Virtualisation
Haute disponibilité

Antoine Souques
corum@via.ecp.fr

Virtualisation ≠ Haute disponiblité

- Haute disponibilité : rendre un service disponible le plus longtemps possible

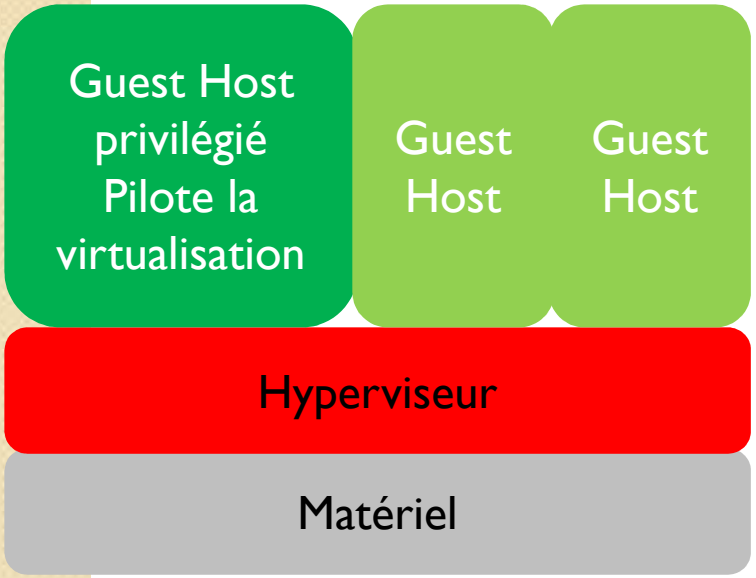
- Virtualisation : abstraction entre le matériel et les appels

La Virtualisation

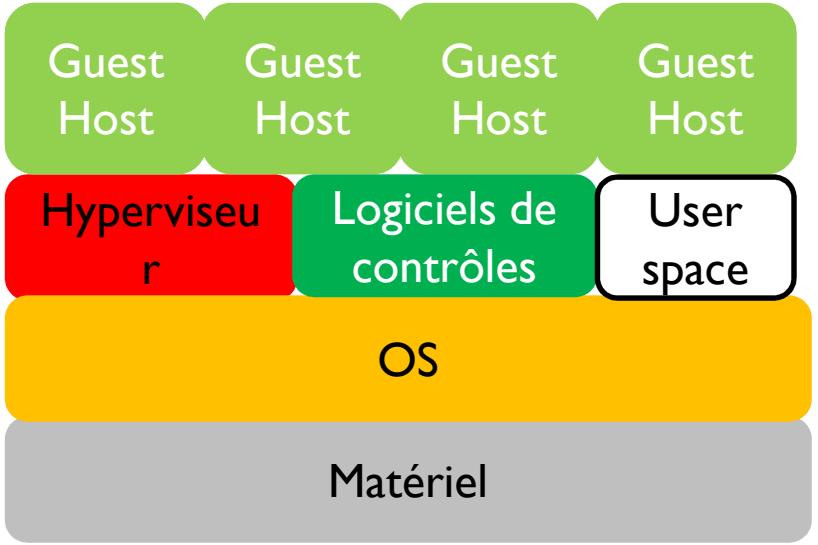
- Plusieurs machines (guest host : les pokemon) tourne sur une seule grosse machine (hyperviseur : indra, ravana, bhaga)
- Les hyperviseurs émulent ou simule les périphérique
- Complete dissociation entre le materiel et les OS invités
- On peut donc faire tourner des machines complètements différentes, avec du « materiel » différent
- Permet d'avoir des serveurs à faible coup, et donc de dissocier les services

Type d'hyperviseurs

Hyperviseur de type I



Hyperviseur de type II



Types de machines virtuelles

- Emulation
  - L'hyperviseur crée des périphériques qui, vu de la machine sont comme des périphériques rééls

  → Virtualisation complète

- Simulation
  - L'hyperviseur fournit des interfaces à la machine jouant le rôle de périphériques

  → Paravistualisation

La virtualisation par containeur

- Pas vraiment une virtualisation
- Plus un isolement
- Une application utilise les ressource de la machine physique tout en étant cloisonné
- Exemple : chroot

