CPSレプリカセットでアービターノードを管理する手順

内容

概要

このドキュメントでは、Cisco Policy Suite(CPS)レプリカセットでアービターノードを管理する手順について説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- Linux
- CPS
- MongoDB

注:CPS CLIへのrootアクセス権限が必要であることを推奨します。

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- CPS 20.2
- ユニファイドコンピューティングシステム(UCS)-B
- MongoDB v3.6.17およびv3.4.16

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

背景説明

CPSは、MongoDBを使用して基本データベース(DB)構造を構成します。ADMIN、Subscriber Profile Repository(SPR)、BALANCE、SESSION、REPORTING、およびAUDITなど、さまざまな目的に対応する複数のレプリカセットを備えています。

MongoDBのレプリカセットは、同じデータセットを維持する単一プロセスのグループです。レプリカセットは、冗長性と高可用性(HA)を提供します。異なるDBサーバ上に複数のデータのコピーを作成することで、ロードシェアの読み取り操作が可能になります。

状況によっては（プライマリとセカンダリがあるものの、コストの制約により別のセカンダリの追加が禁止されている場合など）、モノゴインスタンスをアービターとしてレプリカセットに追加し、選挙で投票することができます。1人のアービターの投票数は正確に1票です。デフォルトでは、アービターのプライオリティは0です。

アービターは、レプリカセットの一部であるが、データを保持しない（つまり、データの冗長性を提供しない）単一のインスタンスです。しかし、彼らは選挙に参加することができます。アービターはプライマリの選出に参加しますが、アービターはデータセットのコピーを持たず、プライマリになることはできません。

アービターのリソース要件は最小限で、専用ハードウェアは必要ありません。ネットワークを監視するだけのアプリケーションサーバまたはホストにアービターを導入できます。

アービターはデータを保存しませんが、アービターのmongodプロセスがレプリカセットに追加されるまで、アービターは他のmongodプロセスと同様に動作し、一連のデータファイルとフルサイズのジャーナルを使用して起動します。

次にレプリカセットの例を示します。 **set07**.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------------
----------------------------------------|
```

問題

アービターに問題がある、またはレプリカセットのアービターを変更する必要がある場合は、現在のアービターを削除し、新しいアービターをレプリカセットに追加する必要があります。

レプリカ・セット内のアービターを管理する手順

ステップ 1：CPSと新しいアービターでmongoシェルのバージョンを確認します。このコマンドは、レプリカ・セットおよび新しいアービター・ノード内のプライマリsessionmgrから実行します。

sessionmgrからの出力例：

```
[root@sessionmgr02 ~]# mongo --version
```

```
MongoDB shell version v3.6.17
```
Mongoシェルのバージョンがプライマリsessionmgrと新しいアービターの両方で同じ場合、または新しいアービターmongoシェルのバージョンが上位の場合は、ステップ6に進みます。

新しいアービターmongoシェルのバージョンが低い場合は、を設定する必要があります **featureCompatibilityVersion** 次の手順では、レプリカセットの管理データベースの値が小さくなります。

新しいarbiter mongoシェルのバージョンがCPS sessionmgrのバージョンよりも低い場合の例：

```
[root@pcrfclient02 ~]# mongo --version
MongoDB shell version v3.4.16
```
ステップ 2：レプリカセットのプライマリmongoインスタンスにログインします。

```
Command template:
#mongo --host <sessionmgrXX> --port <Replica Set port>

Sample command:
#mongo --host sessionmgr02 --port 27727
```
ステップ 3： 次のコマンドを実行して、現在の **featureCompatibilityVersion** レプリカ・セットの管理データベース内。

```
set07:PRIMARY> db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
{
"featureCompatibilityVersion" : {
"version" : "3.6"
},
"ok" : 1,
"operationTime" : Timestamp(1663914140, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1663914140, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
set07:PRIMARY>
```
ステップ 4：このコマンドを実行して、 setfeatureCompatibilityVersion レプリカ・セットの管理データベースに3.4として保存されます。

```
set07:PRIMARY> db.adminCommand( { setFeatureCompatibilityVersion: "3.4" } )
{ "ok" : 1 }
set07:PRIMARY>
```
ステップ 5： 次のコマンドを実行して、 **featureCompatibilityVersion** は、レプリカセットの管理データベースで3.4に変更されています。

```
set07:PRIMARY> db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
{ "featureCompatibilityVersion" : { "version" : "3.4" }, "ok" : 1 }
set07:PRIMARY>
```
手順 6：Cluster Managerにログインし、 **/var/qps/config/deploy/csv/AdditionalHosts.csv** 新しいアービターの詳細を含むファイル。

```
#vi /var/qps/config/deploy/csv/AdditionalHosts.csv

Provide new arbiter details in this format:

Host Alias IP Address
new-arbiter new-arbiter xx.xx.xx.xx
```

手順 7：CSV設定をインポートします。

```
#/var/qps/install/current/scripts/import/import_deploy.sh
```

ステップ 8：確認 **/etc/hosts** 新しいアービターの情報で更新されました。

```
#cat /etc/hosts | grep arbiter
```

ステップ 9：同期するには、次のコマンドを実行します **/etc/hosts**.

```
#/var/qps/bin/update/synchosts.sh

Syncing to following QNS Servers:
lb01 lb02 sessionmgr01 sessionmgr02 qns01 qns02 pcrfclient01 pcrfclient02
Do you want to Proceed? (y/n):y
lb01
lb02
sessionmgr01
sessionmgr02
qns01
qns02
pcrfclient01
pcrfclient02
```

ステップ 10：pcrfclient VMでmon_dbスクリプトが停止していることを確認します。

```
#monsum | grep mon_db_for
```

停止した場合の出力を次に示します。

```
mon_db_for_lb_failover Not monitored Program
mon_db_for_callmodel Not monitored Program
```

停止していない場合の出力を次に示します。

```
mon_db_for_lb_failover OK Program
mon_db_for_callmodel OK Program
```

> **注**:mon_dbスクリプトが停止していない場合は、それぞれのpcffclient VMでこれらのコマンドを実行して、手動でスクリプトを停止します。

```
#monit stop mon_db_for_lb_failover
#monit stop mon_db_for_callmodel
```

ステップ 11このコマンドをpcrfclient01から実行して、現在のアービタをレプリカセットから削除します（この手順ではset07が例です）。

```
#build_set.sh --session --remove-members --setname set07
```

```
Please enter the member details which you going to remove from the replica-set
Member:Port --------> arbitervip:27727
arbitervip:27727
Do you really want to remove [yes(y)/no(n)]: y
```

ステップ 12Cluster Managerからこのコマンドを実行し、アービターがWLCから削除されたかどうかを **set07**の出力 **set07** 現在のアービターを含めることはできません。

```
#diagnostics.sh --get_replica_status

Expected output:
----------|
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec -|
| Member-2 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- -|
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
```

ステップ 13Cisco IOSソフトウェアリリース12.1 **mongoConfig.cfg** 変更されたレプリカ・セットに適切なアービターを含むファイル。現在のアービター(ARBITER=arbiter)を新しいアービター(ARBITER=new-arbiter)に置き換えます。Cluster Managerからこのコマンドを実行します。

```
#vi /etc/broadhop/mongoConfig.cfg
```

Current configuration :

```
[SESSION-SET2]
SETNAME=set07
OPLOG_SIZE=5120
ARBITER=arbitervip:27727
ARBITER_DATA_PATH=/var/data/sessions.7
MEMBER1=sessionmgr02:27727
MEMBER2=sessionmgr01:27727
DATA_PATH=/var/data/sessions.1/2
[SESSION-SET2-END]
```

必要な設定：

```
[SESSION-SET2]
SETNAME=set07
OPLOG_SIZE=5120
ARBITER=new-arbiter:27727
ARBITER_DATA_PATH=/var/data/sessions.7
MEMBER1=sessionmgr02:27727
MEMBER2=sessionmgr01:27727
DATA_PATH=/var/data/sessions.1/2
[SESSION-SET2-END]
```

ステップ 14：更新されたファイルをコピーする **mongoConfig.cfg** すべてのVMにファイルを保存します。クラスタマネージャからこのコマンドを実行します。

```
#copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg
```

ステップ 15：set07に新しいアービタメンバを追加します。Cluster Managerから、次を実行します。 **/var/qps/install/current/scripts/build/build_etc.sh** コマンドを発行して、 **/etc/directory**.

ステップ 16：新しいアービターメンバーがレプリカセットに追加されたことを確認します。 **build_etc.sh** 次に、AIDOサーバが新しいアービターを使用してレプリカセットを作成または更新するのを待つ必要があります。

```
#diagnostics.sh --get_replica_status

Expected Output:

| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : xx.xx.xx.xx - ARBITER - new-arbiter - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-----------------------------------------|
```

> **注**：新しい監視メンバが追加されていない場合は、次の手順に進みます。それ以外の場合は、ステップ18に進みます。

ステップ 17：新しい監視メンバを強制的に追加するには、クラスタマネージャからこのコマンドを実行します。

```
#build_set.sh --DB_NAME --add-members --setname Setxxx --force
```

ステップ 18：アービターポートがまだアップしていない場合は、新しいアービターノードからこのコマンドを実行して同じポートを起動します。

```
Command syntax:
#/etc/init.d/sessionmgr-XXXXX start

Sample command:
#/etc/init.d/sessionmgr-27727 start
```

ステップ 19：新しいアービターが正常に追加されたことを確認します。

```
#diagnostics.sh --get_replica_status
```

ステップ 20：Cluster Managerからこのコマンドを実行し、DBプライオリティを適宜更新します。

```
# cd /var/qps/bin/support/mongo/
# ./set_priority.sh --db session
# ./set_priority.sh --db spr
# ./set_priority.sh --db admin
# ./set_priority.sh --db balance
# ./set_priority.sh --db audit
# ./set_priority.sh --db report
```

ステップ 21：Cluster Managerからこのコマンドを実行して、レプリカセットの変更を確認します。

```
#diagnostics.sh --get_replica_status
```

```
Expected Output:

| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : xx.xx.xx.xx - ARBITER - new-arbiter - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|----------------------------------------|
```

ステップ 22：pcrfclient VMでmon_dbスクリプトが復元されていることを確認します。そうでない場合は、手動で起動する必要があります。

```
#monsum | grep mon_db_for
```

mon_dbスクリプトを有効にするには、すべてのpcrfclient VMにログインし、次のコマンドを実行します。

```
# monit start mon_db_for_lb_failover
# monit start mon_db_for_callmodel
```

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。