低速のAPIC GUIのトラブルシューティング

内容

概要

クイックスタート

背景説明

- WebサーバとしてのAPIC:NGINX
- 関連ログ

方法

- 最初のトリガーの分離
- NGINXの使用状況と健全性の確認
  - Access.logエントリの形式
  - Access.logの動作
  - NGINXリソース使用状況の確認
- コアの確認
- クライアントからサーバへの遅延の確認
  - ブラウザ開発ツールの「ネットワーク」タブ
  - 特定のUIページの機能強化
  - クライアント>サーバ遅延に関する一般的な推奨事項
- 長いWeb要求の確認
  - システム応答時間 – サーバー応答時間の計算を有効にする

APIC APIの使用上の考慮事項

- スクリプトがNginxに損害を与えないことを確認するための一般的なポインタ
- スクリプトの非効率性への対処
- NGINX要求スロットル

概要

このドキュメントでは、APIC GUIの動作が遅い場合の一般的なトラブルシューティング方法について説明します。

クイックスタート

APIC GUIの速度低下の問題は、スクリプト、統合、またはアプリケーションから送信されるAPI要求の割合が高いことが原因であることがよくあります。APICのaccess.logには、処理された各API要求が記録されます。APICのaccess.logは、Github Datacenterグループの[aci-tac-scripts](#)プロジェクト内の[Access Log Analyzer](#)スクリプトを使用してすばやく分析できます。

背景説明

WebサーバとしてのAPIC:NGINX

NGINXは、各APICで使用できるAPIエンドポイントを担当するDMEです。NGINXがダウンしている場合、API要求は処理できません。NGINXが輻輳している場合、APIは輻輳しています。各APICは独自のNGINXプロセスを実行するため、アグレッシブクエリアの対象がそのAPICだけであれば、そのAPICでNGINXの問題が発生する可能性があります。

APIC UIは、複数のAPI要求を実行して各ページにデータを入力します。同様に、APICのすべての「show」コマンド（NXOSスタイルのCLI）は、複数のAPI要求を実行し、応答を処理し、ユーザに提供するPythonスクリプトのラッパーです。

関連ログ

ログファイル名	場所	どのテクニカルサポートに属しているか	注釈
アクセス。ログ	/var/log/dme/log	APIC 3of3	ACIに依存せず、API要求ごとに1行を提供
エラーログ	/var/log/dme/log	APIC 3of3	ACIに依存せず、nginxエラーを表示（スロットリングを含む）
nginx.bin.log	/var/log/dme/log	APIC 3of3	ACI固有、DMEトランザクションを記録
nginx.bin.warnplus.log	/var/log/dme/log	APIC 3of3	ACI Specificには警告+重大度のログが含まれる

方法

最初のトリガーの分離

影響を受ける内容

- どのAPICが影響を受けますか。1つ、多く、またはすべてのAPICですか。
- UI、CLIコマンド、またはその両方を介して速度低下が見られる場所はどこですか。
- 特定のUIページまたはコマンドのうち、処理速度が遅いものはどれですか。

速度低下はどのように発生しますか。

- これは、1人のユーザに対して複数のブラウザで発生しますか。
- 複数のユーザが速度低下を報告しますか。それとも、単一のユーザまたはユーザのサブセットだけを報告しますか。

- 影響を受けるユーザは、ブラウザからAPICまで同様の地理的な場所やネットワークパスを共有していますか。

最初に速度低下に気付いたのはいつですか。

- ACI統合またはスクリプトは最近追加されましたか。
- ブラウザ拡張機能は最近有効になりましたか。
- ACIの設定に最近の変更はありましたか。

NGINXの使用状況と健全性の確認

Access.logエントリの形式

access.logはNGINXの機能であるため、APICには依存しません。各行は、APICが受信した1つのHTTP要求を表します。APICのNGINXの使用状況を理解するには、この口グを参照してください。

ACIバージョン5.2+のデフォルトのaccess.log形式：

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

次の行は、moquery -c fvTenantを実行したときのaccess.logエントリを表しています。

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

access.logエントリの例をlog_formatにマップします。

log_formatフィールド	例の内容	注釈
$remote_addr	127.0.0.1	この要求を送信したホストのIP
$http_x_real_ip	-	プロキシが使用中の場合は最後のリクエスタのIP
$remote_user	-	一般的には使用されません。nginx.bin.logを確認して、どのユーザが

		ログインして要求を実行したかを追跡します
$time_local	2022年4月7日：20:10:59 +0000	要求が処理された日時
要求($r)	GET /api/class/fvTenant.xml HTTP/1.1	Httpメソッド(GET、POST、DELETE)およびURI
ステータス($s)	200	HTTP応答ステータスコード
$body_bytes_sent送信	1586	応答ペイロードサイズ
$http_referer	-	-
$http_user_agent	Python-urllib	要求を送信したクライアントのタイプ

Access.logの動作

長期間にわたる高レートの要求バースト：

- 1秒あたり15以上の要求を継続的にバーストすると、UIの処理速度が低下する可能性がある
- クエリを実行するホストを特定する
- クエリのソースを減らすか無効にして、APICの応答時間が向上するかどうかを確認します。

4xxまたは5xxの一貫した応答：

- 見つかった場合、nginx.bin.logからエラーメッセージを特定します

NGINXリソース使用状況の確認

NGINXのCPUとメモリの使用状況は、APICからtopコマンドを使用して確認できます。

<#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

NGINXリソースの使用率が高いと、高い割合で処理される要求に直接関連する可能性があります。

コアの確認

NGINXのクラッシュは、APIC GUIの速度低下の問題では一般的ではありません。ただし、NGINXコアが見つかった場合は、TAC SRに接続して分析します。コアを確認する手順については、『ACI Techsupport guide』を参照してください。

クライアントからサーバへの遅延の確認

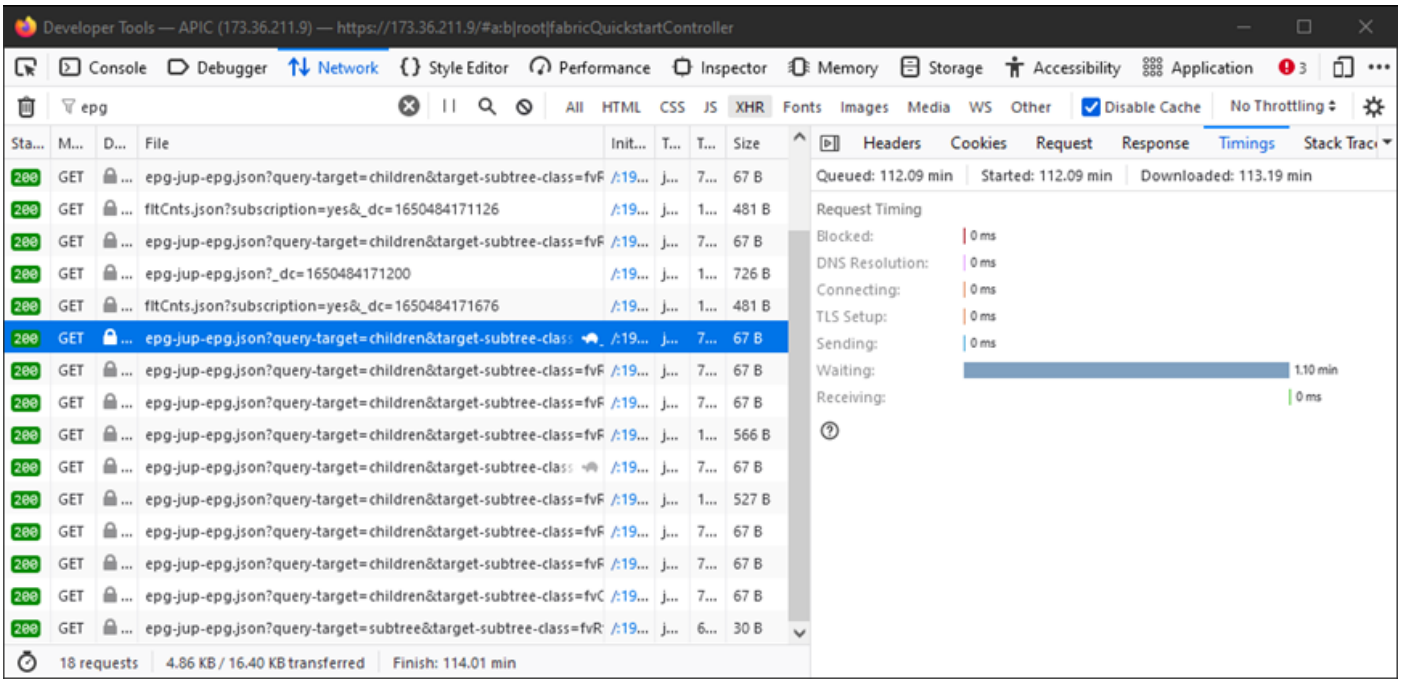
迅速な要求が見つからなくてもユーザのUIの速度が低下し続ける場合は、クライアント（ブラウザ）からサーバ(APIC)への遅延が問題である可能性があります。

これらのシナリオでは、ブラウザからAPICへのデータパス（地理的な距離、VPNなど）を検証します。可能であれば、分離するAPICと同じ地域またはデータセンターにあるジャンプサーバからのアクセスを導入してテストします。他のユーザが同様の遅延を示しているかどうかを検証します。

ブラウザ開発ツールの「ネットワーク」タブ

すべてのブラウザには、ブラウザ開発ツールキット(通常はNetworkタブ内)を介してHTTP要求および応答を検証する機能があります。

このツールを使用すると、図に示すように、ブラウザソースの要求の各段階にかかる時間を検証できます。



APICが応答するまで1.1分待機するブラウザの例

特定のUIページの機能強化

ポリシーグループページ：

Cisco Bug ID CSCvx14621:「ファブリック」タブのIPGポリシーでAPIC GUIのロードが遅くなります。

インベントリページのインターフェイス：

Cisco Bug ID CSCvx90048:「レイヤ1物理インターフェイス設定」操作タブの初期読み込みが長く、「フリーズ」を誘発します。

クライアント>サーバ遅延に関する一般的な推奨事項

Firefoxなどの一部のブラウザでは、ホストごとにより多くのWeb接続がデフォルトで許可されます。

- この設定が、使用されているブラウザのバージョンで設定可能かどうかを確認します
- これは、ポリシーグループページなどのマルチクエリページでは重要です

VPNとAPICへの距離により、クライアントブラウザの要求とAPICの応答所要時間に応じて、全体的なUIの速度が向上します。APICに対して地理的にローカルなジャンプボックスにより、ブラウザのAPIC移動時間が大幅に短縮されます。

