Force multiplier for data science: Introduction to H2O

Hank Roark
@hankroark
hank@h2o.ai

DEMONSTRATION

WHO AMI I

Lead, Customer Data Science @ H2O.ai

John Deere: Research, Software Product Development, High Tech Ventures

Lots of time dealing with data off of machines, equipment, satellites, weather, radar, hand sampled, and on.

Geospatial, temporal / time series data almost all from sensors.

Previously at startups and consulting (Red Sky Interactive, Nuforia, NetExplorer, Perot Systems, a few of my own)

Engineering & Management MIT
Physics Georgia Tech

hank@h2oai.com
@hankroark
https://www.linkedin.com/in/hankroark

WHAT IS H2O?

Math Platform — Open source in-memory prediction engine

- Parallelized and distributed algorithms making the most use out of multithreaded systems
- GLM, Random Forest, GBM, Deep Learning, etc.

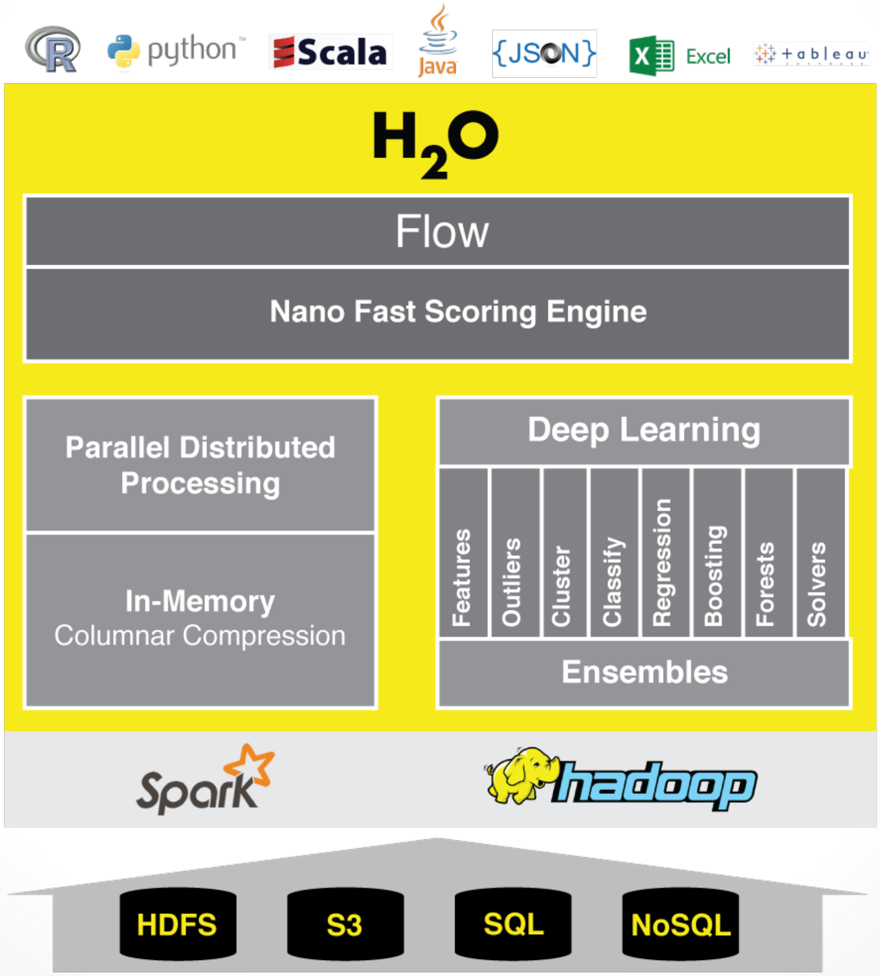
API — Easy to use and adopt

- Written in Java – perfect for Java Programmers
- REST API (JSON) – drives H2O from Browser UI, R, Python, Tableau

Big Data — More data? Or better models? BOTH

- Use all of your data – model without down sampling
- Run a simple GLM or a more complex GBM to find the best fit for the data
- More Data + Better Models = Better Predictions

