Everpure

WHITEPAPER

Beyond CSI

The Portworx Storage Architecture Built for Real World Kubernetes Workloads

Abstract

As Kubernetes adoption accelerates, the need for a robust persistent storage strategy has become mission-critical. While CSI gives platform teams a starting point for storage management, scaling stateful workloads like databases, messaging systems, and AI quickly expose the limitations of CSI's direct mapping approach. Portworx provides a container-native storage implementation that aligns with Kubernetes' dynamic scheduling behavior.

This paper[1] is intended for platform engineers, SREs, and infrastructure experts seeking to move beyond basic persistent volume management toward a scalable, container-native storage strategy for mission-critical enterprise workloads.

What is CSI?

The Container Storage Interface (CSI) is a specification that lets storage vendors deliver persistent storage to container orchestrators like Kubernetes without the need to modify Kubernetes' core codebase. By separating the storage driver lifecycle from the platform, CSI helps vendors release plugins independently of the Kubernetes release cycle.

CSI drivers work by exposing block-level LUNs (essentially units of physical storage with unique identifiers that behave like virtual volumes) from external storage arrays directly to Kubernetes worker nodes, where they are then mounted and used by pods.

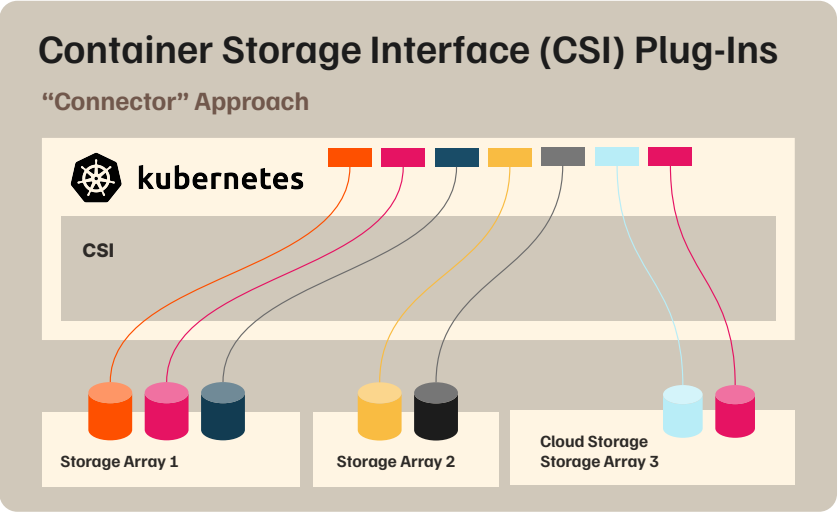


Figure 1: CSI creates numerous individual PV to storage mappings

CSI specifies a baseline of common capabilities for all storage devices, like provisioning and volume lifecycle management. Each vendor then creates plugins that extend the baseline to enable their own unique features and capabilities. So CSI gives storage and platform teams a basic infrastructure-centric entry point for Kubernetes storage connectivity.

Overall, CSI is a satisfactory solution for workloads that are relatively static and therefore don't cause orchestration churn. But as clusters scale, workloads diversify, and operations become more dynamic, the limitations of direct-mapped CSI storage become clear.

Where CSI Falls Short

Organizations faced with Broadcom's price increases and licensing changes to VMware are re-evaluating their virtualization strategies. Those looking for alternatives to VMware are investigating the operational realities of high-scale KubeVirt deployments and running COTS applications in Kubernetes. In these situations the limitations of array-backed CSI become harder to ignore. It introduces substantial operational complexity as Kubernetes environments scale.

Figure 2: Limitations of CSI

Reduced automation from operational inconsistency. In real-world Kubernetes environments, you're likely using more than one kind of storage. Each vendor extends the basic CSI requirements differently, with different features, APIs, operational practices, patches, and roadmaps. For example, the basic CSI spec doesn't tell a vendor how to handle retries, timeouts, or error recovery. This diversity in capabilities complicates the ability of platform teams to standardize operations and requires validation of CSI compatibility across Kubernetes versions.

