KubernetesポッドのCPU/メモリ使用率が高い場合のトラブルシューティング

内容

[概要](#)
[1.ポッドのCPU/メモリの高使用率に関する問題アラート](#)
[1.1. CPUに関するアラート](#)
[1.2.メモリに関するアラート](#)
[2. Kubernetesのプロセス単位のプロファイリング](#)
[2.1. CPUプロファイリング(/debug/pprof/profile)](#)
[2.2.メモリプロファイリング(/debug/pprof/heap)](#)
[2.3. Goroutineプロファイリング(/debug/pprof/goroutine)](#)
[2.4. Kubernetesポッドでのpprofポートの検索](#)
[3.システムから収集すべきデータ](#)
[4.収集されたpprof口グ出力の理解](#)
[4.1.メモリプロファイリング(/debug/pprof/heap)からの出力の読み取り](#)
[五グラファナ](#)
[5.1. CPUクエリ](#)
[5.2.メモリクエリ](#)

概要

このドキュメントでは、セッション管理機能(SMF)またはポリシー制御機能(PCF)として使用されるCloud Native Deployment Platform(CNDP)プラットフォームのCPUまたはメモリの問題をトラブルシューティングする方法について説明します。

1.ポッドのCPU/メモリの高使用率に関する問題アラート

この問題のトラブルシューティングを適切に開始するには、アラートを理解することが重要です。事前に設定されているすべてのデフォルトアラートの説明は、[このリンク](#)にあ[ります。](#)

1.1. CPUに関するアラート

ここでは、アクティブなデフォルトのアラートがトリガーされて、 k8s-pod-cpu-usage-high を参照。

これは、次の名前のポッドに関連しています。 smf-udp-proxy-0 コンテナです。 k8s_smf-udp-proxy_smf-udp-proxy-0_smf

次のコンテナが名前空間にあることがわかります。 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" ]
```

Kubernetes masterで、次のコマンドを入力して影響を受けるポッドを見つけます。

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

1.2.メモリに関するアラート

ここでは、アクティブなデフォルトのアラートがトリガーされて、 container-memory-usage-high を参照。

これは、次の名前のポッドに関連しています。 grafana-dashboard-sgw-765664b864-zwxct コンテナです。
k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwxct_smf_389290ee-77d1-4ff3-981d-58ea1c8eabdb_0

このコンテナーは名前空間にあります：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
```

Kubernetes masterで、次のコマンドを入力して影響を受けるポッドを見つけます。

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

2. Kubernetesのプロセス単位のプロファイリング

2.1. CPUプロファイリング(/debug/pprof/profile)

CPUプロファイリングは、実行中のGoプログラムのCPU使用率をキャプチャして分析するための手法です。

コールスタックを定期的にサンプリングして情報を記録するため、プログラムの時間の大半を費やす場所を分析できます。

2.2.メモリプロファイリング(/debug/pprof/heap)

メモリプロファイリングは、Goアプリケーションのメモリ割り当てと使用パターンに関する洞察を提供します。
メモリリークの特定とメモリ使用率の最適化に役立ちます。

2.3. Goroutineプロファイリング(/debug/pprof/goroutine)

Goroutineプロファイリングは、スタックトレースを表示することで、現在のすべてのGoroutineの動作に関する洞察を提供します。この分析は、プログラムのパフォーマンスに影響を与える可能性がある、スタックまたはリークしているGoroutineを特定するのに役立ちます。

2.4. Kubernetesポッドでのpprofポートの検索

コマンド：

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

出力例：

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

3.システムから収集すべきデータ

問題発生時およびCommon Execution Environment(CEE)に対するアクティブなアラートの間に、問題発生前と発生中/発生後の時間を含むデータを収集してください。

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マスターノード：

```
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.収集されたpprof口グ出力の理解

4.1.メモリプロファイリング(/debug/pprof/heap)からの出力の読み取り

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 .

200などの数値で始まる各セクションは、Goroutineのスタックトレースを表します。

```
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.

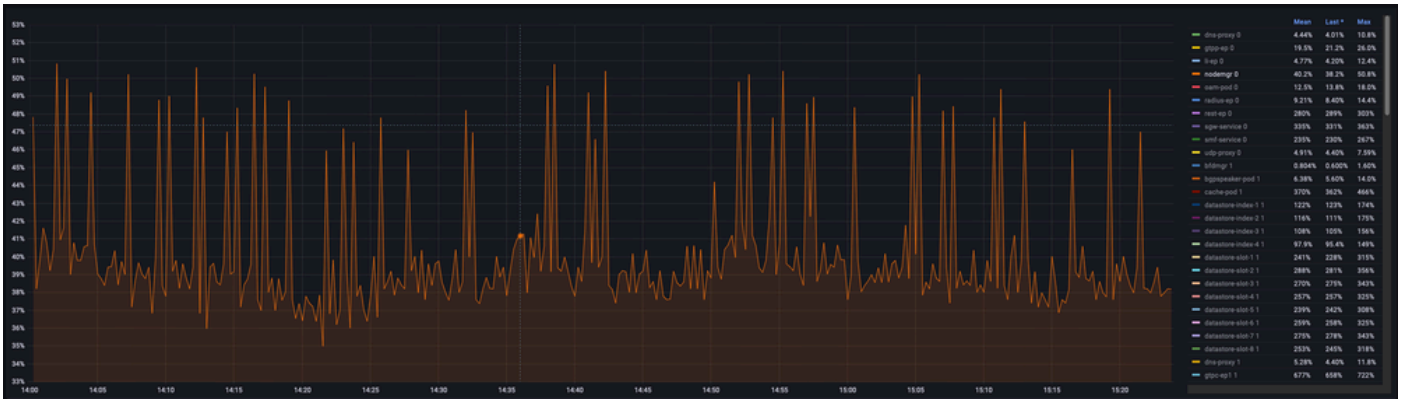
```
200 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x89674b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/g
#    0x89674a    google.golang.org/grpc/internal/transport.newHTTP2Client.func3+0x7a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

五グラファナ

5.1. CPUクエリ

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

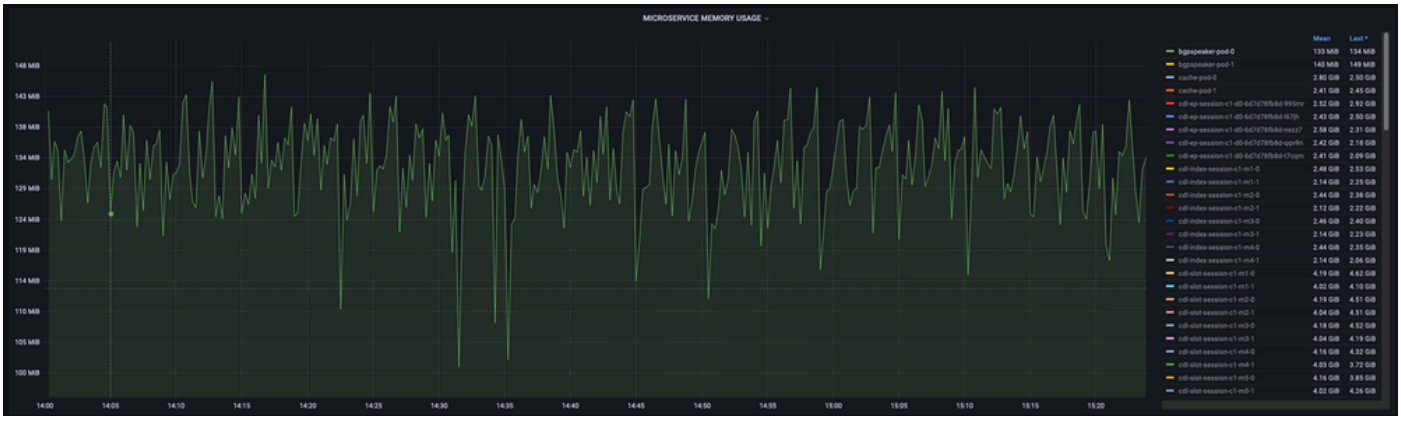
以下に例を挙げます。



5.2.メモリクエリ

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

以下に例を挙げます。



翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。