RAPIDS ACCELERATOR FOR APACHE SPARK TRAINING + HANDS-ON LAB

AGENDA

Part I

- Spark Overview
- Hands-on: CPU queries
- Spark RAPIDS introduction
- Qualification tool

Part II

- Hands-on: GPU queries
- Profiling tool
- Hands-on: GPU tuning

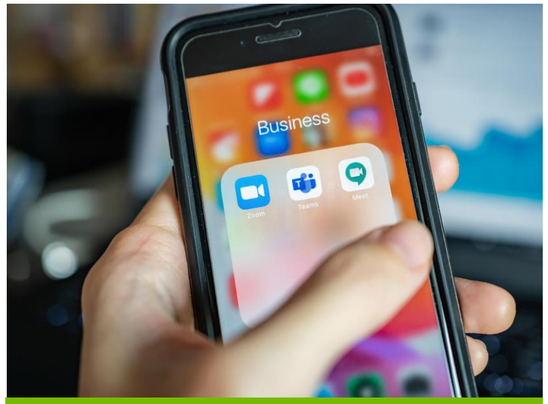
5 Goals For Training

- Learn how **Spark queries** are planned and executed
- Understand why **GPUs optimize Spark** performance
- See how the **Spark RAPIDS qualification tool** helps show the potential of GPU optimization for Spark jobs
- Discover how **GPU processing** differs from CPU processing in Spark to accelerate workloads
- Grasp the **key configurations** for Spark RAPIDS for optimal performance

SPARK OVERVIEW + INTRODUCTION

SPARK IS BROADLY ADOPTED

1M+ servers across 16,000+ enterprises using Spark



CONSUMER INTERNET

Recommender Systems

Advertising Analytics

Audience Segmentation

FINANCIAL SERVICES

Fraud Detection

Risk Analysis

Portfolio Management

TELECOM

Network Quality

IOT Analytics

Transition to 5G

FEDERAL & SLED

Threat Detection

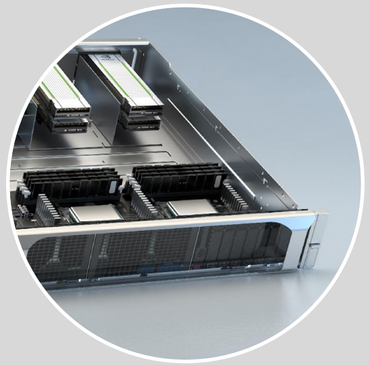
Intelligence Analytics, AI

Geospatial Analytics

Enterprises using Spark

INDICATORS THAT SPARK CUSTOMERS NEED ACCELERATION

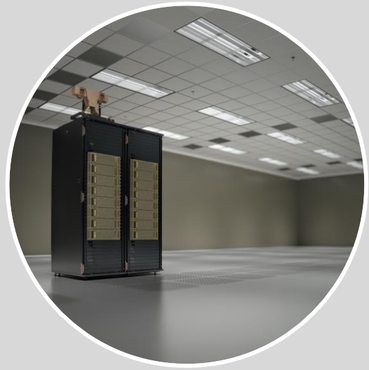
Turn faster data analytics into competitive advantage



TIME TO SOLUTION

GPU parallelism within each node

Columnar processing



INFRASTRUCTURE OPTIMIZATION

Reduced data center costs

Fewer server nodes

Power and space savings

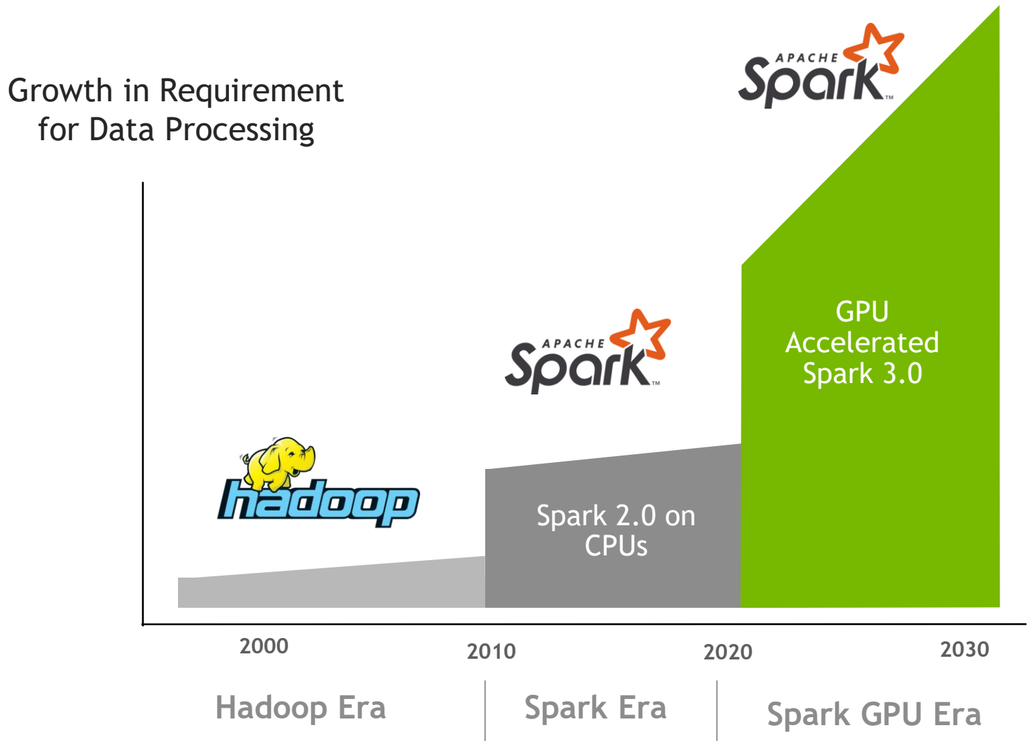
COMPETITIVE DIFFERENTIATION

Faster business decisions, larger data sets

More accurate models & faster iterations

THE LEADING PLATFORM FOR SCALE OUT ANALYTICS

GPU-accelerated Spark 3.x unifies the pipeline for ETL, Machine & Deep Learning



Originally developed at UC Berkeley (2009)

Became top-level Apache Project in 2013

10 years of development /mature & broadly adopted

Optimized for in-memory distributed computing

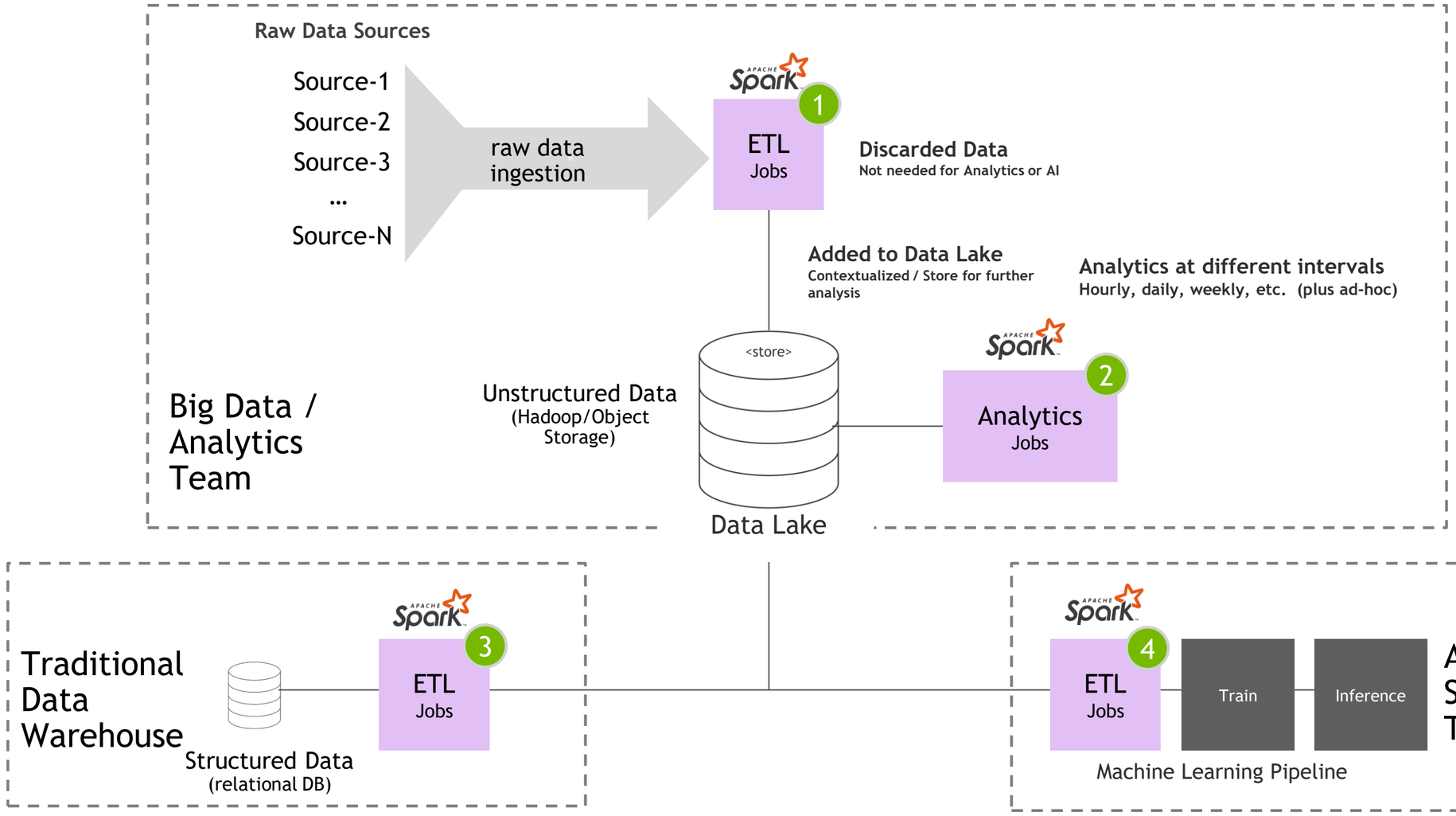
Modern Unified ETL and ML/DL Platform

"These contributions lead to faster data pipelines, model training and scoring for more breakthroughs and insights with Apache Spark 3.0 and Databricks."

