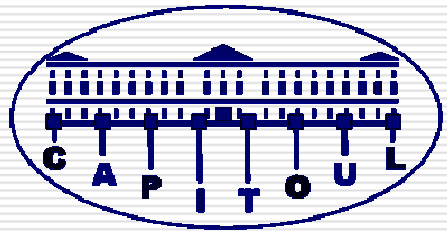
Réunion Virtualisation
Toulouse – 14 décembre 2006



Linux-VServer

Stéphane Larroque
INSA de Toulouse - CRI
135 avenue de Rangueil
31077 Toulouse Cedex 4
stephane.larroque@insa-toulouse.fr

Xavier Montagutelli
Université de Limoges
Service Commun Informatique
123 avenue Albert Thomas
87060 Limoges
xavier.montagutelli@unilim.fr

Plan

- Concepts
- Mise en œuvre
- Retour d'expérience
- Conclusions & perspectives

Plan

- Concepts
- Mise en œuvre
- Retour d'expérience
- Conclusions & perspectives

Éléments d'introduction [1]

- « Virtualisation » au niveau noyau : un seul noyau !
- Vocabulaire : serveurs privés virtuels (virtual private servers ou VPS)
- Division de l'espace des applications en « boîtes » étanches
- Chaque serveur virtuel :
  - Ne voit que ses applications (processus)
  - A son adresse IP
  - A son espace disque

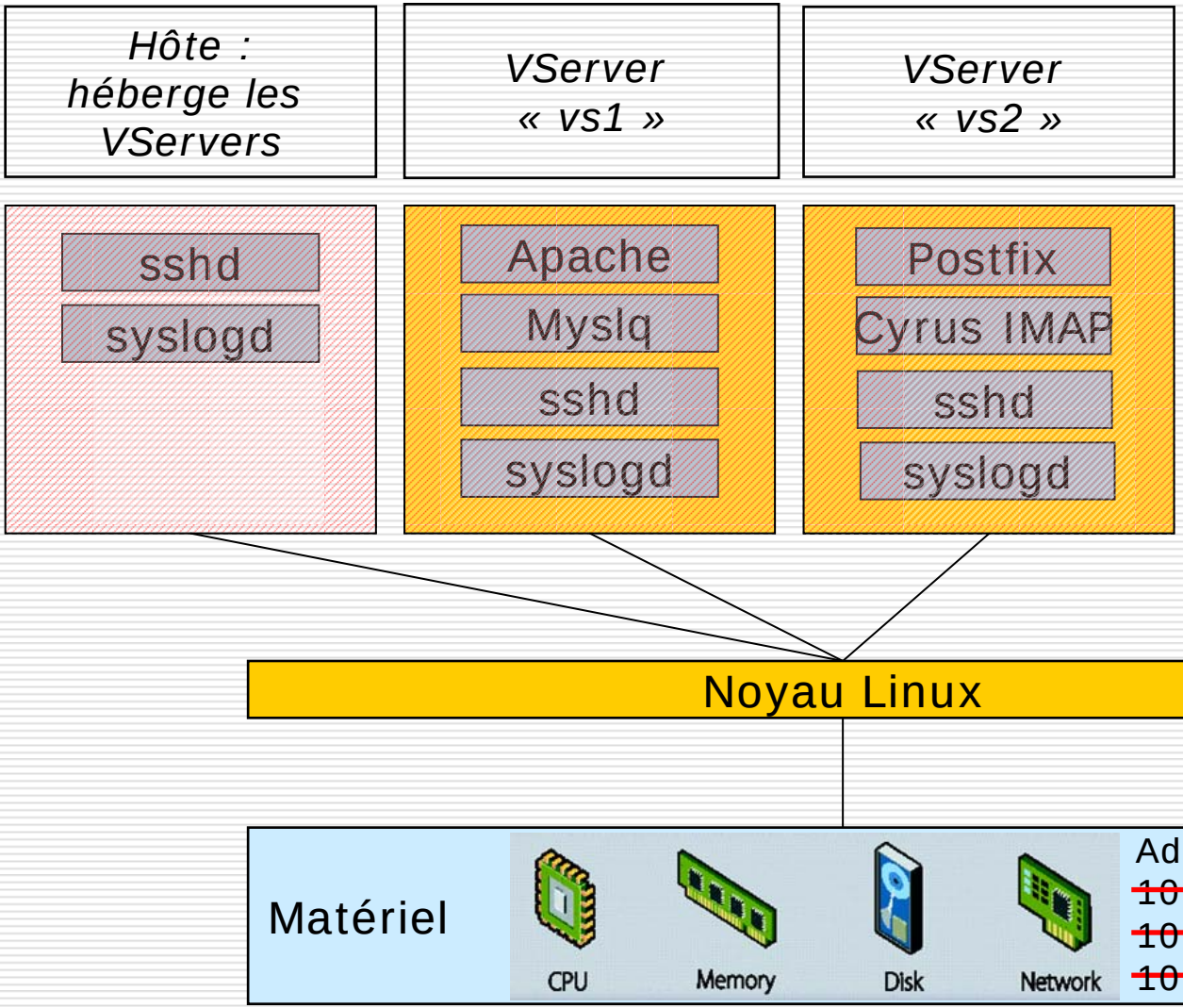
Éléments d'introduction [2]

- Commandes utilisateurs
- Patch sur le noyau Linux
- 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

Éléments d'introduction [3]

- Historique
  - Liste de diffusion : 2001
  - Version 1.0 : novembre 2003
    patch 1146 lignes ajoutées ou modifiées
    Linux 2.4.20
  - Version 1.2 : décembre 2003 → avril 2005
    patch > 2000 lignes
  - Version 2.0 : août 2005
    patch ≈ 10000 lignes
    Linux 2.6.12.4
  - Version 2.0.2.1 : septembre 2006
    patch ≈ 11000 lignes
    Linux 2.6.17.13

Isolation de processus et réseau



- Hôte : voit tout
- vs1 : limité à ses processus, à son IP
- Idem vs2

...

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
- 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 ≈ un VServer

Jouer avec les contextes

```
# chcontext --xid 33 /bin/bash
New security context is 33
# sleep 1000 &
# ps auxw
USER PID  COMMAND
root 4400 /bin/bash
root 4418 sleep 1000
root 4419 ps auxw



# ps auxw | grep sleep
[1]+ Terminated  sleep 1000
# chcontext --xid 44 bash
vcontext: vc_create_context():
    Operation not permitted
```

```
# ps auxw | grep sleep
root 4421 grep sleep
# chcontext --xid 1 ps auxw \
  | grep sleep
root 4418 sleep 1000
root 4427 grep sleep
# kill 4418

# vps auxw
USER PID CONTEXT COMMAND
[..]
root 4400 33     /bin/bash
```

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)

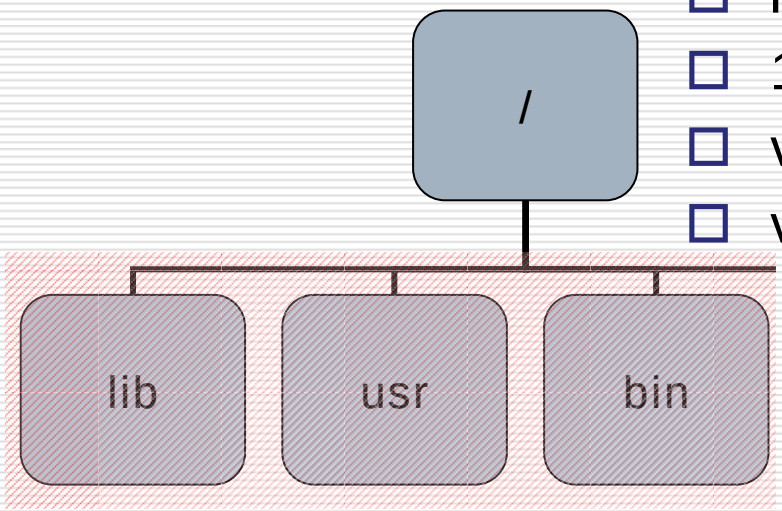
Jouer avec le réseau

```
# ip addr
1: eth0:
  inet 192.168.1.128/24 eth0
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=128 time=1.13 ms

# ip addr add 10.0.0.1 label eth0:XX dev eth0
# ip addr
1: eth0:
  inet 192.168.1.128/24 eth0
  inet 10.0.0.1/32      eth0:XX
# chbind --ip 10.0.0.1 bash
ipv4root is now 10.0.0.1
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Isolation système de fichiers



- Hôte : voit tout le disque
- 1 répertoire par VServer
- vs1 : limité à sa partie
- vs2 : limité à sa partie

Isolation système de fichiers

- Chroot
- 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 à 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

Jouer avec le système de fichiers

```
# vnamespace --new bash
# mount
/dev/sda1 on / type ext3
none on /proc type proc
# mount -o loop /disk.raw /mnt/
# mount
/dev/sda1 on / type ext3
none on /proc type proc
/disk.raw on /mnt type ext2
# echo "Namespace" > /mnt/ok
# cat /mnt/ok
Namespace
#
# cat /proc/mounts | grep mnt
/dev/loop0 /mnt ext2 rw 0 0
#
```

```
# mount
/dev/sda1 on / type ext3
none on /proc type proc
#
#
# mount
/dev/sda1 on / type ext3
none on /proc type proc
/disk.raw on /mnt type ext2
#
# cat /mnt/ok
cat: /mnt/ok: No such file or
    directory
# cat /proc/mounts | grep mnt
#
```

Limitation du super-utilisateur

- Privilèges
  - En général, jeton présenté par un processus pour prouver qu'il est autorisé à faire une action
  - Mais également, norme POSIX, partiellement supportée depuis Linux 2.2
  - On peut fixer une limite aux privilèges d'un contexte $\Rightarrow$ *root* ne pourra pas tout faire
- Nouveaux privilèges. Exemple :
  - CAP_NET_RAW trop fort. Mais sans lui, pas de ping…
  - Solution : VXC_RAW_ICMP

Jouer avec les privilèges

```
# chcontext --xid 33 \
    --cap '!CAP_NET_RAW' bash
New security context is 33

# ping 192.168.1.1
ping: icmp open socket:
    Operation not permitted

# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1)
    56(84) bytes of data.
64 bytes from 192.168.1.1:
    icmp_seq=0 ttl=128
    time=0.719 ms
```

```
# vattribute --set --xid 33 \
    --ccap raw_icmp
```

Isolation et extension de /proc

- /proc
  - Système de fichiers virtuel
  - Accès 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/

Jouer avec /proc

```
# chcontext --xid 33 bash
New security context is 33
# cat /proc/uptime
1628.44 1574.30

# cat /proc/uptime
cat: /proc/uptime: No such file
    or directory
```

```
# showattr /proc/uptime
Awh-ui- /proc/uptime


# setattr --hide /proc/uptime
# showattr /proc/uptime
AwH-ui- /proc/uptime
# grep uptime
    /usr/local/lib/util-
    vserver/defaults/vprocunhide
    -files
/proc/uptime
# cat /proc/virtual/33/status
UseCnt: 3
Tasks:  1
Flags:  0000000200000000
BCaps:  ffffffffffffeff
CCaps:  0000000000000000
Ticks:  0
```

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

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 VServers à travers des liens en dur
  - Gain de place
  - Mise à jour possible

Jouer avec la virtualisation

```
# chcontext --xid 33 bash
New security context is 33
# hostname
scivm-xm4.unilim.fr

# hostname
marseille.jres.org
```

```
# hostname
scivm-xm4.unilim.fr


# vuname --xid 33 –s \
  -t NODENAME=marseille.jres.org
```

Linux-VServer

Limites

- Pas d'interface de boucle locale
- Pas de NFS en mode noyau
- Pas de table de routage par VServer (sauf en jouant avec iproute2) $\Rightarrow$ difficulté pour avoir des VServers sur plusieurs VLAN
- Nécessité de bien connaître GNU/Linux
- Compréhension des mécanismes ci-dessus...

Plan

- Concepts
- Mise en œuvre
- Retour d'expérience
- Conclusions & perspectives

Construction de l'hôte

- Installer une distribution GNU/Linux (minimale)
- Recompiler le noyau après patch
- Compiler et installer les outils
- Finir l'installation « à la main »
  - 3 scripts de démarrage (/etc/init.d/{vprocunhide, vservers-default, rebootmgr})
  - Attribut « barrier » sur le répertoire des VServers
- Configurer les services réseaux de l'hôte pour ne pas faire de bind ANY (sur 0.0.0.0)

Construction d'un VServer [1]

```
# vserver vs build –m apt-rpm --context 33 \
    --hostname vs.jres.org \
    --interface vs=eth0:10.1.1.2 --flags virt_uptime,virt_mem \
    -- -d rhel4
```

- « vserver » : gérer les VServers
- « -m apt-rpm » : méthode de construction
  - Valeurs possibles : skeleton, debootstrap, yum, rpm, apt-rpm
- « -- -d rhel4 » : option pour la méthode
  - « rhel4 » définit la distribution : paquets à installer, script de pré et post création du VServer, …

Construction d'un VServer [2]

- Un VServer =
  - Répertoire de configuration /etc/vservers/<vs>
  - Répertoire chroot /vservers/<vs>
  - Répertoire pour la gestion des paquets /vservers/.pkg/<vs>
- Exemples de paramètres à configurer
  - Mots de passe md5 / mot de passe root
  - /etc/resolv.conf
  - Script /etc/init.d/halt
- Taille typique : serveur avec SSHD + Apache + PHP + MySQL ≈ 300 Mo

Construction par virtualisation

```
# vserver vs build –m skeleton --context 33 \
    --hostname vs.jres.org \
    --interface vs=eth0:10.1.1.2 --flags virt_uptime,virt_mem
```

- Créer un VServer « squelette » (-m skeleton)
- Recopier le serveur réel :
  - Par « tar » / « rsync » / « cpio »
  - Exclure les répertoires inutiles (/dev, /proc, /mnt, /sys)
- Désactiver les services inutiles (détection du matériel par exemple)
- Éventuellement, modifier le script de démarrage ou d'arrêt

Démarrage d'un VServer

```
# vserver vs start
```

- Par défaut : /etc/rc 3
  - Runlevel modifiable : /etc/vservers/<vs>/apps/init/runlevel.start
  - Émulation « faux » processus init
- Ou alors option « plain » (vrai init) /etc/vservers/<vs>/apps/init/style
- Démarré au lancement de l'hôte ? /etc/vservers/<vs>/apps/init/mark
- fstab : dans l'espace de noms du VServer /etc/vservers/<vs>/fstab

Surveillance

```
# vserver-stat
CTX   PROC    VSZ    RSS  userTIME   sysTIME    UPTIME NAME
0       40  81.9M   7.9M   0m06s69   1m06s94   2h25m57 root server
33       9  37.6M  12.4M   0m00s20   0m00s25   0m05s10 vs
```

- Depuis l'hôte
- Autres commandes :
  - vps
  - vtop
- /proc/virtual

Gestion des paquets

- Depuis l'hôte : vrpm, vapt-get
  - Sur plusieurs VServers en même temps
- La gestion des paquets peut
  - Être interne au VServer (délégation)
  - Rester la prérogative de l'hôte

Plan

- Concepts
- Mise en œuvre
- Retour d’expérience
- Conclusions & perspectives

Linux-VServer – Retour d'expérience

- INSA : depuis 2004 (v1.2.10 à v2.0.2.1)
  - 7 serveurs hôtes
    - 3 serveurs physiques avec disques en RAID 1 et LVM
    - 4 serveurs virtuels avec disques virtuels et LVM
  - 32 vserveurs
    - 21 serveurs web
    - 2 serveurs LDAP / 2 serveurs Kerberos
    - 1 serveur d'impression / 1 serveur de noms
    - 1 proxy/cache web / 1 proxy FTP
    - 1 proxy radius / 1 relais ssh
  - Répondre rapidement aux demandes d'hébergement
  - Économiser les ressources

Linux-VServer – Retour d'expérience (2)

- Université de Limoges : depuis septembre 2005 (v2.0)
  - Bi Xeon, RAM 5Go, LVM sur SAN
  - Espace disque presque « ajustable » à volonté
  - Pour sécuriser (cloisonner) les sites webs hébergés (avant, hôtes virtuels Apache)
  - Serveur de courrier étudiant
  - Serveurs de test
  - Répondre à des demandes d'hébergement
  - Disponibilité : si une machine physique tombe en panne, on peut redémarrer ses VServers depuis un autre hôte

Plan

- Concepts
- Mise en œuvre
- Retour d'expérience
- Conclusions & perspectives

Conclusions & perspectives [1]

- Une solution qui a tenu ses promesses
  - Facilité de déploiement
  - Rationalisation des dépenses
- Des points à améliorer
  - Supervision
- Des limitations
  - Étanchéité ?
  - Fragilisation de l'architecture

Conclusions & perspectives [2]