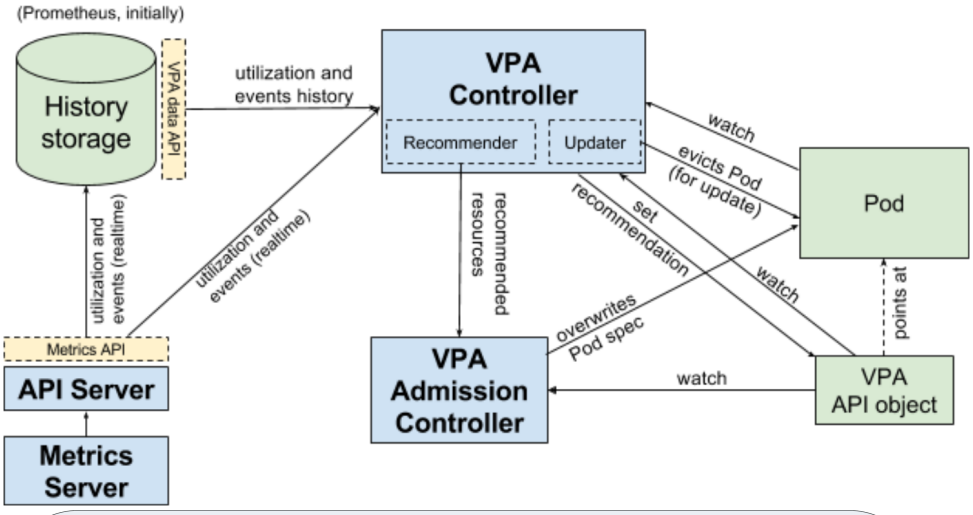
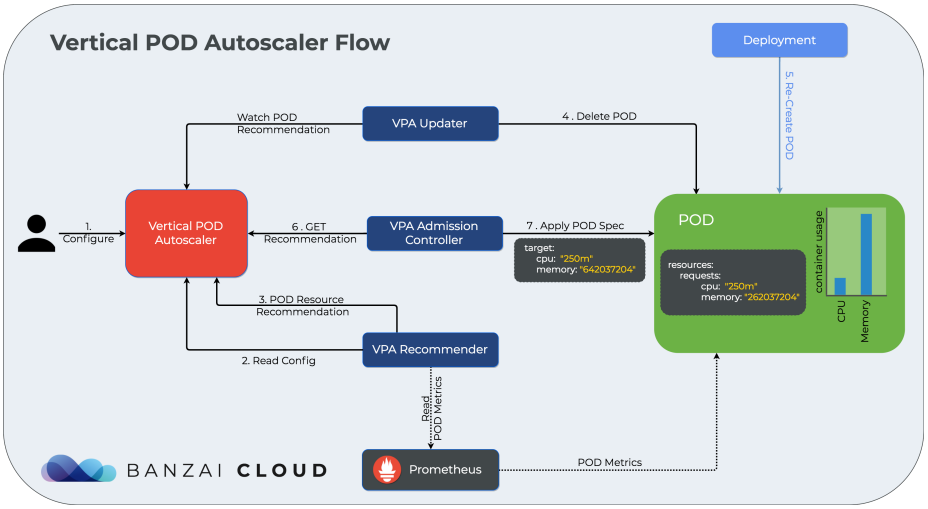
K8S VPA介绍

1 VPA基本介绍

VPA(Vertical Pod Autoscaler)的工作原理图如下所示:





1.1 VPA功能

自动调整容器的CPU/MEM request值

1.2 VPA意义

- 不需要通过运行Benchmark来确定合适的CPU/MEM request值
- 不需要用户手工调整Pod的CPU/MEM request，VPA会自动调整
- 提高集群节点利用率

1.3 VPA基本原理

根据获取Pod的实时metircs(从metrics-server获取CPU和MEM)，自动调整(调大或调小）Pod的CPU/MEM request值

1.4 VPA使用场景

Pod的CPU和MEM在不同时段的使用率差异大的情况下，利用VPA即可解决Pod的运行稳定性又可提高集群节点利用率

1.5 VPA实现方式

VPA通过CRD(CustomResourceDefinitions)实现，所以无法在K8S官网查看VPA的API使用，可通过VPA源码查看API定义

1.6 VPA支持模式

- Auto：
  - 更新“新建的”和“已存在的”Pod CPU/MEM request
  - 当前不支持“in-place”更新，通过kill Pod使其重新创建pod来更新CPU/MEM request
  - 当前Auto完全等同于Recreate，在支持”in-place“更新后，Auto将不会kill Pod，而Recreate将总是kill Pod
  - 当前默认模式是Auto
- Recreate：如上描述
- Initial：只在“新建”Pod时会更新CPU/MEM request，不会针对已存在的Pod进行更新
- Off：不对Pod的CPU/MEM request进行更新，只计算“推荐值”放到VPA对象供查看

1.7 VPA核心组件

- Recommender:
  - 根据历史和当前的实时metrics信息，计算Pod的推荐CPU/MEM request值，然后插入到VPA对象中
  - 历史信息可以通过配置Prometheus获取(具体配置后面介绍)，仅在Recommender启动的初始化时使用
  - 更多细节：https://github.com/kubernetes/autoscaler/blob/master/vertical-pod-autoscaler/pkg/recommender/README.md
- Updater:
  - 检查Pod当前的CPU/MEM request和Recommender推荐的CPU/MEM差异是否过大，如果过大，则驱除Pod(遵守PDB设置)
  - 更多细节：https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler/pkg/updater
- Admission Plugin:
  - 根据Recommender推荐的CPU/MEM request修改新建Pod的CPU/MEM request，然后Pod将根据Recommender的推荐值创建
  - 更多细节：https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler/pkg/admission-controller

1.8 VPA实现细节

- VPA与HPA不同，VPA并不会修改Controller(如Deployment)中的资源请求
  - Admission Plugin通过向APIServer注册了Webhook，在每次Pod创建时，会检查当前要创建的Pod是否匹配VPA对象，如果匹配则使用Recommender推荐的资源请求进行修改
  - 由于VPA不会修改Controller中的资源请求，所以当VPA被删除后，当前已存在Pod的资源请求不会发生变化(即Pod不会被自动更新资源请求），但是新建的Pod会使用Controller中定义的资源请求进行创建
- VPA只根据Pod的CPU/MEM request设置(不看limit设置）来决定是否进行更新。如果Pod配置了Limit，VPA在更新request值时会同时更新limit值，确保limit和request的比例与更新之前一致
- VPA在没有配置资源策略下，其推荐值总是将符合Limit Range的设置
- VPA在配置资源策略下(如Pod的最小CPU request，在VPA中进行配置，后面会介绍)，其推荐值将覆盖Limit Range的设置
- VPA遵守PDB(PodDisruptionBudget)设置

2 VPA安装和使用

2.1 安装

- 下载autoscale仓库：

```
# git clone https://github.com/kubernetes/autoscaler.git
```

- 安装VPA：

```
# cd autoscaler/vertical-pod-autoscaler
# ./hack/vpa-up.sh
```

说明：

  - 如果当前环境已经部署老版本VPA，可以通过执行`./hack/vpa-down.sh`进行卸载
  - vpa-up.sh会先生成相关证书，然后apply vertical-pod-autoscaler/deploy下的yaml文件
  - 可以指定`$REGISTRY`、`$TAG`、yaml`REGISTRY`、`k8s.gcr.io`、`TAG`、`0.6.3`

-

```
# kubectl get pods -n kube-system | grep vpa-
vpa-admission-controller-74d96d5b75-tvbcw   1/1     Running   0          20h
vpa-recommender-56497bb544-g2jmr            1/1     Running   0          72m
vpa-updater-6596964bdf-4bbqj                1/1     Running   0          20h
# kubectl get crd | grep verticalpodauto
verticalpodautoscalercheckpoints.autoscaling.k8s.io   2019-11-26T07:08:53Z
verticalpodautoscalers.autoscaling.k8s.io             2019-11-26T07:08:53Z
# kubectl get mutatingWebhookConfiguration | grep vpa-
vpa-webhook-config              2019-11-26T07:09:08Z
# kubectl get svc -n kube-system | grep vpa-
vpa-webhook          ClusterIP   11.111.233.184   <none>        443/TCP                  20h
#
```

2.2 使用

- 部署一个deployment和一个VPA:

```
# cat examples/hamster.yaml
# This config creates a deployment with two pods, each requesting 100 millicores
# and trying to utilize slightly above 500 millicores (repeatedly using CPU for
# 0.5s and sleeping 0.5s).
# It also creates a corresponding Vertical Pod Autoscaler that adjusts the
# requests.
# Note that the update mode is left unset, so it defaults to "Auto" mode.
---
apiVersion: "autoscaling.k8s.io/v1beta2"
kind: VerticalPodAutoscaler
metadata:
  name: hamster-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind: Deployment
    name: hamster
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hamster
spec:
  selector:
    matchLabels:
      app: hamster
  replicas: 2
  template:
    metadata:
      labels:
```

```
        app: hamster
    spec:
      containers:
        - name: hamster
          image: k8s.gcr.io/ubuntu-slim:0.1
          resources:
            requests:
              cpu: 100m                         <===
              memory: 50Mi                      <===
            limits:
              cpu: 1                            <===
              memory: 500Mi                     <===
          command: ["/bin/sh"]
          args:
            - "-c"
            - "while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done"
# kubectl apply -f examples/hamster.yaml
verticalpodautoscaler.autoscaling.k8s.io/hamster-vpa created
deployment.apps/hamster created
#
```

说明：注意当前的CPU/MEM的request和limit设置

- 查看状态：

```
# kubectl get pods | grep hamster
hamster-6748bf6d5f-4qxpf     1/1     Running   0          34s
hamster-6748bf6d5f-8h287     1/1     Running   0          34s
# kubectl get vpa/hamster-vpa
NAME          AGE
hamster-vpa   42s
# kubectl describe vpa/hamster-vpa
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/v1beta2","kind":"VerticalPodAutoscaler","metadata":
{"annotations":{},"name":"hamster-vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-11-27T06:25:07Z
  Generation:          2
  Resource Version:    2196677
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/verticalpodautoscalers
/hamster-vpa
  UID:                 0f753177-d8b6-4d63-b278-eb1d0174f6e6
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
  Update Policy:
    Update Mode:  Auto
Status:
  Conditions:
    Last Transition Time:  2019-11-27T06:25:33Z
    Message:               No pods match this VPA object         <======
    Reason:                NoPodsMatched
    Status:                True
    Type:                  NoPodsMatched
    Last Transition Time:  2019-11-27T06:25:33Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
```

```
        Cpu:     583m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     2484m
        Memory:  262144k
Events:          <none>
#
```

说明：关于Status中偶现的"NoPodsMatched"信息，社区的回复是: "a reporting problem not a problem with a logic"，详细：https://github.com/kubernetes/autoscaler/issues/2465

- 观察Pod的变化(等2~5分钟)：

```
# kubectl get pods -w
hamster-6748bf6d5f-h5vjr     1/1     Running   0          1m
hamster-6748bf6d5f-w4mng     1/1     Running   0          1m
```

- 查看VPA更新后的Pod资源请求：

```
# kubectl describe pod hamster-6748bf6d5f-h5vjr
Name:           hamster-6748bf6d5f-h5vjr
Namespace:      default
Priority:       0
Node:           comp-stor.localdomain/192.168.50.113
Start Time:     Wed, 27 Nov 2019 14:26:58 +0800
Labels:         app=hamster
                pod-template-hash=6748bf6d5f
Annotations:    vpaUpdates: Pod resources updated by hamster-vpa: container 0: cpu request, memory
request, cpu limit, memory limit
Status:         Running
IP:             193.168.12.10
Controlled By:  ReplicaSet/hamster-6748bf6d5f
Containers:
  hamster:
    Container ID:  docker://731daf6251c00f332078eda52e3efbb0a48da9e99af49544353b2cb84ee4feeb
    Image:         k8s.gcr.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://k8s.gcr.io/ubuntu-slim@sha256:
b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Wed, 27 Nov 2019 14:26:59 +0800
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     5870m                  <===
      memory:  2500Mi                 <===
    Requests:
      cpu:        587m                <===
      memory:     262144k             <===
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-bq857 (ro)
...
```

