
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

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

Agenda

- Kubernetes Introduction
- Kubernetes Components
- Kubernetes Standard Scheduler
- Quality of Service in Kubernetes
- Network Features in Kubernetes
- Enable Network Bandwidth QoS

Kubernetes Introduction

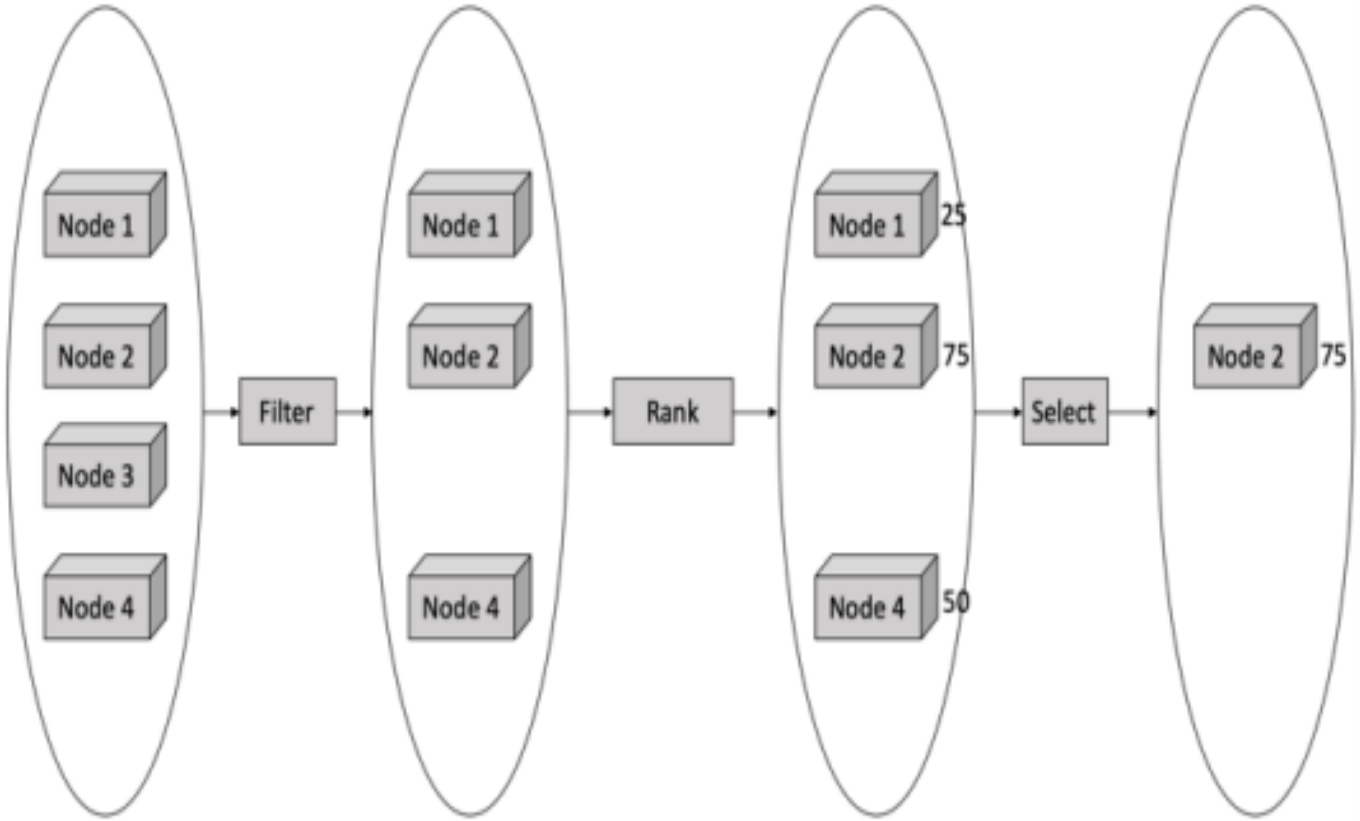
Kubernetes is the de facto standard for containerized applications

- 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

Kubernetes Standard Scheduler



Only CPU and memory are taken into consideration

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

Quality of Service

There are three QoS classes in Kubernetes:

	Guaranteed	Burstable	Best-effort
request/limit	request = limit	request < limit	request = N/A
priority	high	medium	low

