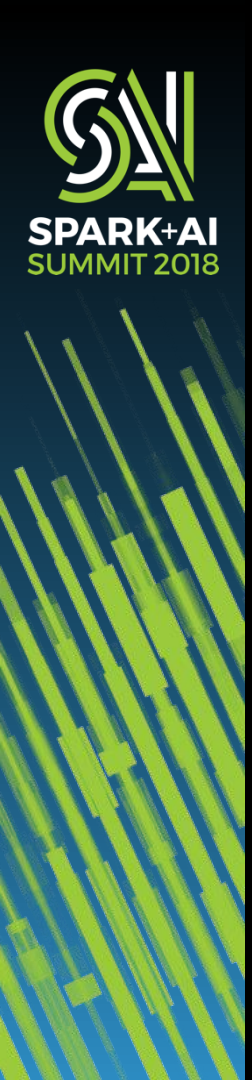


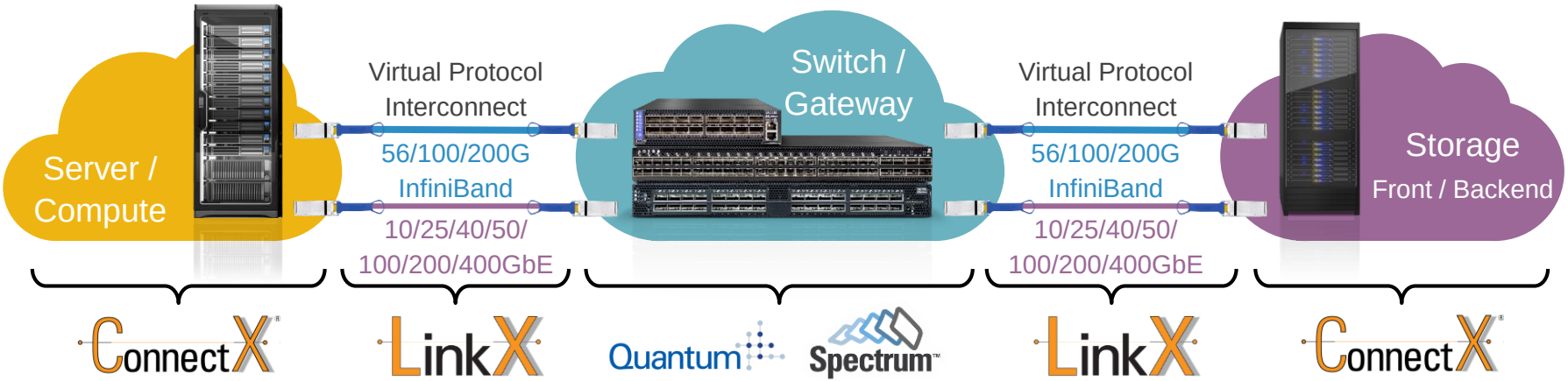

# Accelerated Spark on Azure: Seamless and Scalable Hardware Offloads in the Cloud

Yuval Degani, Mellanox Technologies

Evan Burness, Microsoft Azure

**#HWCSAIS18**

- End-to-end designer and supplier of interconnect solutions: network adapters, switches, system-on-a-chip, cables, silicon and software
- 10-400 Gb/s Ethernet and InfiniBand



# Microsoft Azure

- RDMA capable network, powered by Mellanox
- H-series (Intel CPUs with FDR InfiniBand)
- NC-series (Nvidia GPUs with FDR InfiniBand)
- Only major Cloud provider with RDMA
- Run simulation and AI workloads at large-scale
- Dozens of RDMA clusters around the world

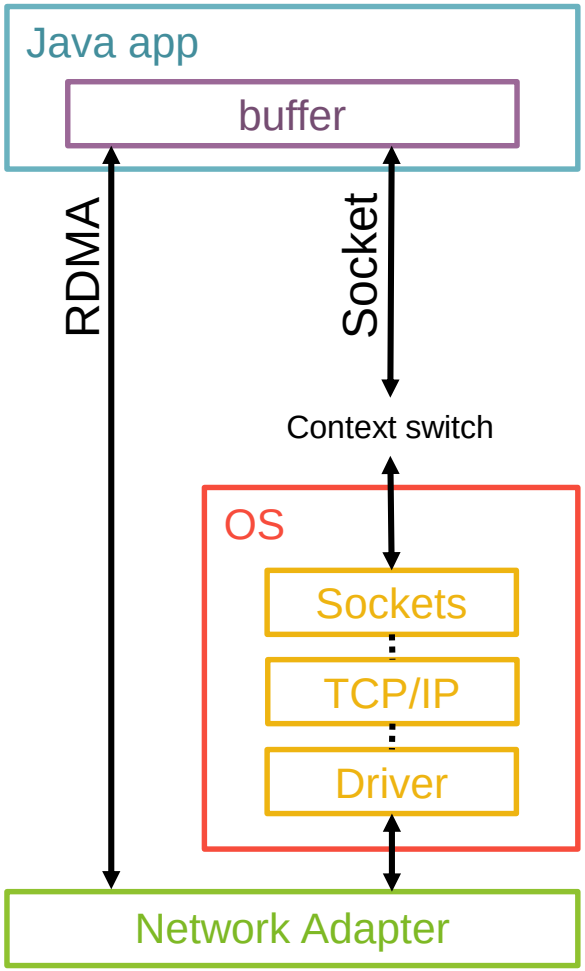
# Why are we here?

- Azure hardware accelerated networks will soon support general-purpose RDMA (on top of SR-IOV)

- SparkRDMA Shuffle Plugin (appeared at Spark Summit Europe 2017) can now be used in the cloud, providing instant speedups for Spark jobs

# What's RDMA?

- Remote Direct Memory Access
  - Read/write from/to remote memory locations
- Zero-copy
- Direct hardware interface – bypasses the kernel and TCP/IP in IO path
- Flow control and reliability is offloaded in hardware
- Sub-microsecond latency
- Supported on almost all mid-range/high-end network adapters

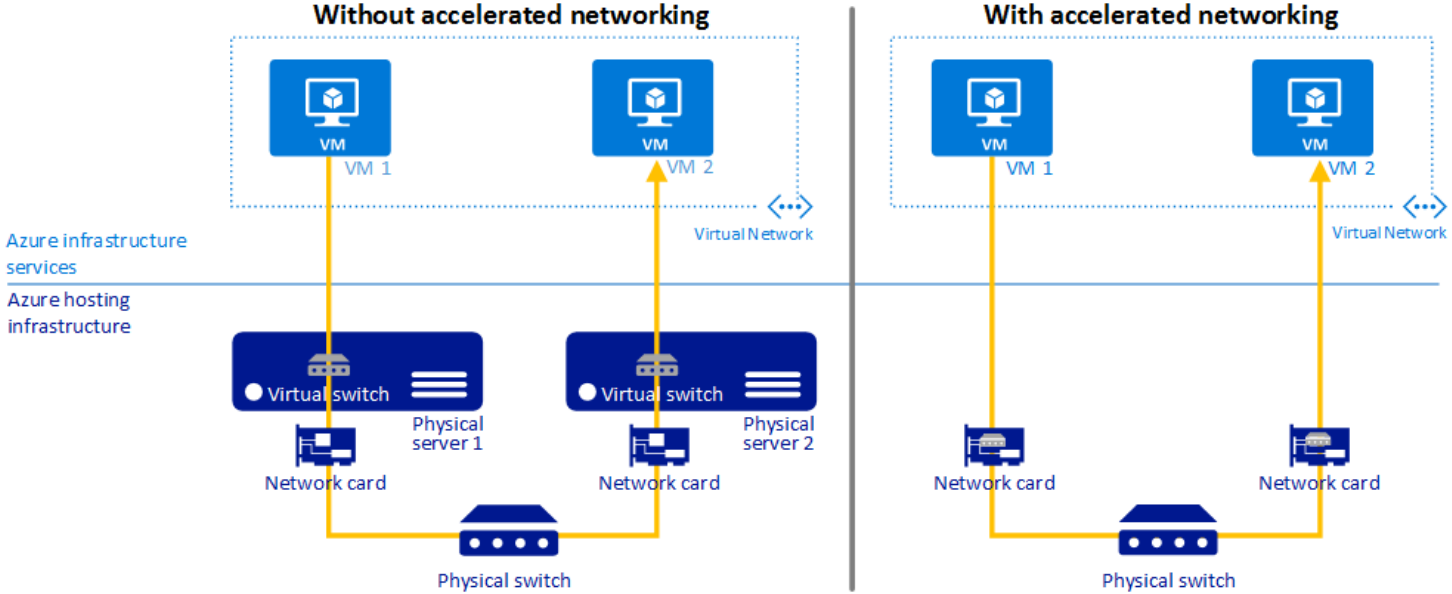


# RDMA on Azure

- No need for buying expensive hardware
- Lowest latency on the Cloud (~2.5 uSec)
- Pre-built OS images for easy deployment
- K80, P100, and V100 GPUs with InfiniBand
- Other uses cases for RDMA on Azure:

# RDMA on Azure

Azure accelerated networking is build on top of SR-IOV (Single Root Input/Output Virtualization) hardware support provided by Mellanox ConnectX network cards



