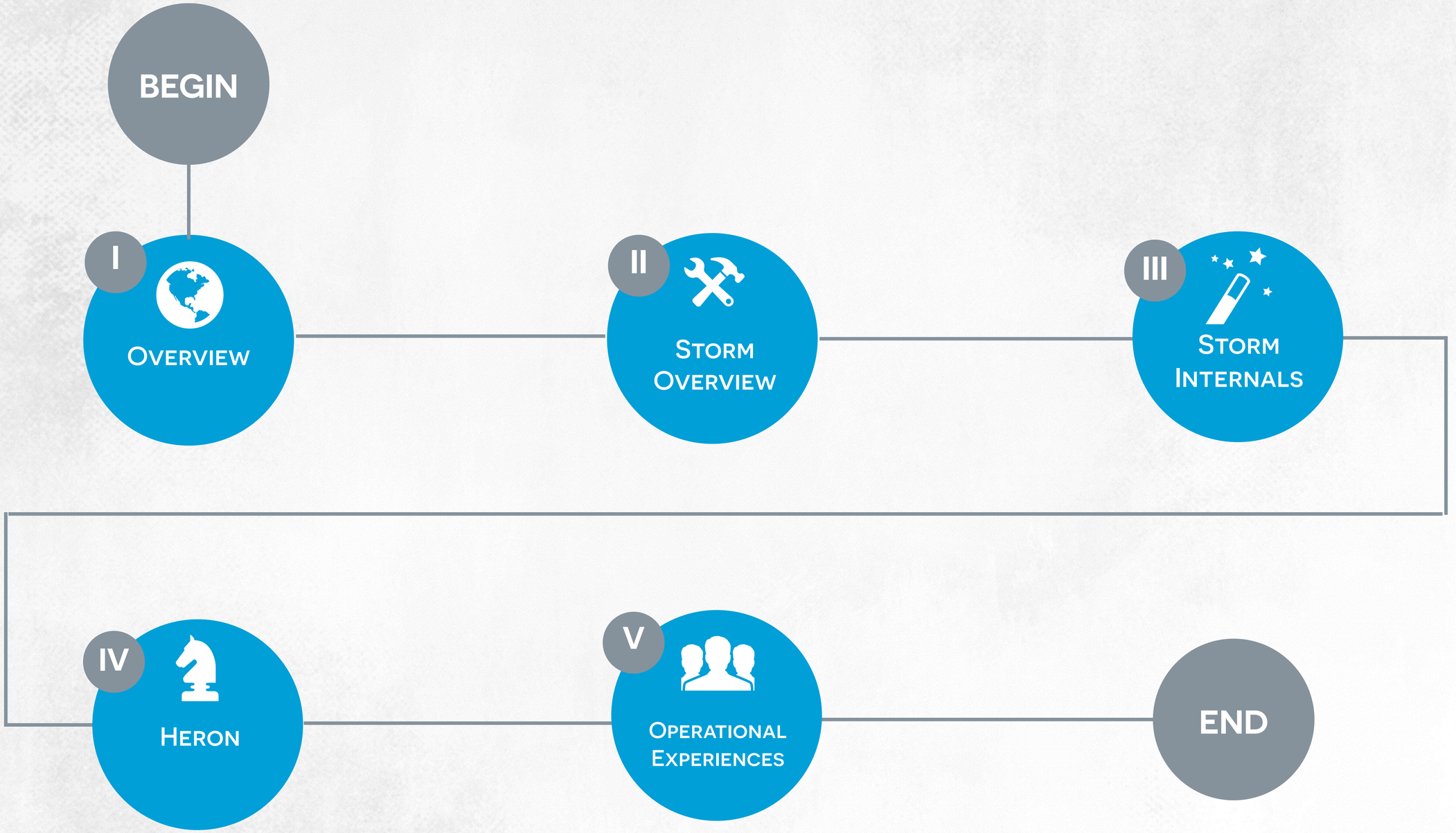
Scalable Streaming Analytics

KARTHIK RAMASAMY

@KARTHIKZ

TALK OUTLINE

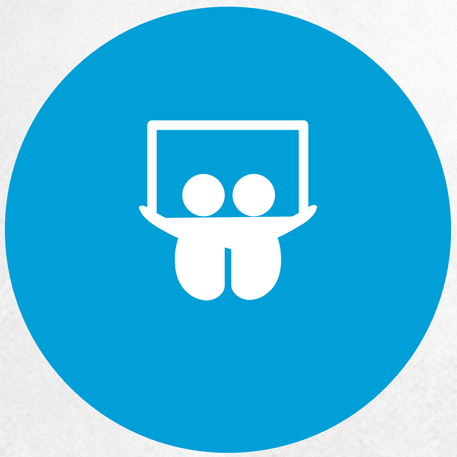


WHAT IS ANALYTICS?

according to Wikipedia

DISCOVERY

Ability to identify patterns in data



COMMUNICATION

Provide insights in a meaningful way

TYPES OF ANALYTICS

varieties

CUBE
ANALYTICS

PREDICTIVE
ANALYTICS

DIMENSIONS OF ANALYTICS

variants

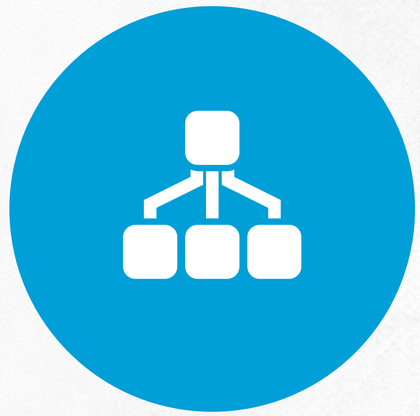
STREAMING

Ability to analyze the data immediately after it is produced

INTERACTIVE

Ability to provide results instantly when a query is posed

BATCH



Ability to provide insights after several hours/days when a query is posed

STREAMING VS INTERACTIVE

INTERACTIVE ANALYTICS

Static Batch
Results/Reports

Queries

Bulkload

Data

Database Server

Data Storage

STREAMING ANALYTICS

Real time alerts,
Real time analytics
Continuous visibility

Results

Data Stream Processing

Data Storage

Queries

WHAT IS REAL TIME?

msecs or secs or mins?

< 500 MS

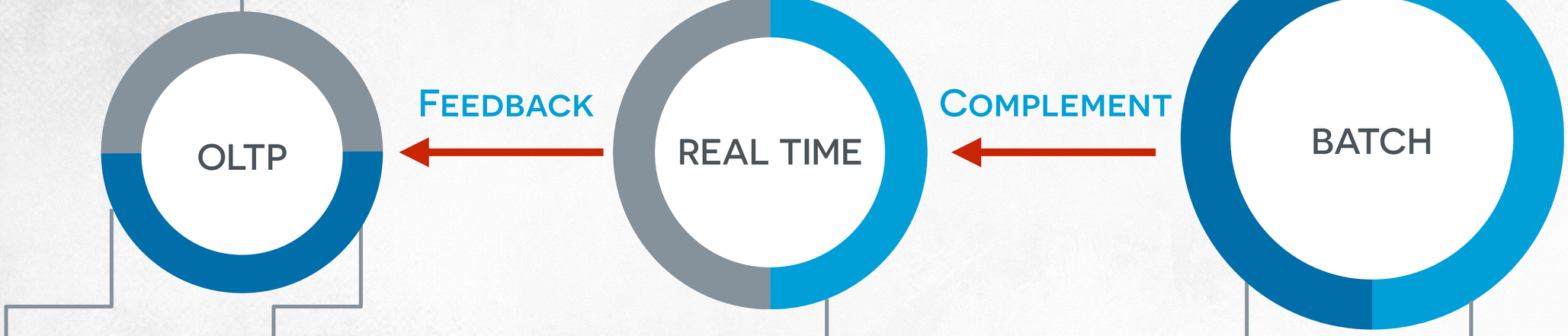
latency sensitive

> 1 SEC

approximate

> 1 HOUR

high throughput



deterministic workflows

fanout Tweets search for Tweets

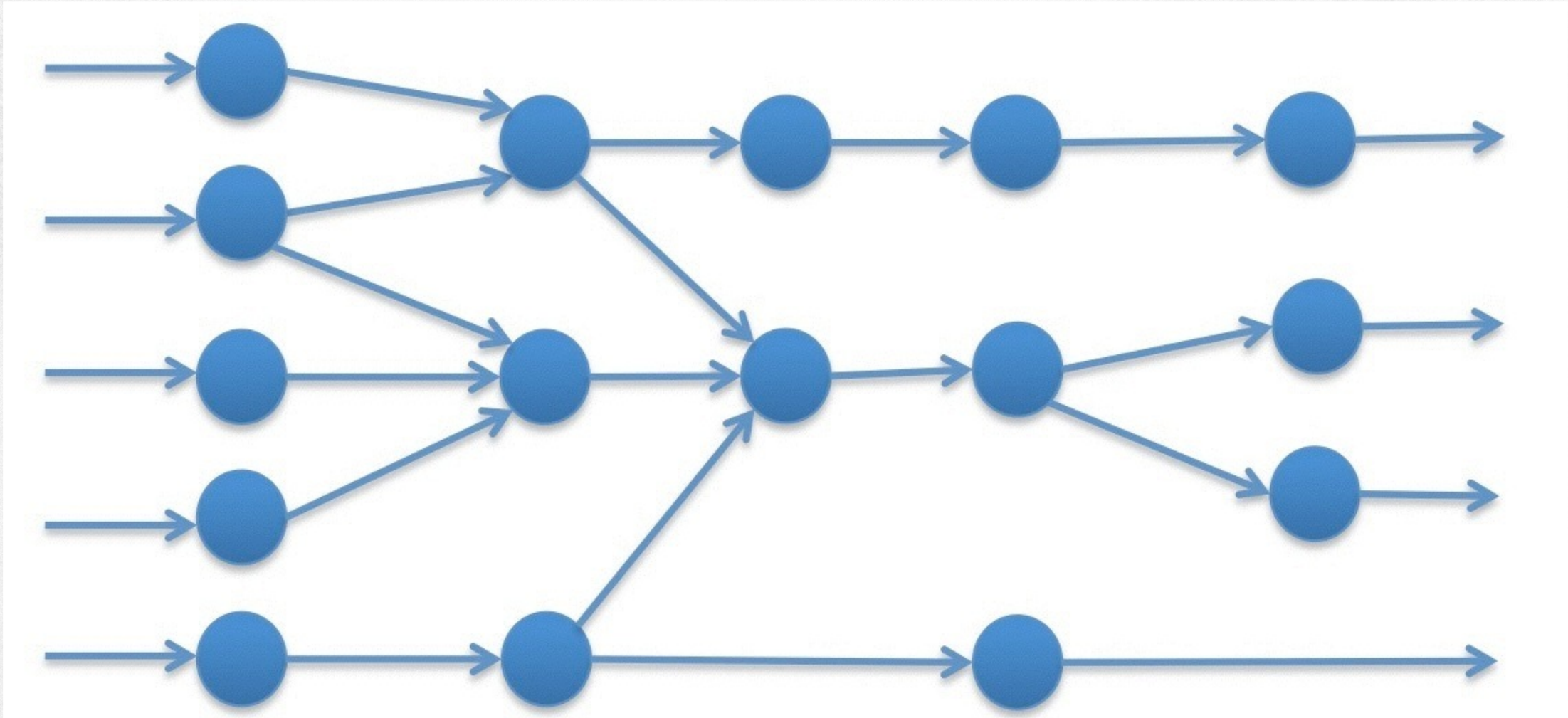
ad impressions count hash tag trends

adhoc queries

monthly active users relevance for ads

STREAMING DATA FLOW

varieties



STREAMING SYSTEMS

first generation - SQL based

NIAGARA Query Engine

Stanford Stream Data Manager

Aurora Stream Processing Engine

Borealis Distributed Stream Processing Engine

Cayuga - Stateful Event Monitoring

STREAMING SYSTEMS

next generation - too many

S4 distributed stream computing platform

STORM

samza

Spark

PULSAR

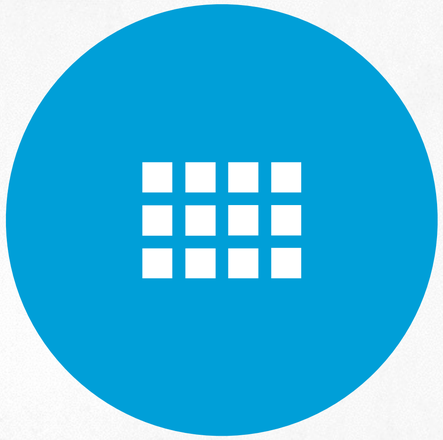
STORM OVERVIEW

I

WHAT IS STORM?

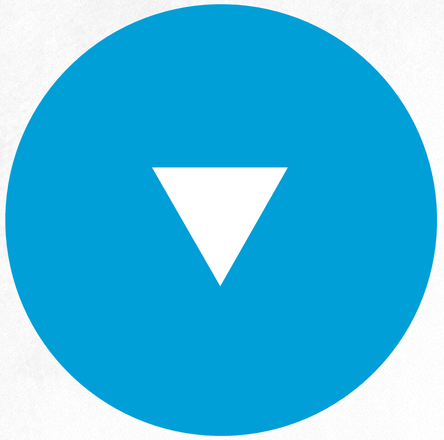
Streaming platform for analyzing **realtime data** as they arrive, so you can react to data **as it happens**.

