Container Storage Best Practices in 2017

Myth-busting and taking state of the present

Keith Resar
Red Hat Solution Architect
January 24th, 2017
Agenda

- Container Storage Myths
- Container Storage Primer
- Review 6 Storage Drivers
- Choose a Storage Driver
Container Storage History / Myths

http://en.wikifur.com/wiki/Sloth_(species)
Container Storage Level Set
Image : Container :: Class : Object

For humans, read this to say:
An image is to a container, as a class is to an object.

```bash
> ls -l /image; echo $?
0

> pgrep image; echo $?
1
```
Data Volume Storage

Can be extended to support many endpoints and protocols using installable docker plugins.

- Local
- LVM
- GlusterFS
- Ceph
- NFS
- iSCSI
Container Storage

Data Volume Storage

CONTAINER

READ-WRITE LAYER

IMAGE

/data01
IMAGE

E8E3A9F82AF5  55 MB

AAD5D4C7BBA9  192 MB

httpd
rhel7:latest
CONTAINER (INSTANTIATION)

READ/WRITE LAYER

IMAGE

E8E3AAF82AF5  55 MB
AAD5D4C7BBA9  192 MB

container layer

httpd
rhel7:latest
IMAGE

E8E3AAF82AF5  55 MB
AAD5D4C7BBA9  192 MB

httpd
rhel7:latest
Copy-on-write Strategy

container layer

httpd

rhel7:latest
Container Storage Drivers
## Available Storage Drivers

<table>
<thead>
<tr>
<th>Technology</th>
<th>Driver</th>
<th>Introduction</th>
<th>File vs. Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFS</td>
<td>vfs</td>
<td>origin</td>
<td>* File</td>
</tr>
<tr>
<td>AUFS</td>
<td>aufs</td>
<td>origin</td>
<td>File</td>
</tr>
<tr>
<td>OverlayFS</td>
<td>overlay/overlay2</td>
<td>Aug 2014 (1.11)</td>
<td>File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>June 2016 (1.12)</td>
<td></td>
</tr>
<tr>
<td>Device Mapper</td>
<td>devicemapper</td>
<td>Sept 2013 (0.7)</td>
<td>Block</td>
</tr>
<tr>
<td>Btrfs</td>
<td>btrfs</td>
<td>Nov 2013</td>
<td>File</td>
</tr>
</tbody>
</table>
vfs Driver  (1 of 6)

Naive implementation lacking union filesystem and copy-on-write
### VFS Driver (1 of 6)

Naive implementation lacking union filesystem and copy-on-write

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference compatibility model</td>
<td>No shared memory, union filesystem, or copy-on-write</td>
<td>Not for production use</td>
</tr>
<tr>
<td>Useful for docker-in-docker scenarios to avoid nesting storage drivers</td>
<td></td>
<td>Important support role for storage driver development</td>
</tr>
</tbody>
</table>
AUFS Driver  (2 of 6)
The original docker storage driver

@KeithResar

https://docs.docker.com/engine/userguide/storagedriver/ufs-driver/
# AUFS Driver  (2 of 6)

The original docker storage driver

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battle hardened driver</td>
<td>Carried patch to mainline Linux kernel limits distro support</td>
<td>Default for non-RH, will meet majority of needs</td>
</tr>
<tr>
<td>Performant and stable for wide range of use cases</td>
<td>File level implementation impacts copy-on-write</td>
<td>Expectation that it will be supplanted by an Overlay implementation</td>
</tr>
<tr>
<td>Supports shared memory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

@KeithResar
Overlay Driver  (3 of 6)

Legacy union filesystem driver, superseded by overlay2

https://docs.docker.com/engine/userguide/storagedriver/overlayfs-driver/
Overlay Driver  (3 of 6)
Legacy union filesystem driver, superseded by overlay2

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete union filesystem merged into the mainline kernel</td>
<td>Architecture drove explosive inode usage, often to the point of exhaustion</td>
<td>Used for backward compatibility in pre-4.0 kernels</td>
</tr>
<tr>
<td>Shared memory</td>
<td>Slow commit performance</td>
<td>Broad distro support beyond aufs</td>
</tr>
</tbody>
</table>
Overlay2 Driver  (4 of 6)

