

Achieving 10× Faster VM Migration to KubeVirt with Storage Offload in Forklift

Ryosuke Tatsumi
Chief Researcher, *Hitachi America*



- Background
- VM Migration Bottleneck
- Storage Offload Approach - XCOPY
- Technical Challenge
- Storage Offload Implementation Architecture
- Performance Result
- Contribution to Open Source

VM Migration Throughput Limits KubeVirt Adoption

KubeVirt is powerful, but VM migration throughput is barrier.

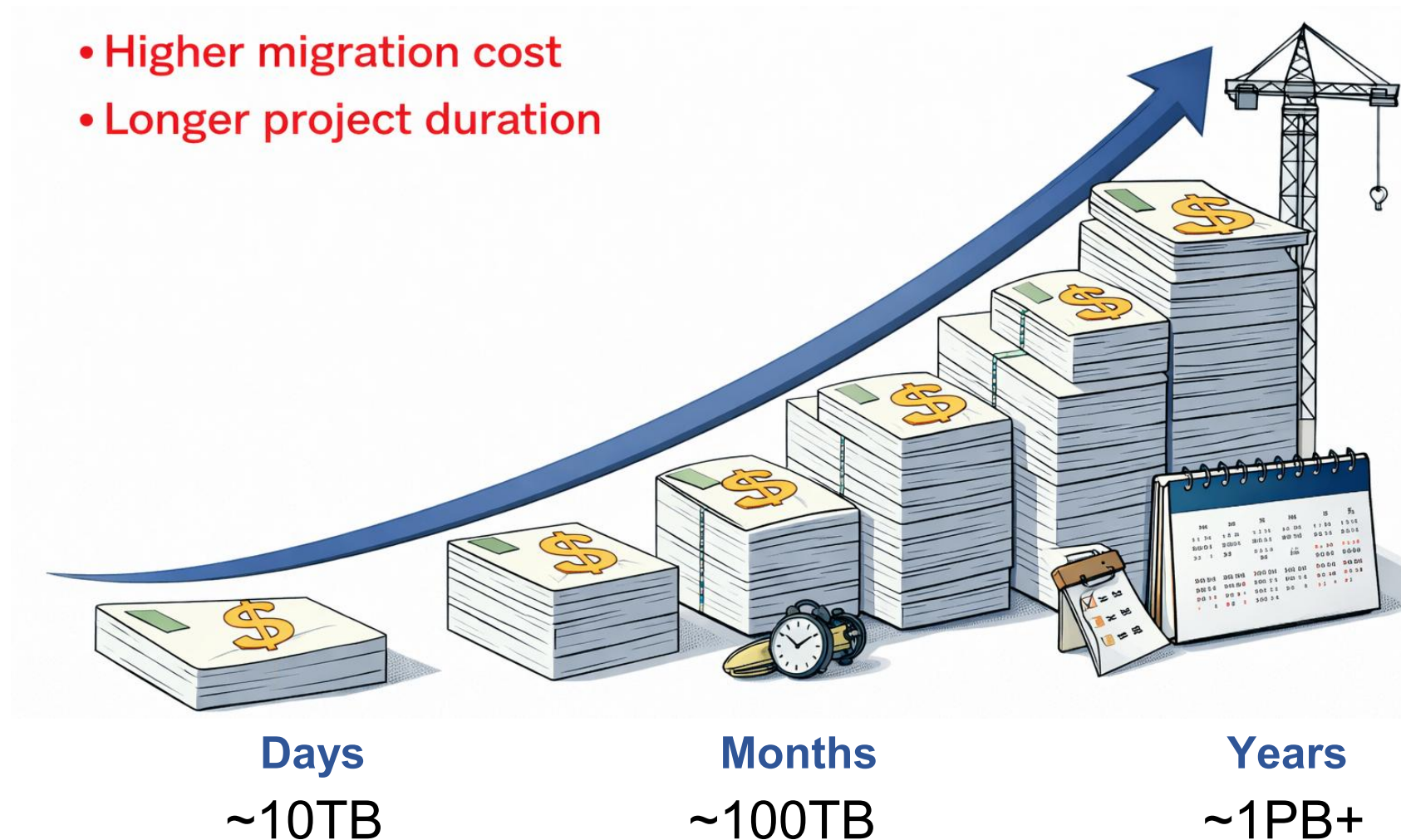
- **Long migration duration**
 - 18 months (Gartner 2025)
- **Maintenance window**
 - Cannot complete a VM migration with large disks (e.g. 30TB+)



