REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT
SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE

En vue de l'obtention d'un Master Académique en informatique

Option : Réseau Mobilité et Système Embarqué

Thème

Allocation de ressource dans le cloud computing

Dirigée par :

Mme Aoudjit R.

Réalisée par :

AMMOUR Lyza.

CHOUGGAR Chanez.

Année universitaire : 2016 /2017

Remerciement

Remerciements :

En terminant notre mémoire de fin d'étude, il nous est agréable d'adresser nos vifs remerciements à tous ceux qui nous ont aidés de près ou de loin à élaborer cet ouvrage.

D'abord nous tenons à remercier ***DIEU*** le tout puissant de nous avoir donné la santé, le courage et la foi pour réaliser ce modeste travail avec volonté.

Nous remercions en particulier notre promotrice Mme ***Aoudjit*** pour ses conseils précieux, ses critiques constructives et surtout pour les nuits blanches qu'elle faisait juste pour nous aider et sans oublier sa patience avec nous sa disponibilité et sa modestie aussi.

Nous remercions aussi notre prof adoré Mme ***Oukfif*** qui nous a accueillis à tout moment comme si on était ses binômes.

Nous remercions les ***membres de jury*** d'avoir accepté de juger notre modeste travail.

Ainsi que tous ***nos professeurs*** qui nous ont enseignés durant nos études à la faculté.

En fin nous tenons à remercier tous ***nos collègues d'études***, particulièrement notre promotion.

Dédicaces

Dédicaces

Je dédie ce travail à

Mon Cher papa (A.Y),

Ma très chère maman,

Mes très chers grands-parents,

Mes oncles et tantes, leurs enfants

Et tous mes amis.

Dédicace

J'ai la joie et le plaisir de dédier ce modeste travail à :

*A mes très chers parents « **HOCINE** » et « **ZEDJIGA** »pour leur conseils et leur encouragements.*

*A Mes chères sœurettes que j'adore sans limite « **SARAH**» et « **MELISSA** » qui m'ont toujours soutenue malgré leur distance.*

*A Mes très chers frères « **SOFIANE** » et « **YACINE** ».*

*A tous mes tantes, ainsi mes cousines « **Lynda** », « **Lydia** » et « **Razika** ».*

*A Ma **grands-mères FETTA** que dieu la garde pour moi et à la mémoire de **ma grand-mère paternelle** et **mes deux grands-pères** allah yarhemhoum inchallah.*

*Et sans oublier ma meilleure amie **Rymoucha** et ses sœurs **Hinan** et **swara** pour leur aide et leur soutien, **Mr.MPI** et sa fiancé **thalmanith** hhh que je tiens à leur souhaiter un « heureux mariage » et un « joyeux anniversaire » pour ce 4,**Djimi**,**Amelia**, **Katouche**, **Nedjma, Soso**, **Dyhoucha**, et aussi mes **camarades de classe** sans oublier mes deux profs adorées « **madame Oukfif** et **madame Aoudjit** » que j'adooore énormément. Malgré c'est la fin avec tout le monde je tiens à leur remercier pour cette belle année et je souhaite que du bonheur et une bonne continuation pour tout le monde.*

*Enfin je remercie énormément **les gardiens du département** qui m'ont aidé souvent à chercher les profs hhh surtout **Da remdhan** qui me ramenait souvent quoi manger et pour ses services consécutifs.*

THANMIRTH

Lezlouza

Sommaire

SOMMAIRE

SOMMAIRE

SOMMAIRE

SOMMAIRE

Liste des figures

LISTE DES FIGURES

Introduction générale

Introduction générale :

Le cloud computing (informatique dans les nuages) est une notion propagée ces dernières années dans le monde de l'informatique et s'est imposé comme un paradigme majeur d'utilisation des ressources informatiques. Ces ressources mises à disposition par le fournisseur cloud sont accessibles via un réseau informatique sous forme de services.

Avec le développement du cloud computing, l'échelle des centres de données continue d'être étendue, et un centre de données en nuage utilise habituellement des milliers de machines physiques. La technologie de virtualisation permet à une machine physique de supporter plusieurs machines virtuelles et chaque VM exécute différents types de services et applications qui sont des requêtes de ressources qui parviennent par les utilisateurs.

Avec la popularité du cloud computing, la consommation d'énergie des centres de données en nuage augmente de plus en plus. Selon l'estimation d'Amazon [34], les coûts liés à l'énergie des centres de données en nuage représentent 42% des coûts d'exploitation totaux. En outre, l'augmentation de la consommation d'énergie a entraîné une augmentation spectaculaire des émissions de CO2 et a affecté directement notre environnement. Par conséquent, la réduction de la consommation d'énergie est un problème urgent qui doit être résolu car elle réduira non seulement les coûts de l'énergie mais aussi la durabilité environnementale.

Une des solutions à ce problème est l'allocation de machines virtuelles dans des machines physiques appropriées. Le placement raisonnable est un facteur clé dans le bon fonctionnement et l'économie d'énergie des centres de données en nuage et est devenu progressivement un axe de recherche ces dernières années.

La plupart des études existantes ne tiennent pas compte de tous les objectifs, mais seulement en examinant certains. Néanmoins, ces objectifs doivent être considérés simultanément dans de nombreux scénarios.

Dans ce travail, nous proposons un nouveau modèle de placement de machines virtuelles à plusieurs objectifs en minimisant la consommation d'énergie et en essayant d'équilibrer la charge des serveurs à l'aide d'une métaheuristique. Notre méthode se focalise sur les algorithmes génétiques multiobjectifs.

Ce mémoire est subdivisé en quatre principaux chapitres , dans le premier chapitre intitulé « généralisés sur le cloud computing » nous présentons le cloud computing , son évolution , ses caractéristiques , ses modèles de déploiements et de services ; nous ferons ensuite le pas vers la virtualisation et quelques notions qui y ont une relation ; dans le deuxième chapitre qui a pour titre « l'allocation de ressource dans le cloud », nous définirons d'une manière détaillée le concept d'allocation de ressource dans le cloud computing ; nous passerons ensuite à la présentation de nos objectifs qui sont la minimisation de la consommation énergétique et la charge de travail . Au niveau du troisième chapitre intitulé « Analyses et conception », nous ferons une analyse de différents diagrammes issus de plusieurs exécutions de notre programme basés sur un ensemble de fonctions objectifs. Pour finir avec un dernier chapitre qui résume l'ensemble des outils et langages que nous avons utilisés tout au long de notre travail.

Chapitre I :

Généralité sur le cloud computing

Introduction :

L'informatique est encore en train de se transformer, elle se dématérialise et devient cloud computing. La puissance informatique devient ainsi virtuelle et se consomme aux besoins des clients et devient extensible.

Nous verrons dans les prochaines sections de ce chapitre la définition du cloud computing, son évolution, ses caractéristiques et ses modèles de déploiements et de services, nous parlerons également de la virtualisation dans le cloud, de son fonctionnement et des domaines de son utilisation, ainsi que quelques notions qui y ont une relation.

I. Définitions et concepts du Cloud Computing :

I.1 Cloud computing vue d'ensemble : [1]

Cloud Computing est un moyen par lequel nous pouvons accéder aux applications en tant qu'utilitaires, sur Internet. Il nous permet de créer, configurer et personnaliser des applications en ligne.

I.2 Définition du cloud : [2]

Le *Cloud Computing* ou l'informatique en nuage est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet. Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. Le cloud computing se caractérise par sa grande souplesse : selon le niveau de compétence de l'utilisateur client, il est possible de gérer soi-même son serveur ou de se contenter d'utiliser des applicatifs distants en mode SaaS. Selon la définition du National Institute of Standards and Technology (NIST), le cloud computing est l'accès via un réseau de télécommunications, à la demande et en libre-service, à des ressources informatiques partagées configurables. Il s'agit donc d'une délocalisation de l'infrastructure informatique.

Le cloud computing est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs (machines) distants auxquels les usagers se connectent via une liaison Internet sécurisée cela est illustrée dans la **Figure I.1**.

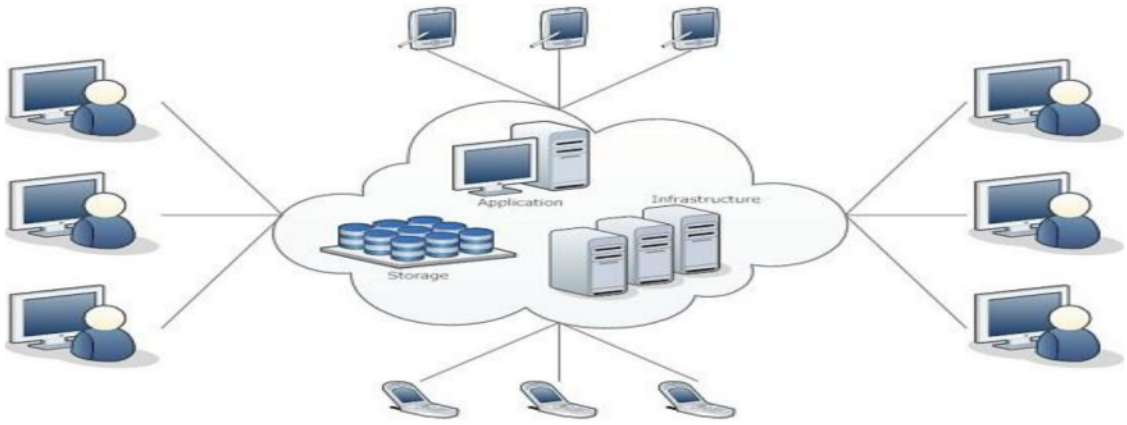


Figure I.1 : Infrastructure globale du cloud.

I.3 L'évolution du cloud computing : [3]

L'ère de l'information a connu de nombreuses évolutions. Nous avons débuté avec les mainframes, et évolué vers les mini-ordinateurs, puis sont venus les ordinateurs personnels jusqu'à atteindre le cloud computing. La figure ci-dessous illustre les différentes technologies dans l'ère de l'information:

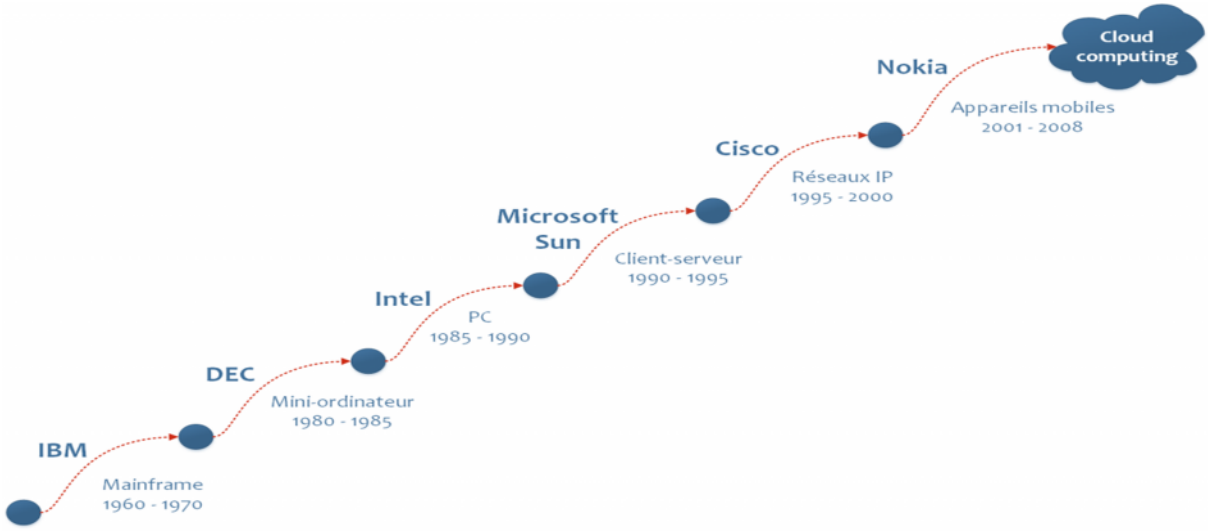


Figure I.2 : les différentes technologies dans l'ère de l'information.

Le cloud computing est une technologie qui n'est pas tout à fait récente comme on pourrait le penser; ses traces remontent aux années soixante, à cette époque les utilisateurs accédaient depuis leurs terminaux à des applications fonctionnant sur des systèmes centraux (mainframes) qui correspondaient aux serveurs du cloud. Cette idée continue d'évoluer dans l'ère du temps, elle prend naissance en 1990 et surtout en 1991 avec la naissance d'Internet et la mise sur le marché du logiciel *CERN* qui a été le premier logiciel accessible par le Web. Cette idée continue de progresser avec la découverte d'*EBay* et d'*Amazon* en 1995. Le processus dévolution s'accélère ensuite, et en 2000, l'arrivée de *Google* fournit une réelle émergence du cloud computing.

I.4 Les modèles de déploiement du cloud computing : [4]

Les modèles de déploiement définissent le type d'accès au cloud, il peut avoir l'un des quatre types d'accès suivants :

I.4.1 Le cloud public :

Le cloud public permet aux systèmes et aux services d'être facilement accessibles par tous les utilisateurs; il est peut-être moins sécurisé en raison de son ouverture, exemple : le courrier électronique.

Dans un cloud public, l'environnement est entièrement détenu par la société qui met à disposition ses services cloud. Le fournisseur conçoit, gère, maintient et fait évoluer les ressources en

fonction des besoins des clients au fil du temps. La sécurité est donc de la responsabilité du fournisseur qui en assure la gestion.

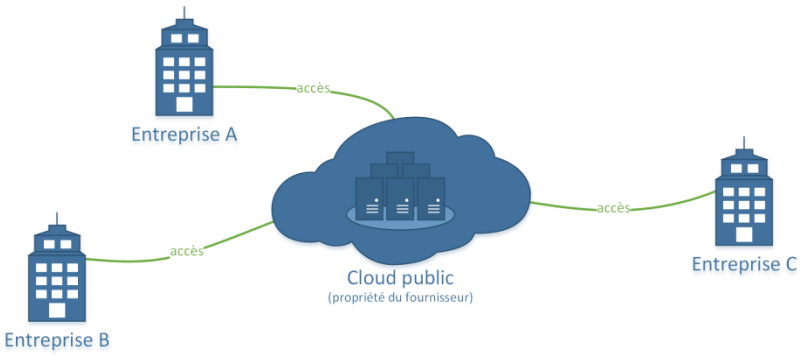


Figure I.3 : Modèle de déploiement d'un cloud public.

I.4.2 Le cloud privé :

Les réseaux, serveurs, et infrastructures de stockage associés aux clouds privés sont dédiés à une seule entreprise et ne sont pas partagés avec d'autres. Le cloud est entièrement contrôlé par l'entreprise elle-même, par conséquent les risques de sécurité associés à un cloud privé sont minimisés. Ce haut degré de contrôle et de transparence permet au propriétaire d'un cloud privé de se conformer plus facilement à des normes, politiques de sécurités ou conformités réglementaires qui peuvent être requises dans certains domaines.

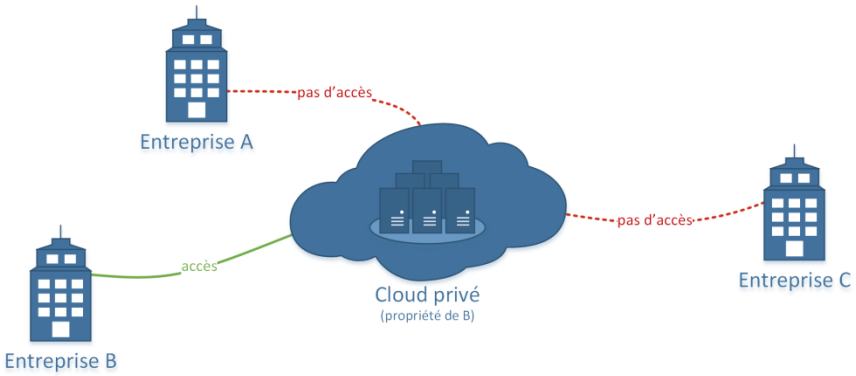


Figure I.4 : Modèle de déploiement d'un cloud privé.

I.4.3 Le cloud communautaire :

Le cloud de type communautaire est un modèle de déploiement partagé entre plusieurs entreprises géré, et sécurisé par l'ensemble des participants ou par un fournisseur de service.

L'infrastructure commune est spécifiquement conçue pour répondre aux exigences de la communauté ; à titre d'exemple, des organismes gouvernementaux, des hôpitaux ou des entreprises de télécommunication qui auraient des contraintes de réseau, de sécurité, de stockage, de calcul ou

d'automatisation similaires pourraient trouver des intérêts communs à déployer collectivement un cloud communautaire.

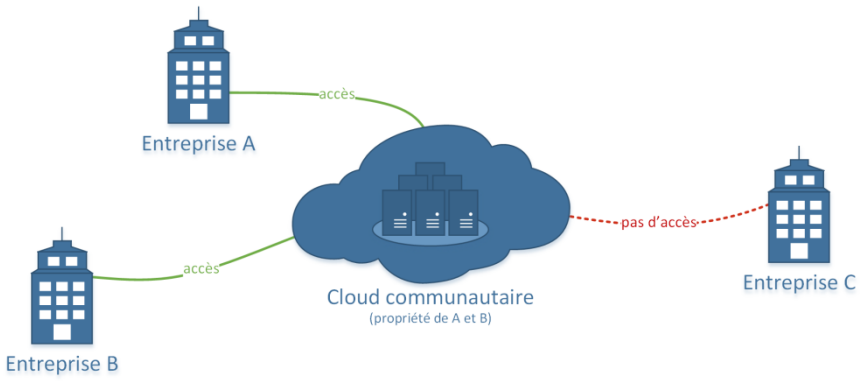


Figure I.5 : Modèle de déploiement d'un cloud communautaire.

I.4.4 Le cloud hybride :

Le cloud hybride est la combinaison de plusieurs modèles de déploiement de clouds. Les différents clouds qui la composent restent des entités indépendantes à part entière, mais sont reliés entre elles par une technologie normalisée ou propriétaire qui permet la portabilité des données et des applications. Elle a la possibilité de privilégier l'utilisation de son cloud privé tout en gardant la possibilité de déborder sur une offre de cloud public en cas de besoin temporaire.

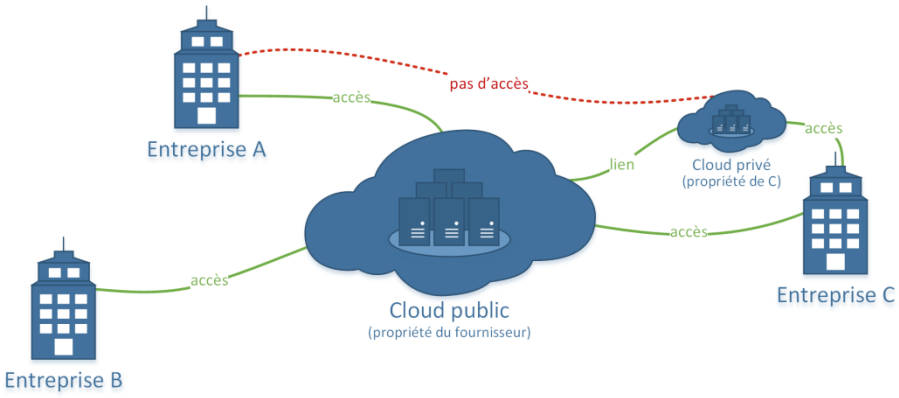


Figure I.6 : Modèle de déploiement d'un cloud hybride.

I.5 Les modèles de services de cloud computing : [5]

Les modèles de service sont les modèles de référence sur lesquels le Cloud Computing est basée. Ceux-ci peuvent être catégorisés en trois modèles de services de base, comme indiqué ci-dessous:

- Infrastructure en tant que service (IaaS)
- Plate-forme en tant que service (PaaS)
- Logiciel en tant que service (SaaS)

Il existe de nombreux autres modèles de service qui peuvent tous prendre la forme comme **XaaS,** à savoir, **Tout comme un service.** Cela peut être du **réseau** en **tant que service, affaires** en **tant que service, identité** en **tant que service, base** de **données** en **tant que service** ou **stratégie** en **tant que service.**

L'infrastructure en tant que service **(IaaS)** est le plus haut niveau de service de base. Chacun des modèles de service utilise le modèle de service sous - jacent, à savoir, chacun hérite du mécanisme de sécurité et de gestion du modèle sous - jacent, comme le montre le schéma suivant:

Figure I.7: les différentes couches des services du cloud computing

I.5.1 IaaS (Infrastructure as a service) :

IaaS fournit l'accès aux ressources fondamentales telles que les machines physiques, machines virtuelles, le stockage virtuel, etc. ils fournissent aux utilisateurs une capacité de traitement, de stockage, de bande passante, de réseaux, de serveurs virtuels et d'autres ressources de calcul.

L'utilisateur n'a aucun contrôle sur l'infrastructure sous-jacente du provider de Cloud mais il a le contrôle sur les systèmes d'exploitation, le stockage et les applications hébergées sur les serveurs virtuels.

Le déploiement sur IaaS nécessite des compétences sur l'administration de systèmes d'exploitation et de serveurs. L'IaaS s'adresse donc à des administrateurs et non pas à des développeurs.

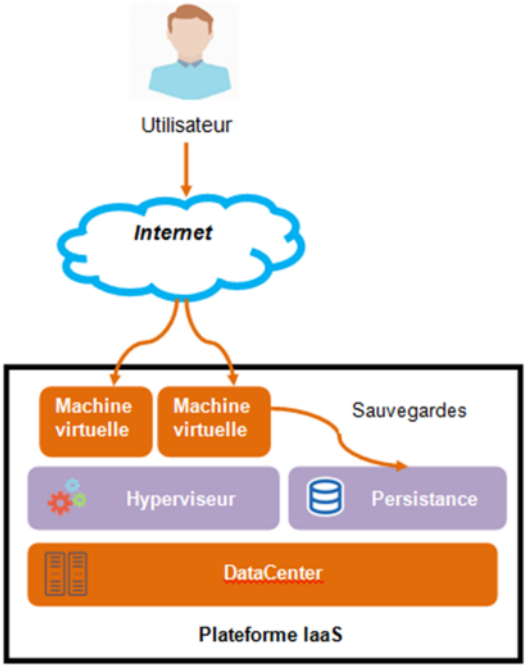


Figure I.8 : les infrastructures IaaS.

I.5.2 PaaS (Platform as a service) :

PaaS fournit l'environnement d'exécution pour les applications, les outils de développement et de déploiement, etc.

Cette plateforme peut être utilisée pour exécuter des sites web, des SaaS ou tout développement issu de l'entreprise. Ces développements spécifiques doivent respecter le langage de développement et l'architecture de la plateforme PaaS c'est-à-dire les applications déployées sur l'infrastructure du provider sont basées sur des langages de programmation et des outils supportés par ce dernier.

Les développeurs utilisent les outils du fournisseur pour créer leurs propres applications, ils peuvent souvent créer des applications web sans avoir à installer le moindre outil sur leur poste de travail. Ils ont ensuite la possibilité de déployer leurs applications sans pour autant avoir besoin de disposer de compétences en administration système ; cela permet aux développeurs et aux petites entreprises de déployer des applications web sans devoir supporter la complexité et les coûts engendrés par l'achat et la configuration de serveurs.

L'utilisateur n'a aucun contrôle sur l'infrastructure sous-jacente du provider de Cloud (Serveurs, Stockage, systèmes d'exploitation, réseaux, etc.) mais il a le contrôle sur les applications déployées et l'environnement d'hébergement (paramétrage des services Web, structure des bases de données, nombre de serveurs…).

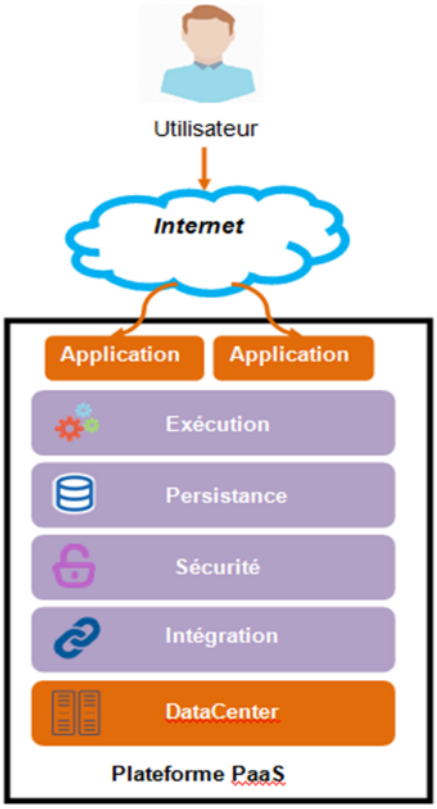


Figure I.9 : Les plateformes PaaS.

I.5.3 SaaS (Software as a service): [6] [7] [8]

Le modèle SaaS fournit un logiciel sous forme de service. La différence entre SaaS et logiciel est essentielle. En effet, les SaaS proposent des logiciels opérationnels, prêt à l'emploi, sans passer par une étape d'installation et sans aucune tâche de maintenance. Concrètement, lorsqu'un client sollicite une solution SaaS, il se connecte simplement au service depuis un navigateur. Les SaaSs s'adressent donc aux utilisateurs finaux.

- ***Fonctionnement :***

En utilisant le mode SaaS**,** l'entreprise n'hébergera pas ses applications et ne stockera pas ses données en interne. Il n'est donc pas nécessaire d'acquérir directement ces applications et posséder des serveurs pour les héberger. De plus, la maintenance et les mises à jour des applications seront gérées en externe par le prestataire.

Les SaaS sont exécutés sur des plateformes conçues pour une utilisation simultanée par un grand nombre de collaborateurs qui travaillent dans de nombreuses entreprises différentes. Ces plateformes sont mises à disposition par des acteurs (comme Google ou SalesForce).

Les utilisateurs n'ont aucun contrôle sur l'infrastructure du Cloud que ce soit sur les aspects techniques (Datacenter, réseau, serveurs, stockage, etc.) ou sur les aspects fonctionnels (version de l'application, fonctions disponibles, etc.).

Les utilisateurs de l'entreprise devront simplement disposer d'un ordinateur et des codes d'accès au service en ligne pour pouvoir travailler**.**

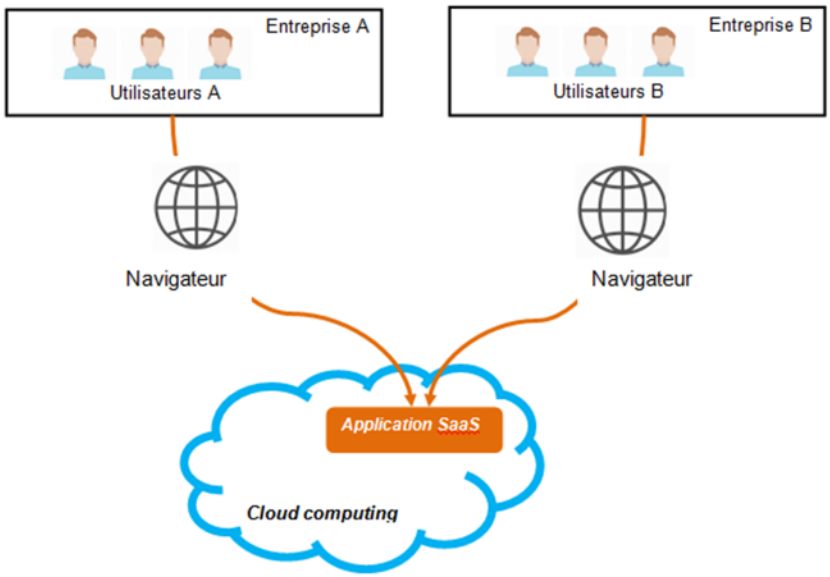


Figure I.10: Vision générale des SaaS.