Lessons learned from original overlay, and looking forward to continued maturity
## Overlay2 Driver  (4 of 6)

Lessons learned from original overlay, and looking forward to continued maturity

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retains all benefits of overlay (shared memory, broad distro support)</td>
<td>Relatively young codebase (initial release with Docker 1.12 in June 2016)</td>
<td>With maturity may be the best route forward for consistent defaults across many Linux distributions</td>
</tr>
<tr>
<td>Resolves inode exhaustion problems</td>
<td>File-based so copy-on-write operations may be expensive</td>
<td></td>
</tr>
</tbody>
</table>
Devicemapper Driver   (5 of 6)
Lvm integrated block-based storage driver
Devicemapper Driver  (5 of 6)
Lvm integrated block-based storage driver, default on RHEL

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block-based solution offers efficient copy-on-write</td>
<td>Manual setup is intimidating</td>
<td>Red Hat go-to graphdriver with mature codebase</td>
</tr>
<tr>
<td>Quota support</td>
<td>No shared memory support</td>
<td></td>
</tr>
<tr>
<td>Available direct and loop modes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Btrfs Driver   (6 of 6)

Another next generation filesystem, with a continued heavy development requirement
Btrfs Driver  (6 of 6)

Another next generation filesystem, with a continued heavy development requirement

<table>
<thead>
<tr>
<th>The Good</th>
<th>The Bad</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Now offers SELinux support and quota</td>
<td>No page-cache sharing between containers</td>
<td>Btrfs hasn’t been a mainstream choice for Linux distros, driving less attention and less testing</td>
</tr>
<tr>
<td></td>
<td>Small writes can lead to out-of-space conditions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requires btrfs specific tools rather than Linux native</td>
<td></td>
</tr>
</tbody>
</table>
Choosing a Storage Driver
Benchmark Approach

Benchmarking is treacherous and confusing, and often done poorly - which means that you need to take any benchmark results with a large grain of salt.

If you've spent less than a week studying a benchmark result, it's probably wrong.

(Running a benchmark is the easy part. Understanding a benchmark can take much longer.)
Benchmark 1: Reading Files

- Reading Small Files
- Reading Large Files
- Reading File Tree
Naive benchmarking, for discussion purposes only. Don’t trust this!
Naive benchmarking, for discussion purposes only. Don’t trust this!

@KeithResar
Naive benchmarking, for discussion purposes only. Don’t trust this!
Naive benchmarking, for discussion purposes only. Don’t trust this!
Benchmark 2: Appending to Files

- Appending to Small Files
- Appending to Large Files
- Appending to File Tree
Naive benchmarking, for discussion purposes only. Don’t trust this!
Naive benchmarking, for discussion purposes only. Don’t trust this!
Naive benchmarking, for discussion purposes only. Don’t trust this!
## Storage use cases

<table>
<thead>
<tr>
<th>Technology</th>
<th>Attributes</th>
<th>Good Use Case</th>
<th>Bad Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUFS</td>
<td>Stable, Production Ready, Good Memory Use</td>
<td></td>
<td>High Write Activity</td>
</tr>
<tr>
<td>Btrfs</td>
<td>Mainline Kernel</td>
<td></td>
<td>High Write Activity</td>
</tr>
<tr>
<td>Overlay</td>
<td>Stable, Good Memory Use, Mainline Kernel</td>
<td></td>
<td>Container Churn</td>
</tr>
<tr>
<td>Devicemapper (loop)</td>
<td>Stable, Mainline Kernel</td>
<td></td>
<td>Production, Performance</td>
</tr>
<tr>
<td>Devicemapper (direct-lvm)</td>
<td>Stable, Production Ready, Mainline Kernel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Resources

Storage Drivers in Docker: A Deep Dive
https://integratedcode.us/2016/08/30/storage-drivers-in-docker-a-deep-dive/

The Docker community has documented a good bit of this detail in the official storage driver documentation
https://docs.docker.com/engine/userguide/storagedriver/selectadriver/

Docker Issues and Tips (aufs/overlay/btrfs..)
https://github.com/AkihiroSuda/issues-docker#docker-issues-and-tips-aufsoverlaybtrfs

Comprehensive Overview of Storage Scalability in Docker (2014)
https://developers.redhat.com/blog/2014/09/30/overview-storage-scalability-docker/
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews