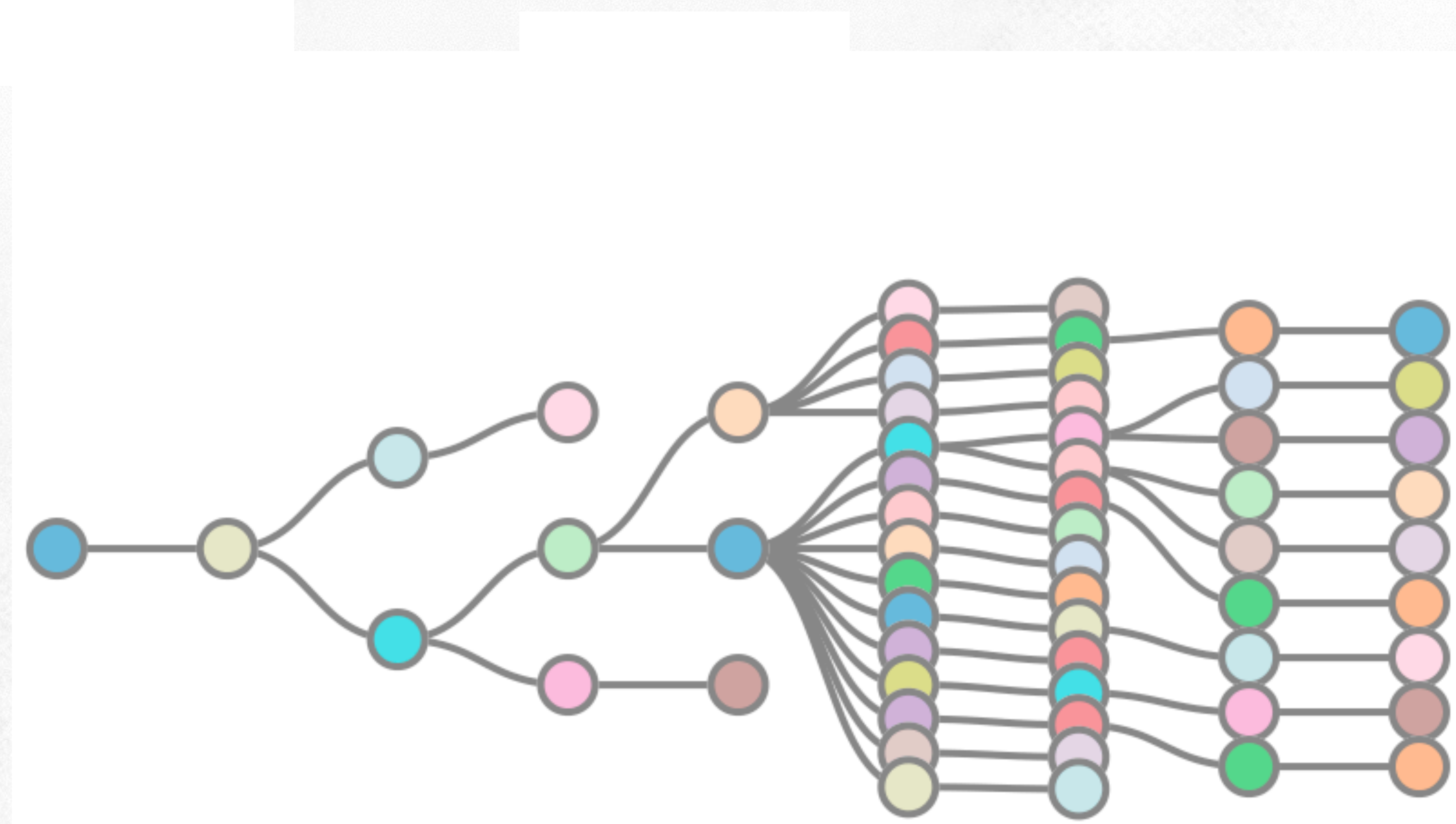
HERON SAMPLE TOPOLOGIES











SAMPLE TOPOLOGY DASHBOARD

HERON @TWITTER

STORM is decommissioned

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

HERON PERFORMANCE