ACCURACY WITH SPEED AND SCALE



H_2O

Flow

Nano Fast Scoring Engine

Parallel Distributed Processing

In-Memory
Columnar Compression

Deep Learning

Features

Outliers

Cluster

Classify

Regression

Boosting

Forests

Solvers

Ensembles

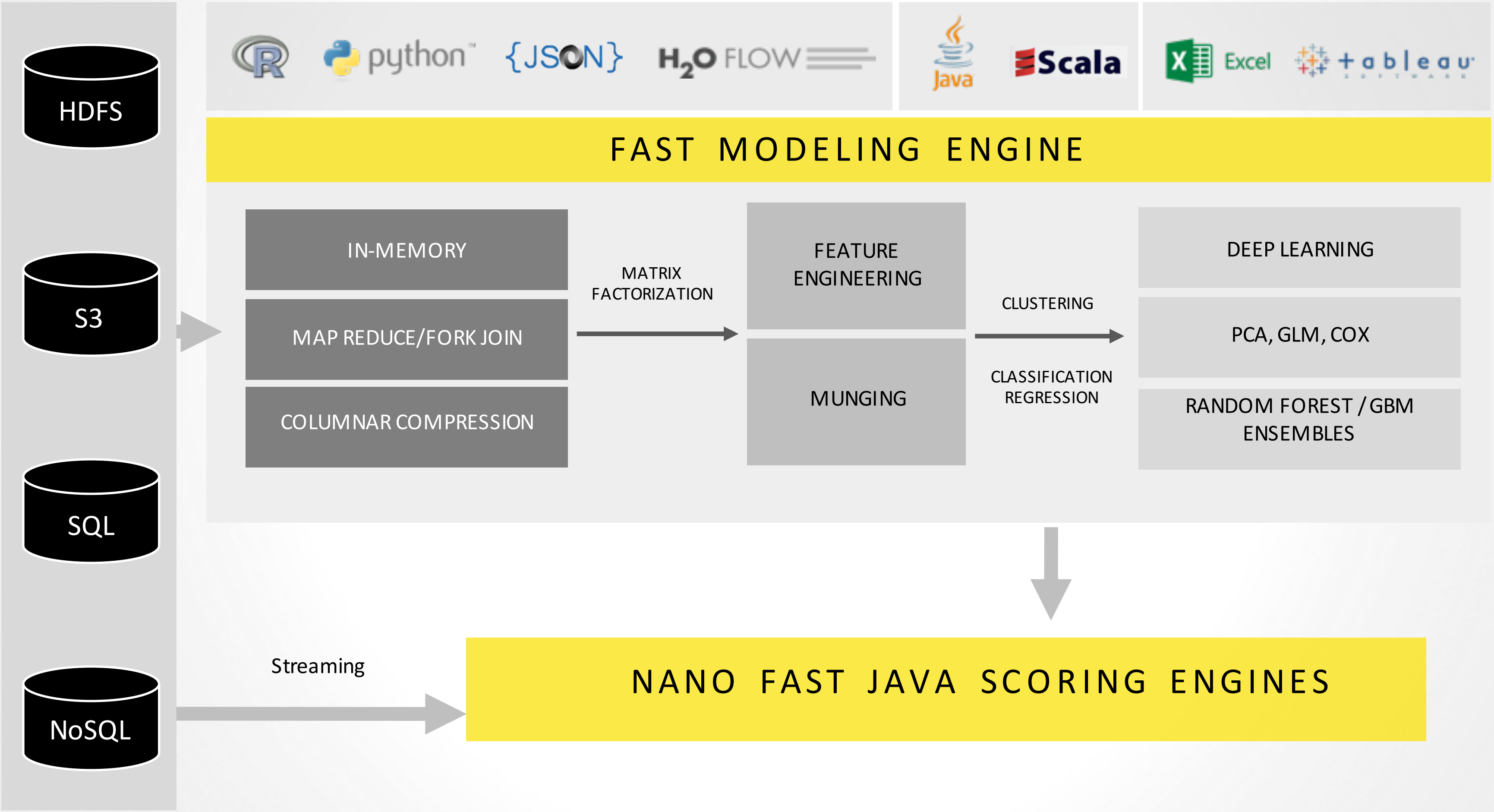
HDFS

S3

SQL

NoSQL

ACCURACY WITH SPEED AND SCALE



HDFS

S3

SQL

NoSQL

R python™ {JSON} H2O FLOW

Java Scala

Excel tableau

FAST MODELING ENGINE

IN-MEMORY

MAP REDUCE/FORK JOIN

COLUMNAR COMPRESSION

MATRIX FACTORIZATION

FEATURE ENGINEERING

MUNGING

CLUSTERING

CLASSIFICATION REGRESSION

DEEP LEARNING

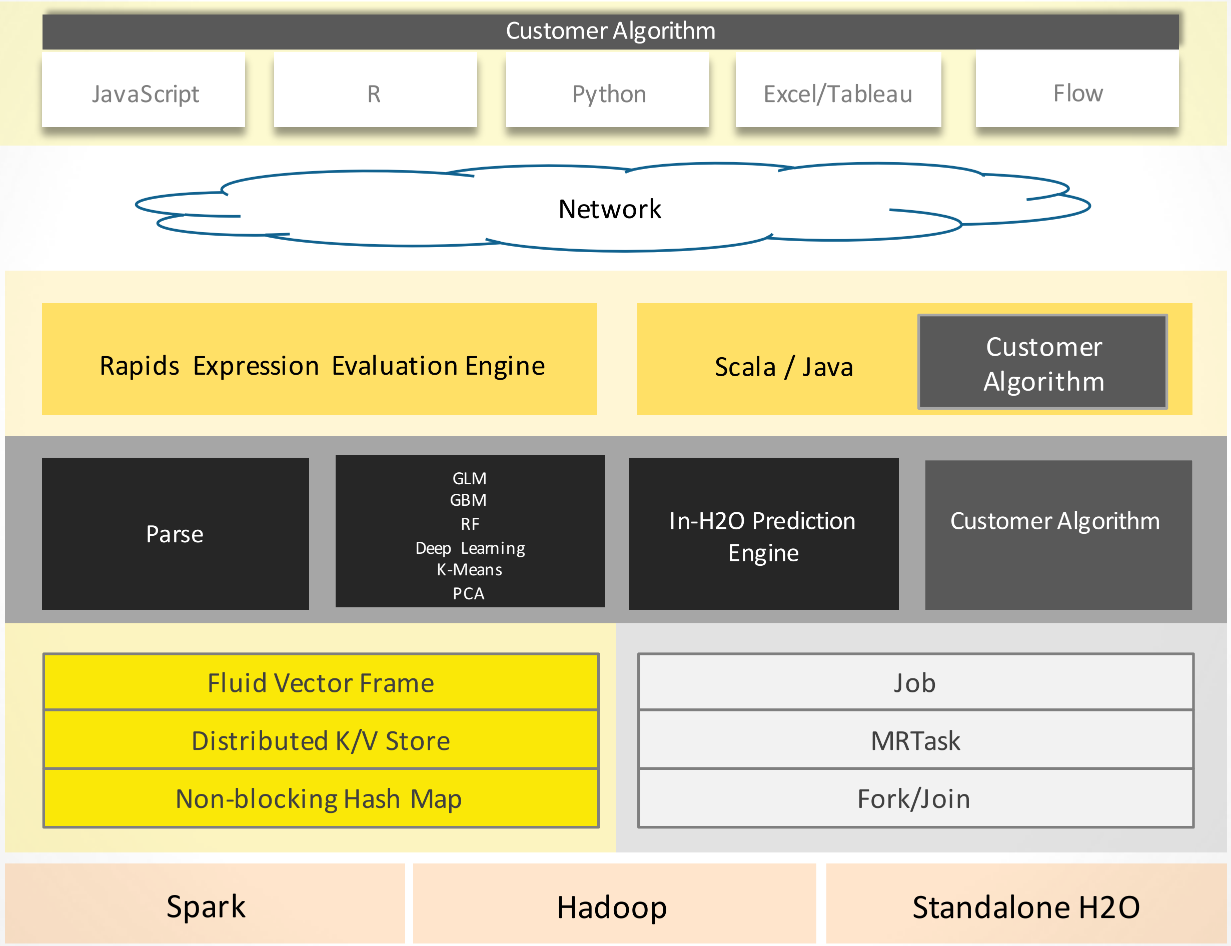
PCA, GLM, COX

RANDOM FOREST / GBM ENSEMBLES

Streaming

NANO FAST JAVA SCORING ENGINES

H2O SOFTWARE STACK



Hadoop (and YARN)

- You can launch H2O directly on Hadoop:

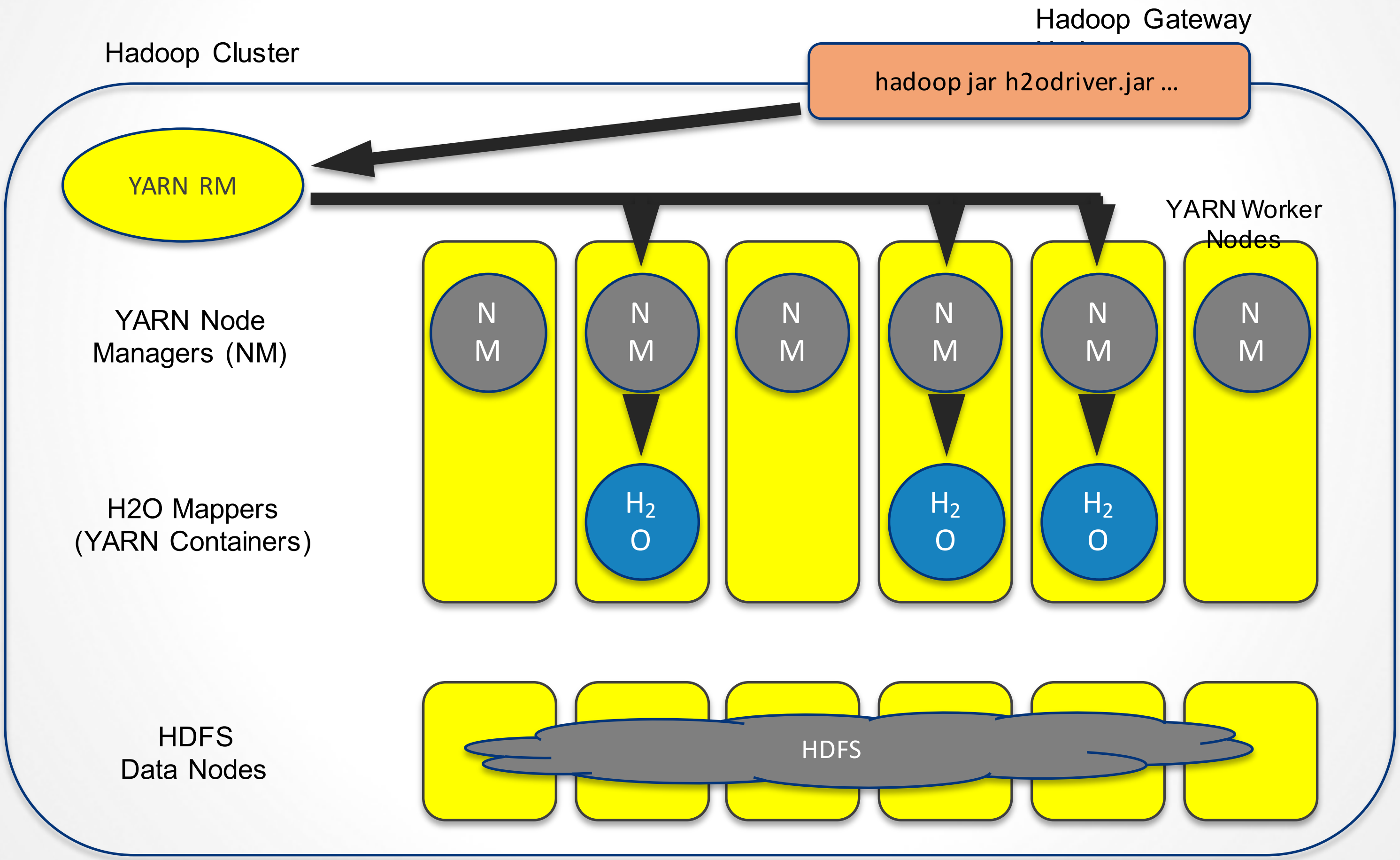
  ```
  $ hadoop jar h2odriver.jar … –nodes 3 –mapperXmx 50g
  ```

