KERLABS

Déploiement d'un cluster - Kerrighed

Jean Parpaillon

Sommaire

- **Kerlabs**
  - Présentation
  - Nos métiers
  - Nos activités
- **Kerrighed**
  - Contexte
  - Principes du SSI
  - Présentation de Kerrighed
  - Déploiement de Kerrighed
  - Utilisation de Kerrighed

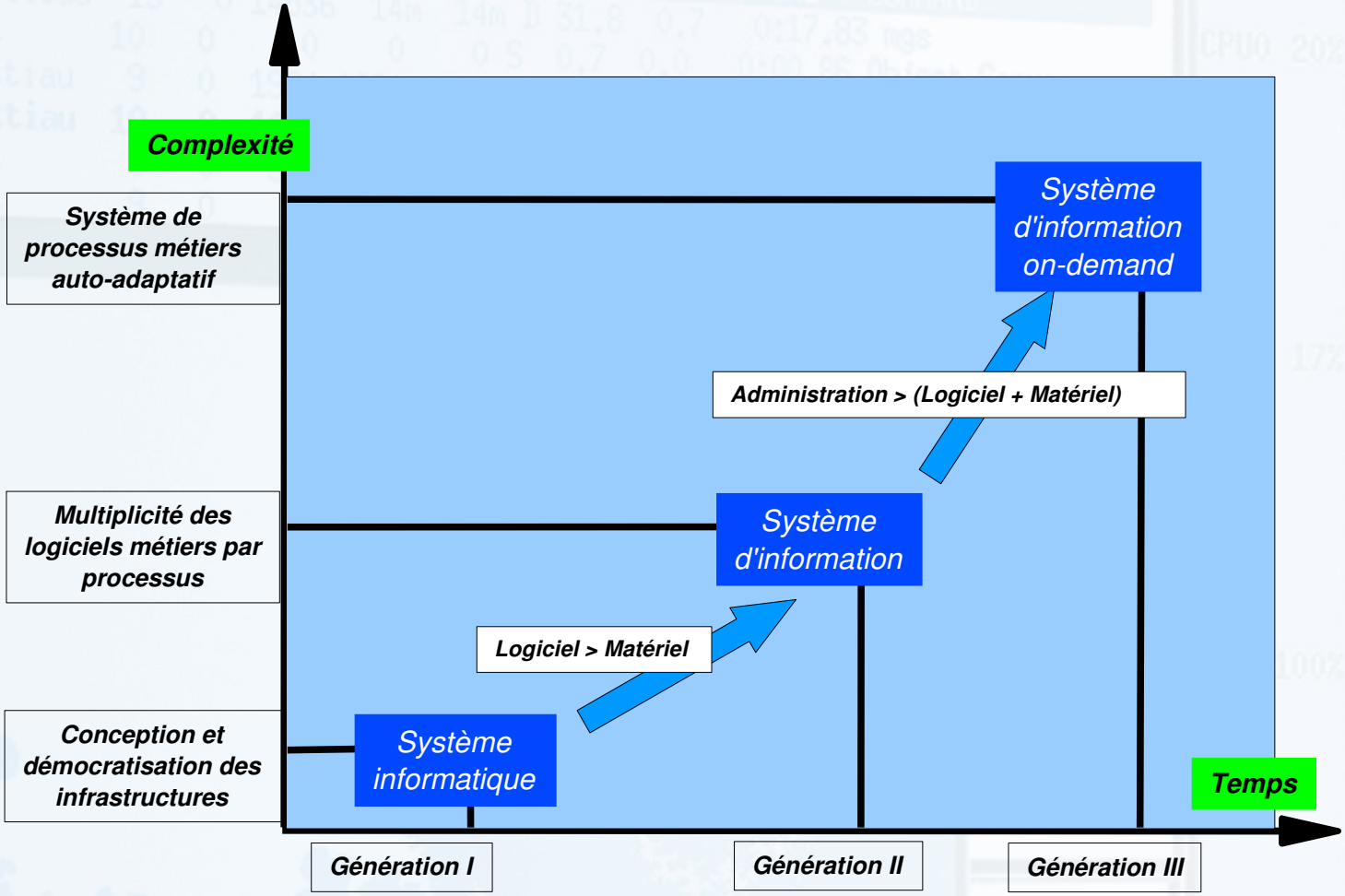
Kerlabs

Kerlabs – Qui sommes-nous ?

- **KERLABS est une *spin-off* de l'INRIA**
  - Industrialisation de la technologie Kerrighed
  - Kerrighed: système d'exploitation à image unique
  - 3 thèses de doctorats, 20 homme années de R&D
- **Création de la société en 2006**
  - SAS au capital de 200k€
  - Siège social à Rennes
  - Effectif : 7 personnes
- **Nos savoir-faire**
  - Système d'exploitation distribué
  - Architectures et programmation parallèles
  - Système de communication haut-débit

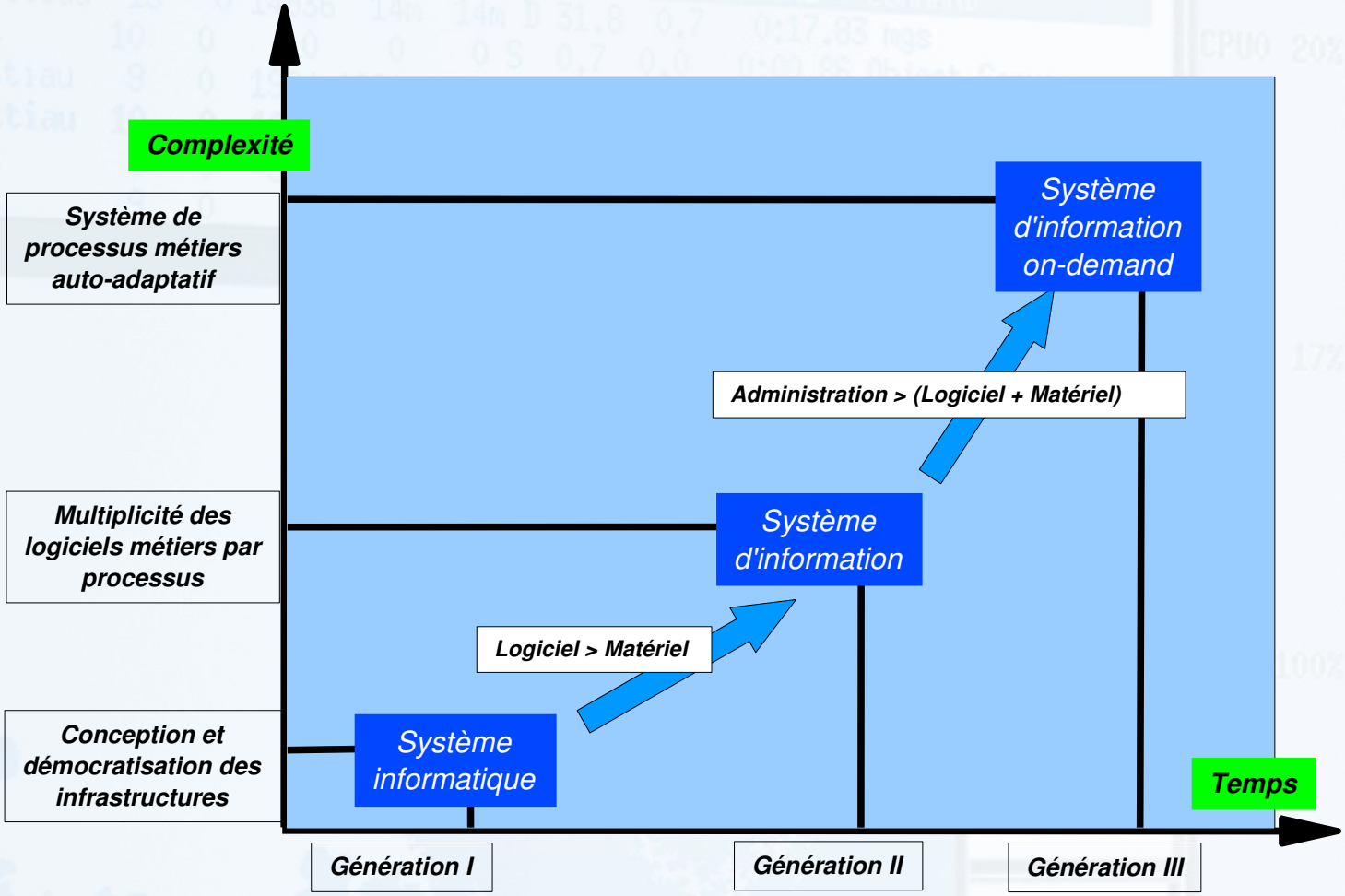
Kerlabs – Notre mission

La conception, la fourniture et le support
de plate-formes en puissance de calcul ou applicatif à la demande
en faveur des entreprises quelque soit leur taille, activité et localisation



Kerlabs – Notre stratégie

Proposer à nos clients un scénario de migration vers un Système d'Information On-Demand ainsi que les prestations, les outils et les services associés



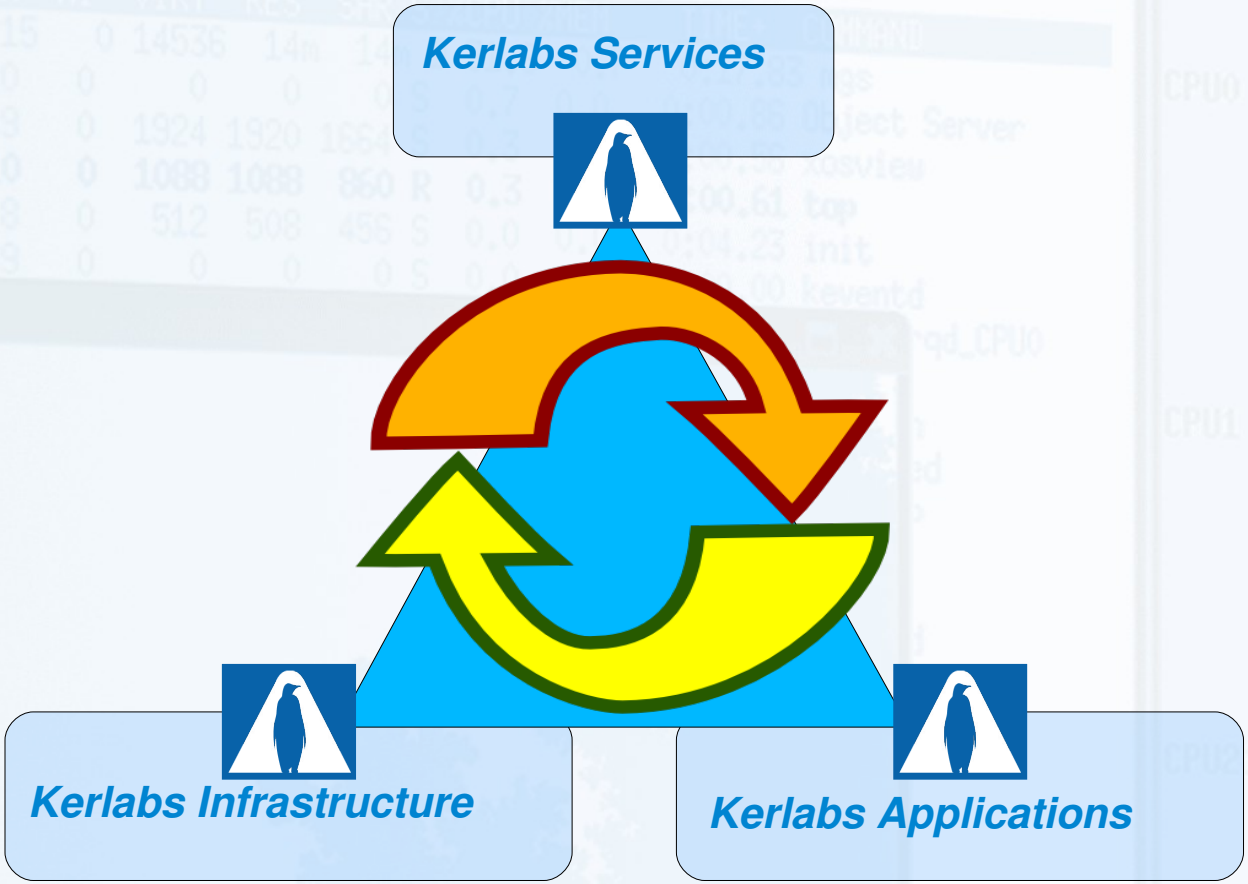
Kerlabs – Nos métiers

Virtualisation des ressources informatiques

- **Concevoir, développer, maintenir et supporter**
  - Plate-formes *On-Demand*
  - Machines *SMP* virtuelles
- **Services associés**
  - Conseil, intégraton & déploiement
  - Support & transfert de compétences
- **Architectures et codes parallèles**
  - Assistance aux équipes de développement
  - Laboratoire de tests et certifications

