Let's decompose storage (again)

Why? How? Huh?

MSST May 17 2017

Evan Powell
What's new?

Development Model
- Waterfall
- Agile
- DevOps

Application Architecture
- Monolithic
- N-tier
- Microservices

Deployment & Packaging
- Bare Metal
- Virtual Servers
- Containers

Application Infrastructure
- Data Center
- Hosted
- Hybrid Cloud

Storage
- Scale Up
- Scale Out
- Storage as a Service

http://www.slideshare.net/ColleenCorrice/persistent-storage-for-containerized-applications
Layering

CERTIFIED SYSADMIN(S)

Supervised Provisioning
Manage Storage Upgrades!
Manage 100s of Volumes
Managed Upgrades

NFS, iSCSI

Enterprise Storage
(SAN, NAS, ScaleOut,
DP, DR, Backup,
Compression, Dedup)

POSIX
(IO, Checksum, Snapshots)

Storage Medium
(Disk, Flash, RAM)

Format Disks and Use.
Manage Few Disks

(3-TIER APPs)

(LOCAL APPs)

GEEK
# Layering

<table>
<thead>
<tr>
<th>(MICRO APPS)</th>
<th>(CLOUD SERVICE)</th>
<th>(3-TIER APPs)</th>
<th>(LOCAL APPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEVOPS AT SPEED</strong></td>
<td><strong>PAAS ADMINs</strong></td>
<td><strong>CERTIFIED SYSADMIN(S)</strong></td>
<td><strong>GEEK</strong></td>
</tr>
<tr>
<td>Auto-magically Provisioned</td>
<td>Auto-Provisioned</td>
<td>Supervised Provisioning</td>
<td>Format Disks and Use.</td>
</tr>
<tr>
<td>Auto-managed Storage (ML)</td>
<td>Delegated Administration</td>
<td>Manage Storage Upgrades!</td>
<td>Manage Few Disks</td>
</tr>
<tr>
<td>Data Mobility</td>
<td>Manage Shared Infra</td>
<td>Manage 100s of Volumes</td>
<td>Manage Data Mobility</td>
</tr>
<tr>
<td>Manage Millions of Volumes</td>
<td>Manage 1000s of Volumes</td>
<td>Managed Upgrades</td>
<td>Seamless Upgrades</td>
</tr>
<tr>
<td>Seamless Upgrades</td>
<td>Scheduled Upgrades</td>
<td>Auto-magically Provisioned</td>
<td>Auto-managed Storage (ML)</td>
</tr>
<tr>
<td><strong>OpenEBS</strong></td>
<td><strong>OpenStack Cinder, VASA</strong></td>
<td><strong>NFS, iSCSI</strong></td>
<td><strong>XFS, ZFS, LVM, EXT4</strong></td>
</tr>
<tr>
<td>Containerized Storage (Hyper-Converged, Auto-Scaling, Auto-Upgradable, Yaml Driven)</td>
<td>Cloud Storage - SDS (Public, Private, Hybrid)</td>
<td>Enterprise Storage (SAN, NAS, ScaleOut, DP, DR, Backup, Compression, Dedup)</td>
<td>POSIX (IO, Checksum, Snapshots)</td>
</tr>
<tr>
<td><strong>OpenSource Technology Stack</strong></td>
<td></td>
<td></td>
<td><strong>Storage Medium (Disk, Flash, RAM)</strong></td>
</tr>
</tbody>
</table>
Happy days!

Manifests express intent stateless.
K8S used rarely for apps requiring persistence because they require brittle tight coupling.

Hard wired connections via plug-in

NAS  SAN  S3  NAS  SAN  S3
No changes to DevOps workflow even for containers requiring persistence. Users manifest their intent and the storage and storage controllers adjust automatically as needed.

