NVIDIA

NVIDIA RAPIDS Accelerator for Apache Spark ML

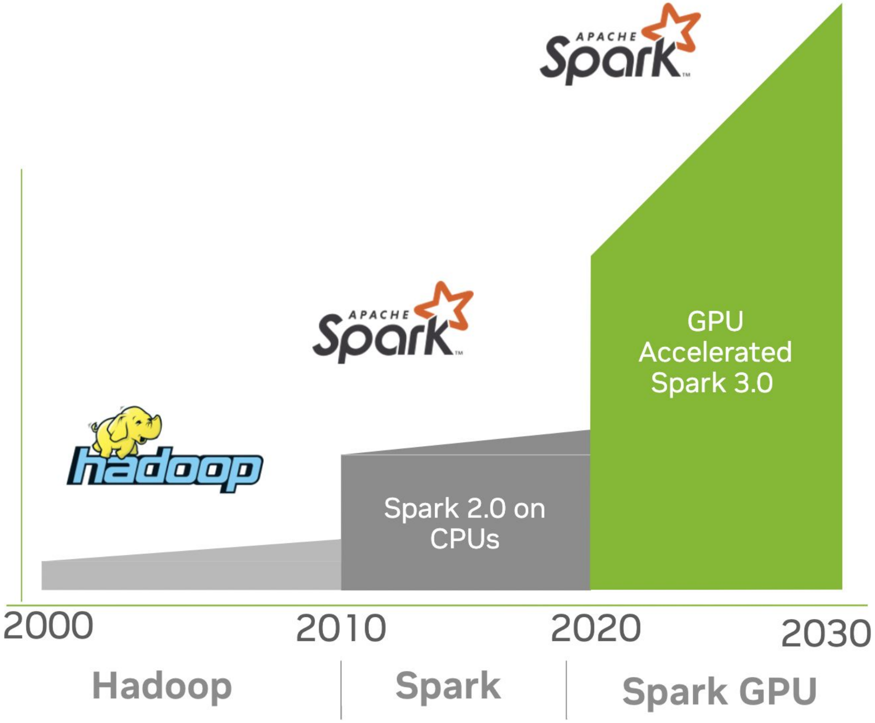
GPU Accelerated Distributed ML in Spark Clusters

Speakers: Jinfeng Li and Erik Ordentlich (joint work with Bobby Wang and Lee Yang)

