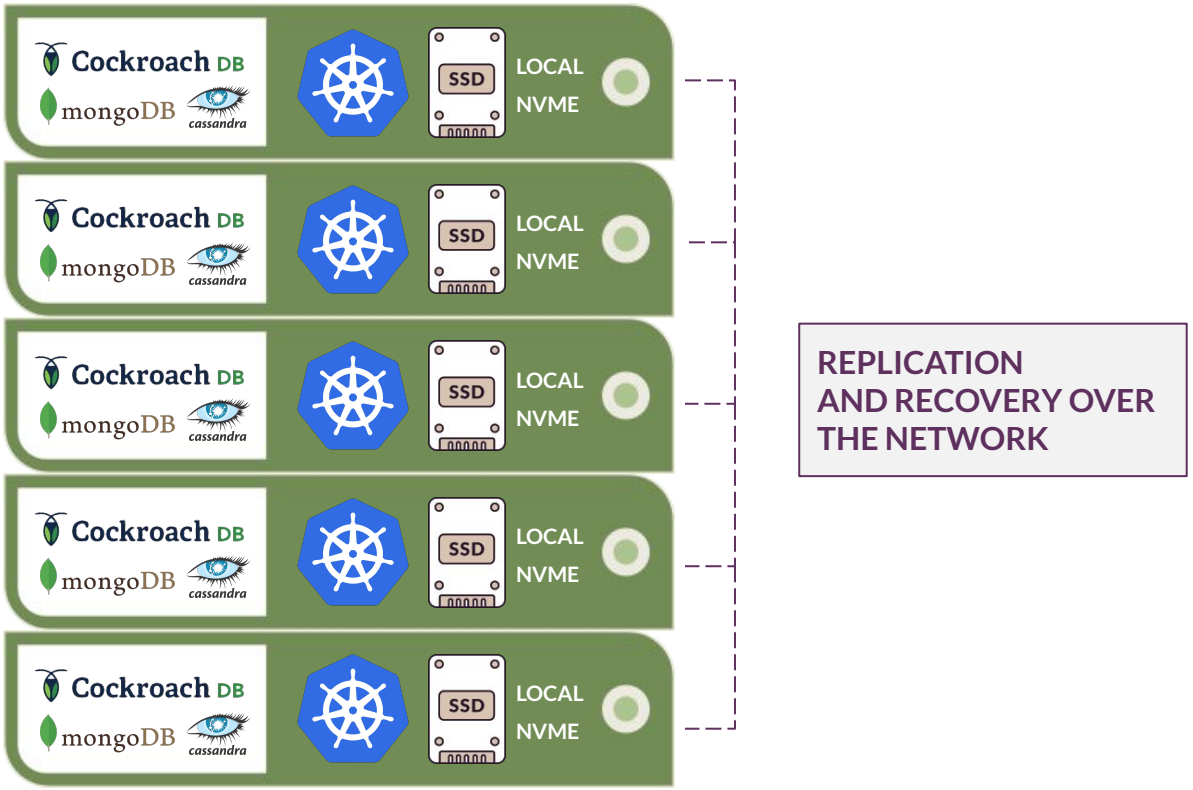
lightbits

Data Access at the Speed of Light(bits) in an Open Programmable Infrastructure World