Only support CPU and memory

Network Features in Kubernetes (1/3)

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

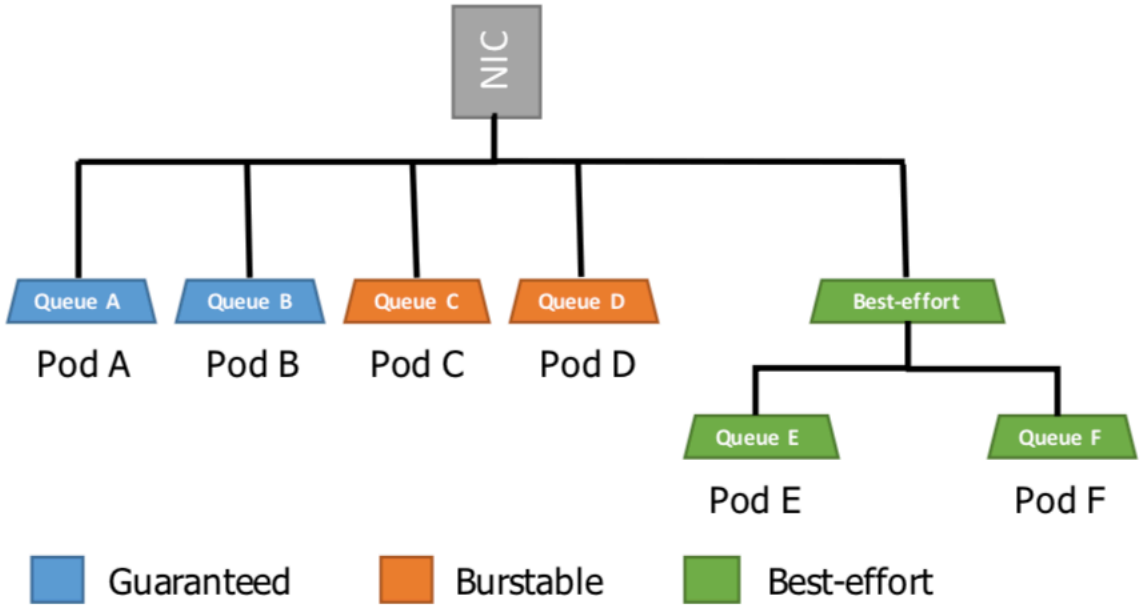
BareMetal Deployment

Physical NIC:

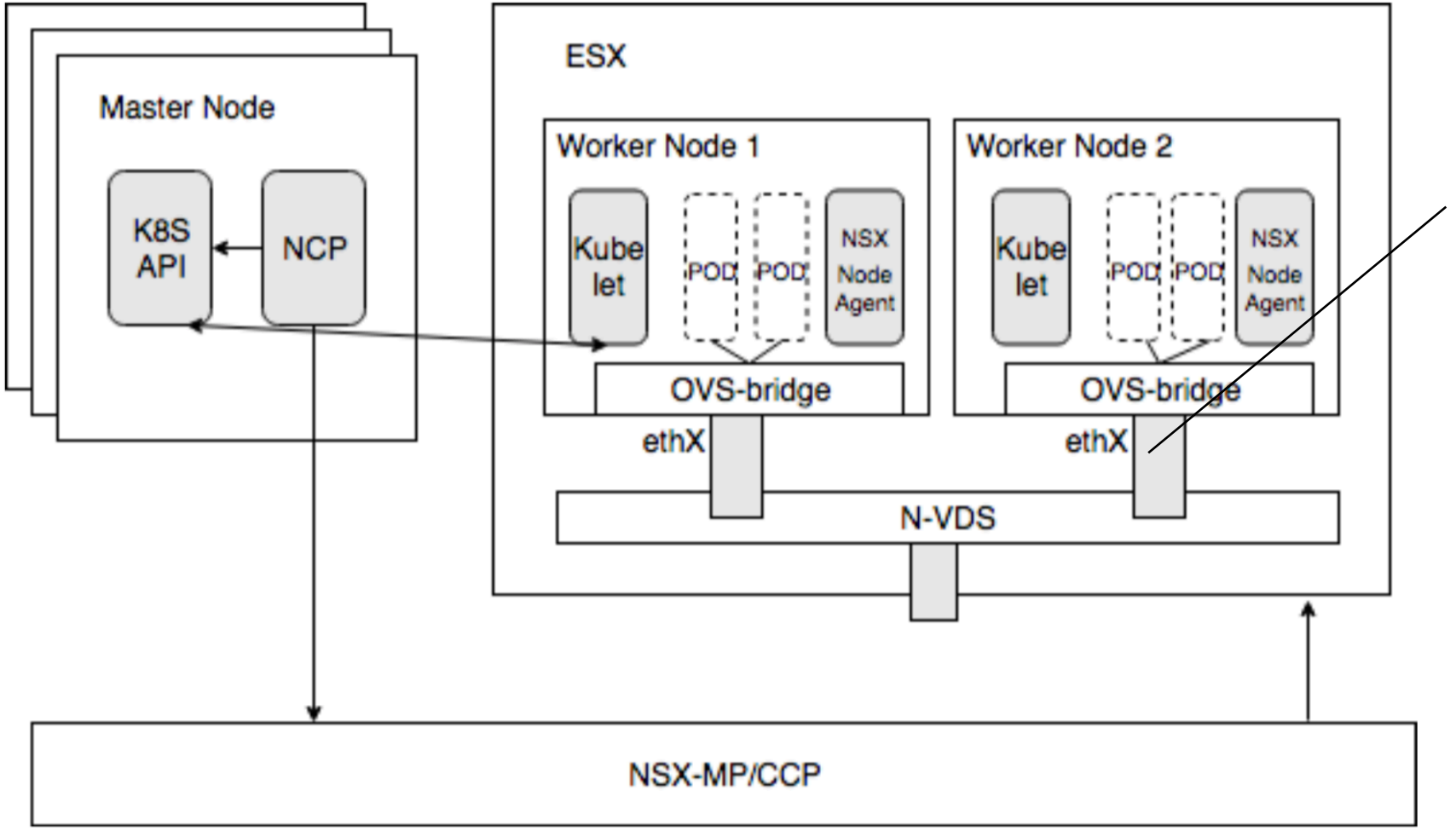
- Fixed bandwidth

Ingress/Egress control on Pod:

- Linux Traffic Control