Matei Zaharia, creator of Apache Spark and chief technologist at Databricks

APACHE SPARK USAGE IN THE MODERN ENTERPRISE

Use Cases for Apache Spark



Use Cases

1. Raw Data processing into data lake (ETL); typically streaming data
2. Analytics on unstructured or semi-structured data in data lake
3. Loading data from data lake into Enterprise Data Warehouse (ETL)
4. Pre-processing (feature engineering) data from data lake for training

SPARK 3.X ON NVIDIA GPUs

Accelerate data science pipelines without code changes

Faster Execution Time

Accelerate data preparation

Quickly move to next stages of the pipeline

Focus on most-critical activities

Streamline Analytics to AI

Orchestrate end-to-end pipelines

From ETL to model training to visualization

Same infrastructure for Spark and ML/DL frameworks

Reduced Infrastructure Costs

Complete jobs faster with less hardware

Save on-prem and in the cloud

Do more with less

WHY CUSTOMERS WOULD MOVE TO SPARK 3?

Key Features in Spark 3

Integration of ETL and DL into a single pipeline

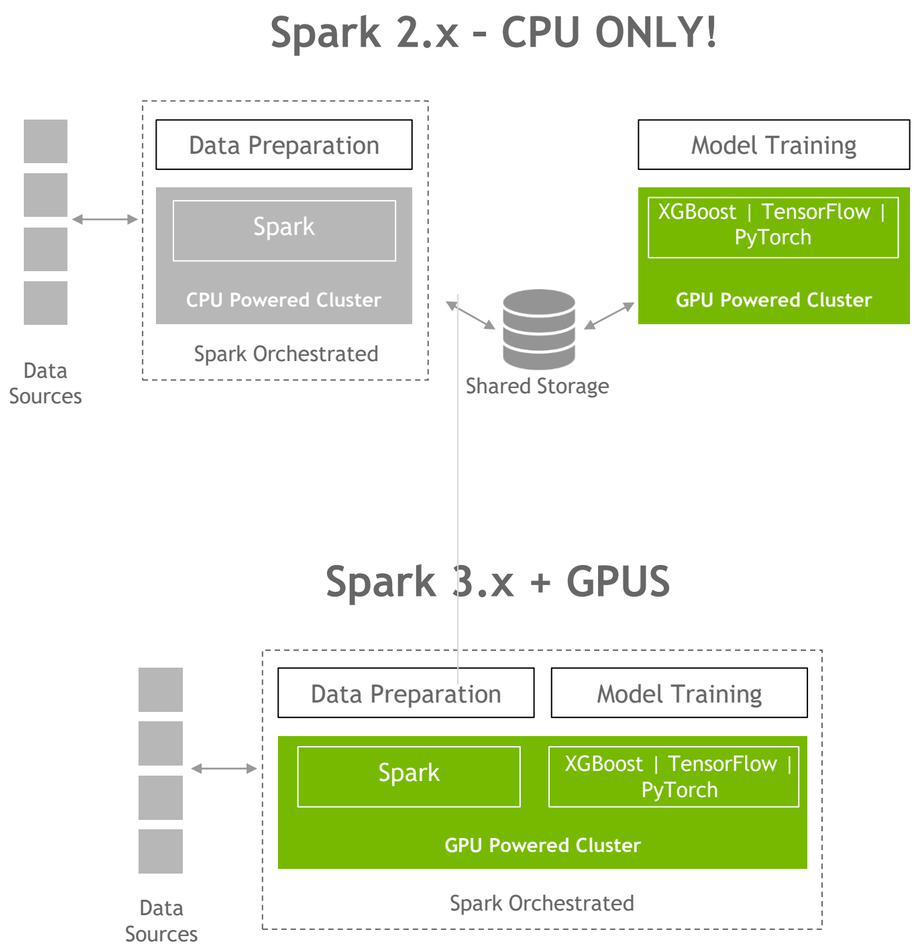
- ❑ New Spark platform features (barriers, etc.) and libraries (Horovod, XGBoost4J) to better support ML/DL frameworks

Performance benefits

- ❑ **Making GPUs First Class Citizens for Acceleration**
- ❑ Adaptive Query Execution for SQL
- ❑ Dynamic Partition Pruning

Other Misc. Benefits

- ❑ Enhanced Graph Support
- ❑ Better Kubernetes Support
- ❑ Improved Language Support



HANDS-ON: NDS QUERY 4 (CPU)

RAPIDS ACCELERATOR FOR APACHE SPARK SOLUTION OVERVIEW

NVIDIA INNOVATIONS IN SPARK 3.X

Accelerate data science pipelines without code changes

RAPIDS Accelerator for Spark 3.0

Intercepts and accelerates SQL and DataFrame operations, dramatically improving ETL performance

Modifications to Spark Components

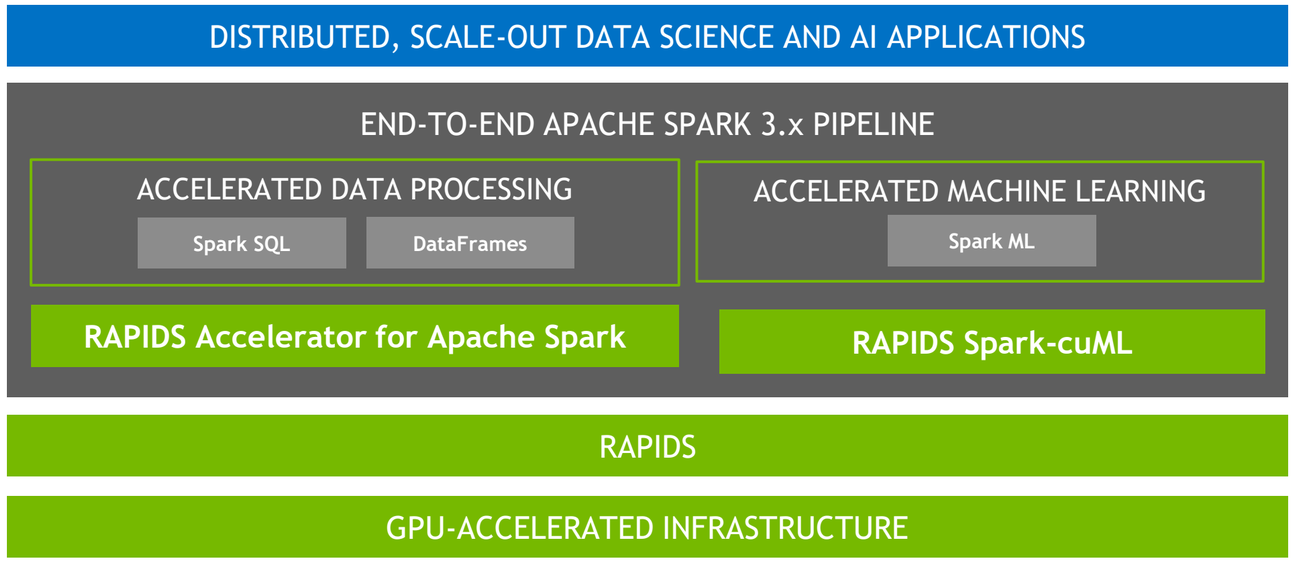
Columnar processing support in the Catalyst query optimizer

Spark shuffle implementation that optimizes the data transfer between Spark processes

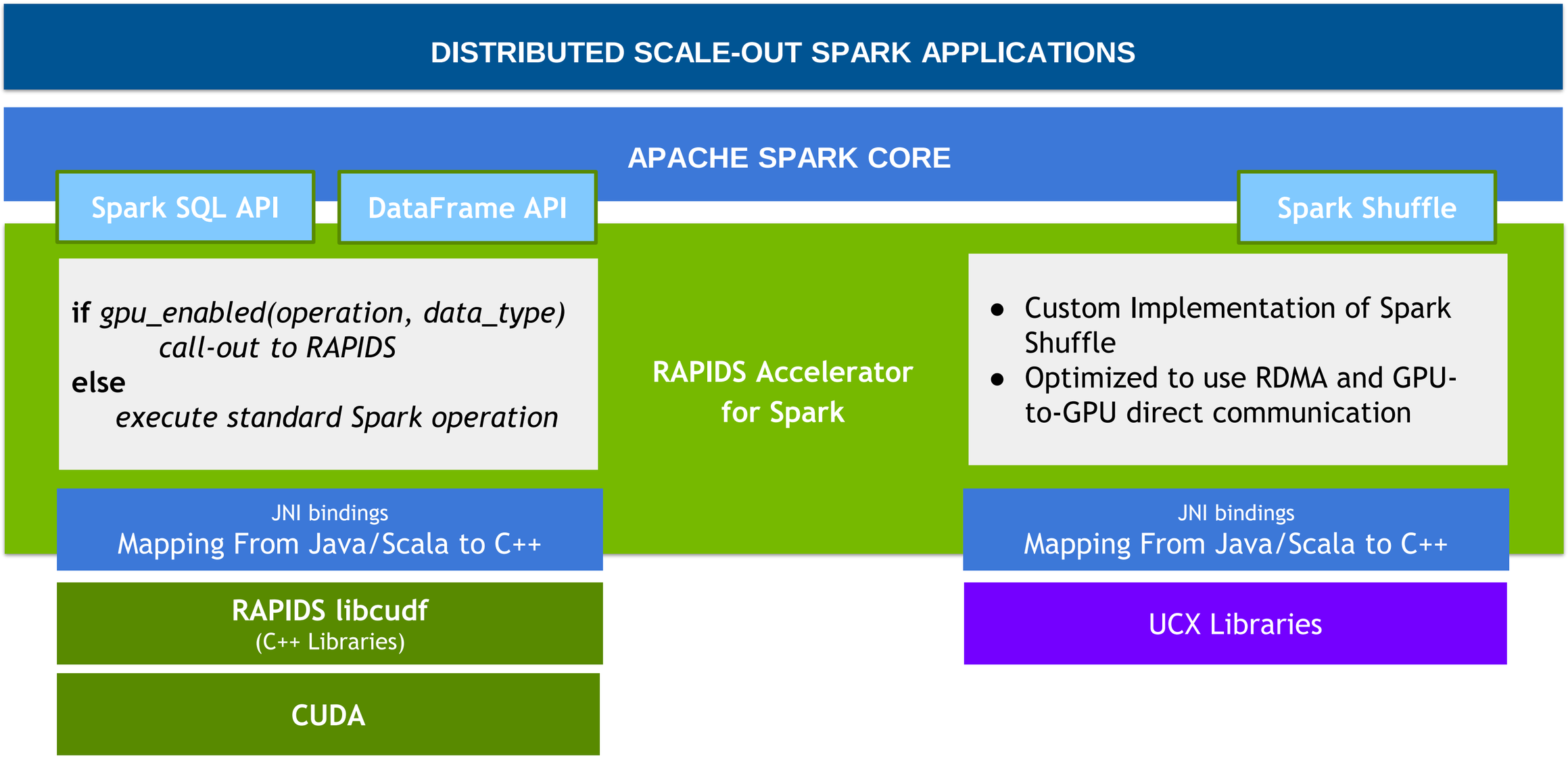
GPU-Aware Scheduling in Spark

Spark 3.0 places GPU-accelerated workloads directly onto servers containing the necessary GPU resources

Spark standalone, YARN, and Kubernetes clusters



RAPIDS ACCELERATOR FOR APACHE SPARK



SPARK SQL & DataFrame COMPILATION FLOW

QUERY

```
SELECT product_id, ds,
max(price) - min(price) AS
range FROM bar GROUP BY
product_id, ds
```

```
bar.groupBy(
    col("product_id"),
    col("ds"))
.agg(
    max(col("price")) -

min(col("price")).alias("range"))
```

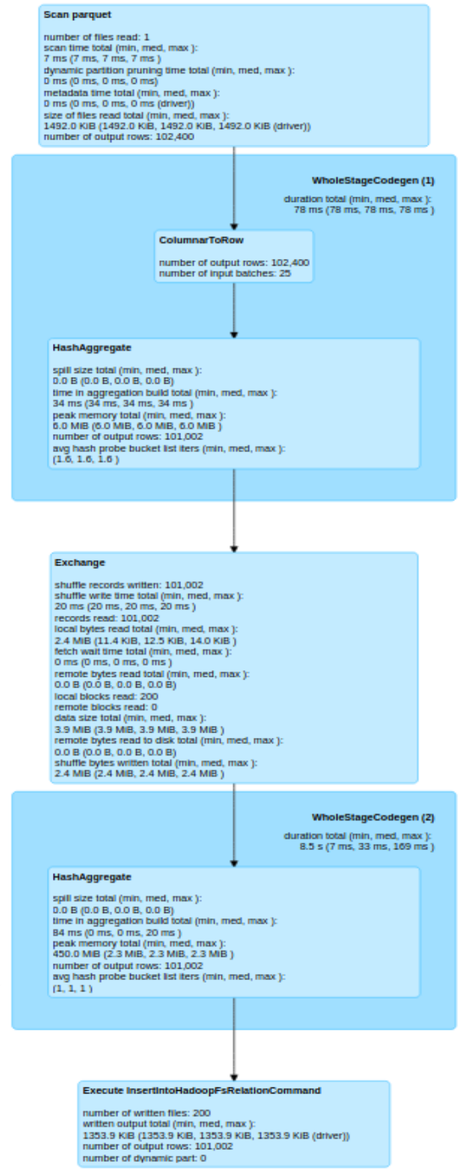
DataFrame

Logical Plan

Physical Plan

RDD[InternalRow]

CPU PHYSICAL PLAN



SPARK SQL & DataFrame COMPILATION FLOW

With RAPIDS ACCELERATOR FOR APACHE SPARK 3.x

QUERY

```
SELECT product_id, ds,
max(price) - min(price) AS
range FROM bar GROUP BY
product_id, ds
```

```
bar.groupBy(
    col("product_id"),
    col("ds"))
.agg(
    max(col("price")) -

min(col("price")).alias("range"))
```

RAPIDS SQL Plugin

DataFrame

Logical Plan

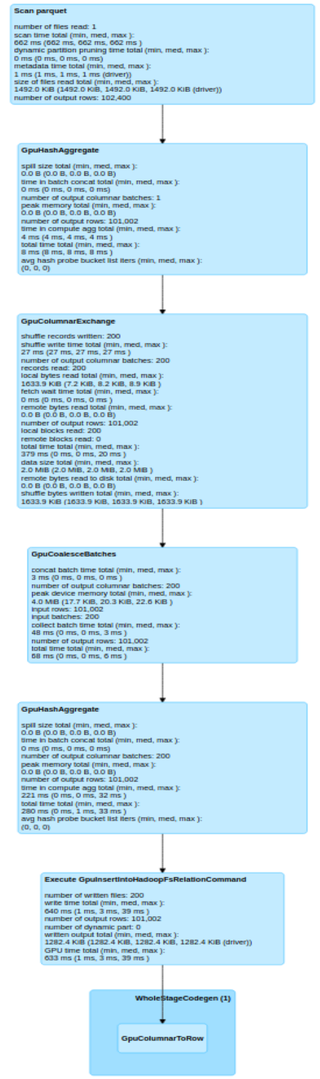
Physical Plan

GPU Physical Plan

RDD[ColumnarBatch]

RDD[InternalRow]

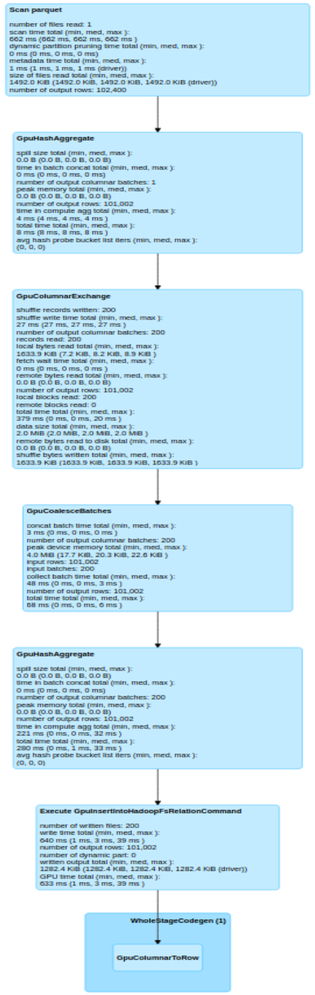
GPU PHYSICAL PLAN



SPARK SQL & DataFrame COMPILATION FLOW

GPU vs CPU

GPU PHYSICAL PLAN



Read Parquet File

First Stage Aggregate

Shuffle Exchange

Combine Shuffle Data

Second Stage Aggregate

Write Parquet File

Convert to Row Format

Read Parquet File

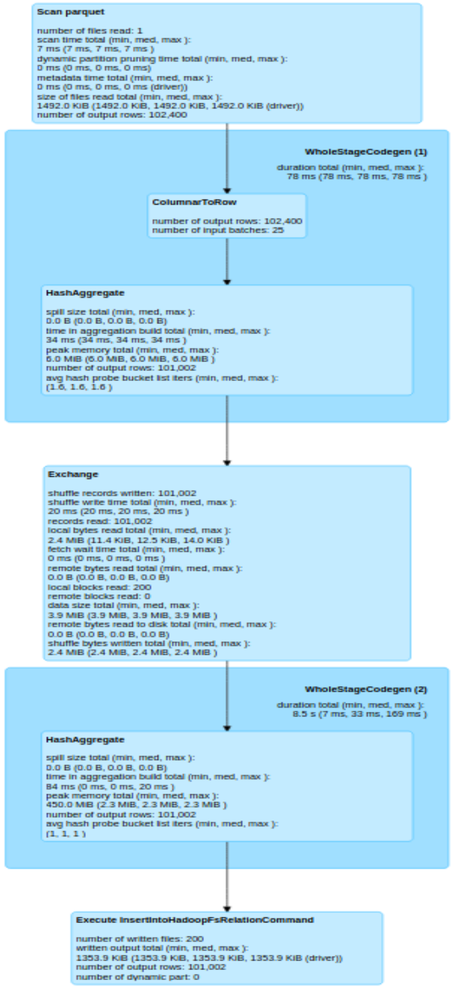
Convert to Row Format

First Stage Aggregate

Shuffle Exchange

Second Stage Aggregate

Write Parquet File



CPU PHYSICAL PLAN

WILL MY SPARK WORKLOAD ACCELERATE WITHOUT CHANGES?

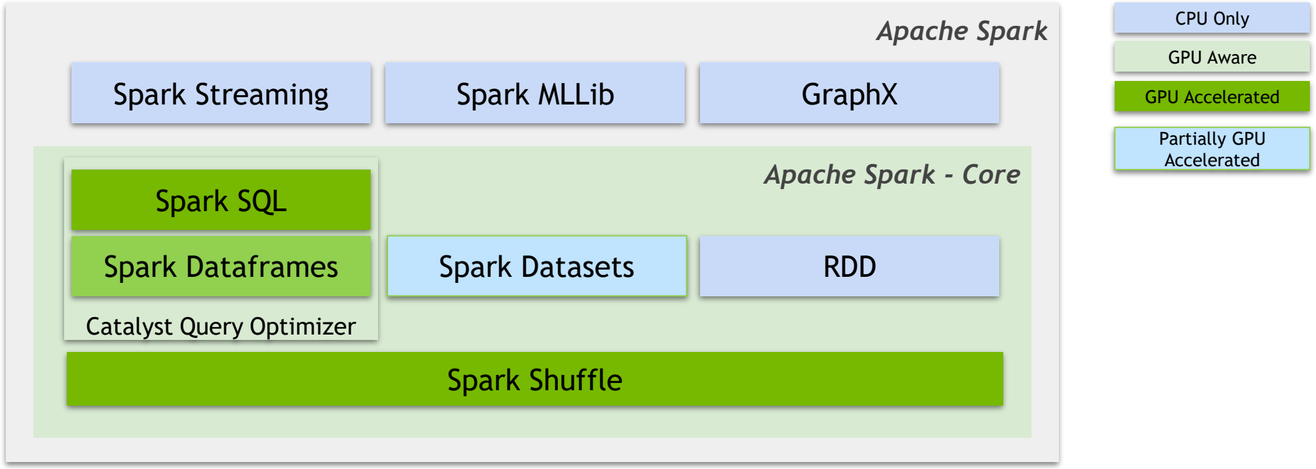
If I know my Spark workload characteristics...

	Accelerates Well on GPUs	Not for GPUs
Data Pipeline Use Cases	• Data Mining, Analytics and BI • Batch processing and writing large datasets to a Data Warehouse • Data extraction, aggregation and feature preparation for ML Training & Inference	• Real-time Streaming Analytics/AI pipeline • Online Transaction Processing (OLTP) • Data Pipeline with custom code
Technical Characteristics	• Batch processing of GB+ data sets • Parquet, ORC, CSV data formats • HDFS, S3-compatible, or V2 data sources • DataFrame/SQL (join, agg, sort, window), Selected Hive & Scala UDFs	• Stream processing • Spark RDD, MLLib, Dataset, GraphX, Streaming libraries

If I am unsure...

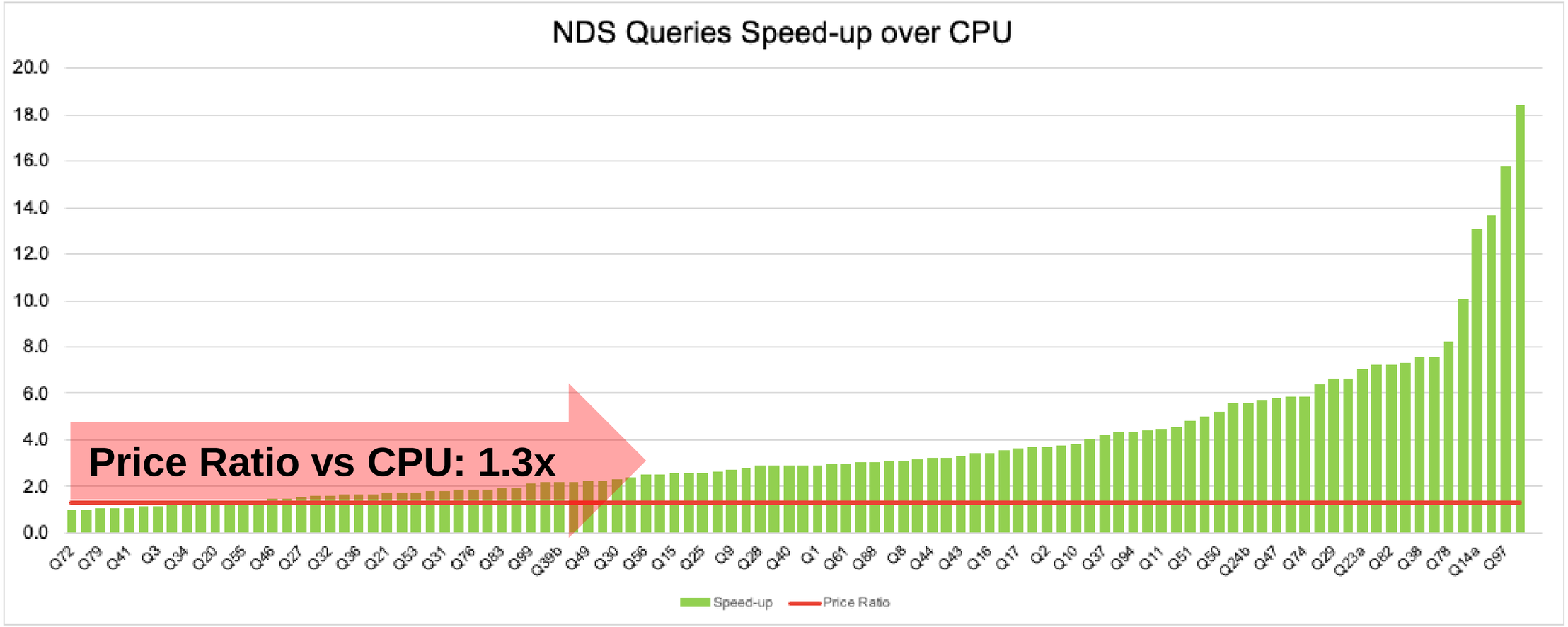
Use the Qualification Tool

- Review Spark history logs from existing CPU jobs
- Understand how much of the workloads could execute on GPUs
- Get tips on optional code optimizations for GPUs



GPU Speed-up for each NDS Query

Based on 103 NVIDIA Decision Support (NDS) benchmark (3TB Dataset)



*CPU-only 4-node cluster: 4xn1-standard-32 (32vCPU, 120GB RAM)
*GPU 4-node cluster: 4xn1-standard-32 (32vCPU, 120GB RAM) and 8xT4 NVIDIA GPU
*NDS stands for NVIDIA Decision Support benchmark that is derived from the TPC-DS benchmark and is used for internal testing. Results from NDS are not comparable to TPC-DS

No code change with 75% cost savings

NDS Queries performance benchmark uses a 3TB dataset with decimal datatype

Data Description	
# of Customers	30M
# of Stores	1350
# of Warehouse	22
# of Web Sales	2.1B Records
# of Store Sales	8.6B Records
# of Catalog Sales	4.3B Records
# of Promotions	1000

Query Type	Description	Sample Queries	Performance (Avg Speed up over CPU)
Interactive	1-3 months of data scanned – Simple join queries	6,10,12,19,20,21, 40,42,52,55	2.5x
Reporting	1 year of data scanned – Simple join queries	1,3,7,13,22,26,27, 30,31,53,89	2.4x
Analytic	Multiple years, customer patterns	2,8,9,11,17,18,24, 34,38,44,59	4.5x
Complex	Multiple table joins, windows, extensive subqueries	4,5,14,23,28,64, 82,94,97	9.2x

*NDS stands for NVIDIA Decision Support benchmark that is derived from the TPC-DS benchmark and is used for internal testing. Results from NDS are not comparable to TPC-DS

HANDS-ON: SPARK UI (CPU)

HANDS-ON: NDS QUERY 67 (CPU)

