Club des Utilisateurs de Micro-ordinateurs dans l'Education

Stage Virtualisation Serveurs

Juin 2008

Xavier Montagutelli
Université de Limoges
Service Commun Informatique
xavier.montagutelli@unilim.fr

Hubert Chomette
Université de Limoges
École Nationale Supérieure d'Ingénieurs de Limoges
hubert.chomette@unilim.fr

Licence

Plan

- Pourquoi virtualiser ?
- Techniques de virtualisation
- Linux VServer
  - Architecture
  - Mise en œuvre
  - Retour d’expérience
- Conclusions & perspectives
- TP

Plan

- Pourquoi virtualiser ?
- Techniques de virtualisation
- Linux VServer
  - Architecture
  - Mise en œuvre
  - Retour d’expérience
- Conclusions & perspectives
- TP

Pourquoi virtualiser ? Etat des lieux

- Prolifération des serveurs
  - ↗ Quantité et traitements de données numériques
  - 1 application = 1 serveur : incompatibilités, facilité, sécurité, simplification de l'administration, contrainte des éditeurs
  - Tests et développements
  - Redondance
- ⇨ Salles techniques saturées, sous-dimensionnées (climatisation, protection électrique)
- ⇨ Matériel actif (réseau, SAN) ↗
- ⇨ Temps perdu en « logistique » : commande des machines, réception, installation, déploiement OS
- ⇨ Matériel pas homogène

Pourquoi virtualiser ? Etat des lieux (2)

- Machines de plus en plus puissante (en CPU, en mémoire)
- ⇨ Sous-utilisation

- Bilan : coûts financiers et humains élevés, et une perte de ressources
- La virtualisation : un outil de **consolidation** (regroupement des ressources pour optimiser leur administration / utilisation)
  Exemple : stockage, serveurs

Pourquoi virtualiser ? Objectifs

- Réduction des coûts (matériels, maintenance, …)
- Amélioration du niveau de service et flexibilité
- Renforcement de la sécurité
- Simplification de l’administration

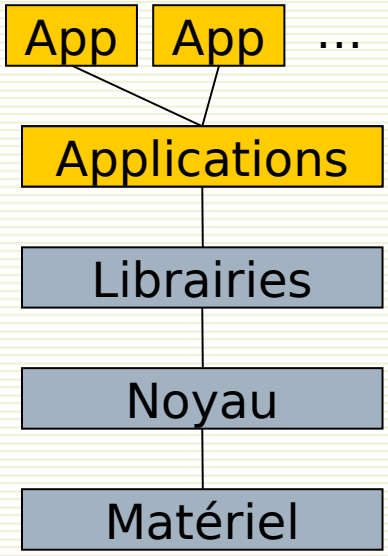
Pourquoi virtualiser ? Une évolution forte

- Beaucoup d'articles
  - January 9, 2006 - Network World
    Virtualization a Hot Technology for 2006
- Banalisation / démocratisation des solutions
  - Intégré dans les distributions Linux (Fedora, Redhat, Debian, Mandriva)
  - Intégré dans Windows, gratuit (virtual server ou Hyper-V)
    http://www.microsoft.com/windowsserver2008/en/us/virtualization-consolidation.aspx
  - February 6, 2006 : VMware Introduces Free VMware Server

Plan

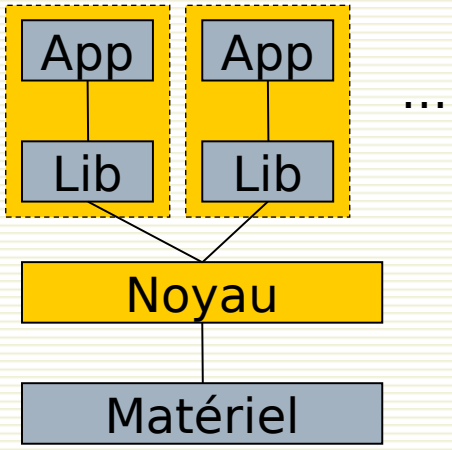
- Pourquoi virtualiser ?
- Techniques de virtualisation
- Linux VServer
  - Architecture
  - Mise en œuvre
  - Retour d’expérience
- Conclusions & perspectives
- TP

Techniques de virtualisation – Niveau applicatif



- La virtualisation peut intervenir à différents niveaux
- Au niveau applicatif : l'application fait croire qu'il y a plusieurs services
  - Hôtes virtuels Apache, domaines virtuels Postfix, ...
  - Performances optimales

Techniques de virtualisation – Conteneur

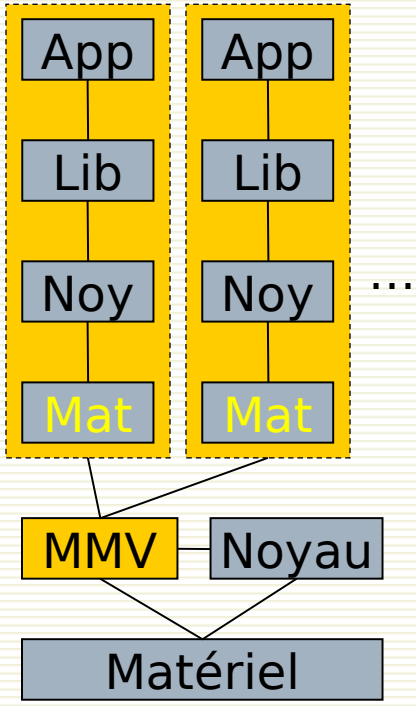


- Au niveau du noyau : **séparation des applications**, regroupées dans des « cages » étanches
  - Un seul noyau, qui fait croire à plusieurs machines
  - Il répartit les ressources
  - BSD Jails, Solaris Zones, **Linux VServer**, OpenVZ
  - Performances excellentes

Techniques de virtualisation – Conteneur Linux

- Pas de solution « native » et complète sous Linux
- Des projets anciens : Linux-VServer, OpenVZ
- Des composants déjà existant dans le noyau (chroot, capacités, VFS namespace)
- Un intérêt croissant et une volonté d'intégrer des composants manquants dans le noyau
  - 2.6.18 : *UTS namespace*
  - 2.6.24 : *PID namespace*
  - 2.6.24 : *control groups* (cgroups), nommé dans un premier temps *process container*
  - 2.6.24 : *network namespace*
- Des éléments qui pourraient servir pour du checkpoint/restart, de la migration de processus, des politiques d'ordonnancement par usager, …

Techniques de virtualisation – Hyperviseur



- Un *hyperviseur* (de type 1), ou *moniteur de machine virtuelle*, fonctionne directement au-dessus du matériel
  - VMware ESX, Microsoft Hyper-V, **Xen**
  - Les instructions machines s'exécutent en grande partie nativement sur le processeur
  - Performances bonnes à très bonnes
  - NB : un hypercall ou appel hyperviseur signifie que le système invité fait directement un appel à l'hyperviseur (par référence aux appels systèmes, syscall) pour exécuter des instructions *sensibles*

Techniques de virtualisation – Hyperviseur x86

- Le processeur x86 « classique » se prête mal à la virtualisation, malgré son mode dit *protégé* qui offre 4 niveaux de privilèges différents (ring 0 réservé au noyau, ring 3 pour les applications)
- L'hyperviseur et la technique des hypercall sont là pour pallier ses carences : allocation mémoire, interception (ou détournement par hypercall) des instructions processeurs sensibles
- Les technologies Intel-VT / AMD SVM (ou AMD-V) introduites en 2006 ont rendu le processeur « virtualisable », ce qui allège le travail de l'hyperviseur
- Très bientôt, le matériel apportera aussi la virtualisation pour la mémoire

Plan

- Pourquoi virtualiser ?
- Techniques de virtualisation
- Linux VServer
  - Architecture
  - Mise en œuvre
  - Retour d’expérience
- Conclusions & perspectives
- TP

Linux VServer – Introduction (1)

- Idée : séparer l'espace utilisateur d'un système GNU/Linux (« hôte ») en unités distinctes (« serveurs privés virtuels » ou « vservers »)
- Patch sur le noyau Linux (`patch-<version_linux>-vs<version_vserver>`)
- Commandes utilisateurs (`util-vserver`)
- http://linux-vserver.org/
- Liste de diffusion http://list.linux-vserver.org
- **#vserver** sur **irc.oftc.net**
- Début du projet en 2001, utilisable depuis 2003

Linux VServer – Introduction (2)

- Historique
  - Liste de diffusion : 2001
  - Version 1.0, novembre 2003 (Linux 2.4.20)
    patch 1146 lignes ajoutées ou modifiées
  - **Version 1.2, décembre 2003 → janvier 2005**
    patch > 2000 lignes
  - **Version 2.0, août 2005 (Linux 2.6.12)**
    patch ≈ 10000 lignes
  - Version 2.2.0, nov. 2006 (Linux 2.6.20)
  - **Version 2.2.0.7, mars 2008 (Linux 2.6.22.19)**
    patch ≈ 17000 lignes, 458 fichiers
  - Adaptation en cours aux 2.6.24+ avec patchs
    « containers »

VServer – Isolation de processus

- **Contexte** : nouvelle structure du noyau, identifié par un entier
- Chaque processus fait partie d'un contexte
- Interactions entre processus (signaux, IPC…) limitées à un contexte ($\Rightarrow$ *isolation* plutôt que *virtualisation*)
- Contexte de l'hôte : 0
  - Peut créer de nouveaux contextes
  - Peut changer de contexte
- Contexte « spectateur » : 1
  - Peut voir les processus de tous les contextes
- Un contexte $\approx$ un vserver

VServer – Isolation réseau

- L'hôte dispose de plusieurs adresses réseaux (éventuellement des alias)
- Les processus d'un vserver sont limités à une (ou plusieurs) adresse(s). Plus précisément, les processus vont être rattachés à un « *network context* » (introduits dans VServer 2.2)
- Attention, les applications de l'hôte doivent être « bindées » sur l'adresse IP qui lui est dédiée

VServer – Isolation système de fichiers

- Chroot : la racine apparente du FS (/) est en réalité un répertoire de l'hôte
- Nouvel attribut du système de fichiers pour se prémunir de l'évasion (barrier)
- Utilisation des espaces de noms (namespaces) de la couche VFS : chaque VServer a son namespace et une vue différente du FS
- Possibilité d'associer un fichier a un contexte
  - Clé d'accès
  - Nécessaire pour avoir une limite disque par VServer et des quotas par VServer dans le cas d'une partition partagée

VServer – Limitation du super-utilisateur

- Capacités
  - Brouillon de norme POSIX, partiellement supportée depuis Linux 2.2
  - Jeton présenté par un processus pour prouver qu'il est autorisé à faire une action
  - Exemple : créer un fichier périphérique (MKNOD)
  - On peut fixer une limite aux capacités d'un contexte ⇒ *root* ne pourra pas tout faire (*bounding capabilities*, bcaps)
- Nouvelles capacités (*context capabilities*, *ccaps*). Exemple :
  - CAP_NET_RAW trop fort. Mais sans lui, pas de ping…
  - Solution : VXC_RAW_ICMP

VServer – Isolation et extension de /proc

- /proc
  - Système de fichiers virtuel
  - Accès (lecture ou lecture/écriture) aux informations du noyau
- Nécessaire dans un vserver : uptime, liste des processus, type de cpu, mémoire utilisée, points de montage, ...
- Mais pas tout
  - Les processus des autres contextes n'apparaissent pas
  - Certaines entrées sont « cachées » à l'aide d'attributs supplémentaires
- Extensions sous /proc/virtual/ et /proc/virtnet/

VServer – Limiter les ressources

- Coopération des processus et allocation des ressources : par le noyau (classiquement)
- ulimit par vserver : limitation de la mémoire, du nombre de processus, ...
- Consommation de CPU limitable par un algorithme « seau de jeton »
- Disque
  - Une partition par vserver...
  - ... ou utiliser le marquage des fichiers par contexte

VServer – Autres éléments

- Virtualisation d'informations systèmes
  - Nom d'hôte, version et release d'OS, type de machine, processeur (*utsname*, uname –a)
  - Uptime
  - Quantité de mémoire disponible (en fonction des limites fixées)
- Unification
  - Partager des fichiers entre VServer à travers des liens en dur, idéalement tout / sauf …
  - Gain de disque
  - Mise à jour

