

Cahier de TPs

Linux - Administration avancée

jour 4

François MICAUX - ACTILIS - 26.10

Table des matières

JOUR 4

Module 14 Analyse de la charge et de l'activité système 3

Module 15 Optimiser les performances 7

Module 16 Collecter et grapher des métriques système 11

Module 14

Analyse de la charge et de l'activité système

TP : charge, outils instantanés et sysstat

Objectif et machine

Produire une charge **CPU**, puis une charge d'**I/O**, et les lire avec les outils instantanés, la **PSI** et `sysstat`.

La machine

Une VM AlmaLinux 10 avec au moins 2 CPU, un accès à Internet et un compte qui dispose de `sudo`. Tout se fait en SSH, avec **deux sessions** ouvertes en parallèle : l'une produit la charge, l'autre l'observe.

```
$ ssh admin@vm3-ks
```

> **Un redémarrage au programme**
>
> L'étape 1 active la PSI : elle demande un redémarrage de la VM.

Étape 1 : Activer PSI

```
$ ls /proc/pressure
sudo grubby --update-kernel=ALL --args=psi=1
sudo grubby --info=DEFAULT | grep -o 'psi=1'
sudo systemctl reboot
```

Après le redémarrage :

```
$ ls /proc/pressure
cat /proc/pressure/cpu /proc/pressure/io /proc/pressure/memory
```

Questions :

1. Que répond `ls /proc/pressure` avant et après ?
2. Quelles lignes donnent `/proc/pressure/io` ? Que signifient `some` et `full` ?

```
$ nproc
cat /proc/loadavg
```

Questions :

3. Combien de CPU la VM a-t-elle ?

Étape 2 : Une charge CPU

Dans la **session 1**, lancer trois calculs en arrière-plan :

```
$ for i in 1 2 3; do sha256sum /dev/zero > /dev/null & done
jobs
```

Dans la **session 2**, après une minute :

```
$ uptime
```

Questions :

1. Quelle charge `uptime` affiche-t-il, face au nombre de CPU ?

```
$ top -b -n1 | head -12
vmstat -w 1 3
```

Questions :

2. Que valent `r`, `id` et `wa` dans `vmstat` ?

```
$ mpstat -P ALL 1 1
pidstat -u 1 1
```

Questions :

3. Quel outil désigne les processus responsables ?

Étape 3 : Par CPU, puis arrêter la charge

`mpstat -P ALL` détaille l'usage de chaque coeur. Dans la session 1, arrêter les calculs :

```
$ kill $(jobs -p)
jobs
```

Dans la session 2, avant puis après l'arrêt :

```
$ cat /proc/pressure/cpu
uptime
```

Questions :

1. `mpstat` montre-t-il un CPU plus chargé que l'autre ?
2. Que donne `some` dans `/proc/pressure/cpu` pendant la charge ?
3. La charge retombe-t-elle tout de suite après l'arrêt ? Pourquoi ?

Étape 4 : Charge élevée, CPU libre

Session 1 : 15 000 blocs de 4 Kio synchronisés sur le disque. Sans `oflag=dsync`, `dd` écrit dans le cache de l'hôte et aucune attente n'apparaît.

```
$ dd if=/dev/zero of=/var/tmp/tp-io bs=4k count=15000 oflag=dsync
rm /var/tmp/tp-io
```

Session 2, pendant l'écriture :

```
$ vmstat -w 1 3
sar -u 1 3
```

Questions :

1. Quelles colonnes de `vmstat` et de `sar -u` montrent l'attente d'I/O ?

```
$ iostat -x -d -k vda 1 2
```

Questions :

2. Que valent `%util` et `w_await` dans `iostat` ?

```
$ pidstat -d 1 2
```

Questions :

3. Quel processus engendre les écritures ? Quel outil le montre ?

```
$ cat /proc/pressure/io
```

Questions :

4. Que valent `some` et `full` dans `/proc/pressure/io` ?

Étape 5 : La mémoire

```
$ free -h
grep -E 'MemTotal|MemAvailable|^Cached' /proc/meminfo
```

Questions :

1. Quelle colonne de `free` indique la mémoire réellement allouable ?

```
$ sudo sh -c 'echo 3 > /proc/sys/vm/drop_caches'
free -h
vmstat -s | head -8
```

Questions :

