Probleemoplossing voor gebruik van hoge CPU/geheugen op Kubernetes Pods

Inhoud

Inleiding

In dit document wordt beschreven hoe u problemen met CPU- of geheugenbeheer kunt oplossen op het CNDP-platform (Cloud Native Implementation Platform) dat wordt gebruikt als Session Management Functie (SMF) of Policy Control Functie (PCF).

1. Probleemmelding bij hoge CPU/geheugen op pod

Het begrijpen van de waarschuwing is belangrijk om een goede start te hebben bij het oplossen van dit probleem. Op deze link vindt u een uitleg van alle standaardwaarschuwingen die vooraf zijn ingesteld.

1.1. Waarschuwing voor CPU

Hier is er een actief standaard alarm dat wordt geactiveerd genaamd k8s-pod-cpu-usage-high .

Je ziet dat dit gerelateerd is aan een peul met de naam: smf-udp-proxy-0 en het is een container: k8s_smf-udp-proxy_smf-udp-proxy-0_smf

U ziet dat deze container in naamruimte is: 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" ]
```

Op Kubernetes master, vind de getroffen peul door dit commando in te voeren:

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

1.2. Waarschuwing met het oog op geheugen

Hier is er een actief standaard alarm dat wordt geactiveerd genaamd container-memory-usage-high .

Je kunt zien dat dit gerelateerd is aan een peul met de naam: grafana-dashboard-sgw-765664b864-zwxct en het is een container: k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwxct_smf_389290ee-77d1-4ff3-981d-58ea1c8eabdb_0

Deze container bevindt zich in een naamruimte: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_o
annotations [ "summary: Pod grafana-dashboard-sgw-765664b864-zwxct/k8s_istio-proxy_grafana-dashboard-sgw
```

Op Kubernetes master, vind de getroffen peul door dit commando in te voeren:

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

2. Profilering per proces van Kubernetes

2.1. CPU-profilering (debug/debug/prof/profiel)

CPU profiling dient als een techniek voor het vastleggen en analyseren van het CPU-gebruik van een actief Go-programma.

Het steekproeven de vraagstapel periodiek en registreert de informatie, die u toestaan om te analyseren waar het programma het grootste deel van zijn tijd doorbrengt.

2.2. Geheugenprofilering (debug/debug/prof/head)

Geheugenprofilering biedt inzicht in geheugentoewijzing en gebruikspatronen in uw Go-toepassing.
Het kan u helpen geheugenlekken te identificeren en geheugengebruik te optimaliseren.

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

Goroutine profiling geeft inzicht in het gedrag van alle huidige Goroutines door hun stack sporen weer te

geven. Deze analyse helpt bij het identificeren van geplakte of lekkende Goroutines die de prestaties van het programma kunnen beïnvloeden.

2.4. Zoek de poort op een Kubernetes Pod

Opdracht:

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

Voorbeelduitvoer:

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

3. Te verzamelen gegevens uit het systeem

Gedurende de tijd van het probleem en de actieve waarschuwing over gemeenschappelijke uitvoeringsomgeving (Common Execution Environment, CEE), verzamel de gegevens die de tijd voor en tijdens/na het probleem bestrijken:

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-hoofdknooppunt:

```
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. Inzicht in de verzamelde prprof-loguitgangen

4.1. Uitloop van geheugenprofilering (debug/debug/profs/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 .

Elke sectie die met een aantal (bijvoorbeeld, 200) begint vertegenwoordigt een stapelspoor van een 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.

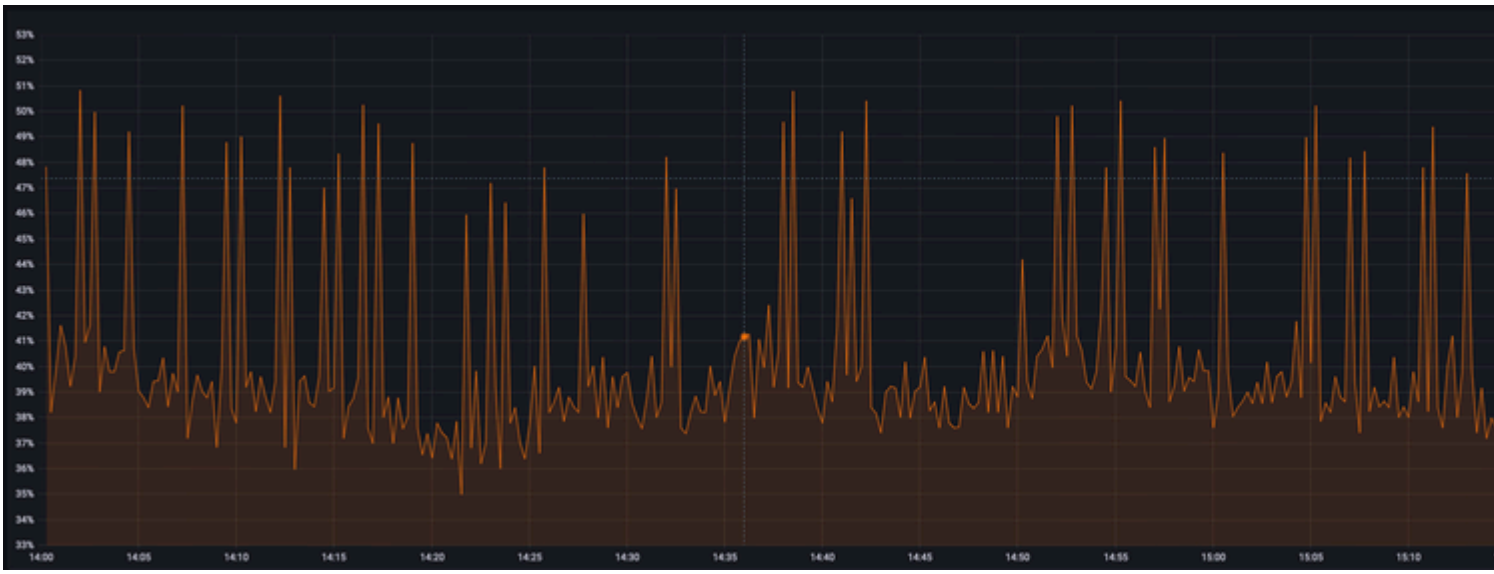
```
200 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x89674b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace/
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/gt
#    0x89674a    google.golang.org/grpc/internal/transport.newHTTP2Client.func3+0x7a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

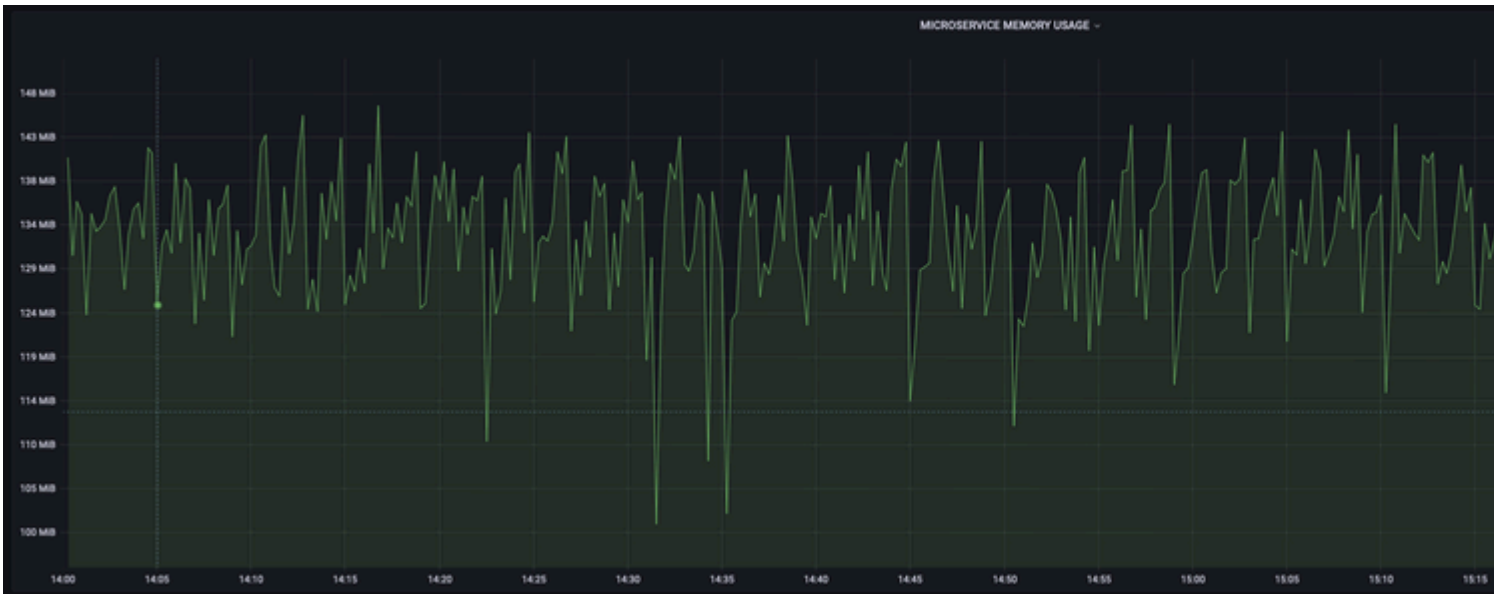
Voorbeeld:



5.2. Memory Query

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

Voorbeeld:



Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.