ルータから宛先へのモデル駆動型テレメトリ セッションの確立

テレメトリのストリーミングは、ネットワーク ヘルスをモニタするための新しいパラダイムです。関心のある設定データおよび運用データを Cisco IOS XR ルータから効率的にストリーミングするメカニズムを提供します。このストリーミングされるデータは、モニタリングおよびトラブルシューティングのため、構造化された形式でリモート管理ステーションに送信されます。

テレメトリ データを使用して、データ レイクを作成します。このデータを分析することにより、ネットワークのプロアクティブなモニタリング、CPUおよびメモリの使用率のモニタリング、パターンの特定、予測的な方法でのネットワークのトラブルシューティングを行い、自動化を使用した復元力のあるネットワークを作成するための戦略を考案します。

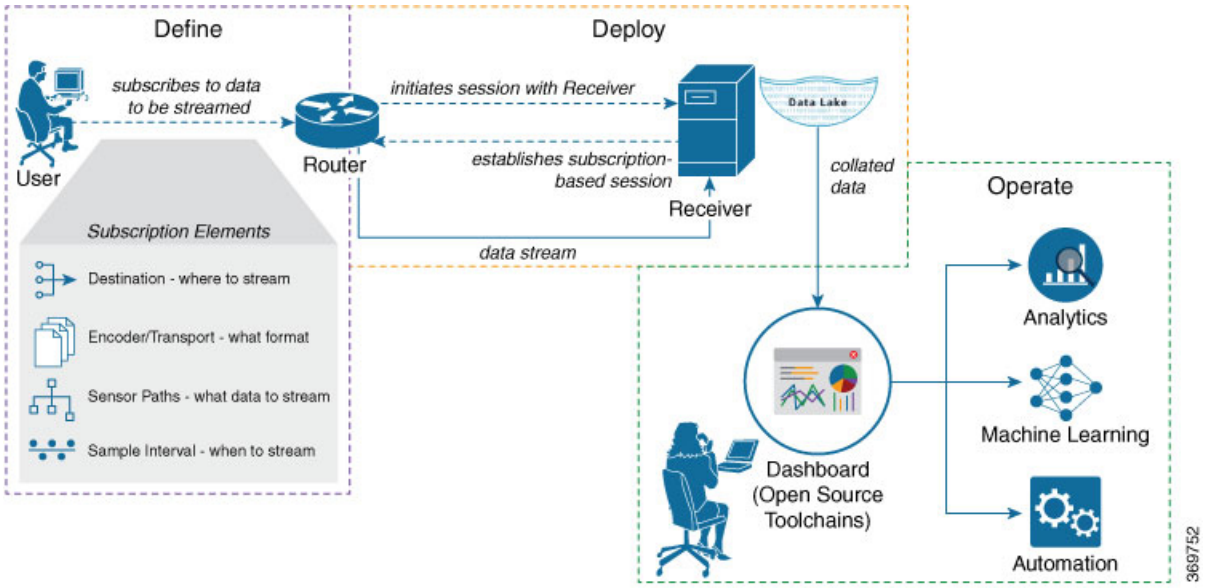
テレメトリは、関心のあるデータをセンサー パスの形でサブスクライブするサブスクリプション モデルで機能します。センサー パスは、OpenConfig データ モデルまたはネイティブのシスコ データ モデルを記述します。テレメトリのための OpenConfig データ モデルおよびネイティブ データ モデルには、バージョン管理のためのホスティング サービスを提供するソフトウェア開発プラットフォームである Github からアクセスできます。ルータと受信者の間にテレメトリ セッションを確立することによって、どのユーザがサブスクリプションを開始するかを選択します。セッションは、ダイヤルアウト モードまたはダイヤルイン モードのいずれかを使用して確立されます。これについては、「テレメトリを使用したネットワーク モニタリング戦略の拡張」の記事を参照してください。