Migration Time Scales with Data Size

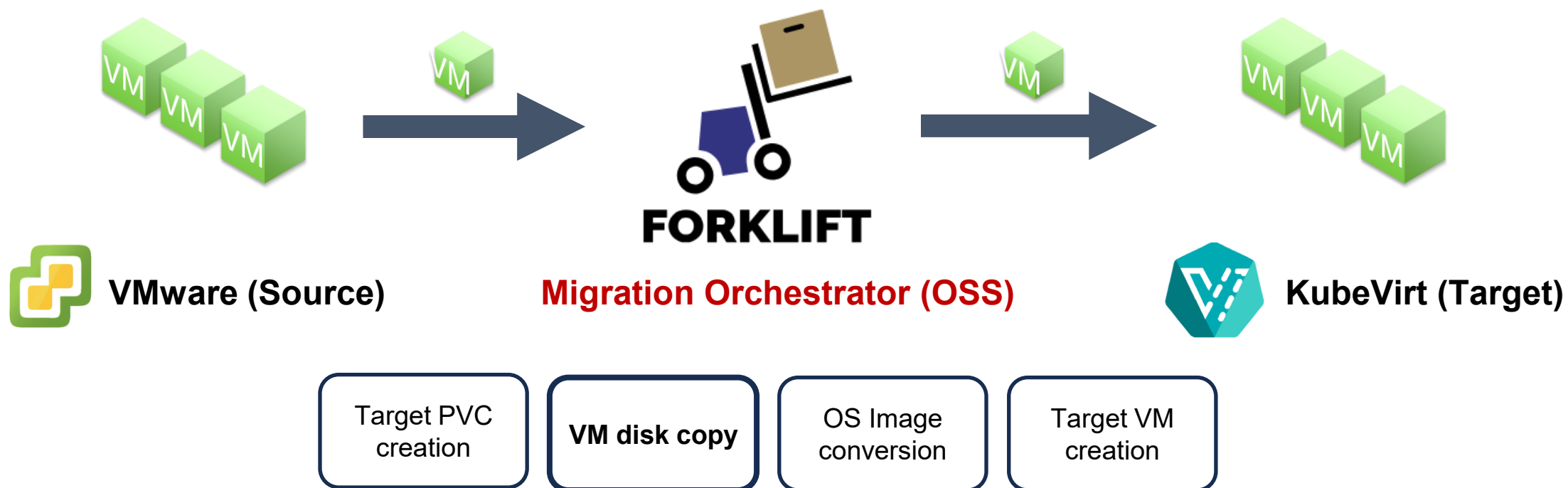
More data → longer migration → more cost

- Higher migration cost
- Longer project duration



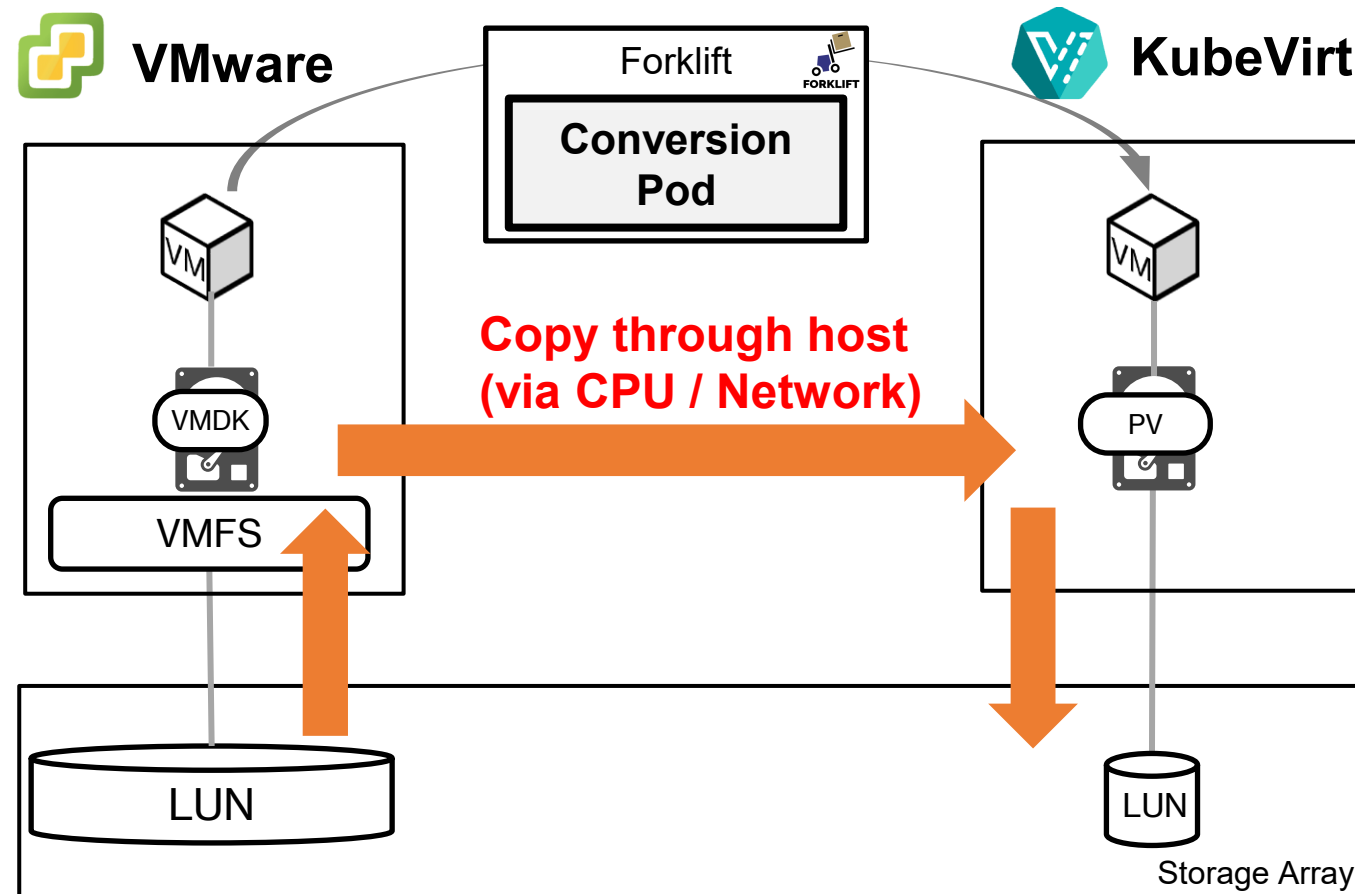
What is Forklift?

