

Introduction to Hadoop Tools

Timothy Spann

2017 Future of Data – Princeton Meetup
May 16, 2017



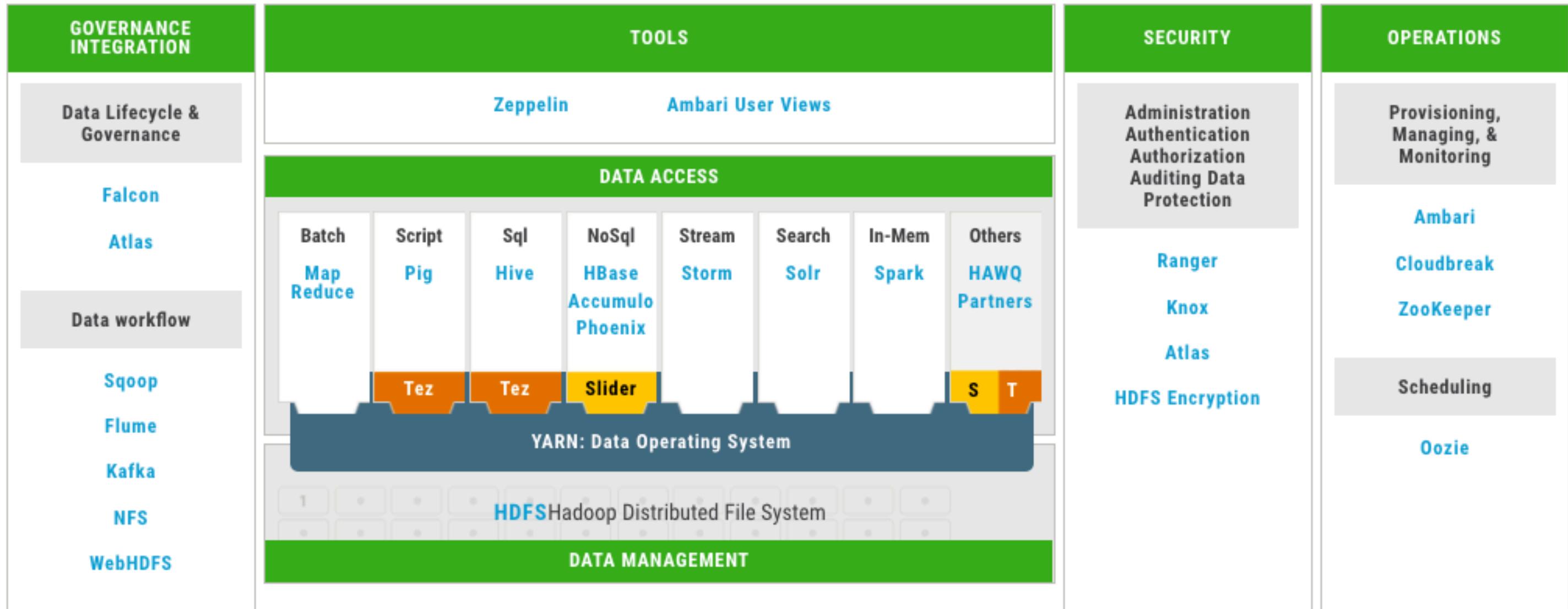
Agenda

- 6:00pm – 6:45pm Registration and Food
- 6:45pm – 7:00pm Introduction and Welcome
- 7:00pm – 7:45 pm Install and Cloud by Milind Pandit
- 7:45pm – 8:30 pm Tools by Tim Spann
- 8:30pm – 8:45pm Questions

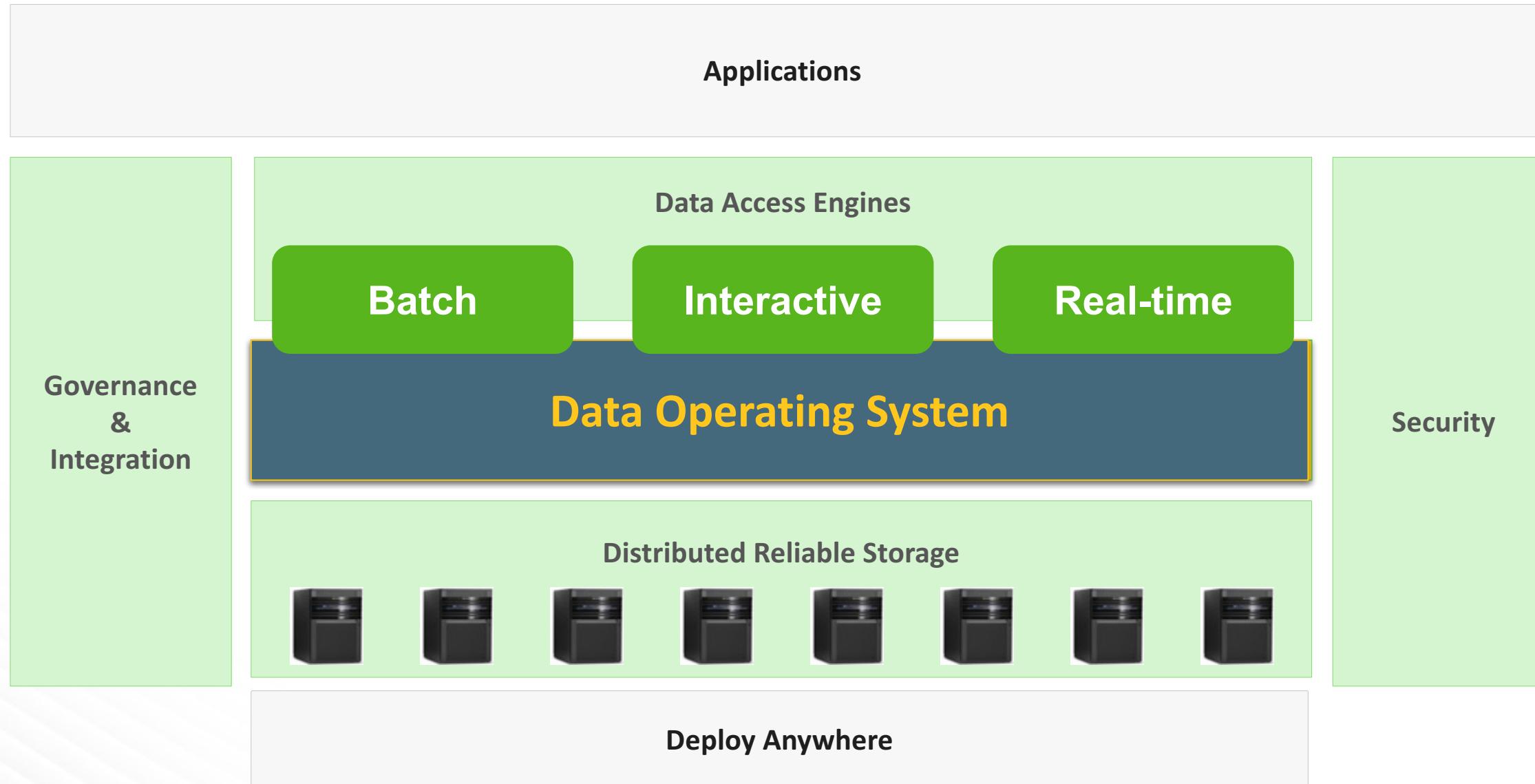
All Your Data Are Belong to Hadoop

- ◆ CSV
- ◆ JSON
- ◆ TSV
- ◆ TEXT
- ◆ PDF
- ◆ XML
- ◆ HTML
- ◆ AVRO
- ◆ PARQUET
- ◆ ORC
- ◆ Sequence File
- ◆ HFile
- ◆ JPEG
- ◆ PNG
- ◆ MP4
- ◆ RTF
- ◆ Word
- ◆ Excel

