Typesafe

You say big data
I say fast data

Nilanjan Raychaudhuri

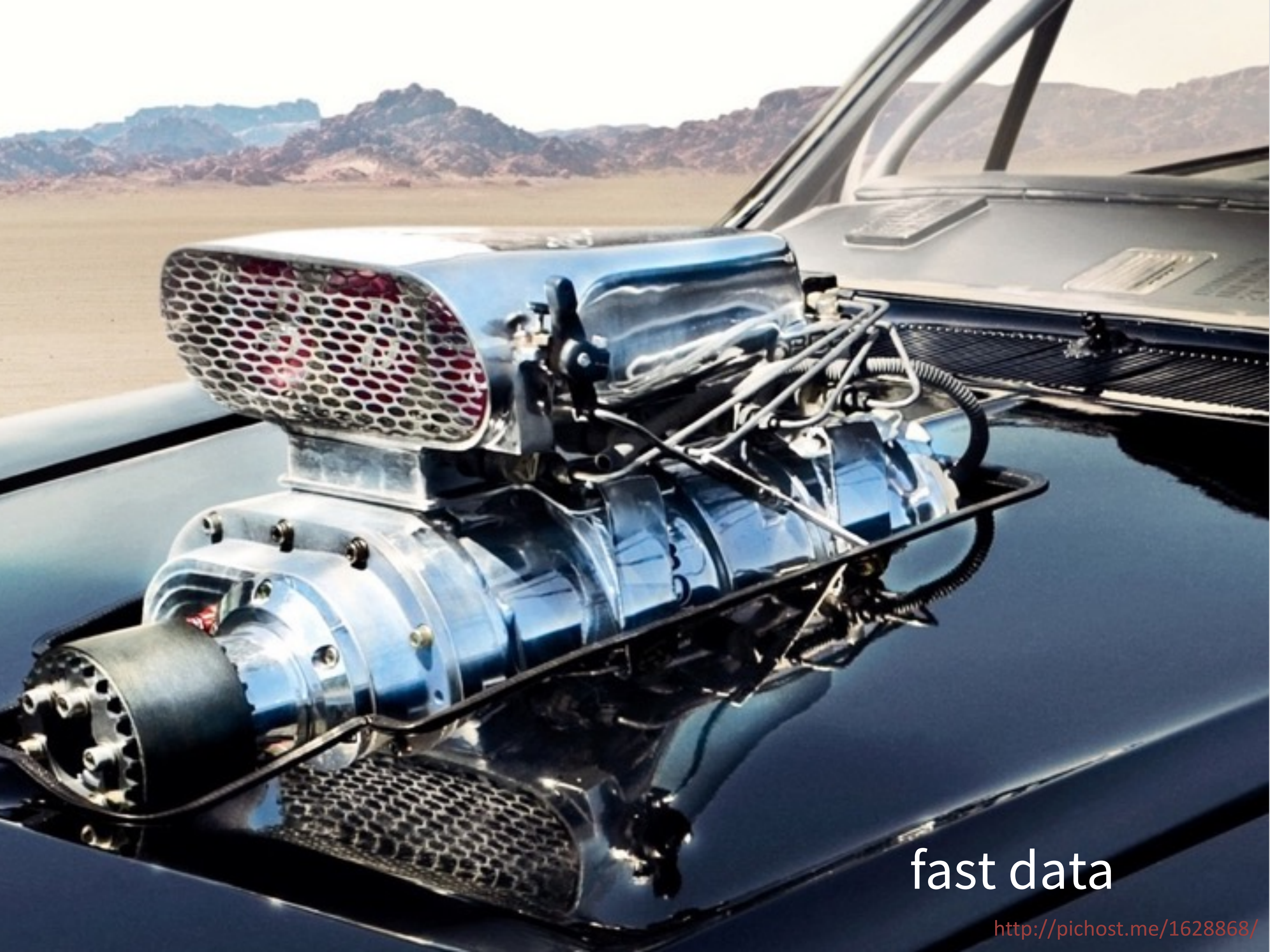
@nraychaudhuri

https://caneelian.files.wordpress.com/2010/11/lightbulb_dark.jpg

Big data

Big Data

Word for storing, managing and making money of very large dataset



fast data

http://pichost.me/1628868/

Fast Data

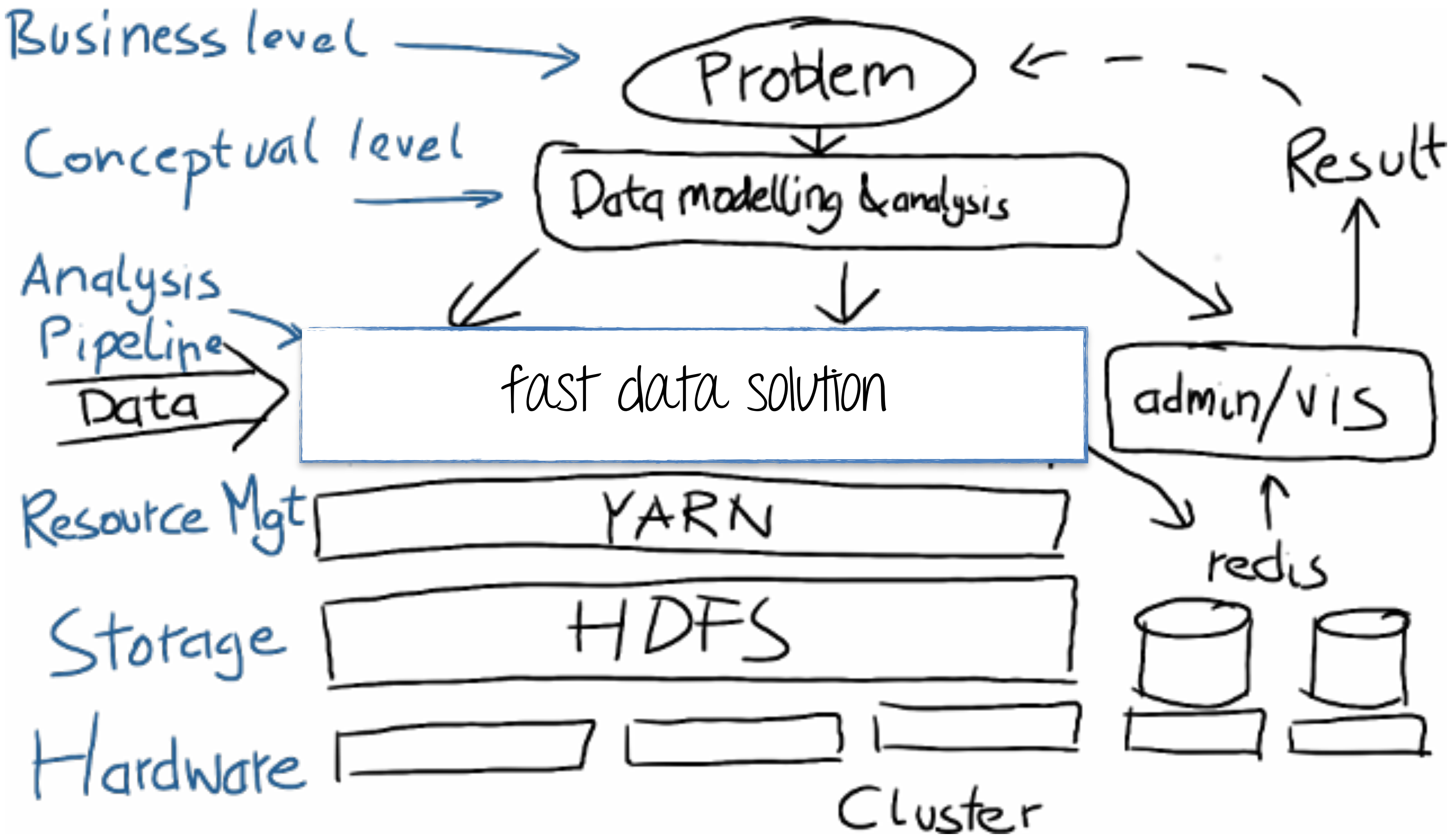
Word for using fast, real time analytics, online machine learning for profit

Definition

The phrase ***Fast Data*** captures this range of new systems and approaches, which balance various tradeoffs to deliver timely, cost-efficient streaming data processing.