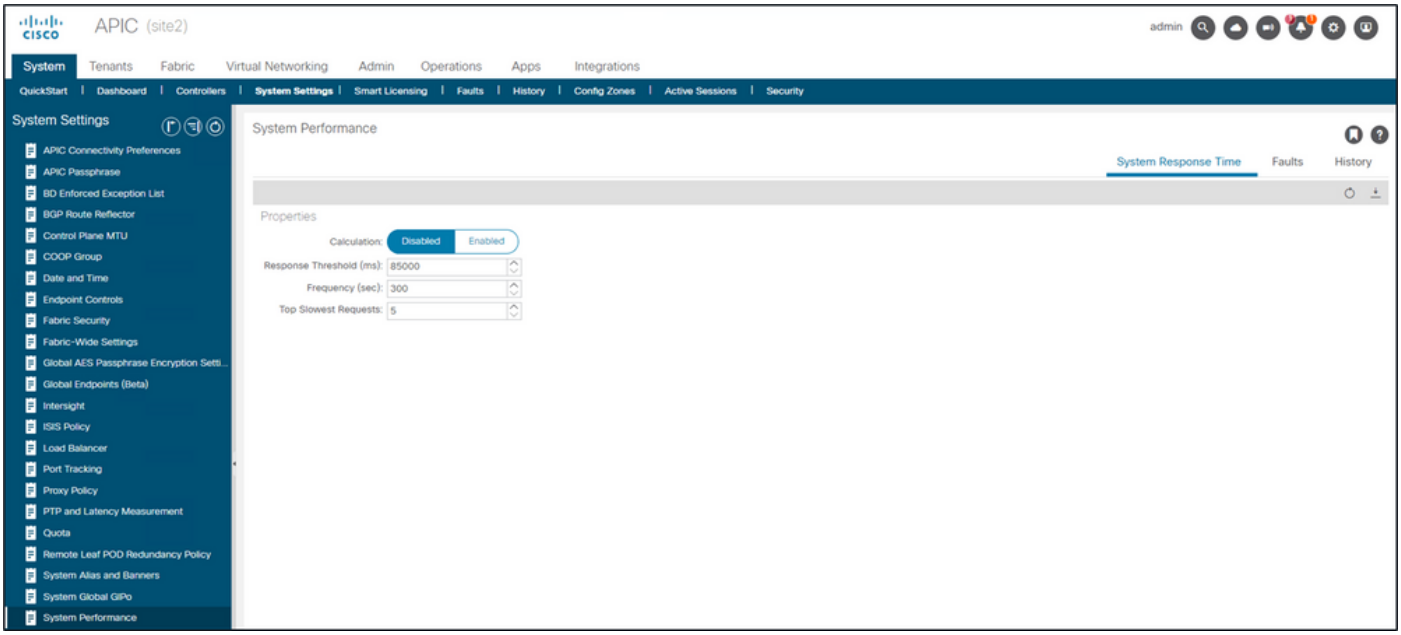
長いWeb要求の確認

Webサーバ（APIC上のNGINX）が大量の長Web要求を処理する場合、並行して受信される他の要求のパフォーマンスに影響を与える可能性があります。

これは、APICなどの分散データベースを持つシステムに特に当てはまります。単一のAPI要求でファブリック内の他のノードに送信される追加の要求とルックアップが必要になる場合があり、その結果、応答時間が予想以上に長くなる可能性があります。このような長いWeb要求が短い時間枠内にバーストすると、必要なリソースの量が増え、応答所要時間が予想外に長くなる可能性があります。さらに、受信した要求がタイムアウトする（90秒）可能性があり、ユーザの観点から予期しないシステム動作が発生します。

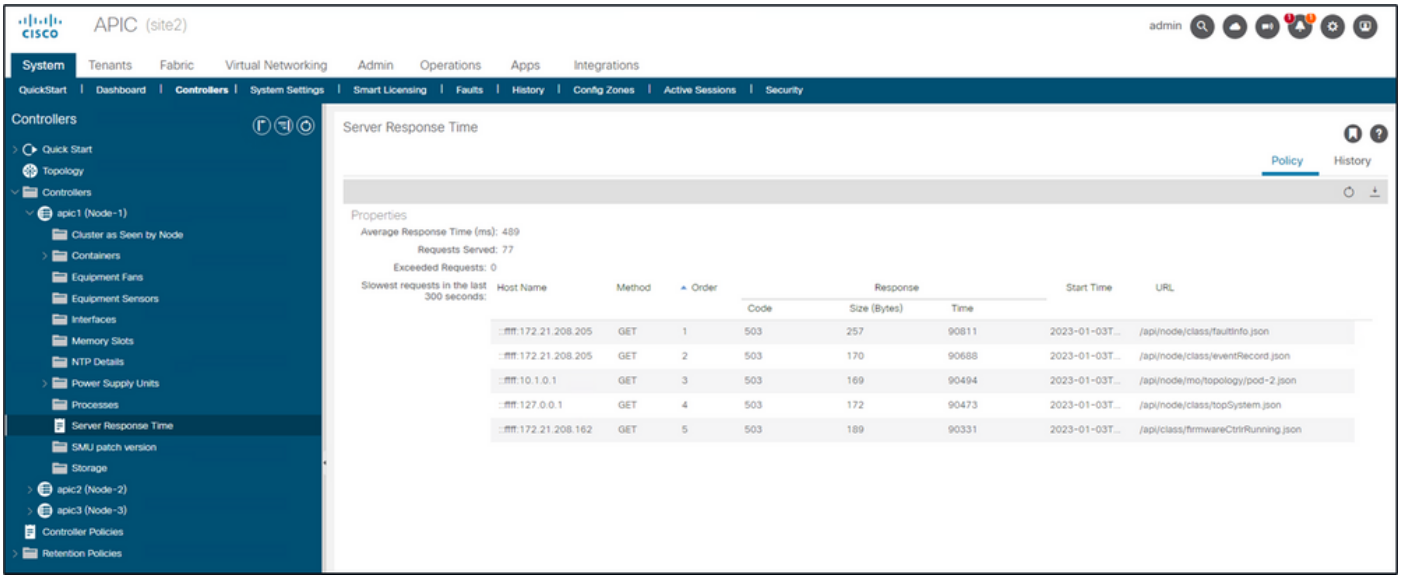
システム応答時間 – サーバー応答時間の計算を有効にする

4.2(1)+では、処理に時間がかかったAPI要求を追跡して強調表示する「システムパフォーマンス計算」を有効にできます。



計算は、[システム] - [システム設定] - [システムパフォーマンス]から有効にできます。

「計算」が有効になると、ユーザはコントローラの下の特定のAPICに移動して、過去300秒以内で最も遅いAPI要求を表示できます。



システム – コントローラ – コントローラフォルダ – APIC x – サーバ応答時間

APIC APIの使用上の考慮事項

スクリプトがNginxに損害を与えないことを確認するための一般的なポインタ

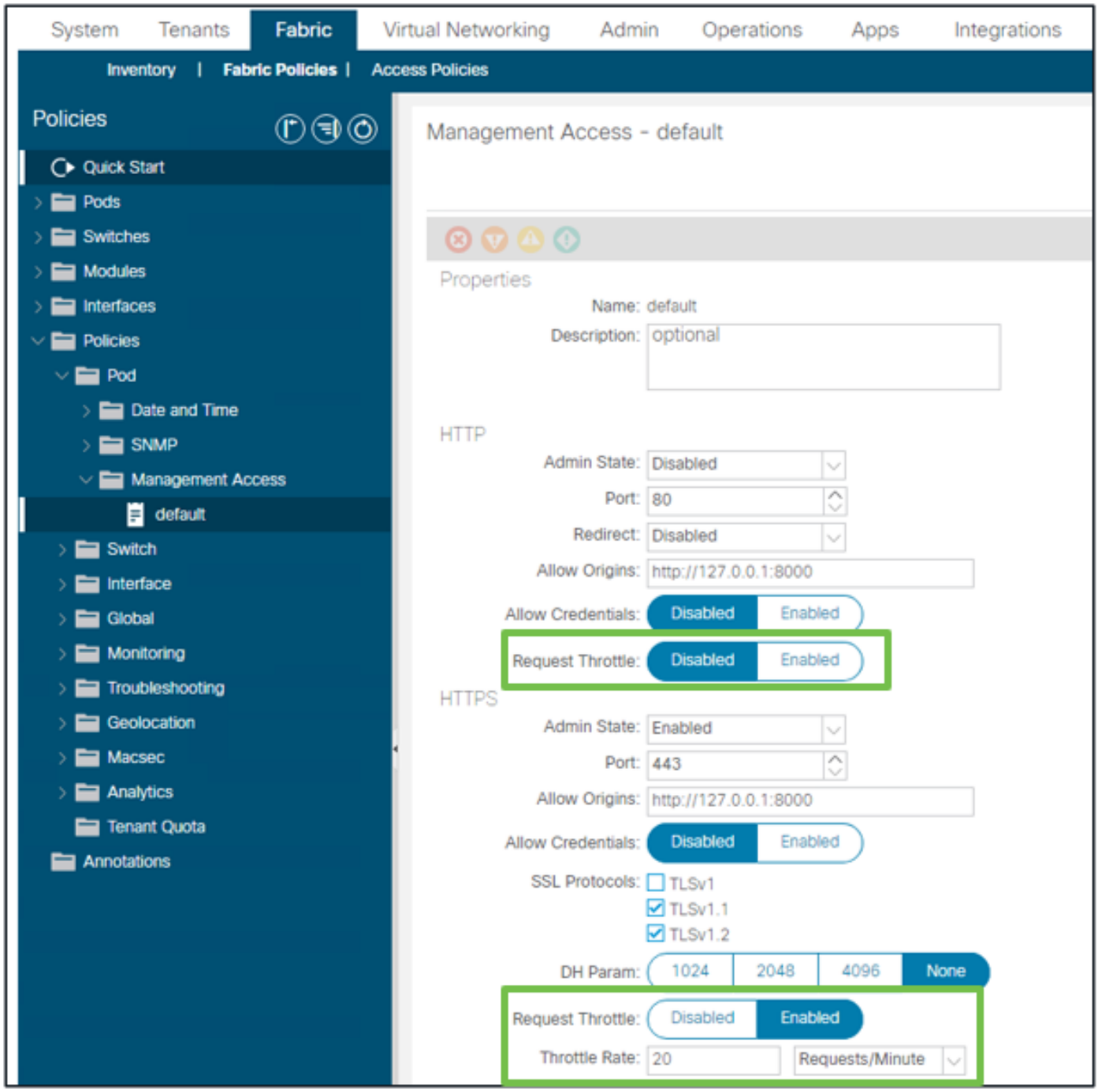
- 各APICは独自のNGINX DMEを実行します。
  - APIC 1のNGINXだけがAPIC 1への要求を処理します。APIC 2および3のNGINXはこれらの要求を処理しません。
- 一般的に、1秒あたり15以上のAPI要求が長時間にわたって発生すると、NGINXの機能が低下します。
  - 検出された場合は、要求の攻撃性を軽減します。
  - 要求ホストを変更できない場合は、APICのNGINXレート制限を検討します。

スクリプトの非効率性への対処

- 各API要求の前にログイン/ログアウトしないでください。
  - 1つのログインセッションのデフォルトのタイムアウトは10分です。この同じセッションを複数の要求に使用し、有効期間を延長するために更新できます。
  - 『Cisco APIC REST APIコンフィギュレーションガイド – REST APIへのアクセス – APIセッションの認証と維持』を参照してください。
- スクリプトで、クエリーフィルタを使用してクエリーを1つの論理親クエリーにまとめるのではなく、親を共有する多くのDNをクエリーする場合。
  - 『Cisco APIC REST APIコンフィギュレーションガイド – REST APIクエリの構成 – クエリースコーピングフィルタの適用』を参照してください。
- オブジェクトまたはオブジェクトのクラスの更新が必要な場合は、迅速なAPI要求の代わりにwebsocketサブスクリプションを検討してください。

NGINX要求スロットル

4.2(1)+以降で使用可能なユーザは、HTTPとHTTPSに対して個別に要求スロットルを有効にできます。



Fabric - Fabric Policies - Policiesフォルダ – Management Accessフォルダ – デフォルト

有効な場合：

- NGINXが再起動され、コンフィギュレーションファイルの変更が適用されます
  - 新しいゾーンhttpsClientTagZoneがnginx configに書き込まれます
- スロットル率は、Requests per Minute(r/m)またはRequests per Second(r/s)で設定できます。
- 要求スロットルは、NGINXに含まれるレート制限の実装に依存します
  - /api/ URIに対するAPI要求は、ユーザ定義のスロットルレート+ burst=（スロットルレートx 2）+ nodelayを使用します
    - 2r/s + burst=4 + nodelayでレート制限する/api/aaaLoginおよび/api/aaaRefresh用の設定不可能なスロットル(ゾーンaaaApiHttps)があります
  - 要求スロットルは、クライアントIPアドレスごとに追跡されます

- APIC self-ip(UI + CLI)から送信されたAPI要求はスロットルをバイパスします
- ユーザ定義のスロットルレート+バーストしきい値を超えるクライアントIPアドレスは、APICから503応答を受信します
- これらの503は、アクセスログ内で関連付けることができます
- error.logには、スロットリングがいつアクティブ化されたか(ゾーンhttpsClientTagZone)およびどのクライアントホストに対してアクティブ化されたかを示すエントリが含まれます

<#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"
```

原則として、要求スロットルは、クエリ集約型クライアントによって引き起こされるDDOSに似た症状からサーバ(APIC)を保護するためだけに機能します。アプリ/スクリプトロジックの最終的なソリューションについて、要求の多いクライアントを理解し、切り分けます。

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。