- H2O uses Hadoop MapReduce to get CPU and Memory on the cluster, not to manage work
  - H2O mappers stay at 0% progress forever
    - Until you shut down the H2O job yourself
  - All mappers (3 in this case) must be running at the same time
  - The mappers communicate with each other
    - Form an H2O cluster on-the-spot within your Hadoop environment
  - No Hadoop reducers(!)

- Special YARN memory settings for large mappers
  - yarn.nodemanager.resource.memory-mb
  - yarn.scheduler.maximum-allocation-mb

- CPU resources controlled via –nthreads h2o command line argument

H2O on YARN Deployment



Hadoop Gateway

```
hadoop jar h2odriver.jar …
```

Hadoop Cluster

YARN RM

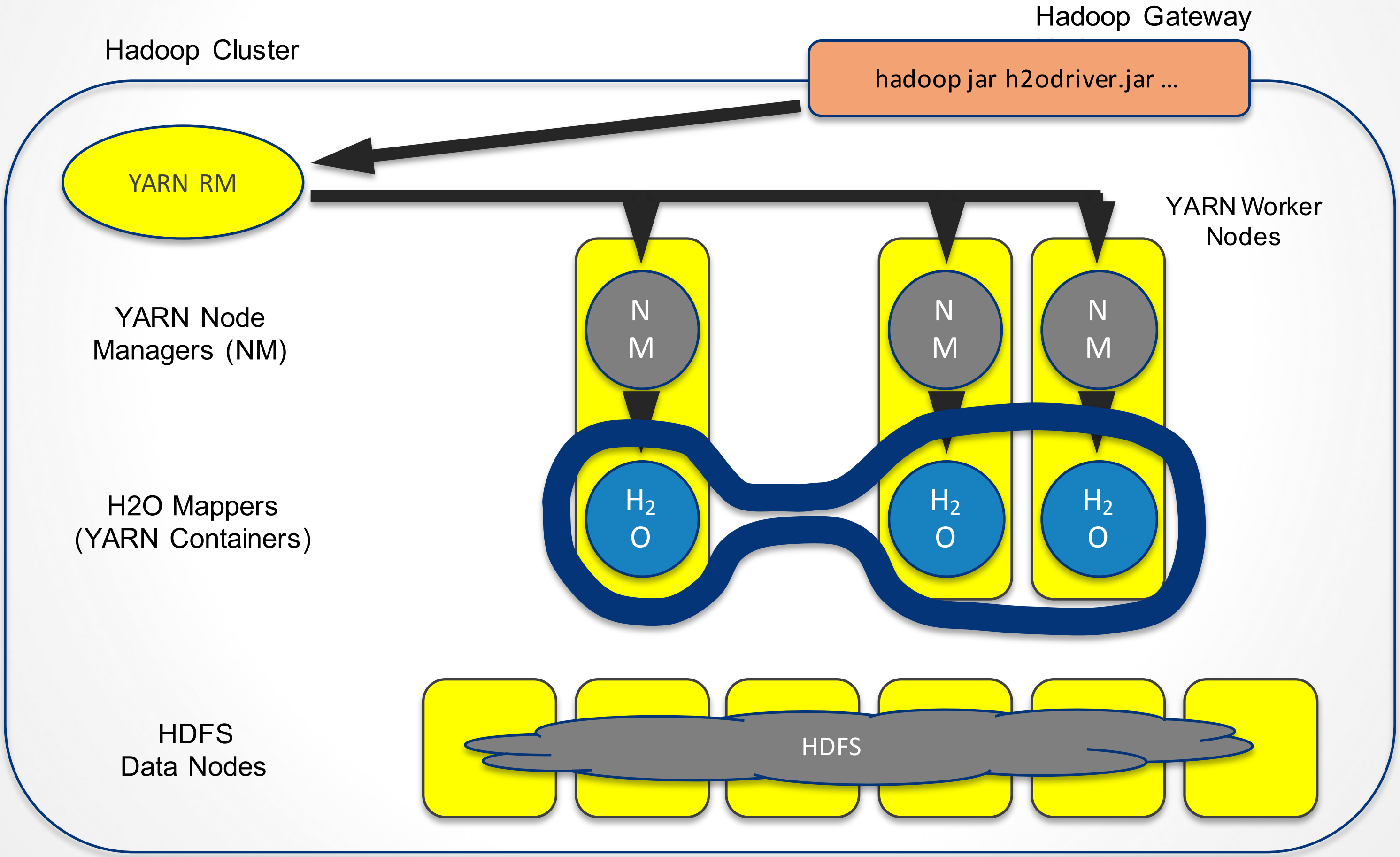
YARN Worker Nodes

YARN Node Managers (NM)

H2O Mappers (YARN Containers)

HDFS Data Nodes

Now You Have an H2O Cluster



Hadoop Gateway

Hadoop Cluster

hadoop jar h2odriver.jar ...

YARN RM

YARN Worker Nodes

YARN Node Managers (NM)

N
M

N
M

N
M

H2O Mappers (YARN Containers)

