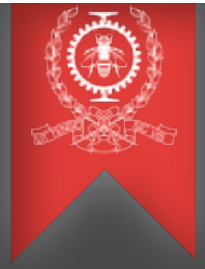


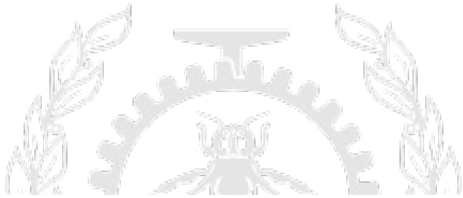
Processus serveurs pour l'infonuagique

Module 3
INF8480 Systèmes répartis et infonuagique

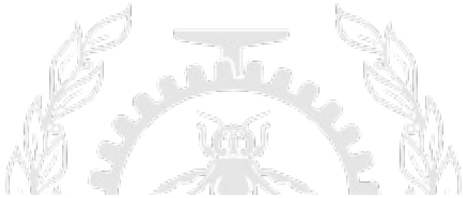
Michel Dagenais Raphaël Beamonte

École Polytechnique de Montréal
Département de génie informatique et génie logiciel

Sommaire

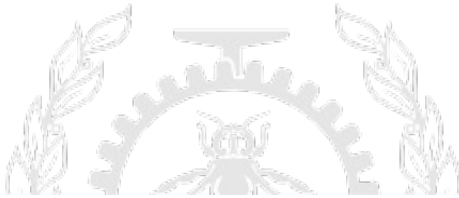


Processus serveurs pour l'infonuagique



L'infonuagique, qu'est-ce?

- Distributed and Cloud Computing, from Parallel Processing to the Internet of Things, 2011: virtualization, scalability, SOA, MOM, Security, Discovery, Databases, Grid, Peer-to-peer, Overlay networks, Fault tolerance, Ubiquitous computing, Internet of things, Wireless sensor networks, Social networks...
- John Gage, 1988, Sun Microsystems: the network is the computer!
- Jacques Gélinas, 2001: avec les vserver, chaque service est dans un serveur séparé de configuration plus générique.
- 2013, Quel modèle de serveur: cat or cow, pet versus cattle.
- Découpler les services des serveurs! Eliminer les serveurs individuels au profit de parcs de serveurs qui offrent une grande économie d'échelle.



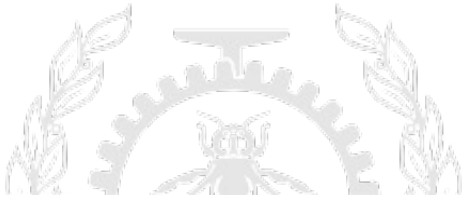
Environnements infonuagiques

- VMWare, 1998: consolider plusieurs services sur quelques serveurs redondants grâce à la virtualisation.
- Amazon EC2, 2006: instances d'ordinateurs à louer, "Infrastructure as a service" (IaaS). Virtualisation avec Zen.
- Eucalyptus, 2008: "Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems", clone du logiciel de EC2, initialement ouvert et ensuite à base ouverte.
- OpenStack, 2010: démarré par un large consortium de compagnies de haute technologie, en réponse à Eucalyptus qui n'était plus vraiment ouvert.
- Kubernetes, 2015: les conteneurs sont plus efficaces que les machines virtuelles, pour les infrastructures privées, ou par-dessus des machines virtuelles infonuagiques.

Le marché de l'infonuagique

- Amazon demeure la référence et le leader avec environ 33% du marché. Tous reprennent ses API pour assurer une transition facile.
- Microsoft a réussi à percer ce marché, avec une part d'environ 22% pour Azure, grâce à des prix agressifs (gratuit à l'essai), sa force de vente, et ses applications infonuagiques (Outlook, Office 365...).
- Google a réussi à se démarquer aussi (logiciels libres, interoperabilité, sécurité) avec 10% du marché.
- D'autres joueurs ont une présence importante comme Alibaba et IBM.
- OpenStack et VMWare sont utilisés par plusieurs fournisseurs Internet pour offrir des services infonuagiques.
- Pour leurs systèmes internes, plusieurs compagnies utilisent Kubernetes ou des solutions équivalentes de conteneurs.

Processus serveurs pour l'infonuagique

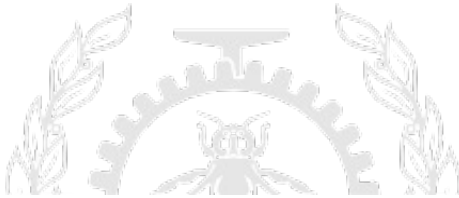


Virtualisation

- Conteneurs: plusieurs espaces de nom dans le système d'exploitation, gérés par le noyau (Linux LXC, Docker).
- Virtualisation logicielle: simulateur d'exécution (Bochs), traducteur dynamique (VMWare, Valgrind).
- Virtualisation matérielle: le processeur peut intercepter ou déléguer certaines instructions privilégiées afin de supporter efficacement la virtualisation (VMWare, KVM).
- Paravirtualisation: le système d'exploitation invité collabore en redirigeant ses requêtes vers le système hôte (Xen, VMWare, KVM).
- Hyperviseur (micro-noyau qui gère les interruptions et protections, e.g. Xen) ou système d'exploitation hôte (e.g. KVM).

Conteneurs

- Un seul noyau avec des espaces de noms séparés par conteneur pour les PID, IPC, usagers, réseau, /proc, hostname, fichiers.
- Chaque conteneur peut rouler une version différente des librairies, applications... mais il n'y a qu'un seul noyau en exécution.
- Aucun coût additionnel en performance, sauf la mémoire non partagée par les versions différentes, si c'est le cas.
- Linux LXC (aussi V-server, OpenVZ, Docker), FreeBSD jails, Solaris containers.

