**Virtualisation de systèmes d'exploitation**

La **virtualisation de systèmes d'exploitation** est une technique consistant à faire fonctionner en même temps, sur un seul ordinateur, plusieurs systèmes d'exploitation comme s'ils fonctionnaient sur des ordinateurs distincts.

La virtualisation de systèmes d'exploitation a plusieurs intérêts :

* Utiliser un autre système d'exploitation sans redémarrer son ordinateur, afin d'utiliser des programmes ne fonctionnant pas nativement dans Ubuntu ;
* Exploiter des périphériques ne fonctionnant pas dans Ubuntu mais fonctionnant dans d'autres systèmes d'exploitation ;
* Tester des systèmes d'exploitation en cours de développement sans compromettre un environnement quotidien stable ;
* Tester des logiciels dans des environnements contrôlés, isolés et sécurisés ;
* Transporter ses systèmes d'exploitation d'un ordinateur à l'autre, une machine virtuelle fonctionnant sur n'importe quel ordinateur disposant d'un hyperviseur compatible.

Les particuliers et les PME/PMI seront généralement plus intéressés par la perspective de faire fonctionner deux systèmes d'exploitation différents en même temps, afin d'exécuter des logiciels qui sont compatibles avec l'un mais pas avec l'autre. Les grandes entreprises, elles, ont de plus en plus recours à la virtualisation afin de gagner de la place dans les salles de serveurs, faciliter les installations et les redémarrages après incidents, et développer et sécuriser les réseaux d'entreprises.

**Principe général et terminologie**

Plusieurs types de virtualisation existent, mais tous fonctionnent selon un même principe :

1. Un système d'exploitation principal (appelé *système d'exploitation hôte*) est installé dans l'ordinateur et sert de système d'accueil à d'autres systèmes d'exploitation ;
2. Dans le système d'exploitation hôte, un logiciel de virtualisation (appelé *hyperviseur*) est installé. Celui-ci crée des environnements clos, isolés, avec des ressources bien précises : ces environnements clos sont appelées des *machines virtuelles* ;
3. D'autres systèmes d'exploitation (appelés *systèmes d'exploitation invités*) peuvent alors être installés dans des machines virtuelles. Leur instance est totalement isolée du système hôte et des autres systèmes invités.

En bref, la virtualisation est une méthode faisant fonctionner un ou des systèmes d'exploitation invités dans des machines virtuelles, au-dessus d'un système d'exploitation hôte.

**Virtualisation complète : un choix privilégié chez le particulier**

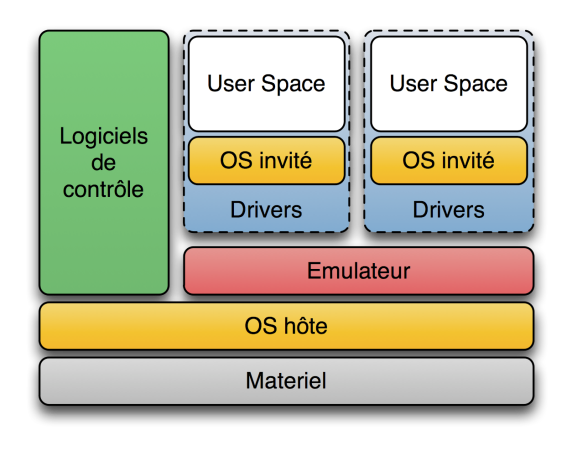


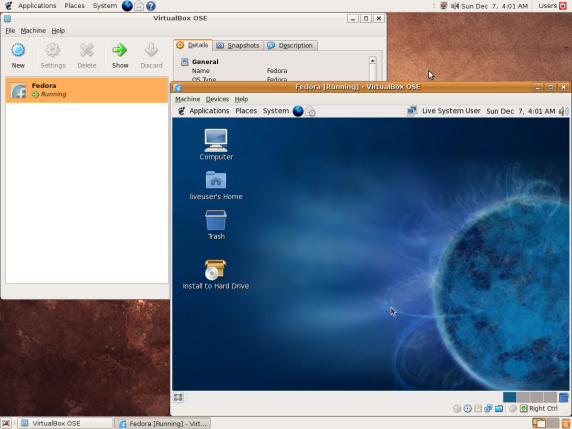
Schéma classique

La virtualisation dite *complète* permet de faire fonctionner n'importe quel système d'exploitation en tant qu'invité dans une machine virtuelle. Pour l'utilisateur final, ce type de virtualisation est la plus simple à mettre en place et est la plus pratique.

* **Principe :** L'hyperviseur1) crée un environnement virtuel complet simulant littéralement un nouvel ordinateur complet, avec du "faux matériel". À quelques rares exceptions, le système d'exploitation invité (installé dans la machine virtuelle) ne communique qu'avec ce faux matériel simulé, rendant étanche l'environnement virtualisé.
* **Limitations :** Ce type de virtualisation ne permet de virtualiser que des systèmes d'exploitation prévus pour la même architecture matérielle que le processeur physique de l'ordinateur hôte. Par exemple, un ordinateur équipé d'un processeur Intel x86 sera incapable de virtualiser un système d'exploitation prévu pour fonctionner dans une architecture PowerPC.

**Quelques hyperviseurs de virtualisation complète :**

* **VirtualBox**
* Logiciels de virtualisation de VMWare : VMWare Player, VMWare Workstation, VMWare Server
* Parallels Desktop 4 for Windows & Linux
* KVM

**Virtualisation de systèmes d'exploitation 64 bits**

Pour être en mesure de virtualiser un système d'exploitation 64 bits, le microprocesseur de votre ordinateur hôte doit être doté d'une technologie appelée *virtualisation assistée matériellement* (ou *accélération matérielle*). Cette technologie est identifiée *AMD-V* chez AMD et *Intel VT-x* chez Intel. Généralement incluse dans les processeurs récents, la virtualisation matérielle doit toutefois bien souvent être activée dans le BIOS/UEFI de l'ordinateur et paramétrée convenablement dans les paramètres de la machine virtuelle. Tous les processeurs ne sont pas capables de virtualisation matérielle. *Liste des processeurs gérant la virtualisation matérielle*

Si votre microprocesseur est listé dans la liste précédente et vous souhaitez savoir si la virtualisation assistée matériellement est activée dans les propriétés de votre BIOS/UEFI, ou si votre microprocesseur n'y est pas listé et vous souhaitez savoir si cette technologie y est présente, exécutez la commande suivante dans un terminal :

egrep '(vmx|svm)' /proc/cpuinfo

Cette commande vous retourne les fonctionnalités de votre microprocesseur et met en évidence les fonctionnalités de virtualisation assistée matériellement. Lorsque les termes **svm** ou **vmx** sont mis en évidence dans le retour de la commande, votre microprocesseur dispose de la fonctionnalité de virtualisation assistée matériellement; il peut donc virtualiser des systèmes d'exploitation 64 bits. Dans le cas contraire, vous ne pouvez virtualiser que des systèmes d'exploitation 32 bits.

**Aller plus loin...**

La virtualisation complète n'est qu'un seul type de virtualisation. Celui-ci est généralement plutôt adapté pour une utilisation par un particulier ou par une PME/PMI, afin de faire fonctionner un environnement de travail pour y faire fonctionner certains logiciels particuliers non gérés par le système d'exploitation hôte. Mais de nombreux autres types de virtualisation existent, bien plus efficaces pour accomplir des tâches précises. Ci-dessous sont décrits très succinctement certains autres types de virtualisation, avec quelques exemples d'hyperviseurs déployant ceux-ci.



Logo de la firme Vmware

**Émulation**

L'hyperviseur crée un ordinateur virtuel simulé complet. La différence entre l'émulation et la virtualisation complète est la simulation d'un microprocesseur : en virtualisation complète, le processeur simulé par la machine virtuelle est nécessairement identique à celui de l'ordinateur hôte. En émulation, celui-ci peut être de type différent. Ceci permet d'émuler des architectures matérielles différentes (ordinateurs, consoles de jeux, bornes d'arcade…) au-dessus d'un système hôte. Les performances en émulation sont toutefois médiocres. (**Plus d'informations**)

* QEmu (sans accélérateur KQemu)
* Bochs

**Paravirtualisation**

Dans ce type de virtualisation, l'hyperviseur2) et le système d'exploitation invité coopèrent. En virtualisation complète, lorsque l'invité a besoin de communiquer avec du matériel, celui-ci n'a accès qu'à du faux matériel présenté par l'hyperviseur ; en paravirtualisation, l'hyperviseur capture les appels système de l'invité et les transmet au matériel. En somme, l'invité exploite directement le matériel de l'ordinateur hôte. La paravirtualisation offre des performances optimales, mais nécessite un système d'exploitation modifié, conscient qu'il fonctionne dans un environnement virtualisé. (**Plus d'informations**)

* Xen *(moniteurs de contrôle : virt-manager, enomalism)*
* kvm
* xVM

**Environnement Virtuel (Operating system-level virtualization)**

Chaque environnement exécuté possède son propre espace mémoire, afin d'être isolé des autres, les autres ressources systèmes sont partagées (pilotes, noyau…). Les instructions ne sont pas interceptées, et l'environnement exécuté doit impérativement être compatible avec celui du système hôte. (On ne peut pas, par exemple, exécuter Microsoft® Windows® dans un environnement virtuel au-dessus d'un hôte de type Linux). Ce type de virtualisation permet d'exécuter plusieurs fois le même environnement dans des zones mémoires séparées, mais les ressources partagées empêchent d'obtenir une isolation sûre entre les environnements invités.

* Linux-VServer
* OpenVZ
* LXC
* chroot

**Etudions OpenVZ**



Logo OpenVZ

**OpenVZ** est une technique de virtualisation de niveau système d'exploitation basée sur le noyau Linux. OpenVZ permet à un serveur physique d'exécuter de multiples instances de systèmes d'exploitation isolés, qualifiées de serveurs privés virtuels (VPS) ou environnements virtuels (VE).

Par rapport aux machines virtuelles telles que VMware et aux techniques de paravirtualisation telles que Xen, OpenVZ offre moins de flexibilité dans le choix du système d'exploitation : les systèmes d'exploitation invité et hôte doivent être de type Linux, bien que les distributions de Linux puissent être différentes dans des Environnements Virtuels différents. Cependant, la virtualisation au niveau du système d'exploitation choisie par OpenVZ offre une meilleure performance, une meilleure mise à l'échelle (i.e. capacité à monter en charge), une meilleure densité, une meilleure gestion dynamique des ressources, et une meilleure facilité d'administration que ses alternatives. Selon le site Web d'OpenVZ, cette méthode de virtualisation introduirait une très faible pénalité sur les performances : 1 à 3 % de pertes seulement par rapport à un ordinateur physique.

OpenVZ est la base de Virtuozzo, un produit (de) propriétaire fourni par SWsoft, Inc. OpenVZ est distribué sous la Licence publique générale GNU version 2.

OpenVZ comprend le noyau Linux et un jeu de commandes pour l'utilisateur.

## Noyau

Le noyau d'OpenVZ est un noyau Linux modifié qui ajoute la notion d'environnement virtuel. Le noyau fournit la virtualisation, l'isolement, la gestion de ressource, et les fonctions "checkpoint/restart".

### Virtualisation et isolement

Un Environnement Virtuel est une entité séparée, et du point de vue de son propriétaire, il ressemble à un vrai serveur physique. Il possède en propre :

Fichiers

Sont virtualisés : les bibliothèques système, applications, /proc et /sys, les verrous, etc.

Utilisateurs et groupes

Chaque Environnement Virtuel a son propre compte système "root", ainsi que ses propres utilisateurs et groupes.

Arborescence de processus

Chaque Environnement Virtuel voit seulement ses propres processus (à partir de l'init). Les Identifiants de processus (PID) sont virtualisés, de telle sorte que l'identifiant « PID » de la tâche init soit 1, conformément à l'habitude.

Réseau

L'interface réseau virtuelle permet à un Environnement Virtuel d'avoir ses propres adresses IP, aussi bien qu'un ensemble de règles pare-feu Netfilter (iptables) et de routage.

Périphériques

Si nécessaire, on peut accorder à n'importe quel Environnement Virtuel l'accès à de vrais périphériques tels que des interfaces réseau, des ports séries, des partitions de disque, etc.

IPC

Mémoire partagée, sémaphores, file d'attente de messages.

### Gestion des ressources

Comme tous les VEs utilisent le même noyau, la gestion des ressources est d'importance primordiale. Chaque VE doit rester dans ses limites et ne pas affecter d'autres VEs.

La gestion des ressources d'OpenVZ est composée des trois éléments suivants : des quotas disque à deux niveaux, un scheduler fair CPU, et des beancounters utilisateurs. Toutes ces ressources peuvent être modifiées pendant le temps d'exécution d'un VE, sans avoir à rebooter. Il est par exemple possible d'allouer à chaud plus de mémoire à un VE, en changeant des paramètres appropriés à la volée. C'est une fonctionnalité souvent impossible ou complexe avec d'autres approche de virtualisation telles que la VM ou l'hyperviseur.

