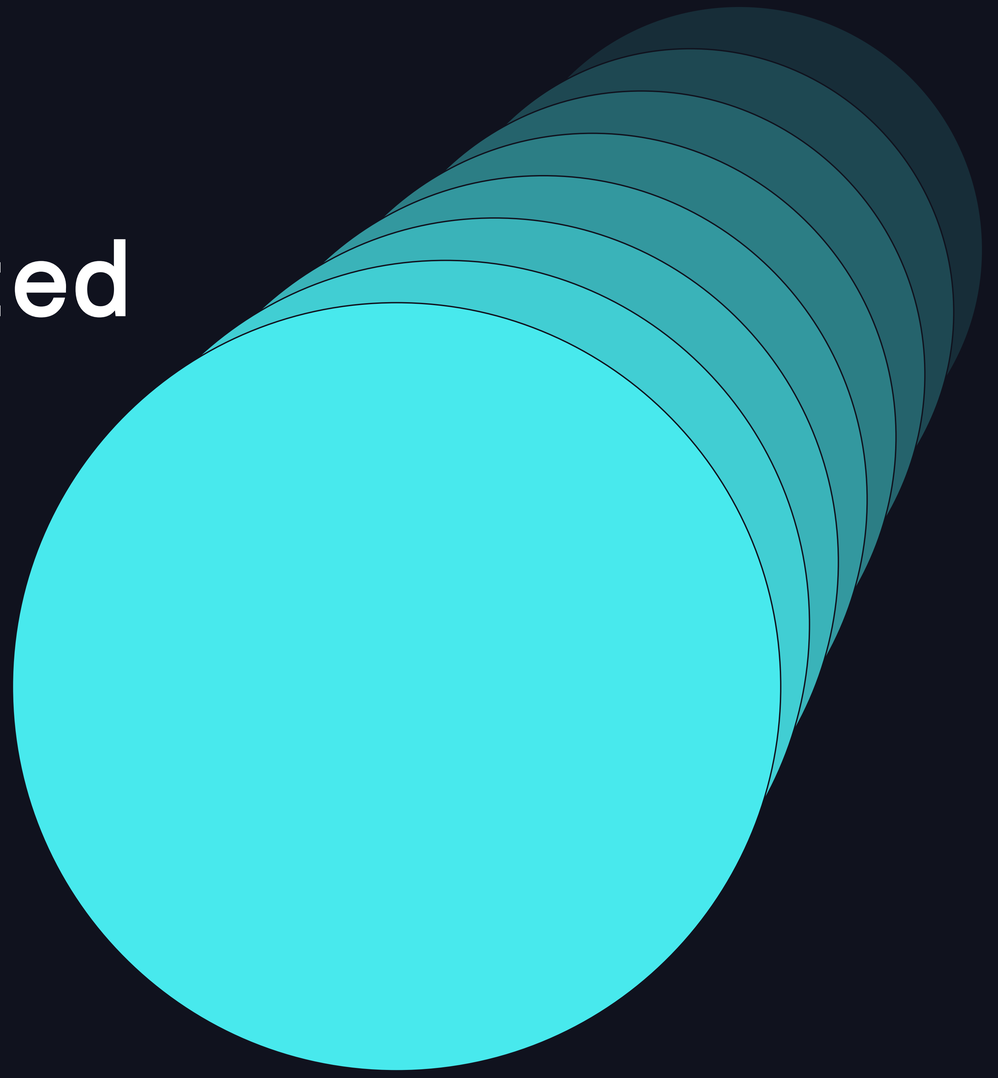
DATA+AI SUMMIT BY databricks

Spark RAPIDS ML: GPU Accelerated Distributed ML in Spark Clusters

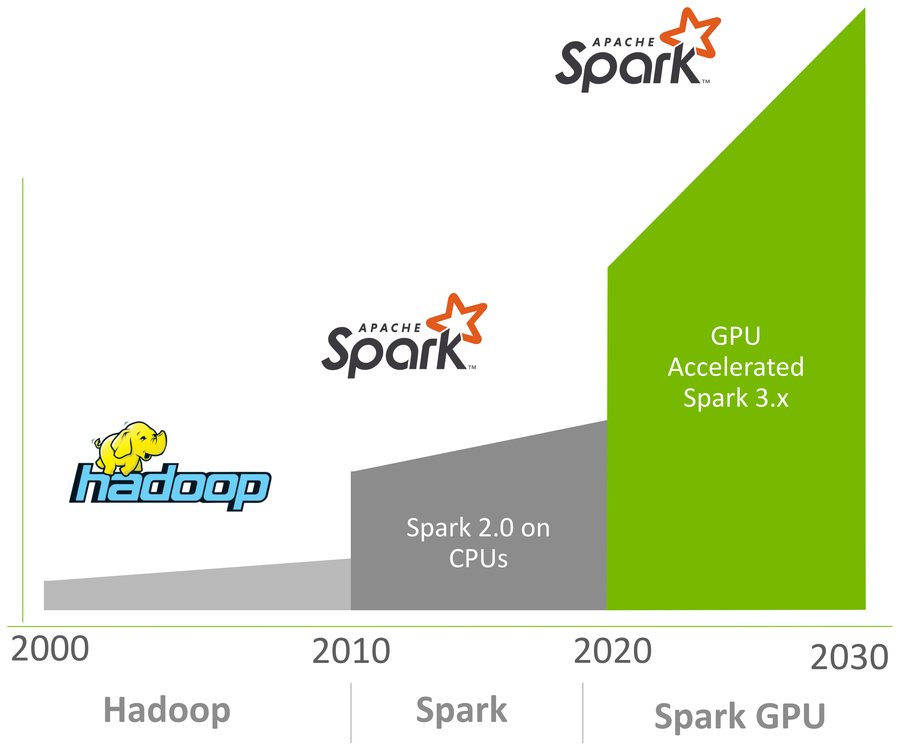


Jinfeng Li and Erik Ordentlich
NVIDIA
June 2024

Scaling Apache Spark With GPUs

RAPIDS Accelerator for Apache Spark

Growth in Requirement for Data Processing



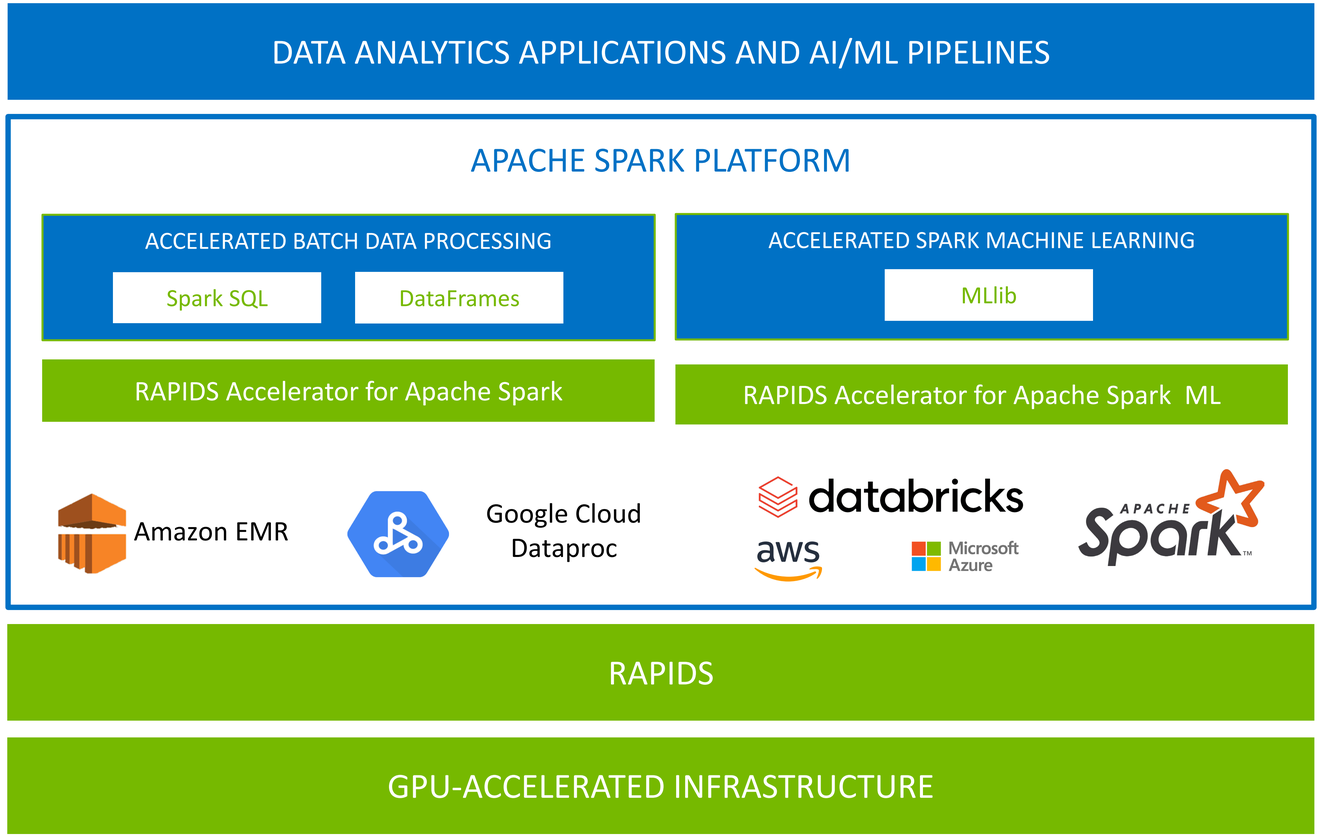
Key Spark 3 innovations

Columnar processing support in the Catalyst query optimizer – allows efficient GPU acceleration

GPU-aware scheduling of executors with a specified number of GPUs and how many GPUs for each task

NVIDIA RAPIDS Accelerator

Key technologies for GPU acceleration



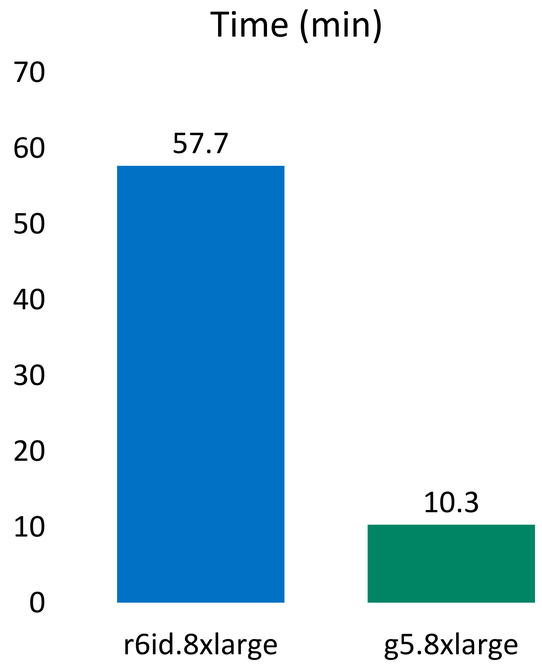
No Query Changes

