
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Improving Longhorn Performance With SPDK

Keith Lucas
Principal Software Engineer, Oracle Labs

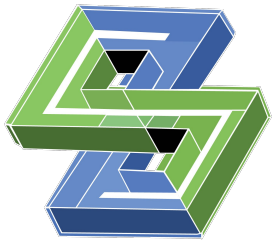
David Ko
Senior Engineering Manager, SUSE

Agenda

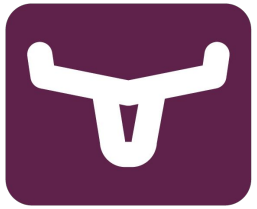
- What is Longhorn
- How Current Longhorn Works
- What Challenges Current Longhorn Has

- What is SPDK
- How New Longhorn Works
- What Benefits New Longhorn Introduces

- Preliminary Performance Benchmark
- Further Areas for Improvement
- What Is Next? Longhorn SPDK



SPDK



LONGHORN

What is Longhorn

- Highly available, software-defined persistent block storage for Kubernetes
- Lightweight, reliable, and easy-to-use
- Adds persistent volume support to any certified K8s cluster
- Storage Agnostic – any ext4/xfs filesystem can be added to a Longhorn cluster
- NFS and S3 compatible (backup storage)
- Kubernetes-first design implemented in CRDs and controller pattern
- Open source and owned by the CNCF

How Current Longhorn Works

Volume Elements

- Volume Frontend (iSCSI)
- Volume (Engine)
- Volume replica (Replica)

Volume Lifecycle

- CSI
- PVC/PV

Data Placement

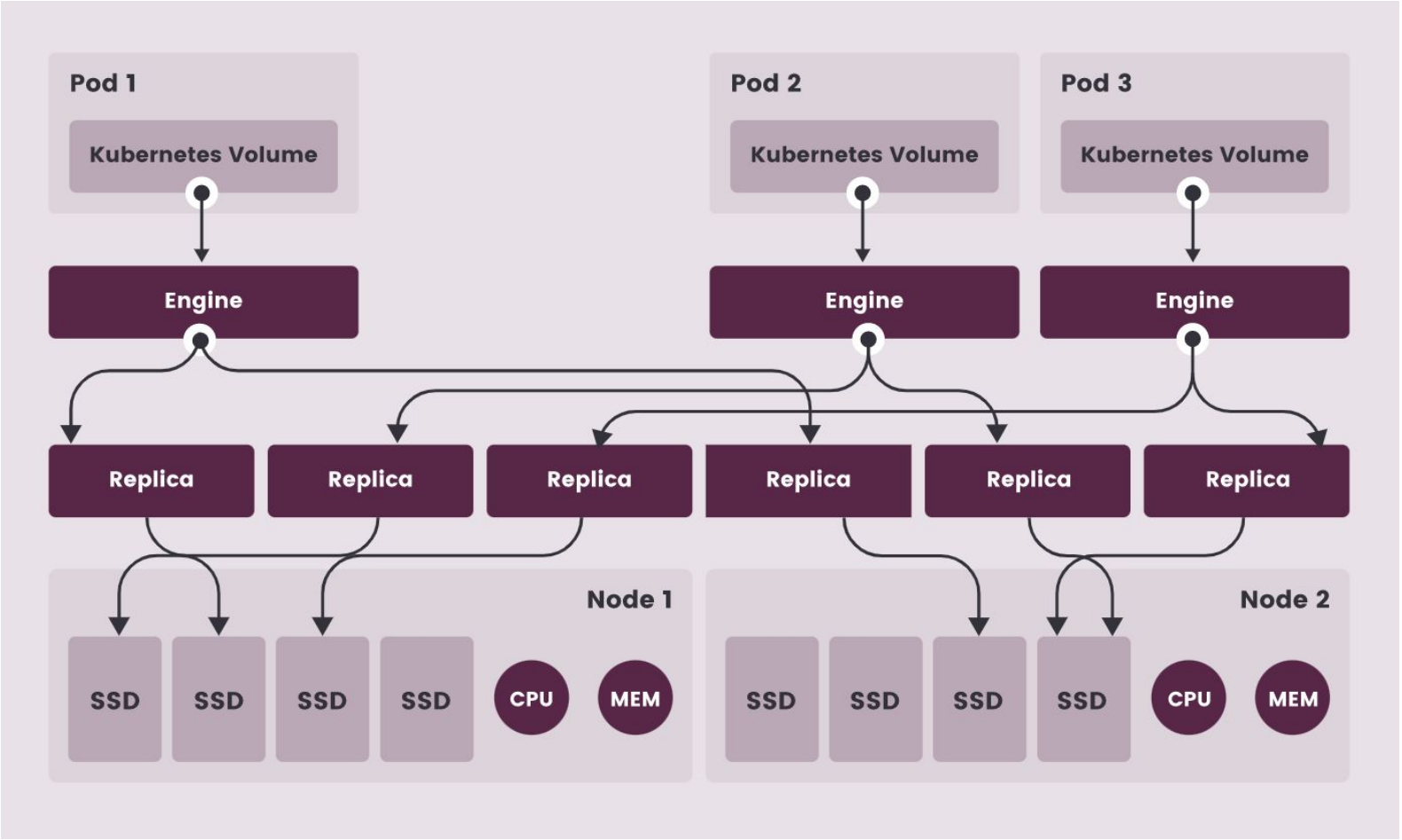
- Longhorn disk (FS on host)

Deployment

- Segregated Microservice

Control Plane

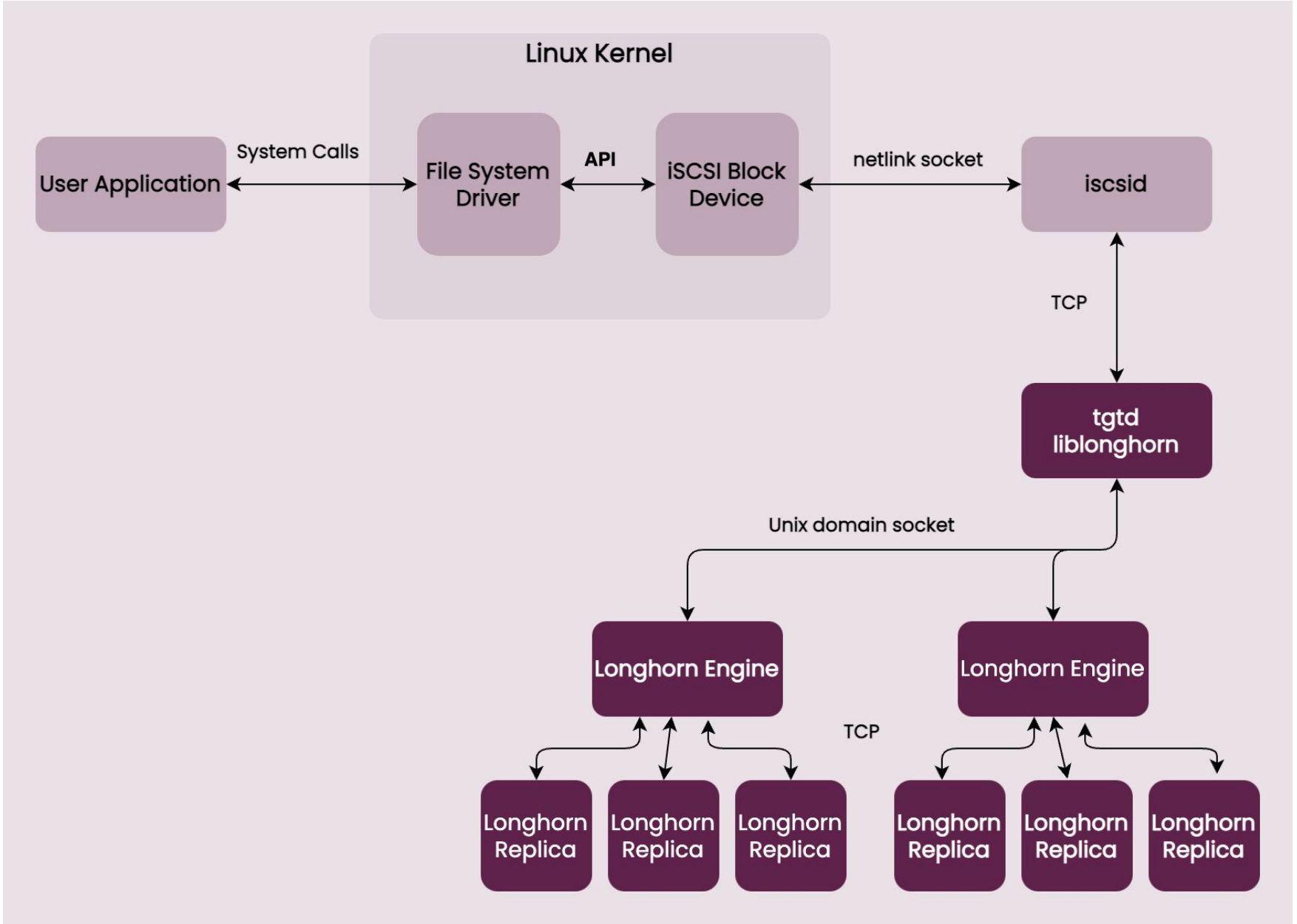
- Kubernetes Controller + CR



Longhorn Engine & Replica

- Volume Frontend (iSCSI)
  - open-iscsi (iSCSI initiator)
- Volume (Engine)
  - longhorn tgtd (iSCSI target)
  - longhorn engine (TCP data server and volume controller)
- Volume replica (Replica)
  - longhorn replica (local, remote TCP data server and replica controller)
  - Data operation (snapshot, rebuild, coalesce/merge, prune, purge, backup, etc)

Longhorn Engine & Replica - Data Path



What Challenges Current Longhorn Has

- Too many components
  - iscsid, longhorn tgtd and longhorn engine process for each volume
- Longhorn engine (data plane) & tgtd, 2 hops extra cost
  - Method allocation and garbage collection for each I/O operation
- I/O model limitation, difficult to change the implementation