Forklift is an open-source tool for migrating VMs to KubeVirt.
It orchestrates VM migration workflow



Host-Based Data Path Becomes the Bottleneck

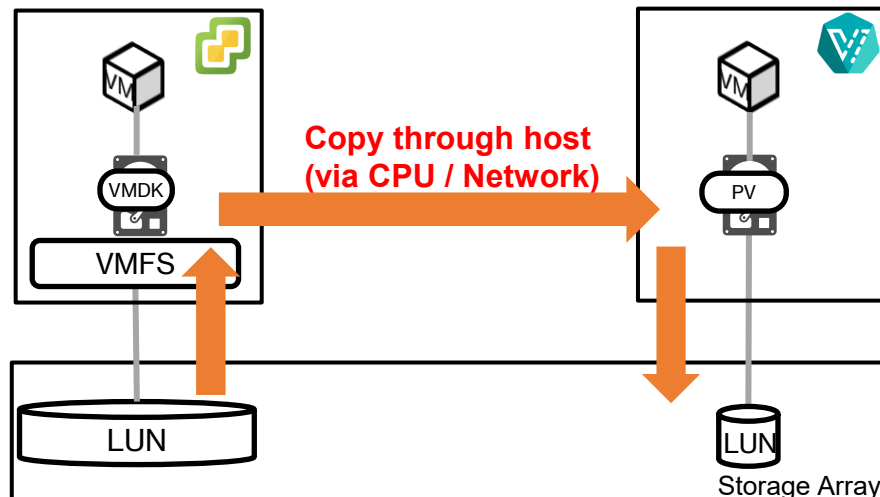
Forklift flows all VM data through a host-based copy using conversion pod.
→ This becomes the throughput bottleneck.



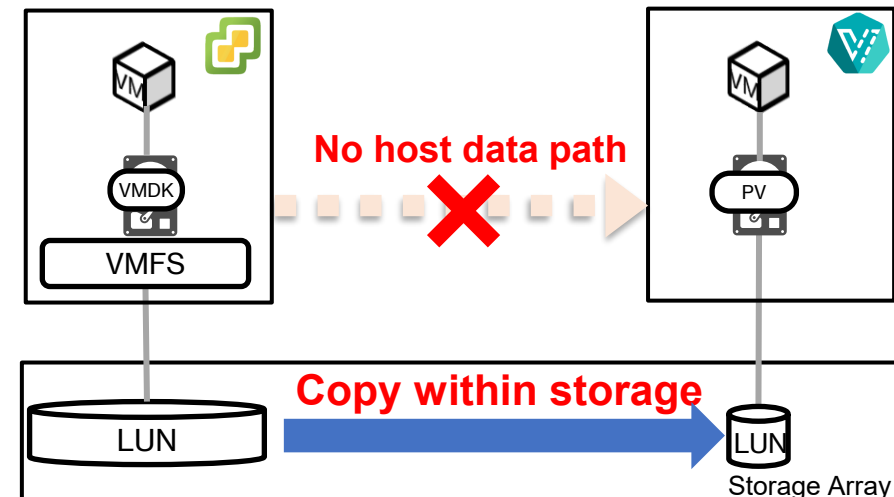
Solution Concept: Changing data path

To remove the bottleneck, we aim to change the data path from host-based copy to storage-based copy

Before (Host-based copy)



After (Storage-based copy)



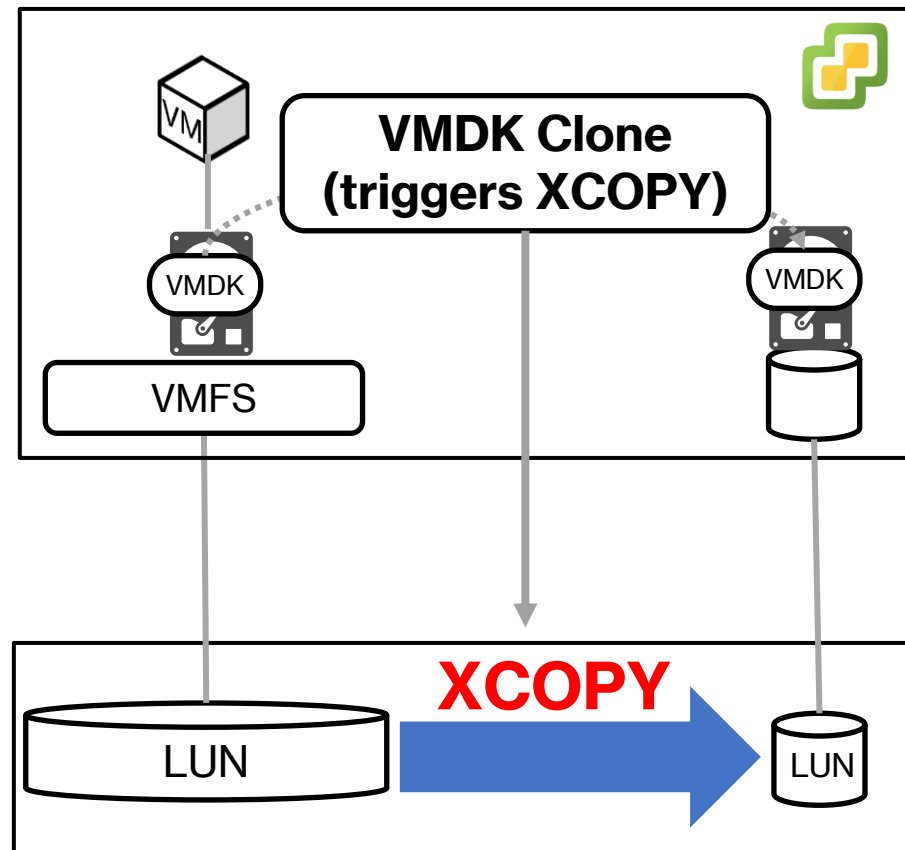
Eliminates bottlenecks on hosts

XCOPY Enables Storage-side Data Copy

XCOPY = Storage-side data copy (no host involved)

