Spark DPU

Ofir Farjon | UCF Workshop 2023

Agenda

- Introduction
- Motivation - Shuffle
- Proposed Solution
- **Architecture Components**
- Current Status

What is Spark?

A brief history

2004

Introduction of MapReduce

Jeffrey Dean and Sanjay Ghemawat published a paper called “MapReduce: Simplified Data Processing on Large Clusters”.

2006.

Hadoop

Mike Cafarella and Doug Cutting—were so convinced of MapReduce’s importance that they decided to build a free clone of the system from scratch. They eventually called their project Hadoop.

2014

Apache Spark

Spark (like Hadoop) is a data processing engine, but it has some benefits over Hadoop, e.g. in-memory processing

2019

SparkUCX Shuffle Plugin

Uses RDMA to perform shuffle data transfers in Spark.

2020

Nvidia Rapids for Spark

Accelerator for Apache Spark.

2021

SparkUCX with AM

Shuffle plugin to perform shuffle data transfers using UCX with AM.

2023

SparkDPU

Shuffle plugin for Spark utilizing Nvidia DPU and UCX for offloading network communication and using NVMe for data storage.

What is Spark?

MapReduce

- MapReduce is a programming model for processing and generation of large datasets.
- Map function, written by the user, takes an input pair and produces a set of intermediate key-value pairs.
- The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function
- The Reduce function, also written by the user, takes an intermediate key and a set of values for that key. It merges these values to form a possibly smaller set.
- Word count: the problem of counting the number of occurrences of each word in a large collection of documents

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

MapReduce: Simplified Data Processing on Large Clusters

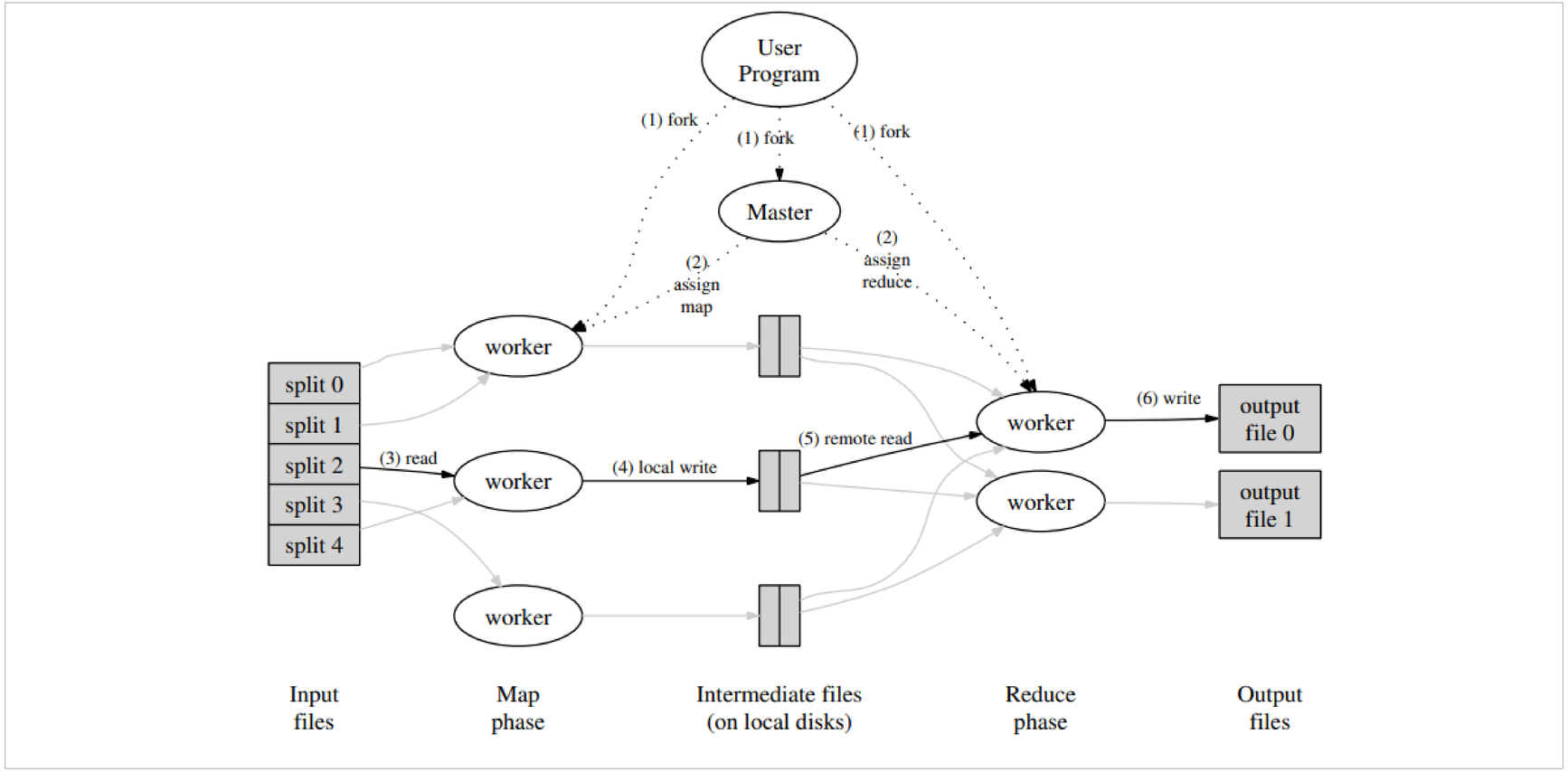
Motivation

What is Shuffle?

- In Apache Spark, shuffling happens when data need to be redistributed across the cluster.
- By default, shuffle relies on traditional socket-based TCP/ IP communication.
- Shuffling can significantly impact performance in Spark, especially for large datasets:

  “When data is growing explosively over time, the amount of data that needs Shuffle is also increasing, and it is found that 30% of the time is spent on Shuffle exchanging data in overall task execution."

Execution Overview



MapReduce: Simplified Data Processing on Large Clusters

Proposed Solution

Design

- Host
  - Writes data to the NVMe drive
