DATA+AI SUMMIT 2022

Scalable XGBoost on GPU clusters

Jiaming Yuan
Software engineer, NVIDIA

Bobby Wang
Distributed system software engineer, NVIDIA

ORGANIZED BY databricks

Agenda

Jiaming Yuan

- Introduction to XGBoost
- New categorical feature and multiple outputs support
- GPU acceleration

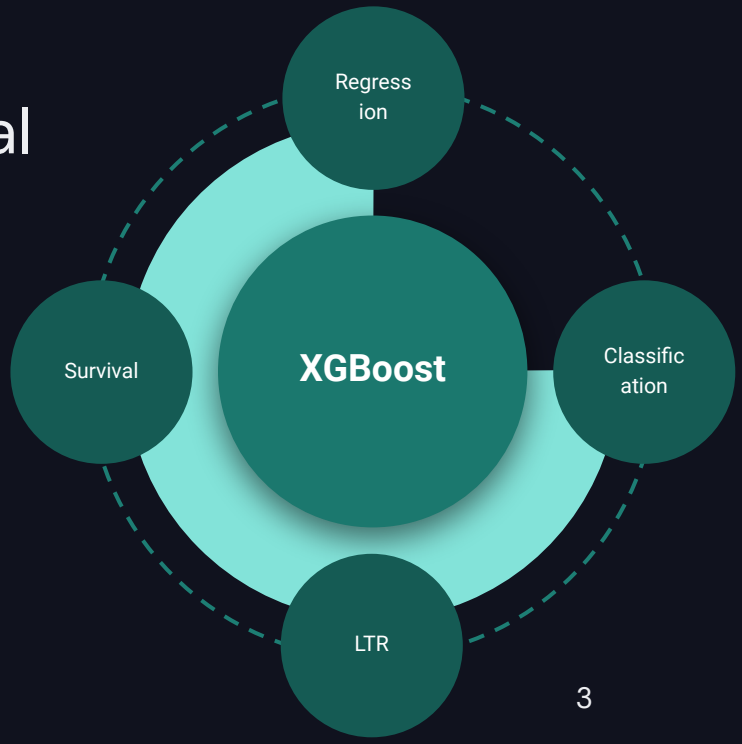
Bobby Wang

- XGBoost + Spark + GPU

XGBoost

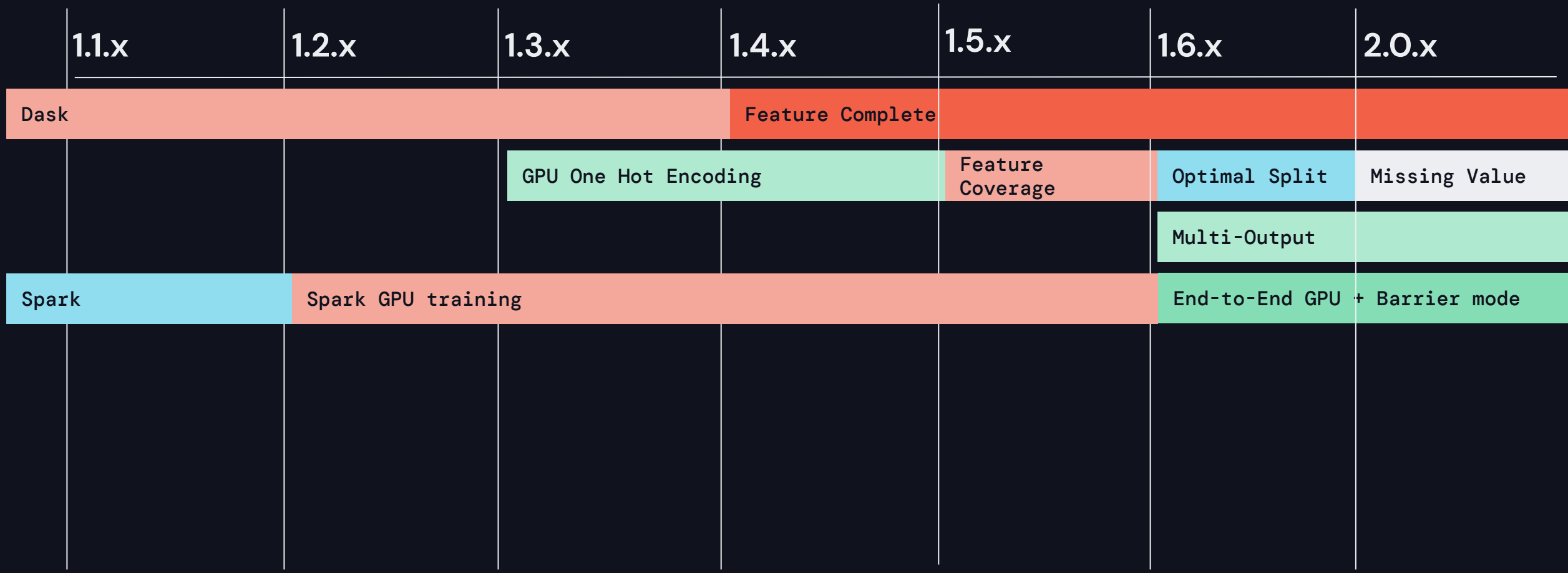
Open source gradient boosting library

- Supports regression, classification, ranking, survival training, and user defined objectives
- Efficient and scalable for handling large data size.
- Reusable model (Slicing, JSON, UBJSON)
- Supports different types of data including categorical features, multi-target regression and multi-label classification.



