WAYS TO PRODUCTIONIZE H2O

Tom Kraljevic

tomk@h2o.ai

H2O Open Tour 2016, New York City

July 19, 2016

WHERE TO FIND THE SLIDES

They’re in GitHub…

h2oai/h2o-meetups/tree/master/
2016_07_19_H2O_Open_Tour_NYC_Prod

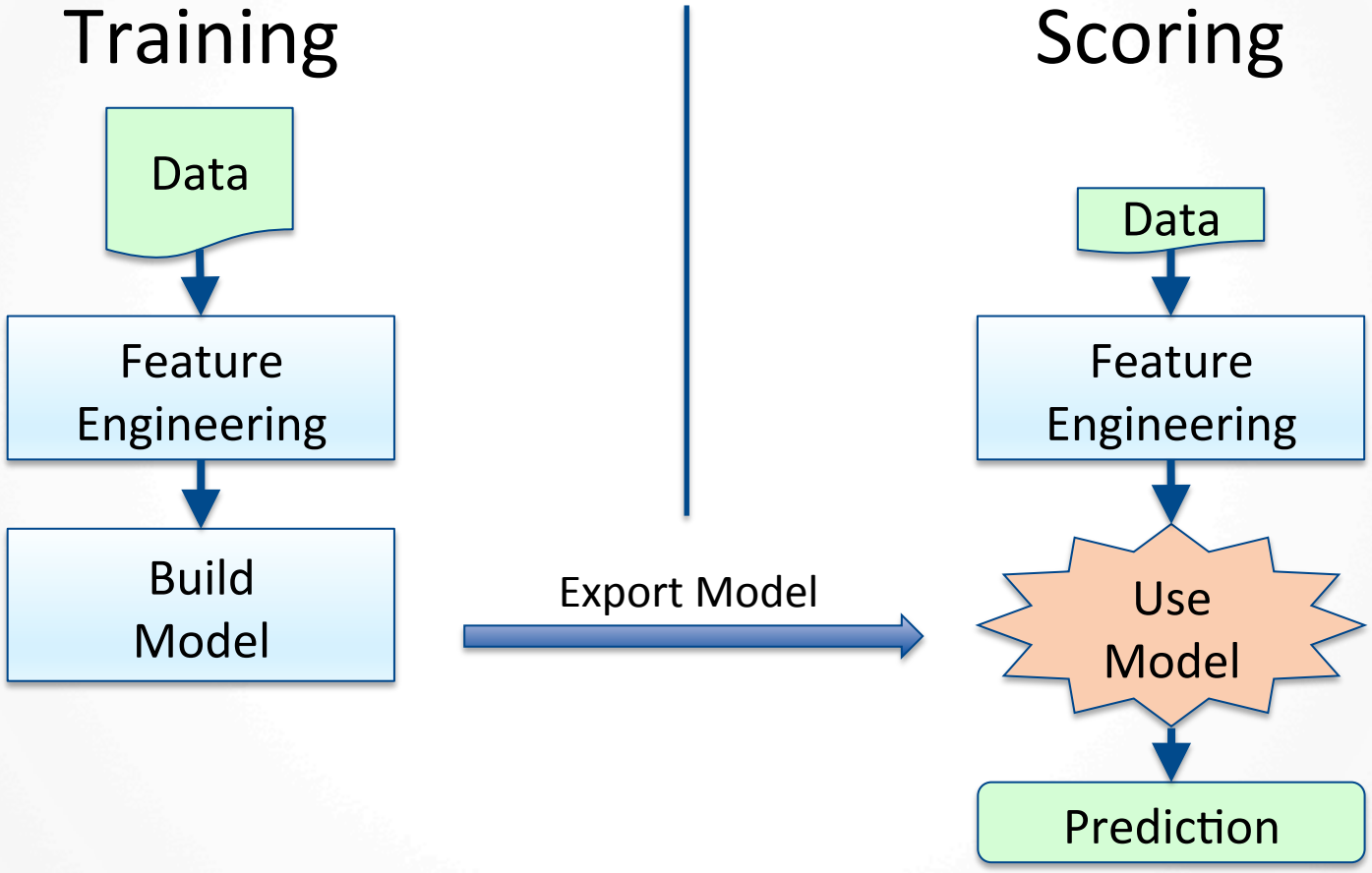
DEFINING "IN PRODUCTION"?

- Removal would impact value to the business
  - Dollars
  - Time

OUTLINE

- Motivation
- Productionizing Model Building
  - Training
- Productionizing Model Use
  - Scoring aka Predicting
- Q & A

TYPICAL DEPLOYMENT APPROACH



PRODUCTIONIZING MODEL BUILDING (TRAINING)

H2O TRAINING DIMENSIONS

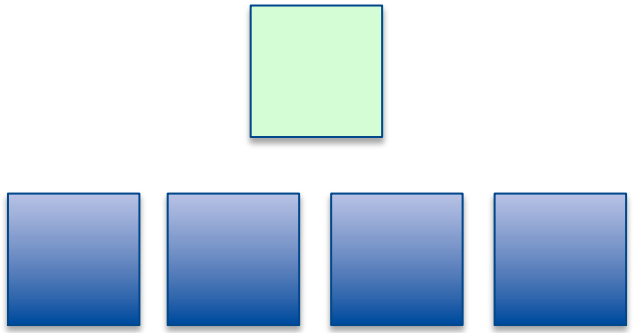
- Where does the data live?
  - Hadoop
    - HDFS, Hive
  - S3 / AWS
  - Database
  - Local drive (laptop, desktop, server)
- How big is the data?
  - Fits in memory of one host? (Or not)
  - "Too small"?
- How often should models be re-trained?
  - Daily? Weekly? Monthly? Quarterly?
- Where will model training happen?
  - Hadoop YARN job
  - EC2 instances
  - Bare metal cluster
  - A really big server
  - Laptop
- Resource requirements (how many nodes?)
  - CPUs
  - Memory
  - Time to build a model