Containers and underlying storage, local on host or dedicated storage pods OR remote S3 or EBS storage all grouped into a storage cloud that just works.
Architecture and Design

- Powered by Linux, Go and OpenSource
- Built and Delivered as Containers / Micro-services
- Longhorn, Gotgt, Kubernetes, Consul
## Design Goals and Constraints

<table>
<thead>
<tr>
<th>Goal</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault tolerant and secure by default</td>
<td>Developer and Operators Friendly</td>
</tr>
<tr>
<td>Low entry barrier, easy to setup</td>
<td>Completely OpenSource (Apache license)</td>
</tr>
<tr>
<td>Storage optimized for Containerized Applications</td>
<td>Microservices based</td>
</tr>
<tr>
<td>Horizontally scalable to millions of Containers</td>
<td>DevOps architecture</td>
</tr>
<tr>
<td>Seamless integration into existing private and public cloud environments</td>
<td></td>
</tr>
<tr>
<td>Non-disruptive upgrades</td>
<td></td>
</tr>
</tbody>
</table>
Overview & Terminology

K8s master

Storage Driver

HTTPS (manage)

OpenEBS Maya master

K8s minions

Network (Flannel)

Network (Flannel*)

Pod

Data

OpenEBS VSMs / Storage Pod

Local Storage

Remote Storage

OpenEBS Storage Hosts
Deployment - Hyper-Converged

K8s master
OpenEBS
Maya-K8s Adaptors

Network (Flannel)

Pod
TCMU
TCP

Storage Pods (3)

K8s minions
OpenEBS Maya Storage
Orchestrator
VSM - Storage in Containers

VSM / Storage Pod

OpenEBS Storage Hosts

Storage

- Local Disks
- NAS or SAN
- Cloud Storage

Data (iSCSI/TCMU)

- Frontend Containers
- Inline Replication
- Backend Containers to Persist Data (Cached, Protected)
- Container (Docker)
- Maya Storage Orchestrator
- NVMe Flash
- Multiple Storage Backends
Jiva - Containerized Storage Image

Container Image
https://hub.docker.com/r/openebs/jiva/

- qorum-net
- fapi-server
- bapi-server
- syncer
- snap-qcow
- snap-s3
- snap control
- snap r/w
- snap-local
- key-vault
- stats-db
- key-vault
- fe-iscsi
- fe-tcmu
- IO Processor
- Replication/Multiplex
- meta-cache
- QoS
- IO Store
- TTA
- Hot Data Deduplication
- Compression/Encryption
- Backend Store
- QoS
- NVF RWC
- TTA Pull/Push
- OpenEBS Container runtime (Maya)
Maya - Container Storage Orchestration

Integrations

OpenEBS Maya Master

OpenEBS Storage Host

Integrations

OpenEBS Maya Master

OpenEBS Storage Host

Integrations
Storage Internals

- Capacity Management
- QoS
- Access - iSCSI, TCMU
- Snapshot / Restore (S3)
- Backup / Migration
- Caching/Tiering
- Replication / Rebuild
OpenEBS - Core differentiations

- The block storage software is made into a micro service
- The ‘micro service’ has its own block protocol stack, tiering engine, QoS engine and ML prediction capability
- The block storage knowledge is maintained on a per-volume basis. The data of each volume is divided into cold-data and hot-data. Cold-data resides on NVMe-Flash or on 3DX-Memory. Hot-data resides in slower disks / SAN/ Cloud-Storage/S3
- The metadata knowledge also is maintained at a volume level (not the entire storage). This saves us from the issue of huge-metadata-sifting at scale. The traversal through meta-data depends on the “size of the volume” and not on the “number of volumes”.
- Within the volume the meta data is not managed at “block-level” but at “chunk-level”. Typical block-size is 4KB and Typical chunk-size is 4 MB. This results in the huge reduction of metadata size of the block-volume that needs to be maintained.
- Checksum - One of the important metadata is checksum. OpenEBS guarantees bit-rot protection through the use of checksums. The checksums are managed at a chunk level only on Cold-Data. The checksums are not managed on hot-data, the blocks go in and out of chunks on the hot-data without the need of checksum calculation on the fly.
- Deduplication-while-tiering: Deduplication has capacity benefits but kills performance (either inline or offline). But in OpenEBS, we do this while moving the data from hot-to-cold tiers. In effect, the benefits of deduplication without the performance penalty.
Cchchchcunking

Data block to chunkbook mapper

lun1.1.metabook  lun1.2.metabook

lun chunkbooks

Blocks to chunks (coalescing)

lun1 data chunks  lun2 data chunks  lun3 data chunks

Data blocks in NVRAM (NVDIMM)

Read - uncompress  Write Dedup and compress

Data in nvme Flash

Nvme - RAID

XFS

LVM RAID

Kernel

User land
OpenEBS - Metadata at scale is not an issue

The volume IO processing has to deal with the global metadata of 8TB

Metadata of the volume is managed at chunk level

The volume IO processing has to deal with the volume metadata of 100GB
Storage Data format

Meta table of the lun (Fixed size)

Two dimensional array indexed chunk number.

Chunk number (computed with block byte range)
- Location (fast memory/chunk)
- A
- B
- C
- D
- ?

Chunk table1 (Fixed size)

Chunk table2 (Fixed size)

Block layout of the lun inside the xfs file

/xfs/outside/lun1 (sparse file)
Storage Interface

- HardDisks
- SAS/SATA Flash
- NVMe Flash
- PCIe Flash
- S3
- Cloud Block Storage
VSM Network Interface

- Host Networking
- VLANs / IPSpaces
Ease of Configuration

- VSM Configuration Spec
- Infra Spec
- Integrate into K8s / EBS Compatible
Integration to Orchestration

Options to consume the storage by containers:

- iSCSI Driver (Pre-provisioned)
- Maya Volume Driver
- Integrated Orchestration
Storage Connectivity - iSCSI Claims

K8s master

- iSCSI Driver
- HTTPS (manage)

OpenEBS Maya master

- OpenEBS Storage Hosts
- Local Storage
- Remote Storage

K8s minions

- Network (Flannel)
- Network (Flannel*)

Pod

- OpenEBS VSMs / Storage Pod

iSCSI Connectivity - iSCSI Claims

K8s master

- iSCSI Driver
- HTTPS (manage)
Storage Connectivity - Maya Driver

K8s master

Maya Driver

HTTPS (manage)

OpenEBS Maya master

K8s minions

Network (Flannel)

Network (Flannel*)

Pod

iSCSI

OpenEBS VSMs / Storage Pod

Local Storage

Remote Storage

OpenEBS Storage Hosts
Storage Connectivity - Shared Orchestration

K8s master → Maya Driver

HTTPS (manage)

OpenEBS master

K8s minions

Network (Flannel)

Pod

TCMU

TCP

OpenEBS VSMs / Storage Pod

OpenEBS Storage Hosts

Local Storage

Remote Storage
Resiliency and Fault Tolerance

- Scaleout
- Blue-Green Upgrades - Infra
- Rolling Upgrades - VSMs
- High-Availability
Security

- Data Security
- Encryption
- Secure Delete
Telemetry

- Monitoring and Troubleshooting
- Analytics
Performance

- IO Latency
- Provisioning
- Analytics
Scale

- Capacity
- Number of Volumes
Deployment Flexibility

OpenEBS Deployment Options for:
- Dedicated Storage (External)
- Hyper-converged
- Hybrid-Cloud (AWS)
OpenEBS Roadmap

- Soft launch / Basic version
  - Containerized controller
  - Longhorn integration basics

- 0.1 - Basic k8s integration
- 0.2 - Basic k8s integration
  - K8s Provisioning via EBS-like driver
  - S3 building blocks
  - Complete longhorn integration
  - First usable release for community consumption

- 0.3 - Full k8s integration (Hyper-Converged)
  - Tiering and QoS demonstrated
  - Building blocks of ML for storage analytics
  - DevOps operator use cases demonstrated

- 0.4 - Tiering and distributed storage
  - Complete K8s integration
  - Fully hyper-converged
  - Support for local backing store
  - Building blocks of QoS
  - Building blocks of Tiering

- 1.0 - First enterprise edition
  - Integration into Enterprise LDAP w/ role based access control
  - All features supportable at scale
  - DevOps operator use cases demonstrated
  - Integration into Enterprise LDAP w/ role based access control
  - First usable release for community consumption

Timeline:
- Jan, 2017
- Mar, 2017
- Jun, 2017
- Sep, 2017
- Nov, 2017
- Nov, 2017
Join the community
#slack
slack.openebs.io

@openebs

blog.openebs.io

https://github.com/openebs
Stateful containers?!

“For which workloads or application use cases have you used/do you anticipate to use containers?”

- Data Apps: 77%
- Cloud Apps: 71%
- Systems of Engagement: 62%
- Systems of Record: 62%
- Web and Commerce Software: 57%
- Mobile Apps: 52%
- Social Apps: 46%

http://www.slideshare.net/Red_Hat_Storage/red-hat-storage-day-boston-persistent-storage-for-containers
Stateful containers?!