Containerized Storage for Containers

September 2017
@OpenEBS
Evan Powell - @epowell101
Team of storage and DevOps engineers with a 5 year track record of delivering containerized storage (built on jails) to NTT, eSilicon, TCS and other service providers.

CloudByte – our SDS brand.
Who?

Team of storage and DevOps engineers with a 5 year track record of delivering containerized storage (built on jails) to NTT, eSilicon, TCS and other service providers.

**CloudByte** – our SDS brand.

---

**Evan Powell, CEO**

Director of Ops & BD who started founding companies back in 2000 and is still at it. Also an investor and advisor and friend to enterprise focused entrepreneurs.

**Started as CloudByte / Open EBS**

CEO 7/1/17
OpenEBS stars on GitHub (vanity)
What if storage for container native applications was itself container native?
Minions run on physical nodes
PODs group containers, share an IP address, and each include a Kubelet agent
K8S Master services include: etcd, APIs, the scheduler, the control manager & others
Architecture: Kubernetes + OpenEBS

Data Containers run in PODs on physical machines

Data Containers group physical disks local or remote, manage replicas, provide QoS control & more

OpenEBS brain runs on the Master; delivers services such as: APIs, a storage scheduler, analytics…
DAS increasingly preferred to NAS/SAN

DAS

Benefits:
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

“We have ~100k nodes of Cassandra alone, and use DAS because it is easier even if it burns energy and capEx.”

Other example DAS users moving to Kubernetes:

- Netflix
- Booking.com
- Tenable
- Walmart
- Reddit

2017 Storage Developer Conference. © CloudByte Inc. All Rights Reserved.
DAS increasingly preferred to NAS/SAN

**DAS**

**Benefits:**
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

**Concerns:**
- Under-utilized hardware
  - 10% or less utilization
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

“**We have ~100k nodes of Cassandra alone, and use DAS because it is easier even if it burns energy and capEx.**”

**Other example DAS users moving to Kubernetes:**

- Netflix
- Booking.com
- Tenable
- Walmart
- GitHub
DAS and Distributed

**DAS**

**Benefits:**
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

**Concerns:**
- Under-utilized hardware
  - 10% or less utilization
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

**Distributed**

**Benefits:**
- Centralized management
- Greater density and efficiency
- Storage features such as:
  - Data protection
  - Snapshots for versioning

**Concerns:**
- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible
DAS and Distributed

**DAS**

**Benefits:**
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

**Concerns:**
- Under-utilized hardware ○ 10% or less utilization
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

**Distributed**

**Benefits:**
- Centralized management
- Greater density and efficiency
- Storage features such as:
  - Data protection
  - Snapshots for versioning

**Concerns:**
- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible
DAS

Under-utilized hardware

Hard to manage

Lacks all storage features

Hard to support workload mobility
YASSS: yet another scale out storage system

Not tunable for containers

Lock-in & not x cloud

Latency & Big O

Monolithic mess
YASSS: yet another scale out storage system

- Not tunable for containers
- Latency & Big O
- Lock-in & not in cloud
- Monolithic mess

Huge blast radius
Yet Another Scale Out Storage System
YASSS

Yet Another Scale Out Storage System
Container Attached Storage = DAS++

OpenEBS = “CAS”
- Simple
- No new skills required
- Per microservice storage policy
- Data protection & snapshots
- Reduces cloud vendor lock-in
- Eliminates storage vendor lock-in
- Highest possible efficiency
- Large & growing OSS community
- Natively cross cloud

DAS

Benefits:
- Simple
- Ties application to storage
- Predictable for capacity planning
- App deals with resiliency
- Can be faster

Concerns:
- Under-utilized hardware
  - 10% or less utilization
- Wastes data center
- Difficult to manage
- Lacks storage features
- Cannot be repurposed - made for one workload
- Does not support mobility of workloads via containers
- Cross cloud impossible

Distributed

Benefits:
- Centralized management
- Greater density and efficiency
- Storage features such as:
  - Data protection
  - Snapshots for versioning

Concerns:
- Additional complexity
- Enormous blast radius
- Expensive
- Requires storage engineering
- Challenged by container dynamism
- No per microservice storage policy
- I/O blender impairs performance
- Locks customers into vendor
- Cross cloud impossible
Container Attached Storage

Tunable per container

Cross cloud portability

Cross SAN portability

No special skills needed!
Why per micro-service granularity important?

Today’s scale out storage systems are almost infinitely configurable. Matching their configurations to static workloads is difficult. In a world of dynamic, ever evolving workloads due to micro-services and multi-cloud deployments, it is not a tractable problem for humans.

Let’s say you want to benchmark for your workload:

<table>
<thead>
<tr>
<th>Number of performance-affecting configuration parameters (*)</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of values each parameter can have (**)</td>
<td>4</td>
</tr>
<tr>
<td>Number of workloads (***</td>
<td>32</td>
</tr>
<tr>
<td>Time to run each (workload, config) benchmark (****)</td>
<td>10min</td>
</tr>
<tr>
<td>Number of runs (******)</td>
<td>10</td>
</tr>
</tbody>
</table>
Why per micro-service granularity important?

Today’s scale out storage systems are almost infinitely configurable. Matching their configurations to static workloads is difficult. In a world of dynamic, ever evolving workloads due to micro services and multi-cloud deployments, it is not a tractable problem for humans.

Let’s say you want to **benchmark** for your workload:

<table>
<thead>
<tr>
<th>Duration of run</th>
<th>6,700 years</th>
</tr>
</thead>
</table>

| Number of performance-affecting configuration parameters (*) | 10 |
| Number of values each parameter can have (**) | 4 |
| Number of workloads (***) | 32 |
| Time to run each (workload, config) benchmark (****) | 10min |
| Number of runs (***** ) | 10 |

---

*Source: Alex Aizman’s blog: https://storagetarget.com/author/alexaizman/*
Normal behavior

1. CI/CD pipeline deploys:
   
   cassandra

2. OpenEBS reads intent
   FROM K8S For example:
   - 2 sync copies
   - 50 / 50 cloud
   - 2 Data centers
Normal behavior

1. CI/CD pipeline deploys:
   - **cassandra**

2. OpenEBS reads intent FROM K8S For example:
   - 2 sync copies
   - 50 / 50 cloud
   - 2 Data centers
Normal behavior

1. CI/CD pipeline deploys:
   - cassandra

2. OpenEBS reads intent FROM K8S For example:
   - 2 sync copies
   - 50 / 50 cloud
   - 2 Data centers

3. writes STORED locally:
   - working set in NVM
   - rebalancing in background
Normal behavior: differentiation

1. Transparency:
   - works across clouds, & CAS, seamlessly

2. FAST:
   - Much faster than DAS or network attached
   - Could be 100 or even 1000x faster
Normal behavior: differentiation

1. Transparency:
   - works across clouds, & CAS, seamlessly

2. FAST:
   - Much faster than DAS or network attached
   - Could be 100 or even 1000x faster

3. Happier Cassandra:
   - When Cassandra nodes fail, rebalancing, which degrades performance, typically NOT needed

4. Happier CFO:
   - Far greater utilization
   - No more cloud lock-in
## Storage Orchestration - K8s Vs OpenEBS

<table>
<thead>
<tr>
<th>Functionality</th>
<th>K8s</th>
<th>OpenEBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Life-cycle - provisioning, attaching/detach, de-provisioning</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Monitoring the volume data</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Scheduling the volume provisioning intelligently</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Smart space management</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Availability of volume</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Application aware backup/snapshots</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>QoS guarantee (latency guarantee)</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Smart movement of the volumes across racks and regions</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Cataloging of storage requirements for different applications</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Storage just fades away as a concern
Benefits of Cloud Native approach

OpenEBS always available multi-cloud block services are delivered via containerized microservices

- **Granularity**: Each pod has its own block protocol stack, tiering engine, QoS engine and more. SLAs per POD.

- **Performance & efficiency**: Access patterns maintained per-volume. Hot-data on NVMe-Flash or on 3DX-Memory. Cold-data on slower disks / SAN/ Cloud-Storage/S3.

- **Scalability**: Metadata based on “size of the volume” & not on “number of volumes”. Can reduce metadata traversal 100x.

- **Blast radius**: Eliminates risk of scale out storage system corruption.

And much, much more such as cost savings due to resource efficiency, innate multitenancy, and cross cloud portability.

The **Conway’s Law** benefits might be the most important - *innovation unlocked*
Parting thoughts

Cloud Native
Parting shots

Cloud Native

Cloud Washing
Get started with OpenEBS

```
kubectl apply -f openebs-operator.yaml

kubectl apply -f openebs-config.yaml
```
Q&A

Please tell us we are crazy.
Find out HOW it works!
Meet some folks who want storage to be
fixed once and for all.

Slack.openebs.io
blog.openebs.io
Github.com/openEBS
@openebs