（注） モデル駆動型テレメトリを使用したリアルタイムでのネットワーク管理がもたらす力については、こちらのビデオをご覧ください。

この記事では、ルータが受信者にダイヤルアウトしてテレメトリ セッションを確立するダイヤルアウト モードについて説明します。このモードでは、宛先とセンサーパスが設定され、1 つ以上のサブスクリプションにまとめられます。ルータは、サブスクリプション内の各宛先とのセッション確立を継続的に試行し、データを受信者にストリーミングします。サブスクリプションのダイヤルアウト モードは永続的です。セッションが終了すると、ルータは一定の間隔で受信者との新しいセッションを継続的に再確立しようとします。

次の図は、ダイヤルアウト モードの概要を示しています。

図 *1:* ダイヤルアウト モード



この記事では、CPU使用率のモニタリングを示す使用例を使用して、ネットワークの可視性の向上およびネットワークを安定させるための情報を得たうえでの意思決定にテレメトリ データのストリーミングがどのように役立つかを説明します。

• ネットワーク インフラストラクチャを計画するため、テレメトリ データを使用して CPU 使用率をモニタする （2 ページ）

ネットワークインフラストラクチャを計画するため、テレメトリ データを使用して CPU 使用率をモニタする

この使用例では、ダイヤルアウト モードで、テレメトリ データを使用して CPU 使用率をプロアクティブにモニタする方法を示します。CPU使用率をモニタすることにより、ネットワーク内のストレージ機能を効率化することができます。この使用例では、オープンソースの収集スタックにある、テレメトリ データの保存および分析に使用するツールについて説明します。

（注） データ モデル、オープンソースのコレクタ、エンコーディングを利用し、それらをモニタリング ツールに統合するためにモデル駆動型テレメトリを設定する方法については、こちらのビデオをご覧ください。

テレメトリには、次のワークフローがあります。

- **定義**：ルータから受信者にデータをストリーミングするためのサブスクリプションを定義します。サブスクリプションを定義するには、宛先グループとセンサーグループを作成します。
- **展開**：ルータは、サブスクリプションベースのテレメトリ セッションを確立し、受信者にデータをストリーミングします。ルータでサブスクリプションの展開を確認します。
- **運用**：オープンソース ツールを使用してテレメトリ データを消費および分析し、分析に基づいて必要なアクションを実行します。

始める前に

ルータと受信者の間に L3 接続があることを確認します。

ルータから受信者にデータをストリーミングするためのサブスクリプションを定義する

サブスクリプションを作成し、ルータから宛先にストリーミングする対象データを定義します。

手順

ステップ 1 ルータからテレメトリ データを収集する宛先を 1 つ以上作成します。宛先に関する詳細を格納する宛先グループを定義します。宛先グループには、宛先アドレス（ipv4 または ipv6）、ポート、トランスポート、およびエンコーディング形式を含めます。

例：

データ モデルを使用して宛先グループを作成する

この例では、ネイティブ データモデル `Cisco-IOS-XR-um-telemetry-model-driven-cfg.yang` を使用しています。

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-config>
    <source>
      <candidate/>
    </source>
    <filter>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <destination-groups>
          <destination-group>
            <destination-id>CPU-Health</destination-id>
            <ipv4-destinations>
              <ipv4-destination>
                <ipv4-address>172.0.0.0</ipv4-address>
                <destination-port>57500</destination-port>
                <encoding>self-describing-gpb</encoding>
                <protocol>
                  <protocol>tcp</protocol>
```

```
              </protocol>
            </ipv4-destination>
          </ipv4-destinations>
        </destination-group>
      </destination-groups>
    </telemetry-model-driven>
  </filter>
</get-config>
</rpc>
```

CLI を使用して宛先グループを作成する