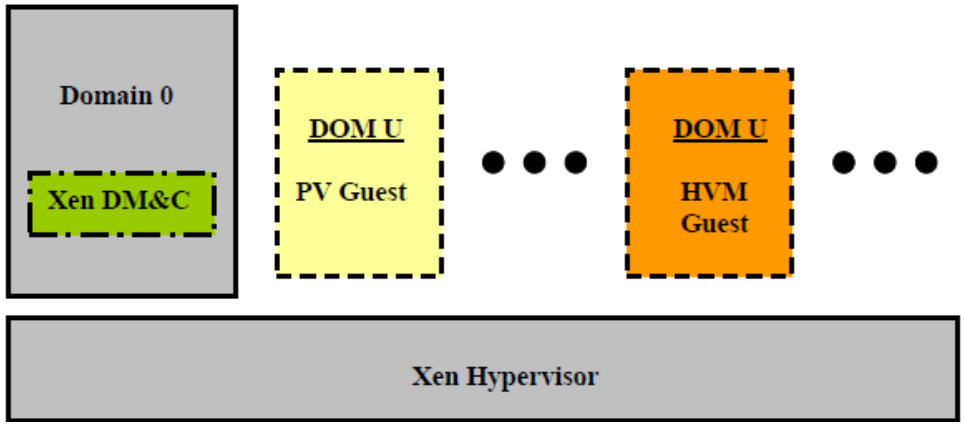
Quelques noms

- Xen : premier hyperviseur libre
  - + robuste, fiable, à fait ses preuves. Base de nombreux hyperviseurs propriétaires (LongHorn)
  - - noyau à part entière : lourd développement
- KVM : le challenger
  - + Juste un programme comme les autres, dynamique
  - - Petits retards de performances, nécessite du matériel spécifique
- Mais aussi : Vmware, VirtualBox, UML, OpenVZ, qemu…

Logiciel de contrôle

- Permet de piloter l'hyperviseur
- Actions standards : démarrage, arrêt, reboot, attachement de console…
- Actions plus avancés : migration d'instances entre deux hyperviseurs identiques

Xen (Cambridge, 2003)



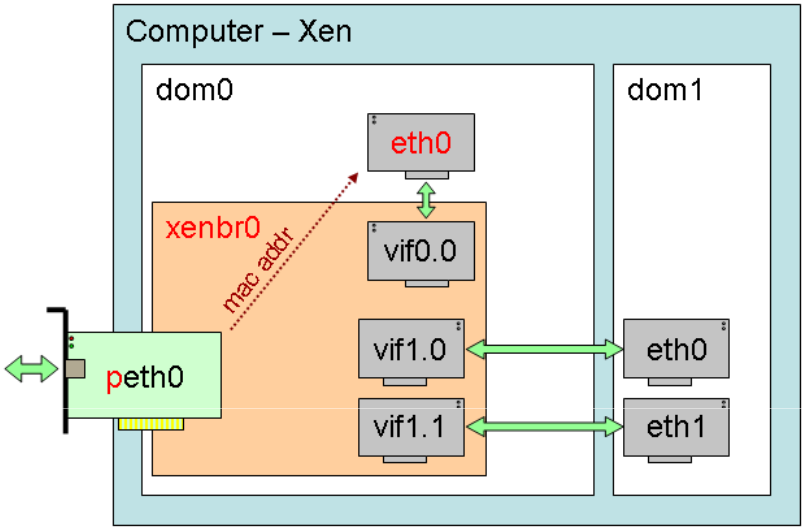
- Hyperviseur de type I avec paravirtualisation et virtualisation complète
- Dom 0 : linux modifié, contenant les driver du materiel, et les logiciels de contrôle
- Doms U : Systèmes invités
- En dehors du noyau (nécessité de patcher)

Gestion des DomU

```
kernel = '/boot/vmlinuz-2.6-xenU'
ramdisk = '/boot/initrd.img-2.6.26-2-xen-amd64'
memory = 2048
vcpus = 4
name = 'draco.via.ecp.fr'
vif = ['mac=aa:00:00:b0:c6:ab, bridge=xen-br0']
disk = ['phy:/var/run/ganeti/instance-
    disks/draco.via.ecp.fr:0,sda,w','phy:/var/run/ganeti/instance-
    disks/draco.via.ecp.fr:1,sdb,w']
root = '/dev/sda1'
on_poweroff = 'destroy'
on_reboot = 'restart'
on_crash = 'restart'
extra = 'ro‘
```

On utilise xm pour les commander

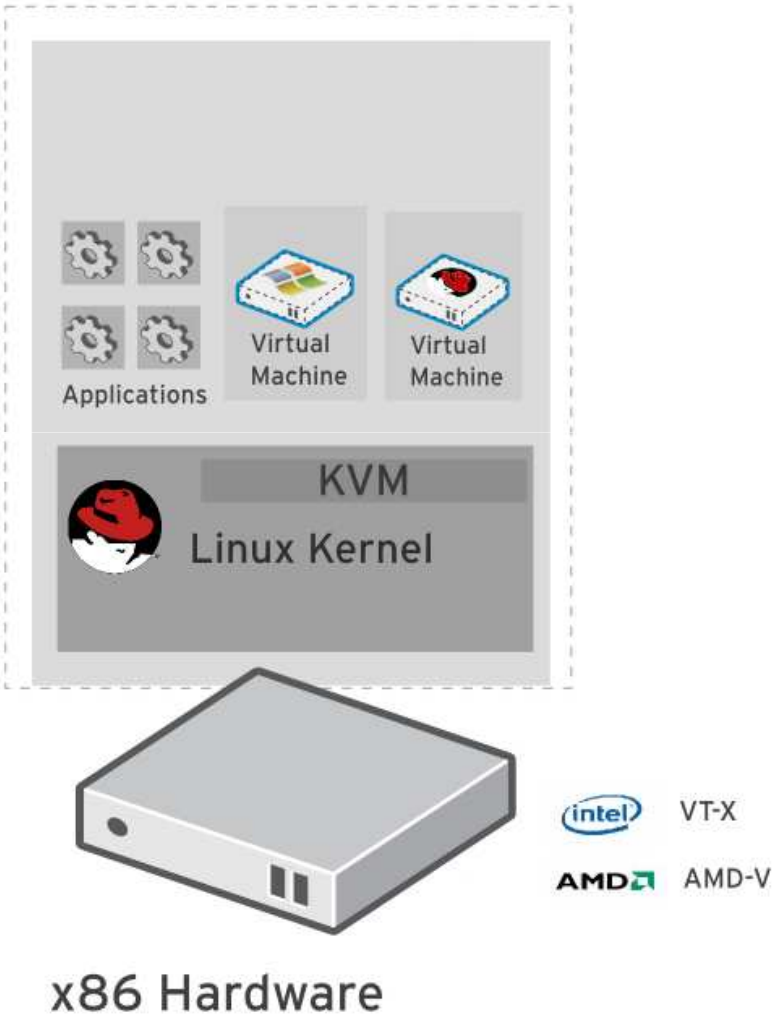
La gestion du réseu (bridge)



- Pour chaque interface des DomUs, une interface vifx.y est crée dans le Dom0
- Les deux sont relié par un fil virtuel
- Toutes les interfaces vif x.y sont dans un même bridge
- Possibilité de choisir le bridge
- Choix des interfaces physiques à inclure dans le bridge

KVM (2007)

- Linux est un très bon hyperviseur
- Les MV ne sont que des threads
- Orienté virtualisation complète
- Paravirtualisation possible (virtio)
- Intégré au noyau



Gestion des machines virtuelles

- Pas de fichiers de configurations
- Gestion de base avec qemu-kvm

```
$ /usr/bin/qemu-kvm -M pc \
     -m 1024 \
     -smp 1 \
    -name kvmnode1 \
    -monitor stdio \
    -boot n \
    -drive file=/nfs/vms/kvmnode1,if=ide,index=0 \
    -net nic,macaddr=54:52:00:53:20:00,vlan=0 \
    -net tap,script=no,vlan=0,ifname=tap0 \
    -serial stdio\
    -nographic \
    -incoming tcp:0:4444
```

Haute disponibilité par la virtualisation

- Les machines hôtes ne sont que des entités logique
- Migration possible à faible coup

- Mais la virtualisation n'est pas la solution parfaite à la haute disponibilité

Exemple : ne peut rien faire contre rm –rf /

RAID (1987, Berkley)

But : assembler des diques de mauvaise qualités pour avoir un ensemble de bonne qualité

- RAID0 : stripping
- RAID1 : mirroring
- RAID5 : striping with distributed parity
- RAID6 (striping with double distributed parity)

- Peuvent être combinés
- RAID logiciel inclus dans le noyau (md)

LVM (1997)

- Permet une plus grande souplesse dans la gestion des disques
- Permet de créer des disques logiques
- Interet pour la virtualisation : un disque de MV est un LV

DRBD (2001, Linbit)

- RAID1 over TCP/IP
- Configuration primary/secondary ou primary/primary (nécessite un système de gestion partagée)
- Inclus dans le noyau depuis 2.6.33

Systèmes distribués

- Clvm : permet d'utiliser du lvm sur des devices partagés
- GFS, OCFS : systèmes de fichiers pouvant être monté en même temps à plusieurs endroits
- Reposent sur un système de messagerie, comme openais ou cman

Disques en réseau

- Technologie permettant d'exporter un volume vers un serveur client
- Du coté du client, il s'agit d'un disque comme les autres
- Interet : séparation du stockage et de la puissance de calcul
- Technologies : ATA over Ethernet (AoE), iSCSI, FiberChannel (FC)

AoE

- Protocole reposant sur la couche link du réseau
- Trames ethernet spéciales
- Léger
- Non routable
- Peu de contrôle possible

iSCSI

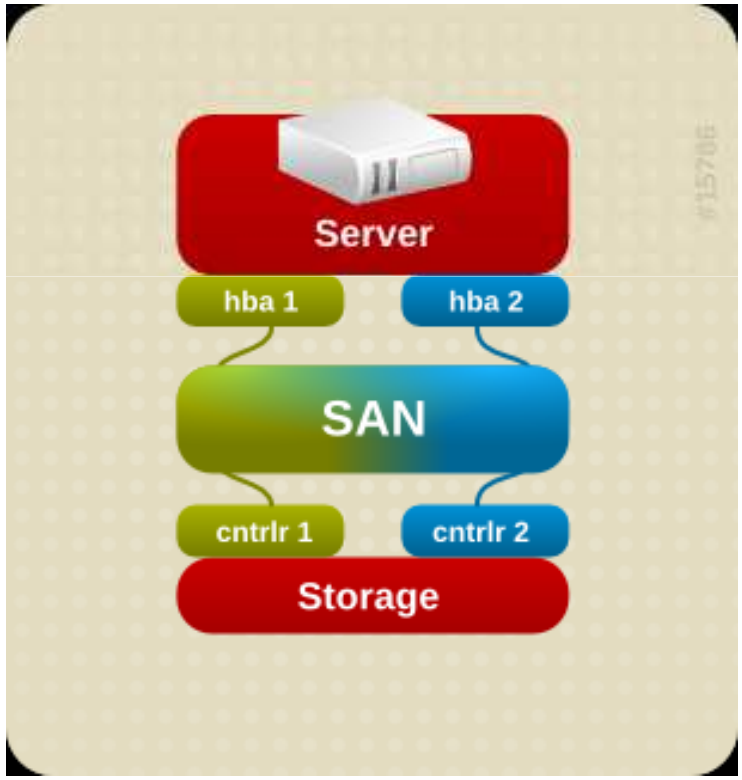
- Encapsulation de trames SCSI dans du TCP/IP
- Beaucoup plus lourd que l'AoE
- Crée une charge CPU non négligeable
- S'appuie sur la puissance du TCP
- Permet d'utiliser tous les outils autour de TCP/IP

FC

- Permet de transférer les trames SCSI sur une fibre optique
- Gros débit
- Peu de surcharge (presque comme du SCSI)
- Requière un réseau fibre spécifique
- Coute cher

Multipathing

- Un device peut être accessible par différents chemin (plusieurs controleurs, mirroring en drbd…)
- dm-multipath permet de repérer les différents chemins pour accéder à un même device
- Failover ou load balancing entre les différents chemains



Besoin d'outils de management

- Car les outils standards ne sont pas forcément facile d'accès
- Pour interfacer la virtualisation avec d'autres technologies
- Pour manager efficacement plusieurs hyperviseurs, potentiellement différents

Coût : perte de fonctionnalité

Quelques noms

- Libvirt : librairie C permettant de dévelloper son propre manager
- Virsh : implementation basique de libvirt
- oVirt
- Convirt
- **Ganeti**
- Openebula

Et bien d'autres

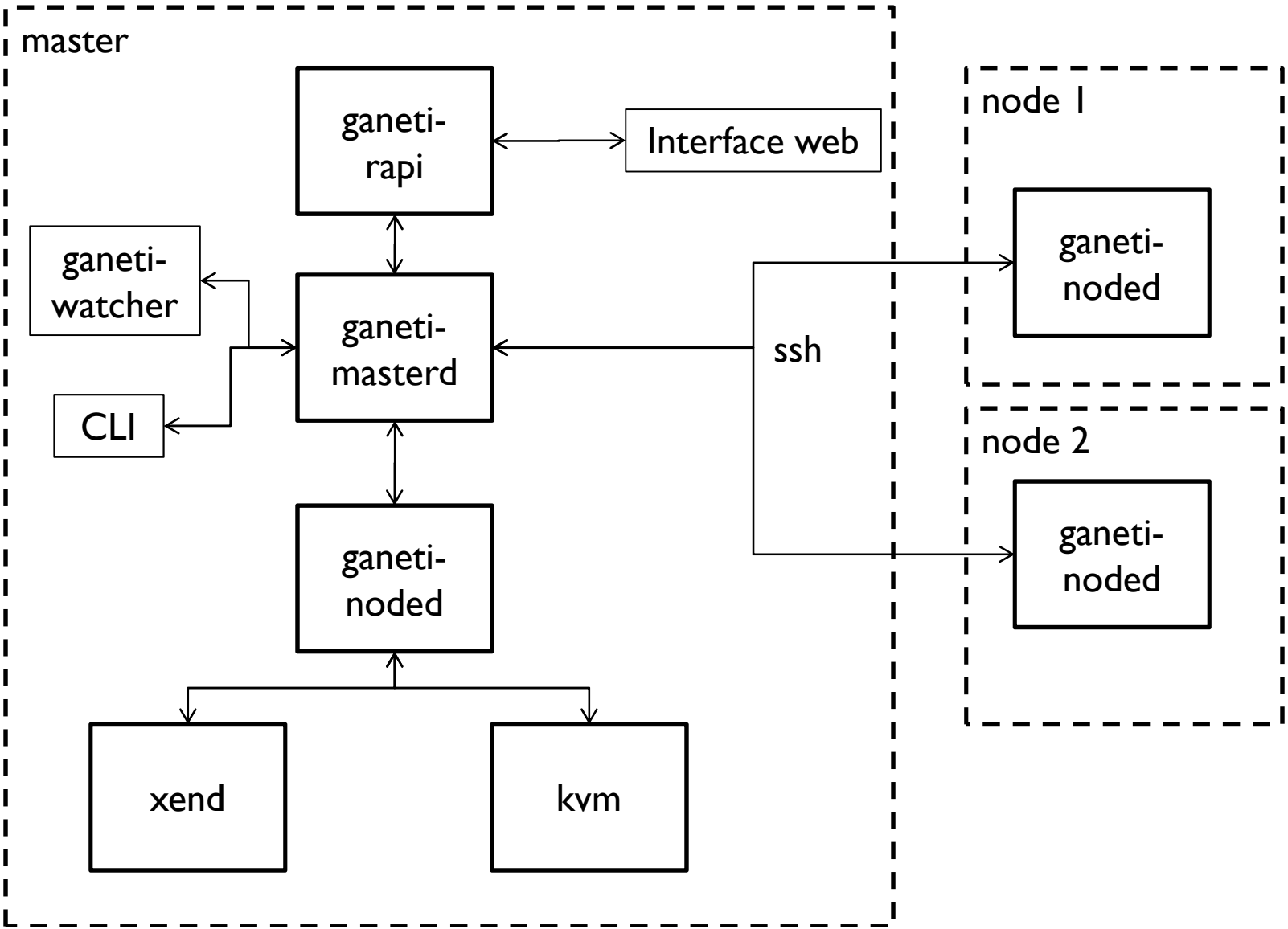
Ganeti (google, 2007)

- Permet de gérer un cluster de virtualisation
- Support de xen et KVM
- Pas de services exterieurs requis
- Haute disponibilité possible avec DRBD
- D'apparence facile

Mais

- Introduit une dissymétrie entre les serveurs
- Fonctionnement interne un peu difficile
- Pas de support de disques exportés

Fonctionnement de ganeti : démons



Fonctionnement de ganeti : CLI

- gnt-cluster
- gnt-node
- gnt-instance
- gnt-backup
- gnt-os
- gnt-job

- Chaque tache est inséré dans une file
- En générale retourne quand la tache est executée
- Attention : ^C ≠ annulation de la tâche

A VIA : aujourd'hui

- Xen
- LVM (pour les disques des machines virtuelles)
- Ganeti
- La plupart redondées en DRBD
- Tous les services critiques sont des MV redondées

A VIA : le projet

- 2 serveurs donnée avec réplication DRBD
- Export d'un gros device en iSCSI
- Utilisation du multipath pour la redondance
- Utilisation de clvm pour créer des disques de MV
- Changement d'hyperviseur : kvm

- Quel manager prendre ?

Quelques liens

- http://www.xen.org/
- http://www.linux-kvm.org
- http://raisin.u-bordeaux.fr/IMG/pdf/cours_xen.pdf
- http://code.google.com/p/ganeti/
- http://wikipedia.org/