Fehlerbehebung in einer langsamen APIC-GUI

Inhalt

Einleitung

In diesem Dokument wird die allgemeine Methode zur Fehlerbehebung bei einer langsamen grafischen Benutzeroberfläche des APIC beschrieben.

Schnellstart

Es wird häufig festgestellt, dass langsame APIC-GUI-Probleme das Ergebnis einer hohen Rate von API-Anfragen sind, die von einem Skript, einer Integration oder einer Anwendung stammen. Die Datei access.log eines APIC protokolliert jede verarbeitete API-Anforderung. Die Datei access.log eines APIC kann mit dem Skript Access Log Analyzer innerhalb des aci-tac-scripts-Projekts der Github Datacenter-Gruppe schnell analysiert werden.

Hintergrundinformationen

APIC als Webserver - NGINX

NGINX ist die DME, die für die auf den einzelnen APICs verfügbaren API-Endpunkte verantwortlich ist. Wenn NGINX ausgefallen ist, können API-Anfragen nicht verarbeitet werden. Bei Überlastung von NGINX ist die API überlastet. Jeder APIC führt seinen eigenen NGINX-Prozess aus. Daher kann es vorkommen, dass nur ein APIC NGINX-Probleme hat, wenn nur dieser APIC Ziel aggressiver Queries ist.

Die APIC-Benutzeroberfläche führt mehrere API-Anforderungen aus, um jede Seite auszufüllen. In

ähnlicher Weise sind alle APIC-Befehle zum Anzeigen (CLI im NXOS-Stil) Wrapper für Python-Skripts, die mehrere API-Anforderungen ausführen, die Antwort verarbeiten und dann an den Benutzer weiterleiten.

Relevante Protokolle

Protokolldateiname	Location (Standort)	In welchem Technologiesupport befinden sich die Lösungen?	Kommentare
access.log	/var/log/dme/log	APIC 3 von 3	ACI-unabhängig, 1 Leitung pro API-Anfrage
error.log	/var/log/dme/log	APIC 3 von 3	ACI-unabhängig, zeigt nginx-Fehler an (einschl. Drosselung)
nginx.bin.log	/var/log/dme/log	APIC 3 von 3	ACI-spezifisch, protokolliert DME-Transaktionen
nginx.bin.warnplus.log	/var/log/dme/log	APIC 3 von 3	ACI-spezifisch enthält Protokolle mit Warn- und Schweregrad

Methodik

Isolierung des anfänglichen Triggers

Was ist davon betroffen?

- Welche APICs sind betroffen, einer, viele oder alle APICs?
- Wo ist Langsamkeit zu beobachten? Über die Benutzeroberfläche, CLI-Befehle oder beides?
- Welche Benutzeroberflächenseiten oder -befehle sind langsam?

Wie wird die Langsamkeit erlebt?

- Wird dies bei mehreren Browsern für einen einzelnen Benutzer beobachtet?
- Melden mehrere Benutzer Langsamkeit oder nur eine einzelne/einen Teil der Benutzer?
- Nutzen die betroffenen Benutzer einen ähnlichen geografischen Standort oder Netzwerkpfad vom Browser zum APIC?

Wann wurde die Langsamkeit zum ersten Mal bemerkt?

- Wurde kürzlich eine ACI-Integration oder ein ACI-Skript hinzugefügt?
- Wurde kürzlich eine Browsererweiterung aktiviert?
- Gab es kürzlich eine Änderung an der ACI-Konfiguration?

NGINX-Nutzung und -Status überprüfen

Format des Access.log-Eintrags

access.log ist eine Funktion von NGINX und daher APIC-unabhängig. Jede Zeile steht für eine HTTP-Anforderung, die der APIC empfangen hat. Lesen Sie dieses Protokoll, um die NGINX-Verwendung eines APIC zu verstehen.

Das Standardformat "access.log" für die ACI Version 5.2+:

```
log_format proxy_ip '$remote_addr ($http_x_real_ip) - $remote_user [$time_local]'
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

Diese Zeile stellt einen access.log-Eintrag dar, wenn ein moquery -c fvTenant ausgeführt wird:

```
127.0.0.1 (-) - - [07/Apr/2022:20:10:59 +0000]"GET /api/class/fvTenant.xml HTTP/1.1" 200 15863 "-" "Pyth
```

Zuordnung eines Beispieleintrags access.log zu log_format:

Logformat-Feld	**Inhalt aus Beispiel**	**Kommentare**
$remote_addr	127.0.0.1	IP des Hosts, der diese Anforderung gesendet hat
$http_x_real_ip	-	IP des letzten Anforderers, wenn Proxys verwendet werden
$remote_user	-	Nicht in der Regel verwendet. Überprüfen Sie nginx.bin.log, um festzustellen, welcher Benutzer sich angemeldet hat, um Anforderungen auszuführen.
$time_local	07.04.2022:20:10:59 +0000	Wann wurde die Anforderung verarbeitet?
Anfrage in USD	/api/class/fvTenant.xml HTTP/1.1 herunterladen	HTTP-Methode (GET, POST, DELETE) und URI
$ Status	200	HTTP-Antwortstatuscode
$body_bytes_sent	1586	Größe der Antwortnutzlast

$http_referer	-	-
$http_user_agent	Python-Urllib	Welcher Clienttyp hat die Anforderung gesendet?

Verhalten von Access.log

Anforderungen mit hoher Rate treten über einen langen Zeitraum hinweg auf:

- Kontinuierliche Spitzen von mehr als 15 Anfragen pro Sekunde können eine verlangsamte Benutzeroberfläche verursachen.
- Identifizieren Sie die Hosts, die für die Anfragen verantwortlich sind.
- Reduzieren oder deaktivieren Sie die Quellenangabe, um festzustellen, ob sich dadurch die Reaktionszeit des APIC verbessert.

Konsistente 4xx- oder 5xx-Antworten:

- Wenn gefunden, geben Sie die Fehlermeldung aus nginx.bin.log ein.

NGINX-Ressourcennutzung überprüfen

Die NGINX-CPU- und Speicherauslastung kann mit dem **obersten** Befehl des APIC überprüft werden:

<#root>

```
top - 13:19:47 up 29 days, 2:08, 11 users, load average: 12.24, 11.79, 12.72
Tasks: 785 total, 1 running, 383 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.5 us, 2.0 sy, 0.0 ni, 94.2 id, 0.1 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 13141363+total, 50360320 free, 31109680 used, 49943636 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 98279904 avail Mem

PID   USER PR NI VIRT    RES   SHR   S %CPU %MEM TIME+     COMMAND
21495 root 20 0  4393916 3.5g 217624 S
```

2.6

2.8 759:05.78

nginx.bin

Eine hohe NGINX-Ressourcennutzung kann direkt mit einer hohen Rate verarbeiteter Anfragen in Zusammenhang stehen.

Auf Kerne prüfen

Ein NGINX-Absturz ist nicht typisch für Probleme mit der langsameren APIC-GUI. Wurden NGINX-Kerne gefunden, fügen Sie sie zur Analyse einem TAC-Serviceticket hinzu. Im ACI-Technischen Support-Handbuch finden Sie die Schritte zur Suche nach Kernen.

Client-Server-Latenz überprüfen

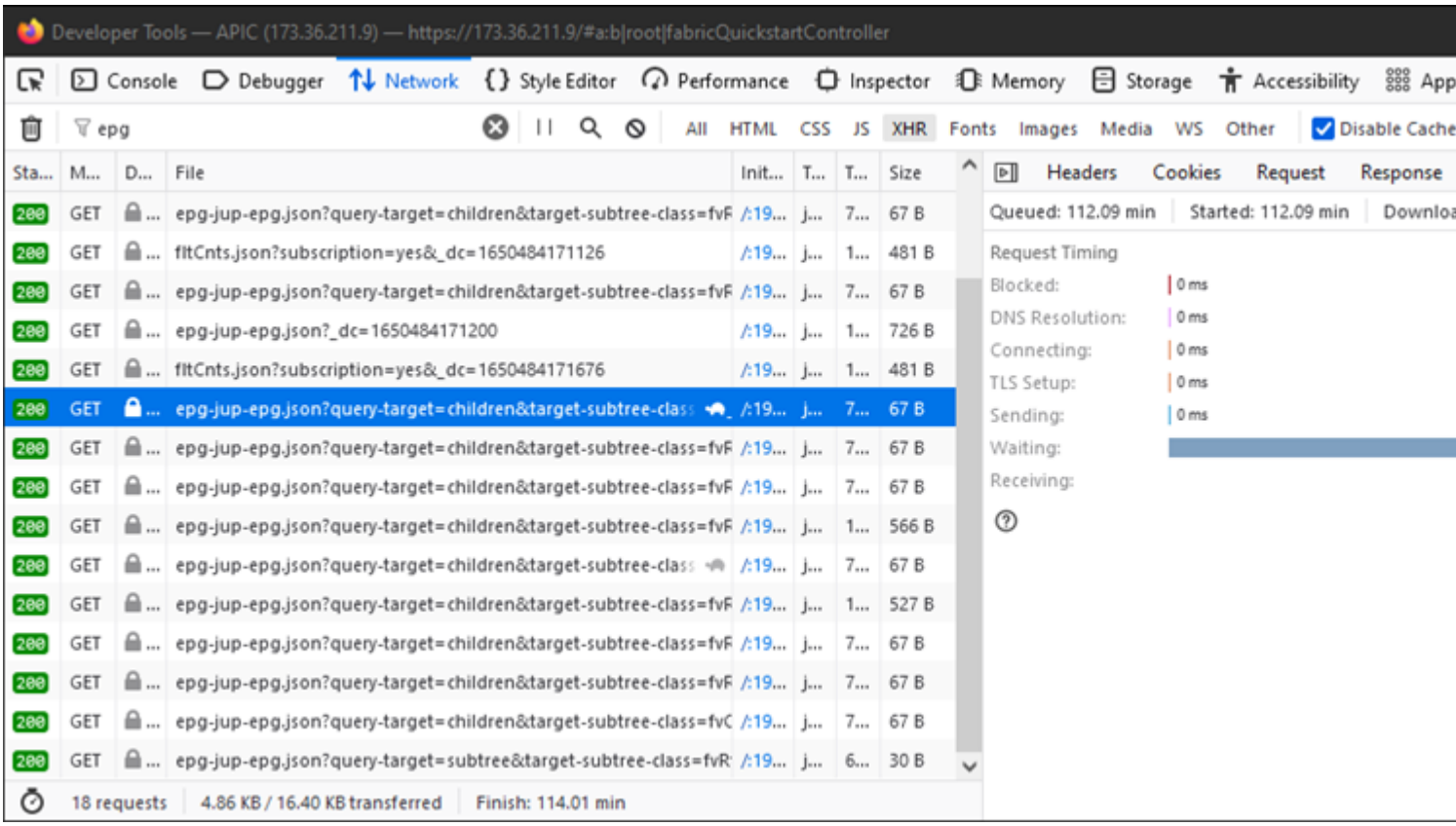
Wenn keine schnellen Anfragen gefunden werden, ein Benutzer jedoch weiterhin eine langsame Benutzeroberfläche aufweist, kann es zu einer Latenz zwischen Client (Browser) und Server (APIC) kommen.

Validieren Sie in diesen Szenarien den Datenpfad vom Browser zum APIC (geografische Entfernung, VPN usw.). Wenn möglich, Bereitstellung und Test des Zugriffs von einem Jump Server in derselben geografischen Region oder demselben Rechenzentrum wie die zu isolierenden APICs. Überprüfen Sie, ob andere Benutzer eine ähnliche Latenz aufweisen.

Browser-Entwicklungstools Registerkarte Netzwerk

Alle Browser können HTTP-Anfragen und -Antworten über ihr **Browser Development** Toolkit validieren, in der Regel über die Registerkarte **Netzwerk**.

Dieses Tool kann verwendet werden, um die Zeit zu validieren, die für jede Phase der Browser-Anfragen benötigt wird, wie im Bild gezeigt.



Beispiel für einen Browser, der 1,1 Minuten auf Antwort des APIC wartet

Verbesserungen für bestimmte Benutzeroberflächenseiten

Seite "Policy Group":

Cisco Bug-ID CSCvx14621 - Die APIC-GUI wird langsam in die IPG-Richtlinien auf der Registerkarte "Fabric" geladen.

Schnittstelle auf der Inventarseite:

Cisco Bug-ID CSCvx90048 - Initial Load von "Layer 1 Physical Interface Configuration" (Physische Layer-1-Schnittstellenkonfiguration), Registerkarte "Operational" (Betrieb) ist lang/induziert "Freeze" (Einfrieren).

Allgemeine Empfehlungen für Client > Server-Latenz

Bestimmte Browser wie Firefox ermöglichen standardmäßig mehr Webverbindungen pro Host.

