
CERN

Dynamic Storage Provisioning of Manila/CephFS Shares on Kubernetes

Róbert Vašek <robert.vasek@codefreax.org>
Ricardo Rocha <ricardo.rocha@cern.ch> @ahcorporto

home.cern

Table of Contents

We are here!

Introduction

Container Storage Interface

CSI CephFS

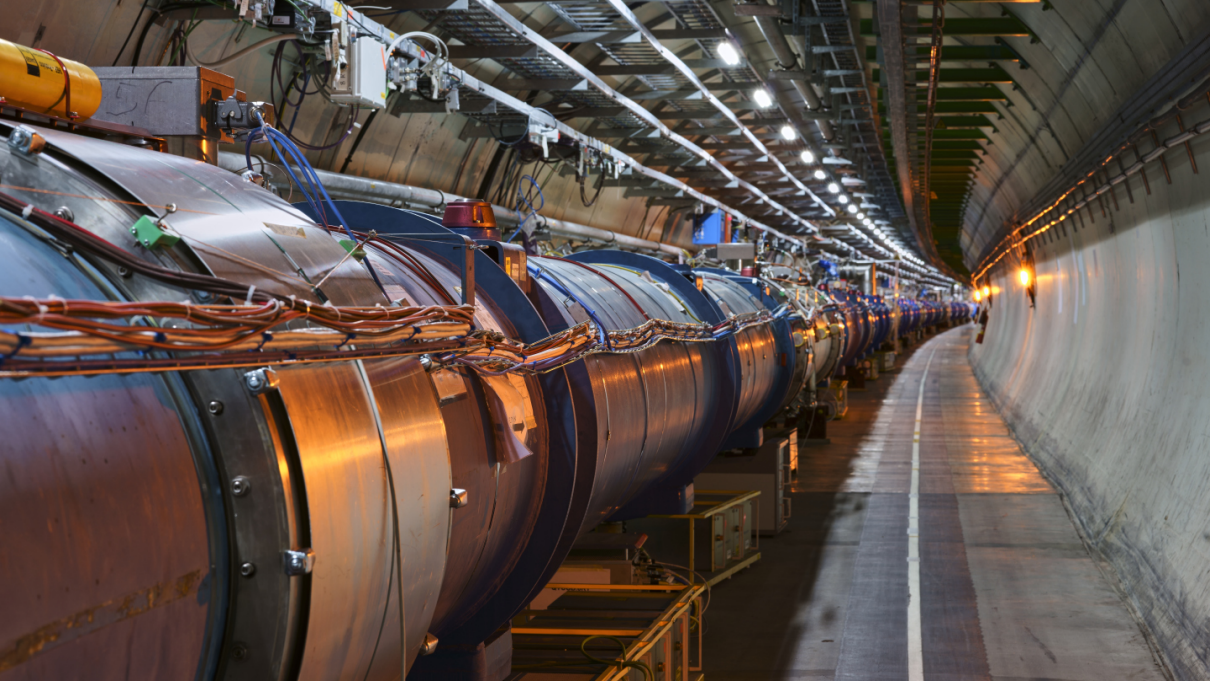
Manila shares with Kubernetes

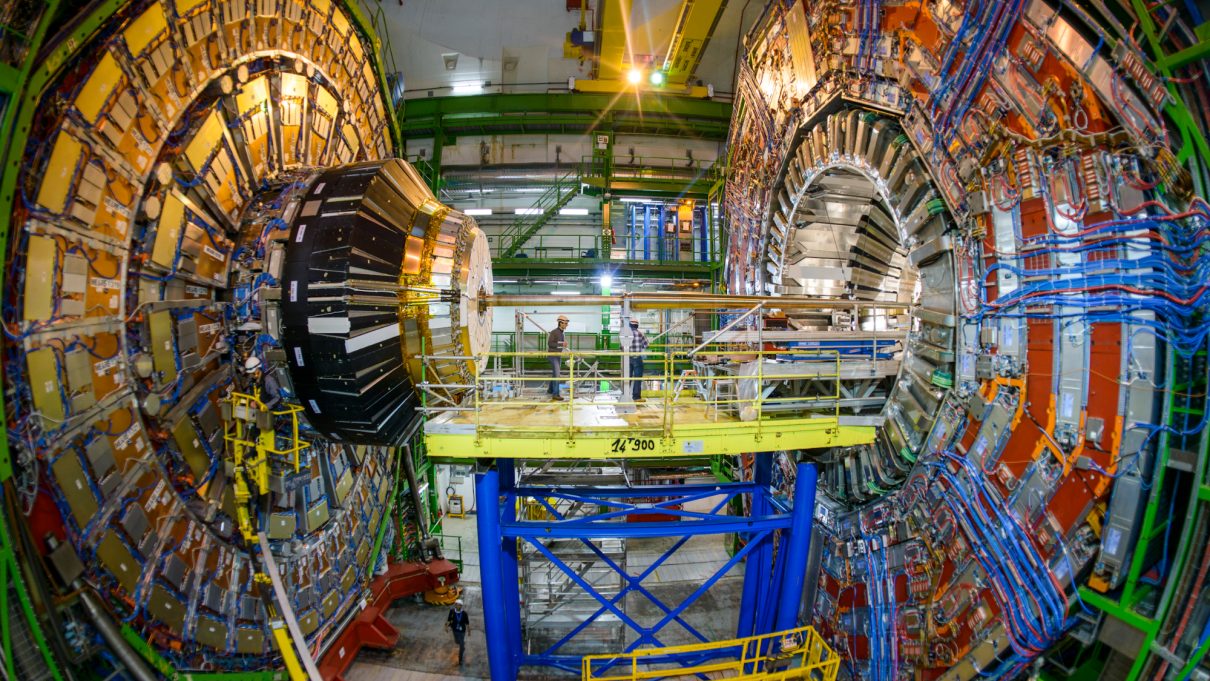
Results, numbers, plots...