NVIDIA

Workload Qualification Guide

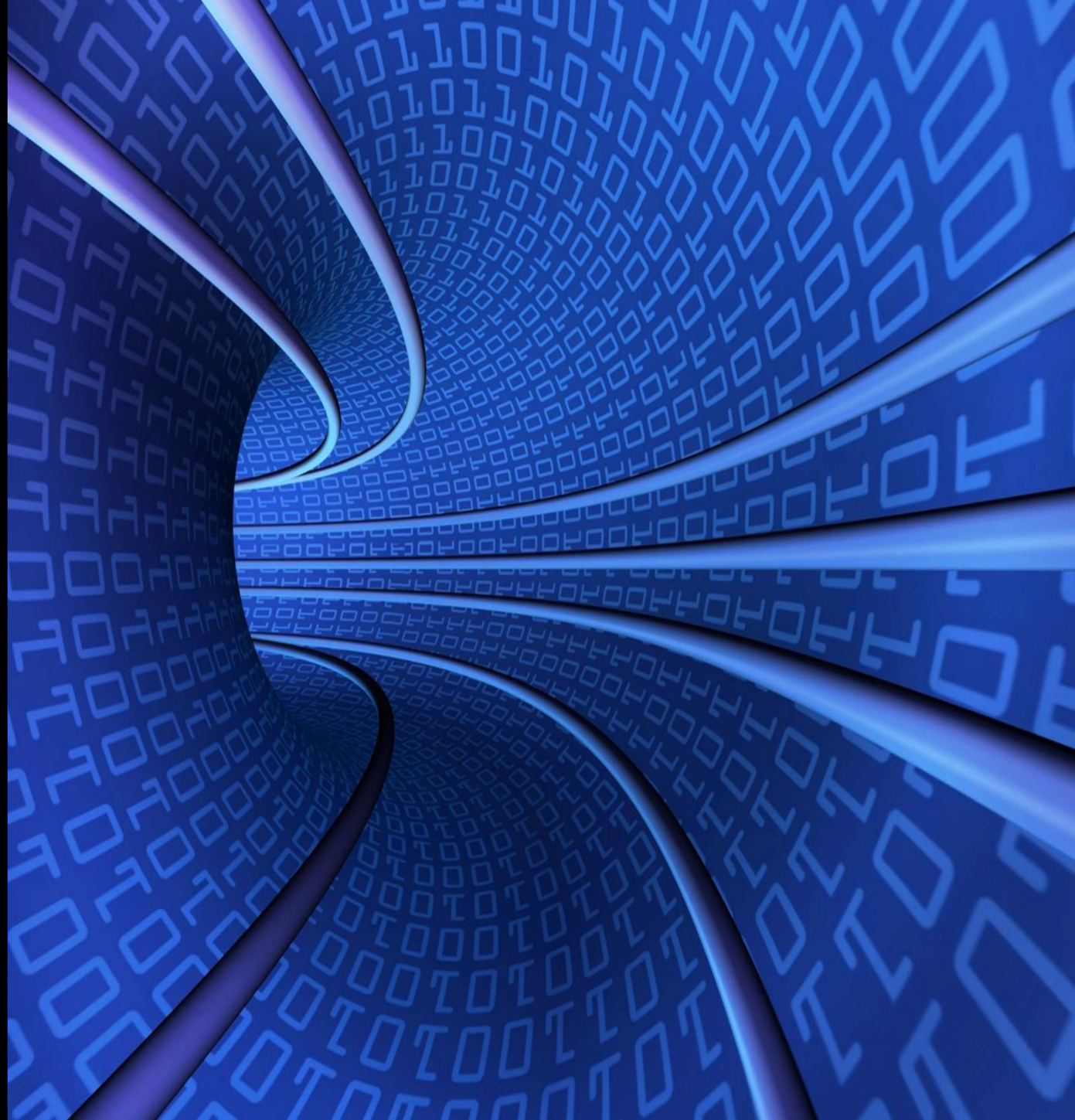
RAPIDS Accelerator for Apache Spark

RAPIDS + Spark

What is the potential impact?

Overview of the workflow qualification tool

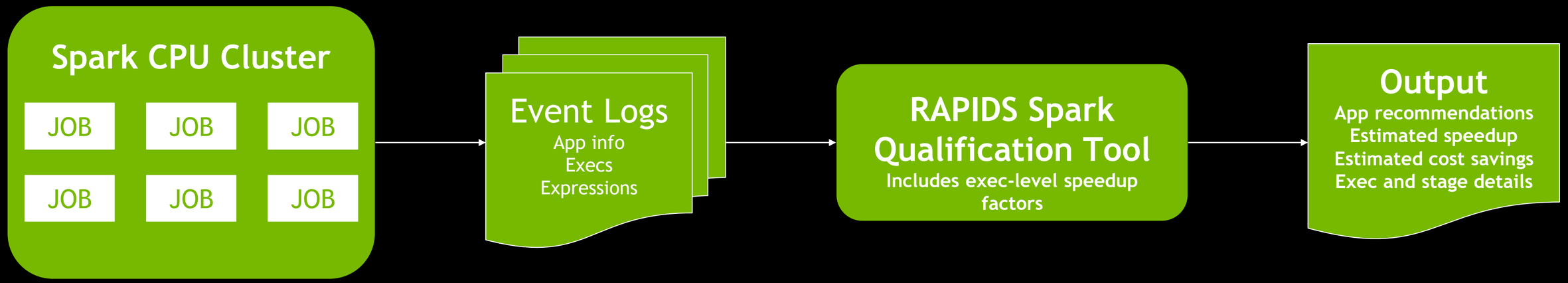
Guide to running the qualification tool and interpreting the output



OVERVIEW OF THE QUALIFICATION TOOL

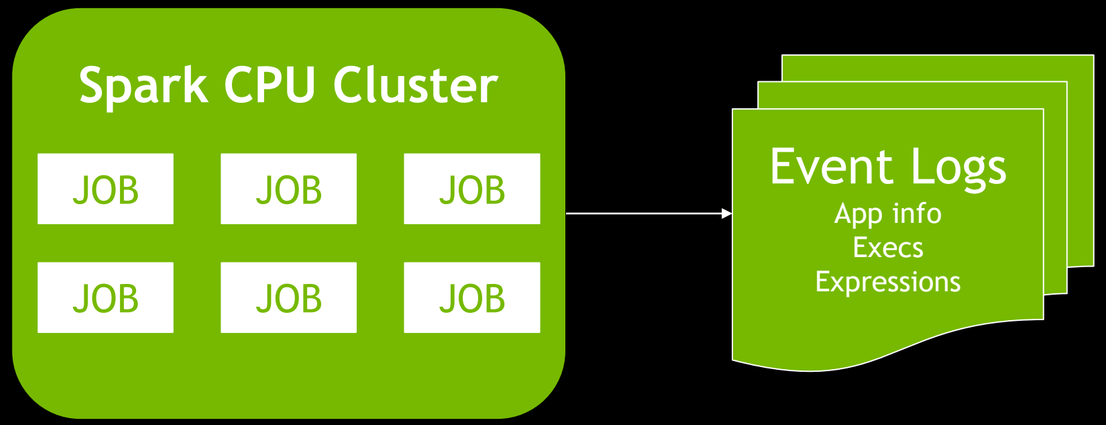
CPU WORKLOAD QUALIFICATION

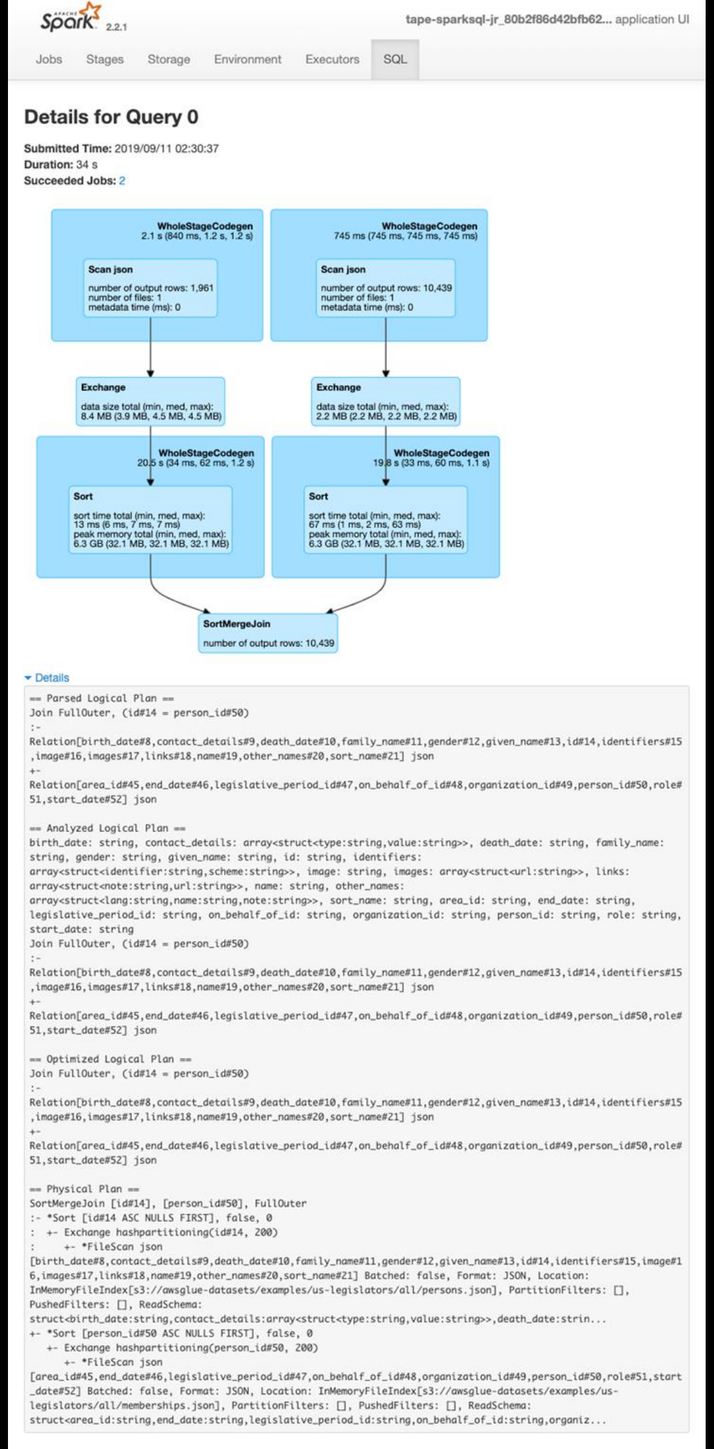
Predicting The Benefit of Spark + GPUs



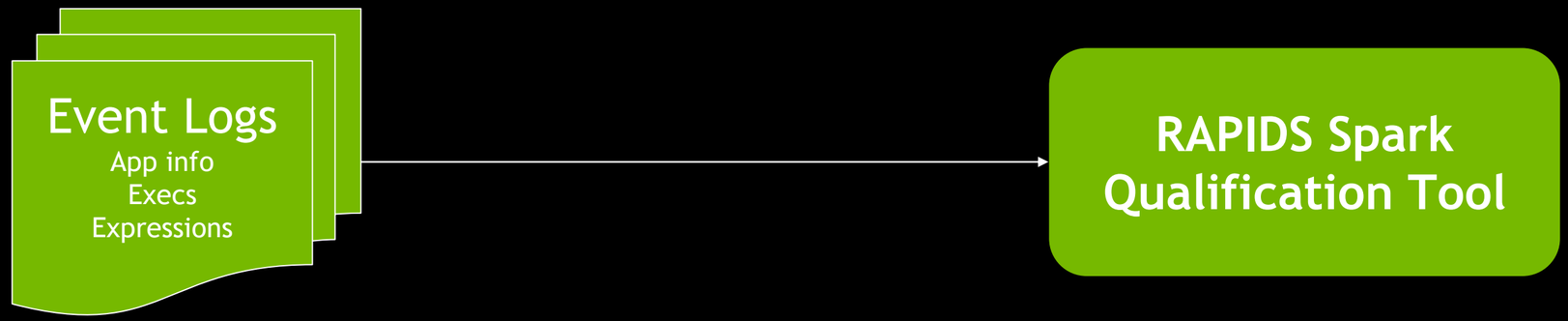
Tool supports event logs from
Spark 2.x and Spark 3.x jobs