I.6 Caractéristiques du cloud computing : [9] [6] [10]

Il existe cinq caractéristiques clés du cloud computing. Ils sont affichés dans le schéma suivant:

1. ***Self-service, à la demande*** : Capacité à fournir une ressource automatiquement, sans requérir d'interaction humaine coté fournisseur.

Cloud Computing permet aux utilisateurs d'utiliser les services Web et les ressources sur demande. On peut se connecter à un site Web à tout moment et les utiliser.

2. ***Bande passante élevée et accès réseau universel*** : les ressources mises à disposition des utilisateurs sont accessibles via Internet de n'importe quel endroit et en utilisant des plateformes hétérogènes (PC, téléphone, mobile, PDA, ordinateur portable, etc.) grâce à la capacité actuelle des réseaux.

3. ***Ressource partagées, mises en commun (pooling)*** : Cloud Computing permet à plusieurs locataires de partager un pool de ressources. On peut partager une instance physique, de base de données et d'infrastructure de base.

4. ***Elasticité*** : possibilité de faire évoluer très rapidement la capacité fournie, que ce soit en plus ou en moins.

5. ***Mesurable*** : capacité à mesurer le service fourni. L'utilisation des ressources peut être surveillée, contrôlée et signalée, offrant la transparence pour le fournisseur et le consommateur du service utilisé. Cette caractéristique permet en particulier de payer le service à l'usage « pay as you go ».

Le pay as you go : les utilisateurs paient les ressources qu'ils utilisent en fonction de leur consommation réelle et précise.

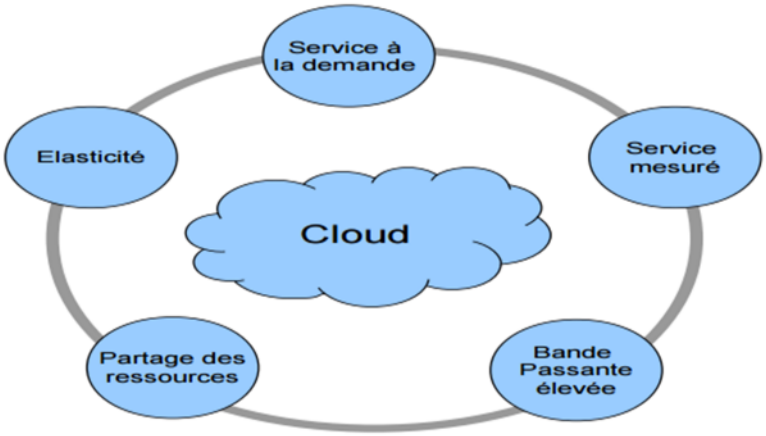


Figure I.11: Les cinq caractéristiques du Cloud Computing.

II. Les technologies du cloud computing : [1]

Il existe certaines technologies qui fonctionnent derrière les plateformes informatiques en nuage rendant le cloud computing flexible, fiable et utilisable. Ces technologies sont listées ci-dessous:

- Virtualisation.
- Architecture orientée service (SOA).
- Calcul en grille.
- Informatique utilitaire.

II.1 Virtualisation :

La virtualisation se définit comme l'ensemble des techniques matérielles et/ou logiciels qui permettent de faire fonctionner sur une seule machine physique différents systèmes d'exploitation. Séparés les uns des autres, l'utilisation de ces différents OS donne à l'utilisateur l'illusion d'être en présence de plusieurs machines distinctes.

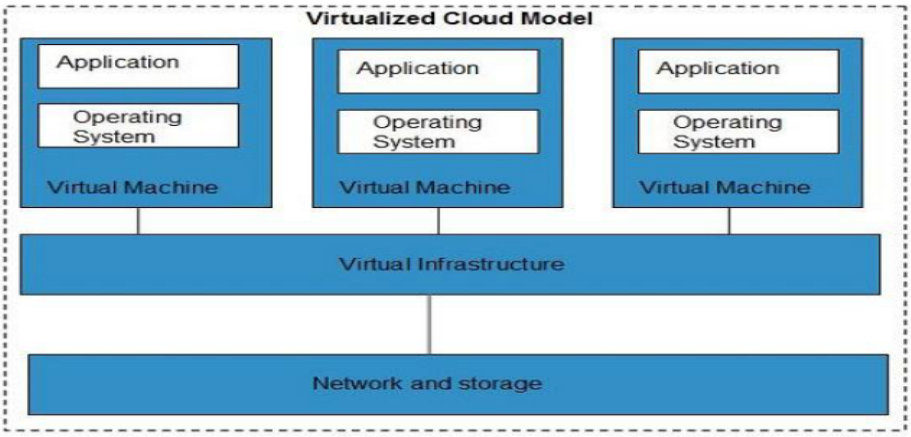


Figure I.12 : Model de nuage virtuel.

L'architecture Multi-tenant offre une isolation virtuelle parmi les locataires multiples par conséquent, les organisations peuvent utiliser et personnaliser l'application comme si elles avaient chacune leur propre instance.

II.2 Architecture Orienté Services (SOA) :

L'architecture orientée service permet d'utiliser les applications comme un service pour d'autres applications quel que soit le type de fournisseur, le produit ou la technologie. Par conséquent, il est possible d'échanger des données entre les applications de différents fournisseurs sans programmation supplémentaire ou d'apporter des modifications aux services. Architecture orientée vers le service de cloud computing.

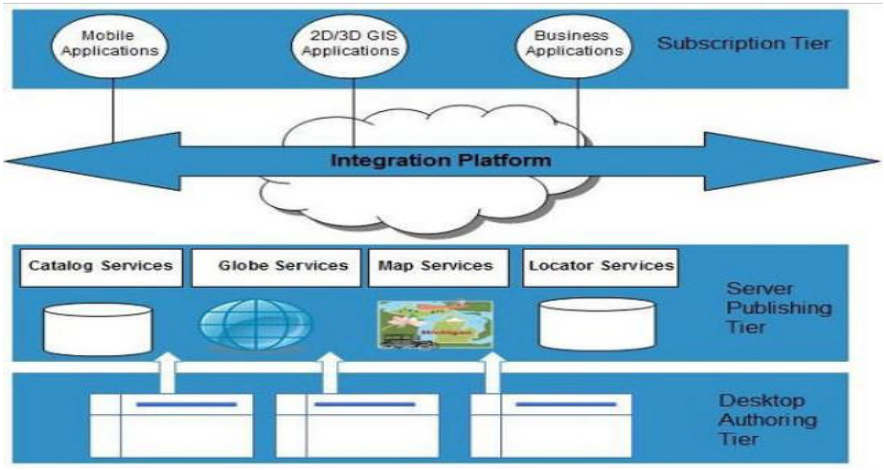


Figure I.13 : environnement SOA.

II.3 Calcul en grille :

Grid Computing se réfère à l'informatique distribuée dans laquelle un groupe d'ordinateurs de plusieurs emplacements sont reliés entre eux pour atteindre un objectif commun. Ces ressources informatiques sont hétérogènes et géographiquement dispersées. Grid Computing brise la tâche

complexe en petits morceaux. Ces pièces plus petites sont distribuées aux CPU qui résident dans la grille.

Figure I.14 : grid computing.

II.4 Informatique utilitaire :

L'informatique utilitaire est basée sur le modèle Pay per Use. Il offre des ressources informatiques à la demande en tant que service aux clients. Le cloud computing, le grid computing et les services informatiques gérés sont basés sur le concept de l'informatique utilitaire.

III. L'architecture du cloud computing : [1]

L'architecture du cloud computing se réfère aux composants et sous-composants requis pour le cloud computing. Ils se composent généralement d'une plate-forme frontale ou front-end (gros client, client léger, appareil mobile), de plate-formes de back-end (serveurs, stockage), d'une livraison basée sur un cloud et d'un réseau (Internet, Intranet, Intercloud). Combinés, ces composants forment l'architecture du cloud computing.

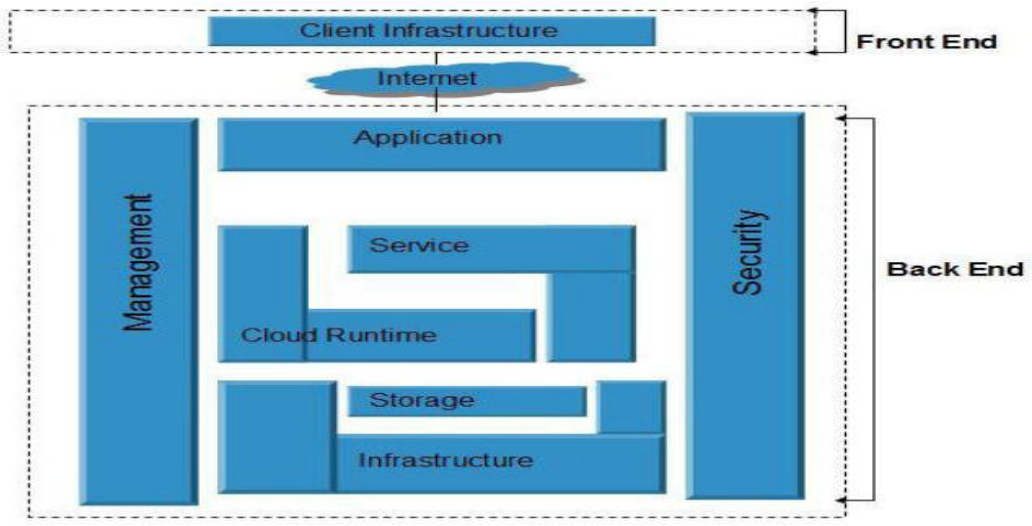


Figure I.15: Architecture du cloud computing.

III.1 L'extrémité avant (front-end) :

Front End fait référence à la partie client du système de cloud computing. Il se compose d'interfaces et d'applications requises pour accéder aux plateformes de cloud computing, par exemple : Navigateur Web.

III.2 L'extrémité arrière (back-end) :

Back-End fait référence au cloud lui-même. Il se compose de toutes les ressources requises pour fournir des services de cloud computing ; d'un vaste stockage de données, de machines virtuelles, de mécanismes de sécurité, de services, de modèles de déploiement, de serveurs, etc.

Il incombe à l'arrière-plan de fournir des mécanismes de sécurité intégrés, des contrôles de trafic et des protocoles. Le serveur utilise certains protocoles, connus sous le nom de middleware, qui aide les périphériques connectés à communiquer les uns avec les autres.

IV. Les composants de l'infrastructure du cloud computing : [1]

L'infrastructure cloud se compose de serveurs, de stockage, de réseaux, de logiciels de gestion, de logiciels de déploiement et de virtualisation de plateforme.

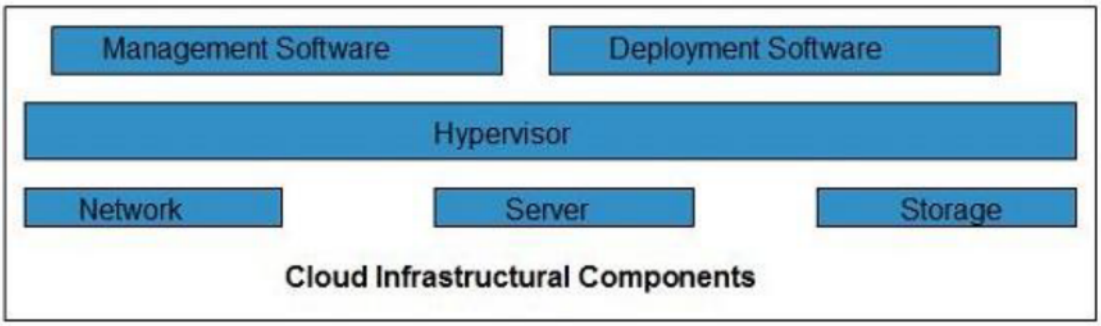


Figure I.16 : Composants d'infrastructure en nuage.

IV.1 Hyperviseur :

L'hyperviseur est un programme de micro logiciel ou de bas niveau qui agit comme un Virtual Machine Manager. Il permet de partager l'instance physique unique de ressources cloud entre plusieurs locataires. Nous le détaillerons plus loin.

IV.2 Logiciel de gestion :

Le logiciel de gestion permet de maintenir et de configurer l'infrastructure de cloud.

IV.3 Logiciel de déploiement :

Le logiciel de déploiement aide à déployer et à intégrer l'application sur le cloud.

IV.4 Réseau :

Le réseau est le composant clé de l'infrastructure cloud. Il permet de connecter des services cloud sur Internet. Il est également possible de fournir du réseau comme un utilitaire sur Internet, autrement dit, le consommateur peut personnaliser l'itinéraire réseau et le protocole.

IV.5 Serveur :

Le serveur permet de calculer le partage des ressources et d'offrir d'autres services tels que l'allocation des ressources et la désaffectation, le suivi des ressources, la sécurité, etc.

IV.6 Espace de stockage :

Cloud utilise un système de fichiers distribué à des fins de stockage. Si l'une des ressources de stockage échoue, elle peut être extraite d'une autre qui rend le cloud computing plus fiable.

V. La virtualisation dans le cloud computing : [11]

V.1 Définition :

La construction d'une architecture Cloud doit répondre à une exigence d'élasticité, ce qui suppose une allocation dynamique des ressources et théoriquement illimitées. De plus, cette architecture doit pouvoir gérer une montée en charge illimitée dans un environnement multiserveurs avec un nombre de serveurs extrêmement variable et qui doit pouvoir augmenter ou diminuer de manière instantanée .

Pour cela, plusieurs études ont été proposées dans la littérature pour déterminer le nombre de serveurs à utiliser de façon à satisfaire des critères de qualités de service inscrits dans le SLA.

Par ailleurs, l'existence de plusieurs fournisseurs rend l'interopérabilité encore plus difficile. La construction d'une architecture Cloud est donc une problématique éminemment technique et se base essentiellement sur deux notions, à savoir la virtualisation et l'Architecture Orientée Service (SOA).

La virtualisation est un concept qui s'est imposé depuis la deuxième moitié de la décennie 2000 à 2010 et qui a rendu crédible, au même titre qu'Internet, la notion même du Cloud. En effet, la virtualisation est totalement indissociable de la notion d'élasticité et de montée en charge.

La virtualisation est un concept qui permet de subdiviser un serveur en plusieurs machines virtuelles qui opèrent indépendamment et en se partageant les ressources disponibles sur le serveur. L'utilisation des ressources disponibles sur le serveur sont optimisées via un composant appelé hyperviseur.

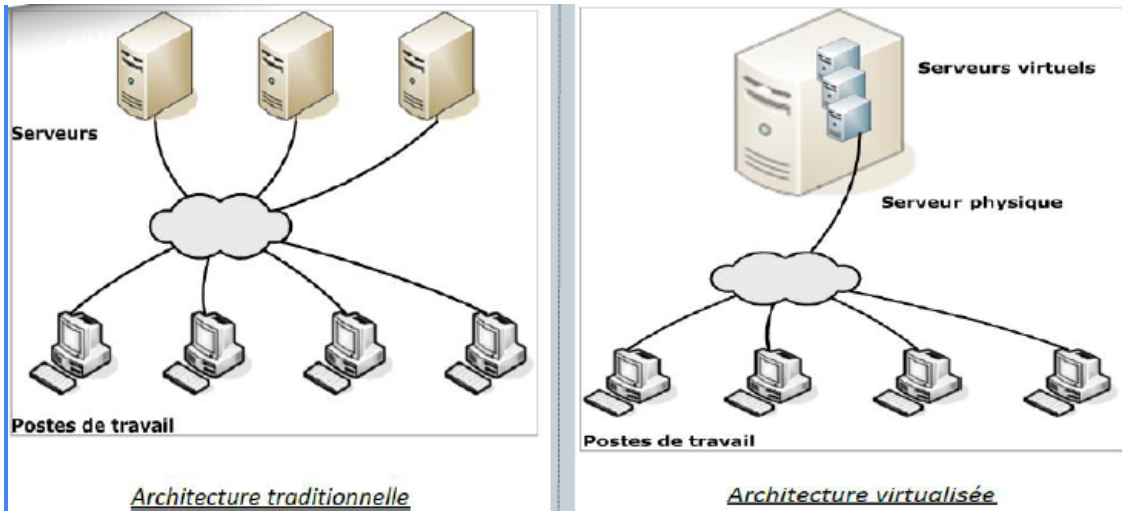


Figure I.17 : différence entre l'architecture traditionnelle et l'architecture virtualisé.

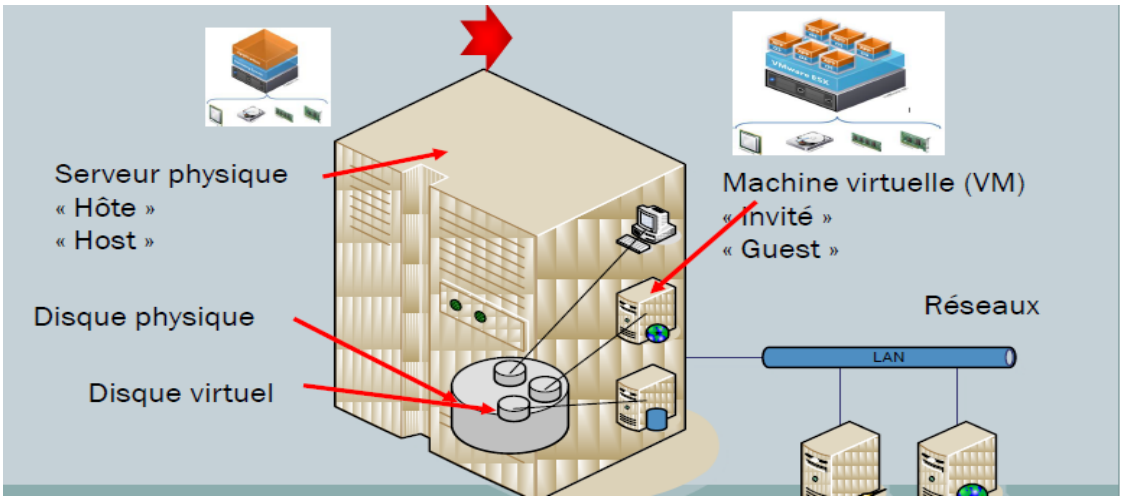


Figure I.18 : virtualisation au sein d'un seul serveur.

V.2 Hyperviseur : [12]

L'hyperviseur est également connu sous le nom de Virtual Machine Monitor (VMM), il peut être un logiciel, un micro logiciel ou un matériel qui donne allusion aux ordinateurs invités (machines virtuelles) qu'ils fonctionnent sur un matériel physique. L'hyperviseur gère les requêtes des machines virtuelles pour accéder aux ressources matérielles (RAM, CPU, NIC, etc.) agissant comme une machine indépendante. Deux types d'hyperviseur sont connus :

- Hyperviseur de type 1 appelé Native ou Bare Metal
- Hyperviseur de Type 2 appelé Hyperviseur hébergé

a- Hyperviseur de type 1 :

Un hyperviseur de type 1 est un système qui s'installe directement sur la couche matérielle du serveur. Ces systèmes sont allégés de manière à se concentrer sur la gestion des systèmes d'exploitation invités. Ceci permet de libérer le plus de ressources possible pour les machines

virtuelles. Toutefois, il est possible d'exécuter uniquement un hyperviseur à la fois sur un serveur. Parmi les hyperviseurs de type 1 on trouve des systèmes comme Xen, VMware ESX et Proxmox.

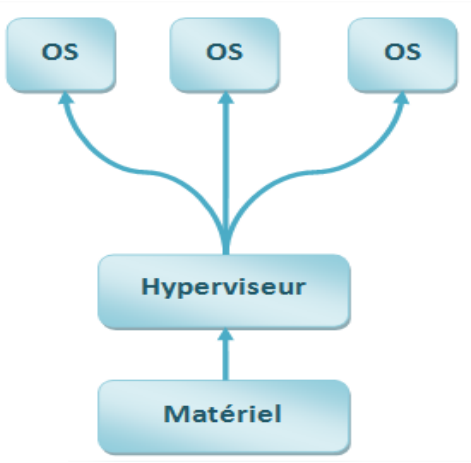


Figure I.19 : représentation du placement d'un hyperviseur de type 1.

b- Hyperviseur de type 2 :

Un hyperviseur de type 2 est un logiciel qui s'installe et s'exécute sur un système d'exploitation déjà en place. De ce fait, plus de ressources sont utilisées étant donné qu'on fait tourner l'hyperviseur et le système d'exploitation qui le supporte, il y a donc moins de ressources disponible pour les machines virtuelles. L'intérêt qu'on peut trouver c'est le fait de pouvoir exécuter plusieurs hyperviseurs simultanément vu qu'ils ne sont pas liés à la couche matérielle. Parmi les hyperviseurs de type 2, on trouve VMware Player, VMware Workstation, VirtualPC et VirtualBox.

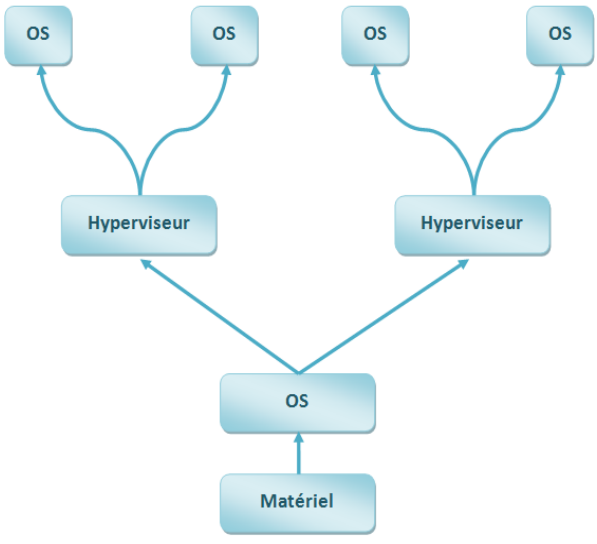


Figure I.20 : représentation du placement d'un hyperviseur de type 2.

- ***Fonctionnement :***

En plus d'être chargée de la création des VMs, la VMM (hyperviseur) a pour rôle de partager et de coordonner les accès aux ressources de l'hôte physique sous-jacent entre les OS invités (demandes clients). Bien qu'ils soient virtuels, une VM dispose d'un ensemble de périphériques. Une VM possède une puissance de calcul (nombre et capacité du CPU), un espace de stockage (en général représenté par des fichiers sur les systèmes de stockage), une quantité de mémoire et une ou plusieurs cartes réseaux. Au cours de son fonctionnement, tout accès de l'OS invité à un périphérique virtuel est géré par la VMM qui se charge d'interpréter et d'exécuter la demande sur le périphérique réel sollicité. Cette architecture et ce mécanisme de translation sont entièrement transparents à l'utilisateur final qui s'imagine disposer d'une machine physique réelle. Ceci est rendu possible grâce au respect des critères suivants établis par Popek et Goldberg pour les VMM :

- ***Equivalence :*** toute exécution d'application dans un système virtualisé doit être identique à une exécution sur une machine physique ; exception faite du temps d'exécution lié à la disponibilité des ressources,
- ***Efficacité :*** la majorité des instructions du système invité doit directement être exécutée par le processeur sans intervention du logiciel de virtualisation,
- ***Contrôle de ressources :*** l'ensemble des ressources est géré de façon exclusive par le logiciel de virtualisation ; Ainsi, les applications déployées dans les VMs sont exécutées de façon identique à une exécution sur un support physique et ne nécessitent aucune modification.

V.3 Domaines de la virtualisation : [13] [14]

Cette animation montre les niveaux de virtualisation des éléments clés d'un ordinateur sur le cloud computing : la virtualisation de serveur, la virtualisation de stockage, la virtualisation d'application et la virtualisation de réseau. Il est possible de mixer chacune de ces technologies pour obtenir une architecture la plus flexible qu'il soit.

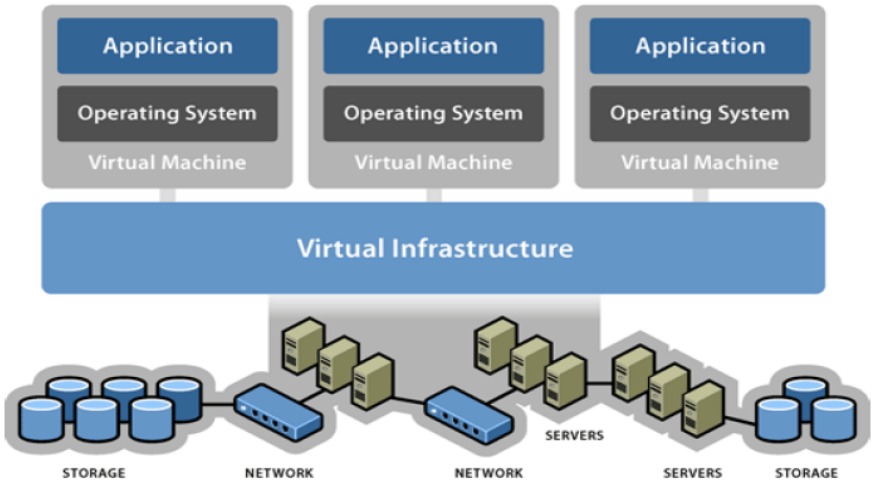


Figure I.21 : domaines de la virtualisation.

- ***Virtualisation du serveur :***

Pour mieux rentabiliser son parc informatique et utiliser plus efficacement les ressources des serveurs disponibles, il est intéressant de les "virtualiser". Pour cela, par un principe d'émulation, une couche logicielle isole les ressources physiques des systèmes d'exploitation. Ceux-ci s'exécutent alors sur des "machines virtuelles". Par ce principe plusieurs systèmes d'exploitation peuvent cohabiter sur une même machine, indépendamment l'un de l'autre

- ***Virtualisation du stockage :***

La virtualisation du stockage permet de masquer les spécificités physiques des unités de stockage. Elle forme une couche de virtualisation active et transparente entre les périphériques de stockage sur disque afin d'optimiser la disponibilité, les performances et l'utilisation des petits et grands Datacenters. L'ensemble intègre des fonctions gérées de manière centralisée de protection des données, de provisionnement, de mise en cache, de réplication et de migration.

- ***Virtualisation d'application :***

La virtualisation d'application permet de dissocier l'application du système d'exploitation hôte et des autres applications présentes afin d'éviter les conflits. Cette solution est particulièrement pratique pour simplifier l'administration du parc informatique notamment lors de l'installation des nouveaux releases.

