Spark and Hadoop Perfect Together

Arun Murthy
Hortonworks Co-Founder
@acmurthy

Data Operating System

Enable all data and applications

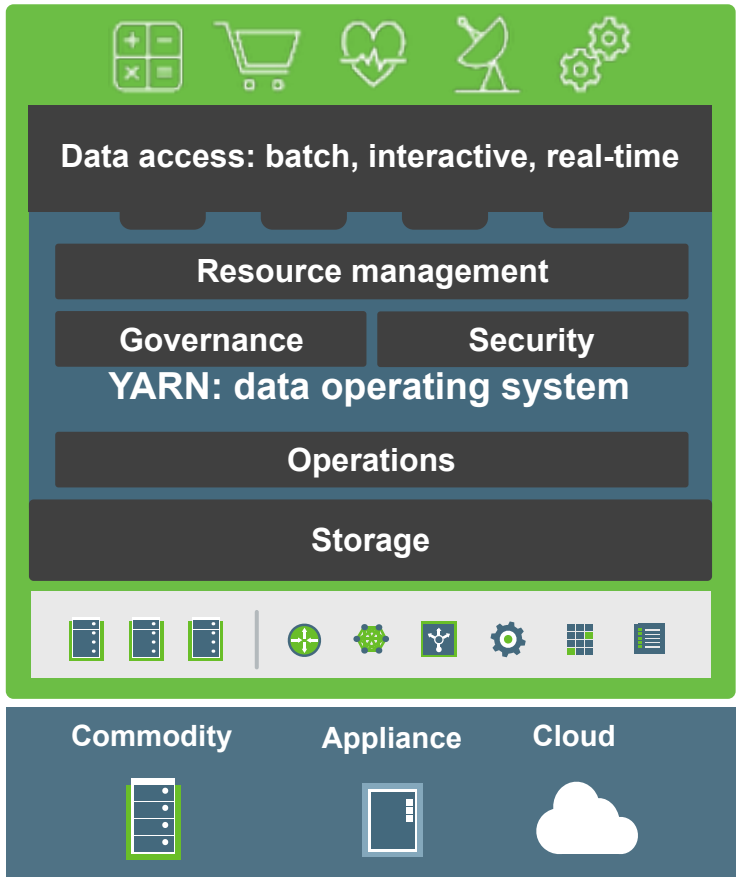
TO BE

accessible and shared

BY

any end-users

Data Operating System



Hadoop/YARN-powered data operating system

100% open source, multi-tenant data platform for any application, any data set, anywhere.

Built on a centralized architecture of shared enterprise services:

Scalable tiered storage

Resource and workload management

Trusted data governance & metadata management

Consistent operations

Comprehensive security

Developer APIs and tools

Hortonworks

Why We Love Spark at Hortonworks

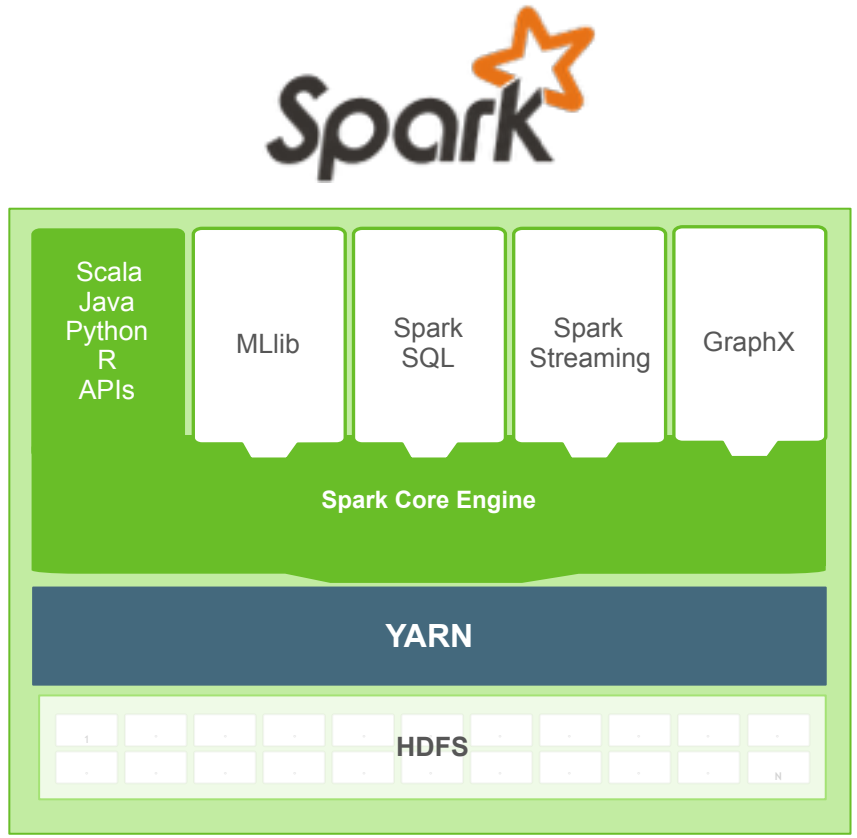
Elegant Developer APIs
DataFrames, Machine Learning, and SQL

Made for Data Science
All apps need to get predictive at scale and fine granularity

Democratize Machine Learning
Spark is doing to ML on Hadoop what Hive did for SQL on Hadoop

