より高い TPS で最適な CPS パフォーマンスになるようにパラメータを調整する

目次

概要

このドキュメントは、トラフィックが多い時のパフォーマンスの問題を診断し、高い Transactions Per Second（TPS）で最適なパフォーマンスを得るために Cisco Policy Suite（CPS）パラメータを調整するのに役立ちます。

問題診断

1. consolidated-engine ログを分析し、「2001-DIAMETER_SUCCESS」以外の diameter 結果コードを探します。 例：
```
[root@pcrfclient01 broadhop]#zcat consolidated-engine_07Apr15_16_06_37.1.log.gz| grep
"Result-Code" | grep -v 2001|cut -c16-19|sort -u
3002
5002
5012
```
注: この出力には、3002-DIAMETER_UNABLE_TO_DELIVER、5002-DIAMETER_UNKNOWN_SESSION_ID、5012-DIAMETER_UNABLE_TO_COMPLY が示されています。diameter 結果コードの詳細は、RFC 3588 で確認できます。最適なパフォーマンスが得られるように設定されていない CPS の場合は、5012-DIAMETER_UNABLE_TO_COMPLY が頻出します。
2. consolidated-engine ログで、Diameter 結果コード 5012 の出現回数を確認してください。 例：
```
[root@pcrfclient01 broadhop]#zcat consolidated-engine_07Apr15_23_16_35.1.log.gz|
grep "Result-Code" | grep 5012|wc -l
6643
[root@pcrfclient01 broadhop]#zcat consolidated-engine_07Apr15_16_06_37.1.log.gz|
grep "Result-Code" | grep 5012|wc -l
627
[root@pcrfclient01 broadhop]#zcat consolidated-engine_07Apr15_16_26_37.1.log.gz|
grep "Result-Code" | grep 5012|wc -l
2218
[root@pcrfclient01 broadhop]#zcat consolidated-engine_07Apr15_16_46_35.1.log.gz|
grep "Result-Code" | grep 5012|wc -l
0
```
5012 の diameter 結果コードが高い TPS で頻出している場合は、この手順にある口グの追加検証に進みます。

3. consolidated-engine ログで、Policy and Charging Rules Function（PCRF）から結果コード 5012 と DiameterResponseMessage が送信される前に、「connection wait timeout after 0 ms」エラーが記録されていないか確認してください 。 例：

```
<snip>
INFO : (balance) Error found, rolling back transaction
ERROR  : (core) Error processing policy request: com.mongodb.DBPortPool$Connection
WaitTimeOut: Connection wait timeout after 0 ms
com.mongodb.DBPortPool.get(DBPortPool.java:222)
com.mongodb.DBTCPConnector$MyPort.get(DBTCPConnector.java:413)
com.mongodb.DBTCPConnector.innerCall(DBTCPConnector.java:238)
com.mongodb.DBTCPConnector.call(DBTCPConnector.java:216)
com.mongodb.DBApiLayer$MyCollection.__find(DBApiLayer.java:288)
com.mongodb.DBApiLayer$MyCollection.__find(DBApiLayer.java:273)
com.mongodb.DBCollection.findOne(DBCollection.java:728)
com.mongodb.DBCollection.findOne(DBCollection.java:708)
com.broadhop.balance.impl.dao.impl.MongoBalanceRepository$6.findOne(MongoBalance
Repository.java:375)
<snip>
```

注: CPS バージョン 5.5 以降で使用可能な top_qps.sh コマンドを使用して、問題が発生している最中の CPS システムの TPS を確認できます。