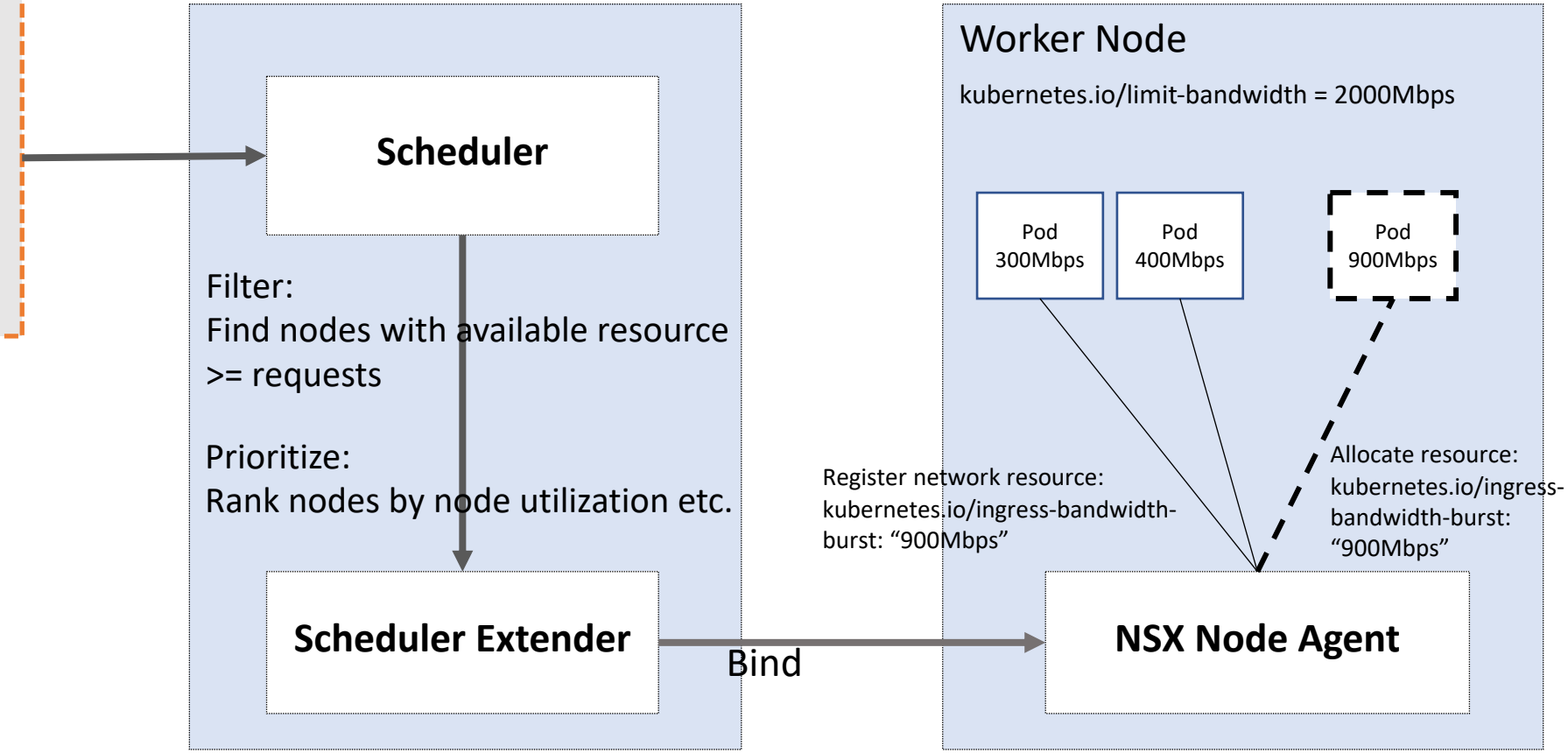
Virtual Machine Deployment
Kubernetes Cluster on vSphere with NSX-T



Dedicated virtual NIC for data plane(Pod traffic) and controlled by NSX virtual switch

Workflow


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



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

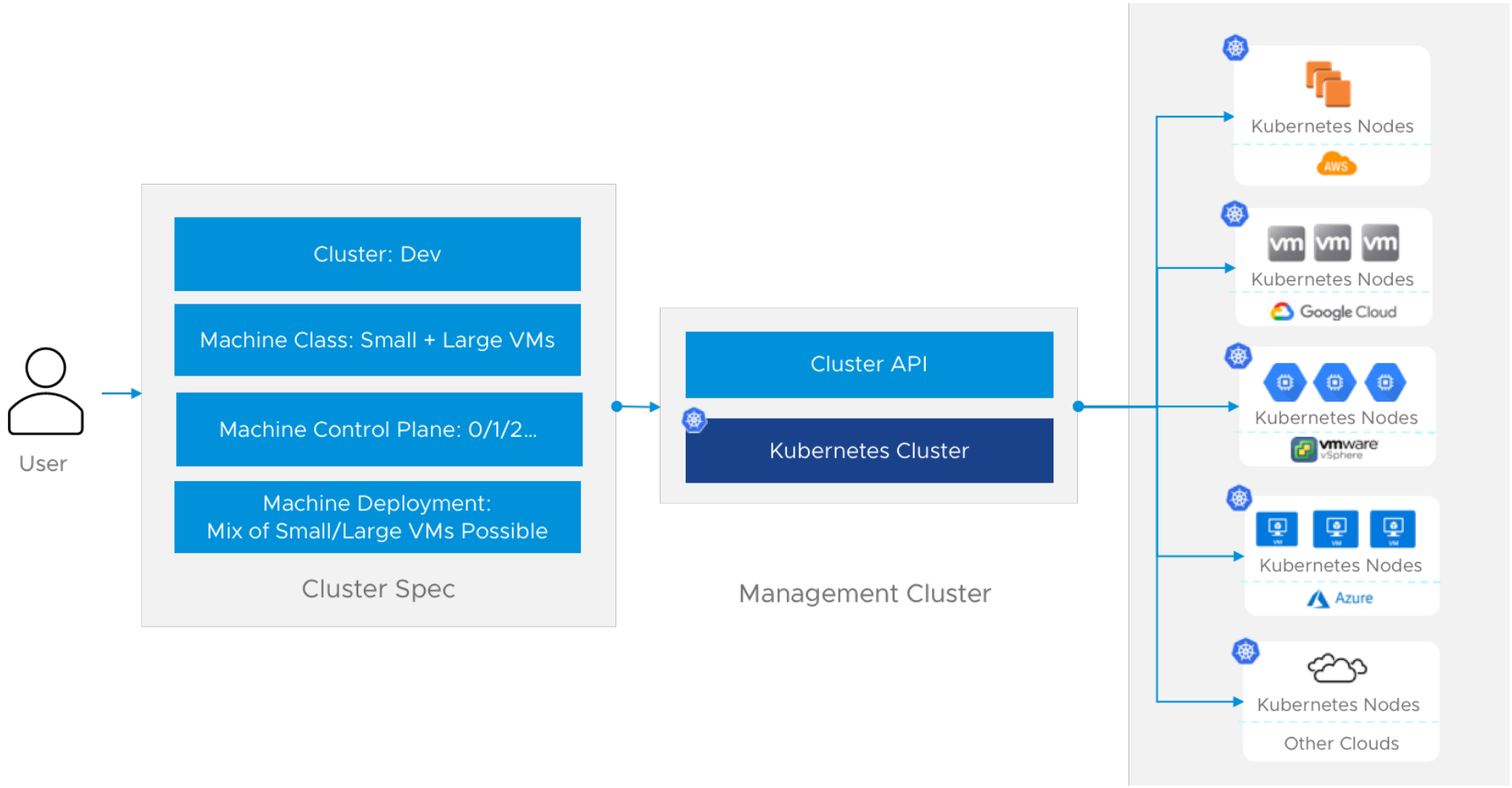
Reserve bandwidth on worker nodes and mark them with annotation

