Introduction à la virtualisation en DUT R&T : retour d'expérience

Camille Coti
coti@lipn.fr
IUT de Villetaneuse, Université Paris 13

***Résumé*—Le nouveau Programme Pédagogique National de DUT R&T inclut une partie sur la virtualisation et le cloud, dans un module d'administration système. Le but de cette article est de présenter l'approche qui a été choisie à l'IUT de Villetaneuse dans la mise en place de ce module particulièrement ambitieux compte tenu de ses attentes et des compétences des étudiants en début de module compte tenu de son positionnement relativement précoce dans la formation.**

I. Introduction

Le nouveau Programme Pédagogique National de DUT R&T inclut une partie sur la virtualisation et le cloud, dans un module d'administration système portant le code M2102.

Le but de cette article est de présenter l'approche qui a été choisie à l'IUT de Villetaneuse dans la mise en place de ce module particulièrement ambitieux compte tenu de ses attentes et des compétences des étudiants en début de module compte tenu de son positionnement relativement précoce dans la formation.

Ce module est particulièrement ambitieux, notamment parce qu'il apparaît en début de formation. Dans notre cas, il est placé en fin de deuxième semestre. Les étudiants ont alors un recul limité sur les systèmes qu'ils manipulent et une expérience courte dans l'utilisation des systèmes Unix. Cependant, ce module peut également être utilisé, à l'inverse, comme un outil de compréhension de ce qu'est un système d'exploitation.

Le module M2102 peut être vu en deux parties. La première présente des notions d'administration système. La deuxième est consacrée à un pan entier du module qui est actuellement très demandé auprès de nos étudiants en stage : la virtualisation.

Cet article se concentre sur cette deuxième partie, et en particulier sur une façon dont ont été amenées les notions de virtualisation dans notre formation et comment celles-ci peuvent, de façon concomitante, aider les étudiants à prendre du recul et mieux appréhender l'architecture d'un système d'exploitation.

Tout d'abord, cet article part des connaissances des étudiants en arrivant à cette partie du module M2102 dans la section II. Nous présentons ensuite la façon dont les notions nécessaires à la présentation des techniques de virtualisation sont amenées dans la section III, avec notamment des définitions de ces techniques dans la partie III-A, la notion d'arborescence du système de fichiers dans la partie III-B et celle de noyau du système d'exploitation dans la partie III-C. nous présentons ensuite des exemples de mise en pratique dans la section IV. Enfin, nous présentons la façon dont le Cloud est amenée entre autres en tant qu'application possible dans la section V, mais aussi comme un modèle économique et une externalisation des ressources.

II. Pré-requis

Les étudiants commençant ce module utilisent un système Unix (Linux) en ligne de commande depuis un semestre et demi. Ils sont alors raisonnablement familiarisés avec les commandes de bases et des notions comme l'arborescence de fichiers. Ils ont en outre l'habitude d'utiliser le compte de super-utilisateur quand nécessaire, ou de s'octroyer temporairement les droits idoines (`sudo`).

Cependant, ils ont à ce moment une conscience limitée de l'architecture logicielle des équipements informatiques. Ils n'ont notamment pas encore entendu parler de noyau, de pilotes, de l'ordonnanceur, de la séparation entre l'espace noyau et l'espace utilisateur.

III. Amener les notions de virtualisation

Introduire la virtualisation et ses concepts repose sur un ensemble de notions d'architecture des systèmes d'exploitation. Il faut les amener un par un pour donner aux étudiants les briques de base pour appréhender ces concepts.

Ces concepts peuvent être introduits l'un après l'autre et illustrés directement par un exemple d'application dans un système de virtualisation ou d'isolation.

Ainsi, en replaçant la virtualisation dans le système d'exploitation, ce module permet d'aborder de façon progressive à la fois les principes de la virtualisation et des concepts fondamentaux d'architecture des systèmes d'exploitation.

A. Définitions et distinctions

Il est nécessaire d'apporter quelques définitions précises aux étudiants, pour que les étudiants connaissent les contours des concepts évoqués et sachent les distinguer :

- L'*émulation* consiste à faire passer une architecture matérielle pour une autre du point de vue de l'application qui s'exécute dessus. Concrètement, cela revient à permettre à un code binaire censé s'exécuter sur une machine A de s'exécuter sur une machine B.
- La *virtualisation* permet d'exécuter plusieurs machines sur une machine physique. Elle ne fait pas passer un matériel pour un autre.

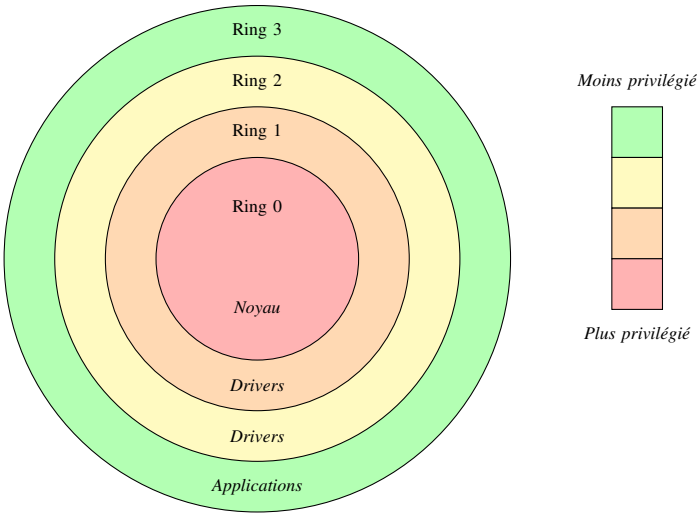
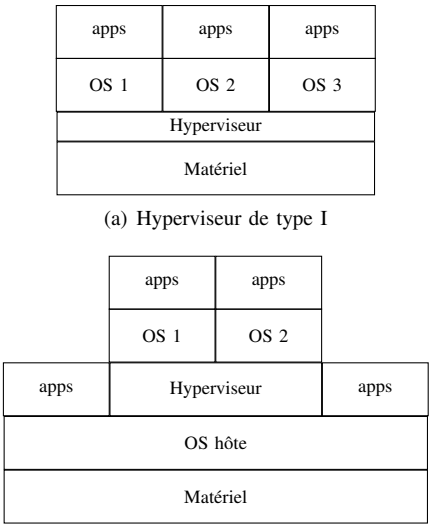


FIGURE 1. Inclusion des anneaux de privilèges



(a) Hyperviseur de type I

(b) Hyperviseur de type II

FIGURE 2. Deux types d'hyperviseurs : I et II.

– L'*isolation* permet de ne donner à une application qu'une vision limitée de l'arborescence du système de fichiers. Ainsi, sur le disque dur, l'application est confinée des autres applications.

D'après [3], la virtualisation permet de "faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes". Les techniques disponibles sont alors classifiées dans quatre familles: les isolateurs, les noyaux en espace utilisateur, les machines virtuelles et les para-virtualiseurs ou hyperviseurs.

B. Système de fichiers et arborescence

La première façon d'isoler un système, et la plus rudimentaire, est l'utilisation de `chroot`. En plus d'un premier outil de sécurisation (important mais cependant très limité et peu fiable), il permet d'aborder une première façon d'isoler un système tout en illustrant la notion d'arborescence de fichiers et de manipuler cette arborescence.

Comme vu à la section III-A, c'est un isolateur, c'est-à-dire un outil permettant d'isoler un processus du reste du système, mais qui n'exécute pas de système d'exploitation invité.

Un certain temps est consacré à la notion de *racine du système de fichiers*, notamment en insistant sur le fait que, par définition, on ne puisse pas remonter plus haut que la racine. Les illustrations de sous-arbre et de manipulations dans l'arborescence pour en déplacer la racine telle qu'elle est vue par un processus permettent de faire travailler les étudiants sur la représentation arborescente des systèmes de fichiers.

C. Noyau du système d'exploitation

Les notions de virtualisation sont amenées par des définitions et des notions de base sur les systèmes d'exploitation. Cette partie du module commence donc, non pas sur de la virtualisation à proprement parler, mais par des notions de système. L'orchestration de l'accès aux ressources matérielles est une notion prépondérante ici.

Sont alors introduits les concepts de *noyau*, d'*ordonnanceur* et de *gestion de la mémoire*. Enfin, la notion d'espace noyau et d'espace utilisateur est présentée avec la distinction entre les différents anneaux de privilèges (figure 1).

À partir de ces notions, le principe important à retenir est le fait qu'on ne peut exécuter qu'un seul noyau en espace noyau, et que l'on doit impérativement y exécuter un noyau. La distinction est alors faite entre système hôte et système invité. Les systèmes invités sont, du point de vue du système hôte, de simples applications.

Les ressources de la machine (mémoire, CPU...) sont gérées par le système hôte. Les systèmes invités accèdent à ces ressources ; lorsqu'elles ont besoin d'en obtenir, elles en font la demande au système hôte. Ainsi, la gestion de ces ressources reste centralisée auprès d'un seul noyau : celui du système hôte.

Cette distinction est indispensable pour ne pas créer de confusion chez les étudiants : si on exécute plusieurs systèmes d'exploitation sur la machine, un seul a accès à l'espace noyau. Cette idée doit impérativement être claire dans l'esprit des étudiants : d'après les évaluations en fin de module, il semble qu'elle soit bien passée chez la majorité d'entre eux.

Cette distinction permet, de manière générale, d'introduire et d'illustrer des notions de système d'exploitation qui peuvent être utiles aux étudiants pour mieux comprendre et appréhender le système :

– La notion de noyau, qui n'a pas été vue avant dans le déroulement de la première année, et son rôle ;

– Les anneaux de protection, et la distinction espace noyau contre espace utilisateur ;

– La gestion de l'accès mémoire, et en particulier à quelles zones de mémoire ont accès les processus (en expliquant au passage ce qu'est une erreur de segmentation).

En affinant la notion de système hôte, on arrive à la notion d'hyperviseur, et à la distinction entre hyperviseur de type I

```
root@arthas:~ # ./linux-3.11.0-uml ubda=disk.img ubdb=swap.img \
       root=/dev/ubda mem=256M
```

FIGURE 3. Commande de lancement d'un noyau UML en espace utilisateur.

```
 --gnome-terminal---bash---marionnet-------linux-2.6.18-gh--|----15*[linux-2.6.18-gh]
                                          |                  |----xterm-----port-helper
                                          |--2*[vde_switch]
                                          |--wirefilter
                                          |-10*[{marionnet}]
```

FIGURE 4. Extrait du résultat de la commande `pstree` avec un noyau exécuté en espace utilisateur lancé par Marionnet.

```
coti@arthas:~$ sudo xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  1962     2     r-----   1932.1
clientXen.domainelocal.com                   5   256     1     --p---     15.0
serveurXen.domainelocal.com                  7   256     1     -b----     20.6
```

FIGURE 5. Observation de la liste des machines virtuelles et de leur état avec Xen.

(ou *baremetal*) et de type II. Les piles logicielles du système pour ces deux types d'hyperviseurs sont représentées par les figures 2(a) et 2(b).

Un *hyperviseur* est un noyau hôte allégé dont le seul rôle est d'exécuter des noyaux invités. Il existe deux types d'hyperviseurs :

- Hyperviseur de *type I*, ou natif, ou bare metal : il est exécuté directement sur le matériel de la machine. Il s'agit d'une fine couche logicielle située entre le matériel et le système. Tous les noyaux s'exécutant dessus sont des noyaux invités.
- Hyperviseur de *type II*, ou hébergé : il se situe entre le système d'exploitation hôte et les systèmes invités.

Dans le cas d'un hyperviseur de type I, l'hyperviseur est en réalité un noyau allégé qui s'exécute en espace noyau ; les noyaux invités s'exécutent tous en espace utilisateur. C'est l'hyperviseur qui a la charge des tâches fondamentales du noyau, comme l'ordonnancement des processus et la gestion de la mémoire.

Dans le cas d'un hyperviseur de type II, c'est le noyau hôte qui a cette tâche. L'hyperviseur sert lui à faire tourner les noyaux invités.

IV. Mise en place pratique

A. Isolation

La première mise en place pratique est l'isolation d'un système avec `chroot`. Les manipulations qui vont avec permettent aux étudiants de se confronter de façon concrète à la notion de racine du système de fichiers.

De plus, la mise en place d'un environnement chrooté minimal leur permet de prendre conscience de ce dont a besoin un programme pour s'exécuter : exécutable, bibliothèques dynamiques.

On peut citer d'autres isolateurs, comme BSD Jail, VServer et OpenVZ.

B. Machine virtuelle

La prise de contact la plus facile d'abord pour les étudiants est l'utilisation d'un système de virtualisation disposant d'une interface graphique, comme par exemple VirtualBox. Ce système permet en quelques clics de créer une machine virtuelle. En introduction, il permet de replacer les notions vues en cours et d'observer le comportement du système invité (pas d'accès au système hôte, notamment).

C. Noyau en espace utilisateur

L'exécution d'un noyau en espace utilisateur avec User-Mode Linux est une application directe de la notion de noyau exécuté comme n'importe quel programme en espace utilisateur. La compilation d'un noyau UML et son démarrage dans un environnement chrooté permet aux étudiants de prendre conscience de ce qu'est un noyau et de quoi il a besoin pour s'exécuter.

On lance un noyau en espace utilisateur spécialement compilé en lui passant comme paramètres les chemins vers des images de systèmes de fichier (au moins le système de fichiers racine, éventuellement un espace de swap). On lance bien le noyau comme on lancerait n'importe quel programme, comme indiqué à la figure 3.