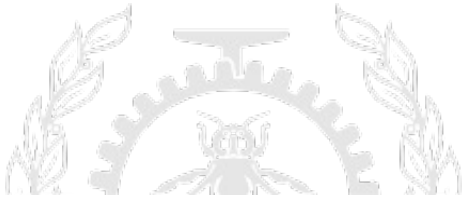


Simulateurs d'exécution

- Programme qui lit et interprète les instructions en simulant le matériel. L'hôte peut être un Intel et l'ordinateur simulé un ARM.
- Certains simulateurs peuvent aussi calculer le nombre de cycles écoulés et s'interfacer à GDB.
- Différentes techniques: interprétation une instruction à la fois, recompilation dynamique par segments, remplacement de certaines instructions et exécution directe des autres.
- BOCHS, QEMU, VMWare et VirtualBox sans support matériel, Valgrind.
- Environ 2 (remplacement de certaines instructions), 5 (recompilation dynamique) ou 50 (interprétation) fois plus lent.

Virtualisation matérielle

- Support matériel (Intel VT, AMD V) pour intercepter ou rediriger certaines opérations.
- Assigner un périphérique à une VM (PCI passthrough), démultiplexer par VM les arrivées de paquets dans la carte réseau, déléguer la table de pages...
- Linux KVM, VMWare et VirtualBox avec support matériel.
- Entre même vitesse et 2 fois plus lent selon le degré d'E/S et d'interaction avec le système d'exploitation.

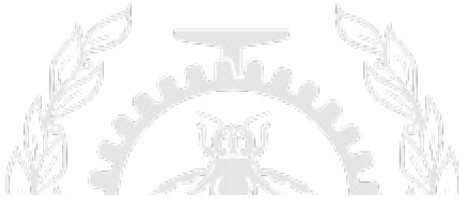


Paravirtualisation

- Le système d'exploitation est modifié pour faire un appel efficace au système d'exploitation hôte. Pas besoin d'intercepter les opérations d'accès au matériel et d'émuler le matériel.
- Plus d'une centaine d'opérations de bas niveau du noyau Linux (lire CR0, désactiver interruptions, lire bloc, changer table de page...) sont appelées à travers la table paravirt_ops qui pointe vers la fonction native ou virtualisée.
- Utilisé par Xen mais aussi VMWare, VirtualBox et KVM avec certains pilotes d'interface virtualisés (disque, réseau, affichage).
- Entre même vitesse et deux fois plus lent, selon le type de charge et les opérations qui sont virtualisées ou non.

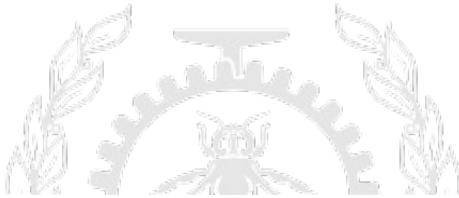
Hyperviseur

- Linux virtuel sous Windows réel ou l'inverse?
- Hyperviseur, système d'exploitation minimal qui gère les interruptions et les accès aux périphériques, pour les répartir entre les systèmes d'exploitation des machines virtuelles.
- Xen. Linux domaine 0 qui parle aux périphériques et Linux domaines 1, 2... qui sont les machines virtuelles invitées dont les requêtes sont passées par Xen au domaine 0.
- Xen peut maintenant accepter des invités Windows grâce au support de virtualisation matériel.
- Hyperviseur: solution élégante ou un OS de plus inutilement?



Bénéfices de la virtualisation

- Image logicielle isolée du matériel, utile lorsque les licenses sont attachées au matériel ou pour portabilité.
- Possibilité de cohabitation entre plusieurs systèmes d'exploitation ou versions, plutôt que double amorçage.
- Isolation des services à des fins de sécurité ou de gestion. Plusieurs serveurs virtuels de différents groupes peuvent coexister sur le même serveur physique.
- Modularisation des services: démarrer les serveurs virtuels voulus: base de donnée, courriel, Web...

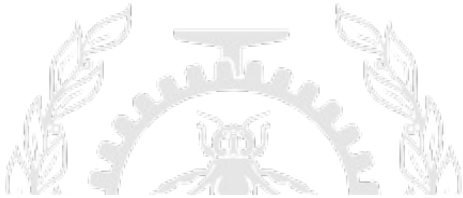


Coût de la virtualisation

- Certaines instructions causent des interruptions et sont émulées; moins avec le support matériel.
- Accès indirect aux périphériques; moins avec la paravirtualisation ou la virtualisation des I/O (IOMMU).
- Changements de contexte plus nombreux, application, système d'exploitation invité, hyperviseur; moins avec la délégation de tables de pages aux invités.
- Préallocation de la mémoire à chaque machine virtuelle (Xen), n'est pas toujours requis (Xen balloon, KVM).
- Surcoût d'avoir plusieurs copies en mémoire du noyau et des exécutables courants (libc, bash...); moins avec Kernel Samepage Merging.