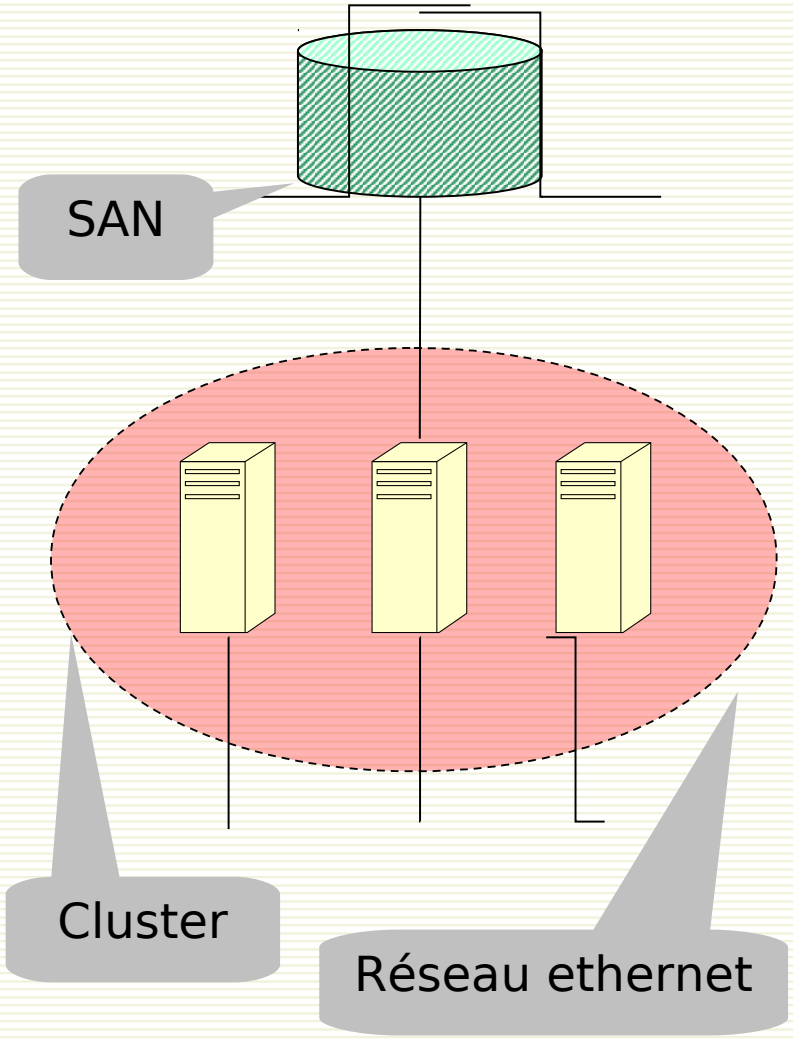
VServer – Limites

- Interface de boucle locale : pseudo-loopback dans la branche de développement 2.3
- Support IPv6 : dans 2.3
- NFS en mode noyau : non
- Migration à chaud de VServer : non
- Des vservers sur plusieurs VLAN : possible
- Netfilter par VServer : non, nécessite une virtualisation complète de la couche réseau
- Manque une interface qui simplifierait la gestion
- Nécessité de bien connaître GNU/Linux et compréhension des mécanismes ci-dessus…

VServer – Retour d’expérience

- INSA de Toulouse : version 1.2 depuis 2004
  - P4, RAM 1Go, LVM sur disques locaux en miroirs
  - Serveur www institutionnel
  - 8 autres serveurs (applications web), pour répondre à des demandes rapidement
- Université de Limoges : version 2.x depuis septembre 2005
  - 3 serveurs bi-Xeon, RAM 8 Go
  - Serveurs de courrier des étudiants (14000), serveur web institutionnel, webmail, ENT, DNS secondaire, moodle, FTP, …

VServer – Retour d'expérience (2)



- Architecture en production à l'université de Limoges (janvier 2008)
- 3 serveurs (bi-Xeon EM64T, RAM 8 Go) formant un cluster Red Hat
- Un disque (du SAN, 700 Go) accessible à tous les membres
- Disque intégré dans un *Volume Group* Cluster LVM
- Pour migrer (à froid) les VServers d'un hôte vers un autre
- Avec surveillance automatique des hôtes et des VServers (haute-dispo)

Plan

- Pourquoi virtualiser ?
- Techniques de virtualisation
- Linux VServer
  - Architecture
  - Mise en œuvre
  - Retour d’expérience
- Conclusions & perspectives
- TP

Conclusions & perspectives

- ☺ Une solution qui tient ses promesses : efficace, léger, robuste
- ☹ Après son apprentissage ...
- ☹ Manque d'intégration dans les distributions
- ☹ Des points à améliorer dans ou autour de VServer
  - Outils de supervision
  - Outils d'administration
- ➲ Approche de consolidation à intégrer dans une démarche globale ?

Pages importantes

- http://2005.jres.org/paper/109.pdf
- http://www.renater.fr/Video/JRES/TutoJRESMars2008/P/Perrot/techvirtualisation-tutojre

- http://linux-vserver.org/Paper
- http://linux-vserver.org/Feature_Matrix
- http://linux-vserver.org/Frequently_Asked_Questions
- http://linux-vserver.org/Capabilities_and_Flags
- http://linux-vserver.org/CPU_Scheduler
- http://linux-vserver.org/Resource_Limits
- http://linux-vserver.org/Memory_Limits
- http://www.nongnu.org/util-vserver/doc/conf/configuration.html