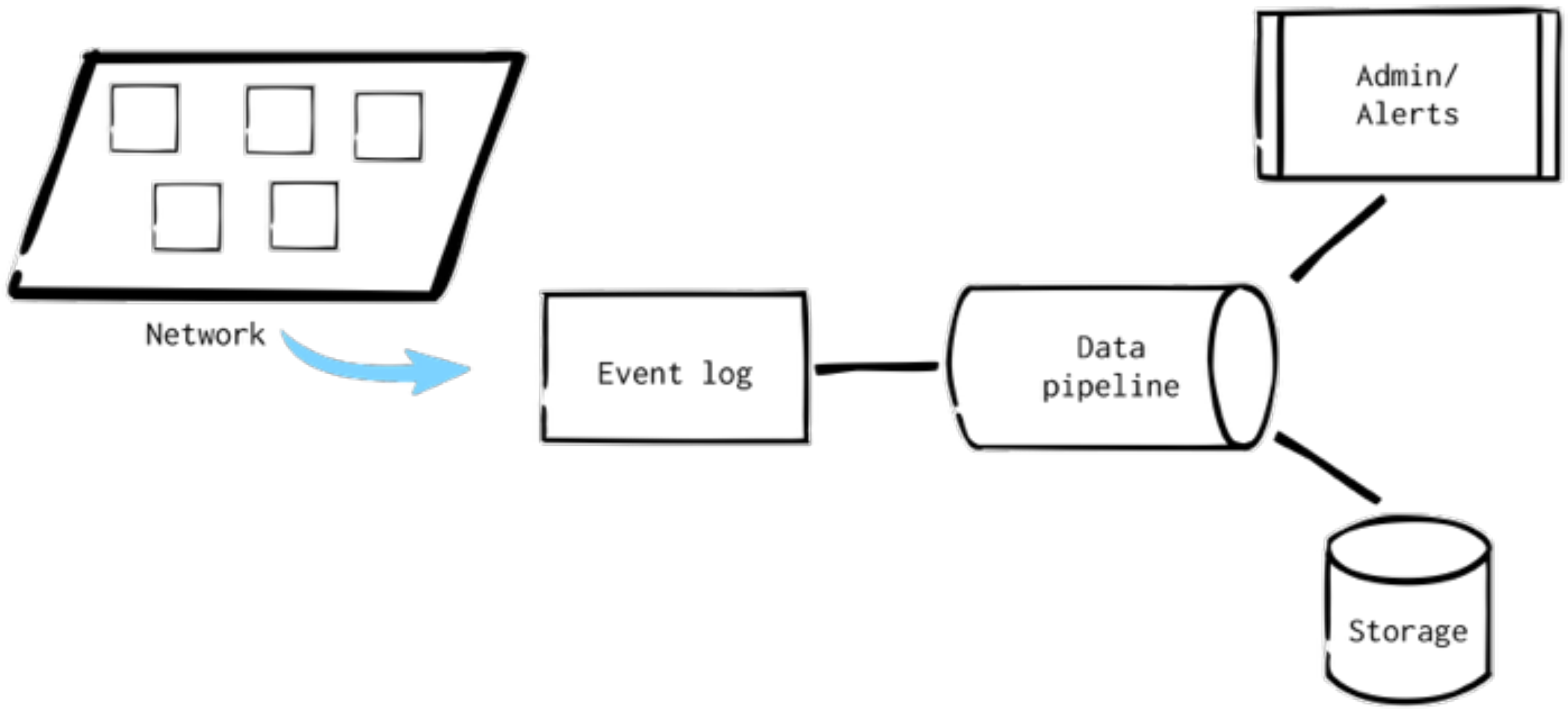
"You can't expect the value of data to just appear out of thin air. Data isn't fissile material. It doesn't spontaneously reach critical mass and start producing insights."

