Cluster Capacity Issues

Introduction

The RHACM observability dashboard is either incorrect or can be misleading with respect to CPU request as a percentage of allocatable CPU within the cluster. This is a key metric for cluster health and capacity management, so it is essential that this is accurate.

A locally customised prometheus expression appears to be more accurate and can be further developed and integrated into our cluster fleet management.

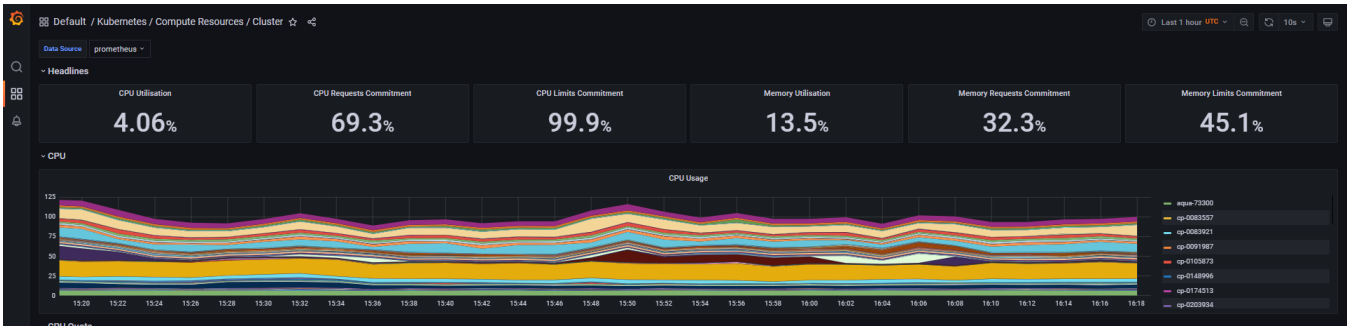
We will raise a case with Red Hat for RCA on the dashboard metrics & suggestions on an ideal metric for alerting and capacity management of our fleet of clusters.

Diagnosis

Ilustration from a production cluster (useast16), using RHACM grafana dashboard

This shows near linear growth of CPU requests due to "completed" jobs and shows the current value to be well over 100% (112%) ; which is patently inaccurate as the kubernetes scheduler will not overallocate cpu_requests.

The local openshift grafana dashboard is showing a current snapshot value of 69.3% which also contradicts the RHACM observability dashboard.



The locally modified Prometheus expression appears more accurate , but also differs from the local openshift grafana dashboard:

Using the PROMQL below shows a cpu request utilisation of 76% of available & allocatable cpu resource as compared to 69.3% from openshift grafana & 112% from RHACM grafana

Modified prometheus expression for accurate utilisation of CPU requests as a percentage of allocatable resource

```
sum((kube_pod_container_resource_requests{resource="cpu"} * on (pod,namespace) group_left (phase)
kube_pod_status_phase{phase="Running"}) * on (node) group_left (role) kube_node_role{role="app"} )/
sum(kube_node_status_allocatable{resource="cpu"} * on (node) group_left(role) kube_node_role{role="app"})
```

Notes:

The expression attempts to be more accurate by:

- Aggregating cpu requests, **kube_pod_container_resource_requests{resource="cpu"}** , for pods in "Running" state only & for pods running on "APP" nodes.
- The cpu allocatable resource is also modified by only inclusing resources from "app" worker nodes.

While this is a custom prometheus expression for our "app" worker nodes, we will ask Red Hat to see if this can be improved?