Disruptive upgrades, workload rebalancing, and data migrations. CSI is an extension of traditional storage architectures, but it's not designed to support Kubernetes' level of dynamic, distributed scheduling. Especially in multi-cloud, multi-storage environments, CSI's operational mismatch with Kubernetes scheduling complicates upgrades, patching, and data migrations. This extends maintenance windows and builds technical debt. Each scenario involves moving data from one backend to another, whether to retire old hardware, redistribute load, or transition to a new storage system. These operations typically require manual coordination across teams and some form of data migration from one CSI volume to another using snapshots, replication, or application-level tools.

The result? Added risk, increased platform complexity, and unwanted downtime or degraded service during cutovers. Initial synthetic benchmarks may seem performant but can miss issues that real-world workloads at scale eventually surface.

Disaster recovery planning or execution must consider the limitations of each CSI driver, as not all support replication, asynchronous disaster recovery, or even native snapshots.

With CSI, operational alignment between storage and platform teams becomes increasingly critical. Every storage-related lifecycle event requires coordinated activity between the storage team, the platform team, and often the application owners.

Kubernetes Mobility vs CSI Rigidity

Kubernetes maximizes resource use through high workload mobility, but this mobility strains traditional storage arrays. While CSI drivers automate the process, the hidden overhead of moving volume attachments creates significant friction during cluster maintenance or node patching.

Volume Migration Overhead

Every time a pod moves, the storage layer must execute a heavy four-step sequence:

1
Map
API makes calls to the array to map LUNs to the new host.

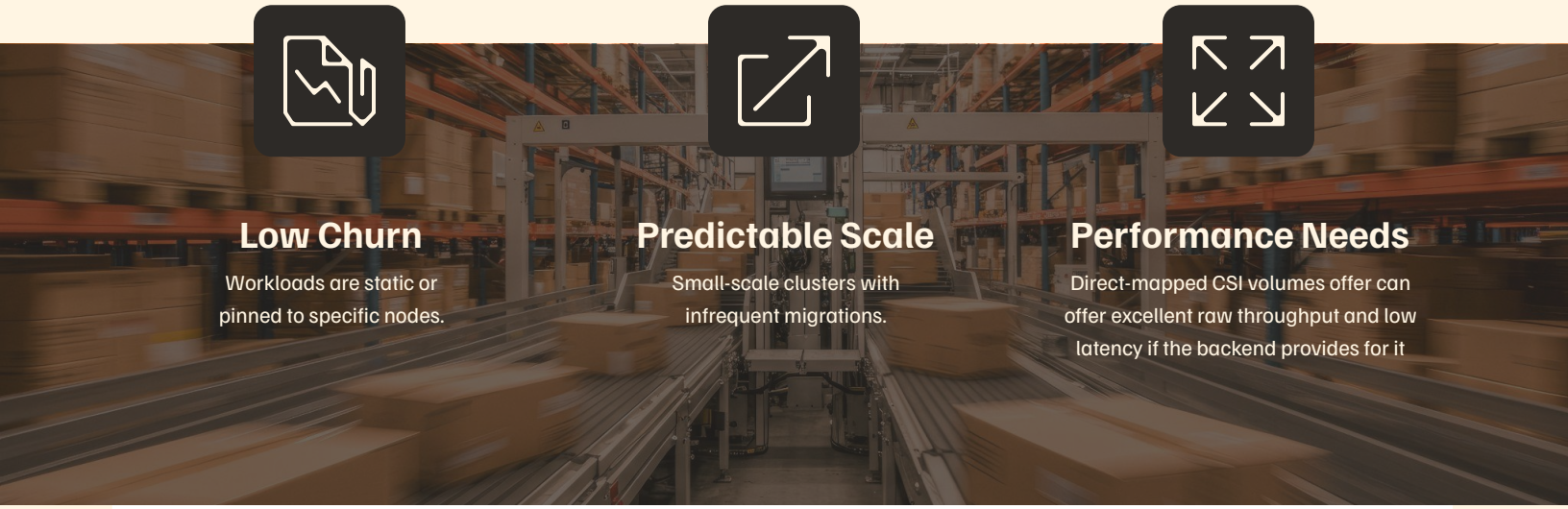
2
Attach
Host-level SCSI bus scans, multipath builds, and udev events.

3
Detach
Teardown of multipath and SCSI devices on the original host.

4
Unmap
API call to the array to remove access from the original host.

This process doesn't just impact worker nodes; it creates a control plane bottleneck. At scale, especially when multiple clusters share one array, the sheer volume of map/unmap operations and device rescans can spike latency across the entire environment, risking congestion during high-churn orchestration windows.

Despite these overheads, CSI remains a strong choice for environments with certain characteristics:



Low Churn
Workloads are static or pinned to specific nodes.

Predictable Scale
Small-scale clusters with infrequent migrations.

Performance Needs
Direct-mapped CSI volumes offer can offer excellent raw throughput and low latency if the backend provides for it

In sum, while CSI will work for relatively static, low-mobility workloads, **the technical debt that it creates makes migration to a true data management platform a bigger engineering challenge** when the time comes to address dynamic, large-scale deployments.

Portworx: The Container-Native Datastore for Kubernetes

While CSI maps individual storage volumes directly to pods, Portworx shifts the storage paradigm from individual LUN management to a container-native layer that abstracts and unifies the underlying infrastructure, whether on-prem, cloud, or edge. It does this via a container-native storage layer that runs inside the Kubernetes cluster. Portworx decouples data and storage operations from the underlying hardware and presents a single software-defined abstraction. This globally available storage pool is what your applications use, whether they're in containers or KubeVirt virtual machines.

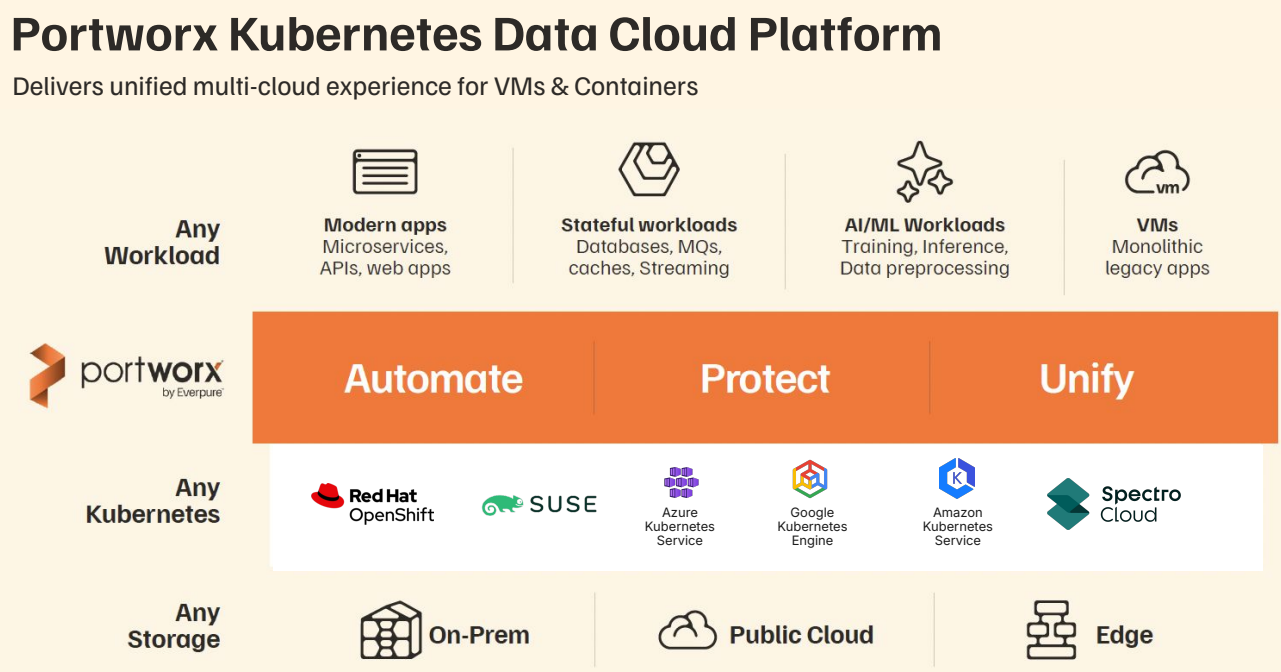


Figure 3: Portworx creates a unified in-cluster storage abstraction across all locations

Unlike a disjointed set of hardware-specific CSI plugins, Portworx delivers automation for hybrid and multi-cloud consistency.

- ☑ Aggregates storage across premises and clouds into a single fabric without manual CSI driver workflows
- ☑ Manages data lifecycle, protection, and mobility with no disruption to applications
- ☑ Goes beyond CSI with data protection, backup, disaster recovery, and more
- ☑ Supports any vendor's block or cloud storage

Reducing Storage Churn for Efficiency at Scale

In vSphere terms, Portworx echoes the VMware Datastore model. Instead of mapping a new LUN for every pod and creating a cascade of rescans during lifecycle events, Portworx presents a small number of stable, pooled devices to each node.

During a high-churn event like a rolling upgrade, the difference in system load is dramatic:

Array-Backed CSI

Potentially thousands of attach, detach, and rescan operations that overwhelm the array control plane in a large cluster.

Portworx

Only a few drive events occur per node (e.g., ~2 per node). Virtual volumes are already replicated and available across the fabric, requiring **no per-volume remaps or rescans.**

This way, Portworx acts as a facade layer, allowing platform teams to change underlying infrastructure (multi-cloud or on-prem) without involving application owners. This reduces operational risk, simplifies CI/CD pipelines, and ensures consistent performance regardless of workload mobility.

Portworx offers several benefits over standard CSI plugins:

Capability	CSI Plugins	Portworx Enterprise
Control Plane Velocity	Limited by hardware API rate-limiting; serial processing of volume events.	Distributed software control plane; parallelized volume orchestration.
RTO (Recovery Time)	**High.** Thousands of Attach/Detach operations required per cluster.	**Low**. Instantaneous volume availability via software-defined storage fabric.
DR Complexity	"Bolted-on" per-volume replication; requires custom scripting for failover.	Native, namespace-aware asynchronous replication and one-click failover.
Scaling Limit	Significant overhead at >10,000 volumes; API "thundering herd" risk.	Designed for hundreds of thousands of volumes; decouples platforms from hardware limits.
Operational Uniformity	Dependent on specific CSI driver and array firmware versions.	Consistent storage operations across all clusters and underlying hardware.

Figure 4: Portworx benefits over array-backed CSI

Performance Test: Live Migration of 400 VMs

Portworx created a live migration test[2] with our new KubeVirt benchmarking tool to assess the time to move a group of virtual machines from one host to another without service interruption. In this scenario, all VMs (up to 400 in parallel) are migrated from their original compute node to a new node.

This test stresses several subsystems simultaneously:

- Storage and I/O coherency
- Scheduler/ orchestration coordination
- System handling of hundreds of concurrent state transfers

This type of test is essential for validating cluster resilience, workload mobility, maintenance operations, and infrastructure elasticity.

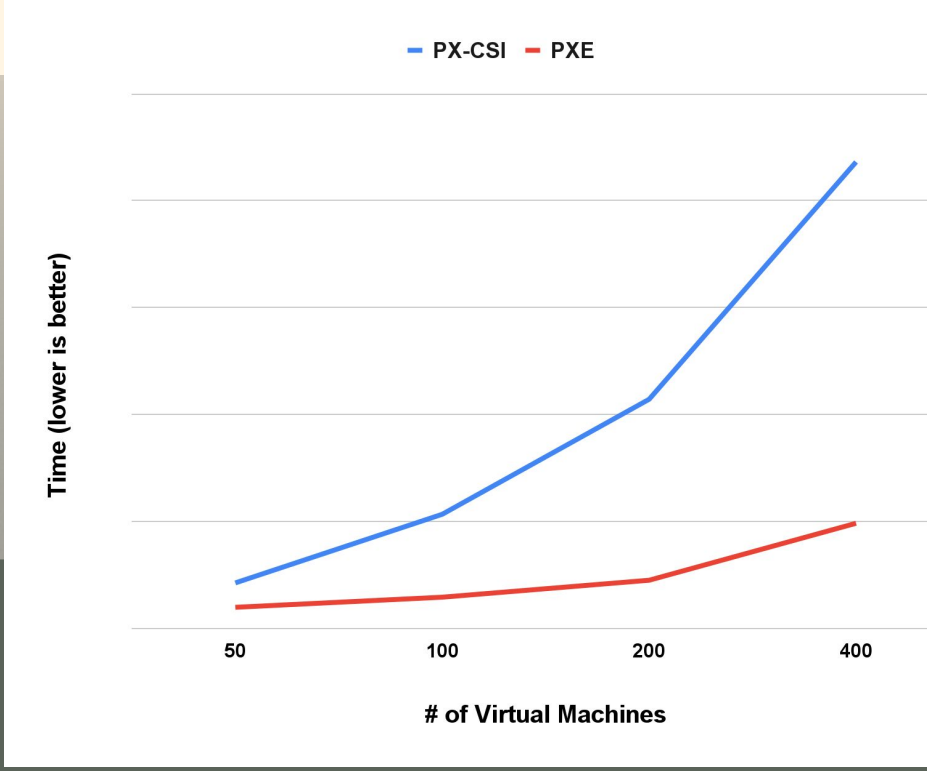


Figure 5: Live Migration Test Results

At every VM count from 50 to 400, Portworx completes live migrations dramatically faster than even Portworx's own optimized CSI driver.

Portworx scales much more efficiently and is approximately

4.4x faster

than CSI at 400 VMs.

Portworx: Container-Native Storage That Begins Where CSI Ends

CSI has an important role to play in the Kubernetes storage ecosystem, especially for simple use cases or smaller, static environments. But as clusters scale, workloads diversify, and operations become more dynamic, the limitations of direct-mapped, vendor-specific CSI storage become clear.

Portworx Enterprise addresses these challenges head-on by offering:

Virtual volumes with built-in replication and HA

Streamlined orchestration and near-zero infrastructure churn during upgrades

A consistent control plane across diverse environments

Application-aware, policy-driven storage management

It's the difference between orchestrating thousands of LUNs to multiple clusters versus operating a purpose-built, cloud-native storage platform. If you're serious about running stateful workloads in Kubernetes, Portworx isn't just an optimization—it's an enabler.

Resources

Video: External Arrays + CSI vs Cloud-Native Storage

White Paper: Top 5 Reasons CSI Drivers Fall Short

Knowledge Hub: A Complete Guide to Kubernetes CSI

Next Steps

Try Portworx for free at https://central.portworx.com/landing/register

[1] This white paper focuses on array-backed CSI drivers, such as those used with vCenter-integrated environments, SAN/NAS arrays, or cloud block storage like AWS EBS, Azure Disk, or Google Persistent Disk. CSI drivers for local storage, which use node-local disks or NVMe, follow a different architecture with their own trade-offs.

[2] The performance results reflect a specific configuration in our lab and should not be interpreted as indicators of system scale in customer environments.

Visit Our Website | 800.379.PURE

portworx by Everpure