

# SPARK THE HARD WAY:

Lessons from Building an On-Premise Analytics Pipeline

Damian Miraglia  
Joseph de Castelnau  
Nielsen MROI Solutions



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE  
JUNE 6-8, 2016 SAN FRANCISCO

# Nielsen MROI Solutions

## A Brief History

1991

- Marketing Analytics  
Founded by Ross Link

2011

- Marketing Analytics  
Acquired by Nielsen

2015

- Digital Media Consortium II
- Multi-Touch Attribution



SPARK SUMMIT 2016

# Digital Media Consortium II

## Industry Collaboration



### Objectives:

- Test and improve industry practices for the measurement of digital media
- Understand the best way to use newly available, granular data
- Measure and optimize return on the billions of dollars invested in marketing



Advertising  
Campaigns



Worth of Sales  
Analyzed

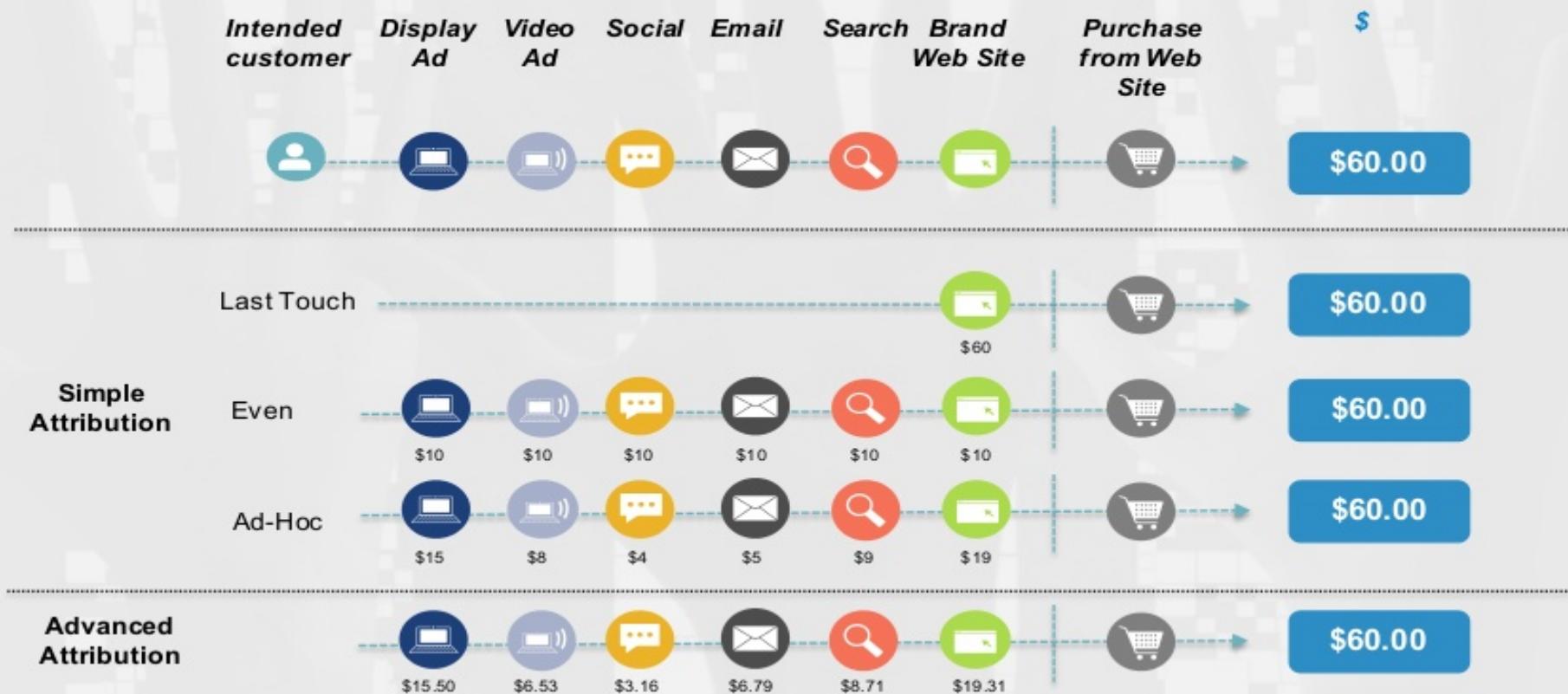