GUARANTEED MESSAGE PROCESSING

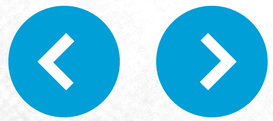


HORIZONTAL SCALABILITY

ROBUST FAULT TOLERANCE



CONCISE CODE- FOCUS ON LOGIC



STORM DATA MODEL

TOPOLOGY

Directed acyclic graph

Vertices=computation, and edges=streams of data tuples

SPOUTS

Sources of data tuples for the topology

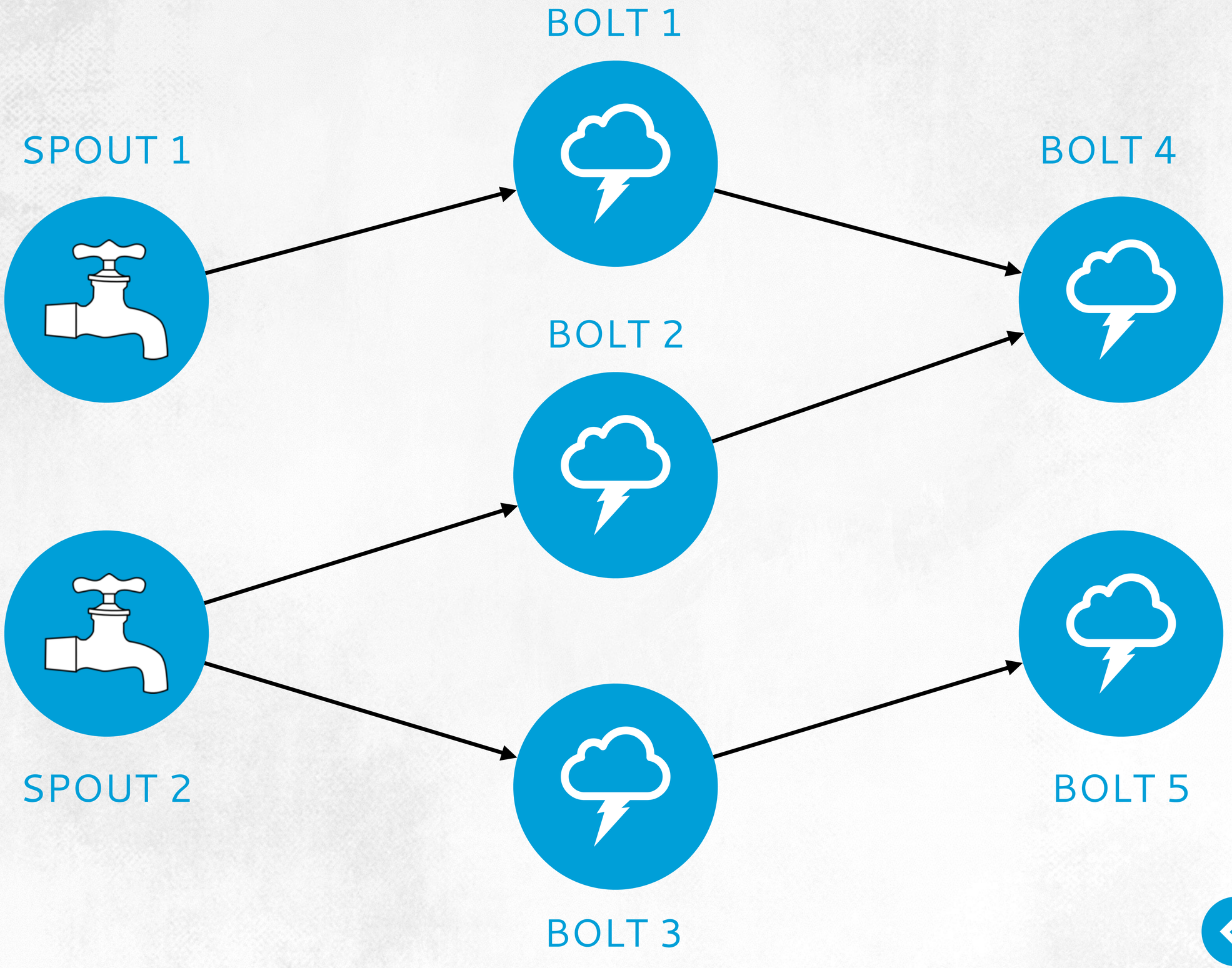
Examples - Kafka/Kestrel/MySQL/Postgres

BOLTS

Process incoming tuples and emit outgoing tuples

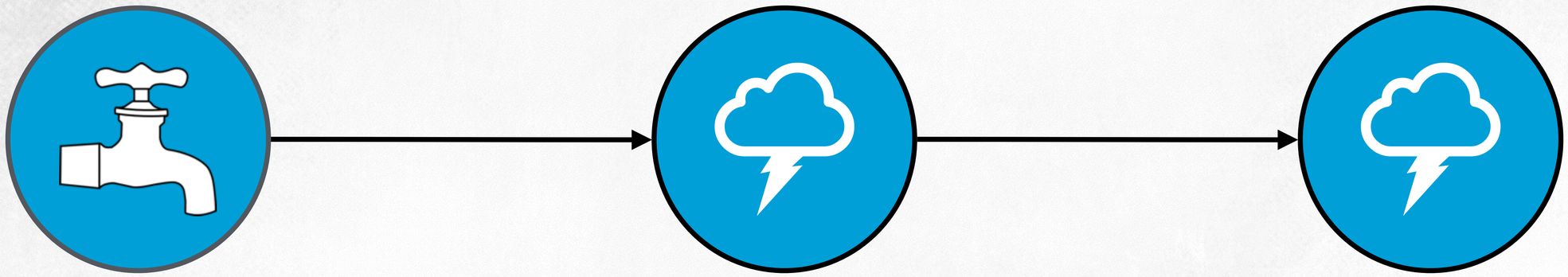
Examples - filtering/aggregation/join/arbitrary function

STORM TOPOLOGY



WORD COUNT TOPOLOGY

Live stream of Tweets



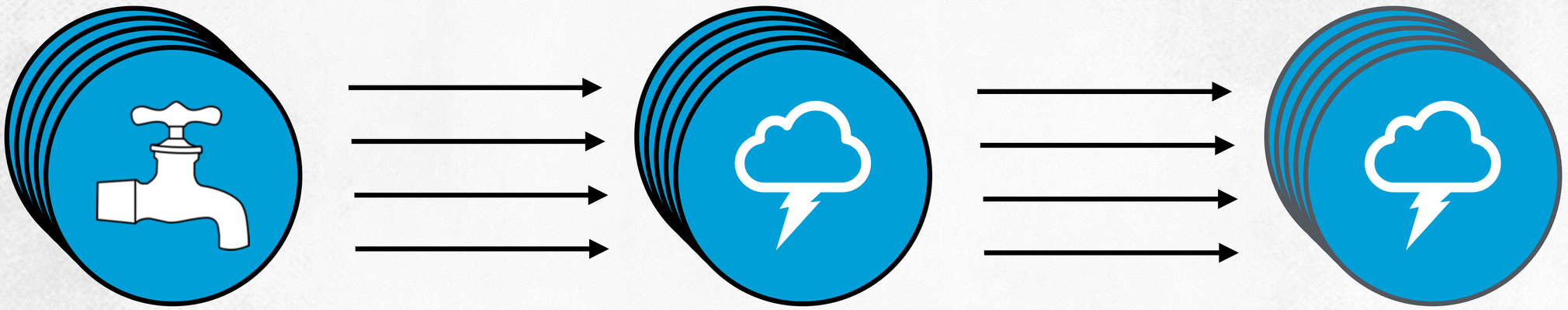
TWEET SPOUT

PARSE TWEET BOLT

WORD COUNT BOLT

LOGICAL PLAN

WORD COUNT TOPOLOGY



TWEET SPOUT TASKS

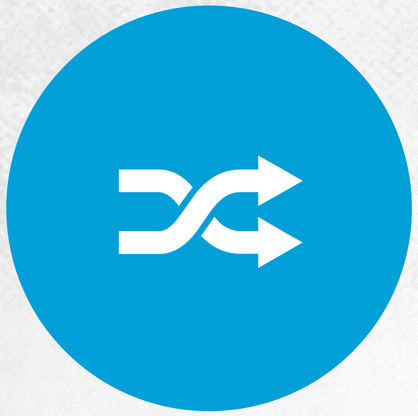
PARSE TWEET BOLT TASKS

WORD COUNT BOLT TASKS

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

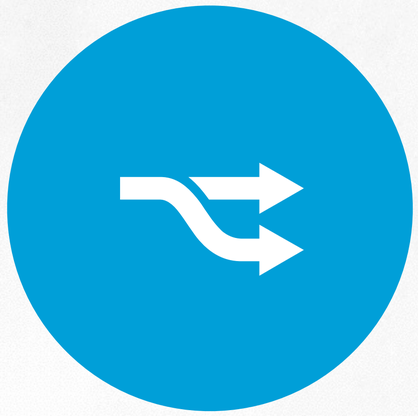
STREAM GROUPINGS

SHUFFLE GROUPING



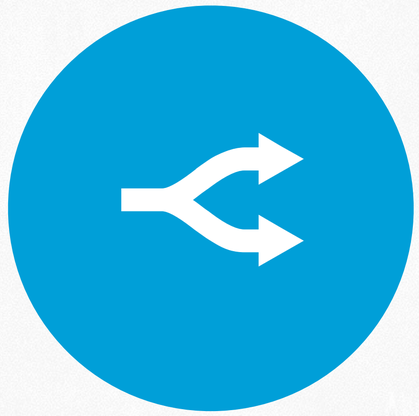
Random distribution of tuples

FIELDS GROUPING



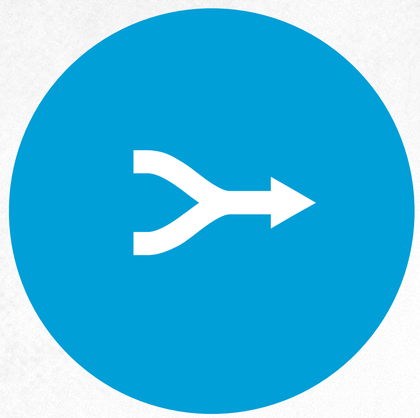
Group tuples by a field or multiple fields

