YDB

Working with Raw Disk Drives in Kubernetes — YDB's Experience

在Kubernetes中使用原始磁盘驱动器——YDB的经验

Ivan Blinkov
VP, Product and Open-Source

Ivan Blinkov

- Over a decade of experience in the database management systems (DBMS) development industry
- Talked with countless DBMS users and stakeholders to understand how and why they ended up with a specific solution
- Worked on a handful of DBMS products, including two open-source ones:

ClickHouse YDB

Agenda

1

What’s YDB and why it works with raw disk drives directly via block devices?

2

Nuances of working with raw disk drives in Kubernetes

3

Lessons learned along the way

Why YDB works with raw disk drives directly?
(without any filesystem)

YDB: Open-Source Distributed SQL Database

Mission critical

- Designed for services with 24×7 uptime requirements
- Serializable consistency
- Adapts to workloads
- Security features

Highly available

- Survives AZ plus rack failure without human intervention
- Seamless upgrades
- Self-healing
- Smart SDKs

Data platform

- Row-oriented tables (OLTP)
- Column-oriented tables (OLAP)
- Topics (persistent queues)
- Federated queries
- Multitenancy

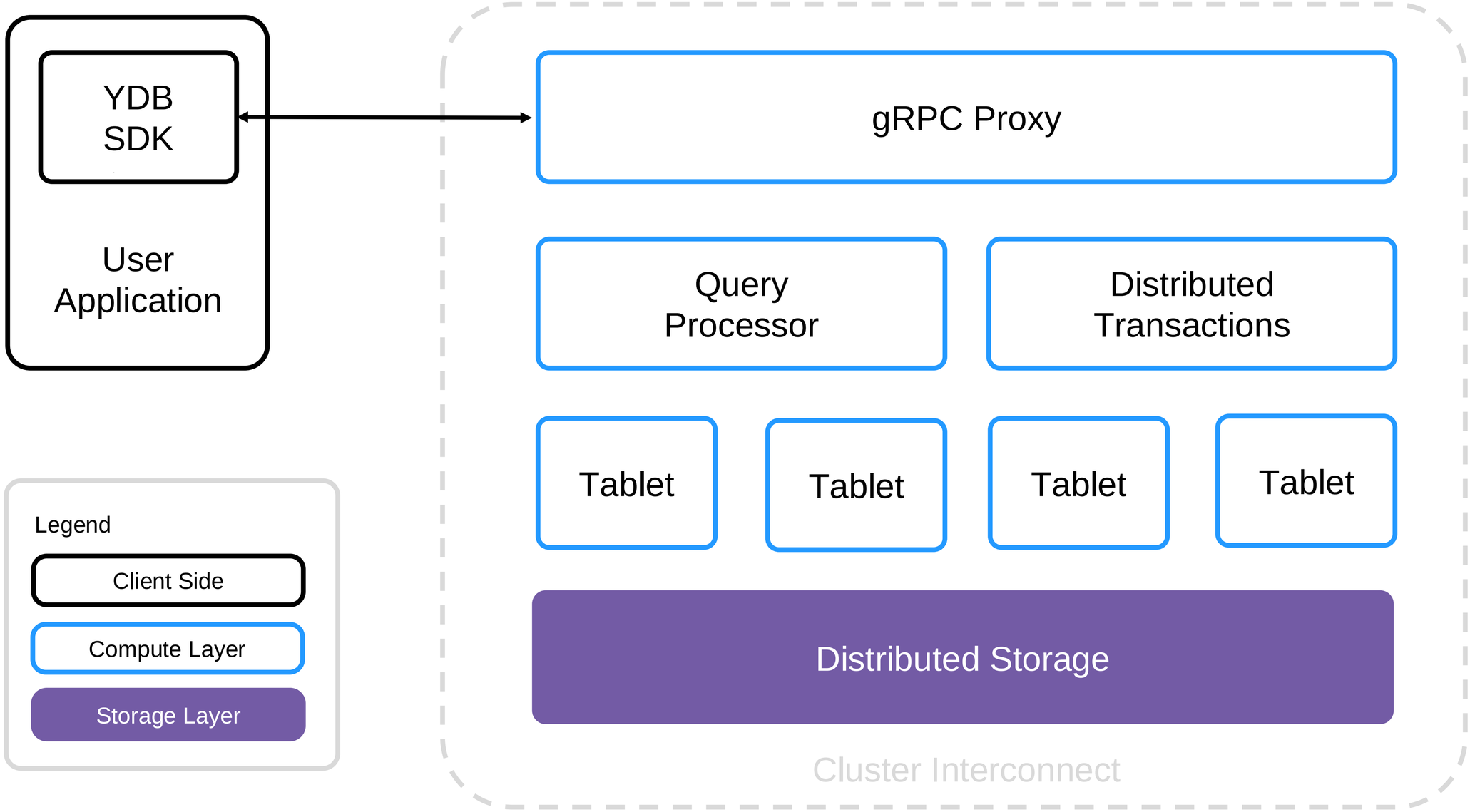
Typical YDB use cases

- Finance
- E-commerce
- Ride-hailing
- Advertisement
- Telecom
- Logistics
- AI services
- Infrastructure

Summary of YDB history

2014	Started as an in-house infrastructure technology
2020	Provided as a managed cloud service
2021	Kubernetes compatibility prototype
2022	Published to open-source under Apache 2.0 license
2023	First production clusters running in Kubernetes

YDB high-level architecture



What impacts distributed database performance?

Latency