Community
Broad developer, customer and partner interest

Realize Value of Data Operating System
A key tool in the Hadoop toolbox



Hortonworks

Let’s Talk Real-World Use-Cases

Hortonworks

USE CASE: webtrends

Spark

HDP

Streaming & Machine Learning for Web Analytics

CHALLENGE	SOLUTION	IMPACT
Cost: Storage and processing not economical at scale Silos: Separate clusters — Hadoop & Spark Analytics: Retroactive view limited predictive capabilities	Spark Streaming for ingesting events in real-time SparkML for predictive analytics	Cost: Hardware costs reduced 25-50% YARN: Shared cluster for Spark, MapReduce, Hive etc. Analytics: Processes 10 billion events daily at 20 milliseconds per event

Hortonworks

Spark

HDP

Claims Re-imbursement Processing with Spark

CHALLENGE	SOLUTION	IMPACT
Overwhelmed by data ingest rates Team expertise in R Lots of key features like textual features not incorporated	Use Spark to optimize claims reimbursements process Leverage Spark's machine learning capabilities to process and analyze all claims.	Insurance companies can now process all claims and detect over and under payment More accurate over and under payment detection

Hortonworks

On-going Customer Use Cases with Spark

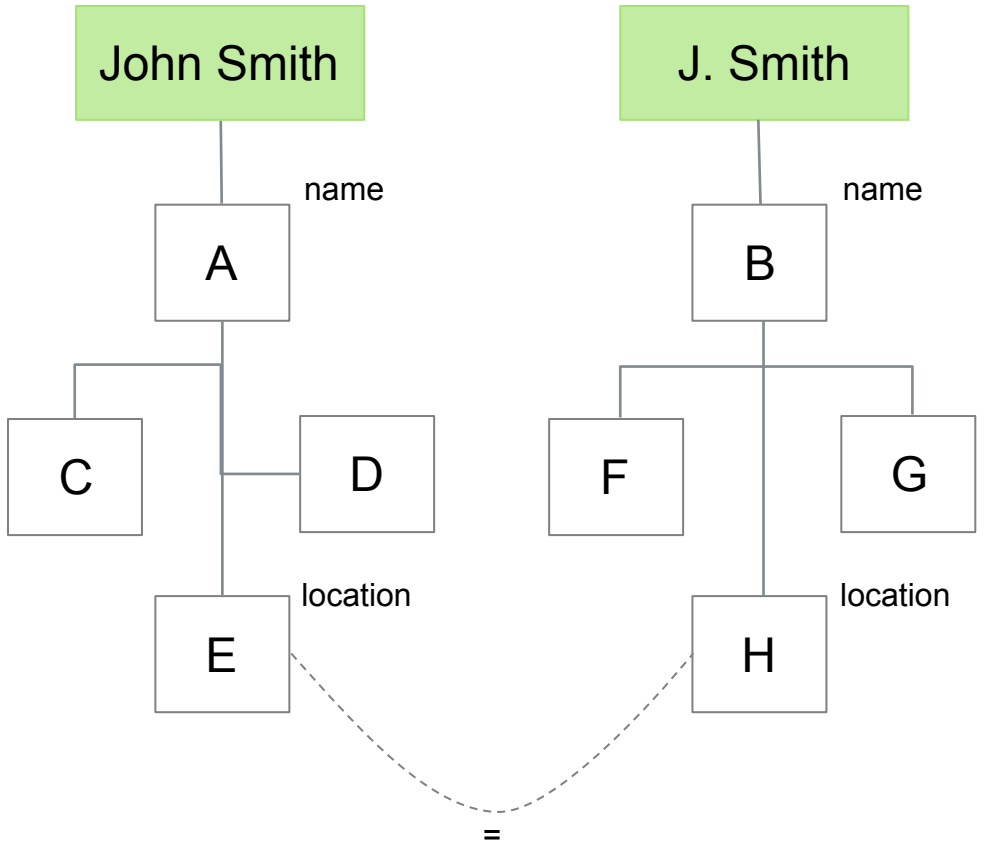
HealthCare

Entity Resolution

- Different feeds represent entities in different forms
  – need to identify same entities
- No standard packages available to do the job.
- Additional features need to be derived to expand context and disambiguate

Requirement

- Extensible entity resolution framework
  – mix supervised and unsupervised learning