HERON PERFORMANCE

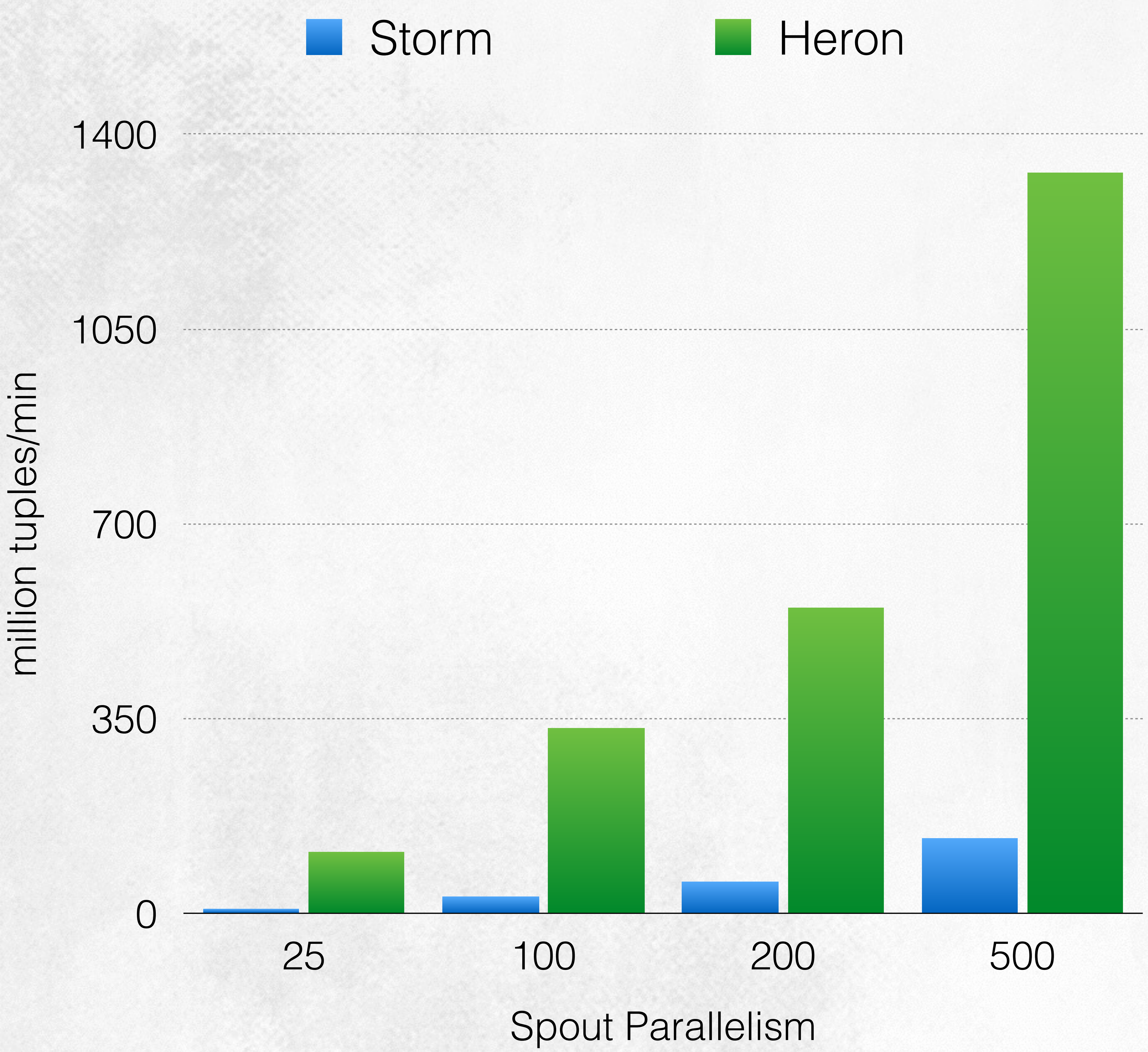
Settings

COMPONENTS	EXPT #1	EXPT #2	EXPT #3	EXPT #4
Spout	25	100	200	300
Bolt	25	100	200	300
# Heron containers	25	100	200	300
# Storm workers	25	100	200	300