- Marko Karppinen

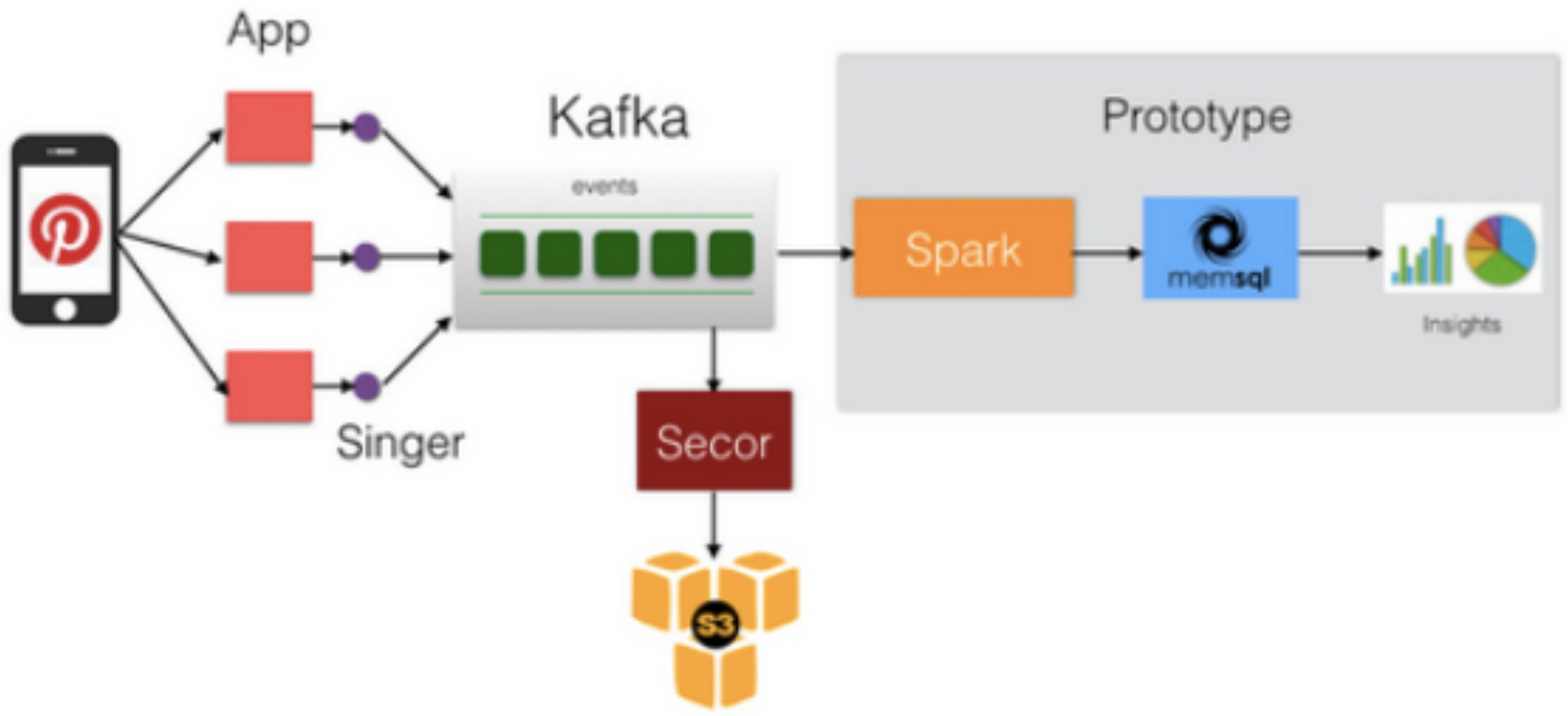


http://blog.mikiobraun.de/2014/10/parts-bug-no-car-big-data-infrastructure.html

Detecting network intrusion

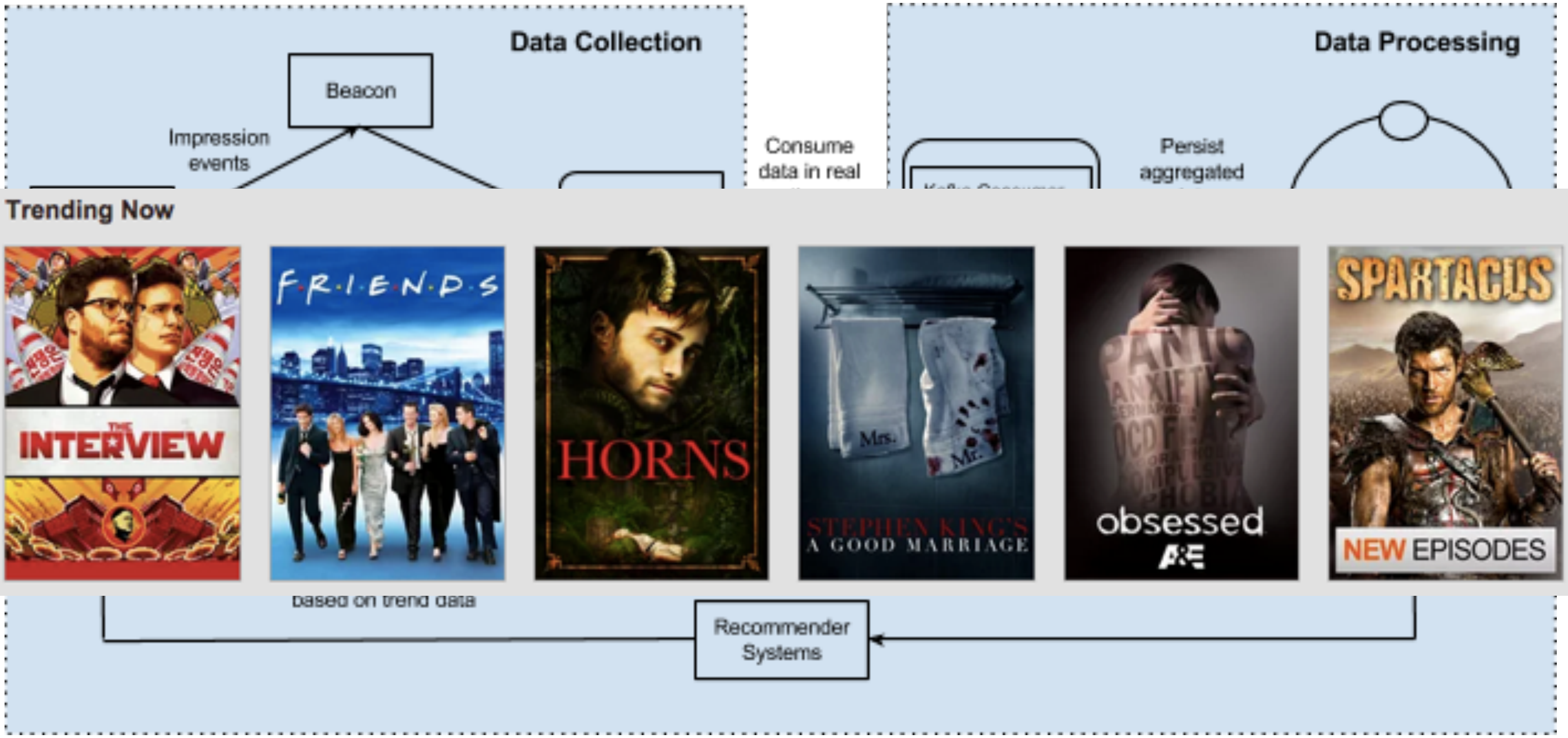


Faster data analysis @ Pinterest



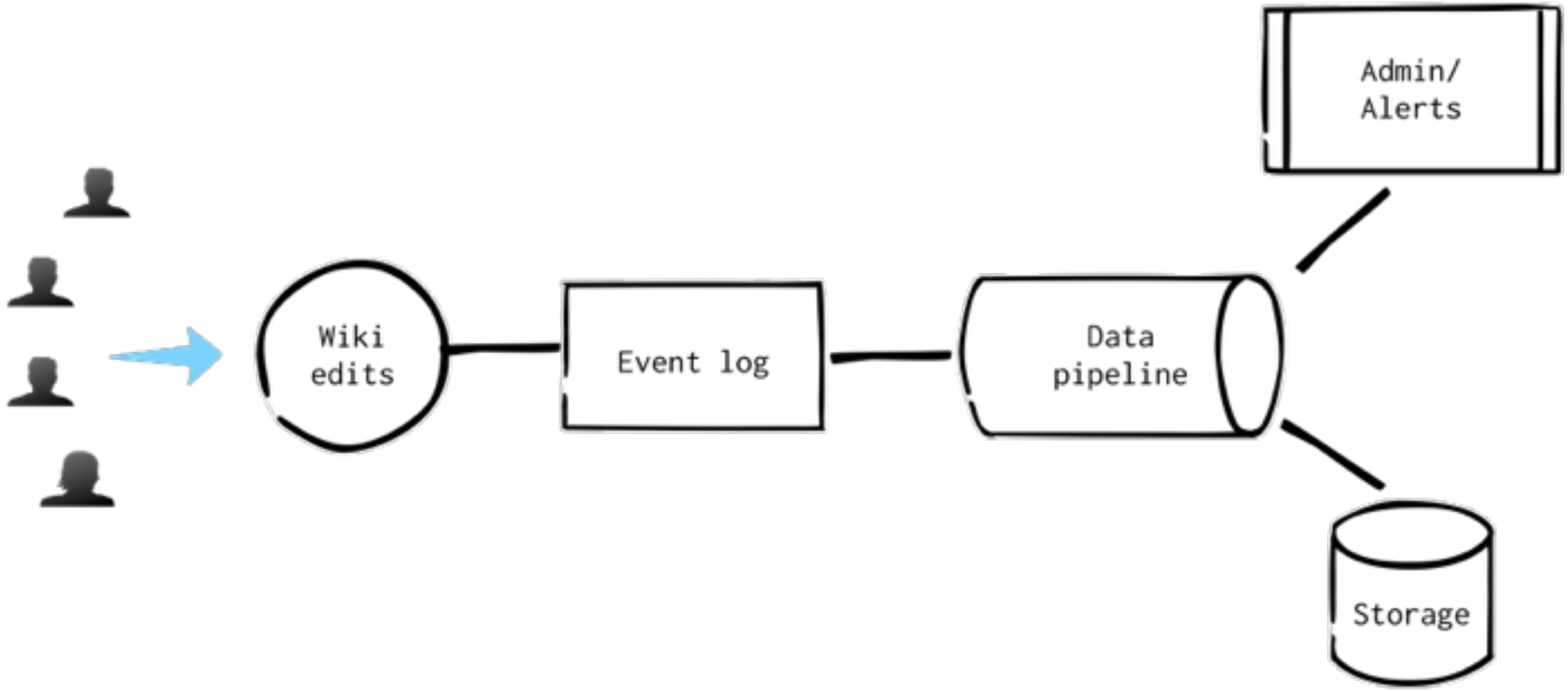
http://engineering.pinterest.com/post/111380432054/real-time-analytics-at-pinterest

Recommendation engine @ Netflix



http://techblog.netflix.com/2015/02/whats-trending-on-netflix.html

Predict breaking news

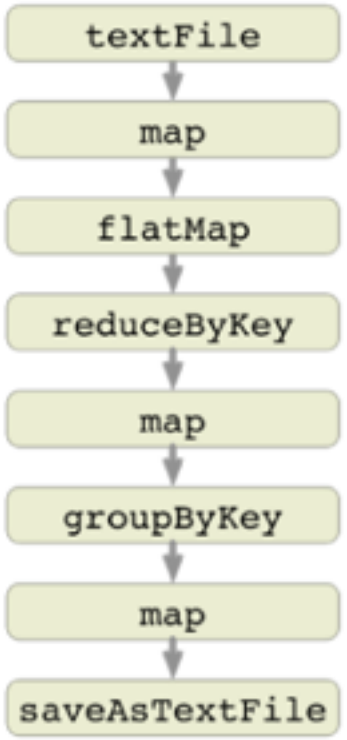


What big data/fast data has to do with functional programming?

Functional programming is the killer Paradigm for Data Apps

- Dean Wampler, Ph.D

Dataflow programming



```
sparkContext.textFile("/path/to/input")
.map { line =>
  val array = line.split(",", 2)
  (array(0), array(1))
}.flatMap {
  case (id, contents) => toWords(contents).map(w => ((w,id),1))
}.reduceByKey {
  (count1, count2) => count1 + count2
}.map {
  case ((word, path), n) => (word, (path, n))}
.groupByKey
.map {
  case (word, list) => (word, sortByCount(list))
}.saveAsTextFile("/path/to/output")
```

What is streaming?

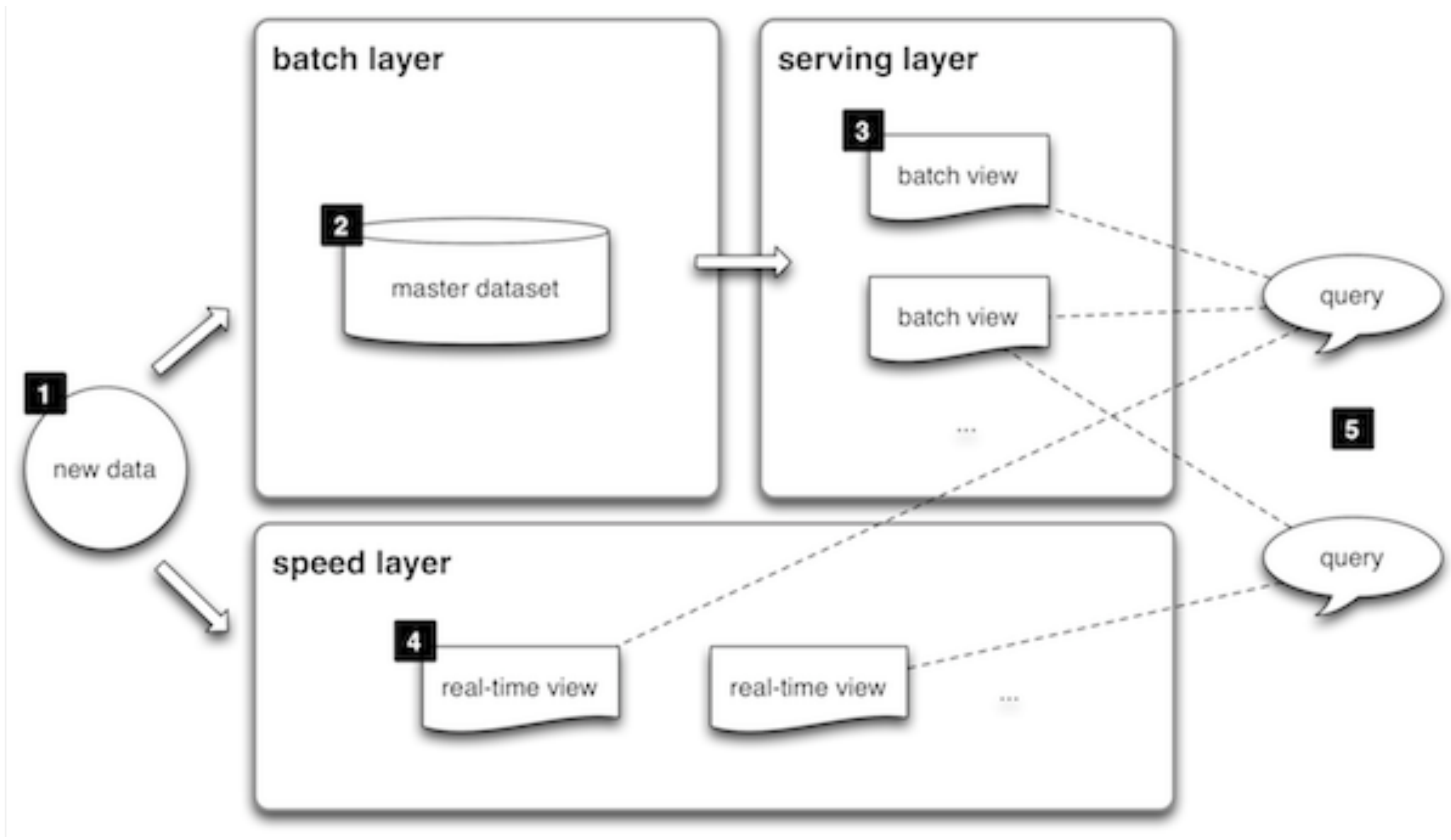
Define: Streaming

- A type of data processing engine that is designed with infinite data sets in mind
- Other common terms for streaming: unbounded data, low latency, approximate, and/or speculative results