Lower Lane Environment testing

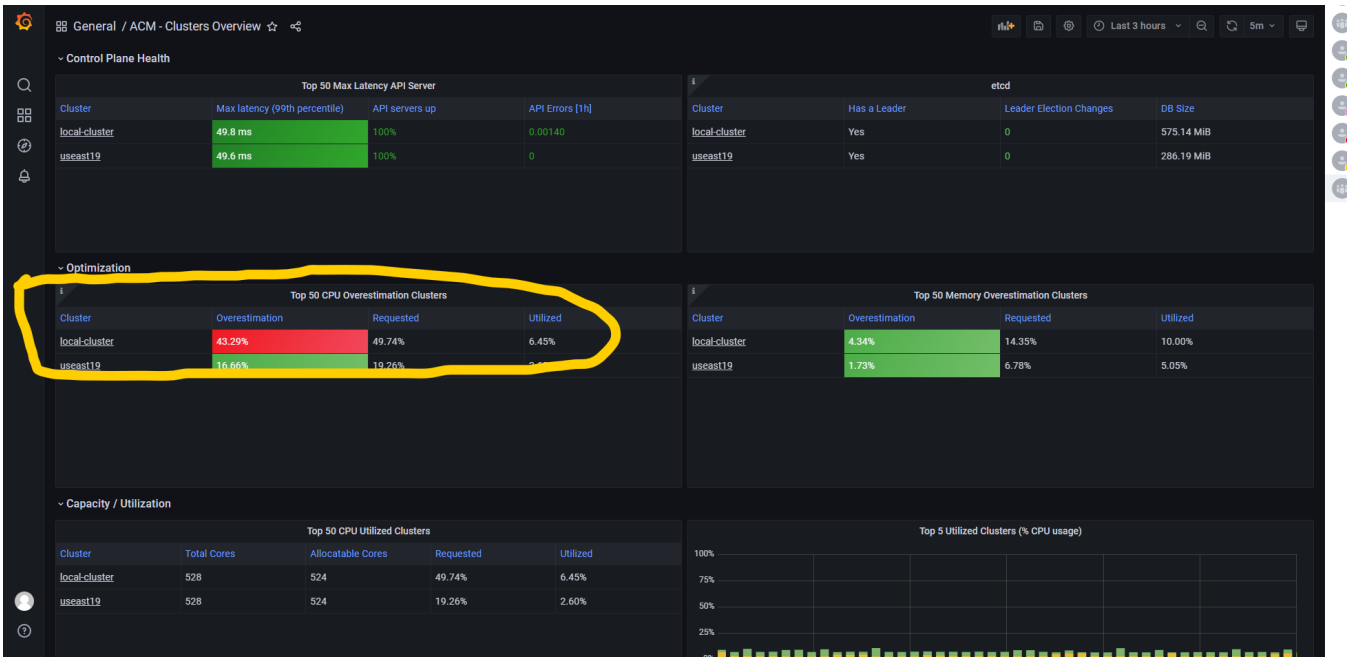
A more extreme example test in a lower lane environment where we used a cronjob to continuously run with a high CPU request; the accumulated completed jobs demonstrate that the cpu requests that accumulate are not actually impacting the cluster.

Sample cronjob details:

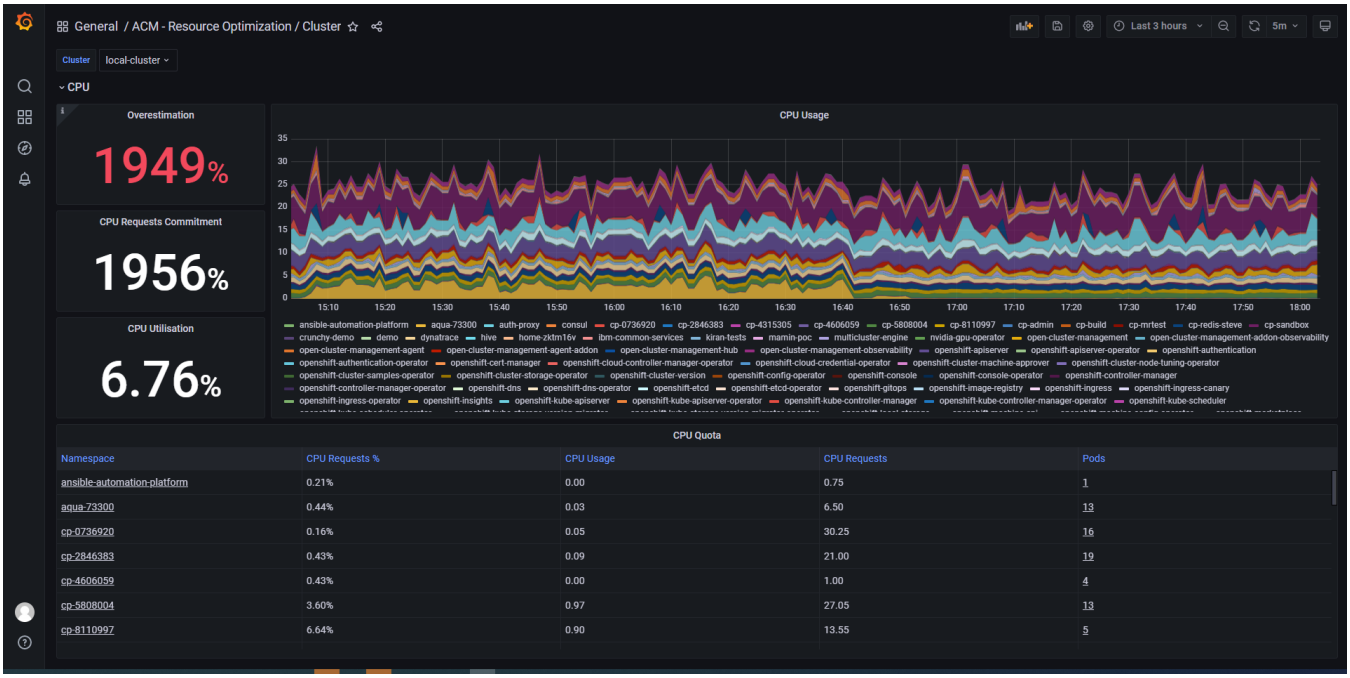
sample cronjob

```
apiVersion: batch/v1
kind: CronJob
metadata:
  creationTimestamp: "2023-03-06T17:17:10Z"
  generation: 5
  name: example1
  namespace: zkys6ky-gpu-namespace
  resourceVersion: "791697289"
  uid: 93c5a6bb-40d6-4a47-be21-45bd99822a1c
spec:
  concurrencyPolicy: Forbid
  failedJobsHistoryLimit: 9999999
  jobTemplate:
    metadata:
      creationTimestamp: null
    spec:
      template:
        metadata:
          creationTimestamp: null
        spec:
          containers:
          - args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
            image: registry-eng.sdi.corp.bankofamerica.com/bac/ubi8-minimal:latest
            imagePullPolicy: Always
            name: hello
            resources:
              requests:
                cpu: "20"
                memory: "100"
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
          dnsPolicy: ClusterFirst
          restartPolicy: Never
          schedulerName: default-scheduler
          securityContext: {}
          terminationGracePeriodSeconds: 30
  schedule: '* * * * *'
  successfulJobsHistoryLimit: 9999999
  suspend: false
```

