對Kubernetes Pods上的CPU/記憶體使用率過高進行故障排除

目錄

簡介

本文描述如何對用作會話管理功能(SMF)或策略控制功能(PCF)的雲本地部署平台(CNDP)平台上的CPU或記憶體問題進行故障排除。

1. Pod上的問題警報高CPU/記憶體

瞭解警報對於順利開始解決此問題非常重要。所有預配置的預設警報的說明位於此連結。

1.1.CPU警報

這裡有一個活動的預設警報，其觸發名稱為 k8s-pod-cpu-usage-high .

您會看到它與名為： smf-udp-proxy-0 它是個容器： k8s_smf-udp-proxy_smf-udp-proxy-0_smf

您看到此容器位於名稱空間中： 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" ]
```

在Kubernetes主裝置上，輸入以下命令查詢受影響的pod:

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

1.2.記憶體警報

這裡有一個活動的預設警報，其觸發名稱為 container-memory-usage-high .

您可以看到它與Pod有關，名稱為： grafana-dashboard-sgw-765664b864-zwxct 它是個容器： k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwxct_smf_389290ee-77d1-4ff3-981d-58ea1c8eabdb_0

此容器位於名稱空間中：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-zw
labels [ "alertname: container-memory-usage-high" "beta_kubernetes_io_arch: amd64" "beta_kubernetes_io_
annotations [ "summary: Pod grafana-dashboard-sgw-765664b864-zwxct/k8s_istio-proxy_grafana-dashboard-sg
```

在Kubernetes主裝置上，輸入以下命令查詢受影響的pod:

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

2. Kubernetes Per-Process分析

2.1. CPU分析(/debug/pprof/profile)

CPU分析用作捕獲和分析正在運行的Go程式的CPU使用率的技術。

它定期對呼叫堆疊進行取樣並記錄資訊，從而使您能夠分析程式大部分時間都花在什麼地方。

2.2.記憶體分析(/debug/pprof/heap)

記憶體分析可深入瞭解Go應用程式中的記憶體分配和使用模式。
它可以幫助您識別記憶體洩漏和最佳化記憶體利用率。

2.3.戈羅廷輪廓分析(/debug/pprof/goroutine)

Goroutine分析通過顯示所有當前的Goroutines的堆疊跟蹤來深入瞭解其行為。此分析有助於識別可能影響程式效能的卡住或滲漏的Goroutines。

2.4.在Kubernetes Pod上找到pprof Port

指令:

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

輸出示例：

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

3.從系統中收集的資料

在出現問題和通用執行環境(CEE)活動警報期間，請收集問題前和問題中/問題後的時間資料：

中東歐：

```
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主節點：

```
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.瞭解收集的pprof日誌輸出

4.1.從記憶體分析讀取輸出(/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 .

每個以數字開頭的部分（例如200）代表一個古羅津的堆疊跟蹤。

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

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.

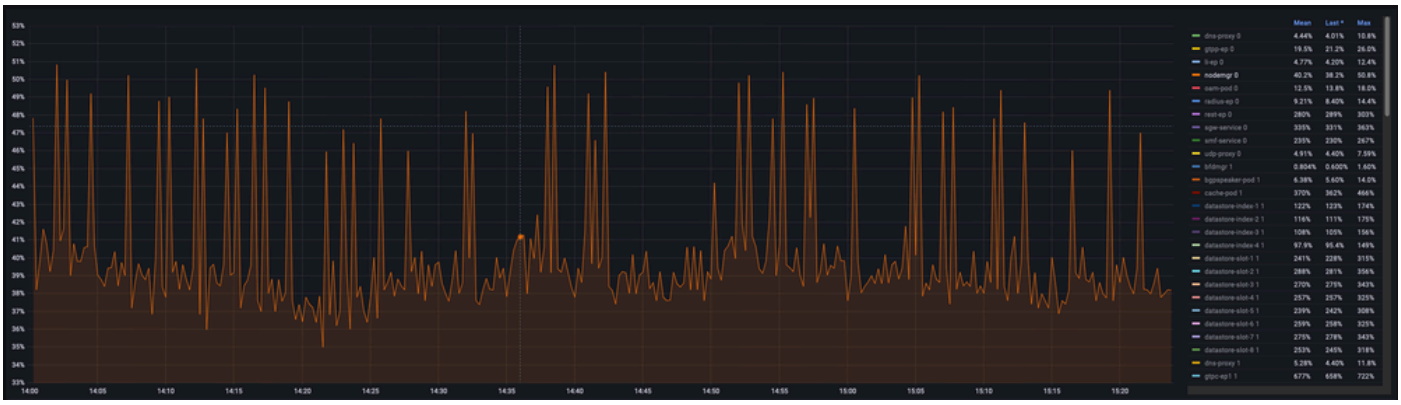
```
200 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x89674b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace/
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/g
#    0x89674a    google.golang.org/grpc/internal/transport.newHTTP2Client.func3+0x7a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

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/g
#    0x897b2a    google.golang.org/grpc/internal/transport.newHTTP2Server.func2+0xca    /opt/workspace/
```

5.格拉法納

5.1. CPU查詢

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

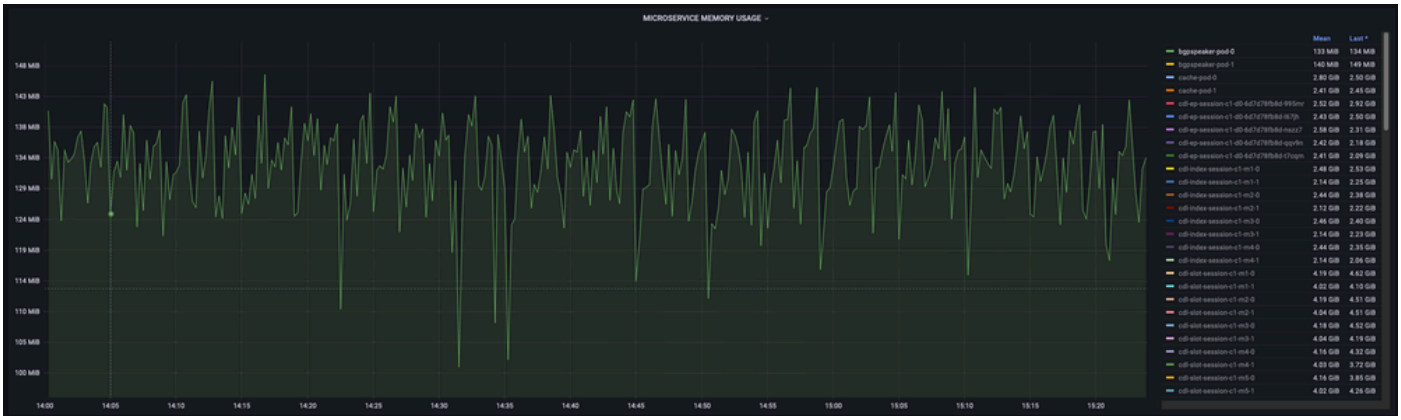
範例：



5.2.記憶體查詢

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

範例：



關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。