```
##Configuration with tls-hostname##
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group CPU-Health
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 57500
Router(config-model-driven-dest-addr)#encoding self-describing-gpb
Router(config-model-driven-dest-addr)#protocol tcp
Router(config-model-driven-dest-addr)#commit
```

ここで、

- `CPU-Health` は、宛先グループの名前です
- `172.0.0.0` は、データがストリーミングされる宛先の IP アドレスです
- `57500` は、宛先のポート番号です
- `self-describing-gpb` は、データがエンコードされ、宛先にストリーミングされる形式です
- `tcp` は、データが宛先に転送されるプロトコルです

ステップ 2 センサーパスを使用して、ルータからストリーミングするデータのサブセットを指定します。センサー パスは、YANG データモデルの階層内のパスを表します。センサー パスを含むセンサーグループを作成します。

例：

データ モデルを使用して CPU 使用率のセンサーグループを作成する

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <sensor-groups>
          <sensor-group>
            <sensor-group-identifier>Monitor-CPU</sensor-group-identifier>
            <sensor-paths>
              <sensor-path>

<telemetry-sensor-path>Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization</telemetry-sensor-path>

              </sensor-path>
            </sensor-paths>
          </sensor-group>
        </sensor-groups>
```

```
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

CLI を使用して CPU 使用率のセンサーグループを作成する

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group Monitor-CPU
Router(config-model-driven-snsr-grp)# sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
Router(config-model-driven-snsr-grp)# commit
```

ここで、

- `Monitor: CPU` は、センサーグループの名前です
- `Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization` は、データのストリーミング元となるセンサー パスです。

ステップ 3 ルータからストリーミングされるテレメトリ データをサブスクライブします。サブスクリプションは、宛先グループをセンサーグループにバインドし、ストリーミング方式を設定します。ストリーミング方式は、パターン駆動型テレメトリまたはイベント駆動型テレメトリにすることができます。

例：

（注） イベント駆動型テレメトリの設定は、サンプル間隔が異なる点を除き、パターン駆動型テレメトリに似ています。サンプル間隔の値を 0（ゼロ）に設定すると、イベント駆動型テレメトリのサブスクリプションが設定され、間隔をゼロ以外の値に設定すると、パターン駆動型テレメトリのサブスクリプションが設定されます。

データ モデルを使用してサブスクリプションを作成する

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <telemetry-model-driven
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-telemetry-model-driven-cfg">
        <subscriptions>
          <subscription>
            <subscription-identifier>CPU-Utilization</subscription-identifier>
            <sensor-profiles>
              <sensor-profile>
                <sensorgroupid>Monitor-CPU</sensorgroupid>
                <sample-interval>30000</sample-interval>
              </sensor-profile>
            </sensor-profiles>
            <destination-profiles>
              <destination-profile>
                <destination-id>CPU-Health</destination-id>
              </destination-profile>
            </destination-profiles>
           <source-interface>Interface1</source-interface>
          </subscription>
        </subscriptions>
```

```
      </telemetry-model-driven>
    </config>
  </edit-config>
</rpc>
```

CLI を使用してサブスクリプションを作成する

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription CPU-Utilization
Router(config-model-driven-subs)#sensor-group-id Monitor-CPU sample-interval 30000
Router(config-model-driven-subs)#destination-id CPU-Health
Router(config-model-driven-subs)#source-interface Interface1
Router(config-model-driven-subs)#commit
```

ここで、

- `CPU-Utilization` はサブスクリプションの名前です
- `Monitor: CPU` は、センサーグループの名前です
- `CPU-Health` は、宛先グループの名前です
- `Interface1` は、テレメトリ セッションの確立に使用される送信元インターフェイスです。VRF と送信元インターフェイスの両方が設定されている場合、送信元インターフェイスは、宛先グループで指定されたものと同じ VRF にあることが必要です。
- `30000` は、ミリ秒単位のサンプル間隔です。サンプル間隔は、2 つのデータ ストリーム間の時間間隔です。この例では、サンプル間隔は 3 万ミリ秒（30 秒）です。

サブスクリプションの展開を確認する

ルータは、受信者にダイヤルアウトし、サブスクリプション内の各宛先とのセッションを確立します。セッションが確立されると、ルータはデータを受信者にストリーミングしてデータ レイクを作成します。

