排除速度慢的APIC GUI故障

目錄 簡介 快速入門 <u>背景資訊</u> <u>APIC作為Web伺服器 — NGINX</u> 相關日誌 方法 <u>隔離初始觸發器</u> <u>檢查NGINX使用情況和運行狀況</u> Access.log條目格式 <u>Access.log行為</u> <u>檢查NGINX資源使用情況</u> 檢查核心 檢查客戶端到伺服器的延遲 <u>瀏覽器開發工具網路頁籤</u> <u>特定UI頁面的增強功能</u> 客戶端>伺服器延遲的一般建議 <u>檢查Long-Web請求</u> 系統響應時間 — 啟用伺服器響應時間的計算 APIC API使用注意事項 <u>確保指令碼不會損害Nginx的一般指標</u> <u>解決指令碼效率低下問題</u> <u>NGINX請求限制</u>

簡介

本文檔介紹對APIC GUI體驗緩慢進行故障排除的一般方法。

快速入門

人們經常發現,APIC GUI問題緩慢是由於來自指令碼、整合或應用程式的API請求速率過高造成的 。APIC的access.log會記錄每個已處理的API請求。可以使用Github資料中心組<u>aci-tac-scripts專案</u> <u>中的訪問日誌分析器</u>指令碼<u>快速分析APIC的access.</u>log。

背景資訊

APIC作為Web伺服器 — NGINX

NGINX是負責每個APIC上可用的API終端的DME。如果NGINX關閉,則無法處理API請求。如果 NGINX擁塞,則API擁塞。每個APIC都運行自己的NGINX進程,因此如果任何主動式查詢器僅針對 APIC,則可能只有一個APIC會出現NGINX問題。

APIC UI執行多個API請求以填充每個頁面。同樣,所有APIC 'show'命令(NXOS Style CLI)都是執行 多個API請求、處理響應然後提供給使用者的python指令碼的包裝程式。

相關日誌

日誌檔名	位置	它位於哪一技術 支援中	意見
access.log	/var/log/dme/log	APIC 3of3	與ACI無關,每個API請求提供1行
error.log	/var/log/dme/log	APIC 3of3	ACI不可知,顯示nginx錯誤(包括限制)
nginx.bin.log	/var/log/dme/log	APIC 3of3	特定於ACI,記錄DME事務
nginx.bin.warnplus.log	/var/log/dme/log	APIC 3of3	ACI特定包含警告+嚴重性的日誌

方法

隔離初始觸發器

什麼受到影響?

- 哪些APIC會受到影響;一個、多個APIC還是所有APIC?
- 在哪裡可以看到慢度;通過UI、CLI命令還是同時使用兩者?
- 哪些特定UI頁面或命令速度較慢?

這種緩慢的感受如何?

- 對於單個使用者而言,是否會在多個瀏覽器中看到這種情況?
- 多個使用者是否報告速度緩慢?還是僅報告單個/子集使用者?
- 受影響的使用者是否共用從瀏覽器到APIC的相似地理位置或網路路徑?

遲緩是何時被首次注意到的?

- 最近是否新增了ACI整合或指令碼?
- 最近是否啟用了瀏覽器擴展?
- ACI配置最近有變化嗎?

檢查NGINX使用情況和運行狀況

Access.log條目格式

access.log是NGINX的一項功能,因此與APIC無關。每行代表APIC收到的1個HTTP請求。參考此 日誌,瞭解APIC的NGINX使用情況。

ACI 5.2+版上的預設access.log格式:

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

此行表示執行moquery -c fvTenant時的access.log條目:

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

示例access.log條目對映到log_format:

log_format欄位	示例內容	意見
\$remote_addr	127.0.0.1	傳送此請求的主機的IP
\$http_x_real_ip	-	使用代理時最後一個請求者的IP
\$remote_user	-	一般不使用。選中nginx.bin.log以跟蹤登入 執行請求的使用者
\$time_local	2022年4月07日:20:10:59 +0000	處理請求的時間
\$request	獲取/api/class/fvTenant.xml HTTP/1.1	Http方法(GET、POST、DELETE)和URI
\$status	200	<u>HTTP響應狀態代碼</u>
\$body_bytes_sent	1586	響應負載大小

\$http_referer	-	-
\$http_user_agent	Python-urllib	傳送請求的客戶端型別

Access.log行為

在一段較長的時間內發生高速請求突發:

- 每秒15個以上請求的連續突發會導致使用者介面速度變慢
- 確定負責查詢的主機
- 減少或禁用查詢源,檢視這是否提高了APIC響應時間。

一致的4xx或5xx響應:

• 如果找到,請識別來自nginx.bin.log的錯誤消息

檢查NGINX資源使用情況

可以使用APIC中的top命令檢查NGINX CPU和記憶體使用情況:

<#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

高NGINX資源使用率直接與高處理請求率相關。

檢查核心

NGINX崩潰不常用於慢速APIC GUI問題。但是,如果找到NGINX核心,請將其連線到TAC SR進行 分析。有關檢查核心的步驟,請參閱<u>ACI技術支援指南</u>。

檢查客戶端到伺服器的延遲

如果找不到快速請求,但使用者繼續表現出UI緩慢,則問題可能是客戶端(瀏覽器)到伺服器

(APIC)延遲。

在這些情況下,驗證從瀏覽器到APIC的資料路徑(地理距離、VPN等)。如果可能,請部署並測試 從與APIC位於同一地理區域或資料中心內的跳轉伺服器進行的訪問。驗證其他使用者是否顯示類似 的延遲量。

瀏覽器開發工具網路頁籤

所有瀏覽器都能夠通過其Browser Development Toolkit(通常在Network(網路)頁籤中)驗證 HTTP請求和響應。

此工具可用於驗證來自瀏覽器的請求的每個階段所需的時間,如下圖所示。

1			ols — APIC (173.36.2					abricQ		rtCon												
R	0	Consol	e D Debugger	↑↓ Network	{}	Style Edito	or Q	Perfor	mance	¢	Insp	pector	0	Memory	🗄 Sta	orage	🕇 Access	sibility	🗱 Appli	cation	9 3	 C
Û	∀ ep	g			8	ΠQ	\otimes	All	HTML	css	JS	XHR	Fon	nts Images	Media	a WS	Other	🗸 Dis	sable Cache	No Th	rottling	: ☆
Sta	м	D	File						Init	т	т	Size	^	B Hea	ders	Cookies	Reque	est l	Response	Timings	Sta	:k Trace 🔻
200	GET		epg-jup-epg.json?d	query-target= chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B		Queued: 11	12.09 mir	n Star	ted: 112.09	9 min	Download	ded: 113.1	9 min	
200	GET		fltCnts.json?subscri	iption=yes&_dc=	= 1650	484171126			/:19	j	1	481 B		Request Tir	ming							
200	GET		epg-jup-epg.json?c	query-target=chi	ldren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B		Blocked:		0 ms						
200	GET		epg-jup-epg.json?_	_dc=1650484171	200				/:19	j	1	726 B		DNS Resolu	ution:	0 ms						
200	GET		fitCnts.json?subscri	iption=yes&_dc=	= 1650	484171676			/:19	j	1	481 B		Connecting TLS Setup:	g:	0 ms						
200	GET	۵	epg-jup-epg.json?c	query-target=chi	Idren	&target-su	ibtree-cla	ss 🦡	/:19	j	7	67 B		Sending:		0 ms						
200	GET		epg-jup-epg.json?c	query-target=chi	ldren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B		Waiting:							1.10 m	in
288	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B		Receiving:							0 ms	
200	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	1	566 B		0								
200	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss: v@	/:19	j	7	67 B										
200	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	1	527 B										
200	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B										
200	GET		epg-jup-epg.json?d	query-target=chi	Idren	&target-su	btree-cla	ss=fvF	/:19	j	7	67 B										
200	GET		epg-jup-epg.json?c	query-target=chi	Idren	&target-su	btree-cla	ss=fvC	/:19	j	7	67 B										
200	GET		epg-jup-epg.json?c	query-target=sul	btreeð	ktarget-sul	btree-clas	ss=fvR	/:19	j	6	30 B	~									
Ō	18 re	auests	4.86 KB / 16.40 K	(B transferred	Finis	h: 114.01 n	nin															

等待1.1分鐘以供APIC響應的瀏覽器示例

特定UI頁面的增強功能

「策略組」頁:

思科漏洞ID CSCvx14621 - APIC GUI在交換矩陣頁籤中的IPG策略上緩慢載入。

Inventory頁面下的介面:

思科錯誤ID <u>CSCvx90048</u>—「Layer 1 Physical Interface Configuration」(第1層物理介面配置)Operational(操作)頁籤的初始載入為長/導致「凍結」。

