Troubleshoot High CPU/Memory Utilisation on Kubernetes Pods

Contents

Introduction

This document describes how to troubleshoot CPU or Memory issues on the Cloud Native Deployment Platform (CNDP) platform that is utilized as Session Management Function (SMF) or Policy Control Function (PCF).

1. Problem Alert High CPU/Memory on Pod

Understanding the alert is important to have a good start on troubleshooting this problem. An explanation of all default alerts that are pre-configured is at this link.

1.1. Alert for CPU

Here, there is an active default alert that is triggered named k8s-pod-cpu-usage-high .

You see that is related to a pod named: smf-udp-proxy-0 and it is a container: k8s_smf-udp-proxy_smf-udp-proxy-0_smf

You see that this container is in namespace: smf

```
alerts active detail k8s-pod-cpu-usage-high 36fbd5e0bbce
severity major
type "Processing Error Alarm"
startsAt 2024-02-23T12:45:44.558Z
source smf-udp-proxy-0
summary "Container: k8s_smf-udp-proxy_smf-udp-proxy-0_smf of pod: smf-udp-proxy-0 in namespace: smf has
labels [ "name: k8s_smf-udp-proxy_smf-udp-proxy-0_smf" "namespace: smf" "pod: smf-udp-proxy-0" ]
```

On Kubernetes master, find the impacted pod by entering this command:

```
master $ kubectl get pods smf-udp-proxy-0 -n smf
```

1.2. Alert for Memory

Here, there is an active default alert that is triggered named container-memory-usage-high .

You can see that is related to a pod named: grafana-dashboard-sgw-765664b864-zwxct and it is a container: k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwxct_smf_389290ee-77d1-4ff3-981d-58ea1c8eabdb_0

This container is in namespace:smf

```
alerts active detail container-memory-usage-high 9065cb8256ba
severity critical
type "Processing Error Alarm"
startsAt 2024-04-25T10:17:38.196Z
source grafana-dashboard-sgw-765664b864-zwxct
summary "Pod grafana-dashboard-sgw-765664b864-zwxct/k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwx
labels [ "alertname: container-memory-usage-high" "beta_kubernetes_io_arch: amd64" "beta_kubernetes_io_c
annotations [ "summary: Pod grafana-dashboard-sgw-765664b864-zwxct/k8s_istio-proxy_grafana-dashboard-sgw
```

On Kubernetes master, find the impacted pod by entering this command:

```
master $ kubectl get pods grafana-dashboard-sgw-765664b864-zwxct -n smf
```

2. Kubernetes Per-Process Profiling

2.1. CPU Profiling (/debug/pprof/profile)

CPU profiling serves as a technique for capturing and analyzing the CPU usage of a running Go program.

It samples the call stack periodically and records the information, allowing you to analyze where the program spends most of its time.

2.2. Memory Profiling (/debug/pprof/heap)

Memory profiling provides insights into memory allocation and usage patterns in your Go application.
It can help you identify memory leaks and optimize memory utilization.

2.3. Goroutine Profiling (/debug/pprof/goroutine)

Goroutine profiling provides insights into the behavior of all current Goroutines by displaying their stack traces. This analysis helps in identifying stuck or leaking Goroutines that can impact the performance of the program.

2.4. Find pprof Port on a Kubernetes Pod

Command:

```
master:~$ kubectl describe pod <POD NAME> -n <NAMESPACE> | grep -i pprof
```

Example output:

```
master:~$ kubectl describe pod udp-proxy-0 -n smf-rcdn | grep -i pprof
PPROF_EP_PORT: 8851
master:~$
```

3. Data to Collect from the System

During the time of the issue and active alert on Common Execution Environment (CEE), please collect the data that covers time before and during/after the issue:

CEE:

```
cee# show alerts active detail
cee# show alerts history detail
cee# tac-debug-pkg create from yyyy-mm-dd_hh:mm:ss to yyyy-mm-dd_hh:mm:ss
```

CNDP master node:

```
General information:
master-1:~$ kubectl get pods <POD> -n <NAMESPACE>
master-1:~$ kubectl pods describe <POD> -n <NAMESPACE>
master-1:~$ kubectl logs <POD> -n <NAMESPACE> -c <CONTAINER>

Login to impacted pod and check top tool:
master-1:~$ kubectl exec -it <POD> -n <NAMESPACE> bash
root@protocol-n0-0:/opt/workspace# top

If pprof socket is enabeled on pod:
master-1:~$ kubectl describe pod <POD NAME> -n <NAMESPACE> | grep -i pprof
master-1:~$ curl http://<POD IP>:<PPROF PORT>/debug/pprof/goroutine?debug=1
master-1:~$ curl http://<POD IP>:<PPROF PORT>/debug/pprof/heap
master-1:~$ curl http://<POD IP>:<PPROF PORT>/debug/pprof/profile?seconds=30
```

4. Understanding Collected pprof Log Outputs

4.1. Reading Output from Memory Profiling (/debug/pprof/heap)

This line indicates that a total of 1549 goroutines were captured in the profile. The top frame (0x9207a9) shows that the function google.golang.org/grpc.(*addrConn).resetTransport is being executed, and the line number in the source code is clientconn.go:1164 .

Each section starting with a number (for example, 200) represents a stack trace of a Goroutine.

```
goroutine profile: total 1549
200 @ 0x4416c0 0x415d68 0x415d3e 0x415a2b 0x9207aa 0x46f5e1
#    0x9207a9    google.golang.org/grpc.(*addrConn).resetTransport+0x6e9    /opt/workspace/gtpc-ep/pkg/m
```

The first line in each section shows the number of goroutines with the same stack trace. For example, there are 200 goroutines with the same stack trace represented by memory addresses (0x4416c0 , 0x415d68, and more.). The lines that start with # represent the individual frames of the stack trace. Each frame shows the memory address, function name, and the source code location (file path and line number) where the function is defined.

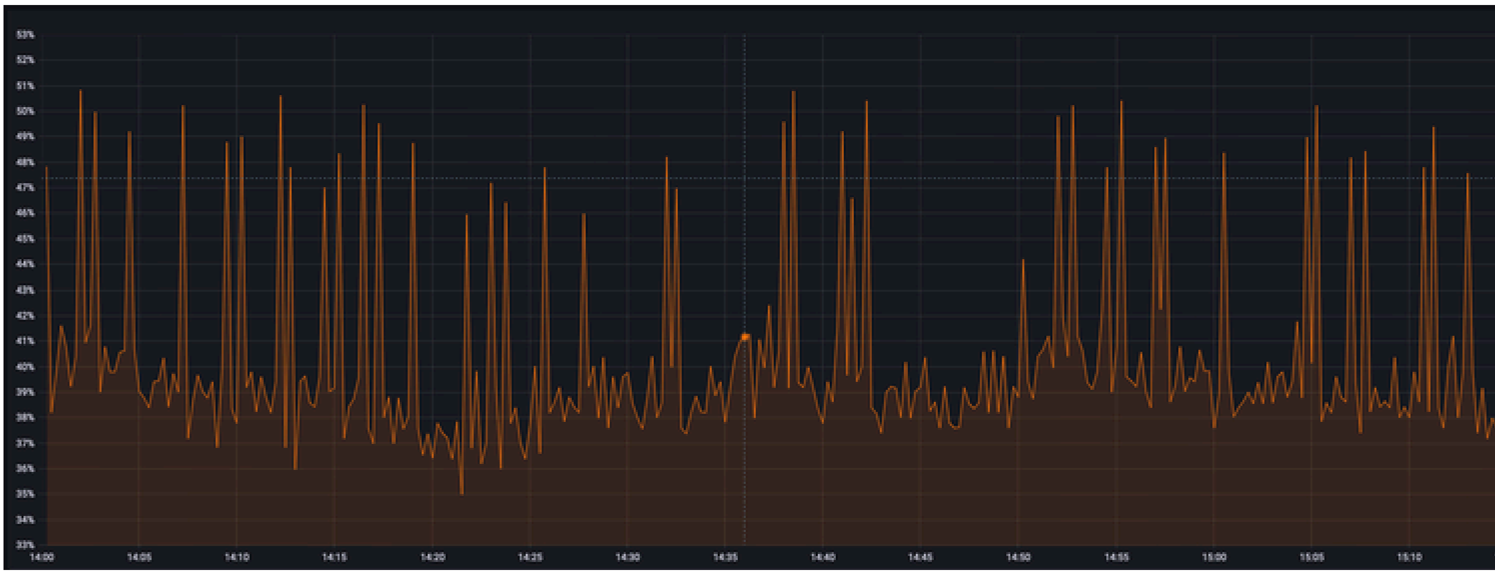
```
200 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x89674b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace/
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/gt
#    0x89674a    google.golang.org/grpc/internal/transport.newHTTP2Client.func3+0x7a    /opt/workspace/g

92 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x897b2b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace/
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/gt
#    0x897b2a    google.golang.org/grpc/internal/transport.newHTTP2Server.func2+0xca    /opt/workspace/g
```

