TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CRC	Centre de Recherches sur les Communications
FTP	File Transport Protocol
HTML	HyperText Markup Language
GES	Gaz à Effet de Serre
GSN	Green Star Network
HTPP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
MDS	Monitoring and Discovery Services
OASIS	Organization for the Advancement of Structured Information Standards
OV	Organisation Virtuelle
OSGA	Open Grid Service Architecture
OSGI	Open Services Gateway Initiative
PDU	Power Distribution Unit
RDF	Resource Description Framework
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
WSDL	Web Services Description Language

WSRF	Web Services Resource Framework
XML	Extensible Markup Language

CHAPITRE 1

INTRODUCTION

1.1 Présentation du contexte

Le besoin a toujours guidé l'être humain à de grandes inventions. La mise en place et le développement d'infrastructure a eu et continu d'avoir un grand impact sur notre mode vie. La création des chemins de fer, des routes, des réseaux électriques, des réseaux téléphoniques, etc. ont changé notre mode de vie, non seulement sur l'aspect économique, où ces infrastructures supportent des industries de plusieurs milliards de dollars, mais aussi sur l'aspect social (Cronon, 1992). Sans ces infrastructures, notre vie ne sera pas semblable à celle qu'on connait actuellement. Malgré le fait que ces infrastructures soient en apparence différentes, elles partagent un ensemble de caractéristiques communes. Pour étudier l'impact, sur nos sociétés, du développement d'infrastructure dans le domaine des technologies de l'information, la comparaison peut être faite avec les autres types infrastructures. En effet, l'infrastructure de transport a participé à la création et au développement de villes entières. De la même façon, l'infrastructure informatique regroupe autour d'elle des gens, des organisations et des compagnies, qui sont intéressés par l'information, les services et les applications qu'offre le web, et qui permettent de faciliter le travail, le commerce, etc. Cette activité humaine sur Internet génère une grande quantité de données sous plusieurs formes : textuelles et non textuelles, structurées, semi-structurées ou non structurées. Plusieurs applications, notamment dans le domaine scientifique, produisent aussi d'énormes quantités de données telles que les projets de recherches sur le génome humain, l'astronomie, la physique comme le projet du grand collisionneur d'hadrons qui va théoriquement, générer une grande quantité de données (Baud, Casey, Lemaitre, & Nicholson, 2005).

Ces données doivent être stockées, analysées et traitées. Cependant, les ressources disponibles ne sont pas souvent suffisantes pour répondre à la demande croissante en matière d'infrastructure informatique nécessaire à cet effet. La dernière décennie a vu naitre un

nouveau paradigme, en l'occurrence la virtualisation, qui a pour objectif de résoudre le problème de la demande en matière d'infrastructure. Basée sur l'idée du partage et de l'utilisation des ressources communes, cette idée n'est pas aussi récente. En effet, on a déjà partagé les supercalculateurs. Historiquement, la technologie des grilles informatiques est la première technologie qui a permis le partage d'un grand nombre de types de ressources géographiquement distantes.

De nouvelles technologies de virtualisation ont émergé récemment, en l'occurrence les technologies des nuages informatiques qui consistent tout particulièrement à partager des ressources. Les fournisseurs d'infrastructures mettent en commun un ensemble de ressources dans un espace accessible aux différents usagers autorisés à utiliser ces infrastructures.

Les nuages informatiques offrent un accès orienté service pour une vaste gamme de ressources, matérielles ou logicielles. L'intérêt suscité par les nuages informatiques ne cesse de prendre de l'ampleur, aussi bien dans les milieux industriels que dans les milieux académiques. En comparant aux technologies des grilles, l'accès aux ressources s'effectue d'une façon plus fiable et plus transparente.

Les nuages informatiques en particulier et les organisations virtuelles en général offrent une grande quantité de ressources hétérogènes. On peut trouver différents types de ressources : des serveurs, des supports de stockages, des supercalculateurs, etc. avec différents systèmes d'exploitation comme Windows, Linux, MAC OS, des micrologiciels qui font fonctionner certain type de ressources, et même pour chaque type de ressource on peut trouver différentes versions de système d'exploitation ou micro logiciel.

Cette flexibilité permet aux fournisseurs ainsi qu'aux utilisateurs de virtualiser et de partager un grand nombre de types de ressources hétérogènes. Mais aussi, elle les met devant de grands défis. L'un des problèmes auquel il faut faire face est le problème de la description et de la recherche des ressources dans un environnement virtuel. Un utilisateur qui souhaite soumettre des tâches à un système virtuel doit d'abord chercher les ressources nécessaires

pour l'exécution de ses tâches. Un problème clé dans la mise en œuvre d'une telle technologie est le système de découverte de ressources. Un mécanisme de virtualisation, que ce soit, un nuage informatique ou une grille informatique, doit se doter d'un système adéquat.

Les traditionnelles méthodes de recherche des ressources web se basent sur une recherche avec des mots clés qui représentent au mieux la ressource nécessaire à l'exécution d'une tâche spécifique. Cette méthode consiste à retourner les ressources dont les descriptions contiennent les mêmes mots clés que comporte la requête. Malheureusement, cette méthode ne donne pas satisfaction dans le cas ou les ressources sont mal décrites, de même si les requêtes des utilisateurs manquent de précision ou comportent des confusions. Car souvent, les utilisateurs ne sont pas capables de bien spécifier leurs besoins où ne choisissent pas les mots les plus significatifs. Par ailleurs, plusieurs ressources sont décrites avec des mots clés différents que ceux utilisés par l'usager dans sa requête, mais sont sémantiquement identiques. Ces ressources ne seront malheureusement pas retournées. Tout ça conduit à une sous exploitation des ressources et du potentiel existant.

1.2 Problématique

Si la virtualisation se positionne comme une solution prometteuse face à l'augmentation de la demande en matière d'infrastructure informatique, la recherche des ressources dans le contexte des organisations virtuelles interconnectées via Internet peut être une fastidieuse tâche. En effet, afin d'être performant et efficace, un système de recherche de ressources web, dans le contexte des nuages informatiques, doit absolument répondre aux questions suivantes :

- Comment modéliser et décrire des ressources qui sont hétérogènes ?

  Pour que le système de découverte de ressource soit capable de retrouver les ressources souhaitées, il est primordial de trouver une méthode avec laquelle les ressources peuvent être modélisées et décrites.
- Comment représenter les ressources ?

Afin qu'un système de découverte de ressources puisse retrouver la ressource souhaitée dans un environnement distribué, les ressources doivent être représentées d'une manière efficace. Chaque ressource doit être adressable convenablement et de la meilleure façon.

- Comment résoudre le problème de la précision?

  Tout système de recherche doit être précis et le résultat doit comporter le moins que possible de ressources non souhaitées.

- Comment résoudre le problème de la sémantique ?

  Un système de recherche dans un contexte de nuage informatique doit aller au-delà des mots clés et trouver les ressources qui sont décrites avec des mots syntaxiquement différents, mais sémantiquement identiques. Par exemple, une ressource qui décrit un serveur qui possède un système d'exploitation Ubuntu peut être compatible avec une ressource possédant un système d'exploitation Fedora. Elles font partie du même groupe, à savoir celui des ressources qui possèdent un système d'exploitation Linux.

- Comment traiter les requêtes des utilisateurs ?

  Les requêtes des utilisateurs peuvent être ambigües ou comprendre des mots clés qui ne sont pas les plus communs. Le traitement des requêtes peut augmenter les performances d'un système de découvert de ressources. Par exemple, une requête formulée de la manière suivante : {‘‘Ubuntu’’, ‘‘512 Mo’’, ‘‘HDD=30 G’’} doit être prise en charge. Pour cela elle peut être reformulée en une requête type, en ajoutant d'autres indications, comme suit : {‘‘système d'exploitation=Ubuntu’’, ‘‘Mémoire=512 Mo’’, ‘‘disque dure=30 G’’}.

1.3 Objectifs

L'objectif principal de notre travail consiste à développer une nouvelle approche pour la découverte des ressources web afin de passer outre les limitations des systèmes actuels dans le contexte des organisations virtuelles. Quant aux objectifs spécifiques, nous devrions répondre à l'ensemble des problèmes soulevés dans le paragraphe précédent. À savoir, i) la description des ressources, ii) la représentation des ressources, iii) le problème de la

sémantique et iv) l'ambigüité des requêtes. Dans ce travail nous présentons un mécanisme performant de description et de découverte de ressources dans un environnement de nuage informatique. Pour cela, nous proposons donc, un cadre de description et de recherche des ressources dans un environnement virtuel. La description se fait à travers l'utilisation des ontologies. Les ressources sont représentées à l'aide de la norme RDF (Resource Description Framework) ce qui permet d'effectuer des recherches sémantiques. Afin d'augmenter la précision, les ressources sont regroupées dans différents clusters. Le processus de découverte et de classification automatique est basé sur les réseaux Bayésiens. Les requêtes utilisateurs peuvent être étendues en ajoutant des synonymes des mots que comportent ces requêtes, si ces mots donnent de faibles résultats de recherche. Un thésaurus qui regroupe les synonymes est construit à cette fin.

1.4 Plan du mémoire

Le présent mémoire est constitué de quatre chapitres, en plus de l'introduction. Dans le chapitre deux, nous présentons une revue de littérature pour illustrer les travaux de recherche dans les grilles informatiques, un domaine proche de l'informatique en nuage auquel nous nous intéressons. Dans le troisième chapitre, une introduction à la technologie des nuages informatiques est présentée ainsi que le cadre de développement IaaS Inocybe. Dans le quatrième chapitre, nous présentons notre approche. Le cinquième chapitre présente la phase de validation et le projet dont lequel on a effectué nos expériences. La conclusion finale dresse un bilan de ce qui est réalisé et les recommandations pour les futurs travaux. La figure 1.1 présente un schéma synoptique des éléments présentés dans ce mémoire.

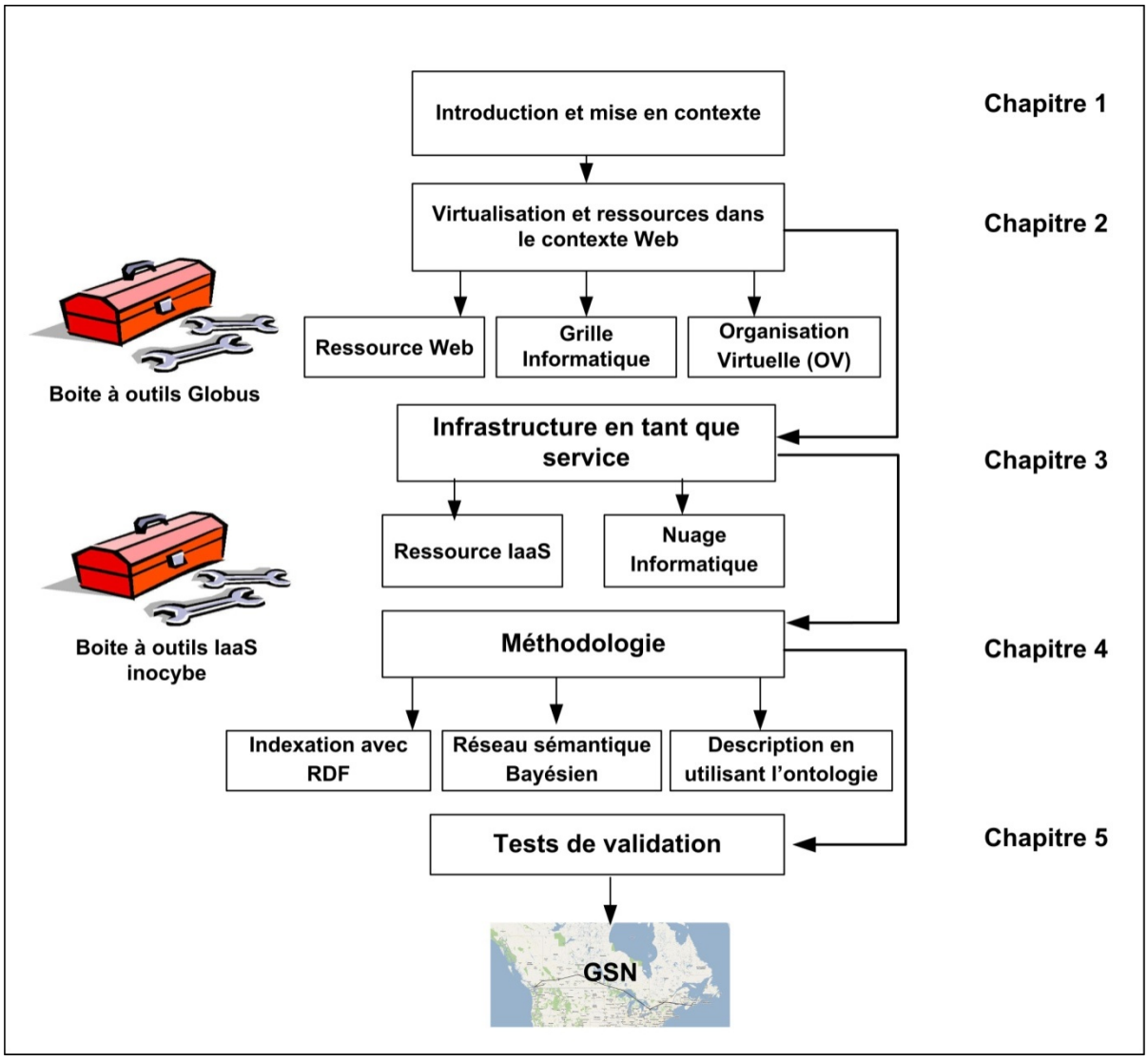

Figure 1.1 Schéma synoptique des éléments présentés dans ce mémoire.

CHAPITRE 2

LA VIRTUALISATION

2.1 Introduction

L'Internet a permis l'émergence du web ; cette technologie a rendu possible l'accès et le portage du contenu de celui-ci. Ainsi, d'énormes quantités d'information peuvent être partagées, consultées et échangées. Le web a permis le développement de plusieurs activités tel que le commerce électronique qui est l'un des plus importants propulseurs d'Internet. Le nombre de pages Web a explosé en même temps que le nombre d'internautes. Cette situation a engendré un besoin croissant en matière de stockage (Centre de Données) et de calcul (Serveurs).

Les infrastructures dans le domaine des nouvelles technologies d'informatique et de communication (TIC) sont en train de grandir d'une façon exponentielle.

La virtualisation permet de rentabiliser davantage les infrastructures jusqu'ici mal exploité et de tirer ainsi un meilleur profit de l'Internet. Cette technologie fournit le support nécessaire à l'intégration de différentes technologies, applications, fichiers et ressources informatiques. De plus, la virtualisation permet le partage global de ces ressources au-delà de ce qui a été possible jusqu'a là à travers le Web. Ainsi, un nouveau type d'organisation, nommé organisation virtuelle (OV), a vu le jour.

2.2 Organisation virtuelle

Le partage et la mise en commun des ressources par une ou plusieurs institutions forment ce qu'on appelle une organisation virtuelle (Kesselman, Concepts and Architecture, 2004). Ces ressources peuvent être des infrastructures physiques comme des serveurs, des supports de stockage, ou tout autre matériel, ou bien des ressources logicielles comme des bases de

données, des applications de calcul, des logiciels de comptabilité, etc. Les ressources dans une organisation virtuelle sont distribuées. Souvent, elles sont situées dans différents sites géographiquement distants. Ces ressources sont la propriété d'un seul organisme, mais elles peuvent appartenir, en même temps, à plusieurs organisations virtuelles. Les ressources sont évidemment partagées par les membres de l'organisation virtuelle. Cependant, il peut y avoir une restriction sur certaines ressources selon la politique de chaque organisation virtuelle. Les ressources dans une organisation virtuelle sont hétérogènes. Elles sont de tout type, différent système d'exploitation, différent modèle, etc. L'accès aux ressources d'une organisation virtuelle et fortement concurrentiel. S'il est autorisé, tout membre de l'organisation doit avoir la possibilité d'utiliser n'importe quelle ressource partagée.

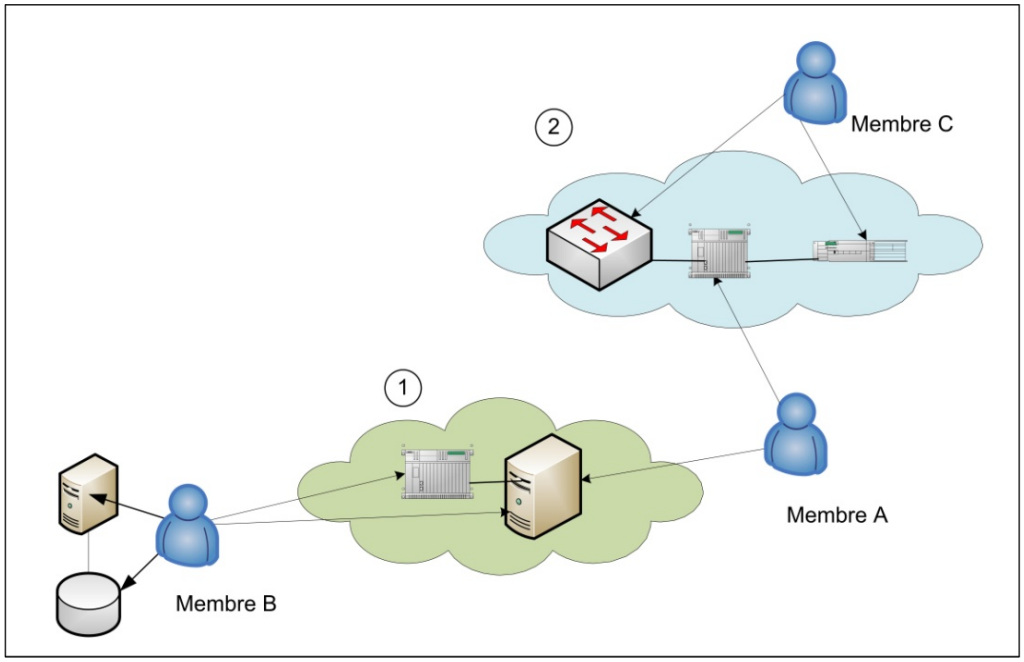

Figure 2.1 Organisations virtuelles.

La figure 2.1 représente deux organisations virtuelles 1 et 2 avec trois membres qui partagent un certain nombre de ressources. On peut voir que le membre A participe dans les deux organisations alors que les membres B et C appartiennent à une seule organisation virtuelle. Les membres A et C partagent toutes leurs infrastructures alors que le membre B partage juste une partie. Le propriétaire d'une ressource peut définir certaines conditions pour

l'utilisation de sa ressource, il peut rendre sa ressource disponible juste une période de temps, restreindre l'accès et définir les membres où les groupes qui ont droit à l'utiliser, et à quelle fin ils peuvent utiliser cette ressource. L'implémentation d'un tel type de contrainte nécessite une politique d'utilisation et un mécanisme capable de gérer les droits d'accès, l'authentification et les autorisations.

Les ressources disponibles dans une organisation virtuelle changent fréquemment, une ressource peut rejoindre l'organisation et d'autres peuvent la quitter. Les politiques d'utilisation peuvent elles aussi changer, et des membres peuvent avoir l'accès à certaines ressources dont l'accès était jusque-là interdit. Un tel environnement nécessite un mécanisme de découverte de ressources qui est capable de déterminer quelle ressource est disponible et sous quelles conditions.

2.3 Les trois couches de virtualisation

L'implémentation d'une organisation virtuelle s'effectue sur trois couches (Kesselman, Concepts and Architecture, 2004) : la couche application, la couche service et la couche infrastructure (figure 2.2). Les programmes dans la couche application envoient leurs tâches à la couche infrastructure, pour qu'ils soient exécutés, via la couche service. Le résultat est retourné vers la couche application par la couche infrastructure via la couche service. L'interaction entre ces trois couches est appelée domaine de la virtualisation des ressources. Ce domaine définit les interfaces standards et les méthodes pour créer les services, gérer et contrôler les ressources, etc.

2.3.1 La couche application

La couche application offre un ensemble de bibliothèques, un environnement de développement et de test, ainsi que des outils pour le contrôle et l'exécution des tâches utilisateurs dans l'environnement virtuel. Cette couche fait la correspondance entre les

applications utilisateurs traditionnelles, par exemple les applications de bureau, et les applications qui vont être exécutées dans un contexte virtuel.

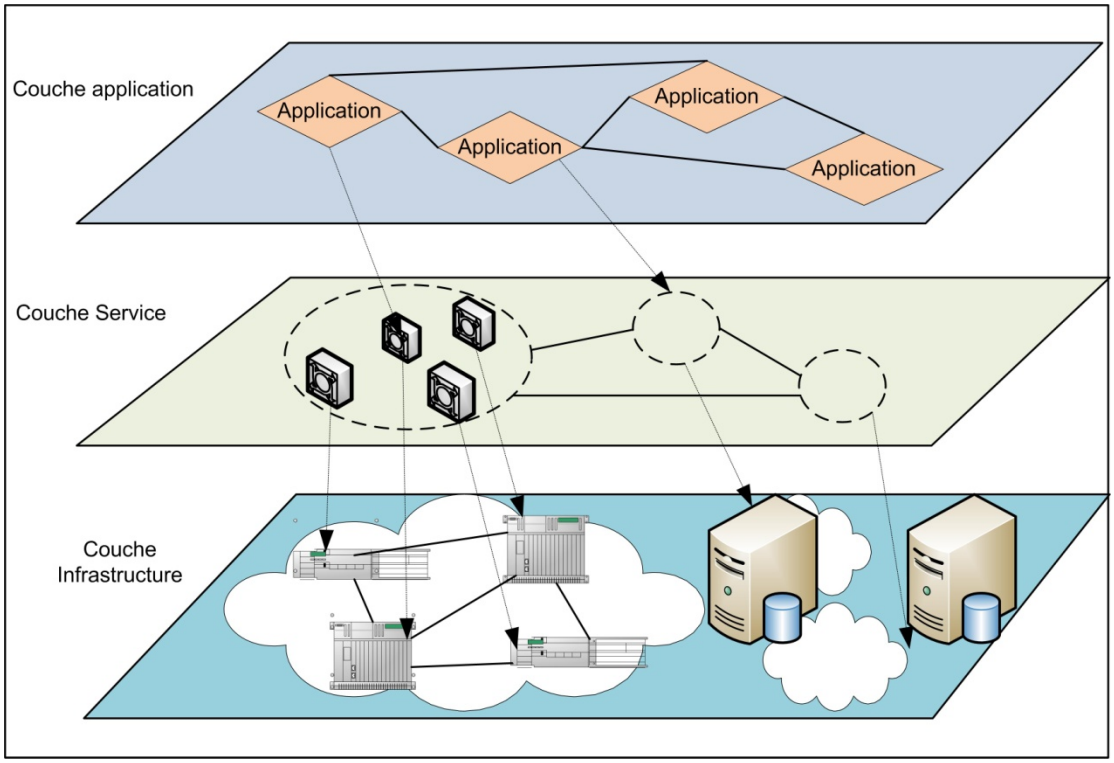

Figure 2.2 Les trois couches de la virtualisation.

Les technologies de la virtualisation offrent les outils nécessaires à l'encapsulation des services requis pour l'exécution des applications utilisateurs d'une manière transparente.

2.3.2 La couche service

La couche service est responsable de faire la correspondance entre les services utilisés par les applications et les ressources dans la couche infrastructure. Cette couche envoie les tâches soumises vers les services de la couche infrastructure et retourne les résultats vers les demandeurs de ces services. Comme un routeur, elle fait les correspondances entre les différents communicants. Cependant, cette couche ne se contente pas de faire le routage et la distribution des demandes et des réponses entre les couches application et infrastructure, elle

peut maintenir des modèles qui représentent les ressources dans la couche infrastructures et les applications qui les sollicitent.

Cette couche occupe un rôle central dans la technologie de virtualisation, elle s'occupe de la gestion des ressources et de l'ordonnancement dans toute l'infrastructure.

2.3.3 La couche infrastructure

La couche infrastructure gère les ressources où les applications vont être réellement exécutées. Cette couche représente les nœuds où les ressources physiques résident. Souvent, on y trouve une vaste gamme de ressources hétérogènes, comme des serveurs, des ordinateurs, des supercalculateurs, des supports de stockages et toutes les autres ressources qui peuvent être mise à la disposition des utilisateurs.

2.4 La technologie des grilles

La grille informatique ou "Grid Computing" est une infrastructure virtuelle distribuée. Par infrastructure, on désigne tout environnement qui comporte les équipements matériels ainsi que tous les logiciels disponibles pour les utilisateurs. Un tell environnement vise à permettre le partage des ressources distribuées (Foster, 2001). La sélection et l'utilisation sont basées sur la disponibilité, la capacité, les performances et le coût de ces ressources. Les fournisseurs et les utilisateurs des ressources dans cette technologie forment des organisations virtuelles. On parle d'organisations virtuelles parce qu'il n'y a pas de lien physique direct qui relie les membres de ces organisations. Les liens sont justes sur le plan logique.

L'idée derrière les grilles informatique n'est pas très récente. Par exemple, on a déjà partagé les supercalculateurs. En effet, en 1965 débute le projet Multics (Multiplexed Information and Computing Service) basé sur le partage du temps de processeur. Ce projet a donné

naissance à une application qui a été utilisée jusqu'a l'année 2000. Dans les années 90, le concept du partage d'une grande gamme de ressources hétérogènes et étendues sur un vaste réseau (Internet) a été présenté sous le nom de “Grid computing”. L’idée de Grid Computing est inspirée du réseau électrique (Vladimir, 2006). Les utilisateurs sont interconnectés au même réseau de distribution d'électricité, où ils peuvent aller chercher et consommer la quantité nécessaire pour le fonctionnement des différents appareils électriques par exemple. Tout le monde partage et le réseau, et la source d'énergie. Les gens du TI ont adopté le même concept (et même le mot *grille*) pour faire face à la demande croissante en matière d'infrastructure. On a pensé à partager les ressources qui sont déjà connectées via Internet. Un utilisateur qui aura besoin d'un disque dur dont la capacité dépasse celle disponible localement, a la possibilité de se faire allouer un espace disque virtuel disponible dans l'organisation virtuelle dont il fait partie. Pour le bon fonctionnement de cette technologie, certains aspects, d’ordre technique et organisationnel, doivent être bien définis. Les protocoles qui régissent les droits d'utilisations ainsi que certaines politiques de sécurité doivent être établi par exemple le rôle de chacun dans l’organisation, qui le droit d’utiliser certaines ressources ? Qui possède le contrôle ? etc. Selon *Ian Foster* un Grid Compute system doit satisfaire trois principaux objectifs (Vladimir, 2006)

2.4.1 La gestion décentralisée des ressources

Une grille informatique doit avoir le pouvoir d'intégrer des ressources et des utilisateurs qui appartiennent à des domaines différents. Ce qui permettra d'utiliser des ressources qui appartiennent à différentes OV. Une grille doit aussi gérer les questions de sécurité, les politiques d’adhésion, de paiement, etc.

2.4.2 Standard et code source libre

Comme il a été déjà mentionné, les ressources sur une grille sont hétérogènes. Sur un système de grille, nous pouvons trouver différentes ressources, telles que des ordinateurs de

bureaux, des grappes, des supercalculateurs, des supports de stockage, etc. qui utilisent une grande variété de systèmes d'exploitation et d'architectures. Le défi est de les faire collaborer pour exécuter les tâches des utilisateurs. La seule façon de faire est d'utiliser les standards, code source libre, des protocoles universels et les interfaces qui fournissent un langage commun que tout le monde sur la grille peut comprendre. De cette façon, les grilles informatiques supportent l'hétérogénéité et l'interopérabilité.

Dans cette technologie, il est essentiel que l'authentification, l'autorisation, la découverte de ressources, et l'accès aux ressources doivent êtres effectués via des standards et des codes sources libres.

2.4.3 Qualité de service

Les grilles forment un système fortement concurrentiel. De plus, la nature dynamique des ressources, qui apparaissent et disparaissent sans préavis, rend leurs manipulations délicates. Les ressources devraient être utilisées de manière coordonnée pour fournir une qualité de service, minimiser les temps de réponse, et augmenter le débit de transfert de données et leurs disponibilités pour répondre aux besoins complexes des utilisateurs. Le système des grilles doit garantir un accès sécuritaire aux ressources et mettre à jours l'état de chaque ressource pour assurer la cohérence des données.

2.5 L’architecture d’une grille

Les grilles informatiques incarnent une combinaison d'une architecture décentralisée pour la gestion des ressources et une architecture en couches hiérarchiques pour la mise en œuvre des divers services qui les constituent (Daniel, 2005). La création d'un système complet de grille informatique nécessite une grande variété de protocoles, services, et logiciel de développement. Dans ce qui suit, on va décrire l'architecture en couche proposée par (Kesselman, 2004). Dans figure 2.3, on peut voir les différentes couches qui représentent

l'architecture d'une grille. Chaque couche peut communiquer avec n'importe quelle couche inférieure. L'architecture comporte les couches suivantes (Thierry, 2003) :

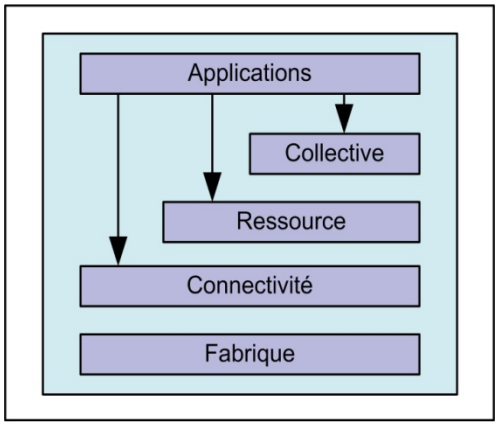

Figure 2.3 Architecture d'une grille.

2.5.1 La couche Applications

La première couche est la couche *Applications*, qui représente l'ensemble des fonctions qui vont êtres exécutées dans la grille. C'est dans cette couche que résident les applications utilisateurs. On y trouve l'ensemble des logiciels clients de la grille, qui fournissent les services dont les utilisateurs auront besoin. Ces applications font appel à leurs tours aux services des autres couches de l'architecture. Ils font appel à la couche collective et ressources pour retrouver les ressources demandées. Une fois les ressources retournées les applications utilisent les services de la couche fabrique pour contrôler ces ressources.

2.5.2 La couche collective

La couche *collective* comporte les services qui gèrent les multi-ressources. S'appuyant sur les services fournis dans la couche *ressource*, cette couche gère les tâches utilisateurs qui font appel à plusieurs ressources simultanément. La couche collective s'occupe de la coordination, de l'allocation et éventuellement de la collaboration entre les ressources sélectionnées, ainsi que la facturation et la sécurité. Elle est responsable de toute l'orchestration des ressources en

offrant aussi les services suivants :

Registres des ressources : C'est le service qui enregistre les ressources dans l'annuaire. Ainsi, il rend possible la découverte des ressources, sur la base de leurs propriétés. L'annuaire joue un rôle primordial dans le bon fonctionnement d'un système de grille. Il est comme une base de données, où les ressources sont enregistrées avec leurs propriétés. Quand une application sollicite l'utilisation d'une ressource, l'annuaire est interrogé et l'adresse de la ressource qui répond au mieux à l'exécution de cette application est retournée. Une fois l'emplacement de la ressource est localisé l'application peut exécuter ses tâches.

Allocation et ordonnancement : Généralement un utilisateur qui veut exécuter des tâches dans un système de grille informatique ne dispose pas du pouvoir de choisir quelle ressource il va utiliser. C'est le service d'allocation et d'ordonnancement qui choisit et sélectionne la ressource qui va être utilisée pour exécuter une tâche bien spécifiée. Il utilise l'annuaire pour trouver la ressource ou les ressources nécessaires, et alloue cette ou ces ressources pour exécuter les tâches utilisateurs.

Le service de contrôle : Ce service veille sur le bon fonctionnement des ressources de la grille. Il vérifié les ressources et s'assure qu'il n'y a pas d'anomalie dans le fonctionnement de ces dernières.

Service de gestion de données : Les applications sur un système de grille utilisent un ensemble de données pour accomplir leurs tâches. Ce service se charge de garder une trace de ces ensembles de données et de les transférer à la ressource qui en a besoin.

2.5.3 La couche de ressources

Dans cette couche, on trouve tous les services et protocoles qui permettent la gestion des ressources individuellement. Ses services s'occupent de l'initialisation des ressources, leurs

contrôles, et leurs surveillances. Elle s'occupe aussi de la facturation de l'utilisation de la ressource partagée. Cette couche n'est pas concernée par les interactions globales entre les ressources. Car ces dernières sont prises en charge par la couche *collective*. La couche ressource s'intéresse principalement aux caractéristiques proprement dites des ressources. Elle fait appelle aux services des couches *connectivité* et *fabrique* pour collecter les informations sur les ressources qui seront utilisées pour leur gestion. Cette couche prend en charge aussi la surveillance et le contrôle des opérations. Elle peut mettre fin à une opération par exemple. Elle s'occupe aussi de la notification aux couches supérieures des erreurs qui peuvent se produire. Dans cette couche, on distingue deux principales classes de protocoles :

Les protocoles d'information : Les protocoles d'information sont utilisés pour obtenir des informations sur la structure d'une ressource, par exemple, sa configuration, sa stratégie de sécurité, sa politique d'utilisation comme les coûts d'utilisation, et aussi d'autres informations dynamiques comme l'état actuel de la ressource ou la charge de travail de la ressource à un moment donné.

Les protocoles de gestion : Les protocoles de gestion sont utilisés pour négocier l'accès à une ressource partagée, en précisant certaines conditions. Ces protocoles définissent les exigences demandées pour l'utilisation d'une ressource. Les conditions d'accès, par exemple, si une réservation est nécessaire, ou une connaissance préalable des opérations qui vont être effectuées est nécessaire à l'accès à une base de données.

2.5.4 La couche connectivité

La couche connectivité présente les protocoles fondamentaux qui permettent à la ressource de communiquer. Cette couche implémente tous les protocoles qui permettront aux ressources de communiquer. On y trouve tous les principaux protocoles, tels que les protocoles d'internet (IP), et de transport comme TCP et UDP, application (HTTP, DNS), etc. Le but est de garantir une communication rapide, fiable et sécurisée. Face à ce défi, la

technologie des grilles informatiques a vu naitre d'autres protocoles propres à cette technologie. Essentiellement des protocoles de sécurité comme le GSI (Grid Security Infrastructure).

2.5.5 La couche fabrique

La dernière couche est la couche fabrique qui présente les ressources à partager. C'est la couche qui représente les ressources du point de vue physique telles que les ordinateurs, les grappe, les supercalculateurs, les ressources de stockage, les ressources réseau, les bases de données, etc. Les ressources peuvent êtres aussi bien des entités physiques que des entités logiques, par exemple, les bases de données et les machines virtuelles qui roulent dans les serveurs.

Lorsque la grille reçoit une requête d'une application qui a besoin d'utiliser une ressource, c'est au niveau de la couche fabrique que s'effectue l'accès à la ressource qu'elle soit physique, comme un appareil ou une composante électronique (routeur, serveur, etc.), ou logique comme une base de données ou un logiciel de calcul.

2.6 Open Grid Service Architecture (OSGA)

OSGA est une initiative qui définit un ensemble de recommandations pour mettre en œuvre un système de grille. Un système de grille consiste en plusieurs composantes. En plus des services importants tels que les services de sécurité et les services de gestion de données, les recommandations OSGA comportent les services suivants :

Le service de gestion des organisations virtuelles : Le rôle de ce service est de gérer les nœuds et les utilisateurs dans un système de grille. Il définit les membres de chaque organisation virtuelle, à savoir quel utilisateur ou quel nœud appartiennent à quelle organisation virtuelle.

Le service de découverte et gestion des ressources : Ce service permet aux applications qui utilisent la grille de trouver les ressources dont elles ont besoin, et de les gérer.

Le service de gestion des tâches : Ce service fournit aux utilisateurs de la grille les moyens de soumettre leurs tâches.

Ces derniers services sont en constante interaction. Chaque service a la possibilité de faire appel aux fonctions des autres services. Par exemple, le service de gestion des tâches souvent fait appel au service de découverte de données afin de trouver la ressource la plus appropriée pour exécuter une tâche particulière.

Il est clair que l'implémentation de ce nombre élevé de services et de leurs interactions peut causer des problèmes lors de la mise en service. Imaginons si chaque fournisseur d'infrastructure décide de fournir des services de gestion des tâches propres à lui, non seulement il propose des services d'une manière complètement différente, mais aussi via des interfaces différentes. Il serait très difficile, voire même impossible, de faire fonctionner un système de grille.

La solution est dans la normalisation. Il faut définir des standards d'interfaces communs pour chaque type de service.

L'Open Grid Services Architecture, développé par le Global Grid Forum, vise à définir une architecture commune, standard et ouverte pour les applications destinées à un environnement de grille. L'objectif de l'OGSA est de normaliser les services que l'on trouve dans une application de grille (services de gestion des tâches, services de gestion des ressources, les services de sécurité, etc.) en spécifiant un ensemble d'interfaces standard pour ces services.

Pour supporter cette architecture les concepteurs ont choisi d'utiliser la technologie des

Services Web afin de prendre en charge l'aspect communication dans cet environnent distribue. Le choix a été porté sur les services Web parce que c'est une technologie indépendante de la plateforme d'exécution (Sotomayor, 2006) qui requiert des services avec état. Bien que la technologie des services Web soit la meilleure option, elle ne répond pas à l'une des exigences les plus importantes d'OGSA. En effet, l'intergiciel utilisé doit offrir des services avec état ce qui n'est pas le cas avec les Services Web. La solution a été donc d'adopter les spécifications de WSRF (Web Service Resource Framework) (Sotomayor, 2006).

2.7 Les types de grilles

Il existe différents types de grilles informatiques selon leurs utilisations. Deux principaux types de grilles sont connus : les grilles de calcul et les grilles de données.

2.7.1 Grille de calcul

Les grilles de calcul sont principalement utilisées pour effectuer des calculs intensifs qui nécessitent une grande capacité souvent non disponible dans les ordinateurs ordinaires. Alors, les grilles de calcul sont utilisées pour rassembler un nombre d'ordinateurs pour construire un supercalculateur virtuel.

2.7.2 Grille de données

Les grilles de données sont dédiées au stockage de grande quantité de données. De la même façon que les grilles de calcul, les grilles de données forment un grand support de stockage virtuel.

2.8 Web Services Resource Framework WSRF

Le Web Services Resource Framework est une spécification développée par *OASIS*. WSFR

est le fruit de la collaboration des deux communautés, Grille et Service Web. Le but de WSRF est d'offrir la possibilité de définir un état pour un service Web. Afin de comprendre comment le WSRF permet à un service Web d'avoir un état ; on doit d'abord comprendre le fonctionnement d'un service Web.

2.8.1 Service Web

Un service Web est un composant logiciel qui permet l'interaction entre deux machines ou bien entre deux logiciels via le web. C'est une technologie qui permet de créer des applications client/serveur de la même nature que les technologies CORBA, RMI, EJB, etc. Contrairement à une page web qui est destinée à un humain en utilisant le standard HTML, le service Web est destiné au logiciel et machines. Par exemple, une personne qui a besoin de savoir le temps qu'il fait aujourd'hui il lui suffit d'ouvrir la page web de *meteomedia.com* et soumettre le code de la région voulu pour que les informations métrologiques soient affichées.

Supposant que le service de déneigement de la ville envoie un message à tous les employés qui travaillent dans ce service pour rejoindre leurs poste de travail dès qu'il y a un risque de précipitation qui dépasse 5 cm de neige avec une probabilité supérieure a 50%. Un employé de permanence vérifié aux 5 heures le bulletin météo d'Environnement Canada. Si la ville veut automatiser cette procédure, afin de réduire l'erreur et d'augmenter l'efficacité, elle peut utiliser le client du service web qui va vérifier périodiquement le serveur du service web mis à la disposition des utilisateurs par Environnent Canada et dès que les conditions sont réunies ce service envoie le message à tous les employés. On appelle ce système un service Web de météorologie ou de déneigement.

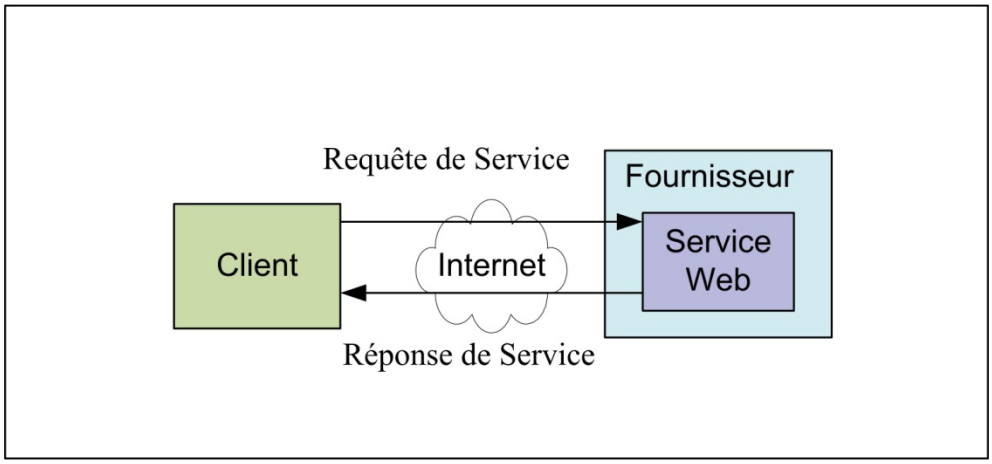

Figure 2.4 Service Web.

La figure 2.4 montre le client qui va utiliser les services du serveur pour obtenir les informations météorologiques.

Les autres technologies comme CORBA, RMI, EJB peuvent aussi fournir les moyens pour mettre en fonction un tel service, mais les services Web possèdent les avantages suivants par rapport à ces technologies :

- Les services Web sont indépendants de la plateforme d'exécution et du langage de programmation, du fait que les services Web utilisent le standard XML. En effet un programme client programmé en C++ et s'exécutant sous Windows, peut faire appel à un Service Web programmé en Java et s'exécutant sous Linux.
- Les services Web utilisent le protocole HTTP pour transmettre des messages ce qui fait qu'ils traversent facilement les routeurs et les pare-feux. Les autres technologies telles que CORBA éprouvent souvent des difficultés avec les pare-feux.
- Un autre avantage majeur est que les services Web sont à faible couplage. Le client ne possède aucune information sur le serveur jusqu'au moment de l'invocation. Contrairement aux autres technologies qui sont à fort couplage.

Comme toute autre technologie, le service Web possède aussi des inconvénients. En particulier dû à l'utilisation de la technologie XML pour transmettre les données. Ceci

implique des problèmes tels que la transparence et le surcharge.

2.8.2 Invocation d'un service Web

Dans la figure 2.5 on peut voir les différentes étapes dans une invocation typique d'un service web. La première étape consiste souvent dans l'invocation du système de découverte des services Web, lui même aussi est un service web.

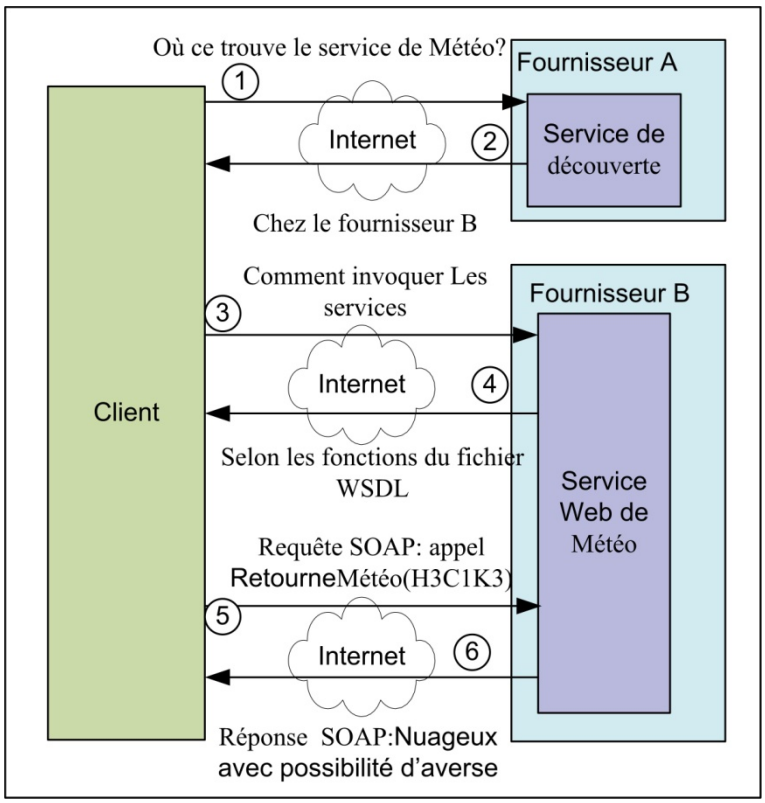

Figure 2.5 Invocation d'un service Web.

Dans le cas où le client connait l'adresse du Service Web dont il a besoin, il va l'invoquer directement sans faire appel au Service de découverte. Dans la deuxième étape, le service de découverte retourne une adresse d'un service web capable de fournir les services dont le client a besoins.

Dans la troisième étape, le client connaît où trouver le service dont il a besoin, mais il ne connaît pas comment invoquer ses services. En d'autres termes, il ne connaît pas les interfaces à appeler. Alors, il doit interroger le service lui demandant de se décrire.

Dans la quatrième étape, le service retourne le fichier WSDL qui décrit toutes les interfaces avec lesquelles le client peut invoquer les Services. Dans la cinquième étape, le client envoie une requête SOAP qui contient la commande ainsi que les paramètres, si nécessaires, que le service va exécuter. Et finalement, le Service retourne le résultat en utilisant une réponse SAOP lui aussi.

2.8.3 Architecture d'un Service Web

L'architecture des services Web peut être présentée de plusieurs façons selon le niveau de détail des protocoles qu'on veut présenter. Dans la figure 2.6 nous pouvons voir la pile qui représente cette architecture (W3C, 2004).

Service des processus : Cette partie de l'architecture contient plusieurs services Web. Par exemple, le service de la découverte qui nous permet de localiser un service particulier parmi un ensemble de services Web.

Service de la description : L'une des caractéristiques les plus intéressantes des services Web, c'est le fait qu'ils sont auto descriptive. Une fois que le service a été localisé, on peut lui demander de se présenter et décrire les opérations qu'il fournie et comment les invoquer. Cette fonctionnalité est faite via le standard *Web Services Description Language (WSDL).*

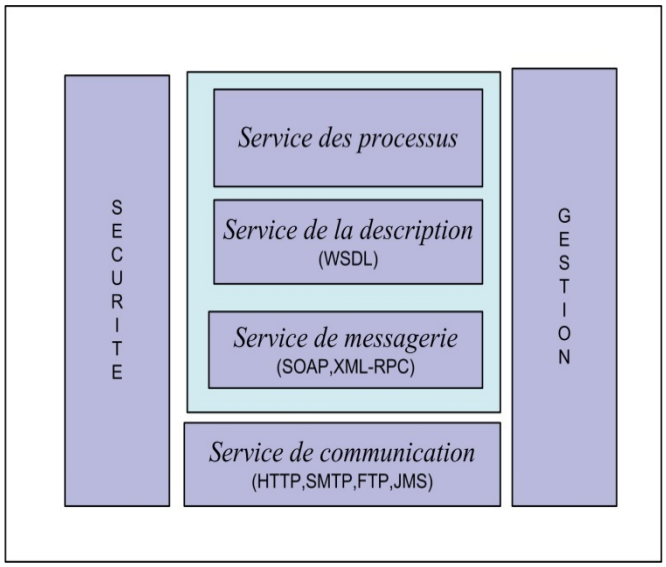

Figure 2.6 Architecture d'un Service Web.

Service de messagerie : L'objectif de cette couche est de fournir les moyens et les outils pour accéder et faire appel à différents services Web. SOAP (Simple Object Access Protocol) est le principal protocole d'échange de messages entre le client et le serveur. D'autres protocoles peuvent êtres utilisés afin d'échanger les messages dans les services web tels que XML-RPC, mais le protocole SOAP c'est imposé comme un standard pour les services Web.

Service de communication : Le protocole utilisé pour le transport des messages est le HTML (HyperText Transfer Protocol). Bien sûr, d'autres protocoles peuvent être utilisés pour le transport des messages, mais actuellement le HTML est le plus utilisé.

Service de sécurité : Les services web utilisent le format XML pour l'échange de données. Comme toute application Internet, l'aspect sécurité doit êtres pris en charge dans le but de protéger les données d'où le développement de la norme WS-Security.

2.8.4 Le protocole SOAP

SOAP (Simple Object Access Protocol) est un protocole de communication client/serveur. Ce

protocole a l'avantage d'être simple et léger. Le principe de communication est basé sur l'échange des messages sous format XML, généralement via le protocole de transport HTTP (Li, 2005).

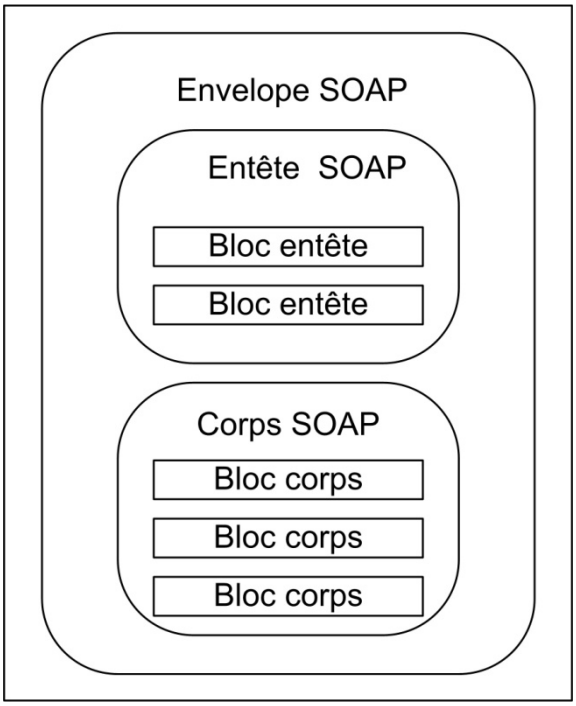

Figure 2.7 Les différents éléments d'un message SOAP.

Comme on peut le voir dans la figure 2.7, un message SOAP est constitué de l'élément enveloppe, qui est l'élément racine. À l'intérieur de cet élément se trouve l'élément *entête* et l'élément *corps*. Il peut y avoir aussi un élément *fault* pour prendre en charge d'éventuelles erreurs. Ce format standard est utilisé pour envoyer une requête d'un client vers le service, et pour envoyer aussi la réponse du service vers le client.

Enveloppe : l'élément *enveloppe* (figure 2.8) à le format suivant :
Il est essentiel qu'il contienne le *Name Space* suivant : *xmlns:env="http://www.w3.org/2001/12/soap-envelope*
Tous les documents SOAP doivent avoir ce nom d'espace. Le nom *env* peut prendre n'importe quelle valeur.

```
<?xml version="1.0"?>
<env:Envelope  xmlns:env="http://www.w3.org/2001/12/soap-envelope" >

<env:Header>
</env:Header>

<env:Body>
</env:Body>

</env:Envelope>
```

Figure 2.8 Élément enveloppe.

Entête : L'élément entête (header) est optionnel. Il contient des informations qui ne font pas partie du corps (*body*) du message SOAP. Le *Name Space* utilisé dans cet élément doit être le même que celui de l'enveloppe. On peut mettre n'importe quelle information dans l'élément entête, toutefois il existe des éléments standards tels que *mustUnderstand*, *encodingStyle*, *role* et *relay*.

Corps : Le corps (Body) est l'élément principal d'un message SOAP. C'est la partie qui contient les données échangées entre le client et le service Web. Le corps peut contenir n'importe quelle information valide.

Fautl : l'élément *fault* est retourné, à l'intérieur du corps, par le service web pour indiquer qu'il y a eu une erreur lors du traitement d'un message SOAP.

La figure 2.9 montre une requête SOAP émise par un client afin d'invoquer les opérations offertes par le service Web de la météo.

```
POST /InStock HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.synchromedia.ca/weather">
  <m:GetWeather>
    <m:CodePostale>H3C1K3</m:CodePostale>
  </m:GetWeather>
</soap:Body>

</soap:Envelope>
```

Figure 2.9 Requête SOAP.

Et comme réponse à cette requête, le serveur du service Web répond par un message SOAP, illustre à la figure 2.10, qui correspond aux informations demandées par le client.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.synchromedia.ca/weather">
  <m:GetWeatherResponse>
    <m:Temperature>8</m:Temperature>
     <m:ConditionsActuelles>Faible pluie</m:ConditionsActuelles>
  </m:GetWeatherResponse>
</soap:Body>