HERON PERFORMANCE

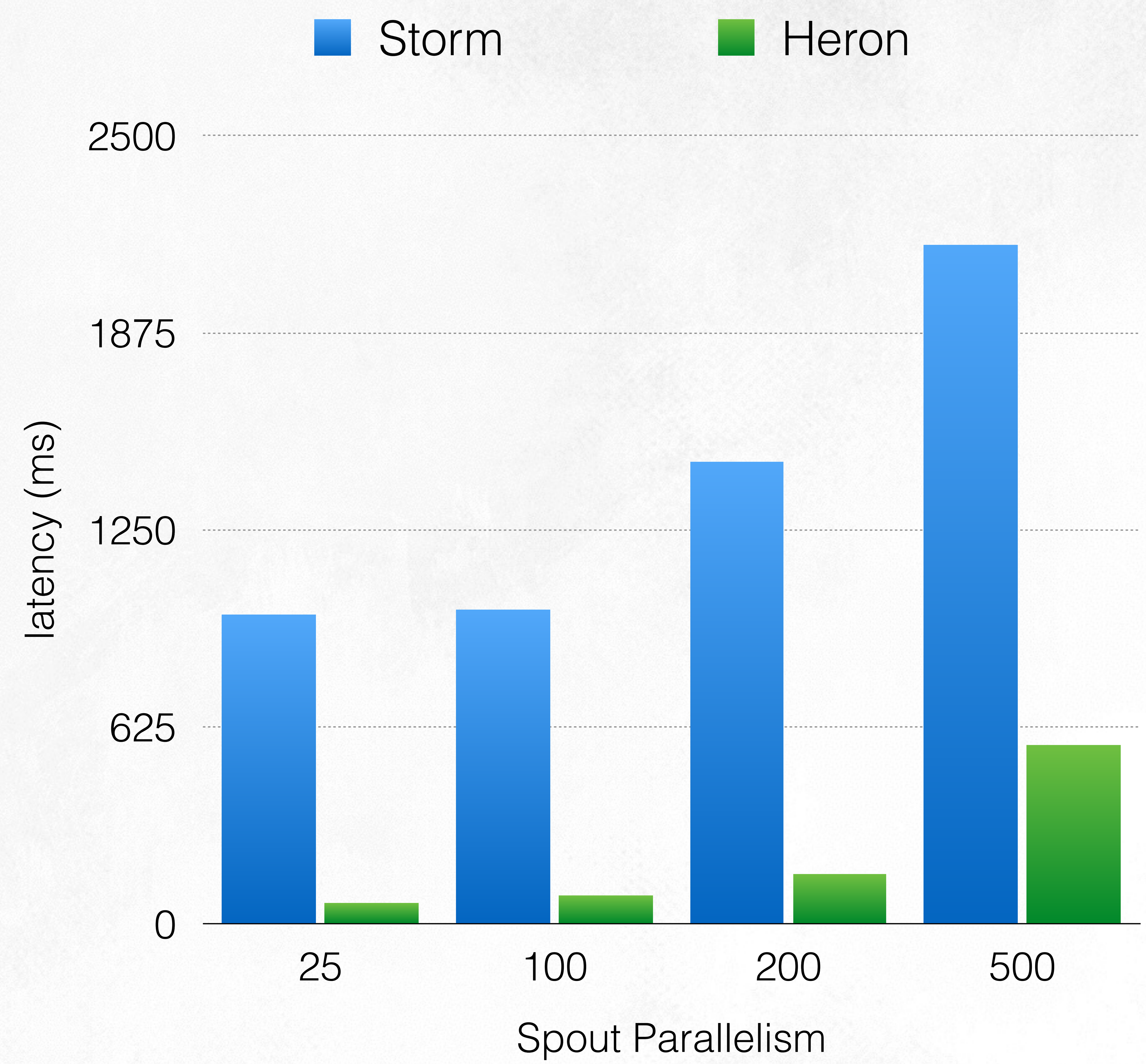
Word count topology - Acknowledgements enabled

Throughput



10-14x

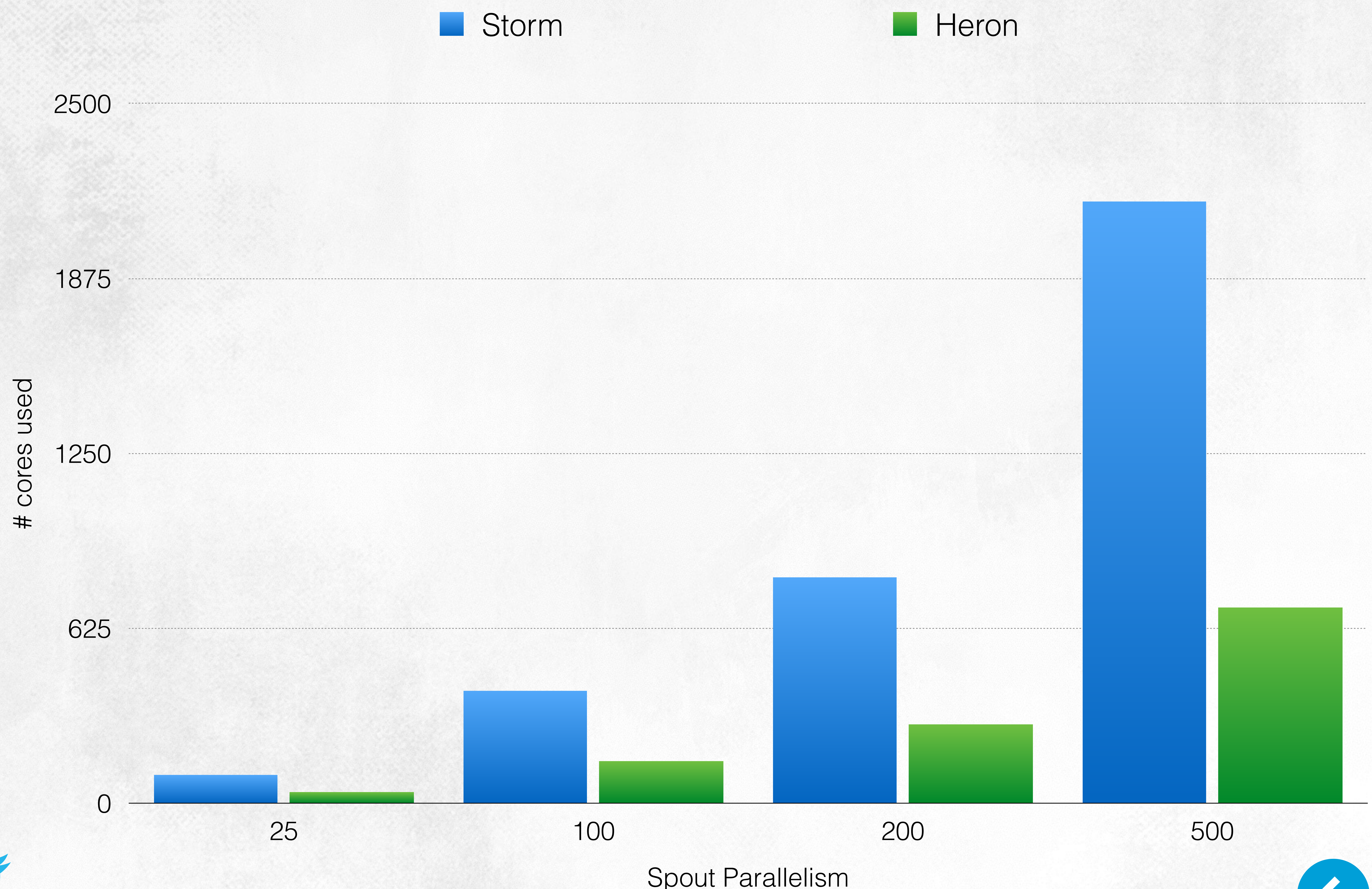
Latency



5-15x

HERON PERFORMANCE