Kerlabs – Notre offre de services

Virtualisation des ressources informatiques



***Source** : Kerlabs, 2006*

Kerlabs – Boîte à outils...(1)

- **Montant des investissements**
  - Meilleur rapport prix/performance
  - Ajuster en permanence les budgets aux besoins
- **Dépenses d'exploitation**
  - Simplifier les opérations courantes d'exploitation
  - Diminuer les charges d'exploitation
- **Dépenses en énergie**
  - Optimiser les consommations en énergie
  - Diminuer les achats et conso en énergie des produits connexes

Kerlabs – Boîte à outils...(2)

- **Efficacité globale**
  - Absorber les pointes de charge de travail
  - S'adapter aux états de sous-emploi des ressources informatiques
  - En fonction des cycles de vie des processus business
- **Efficience globale**
  - Maintenir un haut niveau de continuité de service
  - Pour les plateformes *On-Demand* et Machines *SMP* virtuelles

Kerlabs – Infrastructure (1)

Plateformes On-Demand : ossature physique

- **Un socle d'hébergement**
  - Baies 19' avec éléments de commutation
  - Accueil des serveurs de stockage et des lames de calcul
- **Serveurs de stockage**
  - Dédiés à la gestion du système de fichiers et stockage
  - Équipements empilables, blade, NAS ou SAN pré-existants
- **Lames de calcul**
  - Traitement et communications avec les utilisateurs
  - Équipements 1U alloués aux machines *SMP* virtuels

→ **Indépendance vis-à-vis des constructeurs**

Kerlabs – Infrastructure (2)

Plateformes On-Demand : ossature logique

- **Machines SMP virtuelles à géométrie variable**
  - Usage des composantes matérielles de la plateforme *On-Demand*
  - Un serveur de stockage et entre 2 et 256 lames de calcul
- **Virtualisation des ressources informatiques**
  - Fusion des ressources *via* la techonologie Kerrighed
  - Fusion processeurs et mémoire
- **Équilibrage automatique de la charge de travail**
  - Équilibrage sur tous les processeurs actifs
  - Mécanisme d'équilibrage par défaut entièrement configurable

→ **Usage de machines SMP virtuelles**

Kerlabs – Infrastructure (3)

Plateformes On-Demand : outil de provisionning

- **Principe d'étanchéité entre machines SMP virtuelles**
  - KLS Manager: outil de provisionning
  - Création, ajout, retrait de machines SMP virtuelles
- **Principe d'allocation dynamique des ressources**
  - Allocation dynamique des lames de calcul aux machines SMP virtuelles
  - Avec lames de calcul en « spare » : en attente d'affectation

→ **Simplicité d'usage**

Kerlabs – Applications

- **Laboratoire de tests et certifications**
  - Mise à disposition de nos ressources de tests et de certification
  - Tests et certification par nos équipes
  - Maintien d'une bibliothèque d'application par nos équipes
- **Assistance aux équipes de développement**
  - Conseil & assistance en programmation parallèle
  - Transfert de compétences aux équipes de développement

→ Promouvoir la programmation parallèle

Kerlabs – Services

- Audit & conseil
- Intégration Système, Réseau & Logiciel
- Déploiement sur site
- Support & Assistance
- Transfert de compétences

Kerrighed
- Contexte -

Des processeurs

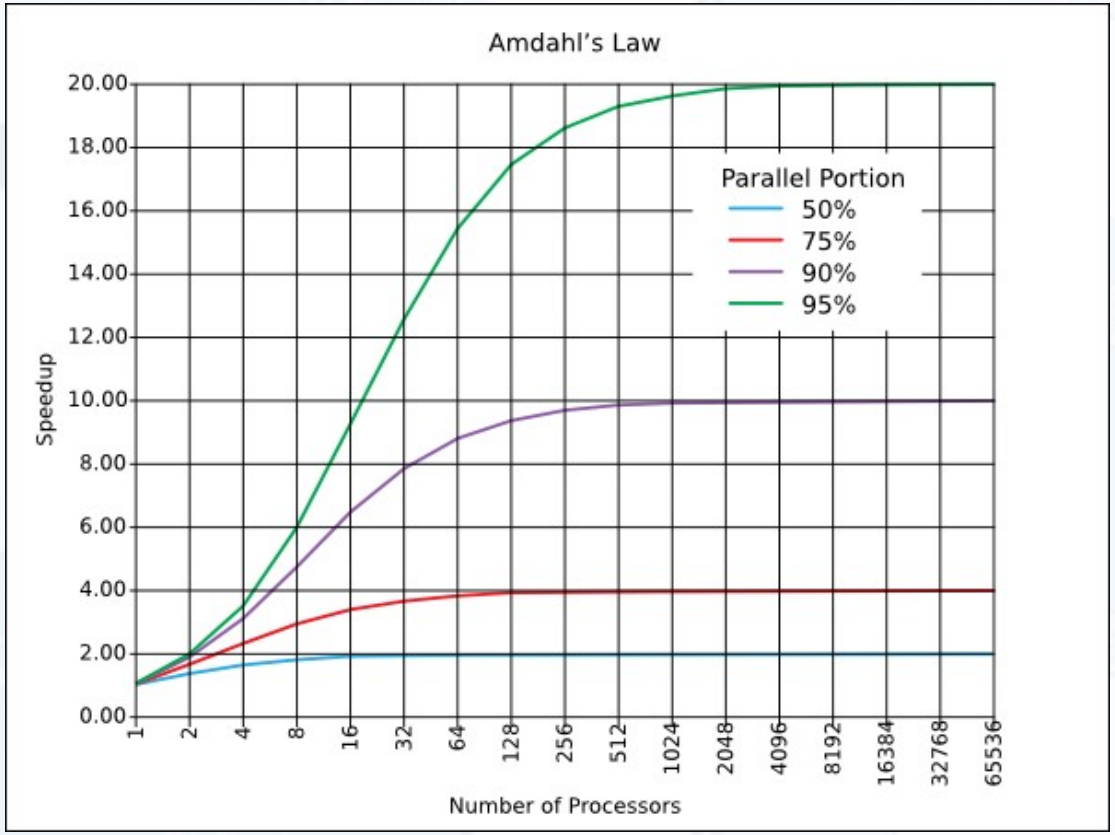
- **Nombre de cœurs/processeur en augmentation**
  - 2005/2006 : 2 cœurs
  - 2007 : 4 cœurs
  - 2008/2009 : 6 cœurs
  - 2010 : 12 cœurs
- **Fréquence des processeurs en régression**
  - Intel
    - Xeon « Irwindale » (2005) : 3.8GHz
    - Xeon « Nehalem » (2009) : 3.33GHz
  - Opteron
    - Opteron x2xx « Santa Ana/Santa Rosa » (2006) : 3.2GHz
    - Opteron x4xx « Istanbul » (2009) : 2.8GHz
- **Consommation (par cœur) en baisse**

Des applications (1)

- **Augmentation du parallélisme des applications**
- **Limitée par la loi de Amdahl appliquée à la parallélisation**
  - gain de performance pour une proportion d'activité parallélisable donnée

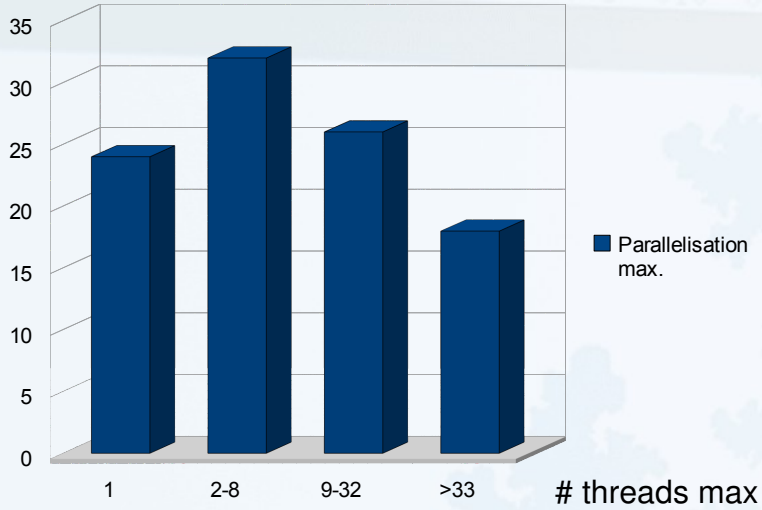
$$R = \frac{1}{(1-P) + \frac{P}{N}}$$

R : *gain de performance*
P : *portion de code parallélisable*
N : *nombre de processeurs*

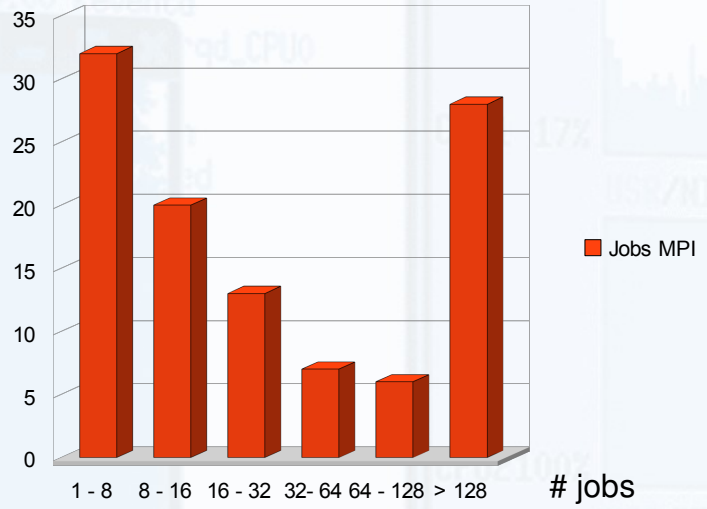


Des applications (2)

- **La plupart des applications se contentent de « constellations »**
  - « A constellation is a cluster of large SMP nodes scaled such that the number of processors per node is greater than the number of nodes » Tom Sterling



Parallélisation max. applications ISV
(78 applications)

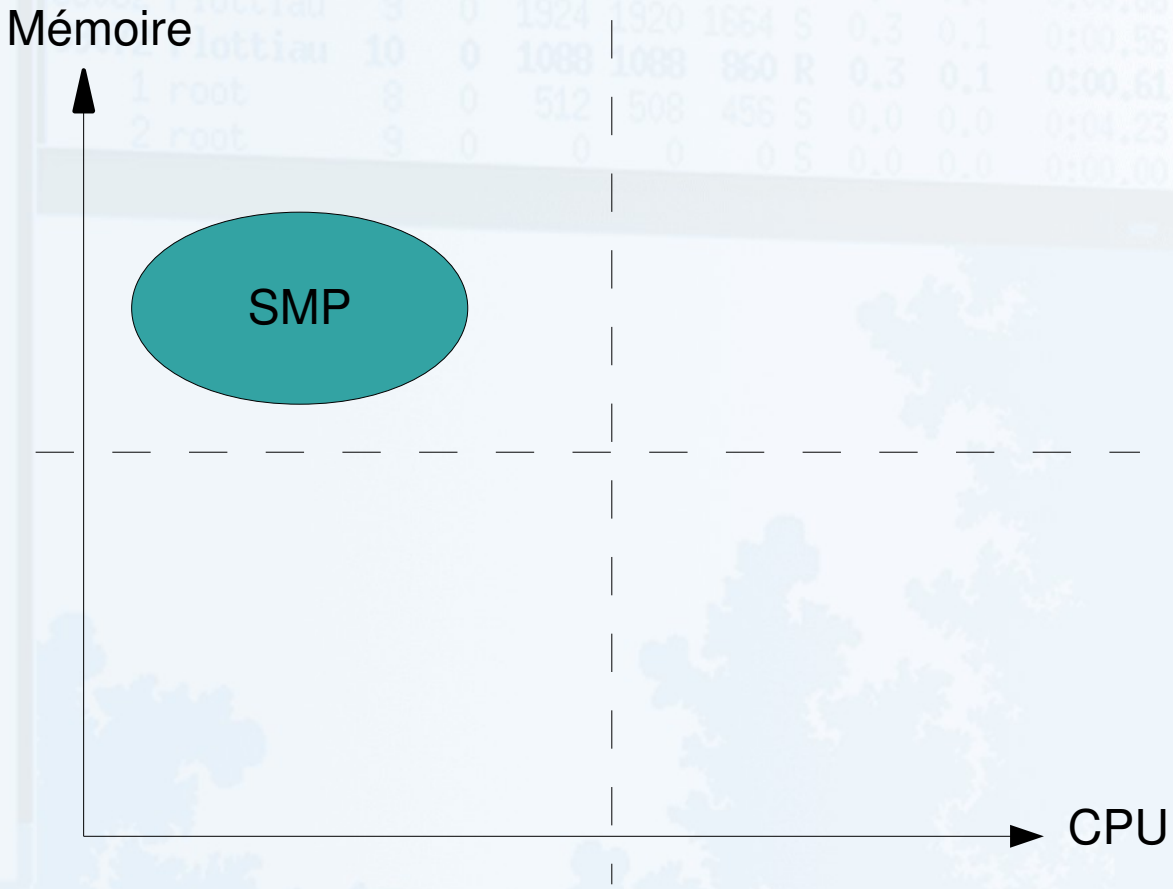


Parallélisation applications MPI
(enquête auprès de 106 utilisateurs)

Source: « The Personal Cluster: Coming to a desk near you » http://linux-mag.com/id/7243/

Architectures existentes (1)

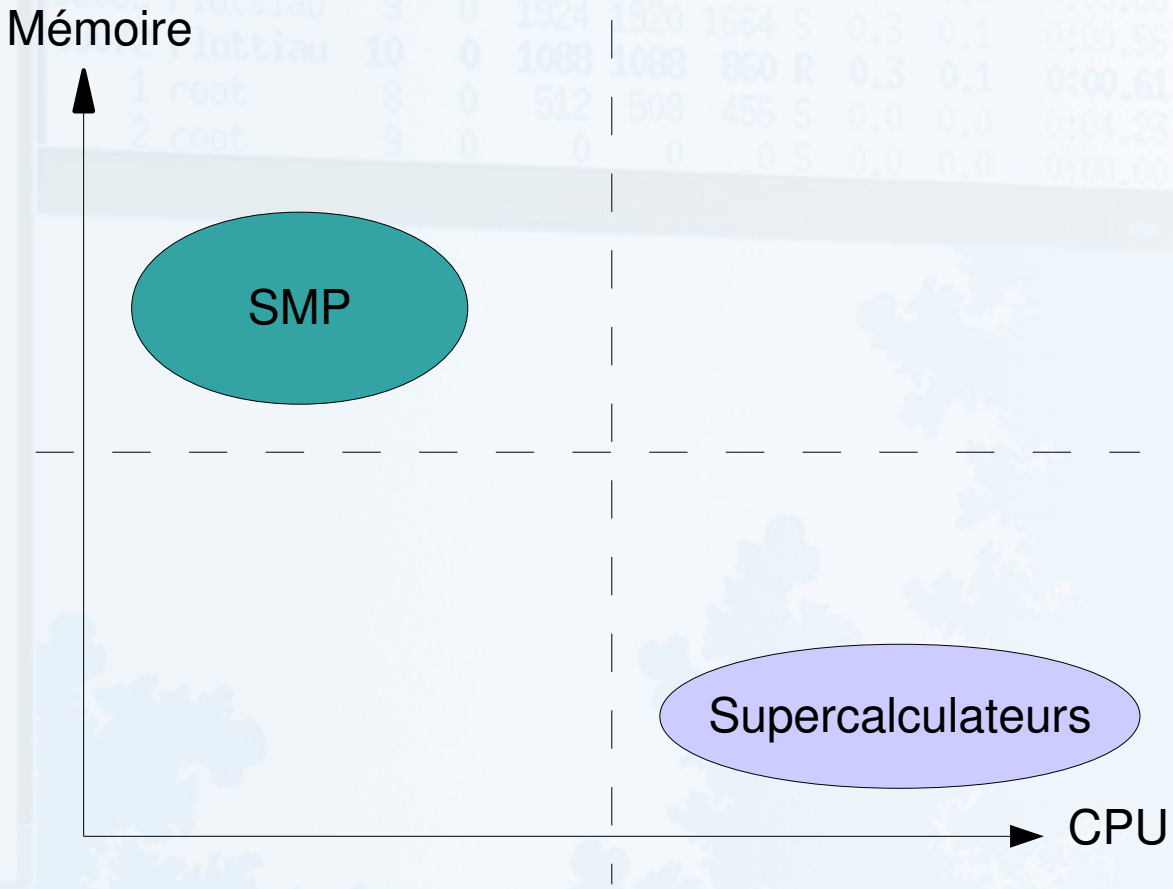
- **Réponse au besoin de mémoire ou besoin de CPU**



- **SMP**
  - Mémoire disponible pour chaque processus
  - Ressource CPU limitées (8, 32 voies)
  - Prix exponentiel

Architectures existentes (2)

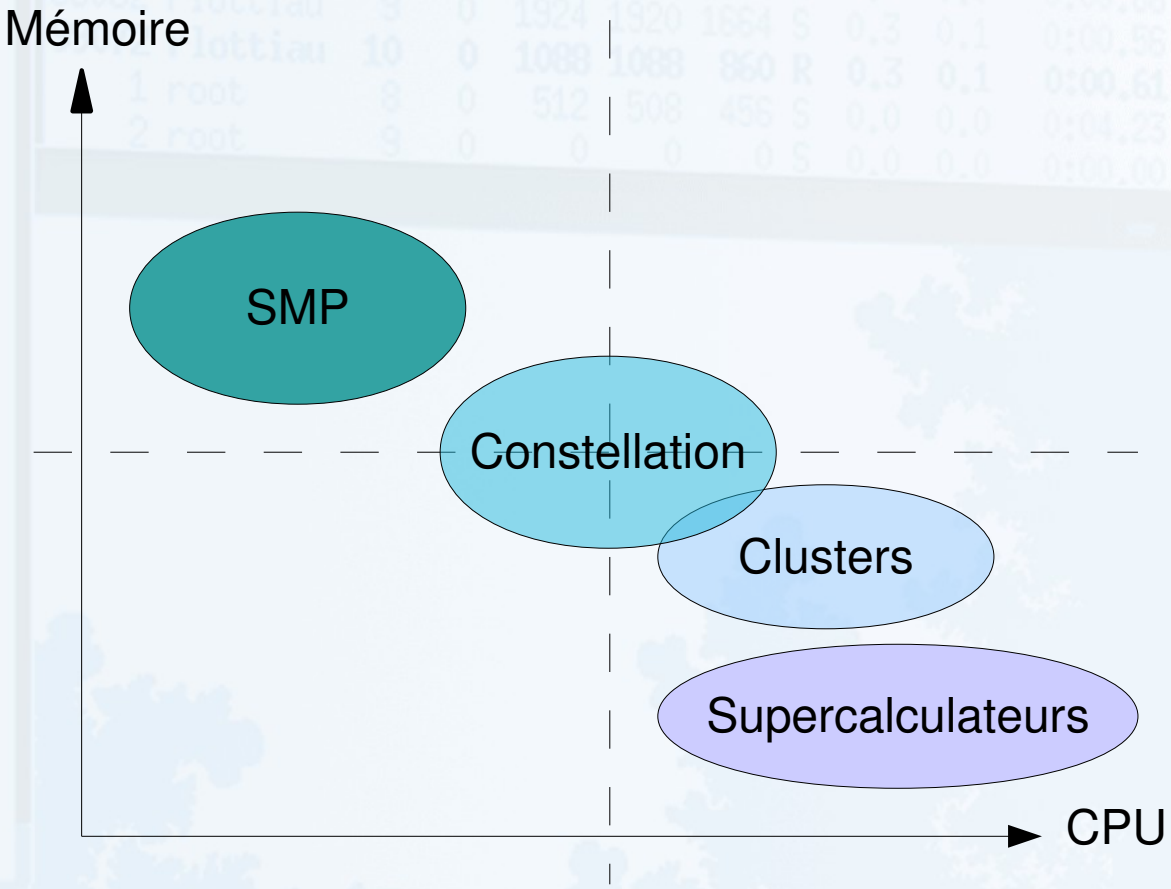
- **Réponse au besoin de mémoire ou besoin de CPU**



- **Supercalculateurs**
  - Grand nombre de CPUS
    +100 000
  - Mémoire par processus limitée

Architectures existentes (3)

- **Réponse au besoin de mémoire ou besoin de CPU**



- **Clusters (Beowulf)**
  - Grand nombre de CPUS +100 000
  - Mémoire par processus limitée
  - Prix « modique » : usage de composants standards
- **Constellations**
  - Petits clusters
  - Processeurs multi-cœurs

Kerrighed
- Principes des SSI -

Problématiques clusters

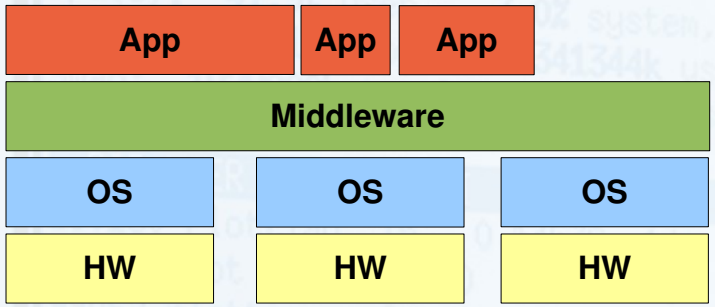
- Beaucoup de nœuds identiques (à peu près)
- Complexité d'administration
- Complexité d'utilisation

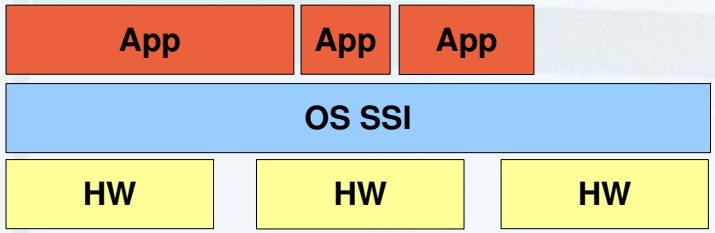
Idées

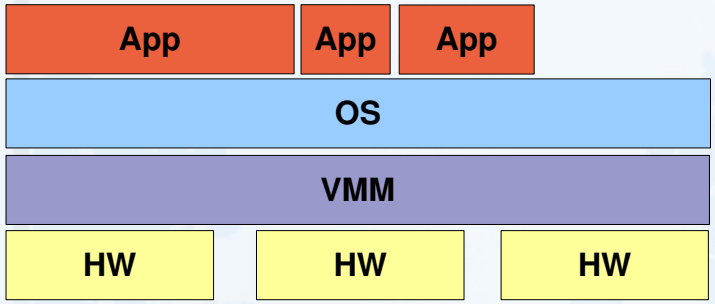
- **Abstraction (virtualisation) des ressources**
  - CPU : exécution des applications sur tous les nœuds du cluster
  - Mémoire : utilisation de plus de mémoire que dispo sur un nœud, si nécessaire
  - Disque : vue d'un seul espace de fichiers
  - Autres...
- **OS pour cluster ?**
- **Éventuellement, utilisation d'application non spécifiques aux clusters**

- **→ C'est le Single System Image**

Mises en œuvres (1)

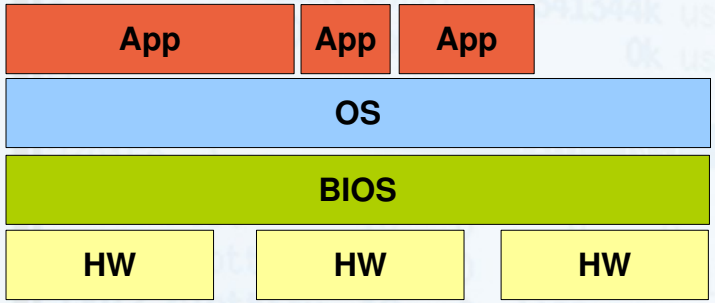




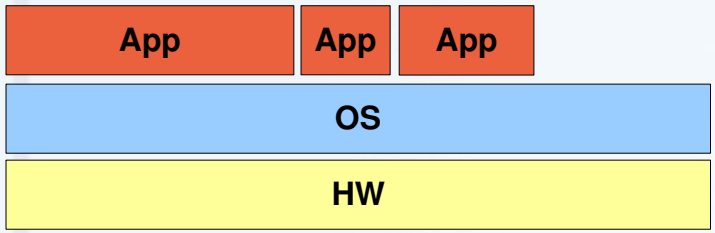


- **Middleware**
  - MPI, CORBA, Java RMI, *etc.*
  - Optimisation et fonctionnalitées limitées
  - Programmation spécifique
- **OS**
  - Kerrighed, OpenMosix, OpenSSI
  - Compatibilité des applications, optimisation
  - Complexe à développer et maintenir
- **VMM**
  - ?
  - OS non modifié
  - Optimisation très limitée, scalabilité

Mises en œuvres (2)



- **BIOS**
  - ScaleMP (commercial)
  - (Relativement) simple
  - Optimisation très limitée, scalabilité (*cf* VMM)
  - Dépendance matériel forte



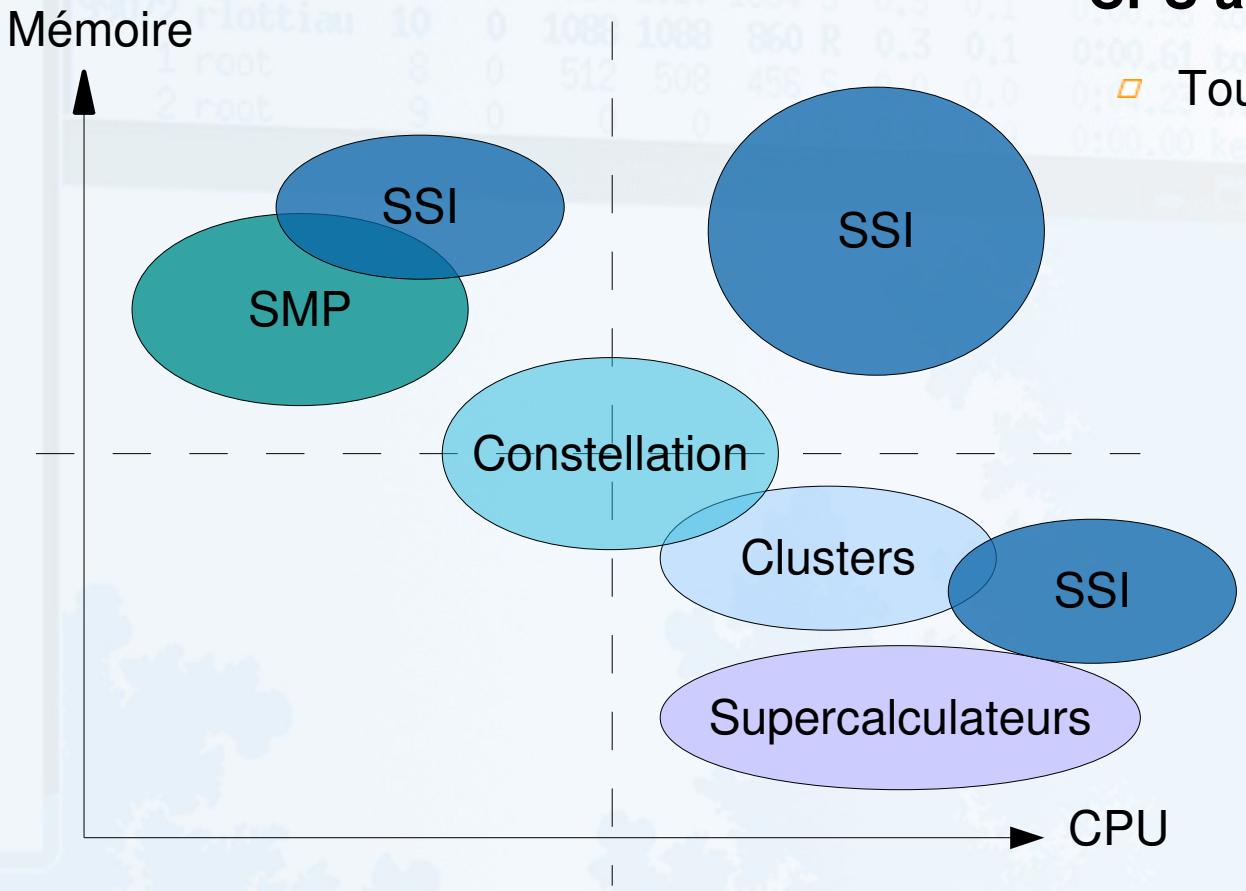
- **Matériel**
  - Voir SMP

SSI au niveau noyau

- **Nécessite**
  - Gestion globale de la mémoire
  - Gestion globale des processus
  - Gestion globale des communications réseau
  - Gestion globale du stockage

Le SSI comme architecture

- **Mémoire aggrégée**
  - Entièrement disponible pour les processus
- **CPU aggrégés**
  - Tous disponibles pour les processus



Kerrighed
- Présentation -

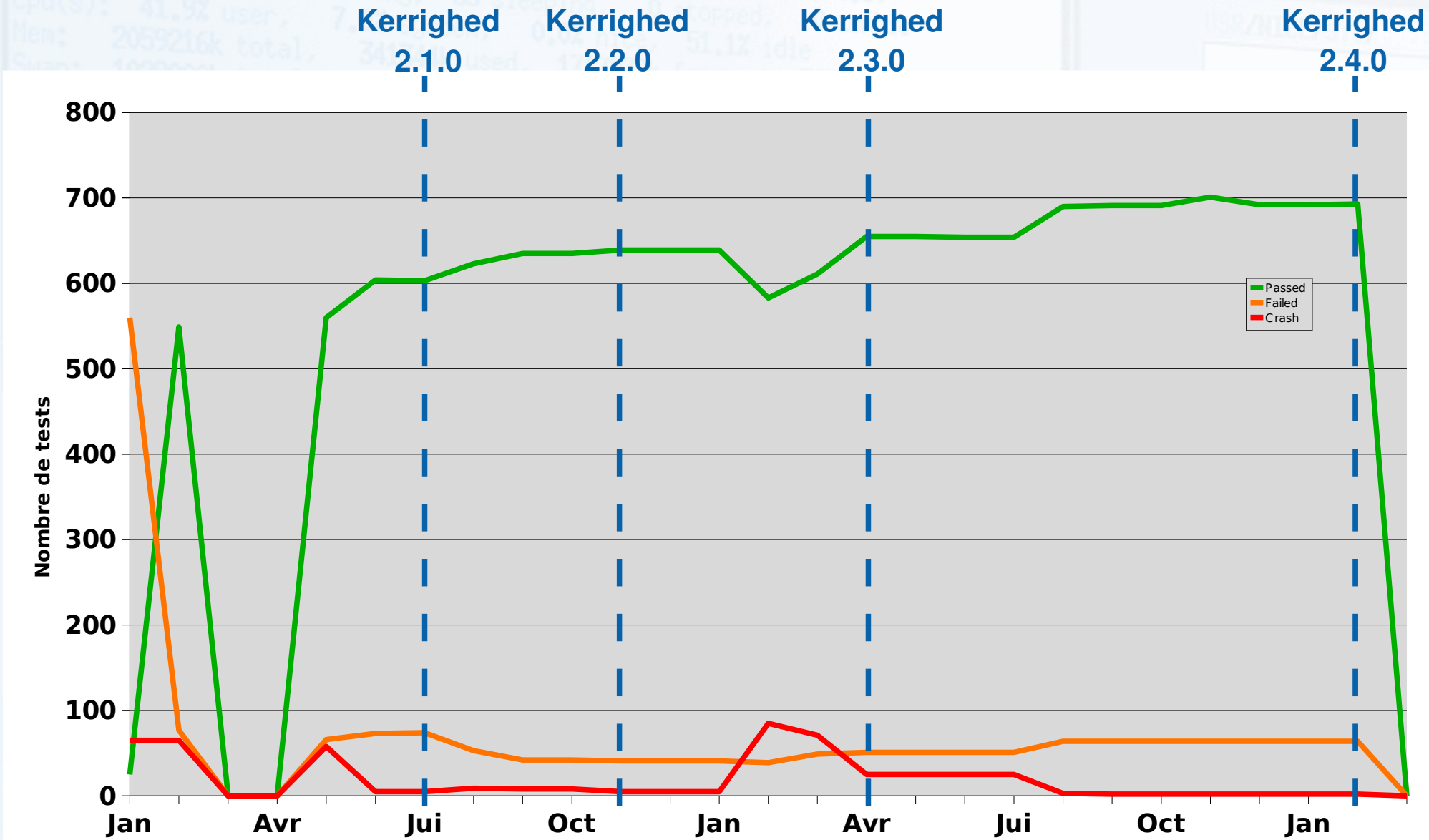
Kerrighed : le projet (1)

