SPARK+AI SUMMIT 2020

Organized by databricks

Deep Dive into GPU Support in Apache Spark 3.x

Robert Evans and Jason Lowe

NVIDIA

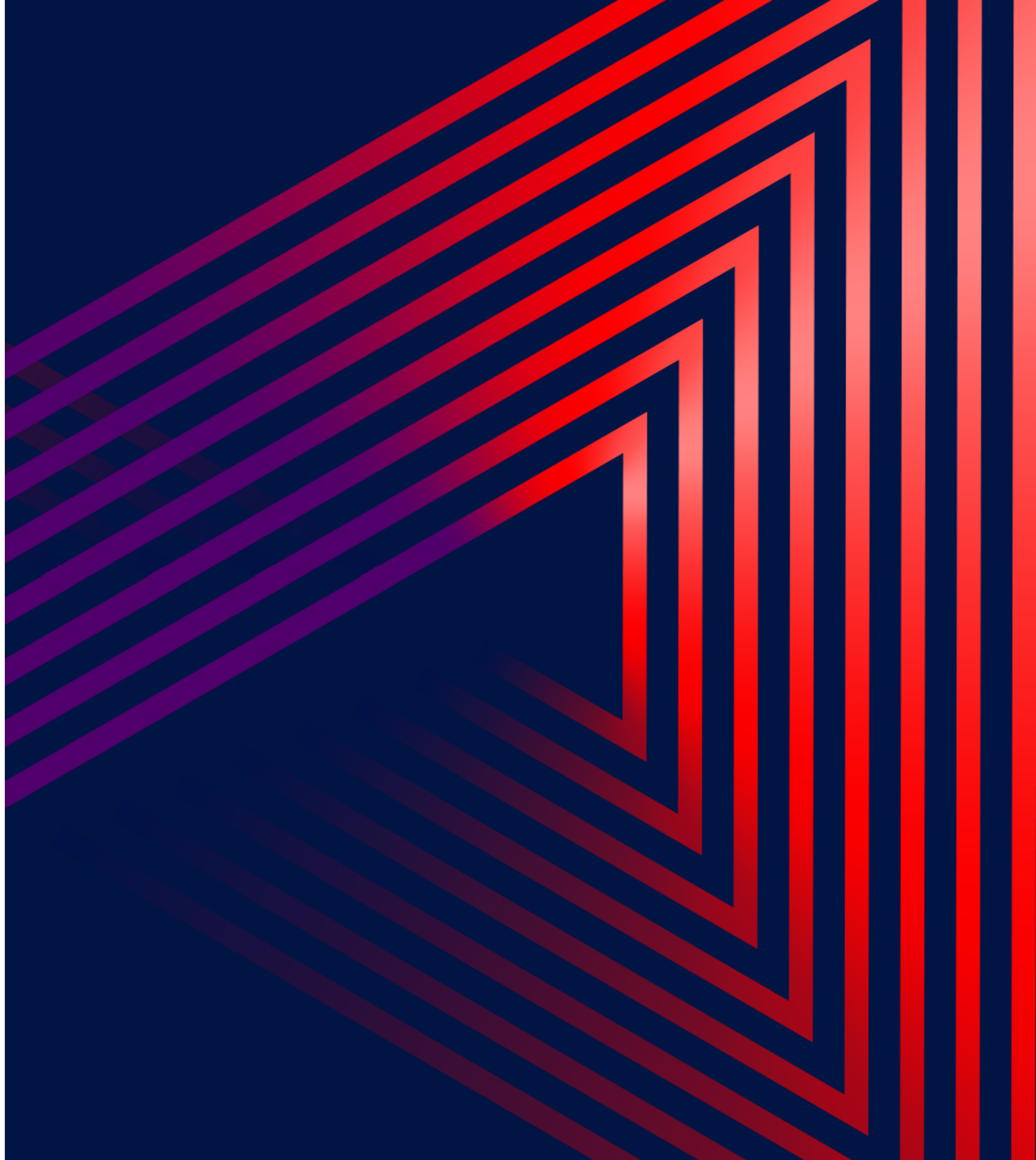
Agenda

GPU Features in Apache Spark 3

Accelerated SQL/DataFrame

Accelerated Shuffle

What's Next



GPU Features in Apache Spark 3

Accelerator-Aware Scheduling

GPUs are now a schedulable resource

- SPARK-24615
- Request resources
  - Executor
  - Driver
  - Task
- Resource discovery
- API to determine assignment
- Supported on YARN, Kubernetes, and Standalone

GPU Scheduling Example

```
./bin/spark-shell  --master yarn --executor-cores 2 \
  --conf spark.driver.resource.gpu.amount=1 \
  --conf spark.driver.resource.gpu.discoveryScript=/opt/spark/getGpuResources.sh \
  --conf spark.executor.resource.gpu.amount=2 \
  --conf spark.executor.resource.gpu.discoveryScript=./getGpuResources.sh \
  --conf spark.task.resource.gpu.amount=1 \
  --files examples/src/main/scripts/getGpusResources.sh
```

GPU Discovery Script Example

```
#!/bin/bash
#
# Outputs a JSON formatted string that is expected by the
# spark.{driver/executor}.resource.gpu.discoveryScript config.
#
# Example output: {"name": "gpu", "addresses":["0","1","2","3","4","5","6","7"]}

ADDRS=$(nvidia-smi --query-gpu=index --format=csv,noheader \
        | sed -e :a -e N -e'$!ba' -e 's/\n/","/g')
echo {\"name\": \"gpu\", \"addresses\":[\"$ADDRS\"]}
```

GPU Assignments API

```
// Task API
val context = TaskContext.get()
val resources = context.resources()
val assignedGpuAddrs = resources("gpu").addresses
// Pass assignedGpuAddrs into TensorFlow or other AI code


// Driver API
scala> sc.resources("gpu").addresses
Array[String] = Array(0)
```

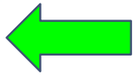
GPU Scheduling UI

Spark 3.0.0-SNAPSHOT | Jobs | Stages | Storage | Environment | Executors | **Spark shell** application UI

Executors

Show Additional Metrics

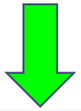
- [ ] Select All
- [ ] On Heap Memory
- [ ] Off Heap Memory
- [x] Resources



Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(2)	0	0.0 B / 8.7 GiB	0.0 B	2	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(2)	0	0.0 B / 8.7 GiB	0.0 B	2	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0

Executors

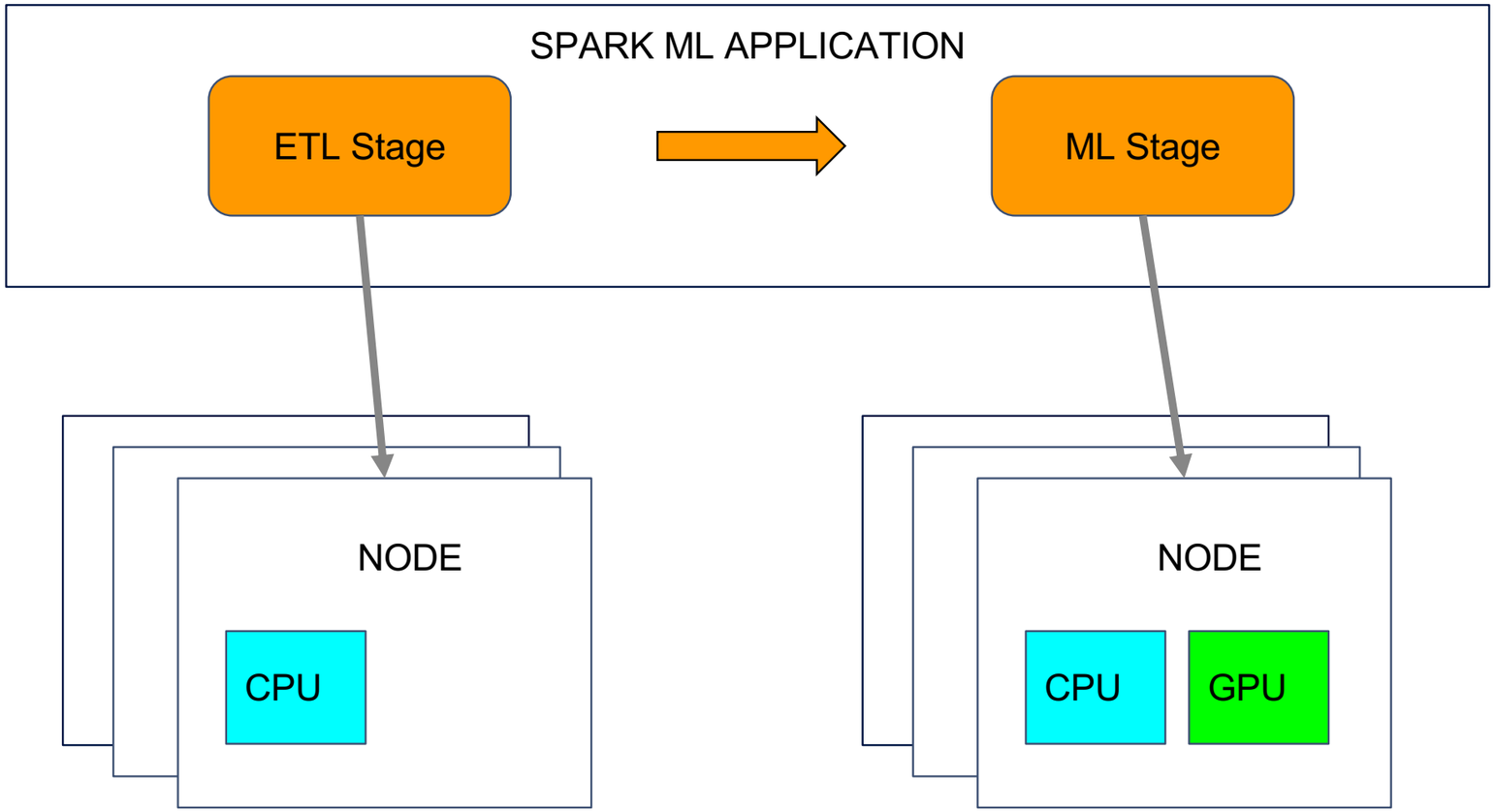


Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Resources	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	10.28.9.112:42305	Active	0	0.0 B / 8.4 GiB	0.0 B	0		0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
1	tomg-x299:37047	Active	0	0.0 B / 366.3 MiB	0.0 B	2	gpu: [0, 1]	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

