KVM

Kernel-based Virutal Machine

The free and open virtualisation infrastructure

9 Juin 2011

Réalisé par : Sébastien Pasche

Supervisé par : Gérald Litzistorf

Projet d'approfondissement - hepia- 2011

La thématique et les couleurs sont inspirées du magazine Muffin et de Gnome 3

PLAN DE PRÉSENTATION

Introduction

- La virtualisation, un effet de mode ?
- Pourquoi choisir la virtualisation
- Anatomie d'un hyperviseur

KVM

- Les volontés derrière KVM
- Architecture de KVM
- Analyse et améliorations possibles de KVM

Environnement KVM

- La paravirtualisation avec VirtIO
- Les limites et améliorations possibles pour VirtIO
- API LibVirt

Pratique

- Installation d'un environnement KVM
- Automatisez-la !
- Création de machines virtuelles
- Automatisez-la !

Fin

- Conclusion
- Méta-compétances
- Questions

Introduction

Une petite introduction à la virtualisation

«Painting is an illusion, a piece of magic, so what you see is not what you see.»

Philip Guston

LA VIRTUALISATION, UNE MODE ?

- MIT : The supervisor program of CTSS 1950s
- Christopher Strachey : Time Sharing in Large Fast Computers June, 1959.
- IBM Watson Research : M44/44X Project 1960s
- IBM : CP/CMS 1967
- IBM : VM370 1972 (Still in uses)
- Robert P. Goldberg : Survey of Virtual Machines Research 1974
- IBM : ZVM 2000
- Microsoft acquired Connectix Corporation 2003
- EMC announced its plans to acquire VMware for $635 million. 2003
- IBM : System X (KVM base) 2010

Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others.

Amit Singh - OS Architect at Google

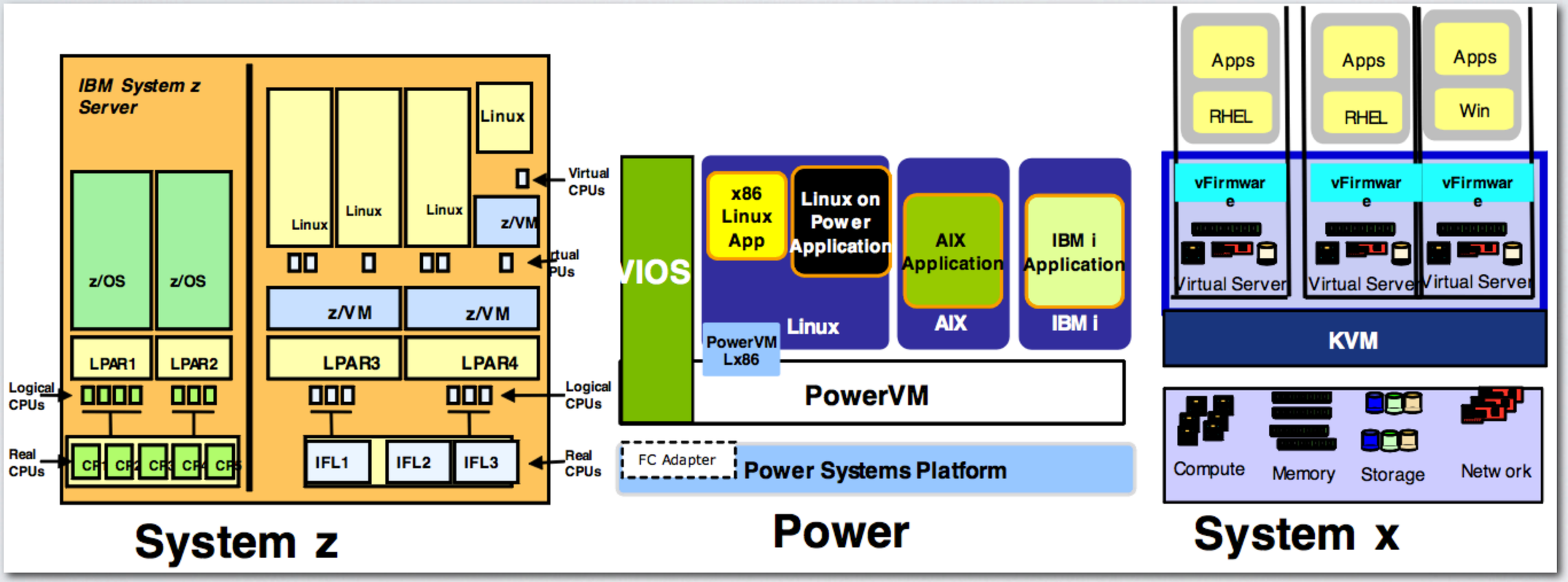
Virtual machine systems were originally developed to correct some of the shortcomings of the typical third generation architectures and multi-programming operating systems

Robert P. Goldberg : Survey of Virtual Machines Research

LA VIRTUALISATION, UNE MODE ?

Si on demande à IBM ...

1967 CP/CMS 1972 VM/370 1990 VM/ESA 2000 Z/VM 2009 KVM



POURQUOI CHOISIR LA VIRTUALISATION Servez-vous !

Virtual machines can be used to consolidate the workloads of several under-utilized servers to fewer machines, perhaps a single machine (server consolidation). Related benefits (perceived or real, but often cited by vendors) are savings on hardware, environmental costs, management, and administration of the server infrastructure.

> You can treat application suites as appliances by "packaging" and running each in a virtual machine.

Virtual machines can be used to provide secure, isolated sandboxes for running untrusted applications. You could even create such an execution environment dynamically - on the fly - as you download something from the Internet and run it. You can think of creative schemes, such as those involving address obfuscation. Virtualization is an important concept in building secure computing platforms.

Virtual machines allow for powerful debugging and performance monitoring. You can put such tools in the virtual machine monitor, for example. Operating systems can be debugged without losing productivity, or setting up more complicated debugging scenarios.

Virtualization on commodity hardware has been popular in co-located hosting. Many of the above benefits make such hosting secure, cost-effective, and appealing in general.

Virtualization can be an effective means of providing binary compatibility.

Virtual machines make software easier to migrate, thus aiding application and system mobility.

Virtual machines can provide the illusion of hardware, or hardware configuration that you do not have (such as SCSI devices, multiple processors, ...) Virtualization can also be used to simulate networks of independent computers.

Virtualization can enable existing operating systems to run on shared memory multiprocessors.

he need to run legacy applications is served well by virtual machines. A legacy application might simply not run on newer hardware and/or operating systems. Even if it does, if may under-utilize the server, so as above, it makes sense to consolidate several applications. This may be difficult without virtualization as such applications are usually not written to co-exist within a single execution environment (consider applications with hard-coded System V IPC keys, as a trivial example).

> Virtual machines can be used to create operating systems, or execution environments with resource limits, and given the right schedulers, resource guarantees. Partitioning usually goes hand-in-hand with quality of service in the creation of QoS-enabled operating systems.

> Virtual machines can isolate what they run, so they provide fault and error containment. You can inject faults proactively into software to study its subsequent behavior.

Virtual machines can be used to create arbitrary test scenarios, and can lead to some very imaginative, effective quality assurance.

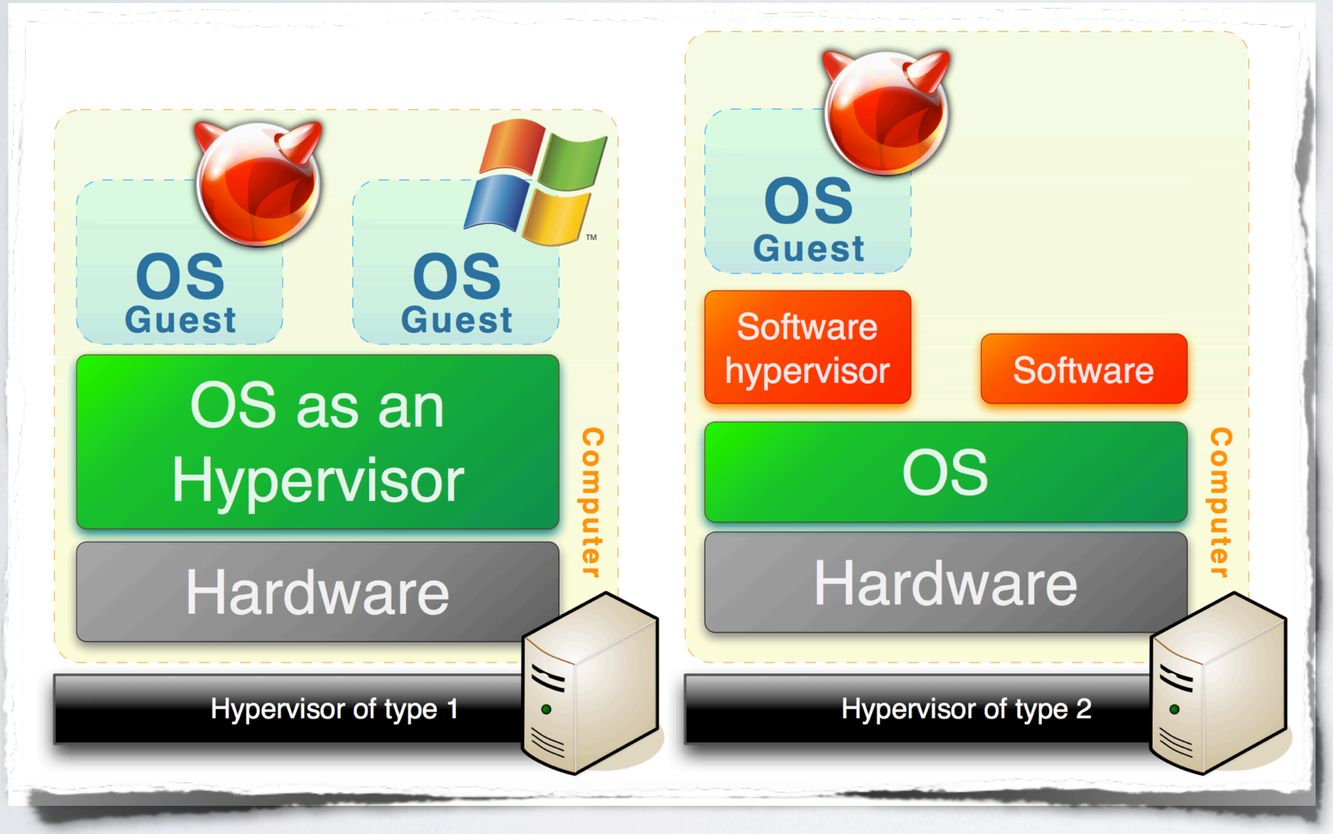
Virtualization can be used to retrofit new features in existing operating systems without "too much" work.

Virtualization can make tasks such as system migration, backup, and recovery easier and more manageable.

Green http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf

http://www.bitpipe.com/tlist/Virtualization.html http://www.kernelthread.com/publications/virtualization/

ANATOMIE D'UN HYPERVISEUR



- Un hyperviseur est la couche logiciels qui permet de partager le matériel présent
- Toutes les solutions de virtualisation ne sont pas égales

Les système d'exploitation virtualise l'accès aux ressources pour les processus. Un hperviseur fait la même chose, mais pour un système d'exploitation complet au lieu d'un processus.

ANATOMIE D'UN HYPERVISEUR

- Un hyperviseur est la colle qui permet aux systèmes invités de fonctionner en même temps que le système hôte sur une même machine

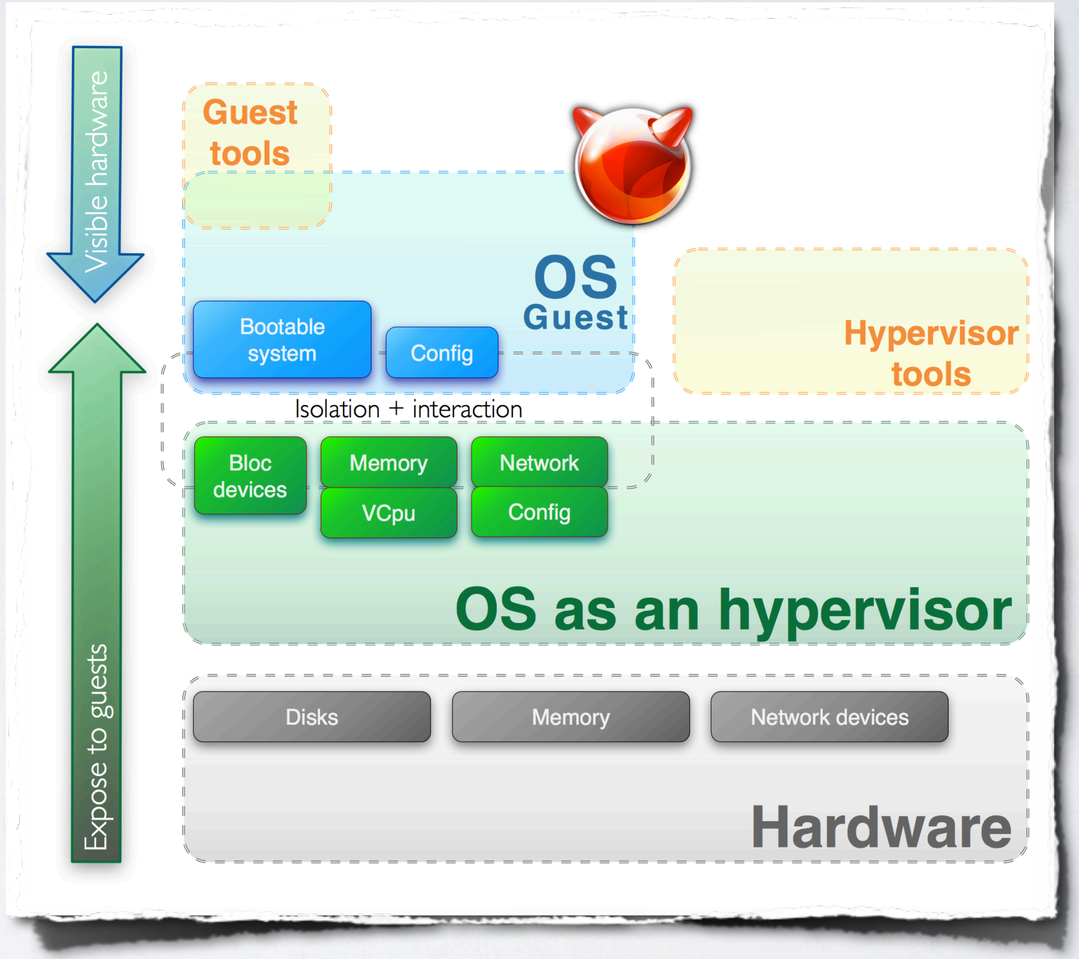
- Composants minimaux pour une virtualisation réussie :

  - Une image du système à démarrer par exemple : un noyau Linux

  - Quelques éléments de configuration (IP, mémoire allouées, nombre de vcpu, etc.)

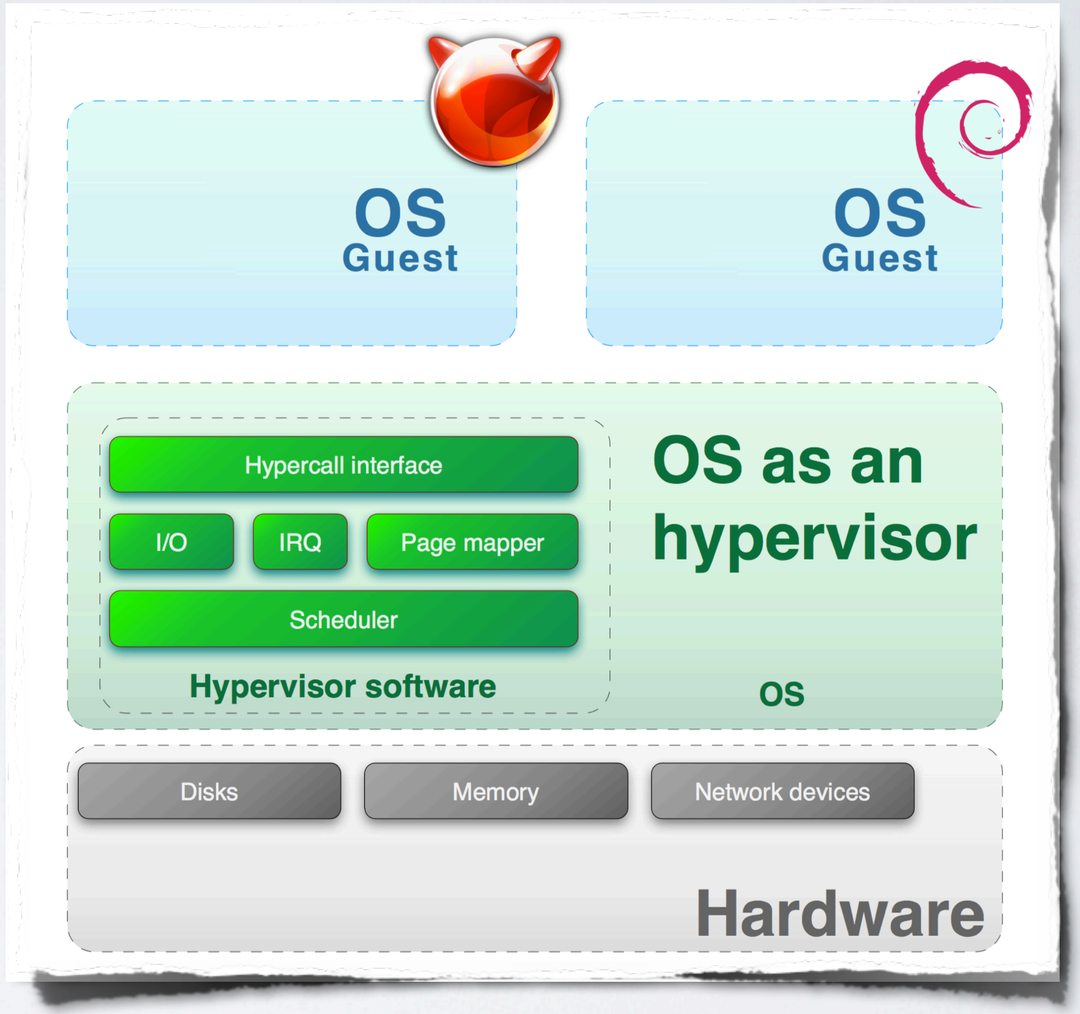
  - Un disque virtuel (de l'espace de stockage)

  - Une interface réseau (moyen de communication)



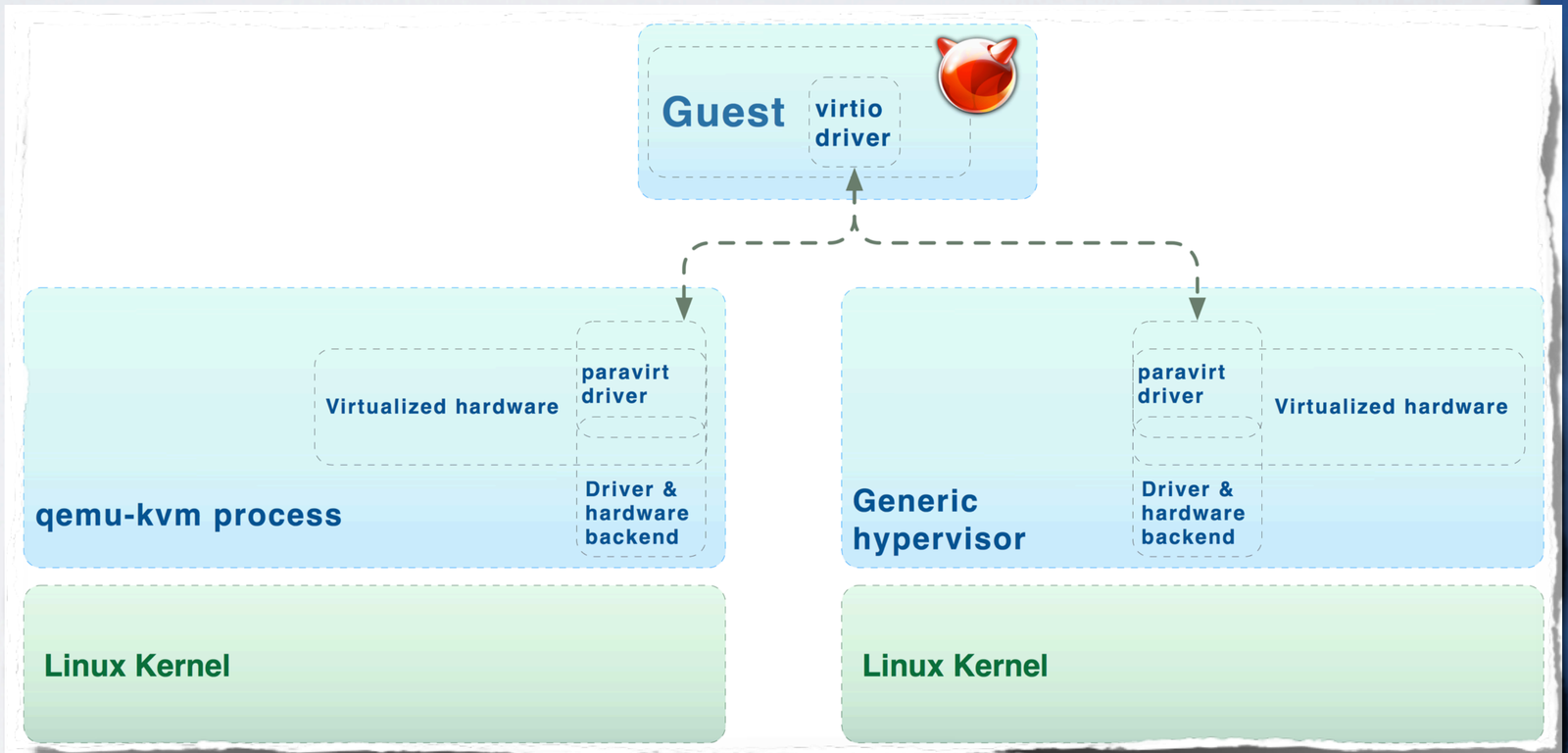
ANATOMIE D'UN HYPERVISEUR

- Un hyperviseur c'est au minimum :
  - Une gestion des IOs
  - Un gestion des interruptions
  - Une gestion de la mémoire
  - Un scheduler
- Les «hypercals» sont les communications entre les systèmes invités et l'hôte (exactement comme les «system calls» entre l'espace utilisateur et système d'un OS



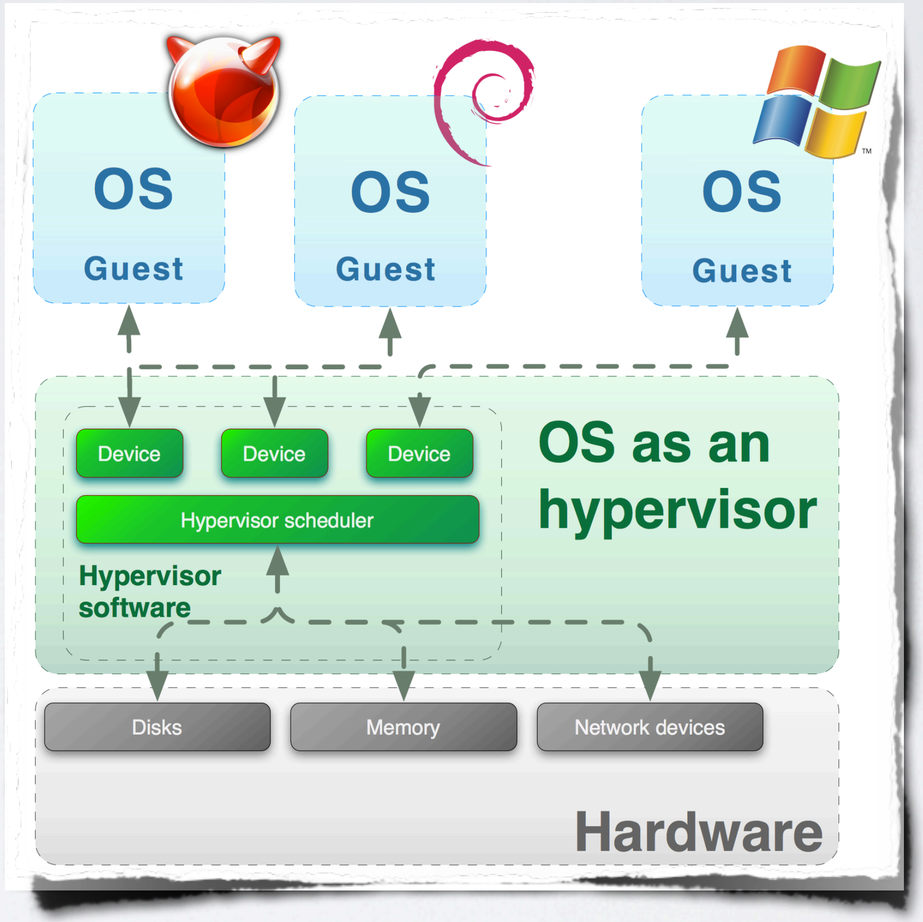
QUELQUES CONCEPTS

- Paravirtualisation
- Périphérique en espace hyperviseur
- Périphérique en espace utilisateur
- Pass through



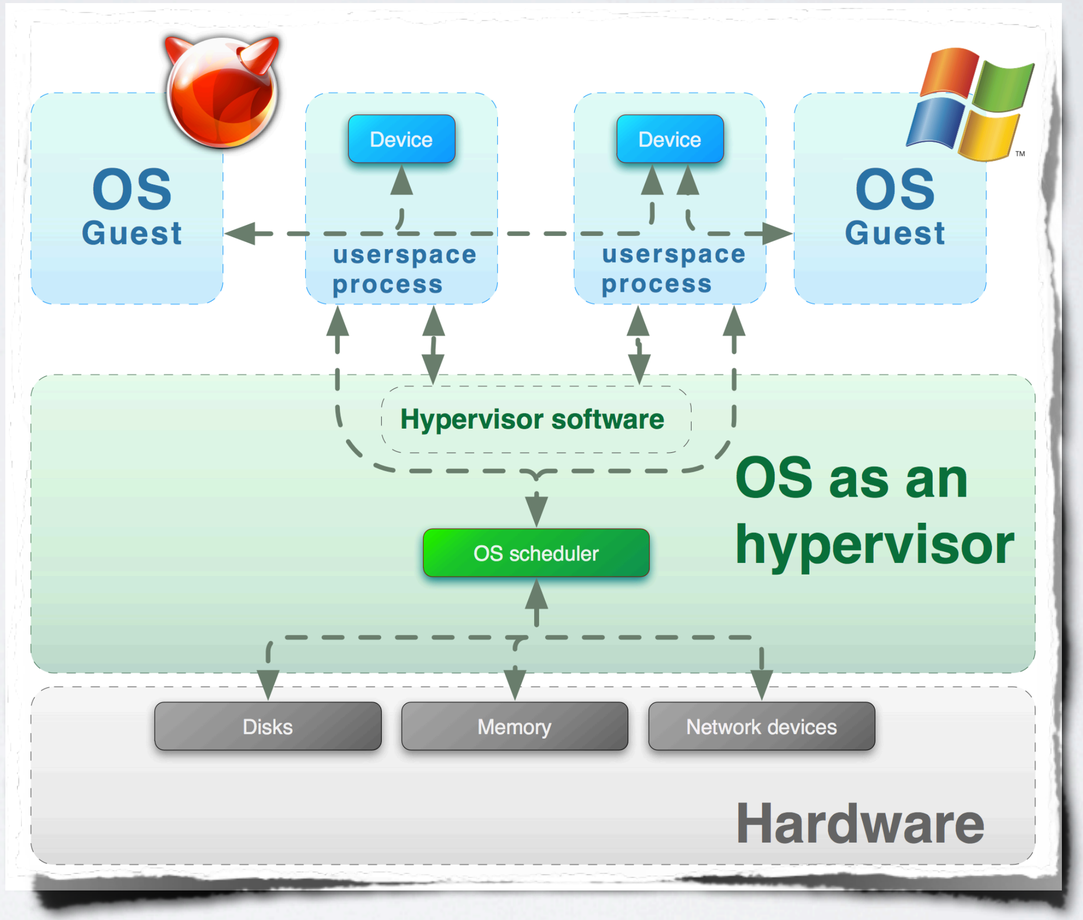
QUELQUES CONCEPTS

- Paravirtualisation
- Périphérique en espace hyperviseur
- Périphérique en espace utilisateur
- Pass through



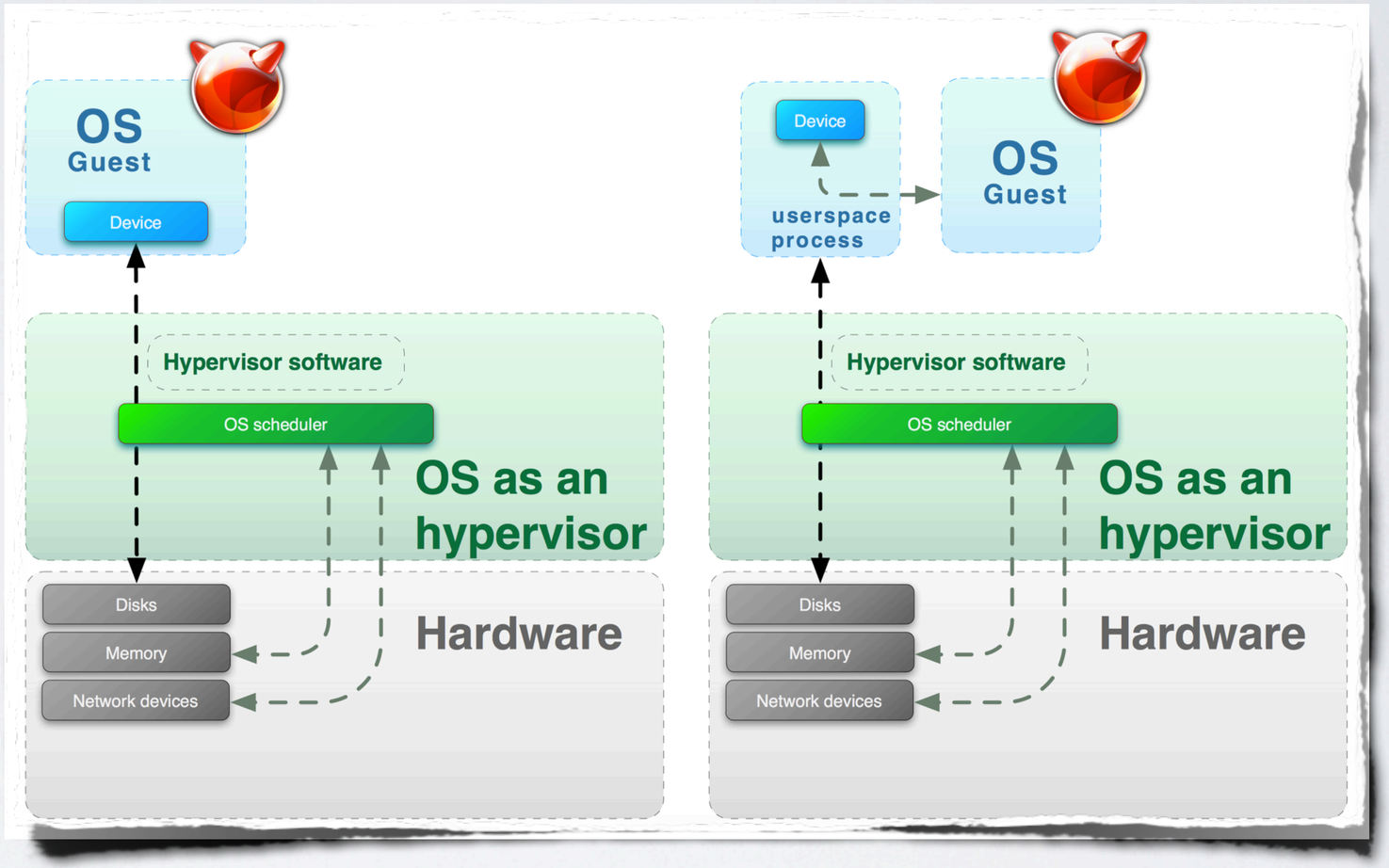
QUELQUES CONCEPTS

- Paravirtualisation
- Périphérique en espace hyperviseur
- Périphérique en espace utilisateur
- Pass through



QUELQUES CONCEPTS

- Paravirtualisation
- Périphérique en espace hyperviseur
- Périphérique en espace utilisateur
- Pass through



KVM

Petite visite de KVM

«A dreamer is one who can only find his way by moonlight, and his punishment is that he sees the dawn before the rest of the world»

Oscar Wilde

CONTEXTE DE KVM

- X86 n'est pas une architecture simple à virutaliser
  - Pas imaginée pour
  - Certaines instructions agissent différemment suivant les niveaux de privilèges
  - Les cpu avec des privilèges demandent de gérer des invités avec des niveaux de privilèges
    - Instructions INTEL / AMD
  - Souffre énormément du «trap-and-emulate»
- Quelques approches de la virutalisation x86
  - Binary translation
  - Se substituer en cas d'exécution de code privilégié
  - Demande beaucoup de remplacements pour garantir l'illusion
- La paravirtualisation
  - Le système invité est au courant de ces instructions restreignantes
  - L'hyperviseur propose des services pour les remplacer (hypercall)
  - On érode une couche d'abstraction pour améliorer les performances

PRÉSENTATION DE KVM

- Permettre l'utilisation des instructions INTEL/AMD en espace utilisateur
  - Exposer des fonctions de virtualisation de manière sécurisée
  - Une interface : /dev/kvm
  - Une intégration et une maintenance rapides
  - Disponible depuis le noyau 2.6.20 (2006)
    - Intégration au noyau linux en 3 mois (très rapide)
    - Construit de manière 100% parallèle au noyau Linux (suit les updates et les patchs)
    - Ajoute un mode de fonctionnement guest
- Evolution constante
  - Dernières évolutions de la virtualisation X86
  - Reconnu comme une partie intégrante de Linux
  - Portés sur plusieurs architectures (s390, PowerPC, IA64 et bientôt ARM

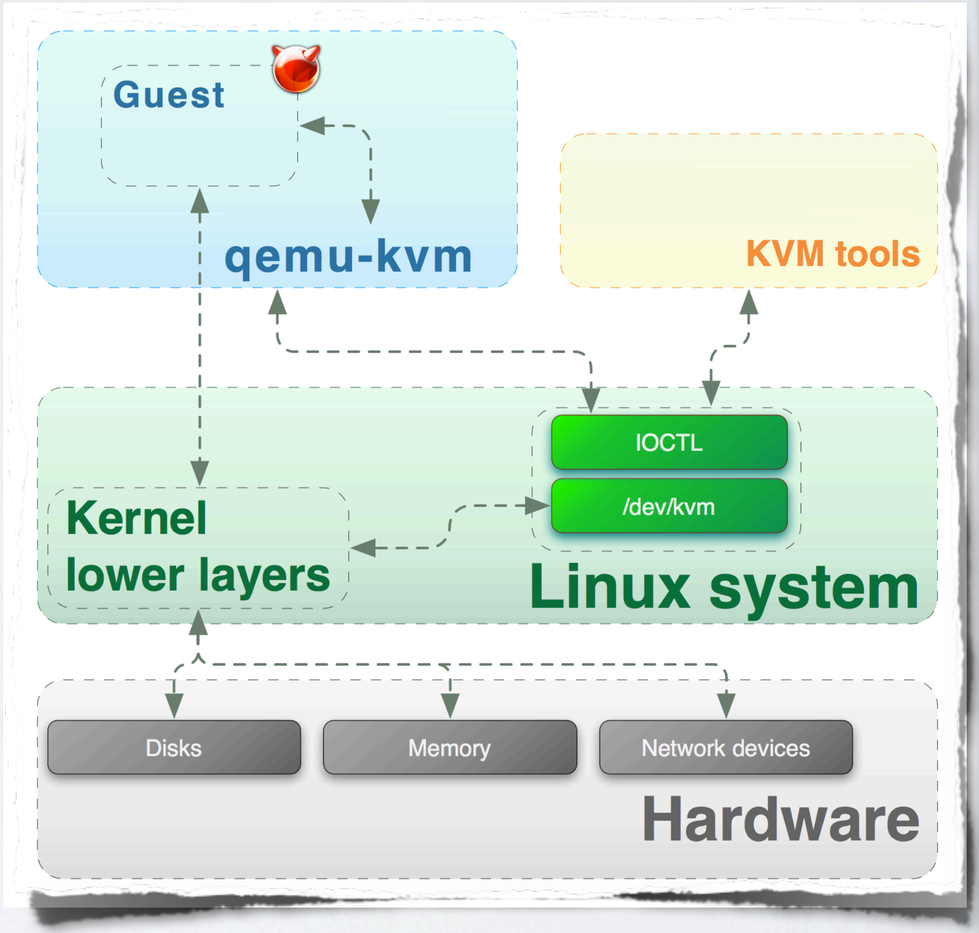
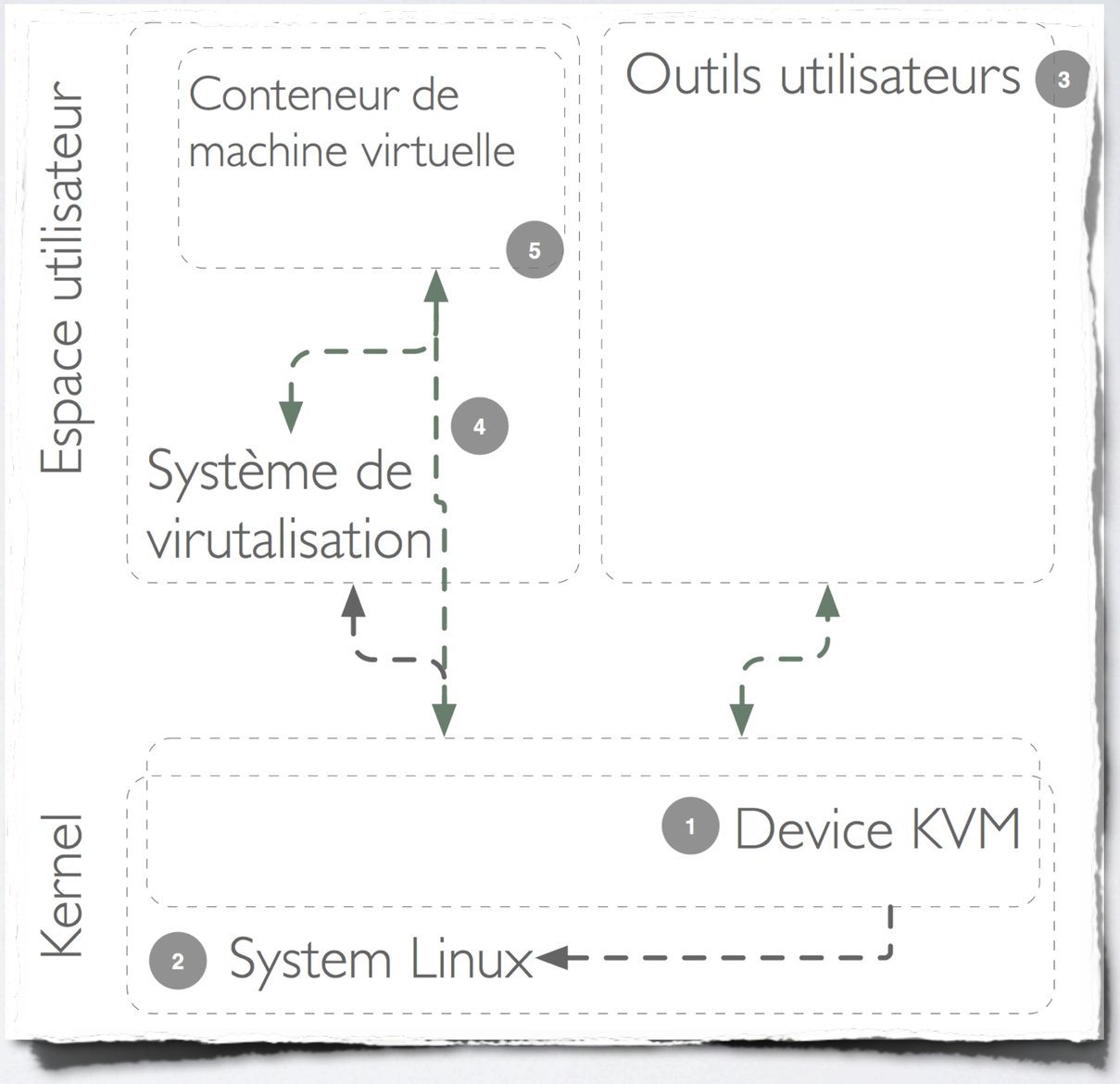


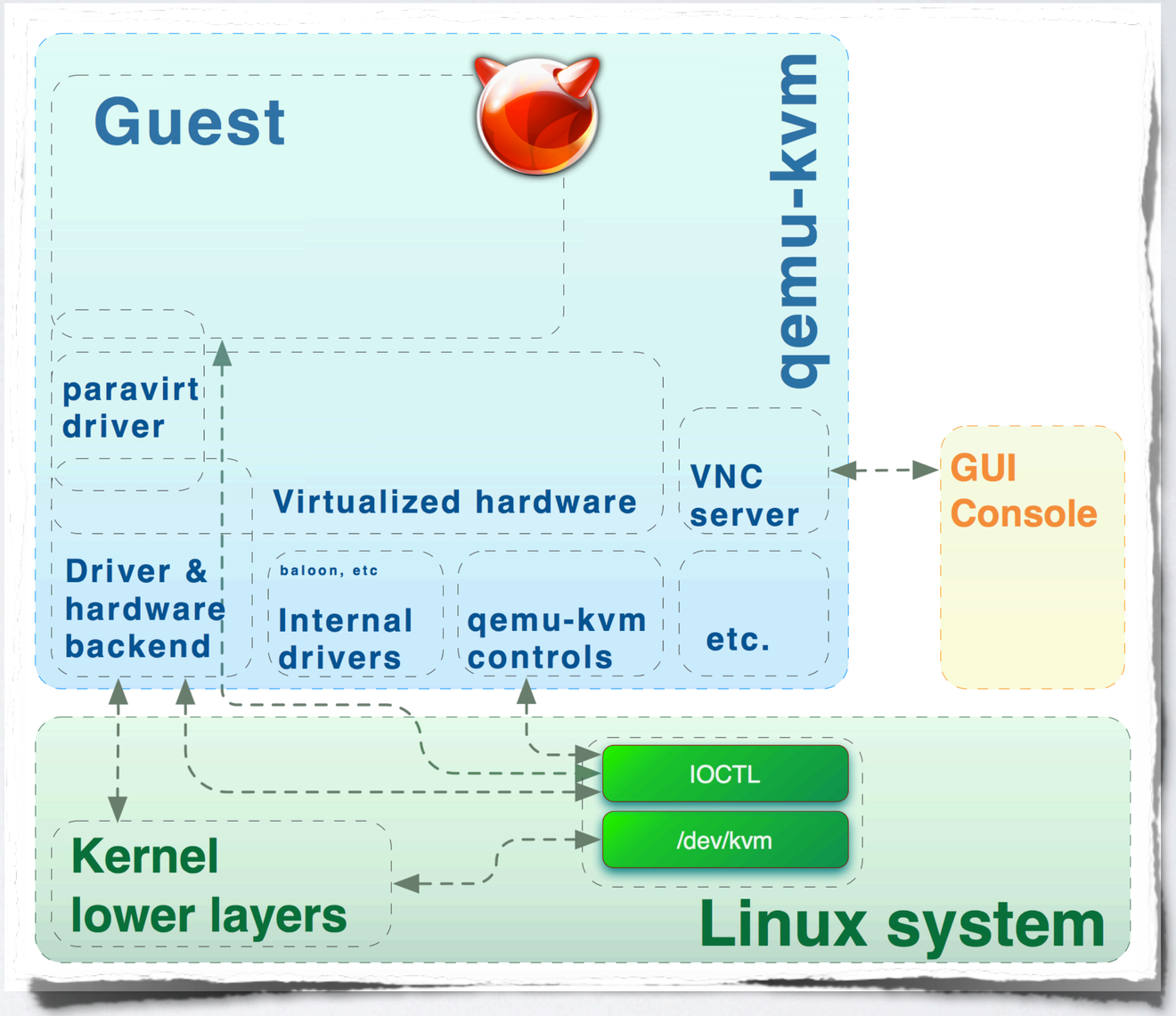
SCHÉMA BLOC DE KVM

- Kernel
  - 1 Le périphérique KVM
  - 2 Le système hôte
- Espace utilisateur (outils)
  - 3 Les outils d'administration
- Espace utilisateur (conteneur)
  - 4 Conteneur d'une machine virtuelle
  - 5 Machine virtuelle contenue



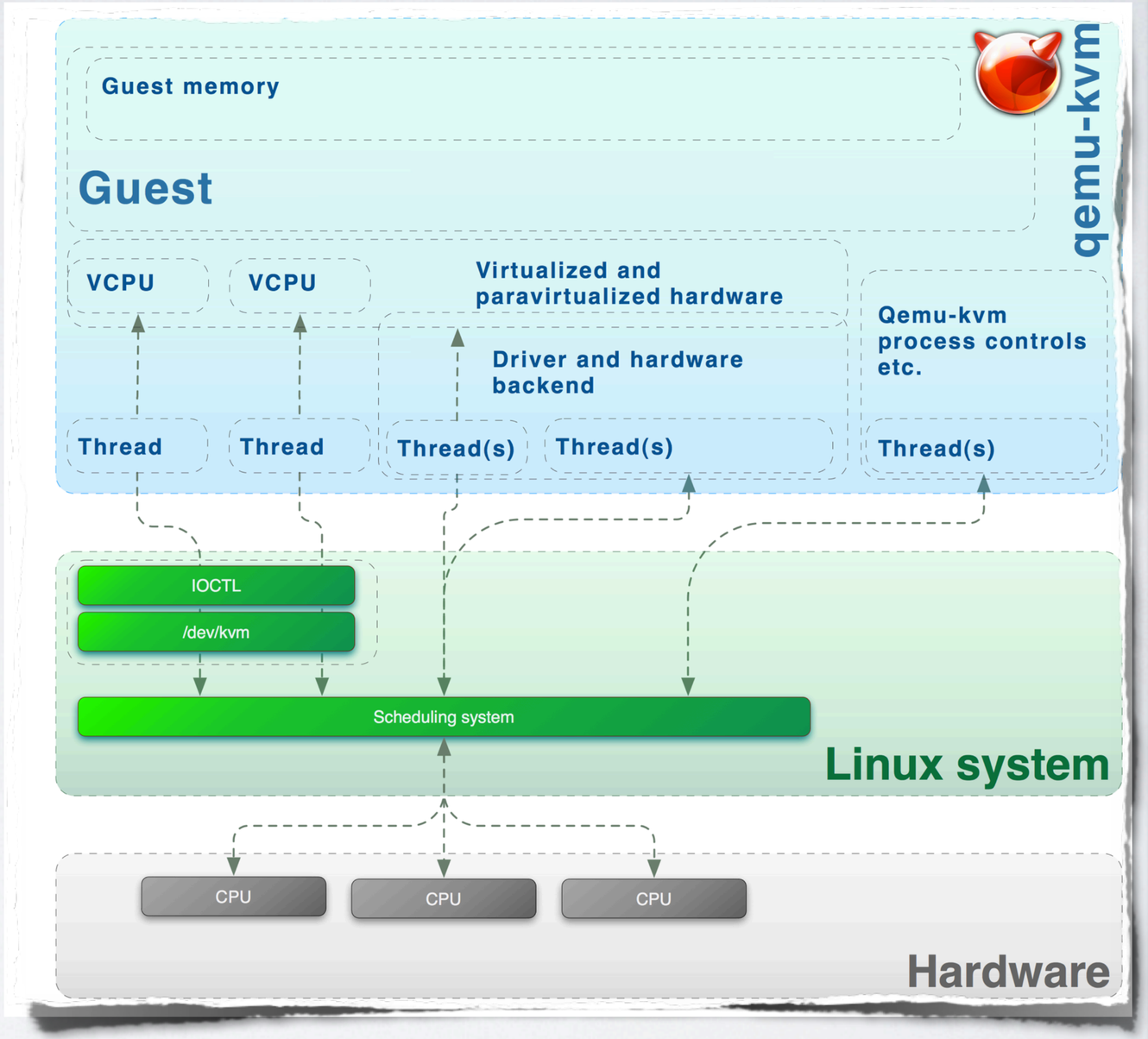
CONTENEUR DE MACHINE VIRTUELLE

- Le système invité n'est pas le seul consommateur de ressources
- 1 processus qemu-kvm par machine virtuelle
  - Du hardware virtualisé
  - Backend de la paravirtualisation
  - Mécaniques internes
  - Contrôles
  - Communications avec l'extérieur
    - Console, VNC
    - Bientôt SPICE (actuellement disponible)



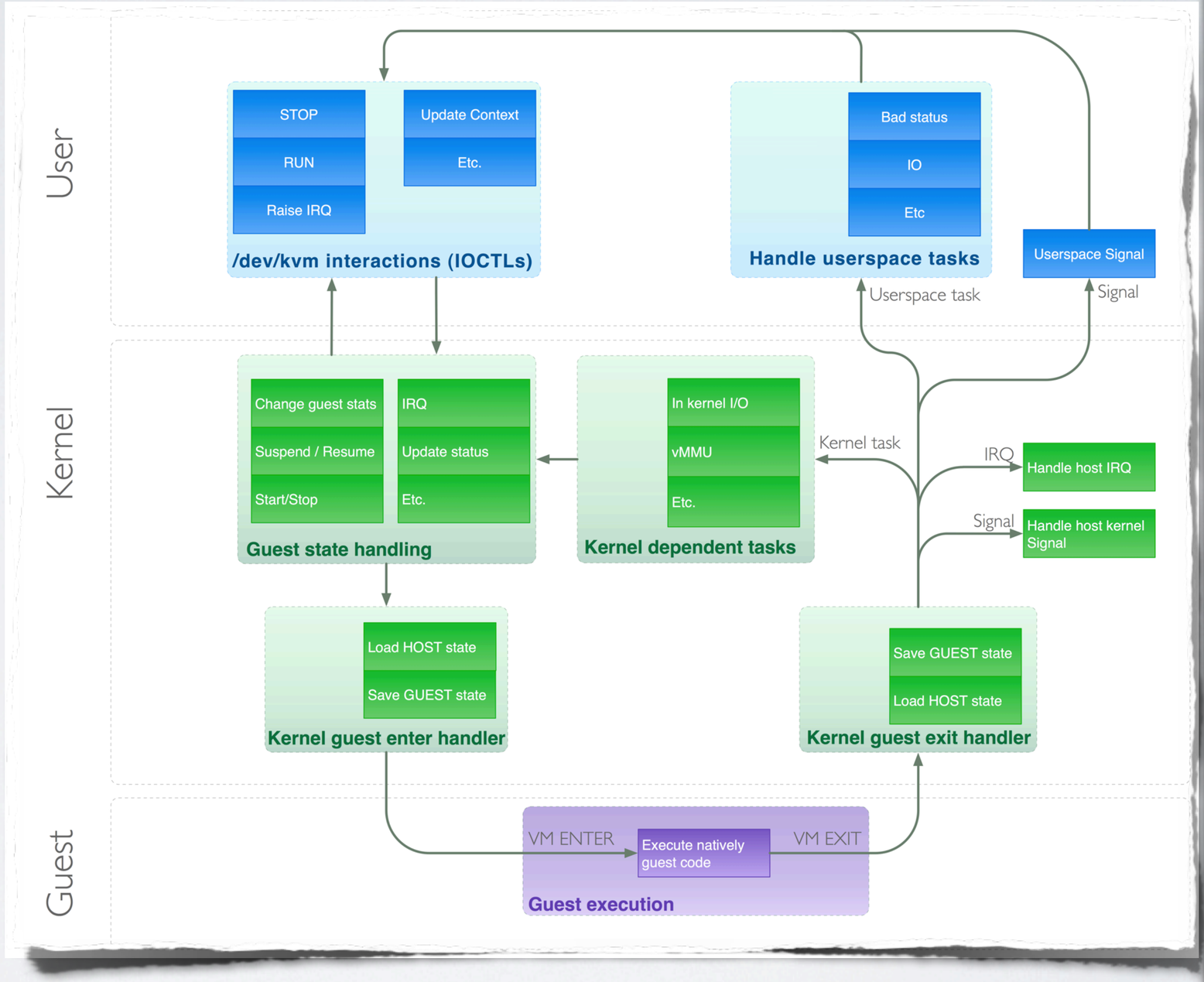
KVM EN FONCTIONNEMENT

- 1 thread par cpu virtuel
- Les drivers peuvent créer des threads
- Les processus de contrôle peuvent créer des threads
- Le device KVM en interne peut créer des threads
- Le noyau Linux par sa couche AIO, VFS, etc peut créer des threads



KVM EN FONCTIONNEMENT

- Ne pas confondre
  - Mode noyau
  - Ring cpu
  - Mode cpu
- Sortie lourde (retour en espace utilisateur) (Lightweight VM EXIT)
- Sortie simple (retour en espace kernel) (Heavy-weight VM EXIT)

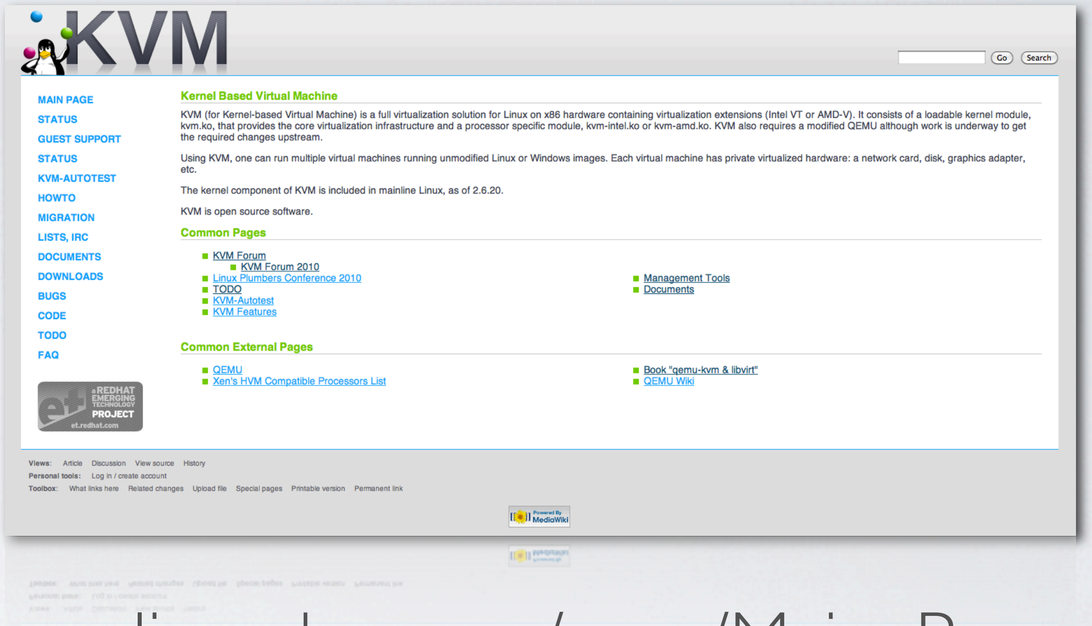


CONSIDÉRATIONS, LES PLUS ET LES MOINS

- L'inclusion dans le Kernel Linux apporte de nombreux avantages
  - Scheduler
  - Gestion de la mémoire (Huge page, KSM, etc.)
  - Couches IO
  - Gestion de l''énergie
  - CPU hotplugin, drivers, etc.
- L'inclusion dans Linux apporte de nombreux avantages
  - Gestion réseau
  - Manipulation des images
  - Sécurité
  - Stack logiciel de Linux
  - Logging, Tracing, Debugging, etc.
- Le chemin logiciel (code) entre les modes de fonctionnement
- Le coût des changements de mode
  - En moyenne 10 000 cycles cpu pour les changements lourds
- Exemple :
  - 1 : Changement d'état du hardware (VM EXIT).
  - 2 : Changement d'état du à du software (IOCTL, IRQ, Kernel, etc).
  - 3 : Analyse de la raison du changement d'état.
  - - : APIC Kernel : Lightweight VM EXIT
  - - : IO APIC ou PIC Kernel Lightweight VM EXIT
  - - : Réception d'un paquet réseau. Heavy-weight VM EXIT
  - - : Écriture d'un bloc sur un disque. Heavy-weight VM EXIT
  - 4 : Gestion software du changement d'état.
  - 5 : Changement d'état au niveau matériel (VM ENTRY).

AMÉLIORATIONS PROBABLES

- Les changements de mode :
  - Effectuer des changements partiels de mode
  - Changer de mode au dernier moment (paravirt)
  - Retarder les changements de mode dans le moteur de virtualisation avec des éléments pas forcément utilisés avec une grande fréquence
    - FPU
    - Debug registers
    - Model-specific registers (MSRs)
- Ces modifications demanderaient :
- Etre capable de détecter le mode guest
  - Très dépendant du matériel



http://www.linux-kvm.org/page/Main_Page

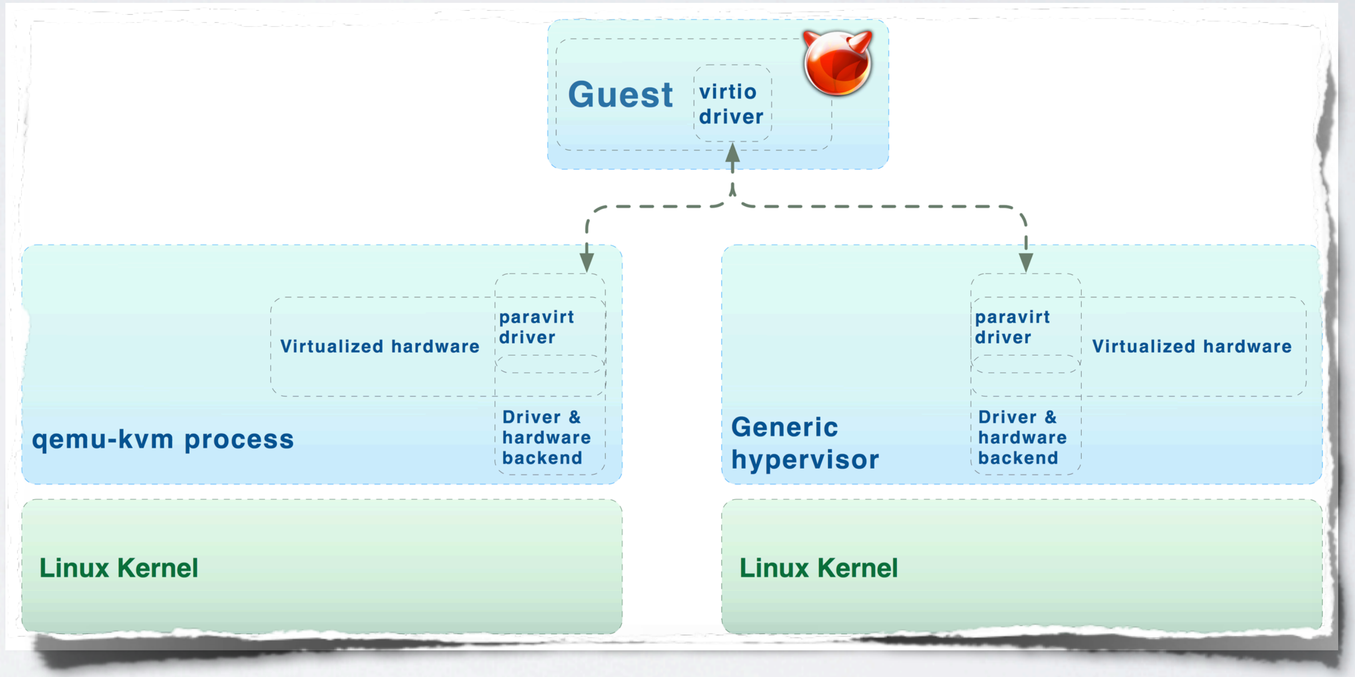
Environnement KVM

Les indispensables

«The hard must become habit. The habit must become easy. The easy must become beautiful.»
Doug Henning quotes (Canadian Magician)

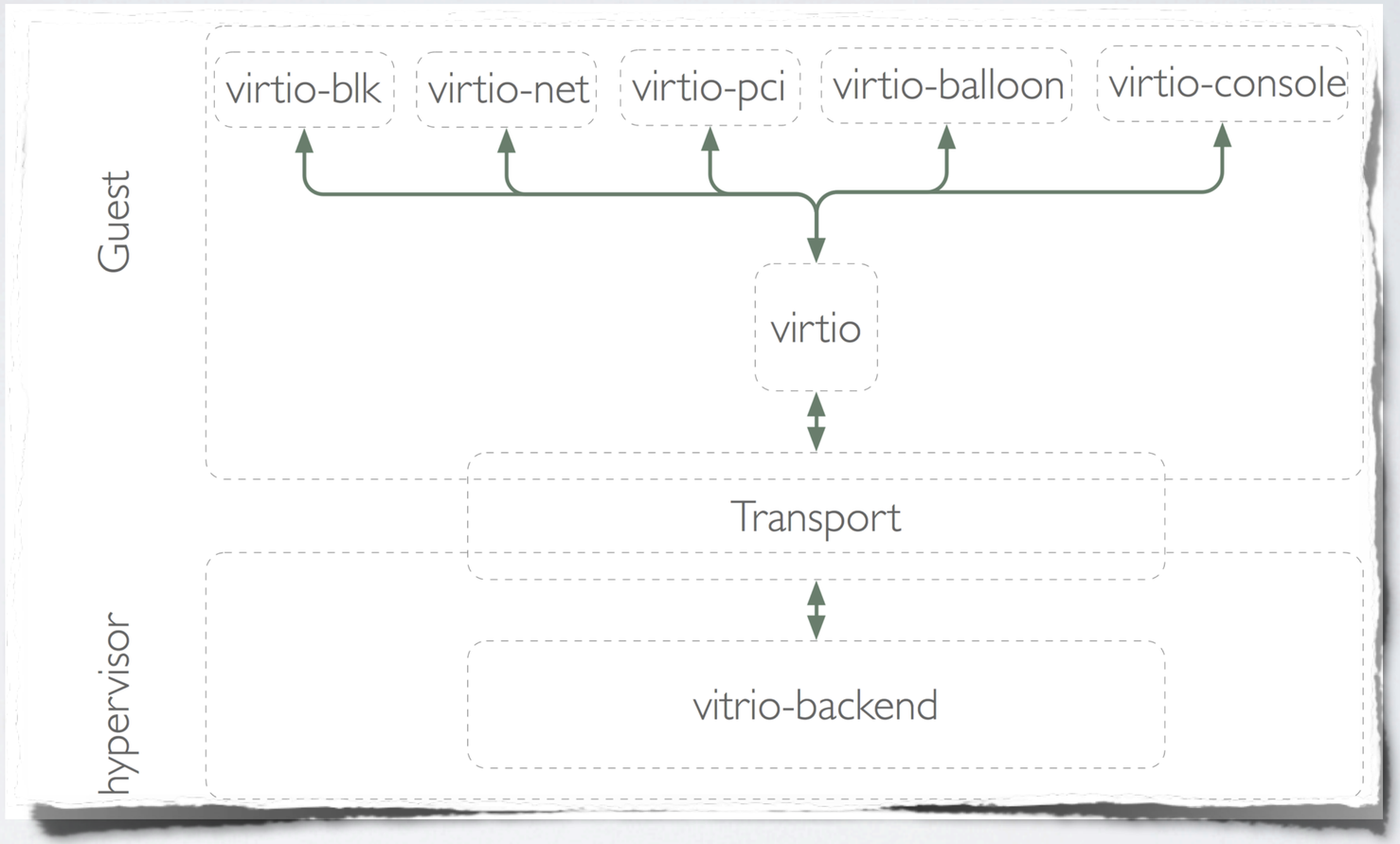
PARAVIRTUALISATION AVEC VIRTIO

- Originellement développée pour l'hyperviseur lguest
- Maintenant la solution de paravirtualisation par défaut installée dans le kernel Linux
  - >=2.6.25



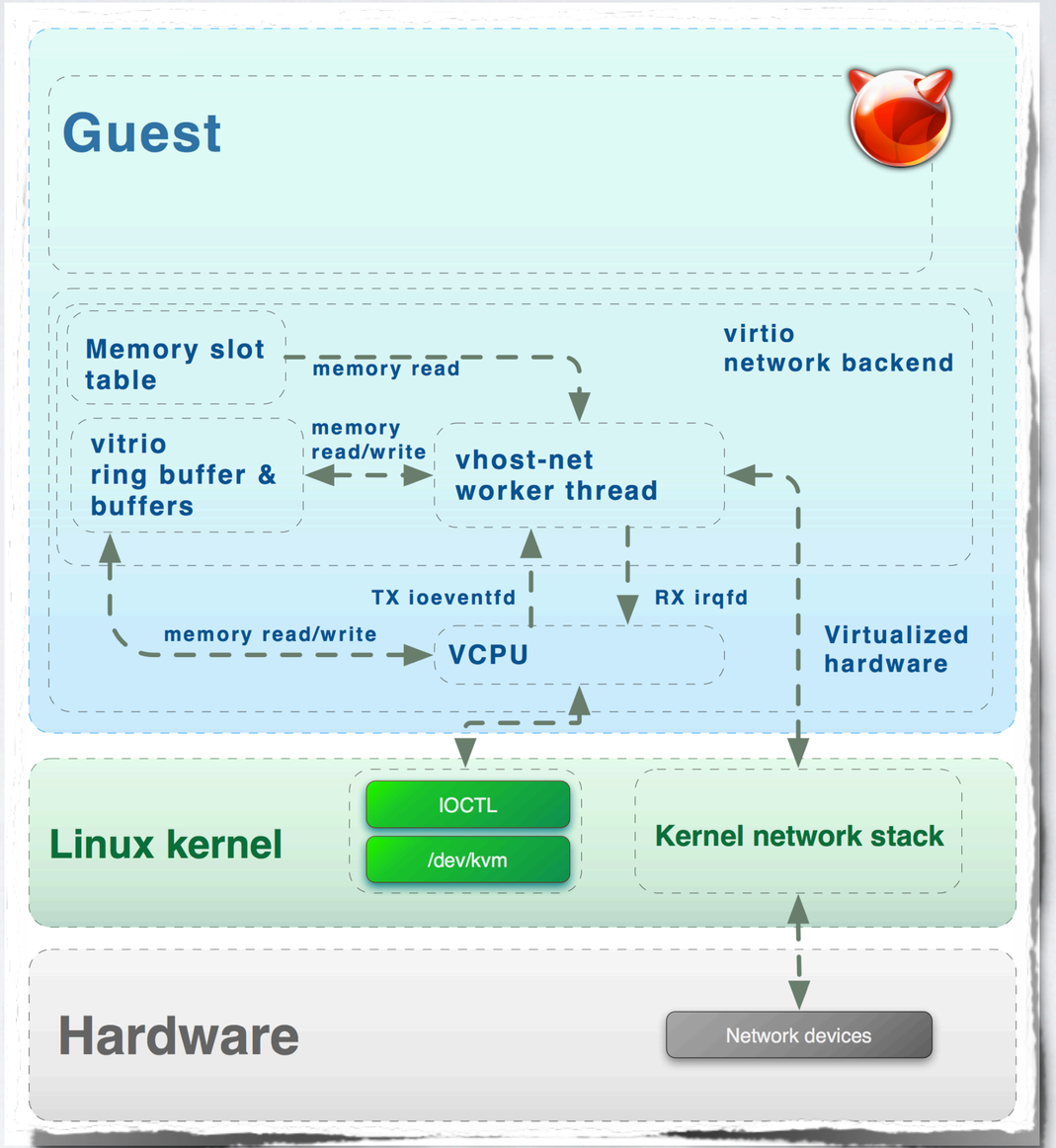
PARAVIRTUALISATION AVEC VIRTIO

- Propose une panoplie de périphériques
  - Type block
  - PCI passthrouhg
  - Console
  - Balloon (mémoire dynamique)
  - Network
- Communication à base de tampons
  - ex :
    - 1 tampon pour les requêtes RX
    - 2 tampons pour les réponses TX
- Canal de communication «Transport» générique.
- Intelligence dans le driver et dans le backend



AMELIORATION SUR LA PARAVIRTUALISATION

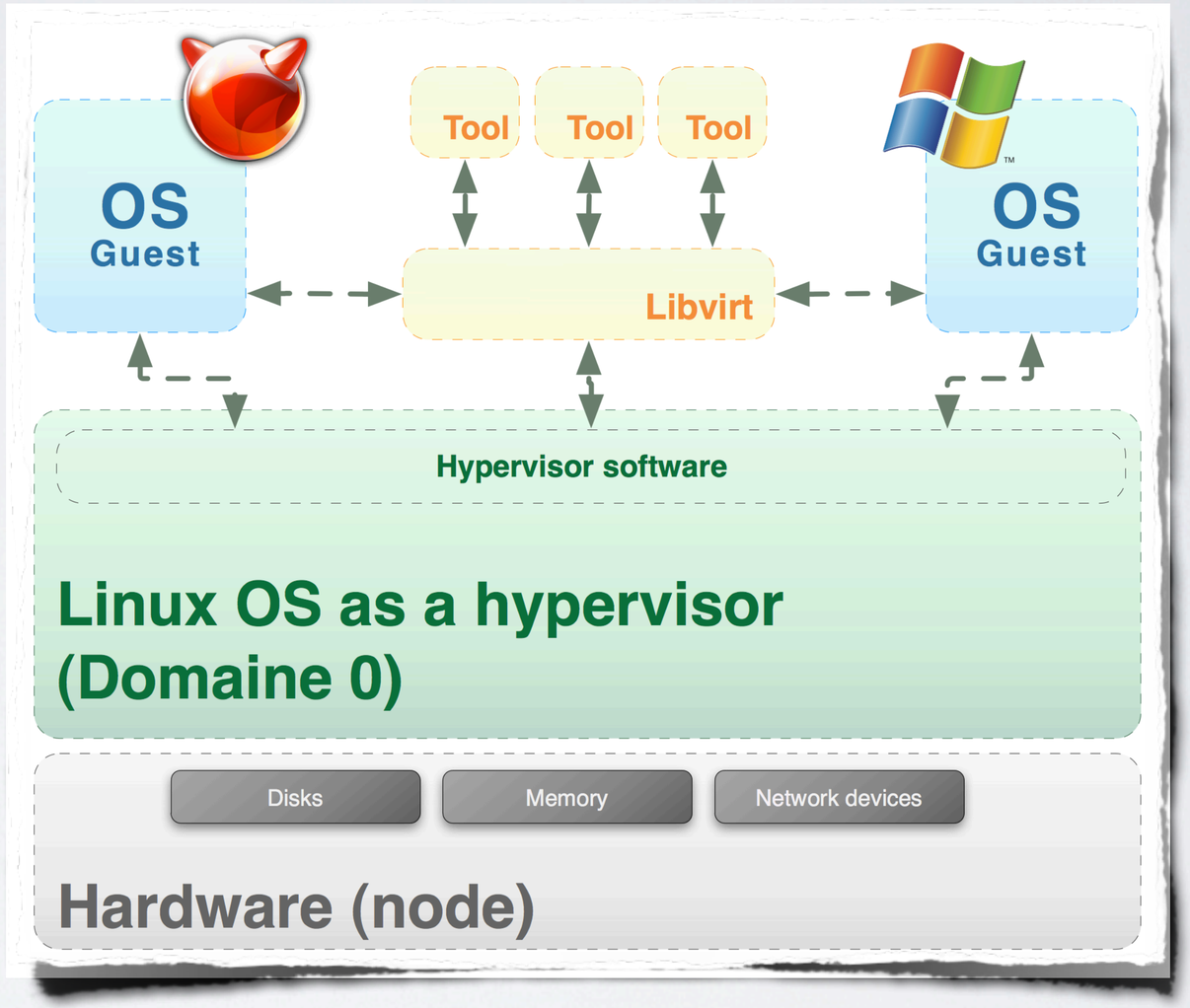
- Optimisations du côté Linux
  - Huge page (plus de données transférées)
  - Optimisation NUMA du scheduler (accès à la mémoire)
- Améliorer les effets de bord des «spin-lock-holder» (attentes actives)
- Limiter les copies inutiles de données
- Hardware conçu dans une optique de virtualisation



VirtIO Network backend

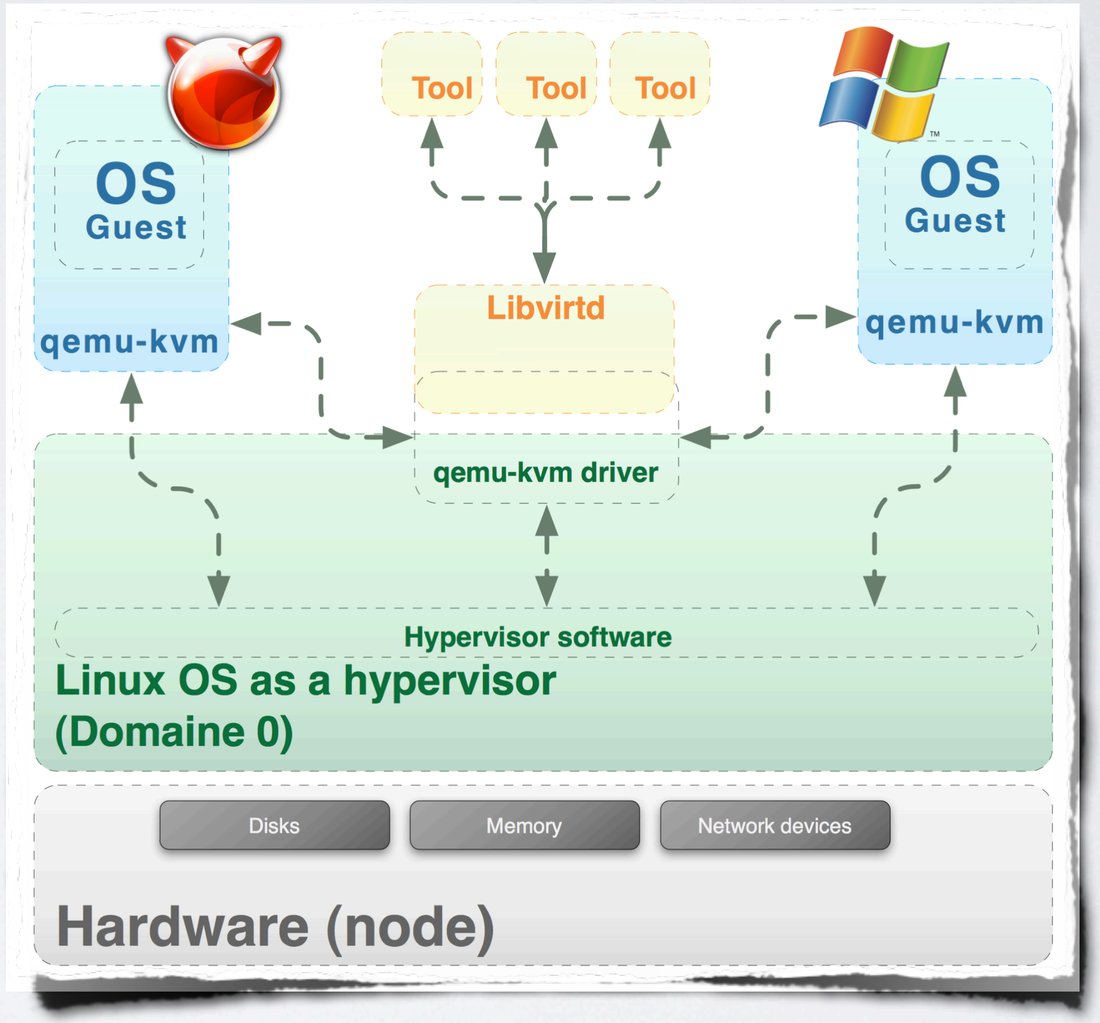
LIBVIRT LA LIBRAIRE MULTIFONCTIONS

- Proposer un set de fonctionnalités communes à plusieurs hyperviseurs
  - 1 driver par hypervieur
  - Plusieurs hyperviseurs par host possibles
  - Tous les hyperviseurs ne proposent pas toutes les fonctionnalités que libvirt propose
  - Libvirt ne propose pas forcément toutes les fonctionnalités qu'un hyperviseur propose
- Terminologie
  - Host = Node
  - Hyperviseur = Domain0
  - Machine virtuelle = Domaine test1, Domaine toto, etc..
- Lien entre hyperviseur et outils réalisé par le démon libvirtd
- libvirtd permet une communication à distance via SSH
- Libvirt existe sur de très nombreux langages
- Technologie de base de la majorité des outils de virtualisation



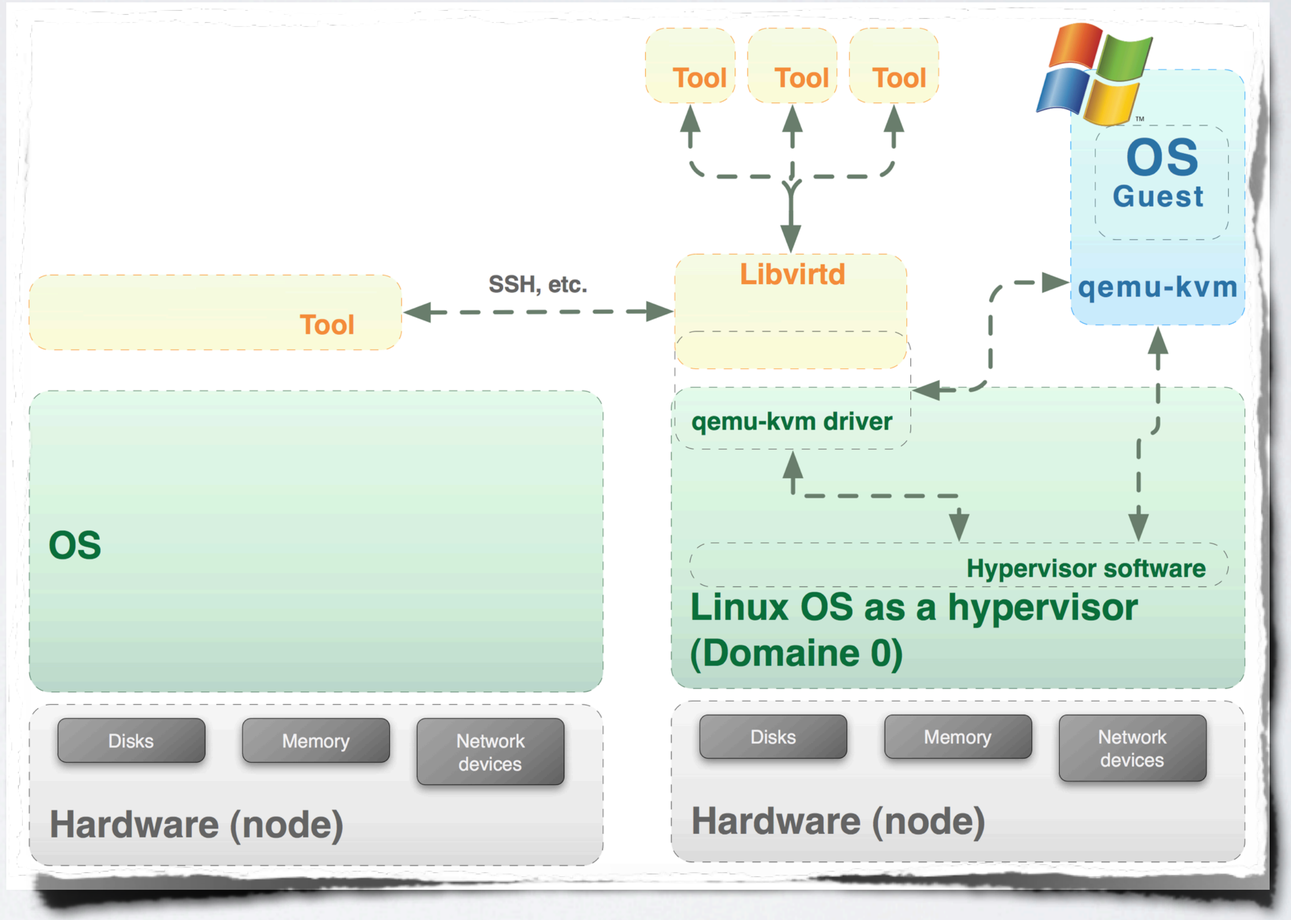
LIBVIRT LA LIBRAIRE MULTIFONCTIONS

- Proposer un set de fonctionnalités communes à plusieurs hyperviseurs
  - 1 driver par hypervieur
  - Plusieurs hyperviseurs par host possibles
  - Tous les hyperviseurs ne proposent pas toutes les fonctionnalités que libvirt propose
  - Libvirt ne propose pas forcément toutes les fonctionnalités qu'un hyperviseur propose
- Terminologie
  - Host = Node
  - Hyperviseur = Domain0
  - Machine virtuelle = Domaine test1, Domaine toto, etc..
- Lien entre hyperviseur et outils réalisé par le démon libvirtd
- libvirtd permet une communication à distance via SSH
- Libvirt existe sur de très nombreux langages
- Technologie de base de la majorité des outils de virtualisation



LIBVIRT LA LIBRAIRE MULTIFONCTIONS

- Proposer un set de fonctionnalités communes à plusieurs hyperviseurs
  - 1 driver par hypervieur
  - Plusieurs hyperviseurs par host possibles
  - Tous les hyperviseurs ne proposent pas toutes les fonctionnalités que libvirt propose
  - Libvirt ne propose pas forcément toutes les fonctionnalités qu'un hyperviseur propose
- Terminologie
  - Host = Node
  - Hyperviseur = Domain0
  - Machine virtuelle = Domaine test1, Domaine toto, etc..
- Lien entre hyperviseur et outils réalisé par le démon libvirtd
- libvirtd permet une communication à distance via SSH
- Libvirt existe sur de très nombreux langages
- Technologie de base de la majorité des outils de virtualisation



Pratique

Obtenir KVM et créer des machines virtuelles

«An ounce of practice is worth more than tons of preaching.»

Mohandas Gandhi

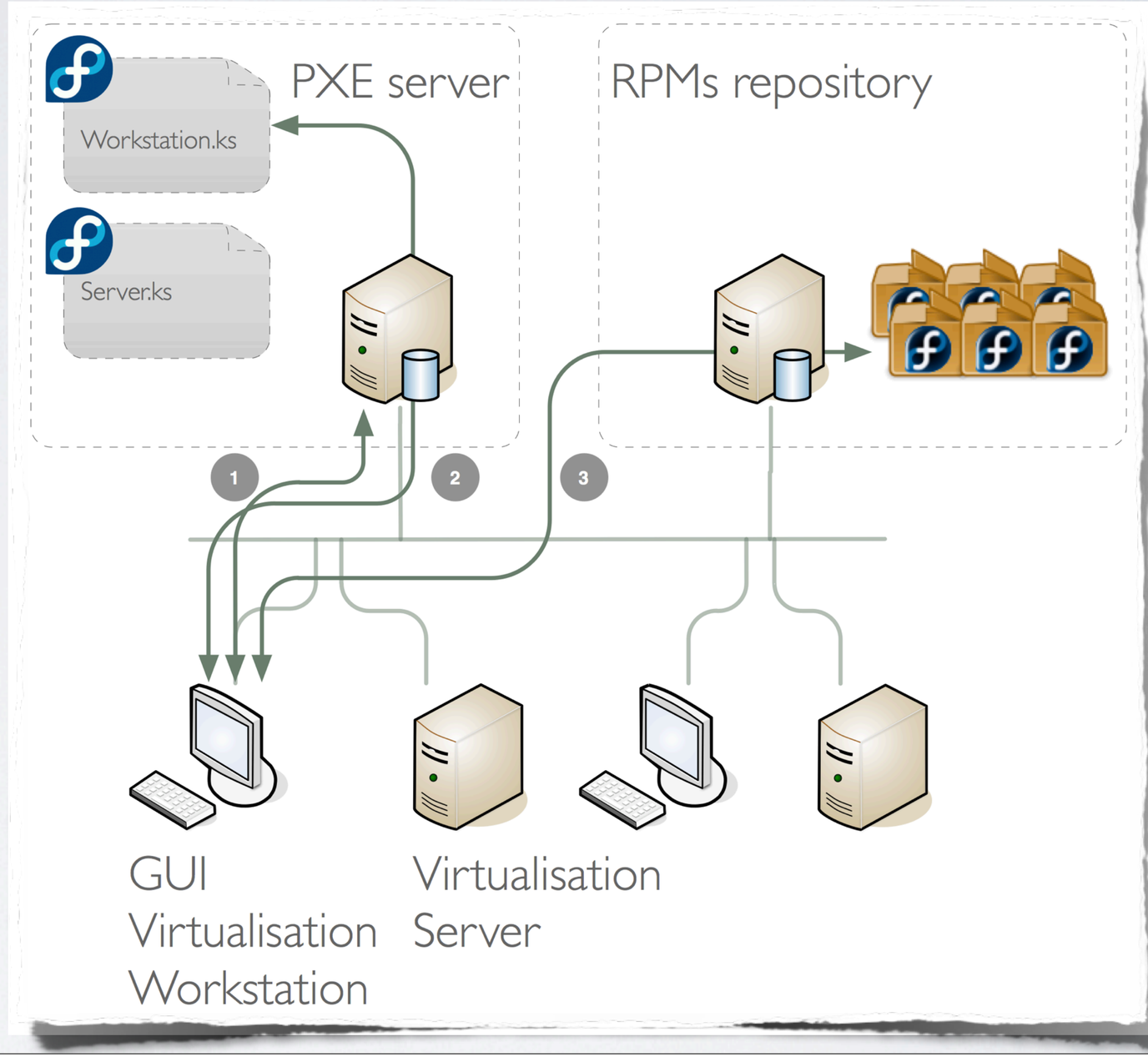
INSTALLER KVM

- Disponible sur la plupart des distributions Linux
  - EX : Fedora : #yum groupinstall virtualization
- Solution pour une installation automatisée
- NetInstall commander par Kickstart (appliquée à hepia pour les tests)
- NetInstall commander par Kickstart démarrée par un boot PXE (implémentée à hepia)
- Live-CD de virtualisation
- Etc.

INSTALLER KVM

Proposition HEPIA

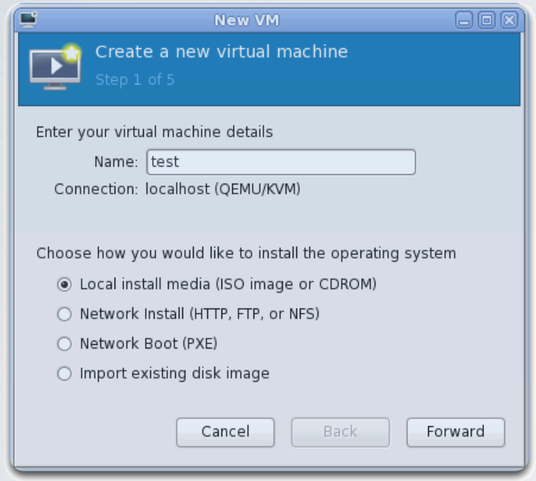
- 1 Boot PXE
- 2 Sélection (serveur ou workstation)
- 3 Installation avec les paquets du dépôt

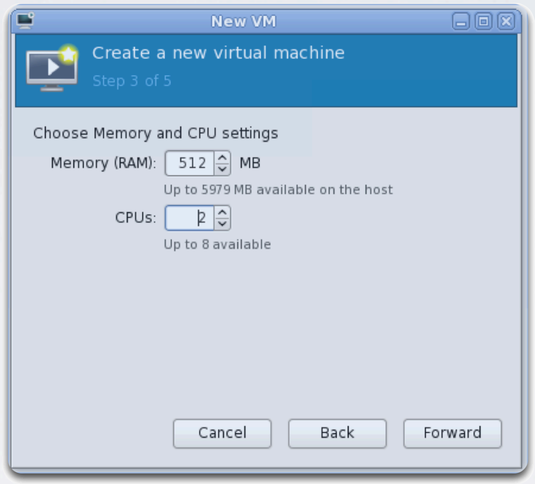


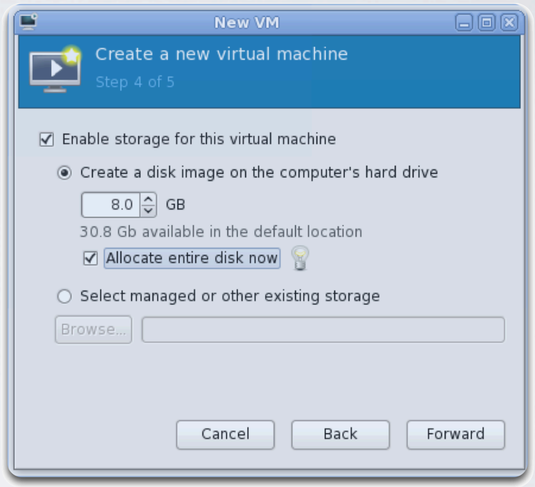
CRÉATION DE MACHINES VIRTUELLES

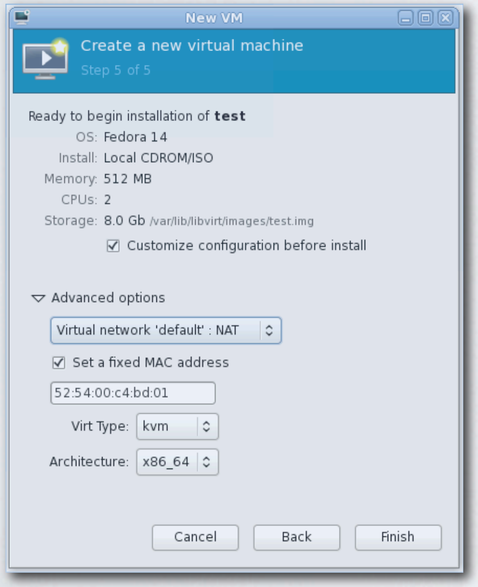
- Via qemu-kvm
  - qemu-img create -f qcow2 <Image_Name> <size>
  - qemu-kvm -hda <Image_Name> -m 512 -cdrom </Path/to/the/ISO/Image> -boot d -vga std
  - Par défaut, une machine n'est pas prise en charge par libivrt
    - Il faut ajouter la machine (engendre une redéfinition du hardware)
      - virt-install --import --accelerate --disk path=<Image_Name> --os-type linux --os-variant <OS> --ram 512 --name <New Name>
- Il est possible de créer et/ou installer directement dans libvirt
  - virt-install --ram=512 --vcpus=2 --vnc --network=bridge:virbr0 --keymap=fr-ch --os-type=linux --os-variant=fedora14
- Libvirt gère ses machines avec un format XML
  - Il est possible d'extraire le XML
    - virsh dumpxml <domaine name>
  - Il est possible de construire une machine à partir du XML
    - virsh create domaine.xml

CRÉATION DE MACHINES VIRTUELLES

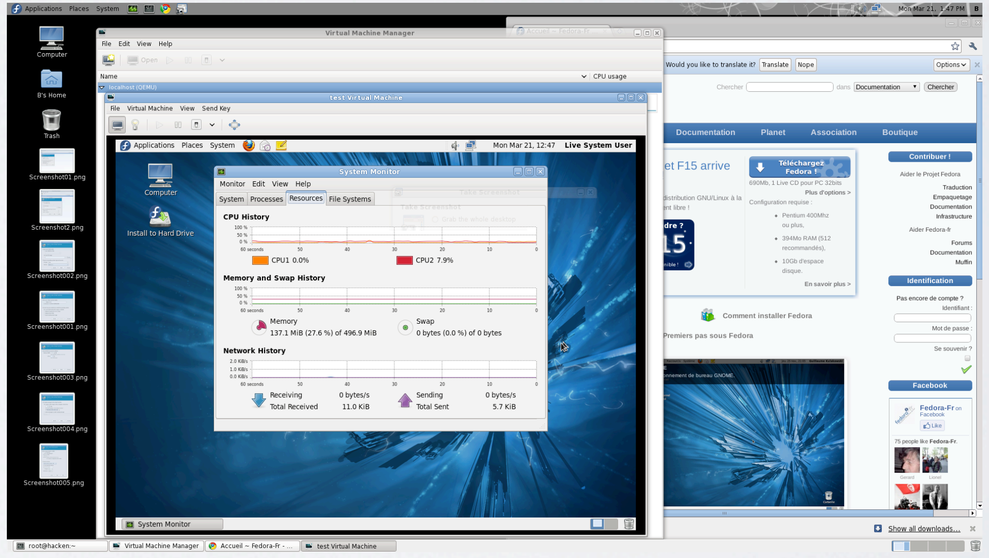






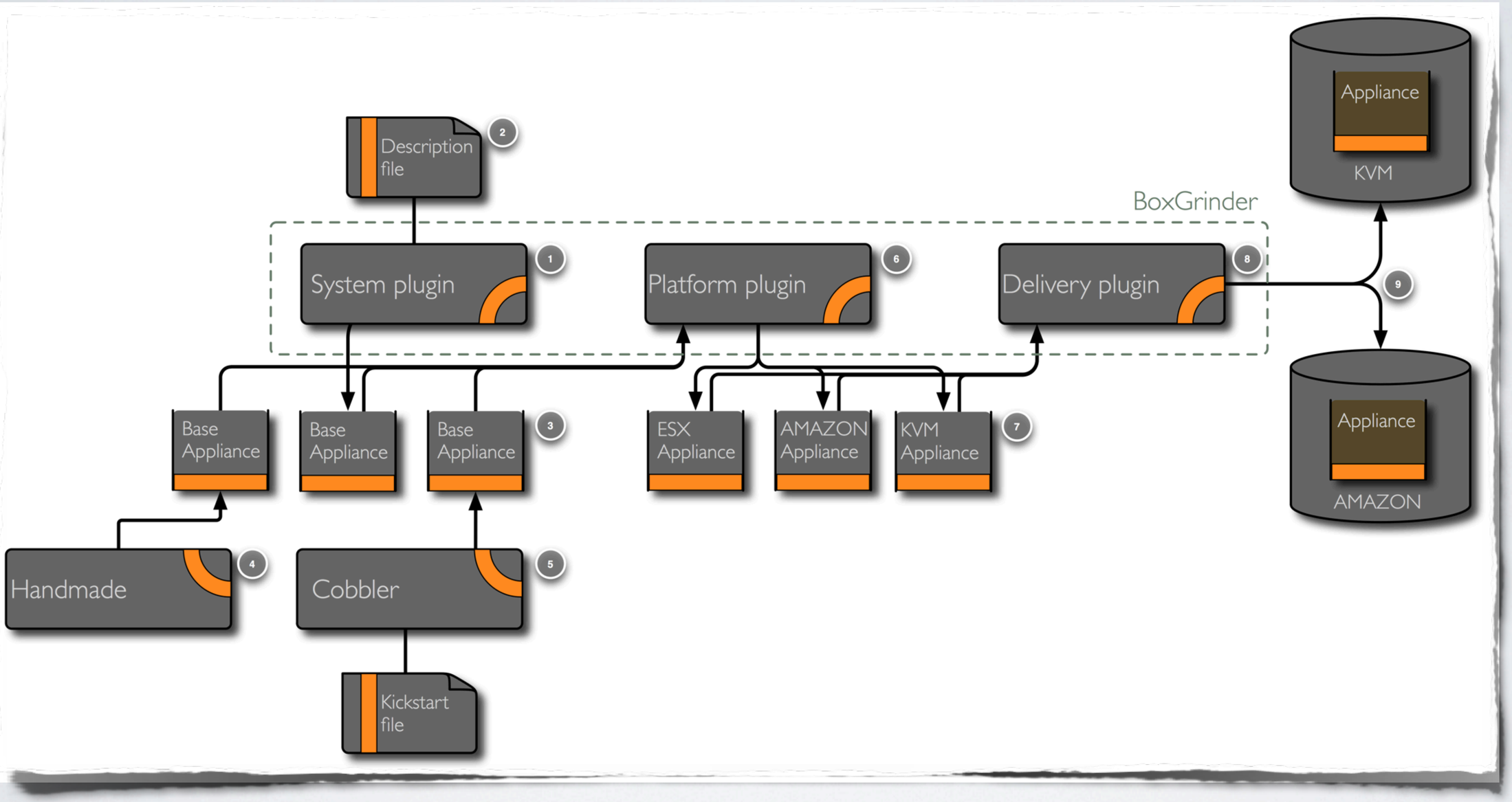


Utiliser Virt-Manager



AUTOMATISATION

Approche par BoxGrinder



FIN

L’heure du bilan

«vir bonus est is, qui prodest quibus potest, nocet nemini»

Cicéron, De Officiis, 3.64

BILAN DE PROJET

- Rappel des objectifs principaux
  - Fournir au laboratoire de virtualisation de hepia une première analyse technique de la solution KVM
    - Les objectifs ont été atteints et le rapport fournit une documentation de l'architecture, des forces et des faiblesses de KVM et des principaux outils importants
  - Aider à préparer la migration vers un environnement KVM complet et automatisé
    - L'environnement à pu être spécifié et les premières phases de migration lancées
  - Si le temps le permet, analyser la solution sheepdog
    - Le temps et le status «en développement» de la solution n'ont pas permis de créer une documentation à ce sujet.

FIN

BILAN DE KVM

- KVM est clairement une solution moderne d'avenir
- KVM demande des connaissances Linux pour être utilisé de manière avancée
- Il manque des grands outils de management
  - La demande créera l'offre
- La solution est soutenue par des grands noms de l'informatique
- Depuis le 17 mai 2011, IBM, Intel, Red Hat, HP, BMC Software, Eucalyptus Systems, et SUSE ont créés un groupement (Open Virtualization Alliance) dans le but de développer et de promouvoir KVM comme la solution de virtualisation du futur.

http://www.openvirtualizationalliance.org/

CE QUI AURAIT PU ÊTRE MIEUX RÉALISÉ

- L'analyse et la documentation de KVM ont pris réellement plus de temps que prévu
- Des parties pratiques n'ont pu être documentées pour des raisons de temps et d'ampleur du rapport
  - Couche assembleur X86
  - Gestion mémoire avec KSM et MOM

FIN

ANALYSE DES META-COMPÉTANCES

- Sait choisir et appliquer la méthode adéquate de projet
- Sait exploiter les ressources internes et identifier les ressources
- A le sens de l'initiative personnelle et des responsabilités

- Est capable de spécifier, planifier, concevoir, … les normes et standards
- Est en mesure de proposer et comparer des solutions
- Est capable de se mettre à la place de l'utilisateur

- Simplification du cahier des charges
- Sélection des priorités et des objectifs
- Choix de ne pas tout montrer dans le rapport pour privilégier les objectifs prioritaires
- Travailler en collaboration avec les assistants et diplomants
- Remise en question des bonnes pratiques
- Evaluer les solutions pour l'environnement hepia
- Choisir les solutions adaptées au client

FIN

ANALYSE DES META-COMPÉTANCES

- Sait utiliser à bon escient les concepts … stockage de l'info
- Sait appliquer des méthodologies de travail appropriées

- Simplification du cahier des charges
- Travail en collaboration avec le professeur et mandant

- A appris à auditer un système d'info
- A appris à auditer la sécurité d'un SI

- Analyse et critiques des solutions évaluées

Questions ?

Sébastien Pasche

sebastien. pasche@heig-vd.ch

09.06.2011