Table des matières

Remerciements

Je remercie le département de Mathématiques et d'Informatique pour m'avoir accueilli à l'occasion de ce stage.

Je tiens aussi à remercier l'ensemble des personnes des équipes réseaux et base de données et plus particulièrement les chefs d'équipes respectifs : NIANG Ibrahima, et NDIAYE Samba.

Je remercie aussi mes encadreurs Yahya SLIMANI, NDIAYE Samba, NIANG Ibrahima pour leurs précieux conseils et leur soutien tout au long de ce stage.

Je remercie tous les enseignants de la section informatique pour le soutien

Enfin, j'ai une pensée pour tous mes collègues de travail pour leur aide et la bonne ambiance qu'ils ont pu apporter.

iii

Chapitre 2

ETAT DE L'ART SUR LA DISTRIBUTION

Les besoins en informations et l'utilisation des données peuvent être variés en fonction du site. De ce fait, une organisation centralisée des données peut être non adéquate à cette architecture répartie. Par exemple un entrepôt de données réparti pourra répondre plus efficacement aux besoins des utilisateurs. Les données peuvent être organisées par sujet et une meilleure utilisation de l'entrepôt est garantie. Cependant, nous présentons l'intérêt de la répartition en présentant trois systèmes répartis.

II-1 REPARTITION DES RESSOURCES

Les systèmes informatiques centralisés permettent une utilisation optimale des ressources mais surtout une gestion des données en mode centralisé ; ils se prêtent très bien aux traitements par lots (*batch*).

En revanche, dans le cas des entrepôts des données, des contraintes signifient que la centralisation peut se refléter négativement sur sa performance et ses objectifs. Les systèmes centralisés ont subi une dégradation due principalement aux avancées considérables de l'informatique qui ont favorisé l'émergence de nouvelles architectures constituées d'ordinateurs géographiquement distribués. Il s'agit des systèmes distribués (figure 2.1).

De nouveaux concepts sont proposés pour désigner ces systèmes, tels que: *multi-ordinateur*, ou *réseau de stations de travail* (*Network of Workstation*) ou, plus récemment, réseau pair à pair (*Peer-to-Peer Computing*) ou grille de calcul (*Grid Computing)* **[1].**

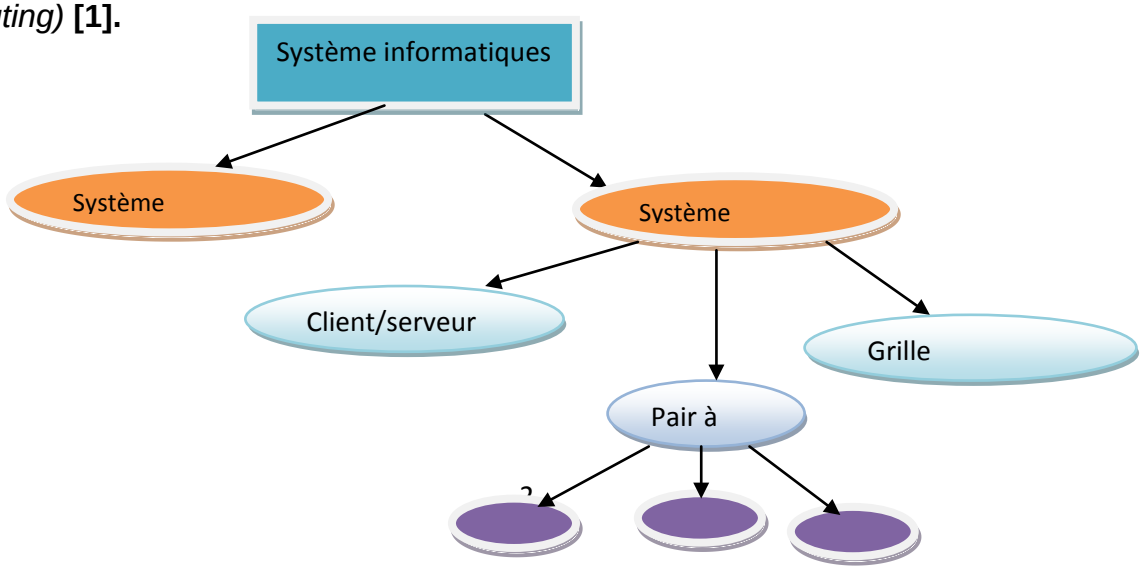


Figure 2.1 : Systèmes informatiques

II.1.1 Les systèmes répartis

a) Principe

Un système réparti englobe une classe de systèmes classiques comme : système client-serveur, système multi-agents. Ils sont caractérisés par un principe de fonctionnement simplifié et permettent une gestion de données efficace à certaine échelle. Bien que, une multitude de travaux ont été proposées par la plus part des équipes de recherche dans ce domaine.

b) Avantages

Une gestion décentralisée nécessite quelques points qui sont :

- Plusieurs machines de natures différentes peuvent être reliées et ainsi pouvoir échanger de l'information. La configuration d'une architecture distribuée peut croître en fonction des besoins réels des utilisateurs.
- Une plus grande disponibilité c-à-d un élément défaillant d'une architecture peut être mis hors service sans entraîner un arrêt complet du système. Les traitements effectués par cet élément défaillant peuvent être éventuellement repris sur une autre machine de l'architecture distribuée.

c) Inconvénients

Sensibilité et instabilité aux défaillances : pour chaque nœud, il existe un chemin à tout autre nœud du graphe alors les sites peuvent communiquer soit directement, soit indirectement via les canaux de communications par conséquence, la défaillance d'un canal ou d'un processus modélise une défaillance affectant un nœud ou une ligne du réseau.