  - I/O model is based on filesystem sparse files opened with O_DIRECT, blocking read and write system calls.
  - Significant changes would be needed for newer I/O methods in Linux like libaio or io_uring
  - Lack of library support for these methods in Go.
- Number of processes increases with each block device

What is SPDK

SPDK - Software Performance Development Kit

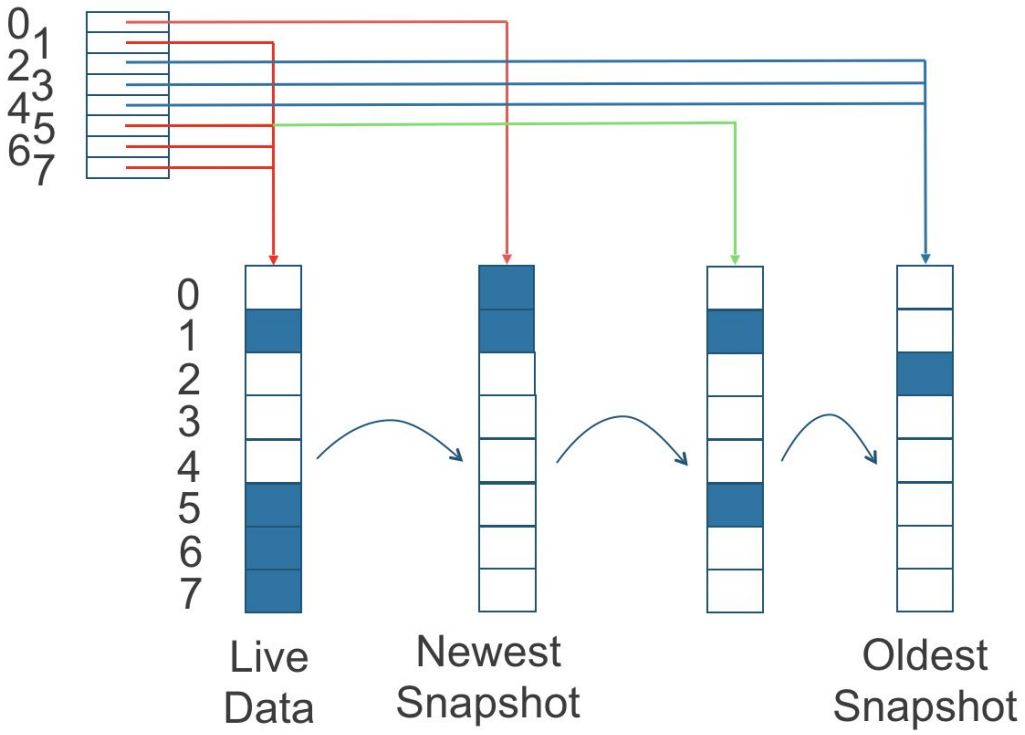
- Used in high performance cloud applications
- Has a generic “block device” application layer with many different implementations, easy to implement new block devices
- Has support for exposing block devices over several widely supported network protocols for remote block devices: iSCSI and NVMe over Fabrics
- Has a logical volume feature which stores data in a series of sparse snapshots
- Designed for asynchronous programming
- Uses memory pools to minimize memory allocation

Longhorn Sparse Provisioning vs SPDK Logical Volumes

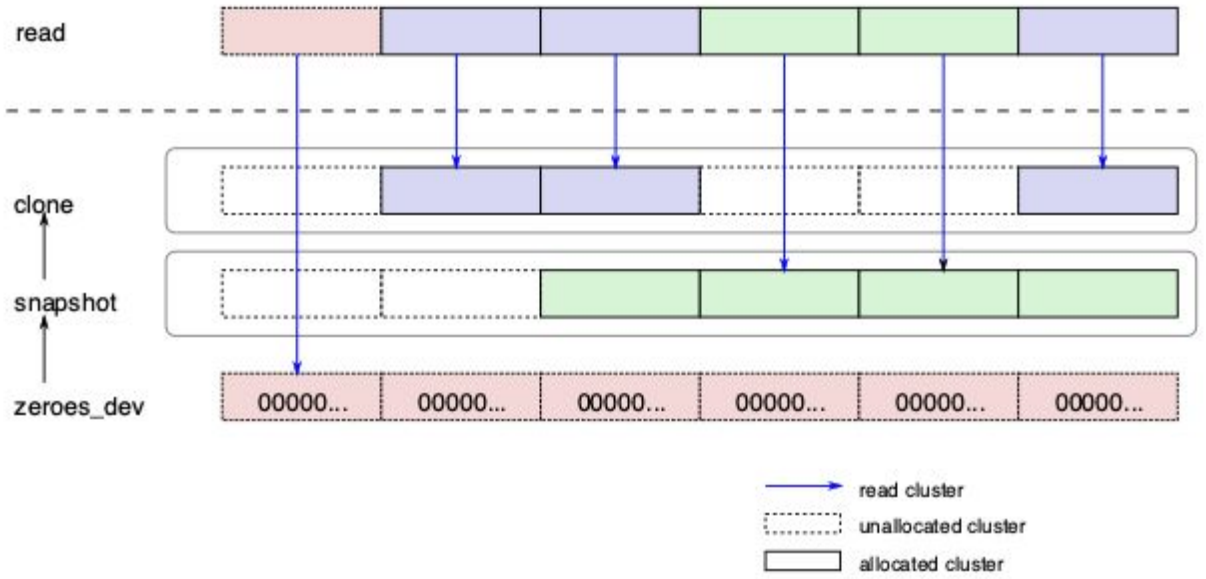
- Longhorn uses sparse files for thinly provisioning volumes and for snapshots
  - A hierarchy of sparse snapshot files represent a volume
  - Each volume is in a separate directory on the file system
- SPDK has logical volumes
  - One SPDK block device can host multiple logical volume block device
  - Each logical volume is also a hierarchy of sparse “blobs”