说明：可以发现request和limit值都发生了变化，但limit和request之间的比列没有变化

2.3 卸载

```
./hack/vpa-down.sh
```

3 VPA高级配置

- 为了支持对replicas为1的pod进行自动更新，进行如下配置：

```
+++ b/vertical-pod-autoscaler/deploy/updater-deployment.yaml
@@ -24,6 +24,8 @@ spec:
       containers:
         - name: updater
           image: k8s.gcr.io/vpa-updater:0.6.3
+          args:
+            - "--min-replicas=1"
           imagePullPolicy: Always
           resources:
             limits:
```

说明：

  - `vpa-updatermin-replicas2replicas2pod`
  - 更多细节：https://github.com/kubernetes/autoscaler/issues/2388

- 为了支持从prometheus获取历史数据，进行如下配置：

```
--- a/vertical-pod-autoscaler/deploy/recommender-deployment.yaml
+++ b/vertical-pod-autoscaler/deploy/recommender-deployment.yaml
@@ -24,6 +24,10 @@ spec:
       containers:
       - name: recommender
         image: k8s.gcr.io/vpa-recommender:0.6.3
+        args:
+        - --v=4
+        - --storage=prometheus
+        - --prometheus-address=http://prometheus.skydiscovery.svc.cluster.local:9090
         imagePullPolicy: Always
         resources:
           limits:
```

说明：

  - 如上示例表示prometheus被部署在skydiscovery namespace，service监听端口号为9090
  - prometheus从cadvisor获取的metrics需要有label： job=kubernetes-cadvisor
  - vpa-recommender在启动时会输出信息："Initializing VPA from history provider"
  - 更多细节：https://github.com/kubernetes/autoscaler/blob/master/vertical-pod-autoscaler/FAQ.md

- 支持VPA资源配置

```
# cat vpa-resource-policy.yaml
---
apiVersion: "autoscaling.k8s.io/v1beta2"
kind: VerticalPodAutoscaler
metadata:
  name: hamster-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind: Deployment
```

```
    name: hamster
  updatePolicy:
    updateMode: "Auto"
  resourcePolicy:
    containerPolicies:
    - containerName: hamster
      maxAllowed:
        cpu: 0.5
        memory: 200Mi
```

说明：

- containerName必须要配置
- 配置表明VPA matched的Pod的CPU request最大为0.5， MEM request最大为200Mi

4 VPA已知限制

- VPA更新需要Pod被重新创建，创建后的Pod可能与之前运行在不同的节点
- VPA更新是通过驱除Pod来实现的，但如果Pod没有Controller(如Deployment)，那么Pod将不会被自动新建，所以VPA不会驱除一个不在任何Controller下的Pod
- VPA推荐的资源请求可能超过当前系统可用资源，因此会出现Pod被驱除后无法成功调度，一直处于Pending状态
- VPA不能和基于CPU/MEM的HPA一起使用，VPA可以和基于custom和external的HPA一起使用
- 一个Pod不能匹配多个VPA对象，否则可能出现不确定的行为

5 VPA未来工作

- 支持“in-place”更新资源请求
- 支持VPA和HPA(基于CPU和MEM)一起使用
- 在Pod中添加配置，只有存在匹配的VPA才可以创建

6 参考

- https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler
- https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/vertical-pod-autoscaler.md
- https://cloud.google.com/kubernetes-engine/docs/concepts/verticalpodautoscaler?hl=zh-cn
- https://banzaicloud.com/blog/k8s-vertical-pod-autoscaler/
- https://medium.com/magalix/kubernetes-autoscaling-101-cluster-autoscaler-horizontal-pod-autoscaler-and-vertical-pod-2a441d9ad231
- http://bazingafeng.com/2019/04/06/k8s-vpa/
- https://cizixs.com/2018/06/25/kubernetes-resource-management/