H_2
O

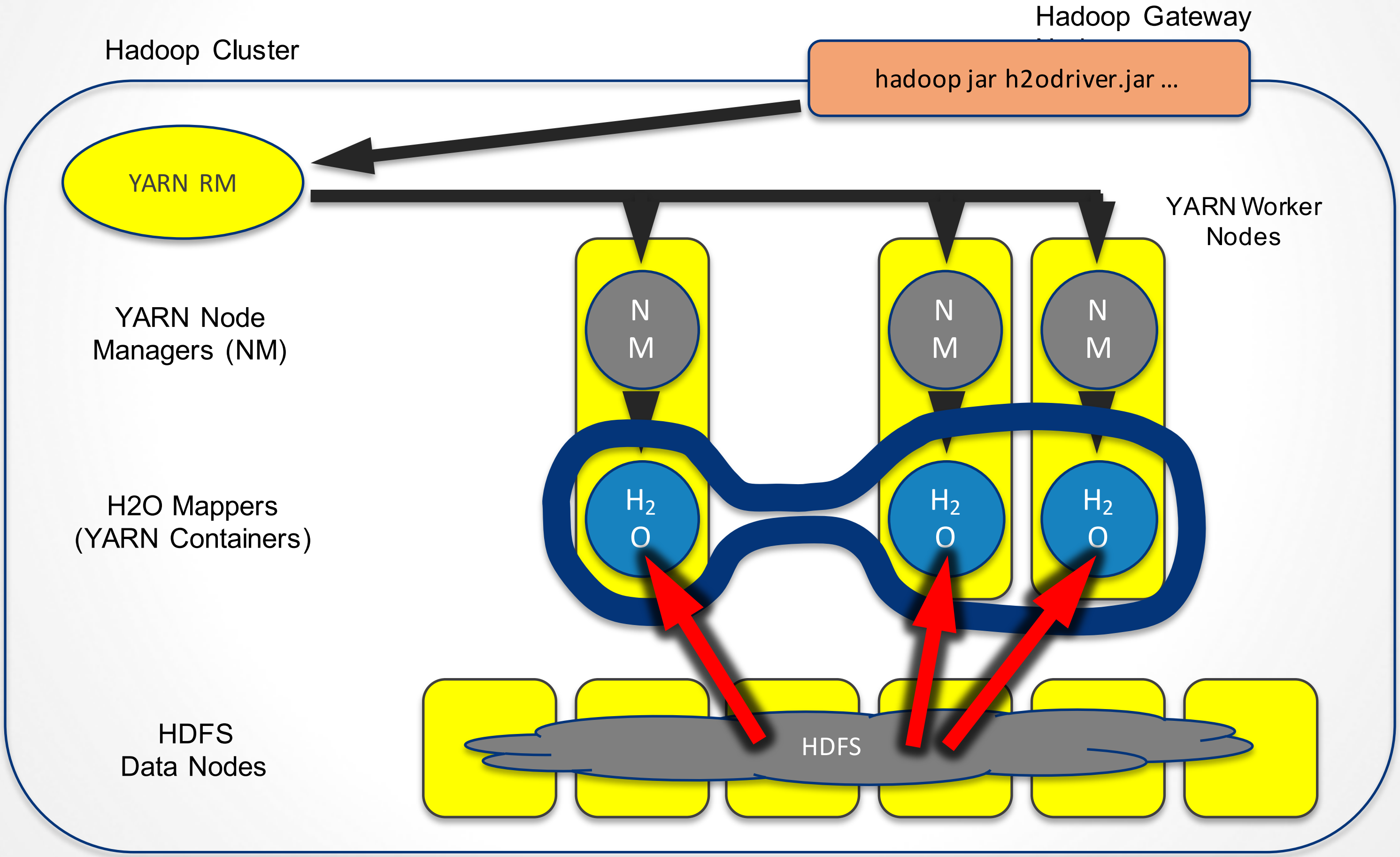
H_2
O

H_2
O

HDFS Data Nodes

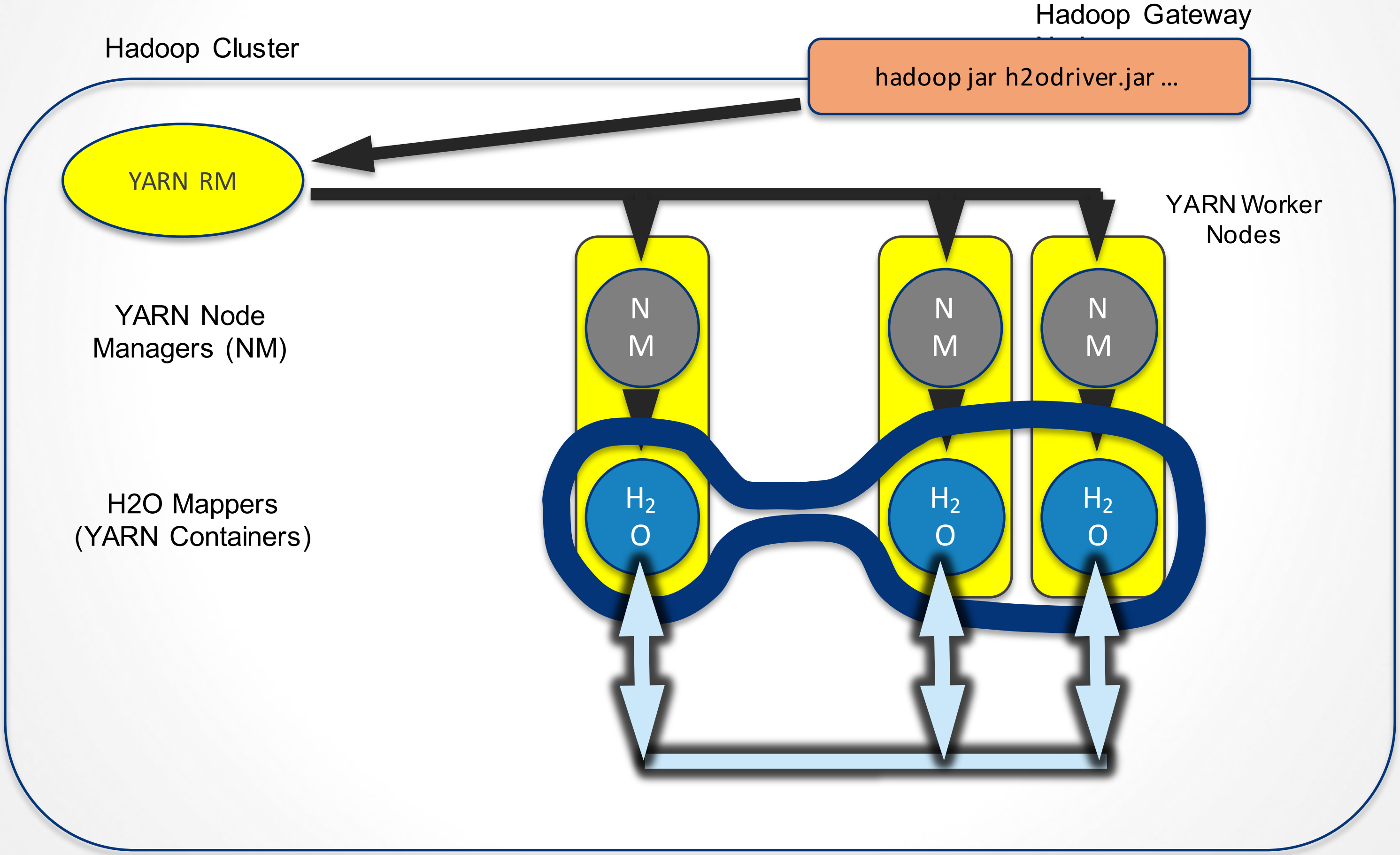
HDFS

Read Data from HDFS *Once*



```
hadoop jar h2odriver.jar ...
```

Build Models *in-Memory*



Hadoop Gateway

hadoop jar h2odriver.jar …

Hadoop Cluster

YARN RM

YARN Worker Nodes

YARN Node Managers (NM)

H2O Mappers (YARN Containers)

