ONE MILLION CLICKS PER MINUTE

WITH KAFKA AND CLOJURE

Devon Peticolas | @dproi

HOW **NOT** TO PROCESS ONE MILLION CLICKS PER MINUTE

WITH KAFKA AND CLOJURE

Devon Peticolas | @dproi

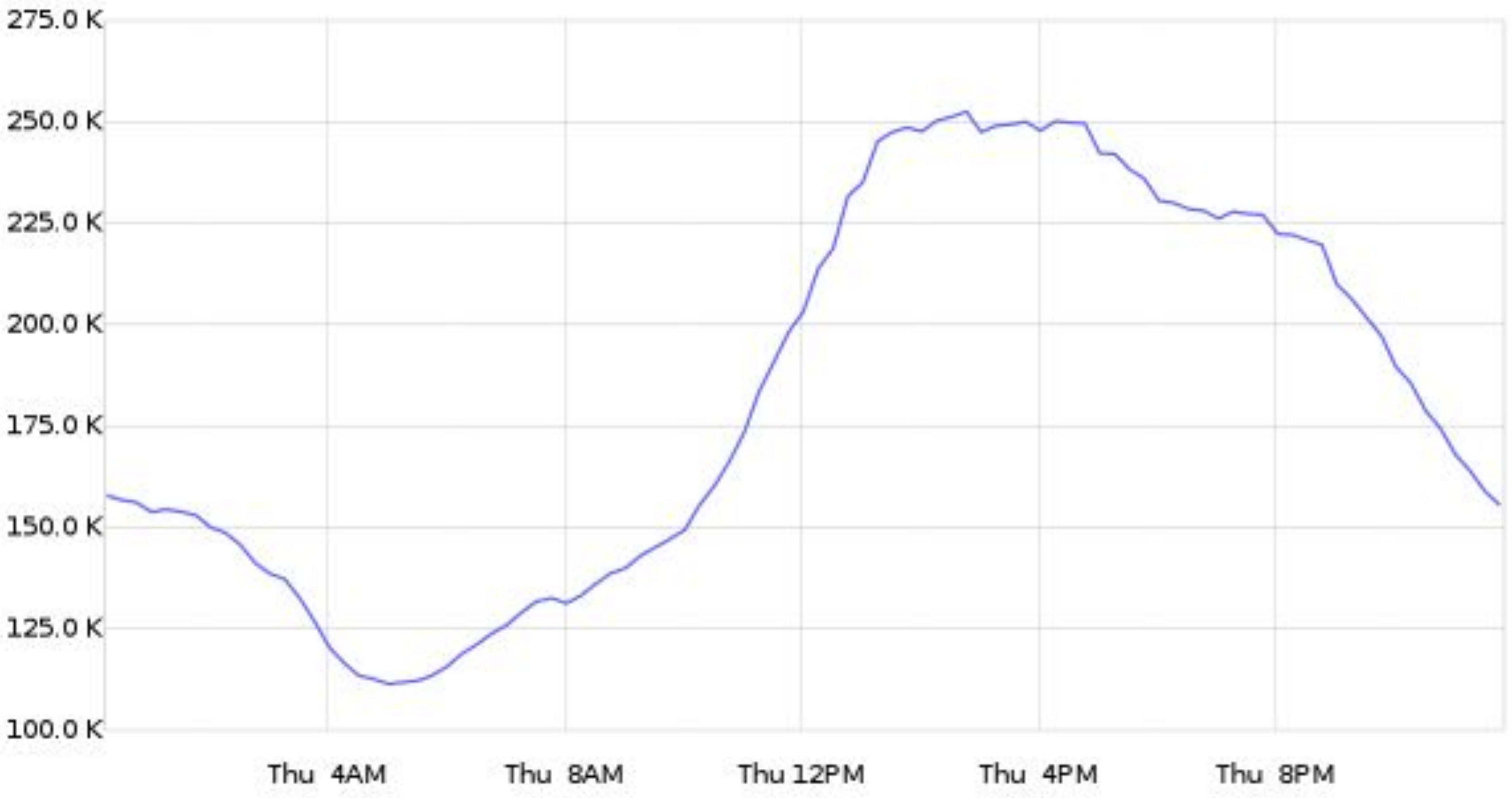
HOW **NOT** TO PROCESS ONE MILLION CLICKS PER MINUTE