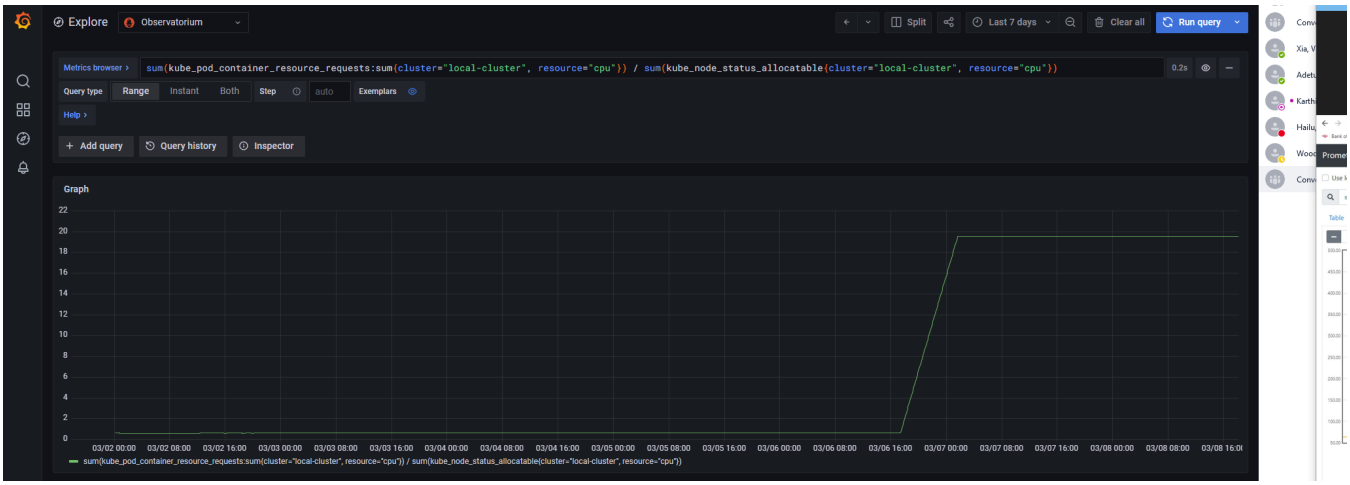
CPU request commitment after 24 hours, shown by the overview RHACM observability dashboard looks normal at 49.74% for "local-cluster":



However, if we drill down, into local-cluster, we see incorrect values for Overestimation & CPU Requests Commitment of 1956% :



If we "explore" the metric "CPU requests Commitment" and display a range we see the effect of our test ranging from approximately 50% to 1950%



Conclussion:

I think this clearly demonstrates that thre is a bug in the RHACM observability dashboards.

We also want to confirm the best metrics to monitor cpu requests as a real percentage of available space.

The metric below is from the "kubernetes/Compute Resources/Cluster" dashboard shows 49.84% , which matches the RHACM overview dashboard:

metric from OCP dashboard

```
sum(namespace_cpu:kube_pod_container_resource_requests:sum{cluster=""}) / sum(kube_node_status_allocatable{job="
kube-state-metrics",resource="cpu",cluster=""})
```

While the custom query, shows a more conservative value of 59%

custom promql

```
sum((kube_pod_container_resource_requests{resource="cpu"} * on (pod,namespace) group_left (phase)
kube_pod_status_phase{phase="Running"}) * on (node) group_left (role) kube_node_role{role="app"} )/
sum(kube_node_status_allocatable{resource="cpu"} * on (node) group_left(role) kube_node_role{role="app"})
```

Results: