PostgreSQL on CLUSTERPRO for Linux HOWTO

1　はじめに

この文章は、CLUSTERPRO for Linux上でPostgreSQLを動作させる際に参考となる情報を記述したものです。PostgreSQLを片方向および双方向スタンバイで運用するための設定方法や注意点を述べます。

この文章を書くにあたって次のディストリビューションと同梱されているPostgreSQLを使用しました。このほかのバージョンのディストリビューションやPostgreSQLを別途インストールした場合でも、いくつかの設定項目の読み替えでクラスタ化できると思われます。

・RedHat Enterprise Linux AS(v.2.1) + PostgreSQL(7.2.1-3)

・CLUSTERPRO SE for Linux Ver2.1

2　責任範囲

この文章は、PostgreSQLをクラスタ環境で使用するための注意点や設定例を参考情報として示すものであり、これらの動作保証を行うものではありません。

3　事前準備

PostgreSQLを両サーバにインストールし、PostgreSQLのユーザを作成しておいてください。

（詳しくは、PostgreSQLのドキュメントをご覧ください）

Linuxのインストール時にディストリビューションのインストール画面でPostgreSQLのインストールを指定している場合は、通常、特に準備を行う必要はありません。ただし、ディストリビューションによっては、PostgreSQLインストール時にデータベースサーバを自動起動するものもあります。その場合は、自動起動の設定を解除してください。自動起動の設定解除は以下のコマンドを実行し、リブートしてください。

chkconfig --level *xx* postgresql off

（*xx*：ランレベル）

4　片方向スタンバイ用PostgreSQL環境の構築手順

以下にPostgreSQLを片方向スタンバイで運用する際の構築方法を説明します。双方向スタンバイにする際の片側のシステムも同様の方法で構築します。双方向スタンバイのもう片方のシステムの構築は、後述します。

4.1　前提環境

以下のような片方向スタンバイ環境を想定し、説明します。

クラスタサーバ環境

	サーバ１(運用系)	サーバ２(待機系)
実IPアドレス	10.0.0.1	10.0.0.2
切替パーティション	/mnt/PostgreSQL1	

PostgreSQLのクライアント環境

	クライアント１	クライアント２
実IPアドレス	10.0.1.1	10.0.1.2

グループ情報

	グループ１
フローティングIPアドレス	10.0.0.11
切替パーティション	/mnt/PostgreSQL1

データベース環境

データベース名	DATABASE1
データファイルの場所	/mnt/PostgreSQL1/data
TCP/IP接続時のポート番号	5432

4.2　データベース領域の初期化

下記の手順で、データベース領域を初期化します。

(1)グループ１をサーバ１で起動する。

(2)PostgreSQLユーザでサーバ１のLinuxにログインする。

(3)initdb -D /mnt/PostgreSQL1/data

注意)本段階では、グループ１のスクリプトでPostgreSQLの起動・終了用の記述はありません。既定値のスクリプトのままでよいです。スクリプトの記述は、4.6の段階で行います。

注意)/mnt/PostgreSQL1/dataは、initdb実行時に作成されるため、事前に作成しないでください。作成

されている場合はinitdb実行時エラーになります。

4.3　データベースの作成

下記の手順で、PostgreSQLのデータベースを作成します。

まず(1)でPostgreSQL(データベースサーバ)を起動して、(2)でデータベースの作成となります。

(1)pg_ctl start -D /mnt/PostgreSQL1/data -l /dev/null -o '-i -p 5432'

(2)createdb DATABASE1 -p 5432

4.4　データベースのユーザ作成

下記の手順で、PostgreSQLのデータベースに接続するためのユーザを作成します。

createuser -P ユーザ名 -p 5432

createuserでユーザを作成する際に、データベース作成権限などを聞いてきますので、必要に応じて答えてください。

4.5　クライアント認証

クライアントからのデータベース接続制御は、pg_hba.confファイルで設定します。pg_hba.confファイルは、PGDATA環境変数で示すディレクトリの下にあります。(例：/var/lib/pgsql/data)