Showing 1 to 2 of 2 entries Previous 1 Next

Stage Level Scheduling



Stage Level Scheduling

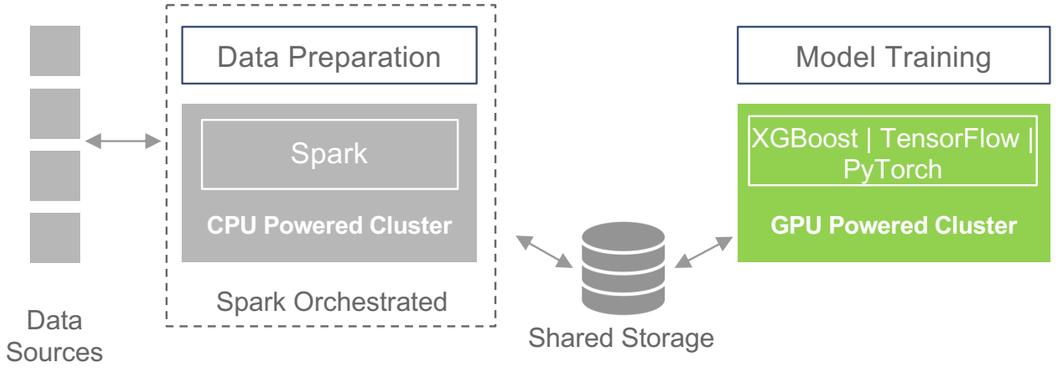
- SPARK-27495
- Specify resource requirements per RDD operation
  - Spark dynamically allocates containers to meet resource requirements
  - Spark schedules tasks on appropriate containers
- Coming soon in Spark 3.1

SQL Columnar Processing

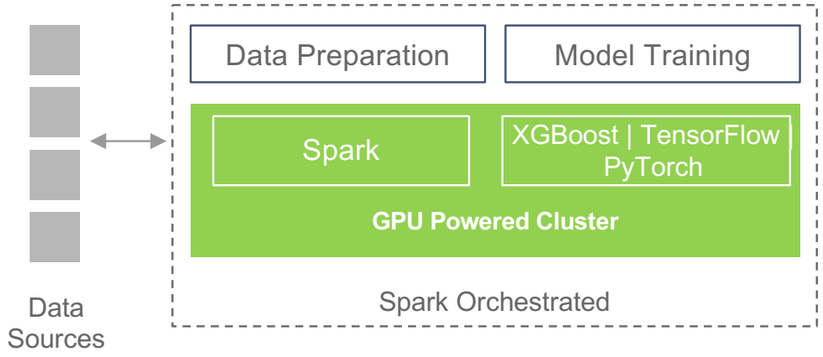
- SPARK-27396
- Catalyst API for columnar processing
  - Plugins can modify query plan with columnar operations
  - Plan nodes can exchange `RDD[ColumnarBatch]` instead of `RDD[Row]`
- Enables efficient processing by vectorized accelerators
  - SIMD
  - FPGA
  - GPU

Spark 3 with Project Hydrogen

Spark 2.x



Spark 3

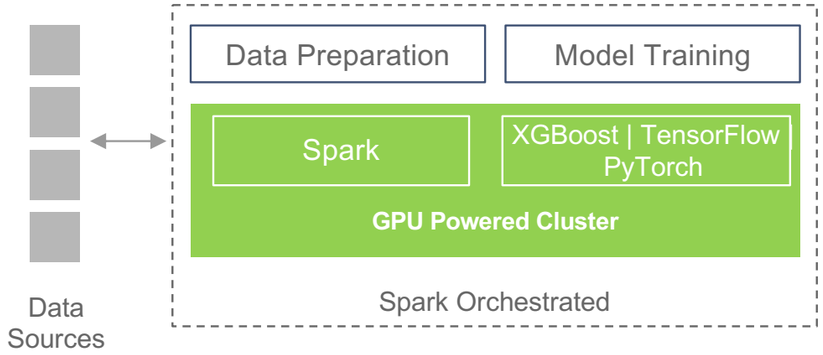


Spark 3 with Project Hydrogen

Enabling end-to-end acceleration

- Single pipeline
  - Ingest
  - Data preparation
  - Model Training
- Infrastructure is consolidated and simplified
- ETL can be GPU-accelerated

Spark 3



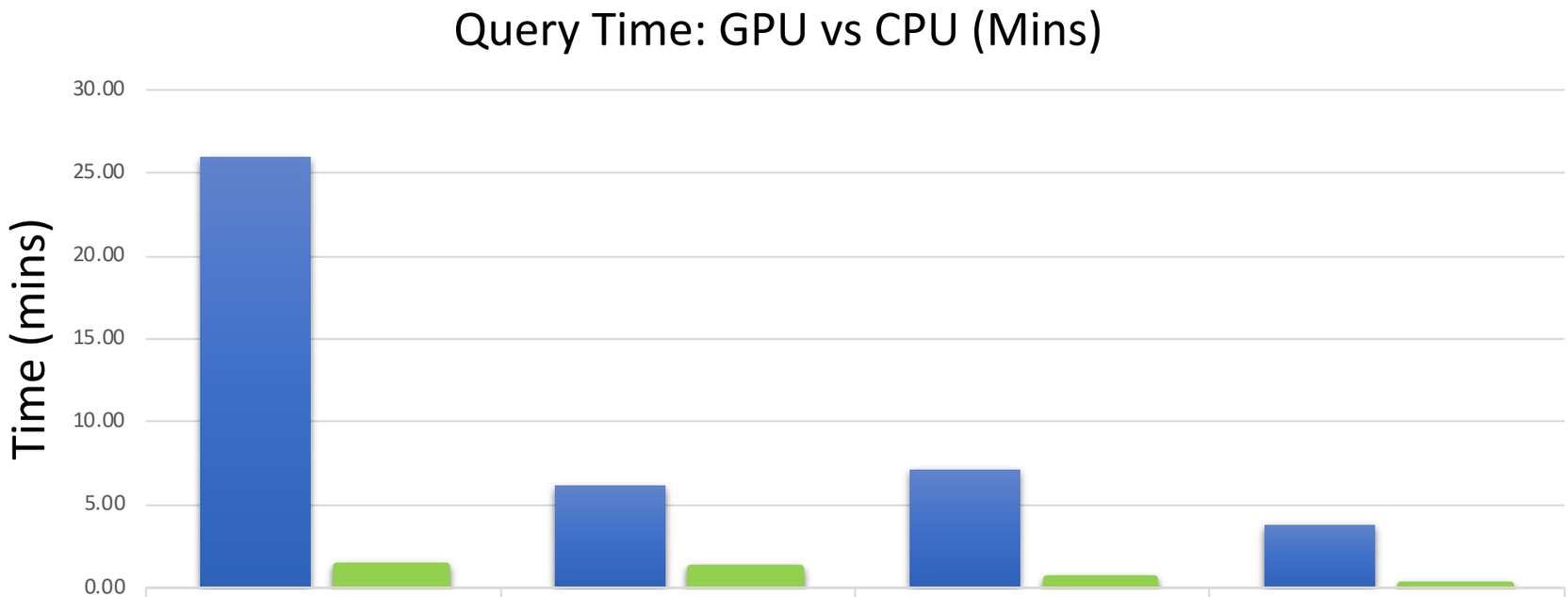
Accelerated SQL/DataFrame

Accelerated ETL?

Can a GPU make an elephant fast?

Yes

TPCx-BB Like Benchmark Results (10TB Dataset, Two Node DGX-2 Cluster)*



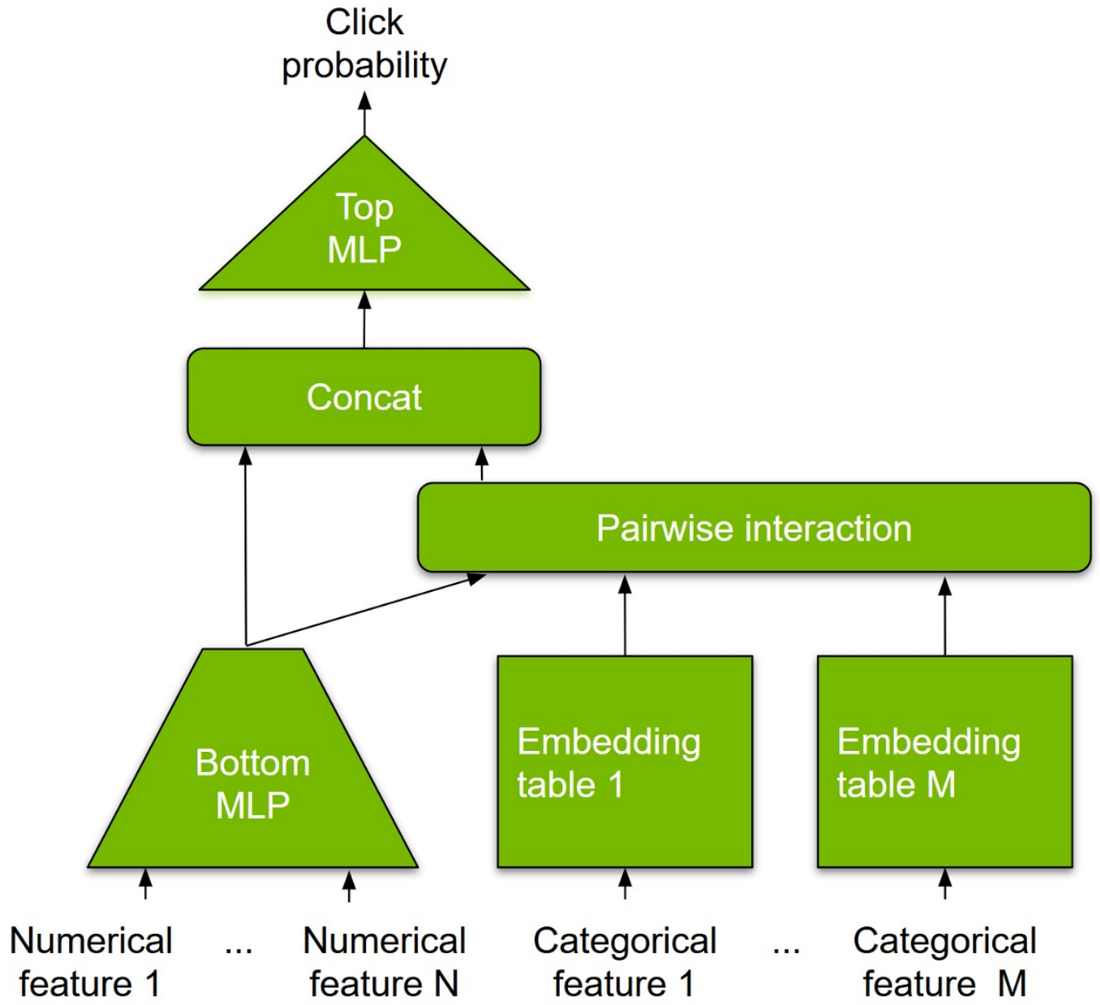
	Query #5	Query #16	Query #21	Query #22
CPU	25.95	6.16	7.13	3.80
GPU	1.31	1.16	0.56	0.14

Environment: Two DGX-2 (96 CPU Cores, 1.5TB Host memory, 16 V100 GPUs, 512 GB GPU Memory)
* Not official or complete TPCx-BB runs (ETL power only).

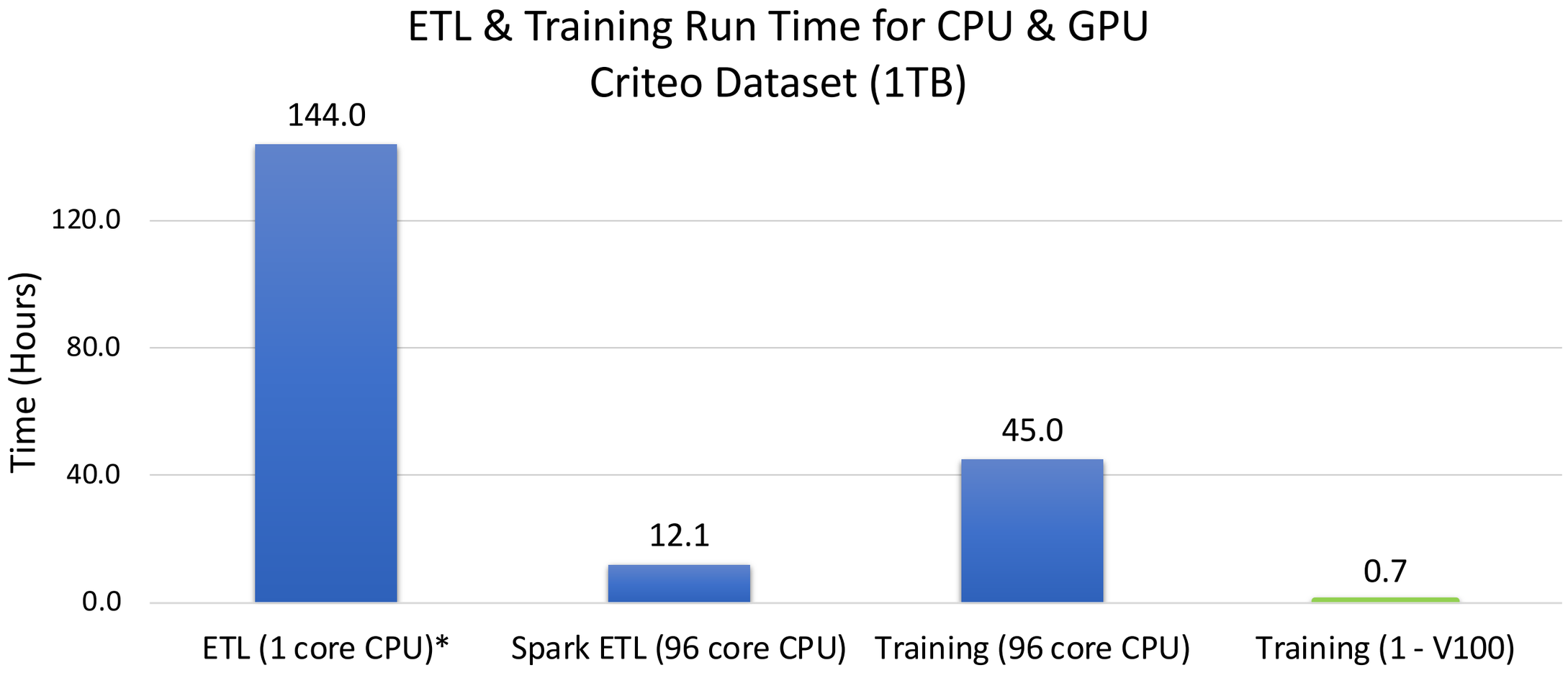
Deep Learning Recommendation Machines

Example use case: Criteo Dataset

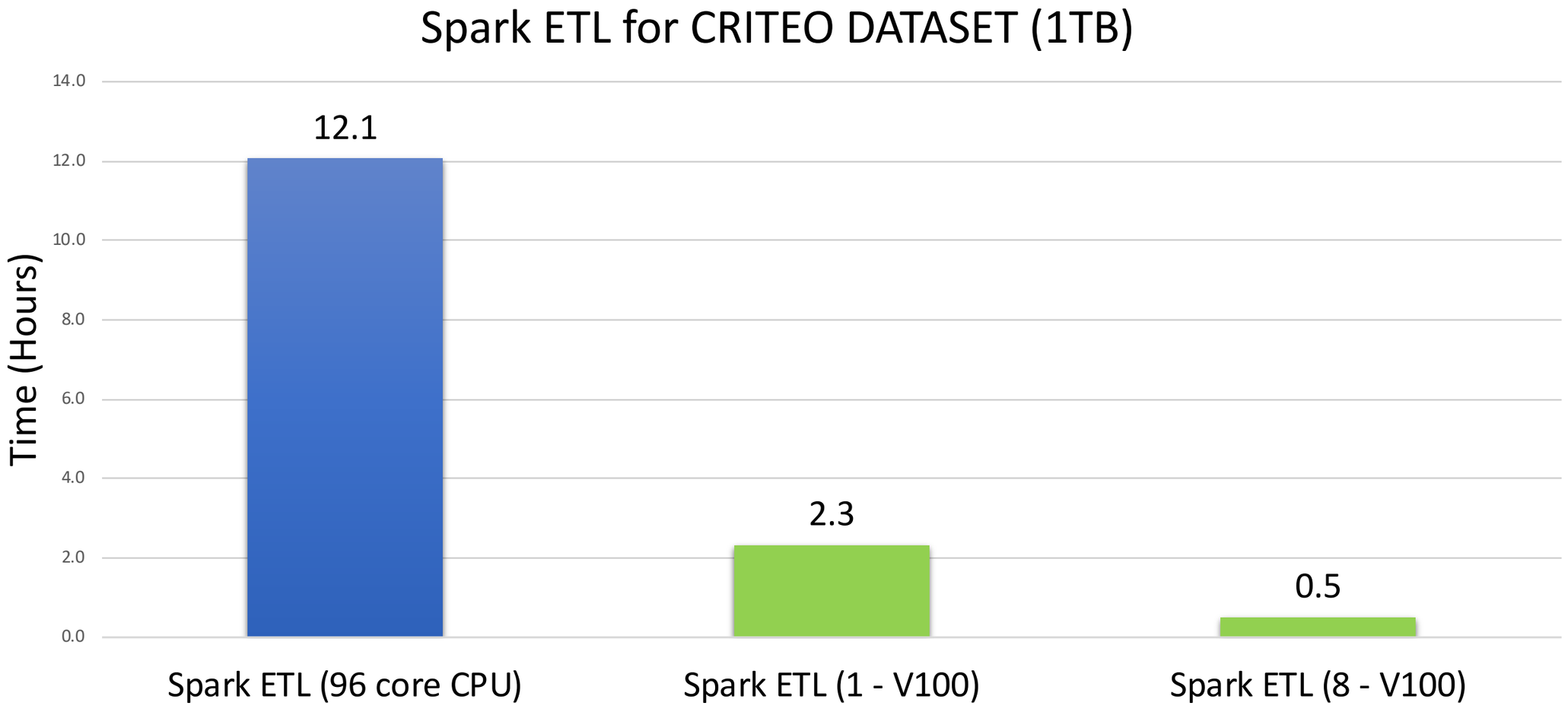
- Anonymized 7-day clickstream (1 TB)
- Convert high-cardinality strings to contiguous integer IDs
- DLRM github repo has turnkey scripts



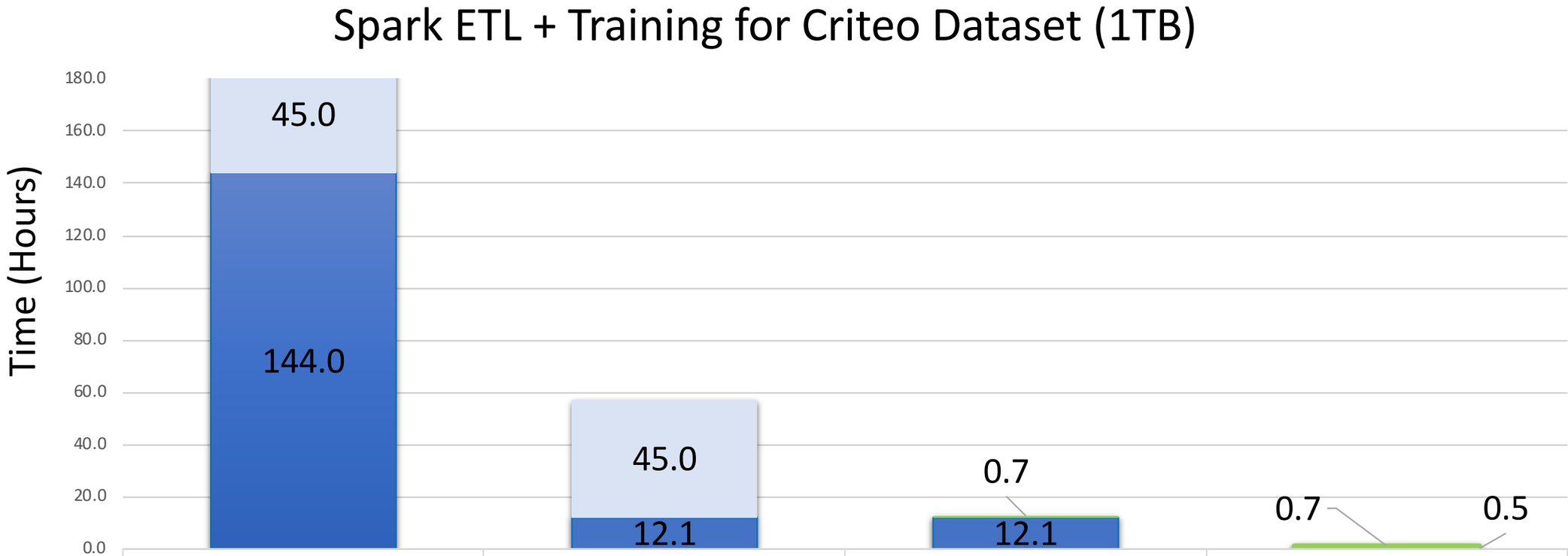
DLRM on Criteo Dataset (Past)



DLRM ETL on Criteo Dataset (Present)



DLRM End-to-End on Criteo Dataset (Present)