What is 96% of the universe made of?

What was the state of matter just after the Big Bang?

Why isn’t there anti-matter in the universe?




14'900

Cloud resources

Used	Available	Used	Available	Used	Available
307.2 K cores	300.1 K cores	858.9 TiB RAM	937.6 TiB RAM	10.3 PiB disk	15.6 PiB disk

Openstack services stats

Users	Projects	VMs	Magnum clusters	Hypervisors	Images	Baremetal nodes
3491	4512	36001	412	9214	3178	1295

Volumes	Volume size	Fileshares	Fileshares size
6155	1.91 PiB	366	215 TiB

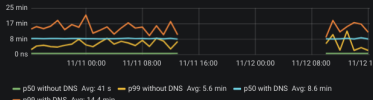
Resource overview by time

VMs created/deleted

VM changes

Total VMs

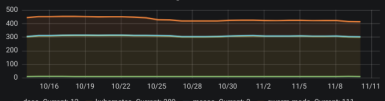
Average VM boot time



Projects and users

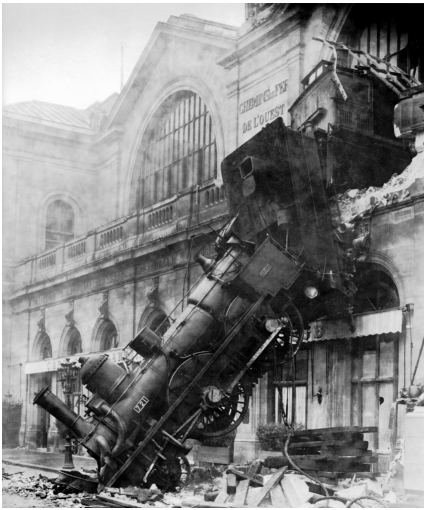
Hypervisors

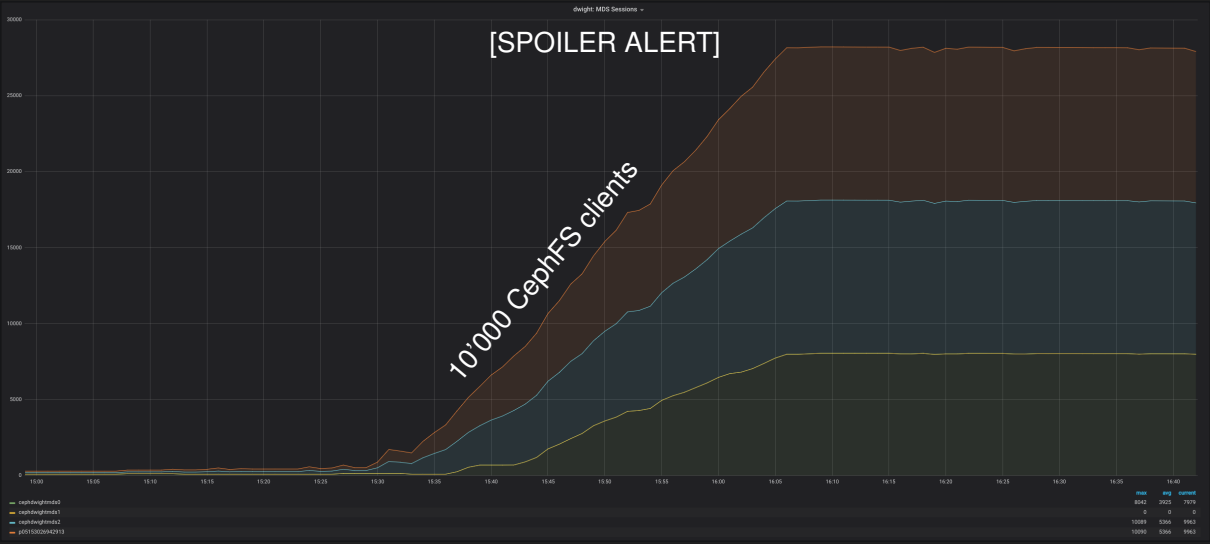
Magnum clusters



Dynamic Storage Provisioning of Manila/CephFS Shares on Kubernetes

...working title *"From a train wreck to a train ride"*




dwight: MDS Sessions
[SPOILER ALERT]
10’000 CephFS clients
30000
25000
20000
15000
10000
5000
0
15:00
15:05
15:10
15:15
15:20
15:25
15:30
15:35
15:40
15:45
15:50
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
max
avg
current
cephdwightmds0
8042
3925
7979
cephdwightmds1
0
0
0
cephdwightmds2
10089
5366
9963
p0515302604213
10090
5366
9963

We are here!

Introduction

Container Storage Interface

CSI CephFS

Manila shares with Kubernetes

Results, numbers, plots...