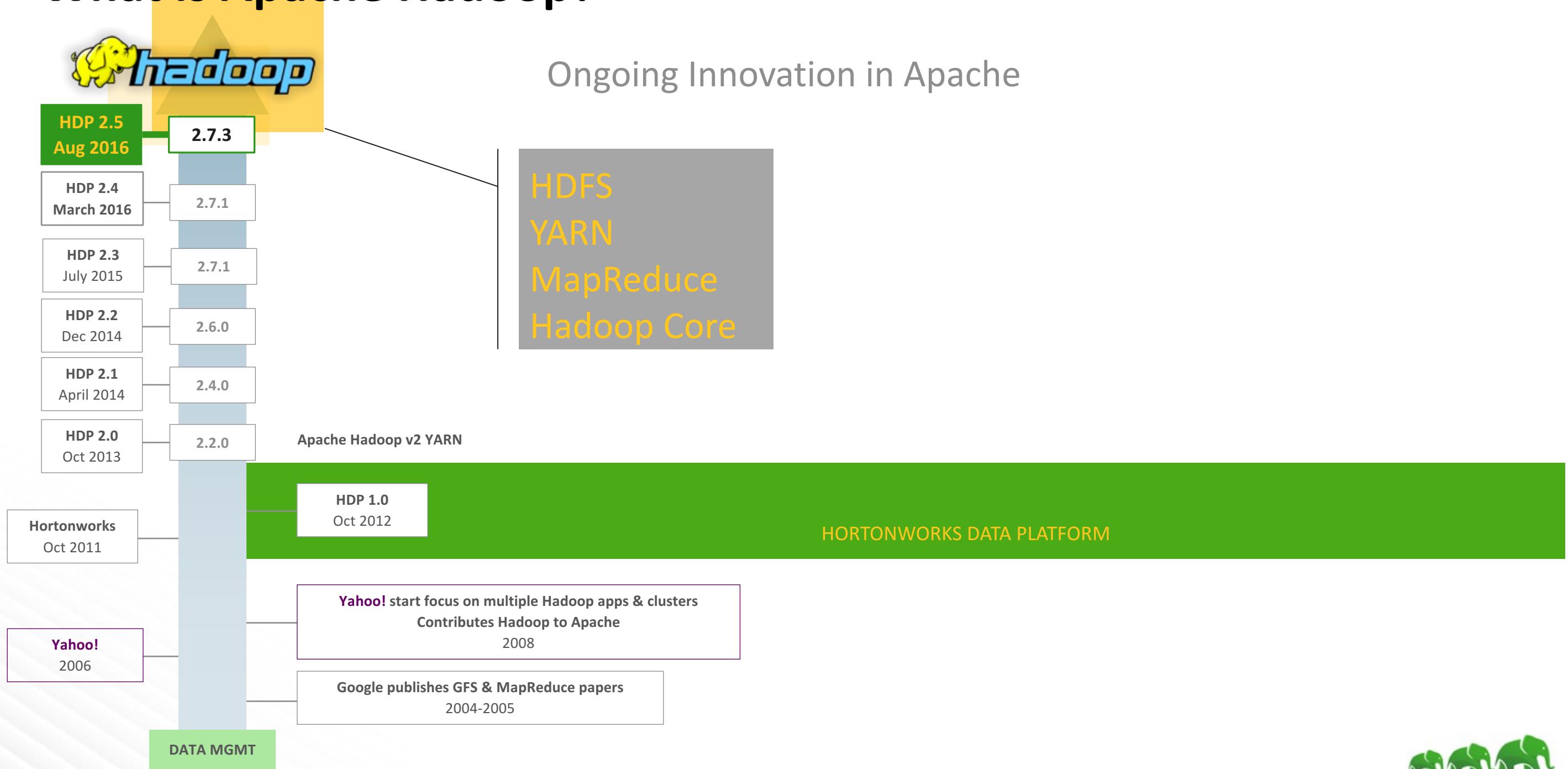
Apache Big Data Ecosystem



Hadoop Architecture

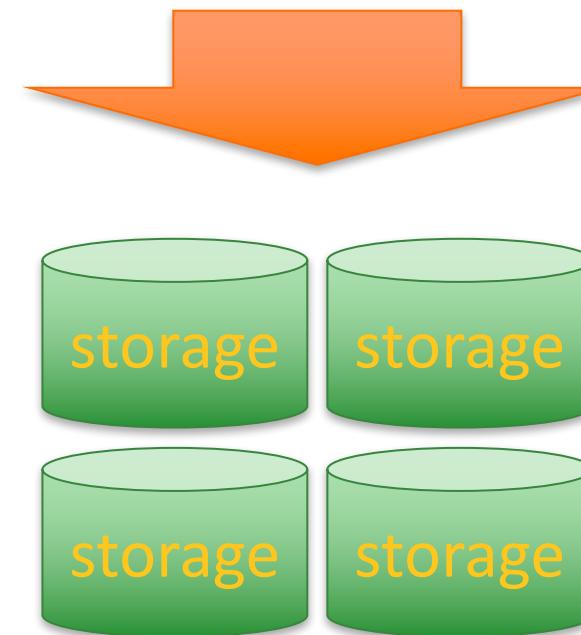


What is Apache Hadoop?

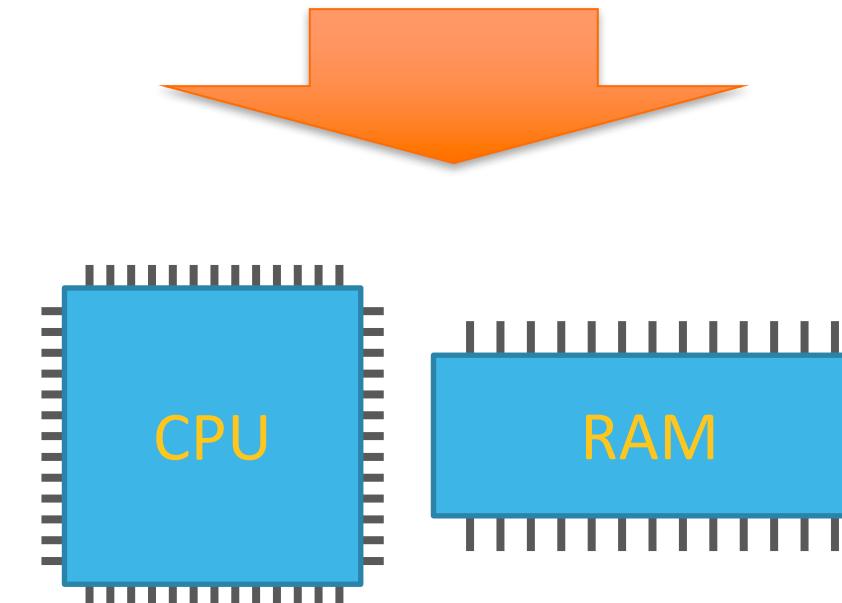


Apache Hadoop = Storage + Compute

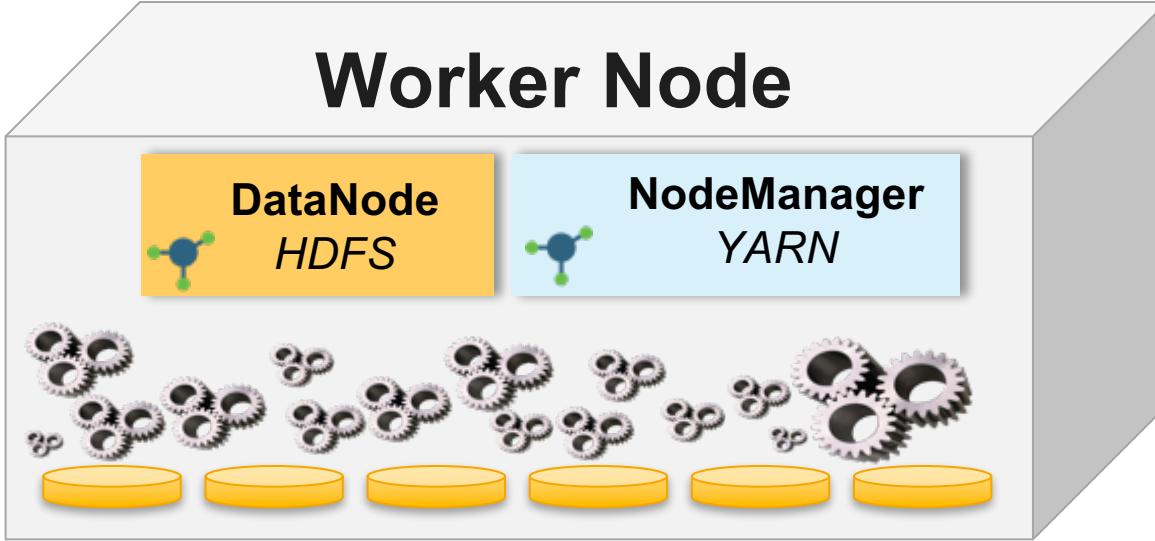
Hadoop Distributed File
System (HDFS)



Yet Another Resource
Negotiator (YARN)



Core **hadoop**



Disk, CPU, Memory

NameNode
HDFS

/directory/structure/in/memory.txt

ResourceManager
YARN

Resource management + scheduling

 **Hadoop daemon**

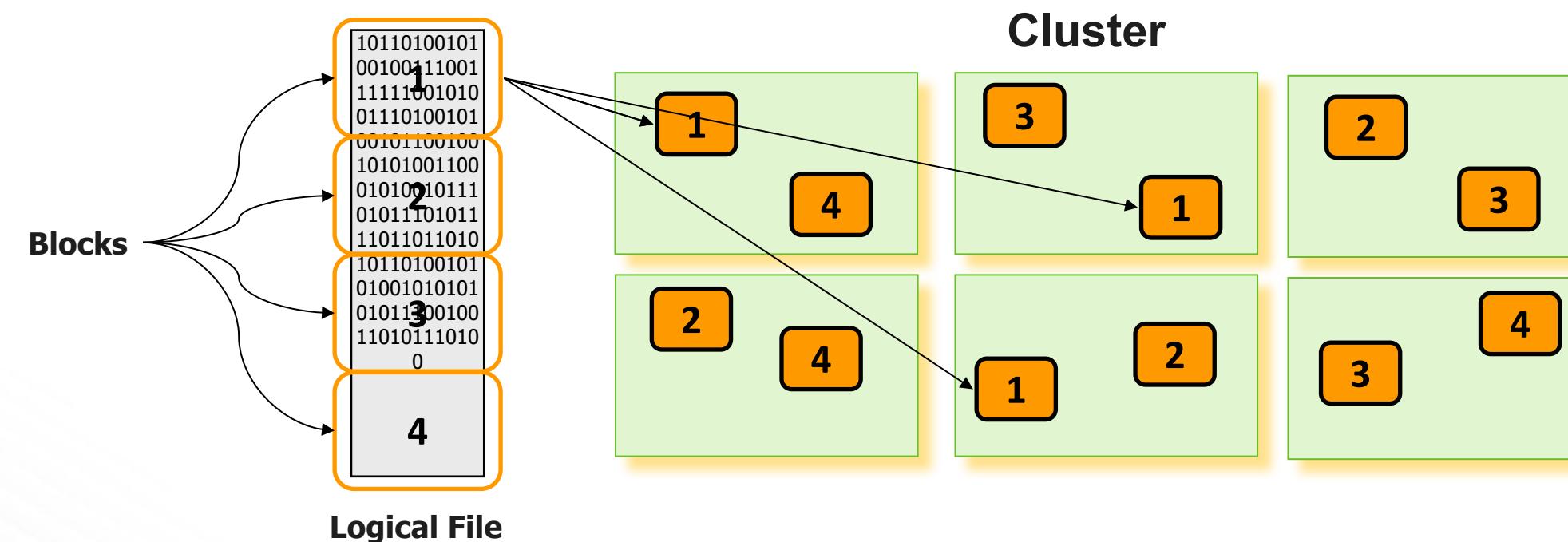
 **User application**



Hadoop Distributed File System (HDFS)

Fault Tolerant Distributed Storage

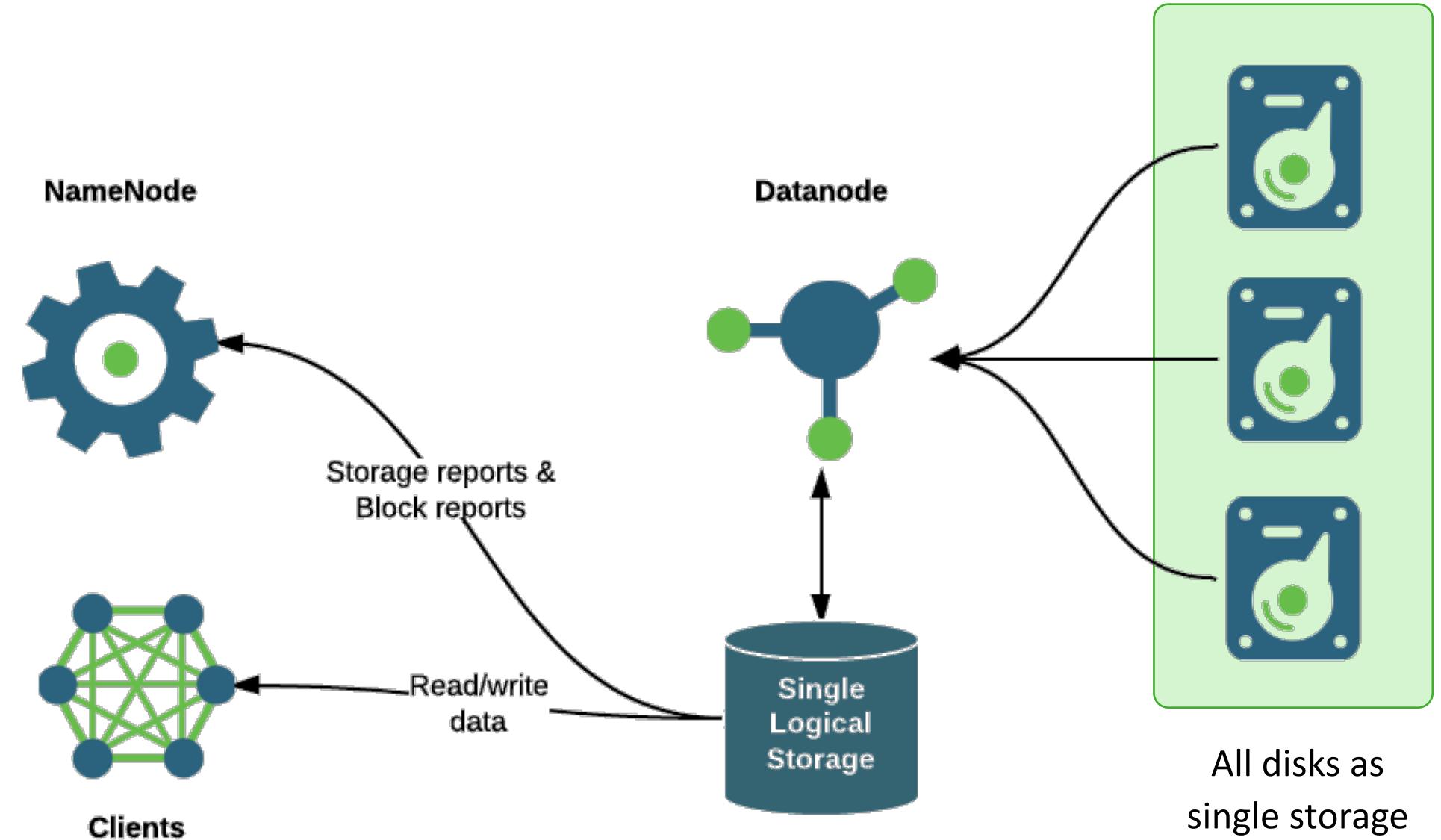
- Divide files into big blocks and distribute 3 copies **randomly** across the cluster
- Processing Data Locality
 - Not Just storage but computation



HDFS Storage Architecture - Before

Before

- DataNode is a single storage
- Storage is uniform - Only storage type Disk
- Storage types hidden from the file system



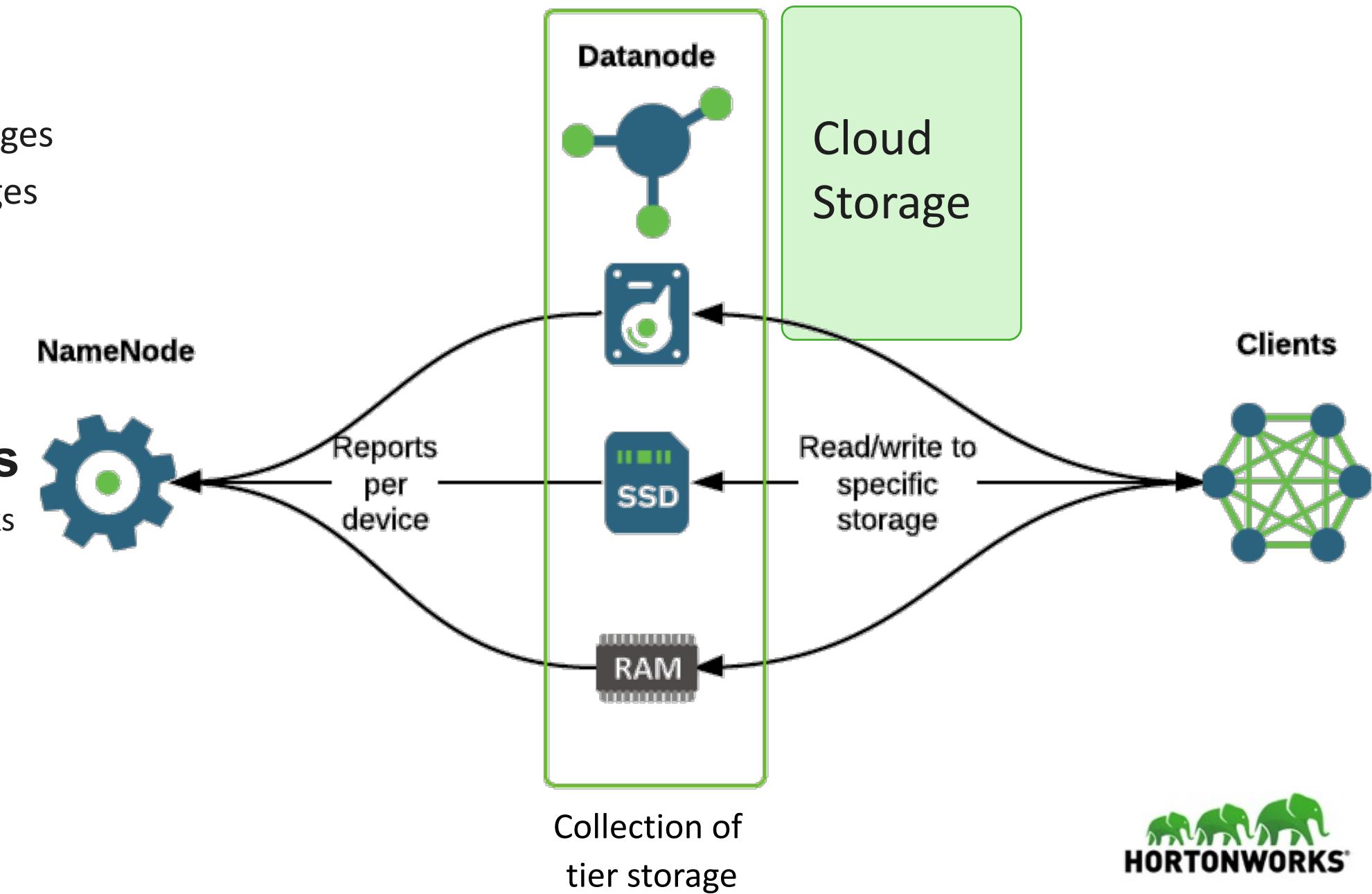
HDFS Storage Architecture - Now

New Architecture

- DataNode is a collection of storages
- Support different types of storages
 - Disk, SSDs, Memory

Block Storage Policies

- Describes how to store data blocks in HDFS



It Looks Like a File System

The screenshot shows a file browser interface for an HDFS-like system. At the top, the path is / user / it1 / geolocation. There are buttons for '+ New directory', 'Browse...', 'Select files to upload.', and a close button 'X'. Below the path, there's a folder icon labeled 'geolocation' with an edit icon. To the right is a search bar 'Search File Names' with a magnifying glass icon. The main area is a table with the following columns: Name, Size, Last Modified, Owner, Group, Permission, and actions (download, move, delete, checkmark). The table contains two rows:

Name	Size	Last Modified	Owner	Group	Permission	
geolocation.csv	514.3 kB	2016-03-13 19:42	maria_dev	hdfs	-rw-r--r--	<input type="checkbox"/>
trucks.csv	59.9 kB	2016-03-13 19:41	maria_dev	hdfs	-rw-r--r--	<input type="checkbox"/>

Below the table, there's a terminal window showing the output of the command 'hdfs dfs -ls /user/it1/geolocation':

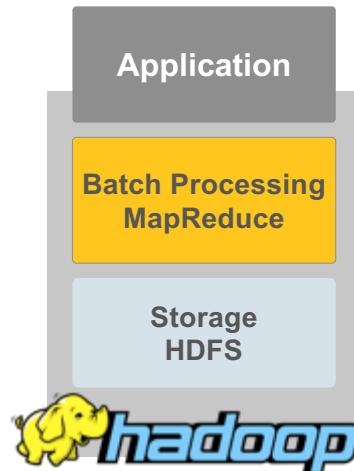
```
[it1@sandbox ~]$ hdfs dfs -ls /user/it1/geolocation
Found 2 items
-rw-r--r-- 3 maria_dev hdfs      526677 2016-03-13 23:42 /user/it1/geolocation/geolocation.csv
-rw-r--r-- 3 maria_dev hdfs      61378 2016-03-13 23:41 /user/it1/geolocation/trucks.csv
[it1@sandbox ~]$
```

WebHDFS

<http://princeton.server.com:50070/webhdfs/v1/demo/clickstream/weblogs/clickstream-feed-generated.tsv?OP=open>

<https://community.hortonworks.com/articles/60480/using-images-stored-in-hdfs-for-web-pages.html>

Hadoop emerged as foundation of new data architecture



Apache Hadoop is an open source data platform for managing large volumes of high velocity and variety of data

- Built by Yahoo! to be the heartbeat of its ad & search business
- Donated to Apache Software Foundation in 2005 with rapid adoption by large web properties & early adopter enterprises
- Incredibly disruptive to current platform economics

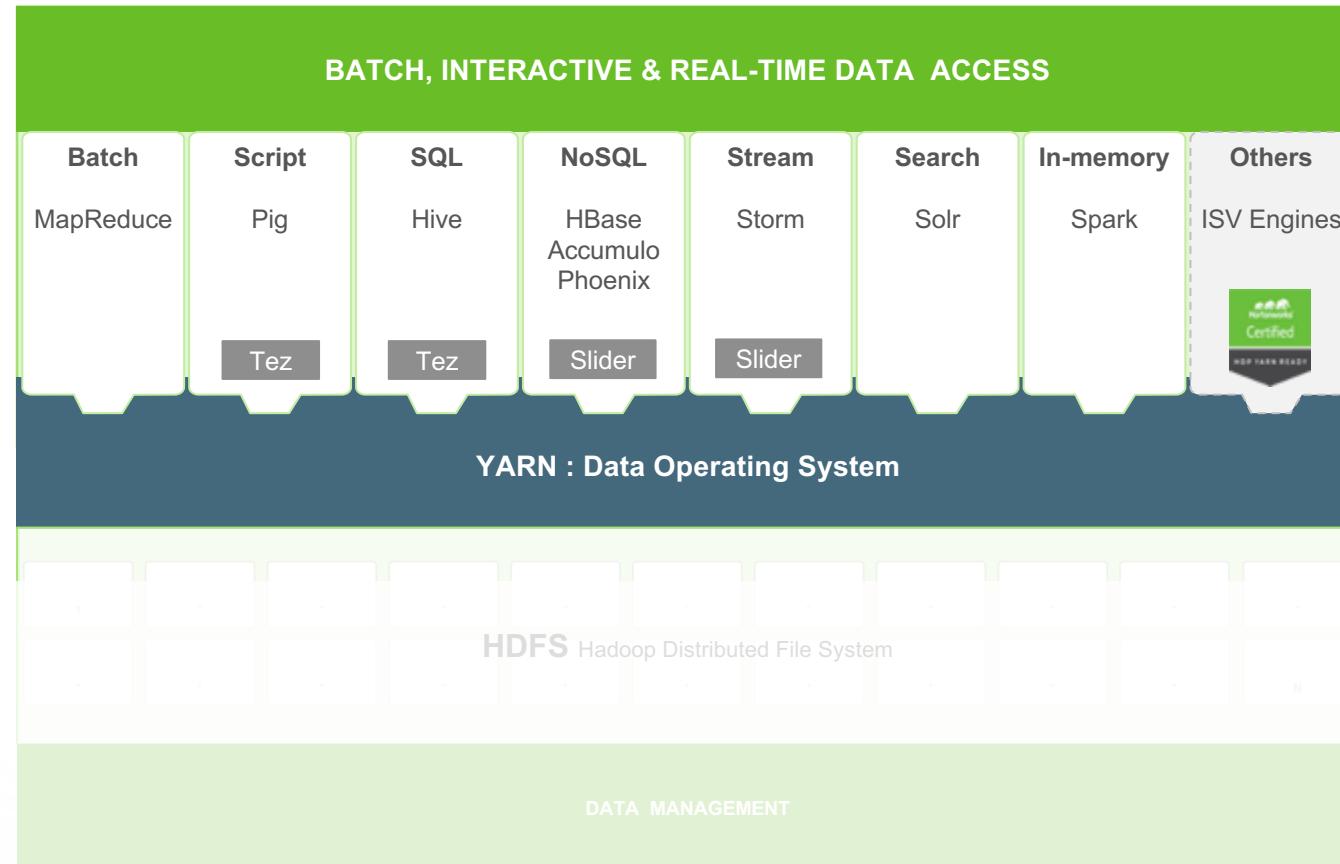
Traditional Hadoop Advantages

- ✓ Manages new data paradigm
- ✓ Handles data at scale
- ✓ Cost effective
- ✓ Open source

Traditional Hadoop Had Limitations

- ✗ Batch-only architecture
- ✗ Single purpose clusters, specific data sets
- ✗ Difficult to integrate with existing investments
- ✗ Not enterprise-grade

YARN extends Hadoop into data center leaders



YARN The Architectural Center of Hadoop

- Common data platform, many applications
- Support multi-tenant access & processing
- Batch, interactive & real-time use cases
- Supports 3rd-party ISV tools
(ex. SAS, Syncsort, Actian, etc.)



YARN Ready Applications

Facilitates ongoing innovation and enterprise adoption via ecosystem of new and existing “YARN Ready” solutions

Overview of SQL on Hadoop Solutions



Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.



Spark's module for working with structured data. Run SQL queries alongside complex analytic algorithms.



High performance relational database layer over HBase for low latency applications.

Traditional
MPP on
Hadoop



Many traditionally architected MPP solutions have been ported to Hadoop and some new ones have been developed from scratch.

SQL on Hadoop: Vitals

Project	First GA Release	Lines of Code (June 2015) ^(*)	Most Typical Use
Apache Hive	April, 2009 (7 Years)	1 Million	EDW / ETL Offload
SparkSQL	March, 2015 (4 Months)	56.6k	Exploratory Analytics
Apache Phoenix	March, 2014 (2 Year)	200k	Low-Latency Dashboards

Apache Hive: Fast Facts

Most Queries Per Hour

100,000 Queries Per Hour
(Yahoo Japan)

Analytics Performance

100 Million rows/s Per Node
(with Hive LLAP)

Largest Hive Warehouse

300+ PB Raw Storage
(Facebook)

Largest Cluster

4,500+ Nodes
(Yahoo)

Phoenix and HBase: Fast Facts

Largest Database

5 Petabytes
(Flurry)

Fastest Ingestion

10 Million Events/s
(Yahoo)

Best Known App

Facebook Messages
(Facebook)

Biggest SQL App

Real-Time SQL on 140m+ Records
(PubMatic)

Apache Hive: Journey to SQL:2011 Analytics

Data Types	SQL Features	File Formats
Numeric	Core SQL Features	Columnar
FLOAT/DOUBLE	Date, Time and Arithmetical Functions	ORCFile
DECIMAL	INNER, OUTER, CROSS and SEMI Joins	Parquet
INT/TINYINT/SMALLINT/BIGINT	Derived Table Subqueries	
BOOLEAN	Correlated and Uncorrelated Subqueries	
String	UNION ALL	Text
CHAR / VARCHAR	UDFs, UDAFs, UDTFs	CSV
STRING	Common Table Expressions	Logfile
BINARY	UNION DISTINCT	
Date, Time	INTERSECT, EXCEPT	Nested / Complex
DATE	Non-Equality Joins	Avro
TIMESTAMP		JSON
Interval Types		XML
Complex Types	Advanced Analytics	Custom Formats
ARRAY	OLAP and Windowing Functions	
MAP	CUBE and Grouping Sets	
STRUCT		Other Features
UNION		XPath Analytics
	Nested Data Analytics	Procedural Extensions (PL/Hive)
	Nested Data Traversal	
	Lateral Views	
	ACID Transactions	
	INSERT / UPDATE / DELETE	
	MERGE	

Legend

- Hive 1.0
- Hive 1.2
- Future





Apache Hive facilitates **querying and managing large datasets.**

Hive provides SQL on Hadoop.

Data analysts use Hive to **explore, structure and analyze** that **data** using the familiar comfortable SQL syntax they are used to.

Hive also comes with **HCatalog**; a global metadata management layer that exposes **Hive table metadata** to all other Hadoop applications.



Apache Zeppelin is a Web-based notebook that enables interactive data analytics.

Data Scientists and End-Users alike can make beautiful data-driven, interactive and collaborative documents with SparkSQL, Scala, Python, JDBC connections, Files, and more.

Notebooks contain code samples, source data, descriptive markup, result sets, and rich visualizations.

Australian Dataset (SparkSQL example)



Register RDD as table

FINISHED ▶ ✎ 📈 ⏷

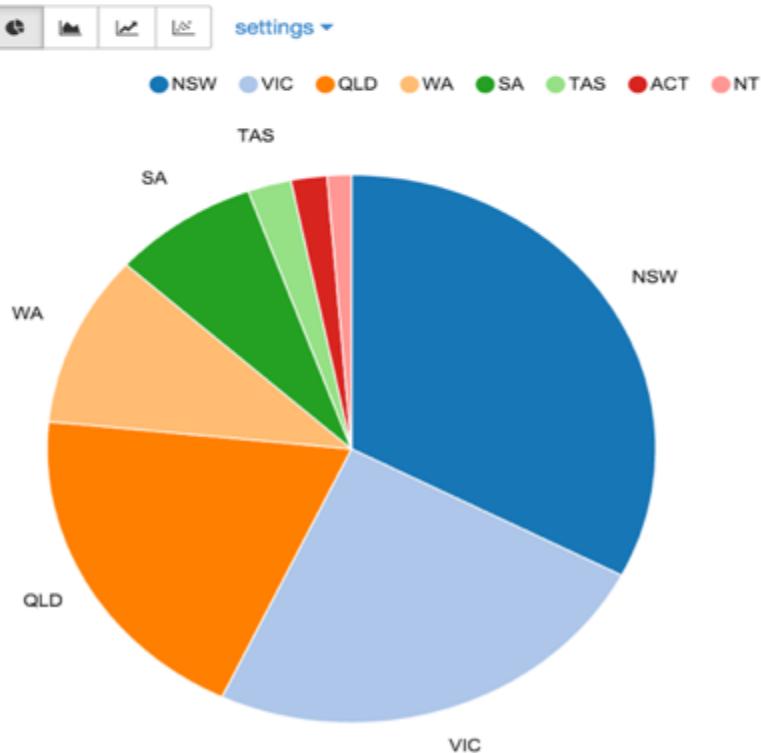
```
case class Health (year: String, state: String, category: String, funding_src1: String, funding_src2: String, spending: Integer)
val health = dataset.map(k=>k.split(",")).map(
  k => Health(k(0),k(1),k(2),k(3), k(4), k(5).toInt)
)
// toDF() works only in spark 1.3.0.
// For spark 1.1.x and spark 1.2.x,
// use below instead:
// health.registerTempTable("health_table")
health.toDF().registerTempTable("health_table")
```

```
defined class Health
health: org.apache.spark.rdd.RDD[Health] = MapPartitionsRDD[7] at map at <console>:33
Took 3 seconds
```

Spending (in billions) by state

FINISHED ▶ ✎ 📈 ⏷

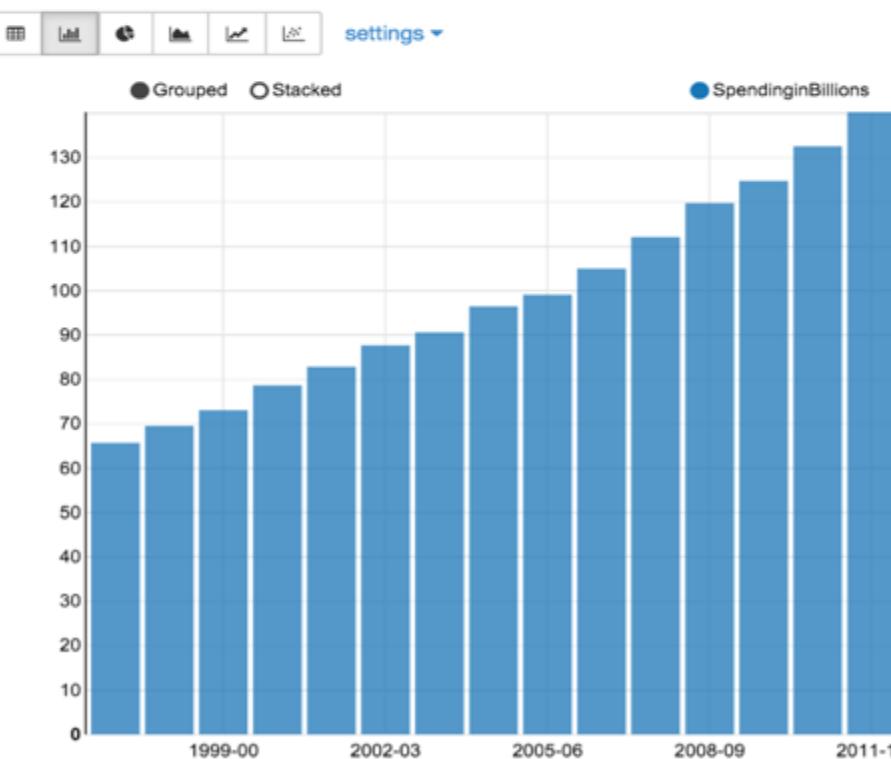
```
%sql
select state, sum(spending)/1000 SpendinginBillions
from health_table
group by state
order by SpendinginBillions desc
```



Spending (In Billions) By Year

FINISHED ▶ ✎ 📈 ⏷

```
%sql
select year,sum(spending)/1000 SpendinginBillions
from health_table
group by year
order by SpendinginBillions
```



Spending (in billions) by area

FINISHED ▶ ✎ 📈 ⏷

```
%sql
select category, sum(spending)/1000 SpendinginBillions
from health_table
group by category
order by SpendinginBillions desc
```

category	SpendinginBillions
Public hospitals	445.845
Medical services	272.507
Private hospitals	121.022
Benefit-paid pharmaceuticals	104.221
Dental services	90.786
Community health	75.765
Capital expenditure	72.698
All other medications	70.508
Other health practitioners	51.382
Administration	41.029
Research	40.074
Aids and appliances	37.155
Patient transport services	28.174
Public health	27.072
Medical expense tax rebate	0.0



Ambari Views are a built-in set of Views that are pre-deployed for you to use with your cluster.

These GUI components increase ease-of-use to end users. Current Ambari Views include Hive, Pig, Tez, Capacity Scheduler, File, HDFS.

The **Ambari Views Framework** allow developers to create new user interface components that plug into the Ambari web interface.

Ambari foo 0ops Alerts

Dashboard Services Hosts Alerts Admin ambari-qa

Hive Query Saved Queries History UDFs

Database Explorer

consumption

Search tables...

Databases

- consumption
- power
- power2
- adate
- atime
- global_active_power
- voltage
- global_intensity
- sub_metering_1
- sub_metering_2
- sub_metering_3
- power3
- power4
- sample_03
- sample_04
- default

Query Editor

Worksheet

```
1 insert into table power4
2 select adate, sum(p.Global_active_power)
3 from power p
4 join power2 p2
5 on p.adate=p2.adate
6 group by p.adate;
```

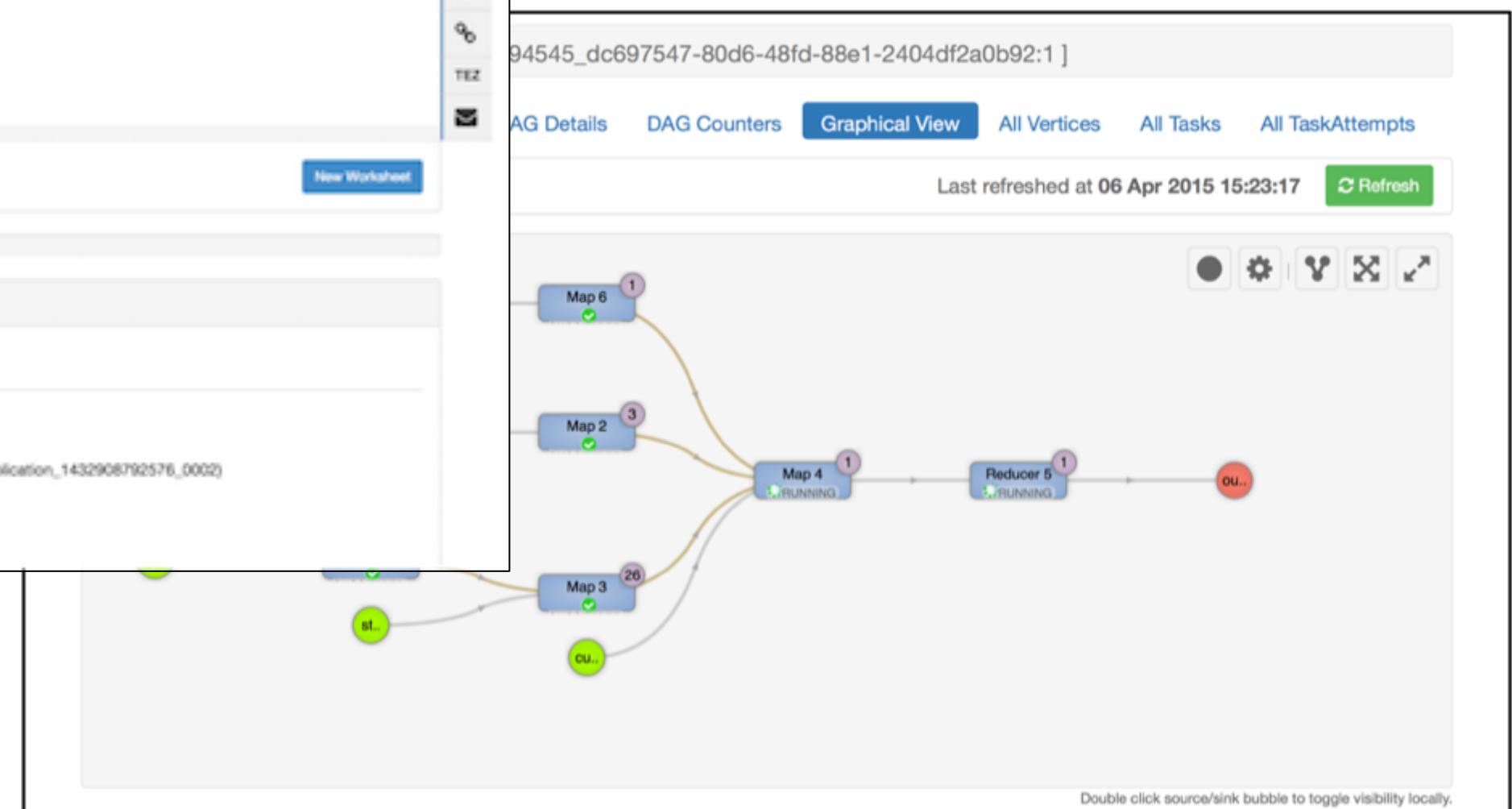
Execute Explain Save as... New Worksheet

30%

Query Process Results (Status: RUNNING)

Logs Results

INFO : Tez session hasn't been created yet. Opening session
 INFO :
 INFO : Status: Running (Executing on YARN cluster with App id application_1432908792576_0002)
 INFO : Map 1: 0/- Map 3: 0/- Reducer 2: 0/4
 INFO : Map 1: 0/1 Map 3: 0/1 Reducer 2: 0/4
 INFO : Map 1: 0/1 Map 3: 0/1 Reducer 2: 0/4

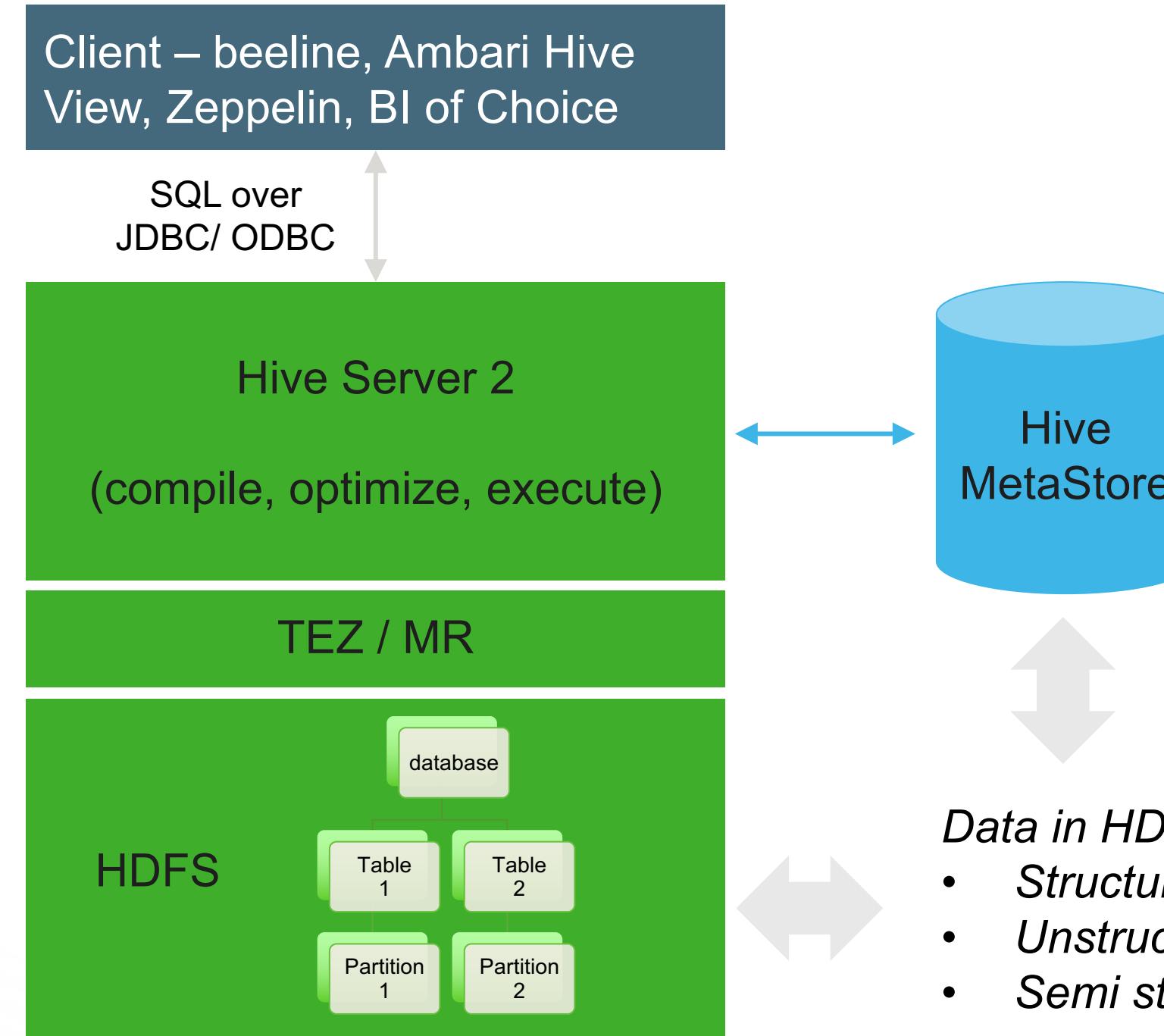


Apache Hive Architecture

Interpreter

Distribution Engine

Data Storage



Schema definitions

- Data in HDFS*
- Structured
 - Unstructured
 - Semi structured

Submitting Hive Queries

- ◆ Hive CLI

- Traditional Hive client that connects to a HiveServer instance
 - \$ hive
hive>

- ◆ Beeline

- A new command line client that connects to a HiveServer2 instance
 - \$ beeline
beeline> !connect
jdbc:hive2://hostname:10000 *username password*
org.apache.hive.jdbc.HiveDriver



Best Practices for Data Loading - Create External Table, Create ORC Table

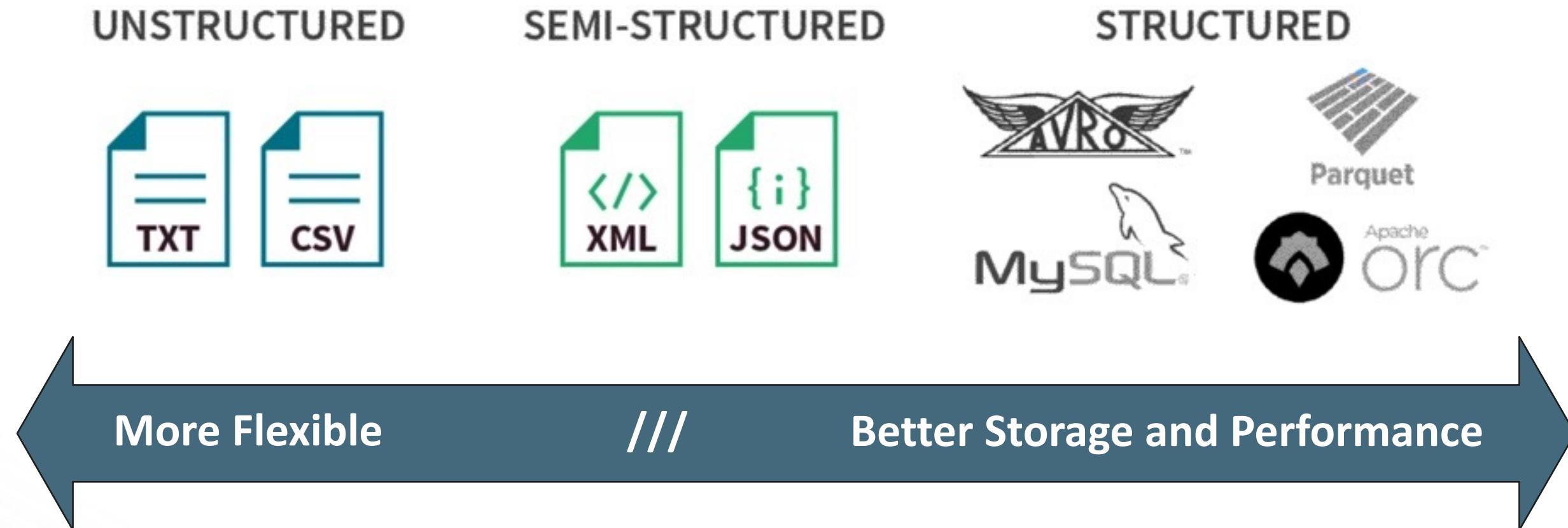
```
hdfs dfs -copyFromLocal cars.csv /visdata
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS Cars(Name STRING, Miles_per_Gallon INT, Cylinders  
INT, Displacement INT, Horsepower INT, Weight_in_lbs INT, Acceleration DECIMAL, Year DATE,  
Origin CHAR(1)) COMMENT 'Data' ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
STORED AS TEXTFILE location '/visdata';
```

```
CREATE TABLE IF NOT EXISTS mycars( Name STRING, Miles_per_Gallon INT, Cylinders INT,  
Displacement INT, Horsepower INT, Weight_in_lbs INT, Acceleration DECIMAL, Year DATE, Origin  
CHAR(1)) COMMENT 'Data' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS  
ORC;
```

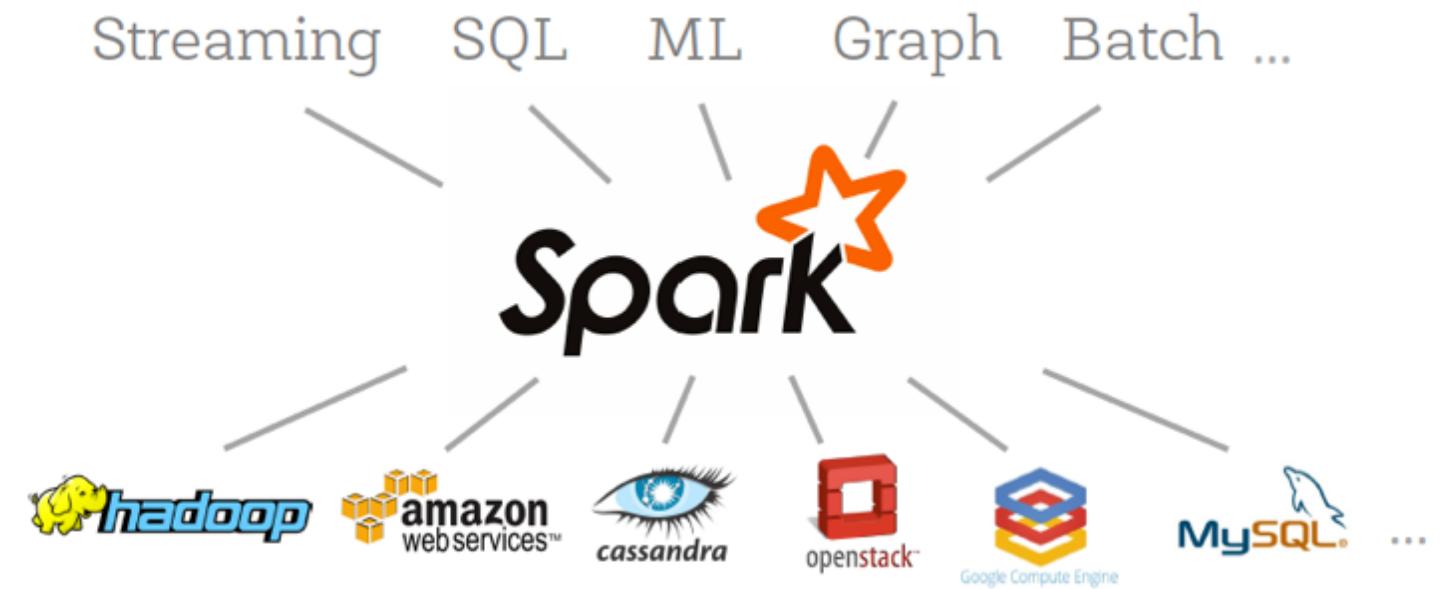
```
INSERT OVERWRITE TABLE mycars SELECT * FROM cars;
```

Data Formats



What Is Apache Spark?

- ◆ **Apache open source project** originally developed at AMPLab (University of California Berkeley)
- ◆ **Unified data processing engine** that operates across varied data workloads and platforms



Why Apache Spark?

- ◆ **Elegant Developer APIs**
 - Single environment for data munging, data wrangling, and Machine Learning (ML)
- ◆ **In-memory computation model – Fast!**
 - Effective for iterative computations and ML
- ◆ **Machine Learning**
 - Implementation of distributed ML algorithms
 - Pipeline API (Spark ML)

Spark SQL

Structured Data

Spark Streaming

Near Real-time

Spark MLlib

Machine Learning

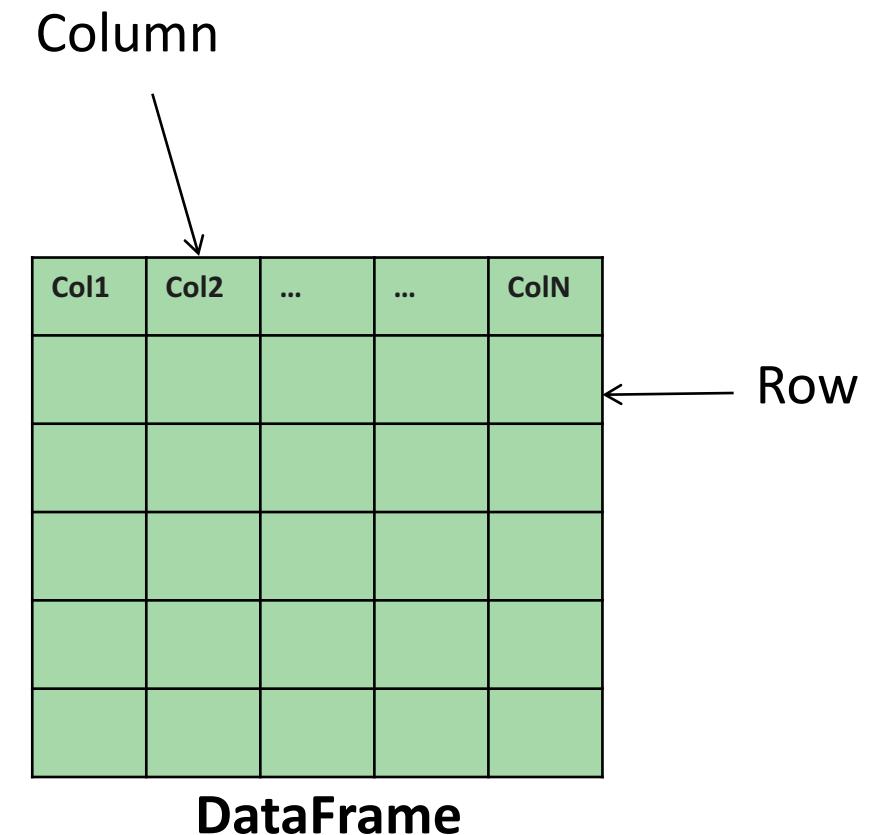
GraphX

Graph Analysis



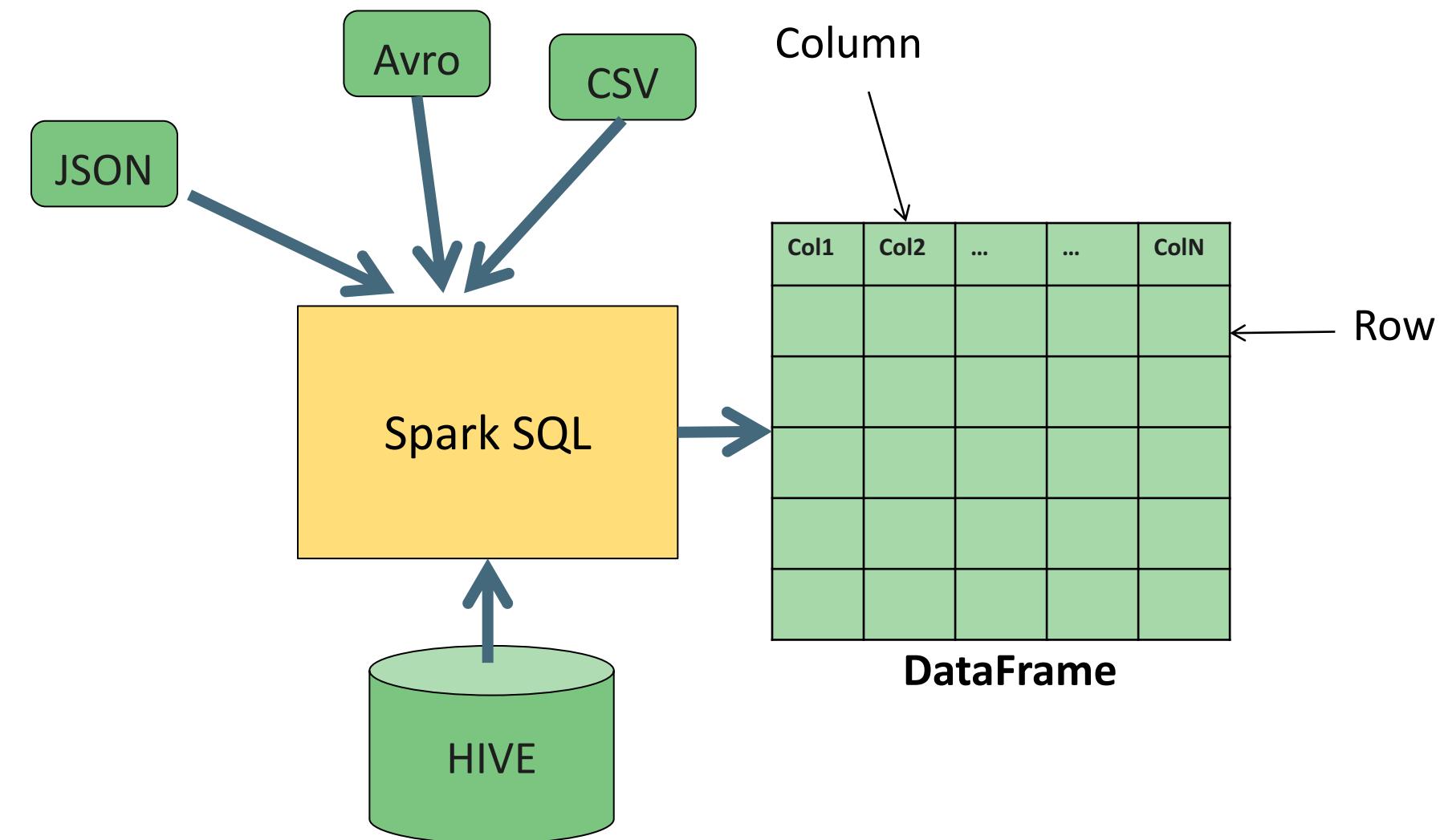
DataFrames

- ◆ **Distributed collection of data organized into *named columns***
- ◆ Conceptually equivalent to a table in **relational DB** or a **data frame in R/Python**
- ◆ API available in Scala, Java, Python, and R



Data is described as a DataFrame with **rows**, **columns**, and a **schema**

Sources



Create a DataFrame

Example

```
val path = "examples/flights.json"  
val flights = spark.read.json(path)
```

Rapid Ecosystem Adoption: 210+ Processors

FTP
SFTP
HL7
UDP
XML
⋮
HTTP
Email
HTML
Image
Syslog
AMQP

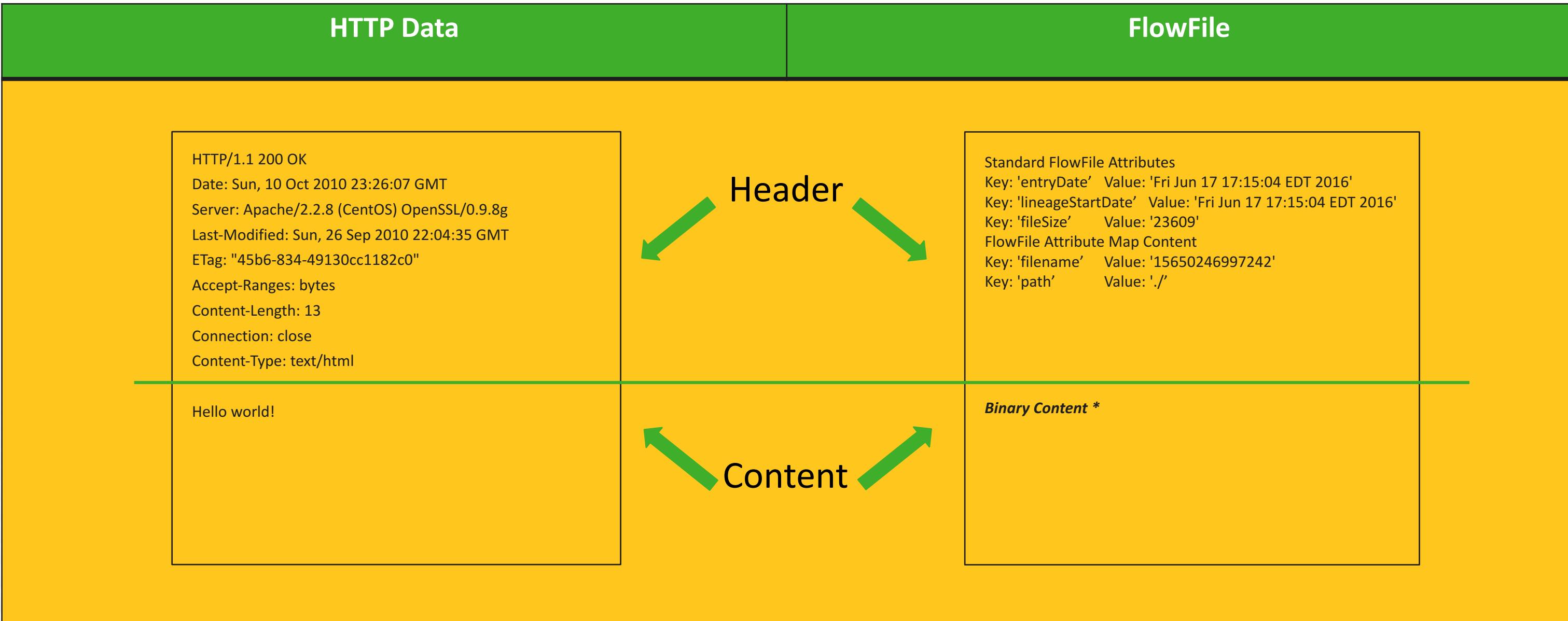


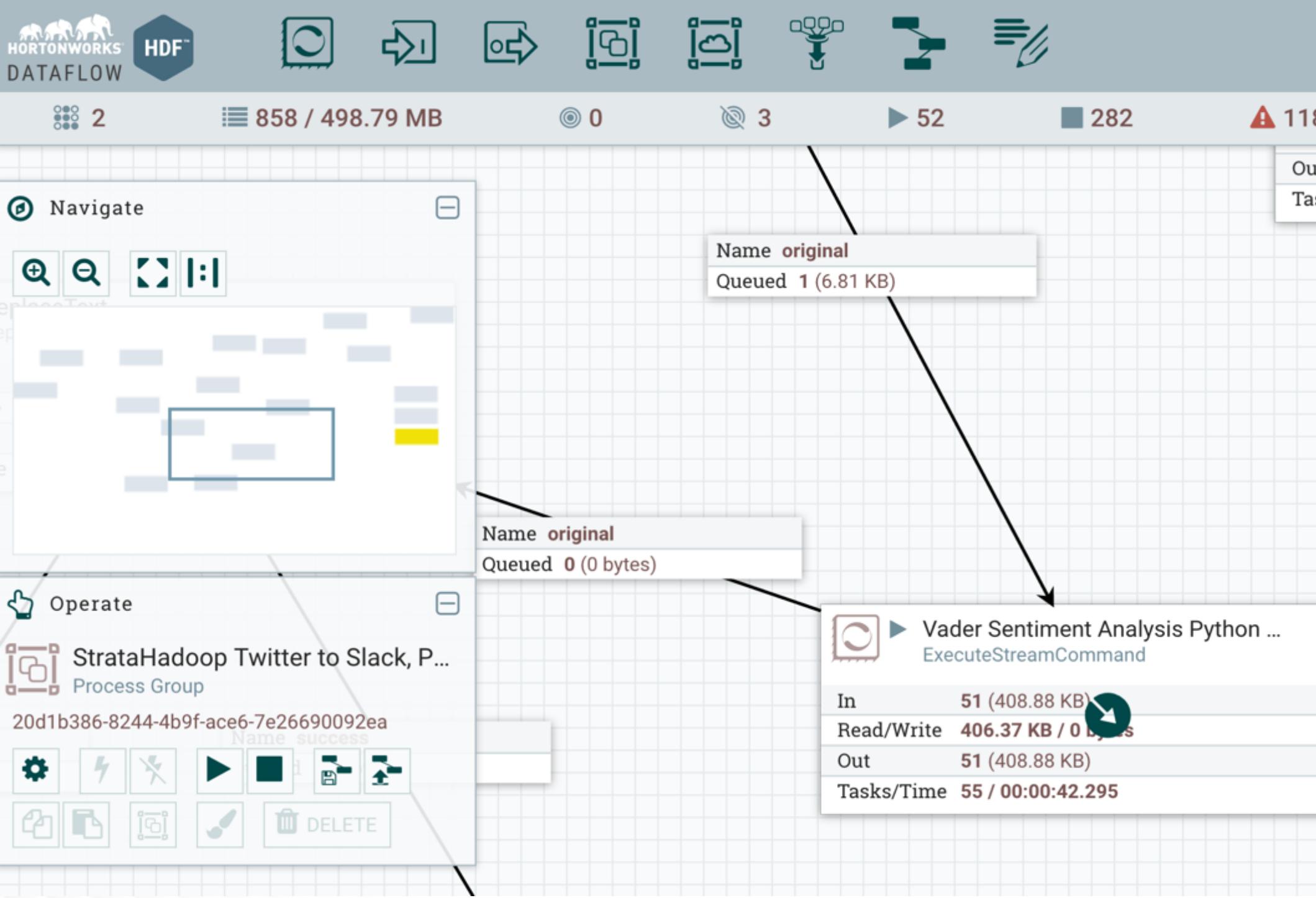
Hash	Encrypt	GeoEnrich
Merge	Tail	Scan
Extract	Evaluate	Replace
Duplicate	Execute	Translate
Split	⋮	⋮

Route Text
Route Content
Route Context
Control Rate
Distribute Load



FlowFiles are like HTTP data





<http://hortonworks.com/blog/hdf-2-0-flow-processing-real-time-tweets-strata-hadoop-slack-tensorflow-phoenix-zeppelin/>

Quick Terms and Reference

FlowFile: Each piece of "User Data" (i.e., data that the user brings into NiFi for processing and distribution) is referred to as a FlowFile. A FlowFile is made up of two parts: Attributes and Content. The Content is the User Data itself. Attributes are key-value pairs that are associated with the User Data.

Processor: The Processor is the NiFi component that is responsible for creating, sending, receiving, transforming, routing, splitting, merging, and processing FlowFiles. It is the most important building block available to NiFi users to build their dataflows.

<https://nifi.apache.org/docs/nifi-docs/html/getting-started.html>

<https://nifi.apache.org/docs/nifi-docs/html/overview.html>

<http://www.slideshare.net/aldrinpiri/apache-nifi-crash-course-san-jose-hadoop-summit-66967077>

<https://hortonworks.com/hadoop-tutorial/learning-ropes-apache-nifi/>

Input

InvokeHttp

GetTwitter

Processing

RouteOnAttribute

ExecuteStreamCommand

UpdateAttribute

Output

PutHDFS: Have access to your Hadoop HDFS from the NIFI box and have this configuration:
`/etc/hadoop/conf/core-site.xml` . Also create a directory to use like `hdfs dfs –mkdir /nifi-place`

PutSQL: Create a connection pool, know your JDBC information.

1. **Phoenix:** `URL= jdbc:phoenix:clusterzookeeper:2181:/hbase-unsecure`
`org.apache.phoenix.jdbc.PhoenixDriver file:///opt/demo/phoenix-client.jar`
`User= root pool=2` You will need the JDBC JAR on the local file system.
2. **MySQL:**
`jdbc:mysql://tspanndev11.field.hortonworks.com:3306/datacom.mysql.jdbc.Driver/usr/share/java/mysql-connector-java.jar`

PutSlack

Need to get your webhook URL from your slack site. You can go to slack.com and get your own free room to test with.

<https://api.slack.com/incoming-webhooks>

Local TensorFlow via Python or C++ Binary

```
python classify_image.py --image_file /opt/demo/dronedataold/Bebop2_20160920083655-0400.jpg
solar dish, solar collector, solar furnace (score = 0.98316)
window screen (score = 0.00196)
manhole cover (score = 0.00070)
radiator (score = 0.00041)
doormat, welcome mat (score = 0.00041)
```

```
bazel-bin/tensorflow/examples/label_image/label_image --
image=/opt/demo/dronedataold/Bebop2_20160920083655-0400.jpg
tensorflow/examples/label_image/main.cc:204] solar dish (577): 0.983162
tensorflow/examples/label_image/main.cc:204] window screen (912): 0.00196204
tensorflow/examples/label_image/main.cc:204] manhole cover (763): 0.000704005
tensorflow/examples/label_image/main.cc:204] radiator (571): 0.000408321
tensorflow/examples/label_image/main.cc:204] doormat (972): 0.000406186
```

Local Sentiment Analysis via Python

/opt/demo/sentiment/run.sh

```
python /opt/demo/sentiment/sentiment.py "$@"
```

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import sys
sid = SentimentIntensityAnalyzer()
ss = sid.polarity_scores(sys.argv[1])
print('Compound {0} Negative {1} Neutral {2} Positive {3}'.format(
    ss['compound'],ss['neg'],ss['neu'],ss['pos']))
```

or

```
if ss['compound'] == 0.00:    print('Neutral')
elif ss['compound'] < 0.00:   print ('Negative')
else:                      print('Positive')
```

Installing NLTK for Python 2.7

<https://pip.pypa.io/en/latest/installing/>

<http://www.nltk.org/install.html>

```
wget https://bootstrap.pypa.io/get-pip.py
```

```
python get-pip.py
```

```
sudo pip install -U nltk
```

```
sudo pip install -U numpy
```

Installing TensorFlow for Python 2.7

Installing TensorFlow is a very difficult exercise, after getting NLTK you can start the process. You will need most of the development tools for Python, C, C++, Bezel, Pip and more. A beefy machine with a lot of RAM, CPUs and GPUs would be useful.

Check out my install article for a guide:

<https://dzone.com/articles/deep-learning-resources>

Results

Twitter From Strata Hadoop Processing

Twitter from Strata Hadoop using Phoenix, NiFi, TensorFlow, Python

Took 0 sec. Last updated by anonymous at October 20 2016, 2:19:46 AM. (outdated)

```
%phoenix
```

```
select count(*) from tweets
```



COUNT(1)

306,314

Results

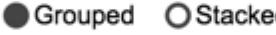
 **Zeppelin** Notebook anonymous

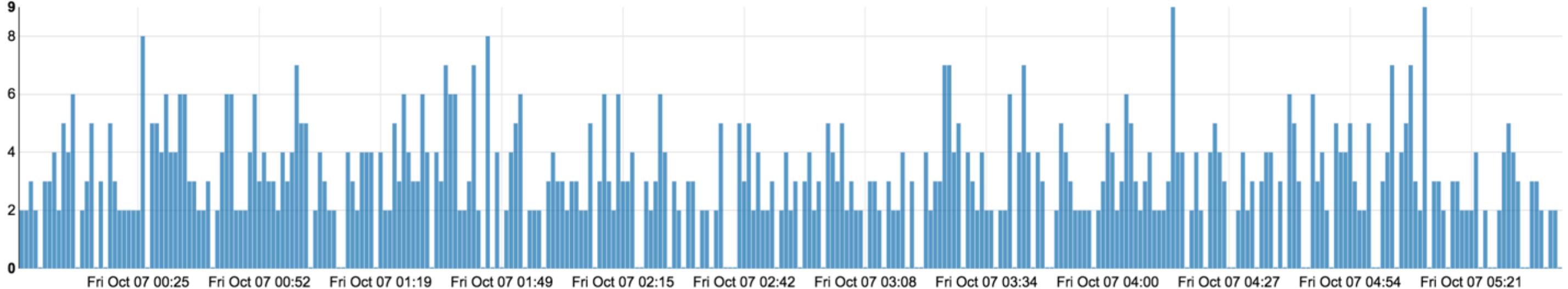
Twitter From Strata Hadoop Processing



```
%phoenix
select minute, sentiment, count(1) as cnt from
(
  select substr(time, 0, 16) as minute, handle, sentiment, length(msg) as wordcount, msg from tweets
  order by time desc
) sub1
group by minute, sentiment
order by minute
```



The chart displays the count of tweets (cnt) for each sentiment category (positive, neutral, negative) across one-minute intervals on Friday, October 7, 2016. The Y-axis represents the count (0 to 9), and the X-axis shows time intervals from 00:00 to 05:21. The chart uses a blue color scheme for all bars, indicating a single sentiment category per bar.

Installation

Download the binary from here:

<http://hortonworks.com/downloads/#dataflow>

Or here:

<https://nifi.apache.org/download.html>

Or on Mac:

brew install nifi

<https://nifi.apache.org/docs/nifi-docs/html/getting-started.html#starting-nifi>

bin/nifi.sh start

bin/nifi.sh install (now it's installed as a service on Linux)

Contact:

Timothy Spann @PaaSDeV

www.meetup.com/futureofdata-princeton

community.hortonworks.com/users/9304/tspann.html

<http://wwwcoreservlets.com/hadoop-tutorial/>

<https://www.slideshare.net/HadoopSummit/hadoop-crash-course>





Hortonworks Community Connection

The screenshot displays the Hortonworks Community Connection website's 'ANSWERS' section. It features a grid of categories with counts and associated tags:

- DS, Analytics & Spark**: 77 tags (Spark, Hive, sparksql, hdp2.3.2, pyspark)
- Governance & Lifecycle**: 50 tags (Falcon, Oozie, Sqoop, teradata, apache falcon)
- Data Ingestion & Streaming**: 128 tags (Nifi, hdf, Sqoop, Flume, Storm)
- Community Help**: 27 tags (help, community, forums, answerhub, forum)
- Cloud & Operations**: 282 tags (Ambari, installation, Hive, operations, HDFS)
- Data Processing**: 227 tags (Hive, Tez, Hbase, Pig, MapReduce)
- Sandbox & Learning**: 90 tags (Sandbox, Hive, sandbox-2.3.2, Pig, virtualbox)
- Hadoop Core**: 147 tags (HDFS, YARN, hadoop, Ambari, Hbase)

Other sections visible include a 'LEADERBOARD' with user profiles, 'POPULAR TAGS' (Hive, Ambari, HDFS, Spark, Nifi, security, Hbase, kerberos, Ranger, Sandbox, YARN, hadoop), and a 'RECENT BADGES' section.

- Full Q&A Platform (like StackOverflow)
- Knowledge Base Articles
- Code Samples and Repositories

Read access for everyone, join to participate and be recognized



Community Engagement

One Website!

4,000+

Registered Users

10,000+

Answers

15,000+

Technical Assets

The screenshot shows the Hortonworks Community Connection website interface. At the top, there's a search bar, a 'SEARCH' button, and a 'ASK A QUESTION' button. Below the search bar, it says '1302 Questions'. To the right, there are 'CREATE', 'TRACKS', and user profile icons. The main area is divided into several sections:

- ANSWERS**: Sub-sections include DS, Analytics & Spark (54 answers), Governance & Lifecycle (22 answers), Data Ingestion & Streaming (60 answers), Cloud & Operations (125 answers), Data Processing (115 answers), Sandbox & Learning (50 answers), and Hadoop Core (42 answers).
- LEADERBOARD**: Shows a grid of user profiles.
- POPULAR TAGS**: Includes tags like Hive, Ambari, HDFS, Spark, Nifi, security, Hbase, kerberos, Ranger, Sandbox, YARN, hadoop, etc.
- RECENT BADGES**: Lists recent badge earners: Ryan Tomczik, vshukla, mettleton, and surrender nath reddy.