SPDK-CSI : Bring SPDK to Kubernetes Storage

Yibo Cai

arm
About Me

- Principal Software Engineer from Arm
- 20 years in software development
- Work on improving Open Source Software (OSS) ecosystem on Arm servers
- Focused area
  - Storage
  - Data Science
Agenda

- SPDK and Kubernetes Storage overview
- Container Storage Interface (CSI)
  - CSI Internals
  - Kubernetes CSI support
- SPDK-CSI Implementation details
- Community
  - Contribution guidelines
  - Project status and plans
SPDK, Kubernetes Storage
What is SPDK

- Quoted from https://spdk.io/
  - The Storage Performance Development Kit (SPDK) provides a set of tools and libraries for writing high performance, scalable, user-mode storage applications.

- Key techniques
  - Interact with hardware directly in user space
  - Polling data readiness instead of interrupt
  - No locks in I/O path
What is SPDK

**SPDK ARCHITECTURE**

**Block Storage Protocols**
- Networking: NVMe-oF (RDMA, TCP, FC), iSCSI
- Virtualization: vhost-scsi, vhost-blk

**File Storage Services**
- Filesystems: BlobFS

**Block Storage Services**
- Partitioning: Logical Volumes, GPT
- Caching: OCF
- Host FTL: Open Channel
- Pooling: RAID-0
- Transforms: Crypto, Compression

**Block Storage Providers**
- NVMe, io_uring, Linux AIO, virtio, iSCSI, Ceph RBD

**Drivers**
- NVMe (PCIe, RDMA, TCP), virtio (scsi, blk)
SPDK Network Storage

- Network storage protocols
  - SPDK implements NVMe-oF and iSCSI targets above its block device layer

- Volume management
  - SPDK Logical Volume provides flexible block device interface to local applications and network targets

- Remote configuration
  - SPDK supports JSON-RPC for remote configuration of Logical Volumes and NVMe-oF/iSCSI targets
Kubernetes Storage

- Kubernetes volume driver: a brief history
  - In-Tree: storage driver coupled in Kubernetes code base. Deprecated, legacy code will be removed.
  - FlexVolume: exec based API for volume plugins. Hard to deploy and manage dependency.
  - Container Storage Interface (CSI): Addresses pains of In-Tree and FlexVolume. Standardizes storage system integration with Kubernetes.
    - Kubernetes CSI Drivers List
Kubernetes Storage

- Dynamic volume management
  - Storage Class
  - Persistent Volume (PV)
  - Persistent Volume Claim (PVC)

- How dynamic volume provisioning works
  - Pod claims block storage from a storage class (CSI driver)
  - CSI controller driver creates the block device through storage provider (maybe local or on the cloud)
  - CSI node driver mounts the block device to Pod
Dynamic Volume Provisioning

kind: StorageClass
metadata:
name: my-sc
provisioner: my-csi-driver

kind: PersistentVolumeClaim
metadata:
name: my-pvc
spec:
resources:
requests:
storage: 1Gi
storageClassName: my-sc

kind: Pod
spec:
......
volumes:
- name: spdk-volume
  persistentVolumeClaim:
    claimName: my-pvc

kind: PersistentVolume
apiVersion: v1
kind: PersistentVolume
metadata:
name: my-pvc

kind: StorageClass
metadata:
name: my-sc
provisioner: my-csi-driver

kind: CSI
Node
Driver

kind: CSI
Controller
Driver

Provision volume

Mount to Pod

Mount volume to Node

Mount to Pod

CSI
Node
Driver

Kubernetes Cluster

Storage Provider

SPDK

2020 Storage Developer Conference. © Arm Limited. All Rights Reserved.
SPDK-CSI: Bring SPDK to Kubernetes

- **Goals**
  - Bring SPDK to Kubernetes storage through NVMe-oF, iSCSI
  - Supports dynamic volume provisioning
  - Enables Pods to use SPDK for transient or persistent storage

- **Non-goals**
  - Not a complete storage solution
  - SPDK services management is out of the scope
Container Storage Interface (CSI)
Container Storage Interface (CSI)

- From [Kubernetes CSI Developer Documentation](#)
  - The **Container Storage Interface** (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes.
  - NOTE: CSI is a general protocol, not for Kubernetes only.

