对Kubernetes Pod上的CPU/内存使用率过高进行故障排除

目录

简介

本文档介绍如何对用作会话管理功能(SMF)或策略控制功能(PCF)的云本地部署平台(CNDP)平台上的CPU或内存问题进行故障排除。

1. Pod上的问题警报高CPU/内存

了解警报对于顺利开始解决此问题非常重要。所有预配置的默认警报的说明位于此链接。

1.1.CPU警报

此处，有一个活动的默认警报被触发，其名称为 k8s-pod-cpu-usage-high .

您会看到它与名为： smf-udp-proxy-0 它是个容器： k8s_smf-udp-proxy_smf-udp-proxy-0_smf

您看到此容器位于命名空间中： 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主设备上，输入以下命令查找受影响的Pod:

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

1.2.内存警报

此处，有一个活动的默认警报被触发，其名称为 container-memory-usage-high .

您可以看到它与名为： grafana-dashboard-sgw-765664b864-zwxct 它是个容器： k8s_istio-proxy_grafana-dashboard-sgw-765664b864-zwxct_smf_389290ee-77d1-4ff3-981d-58ea1c8eabdb_0

此容器位于命名空间中：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主设备上，输入以下命令查找受影响的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分析用作捕获和分析正在运行的Go程序的CPU使用率的技术。

它定期对调用堆栈进行采样并记录信息，使您可以分析程序将大部分时间用于何处。

2.2.内存分析(/debug/pprof/heap)

内存分析可让您深入了解Go应用中的内存分配和使用模式。
它可以帮助您识别内存泄漏和优化内存利用率。

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

Goroutine分析通过显示所有当前的Goroutines的栈跟踪提供对其行为的洞察。此分析有助于识别可能会影响程序性能的卡住或渗漏的Goroutines。

2.4.在Kubernetes Pod上找到pprof Port

命令:

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

示例输出：

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

3.从系统收集数据

在出现问题和通用执行环境(CEE)活动警报期间，请收集问题前和问题中/问题后的时间数据：

中东欧：

```
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主节点：

```
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.了解收集的pprof日志输出

4.1.读取内存分析的输出(/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 .

以数字开头的每个部分（例如200）代表一个金鱼碱的栈跟踪。

```
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.

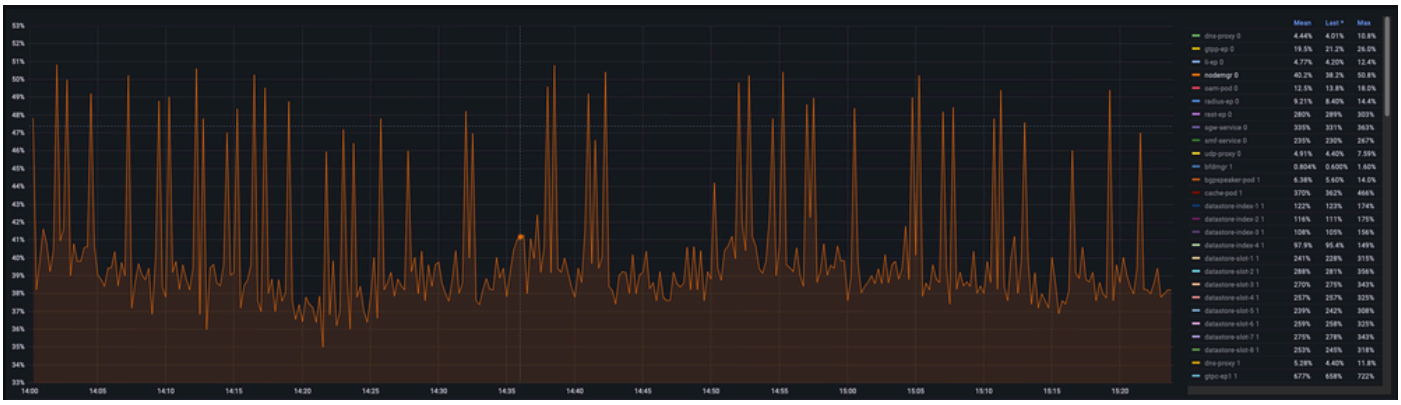
```
200 @ 0x4416c0 0x45121b 0x873ee2 0x874803 0x89674b 0x46f5e1
#    0x873ee1    google.golang.org/grpc/internal/transport.(*controlBuffer).get+0x121    /opt/workspace/
#    0x874802    google.golang.org/grpc/internal/transport.(*loopyWriter).run+0x1e2    /opt/workspace/g
#    0x89674a    google.golang.org/grpc/internal/transport.newHTTP2Client.func3+0x7a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

5.1. CPU查询

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

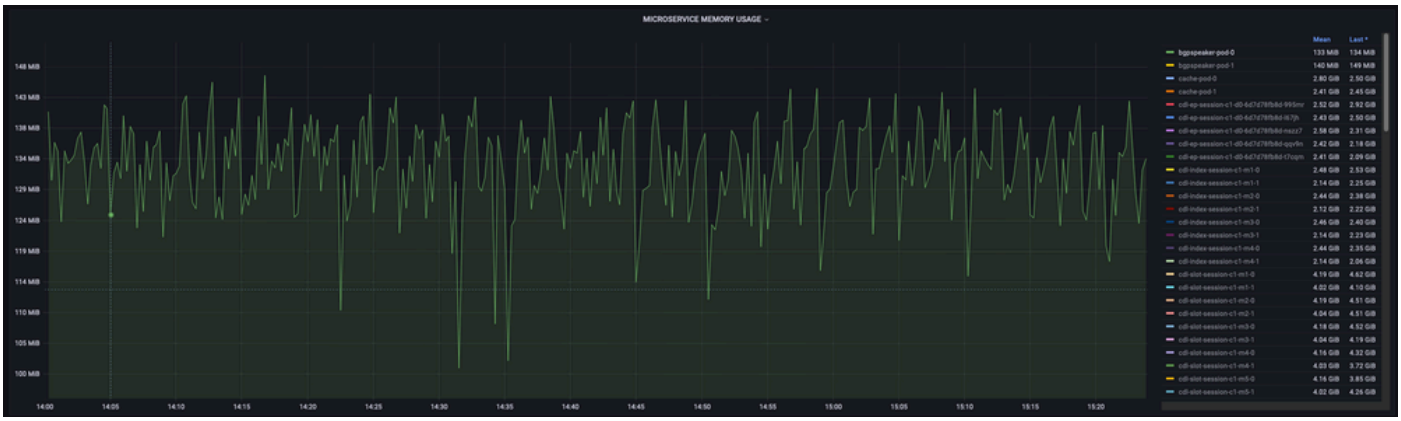
示例：



5.2.内存查询

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

示例：



关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。