</soap:Envelope>
```

Figure 2.10 Réponse SOAP.

2.8.5 Web Service Description Language

Le Web Service Description Language (WSDL) est un fichier écrit sous format XML (Shah, 2007). Cela rend le fichier WSDL indépendant de la plateforme. Le WSDL décrit l'ensemble des services Web offertes par le serveur.

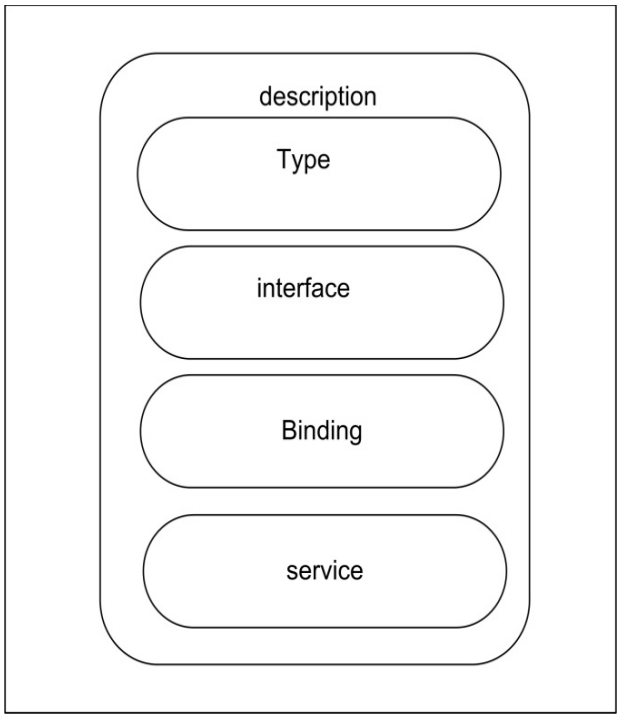

Figure 2.11 Les différentes parties d'un fichier WSDL.

Le fichier de description des services web peut être transformé en modèle d'objet avec n'importe quel langage de programmation, sous n'import quelle plateforme et envoyé avec tout type de système de messagerie.

Dans le WSDL version 2.0 on peut voir les éléments suivants (figure 2.11) :
Description : Est l'élément racine du fichier WSDL 2.0. Tous les autres éléments du fichier WSDL sont des fils de cet élément.

Type : L'élément type spécifié les types de données qui sont échangées entre le client et le service Web. Par défaut, ces types de données sont décrits en utilisant le Schéma XML.
Interface : L'élément d'interface décrit les opérations que le service web peut exécuter. Et pour chaque opération il définit les messages qui sont échangés sous forme d'entrée/sortie. Il peut aussi spécifier les messages d'erreurs par défaut.

Binding : l'élément *binding* décrit le protocole de transport utilisé pour accéder au service web. Généralement le protocole HTTP est le plus utilisé.

Service : cet élément fournit l'emplacement du service web que le client souhaite invoquer. Généralement il contient une adresse URL qui indique l'endroit où le service peut être invoqué.

Documentation et **import** sont deux éléments facultatifs qui contiennent respectivement des notes lisibles par l'être humain et des fichiers Schéma XML ou d'autres fichiers WSDL importés.

2.8.6 Service Resource Framework

Comme on l'a vu précédemment, le service web ne possède pas de mécanisme pour garder un état, c.-à-d. le service web ne garde pas les informations d'une invocation client à une autre. Dans l'exemple de la figure 2.12, l'appel du premier utilisateur à la méthode *Ajouté* du service n'a aucun effet sur l'appel du deuxième et troisième utilisateur parce que le service web ne garde aucune trace des appels.

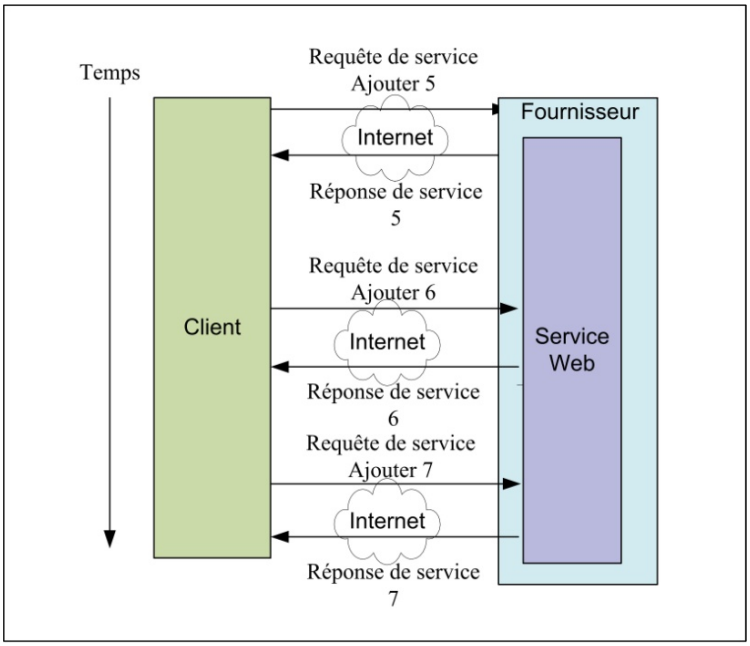

Figure 2.12 Service Web sans état.

Dans le contexte des grilles, une ressource est censée avoir un état. La solution donc, vient de WSRF qui est un ensemble de spécifications. Entre autres, il introduit le WS-Resource pour modéliser et gérer les informations d'état des services Web.

2.8.7 La ressource dans le contexte des grilles

Pour permettre à un service web d'avoir un état et de garder les informations, il a fallu séparer le service web de son état. Le service web continue à faire son travail et l'état est délégué à une autre entité séparée du service web qu'on appelle *Ressource* (figure 2.13). Le rôle de la ressource est de garder les informations sur l'état du service web et de le mettre à jour en cas d'invocation. Alors, on peut définir une ressource web comme un service web avec état.

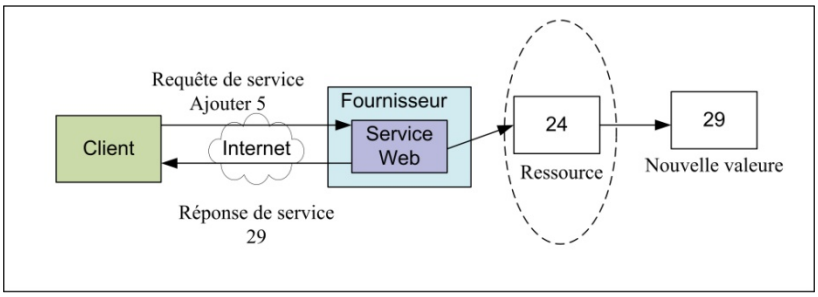

Figure 2.13 Service Web avec état.

La ressource peut contenir plus qu'une valeur. Elle peut représenter des objets complexes et avoir n'importe quelle structure.

2.8.8 WS-Resource

Le *WS-Resource* est défini comme une entité composée d'un service web et d'une ressource chargée de garder les informations sur l'état du service web (figure 2.14).

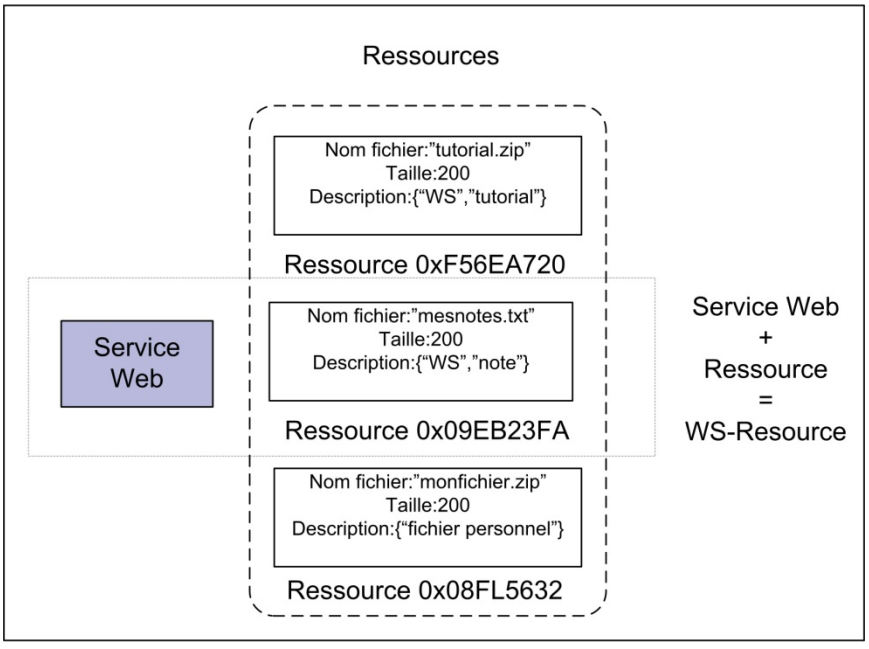

Figure 2.14 Ressource Web.

Les WS-ressources peuvent être créés et détruits, et leur état peut être interrogé ou modifié par des messages SOAP. Les WS-Resources possèdent quatre caractéristiques qui sont très importantes en génie logiciel, connues sous le nom des propriétés ACID.

Atomicité : La manipulation et les mises à jour des ressources doivent être faites avec une transaction atomique qui implémente le principe de *tout ou rien*. Étant donné que les ressources sont dans un contexte réparti, cette condition garantit l'intégrité des données.

Cohérence : Les ressources et leurs états doivent toujours être cohérents, même après des crashes ou des problèmes qui peuvent affecter le bon fonctionnement d'une grille.

Isolation : Les mises à jour de l'état des ressources doivent être isolées au sein d'une unité de travail transactionnel afin que d'autre opérations ne puissent accéder les données qui on était modifiées par des transactions non encore terminées.

Durabilité : Les mises à jour apportées aux états d'une ressource doivent être permanentes.

2.8.9 Les spécifications des WSRF

Les principales spécifications des WSRF sont :

WS-ResourceProperties : Le *WS-ResourceProperties* définit les moyens par lesquels les propriétés d'un WS-Resource peuvent être déclarées dans le cadre d'un service Web. L'une des caractéristiques du WS-Resource est l'ensemble de propriétés associées à la ressource. Cette spécification définit aussi les normes avec lesquelles les propriétés d'une WS-Resource peuvent être incluses dans l'interface du service Web. La déclaration des propriétés de la WS-Resource représente une projection ou une vue sur l'état de la ressource.
Le but de cette spécification est de définir les standards pour uniformiser la terminologie, les concepts, les opérations, ainsi que les fichiers WSDL et XML nécessaires pour exprimer les

propriétés de la ressource pour l'associer à l'interface du service Web. Ces standards visent aussi, à uniformiser les messages qui interrogent la ressource, à fin de mettre à jour les propriétés du WS-Resource

WS-ResourceLifetime : Cette spécification définit le cycle de vie d'une ressource. Elle précise quand et comment la ressource est créée ou détruite. Elle définit également les moyens par lesquels une WS-Resource peut s'autodétruire après l'expiration d'une période donnée de temps (OASIS, 2004).

WS-Notification : Le WS-Notification (OASIS, Web Services Base Notification, 2004) définit le mécanisme d'inscription et de notification à un sujet d'intérêt (Topic) publieur/observateur. Une ressource qui doit faire des notifications doit s'inscrire à un publieur et celle qui veut recevoir des notifications doit s'inscrire à un observateur. Le WS-Notification a été divisé en trois autres spécifications : WS-BaseNotification, WS-BrokeredNotification, et WS-Topics.

WS-BaseFaults : Un concepteur d'une application de services Web utilise souvent des interfaces définies par les fournisseurs qui les exposent via d'autres services Web. Dans ce genre d'application, la gestion des erreurs est problématique du fait que chaque fournisseur traite les erreurs à sa manière. La prise en charge et la gestion des erreurs sont plus faciles quand l'information disponible dans les messages d'erreurs des différentes interfaces est conforme à un certain standard. Le but de cette spécification est d'uniformiser les messages d'erreurs. Le *WS-BaseFaults* propose un schéma XML pour les erreurs de base (OASIS, Web Services Base Faults 1.2, 2006).

WS-ServiceGroup : Un groupe service est un ensemble hétérogène de services Web. Il fournit les moyens pour regrouper les services Web. Les membres d'un groupe service sont représentés en utilisant des composants appelés des entrées. Une entrée est une WS-Resource (OASIS, Web Services Service Group 1.2, 2005).

WS-Addressing : Une fois le mécanisme de découverte a localisé le service Web, il retourne l'adresse du service en fournissant son URI (Uniform Resource Identifiers*). Le WS-Addressing* est un mécanisme qui fournit en plus de l'adresse de la destination, l'adresse de la réponse ainsi que l'adresse ou les messages d'erreurs sont retournés. Le WS-Addressing définit deux mécanismes, le *message addressing properties* et le *endpoint*, avec lesquels le service web peut être invoqué d'une manière transparente et totalement indépendante du protocole de transport, qu'il soit HTTP, SMTP, XMPP ou tout autre type de transport, et indépendant aussi des systèmes de messagerie. Le WS-Addressing est l'une des spécifications les plus importantes du WSRF (W3C, Web Services Addressing , 2004).

2.9 Globus toolkit

Construire un système de grille est un grand défi technologique. Beaucoup de travaux ont été faits pour réaliser les outils qui aident à la mise en œuvre d'un tel système. Le plus connu est sans doute Globus toolkit. Il est considéré comme le pionnier des cadres logiciels (Framework) des systèmes de grille. Globus toolkit est l'un des plus utilisés aussi bien dans des projets de recherche que dans d'industrie. Nous prenons ce Framework comme un exemple d'un système de grille.

L'émergence de l'idée d'une technologie des grilles, et le développement rapide d'infrastructure de réseau à haut débit et à faible coût ont créé le besoin d'un middleware capable d'offrir, aux utilisateurs souhaitant développer des grilles, les outils nécessaires pour mettre en œuvre une telle infrastructure. Dans ce contexte favorable est né le projet Globus toolkit.

Globus toolkit est un projet américain réalisé par l'équipe d'Ian Foster au sein du laboratoire Argonne de l'université de Chicago. L'objective du projet est de fournir un environnement de gestion de grille (midleware ou Framework). À partir de 1997, le Globus Toolkit version 2 (GT2) est devenu un standard pour les technologies de grilles. GT2 offre et implémente un ensemble de protocoles, des API (Application Programming Interface) et des services, sous

forme de boite à outils “toolkit”, permettant de mettre en œuvre un système de grille. Ses services suivent les recommandations de l’OSGA. On y trouve tous les services nécessaires pour répondre à la problématique des grilles telle que l'authentification, la découverte des ressources, la sécurité, etc.

2.9.1 L’architecture de Globus

C’est Ian Foster qui est derrière l'architecture en couche présentée précédemment. Cette architecture est utilisée dans l'implémentation de Globus (Foster, 2001). Tout d'abord, il faut savoir que Globus Toolkit est constitué de plusieurs modules indépendants les uns des autres. Dans Globus toolkit 4 (la version 5 vient d'être lancée au moment de rédiger ce mémoire), on peut distinguer cinq principaux modules, assurant des fonctionnalités essentielles énumérées dans l'architecture proposée par OSGA.

Le premier module est dédié à l'aspect sécurité. Comme toute application distribuée, la sécurité occupe une place primordiale. Pour permettre le partage et l'utilisation des ressources sans risque, le système de sécurité doit protéger les communications effectuées au sein d'une organisation virtuelle. Les outils de sécurité prennent en charge la vérification de l'identité des utilisateurs et/ou des services. Chaque utilisateur doit s'authentifier avant l'utilisation des services de la grille. Ce module s'occupe aussi de la protection de l'intégrité et la confidentialité des communications (protection des messages) et détermine les applications qui sont autorisées à utiliser les ressources (autorisation).

Globus Toolkit 4 offre deux possibilités d'authentification et d'autorisation. Une basée sur un service Web et l'autre est non basée sur le service Web.

Le module de sécurité quant à lui est basé sur l'infrastructure de grille de sécurité (GSI). La plupart des termes et de concepts utilisés dans la description de GSI proviennent de la cryptographie à clé publique.

Le second module s'assure d'une fonctionnalité très importante aussi, à savoir, le transfert de données entre les nœuds de la même grille. La figure 2.15 montre les différents modules de Globus toolkit 4.

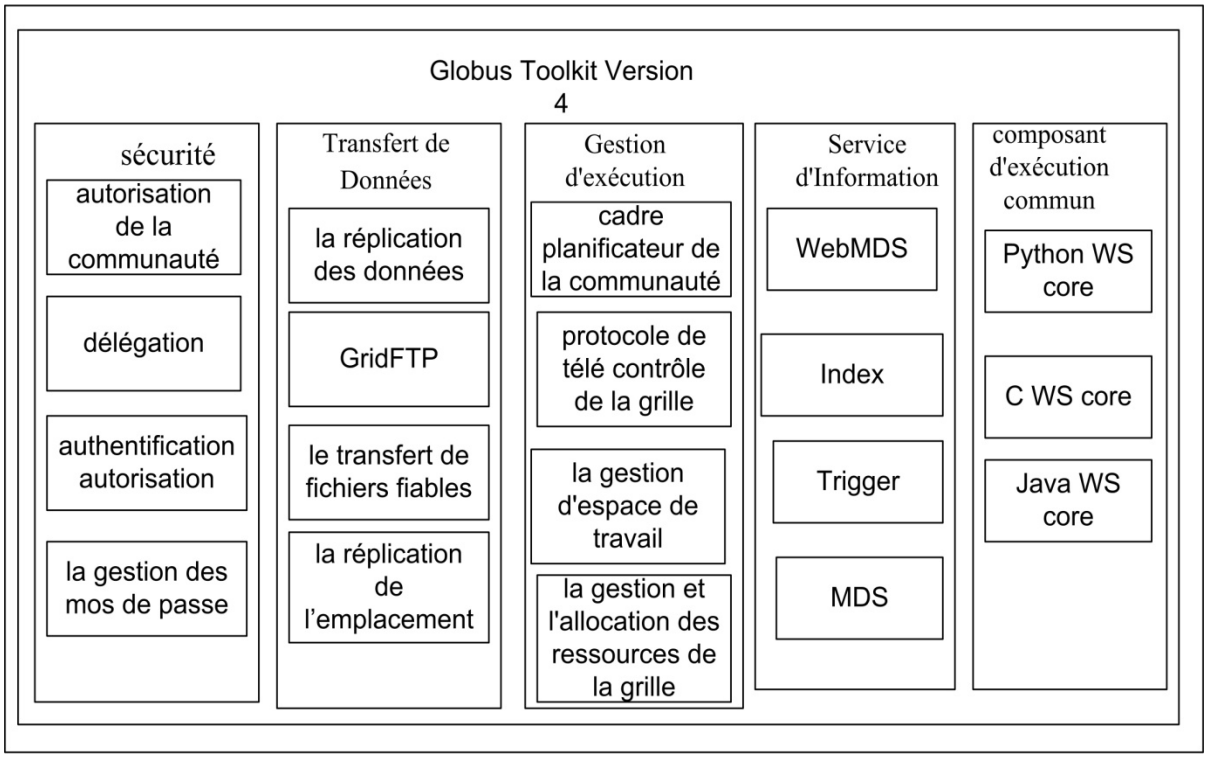

Figure 2.15 Les modules de Globus toolkit 4

Le module de transfert de Données : Pour le transfert de données, Globus toolkit utilise principalement le Grid FTP (Grid File Transfer Protocol). Le Grid FTP est un protocole normalisé, basé sur le protocole classique FTP (File Transfer Protocol) adapté à l'environnement des grilles. Ce protocole permet aux différentes ressources de la grille de transférer des données d'une façon sécuritaire, fiable et efficace. GridFTP offre plusieurs fonctionnalités de transfert de données telles que le transfert parallèle pour augmenter l'efficacité du réseau, la possibilité de la reprise du transfert en cas d'échec ou de détection de fautes, la vérification de l'intégrité des données. Il permet aussi le transfert direct entre deux ressources ou un transfert indirect via une ressource tierce.

Le module de gestion d'exécution : Le module de gestion d'exécution s'occupe des tâches qui sont exécutées dans les différentes ressources dans la grille. Il s'occupe de l'initialisation, la gestion et le contrôle des tâches. Il permet aussi de lister les tâches qui sont soumises pour être exécutées dans une grille, de les exécuter ou de les annuler. Il peut aussi détecter et gérer les problèmes liés aux tâches qui s'exécutent dans la grille.

Le module composant d'exécution commun : Ce module fournit un ensemble de bibliothèques et d'outils qui permettent aux utilisateurs et aux programmeurs de développer des applications de grille. Pour développer leurs applications, les utilisateurs auront le choix entre trois langages de programmation C, Python et Java.

Le module Service d'Information : Le service d'Information connue sous le nom Monitoring and Discovery Services (MDS) est un ensemble de services web dédié à la surveillance et la découverte des ressources et services dans une grille. Il permet aux utilisateurs de connaitre les ressources d'une organisation virtuelle. Le MDS offre une interface pour enregistrer et interroger les informations et les données disponibles sur ressources. Il contient aussi un *trigger* qui peut être configuré pour qu'il se déclenche quand un certain nombre de conditions sont réunies. Par exemple, une fille d'attente sur une ressource qui atteint un seuil précis. Des détails seront donnés dans le chapitre suivant.

2.10 La découverte des ressources dans un environnement virtuel

Les ressources disponibles dans une organisation virtuelle sont caractérisées par la fréquence élevée de changement de leurs états. Une nouvelle ressource peut être créée tandis qu'une autre peut être détruite ou pas accessible, d'autre change de propriété, etc. Par exemple, un serveur qui augmente ses capacités de stockage, ou qui ajoute de nouveaux logiciels, etc. Dans ce contexte un système de virtualisation doit se doter d'un mécanisme performant de découverte de ressources. Typiquement un mécanisme de découverte de ressource doit chercher et trouver les ressources qui correspondent au mieux à l'exécution d'une tâche donnée. En plus de ça, le système doit respecter les critères de base de tout moteur de

recherche à savoir la précision et l'optimisation de temps. Globus toolkit propose le module MDS pour effectuer cette fonctionnalité.

2.11 Monitoring and Discovery Services (MDS)

Ce module est composé d'un ensemble de composantes pour le suivi et la découverte des ressources et services dans la grille. Dans Globus toolkit 4, il porte la même version MDS4. MDS4 utilise les normes et spécifications définies dans WSRF, et le WS-Notification. Son rôle est de collecter les informations sur les ressources disponibles dans une organisation virtuelle et les stocker dans un index. Cet index est accessible via un service web.

Ce système permet aux utilisateurs de trouver et contrôler les ressources qui font partie d'une organisation virtuelle. Les services de MDS fournissent des interfaces qui permettent d'interroger l'index des ressources selon leurs propriétés. Il permet aussi de s'inscrire pour demander une ressource quelconque. MDS donne aussi la possibilité de définir un des *trigger* qui se déclenche quand un certain nombre de conditions sont réunies (Jennifer M. Schopf, 2006).

WS MDS contient des services qui peuvent acquérir leurs informations via une interface extensible qui peut être utilisée pour interroger les fichiers WSRF afin d'avoir les propriétés des ressources.

Comme illustré à la figure 2.16, MDS est un protocole sous forme de sablier qui définit des protocoles standards pour accéder et transmettre les données. Il définit aussi des schémas XML pour la représentation des données, principalement le schéma GLUE. Dans le goulet d'étranglement du sablier on peut voir les différentes interfaces de MDS qui permettent aux utilisateurs d'accéder à la source d'informations, et le service qui transforme leurs schémas de représentation initiale à un schéma XML approprié (GLUE.), qui va être transmis via les normes WSRF et WS-Notification. Toujours dans le goulot d'étranglement, les applications peuvent utiliser un ensemble d'outils sous forme de services Web pour lancer des requêtes,

s'enregistrer, s'abonner et faire des notifications à ensemble de sources de données.

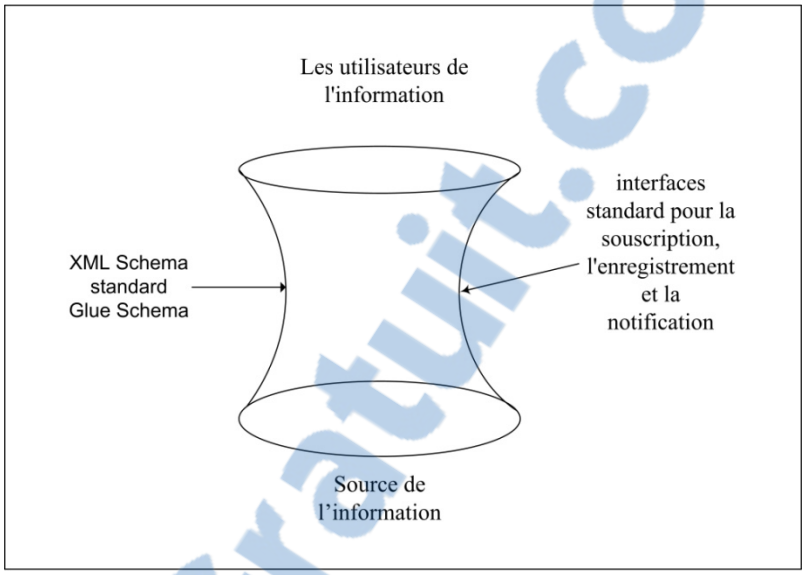

Figure 2.16 Le protocole MDS.

On distingue deux services basés sur un service Web : un service d'index, qui collecte les données à partir de diverses sources et fournit une interface requête/souscription afin d'utiliser ces données, et un service trigger, qui rassemble des données provenant de diverses sources, et qui peut être configuré pour se déclencher en se basant sur ces données.

2.11.1 Service d'index

Le service index de MDS recueille et rend les informations sur les ressources de la grille disponibles sous forme de propriétés des ressources. C'est un registre similaire à l'UDDI (Universal Description, Discovery and Integration) (Lacey, 2004), mais beaucoup plus flexible. En plus de stoker les adresses, ou on peut trouver les ressources, il peut y avoir aussi une copie de ces ressources. Cette copie est mise à jour grâce à un mécanisme de rafraîchissements périodique (Jennifer M. Schopf, 2006). Chaque conteneur Globus qui a installé WS MDS aura automatiquement un index des services par défaut. Tout service GRAM (Gestionnaire d'allocation de ressource), RFT, ou CAS qui est en cours d'exécution

dans ce conteneur s'enregistre automatiquement au service d'index.

2.11.2 Déclencheur (Trigger)

Ce service recueille les données et les compare à un ensemble de conditions définies dans un fichier de configuration. Quand une condition est remplie, une action prend lieu. Par exemple l'envoi d'un message à un administrateur système lorsque l'espace disque sur un serveur a atteint un seuil défini.

Les deux derniers services index et trigger sont implémenté grâce au Framework d'agrégation *Aggregator Framework,* qui fournit un mécanisme commun pour collecter les données.

2.11.3 Aggregator Framework

Le *Framework Aggregator* est un logiciel utilisé pour créer des services qui collectent des données agrégées. Les Services, tels que l'index et les services de déclenchement, construits en utilisant le Framework d'Agrégation sont parfois appelés services d'agrégation. Ces services fonctionnent de la manière (Globus, 2006) suivante :

- Ils recueillent les informations au moyen de l'*Aggregator source*. Un *Aggregator source* est une classe Java qui implémente une interface, définie dans le cadre du *Framework Aggregator*, pour recueillir des données sous format XML.
- Ils utilisent un mécanisme de configuration commun pour maintenir des informations afin de déterminer quelle classe *aggregator source* à utiliser et quels sont ses paramètres associés, qui spécifient généralement les données à obtenir, et de quel endroit ils peuvent être obtenus. Ces services fonctionnent de la même façon que Garbage Collactor de JVM. Si les données d'un enregistrement ne sont pas mises à jour après une certaine période, cet enregistrement est supprimé.

2.11.4 L'interface utilisateur WebMDS

Le WebMDS est une interface web pour visualiser les propriétés des ressources qui sont disponibles dans les fichiers WSRF. Il constitue un outil convivial pour le service d'indexation. Le WebMDS utilise des requêtes pour aller chercher les propriétés des ressources et afficher les résultats dans différents formats.

2.12 La découverte des ressources et la sémantique

Comme on vient de le voir précédemment, la spécification WSRF définit le WS-ResourceProperties pour représenter les données de la ressource (WS-Resource). On y trouve les propriétés de la ressource ainsi que son état. L'accès à ces propriétés est effectué via un certain nombre de messages standards. En utilisant ces messages, on peut consulter, modifier et interroger les propriétés des ressources. Les spécifications WSRF définissent aussi les moyens par lesquels les utilisateurs des ressources reçoivent les mises à jour chaque fois qu'une modification affecte les propriétés des ressources utilisées. Ces mises à jour sont reçues à l'aide des spécifications WS-Notification. Le WS-Servicegroup est une autre spécification de WSRF qui définit les moyens par lesquels les ressources peuvent être regroupées ou agrégées, selon le type des ressources ou selon un domaine bien spécifique. Ce regroupement facilite la découverte des ressources et le suivi de leurs mises à jour.

La découverte des ressources que propose Globus toolkit est basée sur un index où les informations sur les ressources en provenance de plusieurs sources sont collectées, agrégées et rendus disponibles pour les utilisateurs principalement via un service web. Le service de l'index expose les informations sur les ressources sous forme de fichiers XML. Cependant, un aspect important, celui de la sémantique, n'est pas pris en charge par le système de découverte de ressource dans Globus (MDS). MDS ne fournit pas la possibilité de faire une recherche sémantique. Cela peut affecter les performances de la recherche des ressources. Par exemple un utilisateur qui cherche une ressource avec un système d'exploitation Linux, une

recherche par mots clés va éliminer les ressources qui ont des systèmes d'exploitation Ubuntu alors que ces ressources pouvaient répondre au besoin de cet utilisateur du fait qu'Ubuntu est un système qui fait partie de la famille Linux.

Des travaux ont été effectués pour combler ce manque. Dans ce qui suit, nous allons voir les travaux de (Pahlevi & Kojima, 2005), qui propose un Framework pour créer, regrouper et maintenir des informations sémantiques sur les ressources sous forme de métadonnée.

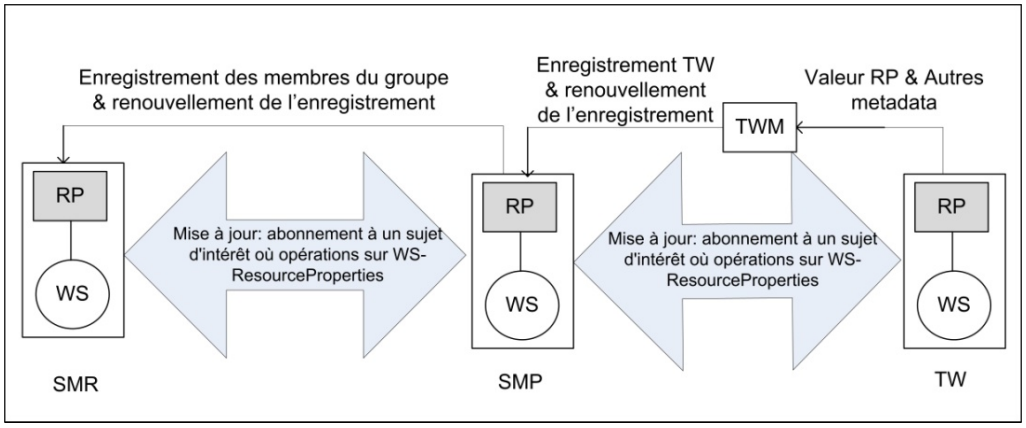

Figure 2.17 Le Framework pour une découverte sémantique.

Dans la figure 2.17 on peut voir les différents composants qui forment le Framework proposé, où chaque ressource (représenté par WS-Resource) cible TW (target WS-Resource) est associée à un service gestionnaire appelé TW Manager (TWM) et à un autre service web appelé fournisseur des métadonnées sémantiques *Semantic Metadata Provider* (SMP). Le service gestionnaire TWM reçoit les métadonnées qui proviennent de la ressource cible TW. Après avoir analysé ces métadonnées, il crée les correspondances sémantiques sous forme de métadonnées sémantiques. Par la suite, le TWM enregistre la ressource cible TW en fournissant, au SMP, les métadonnées qu'il vient de produire. Le TWM prend aussi en charge l'enregistrement des mises à jour de la ressource cible. Le Framework propose d'enregistrer chaque ressource cible dans un groupe qui contient les ressources du même domaine. Le fournisseur sémantique des métadonnées (SMP) enregistre les ressources cibles dans un répertoire, sous forme de ressource web, appelé répertoire des métadonnées sémantiques

Semantic Metadata Repository (SMR). Le fournisseur sémantique des métadonnées (SMP) est une ressource web (WS-Resource) comme le gestionnaire des ressources cible (TWM). Il fonctionne de la même façon. Il enregistre les ressources cibles en fournissant leurs métadonnées sémantiques tout en prenant en charge la mise à jour. Le rôle du répertoire des métadonnées sémantiques *Semantic Metadata Repository* (SMR) est de maintenir les métadonnées qui proviennent du fournisseur sémantique des métadonnées *Semantic Metadata Provider* (SMP). Pour décrire les ressources web, le Framework proposé utilise une ontologie définie pour décrire les propriétés et les capacités des services Web.

Dans (Balachandar R. Amarnath, 2009) les auteurs proposent une approche similaire à la notre. Cependant, cette approche est complètement liée à la technologie des gilles, en particulier aux spécifications WSRF.

2.13 Conclusion

Dans le domaine des technologies d'informations, le besoin en infrastructure de calcul et de stockage ne cesse de grandir. Les entreprises doivent faire face à des défis colossaux, qui nécessitent de grands investissements. La durée de vie des infrastructures est devenue si courte que l'investissement dans de nouvelles infrastructures est un choix à risque. On fait face à une spirale sans fin. Un cercle vicieux où à chaque fois qu'il y a des équipements plus performants chaque fois le besoin grandit. Dans ce chapitre nous avons vu la technologie de virtualisation qui est en train d'émerger et de proposer une solution à ce problème. Cette solution consiste à regrouper des ressources hétérogènes pour former une organisation virtuelle où chaque membre peut avoir l'accès à des ressources matérielles ou logiciels qui ne sont pas disponibles localement.

La technologie des grilles est l'une de ces technologies de virtualisation. Elle repose sur le principe que chaque unité de stockage non utilisée ou chaque unité de calcul non exploitée constitue une perte qui peut se quantifier si on considère l'énergie, le coût des locaux, le coût initial des machines, les techniciens qui sont payés, etc. Cette technologie propose l'idée du

partage des ressources où chaque membre qui appartient à une OV, met à la disposition de cette dernière ses ressources et peut solliciter les ressources disponibles des autres membres. Nous avons vu que cette technologie est basée sur les services Web. Pour créer des ressources possédant des états, cette technologie fait appel au standard WSRF. Ce dernier définit un certain nombre de spécifications. Ces spécifications constituent des informations sur les propriétés des ressources et qui permettent au mécanisme de découverte de trouver les ressources nécessaires à l'exécution des tâches des utilisateurs. Ainsi, nous avons présenté le Framework Globus qui est un ensemble de modules permettant de mettre en œuvre une application de grille. Le système de découverte de ressources dans Globus est basé sur l'index MDS. C'est un ensemble de services qui permet le suivi et la découverte de ressources qui font parties d'une OV. Ces services, inclus dans MDS, fournissent les interfaces qui peuvent être utilisées pour l'interrogation les propriétés de ressources définies selon les spécifications de WSRF. Bien que, les grilles informatiques représentent une grande révolution dans le domaine des nouvelles technologies, leur mise en pratique reste un grand défi et se heurte à plusieurs problèmes techniques qu'il faudra résoudre. Les limites à la réalisation d'un tel système sont imposées, principalement, par la façon dont il est mis en pratique qui repose sur des *Framework* souvent difficiles à utiliser à l'image de Globus. Créer une OV à l'aide de ces outils et Framework rend la tâche plus ardue du fait qu'ils implémentent une gestion des ressources qui repose sur les spécifications de WSRF qui sont lourdes à implémenter. En effet, pour chaque ressource de l'OV un ensemble de spécifications doit être implémentées. La ressource doit gérer plusieurs aspects tels que la sécurité, la communication, le modèle etc. Cette façon de posséder rend l'OV vulnérable aux défaillances des nœuds. Les utilisateurs des grilles doivent faire face à d'autres problèmes tels que la gestion des accès concurrents, la tolérance aux erreurs, etc. Par ailleurs plusieurs façons pour la mise en œuvre d'une OV sont proposées. Un système d'exploitation dédié à la technologie des grilles a vu le jour (Morin, 2007). Ce système part du principe d'intégrer des modules dans un système d'exploitation afin qu'il soit adapté à la technologie des grilles. Alors qu'un système d'exploitation traditionnel gère une seule machine, le système d'exploitation destiné à la technologie des grilles sera capable de gérer plusieurs ressources simultanément. Ce système

est actuellement dans la phase des tests et des versions d'évaluations sont disponibles. En attendant une technologie capable de fournir une mise en œuvre d'une technologie de grille à grande échelle, les nuages informatiques eux ont fait leur preuve pratique. Ils présentent plusieurs avantages par rapport aux grilles informatiques. Dans le chapitre suivant, nous allons présenter les nuages informatiques ainsi que le Framework proposé pour le développement des applications qui offrent de l'infrastructure en tant que service.

CHAPITRE 3

LE NUAGE INFORMATIQUE

3.1 Introduction

Dans le chapitre précédent, nous avons vu la réponse qu'apporte la technologie des grilles à la demande croissante en matière d'infrastructure. Cette technologie propose le partage des ressources non utilisées afin d'augmenter leur rentabilité. Ainsi, la technologie des grilles vient avec l'idée d'interconnecter les ressources (machines) via le réseau Internet et de permettre aux utilisateurs d'aller chercher les capacités de calcul et de stockage nécessaires non disponibles localement; et de ne payer que pour la "quantité consommée" de la même façon qu'un consommateur d'électricité ou de gaz de ville. Le problème qui peut se poser est celui de la surconsommation c.-à-d. un utilisateur qui réserve des ressources dont la capacité est supérieure à son besoin. On peut trouver le même phénomène dans les réseaux électriques : par exemple, la surconsommation de l'électricité à l'heure de pointe peut causer un crash du réseau. Dans ce cas, le but de rentabiliser l'utilisation des ressources au maximum est menacé par certains comportements non responsables ou par ignorance. Une nouvelle forme de virtualisation vient faire face à ce manque. Les nuages informatiques ou "*Cloud computing*" permettent de fournir des ressources à la demande. Les utilisateurs soumettent leurs demandes et les fournisseurs d'infrastructure les traitent en offrant les services nécessaires. Par exemple, un utilisateur peut formuler une demande d'une machine avec un système d'exploitation Linux, une capacité de stockage quelconque et un ensemble de logiciels. Cette requête est prise en charge par le système de virtualisation ; une machine virtuelle est alors créée avec les caractéristiques spécifiées. Ce type de service est appelé infrastructure comme service. Un exemple concret est l'utilisation par le journal New York Times du service Amazon EC2 afin de traiter une grande quantité de données archivées. Cette opération a nécessité 36 h de calcul seulement, avec la création de plusieurs machines virtuelles sur le *Cloud*, au lieu de plusieurs jours même voir des mois si le journal n'avait pas utilisé cette technologie (Myerson, 2009).

3.2 Les nuages informatiques

La technologie de *Cloud computing* permet aux utilisateurs d'utiliser des infrastructures de grandes capacités sans avoir à recourir à un grand investissement. Pour illustrer le concept, considérons un développeur de logiciels. Pour tester la compatibilité de son application, il a besoin de trois plateformes différentes : une Windows, Linux et Mac OS. Au lieu d'acheter trois machines il peut installer une machine virtuelle type VMware qui coutera beaucoup moins cher que de procurer trois machines physiques. Les nuages informatiques ont pour but d'offrir des services qui vont au-delà des applications classiques. Stocker non plus sur un poste local, mais plutôt sur un ‘‘nuage’’ de serveurs. Il s'agit d'une informatique distribuée qui offre l'infrastructure sous forme de services (SOGETI, 2009). Le *Cloud* offre plusieurs avantages pour les entreprises et même pour les simples utilisateurs. Ils n'ont pas à acheter leur propre matériel, tels que les serveurs, les supports de stockage, les routeurs, etc. Ces ressources sont plutôt louées le temps nécessaire à l'exécution d'une tâche particulière. Aussitôt le besoin comblé elles seront libérées. Cela offre plusieurs avantages (Moulard, 2009) :

- Réduire les charges liées à infrastructures et à tout ce qui vient avec. Comme les locaux où le matériel sera entreposé, le cout de l'énergie consommée et le coût de la main-d'œuvre qualifiée qui gère ces installations.
- Faire des économies sur les licences des logiciels. Les utilisateurs ne se soucient plus de la maintenance des infrastructures, la mise à jour des logiciels, etc.
- Les utilisateurs peuvent avoir l'accès à leurs données et leurs applications de n'importe quel endroit via internet.

Le *Cloud computing* peut être catégorisé en trois types selon les services qu'il offre (Sushil Bhardwaj, 2010) à savoir : le logiciel comme service, la plateforme comme service et l'infrastructure comme service.

3.2.1 Le logiciel comme service SaaS

Le *Software as a Servcie* ou *SaaS* est une technologie qui propose des logiciels en ligne sous forme de service. Par exemple, des logiciels de traitement de texte et des tableurs comme Google docs, Microsoft Web App, zoho docs, etc. L'utilisateur a un accès aux différents logiciels ; même s'il ne les a pas installés localement sur sa machine. L'intérêt de cette technologie est d'offrir aux utilisateurs un grand choix de logiciels sans les installer localement, et sans avoir besoin d'acheter des licences, et même si le système d'exploitation est incompatible.

3.2.2 La plate forme comme service PaaS

Le *Platform as a Service* ou *PaaS* quant à lui, propose des solutions plus élaborées que le SaaS. Il propose des plateformes clés en main pour développer essentiellement des applications web. L'utilisateur a sous sa main les outils nécessaires pour développer son application web et la déployer sans avoir de grandes compétences dans le domaine du développement des applications Web. Exemple d'application PaaS AppEngine de Google qui est basé sur Python, Java et Djiango (Ciurana, 2008).

3.2.3 L'infrastructure comme service IaaS

Infrastructure as a Service ou IaaS est utilisée pour fournir de l'infrastructure sous forme de services. Souvent dans un environnement virtuel, l'utilisateur peut commander de l'infrastructure, essentiellement des ordinateurs et des serveurs, des supports de stockage, des routeurs ainsi que des logiciels. Cette technologie permet aux utilisateurs de commander de l'infrastructure d'une façon évolutive selon leurs besoins. Quand, ils ont besoin de plus de ressources ils peuvent en demander. Quand ils en n'ont plus besoin, ils peuvent libérer ces ressources et ne payer que pour le temps qu'ils ont utilisé. Par exemple, un utilisateur peut créer une machine virtuelle avec un serveur Ubuntu comme système d'exploitation et un

serveur Web apache Tomcat afin d'héberger un site Web. Actuellement, il y a deux principaux fournisseurs de cette technologie, il s'agit de Rackspace et Amazon, qui offrent une interface Web pour les utilisateurs afin de créer et de gérer leurs machines virtuelles.

3.3 Architecture d'un nuage de calcul

Comme l'on peut voir dans la figure 3.1 l'architecture générale d'un *Cloud*, comprend quatre principales couches (Nguyen, 2010).

Dans la couche inférieure, niveau système, on peut trouver les ressources physiques telles que des dispositifs électroniques, des supports de stockage, des serveurs, ainsi de suite. Ces ressources sont complètement transparentes aux utilisateurs finaux. En effet, les ressources physiques sont rendues transparentes et complètement virtualisées à l'aide d'un ensemble de logiciels (middleware) et des services qui permettent aux utilisateurs d'utiliser ces ressources et de les partager. La deuxième couche consiste en un middleware. Cette couche offre les applications ainsi que les différents API et interfaces nécessaires à l'utilisation, le contrôle et la gestion des ressources physiques. Elle constitue une boite à outils qui permet de développer des applications qui rendent les ressources virtuelles. Cette couche peut se diviser elle-même en deux sous-couches le corps du middleware qui comporte les interfaces communes et nécessaires à la gestion de tous les types de ressources physiques. Il s'occupe de mettre en œuvre l'IaaS. La deuxième sous-couche consiste au middleware utilisateur. Cette partie offre les interfaces communes qui permettent de fournir la plateforme comme service (PaaS). Quant au rôle de la couche supérieure, consiste à fournir les applications comme service (SaaS).

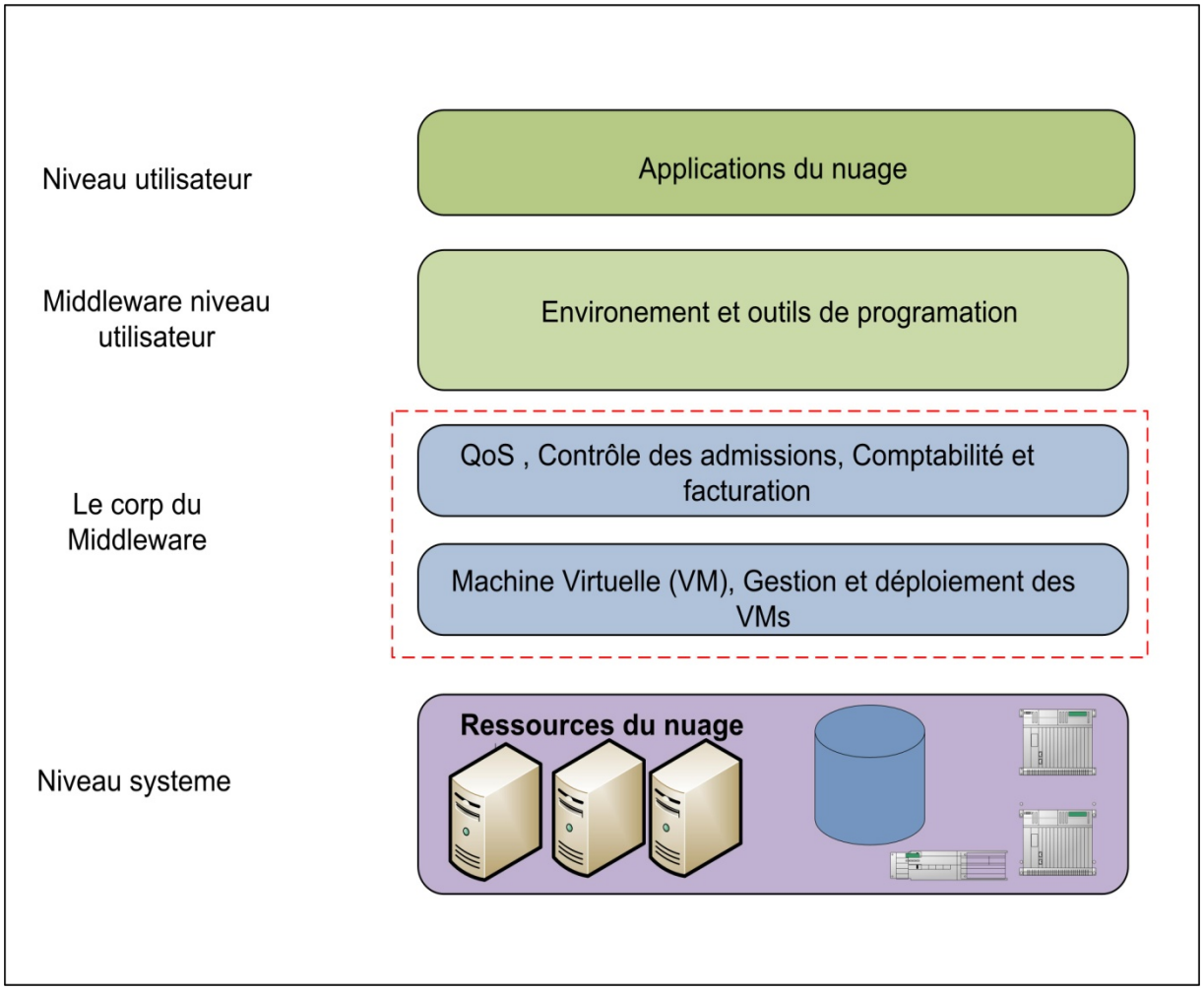

Figure 3.1 Architecture d'un nuage informatique

3.4 Nuage versus grille

Du fait que les nuages informatiques et les grilles informatiques font partie des technologies de virtualisation, il existe beaucoup de similarité entre les deux technologies. Ce chevauchement peut créer de la confusion. Dans ce qui suit, nous allons voir les similarités et les différences entre ces deux technologies.

3.4.1 Similarités

Les deux technologies sont utilisées pour la virtualisation. Le nuage informatique est

considéré comme le fruit de la technologie des grilles ; qui est la pionnière dans le domaine des technologies de la virtualisation. Les deux technologies se rejoignent dans le but de réduire les coûts pour obtenir de nouvelles infrastructures. Elles permettent l'acquisition des ressources à la demande et d'une façon évolutive. En effet, les utilisateurs, dans les deux technologies, peuvent augmenter ou diminuer leurs capacités ou le nombre des ressources selon le besoin.

3.4.2 Différences

Si les deux technologies permettent de partager les ressources, la philosophie n'est pas la même. Dans la grille, un ensemble d'utilisateurs peut accéder et partager une très grande quantité de ressources, alors que dans le nuage informatique un grand nombre d'utilisateurs partage un ensemble limité de ressources. Ce genre de partage rend le nuage informatique plus rentable. Dans la technologie des grilles, un middleware est utilisé pour assembler un nombre de ressources (des disques durs, processeurs, etc.) pour former une ressource plus performante qui possède une capacité plus grande. Chaque partie est considérée comme un nœud. Alors si un nœud tombe en panne ou bien cesse d'exister (le fournisseur le retire) tout le système risque de tomber. Alors que dans la technologie du Cloud computing chaque nœud est géré via un middleware séparé avec un contrôleur global qui permet de contrôler tous les nœuds. Si un nœud tombe en panne, il n'y a aucune incidence sur les autres nœuds.

3.5 IaaS inocybe

Dans le début des années 2000, l'organisme CANARIE (le réseau évolué de la recherche et de l'innovation du Canada) lance le projet User Controlled LightPaths (UCLP) (Grasa, 2008). Ce projet visait à créer un contrôleur pour les réseaux optiques. En 2005, CANARIE lance UCLPv2 un deuxième programme qui visait à créer des éléments d'infrastructure ou de l'infrastructure comme un service pour leurs réseaux optiques. Trois ans plus tard, la compagnie Inocybe technologie (une compagnie montréalaise), le centre canadien de

recherche en communication CRC et i2CAT (le centre espagnol pour la recherche en communication) proposaient le cadre logiciel (Framework) IaaS (IaaS Inocybe est le nom commercial). L'objective est de concevoir un cadre logiciel capable d'offrir les outils nécessaires à la réalisation d'un environnement virtuel complètement indépendant de la boite à outils de Globus toolkit, jugée difficile à utiliser et à adapter.

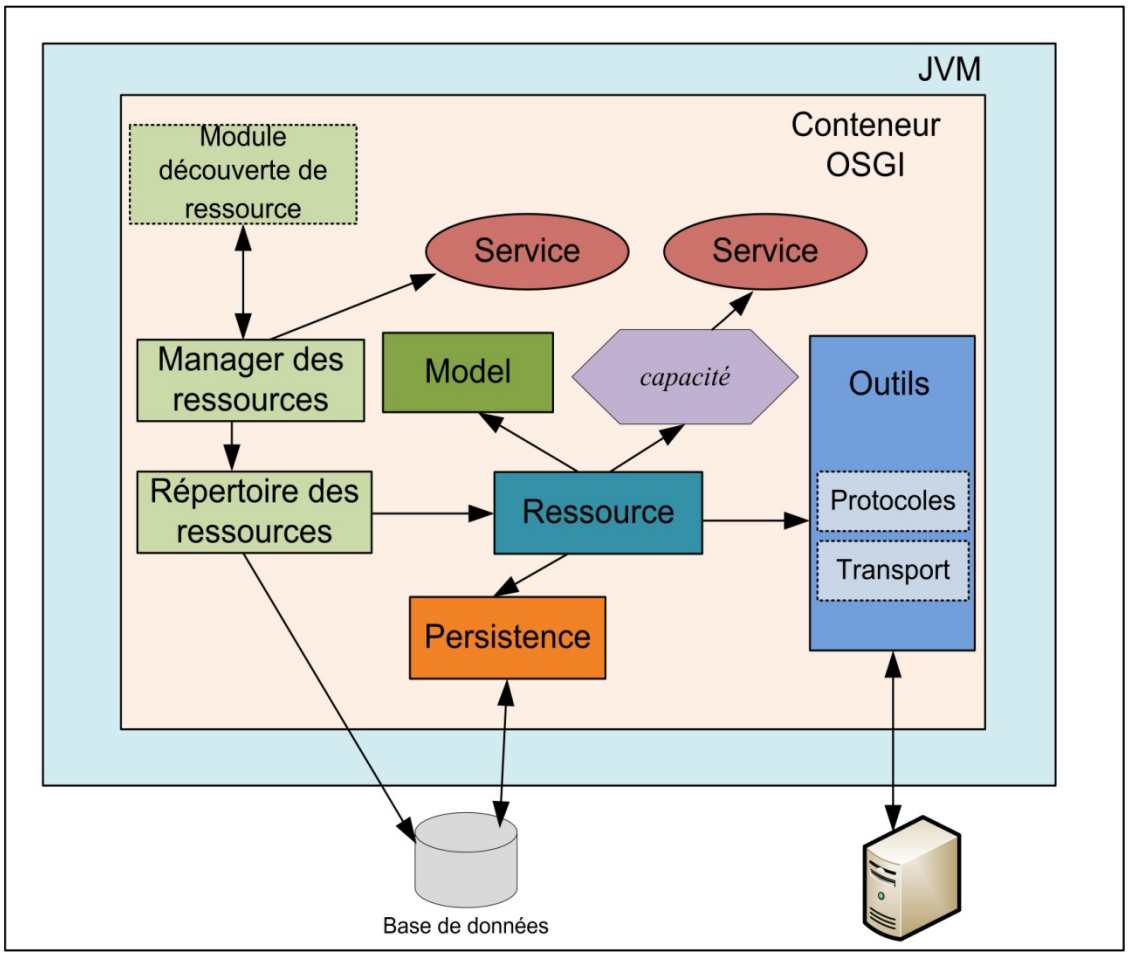

Figure 3.2 Les principaux modules du Framework IaaS

Profitant de l'émergence d'un ensemble de technologies notamment l'OSGI (Open Services Gateway Initiative), IaaS Inocybe offre un nombre de composantes ou modules appelés paquet OSGI. Le paquet OSGI sera présenté plus tard. Ces modules offrent un ensemble d'outils qui servent à la virtualisation des ressources. Chaque module fonctionne

indépendamment des autres. Les développeurs ont le libre choix d'utiliser les modules qu'ils ont besoin ou de surcharge les autres pour mieux les adapter à des besoins spécifiques. La figure 3.2 présente l'architecture globale de la dernière version d'IaaS.

3.5.1 Le module ressource

La ressource est un module central d'IaaS, utilisé lors de l'interaction avec des éléments matériels (périphériques) ou avec des logiciels (applications). La ressource se positionne dans les couches inférieures de l'architecture d'IaaS. Ce module utilise les services offerts par le module Protocole et le module Transport afin d'implémenter ses couches de transport et de protocole qui servent lors de la communication avec les appareils physiques que contrôle la ressource. Le module ressource analyse les requêtes qu'il reçoit et détermine quelles commande ou action à exécuter. Il notifié les couches supérieures en utilisant des déclencheurs et des événements.

3.5.2 L'architecture du module ressource

L'architecture proposée pour le module ressource est une architecture modulaire. Lors de la création d'un module ressource, cette architecture offre beaucoup de flexibilité. Cela permet au développeur de choisir les modules dont il a besoin pour configurer son application afin de refléter le fonctionnement du matériel qu'il veut contrôler. Les différentes composantes d'un module ressource sont :

Le composant transport : Le composant transport est un composant qui sert à établir une connexion avec le matériel physique à contrôler. Il joue le rôle de passerelle entre le protocole et le matériel physique. Les protocoles de transport supporté par IaaS sont TCP, UDP, SSL, Telnet et HTTP/S.

Le composant protocole : Pour communiquer avec certains matériels, il est nécessaire de

formater les messages selon un protocole bien défini par le fabriquant. Ce composant permet de créer ou d'utiliser les protocoles qui sont déjà offerts par IaaS. Il fournit les moyens de créer des objets et de les envoyer d'une façon synchrone ou asynchrone. Un exemple de protocole est le Command Line Interface Protocol (CLIP) qui est un protocole générique pour analyser les lignes de commandes. Ce protocole est utile notamment lors de la communication et l'envoi des commandes vers des dispositifs physiques.

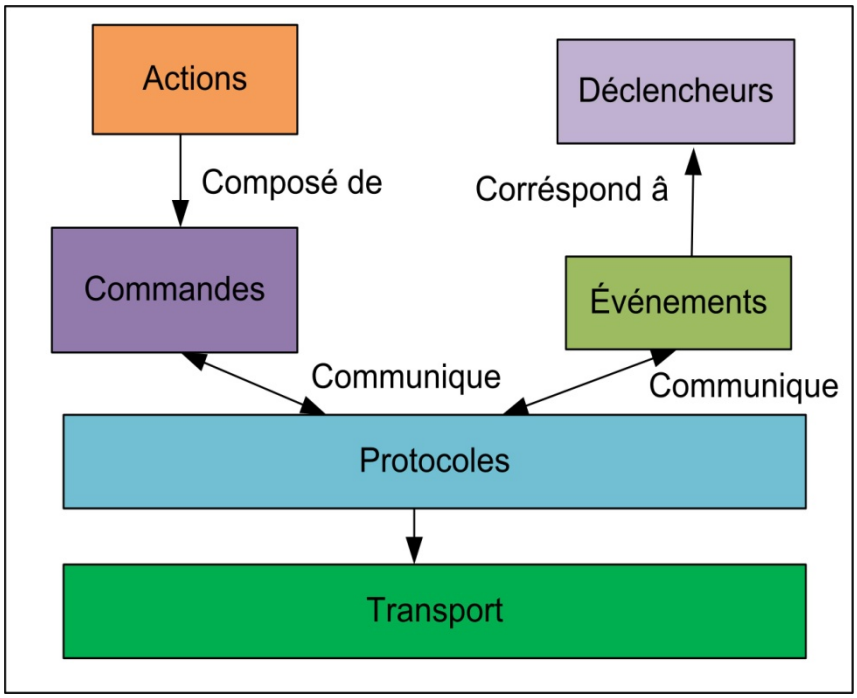

Figure 3.3 Une ressource IaaS.

Le composant commande : ce composant offre la possibilité de créer les commandes supportées par le matériel qui est contrôlé par la ressource. Chaque commande fait partie d'un ensemble de commande (CommandSet). La ressource peut avoir un ou plusieurs ensembles de commandes pour supporter les différents types du matériel ou les différentes versions du micro logiciel. Chaque commande doit être implémentée dans une classe séparée afin de faciliter la maintenance et la gestion des versions.

Le composant action : Certaines fonctionnalités de matériel nécessitent l'envoi de plusieurs

commandes simultanément. Le rôle de ce composant est l'envoi de multiples commandes à la fois.

Le composant événement : Un événement fournit une réponse à un message de notification généré pour le matériel géré par la ressource (comme un signal d'alarme par exemple).

Le composant déclencheur : les déclencheurs sont liés aux événements. Le but des deux composants est le même. Ils servent à notifier les couches supérieures. Chaque matériel physique peut avoir un ensemble de déclencheurs, et chaque déclencheur est lié à un ou plusieurs événements. Par exemple, si certains événements se produisent souvent, le déclencheur les détecte et prend l'action adéquate, comme envoyer un courriel.

3.5.3 Le module modèle

Ce module est utilisé pour représenter l'état actuel de la ressource. Par défaut ce module n'exécute aucune action. Il est considéré comme conteneur d'information. Par exemple, il peut contenir l'état d'un appareil (en marche/arrêter), la taille de la mémoire utilisée dans le cas échéant, etc.

3.5.4 Le module capacité (capability)

Les capacités (capabilities) sont des modules qui offrent des services utilisés par les ressources. Principalement, les capacités sont utilisées par les ressources, pour offrir les opérations génériques. Par exemple, la fonction d'archivage, la mise à jour des données modèles etc. Des nouvelles capacités peuvent être ajoutées et créer selon le besoin.

