

TensorFrames: Google Tensorflow with Apache Spark

Timothée Hunter
Databricks, Inc.



About Databricks

Why Us

- Created Apache Spark to enable big data use cases with a single engine.
- Contributes 75% of Spark's code - 10x more than others.



Our Product

- Bring Spark to the enterprise: The just-in-time data platform.
- Fully managed platform powered by Apache Spark.
- A unified solution for data science and engineering teams.



About me

Software engineer at Databricks

Apache Spark contributor

Ph.D. UC Berkeley in Machine Learning

(and Spark user since Spark 0.2)

Outline

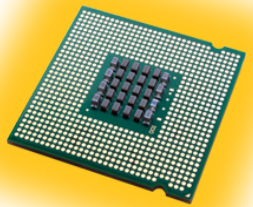
- **Numerical computing with Apache Spark**
- Using GPUs with Spark and TensorFlow
- Performance details
- The future

Numerical computing for Data Science

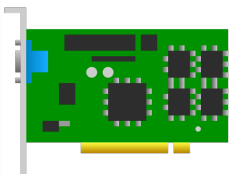
- Queries are data-heavy
- However algorithms are computation-heavy
- They operate on simple data types: integers, floats, doubles, vectors, matrices

The case for speed

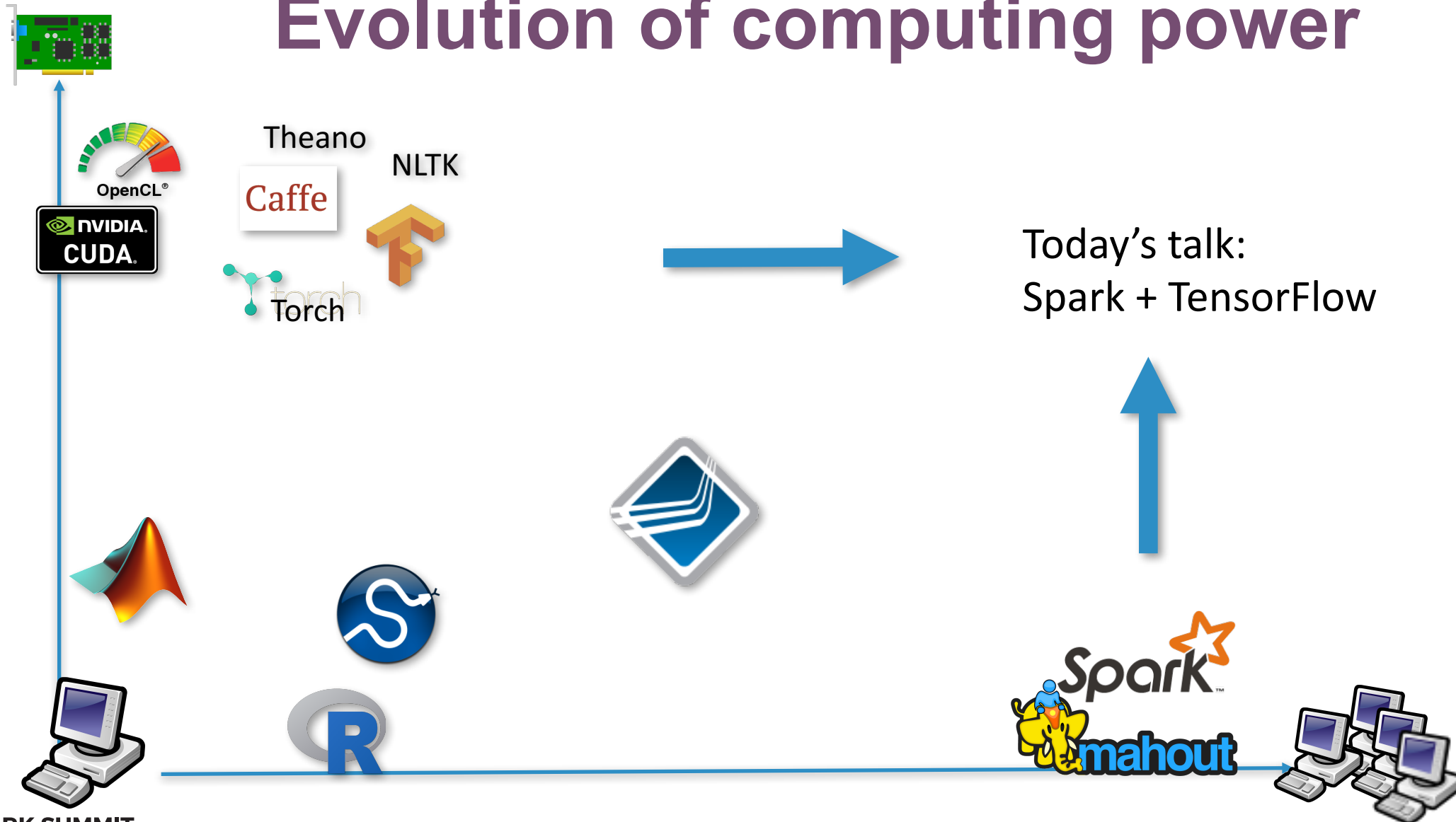
- Numerical bottlenecks are good targets for optimization
- Let data scientists get faster results
- Faster turnaround for experimentations
- How can we run these numerical algorithms faster?



Evolution of computing power



Evolution of computing power



Evolution of computing power

- Processor speed cannot keep up with memory and network improvements
- Access to the processor is the new bottleneck
- Project Tungsten in Spark: leverage the processor's heuristics for executing code and fetching memory
- Does not account for the fact that the problem is *numerical*

Outline

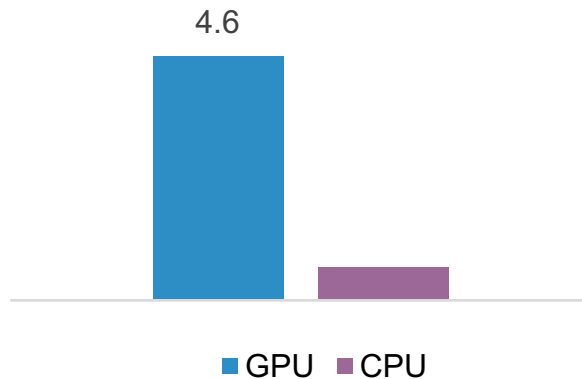
- Numerical computing with Apache Spark
- **Using GPUs with Spark and TensorFlow**
- Performance details
- The future

GPGPUs

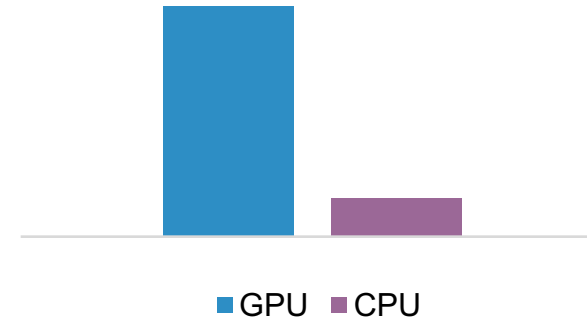


- Graphics Processing Units for General Purpose computations

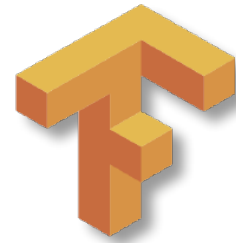
Theoretical peak
throughput
(Tflops, single precision)



Theoretical peak
bandwidth
(GB/s)



Google TensorFlow

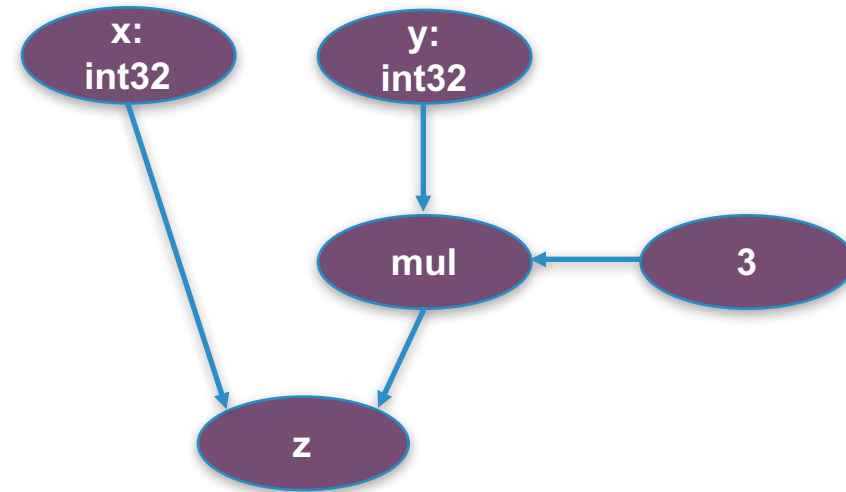


- Library for writing “machine intelligence” algorithms
- Very popular for deep learning and neural networks
- Can also be used for general purpose numerical computations
- Interface in C++ and Python

Numerical dataflow with Tensorflow

```
x = tf.placeholder(tf.int32, name="x")
y = tf.placeholder(tf.int32, name="y")
output = tf.add(x, 3 * y, name="z")
```

```
session = tf.Session()
output_value = session.run(output,
    {x: 3, y: 5})
```