  - Updates DPU with the locations of the blocks.
  - Sends fetch requests to DPUs.
- DPU
  - Serves fetch block requests from other hosts by reading blocks from the NVMe to the host memory and sending them back to the requester using RDMA.

Proposed Solution

Main goal: achieve performance improvement, by leveraging hardware resources utilization

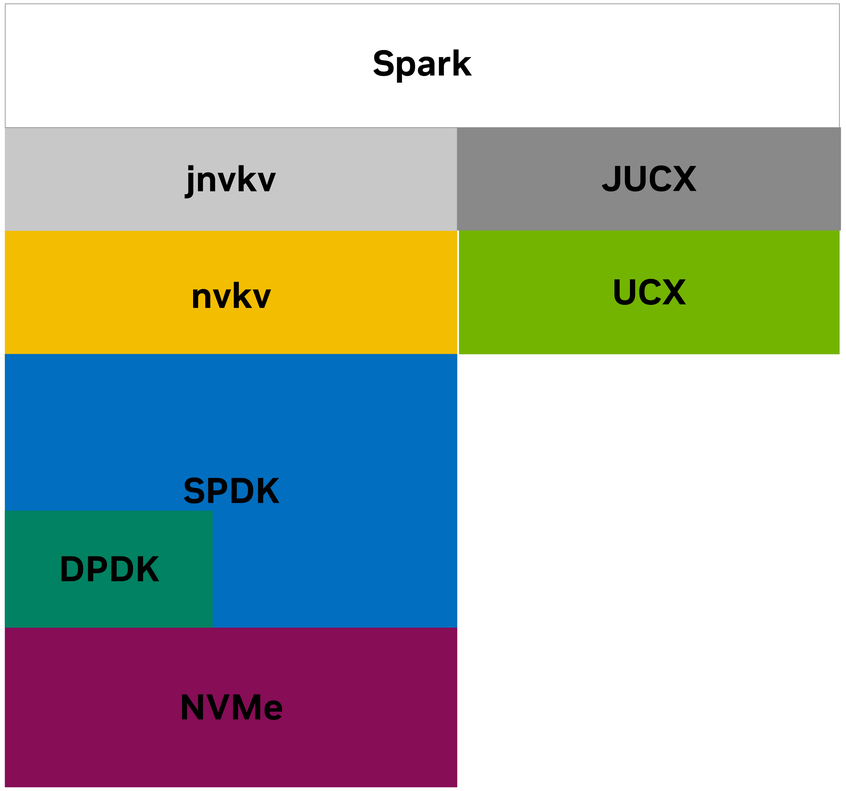
- Network - improve network capabilities
  - Increase bandwidth by using RDMA
  - Future optimization: move the progress task to the DPU by implementing RNDV write operation. According to our tests, it can potentially improve network capabilities.
  - Reduce number of endpoints: instead of having O(#Executors * #Executors) connections, we will only need O(#Executors * #DPUs) = O(#Executors * #nodes) connections.

    #Executors >> #Nodes

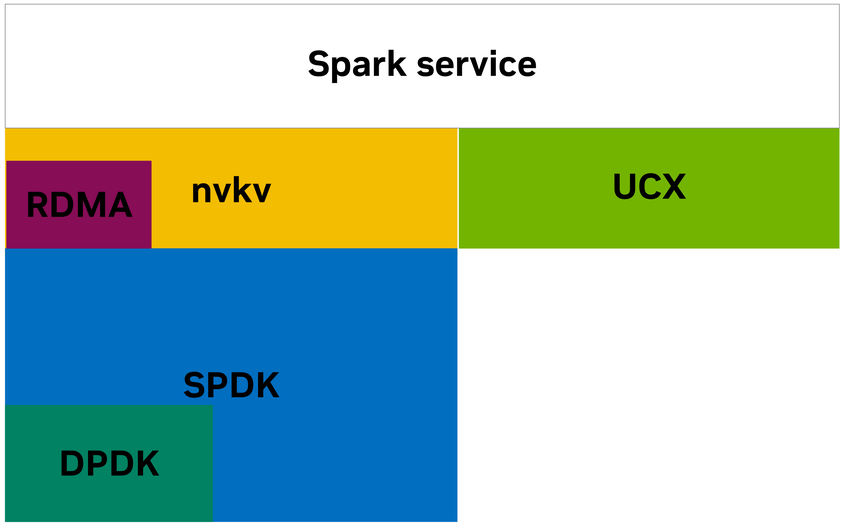
- Storage - improve I/O capabilities:
  - By using multiple fast storage NVMe devices.
- Offloading – improve CPU utilization:
  - By using DPU to handle fetch block requests and progress queues.

Architecture Components

Host

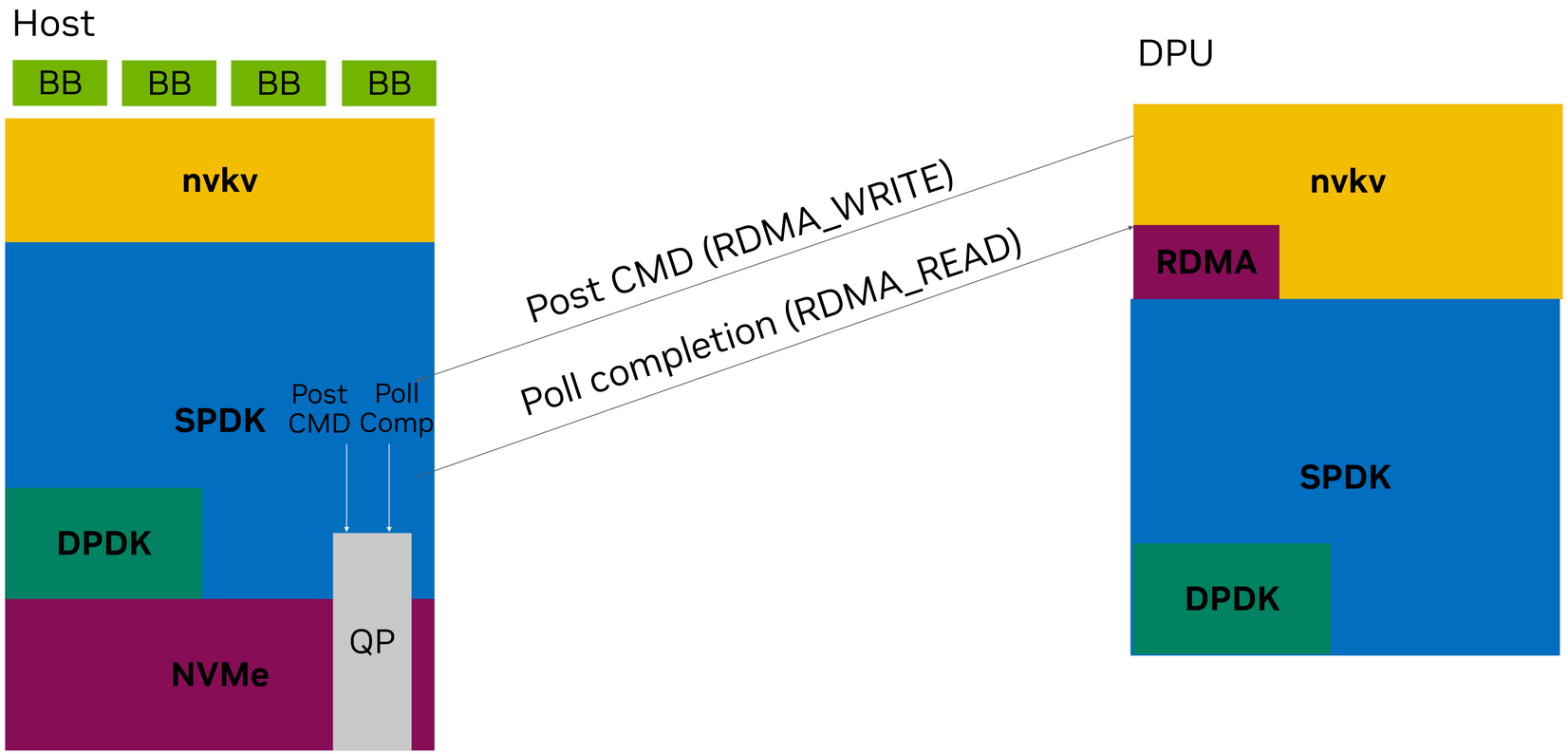


DPU



Storage

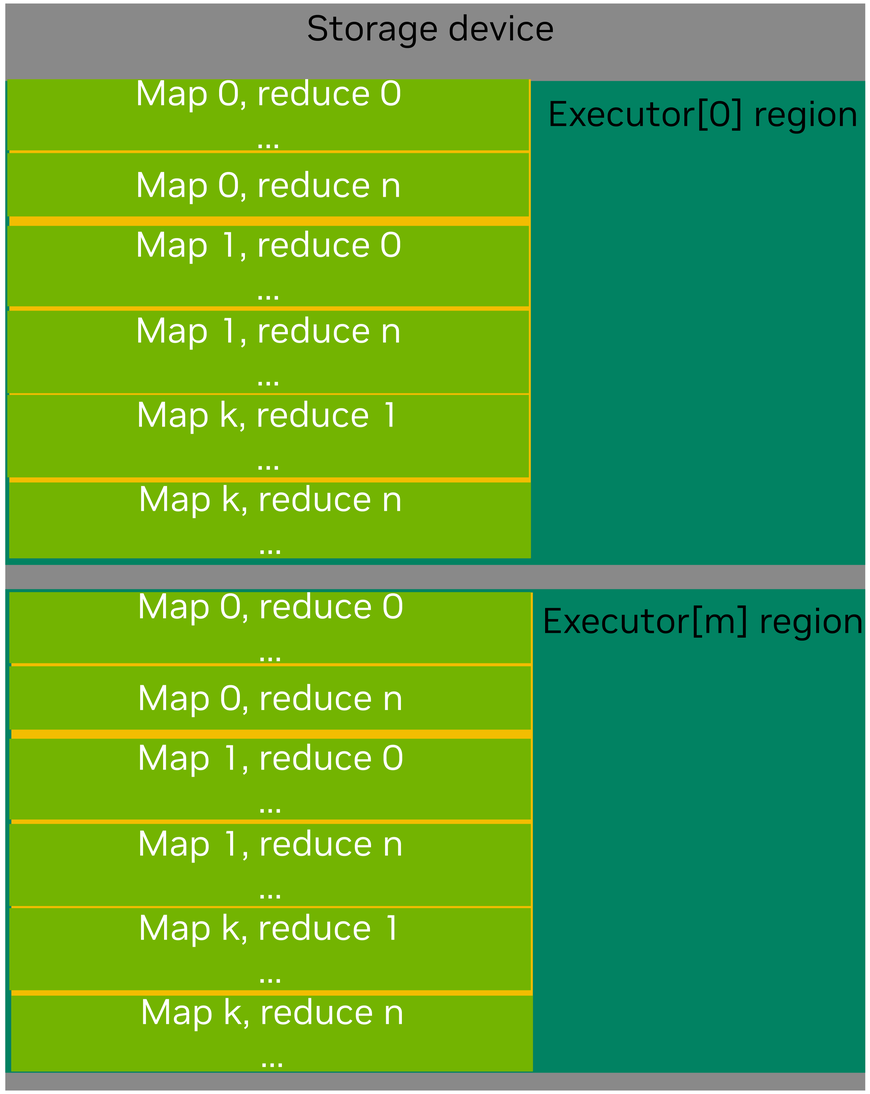
nvkv



Storage

Data Layout

- Raw data
- Block locations are kept in memory (DPU) instead of index files.



