Notice pour la métrique 2.2 (Mémoire)

Récupération des informations

Pour récupérer des informations sur la mémoire sous un système Linux, il faut aller chercher dans le fichier `/proc/meminfo`, dans le quel on trouve beaucoup d'informations. Pour étudier la mémoire, on va se focaliser sur 2 données, la mémoire "classique" et la mémoire swap. La mémoire "classique" est celle utilisée par le système par défaut, quand cette mémoire est pleine, c'est la mémoire Swap qui prend le relai, bien qu'elle soit plus lente.

Utilisation de bash

Pour récupérer ces données, nous allons utiliser bash. Mais avant de faire quoi que ce soit, nous allons créer un fichier csv vide. Pour celà, j'ai décidé de faire un script qui permet de reset le fichier de capture `data.csv`.

```
#!/bin/bash
# created by Brewal Guyon
# last modification 17-11-22

line="time,MemTotal,MemFree,MemAvilable,SwapCached,SwapTotal,SwapFree"
echo $line > data.csv

exit 0
```

Ce script permet d'écraser les données de `data.cvs` et de les remplacer par la ligne qui contient les noms de colonnes.

Maintenant que l'on a un fichier propre, il est temps d'éxtraire les données.

```
#!/bin/bash
# created by Brewal Guyon
# last modification 17-11-22

location='/proc/meminfo'

res=`cat $location | grep -E "^(Mem|Swap)"`

IFS=$'\n'

echo -n `date +'%s'`, >> /root/Documents/sae15/data.csv
```

```
for d in $res:
do
   line=$line`echo $d | tr -s ' ' | tr -d ':' |  cut -d ' ' -f 2-3`,
done

echo -n ${line::-1} >> /root/Documents/sae15/data.csv
echo "" >> /root/Documents/sae15/data.csv

# tr -s -> replace multiple -s into one | tr -s -> remove all (: because there is :

exit 0
```

Le script va d'abord chercher les lignes qui commencent par Mem ou Swap dans le contenu de `/proc/meminfo`. Ensuite il ajoute la date à la nouvelle ligne de `data.csv`. Le script continu faisant une boucle qui prend chaque ligne récupéré par la commande au dessus. Pour chaque de ces lignes, le script remplace les espaces multiples par un seul et enlève les `:`. Pour comprendre le reste de la ligne, prenons un exemple d'une ligne du fichier.

```
MemTotal:        203124 kB
```

Le debut du script (qui a été expliqué plus haut) va transformer cette ligne en

```
MemTotal 203124 kB
```

Le cut va par la suite récuperer les champs de 2 à 3, c'est à dire le nombre et l'unité.

Le script va finir par ajouter la ligne au fichier csv, en enlevant le dernier caractère qui est une virgule en trop. Et enfin, le script va finir par ajouter un saut de ligne en fin de ligne.

Pour faire varier les données, j'ai choisi de créer une variable qui va grandir exponentiellement. Pour ça, j'ai fait un script bash.

```
#!/bin/bash
# created by Brewal Guyon
# last modification 24-11-22




a="hello"

for i in {0..8}
do
a=$a$a$a$a$a
```

```
done

echo taille de la variable : $(echo $a | wc -c)
# cree une variable assez longue, pour faire varier la memoire

for i in {0..10}
do
echo phase $i/10 du script

echo mise en pause du script pendant 60s
sleep 60

echo changement de la taille de la variable
a=$a$a
echo taille de la variable : $(echo $a | wc -c)
done

exit 0
```

Le script fait varier la taille de la variable toute les 60s (fréquence max de cron).Il faut donc executer les commandes suivantes pour capturer les données.

```
./varier.sh
systemctl restart cron
```

Le script varier donne ce résultat suivant en console.

```
root@debian10:~/Documents/sae15# ./varier.sh
taille de la variable : 9765626
phase 0/10 du script
mise en pause du script pendant 60s
changement de la taille de la variable
taille de la variable : 19531251
phase 1/10 du script
mise en pause du script pendant 60s
changement de la taille de la variable
taille de la variable : 39062501
phase 2/10 du script
mise en pause du script pendant 60s
....
```