客戶端>伺服器延遲的一般建議

預設情況下,某些瀏覽器(如Firefox)允許每個主機擁有更多的Web連線。

- 檢查此設定是否可以在使用的瀏覽器版本上配置
- 對於多查詢頁(如「策略組」頁),這一點更為重要

VPN和與APIC的距離會根據客戶端瀏覽器請求和APIC響應旅行時間增加整體UI延遲。在APIC上本地的跳轉框可顯著減少瀏覽器到APIC的旅行時間。

檢查Long-Web請求

如果Web伺服器(APIC上的NGINX)處理大量的長Web請求,這可能會影響並行接收的其他請求 的效能。

對於具有分散式資料庫的系統(如APIC)尤其如此。單個API請求可能需要向結構中的其他節點傳 送額外的請求和查詢,這可能會導致預期的更長的響應時間。這些長Web請求在短時間幀內突發 ,會增加所需的資源量,導致意外的更長的響應時間。此外,接收的請求隨後會超時(90秒),導 致從使用者角度而言出現意外的系統行為。

系統響應時間 — 啟用伺服器響應時間的計算

在4.2(1)+中,使用者可以啟用「系統效能計算」,跟蹤並突出顯示花費時間處理的API請求。

altalta cisco	APIC (s	ite2)									admin 🔍 🤇	00	00
System	Tenants	Fabric	Virtual Networking	Admin Op	erations Apps	Integrations							
QuickStart	I Dashboard	Controllers	System Settings	Smart Licensing	I Faults I History	I Config Zones	Active Sessions	Security					
System Se	ttings	090	System Perform	nance									00
APIC C	onnectivity Prefere	nces									Sustem Desponse Time	Enulte	History
APIC P	assphrase									_	System Response Time	rauns	matory
🗧 BD Enfo	proed Exception Lis												0 <u>*</u>
📄 BGP Ro	ute Reflector		Properties										
Control	Plane MTU		Ci	iculation: Disabled	Enabled								
COOP (Group		Response Threst	old (ms): 85000	0								
P Date an	d Time		Freque	ncy (sec): 300	0								
Endpoir	nt Controls		Top Slowest	Requests: 5	0								
Fabric :	Security												
Fabric-	Wide Settings												
Giobal	AES Passphrase Er	cryption Setti.											
	Endpoints (beta)												
	in Irv												
Load B	slancer												
Port Tra	icking												
Proxy P	olcy												
PTP an	d Latency Measure	ment											
🖗 Quota													
Remote	Leaf POD Redund	ancy Policy											
E System	Alias and Banners												
F System	Global GIPo												
System	Performance												

可以從系統 — 系統設定 — 系統效能啟用計算

啟用「計算」後,使用者可以導航到控制器下的特定APIC,檢視最近300秒內最慢的API請求。

cisco	APIC (s	site2)										admin 🔇 🔿 🗗 💱	00
System	Tenants	Fabric	Virtual Network	ing Admin Operations	Apps Integ	rations							
QuickStart	Dashboard	I Controlle	rs System Setti	ngs Smart Licensing Faults	I History I Config	Zones I /	Active Session	s I Security					
Controllers			©3¢	Server Response Time									0.0
O Quick St	art											Policy	History
Topology												Policy	matory
Controlle	MS												0 ±
V 🗊 apic1	(Node-1)			Properties									
	luster as Seen by	Node		Average Response Time (n Requests Serv	15): 489 ed: 77								
> 🔤 🛛	ontainers			Exceeded Requests	0								
E E E	aupment Fans			Slowest requests in the las	Host Name	Method	 Order 		Response		Start Time	URL	
E	uipment Sensor			300 seconds				Code	Size (Bytes)	Time			
	terfaces				::fff:172.21.208.205	GET	1	503	257	90811	2023-01-03T	/api/node/class/faultinfo.json	
	emory Slots				::fff:172.21.208.205	GET	2	503	170	90688	2023-01-03T	/api/node/class/eventRecord.json	
	TP Decails				mm:10.1.0.1	GET	3	503	169	90494	2023-01-03T	/api/node/mo/topology/pod=2 ison	
	wer Suppy Unic	`			-89-127.0.0.1	GET	4	503	172	00473	2023-01-037	Inclined a Iclass Boo Suttem ison	
l i s	over Response T	ime				0.57		503	100	00000	2020 01 007	lapinosijoissijopojalen jaar	
	MU patch version					GET	5	503	169	90331	2023-01-031	/api/class/nrmwareCtrirkunning.json	
	orage												
> 🖨 apic2	(Node-2)												
> apic3	(Node-3)												
Controlle	r Policies												
> 🚞 Retention	n Policies												

