



Enabling a Large Analytic Userbase on Hadoop & Co

Justin Coffey, j.coffey@criteo.com, @jqcoffey — GOTO Amsterdam 2015

What we'll be taking (lots) about

I'll be presenting an overview of our analytics stack and how it was designed to enable our internal analytics user base get to the data they need when they need it.

We'll be covering:

- The major technical components and how we use them
- Some of our interactions with our stakeholders and how that impacts our roadmap
- Some takeaways about what we've learned along the way

A note

Ask your questions at any time.

On with it, then. Who is Criteo?

Criteo is the biggest tech company
you've never heard of.

Criteo: what we do

Criteo is a Performance Advertising Company

We drive conversions for our advertisers across multiple channels

Mobile
Banner Ads **Search**
Email

We pay for displays, charge for traffic to our advertisers and optimize for conversions

With a **90% retention rate**, our clients **love** us!

Criteo: the human scale

- **1500+** total employees
- **30+** offices world wide
- **250+** engineers
- **~100** analysts

Criteo: the technology scale

- **50 Billion** total events logged per day
- **3 Billion** banners served per day
- **1 Billion** unique users per month
- **15 Million** predictions *per second*
- **7** Datacenters
- **10,000+** Servers

Criteo: the technology scale part deux

Thursday June 18th 2015

Criteo Releases Industry's Largest-Ever Dataset for Machine Learning to Academic Community

<http://www.criteo.com/news/press-releases/2015/06/criteo-releases-industrys-largest-ever-dataset/>

<http://labs.criteo.com/downloads/download-terabyte-click-logs/>

<https://www.kaggle.com/c/criteo-display-ad-challenge>

Scale of the Analytics Stack at Criteo



25+ TB ingested / day



1000+ jobs / day

5+ PB

**Under
Management**



**100+ Analysts
200+ Engineers**



**1000+
Sales and Ops**

A fairly simple stack

Tableau and ROLAP Cube
for Structured Data Access



Hive and Vertica for
Data Warehousing



Cascading, Scalding and
Hive for Data Transformation



Hadoop for Primary Storage
and MapReduce



Let's talk about the bottom end of the stack.

Hadoop, what is it good for?

It's long memory (duh)

Unlimited (wink, wink) data processing capacity

Stuff we can't (yet) do in Vertica*

*yes, we suck—we have not yet finished our Impala vs Vertica benchmarks (more on this later)

Hadoop, how big is yours?

Amsterdam Cluster:

- CDH4.6
- 1100 nodes — no more space
- 14,000 physical cores
- 105TB of Memory
- 37PB of raw storage



Copyright © 2014 Criteo

Paris Next-Gen Cluster:

- CDH5.4
- 600 nodes — in room sized for 5K nodes
- 14,400 physical cores
- 153TB of Memory
- An obscene amount of raw storage



Hive at the core

Hive does two things:

- Provides an incredibly agile data transformation framework
- Allows any idiot to process terabytes of data

The end result:

- Thousands of data transformation jobs created over 2.5 years
- Huge pressure release for building truly large scale analytics RDBMS (ie PB-scale Vertica)

One more thing on Hive

Hive is not sexy

It's just plain SQL and it's easy to dismiss as a tool for noobs

It has a reputation for being slow


But this is changing with Tez and Hive on Spark

It's not type safe

But neither is/was cascading/scalding

Without Hive, we wouldn't have one tenth of the actionable data we have today.