- ***Virtualisation de réseau :***

La virtualisation réseau consiste alors à faire migrer l'ensemble des fonctions des équipements dans un logiciel : commutation, routage, VPN, etc.
Ainsi, chaque client cloud dispose de capacités réseau aux côtés des capacités serveur et stockage. Même si, le plus souvent, cela reste transparent pour lui. Ce réseau est dit « overlay », ce qui signifie qu'une couche logicielle se superpose au-dessus du matériel pour le piloter. Puis, des technologies de tunnel sont utilisées pour interconnecter ces réseaux virtuels: Vlan (limité à 4096 réseaux), et plus récemment VXLAN ou NVGRE.

V.4 Les méthodes de virtualisation : [15] [16]

La virtualisation doit s'adapter aux différentes briques technologiques d'une infrastructure. Il existe trois variantes d'architecture de virtualisation existent:

- La virtualisation complète
- La virtualisation émulation
- Paravirtualisation

V.4.1 La virtualisation complète :

La virtualisation complète, c'est un système d'exploitation qui exécute un logiciel nommé hyperviseur.

L'hyperviseur va permettre l'exécution de plusieurs machines virtuelles sur la machine physique. Il gère les accès mémoire, l'allocation du CPU et toutes les ressources nécessaires aux machines virtuelles.

Dans la virtualisation complète, l'hyperviseur gère l'ensemble des requêtes des machines virtuelles ce qui permet aux machines virtuelles de fonctionner sans aucune modification de leur noyau.

Autrement dit, les machines virtuelles ne savent pas qu'elles s'exécutent de manière virtuelle.

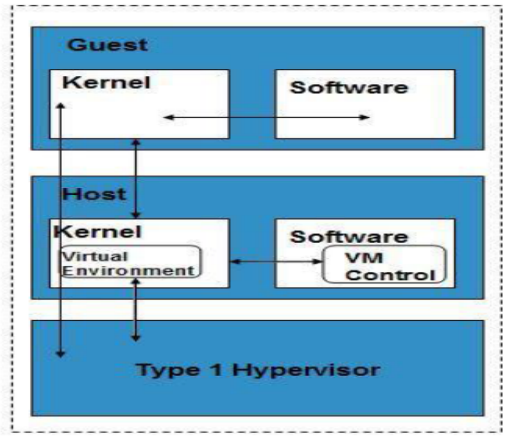


Figure I.22 : diagramme de la virtualisation complète.

V.4.2 Emulation :

Dans l'émulation**,** la machine virtuelle simule le matériel et devient ainsi indépendante de celui-ci. Dans ce cas, le système d'exploitation invité ne nécessite pas de modification.

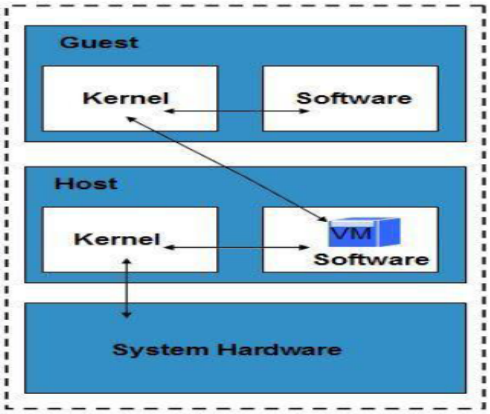


Figure I.23 : diagramme d'une émulation.

V.4.3 Paravirtualisation :

Le code de l'OS invité est modifié afin d'appeler directement l'hyperviseur

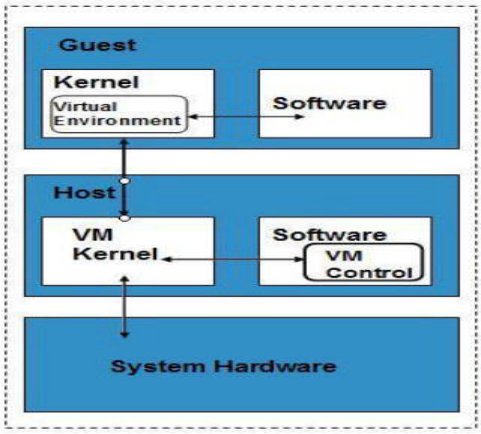


Figure I.24 : diagramme de la paravirtualisation.

VI. La migration des machines virtuelles : [17]

En virtualisation, la migration de machines virtuelles consiste à déplacer l'état d'une machine virtuelle, d'un hôte physique à un autre.

Il existe plusieurs raisons pour lesquelles il est nécessaire de migrer des systèmes d'exploitation, la plus importante étant l'équilibrage de la charge de travail sur les serveurs physiques. Il est également nécessaire de migrer des machines virtuelles lorsqu'un hôte physique est défectueux ou nécessite une maintenance.

Il faut également tenir compte du temps d'arrêt de la machine virtuelle lors de cette migration. La plus grosse charge de travail consistant à migrer la mémoire vive, il est nécessaire d'opter pour des stratégies de migration différentes en fonction des exigences.

VI.1 Technique des migrations des VMs : [17] [43]

Les techniques de la migration sont classifiées globalement en deux types : migration à chaud (live migration) et migration à froid (no0 live migration).

VI.1.1 Migration à froid :

Cette stratégie est la plus basique, il est nécessaire de mettre la machine virtuelle hors tension afin de copier l'état de sa mémoire sur l'hôte cible. Elle est utilisée avec la stratégie de répartition de charge dynamique lorsque l'hôte reçoit plus de requêtes qu'il ne sait en gérer. La décision de migration se fait sur base d'un calcul des requêtes entrantes, de l'utilisation du processeur, des ressources disponibles sur l'hôte physique ainsi que de la consommation énergétique.

L'avantage principal de cette méthode est qu'elle n'impliquera aucune faute lors de la migration de la mémoire. La machine étant arrêtée, les services ne seront plus disponibles et la mémoire ne sera pas altérée sur l'hôte source. La machine une fois migrée pourra reprendre son activité dans le même état que lorsqu'elle s'est arrêtée.

Elle comporte par contre un désavantage certain. À partir du moment où la machine est arrêtée, les services fonctionnant sous celle-ci ne seront plus disponibles durant toute la durée de la migration. Ce qui pose évidemment des problèmes majeurs pour des applications nécessitant une haute disponibilité. De plus, la mémoire étant transférée entièrement sur l'hôte cible, la migration à froid provoque un ralentissement sur tout le réseau du cloud.

VI.1.2 Migration à chaud :

Les systèmes à chaud (live migration) est un outil utile pour la migration des systèmes d'exploitation complets ainsi que leurs applications entre les machines physiques éloignées des data centers et des clusters c'est une technologie impressionnante qui facilite l'équilibrage de charge, la gestion des pannes, la maintenance du système et la réduction de la consommation d'énergie. La migration à chaud des machines virtuelle est le processus de transférer une ou plusieurs machines virtuelles d'une machine physique ou d'un espace de stockage vers un autre sans interrompre l'état de la VM. Cette technique est utile dans plusieurs scénarios, parmi lesquels nous avons :

- ✓ Une VM dans un nœud physique défaillant peut être migré vers un autre nœud non défaillant.
- ✓ Les VMs en mode repos sur une machine physique peuvent être migré vers d'autres hôtes pour optimiser l'utilisation des ressources.
- ✓ Les Vms sur un nœud physique chargé peuvent être migrés vers diffèrent nœud pour équilibrer les charges.

Les techniques de la migration à chaud sont caractérisées par 2 types basés sur le processus de la copie mémoire. Ces techniques sont : « post-copy » et « pre-copy ».

a. Live migration « post-copy » :

Dans cette technique, la VM est suspendue dans la machine source et son état d'exécution (CPU, les registres et les pages mémoires nécessaires pour activer la VM dans la machine destination) est transféré vers la machine destination. La VM commence son exécution sur la machine destination même si les pages mémoires n'ont pas été copiées complètement.

Néanmoins, des défauts de pages peuvent se produire lorsque les pages mémoires de la VM ne sont pas disponibles sur l'hôte destination. L'inconvénient de cette méthode est l'apparition des défauts de page mémoire dans la machine destinatrice.

La figure démontre le principe de fonctionnement de la migration Post-copy.

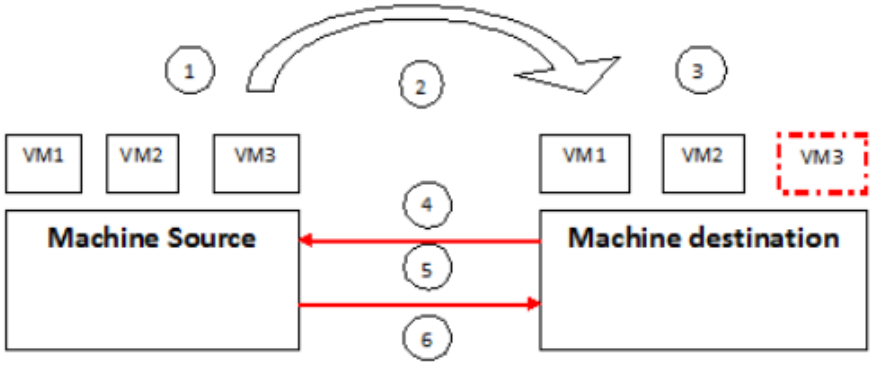


Figure I.25 : processus de la migration post-copy.

Nous prenons comme exemple la migration de la VM3 :

1. La VM3 est suspendue dans la machine source.
2. L'état d'exécution de la VM3 est transféré vers la machine destination.
3. La VM3 est activée dans la machine destination.
4. Générer des défauts de pages mémoires.
5. Transférer les pages à défauts correspondantes
6. Les étapes 4 et 5 sont répétées jusqu'à ce que toutes les pages mémoires soient transférées vers la machine destination et la VM3 s'exécute.

b. *Live migration « Pre-copy » :*

C'est une technique de la migration à chaud qui est utilisée par l'hyperviseur XEN, KNV et VMware. Elle envoie la VM entière vers la machine destination durant la première itération puis transfert de manière itérative les pages mémoires modifiées. Ce processus se poursuit jusqu'à ce que toutes les pages mémoires suffisantes soient copiées sur l'hôte destination. La figure résume le principe de la migration Pre-copy de la VM3 en 5 étapes comme suit :

1. Envoyer toute les pages mémoires de la VM3 de la machine source vers la machine destination.
2. Transférer les pages mémoires modifiées d'une manière itératives.
3. Arrêter la VM3 dans la machine source.
4. Envoyer les états restants de la VM3.
5. Activer la VM3 dans la machine destination.

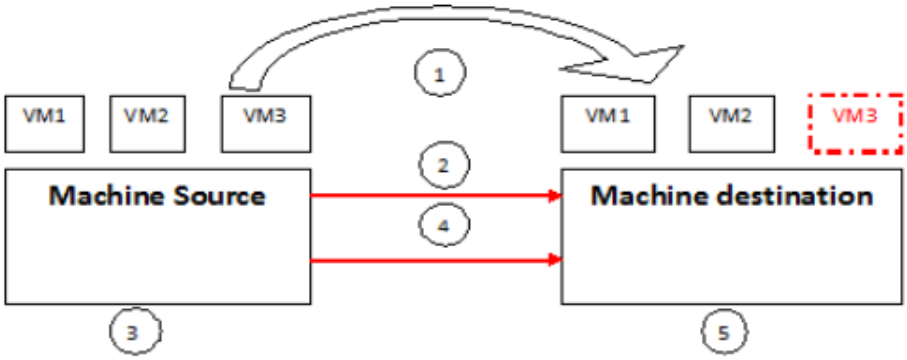


Figure I.26 : processus de la migration Pre-copy.

En cas de panne de la machine destination, la technique Pre-copy est bien meilleure que la Post-copy. La récupération de la VM3 est possible par la méthode Pre-copy car la dernière copie mémoire est conservée dans un lieu sur la machine source y compris l'état du CPU.

VII. Avantages et risques du cloud computing : [18] [19] [20] [21]

VII.1 Avantages du cloud computing :

Cloud Computing présente de nombreux avantages. Certains d'entre eux sont énumérés ci-dessous:

- ✓ Pouvoir d'accès aux applications en tant qu'utilitaires, sur Internet.
- ✓ Manipuler et configurer l'application en ligne à tout moment.
- ✓ Accéder et manipuler une application cloud sans installer pour autant une pièce de logiciel spécifique.
- ✓ Les ressources dans le cloud sont disponibles sur le réseau de manière à offrir un accès indépendant de la plateforme à tout type de clients.
- ✓ Cloud Computing est très rentable car il fonctionne à des rendements plus élevés avec une utilisation accrue. Il suffit d'avoir une connexion Internet.
- ✓ Cloud Computing offre un équilibrage de charge qui le rend plus fiable.
- ✓ Pour assurer un haut niveau de sécurité et de disponibilité des applications, les fournisseurs disposent de plusieurs centres de données réparties géographiquement.

VII.2 Risques du cloud computing :

Les grandes catégories de risques liées à l'adoption du Cloud se déclinent selon les points suivants :

- ***Sécurité et privacité :***

C'est la plus grande préoccupation concernant le cloud computing. Étant donné que la gestion des données et la gestion de l'infrastructure dans le cloud sont fournies par des tiers, il est toujours possible de transférer les informations sensibles à ces fournisseurs.

Bien que les fournisseurs de cloud computing garantissent des comptes protégés par un mot de passe plus sécurisé, tout signe de violation de sécurité entraînerait la perte des clients et des entreprises.

- ***Connexion :***

C'est l'autre goulet d'étranglement. Si l'utilisateur n'a pas de connexion internet, ou une connexion insuffisante, il ne pourra pas accéder à sa plateforme de travail. L'idée dans ce cas est de mettre le travail sur une application locale qui synchronise ensuite les données avec le serveur dès que l'utilisateur a à nouveau accès au réseau. Le problème de la sécurité des données en local se pose donc à nouveau.

- **Consommation d'électricité :**

Le plus gros problème du cloud computing c'est la consommation élevé de l'électricité et l'énergie dépensé par les Datacenter, Ce point sera bien détaillé dans le chapitre suivant.

- **Localisation des données** :

La dématérialisation des données sur différents sites physique de stockage peut conduire à un éclatement des données et leur répartition sur différentes zones géographiques.

Conclusion :

Dans ce chapitre, nous avons présenté une vue d'ensemble du cloud computing, son évolution, ses caractéristiques et ses modèles de déploiements et de services. Nous avons également défini la virtualisation et sa relation avec le cloud computing ; pour finir avec une présentation de ses intérêts et risques. Nous verrons dans le prochain chapitre de ce travail, le fonctionnement du cloud computing du côté fournisseur, la définition de la notion d'allocation de ressources avec la minimisation de la consommation d'énergie que nous présenterons également par la suite.

Chapitre II :

Allocation de ressource dans Le cloud

Introduction :

Comme nous l'avons vu précédemment, le cloud computing est la fourniture de ressources informatiques à la demande à travers un réseau Internet sur une base payante. Pour réaliser cela le fournisseur doit effectuer une allocation qui lui permettra de succomber aux besoins des utilisateurs, tout en assurant une réduction de l'énergie et un équilibrage de charge de ses serveurs.

Nous verrons dans les prochaines sections de ce chapitre une présentation du concept allocation de ressources dans le cloud computing, sa stratégie d'allocation en prenant en compte dans un premier temps la contrainte de consommation d'énergie que nous optimisons avec les métaheuristiques.

I. Définitions de bases :

I.1 Comment se fait l'allocation des ressources dans le cloud computing : [22] [23]

L'environnement de cloud computing se compose de deux acteurs majeurs : les utilisateurs et les fournisseurs. Les utilisateurs ont des applications avec des charges de travail variables à exécuter dans des centres de données cloud, gérés par les fournisseurs. Les fournisseurs de leur côté détiennent des ressources informatiques massives dans leurs grands centres de données (Datacenter) et allouent ces ressources aux utilisateurs.

L'allocation de ressource est le processus d'attribution des ressources disponibles pour les applications de cloud computing sur internet. L'allocation des ressources qui n'est pas gérée avec précision empêche le bon fonctionnement des services. L'approvisionnement de ressources résout ce problème en permettant aux fournisseurs de services de gérer les ressources pour chaque application.

I.2 Approvisionnement des ressources : [24]

L'approvisionnement des ressources est un terme utilisé dans le monde de l'informatique, désignant l'allocation automatique de ressources.

Les outils de provisioning sont des outils de gestion de configuration permettant d'installer et de configurer des logiciels à distance (télédistribution), ou encore d'allouer de l'espace disque, de la puissance ou de la mémoire.

Dans le monde des télécommunications, le provisioning consiste à adapter un service aux besoins d'un client. Dans certains cas l'utilisateur peut même effectuer lui-même certaines opérations : on parle dans ce cas de « self-provisioning».

Au sens large, le provisioning est l'affectation plus ou moins automatisée de ressources à un utilisateur.

I.3 L'allocation de ressources dans le cloud computing : [25]

L'utilisateur envoie une demande (request VM) contenant ces besoins en ressources au fournisseur, la demande est généralement effectuée via une réservation (CPU, RAM, espace disque, bande passante, etc.).Lorsque ce dernier reçoit la demande, il cherche des ressources pour satisfaire la demande et alloue ces derniers à l'utilisateur demandeur, généralement sous forme de machines

virtuelles (VM). Le fournisseur, en fonction des stratégies de gestion de son infrastructure, propose plusieurs formes de réservation :

- **Réservation immédiate**: le besoin en ressource doit immédiatement être satisfait,
- **Réservation programmée (ce qu'on appelle aussi la planification de ressource du coté fournisseur)**: le client indique la quantité de ressource souhaitée, la date de disponibilité de la ressource et la durée d'utilisation.

Ensuite, l'utilisateur utilise les ressources qui lui sont assignées pour exécuter son application. Ainsi au déclenchement de la réservation, les VMs sont démarrées et toutes les opérations de configuration permettant leur accès et leur exploitation à distance par l'utilisateur sont effectuées. Les ressources sont libérées au terme de la réservation. Toutefois, le client peut établir un contrat dit SLA à durée indéterminée avec le fournisseur et avoir ainsi immédiatement accès aux ressources en cas de besoin.

Par ailleurs, le mécanisme d'allocation de ressources vise à réduire le gaspillage de ressources, le temps de répartition des tâches mineurs et la satisfaction des utilisateurs. Tout d'abord, il définit le problème du placement de machines virtuelles particulières sur les machines physiques présentées, en particulier pour le modèle de demande de réservation avancée.

I.4 Différence entre allocation, approvisionnement et planification des ressources: [26]

Le concept de chevauchement entre l'approvisionnement des ressources, l'allocation des ressources et aussi la planification des ressources est brièvement définie par :

L'approvisionnement en ressources est l'attribution des ressources d'un fournisseur de services à un client, alors que l'allocation des ressources consiste à distribuer des ressources économiquement entre des groupes de programmes concurrents ou des utilisateurs, Et la planification des ressources est un calendrier d'allocation des ressources où les ressources sont partagées et disponibles à certains moments, et les événements informatiques sont planifiés pendant ces périodes. En d'autres termes, c'est le processus consistant à définir quand une activité de calcul devrait commencer ou se terminer, en fonction de ses :

- Activités prédécesseur
- Des relations antérieures
- Des ressources allouées
- De la durée.

En outre, l'allocation des ressources en nuage est la procédure de découverte, de sélection, de provisionnement, de planification des applications et de gestion des ressources.

I.5 Différence entre allocation d'une VM et allocation d'une ressource : [27] [28]

Placement de VM ou allocation d'une VM: est un problème typique dans les environnements en nuage. Une fois qu'une machine virtuelle a été admise dans le cloud, l'objectif du placement VM est de trouver le meilleur serveur ou machine physique (PM) pour exécuter la machine virtuelle.et cela est traduit comme l'attribution d'une machine virtuelle à un hôte.

D'autre part, l'allocation de ressources est un terme générique dans l'administration de systèmes de serveurs et en grande partie dans un environnement Cloud où un ensemble limité de ressources est partagé par plusieurs applications (VM).Le partage des ressources est transparent, chaque machine virtuelle pense que c'est la seule fonctionnant sur un nœud.

L'allocation de ressources a donc trait à la façon de partager efficacement les ressources limitées (CPU, MEM, DISK, NET) entre toutes les VM sur l'hôte de telle sorte que l'objectif de service de chaque machine virtuelle soit respecté tout en assurant une utilisation optimale de la ressource disponible pour réduire les coûts (énergie par exemple). Cela conduit à de nombreux problèmes d'optimisation.

II. Allocation de ressources : [26] [29]

Les ressources dans le cloud computing ne sont que la source pour répondre aux besoins informatiques requis. Ce sont deux types, ressources matérielles et ressources logicielles. Les serveurs et les centres de données sont des ressources matérielles. Les logiciels fonctionnant sur le côté du client et du serveur, les informations et les applications sont des ressources logicielles. L'allocation de ressources est discutée dans de nombreux domaines informatiques, comme la gestion des centres de données, les systèmes d'exploitation et l'informatique en grille.

Un système de répartition des ressources (RAS) en Cloud Computing peut être considéré comme tout système qui a pour but de garantir que les besoins des applications sont bien traités par l'infrastructure du fournisseur. Parallèlement à cette garantie pour le développeur, les mécanismes d'allocation de ressources devraient également tenir compte de l'état actuel de chaque ressource dans l'environnement Cloud, afin d'affecter les algorithmes pour mieux répartir les ressources physiques et / ou virtuelles avec les applications des développeurs, réduisant ainsi le coût opérationnel de l'environnement des nuages. Un point important lors de l'allocation de ressources pour les demandes d'arrivée est la façon dont les ressources sont modélisées.

Il existe de nombreux niveaux d'abstraction des services qu'un cloud peut offrir aux développeurs et de nombreux paramètres qui peuvent être optimisés lors de l'allocation. La modélisation des ressources devraient tenir compte au moins de ces exigences afin que le RAS fonctionne correctement. Les ressources en nuage peuvent être considérées comme une ressource (physique ou virtuelle) que les développeurs peuvent exiger du Cloud. Par exemple, les développeurs peuvent avoir des exigences de calcul, telles que la CPU, la mémoire et le stockage et les exigences du réseau, telles que la bande passante et le délai.

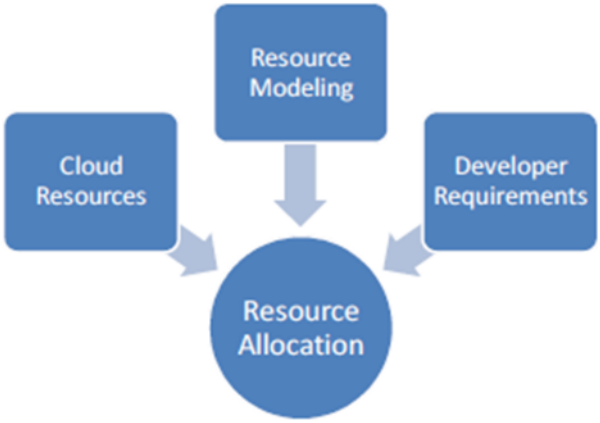
En outre, d'autres exigences sont également possibles pour les Clouds, comme la topologie du réseau de tous les nœuds, le délai maximal entre les nœuds et l'interaction entre différentes applications.

Généralement, les ressources sont positionnées dans un datacenter. Ceux-ci sont utilisés par plusieurs clients et doivent être vigoureusement affectés et ajustés selon les besoins.

Il est important de noter que les développeurs et les clients peuvent voir ces ressources finies comme illimitées et l'outil qui rendra cela possible est le RAS.

Le RAS devrait traiter ces demandes imprévisibles de manière flexible et transparente. Cette flexibilité devrait permettre l'utilisation dynamique des ressources physiques, évitant ainsi à la fois le sous-provisionnement et le sur-provisionnement des ressources.

De cette façon, on peut considérer que les ressources Cloud, la modélisation des ressources et les exigences du développeur sont les entrées d'un système d'allocation de ressources, comme le montre la Figure.



FigureII.1 : les entrées d'un système d'allocation de ressources.

II.1 L'allocation efficace des ressources : [30] [25]

L'allocation des ressources disponibles aux consommateurs de cloud est une tâche énorme. Le provisioning des ressources cloud se fait par tenu en compte des accords de niveau de service (SLA). Une allocation efficace des ressources, devrait éviter les situations suivantes [31]:

- ***Sur-provisioning :*** se pose lorsque le fournisseur alloue pour le consommateur des ressources plus que la demande.
- ***Sous-provisioning :*** se produit lorsque l'allocation des ressources est moins que la demande.
- ***Situation conflictuelle de ressource :*** se pose lorsque deux applications tentent d'accéder à la même ressource en même temps.
- ***Le manque en ressources :*** se pose lorsque les ressources sont limitées.
- ***La fragmentation des ressources :*** cette situation se pose lorsque les ressources sont isolées. (Il y a suffisamment de ressources, mais l'allocation est impossible.)

II.2 Allocation de machines virtuelles dans le cloud : [32]

L'informatique en nuage est stimulée par des économies d'échelle. Un système en nuage utilise la technologie de virtualisation pour fournir des ressources (par exemple, CPU, mémoire) aux utilisateurs sous forme de machines virtuelles. La machine virtuelle, qui est un bac à sable pour application utilisateur, s'adapte bien dans l'environnement éducatif pour fournir des ressources informatiques pour les besoins en enseignement et en recherche. Selon la vue du propriétaire de la ressource, ils veulent réduire les coûts d'exploitation qui induit la facture électronique du système de centre de données à grande échelle.

Le problème d'allocation de Machine Virtuelle(VM) dans les centres de données virtualisées est un sujet difficile. Il peut être vu dans la cartographie statique et dynamique.

Une allocation VM à pour but de cartographier chaque VM à des machines physiques (PM) pour optimiser une fonction objectif donnée. La fonction objectif peut être de maximiser les performances, de minimiser la consommation d'énergie, ou de maximiser les bénéfices du fournisseur, etc. L'allocation statique de la VM peut être considérée comme un problème d'emballage vectoriel bidimensionnel (VBD), Les machines virtuelles sont des articles et les machines physiques sont des bacs. Le problème de remplissage du VBPd ($d \geq 1$) est NPHard. Une allocation de VM dynamique diffère de l'allocation de VM statique, c'est que dans l'allocation de VM dynamique, un événement système (par exemple une faible utilisation du processeur, une température du système, une défaillance matérielle ou logicielle, etc.) peut déclencher une re-cartographie de l'ensemble des VM et du jeu Des PM.

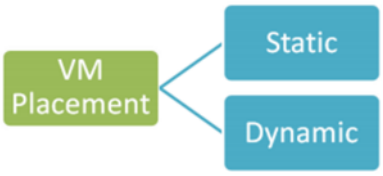
II.3 Placement de machines virtuelles dans le Cloud Computing : [33]

L'un des principaux concepts du cloud computing est la virtualisation. Il a l'avantage majeur de permettre l'exécution de plusieurs instances du système d'exploitation sur une seule PM, permettant ainsi une utilisation complète des capacités matérielles de la PM. Ces instances du système d'exploitation sont appelées machines virtuelles. Le placement des machines virtuelles dans l'environnement de cloud computing est très crucial car le nombre d'utilisateurs de cloud est en hausse. La planification des machines virtuelles affecte grandement la performance et le débit du système entier. La virtualisation dans une machine physique est prise en charge par l'Hypervisor comme discuté dans le chapitre précédent.