ALL GROUPING



Replicates tuples to all tasks

GLOBAL GROUPING

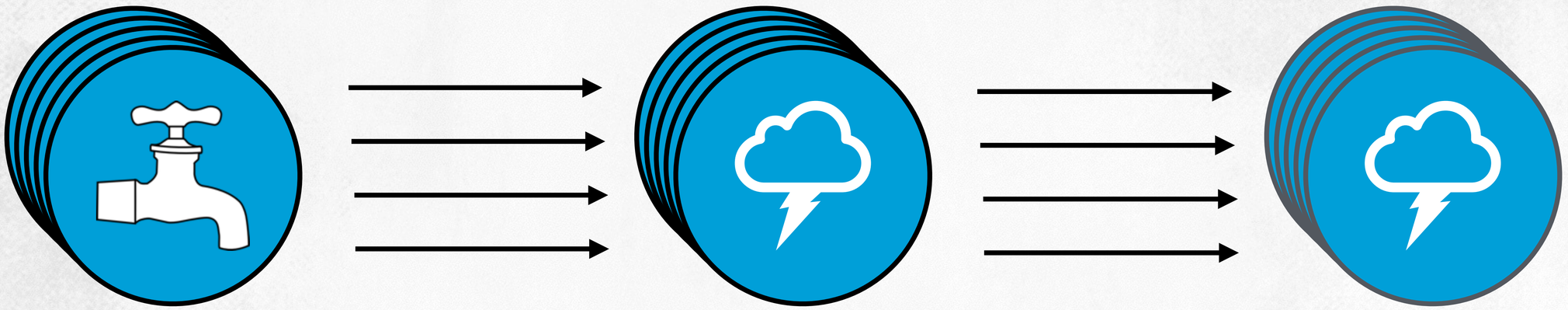


Sends the entire stream to one task

WORD COUNT TOPOLOGY

SHUFFLE GROUPING

FIELDS GROUPING



TWEET SPOUT TASKS

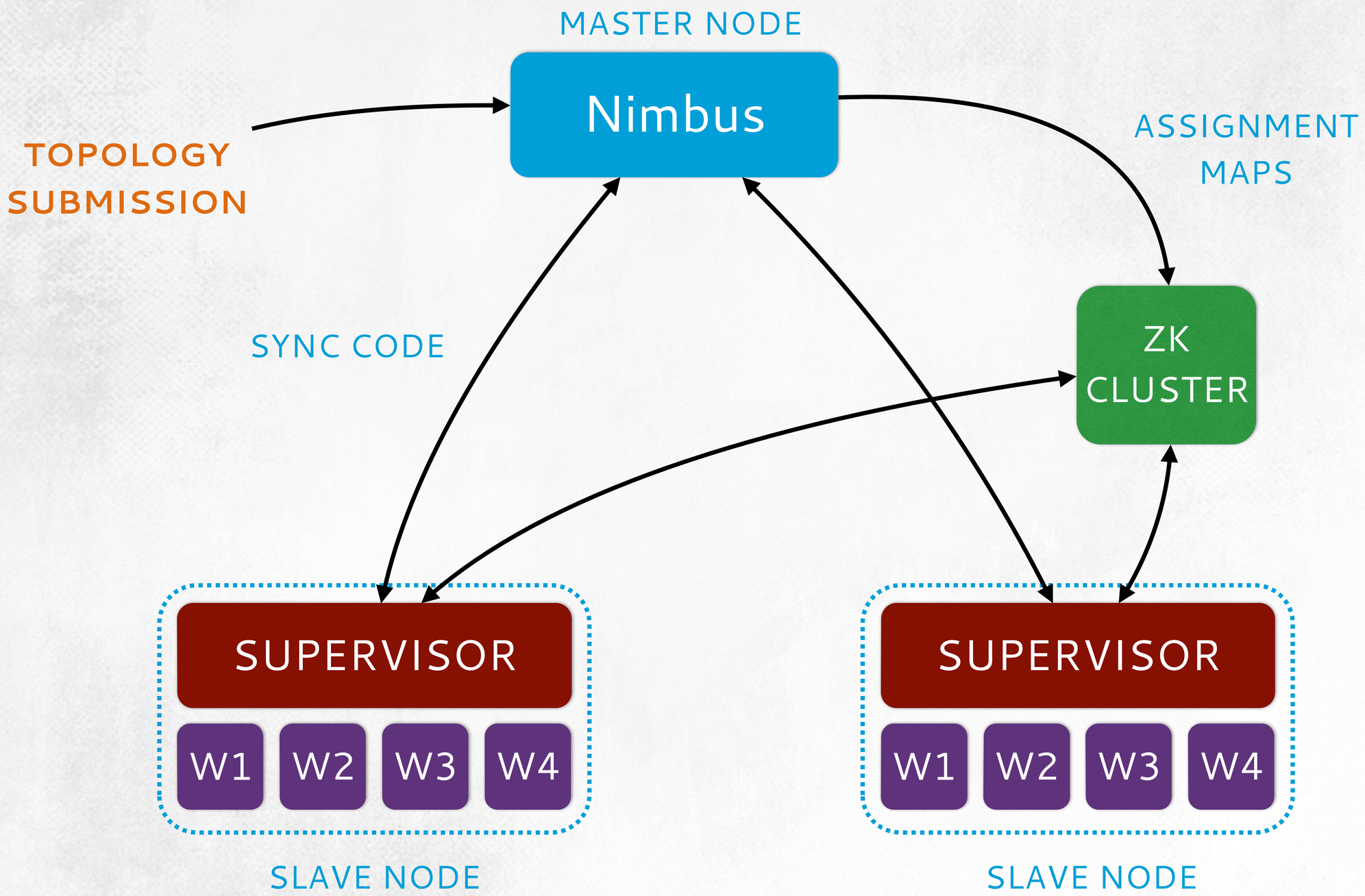
PARSE TWEET BOLT TASKS

WORD COUNT BOLT TASKS

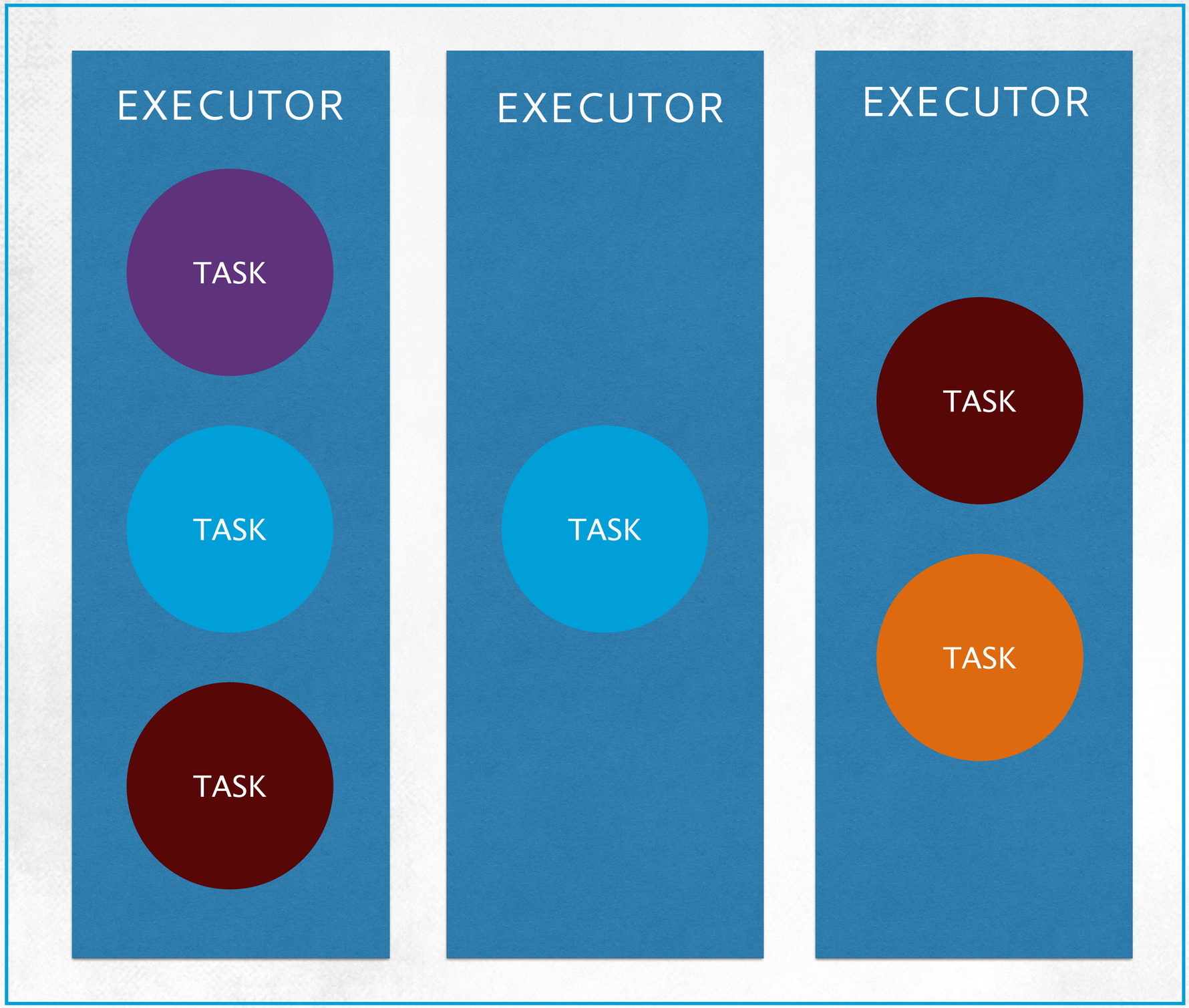
II

STORM INTERNALS

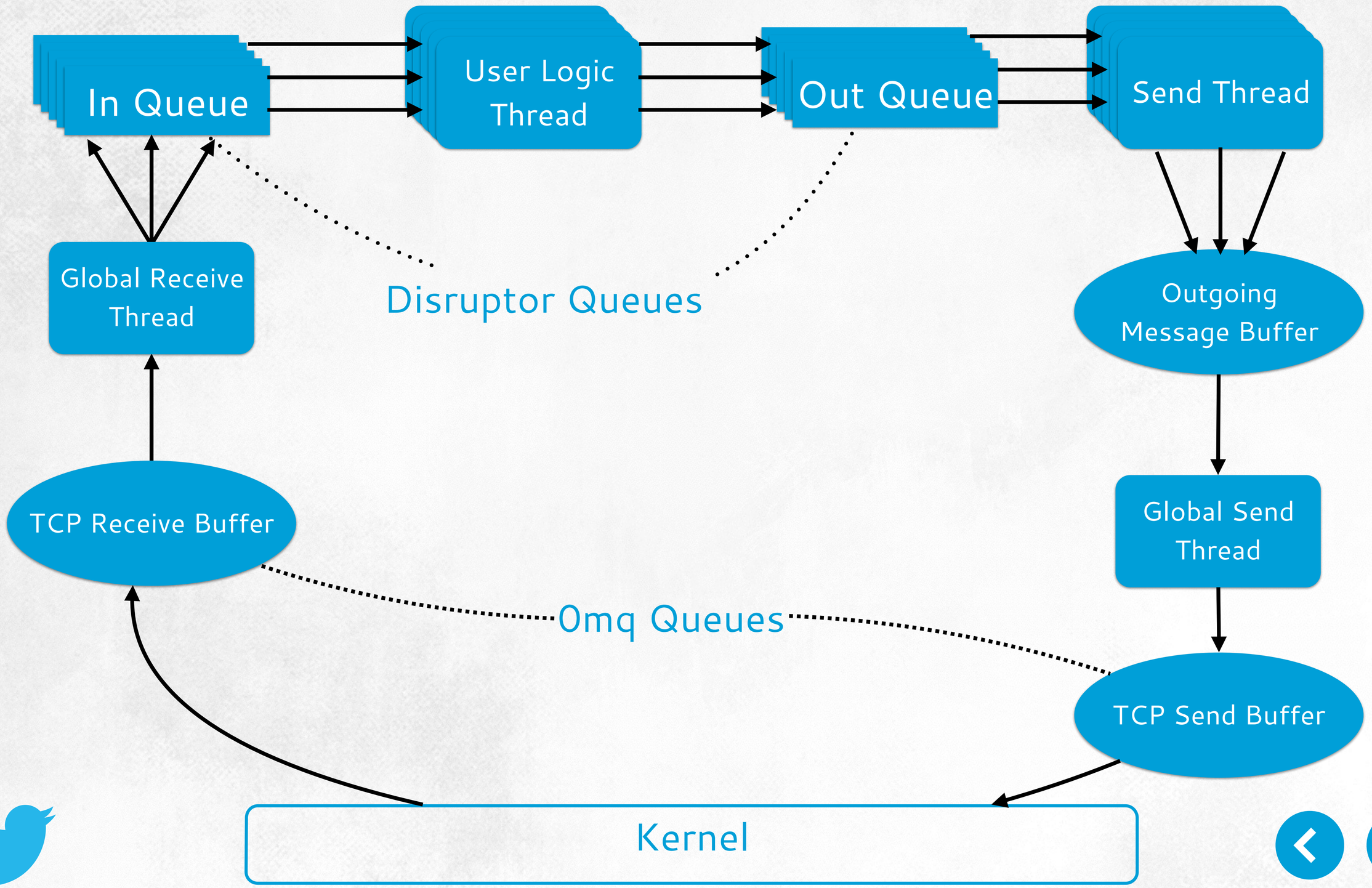
STORM ARCHITECTURE



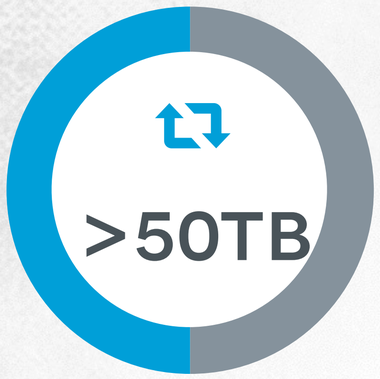
STORM WORKER



DATA FLOW IN STORM WORKERS

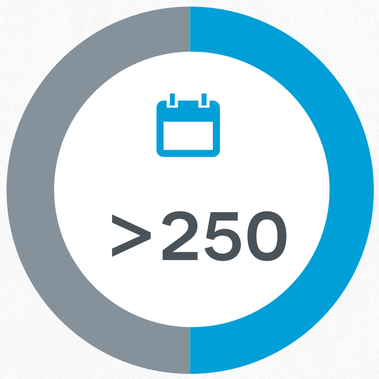


STORM @TWITTER

