テストドキュメントの目次

内容

はじめに

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リソースの使用状況の確認
- 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

はじめに

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の処理速度が遅い問題は、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発生す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、1つのAPICだけがNGINX問題を抱え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入力します。同様に、すべてのAPIC 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非依存、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（ファイル名を指定して実行）	/var/log/dme/log	APIC 3of3	ACI固有には警告+重大度のログが含まれる

方法

最初のトリガーの分離

影響を受けるものは何か

- どのAPICが影響を受けるか（1つ、複数、またはすべてのAPIC）。
- 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処理速度が遅いのはどれですか。

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見られますか。
- 複数のユーザが速度低下を報告していますか。それとも、単一のユーザまたは一部のユーザだけを報告していますか。
- 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すか。

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形式は次のとおりです。

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

この行は、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要求者のIP
$remote_user	-	通常は使用されません。nginx.bin.logを確認し、どのユーザがログインして要求を実行したかを追跡する
$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40以上の要求が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
- クエリー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。

一貫した4xxまたは5xx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

APICのaccess.logは、Github Datacenterグループ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。

NGINXリソースの使用状況の確認

NGINXのCPUおよびメモリの使用量は、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、処理される要求の割合が高くな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

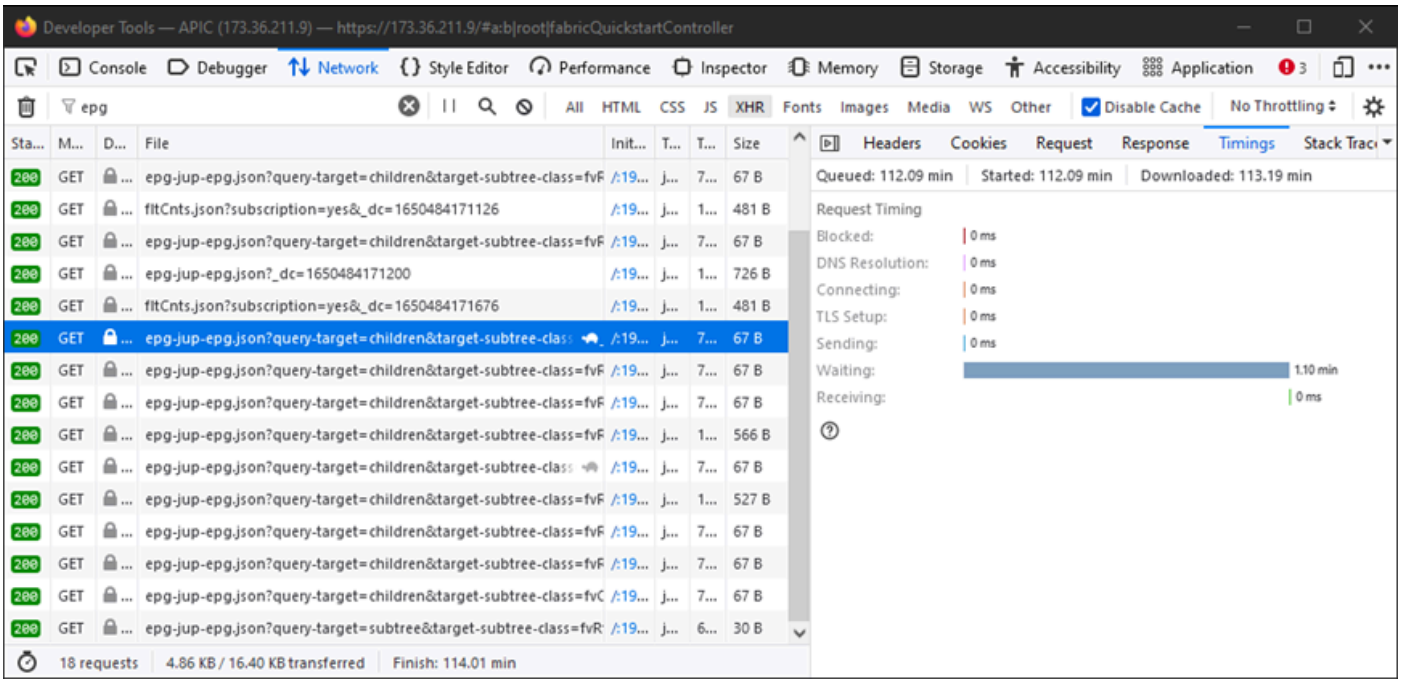
迅速な要求が見つからず、ユーザのUIの速度が低下し続ける場合、問題はクライアント（ブラウザ）からサーバ(APIC)への遅延である可能性があります。

