Data Pipeline at Tapad

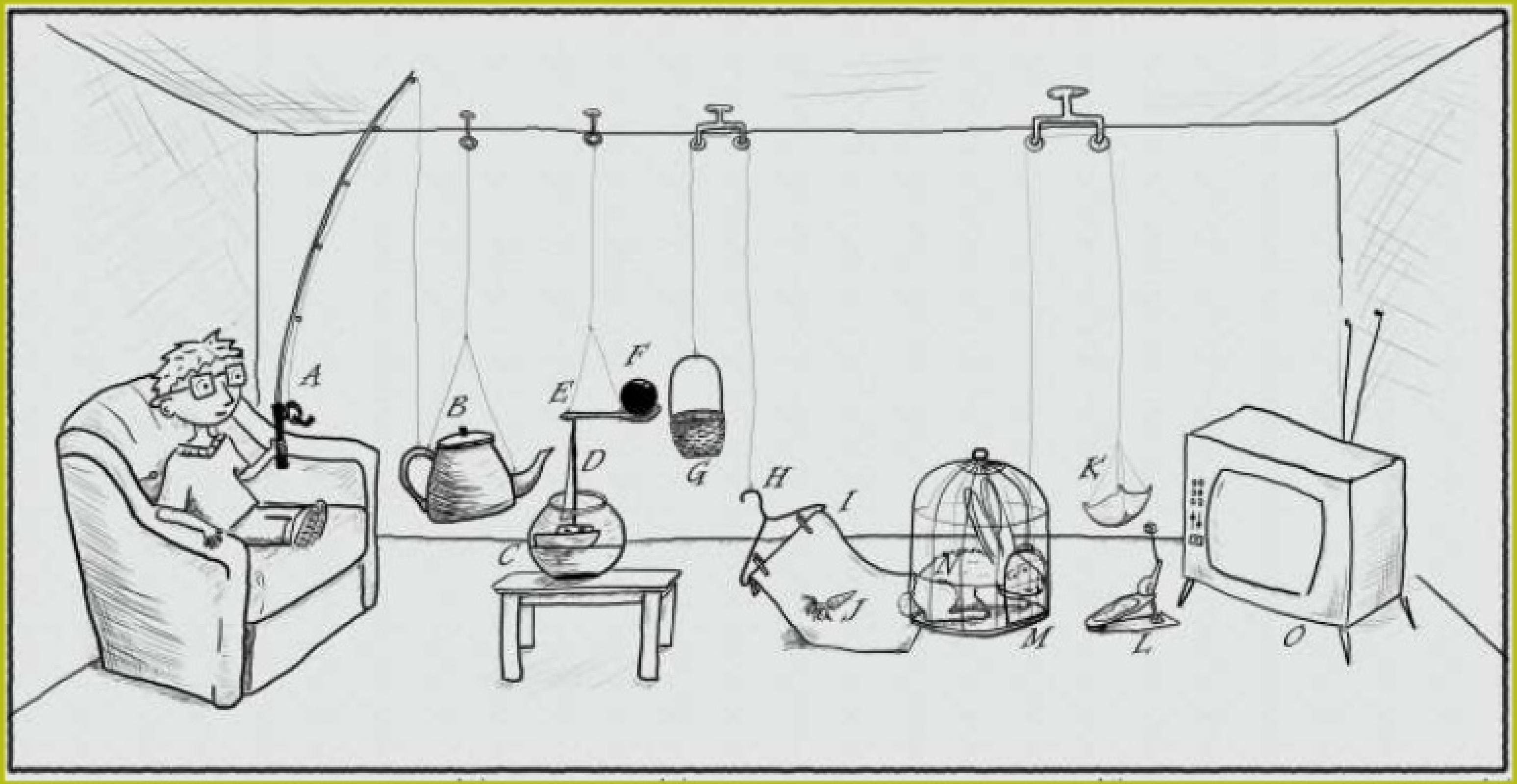
@tobym
@TapadEng

Who am I?

Toby Matejovsky
First engineer hired at Tapad 3+ years ago
Scala developer

@tobym

What are we talking about?