Longhorn Sparse Provisioning vs SPDK Logical Volumes

Longhorn Sparse Provisioning



SPDK Logical Volumes



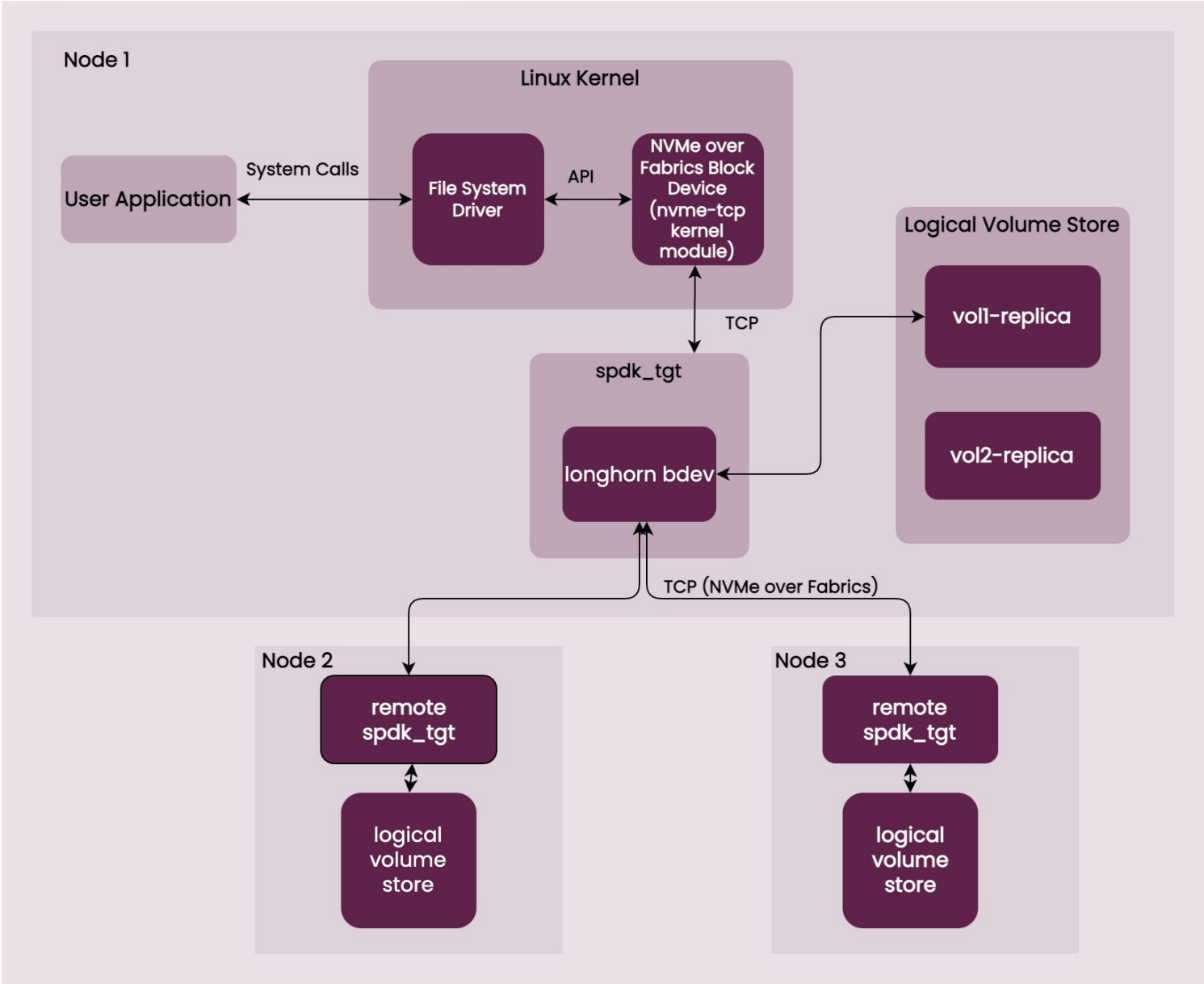
SPDK's Asynchronous Programming Model

- API functions don't block
- API is asynchronous and event driven
  - Schedule an operation to be performed
  - Main loop checks status of all pending operations continuously
  - Notify via a callback when the operation is complete
- Achieves better performance than traditional blocking I/O

How New Longhorn Works - Longhorn with SPDK

- Use SPDK's NVMe over Fabrics for Longhorn's block device instead of iSCSI.
  - Linux has mature NVMe over Fabrics device support
  - The Linux kernel has direct NVMe over Fabrics support without the need for an iscsid equivalent.
- Implement the core Longhorn engine functionality as a custom "block dev" in SPDK.
  - Consists of a set of replicas which are local logical volume bdevs or remote NVMe over Fabrics logical volume bdevs (these are akin to current longhorn replicas)
  - Distributes all write operations to a replicas.
  - Chooses one of the replicas for read operations.
  - Supports snapshots and rebuilding new volumes when added.
- One SPDK process per Kubernetes node. Each SPDK process can handle multiple block devices.