A Brief History

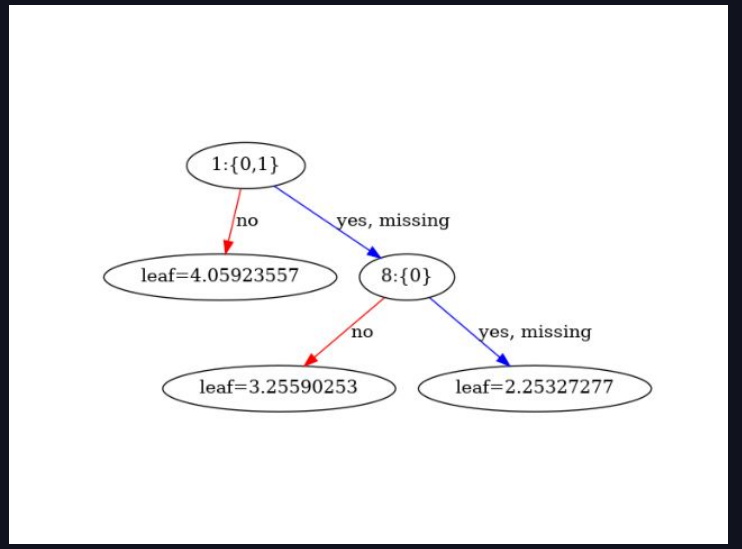
Timeline



Categorical data

Categorical Feature

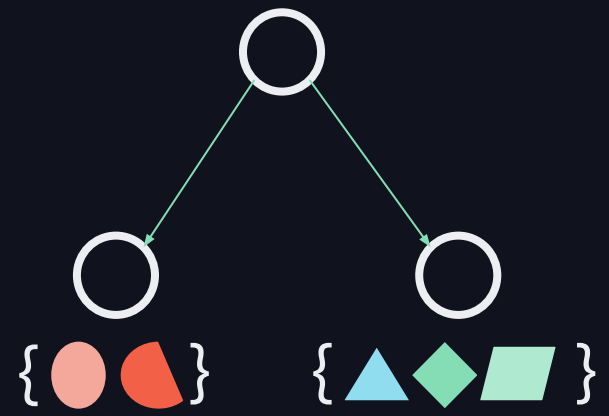
- Often seen in tabular datasets
- Most commonly used with data encoding (one-hot, target)
- Tree model is well suited for it.



Categorical Feature in XGBoost

Experimental Feature

- Built-in support for one-hot encoding.
  - Tree split on each category, no need for sparse data from explicit encoding.
- Support optimal split.
  - Split the set of categories into tree branches.
  - Partition the categories optimally based on sorted histogram.
- Runs on GPU.
  - Users can pass categorical data as a cuDF dataframe, and run the entire training session on GPU.



.. _XGBoost Categorical feature handling tutorial: https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html

.. _On Grouping for Maximum Homogeneity: https://www.tandfonline.com/doi/abs/10.1080/01621459.1958.10501479

.. _The Elements of Statistical Learning: https://link.springer.com/book/10.1007/978-0-387-84858-7

.. _LightGBM\: A Highly Efficient Gradient Boosting Decision Tree: https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

Let’s code

```
.. code:: python

X["cat_feature"].astype("category")

# Supported tree methods are `gpu_hist`, `approx`, and `hist`.
clf = xgb.XGBClassifier(tree_method="gpu_hist", enable_categorical=True)
clf.fit(X, y)
# Must use JSON/UBJSON for serialization
clf.save_model("categorical-model.json")
```

Use a dataframe with categorical data, cuDF or pandas

Use a supported tree method, enable categorical data.

Well, we want to use the model later right?

Ecosystem

Accelerated Inference

- Treelite
- Fast Inference Library (cuml)

```
from cuml import ForestInference

# Load the classifier previously saved with xgboost model_save()
model_path = 'model.json'
fm = ForestInference.load(model_path, output_class=True)
# Generate random sample data
X_test, y_test = sklearn.datasets.make_classification()
# Make predictions (as a GPU array)
fil_preds_gpu = fm.predict(X_test.astype('float32'))
```

source:
https://medium.com/rapids-ai/rapids-forest-inference-library-prediction-at-100-million-rows-per-second-19558890bc35

Future Work

- On going work for sparse data handling.
- More parameters for regularization.
- Performance.
- Other algorithms.

DATA+AI
SUMMIT 2022

Multiple Outputs

Multiple Outputs

- Multi-target regression
- Multi-label classification

```
from sklearn.datasets import make_multilabel_classification
import numpy as np

X, y = make_multilabel_classification(
    n_samples=32, n_classes=5, n_labels=3, random_state=0
)
clf = xgb.XGBClassifier(tree_method="gpu_hist")
clf.fit(X, y)
```

Why?

- More efficient as XGBoost can reuse data structures across all targets.
- Custom objective and callback functions.
- Early stopping.
- More features are coming.

Future Work

- Correlated targets.
- Sparse targets. (multi-label)

GPU Acceleration

XGBoost accelerated by GPU

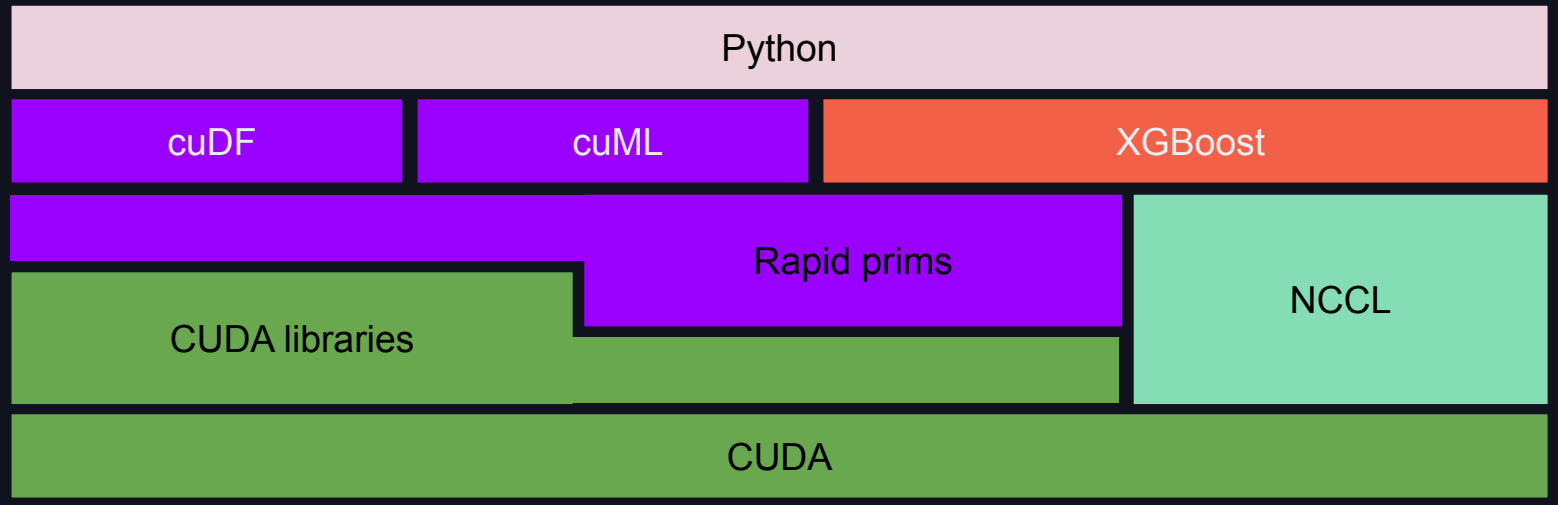
End to end

- Model training for all tasks
- Inference
- Model explanation
- Evaluation
- GPU inputs and outputs
- Deterministic result
- Distributed