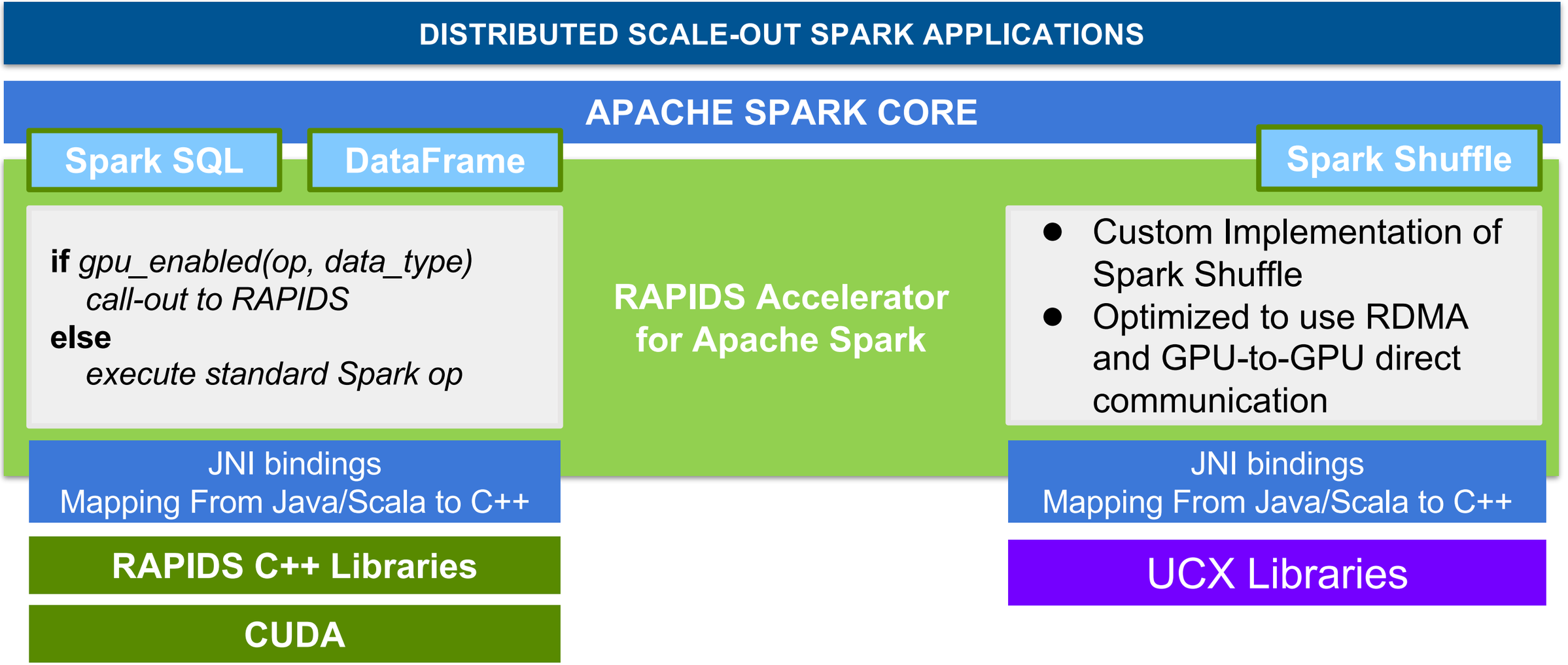


	Original CPU (1 Core for ETL, 96 Core CPU for Training)	Spark CPU (96 Core for ETL & Training)	Spark CPU (96 Core for ETL) & Spark GPU (1-V100 Training)	Spark GPU (8-V100 for ETL & 1-V100 Training)
Training	45.0	45.0	0.7	0.7
ETL	144.0	12.1	12.1	0.5

"The more you buy, the more you save."

Jensen Huang
GPU Technology Conference 2020

RAPIDS Accelerator for Apache Spark (Plugin)



RAPIDS Accelerator for Apache Spark 3.0 Plugin

No Code Changes

Same SQL and DataFrame code

```
spark.conf.set("spark.rapids.sql.enabled", "true")
```

```
start = time.time()
spark.sql("""
select
        o_orderpriority,
        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()
time.time() - start
```

What We Support

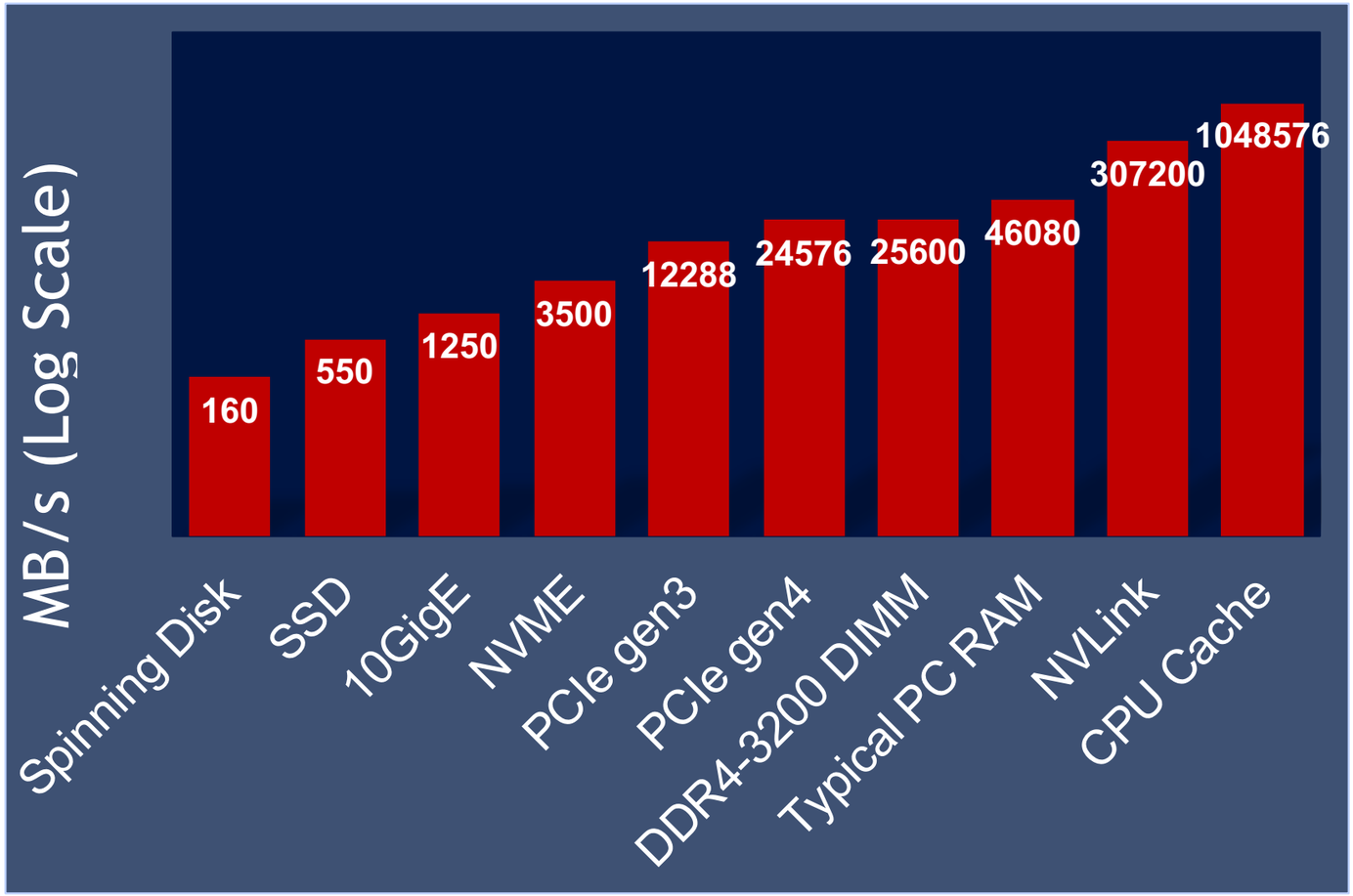
and growing...

!	and	current_date	hour	like	negative	rollup	sum	Parquet Writing
%	asin	current_timestamp	if	ln	not	row_number	tan	ANSI casts
&	atan	date	ifnull	locate	now	rtrim	tanh	TimeSub for time ranges
*	avg	datediff	in	log	nullif	second	timestamp	startswith
+	bigint	day	initcap	log10	nvl	shiftleft	tinyint	endswith
-	boolean	dayofmonth	input_file_block_length	log1p	nvl2	shiftright	trim	contains
/	cast	degrees	input_file_block_start	log2	or	shiftrightunsigned	ucase	limit
<	cbrt	double	input_file_name	lower	pi	sign	upper	order by
<=	ceil	e	int	ltrim	posexplode*	signum	when	group by
<=>	ceiling	exp	isnan	max	position	sin	window	filter
=	coalesce	explode*	isnotnull	mean	pow	sinh	year	union
==	concat	expm1	isnull	min	power	smallint	|	repartition
>	cos	first	last	minute	radians	spark_partition_id	~	equi-joins
>=	cosh	first_value	last_value	mod	rand*	sqrt	CSV Reading*	select
^	cot	float	length	monotonically_increasing_id	regexp_replace*	string	Orc Reading	
abs	count	floor	lcase	month	replace	substr	Orc Writing	
acos	cube	from_unixtime		nanvl	rint	substring	Parquet Reading	

Is This a Silver Bullet?

No

- Small amounts of data
  - Few hundred MB per partition for GPU
- Cache coherent processing
- Data Movement
  - Slow I/O (networking, disks, etc.)
  - Going back and forth to the CPU (UDFs)
  - Shuffle
- Limited GPU Memory



But It Can Be Amazing

What the SQL plugin excels at

- High cardinality data
  - Joins
  - Aggregates
  - Sort
- Window operations
  - Especially on large windows
- Aggregate with lots of distinct operations
- Complicated processing
- Transcoding
  - Encoding and compressing Parquet and ORC is expensive
  - Parsing CSV is expensive

How Does It Work

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"))
```

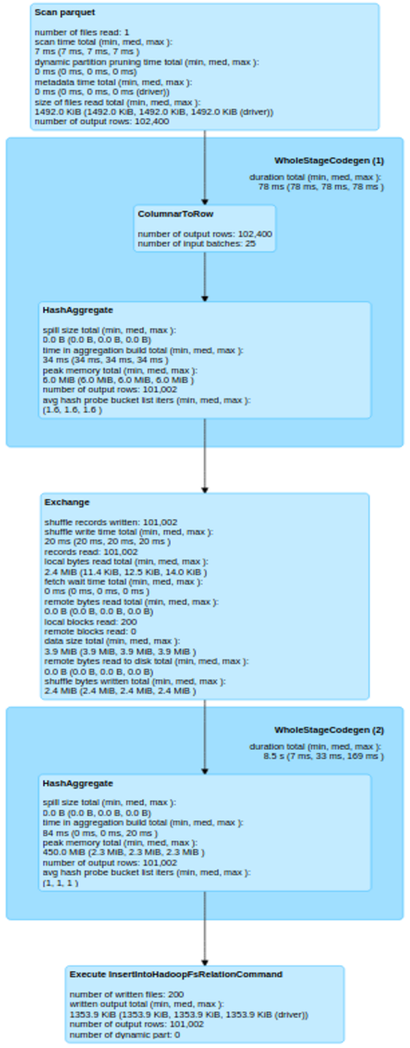
DataFrame

Logical Plan

Physical Plan

RDD[InternalRow]

CPU PHYSICAL PLAN



Spark SQL & DataFrame Compilation Flow

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

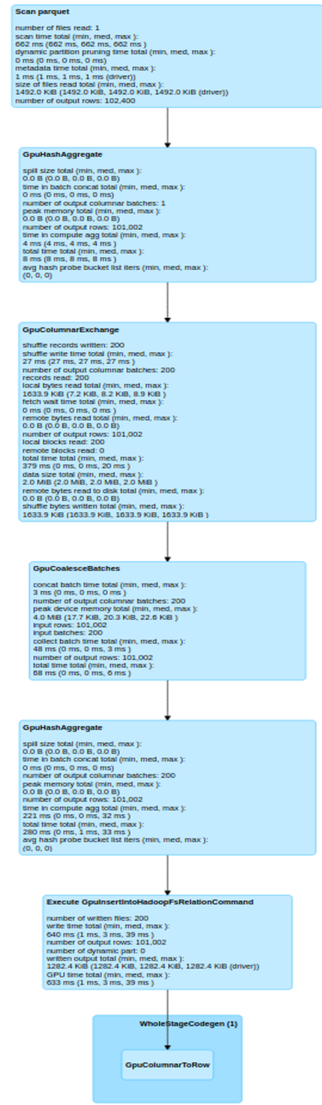
Physical Plan

Physical Plan

RDD[ColumnarBatch]

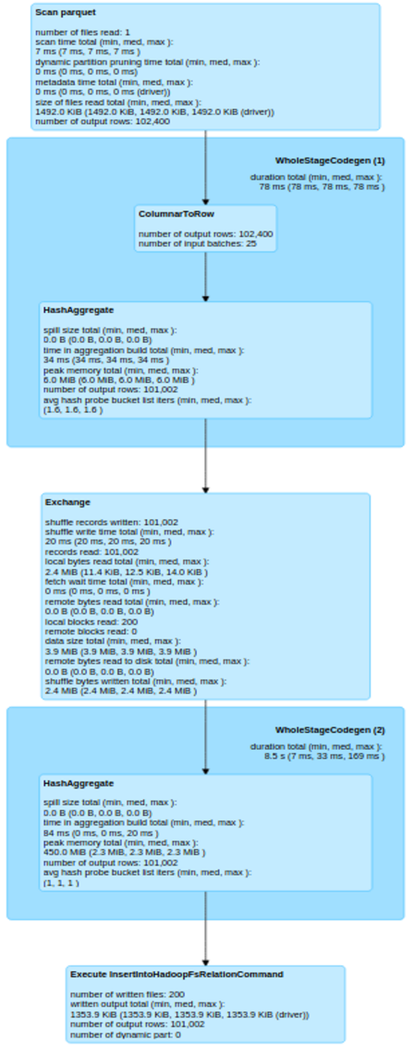
RDD[InternalRow]

GPU PHYSICAL PLAN



Spark SQL & DataFrame Compilation Flow

CPU PHYSICAL PLAN



Read Parquet File

Convert to Row Format

First Stage Aggregate

Shuffle Exchange

Second Stage Aggregate

Write Parquet File

Read Parquet File

First Stage Aggregate

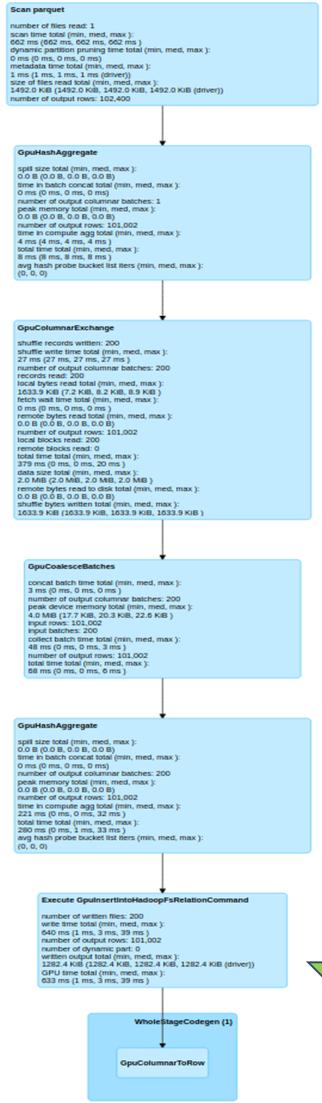
Shuffle Exchange

Combine Shuffle Data

Second Stage Aggregate

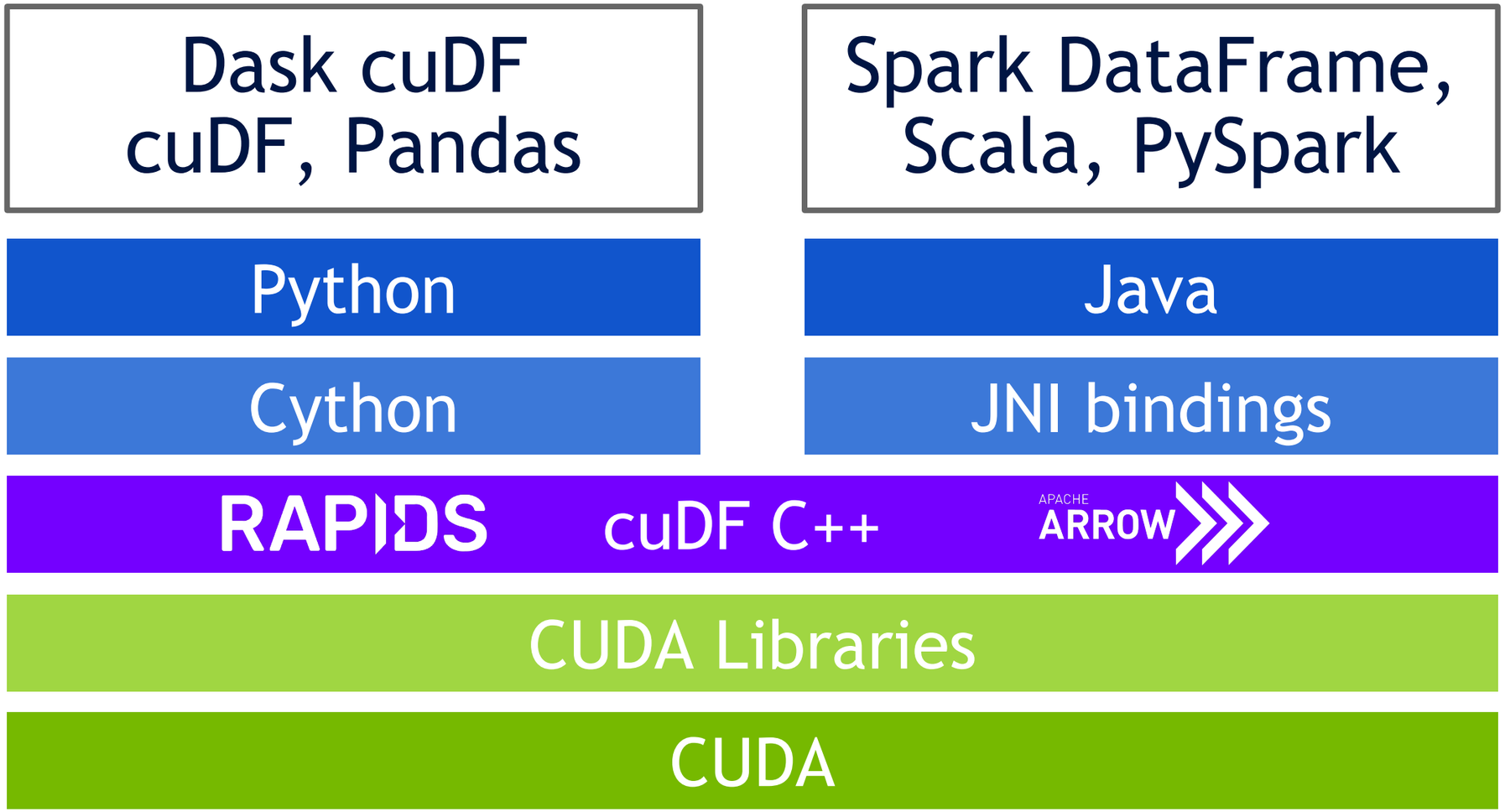
Write Parquet File

Convert to Row Format



GPU PHYSICAL PLAN

ETL Technology Stack



Demo

Demo Cluster Setup