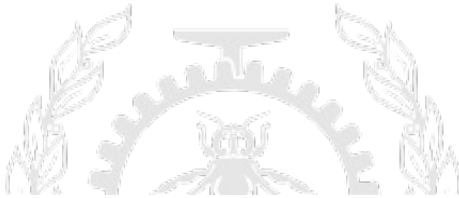
Virtualisation du réseau

- Réseaux et commutateurs virtuels à l'intérieur d'un noeud pour connecter les noeuds virtuels; Linux TUN/TAP (network tunnel, network tap).
- VLAN: réseau local virtuel séparé du reste du réseau local (étiquette ajoutée à chaque paquet Ethernet, gestion des diffusions générales sur le VLAN).
- VPN/VPLS: connexions multi-point virtuelles privées par-dessus le réseau public.
- Le résultat est un réseau dédié virtuel (overlay network); latence, bande passante, qualité de service...

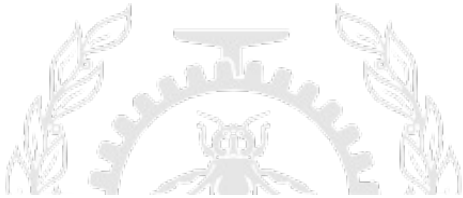


Migration

- Pour équilibrer la charge ou libérer le matériel qui requiert un entretien.
- Déplacer une image en exécution d'une machine virtuelle à l'autre; revient à migrer une machine virtuelle d'un ordinateur physique à un autre de manière transparente.
- Contraintes de même réseau local, mêmes fichiers accessibles, pas de 64 vers 32 bits, matériel virtuel identique.
- Copier toutes les pages de l'image en traçant celles qui sont remodifiées dans l'intervalle. Faire une seconde et possiblement troisième passe. Tout suspendre, copier les pages encore modifiées et poursuivre sur l'autre ordinateur.



Processus serveurs pour l'infonuagique



Serveurs virtuels/instances

- Plusieurs catégories de noeuds, et plusieurs "tailles" (CPU, mémoire, GPU, . . .)
- Au centre de plusieurs autres services qui travaillent en corrélation:
  - services de stockage;
  - services de bases de données;
  - répartition de requêtes réseaux;
  - service de mise à l'échelle (allocation +/- flexible de noeuds supplémentaires).
- Plusieurs zones de disponibilité.

EC2 **Nova** **Azure VM**

Images / modèles

- Fournit des images de systèmes utilisables directement, ou après avoir appliqué des modifications dessus (cliché d'un serveur).
- Possibilité de créer ses propres images Linux et de les importer; attention à utiliser les bonnes options selon le service utilisé (pour la paravirtualisation et les pilotes, par exemple).
- Le format accepté pour les images diffère selon le service, mais les formats communs comme vmdk (VMWare), vhd (Hyper-V) et raw (KVM) sont souvent acceptés.

AMI (Amazon Machine Image)

Glance

Azure VM Images

Enchères de calcul (services commerciaux)

- Il est possible de spécifier un prix de lancement d'instances pour un gros calcul à effectuer à bas prix.
- Lorsque le prix est sous le seuil spécifié, les instances demandées sont démarrées.
- Le prix fluctue sous le prix spécifié. Si le prix remonte, les instances peuvent être arrêtées.
- Requête unique ou persistente.
- Le prix "spot" est souvent moins de 30% du prix régulier.
- Modèle pour utiliser les instances qui vont et viennent (nombre variable de noeuds de travail).

Spot Instances **N/A** **Low-priority VMs**

Utilisation des instances

- Interface Web pour commander les instances, ligne de commande ou API.
- Définition de règles d'accès pour le groupe de sécurité contenant l'instance.
- Sélection d'une image, d'une taille d'instance, du nombre d'instances, de la zone de disponibilité, et du groupe de sécurité, puis démarrage.
- Des métadonnées sont disponibles pour chaque instance (paramètres, adresses, numéro d'instance. . .).
- Connexion par SSH à l'instance.

Services de stockage

Instance storage: stockage pour la durée d'une instance, avec l'image comme contenu initial de la partition racine.

Block Storage: partition de disque pour stockage permanent qui peut être attachée à une instance à la fois.

Object Storage: stockage permanent, extensible, accessible de plusieurs instances en lecture et écriture.

Shared file storage: partition montée via le réseau pour stockage permanent, qui peut être attachée à plusieurs instances à la fois.

Block: EBS
Object: S3
Shared: EFS

Block: Cinder
Object: Swift
Shared: Manila

Block: Page BLOB
Object: Block BLOB
Shared: Azure File Storage

Services de bases de données

- Bases de données relationnelles: généralement MySQL, Oracle ou PostgreSQL.
  - Pour les services commerciaux:
    - Différentes tailles possibles, payées à l'heure et au transfert.
    - Sauvegardes et mises à jour intégrées.
- Bases de données NoSQL:
  - Les services commerciaux ont souvent des solutions propriétaires qui acceptent ou non les requêtes de type SQL;
  - Les services à source ouvert utilisent des solutions à source ouvert existantes (Cassandra, MongoDB, . . .)
- Simplifie la conception de systèmes avec répartiteur de charge, instances de service élastiques sans données, et base de données centrale.

SQL: RDS (up to 16TB)
NoSQL: DynamoDB

SQL: Trove
NoSQL: Trove

SQL: SQL Database (up to 4TB)
NoSQL: CosmosDB

Répartiteur de charge

- Surveille un nom de noeud (e.g. www.macompagnie.com) et un numéro de port (e.g. 80).
- Répartit les requêtes reçues sur ce port entre les instances enregistrées pour le servir.
- Répartition entre les instances dans différentes zones de disponibilité.
- Maintien de métriques sur le niveau de service dans le répartiteur de charge: latence, nombre de requêtes, nombre d'instances en santé, etc.
- Envoi de la prochaine requête à l'instance répondant aux critères de répartition (architecture, zone géographique, mémoire vive, nombre de coeurs) la moins chargée (charge actuelle, nombre d'instances, puissance du noeud. . .)
- Arrêt d'envoi de requêtes aux instances non fonctionnelles.

ELB (Elastic Load Balancer) **LBaaS** **Azure Load Balancer**

Service de mise à l'échelle/nuage élastique

- Maintien d'un niveau de service en surveillant le nombre d'instances valides et le taux d'utilisation du CPU.
- Redémarre les instances non valides.
- Démarre de nouvelles instances si le taux d'utilisation du CPU dépasse un certain seuil.
- Arrête des instances si le taux d'utilisation du CPU descend sous un certain seuil.
- Les services commerciaux limitent souvent le nombre d'opérations de réduction pouvant être faites, il faut prendre plus de temps avant d'agir (e.g. CPU à 80% pendant 10mn = ajout d'une instance, CPU à 5% pendant 4h = retrait de X instances).

AWS Auto Scaling **Heat with Autoscaling** **Azure AutoScaling**

Surveillance

- Métriques mesurées régulièrement et conservées pendant un temps donné à propos des instances et volumes de stockage en blocs; certains services permettent de récupérer plus d'informations sur plus de systèmes.
- Instance: taux d'utilisation du CPU, accès en entrée et en sortie (opérations et octets), nombre d'octets envoyés et reçus par réseau.
- Block Storage: accès en entrée et en sortie (opérations et octets), temps de lecture et d'écriture, temps morts, longueur moyenne de la file.
- Émission de notifications/alarmes lorsque certains évènements se produisent.

Cloudwatch
(mesures aux 1 ou 5mn, conservées 2 semaines)

Synaps, Telemetry
(Ceilometer)

Azure Monitor

Gestion d'identité

- Répertoire des usagers.
- Authentification par mot de passe ou par jetons (et parfois de l'authentification en plusieurs temps).
- Permissions, rôles et politiques d'accès, qui permettent notamment un contrôle d'accès détaillé aux ressources.
- Fournit généralement une intégration du répertoire de l'organisation, via Windows Active Directory ou LDAP par exemple.
- Peut fournir une liste des services disponibles pour un utilisateur authentifié.

IAM (Identity and Access Management)

KeyStone

Azure Active Directory

Nuage privé dans le nuage public

- Réseau virtuel.
- Adresses IP choisies par l'utillisateur.
- Tables de routage.
- Couche de sécurité supplémentaire avec listes de contrôle des accès.
- Filtrage des sorties en plus des entrées pour chaque instance.
- Possibilité de limiter l'accès de l'Internet à une passerelle IPSec.

VPC
(Virtual Private Cloud)

Neutron

VNET
(Virtual NETwork)

Orchestration de travaux

- Format déclaratif qui peut facilement être mis dans un système de gestion du code source avec version.
- Outils pour activer la configuration définie par la recette.
- Définition des paramètres à spécifier.
- Déclaration de la configuration en utilisant les paramètres qui devront être fournis par l'usager.
- Utile pour organiser des tâches sur l'ensemble du système, faire des actions sur une partie du parc de machines virtuelles, etc.

AWS Batch **Heat** **Azure Batch**

Files de messages

- Systèmes de files de messages, avec publication/abonnement (publish/subscribe) ou notification.
- Pour par exemple: distribution de tâches à des noeuds, collecte de données, envoi de commandes à plusieurs destinataires, réception de réponses de multiples destinataires, outils de synchronisation en continu. . .
- Pour un très grand nombre de files, messages, destinataires. . .

SQS (Simple Queue Service)

Zaqar

Azure Queue Storage

Mégadonnées (*Big Data*)

- Services pour facilement déployer et mettre à l'échelle des systèmes de traitement de données comme Hadoop, Apache Spark, HBase, Presto, Hive, Flink. . .
- Réutilise en général les autres services disponibles (images, noeuds, stockage, authentification...), mais sans être gérés directement par l'utilisateur.
- Les services commerciaux rendent "invisible" cette utilisation des autres services, le service de mégadonnées étant facturé à part.
- Permet de gérer des cas communs d'utilisation des mégadonnées: analyse de fichiers journaux, indexage de sites web, transformation de données, apprentissage machine, analyses financières, simulation scientifique. . .

EMR
(Elastic Map Reduce)

Sahara (Savanna)

HDInsight

Les services pour l'infonuagique: Discussion

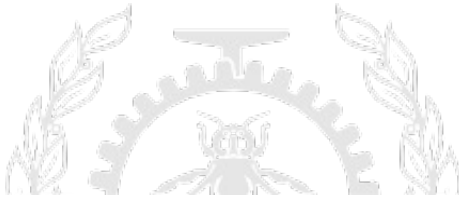
Utilisation des services commerciaux (AWS, Azure, ..)

- Permet de mettre sur pied une grappe ou un service Web de grande envergure très rapidement avec un coût initial presque nul.
- Evite les coûts de locaux ou d'équipes d'entretien.
- Attention, les frais d'utilisation et de transferts s'accumulent.
- Plus cher qu'une solution maison si on possède une très grande grappe, gérée efficacement et pleinement utilisée.
- Excellente solution pour les petites ou moyennes entreprises, la capacité excédentaire ou ponctuelle, comme plan de contingence, pour un démarrage rapide...

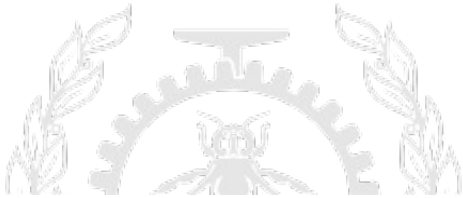
Les services pour l'infonuagique: Discussion

Qu'offre OpenStack pour un nuage privé vs AWS ou Azure?

- Convergence d'un grand nombre de joueurs autour de quelques technologies clé (Linux, KVM, API de AWS EC2, Réseaux virtuels).
- Infrastructure infonuagique entièrement libre qui offre des fonctionnalités semblables à ce qui a été mis de l'avant par Amazon/Azure.
- Progrès très rapide. Grandes différences d'une année à l'autre.
- L'essentiel de la fonctionnalité requise est maintenant disponible.

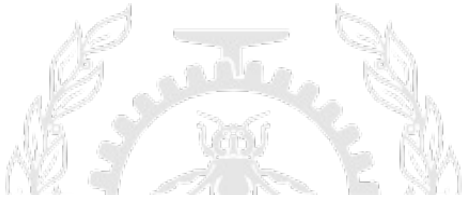


Processus serveurs pour l'infonuagique



Docker

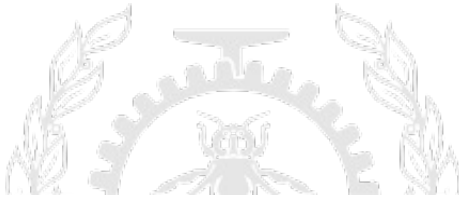
- C'est une image, c'est un conteneur, c'est une compagnie?
- Format d'image pour exécuter un conteneur sur Linux.
- Environnement d'exécution pour rouler une image sur Linux. . . ou sur d'autres systèmes comme Windows.
- Compagnie qui offre des outils de haut niveau pour gérer les conteneurs, au-delà des fonctions de base de Docker.
- Le format d'image et l'environnement d'exécution ont été transférés au Open Container Initiative de la Linux Foundation qui regroupe de nombreuses entreprises.



Dockerfile

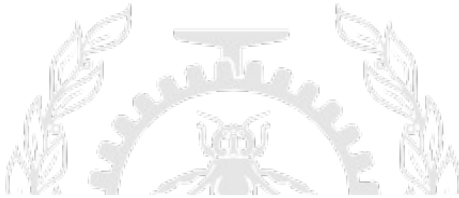
- Déclaration et recette pour créer et rouler une image.

```
FROM ubuntu:latest
RUN apt-get update
RUN apt-get install -y wget
RUN apt-get install -y build-essential tcl8.5
RUN wget http://download.redis.io/releases/redis.tg
RUN tar xzf redis.tgz
RUN cd redis-stable && make && make install
RUN ./redis-stable/utils/install_server.sh
EXPOSE 6379
ENTRYPOINT ["redis-server"]
```



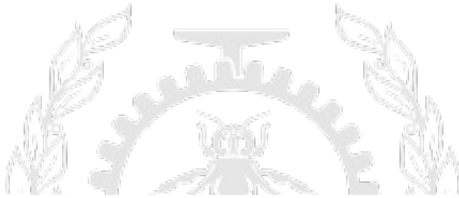
Kubernetes

- Contribué par Google au Cloud Native Computing Foundation de la Linux Foundation en 2015.
- Orchestration de conteneurs typiquement avec Docker.
- Rapidement supporté par les fournisseurs d'infonuagique et très populaire pour les nuages internes aussi.
- Peut être déployé par-dessus des noeuds natifs ou des machines virtuelles.
- Très bonne mise à l'échelle, si c'est assez bon pour Google...



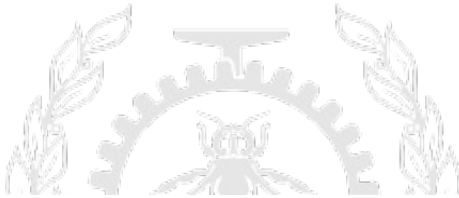
Kubernetes nodes

- Pod: groupe de conteneurs qui s'exécutent sur un même noeud pour offrir un service.
- Noeud: machine virtuelle ou noeud physique disponible pour rouler des conteneurs, qui exécute:
  - Docker runtime: engin pour exécuter les conteneurs;
  - Kubelet daemon: processus pour gérer, arrêter ou démarrer, les conteneurs sur le noeud;
  - Kube-proxy: processus pour gérer les communications, qui redirige les requêtes au bon conteneur;
  - Cadvisor: agent qui collecte diverses métriques qui peuvent être utilisées pour le monitoring ou pour gérer les pannes et la mise à l'échelle.



Kubernetes Cloud Controller Manager

- Control plane: orchestration des conteneurs, typiquement avec redondance, sur les noeuds à l'aide de:
  - Etcd: base de donnée clé-valeur, répartie et persistente, avec notification de changement, représentant la configuration désirée;
  - API server: reçoit les requêtes REST et accède etcd;
  - Ordonnanceur: choisit quel "pod" (groupe de conteneur) roule sur quel noeud.
  - Controller manager: collection de modules de commande qui gèrent l'orchestration des conteneurs pour la réplication, la mise à l'échelle...



Kubernetes service répliqué

```
$ kubectl create -f svc.yaml

# Répartiteur pour le service
apiVersion: v1
kind: Service
metadata:
  name: simpleservice
spec:
  ports:
  - port: 80
    targetPort: 9876
  selector:
    app: sise
```

```
$ kubectl create -f rc.yaml

# Conteneurs répliqués
apiVersion: v1
kind: ReplicationController
metadata:
  name: rcsise
spec:
  replicas: 2
  selector:
    app: sise
  template:
    metadata:
      name: somename
      labels:
        app: sise
    spec:
      containers:
      - name: sise
        image: img/serv:0.5.0
        ports:
        - containerPort: 9876
```

Kubernetes : Discussion

- Essor très rapide car technologie mature qui répond à un besoin très présent.
- Approche typique de Google avec mécanismes simples mais puissants qui se mettent bien à l'échelle.
- Pourquoi avoir la migration de conteneur lorsqu'on a déjà la tolérance aux pannes.
- Bon pour les déploiements où on contrôle bien l'ensemble de l'application car plus efficace mais moins transparent que les VM.
- D'autres outils similaires ont vu le jour comme Docker Swarm ou Apache Mesos
- Peut être déployé via des services pour l'infonuagique:

ECS
(EC2 Container Service)

Magnum

Azure Container Service

Charge serveur – EX. 3.1

Un serveur donné reçoit les requêtes de plusieurs clients. Le temps d'envoi d'une requête sur le réseau est de 2ms et le temps de réception de la réponse est aussi de 2ms. Le client prend 10 ms pour préparer chaque requête à envoyer et 5 ms pour traiter la réponse reçue, avant de passer à la requête suivante. Le serveur prend 8ms pour traiter chaque requête.

i) Si un seul client envoie ses requêtes au serveur, combien de requêtes par seconde seront envoyées par le client?

ii) Si le serveur traite séquentiellement les requêtes reçues avec un seul fil d'exécution, quel est le nombre maximum de requêtes pouvant être traitées par seconde ?

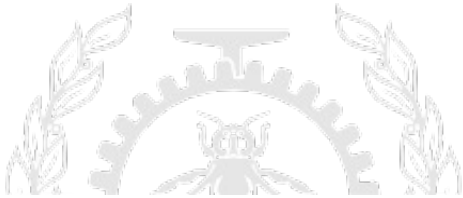
iii) Quel est le nombre maximum de clients qui peuvent être supportés (en supposant qu'ils envoient chacun le nombre de requêtes calculé en i) avant que le serveur ne soit utilisé à 100%

Charge serveur – Solution - EX. 3.1

i. Le traitement d'une requête prend 10 (préparation) + 2 (envoi) + 8 (serveur) + 2 (réception) + 5 (traitement) = 27 ms au total. Un client peut donc envoyer 1/0.027 = 37 requêtes / seconde.

ii. Au maximum, le serveur peut gérer 1/0.008 = 125 requêtes / secondes.

iii. Pour saturer le serveur, il faut avoir $n \times 37 \geq 125$ où n est le nombre de clients, soit $n \geq 3.3$.



Serveur –- EX. 3.2

Un module de micro-service dans un système reparti reçoit de nombreuses requêtes. Chaque requête occupe un cœur de CPU pendant 10ms et en plus, dans 20% des cas, un disque pendant 100ms. Ce service est exécuté sur 8 cœurs de CPU et 16 disques qui lui sont dédiés. On suppose que les requêtes sont bien réparties entre les cœurs de CPU et entre les disques. Chaque client qui fait appel à ce service génère en moyenne 10 requêtes par seconde. Le service utilise plusieurs fils d'exécution afin de servir en parallèle les requêtes.

i) Quel est le facteur limitant entre les cœurs de CPU et les disques?

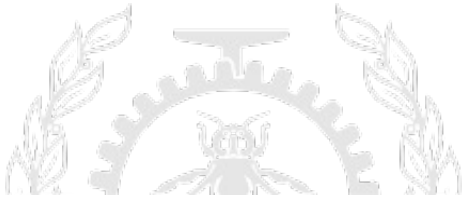
ii) Quel est le nombre maximal de clients possible avant que le service ne sature?

iii) Combien de fils d'exécution devrait-on rouler sur le serveur au minimum pour bien utiliser toutes les ressources disponibles?

Serveur –- Solution - EX. 3.2

i) Quel est le facteur limitant entre les cœurs de CPU et les disques?

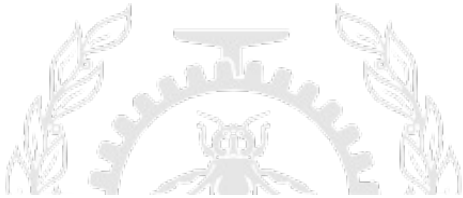
- Chaque requête prend 10ms sur un cœur de CPU →
- Chaque cœur peut servir 1000ms/s / 10ms/r = 100 requêtes / seconde.
- 8 cœurs de CPU chaque 100 requêtes/seconde = 800 requêtes / seconde.
- Chaque requête occupe dans 20% des cas un disque pendant 100ms → chaque requête prend en moyenne 0.2 x 100ms = 20ms de disque.
- Chaque disque peut servir 1000ms/s / 20ms/r = 50 requêtes / seconde → Pour les 16 disques le total est de 800 requêtes / seconde.
- CONCLUSION : C'est donc un système très équilibré avec les disques et les cœurs de CPU à un niveau équivalent, les deux étant limitants au même niveau.



Serveur –- Solution - EX. 3.2

ii. Quel est le nombre maximal de clients possible avant que le service ne sature?

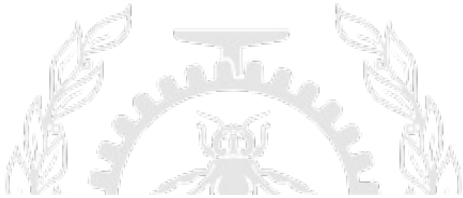
- Le système sature donc avec 800 requêtes / seconde
- Ce qui est équivalent à (800 requêtes / seconde) / (10 requêtes / client-seconde) = 80 clients



Serveur –- Solution - EX. 3.2

iii. Combien de fils d'exécution devrait-on rouler sur le serveur au minimum pour bien utiliser toutes les ressources disponibles?