Hurdles

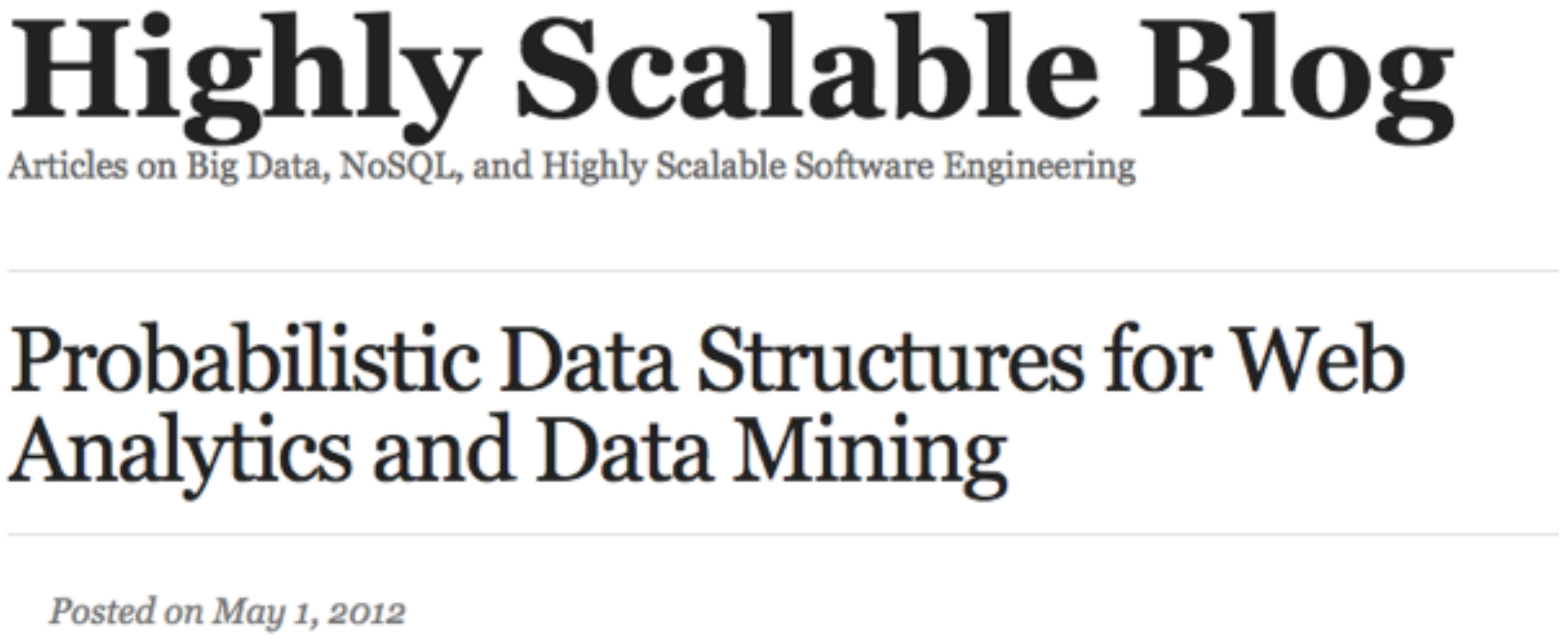
Lambda architecture

Typical Lambda architecture



Latency

new tools…



Highly Scalable Blog

Articles on Big Data, NoSQL, and Highly Scalable Software Engineering

Probabilistic Data Structures for Web Analytics and Data Mining

Posted on May 1, 2012

https://highlyscalable.wordpress.com/2012/05/01/probabilistic-structures-web-analytics-data-mining/

We need new tools

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2  SECONDS
```

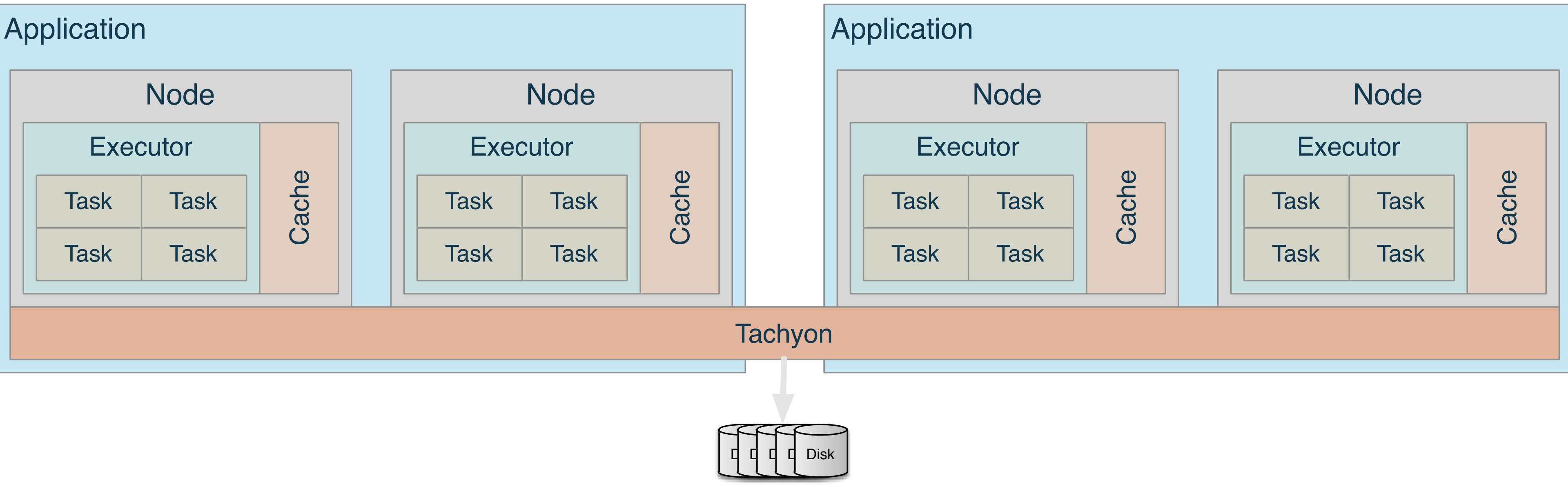
Queries with Time Bounds

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
ERROR 0.1 CONFIDENCE 95.0%
```

Queries with Error Bounds

BlinkDB

new tools…


Application
Node
Executor
Task
Task
Task
Task
Cache
Node
Executor
Task
Task
Task
Task
Cache
Application
Node
Executor
Task
Task
Task
Task
Cache
Node
Executor
Task
Task
Task
Task
Cache
Tachyon
Disk

Online ML

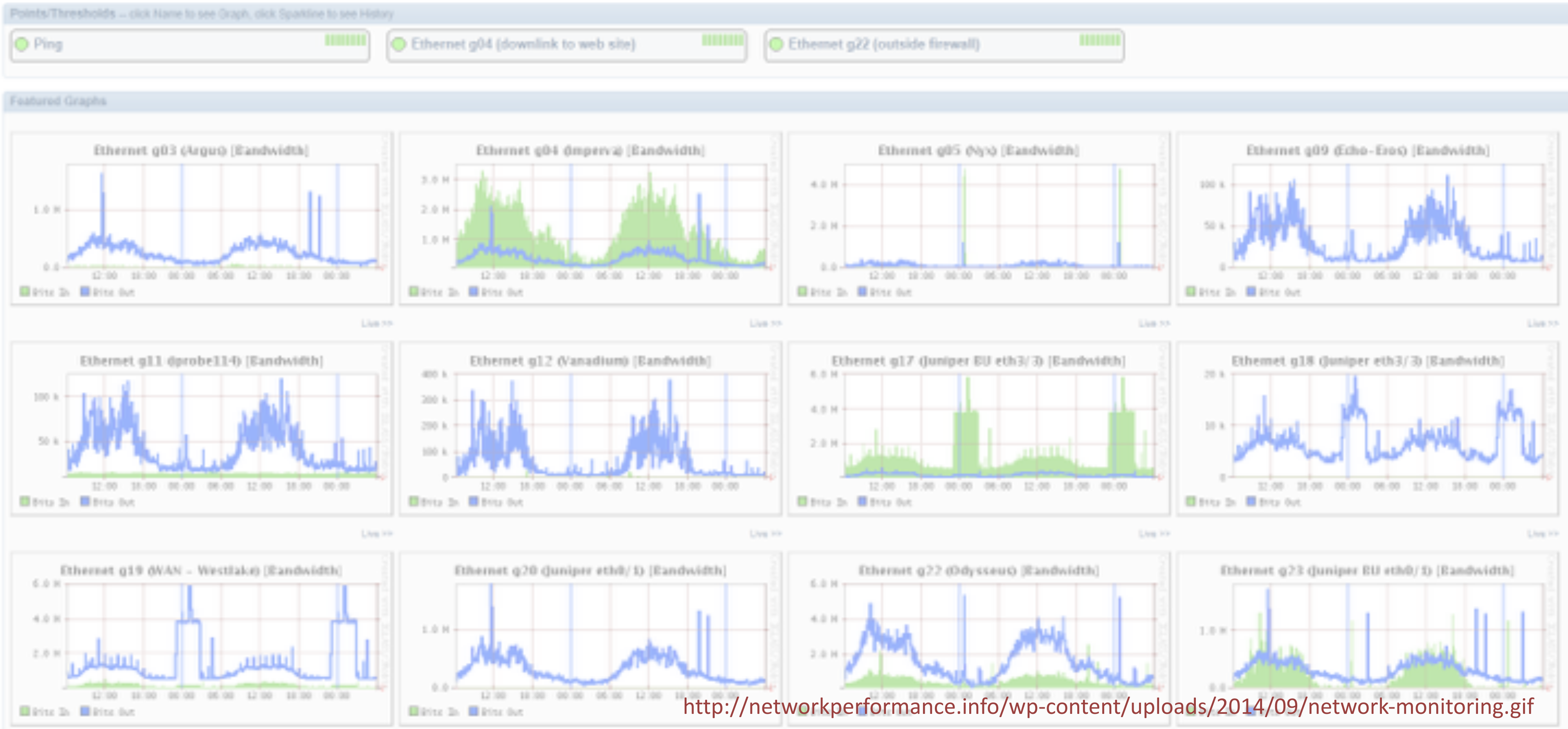
Computation model

```
public static class LineIndexMapper
 extends MapReduceBase
 implements Mapper<LongWritable, Text,
                   Text, Text> {
 private final static Text word =
  new Text();
 private final static Text location =
  new Text();

 public void map(
  LongWritable key, Text val,
  OutputCollector<Text, Text> output,
  Reporter reporter) throws IOException {

  FileSplit fileSplit =
   (FileSplit)reporter.getInputSplit();
```

Monitoring



Points/Thresholds – click Name to see Graph, click Sparkline to see History
Ping
Ethernet g04 (downlink to web site)
Ethernet g22 (outside firewall)
Featured Graphs
Ethernet g03 (Argus) [Bandwidth]
Ethernet g04 (Imperva) [Bandwidth]
Ethernet g05 (Nyx) [Bandwidth]
Ethernet g09 (Echo-Eros) [Bandwidth]
Ethernet g11 (probe114) [Bandwidth]
Ethernet g12 (Vanadium) [Bandwidth]
Ethernet g17 (Juniper BU eth3/3) [Bandwidth]
Ethernet g18 (Juniper eth3/3) [Bandwidth]
Ethernet g19 (WAN - Westlake) [Bandwidth]
Ethernet g20 (Juniper eth0/1) [Bandwidth]
Ethernet g22 (Odysseus) [Bandwidth]
Ethernet g23 (Juniper BU eth0/1) [Bandwidth]
Bits In
Bits Out
Live >>

http://networkperformance.info/wp-content/uploads/2014/09/network-monitoring.gif

Wish list

Streaming platform feature list

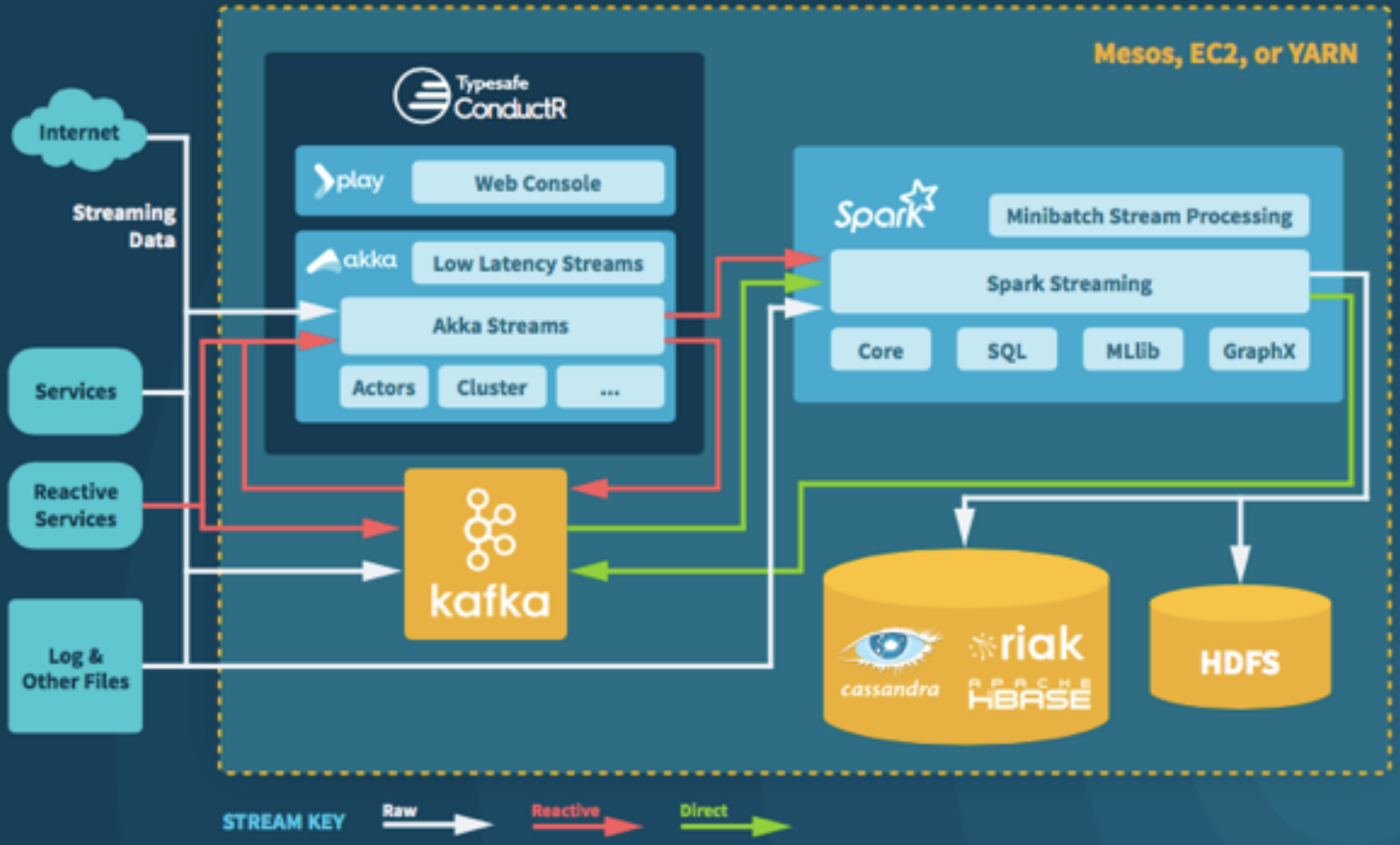
- Should be programmer friendly (Scala and FP)
- Should be fault tolerant
- Should support high throughput/low latency
- Should integrate with batch systems (Hadoop)
- Should run machine learning algorithm
- Should not overwhelm the consumer
- Should provide integration points for other streaming systems

And that streaming platform is…

not there yet

but...something is emerging

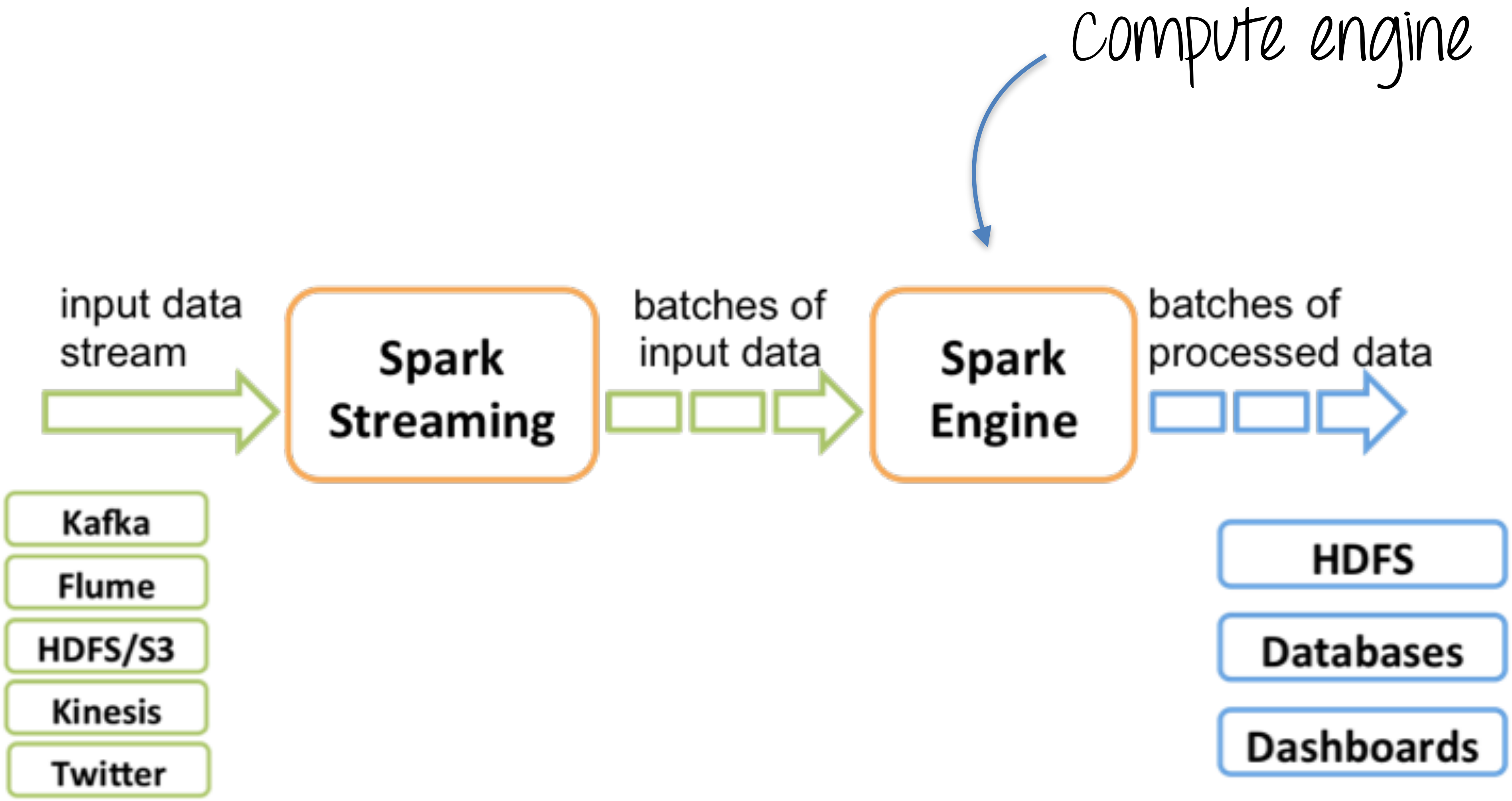
Fast data platform



Spark Streaming

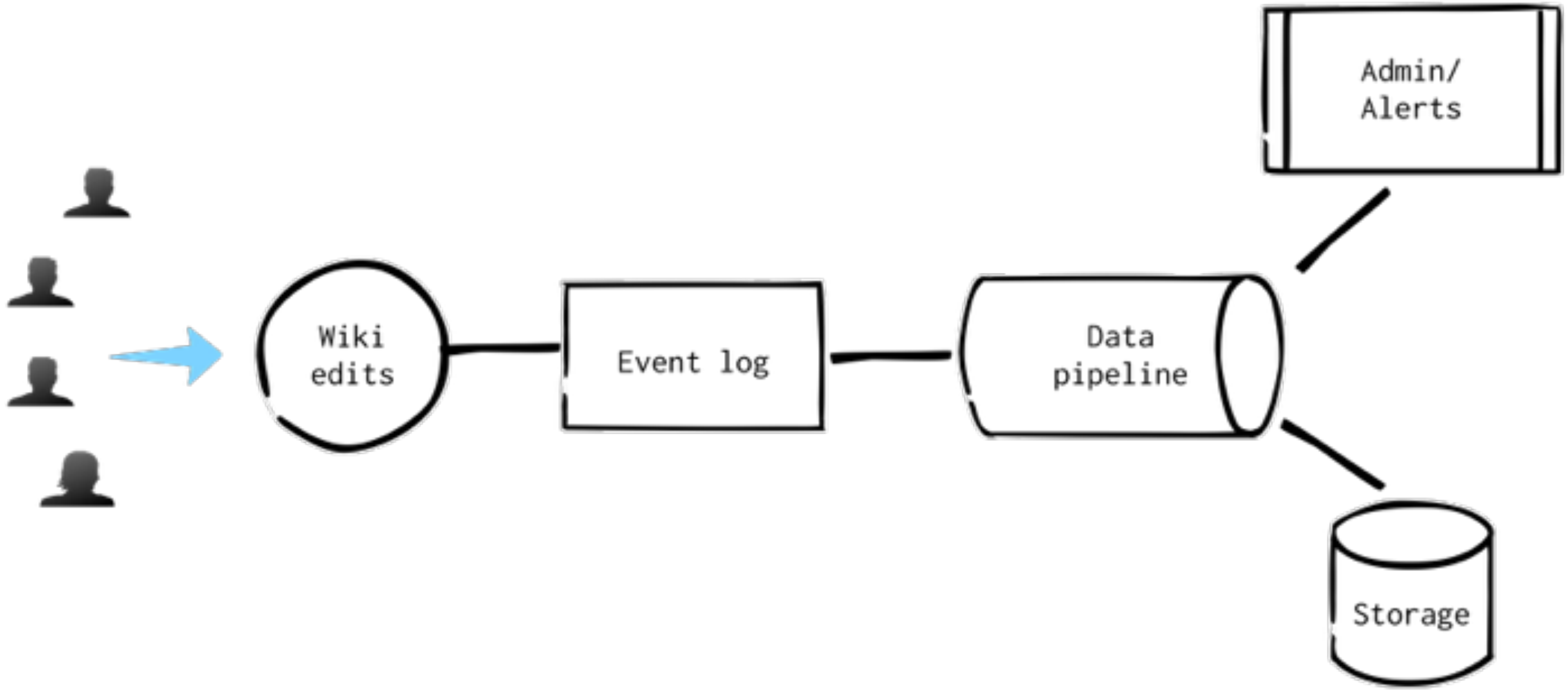
Captures time slices of events

Data pipeline

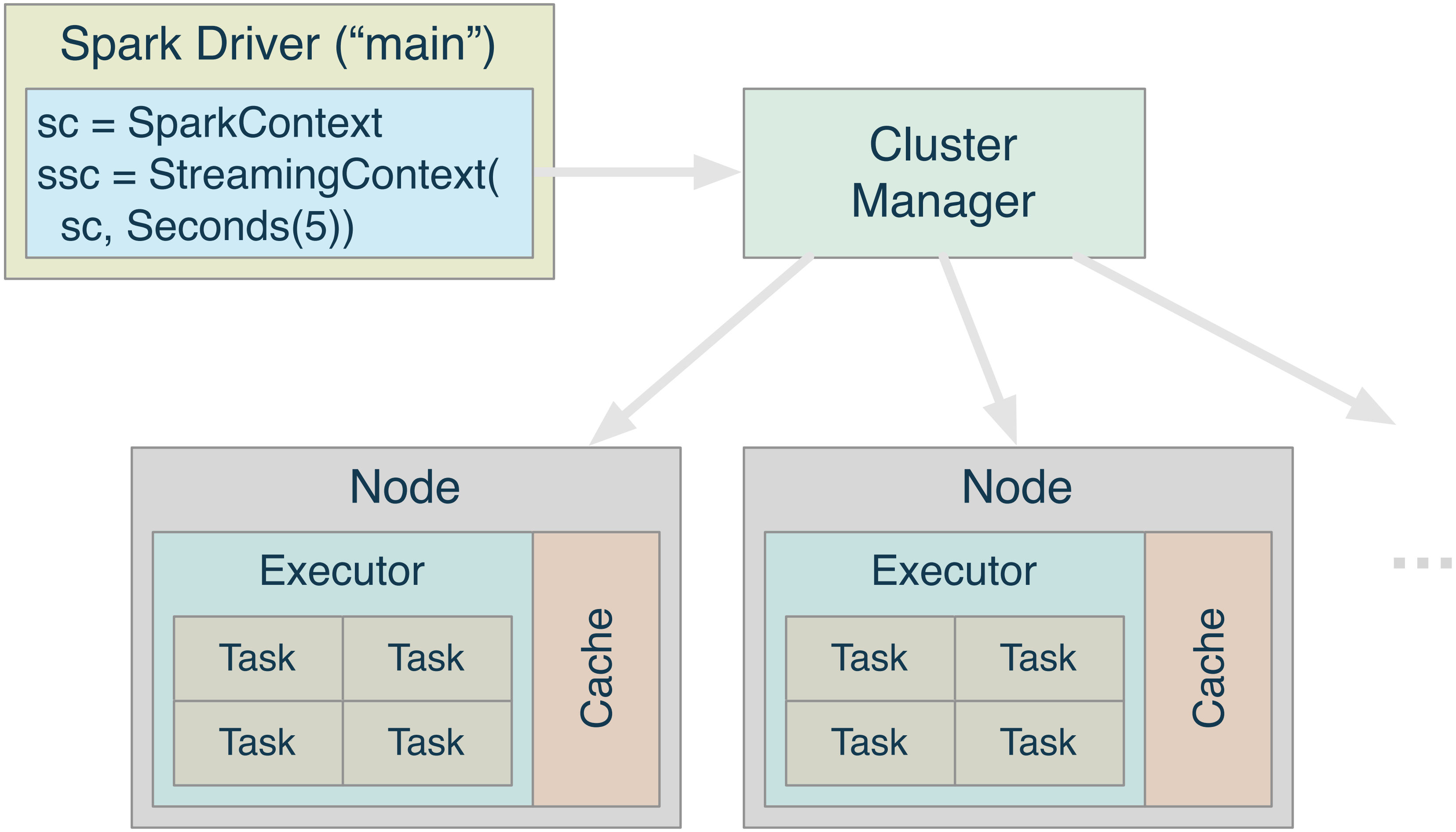


Show me the code

Predict breaking news



Under the hood



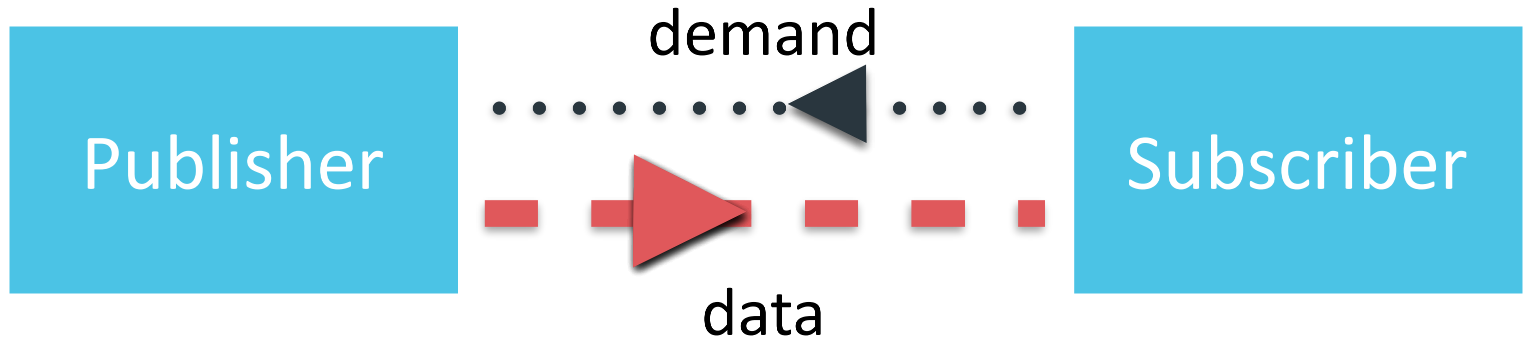
```
sc = SparkContext
ssc = StreamingContext(
  sc, Seconds(5))
```

Reactive Streams

Definition

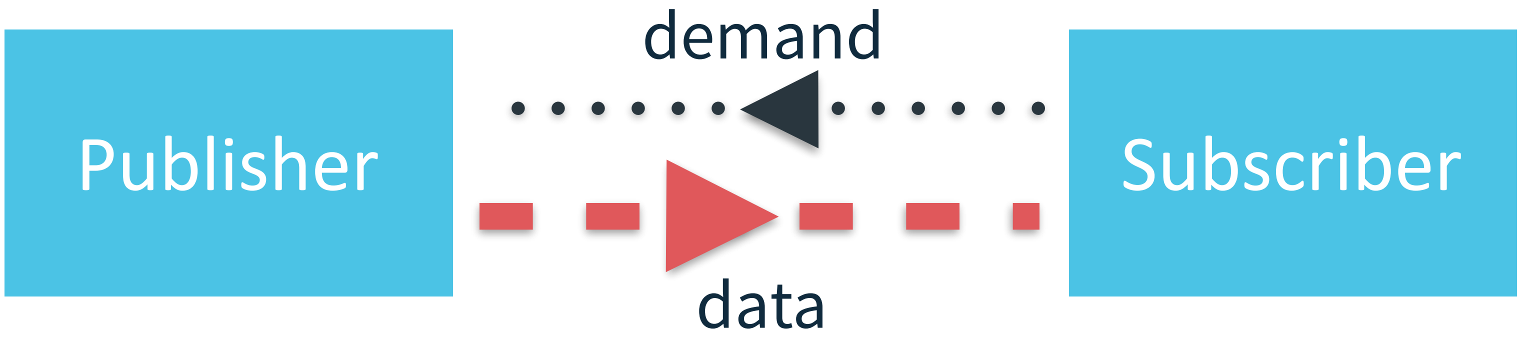
A standard for asynchronous stream processing with non-blocking back pressure

Supply and Demand

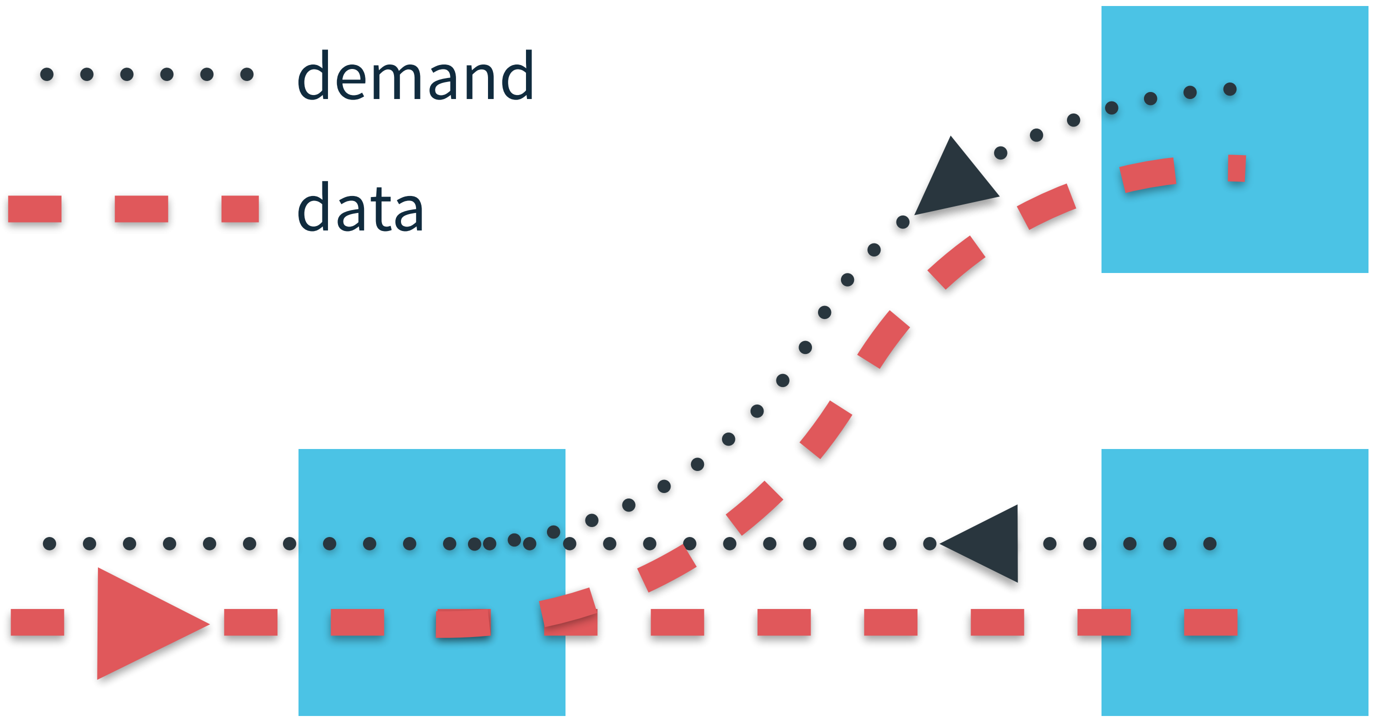


Dynamic Push–Pull

- "push" behavior when consumer is faster
- "pull" behavior when producer is faster

