
Σ
@summingbird

Streaming MapReduce with Summingbird

Tuesday, September 3, 2013 | By Sam Ritchie (@sritchie) [15:47 UTC]

Tweet

Today we are open sourcing Summingbird on GitHub under the ALv2.

Twitter Open Source
@TwitterOSS
Follow

we're thrilled to open source @summingbird, streaming mapreduce with @scalding and @stormprocessor #hadoop blog.twitter.com/2013/streaming...

9:04 AM - 3 Sep 2013

Streaming MapReduce with Summingbird | Twitter Blogs

By **Chris Aniszczyk** @cra

Today we are open sourcing Summingbird on GitHub under the ALv2. Summingbird is a library that lets you write streaming MapReduce programs th.....

Twitter Engineering @TwitterEng

59 RETWEETS **46** FAVORITES

Related Posts

Announcing Parquet 1.0: Columnar Storage for Hadoop

A Storm is coming: more details and plans for release

Scalding 0.8.0 and Algebird

Accounts to Follow

Summingbird
@summingbird
Twitter's streaming MapReduce API. Always watching.

Twitter Open Source
@TwitterOSS
Open Programs at Twitter.

Tweets

Follow

Twitter Engineering @TwitterEng 7h

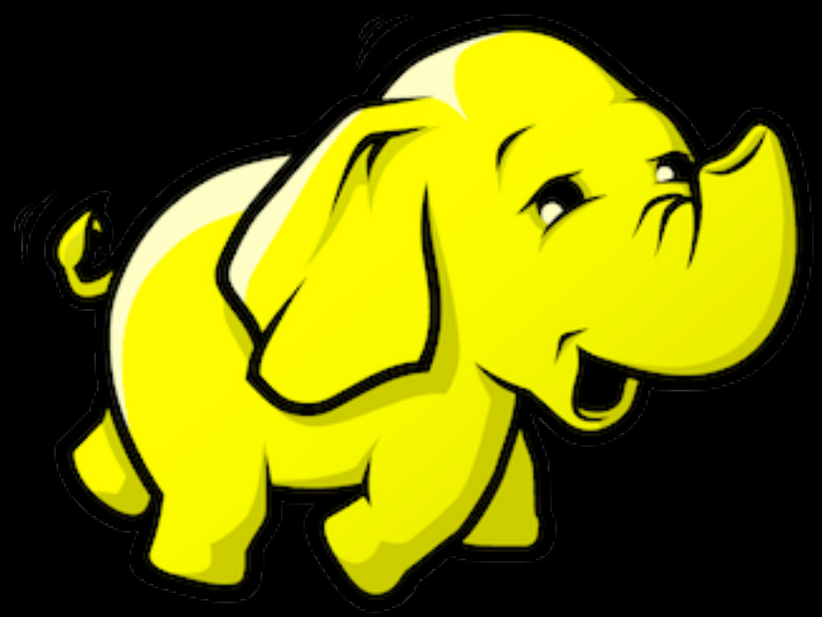
We just open sourced @summingbird, streaming mapreduce with @scalding and

Oscar Boykin - @posco
Sam Ritchie - @sritchie
Ashu Singhal - @daashu

- What is Summingbird?
- What can it do today?
- Functional Design
- Currently deployed systems
- Upcoming Features

Vision

Write your logic once.



Twitter's Scale

- 200M+ Active Monthly Users
- 500M Tweets / Day
- Several 1K+ node Hadoop clusters

Solve systems problems **once**.

Make **non-trivial**
realtime compute
as accessible
as Scalding.

What is Summingbird?

- Declarative Streaming Map/Reduce DSL
- Realtime platform that runs on Storm.
- Batch platform that runs on Hadoop.
- Batch / Realtime Hybrid platform

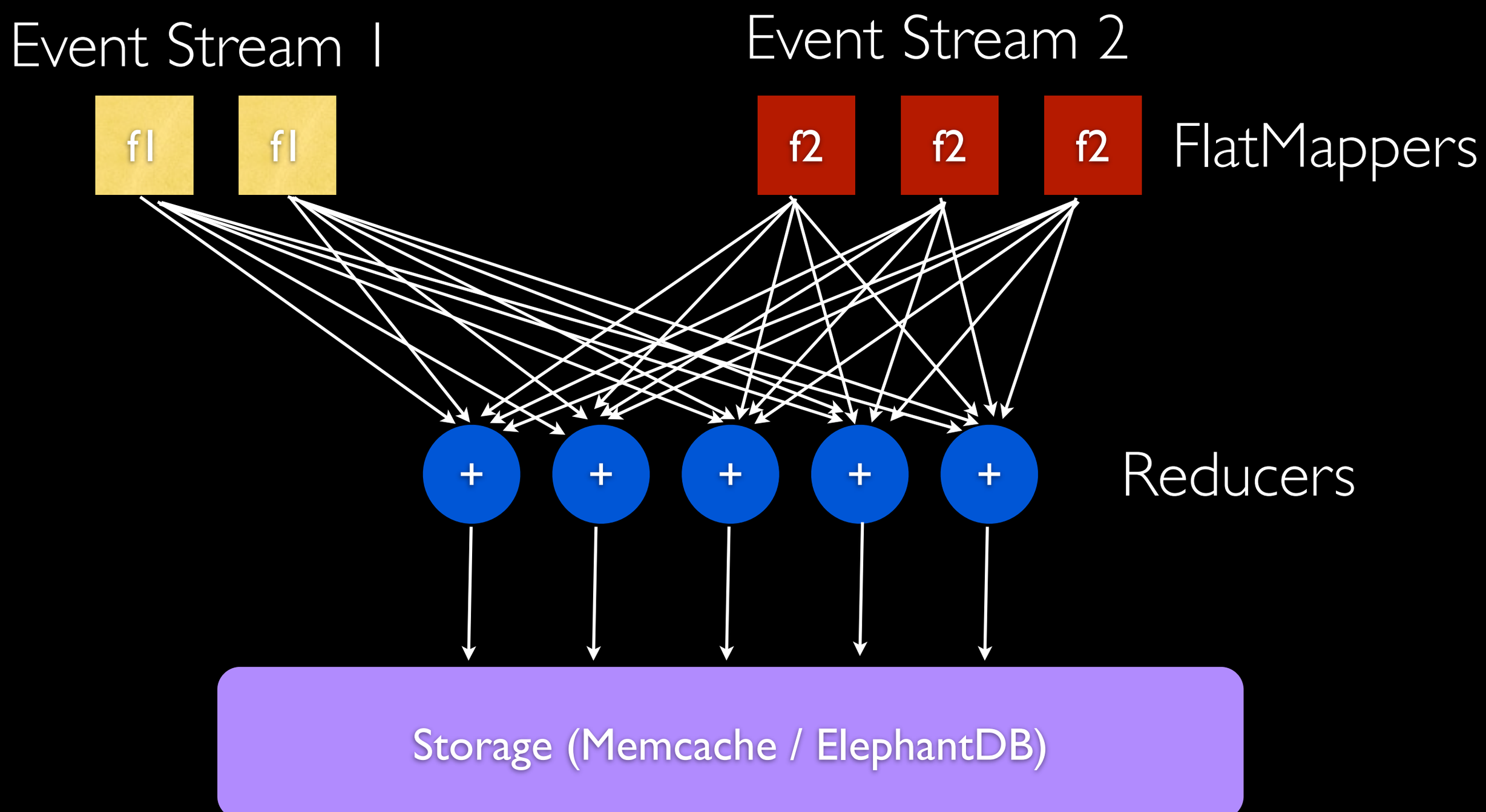
```
val impressionCounts =
  impressionHose.flatMap(extractCounts(_))

val engagementCounts =
  engagementHose.filter(_.isValid)
    .flatMap(engagementCounts(_))

val totalCounts =
  (impressionCounts ++ engagementCounts)
    .flatMap(fanoutByTime(_))
    .sumByKey(onlineStore)

val stormTopology =
  Storm.remote("stormName").plan(totalCounts)
val hadoopJob =
  Scalding("scaldingName").plan(totalCounts)
```