NM

NM

NM

H_2O

H_2O

H_2O

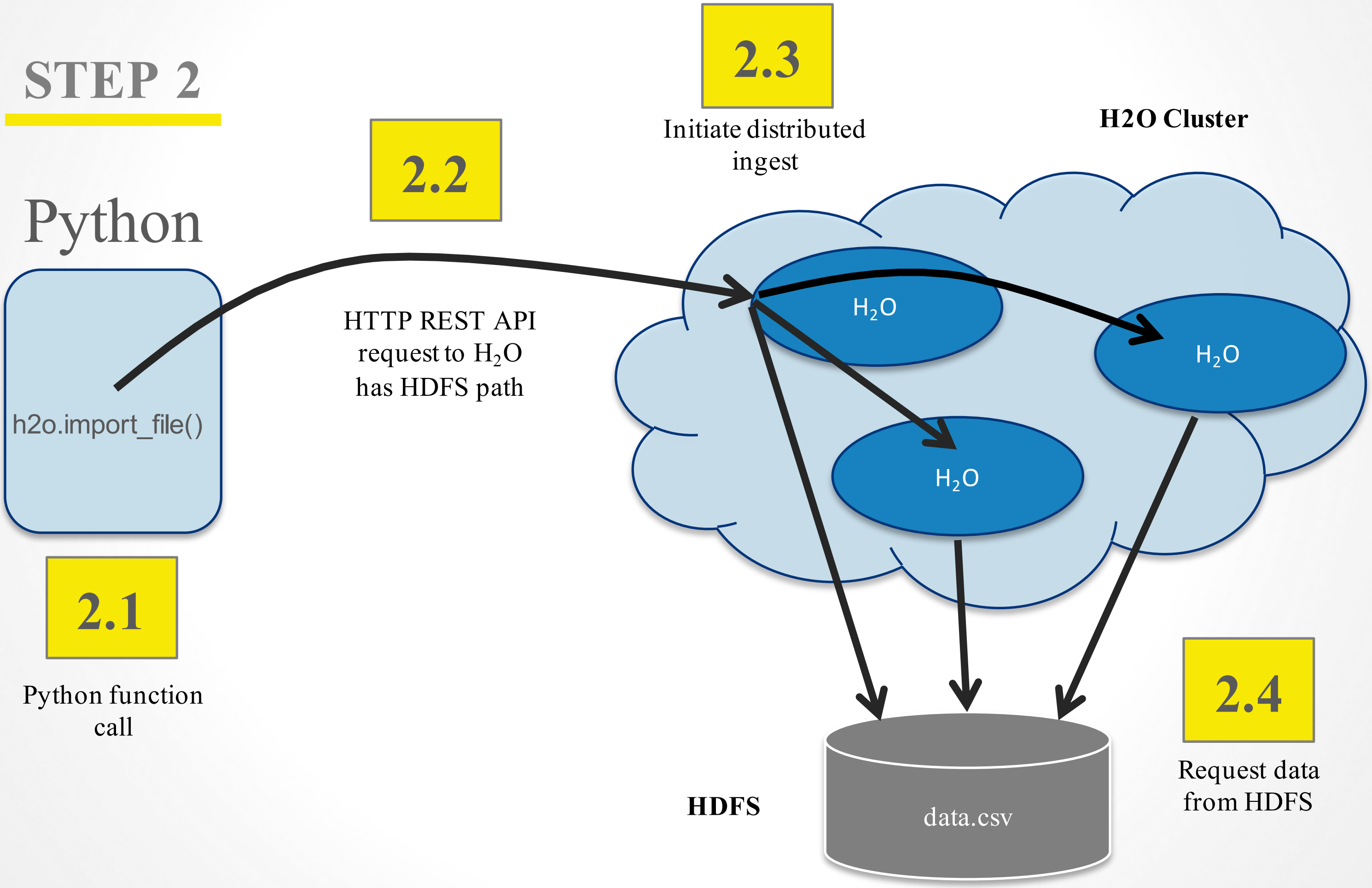
PYTHON AND R OBJECTS ARE PROXIES FOR BIG DATA

STEP 1

Python user

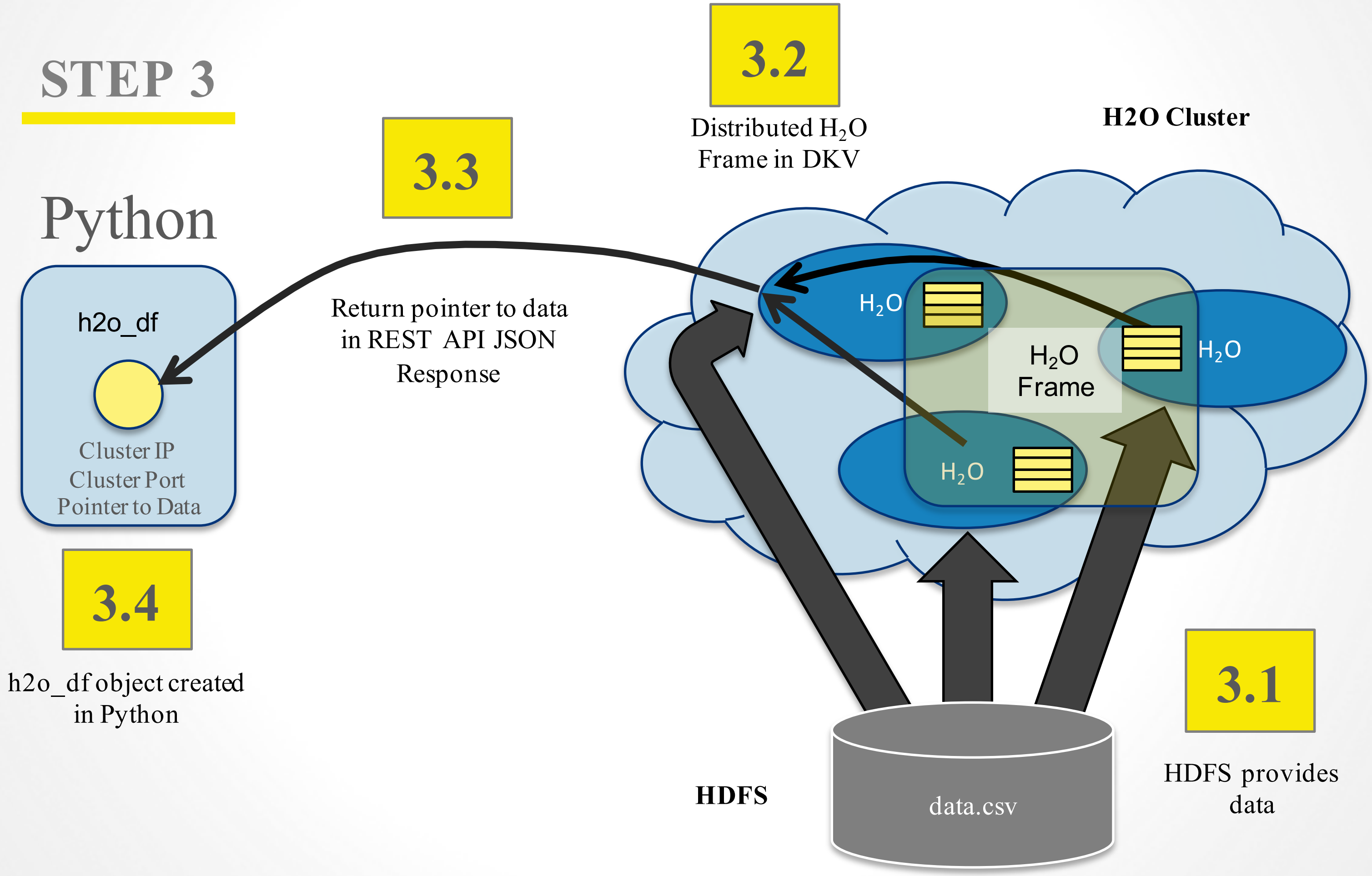
```
h2o_df = h2o.import_file(“hdfs://path/to/data.csv”)
```

