### Table des matières

**Quels process ont utilisé de la swap ?** ..... 3
***Avec top*** ..... 3
***Avec sar*** ..... 3
**Quantité réelle de mémoire utilisée par un process** ..... 3
**Script utiles** ..... 4
***Conso par process pour un user donné*** ..... 4
**Mémoire cache** ..... 5
**Récupérer des infos sur la mémoire** ..... 5
**Liens en vrac** ..... 5

# Quels process ont utilisé de la swap ?

### Avec top

Avec *top* on peut le déterminer. *top → f → p → espace* pour choisir le champ **SWAP** qui nous intéresse. On peut voir aussi l'activité globale de la machine avec *procinfo*.

### Avec sar

- sar -r

```
root@old_machine:/root> sar -r 1 5
Linux 2.4.21-37.ELhugemem (old_machine)        02/05/2008

08:15:47 AM kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree kbswpused  %swpused  kbswpcad
08:15:48 AM     10216  16294308     99.94     51144  15813208  13808660   4017116     22.54     20504
08:15:49 AM     10212  16294312     99.94     51144  15813208  13808660   4017116     22.54     20504
08:15:50 AM     10188  16294336     99.94     51144  15813228  13808660   4017116     22.54     20504
08:15:51 AM     10808  16293716     99.93     51152  15812740  13808660   4017116     22.54     20504
08:15:52 AM     10640  16293884     99.93     51152  15812764  13808660   4017116     22.54     20504
Average:        10413  16294111     99.94     51147  15813030  13808660   4017116     22.54     20504
```

- sar -W

```
root@parcl1102325:/root> sar -W |tail -6

                pswpin/s pswpout/s
09:35:00 AM      0.09      0.00
09:40:01 AM      0.01      0.00
09:45:00 AM      0.05      0.00
09:50:00 AM      0.01      0.00
09:55:00 AM      0.07      0.00
Average:         0.60      0.87
```

Ou ce code permet d'avoir des infos sur les process qui swappent :

```
#!/bin/bash

ps -ef |awk '{print $2}'|grep -v "PID" |sort -u |while read i
do
sar -x $i 1 |egrep -v "Average|`hostname`"
done
```

# Quantité réelle de mémoire utilisée par un process

Soit le process suivant :

```
www-data 13037 0.0 3.4 89672 34664 ? S Sep04 1:38 /usr/sbin/apache2 -k start -DSSL
```

D'après *ps* on peut penser que le process utilise **89 672 Ko** de mémoire ce qui est faux. On utilise *pmap* pour obtenir des infos précises sur le process :

```
dedibox:/# pmap -d 13037
13037:   /usr/sbin/apache2 -k start -DSSL
Address   Kbytes Mode  Offset           Device    Mapping
08048000     368 r-x-- 0000000000000000 008:00005 apache2
080a4000      12 rw--- 000000000005b000 008:00005 apache2
080a7000    7432 rw--- 00000000080a7000 000:00000   [ anon ]
b2fe0000    1416 r---- 0000000000000000 008:00005 locale-archive
b3142000      56 rw-s- 0000000000000000 000:00007 zero (deleted)
b3150000   65536 rw-s- 0000000000000000 000:00007 zero (deleted)
b7150000     252 r-x-- 0000000000000000 008:00005 libmysqlclient.so.14.0.0
b718f000     796 rw--- 000000000003e000 008:00005 libmysqlclient.so.14.0.0
b7256000       8 rw--- 00000000b7256000 000:00000   [ anon ]
b7258000      40 r-x-- 0000000000000000 008:00005 mysql.so
... truncated output ...
b7f6f000     160 rw--- 00000000b7f6f000 000:00000   [ anon ]
b7f97000       8 r-x-- 0000000000000000 008:00003 libdl-2.3.2.so
b7f99000       4 rw--- 0000000000002000 008:00003 libdl-2.3.2.so
b7f9c000      20 r-x-- 0000000000000000 008:00005 mod_cgi.so
```

```
b7fa1000       4 rw--- 0000000000005000 008:00005 mod_cgi.so
b7fa2000       4 rw--- 00000000b7fa2000 000:00000   [ anon ]
b7fa3000       4 r-x-- 00000000b7fa3000 000:00000   [ anon ]
b7fa4000      88 r-x-- 0000000000000000 008:00003 ld-2.3.2.so
b7fba000       4 rw--- 0000000000015000 008:00003 ld-2.3.2.so
bfd33000     132 rw--- 00000000bfd33000 000:00000   [ stack ]
mapped: 89672K    writeable/private: 9516K    shared: 65592K
```

Le process utilise en fait **9 516K** de mémoire. 2 explications à cela :

1. La commande *ps* comptabilise, entre autres, toutes les librairies pouvant être partagées entre process.
2. On voit également que les librairies sont présentes en double : 1 fois pour les *code segments* et 1 fois pour les *data segments*.