- Network hops required for distributed transactions
- Caching
- Slow disk drives
- Retries

Throughput

- Hardware capacity
- Efficient algorithms
- Transaction contention
- No unnecessary overhead

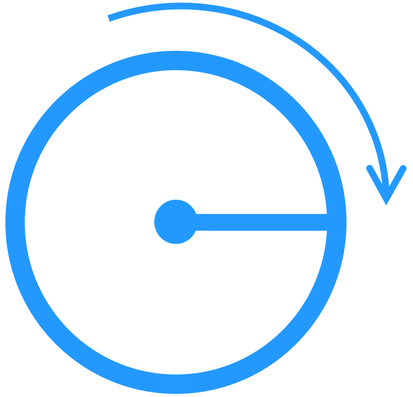
Typical performance goal: maximize throughput while maintaining reasonable latencies

What a virtual filesystem provides?

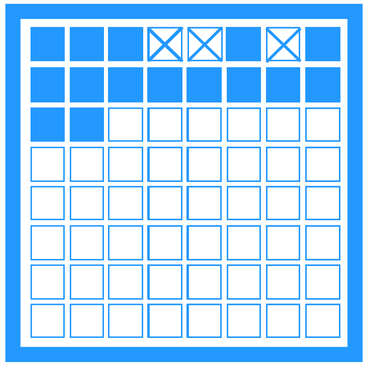
- Generic abstractions: files, folders, permissions, etc.
- Generic operations with these abstractions
- Mounting multiple devices into a single namespace
- Hiding device implementation details
- I/O scheduler and page caching
- Sometimes encryption and compression

How to live without a filesystem?

- Design data layout that makes sense for the workload
- Consider physical device properties
- Don’t trust someone else to manage caches:
  - Open block device files with `O_SYNC | O_DIRECT`
  - Disable device write cache
- Application-level I/O scheduler
- Checksum everything



HDD



NVMe

Nuances of working with raw disk drives in Kubernetes

Naïve approach: mount block devices to containers

- Simple, straightforward, and built-in
- Containers need superuser privileges to work with the block device
- Breaks the least necessary privileges principle
- Not suitable for production environments

```
spec:
  containers:
    securityContext:
      privileged: true
   ...
  volumes:
  - name: device-dir
     hostPath:
       path: /dev
```

PersistentVolume with volumeMode: Block

- Great option for new clusters with dynamic volume provisioning
- Not so much for self-managed Kubernetes cluster on bare-metal spanning several generations of legacy hardware
- StatefulSet supports only one set of PersistentVolume's

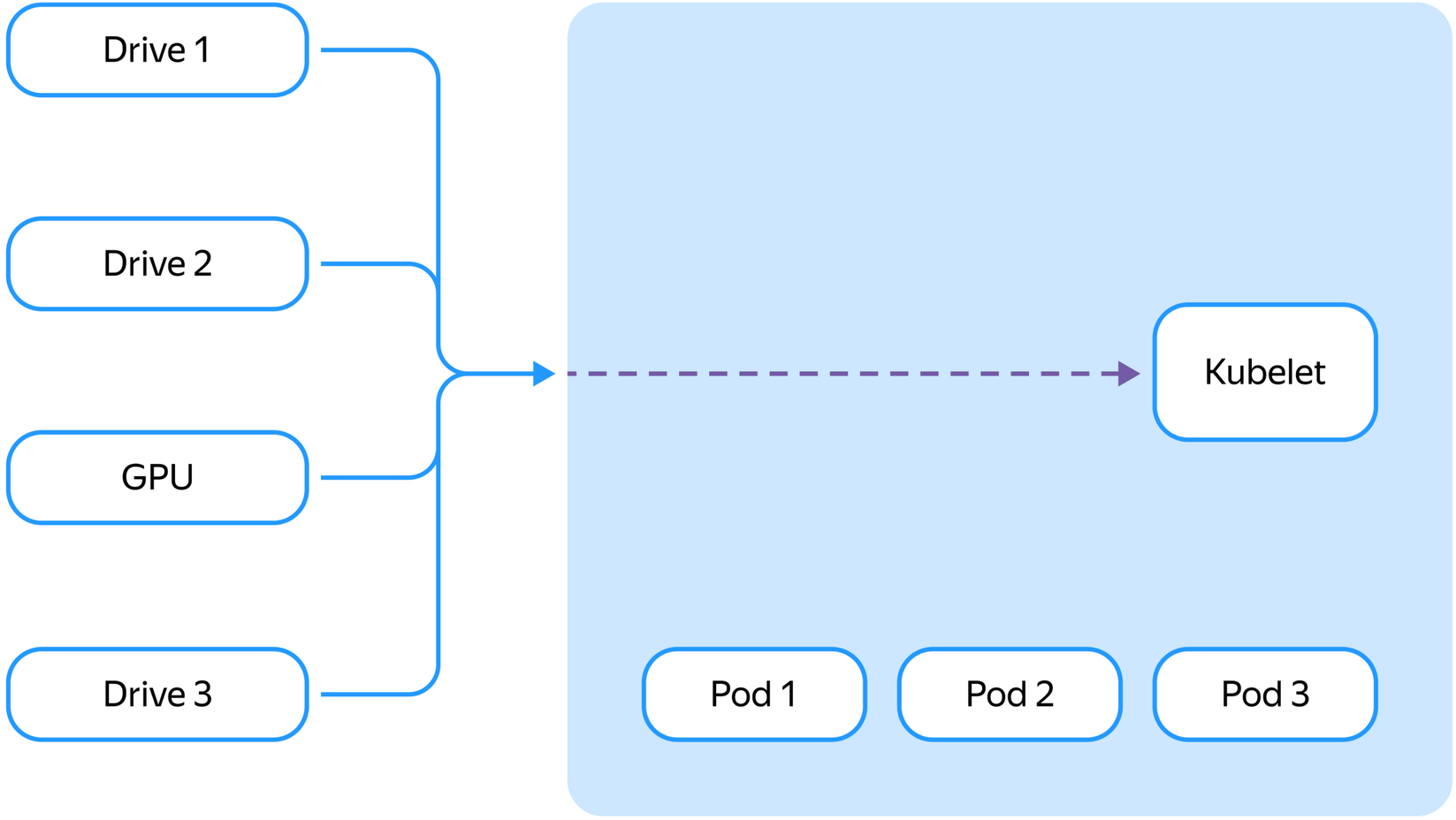
```
spec:
  dataStore:
  - accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 256Gi
    volumeMode: Block
```