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。他のユーザの遅延が同程度であ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

すべてのブラウザは、ブラウザ開発ツールキット(通常はNetworkタブ内)を使用して、HTTP要求および応答を検証できます。

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ポリシーでの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、フリーズを引き起こします。

クライアント>サーバ遅延に関する一般的な推奨事項

Firefoxなどの特定のブラウザでは、デフォルトでホストごとにより多くのWeb接続を使用できます。

- 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より重要になります

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応答の移動時間に応じて、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。

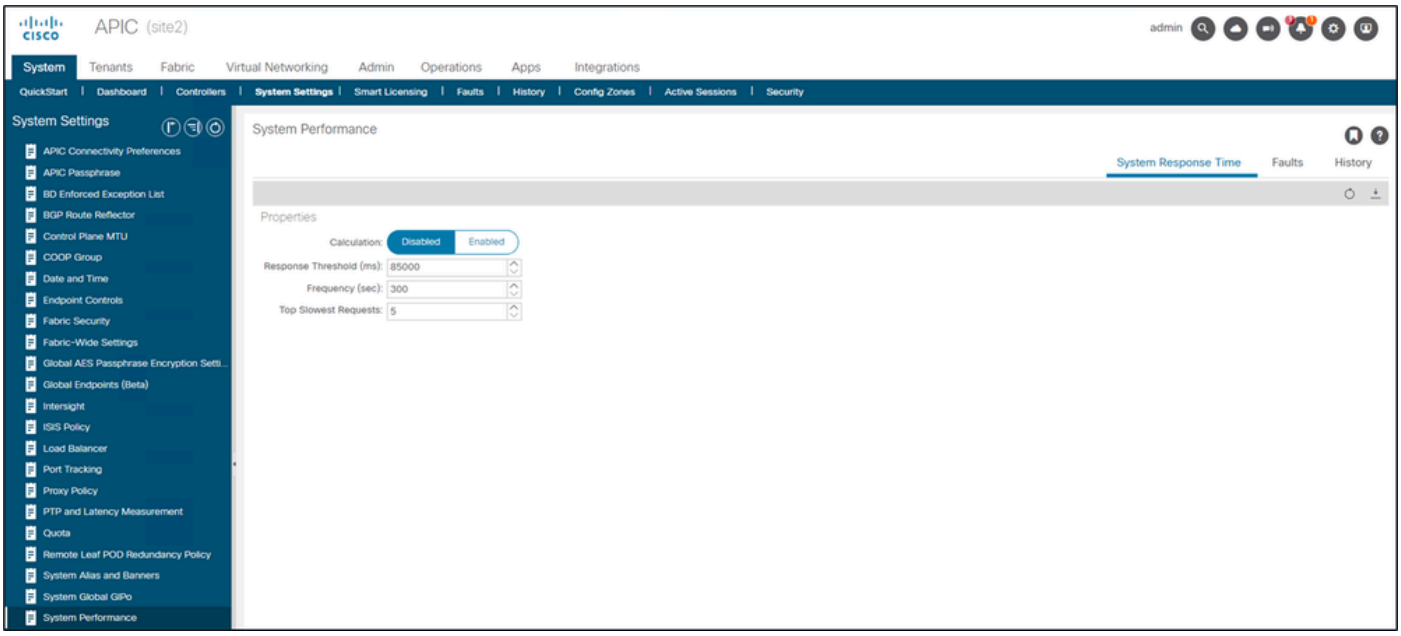
長いWeb要求の確認

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。1つのAPI要求で、ファブリック内の他のノードに対して追加の要求とルックアップを送信する必要があるため、応答時間が大幅に長くなる可能性があります。このような長いWeb要求が短い時間内にバーストすると、必要なリソースの量が増え、応答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、ユーザの観点から見ると予期しないシステム動作が発生します。

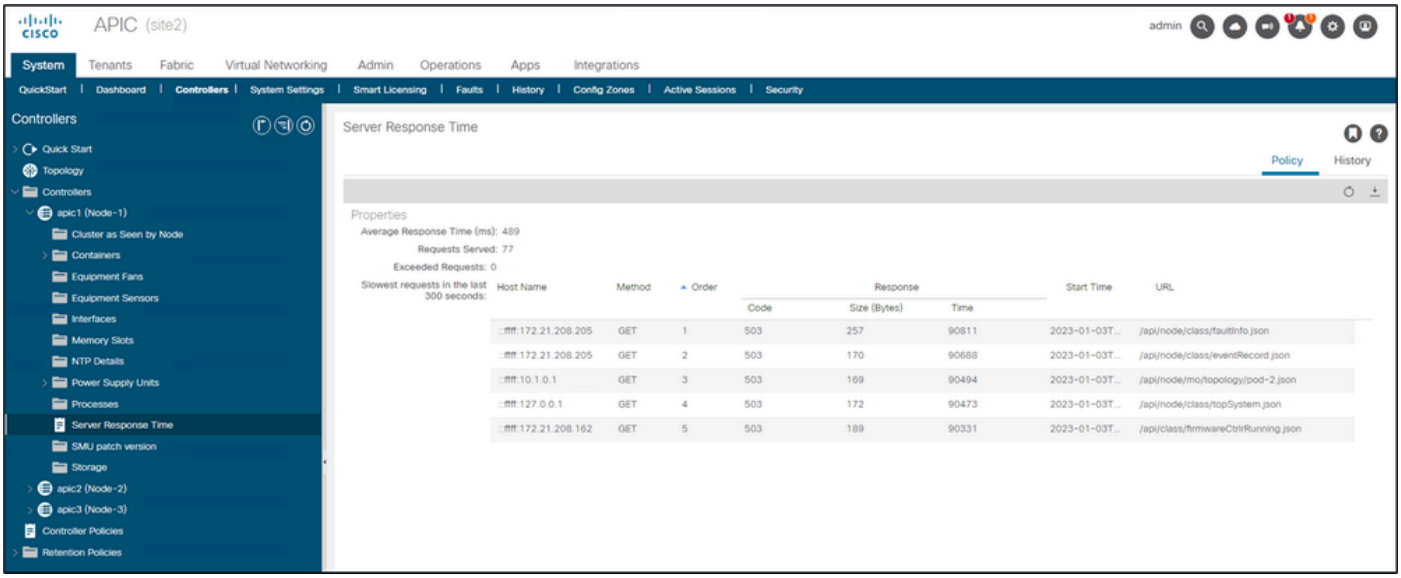
システム応答時間 – サーバー応答時間の計算を有効にする

4.2(1)+では、処理時間が長かったAPIリクエスト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秒以内の最も遅い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。
- 一般に、1秒あたり40以上のAPIリクエストが長期間にわたって発生すると、NGINXの機能が低下します。
  - 見つかった場合は、要求のアグレッシブさを減らします。

- 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てください。

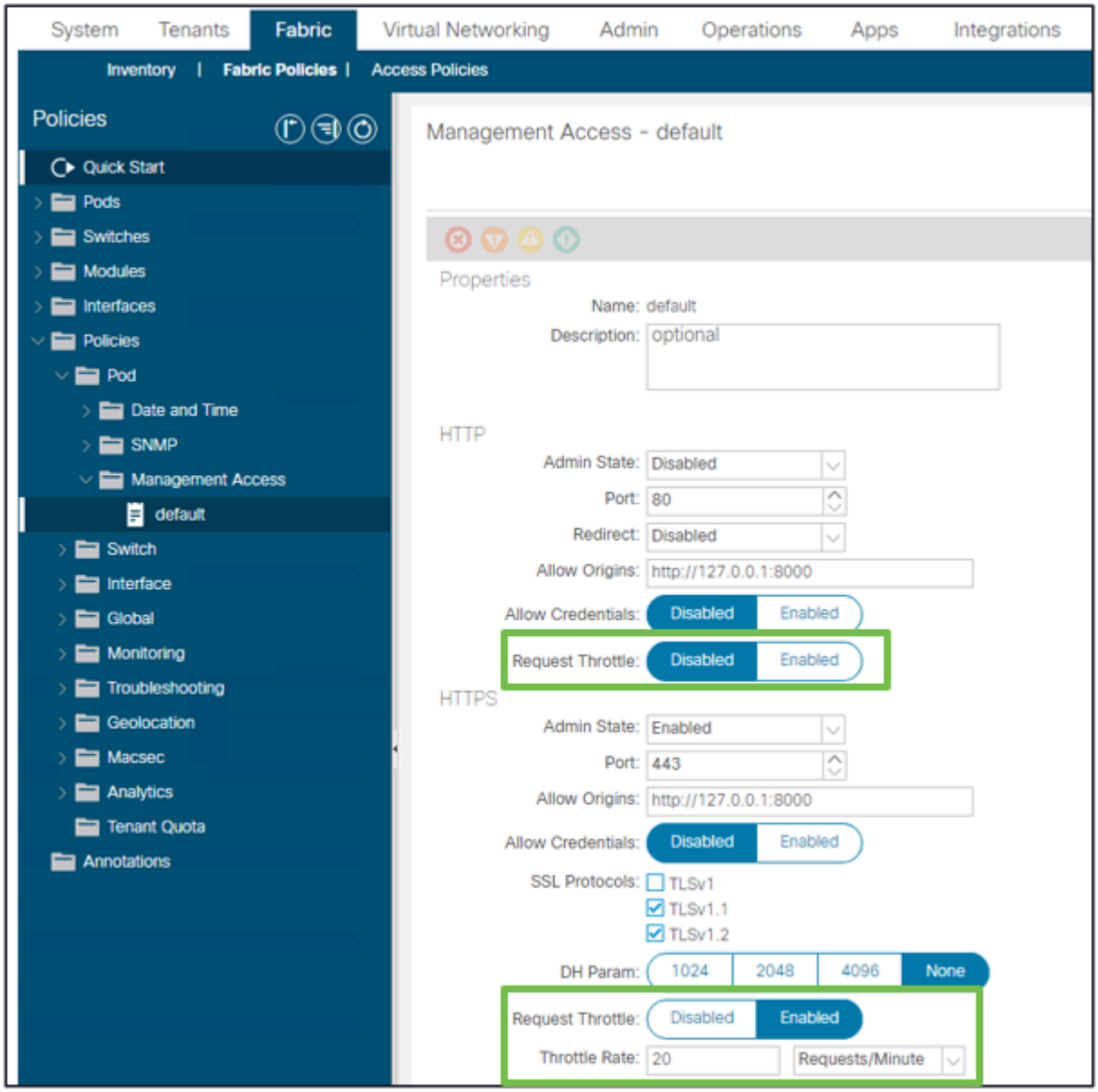
スクリプトの非効率性に対処

- 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、親を共有する多くのDNを照会する場合、クエリーフィルタを使用してクエリを1つの論理親クエリに折りたたむ代わりに、
  - 『Cisco APIC REST APIコンフィギュレーションガイド – REST APIクエリの作成 – クエリースコープフィルタの適用』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、高速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)+で使用可能で、ユーザはHTTPおよびHTTPSに対して個別に要求スロットルを有効にできます。

注:ACIバージョン6.1(2)以降、この機能でサポートされる最大レートは10,000 r/mから40 requests per second(r/s)または2400 requests per minute(r/m)に低下しました。