Criteo Contributes Native Parquet Support to Hive

 Hive / HIVE-5783

Native Parquet Support in Hive

Agile Board

Details

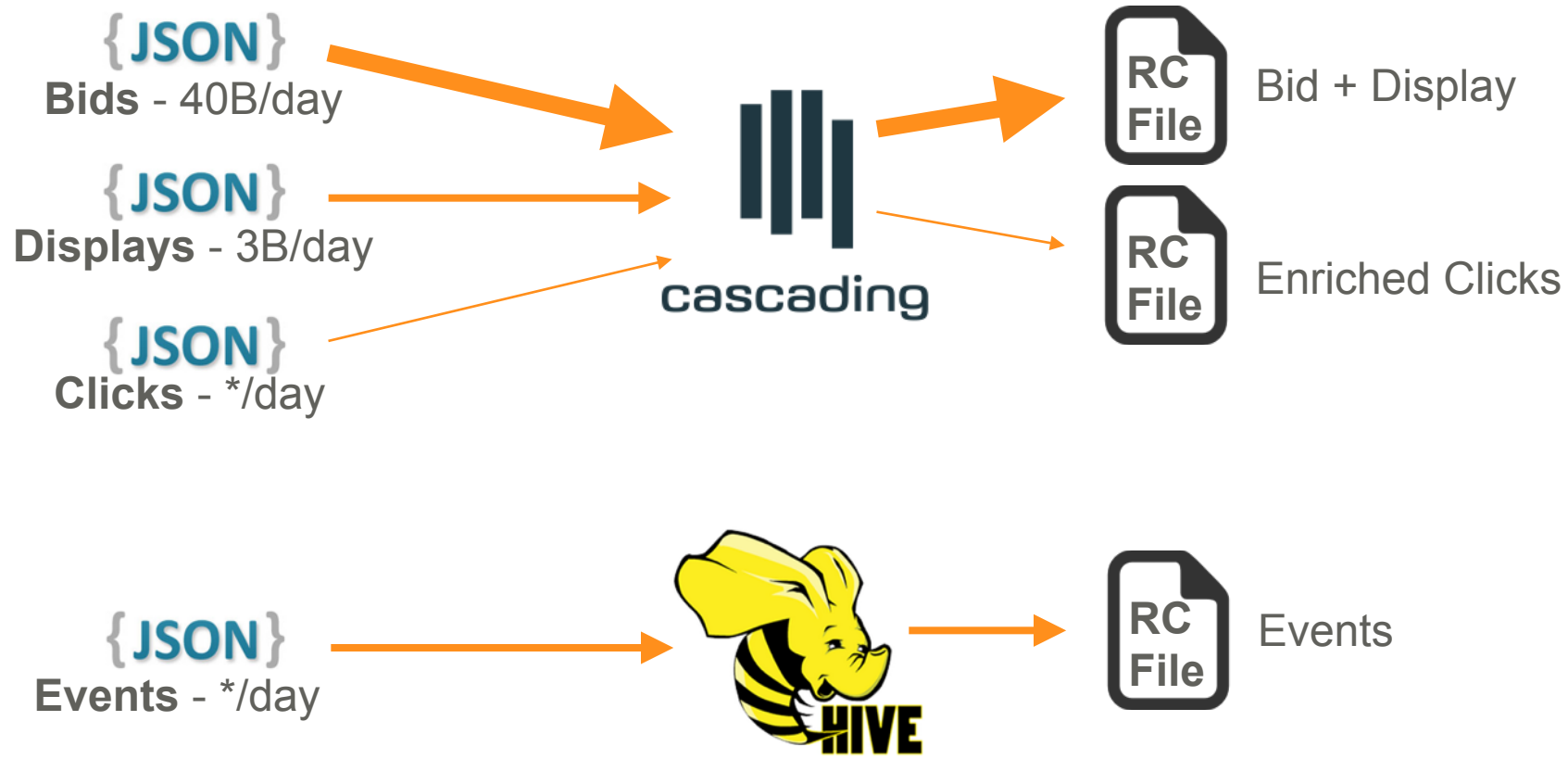
Type:	 New Feature	Status:	RESOLVED
Priority:	 Minor	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	0.13.0
Component/s:	Serializers/Deserializers		
Labels:	Parquet		
Release Note:	Added support for 'STORED AS PARQUET' and for setting parquet as the default storage engine.		

<https://issues.apache.org/jira/browse/HIVE-5783>

<http://blog.cloudera.com/blog/2014/02/native-parquet-support-comes-to-apache-hive/>

Cascading

Cascading is used to generate our primary analytic logs



For those who don't already know, Vertica is a shared-nothing MPP columnar RDBMS (yes, it is buzzword compliant)

- Distribute and order data to optimize for specific query scenarios
- Query times on many billion row tables in seconds is pretty trivial
- Good for OLAP, very bad for OLTP
- Our cluster: 50TB on 50 CPU heavy nodes

Vertica! (our theory)



An Observation: Vertica can do many things, but is best at accelerating ad-hoc queries

A Decision: load the **business critical** subset of our Hive DW into Vertica, and don't allow data to be built or loaded from anywhere else

The Result: with a modicum of tuning, and nearly no day-to-day maintenance, analytic query throughput skyrockets (wrt Hive)

Vertica! (data loading)



We load about 2TB of data per day into Vertica

It arrives mostly in daily batches and takes about an hour in total

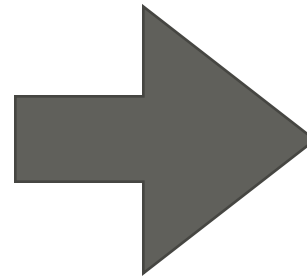
We load via hadoop streaming jobs that use the vertica command line tool (vsq) to bulk insert

Vertica! (performance)



Distinct count across 100B rows:

```
SELECT
  COUNT(DISTINCT user_id_fast) AS users,
  COUNT(0) AS displays
FROM
  datamart.fact_displays
GROUP BY
  DAY
ORDER BY
  DAY
```



Query time 59 seconds:

	users	displays	
1	232 549 419	2 434 833 375	
2	229 432 954	2 365 051 378	
3	220 352 068	2 099 690 272	
4	210 332 574	1 982 314 638	
5	180 524 973	1 662 420 014	
6	186 951 923	1 765 482 137	
7	231 459 191	2 309 580 518	
8	227 055 794	2 171 543 464	
9	222 040 964	2 096 840 105	
10	219 693 745	2 100 773 028	
11	212 387 662	2 020 609 419	
12	180 717 842	1 691 586 327	
13	189 990 551	1 839 440 453	
14	230 384 189	2 315 193 193	
15	225 742 068	2 178 733 760	
16	222 123 617	2 097 221 589	
17	218 928 165	2 083 130 764	
18	208 561 104	1 934 309 270	
45 row(s) fetched - 59087ms			

Remember, Vertica caches nothing...
...not even query plans.

Moving up, and closer to our users.

Now, let's find out a bit more about
what our users need.

Needs Analysis

Hey, analysts, what y'all wanna do with all that data?



Seriously, it was nearly that.

Okay, wait! We do have an early use case!

- Dilemma: local analysts need to run year long analyses for a given advertiser
 - Full dataset in question is about 500TB
 - 500K - 2M mappers needed depending on split size
 - Translates to ~3M minutes of CPU time
 - Run time in days on a reasonable sized cluster

Domain Specific Partitioning for Fast Queries

- Solution: partition in Hive by advertiser ID % 1000
 - Distribution not perfect, but pretty good and easy to implement
 - Workload typically divided by 500 to 1500
 - Run times in minutes to an hour or so for complex queries

ROLAP SSAS Cube on Vertica



An idea: let's unleash Vertica's query power on a ROLAP cube. That should hold off a few needs requests.

After a few stops and starts, it actually works and on cube-like slice and dice workloads it's even quite fast.

The trouble is, our users are using it as an extraction engine and `select * from a GAGILLION rows with no predicate` turns out to be slow.

ROLAP SSAS Cube on Vertica



The real solution appears to be many small domain specific cubes and leave the extract work to scheduled jobs.

We built our existing cube to do everything, because this is what was asked for by our primary stakeholder, FP&A.

btw, we're looking for a PM to help us work better with FP&A

Tableau for analytic dashboarding



Our users tell us they need to display charts and stuff.

Internal testers like Tableau, so we go with that.

Tableau believes in enabling analysts to easily produce complex analytic dashboards.

on their own.

Yeah, it's been a bit rough making *that* work at scale.

btw, we're hiring a Tableau PM.

Tableau: A Snapshot



Site Dashboard

Workbooks

717

Workbooks Views

1,818

Disk Usage (MB)