Digital Impressions  
Measured



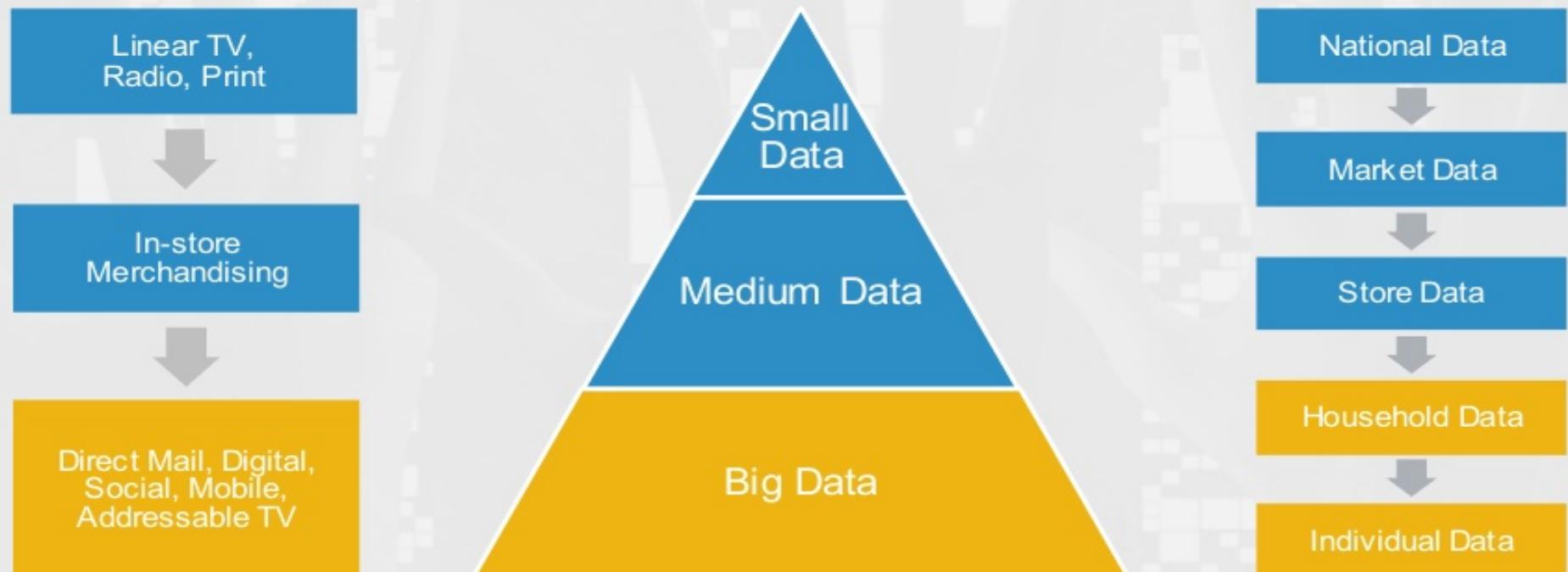
# Multi-Touch Attribution

## Attributing Sales Across Media Tactics



# MTA Data Volume

## Big Data Requires New Tools



# Algorithm Development Process

Path from Prototype to Product

 Research Statisticians

Prototype

Prototype in SAS/Netezza and test  
against real data



Data Engineers

Scale

Adapt the model for Spark and  
optimize performance



Software Engineers

Report

Build visualizations and navigation to  
explore insights from the model



SPARK SUMMIT 2016

# Linear Path Does Not Work

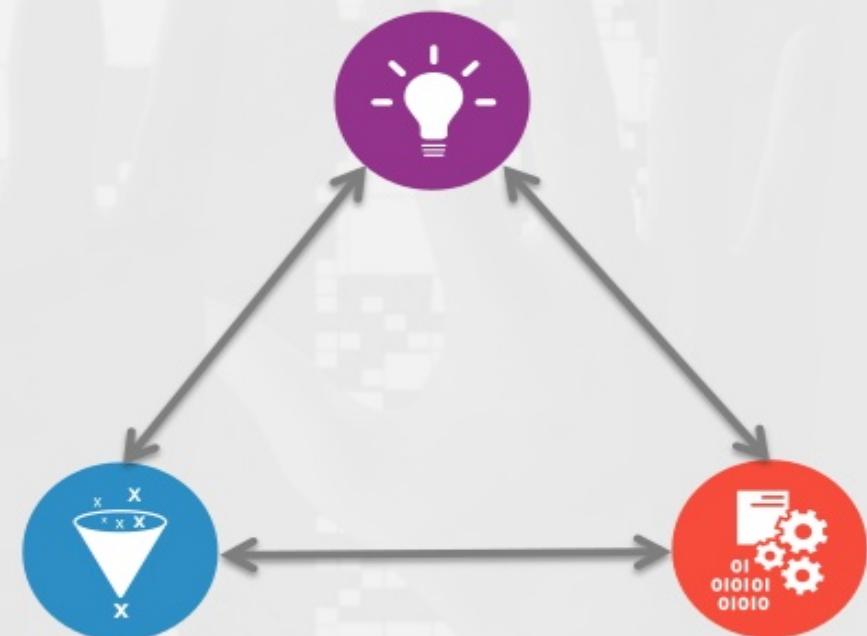
What We Learned the Hard Way

“Lift and Shift”  
ignores platform  
differences

Engineers and  
Statisticians  
should pair  
program

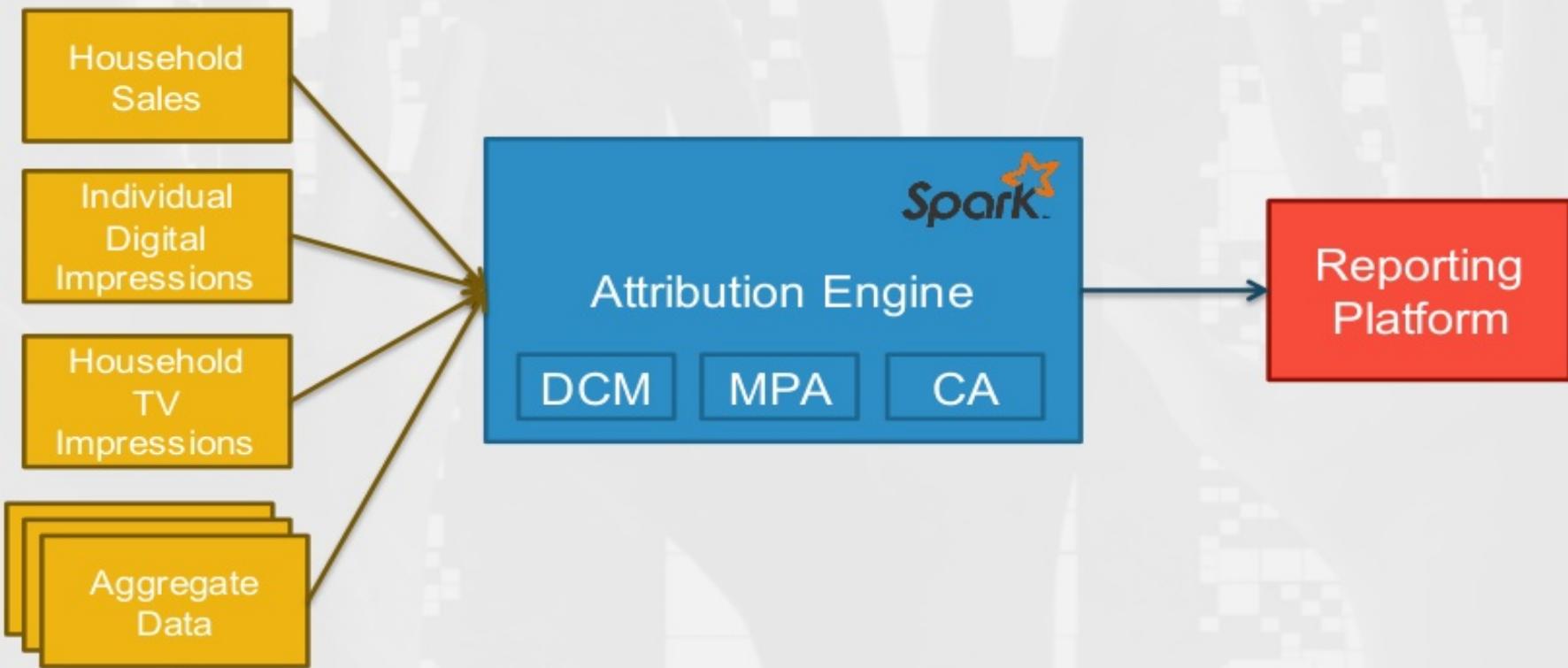
Collaboration  
across functions  
is key

Models must be  
built on Spark  
from the start



# MTA Data Flow is Straightforward

A Series of Tubes



# MTA Data Issues

Digital Data is Messy



## Missing Impressions

- Many impressions for each event are not collected by the DMP



## Mismatched Impressions

- Cookies are often onboarded incorrectly



## DMP Decertification

- Publishers disabled ability to tag ads across a wide range of properties



## Off-Target Data

- Data is sometimes comingled or over/under filtered

# Data Sensitivity and Restrictions

In agile fashion: MVP on-prem, V1.x on cloud



**While we are resolving restrictions & limitations, we started with on-prem solution:**

- Retailers reluctant to allow sales data on sensitive cloud system
- Privacy concerns require Nielsen to operate with all governance set to green.