- Überprüfen Sie, ob diese Einstellung für die verwendete Browserversion konfigurierbar ist.
- Dies ist für Seiten mit mehreren Abfragen von größerer Bedeutung, z. B. für die Seite der Richtliniengruppe

VPN und die Entfernung zum APIC erhöhen die Gesamtlangsamheit der Benutzeroberfläche angesichts der Anfragen von Client-Browsern und der Reaktionszeit des APIC. Ein geografisch lokales Jump-Box für die APICs verkürzt den Browser erheblich auf die APIC-Reisezeiten.

Überprüfen auf Long-Web-Anforderungen

Wenn ein Webserver (NGINX auf dem APIC) eine große Anzahl von Long-Web-Anfragen verarbeitet, kann dies die Leistung anderer parallel empfangener Anfragen beeinträchtigen.
Dies gilt insbesondere für Systeme mit verteilten Datenbanken, wie APICs. Eine einzelne API-Anforderung

Systemreaktionszeit - Berechnung der Serverreaktionszeit aktivieren

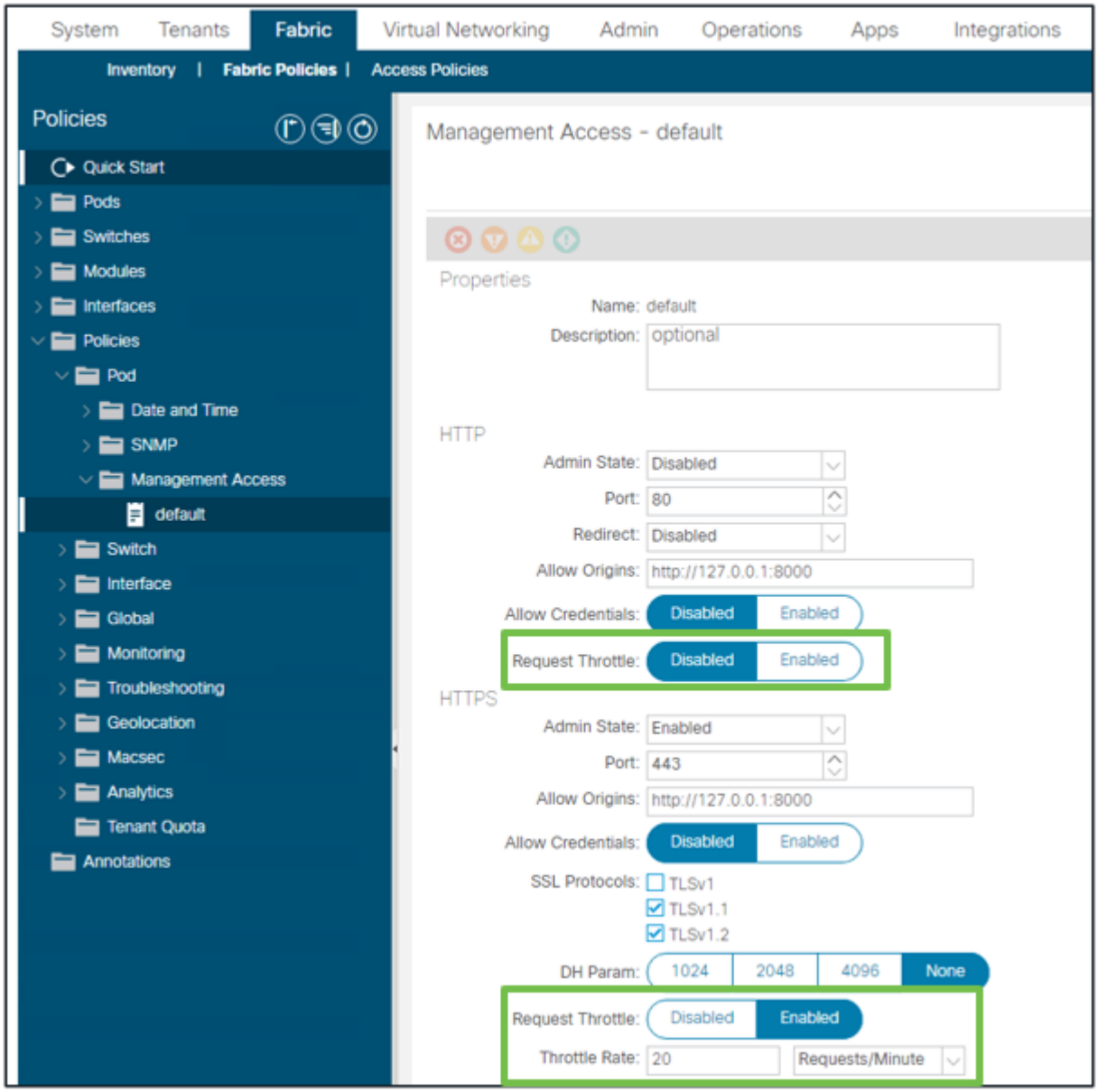
- Im Allgemeinen schränkt mehr als 15 API-Anfragen pro Sekunde über einen langen Zeitraum NGINX ein.
    - Reduzieren Sie ggf. die Aggressivität der Anfragen.
    - Wenn der Host für Anfragen nicht geändert werden kann, sollten Sie NGINX-Ratenlimits auf dem APIC in Betracht ziehen.

Ineffizienzen bei Adressskripten

- Melden Sie sich nicht vor jeder API-Anforderung an bzw. ab.
    - Das Standard-Timeout für eine Anmeldesitzung beträgt 10 Minuten. Dieselbe Sitzung kann für mehrere Anforderungen verwendet und zur Verlängerung der Gültigkeitsdauer aktualisiert werden.
    - Siehe Cisco APIC REST API Configuration Guide - Accessing the REST API - Authenticating and Maintaining an API Session.
- Wenn das Skript viele DNs abfragt, die ein übergeordnetes Objekt gemeinsam nutzen, statt die Abfragen in eine einzelne logische übergeordnete Abfrage mit Abfragefiltern zu reduzieren.
    - Siehe Cisco APIC REST API-Konfigurationsleitfaden - Erstellen von REST API-Abfragen - Anwenden von Abfragebereichsfiltern.
- Wenn Sie Aktualisierungen für ein Objekt oder eine Objektklasse benötigen, sollten Sie WebSocket-Abonnements anstelle von schnellen API-Anforderungen in Betracht ziehen.

NGINX-Anforderungsdrosselung

