Maximizing Network Throughput for Container Based Storage

David Borman
Quantum
Agenda

- Assumptions
- Background Information
- Methods for External Access
  - Descriptions, Pros and Cons
- Summary
Assumptions

- Services only run one instance
  - No load balancing to multiple back-ends
- External access to the service
  - Clients are not running on cluster nodes
- Throughput is the primary goal.
Some of the Technology

- Docker
- Kubernetes (k8s)
- Ceph (and Rook)
- iSCSI
Some of the Pieces

- Docker containers
- Kubernetes pods
- Kubernetes deployments
Docker

“Docker is a software technology providing containers … [using] resource isolation features of the Linux kernel such as cgroups and kernel namespaces … [providing] resource limiting, including the CPU, memory, block I/O, and network.” - https://en.wikipedia.org/wiki/Docker_(software)
“Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.” - https://kubernetes.io
Ceph

- “Ceph is a unified, distributed storage system designed for excellent performance, reliability and scalability.” - [http://ceph.com](http://ceph.com)

“Rook orchestrates battle-tested open-source storage technologies including Ceph … Rook is designed to run as a native Kubernetes service”
- https://rook.io
iSCSI Target implementations

- User Space
  - iscsi_tgt from the Intel SPDK
    - Using librbd and librados
- Kernel
  - Linux LIO
    - With kernel RBD module
“A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them ... For non-native applications, [it] offers a virtual-IP-based bridge to Services which redirects to the backend Pods”

- https://kubernetes.io/docs/concepts/services-networking/service/
K8S Service: Proxy-mode: iptables

- https://kubernetes.io/docs/concepts/services-networking/service/
K8S Service: Proxy mode: User Space

- https://kubernetes.io/docs/concepts/services-networking/service/
iSCSI

Kubernetes pods with IP addresses

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
<th>IP</th>
<th>NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>rook-operator-4134348477-hdjxh</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>36m</td>
<td>10.2.79.2</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>kube-system</td>
<td>calico-node-5z9bm</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>172.17.4.201</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>calico-node-gts4h</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>39m</td>
<td>172.17.4.203</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>kube-system</td>
<td>calico-node-thwmmw</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>172.17.4.101</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>kube-system</td>
<td>calico-node-v6wwv</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>172.17.4.202</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>kube-system</td>
<td>calico-policy-controller-7v74k</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>172.17.4.201</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>heapster-v1.2.0-3863399399-32f04</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>38m</td>
<td>10.2.69.4</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-apiserver-172.17.4.101</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>172.17.4.101</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-controller-manager-172.17.4.101</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>40m</td>
<td>172.17.4.101</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-dns-1358247298-w0mt</td>
<td>4/4</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>10.2.69.2</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-dns-autoscaler-2586315044-c8c48</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>10.2.69.3</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-proxy-172.17.4.101</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>40m</td>
<td>172.17.4.101</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-proxy-172.17.4.201</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>40m</td>
<td>172.17.4.201</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-proxy-172.17.4.202</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>40m</td>
<td>172.17.4.202</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-proxy-172.17.4.203</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>39m</td>
<td>172.17.4.203</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-scheduler-172.17.4.101</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>40m</td>
<td>172.17.4.101</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>kube-system</td>
<td>kubernetes-dashboard-3619675109-q1kz0</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>41m</td>
<td>10.2.79.5</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>rook</td>
<td>mon0</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>36m</td>
<td>10.2.87.2</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>mon1</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>36m</td>
<td>10.2.87.3</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>mon2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>36m</td>
<td>10.2.79.3</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>rook</td>
<td>osd-hwdr2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
<td>10.2.69.5</td>
<td>172.17.4.201</td>
</tr>
<tr>
<td>rook</td>
<td>osd-j6gj4</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
<td>10.2.87.4</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>osd-m71z8</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
<td>10.2.15.2</td>
<td>172.17.4.101</td>
</tr>
<tr>
<td>rook</td>
<td>osd-td2lc</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
<td>10.2.79.4</td>
<td>172.17.4.202</td>
</tr>
<tr>
<td>rook</td>
<td>rook-api-3722659863-53d33</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
<td>10.2.87.5</td>
<td>172.17.4.203</td>
</tr>
</tbody>
</table>
## Kubernetes pods with IP addresses

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>IP</th>
<th>NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>kube-system</td>
<td>calico-node-gts4h</td>
<td>172.17.4.203</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>kube-system</td>
<td>kube-proxy-172.17.4.203</td>
<td>172.17.4.203</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>mon0</td>
<td>10.2.87.2</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>mon1</td>
<td>10.2.87.3</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>osd-j6qj4</td>
<td>10.2.87.4</td>
<td>172.17.4.203</td>
</tr>
<tr>
<td>rook</td>
<td>rook-api-3722659863-53d33</td>
<td>10.2.87.5</td>
<td>172.17.4.203</td>
</tr>
</tbody>
</table>
Flannel and Calico