- **Projet initié par l'IRISA en 1999 dans l'équipe-projet PARIS**
  - Sous la direction de Christine Morin
  - Collaboration entre INRIA, EDF et Université Rennes 1
  - 3 thèses de doctorat
  - Nombreux ingénieurs et stagiaires
  - 30 années * hommes de R&D
- **2006 : un projet communautaire**
  - Création de Kerlabs, *spin-off* de l'INRIA, en 2005
  - Partenariat avec le projet Européen XtreemOS (OS pour grille)
  - Nombreux contributeurs
  - Site web, maillists, bug tracker, *etc.*

Kerrighed : le projet (2)

- **De la recherche à la production**
- **Dépôt des sources public**
- **SVN (jusqu'à 2.4.x) : `svn://scm.gforge.inria.fr/svn/kerrighed/trunk`**
  - Git (depuis 2.5) : `git://git-externe.kerlabs.com`
    - Mirroir : `http://mirrors.git.kernel.org`
- **Facilités de déploiement**
  - Procédure d'installation standard (`autotools`)
  - Paquets Debian (non à jour), Mandriva
  - OSCAR (non maintenu)
  - LiveCD
- **Batterie de 700 tests environ**
  - LTP + tests spécifiques Kerlabs

Kerrighed : le projet (3)



Kerrighed : en bref

- **Système d'exploitation à Image Unique**
- **Extension d'un noyau connu : Linux**
  - Modifications limitées
  - Interfaces connues
- **Ordonnancement globale des processus, configurable**
- **Un ensemble de fonctionnalités (dés)activable pour chaque processus**
  - Migration
  - Démarrage à distance (*fork*)
  - Mémoire globale
  - Vue globale /proc
  - Point d'arrêt/reprises
- **Transparent et configurable**

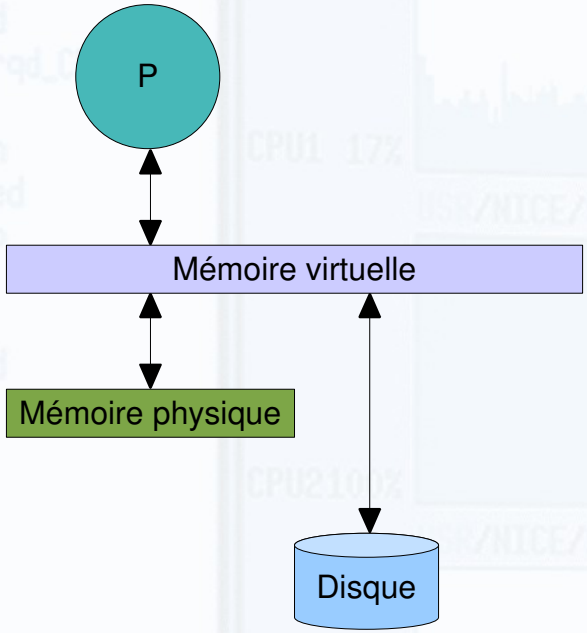
Mécanismes connus

- **Aucune modification des applications existantes**
- **Mise à profit des développements de Linux**
- **Environnement utilisateur familier**
- **Illustration à travers 2 examples :**
  - Gestion de l'agrégation mémoire
  - Intégration des conteneurs Linux

Agrégation mémoire (1)

- **1 processus utilise plus de mémoire que disponible sur un nœud**
- **Système classique : hiérarchie à 2 niveaux**
  - Mémoire centrale
  - Disque dur : débit 100 – 1000 fois < mémoire, latence 100 000 fois > mémoire
- **Le *swap* tue les performances !**

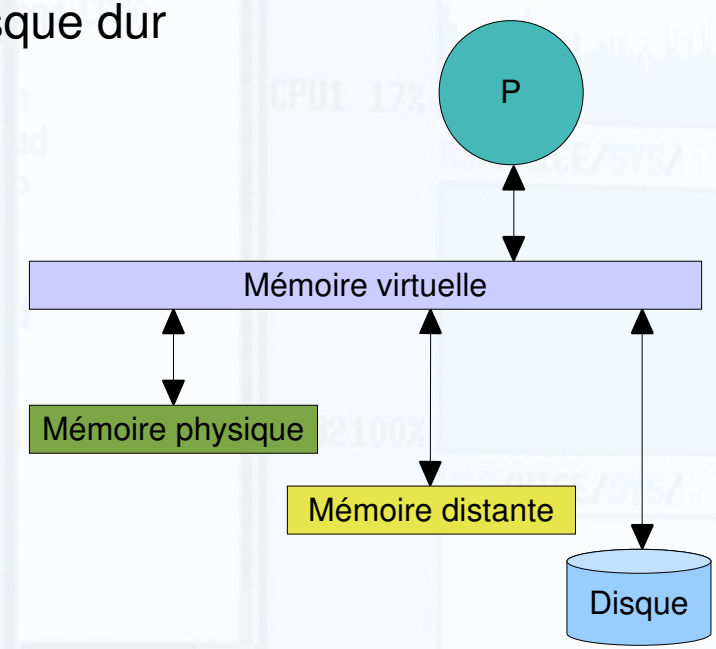
	Capacité	Débit	Latence
Mémoire centrale	2 Go	~5 Go/s	~ 50 ns
Disque dur	100Go	~5 – 50 Mo/s	~ 5 ms



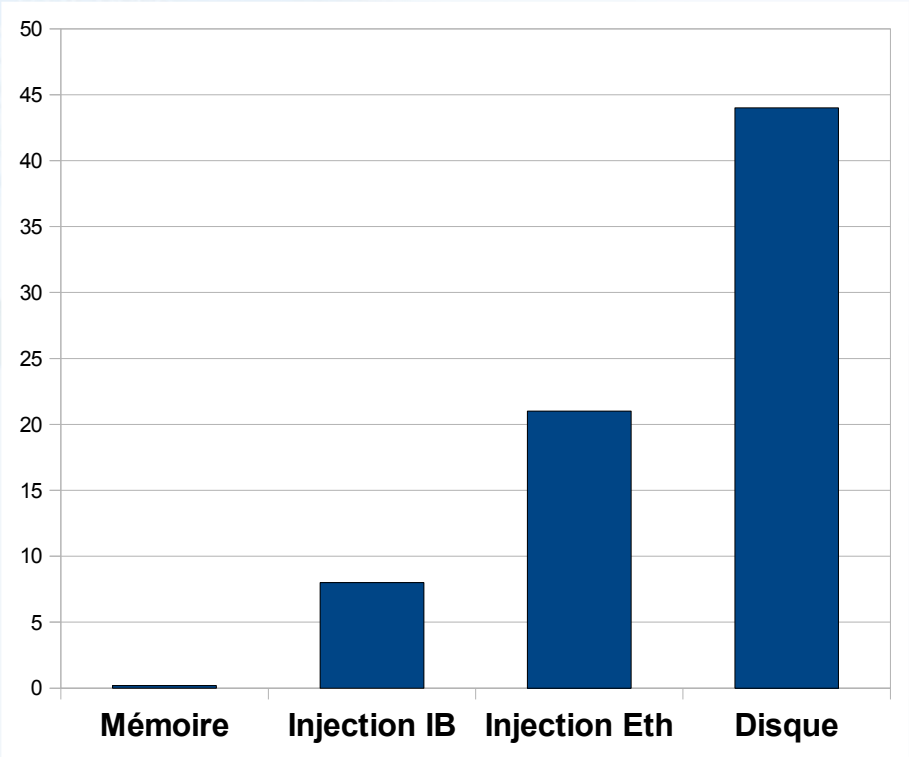
Agrégation mémoire (2)

- **Ajout d'un niveau intermédiaire :**
  - Mémoire distante (*via* réseau)
- **Débits**
  - 2 – 40 fois < mémoire, 2,5 – 500 fois > disque dur
- **Latence**
  - 33 – 500 fois > mémoire, 200 – 2500 fois < disque dur

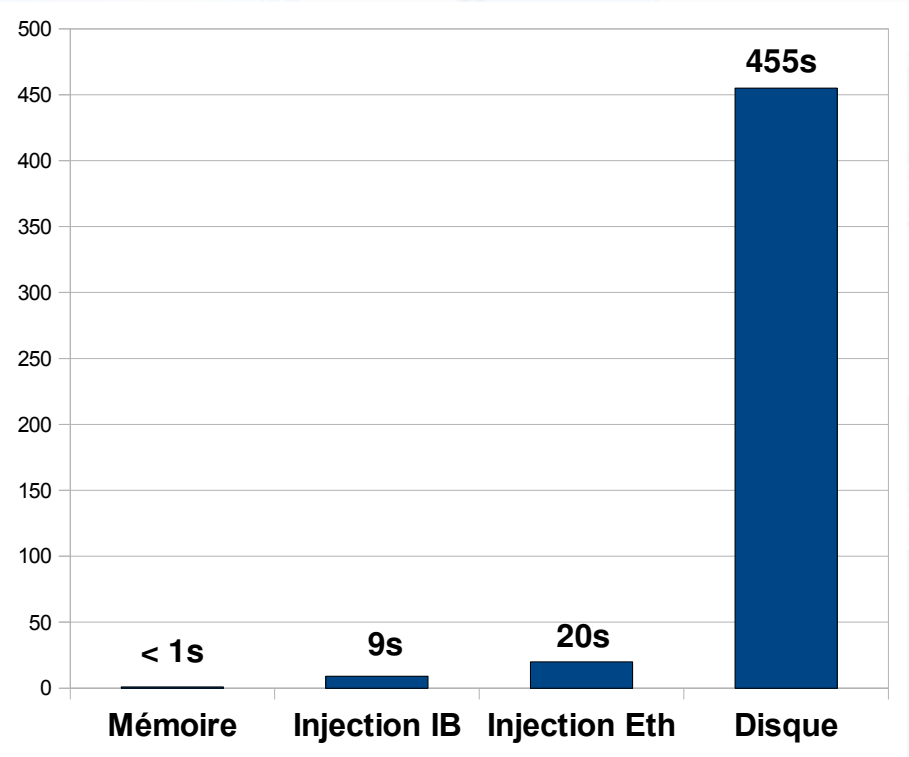
	Capacité	Débit	Latence
Mémoire centrale	2 Go	~5 Go/s	~ 50 ns
Mém. Distante (GbE)	16 Go	~120 Mo/s	~30 µs
Mém. Distante (IB 4x DDR)	16 Go	~2.5 Go/s	~2 µs
Disque dur	80 Go	~5 – 50 Mo/s	~ 5 ms



Agrégation mémoire : performances (1)