ファブリック – ファブリックポリシー – ポリシーフォルダ – 管理アクセスフォルダ – デフォルト

有効な場合：

- コンフィギュレーションファイルの変更を適用するためにNGINXが再起動されます
  - 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でのレート制限である/api/aaaLoginおよび/api/aaaRefreshに対して、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、スロットリングがアクティブ化されたタイミング(ゾーン httpsClientTagZone)およびどのクライアントホストに対して実行されたかを示すエントリが含まれています

<#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、クエリアグレッシブ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)を保護するためにのみ機能します。アプリケーション/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、要求の厳しいクライアントを理解して切り分けます。

推奨事項

これらの推奨事項は、特に単一のソースで大量のAPI呼び出しを行わないシナリオにおいて、APICの負荷と運用上のストレスを軽減するように設計されています。これらのベストプラクティスを実装することで、ファブリック全体で不要な処理、ロギング、およびイベント生成を最小限に抑え、システムの安定性とパフォーマンスを向上させることができます。これらの提案は、個別のインシデントではなく集約された動作がAPIC負荷の一因となっている環境で特に有効です。

ACLロギングの無効化

通常の動作中は、ACLロギングがオフになっていることを確認してください。トラブルシューティングまたはデバッグのためにスケジュールされたメンテナンスの時間帯にのみ有効にしてください。継続的なロギングは、特に複数のスイッチにまたがる大量のトラフィックがドロップされた場合に、過剰な情報メッセージを生成し、APICのワークロードを増加させる可能性があります。

詳細については、『Cisco APICセキュリティ設定ガイド』（5.2.xガイドのリンク）を参照してください。
https://www.cisco.com/c/en/us/td/docs/dcn/aci/apic/5x/security-configuration/cisco-apic-security-configuration-guide-release-52x/security-policies-52x.html

syslogの変換を重大なイベントに制限する

重大度ALERTのsyslogメッセージだけがeventRecordsに変換されるようにシステムを設定します。ノイズの多いイベントによってAPICが過負荷状態にならないように、INFORMATIONレベル（ACL.loggingを含む）の変換は避けてください。

1. Fabric → Fabric Policies → Policies → Monitoring → Common Policy → Syslog Message Policies → Defaultの順に移動します。
2. ファシリティフィルタを調整して、syslogの重大度をアラートに設定します。

スケルチの重要でないイベントコード

監視に関係のないイベントコードを抑制（スケルチ）して、ノイズを低減する必要があります。イベントコードE4204939をスケルチするには、任意のAPIC CLIで次のコマンドを使用します。

```
bash
icurl -k -sX POST -d '<fabricInst><monCommonPol><eventSevAsnP code="E4204939" sev="squelched"/></monCom
```

確認するには、次のコマンドを実行します。

```
bash
icurl -k -sX GET 'https://localhost/api/node/class/eventSevAsnP.xml' | xmllint --format -
```

または、UIを使用して次の項目を確認します。
ファブリック>ファブリックポリシー>ポリシー>モニタリング>共通ポリシー>イベント重大度割り当てポリシー

NDサブスクリプション更新の最適化

3.2.2mまたは4.1.1gより前のバージョンのNDによって管理されるファブリックの場合、サブスクリプションの更新間隔を最適化するために、これらのバージョンのいずれか、またはそれ以降にアップグレードします。以前のバージョンはMOあたり45秒ごとに更新されるため、規模に応じて、1日あたり300,000を超えるAPIC要求が発生する可能性があります。更新されたバージョンで

は、サブスクリプションのタイムアウトが3600秒（1時間）に増加し、更新が1日あたり約5,000回に減少します。

Intersight関連のクエリーのモニタ

Intersightが有効なファブリックは、DCコネクタから定期的なトップシステムクエリ（15秒ごと）を生成し、APICの負荷を増大させます。
リリース6.1.2以降では、このクエリはオーバーヘッドを削減するために最適化されています。

レコードの保持ポリシーの調整

過剰なレコードの累積を防ぐには、eventRecord、faultRecord、およびhealthRecordのリテンションポリシーを1,000に設定します。これは、特定の運用活動に対して定期的にこれらのレコードを抽出する場合に特に便利です。運用およびトラブルシューティングの要件に対するモニタリングの精度を低下させることの影響を常に評価します。

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。