Kubelet device plugin

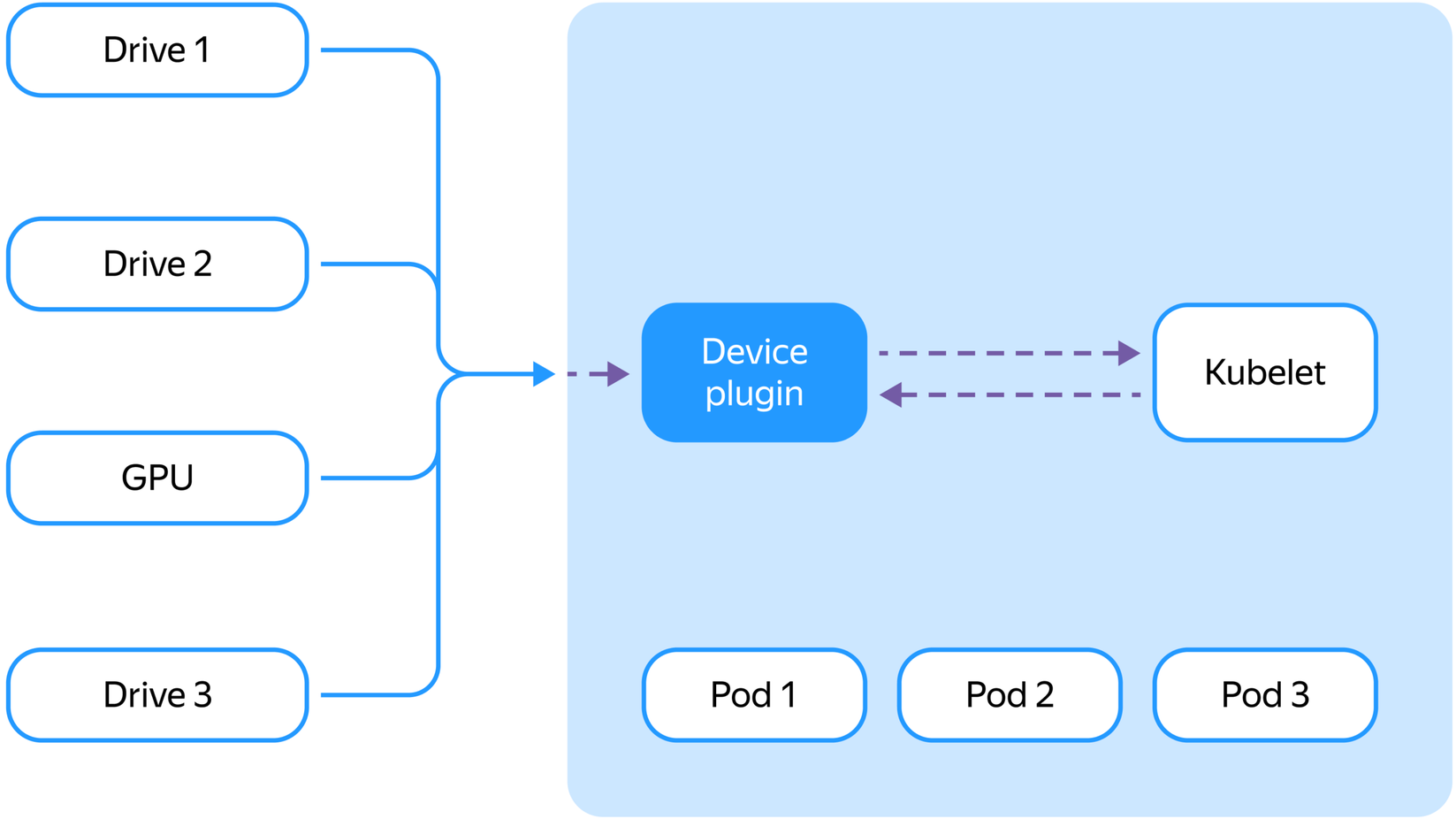
- An API for advertising system hardware resources
- Initially designed for GPUs, FPGAs, etc.
- Can be used for disk drives too

```
resources:
    limits:
      ydb-disk-manager/hostdev: "1"
    requests:
      ydb-disk-manager/hostdev: "1"
```

