



Débugger les rongeurs de ressources...

Le 20 février 2008 à 13:07.

Non pas que je sois un malade de vitesse, loin de là, mais il n'y a rien de plus agaçant que d'avoir son système qui répond avec un quart de seconde de retard sans en comprendre la cause. Et comme ce dont on dispose le plus facilement sur n'importe quel système *nix est une console, voici une batterie d'outils en mode texte pour débuser les rongeurs de ressources.

En mode texte donc, des trucs très "low-tech" sans icônes, ni menus déroulant, ni interfaces web-ajax-2.0-machin-bidule. Ils se lancent en tout simplicité par une commande dans un terminal, en local ou à distance avec SSH. La majorité de ces utilitaires sont disponibles avec toutes les distributions et, lorsque ce n'est pas le cas, est indiqué l'URL où le télécharger. Enfin texte ne voulant pas dire anti-ergonomique, nombre d'entre elles utilisent la librairie `ncurses` permettant un contrôle au clavier, la colorisation et même l'utilisation de la souris.

les top's

Les outils de la famille "top" proposent généralement l'affichage d'un tableau plus ou moins dense, coloré ou interactif composé d'une liste d'items (ip, processus, etc) triés selon le critère que l'on cherche à étudier. La majorité d'entre eux disposent en outre d'un résumé de consommation globale.

Mémoire et CPU

La commande `top` est aussi répandue sur les systèmes *NIX que `vi`. Vous pouvez toujours compter sur elle car elle sera toujours installée sur une machine, quelle que soit la distribution utilisée.

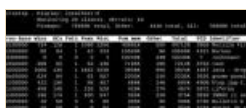
Cet outil vous permet d'afficher la liste des processus qui tournent sur une machine, classée par consommation de CPU. Cela permet en un coup d'oeil de voir quels sont les processus fous. La touche `?` permet d'avoir accès aux nombreuses options de tri, de choix de colonne, etc. Une commande à maîtriser donc (comme `vi`) 😊.



Mais bien mieux que `top`, nous avons `htop`. Il faut bien sûr l'installer mais cela vaut le coup car cet outil rajoute aux fonctions de son ancêtre, la couleur et l'interactivité grâce à `ncurses`. Vous pouvez donc sélectionner les processus, les tuer (touche `k`) les grouper par thread, etc. En pressant la touche `p`, l'ensemble des processus sera trié par leur charge machine et avec `m` ce sera par mémoire consommée.

Pour finir, un petit passage sur un top que j'aime beaucoup mais qui n'est utilisable que sur les kernels récents : `power top`. Développé par Intel pour optimiser la vie des batteries des ordinateurs portables, il permet de savoir quels sont les processus qui consomment le plus d'énergie en se basant sur divers paramètres dont la tendance qu'ils ont à réveiller le kernel. Et cerise sur le gâteau, ce top là a l'intelligence de vous fournir des conseils pour optimiser la consommation en fonction de ce qu'il constate. Un must pour qui est sensibilisé au "green computing".

X-Windows



Pour X-Windows, l'outil obligatoire est `xrestop`. Il vous permet d'auditer en temps réel les applications qui consomment le plus de ressources graphique.

Alors je vous préviens, cet outil est très mauvais pour le moral. Vous y découvrez par exemple que ce qui mange le plus de ressource peut être votre fond d'écran, la zoolie transparence dans votre console, ou plus classiquement ce maudit Firefox...

La première fois que j'ai testé l'outil la consommation mémoire du panda boulimique atteignait 1go !! Et, Ô surprise, même fermer tous les onglets n'y a rien changé. La seule "solution" a été de redémarrer le navigateur pour revenir à une consommation "normale" et ce avec EXACTEMENT les mêmes onglets ouverts sur les mêmes pages...

Le réseau



Pour auditer les mangeurs de bande passante nous disposons de plusieurs outils. Le plus simple à utiliser est `jnettop` qui affiche un classement des IP par KB/S reçus ou émis. Comme quasiment tous les outils "top" dédiés au réseau, `jnettop` utilise la librairie `libpcap`. Un outil équivalent mais moins précis est `iftop`.

Pour ceux qui ont besoin d'une analyse plus fine, l'outil `iptraf` fournit une interface un peu complexe en `ncurses` avec la possibilité de créer des fichiers de logs des interfaces auditées.

Un peu simpliste `nethogs` permet lui aussi d'afficher les consommations réseau mais a la particularité de faire cela par processus, permettant ainsi de savoir précisément quelle application s'amuse à sucer la bande passante.

Plus spécialisé, il existe aussi `apachetop` qui, à partir des logs du serveur, permet d'observer en temps réel la

consommation en bande passante des requêtes faites sur un site.

Plus spécialisé encore, et comme `nethogs` ou `iptraf`, basé sur la librairie `libpcap`, nous avons `dnstop` dédié aux propriétaires d'un serveur `bind` et permettant d'auditer la consommation exprimée en % du nombre total de requêtes par classe d'IP, domaine, sous domaine, etc.

Base de données

Allez, pour l'occasion, on va dire que `mySql` est une base de donnée, qui en l'occurrence dispose d'un outil bien pratique pour en auditer la consommation par processus. Il s'agit de `mytop` qui dispose d'une interface `ncurses` permettant de savoir quel utilisateur met la base à genoux.

PostgreSQL est quant à lui plus à la traîne en ce domaine. Rien de natif et juste un projet en python non abouti et un autre en perl ^[1] que je n'ai pas encore testé.

Les traceurs

Une autre famille d'outil bien pratique pour débusquer les mangeurs de ressources est les « traceurs ». Les traceurs permettent d'obtenir une liste d'actions menées par un ou plusieurs processus ou ressources physiques. Cela donne généralement lieu à des "logs" qu'il faut ensuite analyser.

Exécution des programmes

Pour commencer, nous avons le peu connu et néanmoins indispensable `strace` qui vous permet de lister en temps réel les appels système fait par une commande lors de son exécution. Cela permet typiquement de comprendre pourquoi la maudite commande `truc` bloque sans que l'on sache pourquoi. Cela permet aussi de savoir quels sont les fichiers que la commande `truc` cherche à ouvrir. Bref, essayez là, c'est très explicite.

Mais pour passer à la vitesse au dessus, le mieux est `oprofile` ^[2]. En effet, à mis chemin entre un traceur et un `top`, cet outil permet d'auditer Linux dans sa globalité et de savoir quel est le temps passé dans chacune des librairies du système. Grâce à lui vous pouvez par exemple constater que c'est la librairie `flash` qui est en train de consommer 60% de votre CPU alors qu'un simple `top` ne vous montrait que la boulimie de Firefox.

`oprofile` est composé de deux éléments : un démon (`oprofiled`) et un outil pour récupérer les statistiques (`opreport`). Vous pouvez sans aucune modification du noyau lancer le démon avec l'option `--no-vmlinux`. Mais pour obtenir une granularité descendant au niveau kernel, il faut que vous ayez compilé vous-même votre noyau ^[3] pour le fournir à l'outil sous la forme d'un fichier `vmlinux` décompressé et doté de symboles. Vous remplacerez alors l'option `--no-vmlinux` par `--vmlinux=/chemin/vers/vmlinux`.

Trafic réseau

De la même manière que pour les appels systèmes, il est possible de tracer les actions faites sur le réseau. Tout d'abord avec l'excellent `tcpdump` qui comme les "*top réseau*" utilise `libpcap`, mais cette fois pour tracer chaque requêtes faites sur un LAN. Son utilisation est extrêmement simple et la syntaxe des règles s'inspire fortement de son grand frère graphique `wireshark`. Par exemple pour lister l'ensemble des requêtes faites sur l'interface `eth0` en TCP sur le port 80 :

```
|| tcpdump -i eth1 tcp port 80
```

Plus spécifique au protocole HTTP, nous avons `httpdebug` ^[1] qui faisant office de proxy ^[4] et vous permet de tracer en temps réel l'ensemble des requêtes HTTP, et ce en clair. Idéal pour débusquer les pages anodines qui en réalité fliquent vos usages...

<http://artisan.karma-lab.net/node/1209>
(C) artisan numerique - CC BY-SA

Liens:

[1] <http://artisan.karma-lab.net/node/1209>

[2] <http://oprofile.sourceforge.net/>

[3] <http://artisan.karma-lab.net/node/18>

[4] <http://fr.wikipedia.org/wiki/proxy>