Ein Benutzer, der in Version 4.2(1)+ verfügbar ist, kann die Anforderungsdrosselung für HTTP und HTTPS unabhängig aktivieren.



Fabric - Fabric-Richtlinien - Richtlinienordner - Verwaltungszugriffsordner - Standard

Wenn aktiviert:

- NGINX wird neu gestartet, um Konfigurationsdateiänderungen anzuwenden.
  - Eine neue Zone, **httpsClientTagZone**, wird in die nginx-Konfiguration geschrieben.
- Die Drosselungsrate kann in **Anfragen pro Minute (**r/m) oder **Anfragen pro Sekunde (**r/s) festgelegt werden.
- Die Anfragedrosselung basiert auf der Implementierung der Ratenbegrenzung in NGINX.
  - API-Anfragen für den **/api/**URI verwenden die benutzerdefinierte Drosselungsrate + Burst= (Drosselungsrate x 2) + nodelay
    - Es gibt eine nicht konfigurierbare Drossel (Zone **aaaApiHttps**) für **/api/aaaLogin** und **/api/aaaRefresh,** die Durchsatzbegrenzungen bei 2r/s + Burst=4 + nodelay hat.
  - Request Throttle wird pro Client-IP-Adresse nachverfolgt.
  - API-Anfragen von der APIC-Self-IP (UI + CLI) umgehen die Drosselung.
  - Jede Client-IP-Adresse, die die benutzerdefinierte Drosselungsrate + den Burst-Schwellenwert überschreitet, erhält eine Antwort des APIC von 503.
  - Diese 503s können in den Zugriffsprotokollen korreliert werden.
  - error.log enthält Einträge, die angeben, wann die Drosselung aktiviert wurde (Zone **httpsClientTagZone**) und für welche Client-Hosts

<#root>

```
apic#
```

less /var/log/dme/log/error.log

```
...
2023/04/17 20:19:14 [error] ...
```

limiting requests

```
, excess: 40.292 by zone "
```

httpsClientTagZone

```
", client: h.o.s.t, ... request: "GET /api/class/...", host: "a.p.i.c"
2023/04/17 20:19:14 [error] ...
```

limiting requests

```
, excess: 40.292 by zone "
```

httpsClientTagZone

```
", client: h.o.s.t, ... request: "GET /api/node/...", host: "a.p.i.c"
```

In der Regel dient Request Throttle nur dazu, den Server (APIC) vor DDOS-ähnlichen Symptomen zu schützen, die durch abfrageaggressive Clients hervorgerufen werden. Verstehen und isolieren Sie den anforderungsaggressiven Client für endgültige Lösungen in der App-/Skriptlogik.

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.