Hortonworks

On-going Customer Use Cases with Spark

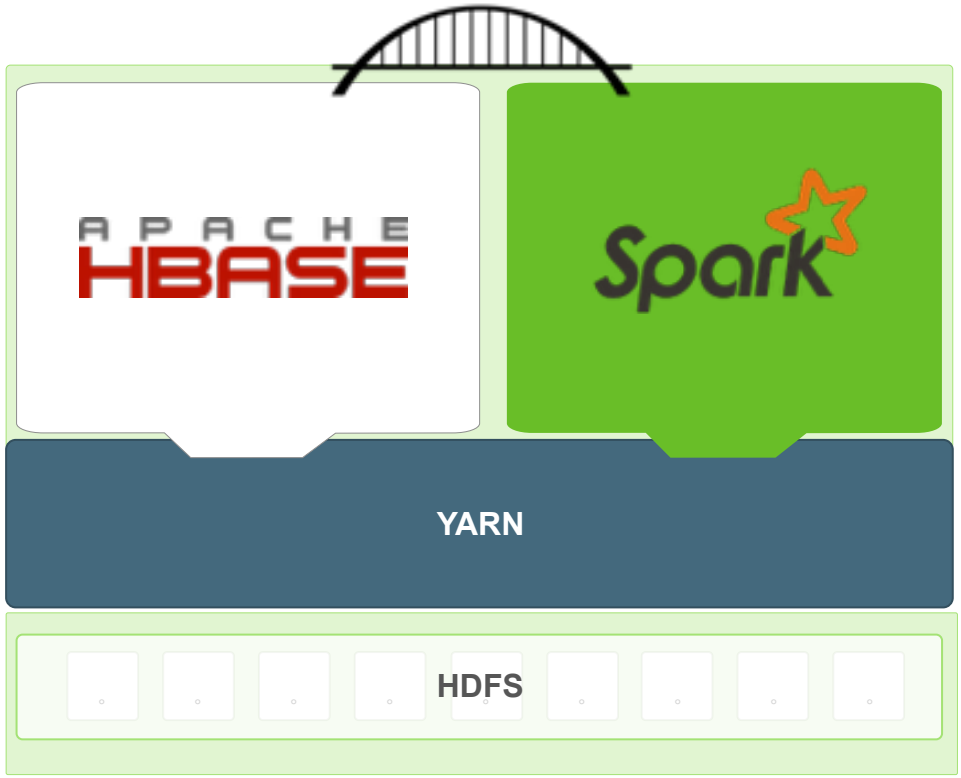
Finance

Efficiently bring HBase Data into Spark

- HBase is the operational store of record
- End user Analytics via Spark

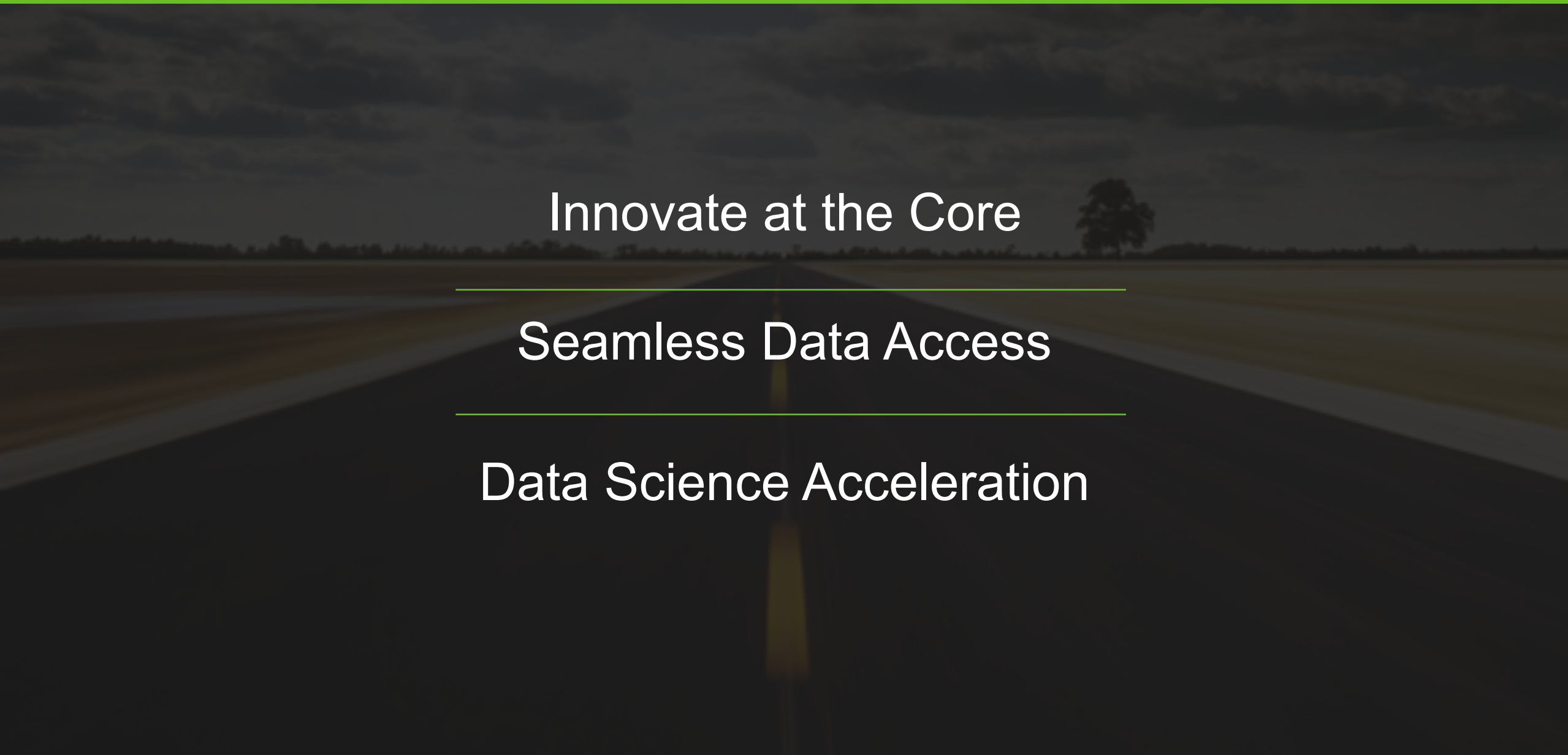
Requirement:

- Efficient scans via predicate pushdown via co-processors



Hortonworks

Spark + Hadoop - The Road Ahead

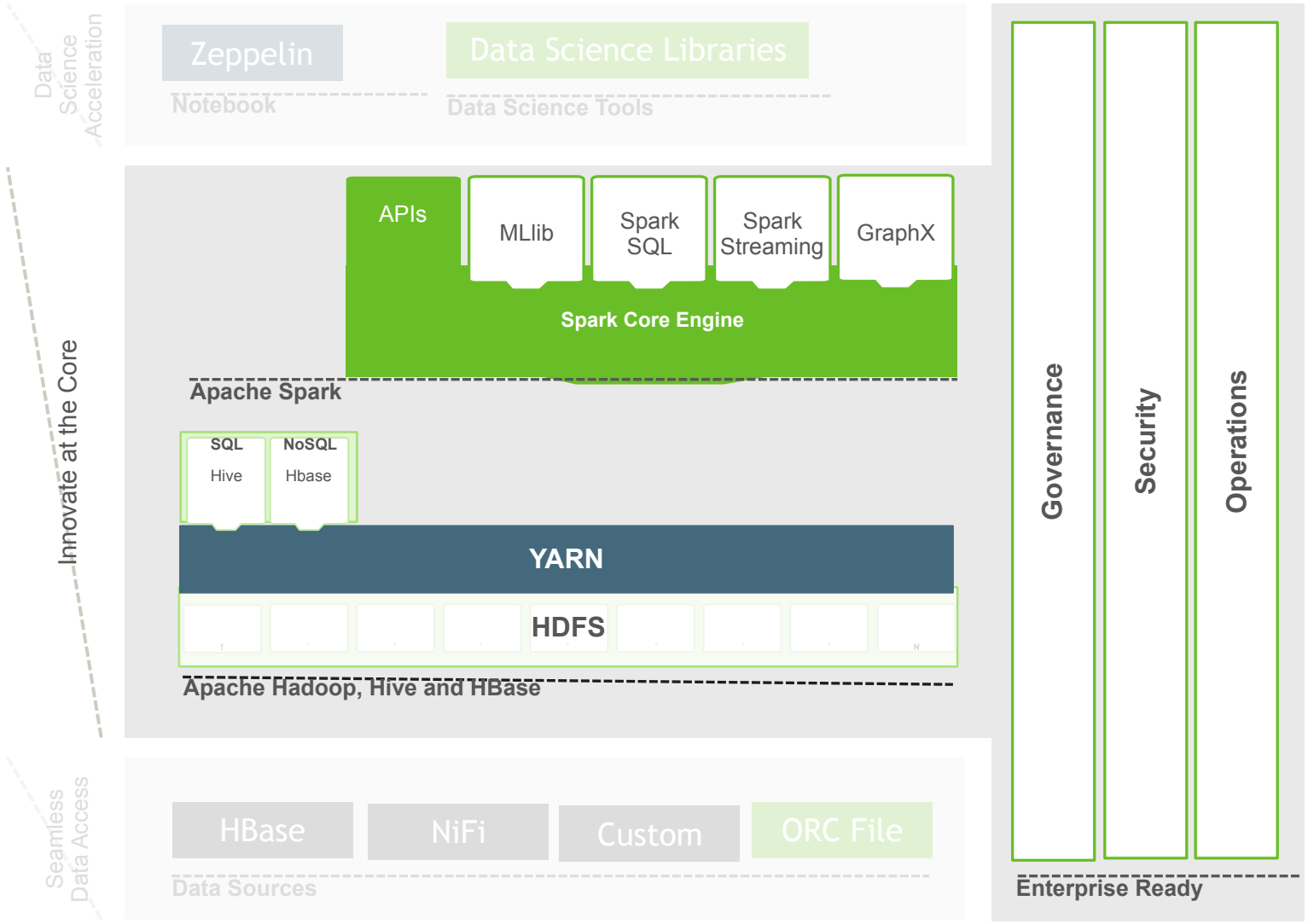


Innovate at the Core

Seamless Data Access

Data Science Acceleration

Spark + Hadoop - The Road Ahead



Innovate at the Core

Storage

- RDD Sharing with HDFS Memory Tier

Resource Management

- Spark on YARN improvements

Security

- Enhance SparkSQL Security
- Enhance Wire Encryption

Governance

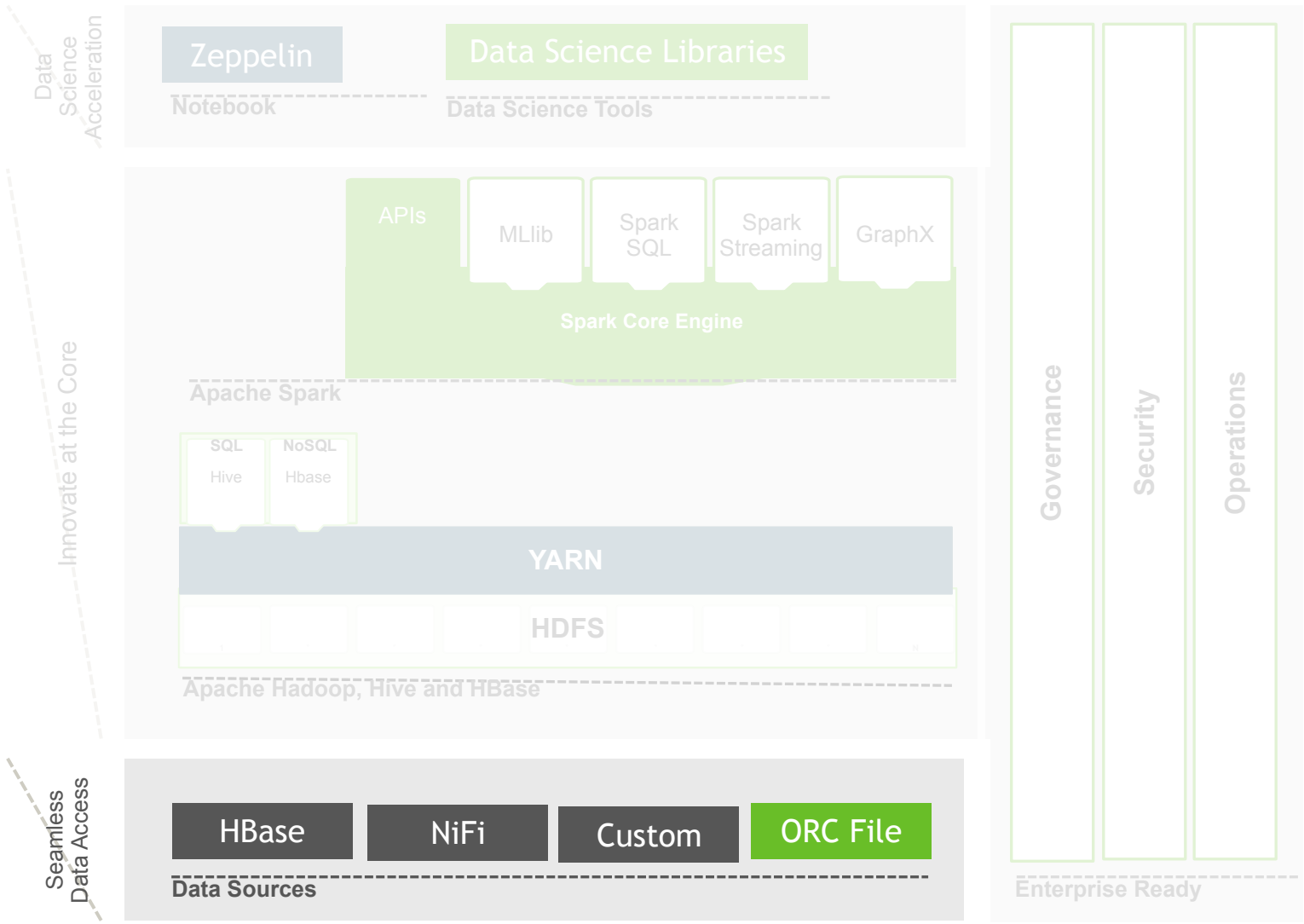
- Integration with Atlas

Operations

- Ambari to Support Multiple Spark Versions

Hortonworks

Spark + Hadoop - The Road Ahead



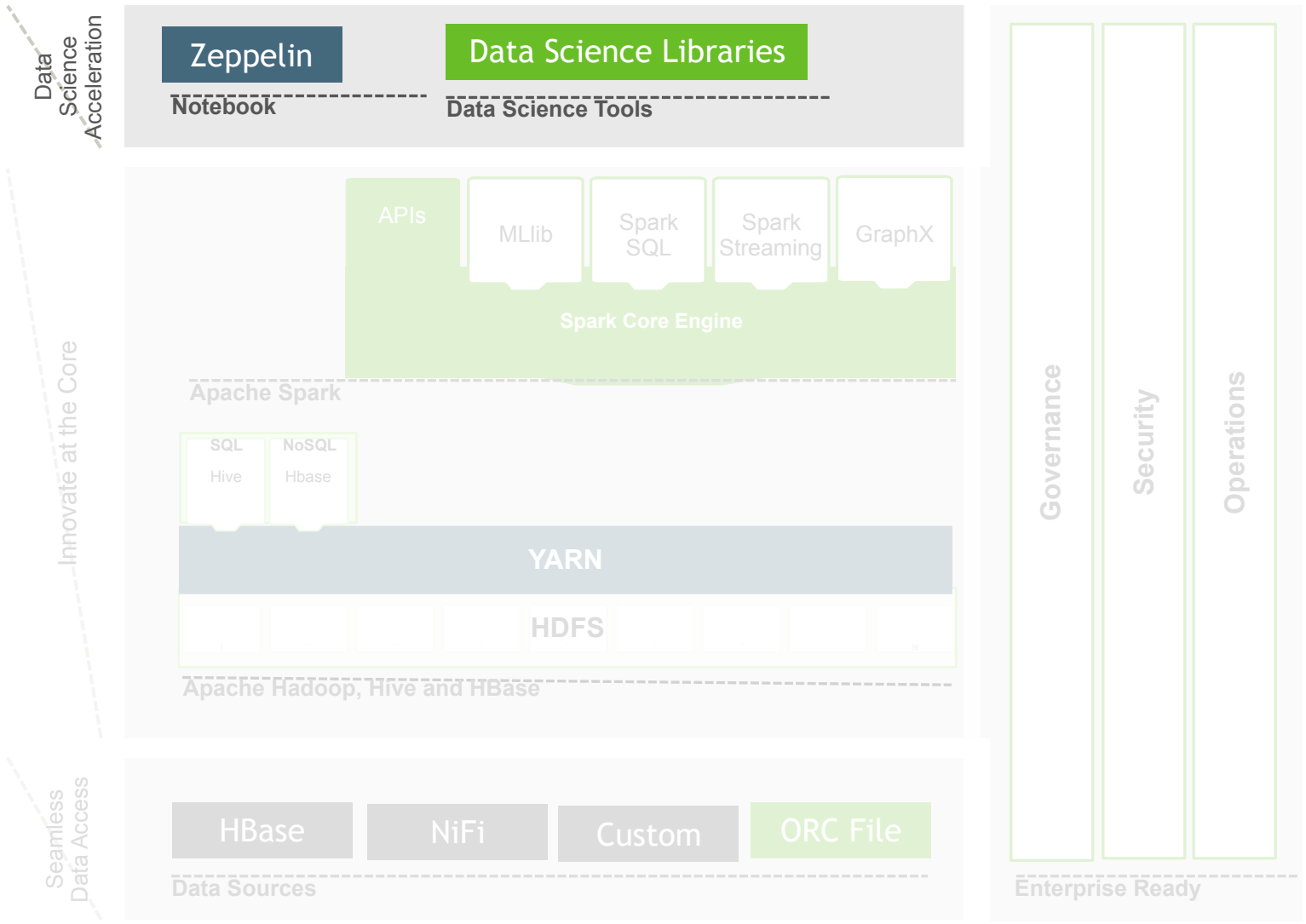
Seamless Data Access

Data Access

- Hive ORC
- Hbase Connector
- NiFi Streams

Hortonworks

Spark + Hadoop - The Road Ahead



Data Science Acceleration

Apache Zeppelin