System - Controllers - Controllers資料夾 — APIC x — 伺服器響應時間

APIC API使用注意事項

確保指令碼不會損害Nginx的一般指標

- 每個APIC運行自己的NGINX DME。
 - 。只有APIC 1的NGINX處理對APIC 1的請求。APIC 2和3的NGINX不處理這些請求。
- 一般來說,在很長一段時間內每秒鐘有15個以上的API請求會使NGINX衰弱。
 - 如果找到,則減少請求的攻擊性。
 - ◎ 如果無法修改請求主機,請考慮對APIC執行<u>NGINX速率限制</u>。

解決指令碼效率低下問題

- 在每個API請求之前不要登入/註銷。
 - 一個登入會話的預設超時為10分鐘。此同一會話可用於多個請求,並可以刷新以延長有 效時間。
 - 。請參閱思科APIC REST API配置指南 訪問REST API 身份驗證和維護API會話。
- 如果您的指令碼查詢共用父級的多個DN,而不是使用查詢過濾器將查詢摺疊為單個邏輯<u>父級</u> <u>查詢</u>。

◎ 請參閱<u>Cisco APIC REST API配置指南 — 合成REST API查詢 — 應用查詢範圍篩選器</u> 。

• 如果您需要更新對象或對象類,請考慮<u>websocket預訂</u>而不是快速API請求。

NGINX請求限制

4.2(1)+版本中提供,使用者可以獨立啟用針對HTTP和HTTPS的請求限制。

System Tena	ints Fabric	Virtual Networking	Admin	Operations	Apps	Integrations
Inventory	Fabric Policies A	Access Policies				
Policies	(C)	Management A	nccess - def	fault		
Ouick Start		Management A	iccess - uei	Iduit		
> Pods						
> 🚞 Switches						
> 🖿 Modules		Properties				
> 🚞 Interfaces		ropences	Name: defau	ult		
✓		De	scription: opti	onal		
V 🖿 Pod						
> 🚞 Date and	Time	UTTO				
> 🖿 SNMP		Adr	nin State: Disa	bled	~	
V 🖿 Manager	nent Access		Port: 80		0	
E defaul	t		Redirect: Disa	bled	1	
> Switch		Allov	v Origins: http:	//127.0.0.1:8000		
> Global		Allow Cre	edentials: Di	sabled Enable	be	
> Monitoring		Desman	Threelies	inabled Eachle	<u> </u>	
> E Troubleshool	ting	LITTE	Throttle:	Sabled Eliable		
> 🚞 Geolocation		Adr	nin State: Enab	bled	~	
> 🚞 Macsec		1	Port: 443		0	
> E Analytics		Allov	v Origins: http:	//127.0.0.1:8000		
Tenant Quota	a	Allow Cre	edentials: Di	sabled Enable	bd)	
Annotations		SSL F	Protocols: 🗖 TI	Sv1		
			2 ΤΙ	.Sv1.1		
				.Sv1.2		
			H Param: (10	2048	4096 No	ne
		Request	Throttle: Di	sabled Enable	ed	
		Thro	ttle Rate: 20	Req	uests/Minute	\sim

交換矩陣 — 交換矩陣策略 — 策略資料夾 — 管理訪問資料夾 — 預設

啟用時:

• 重新啟動NGINX以應用配置檔案更改

新區域httpsClientTagZone將寫入nginx config

- 限制速率可以設定為每分鐘請求數(r/m)或每秒請求數(r/s)。
- 請求限制依賴於NGINX<u>中包括的速率限制實施</u>
 - ◎ 針對/api/URI的API請求使用使用者定義的限制速率+突發=(限制速率x 2)+無延遲
 - /api/aaaLogin和/api/aaaRefresh有一個不可配置的限制(zone aaaApiHttps),速率
 限製為2r/s + burst=4 + nodelay
 - ·基於每個客戶端IP地址跟蹤請求限制
 - 。源自APIC自身IP(UI + CLI)的API請求繞過限制
 - 。任何超過使用者定義的限制速率+突發閾值的客戶端IP地址都會收到來自APIC的503響應

- 。這些503可在訪問日誌中關聯
- error.log將包含指示何時啟用限制(區域httpsClientTagZone)以及針對哪些客戶端主機的 條目

<#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"

通常,Request Throttle僅用於保護伺服器(APIC)免受查詢密集型客戶端引起的DDOS樣症狀的影響。在應用/指令碼邏輯中瞭解並隔離請求攻擊型客戶端,以獲得最終解決方案。

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)。