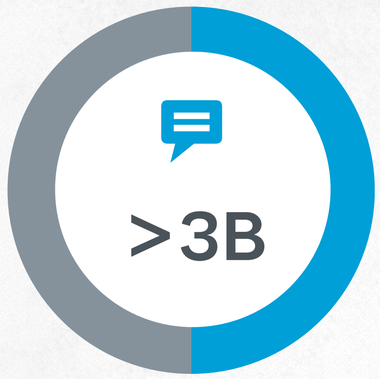


Large amount of data produced every day

Largest storm cluster

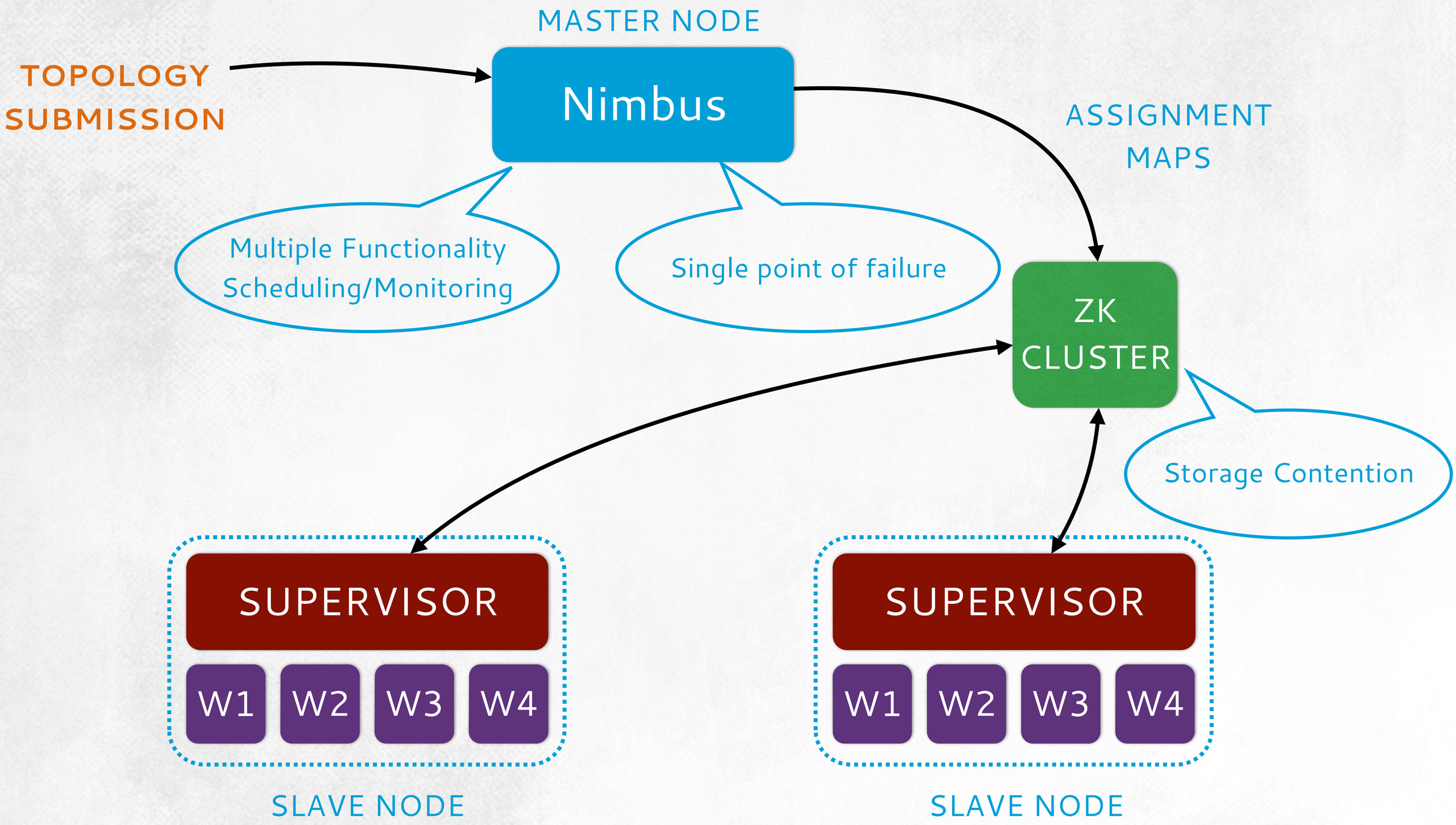


Several topologies deployed

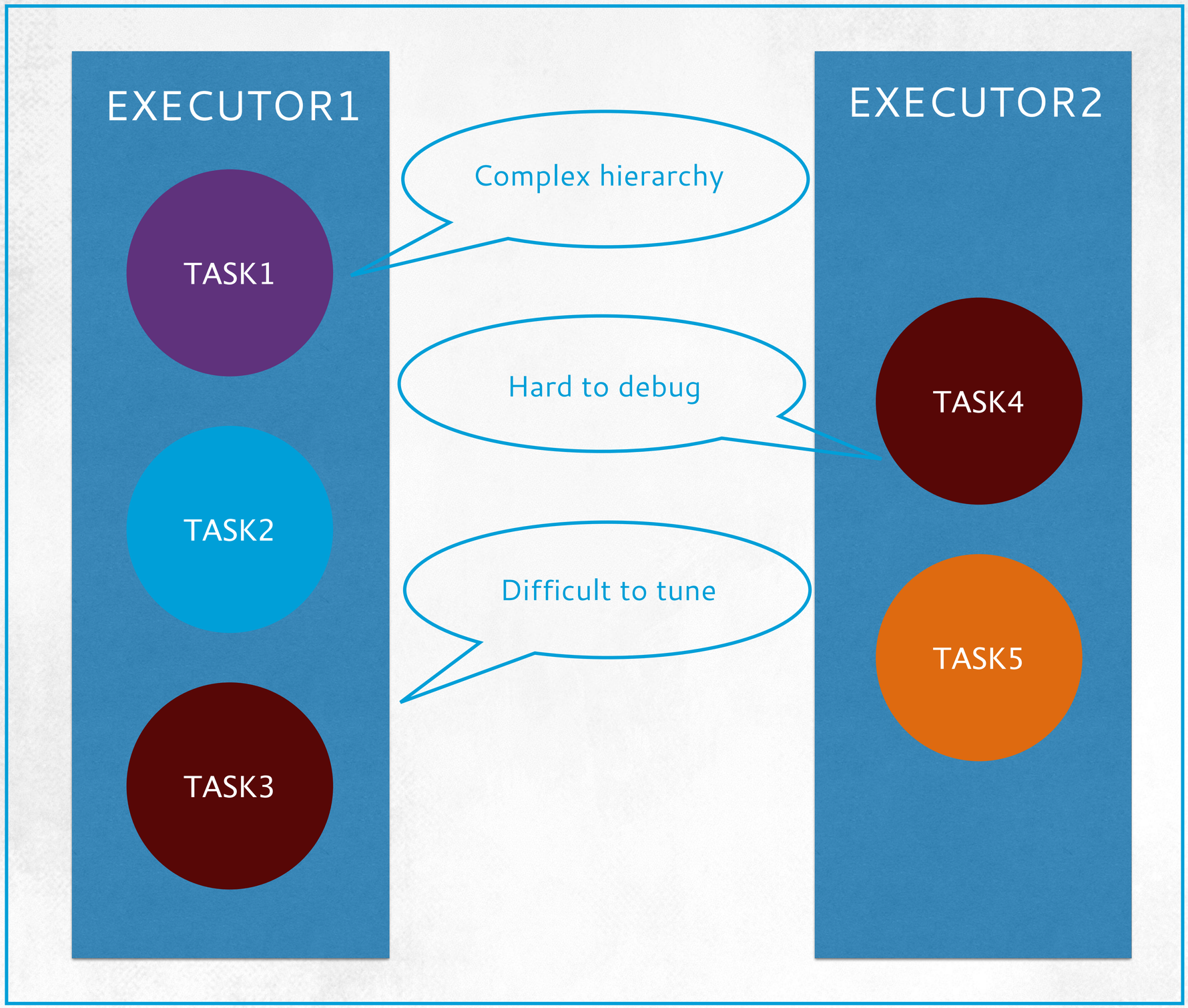


Several billion messages every day

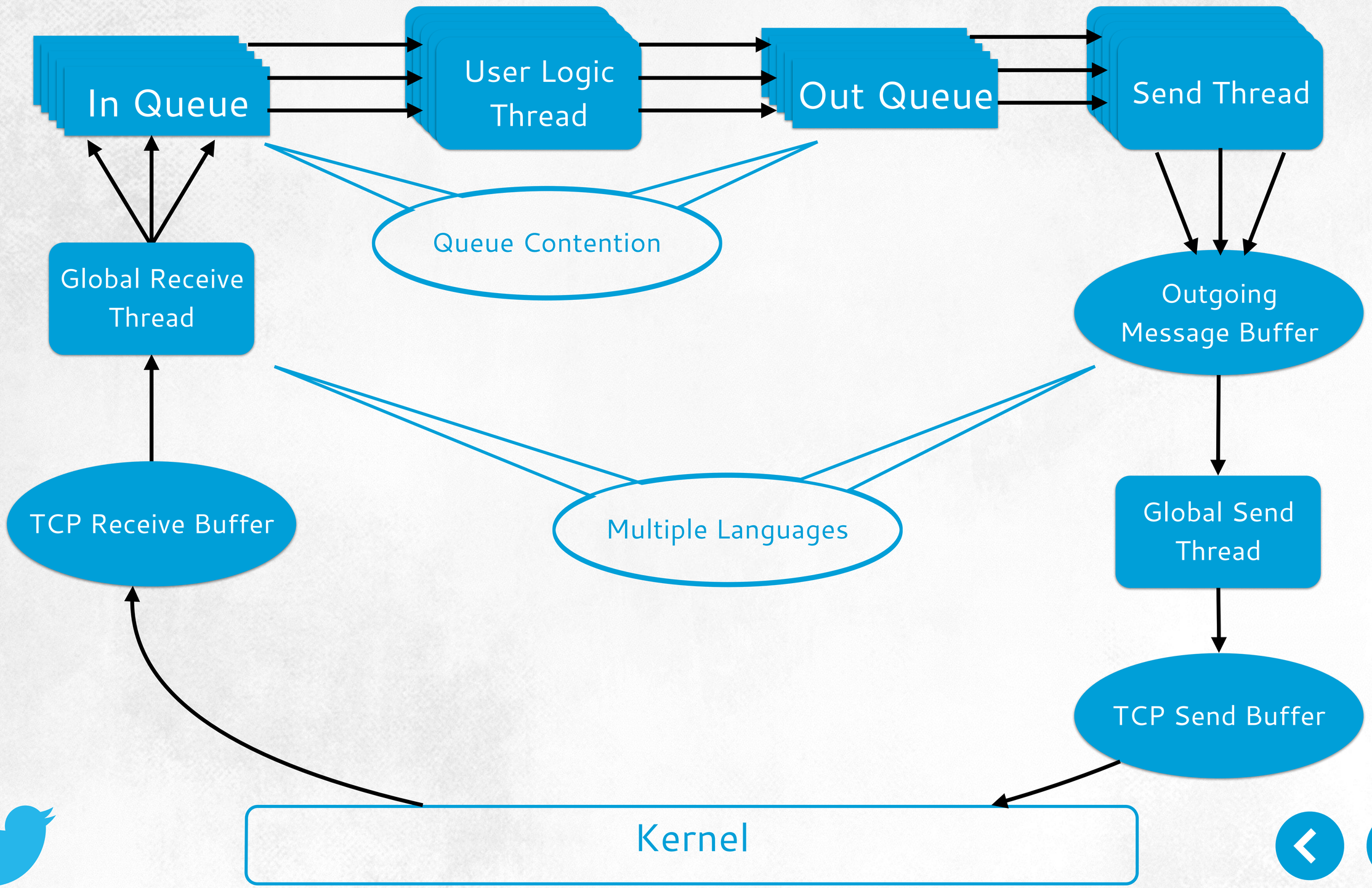
STORM ARCHITECTURE



STORM WORKER

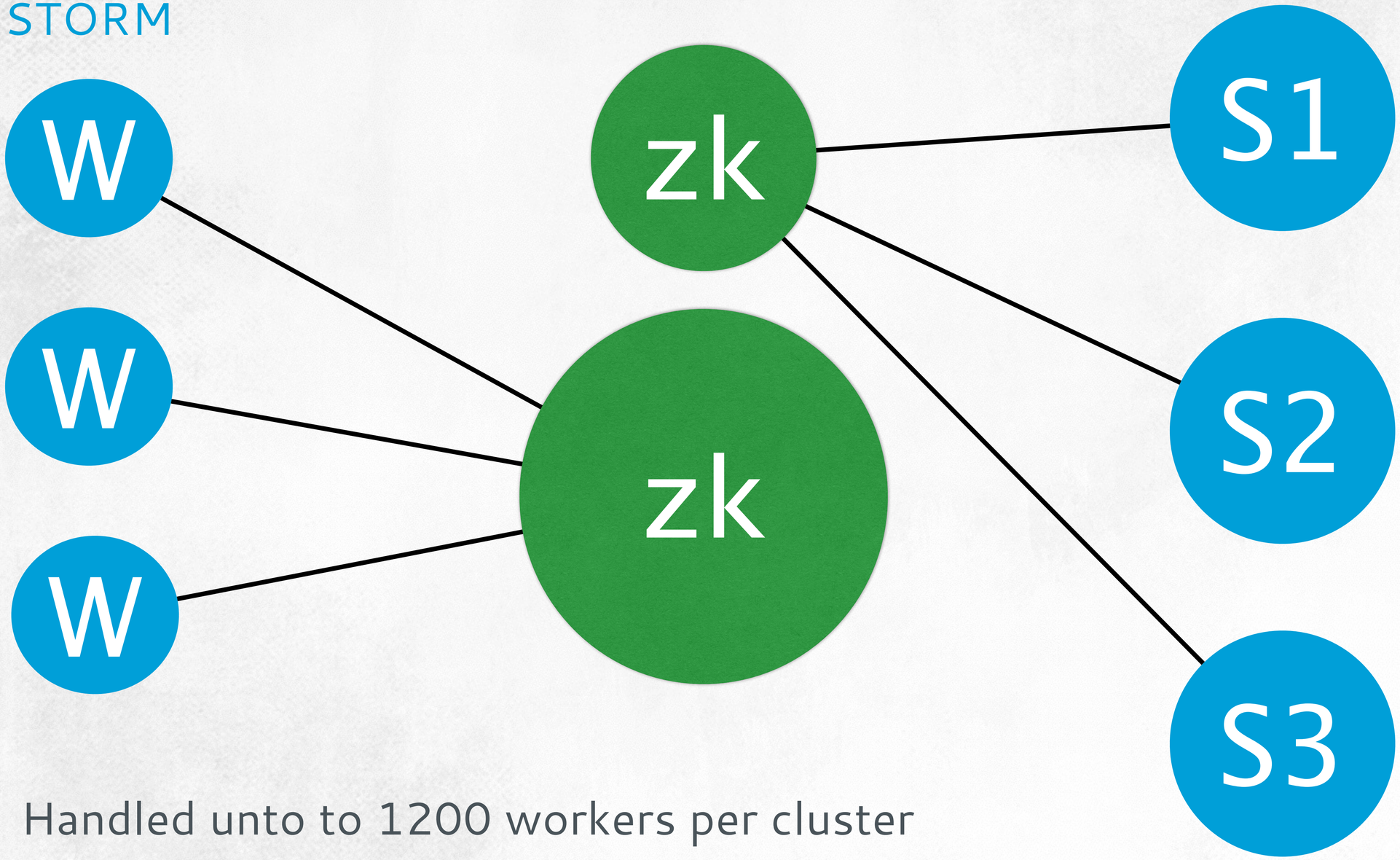


DATA FLOW IN STORM WORKERS



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

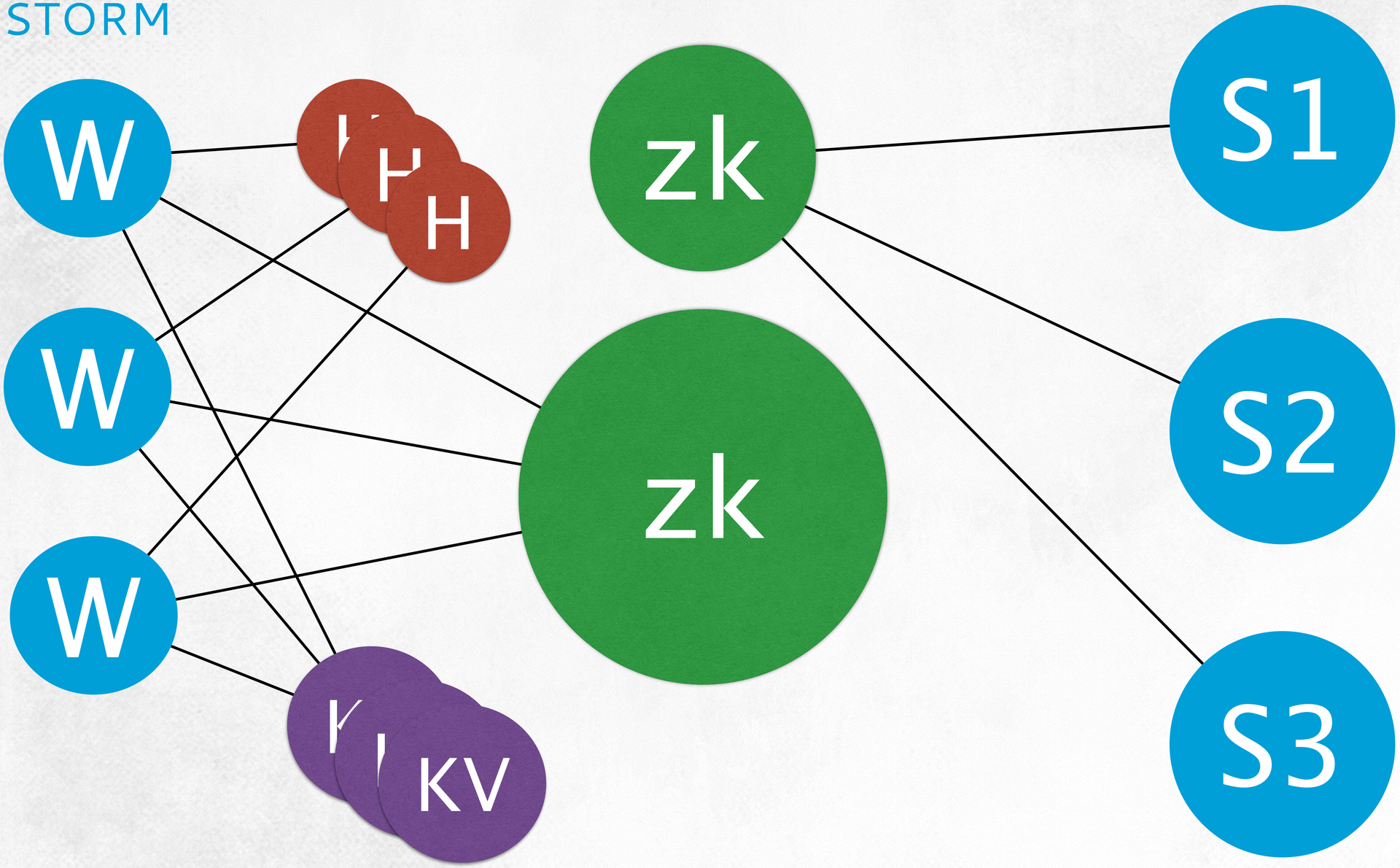
33%

STORM RUNTIME

Workers write heart beats every 3 secs

OVERLOADED ZOOKEEPER

Heart beat daemons