Under the hood

# Spark’s Shuffle Internals

# Spark's Shuffle Basics

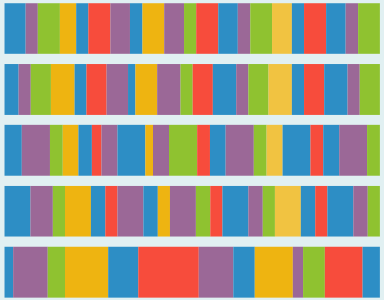
Map

Reduce

# Spark's Shuffle Basics

Map

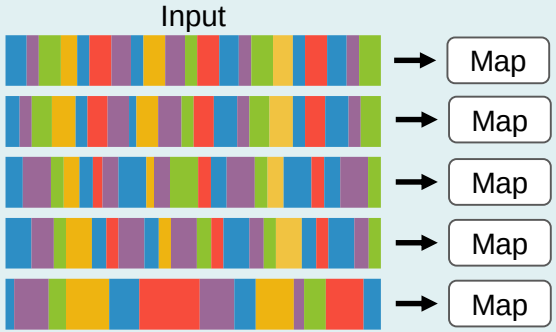
Input



Reduce

# Spark's Shuffle Basics

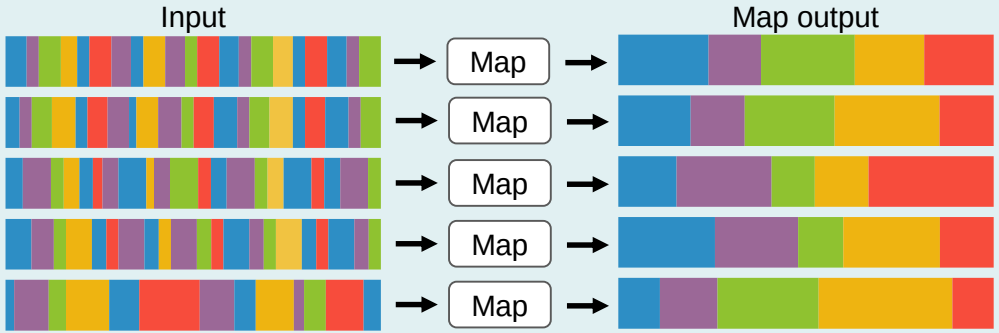
Map




Reduce

# Spark's Shuffle Basics

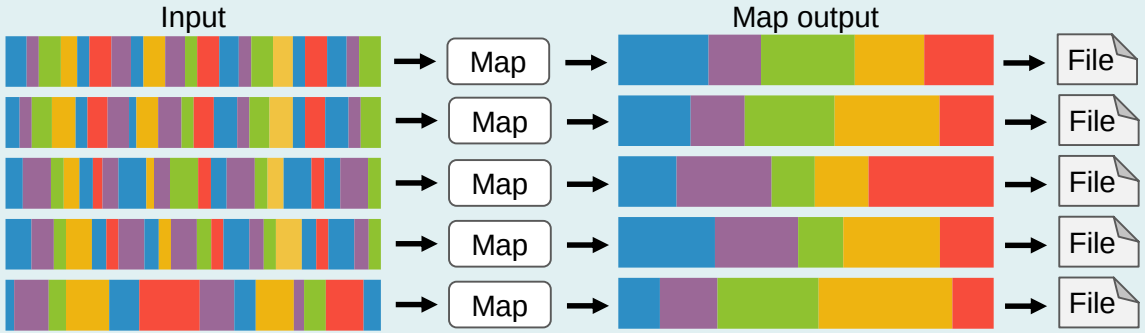
Map




Reduce

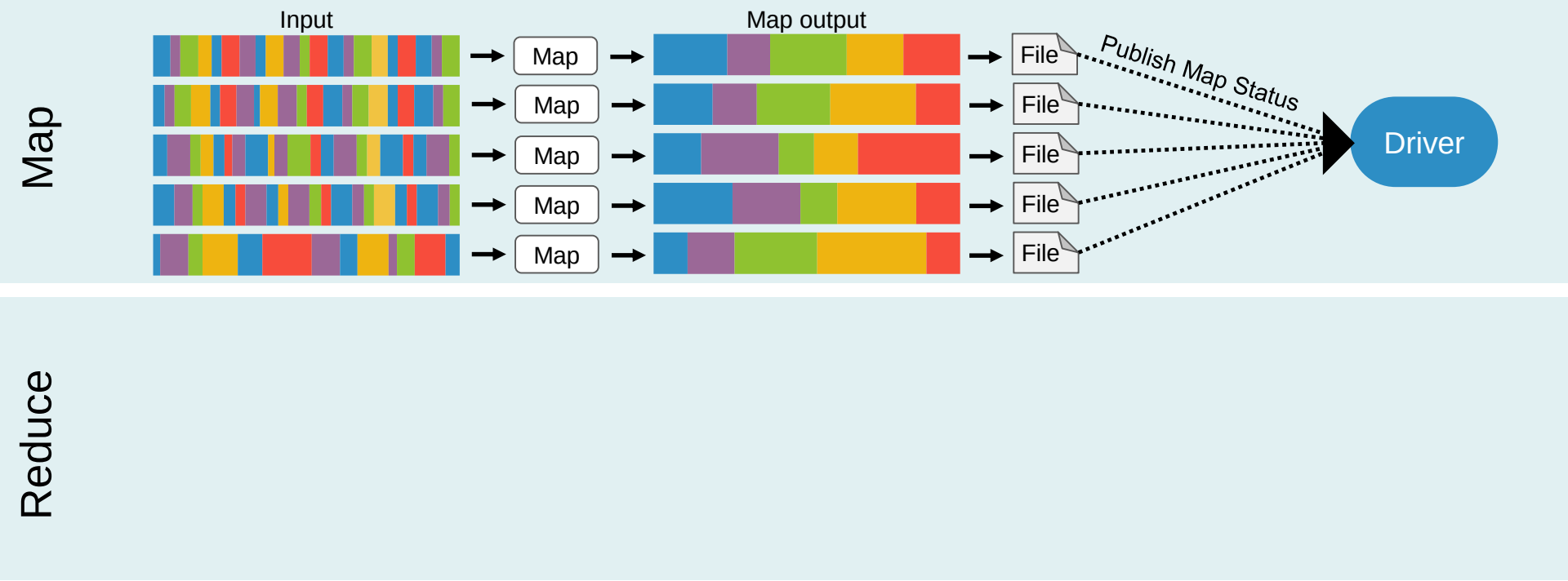
# Spark's Shuffle Basics

Map

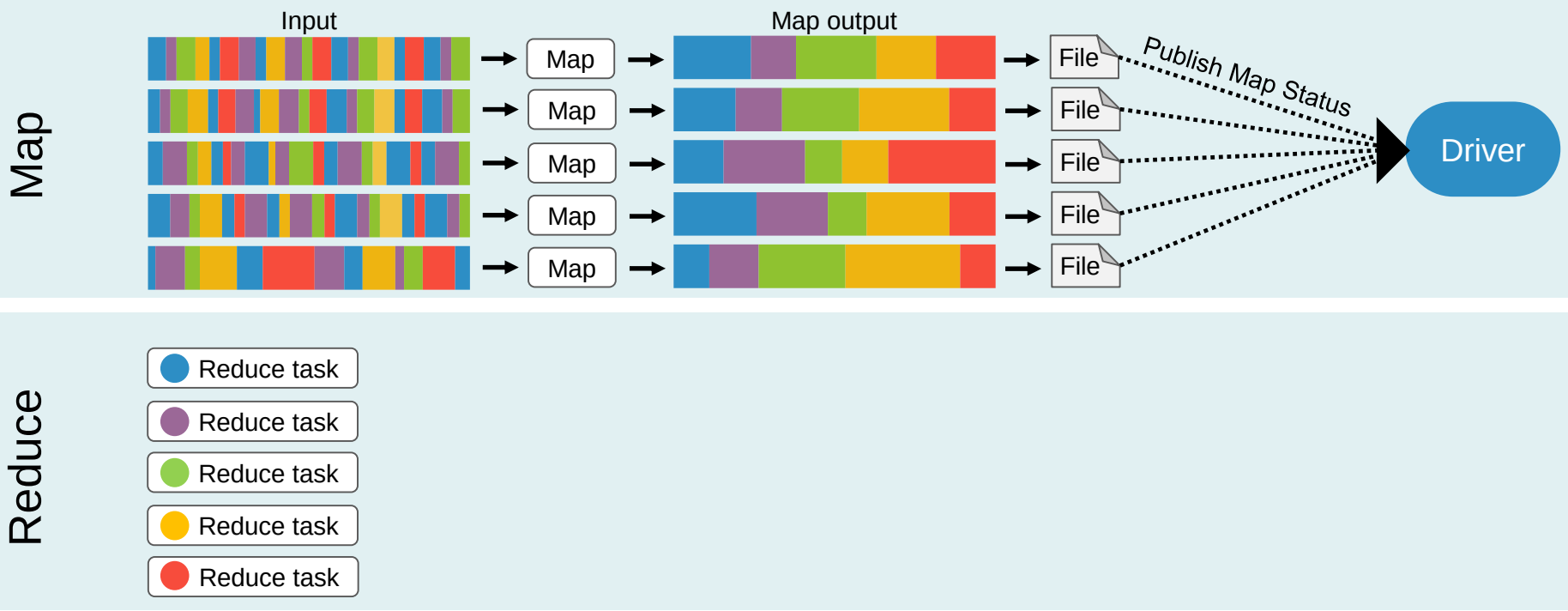



Reduce

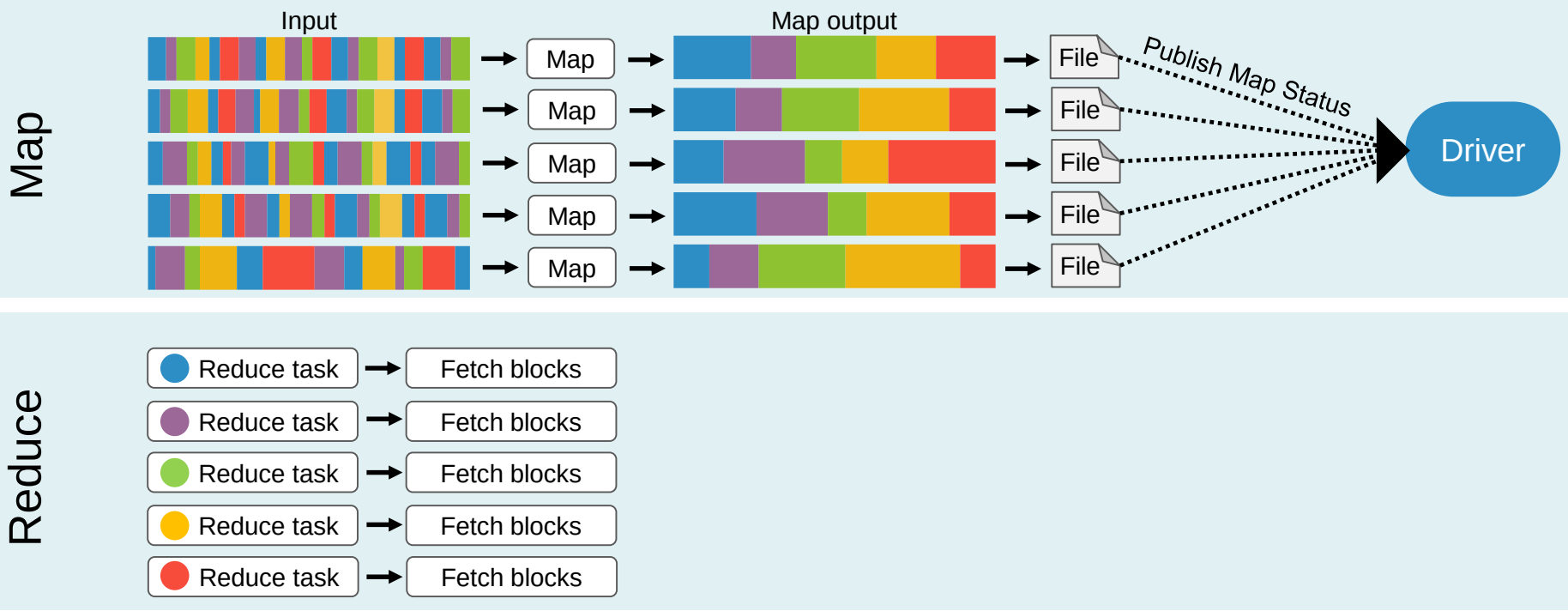
# Spark's Shuffle Basics



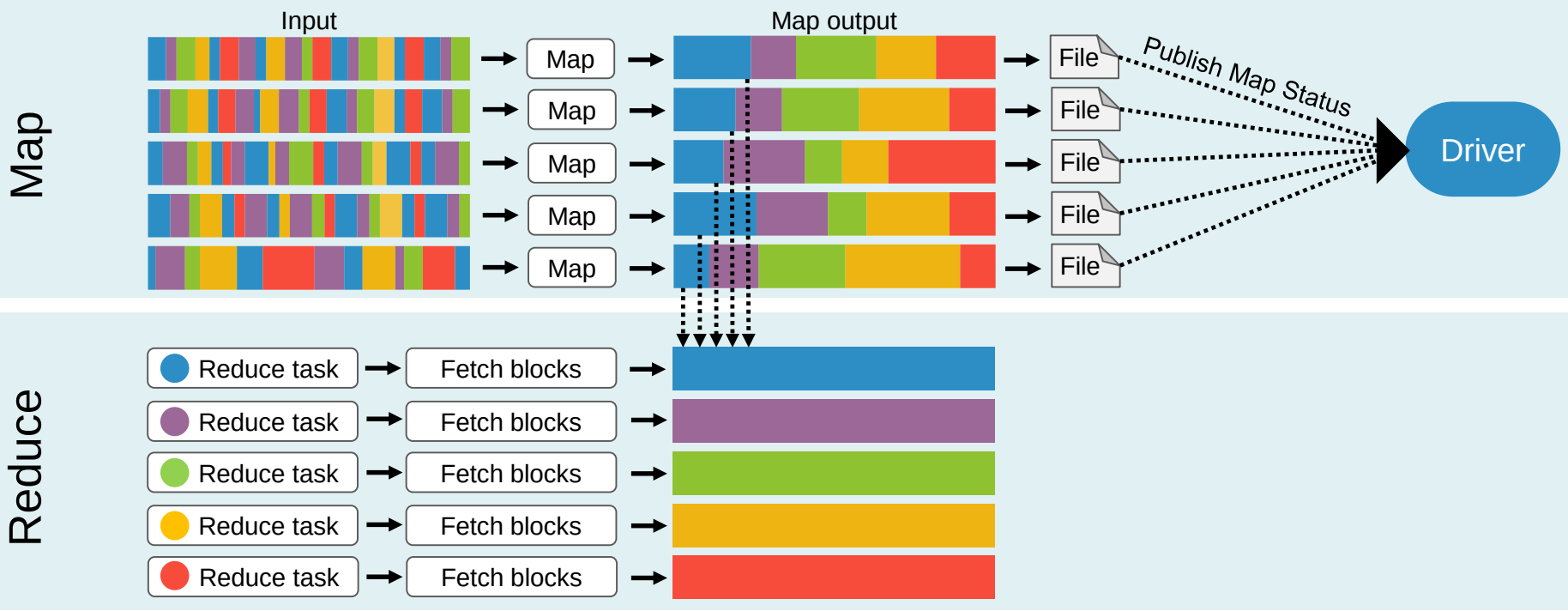
# Spark's Shuffle Basics



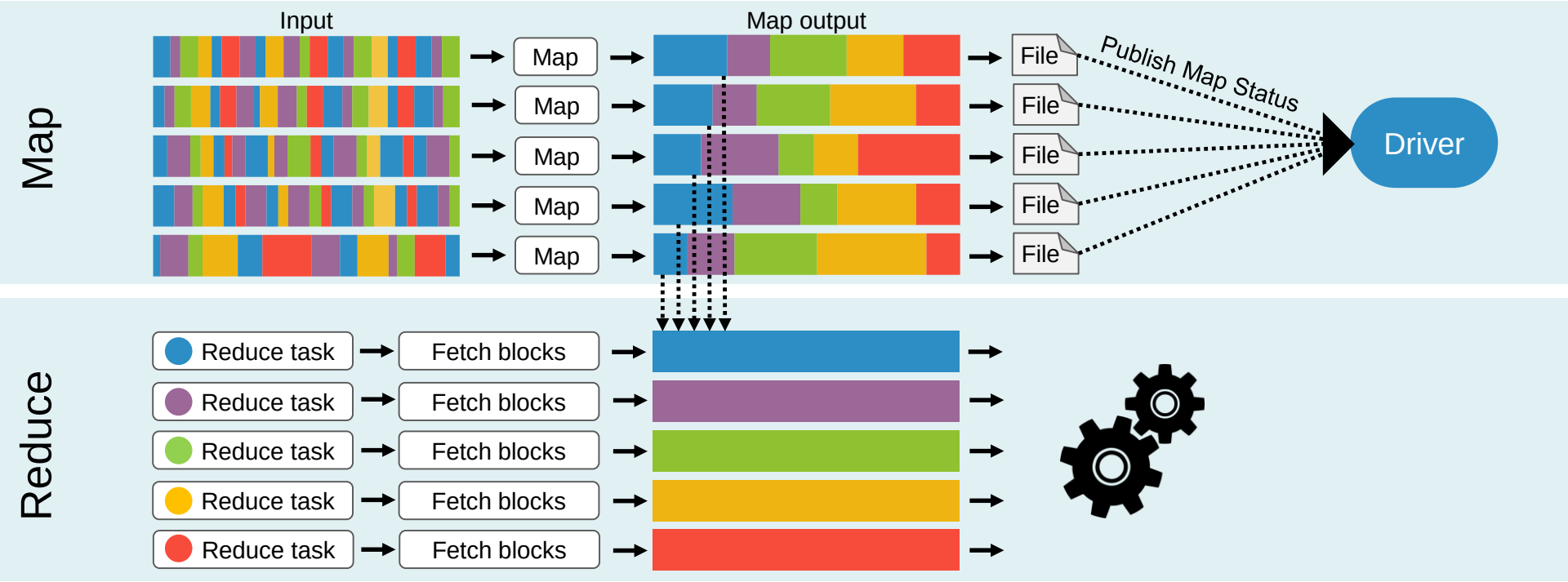
# Spark's Shuffle Basics



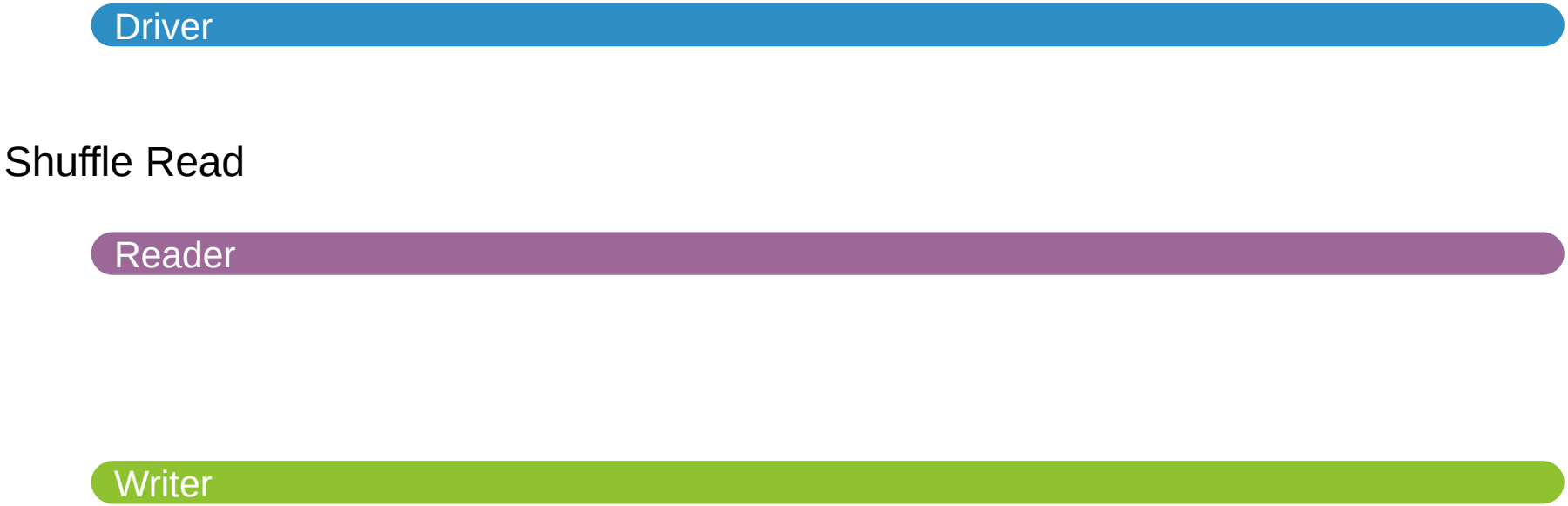
# Spark's Shuffle Basics



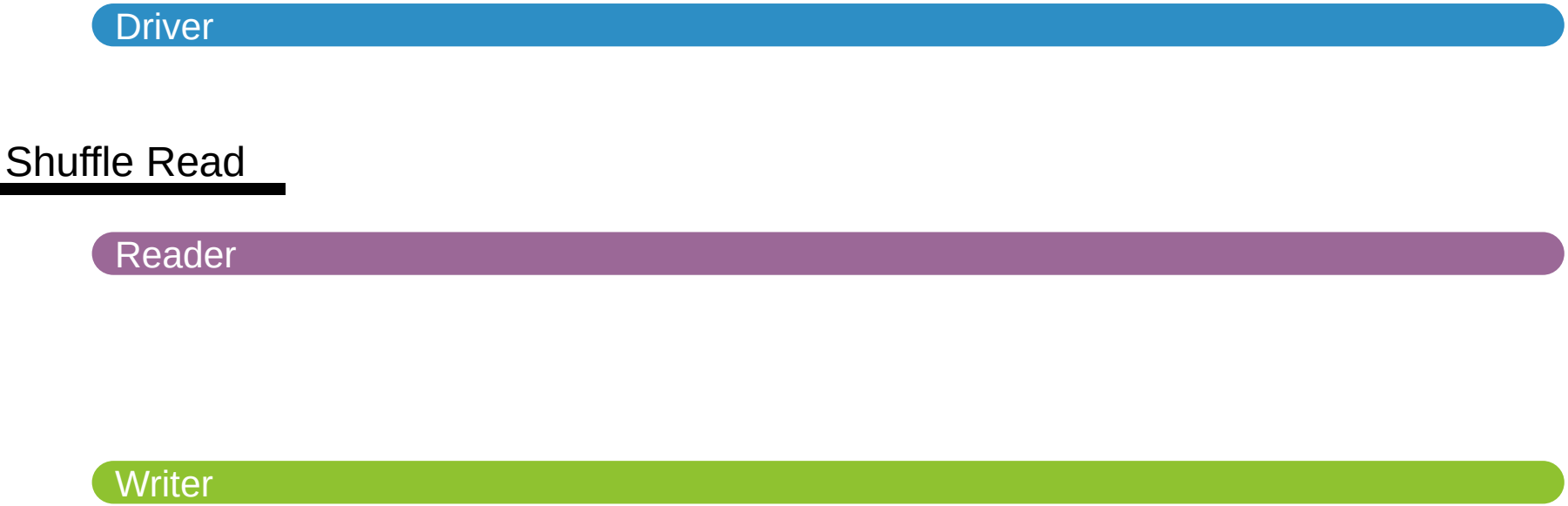
# Spark's Shuffle Basics



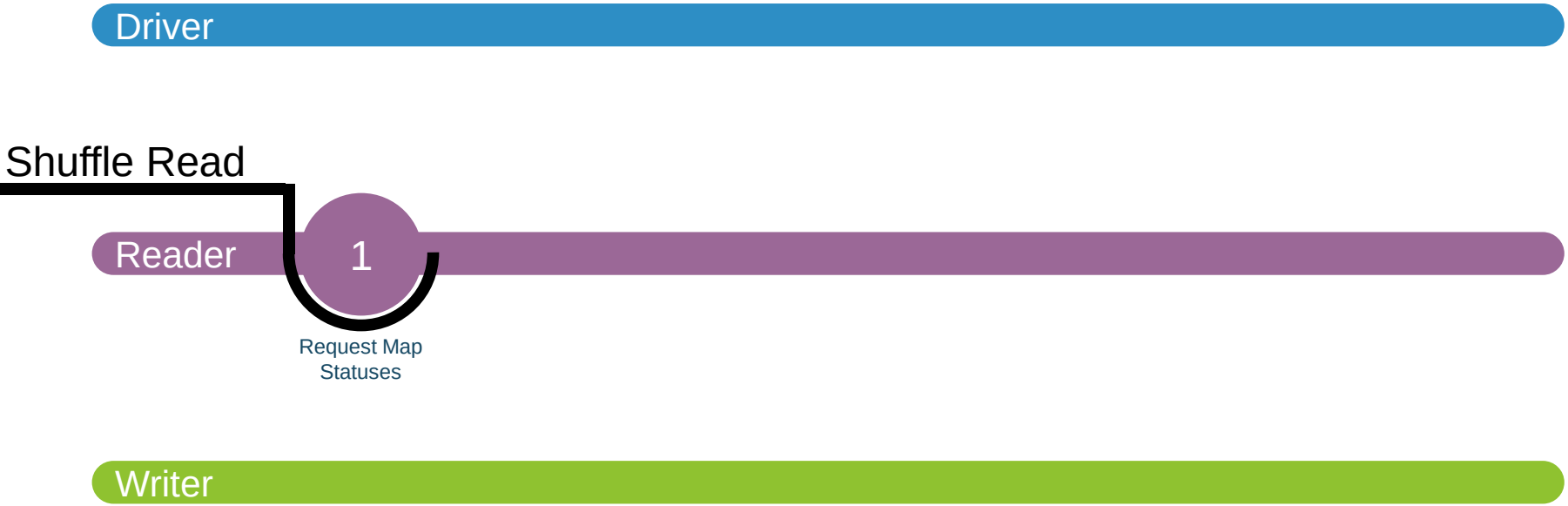
# Spark's Shuffle Read Protocol



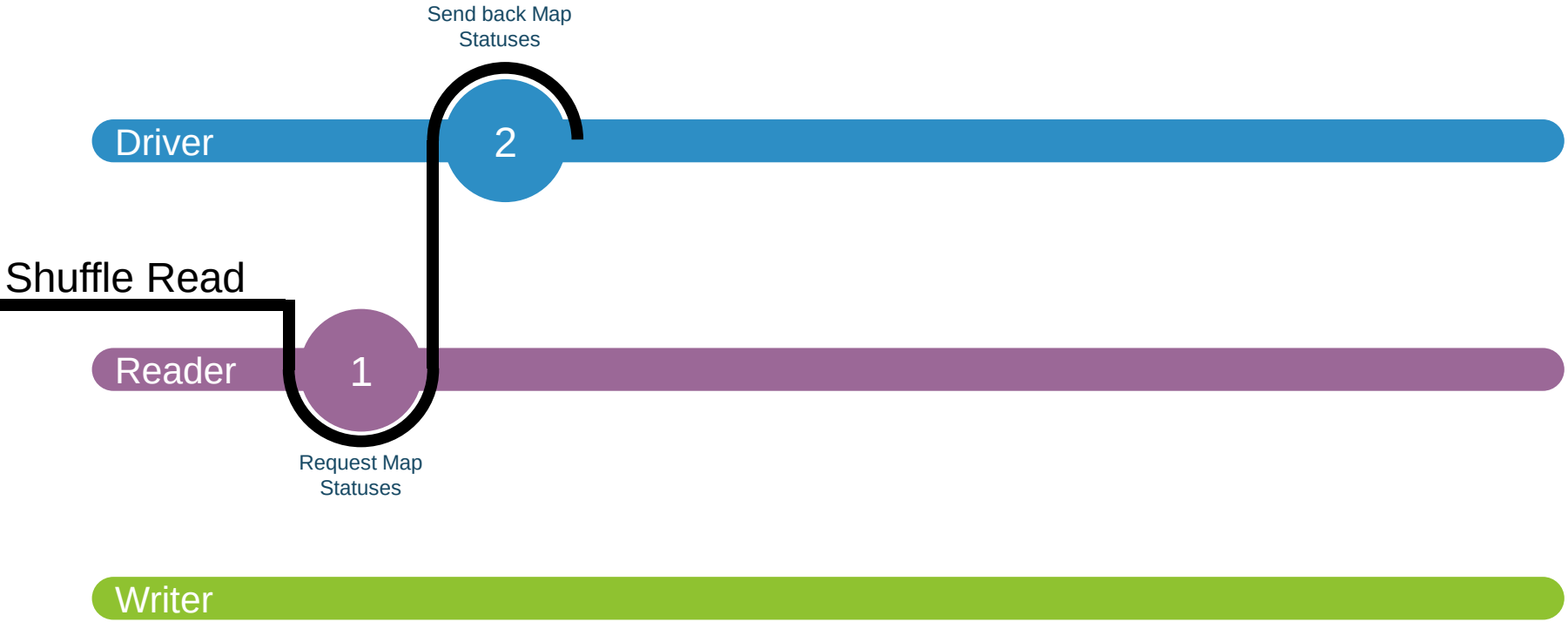
# Spark's Shuffle Read Protocol



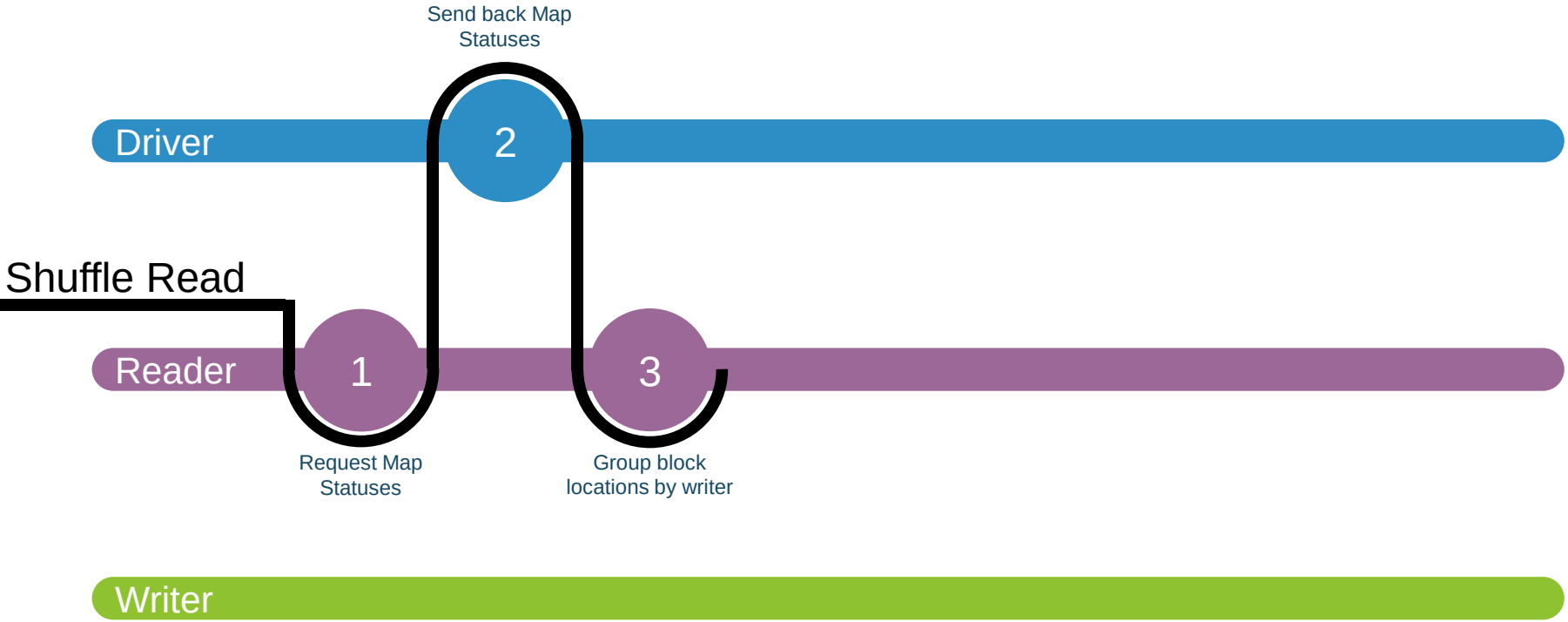
# Spark's Shuffle Read Protocol



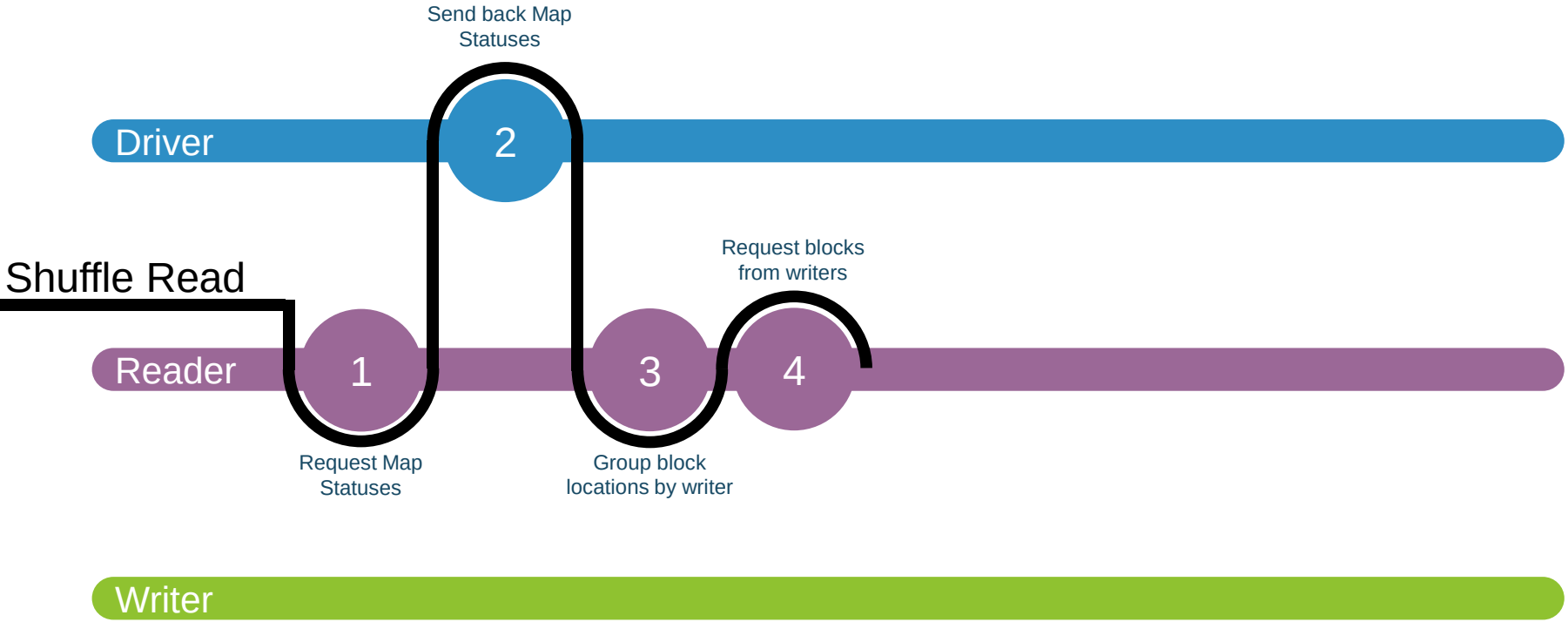
# Spark's Shuffle Read Protocol



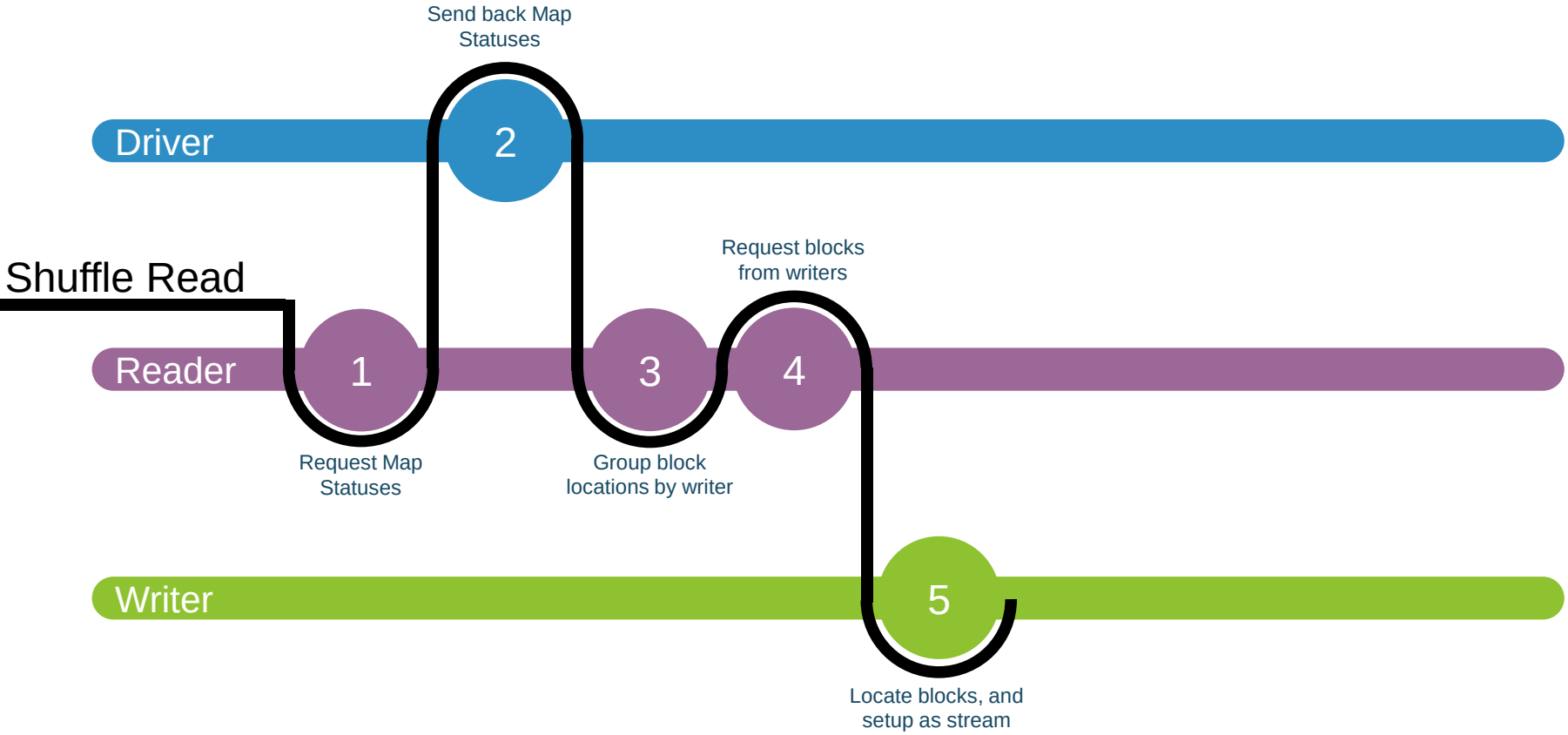
# Spark's Shuffle Read Protocol



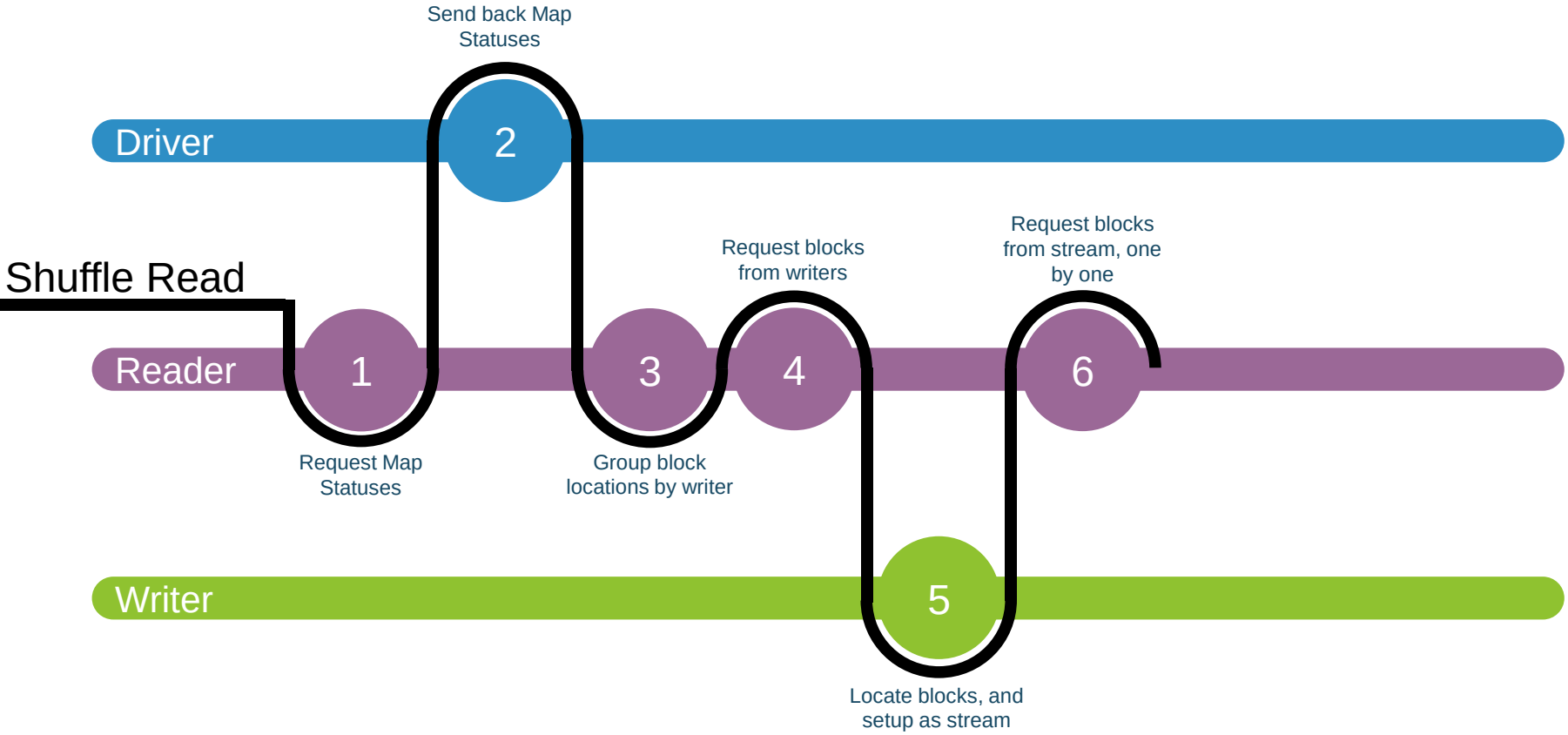
# Spark's Shuffle Read Protocol



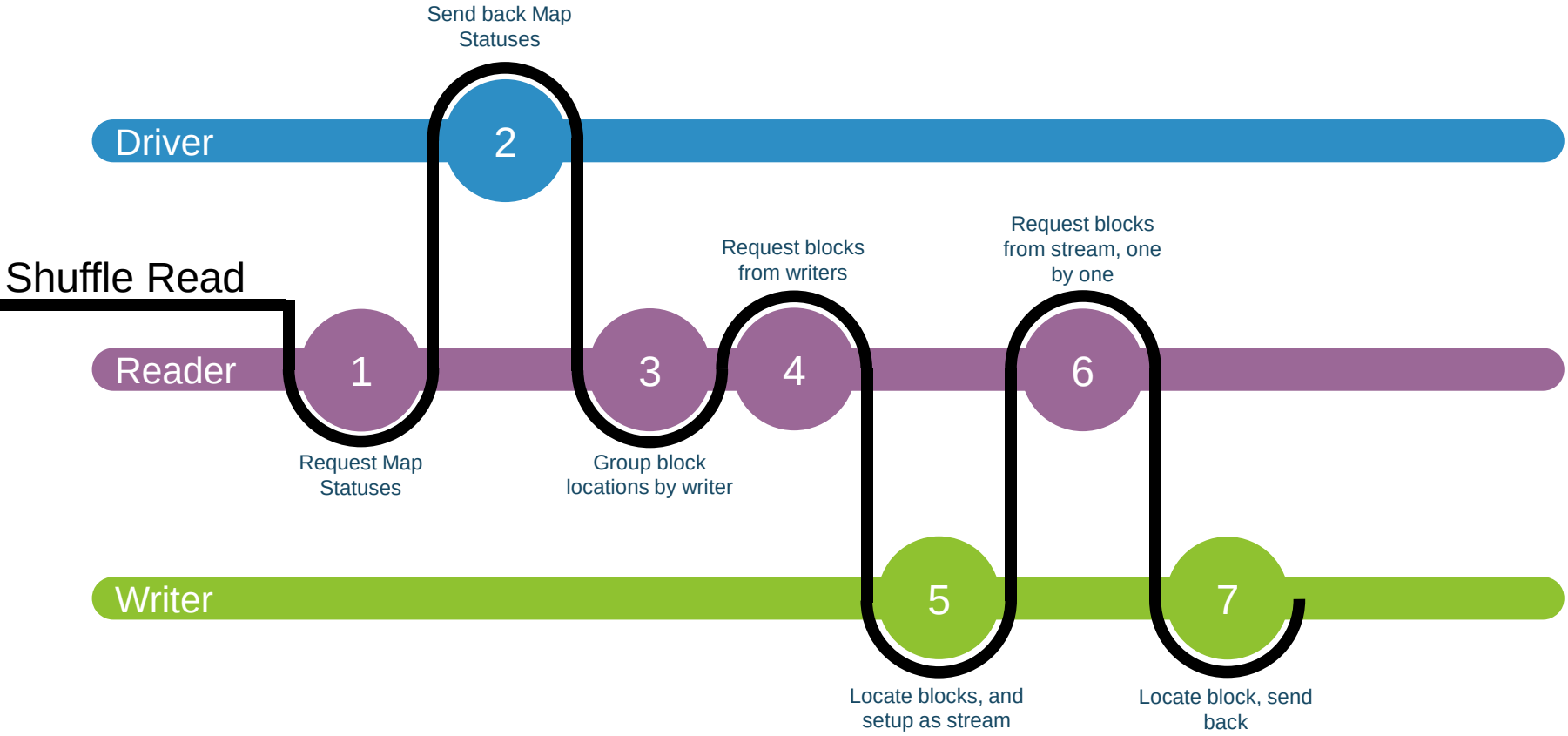
# Spark's Shuffle Read Protocol



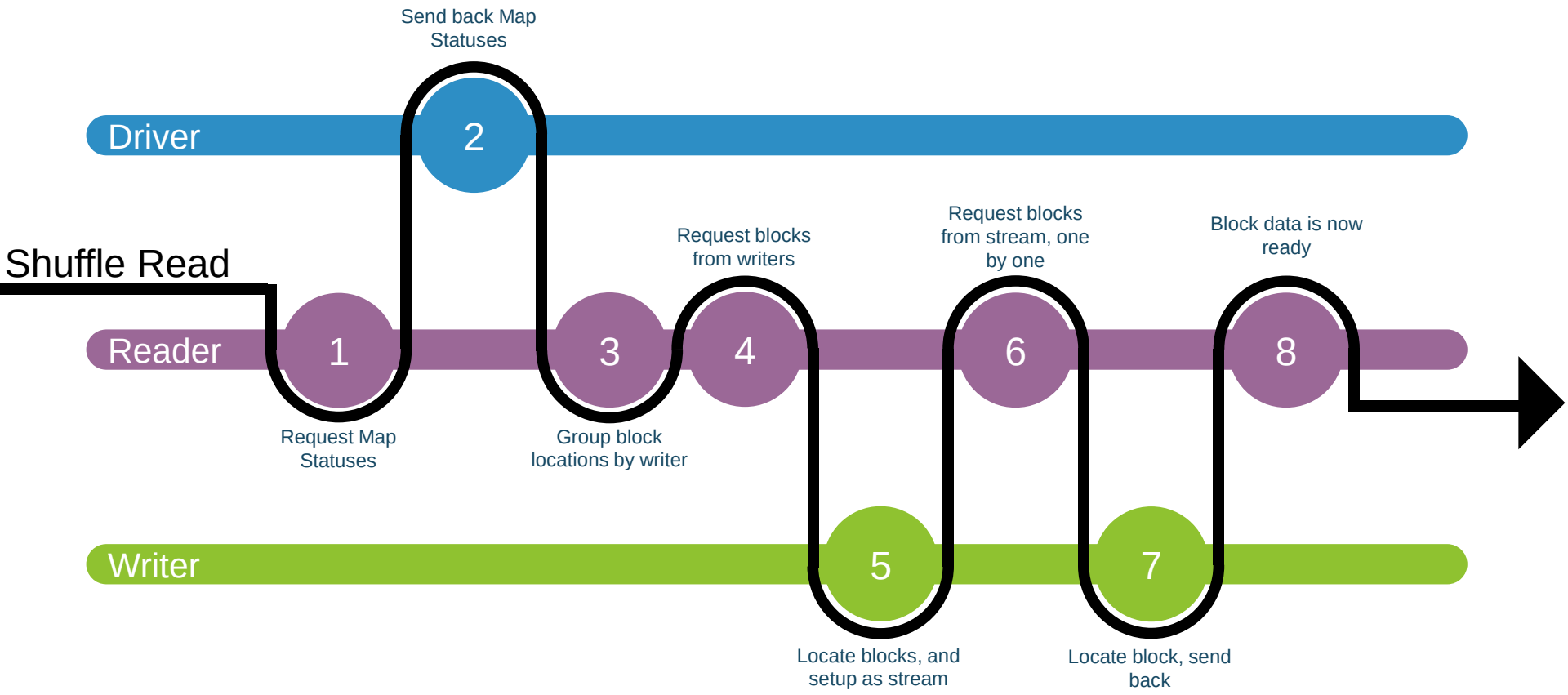
# Spark's Shuffle Read Protocol



# Spark's Shuffle Read Protocol



# Spark's Shuffle Read Protocol



# The Cost of Shuffling

- Shuffling is very expensive in terms of CPU, RAM, disk and network IOs
- Spark users try to avoid shuffles as much as they can
- Speedy shuffles can relieve developers of such concerns, and simplify applications

Accelerating Shuffle with RDMA

# SparkRDMA Shuffle Plugin

# SparkRDMA

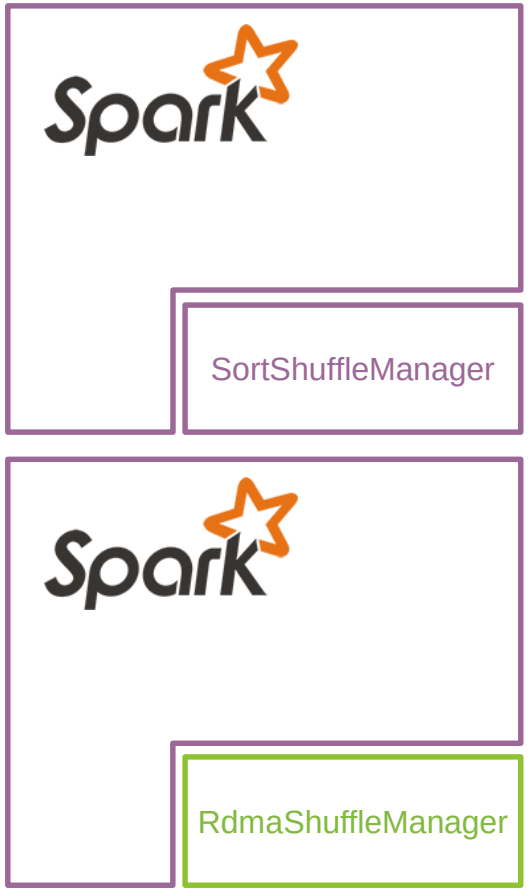
- Dedicated session at Spark Summit Europe 2017: Accelerating Shuffle: A Tailor-Made RDMA Solution for Apache Spark

- Open-source and free to use: https://github.com/Mellanox/SparkRDMA