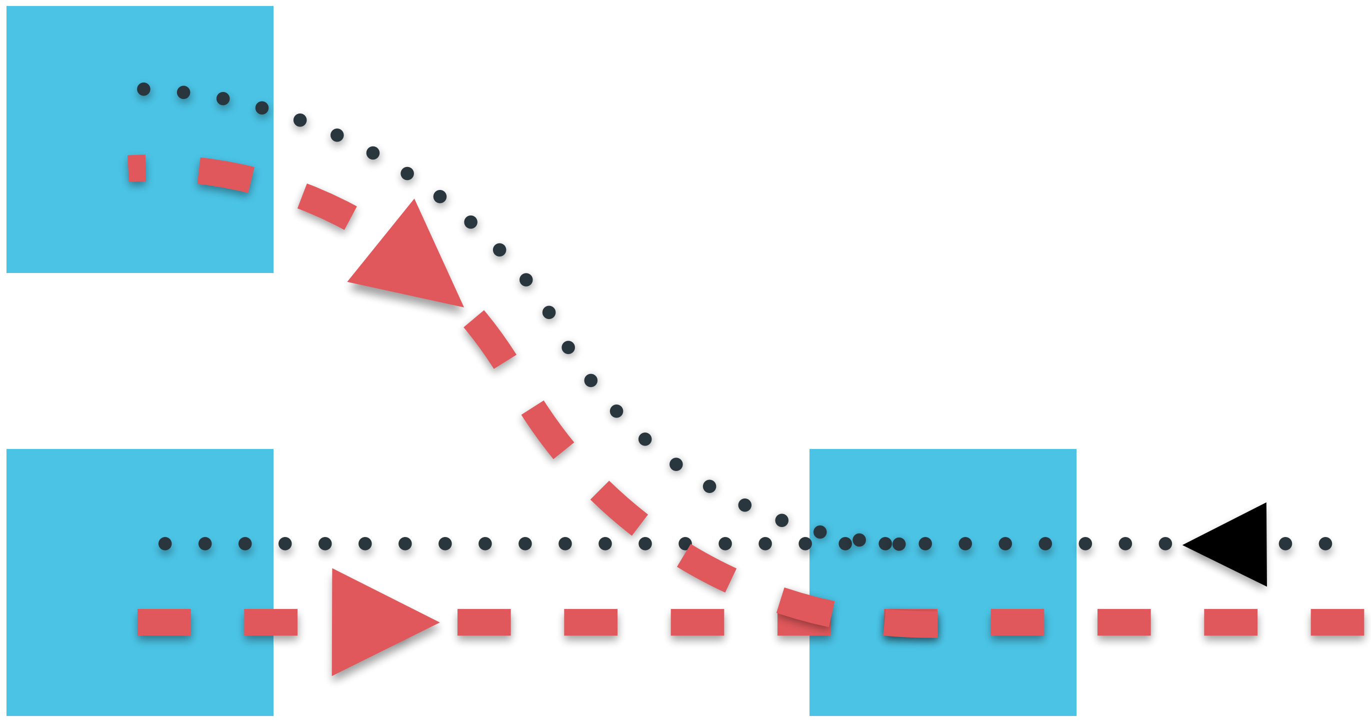


Explicit Demand: Tailored Flow Control



splitting the data means merging the demand

Explicit Demand: Tailored Flow Control

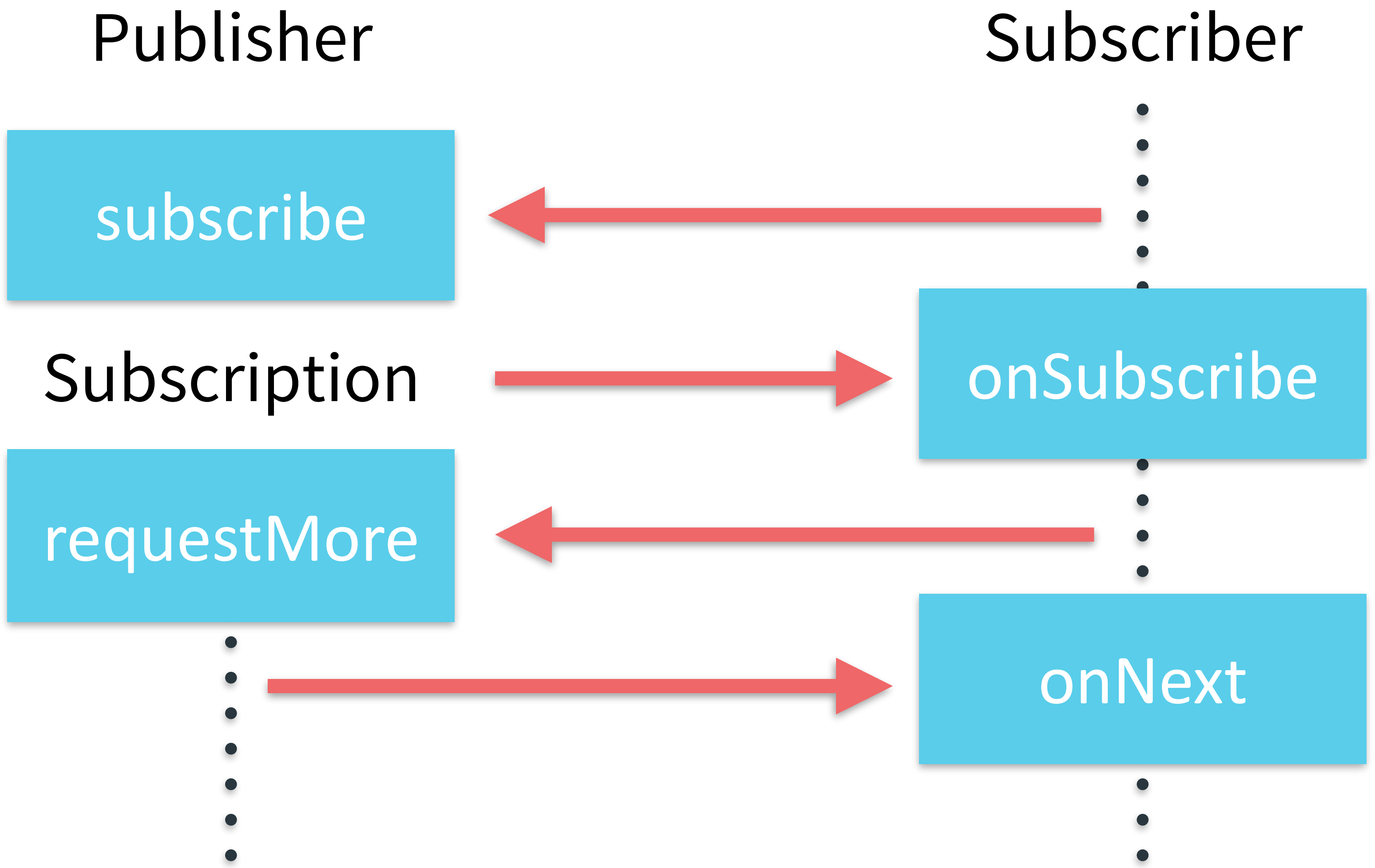


merging the data means splitting the demand

The Meat

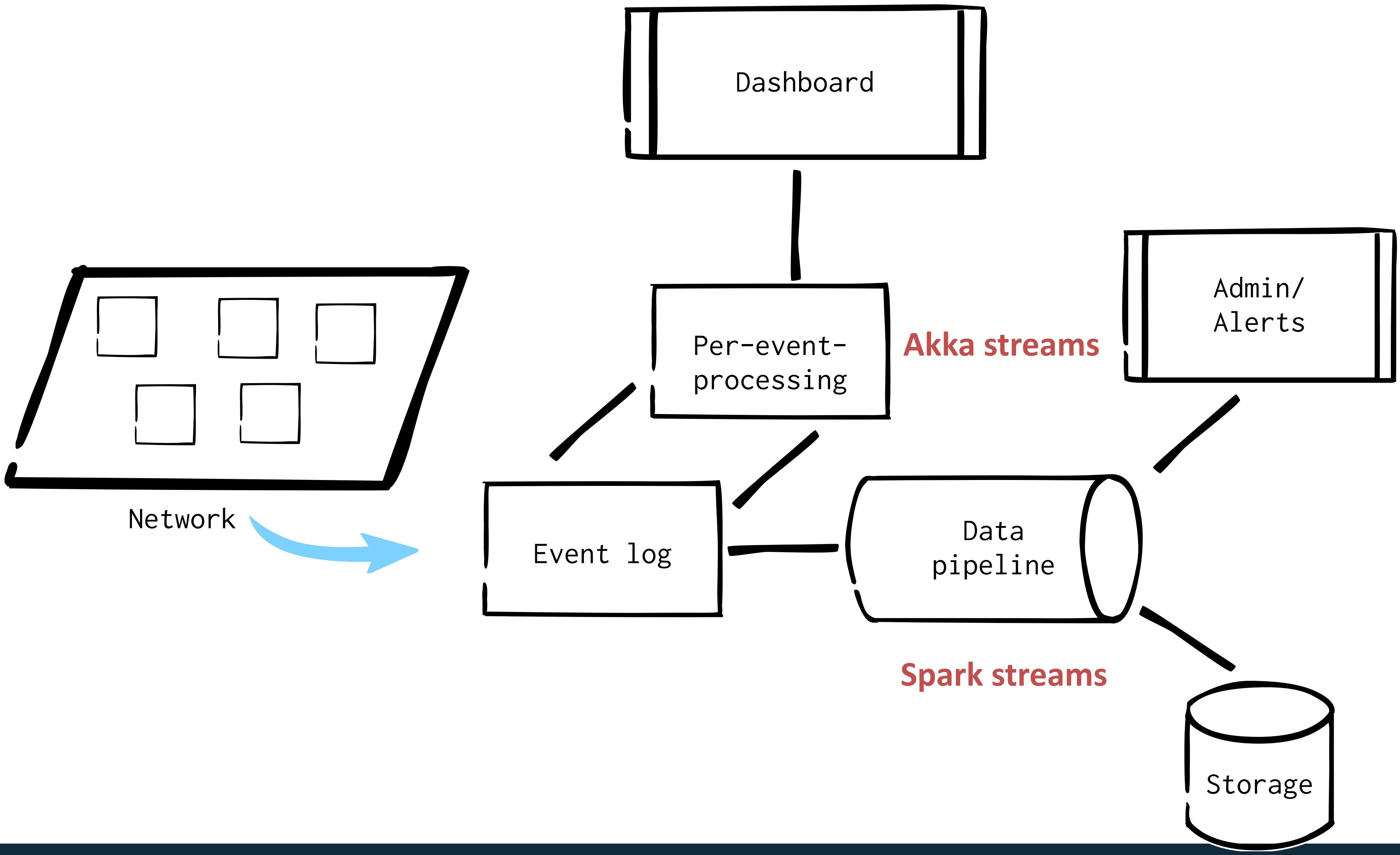
```
trait Publisher[T] {
  def subscribe(sub: Subscriber[T]): Unit
}
trait Subscription {
  def requestMore(n: Int): Unit
  def cancel(): Unit
}
trait Subscriber[T] {
  def onSubscribe(s: Subscription): Unit
  def onNext(elem: T): Unit
  def onError(thr: Throwable): Unit
  def onComplete(): Unit
}
```

How does it Connect?



Network intrusion

Detecting network intrusion



Typesafe

```scala
val sc = new SparkContext("local[3]", "Intro")
val ssc = new StreamingContext(sc, Seconds(1))
ssc.checkpoint("data/checkpoint")

val kmeans = new StreamingKMeans()
  .setK(10)
  .setRandomCenters(3, 100.0)
  .setHalfLife(5, "batches")
```

```
val receiver = SparkEnv.get.actorSystem.actorOf(trainingDataReceiver,
  "training-data-source")
val trainingStream =
  ssc.actorStream[Vector](TrainingDataStream.props(receiver, kmeans),
  "training-stream")

val rawData: DStream[(String, Vector)] = KafkaUtils
  .createStream(ssc = ssc, kafkaParams, topics,
    StorageLevel.MEMORY_ONLY)
  .map(_._2).map(line => (line, labelAndVector(line)._2))
```

```
kmeans.trainOn(trainingStream.map(_._2))
kmeans.predictOnValues(networkStream).print()

ssc.start()
ssc.awaitTermination()
```

Q & Option[A]

@nraychaudhuri

Typesafe

©Typesafe 2014 – All Rights Reserved