Au final on se retrouve avec une valeur faussée. Et effectivement en utilisant *ps* si on prend, par exemple, tous les process Apache d'une machine on arrive à un mauvais résultat :

```
root     11519  0.0  0.0 83184  248 ?        Ss   Sep04   0:08 /usr/sbin/apache2 -k start -DSSL
www-data 13037  0.0  3.4 89672 34668 ?       S    Sep04   1:44 /usr/sbin/apache2 -k start -DSSL
www-data 16412  0.0  2.6 87656 26816 ?       S    09:18   0:47 /usr/sbin/apache2 -k start -DSSL
www-data 16519  0.0  2.8 87524 29280 ?       S    09:20   0:28 /usr/sbin/apache2 -k start -DSSL
www-data 19089  0.0  2.9 87364 30380 ?       S    09:41   0:32 /usr/sbin/apache2 -k start -DSSL
www-data 23052  0.0  2.4 86716 24732 ?       S    14:01   0:27 /usr/sbin/apache2 -k start -DSSL
www-data 11447  0.0  2.0 87780 20672 ?       S    16:28   0:20 /usr/sbin/apache2 -k start -DSSL
www-data 14217  0.1  2.1 85504 21440 ?       S    16:51   0:27 /usr/sbin/apache2 -k start -DSSL
www-data   880  0.2  1.2 85992 12556 ?       S    22:47   0:03 /usr/sbin/apache2 -k start -DSSL
www-data   883  0.9  3.1 105212 32136 ?      S    22:47   0:14 /usr/sbin/apache2 -k start -DSSL
www-data   884  0.6  3.1 105436 32260 ?      S    22:47   0:09 /usr/sbin/apache2 -k start -DSSL
```

Cela voudrait dire qu'Apache seul utilise environ 1 Go de RAM Et sur une machine ne disposant que d'1Go de mémoire physique + 1 Go de swap ça semble difficile. A la rigueur la 6ème colonne du *ps* se rapprocherait de la vraie valeur.

On peut donc en déduire combien Apache consomme réellement :

```
dedibox:~# ps aux|grep apac|grep -v grep|awk '{print $2}'|while read i; do pmap -d $i|grep private; done
mapped: 83184K    writeable/private: 4456K    shared: 65592K
mapped: 87656K    writeable/private: 7500K    shared: 65592K
mapped: 87524K    writeable/private: 7368K    shared: 65592K
mapped: 87364K    writeable/private: 7208K    shared: 65592K
mapped: 86896K    writeable/private: 6740K    shared: 65592K
mapped: 87780K    writeable/private: 9052K    shared: 65592K
mapped: 85504K    writeable/private: 6776K    shared: 65592K
mapped: 85992K    writeable/private: 7264K    shared: 65592K
mapped: 105228K    writeable/private: 26488K    shared: 65592K
mapped: 105436K    writeable/private: 26708K    shared: 65592K
mapped: 87688K    writeable/private: 8960K    shared: 65592K
```

Soit 118 520 Ko.

Source : (extraits)

If you go through the output, you will find that the lines with the largest Kbytes number are usually the code segments of the included shared libraries (the ones that start with “lib” are the shared libraries). What is great about that is that they are the ones that can be shared between processes. If you factor out all of the parts that are shared between processes, you end up with the “writeable/private” total, which is shown at the bottom of the output. This is what can be considered the incremental cost of this process, factoring out the shared libraries. Therefore, the cost to run this instance of KEdit (assuming that all of the shared libraries were already loaded) is around 2 megabytes. That is quite a different story from the 14 or 25 megabytes that ps reported.

One important thing to note about the output is that each shared library is listed twice; once for its code segment and once for its data segment. The code segments have a mode of “r-x–”, while the data is set to “rw—”. The Kbytes, Mode, and Mapping columns are the only ones we will care about, as the rest are unimportant to the discussion.

http://virtualthreads.blogspot.com/2006/02/understanding-memory-usage-on-linux.html

 Attention cependant, selon les programmes utilisés (apache, MySQL, etc) ces constations ne sont pas forcément vraies.

## Script utiles

### Conso par process pour un user donné

```
USER=was6; ps -fu $USER|grep -v PID|awk '{print "cat /proc/"$2"/status|egrep \"Name|VmRSS\";echo"}' |sh
```

## Mémoire cache

- Voir le cache utilisé avec :

```
free -m
sar -r
```

- Voir le contenu du cache avec :

```
slabtop
```

- Gérer le cache avec :

```
sync
echo 1 > /proc/sys/vm/drop_caches  # free pagecache
     [OR]
echo 2 > /proc/sys/vm/drop_caches  # free dentries and inodes
     [OR]
echo 3 > /proc/sys/vm/drop_caches  # free pagecache, dentries and inodes
sync  # forces the dump to be destructive
```

```
vfs_cache_pressure
pagecache
```

- Voir la fragmentation mémoire

```
cat /proc/buddyinfo
```

## Récupérer des infos sur la mémoire

Un tar est disponible ici.

## Liens en vrac

http://www.eygle.com/digest/2007/07/linux_memory_management_or_why.html
http://www.dba-oracle.com/t_tuning_linux_kernel_2_6_oracle.htm
http://www.puschitz.com/TuningLinuxForOracle.shtml
http://www.oreilly.fr/contenu/2007/04/26/surveiller-la-consommation-m%C3%A9moire-sous-linux
http://forums.gentoo.org/viewtopic.php?p=1210748
http://www.westnet.com/~gsmith/content/linux-pdflush.htm

From:
https://unix.ndlp.info/ - **Where there is a shell, there is a way**

Permanent link:
**https://unix.ndlp.info/doku.php/informatique:nix:linux:linux_mem?rev=1340107973**

Last update: **2012/06/19 12:12**