Map/Reduce


Event Stream 1
Event Stream 2
f1
f1
f2
f2
f2
FlatMappers
+
+
+
+
+
Reducers
Storage (Memcache / ElephantDB)

FlatMap

```
flatMap: T => TraversableOnce[U]

// g: (x: T => U)
map(x) = flatMap(x => List(g(x))

// pred: T => Boolean
filter(x) = flatMap { x =>
  if (pred(x)) List(x) else Nil
}
```

- Source[+T]
- Store[-K, V]
- Sink[-T]
- Service[-K, +V]

The Four Ss!

- Source[+T]
- Store[-K, V]
- Sink[-T]
- Service[-K, +V]

Store[-K, V]:

What values are allowed?

```scala
trait Monoid[T] {
  def zero: T
  def plus(l: T, r: T): T
}
```

- Tons O'Monoids:
- CMS, HyperLogLog, ExponentialMA, BloomFilter, Moments, MinHash, TopK

https://github.com/twitter/algebird/tree/develop/algebird-core/src/main/scala/com/twitter/alg

d Libra | RB | My Delicious | SavePublishing | Bookmark on Delicio | Add to Wish List

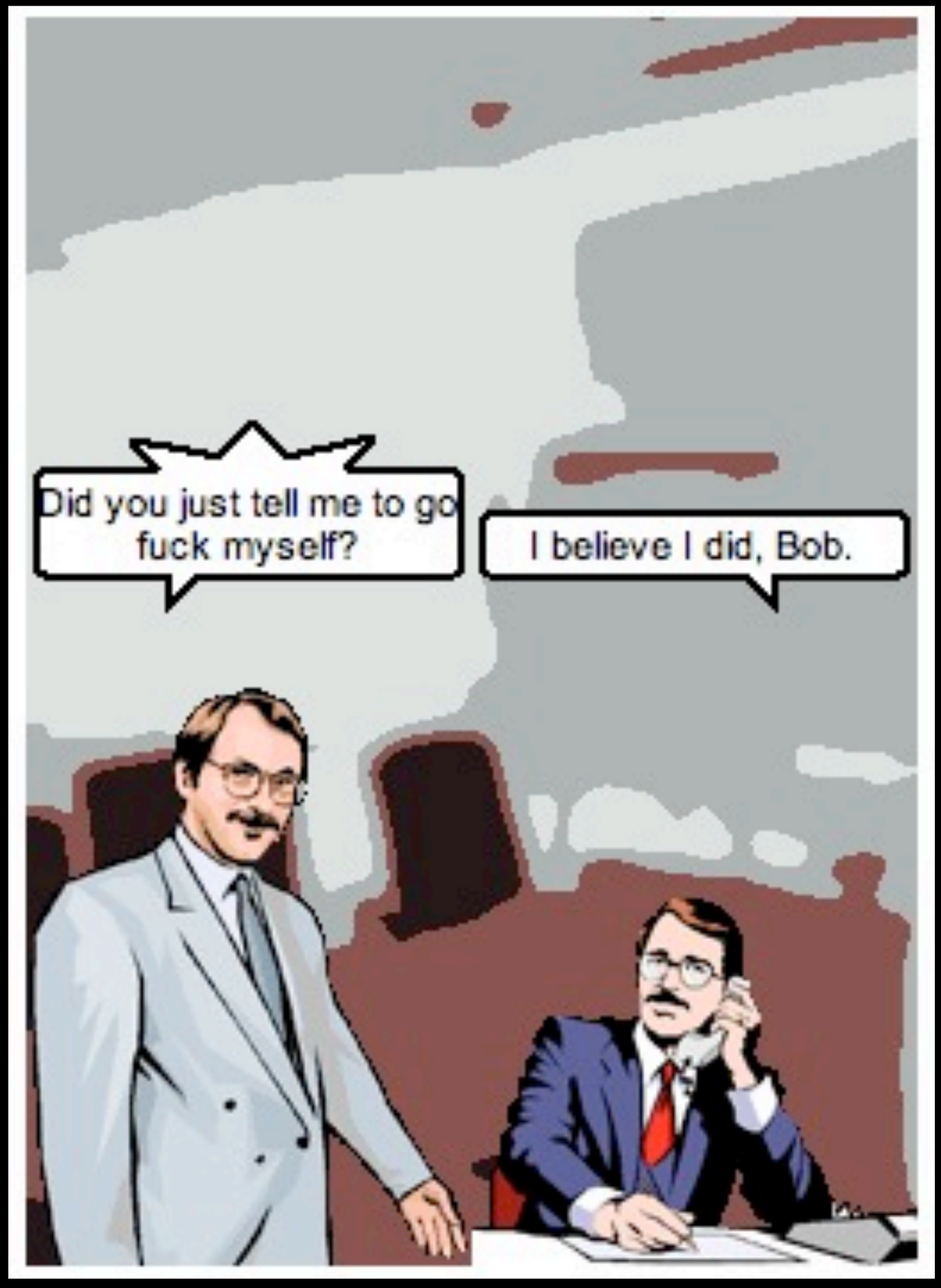
algebird / algebird-core / src / main / scala / com / twitter / **algebird** /

Merge pull request #136 from ccsevers/add_foldM ···

johnynek authored 3 days ago

..

mutable	a month ago	Adds priority queue aggregator [johnyne
AdjoinedUnitRing.scala	18 days ago	Cleans up intTimes [johnynek]
AffineFunction.scala	a month ago	test [sritchie]
Aggregator.scala	a month ago	test [sritchie]
Approximate.scala	a month ago	test [sritchie]
AveragedValue.scala	a month ago	test [sritchie]
BloomFilter.scala	a month ago	test [sritchie]
CountMinSketch.scala	3 days ago	Hotfix for CMS [johnynek]
DecayedValue.scala	a month ago	test [sritchie]
DecayedVector.scala	a month ago	test [sritchie]
Eventually.scala	a month ago	add eventually [sritchie]
Field.scala	a month ago	test [sritchie]
GeneratedAbstractAlgebra.scala	a month ago	test [sritchie]
Group.scala	18 days ago	Cleans up intTimes [johnynek]
HyperLogLog.scala	a month ago	test [sritchie]
IndexedSeq.scala	a month ago	test [sritchie]
JavaMonoids.scala	a month ago	test [sritchie]
MapAlgebra.scala	17 days ago	Adds a comment (to restart travis) [johny
Metric.scala	a month ago	test [sritchie]
MinHasher.scala	a month ago	test [sritchie]


Did you just tell me to go fuck myself?
I believe I did, Bob.

Associativity

```
;; 7 steps
```

a0 + a1 + a2 + a3 + a4 + a5 + a6 + a7

```
;; 7 steps
(+ a0 a1 a2 a3 a4 a5 a6 a7)
```

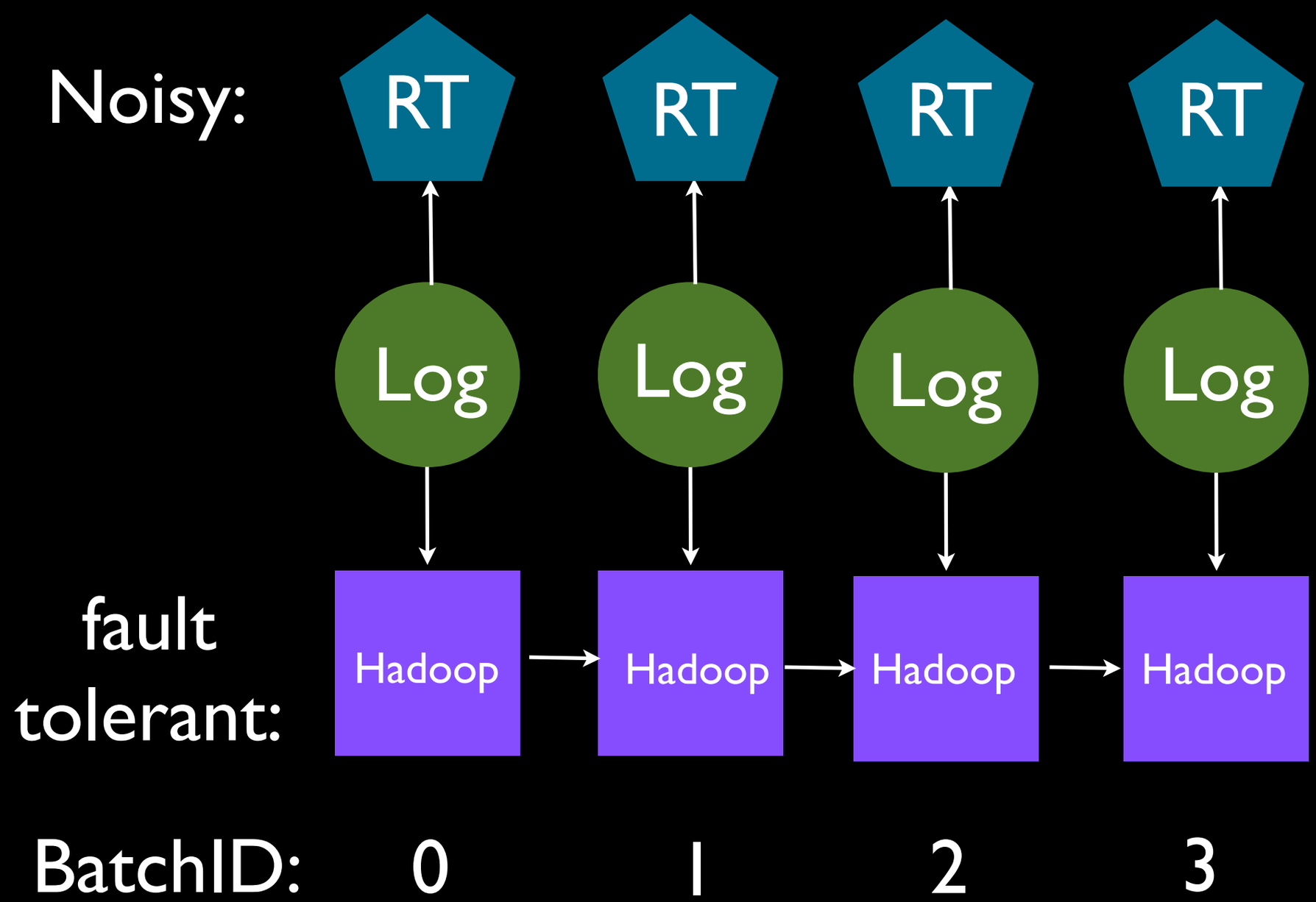
```
;; 5 steps
(+ (+ a0 a1)
   (+ a2 a3)
   (+ a4 a5)
   (+ a6 a7))
```

```
;; 3 steps
(+ (+ (+ a0 a1)
      (+ a2 a3))
   (+ (+ a4 a5)
      (+ a6 a7)))
```