- Add jar to classpath and set spark.plugins config
- No change to SQL and DataFrame code
- Compatible with PySpark, SparkR, Java, Scala and other DataFrame-based APIs
- Seamless fallback to CPU for unsupported operations

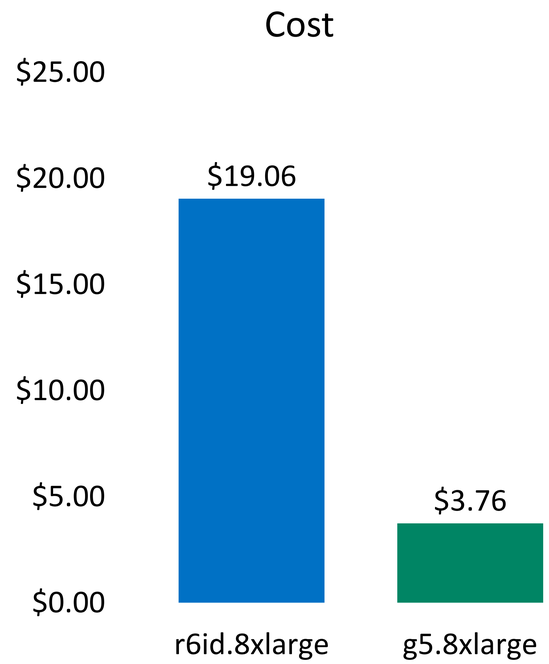
```
spark.sql( """

    SELECT
        o_order_priority
        count(*) as order_count
    FROM
        orders
    WHERE
        o_orderdate >= DATE '1993-07-01'
        AND o_orderdate < DATE '1993-07-01' +
interval '3' month
        AND EXISTS (
            SELECT
                *
            FROM lineitem
            WHERE
                l_orderkey = o_orderkey
                AND l_commitdate < l_receiptdate
             )
    GROUP BY
        o_orderpriority ORDER BY o_orderpriority

""" ).show()
```