Host - DPU protocol

Message types

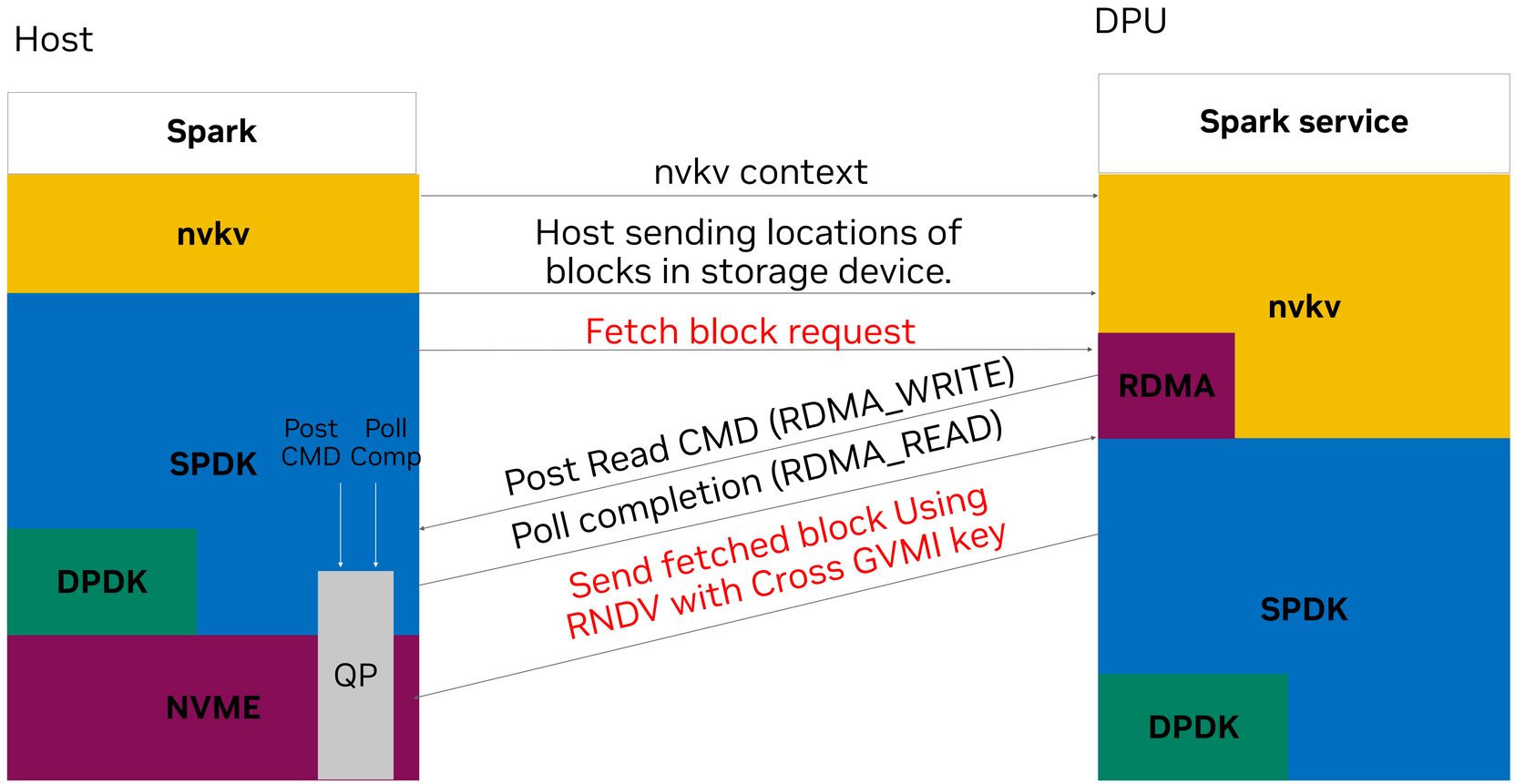
- Host -> DPU
  - nvkv context information
  - Blocks information
  - Fetch block Request
- DPU -> Host
  - nvkv remote context address
  - Fetched block data

SparkDPU Shuffle Manager

- Shuffle Manager
  - Initializes nvkv.
  - Connects to local DPU and sends nvkv context with BB information.
  - Connects to remote nvkv.
  - Establishes connections with all DPUs in the cluster.
- Shuffle Writer
  - Writes blocks to NVMe as raw data.
  - Updates DPU with blocks' offsets and lengths.
- Shuffle Reader
  - Sends fetch block requests to the DPU that belongs to the host owning the blocks.