De plus, l'utilisation d'outils comme `pstree` permet aux étudiants de visualiser l'exécution de leur noyau comme n'importe quel autre processus.

Par exemple, la plate-forme Marionnet [4], qu'ils ont l'habitude d'utiliser en réseaux, lance des noyaux UML, qui eux-mêmes lancent des processus comme un terminal. On peut voir un extrait de la sortie de cette commande à la figure 4.

D. Hyperviseur

La notion d'hyperviseur peut être directement illustrée avec Xen, qui est un hyperviseur de type I. L'hyperviseur de Xen fait une distinction entre le domaine 0, ou domaine privilégié, qui a accès à toutes les ressources de la machine, et le domaine non-privilégié, ou domU, dans lequel s'exécutent des

machines virtuelles. Cependant, les noyaux du domU doivent être compilés spécialement pour Xen. Son intégration dans le noyau Linux le rend particulièrement facile à mettre en œuvre dans le cadre d'une séance de TP.

Xen fournit un ensemble d'outils de gestion de ces machines virtuelles. Celles-ci peuvent être lancées, mises en pause, observées, sauvegardées ou checkpointées pour les relancer sur une autre machine...

L'ensemble d'utilitaires est très complet et rend la manipulation des machines virtuelles très facile. Cependant, c'est une approche complémentaire d'UML mais où l'étudiant voit moins ce qui se passe. Les utilitaires fournis apportent de la facilité, mais cette facilité abstrait l'observation et les manipulations de base. Par conséquent, Xen peut s'inscrire dans le cadre d'une séance de travaux pratiques qui en suit une autre portant sur UML, de façon complémentaire.

On peut voir par exemple les informations données par la commance `xm list` sur la figure 5. On voit alors trois machines virtuelles :

- Domaine-0, qui est la machine virtuelle lancée au démarrage, dans laquelle l'utilisateur est connecté ;
- clientXen.domainelocal.com et serveurXen.domainelocal.com, qui sont deux machines en domU, la première étant en pause.

De plus, Xen permet relativement facilement de mettre en place un réseau virtuel entre les machines. Ce réseau peut utiliser un pont ou un NAT. Outre cette notion de réseau virtuel, il permet donc de réutiliser des notions de réseau vues précédemment.

Sous Linux, on peut également citer KVM, qui est un hyperviseur de type 1. Ce dernier est intégré au noyau Linux : ce dernier en fait un "gros hyperviseur". Le noyau hôte est l'hyperviseur, et il fait tourner des noyaux invités.

À l'inverse, VirtualBox, QEMU et VMware Workstation sont des hyperviseurs de type II : ils tournent à l'intérieur du système d'exploitation hôte.

E. Autres possibilités

D'autres exemples peuvent être imaginés, comme l'utilisation des Linux Containers (`lxc`) pour illustrer le cloisonnement entre les containers, ou qemu, pour illustrer l'émulation.

Les TP se sont déroulés d'une façon étonnamment facile, les étudiants prenant de façon "naturelle" le fait que le système invité soit exécuté d'une façon un peu particulière du fait de son exécution en espace utilisateur. Cette facilité est à moduler avec l'aisance particulière des étudiants de l'année 2013-2014 avec la ligne de commande, qui leur a permis notamment de compiler un noyau Linux avec les outils Debian sans difficulté particulière.

V. Le cloud

Le Programme Pédagogique National spécifie que ce module doit également contenir une introduction au cloud, ou informatique dans le nuage. Le cloud peut être vu de deux façons : c'est à la fois un système intrinsèquement multi-utilisateurs, en milieu hostile, et nécessitant ainsi une forte isolation entre les données ou les processus appartenant à chaque utilisateur, mais également une forme particulière de ressources informatiques, externalisées.

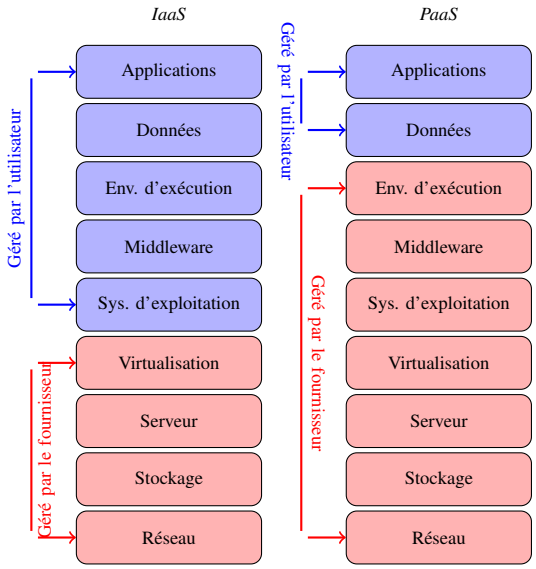


FIGURE 6. Piles correspondant à deux types de services de cloud.

La difficulté de la présentation de l'informatique dans le nuage réside dans l'inondation marketing résidant autour de ce qui est présenté comme révolutionnant l'informatique et résolvant tous les problèmes aujourd'hui. Les étudiants ont entendu parler de ce qu'ils croient être le cloud et ont tendance à confondre beaucoup d'outils avec "le cloud".

Le cloud est parfois réduit dans l'imaginaire non-technique aux services de stockage en réseau, comme DropBox. Certains assimilent le cloud à la virtualisation.

L'idée a été de présenter le cloud comme un service externalisé et un modèle économique associé reposant sur le *Service Level Agreement*. La notion de fiabilité de service dans le cloud [1], pouvant être généralisée aux équipements réseaux en général (liaisons louées par exemple), est ainsi introduite.

À partir des idées reçues des étudiants sur le cloud, on peut présenter à quel type de service de cloud ces exemples correspondent : *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) ou *Service as a Service* (SaaS). Ces types de service sont souvent représentés avec la pile de composants matériels et logiciels d'un service en distinguant ce qui est pris en charge par le prestataire et ce qui l'est par le client (deux exemples sont représentés figure 6).

Enfin, il est important de distinguer le cloud de la virtualisation : si certains services de cloud "historiques" comme Amazon Elastic Compute Cloud (EC2) reposent sur de la virtualisation pour fournir à l'utilisateur l'environnement de son choix, certains services sont accessibles depuis le navigateur (les Google Docs peuvent être considérés comme un service de cloud de la sorte) et certains proposent des piles logicielles complètes sans reposer sur aucune couche de virtualisation (comme SlapOS [2]).

Ces distinctions permettent de faire du tri dans l'esprit des étudiants et de clarifier ce qui est derrière des mots

dont ils entendent parler au quotidien sans réellement savoir concrètement à quoi ils correspondent.

VI. Conclusion

Cet article a présenté l'approche suivie à l'IUT de Villetaneuse pour introduire les notions de virtualisation en première année de DUT R&T dans la deuxième partie du module d'administration système M2102 du PPN 2013. Le parti pris a été de commencer par présenter les notions d'architecture des systèmes d'exploitations nécessaires à une définition rigoureuse des mécanismes de virtualisation.

Cette approche tente à la fois d'apporter une vision claire de ce qu'est la virtualisation tout en apportant aux étudiants une compréhension et un recul sur le système qu'ils appréhendent, à travers des concepts comme celui de noyau du système d'exploitation, d'anneau de protection, de gestion de la mémoire...

L'objectif global de cette partie de module est finalement assez modeste: il est de faire prendre conscience aux étudiants qu'on exécute exactement un système d'exploitation en espace noyau, d'éventuels systèmes invités étant exécutés en espace utilisateur comme n'importe quelle application. D'après les évaluations, cette idée semble être bien passée chez une grande majorité d'étudiants.

Enfin, un prolongement des notions d'isolation des données et des processus est présentée à travers les principes du cloud computing. Le but a été ici de clarifier la définition du cloud et de le présenter à la fois d'un point de vue technique, mais aussi économique à travers le service rendu, et organisationnel avec l'externalisation des ressources.

Références

[1] Maurice Gagnaire, Felipe Dia, Camille Coti, Christophe Cerin, Kazuhiko Shiozaki, Yingjie Xu, Pierre Delort, Jean-Paul Smets, Jonathan Le Lous, Stephen Lubiarz, Pierrick Leclerc: *Downtime statistics of current cloud solutions*, International Working Group on Cloud Computing Resiliency, Tech. Rep, 2012.

[2] Jean-Paul Smets-Solanes, Christophe Cerin, Romain Courteaud: *Slapos: A multi-purpose distributed cloud operating system based on an erp billing model*. In Proceedings of the 2011 IEEE International Conference on Services Computing (SCC), pp. 765-766. IEEE.

[3] Benjamin Quétier: *EmuGrid : études de mécanismes de virtualisation pour l'émulation conforme de grilles à grande échelle*. Thèse de doctorat en informatique, Université Paris Sud-XI, 115p, 2008.

[4] Jean-Vincent Loddo: *Marionnet : un logiciel graphique pour l'apprentissage et l'enseignement des réseaux locaux d'ordinateurs*, Premier Workshop pédagogique "Réseaux & Télécoms", Saint-Pierre de la Réunion (France), 2007.