5000 workers per cluster

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

III

HERON

HERON DESIGN GOALS

FULLY API COMPATIBLE WITH STORM

Directed acyclic graph

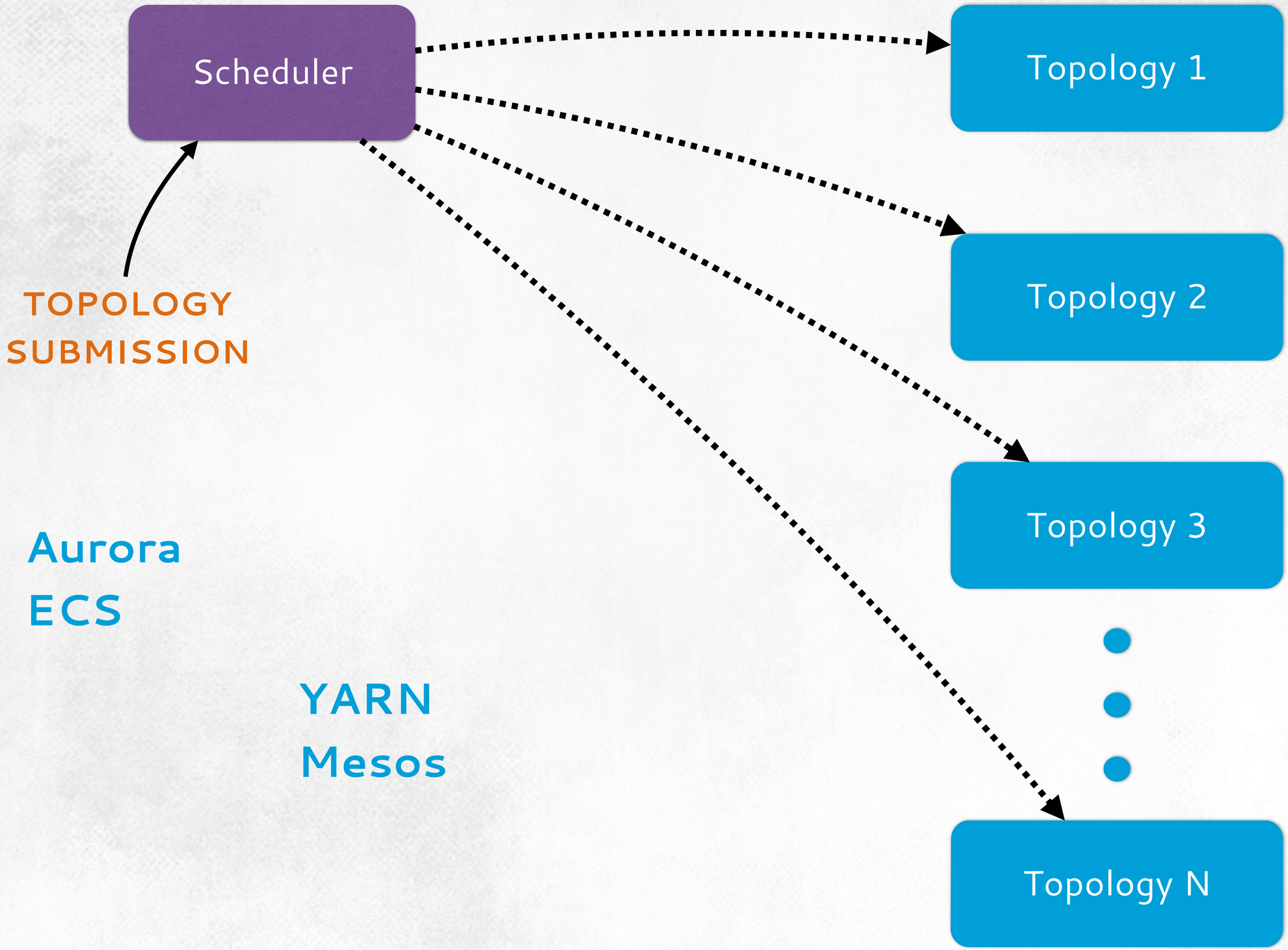
Topologies, spouts and bolts

USE OF WELL KNOWN LANGUAGES

No Clojure

C++/JAVA/Python

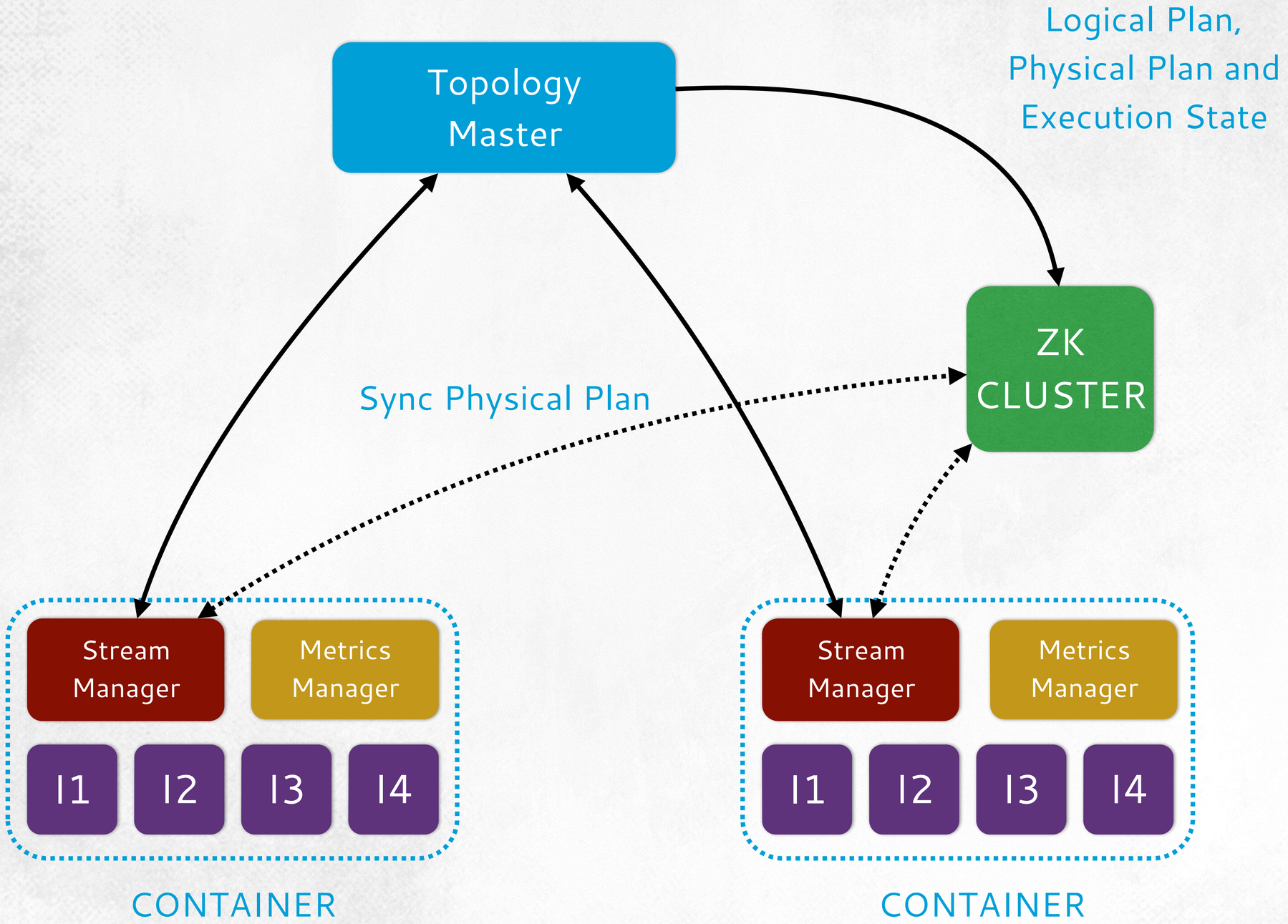
HERON ARCHITECTURE



Aurora
ECS

YARN
Mesos

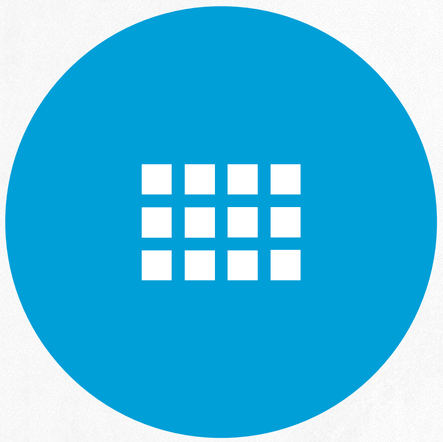
TOPOLOGY ARCHITECTURE



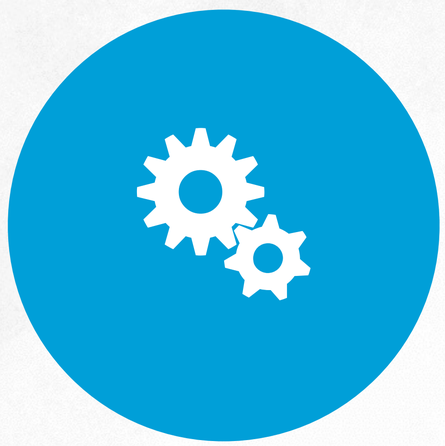
TOPOLOGY MASTER

Solely responsible for the entire topology

ASSIGNS ROLE

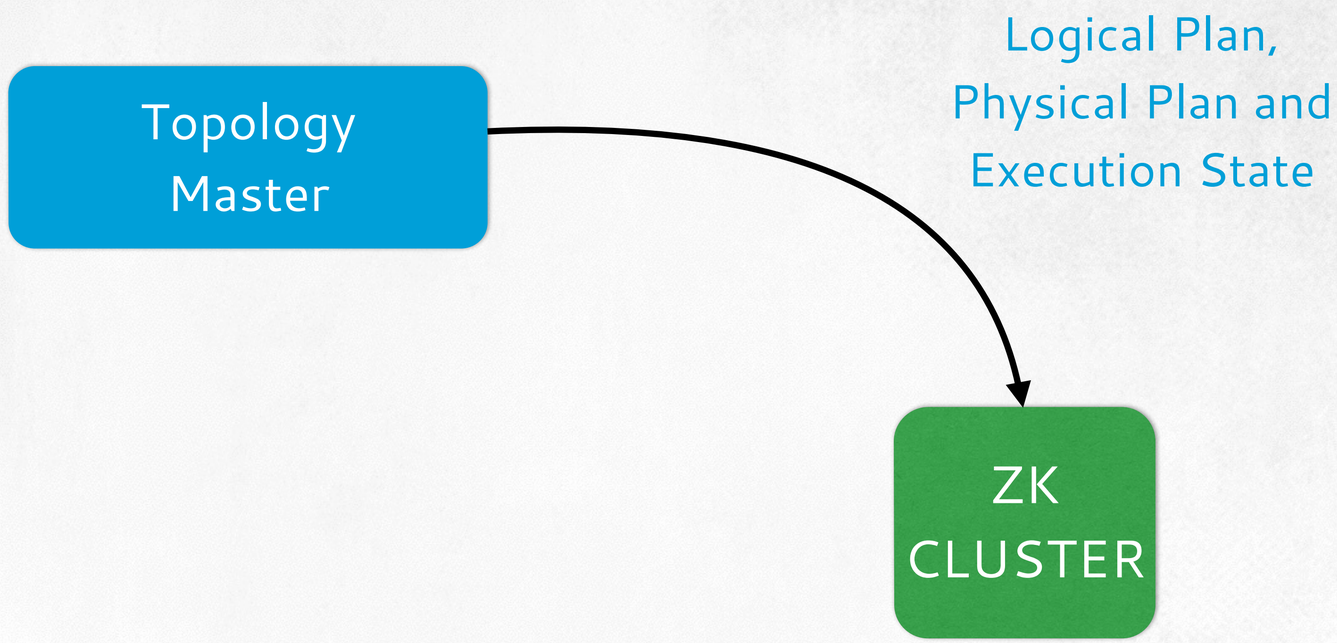