Container Storage Interface - motivation

Storage

Container Storage Interface - motivation

CO 1

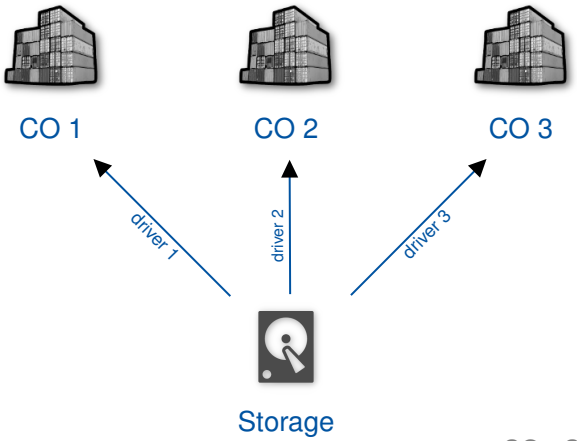
CO 2

CO 3

Storage

CO - Container Orchestrator

Container Storage Interface - motivation



CO - Container Orchestrator

Container Storage Interface - motivation

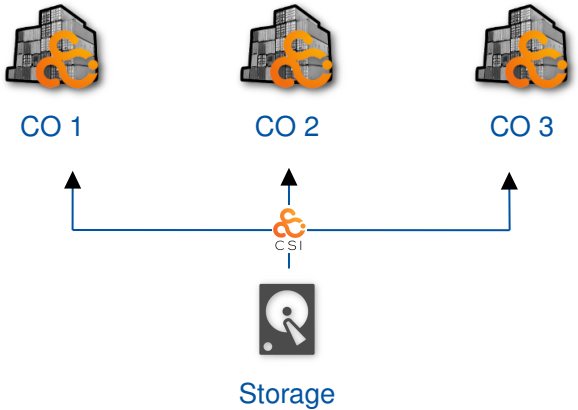
From driver's POV:

- Lack of standardization
- Higher development and maintenance costs

From CO's POV:

- Volume plugin development is tightly coupled with release cycles of the CO
- Bugs in volume plugins can crash critical components
- Volume plugins get full privileges
- Difficult dependency management

Container Storage Interface - motivation



CO - Container Orchestrator

Container Storage Interface

Overview

- Industry standard for cluster-wide storage plugins
- Collaboration of communities incl. Kubernetes, Mesos, Docker and Cloud Foundry
- Defines the protocol between a CO and a plugin
- Plugins are CO-agnostic
- Write once – use everywhere, *just works™*

CONTAINER
STORAGE
INTERFACE

Container Storage Interface

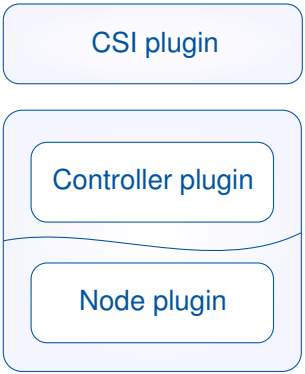
- First alpha released in Dec 2017
- Working implementation in Kubernetes 1.9 already, a lot of changes since then, some of those were breaking
- Other COs soon to follow

December 2017	v0.1.0
March 2018	v0.2.0
June 2018	v0.3.0
⋮	
just today	v1.0.0-rc2
end of Nov 2018	v1.0.0

CSI Services

CSI RPC services (endpoints):

- *Identity service:* allows a CO to query for plugin's capabilities, health probes and other metadata. Must be implemented by both controller and node plugins, you'll see why in a bit.
- *Controller service:* creates, deletes, lists volumes and their snapshots.
- *Node service:* (un)stages, (un)publishes volumes on a node.



CSI Architecture

CSI RPCs quick overview

*Controller Service**

- CreateVolume
- DeleteVolume
- ControllerPublishVol*
- ControllerGetCaps
- ...

Node Service

- NodeStageVolume*
- NodePublishVolume
- NodeGetCapabilities
- ...

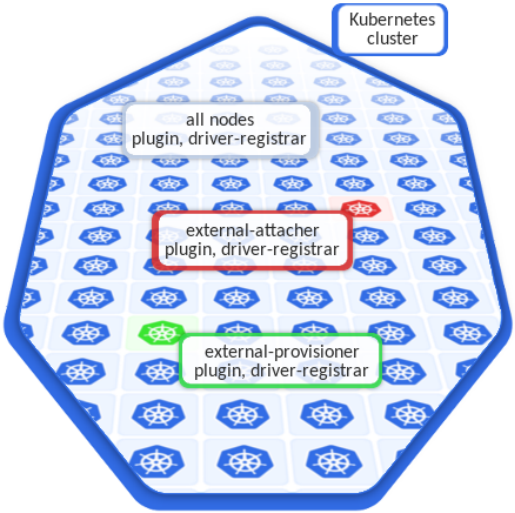
Identity Service

- GetPluginInfo
- GetPluginCapabilities
- ...

* optional

CSI in Kubernetes

- In-tree CSI volume plugin in kubelet
  `Node(Un)StageVolume`
  `Node(Un)PublishVolume`
- Side-car containers
  - driver-registrar
    plugin discovery,
    registers the driver with kubelet
  - external-provisioner
    `CreateVolume`
    `DeleteVolume`
  - external-attacher
    `ControllerPublishVolume`
    `ControllerUnpublishVolume`



We are here!

Introduction

Container Storage Interface

CSI CephFS

Manila shares with Kubernetes

Results, numbers, plots...