- Supports any RDMA-capable device
  - Ethernet (RoCE – RDMA over Converged Ethernet)
  - InfiniBand

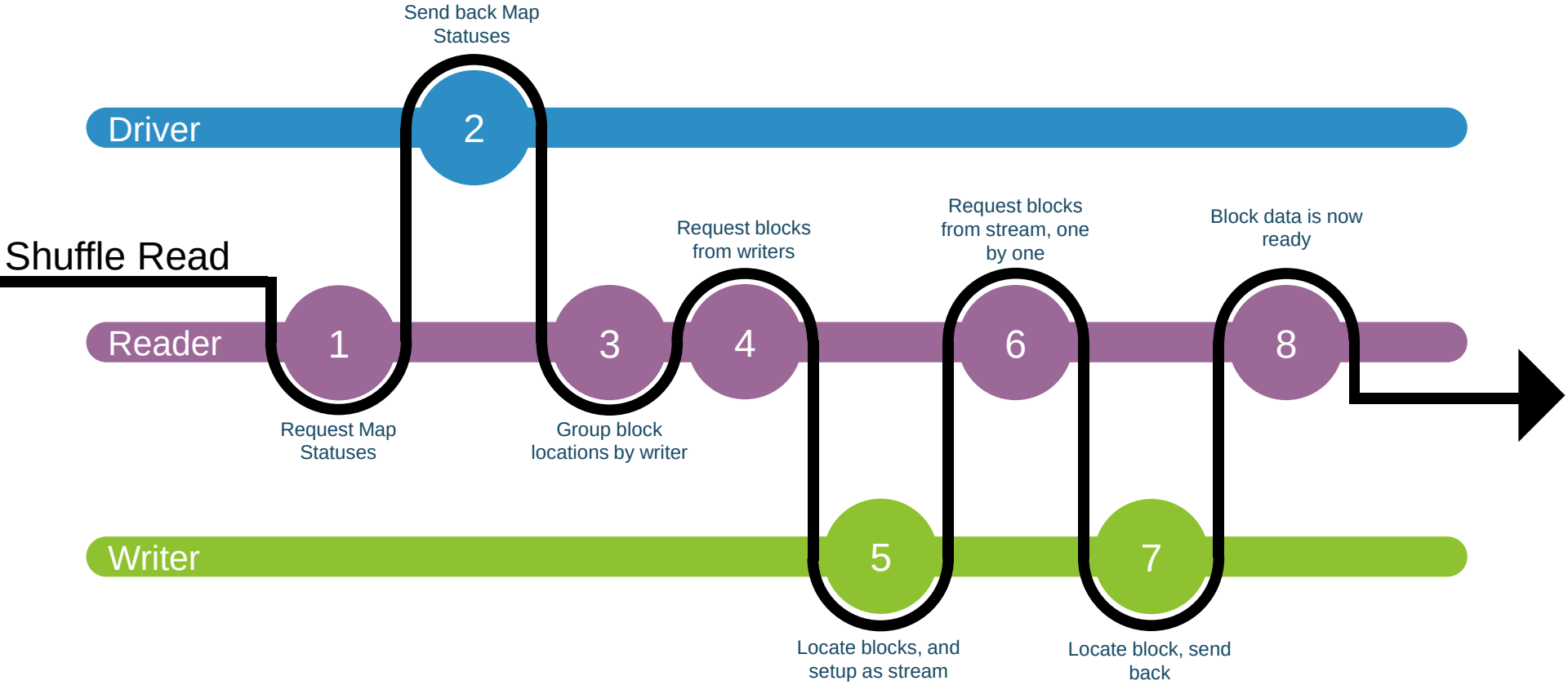
# SparkRDMA - Design Notes

- Entire Shuffle-related communication is done with RDMA
  - RPC messaging for meta-data transfers
  - Block transfers
- SparkRDMA is an independent plugin
  - Implements the ShuffleManager interface
  - No changes to Spark's code – use with any existing Spark installation
- Reuses Spark facilities
  - Maximize reliability
  - Minimize impact on the data
- No functionality loss of any kind, SparkRDMA supports:
  - Compression
  - Spilling to disk
  - Recovery from failed map or reduce tasks

