Université de Reims Champagne Ardennes

Mise en place d'une solution dévaluation des performances de solutions de virtualisation, basée sur une expérience utilisateur

Auteur :
HERARD Joffrey

Responsable :
Olivier FLAUZAC

20 avril 2017

Résumé

L'objet de notre étude a porté sur l'ensemble des solutions de virtualisation disponibles et possibles dans le cadre de la machine support mise à disposition. Il semble essentiel de rappeler que ces solutions sont variées dans leur fondamentalisme, leurs principes ou bien encore dans leurs domaines traditionnels d'utilisation. Afin de mettre en place une solution d'évaluation des solutions de virtualisation du point de vue des utilisateurs nous avons défini des scénarios de tests et des domaines d'études spécifiques de test sur chaque composant du système qui pourrait être utilisés par un ensemble d'utilisateurs. Pour cela, il a fallut définir ce qui allait être étudié sur chaque composant mais aussi comment mettre en oeuvre ces tests sur l'ensemble de ces machines. Je pus alors me poser un certain nombre de questions : Quels outils existent ? Comment faire la mise en place de ces outils ? Comment faire l'analyse de l'ensemble des résultats ? Comment faire l'établissement des liens et d'une échelle de performances ? Comment faire pour délimiter les meilleurs domaines de chaque solutions de virtualisation ?

Table des matières

Chapitre 1

Présentation du projet

Au travers de cette présentation il sera présenté l'ensemble du sujet, ainsi que les problématiques qu'il soulève, une définition précise des différents termes employés, l'explication des choix qui ont été fait sur certaines technologies et enfin l'ensemble des solutions émises pour résoudre les problématiques levées.

1.1 Sujet

Ce stage ayant pour sujet la mise en place d'une solution d'évaluation des performances de solutions de virtualisation basée sur une expérience utilisateur, il a tout d'abord été nécessaire de définir plusieurs termes afin de cerner ce dont-il était vraiment question. Ainsi, j'ai d'abord déterminé les termes qu'il était intéressant de développer, tels que : les solutions de virtualisation, l'évaluation de ces solutions et l'expérience utilisateur. Pour commencer, portons notre attention sur les solutions de virtualisation afin d'illustrer de quoi il s'agit.

Définition 1. *Solutions de virtualisation : La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d'exploitation / applications comme un simple logiciel, sur un ou plusieurs ordinateurs. Actuellement séparé en deux grands domaines, lhyper-vision (eux même divisés en deux types : Type 1 et type 2) et la conteneurisation.*

Après avoir défini en détails les termes des solutions de virtualisation il est nécessaire d'expliquer le processus d'évaluation des performances.

1.2 Objectifs

1. Savoir évaluer les différentes solutions de virtualisation.
2. Effectuer une démarche scientifique cohérente.
3. Effectuer un travail collaboratif avec des personnes dexpériences sur le sujet, ce qui malgré tout est plus ou moins inédit pour moi (Jentends que la plupart des travaux réalisés ces dernières années ont été fait avec des gens de même expérience que moi).
4. Établir des statistiques, sur les résultats qui seront obtenus. En ressortir des résultats des variables qui sortent du lot.
5. Établir un tableau récapitulatif afin de pouvoir résumer toutes les données obtenues.

1.3 Problématique soulevée

Comment mettre en oeuvre des scénarios afin d'évaluer les différentes solutions de virtualisation et de conteneurisation du point de vue utilisateurs ? Tout cela de manière efficace et juste sans interférer dans les résultats.

1.4 Hypothèse de solution

Il existe un ensemble de logiciels type pour le provisionning de machine virtuelle ainsi que des outils permettant de réaliser des benchmark neutres dans leurs prises de ressources.

Chapitre 2

Analyse de l'existant

Dans ce chapitre, nous verrons quelles technologies existent pour répondre à notre problématique.

2.1 Virtualisation

2.1.1 Hyperviseurs

Définition 2. *Hyperviseurs : C'est une plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps. Il en existe deux catégories :*

- *La première bien nommée Type 1 : est un logiciel de virtualisation installé directement sur le matériel informatique, il contrôle non seulement le matériel, mais aussi un ou plusieurs systèmes d'exploitation invités.*
- *La deuxième bien nommée Type 2 : sont des applications de virtualisation qui sexécutent non pas directement sur du hardware mais sur un système dexploitation.*

Il existe un ensemble d'Hyperviseur de type 1, que l'on peut lister de manière non exhaustive :

- CP
- XEN
- ESX Server
- LPAR
- Hyper-V
- Proxmox
- KVM

Il existe un ensemble d'Hyperviseur de type 2, que l'on peut lister de manière non exhaustive :

- VMWare Server
- VMWARE Workstation
- QEMU
- Hyper-V
- Parallels Workstation
- Parallels Dekstop
- VirtualBox

2.1.2 Conteneurs

Définition 3. *Conteneurs : C'est de créer des instances dans un espace isolé au lieu. En d'autres termes, le partage du matériel est de rendre disponible de nombreux opérateurs sur le noyau lui-même plutôt que d'une autre couche.*

Il existe un ensemble de gestion de conteneurs, que l'on peut lister de manière non exhaustive :

- LXC
- Docker

2.2 Provisionning

Définition 4. *Provisionning : C'est l'approvisionnement de machines, afin de mettre en place des configurations, des allocations automatiques de ressources, voir même des installations de logiciel, gestions de configuration, maintenance système. Globalement il sert à faire de la gestion de groupe de machines.*

Initialement le provisioning était du scripting manuel, voir même des solutions Client/Serveur, avec un serveur de configuration et un ensemble d'agents de gestion placés sur les machines à administrer. On appelle cela un framework dexécution distantes depuis une station de gestion. Le scripting a malgré tout des limites, cela représente un travail fastidieux, il fait souvent face à un problème d'hétérogénéité. On peut se retrouver face un script spécifique pour une opération, par serveur, par service. Voici une liste :

— Ansible
— Vagrant
— Saltstack
— Libvirt

2.3 Benchmarking

Une évaluation des performances est appelée souvent dans le monde anglophone un Benchmark. Ce terme est défini comme suit.

Définition 5. *Benchmarking : Évaluation des performances d'un système par simulation des conditions réelles d'utilisations.*

Il existe un certain nombre de benchmark référence en majeur partie par OpenBenchmark.com, les sources de ces benchmark sont réalisées avec le célèbre outil nommé Phoronix tests suite.

Chapitre 3

Analyse des besoins

Dans cette troisième partie, il va être abordé l'ensemble des besoins qui permettront d'émettre un début de réponse par rapport au sujet et à la problématique.

3.1 Plan d'expériences

Après une analyse des besoins fonctionnels du projet, il va être détaillé chaque étapes du test.

3.1.1 Sur les tests

Lors des tests qui seront exécutés il sera nécessaire de choisir précisément quoi et comment évaluer chaque composants, on l'a dit auparavant il y a un ensemble de scénarios à définir. Pour autant de ces scénarios effectuer différentes variantes de tests pour chaque technologies de virtualisation, et de conteneurisation. Afin d'avoir des résultats cohérents, la neutralité de la sonde doit être importante. Définir la composante environnementale pour chaque tests, ce qui est appelé environnement est ce qui ne serait pas juger comme influent à travers mes tests pourtant il le sera, je peut citer par exemple la vitesse du processeur, sa température, ou encore la vitesse de connexion pour les tests réseaux, bien entendu l'alimentation des composants qui peut varier, la politique de gestion des ressource vis à vis de chaque technologies de conteneurisation et dhypervisions. Pour chaque composant la question sera toujours la même, à savoir :

Pourquoi tester et tester quoi ?

Pour chaque composants détaillés ci dessous, un schéma est en annexes concernant le détail du cheminement de chaque tests possibles pour chaque éléments.

3.1.2 Expérimentation disque dur

3.1.2.1 Pourquoi tester le disque dur ?

Le disque dur peut être utilisé lors d'enregistrement de fichiers, assez simplement du téléchargement de fichier, de lécriture pure, de l'enregistrement d'évènements locaux. Il est le membre de la machine le plus important pour le stockage de données.

3.1.2.2 Tester le disque dur sur quoi ?

Dans la façon de voir, l'architecture de lordinateur pour les tests, est importante, notamment la taille des blocs pour lécriture ou la lecture. L'évaluation sur le test, doit-il se faire avec un accès en séquentielle ou en aléatoire ? On peut résumer de manière textuelle ainsi :

— Test de Performance en Écriture
  1. Évaluation des temps de réponses.
  2. Évaluation des Vitesse de transfert.
  3. Évaluation de la meilleurs performance possible (Maximum en vitesse écriture)sur des fichiers de différente taille.

— Test de Performance en Lecture
  1. Évaluation des temps de réponses.
  2. Évaluation de la meilleurs performance possible (Maximum en vitesse lecture) sur des fichiers de différente taille.

— Évaluation des vitesse du cache.

3.1.3 Expérimentation processeur

3.1.3.1 Pourquoi tester le processeur ?

Le processeur peut être sollicité lors de calculs, notamment dans le cadre des hautes performances.

3.1.3.2 Tester le processeur sur quoi ?

Un CPU peut être testé sur cette ensemble de choses-ci :
— Évaluation de la vitesse de la lecture Mémoire.
— Évaluation de la vitesse de la écriture Mémoire.
— Évaluation de la vitesse de la copie Mémoire.
— Évaluation de la vitesse de calcul flottant à précision simple.
— Évaluation de la vitesse de calcul flottant à précision double.
— Évaluation des opérations d'entrée sortie par secondes.
— Évaluation des opérations de chiffrement.
— Évaluation des opérations de hachage.

3.1.4 Expérimentation carte graphique

3.1.4.1 Pourquoi tester la carte graphique ?

3.1.4.2 Tester la carte graphique sur quoi ?

— Évaluation de la vitesse de la lecture Mémoire.
— Évaluation de la vitesse de la écriture Mémoire.
— Évaluation de la vitesse de la copie Mémoire.
— Évaluation de la vitesse de calcul flottant à précision simple.
— Évaluation de la vitesse de calcul flottant à précision double.
— Évaluation des opérations d'entrée sortie par secondes.
— Évaluation des opérations de chiffrement.
— Évaluation des opérations de hachage.

3.1.5 Expérimentation réseaux

3.1.5.1 Pourquoi tester le réseaux ?

3.1.5.2 Tester le réseaux sur quoi ?

— Chaque tests doit être réalisés avec des paquets de taille croissante avec le temps.
— Vitesse de Download à estimer.
— Vitesse d'Upload à estimer.

3.2 Choix sur les outils de virtualisation

Comme précédemment, nous avons choisi de distinguer deux catégories pour les besoins non-fonctionnels. D'une part, nous avons les besoins non-fonctionnels pour les [...], et d'autre part ceux pour [...]. Nous avons aussi pris en compte les contraintes de développement, que nous détaillerons à la fin de cette partie.

Il est nécessaire pour être équitable. Et surtout au vue des différentes définitions apportées hier et aujourd'hui. De repartir nos tests sur l'ensemble de ces 3 technologies qui nous sont accessibles. Par conséquent il faut être cohérent dans nos choix. De plus il y a aussi à considérer cet effet de mode sur le fait d'utiliser des conteneurs absolument partout et pour n'importe quoi par conséquent loutil Docker et l'outil LXC qui sont assez populaires sont à étudier. Premièrement Docker, il est nécessaire d'évaluer si Docker a effectivement tous les points positifs que on lui vente. Tout comme LXC, sachant de plus que Docker est basé sur LXC. Il serait intéressant dévaluer si cette technologie avec la couche qu'apporte docker a un vrai plus ou un moins.