Parallelism

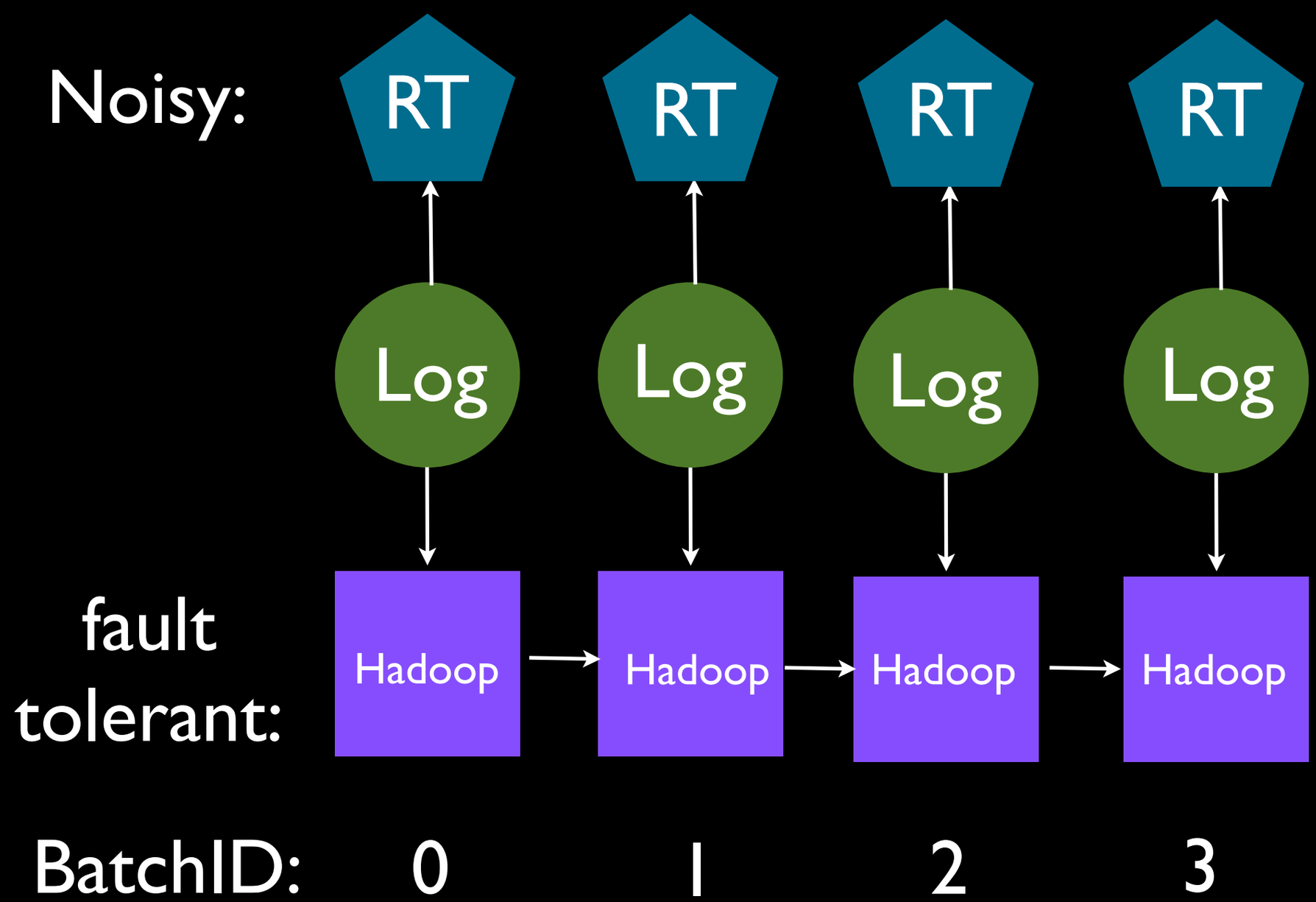
Associativity

Batch / Realtime



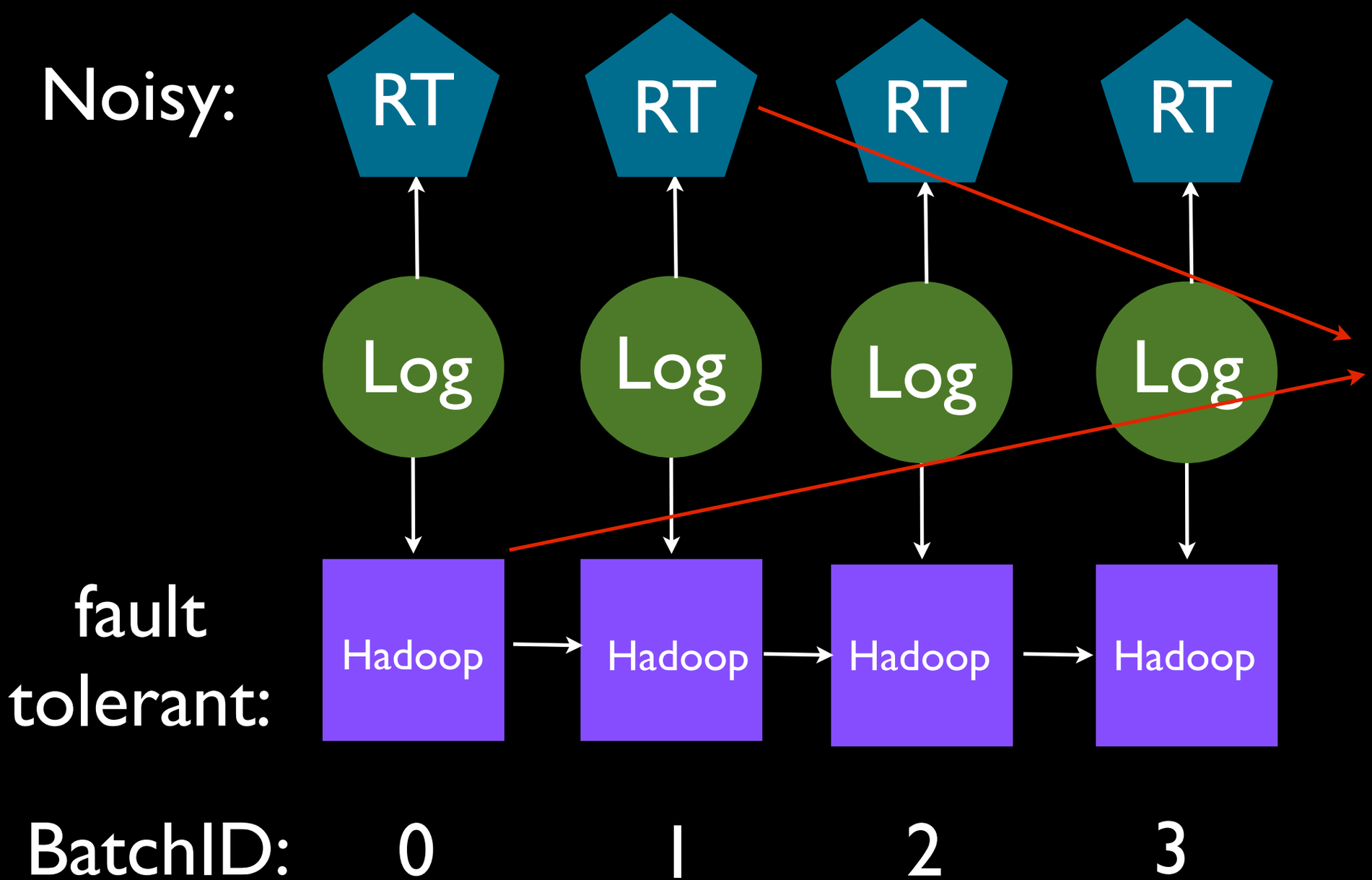
Realtime sums from 0, each batch

Batch / Realtime



Hadoop keeps a total sum (reliably)

Batch / Realtime



Sum of RT Batch(i) + Hadoop Batch(i-1) has bounded noise, bounded read/write size

Tweet Embed Counts

Four more years. pic.twitter.com/bAJE6Vom

8:16 PM - 6 Nov 2012

800,833 RETWEETS 299,776 FAVORITES

How to Embed a Tweet on your Website

Every Tweet on twitter.com and Tweetdeck has a set of Tweet actions at the bottom, including Reply, Retweet, Favorite, and More. Click the "More" Tweet action and select "Embed Tweet":

Popular Latest

0

Acquires Mobile Advertising
ge MoPub
0

nn Is Excited About Twitter.
0

CEO Talks About How To Be
r
0

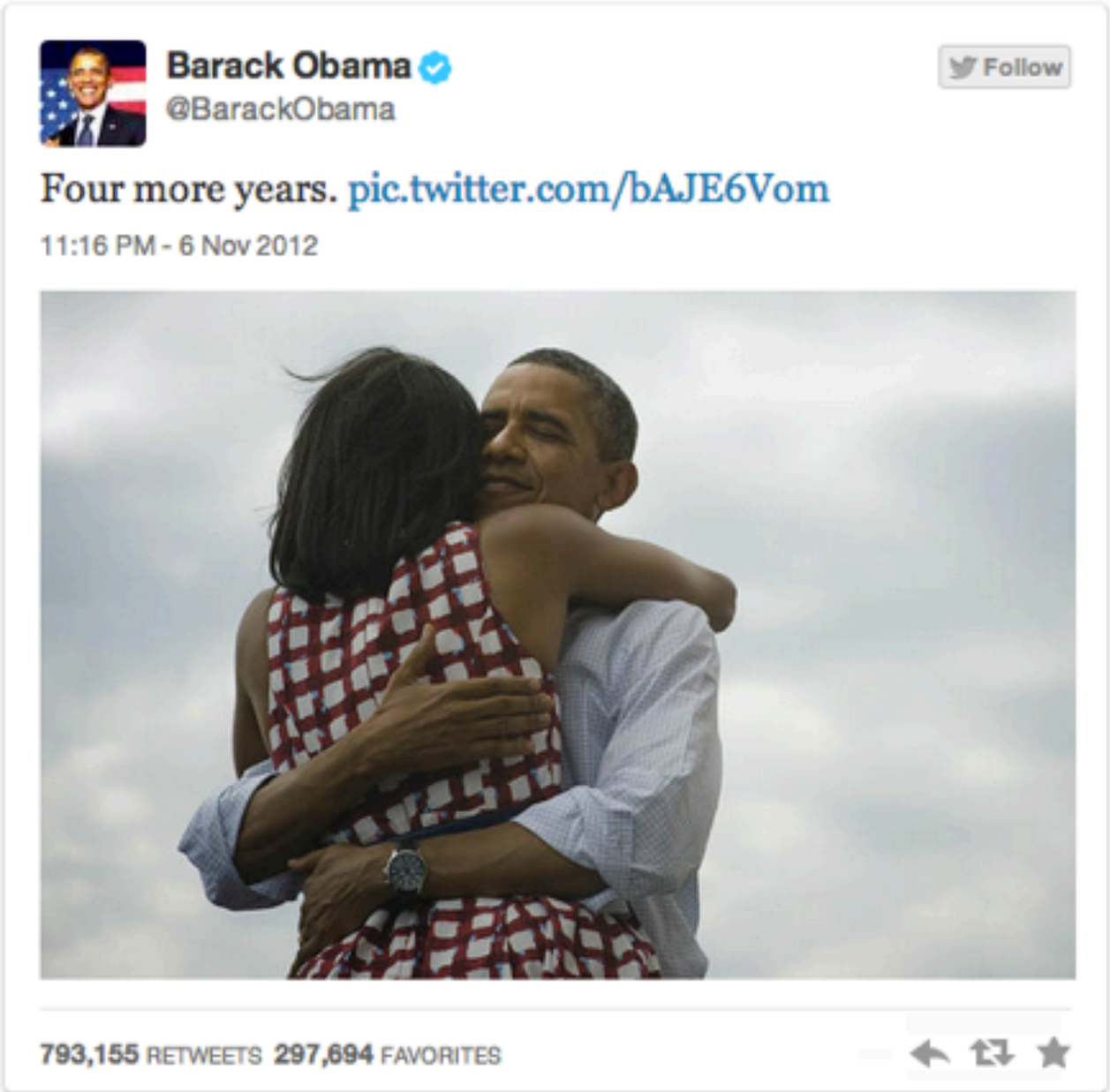
low Lets You Access ALL
ITTERS
0

ur Twitter Profile May be an
0

Amplify Signs Its First Video
ship in Europe
0

Load More

This one pretty much speaks for itself. A simple message that acknowledges what everyone reading the tweet will already know.



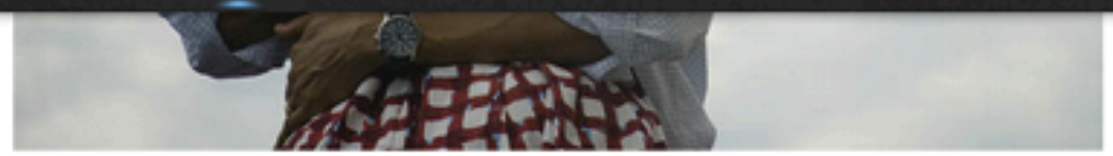
President Obama's message became the most shared tweet of all time within a day of it going out.

9. Bill Gates' tribute to Steve Jobs

tter / BarackObama: Fo × | 10 tweets that gripped the × | (1) Twitter × | Taking Screenshots in Mac ×

witter, Inc. [US] https://twitter.com/BarackObama/status/266031293945503744

Home | Connect | Discover | Me | Search

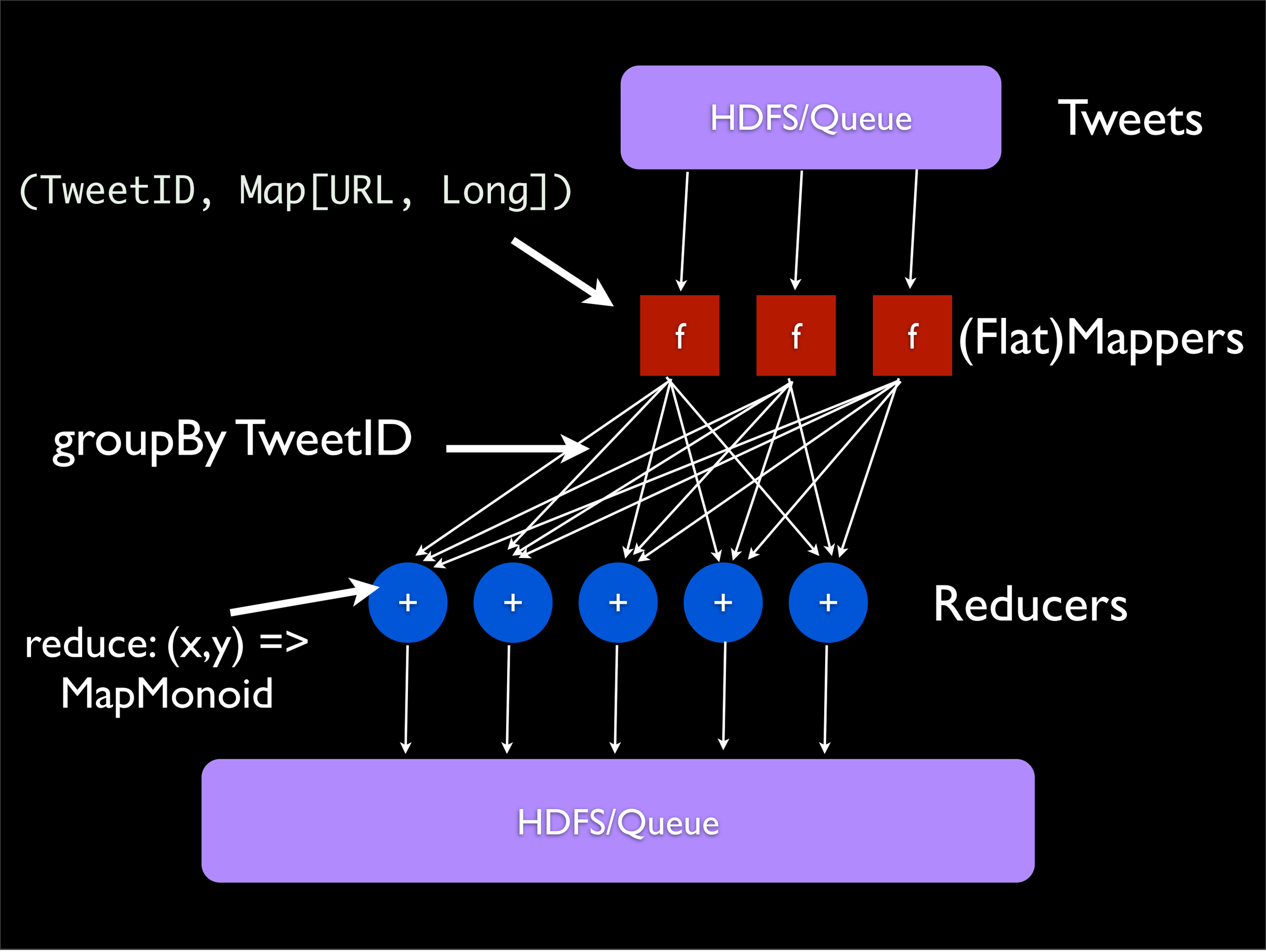


793,155 RETWEETS | 297,694 FAVORITES

8:16 PM - 6 Nov 12 | Flag media

Related headlines

- Overview Embedded Tweets make it possible for you take ...
  Twitter API @twitterapi
- Key statistics, most-followed users, the most popular twee...
  Yahoo News @YahooNews
- Nach „Hashtake"-Versprecher von ZDF-Reporter: Welche Be...
  BILD @BILD
- Du premier tweet de Jack Dorsey à celui du pape François, ...
  Nouvel Observateur @LeNouvelObs
- Twitter swelled to 200 million monthly active users in 2012. ...
  Mashable @mashable
- Twitter is synonymous with many things. It's where people ...
  The Next Web @TheNextWeb
- Au terme d'une longue campagne, Barack Obama a été réél...
  Le Monde @lemondefr
- Andy Murray's big Wimbledon win made history in sports a...
  GigaOM @gigaom
- It may already be social media's most-shared image ever --...
  CNN @CNN
- What is the most tweeted about event of all time? If you we...
  Metro @MetroUK
- HOLLANDE - On a beau dire, ces petits instants volés, dans...
  Le HuffPost @LeHuffPost
- In 55 wide-ranging lists, TIME surveys the highs and lows, t...


HDFS/Queue
Tweets
(TweetID, Map[URL, Long])
f
f
f
(Flat)Mappers
groupBy TweetID
+
+
+
+
+
Reducers
reduce: (x,y) =>
MapMonoid
HDFS/Queue

```scala
object OuroborosJob {
  def apply[P <: Platform[P]](source: Producer[P, ClientEvent], sink: P#Store[OuroborosKey, OuroborosValue]) =
    source.filter(filterEvents(_))
      .flatMap { event =>
      val widgetDetails = event.getWidget_details
      val referUrl: String = widgetDetails.getWidget_origin
      val timestamp: Long = event.getLog_base.getTimestamp
      val widgetFrameUrlOpt: Option[String] = Option(widgetDetails.getWidget_frame)
      for {
        tweetId: java.lang.Long <- javaToScalaSafe(event.getEvent_details.getItem_ids)
        timeBucketOption: Option[TimeBucket] <- timeBucketsForTimestamp(timestamp)
      } yield {
        val urlHllOption = canonicalUrl(referUrl).map(hllMonoid.create(_))
        val widgetFrameUrlsOption = widgetFrameUrlOpt map { widgetUrl: String =>
          widgetFrameUrlsSmMonoid.create((referUrl, (widgetFrameUrlSetSmMonoid.create((widgetUrl, 1L)), 1L)))
        }
        val impressionsValue: OuroborosValue = RawImpressions(
          impressions = 1L,
          approxUniqueUrls = urlHllOption,
          urlCounts = Some(embedCountSmMonoid.create((referUrl, 1L))),
          urlDates = Some(embedDateSmMonoid.create((referUrl, timestamp))),
          frameUrls = widgetFrameUrlsOption
        ).as[OuroborosValue]
        Seq(
          (OuroborosKey.ImpressionsKey(ImpressionsKey(tweetId.longValue, timeBucketOption)), impressionsValue),
          (OuroborosKey.TopTweetsKey(TopTweetsKey(timeBucketOption)), topTweetsValue)
        )
      }
    }.sumByKey(store)
      .set(MonoidIsCommutative(true))
}
```

```scala
object OuroborosJob {
  def apply[P <: Platform[P]](source: Producer[P, ClientEvent], sink: P#Store[OuroborosKey, OuroborosValue]) =
    source.filter(filterEvents(_))
      .flatMap { event =>
      val widgetDetails = event.getWidget_details
      val referUrl: String = widgetDetails.getWidget_origin
      val timestamp: Long = event.getLog_base.getTimestamp
      val widgetFrameUrlOpt: Option[String] = Option(widgetDetails.getWidget_frame)
      for {
        tweetId: java.lang.Long <- javaToScalaSafe(event.getEvent_details.getItem_ids)
        timeBucketOption: Option[TimeBucket] <- timeBucketsForTimestamp(timestamp)
      } yield {
        val urlHllOption = canonicalUrl(referUrl).map(hllMonoid.create(_))
        val widgetFrameUrlsOption = widgetFrameUrlOpt map { widgetUrl: String =>
          widgetFrameUrlsSmMonoid.create((referUrl, (widgetFrameUrlSetSmMonoid.create((widgetUrl, 1L)), 1L)))
        }
        val impressionsValue: OuroborosValue = RawImpressions(
          impressions = 1L,
          approxUniqueUrls = urlHllOption,
          urlCounts = Some(embedCountSmMonoid.create((referUrl, 1L))),
          urlDates = Some(embedDateSmMonoid.create((referUrl, timestamp))),
          frameUrls = widgetFrameUrlsOption
        ).as[OuroborosValue]
        Seq(
          (OuroborosKey.ImpressionsKey(ImpressionsKey(tweetId.longValue, timeBucketOption)), impressionsValue),
          (OuroborosKey.TopTweetsKey(TopTweetsKey(timeBucketOption)), topTweetsValue)
        )
      }
    }.sumByKey(store)
      .set(MonoidIsCommutative(true))
}
```

Filter Events

```
object OuroborosJob {
  def apply[P <: Platform[P]](source: Producer[P, ClientEvent], sink: P#Store[OuroborosKey, OuroborosValue]) =
    source.filter(filterEvents(_))
      .flatMap { event =>
        val widgetDetails = event.getWidget_details
        val referUrl: String = widgetDetails.getWidget_origin
        val timestamp: Long = event.getLog_base.getTimestamp
        val widgetFrameUrlOpt: Option[String] = Option(widgetDetails.getWidget_frame)
        for {
          tweetId: java.lang.Long <- javaToScalaSafe(event.getEvent_details.getItem_ids)
          timeBucketOption: Option[TimeBucket] <- timeBucketsForTimestamp(timestamp)
        } yield {
          val urlHllOption = canonicalUrl(referUrl).map(hllMonoid.create(_))
          val widgetFrameUrlsOption = widgetFrameUrlOpt map { widgetUrl: String =>
            widgetFrameUrlsSmMonoid.create((referUrl, (widgetFrameUrlSetSmMonoid.create((widgetUrl, 1L)), 1L)))
          }
          val impressionsValue: OuroborosValue = RawImpressions(
            impressions = 1L,
            approxUniqueUrls = urlHllOption,
            urlCounts = Some(embedCountSmMonoid.create((referUrl, 1L))),
            urlDates = Some(embedDateSmMonoid.create((referUrl, timestamp))),
            frameUrls = widgetFrameUrlsOption
          ).as[OuroborosValue]
          Seq(
            (OuroborosKey.ImpressionsKey(ImpressionsKey(tweetId.longValue, timeBucketOption)), impressionsValue),
            (OuroborosKey.TopTweetsKey(TopTweetsKey(timeBucketOption)), topTweetsValue)
          )
        }
    }.sumByKey(store)
      .set(MonoidIsCommutative(true))
}
```

Filter Events

Generate KV Pairs

```
object OuroborosJob {
  def apply[P <: Platform[P]](source: Producer[P, ClientEvent], sink: P#Store[OuroborosKey, OuroborosValue]) =
    source.filter(filterEvents(_))
      .flatMap { event =>
      val widgetDetails = event.getWidget_details
      val referUrl: String = widgetDetails.getWidget_origin
      val timestamp: Long = event.getLog_base.getTimestamp
      val widgetFrameUrlOpt: Option[String] = Option(widgetDetails.getWidget_frame)
      for {
        tweetId: java.lang.Long <- javaToScalaSafe(event.getEvent_details.getItem_ids)
        timeBucketOption: Option[TimeBucket] <- timeBucketsForTimestamp(timestamp)
      } yield {
        val urlHllOption = canonicalUrl(referUrl).map(hllMonoid.create(_))
        val widgetFrameUrlsOption = widgetFrameUrlOpt map { widgetUrl: String =>
          widgetFrameUrlsSmMonoid.create((referUrl, (widgetFrameUrlSetSmMonoid.create((widgetUrl, 1L)), 1L)))
        }
        val impressionsValue: OuroborosValue = RawImpressions(
          impressions = 1L,
          approxUniqueUrls = urlHllOption,
          urlCounts = Some(embedCountSmMonoid.create((referUrl, 1L))),
          urlDates = Some(embedDateSmMonoid.create((referUrl, timestamp))),
          frameUrls = widgetFrameUrlsOption
        ).as[OuroborosValue]
        Seq(
          (OuroborosKey.ImpressionsKey(ImpressionsKey(tweetId.longValue, timeBucketOption)), impressionsValue),
          (OuroborosKey.TopTweetsKey(TopTweetsKey(timeBucketOption)), topTweetsValue)
        )
      }
    }.sumByKey(store)
      .set(MonoidIsCommutative(true))
}
```