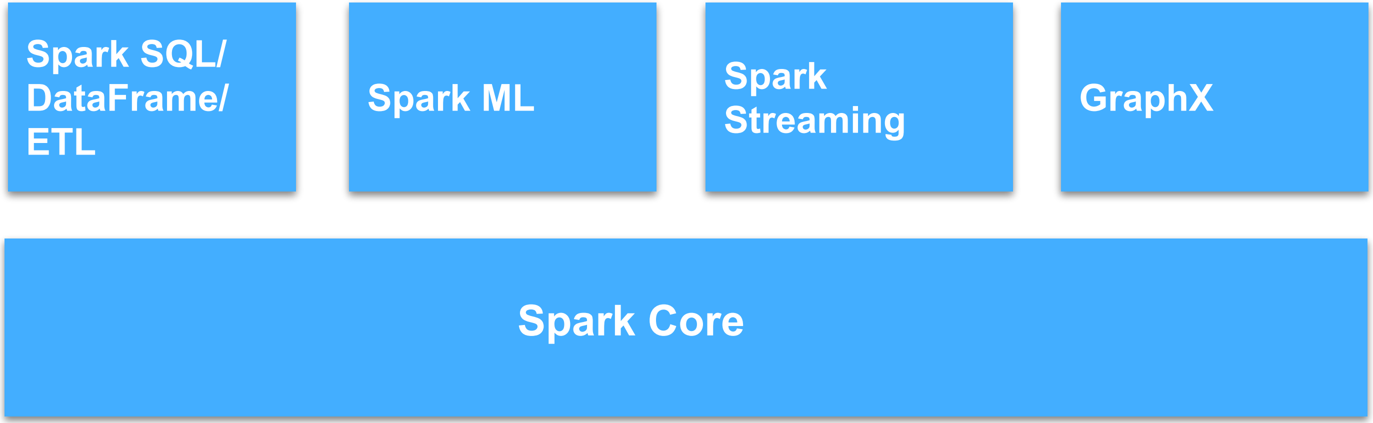
Introduction and Motivation

Scaling Apache Spark with GPUs

Growth in requirement for data processing



Apache Spark Components



Spark SQL/ DataFrame/ ETL

Spark ML

Spark Streaming

GraphX

Spark Core

Apache Spark Components

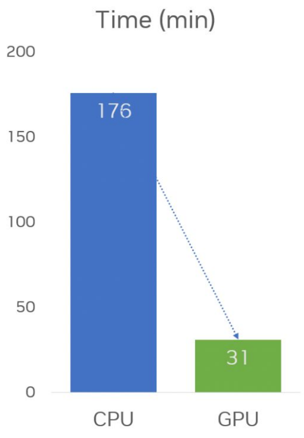
GPU Acceleration

Spark SQL/ DataFrame/ ETL

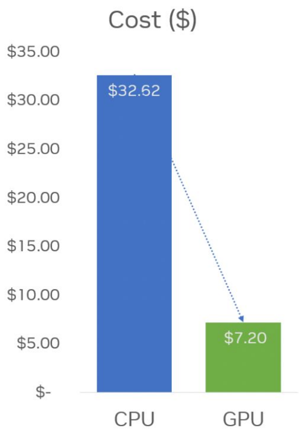
Spark ML

RAPIDS Accelerator for Apache Spark
Spark ETL

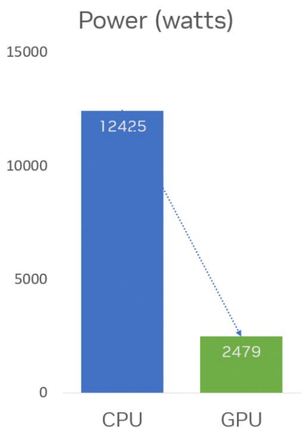
- https://github.com/NVIDIA/spark-rapids
- Provides GPU acceleration for Spark SQL + DataFrames
- NO code change



5.7x speedup



4.5x cost savings



5x more efficient

Additional info:

https://venturebeat.com/data-infrastructure/gtc-2023-nvidia-shares-how-rapids-can-future-proof-apache-spark

https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s52202/

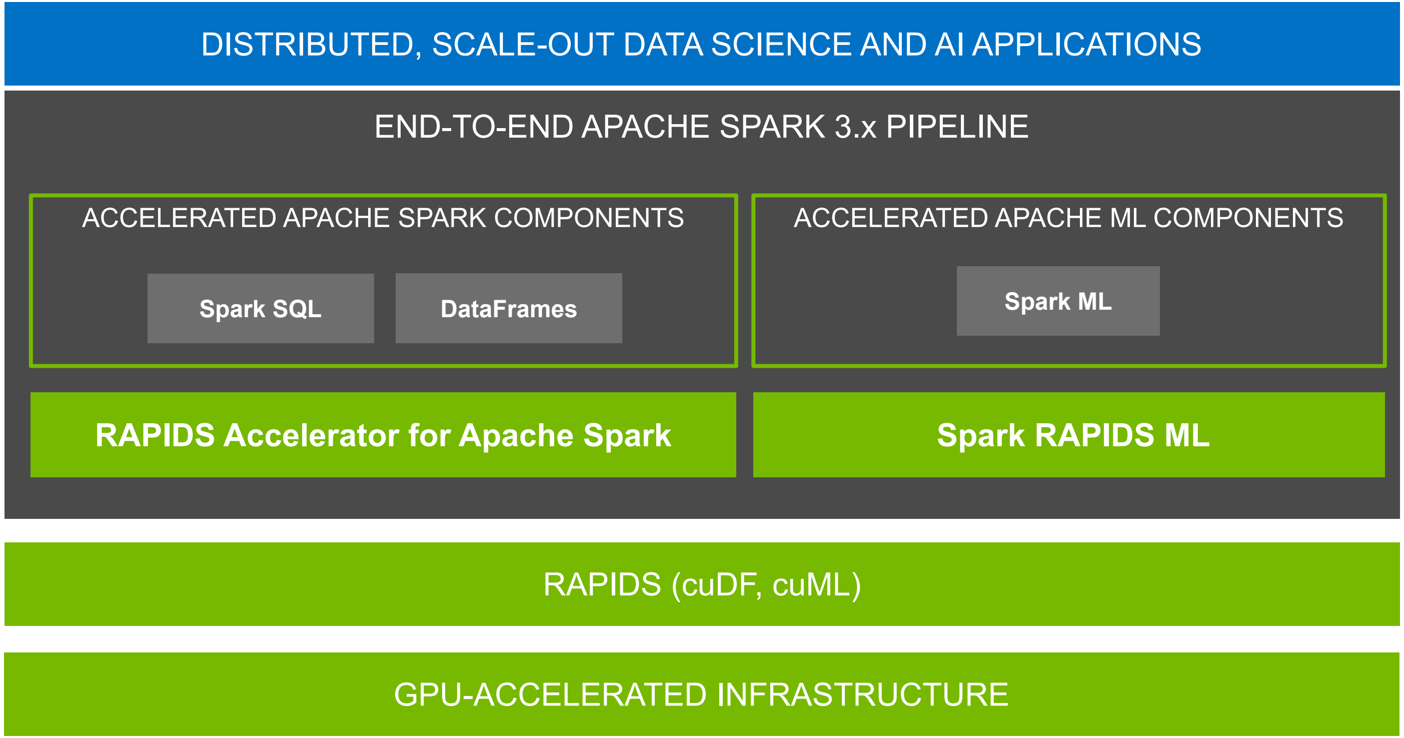
RAPIDS Accelerator for Apache Spark
Spark ML

- Spark ML is a key part of Apache Spark, providing distributed implementations of many ML algorithms, but **CPU only**.

Spark RAPIDS ML

- https://github.com/NVIDIA/spark-rapids-ml
- New pure python open source library to GPU accelerate pySpark ML DataFrame API

RAPIDS Accelerator for Spark ETL & ML



DISTRIBUTED, SCALE-OUT DATA SCIENCE AND AI APPLICATIONS

END-TO-END APACHE SPARK 3.x PIPELINE

ACCELERATED APACHE SPARK COMPONENTS

Spark SQL

DataFrames

ACCELERATED APACHE ML COMPONENTS

Spark ML

RAPIDS Accelerator for Apache Spark

Spark RAPIDS ML

RAPIDS (cuDF, cuML)

GPU-ACCELERATED INFRASTRUCTURE

Spark RAPIDS ML

Initially Supported Algorithms

- KMeans
- PCA
- LinearRegression
  - Ridge
  - ElasticNet
- RandomForestClassifier
- RandomForestRegressor

- Non-Spark ML algorithm:
  - Exact K-Nearest Neighbors
    - similar APIs as Spark ML’s LSH (BucketedRandomProjectionLSH)

Spark RAPIDS ML

Key Objectives

- **API:**
  - Compatible with pyspark.ml DataFrame style apis
  - Requires no application code change
    - Just a package import change
- **Speedup and cost benefits:**
  - Significantly improve on PySpark CPU perf and cost
- **Architecture:**
  - Leverages NVIDIA RAPIDS cuML accelerated ML library

API

pyspark.ml

pyspark.ml

```
from pyspark.ml.clustering import KMeans

kmeans_estm = KMeans()\
.setK(100)\
.setFeaturesCol("features")\
.setMaxIter(30)

kmeans_model = kmeans_estm.fit(pyspark_data_frame)

kmeans_model.write().save("saved-model")

transformed = kmeans_model.transform(pyspark_data_frame)
```

spark_rapids_ml

spark_rapids_ml

```
from spark_rapids_ml.clustering import KMeans

kmeans_estm = KMeans()\
.setK(100)\
.setFeaturesCol("features")\
.setMaxIter(30)

kmeans_model = kmeans_estm.fit(pyspark_data_frame)

kmeans_model.write().save("saved-model")

transformed = kmeans_model.transform(pyspark_data_frame)
```

Benefits

- Spark devs do not need to learn new API

- Easy to migrate existing applications
  - Example: GPU acceleration compatible with Spark ML Pipelines, Crossvalidator, etc.

Performance

Benchmarking

Workload & Data

- estimator.fit(data_df) [i.e. training]
  - data_df read from Parquet format in AWS S3
- Compute intensive synthetic workloads:
  - 1 million rows
  - 3000 dimensional vectors
  - Data available in S3 public bucket.

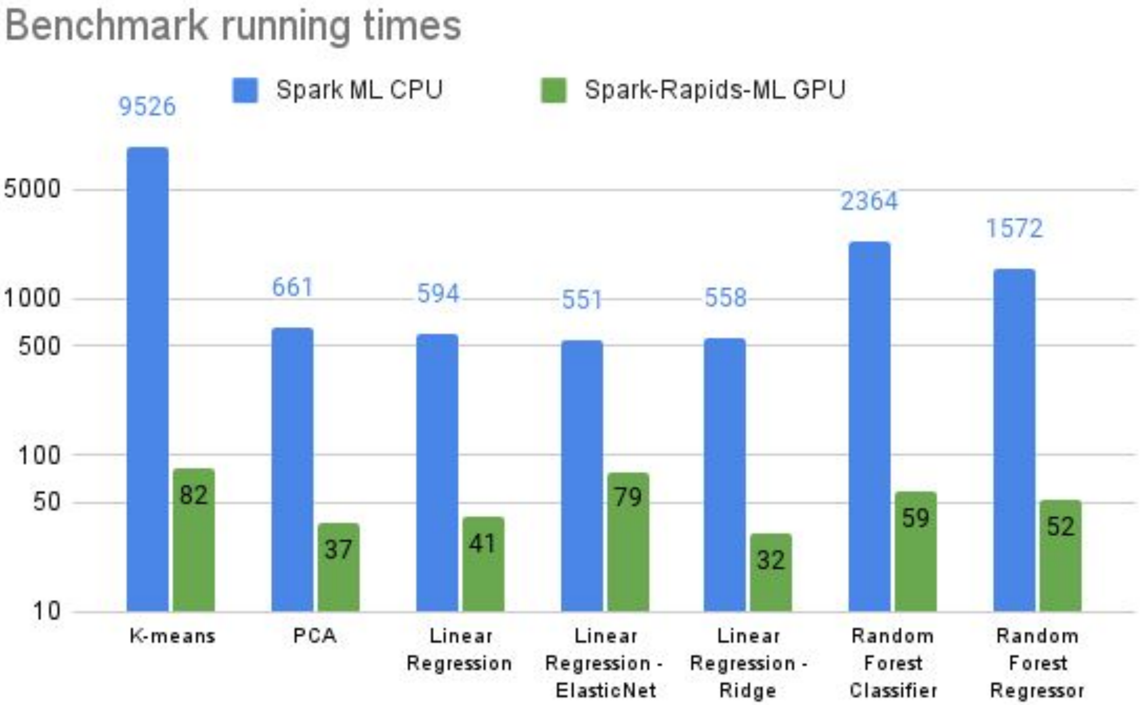
Environment

- Databricks AWS hosted Spark
- 3 node clusters (2 exec, 1 driver)
  - CPU clusters - 32 GB, 8 cores [m4dn.2xlarge]
  - GPU clusters - 32 GB, 8 cores, 1 NVIDIA A10 GPU [g5dn.2xlarge]

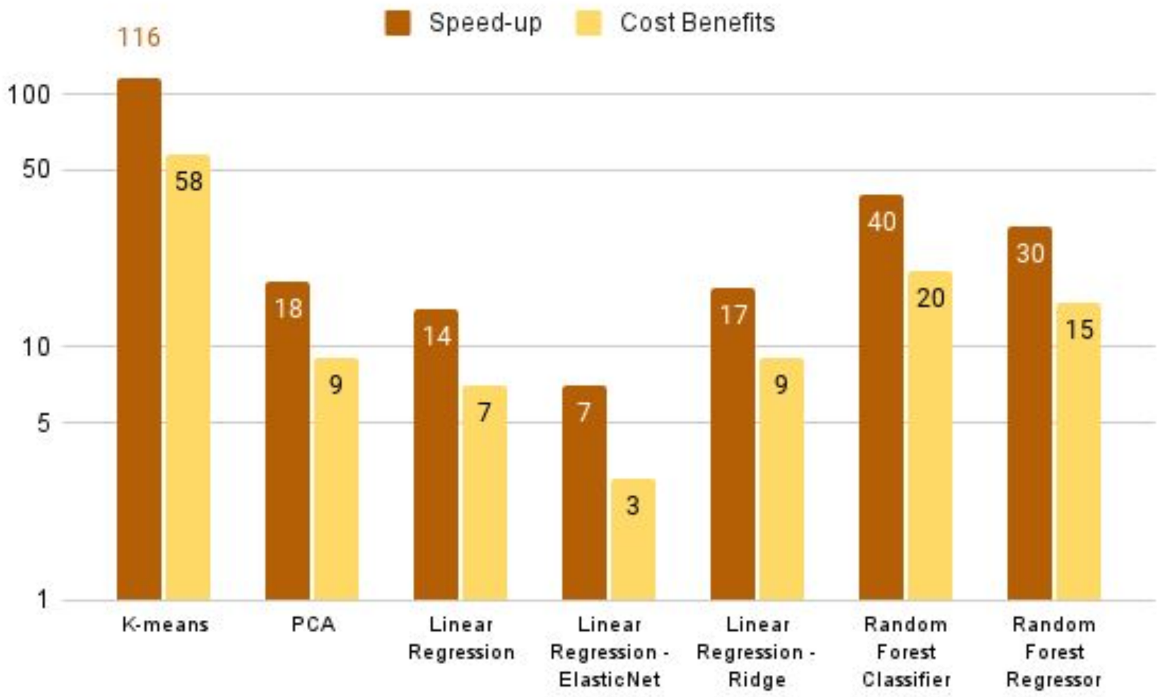
Instructions and scripts to reproduce:
https://github.com/NVIDIA/spark-rapids-ml/tree/main/python/benchmark#databricks

[Repo also has instructions for GCP Dataproc and AWS EMR]

Training/fit time: 10x-100x faster



Cost Benefits: 3x-58x



Architecture

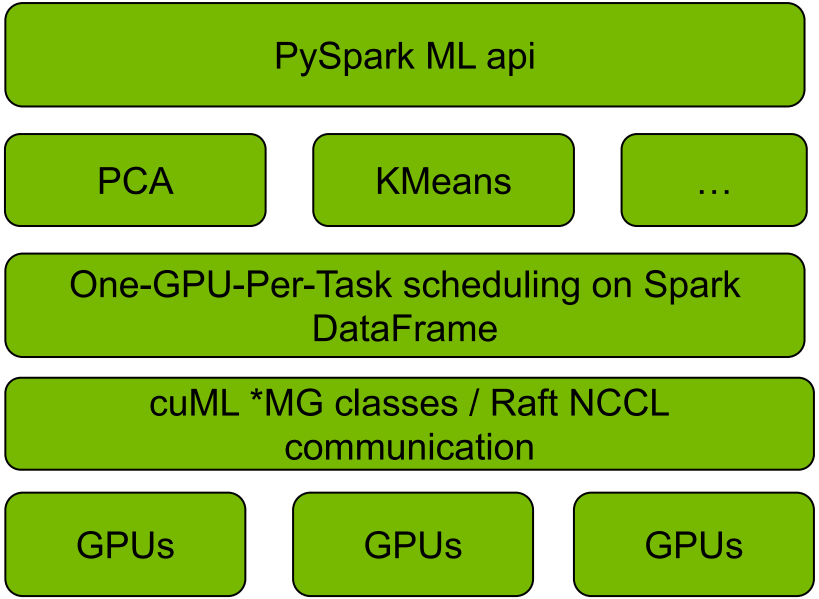
RAPIDS cuML

- cuML is a suite of fast, GPU-accelerated machine learning algorithms
- Python API mirrors Scikit-learn, but 10-50x faster
- C++ and CUDA backend
- Diverse multi-node multi-gpu algorithms
  - GPU-accelerated communication (PCIe, NVLINK, InfiniBand Verbs)
  - DASK API

Clustering	KMeans, DBSCAN
Dimensionality Reduction	PCA, SVD, UMAP
Linear Model	Linear Regression, Lasso, Ridge, ElasticNet, Naive Bayes
Nonlinear Model	Random Forest Classifier/Regressor, KNN Classifier/Regressor
Other	NearestNeighbors

https://github.com/rapidsai/cuml

Distributed cuML integration


PySpark ML api
PCA
KMeans
…
One-GPU-Per-Task scheduling on Spark DataFrame
cuML *MG classes / Raft NCCL communication
GPUs
GPUs
GPUs

- Use PySpark ML api for customizing algorithmic parameters and data
- Create wrappers of cuML Multi-GPU classes and raft NCCL communication layer
- Use PySpark APIs (task-per-gpu scheduling, repartition, mapInPandas, barrier, and broadcast) to setup the *MG cluster with NCCL and process data

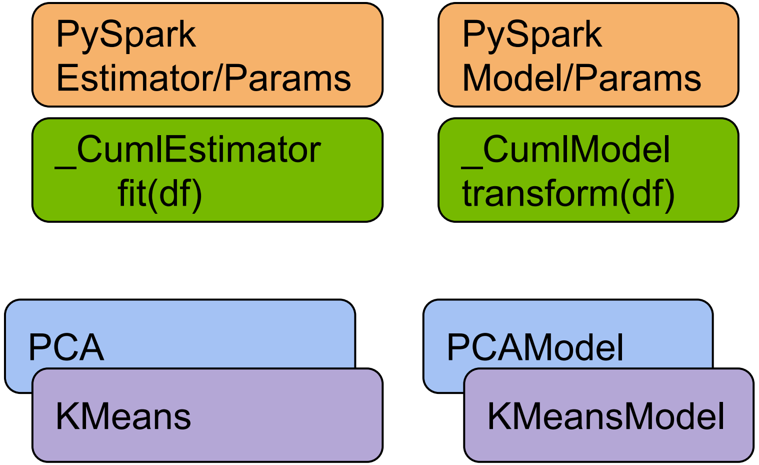
Distributed cuML integration

```
class KMeans:
    □
    def fit(dataset):
        cuml_params = map_to_cuml(spark_ml_params)
        model = ( dataset.repartition(num_workers)
                              .mapInPandas(train_udf, …)
                              .rdd.barrier()
                                □
                              .collect()[0] )
        return model
        □
```

Distributed cuML integration

```
def train_udf(pandas_dfs):
    numpy_arrays = convert(pandas_dfs)
    comms_handle = bootstrap_comms(pyspark.BarrierTaskContext)
    model = ( cuml.....KMeansMG(cuml_params, comms_handle)
                               .fit(numpy_arrays) )
    if worker==0:
        return model
    else:
        return None
```

Class architecture



- Extend the PySpark Estimator/Model to integrate into PySpark ML (eg, Pipeline, tuning)
- Automatically map PySpark algorithm parameters to cuML parameters.
- Support estimator/model persistence as in PySpark.

- The “fit” in _CumlEstimator handles the common steps like Data preparation, Parameter Validation, Gpu resources, cuML Context including Handle and NCCL, and Model construction and so on.
- The “transform” in _CumlModel similarly handles the corresponding common “transform” steps.

Usage

Installation

(Executor and Driver Environments)

- RAPIDS dependencies

```
conda create -n rapids-23.04 -c rapidsai -c nvidia -c conda-forge \
cuml=23.04 python=3.8 cudatoolkit=11.x
```

OR

```
pip install cudf-cu11 cuml-cu11 --extra-index-url=https://pypi.nvidia.com
```

- Spark RAPIDS ML

```
[ conda activate rapids-23.04 ]
pip install spark-rapids-ml
```

Submitting Application

```
spark-submit --master ${SPARK_MASTER} --deploy-mode cluster --num-executors 2
--executor-memory 20g --driver-memory 10g \
        --conf spark.task.resource.gpu.amount=1 \
        --conf spark.executor.resource.gpu.amount=1 \
        --conf
spark.executor.resource.gpu.discoveryScript=./getGpusResources.sh \
        --conf
spark.files=${SPARK_HOME}/examples/src/main/scripts/getGpusResources.sh \
        spark-rapids-ml-kmeans-application.py
```

https://eordentlich.github.io/spark-rapids-ml/notebooks/kmeans-with-output-databricks.html

Demo

Take aways

- Open source with Apache v2 license
- No application code change
- 3x to 50x cost benefits
- Can be run on-prem and on all CSPs

Apache Spark 3.x

Databricks

Google Cloud Dataproc

Amazon EMR

Links

1.

https://github.com/NVIDIA/spark-rapids-ml

Tech Blog