2. La colonne `buff/cache` est-elle de la mémoire perdue ? Que montre-t-on en la vidant ?

Étape 6 : Enregistrer et rejouer

Vérifier la collecte, puis enregistrer soi-même :

```
$ rpm -q sysstat
sudo dnf install -y sysstat
sudo systemctl enable --now sysstat
systemctl list-timers 'sysstat*' --no-pager
ls /var/log/sa
```

Produire de la charge pendant un enregistrement :

```
$ sha256sum /dev/zero > /dev/null &
sar -o /var/tmp/tp.sa 1 5 > /dev/null
kill %1
sar -u -f /var/tmp/tp.sa
sar -q -f /var/tmp/tp.sa
```

Questions :

1. Quel mécanisme déclenche la collecte de `sysstat` ? Où sont les fichiers `saXX` ?
2. Que gagne-t-on à enregistrer plutôt qu'à observer en direct ?

Étape 7 : Exporter avec `sadf`

```
$ sadf -d /var/tmp/tp.sa -- -u | head -3
sadf -j /var/tmp/tp.sa -- -u | head -6
```

Questions :

1. Quels formats d'export `sadf` propose-t-il ?

```
$ sar -q PSI 1 3
sar -n DEV 1 2
rm /var/tmp/tp.sa
```

Questions :

2. Quelle option de `sar` affiche la PSI ? Que faut-il pour qu'elle fonctionne ?

> **Nettoyer**
>
> Retirer `psi=1` si la PSI n'est plus utile : `sudo grubby --update-kernel=ALL --remove-args=psi=1`, puis redémarrer. Le TP d'optimisation qui suit peut s'en servir.

Module 15

Optimiser les performances

TP : mesurer, régler le disque, les processus et le réseau

Objectif et machine

Appliquer la méthode « **mesurer, régler, re-mesurer** » : tester un disque, changer le scheduler d'I/O, régler un montage, appliquer un profil `tuned`, hiérarchiser des processus, lire le démarrage, tester le réseau.

La machine

La VM du TP précédent (AlmaLinux 10, 2 CPU, accès à Internet, compte avec `sudo`), en SSH. Les outils de ce TP viennent d'AppStream, de BaseOS et d'EPEL :

```
$ sudo dnf install -y epel-release
sudo dnf install -y fio ioping ps_mem iperf3 mtr time tuned
```

> ⚠ Un seul réglage à la fois
>
> Chaque réglage est noté avec sa valeur d'origine et remis à l'état initial à l'étape 12.

Étape 1 : Identifier le support

```
$ lsblk -d -o NAME,ROTA,DISC-GRAN,MODEL
```

Questions :

1. Que signifie la colonne `ROTA` ? Quelle valeur la VM annonce-t-elle ?

```
$ for d in vda vdb vdc; do echo $d $(cat /sys/block/$d/queue/scheduler); \
  done
```

Questions :

2. Quel scheduler est actif sur chaque disque, et que signifient les crochets ?

Étape 2 : Mesurer avec `fio` et `ioping`

Un fichier de test dans `/var/tmp`, pas un périphérique :

```
$ df -h /var/tmp
fio --name=test --filename=/var/tmp/fio-test --size=256M --direct=1 \
    --rw=randrw --bs=4k --numjobs=2 --runtime=10 --time_based \
  --group_reporting
rm /var/tmp/fio-test
```

Questions :

1. Quels débits et IOPS `fio` annonce-t-il en lecture et en écriture ?

```
$ ioping -c 5 /var/tmp
```

Questions :

2. Quelle latence `ioping` mesure-t-il (`min/avg/max`) ?

3. Pourquoi ne pas lancer `fio` sur `/dev/vda` ?

Étape 3 : Changer le scheduler d'I/O

Sur `vdb`, un disque sans données *(disque supplémentaire du TP « Stockage », laissé vierge, rattaché à la VM à la fin du TP « Booter autrement »)* :

```
$ echo mq-deadline | sudo tee /sys/block/vdb/queue/scheduler
cat /sys/block/vdb/queue/scheduler
```

Rendre le réglage **durable** avec une règle `udev`. Créer `/etc/udev/rules.d/60-tp-scheduler.rules` avec un éditeur lancé en root (`sudo`) :

```
ACTION=="add|change", KERNEL=="vdb", ATTR{queue/scheduler}="none"
```