H2O TRAINING SIZING BEST PRACTICE

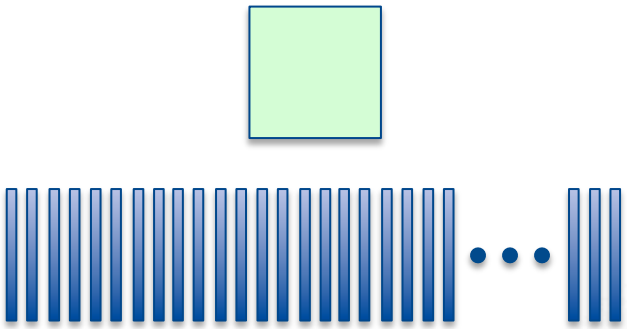
- For a novice getting started with a decent sized dataset:
  - Size total cluster memory to 4x on-disk data size (data for model training, not “total data”)
  - Use nodes of at least 10 GB, preferably larger
  - Use a small number of big nodes
    - e.g. 8 nodes x 30 GB each
    - For a YARN environment, you will have to adjust default settings to start big containers
  - ***Definitely DO NOT use 200 nodes x 1 GB each***
    - Caution: Hadoop MapReduce veterans often start this way!
    - 1 GB is way too small for *any* workload!

H2O TRAINING SIZING BEST PRACTICE

Start
With This



For the
Experts

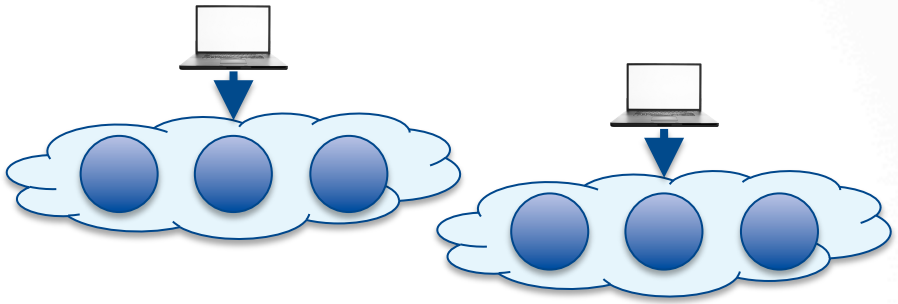


H2O USER PROVISIONING BEST PRACTICE

- Individual H2O jobs (aka application instances, aka clusters) are not multi-tenant
  - Think of YARN as providing multi-tenancy and H2O as a single job
    - H2O instance per interactive user, or per batch job
  - Steam product from H2O.ai simplifies management of H2O clusters

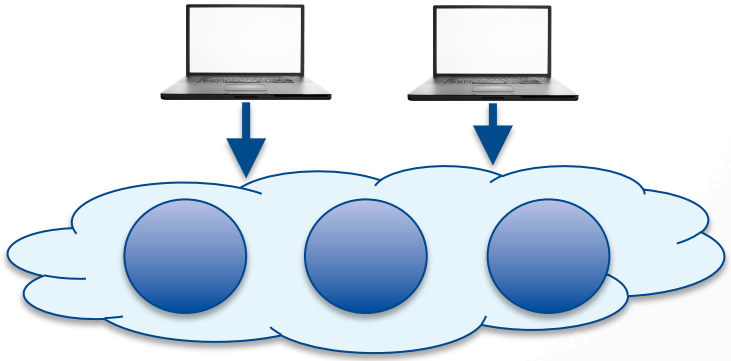
H2O USER PROVISIONING BEST PRACTICE

Recommended



1 User per
H2O Instance

Not
Recommended



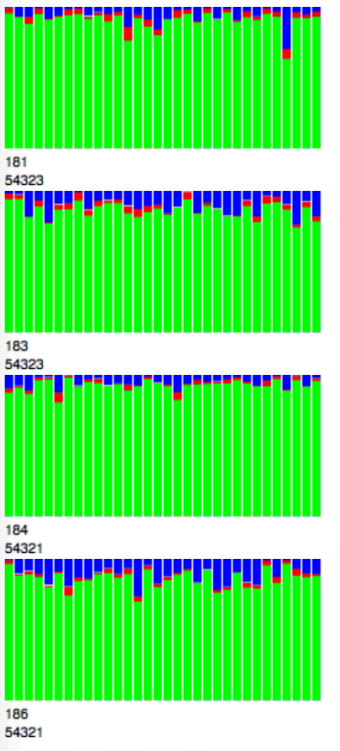
Shared Group
H2O Instance

H2O MONITORING BEST PRACTICE

- Monitor your H2O instance's resource utilization
  - Watch the Water Meter in the Flow Web UI to see:
    - If your hardware cluster is already being slammed before H2O even starts doing anything
    - If your H2O instance is using CPU when you expect it to
  - Look at H2O log files for detailed memory utilization information (advanced)

H2O MONITORING BEST PRACTICE

CPU Monitoring



Memory Monitoring (advanced)

yarn logs -applicationId application_1457562501251_2398

```
07-18 20:18:03.364 172.16.2.184:54321    31423  main      INFO: ----- H2O started  -----
07-18 20:18:03.430 172.16.2.184:54321    31423  main      INFO: Build git branch: rel-turchin
07-18 20:18:03.430 172.16.2.184:54321    31423  main      INFO: Build git hash: c09d9da94e7c61eb7ef4573f9f5029e0ca501469
07-18 20:18:03.430 172.16.2.184:54321    31423  main      INFO: Build git describe: jenkins-rel-turchin-3
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Build project version: 3.8.2.3
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Built by: 'jenkins'
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Built on: '2016-04-25 13:07:53'
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Java availableProcessors: 32
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Java heap totalMemory: 39.45 GB
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Java heap maxMemory: 39.45 GB
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Java version: Java 1.7.0_67 (from Oracle Corporation)
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: JVM launch parameters: [-XX:NewRatio=8, -Djava.net.preferIPv4Stack=true, -Dhdp.version=2.2.6.3-1, -Xms40g,
-Xmx40g, -XX:PermSize=256m, -verbose:gc, -XX:+PrintGCDetails, -XX:+PrintGCTimeStamps, -Dlog4j.defaultInitOverride=true,
-Djava.io.tmpdir=/opt2/hdp/yarn/local/usercache/tomk/appcache/application_1457562501251_2398/container_e13_1457562501251_2398_01_000005/tmp,
-Dlog4j.configuration=container-log4j.properties, -Dyarn.app.container.log.dir=/opt2/hdp/yarn/log/application_1457562501251_2398/
container_e13_1457562501251_2398_01_000005, -Dyarn.app.container.log.filesize=0, -Dhadoop.root.logger=INFO,CLA]
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: OS version: Linux 3.13.0-61-generic (amd64)
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: Machine physical memory: 251.89 GB
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: X-h2o-cluster-id: 1468898282070
07-18 20:18:03.431 172.16.2.184:54321    31423  main      INFO: User name: 'tomk'
07-18 20:18:03.432 172.16.2.184:54321    31423  main      INFO: Possible IP Address: br2 (br2), 172.16.2.184
07-18 20:18:03.432 172.16.2.184:54321    31423  main      INFO: Possible IP Address: lo (lo), 127.0.0.1
07-18 20:18:03.432 172.16.2.184:54321    31423  main      INFO: Internal communication uses port: 54322
07-18 20:18:03.432 172.16.2.184:54321    31423  main      INFO: Listening for HTTP and REST traffic on http://172.16.2.184:54321/
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO: Frame distribution summary:
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:                                Size    Number of Rows  Number of Chunks per Column  Number of Chunks
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO: 172.16.2.181:54323    918.6 MB   29811120.000000                  131.000000       4061.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO: 172.16.2.183:54323    883.7 MB   29023380.000000                  127.000000       3937.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO: 172.16.2.184:54321    919.5 MB   28959996.000000                  127.000000       3937.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO: 172.16.2.186:54321    925.5 MB   28900760.000000                  127.000000       3937.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:                mean    911.8 MB   29173814.000000                  128.000000       3968.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:                 min    883.7 MB   28900760.000000                  127.000000       3937.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:                 max    925.5 MB   29811120.000000                  131.000000       4061.000000
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:              stddev     16.5 MB     370495.062500                    1.732051         53.693577
07-18 20:29:58.542 172.16.2.184:54321    31423  FJ-1-15   INFO:               total      3.56 GB  116695256.000000                  512.000000      15872.000000
792.586: [GC [PSYoungGen: 2353508K->1553406K(3107328K)] 6945343K->6408429K(40390144K), 0.1256170 secs] [Times: user=2.19 sys=0.53, real=0.12 secs]
793.632: [GC [PSYoungGen: 3107117K->1553392K(3107328K)] 7962139K->7350065K(40390144K), 0.1929720 secs] [Times: user=2.61 sys=1.67, real=0.20 secs]
794.524: [GC [PSYoungGen: 3106455K->1221115K(3107328K)] 8903127K->8571453K(40390144K), 0.1658980 secs] [Times: user=2.92 sys=0.77, real=0.17 secs]
795.501: [GC [PSYoungGen: 2774868K->1496653K(3107328K)] 10125206K->9965740K(40390144K), 0.1611300 secs] [Times: user=2.81 sys=0.82, real=0.16 secs]
796.684: [GC [PSYoungGen: 3050216K->986551K(3107328K)] 11519302K->10546313K(40390144K), 0.1470070 secs] [Times: user=2.03 sys=1.27, real=0.15 secs]
797.983: [GC [PSYoungGen: 2540021K->781681K(3107328K)] 12099782K->11006879K(40390144K), 0.1154810 secs] [Times: user=1.38 sys=1.21, real=0.12 secs]
```

ADDITONAL YARN BEST PRACTICES

- Limiting memory occurs “naturally”
- No CPU usage limit by default, but you can use:
  - YARN cgroups
  - H2O –nthreads startup option
- Use node labels to direct container placement
- Capacity and Fair schedulers are both fine
- Many people provision YARN queues to manage resource use for teams
- You ***must*** disable preemption for H2O jobs

PRODUCTIONIZING MODEL USE (SCORING AKA PREDICTING)

HIGH-LEVEL H2O DIMENSIONS

In-H2O

- Batch
- Many rows at once
  - think Frames of data
- R / Python / Scala
  - script driving H2O
- Script

POJO

- Real-time
- One row at a time
- Java / Scala
  - embedded or as a service
- Application

OTHER INDEPENDENT DIMENSIONS

Choice A

- On-premise
- Data Science
- Micro-service with REST API
- Model only

- Train and predict together
- Pre-calculated

Choice B

- Cloud
- Computer Science
- Embedded
- Model +
  Feature engineering
- Train and predict separately
- Calculate live

DATA ENVIRONMENTS

- Hive
- Databases
  - Oracle, MS SQL, SQLite, etc.
- Cloud databases (Amazon RDS)
- Teradata
- Bare flat files
- Hadoop HDFS
- Amazon S3
- Storm spout
- Spark stream
- Apache Apex

MODEL ENVIRONMENTS

- Laptop / Desktop
- Hadoop
- Standalone cluster
- Steam
- Cloud (e.g. EC2)
- Cloud endpoint (e.g. AWS Lambda)
- Storm bolt
- Spark stream
- Apache Apex
- Hive UDF
- POJO called from R

KEY PARAMETERS

- Model training
  - Frequency
  - Cost (memory, CPU, time)
- Prediction SLA
  - Throughput
  - Latency
  - Availability
- Security

EXAMPLE DESIGN PATTERNS

In-H2O

- Driven from Flow/R/ Python/Scala
  - Often launching on Hadoop
- Embedding H2O (droplet)

POJO

- JUnit test
- Craigslist app (Sparkling Water)
- App-consumer-loan (Jetty servlet)
- App-malicious-domain (AWS lambda)
- POJO from R (jar file)
- Journals.ai desktop app (SQlite DB)
- Storm bolt
- Hive UDF
- Steam from H2O.ai (REST API)

