Cloud Computing

Concepts de virtualisation[1]

Danilo Carastan dos Santos

danilo.carastan-dos-santos@univ-grenoble-alpes.fr

2024

[1] Adapté du support développé par Thomas ROPARS et Renaud LACHAIZE

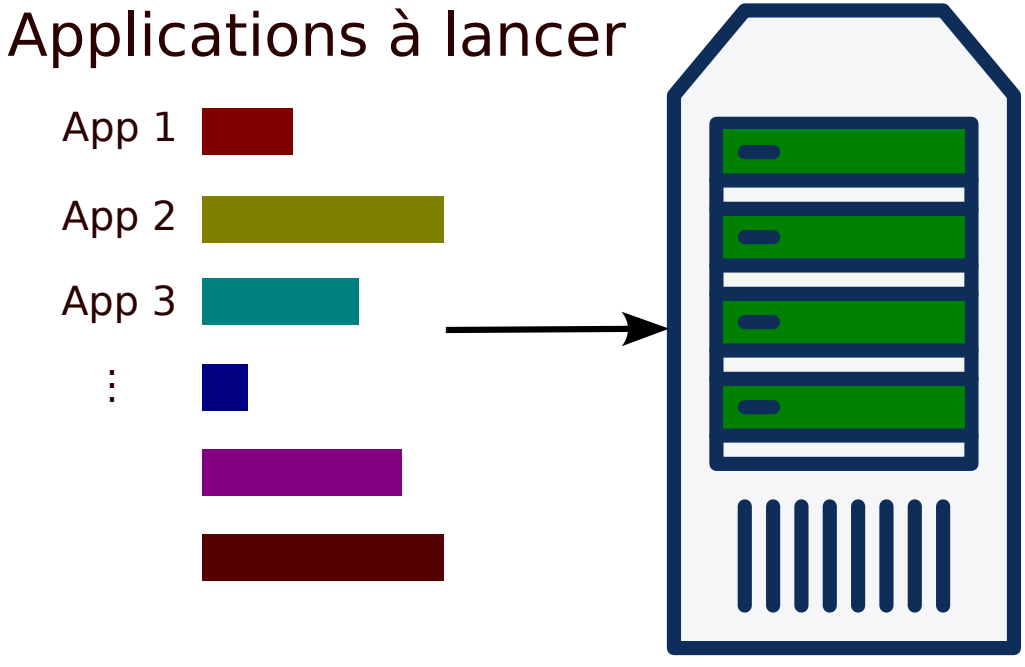
Problematique

Applications :

- Email
- Partage de données

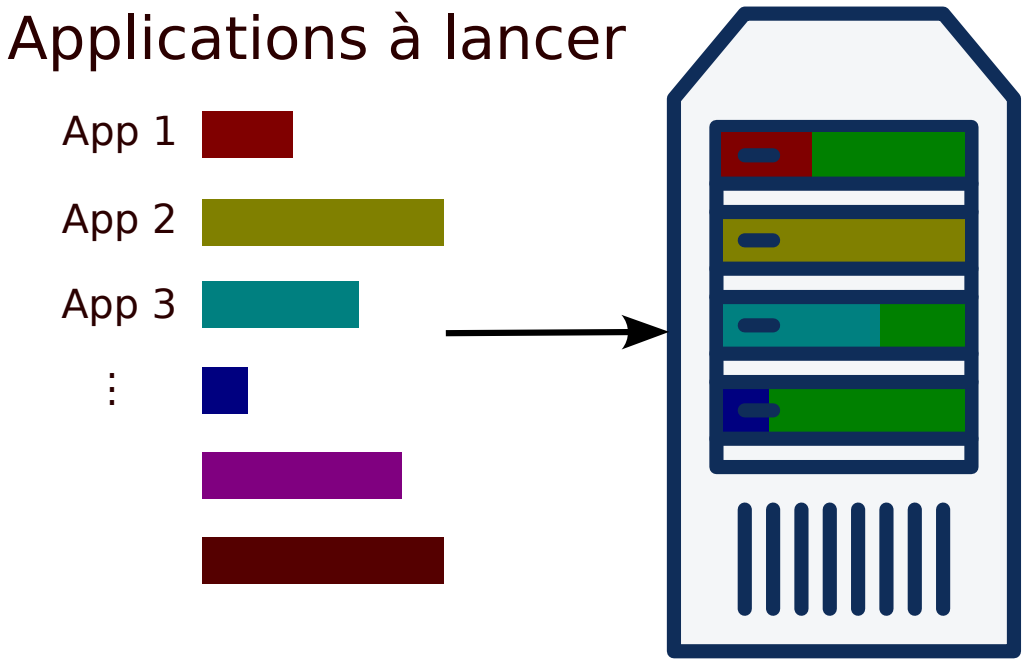
Chaque application a de prérequis spécifiques de ressources physiques/logiciels

- Utilisation de CPU
- Mémoire
- Stockage
- Système d'exploitation
- Librairies/Intergiciels spécifiques



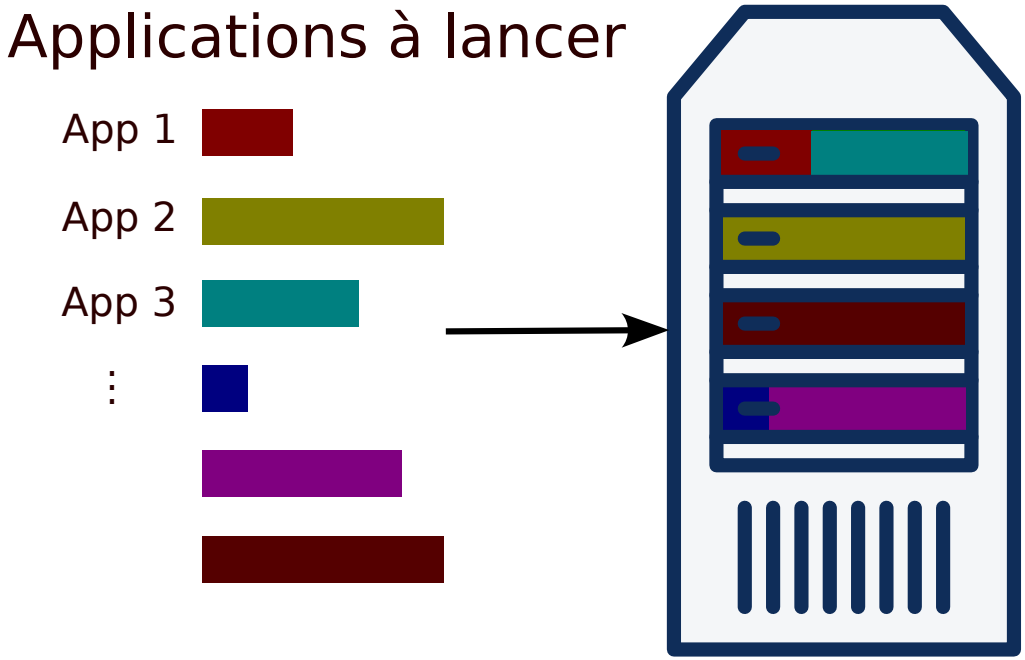
Une première solution naïve

- Une application par serveur
- Inconvénient : sous-utilisation des serveurs



Une deuxième solution

- Plusieurs applications par serveur
- Problèmes :
  - Contrôle de ressources
  - Isolation d'applications



Exemple :

- Apps A et B tournent dans un même serveur. App A a besoin une mise à jour du système d'exploitation. Cette MàJ peut affecter B.

Une meilleure solution : virtualisation

- **Idée :** Créer plusieurs ressources virtuelles (Machine Virtuelle - VM) à partir d'une ou plusieurs ressources physiques
- Dans une VM nous ne pouvons pas distinguer entre ressources physiques ou virtuelles
- **Concept clé :** Hyperviseur
  - Est une couche logicielle
  - Fournit l'abstraction d'une ou plusieurs machines "nues" (processeurs, mémoire et périphériques) au-dessus d'une véritable machine physique
- Intérêt :
  - Faire tourner plusieurs systèmes d'exploitation simultanément sur la même plateforme matérielle
  - Sauvegarder/rembobiner l'état d'un système
  - Sécurité (renforcement de l'isolation de certaines applications)

Virtualisation

- Hyperviseur de type I (natif) : Couche logicielle de plus bas niveau. Système d'exploitation spécialisé. Plus efficace
- Exemples : VMware ESX, Xen
- Hyperviseur de type II (hébergé) : Application intermédiaire (intergiciel) qui s'appuie sur un système hôte sous-jacent
- Moins efficace (plus de couches) mais plus simple à installer/utiliser
- Exemples : VMware workstation/fusion/player, VirtualBox

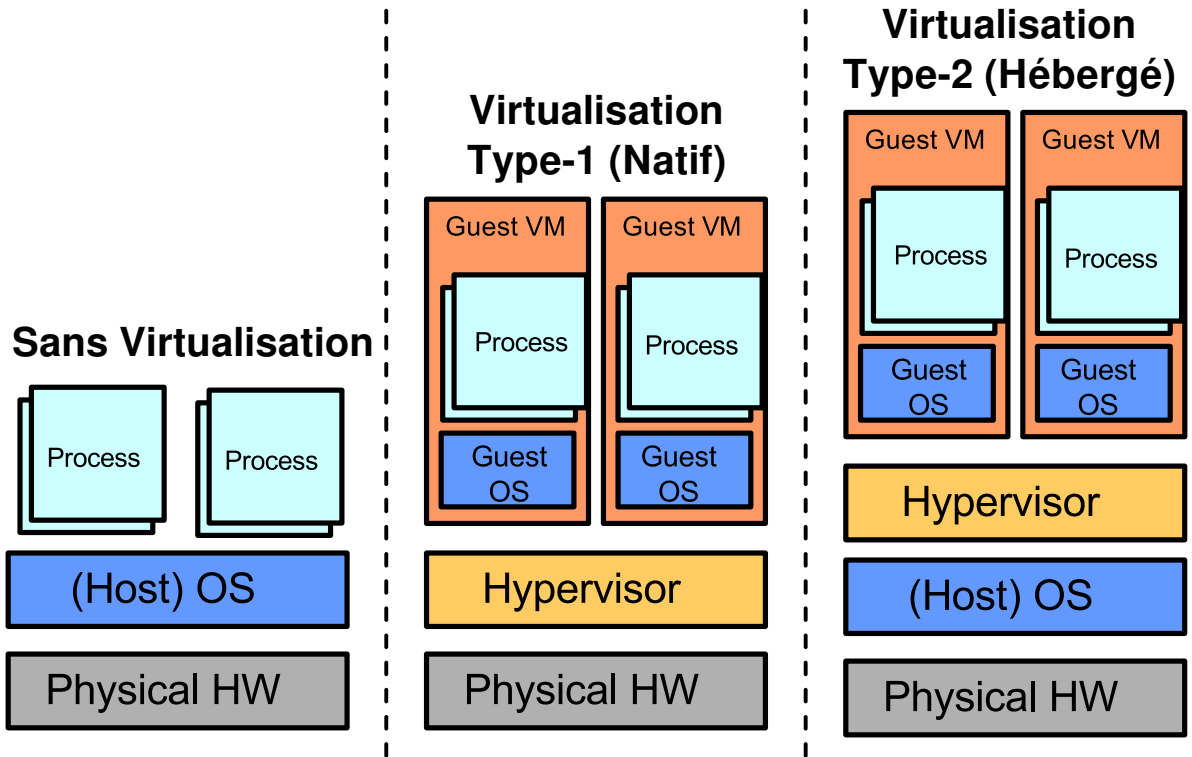


Image d'une machine virtuelle

- Snapshot of a virtual machine
- Allows for a faster deployment of VMs. No need to install OS + additional software every time a vm is launched

Scalabilité

Scaling

- Terme générique lié au comportement d'une application si la charge de travail et/ou les ressources changent
- Deux types :
  - **Scalabilité faible (Weak scaling) :** "Puis-je faire plus de travail (traiter plus de requêtes, traiter plus de données) si j'ai plus de ressources ?"
  - **Scalabilité forte (Strong scaling) :** "Puis-je faire la même charge de travail dans moins de temps si j'ai plus de ressources ?"

Elasticité

Autoscaling

- **Idée générale :** Adapter les ressources en fonction de la charge de travail courante
- Exemple : “Mon application de commerce électronique basée sur le Web devrait traiter n’importe quelle requête en moins de 500 ms, quel que soit le nombre total de requêtes qu’elle traite actuellement.”
- “Juste ce qu’il faut” :
  - Pas assez de ressources $\rightarrow$ traitement plus long (perte de qualité de service)
  - Trop de ressources $\rightarrow$ ressources non utilisées (et on les paye quand même)

Scalabilité Verticale

Vertical Scaling

- **Idée :** remplacer les machines (virtuelles) avec des machines (virtuelles) plus puissantes
- **Avantages :**
  - Plus simple
- **Inconvénients :**
  - Scalabilité limitée
  - Coût non linéaire de ressources