Les demandes de machine virtuelle suivent une approche de distribution à deux niveaux. Un grand nombre de machines physiques sont déployées dans un centre de données. Un fournisseur de services Cloud peut disposer de plusieurs centres de données. Ainsi, les demandes VM doivent d'abord être distribuées de manière optimale sur les centres de données. Les requêtes dans chaque centre de données sont ensuite distribuées sur des machines physiques.

Il existe deux types de placement de la machine virtuelle dans les machines physique :

- Placement statique
- Placement dynamique



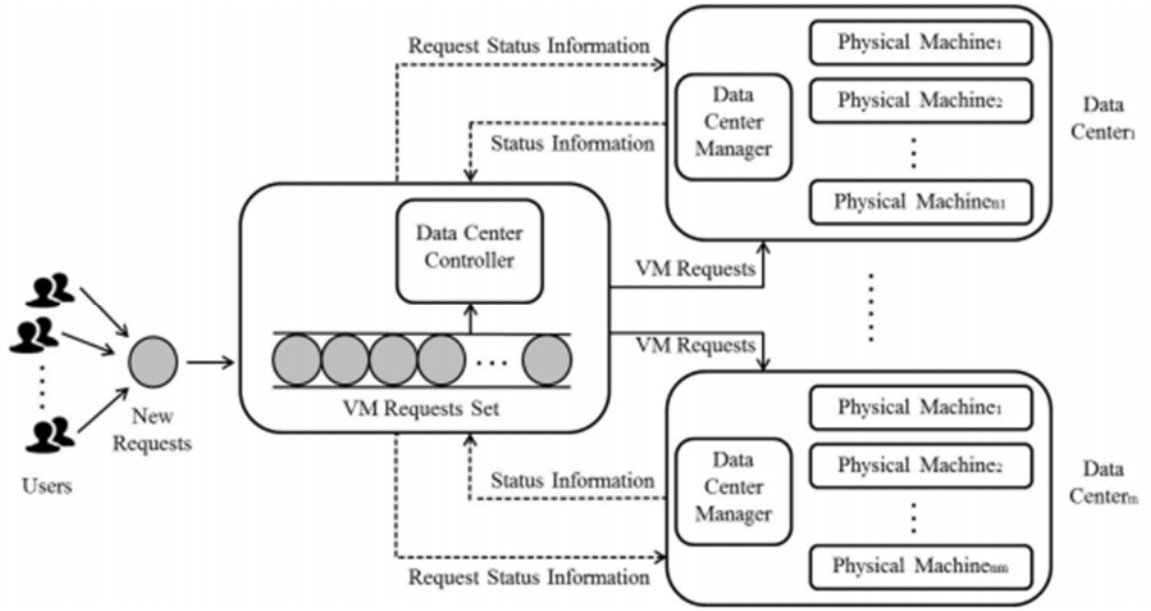
FigureII.2 : Placement de machine virtuelle.

III. Allocation statiques des machines virtuelles dans le cloud :

Le placement statique des machines virtuelles se fait soit au démarrage du système, soit en mode hors ligne. Il s'agit du placement initial des VM dans l'environnement de cloud. Aucune correspondance préalable des machines virtuelles n'est trouvée. Ce type de placement de machine virtuelle ne tient compte ni des états des machines virtuelles et des machines physiques, ni du taux d'arrivée des demandes des utilisateurs.

L'explication schématique du modèle centralisé pour le positionnement statique de la machine virtuelle selon l'approche à deux niveaux est expliquée ci-dessous. Tout d'abord, le placement des machines virtuelles sur les centres de données et le placement des demandes sur les PMs dans un centre de données particulier sont affichés.

La **FigureII.3** présente le flux de requêtes VM des utilisateurs jusqu'à leur distribution sur les centres de données.

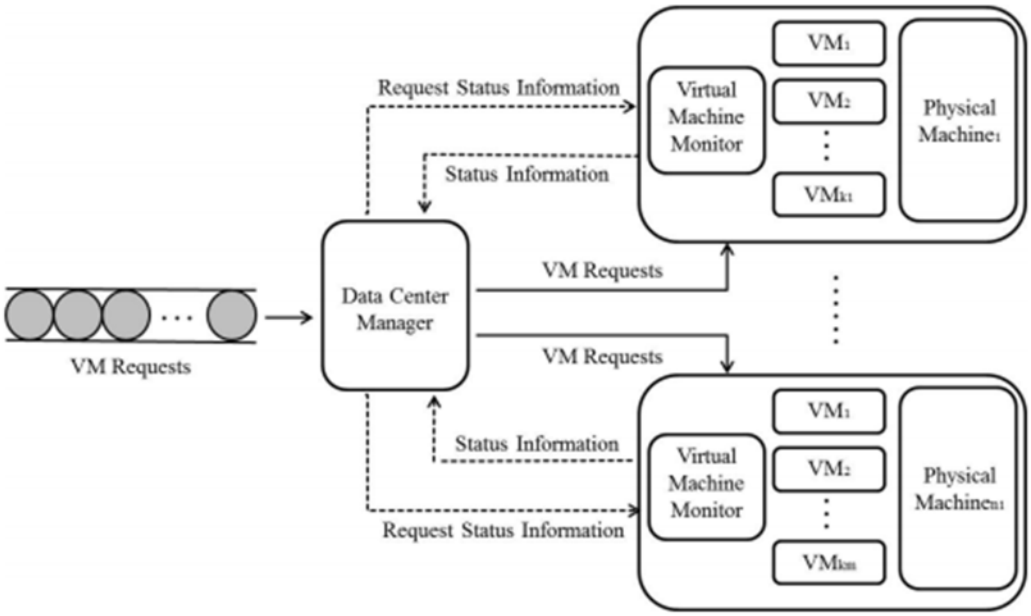


FigureII.3 : Placement statique de machines virtuelles sur des centres de données.

Les utilisateurs de Cloud utilisent les services fournis par le fournisseur de services cloud et émettent des demandes. Ces requêtes se présentent sous la forme de requêtes de machines virtuelles car chaque requête sera terminée sur une machine virtuelle au-dessus d'une machine physique. Les requêtes VM constituent le jeu de requêtes VM. Chaque centre de données dispose de plusieurs machines physiques et d'un gestionnaire de centre de données pour contrôler toutes les PM. Afin de contrôler tous les centres de données, un contrôleur de centre de données est utilisé.

Le contrôleur de centre de données reçoit les requêtes VM définies. Il demande ensuite des informations d'état à tous les gestionnaires de centres de données qui ont les informations de leurs centres de données respectifs. Les requêtes VM contiennent des informations sur les ressources (CPU, RAM, Réseau, etc.) nécessaires pour compléter la requête. Les gestionnaires de centre de données envoient des informations sur les ressources disponibles au contrôleur. Le contrôleur de centre de données distribue de manière optimale les requêtes VM aux gestionnaires de centre de données suivant un algorithme heuristique. Cette approche pour distribuer des requêtes VM est une approche centralisée, où la décision de planifier des demandes dépend du contrôleur central.

La **FigureII.4** illustre le placement statique des requêtes de machines virtuelles dans des machines physiques dans un seul centre de données.



FigureII.4 : Placement statique de machines virtuelles sur des machines physiques.

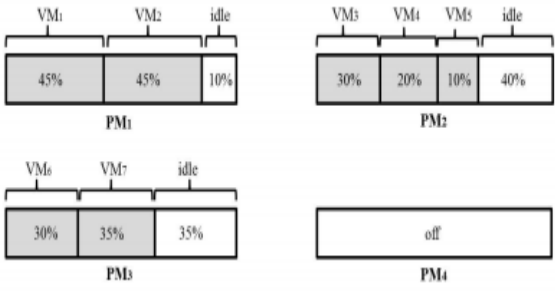
Dans un centre de données, il existe plusieurs machines physiques sur lesquelles se trouve le Virtual Machine Monitor [VMM] (Hypervisor). Le VMM est responsable de la virtualisation des PM. Le gestionnaire de centre de données envoi des requêtes aux VMM pour fournir les informations d'état de toutes les machines physiques. Le gestionnaire de centre de données possède déjà les informations des requêtes VM. À l'arrivée des informations d'état des VMM, le gestionnaire de DC place les requêtes VM sur les machines physiques individuelles afin de les traiter et de renvoyer la réponse aux utilisateurs de nuages. Le placement de requêtes VM sur des machines physiques est également une approche centralisée car la décision est prise par une autorité centrale, en l'occurrence le gestionnaire de centre de données.

IV. Placement dynamique des machines virtuelles dans le cloud computing :

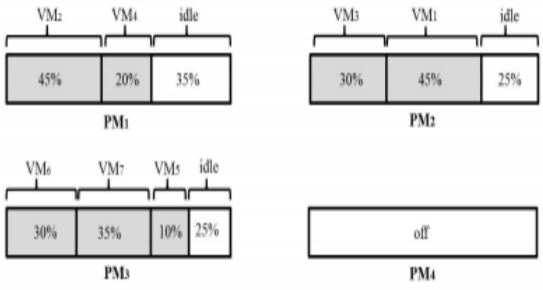
Le positionnement dynamique des machines virtuelles place ces dernières sur la base d'un mappage existant, ce qui contraste avec le placement statique des machines virtuelles qui commence sans mise en correspondance.

Le placement dynamique de machines virtuelles vise à obtenir des solutions optimales à partir du mappage existant à un coût minimum. Cela n'entraînerait pas ou arrêter les VM déjà en cours d'exécution, donc la solution de placement devrait fournir la liste des migrations en direct à exécuter afin d'atteindre l'état optimal à partir de l'état existant. L'ensemble du processus est en contraste avec le placement statique des machines virtuelles où les machines virtuelles peuvent être arrêtés et redémarrés ce qui augmente la consommation d'énergie et dégrade ainsi la performance du système complet. Afin d'exécuter le placement dynamique des machines virtuelles, les états des machines physiques devraient également être pris en considération.

Voici un exemple de placement dynamique de machines virtuelles.



FigureII.5 : Solution Initial i.



FigureII.6 : Solution finale s1.

La **FigureII.5** montre la solution initiale i pour le problème de placement dynamique de VM. Sept machines virtuelles sont placées sur quatre PM. PM4 est hors tension car aucune VM ne s'exécute. En supposant que toutes les PM ont la même quantité de ressources, les VM doivent être distribués dynamiquement sur différents PMs afin de réduire la consommation d'énergie.

Une telle solution est représentée sur la **FigureII.6.** Pour atteindre la solution s1 à partir de i, des migrations doivent être effectuées. La liste des migrations est: VM5 de PM2 à PM3, migration de VM1 de PM1 vers PM2 et finalement migration de VM4 de PM2 vers PM1.

V. Consommation d'énergie dans le cloud computing : [34]

A travers la définition de Cloud, on entrevoit déjà de nombreux points positifs. Et comme nous avons cité dans le chapitre précèdent que les avantages du Cloud sont effectivement nombreux : stockage à distance, messagerie électronique, etc. tout ceci, grâce à des datas centers en France ou ailleurs. Un data center est un centre de serveurs sur lesquels sont stockés, et à travers lesquels transitent toutes les données des solutions Cloud. Mais tout cela implique des dépenses énergétiques énormes.

L'augmentation continue de la consommation d'energie dans les clouds computing souleve une grande préoccupation pour les fournisseurs de services.Leur objectif principal est de fournir des services aux clients tout en augmentant leurs revenus et en consommant plus efficacement l'energie consommée par leur infrastructures.outre les couts d'exploitation écrasante et le cout total d'acquisition (TCA) provoque une augmentation de la consommation energitique,une autre préoccupation croissante est l'impact sur l'environnement en termes de dioxyde de carbone(CO2).

Les centres informatiques sont devenus très gourmands en consommation énergétique. Il leur faut de l'électricité pour, d'une part, faire tourner les milliers d'ordinateurs nécessaires à traiter, enregistrer, retrouver toutes ces données, si on à la quantité de données que chaque utilisateur d'Internet exploite par jour en ligne en utilisant les services de l'informatique en nuage… ou simplement, à la quantité de réponses que renvoie le moteur de recherche Google à chaque opération de recherche, on trouvera que c'est une masse de données énorme qui est en jeu.

Il faut ensuite s'imaginer les milliers, les millions, puis les milliards de webnautes qui sont connectés tous les jours. Les chiffres sont abasourdissants. Pour la petite idée, les recherches Google impliquent chaque mois 3.900.000 kWh, ou encore 260.000 kilogrammes (kg) de CO2. Cela correspond à la quantité d'énergie qui permettrait de maintenir une lampe à incandescence de puissance 100 watts allumée pendant 4.534 ans, rien que ça. Pas étonnant que les statistiques sur la consommation énergétique mondiale augmentent de façon vertigineuse. Et d'autre part, refroidir ces systèmes qui ont la fâcheuse tendance à chauffer. En France, différentes études estiment que les datacenters installés sur le territoire absorbent à eux seuls 9% de la consommation totale d'électricité du pays ***[35]***. Corollaire, ils émettent du CO2. Ils seraient d'ores et déjà responsables de 2% des émissions mondiales de CO2, soit au moins autant que le trafic aérien.

Rien d'étonnant donc à ce que les projets de recherche se multiplient pour tenter de réduire la consommation énergétique de ces datacenters, surtout au moment de la transformation numérique de tous les secteurs d'activité, qui augmente le nombre de datacenters partout dans le monde.

Le premier moyen, et le plus simple, retenu actuellement par nombre d'opérateurs de datacenters est de construire ces centres dans des régions fraîches et si possible à proximité de la mer, d'un lac ou d'un fleuve. Ces localisations présentent un double avantage : la température basse permet une climatisation par l'air mais de faible ampleur ; il est possible de puiser de l'eau à basse température pour alimenter le système de refroidissement des serveurs. Les opérateurs cherchent aussi à s'approvisionner en énergies renouvelables afin d'améliorer leur empreinte carbone.

D'autres projets se sont concentrés sur l'analyse de l'activité énergétique d'un centre afin de l'optimiser et de réaliser des gains d'énergie plus ou moins importants. Mais c'est surtout via la récupération de chaleur émise par les datacenters que le bilan énergétique d'un datacenter peut être amélioré. Plusieurs centres, implantés à proximité de zones d'habitation, alimentent ainsi un réseau de chaleur qui chauffe les appartements et les maisons du voisinage.

Le projet le plus original dans ce domaine est sans conteste celui d'un jeune ingénieur français qui a imaginé un processus totalement différent. Plutôt que de récupérer la chaleur pour la transporter vers les lieux d'habitation, il installe des radiateurs numériques dans les appartements, radiateurs qui sont en fait des ordinateurs ! Au lieu de traiter les informations dans un datacenter, il répartit les traitements et les fait réaliser par les ordinateurs installés dans les appartements. La chaleur est ainsi directement utilisée pour chauffer le logement. La solution a été testée dans 100 logements à Paris avec succès ***[34]***. Le modèle économique repose entièrement sur les clients qui consomment les traitements informatiques, de fait, le chauffage est gratuit pour les occupants des logements.

V.1 Consommation d'énergie par le data center : [35] [36]

Plus nos activités privées et professionnelles se numérisent, plus nous consommons de l'électricité. La consommation d'un data center en termes d'énergie s'exprime en milliers de kWh (kilowattheure). L'explication est toute simple quand on comprend le principe de fonctionnement de cet ensemble de techniques. Les serveurs ou les super-serveurs, qui sont le chemin de transition et de stockage des données, doivent nécessairement rester fonctionnels à plein temps. Il serait quand-même bien dommage si l'on devait attendre une certaine heure précise avant de pouvoir envoyer un mail, d'autant plus avec le décalage horaire… ce serait catastrophique. Alors les machines doivent être allumées à toutes heures, et par conséquent doivent être alimentées sans cesse par de l'électricité.

V.2 Consommation d'énergie par les serveurs : [37]

Dans une vision axée sur l'économie et l'écologie, les ressources informatiques utilisées par les entreprises sont rarement en adéquation avec leurs besoins réels. Elles disposent par exemple souvent de plusieurs serveurs fonctionnant à moins de 15 % de leur capacité totale, celle-ci n'étant là que pour faire face à tout moment aux pointes de consommation excessive. Un serveur physique chargé à 15 % ne consomme pas moins d'énergie qu'un autre chargé à 90 %, et dans une moindre mesure, prend autant de place physique dans nos bureaux… L'idée de la virtualisation et la régulation de charge du travail sont deux techniques qui permettent la réduction de l'énergie consommée par les différents serveurs.

V.2.1 Technique de réduction de l'énergie consommée par les serveurs :

a. La virtualisation : [15]

La virtualisation est donc de regrouper plusieurs serveurs sur une même machine afin de rentabiliser l'investissement, la place et la consommation. Aussi, cela permet un bien plus grande gestion dans la répartition des charges et la reconfiguration en cas d'évolution ou de défaillance passagère.

b. Régulation de la charge du travail : [38]

Réduire la consommation d'énergie passe aussi par la diminution du nombre de serveurs en marche. Avant d'éteindre un serveur, les applications qui tournent sur celui-ci doivent être transférées à un autre. Pour cela, les chercheurs utilisent le principe de machine virtuelle. Afin de libérer la charge de travail d'un serveur, les chercheurs ont deux possibilités : soit ils suspendent le calcul, soit ils effectuent une migration à l'aide de machines virtuelles. Cet ordonnancement des tâches sur les serveurs est un problème complexe. C'est avant tout une question de placement et de répartition.

Jean-Marc Menaud explique : « Le principe de ce placement est proche du principe du remplissage d'un sac à dos. Vous partez faire un trek avec un sac à dos de 60 litres. Vous avez le choix entre une multitude d'aliments à emporter avec vous. Chaque aliment a une valeur calorique, un volume et un poids. Votre objectif est d'accumuler un maximum de calories avec la contrainte d'un sac à volume fixe, tout en minimisant le poids final. La solution est simple lorsqu'il n'y a que 5 aliments. Mais si vous en avez 10 000, le problème se complexifie car on ne peut pas tester toutes les possibilités. Ici notre situation est similaire. Un serveur est un sac à dos qui peut contenir une certaine quantité de machines virtuelles. On doit maximiser le service rendu (les calories) en minimisant l'énergie (le poids) ».

c. Allocation efficace de machines virtuelle :

Une allocation efficace des machines virtuelles sur PM peut entraîner une meilleure utilisation des ressources et entraîner une économie d'énergie. Dans ce point, nous visons à proposer une politique de placement de VM améliorée et efficace en énergie pour l'équilibrage de charge dans un environnement Cloud qui place les machines virtuelles de telle sorte que la situation de surcharge et de

sous-charge des hôtes soit traitée et entretienne un accord de niveau de service (SLA) entre le fournisseur Cloud et l'utilisateur. De plus, nous proposons un algorithme permettant de réduire la consommation d'énergie et d'obtenir un meilleur équilibrage de charge.

Comme nous avons défini précédemment que le problème de mappage de VM dans les PM est un problème de bin packing (emballage de bac) ou un problème de sac à dos, ce genre de problème est résolu à l'aide de l'un des algorithmes d'optimisation.

VI. Les algorithmes d'optimisation : [39]

Les différents types d'algorithmes qui permettent d'obtenir des solutions pas trop mauvaises, mais non exactes, à des problèmes d'optimisation sont catégorisés comme suit :

a. Les algorithmes d'approximation.
b. Les algorithmes heuristiques.
c. Les algorithmes probabilistes.
d. Les métaheuristiques.

VI.1 Algorithmes d'approximation : [39]

Un Algorithmes d'approximation, plutôt que résoudre de façon exacte un problème et trouver une solution optimale, permet d'obtenir une solution pas trop éloigné de la solution optimale, donc une solution qui approxime la solution optimale.

VI.2 Les algorithmes heuristiques : [39]

Un algorithme heuristique produit aussi une solution qui approxime la solution optimale.

Une heuristique est une technique de résolution spécialisée à un problème. Elle ne garantit pas la qualité de la solution obtenue, il peut même exister certains cas où l'heuristique produira une solution très mauvaise, ou encore sera tout à fait incapable de produire une solution.

VI.3 Les algorithmes probabilistes : [40]

Un algorithme probabiliste est un algorithme qui utilise une source de hasard. Plus précisément le déroulement de l'algorithme fait appel à des données tirées au hasard. Par exemple à un certain point de l'exécution, on tire un bit 0 ou 1, selon la loi uniforme et si le résultat est 0, on fait une certaine action A et si c'est 1, on fait une autre action. On peut aussi tirer un nombre réel dans l'intervalle [0,1] ou un entier dans un intervalle [i..j].

VI.4 Les métaheuristiques : [39]

Une métaheuristique est une stratégie générale de résolution de problème qui utilise, coordonne et guide d'autres heuristiques pour obtenir une solution à un problème difficile. Alors que les heuristiques sont généralement spécifiques à un problème, les métaheuristiques sont conçues pour s'adapter à chaque problème.

Une caractéristique commune à de nombreuses métaheuristiques est qu'elles ont été développées en s'inspirant de processus qu'on retrouve dans la nature, par exemple ,en physique ou en

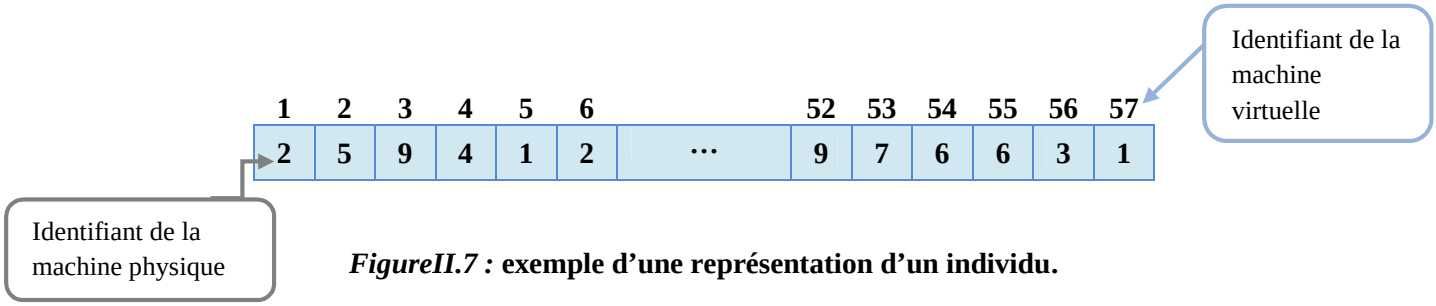
biologie. Parmi les plus connus, on retrouve les algorithmes évolutionnaires et les algorithmes de colonie de fourmis.

VI.4.1 Les algorithmes de colonie de fourmis :

Cette approche s'efforce de simuler la capacité collective de résolution de certains problèmes, observée chez une colonie de fourmis, dont les membres sont pourtant individuellement dotés de faculté très limitées. En particulier les entomologistes ont analysé la collaboration qui s'établit entre les fourmis pour aller chercher de la nourriture à l'extérieur de la fourmilière, il est remarquable que les fourmis suivent toujours le même chemin, et que ce chemin soit le plus court possible. Cette conduite est le résultat d'un mode de communication indirecte, via l'environnement : la « stigmergie ». Chaque fourmi dépose, le long de son chemin, une substance chimique, dénommée 'phéromone' ; tous les membres de la colonie perçoivent cette substance et orientent préférentiellement leur marche vers les régions plus « odorante ».

VI.4.2 Les algorithmes évolutionnaires : [39] [41]

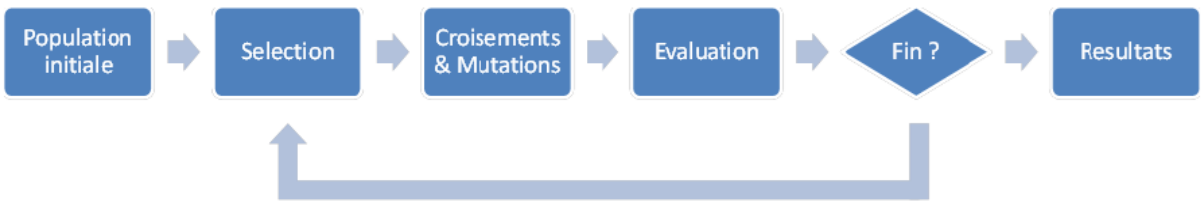
Les algorithmes évolutionnaires sont inspirés du domaine de la biologie ; les techniques utilisées reproduisent le schéma d'évolution des espèces et en adoptent le vocabulaire. Les solutions sont appelées individus et l'algorithme traite simultanément plusieurs individus ; un exemple d'individu utilisé dans la simulation de notre problématique est illustré ci bas :



***FigureII.7 :* exemple d'une représentation d'un individu.**

L'ensemble de ces individus est appelé population et évolue à chaque itération de l'algorithme. La population relative à une itération donnée s'appelle génération, on obtient donc une nouvelle génération à chaque itération. Les individus qui servent à produire la nouvelle génération sont appelés parents et les individus résultants sont appelés enfants ou fils.

Le principe des algorithmes évolutionnaires est simple, l'idée est de faire évoluer une population initiale grâce à des mécanismes de reproduction et de sélection de manière à obtenir des individus de plus en plus performants; la figure ci-dessous illustre cela.



***FigureII.8:* Fonctionnement des algorithmes génétiques.**

- ***Population initiale :***

Initialement, on génère un nombre aléatoire d'individus afin de construire la population initiale.

- ***Sélection :***

La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de converger vers l'optimum global.

Il existe plusieurs techniques de sélection. Voici les principales utilisées :

Sélection par rang **:** Cette technique de sélection choisit toujours les individus possédant les meilleurs scores d'adaptation.

Probabilité de sélection proportionnelle à l'adaptation **:** Technique de la roulette ou roue de la fortune, pour chaque individu, la probabilité d'être sélectionné est proportionnelle à son adaptation au problème.

Sélection par tournoi **:** Cette technique utilise la sélection proportionnelle sur des paires d'individus, puis choisit parmi ces paires l'individu qui a le meilleur score d'adaptation.

Sélection uniforme **:** La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation.

Voici un exemple des individus une fois la sélection effectuée

1	2	3	4	5	6	7	8	9	10
2	5	9	4	1	2	9	7	6	6

1	2	3	4	5	6	7	8	9	10
1	5	5	4	3	1	8	8	4	6

1	2	3	4	5	6	7	8	9	10
8	3	2	6	1	8	4	2	3	3

1	2	3	4	5	6	7	8	9	10
1	5	5	4	3	1	8	8	4	6

1	2	3	4	5	6	7	8	9	10
8	3	2	6	1	8	4	2	3	3

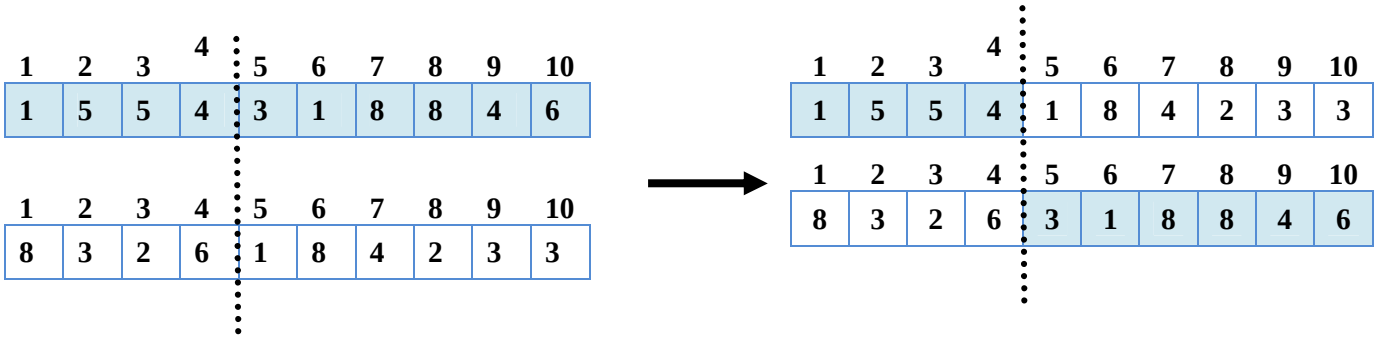
On réalise ensuite un croisement entre deux chromosomes parmi la population restante.