CSI CephFS overview

`github.com/ceph/ceph-csi`

- Provides an interface between a CSI-enabled Container Orchestrator and the Ceph cluster
- Provisions and mounts CephFS volumes
- Supports both the kernel CephFS client and the CephFS FUSE driver

+

CSI CephFS overview

Compared to Kubernetes in-tree CephFS volume plugin

- In-tree volume plugins to be eventually migrated to CSI
- Decoupled from Kubernetes
- Ability to choose between mounting tools
- Planned support for volume expansion, snapshots

We are here!

Introduction

Container Storage Interface

CSI CephFS

Manila shares with Kubernetes

Results, numbers, plots...

Manila external provisioner for Kubernetes overview

`github.com/kubernetes/cloud-provider-openstack`

- Provisions new Manila shares, fetches existing ones
- Maps them to Kubernetes *PersistentVolume* objects
- Currently supports CephFS shares only
  (both in-tree CephFS plugin and csi-cephfs)
- Supports authentication using both user credentials as well as trustees
  - Magnum → Kubernetes + manila-provisioner StorageClass + trustee secrets = Manila support out-of-the-box
- The future is in CSI

We are here!

Introduction

Container Storage Interface

CSI CephFS

Manila shares with Kubernetes

Results, numbers, plots...

Benchmarks

Goals

1. Verify the CSI CephFS implementation for common use cases
2. Verify the Manila Provisioner implementation
3. Test CSI CephFS driver behavior on a heavy loaded cluster

Client

1. Kubernetes v1.12.1, csi-cephfs 0.3.1

Clusters

1. *Dwight:* 3x24 HDD OSDs, 3 MDS, Ceph Luminous Bluestore
2. *Jim:* 300 SSD OSD, 2 MDS, Ceph Luminous Bluestore, hyper-converged

Benchmarks

Methodology

1. Provision *s* CephFS shares using *manila-provisioner*
2. Create a Deployment with *r* replicas, sized so we get one pod per node
3. Mount *s* provisioned shares into each pod using csi-cephfs (fuse)
4. Measure time taken for all pods to become *Running*, MDS sessions, hcr/s

Tests

1. *idle:* do nothing
2. *busy:* unpack a large archive (linux kernel)

Parameters

1. $s = 100, r = 100$; 10'000 idle clients
2. $s = 10, r = 100$; 1'000 busy clients

Idle benchmark - attempt #1

Our very first test of csi-cephfs with concurrent workloads

Preparation

- 10 CephFS shares
- 100 replicas
- The goal is to have 1’000 idle clients running

Idle benchmark - attempt #1

Our very first test of csi-cephfs with concurrent workloads

Preparation

- 10 CephFS shares
- 100 replicas
- The goal is to have 1’000 idle clients running

Outcome

►:(

Idle benchmark - attempt #1

```
Aug 30 09:51:45 cci-cephfs-scale-003-n3tf4nqlzisk-minion-56.cern.ch runc[3255]:
↪ E0830 09:51:45.410380    3270 csi_attacher.go:137] kubernetes.io/csi:
↪ attacher.WaitForAttach failed for volume
↪ [pvc-c5848f32-ac39-11e8-bbfb-02163e01b7c5] (will continue to try):
↪ volumeattachments.storage.k8s.io
↪ "csi-4f2dbe5cb257e7d7b172c4a1e6a1d26bfff82dabeb91e441c527d46f368f1615" is
↪ forbidden: User "system:node:cci-cephfs-scale-003-n3tf4nqlzisk-minion-56" cannot
↪ get volumeattachments.storage.k8s.io at the cluster scope: no path found to
↪ object
```

Idle benchmark - attempt #1

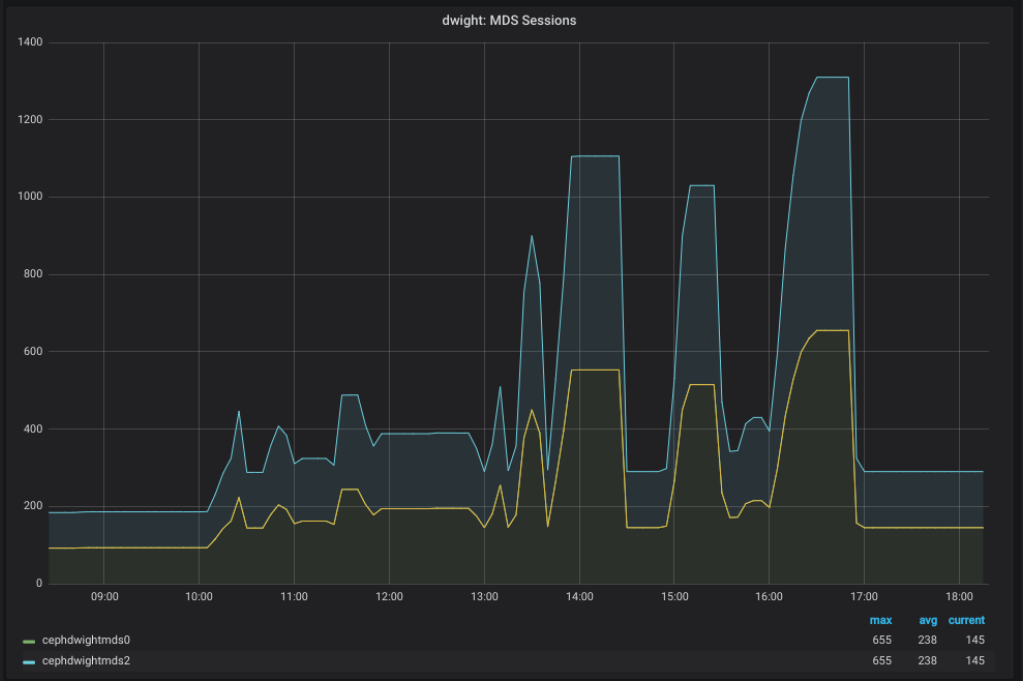
```
User "system:node:NODE_NAME" cannot get volumeattachments at the
cluster scope:  no path found to object
```

- Provisioning of shares worked just fine
- Some pods survived
- Others that reported this error would never recover

Idle benchmark - attempt #2

A script that:

- Scales the deployment in small increments
- Kills pods that take too long to create (got stuck in the *VolumeAttachment* error)


dwight: MDS Sessions
1400
1200
1000
800
600
400
200
0
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
max
avg
current
cephdwightmds0
655
238
145
cephdwightmds2
655
238
145

Idle benchmark - attempt #2

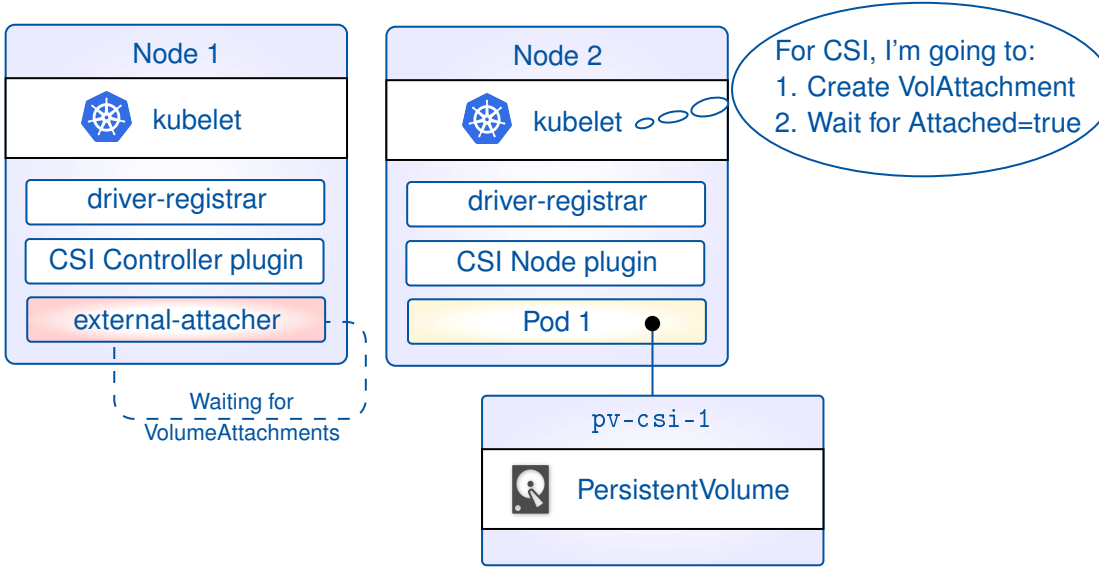
Outcome

- We've managed to get up to 655 concurrent clients (could be even more)
- Slow and ugly but somehow working

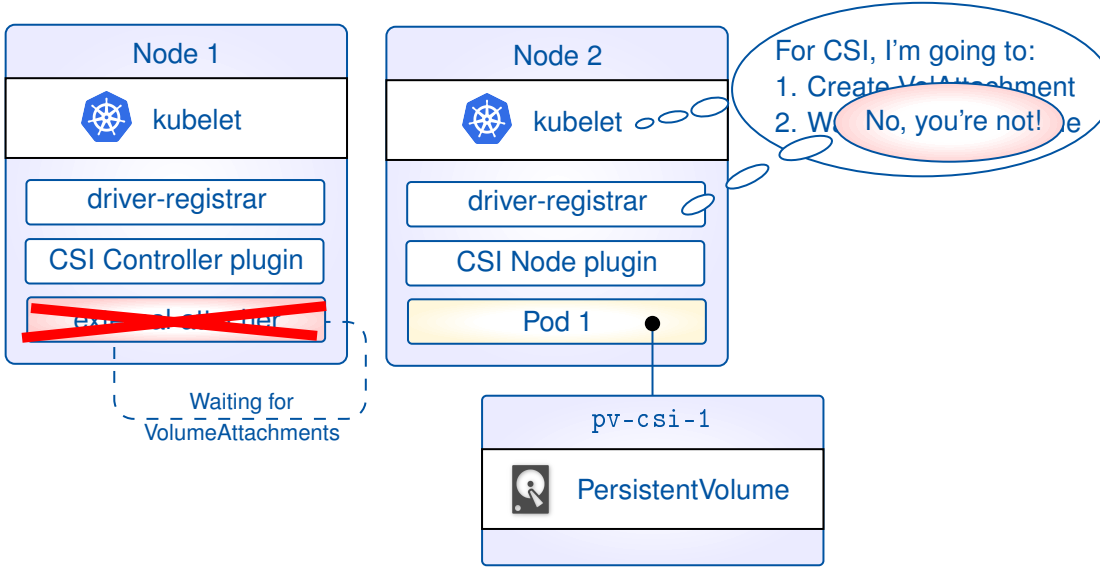
Idle benchmark - attempt #3

"Third time's the charm"?

- Kubernetes 1.12, driver-registrar 0.4 released
- Kubelet plugin registration of CSI drivers
- *CSISkipAttach*
  - Skips the creation of *VolumeAttachment* objects
  - Volumes are marked as attached immediately



* external-provisioner omitted from image



* external-provisioner omitted from image

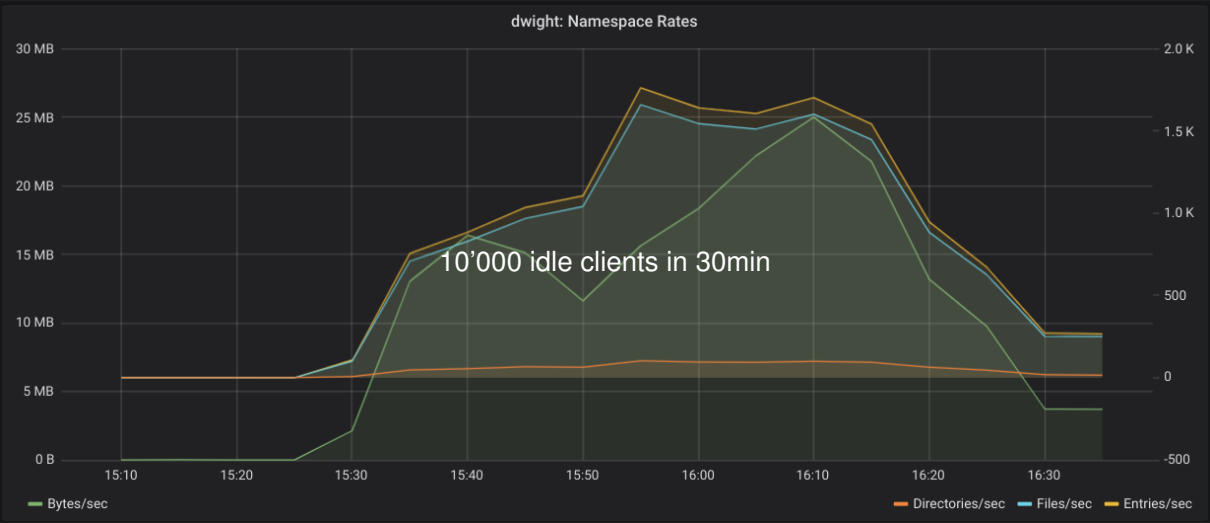
Idle benchmark - attempt #3

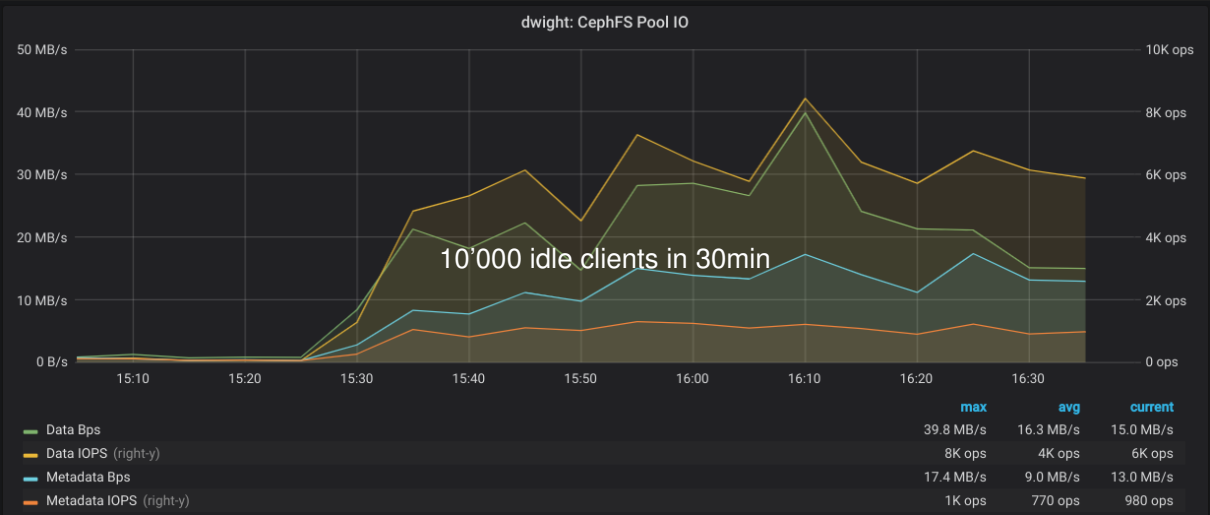
- We resumed our tests using the new versions of Kubernetes and driver-registrar
- Parameters: 100 CephFS Shares * 100 replicas = 10’000 idle clients
- Gradual, gentle scale up

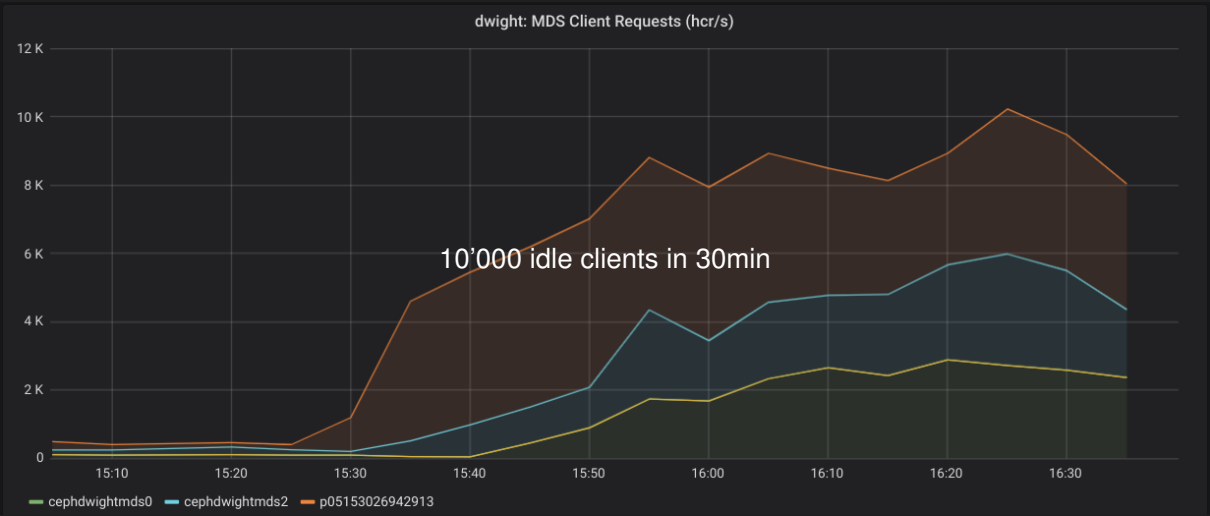
Idle benchmark - attempt #3

```
15:29:28 : scaling to 5 replicas
15:32:33 : scaling to 10 replicas
...
15:57:18 : scaling to 80 replicas
15:58:51 : scaling to 85 replicas
16:00:29 : scaling to 90 replicas
16:02:26 : scaling to 95 replicas
16:04:16 : scaling to 100 replicas
```


dwight: MDS Sessions
30000
25000
20000
15000
10000
5000
0
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
16:30
10’000 idle clients in 30min
max
avg
current
cephdwightmds0
8042
4039
8011
cephdwightmds2
10082
5551
10056
p05153026942913
10082
5552
10056


dwight: Namespace Rates
30 MB
25 MB
20 MB
15 MB
10 MB
5 MB
0 B
2.0 K
1.5 K
1.0 K
500
0
-500
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
16:30
10’000 idle clients in 30min
Bytes/sec
Directories/sec
Files/sec
Entries/sec


dwight: CephFS Pool IO
50 MB/s
40 MB/s
30 MB/s
20 MB/s
10 MB/s
0 B/s
10K ops
8K ops
6K ops
4K ops
2K ops
0 ops
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
16:30
10’000 idle clients in 30min
max
avg
current
Data Bps 39.8 MB/s 16.3 MB/s 15.0 MB/s
Data IOPS (right-y) 8K ops 4K ops 6K ops
Metadata Bps 17.4 MB/s 9.0 MB/s 13.0 MB/s
Metadata IOPS (right-y) 1K ops 770 ops 980 ops


dwight: MDS Client Requests (hcr/s)
12 K
10 K
8 K
6 K
4 K
2 K
0
10’000 idle clients in 30min
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
16:30
cephdwightmds0
cephdwightmds2
p05153026942913

Idle benchmark - attempt #3

Some bits still to be understood

```
mds.cephdwightmds2 mds.1 redacted:6800/2246956767 2238 : cluster [WRN] evicting
↪  unresponsive client cci-cephfs-scale-001-3vimalfm74dd-minion-76.cern.ch
↪  (674953366), after 304.623890 seconds
mds.cephdwightmds0 mds.2 redacted:6800/2942371108 2167 : cluster [WRN] evicting
↪  unresponsive client cci-cephfs-scale-001-3vimalfm74dd-minion-76.cern.ch:
↪  (674953366), after 304.277996 seconds
```