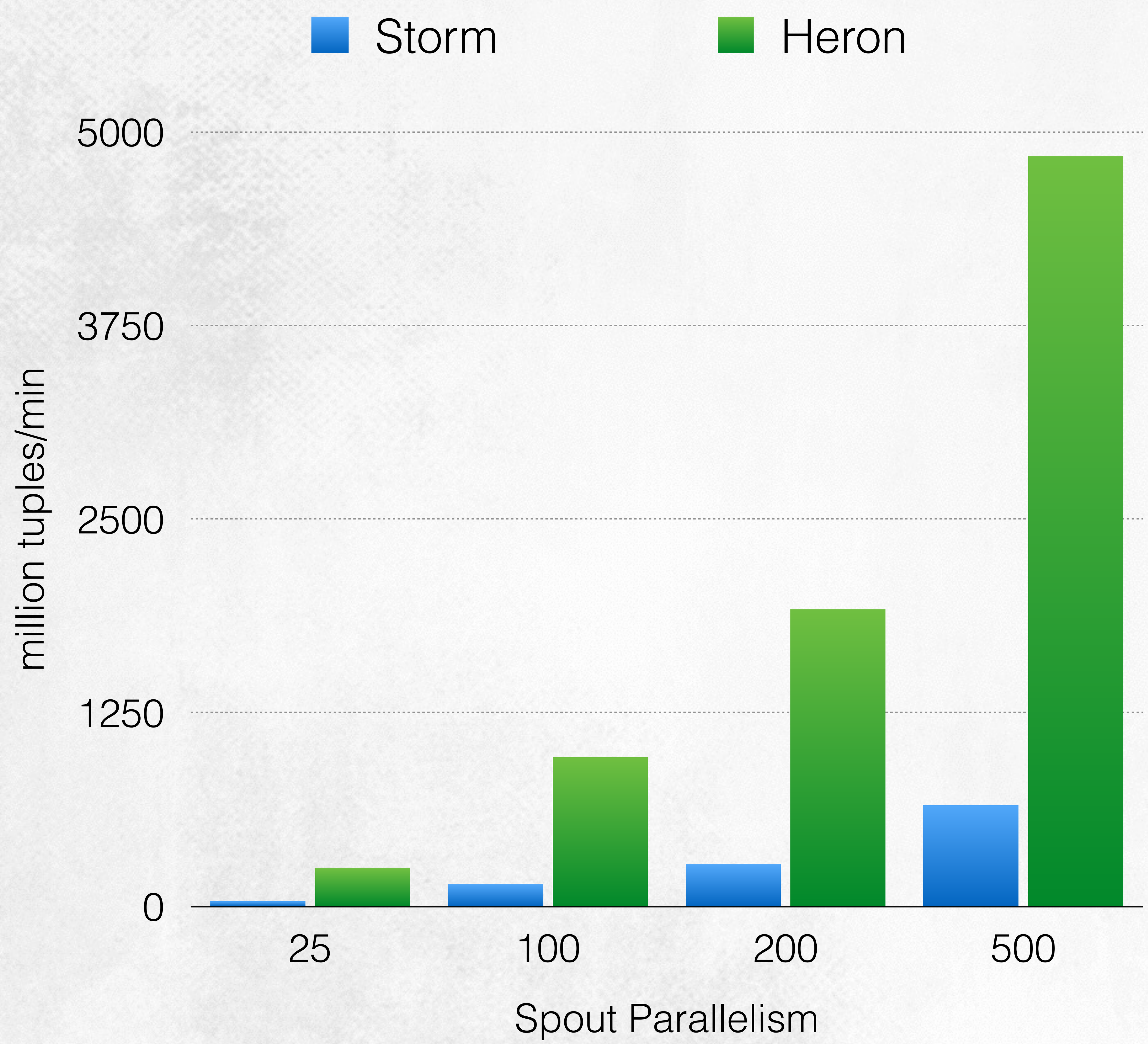
Word count topology - CPU usage

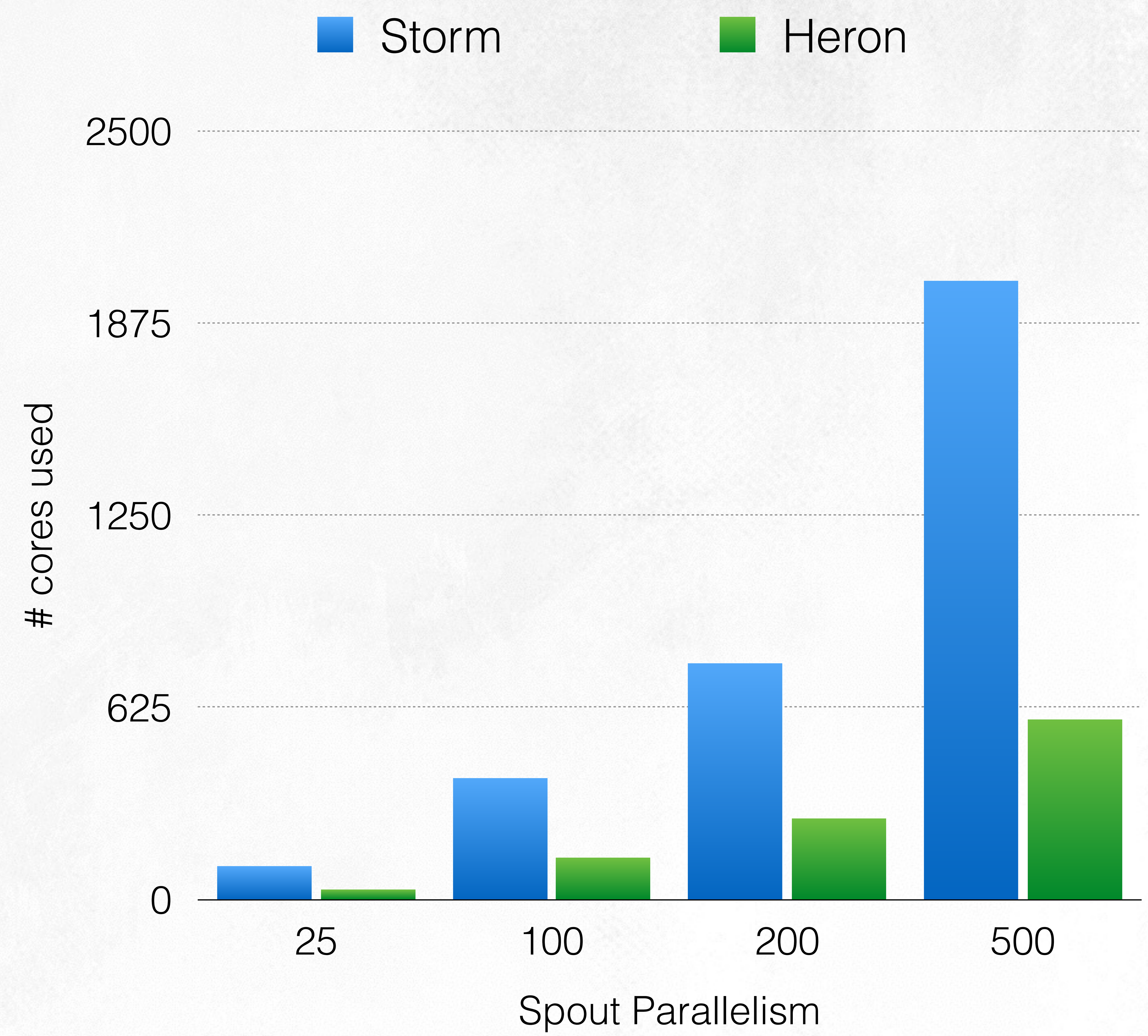


2-3x

HERON PERFORMANCE

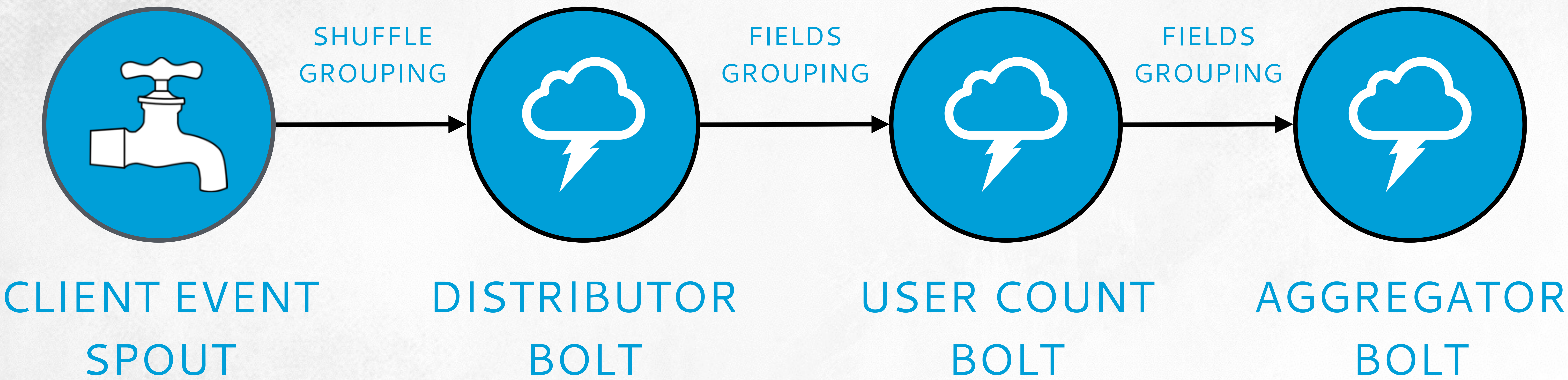
Throughput and CPU usage with no acknowledgements - Word count topology