NVIDIA Decision Support Benchmark 3TB, AWS EC2

Apache Spark 3.4.1, RAPIDS Spark release 24.04



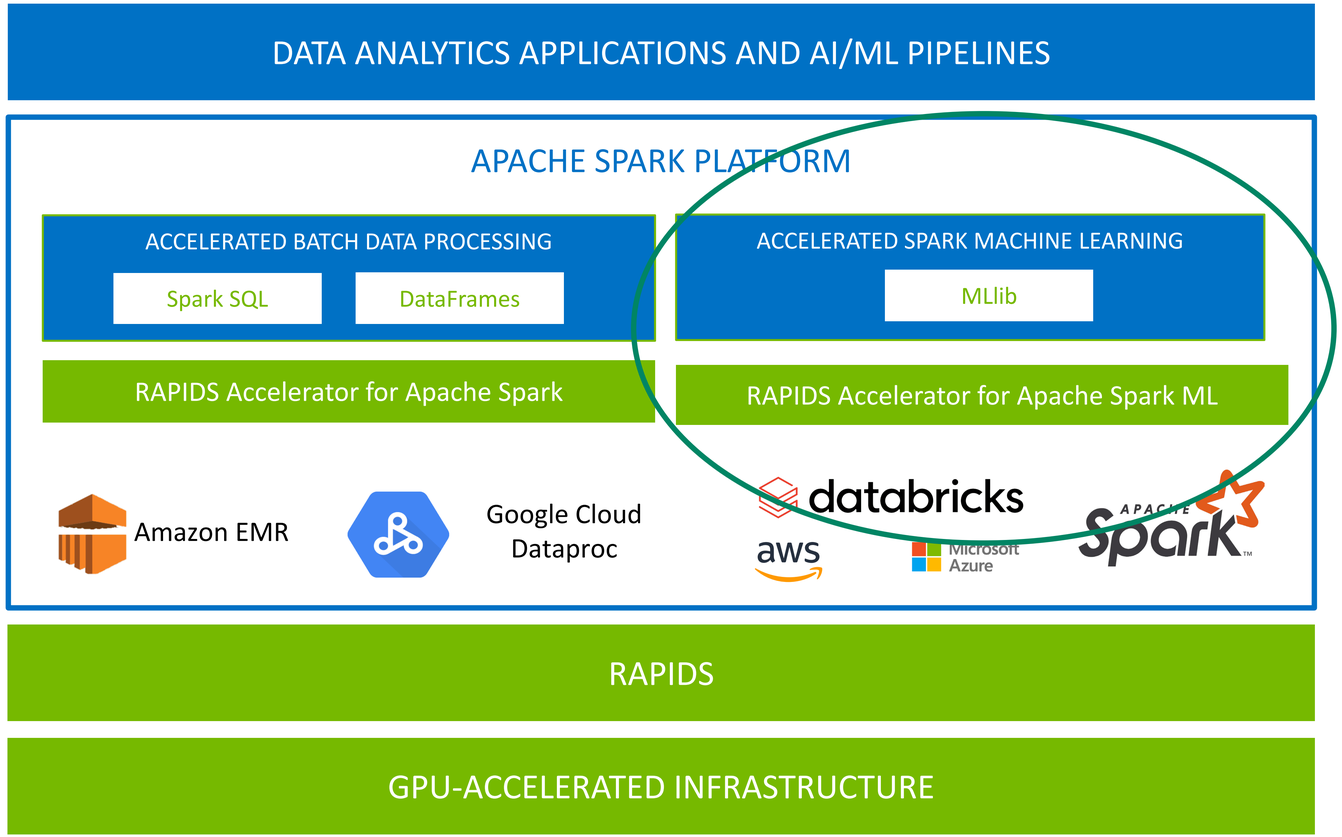
5.5x faster



80% cost savings

NVIDIA RAPIDS Accelerator

Key technologies for GPU acceleration

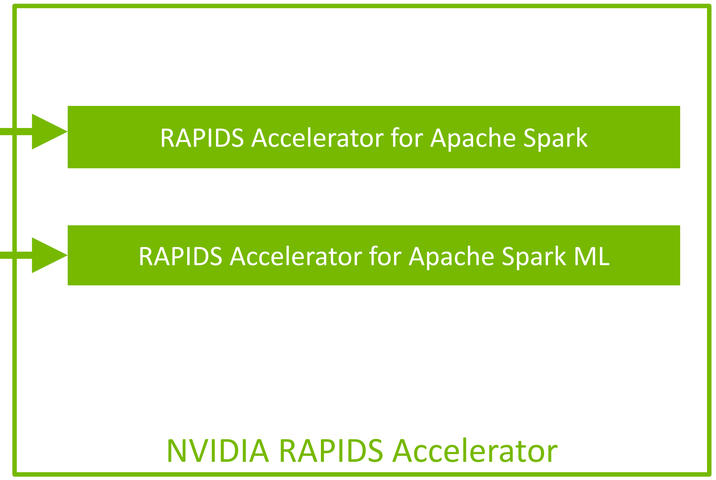


NVIDIA / **spark-rapids-ml**

RAPIDS Spark ML

Motivation

```
spark.sql("SELECT * FROM range(10) where id > 7")

df.join(df2, 'name').select(df.weight, df2.height)

pyspark.ml.clustering.KMeans().fit(df)
```



Package Import Change

- Compatible with pyspark.ml DataFrame APIs
- Requires no application code change
- Package import change

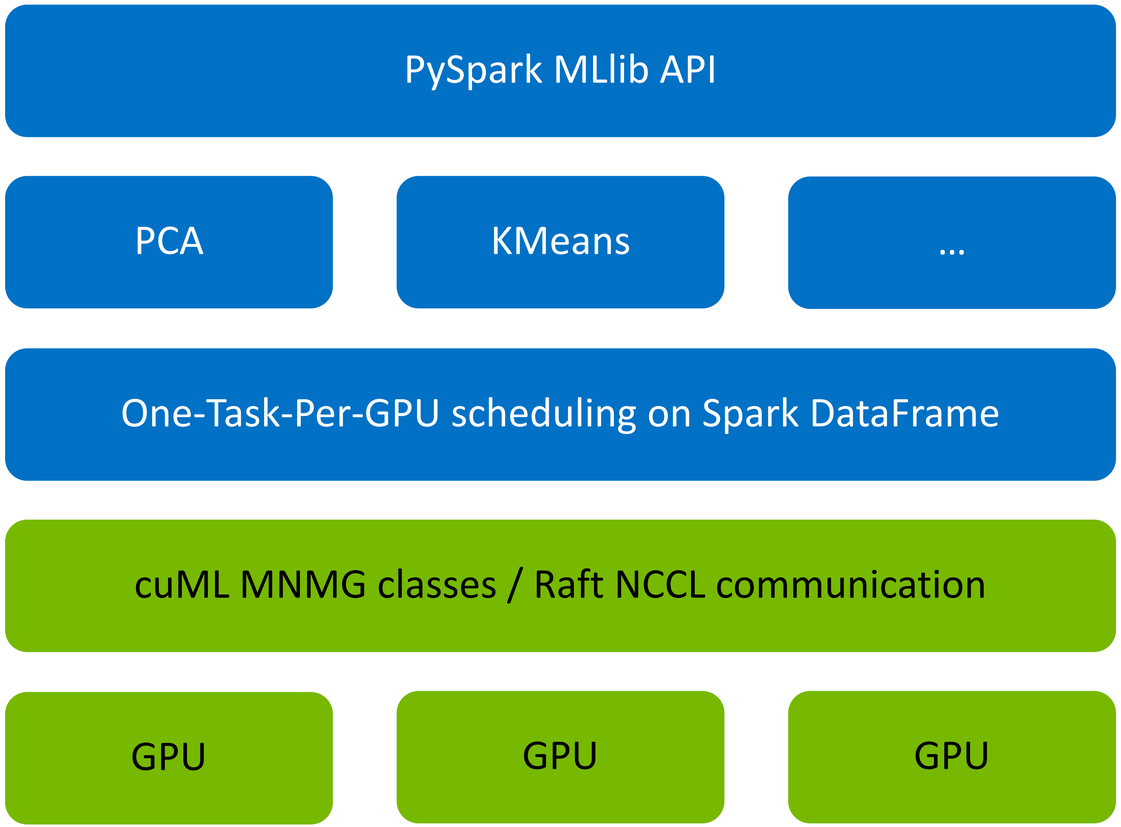
```
from pyspark.ml.clustering import Kmeans

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)
```