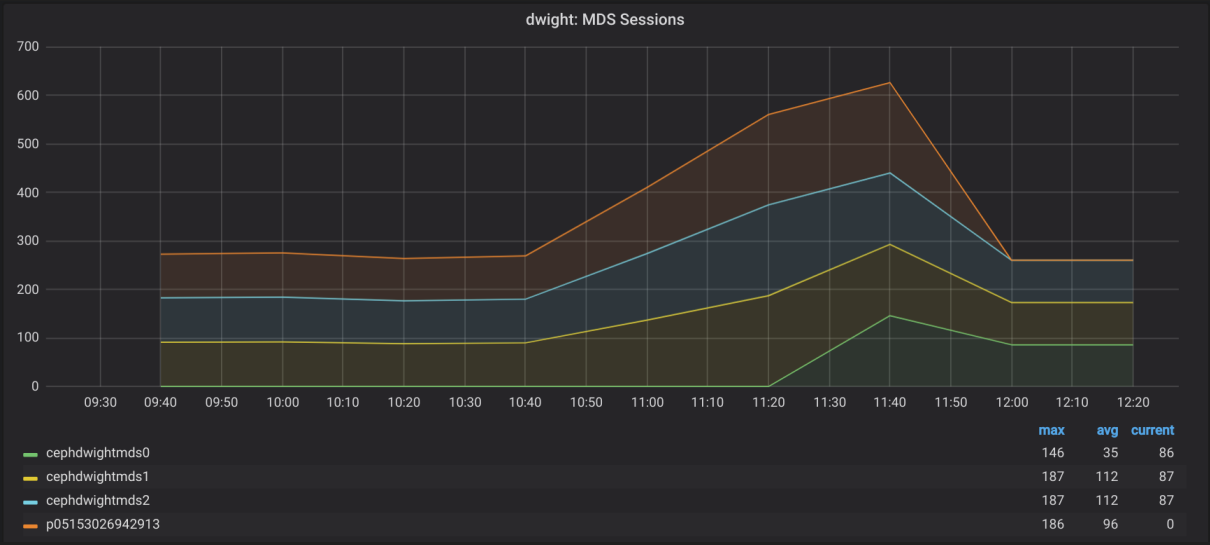
Idle benchmark - attempt #3

Some bits still to be understood

cci-cephfs-scale-001-3vimalfm74dd-minion-13	Ready	<none>	8d	v1.12.1
cci-cephfs-scale-001-3vimalfm74dd-minion-14	NotReady	<none>	8d	v1.12.1

Busy benchmark - attempt #1

- Each client extracting the linux kernel
- Slowly ramp up until it breaks


dwight: MDS Sessions
700
600
500
400
300
200
100
0
09:30
09:40
09:50
10:00
10:10
10:20
10:30
10:40
10:50
11:00
11:10
11:20
11:30
11:40
11:50
12:00
12:10
12:20
max avg current
cephdwightmds0 146 35 86
cephdwightmds1 187 112 87
cephdwightmds2 187 112 87
p05153026942913 186 96 0

Dan Van Der Ster 11:44 AM

i think you crashed something

mds's are flapping for some reason

HEALTH_OK now....

Ricardo Brito Da Rocha 11:47 AM

nice!

my day is done then 🙂

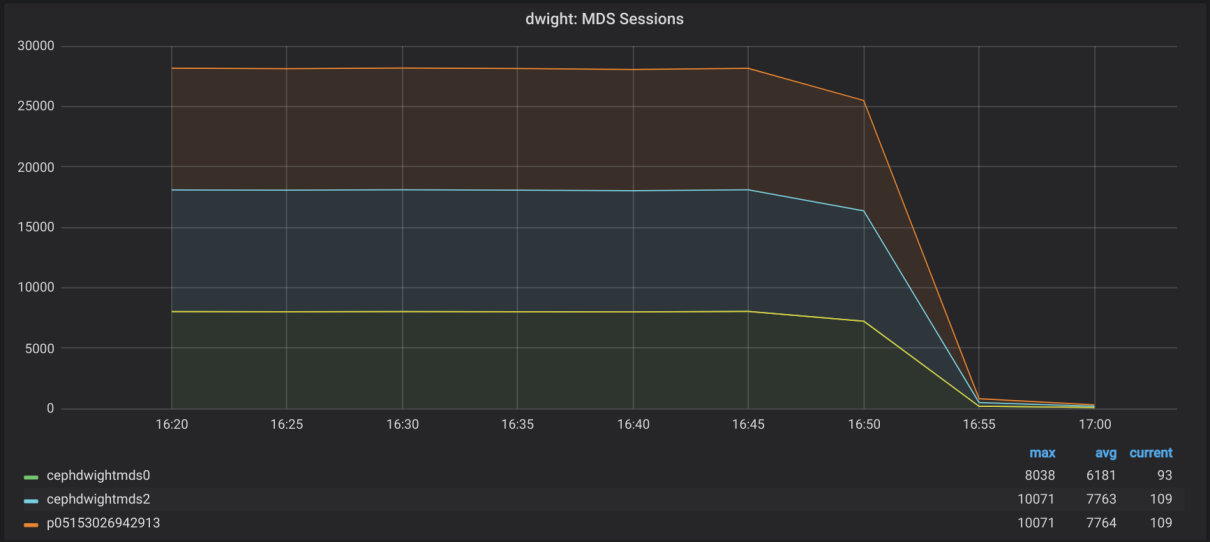
Robert Vasek 1:47 PM

very nice!

Bonus benchmark - client deletion

- Stop 10’000 clients and delete their shares, simultaneously
- Kubernetes and driver OK, CephFS OK, Manila share daemon needed a kick

```
| ... | pvc-08d... | 1 | CEPHFS | deleting  | False | Geneva CephFS Testing | nova |
| ... | pvc-7a9... | 1 | CEPHFS | deleting  | False | Geneva CephFS Testing | nova |
```


dwight: MDS Sessions
30000
25000
20000
15000
10000
5000
0
16:20
16:25
16:30
16:35
16:40
16:45
16:50
16:55
17:00
max
avg
current
cephdwightmds0
8038
6181
93
cephdwightmds2
10071
7763
109
p05153026942913
10071
7764
109

Conclusion & Next Steps

To recap:

- Standardized storage interface for Container Orchestrators with CSI
- Works nicely in Kubernetes, others soon to follow
- manila-provisioner + csi-cephfs handle large concurrency and scaling well
- Already in production at CERN

Next Steps:

- Add support for volume expansion and snapshots
- Make the Manila Provisioner a CSI plugin

Questions?

- Robert Vasek <robert.vasek@codefreax.org>
- Ricardo Rocha <ricardo.rocha@cern.ch> @ahcorporto
- CSI CephFS: https://github.com/ceph/ceph-csi
- Manila Provisioner: https://github.com/kubernetes/cloud-provider-openstack