Cloud-Native Stateful Applications on K8s with Local SSDs

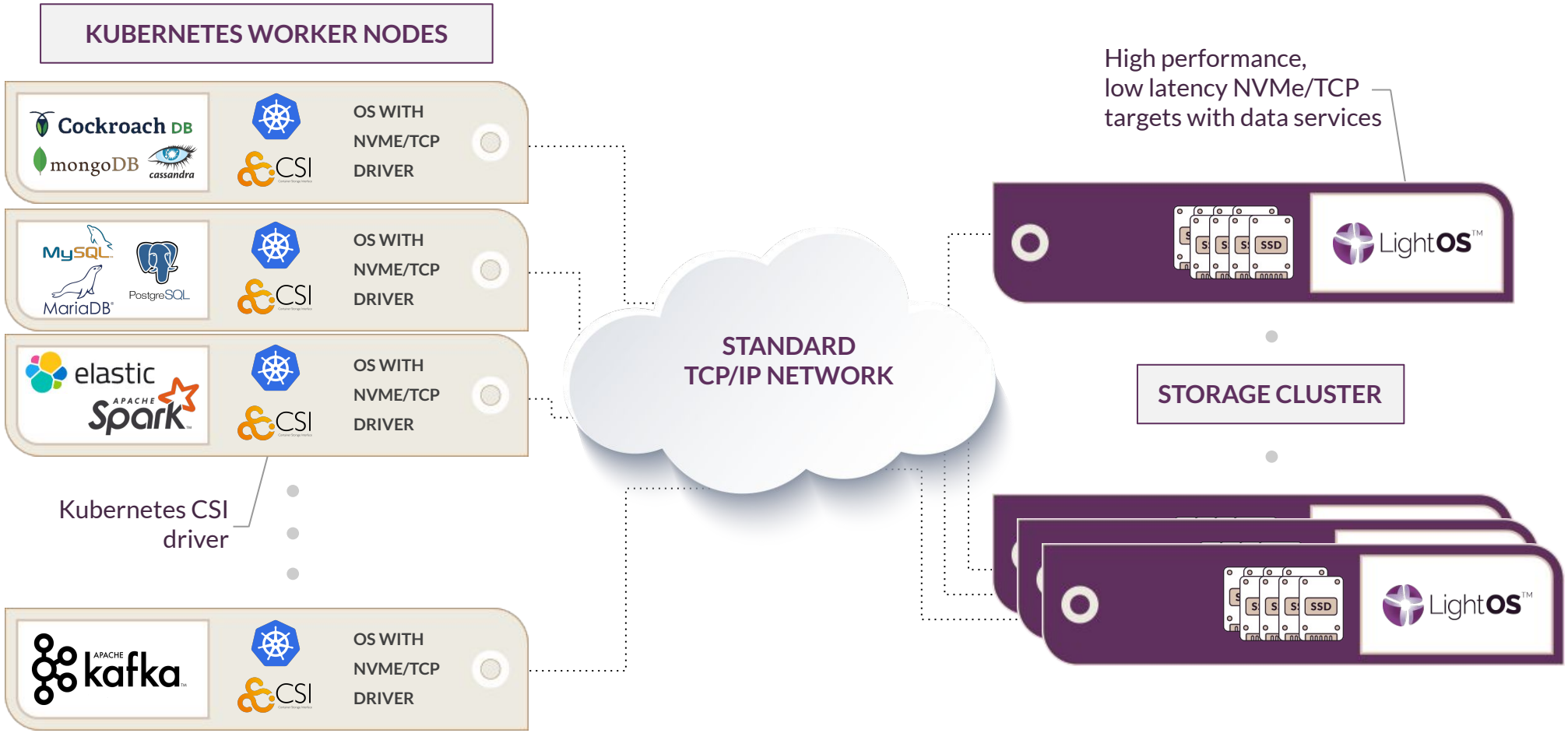


Kubernetes Local Persistent Volumes on NVMe

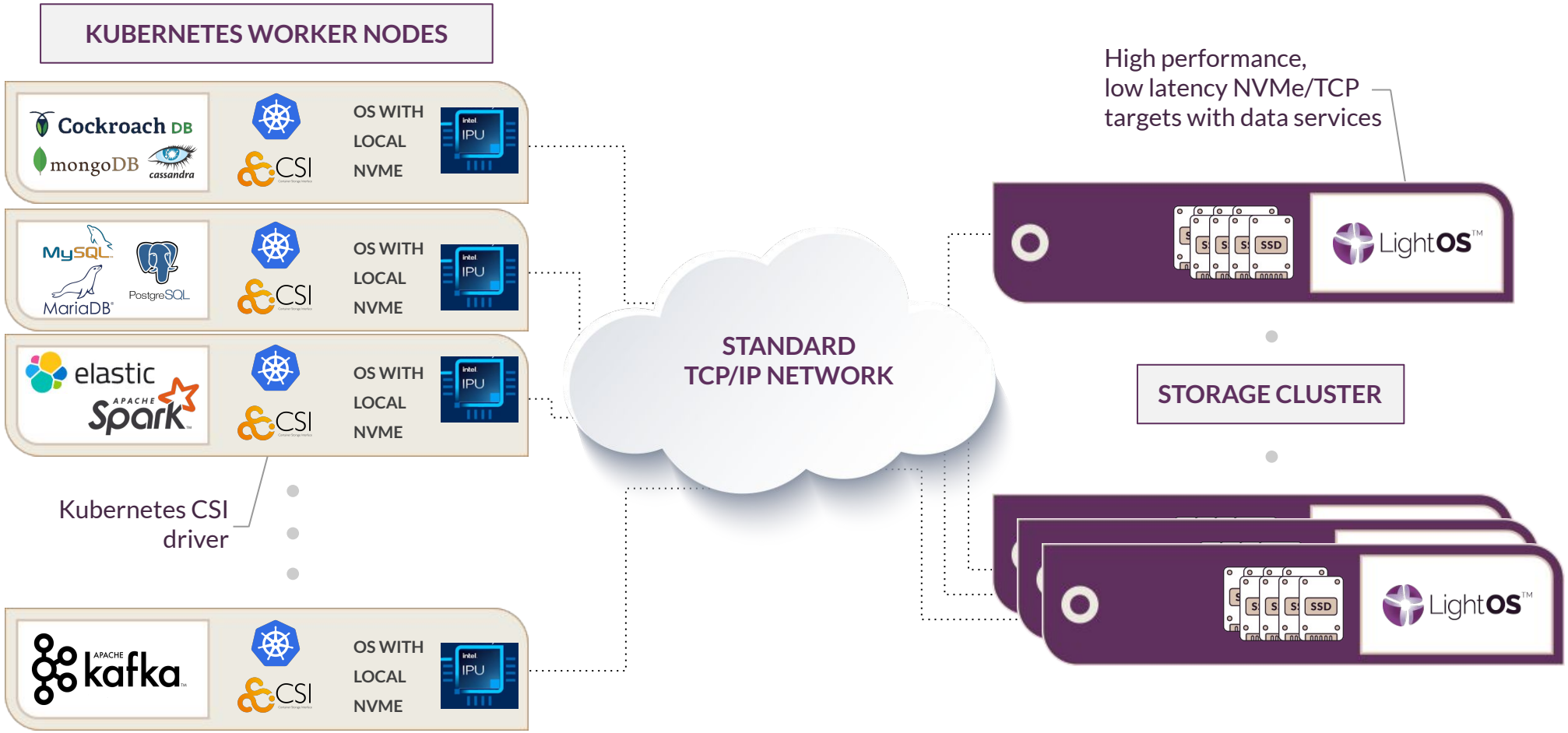
You get local flash performance but:

- Breaks the application and K8s philosophy of portability
- Data and pods assigned to one physical server
- Pod movement is limited or not allowed
- If only some k8s servers have local flash it limits service portability
- If all k8s servers have local flash it results in poor utilization

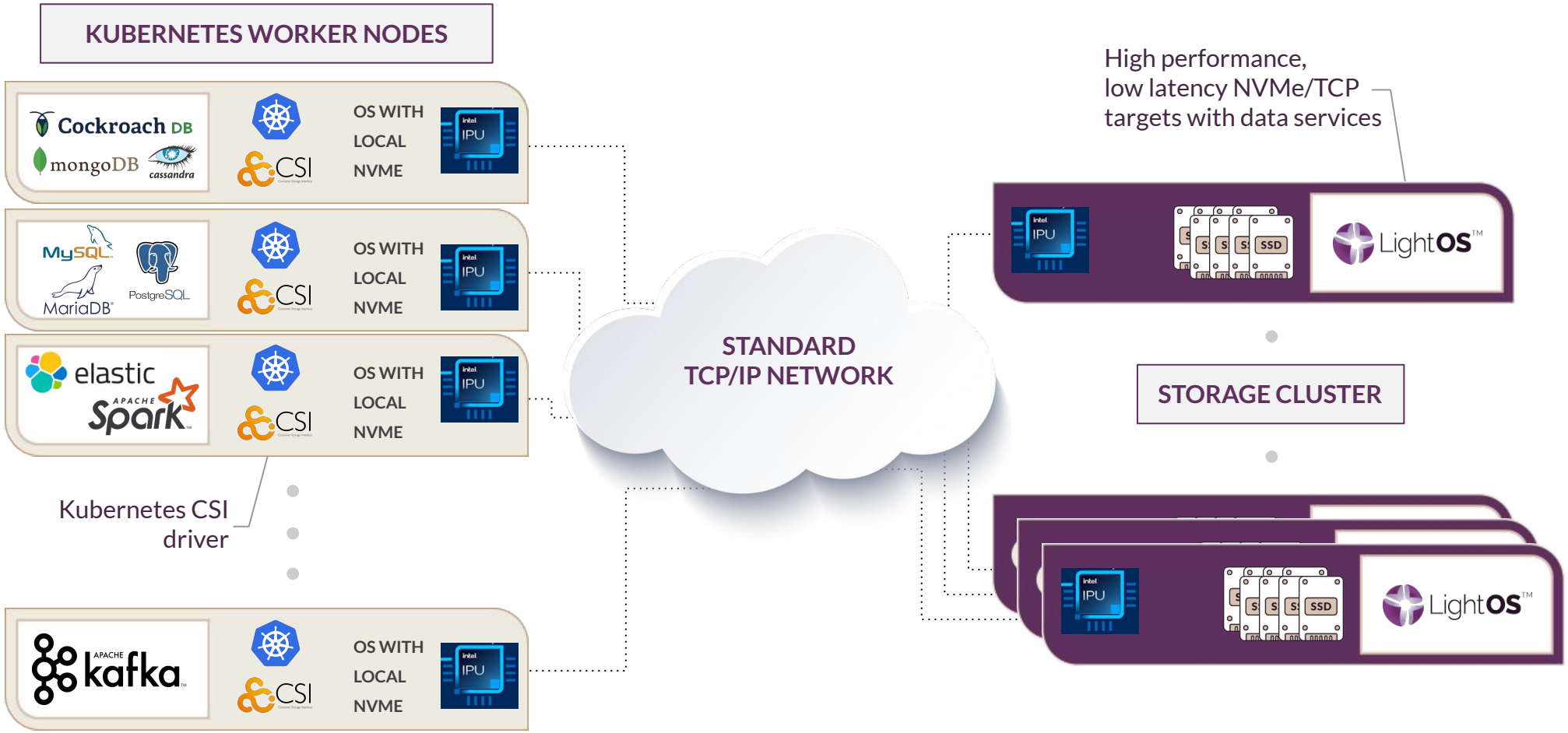
High Performance Software Defined Storage



Things We Will Talk About: IPUs on the Clients

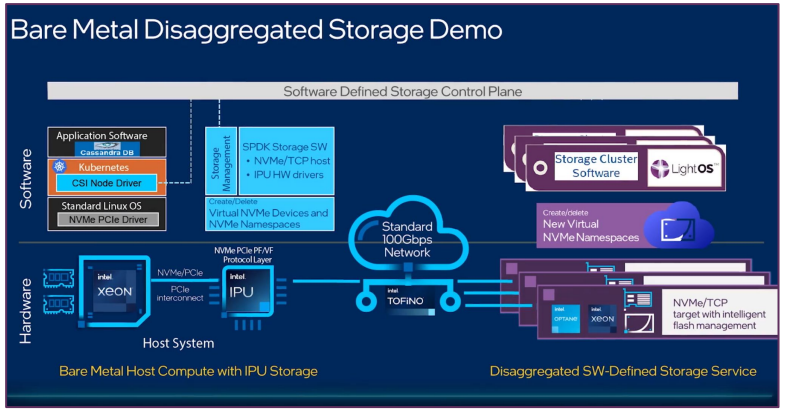


Things We Will Not Talk About: IPU-based Storage Clusters



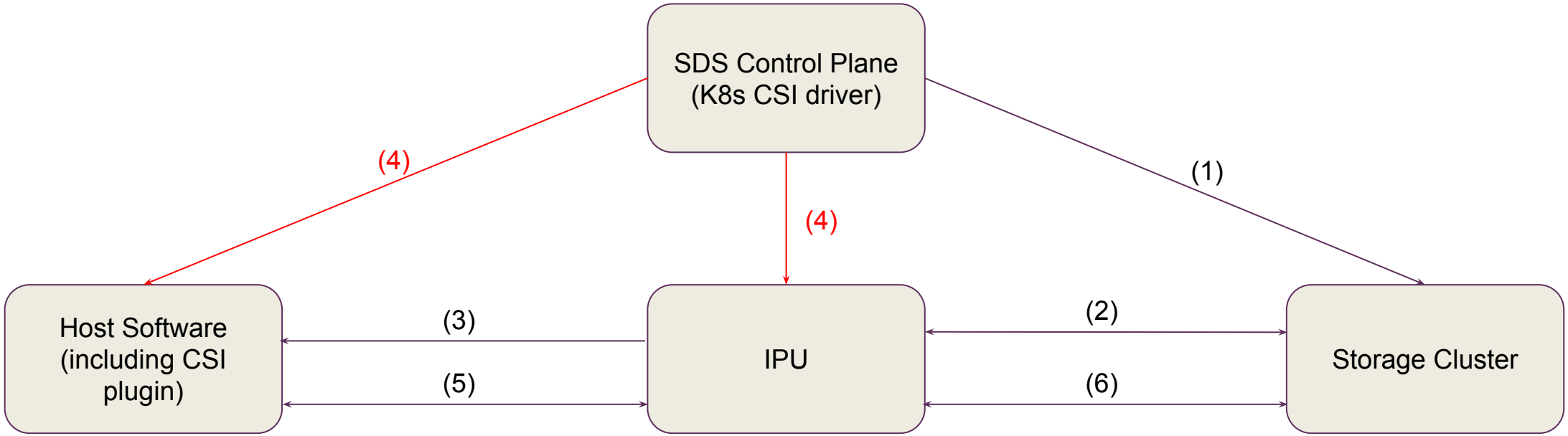
What Exists Today

- ✓ K8s
- ✓ CSI drivers
- ✓ NVMe/TCP for the data path and discovery
- ✓ Programmable IPUs/DPUs/SmartNICs
  - Accelerators
  - SPDK/IPDK
- ✓ NVMe/TCP Software Defined Disaggregated Storage Clusters



See the [IPU Bare Metal Disaggregated Storage demo]

Configuring IPUs for Remote Storage



(1) create volume
(2) discover volume on storage cluster
(3) create emulated local volume on the host

(4) **map remote volume X to IPU PF/VF Y and, in turn, to local block device Z**
(5) access local volume via NVMe
(6) IPU access to remote volume via NVMe/TCP

An IPDK Shopping List

1. “here's our cluster's discovery endpoint, here's the UUID of the volume we want, now surface it as a local NVMe device on the host, connected to this PF or VF”
2. A joint API that is common to most if not all SmartNICs and IPUs
    a. For configuring remote storage
    b. For deployment and provisioning of local services
    c. For VXLANs and network virtualization
    d. For network transport security, e.g., IPsec
    e. For storage data-at-rest encryption/decryption
    f. For end-to-end data integrity configuration (e.g., DIF)
    g. For resource metering and limiting (bandwidth and/or IOPs QoS, rate limiting)
    h. For billing?
3. Support for controlling IPUs both locally from the host and remotely from some centralized management layer
    a. potentially different mgmt access transports, security considerations, "ownership", etc.
4. Simplicity - keep the APIs and abstractions as simple as possible but no simpler. Clear and concise error reporting.
5. Robustness - the APIs should be race-free, safe in the face of retries/crashes/outages/concurrency. For block storage, "it *usually* works" is not considered acceptable.
6. Ultimately: “do one thing and do it well”

Thank You

lightbitslabs.com

info@lightbitslabs.com

lightbits