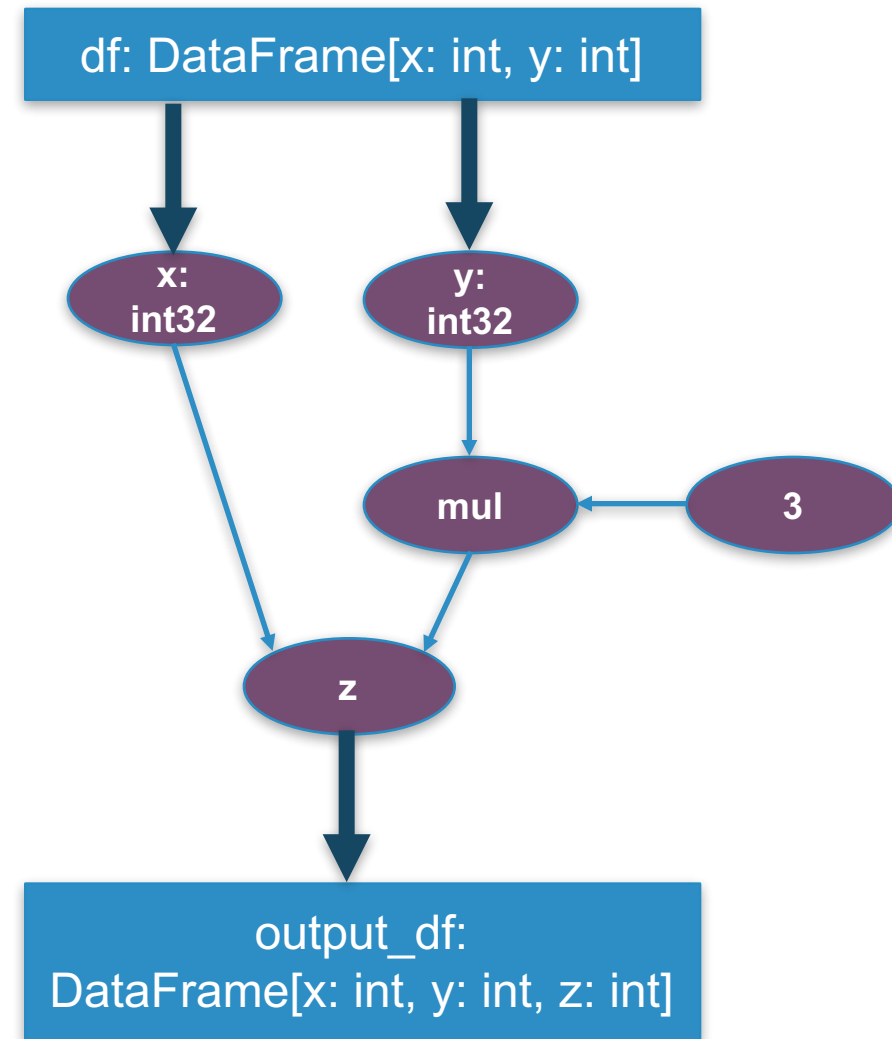
Numerical dataflow with Spark

```
df = sqlContext.createDataFrame(...)

x = tf.placeholder(tf.int32, name="x")
y = tf.placeholder(tf.int32, name="y")
output = tf.add(x, 3 * y, name="z")

output_df = tfs.map_rows(output, df)

output_df.collect()
```

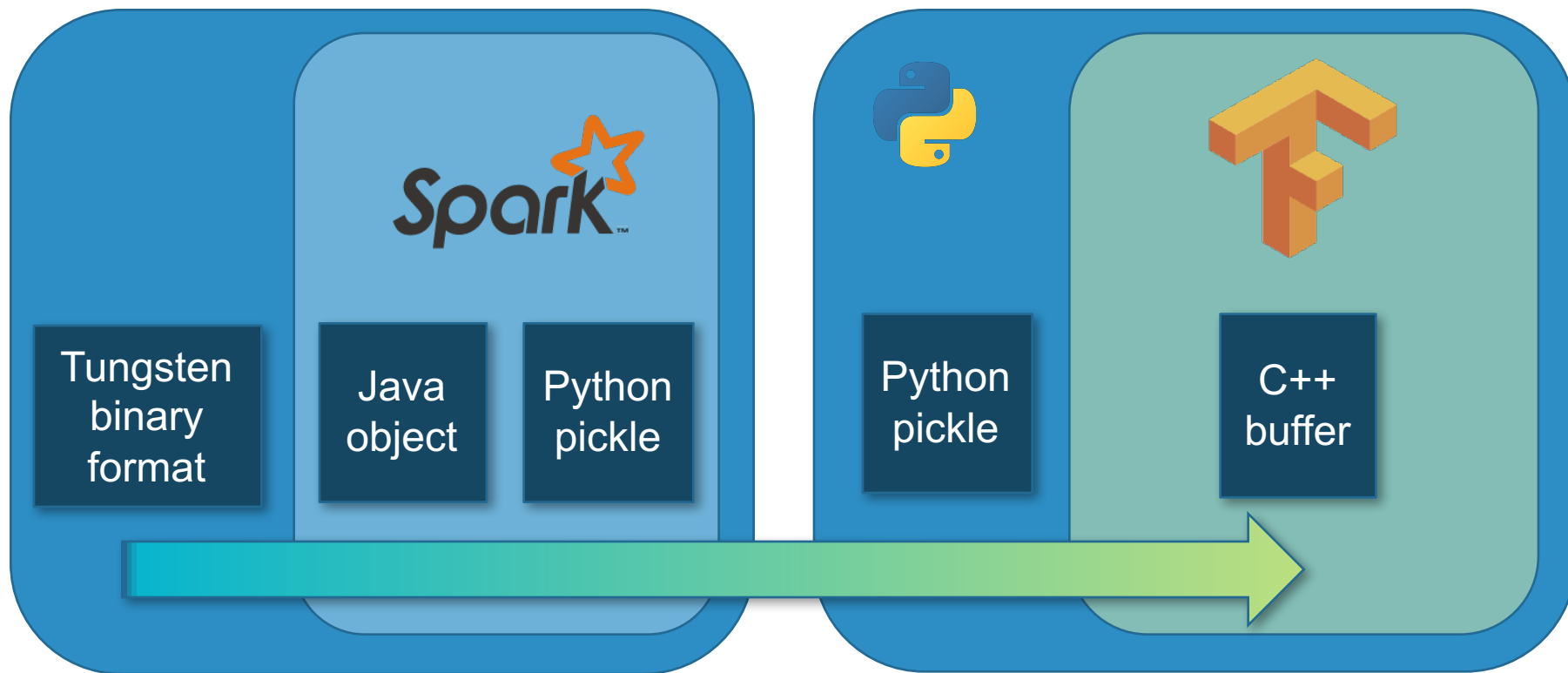


Demo

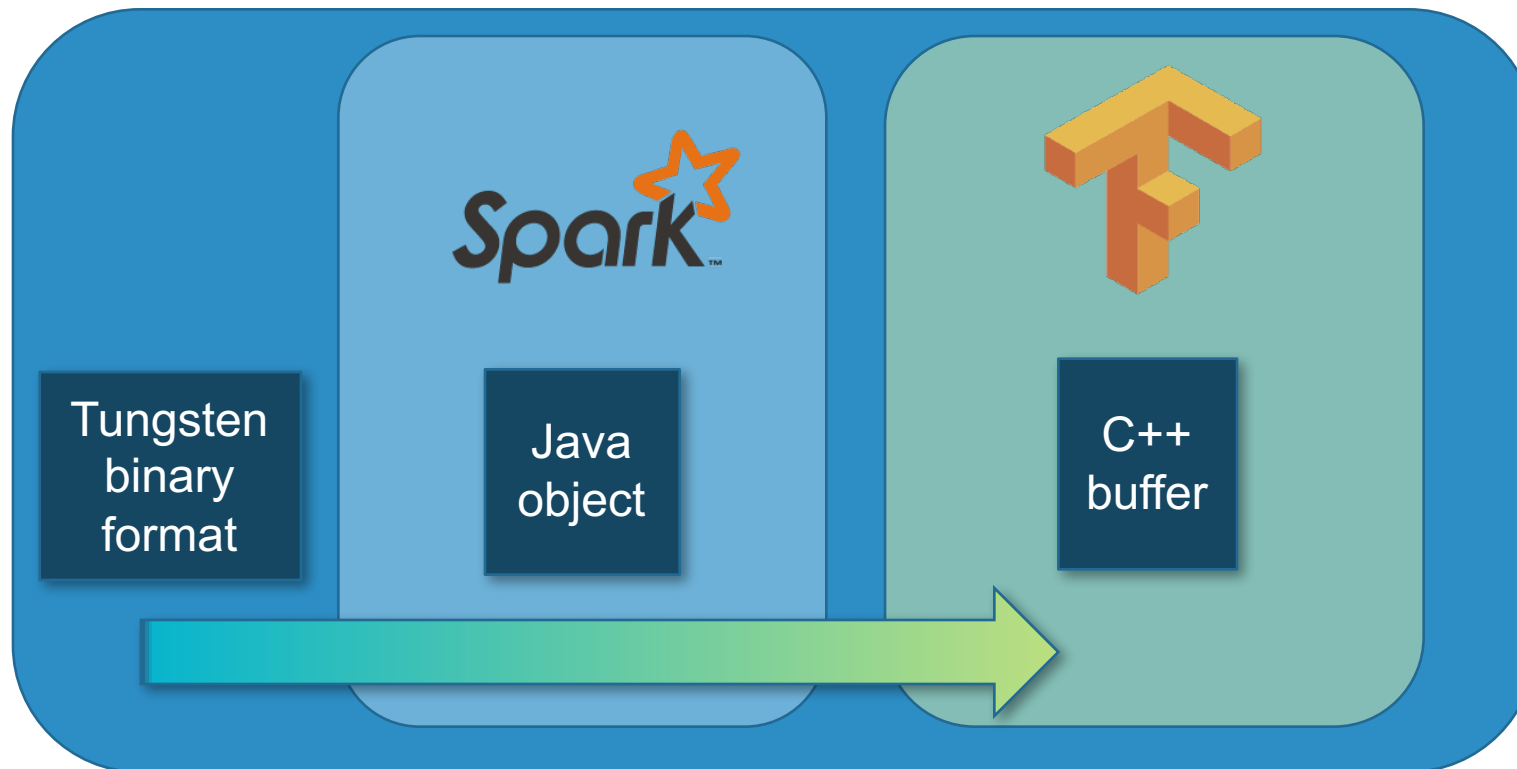
Outline

- Numerical computing with Apache Spark
- Using GPUs with Spark and TensorFlow
- **Performance details**
- The future