- ***Croisement :***

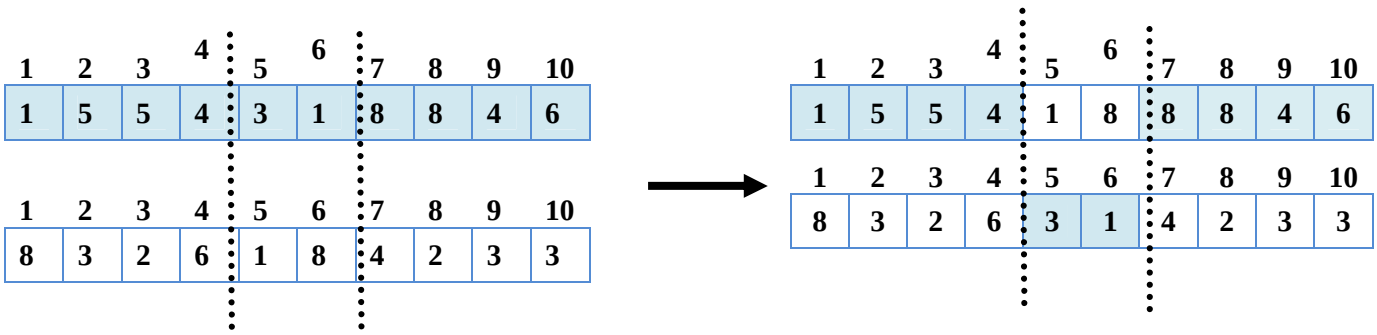
Les opérateurs de croisement produisent un ou deux enfants à partir de deux parents, c'est le résultat obtenue lorsque deux chromosomes partage leurs particularités.

Il existe deux méthodes de croissement : simple ou double croisement.

Le simple croisement consiste à fusionner les particularités de deux individus à partir d'un pivot, afin d'obtenir un ou deux enfants :



Le double croisement repose sur le même principe, sauf qu'il y a deux pivots :



On réalise ensuite une mutation sur les enfants obtenues lors du croisement.

- ***Mutation :***

Les opérateurs de mutation produisent un enfant à partir d'un unique individu, en altérant un gène dans un individu selon un facteur de mutation. Ce facteur est la probabilité qu'une mutation soit effectuée sur un individu.

Voici un exemple de mutation sur un individu ayant un seul chromosome :

1	2	3	4	5	6	7	8	9	10
1	5	5	4	3	1	8	8	4	6

↓

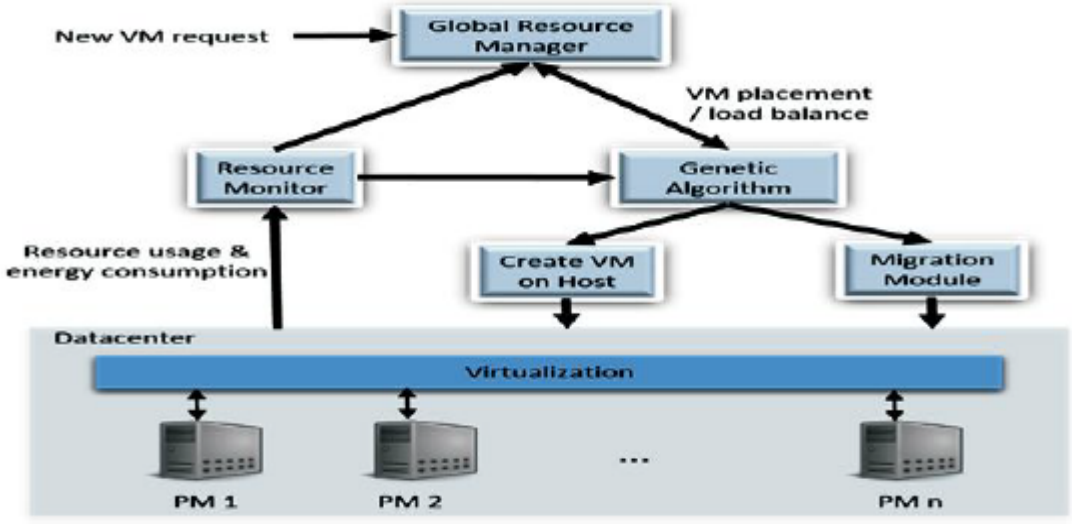
1	2	3	4	5	6	7	8	9	10
1	5	2	4	3	1	8	8	4	6

- ***Evaluation:***

Chaque individu se voit attribuer un score (valeur de **fitness**), suite au calcul d'une fonction objectif ; chaque individu est donc évalué en fonction de cette valeur, et seuls les chromosomes ayant obtenus une valeur de fitness estimée être assez bonne pour faire partie de la génération sont retenus pour constituer la population de travail de la génération suivante. Ainsi, à chaque génération le jeu des opérateurs créant de nouveaux chromosomes tend à ce que la population contienne des individus représentant de meilleures solutions.

- ***Comment appliquer des AG dans un problème du placement des VM: [42]***

Comme nous avons défini précédemment que le problème du placement de machine virtuelle est un problème NP_hard, tant d'auteur ont proposé les Algorithmes génétiques pour résoudre ce genre de problème.



FigureII.9 : Architecture du système du placement optimal des VM

Le système est un grand centre de données composé de différentes machines physiques ayant des capacités de ressources différentes. Chaque machine est caractérisée par une fonction CPU définie en Million d'instructions par seconde (MIPS), et mémoire (RAM), bande passante et stockage de

disque. ***FigureII.9*** donne l'architecture complète du système. Le gestionnaire de ressources globales gère les ressources parmi les machines virtuelles.

L'utilisateur peut demander une machine virtuelle à tout moment et le gestionnaire de ressources global utilise un algorithme génétique pour rechercher la disponibilité des ressources parmi toutes les machines physiques et place la machine virtuelle sur cette machine physique. Il équilibre également la charge entre les machines physiques disponibles à un intervalle de temps régulier.

Rousource Monitor surveille toutes les utilisations des ressources des machines physiques à des intervalles de temps réguliers qui peuvent être utilisés plus tard par le gestionnaire de ressources global (pour vérifier les machines physiques surchargées) et l'algorithme génétique (il considère un tableau historique pendant le placement). Il maintient le tableau de l'historique de l'hôte pour chaque machine physique pour stocker l'utilisation de la ressource de cette machine physique en tant qu'historique. Elle regroupe régulièrement l'utilisation des ressources de toutes les unités physiques et stocke ces valeurs dans le tableau d'historique de l'utilisation des ressources. L'algorithme est utilisé pour le placement de nouvelles machines virtuelles et pour l'équilibrage de charge entre les machines physiques. Il utilise une valeur de seuil supérieure pour satisfaire la demande croissante soudaine d'une machine virtuelle déjà existante sur la machine physique particulière. Dans notre travail nous allons utiliser plusieurs contraintes (CPU, RAM, bande passante, violation SLA…) pour placer les VM dans les machines physiques. Le module de migration est responsable d'effectuer la migration des Machines virtuelles résultant de l'algorithme. Notre algorithme génétique a pour but de trouver le meilleur placement pour les VM en prenant en considération les contraintes cité précédemment.

L'algorithme utilise une approche d'optimisation pour minimiser la consommation d'énergie et la charge du travail des serveurs en estimant l'énergie du serveur après avoir placé une machine virtuelle particulière dans ce dernier.

Conclusion :

Dans ce chapitre, nous avons vu une définition détaillée du concept d'allocation de ressources dans le cloud computing. Nous avons défini sa stratégie d'allocation en tenant en compte de la contrainte de consommation d'énergie que nous optimisons avec les métaheuristiques, nous avons fait également référence à une autre fonction objectif qui est la régulation de la charge de travail des serveurs dans le cloud.

Dans le chapitre suivant, nous verrons l'approche d'optimisation multi-objectif dans la résolution de notre problème de minimisation de la consommation d'énergie et l'équilibrage de la charge.

Chapitre III :

Analyse et conception

Introduction :

Comme nous l'avons vu précédemment, le problème majeur du cloud computing c'est l'affectation des ressources tout en minimisant l'énergie et la charge du travail des serveurs, pour résoudre ce problème on a opté pour une approche d'optimisation qui consiste à appliquer un algorithme d'optimisation pour minimiser ces deux fonctions objectifs.

I. Approches d'optimisation : *[43]*

Cette section présente les alternatives d'optimisation proposées dans les articles étudiés selon l'approche d'optimisation des fonctions objectives étudiées. Les approches d'optimisation identifiées peuvent être classées comme suit:

- Approche d'optimisation mono-objectif (MOP),
- Approche Multiobjectif résolu comme mono-objectif (MAM)
- Approche Multiobjectifs (MO).

I.1 Approche mono-objectif :

Une approche mono-objectif considère l'optimisation d'un seul objectif ou l'optimisation individuelle de plus d'une fonction objectif, une à la fois. Compte tenu du grand nombre de fonctions objectifs proposées et de différentes approches pour la modélisation d'une fonction objectif, l'optimisation multiobjectif pourrait aboutir à des formulations meilleures et plus réalistes du problème VMP, optimisant plus qu'une simple fonction objectif à la fois.

I.2 Approche multi-objectif résolu comme une approche mono-objectif :

L'optimisation de multiples fonctions objectifs combinées en une fonction objectif est considérée comme une approche multi-objectif résolue comme mono-objectif (MAM). Un inconvénient de cette approche est qu'elle nécessite une connaissance approfondie du domaine du problème pour permettre une combinaison correcte des fonctions objectifs, ce qui n'est pas possible dans la plupart des cas.

I.3 Approche multi-objectif :

Une approche d'optimisation multiobjectif pure (MO) considère l'optimisation de plusieurs fonction objectif à la fois et il comprend un ensemble de variables de décision, des fonctions objectifs et des contraintes. Les fonctions objectifs et les contraintes sont des fonctions des variables de décision.

Les problèmes multiobjectifs ont la particularité d'être difficiles à traiter; La difficulté réside dans l'absence d'une relation d'ordre total entre les solutions. Une solution peut être meilleure qu'une autre sur certains objectifs et moins bonne sur les autres. Donc il n'existe généralement pas une solution unique qui procure simultanément la solution optimale pour l'ensemble des objectifs. Ainsi le concept de solution optimale devient moins pertinent en optimisation multi-objectif. Dans ce cas la solution optimale n'est plus une solution unique mais, un ensemble de solutions compromis entre les différents objectifs à optimiser. Il est important pour identifier ces meilleurs compromis de définir une relation d'ordre entre ces éléments. La plus célèbre et la plus utilisée est la relation de dominance au sens Pareto (qu'on développera plus loin). L'ensemble des meilleurs compromis est appelé le front Pareto, la surface de compromis ou l'ensemble des solutions efficaces. Cet ensemble de solutions constitue un équilibre, dans le sens qu'aucune amélioration ne peut être faite sur un objectif sans

dégradation d'au moins un autre objectif. La solution Pareto consiste à obtenir le front de Pareto *PO (*individus non-dominés) ou d'approximer la frontière de Pareto *PO**.

I.3.1 Caractéristiques d'optimisation multiobjectif : *[44]*

a. Fonction objectif :

Une fonction objective sert de critère pour déterminer la meilleure solution à un problème d'optimisation. Elle peut être à minimiser ou à maximiser. Concrètement, elle associe une valeur à une instance d'un problème d'optimisation.

Nous nous basons principalement dans notre projet sur deux fonctions objectif à minimiser : l'énergie et la charge de travail qui sont définies dans la suite de ce travail.

b. Variables de décision :

Les variables sont regroupées dans le vecteur *x []*. C'est en faisant varier ce vecteur que l'on recherche un optimum de la fonction *f*

c. Minimum local :

On dit que f atteint en $\mathbf{x_0}$ un minimum local s'il existe un intervalle **I** de la forme] x0-**ε, x0+ε [**tel que $f(x_0) \leq f(x)$ $\forall x \in I$.
En outre si on a $f(x_0) < f(x)$ $\forall x \in I-\{x_0\}$, on dit qu'il s'agit d'un minimum local strict.

d. Minimum global :

On dit que la fonction numérique f atteint en x_0 un minimum global (sur son domaine de définition D) si $f(x_0) \leq f(x)$ $\forall x \in D$.

I.3.2 La classification des problèmes d'optimisation multi-objectifs : *[44]*

La classification des problèmes d'optimisation multi-objectifs se fait selon quelques critères nous modélisons cela dans le tableau qui suit :

Critère		*Classification*
Nombre de variables de décision	Une variable de décision	Mono-variable
	Plusieurs variables de décision	Muli-variable
Type de variables de décisions	***Nombre réel continu***	Continu
	Nombre entier	Entier ou discret
	Permutation sur un ensemble fini de nombres	Combinatoire
Type de la fonction objectif	Fonction linéaire des variables de décisions	Linéaire
	Fonction quadratique des variables de décisions	Quadratique
	Fonction non-linéaire des variables de décisions	Non- linéaire
Formulation du problème	***Avec contrainte(s)***	Avec contrainte(s)
	sans contraintes	sans contraintes

Figure III.1 : Classification des problèmes d'optimisation multi-objectifs

I.3.3 L'approche Pareto : *[45]*

Lorsqu'on résout un problème d'optimisation multiobjectif, nous obtiendrons une multitude de solutions. Seul un nombre restreint de ces solutions va nous intéresser. Pour qu'une solution soit intéressante, il faut qu'il existe une relation de **dominance** entre la solution considérée et les autres solutions.

I.3.3.1 La dominance au sens de pareto :

a. Définition : *[44] [45]*

Les approches Pareto utilisent la notion de dominance pour comparer les solutions et leur affecter un score ou sélectionner des solutions. Ces approches ont connu un important développement en conjonction avec les algorithmes évolutionnistes à population à partir de la seconde moitié des années 90. Elles sont devenues la principale approche employée pour résoudre les problèmes multi-objectifs du fait de leur capacité à trouver un ensemble potentiellement efficace via la recherche menée sur une population de solutions.

La dominance Pareto ou l'optimalité de Pareto est défini comme suit :

Une solution ***x*** **domine** (≤) au sens de Pareto une solution ***z*** si et seulement si quel que soit i appartenant à {1..n} tel que $f_i(x) \leq f_i(z)$ et il existe i appartenant à {1..n} tel que $f_i(x) < f_i(z)$.

On dit qu'une solution x **domine fortement** une solution z si et seulement si quel que soit i appartenant à {1..n} tel que $f_i(x) < f_i(z)$.

On dit que les deux solutions sont **incompatibles** si et seulement si quel que soit i appartenant à {1..n} tel que ni $f_i(x) < f_i(z)$, ni $f_i(x) > f_i(z)$.

La représentation de la dominance Pareto (Pareto optimal) dans l'espace objectif est appelée front Pareto ; son allure prend des formes différentes selon que les objectifs doivent être minimisés ou maximisés, nous schématisons cela dans la figure suivante avec deux fonctions objectives.

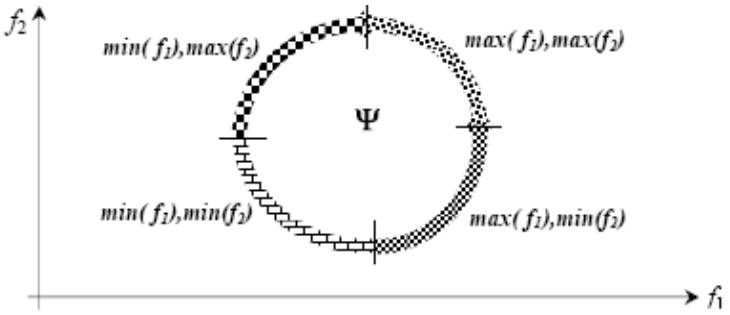


Figure III.2 : Représentation du front Pareto selon tous les cas possibles d'optimisation.

Selon le travail que nous effectuons, le front Pareto que nous obtenons prend la forme de la courbe min (f1), min (f2). Avec min (f1) est la minimisation de la consommation énergétique, et min (f2) est la minimisation de la charge de travail. Les deux fonctions objectifs f1 et f2 sont définies précédemment.

b. Mécanisme de sélection Pareto (Ranking) : *[46]*

La méthode de ranking attribue aux solutions non-dominées de la population courante le meilleur score dite valeur d'adaptation. Puis les solutions, qui ne sont dominées que par les solutions potentiellement efficaces, reçoivent le second meilleur score et ainsi de suite. De cette manière, la population est organisée en couches où chaque couche contient des solutions non comparables entre elles. Le ranking d'une population est illustré dans la figure ci-dessous. Dans cet exemple, les solutions A, B, C, D reçoivent le rang 1 car elles sont dominées par aucune solution. Les solutions E, F et G ont le rang 2 car elles ne sont dominées que par des solutions de rang 1. Enfin, H est affecté au rang 3. Cette fitness sera utilisée dans l'étape de sélection de l'algorithme (c'est ce mécanisme de sélection Pareto qui offre une alternative efficace aux algorithmes évolutionnaires de s'adapter facilement au cas multiobjectif). Cette méthode de ranking a ainsi été utilisée dans les Algorithmes génétiques pour la résolution de plusieurs problèmes, un exemple classique est NSGAII.

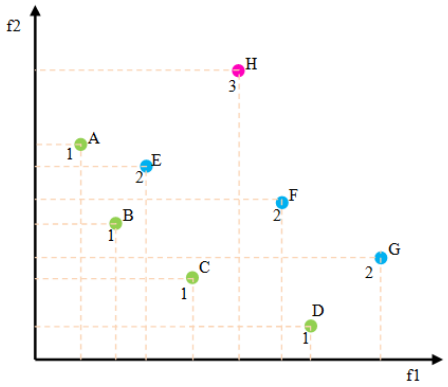


Figure III.3 : exemple de ranking

- **Présentation du NSGAII (Non-dominated Sorting Genetic Algoriyhm II):** ***[46]***

Non-dominated Sorting Genetic Algorithm ou Algorithme génétique de tri sélectif non-dominé est donc un algorithme d'optimisation multiobjectif (MOO pour Multiple Objective Optimization) et il est une instance d'un algorithme évolutionnaire du domaine du calcul évolutif. NSGA est une extension de l'Algorithme génétique pour l'optimisation des fonctions multi objectifs. Il est lié à d'autres algorithmes évolutionnaires d'optimisation multi objectifs (EMOO pour Evolutionary Multiple Objective Optimization Algorithms) (ou bien Algorithmes évolutionnaires multiobjectifs MOEA pour Multiple Objective Evolutionary Algorithms) tels que :

- L'algorithme génétique évalué par vecteur (VEGA pour Vector-Evaluated Genetic Algorithm),
- L'algorithme évolutif Pareto de la force (SPEA pour Strength Pareto Evolutionary Algorithm)
- La Stratégie d'évolution archivée de Pareto (PAES pour Pareto Archived Evolution Strategy).

Dans la deuxième version de NSGA [Deb, 2000] certaines résolutions et améliorations ont été apportées par rapport à toutes les critiques faites sur NSGA : complexité, non élitisme et utilisation de sharing. NSGAII intègre un opérateur de sélection, basé sur un calcul de la distance de *crowding*, très différent de celui de NSGA.

NSGA-II est un algorithme élitiste n'utilisant pas d'archive externe pour stocker l'élite. Pour gérer l'élitisme, NSGAII assure qu'à chaque nouvelle génération, les meilleurs individus rencontrés soient conservés. Pour comprendre le fonctionnement de l'algorithme, plaçons-nous à la génération *t*. Comme le montre la figure, deux populations (P_t et Q_t de taille N) coexistent. La population P_t contient les meilleurs individus rencontrés jusqu'à la génération *t*, et la population Q_t est formée d'individus autres, issus des phases précédentes de l'algorithme.

a. L'élitisme :

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations de croisement et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions. Actuellement, les algorithmes élitistes obtiennent de meilleurs résultats sur un grand nombre de problèmes multiobjectifs.

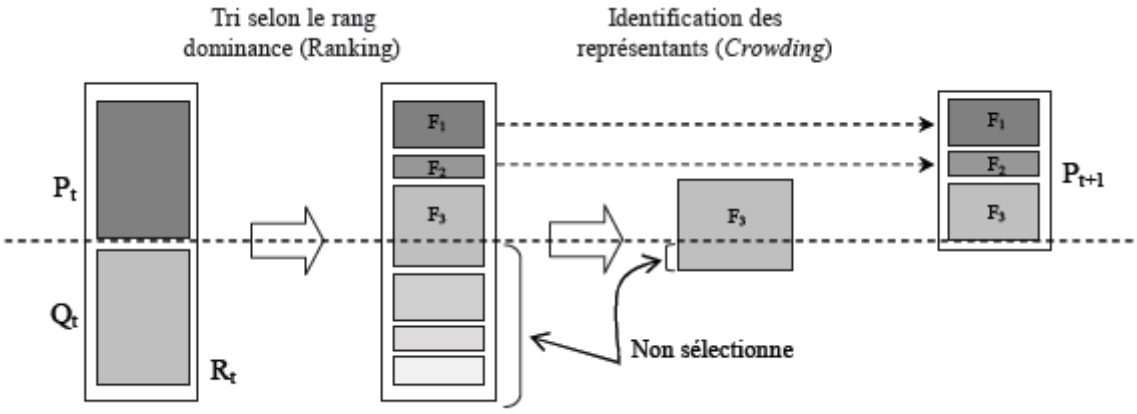


Figure III.4 : Schéma de fonctionnement de NSGAII

Un algorithme qui représente le fonctionnement de NSGAII pour minimiser une fonction de consommation énergétique est donné ci-bas.

SelectParentsByRankAndDistance et *SortByRankAndDistance* sont deux fonctions intégrées qui permettent de gérer la population dans une hiérarchie de fronts de Pareto non-dominés.

La *CrowdingDistanceAssignment* calcule la distance moyenne entre tous les membres du front Pareto [Deb et al Deb2002]. Nous le verrons détaillé plus loin.

La *CrossoverAndMutation* est une fonction qui effectue le croisement et la mutation des algorithmes génétiques.

Intput: *PopulationSize, ProblemSize, Pcrossover, Pmutation*
Output: offspring
Population ←InitializationPopulation (*PopulationSize, ProblemSize*)
EvaluationAgainstObjectiveFunctions (*Population*)
EvaluateConstraints (*Population*)
FastNondominatedSort (*Population*)
Selected ← SelectParentsByRank (*Population, PopulationSize*)
Offspring ← CrossoverAndMutation (*Selected, Pcrossover, Pmutation*)
While (¬ StopCondition ( ))
EvaluationAgainstObjectiveFunctions (Offspring)
EvaluateConstraints (Offspring)
Union ← Merge (*Population*, Offspring)
Fronts ← FastNondominatedSort (Union)
Parents ←Ø
Fronts←Ø

For (Front*i* Є Fronts)
    CrowdingDistanceAssignment (Front*i*)
    If (Size (Parents) +Size (Front*i*) > *PopulationSize)*
        Front← i
    Else
        Parents← merge (Parents, Front*i*)
    End
    End
    If (Size (Parents) < PopulationSize*)*
        Front ← SortByRankAndDistance (Front)
        For (P1 to P*PopulationSize)*
            Parents ← P*i*
        End
    End
Selected ← SelectedParentByRankAndDistance (Parents, *PopulationSize*)
Population ← Offspring
Offspring ← CrossoverAndMutation (Selected, *Pcrossover, Pmutation)*
End
Return (Offspring)

Le calcul de valeur d'adaptation pour NSGAII ne sert pas uniquement pour la sélection des opérateurs de croisement et de mutation, mais intervient aussi dans la sélection des individus à inclure dans *Pt+1* (la population contenant les élites). C'est donc une phase importante pour laquelle les auteurs de NSGAII ont développé une méthode particulière : la distance de crowding.

b. Calcul de la distance de crowding : *[46]*

La distance de *crowding di* d'un point particulier *i* se calcule en fonction du périmètre de l'hypercube ayant comme sommets les points les plus proches de *i* sur chaque objectif. Sur la figure en dessous est représenté l'hypercube en deux dimensions associé au point *i*. Un algorithme de calcul de la distance de *crowding* est détaillé dans [Deb, 2001]. Cet algorithme est de complexité O (MN log(N)), où M est le nombre d'objectifs du problème et N le nombre d'individus à traiter. Une fois tous les *di* calculés (pour chaque critère c, dc (xi) = d (xi, xi−1) + d (xi, xi+1)), il ne reste plus qu'à les trier par ordre décroissant et à sélectionner les individus possédant la plus grande valeur de *crowding*.

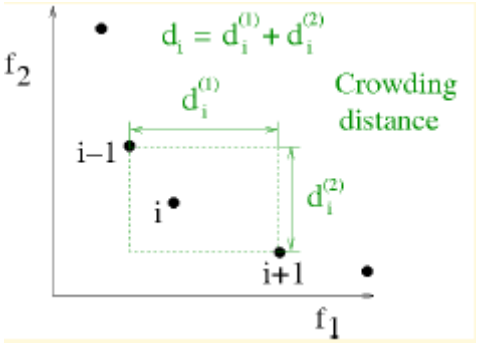


Figure III.5 : calcul d'un représentant (crowding)

Formulation du problème :

Pour résoudre le problème du placement d'une VM dans une PM nous avons opté à l'optimisation de deux fonctions objectif qui sont la minimisation de l'énergie et la charge du travail des serveurs qui sont décrites ci-dessous.

- ***Minimisation de l'énergie :***

La fonction utilisée pour minimiser l'énergie dans un centre de donnée est égal à la somme de l'énergie consommée par chaque machine virtuelle + le coup de migration si cette dernière existe * l'indice de la machine virtuelle considérée.

$$\min Obj_{bMatching} = \sum_{e \in E, e=(v,j)} (l_e + M_e \mathbb{1}_{ij}) x_e$$

Où:

l e: énergie consommée par la VM v sur le serveur j.

M e: cout de déplacement ou de migration (en termes d'énergie) de la VM v sur la PM j.

(M e =alpha*DT+beta) tel que DT représente la quantité totale de données transmises lors d'une migration. Alpha et beta sont des paramètres à déduire depuis des mesures énergétiques réelles Alpha=0.512 et beta=20.165.

L ij =1 si la VM i n'est pas sur le serveur j et 0 sinon.

x e: l'indice de la machine virtuelle v sur le serveur j.

- ***La charge du travail :***

  La fonction utilisée pour minimiser la charge du travail égal à la somme de la charge du chaque PM divisé par la fréquence de celle-ci – la charge moyenne (égal à la charge de toute les PM/par la fréquence de toutes ses dernières).

$$\sum_{j=0}^{P-1} abs\left(\frac{load(PE_j)}{f(PE_j)} - M\right)$$

Load (PEj) : la charge du serveur PEj

M: la charge moyenne qui est calculé par :

$$M = \frac{\sum_{j=0}^{P-1} load(PE_j)}{\sum_{j=0}^{P-1} f(PE_j)}$$

f (*PEj*) : fréquence du serveur *PEj*.

Pour placer une VM dans une PM, nous avons spécifié un ensemble de contraintes qui sont définies ci-dessous :

1. Une machine virtuelle ne peut pas être exécutée sur deux machines physiques différentes (Chaque VM doit être affecté à un seul serveur).

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i = 1, \ldots, n$$

   (M: nombre de serveurs, n: nombre de machine virtuelles)

2. La somme totale des CPUs des machines virtuelles exécutées sur une machine physique doit être inférieure à la CPU de cette machine physique.

$$\sum_{i=1}^{n} cpu_i x_{ij} \leq CPU_j e_j$$

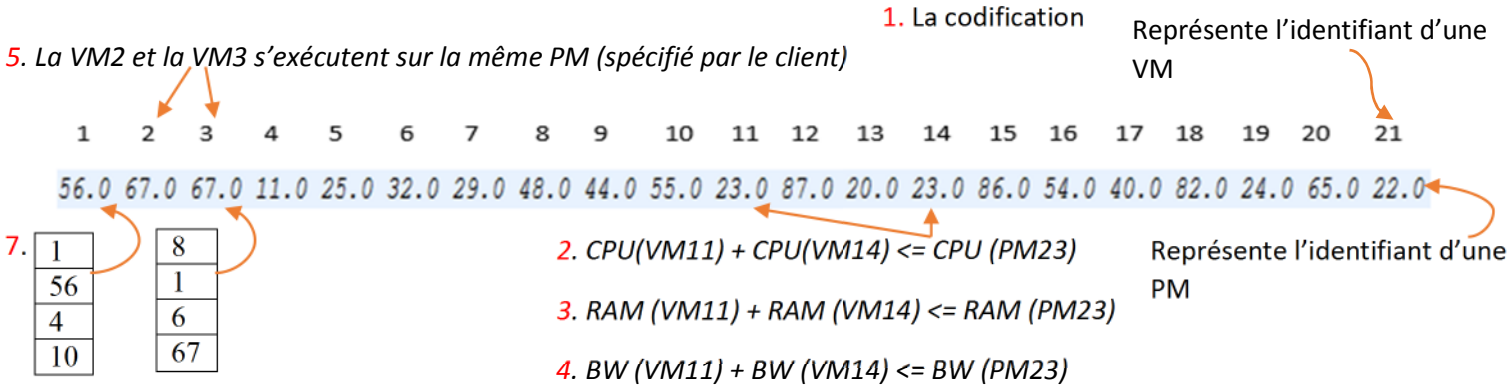
3. La capacité mémoire de la machine physique doit être supérieure à la somme de toutes les mémoires des machines virtuelles qu'elle détient.

$$\sum_{i=1}^{m} MEM_i x_{ij} \leq MEM_j e_j$$

4. La bande passante de la machine physique doit être supérieure à la somme de toutes les bandes passantes des machines virtuelles qu'elle détient.

$$\sum_{i=1}^{m} BW_i x_{ij} \leq BW_j e_j$$

5. Si l'utilisateur demande que l'emplacement de ses VM doivent être le même (ils s'exécutent tous sur la même PM) alors ses VM ne doivent pas se retrouvé dans des PM différentes.
6. Si l'utilisateur demande que l'emplacement de ses VM doivent être diffèrent (ils s'exécutent tous sur des PM différentes) alors ses VM ne doivent pas se retrouvé dans les même PM.
7. Respecter la contrainte SLA (on a spécifié un ensemble de machine physique pour chaque VM et les VM de la solution obtenu ne doivent pas se retrouver en dehors de son ensemble de PM).



Conclusion :

Dans ce chapitre nous avons vu l'approche utilisée pour résoudre le problème de placement de machines virtuelles dans une PM tout en minimisant l'énergie et la charge du travail des serveurs.

Dans le chapitre qui suit nous allons présenter les outils de développement utilisés pour la modélisation et la résolution de notre problème, nous verrons ensuite une analyse des résultats obtenus après l'application de l'algorithme d'optimisation pour la minimisation de la consommation d'énergie et la régulation de la charge du travail.

Chapitre IV :

Réalisation

Introduction :

Cette partie constitue le dernier volet de ce mémoire. Après avoir terminé les phrases d'analyse et de conception, la réalisation est la partie importante consacrée à la présentation du projet et son environnement et développement.

En premier lieu nous présentons les outils utilisés dans le développement de notre projet. Nous passerons ensuite à une analyse des résultats obtenus après l'exécution de notre travail.

I. Outils de développement :

I.1 Présentation d'Excel : *[47]*

Excel est programme informatique auquel nous avons fait référence pour enregistrer les caractéristiques de nos machines virtuelles et serveurs.

	A	B	C	D	E
1	vmId	Mips	RAM	Bandwidtl	Energie
2	1	1	0,5	1	10
3	2	1	1	1	10
4	3	1	2	1	10
5	4	2	4	1	20
6	5	2	8	1	30
7	6	4	16	1	30
8	7	1	1	1	10

Figure IV.1 : Caractéristique des VM

	A	B	C	D	E	F
1	pmId	Mips	RAM	Bandwidth	frequence	Energie
2	1	16	64	10	3,85	5000
3	2	14	32	10	3,5	5000
4	3	10	28	8	2	5000
5	4	60	1024	10	2,8	5000
6	5	16	128	10	2,1	5000
7	6	14	32	10	3,5	5000
8	7	30	512	10	3,49	5000

Figure IV.2 : Caractéristique des PM

I.2 Présentation de l'oracle: *[48]*

Oracle est un système de gestion de base de données relationnel le plus utilisé dans le monde de l'informatique. Nous l'avons intégré dans notre travail pour la sauvegarde des résultats de la meilleure solution obtenue.

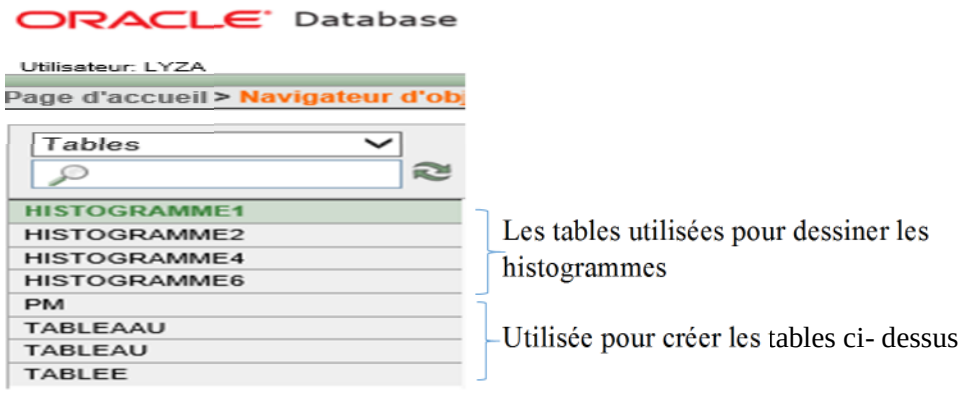


Figure IV.3 : tables de la BDD

I.3 Présentation du langage de programmation (JAVA) : *[49]*

Java est un langage de programmation informatique orienté objet, sa particularité principale est qu'il est facilement portable sur différent système d'exploitation.

I.4 Présentation de l'environnement de développement (eclipse): *[50]*

Eclipse est un environnement de développement intégré, libre, extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

I.5 Présentation du Framework(Jmetal) : *[51]*

JMetal signifie Algorithmes Metaheuristiques en Java, et c'est un Framework orienté objet Java pour résoudre des problèmes d'optimisation multi-objectifs (MOP) avec des techniques métaheuristiques. JMetal fournit un ensemble riche de classes qui peuvent être utilisées comme éléments constitutifs de techniques multi-objectifs; de cette façon, en profitant de la réutilisation des codes, les algorithmes partagent les mêmes composants de base, tels que les implémentations d'opérateurs génétiques et les estimateurs de densité, facilitant ainsi non seulement le développement de nouvelles techniques multi-objectifs, mais aussi la réalisation de différents types d'expériences.

L'inclusion d'un certain nombre d'algorithmes classiques et à la fine pointe de la technologie, de nombreux problèmes habituellement inclus dans les études de performance et un ensemble d'indicateurs de qualité permettent non seulement aux nouveaux arrivants d'étudier les principes de base de l'optimisation multi-objectifs avec des métaheuristiques mais aussi leurs application pour résoudre les problèmes du monde réel.

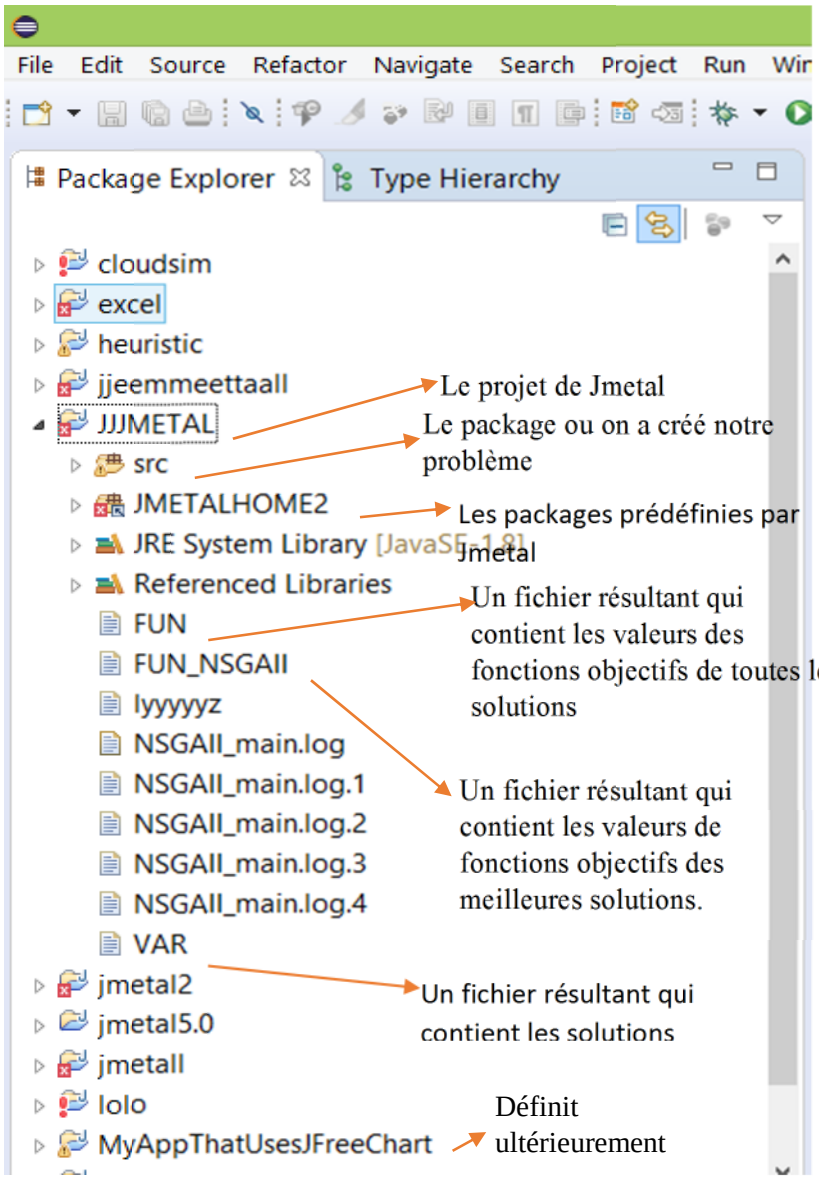


Figure IV.4 : Package de jmetal sous eclipse

1. C'est la classe ou le client interagit sur le placement des VM
2. Permet la connexion à la
3. Effacer les BDD pour une prochaine utilisation
4. Classes qui permettent d'importer les fichiers excel
5. La classe de notre problème
6. Ces classes permettent de dessiner l'histogramme pour choisir la solution moyenne

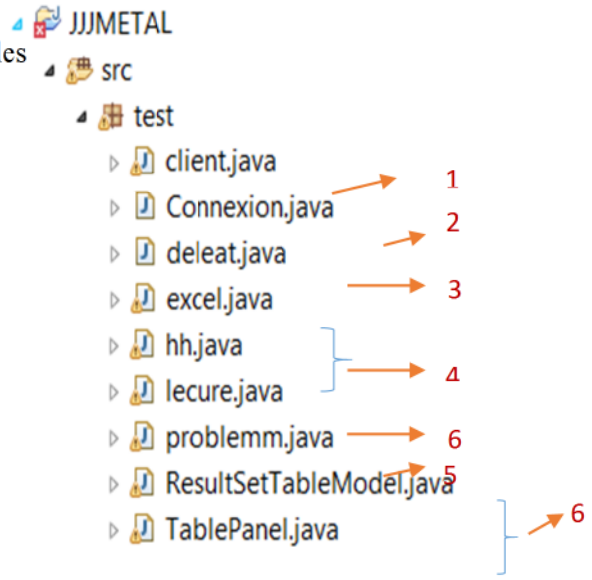


Figure IV.5: ensemble des classes

a. La structure des packages du Jmetal : [51]

Jmetal est composé de six packages qui sont représentés dans la **Figure IV.6,** ces packages sont : core, problems,metaheuristics, operators, encodings, qualityIndicators, util et experiments que nous décrivons brièvement ci-dessous :

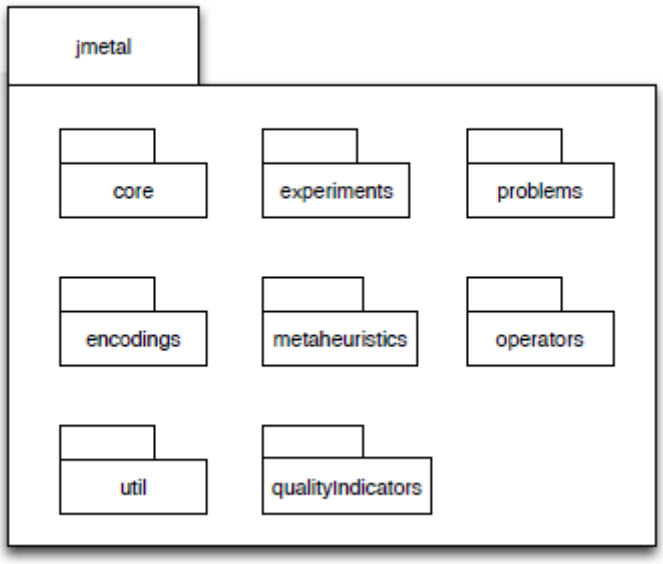


Figure IV.6 : Packages de Jmetal

Package core : ce paquet contient les ingrédients de base à utiliser par les métaheuristiques développées sous jMetal. Les principales classes de ce package sont: Algorithme, operator, problem, variable, Solution, SolutionSet et SolutionType.

Package problems**:** Tous les problèmes disponibles dans jMetal sont inclus dans ce package. Ici, nous pouvons trouver les problèmes avec contraintes (water, Srinivas,etc**)** et les problèmes sans contraintes comme (Kursawe, Schaffer et les problèmes de Benchmark (ZDT,WFG,etc),etc.).

Package Métaheuristics: Ce paquet contient les métaheuristiques mises en œuvre dans jMetal. La liste des algorithmes comprend NSGA-II, SPEA2, PAES, PESA-II, GDE3, FastPGA, MOCell, AbYSS, OMOPSO, IBEA et MOEA / D. Bien que Jmetal soit destiné à l'optimisation multi-objectifs, un certain nombre d'algorithmes à objectif unique sont inclus dans le package jmetal.metaheuristics.singleObjective.
L'algorithme utilisé pour résoudre notre problème c'est le ***NSGAII*** (détaillé dans le chapitre3).

Package jmetal.operators: Ce paquet contient différents types d'objets opérateurs, y compris les opérateurs de croisement, de mutation, de selection et de recherche locale. Nous donnons ensuite un exemple d'opérateur de chaque type:

jmetal.operators.crossover.SBXCrossover: Ce comparateur prend également deux solutions S1 et S2 et effectue un croisement binaire simulé (SBX), en retournant les deux fils obtenues.

jmetal.operators.mutation.Polynomial: les opérateurs de mutations sont généralement appliqués à des solutions uniques, en les modifiant en conséquence, et ils renvoient la solution mutée. Dans ce cas, l'opérateur est une mutation polynomiale.

jmetal.operators.selection.BinaryTournament: les comparateurs de sélection prennent habituellement comme paramètre un ensemble de solutions, renvoyant une solution selon un critère. En particulier, cet opérateur applique un tournoi binaire.

jmetal.operators.localSearch.MutationLocalSearch: ces opérateurs sont destinés à appliquer des stratégies de recherche locales à une solution donnée. La MutationLocalSearch, utilisée dans l'algorithme AbYSS, requiert comme paramètres une solution, un opérateur de mutation,un nombre entier N et un objet jmetal.base.archive; alors l'opérateur de mutation est appliqué itérativement pour améliorer la solution pendant N rounds et l'archive est utilisée pour stocker les solutions non proposées.

Package jmetal.encodings: Ce paquet contient les représentations de variables de base et les types de solution inclus dans le cadre. Dans jMetal version 4.0, les classes suivantes sont incluses:

Variables: Binaire, BinaryReal, BinaryReal (code codé binaire), Int, Permutation, ArrayInt et ArrayReal.

Types de solution: BinarySolutionType, RealSolutionType, BinaryRealRelocalType, IntSolutionType, PermutationSolutionType, ArrayRealSolutionType, ArrayIntSolutionType, IntRealSolutionType (combine les variables Int et Real) et ArrayRealAndBinarySolutionType (combine les variables ArrayReal et Binary).

Package util: Un certain nombre de classes d'utilitaires sont incluses dans cette classe, en tant que générateur de nombres pseudo-aléatoires, différents types d'archives, une classe de quartier à utiliser dans des algorithmes évolutifs cellulaires, des comparateurs, etc.

Package qualityIndicators: Pour évaluer la performance de métaheuristiques multi-objectif, un certain nombre d'indicateurs de qualité peuvent être appliqués. Le paquet contient actuellement six indicateurs:

Distance de génération
Distance de génération inversée
Epsilon additif
spread
spread généralisée
Hypervolume

Package expériences: ce package contient un ensemble de classes destinées à réaliser des études typiques d'optimisation multi-objectifs.

b. *Quelques classes de jmetal :*

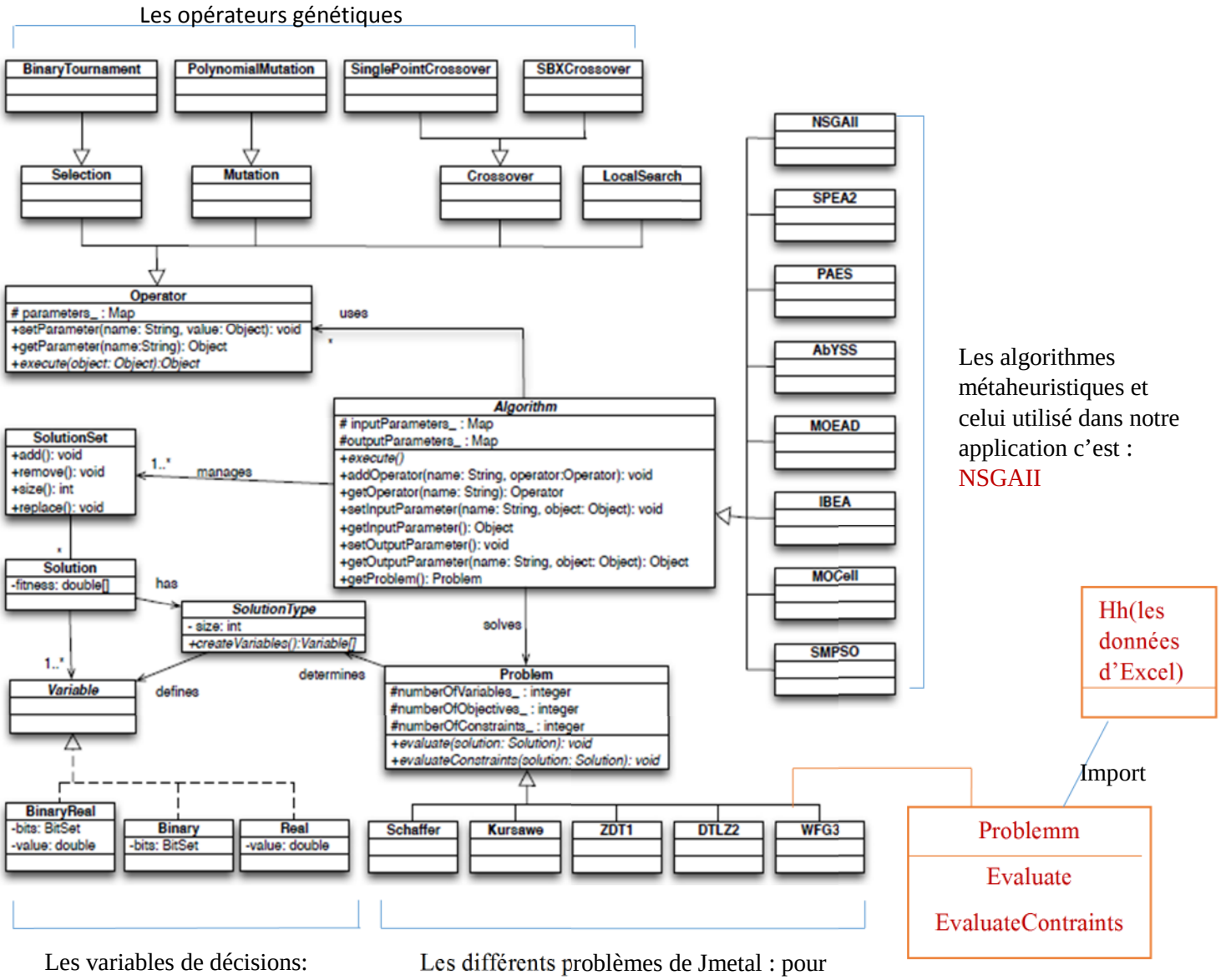


Figure IV.7 : Architecture de Jmetal.

a. *Classe problem :*

Dans jMetal, tous les problèmes héritent de la classe problem. Cette classe contient deux méthodes de base: evaluate () et evaluateConstraints (). Les deux méthodes reçoivent une solution représentant une solution candidate au problème; le premier l'évalue et le second détermine la contrainte globale de violation de cette solution. Tous les problèmes doivent être définis dans la méthode evaluate (), alors que seuls les problèmes ayant des contraintes latérales doivent être évalués dans evaluatecontraintes ().Une caractéristique de conception clé dans jMetal est que le problème désigne les types de solutions autorisées qui conviennent pour le résoudre.

La **Figure IV.8** montre le code utilisé pour mettre en œuvre notre problème. Comme nous pouvons l'observer, il étend le problème de classe *(ligne 34).* Ensuite, une méthode de constructeur est définie pour créer des instances de ce problème *(lignes 76)*, qui comporte deux paramètres: une chaîne contenant un identifiant de type de solution et le nombre de variables de décision du problème. En règle générale, tous les problèmes devraient avoir comme premier paramètre la chaîne indiquant le type de solution. Les caractéristiques de base du problème (nombre de variables, nombre d'objectifs et nombre de contraintes) sont définies dans les *lignes 77 à 80*. Les limites des valeurs des variables de décision sont définies dans les *lignes 83-88*. Les *lignes 91-95* sont utilisées pour spécifier que les représentations de solution autorisées sont codées en arrayréel, de sorte que les objets SolutionType correspondants sont créés et affectés à une variable d'état. Après le constructeur, la méthode evaluate () est redéfinie *(lignes 127-296),* Dans cette méthode, après avoir calculé les deux valeurs de fonctions objectives, elles sont stockées dans la solution en utilisant la méthode setObjective de Solution *(lignes 264 et 265).* La méthode evaluateContraints () est redéfinie *(lignes 299-493),* Dans cette méthode, après avoir calculé toute les contraintes, elles sont stockées dans la solution en utilisant la méthode setOverallConstraintViolation de Solution *(ligne 462).*

```
1  package test;
2  import java.io.File;
33
34 public class problemm extends Problem {
35       hh d= new hh();
36          Object[][] o= d.calcul();
37          Object[][] PM =d.calcul_PM();
38          int i=0,ii=0;
39          int j=0;
40
41           client cl= new client();
42          public int nbr_pm=cl.saisir1();
43
44
45          public int numberOfVariable=nbr_pm;
46                float facteur_equilibre;
47                float pourcentage;
48                float somi=0;
49
50 ////////////////////////////////////////////////////////
51
52                 static Connexion con1 = new Connexion();
53                 static Connexion con2 = new Connexion();
54                 static Connexion con3 = new Connexion();
55                 static Connexion con4 = new Connexion();
56                static Connexion con5 = new Connexion();
57                static Connexion con6 = new Connexion();
58                static Statement stat1,stat2, stat3, stat4,stat5,stat6;
59                static Connection conn;
60                ////////////////////////////////////////////
61                NSGAII_main val=new NSGAII_main();
62                  int val_vm=val.nbr_vm;
63                  ////////////////////////////////////
64          client cl1= new client();
65
66          //String exig=cl.exig();
67          String mem_diff=cl1.mem_diff();
68          Scanner sc = new Scanner(System.in);
69          int vect_vm[]={1,2};
70 ////////////////////////////////////////////////////////
71         int nbr=numberOfVariables_;
72             int[][] matrice ;
73
74
75
76   public problemm(String solutionType, Integer numberOfVariables) {
77     numberOfVariables_   = numberOfVariables;
78     numberOfObjectives_ =  2;
```

Pour importer les caractéristiques des VM et des PM

Pour saisir le nombre de serveur à utiliser

Connexion à la BDD

Pour saisir le nombre de VM à utiliser

Vecteur ou se trouvent les VM qui doivent être dans la même PM ou dans des PM différentes

Nombre de fonction objectif utilisée.