- My point of view
  - Defines a bunch of messages (RPC) between CO and third-party storage driver
  - Volume lifecycle management based on these messages
## Container Storage Example

<table>
<thead>
<tr>
<th>No.</th>
<th>Steps</th>
<th>Run command at</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>App starts</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Create RBD volume in Ceph cluster through Ceph API</td>
<td>Any host can access Ceph</td>
</tr>
<tr>
<td>2</td>
<td>Mount/Format RBD volume to some directory on host</td>
<td>Host where the app runs</td>
</tr>
<tr>
<td>3</td>
<td>Mount host RBD directory to container directory</td>
<td>Host where the app runs</td>
</tr>
<tr>
<td></td>
<td><strong>App stops</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Unmount container directory</td>
<td>Host where the app runs</td>
</tr>
<tr>
<td>5</td>
<td>Unmount host RBD directory</td>
<td>Host where the app runs</td>
</tr>
<tr>
<td>6</td>
<td>Delete RBD volume in Ceph cluster through Ceph API</td>
<td>Any host can access Ceph</td>
</tr>
</tbody>
</table>
## Container Storage Example

### To automate above steps in a container cloud

<table>
<thead>
<tr>
<th>What we need</th>
<th>In CSI Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Step 1, 6] A storage driver to handle Ceph API and create/delete RBD on demand. It can run on any host which has access to Ceph cluster control plane.</td>
<td>Controller Driver</td>
</tr>
<tr>
<td>[Step 2, 3, 4, 5] A storage driver to (un)mount Ceph RBD volumes. It must run on all hosts where containerized app may be scheduled.</td>
<td>Node Driver</td>
</tr>
<tr>
<td>A protocol to define messages between CO and the plugin, so they can cooperate to finish the job.</td>
<td>RPC and Volume Lifecycle</td>
</tr>
</tbody>
</table>
CSI Drivers

- **Controller Driver**
  - Talk to Service Provider (SP) to create/delete volumes

- **Node Driver**
  - Mount/unmount remote volumes to local host

- Controller driver on CO master node
- Node driver instances per CO worker
- CO talks to CSI Drivers with CSI RPC messages
# Key CSI RPCs

<table>
<thead>
<tr>
<th>RPC</th>
<th>Explains</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CO → Controller Driver</strong></td>
<td></td>
</tr>
<tr>
<td>CreateVolume</td>
<td>Create a volume with specific parameters in storage provider</td>
</tr>
<tr>
<td>DeleteVolume</td>
<td>Revert creating</td>
</tr>
<tr>
<td>ControllerPublishVolume</td>
<td>Expose the volume to be accessible from worker node</td>
</tr>
<tr>
<td>ControllerUnpublishVolume</td>
<td>Revert publishing</td>
</tr>
<tr>
<td><strong>CO → Node Driver</strong></td>
<td></td>
</tr>
<tr>
<td>NodeStageVolume</td>
<td>Import remote volume and mount to worker node host</td>
</tr>
<tr>
<td>NodeUnstageVolume</td>
<td>Revert staging</td>
</tr>
<tr>
<td>NodePublishVolume</td>
<td>Bind mount host staging directory to container internal directory</td>
</tr>
<tr>
<td>NodeUnpublishVolume</td>
<td>Revert publishing</td>
</tr>
</tbody>
</table>
Volume Lifecycle

CreateVolume +---------+ DeleteVolume
+---------x| CREATED +---------+
| +--------^--+
| Controller | Controller v
++ Publish | Unpublish +++
|X| Volume | Volume |
++ +-----V-----++
| NODE_READY |
+--------^--+
Node | Node
Stage | Unstage
Volume | Volume
+-----V-----+
| VOL_READY |
+--------^--+
Node | Node
Publish | Unpublish
Volume | Volume
+-----V-----+
| PUBLISHED |
+----------+

From CSI Spec

Figure 6: The lifecycle of a dynamically provisioned volume, from creation to destruction, when the Node Plugin advertises the STAGE_UNSTAGE_VOLUME capability.
Kubernetes CSI Support

- Kubernetes supports CSI well
  - CSI spec 1.0 supported since Kubernetes 1.13

- Common practice
  - Wrap Controller and Node driver in a single binary
    - Select functionality per command line parameters
  - Deploy Controller driver as Deployment or StatefulSet
  - Deploy Node driver as DaemonSet
    - Exactly one instance on each worker node
  - Leverage CSI Sidecar containers to reduce boilerplate code
Kubernetes CSI Support

- Kubernetes Sidecar Containers
  - Watch Kubernetes objects and send RPC to CSI drivers
  - Free CSI drivers from talking directly to API server

<table>
<thead>
<tr>
<th>Sidecar</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Provisioner</td>
<td>Watches for PersistentVolumeClaim objects and triggers [Create</td>
</tr>
<tr>
<td>External Attacher</td>
<td>Watches for VolumeAttachment objects and triggers Controller[Publish</td>
</tr>
<tr>
<td>Node Driver Registrar</td>
<td>Registers the CSI driver with Kubelet to receive Node[Stage</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>
Kubernetes CSI Support

CSI Controller Pod
- CSI Controller Driver
- External Provisioner
- External Attacher
- Unix Socket
- CreateVolume
- DeleteVolume
- ControllerPublishVolume
- ControllerUnpublishVolume

CSI Node Pod
- CSI Node Driver
- Driver Registrar
- Unix Socket
- NodeStageVolume
- NodeUnstageVolume
- NodePublishVolume
- NodeUnpublishVolume

Master Node
- Kubelet
- VolumeAttachment
- Persistent Volume Claim

Worker Nodes
- API Server
- NodeStageVolume
- NodeUnstageVolume
- NodePublishVolume
- NodeUnpublishVolume
SPDK-CSI Overview

Kubernetes Node

- **CSI Node Pod**
  - CSI Node Driver
  - Driver Registrar

- **Work Pod**
  - Nginx

- **Kubelet**

Kubernetes Node

- **CSI Controller Pod**
  - CSI Controller Driver
  - External Provisioner
  - External Attacher

Storage Nodes

- **Control Plane**
  - RDMA, TCP
  - NVMe-oF Target

- **Data Plane**

Network

- API Server
- Volume Plugin

SPDK

© Arm Limited. All Rights Reserved.
SPDK-CSI Controller Driver

Controller configures SPDK network target through JSON-RPC

<table>
<thead>
<tr>
<th>CSI Message</th>
<th>JSON-RPC (NVMf)</th>
<th>JSON-RPC (iSCSI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateVolume</td>
<td>bdev_lvol_create</td>
<td>bdev_lvol_create</td>
</tr>
<tr>
<td>DeleteVolume</td>
<td>bdev_lvol_delete</td>
<td>bdev_lvol_delete</td>
</tr>
<tr>
<td>ControllerPublishVolume</td>
<td>nvmf_subsystem_add_ns</td>
<td>iscsi_create_portal_group</td>
</tr>
<tr>
<td></td>
<td>nvmf_subsystem_add_listener</td>
<td>iscsi_create_initiator_group</td>
</tr>
<tr>
<td>ControllerUnpublishVolume</td>
<td>nvmf_subsystem_remove_ns</td>
<td>iscsi_create_target_node</td>
</tr>
</tbody>
</table>
SPDK-CSI Node Driver

Node connects to SPDK target and mounts remote volume

<table>
<thead>
<tr>
<th>CSI Message</th>
<th>Node (NVMf)</th>
<th>Node (iSCSI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>StageVolume</td>
<td>nvme connect -n \textit{nqn} -a \textit{ip} -s \textit{port} ...</td>
<td>iscsiadm -p \textit{ip:port} -m discovery ... iscsiadm -T \textit{iqn} -p \textit{ip:port} --login ... mount /dev/disk/by-id/\textit{diskid} stagePath</td>
</tr>
<tr>
<td></td>
<td>mount /dev/disk/by-id/\textit{diskid} stagePath</td>
<td>mount /dev/disk/by-id/\textit{diskid} stagePath</td>
</tr>
<tr>
<td>UnstageVolume</td>
<td>nvme disconnect -n \textit{nqn}</td>
<td>iscsiadm -T \textit{iqn} -p \textit{ip:port} --logout ... umount stagePath</td>
</tr>
<tr>
<td></td>
<td>umount stagePath</td>
<td>umount stagePath</td>
</tr>
<tr>
<td>PublishVolume</td>
<td>mount -o bind stagePath \textit{podPath}</td>
<td>mount -o bind stagePath \textit{podPath}</td>
</tr>
<tr>
<td>UnpublishVolume</td>
<td>umount \textit{podPath}</td>
<td>umount \textit{podPath}</td>
</tr>
</tbody>
</table>

* “\textit{nqn, ip, port, diskid, iqn}” are passed from Controller Driver*
Sequence Diagram

Create and Attach Volume

1. User
2. Deploy App
3. CreateVolume
   - bdev_lvol_create
   - nvmf_create_subsystem
4. ControllerPublishVolume
   - nvmf_subsystem_add_ns
   - nvmf_subsystem_add_listener
5. NodeStageVolume
6. NodePublishVolume
7. Kubernetes Cluster
   - SPDK-CSI Controller Driver
   - SPDK-CSI Node Driver
8. SPDK Storage Pool
   - SPDK JSON RPC Service
9. Create Logical Volume
   - Create NVMF subsystem
10. Done

Teardown Volume

1. User
2. Teardown App
3. NodeUnpublishVolume
4. NodeUnstageVolume
5. ControllerPublishVolume
   - nvmf_subsystem_remove_ns
6. DeleteVolume
   - nvmf_delete_subsystem
   - bdev_lvol_delete
7. Unmount Pod dir
8. Unmount staging dir
9. nvmf_disconnect
10. Remove nas
11. Delete Logical Volume
12. Done

Kubernetes Cluster
- SPDK-CSI Controller Driver
- SPDK-CSI Node Driver
- SPDK JSON RPC Service

SPDK Storage Pool
- SPDK JSON RPC Service
SPDK-CSI Deployment

- Deploy Kubernetes cluster
  - Minikube is convenient for development purpose
- Deploy SPDK service
- Deploy SPDK-CSI components
  - Controller, Node, StorageClass, RBAC, etc.
- Validate SPDK-CSI driver with End to End testing
  - Standard way to test third party CSI drivers in Kubernetes
- Too many details, see source code 😊
Some Tips

- **Idempotency and concurrency**
  - Handle duplicated and out of order messages, e.g.,
    - CreateVolume may come again even if volume already created
    - UnpublishVolume may come after DeleteVolume
  - Take care of concurrency, e.g.,
    - Two DeleteVolume messages try to delete same volume at same time

- **Keep it simple**
  - Leverage Kubernetes declarative model in error handling
  - Suppress Controller(Un)PublishVolume messages
    - Do everything in CreateVolume/DeleteVolume handlers
Community
Welcome Contribution

- Code review at SPDK Gerrit
  - `git clone https://review.spdk.io/spdk/spdk-csi`
  - Github mirror: [https://github.com/spdk/spdk-csi](https://github.com/spdk/spdk-csi)

- Development Guidelines
  - [https://spdk.io/development/](https://spdk.io/development/)

- Trello Board
  - [https://trello.com/b/nBujJzya/kubernetes-integration](https://trello.com/b/nBujJzya/kubernetes-integration)
Project Status and Plan

- Status: Alpha
  - Mandatory CSI functionalities are ready

- Plans:
  - Tests and improvements for production level quality
  - New features
    - Topology, volume expansion, snapshot, etc.
    - See Backlogs and Todos at [Trello Board](https://trello.com)
  - Integration with [Rook](https://rook.io)
    - Build a total solution of leveraging SPDK in Kubernetes
References

- Container Storage Interface (CSI) Spec
  - https://github.com/container-storage-interface/spec/
- Kubernetes CSI Documentation
  - https://kubernetes-csi.github.io/docs/
- SPDK JSON-RPC
- SPDK-CSI Design Document
  - https://tinyurl.com/spdkcsi-design-doc
Please take a moment to rate this session.

Your feedback matters to us.