Near-Data Computing Model for Accelerating LHC Data Filtering

Narangerelt Batsoyol, Jonathan Guiang, Philip Chang,
Diego Davila, Aashay Arora, Frank Wuerthwein,
Steven Swanson

University of California, San Diego

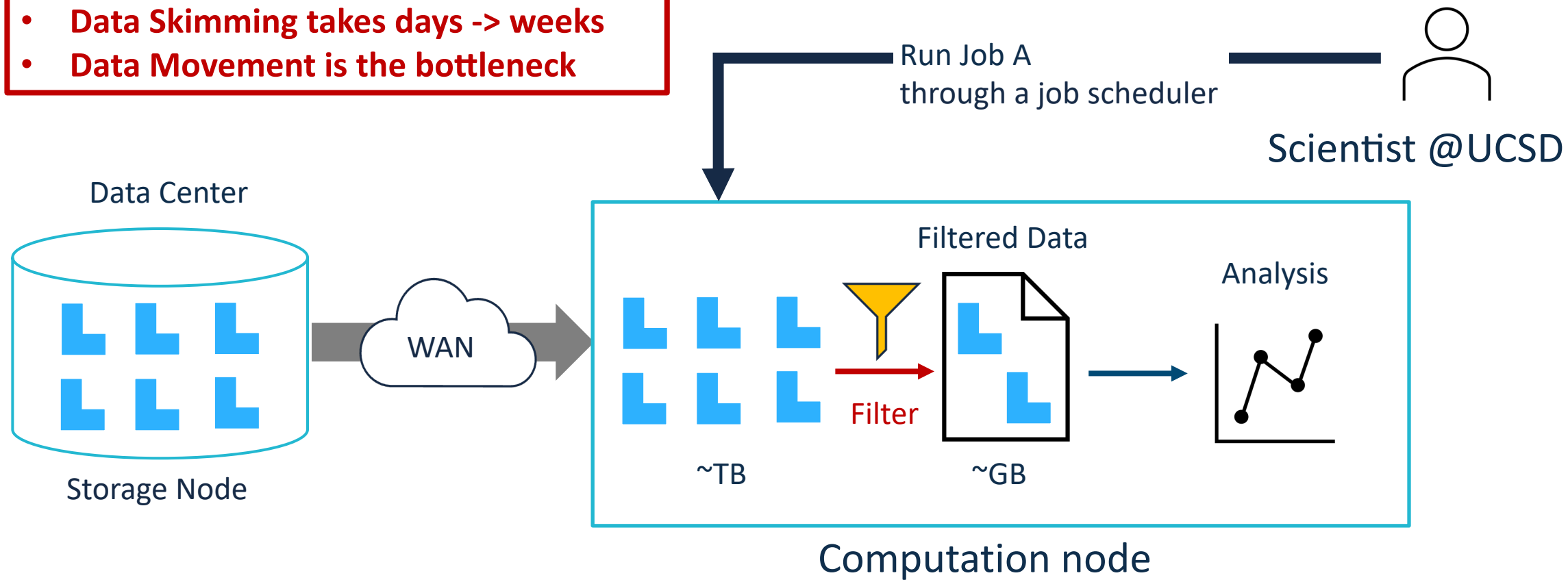
University of Florida

NVSL

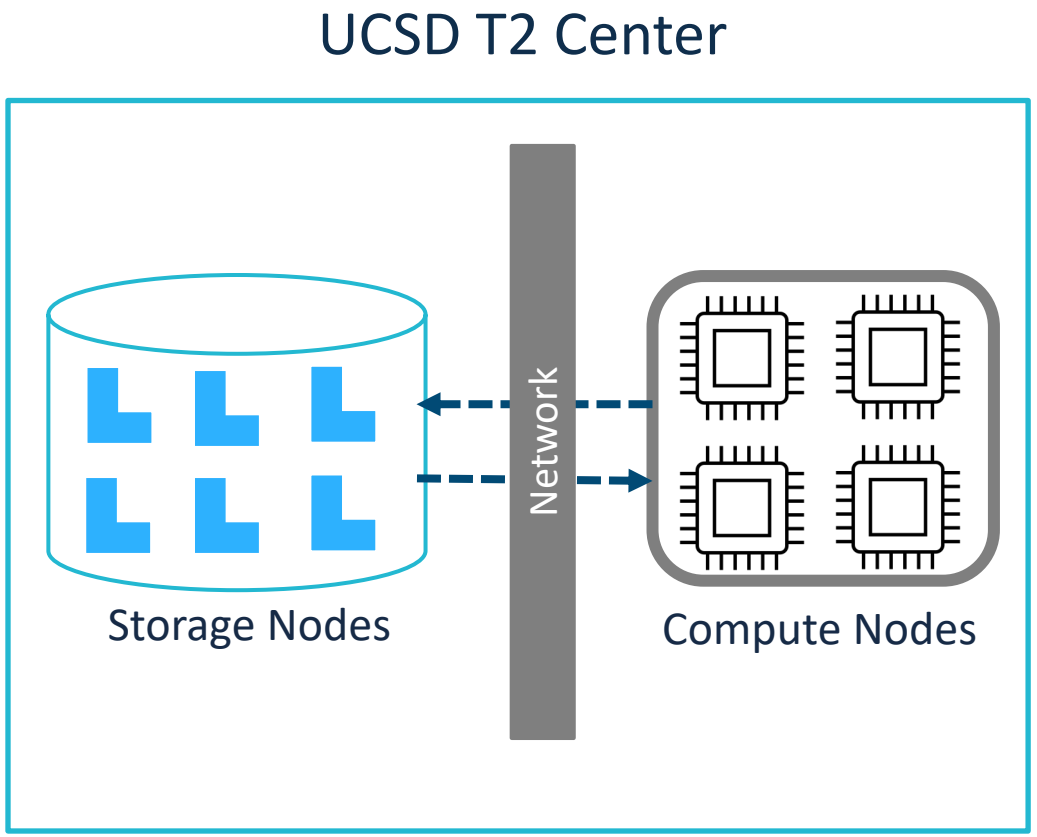
UCSDCSE Computer Science and Engineering

Every data analysis in LHC starts with Data Skimming

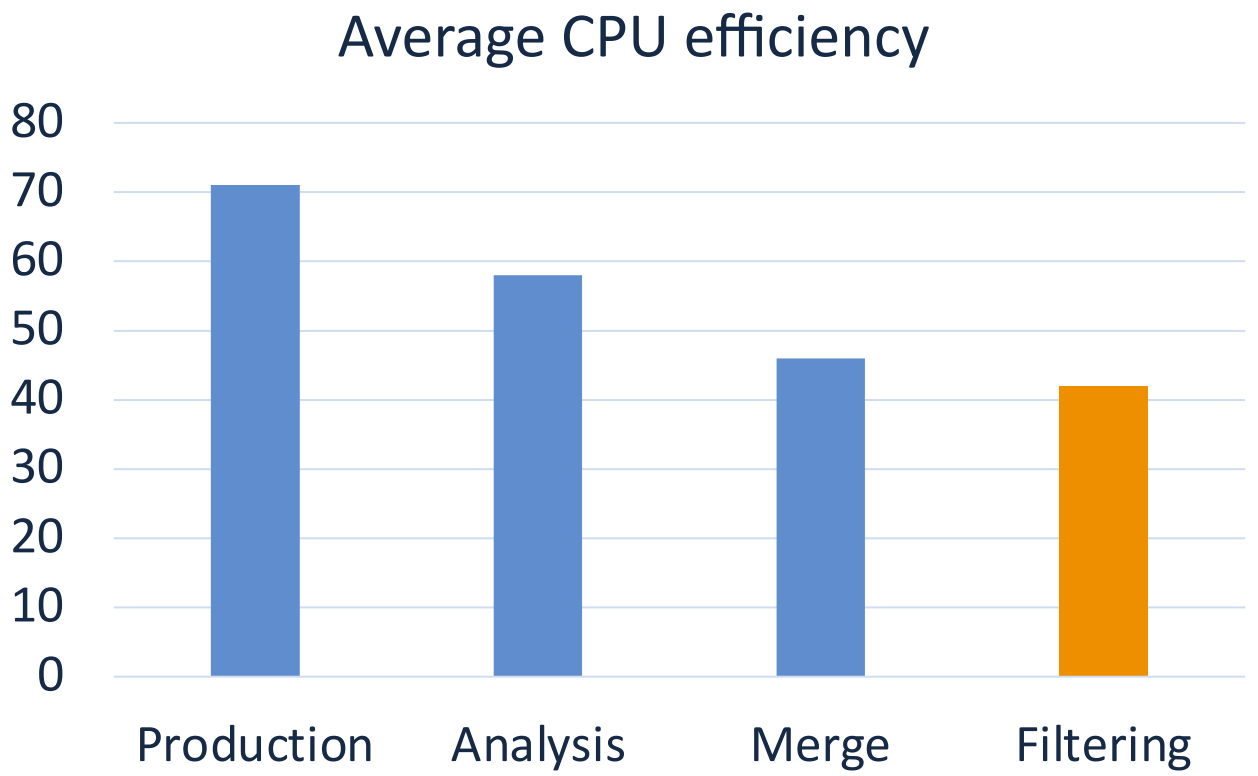
- **Data Skimming takes days -> weeks**
- **Data Movement is the bottleneck**



CPU efficiency in a local cluster

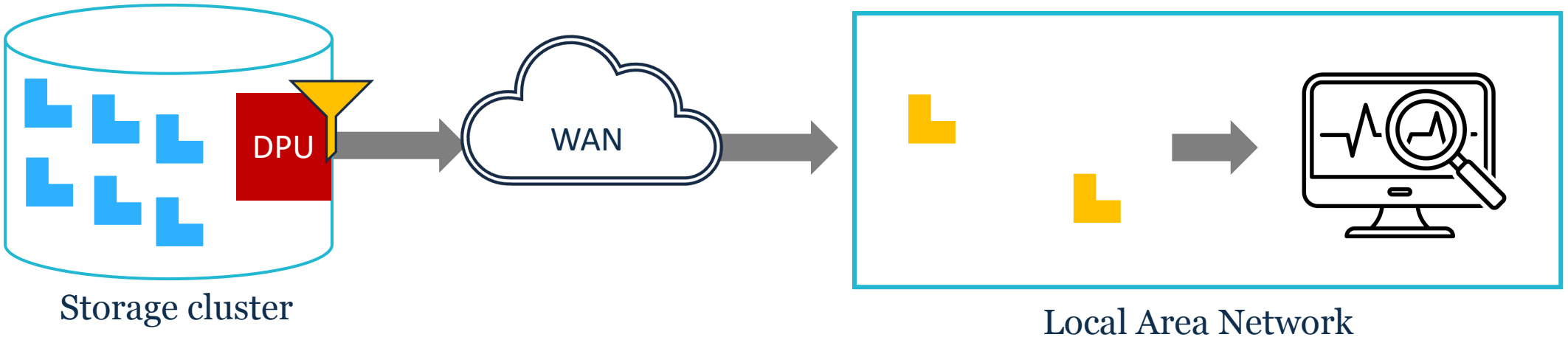


*Data in the local cluster



K. Ellis, C. Brew, G. Patargias, T. Adye, R. Appleyard, A. Dewhurst, and I. Johnson, "Xrootd and object store: A new paradigm," in EPJ Web of Conferences, vol. 245. EDP Sciences, 2020

DPU-assisted Near-Data Filtering



Data Processing Unit (DPU)

Server Networking Evolution

Legacy NIC

- Sends and Receives data

SmartNIC

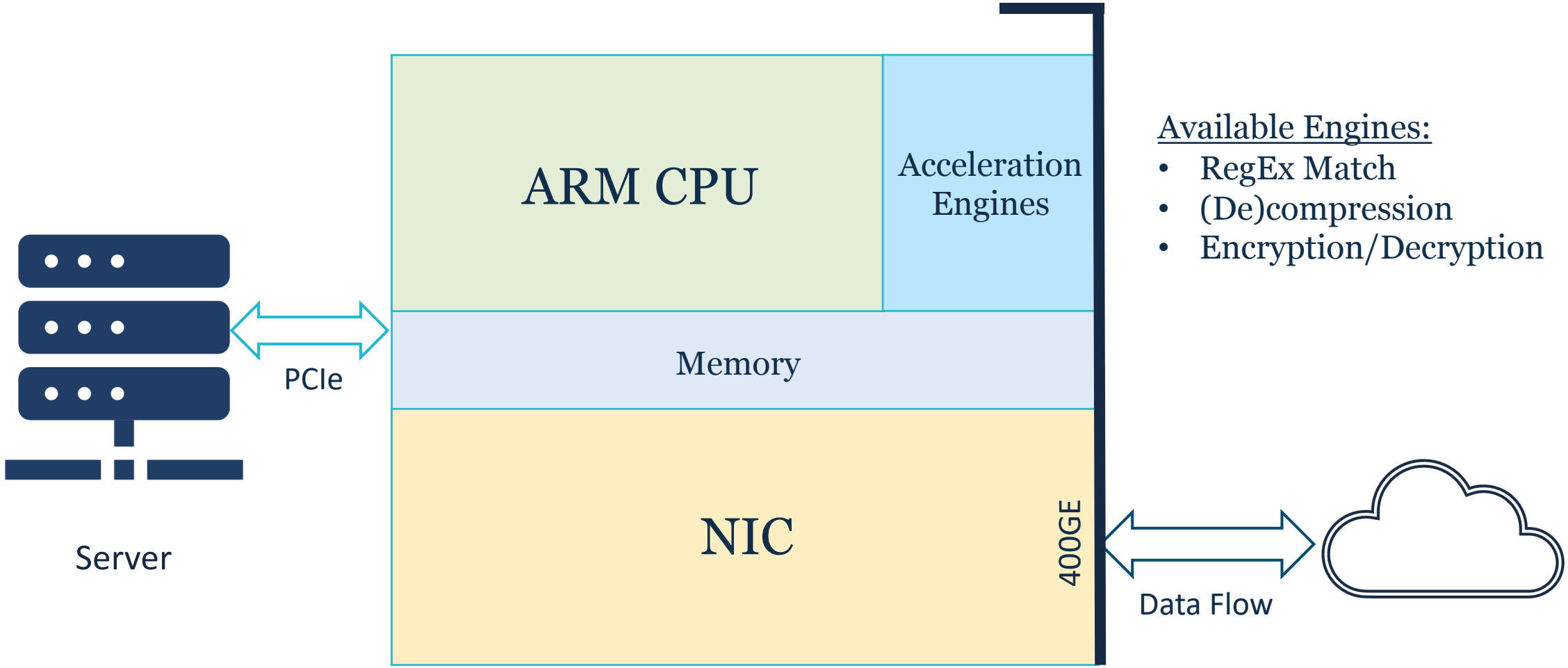
+ Limited processing capabilities
- Network related tasks

DPU

+ Full computation capabilities
- Independent server
- Networking, Security, Storage and System management tasks

NVSL

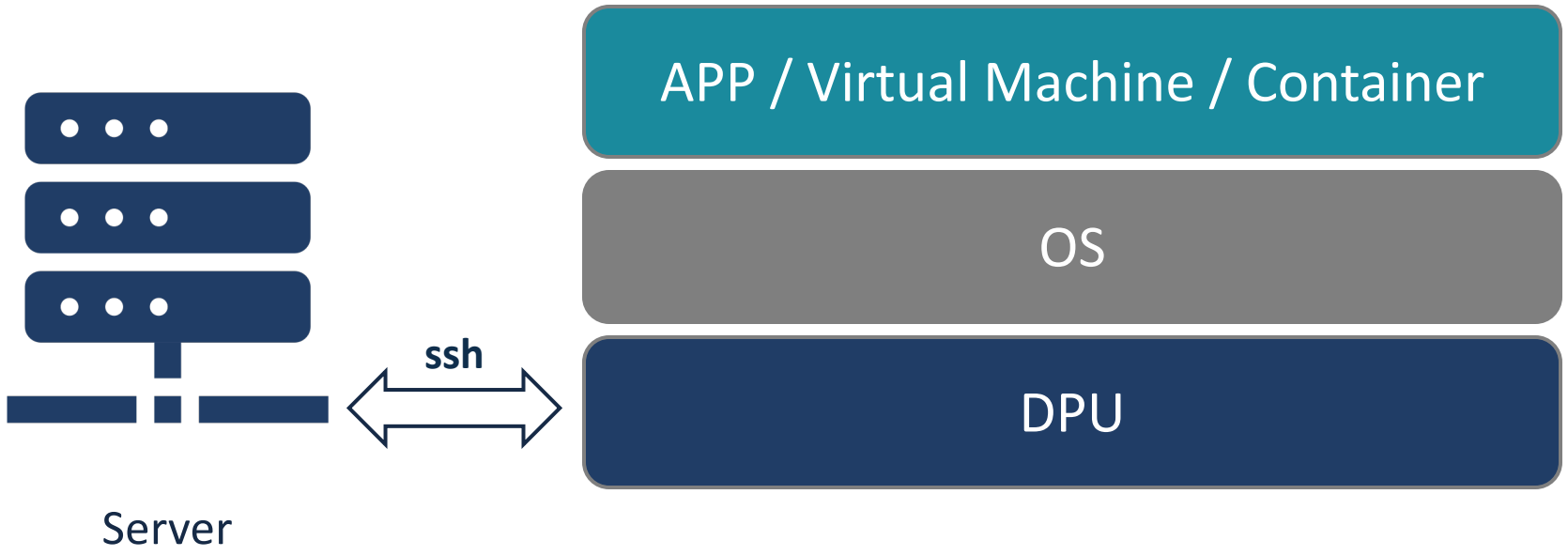
Data Processing Unit (DPU)



Available Engines:

- RegEx Match
- (De)compression
- Encryption/Decryption

Running APP on DPU



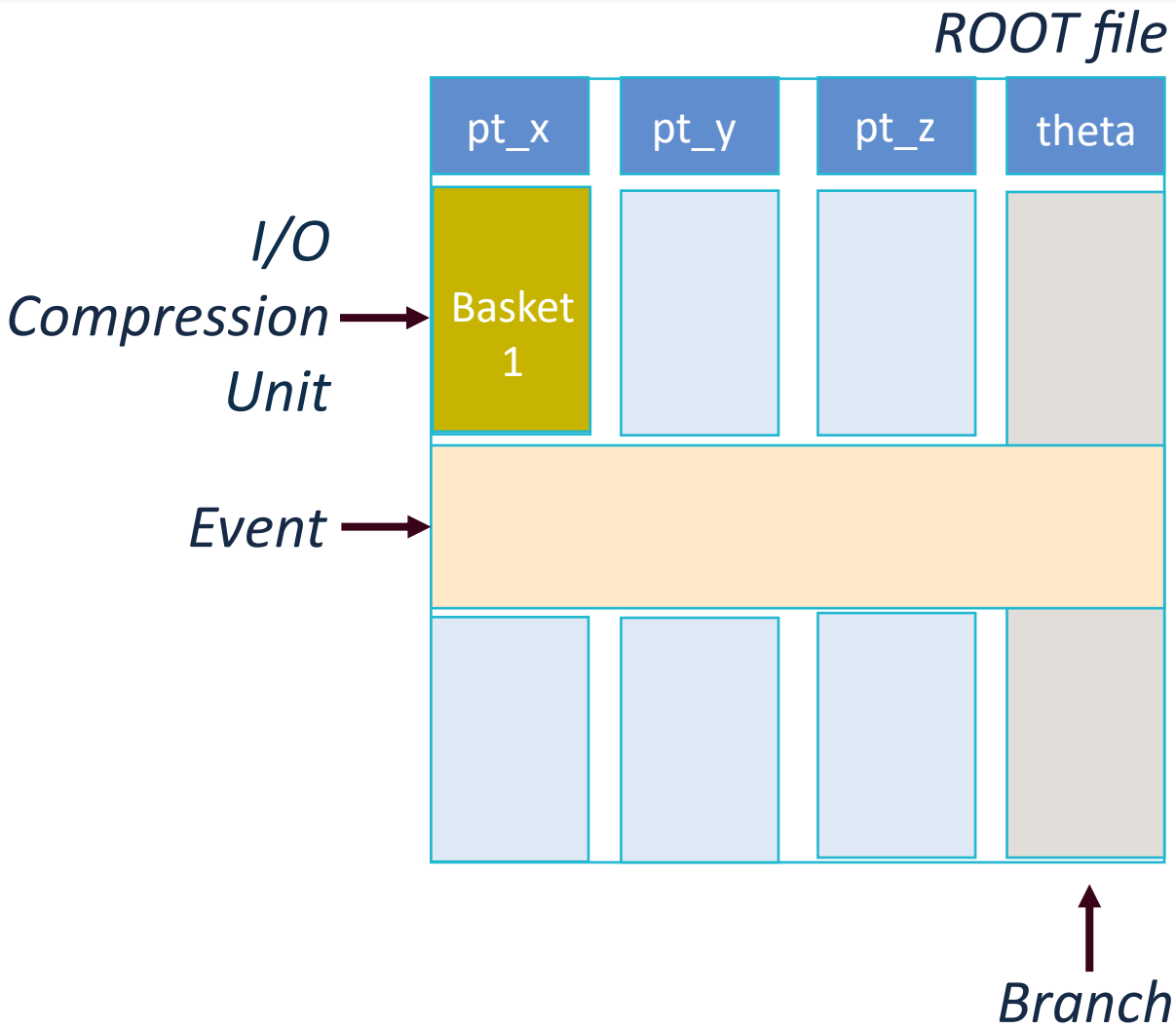
Accelerating LHC Data Filtering with a Near-Data Computing Model

- Use power-efficient DPUs to filter data near-storage
- Optimize the entire system without changing the existing storage infrastructure
  - Code Optimization (data access)
- Streamline the workflow by removing coding complexities, making programming easier for end users
  - 700 lines of code

NVSL

Original Skimming Process

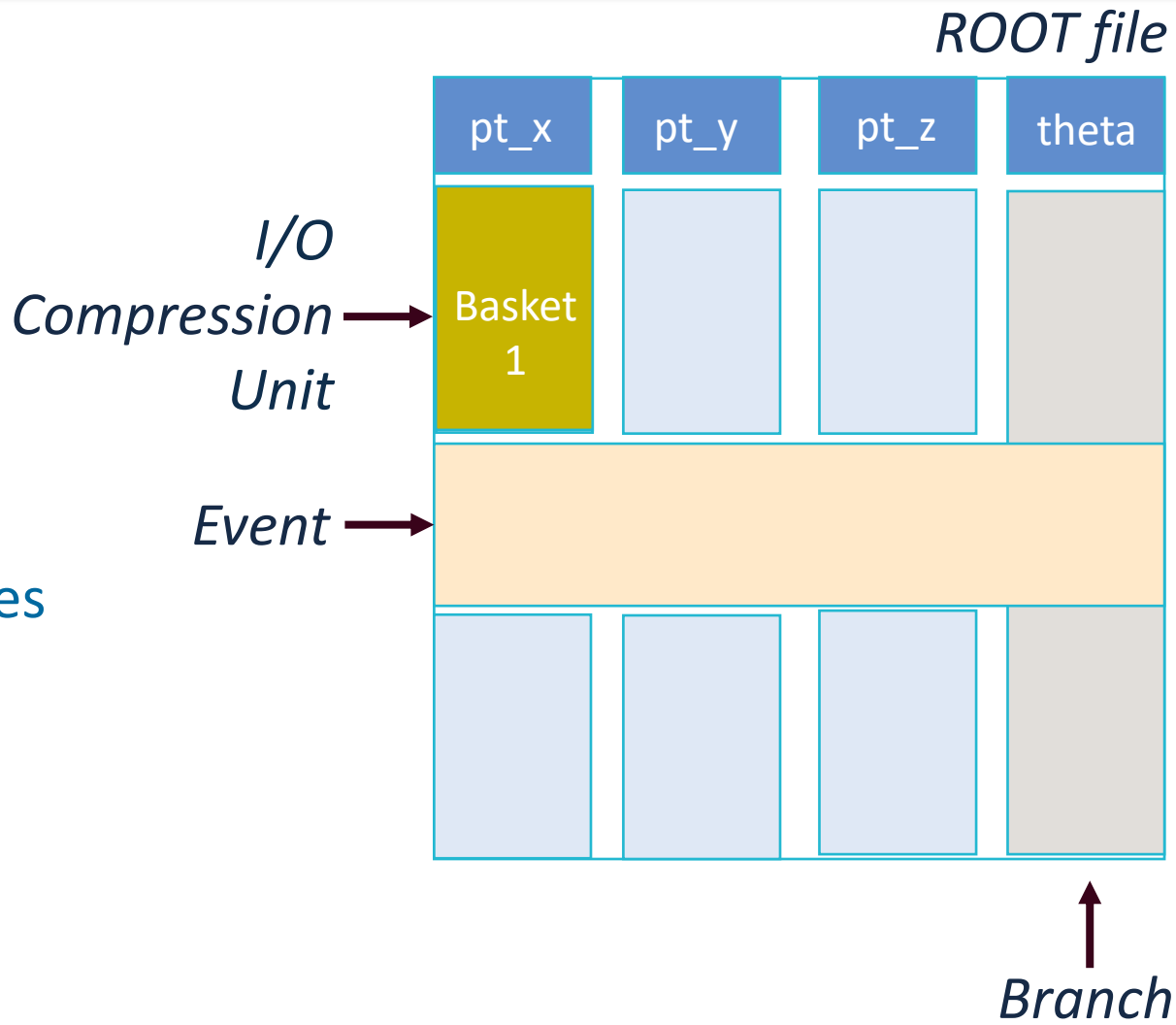
- Skimming consists of:
  1. Fetch basket
  2. Decompress
  3. Deserialize



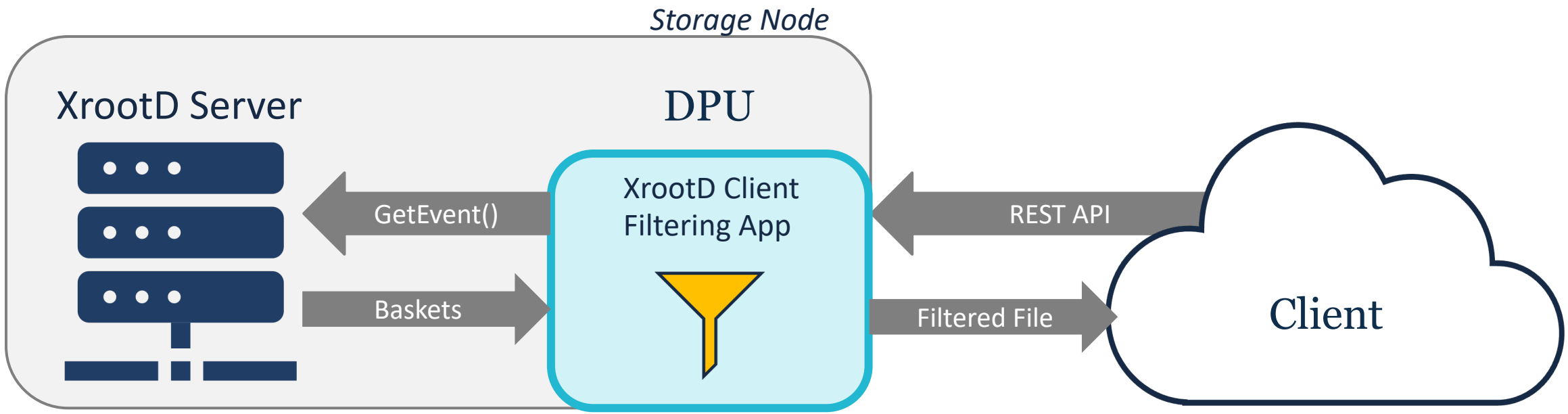
Optimized Skimming

No DPU involvement

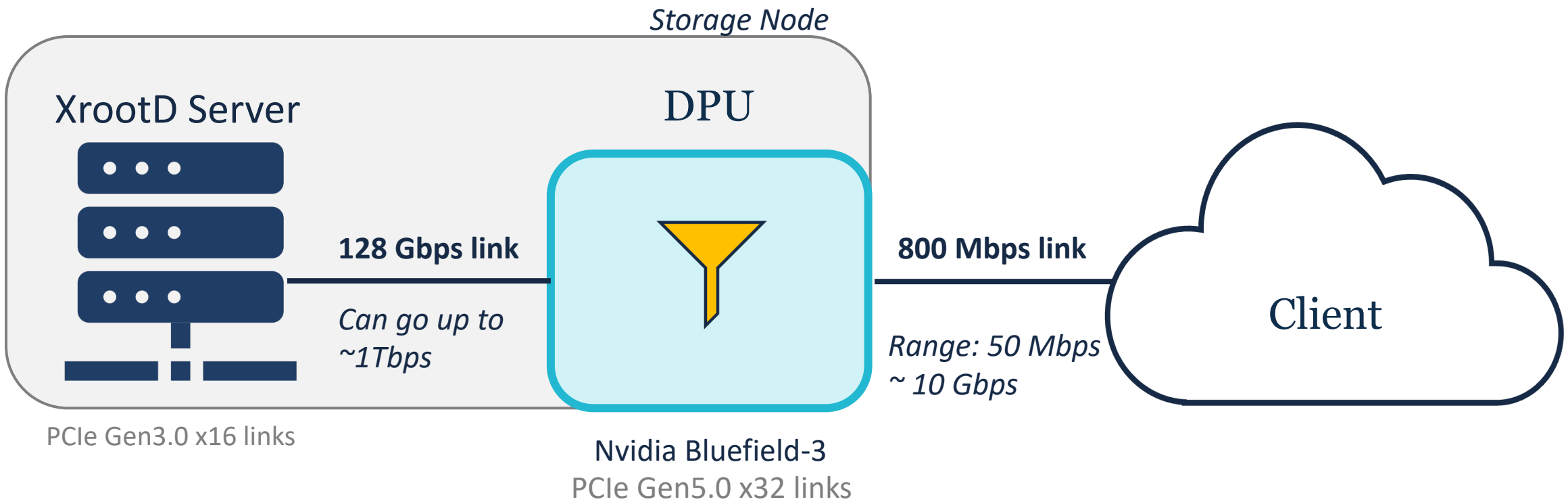
- Selective Branch Loading
  - Minimize unnecessary processing (reading and decompressing).
  - First, check if the event matches selection criteria before processing baskets from other branches.



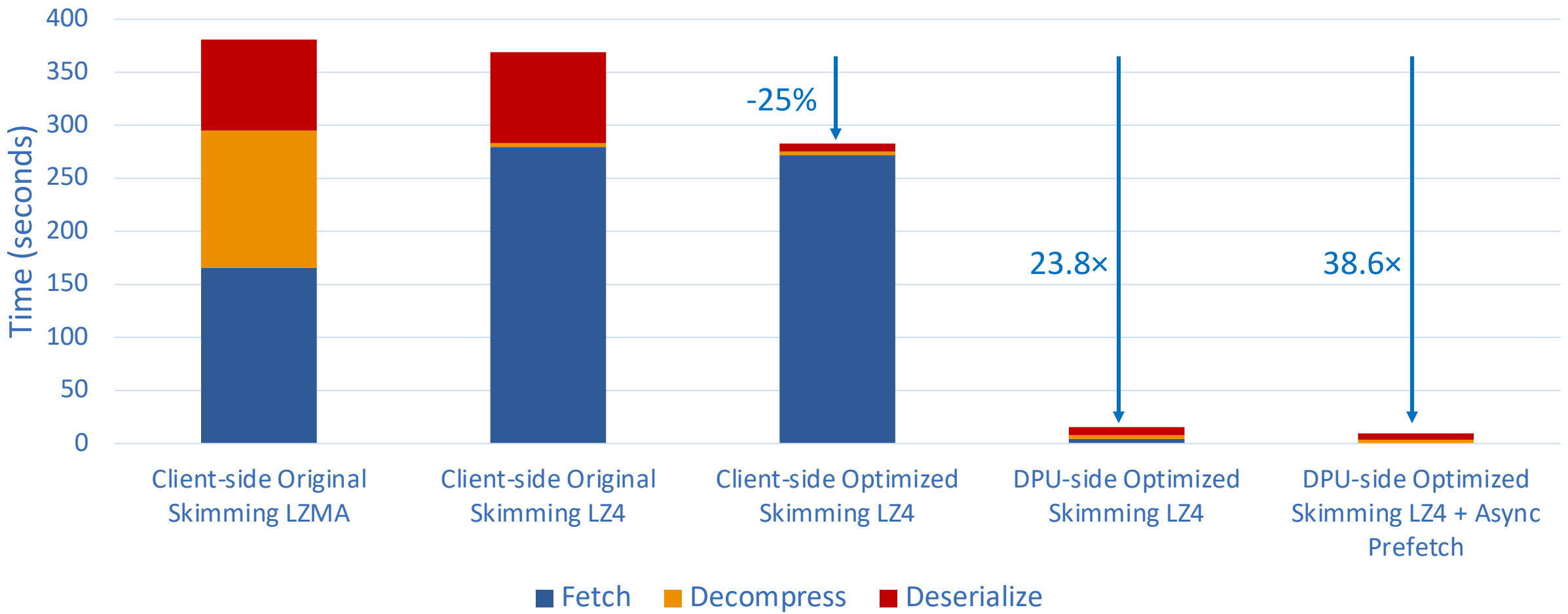
System Setup
DPU as an independent server with its own IP



Evaluation Setup



End-to-End Execution Time



Future Work and Limitation

- Use DPU's decompression acceleration engine
- DPU has limited compute and memory resources
  - 32 GB Memory
  - 16 ARM cores
- Solve the problem of scalability and concurrent requests
- Test on RDataFrame
- Optimize the data layout for network packet-wise filtering instead of entire files.

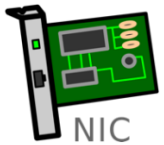
NVSL

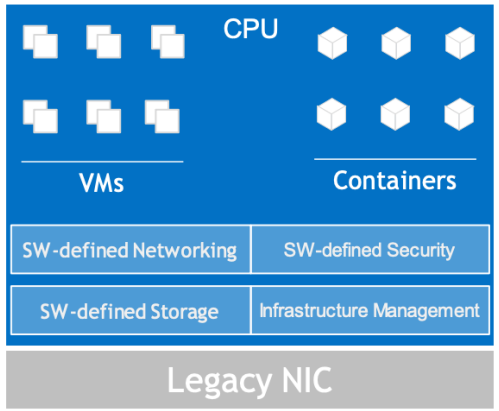
Thanks!

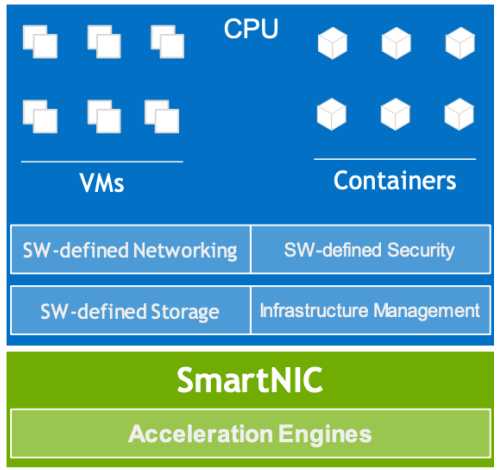
Background Slides

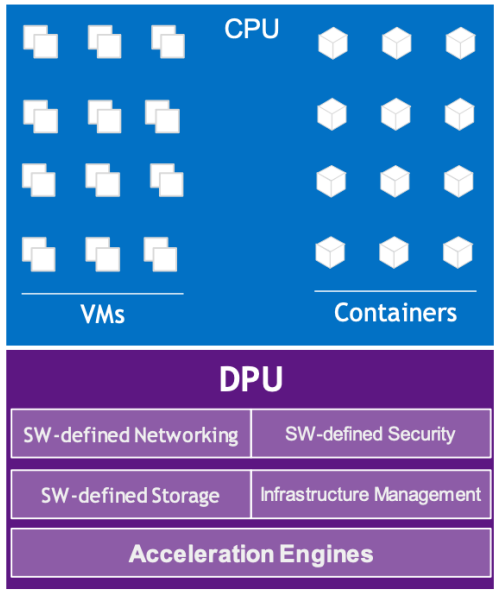
Server Networking Evolution

Software Defined Data Center Infrastructure-on-a-Chip









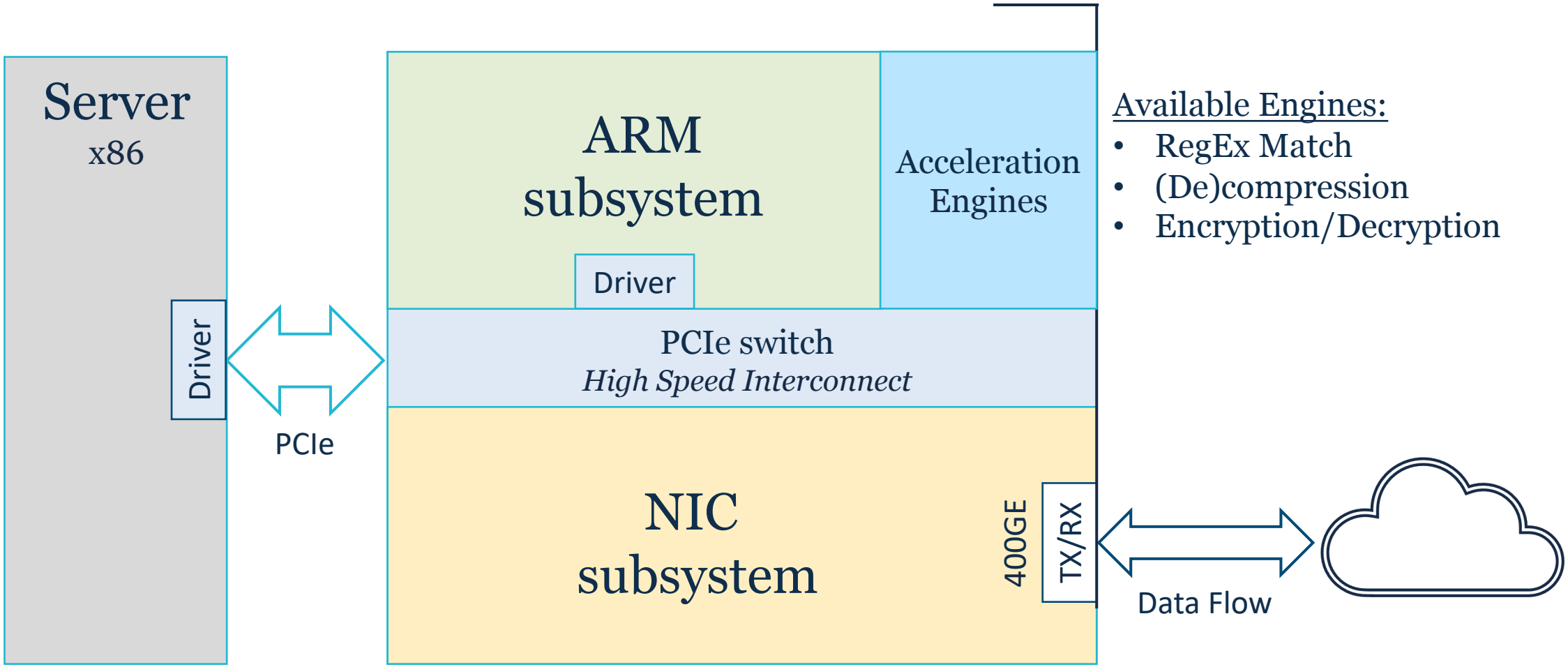
Legacy Infrastructure

Accelerated SW-defined Infrastructure on CPU

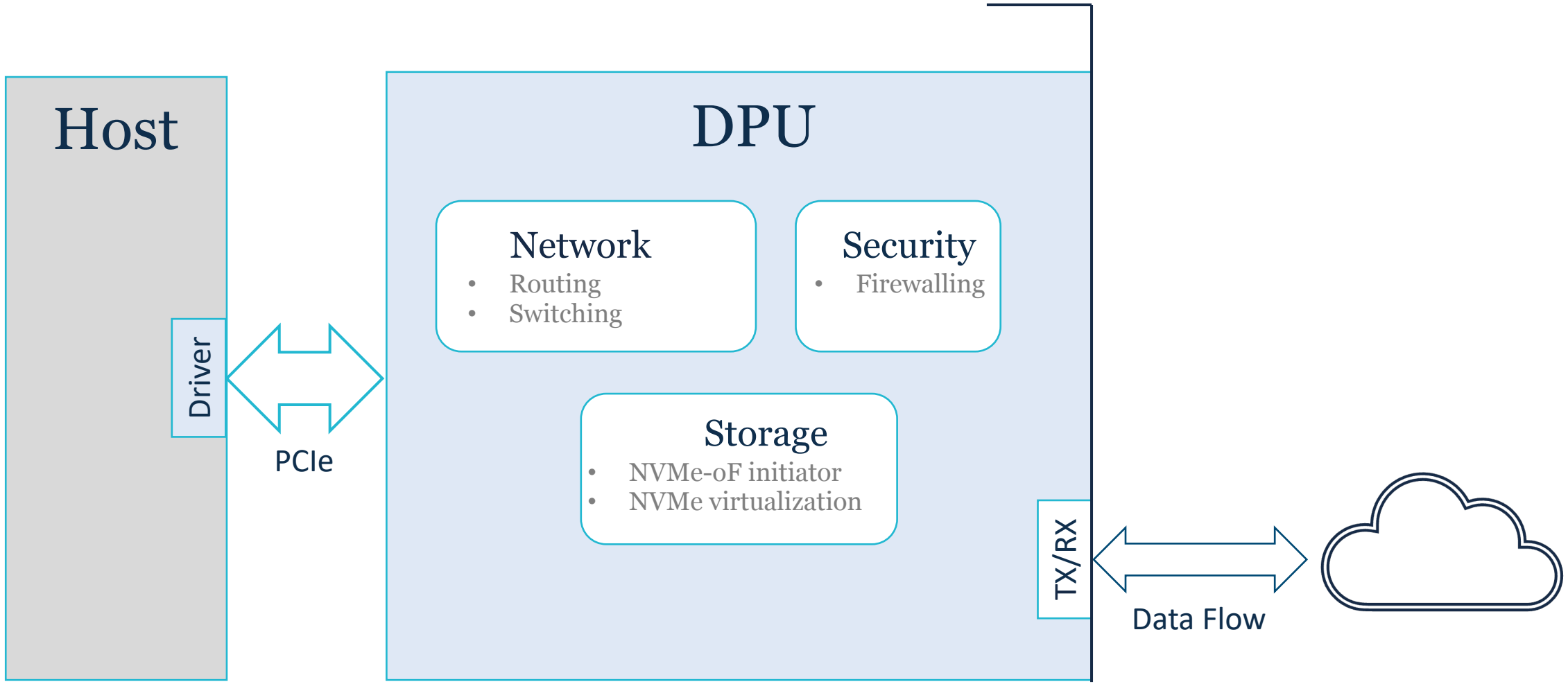
SW-defined Infrastructure on DPU

NVSL

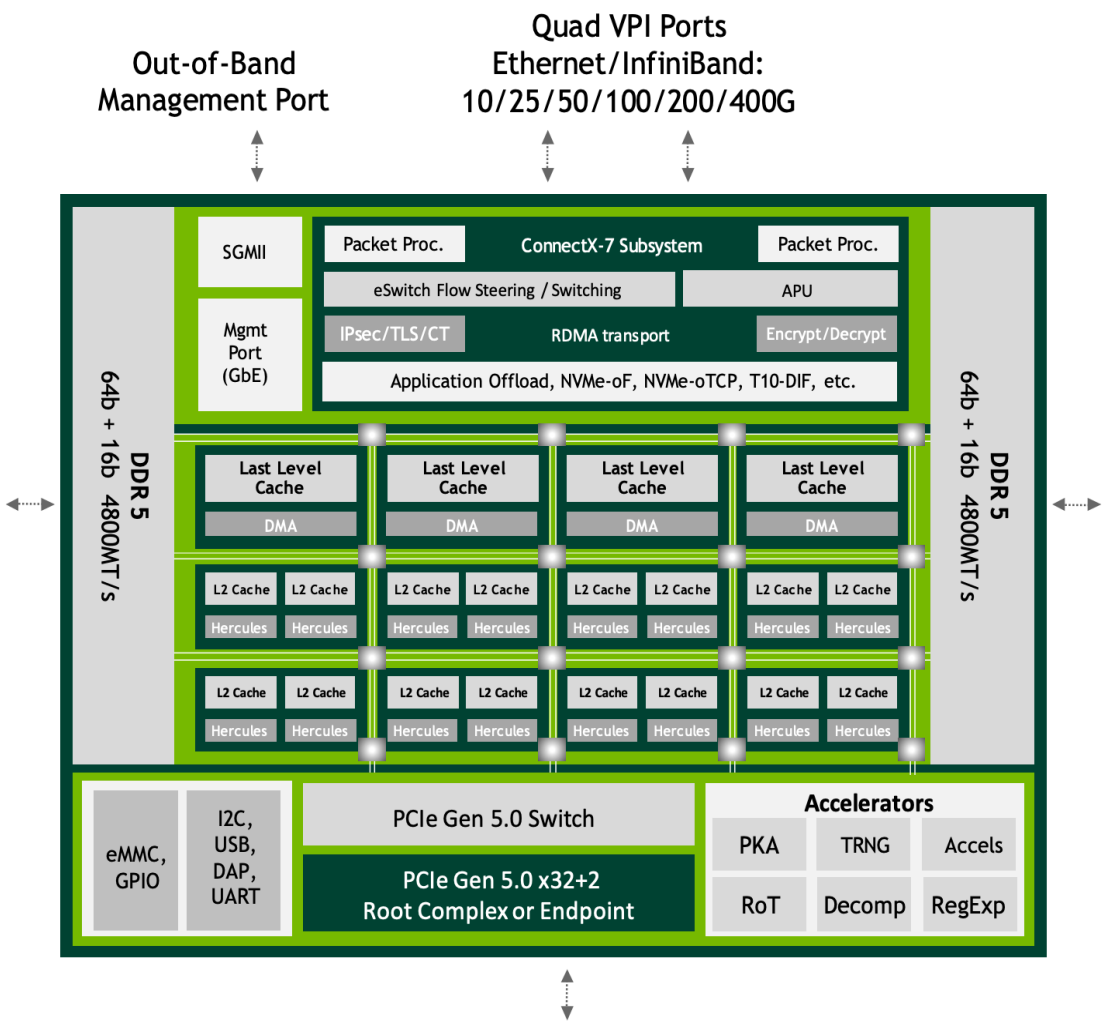
Data Processing Unit (DPU)



DPU


Host
Driver
PCIe
DPU
Network
Routing
Switching
Security
Firewalling
Storage
NVMe-oF initiator
NVMe virtualization
TX/RX
Data Flow

Bluefield-3 DPU from Nvidia

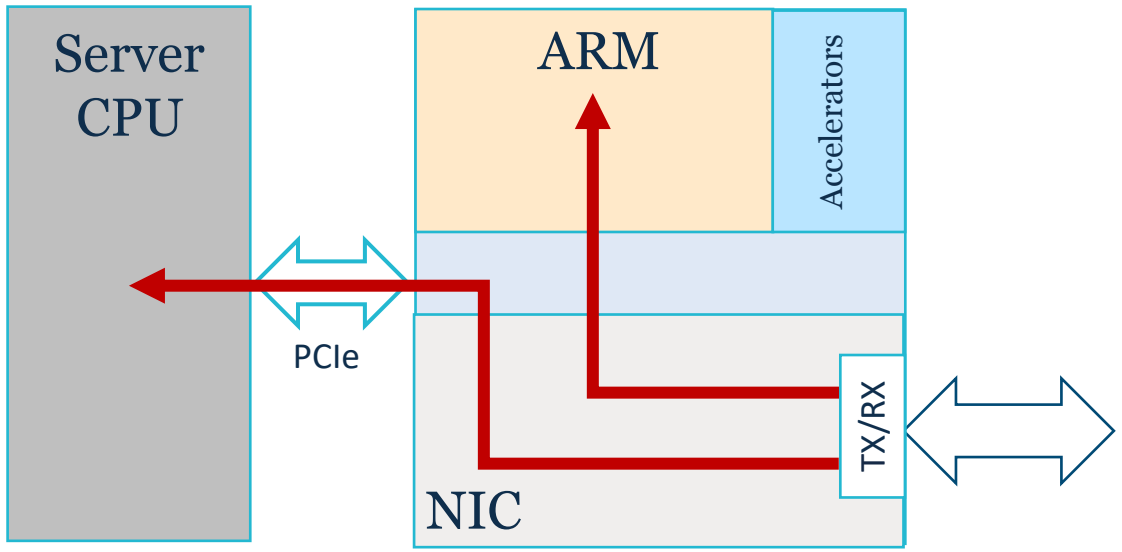


- ConnectX-7
- I/O
  - 2 Ethernet/Infiniband ports up to 400Gb/sec
  - PCIe Gen5 x 32
- 16 ARM Cortex-A78 cores
- 32GB

NVSL

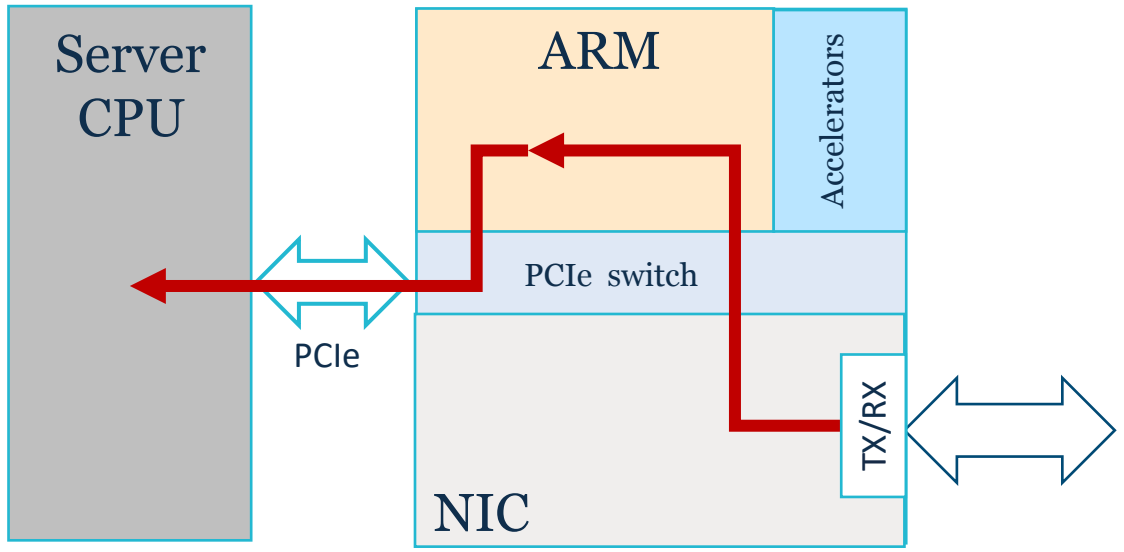
DPU Modes of Operation

Off-path Mode



- Off-path SmartNIC
- DPU act like an independent server
- Own IP and MAC address

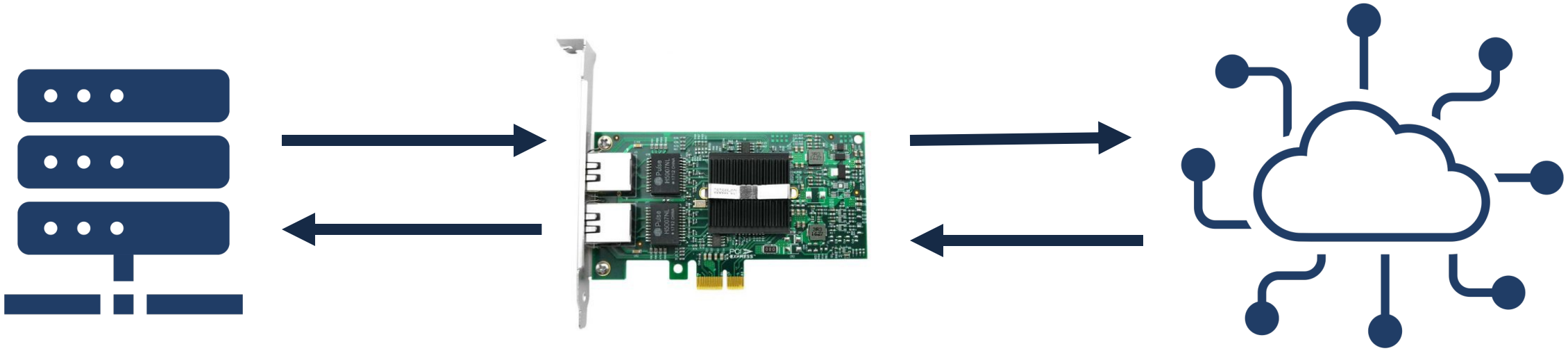
On-path Mode



- On-path SmartNIC
- All network packets go through ARM cores
- Network packet manipulation

NVSL

Server Networking Evolution

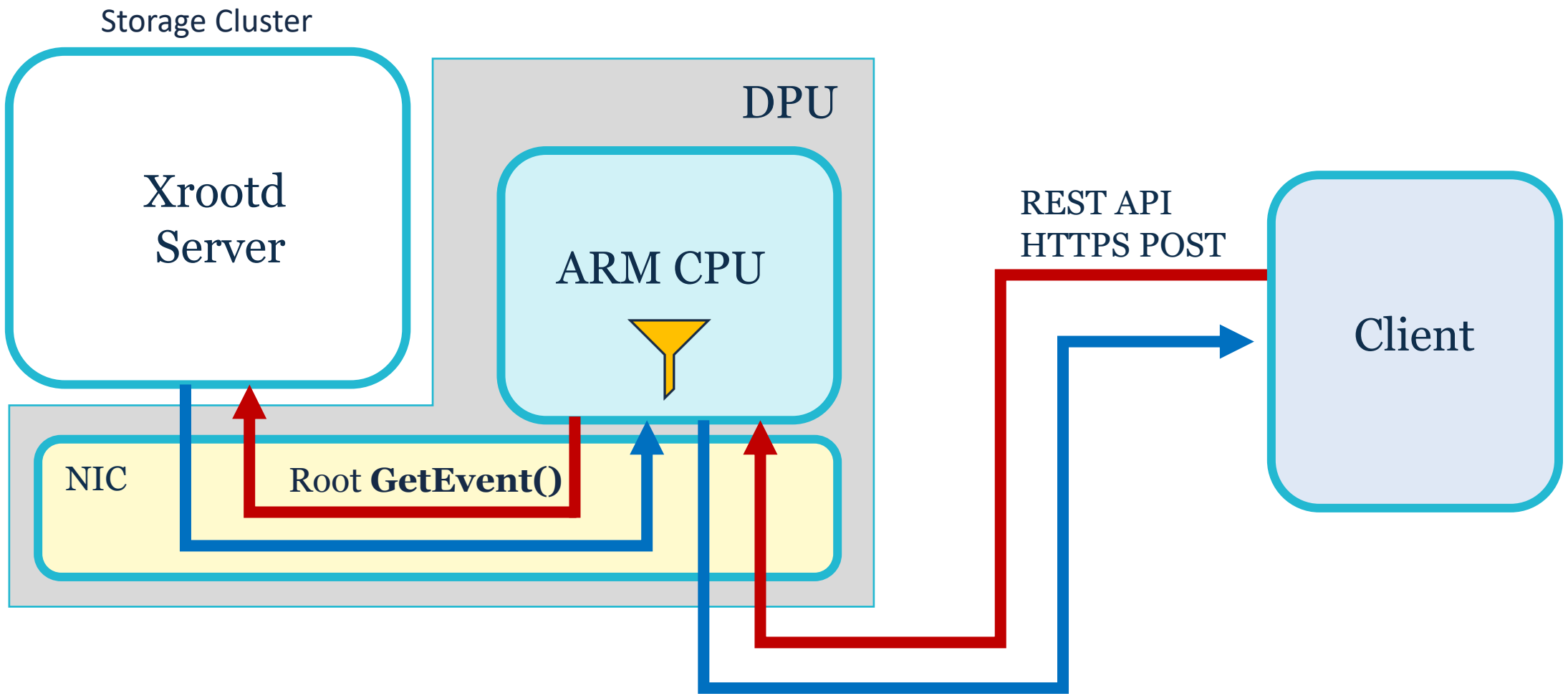


Server

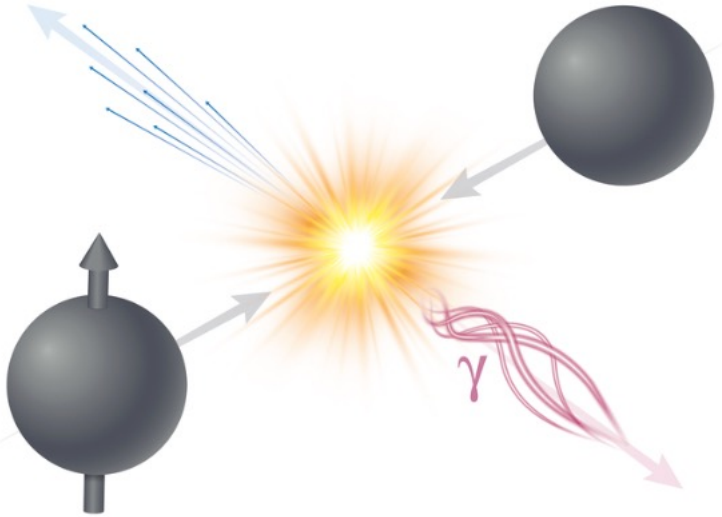
Network Interface Card (NIC)

Ethernet

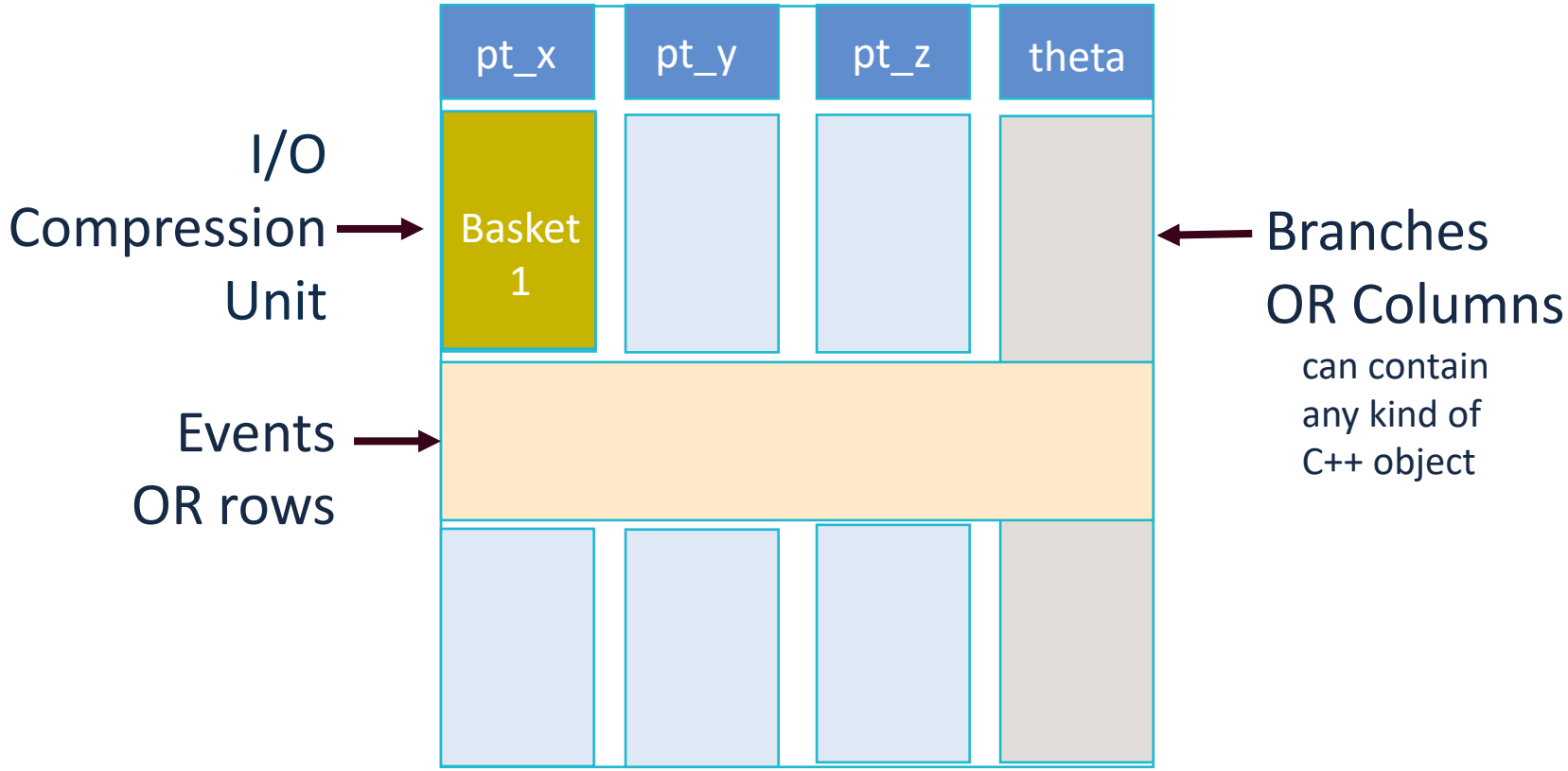
System Setup
DPU as an independent server with own IP



Data Format: ROOT file
In-memory Data Structure: TTree

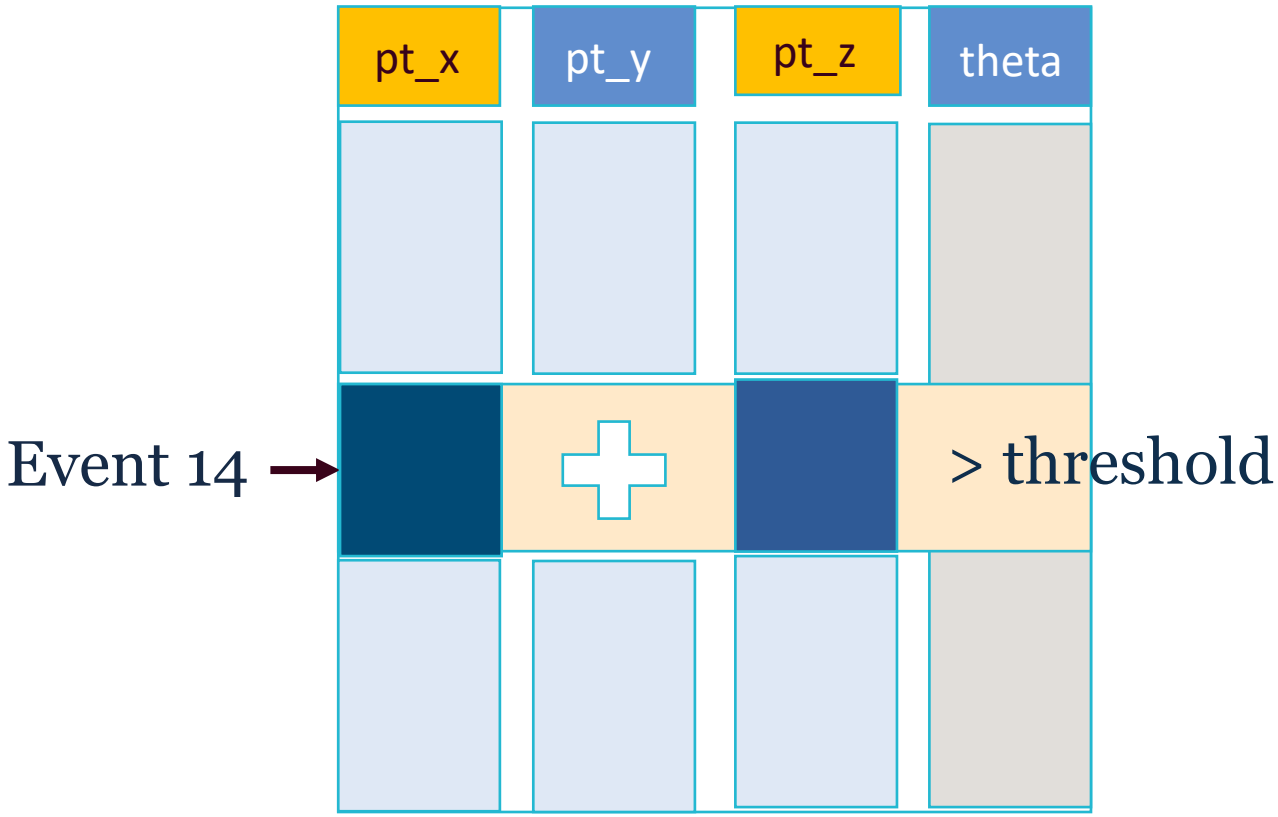


Single Event



- 1M events in a single file
- 10GB uncompressed

Filtering ROOT file



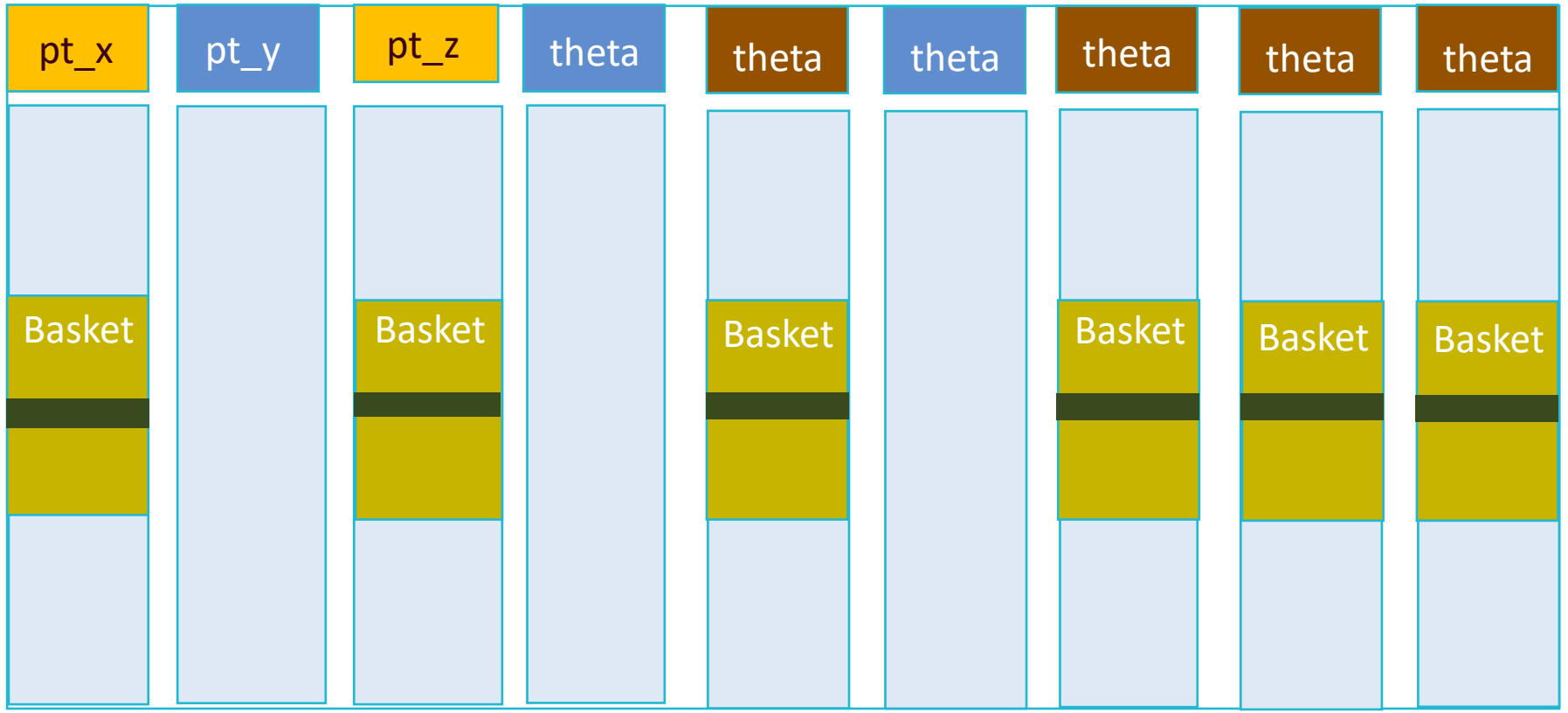
Write Event 14 to Output tree

- Retain < 1% of rows (3-10MB)

Original Data Filtering Code

GetEvent (14) →

- Load Baskets into DRAM
- Decompress Baskets
- Deserialize column entry



Selection criteria

Feature selection

NYSL

Optimized Data Filtering Code:
Selective Branch Loading