Shares	Reservation	Limit
50	Specified Bandwidth	Same as reservation

Mark work node with labels

- kubernetes.io/limit-bandwidth = Specified Bandwidth

Kubernetes Cluster API



Kubernetes Cluster API


```
...
    providerSpec:
     value:
       apiVersion: "vsphereproviderconfig/v1alpha1"
       kind: "VsphereMachineProviderConfig"
       machineSpec:
         datacenter: ""
         datastore: ""
         resourcePool: ""
         networks:
         - networkName: ""
           ipConfig:
             networkType: dhcp
           Bandwidth: ""
         numCPUs: 2
         memoryMB: 2048
```

Mark Pods with Resource Request

Add annotations in Pod spec to request network resource

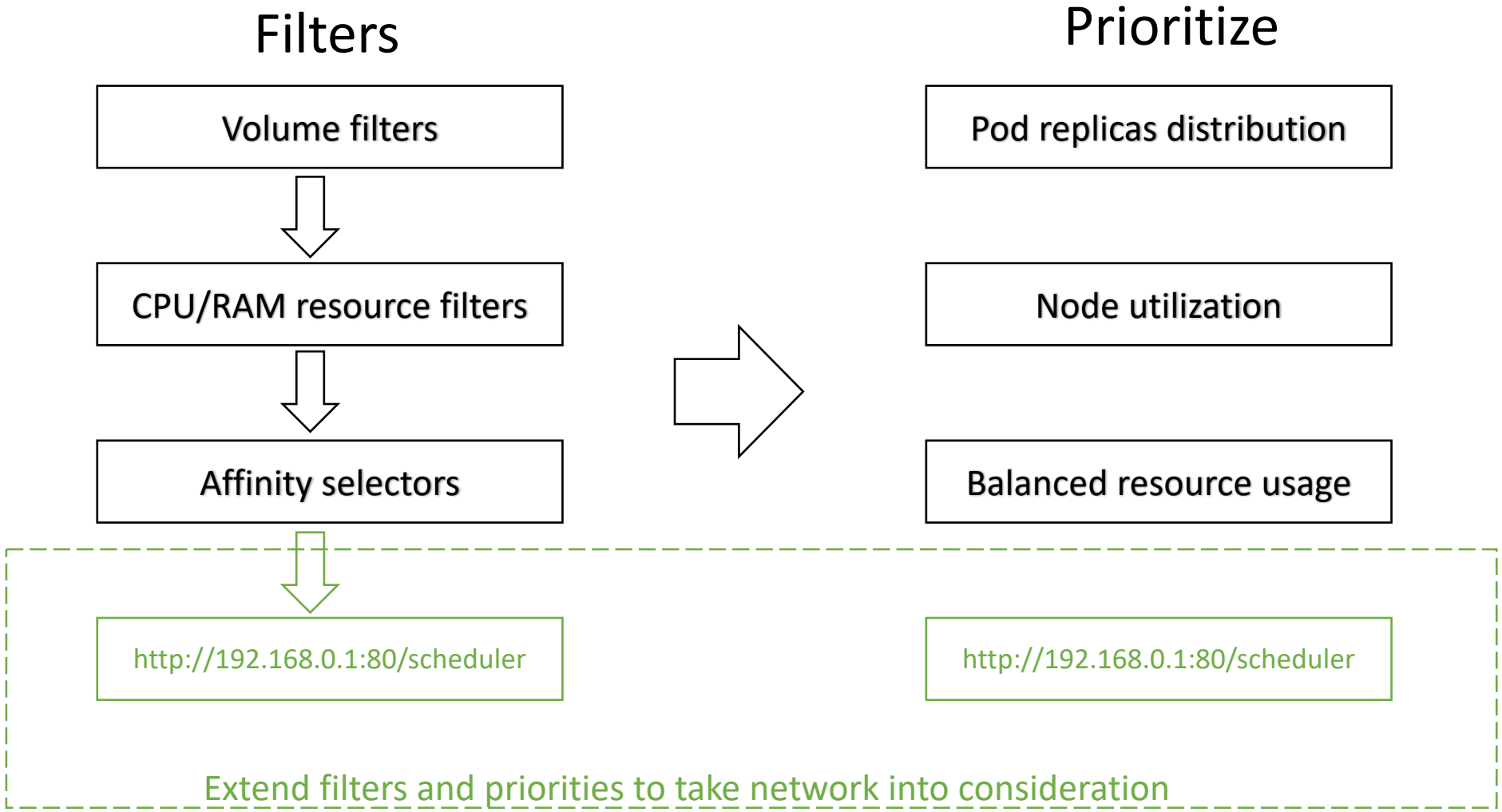
Kubernetes Pod Annotations	NSX-T QoS Properties
kubernetes.io/ingress-bandwidth	Ingress Average Bandwidth
kubernetes.io/ingress-bandwidth-burst	Ingress Peak Bandwidth
kubernetes.io/egress-bandwidth	Egress Average Bandwidth
kubernetes.io/egress-bandwidth-burst	Egress Peak Bandwidth
kubernetes.io/cos	CoS

Extend Kubernetes Scheduler(1/2)

