When you need to add Deep Learning to raise your next round of funding

@holdenkarau

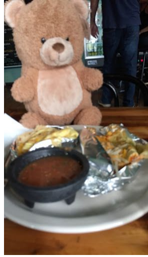
Photo by: brando

Holden:

- My name is Holden Karau
- Prefered pronouns are she/her
- Developer Advocate at Google
- Apache Spark PMC :)
- previously IBM, Alpine, Databricks, Google, Foursquare & Amazon
- co-author of Learning Spark & High Performance Spark
- @holdenkarau
- Slide share http://www.slideshare.net/hkarau
- Linkedin https://www.linkedin.com/in/holdenkarau
- Github https://github.com/holdenk
- Spark Videos http://bit.ly/holdenSparkVideos

Who I think you wonderful humans are?

- Nice enough people
- Don’t mind pictures of cats
- Might know some the different distributed systems talked about
- Possibly know some Python or R

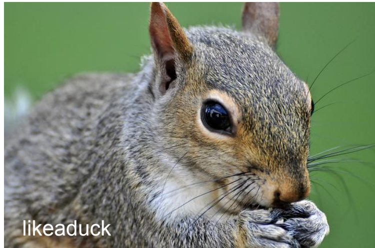


What will be covered?

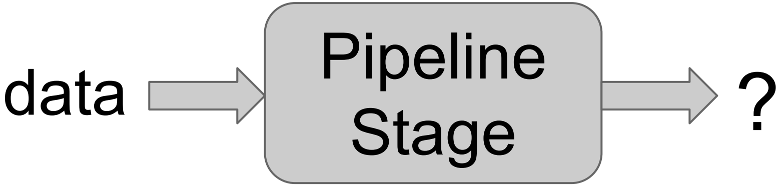
- Why people care about deep learning
- What the different (Spark-ish) deep learning libraries are
- Why the deep learning state of Spark is kind of sad
- How we can make it more awesome

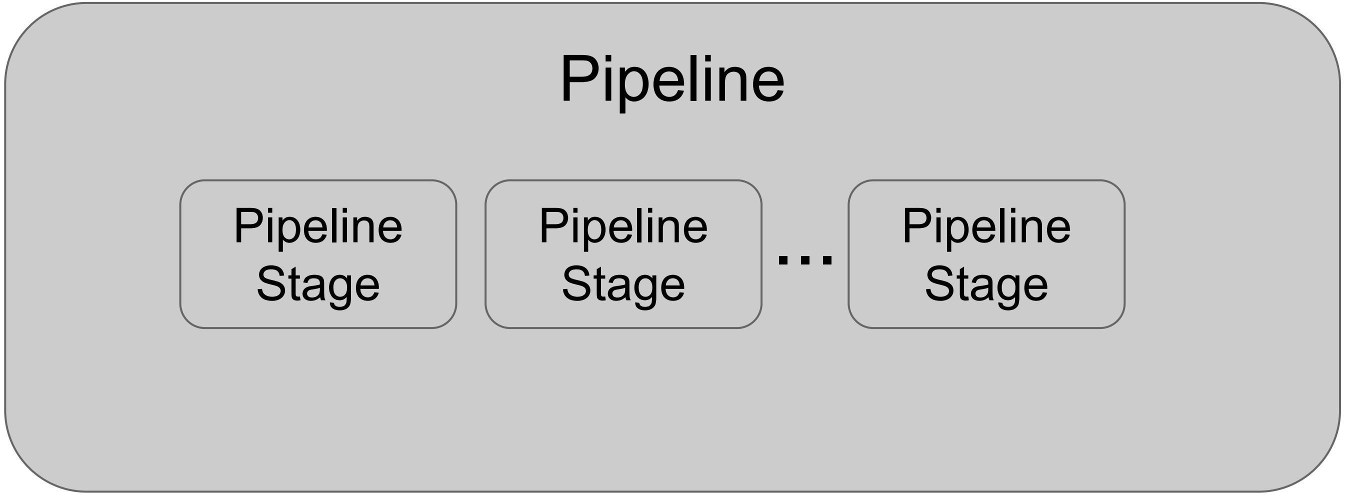
Why people care about deep learning?

- Cat pictures
    - Colourizing cat pictures, generating cat pictures, finding cat pictures.
- Trying to raise more money in San Francisco
- Transfer learning (aka if this can predict cats maybe it can find squirrels)
- I built a data lake and now I need to justify it
- Keeping resume up to-date