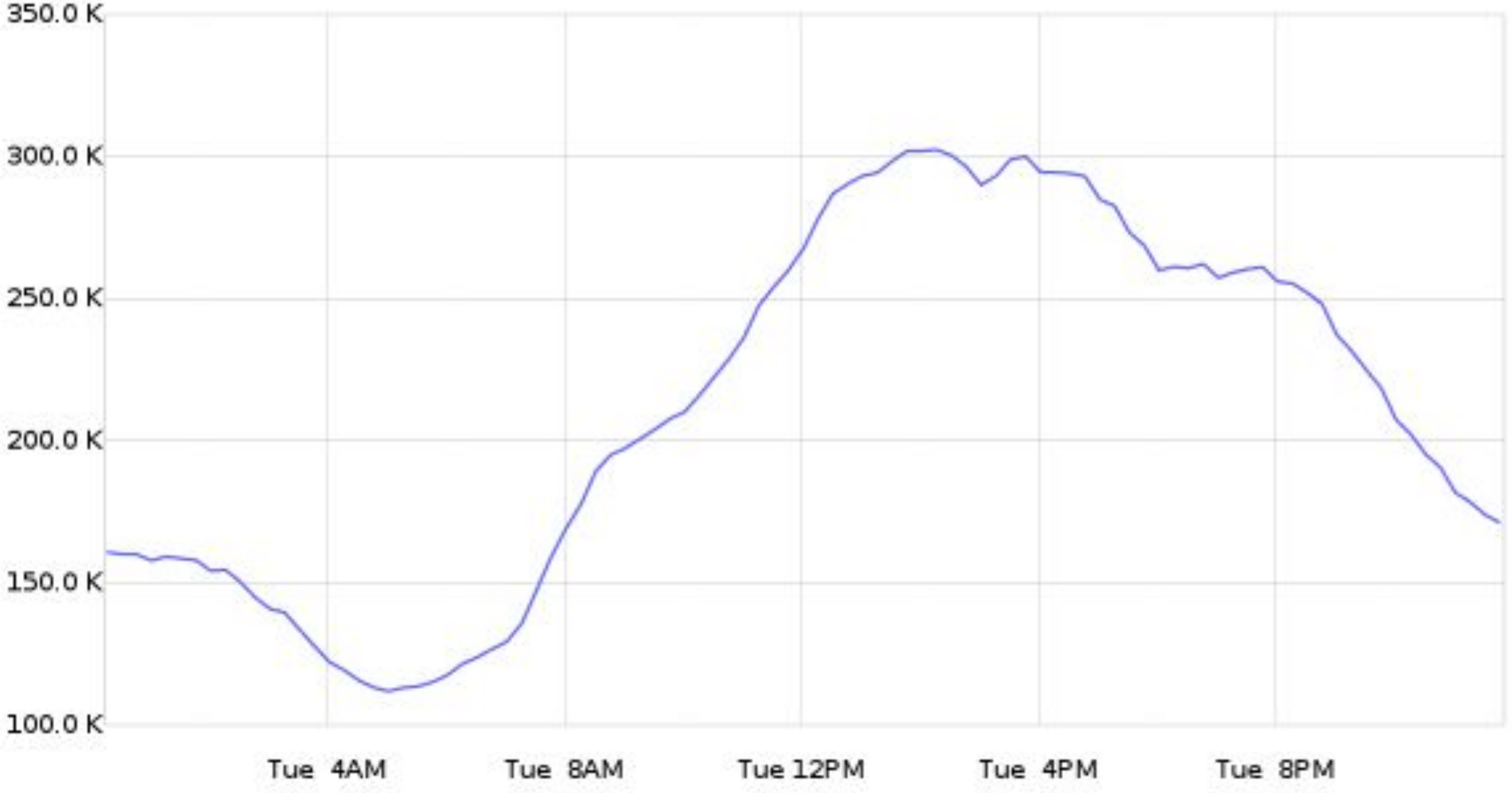
WITH KAFKA AND CLOJURE AND THEN HOW TO

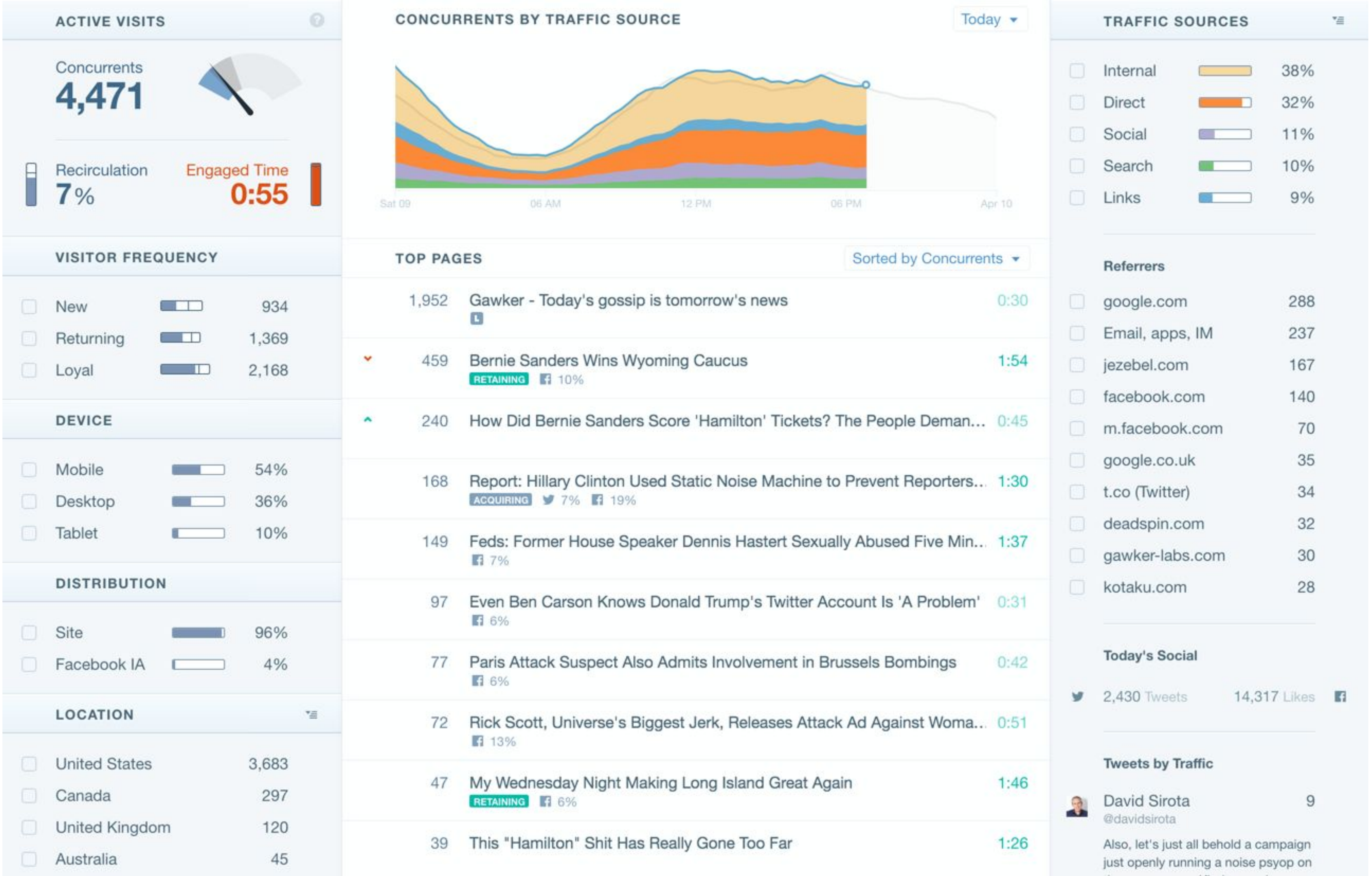
Devon Peticolas | @dproi

github.com/x/slides

Chartbeat


275.0 K
250.0 K
225.0 K
200.0 K
175.0 K
150.0 K
125.0 K
100.0 K
Thu 4AM
Thu 8AM
Thu 12PM
Thu 4PM
Thu 8PM


350.0 K
300.0 K
250.0 K
200.0 K
150.0 K
100.0 K
Tue 4AM
Tue 8AM
Tue 12PM
Tue 4PM
Tue 8PM



ACTIVE VISITS

Concurrents
4,471

Recirculation
7%

Engaged Time
0:55

VISITOR FREQUENCY

New	934
Returning	1,369
Loyal	2,168

DEVICE

Mobile	54%
Desktop	36%
Tablet	10%

DISTRIBUTION

Site	96%
Facebook IA	4%

LOCATION

United States	3,683
Canada	297
United Kingdom	120
Australia	45

CONCURRENTS BY TRAFFIC SOURCE

Today

TOP PAGES

Sorted by Concurrents

1,952	Gawker - Today's gossip is tomorrow's news	0:30
459	Bernie Sanders Wins Wyoming Caucus RETAINING 10%	1:54
240	How Did Bernie Sanders Score 'Hamilton' Tickets? The People Deman...	0:45
168	Report: Hillary Clinton Used Static Noise Machine to Prevent Reporters... ACQUIRING 7% 19%	1:30
149	Feds: Former House Speaker Dennis Hastert Sexually Abused Five Min... 7%	1:37
97	Even Ben Carson Knows Donald Trump's Twitter Account Is 'A Problem' 6%	0:31
77	Paris Attack Suspect Also Admits Involvement in Brussels Bombings 6%	0:42
72	Rick Scott, Universe's Biggest Jerk, Releases Attack Ad Against Woma... 13%	0:51
47	My Wednesday Night Making Long Island Great Again RETAINING 6%	1:46
39	This "Hamilton" Shit Has Really Gone Too Far	1:26

TRAFFIC SOURCES

Internal	38%
Direct	32%
Social	11%
Search	10%
Links	9%

Referrers

google.com	288
Email, apps, IM	237
jezebel.com	167
facebook.com	140
m.facebook.com	70
google.co.uk	35
t.co (Twitter)	34
deadspin.com	32
gawker-labs.com	30
kotaku.com	28

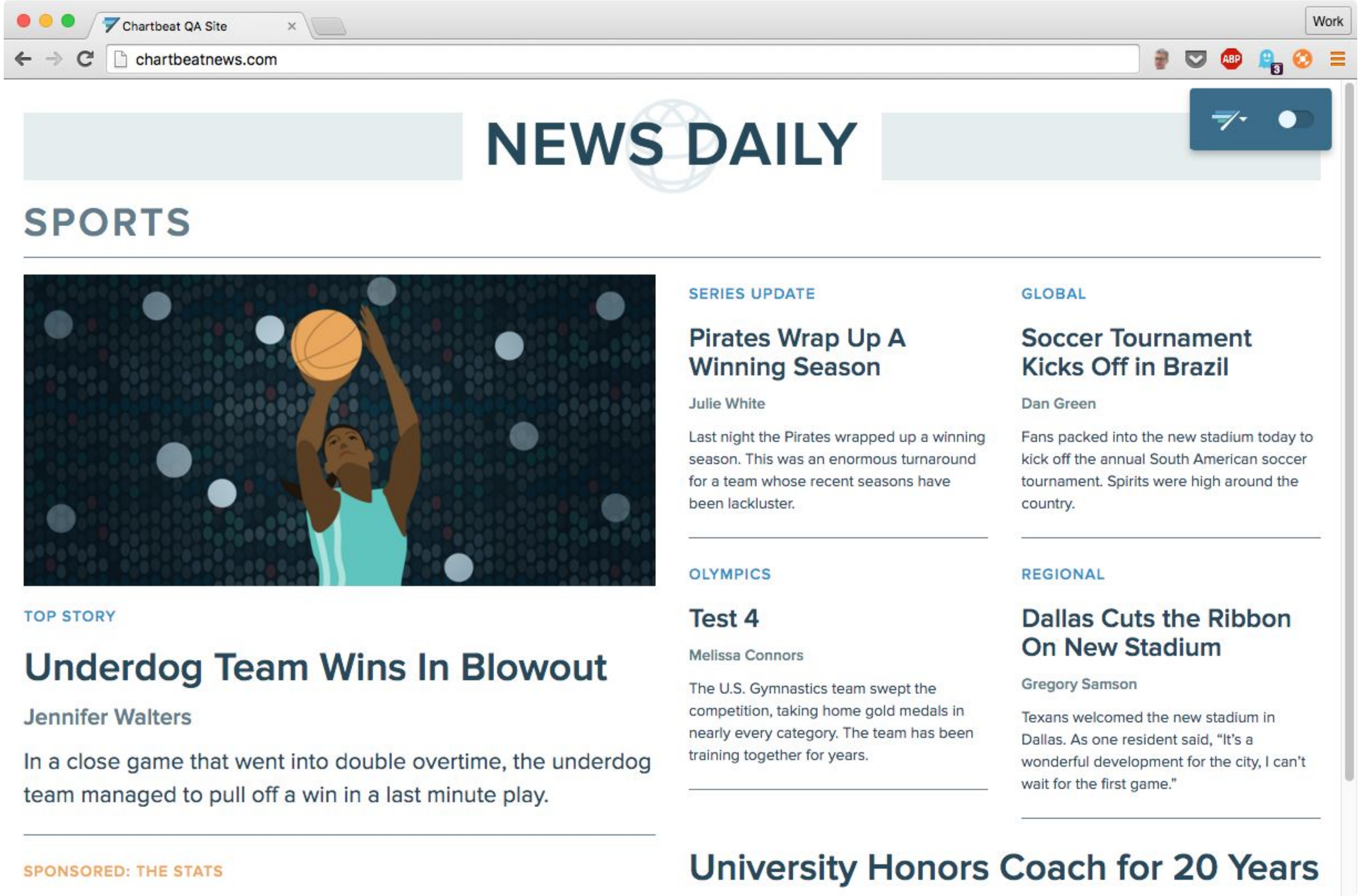
Today's Social

2,430 Tweets 14,317 Likes

Tweets by Traffic

David Sirota 9
@davidsirota

Also, let's just all behold a campaign just openly running a noise psyop on



NEWS DAILY

SPORTS

TOP STORY

Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

SPONSORED: THE STATS

SERIES UPDATE

Pirates Wrap Up A Winning Season

Julie White

Last night the Pirates wrapped up a winning season. This was an enormous turnaround for a team whose recent seasons have been lackluster.

GLOBAL

Soccer Tournament Kicks Off in Brazil

Dan Green

Fans packed into the new stadium today to kick off the annual South American soccer tournament. Spirits were high around the country.

OLYMPICS

Test 4

Melissa Connors

The U.S. Gymnastics team swept the competition, taking home gold medals in nearly every category. The team has been training together for years.

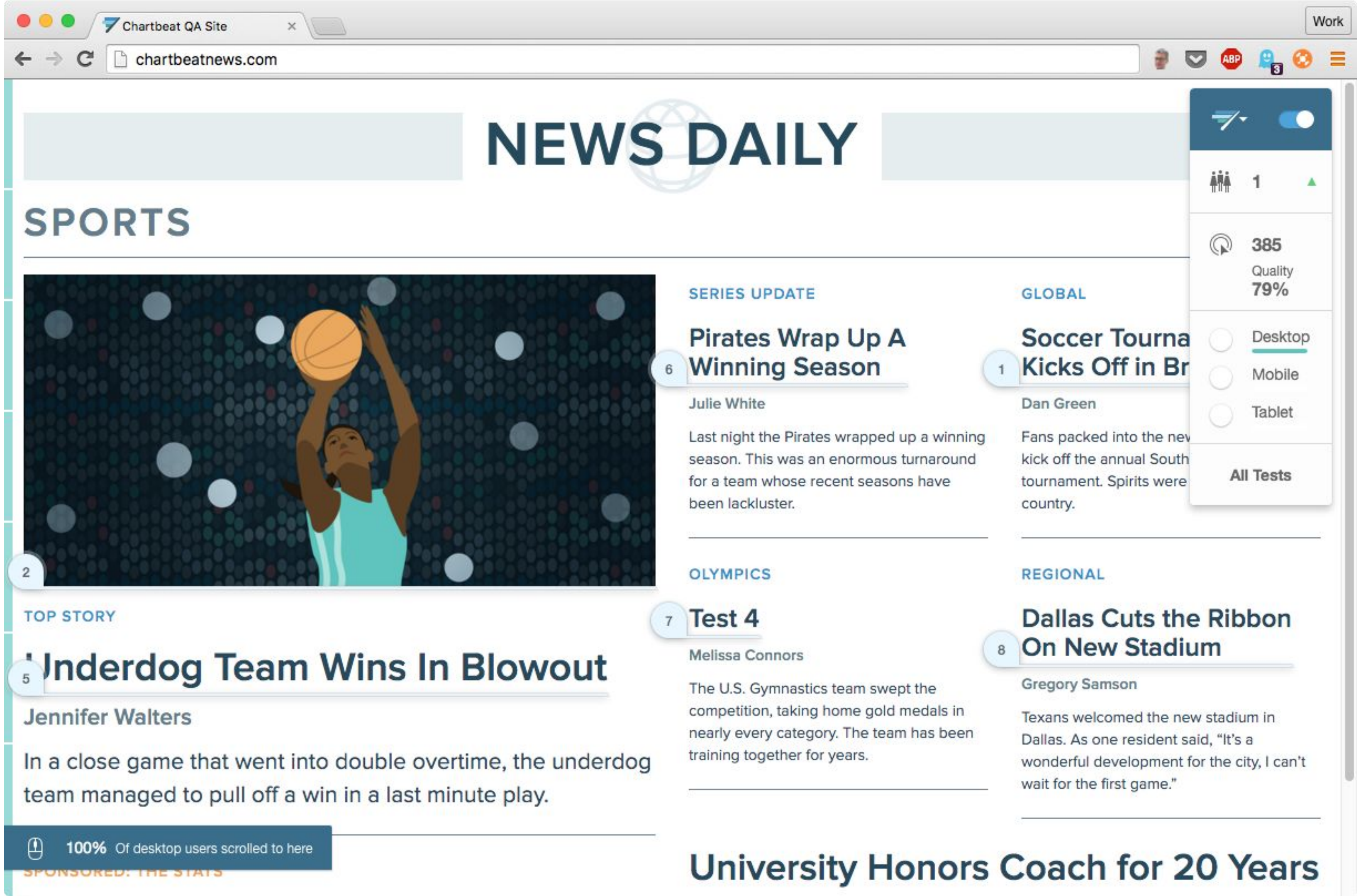
REGIONAL

Dallas Cuts the Ribbon On New Stadium

Gregory Samson

Texans welcomed the new stadium in Dallas. As one resident said, "It's a wonderful development for the city, I can't wait for the first game."

University Honors Coach for 20 Years


Chartbeat QA Site

chartbeatnews.com

Work

NEWS DAILY

SPORTS

1

385
Quality
79%

Desktop
Mobile
Tablet

All Tests

2

TOP STORY

5 Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

100% Of desktop users scrolled to here

SPONSORED: THE STATS

SERIES UPDATE

6 Pirates Wrap Up A Winning Season

Julie White

Last night the Pirates wrapped up a winning season. This was an enormous turnaround for a team whose recent seasons have been lackluster.

GLOBAL

1 Soccer Tourna
Kicks Off in Br

Dan Green

Fans packed into the nev
kick off the annual South
tournament. Spirits were
country.

OLYMPICS

7 Test 4

Melissa Connors

The U.S. Gymnastics team swept the competition, taking home gold medals in nearly every category. The team has been training together for years.

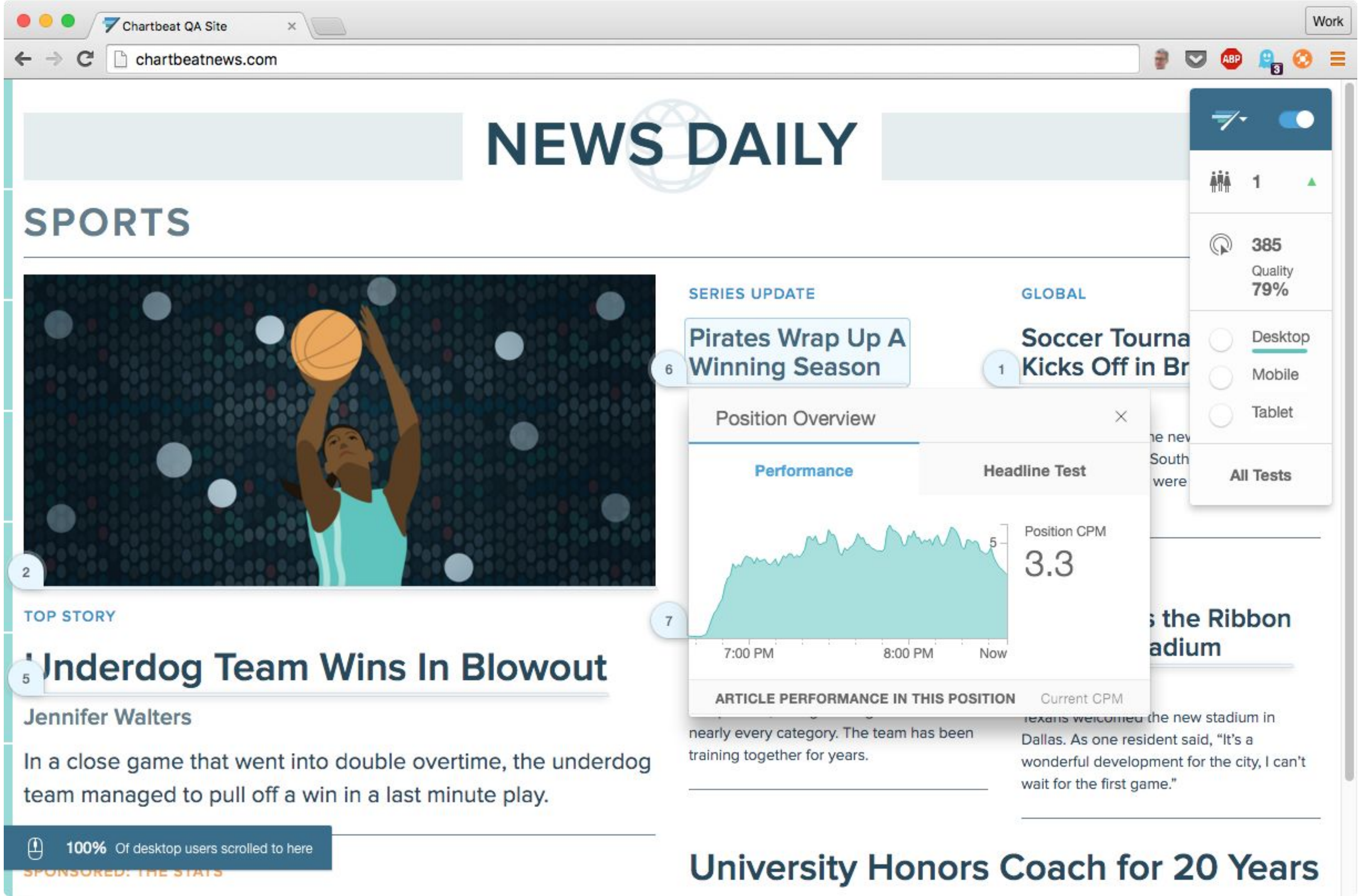
REGIONAL

8 Dallas Cuts the Ribbon On New Stadium

Gregory Samson

Texans welcomed the new stadium in Dallas. As one resident said, “It’s a wonderful development for the city, I can’t wait for the first game.”

University Honors Coach for 20 Years


Chartbeat QA Site

chartbeatnews.com

Work

NEWS DAILY

SPORTS

SERIES UPDATE

6 Pirates Wrap Up A Winning Season

GLOBAL

1 Soccer Tourna Kicks Off in Br

2

TOP STORY

5 Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

Position Overview

Performance | Headline Test

Position CPM

3.3

7

ARTICLE PERFORMANCE IN THIS POSITION Current CPM

nearly every category. The team has been training together for years.

s the Ribbon adium

Texans welcomed the new stadium in Dallas. As one resident said, "It's a wonderful development for the city, I can't wait for the first game."

he new South were

1

385

Quality 79%

Desktop

Mobile

Tablet

All Tests

100% Of desktop users scrolled to here

SPONSORED: THE STATS

University Honors Coach for 20 Years

The User Story

As a…

Homepage Editor

I want…

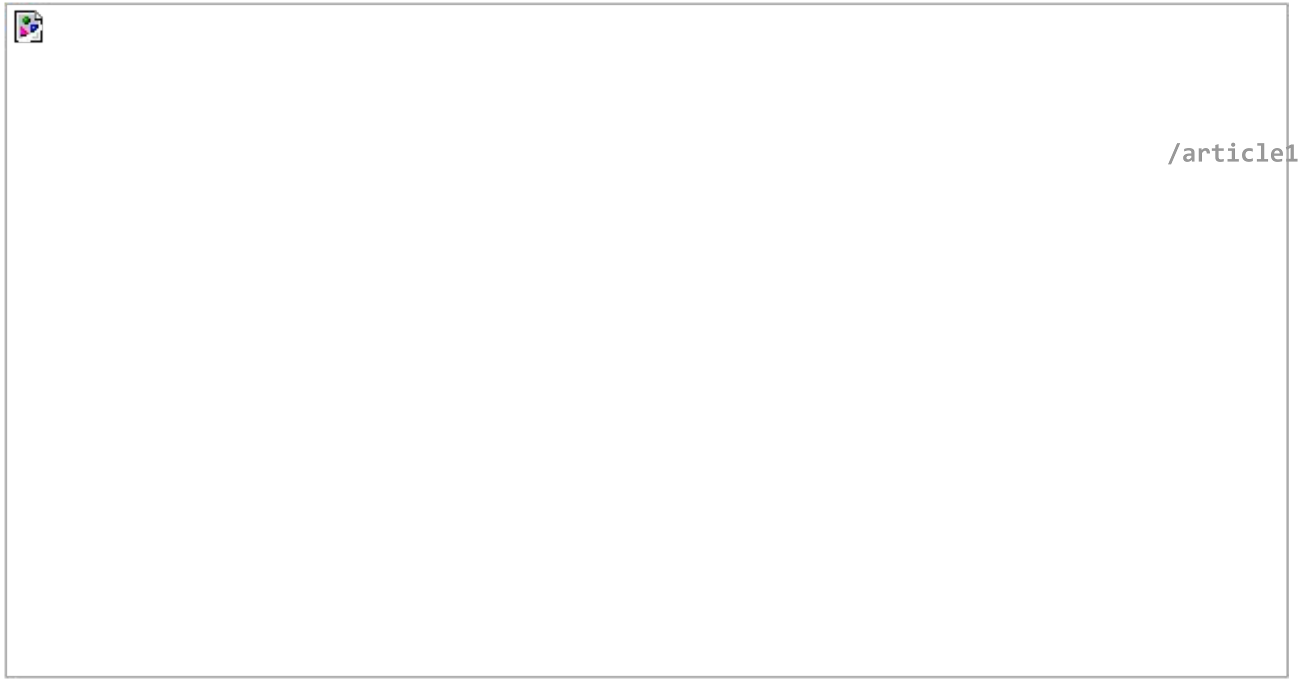
1. Realtime clicks per minute of each article on my homepage.
2. Historical graph of clicks of each article on my homepage.