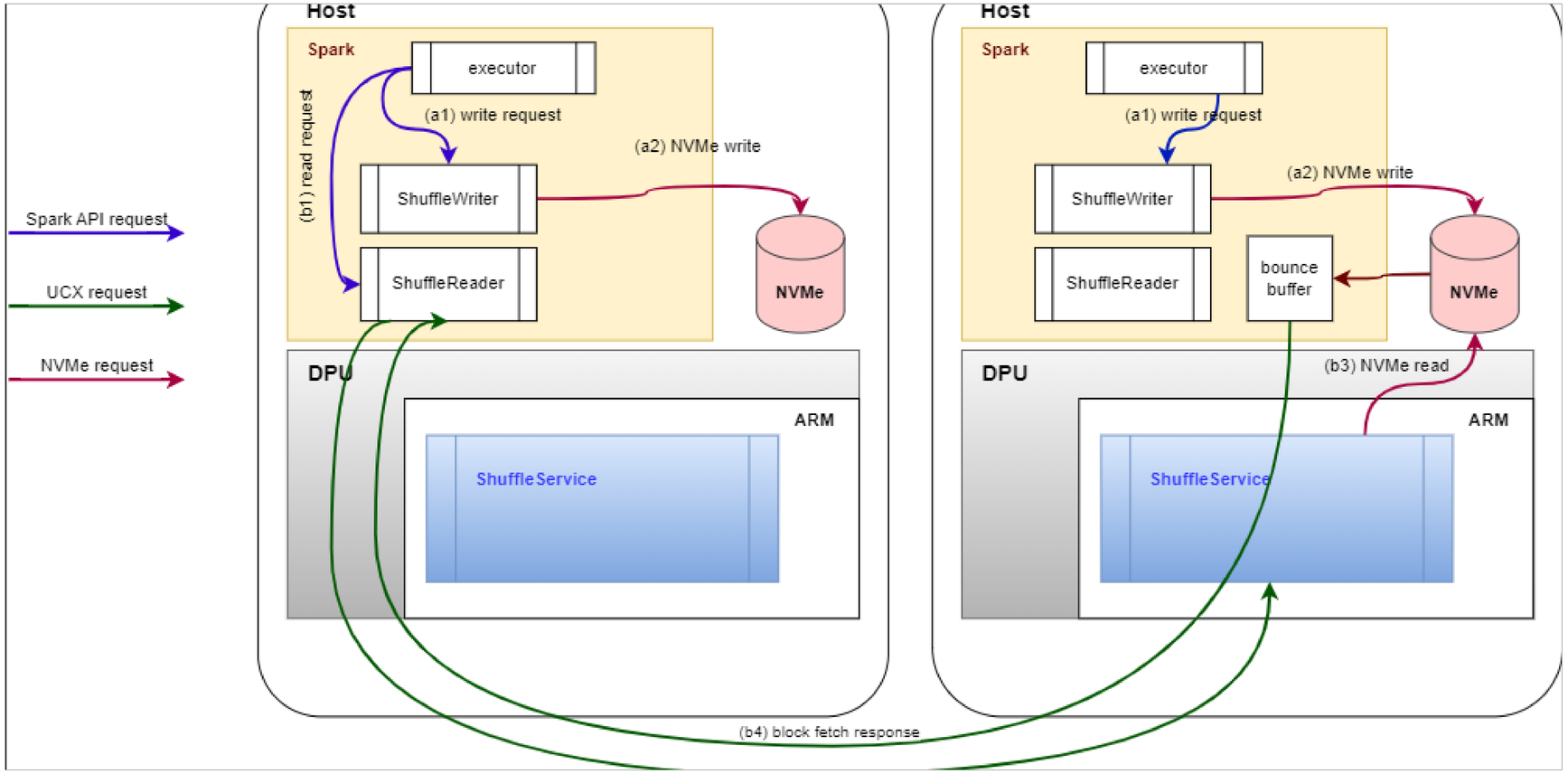
DPU

- nvkv context is received from the host:
  - Nvkv QP ctrl registers address+mkey
  - Huge pages address translation table (virtual to physical address)
  - Host BB memory information
- RDMA is used to control remote NVMe.
- Block addresses received from host are kept in memory (DPU) instead of index files.
- Host memory is used for reading blocks.
- Cross GVMI is used for sending blocks contents from the DPU to the host.



SparkDPU

Design Chart



Current Status

- Stable POC, supporting large scale
- Setup:
  - Cluster of 7 nodes.
  - Each node has a DPU, 2 NVMe and >200GB of memory.
- Benchmark: GroupByTest (Maximizing #Executors, running with 1 core and 10GB of memory each).
- Results

Links and appendices

MapReduce: Simplified Data Processing on Large Clusters

https://github.com/openucx/sparkucx

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

https://github.com/NVIDIA/sparkucx