PYTHON AND R OBJECTS ARE PROXIES FOR BIG DATA



PYTHON AND R OBJECTS ARE PROXIES FOR BIG DATA

STEP 3



3.1

HDFS provides data

3.2

Distributed H_2O Frame in DKV

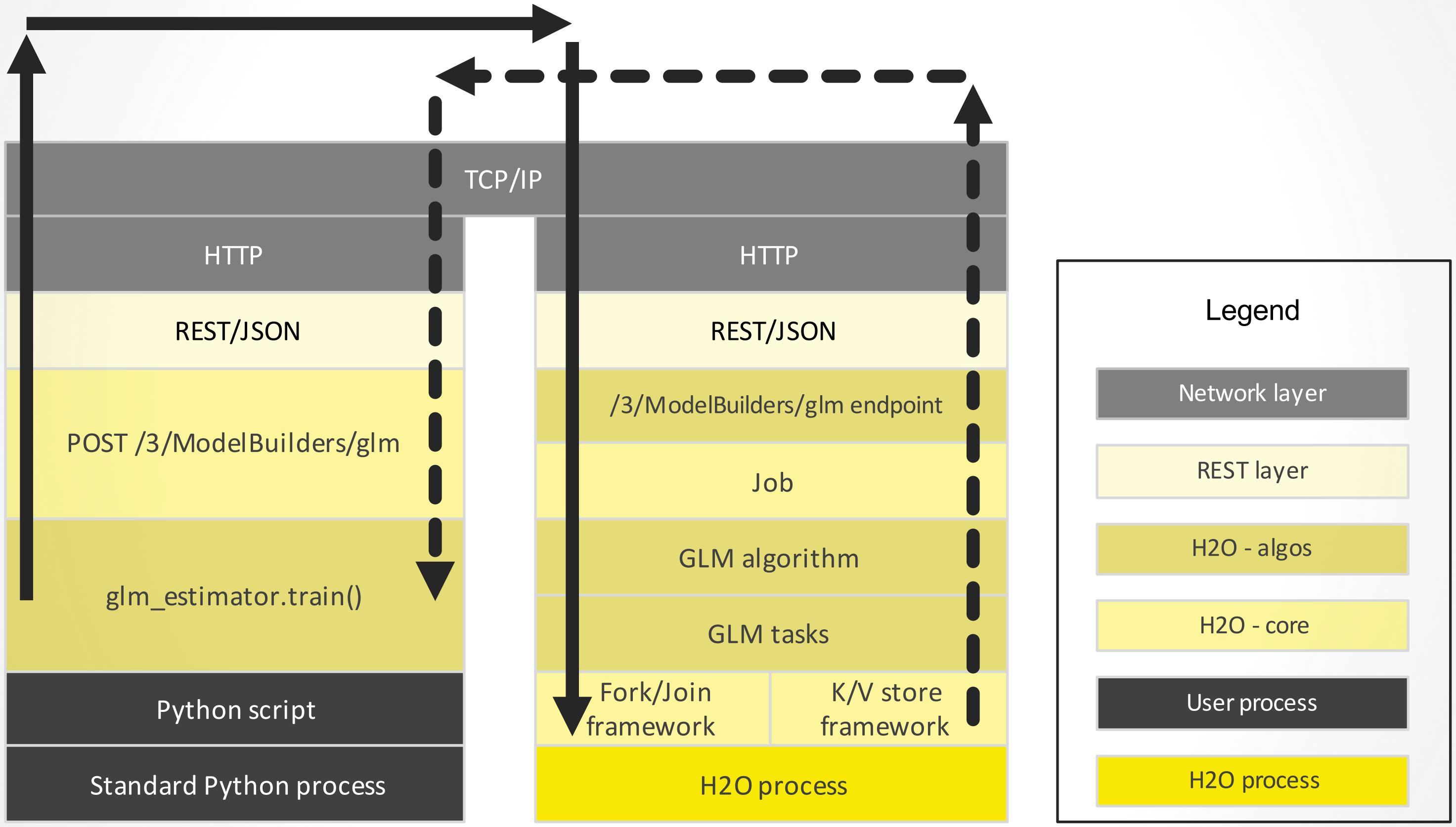
3.3

Return pointer to data in REST API JSON Response

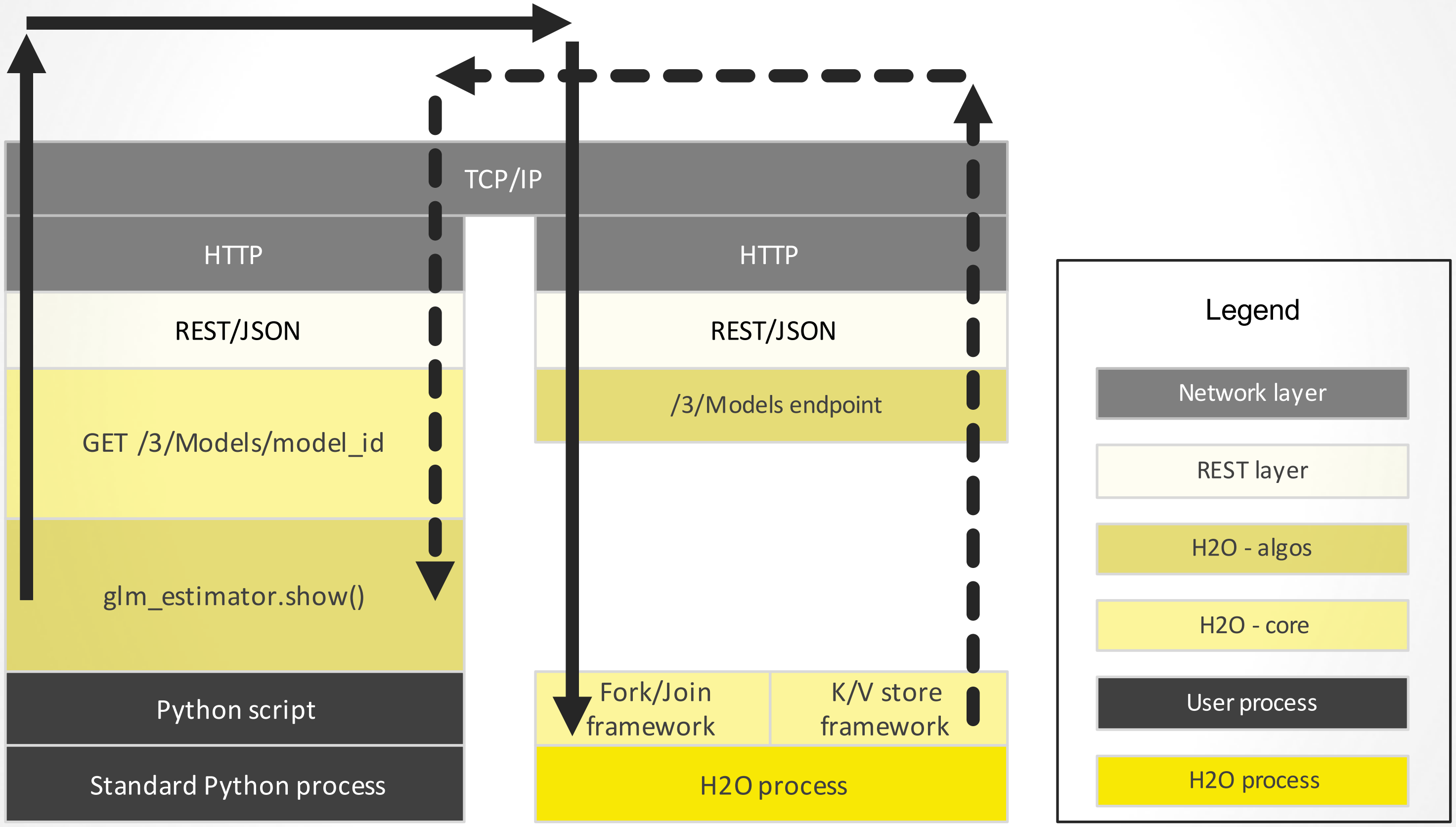
3.4

h2o_df object created in Python

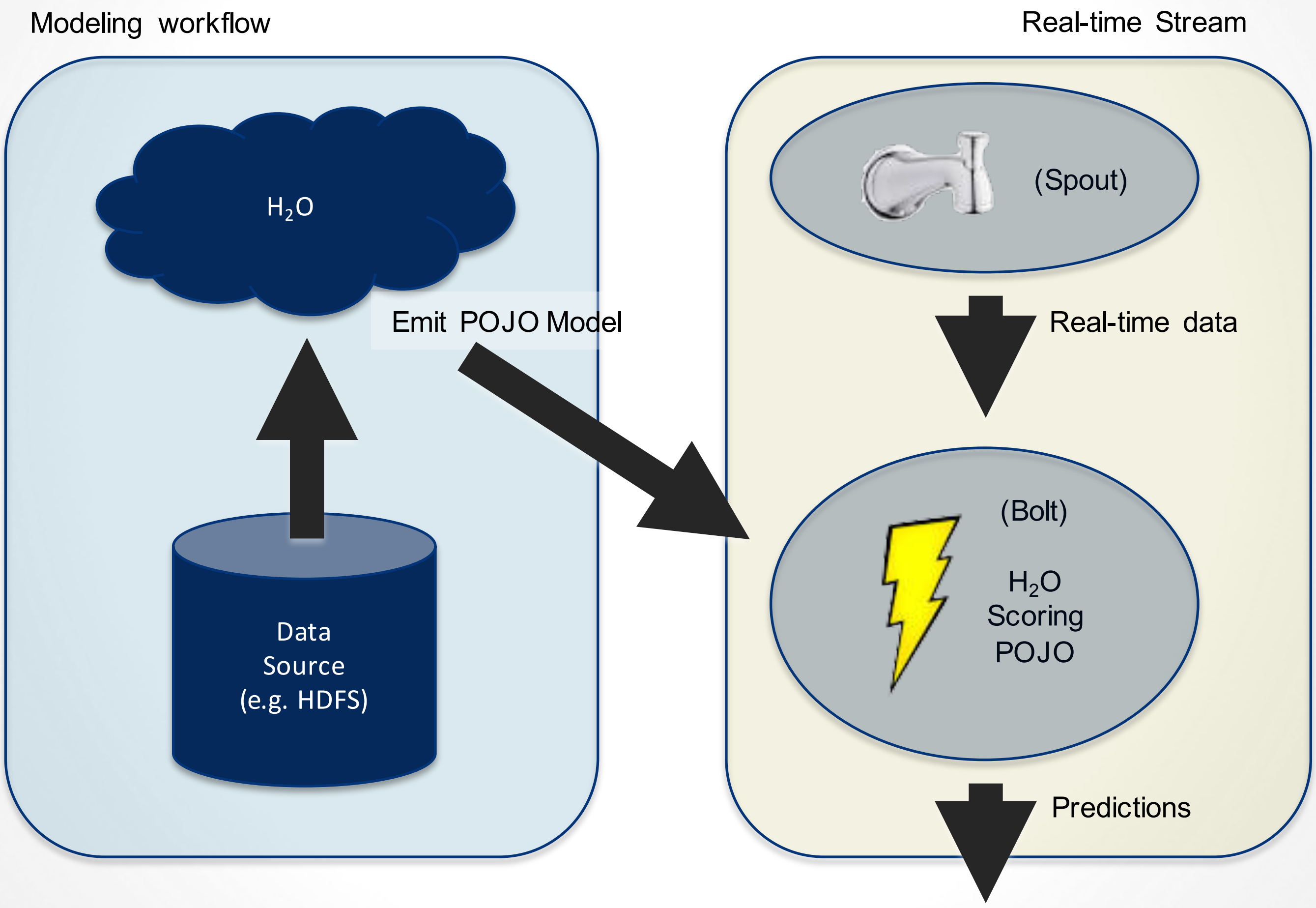
PYTHON SCRIPT STARTING H2O GLM



PYTHON SCRIPT STARTING H2O GLM

