コードレビュー Vol.19

CODE Review

USP 研究所技術研究員
written by 大内智明

更新したファイルを相手サーバに配信する方法について説明します。

定期的に新しいデータを配信

今回は前回に引き続いて、店舗や商品といったマスタ関連のファイルが、どのように管理されて、各業務サーバに配信されているのかを説明していきます（図1）。

マスタ関連のファイルは、一般的に種類も多く、サイズも大小様々なものがあります。そこで、毎回全マスタの配信を行わないようにするために、必要なファイルだけを配信する仕組みにしています。

図1　定期的に新しいデータを配信

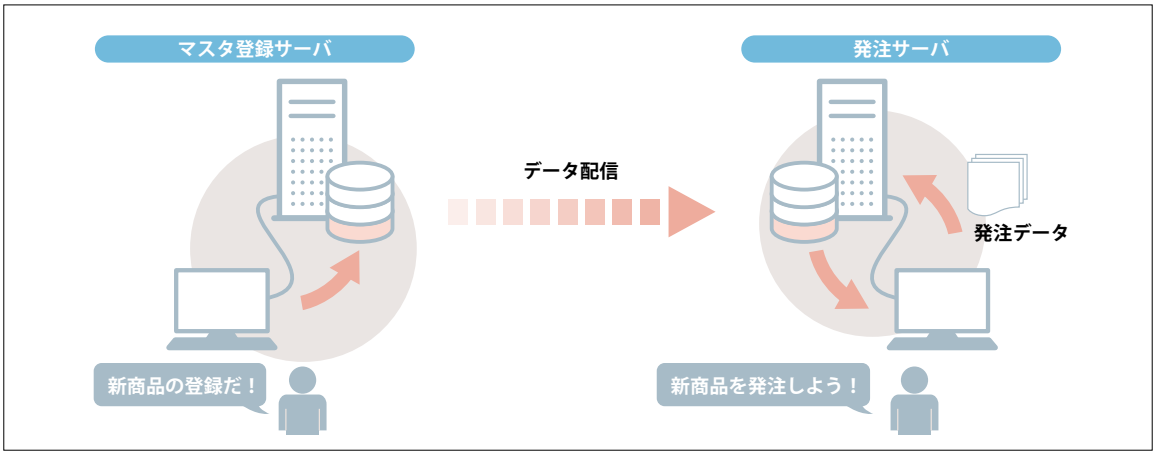

図2　ファイルの同期

最初の同期処理で、配信元と配信先でファイルの内容が一致している。
配信元のファイルに変更が発生すると、次の同期処理で差分の発生した配信先のファイルだけが、送信元のファイルと一致します。

【補足】※ 業務サーバ間でよく使用するファイル配信の方法には、rsync ／ scp ／ sftp があります。その中で rsync を選んだ理由は、rsync だけ差分が発生したファイルを更新（配信）することができます。マスタ関連のファイルは、種類が多くサイズも大きいが、中には殆ど更新がかからないファイルも多くあるため、日中定期的に更新をかける場合には、特に差分の発生した分だけ更新する rsync が有効となります。

技術的な概要（サーバ間でデータ配信を行う）

[1] 配信用定義ファイルを作成。

まず最初に配信元ディレクトリ／配信ファイル／配信先（ホスト名）の配信定義リストを作成することで、ディレクトリ単位または、ファイル単位の配信ができます。

[2] 配信用定義リストに従い、配信処理を実行

配信処理は、rsync（同期コマンド）※を使用して、配信定義用リストに記載した単位で順次処理を行っていきます。その際にリストで記載した単位の中で、ファイルに差分（図2）があれば配信（ファイル更新）を行います。

リスト　ファイルを配信する処理

```
 1 #!/usr/local/bin/bash -xv
 2 # システム名       :USPシステム
 3 # 業務名           :トラン系ＬＶ３の配信
 4
 5 # 走行ログの記録
 6 logd="${HOME}/LOG"
 7 logf="${logd}/LOG.$(basename $0).$(date +%Y%m%d)_$(date +%H%M%S)_$$"  # ログファイル名
 8 echo  "${logf}" &> /dev/null
 9 exec 2> ${logf}
10
11 #-------------------------------------------------------------------------------
12 # 変数の定義
13 # 引数設定項目のセット
14 sday=$1                                          # 処理日付
15 myjob=$2                                         # 配信元ＪＯＢグループ
16 hgrp=$3                                         # 配信処理グループ
17
18 < 中略 >
19
20 # エラー時の終了処理定義
21 ERROR_EXIT(){
22   touch ${semd}/$(basename $0)_${myjob}_${hgrp}.${hostname}.ERROR.${sday}
23   echo "${hostname} $(basename $0)_${myjob}_${hgrp}_${sday} ERROR $(date +%Y%m%d%H%M%S) ${logf}" >> ${logd}/UPCNT
24   exit 1
25 }
26
27 ////////////////////////////////////////////////////////////////////////////
28 # 処理部
29 ////////////////////////////////////////////////////////////////////////////
30 # 自サーバのJOBグループ
31 echo ${myjob} > $tmp-myjob
32 # 配信処理グループ
33 echo ${hgrp}  > $tmp-hgrp
34
35 # 正マシンのみ処理
36 if [ "$(msctrl -host ${hostname} -job ${myjob} -print msflg)" = "M" ] ; then
37
38   touch ${tmp}-sem
39   ////////////////////////////////////////////////////////////////////////////
40   # 配信
41   ////////////////////////////////////////////////////////////////////////////
42
43   # 自サーバのJOBグループ
44   msctrl -ctrl C -host ${hostname} -print job |
45   LANG=c sort -u                                > $tmp-myalljob
46   [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
47
```