SPARK EVENT LOGS





EXEC + EXPRESSION ANALYSIS



Contains details about the logical and physical plan for the job

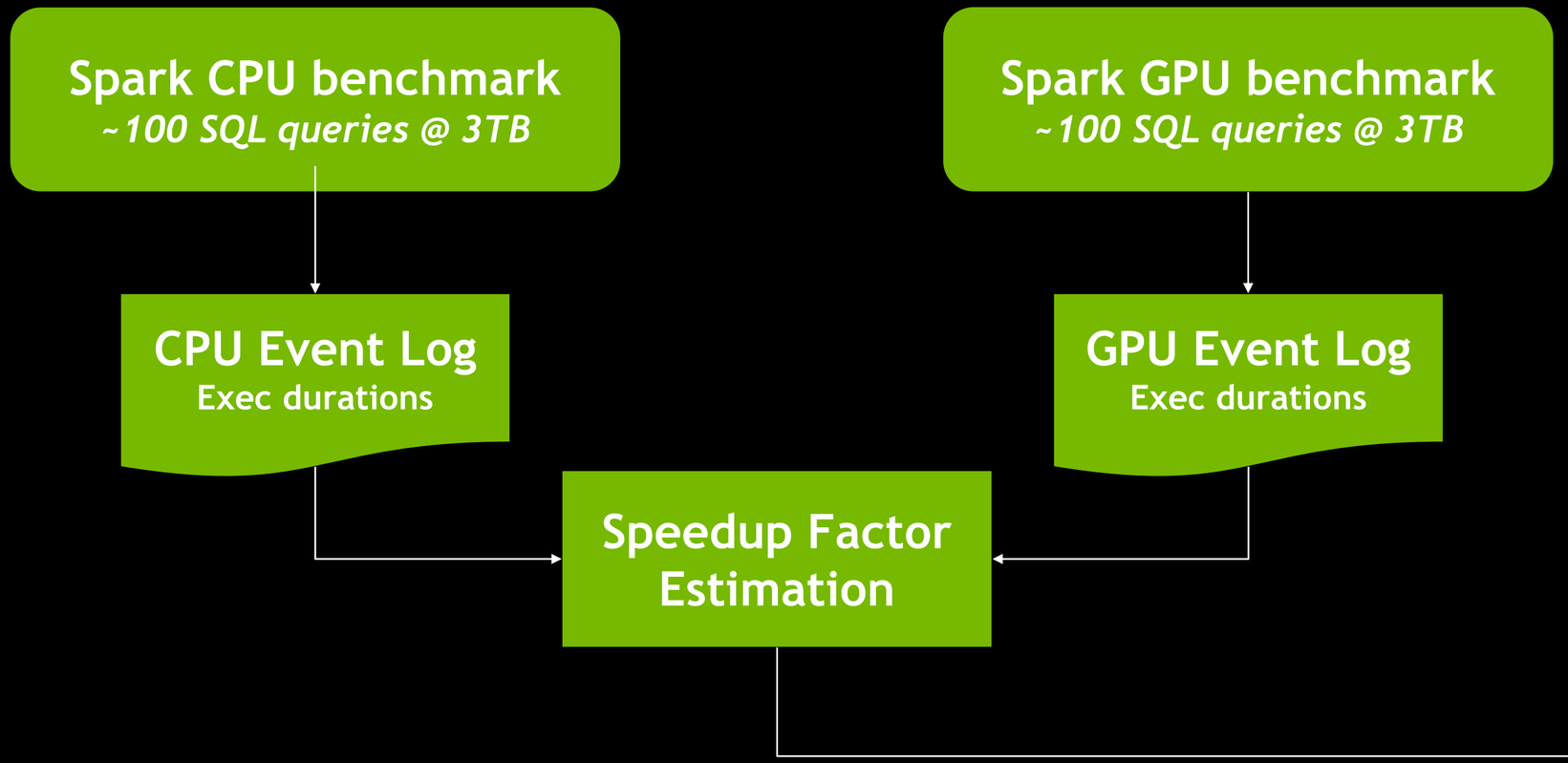
Outputs tree hierarchy for stages, SQL nodes, execs, and expressions

Traverses node trees to analyze

- What nodes (execs + expressions) in a plan tree can run on GPU
- I/O support for read and write formats

For supported nodes, applies GPU speed-up factors per exec + expressions

SPEEDUP FACTOR ESTIMATION



Exec	Speedup
Filter	2.8
Sort	8.0
BroadcastHashJoin	5.1
ShuffleExchange	4.2
HashAggregate	4.5
SortMergeJoin	22.7

OUTPUT + RECOMMENDATIONS

Contains speedup factors per exec and expression that are supported on GPU

For each stage, projects duration on GPU using speedup factors relevant for execs and expressions in the stage plan tree

Recommendation based on estimated speedup

- If > 2.5, Strongly Recommended
- If > 1.3, Recommended
- If ⇐ 1.3, Not Recommended

Cost savings (only applicable for Dataproc currently) is calculated based on existing CPU cluster shape and projected GPU cluster shape

EXAMPLE APP QUALIFICATION

Inputs

Event Log
- Stage 1: 5 minutes
  - HashAggregate: 2 minutes
    - explode expression
  - Sort: 1 minute
  - Project: 1 minute
- Stage 2: 4 minutes
  - HashAggregate: 2 minutes
    - merge_sum expression
  - Sample: 1 minute
  - Project: 1 minute

Speedup Factors for GPU
- Execs
  - HashAggregate: 5.0
  - Sort: 4.2
  - Project: 2.5
  - Limit: 3.1
  - Sample: 3.6
- Expressions
  - explode: supported
  - merge_sum: not supported

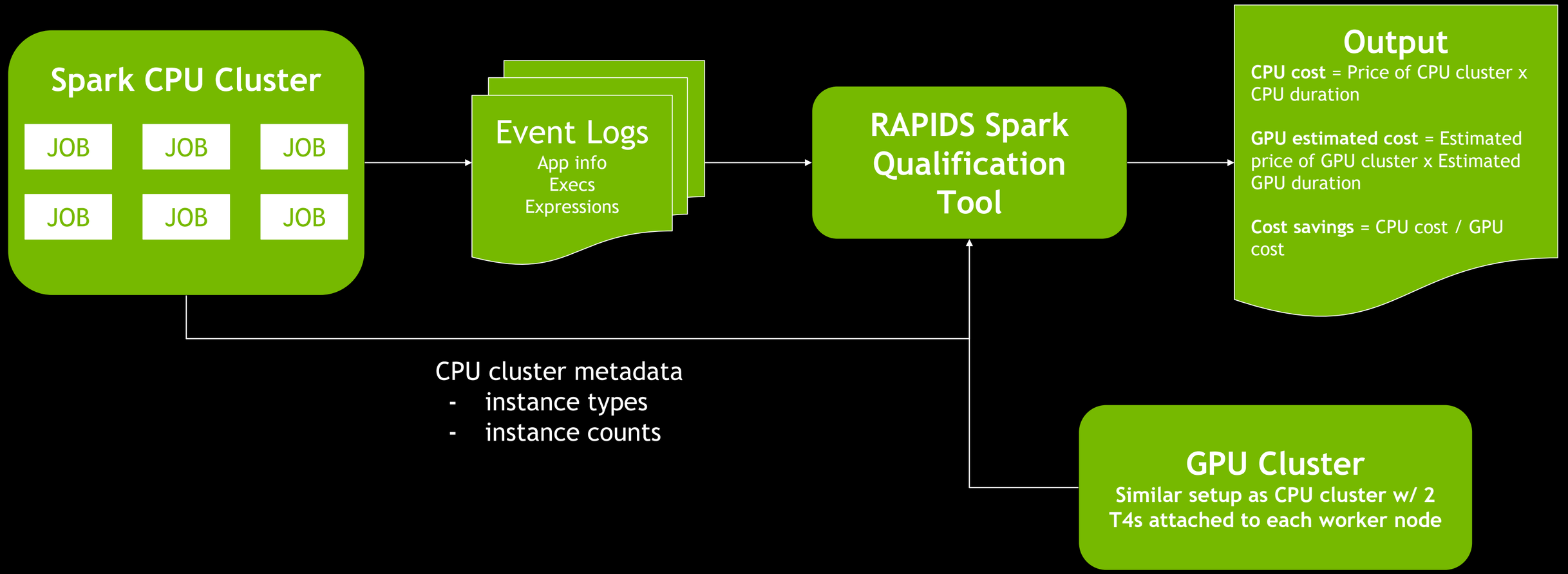
Speedup Estimation

Stage 1
- HashAggregate (supported): 2m -> 24s
- Sort (supported): 1m -> 13s
- Project (supported): 1m -> 24s
- Total: 5m -> 1m 1s = 4.9x speedup

Stage 2
- HashAggregate (not supported): 2m -> 2m
- Sample(supported): 1m -> 17s
- Project (supported): 1m -> 24s
- Total: 5m -> 2m 41s = 1.9x speedup

Total
- **CPU duration: 10m**
- **Estimated GPU duration: 3m 42s**
- **Estimated GPU speedup: 2.7x speedup**

COST SAVING ESTIMATION



RUNNING THE QUALIFICATION TOOL

QUALIFICATION USER GUIDE

User Tools

Prerequisites

- Install package: **pip install spark-rapids-user-tools**

Execution

```
> spark_rapids_user_tools onprem qualification \
    --eventlogs <file-path>
```

Options (optional)

- --filter-apps = filtering criteria of applications listed in CLI output; one of *NONE*, *recommended*, *savings* (default)
- --output-folder = base output directory, defaults to current directory

Documentation: https://pypi.org/project/spark-rapids-user-tools/

QUALIFICATION USER GUIDE

User Tools CLI Summary Output

App Name	Recommendation	Estimated GPU Speedup	Estimated GPU Duration(s)	App Duration(s)
Customer App #1	Strongly Recommended	3.66	651.24	2384.32
Sales App #1	Strongly Recommended	3.14	89.61	281.62
Sales App #2	Strongly Recommended	3.12	300.39	939.21
Customer App #2	Strongly Recommended	2.55	698.16	1783.65

Report Summary:

Total applications	4
RAPIDS candidates	4
Overall estimated speedup	3.10

INTERPRETING THE QUALIFICATION OUTPUT

QUALIFICATION USER GUIDE

Interpreting The Detailed Output

Summary Info

- Recommendation
- Estimated GPU Speed-up
- Estimated GPU Time Saved
- Estimated Cost Savings (only with User Tools)

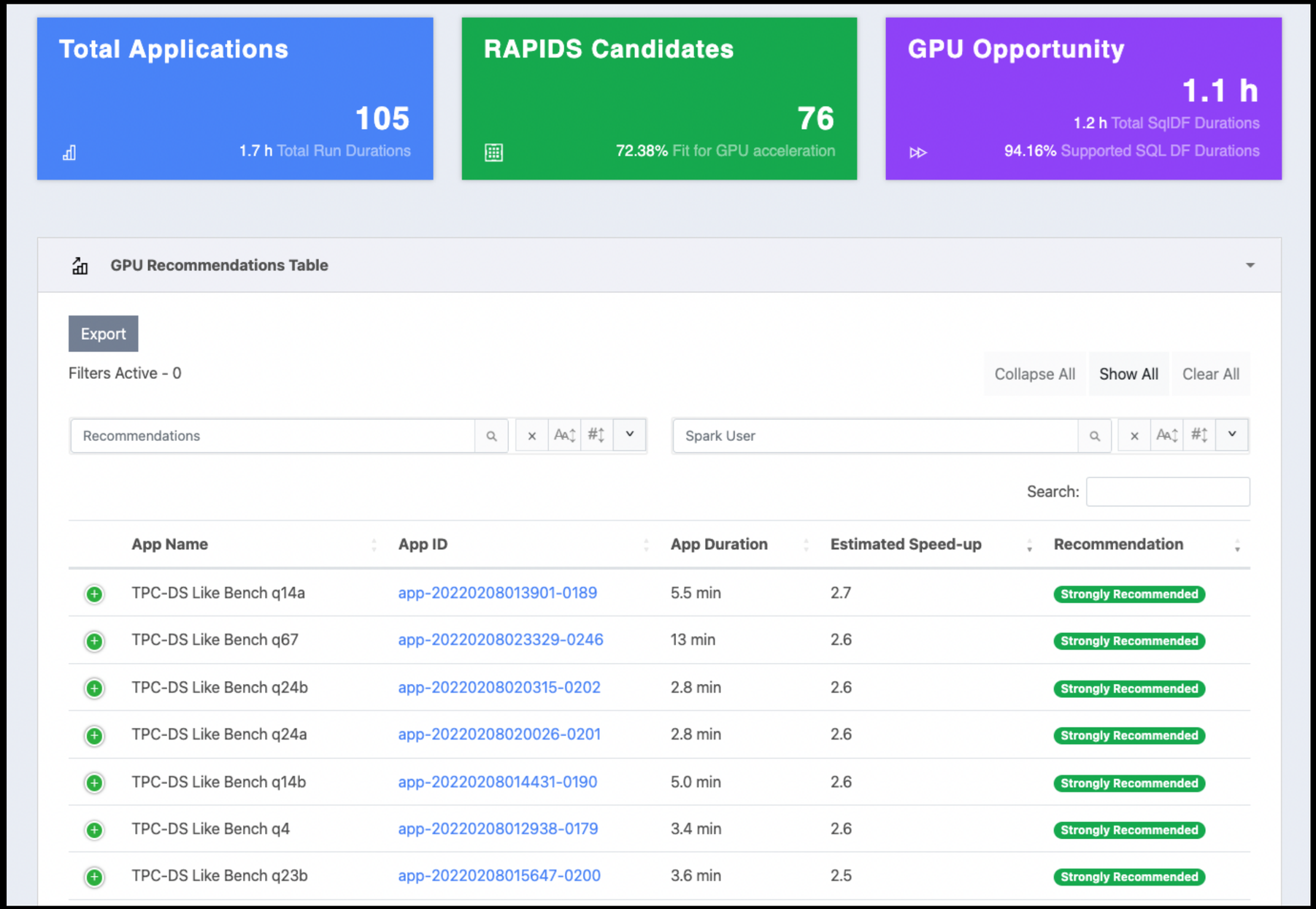
Stages Output

- App ID
- Stage ID
- Average Speedup Factor: the average estimated speed-up of all the operators in the given stage.
- Stage Task Duration: amount of time spent in tasks of SQL Dataframe operations for the given stage.
- Unsupported Task Duration: sum of task durations for the unsupported operators. For more details, see Supported Operators.
- Stage Estimated: True or False indicates if we had to estimate the stage duration.

Execs Output

- App ID
- SQL ID
- Exec Name: example Filter, HashAggregate
- Expression Name
- Task Speedup Factor
- Exec Duration: wall-Clock time measured since the operator starts till it is completed.
- SQL Node Id
- Exec Is Supported: whether the Exec is supported by RAPIDS or not. Please refer to the Supported Operators section.
- Exec Stages: an array of stage IDs
- Exec Children
- Exec Children Node Ids
- Exec Should Remove: whether the Op is removed from the migrated plan.

HTML Report


Total Applications
105
1.7 h Total Run Durations
RAPIDS Candidates
76
72.38% Fit for GPU acceleration
GPU Opportunity
1.1 h
1.2 h Total SqlDF Durations
94.16% Supported SQL DF Durations
GPU Recommendations Table
Export
Filters Active - 0
Collapse All
Show All
Clear All
Recommendations
Spark User
Search:
App Name
App ID
App Duration
Estimated Speed-up
Recommendation
TPC-DS Like Bench q14a
app-20220208013901-0189
5.5 min
2.7
Strongly Recommended
TPC-DS Like Bench q67
app-20220208023329-0246
13 min
2.6
Strongly Recommended
TPC-DS Like Bench q24b
app-20220208020315-0202
2.8 min
2.6
Strongly Recommended
TPC-DS Like Bench q24a
app-20220208020026-0201
2.8 min
2.6
Strongly Recommended
TPC-DS Like Bench q14b
app-20220208014431-0190
5.0 min
2.6
Strongly Recommended
TPC-DS Like Bench q4
app-20220208012938-0179
3.4 min
2.6
Strongly Recommended
TPC-DS Like Bench q23b
app-20220208015647-0200
3.6 min
2.5
Strongly Recommended

HANDS-ON: QUALIFICATION TOOL

HANDS-ON: NDS QUERY 4 (GPU)

HANDS-ON: NDS QUERY 67 (GPU)