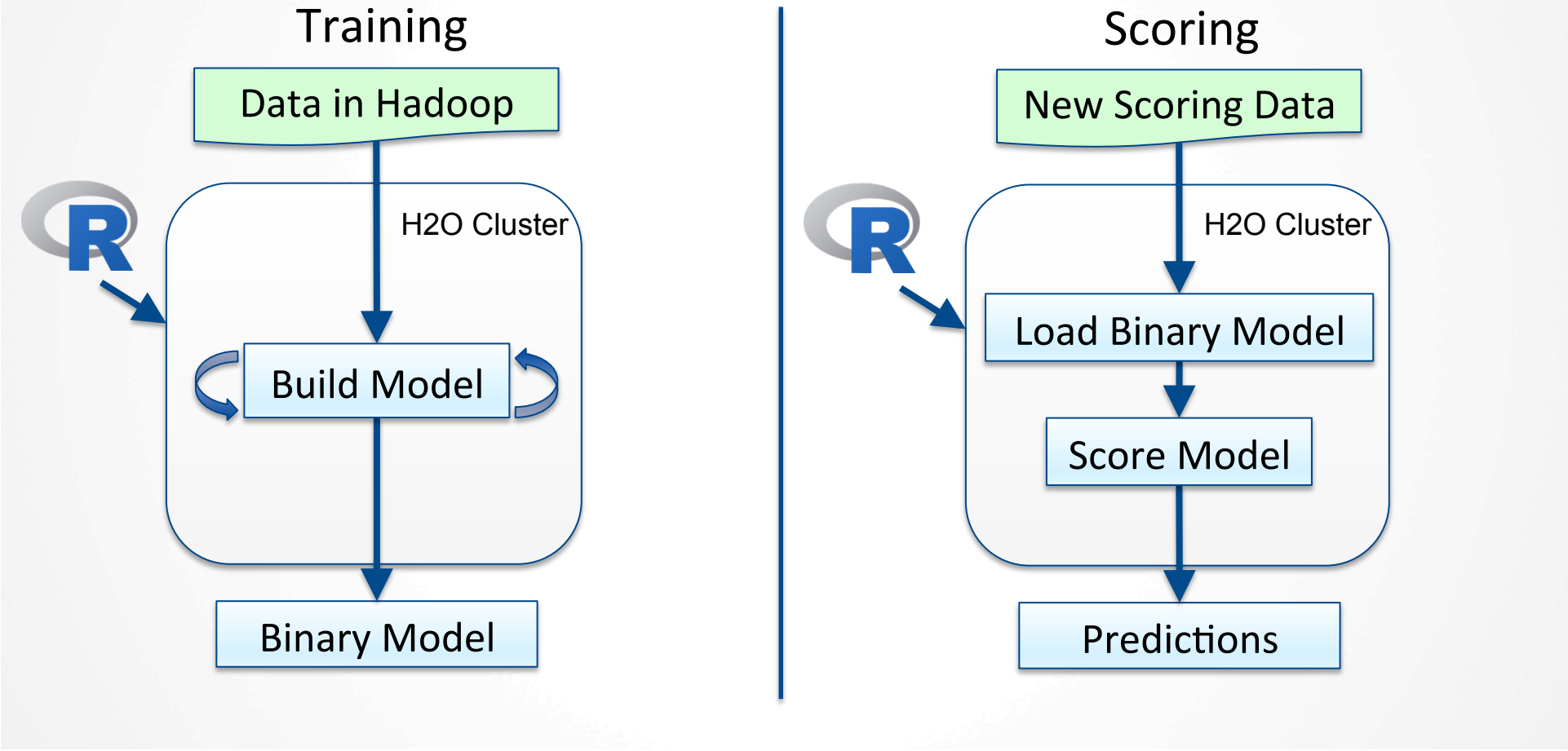
WALKTHROUGH FOR 5 DESIGN PATTERNS

- Batch In-H2O
- Steam
- AWS Lambda
- Hive UDF
- Pre-calculated DB

PURE BATCH H2O DESIGN PATTERN

- Training and scoring data lives in data lake
- Entire process driven by R
- Training occurs periodically
- Scoring occurs frequently