3.5.5 Le module persistance

Le module persistance fournit une couche qui est construite au-dessus du Framework Spring DAO (Data Access Object) afin de permettre aux modèles de sauvegarder les informations dans une base de données.

3.5.6 Le module service

Ce module constitue une couche située complètement en haut dans l'architecture d'IaaS. Il occupe une place primordiale. Le module service offre les interfaces nécessaires pour les applications des utilisateurs. Un service est un paquet OSGI qui fournit une vaste gamme d'interfaces pour les services Web exemple SOAP, REST, RMI.

3.6 Open Services Gateway Initiative (OSGI)

OSGI pour *Open Services Gateway Initiative,* est une technologie qui a émergé en 1999 pour répondre aux besoins des dispositifs électroniques intégrés et leurs logiciels qui font partie de la programmation dite embarquée (Daniel R. G., 2009). La puissance d'OSGI réside dans le fait que ses programmes se devisent en un ensemble de petits modules qui s'appellent *paquet* ou *Bundle* (dans ce qui suit, on appelle paquet tout module qui fonctionne dans une plateforme OSGI.). Chaque paquet peut être installé, désinstallé, mis à jour, démarré et arrêté dynamiquement et indépendamment des autres paquets. Chaque paquet dans un conteneur OSGI offre des services à l'intérieur du conteneur OSGI. Dans le contexte de l'émergence de la technologie Web et en particulier le langage Java, OSGI apporte les choses qui manquaient dans la technologie Java. Parmi les caractéristiques qui ont fait le succès d'OSGI on peut citer :

Déploiement dynamique des paquets : La plus importante caractéristique d'OSGI est le déploiement dynamique des *paquets*. Le déploiement dynamique des applications consiste à

déployer ses applications sans interrompre le fonctionnement de la plateforme ou le conteneur OSGI. Ce qui permet d'installer, de mettre à jour (modifier le code), d'arrêter et de démarrer les applications sans affecter le bon fonctionnement des autres applications ou modules qui utilisent les services de cette application. Cette fonctionnalité a un intérêt majeur dans les applications embarquées. Imaginons qu'un groupe de développeurs travaillent sur le logiciel de la même machine, un robot par exemple. Chaque développeur travaille sur un composant particulier et qui est en constante interaction avec les autres composants développés par ses collègues. Si à chaque fois que le code est changé toute la machine doit être redémarrée pour que la nouvelle version prenne effet ce qui affectera beaucoup le développement. Aussi, si par exemple, la machine est dans un domaine sensible comme la médecine si un composant tombe en panne la machine continue à fonctionner en attendant la réparation de cet élément, ce qui minimise l'impacte des défaillances.

L'auto-découverte des services : La deuxième chose que propose la technologie OSGI est l'auto-découverte des services. Ceci permet à n'importe quel module d'utiliser les services disponibles dont il a besoin. Par exemple, une machine peut utiliser des programmes d'autres machines. En effet, les services offerts par un paquet à l'intérieur d'un conteneur OSGI peuvent être utilisés par plusieurs projets. Une chose qui est très intéressante pour la programmation embarquée du fait que l'espace mémoire est de taille limitée.

La coexistence de plusieurs versions : La troisième chose et qu'il peut exister plusieurs versions du même paquet. Cette option permet la mise à niveau (Upgrade) des applications OSGI en toute souplesse. L'architecture modulaire d'OSGI qui est effectuée via l'enregistrement et la consommation des services entre les paquets rend le système faiblement couplé. Un changement sur un paquet aura de faibles conséquences sur le système global.

3.6.1 Le paquet OSGI

Le paquet est l'élément central de la technologie OSGI. Un paquet est formé d'un groupe de classes Java, d'un ensemble de fichiers appelés ressources et d'un fichier bien particulier nommé *manifeste*, ainsi qu'un ensemble de services nécessaires pour donner aux classes Java qui appartiennent à ce paquet, des comportements plus sophistiqués (Daniel R. G., 2009).

Le paquet déployé dans un conteneur OSGI prend la forme d'un fichier JAR (Java Archetype). Malgré que le paquet soit un fichier JAR, du fait qu'il réside dans un conteneur OSGI, il est isolé de tous les autres fichiers qui roulent dans la machine virtuelle de Java. La communication avec le monde extérieur s'effectue uniquement via le fichier *manifeste*.

3.6.2 Le fichier manifeste

Le fichier manifeste est un simple fichier texte attaché à un paquet OSGI. Il se trouve dans le répertoire META-INF et porte le nom MANIFEST.MF. Il contient les informations du paquet. Ces informations seront reconnues par la plateforme OSGI et c'est leurs interprétations qui donnent le comportement dynamique à un paquet OSGI. La figure suivante présente un exemple d'un fichier manifeste.

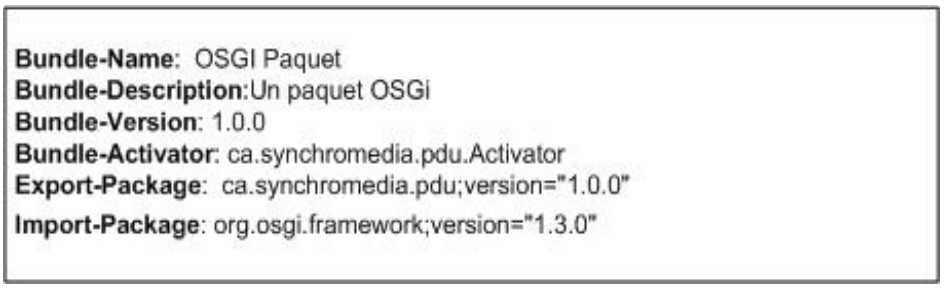

Figure 3.4 Exemple d'un fichier manifeste.

Dans la figure 3.4 on peut voir différents paramètres d'un fichier manifeste. Il existe d'autres qui sont généralement optionnels. Dans un fichier manifeste, on peut trouver les paramètres suivants :

Bundle-Name : Pour assigner un nom au paquet, c'est ce nom qui sera affiché dans le conteneur OSGI.

Bundle-Description : Comporte une brève description du paquet.

Bundle-Version : Désigne la version du paquet.

Bundle-Activator : Indique le nom de la classe qui sera invoqué en premier lieu lorsque le paquet démarre. C'est en quelque sorte le point d'entrée du paquet.

Export-Package : Indique les packages Java (les services) du paquet OSGI qui seront exposés et disponibles pour le monde extérieur.

Import-Package : Exprime les packages Java que le paquet OSGI aura besoin du monde extérieur afin de satisfaire ses dépendances.

Le comportement dynamique du paquet OSGI est rendu possible grâce aux deux paramètres *Import-Package* et *Export-Package*. Contrairement à un fichier JAR ordinaire, le paquet OSGI lors de l'archivage, lui-même au début un fichier JAR, ne comprend pas les librairies et les services qu'il utilise. Ces derniers seront découverts dynamiquement par le paquet. Par exemple si un paquet utilise une librairie spécifique aux manipulations des fichiers XML. Lors de l'archivage cette librairie ne sera pas ajouter au fichier JAR qui représente le paquet mais seulement spécifiée dans les *Import-Package*. Le conteneur OSGI fournira cette librairie à ce paquet. De la même façon, les services offerts par le paquet sont disponibles dans le conteneur OSGI et seront découverts dynamiquement par les autres paquets qui les

utilisent et qui roulent dans le même environnement OSGI. Cette option permit de limiter la taille des paquets OSGI et ainsi économiser l'espace mémoire utilisé. Il faut noter aussi que les packages importés ou exportés comportent un numéro de version, ce numéro n'existe pas dans le cas ou le paquet est unique.

3.7 Le Framework OSGI

Le Framework OSGI constitue l'environnement ou les paquets OSGI sont exécutés. Le Framework OSGI offre un ensemble de services qui supportent le fonctionnement des paquets OSGI. La figure 3.5 montre les différentes couches du Framework OSGI.

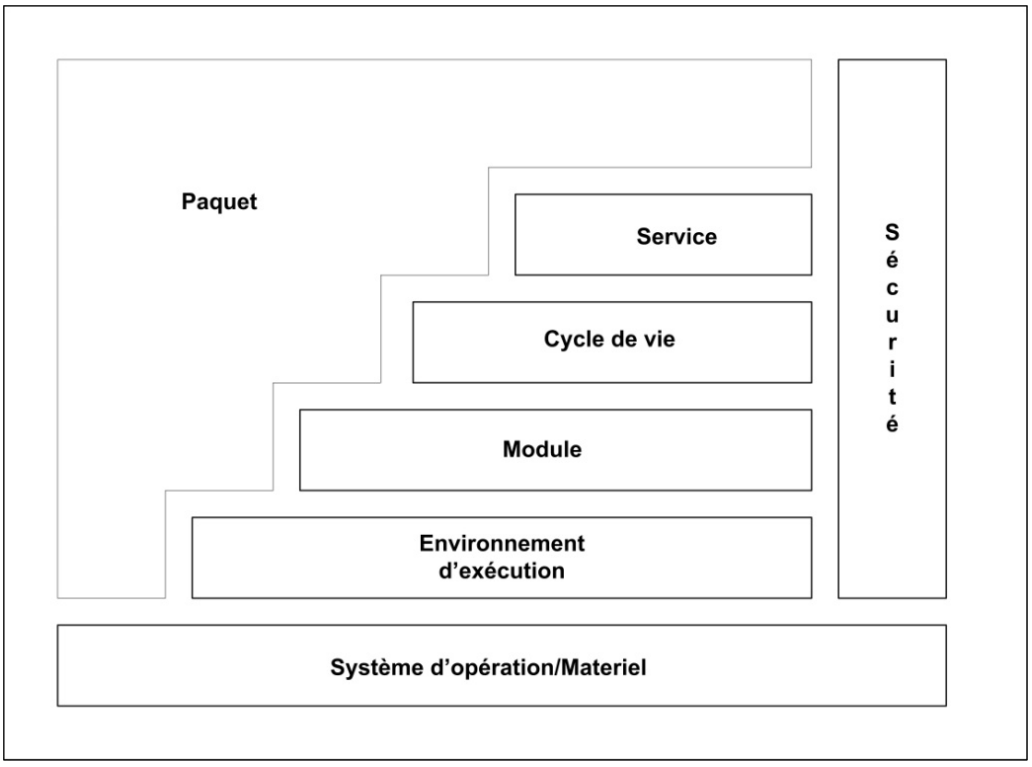

Figure 3.5 Les différents services du Framework OSGI.
Tirée de (Daniel R. G., 2009)

Un environnement OSGI offre un cadre d'exécution pour les différents paquets en fournissant un ensemble de services. Les paquets qui s'exécutent dans un conteneur OSGI

auront la possibilité d'utiliser les services offerts par les différentes couches d'un conteneur OSGI.

3.7.1 La couche sécurité

Cette couche s'occupe de tout ce qui concerne l'aspect sécurité des paquets à l'intérieur du conteneur OSGI. Basée sur l'architecture de sécurité Java. Cette couche définit pour chaque paquet OSGI la façon avec laquelle une signature numérique est attribuée et quels sont les contrôles de sécurité appliqués. La couche de sécurité n'offre aucune API additionnelle qui peut être utilisée par les développeurs. En ce qui concerne les contrôles de sécurité proprement dits, ils sont intégrés avec les services OSGI qu'offre la couche qui gère le cycle de vie d'un paquet.

3.7.2 La couche module

La couche module définit les caractéristiques modulaires nécessaires au Framework lui-même. Cette couche définit entre autres l'anatomie des paquets, les valeurs des paramètres du fichier manifeste que le Framework doit prendre en charge, etc.

3.7.3 La couche cycle de vie

Cette couche définit et gère les différents états possibles qu'un paquet OSGI peut avoir. Comme on l'a déjà mentionné auparavant, la gestion d'un paquet OSGI s'effectue d'une manière distante. L'installation, la désinstallation, le démarrage, l'arrêt et la mise à jour ne nécessitent pas le redémarrage de la plateforme OSGI. Un paquet OSGI peut être dans l'un des états suivants illustrés par la figure 3.6.

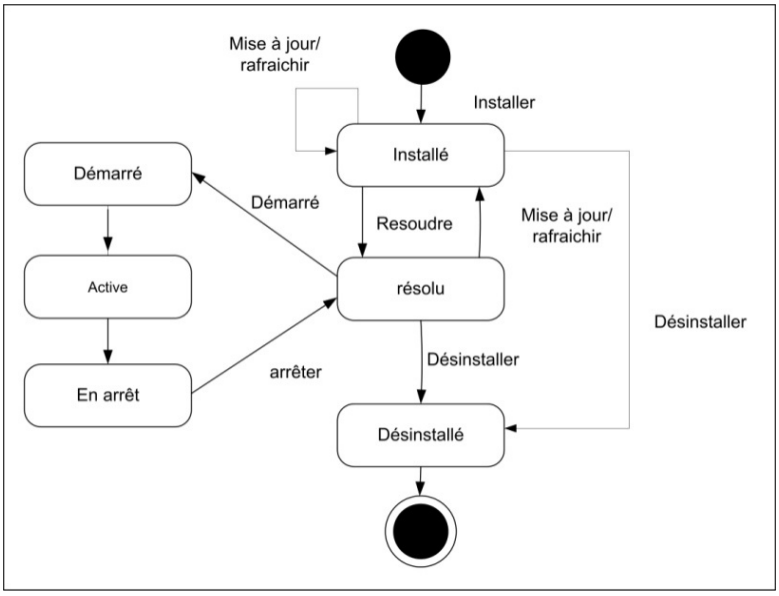

Figure 3.6 Les différents états d'un paquet OSGI.
Tirée de (Daniel R. G., 2009)

Installé : Indique que le paquet OSGI a été validé, installé et il est prêt à être activé.

Résolu : un paquet dans un état résolu indique que toutes ses dépendances déclarées dans le fichier manifeste (import-package) ont été résolues.

En démarrage : un paquet en état de démarrage est dans le processus de transition vers un état actif.

Actif : l'état actif indique que le paquet a été chargé dans le conteneur OSGI, et ses services peuvent être invoqués.

3.7.4 La couche service

Un service OSGI est une instance d'objet Java, enregistré dans un Framework OSGI et mis à la disposition des autres paquets. N'importe quel objet Java peut être enregistré en tant que service. En général, un service est exposé via une interface JAVA. Il est recommandé de définir une interface Java et de déclarer une classe qui implémente cette interface. Ce découplage permet de limiter l'impact des changements sur les services.

Voici quelques exemples des services fournis par un Framework OSGI :

Journalisation : connu sous le nom de *Logging*. Ce service permet l'enregistrement (ou l'affichage selon la configuration) chronologique des données.

Service http : permets l'envoi et la réception des données en utilisant le protocole HTTP.

L'analyse syntaxique XM : Ce service fournit les outils pour que les API spécifiques à XML soient utilisées dans la plateforme OSGI.

Gestion des utilisateurs : ce service sert à la gestion des utilisateurs notamment les autorisations qui sont accordées.

Le suivi des services : ce service permet le suivi des services, il sert à détecter si un service vient de s'enregistrer.

3.8 Conclusion

Dans ce chapitre nous avons introduit les nuages informatiques. Cette technologie consiste à fournir aux utilisateurs des ressources et des services à la demande, contrairement à la technologie de grilles qui fourni des ressources sans regard à leur utilisation (Chen, Zhang, & Huo, 2010). Ceci élimine le surapprovisionnement, notamment quand les services sont facturés. Le nuage informatique peut offrir des logiciels comme service SaaS ou bien des plateformes comme service ou bien une infrastructure complète comme service IaaS. Un IaaS offre des services qui comportent des ressources matérielles et logicielles. Nous avons présenté le Framework IaaS d'inocybe proposé pour mettre en œuvre des organisations virtuelles pour offrir des infrastructures en tant que service basées sur le principe des nuages informatiques. Ce framework est caractérisé principalement pour la simplicité qu'il offre pour la mise en œuvre d'une OV. En effet, la gestion des ressources repose sur le principe d'une ressource, d'un modèle et un d'un ensemble de capacité.

Tableau 3.1 IaaS inocybe VS Globus

	déploiement	**Gestion des ressources**	**développement**
IaaS	Déploiement facile sur un conteneur OSGI.	Basé sur le principe des capacités et un modèle de ressources.	Java.
Globus	Processus d'installation long et compliqué.	Basé sur WSRF.	Java, C, Python.

Le tableau 3.4 présente une comparaison entre IaaS inocybe, qui est développé pour être utilisé dans le domaine des nuages informatiques, et Globus toolkit, qui est adapté aux grilles informatiques. La différence réside dans la gestion des ressources. En effet, dans Globus la ressource implémente un ensemble de spécifications. Par conséquent, la ressource contient et met à jour une grande quantité d'informations. Alors que dans IaaS inocybe, en utilisant le principe modulaire proposé par la technologie OSGI, la ressource délègue plusieurs tâches à différentes composantes spécialisées. Par exemple, les propriétés de la ressource sont déléguées au module modèle. Quant à la gestion et la découverte des ressources dans Globus, elles sont effectuées en utilisant un index appelé MDS. Concernant la gestion et la découverte des ressources dans IaaS inocybe seront présentées dans le chapitre suivant.

CHAPITRE 4

LE CADRE DE DÉCOUVERTE DE RESSOURCE WEB BASÉ SUR L'ONTOLOGIE ET LE RÉSEAU BAYÉSIEN

4.1 Introduction

Les ressources dans un environnement virtuel sont très diversifiées. On y trouve plusieurs types de ressources, aussi bien matérielles que logicielles, qui diffèrent par leurs natures et leurs types. La découverte des ressources est une tâche difficile, mais cruciale. Le problème de la découverte des ressources web est dû principalement à deux facteurs. Premièrement, à la façon dont les ressources sont décrites dans les Framework utilisés actuellement pour la mis en œuvre des OVs. Ils adoptent une façon simple pour la représentation des ressources en regroupant les informations sur ces ressources sous forme de métadonnées qui n'offrent pas des connaissances sur la nature proprement dite des ressources. Le deuxième facteur est dû au nombre élevé d'intervenants dans une OV. Chaque intervenant peut utiliser des mots clés différents de ceux utilisés par les autres intervenants pour décrire les mêmes ressources. Alors, il est important d'adopter une méthode de représentation qui permet un raisonnement sémantique. Ainsi, un système de découverte de ressources basé sur l'inférence sémantique peut régler le problème des ressources sémantiquement identiques mais qui sont décrites avec des mots clés différents. Dans ce chapitre nous présentons une méthode basée sur l'ontologie pour décrire les ressources et sur les réseaux sémantiques Bayésiens pour la découverte des ressources qui prend en charge l'aspect sémantique.

4.2 Les spécifications de la recherche des ressources

Les concepteurs d'IaaS Inocybe ont déterminé un ensemble de spécifications et de méthodes avec lesquelles les utilisateurs du cadre de développement IaaS auront la possibilité d'interroger l'index des ressources afin de faire des recherches sur ces dernières.
Les utilisateurs auront le choix de chercher les ressources de différentes façons :

Une recherche plaine texte, ou bien une recherche par exemple, interroger selon des critères ou bien juste accéder à l'index et faire une recherche manuelle séquentielle.

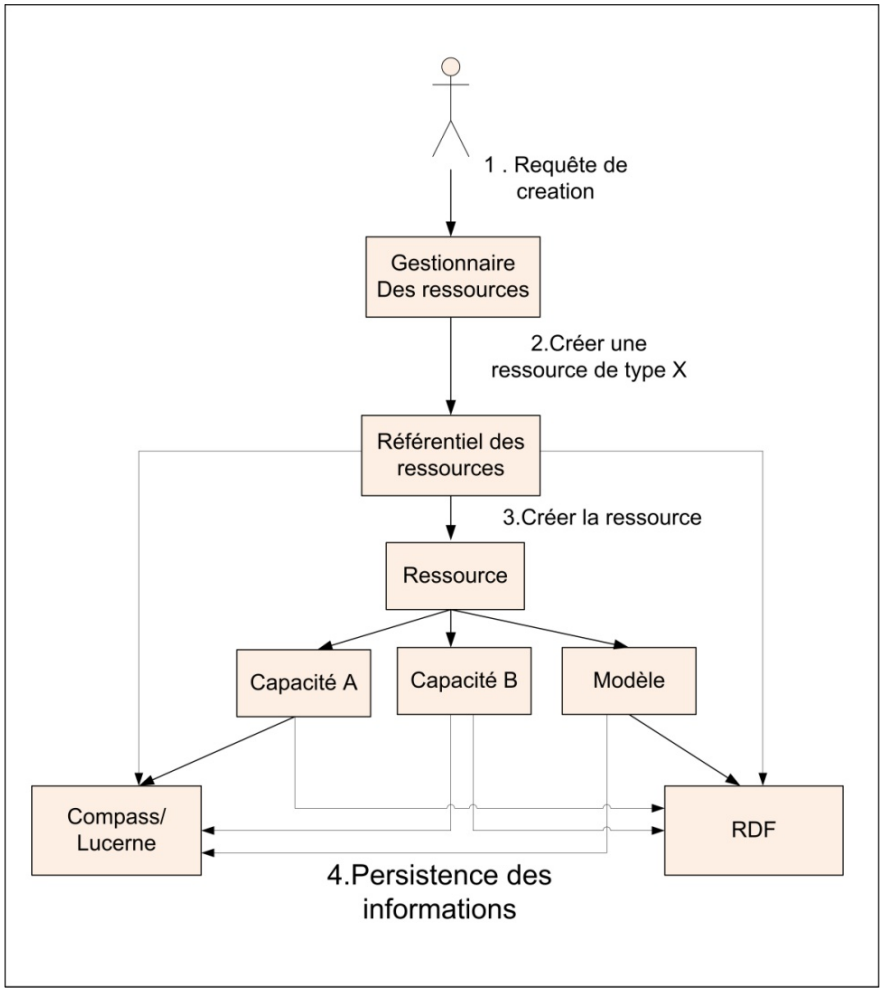

Figure 4.1 Procédure de création d'une ressource.

Au cours du développement, nous avons implémenté la dernière solution. Pour créer, manipuler ou utiliser une ressource, les utilisateurs autorisés peuvent accéder à une interface graphique et visualiser les ressources disponibles. Pour la recherche plaine texte, le choix a été porté sur le moteur de recherche Java Compass/Lucene (Hardy, 2008). Pour la recherche selon les critères, nous avons proposé notre méthode qui est capable de prendre en charge la recherche sémantique. La figure 4.1 montre la procédure de création d'une ressource. Un administrateur ou tout autre utilisateur autorisé peut soumettre une requête pour créer une

ressource, via le gestionnaire des ressources. Le système prend en charge cette requête et crée la ressource dans un répertoire appelé répertoire des ressources. Les métadonnées de cette ressource sont enregistrées dans deux endroits : un index géré par l'utilitaire Compass/Lucene et l'autre est un fichier RDF créé et géré par le Framework que nous proposons. Le grand inconvénient de l'utilisation du moteur Compass/Lucene réside dans le fait que son résultat doit être interprété manuellement. Le moteur ne comporte aucune intelligence et ne retourne que les ressources qui correspondent exactement aux mots clés utilisés dans les requêtes des utilisateurs. Quant à notre Framework, basé sur l'ontologie, permet non seulement de déterminer la ressource recherchée, mais aussi d'avoir des informations sur le type et les attributs de celle-ci grâce à une modélisation et une recherche sémantiques.

Avant de détailler notre méthode, nous présentons brièvement la problématique de la sémantique dans le Web.

4.3 Le Web sémantique

Le web a été conçu pour faciliter la communication, sur Internet, entre l'homme et la machine. La première génération du Web était basée sur des pages web écrites manuellement en langage HTML. La deuxième génération a vu naitre des outils qui permettent de générer les pages web automatiquement. Ces deux générations de web sont destinées à l'interaction humaine. Les formes utilisées permettent uniquement à l'être humain de comprendre les pages web et de tirer profit. Uniquement l'être humain peut naviguer entre ces différentes pages, remplir les champs, interpréter les données et ainsi de suite. Ce type de Web est appelé Web syntaxique constitué d'un ensemble de pages web interconnectés entre elles. Ces pages Web contiennent un grand nombre de données sous forme multimédia, textes, vidéos, Base de données, etc. Les machines s'occupent des tâches basiques non intelligentes à savoir la présentation tandis que l'être humain s'occupe des tâches plus complexes telles que l'interprétation et l'analyse des données, le traitement de l'information, l'établissement des liens entre les pages Web et leurs contenus, etc. Mais les êtres humains ont tendance à

déléguer, aux machines, les tâches difficiles et répétitives. Sauf que les machines ne possèdent pas les mêmes capacités d'interprétation et de raisonnement que possèdent les humains. Dans notre cas, une recherche efficace des ressources web peut impliquer des connaissances préalables. Exemple, trouver '' un serveur, avec un système d'exploitation Ubuntu, et qui utilise l'énergie renouvelable'', il serait difficile de déléguer un agent de recherche sur le web pour trouver et réserver ces ressources avec un ensemble de critères qui ne sont pas évidents pour une machine. Par contre pour un être humain, une telle recherche est plus au moins facile si toutes les informations sont affichées dans le même endroit. Dans le cas où les informations sont dans plusieurs endroits distants ou qui impliquent un nombre très élevé de données à analyser et à combiner, la complexité de cette tâche augmente exponentiellement. Il est plus approprié d'automatiser ce genre d'opération. Mais le problème qui se pose c'est que le Web syntaxique ne supporte pas de telle performance. La solution alors est proposée par le Web sémantique. Le Web sémantique permettra la création des services intelligents tels que les filtres, les systèmes de recherche, etc. Atteindre l'objectif qui permet à une machine d'offrir des services capables de traiter intelligemment le contenu du web nécessite d'autres standards qui ne prennent pas en charge uniquement l'aspect syntaxique des documents, mais aussi l'aspect sémantique (Decker, et al., 2000).

Pour modéliser les connaissances dans les ressources web, le W3C a introduit un modèle en couche figure 4.2, avec ce modèle, le Web est vu comme un vaste réseau interconnecté par des liens sémantiques.

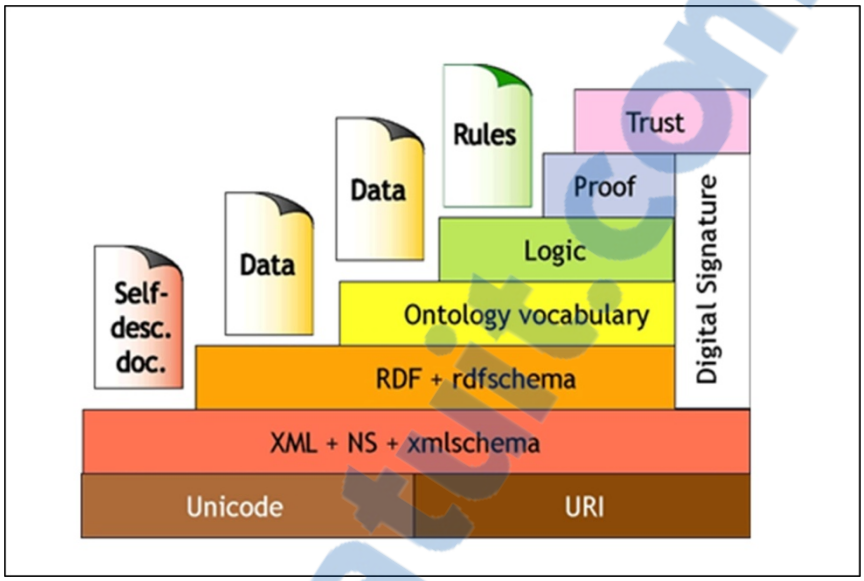

Figure 4.2 Modèle en couche du Web proposé par le W3C. (Schliefnig, 2009)

On distingue trois couches essentielles qui traitent de l'aspect sémantique :

- La couche ontologie qui est responsable de la sémantique et le raisonnement proprement dit.
- La couche représentation de données. La couche d'ontologie fait appel à la couche représentation de données. Cette couche représente les données sous forme d'objet RDF.
- La couche représentation de données à son tour utilise les normes XML et Name Space et XMLShema définit dans la couche qui est responsable des normes et formats d'échanges de données.

4.4 RDF

RDF est un modèle standard pour la représentation des ressources sur le Web. Son rôle et de fournir les métadonnées pour la description des ressources. La puissance du format RDF réside dans le fait qu'il fournit les métadonnées pour décrire les ressources et en même temps il fournit une représentation des ressources.

RDF décrit les ressources à l'aide d'un triplet Objet-Attribut-Valeur. Cette description n'est autre qu'un graphe orienté, où les nœuds sont des objets ou des valeurs, et l'arc qui relie deux nœuds est un attribut qui constitue un lien sémantique ou vocabulaire entre l'objet et sa valeur. Cette relation est appelée déclaration et elle peut être écrite ‘‘Attribut (Objet, valeur)’’.

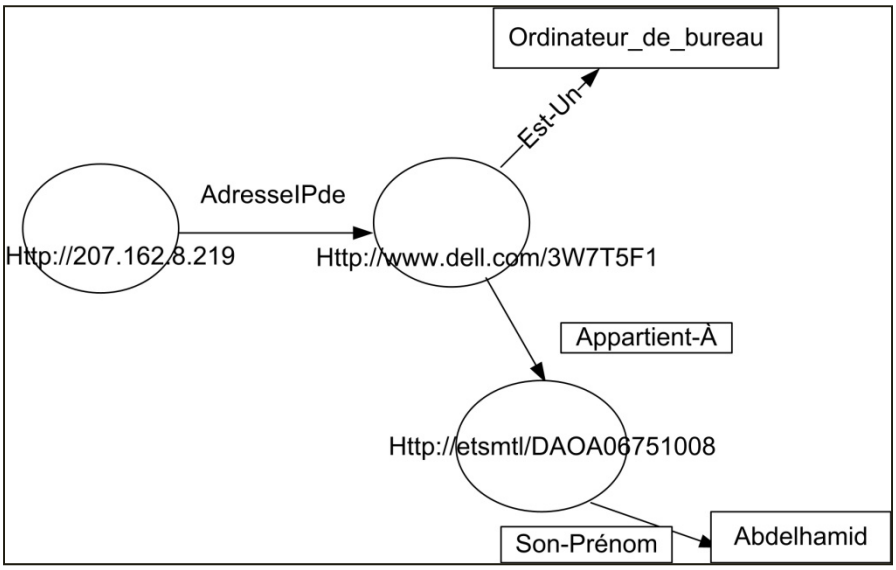

Figure 4.3 Un graphe RDF.

La ressource est tout objet qui possède un URI (Uniform Resource Identifier). Tout objet dans une déclaration peut être une valeur dans une autre. Dans la figure 4.3 on distingue les déclarations suivantes :

AdresseIPde(‘‘http://207.162.8.219’’, ‘‘http://www.dell.com/3W7T5F’’).

Est-un(‘‘http://www.dell.com/3W7T5F’’, ‘‘Ordinateur_de_bureau’’)

Appartient-À (‘‘http://www.dell.com/3W7T5F’’, ‘‘http://etsmtl.ca/DAOA06751008’’)

Son-Prénom (‘‘http://etsmtl.ca/DAOA06751008’’, ‘‘Abdelhamid’’).

4.5 Méthodologie

Afin d'atteindre notre objectif, notre méthodologie comporte les étapes suivantes : la

première étape consiste à modéliser notre domaine en le décrivant à l'aide d'une ontologie. Dans cette étape, les différents concepts qui décrivent le domaine sont dégagés. Ceci permet de définir une taxonomie de classes, et il permet aussi d'énumérer les termes les plus importants. Ensuite vient l'étape de construire notre analyseur sémantique. L'enregistrement des ressources se fait dans une base de données en utilisant RDF. Notre méthode propose de regrouper les ressources dans différents clusters selon le type de la ressource.

4.6 Le cadre de découverte des ressources

La figure 4.4 montre l'architecture globale de la méthode proposée. Nous pouvons trouver les trois intervenants d'une organisation virtuelle et les différentes composantes avec lesquelles ils interagissent :

1) Les fournisseurs d'infrastructures ou des ressources.
2) Les utilisateurs finaux qui souhaitent utiliser les ressources offertes par les fournisseurs.
3) Le fournisseur de service qui joue le rôle d'intermédiaire et s'occupe de l'orchestration des tâches avec les ressources où elles vont être exécutées.

4.6.1 Fournisseur d'infrastructure

Les fournisseurs d'infrastructure prennent en charge l'approvisionnement et la gestion de l'infrastructure où les tâches utilisateurs seront effectuées. Ils peuvent mettre à la disposition des utilisateurs tout type de ressources partageables telles que les serveurs, PC, routeurs, logiciels de programmation, bande passante, etc. L'objectif final d'exposer l'infrastructure est de permettre aux utilisateurs d'utiliser ces ressources contre une redevance payée aux fournisseurs. Afin que cette activité devienne lucrative, les fournisseurs d'infrastructure doivent rendre leurs ressources attractives et accessibles. Ceci peut être obtenu en fournissant une description adéquate des ressources. Cette description facilitera le travail du mécanisme de découverte. Dans cette architecture, les fournisseurs d'infrastructure modélisent leurs ressources à l'aide d'une ontologie où on trouve les différents concepts qui représentent le

domaine d'activité ainsi qu'une description des ressources exposées. Les ressources sont enregistrées à l'aide des outils offerts par les fournisseurs des services. Ces derniers prennent en charge le mécanisme de l'enregistrement et la découverte des ressources.

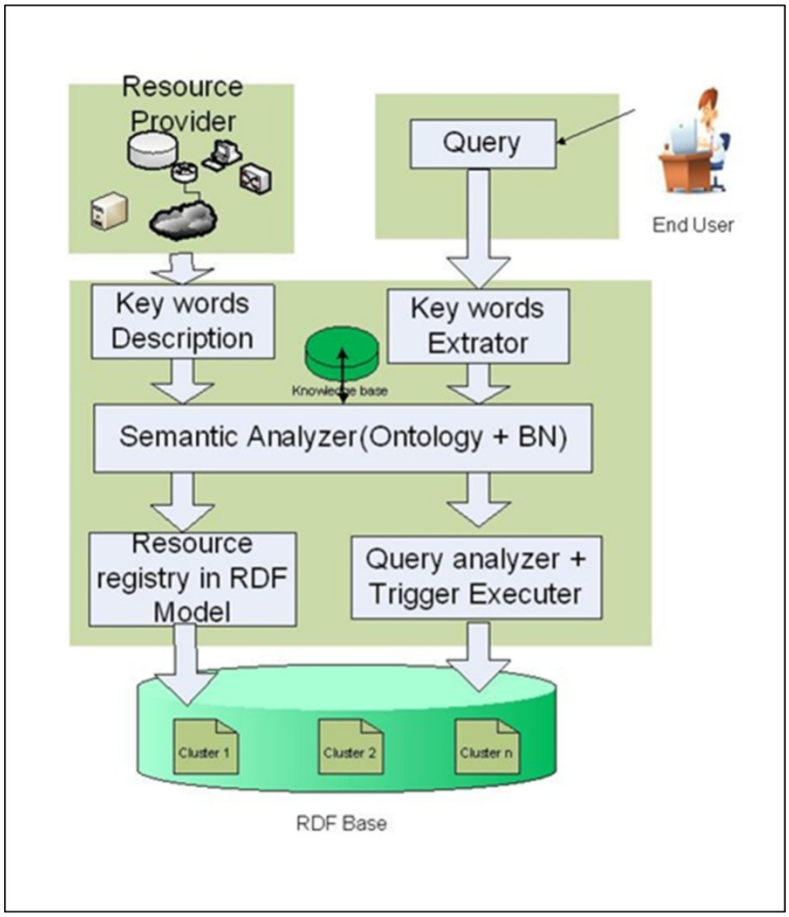

Figure 4.4 L'architecture de la méthode proposée.
Tirée de Daouadji (2010)

4.6.2 Utilisateurs finaux

Les utilisateurs soumettent leurs requêtes via les outils offerts par les fournisseurs de services. Ces requêtes sont analysées pour extraire les mots clés qui représentent la ressource souhaitée. L'analyseur sémantique traite ces informations en se basant sur les connaissances existantes. Si la ressource existe, son adresse est retournée afin qu'elle puisse être utilisée.

4.6.3 Fournisseur de service

Le fournisseur de service est l'intermédiaire entre les utilisateurs et les fournisseurs d'infrastructure. Le fournisseur de service est responsable de mettre en disposition les différents outils logiciels nécessaires au bon fonctionnement de l'organisation virtuelle. Ces outils constituent un inter logiciel dont les fonctionnalités visent à fournir les différents API qui permettent aux fournisseurs d'enregistrer leurs ressources et aux utilisateurs de faire des recherches pour trouver et utiliser les ressources disponibles dans l'organisation virtuelle. L'infrastructure informatique est présentée, à la consommation, sous forme de service. Les fournisseurs d'infrastructures enregistrent les ressources en donnant au système les mots clés qui décrivent ses ressources au mieux, et les utilisateurs soumettent leurs requêtes qui comportent la description des ressources souhaitées.

4.7 La base de connaissance

La recherche des ressources repose sur une base de connaissance. Cette base de connaissance est constituée de deux parties essentielles : l'ontologie qui décrit les ressources et le réseau sémantique Bayésien qui contient l'inférence probabiliste et les liens sémantiques qui caractérisent les ressources. Le module Bayésien est construit à partir des données de l'ontologie fournie par le fournisseur d'infrastructures. Ces données sont entre autres le thésaurus construit à partir des termes de la taxonomie qui décrit les concepts de l'ontologie et de la table de probabilité établie par les experts du domaine. Dans la figure 4.5 nous pouvons voir les principaux composants d'une base de connaissance.

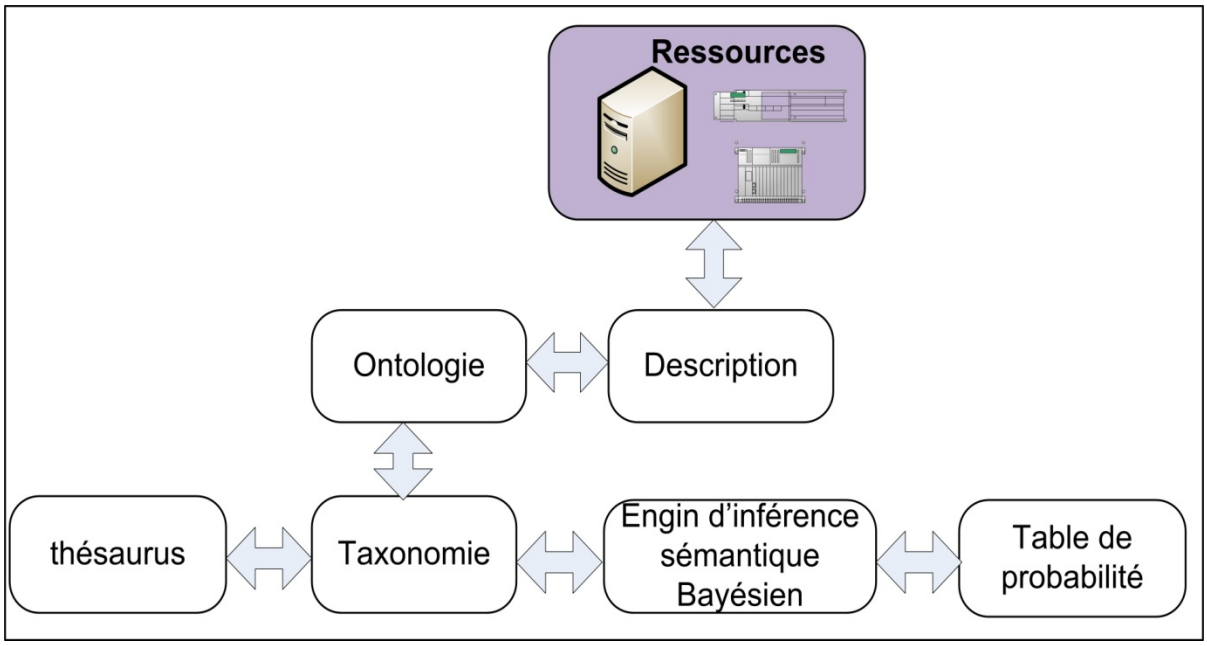

Figure 4.5 Base de connaissance

4.8 L'ontologie

L'ontologie, souvent connue comme le modèle d'un domaine de connaissance, est la description formelle et explicite des concepts d'un domaine bien particulier. La description des concepts, aussi appelés classes, comporte les liens entre les concepts, les terminologies, les métadonnées ou toute autre information pertinente. Les liens entre les concepts peuvent être des liens sémantiques ou des liens hiérarchiques. L'objectif d'une ontologie est la modélisation des connaissances d'un domaine bien particulier. Ce qui permet la construction d'une base de connaisse qui représentera le domaine. Cette base de connaissance sert à donner aux machines (application) la possibilité d'effectuer un raisonnement. Ça permet aussi la réutilisation des connaissances, le partage, la découverte et la communication entre différents utilisateurs et applications sur Internet. Dans notre cas l'ontologie est utilisée pour avoir plus de précision dans la recherche et d'aller au-delà d'une simple recherche basée sur des mots clés pour permettre une recherche sémantique des ressources (Decker, et al., 2000).

4.8.1 Le concept

Le concept est l'objet central d'une ontologie. Un concept est représenté à l'aide d'une classe. Par exemple, une classe *animale* représente tous les animaux. Un animal spécifique est une instance de cette classe. Par exemple, un tigre dans un zoo et un autre qui se trouve dans la jungle sont deux instances différentes du concept de l'animal tigre. Chaque classe possède un nom, et peut avoir d'autres attributs pour la présenter et décrire ses liens avec d'autres classes dans la même ontologie. Par exemple :
Nom : le nom de la classe.
A-Valeur : qui comporte une liste de toutes les instances de la classe.
Sous-Class : pour indiquer que la classe est une sous-classe d'une autre classe plus globale.

4.9 Méthode de construction d'une ontologie

Il y a plusieurs façons de construire une ontologie. Un grand nombre de travaux ont été faits dans ce domaine qui reste un champ de recherche très intéressant notamment la construction automatique des ontologies. Dans notre cas, la construction est manuelle. La plupart de ces méthodes partagent certaines étapes communes. La construction d'une ontologie est souvent un processus itératif. Cette approche commence par une esquisse de classes, qui sera enrichie et raffinée dans chaque itération d'une façon évolutive. Les classes doivent refléter la réalité du domaine visé, elles doivent êtres le plus proches des objets physiques ou bien logiques. On peut énumérer quelques étapes qui peuvent être utilisées comme un guide pour construire une ontologie (Noy & McGuinness, 2001).

4.9.1 Déterminer le domaine de l'ontologie

Des techniques d'analyse utilisées dans d'autres méthodes de modélisation peuvent être empruntées pour la modélisation de l'ontologie. Les méthodes les plus évidentes sont UML et Merise. On commence par déterminer le domaine qu'on souhaite modéliser. Cela peut être

fait on répondant à quelques questions à savoir :

- Qu'est-ce qu'il couvre le domaine? Dans notre exemple le monde animal.
- Quelles informations l'ontologie représente? Les informations (attributs) sont choisies selon le but pour lequel l'ontologie est créée. Pour une application les informations pertinentes ne le seront pas obligatoirement pour une autre.

4.9.2 Déterminer les ontologies existantes qui sont proches

L'une des puissances de l'utilisation des ontologies est la réutilisation des classes déjà développées par d'autres personnes. Après avoir déterminé le domaine de l'ontologie, il est très intéressant d'aller consulter qu'est-ce qu'il a été déjà fait dans des domaines semblables ou très proches. Dans la limite du possible, d'autres ontologies peuvent être étendues, raffinées et adaptées. Plusieurs sites web se sont spécialisés dans le partage des ontologies. Les ontologies sont créées, gérées et enrichies d'une façon collaborative.

4.9.3 Déterminer les classes et la hiérarchie

Les classes qui représentent les différents concepts seront créées à partir des objets du domaine, que ce soit objet logique ou physique. Il y a plusieurs approches pour créer les classes et leurs hiérarchies

L'approche descendante : Dans cette approche, on commence par déterminer les classes les plus générales, et selon le besoin aller aux plus particulières. Par exemple, on peut commencer par déterminer la classe *animale*, en suite, on détermine les sous-classes *herbivore* et *carnivore.*

L'approche ascendante : Dans cette approche, contrairement à la précédente, on commence par déterminer les classes les plus particulières. Ces classes sont regroupées pour former des concepts plus généraux

Combinaison descendante/ascendante : Cette méthode combine les deux approches précédentes. La classe la plus générale est créée, puis une classe très spécifique et en fin d'autres classes sont créées pour regrouper le tout.

4.9.4 Déterminer la terminologie de l'ontologie

Il est très important de déterminer les termes et les vocabulaires les plus génériques qui sont utilisés dans le domaine de l'ontologie qu'on est en train de construire. À chaque concept, on lui associé les termes appropriés.

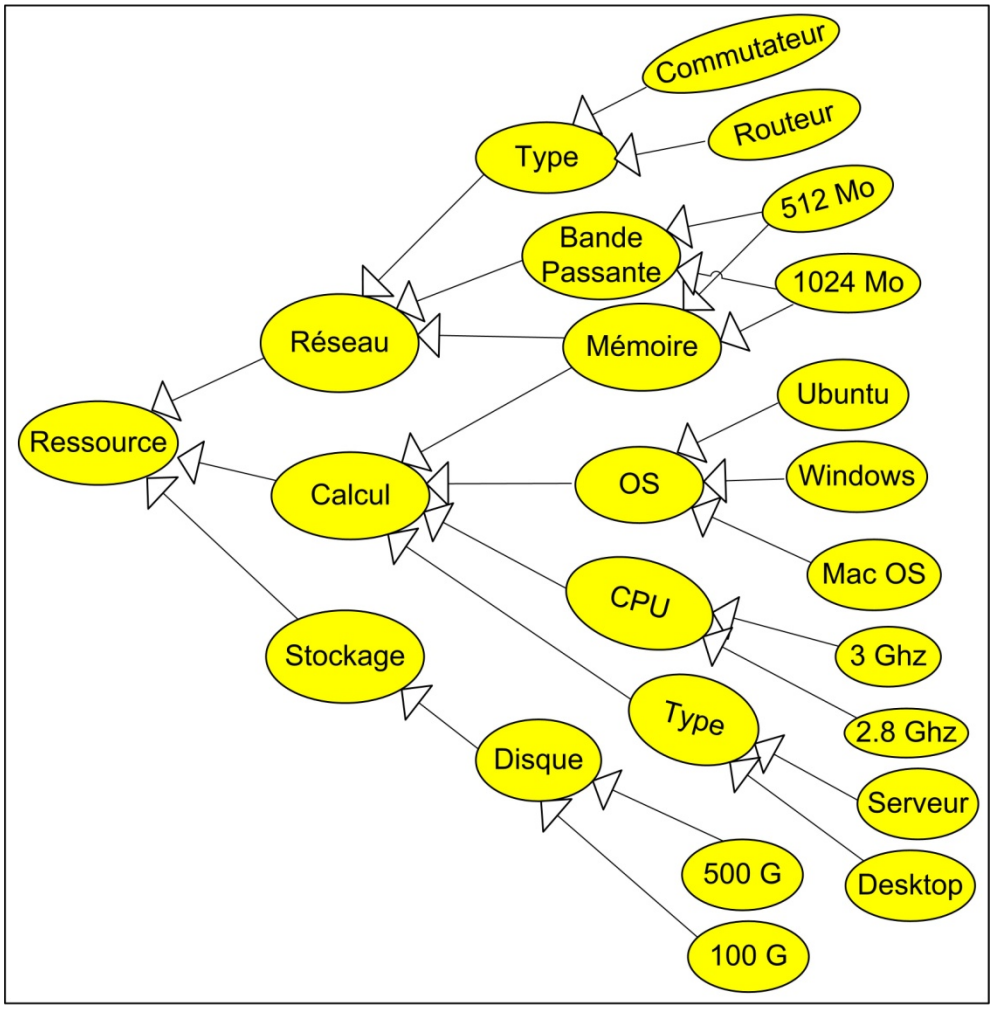

Figure 4.6 Exemple d'une ontologie.

La figure 4.6 montre un exemple d'une ontologie qui représente des ressources appartenant au domaine des nouvelles technologies de l'information (centre de données).

4.10 Le thésaurus

Un thésaurus est un ensemble de termes reliés entre eux par des relations synonymiques. Ils représentent un domaine de connaissance (Wikipedia). Dans notre modèle le thésaurus est utilisé pour éviter que la recherche retourne un résultat inférieur aux attentes. Ceci peut être fixé préalablement par le fournisseur de services. Si la probabilité la plus élevée du concept, est inférieure à un seuil donné, d'autres mots clés issus du thésaurus sont utilisés en plus des mots contenus initialement dans la requête. Une requête étendue donne de meilleurs résultats (Fataicha, Cheriet, Nie, & Suen, 2005). Le thésaurus est utilisé aussi quand le domaine de connaissance couvre un grand nombre de mots clés. Il permet d'éviter d'avoir des tableaux de probabilités de grande dimension. Par exemple, la requête {''Ubuntu'', ''512 Mo'', ''HDD=30 G''} peut être reformulée comme suit {''Ubuntu, Linux'', ''512 Mo, Mega Octet, Mémoire, Memory '', ''HDD=30 G, Disque dur, Harde Disck''}.

4.11 Réseau Bayésien

Les réseaux Bayésiens sont des modèles graphiques qui représentent les relations probabilistes entre un ensemble de variables aléatoires. Les nœuds du graphe représentent les variables, et les liens entre ces nœuds sont des arcs orientés qui représentent la causalité entre les nœuds reliés avec cet arc (Mittal, 2007).

Voici quelques notions de base concernant les réseaux Bayésiens (F. Taroni, 2006) :

- Un ensemble fini de nœuds interconnectés via un ensemble d'arcs forment une structure mathématique appelée un graphe orienté.
- Un nœud X est un père du nœud Y s'il y a un arc de X vers Y. Y est appelé aussi enfant de X.
- Un nœud sans parent est appelé racine.

- Un graphique dirigé connecté avec aucun cycle est appelé un graphique acyclique dirigé (DAG).
- Les nœuds dans le graphe représentent des variables aléatoires. La variable aléatoire peut-être soit discrète, avec un ensemble fini d'états mutuellement exclusifs, ou bien continue.
- Les arcs représentent les relations directes et pertinentes entre les variables (représentées par des nœuds), de façon que pour chaque variable X avec les parents Y_1, Y_2,...,Y_n est associé une probabilité conditionnelle $P(X|Y_1,Y_2,...,Y_n, I)$, où I désigne les connaissances *préalables*, qui sont les connaissances pertinentes ne figurant pas explicitement sous la forme d'un nœud dans le graphe. Si X est un nœud sans parent (racine), les probabilités se réduisent à P (X | I). Il n'y a aucune influence d'autres nœuds dans le graphe.
- Soit X une variable avec n états (X_1, X_2,..., X_n). Si X est un nœud racine, la table de probabilité conditionnelle P(X|I) sera un tableau de n dimensions, contenant la distribution de probabilité P (($X = X_i$), i = 1, . . ., N), avec $n \sum P (X = X_i) = 1$.
- Soit X un nœud qui possède un ou plusieurs parents et Y une variable avec m états. Si Y est un parent de X alors la probabilité conditionnelle P (X | Y) sera un tableau de dimension $m \times n$, contenant toutes les probabilités P(X = x | Y = y). Par exemple, supposons que les variables B et D (figure 4.7) ont chacun deux états possibles et la variable C à trois états. La table de probabilité conditionnelle de C contient 12 entrées $P(C = c_i | B = b_j, D = d_k) = P_{ijk}$, avec (i = 1, 2, 3, j = 1, 2; k = 1, 2) tableau 4.1.

Tableau 4.1 probabilité conditionnelle

$P(C = c_1 \mid B = b_j , D = d_k)$	p_{111}	p_{112}	p_{121}	p_{122}
$P(C = c_2 \mid B = b_j , D = d_k)$	p_{211}	p_{212}	p_{221}	p_{222}
$P(C = c3 \mid B = b_j , D = d_k)$	p_{311}	p_{312}	p_{321}	p_{322}

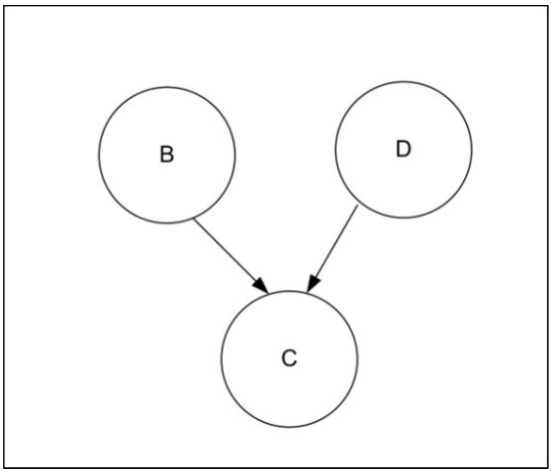

Figure 4.7 Graphe Bayésien.

4.11.1 Réseau Bayésien pour la recherche d'informations

Le réseau Bayésien pour la recherche d'informations est un graphe acyclique dirigé. Les nœuds représentent des variables aléatoires et les arcs la dépendance entre ces variables. Le père pour la cause et l'enfant pour le résultat (Kim, Hong, & Cho, 2005). Chaque arc possède une probabilité ce qui produit une distribution de probabilité jointe. Les nœuds "parents" possèdent une probabilité à priori *P(p)* et les enfants une probabilité conditionnelle *P(c|p)*. Utilisant les conditions d'indépendance, la distribution de la probabilité jointe de $P(x_1,...,x_n)$ peut être donnée par la formule suivante :

$$P(x_1 ; x_2 ; \ldots ; x_n) = \prod P\,(X_i \mid Parents\,(X_i)) \qquad (4.1)$$

Les réseaux Bayésiens pour la recherche d'informations ont été introduits par *Turtle and Croft*. Ils ont montré que le système de recherche d'informations basé sur les réseaux Bayésiens donne de meilleurs résultats comparativement aux autres modèles probabilistes traditionnels. La puissance des réseaux Bayésiens réside dans le fait qu'ils prennent en considération les connaissances à priori (expert) et celles contenues dans les données. Les réseaux Bayésiens ont été aussi appliqués à d'autres problèmes comme la construction automatique d'hypertexte, les filtres d'information, et la classification de documents.

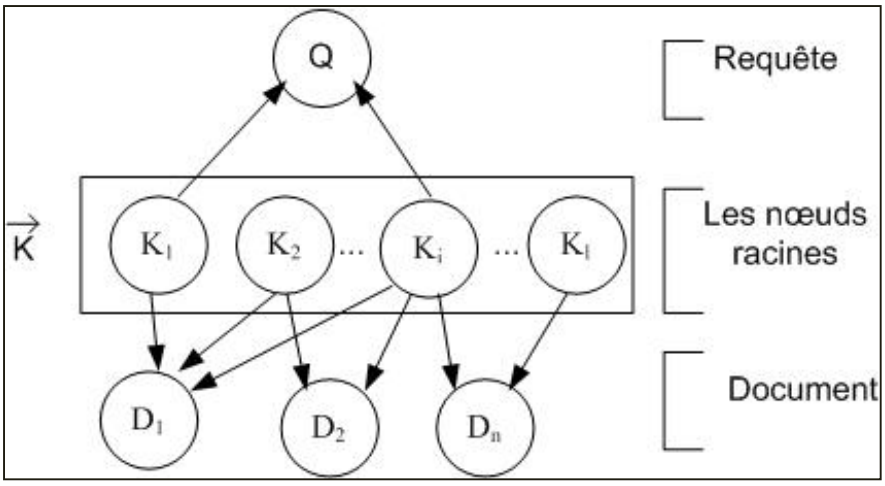

Figure 4.8 Réseau Bayésien pour la recherche d'informations.

Le modèle Bayésien utilisé dans les systèmes d'extraction et de recherche d'informations dans les documents considère les requêtes, les documents et les mots-clés comme des événements.

Dans la figure 4.8, le nœud D_j représente la variable Document et le nœud Q représente une requête utilisateur, et K_i représente un mot possible appartenant à un domaine défini. Le vecteur $\vec{K}$ représente tous les états possibles du nœud racine K_i.

La relation entre le document D_j et une requête Q est interprétée comme la probabilité que le document D_j de s'identifier dans la requête Q. En utilisant la loi de Bayés et les règles de probabilité, on peut calculer la probabilité conditionnelle $P(D_j|Q)$:

$$P(Dj|Q)=\eta \sum_k P(D_j|\overrightarrow{K})P(Q|\vec{K})P(\overrightarrow{K}) \qquad (4.2)$$

4.11.2 L'analyseur sémantique Bayésien

L'analyseur sémantique Bayésien est un analyseur basé sur la loi de Bayés vue plutôt. Dans la figure 4.9, on peut voir le réseau Bayésien utilisé. Inspiré de (Kim, Hong, & Cho, 2005) ce

réseau combine l'inférence probabiliste et l'inférence sémantique. Ce modèle permet de prendre en charge le problème des ressources sémantiquement identiques et syntaxiquement différentes. Il prend aussi en charge les ambigüités qui peuvent s'introduire dans les requêtes. Dans la figure 4.9 on peut voir que l'analyseur est formé de trois couches : la couche requêtes, la couche mots clés et la couche concepts.

La couche requête : Cette couche accepte les demandes d'enregistrement et recherche de ressources. Ces deux opérations sont traitées de la même façon. Dans cette couche les requêtes sont analysées pour extraire les mots clés qui décrivent les ressources afin de pouvoir faire la correspondance entre ces mots clés et les mots qui décrivent les ressources dans la base de données. Traditionnellement, les moteurs de recherche comparent les mots clés avec ceux de la base de données en appliquant une correspondance mot à mot. Le problème qui se pose est que dans un environnement virtuel qui comporte des ressources hétérogènes, la description peut être faite avec des mots syntaxiquement différents, mais sémantiquement identiques. D'où l'utilisation du Bayésien pour faire face à l'ambigüité et établir ainsi les liens sémantiques.

La couche mots clés : Les requêtes sont analysées et les mots clés sont extraits. Dans notre cas, les mots clés sont représentés en utilisant les triplets RDF, $M = O \ U \ A \ U \ V$ (O/Objet, A/Attribut, V/Valeur), ou les objets représentent les caractéristiques de la ressource souhaitée avec leurs valeurs. L'analyseur sémantique est appliqué sur les valeurs alphabétiques étant donné que les valeurs numériques peuvent être exprimées en utilisant le langage de la requête SQL une fois l'objet est défini.

Dans cette couche, on peut trouver aussi des mots synonymes. Ces synonymes servent à étendre les requêtes en cas ou la recherche donne de faibles résultats comme on l'a déjà expliqué dans la section du thesaurus. Les relations entre les mots clés et les synonymes sont des relations sémantiques alors que la relation entre la couche mots clés et les concepts est

une relation probabiliste. La relation sémantique probabiliste dans les réseaux Bayésiens est similaire à celle du modèle traditionnel de recherche d'information.

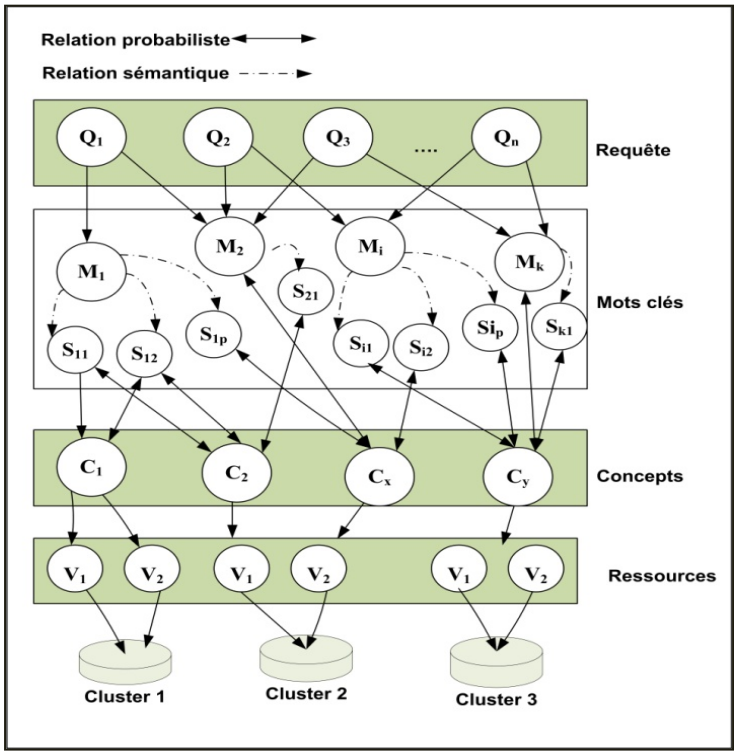

Figure 4.9 Réseau sémantique Bayésien pour la recherche de ressources.

La couche concept : Dans la couche des concepts, on trouve les types de ressources qu'on peut trouver dans les bases de données. Ces concepts seront représentés par des trigger sous forme de requête SPARQL. La requête utilisateur est mappée dans un vecteur $\vec{R}$ = {X_1, X_2, …, X_i, …., X_n} où X_i représente soit un mot clé *M* ou son synonyme *S* qui est obtenu à partir du thésaurus. Les synonymes sont ajoutés à la requête si les résultats de recherche sont faibles. Un seuil de nombre de résultats peut être défini selon la politique de l'organisation virtuelle.

Pour chaque requête utilisateur $\vec{R}$ on calcul sa probabilité de correspondre à un concept C_i

$$
\begin{aligned}
P(C|\vec{R}) &= P\,(C|\, X_1, X_2, \ldots, X_i, \ldots., X_n) \\
&= \frac{P(C)*P(X_1,X_2,\ldots,X_i,\ldots.,X_n|C)}{P(X_1,X_2,\ldots,X_i,\ldots.,X_n)} \\
&\sim P(X_1, X_2, \ldots, X_i, \ldots., X_n|C) \\
&= P(C)*P(X_1|C)*\ldots..*P(X_n|C) \\
&= P(C)\prod_{i=1}^{n} P(X_i|C) \qquad (4.3)
\end{aligned}
$$

$P(C|\vec{R})$ représente la probabilité à postériori de chaque concept, la probabilité P(C) représente l'évidence et finalement $P(X_1, X_2, \ldots, X_i, \ldots., X_n|C)$ la vraisemblance. Après le calcul de la probabilité de tous les nœuds, on sélectionne le nœud concept avec la probabilité la plus élevée. Le concept choisi détermine la ressource qui sera retournée en exécutant une requête SPRQL avec les différents paramètres souhaités. Les expressions régulières sont utilisées pour extraire les variables numériques tels que la taille de la mémoire la vitesse des processeurs, etc. Et pour les variables alphanumériques on utilise la même formule présentée dans (4.3).

L'assignation, par l'expert du domaine, des probabilités dans le réseau Bayésien est effectuée à la main. Pour l'assignation des probabilités nous utilisons un guide inspiré de (Kim, Hong, & Cho, 2005). Le tableau suivant récapitule le principe :

Tableau 4.2 Attribution des probabilités conditionnelles

Condition	**Probabilité $P(X=1|C_i=1)$**
X est citée chaque fois que C_i est cité	0.95-0.99
X est souvent citée chaque fois que C_i est cité	0.60-0.80
X est rarement citée chaque fois que C_i est cité	0.20-0.40
X n'est jamais citée chaque fois que C_i est cité	0.01-0.10

Le tableau 4.2 constitue comme un manuel pour assigner les valeurs pour la probabilité jointe $P(X=1|C_i=1)$. L'assignation de cette probabilité est basée sur le jugement de l'expert du

domaine, qui doit décider de la pertinence de chaque mot clé X par rapport à un concept C_i. Si l'utilisation de ce mot clé est jugée obligatoire pour la description du concept C_i alors la probabilité $P(X=1|C_i=1)$ est comprise entre 0.95 et 0.99.

L'exemple suivant illustre le principe : soit le concept C_1 = {le métier d'avocat} et le concept C_2= {salade à l'avocat} et les mots clés *M*= {emballage, cabinet, prix, salaire}.
Les probabilités de vraisemblances peuvent être données comme suit :*P(avocat|C1) =0.99 , P(avocat|C2) =0.99 et P(emballage|*C_1*)* = 0.30 alors que P(*emballage*|C_2) = 0.95 de même que *P(salaire|*C_1*)*= 0.99 *alors que P(salaire|*C_2*)=0.01*

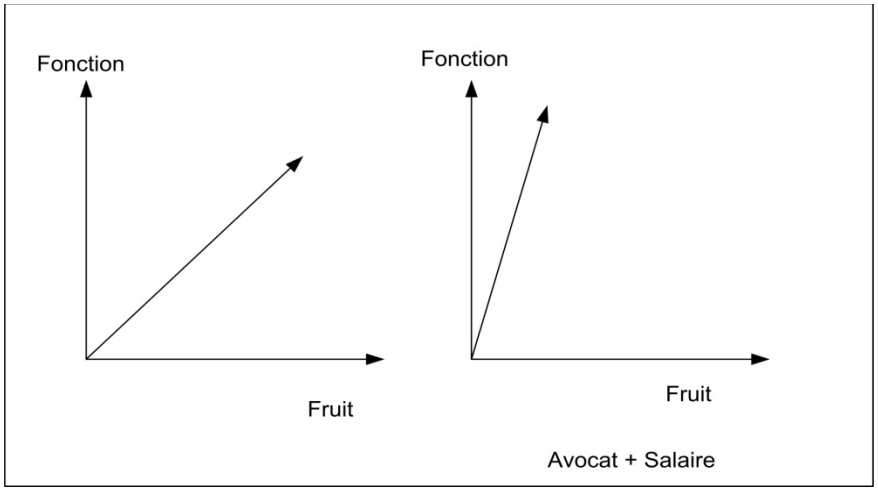

Figure 4.10 Raisonnement sémantique Bayésien.

Pour une requête R qui comporte les mots clés, *salaire* et *avocat* nous calculons la probabilité *P(C|R)* en utilisant la formule (4.3) : *P(R|*C_1*) = P(salaire|*C_1*)* P(avocat|*C_1*)* alors *P(R|*C_1*)* =0.99*0.99 ce qui donne *P(R|*C_1*)* = 0.98 qui est plus grand que P(*R*|C_2) calculer de la même façon = 0.0099. Le concept avec la plus grande probabilité est choisi dans notre cas c'est C_1. Le fait de considérer les mots clés dans leur contexte global influe sur la prise de décision. Chose absente si on considère chaque mot comme entité indépendante figure 4.10.