How New Longhorn Works - Longhorn Components with SPDK



Preliminary Performance Benchmark

- Test Environment
  - CPU: 1 x AMD EPYC 7402P 24-Core Processor
  - RAM: 64GB
  - DISK: 2 x 240GB SSD, 2 x 480GB SSD
  - NIC: 2 x 10Gbps Bonded Ports

- Test Methodology
  - Uses the Longhorn developed kbench utility
  - Uses the fio command to test IOPS, bandwidth, and latency
  - Tested single node using raw disk, the existing Longhorn, and Longhorn with SPDK
  - Tested three nodes with the existing Longhorn and Longhorn with SPDK

Single Node Performance Comparison

Disk

	Read	Write
IOPS Random	88,298	66,998
IOPS Sequential	107,598	89,425
Bandwidth Random I/O	545,822 KiB/sec	506,418 KiB/sec
Bandwidth Sequential I/O	552,562 KiB/sec	517,037 KiB/sec
Latency Random I/O	102,024 ns	35,664 ns
Latency Sequential I/O	30,674 ns	32,998 ns

Current Longhorn

	Read	Write
IOPS Random	41,710	15,557
IOPS Sequential	62,332	27,707
Bandwidth Random I/O	500,057 KiB/sec	360,538 KiB/sec
Bandwidth Sequential I/O	494,365 KiB/sec	397,190 KiB/sec
Latency Random I/O	266,264 ns	248,349 ns
Latency Sequential I/O	224,166 ns	254,604 ns

Longhorn with SPDK

	Read	Write
IOPS Random	67,700	66,964
IOPS Sequential	97,954	82,704
Bandwidth Random I/O	544,774 KiB/sec	506,642 KiB/sec
Bandwidth Sequential I/O	552,061 KiB/sec	510,585 KiB/sec
Latency Random I/O	128,691 ns	60,968 ns
Latency Sequential I/O	57,724 ns	56,785 ns

Single Node Performance Key Points

- Single Node performance with SPDK improves in all categories
- Bandwidth with SPDK close to Disk
- Overall latency with SPDK compared to disk is 25-30 µs versus over 100 µs with Longhorn.

Three Node Performance Comparison

Existing Longhorn

	Read	Write
IOPS Random	50,022	8,377
IOPS Sequential	77,361	22,717
Bandwidth Random I/O	1,145,030 KiB/sec	278,256 KiB/sec
Bandwidth Sequential I/O	652,087 KiB/sec	355,622 KiB/sec
Latency Random I/O	269,326 ns	254,559 ns
Latency Sequential I/O	227,172 ns	253,791 ns

Longhorn with SPDK

	Read	Write
IOPS Random	105,532	66,964
IOPS Sequential	131,949	82,704
Bandwidth Random I/O	1,272,529 KiB/sec	365,686 KiB/sec
Bandwidth Sequential I/O	1,274,302 KiB/sec	366,601 KiB/sec
Latency Random I/O	194,131 ns	169,507 ns
Latency Sequential I/O	190,013 ns	168,261 ns

Three Node Performance Key Points

- SPDK also performs better on a three node scenario over the existing Longhorn Engine
- Latency is worse than single node in the three node scenario because of networking
- Read operations and bandwidth are improved because reads are distributed among the three nodes

Further Areas for Improvement

- User space NVMe support
  - Access NVMe drives from user space driver instead of Linux kernel driver
  - SPDK's claim to fame
  - Initially didn't support this because we wanted to support any kind of disk
- io_uring and other new I/O technologies used in SPDK can be tried easily
- RDMA - A different network transport for NVMe over Fabrics using special NIC drivers

What is Next? Longhorn SPDK

- Longhorn SPDK changes the how the rest of Longhorn will work
  - Change from multiple Longhorn engine processes per node to one Longhorn SPDK process
    - Eliminate Instance Manager which manages all these processes
    - Longhorn CRDs reflect current process model and need update
    - Remove iSCSI and tgt
- New communication mechanism between Longhorn Manager and SPDK
- Develop backup and restore functionality
- Working on technology preview of Longhorn integrated with SPDK

Thank You 🙏

Q & A 🙋

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above to
leave feedback on this session