Package Import Change

- Compatible with pyspark.ml DataFrame APIs
- Requires no application code change
- Package import change for acceleration

```
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)
```

Distributed cuML Integration



Map PySpark MLlib API calls to cuML for supported Algos
Use PySpark APIs (task-per-gpu scheduling, repartition, mapInPandas, barrier, and broadcast) to setup the cuML Multi-Node Multi-GPU cluster

Process data on GPUs using NCCL for communication

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
```

RAPIDS Spark ML

Supported Algorithms

In Spark MLlib only	In Spark MLlib and cuML	In cuML only
CrossValidator	K-Means	Exact k-NN
	Linear Regression	UMAP
	Logistic Regression	DBSCAN
	PCA	IVF-Flat Approximate k-NN
	Random Forest Classifier	
	Random Forest Regressor	

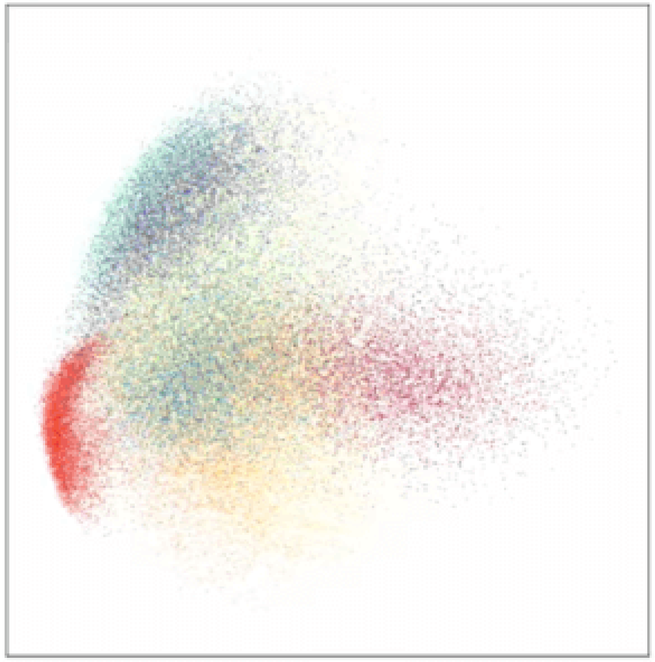
Example: MLlib-like API for GPU Exact k-NN