解決策

1. Policy Builder のスレッド生成の設定をデフォルトの 20 から 50 に変更します。 これを行うには、Policy Builder にログインし、[Reference Data] > [Systems] > [system-1] > [Plugin Configurations] >[Threading Configurations] を選択します。 デフォルトでは（スレッド生成の設定フィールドが空白の場合）、Policy Builder 設定の mongo 接続用のスレッド数は 20 です。低い TPS で実行する場合の要求量を処理できます。 TPS が増加すると、これらのスレッドがビジーになるため、要求を処理するには、より多くのスレッドが必要になります。スレッド数を 50 にすると、より多くのスレッドでより多くの要求を処理できます。これは、約 5000 の TPS を十分に処理できるスレッド数です。これらはポリシー エンジン スレッドであり、「rules」という名前で定義されます。この名前しか設定してはいけません。
2. /etc/broadhop/pcrf/qns.conf ファイルに「Dmongo.client.thread.maxWaitTime=5000」を追加します。 例：

```
cat /etc/broadhop/pcrf/qns.conf
QNS_OPTS="
-DbrokerUrl=failover:(tcp://lb01:61616,tcp://lb02:61616)?randomize=false
-DjmsFlowControlHost=lb02
-DjmsFlowControlPort=9045
-Dcc.collectd.ip.primary=pcrfclient01
-Dcc.collectd.port.primary=27017
-Dcc.collectd.ip.secondary=pcrfclient01
-Dcc.collectd.port.secondary=27017
-DudpPrefix=lb
-DudpStartPort=5001
-DudpEndPort=5003
-DqueueHeartbeatIntervalMs=25
-Dcom.broadhop.memcached.ip.local=lbvip02
-Dmongo.client.thread.maxWaitTime=5000
?
```

Dmongo.client.thread.maxWaitTime は、接続が可能になるまでスレッドが待機する時間（ミリ秒単位）です。 このパラメータを指定しない場合、デフォルト値の 0 ミリ秒と見なされます。 そのため、テストの TPS が高いとエラーが発生します。 /etc/broadhop/pcrf/qns.conf

にこのパラメータを追加すると、テストの TPS が高いときに新しいスレッドが mongo 接続を待機する時間が長くなります。QA が推奨する値は 2000 です。この値は高い TPS でテストされています。 TPS が 5000 より大きい場合は、この値を 5000 ms に設定すると、パフォーマンスを最適化できます。

3. /Etc/broadhop/pcrf/qns.conf に「-Dspr.mongo.socket.timeout=5000」を追加します。デフォルト値は 60000 ミリ秒（60 秒）です。 そのため、他のスレッドで使用可能になるには長い時間がかかります。すぐにタイムアウトになり、他のスレッドが速く処理できるように、推奨設定は 5000 ミリ秒（5 秒）になっています。
4. Policy Builder の [Connections Per Host] の値を、デフォルトの 5 から 20 に変更します。 これを行うには、Policy Builder にログインし、[Reference Data] > [Systems] > [system-1] > [Plugin Configurations] > [USuM Configuration] > [Connections Per Host] を選択します。 これは Quantum Quantum Network Suite（QNS）1 つあたりの mongo DB の接続数です。 QNS が 4 つの場合、合計接続数が 4*20=80 になることを意味します。これは、mongodb の頻繁なアップデートのために必要になります。 したがって、最適なパフォーマンスを得るために、QA の推奨に従って 20 に更新することを推奨します。また、[Db Read Preference] を [SecondaryPreferred] に設定します。これは、すべての QNS がセカンダリからデータを受信し、セカンダリ データベースがビジーの場合に限り、プライマリからデータを受信することを意味します。 こうすると、プライマリ データベースへの負荷が非常に少なくなるため、パフォーマンスの最適化に役立ちます。
5. 適切なルート ログ レベルをシステムに設定します。 過度に多いログは、QNS および LB レベルの処理を妨げます。このため、/etc/brodhop/logback.xml ファイルと /etc/broadhop/controlcenter/logback.xml ファイルのルート ログ レベルを、「warn」以上のレベルに設定することを推奨します。例：

```
[root@pcrfclient01 ~]#cat /etc/broadhop/logback.xml

<snip>
<!-- Configure default Loggers -->
<root level="warn">
<appender-ref ref="FILE" />
<appender-ref ref="SOCKET" />
</root>

</configuration>
```

また、これらのログ レベルも変更します。

```
[root@pcrfclient01 ~]#cat /etc/broadhop/logback.xml

<snip>
<!-- Configure default Loggers -->
<root level="warn">
<appender-ref ref="FILE" />
<appender-ref ref="SOCKET" />
</root>

</configuration>
```

例：

```
[root@pcrfclient01 ~]# cat /etc/broadhop/controlcenter/logback.xml

<snip>

<!-- Configure default Loggers -->
<root level="warn">
<appender-ref ref="FILE" />
</root>

</configuration>
```

これらの変更はすべての仮想マシンに複製する必要があります。 syncconfig.sh を実行してから restartall.sh を実行（または、stopall.sh を実行してから startall.sh を実行）して、これらの変更をすべて適用してください。

警告： これらの変更は、メンテナンスの時間帯にのみ実行してください。