```
Début
//Extraire mots clés de la requête Q
M ← Q
//Pour chaque mot m de M Calculer la probabilité avec chaque concept c de C
Pour chaque m ∈ M
{
Probabilité (c) = Probabilité (c) X Calculer_Probabilité (m|c)
}
//Retourner le concept avec la probabilité la plus élevée.
        Retourner_Max (Probabilité (c))
        Si (Retourner_Max (Probabilité (c)) < Seuil) alors
        {
                // Etendre la requête
                Si Thesaurus ≠ Nul
                {
                Q ← Q+ S
                Thesaurus← Nul
                Retourner au début
                }
                Sinon Retourner_Max(Probabilité (c) )
         }
Sinon
{
// Déterminer les variables de la ressource souhaitée
// Pour les variable alphabétiques utiliser le réseau Bayésien comme au début
// Pour les variable numériques utiliser les expressions régulières
}
Déclencher la requête SPARQL.
```

Figure 4.11 Pseudo code.

La figure 4.11 montre le processus de recherche d'une ressource. La recherche qui commence par extraire les mots clés de la requête. Ensuite on calcule pour chaque mot clé de la requête sa probabilité avec tous les concepts de la base de connaissance en utilisant la loi de Bayés. Le concept avec la probabilité la plus élevée est alors retourné. Si cette probabilité est inférieure à un seuil donné, la requête est étendue en utilisant le thésaurus. Une fois le concept trouvé, de la même façon on détermine les variables alphabétiques. Quant aux variables numériques, elles sont extraites à l'aide d'expressions régulières. Par exemple, pour extraire la valeur de vitesse du processeur on peut utiliser cette expression régulière

suivante : *(\d)+Ghz | cpu (\d)+ | (\d)+ghz* qui veut dire extraire tous les nombre, exprimer a l'aide de *(\d)*, qui sont suivit des lettres *Ghz* ou *ghz*. Ou bien précédées par le mot *cpu*.

4.12 Conclusion

Dans ce chapitre, nous avons présenté les spécifications de la découverte de ressources proposées par les concepteurs du Framework IaaS d'Inocybe. Nous avons proposé un modèle pour la découverte de ressources dans un environnement de nuage informatique qui prend en charge la recherche sémantique des ressources. Ce modèle utilise une base de connaissance afin de permettre une recherche sémantique et de passer outre les limitations des recherches basées sur les correspondances entre les mots clés. Cette base de connaissance est construite à l'aide d'une ontologie. L'ontologie est utilisée pour décrire les ressources d'un domaine particulier (dans notre cas le domaine des TIs). À partir de cette ontologie, nous dégageons les différents liens sémantiques qu'une ressource peut avoir avec ses propres caractéristiques ou avec d'autres types de ressources. Ce qui permet de créer un mécanisme de raisonnement sémantique. Alors, la taxonomie des ressources qui appartiennent au domaine est énumérée. Les termes les plus communs, qui décrivent au mieux les ressources, sont regroupés avec leurs synonymes dans des thésaurus. Une table de probabilité est établie et sera utilisée par le réseau sémantique Bayésien qui relie les concepts avec les termes du thésaurus. Le rôle de ce réseau est de faire face à l'incertitude dans les requêtes due au choix des termes des requêtes. L'inférence sémantique est ainsi faite à l'aide d'un réseau sémantique Bayésien.

CHAPITRE 5

EXPÉRIMENTATIONS ET DISCUSSION

5.1 Introduction

Les expériences, pour valider notre modèle, ont été effectuées dans le cadre du projet Green Star Network (GSN) où la découverte des ressources s'effectue non seulement sur les caractéristiques de la ressource, nécessaires pour l'exécution des tâches utilisateurs, mais aussi sur la nature de la source d'énergie utilisée par la ressource. Dans la section suivant, nous allons présenter le projet GSN afin de bien situer le cadre des tests.

5.2 Le Green Star Network

L'objectif du projet GSN est de faire converger les efforts des industriels, des universités et des organismes gouvernementaux, à l'échelle Canadienne, dans le but commun de réduire les gaz à effet de serre (GES) résultant des technologies de l'information et de la communication. Le résultat attendu est la création d'outils, de protocoles de mesure de carbone, des procédures, des cas d'utilisation pour construire un réseau évolutif de fournisseurs de services TIC, afin d'offrir à leurs clients des services plus écologiques à des prix bas.

Actuellement, il existe deux approches qui sont adoptées pour augmenter l'efficacité énergétique des compagnies qui œuvrent dans le domaine des nouvelles technologies et de réduire les émissions des gazes à effet de serre. La première approche consiste à réduire la consommation unitairement ‘‘ *micro level* ’’. Dans cette approche l'effort est concentré sur le développement des appareils qui consomment moins d'énergie. Les recherches qui suivent cette voie se focalisent sur le design des micro-processeurs et des ordinateurs en développant des architectures *énergie à la demande*, les machines virtuelles et ainsi de suite. Cependant, l'efficacité énergétique au niveau unitaire ce traduira par une augmentation de la consommation au niveau globale dû au postulat de Khazzoom–Brookes qui affirme que ‘‘ *les*

améliorations de l'efficacité énergétique qui, au sens le plus large, est justifiée au niveau microéconomique conduisent à de plus hauts niveaux de consommation d'énergie au niveau macroéconomique.'' (Saunders, 1992). Le projet Green Star Network est motivé par deux hypothèses. La première part de l'énoncé suivant : '' réduire les émissions des gaz à effet de serre au niveau global est la solution la plus appropriée''. D'ailleurs, c'est cette solution qu'adoptent de grandes compagnies comme Google et Microsoft qui commencent à construire leurs centres de données proches des sources d'énergie renouvelables. Malheureusement, la plupart des compagnies qui œuvrent dans ce domaine ne sont pas en mesure de construire leurs centres de données proches de sources d'énergie renouvelable et propre. En plus l'énergie renouvelable, notamment l'énergie solaire et éolienne, n'est pas disponible tout le temps. La deuxième hypothèse est une conséquence de la première. Étant donné qu'il faut réduire les émissions des gaz à effet de serre à l'échelle globale plutôt qu'à l'échelle unitaire il faut construire les centres de données proches de source d'énergie renouvelable et propre. Cette solution n'est pas toujours réalisable. Il est impossible de construire les centres dans des régions où il y a tout le temps de l'énergie renouvelable. La solution est soit de transporter l'énergie renouvelable, lorsque elle est disponible, vers les centres de données. La deuxième solution est de relocaliser les données du centre de données dont l'énergie renouvelable est momentanément non disponible vers les centres de données dont l'énergie renouvelable est disponible et suffisante. L'hypothèse se base que le fait de déplacer les données vers les sources d'énergie renouvelable est plus efficace que déplacer l'énergie renouvelable vers les centres de données. Le principe du projet GSN est ''suivre le vent, suivre le soleil'' c.-à-d. faire déplacer les services et les applications là où il y a l'énergie renouvelable afin de n'utiliser que celle-ci. Le projet GSN est construit en utilisant les outils qu'offre le Framework IaaS présenté précédemment. La mobilité des données est réalisée à l'aide des technologies de nuages informatiques. Les ressources utilisateurs sont créées sous forme de machines virtuelles qui roulent dans des systèmes d'émulation appelés *hyperviseur* (exemple KVM, XEN). Ces hyperviseurs peuvent être contrôlés en utilisant les interfaces de programmation (API) pour la gestion des systèmes de virtualisation dans notre cas Libvert.

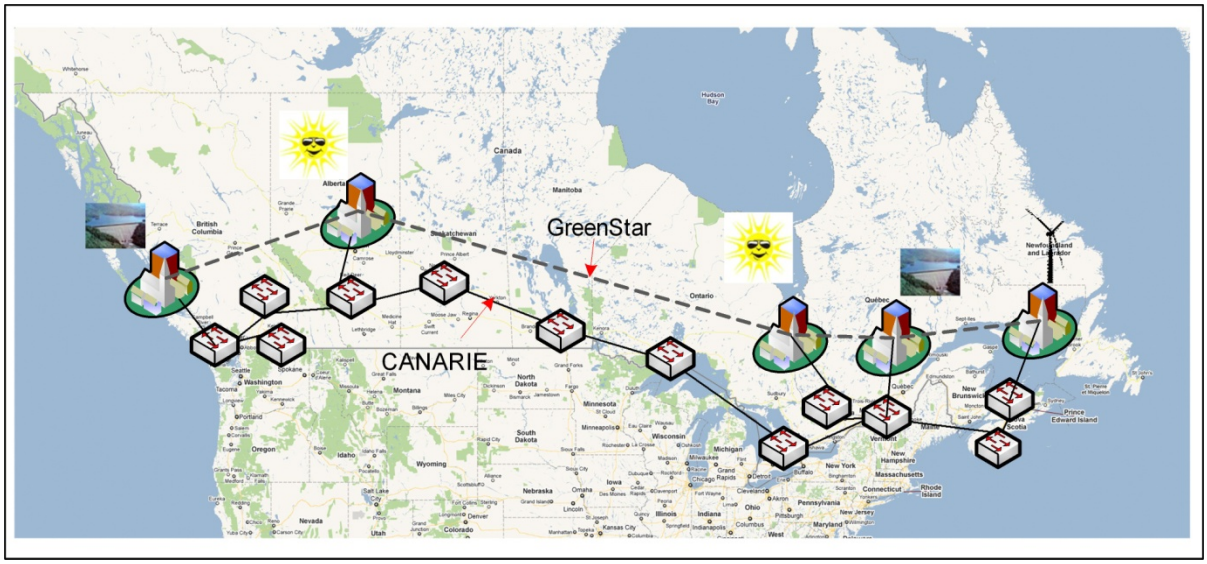

Figure 5.1 Architecture du réseau GSN.

La figure 5.1 représente l'architecture globale du réseau GSN. Le réseau virtuel de GSN est construit sur le réseau physique de CANARIE (Canada's Advanced Research and Innovation Network) le réseau évolué de la recherche et de l'innovation du Canada. Sur l'image on peut voir les différents nœuds que comporte le réseau GSN. Le réseau va poursuivre son extension au niveau international pour construire une organisation virtuelle capable d'offrir des services non polluants, avec des prix raisonnables. Actuellement, le réseau GSN comporte quatre nœuds. Deux nœuds utilisent l'énergie solaire : un à Ottawa au niveau du Centre de Recherche en Communication (CRC) et l'autre à Calgary (Cybera). Un autre nœud alimenté avec l'énergie éolienne et géré par Bastion Host qui se trouve à Truro en Nouvelle-Écosse. Et finalement un nœud à l'ETS (Montréal) alimenté par l'énergie hydraulique. En plus d'offrir des ressources aux utilisateurs, le nœud de l'ETS héberge le gestionnaire de l'organisation virtuelle sous forme de contrôleur de nuage informatique. Chaque utilisateur, autorisé à utiliser les services de GSN, peut se connecter et créer des ressources sous forme des machines virtuelles avec les logiciels souhaités pour exécuter ses tâches. Le gestionnaire de l'organisation virtuelle prend en charge cette requête. Il crée la ressource demandée et veille qu'elle soit exécutée en utilisant au mieux l'énergie renouvelable.

5.3 Déploiement de GSN

Dans la figure 5.2 nous pouvons voir un déploiement typique d'un environnement GSN. Avec trois nœuds : Un qui héberge le gestionnaire de GSN (fournisseur de service) et deux nœuds qui représente chacun un fournisseur d'infrastructure.

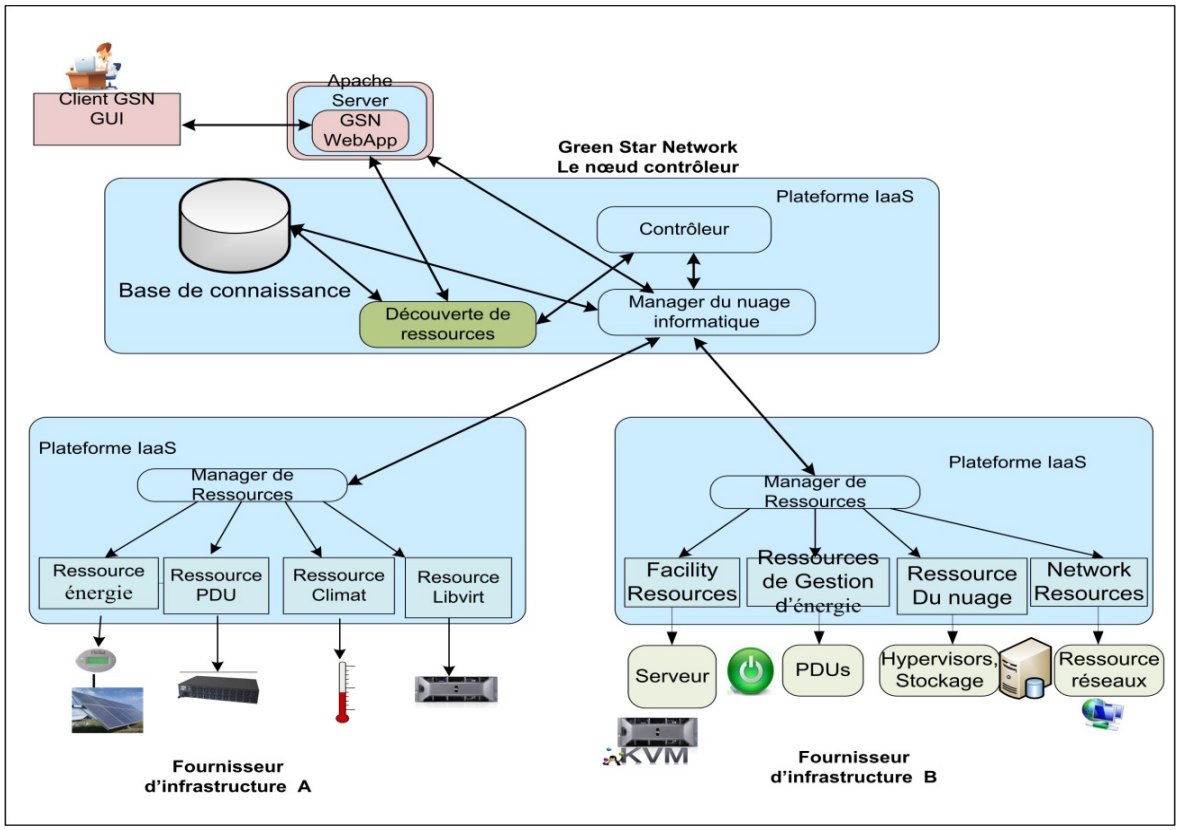

Figure 5.2 Déploiement d'une VO GSN.

Dans le premier nœud le module qui gère l'organisation virtuelle, en l'occurrence le contrôleur, est responsable de donner au GSN l'autonomie et l'intelligence nécessaire pour gérer l'OV d'une manière efficace. Par exemple, il doit prendre les décisions de la migration d'une machine virtuelle d'un serveur à un autre selon certains critères notamment l'efficacité énergétique. Ce nœud peut aussi offrir de l'infrastructure. Afin qu'il soit menu d'une certaine autonomie et capacité de prise de décision, le contrôleur fait appel à notre système de

découverte de ressources, pour localiser les ressources recherchées et prioriser l'utilisation des nœuds alimentés par l'énergie renouvelable. Dans les nœuds qui représentent les fournisseurs d'infrastructures on y trouve les différentes ressources virtualisés accessibles aux utilisateurs pour qu'elles soient partagées. On y trouve par exemple un serveur, un disque dur, des ressources réseau, des PDU pour la mesure d'électricité, etc.

5.4 Ontologie

L'ontologie proposée comprend les modèles de connaissance qui représentent les éléments qu'on trouve dans un centre de données et de leurs sources d'énergie. Ces modèles sont mappés en différentes classes. Chaque classe est définie par un ensemble de propretés et d'attributs. La figure 5.3 illustre l'ontologie proposée. Cette ontologie combine deux grands domaines, celui des TIC et de l'énergie. Le modèle proposé inclut les classes suivantes :

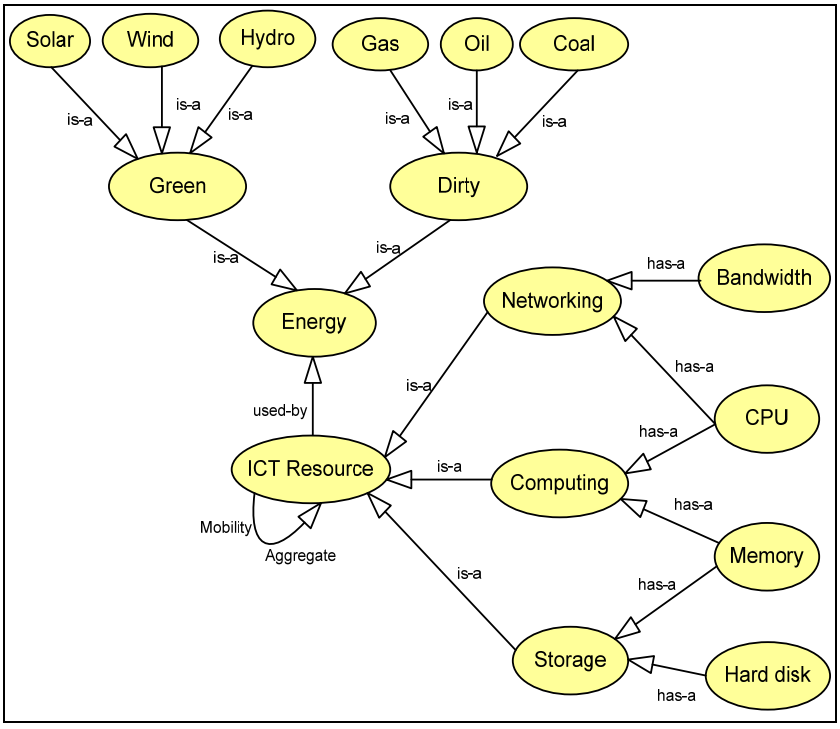

Figure 5.3 Ontologie.
Tirée de Daouadji (2010)

- La ressource TIC qui est une classe générique de tout élément d'un centre de donnée. Elle comporte les caractéristiques communes tel que le nom de la classe, l'identificateur, l'emplacement ainsi que le type d'énergie utilisée pour le fonctionnement de la ressource. La classe ressource possède trois sous-classes :

La première classe est la classe calcul ‘‘Computing’’ qui représente toutes les ressources dédie aux tâches de calculs tels que les serveurs, les calculateurs, etc.

- La deuxième classe est classe réseaux ‘‘ Networking’’ qui représente toutes les ressources utilisées pour l'interconnexion des différentes ressources TIC, telles que les routeurs, les Switch, les Hubs, les connecteurs optiques, multiplexeur-démultiplexeur.

- La troisième classe représente toutes les ressources de stockages, tels que les disques durs, CD, les périphériques multisupport.

Pour chaque ressource TIC, on associé une classe énergie. Cette classe détermine le type et la quantité d'énergie consommée par chaque ressource. La classe énergie est devisée à son tour en deux classes : la classe d'énergies propres ou renouvelables tel que l'énergie solaire, éolienne et hydraulique. Et la deuxième classe représente les énergies polluantes telles que l'énergie fossile comme le Gaz, le Charbon et le Pétrole. Pour chaque type d'énergie est associé un coût.

5.5 Construction du banc de test

Dans le projet GSN, il y a un ensemble de ressources virtualisées. On peut trouver les ressources suivantes : des machines de calcul (Serveur), des supports de stockage, des ressources réseau, des appareils de mesure d'électricité (PDU).
Pour mesurer les performances de notre modèle, nous avons adopté le plan de tests suivant :

Nous avons constitué un ensemble de ressources à partir de données collectées aléatoirement sur Internet (Wikipedia.org et Dell.ca). Le tableau 5.1 comporte les différentes valeurs utilisées.

Tableau 5.1 Les différentes caractéristiques des ressources

Caractéristique	**Valeur**
Mémoire en Mo	64, 128, 51, 1024, 2048, 3072,4096
Système d'exploitation	Ubuntu, Linux, Fedora, Windows XP, Windows Vista, RedHat, Centos.
Disque dur Giga	15, 20, 30, 45, 50, 60, 100, 120, 500
CPU en GHz	2.13, 2.4, 2.8,2.93, 3, 3.2, 3.33, 3.46, 3.6, 3.8
Bande passante Mo	128, 512, 1024,

Les tests ont été effectués sur trois types de ressources : machine de calcul, ressource réseau et support de stockage. Le tableau 5.2 présente les trois types de ressources sur lesquelles ont été effectués les tests ainsi que leurs attributs.

Tableau 5.2 Les ressources utilisées

Ressource	**Exemple**	**Attributs**
Machine de calcul	Serveur , Desktop	OS, Mémoire, CPU, HDD
Réseau	Commutateur , routeurs	CPU, mémoire, bandwidth
Stockage	Disque dur	HDD

Nous utilisons un script pour générer aléatoirement les ressources qui sont équitablement distribuées entre les trois types de ressources. Par contre, dans chaque type de ressource, la distribution des variables des attributs n'est pas uniforme. Par exemple, nous donnons plus de poids au système d'exploitation Ubuntu dans la ressource de calcul. Les attributs qui

caractérisent chaque ressource sont représentés dans l'ontologie de la figure 5.3. En l'absence d'un protocole qui quantifié les émissions de GES dans le domaine des TIC, pour nos tests, nous utilisons l'émission de CO_2 comme coût d'énergie. L'attribut qui représente le type d'énergie est remplacé par un attribut qui représente une province au Canada dans ces tests. Le tableau 5.3 indique les différentes mesures de CO_2 générées par province et par kWh d'électricité consommé dans les ménages (LivClean, 2010). Comme nous pouvons le constater c'est dans la province de l'Alberta où le kWh génère le plus de GES. C'est dû à l'utilisation de l'énergie fossile pour la production de l'électricité. La province où l'énergie électrique est la moins polluante est le Québec où on utilise l'hydro-électricité. Cela veut dire qu'utiliser une ressource au Québec générera moins de GES qu'ailleurs au Canada.

Tableau 5.3 CO_2 t/kWh par province

Province	**CO_2 t/kWh**
AB	0.000930
BC	0.000020
NS	0.000549
ON	0.000180
QC	0.000006

Le but est d'optimiser l'utilisation de chaque Kilo Watt au Québec, afin d'éviter l'utilisation des ressources dans les autres provinces qui sont plus polluantes que celles du Québec. En cas où il n'y a plus de ressources disponibles au Québec les ressources de la province qui génère le moins de GES sont utilisées et ainsi de suite. Le tableau 5.4 représente les valeurs des de probabilités qui seront utilisées dans le réseau Bayésien.

Tableau 5.4 Probabilité conditionnelle.

Concepts	**Mots clés**								
	RAM	Mémoire	CPU	Processeur	Ubuntu	Disque	HDD	Band passante	CISCO
Calcul	0.97	0.97	0.99	0.99	0.95	0.60	0.60	0.47	0.10
Stockage	0.10	0.10	0.10	0.10	0.10	0.99	0.99	0.20	0.05
Réseau	0.60	0.60	0.60	0.60	0.10	0.10	0.10	0.99	0.99

5.6 Test de performance

La première expérimentation a pour but de comparer les performances de la méthode que nous avons proposée qui est basée sur les réseaux Bayésiens et celle qui utilise le moteur de recherche Compass/Lucene basée sur une recherche textuelle des mots clés. Pour une requête qui porte sur une ressource de calcul avec les caractéristiques suivantes : système d'exploitation Ubuntu, mémoire de 1024 Mo, un disque dur d'une taille de 50 Go et un processeur de 2.3 GHz de puissance. Les résultats obtenus peuvent êtres visualisées dans la figure 5.4.

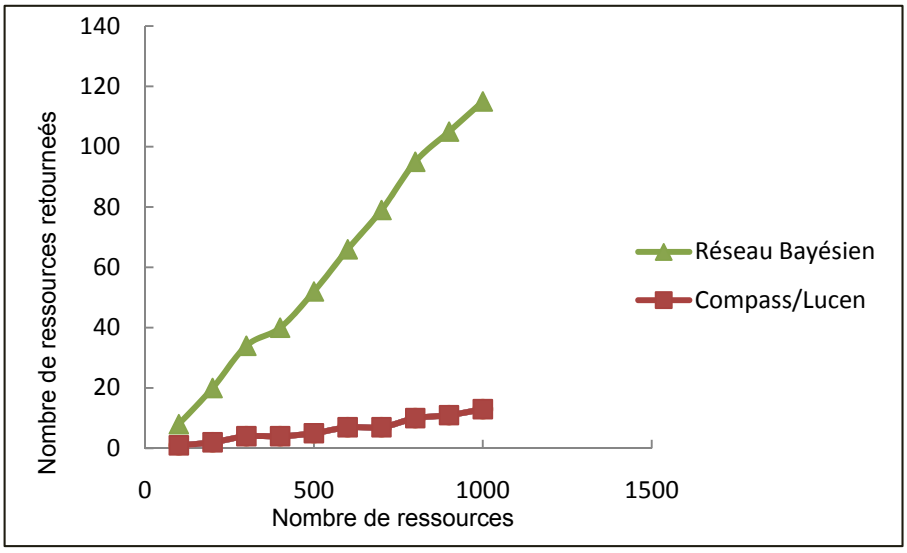

Figure 5.4 Nombre de ressources retournées avec les deux méthodes.

Il est clair que les performances de notre méthode sont nettement supérieures. Ces performances s'expliquent par le fait que dans notre méthode toutes les ressources compatibles sont retournées alors que dans la méthode basée sur une recherche textuelle seules les ressources qui comportent la description exacte sont retournées. Par exemple, une ressource avec une mémoire 2048 Mo sera ignorée alors que cette ressource pourra bien exécuter les tâches demandées à une ressource moins puissante. Notre méthode basée sur l'ontologie et les réseaux Bayésiens possède les connaissances nécessaires sur les ressources qui lui permettent de savoir quel est le type de la ressource ainsi que ses attributs il s'agit et de lancer des requêtes SPARQL (figure 5.5) qui permettent d'aller chercher toutes les ressources qui peuvent être compatibles avec la ressource recherchée.

```
queryString = SELECT ?resource ?memory  ?OS ?cpu ?hdd
 WHERE {
   ?resource <   hasCPU > ?cpu . FILTER (?cpu >= QueryCPU).
    ?resource < hasMemory > ?memory. FILTER (?memory >= QueryMemory).
   ?resource < hasOS > ?OS . FILTER (?OS = QueryOS).
    ?resource < hasHDD > ?hdd. FILTER (?hdd >= QueryHDD). }
```

Figure 5.5 Requête SPARQL.

5.7 Découverte des ressources selon les émissions du CO_2

Dans le cadre du projet GSN, nous avons testé notre méthode dans le but d'utiliser que les ressources qui génèrent le moins de GES. Pour une requête spécifique, la figure 5.6 représente une distribution des ressources selon leur localisation qui correspondent à cette requête. Nous constatons que la distribution suit la distribution normale.

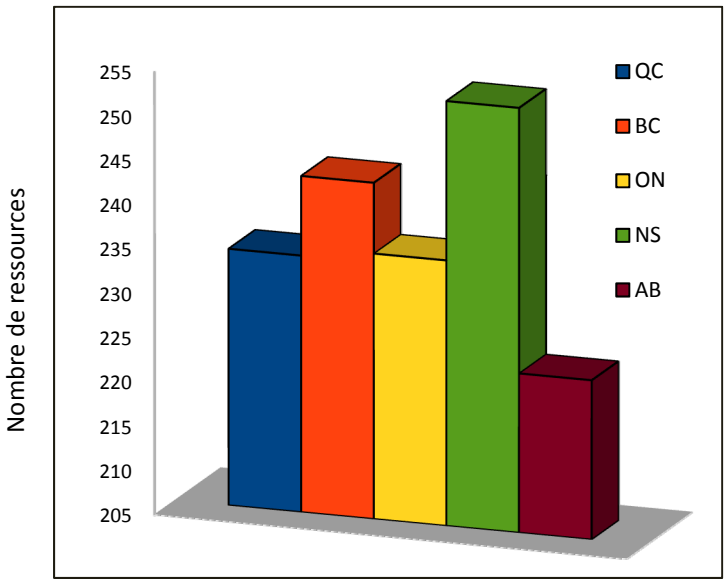

Figure 5.6 Distribution des ressources selon les provinces du Canada.

L'un des cas d'utilisation de GSN consiste à relocaliser l'application GeoChronos (Kiddle, 2010). Actuellement cette application est installée sur des serveurs de l'université de Calgary en Alberta. Cette application nécessite quarante-huit serveurs pour son fonctionnement.

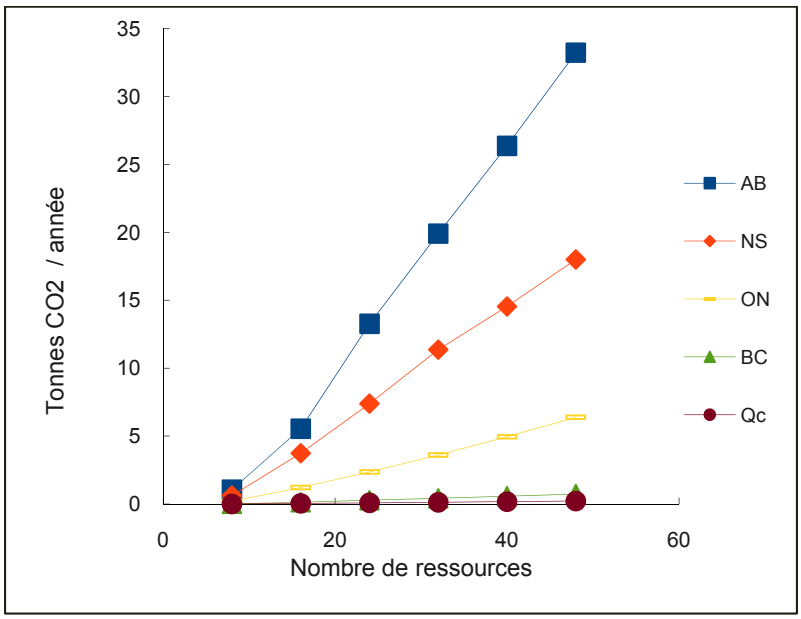

Figure 5.7 Émission du CO_2 selon la province pour l'application GeoChronos. Tirée de Daouadji (2010)

La figure 5.7 compare les émissions de CO_2 que générera le fonctionnement de cette application selon l'endroit où les serveurs seront installés. Il est évident que l'utilisation des ressources localisées au Québec réduira considérablement les émissions des GES. Cependant, la disponibilité des ressources souhaitées, en nombre suffisant, n'est pas toujours garantie. Dans les spécifications, le système qui gère le réseau GSN doit être autonome. Il doit détecter le besoin en ressource et en énergie et localiser les endroits où l'énergie renouvelable est disponible afin de déplacer éventuellement les tâches vers cet endroit. Cette fonctionnalité est prise en charge par le contrôleur du nuage informatique. Ce dernier doit être capable de localiser les ressources les plus optimales, aussi bien au niveau des performances, que de l'aspect écologique. Le cadre de découverte de ressources basé sur l'ontologie et les réseaux Bayésiens fournit les outils nécessaires au contrôleur du nuage informatique pour trouver les ressources nécessaires et optimales.

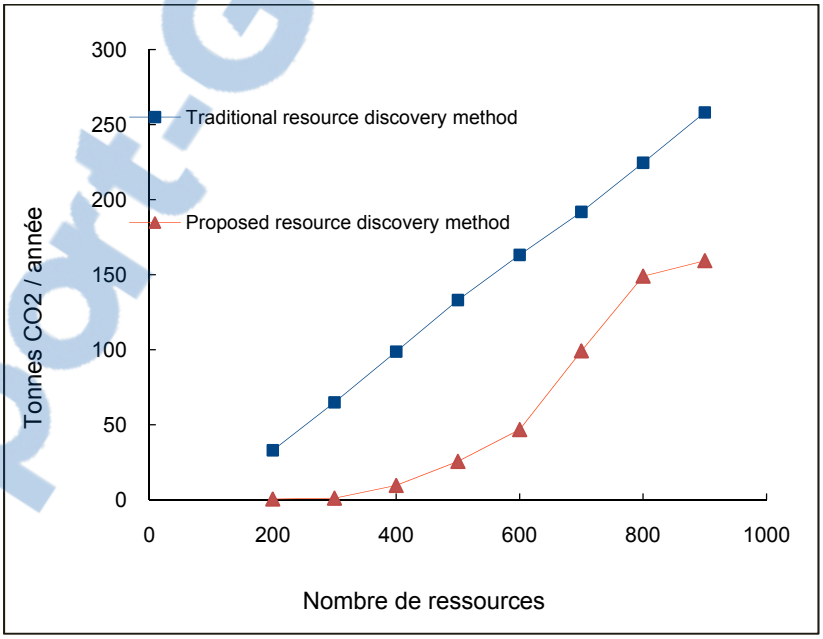

Figure 5.8 Comparaison des émissions du GES des deux méthodes
Tirée de Daouadji (2010).

La figure 5.8 compare les émissions de CO_2 résultants des ressources retournées avec les deux méthodes. La première méthode ne tient pas compte l'aspect environnemental. Les

premières ressources qui correspondent aux requêtes sont utilisées. Alors que dans notre méthode les ressources sont d'abord triées selon les types d'énergies. Les ressources les plus éco-énergétiques sont utilisées en premier. Une fois qu'il n'y a plus de ressources disponibles, on utilise celles qui sont les plus optimales, parmi les ressources restantes, de point de vue des émissions de GES. Les deux graphes ci-dessus montrent clairement que notre stratégie permet une réduction significative des émissions de CO_2

5.8 Conclusion

Dans ce chapitre nous avons présenté un cas d'utilisation où nous avons testé le modèle proposé. Les résultats obtenus démontrent la supériorité de notre modèle par rapport aux méthodes basées sur la correspondance des mots (pattern matching). Ces résultats s'expliquent par le fait que notre méthode ne se contente pas de retourner les ressources qui correspondent aux termes utilisés dans la requête, mais retourne aussi les ressources compatibles. Contrairement à la recherche textuelle qui retourne uniquement les ressources qui s'identifient parfaitement dans les termes utilisés dans les requêtes.

Les avantages de notre méthode ne se limitent pas uniquement dans les performances de la recherche. En utilisant une description des ressources à l'aide d'une ontologie, on possède davantage d'informations sur la nature des ressources disponibles dans le nuage. Ces connaissances permettent de faire des raisonnements. Un moteur de recherche de ressource basé sur une recherche plain texte peut trouver les ressources souhaitées, mais ne possède aucune connaissance sur ces ressources et les résultats retournés doivent êtres interprétés manuellement. Par contre, notre méthode qui est basée sur une recherche sémantique offre la possibilité que les résultats soient interprétés et utilisés par d'autres composantes telles que le contrôleur du nuage informatique.

En outre, notre méthode permet aussi la mise à l'échelle. En effet, nous avons vu comment l'ontologie peut être modifiée, et passer d'un modèle simple a un autre plus complexe, afin de cibler des objectifs spécifiques, comme la réduction des GES dans le cas de GSN.

CONCLUSION

Les demandes en matière d'infrastructure qui support les nouvelles technologies ne cessent d'augmenter. Ceci augmente les coûts liés à ces infrastructures. La virtualisation se positionne comme une solution prometteuse à ce problème. La technologie de grilles est considérée comme la première technologie qui propose la création des organisations virtuelles afin d'offrir des solutions à la croissance phénoménale des demandes en matière d'infrastructures. Cette technologie repose sur l'idée de regrouper des ressources physiques de petite taille pour former des ressources virtuelles de grande taille. Mais la mise en place d'une OV en utilisant cette technologie est une tâche très difficile.

La technologie des nuages informatique est l'une des technologies émergentes qui ont fait ses preuves. Cette technologie se base sur l'idée de virtualiser des ressources physiques en plusieurs ressources virtuelles. Ces dernières sont offertes aux utilisateurs sous forme de services. On parle alors d'infrastructure en tant que service. Nous avons vu que les nuages informatiques offrent plusieurs avantages (Gong, Liu, Zhang, Chen, & Gong, 2010) : du fait qu'ils sont orientés service. Ses services sont faiblement couplés ce qui augmente leurs tolérances aux problèmes. Dans ce mémoire nous avons introduit un cadre de travail dédié à la réalisation de l'infrastructure en tant que service, en l'occurrence IaaS.

Bien que les nuages informatiques soient considérés comme une technologie plus fiable que les grilles informatiques pour la mise en pratique des OV, plusieurs pensent que les deux technologies ont leurs places et envisagent même une collaboration entre les deux technologies dans le futur (Chen, Zhang, & Huo, 2010).
La création des ressources web dans les grilles, notamment à l'aide du Framework Globus, se fait selon les recommandations WSRF. Cette solution impose l'implémentation de plusieurs spécifications.

Les ressources dans un environnement virtuel sont caractérisées par une large gamme de diversité. On y trouve différentes configurations que ce soit matériels ou logiciels. Cette diversité rend la découverte de ressources, qui est primordiale à mettre en œuvre, difficile. Dans ce mémoire nous avons proposé un modèle pour la description et la recherche des ressources dans un environnement virtuel et en particulier le nuage informatique.
Dans un premier temps, nous avons présenté le contexte de nos travaux, où nous avons vu le Framework IaaS qui propose une façon simple pour la création des ressources web. En effet, il se base sur l'idée d'une ressource, d'un modèle et d'un ensemble de capacités au lieu d'implémenter les spécifications WSRF.

Dans un second temps, nous avons présenté le modèle de notre approche. Cette approche se base sur une modélisation à l'aide d'une ontologie qui définit les différents concepts qui décrivent le domaine cible. Quant à la découverte de ressources se base sur un réseau sémantique Bayésien. Et enfin, la représentation des ressources se fait à l'aide de fichier RDF.

Notre modèle ce caractérise par :

- Une description sémantique de ressources en utilisant l'ontologie qui permet de construire une base de connaissance.
- Une recherche sémantique basée sur l'inférence Bayésienne qui prend en considération les liens entre les mots de la même requête.
- Une représentation des ressources à l'aide de la norme RDF.
- Indépendance par rapport aux technologies.
- Une modularité et qui permet la mise à l'échelle évolutive.

Les tests sont effectués, sur des données générées aléatoirement dans le cadre du projet GSN, ont pour but de démontrer l'efficacité de notre méthode et sa supériorité par rapport à une recherche classique basée sur une correspondance mot à mot. Le présent travail a donné lieu

à deux publications (Daouadji, Nguyen, Lemay, & Cheriet, 2010) et (Arojo, Daouadji, Kamel, & Cheriet, 2010).

Perspectives

Il est clair qu'il reste une étape importante à implémenter, celle d'intégrer le module de recherche à celui du contrôleur et gestionnaire du Cloud GSN. Étant donné le but ultime du projet GSN est de concevoir un système autonome capable de prendre les décisions et qui garantit le bon fonctionnement du système ainsi que l'intégrité de données, et qu'il soit capable de trouver les ressources qui sont alimentées que par l'énergie renouvelable. Cette étape permettra de tester notre modèle sur des données réelles. Ceci va, sans doute, ouvrir la voie ver d'autres besoins qui nécessitent un système plus élaboré de gestion et de découverte des ressources web.

ANNEXE I

EXEMPLE D'UN FICHIER WSDL2.0