Avec l'architecture client-serveur, la mémoire commune est localisée sur un ou plusieurs sites serveurs. Elle n'est pas directement visible par les sites clients, mais seulement par l'intermédiaire d'une interface procédurale. Quand les clients accèdent fréquemment à la mémoire commune, le serveur devient goulot d'étranglement, et le coût de communication devient prohibitif : le client-serveur ne passe pas à l'échelle. Ceci montre l'intérêt de la réplication, sur « le partage d'information dans les systèmes répartis de grande échelle »

D’autre part, dans le paradigme des agents mobiles, la mémoire partagée est l’union des mémoires locales. C’est le programme applicatif lui-même qui se déplace vers les données dont il a besoin.

c) Bilan

Si les systèmes classiques ont permis la mise en œuvre de mécanismes efficaces de partage de données à petite échelle, d'autres types de systèmes répartis se sont focalisés sur la gestion des données à grande échelle. Cependant, un autre paradigme de stockage et calcul global a récemment focalisé l’intérêt de la communauté scientifique : systèmes pair-à-pair (*peer to peer*). Ce modèle complémente le monde classique *client-serveur* qui est aujourd’hui à la base de la plupart des traitements sur internet. Les systèmes pair-à-pair ont permit d'agréger des centaines de milliers de nœuds. En effet, les systèmes pair à pair sont souvent plus tolérants aux pannes, assent plus facilement à l ’échelle, et sont plus adaptatifs que leurs contreparties client/serveur**.**

II. 1.2 Les systèmes pair-à-pair

a) Principe

Pair-à-pair désigne une classe d’applications informatiques dédiées à l’échange point multipoint. Ces applications symétrisent la relation des machines (noeuds) qui interagissent.

b) Avantages

Les avantages des systèmes pair-à-pair ou systèmes dits à grande échelle sont nombreux. Ils donnent accès à un grand nombre de ressources, dont le nombre de pairs est supérieur à 10x (x > 2).

Ils disposent d'une administration transparente. Un système pair à pair évolue sans machine(s) dédiée(s) pour son administration. Ces systèmes sont conçus de telle sorte qu'aucun des pairs ne soit indispensable au fonctionnement général : le système n'est pas paralysé par la défaillance d'un ou plusieurs pairs**.** Les qualités de ce système (robustesse, disponibilité, performances...) augmentent avec le nombre d'utilisateurs, et présentent de nombreux avantages (décentralisation, pas de coûts d'infrastructure...).

c) Inconvénients

En contrepartie, de par la volatilité des pairs, le réseau est non fiable. De plus, des problèmes de sécurité peuvent survenir à cause de certains pairs malveillants non écartés du réseau**.**

Ce système ne peut pas assurer la confidentialité des données échangées et les vitesses de transfert sont aléatoires. Chaque utilisateur faisant office de client et de serveur, ils doivent partager leur bande passante entre ces deux activités. Cette bande passante doit, de plus, être partagée entre tous les clients qui téléchargent en même temps ce fichier, par conséquent, les temps de téléchargement peuvent être relativement long.

Donc, une ressource sur un tel système a une durée de vie parfois limitée. Un fichier accessible sur un réseau pair à pair peut très bien ne plus l'être dans l'heure qui suit. Il suffit pour cela que les rares possesseurs de ce fichier ne le mettent plus à disposition au même moment**.**

d) Bilan

Bien que le système pair à pair impose de par sa nature une gestion complexe, mais dispose d'une administration transparente et un accès à un grand nombre de ressources. Aujourd'hui, ces systèmes apparaissent sous différentes applications l'une différente de l'autre, en terme de qualité de service et les exceptions marquées entre ces dernières, d'après les rapports techniques et études comme ceux présenté dans **[2]**.

Finalement, les inconvénients des systèmes pair-à-pair participent grandement à la diminution de l'efficacité d'un tel système, ce qui favorise de penser à l'optimisation de l'ensemble de protocoles et techniques faisant partie de l'architecture pair-à-pair. Par conséquent, nous allons faire un passage à l'analyse d'un autre système à grand échelle: les Grilles informatiques.

II.1.3 Les Grilles informatiques.

Dès les débuts de l'informatique, les scientifiques furent les plus gros consommateurs de puissance de calcul. Depuis, malgré les améliorations sans cesse croissantes des performances des ordinateurs modernes, ceux-ci n'ont jamais réussi à satisfaire pleinement les exigences de la communauté scientifique, tant leurs simulations et modèles gagnaient en complexité.

De plus en plus de projets de recherche impliquent de multiples partenaires pouvant être répartis aux quatre coins du globe. Il devient alors nécessaire de disposer d'une infrastructure commune, facilitant les partages d'informations mais aussi de ressources.

La dernière décennie a également vu l'avènement des réseaux et d'Internet. Ainsi, il ne suffit plus de travailler efficacement mais il faut également pouvoir le faire ensemble.

C'est dans ce contexte, mêlant hautes performances et travail collaboratif, qu'est né le concept de Grille de Calcul.

Le développement des technologies de grille informatique est un axe de travail privilégié actuellement, dans la mesure où la grille représente un outil prometteur pour la mutualisation des ressources informatiques.

Ce chapitre a pour objectif de mettre en lumière les possibilités et les contraintes des grilles informatiques. Il débute par une présentation des grilles informatiques et des grilles de calcul, suivie d'une étude sur les infrastructures matérielles et logicielles. Il se termine par une étude approfondie d'un système d'exploitation de grille.

A) Principe

La grille informatique a été nommée ainsi par analogie avec le système de distribution d'électricité américain (Electric Power Grid). Ce terme été répandu en 1998 par l'ouvrage de Ian Foster et Carl Kesselman [3]. La vision des inventeurs de ce terme est qu'il sera possible, à terme, de se brancher sur une grille informatique pour obtenir de la puissance de calcul et/ou de stockage de données sans savoir ni où ni comment cette puissance est fournie, à l'image de ce qui se passe pour l'électricité.

Une grille informatique mutualise un ensemble de ressources informatiques géographiquement distribuées dans différents sites.

Un site d'une grille peut être défini selon Sébastien Lacour [4] comme étant un ensemble de ressources informatiques localisées géographiquement dans une même organisation (campus universitaire, centre de calcul, entreprise ou chez un individu) et qui forment un domaine d'administration autonome, uniforme et coordonné. Les ressources informatiques sont aussi bien des liens réseau (câbles, routeurs ou switchs) des machines (simples PC ou calculateurs parallèles) ou des éléments logiciels.

L'introduction de la notion d'« organisation virtuelle » (virtual organization, VO) permet de prendre en compte des aspects socio-économiques dans la définition de la grille. Une organisation virtuelle est un ensemble de sites de grille indépendants, qui se crée

et se dissout dynamiquement au gré de leurs besoins en ressources informatiques : les sites d'une organisation virtuelle partagent une partie de leurs ressources informatiques de manière contrôlée.

Cette infrastructure est qualifiée de virtuelle car les relations entre les entités qui la composent n'existent pas sur le plan matériel mais d'un point de vue logique.

Figure 2.2 : Grille informatique

D'après Ian Foster, une grille consiste en three point checklist (trois critères). Ces points sont liés à la gestion des ressources, un mécanisme permettant d'interroger chaque nœud doit être conçu pour savoir son état, sa charge de travail, …, l'utilisation de protocoles standardisés et la qualité de service.

Une grille informatique peut être définie comme étant l'ensemble des ressources informatiques mises en commun et partagées par les sites d'une organisation virtuelle, ainsi que les moyens de communication entre ces sites, et la spécification de la politique d'accès aux ressources de chaque site par chaque utilisateur.

A-1- Eléments constitutifs

Un système de grille repose sur les éléments suivants :

• des moyens matériels, systèmes de traitement et de calcul (PC, stations de travail, clusters,…) et systèmes de stockage ;

• des mécanismes de communication par des réseaux haut-débit reliant les différents centres ;

• des services GRID réunis au sein d'un Middleware (intergiciel) ;

• des boites à outils génériques (outils de visualisation, bibliothèques de données …) ;

• des logiciels d'applications spécifiques adaptés à l'architecture grille.

Bien que chaque projet ait sa propre architecture, une architecture générale (figure 2.3) est importante pour expliquer certains concepts fondamentaux des grilles.

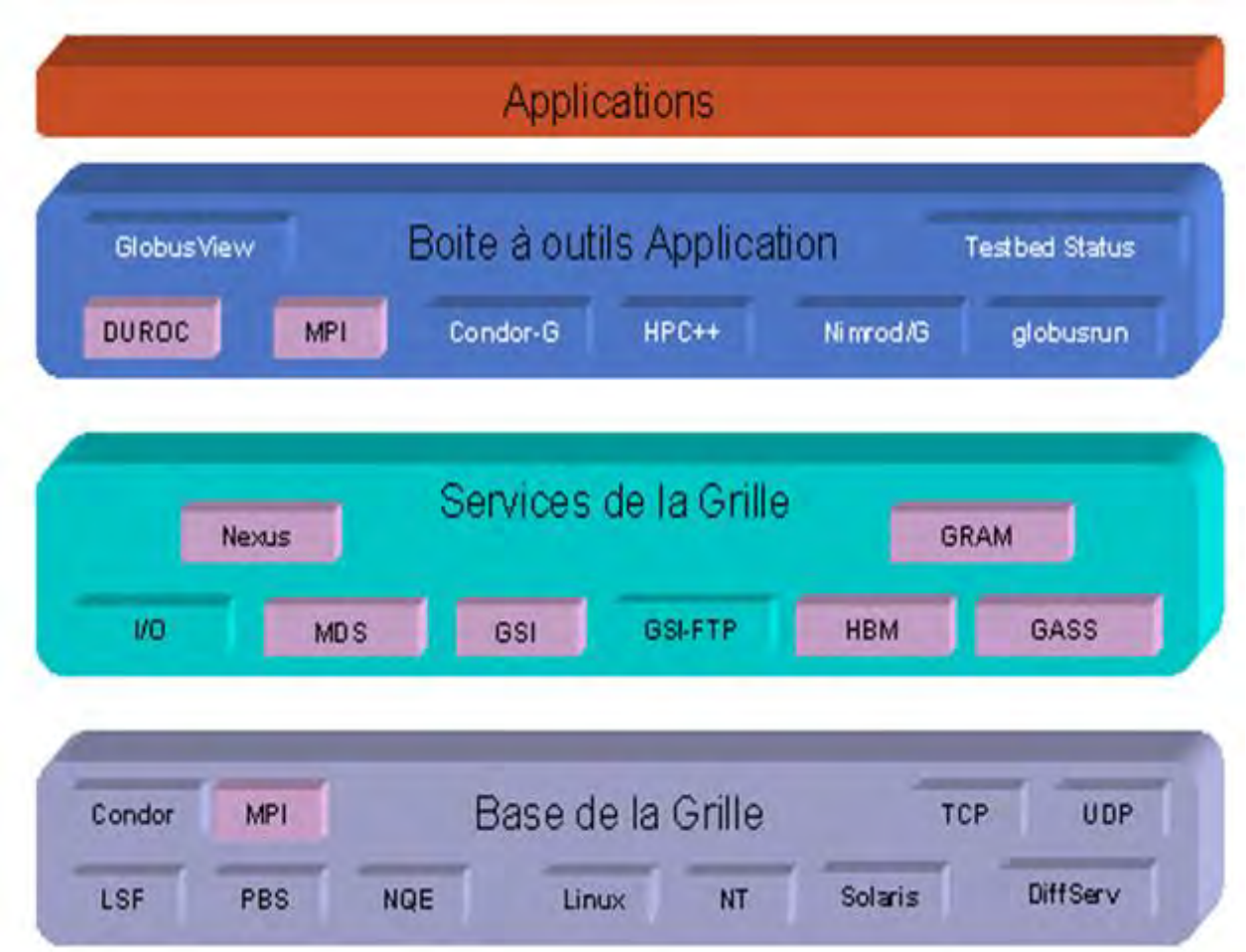


Figure 2.3 : Architecture en couche d'une grille de calcul

A-2 Fonctionnement d'une grille : le middelware

Afin de pouvoir constituer une grille, il est nécessaire d'avoir recours à différents outils qui vont nous permettre de gérer efficacement les ressources disponibles au sein d'une grille. On peut considérer cet ensemble d'outils comme le système d'exploitation de la grille, appelé intergiciel ou encore middelware.

Une grille est une collection de ressources dont pourra bénéficier l'utilisateur. L'ensemble des ressources distribuées constituent la base de la grille. Il est donc impératif de définir clairement quelles vont être ces ressources : postes de travail, serveurs, logiciels, stockage, etc. Mais également des éléments plus complexes comme

un système de gestion de fichiers distribuée, un cluster, ou même une autre grille. Dans certains cas, il est possible que ces ressources soient dispersées en plusieurs endroits, et que chacune soit soumise à des règles différentes. Ainsi, il est indispensable de recenser, d'identifier ces éléments, mais aussi de le décrire le plus précisément possible.

En ce qui concerne la description des ressources, l'idéal serait que chaque élément soit en mesure de se décrire soi-même, ce qui est parfois le cas.

Ces ressources sont en générale réparties en plusieurs types [1] :

- Un **Computing Element** (CE) est une ressource qui offre de la puissance de calcul (Cycles processeur).
    - Un Storage Element (SE) est une ressource offrant une espace pour stocker des données. Un storage element peut aussi offrir des prestations annexes comme la sauvegarde, l'accès via de multiples protocoles, et la pré-allocation d'espace de stockage.
- Equipements spéciaux et logiciels à licence élevée : certains logiciels dont les prix de licence sont élevés seront présents en quelques exemplaires seulement dans la grille. Cette grille permettra donc en les exposants à beaucoup d'utilisateurs, une meilleure utilisation de ces logiciels. Il en va de même pour certains équipements spéciaux comme les microscopes électroniques et les appareils médicaux …
- Un Resource Broker (RB) est un intergiciel qui transfert l'exécution du travail des clients sous forme de "job" au CE le plus probable dans un environnement de calcul hétérogène. Le RB est en charge de la transmission des requêtes afin de satisfaire les demandes clients, en listant les ressources les plus probables pour l'exécution du job, et la récupération des résultats de sortie de ce job

Ces ressources sont potentiellement qualifiées de :

- Partagées : elles sont mises à la disposition des différents consommateurs de la grille et éventuellement pour différents usages applicatifs.
- Distribuées : elles sont situées dans des lieux géographiques différents.
- Hétérogènes : elles sont de toute nature, différant par exemple par le système d'exploitation ou le système de gestion des fichiers :

- Moyens de calcul. En termes de puissance de calcul, on peut aussi bien trouver des supercalculateurs que des ordinateurs de bureau (PC), des serveurs d'exécution, des stations de travail, etc.
- Architectures. En termes d'architecture matérielle, les ordinateurs peuvent être équipés de différents types de processeurs : PowerPC, compatibles i386, des Alphas, des Mips, etc.
- Logiciels. En termes d'installation logicielle, les ordinateurs peuvent avoir différents systèmes d'exploitation avec une version précise. Les logiciels disponibles et leurs versions peuvent également être différents et installés à des endroits variés (compilateurs, bibliothèques de calcul, etc.).
    - Réseaux. En termes de réseaux d'interconnexion entre les ordinateurs, les liens de communication peuvent avoir des débits, latences, gigues, taux de pertes différents.
- Politiques d'accès aux ressources. Chaque site d'une grille décide de façon autonome et indépendante des politiques d'authentification, d'autorisation d'accès aux ressources, et d'attribution des noms d'utilisateur.

- Coordonnées : les ressources sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements).
- Non contrôlées (ou autonomes) : les ressources ne sont pas contrôlées par une unité commune. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle.
- Délocalisées : les ressources peuvent appartenir à plusieurs sites, organisations, réseaux et se situer à différents endroits géographiques.

B) Avantages

Les grilles peuvent être employés en science, dans l'industrie avec par exemple les applications commerciales telles que la fabrication de médicaments, dans la conception automobile, la simulation d'accident, la physique énergétique et en modélisation aérospatiale, en astrophysique, en DAO électronique, dans la modélisation de la

terrestre, dans l'exploitation de données, dans la modélisation financière, et dans d'autres disciplines.

Les grilles touchent un peu à tous les domaines car la partie logiciel est assez complexe et requière l'utilisation de k its de dév eloppement, de pr otocoles et de s ervices spécialement conçu pour ce types d'utilisation, le but étant de connecté plusieurs ordinateurs généralement différents.

C'est ainsi que de nombreuses organisations y font appel afin de procéder à des projets pouvant représenter plusieurs millions de machines à t ravers le monde et ainsi de profiter d'une formidable puissance de c alcul dépassant l'imagination et à un c oût considérablement inférieur à une solution matérielle.

Les grille en tant que nouvel outil informatique a permis de s oulever de nouveaux verrous non seulement sur le plan du déploiement et de l 'ingénierie mais aussi des verrous scientifiques relatifs à la performance, au facteur d'échelle, à la dynamique, à la robustesse, à la sécurité et à la flexibilité.

En effet, les techniques de sécurité dans les grilles de calcul été exploiter par quelques approches voir par exemple [5]. Pour cela de m écanismes d'authentification (des certificats, etc.) sont employé pour assurer la sécurité entre les nœuds de la grille. La soumission des travaux par les nœuds s'avère très efficace avec une transparence de traitement, par exemple les résultats de l 'expérience CIGRI où m oins de 0,01% des tâches se sont terminées sans que le système ait su les gérer ou que l'utilisateur n'ait tenté de les resoumettre.

Enfin, la grille informatique puise sa force et puissance du f ait quelle peut être vue comme un système parallèle (pour aller plus vite) ou comme un système distribué (pour prendre en compte la localisation géographique des ressources) ou bien une combinaison des deux [6].

Aujourd'hui, la gestion des grilles fait l'objet de plusieurs thèmes de recherche. C'est le Globus Toolkit, issu des travaux du Laboratoire d'Informatique d'Argonne, dans l'Illinois, qui fait figure de référence dans la gestion des grilles. C'est d'ailleurs l'étude détaillé du système d'information de cet intergiciel que nous allons voir dans le chapitre qui suit

Chapitre 3

ETUDE DETAILLE DU SYSTEME D'INFORMATION MDS DE GLOBUS

III-1 L'INTERGICIEL D'ACCES AUX GRILLES DE CALCUL GLOBUS.

Globus [42, 43] est un intergiciel sous forme de boîte à outils logicielle (implémentée sous la forme de bibliothèques et de petits programmes en ligne de commande) qui permet d'accéder aux ressources d'une grille. Globus va ainsi gérer les aspects de gestion des ressources et des données, de sécurité, de traitement des travaux, de qualité de service, de communication et d'adaptation [8].

L'essentiel de cet intergiciel est développé à Argonne National Laboratory (États-Unis) par une équipe de recherche dirigée par Ian Foster. Le Globus Toolkit est l'intergiciel d'accès aux grilles le plus utilisé à travers le monde, comme l'illustrent des projets tels que GriPhyN [10], DOE Science Grid (initiative du département de l'énergie américain, [9]), DataTAG (projet européen, [11]), TeraGrid [12], etc.

III-1.1 La sécurité

L'infrastructure logicielle de sécurité de Globus est GSI (Grid Security Infrastructure, [45]) : elle permet l'authentification sécurisée et le chiffrement des communications. GSI est fondé sur le chiffrement à clef publique (certificats X.509) et le protocole de communication SSL (Secure Sockets Layer). Un utilisateur de Globus doit initialiser un proxy (un certificat signé par l'utilisateur) en tapant un mot de passe long. Ce certificat permet d'authentifier l'utilisateur auprès des ressources lors de la soumission de tâches à exécuter sur la grille. Ce proxy permet plusieurs soumissions tandis que l'utilisateur n'a eu besoin de taper son mot de passe qu'une seule fois (single sign-on). GSI est complété par CAS (Community Authorization Service, [59]), qui permet à chaque site de la grille de contrôler l'accès à ses ressources à gros grain (groupes de ressources, groupes d'utilisateurs) et à grain fin (ressources individuelles, utilisateurs individuels).

Il faut noter que ce processus d'authentification et d'autorisation du proxy nécessite l'établissement d'une chaîne de confiance (chain of trust) de l'autorité de certification jusqu'au Proxy en passant par l'utilisateur.

III-1.2 La gestion de ressources et de données

Après identification et authentification des ressources, il ne reste qu'à exécuter les travaux sur ces ressources. Cette étape se fait en utilisant un langage particulier, le RSL pour Ressource Spécification Language [13].

Les utilisateurs sont identifiés par un certificat X.509 qui est converti sur chaque ressource en un nom d'utilisateur local (username Unix par exemple). Pour soumettre à Globus une tâche à exécuter, l'utilisateur doit écrire un script RSL, qui indique, à bas niveau, le ou les fichiers exécutables à lancer, et sur quelles ressources de la grille ces exécutables doivent être lancés. RSL permet aussi de spécifier l'environnement des exécutables (répertoire de travail, variables d'environnement), et les fichiers à rapatrier sur chaque ressource de calcul.

III-1.2.1 Gestionnaire d'allocation de ressource (GRAM)

GRAM (Globus Resource Allocation Manager, [14]) est le module de gestion des ressources de la grille : il supporte la soumission et le contrôle de tâches sur des ressources distantes sans tenir compte de l'hétérogénéité locale (système d'exploitation, ordonnanceur), et il utilise GSI pour l'authentification mutuelle des ressources et des utilisateurs.

Une architecture en couche permet de définir des courtiers de ressources et des co-allocateurs spécifiques aux applications comme étant des services GRAM. La porte d'entrée sur une ressource, permettant de recevoir une requête d'exécution de travaux, s'appelle le « Gatekeeper ». C'est le démon Globus du service GRAM. Cette entrée est unique (voir schéma de l'architecture du GRAM), et elle utilise un port réseau de la ressource, le port TCP 2119, approuvé par l'organisation IANA. C'est ce GateKeeper qui s'interface avec les systèmes de batch locaux.

Le schéma suivant, montre le fonctionnement de l'architecture du GRAM, en association avec le service d'information (MDS) et de sécurité (GSI), lors de l'exécution de travaux sur des ressources. Cette exécution comprend une requête d'allocation de ressources et la création de processus.

Remarque : Le « job manager », ou « scheduler » fait une analyse grammaticale, en anglais « parse », si besoin est, de la requête au format RSL, avec la bibliothèque correspondante.

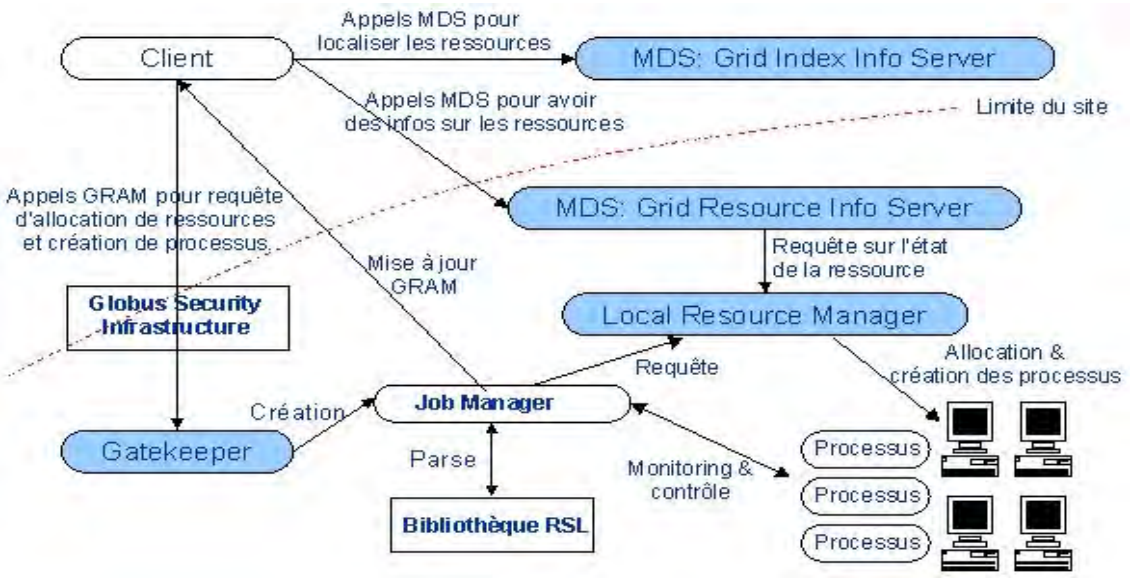


Figure 3.10 Architecture du GRAM

La figure 3.11, montre le fonctionnement de l'architecture de la gestion de ressources du GRAM. Dans le cas d'une machine, possédant plusieurs « scheduler » différents (dans notre exemple LSF, Easy et NQE), cette architecture nous permet de distinguer un seul type de gestionnaire de ressources. A la réception d'une requête par le courtier, celui ci l'envoie vers le co-allocateur qui va « dispatcher » le travail vers les différents gestionnaires de ressources locaux. Le module « application » permet de fournir des librairies propres à l'application en cours. Le module « service d'information » se met à jour suivant l'occupation des ressources locales.

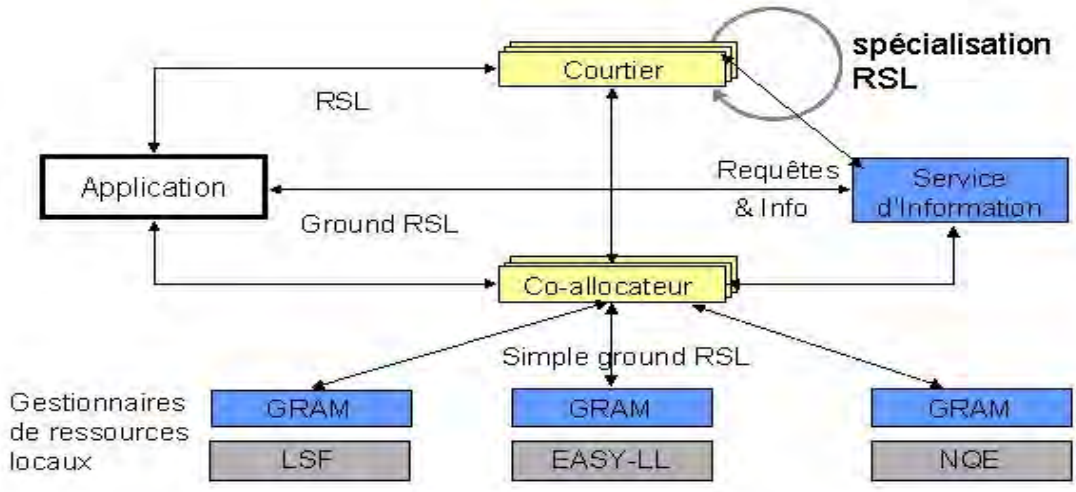


Figure 3.11 Architecture de la gestion de ressources

Dans le cas d'une utilisation de commandes de types multi-requêtes, nous allons allouer simultanément un ensemble de ressources. Pour ceci, Globus dispose d'un composant

appelé DUROC, pour « Dynamically Updated Request Online Co-allocator », ou co-allocateur de requêtes mise à jour dynamiquement.

III-1.2.2 Gestionnaire de données (GASS)

Ce module GASS, pour Global Access to Secondary Storage, va nous permettre d'accéder aux fichiers distants. Ce module va gérer la lecture et l'écriture de données dans des flux d'entrée, de sortie, et d'erreur. L'accès aux fichiers se fait par URL, du type HTTPS://machine:numéro_port. Ce module inclus l'utilisation de cache mémoire

Le module de gestion des données dans Globus est représenté par :

- GASS (Global Access to Secondary Storage, [15]), qui permet de transférer les fichiers d'entrée vers les noeuds de calcul avant exécution, et de rapatrier les fichiers de sortie après exécution ;
- GridFTP [16, 17], un protocole de transfert de fichiers haute performance, sécurisé (grâce à GSI), fiable, optimisé pour des réseaux de communication longue distance à fort débit ;
- RFT (Reliable File Transfer, [18]) permet de contrôler le transfert de fichiers entre deux serveurs GridFTP distants : avec RFT, il est possible d'initier des transferts de fichiers, puis de quitter sa session, et de reprendre sa session depuis une autre machine en reprenant le contrôle du transfert qui s'est poursuivi pendant toute la durée où la session de l'utilisateur était interrompue ;
- RLS (Replica Location Service, [20, 19]) permet de convertir des noms logiques d'éléments de données en des pointeurs (URL) vers les localisations physiques de ces données, éventuellement répliquées.

III-1.*2.3* Le service d'information

Il s'agit du module MDS (Monitoring & Discovery System, [21, 22]) chargé de la gestion de l'information au niveau de la grille. Il permet le stockage d'informations distribuées (statiques et dynamiques) sur les ressources de la grille ainsi que leur localisation. Il est organisé de façon hiérarchique et extensible. Le MDS inclut les mécanismes d'authentification des ressources et des utilisateurs de GSI.

III -2 ETUDE DETAILLEE DU MDS

III-2.1 Fonctionnement

Le besoin d'informations est un point crucial pour les opérations sur la grille et la construction d'applications. Ainsi on a parfois besoins de savoir comment une application détermine quelles ressources sont disponibles, mais surtout « l'état » de la grille à tout instant.

Il faut un système d'information général pour répondre à de telles questions. Le fonctionnement d'une grille dépend de la disponibilité d'informations telles que :

- la configuration des ressources
- l'état instantané d'une ressource
- les informations sur les applications

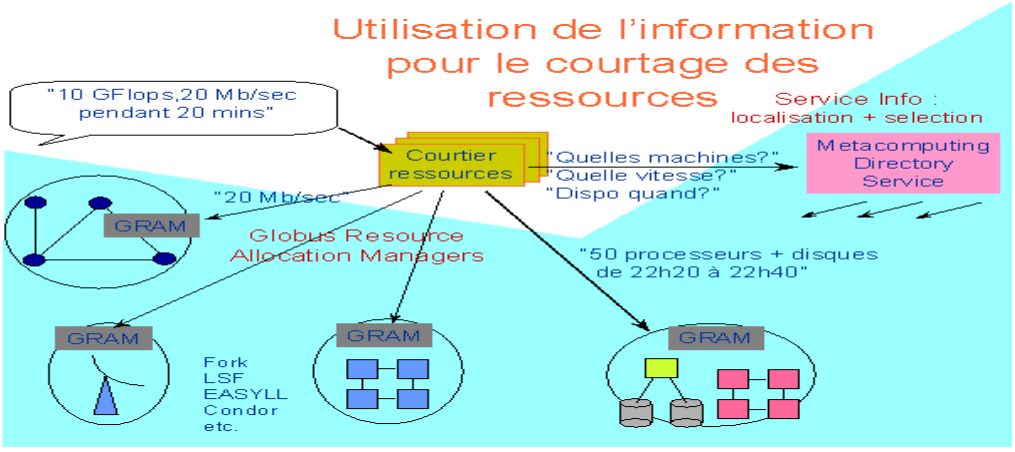


Figure 3.13: Système d'information et courtage des ressources.

Globus Toolkit offre le service MDS qui permet un accès efficace et réparti aux informations relatives aux types et à la disponibilité des ressources contrôlées par tous les sites du système.

Ces informations sont utilisées pour diverses raisons : localisation des ressources ayant une certaine caractéristique (tels que type de processeur, bande passante, performance, disponibilité, etc.), identification du GRAM associé à une ressource et enfin transformation d'une requête de haut niveau (resp. intermédiaire) en une requête intermédiaire (resp. de niveau 0). MDS utilise la structure de données et la librairie de routines (API) définies par le Lightweight Directory Access Protocol (LDAP) pour le

catalogage et la désignation des ressources. Alors, MDS est en mesure de répondre précisément aux questions suivantes quel est l'état général de la grille ? Quels sont les paramètres d'optimisation ? etc.