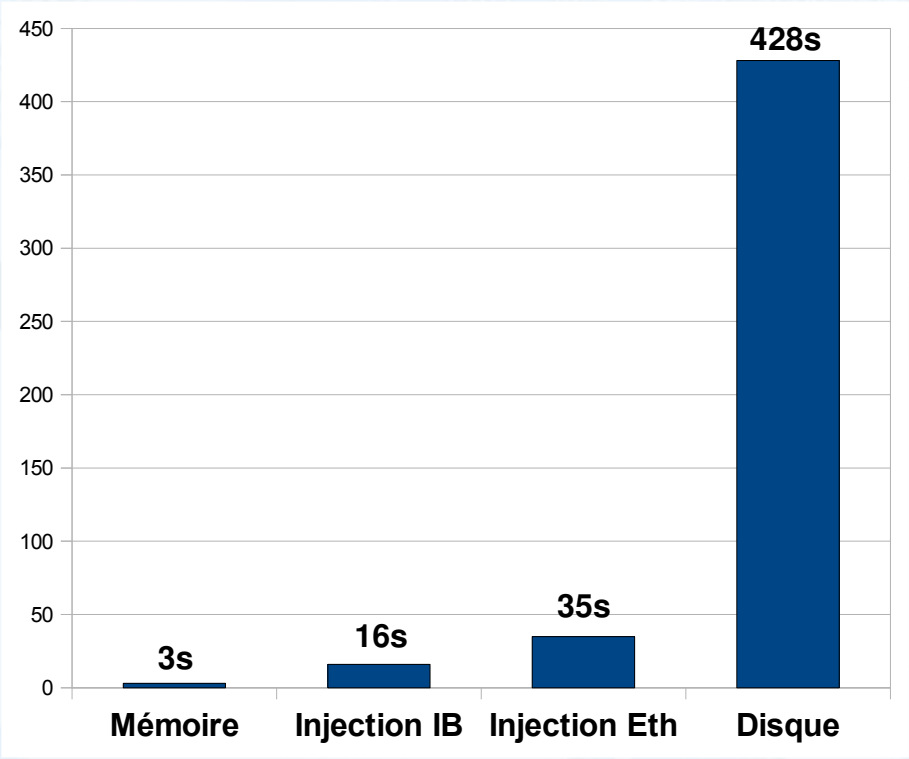


Accès mémoire séquentiel

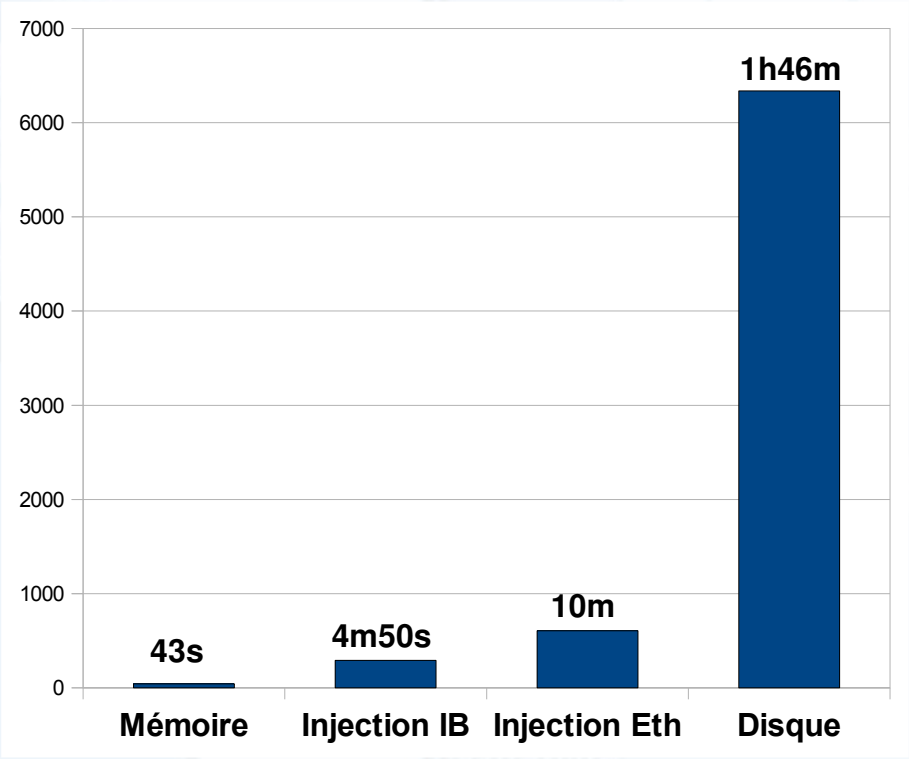


Accès mémoire aléatoire

Agrégation mémoire : performances (2)



Transposée de matrice



Code de mécanique

Agrégation mémoire : optimisations

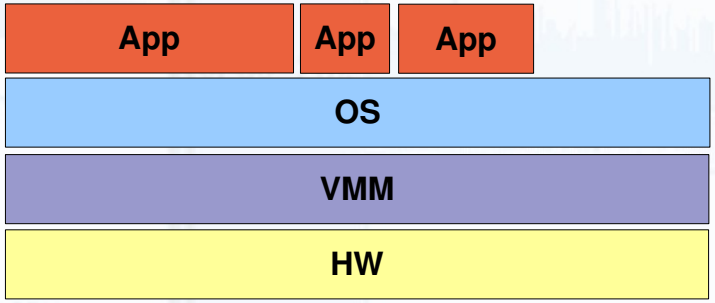
- **Nombreuses optimisations en cours de développement**
  - Pré-chargement
  - Agrégation de pages
  - Détection de motifs d'accès
  - Communication sans recopie
  - Utilisation du RDMA
- **Résultats attendus**
  - GbE: x2
  - Infiniband : x5

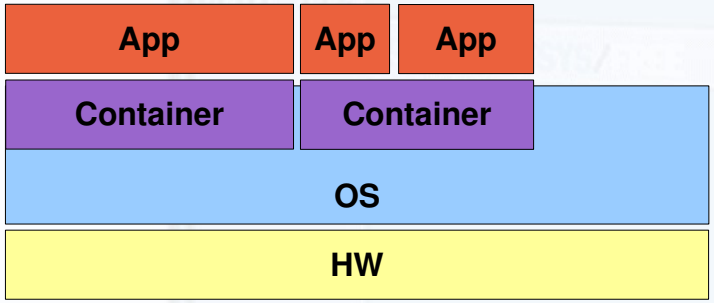
Conteneurs Linux : problématique

- **Problématiques**
  - Isolation des ressources de la grappe :
    - QoS
  - Isolation des ressources des nœuds de la grappe :
    - « *cluster of workstations* »
  - Vie des objets (*pid*, *uid*, etc.) en dehors de la grappe : avant le démarrage, après l'arrêt
    - démarrage/arrêt des machines
    - Point de reprises/redémarrage
- **Virtualisation !**

Conteneurs Linux : techniques existantes

- **Techniques existantes**
  - Virtualisation (plus ou moins) complète :
    - Xen, VMWare, *etc.*
    - Performances médiocres
  - Virtualisation légère : virtualiser les seules ressources nécessaires
    - Vserver, Virtuozzo, *chroot*
    - Linux containers

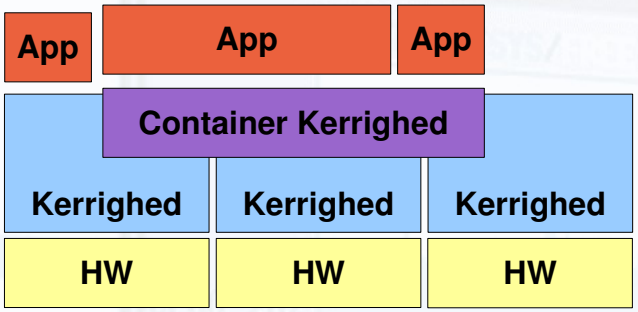




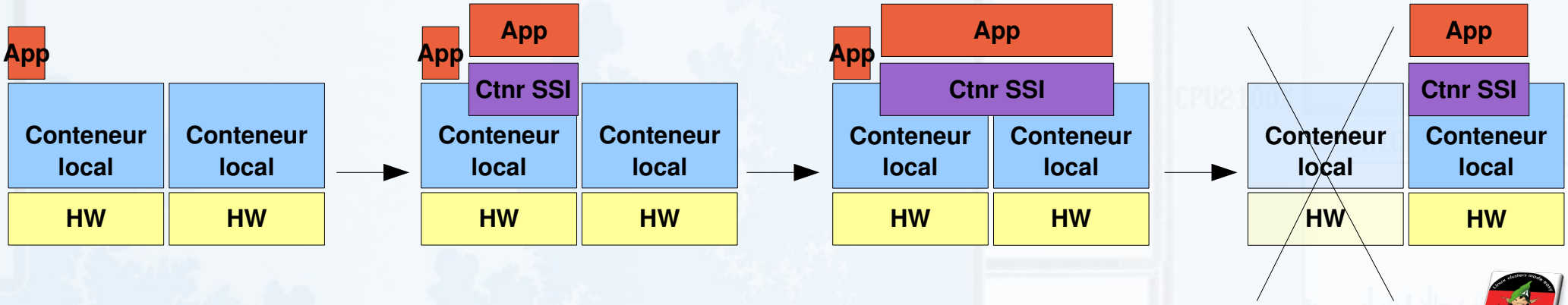
Conteneurs Linux : présentation

- **Conteneurs Linux**
  - Où ?
    - À l'intérieur de l'OS : optimisations possibles
  - Quelles ressources ?
    - PID, hostname, mounts, IPC, user, network
  - Comment ?
    - Hériarchiques
    - QoS
- **Utilisation**
  - Isolation : sécurité, QoS
  - Migration de container : serveurs virtuels
- **Intégration dans le noyau Linux progressive**
  - Depuis 2.6.19

Conteneurs Linux + Kerrighed (1)



- **Kerrighed ajoute un élément**
  - Ressources réparties sur différents nœuds
- **Synopsis du SSI**
  - Chaque nœud démarre le conteneur global (local)
  - Un nœud crée un conteneur distribuable
  - Les autres nœud peuvent rejoindre le conteneur distribuable ou créer un autre conteneur distribuable



Conteneurs Linux + Kerrighed (2)

- **PIDs**
  - Séparation processus migrables/non migrables (init...)
  - Redémarrage d'applications
- **Hostname**
  - un nom par SMP virtuel
- **Montages**
  - résoud initramfs complexes (NFSROOT, autres...)
- **SYSV IPC/UID**
  - Redémarrage d'applications
- **UID**
  - Redémarrage d'applications
- **Réseau**
  - Pas de risque de configuration réseau différente d'un nœud à l'autre (interfaces)

Conteneurs Linux : conclusion

- **Kerrighed profite des mécanismes introduits dans le noyau**
- **État de l'intégration**
  - Fonctionnel depuis Septembre 2009 (version de développement)
  - Permet aujourd'hui
    - Démarrage de conteneurs distribués dans conteneur global
  - Ajout de fonctionnalités supplémentaires au fur et à mesure
    - Court terme : ajout/retrait de nœuds à chaud (SC'09)
    - Conteneurs à l'intérieur du conteneurs distribué

Modifications limitées

- **Kerrighed 2.4.1 :**
  - Linux 2.6.20
  - Patch Kerrighed (sur le noyau) :

```
$ bzcat patches/999_kerrighed.patch.bz2 | diffstat | tail -1
 307 files changed, 32683 insertions(+), 21176 deletions(-)
```

  - Un module noyau
  - Quelques outils (`krgadm`, `migrate`, `restart`, `checkpoint`, *etc.*)

L'ordonnanceur : objectifs

- **Objectif : équilibrer la charge processeur d'une grappe**
  - Déplace des processus d'une machine vers une autre
  - Déplacement automatique
- **Différentes politiques possibles**
  - Injection de tâche
  - Vol de tâche
  - …
- **Différents critères**
  - Charge processeur
  - Charge mémoire
  - Température processeurs
  - ...

L'ordonnanceur : problèmes

- **En général, ordonnanceur codé en dur dans le système**
  - Un algorithme générique
  - Un critère d'ordonnancement : en général « charge processeur »
- **Limitations**
  - Erreur d'ordonnancement dûes à l'algorithme
  - Critères non pertinents
  - Volonté de contrôle de l'utilisateur
- **Pour changer la politique**
  - Au mieux : modifier le code dans le noyau
  - Au pire : impossible !

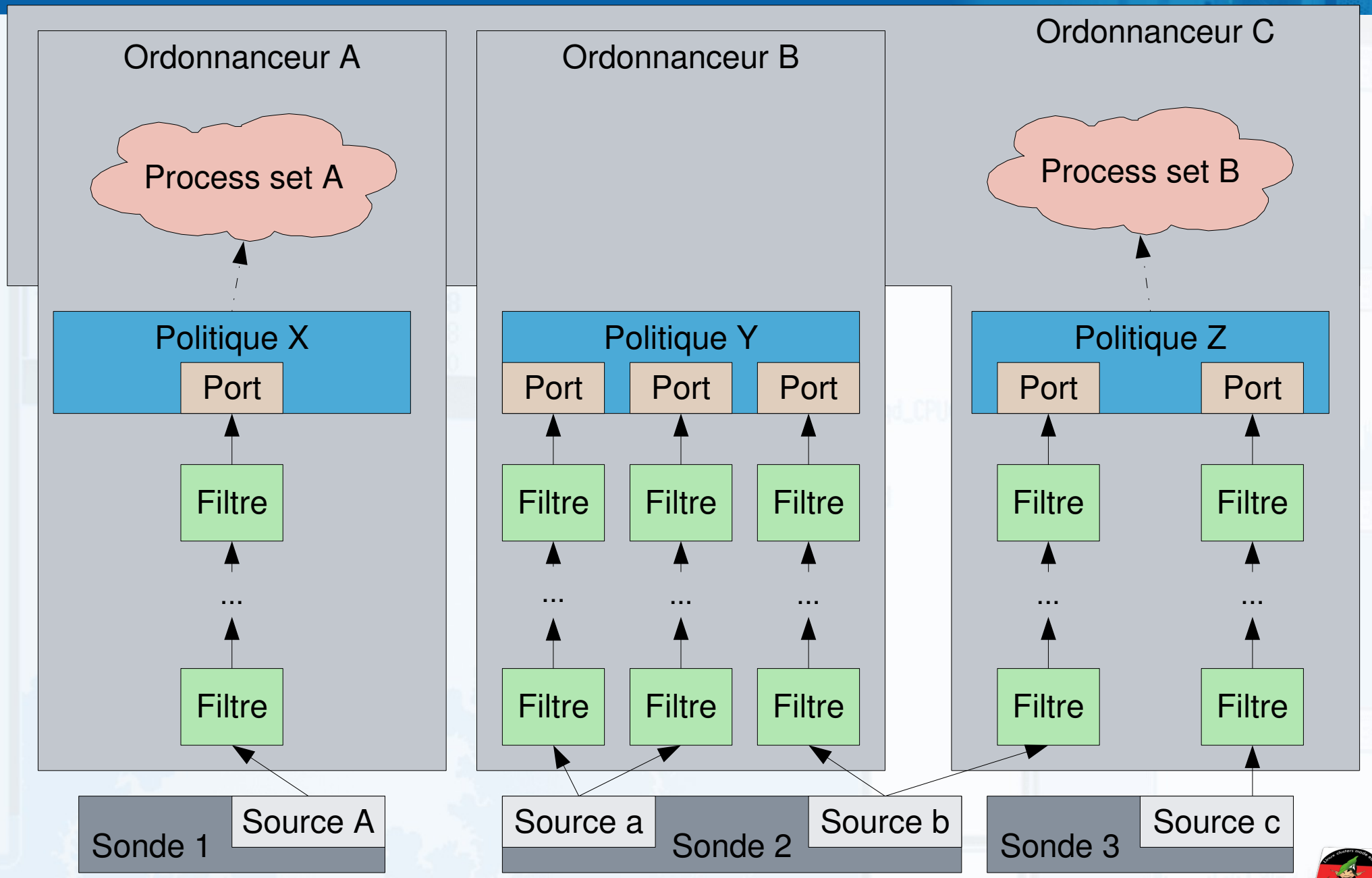
L'ordonnanceur : les capacités

- **Les capacités : premier niveau de contrôle**
- **Contrôle le comportement de l'ordonnanceur**
  - Active/désactive certaines fonctionnalités de l'ordonnanceur
  - Au niveau processus
  - Héritable
- **2 capacités**
  - DISTANT_FORK
    - Active/désactive le placement de tâches
    - Utile pour les tâches de courte durée
  - CAN_MIGRATE
    - Active/désactive la migration de tâches
    - Utile pour les tâche de longue durée

L'ordonnanceur configurable

- **Objectif : modifier à chaud l'ordonnancement de processus**
  - Adapter la politique aux besoins de l'application
  - Créer de nouvelles politiques
    - Affinité disque
    - Affinité réseau (*ex.*: tâches MPI)
    - Équilibrage charge disque
    - Limitation des « points chauds »
    - *Etc.*
  - Contrôler les tâches gérées par les ordonnanceurs
  - Hiérarchiser les ordonnanceurs

L'ordonnanceur configurable : architecture



L'ordonnanceur configurable : fonctionnalités

- **Interface utilisateur**
  - Configuration *via* Config-FS
    - `mkdir <module>` pour charger un module
    - `mkdir <PID>` pour prendre un charge un ensemble de processus
    - `ln -s` pour lier des éléments
  - Configuration globale au cluster
- **Niveau noyau**
  - Modules : sondes, filtres, politiques
  - Interfaces à respecter pour la connexion des éléments

L'ordonnanceur configurable : exemples

- **Ordonnanceur de jetons**
  - Une clé USB dispose de jetons sur chaque nœuds
  - Consommation de jetons très hétérogènes
  - Objectif :
    - Équilibrer la consommation de jetons
    - Prévenir l'admin lorsqu'une clé passe sous un seuil de jetons
- **Priorité de campagne de mails**
  - Envoi de campagne de mails
  - Chaque campagne dispose d'une priorité
  - Objectif :
    - Écouler au plus vite les campagnes les plus prioritaires

Les capacités

- **Même mécanisme que les capacités Linux**
  - Voir `man capabilities(7)`
- **Active/désactive certaines fonctionnalités de Kerrighed**
  - Les capacités sont par processus
  - Éventuellement héritables
  - Interface utilisateur: `krgcapset`
  - 4 ensemble de capacités par processus:
    - Effective, Permitted, Inheritable Permitted, Inheritable Effective
  - Voir pages de manuel : `krgcapset(1)`, `kerrighed_capabilities(7)`

Migration

- **Objectif : équilibrage de charge**
- **Migration des processus d'un nœud à l'autre**
  - Qui ?
    - Capacité CAN_MIGRATE
  - Quand ?
    - Automatique : ordonnancement global
    - Manuel : `migrate <pid> <nodeid>`
  - Comment ?
    - Efficace : contexte déplacé au besoin
    - Flux
      - Utilisation de système de fichier distribué
      - Pour les fichiers locaux (devices, etc.) : redirection (démon FAF)

Démarrage distant

- **Objectif : équilibrage de charge**
- **Tâches courtes : coût migration prohibitif**
- **Démarrer la tâche sur un nœud distant**
  - Qui ?
    - Capacité DISTANT_FORK
  - Quand ?
    - Au démarrage de l'application, si un nœud distant est plus « propice »
  - Comment ?
    - *cf.* migration

Mémoire globale

- **Objectifs : besoins en mémoire supérieure à un nœud**
- **« Swap » réseau**
  - Qui ?
    - Capacité USE_REMOTE_MEMORY
  - Limitations
    - Utilisation d'un réseau rapide, GbE un peu limite

Vue globale /proc

- **Objectif : rendre Kerrighed le plus transparent possible**
- **/proc est utilisé par de nombreuses applications**
- **Avec Kerrighed, le système de fichiers /proc est globale à la grappe:**
  - Ajout d'un répertoire /proc/nodes
    - Informations propres à Kerrighed : état des nœuds, nœud courant, etc.
  - Possibilité de voir un /proc local
    - Capacité SEE_LOCAL_PROC_STAT

Point d'arrêts/reprise : problématique

- **Problématique**
  - MTBF diminue avec l'ajout de machines
  - Avec des applications longues, risque élevé de panne pendant la vie de l'app.
- **Objectif : haute disponibilité**
- **Mécanismes**
  - Sauvegarder l'état de l'application régulièrement
  - Si arrêt inopiné (*crash*), redémarrage de l'application au dernier point de reprise valide.

Point d'arrêts/reprise : processus

- **Interface utilisateur : 2 commandes utilisateur**
  - checkpoint <pid>

```
$ checkpoint 4561236
Identifier: 4561235
Version: 1
Description: No description
Date: Tue Mar 31 14:34:00 2009
```

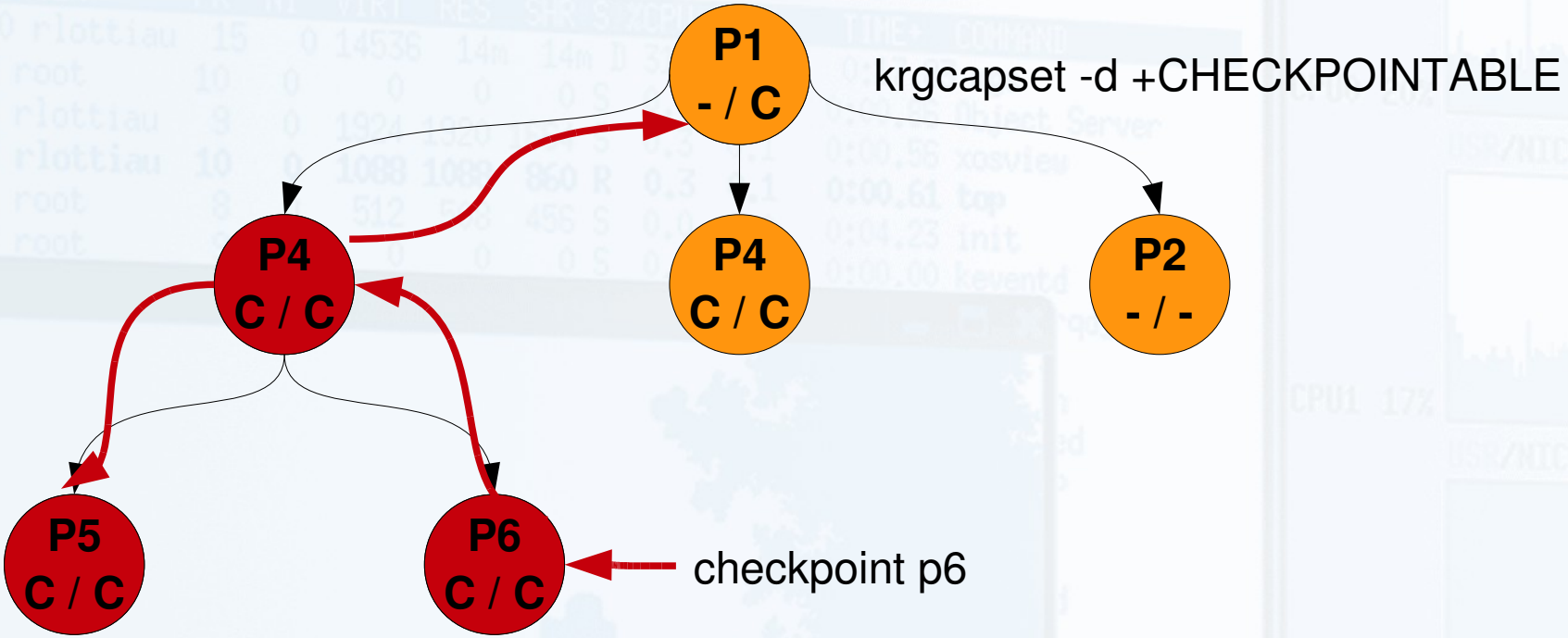
  - restart <application> <version>

```
$ restart 4561235 1
Restarting application 4561235 (v1)...
```


Point d'arrêts/reprise : application (1)

- **Définition de l'application**
  - Ensemble de processus
  - Des liens entre processus : relation de filiation
- **Frontière de point de reprise**
  - Par défaut un processus ne peut être « checkpointé »
  - Capacité CHECKPOINTABLE
  - Permet de définir la frontière de checkpoint

Frontière de checkpoint



Point d'arrêts/reprise : application (2)

```
$ checkpoint 4561236
Freezing application in which process 4561236 is involved...
Checkpointing application in which process 4561236 is involved...
Identifier: 4561235
Version: 1
Description: No description
Date: Tue Mar 31 14:34:00 2009
Unfreezing application in which process 4561236 is involved...
```

```
$ restart 4561236 1
Restarting application 4561235 (v1)...
```


Point d'arrêts/reprise : cas particuliers

- **Cas des fichiers ouverts**
  - Ouvert en lecture : rien à faire
  - Ouvert en écriture : copie externe

Point d'arrêts/reprise : à venir

- **Points de reprises d'objets IPC SysV**
  - Segments mémoires
  - Sémaphores
  - File de messages
- **Point de reprise incrémental**
  - Ne sauvegarder que les données modifiées entre 2 points de reprise
- **Callbacks**
  - Avertir l'application des événement de point de reprise/redémarrage

Kerrighed
- Déploiement -

Vue générale

- **Environnement complet partagé par tous les nœuds de calcul**
  - Création d'un *chroot*
  - Partage du *chroot*
  - Démarrage des nœuds de calculs sur cet environnement
- **Pas de déploiement (boot réseau)**
- **NFSROOT peu utilisé : outils et documentation rares**
- **Scalabilité limitée (NFS) mais possibilité d'autre FS distribué**
  - OCFS
  - Lustre

Création du chroot

- **Chroot**
  - Répertoire qui contient un environnement complet, assez pour être monté comme racine '/'
- **Création : dépend de la distribution**
  - RedHat
    - `rpm -root <dir>`
    - `yum --installroot=<dir>`
  - Debian
    - `Debootstrap` :
      - Permet de choisir la distribution
      - Différents parfums « flavour »
  - Non spécifique : systeminstaller, etc.

Création du chroot : en pratique (1)

- **Créer le chroot de base**

```
$ mkdir -p /home/chroot/mycluster01
$ export CHROOT=/home/chroot/mycluster01
$ debootstrap squeeze $CHROOT http://ftp.fr.debian.org/debian
```

- **Monter un /proc valide dans le chroot**

```
$ mount -t proc none $CHROOT/proc
```

- **Ajouter les paquets essentiels**

```
$ chroot $CHROOT aptitude update
$ chroot $CHROOT aptitude install build-essential lsb-core lsb-release less
htop libncurses5-dev initramfs-tools openssh-server
```

- **Éditer /etc/fstab**

```
$ ( echo 'none    /proc     proc     defaults    0  0';
    echo 'none    /sys      sysfs    defaults    0  0';
    echo 'none    /config   configfs defaults    0  0 ) > $CHROOT/etc/fstab
$ mkdir -p $CHROOT/config
```

- **Supprimer la génération automatique du nom des interfaces réseau**

```
$ rm -f $CHROOT/lib/udev/rules.d/*-persistent-net-generator.rules
```

Création du chroot : en pratique (2)

- **Ajouter un accès SSH (clé)**

```
$ install -d -o root -p root -m 750 $CHROOT/root/.ssh/
$ cat /root/.ssh/id_rsa.pub > $CHROOT/root/.ssh/authorized_keys
```

- **Ajouter un accès mot de passe**

```
$ chroot $CHROOT passwd
```


Installation de Kerrighed

- **Disponibilité de Kerrighed**
  - Dernière release : kerrighed-2.4.1.tar.gz sur http://kerrighed.org
  - Dépôt git : git sur http://mirrors.git.kernel.org/
- **Compilation/Installation**
  - Utilisation des autotools
  - Depuis 2.4.1, entièrement automatisé
  - Scripts d'init existants pour Debian et CentOS/RedHat/Fedora

Installation de Kerrighed : en pratique (1)

- **Télécharger l'archive de Kerrighed**

```
$ wget -P $CHROOT/usr/src \
http://gforge.inria.fr/frs/download.php/23356/kerrighed-2.4.1.tar.gz
```

- **Extraire l'archive et configurer les sources**

```
$ tar xfC $CHROOT/usr/src/kerrighed-2.4.1.tar.gz $CHROOT/usr/src/
$ chroot $CHROOT
$ cd /usr/src && ./configure
```

  - Lors de la configuration, les sources du noyau Linux sont téléchargées, extraites, patchées et configurées.
  - Les sources de Linux sont d'abord recherchées dans /usr/src. Vous pouvez y placer le fichier linux-2.6.20.tar.bz2 pour éviter le téléchargement.
  - Voir les options de ./configure pour des variantes de configuration
    - --help : liste des options
    - --enable-tests : ajoute des utilitaires de test
    - --with-kernel-config[-file] : differents ways of configuring kernel sources
    - *etc.*

Installation de Kerrighed : en pratique (2)

- **Éventuellement configurer le noyau Linux**

```
$ make -C kernel menuconfig
```

- **Compiler et installer Kerrighed**

```
$ make -j`cat /proc/cpuinfo | grep ^proc | wc -l`
$ make install
```

- **Vous avez un environnement avec Kerrighed complet**
- **Nous allons maintenant le partager**
- **Sortons du chroot**

```
$ exit
```


Partage de l'environnement (1)

- **Installer les services nécessaires sur le serveur**

```
$ aptitude install nfs-kernel-server dhcp3-server atftp syslinux
```

- **Partager le chroot par NFS**

```
$ echo "$CHROOT 192.168.0.0/16(rw,async,no_root_squash,no_subtree_check)" >> \
  /etc/exports
$ invoke-rc.d nfs-kernel-server restart
```

- **Créer un fichier /etc/hosts valide et le partager**

```
$ ( for i `seq 1 16`; do \
      echo "192.168.0.$i node$(printf '%02d' $i)"; \
    done ) >> /etc/hosts
$ cp /etc/hosts $CHROOT/etc/
```


Partage de l'environnement (2)

- **Configurer DHCP/TFTP : exemple de /etc/dhcp3/dhcpd.conf**

```
ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;

option space PXE;
option PXE.mtftp-ip               code 1 = ip-address;
option PXE.mtftp-cport            code 2 = unsigned integer 16;
option PXE.mtftp-sport            code 3 = unsigned integer 16;
option PXE.mtftp-tmout            code 4 = unsigned integer 8;
option PXE.mtftp-delay            code 5 = unsigned integer 8;
option PXE.discovery-control      code 6 = unsigned integer 8;
option PXE.discovery-mcast-addr   code 7 = ip-address;

subnet 192.168.0.0 netmask 255.255.0.0 {
    option routers 192.168.0.254;
    allow unknown-clients;

    filename "/srv/tftp/pxelinux.0";

    next-server 192.168.0.254;
    server-identifier 192.168.0.254;

    pool {
        range 192.168.0.1 192.168.0.16;
    }
}
```


Partage de l'environnement (3)

- **Redémarrer le serveur DHCP**

```
$ invoke-rc.d dhcp3-server restart
```

- **Configurer le serveur TFTP (atftpd) et le redémarrer**

```
$ ( echo "USE_INETD=false"; \
    echo "OPTIONS='--daemon --port 69 --tftpd-timeout 300 --retry-timeout 5
--mcast-port 1758 --mcast-addr 239.239.239.0-255 --mcast-ttl 1 --maxthread 100
--verbose=5 /srv/tftp'" ) > /etc/default/atftpd
$ invoke-rc.d atftpd restart
```

- **Créer un initramfs qui gère le NFSROOT**

```
$ sed -i -e 's/^BOOT\s*=.*/BOOT=nfs/' \
   $CHROOT/etc/initramfs-tools/initramfs-tools.conf
$ chroot $CHROOT update-initramfs -c -k 2.6.20-krg
```

- **Partager le noyau, l'initramfs et le système pxelinux**

```
$ ln -s $CHROOT/boot/vmlinuz-2.6.20-krg /srv/tftp
$ ln -s $CHROOT/boot/initrd.img-2.6.20-krg /srv/tftp
$ ln -s /usr/lib/syslinux/pxelinux.0 /srv/tftp
```


Configurer le démarrage PXE

- **Créer un fichier de démarrage pxelinux**

```
$ mkdir -p /srv/tftp/pxelinux.cfg
$ edit /srv/tftp/pxelinux.cfg/mycluster01
```

```
DEFAULT kerrighed
LABEL kerrighed
KERNEL vmlinuz-2.6.20-krg
INITRD initrd.img-2.6.20-krg
ROOT /dev/nfs
APPEND rw nfsroot=192.168.0.254:/home/chroot/mycluster01,rsize=4096,wsize=4096,tcp session_id=1
```

- **Appliquer ce fichier aux nœuds du cluster**

```
$ for i in `seq 1 16`; do \
    ln -s mycluster01 /srv/tftp/pxelinux.cfg/$(gethostip -x 192.168.0.$i); \
  done
```

- **Démarrer les nœuds du cluster !**

Kerrighed
- Utilisation -

Démarrer le cluster

- **Plusieurs méthodes**
  - Automatique : démarrer le cluster quand un certain nombre de nœud Kerrighed est démarré.
    - Boot parameter : `nb_nodes_min`
  - Manuel :
    - Commande `krgadm`
      - `krgadm nodes` : voir les nœuds présents
      - `krgadm cluster start` : démarrer Kerrighed

Commandes spécifiques

- **Kerrighed s'utilise comme un SMP**
- **Commandes spécifiques**
  - krgcapset
  - migrate
  - checkpoint – restart
- **Enjoy !**

Ressources

- **Kerrighed**
  - Site web : http://kerrighed.org
  - Listes de diffusion : kerrighed.dev@listes.irisa.fr, kerrighed.users@listes.irisa.fr
  - IRC : #kerrighed@irc.freenode.net
- **XtreemOS : http://xtreemos.eu**
- **OSCAR : http://oscar.openclustergroup.org**

Questions

- ???

Nous contacter...

KERLABS

www.kerlabs.com

contact@kerlabs.com

KERLABS

80, avenue des buttes de coësmes
35000 RENNES - FRANCE

Tél : +33 6 81 97 23 97