記述形式

host データベース名 IPアドレス ネットマスク 認証方法

データベース名

特定のデータベースに対する場合、そのデータベース名を記述

全てのデータベースに対する場合、allと記述

認証方法

trust 全てのユーザに対して無条件に可

password パスワードを要求

クライアント1からDATABASE1に接続するユーザを全て可する場合

host DATABASE1 10.0.1.1 255.255.255.255 trust

クライアント2から全てのデータベースに接続するユーザからパスワードを要求する場合

host all 10.0.1.2 255.255.255.255 password

4.6　スクリプトの作成

グループ 1 にPostgreSQLの起動・終了スクリプトを作成します。

以下にそれぞれのサンプルスクリプトを示します。

start.bat

```
#! /bin/sh
#**************************************
#*              START.BAT              *
#**************************************

if [ "$ARMS_EVENT" = "START" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog normal1
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog normal2
                else
                        armlog on_other1
                fi
                armlog "PostgreSQL start"
```

```
                su - postgres -c "pg_ctl start -D /mnt/PostgreSQL1/data -l /dev/null -o '-i -p
5432'"
        else
                armlog "ERROR_DISK from START"
        fi
elif [ "$ARMS_EVENT" = "RECOVER" ]
then
        armlog recover
elif [ "$ARMS_EVENT" = "FAILOVER" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog failover1
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog failover2
                else
                        armlog on_ohter2
                fi
                armlog "PostgreSQL start"
                su - postgres -c "pg_ctl start -D /mnt/PostgreSQL1/data -l /dev/null -o '-i -p
5432'"
        else
                armlog "ERROR_DISK from FAILOVER"
        fi
else
        armlog no_arm
fi
armlog exit
```

stop.bat

```
#! /bin/sh
#**************************************
#*               STOP.BAT               *
#**************************************

arm_rel_path() {
        while [ "$1" != "" ]
        do
                armrelpath $1 > /dev/null 2>&1
                relret=$?

                if [ "$relret" = "0" ]
                then
                        armlog "KILL NO PROCESS"
                elif [ "$relret" = "1" ]
                then
                        armlog "KILL SOME PROCESS"
                else
                        armlog "ARMRELPATH ERROR"
                fi

                shift
        done
}

arm_rel_mntpoint() {
        mntpoint=`armlsmnt -l $ARMS_RESOURCELIST`
        mntret=$?

        if [ "$mntret" = "0" ]
        then
                if [ "$mntpoint" != "" ]
                then
                        arm_rel_path $mntpoint
                else
                        armlog "NO MOUNT POINT"
                fi
        else
                armlog "ARMLSMNT ERROR"
        fi
```

```
}

if [ "$ARMS_EVENT" = "START" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog "NORMAL1"
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog "NORMAL2"
                else
                        armlog "ON_OTHER1"
                fi
                armlog "PostgreSQL stop"
                su - postgres -c 'pg_ctl stop -D /mnt/PostgreSQL1/data -m fast'
                arm_rel_mntpoint
        else
                armlog "ERROR_DISK from START"
        fi
elif [ "$ARMS_EVENT" = "FAILOVER" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog "FAILOVER1"
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog "FAILOVER2"
                else
                        armlog "ON_OTHER2"
                fi
                armlog "PostgreSQL stop"
                su - postgres -c 'pg_ctl stop -D /mnt/PostgreSQL1/data -m fast'
                arm_rel_mntpoint
        else
                armlog "ERROR_DISK from FAILOVER"
        fi
else
        armlog "NO_ARM"
fi
armlog "EXIT"
exit 0
```

5　双方向スタンバイ用PostgreSQL環境の構築手順

以下にPostgreSQLを双方向スタンバイで運用する際の構築方法を説明します。

5.1　前提環境

以下のような双方向スタンバイ環境を想定し、説明します。

クラスタサーバ環境

	サーバ１(運用系)	サーバ２(待機系)
実IPアドレス	10.0.0.1	10.0.0.2
切替パーティション	/mnt/PostgreSQL1 , /mnt/PostgreSQL2	

PostgreSQLのクライアント環境

	クライアント１	クライアント２
実IPアドレス	10.0.1.1	10.0.1.2

グループ情報

	グループ１	グループ２
フローティングIPアドレス	10.0.0.11	10.0.0.12
切替パーティション	/mnt/PostgreSQL1	/mnt/PostgreSQL2

データベース環境

データベース名	DATABASE1	DATABASE2
データファイルの場所	/mnt/PostgreSQL1/data	/mnt/PostgreSQL2/data
TCP/IP接続時のポート番号	5432	5433