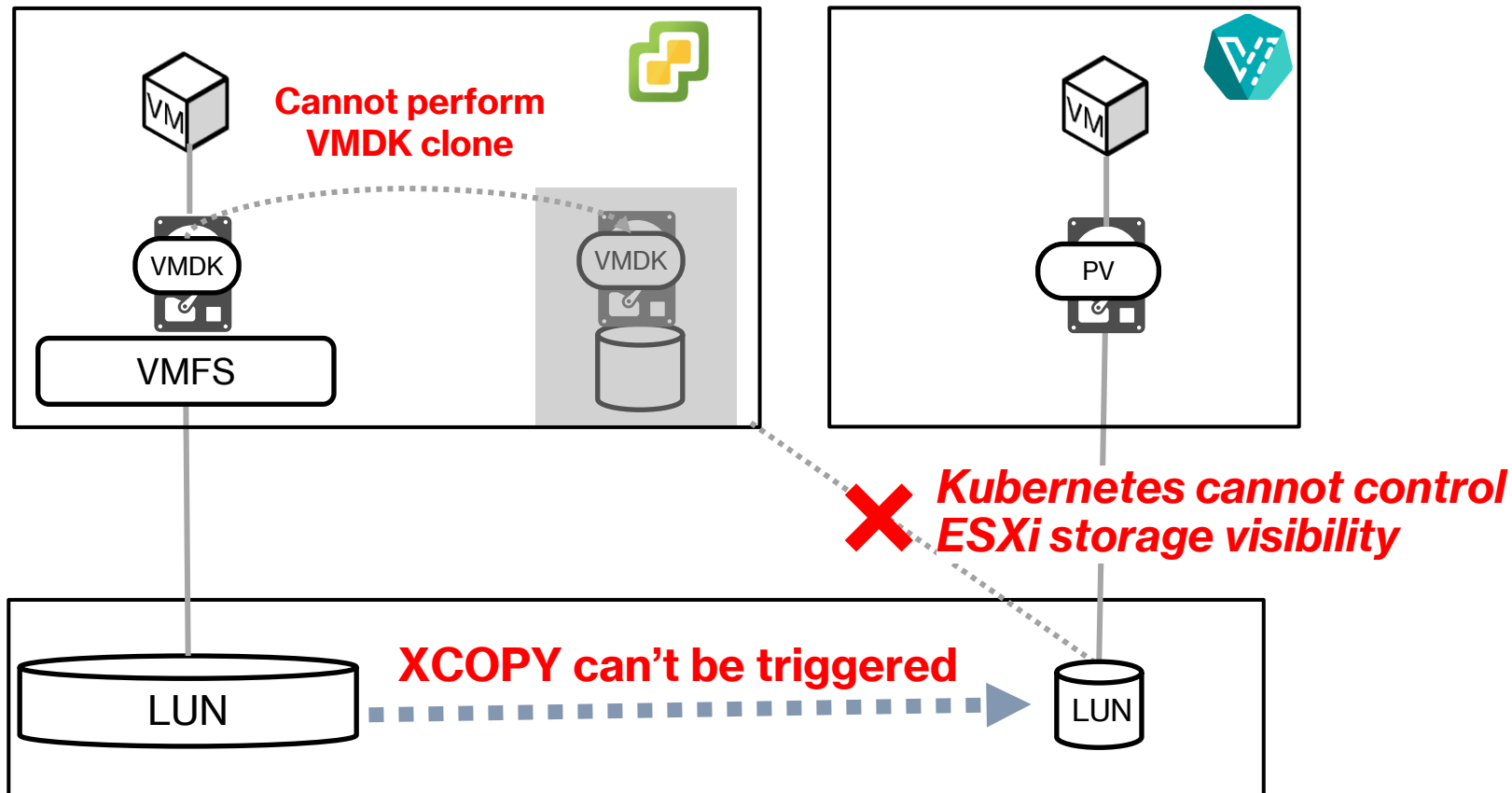
VMDK Clone triggers XCOPY in ESXi

This enables storage-based data path without host data movement



Technical challenge

- VMDK clone requires ESXi to have access to both source and target storage
Kubernetes(Forklift) does not control ESXi storage visibility
→ XCOPY cannot be triggered
→ Vendor specific storage operation is required

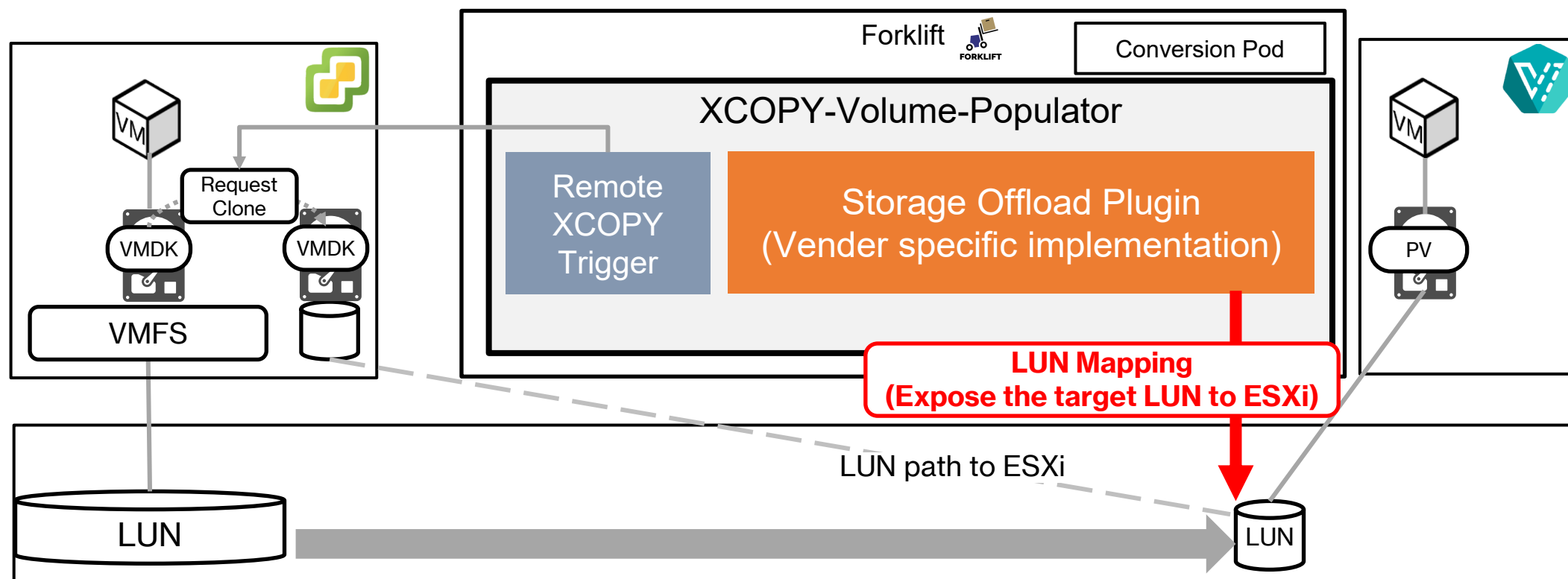


Architecture: Storage Integration

We bridge this gap with storage offload plugin layer.

→ Enables vendor-specific storage operations

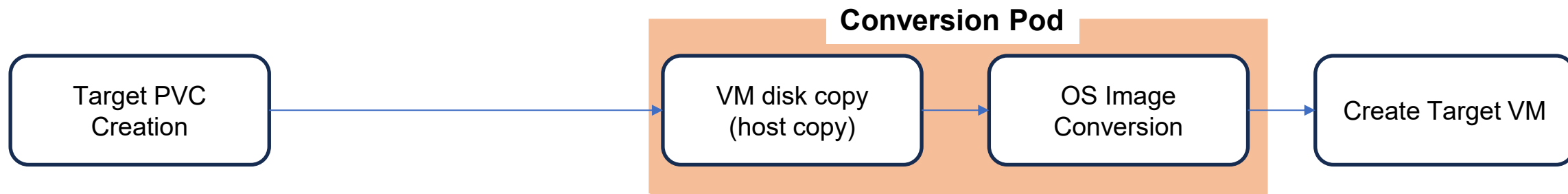
→ It operates to map the target LUN to ESXi to enable trigger XCOPY



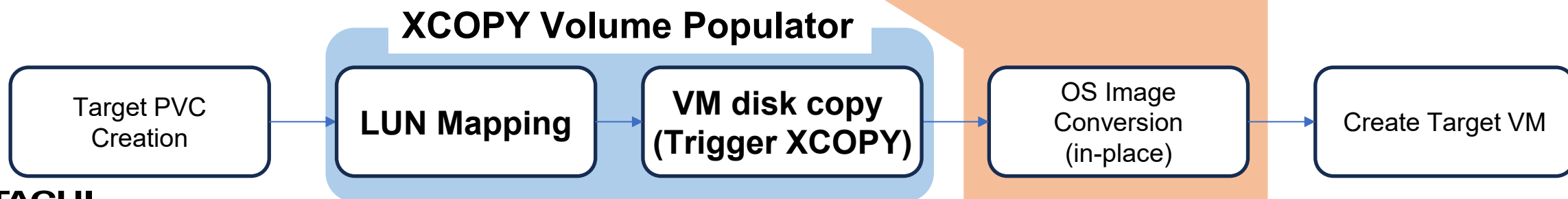
Storage Offloaded Migration Workflow

XCOPY Volume Populator runs as a part of Forklift Workflow
Not change the Entire Pipeline

Before (Host-based copy)