GPU Job Profiling + Tuning

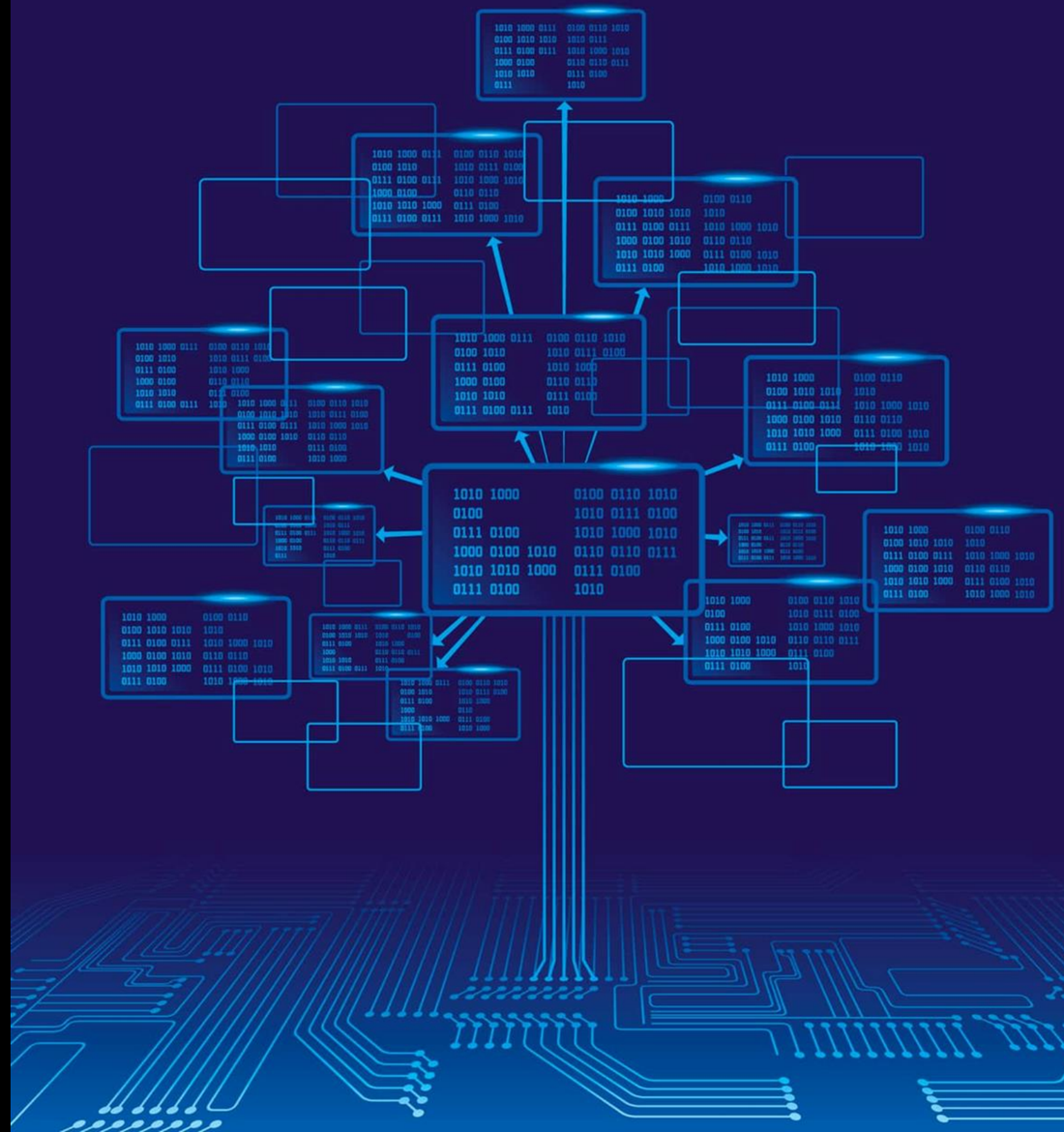
RAPIDS Accelerator for Apache Spark

RAPIDS + Spark

How can I optimize my jobs?

Overview of the GPU job profiling tool

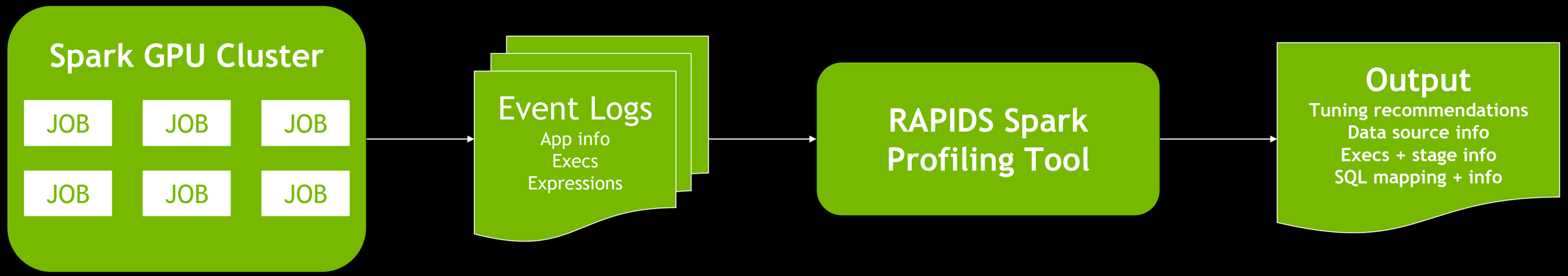
Guide to running the profiling tool and interpreting the output



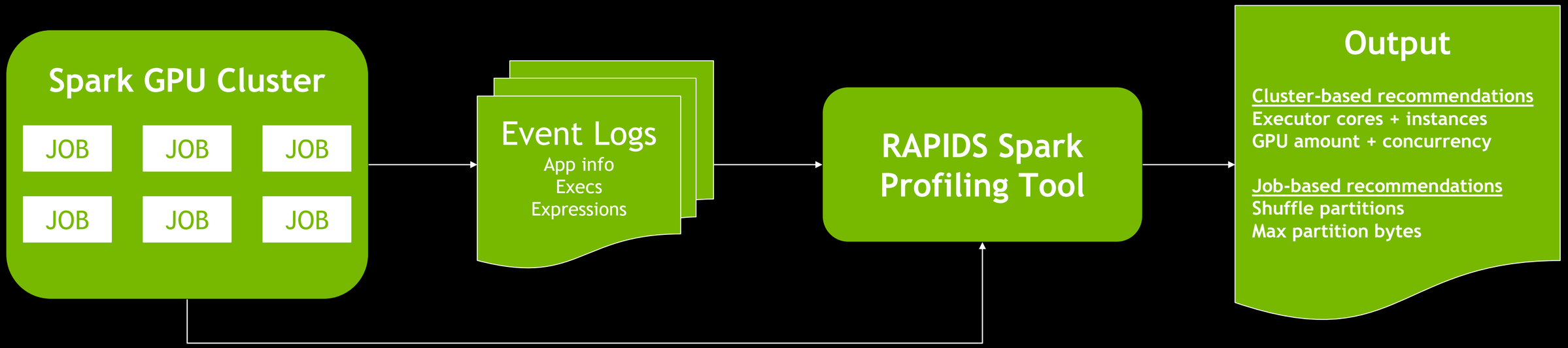
OVERVIEW OF THE PROFILING TOOL

GPU JOB PROFILING + TUNING

Optimizing Performance For Spark Jobs on GPUs



JOB TUNING RECOMMENDATIONS



GPU cluster metadata
- CPU memory
- CPU cores
- GPU memory
- GPU count

RUNNING THE PROFILING TOOL

PROFILING USER GUIDE

User Tools

Prerequisites

- Install package: **pip install spark-rapids-user-tools**

Execution

```
> spark_rapids_user_tools onprem profiling \
    --eventlogs <file-path> \
    --cluster_conf <cluster-metadata>
```

Options (optional)

- --output-folder = base output directory, defaults to current directory

Documentation: https://pypi.org/project/spark-rapids-user-tools/

PROFILING USER GUIDE

User Tools CLI Summary Output

App Name	Recommendations
Customer App #1	--conf spark.executor.cores=8
	--conf spark.executor.instances=10
	--conf spark.executor.memory=51g
	--conf spark.executor.memoryOverhead=7.10g
	--conf spark.rapids.memory.pinnedPool.size=2g
	--conf spark.rapids.sql.concurrentGpuTasks=4
	--conf spark.sql.files.maxPartitionBytes=2g
	--conf spark.sql.shuffle.partitions=400
	--conf spark.task.resource.gpu.amount=0.125
Customer App #2	--conf spark.executor.cores=4
	--conf spark.executor.instances=20
	--conf spark.executor.memory=51g
	--conf spark.executor.memoryOverhead=7.10g
	--conf spark.rapids.memory.pinnedPool.size=2g
	--conf spark.rapids.sql.concurrentGpuTasks=4
	--conf spark.sql.files.maxPartitionBytes=1g
	--conf spark.sql.shuffle.partitions=1000
	--conf spark.task.resource.gpu.amount=0.25

INTERPRETING THE PROFILING OUTPUT

PROFILING USER GUIDE

Interpreting The Detailed Output

Application Info
- App ID
- Start and end time
- Spark version

Executor Info
- Count + cores
- Memory metrics
- GPU count + tasks

Data Source Info
- App ID
- SQL ID
- Format
- Location
- Filters
- Schema

SQL to Stage Info
- App ID
- SQL ID
- Stage ID
- Stage duration
- SQL nodes

SQL Plan Metrics
- App ID
- SQL ID
- Node name
- Metric key and value

Job Tuning Recommendations
- App ID
- Setting(s) to tune
- Comments

TOP 5 CONFIGURATION SETTINGS

Setting	Definition	Default	Recommendation	Tuning
spark.executor.instances	Number of executors for the Spark application	2	Set to the total number of GPUs in your cluster	No tuning is needed To validate that the proper number of executors are launched that match the number of GPUs in the cluster, you can check the Spark UI's executors page
spark.executor.cores	Number of cores to use on each executor, determines number of tasks available to run concurrently	1	Set to the number of CPU cores in a single worker node divided by the number of GPUs per worker node, max of 16 Example: if your worker nodes have 16 cores and 2 GPUs attached to each worker, then set to 8	No tuning is needed
spark.task.resource.gpu.amount	Amount of GPU to allocate for each task	1	Set to 1 divided by spark.executor.cores setting Example: if spark.executor.cores is set to 8, then set spark.task.resource.gpu.amount to 0.125	No tuning is needed If executors are running less tasks than cores, check this configuration in case it is set incorrectly
spark.rapids.sql.concurrentGpuTasks	Number of tasks that are actively sharing the GPU	2	Set it to the amount of GPU memory / 8 GB, and round down; maximum value should be 4 Examples ● For a T4 16GB GPU, dividing by 8GB = 2 so set to 2 ● For an A100 40GB GPU, dividing by 8GB = 5 but max is 4 so set to 4	If out of memory errors are encountered, reduce value If no out of memory errors are encountered, consider increasing if the number of concurrent GPU tasks multiplied by 8 is less than the amount of GPU memory
spark.sql.shuffle.partitions	Number of partitions produced between Spark shuffle stages	200	Start with default If AQE is on, this can be dynamically adjusted	If there are too few partitions, then a task may run out of memory as it will require a larger amount of data to be processed. If there are too many partitions, then the partition processing overhead will have a negative impact on performance.

HANDS-ON: PROFILING TOOL

HANDS-ON: TUNING

CONCLUSION