So that…

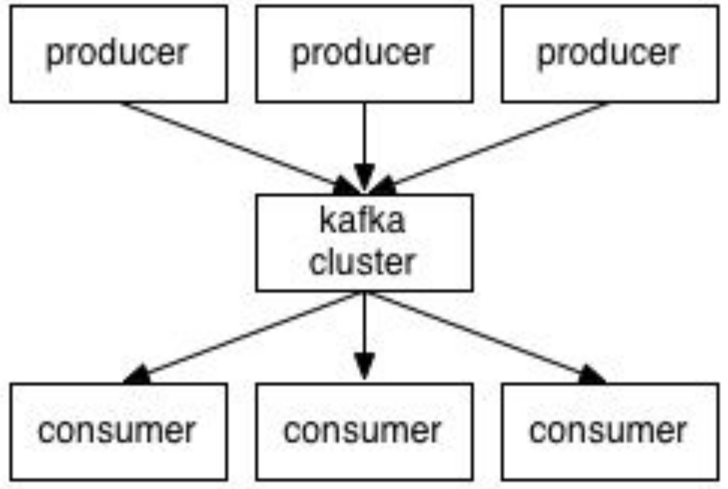
I can make informed decisions about my content on my homepage.

Product Owner

Pinging The “Click”



Apache Kafka



A Message Broker with “Consumers” and “Producers”

- Fast
- High-Throughput
- Fault-Tolerant

- Maintains order (within a partition)
- Data is replayable (within some window)

```clojure
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

```clojure
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

```clojure
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

```clojure
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

```clojure
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

```
(require '[clj-kafka.consumer.zk :refer [consumer messages]])

(defn increment-in-db [src trg min amount]
  ...)

(defn get-minute [ping]
  (- (:ts ping) (mod (:ts ping) 60)))

(defn first? [ping]
  (= 0 (:time-on-page ping)))

(defn unpack [msg]
  (msgpack.core/unpack (.message msg)))

(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

The dangers of an infinite lazy seq

```clojure
(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

$\}$ Holding onto head

#1

The dangers of an infinite lazy seq

```
user=> (def counter (atom 0))
user=> (defn lazy-nums [] (lazy-seq (cons (swap! counter inc) (lazy-nums))))

user=> (doseq [n (take 4 (lazy-nums))] (println n))
1
2
3
4
nil
user=> (doseq [n (take 4 (lazy-nums))] (println n))
5
6
7
8
nil
```

#2

The dangers of an infinite lazy seq

```
user=> (def counter (atom 0))
user=> (defn lazy-nums [] (lazy-seq (cons (swap! counter inc) (lazy-nums))))
user=> (def my-lazy-nums (lazy-nums))
user=> (doseq [n (take 4 my-lazy-nums)] (println n))
1
2
3
4
nil
user=> (doseq [n (take 4 my-lazy-nums)] (println n))
1
2
3
4
nil
```

#3

The dangers of an infinite lazy seq

```
(let [msgs (messages (consumer {...}) "pings")]
  (doseq [msg msgs]
    (let [ping (unpack msg)]
      (when (first? ping)
        (let [src (:http-refer ping)
              trg (:path ping)
              min (get-minute ping)]
          (increment-in-db src trg min 1))))))
```

#1

The dangers of an infinite lazy seq

```
(doseq [msg (messages (consumer {...}) "pings")]
  (let [ping (unpack msg)]
    (when (first? ping)
      (let [src (:http-refer ping)
            trg (:path ping)
            min (get-minute ping)]
        (increment-in-db src trg min 1)))))
```

No more `let [msgs (messages`
Not holding onto head

#4

SLOW

```
(doseq [msg (messages (consumer {...}) "pings")]
  (let [ping (unpack msg)]
    (when (first? ping)
      (let [src (:http-refer ping)
            trg (:path ping)
            min (get-minute ping)]
        (increment-in-db src trg min 1)))))
```

#4

Batching and Grouping

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                          first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5

Batching and Grouping

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                         first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5

Batching and Grouping

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                         first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5

Batching and Grouping

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                         first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5

Batching and Grouping

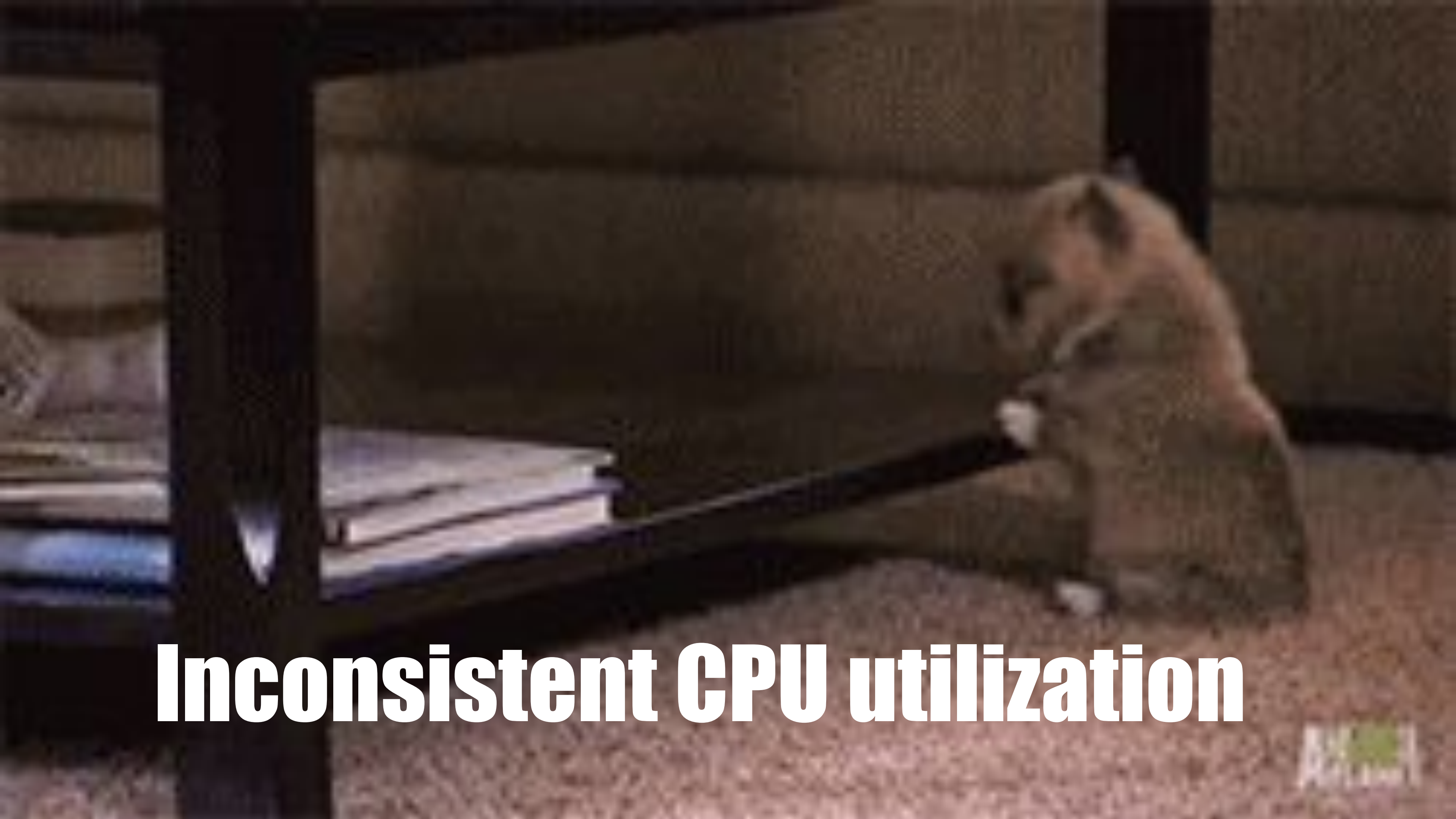
```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                         first-pings)]
      (increment-in-db src trg min (count ps)))))
```

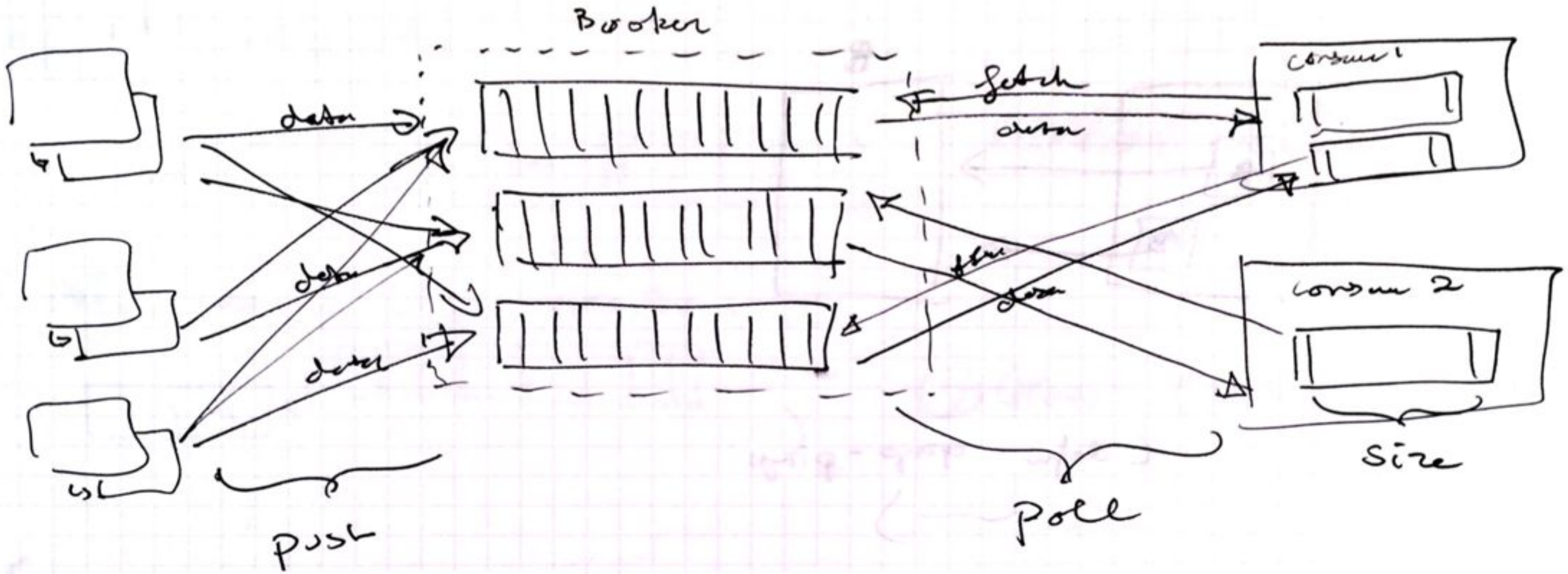
#5

Batching and Grouping

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                          first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5


Inconsistent CPU utilization



Broker
data
data
data
fetch
data
fetch
data
consumer 1
consumer 2
size
push
poll

Not lockstepping network and CPU

```
(doseq [msg-batch (partition 5000 (messages (consumer {...}) "pings"))]
  (let [pings (map unpack msg-batch)
        first-pings (filter first? pings)]
    (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                         first-pings)]
      (increment-in-db src trg min (count ps)))))
```

#5

Not lockstepping network and CPU

```
(let [stream (create-message-stream (consumer {...}) "pings")]
  (loop []
    (let [msg-batch (take 5000 stream)
          pings (map unpack msg-batch)
          first-pings (filter first? pings)]
      (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                           first-pings)]
        (increment-in-db src trg min (count ps)))
      (recur))))
```

#6

Wrapping the KafkaStream in a channel

```
(require '[clojure.core.async :refer [chan onto-chan go-loop <!]])

(let [stream (create-message-stream (consumer {...}) "pings")
      msg-chan (chan 1 (partition-all 5000))]
  (onto-chan msg-chan stream)
  (go-loop []
    (let [msg-batch (<! msg-chan)]
          pings (map unpack msg-batch)
          first-pings (filter first? pings)]
      (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                                       first-pings)]
        (increment-in-db src trg min (count ps)))
      (recur))))
```

#8

Multithreading

```
(let [streams (create-message-streams (consumer {...}) {"pings" 4})]
  (doseq [stream streams]
    (async/thread
      (loop []
        (let [msg-batch (take 5000 stream)
              pings (map unpack msg-batch)
              first-pings (filter first? pings)]
          (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                               first-pings)]
            (increment-in-db src trg min (count ps)))
             (recur))))))
```

#9

partitions = processes * KafkaStreams * num.consumer.fetchers

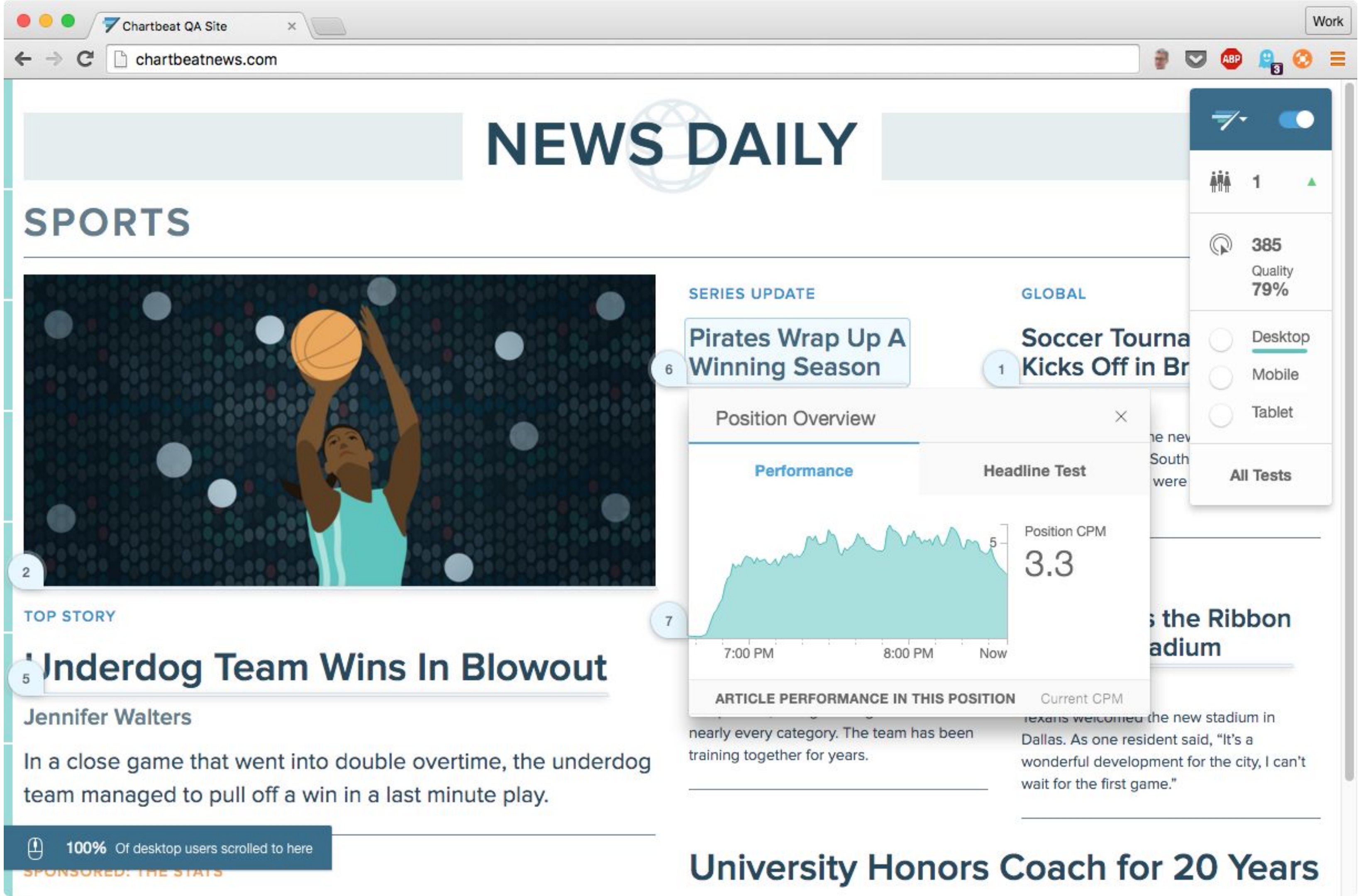
The “new” consumer in beta in Kafka 0.9

```
(let [consumers (repeatedly 4 #(doto (KafkaConsmer. {...}) (.subscribe ["pings"])))]
  (doseq [consumer consumers]
    (async/thread
      (loop []
        (let [msg-batch (.poll consumer 5000)
              pings (map unpack msg-batch)
              first-pings (filter first? pings)]
          (doseq [[[src trg min] ps] (group-by (juxt :http-refer :path get-minute)
                                                first-pings)]
            (increment-in-db src trg min (count ps)))
          (.commitSync consumer)
          (recur))))))
```

#10

SUCCESS!!!

Product Owner


Chartbeat QA Site

chartbeatnews.com

Work

NEWS DAILY

SPORTS

SERIES UPDATE

Pirates Wrap Up A Winning Season

GLOBAL

Soccer Tourna Kicks Off in Br

TOP STORY

Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

nearly every category. The team has been training together for years.

s the Ribbon adium

Texans welcomed the new stadium in Dallas. As one resident said, "It's a wonderful development for the city, I can't wait for the first game."

University Honors Coach for 20 Years

SPONSORED: THE STATS

100% Of desktop users scrolled to here

Position Overview

Performance | Headline Test

Position CPM 3.3

ARTICLE PERFORMANCE IN THIS POSITION Current CPM

1 ▲

385 Quality 79%

Desktop

Mobile

Tablet

All Tests

The *New* User Story

As a…

Homepage Editor

I want…

1. Realtime clicks per minute of each article *and its position* on my homepage.
2. Historical graph of clicks of each article *and its position* on my homepage.

So that…

I can make informed decisions about *the arrangement of* my content on my homepage.

Product Owner

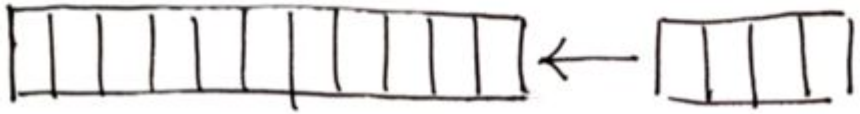
Pinging The “Exit” and the “Click”

/article1

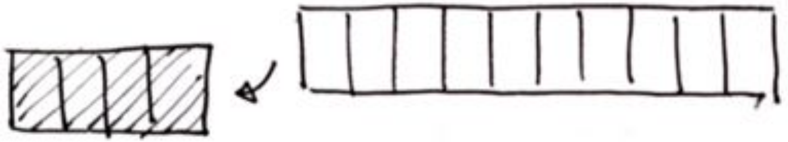
```clojure
(defn group-clicks [first-pings]
  (for [[[http-referer path min] ps] (group-by (juxt :http-refer :path get-minute)
                                               first-pings)]
    {:source http-referer
     :target path
     :minute min
     :num-clicks (count ps)}))

(defn extract-positions [exit-pings]
  (for [exit-ping exit-pings]
    {:source   (:path exit-ping)
     :target   (:href exit-ping)
     :position (:position exit-ping)
     :ts       (:ts    exit-ping)}))

(defn add-new-positions [positions-vec new-positions]
  ...)

(defn decay-old-positions [positions-vec]
  ...)

(defn get-matching-positions [positions-vec {:keys [source target]}]
  ...)