サブスクリプションの展開は、ルータ上で確認できます。

手順

ステップ 1 ルータで、モデル駆動型テレメトリの設定を表示します。

例：

```
Router#show running-config telemetry model-driven
telemetry model-driven
destination-group CPU-Health
address-family ipv4 172.0.0.0 port 57500
encoding self-describing-gpb
protocol tcp
!
sensor-group Monitor-CPU
sensor-path
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization
```

```
!
subscription CPU-Utilization
sensor-group-id Monitor-CPU sample-interval 30000
destination-id CPU-Health
!
```

ステップ 2 サブスクリプションの状態を確認します。`Active` 状態は、ルータがサブスクリプションに基づいて受信者にデータをストリーミングする準備が整っていることを示します。

例：

```
Router# show telemetry model-driven subscription CPU-Utilization

Subscription:  CPU-Utilization                      State: ACTIVE
-------------
  Sensor groups:
  Id                 Interval(ms)        State
  Monitor-CPU        30000               Resolved

  Destination Groups:
  Id                 Encoding            Transport   State    Port    IP
  CPU-Health         self-describing-gpb tcp         Active   57500   172.0.0.0
```

ルータは、サブスクリプションベースのテレメトリ セッションを使用して受信者にデータをストリーミングし、受信側にデータ レイクを作成します。

ネットワークの詳細な分析のためにテレメトリ データを操作する

データ レイクからのテレメトリ データの消費と分析を開始するには、オープンソースの収集スタックを使用できます。この使用例では、収集スタックの次のツールを使用します。

- Pipeline は、データを収集するために使用される軽量ツールです。Network Telemetry Pipeline は、Github からダウンロードできます。`pipeline.conf` ファイルを使用して、コレクタがルータと通信する方法と処理されたデータを送信する場所を定義します。
- Telegraph（プラグイン駆動型サーバ エージェント）および InfluxDB（時系列データベース（TSDB））は、可視化ツールによって取得されるテレメトリ データを保存します。InfluxDB は、Github からダウンロードできます。`metrics.json` ファイルを使用して、TSDB に含めるデータを定義します。
- Grafana は、ルータからストリーミングされたデータのグラフおよびカウンタを表示する可視化ツールです。

つまり、Pipeline は TCP および gRPC テレメトリ ストリームを受け入れてデータを変換し、そのデータを InfluxDB データベースにプッシュします。Grafana は、InfluxDB データベースからのデータを使用してダッシュボードおよびグラフを作成します。Pipeline と InfluxDB は、同じサーバ上でも異なるサーバ上でも実行できます。

ルータは 約 350 のカウンタのデータを 5 秒ごとにストリーミングし、Telegraf は Pipeline からの情報を 1 秒間隔で要求しているとします。CPU 使用率は、次を使用して 3 つのステージで分析されます。

- 最初の値を取得する単一のルータ
- 値の差異を特定し、パターンを把握する 2 台のルータ
- 証拠に基づく結論に達するための 5 台のルータ

これにより、インフラストラクチャ（この場合はCPU）の展開について、情報を得たうえでビジネス上の意思決定を行うことができます。

手順

ステップ 1 Pipeline を開始し、ルータのクレデンシャルを入力します。

（注） 宛先グループで指定する IP アドレスおよびポートは、Pipeline がリスニングしている IP アドレスおよびとポートと一致している必要があります。

例：

```
$ bin/pipeline -config pipeline.conf

Startup pipeline
Load config from [pipeline.conf], logging in [pipeline.log]

CRYPT Client [grpc_in_mymdtrouter], [http://172.0.0.0:5432]
 Enter username: <username>
 Enter password: <password>
Wait for ^C to shutdown
```

ステップ 2 CPU 使用率に関するメトリックを読み取るため、Telegraph 設定ファイルで次の値を追加します。

例：

```
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states.
  report_active = false
```