Nombre de contrainte utilisée

```
80      numberOfConstraints_=  5;
81      problemName_         = "problemm";
82
83      upperLimit_  = new double[numberOfVariables_];
84      lowerLimit_  = new double[numberOfVariables_];
85
86      for (int var = 0; var < numberOfVariables_; var++){
87        lowerLimit_[var] = 1;
88        upperLimit_[var] = numberOfVariable;
89      } // for
90
91      if   (solutionType.compareTo("ArrayReal") == 0)
92          solutionType_ = new ArrayRealSolutionType(this) ;
93      else {
94          System.out.println("Error: solution type " + solutionType + " invalid") ;
95          System.exit(-1) ;
96      }
97      matrice= new int[numberOfVariables_][nbr+1];
98      for(int i = 0; i < numberOfVariables_; i++){
99          matrice[i] = new int[numberOfVariables_];
100     }
101     for(int i = 0; i < numberOfVariables_; i++){
102         for(int j = 0; j < matrice[i].length; j++){
103             matrice[i][j] = PseudoRandom.randInt(1,numberOfVariable);    }
104     }
105     for(int i = 0; i < numberOfVariables_; i++){
106         for(int j = 0; j < matrice[i].length; j++){
107             System.out.print(matrice[i][j] + " ");
108         }
109         System.out.println();    }
110
111     ////////////////////////////////////////////////////////////////
112     /*    for( int i = 0; i<100;i++){
113           System.out.print("\n");
114          for( int j = 0;j<10;j++){
115               System.out.print(o[i][j]+"\t\t\t");
116           }}*/
117     /*    System.out.print("**********************************");
118     for( int ii = 0; ii<100;ii++){
119       System.out.print("\n");
120      for( int jj = 0;jj<10;jj++){
121           System.out.print(PM[ii][jj]+"\t\t\t");
122       }}
123  */
124   } //problemm
125 @Override
```

Les valeurs limites [1..nbr_PM] utilisée pour la codification de la solution

Chaque ligne de cette matrice représente le vecteur d'une VM qui contient les valeurs des PM qu'il doit avoir

```
127 public void evaluate(Solution solution) throws JMException {
128     //DECLARATION
129      XReal vars = new XReal(solution) ;
130      //CPU
131       int cpuvm[]=  new int[numberOfVariables_+1] ;
132       int cpupm[]=  new int[numberOfVariable+1] ;
133      //RAM
134       int RAMvm[]=  new int[numberOfVariables_+1] ;
135       int RAMpm[]=  new int[numberOfVariable+1] ;
136      //BANDE PASSANTE
137       int BWvm[]=  new int[numberOfVariables_+1] ;
138       int BWpm[]=  new int[numberOfVariable+1] ;
139      //FREQUENCE
140       float freqpm[]=  new float[numberOfVariable+1] ;
141      //charge
142       float []load=new float[numberOfVariables_+1];
143       int []contenuu=new int[numberOfVariables_+1];
144       int som=0;
145       float som_freq=0;
146       float somi_facteur=0;
147       float charge=0,charge_facteur=0;
148       //khalouta
149       int[]fils=new int[numberOfVariables_+1];
150       int[] vect=new int[numberOfVariables_];

151       //ENERGIE
152       float energie=0;
153       double alpha=0.512;
154       double beta=20.165;
155       int Evm[]=  new int[numberOfVariables_+1] ;
156       double migration;
157       double DT=150;
158 ////////////////////////////////////////////////////////////
159       //CHARGER EXCEL VM
160       for (int i = 1 ; i <=numberOfVariables_; i++){
161             cpuvm[i] = (int) (double)o[i][1]  ;
162             RAMvm[i] = (int) (double)o[i][2]  ;
163             BWvm[i] = (int) (double)o[i][3]  ;
164             Evm[i]= (int)(double)o[i][4] ;
165             }
166       //CHARGER EXCEL PM
167       for (int i = 1 ; i <=numberOfVariable; i++){
168             cpupm[i] = (int)(double) PM[i][1] ;
169             RAMpm[i] = (int)(double) PM[i][2] ;
170             BWpm[i]  = (int)(double) PM[i][3] ;
171             freqpm[i] =(float)(double) PM[i][4] ;
172           }
173 ////////////////////////////////////////////////////////////
```

```
174   // calculer M
175      ////CALCULER LA CHARGE DE TOUT LES PM
176      for (int ii = 1 ; ii <=numberOfVariables_; ii++){
177          load[ii]=cpuvm[ii];
178          som+=load[ii];
179         // System.out.println("\n somme CPU_VM de tout le vecteur(on l'a besoin pour calculer M)\t"+som);
180      }
181      ////  CALCULER LA FREQUENECE DES PE QUI A UTILISEE
182     ////////////////////////////////////////////////////////////////////////////////////////////////////////////
183      for(int k1=1;k1<=numberOfVariables_;k1++){  //indice du vecteur ya3ni VM
184          contenuu[k1-1]=(int) vars.getValue(k1-1) ;   //contenu du vecteur ya3ni PM
185      }
186      for(int k1=1;k1<=numberOfVariables_;k1++){  //indice du vecteur ya3ni VM
187          int conten=(int) vars.getValue(k1-1)  ;    //contenu du vecteur ya3ni PM
188
189    for(int i2=k1;i2<=numberOfVariables_;i2++){
190      if (conten==contenuu[i2]){
191          contenuu[i2]=0;}
192      }
193      if (contenuu[k1-1]!=0){
194          som_freq+= freqpm[contenuu[k1-1]];
195          //System.out.println("freqpm[\t"+contenuu[k1-1]+"]\t\t"+freqpm[contenuu[k1-1]]);
196          //System.out.println("somme des frequences de tout les PM(on l'a besoin pour calculer M)\t"+som_freq);
197      }
198         }
199              float M=som/som_freq;
200              //System.out.println("M\t"+M);
201 ////////////////////////////////////////////////////////////////////
202         //CALCULER LA VARIANCE
203     /*************************/
204
205       //REMPLIR LE TABLEAU VECT POUR VERIFIER LA REDENDANCE
206        for (int j = 0 ; j <numberOfVariables_; j++){
207                    vect[j]=(int) vars.getValue(j)  ;  //contenu du vecteur ya3ni PM
208              } //for j
209 ////////////////////////////////////////////////////////////////////
210        for(int k=1;k<=numberOfVariables_;k++){  //indice du vecteur ya3ni VM
211           int contenu=(int) vars.getValue(k-1)  ;   //contenu du vecteur ya3ni PM
212           fils[k]=contenu;
213           int t=vect[k-1];
214           if(t!=0){
215               /*energie*/
216               int Somme_Evm=Evm[k];
217              // System.out.println("Somme_Evm=Evm[i]\t"+Somme_Evm);
218               /*charge*/
219              int load_cpu= cpuvm[k];
220               for (int m=k+1 ; m <numberOfVariables_+1; m++){
221                   if((vect[k-1]==vect[m-1])&& (vect[m-1]!=0)){
222                       /*energie*/
223                       Somme_Evm+=Evm[m];
224                  //    System.out.println("Evm\t"+Evm[m]+"\t"+Somme_Evm);
225                       /*charge*/
226                       load_cpu+= cpuvm[m];
227                       vect[m-1]=0;
228                      // System.out.println("fin ><<de calcul de somme");
229                    }//if
230                }//for m
231 /*****************************************************************/
232               //////C POUR L'ENERGIE
233                //System.out.println("\n Somme_Evm["+contenu+"]=\t"+Somme_Evm+"\t\t");
234                migration =alpha*DT+beta; //la quantité des données transmises
235                //System.out.println("migration\t\t"+migration );
236                 if (migration==0){
237                  //  System.out.println("cas avec migration\t\t");
238                    energie = (float) ((Somme_Evm + migration*1)*k);
239                    //System.out.println("energie\t\t"+energie);
240                }
241                else {
242                    //System.out.println("cas sans migration\t\t");
243                    energie = (float) ((Somme_Evm + migration*0)*k);
244                    //System.out.println("energie\t\t"+energie);
245                }
```

Calculer la charge moyenne

```
246 /*****************************************************************/
247              /////C POUR LA CHARGE
248             //System.out.println("somme["+contenu+"]=\t"+load_cpu);
249             charge=Math.abs((load_cpu/freqpm[contenu])-M);
250             charge_facteur=(float) Math.pow(((load_cpu/freqpm[contenu])-M),2);
251             //System.out.println("charge\t"+charge);
252             somi+=charge;
253             //System.out.println("somi\t"+somi);
254             somi_facteur+=charge_facteur;
255
256         }  //t
257       }//for k
258       facteur_equilibre = nbr_pm / somi_facteur;
259       //System.out.println(" float facteur_equilibreeeeeeeeeeee"+facteur_equilibre);
260
261     //  System.out.println("energieeeee\t"+energie);
262     //  System.out.println("charge\t"+somi);
263
264       solution.setObjective(0,somi);     //CHARGE
265       solution.setObjective(1,energie); //ENERGIE
266
267 //*****************************************************************
```

Les deux fonctions objectifs à minimiser

Pour maximiser une fcts sous jmetal ; la fcts suivantes doit être utilisée :
Max(f(x)) = -Min(f(-x))car ce dernier minimise tjrs ses fonctions .

```
268 //AFFICHER LES FILS
269         for(int n=1;n<=numberOfVariables_;n++){
270                         System.out.print(fils[n]+"\t\t");
271                         }
272
273         ////////////////////////////////////////////////////,
274
275          System.out.println();
276          int compte=0;
277
278             for (int i=0;i<fils.length;i++)
279                 for (int j=i+1;j<fils.length;j++)
280                     if ((fils[i]==fils[j]))
281                         fils[j]=0;
282
283             for (int j=0;j<fils.length;j++)
284                     System.out.print(fils[j]+"\t");
285
286             for (int j=0;j<fils.length;j++)
287                 if (fils[j]!=0)
288                     compte+=1;
289
290             System.out.print("\n compte =\t"+compte);
292         pourcentage=(float)compte/(float)(nbr_pm);
293         System.out.print("\n pourcentage\t=\t"+compte+" /"+nbr_pm+"\t=\t"+pourcentage);
294
295     ////////////////////////////////////////////////////////////////////////////
296      }//evaluate
297 //-------------------------------------------------------------------
298
299 public void evaluateConstraints(Solution solution) throws JMException {
300         hh d= new hh();
301         Object[][] o= d.calcul();
302         Object[][] PM =d.calcul_PM();
303        System.out.println("EVALUATTION CONTRAINTE \t");
304        int g=0;
305      float CPU=0,RAM=0,BW=0;
306        double [] contraint = new double[numberOfConstraints_];
307          int[] vect=new int[numberOfVariables_];
308           XReal vars = new XReal(solution) ;
309           double tot=0.0;
310
311           float cpuvm[]=  new float[numberOfVariables_+1] ;
312           float cpupm[]=  new float[numberOfVariable+1] ;
313
314           float RAMvm[]=  new float[numberOfVariables_+1] ;
315           float RAMpm[]=  new float[numberOfVariable+1] ;
```

Pour calculer le pourcentage des serveurs utilisé

```
317         float BWvm[]=  new float[numberOfVariables_+1] ;
318         float BWpm[]=  new float[numberOfVariable+1] ;
319          int number=0;
320
321          for (int i = 1 ; i <=numberOfVariables_; i++){
322                cpuvm[i] = (float) (double)o[i][1]  ;
323                RAMvm[i] = (float) (double)o[i][2]  ;
324                BWvm[i] = (float) (double)o[i][3]  ;
325                }
326          for (int ii = 1 ; ii <=numberOfVariable; ii++){
327                cpupm[ii] = (float)(double) PM[ii][1] ;
328                RAMpm[ii] = (float)(double) PM[ii][2] ;
329                BWpm[ii] = (float)(double)  PM[ii][3] ;
330              }
331          for (int j = 0 ; j <numberOfVariables_; j++){
332                 vect[j]=(int) vars.getValue(j)  ;  //contenu du vecteur ya3ni PM
333          } //for j
334         for (int i=1 ; i <numberOfVariables_+1; i++){
335                 int contenu =(int) vars.getValue(i-1)  ;
336                 int t=vect[i-1];
337                 if(t!=0){
338                     float Somme_VM = cpuvm[i];
339                     float Somme_RAM=RAMvm[i];
340                     float Somme_BW=BWvm[i];

341                      for (int m=i+1 ; m <numberOfVariables_+1; m++){
342                          if((vect[i-1]==vect[m-1])&& (vect[m-1]!=0)){
343                              Somme_VM+= cpuvm[m];
344                              Somme_RAM+=RAMvm[m];
345                              Somme_BW+=BWvm[m];
346                              vect[m-1]=0;
347                           }//if
348                       }//for m
349                     System.out.println("somme_VM["+contenu+"]=\t"+Somme_VM);
350                     System.out.println("cpupm["+contenu+"]=\t"+cpupm[contenu]);
351
352                     System.out.println("sommeRAM["+contenu+"]=\t"+Somme_RAM);
353                     System.out.println("RAMpm["+contenu+"]=\t"+RAMpm[contenu]);
354
355                     System.out.println("sommeBW["+contenu+"]=\t"+Somme_BW);
356                     System.out.println("BWpm["+contenu+"]=\t"+BWpm[contenu]);
357
358                         CPU=cpupm[contenu]-Somme_VM ;
359                     //  contrain=cpupm[contenu]-Somme_VM ;
360                     //System.out.println("ccontraint0 pour les cpu\t"+contrain);
361                         System.out.println("ccontraint0 pour les cpu\t"+CPU);
362                         RAM=RAMpm[contenu]-Somme_RAM ;
363                         //  contrain=cpupm[contenu]-Somme_VM ;
364                         //System.out.println("ccontraint0 pour les cpu\t"+contrain);
```

```
                        System.out.println("ccontraint1 pour les RAM\t"+RAM);
                        BW=BWpm[contenu]-Somme_BW ;
                        System.out.println("ccontraint2 pour les BW\t"+BW);
                        System.out.println("");

// 1iere contrainte:::: la CPU
                    if (CPU<0){
                          contraint[0]+=CPU;
                          System.out.println("Y aaa uneee contrainteee CPU \t"+contraint[0]);
                          number++;
                    }
// 2ieme contrainte:::: la RAM
                    if (RAM<0){
                          contraint[1]+=RAM;
                          System.out.println("Y aaa uneee contrainteee RAM \t"+contraint[1]);
                          number++;
                    }


// 3ieme contrainte:::: la BANDE PASSANTE
                    if (BW<0){
                          contraint[2]+=BW;
                          System.out.println("Y aaa uneee contrainteee RAM \t"+contraint[2]);
                          number++;
                    }

      }//if
}//for i
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

//4ieme contrainte:::: l'emplacemet des VM

    ///////cas ou l'emplacement des VM est le meme
        if(mem_diff.equals("oui")){
                for(int ll=0;ll<vect_vm.length-1;ll++){
                    for(int l=ll+1;l<vect_vm.length;l++){
                    int cont1=(int)vars.getValue(vect_vm[ll]);
                    //System.out.println("vect_vm[ll]\t "+vect_vm[ll]+"cont1\t"+cont1);
                    int cont2=(int)vars.getValue(vect_vm[l]);
                    //System.out.println("vect_vm[l]\t "+vect_vm[l]+"cont2\t"+cont2);
                    if(cont1!=cont2){
                         contraint[3]+=-10;
                         System.out.println("ya une violation ds l'emplacement");
                    }
            }
                }
            }

    ///////cas ou l'emplacement des VM est different
        else if(mem_diff.equals("non")){
            for(int ll=0;ll<vect_vm.length-1;ll++){
                for(int l=ll+1;l<vect_vm.length;l++){
                int cont1=(int)vars.getValue(vect_vm[ll]);
                //System.out.println("vect_vm[ll]\t "+vect_vm[ll]+"cont1\t"+cont1);
                int cont2=(int)vars.getValue(vect_vm[l]);
                //System.out.println("vect_vm[l]\t "+vect_vm[l]+"cont2\t"+cont2);
                if(cont1==cont2){
                     contraint[3]+=-10;
                     System.out.println("ya une violation ds l'emplacement");
                }
                }
            }
        }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//5ieme contrainte:::: SLA
        //int nbr=PseudoRandom.randInt(1,numberOfVariable);
        int t2=0;
        boolean trouv=false;
        for (int t=0;t<numberOfVariables_;t++){
        int contenu_VM=(int)vars.getValue(t);

        System.out.println("contenu_VM\t"+contenu_VM);
        System.out.println("************");
```

```
438         while(trouv==false && t2<numberOfVariables_){
439             System.out.println("matrice["+t+"]["+t2+"]\t"+matrice[t][t2]);
440             if(matrice[t][t2]!=contenu_VM ){
441                 t2++;
442             }
443             else {trouv =true;}
444             }
445
446         if (trouv==false){
447              contraint[4]+=-10;
448                 System.out.println(" oulach");
449
450         }else {
451     System.out.println("il se trouve sur la ligne\t"+(t+1)+"\t"+"et sur la position de \t"+(t2+1));
452             }
453     }
454 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
455
456          for(int a=0;a<numberOfConstraints_;a++){
457              if(contraint[a]<0){
458                  tot+=contraint[a];
459                  System.out.println("Y aaa uneee contrainteeeeeee iciiiiii \t"+tot);
460                  g++;
461              }}
462          solution.setOverallConstraintViolation(tot);
463          System.out.println("solution.getOverallConstraintViolation() \t"+solution.getOverallConstraintViolation());
464          solution.setNumberOfViolatedConstraint(g++);
465          System.out.println("----------------------------------------");
466
467              if (solution.getOverallConstraintViolation() == 0.0) {
468                   String req6 = "Insert into TABLEAU values('" +(int)solution.getObjective(0)+ "','"
469               +(int)solution.getObjective(1)+"','"+facteur_equilibre+"')";
470
471                 System.out.println(" float facteur_equilibre"+facteur_equilibre);
472                 //////////////////////////
473                 String req8 = "Insert into TABLEAAU values('" +(int)solution.getObjective(0)
474                 + "','" +(int)solution.getObjective(1)+ "','"+pourcentage+"')";
475                 System.out.println("pourcentage"+pourcentage);
476                 ///////////////////////
477                 String req18 = "Insert into PM values('" +(int)solution.getObjective(0)
478                 + "','" +(int)solution.getObjective(1)+ "','"+nbr_pm+"')";
479                 ////////////////////
480                try {
481                    stat1 = con1.obtenirConnexion().createStatement();
482                      stat1.executeUpdate(req6);
483                      stat1.executeUpdate(req8);
484                      stat2 = con1.obtenirConnexion().createStatement();
485                      stat2.executeUpdate(req18);
486                  } catch (SQLException e) {
487                      // TODO Auto-generated catch block
488                      e.printStackTrace();
489                  }
490
491                 }
492
493   } // evaluateConstraints
494  } // problemm
```

L'insertion dans la base de données pour une future utilisation.

Figure IV.8 : classe de notre problème « problemm »

b. Classe Arrayreal (pour la codification de la solution) :

Comme nous avons définies précédemment que la solution est codée dans la classe arrayreal du package jmetal.encoding.variabale, la solution comme montré dans la **Figure IV.9** ci-dessous est un vecteur du taille numberOfvariables et le contenu est un intervalle des limites des valeurs des variables de décision déclaré dans la classe « problemm »

```
  public ArrayReal(int size, Problem problem)  {
    problem_ = problem;
    size_ = size;


array_ = new double[size_];

    System.out.println("");

    for (int i = 0; i < size_; i++) {

        array_[i]=PseudoRandom.randInt((int)problem_.getLowerLimit(i),(int)problem_.getUpperLimit(i));

        System.out.print(array_[i]+"\t\t");

    } // for
    /**********************************************************************************************
   } // Constructor
```

C'est un vecteur d'entier, son intervalle est entre 1 et le nombre e PM (montré dans les déclarations de la classe « problemm »

Figure IV.9 : le constructeur de la classe ArrayReal ou on a codifié notre solution

Donc pour lancer l'exécution du projet Jmetal on a utilisé la classe NSGAII_main qui utilise la métaheuristique « NSGAII » et qui fait appel à la classe de notre problème « problemm », cette dernière prend comme paramètre le type de la solution « arrayreal »et le nombre de variable « la taille de l'individu » comme montrée ci-dessous.

```
      nbr_vm=cl.saisir();
problem = new  problemm("ArrayReal", nbr_vm);
```

Saisie par le client

Figure IV.10 : Codification d'une solution.

I.6 Présentation de Jfreechart: *[51]*

JFreeChart est une API Java permettant de créer des graphiques et des diagrammes de très bonne qualité. Cette API est open source et sous licence LGPL. En revanche, la documentation est payante. On a utilisé cette API pour dessiner les histogrammes.

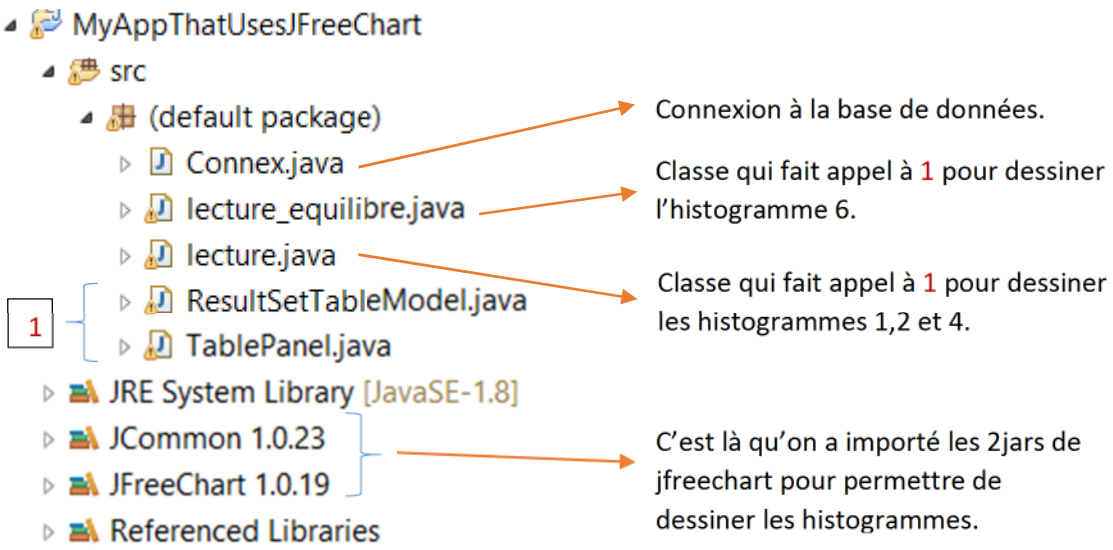


Figure IV.11 :L'ensemble des classes utilisées pour dessiner les histogrammes.

Résultats de la simulation :

Analyse des résultats :

Après avoir exécuté le programme en appliquant l'algorithme NSGAII à notre problème on aura le tableau suivant :

Charge	Energie(j)	Nbr_Vm
582895	120	12
227774	270	12
513490	160	12
2271001	80	12

Figure IV.12 : Tableau de meilleures solutions trouvées.

Ce tableau contient les résultats des fonctions objectifs de chaque solution, pour déterminer quelle est la solution moyenne on sélectionne sur une en utilisant l'histogramme ci-dessous :

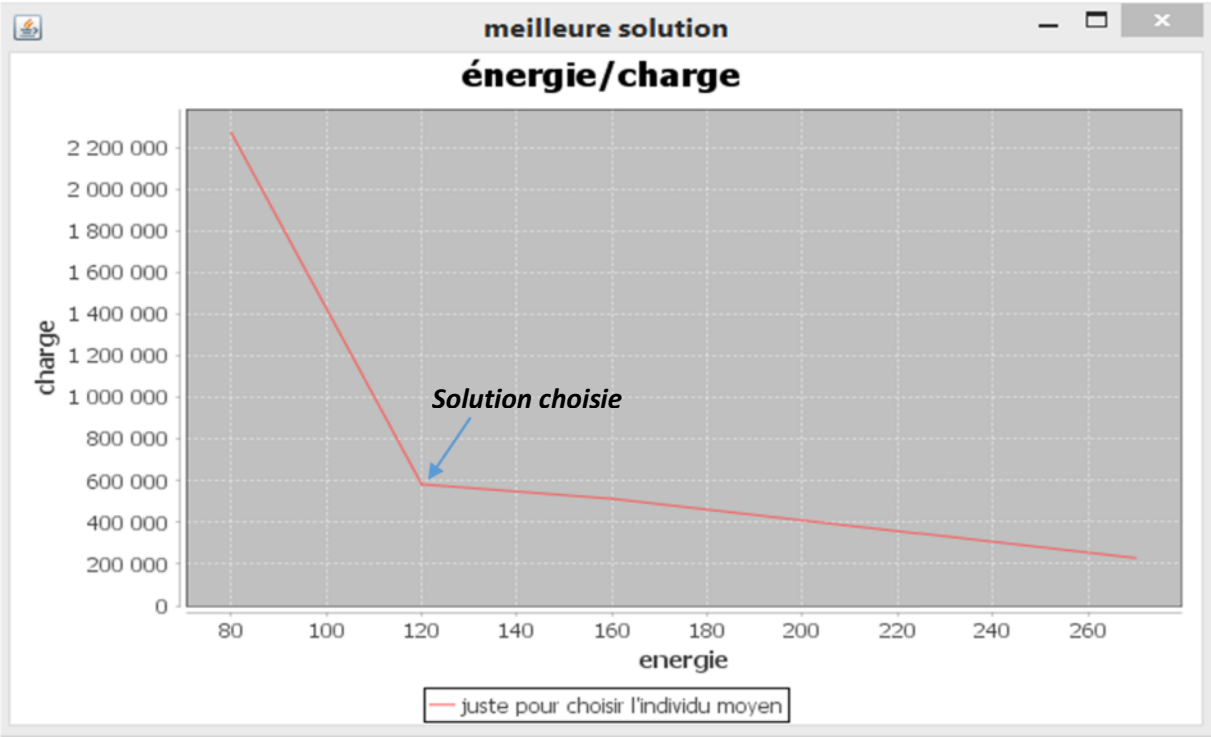


Figure IV.13 : Histogramme qui représente la charge en termes d'énergie pour indiquer quelle est la meilleure solution à garder.

Pour chaque exécution du programme nous avons spécifié un nombre de paramètres a étudié tel que : le pourcentage des serveurs utilisés, l'énergie économisée et l'équilibrage de charge et nous avons eu les résultats montrés dans le tableau de la Figure IV.14 et les histogrammes définis ci-dessous :

Nbr VM	Nbr serveurs utilisés	Pourcentage Serveurs utilisés	Energie dépensé	Energie économisé	Charge du travail	Nbr_ serveurs
150	72	0.8	4410	590	65490	Pour 90 serveurs
175	78	0.8666667	1720	3280	90816	
200	80	0.8888889	1970	3030	92772	

Figure IV.14 :Tableau qui présente une étude détaillé de chaque solution.

Ce tableau présente une étude détaillée de chaque solution obtenue et choisie en fixant le nombre de serveur a 90 et en variant le nombre de VM(150 ;175 ;200).

Pour chaque individu on lui a calculé sa charge du travail ,son énergie dépensée,son énergie économisée qui est égal a l'énergie total(5000 dans notre cas)-l'énergie dépensée,pourcentage de serveurs utilisés qui est égal au nombre de PM utilisé/nombre total des serveurs.

On remarque bien qu'a chaque fois qu'on augmente le nombre de VM le pourcentage de serveurs utilisés est presque stable comme on le voit dans l'histogramme1 car le nombre de ses derniers augmente d'un leger avantage et ce qui est normal puisque pour chaque placement d'une VM il doit veifier d'abord ses contraintes citée précédemment.

Malgré Cette augmentation des PM due a l'augmentation des VM on remarque que l'énergie dépensée diminue et la charge du travail augmente puis ils essayent de se stabiliser comme on le voit ds les histogrammes 2 et 4.

- ***Pourcentage des serveurs utilisés :***

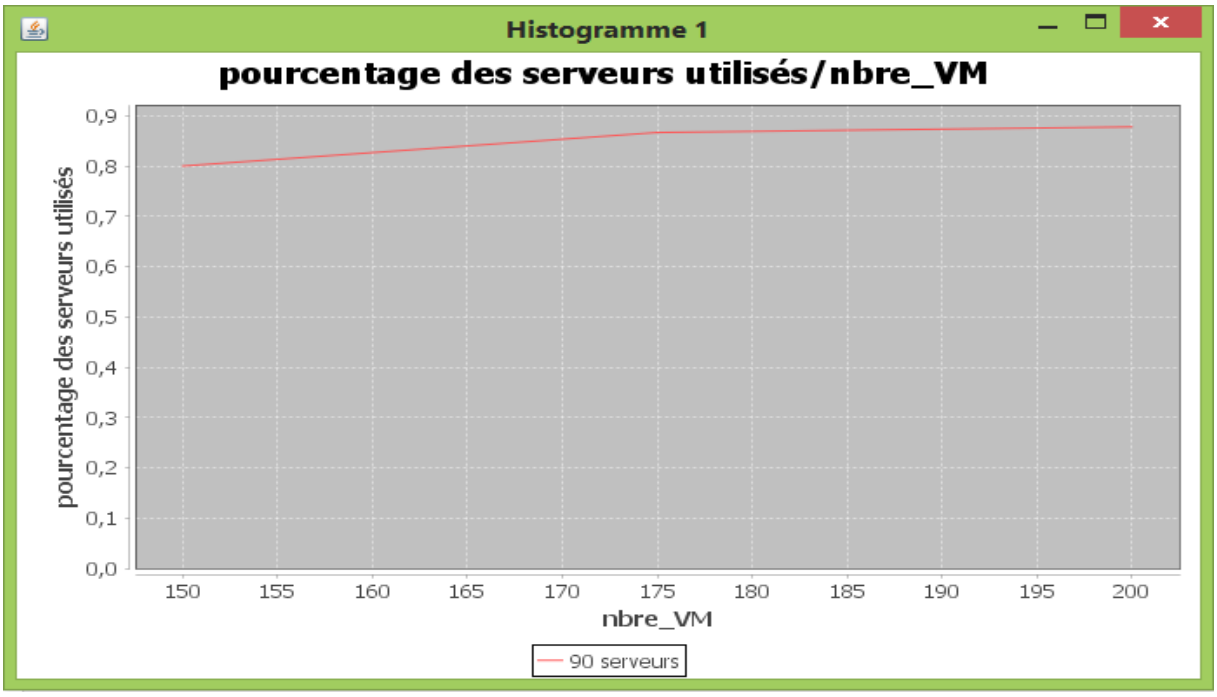


Figure IV.15 : Histogramme qui représente le pourcentage des serveurs utilisés en termes de nombre de VM.

On remarque que pour 150VM ou pour200 VM le pourcentage des serveurs utilisé est presque le même pour 90 serveurs.

- ***Energie économisée en termes de nombre de VM:***

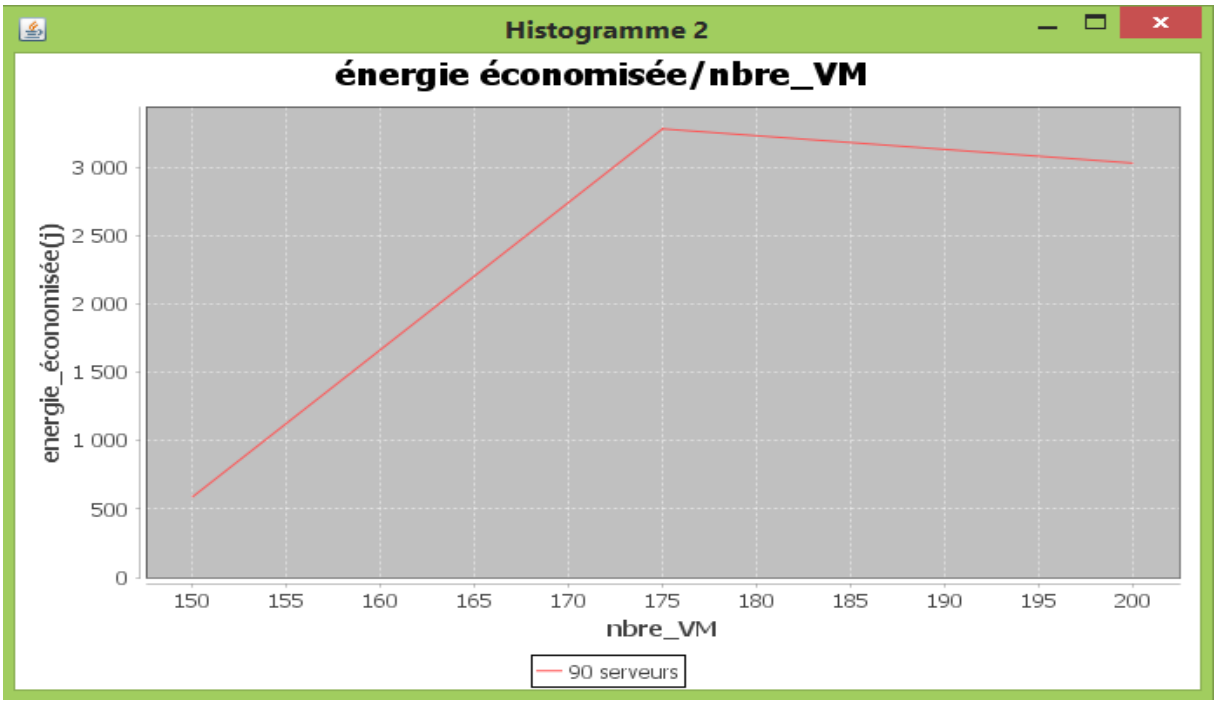


Figure IV.16 : Histogramme qui représente l'énergie économisée(en joule) en termes de nombre de VM.

Dans un premier lieu on remarque que l'énergie économisée augmente ce qui est due à la diminution de l'énergie dépensé puis elle diminue d'une manière négligeable pour un grand nombre de VM et c'est le but de notre algorithme : minimiser l'énergie dépensée par les serveurs en augmentant le nombre de VM.

- ***Energie économisée en termes de nombre de la charge:***

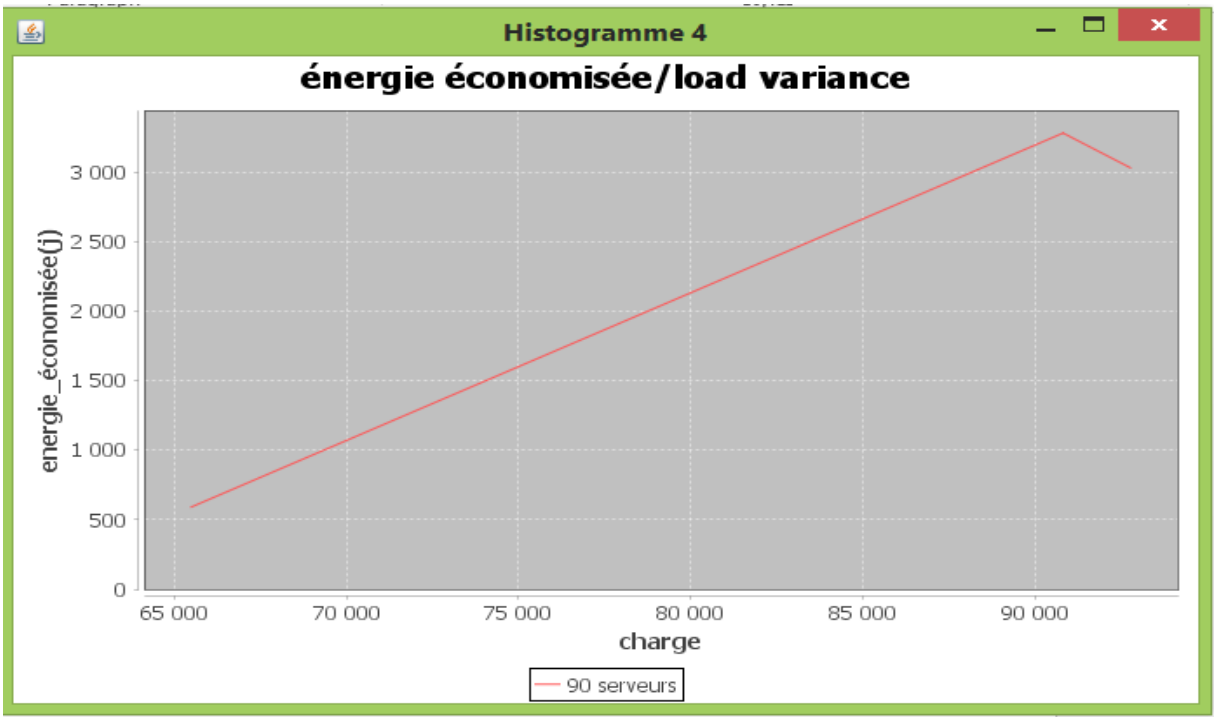


Figure IV.17 : Histogramme qui représente l'énergie économisée en termes de la charge des serveurs.

En variant la charge du travail des serveurs on remarque que l'énergie dépensée par ses derniers diminue donc l'énergie économisée augmente pour atteindre l'apogée pour un nombre de VM qui est égal à 175.

- ***L'équilibrage de la charge :***

Pour tester l'équilibrage de charge des serveurs on a utilisé une fonction qui calcule le facteur d'équilibre de tous les serveurs utilisés par une solution. Le tableau suivant montre le facteur d'équilibre pour un nombre de VM fixé et un nombre de PM varié :

Nbr_PM	Facteur d'équilibrage	Nbr_VM
60	0.9928692	Pour 100 VM
40	0.982956	
30	0.6396515	

Figure IV.18 : Tableau qui étudie le facteur d'équilibre en variant le nombre de PM.

Comme on peut le voir dans le tableau ci-dessus et l'histogramme ci-dessous qu'en variant le nombre de PM le facteur d'équilibre reste presque inchangé pour un nombre de serveur suffisant ce qui signifie que la charge est bien répartie dans tous les serveurs.

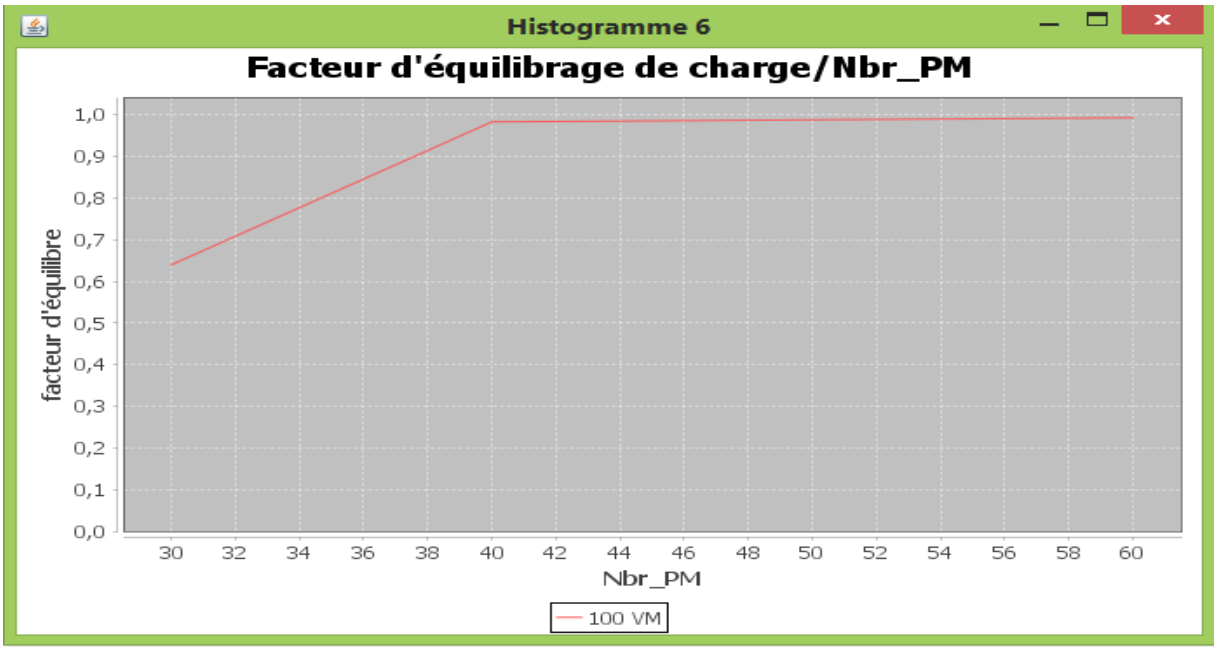


Figure IV.19 Histogramme qui représente l'équilibrage de charge en termes de nombre de PM.

Cet histogramme permet de constater que pour 30 PM l'équilibrage de charge n'est pas vérifié puis à partir de 40 PM nous avons eu un équilibrage parfait entre les serveurs tout en variant le nombre de serveurs utilisés pour un même nombre de machine virtuelle.

Conclusion :

Au cours de cette dernière étape de notre travail, nous avons présenté l'implémentation ainsi que les outils utilisés pour résoudre le problème de placement de VM dans une PM tout en prenant en considération différentes objectifs et contraintes.

Conclusion générale

Conclusion Générale :

Notre mémoire avait pour objectif de résoudre un problème d'allocation des ressources dans le cloud computing à l'aide d'une metaheuristique dont le but est de satisfaire les exigences des fournisseurs du Cloud qui consiste à allouer ses ressources aux utilisateurs tout en minimisant l'énergie dépensée par leur datacenters.

Ce travail nous a permis d'apprendre d'une manière détaillée le fonctionnement du cloud computing, la notion de virtualisation et sa relation avec l'économie d'énergie au sein du cloud. Nous avons également appris le fonctionnement du Framework jMetal ainsi le rôle de chaque classe et de chaque package pour résoudre un problème multi-objectif avec la notion pareto.

Après l'analyse faite sur les résultats obtenus après l'exécution de notre programme de résolution du problème de placement des machines virtuelles dans des serveurs en minimisant la consommation d'énergie et la régularité de la charge de travail, nous avons obtenu des résultats satisfaisants et logiques qui vérifient les contraintes fixées au préalable ainsi nos fonctions objectifs.

Comme perspective, Nous espérons étendre notre travail pour traiter le côté client. Nous espérons également l'évoluer avec un nombre supérieur à deux fonctions objectifs avec plus de contraintes, plus de machines virtuelles et serveurs et avec plusieurs centres de données tout en tenant compte de l'architecture de communication du réseau et de la sécurité dans les traitements.

Liste des matières

RÉFÉRENCE BIBLIOGRAPHIQUE

[1] : https://www.tutorialspoint.com/cloud_computing/cloud_computing_overview.htm

[2] : https://fr.wikipedia.org/wiki/Cloud_computing

[3] : https://www.safaribooksonline.com/library/view/cloud-security-and/9780596806453/ch01.html

[4] : http://blog.3li.com/cloud-les-modeles-de-deploiement/

[5] : COURAUD, Nicolas ***15*** rue vignon 75008 PARIS ***« CLOUD COMPUTING Définitions & Concepts, Enquête et Analyse des Tendances ».***
{http://www.itiforums.com/fichiers/2011_06_17_13_28_37_LIVREBLANC_CLOUD_BR.pdf}

[6] : GUILLAUME, Plouin ***«Cloud computing »,*** ISBN 978-2-10-059875-5 .Edition Dunod, Paris, 2013 (3e edition).265 pages (cote SYX 214).

[7] : http://www.lecoindesentrepreneurs.fr/le-mode-saas/

[8] : http://www.leblogdudirigeant.com/quest-ce-que-le-mode-saas/

[9] : Livre blanc rédigé par le Groupe cloud computing de l'ADIRA, 38cours Eugénie 69003 Lyon ***« Cloud computing ».***
{http://www.adira.org/wp content/uploads/2013/02/2012_ADIRA CloudComputing _LivreBlanc.pdf}.

[10] : http://blog.3li.com/les-caracteristiques-essentielles-du-cloud/

[11] :francois rivard,LAVOISIER.14,rue de provigny,94230,ISBN 987-7462-3815-2,ISSN 1635-7361, ***« cloud computing le systeme d'information sans limite ».***
{https://books.google.dz/books?id=sYe_onZeVkUC&printsec=frontcover&dq=cloud+computing+le+systeme+d%27information+sans+limite&hl=fr&sa=X&ved=0ahUKEwi7rPaO0b3WAhXkLMAKHWxqAEwQ6AEIJDAA#v=onepage&q=cloud%20computing%20le%20systeme%20d'information%20sans%20limite&f=false }

[12] : https://www.it-connect.fr/les-types-dhyperviseurs/

[13] : https://www.piloter.org/techno/support/virtualisation.htm

[14] : http://www.oceanis.fr/infrastructure/virtualisation

[15] : Cyrille DUFRESNES, 31/10/08 ***« La virtualisation ».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwiNkZym0rTWAhVKGhQKHQFJBrwQFghDMAQ&url=http%3A%2F%2Fnotionsinformatique.free.fr%2Farchitecture%2Fvirtualisation.pdf&usg=AFQjCNGQBPKhdfnOhygoHltHjQ8WLtsCw}.

[16] : ***« VIRTUALISATION, CLOUD MOBILE ET SÉCURITÉ »***

[17] : Pankajdeep Kaur and Anita Rani. International Journal of Grid Distribution Computing Vol. 8, No.5, (2015), pp.337-342. ISSN: 2005-4262 IJGDC Copyright ⓒ 2015 SERSC.
«Virtual Machine Migration in Cloud Computing ».
{http://www.sersc.org/journals/IJGDC/vol8_no5/33.pdf}.

[18] : http://www.renaudvenet.com/cloud-computing-avantages-et-inconvenients-2011-01-26.html

[19] André Jeannerot, Novembre 2012, Réalisation DG Consulting Communication, ***« CE QU'IL FAUT SAVOIR SUR LE CLOUD COMPUTING… ».***E-catalogue des Solutions Cloud Computing en Provence-Alpes-Cote d'Azur

{http://www.medinsoft.com/website/custom/module/cms/content/file/Ce_qu_il_faut_savoir_sur_le_Cloud....pdf}.

[20] https://www.techsoup.global/node/2915

[21] http://www.inei.fr/le-cloud-avantages-et-inconvenients/

[22] N. Krishnaveni ET G. Sivakumar, International Journal of Computer Applications Technology and Research (IJCAT), 2013. ***« Survey on Dynamic Resource Allocation Strategy in Cloud Computing Environment».***
{https://www.noexperiencenecessarybook.com/KXLK/survey-on-dynamic-resource-allocation-strategy-in-cloud.html}.

[23] http://theses.univ-oran1.dz/document/15201701t.pdf

[24] https://fr.wikipedia.org/wiki/Provisioning

[25] A-B. Tchana, ***«Système d'Administration Autonome Adaptable: application au Cloud***», thèse de doctorat en informatique, Université de Toulouse, 2011.
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjG8PTG37jWAhWGvRQKHe46ASYQFggvMAE&url=http%3A%2F%2Fthesis.univ-biskra.dz%2F1395%2F1%2FInf_m2_2015.pdf&usg=AFQjCNGYzzO42SkcW4GFH1hIAXAf6Y9Jhw}.

[26] Abdullah Yousafzai, Abdullah Gani, RafidahMd Noor, Mehdi Sookhak, Hamid Talebian, Muhammad Shiraz, Muhammad Khurram Khan. Received: 4 August 2014 / Revised: 5 March 2016 / Accepted: 19 April 2016 /Published online: 12 May 2016. Springer-Verlag London 2016. ***«Cloud resource allocation schemes: review, taxonomy, and opportunities».***

[27] Ms. Renu Krishnan. ISSN: 2319-7242. IJECS Volume 3 Issue 5 May, 2014 Page No.5614-5620***. « Survey Paper for Dynamic Resource Allocation using Migration in Cloud ».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&cad=rja&uact=8&ved=0ahUKEwjZlM2f17jWAhVIQBoKHXsdABUQFghUMAY&url=http%3A%2F%2Fwww.ijecs.in%2Fissue%2Fv3i5%2F8%2520ijecs.pdf&usg=AFQjCNGoGbHf96ZqsKgVoZqW0FY_HTuHEw}

[28] https://www.researchgate.net/post/what_is_the_difference_between_vm_allocation_and_resource_allocation_in_cloud_computing

[29] Glauco Estácio Gonçalves, Patrícia Takako Endo, Thiago Damasceno Cordeiro, André Vitor de Almeida Palhares, Djamel Sadok, Judith Kelner,Bob Melander, Jan-Erik Mångs. XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2011 ***«Resource Allocation in Clouds: Concepts, Tools and Research Challenges».***

[30] V. Vivek, R. Srinivasan ET E. Blessing Rajsingh,International Conference on Modern Trends in Science, Engineering and Technology (ICMTSET) 2013***, « Resource Provisioning Methodologies: An Approach of Producer and Consumer Favorable In Cloud Environment »***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjtiaqb4bjWAhUF2xoKHZI4CUgQFgg_MAA&url=http%3A%2F%2Fwww.ijetae.com%2Ffiles%2FConference_ICMTSET_2013%2FIJETAE_ICMTSET_02.pdf&usg=AFQjCNHPI3ecwhnYFyf6YSKLjF27NiY5WQ}.

[31] Université de Cergy-Pontoise Ecole Doctorale Sciences et Ingénierie ETIS (CNRS UMR 8051) EISTI (Ecole Internationale des Sciences de Traitement de l'Information)
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjBjNjaubjWAhXL1hQKHbXFC2AQFggmMAA&url=http%3A%2F%2Fwww.theses.fr%2F2014CERG0730.pdf&usg=AFQjCNHfp6MOp4X3hpSvtymme-vyUsdhng}

[32] ASHOK Kumar Turuk,Bibhudatta Sahoo,Sourav Kanti Addya.Published in the United States of America by IGI Global Information Science Reference 701 E.Chocolate Avenue

Hershey PA,USA 17033;ISSN 2327-3453:eISSN:2327-3461. ***«Resource Management and Efficiency in Cloud Computing Environments».***
{https://books.google.dz/books?id=1952DQAAQBAJ&pg=PA147&dq=dynamic+vm+placement&hl=fr&sa=X&redir_esc=y#v=onepage&q=dynamic%20vm%20placement&f=false}.

[33] https://www.researchgate.net/publication/312004396_Dynamic_Virtual_Machine_Placement_in_Cloud_Computing

[34] http://www.high-tech-info.fr/cloud-datacenters-environnement

[35] http://datacenter-mag.fr/enjeux/moderation-de-la-consommation-energetique-des-data-centers-americains/

[36] http://www.latribune.fr/entreprises-finance/industrie/energie-environnement/efficacite-energetique-les-data-centers-encore-defaillants-619895.html

[37] http://araneos.com/virtualisation-cloud

[38] https://blogrecherche.wp.imt.fr/2017/01/18/data-center-defi-energetique/

[39] ***«Résolutions de problèmes difficiles: algorithme d'approximation, algorithmes probabilistes, heuristique et métaheuristique».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiur9TUn7fWAhVCBBoKHVEeCqwQFggmMAA&url=http%3A%2F%2Fwww.lacim.uqam.ca%2F~chauve%2FEnseignement%2FINF7440%2FH05%2FHEURISTIQUES-PROBA%2FGUY-approximations-heuristiques.pdf&usg=AFQjCNG1dW09YK8-HaddnItp3Bl8sl5ozQ}.

[40] https://fr.wikipedia.org/wiki/Algorithme_probabiliste

[41] TE de fin d'année Tutorat de Mr Philippe Audebaud***. « ALGORITHMES GENETIQUES »*** {http://souqueta.free.fr/Project/files/TE_AG.pdf}.

[42] :Bandi Madhusudhan and Dr.K.Chandra Sekaran "Proceeding Of International Conference On"Emerging Research in Computing,Information,Communication and Application" ERCICA2013,ISBN: 9789351071020. ***«A Genetic Algorithm approach for Virtual Machine Placement in cloud».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwih_Zary7nWAhWHQBQKHaIcBDkQFggmMAA&url=http%3A%2F%2Fsearchdl.org%2Fpublic%2Fbook_series%2Felsevierst%2F1%2F19.pdf&usg=AFQjCNE6spOQ6qXCVIuWntH6fUXzdV0y3A}.

[43] Fabio Lopez Pires,Itaipu Technological Park (PTI),National University of Asuncion (UNA), [cs.DC] 4 Jun 2015. ***«Virtual Machine Placement Literature Review »***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&cad=rja&uact=8&ved=0ahUKEwjrt5H2ybjWAhUCWhQKHc3FCfYQFghsMAY&url=https%3A%2F%2Farxiv.org%2Fpdf%2F1506.01509&usg=AFQjCNGKdDCaPoIAYsgCGhhjUHGQdyggNQ}.

[44] Yann Collette - Patrick Siarry. ÉDITIONS EYROLLES, 2002, 61, Bld Saint-Germain, 75240 Paris Cedex 05, ISBN 2-212-11168-1***. «Optimisation multiobjectif ».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwjEz6DjrTWAhXGtRoKHZ9nC78QFggyMAI&url=http%3A%2F%2Fsouhil-mouassa.e-monsite.com%2Fmedias%2Ffiles%2Foptimisation multiobjectif1.pdf&usg=AFQjCNGXvGwclhJ4sdxkWvnEyzmnBvF4HA}

[45] Master IAD – DMDC – PATRICE PERNY.LIP6 – Université Paris 6. ***«OPTIMISATION MULTICRITERE».***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=0ahUKEwjT3r-psbnWAhWBqxoKHZkKAKQQFghjMAg&url=http%3A%2F%2Fwww-master.ufr-info p6.jussieu.fr%2F2005%2FIMG%2Fpdf%2Fdmdc5_ notes.pdf&usg=AFQjCNFAeJNt3V1KX20U69uVxpdtusOG5w}.

[46] ARAVIND SESHADRI. ***«A FAST ELITIST MULTIOBJECTIVE GENETIC ALGORITHM:NSGA-II»***
{https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwiDweiTsrnWAhXEiRoKHbibA8YQFgg4MAI&url=https%3A%2F%2Fweb.njit.edu%2F~horacio%2FMath451H%2Fdownload%2FSeshadri_NSGA-II.pdf&usg=AFQjCNEO10lkFCD2UKVmzXh1Kf7sPk3jrQ}.
[47] http://lesdefinitions.fr/excel
[48] https://fr.wikipedia.org/wiki/Oracle_Database
[49] https://fr.wikipedia.org/wiki/Java_(langage)
[50] http://hebergement.u-psud.fr/distribution/logiciels-libres/dev-web-a-programmation/398-eclipse-environnement-de-developpement-java.html
[51] http://jmetal.sourceforge.net/resources/jMetalUserManual43.pdf
[52] https://fr.wikipedia.org/wiki/JFreeChart