MONITORING



METRICS

TOPOLOGY MASTER

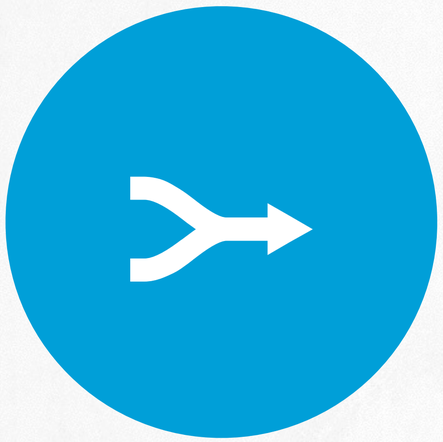


- PREVENT MULTIPLE TM BECOMING MASTERS
- ALLOWS OTHER PROCESS TO DISCOVER TM

STREAM MANAGER

Routing Engine

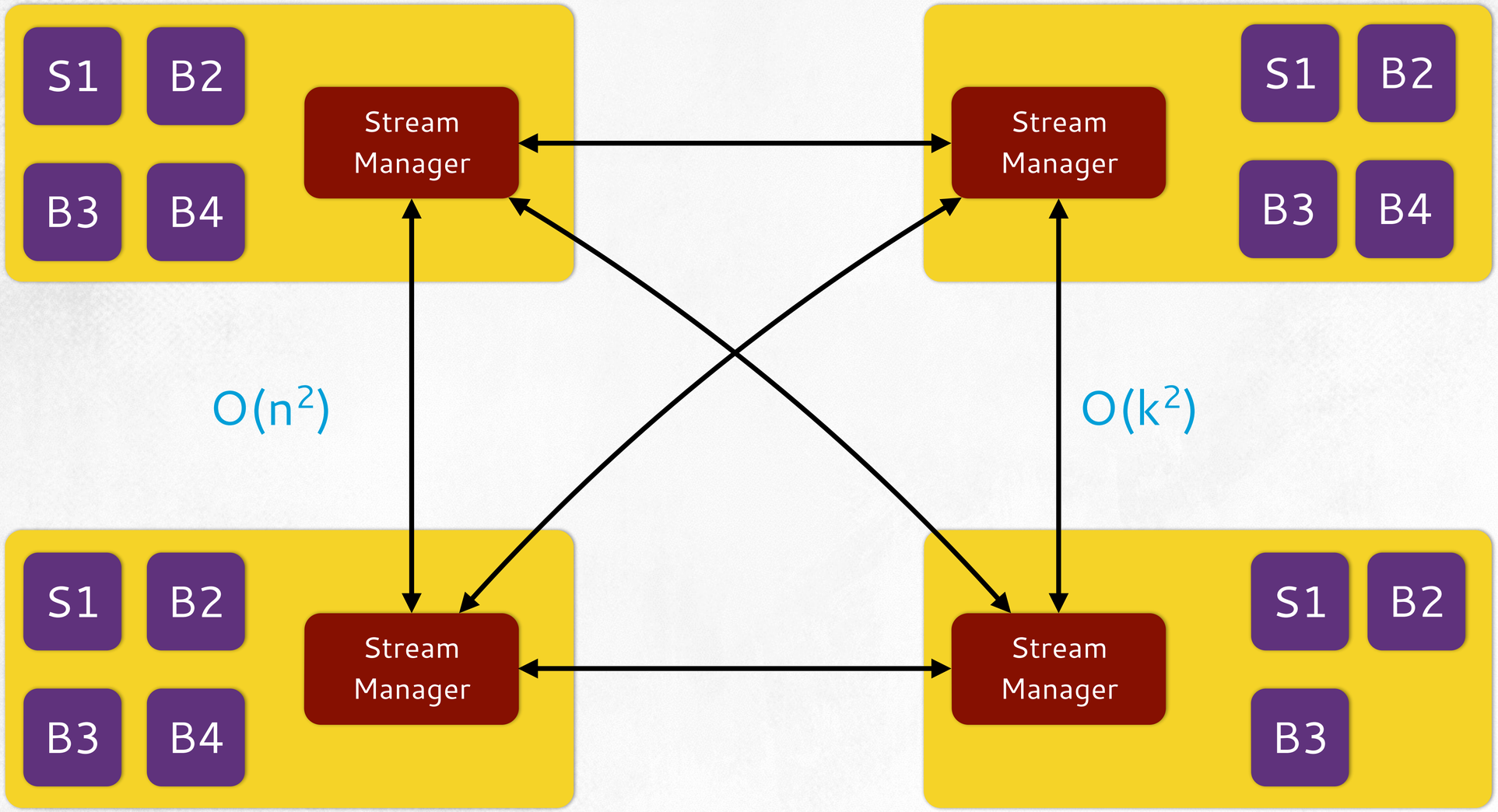
ROUTES TUPLES



BACKPRESSURE

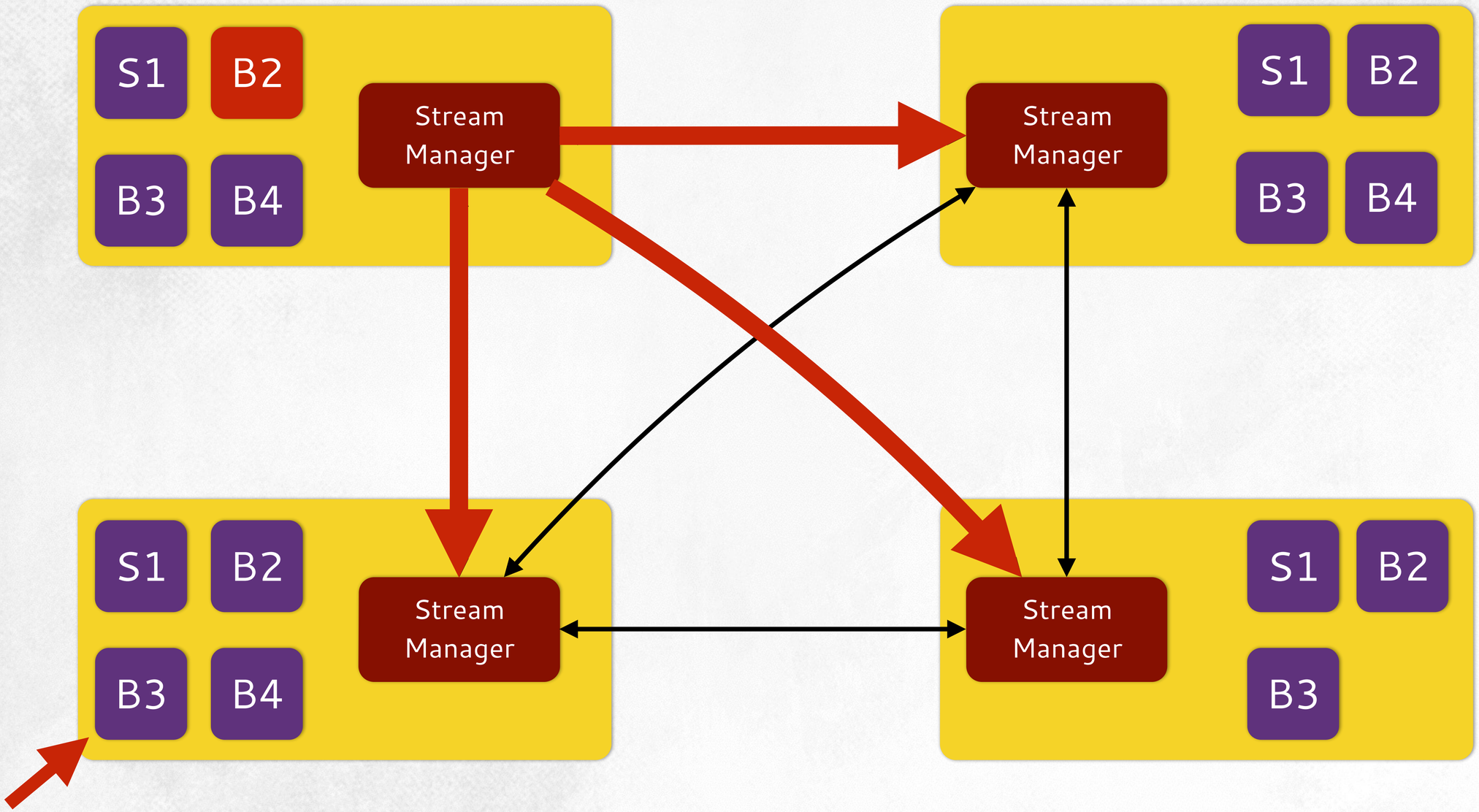
ACK MGMT

STREAM MANAGER



STREAM MANAGER

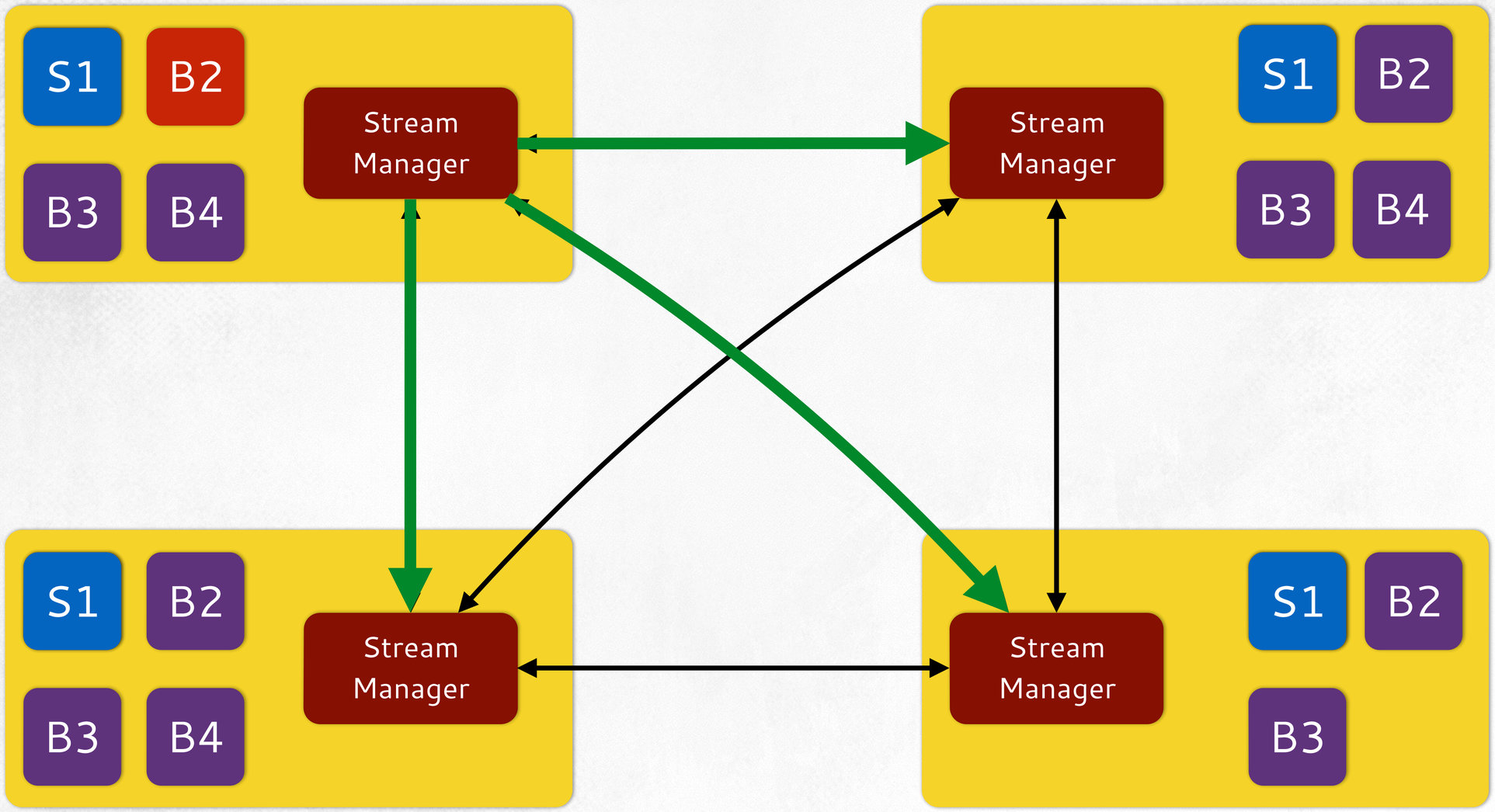
tcp back pressure



SLOWS UPSTREAM AND DOWNSTREAM INSTANCES

STREAM MANAGER

spout 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

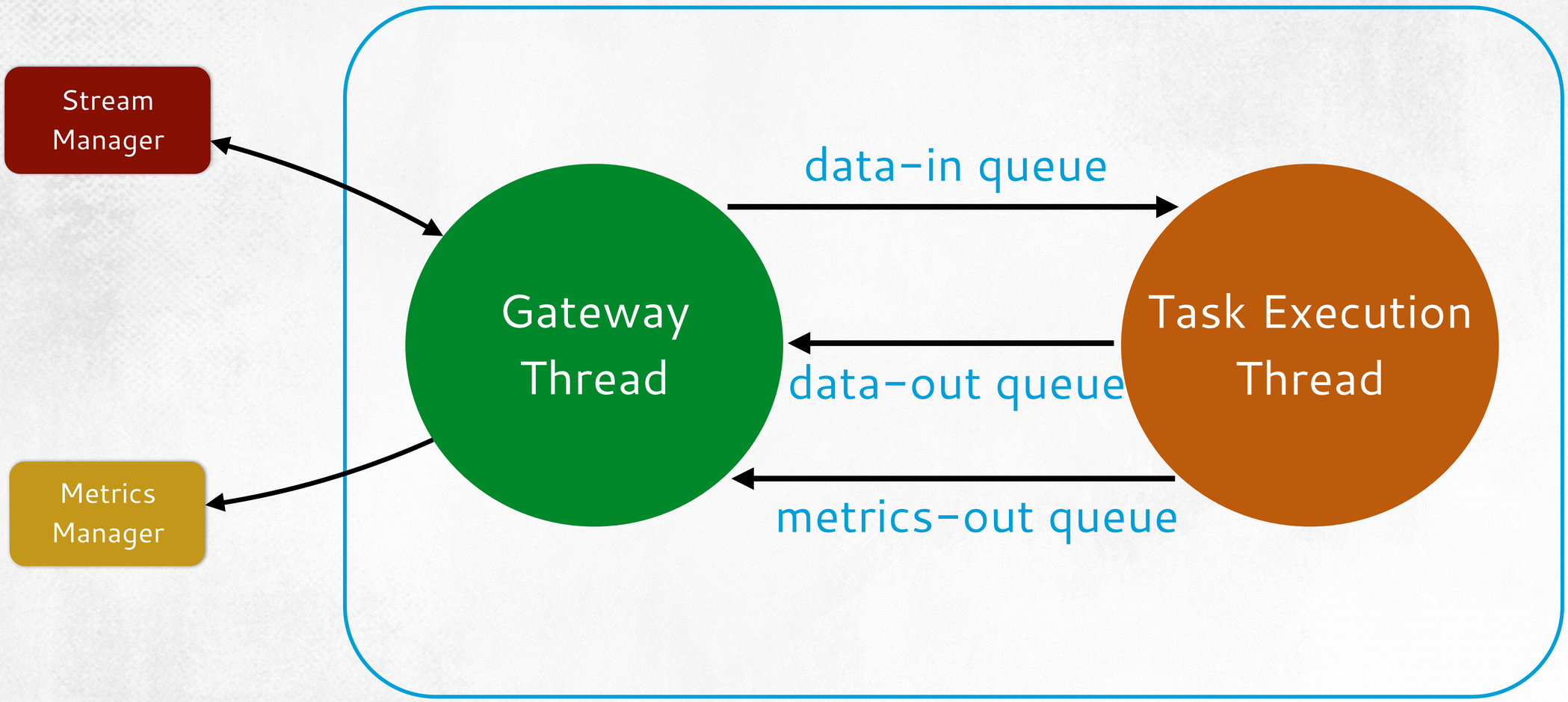
Does the real work!