Filter Events

Generate KV Pairs

Sum into Store

Brief Explanation

This job creates two types of keys:

1: ((TweetId, TimeBucket) => [URL, Impressions])
2: TimeBucket => Map[TweetId, Impressions]

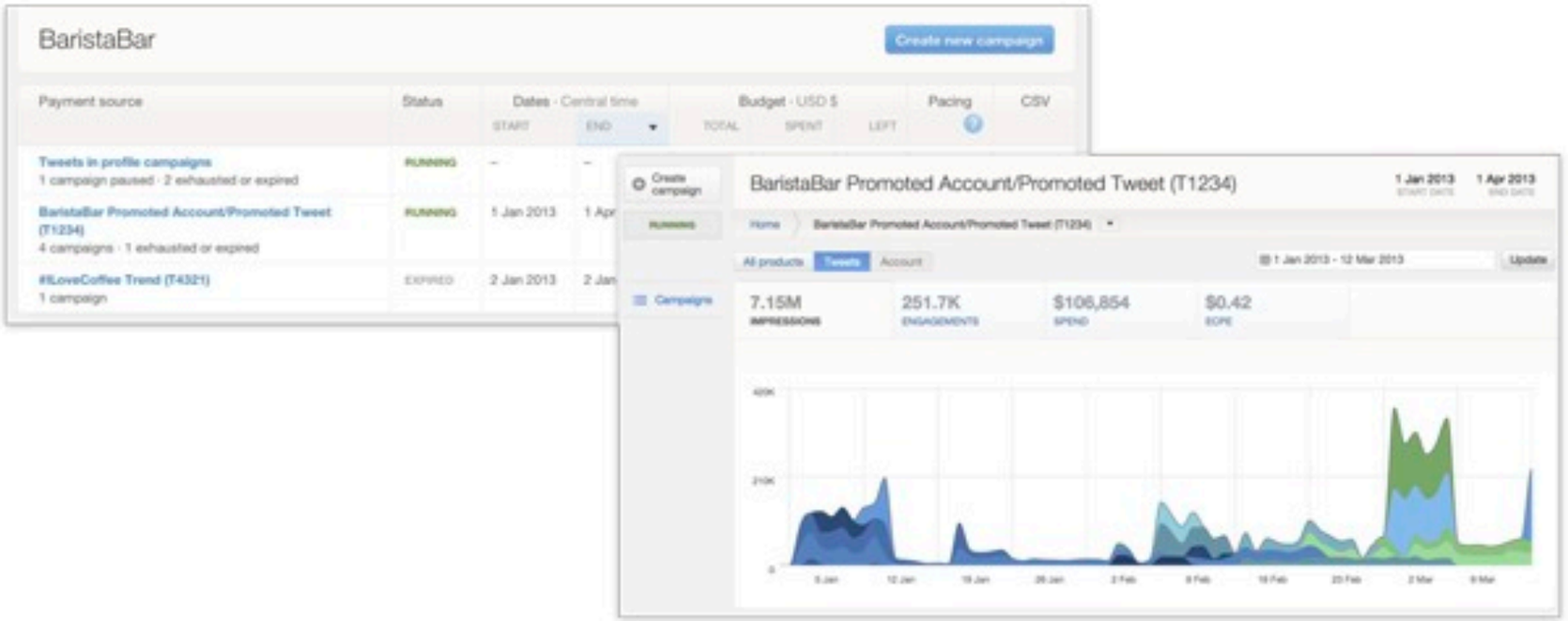
What Else?

Twitter Advertising

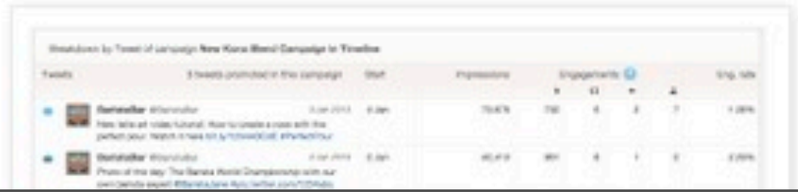
WEDNESDAY, MARCH 13, 2013

The new Twitter Ads center

Today, we're excited to share some changes we've made to the Twitter Ads center. Based on feedback from our advertisers, we've created a revamped experience that improves campaign reporting, provides more visibility into campaign performance analytics and spend, and also makes it easier to manage campaigns in real time.



A major focus of ours is improving campaign analytics. With this in mind, we are now reporting all engagements that Promote Tweets receive — not just engagements that advertisers pay for, but earned media as well. This change gives marketers more complete insight into the impact Promoted Tweets have in driving engagement and exposure on Twitter.



What's Next?

Future Plans

- Akka, Spark, Tez Platforms
- Pluggable graph optimizations
- Metadata publishing via HCatalog
- More tutorials!

Open Source!

rackObama: Fo | (1) Twitter | Issues · twitter/summingb

Inc. [US] https://github.com/twitter/summingbird/issues?state=open

Notes Essays—Peter | The Feynman Lectur | Read | Lift | 750 Words

twitter / summingbird

Browse Issues | Milestones

veryone's Issues	52
ssigned to you	0
reated by you	23
lentioning you	4
milestone selected	
bels	
cleanup	3
client	3
enhancement	9
graph	5
newbie	4
scalding	11
storm	6
bug	0
duplicate	0

52 Open | 187 Closed | Sort: Newest

Close | Label | Assignee | Miles

- Feature/online graph node names
  Opened by ianoc 9 hours ago
- Adds the AlsoProducer node and tes
  Opened by johnynek 16 hours ago
- de-duping self merges in the online p
  Opened by ianoc 19 hours ago 2 comments
- Feature/add storm dot graph
  Opened by ianoc 2 days ago 5 comments
- Create logo for Summingbird
  Opened by caniszczyk 2 days ago
- Add debug info storm config
  Opened by johnynek 3 days ago
- Fix specs that are not using matchers
  Opened by johnynek 4 days ago
- Sinks should always be Sources enh
  Opened by sritchie 4 days ago 1 comment

Summary

- Summingbird is appropriate for the majority of the real-time apps we have.
- It's all about the Monoid
- Data scientists who are not familiar with systems can deploy realtime systems.
- Systems engineers can reuse 90% of the code (batch/realtime merging).


Thank You!
Follow me at @sritchie