```
<wsdl:definitions name="WeatherImpService" targetNamespace="http://ca.project/">
<wsdl:types>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="unqualified"
targetNamespace="http://ca.project/">
<xs:element name="getWeather" type="tns:getWeather"/>
<xs:element name="getWeatherResponse" type="tns:getWeatherResponse"/>
<xs:complexType name="getWeather">
<xs:sequence>
<xs:element minOccurs="0" name="arg0" type="xs:string"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="getWeatherResponse">
<xs:sequence>
<xs:element minOccurs="0" name="return" type="tns:climate"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="climate">
<xs:sequence>
<xs:element minOccurs="0" name="conditionActuelles" type="xs:string"/>
<xs:element minOccurs="0" name="temperature" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
</wsdl:types>
<wsdl:message name="getWeather">
<wsdl:part element="tns:getWeather" name="parameters">
   </wsdl:part>
</wsdl:message>
<wsdl:message name="getWeatherResponse">
<wsdl:part element="tns:getWeatherResponse" name="parameters">
   </wsdl:part>
</wsdl:message>
<wsdl:portType name="Weather">
<wsdl:operation name="getWeather">
<wsdl:input message="tns:getWeather" name="getWeather">
   </wsdl:input>
<wsdl:output message="tns:getWeatherResponse" name="getWeatherResponse">
   </wsdl:output>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="WeatherImpServiceSoapBinding" type="tns:Weather">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getWeather">
<soap:operation soapAction="" style="document"/>
<wsdl:input name="getWeather">
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output name="getWeatherResponse">
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="WeatherImpService">
<wsdl:port binding="tns:WeatherImpServiceSoapBinding" name="WeatherImpPort">
<soap:address location="http://www.synchromedia.ca/weather"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

