K8s Vertical Pod Autoscaling (VPA)

Overview

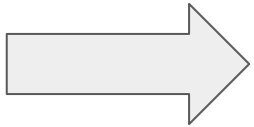
- Serves as a resource definition controller which can recommend values for CPU and memory requests/limits per pod OR can automatically update these values
- Services Running
  - VPA admission hook - similar to our “EC Deployer”: checks whether VPA object is referencing this pod
  - VPA recommender - connects to metrics-server and gets historical + current usage data for pods
  - VPA updater - Runs every 1 minute and check if pod is running in calculated recommended range. If not, then kills the pod before restarting it.

Configuring VPA

- Possible Policies - All focuses on Requests (not limits):
  - **Off**: Simply recommends the best estimate for resources per pod- used for a “dry run
  - **Initial**: Autoscaler only assigns resources on Pod creation and doesn’t change them during lifetime
  - **Auto**: Autoscaler adjusts container resources of running pods by destroying existing pod and recreating it
- Original Deployment spec will be left unchanged (meaning a difference in what is defined and what is running)
  - By default, VPA focuses on identifying ideal request values and so “limits” are proportionally scaled.

Example

```yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: my-rec-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind:       Deployment
    name:       my-rec-deployment
  updatePolicy:
    updateMode: "Off"
```



```yaml
...
  recommendation:
    containerRecommendations:
    - containerName: my-rec-container
      lowerBound:
        cpu: 25m
        memory: 262144k
      target:
        cpu: 25m
        memory: 262144k
      upperBound:
        cpu: 7931m
        memory: 8291500k
...
```

Also supports minimum and maximum requests for the autoscaler (i.e. `minAllowed`, `maxAllowed`

Recommended Values

- **Lower Bound**: When Pod goes below this usage, it will be killed and downscaled
- **Upper Bounds:** When Pod goes above this usage, will be killed and upscaled
- **Target:** Actual amount of resources configured at the next execution of admission
- **Uncapped Target:** ideal resource requests configured if no upper limits provided in VPA definition

Under the Hood

Recommendation Model

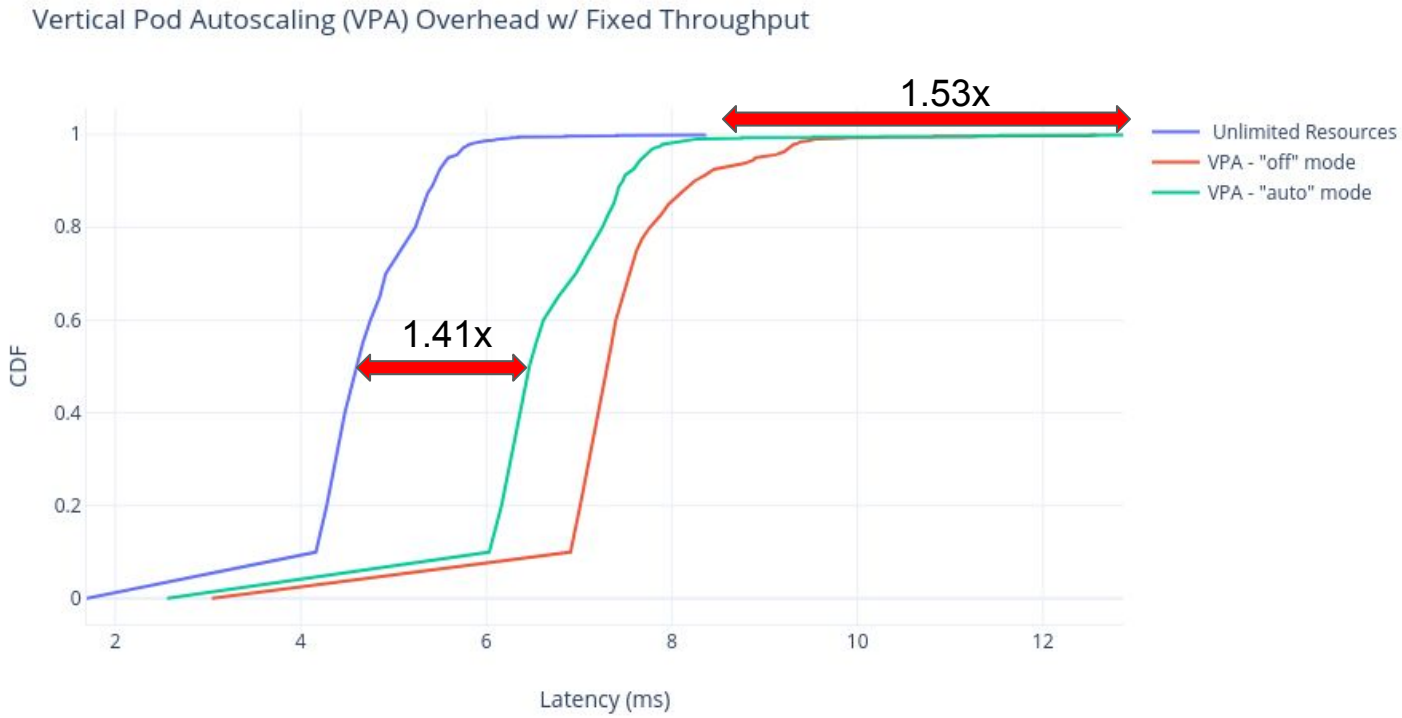
- CPU: Goal is to keep the fraction of time when the container usage exceeds a high percentage (e.g. 95%) of request below a certain threshold (i.e. 1%)
  - CPU Usage is mean usage over a time interval (recommended to be 1/sec)
  - **Open Question: How do you adjust the recommendation before assigning it to a specific pod, based on current state of cluster? (i.e. quota and or available memory)**

- Mem: Goal is to keep the probability of the container usage exceeding the request in a specific time window below a certain threshold (i.e 1%)
  - Ideal to keep the time window in this case long (i.e. a day) to not violate SLOs

Pitfalls + Limitations

- Due to limitations, only way to modify resource requests of a running pod is to recreate the Pod - meaning pod has to be evicted to change resource requests (In-Place updates are merged but inconsistent performance [2])
- Cannot be used with Horizontal Pod Autoscaler (results are unpredictable)
- VPA recommendation might exceed available resources and cause pods to go *pending*
    - Due to lack of application knowledge. I.e. scaling one pod upwards doesn’t scale another pod in the deployment to be scaled downwards

Results - Measuring overhead



References

1. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/vertical-pod-autoscaler.md#recommendation-model
2. https://github.com/kubernetes/enhancements/pull/686#
3. https://cloud.google.com/kubernetes-engine/docs/concepts/verticalpodautoscaler
4. https://cloud.google.com/kubernetes-engine/docs/how-to/vertical-pod-autoscaling
5. https://medium.com/infrastructure-adventures/vertical-pod-autoscaler-deep-dive-limitations-and-real-world-examples-9195f8422724
6. https://github.com/kubernetes/autoscaler/blob/84cbb3bc7923d56c6cffe1b117cb89cb91820243/cluster-autoscaler/core/static_autoscaler.go#L215
7. https://www.openshift.com/blog/how-full-is-my-cluster-part-4-right-sizing-pods-with-vertical-pod-autoscaler
8. https://livewyer.io/blog/2019/06/24/vertical-pod-autoscaling/
9. https://docs.aws.amazon.com/eks/latest/userguide/vertical-pod-autoscaler.html