Databricks (AWS)

CPU

Driver:

- 1 - r4.xlarge
  - 30.5GB Memory
  - 4 Cores
  - 1 DBU

Workers:

- 12 - r4.2xlarge
  - 61GB Memory
  - 8 cores
  - 2 DBU

GPU

Driver:

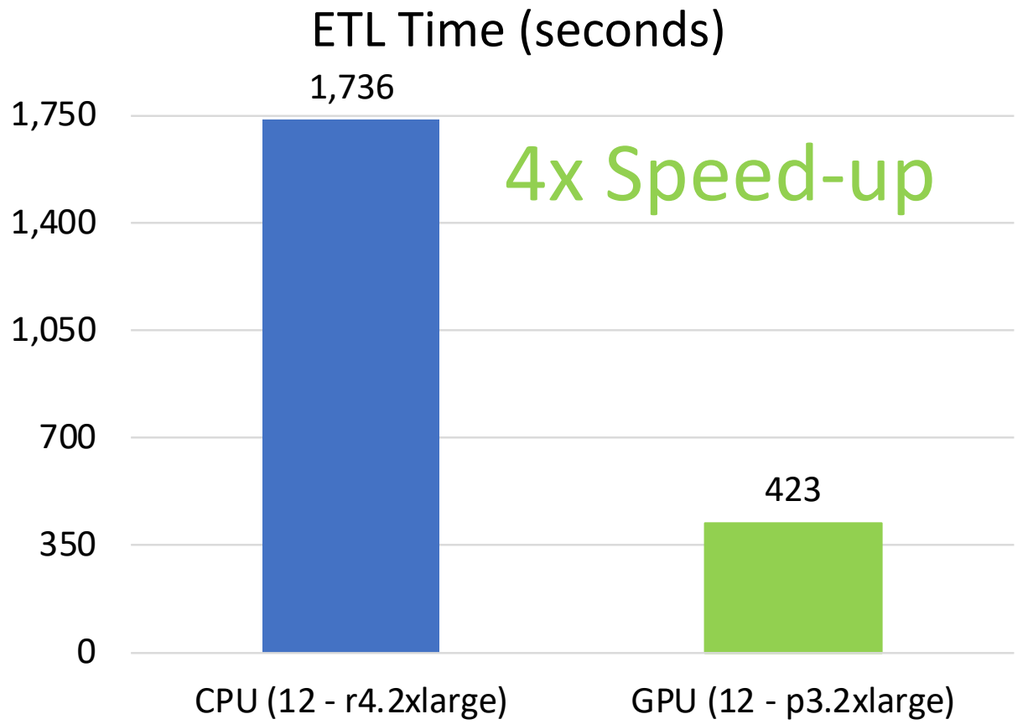
- 1 - p2.xlarge
  - 61GB Memory
  - 4 cores
  - 1 - K80 (Not needed)
  - 1.22 DBU

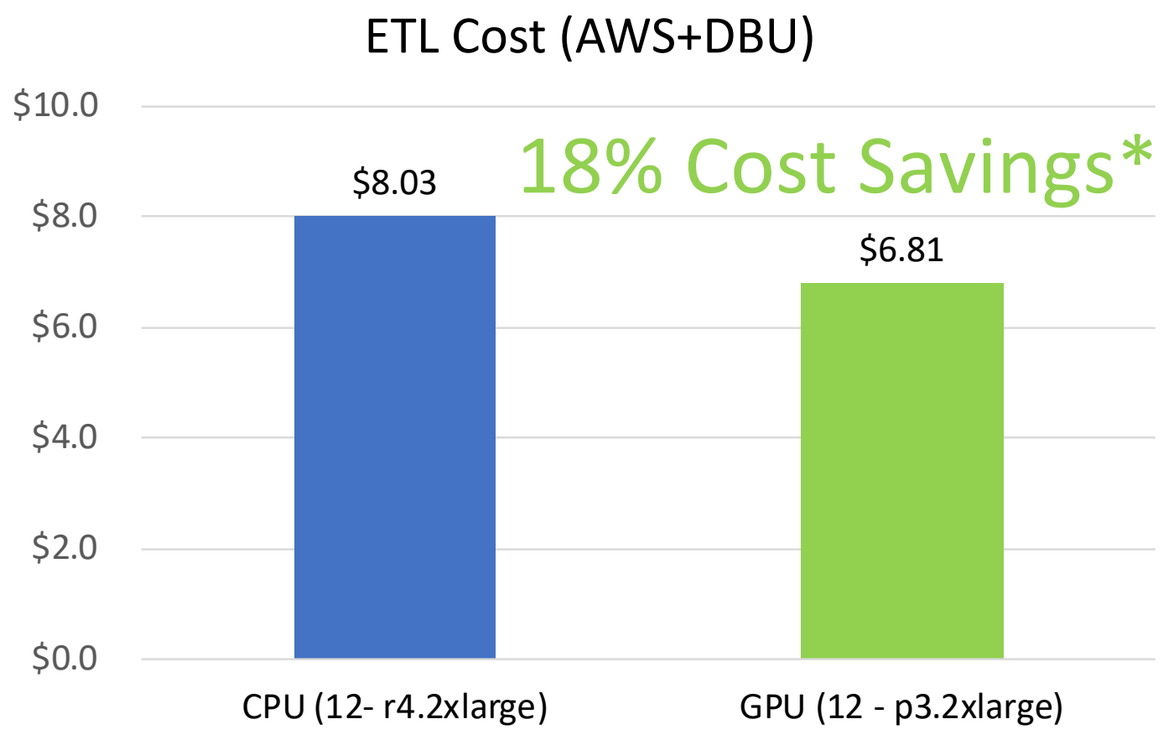
Workers:

- 12 – p3.2xlarge
  - 61GB Memory
  - 1 - V100
  - 8 cores
  - 4.15 DBU

Databricks Demo Results

"The more you buy, the more you save" – Jensen H Huang, CEO NVIDIA





* Costs based on Databricks Standard edition

T4 Cluster Setup

EC2

V100 is optimized for ML/DL training

T4 fits better with SQL processing

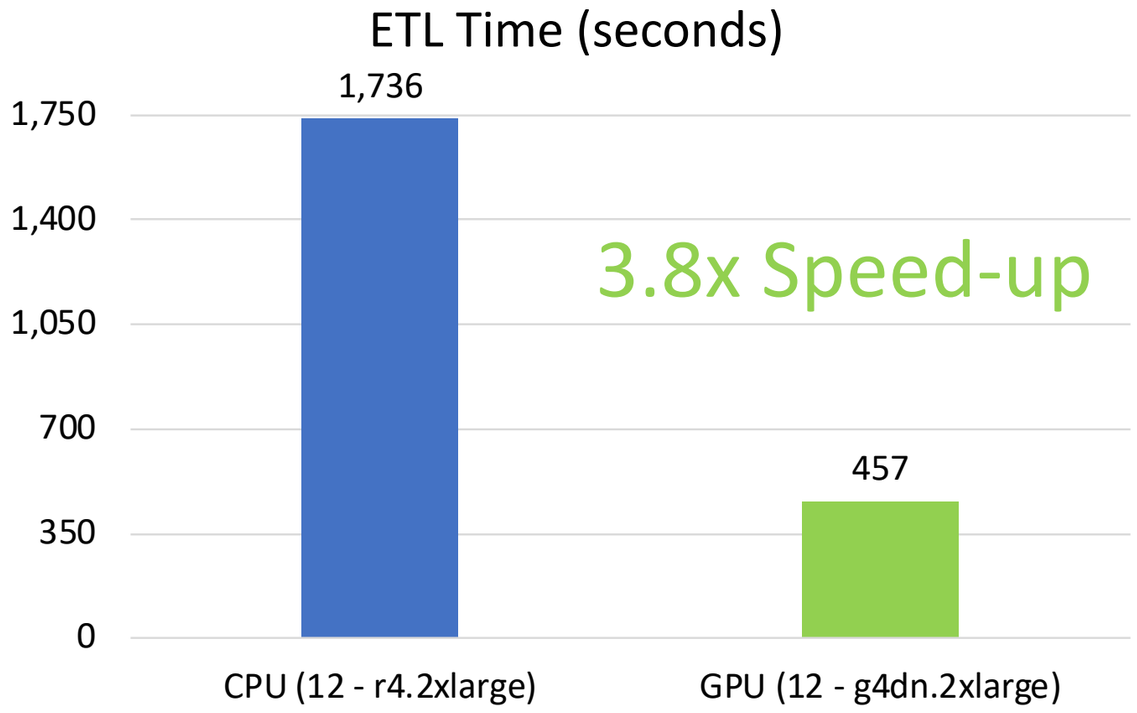
Driver (Ran on one of the worker nodes)

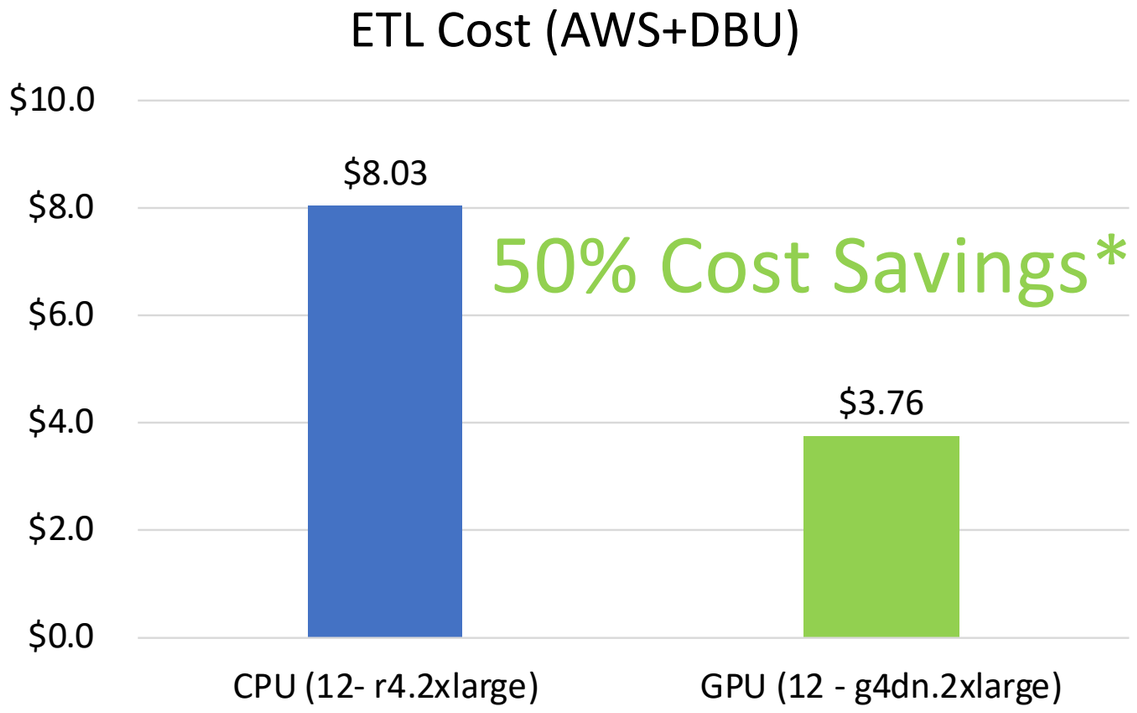
Workers:

- 12 – g4dn.2xlarge
  - 32GB Memory
  - 1 - T4
  - 8 cores

Coming Soon....T4 GPUs on Databricks

Same speed-up as V100 but more savings



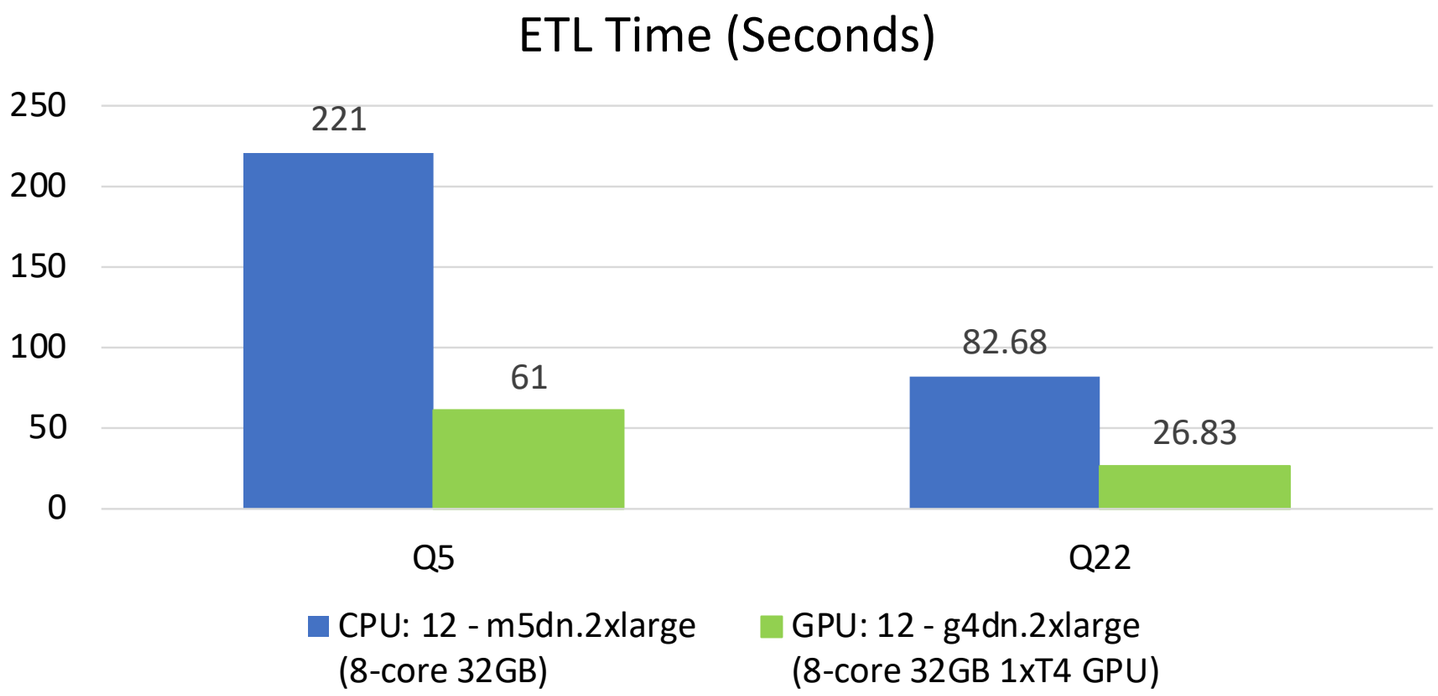