- Spark development and visualization

Data Science Libraries

- Jump start tools leveraging Spark Machine Learning (ie Magellan, Entity Resolution)

Core Machine Learning

- One vs Rest classifiers, multi-class classifiers
- Serializing Machine Learning pipelines
- Data set loader for public data sets

Hortonworks

Open Web-based Notebook for interactive analytics

Features

Ad-hoc experimentation

Deeply integrated with Spark + Hadoop

Supports multiple language backends

Incubating at Apache

Use Case

Data exploration and discovery
Visualization

Interactive snippet-at-a-time experience

“Modern Data Science Studio”

Geospatial Insight

Where do people go on weekends?
Does usage pattern change with time?
Predict the drop off point of a user?
Predict the location where next pick up can be expected?

Identify crime hotspots
How do these hotspots evolve with time?
Predict the likelihood of crime occurring at a given neighborhood

Predict climate at fairly granular level
Climate insurance: do I need to buy insurance for my crops?
Climate as a factor in crime: Join climate dataset with Crimes

Hortonworks

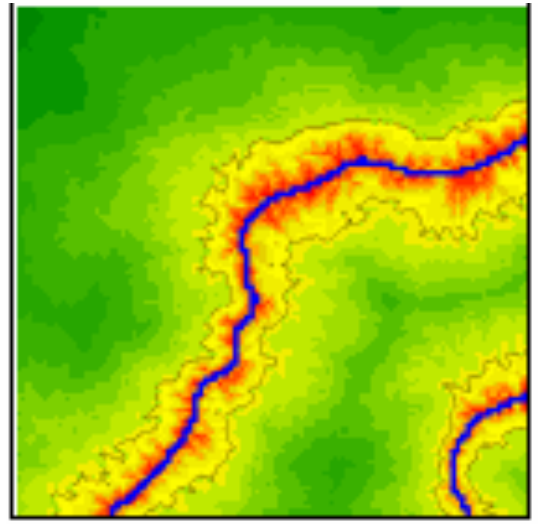
Magellan on Spark Packages

What

- Brings GeoSpatial Analytics to Big Data powered by Spark
- Available at Spark Packages
  http://spark-packages.org/package/harsha2010/magellan

Key Features

- Parse geospatial data and metadata into Shapes + Metadata
- Python and Scala support
- Efficient Geometric Queries
  - ESRI Hive Library
  - **simple** and **intuitive** syntax
- Scalable implementations of common algorithms



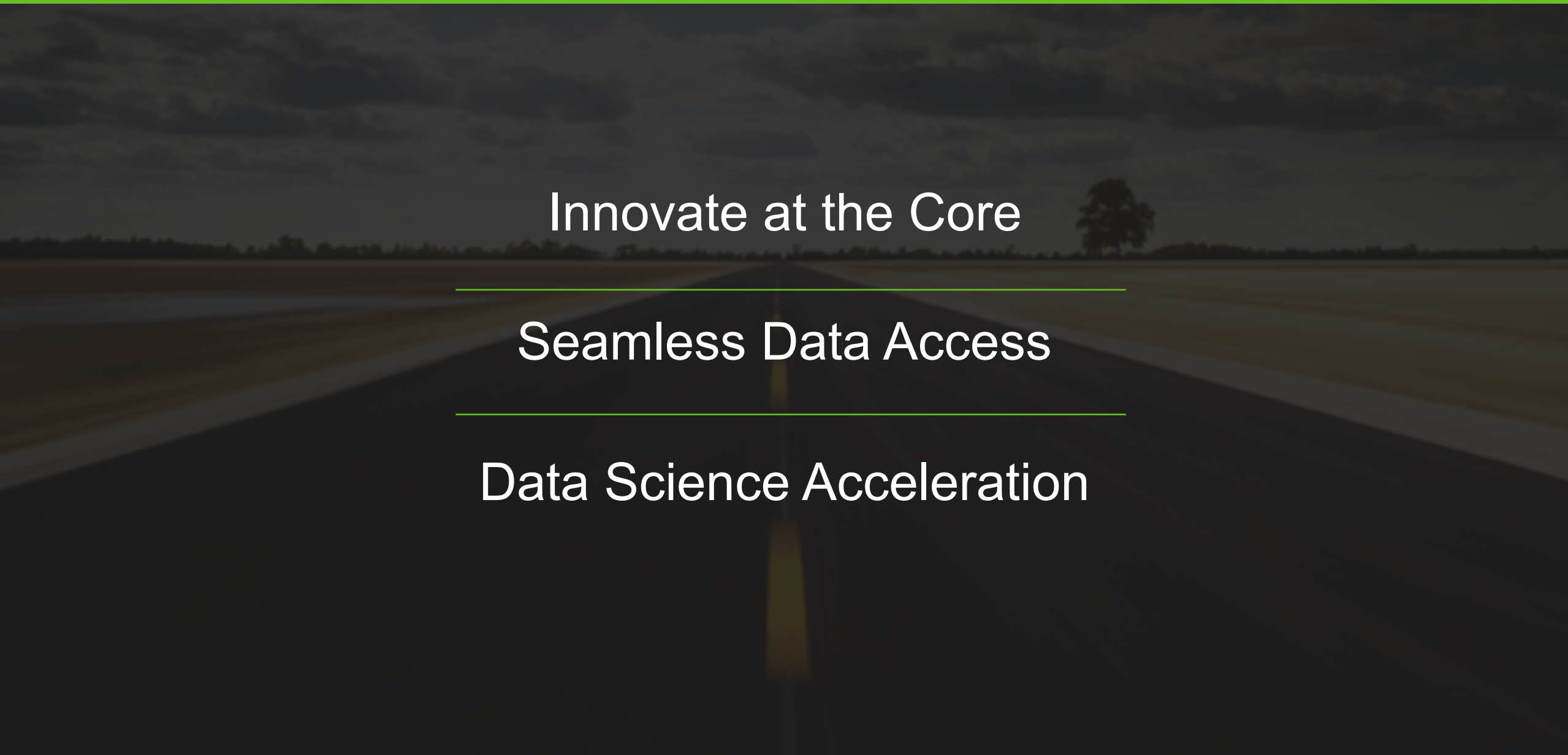
Find Sample Magellan Notebook:
http://tinyurl.com/zeppelinNotebooks

Learn More

- Magellan Blog: http://tinyurl.com/magellanBlog

Hortonworks

Spark + Hadoop - The Road Ahead



Innovate at the Core

Seamless Data Access

Data Science Acceleration

Magellan Walkthrough

Hortonworks

Maximizing Revenue for UBER drivers

Data Insight Opportunity

- Uber publishes anonymized GeoSpatial trip data
- City of San Francisco has an active Open Data program
  - demographics, neighborhoods, traffic

Challenges

- What neighborhood should a driver hangout to maximize revenue?

Solution

- Leverage Spark to transform and aggregate data
- Use Magellan to do GeoSpatial Queries

uber-magellan-nb

FINISHED

```
// Load Neighborhood DataSet as a Data Source using Magellan
val neighborhoods = magellanContext.read.format("magellan").
    load("/neighborhoods").
    select($"polygon", $"metadata").
    cache()

// Join Neighborhoods with Uber dataset to determine the neighborhood each trip event is contained in
val joined = neighborhoods.
join(uberTransformed).
where($"nad83".within($"polygon")).
select($"tripId", $"timestamp", explode($"metadata").as(Seq("k", "v"))).
withColumnRenamed("v", "neighborhood").
drop("k").
cache()

println("%table TripID\tTimestamp\tNeighborhood\n" + joined.take(3).map {
    case Row(tripId: String, timestamp: String, nbd: String) => tripId + "\t" + timestamp + "\t" + nbd
```

TripID	Timestamp	Neighborhood
00,002	2007-01-06T06:23:27+00:00	Marina
00,006	2007-01-04T01:04:58+00:00	Marina
00,008	2007-01-03T00:59:01+00:00	Castro/Upper Market

Zeppelin Notebook Interpreter Connected

uber-magellan-nb

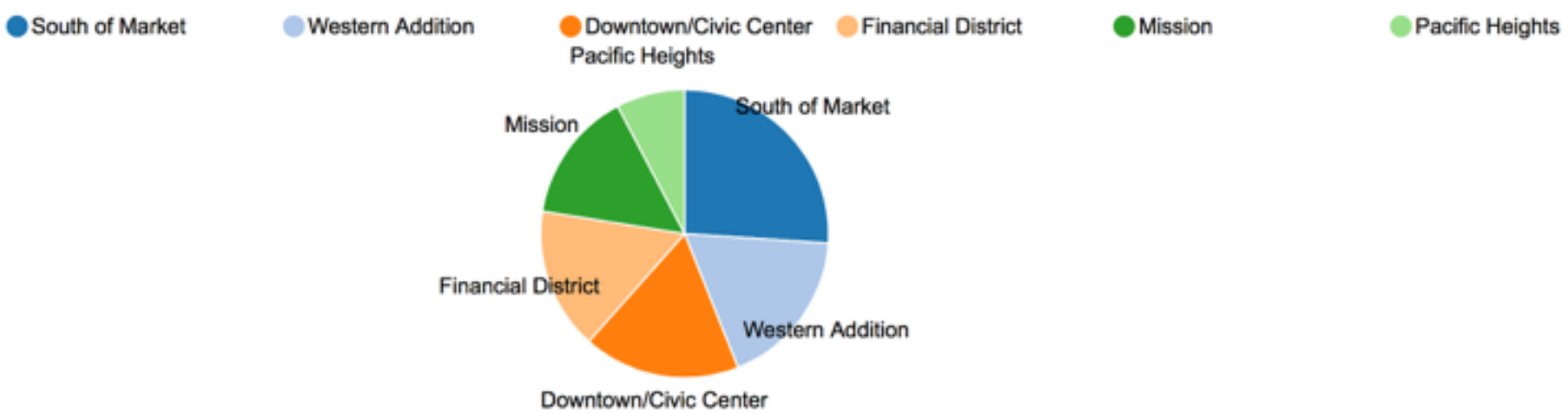
```
// How many uber trips pass through a given neighborhood?

val grouped = joined.
    groupBy($"neighborhood").
    agg(countDistinct("tripId").
    as("trips")).
    orderBy(col("trips").desc)

println("%table Neighborhood\tTrips\n" + grouped.take(6).map { case Row(nbd: String, trips: Long) => nbd + "\t" + trips }.mkString("\n"))
```