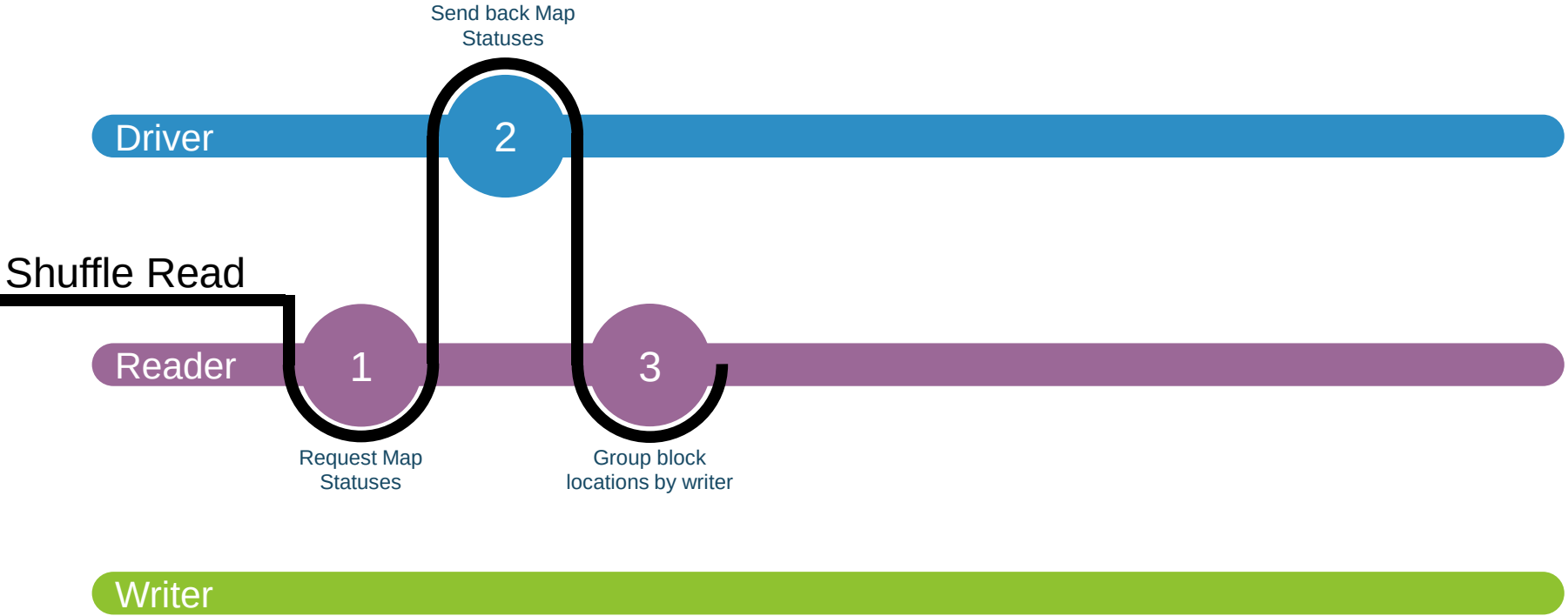


# Shuffle Read Protocol – Standard vs. RDMA

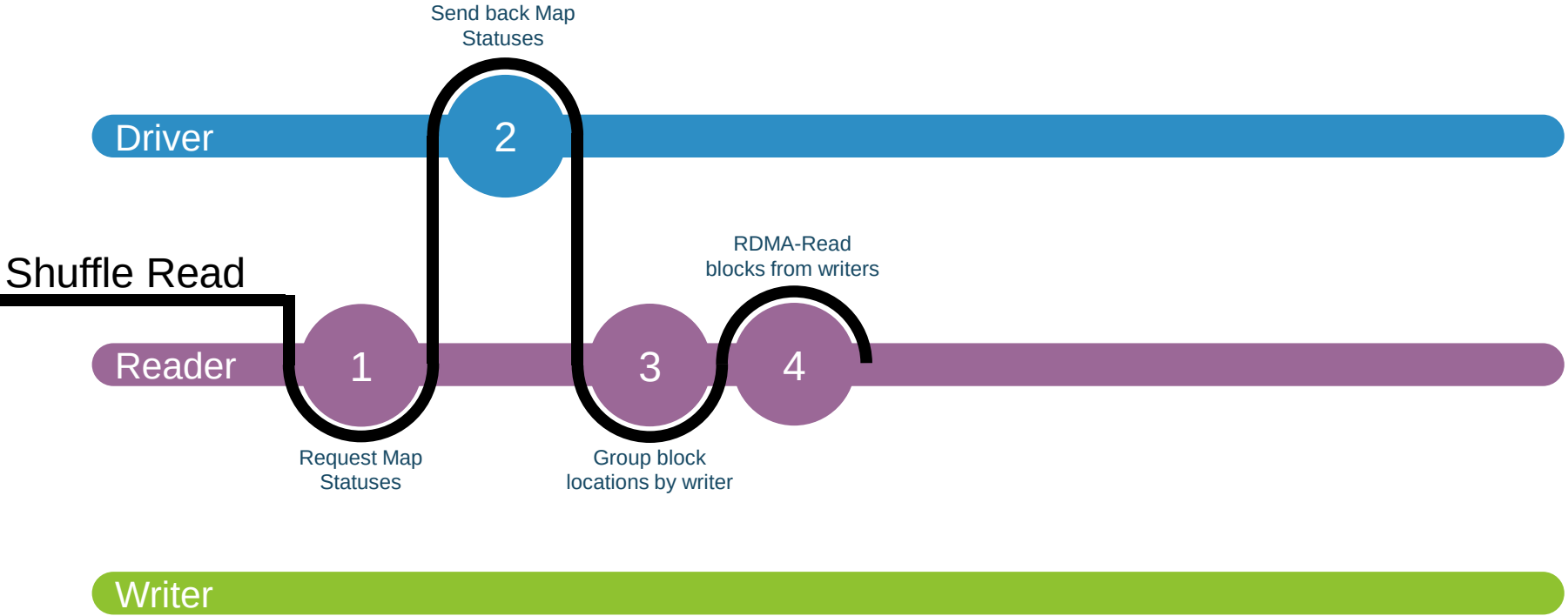
# Shuffle Read Protocol – Standard vs. RDMA



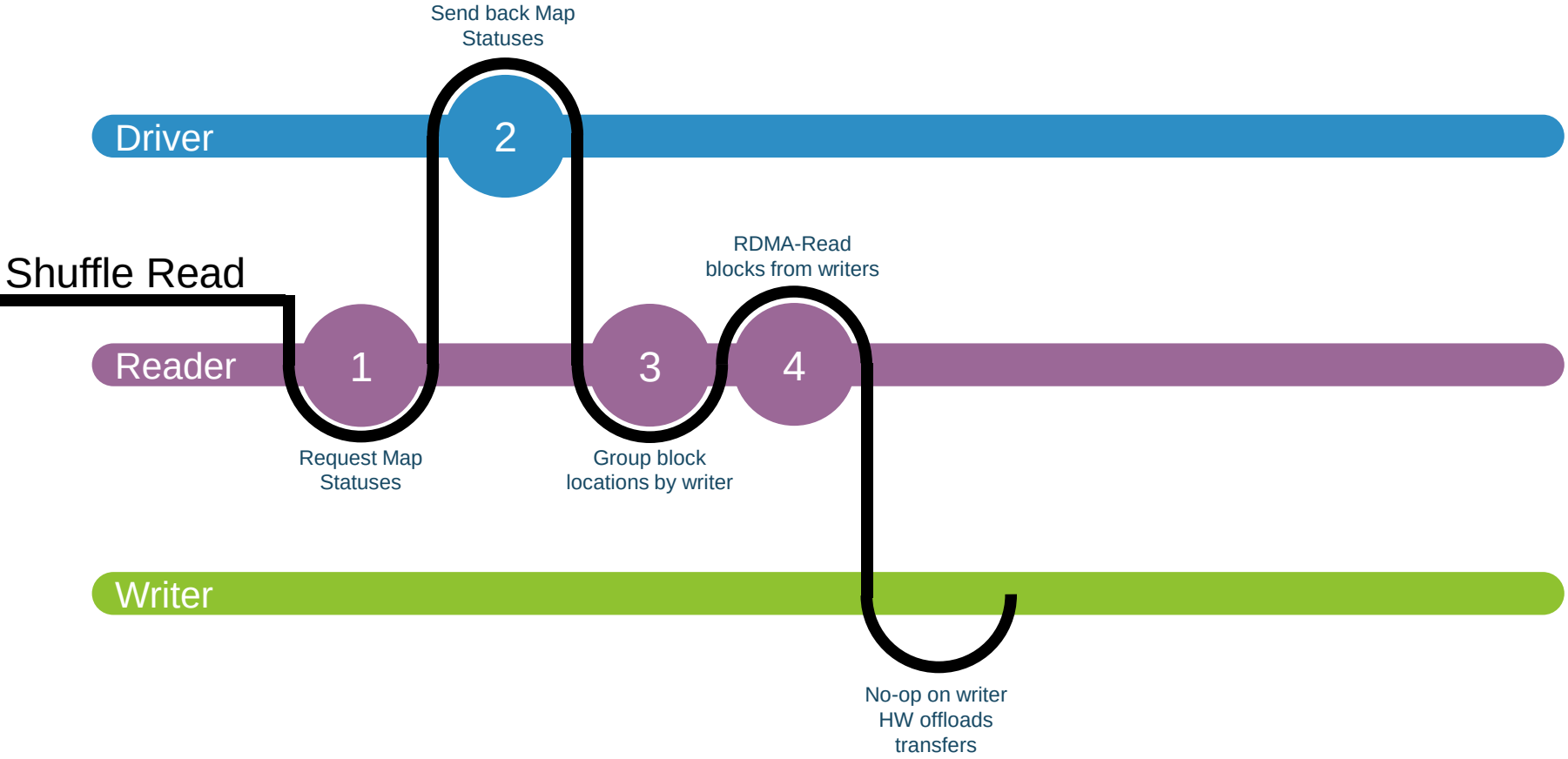
# Shuffle Read Protocol – Standard vs. RDMA



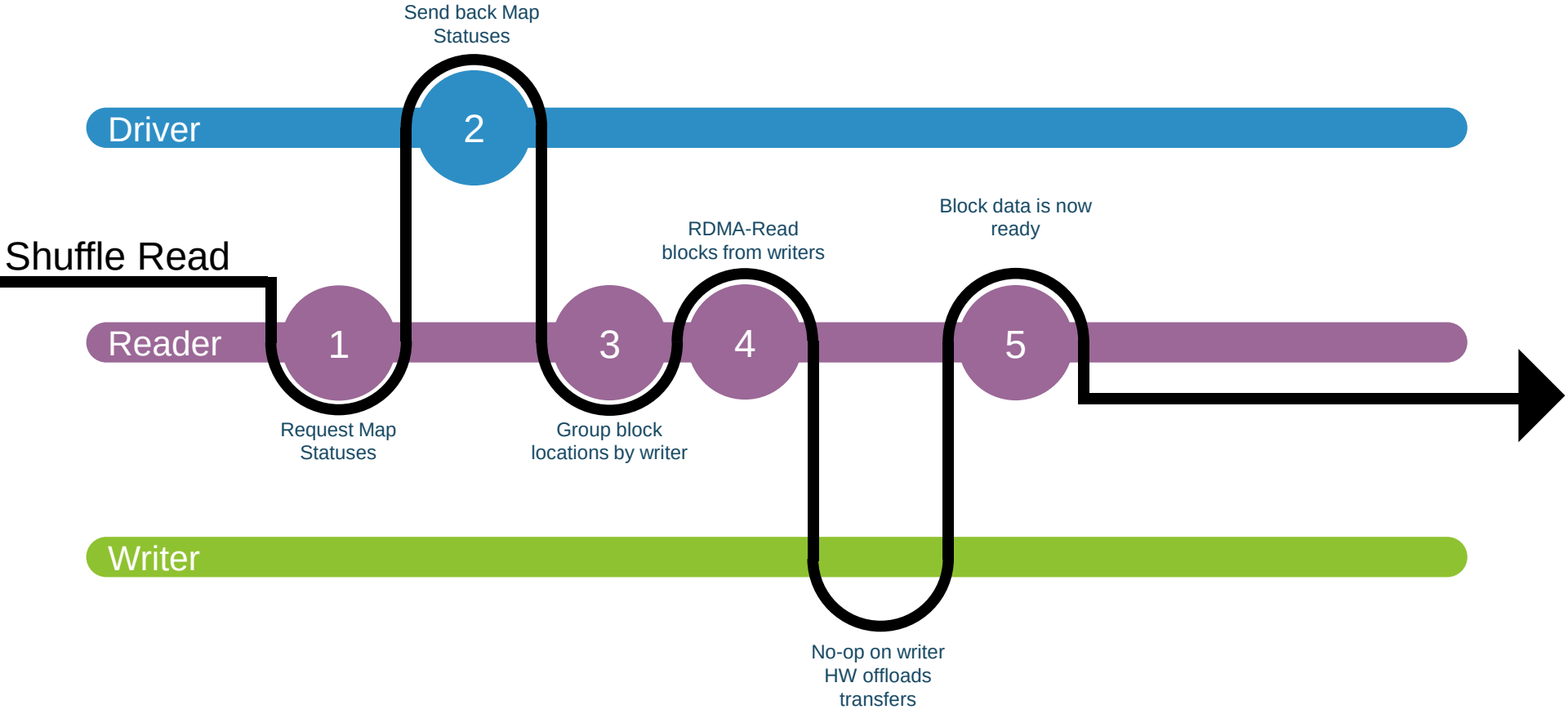
# Shuffle Read Protocol – Standard vs. RDMA