5. Grafana

5.1. CPU Query

```
sum(cpu_percent{service_name=~"[[microservice]]"}) by (service_name,instance_id)
```

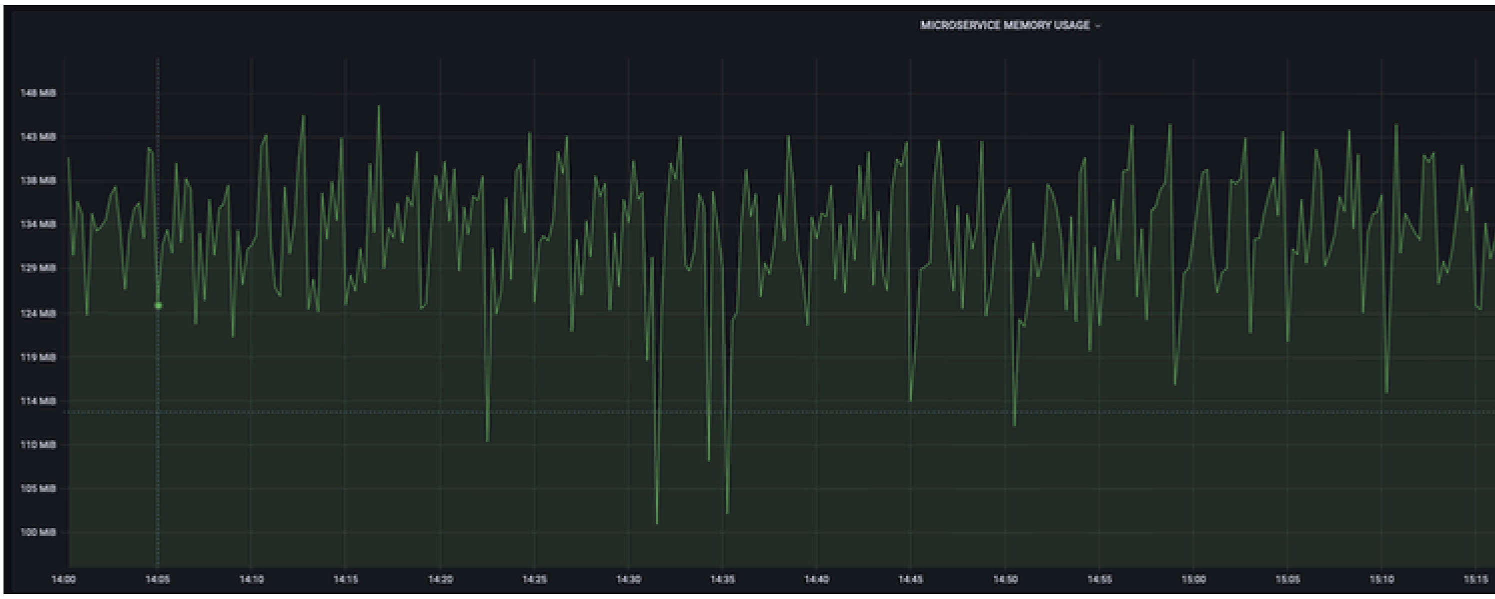
Example:



5.2. Memory Query

```
sum(increase(mem_usage_kb{service_name=~"[[microservice]]"}[15m])) by (service_name,instance_id)
```

Example:


MICROSERVICE MEMORY USAGE
148 MiB
143 MiB
138 MiB
134 MiB
129 MiB
124 MiB
119 MiB
114 MiB
110 MiB
105 MiB
100 MiB
14:00
14:05
14:10
14:15
14:20
14:25
14:30
14:35
14:40
14:45
14:50
14:55
15:00
15:05
15:10
15:15