Pipeline

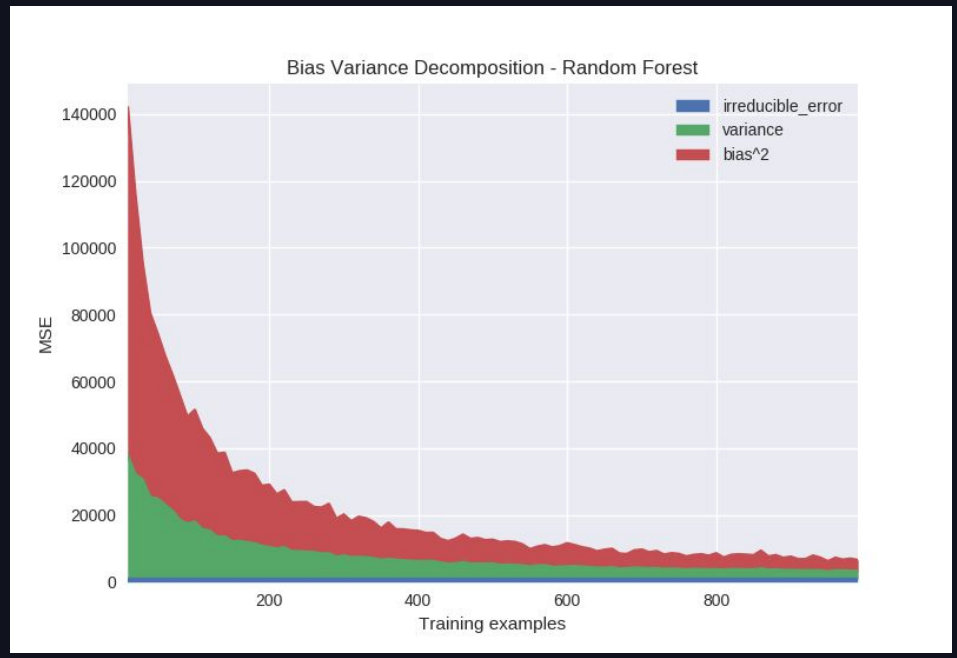
The GPU stack

- Fully integrated environment
- We will talk more about distributed environment



GPU Acceleration

- Efficient memory usage
  - DeviceQuantileDMatrix
  - In-place prediction
- More data is better



source: https://developer.nvidia.com/blog/bias-variance-decompositions-using-xgboost/

Performance – Environment

- System information
- Training parameters

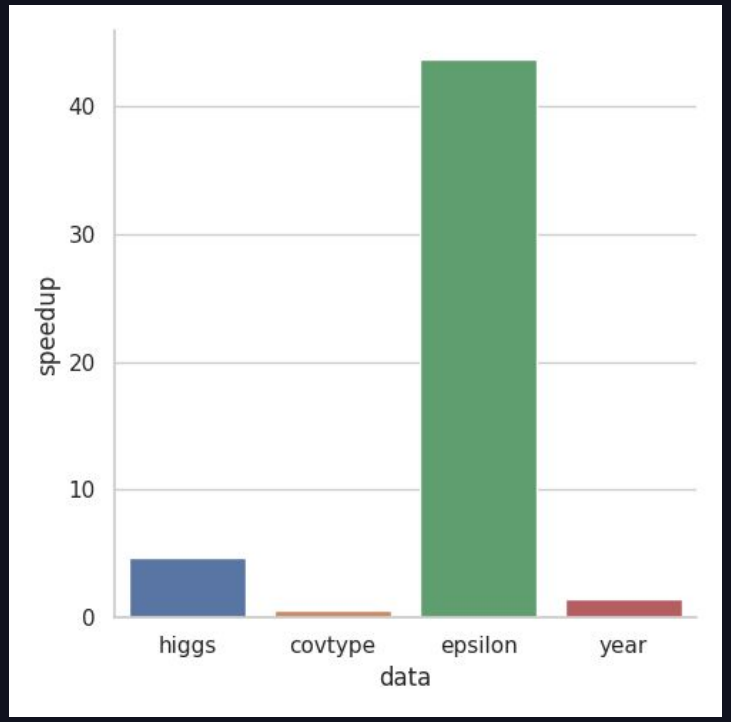
NGC
- V100
- 40 CPU cores (no hyper-threading)

- max_depth=8
- depthwise

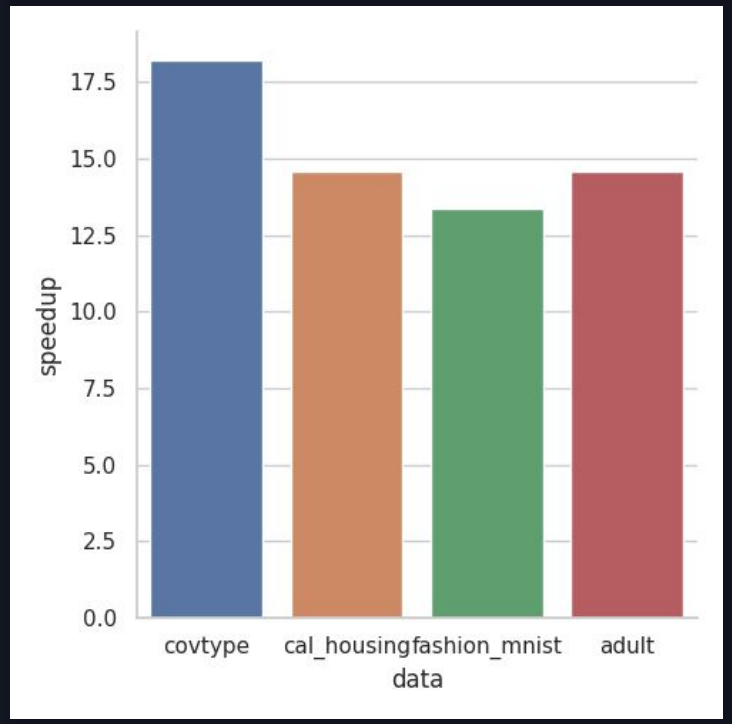
Datasets for SHAP comparison are smaller due to time limit.

Performance

Training



SHAP Contribution



DATA+AI SUMMIT 2022

Distributed XGBoost

Spark and Dask Support

ORGANIZED BY databricks

Distributed XGBoost

- Integration with multiple frameworks
  - Dask
  - Spark (focus of this talk)
- Efficient training
- ROC-AUC and PR-AUC specialised for distributed environments

Integration with Multiple Frameworks

- Dask

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, random_state=0)
reg = xgboost.dask.DaskXGBRegressor(callbacks=[EarlyStopping()])
reg.set_params(tree_method='gpu_hist')

# assigning client here is optional
regressor.client = client
regressor.fit(X_train, y_train, eval_set=[(X_valid, y_valid)])
```

- Spark

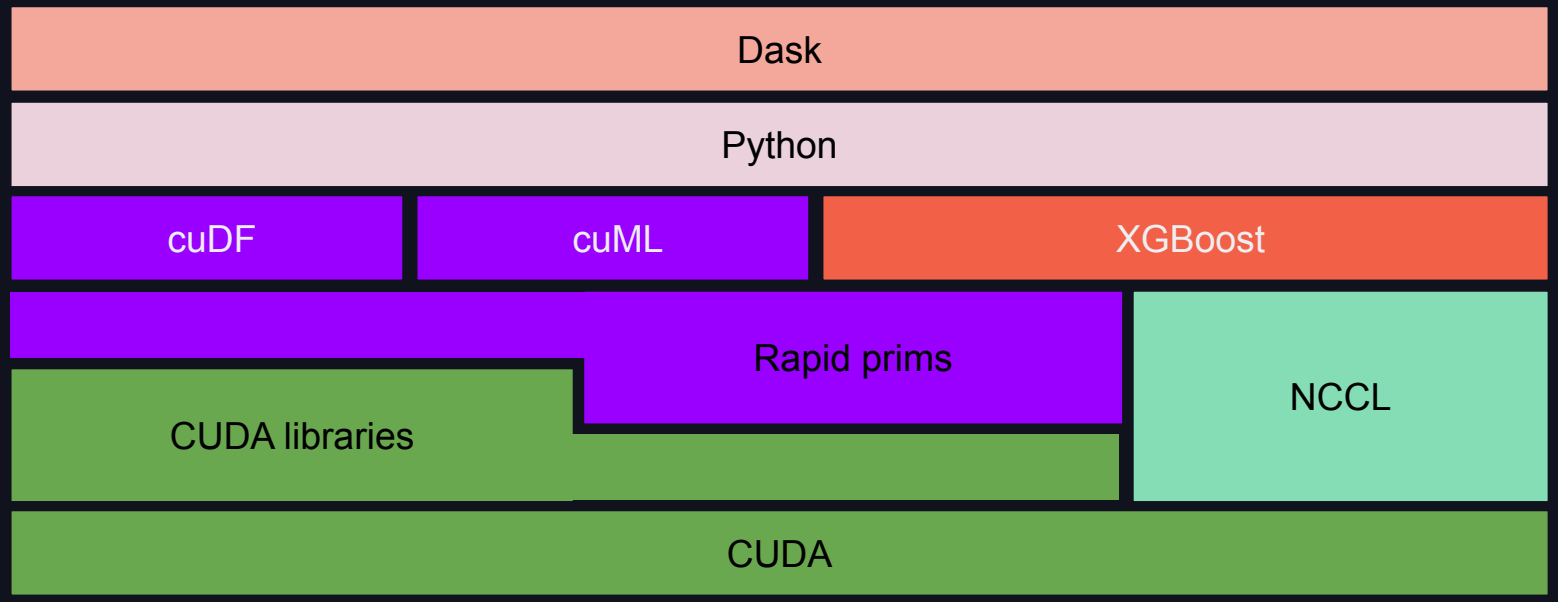
```
val features = Array("sepal length", "sepal width", "petal length", "petal
width")

val xgbClassifier = new XGBoostClassifier(xgbParam)
.setFeaturesCol(features)
 .setLabelCol("class")

val xgbClassificationModel = xgbClassifier.fit(rawInput)
```

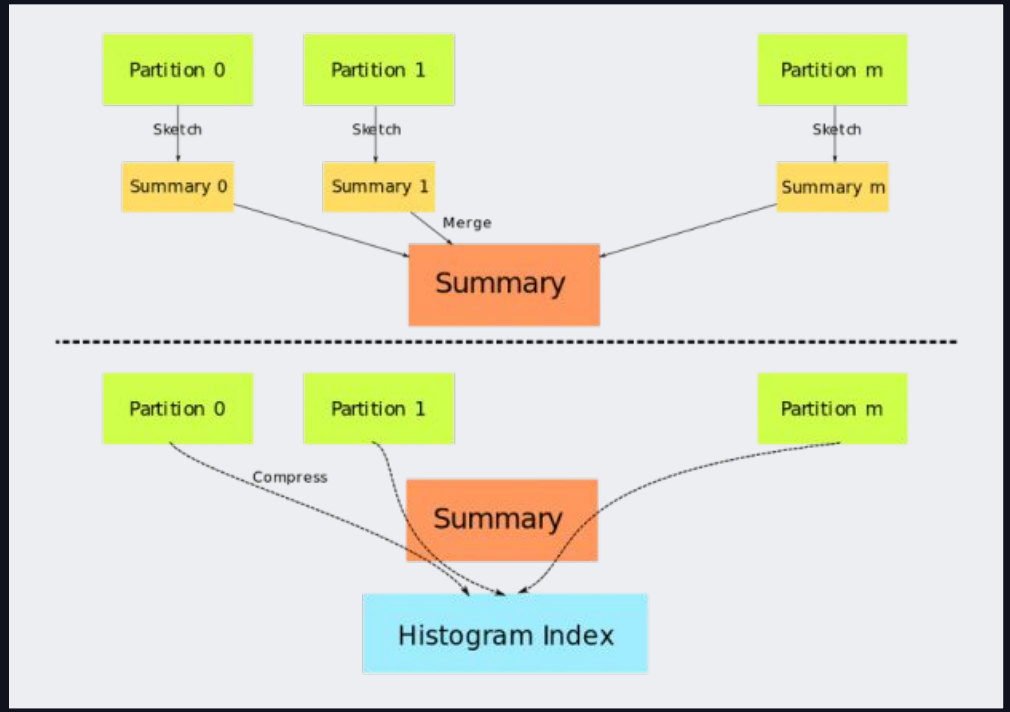
XGBoost + Dask + GPUs

The Python environment you like



End-to-End on GPU

- Scale your computation from a laptop to GPU clusters easily.
- Data stays on GPU.
- Efficient memory usage.
- Enabled for both Dask and Spark.
- Don’t hesitate to get more data.



DATA+AI SUMMIT 2022

XGBoost+Spark+GPU

Bobby Wang
Distributed system software engineer, NVIDIA

- Barrier execution mode
- GPU acceleration

XGBoost JVM packages

Barrier Execution Mode

- Robust training.

Avoid killing SparkContext by introducing barrier execution mode in XGBoost 1.6.1 release.

> **Warning!**
>
> Versions of XGBoost 1.2.0 and lower have a bug that can cause the shared Spark context to be killed if XGBoost model training fails. The only way to recover is to restart the cluster. Databricks Runtime 7.5 ML and lower include a version of XGBoost that is affected by this bug. To install a different version of XGBoost, see Install XGBoost on Databricks.

XGBoost4j-Spark-GPU

- XGBoost4J-Spark-GPU is an open-source library for accelerating distributed XGBoost training on Apache Spark cluster.
- The end-to-end GPU acceleration is enabled by the RAPIDS Accelerator for Apache Spark.

XGBoost4j-Spark-GPU

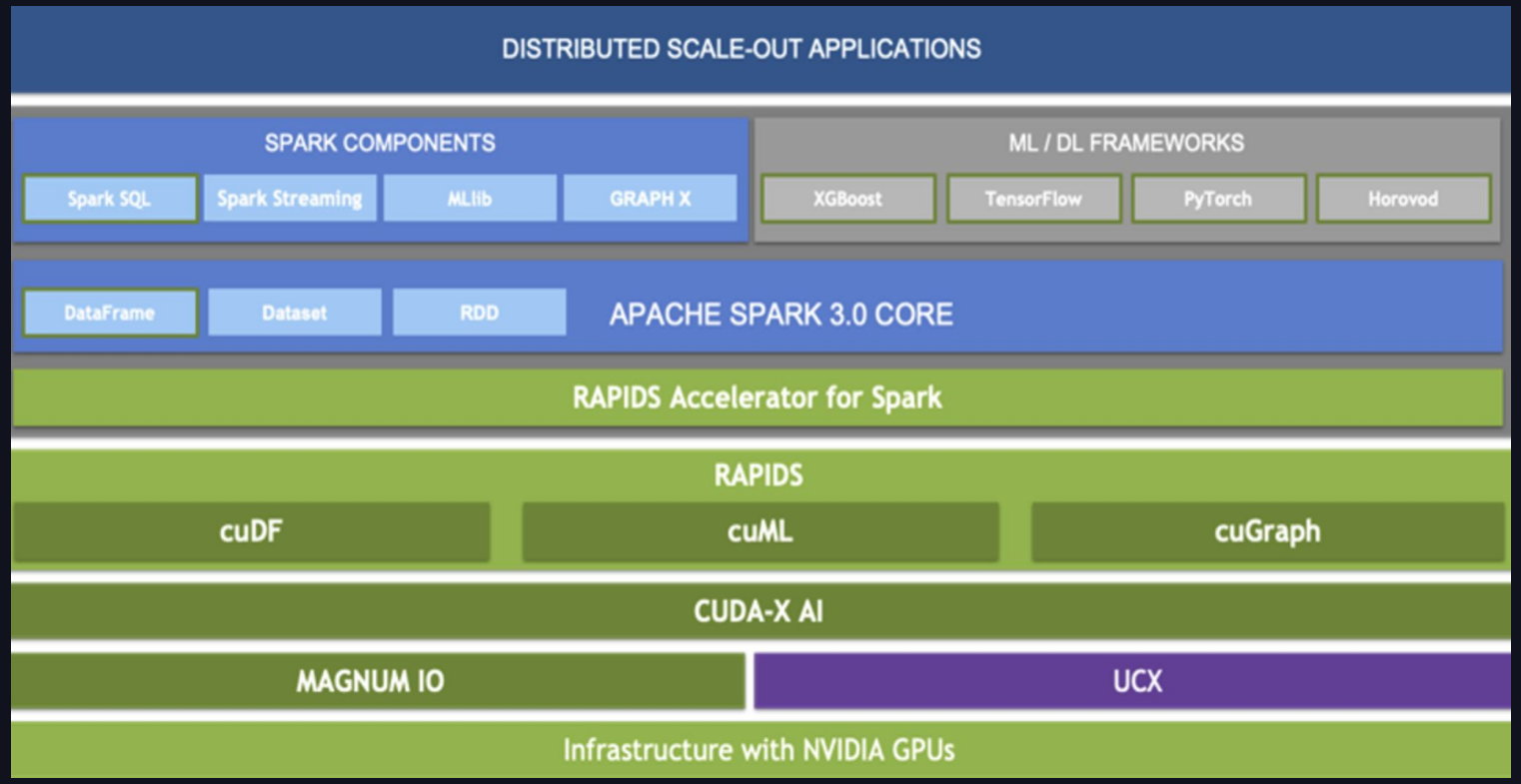
Difference with XGBoost4j-Spark

- XGBoost4j-Spark by itself does not support GPU training.
- XGBoost4j-Spark-GPU extends XGBoost4j-Spark with RAPIDS Accelerator to enable GPU pipeline including training and ETL (extract, transform, load).

XGBoost4j-Spark-GPU

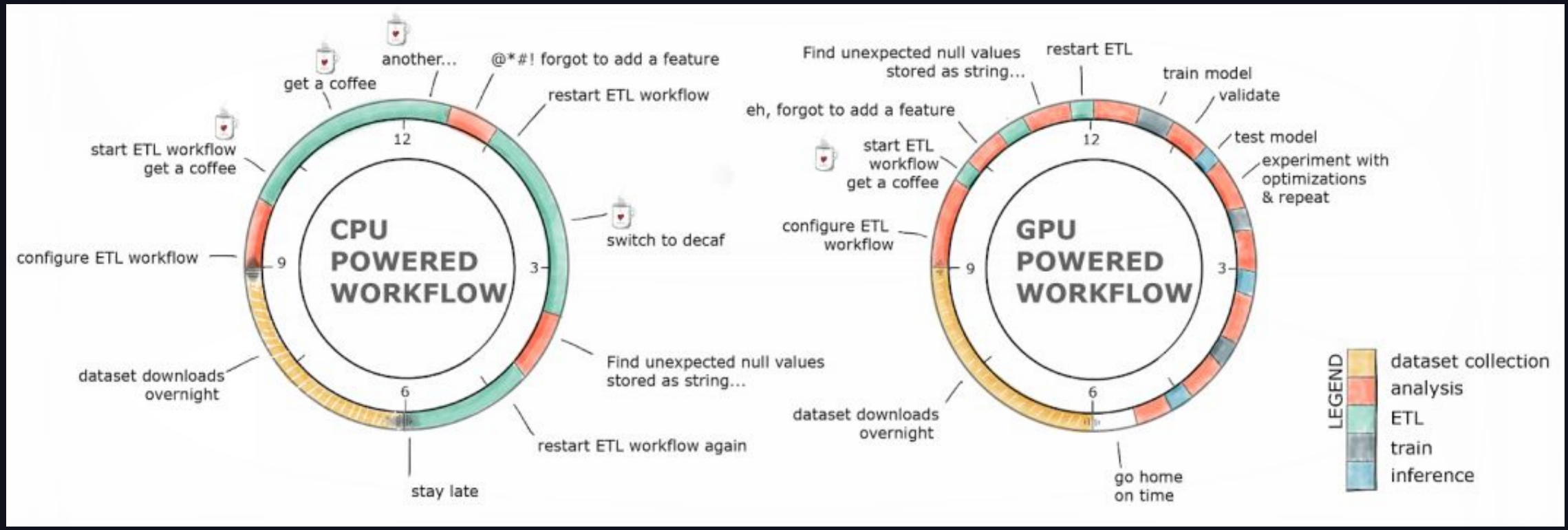
RAPIDS Accelerator

The RAPIDS Accelerator for Apache Spark leverages GPUs to accelerate data processing via the RAPIDS libraries.



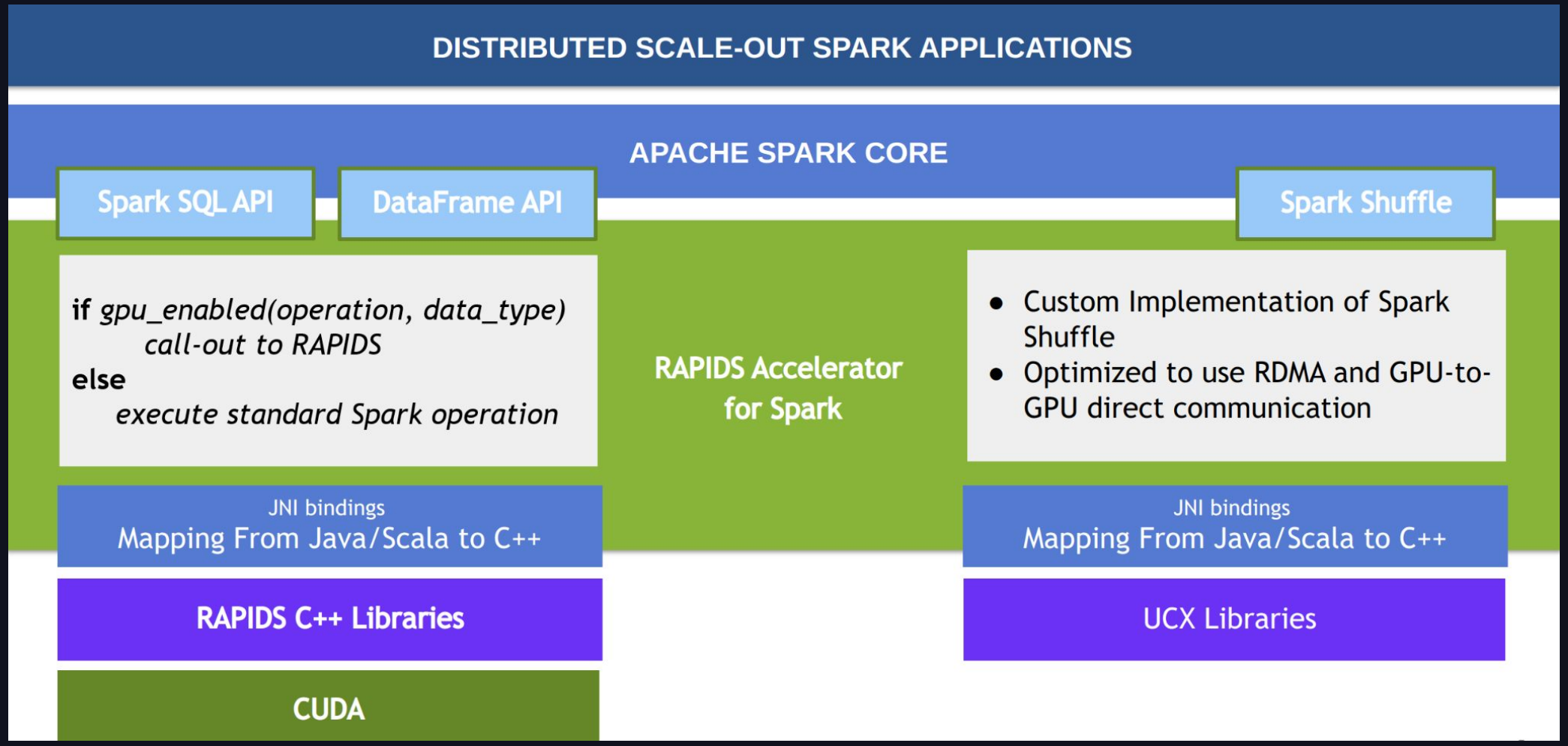
XGBoost4j-Spark-GPU

RAPIDS Accelerator



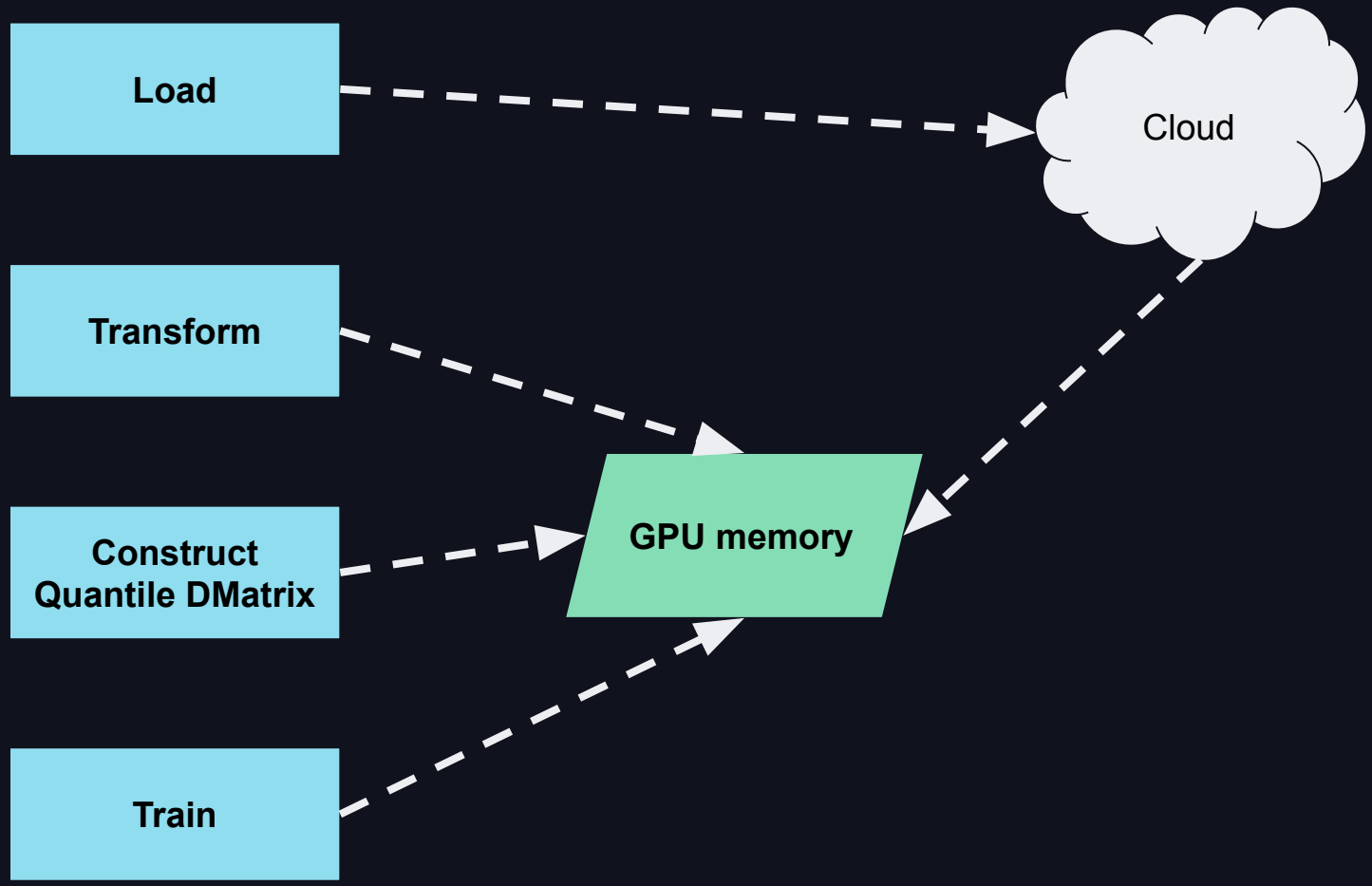
XGBoost4j-Spark-GPU

RAPIDS Accelerator



XGBoost4j-Spark-GPU

RAPIDS Accelerator



XGBoost4j-Spark-GPU

Unify the setFeaturesCol API

Original API

```
def setFeaturesCol(value: String): this.type
```

Accept a vector typed feature name

The new API can also be used in XGBoost4j-Spark project

```
def setFeaturesCol(value: Array[String]): this.type
```

Accept an array of feature column names

XGBoost4j-Spark-GPU

No code change

```
val schema = new StructType(Array(
   StructField("sepal length", DoubleType, true), StructField("sepal width", DoubleType, true),
   StructField("petal length", DoubleType, true), StructField("petal width", DoubleType, true),
   StructField("class", DoubleType, true)))
val input = spark.read.schema(schema).csv(path)
val features = Array("sepal length", "sepal width", "petal length", "petal width")
val xgbClassifier = new XGBoostClassifier(xgbParam)
   .setFeaturesCol(features)
   .setLabelCol("class")
val model = xgbClassifier.fit(input)
```

XGBoost4j-Spark-GPU

Performance test

- cpu + cpu :
  CPU loading + CPU training with XGBoost4j-Spark-GPU 1.6.1
- cpu + gpu:
  CPU loading + GPU training with XGBoost4j-Spark-GPU 1.6.1
- gpu + gpu:
  GPU loading with RAPIDS Accelerator 22.04.0 + GPU training with XGBoost4j-Spark-GPU 1.6.1
- db-python (cpu+gpu):
  Databricks xgboost pyspark based on XGBoost python package 1.5.2

XGBoost4j-Spark-GPU

Performance1

Databricks Cluster:

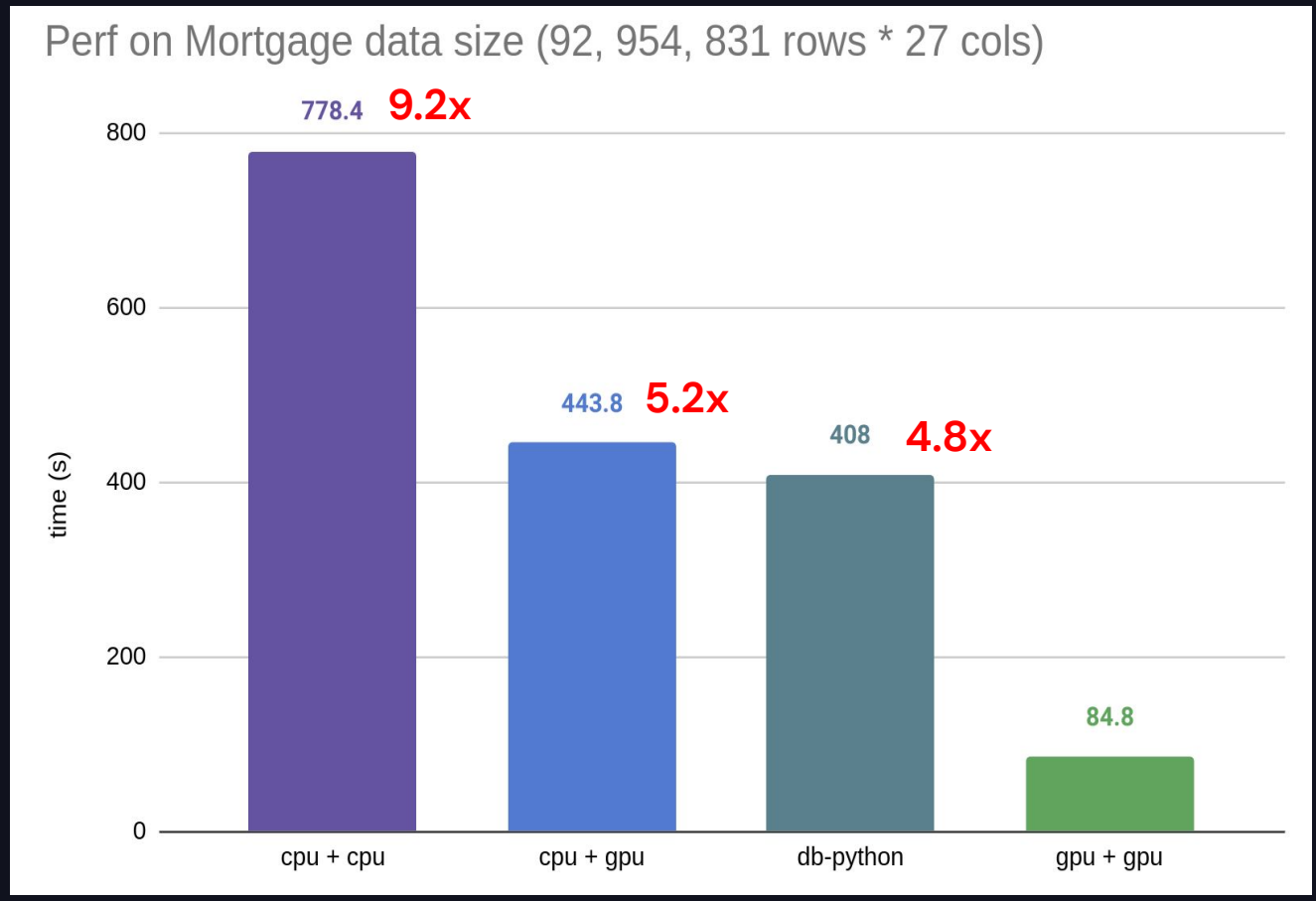
Runtime: 10.4 LTS ML (includes Apache Spark 3.2.1, GPU, Scala 2.12)
Worker Type: 2 * g4dn.2xlarge(32 GB Mem, 8 vCPUs, 1 GPU)
Driver Type: g4dn.xlarge(16 GB Mem, 4 vCPUs, 1 GPU)

Spark Configuration:

```
spark.task.resource.gpu.amount=1
```

Jars:

xgboost4j-gpu_2.12-1.6.1.jar
xgboost4j-spark-gpu_2.12-1.6.1.jar
rapids-4-spark_2.12-22.04.0.jar
cudf-22.04.0-cuda11.jar



XGBoost4j-Spark-GPU

Performance2

Databricks Cluster:

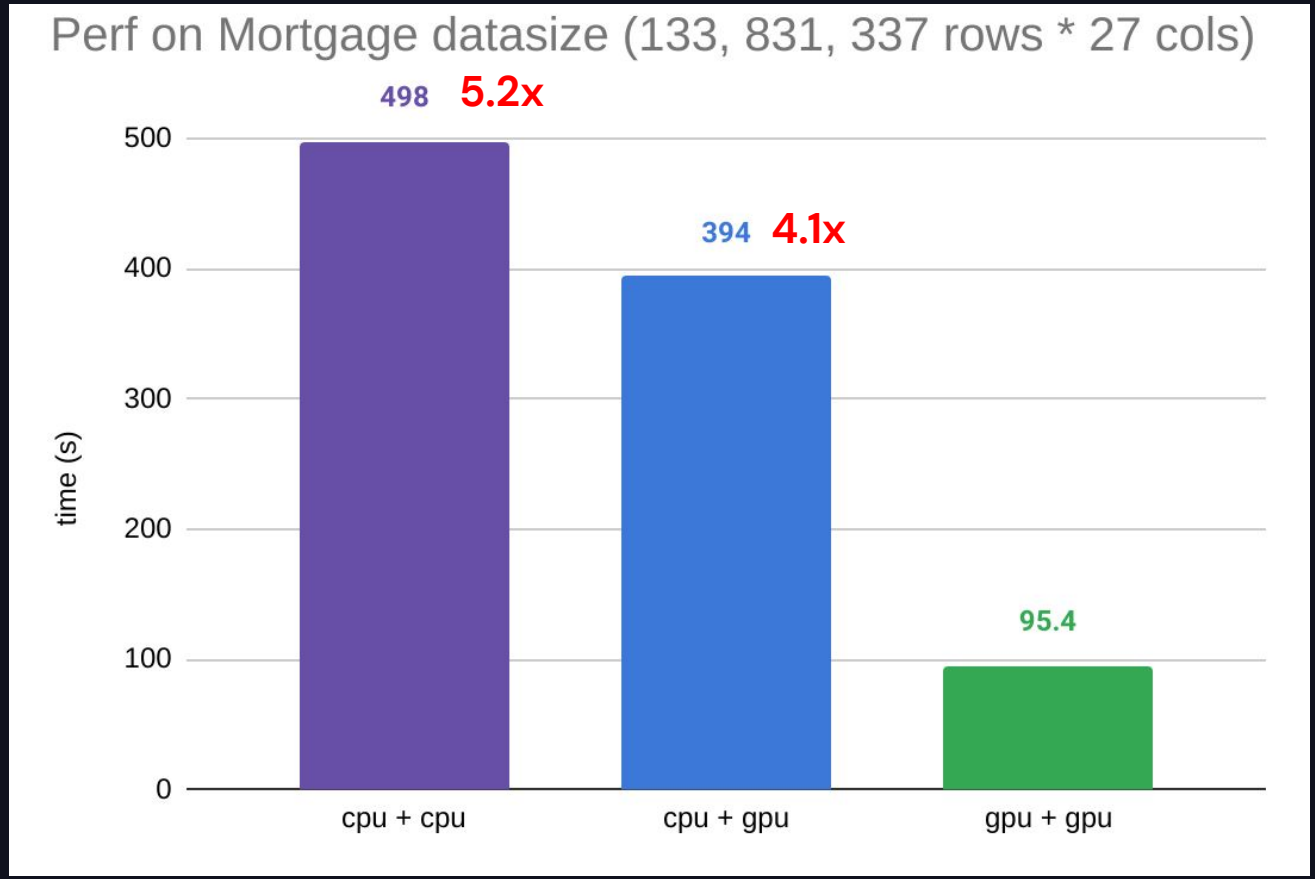
Runtime: 10.4 LTS ML (includes Apache Spark 3.2.1, GPU, Scala 2.12)
Worker Type: 2 * g4dn.2xlarge(32 GB Mem, 8 vCPUs, 1 GPU)
Driver Type: g4dn.xlarge(16 GB Mem, 4 vCPUs, 1 GPU)

Spark Configuration:

```
spark.task.resource.gpu.amount=0.125
```

Jars:

xgboost4j-gpu_2.12-1.6.1.jar
xgboost4j-spark-gpu_2.12-1.6.1.jar
rapids-4-spark_2.12-22.04.0.jar
cudf-22.04.0-cuda11.jar



Reference

- https://developer.nvidia.com/blog/accelerating-xgboost-on-gpu-clusters-with-dask/
- https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html
- https://xgboost.readthedocs.io/en/stable/tutorials/multioutput.html
- https://medium.com/rapids-ai/rapids-forest-inference-library-prediction-at-100-million-rows-per-second-19558890bc35
- Details about the performance for XGBoost4j-Spark-GPU
- RAPIDS Accelerator

DATA+AI
SUMMIT 2022

Questions?

DATA+AI SUMMIT 2022

Thank you

Jiaming Yuan
Software engineer, NVIDIA

Bobby Wang
Distributed system software engineer, NVIDIA