- Si chaque requête reste dans le système 10ms CPU + 20ms disque = 30ms en moyenne
- 800 requêtes/seconde x 1seconde/1000ms x 30ms-thread/requête = 24 threads, soit 24 fils d'exécution.



Serveur –- EX. 3.3

De nombreux clients font chacun 8 requêtes par seconde vers un serveur. Chaque requête demande au serveur 5ms sur un cœur de CPU et en plus, dans 40% des cas, la lecture d'un disque pendant 20ms. Chaque serveur possède 2 cœurs de CPU et 4 disques. Les requêtes sont bien réparties entre les cœurs de CPU et entre les disques. Le service utilise plusieurs fils d'exécution afin de servir en parallèle les requêtes.

i) Quel est le facteur limitant entre les cœurs de CPU et les disques?

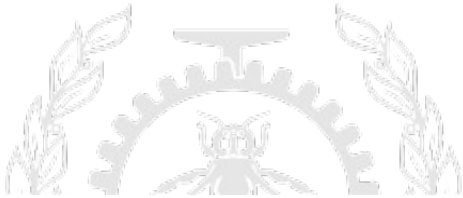
ii) Quel est le nombre maximal de clients possible avant que le service ne sature?

iii) Combien de fils d'exécution devrait-on rouler sur le serveur au minimum pour bien utiliser toutes les ressources disponibles?

Serveur –- Solution - EX. 3.3

i) Quel est le facteur limitant entre les cœurs de CPU et les disques?

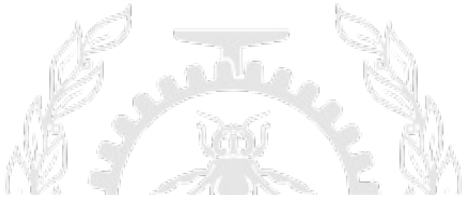
- Chaque requête prend 5ms sur un cœur de CPU →
- Chaque cœur peut servir 1000ms/s / 5ms/r = 200 requêtes / seconde.
- 2 cœurs de CPU chaque 200 requêtes/seconde = 400 requêtes / seconde.
- Chaque requête occupe dans 40% des cas un disque pendant 20ms → chaque requête prend en moyenne 0.4 x 20ms = 8ms de disque.
- Chaque disque peut servir 1000ms/s / 8ms/r = 125 requêtes / seconde → Pour les 4 disques le total est de 500 requêtes / seconde.
- CONCLUSION : Le facteur limitant est les cœurs de CPU qui peuvent supporter 400 requêtes / seconde



Serveur –- Solution - EX. 3.3

ii. Quel est le nombre maximal de clients possible avant que le service ne sature?

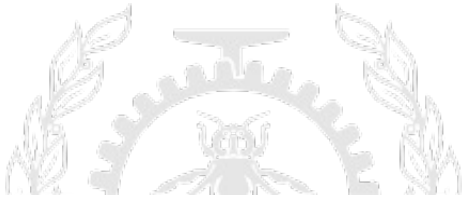
- Le système sature donc avec 400 requêtes / seconde
- Ce qui est équivalent à (400 requêtes / seconde) / (8 requêtes / client-seconde) = 50 clients



Serveur –- Solution - EX. 3.3

iii. Combien de fils d'exécution devrait-on rouler sur le serveur au minimum pour bien utiliser toutes les ressources disponibles?

- Si chaque requête reste dans le système 5ms CPU + 8ms disque = 13ms en moyenne
- 400 requêtes/seconde x 1seconde/1000ms x 1ms-thread/requête = 5,2 threads, soit 6 fils d'exécution.



Serveur –- EX. 3.4

Vous êtes l'expert en infonuagique et des collègues viennent vous consulter pour savoir s'ils devraient utiliser des machines virtuelles (e.g. avec KVM ou VirtualBox) ou des conteneurs (e.g. Kubernetes et Docker).
Le premier collègue veut rouler sur le même système matériel, qui roule Linux, de nouveaux services sur Linux, ainsi qu'un ancien service fourni par une ancienne application Windows.
Le second collègue doit bâtir un serveur qui compile une application pour plusieurs versions différentes de distributions Linux (e.g. Ubuntu 20.04, Ubuntu 22.04, Fedora 35, Fedora 36). Pour chaque version de distribution, il faut utiliser la bonne version de compilateur et les bonnes versions de librairies, mais la version du noyau du système d'exploitation Linux importe peu.

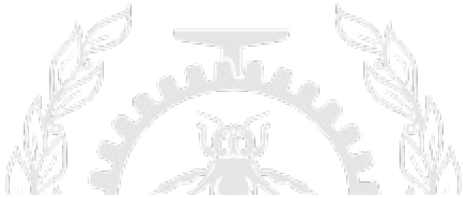
Que suggérez-vous d'utiliser, machine virtuelle ou conteneur, dans chaque cas? Pourquoi?

Serveur – Solution - EX. 3.4

Que suggérez-vous d'utiliser, machine virtuelle ou conteneur, dans chaque cas? Pourquoi?

- Pour le premier cas, pour exécuter une application Windows, il faudra prendre une machine virtuelle afin de pouvoir exécuter ce système d'exploitation différent.

- Pour le second cas, des conteneurs feront très bien l'affaire. Chaque conteneur peut venir avec une distribution différente (librairies, compilateurs...) mais ils sont contraints à utiliser le noyau Linux de l'hôte, ce qui était spécifié comme n'étant pas un problème. Lorsque c'est possible, on préfère utiliser des conteneurs, plutôt que des machines virtuelles, car le surcoût est beaucoup moindre et le temps de démarrage est beaucoup plus court.



Serveur - EX. 3.5

Pour les différents services infonuagiques comme Amazon EC2, on parle de stockage d'instance, de stockage de bloc (EBS), et de stockage d'objets (S3). Expliquez les différences entre ces trois types de stockage.

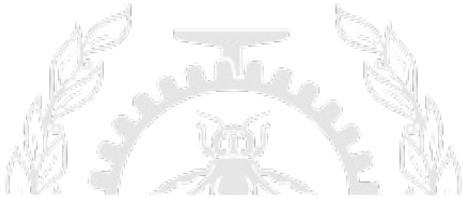
- Le stockage d'instance est propre à chaque activation de l'instance. Son contenu est perdu lorsque l'instance est arrêtée.

- Le stockage de bloc est comme un disque local. Son contenu persiste après l'arrêt de l'instance et il peut être accédé à nouveau par une nouvelle instance. Un stockage de bloc ne peut toutefois être attaché qu'à une seule instance à la fois, comme un disque local.

- Le stockage d'objets est comme une page Web qui supporte GET et PUT. On peut lire son contenu complet, ou remplacer son contenu complet, mais il n'est pas possible d'en remplacer seulement une partie.

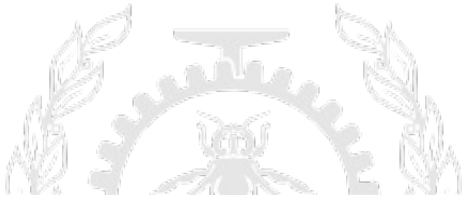
Empiler Kubernetes et OpenStack - EX. 3.6

Peut-on rouler Kubernetes par-dessus OpenStack? OpenStack par-dessus OpenStack? Qu'en est-il de OpenStack par-dessus Kubernetes et Kubernetes par-dessus Kubernetes? Expliquez pour chaque cas comment cela peut ou non se faire, et commentez sur l'opportunité et l'efficacité d'une telle configuration.



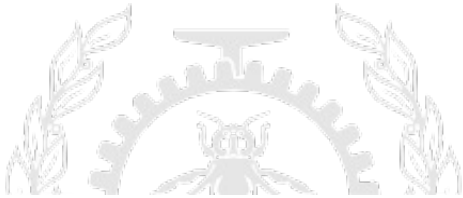
Empiler Kubernetes et OpenStack - Solution - EX. 3.6

Kubernetes par-dessus OpenStack est possiblement la configuration la plus courante. Le fournisseur infonuagique offre des machines virtuelles via OpenStack et le client utilise cela pour un déploiement Kubernetes. Il n'y a qu'un seul niveau de virtualisation, ce qui est supporté par matériel et la performance est adéquate. L'inverse est aussi fréquent. OpenStack est assez compliqué à déployer et une configuration au-dessus de Kubernetes permet de simplifier grandement son déploiement. Un fournisseur infonuagique pourrait facilement déployer OpenStack par-dessus Kubernetes pour offrir des machines virtuelles à ses clients. Encore là, il n'y a qu'un seul niveau de virtualisation et la performance est bonne. Le client peut encore ici rouler Kubernetes par-dessus OpenStack (qui est par-dessus Kubernetes) sans problème et avec une bonne performance.

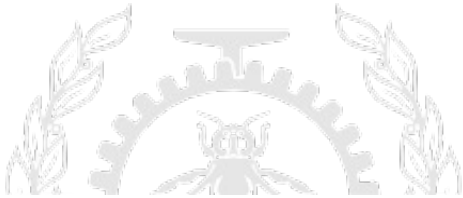


Empiler Kubernetes et OpenStack - Solution - EX. 3.6

Rouler Kubernetes par-dessus Kubernetes est moins fréquent mais est possible avec une bonne performance. La difficulté est que certaines opérations pour configurer les conteneurs de Kubernetes requièrent les privilèges d'administrateur. Les conteneurs au premier niveau devront donc être des conteneurs avec des privilèges spéciaux. OpenStack par-dessus OpenStack, avec la virtualisation à chaque niveau, serait possible mais peu performant. Par contre, il est possible d'utiliser OpenStack en mode bare metal (un système informatique qui exécute des instructions directement sur le matériel sans système d'exploitation) pour orchestrer le déploiement de OpenStack par-dessus des nœuds physiques commandés par des modules de gestion de nœud (e.g., IPMI ou AMT). Tout comme OpenStack par-dessus Kubernetes, ceci est un moyen d'automatiser le déploiement de OpenStack.

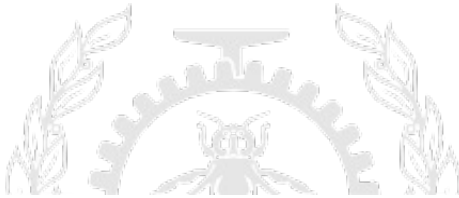


Processus serveurs pour l'infonuagique



Conclusion

- Les ordinateurs, comme les voitures, deviennent interchangeables; un signe de maturité.
- Le temps où un technicien s'occupait de 1 à 10 ordinateurs est révolu.
- Le gouvernement américain veut réduire le nombre de ses centres de données de plus de 1200 (sur environ 3000).
- Les ventes de serveurs sont en mutation.



Résumé