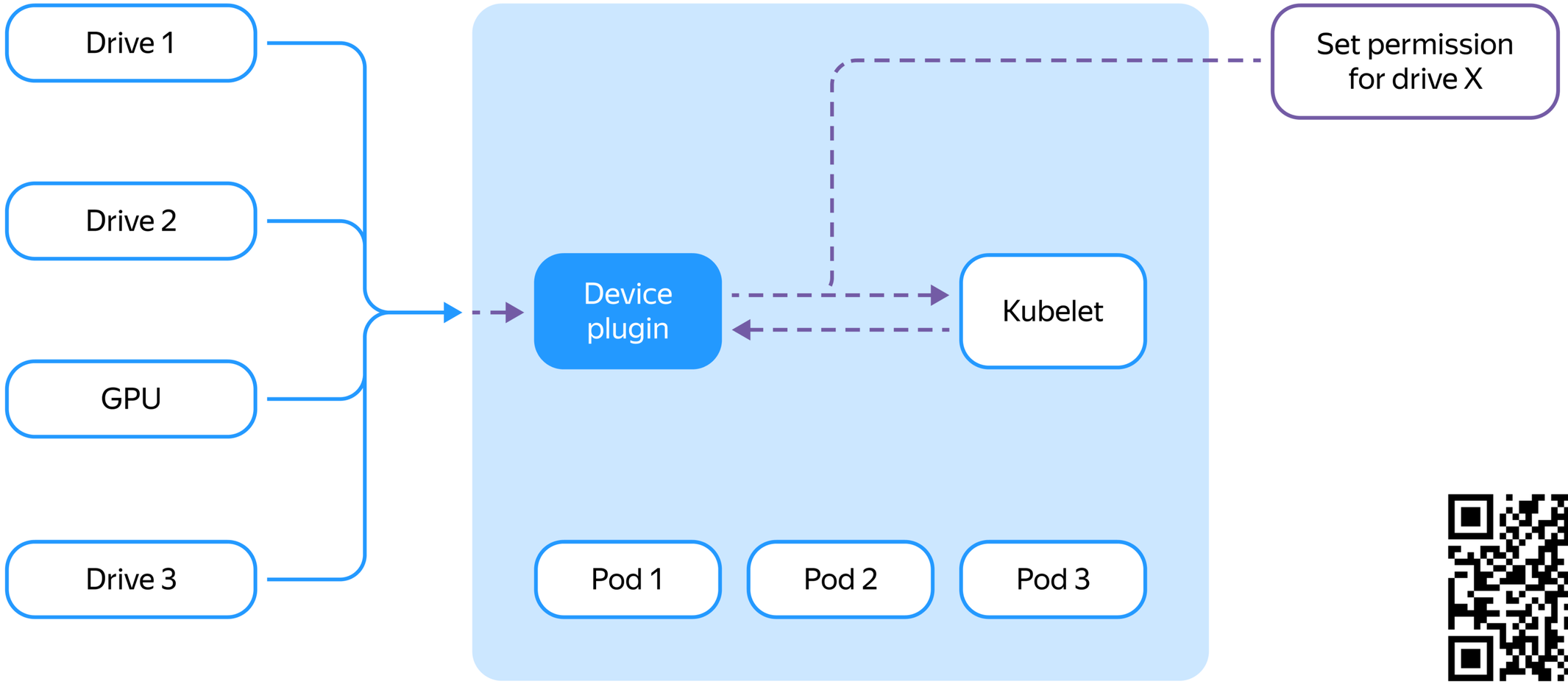
Kubelet device plugin



Kubelet device plugin



Kubelet device plugin



Local Persistence Volume Static Provisioner

- Specifically intended for managing disk volumes, including the block device mode
- Maintained by the SIG Storage of Kubernetes
- Not so actively developed recently

Dynamic Resource Allocation

- Generalized ResourceClaim concept
- Somewhat similar to the “Kubelet device plugin" option
- Thoroughly designed and flexible
- Currently in alpha

Looking forward for production-readiness!

Disk drive type trade-offs

	Performance	Portability	Reliability
Local physical device	🚀🚀🚀	—	—
RAID	🚀	—	✅
Network block device	🚀	✅	✅
Non-replicated network block device	🚀🚀	✅	—

Disk drive type trade-offs

	Performance	Portability	Reliability
Local physical device	🚀🚀🚀	—	—
RAID	🚀	—	✅
Network block device	🚀	✅	✅
Non-replicated network block device	🚀🚀	✅	—

Lessons learned

1. Kubernetes provides tools even for uncommon requirements like persisting data without a filesystem
2. Taking advantage of Kubernetes flexibility is possible even without major architectural changes
3. Where the Kubernetes cluster runs matters

YDB is 100% open-source

Permissive Apache 2.0 License for:

- Core platform is built from scratch in C++
- Kubernetes operator in Go
- SDKs in Java, Python, Go, Rust, Node.js, PHP, etc.
- Documentation in Markdown

Contributors are welcome!

https://github.com/ydb-platform/ydb

YDB

Thank you!

https://ydb.tech

YDB highlights:

- Strong consistency
- Resilience and self-healing
- Elastic scalability
- Various workloads
- PostgreSQL and Kafka compatibility
- 100% open-source under Apache 2.0