コードレビュー

Vol.13

CODE Review

USP 研究所技術研究員
written by 大内智明

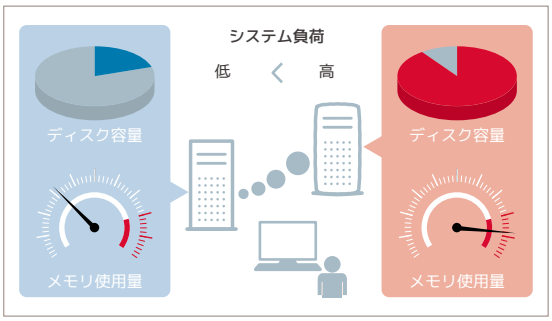
今回は、運用機能の一つであるメモリ情報の監視についてお話します。

監視機能の必要性

業務で使用するサーバーは、正常な動作を継続していく必要があります。そのためには、システム情報を定期的に監視して、取得した情報が正常な動作範囲であること、エラーや警告の発生も許容範囲内であることを確認していく必要があります。

もし、システム利用状況が許容範囲から外れても、常に監視を行っていれば早急にシステム負荷やハードウエア故障に対策／対応ができます（図1 参照）。

図1　システム運用状況



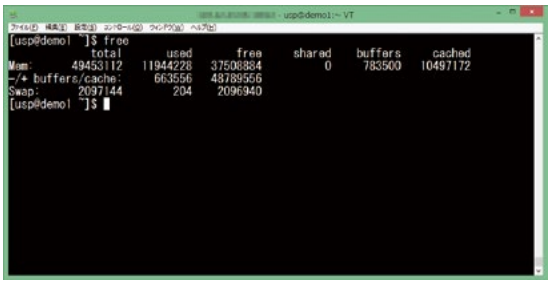
技術的な概要

システムは、定期的にメモリ情報を保存するシェルと保存した情報からメモリの最新情報を取得して、画面に出力する CGI の2セットが必要になります。

2-1. 定期的にメモリ情報を保存するシェルの説明

メモリ情報は、free コマンドを実行することで、物理メモリの使用サイズ・未使用サイズ、仮想メモリの使用サイズ・未使用サイズを取得することができます（画面キャプチャ 1）。メモリ取得は定期的に実行して、累積することで、システム負荷状況を把握することができます。取得したデータは、画面出力用に加工して、日単位のデータとして保存します。

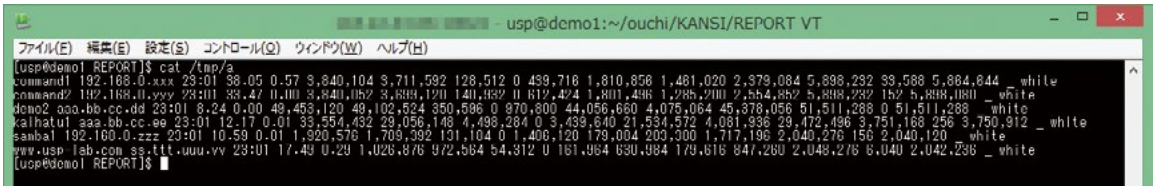
画面キャプチャ 1　データ形式と使う場面



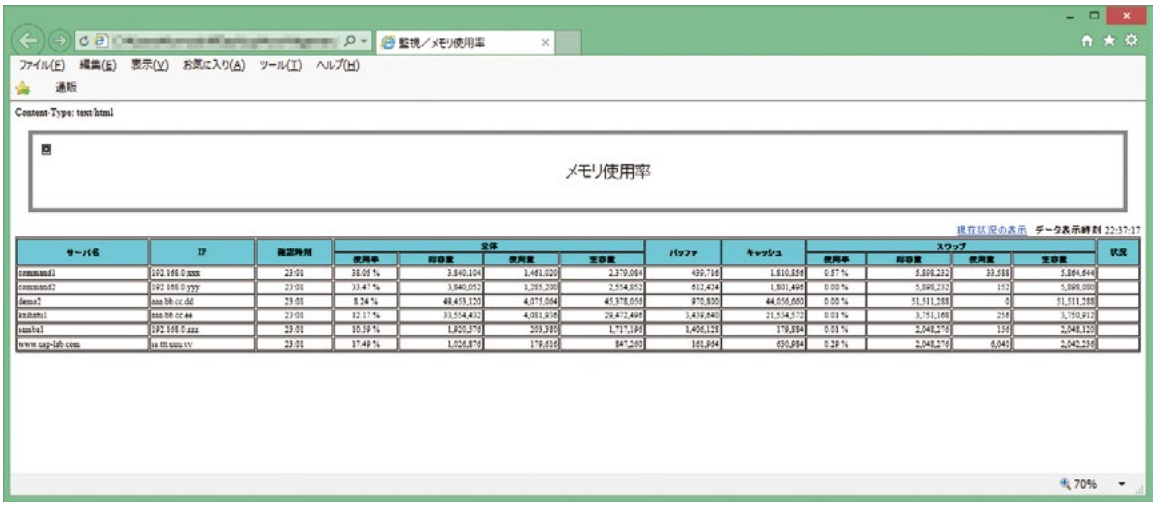
2-2. 画面に出力する CGI の説明

画面出力用データは、保存しているデータから最新情報を取得して、メモリ使用率を元に正常・警告・エラーなどの情報を付加して作成します（画面キャプチャ 2）。次に HTML 用テンプレートに画面出力用データを貼り付けて HTML ファイルを作成します（画面キャプチャ 3）。

画面キャプチャ 2　データ形式と使う場面



画面キャプチャ３　データ形式と使う場面



リスト１　定期的に起動して、複数のサーバーのメモリ使用率を保存するシェル

```
#!/bin/bash -xv
#
# KANSI_MEMORY
# メモリ使用率収集
#
# Written by aoki 20130902

# ログ出力
appd=/home/usp/KANSI
logfile=$appd/LOG/$(basename $0).$(date +%Y%m%d)
exec 2> $logfile

# 変数設定
export LANG=ja_JP.UTF-8
export PATH=/home/UTL:/home/TOOL:$PATH
cgid=$appd/CGI
htmd=$appd/HTML
logd=$appd/LOG
repd=$appd/REPORT
semd=$appd/SEMAPHORE
shld=$appd/SHELL
tbld=$appd/TABLE
tmp=/tmp/$$
today=$(date +%Y%m%d)
todayhms=$(date +%Y%m%d%H%M%S)

# エラーチェックと終了処理の定義
ERROR_CHECK(){
  [ $(plus ${PIPESTATUS[@]}) -eq 0 ] && return
  rm -rf $tmp-*
  touch $semd/$(basename $0).$HOSTNAME.ERROR.$today
  exit 1
}

```

```
35 # 起動時刻
36 touch $semd/$(basename $0).$HOSTNAME.START.$today
37
38 ######################################################################
39 # テーブルを読み込んで用意しておく
40 cat /etc/hosts | decomment | self 2 1 | hsort key=1 > $tmp-ip
41 ERROR_CHECK
42 touch ${repd}/MEMORY.${today}
43
44 # 各サーバーごとに容量を計測
45 for host in $(msctrl -ctrl C -print host); do
46    ip=$(nameread $host $tmp-ip | itouch _ -)
47    ssh ${host} free |
48        yarr |
49        self 8/13 16 17 19/21 |
50        # 1:total 2:used 3:free 4:shared 5:buffers 6:cached 7:bc-used 8:bd-free 9:swap-total 10:swap-used
          11:swap-free
51    awk 'NF==11{print "'${host}'","'${ip}'","'${todayhms}'",$7/$1*100,$10/$9*100,$0}'
52 done |
53 # 1:サーバー名 2:ＩＰ 3:年月日時分秒 4:本体使用率 5:swap使用率
54 # 6:total 7:used 8:free 9:shared 10:buffers 11:cached 12:bc-used 13:bd-free 14:swap-total 15:swap-used 16:swap-
   free
55 LANG=C sort -k1,2                |
56 up3 key=1/3 $repd/MEMORY.$today > $repd/MEMORY.$today.new
57 ERROR_CHECK
58
59 # 最新のものに置き換え
60 mv $repd/MEMORY.$today.new $repd/MEMORY.$today
61 ERROR_CHECK
62
63 ######################################################################
64 # 終了
65 echo "$HOSTNAME $(basename $0) END $(date +%Y%m%d%H%M%S)" >> $logd/UPCNT
66 touch $semd/$(basename $0).$HOSTNAME.END.$today
67 rm -f $tmp-*
68 exit 0
```

画面１　yarr：データを縦型にします

```
$ cat data
0000000 浜地_____ 50 F 76
0000001 鈴田_____ 50 F 46
$ yarr data
0000000 浜地_____ 50 F 76 0000001 鈴田_____ 50 F 46
```

リスト２　保存したメモリ使用率を取得して、画面出力する

```
 1 #!/bin/bash -xv
 2 #
 3 # KANSI_MEMORY.CGI
 4 # メモリ使用率監視
 5 #
 6 # Written by aoki 20130902
 7
 8 #ログの出力
 9 appd=/home/usp/KANSI
10 logfile=$appd/LOG/$(basename $0).$(date +%Y%m%d)
11 exec 2> $logfile
12
13 # 変数設定
14 export LANG=ja_JP.UTF-8
15 export PATH=/home/UTL:/home/TOOL:$PATH
16 cgid=$appd/CGI
17 htmd=$appd/HTML
```

```
18 logd=$appd/LOG
19 repd=$appd/REPORT
20 semd=$appd/SEMAPHORE
21 shld=$appd/SHELL
22 tbld=$appd/TABLE
23 tmp=/tmp/$$
24 today=$(date +%Y%m%d)
25 todayhms=$(date +%Y%m%d%H%M%S)
26
27 # エラーチェックと終了処理の定義
28 ERROR_CHECK(){
29   [ $(plus ${PIPESTATUS[@]}) -eq 0 ] && return
30   rm -rf $tmp-*
31   touch $semd/$(basename $0).$HOSTNAME.ERROR.$today
32   exit 1
33 }
34
35 # 起動時刻
36 touch $semd/$(basename $0).$HOSTNAME.START.$today
37
38 ##############################################################################################################
39 [ "$QUERY_STRING" == "now=1" ] && $shld/KANSI_MEMORY
40
41 # 監視ファイルのデータ形成
42 # 1:サーバー名 2:ＩＰ 3:年月日時分秒 4:本体使用率 5:swap使用率
43 # 6:total 7:used 8:free 9:shared 10:buffers 11:cached 12:bc-used 13:bd-free 14:swap-total 15:swap-used 16:swap-
   free
44 touch $repd/MEMORY.$today
45 cat $repd/MEMORY.$today |
46 getlast 1 2             |   getlastは、画面2参照
47 dayslash HH:MM 3        |   dayslashは、画面3参照
48 marume 4.2 5.2          |   marumeは、画面4参照
49 comma 6/16              |   commaは、画面5参照
50 awk '{s="_";c="white";
51       if(90<=$4       ){s="90%以上危険";  c="red"  }
52       if(80<=$4&&$4<90){s="80%以上注意";  c="pink" }
53       if(70<=$4&&$4<80){s="70%以上要確認";c="white"}
54       print $0,s,c;}' > $tmp-check
55 ERROR_CHECK
56
57 ############################################################
58 # HTML作成
59 ############################################################
60 # 出力
61
62 echo "Content-Type: text/html"
63 echo ""
64 cat ${htmd}/KANSI_MEMORY.HTML            |   calsedは、画面6参照
65 calsed "###DATE###" $(dayslash -d --output HH:MM:SS $todayhms) |   mojihameは、画面7参照
66 mojihame -l###REPORT_LOOP### - $tmp-check  |
67 cat
68
69 ############################################################
70 # 終了
71 touch $semd/$(basename $0).$HOSTNAME.END.$today
72 rm -rf $tmp-*
73 exit 0
```

❷ ❸

画面 2 getlast：同一キーの最終行を出力します

```
$ cat data
0000007 セロリ 20060201 117
0000007 セロリ 20060202 136
0000007 セロリ 20060203 221
0000017 練馬大根 20060201 31
0000017 練馬大根 20060202 127
0000017 練馬大根 20060203 514

$ getlast 1 1 data
0000007 セロリ 20060203 221
0000017 練馬大根 20060203 514
```

画面 3 dayslash：日付時刻に変換します

```
$ echo 20120304 | dayslash yyyy/mm/dd 1
2012/03/04

$ echo 20120304093000 | dayslash HH:MM 1
09:30
```

画面 4 marume：四捨五入、切り上げ、切捨てします

```
2列目の小数第1位と3列目小数第2位を四捨五入
$ echo 01 0.3418 1.5283 | marume 2.0 3.1
01 0 1.5
2列目の小数第1位と3列目小数第2位を切り上げ
$ echo 01 0.3418 1.5283 | marume +age 2.0 3.1
01 1 1.6
2列目の小数第1位と3列目小数第3位を切捨て
$ echo 01 0.3418 1.5283 | marume -sage 2.0 3.2
01 0 1.52
```

画面 5 comma：指定フィールドに 3 桁コンマをふる

```
$ echo 123456789 | comma 1
123,456,789
```

画面 6 calsed：sed コマンドの簡易版

```
$ echo AAA | calsed "AAA" "123"
123

$ echo AAA | calsed "AAA" "1 2 3"
1 2 3
```

画面 7 mojihame:テンプレートに文字をはめ込みます

```
$ cat template
###LINE###
1st=%1
2nd=%2
3rd=%3 4th=%4
###LINE###

$ cat data
a1 b1 c1 d1
a2 b2 c2 d2

$ mojihame -l###LINE### template data
1st=a1
2nd=b1
3rd=c1 4th=d1
1st=a2
2nd=b2
3rd=c2 4th=d2
```

data ファイルの1行ずつをテンプレートにはめ込みます。複数行ある場合は、繰り返します。はめ込む場所は、1列目が %1、2列目が %2・・・・と対応しています。

コードの見どころ

[1] 複数のサーバーのメモリ情報の取得及び保存
（❶ リスト1. 39 ～ 61 行目まで）。
[2] 保存したメモリ情報を画面出力用に加工
（❷ リスト2. 41 ～ 55 行目まで）。
[3] HTML データを作成
（❸ リスト2. 60 ～ 67 行目まで）。

まとめ

業務サーバーを管理する運用システムは、システムの状態を知る上で重要です。常にメモリ監視を行うことで、パフォーマンスの低下、システム処理遅延の一因になるメモリ不足に対して、できるだけ早期に原因究明や対策を行うことができます。今回は紹介していませんが、エラーや警告が発生するタイミングで、自動的にメールで通知することにより、早期にメモリ問題に対して対応が可能になります。