5.2　双方向スタンバイの一方のデータベース環境の作成

双方向スタンバイの場合、２つのデータベース環境を作成する必要があります。その際、一方は、片方向スタンバイ時のデータベース環境の作成と同じ方法で作成します。一方のデータベース環境の作成が完了したら、以下の手順で、もう一方のデータベース環境を作成します。

5.3　データベース領域の初期化

下記の手順で、データベース領域を初期化します。

(1)グループ２をサーバ１で起動する。

(2)PostgreSQLユーザでサーバ１のLinuxにログインする。

(3)initdb -D /mnt/PostgreSQL2/data

5.4　データベースの作成

下記の手順で、PostgreSQLのデータベースを作成します。

(1)pg_ctl start -D /mnt/PostgreSQL2/data -l /dev/null -o '-i -p 5433'

(2)ceratedb DATABASE2 -p 5433

5.5　データベースのユーザ作成

下記の手順で、PostgreSQLのデータベースに接続するためのユーザを作成します。

createuser -P ユーザ名 -p 5433

createuserでユーザを作成する際に、データベース作成権限などを聞いてきますので、必要に応じて答えてください。

5.6　スクリプトの作成

グループ２にPostgreSQLの起動・終了スクリプトを作成します。
以下にそれぞれのサンプルスクリプトを示します。

start.bat

```
#! /bin/sh
#************************************
```

```
#*              START.BAT               *
#**************************************

if [ "$ARMS_EVENT" = "START" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog normal1
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog normal2
                else
                        armlog on_other1
                fi
                armlog "PostgreSQL start"
                su - postgres -c "pg_ctl start -D /mnt/PostgreSQL2/data -l /dev/null -o '-i -p
5433'"
        else
                armlog "ERROR_DISK from START"
        fi
elif [ "$ARMS_EVENT" = "RECOVER" ]
then
        armlog recover
elif [ "$ARMS_EVENT" = "FAILOVER" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog failover1
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog failover2
                else
                        armlog on_ohter2
                fi
                armlog "PostgreSQL start"
        su - postgres -c "pg_ctl start -D /mnt/PostgreSQL2/data -l /dev/null -o '-i -p 5433'"
        else
                armlog "ERROR_DISK from FAILOVER"
        fi
else
        armlog no_arm
fi
armlog exit
```

stop.bat

```
#! /bin/sh
#**************************************
#*               STOP.BAT              *
#**************************************

arm_rel_path() {
        while [ "$1" != "" ]
        do
                armrelpath $1 > /dev/null 2>&1
                relret=$?

                if [ "$relret" = "0" ]
                then
                        armlog "KILL NO PROCESS"
                elif [ "$relret" = "1" ]
                then
                        armlog "KILL SOME PROCESS"
                else
                        armlog "ARMRELPATH ERROR"
                fi

                shift
        done
}

arm_rel_mntpoint() {
```

```
        mntpoint=`armlsmnt -l $ARMS_RESOURCELIST`
        mntret=$?

        if [ "$mntret" = "0" ]
        then
                if [ "$mntpoint" != "" ]
                then
                        arm_rel_path $mntpoint
                else
                        armlog "NO MOUNT POINT"
                fi
        else
                armlog "ARMLSMNT ERROR"
        fi
}

if [ "$ARMS_EVENT" = "START" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog "NORMAL1"
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog "NORMAL2"
                else
                        armlog "ON_OTHER1"
                fi
                armlog "PostgreSQL stop"
                su - postgres -c 'pg_ctl stop -D /mnt/PostgreSQL2/data -m fast'
                arm_rel_mntpoint
        else
                armlog "ERROR_DISK from START"
        fi
elif [ "$ARMS_EVENT" = "FAILOVER" ]
then
        if [ "$ARMS_DISK" = "SUCCESS" ]
        then
                armlog "FAILOVER1"
                if [ "$ARMS_SERVER" = "HOME" ]
                then
                        armlog "FAILOVER2"
                else
                        armlog "ON_OTHER2"
                fi
                armlog "PostgreSQL stop"
                su - postgres -c 'pg_ctl stop -D /mnt/PostgreSQL2/data -m fast'
                arm_rel_mntpoint
        else
                armlog "ERROR_DISK from FAILOVER"
        fi
else
        armlog "NO_ARM"
fi
armlog "EXIT"
exit 0
```