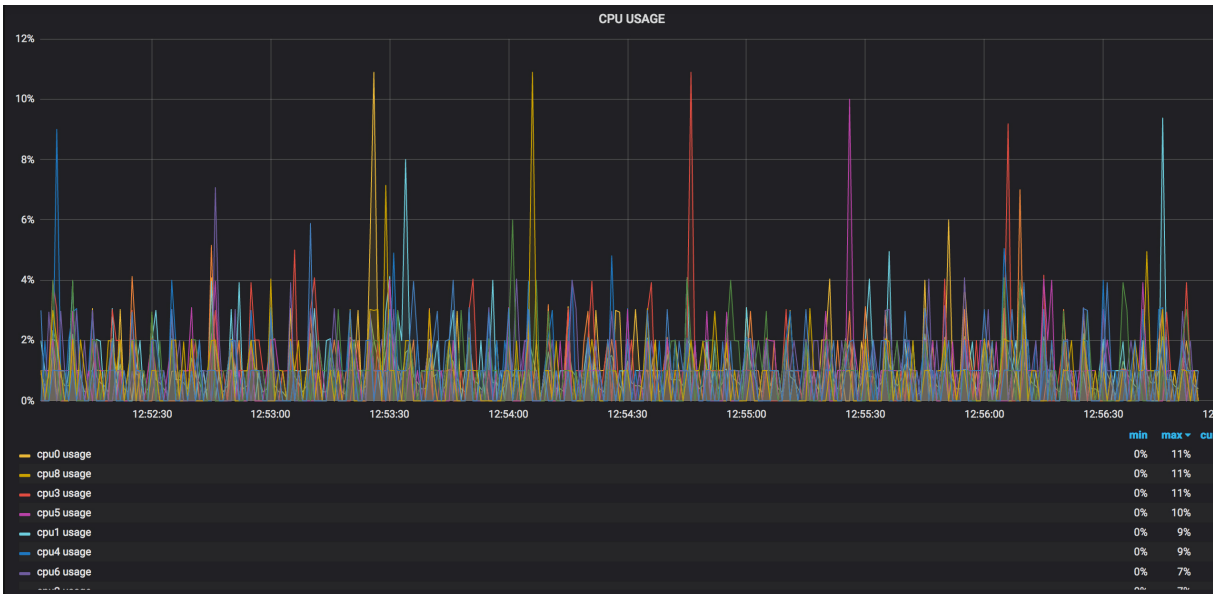
ステップ 3 Grafana を使用してダッシュボードを作成し、CPU 使用率に関するデータを可視化します。