ANNEXE II

FICHIER RDF

```
<?xml version="1.0" encoding="windows-1252"?>
<rdf:RDF
    xmlns:j.0="http://www.Synchromedia.ca/#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:j.1="http://xmlns.com/foaf/0.1/#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://www.racforce#storage11">
    <j.1:hdd rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1000</j.1:hdd>
  </rdf:Description>
  <rdf:Description rdf:about="http://bastionhost.com#ciscorouter7">
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">3.33</j.1:cpu>
    <j.1:bandwidth rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">100</j.1:bandwidth>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://xam.de/foaf.rdf.xml#DelServer1">
    <j.1:province>ns</j.1:province>
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">3.33</j.1:cpu>
    <j.1:hdd rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">30</j.1:hdd>
    <j.1:OS>WindowsXP</j.1:OS>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.racforce#cisco35008">
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">3.33</j.1:cpu>
    <j.1:bandwidth rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">20</j.1:bandwidth>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.synchromedia.ca#dellrouter11">
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">3.33</j.1:cpu>
    <j.1:bandwidth rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">64</j.1:bandwidth>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2048</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://bastionhost.com#HPDesktop9">
    <j.1:province>bc</j.1:province>
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">2.8</j.1:cpu>
    <j.1:hdd rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">30</j.1:hdd>
    <j.1:OS>UbuntuServer</j.1:OS>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://xam.de/foaf.rdf.xml#IBMDesktop11">
    <j.1:province>bc</j.1:province>
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">2.8</j.1:cpu>
    <j.1:hdd rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">50</j.1:hdd>
    <j.1:OS>Ubuntu</j.1:OS>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://bastionhost.com#cisco2006">
    <j.1:cpu rdf:datatype="http://www.w3.org/2001/XMLSchema#double">2.8</j.1:cpu>
    <j.1:bandwidth rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">64</j.1:bandwidth>
    <j.0:memory rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1024</j.0:memory>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.synchromedia.ca#HPServer8">
    <j.1:province>ab</j.1:province>.
.
.
.
</rdf:RDF>
```

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Arojo, R., Daouadji, A., Kamel, M., & Cheriet, M. (2010). A Cluster-based Multiagent Model for Resource Discovery in Virtualized Environments. GRES. Montreal.

Balachandar R. Amarnath, T. S. (2009). Ontology-based Grid resource management. Software Practice and Experience SP&E .

Baud, J.-P., Casey, J., Lemaitre, S., & Nicholson, C. (2005). Performance analysis of a file catalog for the LHC computing grid. 14th IEEE International Symposium on High Performance Distributed Computing (pp. 91 - 99). Geneva: IEEE.

Chen, X., Zhang, S., & Huo, X. (2010). The comparison between cloud computing and grid computing. International Conference on Computer Application and System Modeling (ICCASM). Taiyuan : IEEE.

Ciurana, E. (2008). Developing with Google App Engine. Apress.

Cronon, W. (1992). Nature's Metropolis: Chicago and the Great Wes.

Daniel, M. (2005). Chapter 1 - Introduction. In A Networking Approach to Grid Computing. John Wiley & Sons.

Daouadji, A., Nguyen, K.-K., Lemay, M., & Cheriet, M. (2010). Ontology-Based Resource Description and Discovery Framework for Low Carbon Grid Networks. First IEEE International Conference on Smart Grid Communications (SmartGridComm) (pp. 477-482). Gaithersburg, MD: IEEE.

Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., et al. (2000). The Semantic Web: the roles of XML and RDF. Internet Computing, IEEE , 63.

Grasa, E., S. F. (2008). Extending the UCLP Software with a dynamic optical multicast service to support high performance digital media. International Conference on Optical Network Design and Modeling. (p. 1). Vilanova i la Geltru: IEEE.

F. Taroni, C. A. (2006). chapitre 2-4. In Bayesian Networks and Probabilistic Inference in Forensic Science. John Wiley & Sons.

Foster, I. (2001). The Anatomy of the Grid : Enabling Scalable Virtual Organizations. Springer-Verlag .

Globus. (2006). http://www.globus.org/toolkit/docs/4.2/4.2.0/info/key/mds4KeyConcepts.pdf. Retrieved from http://www.globus.org.

Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2010). The Characteristics of Cloud Computing. International Conference on Parallel Processing Workshops (pp. 275-279). San Diego, CA: IEEE.

Hardy, S. B. (2008). Compass Reference Documentation. Retrieved from http://www.compass-project.org/: http://www.compass-project.org/docs/2.2.0/reference/pdf/compass-reference.pdf

Jennifer M. Schopf, L. P. (2006). Monitoring the grid with the Globus Toolkit MDS4. Journal of Physics , 521-525.

Kesselman, I. F. (2004). Concepts and Architecture. In Kesselman, I. Foster, & Carl, The Grid: Blueprint for a New computing Infrastructure (p. 42). Chicago: Morgan Kaufmann.

Kesselman, I. F. (2004). Service Virtualization:Infrastructure and Applications. In I. F. Kesselman, The Grid: Blueprint for a New computing Infrastructure (p. 183). Chicago: Morgan Kaufmann.

Kiddle, C. (2010). GeoChronos: A Platform for Earth Observation Scientists. OpenGridForum, (p. 28).

Kim, K.-M., Hong, J.-H., & Cho, S.-B. (2005). Intelligent Web interface using flexible conversational agent with semantic Bayesian network. NWESP '05 Proceedings of the International Conference on Next Generation Web Services Practices. Washington: IEEE .

Lacey, P. (2004). UDDI & dynamic Web service discovery. Dr. Dobb's Journal .

Li, M. a. (2005). Chapter 2 - OGSA and WSRF. In The Grid: Core Technologies. John Wiley & Sons.

LivClean. (2010, October 6). http://www.livclean.ca/PE_offset_calculation.pdf. Retrieved from http://www.livclean.ca.

Schliefnig M. (2009, January 20) The Semantic-Web-Trust-Layer: Legal and Social Effects

Mittal, A. a. (2007). Bayesian Network Technologies: Applications and Graphical Models. IGI Global.

Morin, C. (2007). XtreemOS: A Grid Operating System Making your Computer Ready for Participating in Virtual Organizations. 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. (p. 393). Santorini Island : IEEE.

Moulard, G. (2009). Le Cloud Computing principes, état de l'art, avantages, acteurs. Pôle Recherche & Développement groupe HSBI.

Myerson, J. M. (2009). Cloud computing versus grid computing : Service types, similarities and differences, and things to consider. IBM.

Nguyen, K.-K. (2010). Inocybe IaaS Framework: An Overview. Montreal: Synchromedia.

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your. Stanford: Stanford Knowledge Systems Laboratory.

OASIS. (2006, April 1). Web Services Base Faults 1.2. Retrieved from http://docs.oasis-open.org/: http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf

OASIS. (2004, june 10). Web Services Base Notification. Retrieved from http://docs.oasis-open.org: http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

OASIS. (2004, June 10). Web Services Resource 3 Lifetime. Retrieved from http://docs.oasis-open.org: http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-03.pdf

OASIS. (2005, October 7). Web Services Service Group 1.2. Retrieved from http://docs.oasis-open.org: http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-pr-02.pdf

Pahlevi, S., & Kojima, I. (2005). Towards Automatic Service Discovery and Monitoring in WS-Resource Framework. . First International Conference on Semantics, Knowledge and Grid. (p. 106). Beijing: IEEE.

Rubio, G. M. (2009). Chapter 1 - Introduction to OSGi. In Pro SpringSource dm Server. Apress.

Saunders, H. D. (1992). The Khazzoom-Brookes Postulate and Neoclassical Growth. The Energy Journal , 130-148.

Shah, S. (2007). Chapter 2 - Web Services Components. In Hacking Web Services. Cengage Charles River Media.

SOGETI. (2009). État de l'art Cloud Computing Livre blanc. Paris: SOGETI.

Sotomayor, B. a. (2006). Globus Toolkit 4: Programming Java Services. Morgan Kaufmann Publishers.

Sushil Bhardwaj, L. J. (2010). Cloud computing: a study of infrastructure as a service. International Journal of Engineering and Information Technology IJEIT , 60-63.

Thierry, C. (2003). Le projet européen de grille de calcul (DataGrid) Concept de grilles de calcul. Objectifs du projet et état d'avancement. Lyon: Conservatoire National des Arts et Métiers.

Vladimir, S. (2006). Chapter 1 - The Roadmap to High Performance Computing. In Grid Computing for Developers. Cengage Charles River Media.

W3C. (2004, August 10). Web Services Addressing . Retrieved from http://www.w3.org: http://www.w3.org/Submission/ws-addressing/

W3C. (2004, June 10). Web Services Architecture. Retrieved from http://www.w3.org: http://www.w3.org/TR/ws-arch/

Wikipedia. Thésaurus. Retrieved 01 10, 2011, from http://www.wikipedia.org/: http://fr.wikipedia.org/wiki/Th%C3%A9saurus

Fataicha, F., M. C. (2005). Retrieving poorly degraded OCR documents. International Journal of Document Analysis .