Puis la tester :

```
$ sudo udevadm control --reload
sudo udevadm trigger --subsystem-match=block --action=change
cat /sys/block/vdb/queue/scheduler
```

Questions :

1. Le changement par `echo` survit-il à un redémarrage ?
2. Que fait la règle `udev` ?

Étape 4 : Options de montage et TRIM

```
$ findmnt -no OPTIONS /var
sudo mount -o remount,noatime /var
findmnt -no OPTIONS /var
sudo mount -o remount,relatime /var
```

Questions :

1. Quelle option de date d'accès `/var` utilise-t-il au départ ?

```
$ systemctl list-timers fstrim.timer --no-pager
sudo fstrim -v /var
```

Questions :

2. Quel mécanisme lance `fstrim` périodiquement ?

Étape 5 : Un profil `tuned`

```
$ sudo systemctl enable --now tuned
tuned-adm active
tuned-adm recommend
```

Questions :

1. Quel profil est actif, et lequel `tuned` recommande-t-il ?

```
$ sysctl vm.swappiness vm.dirty_ratio
sudo tuned-adm profile throughput-performance
sysctl vm.swappiness vm.dirty_ratio
```

Questions :

2. Quelles valeurs de `vm.swappiness` et `vm.dirty_ratio` avant et après le changement de profil ?

Étape 6 : Où est la mémoire ?

```
$ free -h
sudo ps_mem | tail -6
```

Questions :

1. Quel programme occupe le plus de mémoire selon `ps_mem` ? Quelle grandeur additionne-t-il ?

```
$ sudo slabtop -o | sed -n 7,11p
```

Questions :

2. Que contient `slabtop` ?

> **À savoir**
>
> `smem`, cité dans d'anciennes documentations, n'existe pas dans EPEL 10 : `ps_mem` en tient lieu.

Étape 7 : Hiérarchiser avec `nice`

Trois calculs lancés ensemble, puis un renice :

```
$ sha256sum /dev/zero > /dev/null & P1=$!
sha256sum /dev/zero > /dev/null & P2=$!
sha256sum /dev/zero > /dev/null & P3=$!
sleep 6; ps -o pid,ni,pcpu,comm -p $P1,$P2,$P3
renice -n 19 -p $P3
sleep 6; ps -o pid,ni,pcpu,comm -p $P1,$P2,$P3
kill $P1 $P2 $P3
```

Questions :

1. Quelle part de CPU chaque calcul reçoit-il avant et après `renice` ?
2. Un utilisateur peut-il remettre `P3` à une priorité plus élevée ?

Étape 8 : Mesurer une commande

```
$ /usr/bin/time -v tar -czf /dev/null /usr/lib64 2>&1 | \
  grep -E 'Elapsed|Maximum resident|Percent'
nice -n 19 /usr/bin/time -v tar -czf /dev/null /usr/lib64 2>&1 | \
  grep -E 'Elapsed|Maximum resident'
```

Questions :

1. Quels sont le temps réel et le pic de mémoire (`Maximum resident set size`) de la commande ?
2. Qu'ajoute `/usr/bin/time` à la commande `time` du shell ?

Étape 9 : Analyser le démarrage

```
$ systemd-analyze
```

Questions :

1. Quels temps `systemd-analyze` donne-t-il pour le noyau, l'initrd et l'espace utilisateur ?

```
$ systemd-analyze blame | head -6
systemd-analyze critical-chain | head -12
```

Questions :

2. Le service le plus long de `blame` est-il sur la chaîne critique ?

> **À savoir**
>
> Avec `blame | grep -v '.device'`, les unités de périphériques (virtio, qemu-guest-agent…) ne masquent plus les services.

Étape 10 : Tester le réseau

```
$ ping -c 3 127.0.0.1
ss -s
iperf3 -s -D
iperf3 -c 127.0.0.1 -t 3
pkill iperf3
```

Entre **deux VM**, remplacer `127.0.0.1` par l'adresse de l'autre VM, `iperf3 -s` y étant lancé.

Questions :

1. Quel débit `iperf3` mesure-t-il en bouclage ? Que vaut-il comme mesure du réseau ?
2. Quelle commande donne le nombre de connexions TCP ?