```
>>> from spark_rapids_ml.knn import NearestNeighbors
>>> topk = 2
>>> gpu_knn = NearestNeighbors().setInputCol("features").setIdCol("id").setK(topk)
>>> gpu_model = gpu_knn.fit(data_df)
>>> (_, _, knn_df) = gpu_model.kneighbors(query_df)
>>> knnjoin_df = gpu_model.exactNearestNeighborsJoin(query_df, distCol="EuclideanDistance")
>>> knnjoin_df.show()
+--------------+--------------+-----------------+
|       item_df|      query_df|EuclideanDistance|
+--------------+--------------+-----------------+
|{1, [2.0, 2.0]}|{3, [1.0, 1.0]}|        1.4142135|
|{0, [1.0, 1.0]}|{3, [1.0, 1.0]}|              0.0|
|{2, [3.0, 3.0]}|{4, [3.0, 3.0]}|              0.0|
|{1, [2.0, 2.0]}|{4, [3.0, 3.0]}|        1.4142135|
+--------------+--------------+-----------------+
```

UMAP

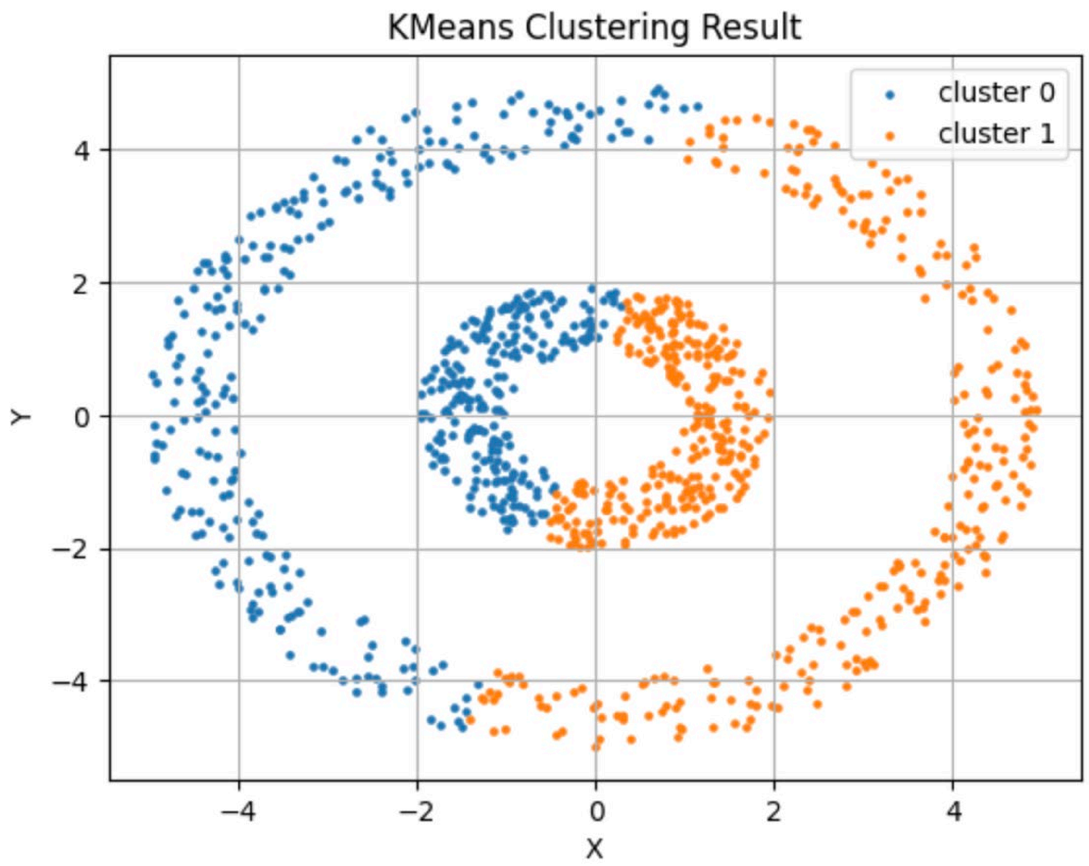
MINIST data embedded into two dimensions by Spark Rapids ML

PCA



DBSCAN

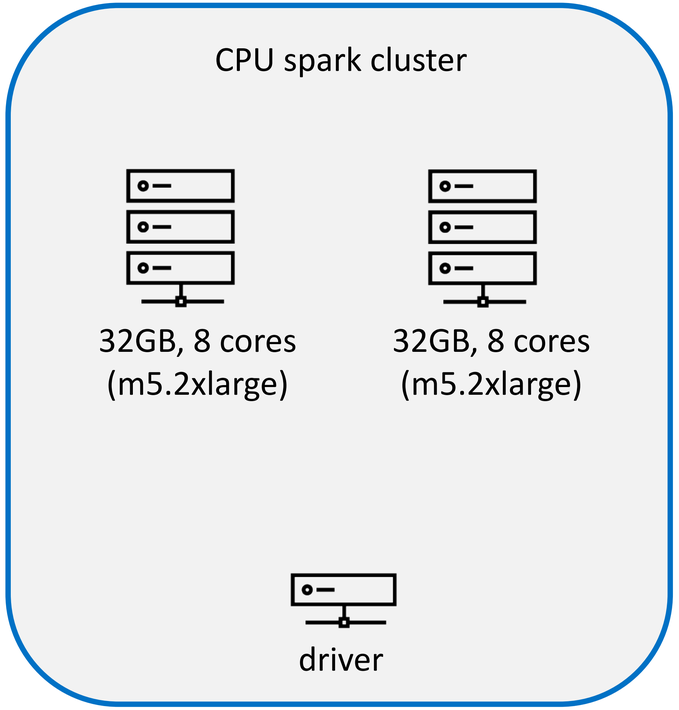

DBSCAN Clustering Result
cluster 0
cluster 1
Y
X
4
2
0
−2
−4
−4
−2
0
2
4

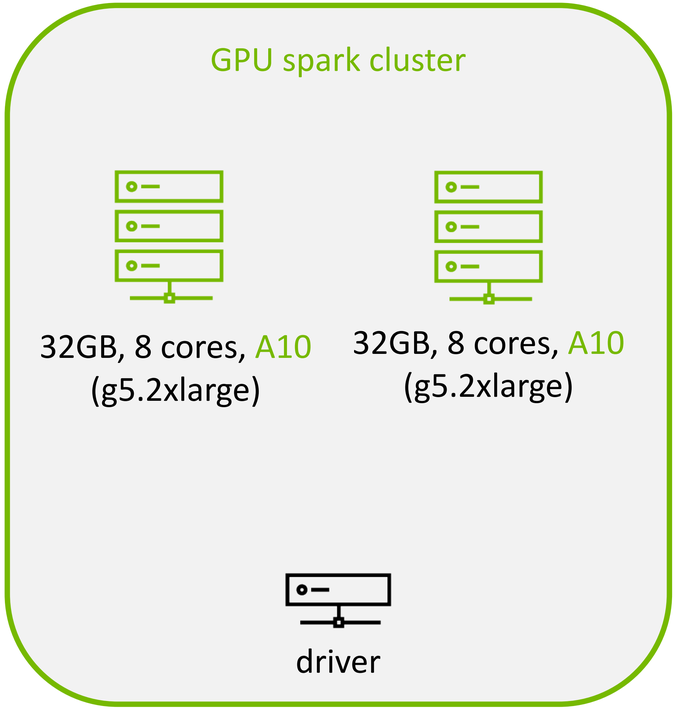

KMeans Clustering Result
cluster 0
cluster 1
Y
X
4
2
0
−2
−4
−4
−2
0
2
4

Microbenchmarking

- **Environment**
  - Databricks AWS hosted Spark

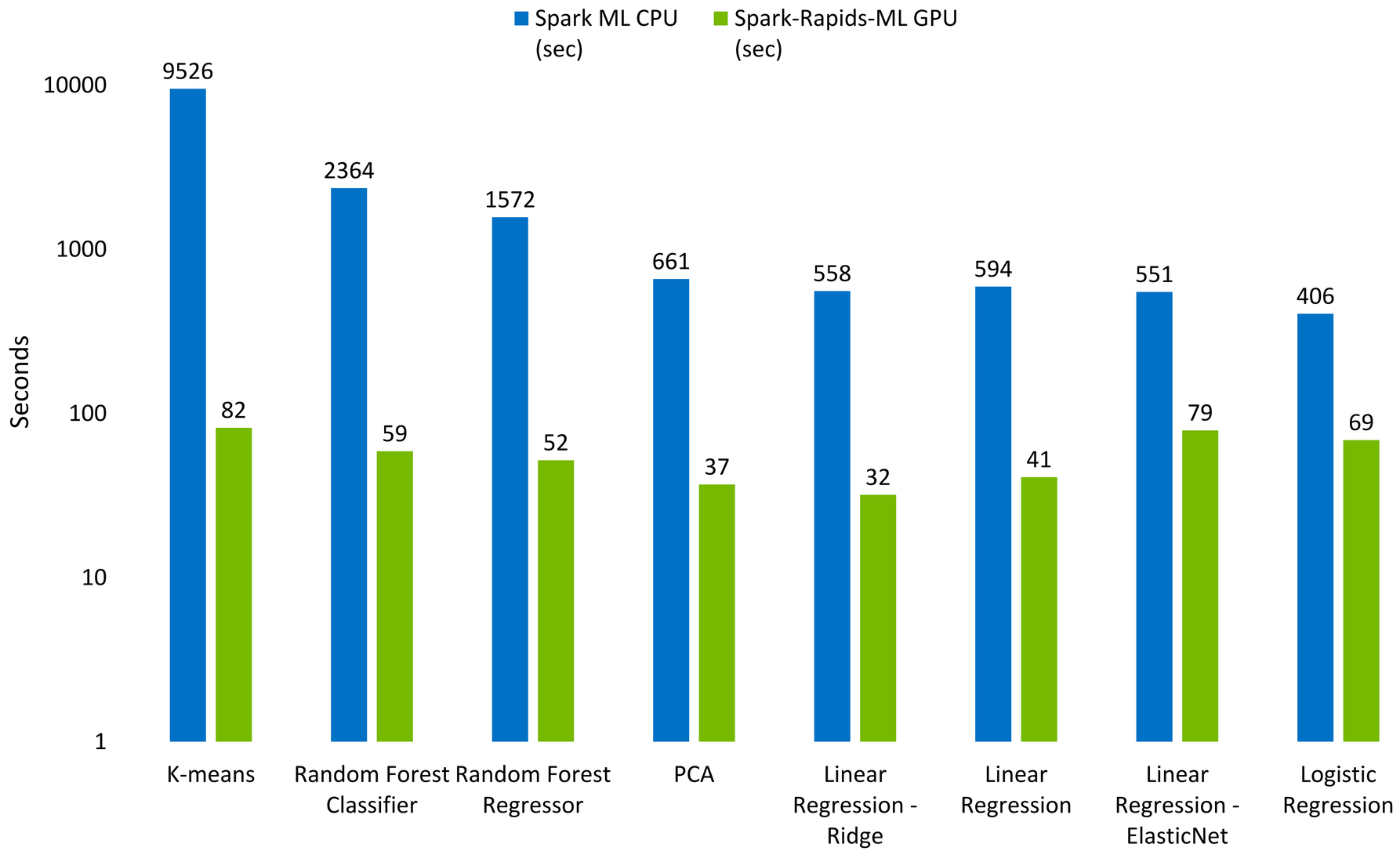
- **Workload & Data**
  - estimator.fit(data_df) [i.e. training]
  - data_df read from Parquet format in AWS S3
  - Compute intensive synthetic workloads:
    - 1 million rows
    - 3000 dimensional vectors
  - Data available in S3 public bucket.



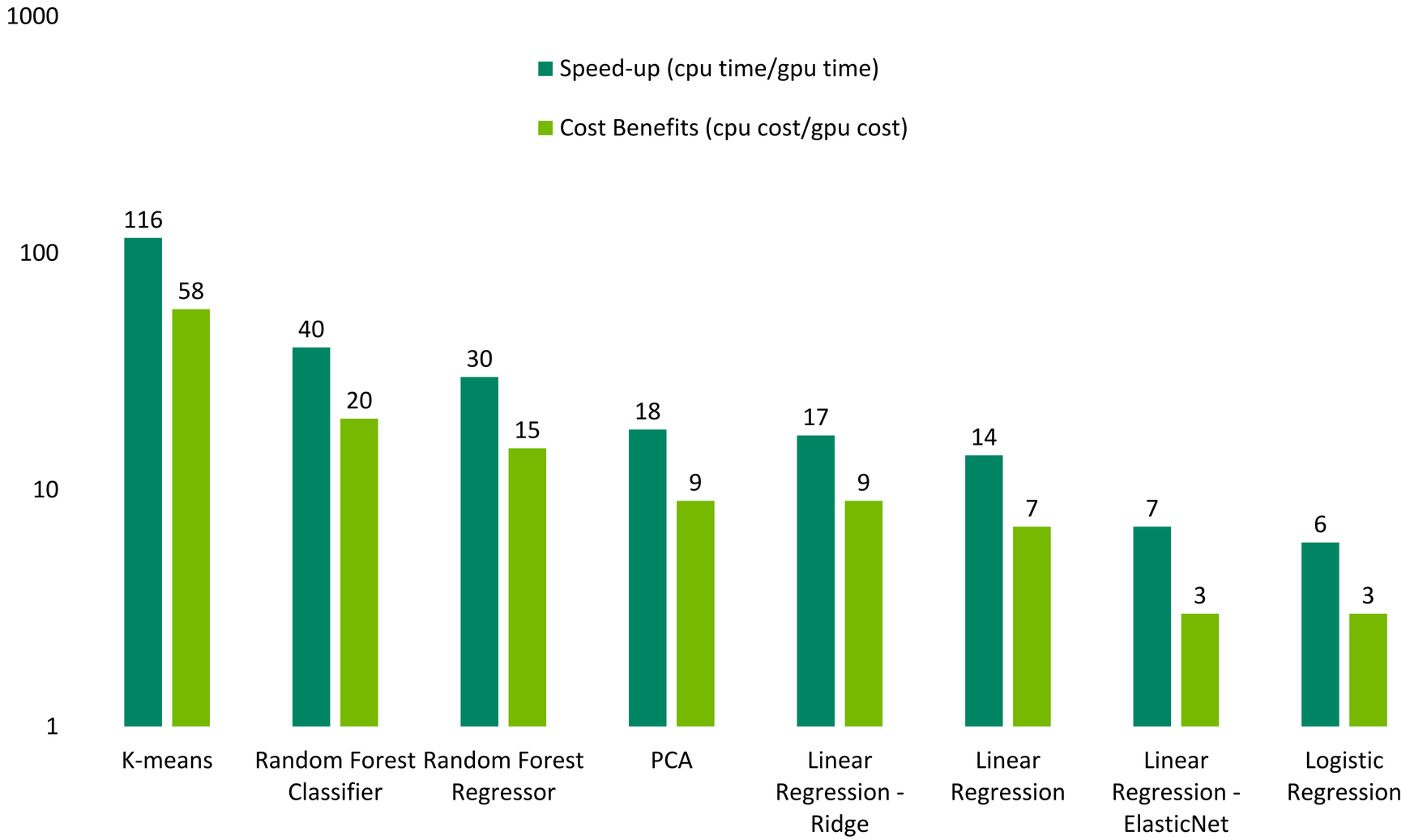


- Instructions and scripts to reproduce: https://github.com/NVIDIA/spark-rapids-ml/tree/main/python/benchmark#databricks
- [Repo also has instructions for GCP Dataproc and AWS EMR]

Training/fit Time: 6x-100x Faster



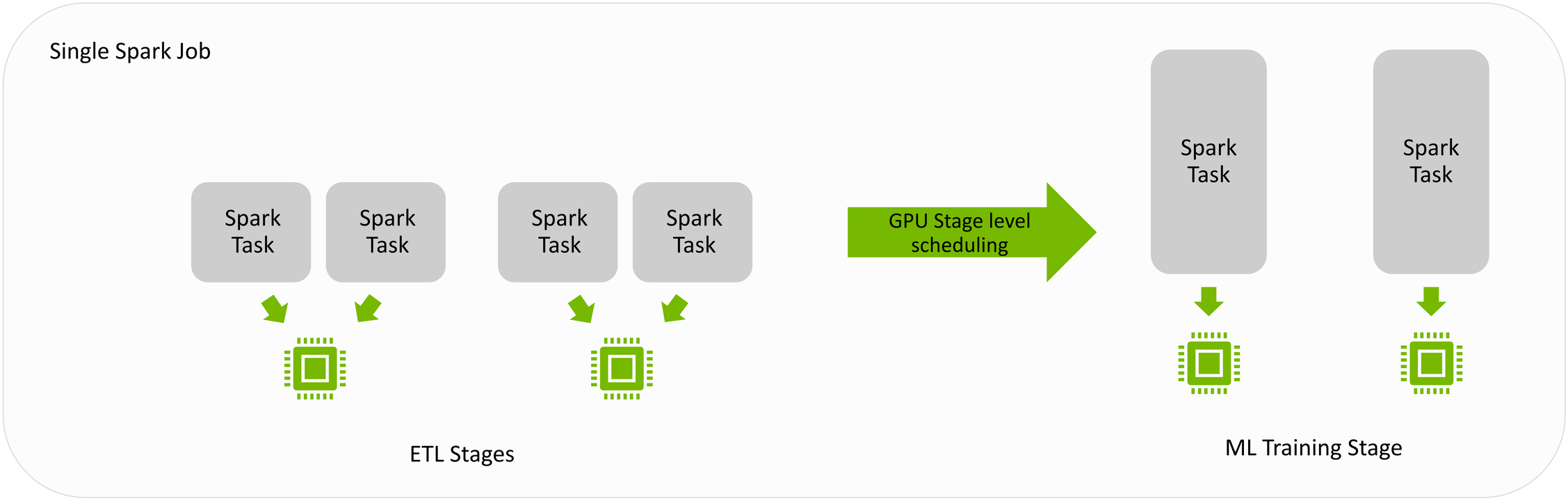
Cost Benefits and Speedups



End-to-End Acceleration

Stage Level Scheduling

- ML training stage runs all tasks at the same time with one Task for each GPU
  - Required by cuML/NCCL layer
- ETL can benefit from multiple Tasks per GPU
- Stage level scheduling:
  - Allows different tasks per GPU on a per stage basis within the same Spark Job.
- GPU aware stage level scheduling



End-to-End Acceleration

Accelerated CrossValidator