Node0
- Pod
- Pod
- Pod
- Flannel
- veth
- vxlan

Node1
- Pod
- Pod
- Pod
- Flannel
- veth
- vxlan
Methods for External Access to Pods

- Kubernetes Service
  - Using Type=NodePort
- Host Networking
  - Kubernetes Pod: hostNetwork: true
- Kernel Bypass
  - Virtual NIC via Intel SPDK
Kubernetes Service - NodePort

- Kube-proxy runs on each node
- Packets are routed internally to the correct node where the pod is running
Kubernetes Service - NodePort - Pros

- Pods can be reached through any node
- Pods can restart, on the same or a different node, and still be reachable at the same IP/TCP address
Kubernetes Service - NodePort - Cons

- Traffic is NATed (source IP, dest TCP port)
- Traffic sent to wrong node is forwarded
  - And encapsulated over vxlan
- service.spec.externalTrafficPolicy=Local
  - Avoids source IP NAT, but…
- Traffic sent to the wrong node is dropped
Client Connects to Wrong Node

Node0
- Pod
- Pod
- Pod

Pod
- veth
- vxlan

ServiceIP
- Kube-proxy

Client

Node1
- Pod
- Pod

Server
- veth

ServiceIP
- Kube-proxy

Client Connects to Wrong Node
Client Connects to Correct Node
Host Networking - Pros

- Exposes the host’s interfaces inside the pod
- No NAT
- Traffic goes directly to the correct node
Host Networking - Cons

- The IP address changes when a pod restarts on a new node
Client Connects with Host Networking

Node0

Pod

Pod

Pod

veth

veth

veth

ServiceIP

Kube-proxy

Flannel

veth

veth

veth

veth

vxlan

Node1

Pod

Pod

ServiceIP

Kube-proxy

Flannel

veth

veth

vxlan

vxlan

Client

Server

Client Connects with Host Networking
Adding keepalived to Host Networking

- K8S contrib project: kube-keepalived-vip
- Manages external Virtual IP (VIP) addresses
- Uses K8S Daemonsets to run on every node
- If a node crashes, VIP is moved to a new node
Adding keepalived - Pros

- If a node crashes, VIP is moved to a new node
  - Connectivity is not tied to any individual node
- External clients only need to know the VIP
Adding keepalived- Cons

- Pod must be constrained to node with the VIP
- VIPs are managed outside of K8S networking
Kernel Bypass

- A virtual NIC is created at the hardware level, and the application manages it directly.
Kernel Bypass - Pros

- Application has direct access to a virtual NIC
- Higher performance
- IP address moves with App
Kernel Bypass - Cons

- Application must manage IP address
- Application needs a TCP/IP implementation
- Requires NIC support
Client Connects via Kernel Bypass

Node0
- Pod
- veth
- Flannel
- Kube-proxy
- ServiceIP

Node1
- Pod
- veth
- Flannel
- Kube-proxy
- ServiceIP

Client
iSCSI Target Backed by Ceph RBD Image

- Export access to Ceph RBD images
  - To clients that are not part of the Ceph cluster
- iSCSI Initiator (client side) is widely implemented
- Options for kernel or user level implementations
Data path: Linux LIO iSCSI target
Kernel Level iSCSI Target - Pros

- Data path is only in the kernel
- No polling needed – interrupt driven
- Uses host networking (or VIPs)
- K8S pod is just for configuration
- No user level memory issues
Kernel Level iSCSI Target - Cons

- Dependent on kernel modules being loaded
- Multiple targets on the same node could be more difficult to configure (from multiple pods)
Data path: Intel SPDK iSCSI target

- **iSCSI initiator**
- IP/TCP
- Ceph Cluster
- **iscsi_tgt Pod**
- **librbd**
User Level iSCSI Target - Pros

- Not dependent on kernel modules
- Multiple pods can be running on the same node
- Easy to update the container
User Level iSCSI Target - Cons

- CPU/memory Usage
  - Linux hugepages
  - Polling for data
Data path: Intel SPDK with Virtual NIC

- iSCSI initiator
- iSCSI tgt
- Pod
- librbd
- IP/TCP
- Ceph Cluster

SDC 2017 Storage Developer Conference. © Quantum Corporation. All Rights Reserved.
Virtual NIC - Pros

- Data goes directly to the application
- No transitions to kernel space
- Application has its own IP address
Virtual NIC - Cons

- Application needs IP/TCP library
- Application must manage its own IP address
- CPU/memory usage
Summary

- We looked at three options for external access:
  - K8S Service, Host Networking, Kernel Bypass
- The best way to reduce latency is to eliminate packet forwarding, encapsulation and NAT
- The semantics of a particular service will often drive the choice
Q & A

Questions?
Resources

- https://rook.io
- http://ceph.com
- https://kubernetes.io
- https://www.docker.com
- http://www.spdk.io
- http://linux-iscsi.org
Resources (continued)

- https://github.com/kubernetes/contrib/tree/master/keepalived-vip