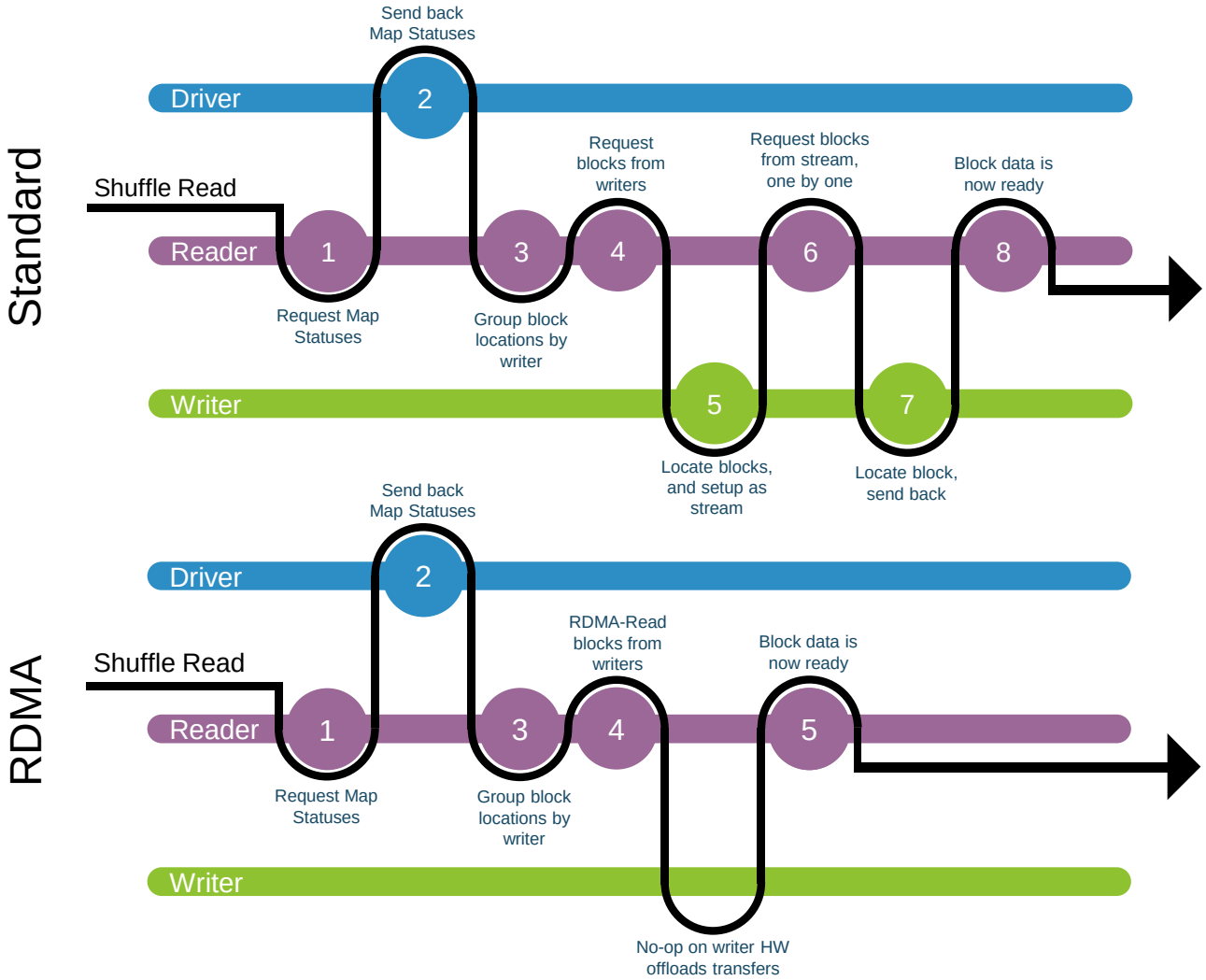


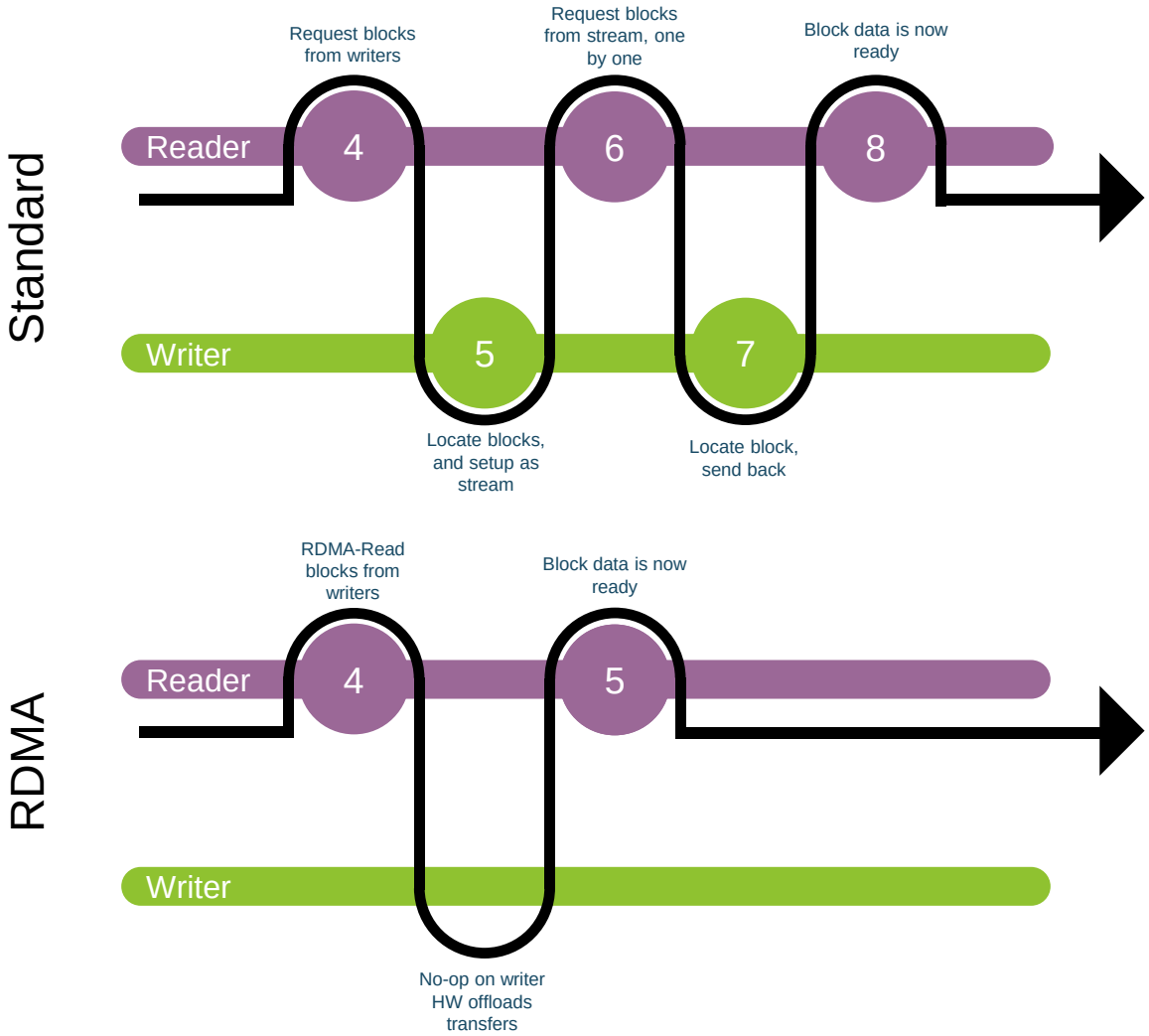
# Shuffle Read Protocol – Standard vs. RDMA



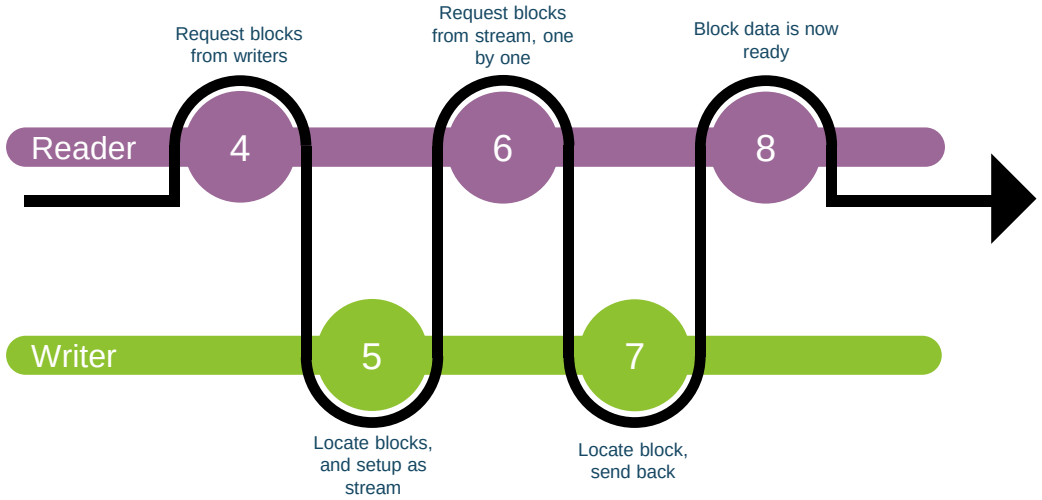
# Shuffle Read Protocol – Standard vs. RDMA



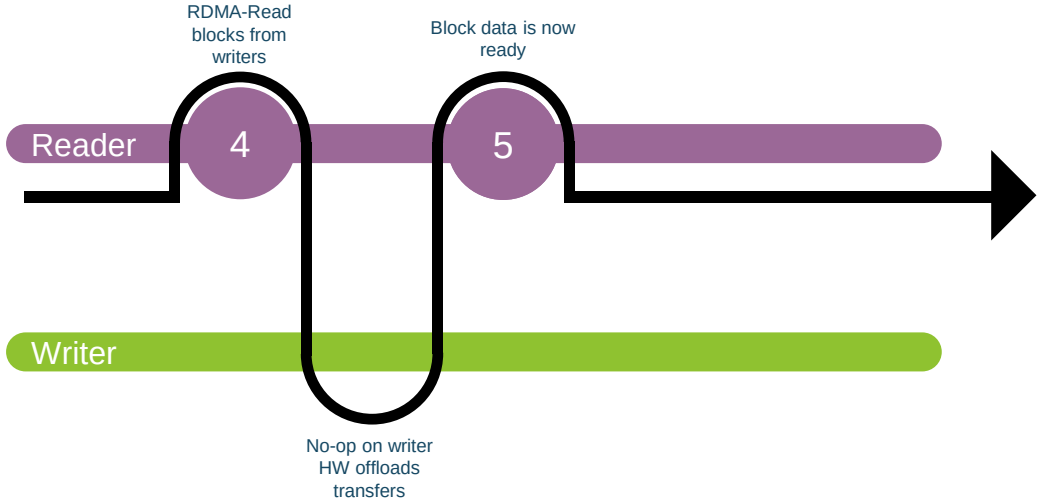

Standard
Send back
Map Statuses
Driver
2
Shuffle Read
Reader
1
3
4
6
8
Request
blocks from
writers
Request blocks
from stream,
one by one
Block data is
now ready
Request Map
Statuses
Group block
locations by
writer
Writer
5
7
Locate blocks,
and setup as
stream
Locate block,
send back
RDMA
Send back
Map Statuses
Driver
2
Shuffle Read
Reader
1
3
4
5
RDMA-Read
blocks from
writers
Block data is
now ready
Request Map
Statuses
Group block
locations by
writer
Writer
No-op on writer HW
offloads transfers


Standard
Request blocks from writers
Request blocks from stream, one by one
Block data is now ready
Reader
4
6
8
Writer
5
7
Locate blocks, and setup as stream
Locate block, send back
RDMA
RDMA-Read blocks from writers
Block data is now ready
Reader
4
5
Writer
No-op on writer HW offloads transfers

Standard




RDMA




Server-side:

- ✓ 0 CPU
- ✓ Shuffle transfers are not blocked by GC in executor
- ✓ No buffering

Client-side:

- ✓ Instant transfers
- ✓ Reduced messaging
- ✓ Direct, unblocked access to remote blocks

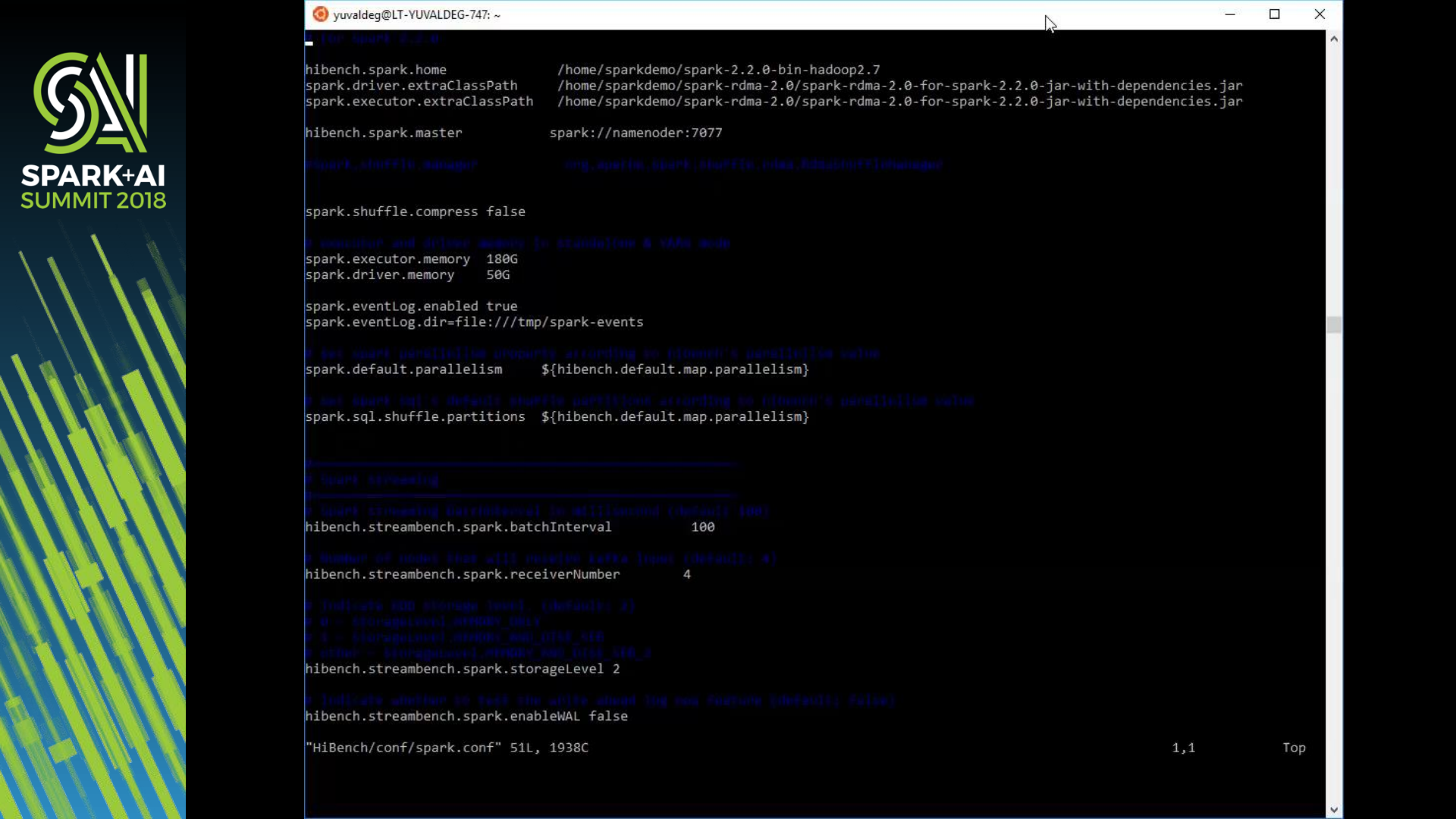
# Benefits

- Substantial improvements in:
  - Block transfer times: latency and total transfer time
  - Memory consumption and management
  - CPU utilization
- Easy to deploy and configure:
  - Packed into a single JAR file
  - Plugin is enabled through a simple configuration handle
  - Allows finer tuning with a set of configuration handles
- Configuration and deployment are on a per-job basis:
  - Can be deployed incrementally
  - May be limited to Shuffle-intensive jobs