1 台のルータ

ルータは 5 秒ごとにカウンタをプッシュします。

すべての CPU コアに負荷が均等に分散され、約 10% または 11% までのスパイクが発生しています。

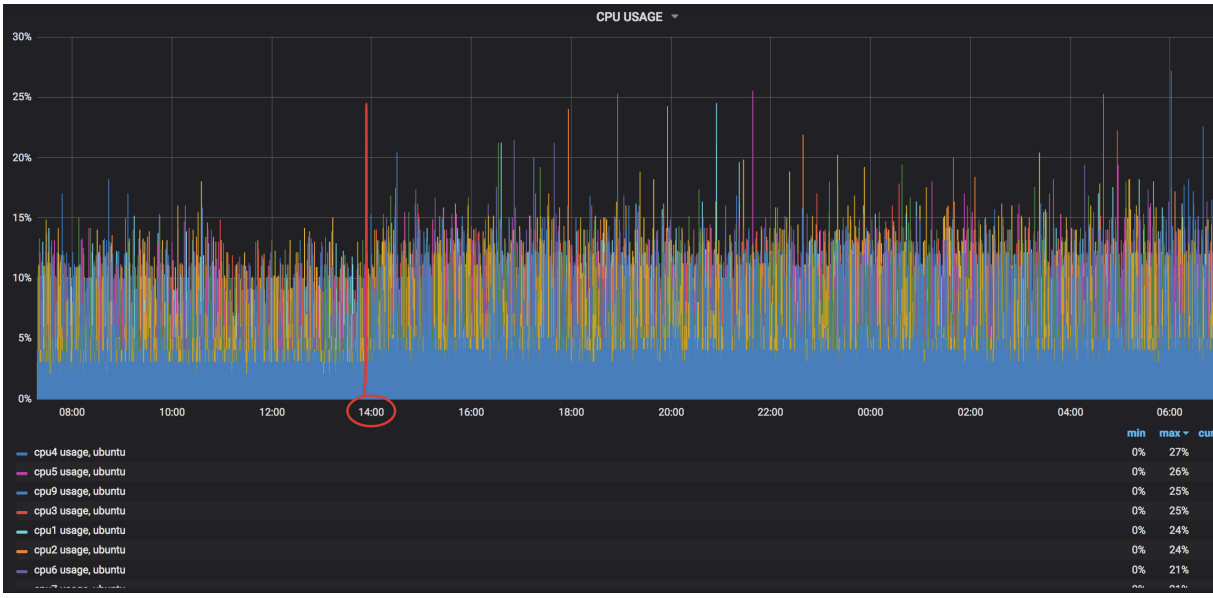
図 *2*：単一ルータの場合の *CPU* 使用率グラフ



2 台のルータ

2 台目のルータがタイムラインの `14:00` に追加され、スパイクが約 25% に増加していることを示しています。中間値は 15% です。

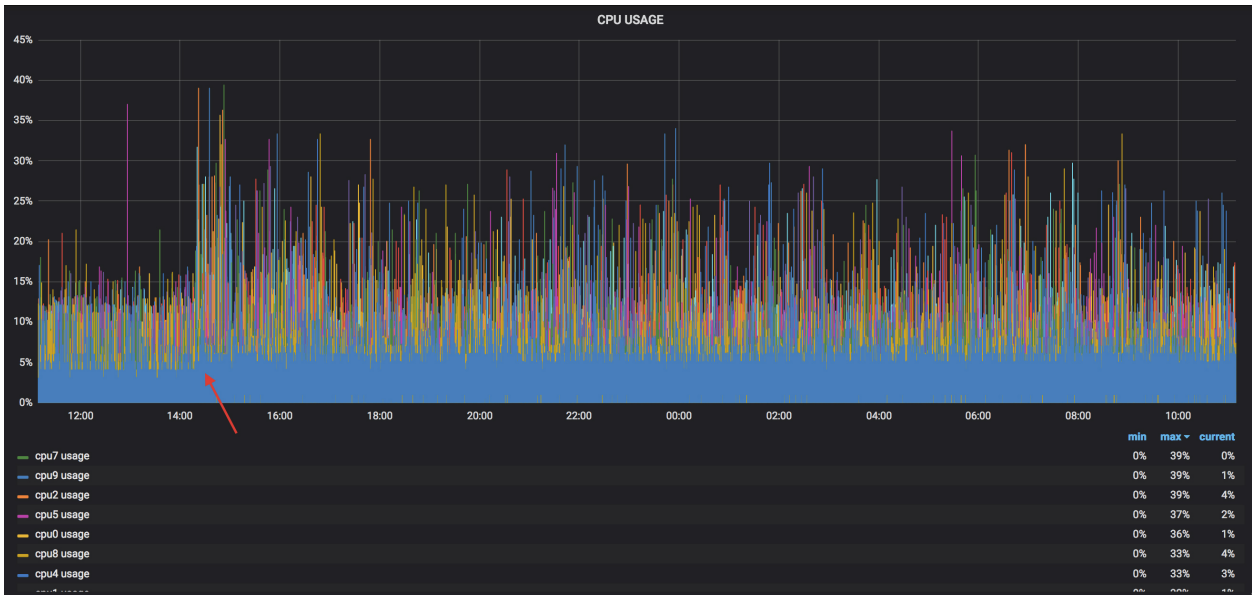
図 *3*：*2* 台のルータを使用した場合の *CPU* 使用率グラフ



5 台のルータ

5 台のルータが使用され、約 40% をピークとするスパイクが発生しています。中間値は 約 22 ～ 25% の範囲です。

図 *4 : 5* 台のルータを使用した場合の *CPU* 使用率グラフ



結論として、テレメトリ データは、プロセスがすべての CPU コアに対しほぼ均等に分散されていることを示しています。コアのサブセットには線形的な増加はありません。この分析は、ストリーミングするカウンタの数に基づいて CPU 使用率を計画するうえで役立ちます。