Outline

- What Tapad does
- Why bother with a data pipeline?
- Evolution of the pipeline
- Day in the life of a analytics pixel
- What’s next

What Tapad Does

Cross-platform advertising and analytics
Process billions of events per day

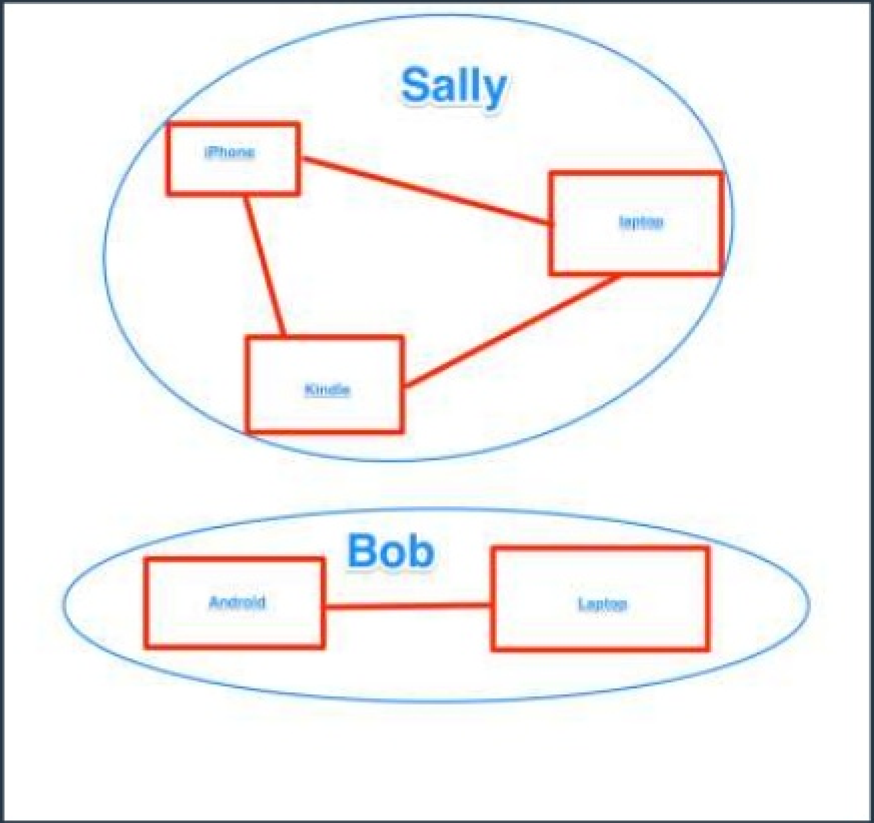
Cross platform?

Device Graph

Node=device
edge=inferred connection

Billion devices
Quarter billion edges

85+% accuracy



Why a Data Pipeline?

Graph building
Sanity while processing big data
Decouple components
Data accessible at multiple stages

Graph Building

Realtime mode, but don’t impact bidding latency

Batch mode

Sanity

Billions of events, terabytes of logs per day
Don’t have NSA’s budget
Clear data retention policy
Store aggregations

Decouple Components

Bidder only bids, graph-building process only builds graph

Data stream can split and merge

Data accessible at multiple stages

Logs on edge of system
Local spool of data
Kafka broker
Consumer local spool
HDFS

Evolution of the Data Pipeline

Dark Ages: Monolithic process, synchronous process

Renaissance: Queues, asynchronous work in same process

Age of Exploration: Inter-process comm, ad hoc batching

Age of Enlightenment: Standardize on Kafka and Avro

Dark Ages

Monolithic process, synchronous process

It was fast enough, and we had to start somewhere.

Renaissance

Queues, asynchronous work in same process

No, it wasn’t fast enough.

Age of Exploration

Inter-process communication, ad hoc batching

Servers at the edge batch up events, ship them to another service.

Age of Enlightenment

Standardize on Kafka and Avro

Properly engineered and supported, reliable

Age of Enlightenment

Standardize on Kafka and Avro

Properly engineered and supported, reliable

Tangent!

Batching, queues, and serialization

Batching

Batching is great, will really help throughput

Batching != slow

Queues

Queues are amazing, until they explode and destroy the Rube Goldberg machine.

“I’ll just increase the buffer size.”

- spoken one day before someone ended up on double PagerDuty rotation

Care and feeding of your queue

Monitor

Back-pressure

Buffering

Spooling

Degraded mode

Serialization - Protocol Buffers

Tagged fields

Sort of self-describing

required, optional, repeated fields in schema

“Map” type:

```
message StringPair {
    required string key = 1;
    optional string value = 2;
}
```

Serialization - Avro

Optional field: union { null, long } user_timestamp = null;

Splittable (Hadoop world)

Schema evolution and storage

pixel server - pixel ingress - kafka - consumer - hdfs - hadoop jobs

Day in the life of a pixel

Browser loads pixel from pixel server

Pixel server immediately responds with 200 and transparent gif, then serializes requests into a batch file

Batch file ships every few seconds or when the file reaches 2K pixel server - **pixel ingress** - kafka - consumer - hdfs - hadoop jobs

Day in the life of a pixel

Pixel ingress server receives 2 kilobyte file containing serialized web requests.

Deserialize, process some requests immediately (update database), then convert into Avro records with schema hash header, and publish to various Kafka topics pixel server - pixel ingress - **kafka** - consumer - hdfs - hadoop jobs

Day in the life of a pixel

Producer client figures out where to publish via the broker they connect to

Kafka topics are partitioned into multiple chunks, each has a master and slave and are on different servers to survive an outage.

Configurable retention based on time

Can add topics dynamically

pixel server - pixel ingress - kafka - **consumer** - hdfs - hadoop jobs

Day in the life of a pixel

Consumer processes are organized into groups

Many consumer groups can read from same Kafka topic

Plugins:

```
trait Plugin[A] {
  def onStartup(): Unit
  def onSuccess(a: A): Unit
  def onFailure(a: A): Unit
  def onShutdown(): Unit
}
```

GraphitePlugin, BatchingLogfilePlaybackPlugin, TimestampDrivenClockPlugin, BatchingTimestampDrivenClockPlugin, …

pixel server - pixel ingress - kafka - **consumer** - hdfs - hadoop jobs

Day in the life of a pixel

```
trait Plugins[A] {

  private val _plugins = ArrayBuffer.empty[Plugin[A]]

  def plugins: Seq[Plugin[A]] = _plugins

  def registerPlugin(plugin: Plugin[A]) = _plugins += plugin
}
```

pixel server - pixel ingress - kafka - **consumer** - hdfs - hadoop jobs

Day in the life of a pixel

```
object KafkaConsumer {
  sealed trait Result {
    def notify[A](plugins: Seq[Plugin[A]], a: A): Unit
  }

  case object Success extends Result {
    def notify[A](plugins: Seq[Plugin[A]], a: A) {
      plugins.foreach(_.onSuccess(a))
    }
  }
}
```

pixel server - pixel ingress - kafka - **consumer** - hdfs - hadoop jobs

```scala
/** Decorate a Function1[A, B] with retry logic */
case class Retry[A, B](maxAttempts: Int, backoff: Long)(f: A => B){
 def apply(a: A): Result[A, B] = {
  def execute(attempt: Int, errorLog: List[Throwable]): Result[A, B] = {
   val result = try {
    Success(this, a, f(a))
   } catch {
    … Failure(this, a, e :: errorLog) …
   }

   result match {
    case failure @ Failure(_, _, errorLog) if errorLog.size < maxAttempts =>
     val _backoff = (math.pow(2, attempt) * backoff).toLong
     Thread.sleep(_backoff)        // wait before the next invocation
     execute(attempt + 1, errorLog) // try again
    case failure @ Failure(_, _, errorLog) =>
     failure
   }
  }
  execute(attempt = 0, errorLog = Nil)
 }
```

pixel server - pixel ingress - kafka - consumer - **hdfs** - hadoop jobs

Day in the life of a pixel

Consumers log into “permanent storage” in HDFS.

File format is Avro, written in batches.

Data retention policy is essential.

