Network Bandwidth-Aware Kubernetes Cluster

Yifeng Xiao, Yang Yu
About Us

Yifeng Xiao

• Senior MTS, VMware China R&D
• Working on scalability and performance of Pivotal Kubernetes Service
• Formerly work on VMware Integrated Container, VMware Integrated OpenStack and VMware Big Data Extension

Yang Yu

• Staff Engineer, VMware China R&D
• Working on VMware Kubernetes products
• Familiar with OpenStack’s networking component Neutron
• Speaker of KubeCon Europe 2018, KubeCon China 2018
• Kubernetes Introduction
• Kubernetes Components
• Kubernetes Standard Scheduler
• Quality of Service in Kubernetes
• Network Features in Kubernetes
• Enable Network Bandwidth QoS
Kubernetes Introduction

Kubernetes is the de facto standard for containerized applications

- Scalability
- Standardized workload
- Multi-Tenants
- Resource management
- Rich APIs
- Ubiquitously available in the cloud
Kubernetes Components

- Container runtime
- kubelet
- etcd
- kube-apiserver
- kube-scheduler
- kube-controller-manager
- kube-proxy
- CNI provider
- CoreDNS
Candidate Nodes go through 3 steps:

- Filter out inappropriate nodes
  - Volume filters
  - CPU/RAM/GPU resource filters
  - Affinity selectors
- Rank the rest nodes
  - Pod replicas distribution
  - Node utilization
  - Balanced resource usage
  - Affinity/Taint Priority
- Select the highest score node

Only CPU and memory are taken into consideration
Quality of Service

There are three QoS classes in Kubernetes:

<table>
<thead>
<tr>
<th></th>
<th>Guaranteed</th>
<th>Burstable</th>
<th>Best-effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>request/limit</td>
<td>request = limit</td>
<td>request &lt; limit</td>
<td>request = N/A</td>
</tr>
<tr>
<td>priority</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
</tbody>
</table>

Only support CPU and memory
Bandwidth sensitive applications:
- Audio/Video streaming applications
- IP telephone
- Web applications, Email
- Interactive online games
- File server, P2P

Network QoS requirements:
- Be scheduled to nodes with sufficient network bandwidth
- Be continuously low network latency and high throughput
Network Features in Kubernetes (2/3)

- CNI plugin
  - Provides L2/L3 connectivity
  - Native Ingress
- Multiple NICs
  - Multus
  - CNI-Genie
  - Networking Service Mesh
- Network Policy
  - Provides micro-segmentation for Pods
- Service mesh
  - Istio
  - Envoy
Network Features in Kubernetes (3/3)

Technologies in virtualized infrastructure:
- PCI Passthrough
- SR-IOV
- Virtual switch hardware acceleration
- DPDK

Technologies in containerized infrastructure:
- device plugins
- Macvlan
Another Approach to Improve Network QoS
Make Kubernetes Smarter

Traffic shaping:
• Reserve network bandwidth on worker nodes
• Ingress/Egress control on Pod

Extend Kubernetes Scheduler:
• Take network resource into account
Physical NIC:
  • Fixed bandwidth

Ingress/Egress control on Pod:
  • Linux Traffic Control
Dedicated virtual NIC for data plane (Pod traffic) and controlled by NSX virtual switch.
Workflow

Worker Node
- kubernetes.io/limit-bandwidth = 2000Mbps

NSX Node Agent
- Register network resource:
  - kubernetes.io/ingress-bandwidth-burst: “900Mbps”

Allocate resource:
- kubernetes.io/ingress-bandwidth-burst: “900Mbps”

Scheduler
- Filter:
  - Find nodes with available resource >= requests
- Prioritize:
  - Rank nodes by node utilization etc.

Scheduler Extender
- Bind

kind: Pod
metadata:
  labels:
    kubernetes.io/ingress-bandwidth-burst: “900Mbps”
  - requests:
    cpu: “100m”
    memory: “128Mi”

Pods
- 300Mbps
- 400Mbps
- 900Mbps

Pod
- 900Mbps
Reserve Bandwidth on Workers

NSX-T Network I/O Control on virtual machine

- Shares - priority weight
- Reservation - the minimum bandwidth, in Mbps
- Limit - the maximum bandwidth, in Mbps

Reserve bandwidth on worker nodes and mark them with annotation

<table>
<thead>
<tr>
<th>Shares</th>
<th>Reservation</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Specified Bandwidth</td>
<td>Same as reservation</td>
</tr>
</tbody>
</table>

Mark work node with labels

- kubernetes.io/limit-bandwidth = Specified Bandwidth
Kubernetes Cluster API
... providerSpec:
  value:
    apiVersion: "vsphereproviderconfig/v1alpha1"
    kind: "VsphereMachineProviderConfig"
    machineSpec:
      datacenter: ""
      datastore: ""
      resourcePool: ""
      networks:
        - networkName: ""
          ipConfig:
            networkType: dhcp
            Bandwidth: ""
      numCPUs: 2
      memoryMB: 2048
Add annotations in Pod spec to request network resource

<table>
<thead>
<tr>
<th>Kubernetes Pod Annotations</th>
<th>NSX-T QoS Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes.io/ingress-bandwidth</td>
<td>Ingress Average Bandwidth</td>
</tr>
<tr>
<td>kubernetes.io/ingress-bandwidth-burst</td>
<td>Ingress Peak Bandwidth</td>
</tr>
<tr>
<td>kubernetes.io/egress-bandwidth</td>
<td>Egress Average Bandwidth</td>
</tr>
<tr>
<td>kubernetes.io/egress-bandwidth-burst</td>
<td>Egress Peak Bandwidth</td>
</tr>
<tr>
<td>kubernetes.io/cos</td>
<td>CoS</td>
</tr>
</tbody>
</table>
Kubernetes allows how to extend standard scheduler with additional filter and prioritize rules

```json
{
    "kind": "Policy",
    "apiVersion": "v1",
    "extenders": [
        {
            "urlPrefix": "http://192.168.0.1:80/scheduler",
            "filterVerb": "filter",
            "prioritizeVerb": "prioritize",
            "weight": 5,
            "enableHttps": false,
            "nodeCacheCapable": false
        }
    ]
}
```
Extend Kubernetes Scheduler (2/2)

Filters

- Volume filters
- CPU/RAM resource filters
- Affinity selectors

Prioritize

- Pod replicas distribution
- Node utilization
- Balanced resource usage

Extend filters and priorities to take network into consideration

http://192.168.0.1:80/scheduler
Algorithm for Extended Rules

Filter out inadequate nodes

- Node Capacity = `kubenetes.io/limit-bandwidth` * 75%
- New Pod request = Max(`kubenetes.io/ingress-bandwidth-burst`, `kubenetes.io/egress-bandwidth-burst`)
- Remaining Capacity = Node Capacity - existing Pods request – new Pod request

<table>
<thead>
<tr>
<th>Remaining Capacity</th>
<th>Greater than or equal 0</th>
<th>Less than 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stay</td>
<td></td>
<td>Remove from candidate list</td>
</tr>
</tbody>
</table>

Prioritize nodes

- Priority Score = ((Node Capacity - sum (Bandwidth Request)) / Node Capacity) * weight
Extended Scheduler

Pod
Request: 600Mbps

Node 1
Capacity: 1000Mbps
Existing Pod request: 300Mbps

Node 2
Capacity: 1000Mbps
Existing Pod request: 500Mbps

Node 3
No bandwidth reservation

Node 4
Capacity: 1000Mbps
Existing Pod request: 0Mbps

Filter

Node 1

Prioritize

Node 1

Node 4

Weight = 10

(1000 - (300 + 600)) / 1000 * 10 = 1

(1000 - 600) / 1000 * 10 = 4
Demonstration

Pods without Network Bandwidth Request
Observed Throughput (Mpbs)

Pods with Network Bandwidth Request
Observed Throughput (Mpbs)
Thank you!