Kubernetes allows how to extend standard scheduler with additional filter and prioritize rules

```
{
  "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)



Algorithm for Extended Rules

Filter out inadequate nodes

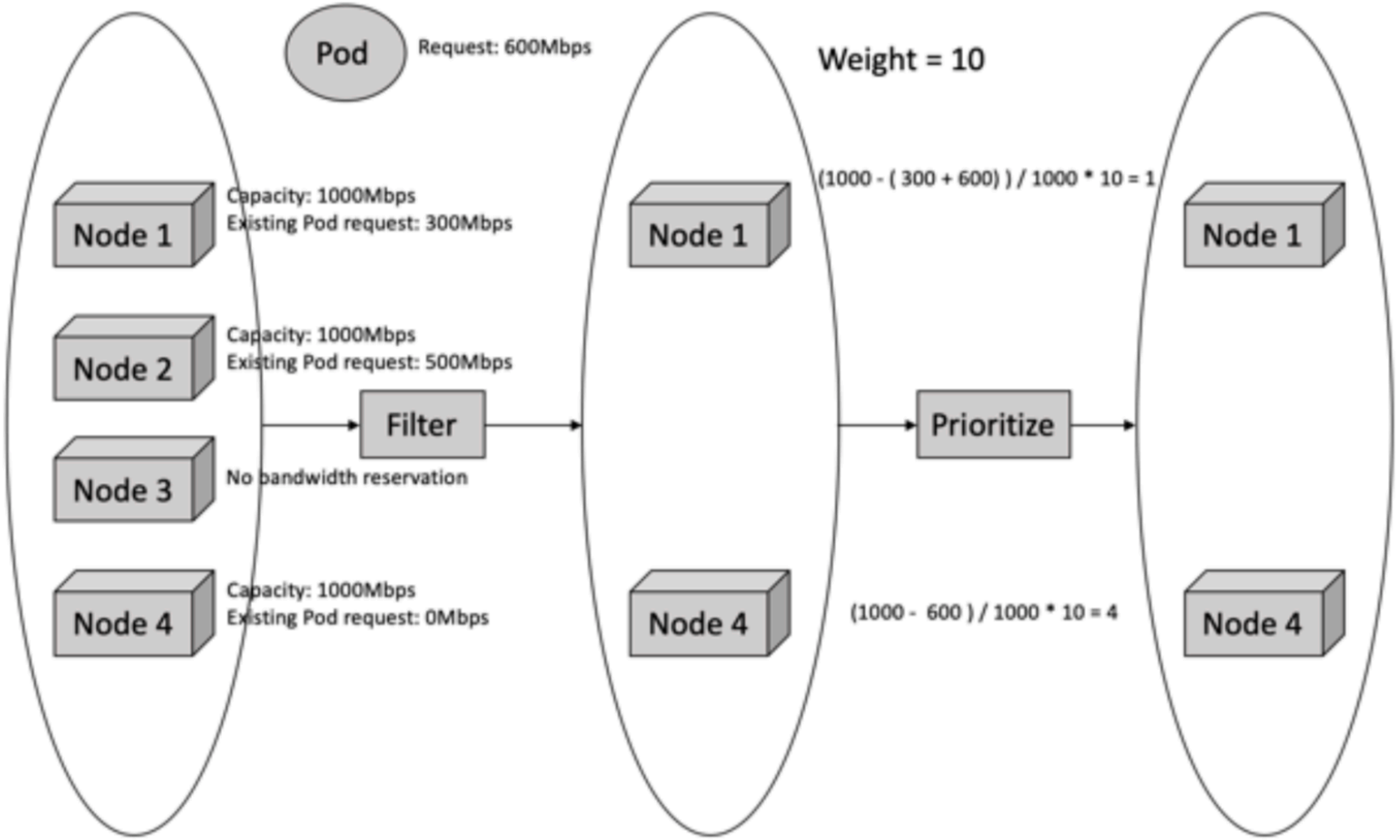
- Node Capacity = kubenetes.io/limit-bandwidth * 75%
- New Pod request = Max (kubenetes.io/ingress-bandwidth-burst, kubenetes.io/egress-bandwidth-burst)