</INTRO>



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE  
JUNE 6-8, 2016 SAN FRANCISCO

**So we're stuck on premise.  
Now what?**



SPARK SUMMIT 2016

# Spark on the Cloud would have been Great

- On Demand
- Elastic Scaling
- Infrastructure as a Service
- Data Science Toolchain Built-In
- Managed Services
- Isolated Environments
- Vendor Support
- Job Scheduling



# Our Cluster

There are Many Like it, but this One is Ours

## Red Hat Enterprise Linux

- Version 7.2
- Compliance through SELinux

## Data/Compute Nodes

- Hortonworks Data Platform
  - Provides Zookeeper, HDFS, Yarn, Hive, Spark\*
  - Cluster Management through Ambari

## Edge Node

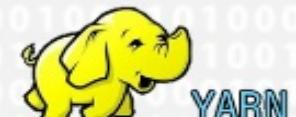
- Adjacent to, but not part of, the cluster
- Used to run drivers, submit jobs

## Chef

- Ensures Consistency across Nodes
- Eases DevOps Burden



SPARK SUMMIT 2016



\*The version of Spark included in HDP typically lags by several months

# Problem: Environment Sharing

There Can Be Only One Cluster



R&D, QA, Staging, and Production all live on the same cluster



Different models have different dependencies and might even run on different versions of Spark

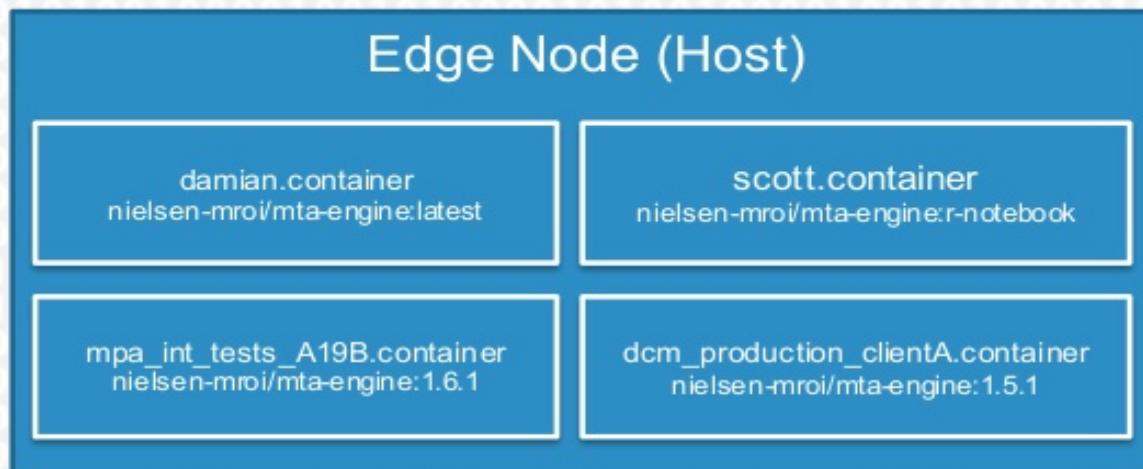


Changing Spark configuration settings may adversely impact other developers



Only one branch of code can be deployed at a time

# Docker Creates Virtual Edge Nodes



docker

Repository	Tag	Image ID	Created	Size
nielsen-mroi/mta-engine	latest	48218b16fe5b	4 weeks ago	3.168 GB
nielsen-mroi/mta-engine	1.6.1	48218b16fe5b	4 weeks ago	3.168 GB
nielsen-mroi/mta-engine	r-notebook	44776f55294a	4 weeks ago	4.03 GB
nielsen-mroi/mta-engine	scala-notebook	17be1ee7089f	4 weeks ago	3.247 GB
nielsen-mroi/mta-engine	1.6.0	f249c7e8729a	6 weeks ago	3.012 GB
nielsen-mroi/mta-engine	1.5.1	0ab5aaafbd94b	12 weeks ago	2.988 GB
centos	latest	778a53015523	8 weeks ago	196.7 MB
centos	7	778a53015523	8 weeks ago	196.7 MB

# **Everything is Isolated, Consistency Assured**

Models

- Spark Version via `spark.yarn.jar`
  - Dependency Management
  - Configuration Settings (Spark/Yarn/Hive)

## Developers

- Code Branches
  - Preconfigured Toolchains

## Continuous Integration

- Executable Container (via bootstrap.sh)
  - Run tests, Export results
  - Triggered by Jenkins on Push

```
FROM centos:7
MAINTAINER nielsen-mrei

# update yum, install wget
RUN yum update -y \
...
# install java
RUN wget --no-cookies --no-check-certificate --header \
...
# install scala
RUN wget http://downloads.typesafe.com/scala/2.10.6/scala-2.10.6.tgz \
...
# install hadoop
RUN wget http://apache.claz.org/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz \
...
# set environment variables needed for spark install
ENV JAVA_HOME=/opt/java \
...
# install spark
RUN wget http://apache.claz.org/spark/spark-1.6.1/spark-1.6.1.tgz \
...
# install os packages
RUN yum -y install \
...
# install python libraries
RUN pip install --upgrade pip \
...
# copy resources, and move them to their destinations
COPY files /
RUN mkdir /opt/spark_dependencies \
...
# set environment variables
COPY env_vars.sh /etc/profile.d/
ENV SPARK_HOME=/opt/spark \
...
# copy bootstrap script and set container to execute it
COPY bootstrap.sh /
RUN chmod root /bootstrap.sh && chmod 700 /bootstrap.sh
ENTRYPOINT ["/bootstrap.sh"]
```



# Problem: Orchestration

Step 1: Collect Data, Step 2: ???, Step 3: Profit!



Triggering long-running, multi-part jobs is risky without resumability.



Models consist of too many steps and have DAGs that are too big for Spark to handle all at once



Models developed by teams of engineers need convenient contracts between their component parts

# Luigi Builds Composable Task Chains



```
import luigi

class MyTask(luigi.Task):
    param = luigi.Parameter(default=42)

    def requires(self):
        return SomeOtherTask(self.param)

    def run(self):
        f = self.output().open('w')
        print >>f, "hello, world"
        f.close()

    def output(self):
        return luigi.LocalTarget('/tmp/foo/bar-%s.txt' % self.param)

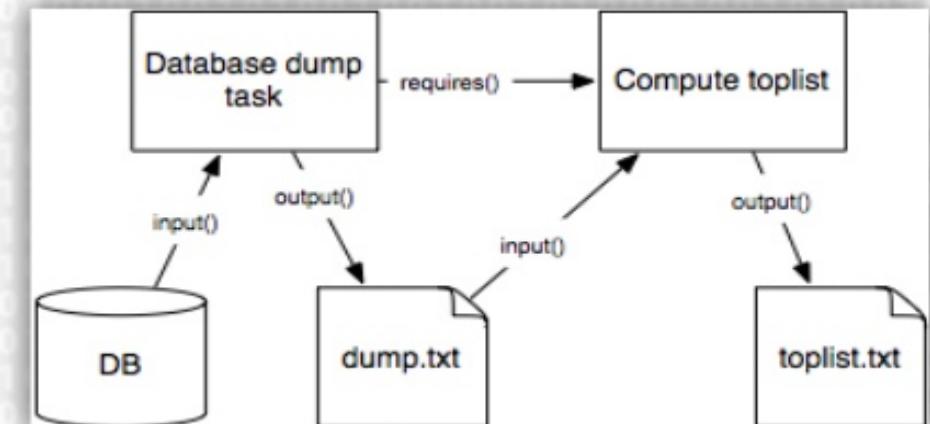
if __name__ == '__main__':
    luigi.run()
```

The business logic of the task

Where it writes output

What other tasks it depends on

Parameters for this task



SPARK SUMMIT 2016

Source: <https://luigi.readthedocs.io/>

# Luigi Provides Reliable Dependency Management

## Dependency Resolution

- Dependency tree
- Tasks run in parallel

## Resumability

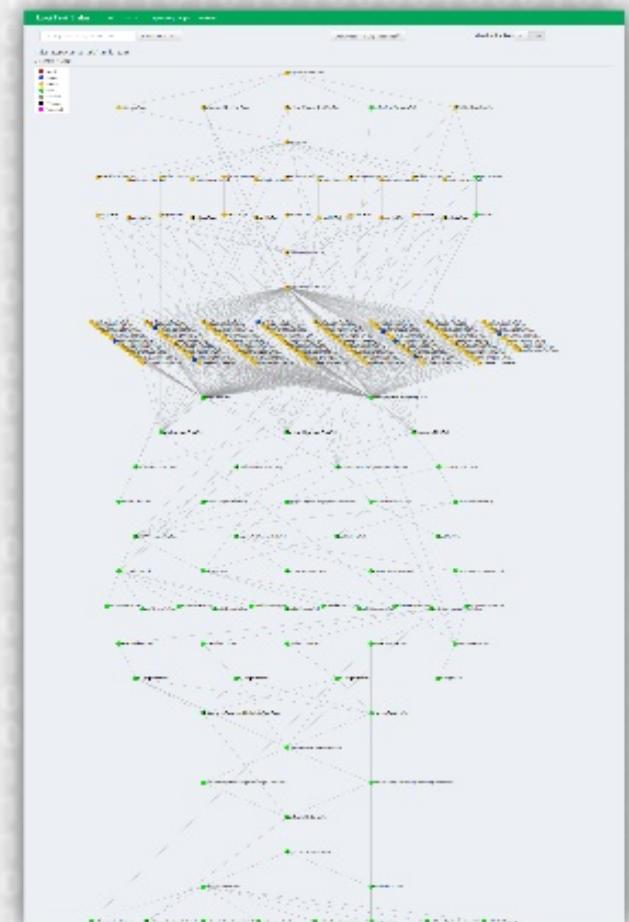
- Target discovery
- Won't rerun completed tasks

## Hackable

- Python code
- Small codebase

## HiveTableTarget(`luigi.Target`)

- Encourages checkpointing
- Solid contract between tasks
- Developer Parallelization



SPARK SUMMIT 2016

# Problem: Performance Concerns

This Cluster ain't Big Enough for the Two of Us



There's only so much RAM, and only so many CPUs in the cluster.