- PySpark API compatibility allows all accelerated Algos to leverage PySpark's built in CrossValidator for hyper parameter tuning
- We can do better:

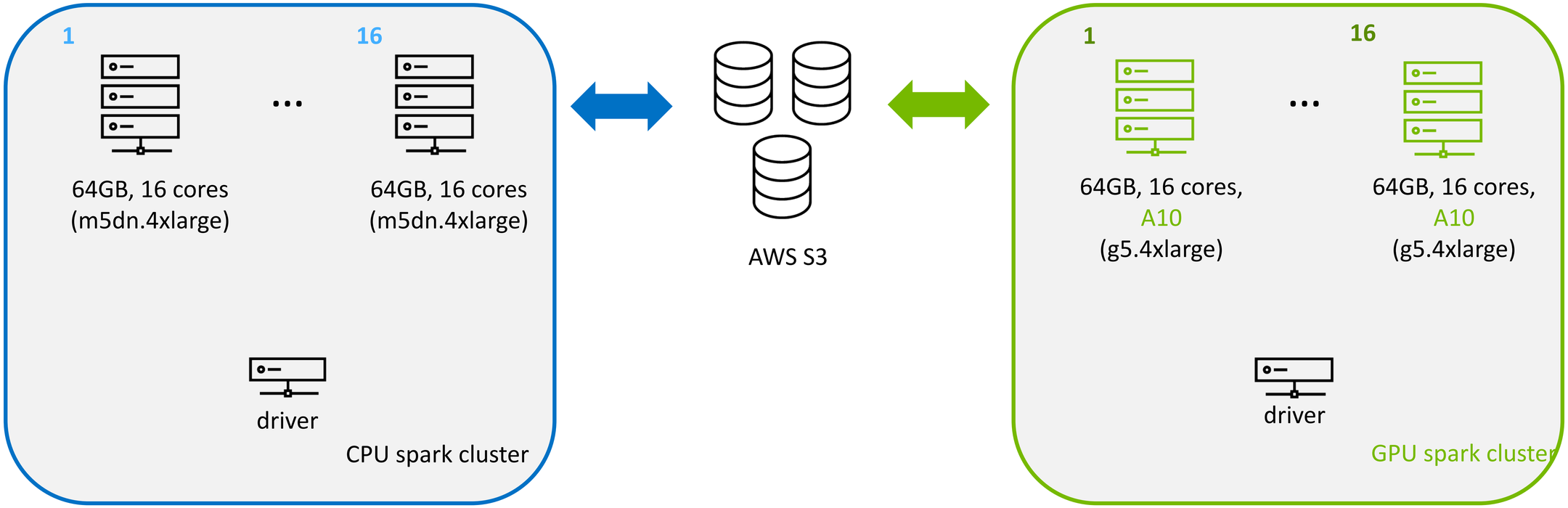
PySpark MLlib CrossValidator

RAPIDS Spark ML CrossValidator

Fannie Mae Mortgage Benchmark

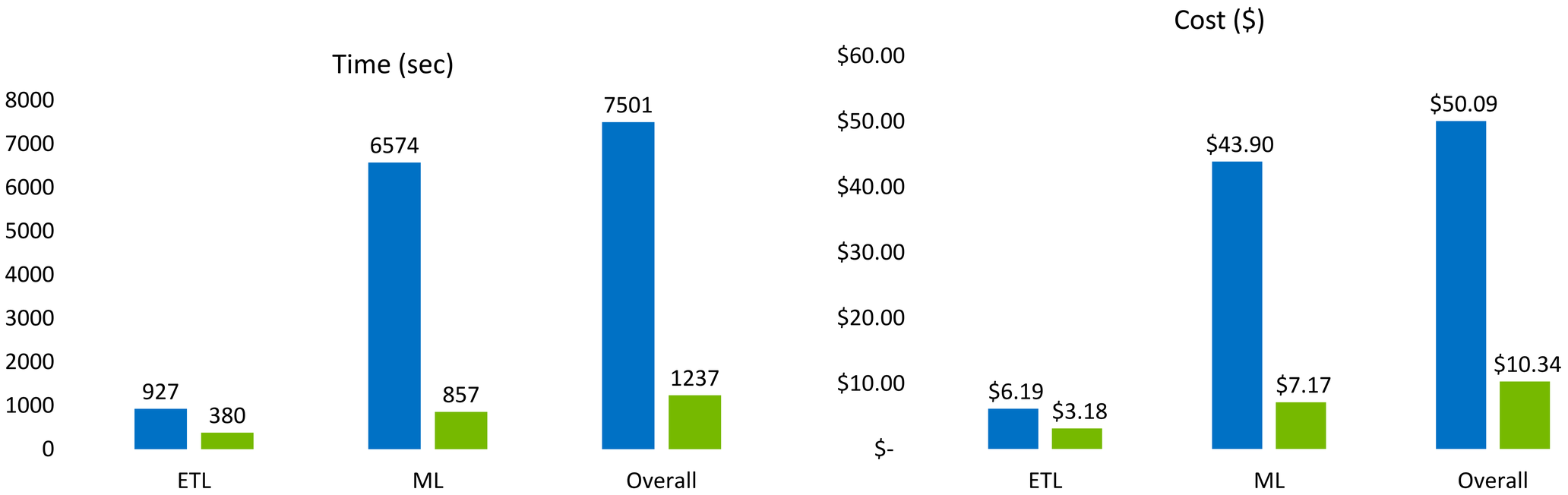
ETL + ML

- End-to-end ETL – ML workload:
  - ETL:
    - Starts with compressed Fannie Mae Single-Family Loan Performance Data ~ 800GB csv dataset converted to 26.8 GB compressed Parquet.
    - Feature engineering to 2.6 billion records by 27 feature dataset with loan delinquency as label. (as in this example: https://github.com/NVIDIA/spark-rapids-examples/.../MortgageETL.ipynb).
  - ML
    - Logistic regression with 3 fold cross validation wrt to log loss over 8 algo parameter choices.
  - Environment: Databricks AWS 16 node clusters:



Fannie Mae Mortgage Benchmark

ETL + ML



6.1x faster end-to-end

4.8x lower cost

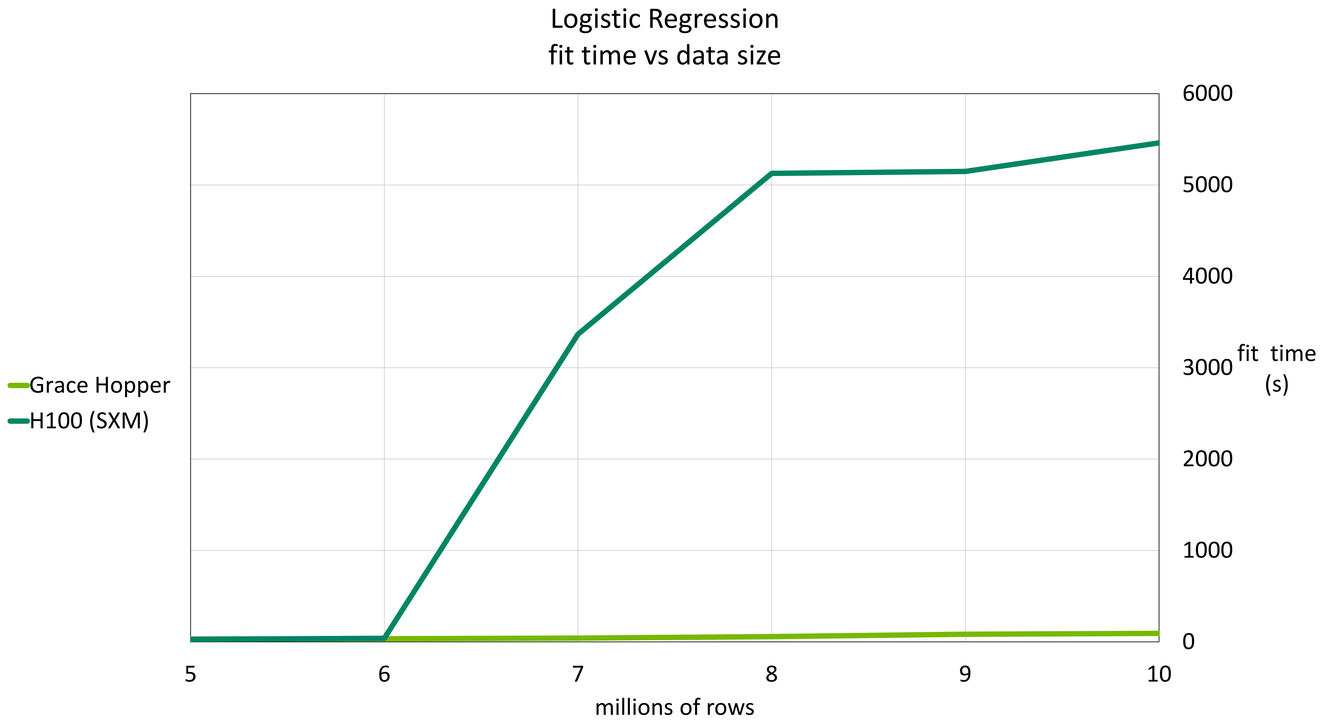
Accuracy: GPU – CPU average CV score < 0.004%

Roadmap

- Better out-of-core
- Blackwell and Grace-Hopper optimizations
- Spark APIs for more algorithms from cuML:
  - Additional approximate nearest neighbor vector search algorithms
  - ARIMA and AutoARIMA
- GPU optimized Pipelines
- Optimizing data transfer from JVM to Python via CUDA IPC
- Additional Spark MLlib algos

Roadmap: Out-of-core

- Rapids Accelerator for Spark MLlib is optimized for data fitting on GPUs.
- When it is not possible to get more GPUs and out-of-core capability is needed:
  - Current solution based on CUDA managed memory and RAPIDS Memory Manager.
  - On road map: leveraging Grace Hopper's unified host and device memory and system malloc.

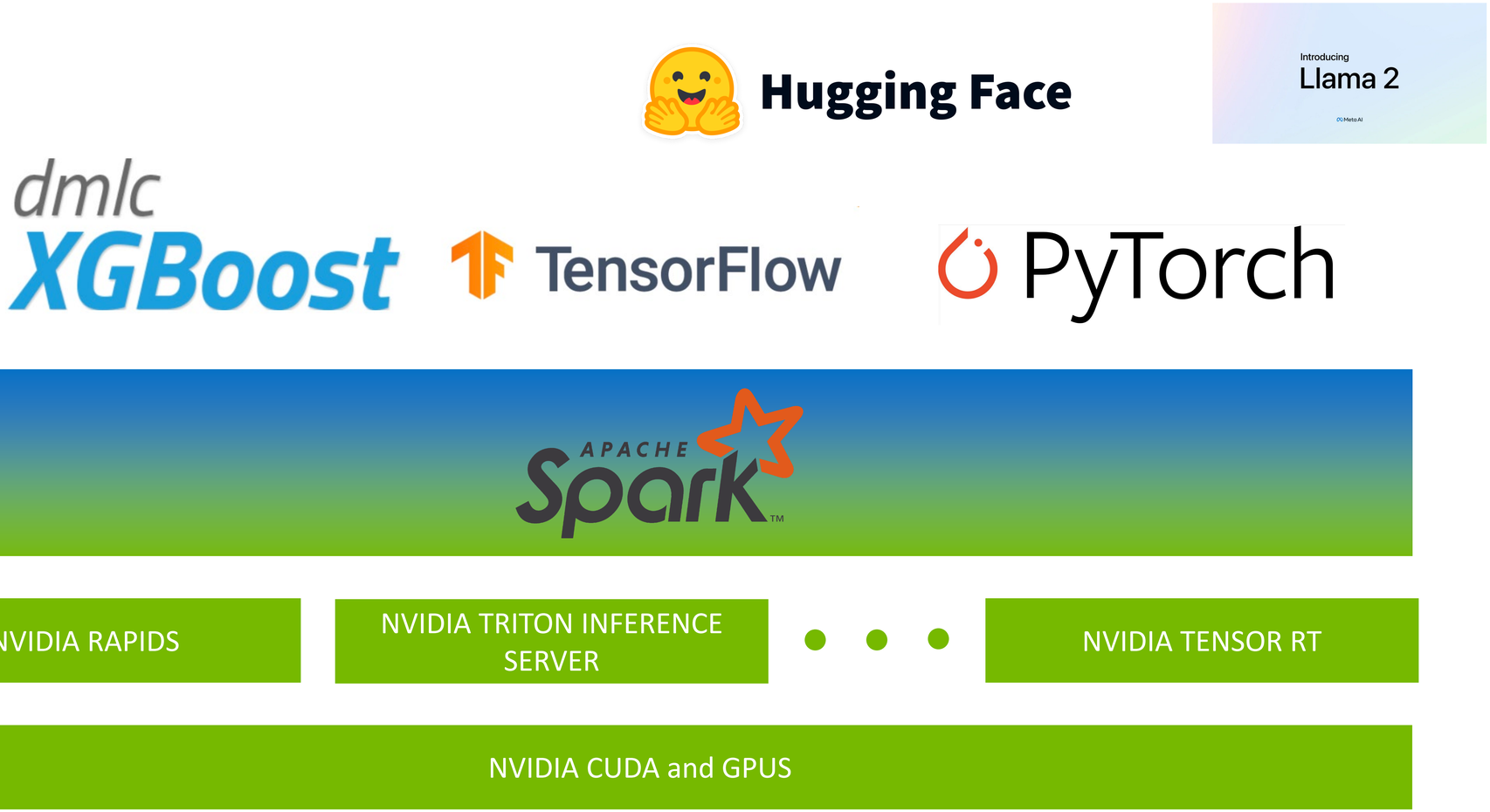


Take Aways

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

Open-source	Cloud		
			
Apache Spark 3+ Community Release	Amazon EMR	Databricks	Google Cloud Dataproc

Broader GPU Accelerated ML/DL on Spark Ecosystem



For More Information

spark-rapids-support@nvidia.com

https://docs.nvidia.com/spark-rapids

https://nvidia.github.io/spark-rapids

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

https://nvidia.github.io/spark-rapids-ml