(...
  (loop [positions []]
    (let [pings (map unpack (take 5000 stream "pings"))
          clicks (group-clicks (filter first? pings))
          positions (conj positions (extract-positions (filter exit? pings)))
          positions (decay-old-positions positions)]
      (doseq [click clicks]
        (let [matches (get-matching-positions positions clicks
              imputed-clicks (impute click matches))]
          (doseq [click imputed-clicks]
            (write-to-db click))))
      (recur positions))))
```

#11

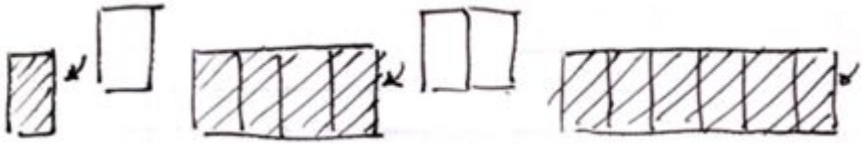
```clojure
(defn add-new-positions [positions-vec new-positions]
  (conj positions-vec new-positions))
```



```clojure
(defn decay-old-positions [positions-vec]
  (let [now (quot (System/currentTimeMillis) 1000)]
    (drop-while #(< (:ts %) (- now 600)) positions-vec)))
```



```clojure
(defn get-matching-positions [positions-vec {:keys [source target]}]
  (filter #(and (= source (:source %))
                (= target (:target %)))
          positions-vec))
```

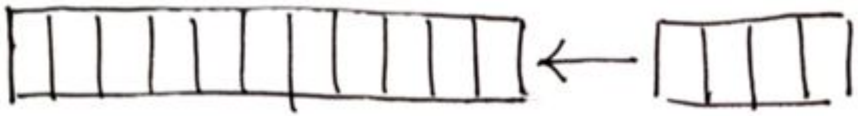


#12

SLOW

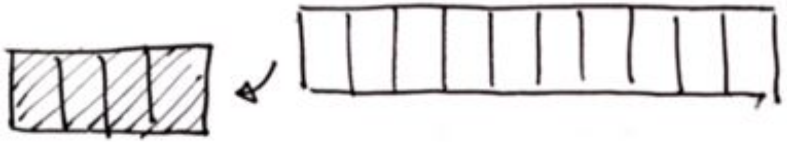
```
(defn add-new-positions [positions-vec new-positions]
  (conj positions-vec new-positions))
```

O(k)



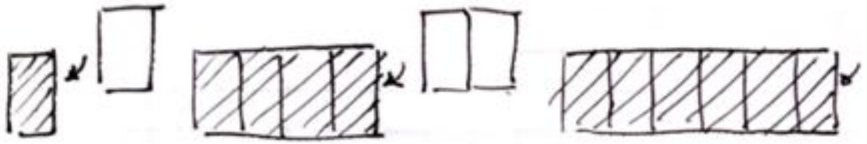
```
(defn decay-old-positions [positions-vec]
  (let [now (quot (System/currentTimeMillis) 1000)]
    (drop-while #(< (:ts %) (- now 600)) positions-vec)))
```

O(n)

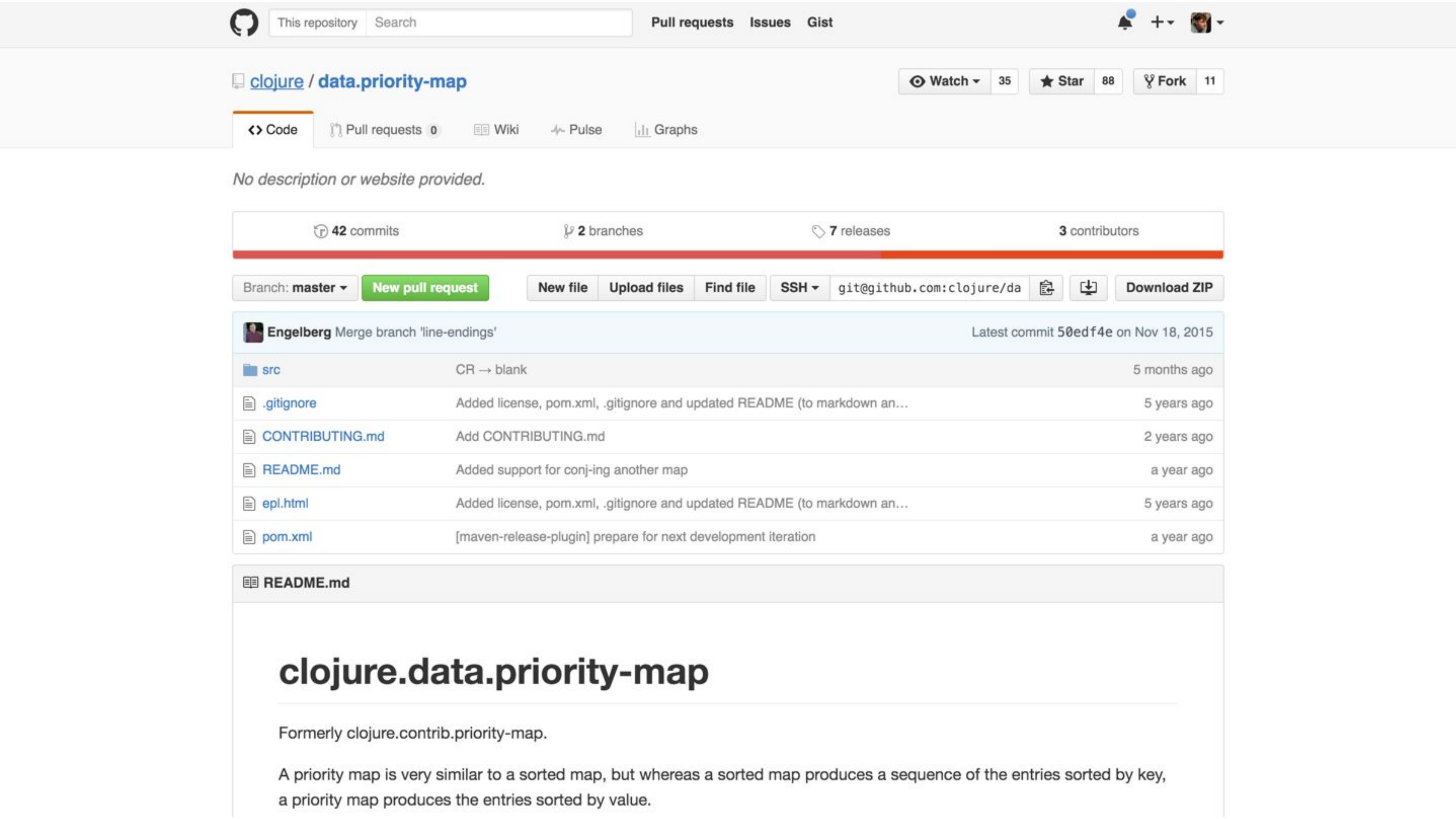


```
(defn get-matching-positions [positions-vec {:keys [source target]}]
  (filter #(and (= source (:source %))
                (= target (:target %)))
          positions-vec))
```

O(n)



#12


This repository Search Pull requests Issues Gist

clojure / data.priority-map

Watch 35 Star 88 Fork 11

Code Pull requests 0 Wiki Pulse Graphs

No description or website provided.

42 commits 2 branches 7 releases 3 contributors

Branch: master New pull request New file Upload files Find file SSH git@github.com:clojure/da Download ZIP

Engelberg Merge branch 'line-endings' Latest commit 50edf4e on Nov 18, 2015

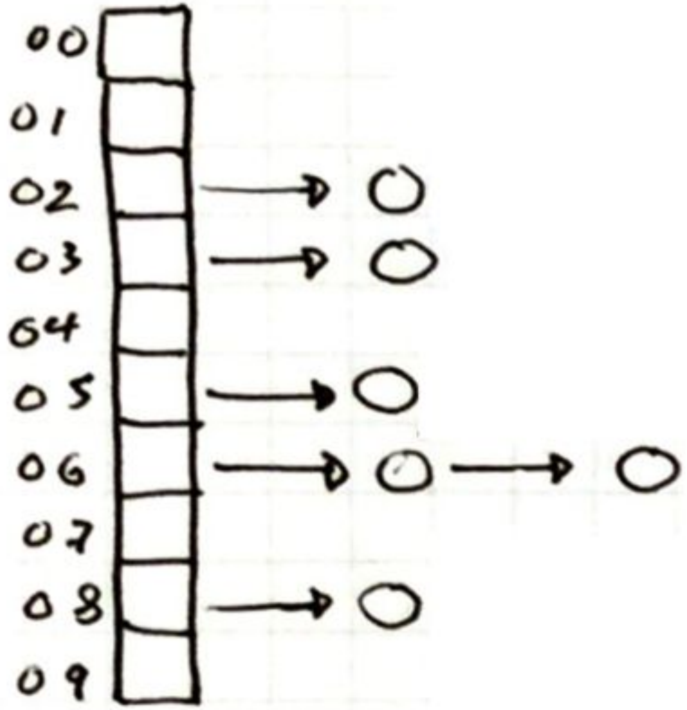
src	CR → blank	5 months ago
.gitignore	Added license, pom.xml, .gitignore and updated README (to markdown an…	5 years ago
CONTRIBUTING.md	Add CONTRIBUTING.md	2 years ago
README.md	Added support for conj-ing another map	a year ago
epl.html	Added license, pom.xml, .gitignore and updated README (to markdown an…	5 years ago
pom.xml	[maven-release-plugin] prepare for next development iteration	a year ago

README.md

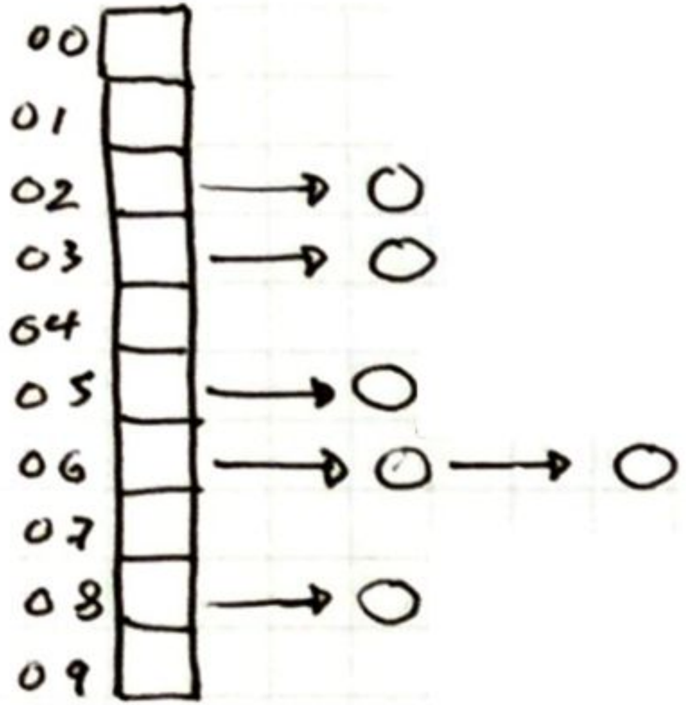
clojure.data.priority-map

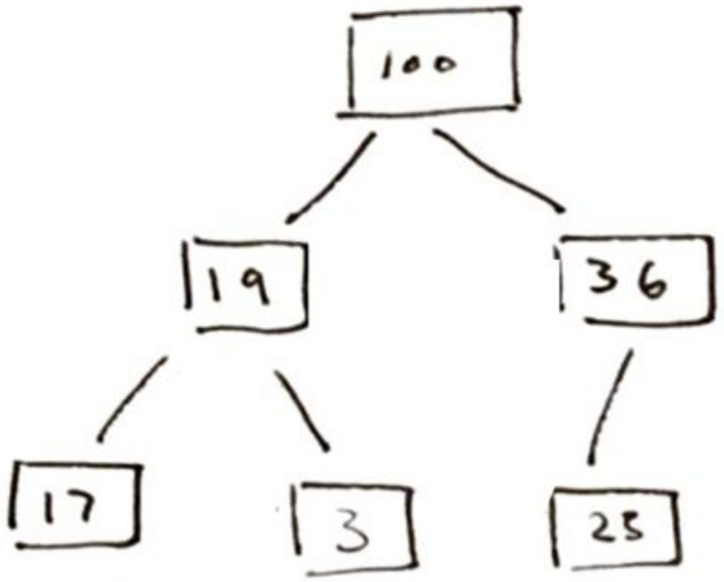
Formerly clojure.contrib.priority-map.

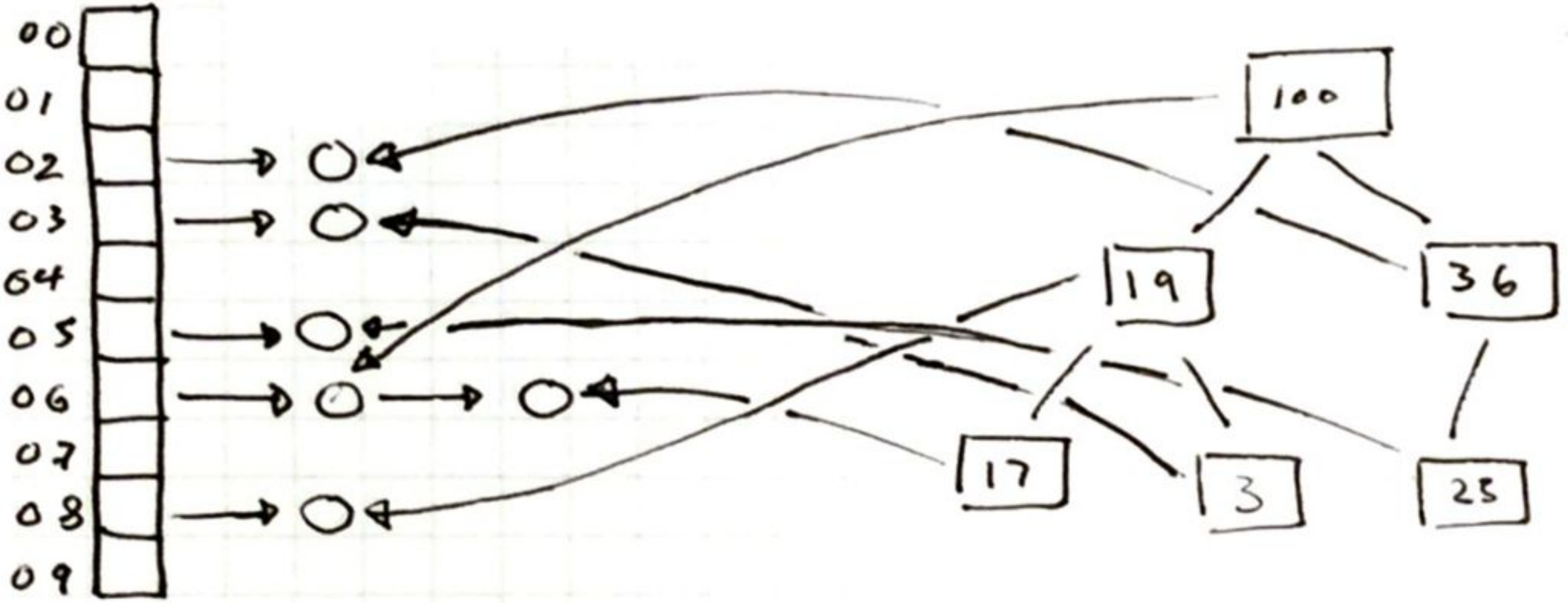
A priority map is very similar to a sorted map, but whereas a sorted map produces a sequence of the entries sorted by key, a priority map produces the entries sorted by value.

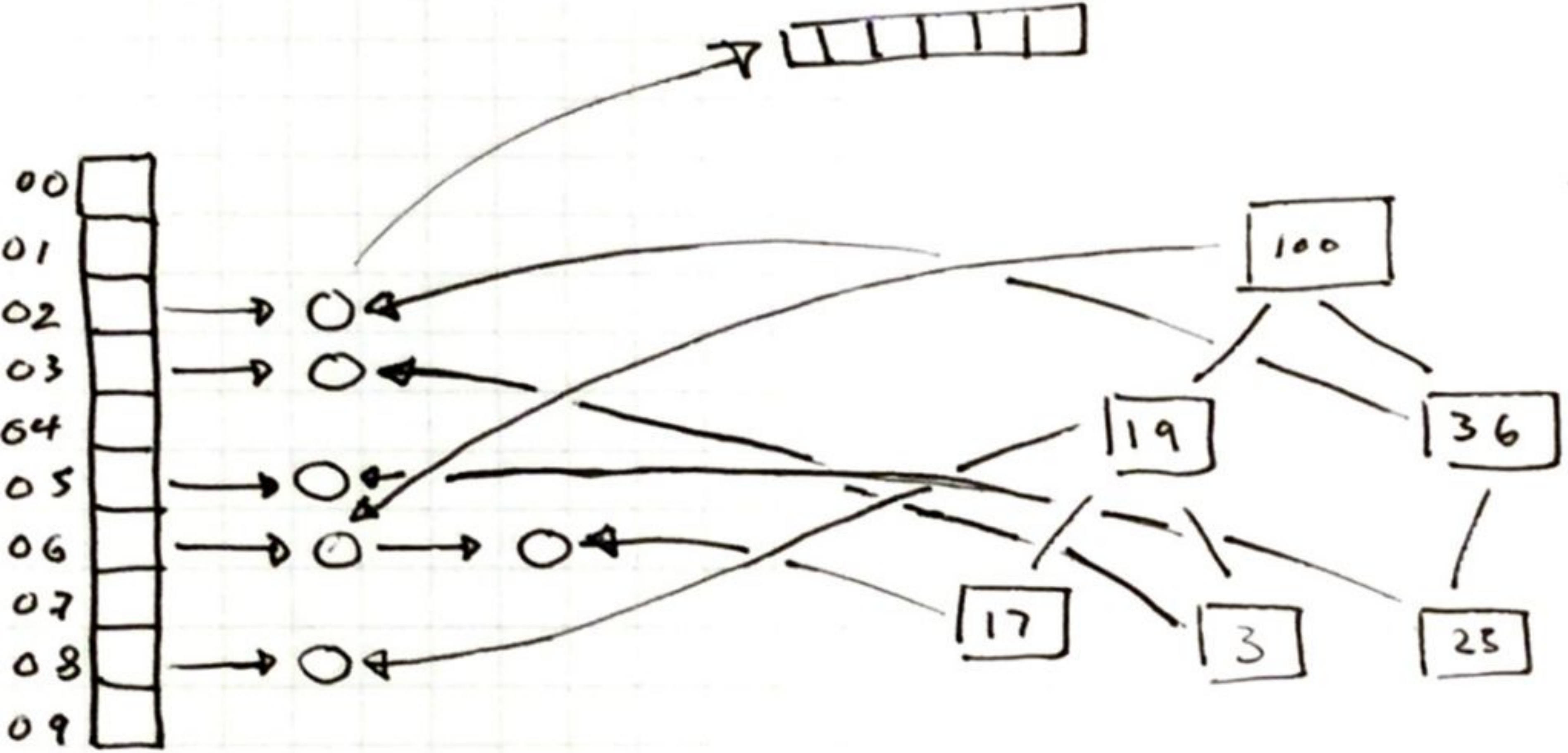


00
01
02
03
04
05
06
07
08
09


00
01
02
03
04
05
06
07
08
09


100
19
36
17
3
25


00
01
02
03
04
05
06
07
08
09
100
19
36
17
3
25


00
01
02
03
04
05
06
07
08
09
100
19
36
17
3
25

```clojure
(priority-map-keyfn-by (comp :ts peek) <)
```

```clojure
(defn add-new-positions [positions-map new-positions]
  (reduce (fn [m {:keys [source target] :as new-pos}]
            (update pos-map
                    [source target]
                    (fnil conj (clojure.lang.PersistentQueue/EMPTY))
                    new-pos))
          positions-map
          new-positions))

(defn decay-old-positions [positions-map]
  (let [now (quot (System/currentTimeMillis) 1000)]
    (loop [m positions-map]
      (let [q (peek m)
            {:keys [ts source target]} (peek q)]
        (if (< ts (- now 600))
          (recur (assoc m [source target] (pop q)))
          m)))))

(defn get-matching-positions [positions-map {:keys [source target]}]
  (get positions-map [source target]))
```

#14

```clojure
(defn add-new-positions [positions-map new-positions]
  (reduce (fn [m {:keys [source target] :as new-pos}]
               (update pos-map
                       [source target]
                       (fnil conj (clojure.lang.PersistentQueue/EMPTY))
                       new-pos))
          positions-map
          new-positions))
```

O(k*log(n))