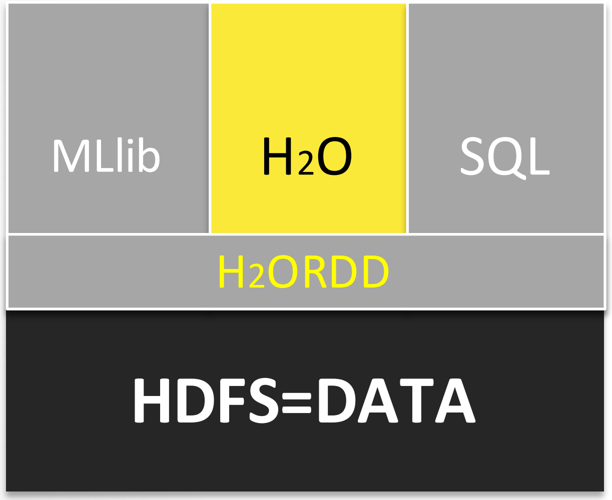


H_2O on Storm


Modeling workflow
H_2O
Emit POJO Model
Data
Source
(e.g. HDFS)
Real-time Stream
(Spout)
Real-time data
(Bolt)
H_2O
Scoring
POJO
Predictions

H_2O – The Killer-App for Spark

H2O Sparkling Water



MLlib

H_2O

SQL

H_2ORDD

HDFS=DATA

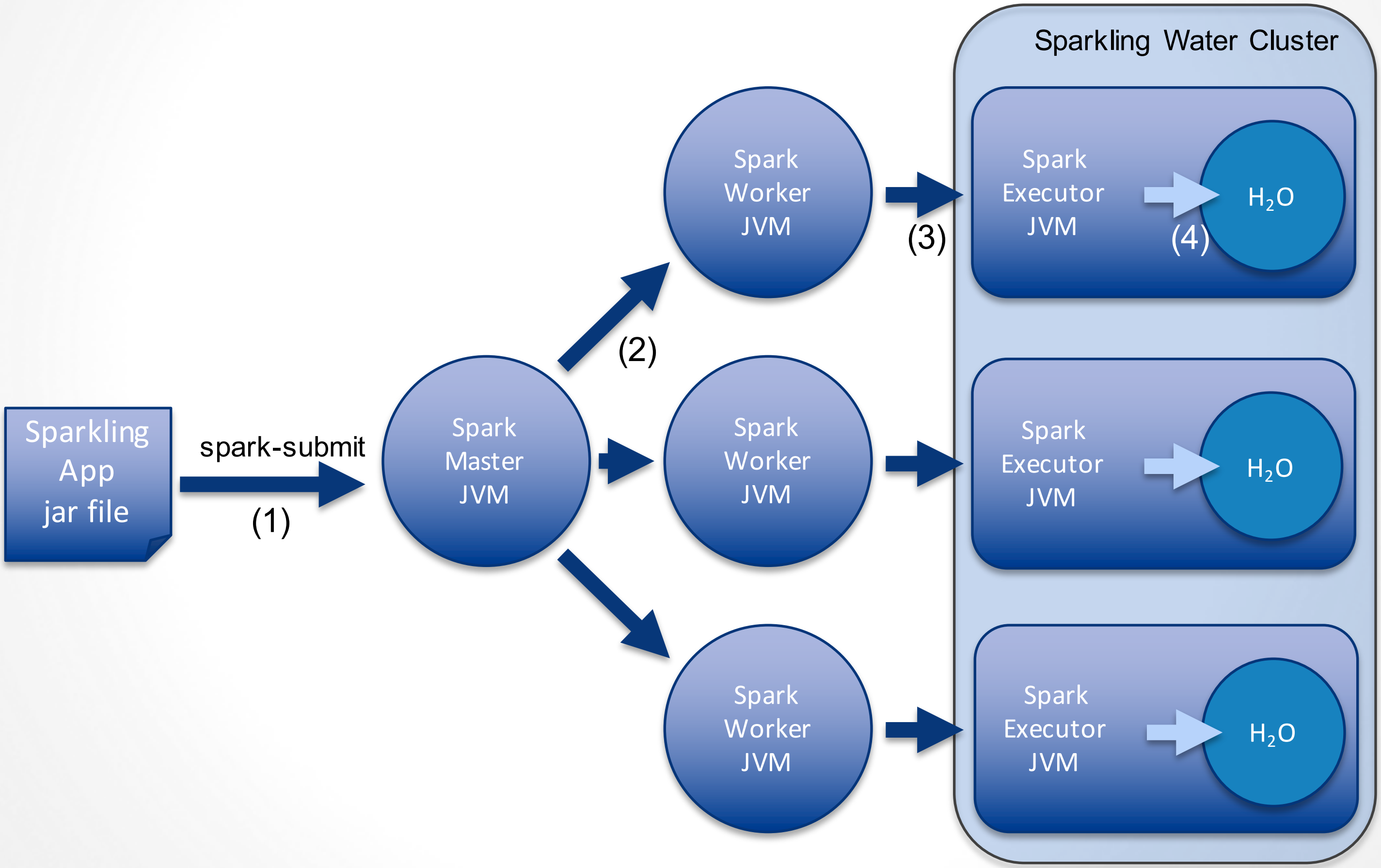
In-Memory	Big Data, Columnar
ML	100x faster Algos
R	CRAN, API, fast engine
API	Spark API, Java MM
Community	Devs, Data Science

H_2O.ai
Machine Intelligence

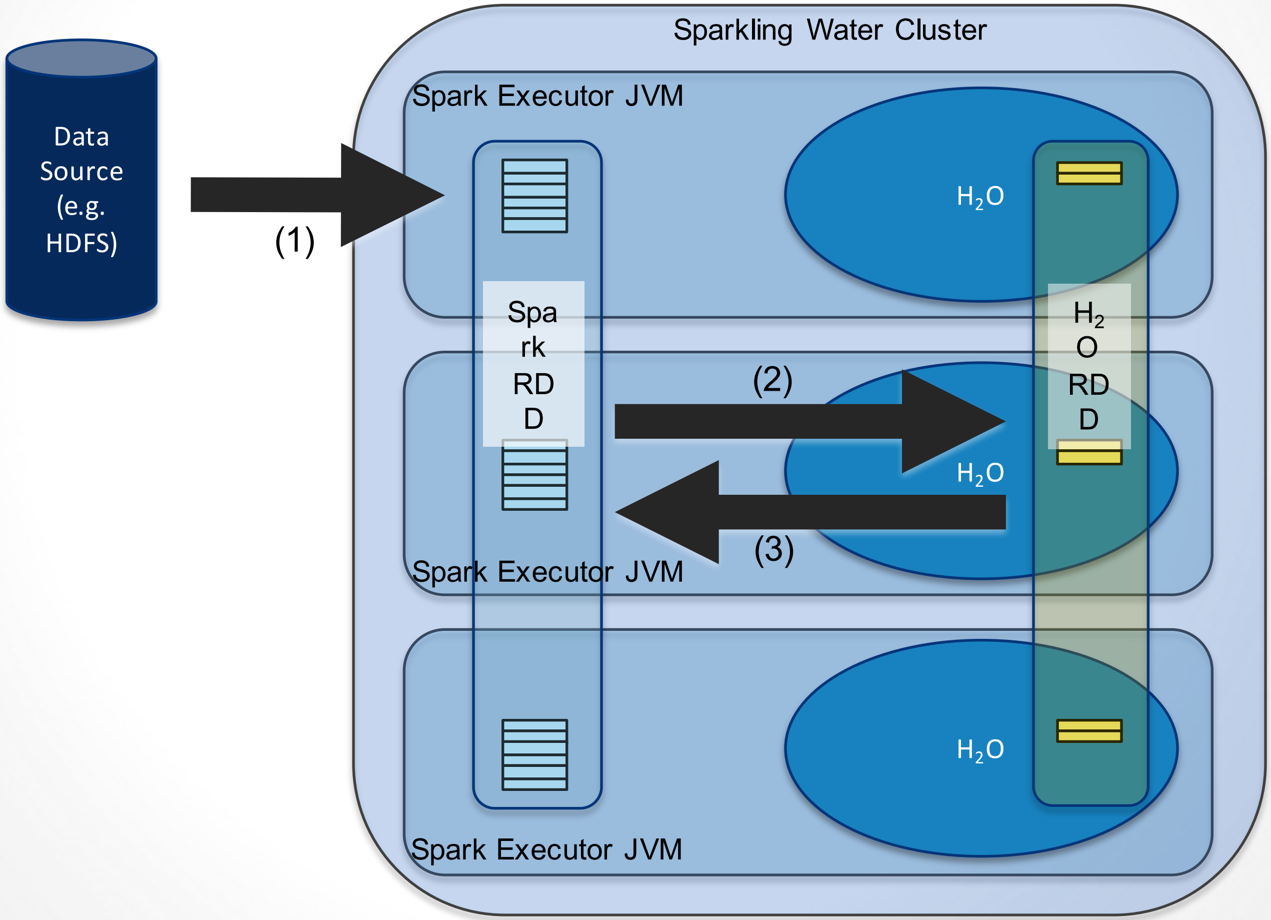
Spark (and Sparkling Water)

- H2O runs as an application on a Spark cluster using spark-submit
  - Standard Spark 1.3+
  - Includes H2O on Spark on YARN
- H2O and Spark nodes share a JVM process
- H2ORDD facilitates easy data sharing between Spark (e.g. Spark SQL, MLlib) and H2O (e.g. Deep Learning)
- Scala & PySpark support

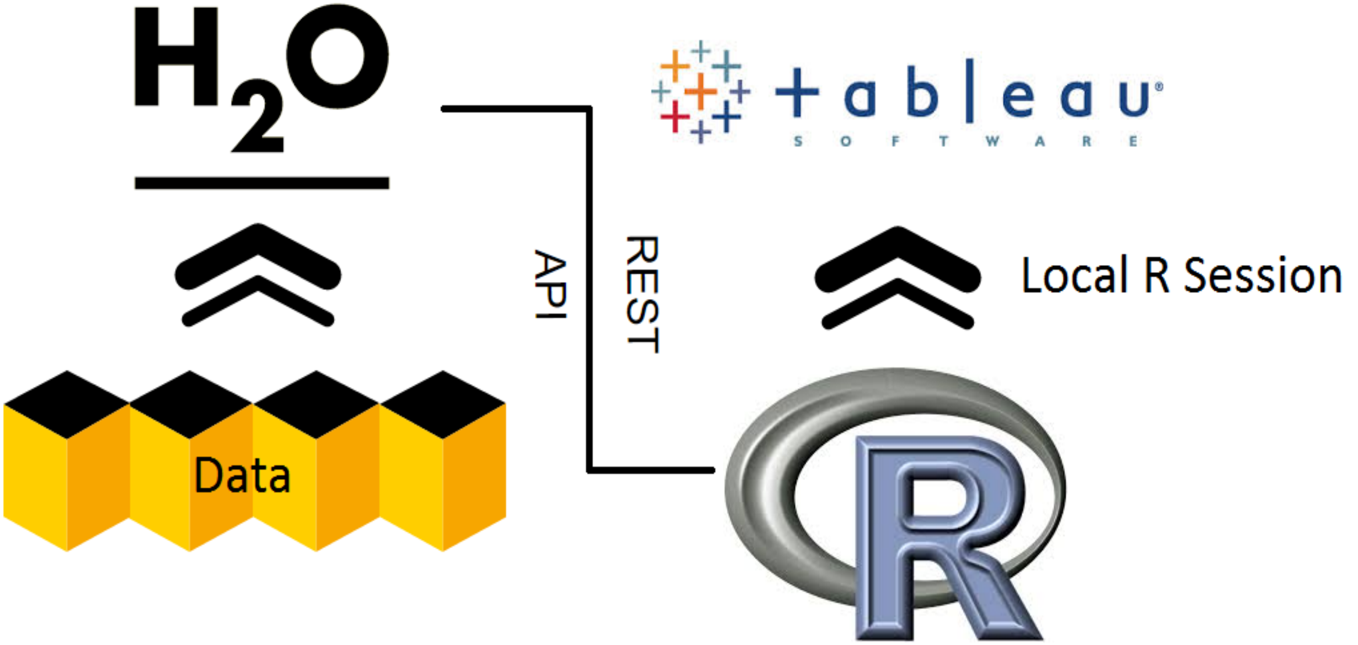
Sparkling Water Application Life Cycle



Sparkling Water Data Distribution



OPEN FOR INTEGRATION



H2O.ai
Machine Intelligence

RESOURCES

- Download and go: http://www.h2o.ai/download
- Documentation: http://docs.h2o.ai/
- Booklets, Datasheet: http://www.h2o.ai/resources/
- Github: http://github.com/h2oai/
- Training: http://learn.h2o.ai/

THANK YOU