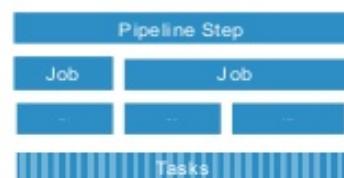


Some jobs are resource hogs and require 50% or more of the cluster resources.



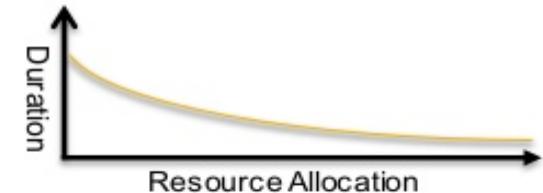
Some resources must be reserved for R&D purposes (see problem 0).

# Meta-Analysis Ensures Optimal Resource Utilization



- Spark breaks a given Step into smaller pieces (**Step -> Jobs -> Stages -> Tasks**)
- Spark efficiently parallelizes these smaller pieces

- Spark's built-in parallelism improves performance as more resources are used
- Performance does not scale sub-linearly forever – **returns diminish**



**Solution:** Experiment with Different Resource Allocations for each Task and Optimize for Throughput

- Steps may take longer than if given all of the resources
- Steps that do not depend on each other can be run in parallel
- **Pipeline runs in less time overall**

# Configuration Alleviates Resource Contention

## YARN Dynamic Resource Pools

- Scheduled allocation of resources
- Unused resources can be reallocated

## Spark Dynamic Allocation

- Define min & max number of executors
- Unused executors are removed and resources returned to pool

## Miscellaneous Spark Configs

- spark.speculation
- spark.shuffle.io.numConnectionsPerPeer
- spark.akka.threads



SPARK SUMMIT 2016

# Problem: Debugging

I Hope You Like Reading Logs



Debugging Spark problems often requires digging through YARN logs.



Particularly heinous failures will sometimes prevent YARN's log aggregation from collecting everything.



SPARK SUMMIT 2016

# Log Management Makes it Easier to Gain Insight

The screenshot shows the Graylog web interface. At the top, there's a search bar and navigation links for Streams, Dashboards, Sources, Systems, Help, and Administrator. Below the search bar, a pie chart titled "Quick Values for facility" shows the distribution of messages by facility: gelf-java (yellow), log-interface (pink), and intra-app (green). A table below the chart lists these values with counts: gelf-java at 81.27% (3,722), log-interface at 18.11% (651), and intra-app at 0.02% (1). The main area displays a histogram of log messages over time, with a prominent peak around 08:02. At the bottom, a table titled "Messages" shows two log entries from "damian.container". Both entries are identical: "2016-03-01 08:02:18,910 [damian.container] [log-interface] [sun@damian] Data-patches[log4j] 2" followed by a stack trace.

# graylog

```
# Set everything to be logged to the console
log4j.rootCategory=ERROR, console
log4j.appenders.console=org.apache.log4j.ConsoleAppender
log4j.appenders.console.target=System.err
log4j.appenders.console.layout=org.apache.log4j.PatternLayout
log4j.appenders.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Set log level for third party logs that are too verbose
log4j.logger.org.eclipse.jetty=WARN
log4j.logger.org.eclipse.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$ExprType=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO

# Send all INFO logs to graylog2
log4j.rootLogger=INFO, graylog2

# Define the graylog2 destination
log4j.appenders.graylog2=org.graylog2.log.GelfAppender
log4j.appenders.graylog2.graylogHost=dayrhemtad005
log4j.appenders.graylog2.facility=gelf-java
log4j.appenders.graylog2.layout=org.apache.log4j.PatternLayout
log4j.appenders.graylog2.extractStackTrace=true
log4j.appenders.graylog2.addExtendedInformation=true
log4j.appenders.graylog2.originHost=damian.container
log4j.appenders.graylog2.additionalFields={'environment': 'DEV', 'application': 'Spark'}
log4j.appenders.graylog2.originHost=damian.container
log4j.appenders.graylog2.additionalFields={'environment': 'DEV', 'application': 'Spark'}
```