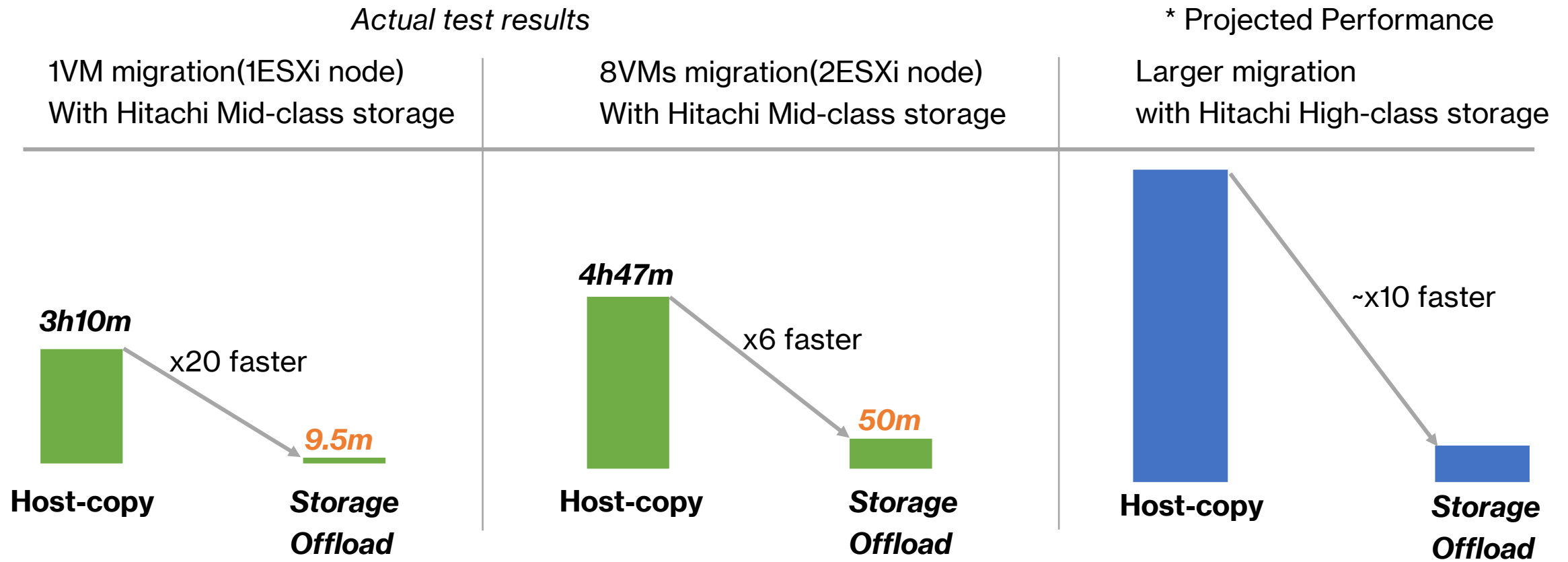


After (Storage-based copy)



Bulk Copy Becomes Up to 10× Faster

Up to x10 faster than standard Forklift



Applicable Across Migration and Datastore Types

Storage offload works across migration types and datastore formats

Category	Item	Storage Offload
Migration Type	Cold Migration	✓
	Warm Migration	✓ ^{*1}
	Live Migration	-- ^{*2}
Datastore Type	VMFS	✓
	vVol	✓ ^{*3}
	RDM	✓ ^{*3}

*1. Storage Offload is applied for bulk copy only.

*2. Forklift doesn't support Live Migration

*3. Some storage model use volume clone functionality for vVol and RDM instead of XCOPY

Applicability

- Most enterprise storage array support XCOPY. Some may not support.
- XCOPY works within the same storage array
 - Fallback copy is used when XCOPY is not work. -- It is still faster than host copy

Performance Characteristics

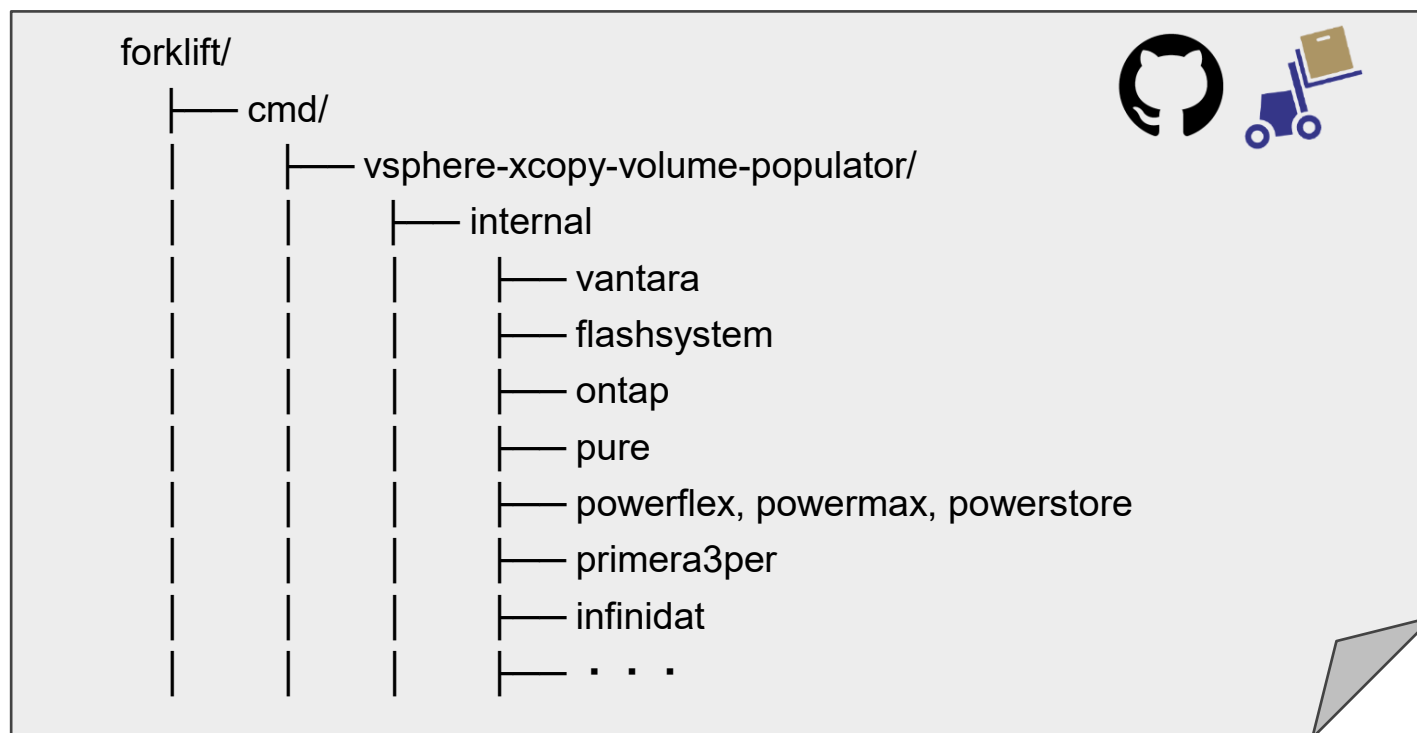
- Performance depends on the storage array capability
- Too large-scale concurrent migrations may impact storage I/O performance

Operation Considerations

- Continuous small-scale concurrent migrations gives predictable completion time.

Contribution to Open-Source Community

- Open Sourced!!
- **Hitachi Vantara and RedHat** initiated a pluggable design for storage offload
- This solution is supported by multiple storage vendors
 - not only Hitachi Vantara, but other major storage vendors.
- Implemented and maintained in Forklift OSS repository
(<https://github.com/kubev2v/forklift>)



How to use storage offload migration

How to enable to use storage offload

- *Prepare Forklift operator and setup a standard migration configuration*
- Create Secret for your storage array
- Add parameters below to StorageMap
 - ① Add plugin option name
Fixed: “*vsphrereXcopyconfig*”
 - ② Add the Secret name
 - ③ Add your storage(vendor) name

StorageMap

```

16 spec:
17   map:
18     - destination:
19       storageClass: sc-e1090-715029
20     offloadPlugin:
21       ① vsphereXcopyConfig:
22       ② secretRef: sopm-vantara-715029
23       ③ storageVendorProduct: vantara
24   source:
25     id: datastore-13256
26     name: Datastore-HAL-VVOL-029
27   provider:
28     destination:
29     apiVersion: forklift.konveyor.io/v1beta1
  
```

Problem

- VM migration does not scale with data size
- Host-based data copy is the bottleneck

Solution

- Change data path to storage offload
- 10x faster migration.

For community

- Bring storage-native capabilities into Kubernetes
- Enables open, pluggable ecosystem



[GitHub-Forklift](#)

HITACHI

 Hitachi Vantara



[Demo Video](#)