Étape 11 : L'algorithme TCP

```
$ sysctl net.ipv4.tcp_congestion_control \
  net.ipv4.tcp_available_congestion_control
sudo modprobe tcp_bbr
sysctl net.ipv4.tcp_available_congestion_control
```

Questions :

1. Quel algorithme est actif au départ ? Lesquels sont disponibles avant et après `modprobe` ?

```
$ sudo sysctl -w net.ipv4.tcp_congestion_control=bbr
sysctl net.ipv4.tcp_congestion_control
```

Questions :

2. Que faut-il pour garder `bbr` après un redémarrage ?

Étape 12 : Remettre à l'état initial

```
$ sudo tuned-adm profile virtual-guest
sudo sysctl -w net.ipv4.tcp_congestion_control=cubic
sudo rm -f /etc/udev/rules.d/60-tp-scheduler.rules
sudo udevadm control --reload
```

Questions :

1. Quelle règle de méthode le TP a-t-il appliquée à chaque réglage ?
2. Quel réglage a eu un effet **mesuré** dans ce TP ?

Module 16

Collecter et grapher des métriques système

TP : Cockpit et PCP, puis Prometheus, une alerte et Grafana

Objectif et machine

Observer une machine avec la solution **native** (Cockpit + PCP), puis monter la pile **Prometheus** : exporter, collecte, requêtes, alerte et tableau de bord Grafana.

La machine

Une VM AlmaLinux 10 avec accès à Internet, un compte qui dispose de `sudo` et **au moins 1 Go libre** sur `/` (Grafana en demande environ 650 Mo). Les pages web s'ouvrent depuis le navigateur du poste de travail, avec l'adresse de la VM.

```
$ ssh admin@vm3-ks
ip -4 -br addr show
df -h /
```

> ⚠ Une seule machine pour toute la pile
>
> Cockpit et Prometheus écoutent tous deux sur le port `9090` : l'étape 3 déplace Prometheus sur `9091`.

Étape 1 : Cockpit et PCP

```
$ sudo dnf install -y cockpit pcp-zeroconf
sudo systemctl enable --now pmcd pmlogger cockpit.socket
systemctl is-active pmcd pmlogger cockpit.socket
pminfo -f kernel.all.load
pmrep -s 3 kernel.all.load mem.util.available
```

Dans le navigateur : `https://<adresse de la VM>:9090`, connexion avec le compte `admin`, page **Metrics**.

Questions :

1. Quelles valeurs `pminfo` donne-t-il pour la charge, et sur quelles durées ?
2. Quels services collectent l'historique que la page **Metrics** affiche ?
3. Quelle page de Cockpit permet de relier une courbe à un service ou à un journal ?

> ℹ À savoir
>
> Le paquet `cockpit-pcp` n'existe plus : la page **Metrics** est fournie par `cockpit-system`.

Étape 2 : L'exporter

```
$ sudo dnf install -y epel-release
sudo dnf install -y node-exporter
sudo systemctl enable --now node_exporter
sudo ss -tlnp | grep 9100
```

Questions :

1. Quel port `node_exporter` écoute-t-il, et quel chemin expose les métriques ?
2. D'où vient le paquet `node-exporter` ?

```
$ curl -s localhost:9100/metrics | grep -E '^node_memory_MemAvailable'
curl -s localhost:9100/metrics | grep -E '^node_cpu.*cpu="0",mode="idle"'
curl -s localhost:9100/metrics | wc -l
```

Questions :

3. Pourquoi `node_cpu_seconds_total` croît-il sans cesse ?

Étape 3 : Prometheus

```
$ sudo dnf install -y prometheus
```

Remplacer le contenu de `/etc/prometheus/prometheus.yml` avec un éditeur lancé en root (*`sudo`*) par :

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: "noeuds"
    static_configs:
      - targets:
          - "localhost:9100"
```

Puis contrôler la syntaxe :

```
$ sudo promtool check config /etc/prometheus/prometheus.yml
```

Étape 4 : Port, rétention et démarrage

Éditer `/etc/default/prometheus` avec un éditeur lancé en root (*`sudo`*) et remplacer la ligne `ARGS=''` par :

```
ARGS='--web.listen-address=0.0.0.0:9091 --storage.tsdb.retention.time=30d'
```