SPARK SUMMIT 2016

# Graylog is Easy to Wire Up with Spark

## Log4J Appender

- Easy setup to forward Spark logs
- Works even if YARN log aggregation fails

## Containerized Installation

- Can be run from Docker containers
- Low setup cost

## Powerful

- Search across many fields
- Dashboards for admins
- Alerts for failures

## Selected sources

Name	Percentage	Message count
Top sources		
subhash.container	78.01%	19,603,676
jenkins.container	9.76%	2,453,687
eric.container	3.67%	922,949
damian.container	2.74%	689,423
sean.container	2.00%	502,525
yue.container	1.78%	446,961
a-a-ron.container	0.64%	161,107
jing.container	0.46%	115,831
metadataservice	0.28%	70,155
mta rest api	0.25%	63,786

# Problem: Development Tooling



Developing with Spark-Submit alone is inefficient



Not all users have the requisite skills to handle the entire development toolchain



SPARK SUMMIT 2016

# Roll Your Own Tooling

The screenshot shows a Jupyter notebook interface with a Python 2 kernel. Cell 5 contains the following code:

```
from pyspark import SparkConf, SparkContext, HiveContext
from pyspark.sql import DataFrame
from pyspark.sql.functions import sum as sqlsum, coalesce, lit

from IPython.display import display
from IPython.display import HTML

def prettify(dataframe, count=10):
    col_to_th = lambda c: "<th>{0}</th>".format(c)
    el_to_td = lambda el: "<td>{0}</td>".format(str(el))
    row_to_tr = lambda row: "<tr>{0}</tr>".format("".join(map(el_to_td, row)))

    table = "<table><thead>{0}</thead><tbody>{1}</tbody></table>"
    head = "<tr>{0}</tr>".format("".join(map(col_to_th, dataframe.columns)))
    body = "".join(map(row_to_tr, dataframe.take(count)))
    html = table.format(head, body)
    display(HTML(html))

DataFrame.show = prettify

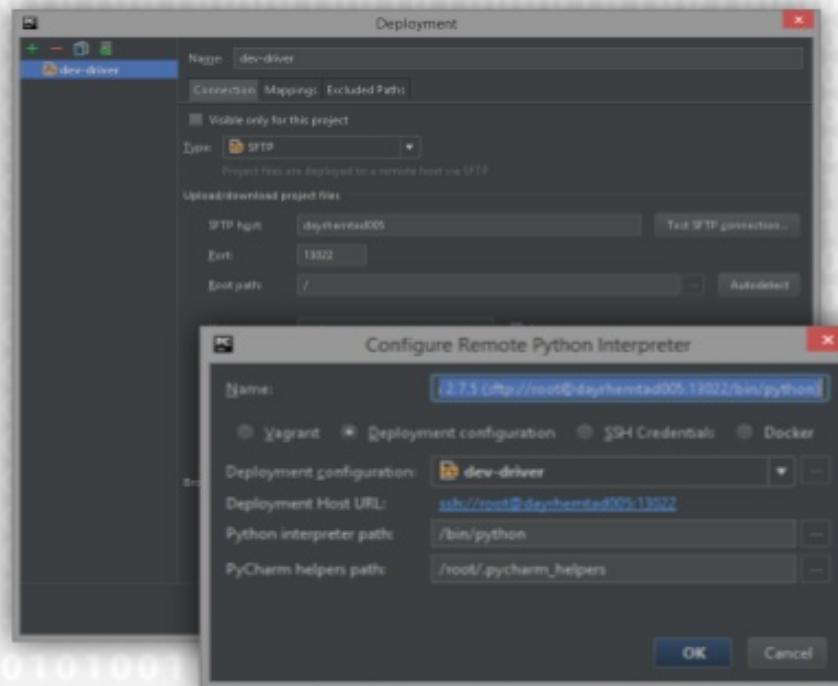
conf = SparkConf()
conf.set("spark.executor.instances", "4")
conf.set("spark.executor.memory", "20g")
conf.set("spark.executor.cores", "25")
conf.set("spark.yarn.executor.memoryOverhead", "8000")
conf.set("spark.shuffle.io.numConnectionsPerPeer", "5")
conf.set("spark.shuffle.io.retryWait", "500ms")
conf.set("spark.reducer.maxSizeInFlight", "128m")
conf.set("spark.speculation", "true")
conf.set("spark.default.parallelism", "500")
```

The screenshot shows the MTA Management interface with the 'Run ProcessAdviewsTask' page open. The page includes the following information:

- Description:** The ProcessAdviewsTask is responsible for combining data from the four adviews profile files: OH Local, OH National, SL Local, SL National. The processing includes unioning the local exposures together, projecting national exposures down to the DMA level, and some mapping/messaging of the exposure dimensions.
- Heads Up!** Looks like ProcessAdviewsTask may have been run with these parameters already! The following table(s) already exist: adv\_table. [Move That Table](#)
- Model:** Damian Test Model
- Task Dependencies:** A table showing dependencies for the LoadAdviewsFileTask:

Task Name	Output	Drop Table
LoadAdviewsFileTask	ADV_GM_NATIONAL	<input type="checkbox"/>
file_name: ADV_GM_NATIONAL.txt		<input type="checkbox"/>

# Get Creative with Your Tools



SparkMPA - [C:\hg\SparkMPA] - ...\\engine\\read\\tasks\\process\_purchase\_data\_task.py - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

SparkMPA engine read tasks process\_purchase\_data\_task.py process\_purchase\_data\_task.py

```
42         self._generate_date_range_task = generate_date_range_task(self, max_id)
43
44     def requires(self):
45         return [
46             self._load_trip_data_from_netezza_task,
47             self._load_purchase_data_from_netezza_task,
48             self._generate_date_range_task
49         ]
50
51     def main(self, sc, hc):
52         self_id = ProcessPurchaseDataTask.cur_id+2070
53         sc = <sparkContext>.SparkContext object at 0x40529d0>
54         max_purchase_date = datetime.strptime(self.netsa_config["max_purchase_date"], "%Y-%m-%d").date()
55         max_purchase_date = 2015-10-31
56         num_model_weeks = int(self.netsa_config["num_model_weeks"])
57         num_model_weeks = 26
58
59         purchase_detail = hc.table(self._load_purchase_data_from_netezza_task.table_name).purchase_detail: DataFrame[AHHLDID,NBR:bigint,POD_ID:bigint,date_range: date]
60         trip_detail = hc.table(self._load_trip_data_from_netezza_task.table_name).trip_detail: DataFrame[AHHLDD,NBR:bigint,POD_ID:bigint,date_range: date]
61
62         purchase_detail = purchase_detail.within_max_detail(max_purchase_date).as(purchase_date, max_model_weeks,
63             date_range)
64
65         purchase_detail.write.saveAsTable(self.table_name, mode="overwrite")
66
67     def output(self):
68         return HiveTableTarget(self.table_name, self.database_name)
```



SPARK SUMMIT 2016

# CI for Spark Development

All	Create Release Packages	Current Jobs	Release-to-Prod (NMA IT)	Services	Spark-MPA	Web	+ <small>more</small>	<small>add description</small>
S	Test Result	Name	Build Parameters	Build Parameters	Last Duration	Last Success	Last Stable	
1	<span>Green</span> N/A	Test Mount Point			1.4 sec	37 min - #4347	37 min - #4347	  
2	<span>Green</span> 0 of 119 failed (±0)	MPA Lib Tests	Branch=develop	TriggeredBy=Damian	12 min	59 min - #215 - feature/post_exposure_matching - Sean	59 min - #215 - feature/post_exposure_matching - Sean	  
3	<span>Green</span> 0 of 46 failed (±0)	MPA Task Tests	Branch=develop	TriggeredBy=Damian	2 min 25 sec	56 min - #215 - feature/post_exposure_matching - Sean	56 min - #215 - feature/post_exposure_matching - Sean	  
4	<span>Green</span> 0 of 7 failed (±0)	MPA Integration Tests	Branch=develop	TriggeredBy=Damian	26 min	30 min - #215 - feature/post_exposure_matching - Sean	30 min - #215 - feature/post_exposure_matching - Sean	  
5	<span>Green</span> 0 of 1 failed (±0)	MPA E2E Tests	Branch=develop	TriggeredBy=Damian	1hr 53 min	12 hr 31 min - #215 - feature/post_exposure_matching - Sean	12 hr 31 min - #215 - feature/post_exposure_matching - Sean	  
6	<span>Green</span> 0 of 1 failed (±0)	MPA Sanity Tests	Branch=develop	TriggeredBy=Damian	31 sec	1 hr 1 min - #215 - feature/post_exposure_matching - Sean	1 hr 1 min - #215 - feature/post_exposure_matching - Sean	  

Icon:   

Legend:  RSS for all  RSS for failures  RSS for just latest builds



# Jenkins



# docker



SPARK SUMMIT 2016

# ORGANIZATIONAL SOLUTIONS



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE  
JUNE 6-8, 2016 SAN FRANCISCO

# Risk Mitigation for the Cloud



## Multi-Cloud

- Keep sensitive data with preferred cloud vendor
- Prevents “all-in” bets
- Pushes workloads toward most innovative technologies
- <> Cloud agnostic



## Encryption

- Data classification first step.
- Encrypting data at rest and in transit is solvable .
- Can be opportunity to couple to an advanced hash attribute matching algo.
- Vendor space can shorten your cycle time based on your IP.



# Next Steps for our Platform



- Cloud (!)
- Much easier end to end analytics development model
- Significantly simpler data wrangling
- Enable the right kind of “Citizen Data Scientist” insights.
- Syndication/Automation when warranted



# THANK YOU.

We are hiring talent.

Email: [damian.miraglia@nielsen.com](mailto:damian.miraglia@nielsen.com)



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE  
JUNE 6-8, 2016 SAN FRANCISCO