16,266

Datasources

228

Extracts

437

Subscriptions

959

Distinct Users

889

Looks

85,375

Site Name
All

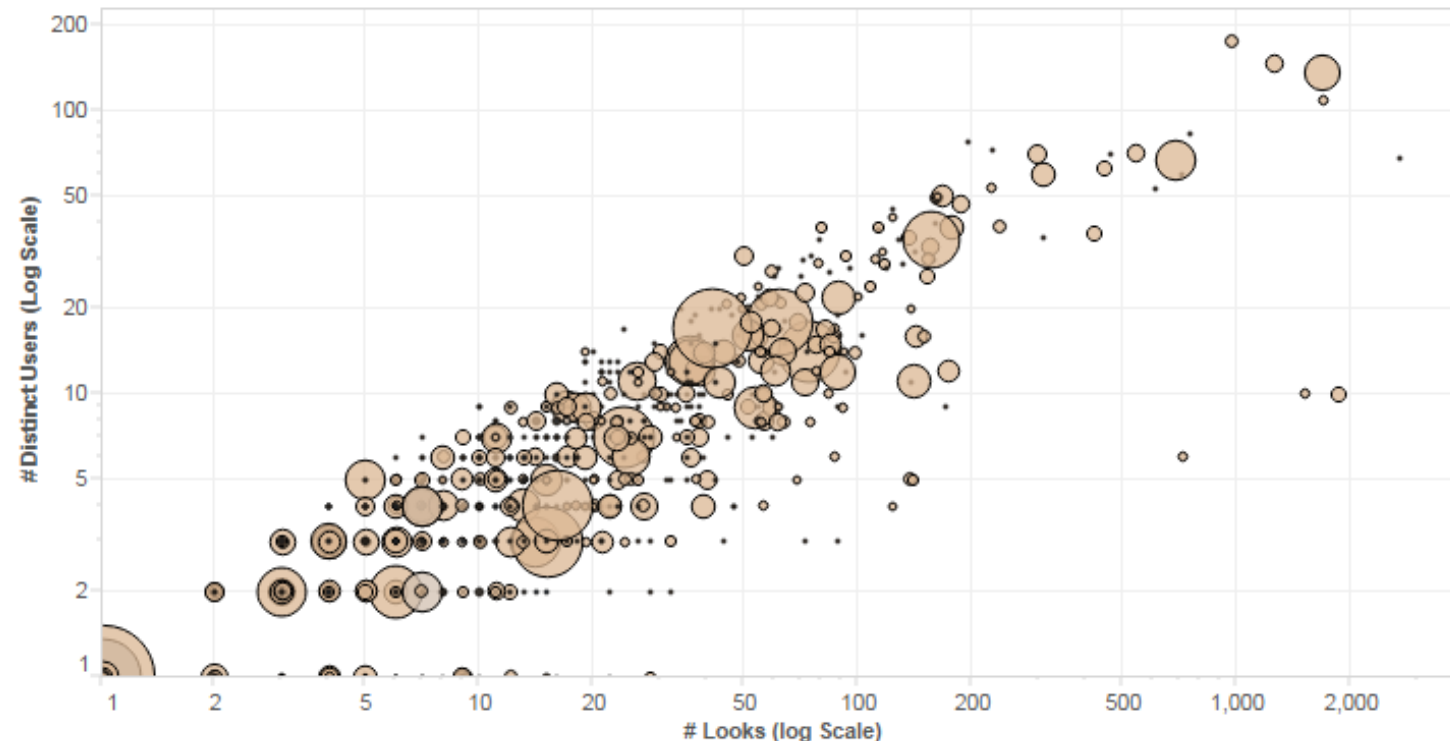
Stat Scope:
Last 30 Days

Select your time zone:
UTC+1 | CET

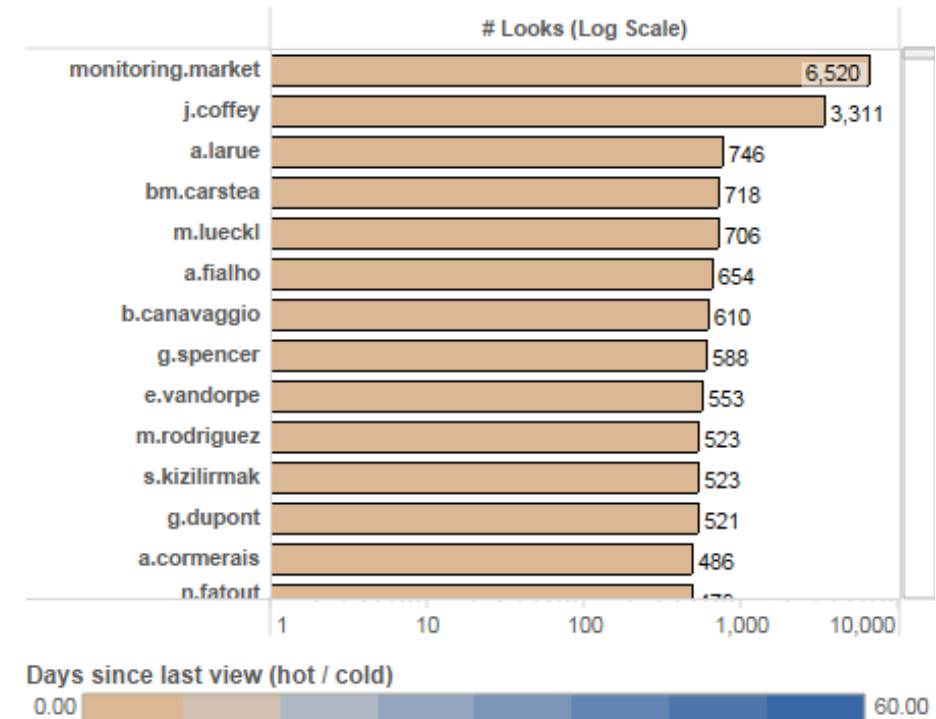


WORKBOOK INFORMATION

Workbook View Popularity *(The size represents loading time in seconds)*



Site Audience



Let's Talk about Data Latency

So, I'm like: btw, how quickly do you need those 500 tables in Vertica?

And they're like: 5 minutes.



And then, I'm like:



Job Scheduling

As I've said, we have thousands of jobs.

Job scheduling is dependency driven (ie a job is only executed when all of its parents have completed successfully)

Average total graph execution time is about 12 hours

Most stuff is done in about half that time

Job Scheduling

Today most workflows use our own tool called “jobr”

It’s a polling scheduler written in ruby with job definitions in ruby template files.

Tomorrow we will fully migrate to “Langoustine”

Also our tool, but written in scala and event-driven with testable workflows.

Job Scheduling: Source Control and Release

Every job is source controlled and deployed via a release process

code sharing

code reviews

easily handle external contributions

rollbacks

git blame

Looking back at what worked and
what didn't

What Works: Simplification

Without question, the most important thing is to simplify

An example: sole sourcing data for Vertica from Hadoop provides an implicit backup. It also allows for easy replication to multiple clusters.

You can't be an expert in everything. Focus is key.

It's easier to train colleagues to contribute to a simple architecture—and external contributions are a Great Thing.

What Works: Scaling Out Instead of Optimizing

Optimizations tend to make systems complex

If your system is already distributed (eg Hadoop, Vertica), scale out (or perhaps up) until that no longer works.

Seriously. It's okay to waste some CPU cycles. Hadoop was practically designed for it.