- Remaining Capacity = Node Capacity - existing Pods request – new Pod request'

	Greater than or equal 0	Less than 0
Remaining Capacity	Stay	Remove from candidate list

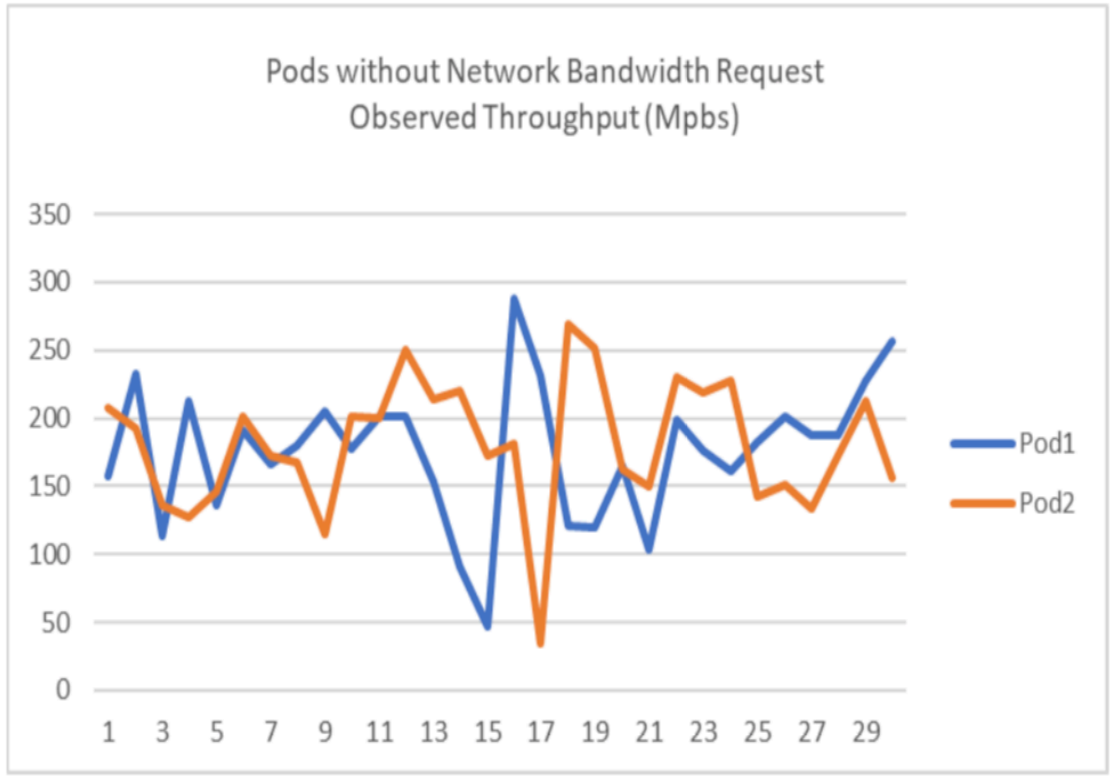
Prioritize nodes

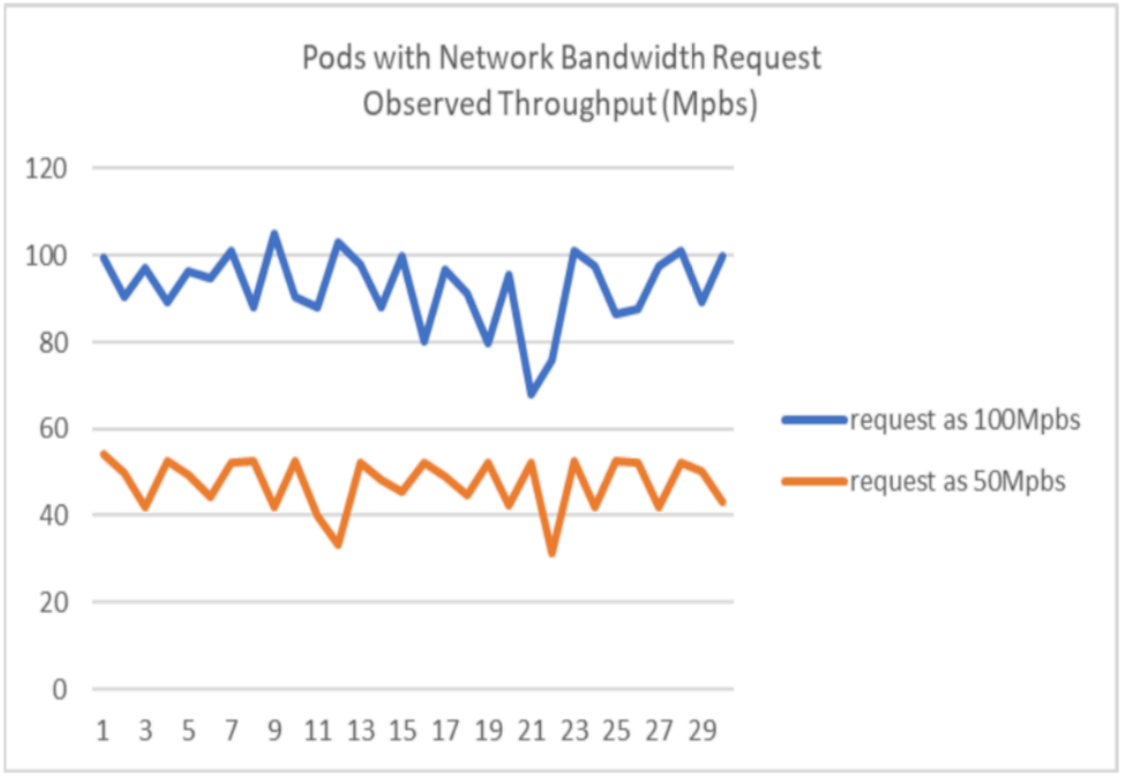
- Priority Score = ((Node Capacity - sum (Bandwidth Request)) / Node Capacity) * weight

Extended Scheduler



Demonstration





Thank you!