Donc on choisi d'évaluer la technologie de conteneurisation proposée par les conteneurs Linux LXC, et celle par Docker.

Maintenant si on se penche sur le cas des Hyperviseur, si on part du même constat. Déjà il y a deux catégorie d'Hyperviseurs, comme dit plus haut, les types 1 & 2. Donc il faut établir des tests sur ces deux types d'hyperviseurs. On peut donc choisir tout comme Docker et LXC, partir sur la popularité de chacun. Il serait intéressant donc de prendre le logiciel d'Oracle VirtualBox. VirtualBox est un hyperviseurs de Type 2 . On peut notifier aussi que cet hyperviseur utilise des technologies comme celles de QEMU qui lui est de Type 2 aussi. Soit l'on peut choisir donc Virtualbox.

Maintenant il reste à évaluer KVM, QEMU, Hyper-V, VMWare, si nous les avions à les choisir, VMWare et Hyper-V serait très intéressant à évaluer. Tout dépend des licences qui sont à notre disposition mais ils sont tout deux des acteurs importants dans les hyperviseurs d'aujourd'hui. QEMU étant un Hyperviseurs de Types 2 il peut être intéressant de l'évaluer mis en face à face de VirtualBox.. KVM est un hyperviseur de type 1, intégré à Linux. Il est utilisé dans Proxmox VE

En somme, il faut bien savoir discerner les différents hyperviseurs à notre disposition, ainsi que de savoir discerner qu'est ce qui est pertinent. Je reviens donc, sur les choix émis hier. Il serait plus pertinent d'utiliser la technologie propre (Comme M.Flauzac l'a émis dans son mail ce matin 10 h 33) et non pas une surcouche. Un test est intéressant sur la technologie. Si on agrémente les technologies d'une surcouche, cela reste t-il tout aussi pertinent.

Par conséquent le choix s'oriente sur :

— LXC
— Docker
— KVM
— QEMU
— Hyper-V
— VMWARE

Table 3.1 – Solutions de virtualisations

Conteneur	Hyperviseur
LXC	KVM
Docker	QEMU
	Hyper-V
	VMWARE

3.3 Choix d'outils d'évaluation

Autrement dit cette section va parler des outils que l'on a auparavant appelé "sonde" dans les paragraphes précédents. On va parler d'outils permettant la répartition des tâches à faire pour ainsi exécuter une évaluation, ensuite on va parler de la récupération de ces données et de ce qui va être prélevé.

3.3.1 Outils d'évaluation personnel

Dans un premier temps après avoir observer ce qui existait comme technologie de Provisionning et de tests. La question c'est pose, si il était intéressant de devoir développer un outils de dévaluations. Après avoir vu loutil Phoronix, on se rend rapidement compte qu'il est compliquer de développer cet outil personnel et d'avoir des résultats a exploite sur la fin avant la fin du stages.

3.3.2 Phoronix

Par conséquent, il est nécessaire de choisir un outils dévaluations conséquent, Phoronix est un outils de suite de tests réputée et assez âgées, il n'est plus a sa version d'essai, il est déjà a ça septième version, plus précisément la 7.0.1 . Historiquement, le 5 juin 2008, la version 1.0 du Phoronix Test Suite a été annoncé. Ce logiciel a été publié sous licence GPLv3 et est conçu pour permettre aux utilisateurs de partager leur tests de logiciels et de matériel informatiques via une interface graphique.

3.4 Choix d'outils d'orchestration

Durant ce stages, il allait être question de pouvoir gérer un ensemble de machines. Un ensemble suffisamment grand qui rendrait le travail de scripting suffisamment fastidieux. Il fallait pouvoir gérer le déploiement des machines virtuelle rappelons le qui sont KVM/QEMU, LXC/Docker. et provoquer lexécution d'un ensemble de test puis de les rapatrier sur la machines dites maître. Il y a un ensemble d'outils qui on ete utilise durant ce stages Saltstack, Libvirt, Phoromatic.

3.4.1 Saltstack

Saltstack est un outils qui fonctionne avec une architecture Client/serveur, utilisant la technologies d'une file de message ZeroMQ (une autre technologies bien connu est celle de RabbitMQ).Il est dépendant,vis a vis de sont modèle client/serveur d'une écoute de port, il fallait donc envisager la possibilité de port bloquer. On appelle le serveur principale qui émet les ordres un «salt master» et les machines qui subissent les ordres des «salt minion»

3.4.1.1 Configuration Maitre

Il est nécessaire sur le serveur, de modifier les fichiers correspondant a la location /etc/salt/master, pour si on le souhaite modifier un ensemble de choses tels que, le réseaux, les ressources, la sécurité, les modules, l'architectures des fichiers, et les logs éventuels. Pour le réseaux on peut y paramétrer les interfaces, les ports découtes, les timeouts. Pour les ressources on peut configurer le nombre de fichiers ouverts, le nombre de travaux, le cache, etc...

3.4.1.2 Configuration Minion

Il est nécessaire sur le client, de modifier les fichiers correspondant a la location /etc/salt/minion, on doit y paramétrer le nom du serveur, et son adresse. On peut y gérer comme sur le master un ensemble d'architecture de fichier, une gestion des modules, la sécurité et les logs.

3.4.1.3 Détails sur le fonctionnement général

Un minion/client doit s'enregistrer auprès du serveur/master, pour pouvoir en subir les ordres. Pour cela le serveur doit être en mode acceptation. L'ensemble des opérations une fois les machines clientes accepter. Ce déroule avec la commande salt, en désignant les machines cibles, et une commande, une fonction personnel, un module a exécuter.

3.4.2 Phoromatic

Phoromatic, comme l'on peu aisément le devine est un outils propose par la suite Phoronix. C'est un serveur d'orchestration de lancement de test, réalise en PHP, utilisant les web-sockets. Typiquement il nécessite un lancement sur la machine que l'on souhaite, puis chaque machines visant a être évaluer doit avoir la suite de test phoronix préalablement installe et ensuite ce connecter au serveur avec une ligne de commande particulières en lui spécifiant l'adresse, le port, le token associe. Avec cet outils on choisi la programmations des test, sur quel support, sur quel machines, sur quel composant, a l'heure que l'on souhaite.

3.4.3 Libvirt

Libvirt est a lui tout seul un ensemble d'outils de gestions de machines virtuelles, et d'actions sur des machines virtuelles. Libvirt est capable d'effectuer une gestion du stockage, du reseaux, d'une installation, de clonage, de sauvegarde. Dans libvirt, l'hote des machines virtuelle, donc la machines physique est appele un noeud. l'hyperviseur est une couche logicielle de virtualisation avec des propriete generiques et specifique, s'executant sur un noeud. Libvirt est capable de gerer un ensemble d'hyperviseur tel que :

- XEN
- Qemu/KVM
- LXC
- OpenVz
- VirtualBox
- VMware ESX Workstation Player
- Hyper-V
- IBM PowerVM
- Virtuozzo

— Bhyve

Le principe de Libvirt, c'est qu'il réalise des opérations sur les domaine, la connexion a un hyperviseur ce fait par URL, n'ayant besoin que de l'hyperviseur et l'adresse. Lexécution des commandes ce fait en shell spécifique, ou en commande shell paramétré avec un fichier XML . Libvirt est fonde sur une API ecrites en : C, C++, C#, Java, PHP, Python, Ruby.

3.5 Impact des services de virtualisation sur les performances et donc les résultats

3.5.1 Impact d'un Hyperviseur

Sachant que le monde des Hyperviseurs est assez vaste, chacun à une politique différentes ne seras-ce que par son choix sur le type de son hyperviseur (1 et 2). Par conséquent il y a forcement des Hyperviseurs qui vont avoir une politique différente sur l'accès concurrent à des ressources. Les hyperviseurs ont besoin d'isoler les interruptions et les accès à la mémoire. C'est très coûteux en termes de performances. Les surcoûts en termes de performances pour virtualiser un système comportent trois aspects principaux : la virtualisation du processeur, de la mémoire et des entrées/sorties.

Processeur L'utilisation d'un hyperviseur au-dessous du système d'exploitation diffère du schéma habituel où le système d'exploitation est l'entité la plus privilégiée dans le système.De nombreuses architectures de processeur ne fournissent que deux niveaux de privilèges. Dans une virtualisation efficace.L'utilisation du processeur a des implications critiques sur les performances des autres caractéristiques du système.Dans l'évaluation des performances d'une machine virtuelle, les processus sont gérés par la machine virtuelle à la place du système d'exploitation sous-jacent. Les threads émulés des environnements multi-thread, en dehors des capacités du système d'exploitation d'origine, sont gérés dans l'espace utilisateur à la place de l'espace noyau, permettant le travail avec des environnements sans support natif des threads. De bonnes performances sur un microprocesseur multi-cur sont obtenues grâce à l'implémentation de threads natifs pouvant attribuer automatiquement le travail à plusieurs processeurs, ils permettent un démarrage plus rapide de processus sur certaines machines virtuelles.

3.5.2 Impact d'un conteneur

A la base, le concept de conteneurisation permet aux instances virtuelles de partager un système d'exploitation hôte unique, avec ses fichiers binaires, bibliothèques ou pilotes. Cette approche réduit le gaspillage des ressources car chaque conteneur ne renferme que l'application et les fichiers binaires ou bibliothèques associés. On utilise donc le même système d'exploitation (OS) hôte pour plusieurs conteneurs, au lieu d'installer un OS (et d'en acheter la licence) pour chaque VM invitée.Le conteneur de chaque application étant libéré de la charge d'un OS, il est nettement plus petit, plus facile à migrer ou à télécharger, plus rapide à sauvegarder ou à restaurer. Enfin, il exige moins de mémoire. La conteneurisation permet au serveur d'héberger potentiellement beaucoup plus de conteneurs que s'il s'agissait de machines virtuelles. La différence en termes d'occupation peut être considérable, car un serveur donné accueillera de 10 à 100 fois plus d'instances de conteneur que d'instances d'application sur VM.

Chapitre 4

Résultats

4.1 Partie processeurs

Intro

4.1.1 Scénario 1 : bla bla

Paragraphe 1 (n'apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

4.1.2 Scénario 2 : bla bla

Bla

4.1.3 Scénario 3 : bla bla

Bla

4.2 Partie Disque dur

Intro Partie

4.2.1 Scénario 1 : bla bla

Paragraphe 1 ('apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

4.2.2 Scénario 2 : bla bla

4.2.3 Scénario 3 : bla bla

4.3 Partie Réseaux

Intro Partie

4.3.1 Scénario 1 : bla bla

Paragraphe 1 (’apparaitra pas dans l’index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

4.3.2 Scénario 2 : bla bla

4.3.3 Scénario 3 : bla bla

4.4 Partie d'utilisation différentes

Intro Partie

4.4.1 Scénario 1 : bla bla

Paragraphe 1 ('apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

4.4.2 Scénario 2 : bla bla

4.4.3 Scénario 3 : bla bla

Chapitre 5

Bilan

Intro / Rappel Contexte
Nous avons donc pu en tirer la problématique suivante :

Problématique du sujet

Bla
Bla

Bla

Bla
Bla

Bla
Bla

Bla
Bla

Bla

5.1 Annexes

Annexes

Annexe 1

Intro

1 Partie 1

Bla

1.1 Sous-partie 1

Bla

1.2 Sous-partie 2

Bla

1.3 Sous-partie 3

Bla

2 Partie 2

Bla

2.1 Sous-partie 1

Bla

2.2 Sous-partie 2

Bla

2.3 Sous-partie 3

Bla

Annexe 2

Intro

Prérequis

Bla
— item1 ;
— item2 ;
— item3 ;
— item4.
Bla

1 Partie 1

Bla

1.1 Sous-parie 1

Bla

1.2 Sous-parie 2

Bla

2 Partie 2

Attention !
Texte d'avertissement

Bla

3 Partie 3

Bla

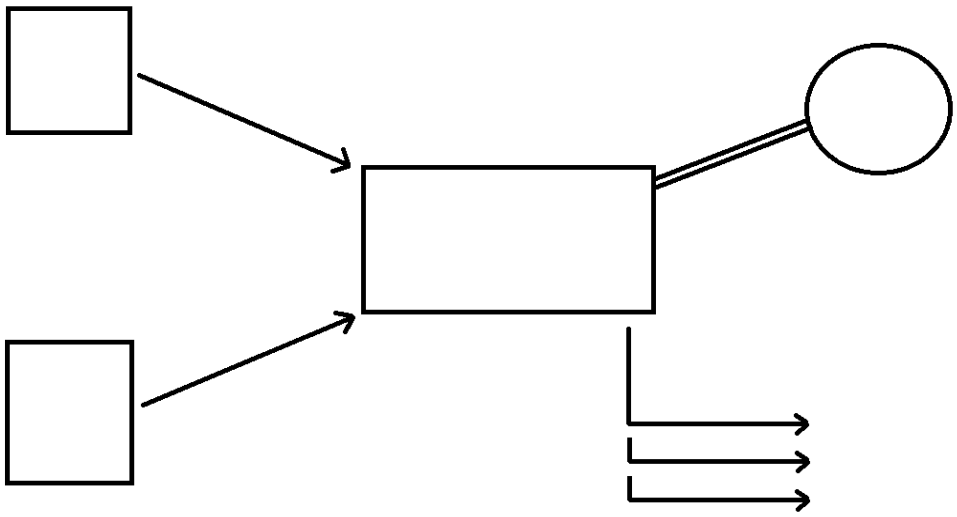

FIGURE 5.1 – Presentation schema

Paragraphe 1

Bla

Paragraphe 2

Bla

Paragraphe 3

Bla

Bibliographie

[1] Documentation Docker https://docs.docker.com/.

[2] Hyper-V Containers https://docs.microsoft.com/en-us/virtualization/windowscontainers.

[3] Linux Containers https://linuxcontainers.org/fr/.

[4] Virtual Box https://www.virtualbox.org/.

[5] Proxmox VE https://www.proxmox.com/en/.

[6] VmWare http://www.vmware.com/fr.html.

[7] QEMU http://www.qemu-project.org/.

[8] Hyper-V https://www.microsoft.com/fr-fr/cloud-platform/server-virtualization.

[9] KVM https://www.linux-kvm.org/.

[10] VM Mark https://www.vmmark.com/products/vmark.html.

[11] VM history par IBM http://www.vm.ibm.com/history/.

[12] An Analysis of Performance Interference Effects in Virtual Environments http://ieeexplore.ieee.org/document/4211036/?arnumber=4211036

[13] The impact of Docker containers on the performance of genomic pipelines https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4586803/

[14] WeiHuang,JiuxingLiu,BulentAbalietDhabaleswarK.Panda,ńACaseforHighPerformanceComputingwithVirtualMac ICS'06,ACM,sérieICS'06,2006,p.125134(ISBN1-59593-282-8,DOI10.1145/1183401.1183421

[15] PiotrLuszczek,EricMeek,ShirleyMooreetDanTerpstra,EvaluationoftheHPCChallengeBenchmarksinVirtualized SpringerBerlinHeidelberg,coll.ńLectureNotesinComputerScienceż,29août2011

[16] GitHub documentation de Phoronix test suite. https://gist.github.com/anshula/728a76297e4a4ee7688d#the-easy-way.

[17] Site officiel de SaltStack https://saltstack.com/.

[18] Documentation de SaltStack. https://docs.saltstack.com/en/latest/.

[19] https://docs.saltstack.com/en/latest/topics/installation/debian.html.

[20] https://www.phoronix-test-suite.com/documentation/phoromatic.html.