ユニケージでは、サーバ構成が正副2台となっている。配信元ファイルは、必ず正サーバから配信するようにする。

msctrl はユニケージコマンド
サーバの正副、役割、ホスト名などの情報が取得できる。

❶

```
48  # 配信定義の取得
49  # 1:配信元JOBGROUP 2:配信処理グループ      3:配信親ディレクトリ 4:配信ファイル/ディレクトリ
50  # 5:配信先JOBGROUP 6:配信データ不在時動作 7:配信停止フラグ      8:削除同期フラグ
51  if ulock --invalid=300 ${fmtd}/DISTRILIST.LV3TRN.LOCK ; then
52     cat ${fmtd}/DISTRILIST.LV3TRN                    |
53     # 有効行で配信先JOBGROUP設定あり、配信停止フラグ="0"抽出
54     gawk '$1!~/^#/&&NF>=8&&$5!="_"&&$7=="0"'         |
55     # ディレクトリの末尾の"/"はとる
56     sed 's/\/ / /g'                                  |
57     # 自サーバの配信対象抽出
58     cjoin0 key=1 $tmp-myjob                          |
59     # 配信処理グループの抽出
60     cjoin0 key=2 $tmp-hgrp                           |
61     self 3 4 5 6 8                                   |
62     LANG=c sort                                      |
63     getlast key=1/3                                  > ${tmp}-dstlist
64     [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || { rm ${fmtd}/DISTRILIST.LV3TBL.LOCK ; ERROR_EXIT ; }
65     # 1:配信親ディレクトリ    2:配信ファイル/ディレクトリ 3:配信先JOBGROUP
66     # 4:配信データ不在時動作 5:削除同期フラグ
67     rm ${fmtd}/DISTRILIST.LV3TRN.LOCK
68  fi
69
70  : > $tmp-dstlist.host
71  self 1/5 $tmp-dstlist |
72  # 1:配信親ディレクトリ 2:配信ファイル/ディレクトリ 3:配信先JOBGROUP 4:配信データ不在時動作 5:削除同期フラグ
73  while read pdir data job noexist delete ; do
74     # 自サーバに存在するJOBGROUPへは配信しない                                                    ❶
75     awk '$1=="'${job}'"' $tmp-myalljob                 > $tmp-myalljob.exist
76     [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
77     [ -s $tmp-myalljob.exist ] && continue
78
79     # 配信先サーバ名の取得、リストへ追記
80     for host in $(msctrl -ctrl C -job ${job} -print host) ; do
81        echo "${pdir} ${data} ${host} ${noexist} ${delete}" >> $tmp-dstlist.host
82        [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
83        # 1:配信親ディレクトリ    2:配信ファイル/ディレクトリ 3:配信先業務ホスト名
84        # 4:配信データ不在時動作 5:削除同期フラグ
85     done
86  : ;
87  done
88  [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
89
90  # 配信先のユニーク化
91  LANG=c sort -u < $tmp-dstlist.host   |
92  # 1:配信親ディレクトリ    2:配信ファイル/ディレクトリ 3:配信先業務ホスト名
93  # 4:配信データ不在時動作 5:削除同期フラグ
94  getlast key=1/3                      > $tmp-dstlist.host_uniq
95  [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
96
97  # ホスト単位のrsyncログ
98  for host in $(cat $tmp-dstlist.host_uniq | self 3 | LANG=C sort -u);do
99      # 夜間配信のログは追記する
100     touch ${rsync_log}_${host}
101 done
102
103 #////////////////////////////////////////////////////////////////////////
104 # 配信 LV3
105 #////////////////////////////////////////////////////////////////////////
```

```
106  # 配信処理
107  # 1:配信親ディレクトリ   2:配信ファイル/ディレクトリ 3:配信先業務ホスト名
108  # 4:配信データ不在時動作 5:削除同期フラグ
109  while read pdir data host noexist delete ; do
110     # 配信先ホスト名を取得して配信処理
111
112     # rsync ログへ出力
113     echo "$(date +%Y/%m/%d_%H:%M:%S) ${pdir} ${data} START....." >> ${rsync_log}_${host}
114
115     # 配信ファイル/ディレクトリが存在しない場合
116     # 配信データ不在時動作="0"
117     if [ "${noexist}" = "0" ] ; then
118        [ ! -e ${pdir}/${data} ] &&                    \
119          { ERROR_EXIT ${pdir}/${data}が存在しません ;   \
120            continue ; }
121     # 配信データ不在時動作="1"
122     elif [ "${noexist}" = "1" ] ; then
123        if [ ! -e ${pdir}/${data} ] ; then
124          # rsync ログへ出力
125          echo "$(date +%Y/%m/%d_%H:%M:%S) ${pdir} ${data} END" >> ${rsync_log}_${host}
126          echo " "  >> ${rsync_log}_${host}
127
128          continue
129        fi
130     else
131           ERROR_EXIT ${fmtd}/DISTRILIST.LV3TRNの定義エラー_${pdir}/${data}
132     fi
133
134     # 配信先親ディレクトリの保障
135     ssh ${host} mkdir -p ${lv3d}/${pdir} < /dev/null
136     [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || { ERROR_EXIT ${pdir}/${data}の配信先ディレクトリ作成エラー ;
        continue ; }
137
138     # ディレクトリ/ファイルの配信(同期)を排他で実施
139     if ssh ${host} ${toold}/ulock --invalid=300 ${pdir}/${data}.gz.LOCK < /dev/null ; then
140        if [ "${delete}" = "0" ] ; then
141          # 配信(完全同期) deleteあり
142          # rsync ログへ出力
143          rsync -e 'ssh -c arcfour' -avz --delete --progress \
144                    ${pdir}/${data} ${host}:${pdir} >> ${rsync_log}_${host}  < /dev/null
145          [ $? -eq 0 ] || ERROR_EXIT ${pdir}/${data}の配信エラー
146        elif [ "${delete}" = "1" ] ; then
147          # 配信(存在するもののみ同期) deleteなし
148          # rsync ログへ出力
149          rsync -e 'ssh -c arcfour' -avz          --progress \
150                   ${pdir}/${data} ${host}:${pdir} >> ${rsync_log}_${host}  < /dev/null
151          [ $? -eq 0 ] || ERROR_EXIT ${pdir}/${data}の配信エラー
152        else
153          ERROR_EXIT ${fmtd}/DISTRILIST.LV3TRNの定義エラー_${pdir}/${data}
154        fi
155
156        # ロック解除
157        ssh ${host} rm ${pdir}/${data}.gz.LOCK < /dev/null
158        [ $? -eq 0 ] || ERROR_EXIT ${pdir}/${data}の配信先ロック解除エラー
159     fi
160
161     # rsync ログへ出力
162     echo "$(date +%Y/%m/%d_%H:%M:%S) ${pdir} ${data} END" >> ${rsync_log}_${host}
```

```
163       echo " "  >> ${rsync_log}_${host}
164
165    : ;
166    done < $tmp-dstlist.host_uniq
167
168    # 配信先別配信終了セマフォを作成する
169    self 3 $tmp-dstlist.host_uniq |
170    # 1:配信先業務ホスト名
171    LANG=c sort -u                |
172    while read host ; do
173       [ -e  ${semd}/$(basename $0)_${myjob}_${hgrp}_${host}.${hostname}.ERROR.${sday} ] && continue
174
175       # 配信終了セマフォを作成
176       touch ${semd}/$(basename $0)_${myjob}_${hgrp}_${host}.${hostname}.END.${sday}
177    done
178 fi
179
180 #////////////////////////////////////////////////////////////////////////////
181 # 終了処理
182 #////////////////////////////////////////////////////////////////////////////
183 < 中略 >
```