KubeVirt

SUMMIT

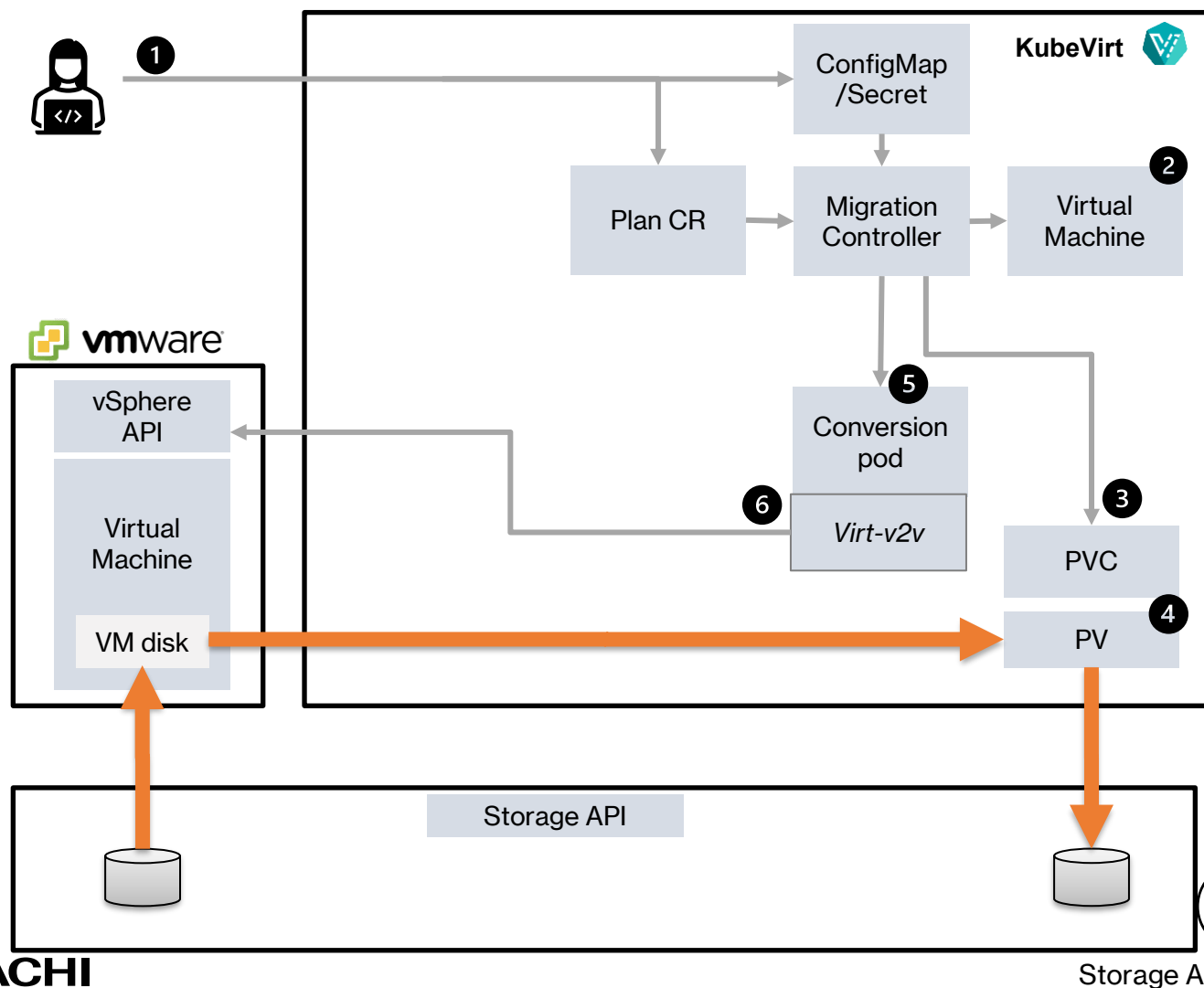
EUROPE

CLOUD NATIVE THEATER

HITACHI

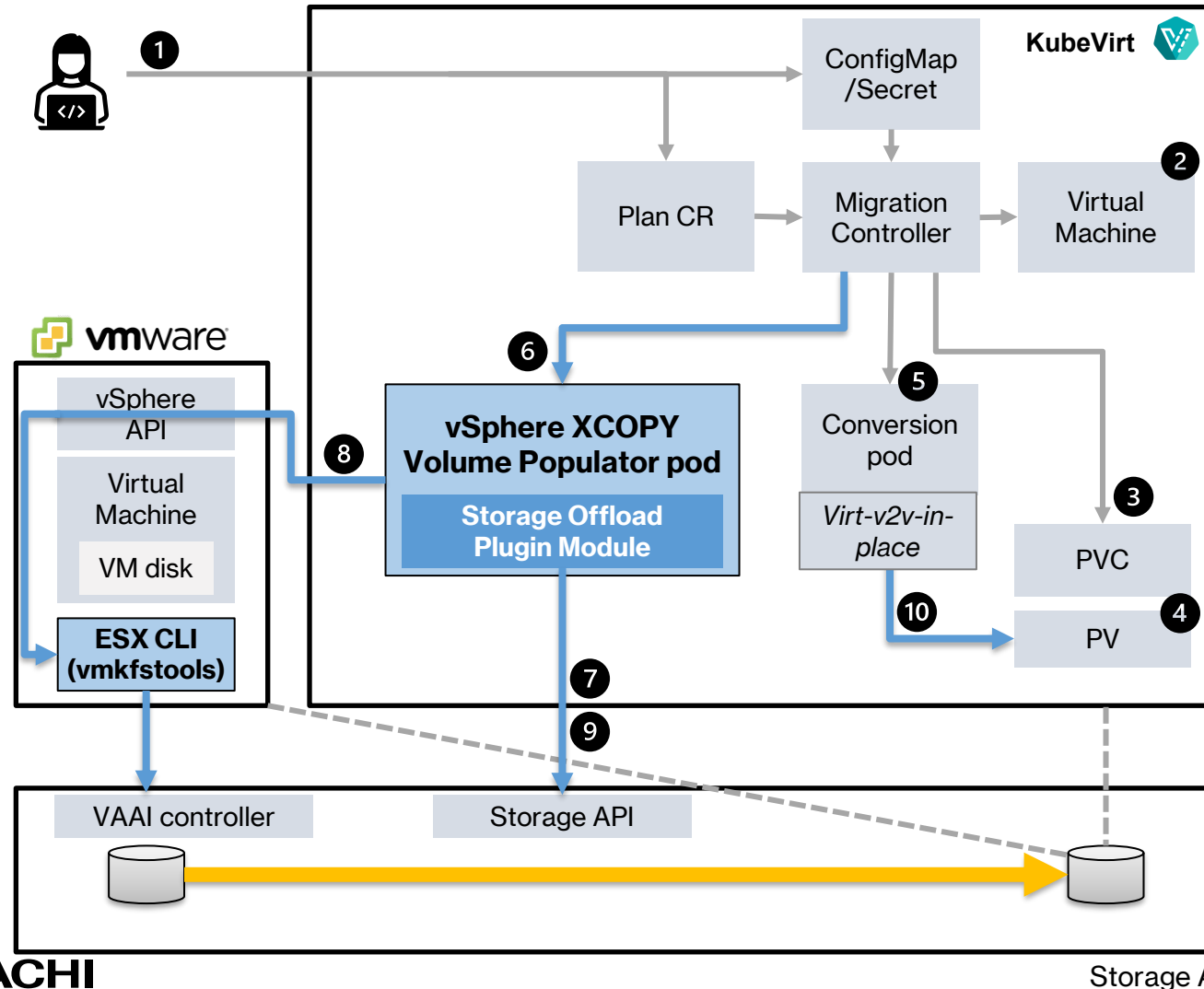
 **Hitachi Vantara**

Appendix. Forklift Workflow



1. Create Plan CR and ConfigMap(User)
2. Create VM CR (Migration CTL)
3. Create PVC (Migration CTL)
4. Create PV and mount volume (CSI driver)
5. Create Conversion Pod (Migration CTL)
6. Data transfer and image conversion via Virt-v2v (Conversion Pod) **host-based data copy**
7. Launch VM pod if it's needed

Appendix. “Forklift +Storage offload” Workflow



1. Create Plan CR and ConfigMap(User)
2. Create VM CR (Migration CTL)
3. Create PVC (Migration CTL)
4. Create PV and mount volume (CSI driver)
5. Create Conversion Pod (Migration CTL)
6. Create vSphere XCOPY Volume Populator Pod
7. Map target LUNs to ESXi
8. Execute remote ESX CLI “vmkfstools”
Storage-based data copy
9. Unmap LUNs from ESXi
10. Image Conversion via Virt-v2v-in-place (Conversion Pod)
11. Launch VM pod if it’s needed (*Finish*)