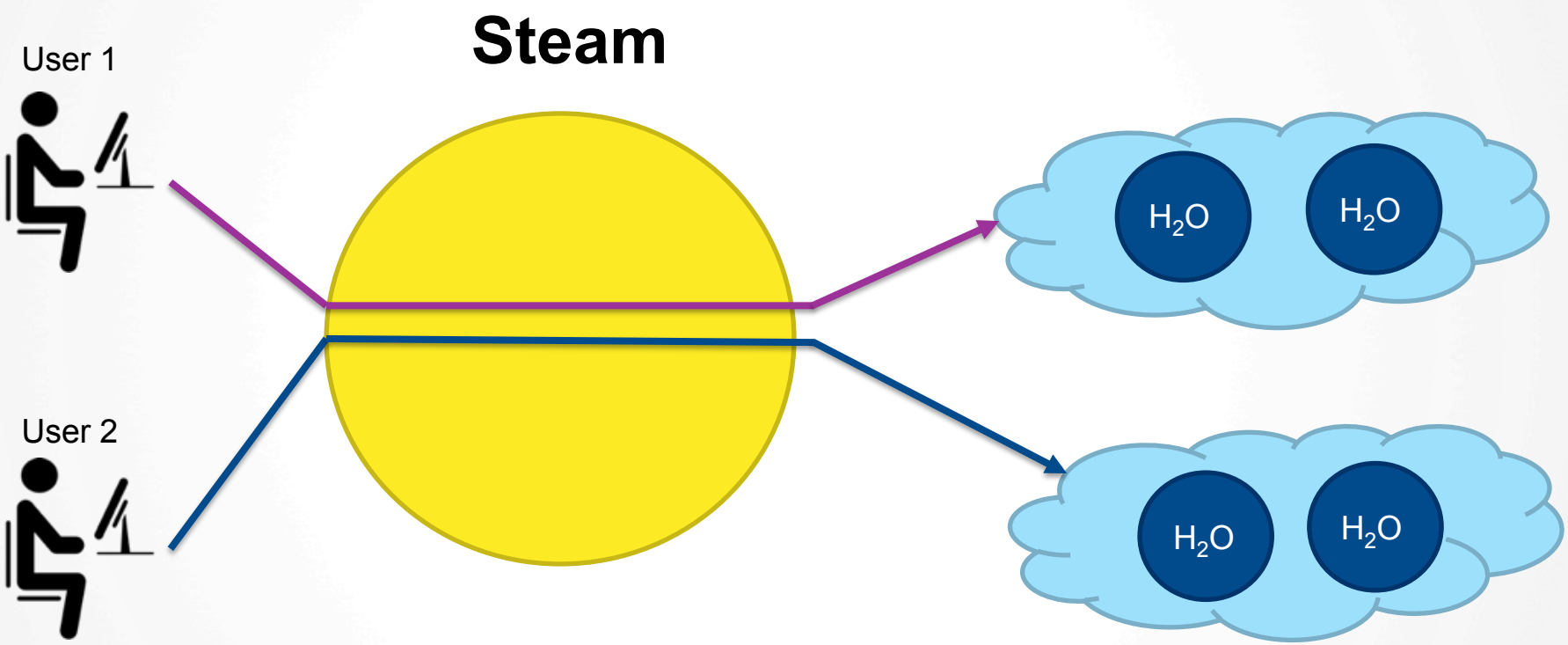
PURE BATCH H2O DESIGN PATTERN



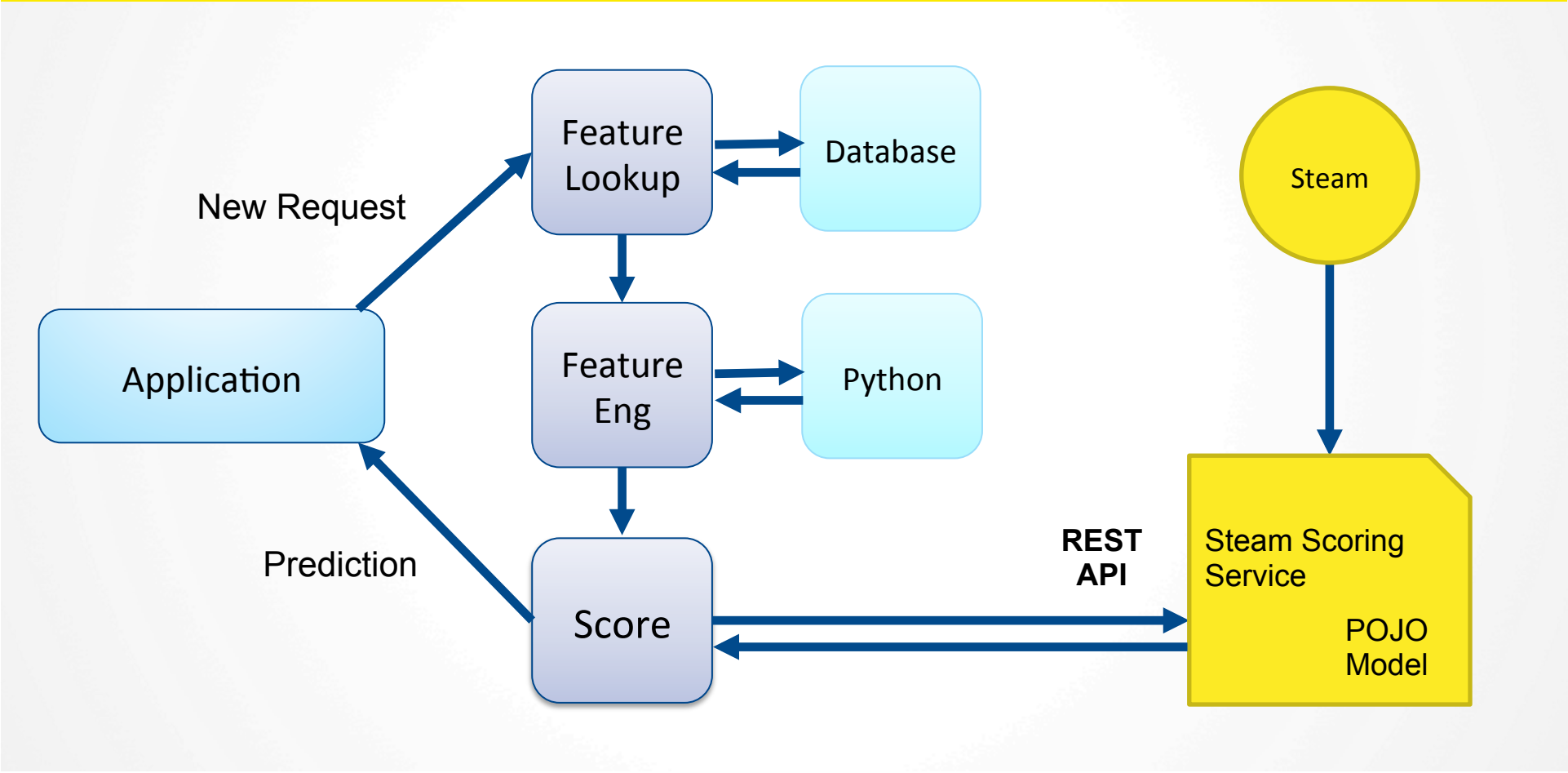
STEAM DESIGN PATTERN

- Training data lives in Hadoop
- POJO models are deployed as a service with a REST API

STEAM DESIGN PATTERN (TRAINING)



STEAM DESIGN PATTERN (SCORING)



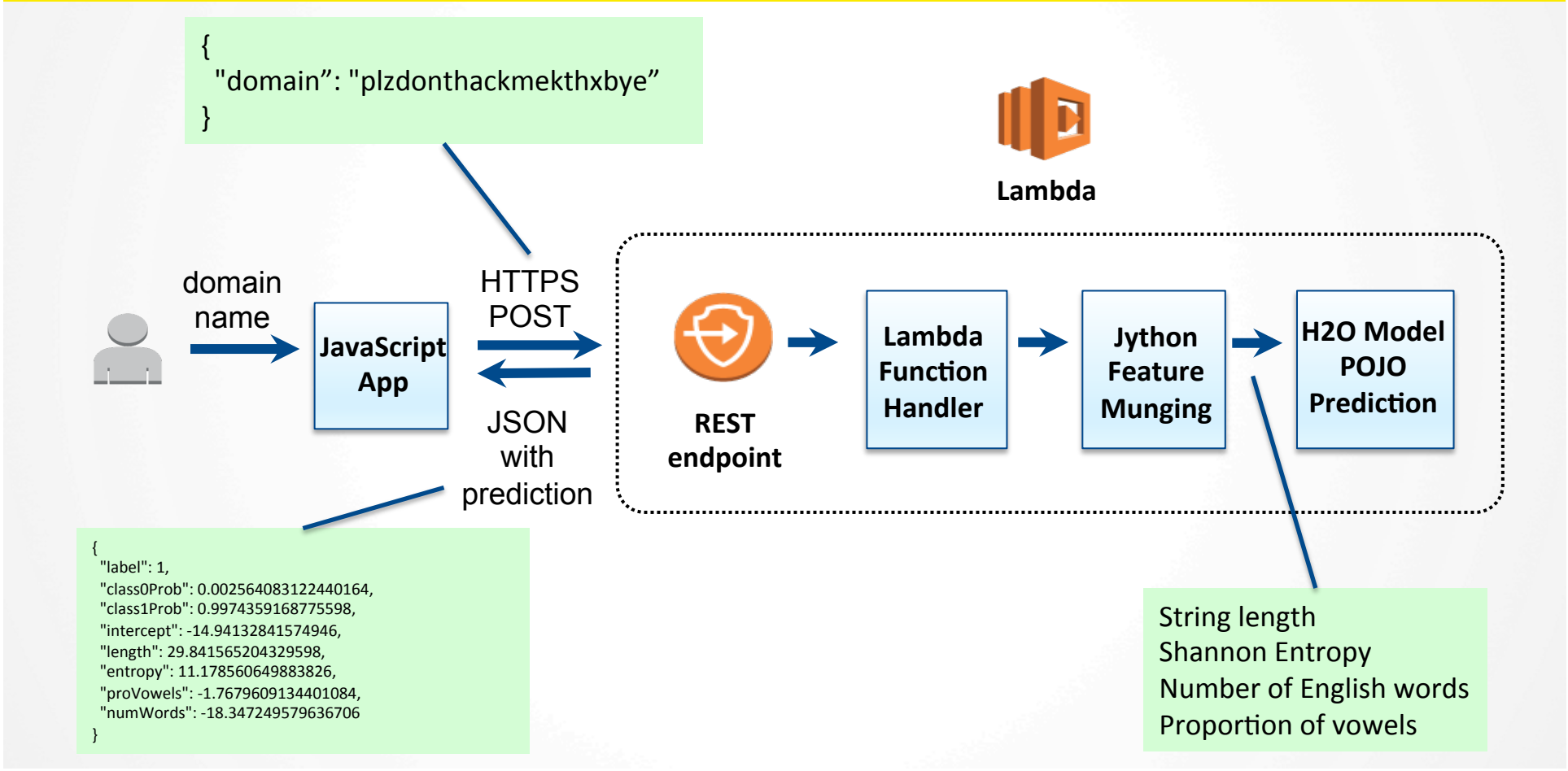
AWS LAMBDA DESIGN PATTERN

- Training data lives in data lake
- Scoring data arrives via API call
- Real-time POJO scoring in AWS Lambda

MOTIVATION FOR AWS LAMDA

USE CASE	REQUIREMENTS	TECHNOLOGY
Feature engineering	- Use of Python for DataSci - Code re-use for training and production - Speed	Jython
Machine Learning Prediction	- High model accuracy - Real-time environment	H2O Generated POJO Model
Deployment to Production	- Easy handoff to Ops - Speed to deployment - Built-in scalability - No infrastructure management	AWS Lambda

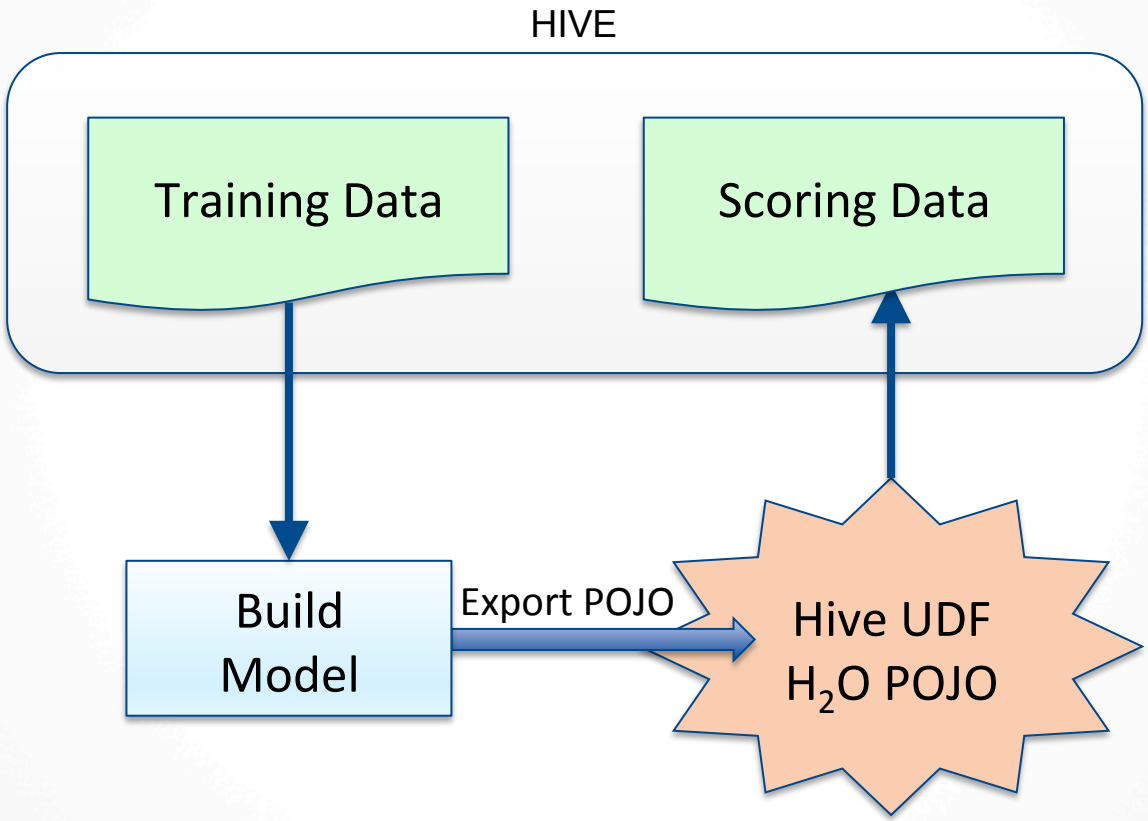
AWS LAMBDA DESIGN PATTERN (SCORING)



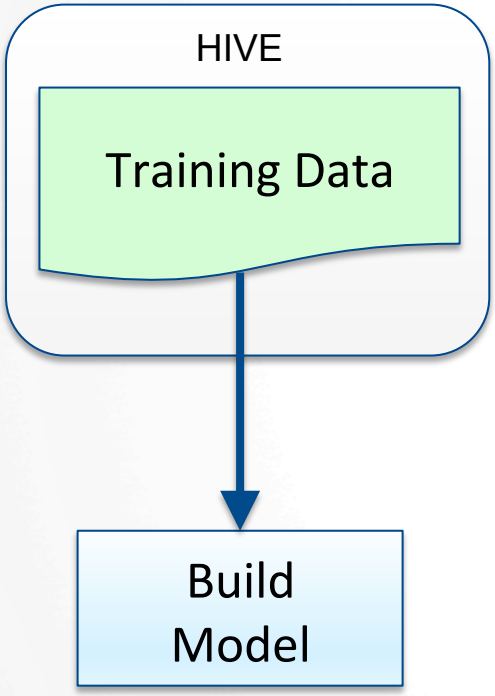
HIVE UDF DESIGN PATTERN

- Training data lives in Hive
- Scoring data lives in Hive
- Batch POJO scoring
- Models built infrequently
- Scoring occurs daily
  - Strict SLA to complete the batch job

HIVE UDF DEPLOYMENT

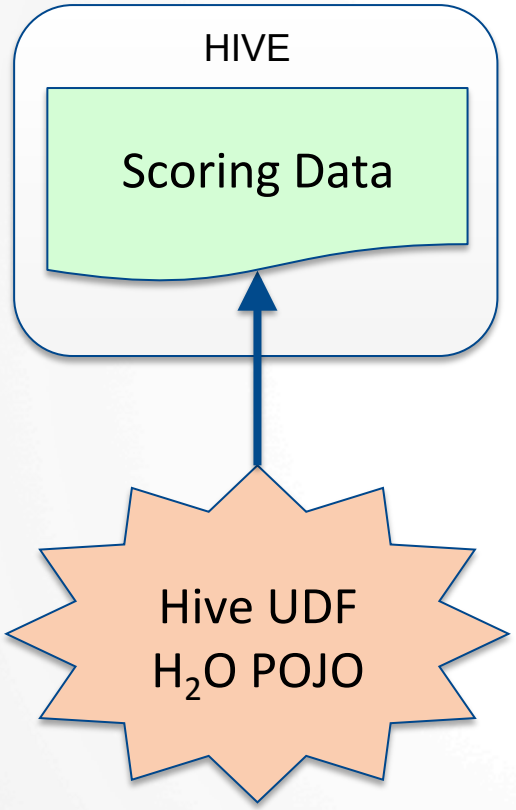


HIVE UDF (TRAINING)



- Prepare training data
  - Data may be stored in unreadable format!
  - Export to read _part files for H_2O
- Launch H_2O
- Build models
- Export models as POJOs

HIVE UDF (SCORING)

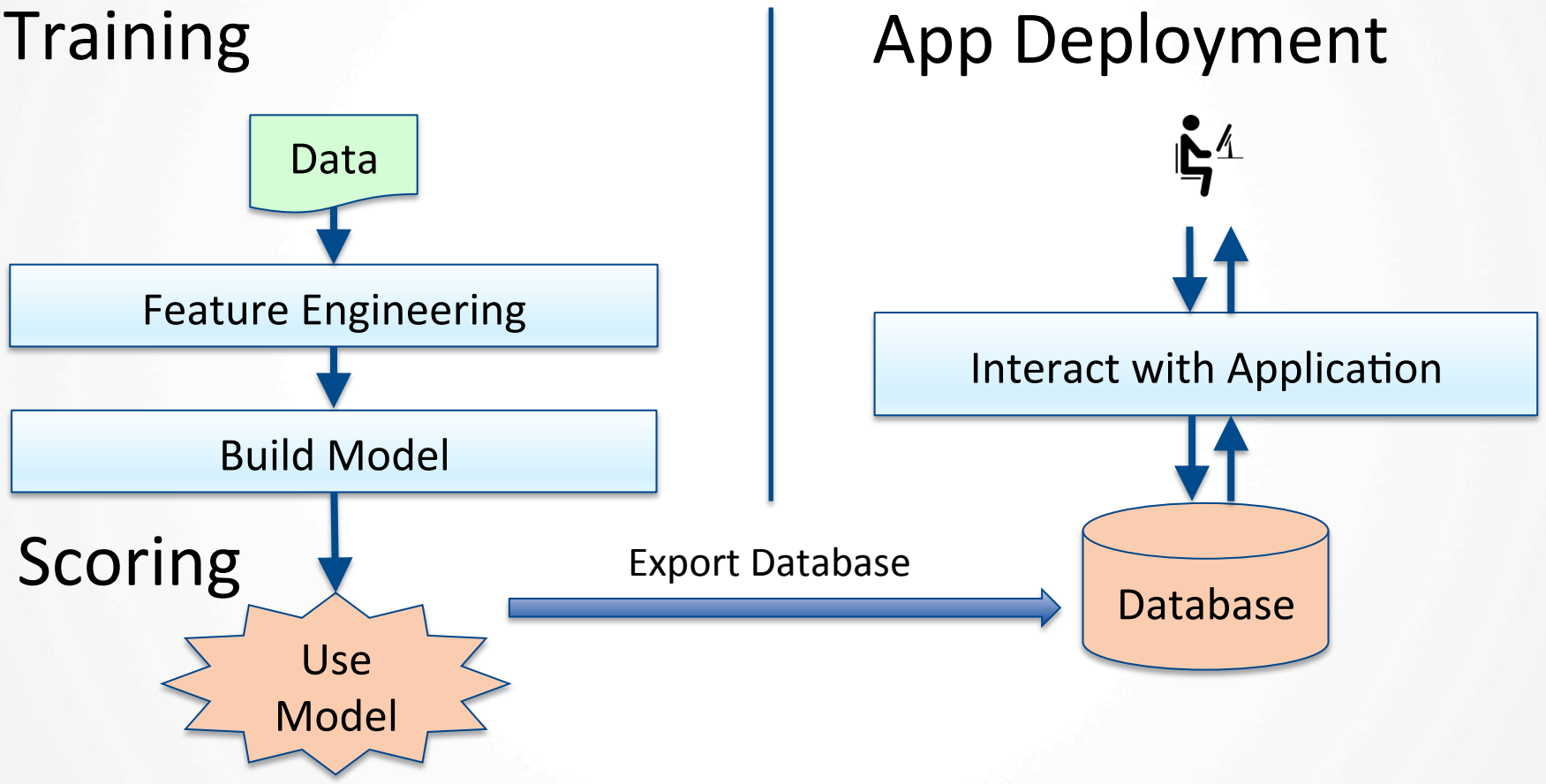


- Create Hive UDF
  - Extend GenerUDF class
  - initialize(): initialize H_2O models
  - evaluate(): loop through models & score
  - Build JAR file
- Hive
  - Load JARs, define UDF, set columns
    - ADD JAR localjars/h2o-genmodel.jar;
      ADD JAR target/ScoreData-1.0-SNAPSHOT.jar;
    - CREATE TEMPORARY FUNCTION fn AS …
    - set hivevar:scoredatacolnames=C1,C2…
  - Score!
    - select fn(${scoredatacolnames}) from TABLE

JOURNALS.AI DESIGN PATTERN

- Driven by R
- Training & scoring on bare metal servers
- Scores pre-calculated and stored in a database (e.g. MS SQL, SQLite)
- Application with domain-specific UI built on top of the database
- Flexible deployment (laptop, server, cloud, etc.)

JOURNALS.AI PRODUCTIONIZED



Journals.ai POWERED BY H_2O.ai

JOURNALS PROCESSED TO DATE 2,525,234,345

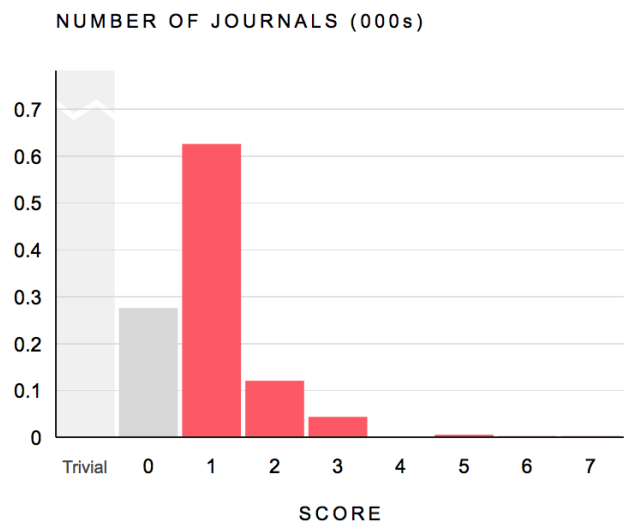
DASHBOARD | TRANSACTION FLOWS

Apex, Inc.

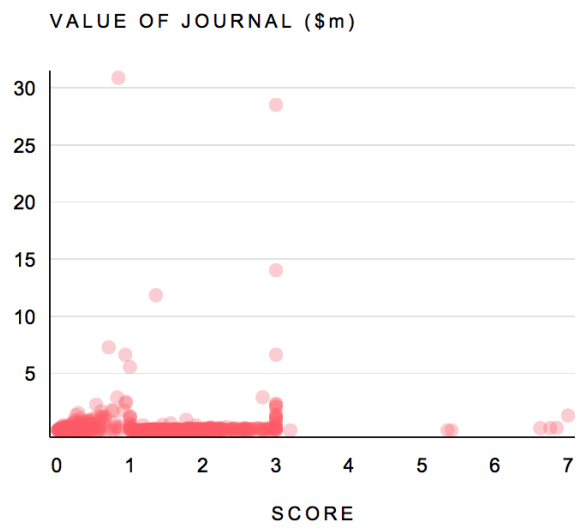
PERIOD 01-01-2015 to 12-31-2015

Reporting currency	*GL system*	*Entity type*	*Industry*
USD	SAP	Subsidiary of Listed Entity	Consumer packaging

Journal by degree of unusualness



Journal by score and value



Q & A

- Thanks for attending!
- Slides in github:
  - h2oai/h2o-meetups/tree/master/2016_07_19_H2O_Open_Tour_NYC_Prod
- Send follow up questions to:

Tom Kraljevic
tomk@h2o.ai