FINISHED

settings



Zeppelin Notebook Interpreter Connected

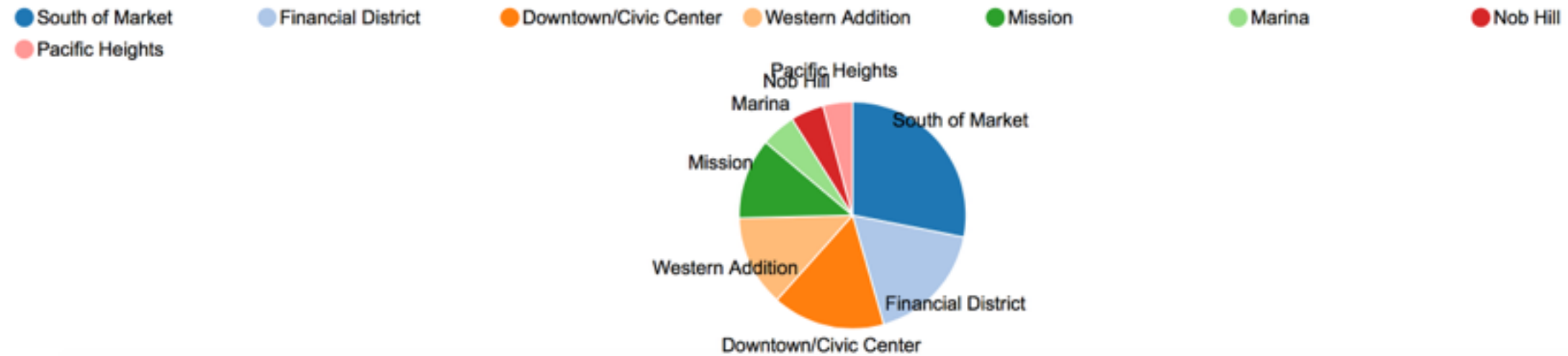
uber-magellan-nb

```
// how many trips start at a given neighborhood?
val df = joined.map { case Row(tripId: String, timestamp: String, neighborhood: String) =>
   (tripId, (timestamp, neighborhood))
  }.groupByKey().map { case (tripId: String, trip: Iterable[(String, String)]) =>
    val sorted = trip.toList.sortBy(_._1)
    val (start_ts, start_nbd) = sorted.head
    val (end_ts, end_nbd) = sorted.last
    (tripId, start_ts, start_nbd, end_ts, end_nbd)
  }.toDF("tripId", "start_timestamp", "start_neighborhood", "end_timestamp", "end_neighborhood")

println("%table Start Neighborhood\tTrips\n" + df.groupBy('start_neighborhood).count().orderBy('count.desc).take(8).map {
    case Row(start_nbd: String, trips: Long) => start_nbd + "\t" + trips
}.mkString("\n"))
```

FINISHED

settings



Zeppelin Notebook Interpreter Connected

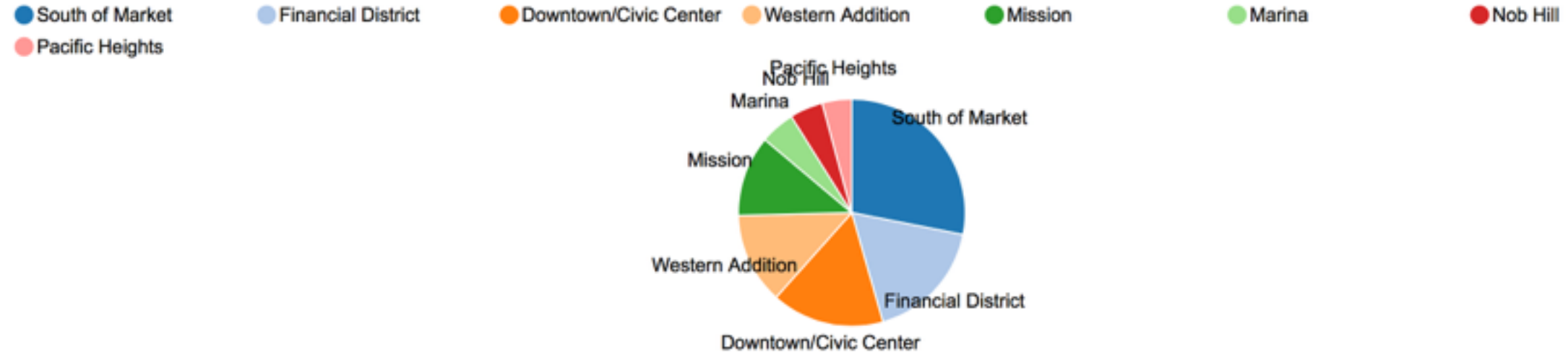
uber-magellan-nb

```
// How many trips start and end in SOMA?

println("%table Start Neighborhood\tTrips\n" + df.groupBy('start_neighborhood).count().orderBy('count.desc).take(8).map {
    case Row(start_nbd: String, trips: Long) => start_nbd + "\t" + trips
}.mkString("\n"))
```

FINISHED

settings



Perfect Together

Hortonworks Data Platform