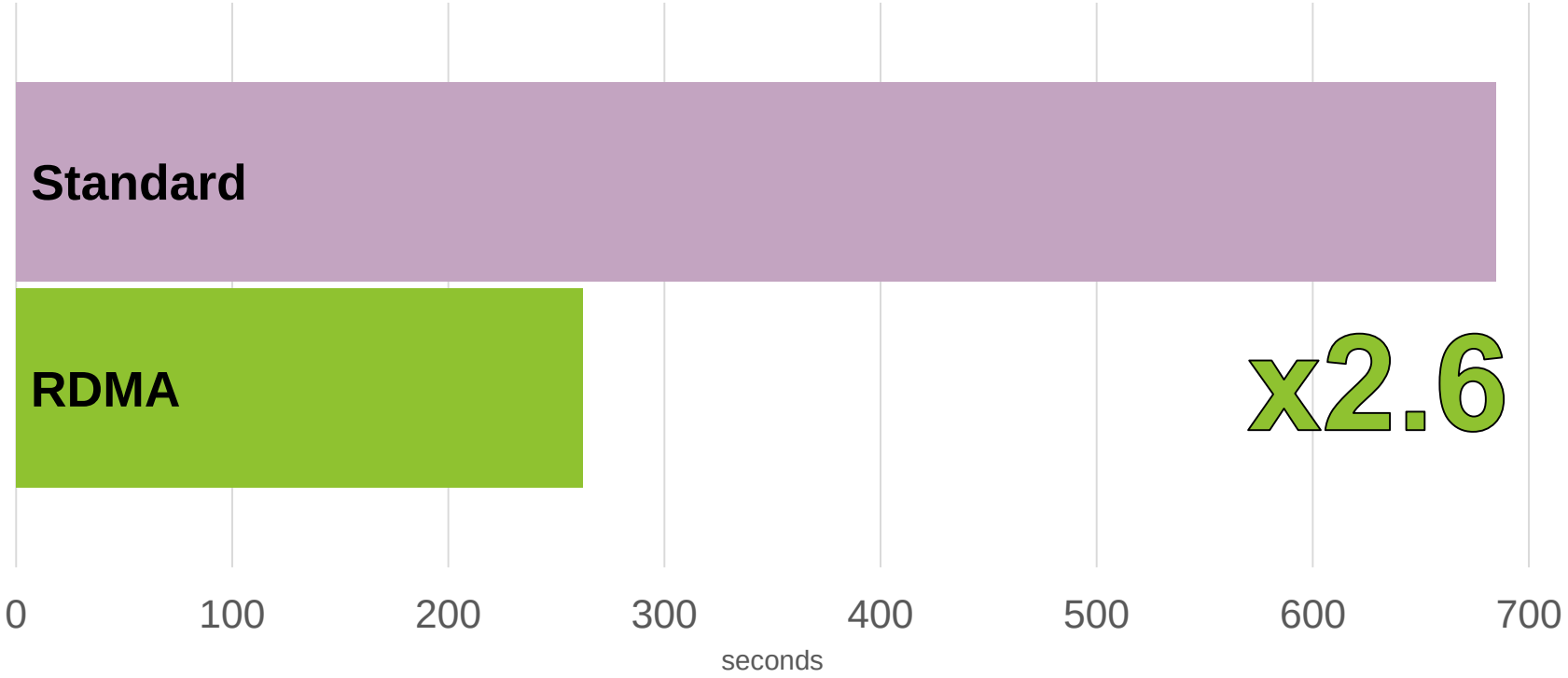
# Demo time!

# Demo Testbed

- Hardware:
  - 8 Azure “h16mr” VM instances
  - Intel Haswell E5-2667 V3
  - InfiniBand FDR (56Gb/s)
  - 224GiB RAM
  - 2000GiB SSD for temporary storage
- Workload:
  - HiBench TeraSort
  - Size: “gigantic” (320GB)
- Ubuntu 16.04
- HDFS on Hadoop 2.7.4
  - No replication
- Spark 2.2.0
  - 1 Master
  - 7 Workers
  - 16 active Spark cores on each node, 112 total

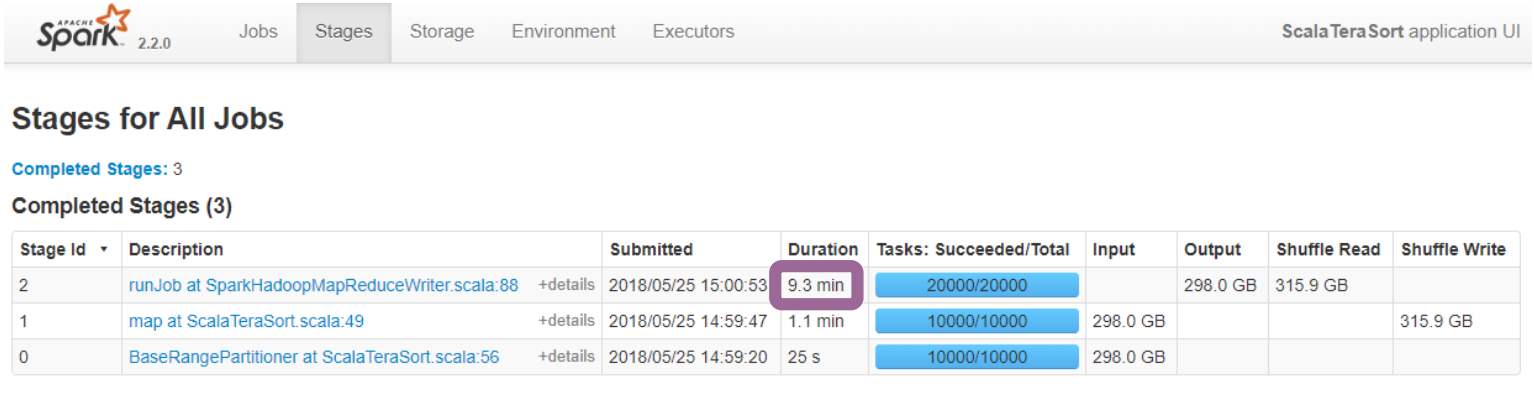

yuvaldeg@LT-YUVALDEG-747: ~
hibench.spark.home /home/sparkdemo/spark-2.2.0-bin-hadoop2.7
spark.driver.extraClassPath /home/sparkdemo/spark-rdma-2.0/spark-rdma-2.0-for-spark-2.2.0-jar-with-dependencies.jar
spark.executor.extraClassPath /home/sparkdemo/spark-rdma-2.0/spark-rdma-2.0-for-spark-2.2.0-jar-with-dependencies.jar
hibench.spark.master spark://namenode:7077
spark.shuffle.compress false
spark.executor.memory 180G
spark.driver.memory 50G
spark.eventLog.enabled true
spark.eventLog.dir=file:///tmp/spark-events
spark.default.parallelism ${hibench.default.map.parallelism}
spark.sql.shuffle.partitions ${hibench.default.map.parallelism}
hibench.streambench.spark.batchInterval 100
hibench.streambench.spark.receiverNumber 4
hibench.streambench.spark.storageLevel 2
hibench.streambench.spark.enableWAL false
"HiBench/conf/spark.conf" 51L, 1938C
1,1
Top

# TeraSort - Performance Results

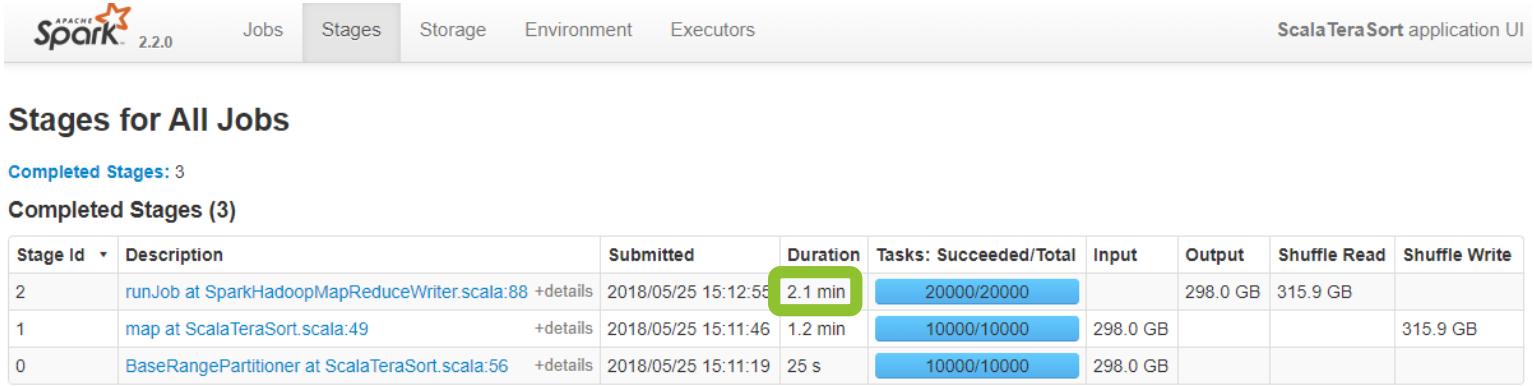


# x4.4 Faster Shuffles!

Standard




RDMA



# x1000 Faster Transfers!

Standard

**Summary Metrics for 20000 Completed Tasks**

| Metric | Min | 25th percentile | Median | 75th percentile | Max |
|---|---|---|---|---|---|
| Duration | 1.0 s | 2 s | 3 s | 4 s | 8 s |
| GC Time | 0 ms | 0 ms | 0 ms | 0 ms | 3 s |
| Output Size / Records | 8.8 MB / 92630 | 13.9 MB / 145609 | 15.1 MB / 158772 | 16.5 MB / 173356 | 23.1 MB / 242307 |
| Shuffle Read Blocked Time | 5 ms | 2 s | 2 s | 3 s | 7 s |
| Shuffle Read Size / Records | 9.4 MB / 92630 | 14.7 MB / 145609 | 16.1 MB / 158772 | 17.5 MB / 173356 | 24.5 MB / 242307 |
| Shuffle Remote Reads | 7.9 MB | 12.6 MB | 13.8 MB | 15.0 MB | 21.5 MB |

RDMA

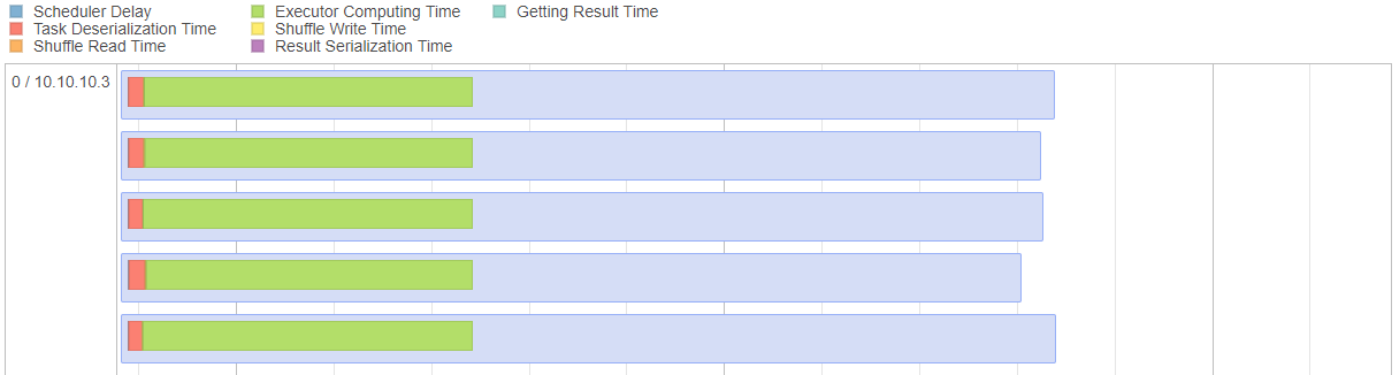
**Summary Metrics for 20000 Completed Tasks**

| Metric | Min | 25th percentile | Median | 75th percentile | Max |
|---|---|---|---|---|---|
| Duration | 0.3 s | 0.6 s | 0.7 s | 0.8 s | 3 s |
| GC Time | 0 ms | 0 ms | 0 ms | 0 ms | 0.2 s |
| Output Size / Records | 8.8 MB / 92630 | 13.9 MB / 145609 | 15.1 MB / 158772 | 16.5 MB / 173356 | 23.1 MB / 242307 |
| Shuffle Read Blocked Time | 0 ms | 1 ms | 2 ms | 3 ms | 0.1 s |
| Shuffle Read Size / Records | 9.4 MB / 92630 | 14.7 MB / 145609 | 16.1 MB / 158772 | 17.5 MB / 173356 | 24.5 MB / 242307 |
| Shuffle Remote Reads | 7.9 MB | 12.6 MB | 13.8 MB | 15.0 MB | 21.4 MB |

# 0 Shuffle Read Time

Standard

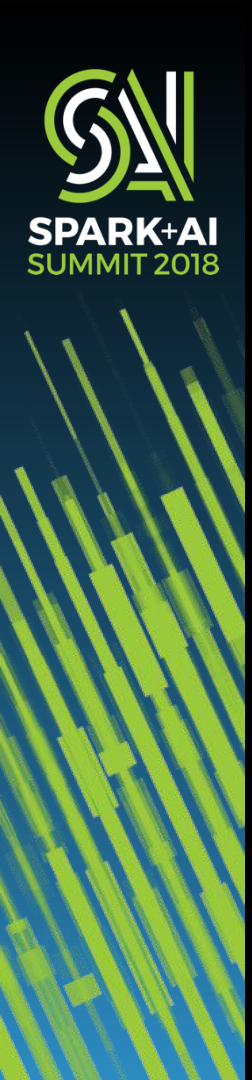

RDMA



# Recap

- SR-IOV+RDMA comes to Azure H and N-series in Fall 2018
- Support for all major MPI
  - MVAPICH, OpenMPI, Intel MPI, Platform MPI, etc.
- General-purpose RDMA support
  - Support for SparkRDMA, Caffe2, TensorFlow or any other RDMA application

- Be on the lookout for more at SC'18 !




# Thank you.