#### Quotas disque à deux niveaux

L'utilisateur root du système hôte (OpenVZ) peut établir des quotas disque par VE, spécifiés en termes de bloc disque et d'inode (plus ou moins le nombre de fichiers). C'est le premier niveau des quotas disque. En plus de ceux-ci, un propriétaire de VE (utilisateur root) peut utiliser les outils habituels de gestions des quotas à l'intérieur de son propre VE pour placer des quotas disque standards par utilisateur et par groupe d'UNIX.

Il est possible d'allouer dynamiquement de l'espace disque à un VE, en modifiant simplement son quota disque (autrement dit, sans qu'il soit nécessaire de redimensionner les partitions du disque).

#### Scheduler Fair CPU

Le scheduler CPU d'OpenVZ est également à deux niveaux. Au premier niveau, le scheduler décide à quel VE le time slice sera affecté, basé sur des cpuunits par VE. Au deuxième niveau, le scheduler standard Linux décide quel processus s'exécutera dans le VE, en utilisant les priorités de processus standards.

L'administrateur d'OpenVZ peut établir différentes valeurs des cpuunits pour des VEs différent, et le temps CPU sera donné à ceux-ci proportionnellement.

Il est en outre possible de limiter le temps CPU, par exemple à 10 % de temps CPU disponible.

#### Beancounters Utilisateur

Les Beancounters utilisateur sont un ensemble de compteurs, de limites, et de garanties par VE. Environ 20 paramètres, soigneusement choisis, couvrent tous les aspects des opérations d'un VE ; ainsi, aucun VE ne peut abuser d'une ressource qui est limitée pour le nœud entier et de cette façon nuire à un autre VE.

Les ressources comptabilisées et contrôlées sont principalement la mémoire et divers objets noyau tels que les segments de mémoire partagés, les buffers réseau, etc. Chaque ressource peut être vue depuis /proc/user\_beancounters et cinq valeurs lui sont associées : utilisation courante, utilisation maximum (pour la vie d'un VE), barrière, limite, et compteur d'échec. La signification de la barrière et de la limite est dépendante du paramètre ; en bref, ceux-ci peuvent être considérés comme une limite soft et une limite hard. Si n'importe quelle ressource atteint la limite, le compteur d'échec est incrémenté, ainsi le propriétaire du VE peut voir si quelque chose de néfaste se produit en analysant la sortie de /proc/user\_beancounters dans son VE.

### Checkpointing et migration live

La fonctionnalité de migration live et de checkpoint pour OpenVZ a été annoncé au milieu d'avril 2006. Elle permet de migrer un VE d'un serveur physique à un autre sans arrêt/relance du VE. Ce processus est connu comme application checkpointing : le VE est gelé et son état entier est sauvegardé dans un fichier sur disque. Ce fichier peut alors être transféré sur une autre machine sur laquelle le VE pourra être restauré. Le délai de migration est de quelques secondes, et ce n'est pas un temps d'arrêt, juste un retard.

Chaque morceau d'état de VE, y compris les connexions de réseau ouvertes, est sauvé. De la perspective de l'utilisateur, cela ressemble à un retard dans la réponse : une transaction de base de données prendra un peu plus longtemps que d'habitude, quand le VE redémarre, l'utilisateur ne note pas que sa base de données fonctionne déjà sur une autre machine.

Cette fonctionnalité rend possible des scénarios tels que la mise à jour du serveur sans avoir besoin de le rebooter : si votre base de données a besoin de plus de mémoire ou de CPU, vous achetez un nouveau serveur, migrez votre VE dessus et vous augmentez ensuite ses limites. Si vous voulez ajouter plus de RAM à votre serveur, vous migrez tous les VEs sur un autre serveur, vous arrêtez le premier, ajoutez la mémoire, puis rebootez et remigrez tous les VEs.

## Commandes utilisateurs

OpenVZ vient avec un outil en ligne de commande pour contrôler les VEs (vzctl), ainsi que des outils pour installer les logiciels dans les VEs (vzpkg).

### vzctl

C'est un outil en ligne de commande simple pour contrôler un VE.

vzctl create VEID [--ostemplate <name>] [--config <name>]

Cette commande crée un nouvel environnement virtuel avec l'identification numérique de VEID, qui sera basé sur un template d'OS (un distro de Linux) et qui aura des paramètres de gestion de ressource pris dans le fichier de configuration spécifié en ligne de commande. Les deux paramètres --ostemplate et --config sont facultatifs, il existe des valeurs par défaut définies dans fichier global de configuration.

vzctl start VEID

Démarre un VE donné, ce qui signifie créer un contexte virtuel d'environnement dans le noyau, en plaçant tous les paramètres de gestion de ressource et démarrer le /sbin/init du VE dans ce contexte.

vzctl stop VEID

Arrête un VE donné. Un VE peut également être arrêté (ou rebooté) par son propriétaire en employant les commandes standards /sbin/halt ou /sbin/reboot.

vzctl exec VEID <command>

Exécute une commande à l'intérieur d'un VE. Pour voir la liste des processus à l'intérieur du VE 102, il suffit de lancer la commande vzctl exec 102 ps ax.

vzctl enter VEID

Ouvrir un shell VE. C'est utile si par exemple le sshd est mort dans ce VE et vous voulez le dépanner.

vzctl set VEID --parameter <value> [...] [--save]

Affecte un paramètre pour un VE. Il y a beaucoup de différents paramètres. Pour ajouter une adresse IP à un VE, lancez vzctl set VEID --ipadd x.x.x.x --save. Pour changer le quota disque d'un VE, lancez vzctl set VEID --diskspace soft:hard --save. Pour fixer la barrière et la limite de mémoire noyau d'un VE, lancez vzctl set VEID --kmemsize barrier:limit --save.

### Templates et vzpkg

Les templates (en français : gabarit) sont des images précréées utilisées pour créer un nouveau VE. Globalement, un template est un ensemble de packages, et un template cache est un tarball d'un environnement chrooté avec ces packages installés. Pendant l'étape de création vzctl, un tarball est déballé. En utilisant la technique template cache, un nouveau VE peut être créé en quelques secondes.

Les outils vzpkg sont un ensemble d'outils facilitant la création de template cache. Ils supportent actuellement rpm et les repositories de type yum. Ainsi, pour créer un template, par exemple, de distribution de Fedora Core 5, vous devez indiquer un ensemble repository (yum) qui ont les rpm FC5, et un ensemble de rpm à installer. De plus, des scripts de pre- et post-install peut être utilisés pour optimiser ou modifier un templace cache. Toutes les données ci-dessus (les repositories, listes de paquets, les scripts, clefs GPG etc.) forment un *template metadata*. Un template cache peut être créé automatiquement à partir d'un *template metadata*; vous lancez juste l'utilitaire vzpkgcache. le vzpkgcache téléchargera et installera les packages énumérés sur un VE provisoire, et emballera le résultat comme un template cache.

Des template cache pour les distros non-RPM peuvent aussi être créées, bien que ce soit plus un processus manuel. Par exemple, ce HOWTO donne des instructions détaillées pour créer un template cache de Debian.

Les template cache suivants (a.k.a. templates precréés) sont actuellement (mai 2006) disponibles :

* dans les dérivés de Red Hat
  + Fedora Core Version 5 (et les versions préccédentes : 3 et 4)
  + CentOS 4 (4.3)
* Gentoo 2006.0 (20060317)
* openSUSE 10
* Debian Sarge et Etch télécharger
* Ubuntu [1]

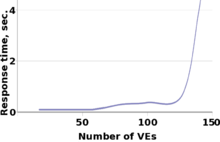
## Principales Fonctionnalités d'OpenVZ

### Scalabilité

Comme OpenVZ utilise un modèle de noyau unique, il se comporte comme le noyau 2.6 de Linux, ce qui signifie qu'il supporte jusqu'à 64 processeurs et jusqu'à 64 GigaOctets de mémoire vive. Un environnement virtuel unique peut être étendu jusqu'à la machine physique dans sa totalité, c'est-à-dire tous les CPU et toute la RAM.

En effet, certains déploient un unique environnement virtuel OpenVZ. C'est étrange à première vue, mais étant donné qu'un VE unique peut employer toutes les ressources matérielles avec une performance proche du natif, tout en bénéficiant d'autres avantages tels que l'indépendance du matériel, la gestion des ressources et la migration à chaud, ceci est un choix évident dans beaucoup de scénarios.

### Densité

[](http://commons.wikimedia.org/wiki/File:Openvz-density.png)

OpenVZ density on a 768Mb (¾Gb) RAM box

OpenVZ peut accueillir des centaines d'environnements virtuels sur un matériel décent (les limitations principales sont la quantité de mémoire vive et le nombre de processeurs).

Le graphique montre la relation du temps de réponse du serveur web Apache dans un VE sur le nombre de VEs. Les mesures ont été faites sur une machine avec 768 Mo (¾ Gb) de RAM ; chaque VE faisait tourner l'ensemble habituel de processus : init, syslogd, crond, sshd et apache. Les daemons d'Apache servaient des pages statiques, qui étaient chargées par le http\_load, et le premier temps de réponse était mesuré. Comme vous pouvez le voir, plus le nombre de VE se développe, plus le temps de réponse se dégrade en raison du manque de mémoire et du swap excessif.

Dans ce scénario, il est possible de faire tourner jusqu'à 120 VEs sur une machine avec ¾ Go de RAM. Ceci s'extrapole de façon linéaire, et il est donc possible de faire tourner jusqu'à près 320 VEs sur une machine avec 2 Gb de RAM.

### Gestion de masse

Un propriétaire (root) du serveur physique OpenVZ (également connu sous le nom de nœud matériel) peut voir tous les processus et les fichiers des VE. Ceci permet de faire de la gestion de masse. Ainsi, par exemple, vous pouvez faire tourner un simple script qui mettra à jour tous les (ou juste certains choisis) VEs immédiatement.

## Scénarios d'utilisation

Les scénarios suivants d'utilisation sont communs à toutes les techniques de virtualisation. Cependant, un avantage unique de la virtualisation de niveau OS comme OpenVZ est de ne pas trop dégrader les performances, ce qui rend ces scénarios plus attrayants.

Sécurité

La virtualisation permet d'isoler chaque service réseau (comme Apache, le serveur mail, le serveur DNS etc.) dans un environnement virtuel séparé. Dans ces conditions, si un intrus trouvait un trou de sécurité dans une des applications, il ne pourrait maltraiter que ce service même ; puisque tous les autres services sont dans des VEs séparés, il ne pourrait pas y accéder.

Consolidation de serveur

Actuellement, la plupart des serveurs sont peu employés. En utilisant OpenVZ, de tels serveurs peuvent être consolidés en les migrant dans des environnements virtuels. Le gain est dans l'espace pris par les racks, les factures d'électricité, et les coûts de gestion.

Hébergement

La virtualisation de niveau OS permet aux hébergeurs de proposer de l'hébergement virtualisé à bon marché. Chaque VE (donc éventuellement, chaque client) a un accès root total, ce qui signifie que le propriétaire du VE peut réinstaller n'importe quoi, et employer des outils telles que les tables IP de Linux (règles de pare-feu).

Développement et test

Les développeurs et testeurs ont souvent besoin d'accéder à un grand nombre de distribution Linux ; parfois ils doivent réinstaller ceux-ci à partir de zéro. Avec OpenVZ, ils peuvent installer rapidement (un nouveau VE peut être créé en une minute) toutes ces distributions sur un même ordinateur, sans avoir besoin de rebooter, en conservant des performances honorables. Le clonage d'un VE est également très simple : il suffit de copier la zone du VE et son fichier de configuration.

Éducatif

Chaque étudiant peut avoir un ou plusieurs VE. On peut jouer avec différentes distributions de Linux. Un nouveau VE peut être (re)créé en une minute.

## Techniques semblables

Il existe d'autres exemples de virtualisation au niveau OS tels que Linux-VServer, BSD Jails, et les Zones Solaris.

## Développements futurs

En mai 2006, le patch d'OpenVZ n'a pas été incorporé dans le noyau Linux standard. Puisqu'il y existe d'autres techniques, concurrentes, de virtualisation, il est difficile de déterminer précisément si, quand, et sous quelle forme les changements pourront être fusionnés. Il y a une discussion en cours sur LKML au sujet des différentes approches de virtualisation de niveau OS, des réalisations possibles, et de leur éventuelle inclusion.

Les développements en cours pour OpenVZ incluent :

* Stabiliser la branche de développements sur le noyau 2.6.17
* Ajouter le support d'IPv6 et bridged networking
* Ajouter l'affinité de VCPU pour VEs
* Ordonnanceur d'E/S de VE basé sur CFQ
* Continuer à maintenir les noyaux spécifiques aux distributions (SUSE10, FC5)
* Continuer à travailler sur la virtualisation dans le noyau standard