What Works: Hive!

Get on the bandwagon, people! Hive rocks!

I'll try avoiding repeating myself here, but without Hive I'd need a small army of data engineers.

And probably, the code would be less maintainable than our hive queries.

What Works: Vertica!

It's either that or 100's of *SQL Servers. You choose.

Vertica lets us do things we were otherwise incapable of doing and with very little DBA overhead (we actually don't have a Vertica DBA!)

Our users consistently tell us it's their favorite tool we provide.

There are other similar tools out there now, of course, so do explore your options for the right fit.

What Doesn't Work: Early Scaling

It's easy to start thinking of every problem as distributed

Distributed design is hard and engenders complex architectures that slow down development and make deployment harder.

Even in a distributed-scale environment like Criteo, many systems work just fine on single (mirrored of course) servers.

What Doesn't Work: Implementing without Specifying

Sounds obvious, but get needs reqs from your users!

In the face of little specification one tends to build “generic” systems.

This generally takes longer to deliver.

And tends to under-perform.

What Can Work: Rapid Prototyping

It's easy to screw up—you need to set expectations

Occasionally quick-and-dirty imposes itself as the obvious solution.

This is okay if expectations are properly set, but this is trickier than you might imagine.

Rapid prototyping for internal testing is generally a Good Thing, of course.

Our Roadmap (a portion thereof)

To Address our Data Latency Issues:

- Wide-scale deployment of Langoustine*
- Hive on Spark or Tez
- Impala vs Vertica benchmarking
- Lambda/Near-line for business critical data

To Address Dashboarding Performance Issues:

- Tableau Release Process*
- In-house visualization framework*

*all items that will be open sourced in the coming weeks/months

Thank You!

Questions?

And, yes, we're hiring. Come see me after the talk.