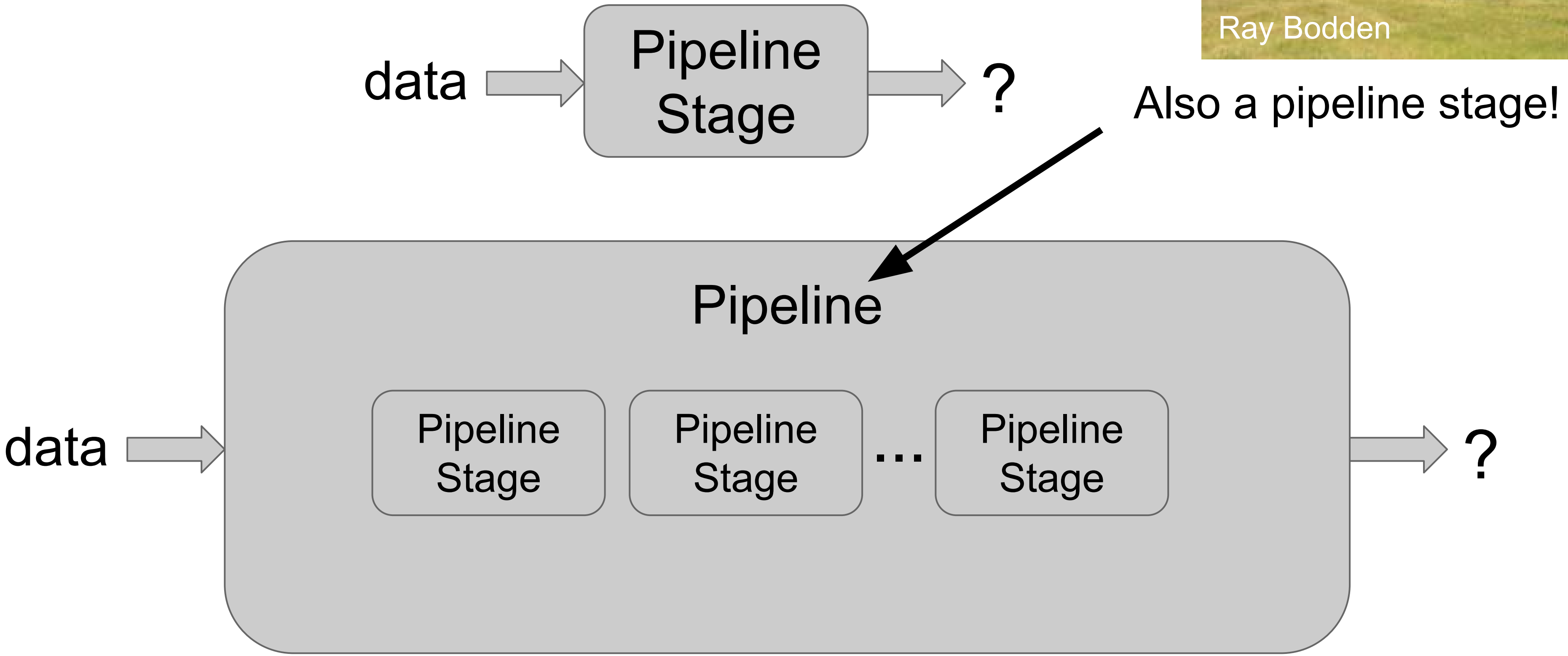


Spark ML Pipelines

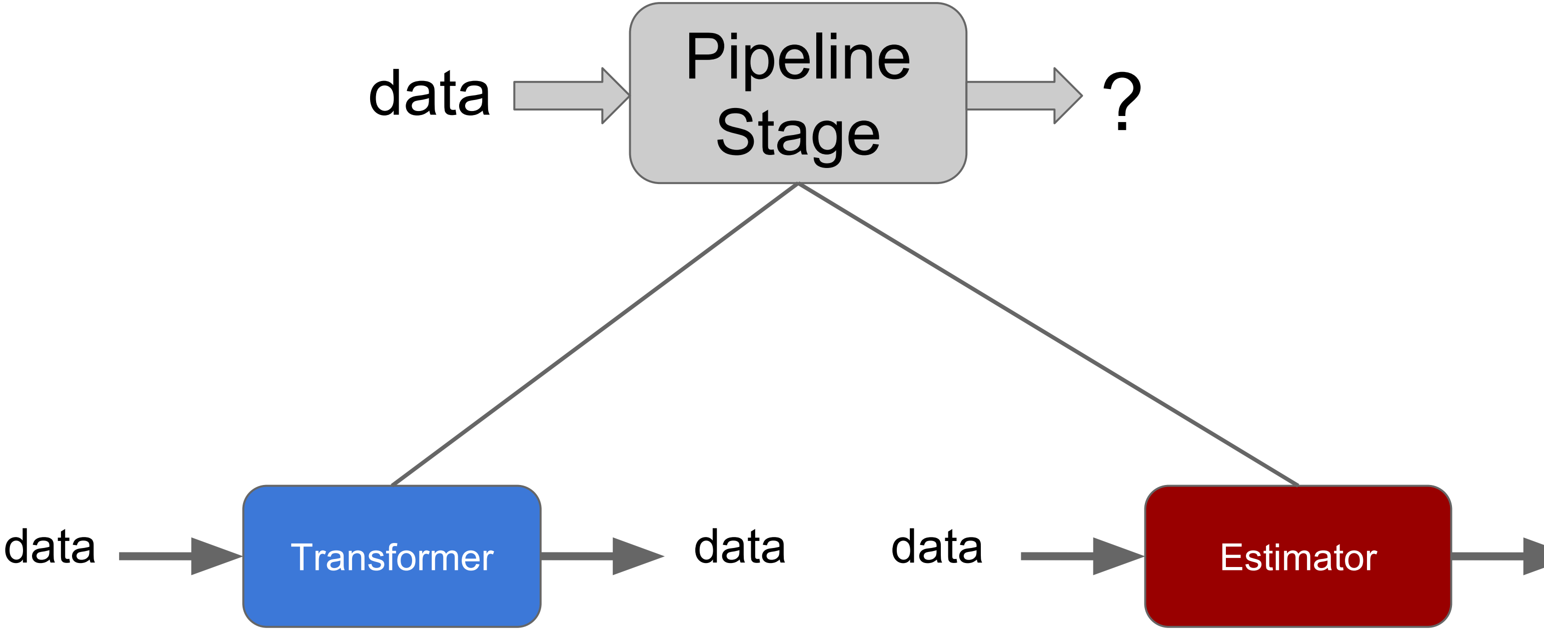




Spark ML Pipelines

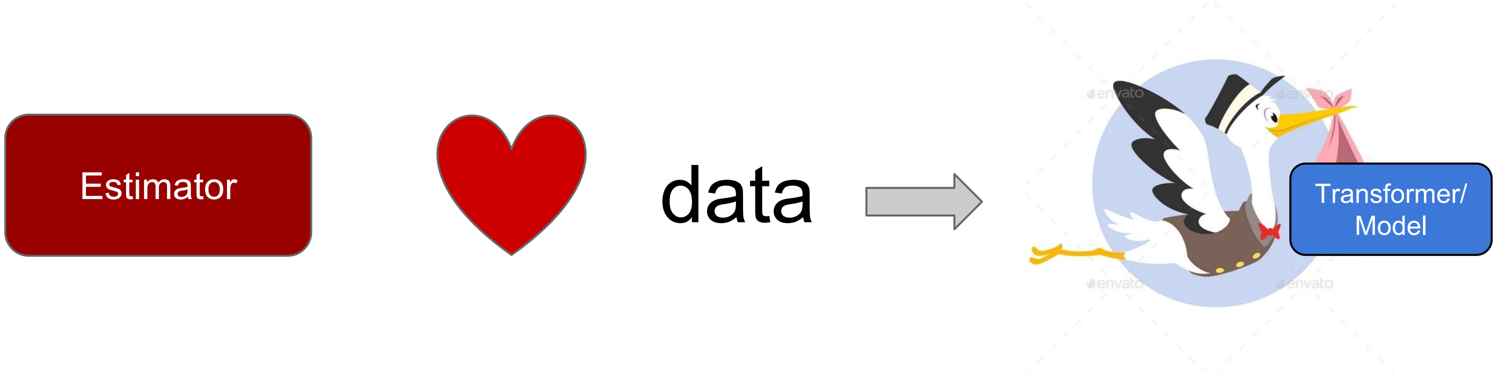


Two main types of pipeline stages



How are transformers made?

```
class Estimator extends PipelineStage {
  def fit(dataset: Dataset[_]): Transformer = {
    // magic happens here
  }
}
```



Old-skool ML (Decision Trees)

- Keeping track of intermediate data and calling fit/transform on every stage is way too much work
- This problem is worse when more stages are used
- Use a pipeline instead!

```
val assembler = new VectorAssembler()
assembler.setInputCols(Array("gre", "gpa", "prestige"))
val sb = new StringIndexer()
sb.setInputCol("admit").setOutputCol("label")
val dt = new DecisionTreeClassifier()
val pipeline = new Pipeline()
pipeline.setStages(Array(assembler, sb, dt))
val pipelineModel = pipeline.fit(df)
```

Yay! You have an ML pipeline!

Photo by Jessica Fiess-Hill

What are your options to raise money?

- DL4J
  - Relatively deep spark integration
  - Largely Java based as the name implies
- Big DL
- Tensorflow
  - TensorFlowOnSpark, TensorFlow_Scala, TensorFrames, ...
- Keras
- PyTorch (sorry I meant JTorch)
- MXNet
- A linear regression library that you call deep learning and hope no one notices
- etc.

So wait do I have to learn Python?

- No…..
- But you will have to (hopefully indirectly) use it / JNI / Py4J or similar
- But if you want to learn Python you can!

Big DL on Spark - Training (custom)

Tomomi

```
val data = // Data prep goes here. How long could that take?
val Array(trng, chk) = data.randomSplit(Array(split, 1 - split))

Optimizer(
  model = buildModel(classNum), sampleRDD = trng,
  criterion = new ClassNLLCriterion[Float](), batchSize = batch
).setOptimMethod(method)
 .setValidation(Trigger.everyEpoch, valRDD, Array(new
Top1Accuracy[Float]), param.batchSize)
  .setEndWhen(Trigger.maxEpoch(1))
  .optimize()
```

Big DL on Spark - Prediction (ML pipeline based)

```
val model = loadModel(param)
val valTrans = new DLClassifierModel(model, Array(3, imageSize,
imageSize))
  .setBatchSize(param.batchSize)
  .setFeaturesCol("features")
  .setPredictionCol("predict")

valTrans.transform(data)
```

DL4J - Hey this looks kind of similar….


```java
JavaRDD<Dataset> = // Data prep goes here.
TrainingMaster trainingMaster = new ParameterAveragingTrainingMaster.Builder(1)
        // Optional config options go here.
        .build();

//Create the SparkDl4jMultiLayer instance
SparkDl4jMultiLayer sparkNetwork = new SparkDl4jMultiLayer(sc, networkConfig,
trainingMaster);

//Fit the network using the training data:
sparkNetwork.fit(trainingData);
```

DL4J - Pipelines?

TensorFlow - So many options, most not* fun in JVM

- TensorFlow Scala - Works in Scala not a lot of distributed support
- TensorFlowOnSpark - Works in Spark but assumes Python
- Regular TensorFlowJava - not guaranteed to be API stable, no Spark magic
- Hops Tensorflow (python only)
- TensorFrames - Experimental only, JVM support-ish (can't define new graphs in JVM but can load Python graphs)
- Horovod (python only for now)

Ok but I want pipelines, they sound cool*

- Spark-deep-learning seems to be the primary package for exposing distributed DL libs in a Spark pipeline like interface
- It's Python only, buuuuut well the next slide might give you some ideas about how you can join me** in the adventures to fix this.

What about the next new shiny tool?

- Probably going to be built with Python APIs
- Spark has an ML pipeline API we can implement
- With a bit of work we can expose arbitrary* Python stages into Scala & Java Spark
- See sparklingml for examples (currently basic spacy)

Exposing a Spark Python into the JVM:*

```
def func(cls, *args):
    lang = args[0]

    def inner(inputString):
        """Tokenize the inputString using spacy for
        the provided language."""
        nlp = SpacyMagic.get(lang)
        return list(map(lambda x: x.text,
list(nlp(inputString))))
    return inner
```

*See sparklingml repo for the questionable magic that wires this together.

Using it in the JVM*:


```
val transformer = new SpacyTokenizePython()
transformer.setLang("en")
val input = spark.createDataset(
  List(InputData("hi boo"), InputData("boo")))
transformer.setInputCol("input")
transformer.setOutputCol("output")
```

*See sparklingml repo for the questionable magic that wires this together.

Ok so I raised some money buuuut...

- It’s kind of slow :(
- You enjoy copying your data around right?
- Why does it take so long to predict?
- Now my investors/interns/engineers want to use [X] deep learning library

PySpark - Everything old is new again

- The Python interface to Spark
- The very fun basis to integrate with many deep learning libraries
- Same general technique used as the bases for the C#, R, Julia, etc. interfaces to Spark
- Fairly mature, integrates well-ish into the ecosystem, less a Pythonrific API
- Has some serious performance hurdles from the design

A quick detour into PySpark's internals

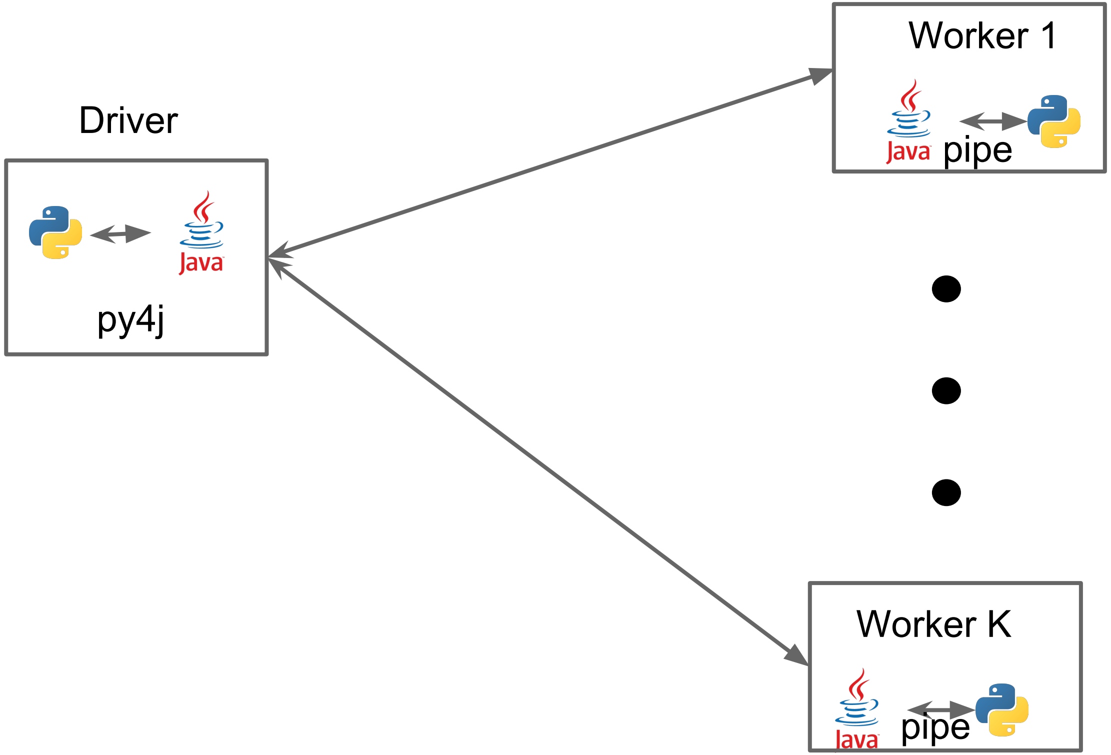
\+

\+ JSON

Spark in Scala, how do we talk to Py libs?

- Py4J + pickling + JSON and magic
  - Py4j in the driver
  - Pipes to start python process from java exec
  - cloudPickle to serialize data between JVM and python executors (transmitted via sockets)
  - Json for dataframe schema
- Data from Spark worker serialized and piped to Python worker --> then piped back to jvm
  - Multiple iterator-to-iterator transformations are still pipelined :)
  - So serialization happens only once per stage

So what does that look like?



So how does this impact DL?

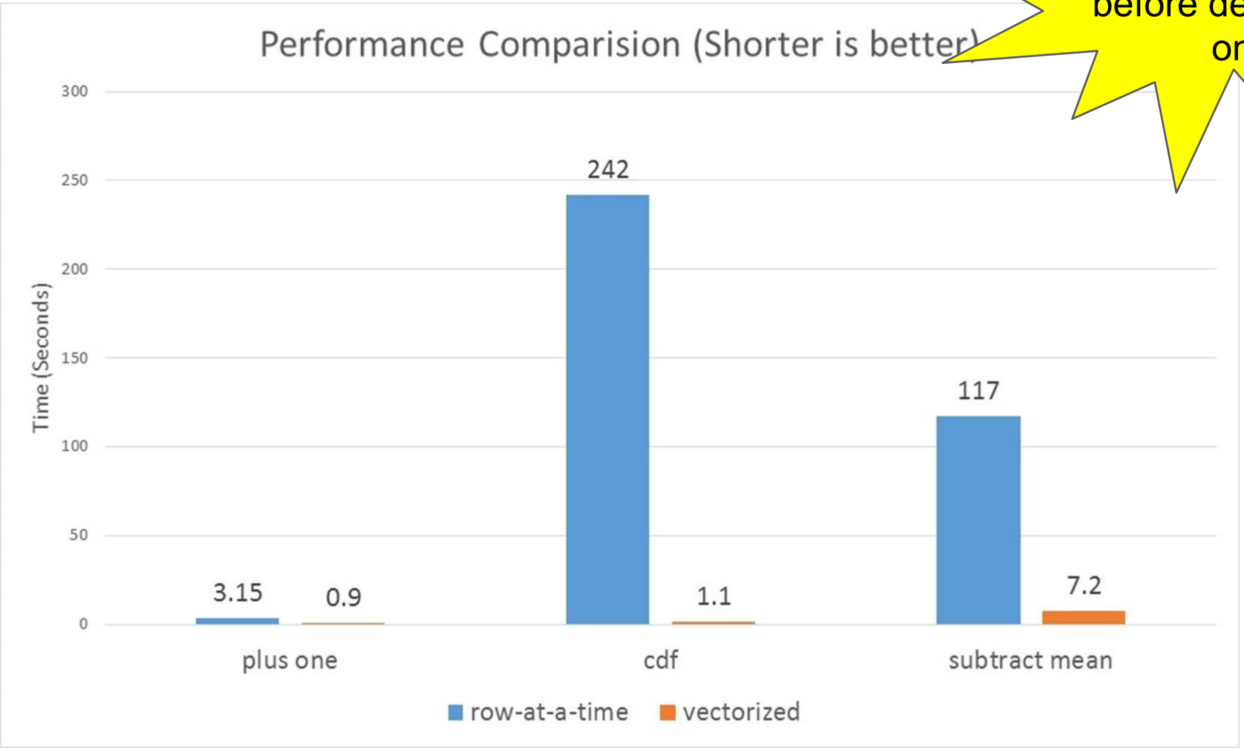
- Double serialization cost makes everything more expensive
- Native memory isn’t properly controlled, can over container limits if deploying on YARN or similar
- Error messages make ~0 sense
- Spark Features aren’t automatically exposed, but exposing them is normally simple

The “future”*: faster interchange

- By future I mean availability starting in the next 3-6 months (with more improvements after).
  - Yes much of this code exists, it just isn’t released yet so I’m sure we’ll find all sorts of bugs and ways to improve.
  - Relatedly you can help us test in Spark 2.3 when we start the RC process to catch bug early!
- Unifying our cross-language experience
  - And not just “normal” languages, CUDA counts yo


*
*
*Arrow: likely the future. I really hope so. Spark 2.3 and beyond!

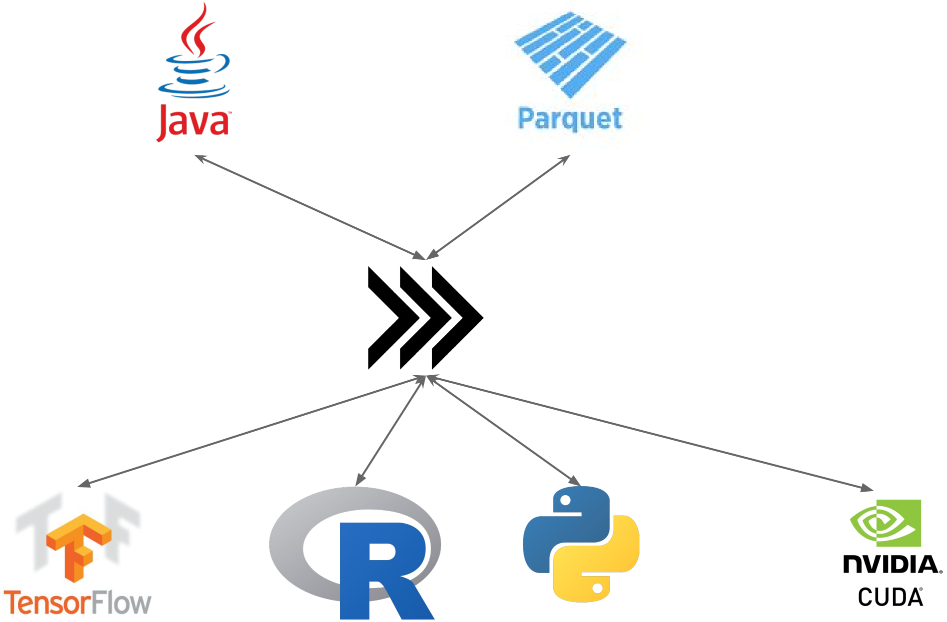
What does the future look like?*



*Vendor benchmark.Verify before depending on.

*Source: https://databricks.com/blog/2017/10/30/introducing-vectorized-udfs-for-pyspark.html.

Arrow (a poorly drawn big data view)



Logos trademarks of their respective projects

Rewriting your friends' Python code

```
spark.catalog.registerFunction(
    "add", lambda x, y: x + y, IntegerType())
```

```
=>
```

```
add = pandas_udf(lambda x, y: x + y, IntegerType())
```

Why now?

- There's been better formats/options for a long time
- JVM devs want to use libraries in other languages with lots of data
  - e.g. startup + Deep Learning + ? => profit
- Arrow has solved the chicken-egg problem by building not just the chicken & the egg, but also a hen house

BEAM: Beyond the JVM

- Adopting a new architecture for going beyond the JVM, come join us
- tl;dr : uses grpc / protobuf
  - Similar to the common design but with more efficient representations (often)
- But exciting new plans to unify the runners and ease the support of different languages (called SDKS)
  - See https://beam.apache.org/contribute/portability/
- If this is exciting, you can come join me on making BEAM work in Python3
  - Yes we still don't have that :(
  - But we're getting closer & you can come join us on BEAM-2874 :D
- No mixed pipeline support right now (would make DL in "Java" easier)

Before I leave: regression is cool

- org.apache.spark.ml.classification
  - BinaryLogisticRegressionClassification, DecissionTreeClassification, GBTClassifier, etc.
- org.apache.spark.ml.regression
  - DecissionTreeRegression, GBTRegressor, IsotonicRegression, LinearRegression, etc.
- org.apache.spark.ml.recommendation
  - ALS
- You can also check out spark-packages for some more
- But possible not your special AwesomeFooBazinatorML

Lots of data prep stages:

- org.apache.spark.ml.feature
    - ~30 elements from VectorAssembler to Tokenizer, to PCA, etc.
- Often simpler to understand while getting started with building our own stages

Custom Estimators/Transformers in the Wild


```scala
/**
 * XGBoost Estimator to produce a XGBoost model
 */
class XGBoostEstimator private[spark](
  override val uid: String, xgboostParams: Map[String, Any])
  extends Predictor[MLVector, XGBoostEstimator, XGBoostModel]
  with LearningTaskParams with GeneralParams with BoosterParams with MLWritable {
```

```scala
trait LuceneTransformer[T <:LuceneTransformer[T]]
    extends UnaryTransformer[String, Array[String], T]
```

```scala
class MXNet extends Predictor[Vector, MXNet, MXNetModelWrap] {

  private val logger: Logger = LoggerFactory.getLogger(classOf[MXNet])
  private val p: MXNetParams = new MXNetParams
  private var _featuresCol: String = _
  private var _labelCol: String = _

  override val uid = UUID.randomUUID().toString

  override def train(dataset: DataFrame) : MXNetModelWrap = {
```

Classification/Regression

xgboost

Feature Transformation

Lucene transformers

Spacy

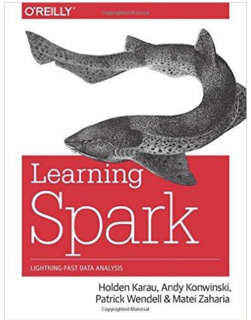
spark-nlp

Deep Learning!

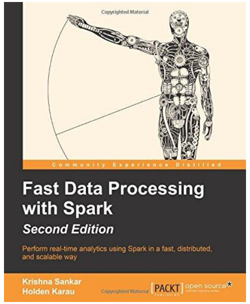
dl4j-spark (deprecated)

References

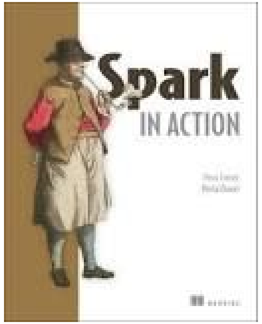
- Apache Arrow: https://arrow.apache.org/
- Brian (IBM) on initial Spark + Arrow https://arrow.apache.org/blog/2017/07/26/spark-arrow/
- Li Jin (two sigma) https://databricks.com/blog/2017/10/30/introducing-vectorized-udfs-for-pyspark.html
- Bill Maimone https://blogs.nvidia.com/blog/2017/06/27/gpu-computation-visualization/



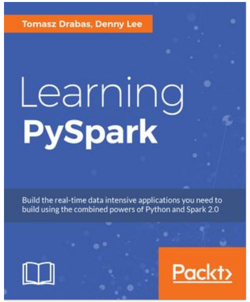
Learning Spark



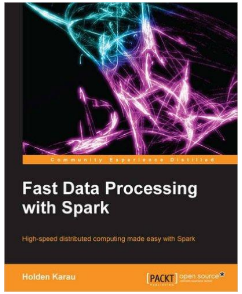
Fast Data Processing with Spark
(2nd edition)



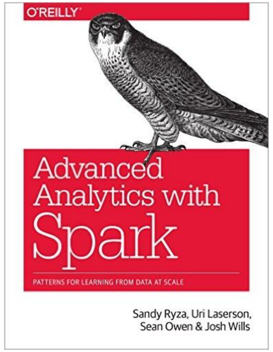
Spark in Action



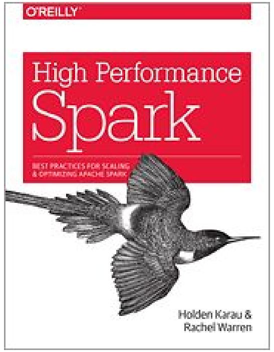
Learning PySpark



Fast Data Processing with Spark
(Out of Date)

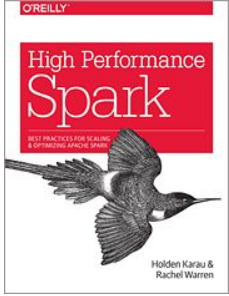


Advanced Analytics with Spark



High Performance Spark

High Performance Spark!



You can buy it today, the O'Reilly folks have it upstairs (& so does Amazon).

Only one chapter on non-JVM and nothing on Arrow or Deep Learning, I'm sorry.

Cats love it*

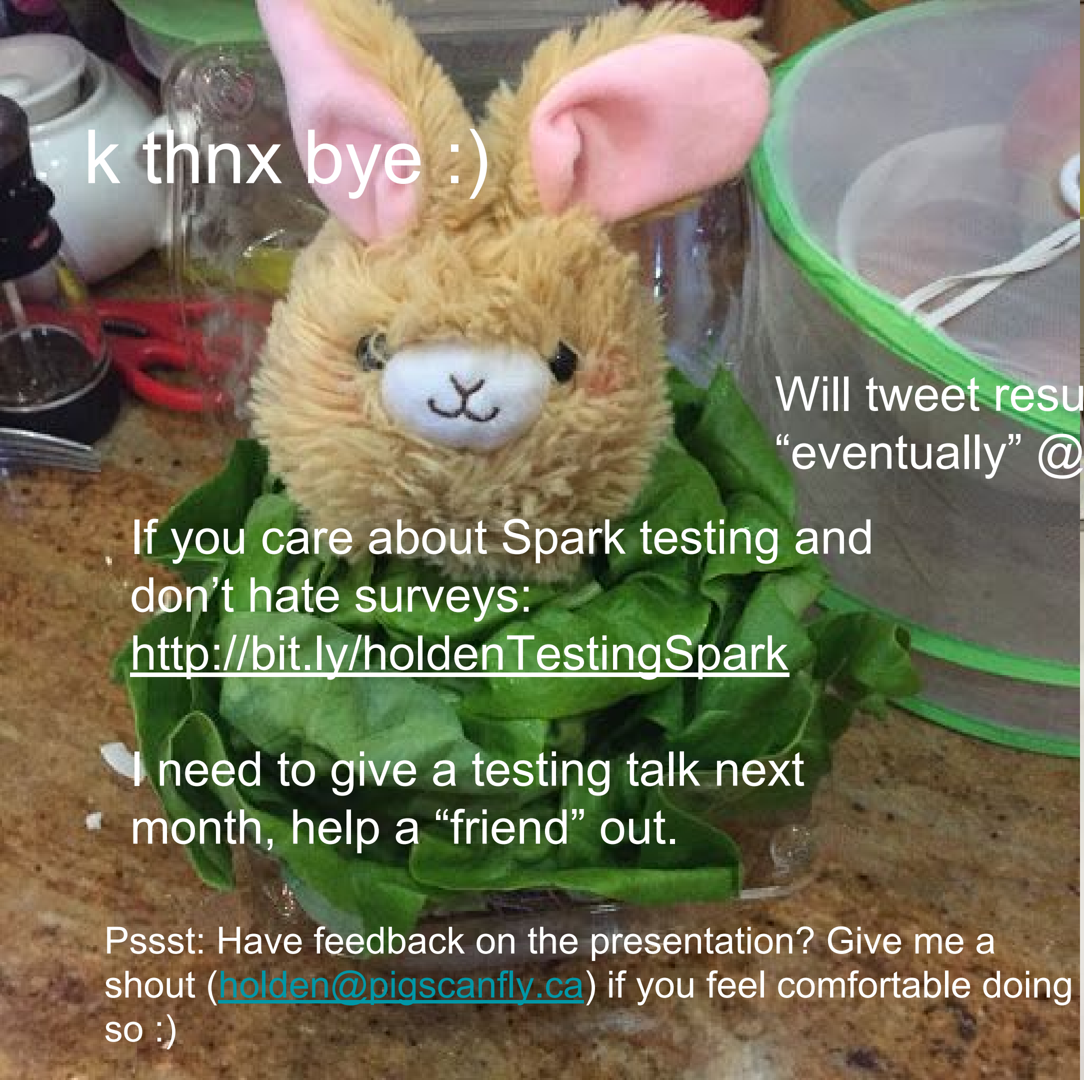
*Or at least the box it comes in. If buying for a cat, get print rather than e-book.

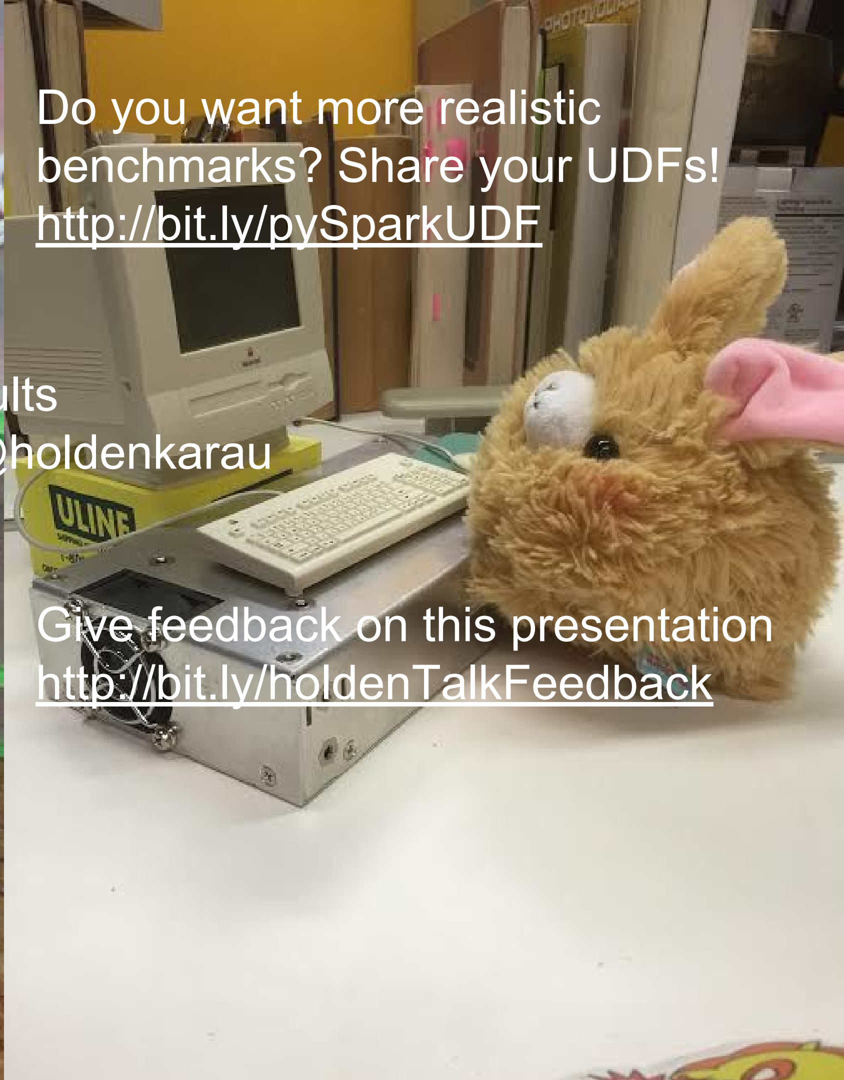
Spark Videos

- Apache Spark Youtube Channel
- My Spark videos on YouTube -
  - http://bit.ly/holdenSparkVideos
- Spark Summit 2014 training
- Paco's Introduction to Apache Spark

And some upcoming talks:

- Strata San Jose
- Strata London
- Kafka Summit London
- QCon Brasil
- QCon AI SF
- Know of interesting conferences/webinar things that should be on my radar? Let me know!





k thnx bye :)

If you care about Spark testing and don’t hate surveys: http://bit.ly/holdenTestingSpark

I need to give a testing talk next month, help a “friend” out.

Will tweet results “eventually” @holdenkarau

Do you want more realistic benchmarks? Share your UDFs! http://bit.ly/pySparkUDF

Give feedback on this presentation http://bit.ly/holdenTalkFeedback

Pssst: Have feedback on the presentation? Give me a shout (holden@pigscanfly.ca) if you feel comfortable doing so :)

Bonus Slides

Maybe you ask a question and we go here :)

Installing the Python dependencies?

- Your machines probably already have python
- But they might not have “special_business_logic”
  - Very special business logic, no one wants change fortran code*.
- Option 1: Talk to your vendor**
- Option 2: Try some open source software
- Option 3: Containers containers containers***
- We’re going to focus on option 2!

*Because it’s perfect, it is fortran after all.

** I don’t like this option because the vendor I work for doesn’t have an answer.

*** Great for your resume!

coffee_boat to the rescue*

```
# You can tell it's alpha cause were installing from github
!pip install --upgrade
git+https://github.com/nteract/coffee_boat.git
# Use the coffee boat
from coffee_boat import Captain
captain = Captain(accept_conda_license=True)
captain.add_pip_packages("pyarrow", "edtf")
captain.launch_ship()
sc = SparkContext(master="yarn")
# You can now use pyarrow & edtf
captain.add_pip_packages("yourmagic")
# You can now use yourmagic in transformations!
```

What’s still going to hurt?

- Per-record streaming
  - Arrow is probably less awesome for serialization
  - But it’s still better than we had before
- Debugging is just going to get worse
- Custom data formats
  - Time to bust out the C++ code and a bottle of scotch / matte as appropriate
  - Or just accept the “legacy” performance

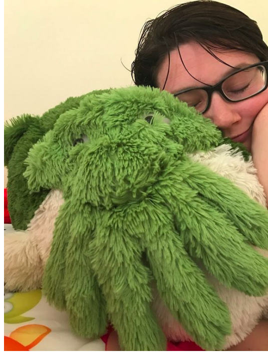
We can do that w/Kafka streams..

- Why bother learning from our mistakes?
- Or more seriously, the mistakes weren't that bad...

Our "special" business logic

```python
def transform(input):
    """
    Transforms the supplied input.
    """
    return str(len(input))
```

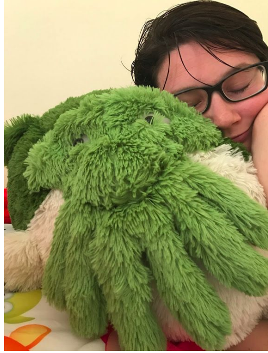
Let’s pretend all the world is a string:



```
override def transform(value: String): String = {
  // WARNING: This may summon cuthuluhu
  dataOut.writeInt(value.getBytes.size)
  dataOut.write(value.getBytes)
  dataOut.flush()
  val resultSize = dataIn.readInt()
  val result = new Array[Byte](resultSize)
  dataIn.readFully(result)
  // Assume UTF8, what could go wrong? :p
  new String(result)
}
```

From https://github.com/holdenk/kafka-streams-python-cthulhu

Then make an instance to use it...

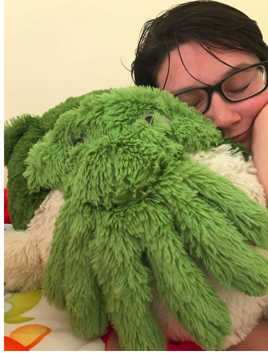


```
val testFuncFile =
  "kafka_streams_python_cthulhu/strlen.py"
stream.transformValues(
    PythonStringValueTransformerSupplier(testFuncFile))
// Or we could wrap this in the bridge but thats effort.
```

From https://github.com/holdenk/kafka-streams-python-cthulhu

Let’s pretend all the world is a string:

```python
def main(socket):
    while (True):
        input_length = _read_int(socket)
        data = socket.read(input_length)
        result = transform(data)
        resultBytes = result.encode()
        _write_int(len(resultBytes), socket)
        socket.write(resultBytes)
        socket.flush()
```



From https://github.com/holdenk/kafka-streams-python-cthulhu

What does that let us do?

- You can add a map stage with your data scientists Python code in the middle
- You’re limited to strings*
- Still missing the “driver side” integration (e.g. the interface requires someone to make a Scala class at some point)
- Probably not any good for deep learning (you likely want bytes)