* Costs based on AWS T4 GPU instance market price & V100 GPU price on Databricks Standard edition

RAPIDS Accelerator on AWS

Based on TPCx-BB like Queries #5 & #22 with 1TB scale factor input

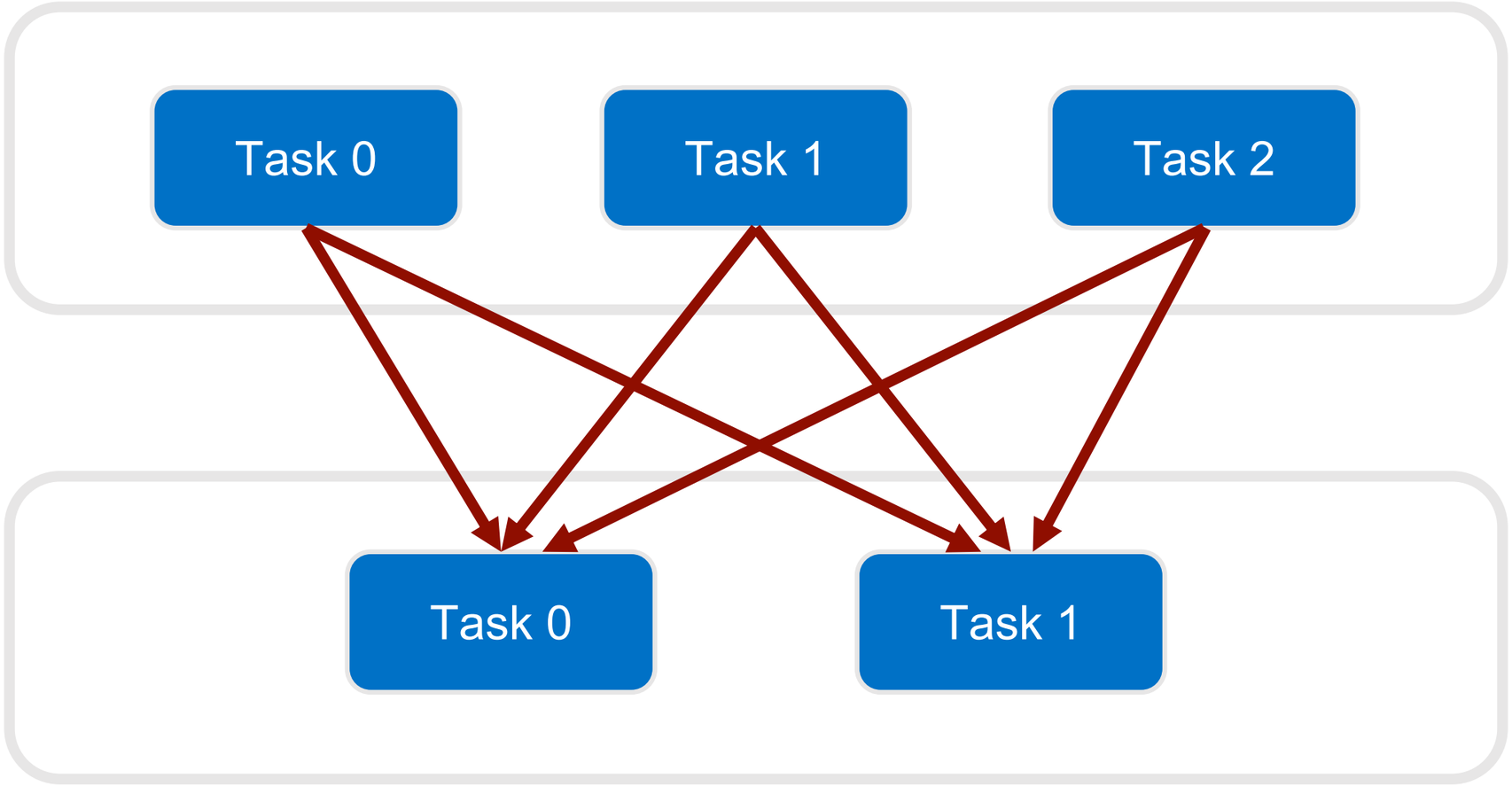
- ~3.5x Speed-up
- ~40% Cost Savings



Accelerated Shuffle

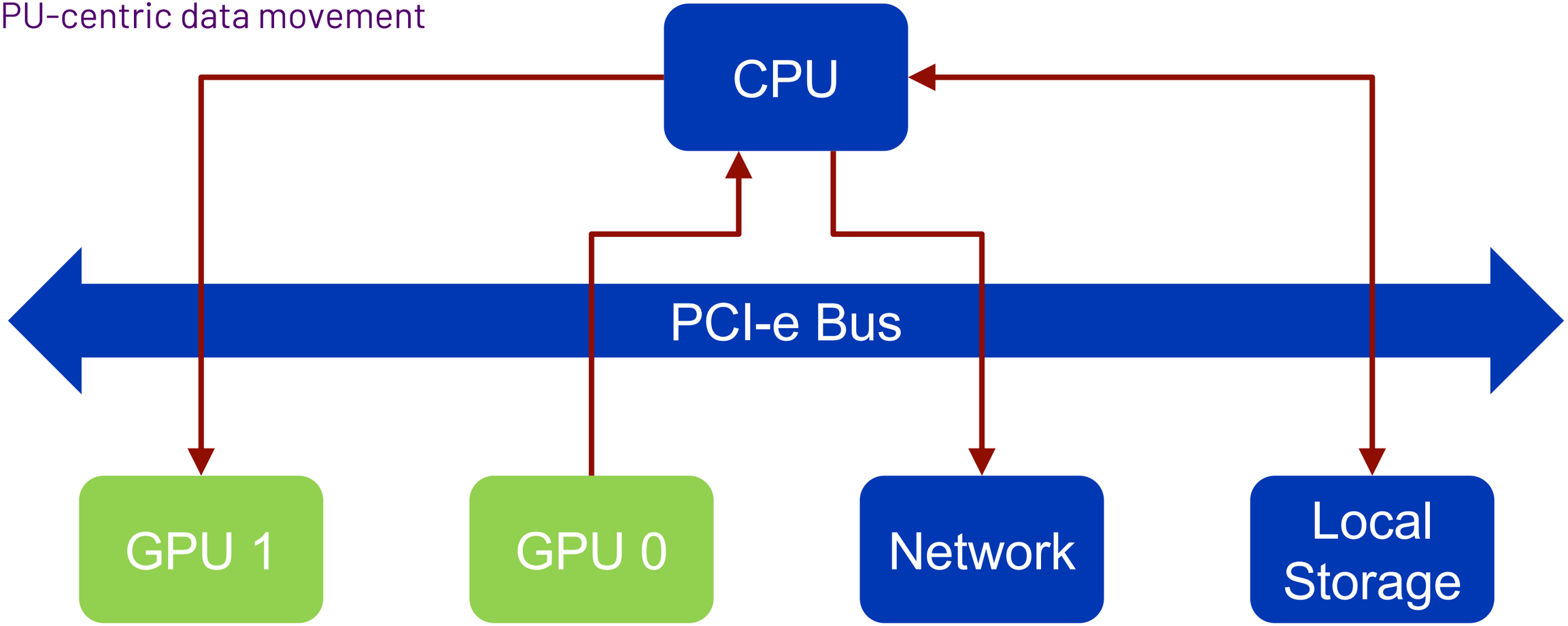
Spark Shuffle

Data exchange between stages



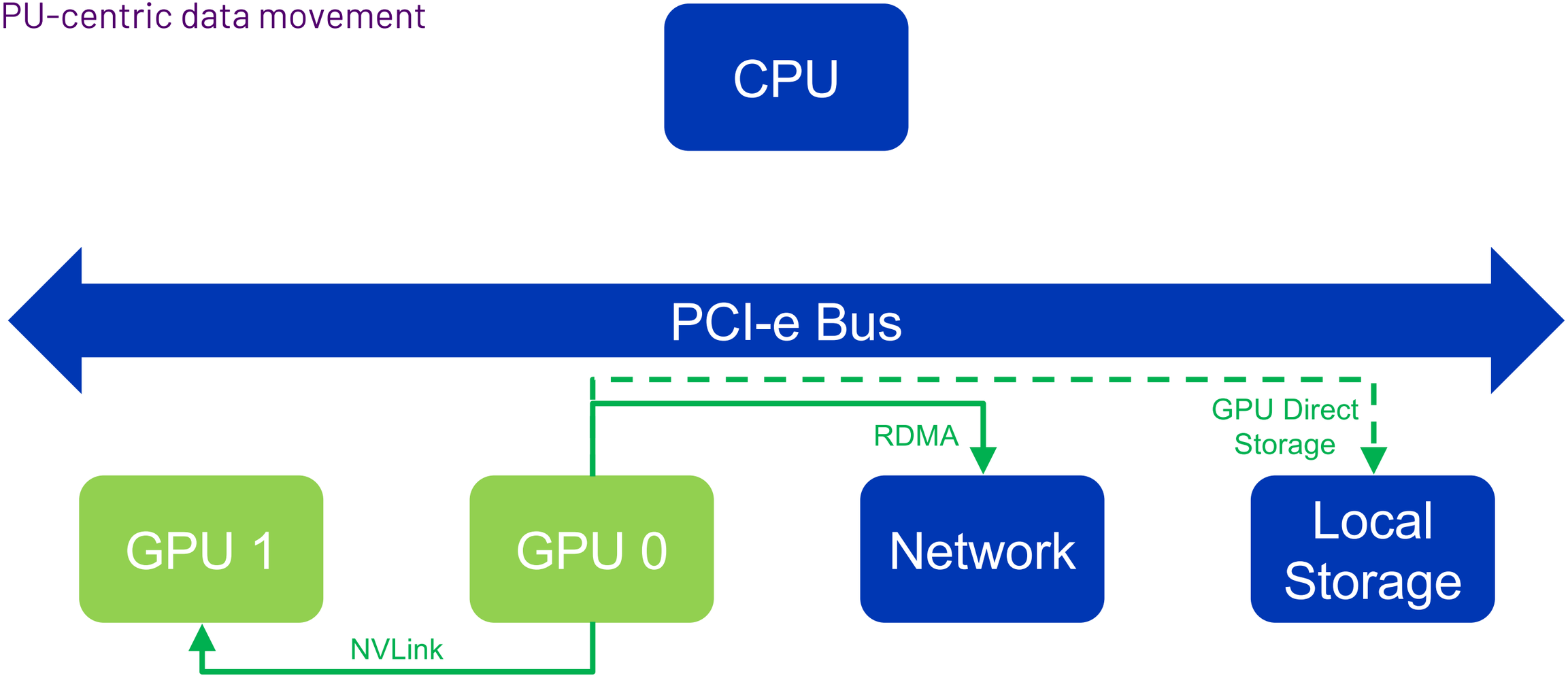
Spark Shuffle

CPU-centric data movement



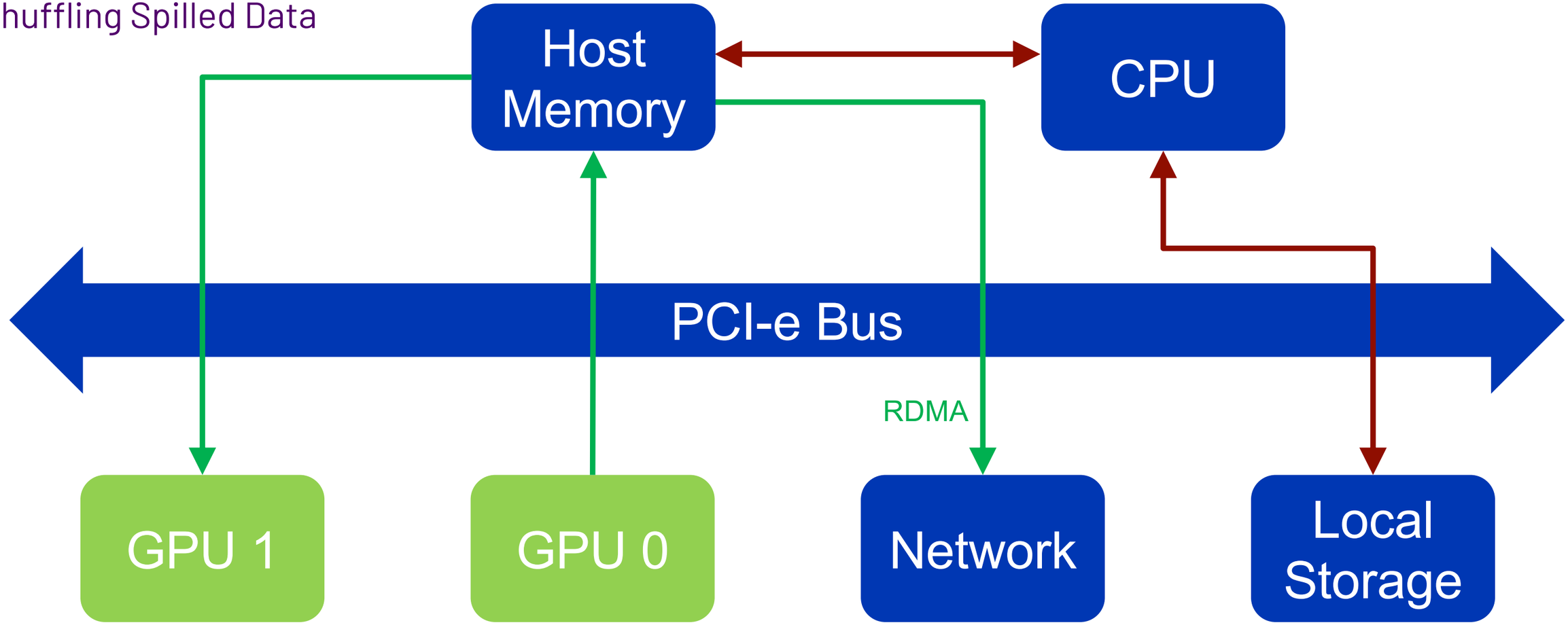
Accelerated Shuffle

GPU-centric data movement



Accelerated Shuffle

Shuffling Spilled Data



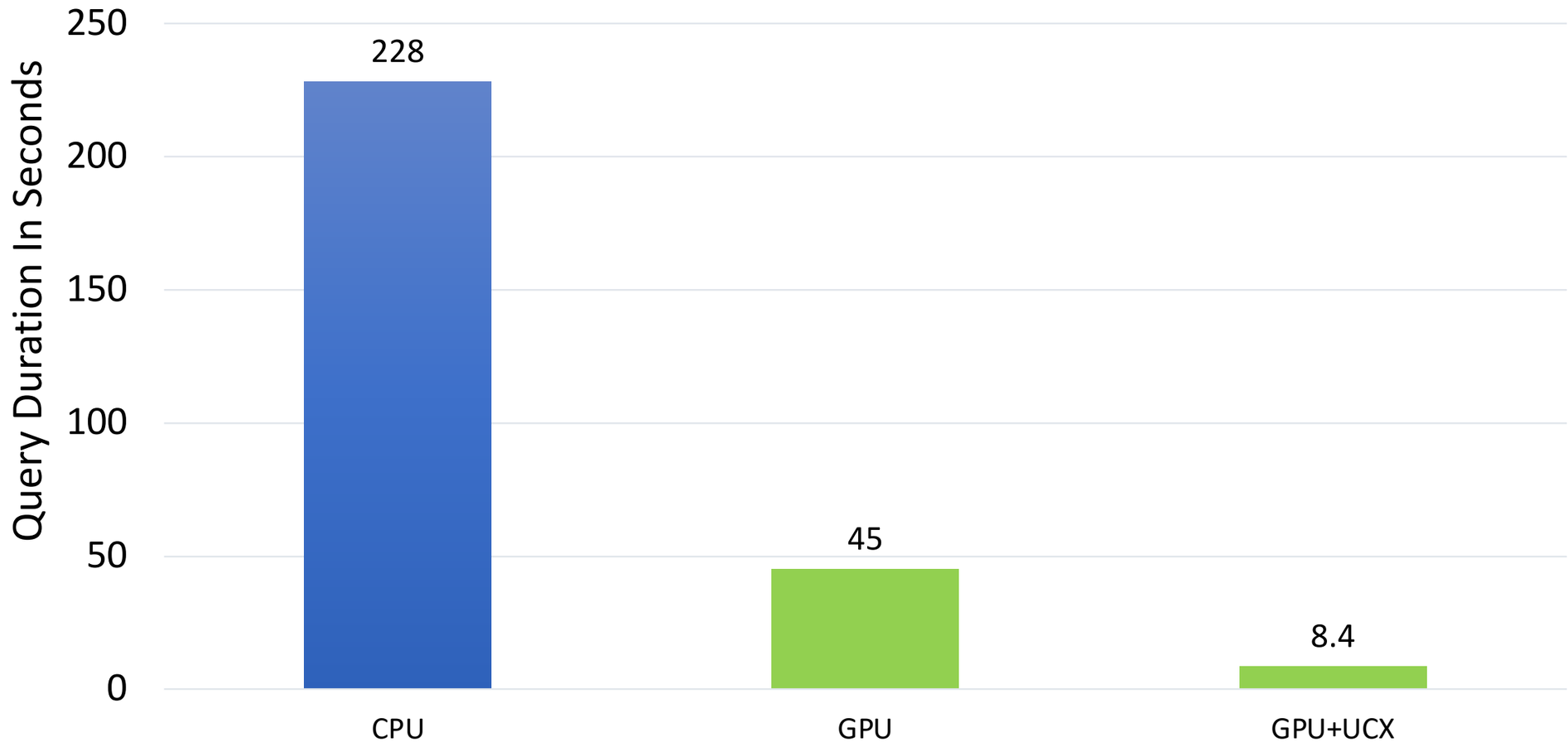
UCX Library

Unified Communication X

- Abstracts communication transports
- Selects best route(s)
  - TCP
  - RDMA
  - Shared Memory
  - CUDA IPC
- Zero-copy GPU transfers over RDMA
- RDMA requires network support
  - Infiniband
  - RoCE
- http://openucx.org

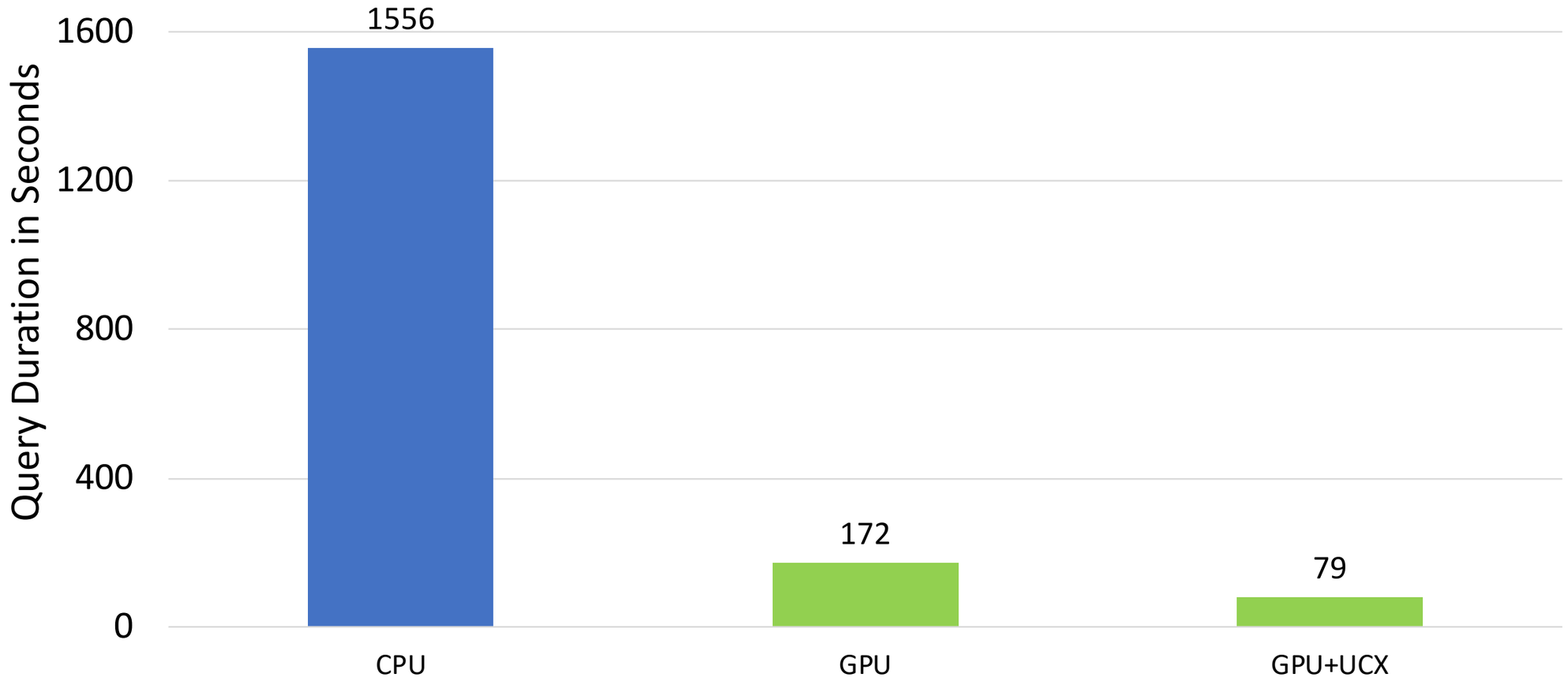
Accelerated Shuffle Results

Inventory pricing query



Accelerated Shuffle Results

ETL for logistical regression model



What’s Next?

What's Next

Coming Soon

- Open Source (DONE)
  - https://github.com/NVIDIA/spark-rapids
  - https://nvidia.github.io/spark-rapids/
- Nested types
  - Arrays
  - Structs
  - Maps
- Decimal type
- More operators

Further Out

- GPU Direct Storage
- Time zone support for timestamps
  - Only UTC supported now
- Higher order functions
- UDFs

Where to Get More Information

- https://NVIDIA.com/Spark
  - Please use the “Contact Us” link to get in touch with NVIDIA’s Spark team
- https://github.com/NVIDIA/spark-rapids
- https://nvidia.github.io/spark-rapids/

- Listen to Adobe’s Email Marketing Intelligent Services Use-Case

databricks

- Free e-book at NVIDIA.com/Spark-Book

Feedback

Your feedback is important to us.

Don’t forget to rate and review the sessions.

SPARK+AI SUMMIT 2020

Organized by databricks