❷

❸

画面1　ulock

排他制御を行う

```
if ulock lock; then
  #
  # 読み書きなどの処理
  #
  rm -f lock
fi
```

排他区間

lockファイルを作成してからlockファイルを削除する間が、排他区間となる。この排他区間は、終了するまで他プロセスが同じ処理をすることができない。

画面2　cjoin0

cjoin0 key=<n> <master> <transation>

テキストファイル<tran>の"key=<n>"で指定したキーフィールドがマスターファイル<master>の第1フィールド(キーフィールド)とマッチした行のみを<tran>から抽出して出力。

```
$ cat master
0000003 杉山______ 26 F
0000005 崎村______ 50 F
0000007 梶川______ 42 F
$ cat tran
0000005 82 79 16 21 80
0000001 46 39 8  5  21
0000004 58 71 20 10 6
0000009 60 89 33 18 6
0000003 30 50 71 36 30
0000007 50 2  33 15 62
$ cjoin0 key=1 master tran > ok
$ cat ok
0000005 82 79 16 21 80
0000003 30 50 71 36 30
0000007 50 2  33 15 62
```

コードの見どころ

[1] 配信定義から配信元ファイル／配信先ファイル／配信先ホストの配信用リストを作成します。
(❶43 ～ 101行目)

[2] 配信定義の同期条件に従い、配信処理を行います。
(❷109 ～ 166行目)

[3] 正常に配信が終了したことを記録します。
(❸169 ～ 177行目まで)

まとめ

前回号で説明したようにユニケージでは、中規模以上の開発になると、業務・データ単位でサーバを分散配置することがあるため、データ配信が必要となります。配信は、ディレクトリ単位／ファイル単位で管理することができるので、配信先の業務サーバで使用するファイルだけを配信できるため、配信時のネットワーク負荷も最小限に抑えることができます。