```
$ grep ^ARGS /etc/default/prometheus
sudo systemctl enable --now prometheus
curl -s localhost:9091/-/ready
curl -s localhost:9091/api/v1/targets | python3 -m json.tool | \
  grep -E 'health|scrapeUrl'
```

Questions :

1. Que contrôle `promtool check config` ?
2. Pourquoi `--web.listen-address` ? Que se passerait-il sans lui ?
3. Quel état la cible `noeuds` affiche-t-elle ?

Étape 5 : Interroger avec PromQL

Dans le navigateur (`http://<VM>:9091`, onglet **Graph**) ou avec `curl` :

```
$ q() { curl -s -G localhost:9091/api/v1/query --data-urlencode "query=$1" |
    python3 -c 'import sys,json;print(json.load(sys.stdin)["data"]["result"][0]["value"]
[1])'; }
  q 'node_memory_MemAvailable_bytes'
  q '100 * node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes'
  q '100 * (1 - avg by (instance)(rate(node_cpu_seconds_total{mode="idle"}[5m])))'
  q 'node_filesystem_avail_bytes{mountpoint="/"}
     / node_filesystem_size_bytes{mountpoint="/"}'
```

Pour le CPU, `sha256sum /dev/zero > /dev/null &` produit de la charge (`kill %1` l'arrête). Ouvrir l'interface demande peut-être `sudo firewall-cmd --add-port=9091/tcp`.

Questions :

1. Quel pourcentage de mémoire est disponible ? Quel pourcentage de CPU est utilisé (attendre 5 minutes de données, ou réduire la fenêtre à `[1m]`) ?
2. Quelle fraction du système de fichiers `/` reste libre ?
3. Que fait `rate(...[5m])`, et `avg by (instance)` ?

Étape 6 : Une règle d'alerte

Un seuil volontairement haut pour déclencher l'alerte :

Créer `/etc/prometheus/rules.yml` avec un éditeur lancé en root (*`sudo`*) :

```
groups:
  - name: noeud
    rules:
      - alert: DisqueRacinePresquePlein
        expr: node_filesystem_avail_bytes{mountpoint="/"}
              / node_filesystem_size_bytes{mountpoint="/"} < 0.95
        for: 30s
        labels:
          severity: warning
        annotations:
          summary: "Moins de 95 % libres sur / ({{ $labels.instance }})"
```

Déclarer ce fichier dans `/etc/prometheus/prometheus.yml`, avec un éditeur lancé en root (*`sudo`*), avant `scrape_configs:` :

```
rule_files:
  - "rules.yml"
```

```
$ sudo promtool check config /etc/prometheus/prometheus.yml
  sudo systemctl reload prometheus
```

Étape 7 : Observer l'alerte

Après une à deux minutes :

```
$ curl -s localhost:9091/api/v1/alerts | python3 -m json.tool | \
    grep -E 'alertname|state'
```

Questions :

1. Quels états l'alerte traverse-t-elle, et quel rôle joue `for: 30s` ?
2. Quel composant route ensuite l'alerte vers un mail ou Slack ?

> **Dans un cas réel**
>
> Le seuil serait `< 0.10` et `for: 10m`, comme dans le support.

Étape 8 : Grafana et le dashboard 1860

```
$ sudo dnf install -y grafana
sudo systemctl enable --now grafana-server
curl -s localhost:3000/api/health
sudo firewall-cmd --add-port=3000/tcp
```

Dans le navigateur, `http://<VM>:3000`, connexion `admin` / `admin` (changer le mot de passe) :

1. **Connections** puis **Add data source** : Prometheus, URL `http://localhost:9091`, **Save & test**.
2. **Dashboards** puis **New**, **Import**, ID **1860**, **Load**, choisir la source Prometheus, **Import** (le navigateur doit joindre `grafana.com`).

Questions :

1. Quel port Grafana utilise-t-il, et d'où vient son paquet ?
2. Quels panneaux le dashboard 1860 affiche-t-il sans qu'aucun ne soit construit ?

Étape 9 : Nettoyage et bilan

```
$ sudo systemctl disable --now grafana-server prometheus node_exporter
sudo systemctl disable --now cockpit.socket pmlogger pmcd
sudo dnf remove -y grafana prometheus node-exporter cockpit pcp-zeroconf
sudo rm -rf /var/lib/prometheus /var/lib/grafana
sudo rm -rf /etc/prometheus /etc/default/prometheus.rpmsave
```

Questions

1. Quelle différence de principe entre Cockpit + PCP et la pile Prometheus ?
2. Qu'est-ce qui distingue la collecte par *pull* de Prometheus de l'ancien poll SNMP de Cacti ?