RUNS ONE TASK

EXPOSES API

COLLECTS METRICS

HERON INSTANCE

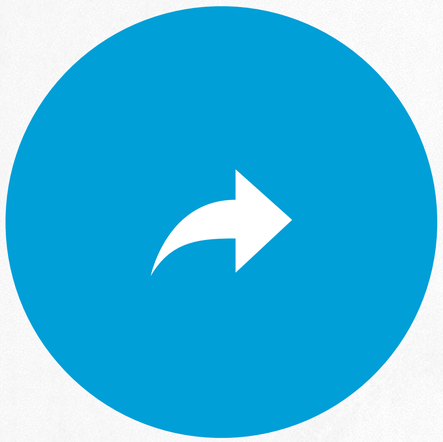


BOUNDED QUEUES - TRIGGERS GC IN LARGE TOPOLOGIES

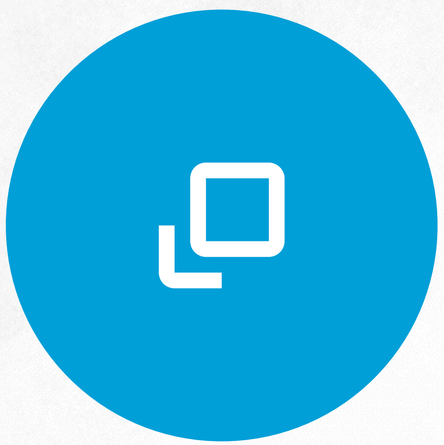
METRICS MANAGER

Optical Nerve

GATHERS METRICS



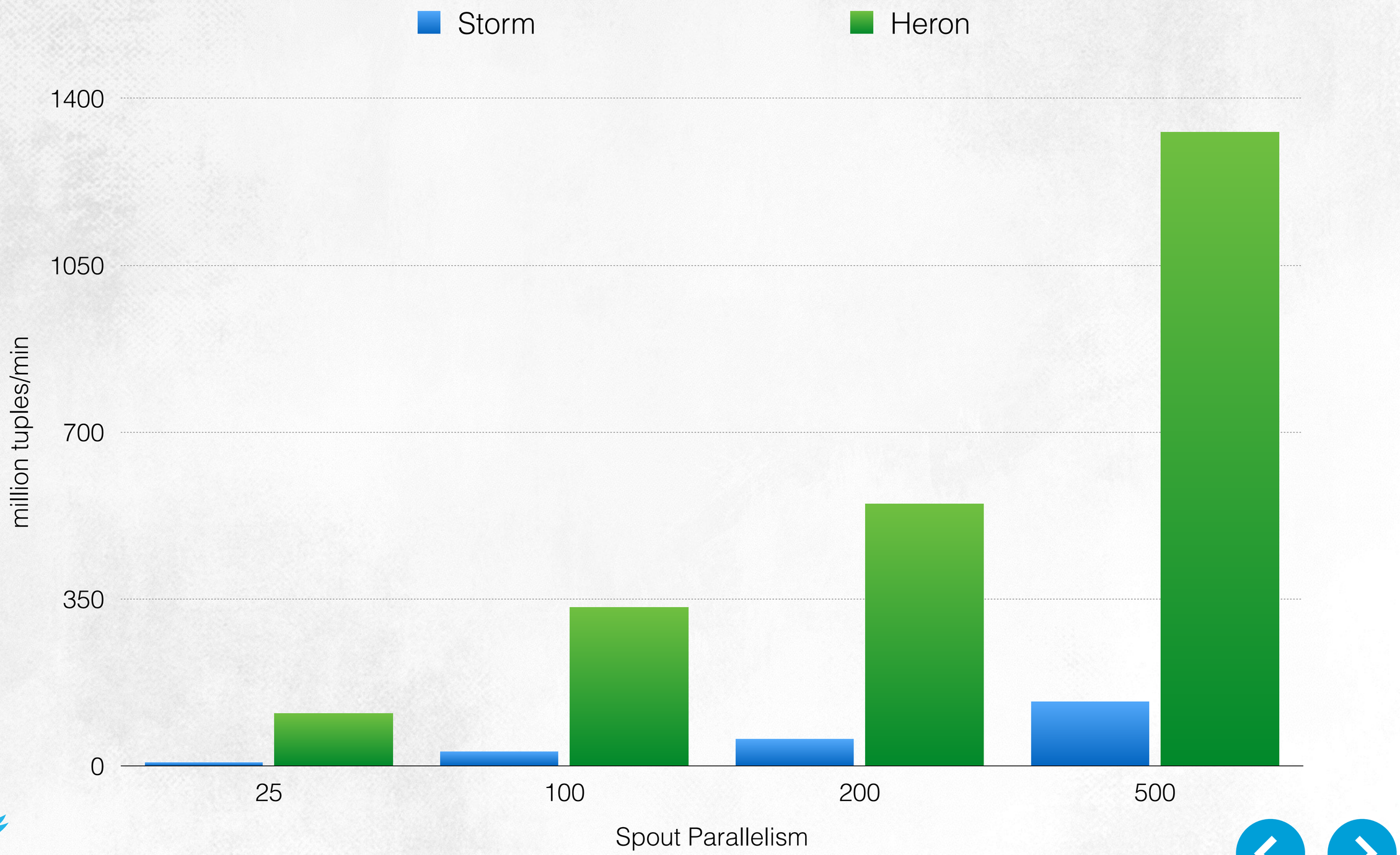
SCRIBES



ABSTRACTED

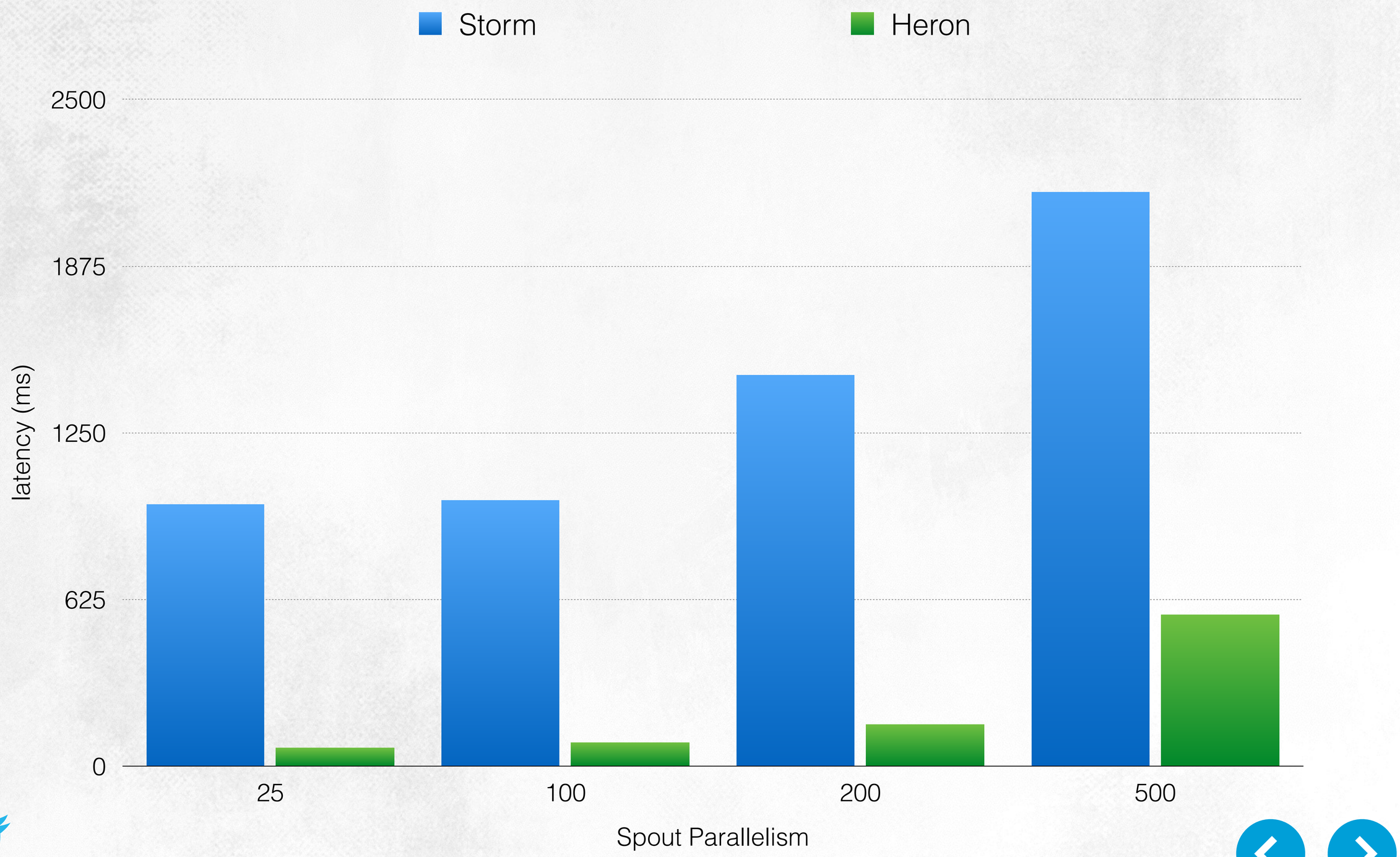
HERON PERFORMANCE

Throughput with acknowledgements - Word count topology



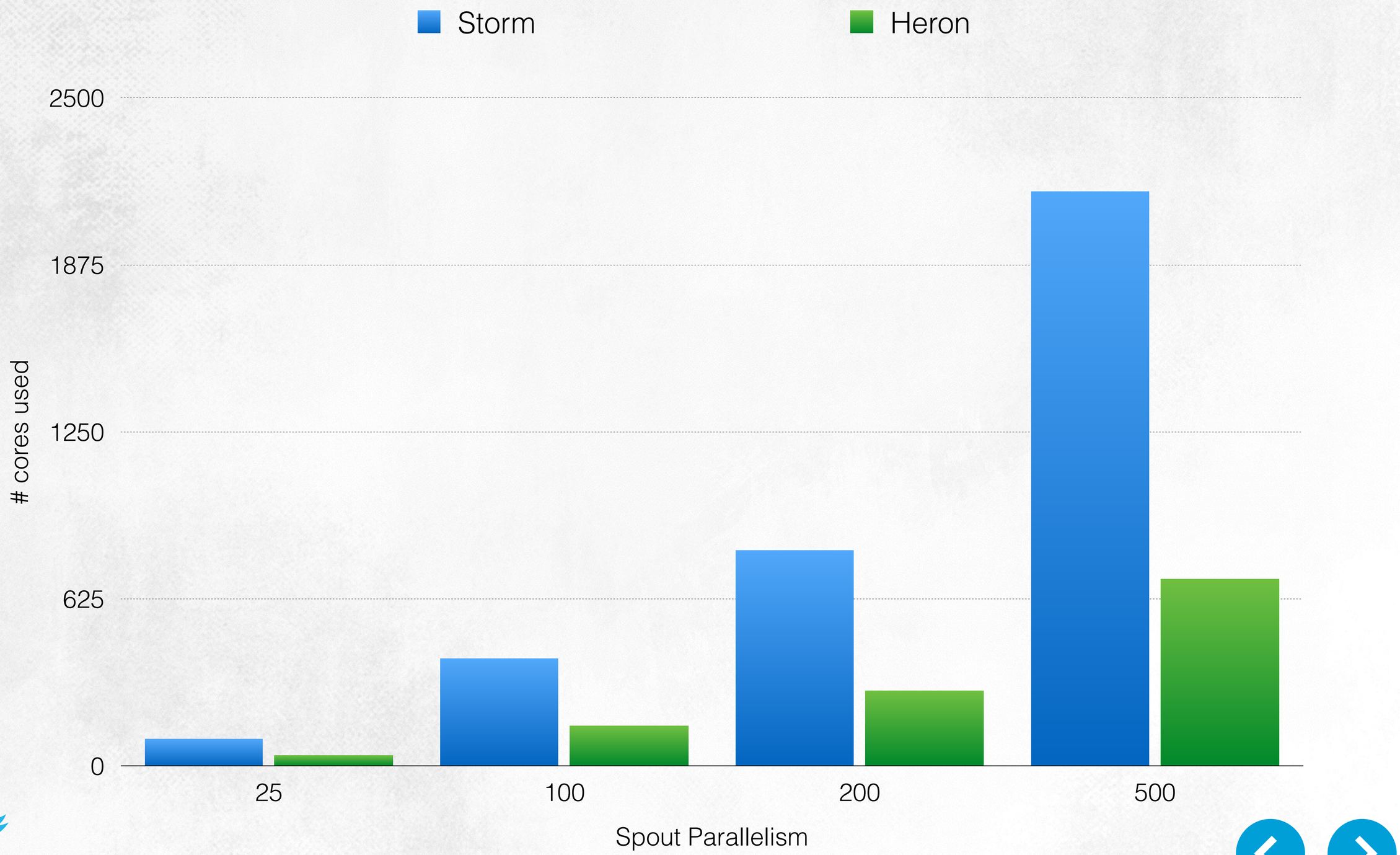
HERON PERFORMANCE

Latency with acknowledgements enabled - Word Count Topology



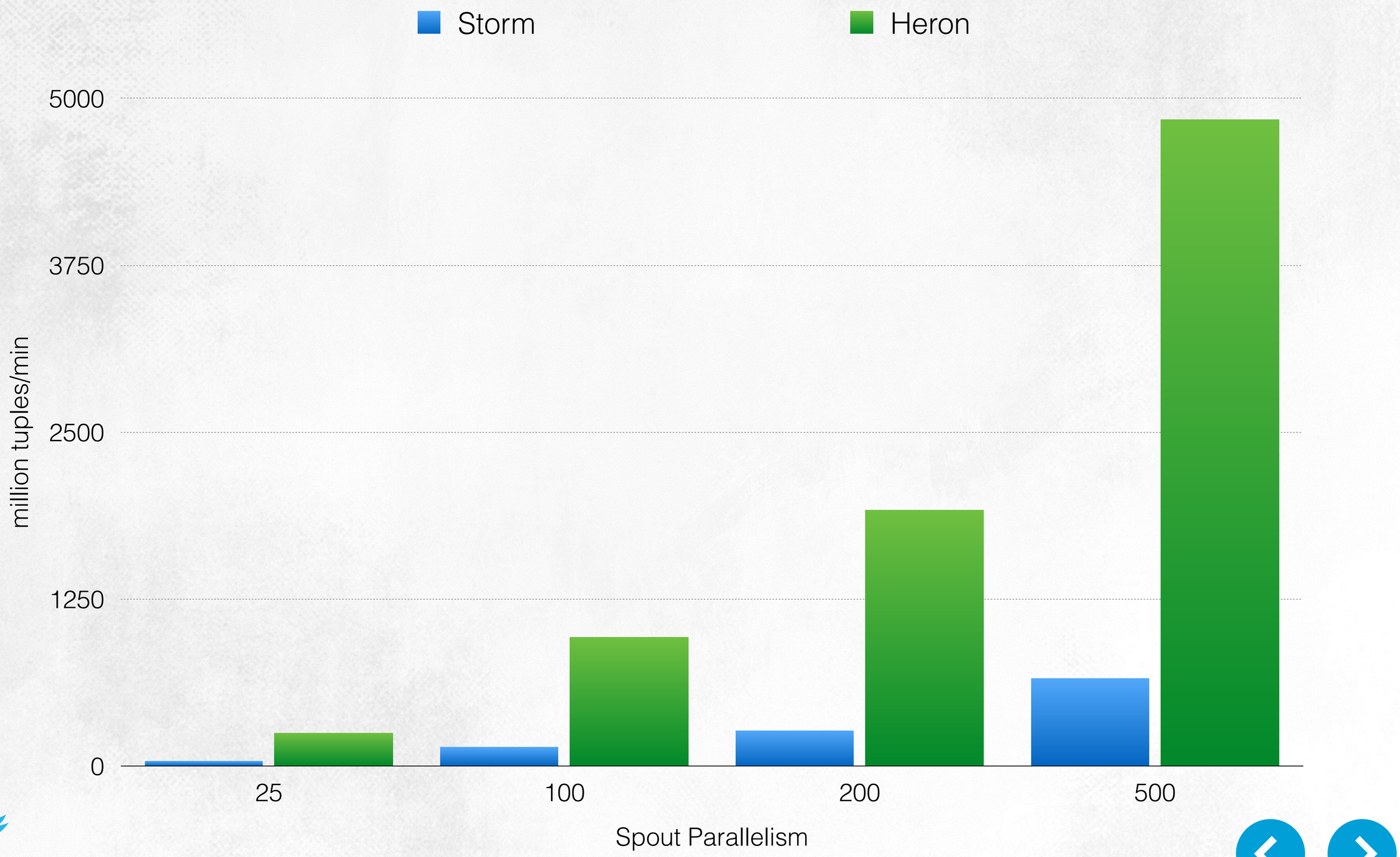
HERON PERFORMANCE

CPU usage with acknowledgements enabled - Word Count Topology



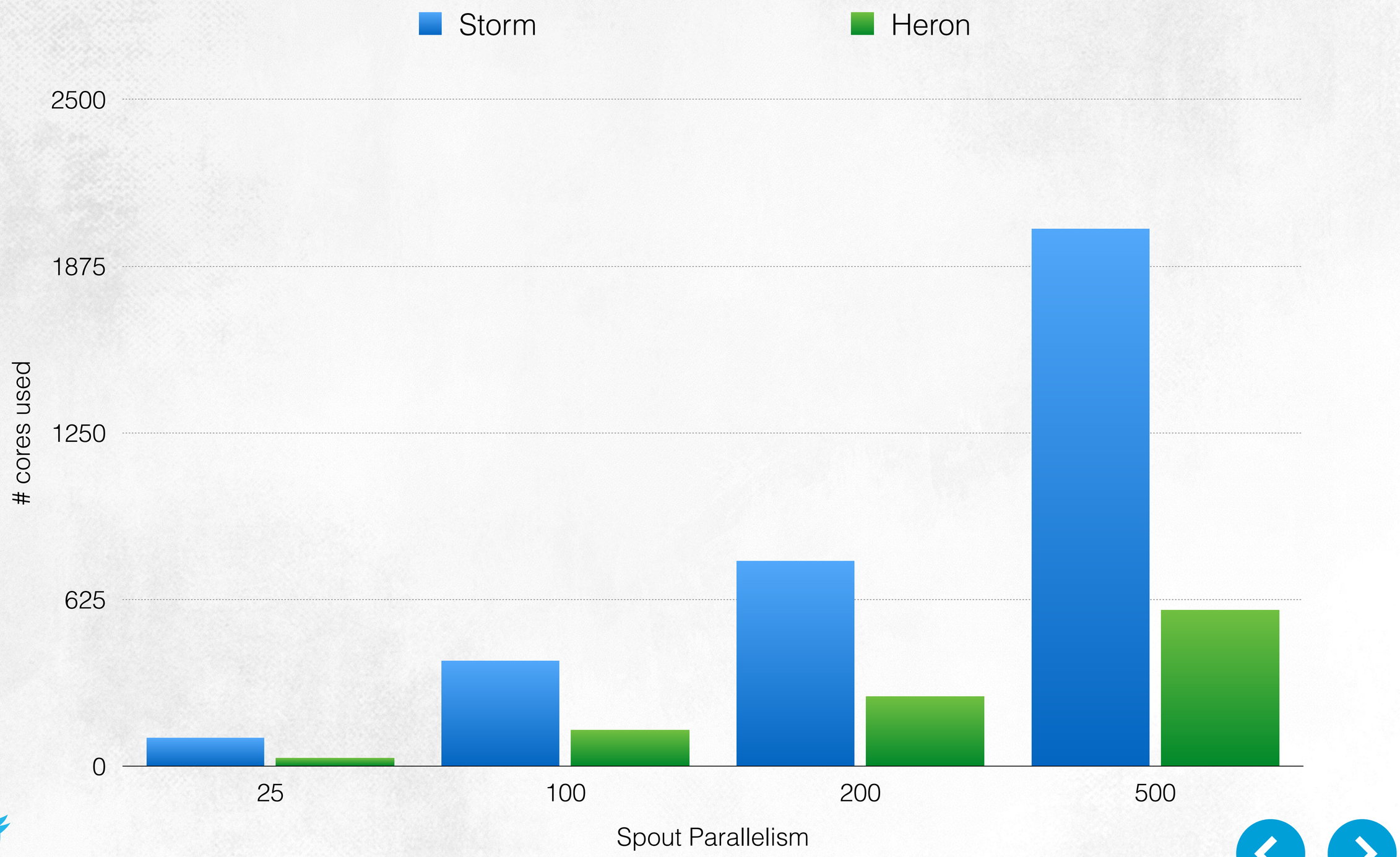
HERON PERFORMANCE

Throughput with no acknowledgements - Word count topology



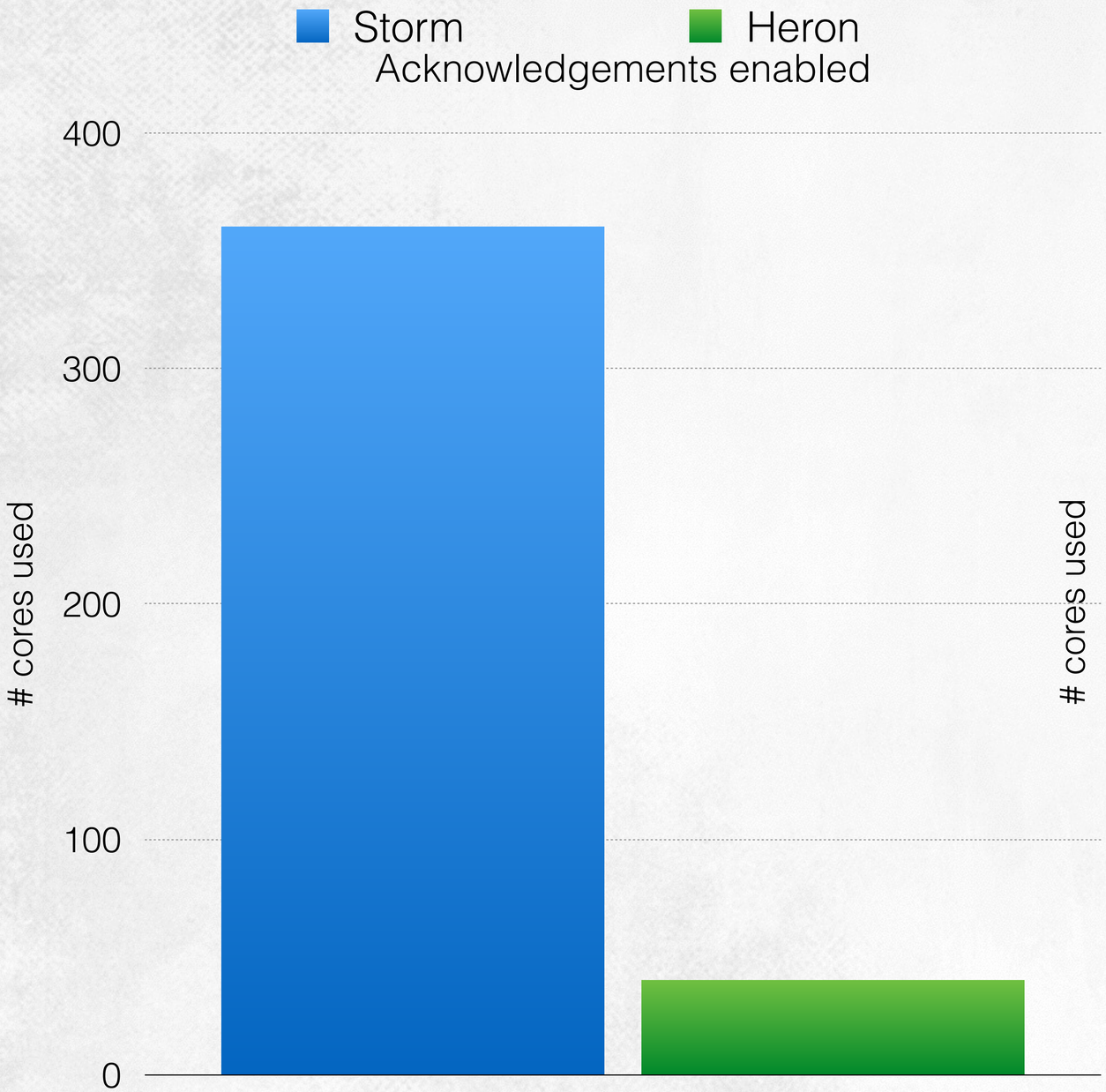
HERON PERFORMANCE

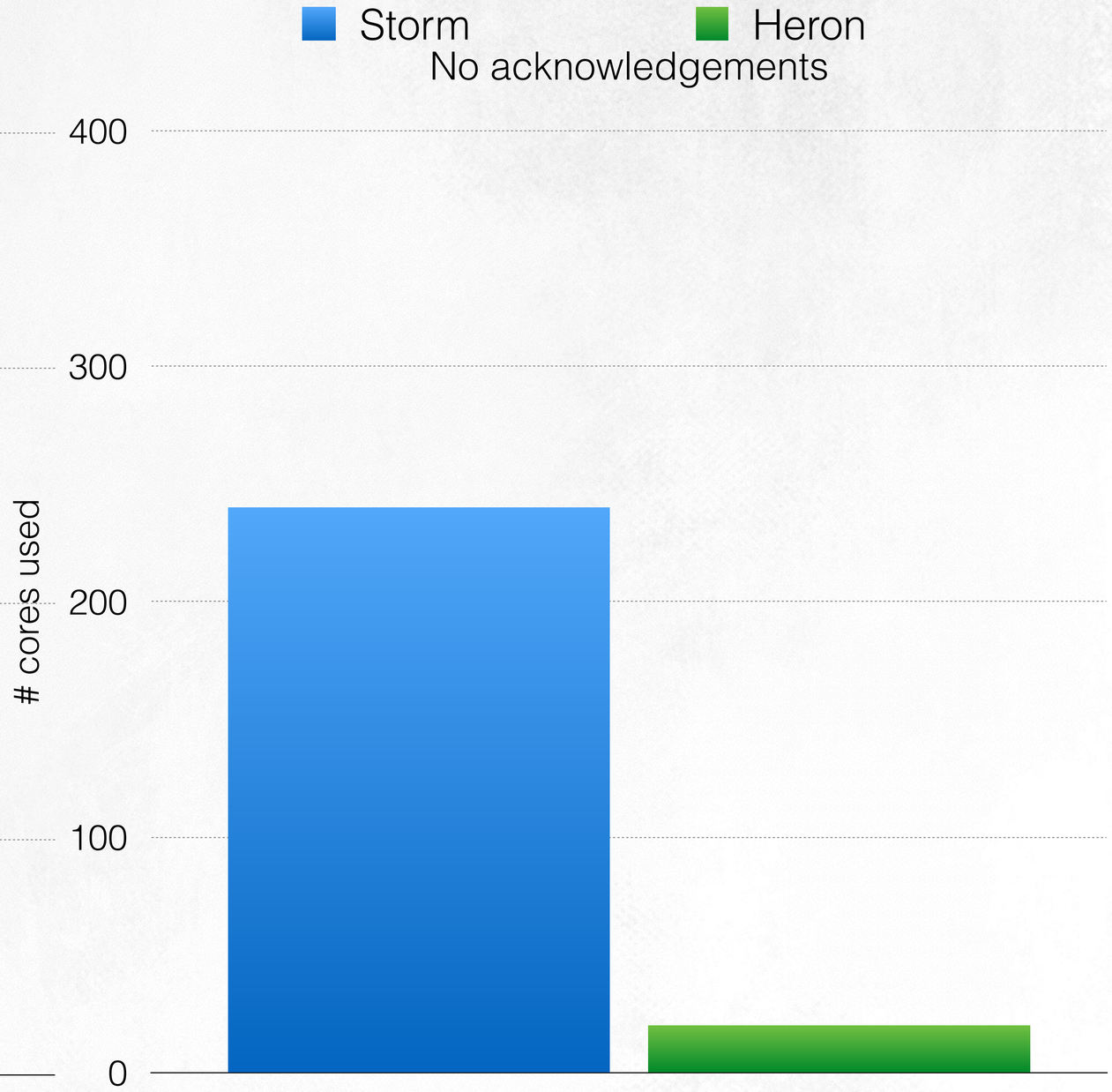
CPU usage with no acknowledgements - Word Count Topology