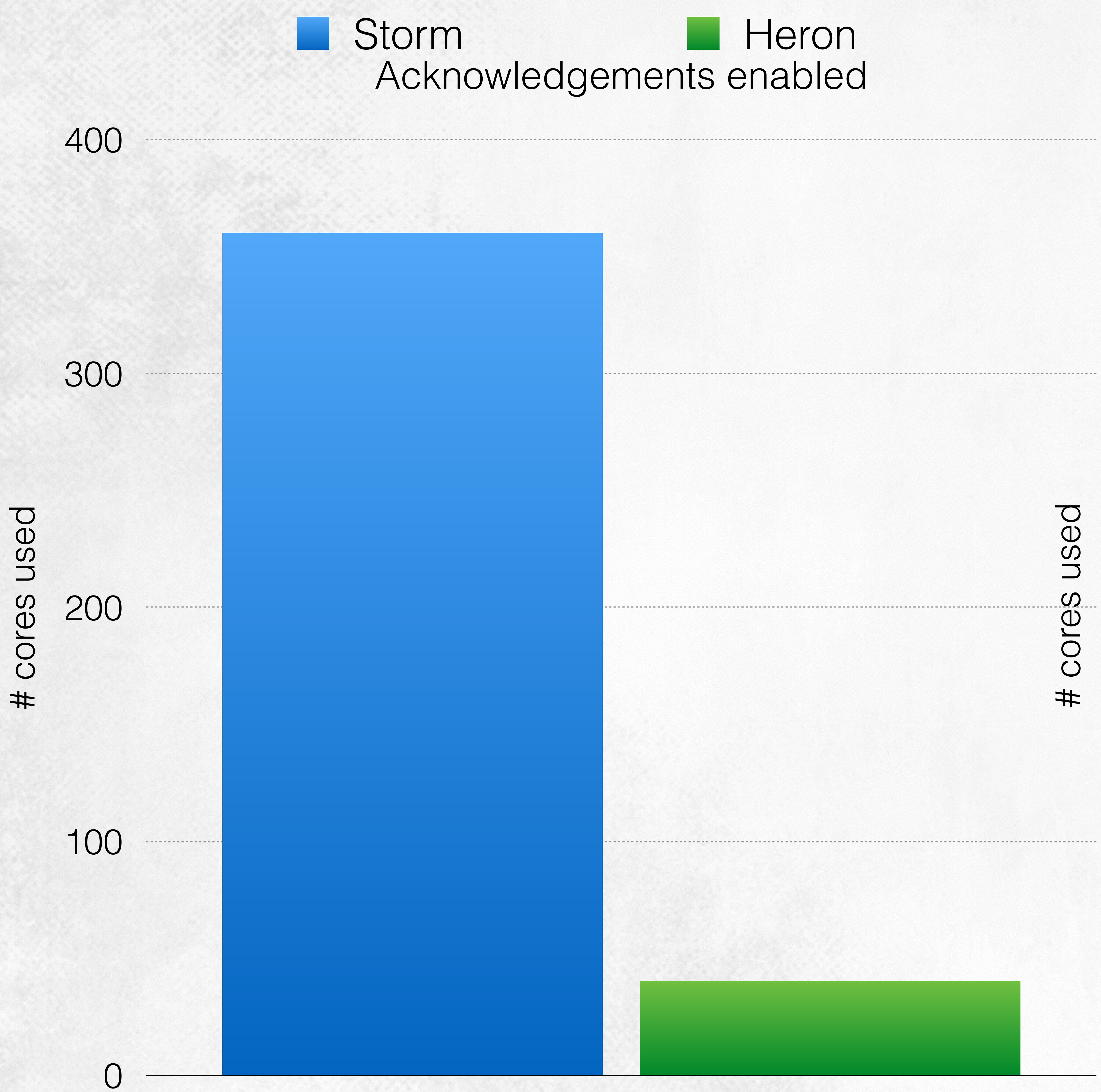
HERON EXPERIMENT

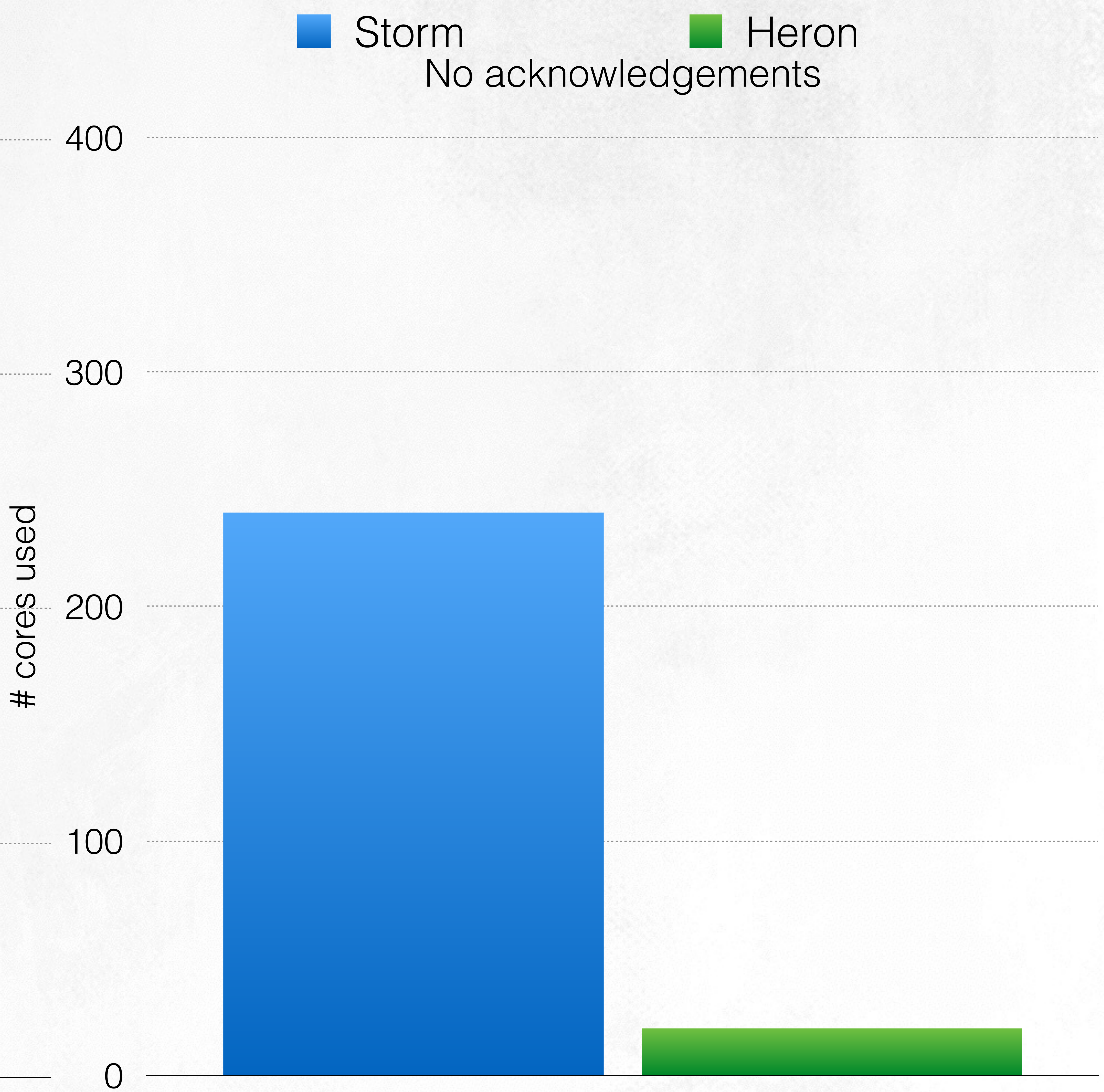
RTAC topology



HERON PERFORMANCE

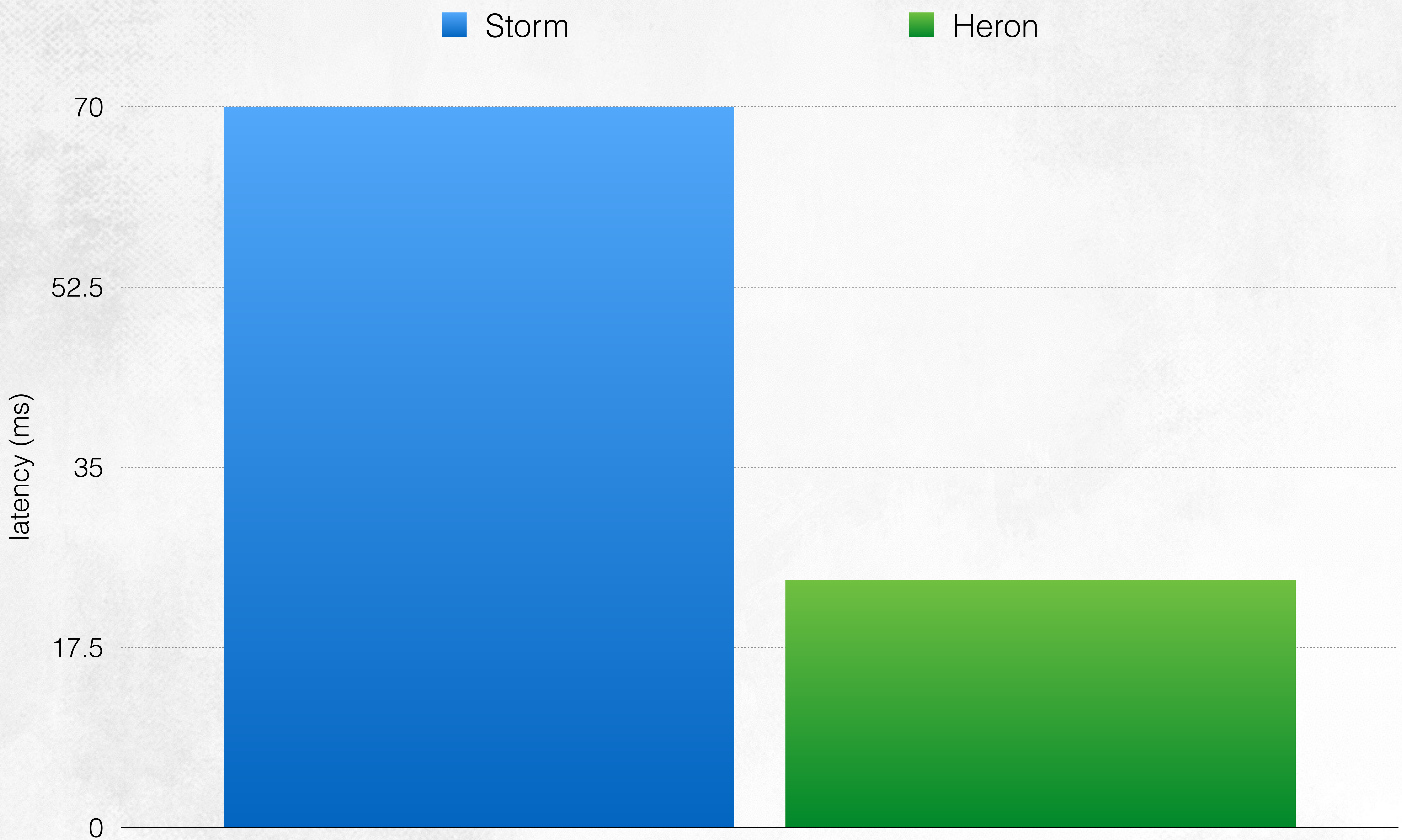
CPU usage – RTAC Topology





HERON PERFORMANCE

Latency with acknowledgements enabled - RTAC Topology



CURIOUS TO LEARN MORE...

Twitter Heron: Stream Processing at Scale

Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel[*,1], Karthik Ramasamy, Siddarth Taneja

@sanjeevrk, @challenger_nik, @Louis_Fumaosong, @vikkyrk, @cckellogg, @saileshmittal, @pateljm, @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, @pateljm, @sanjeevrk, @jason_j, @krishnagade, @Louis_Fumaosong, @jakedonham, @challenger_nik, @saileshmittal, @squarecog

Twitter, Inc., *University of Wisconsin – Madison

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.