```clojure
(defn decay-old-positions [positions-map]
  (let [now (quot (System/currentTimeMillis) 1000)]
    (loop [m positions-map]
      (let [q (peek m)
            {:keys [ts source target]} (peek q)]
        (if (< ts (- now 600))
          (recur (assoc m [source target] (pop q)))
          m)))))
```

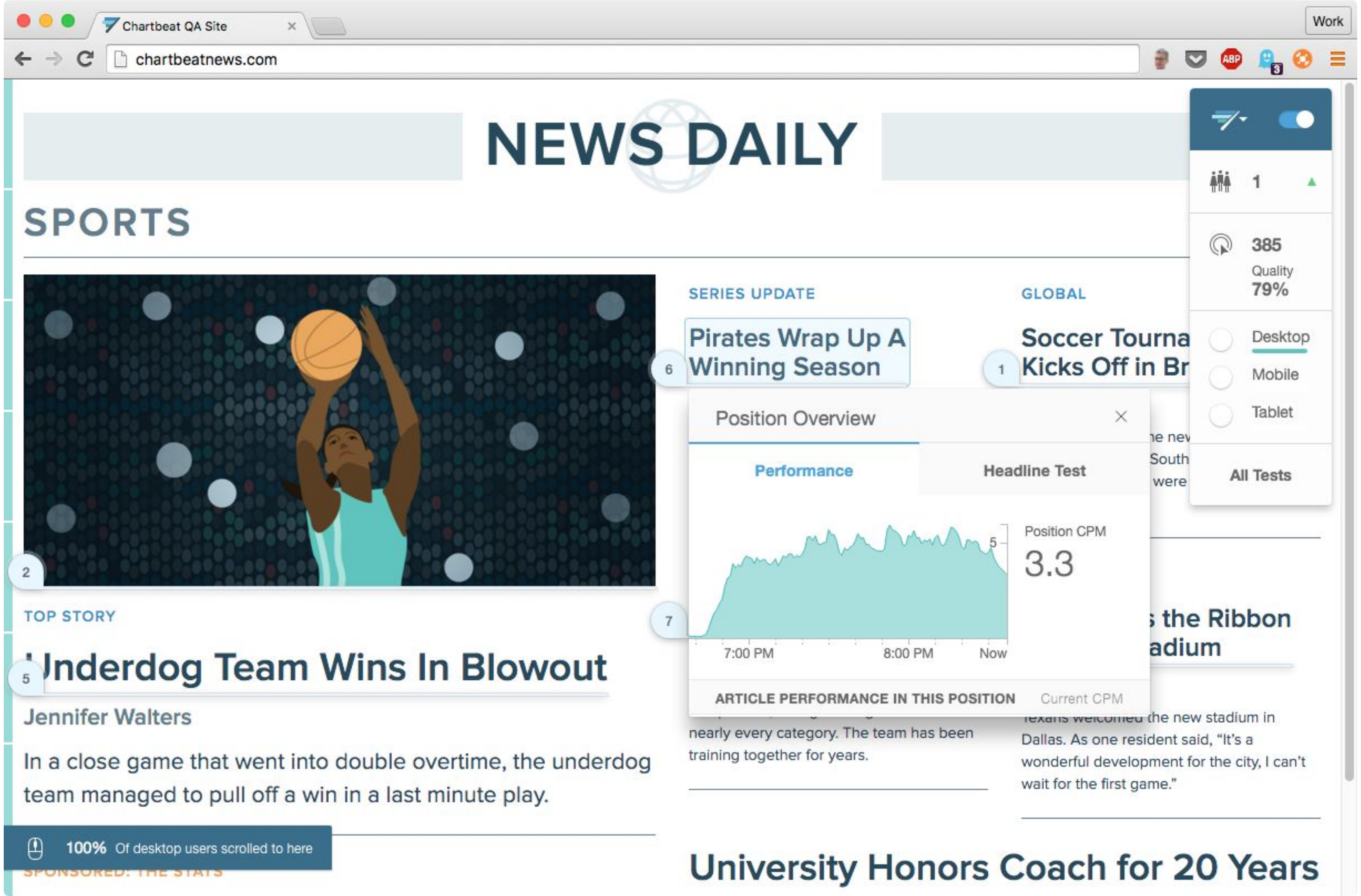
O(d*log(n))

#14

```clojure
(defn get-matching-positions [positions-map {:keys [source target]}]
  (get positions-map [source target]))
```

O(1)

SUCCESS!!!



Chartbeat QA Site

chartbeatnews.com

Work

NEWS DAILY

SPORTS

SERIES UPDATE

Pirates Wrap Up A Winning Season

GLOBAL

Soccer Tourna
Kicks Off in Br

1

385

Quality
79%

Desktop

Mobile

Tablet

All Tests

Position Overview

Performance

Headline Test

Position CPM
3.3

7:00 PM 8:00 PM Now

ARTICLE PERFORMANCE IN THIS POSITION Current CPM

he new
South
were

s the Ribbon
adium

TOP STORY

Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

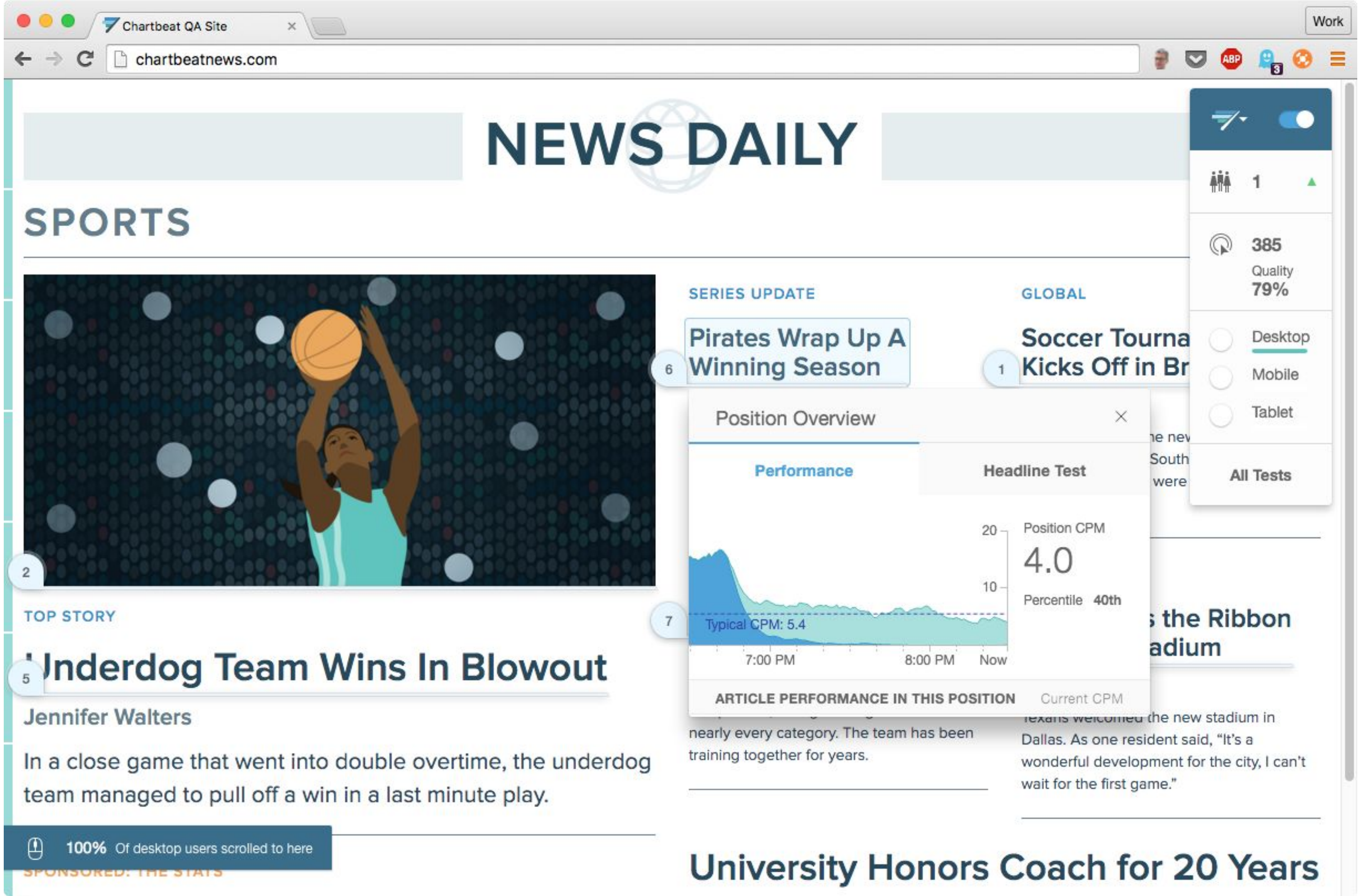
nearly every category. The team has been training together for years.

Texans welcomed the new stadium in Dallas. As one resident said, "It's a wonderful development for the city, I can't wait for the first game."

100% Of desktop users scrolled to here

SPONSORED: THE STATS

University Honors Coach for 20 Years



NEWS DAILY

SPORTS

SERIES UPDATE

Pirates Wrap Up A Winning Season

GLOBAL

Soccer Tourna Kicks Off in Br

Position Overview

Performance

Headline Test

Position CPM

4.0

Percentile 40th

ARTICLE PERFORMANCE IN THIS POSITION Current CPM

TOP STORY

Underdog Team Wins In Blowout

Jennifer Walters

In a close game that went into double overtime, the underdog team managed to pull off a win in a last minute play.

100% Of desktop users scrolled to here

SPONSORED: THE STATS

he nev South were

s the Ribbon adium

nearly every category. The team has been training together for years.

Texans welcomed the new stadium in Dallas. As one resident said, "It's a wonderful development for the city, I can't wait for the first game."

University Honors Coach for 20 Years

What have we learned?

- Beware the head of infinite lazy sequence
- Batch when possible
- Don’t lockstep network and CPU, embrace laziness
- Multithread when possible
- clojure.data.priority-map is the bomb
- **Writing and refactoring a complicated consumer is made simple in clojure**

The JVM is a first-class citizen in stream processing

Framework	Written In	Recommended API
Kafka	Scala	Java
Samza	Scala	Java
Storm	Clojure	Java
Spark	Scala	Scala/Java

Clojure is for ~~kafka~~ stream processing

Thank You