It is a communication problem

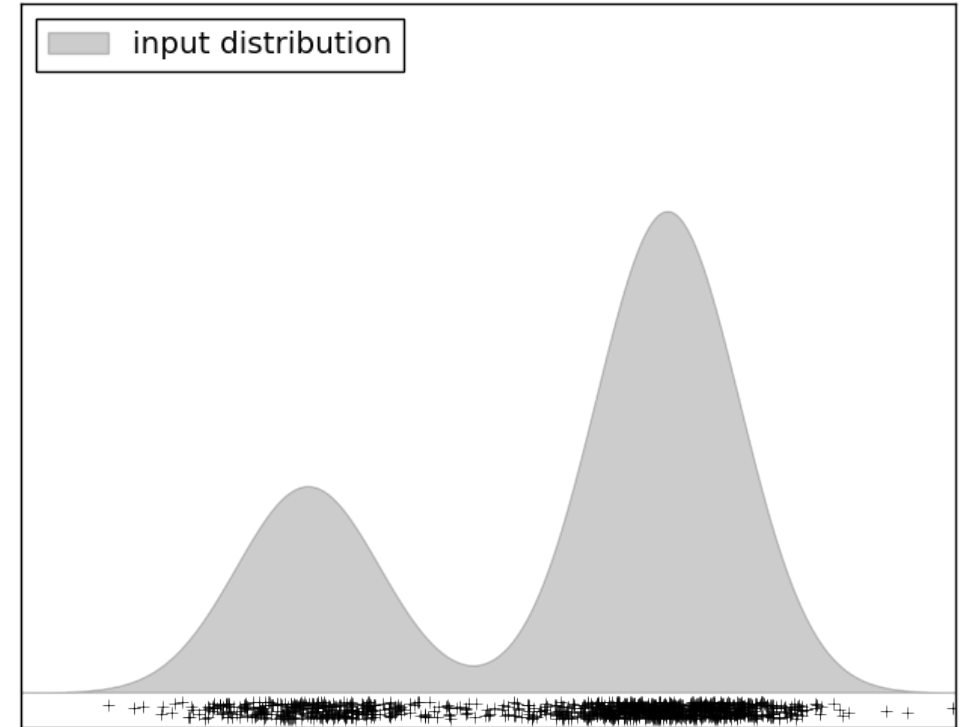


TensorFrames: native embedding of TensorFlow



An example: kernel density scoring

- Estimation of distribution from samples
- Non-parametric
- Unknown bandwidth parameter
- Can be evaluated with goodness of fit



An example: kernel density scoring

- In practice, compute:

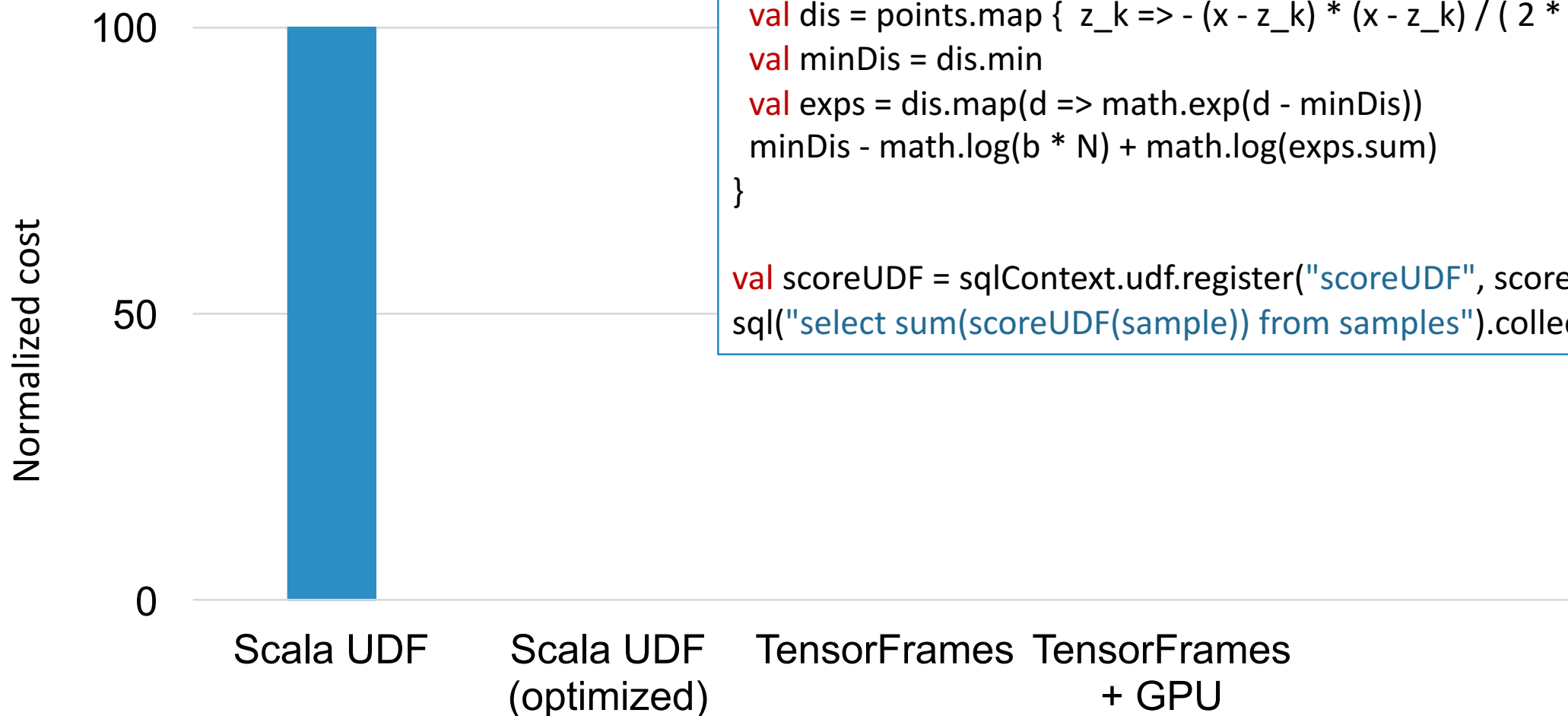
$$\frac{1}{L} \sum_x \text{score}(x)$$

with:

$$\text{score}(x) = \log \left[\frac{1}{N} \sum_{k=1}^N \exp \left(-\frac{(x - z_k)^2}{\sqrt{2\pi}b} \right) \right]$$

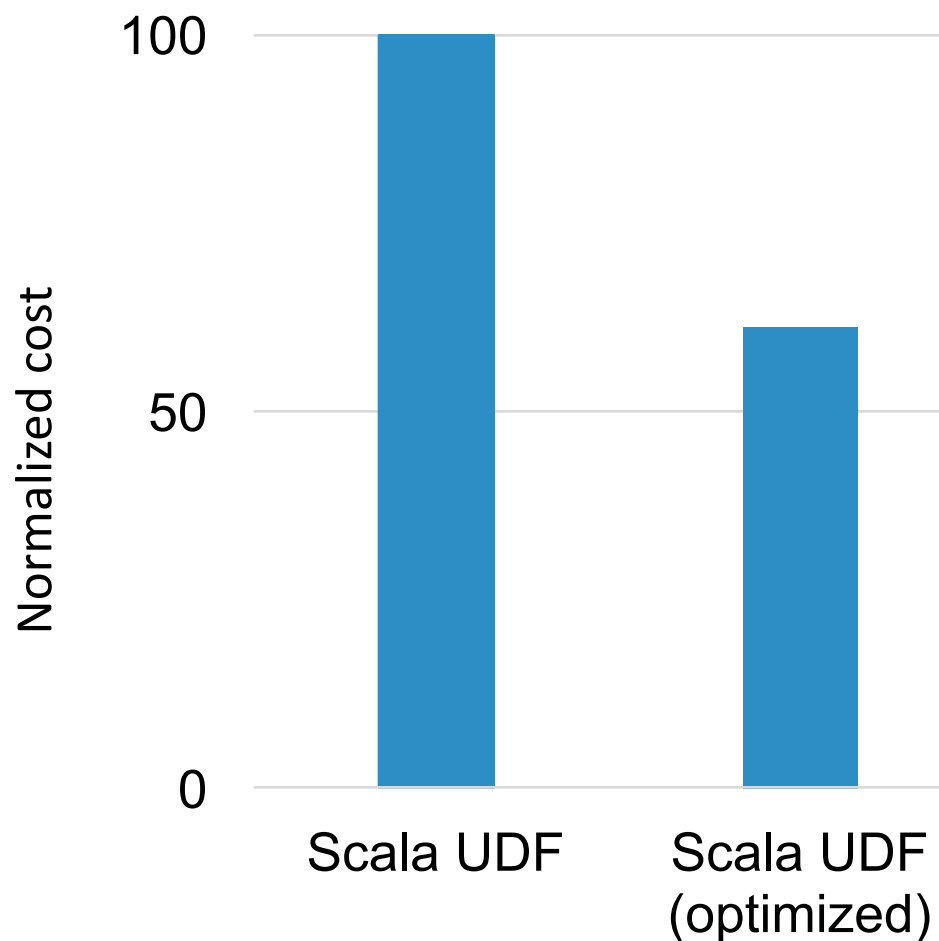
- In a nutshell: a complex numerical function

Speedup



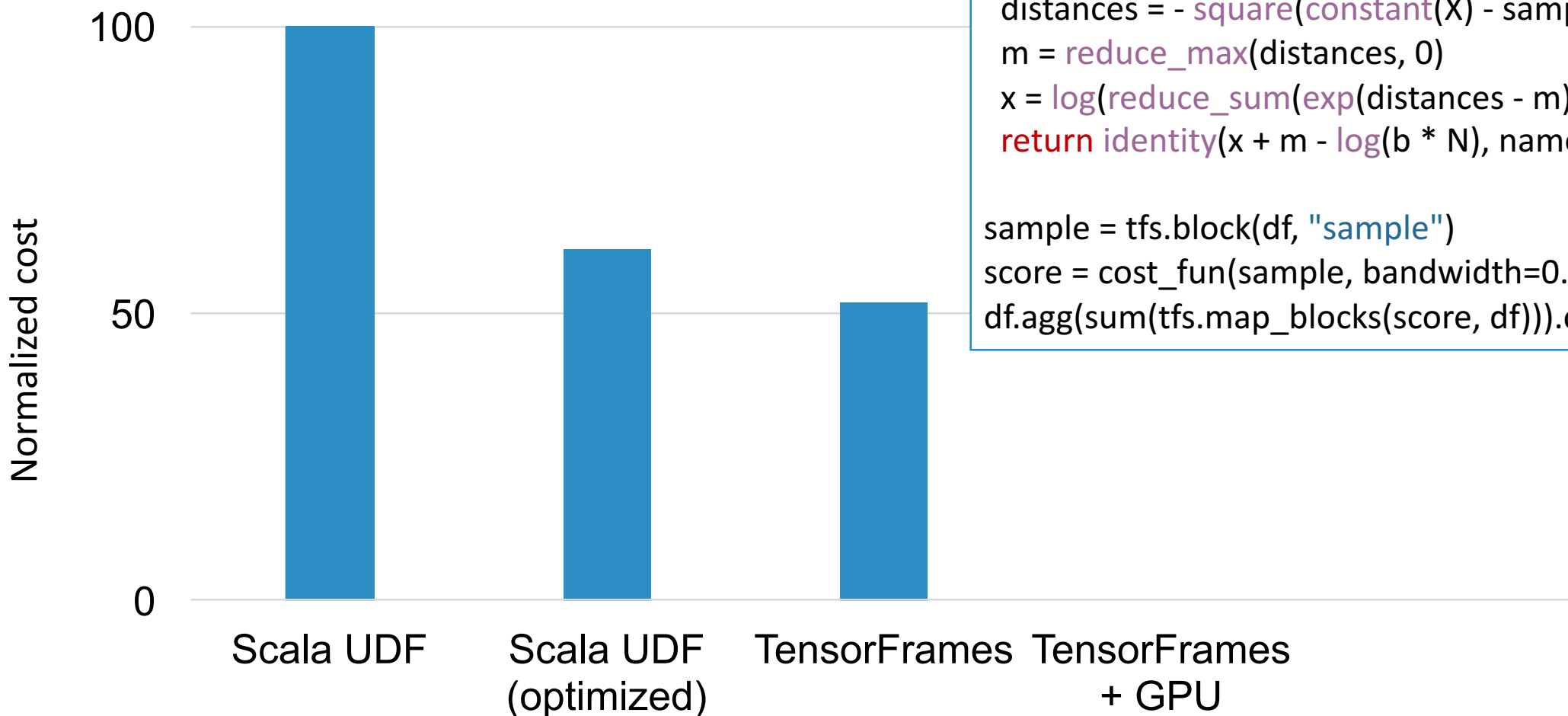
```
def score(x: Double): Double = {  
  val dis = points.map { z_k => - (x - z_k) * (x - z_k) / ( 2 * b * b) }  
  val minDis = dis.min  
  val exps = dis.map(d => math.exp(d - minDis))  
  minDis - math.log(b * N) + math.log(exps.sum)  
}  
  
val scoreUDF = sqlContext.udf.register("scoreUDF", score _)  
sql("select sum(scoreUDF(sample)) from samples").collect()
```

Speedup



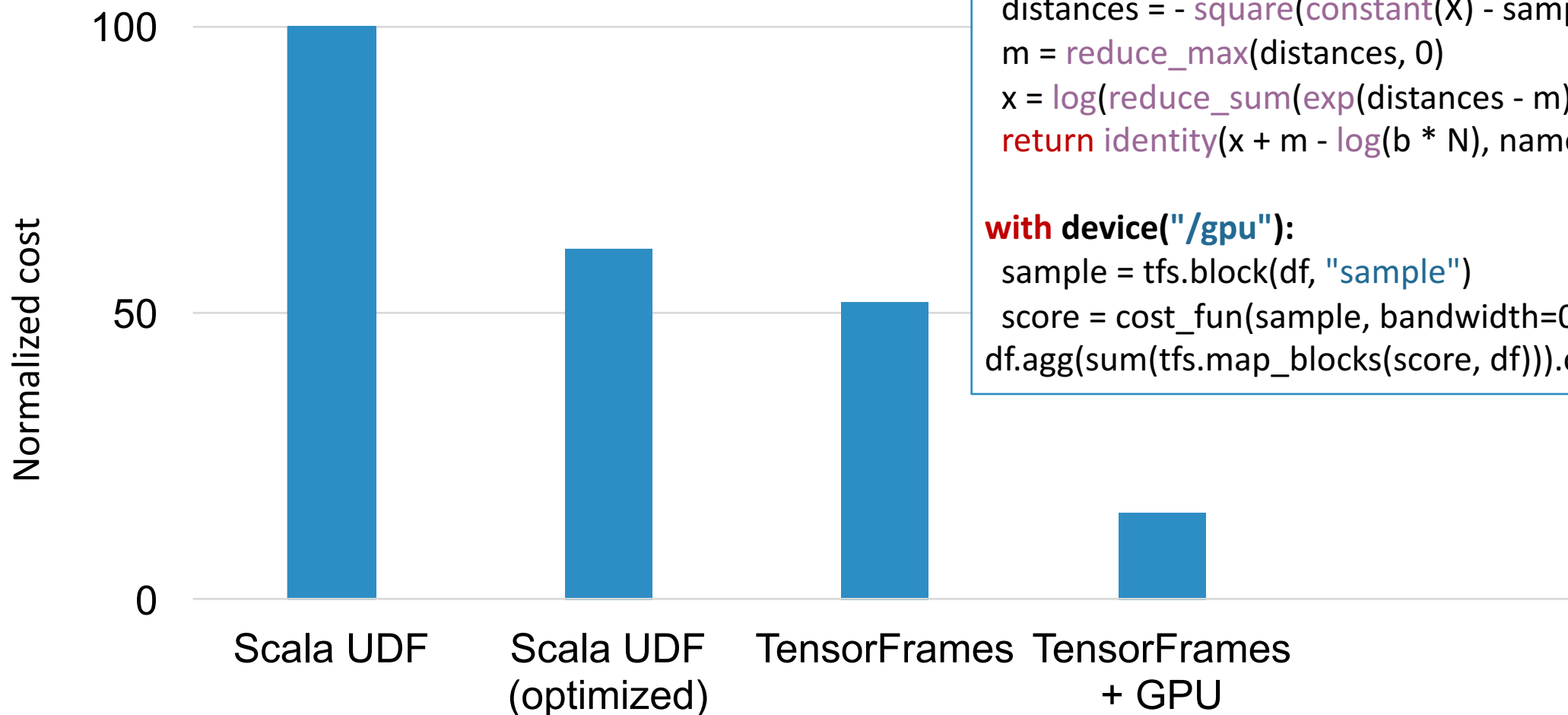
```
def score(x: Double): Double = {  
  val dis = new Array[Double](N)  
  var idx = 0  
  while(idx < N) {  
    val z_k = points(idx)  
    dis(idx) = - (x - z_k) * (x - z_k) / ( 2 * b * b)  
    idx += 1  
  }  
  val minDis = dis.min  
  var expSum = 0.0  
  idx = 0  
  while(idx < N) {  
    expSum += math.exp(dis(idx) - minDis)  
    idx += 1  
  }  
  minDis - math.log(b * N) + math.log(expSum)  
}  
  
val scoreUDF = sqlContext.udf.register("scoreUDF", score _)  
sql("select sum(scoreUDF(sample)) from samples").collect()
```

Speedup



```
def cost_fun(block, bandwidth):  
    distances = - square(constant(X) - sample) / (2 * b * b)  
    m = reduce_max(distances, 0)  
    x = log(reduce_sum(exp(distances - m), 0))  
    return identity(x + m - log(b * N), name="score")  
  
sample = tfs.block(df, "sample")  
score = cost_fun(sample, bandwidth=0.5)  
df.agg(sum(tfs.map_blocks(score, df))).collect()
```

Speedup



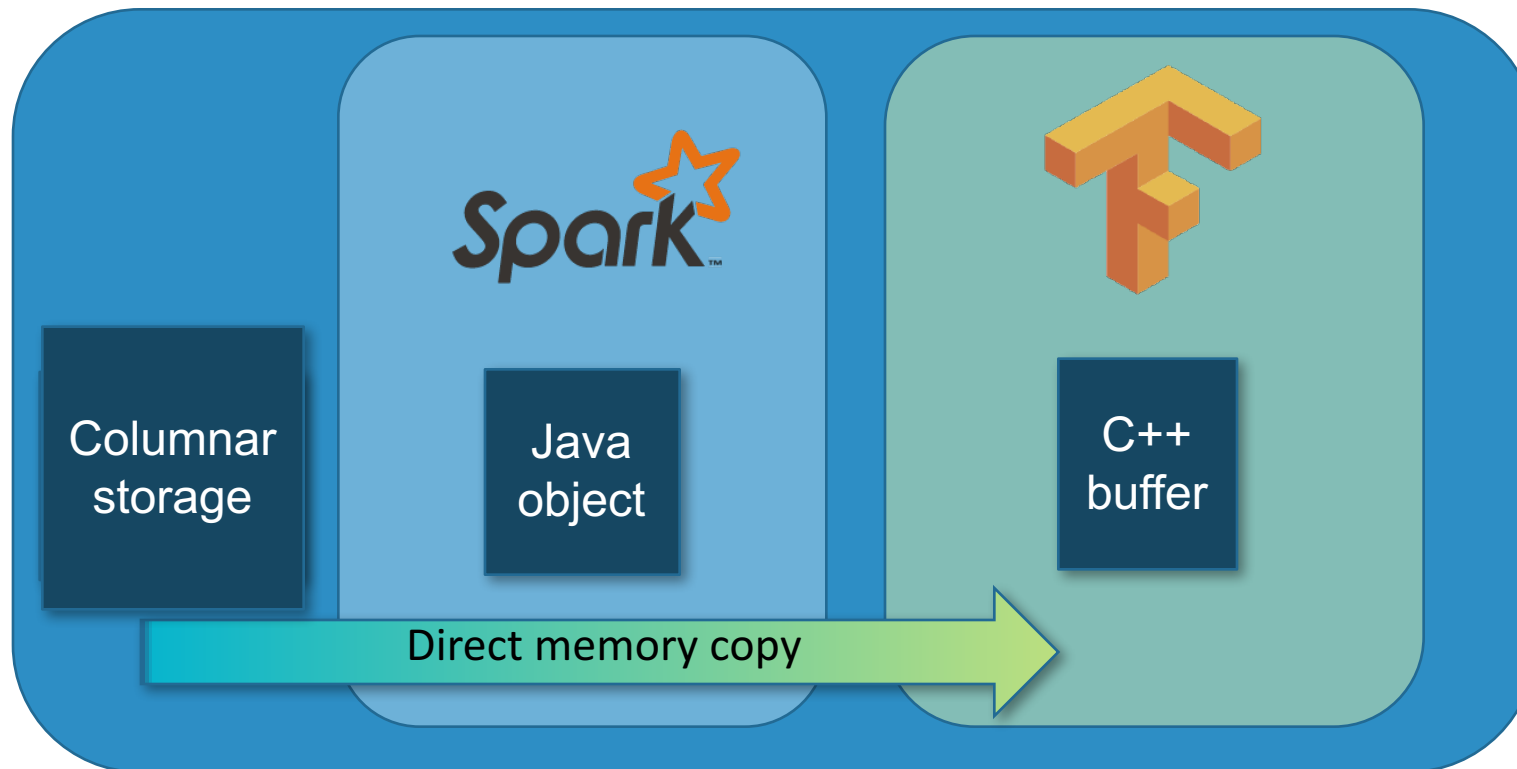
```
def cost_fun(block, bandwidth):  
    distances = - square(constant(X) - sample) / (2 * b * b)  
    m = reduce_max(distances, 0)  
    x = log(reduce_sum(exp(distances - m), 0))  
    return identity(x + m - log(b * N), name="score")
```

```
with device("/gpu"):  
    sample = tfs.block(df, "sample")  
    score = cost_fun(sample, bandwidth=0.5)  
    df.agg(sum(tfs.map_blocks(score, df))).collect()
```


Outline

- Numerical computing with Apache Spark
- Using GPUs with Spark and TensorFlow
- Performance details
- **The future**

Improving communication



The future

- Integration with Tungsten:
 - Direct memory copy
 - Columnar storage
- Better integration with MLlib data types

Recap

- Spark: an efficient framework for running computations on thousands of computers
- TensorFlow: high-performance numerical framework
- Get the best of both with TensorFrames:
 - Simple API for distributed numerical computing
 - Can leverage the hardware of the cluster

Try these demos yourself

- TensorFrames source code and documentation:
github.com/databricks/tensorframes
spark-packages.org/package/databricks/tensorframes
- Demo notebooks available on Databricks
- The official TensorFlow website:
www.tensorflow.org

Thank you.



databricks™

making big data simple