HERON PERFORMANCE

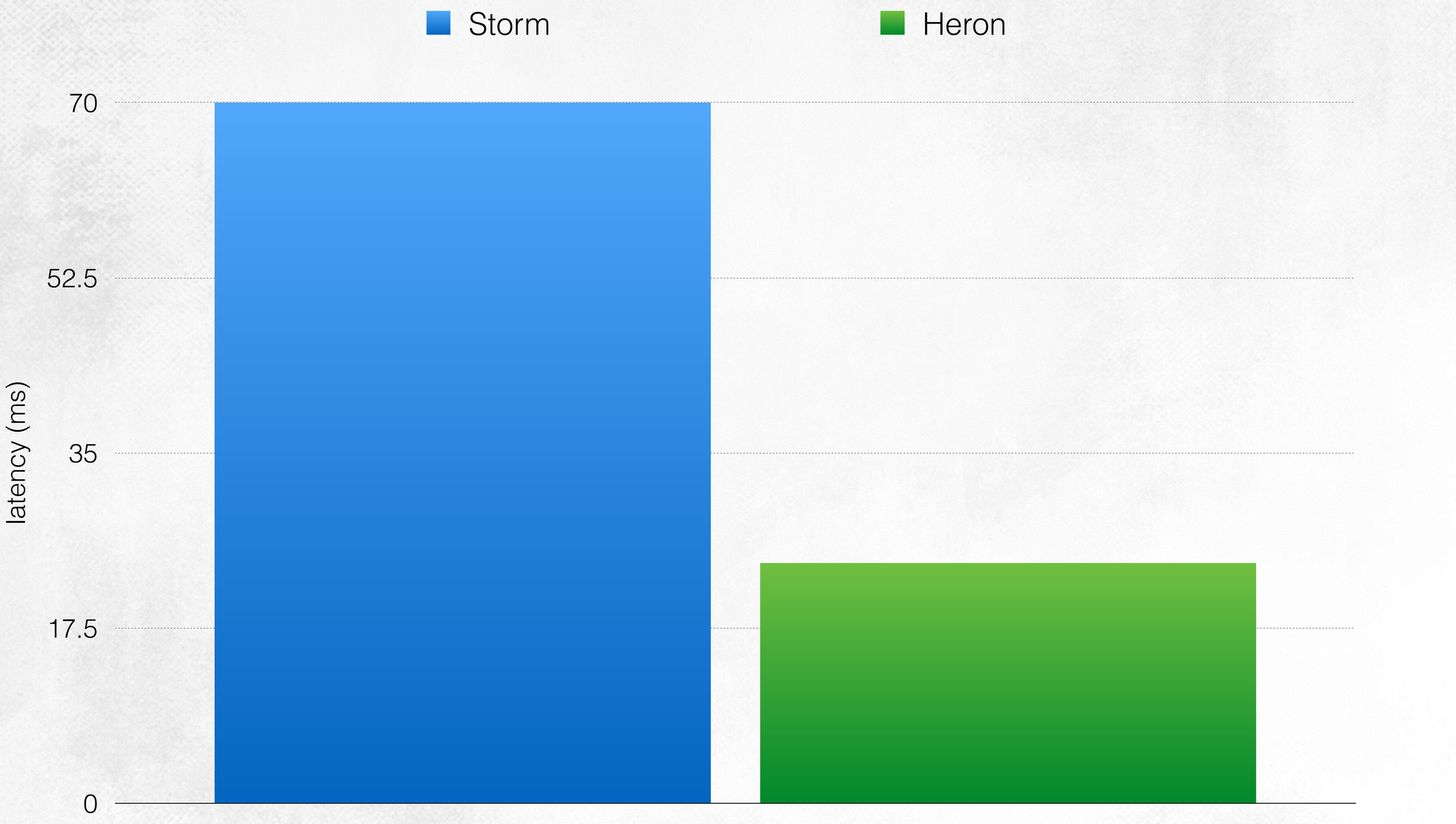
CPU usage – RTAC Topology





HERON PERFORMANCE

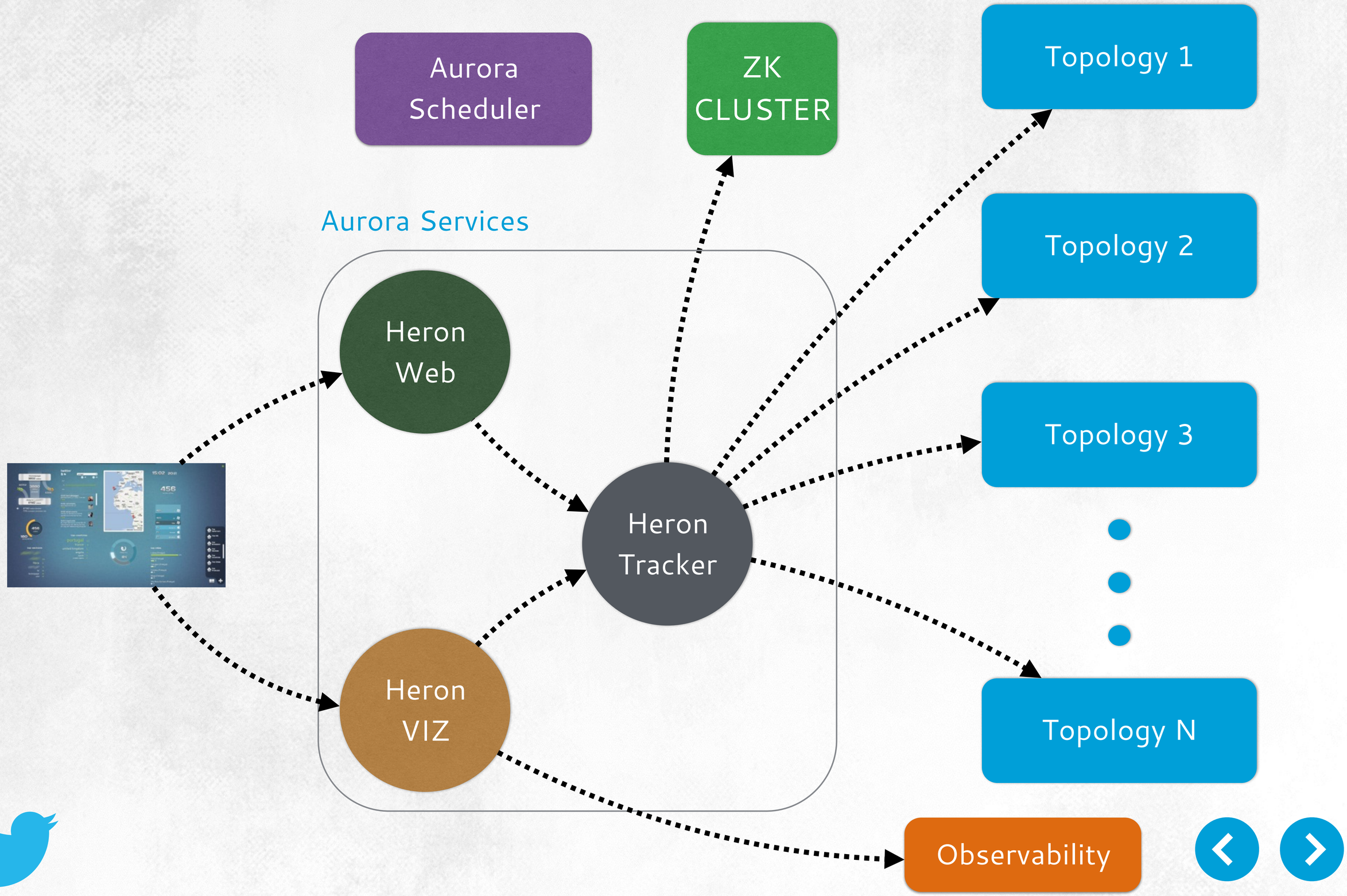
Latency with acknowledgements enabled - RTAC Topology



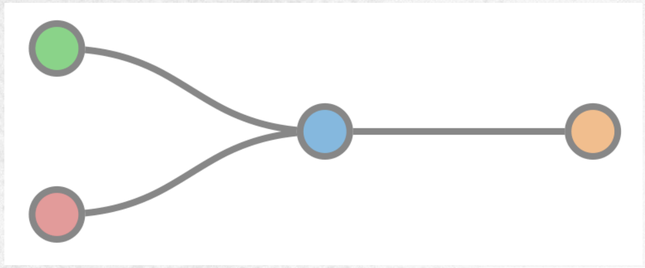
IV

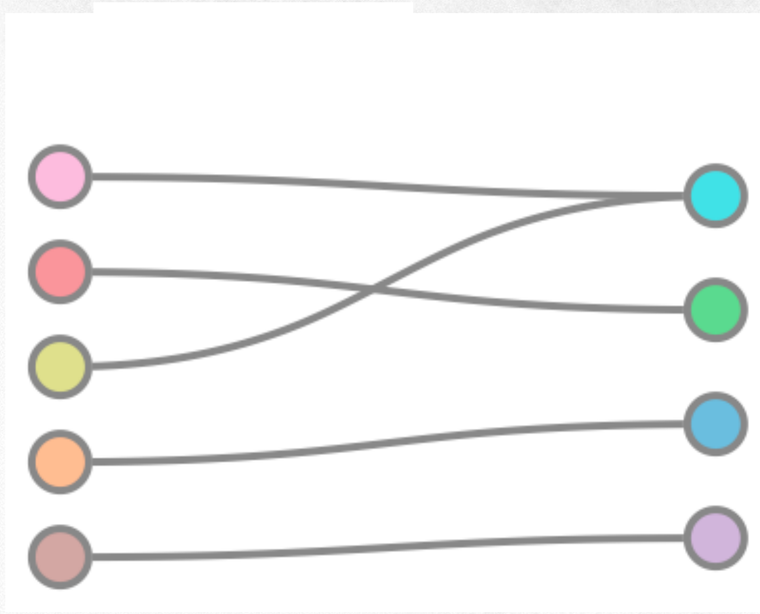
OPERATIONAL EXPERIENCES

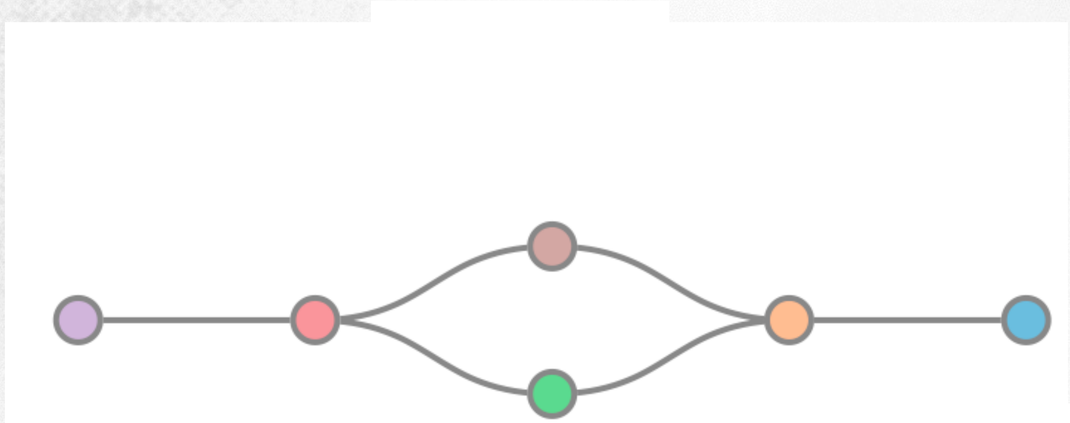
HERON DEPLOYMENT

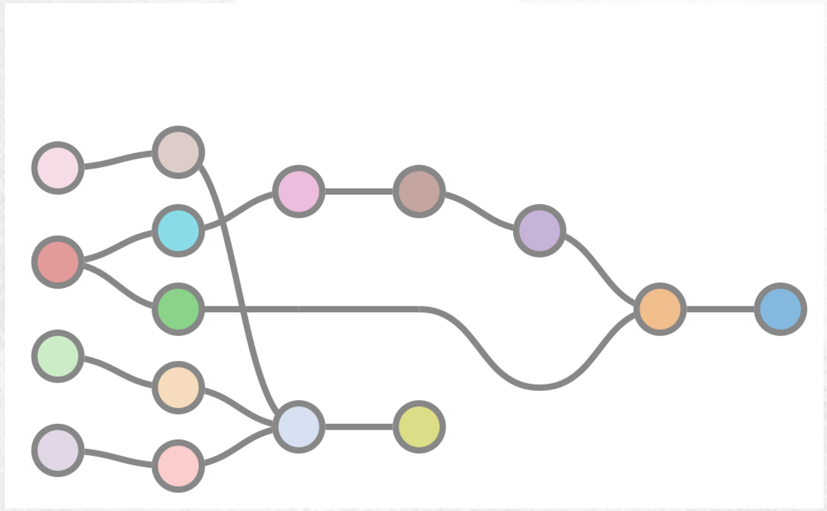


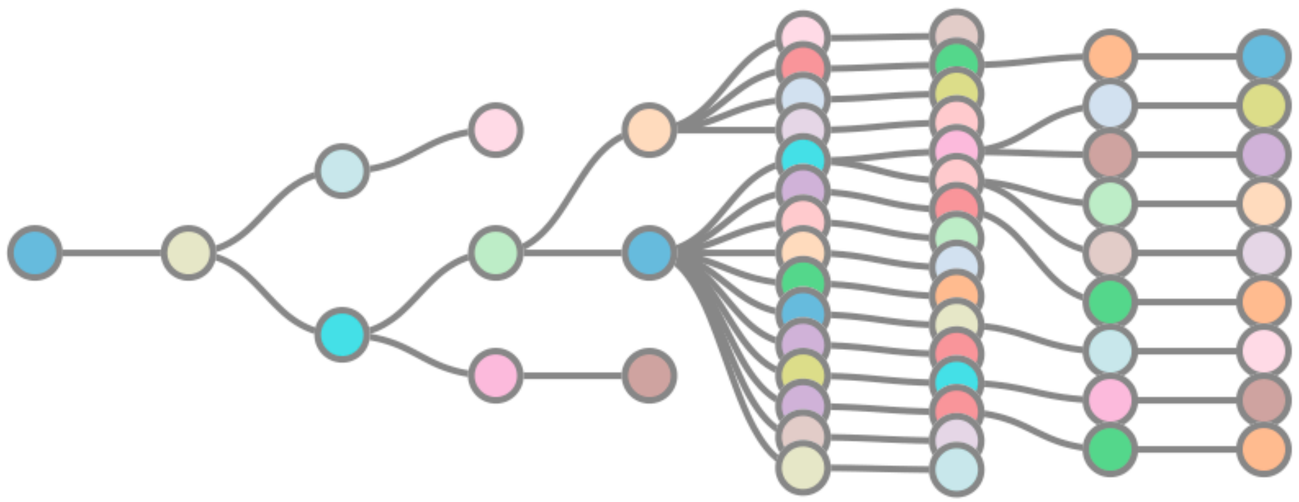
HERON SAMPLE TOPOLOGIES









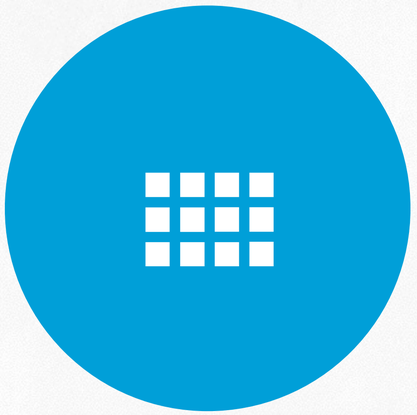


OPERATIONAL EXPERIENCE

SERVICE-LESS

All topologies run under topology owner's role

CLUSTER-LESS



Everything runs on Aurora

TENSION-LESS

No more 2am pages

DEVELOPER EXPERIENCE

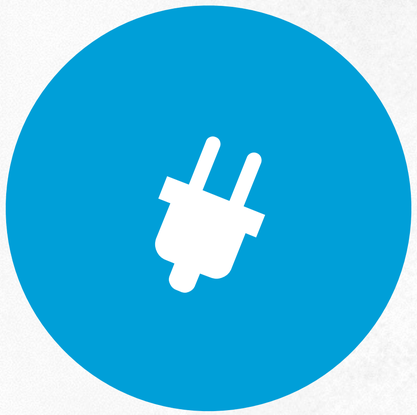
DEBUG

Faster iteration

TUNE

Better resource utilization

DEPLOY



Devel to prod in 5min

MIGRATION EXPERIENCE

SMALL

Couple of hours

MEDIUM

Lots of savings

LARGE

Summingbird tuning takes time

CURRENT WORK

CURRENT WORK

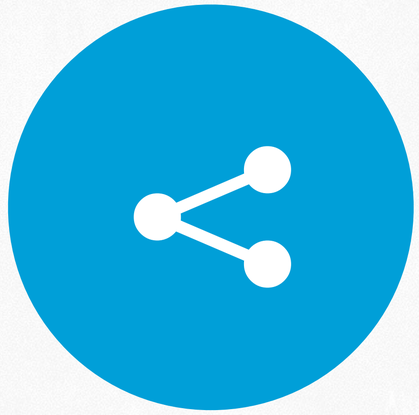
SERIALIZATION

Use Java Reflection

TUNING

Determine optimal set of parameters

ELASTIC



Grow/Shrink based on data

CONFIGURATION

Update topology without restarting

QUESTIONS AND ANSWERS

Go ahead. Ask away.