pixel server - pixel ingress - kafka - consumer - hdfs - **hadoop jobs**

Day in the life of a pixel

Hadoop 2 - YARN

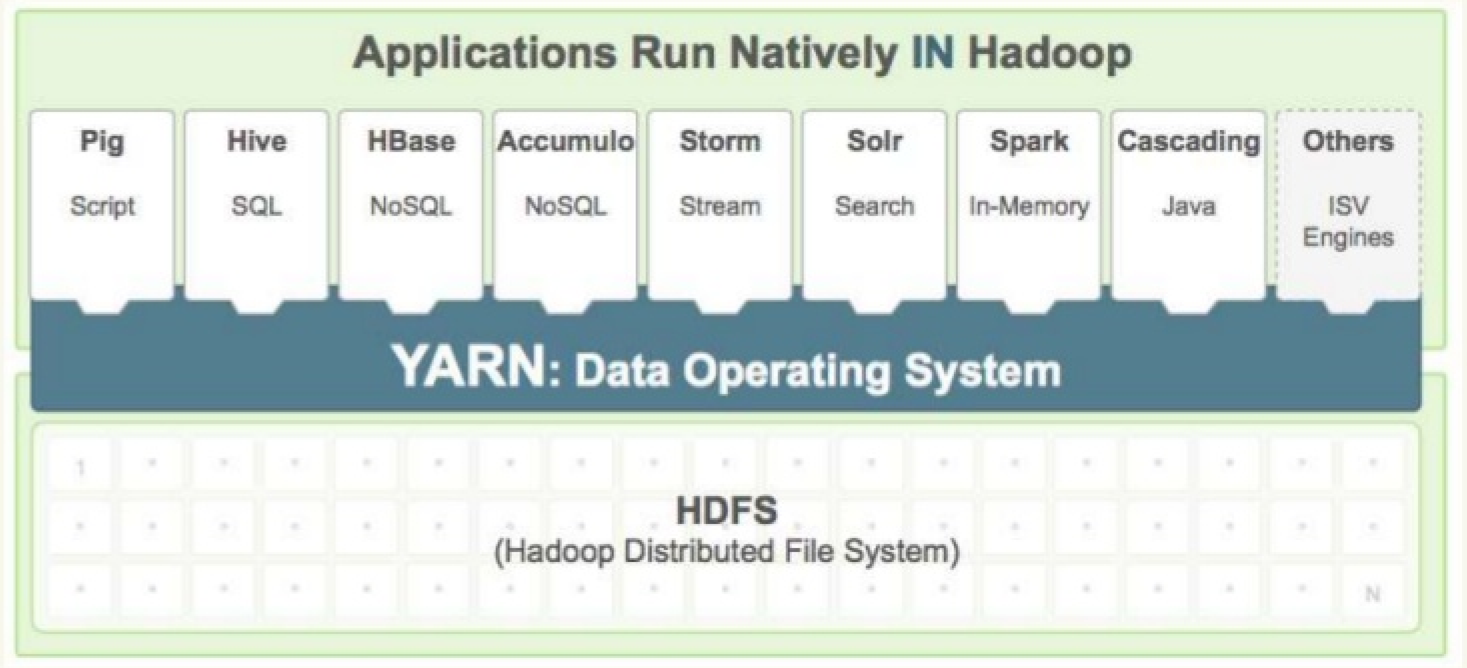
Scalding to write map-reduce jobs easily

Rewrite Avro files as Parquet

Oozie to schedule regular jobs

pixel server - pixel ingress - kafka - consumer - hdfs - **hadoop jobs**

YARN



pixel server - pixel ingress - kafka - consumer - hdfs - **hadoop jobs**

Scalding

```
class WordCountJob(args : Args) extends Job(args) {
  TextLine( args("input") )
    .flatMap('line -> 'word) { line : String => tokenize(line) }
    .groupBy('word) { _.size }
    .write( Tsv( args("output") ) )

  // Split a piece of text into individual words.
  def tokenize(text : String) : Array[String] = {
    // Lowercase each word and remove punctuation.
    text.toLowerCase.replaceAll("[^a-zA-Z0-9\\s]", "").split("\\s+")
  }
}
```

Parquet

Column-oriented storage for Hadoop

Nested data is okay

Projections

Predicates

pixel server - pixel ingress - kafka - consumer - hdfs - **hadoop jobs**

Parquet

```
val requests = ParquetAvroSource
  .project[Request](args("requests"), Projection[Request]("header.query_params", "partner_id"))
  .read
  .sample(args("sample-rate").toDouble)
  .mapTo('Request -> ('queryParams, 'partnerId)) { req: TapestryRequest =>
    (req.getHeader.getQueryParams, req.getPartnerId)
  }
```

pixel server - pixel ingress - kafka - consumer - hdfs - **hadoop jobs**

Oozie

```
<workflow-app name="combined_queries" xmlns="uri:oozie:workflow:0.3">
    <start to="devices-location"/>
    <!--<start to="export2db"/>-->

    <action name="devices-location">
        <shell xmlns="uri:oozie:shell-action:0.1">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>

            <exec>hadoop</exec>
            <argument>fs</argument>
            <argument>-cat</argument>
            <argument>${devicesConfig}</argument>

          <capture-output/>
        </shell>

        <ok to="networks-location"/>
        <error to="kill"/>
      </action>
```

pixel server - pixel ingress - kafka - **consumer** - hdfs - **hadoop jobs**

Day in the life of a pixel

Near real-time consumers and batch hadoop jobs generate data cubes from incoming events and save those aggregations into Vertica for fast and easy querying with SQL.

Stack summary

Scala, Jetty/Netty, Finagle
Avro, Protocol Buffers, Parquet
Kafka
Zookeeper
Hadoop - YARN and HDFS
Vertica
Scalding
Oozie, Sqoop

What’s next?

Hive
Druid
Impala
Oozie alternative

Thank You

yes, we're hiring! :)

@tobym

@TapadEng

Toby Matejovsky, Director of Engineering
toby@tapad.com
@tobym

TAPAD