MDS permet d'optimiser les collectes, les échanges et les accès aux informations grâce à un mécanisme reposant sur trois services :

- GRIS (Grid Resoource Information Service) : des serveurs de GRIS sont chargés du services de description de ressource. Ils donnent des informations sur des ressources spécifiques.
- GIIS (Grid Index Information Service) : des serveurs de GIIS sont chargé du services d'annuaire. Ils consultent des collections de ressources, sur un ensemble de serveurs GRIS ;
- Referral Service (referrel : fréquence de consultation), relie les serveurs GRIS et/ou GIIS dans un seul espace de nom.

Dans la pratique Globus n'utilise souvent qu'un seul serveur GIIS pour optimiser les performances des réponses aux requêtes, car en fait l'environnement d'une organisation est relativement statique en termes d'information

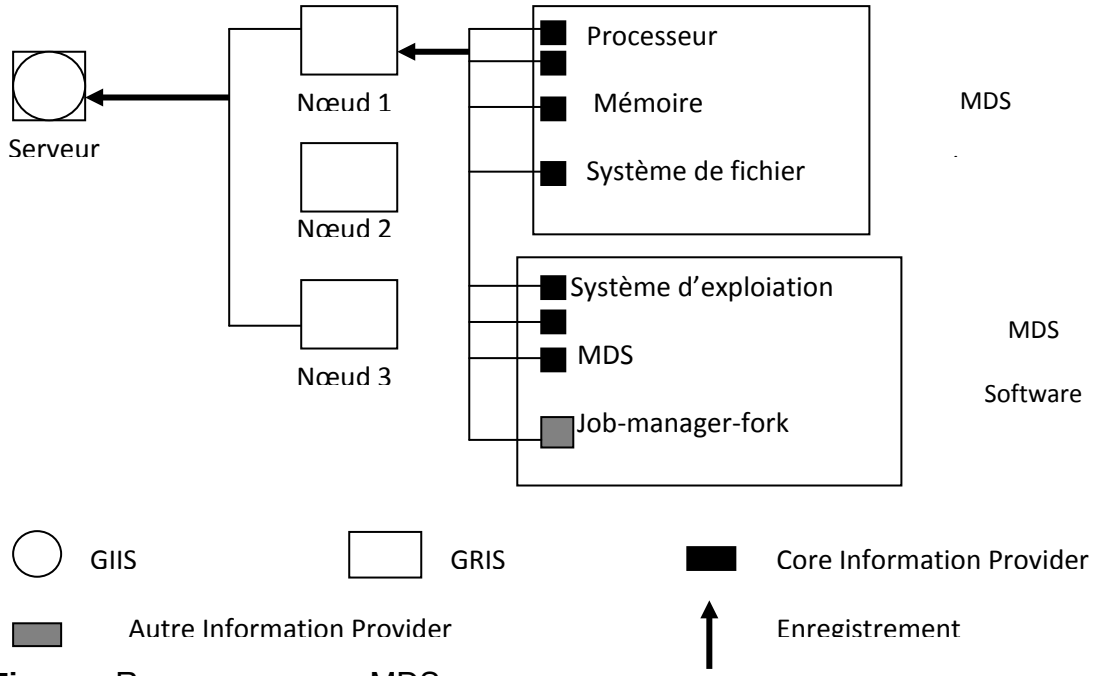


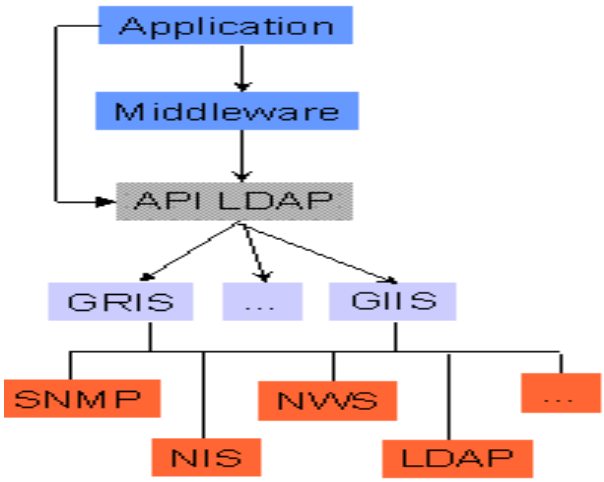
Figure : Ressources avec MDS

III- 2 .2 Approche Mds

MDS utilise le standard LDAP (« Lightweight Directory Access Protocol ») comme base pour la représentation et l'accès aux données. Le service MDS d'une grille construite avec Globus repose donc sur un ensemble de serveurs LDAP peuplant la grille.

Les informations sont stockées dans un annuaire de ressources distribué. MDS utilise la structure de données et la librairie de routines (API) définies par le Lightweight Directory Access Protocol (LDAP) pour le catalogage et la désignation des ressources. Plusieurs serveurs LDAP sont utilisés et sont optimisés par des fonctions particulières

- Mise à jour par :
  - Outils et fournisseurs d'informations
  - Applications (i.e., utilisateurs)
  - Outils générant l'info à la demande
- Informations disponibles dynamiquement par :
  - Outils
  - Applications
- Schéma spécifique à Globus
  - Représentation centrée sur l'hôte



III-2.3 Composants du MDS

MDS utilise plusieurs composants parmi lesquels on a les serveurs LDAP standards tel que OpenLDAP, Netscape, Oracle, ...

MDS permet d'optimiser les collectes, les échanges et les accès aux informations grâce à un mécanisme reposant sur trois classes de serveurs : GRIS, GIIS, Referral Service. Ces outils sont intégrés à Globus. Pour la manipulation du MDS, Il existe différents outils parmi lesquels la ligne de commande, les scripts et les GUIs

III-2.3-1 Lightweight Directory Access Protocol (LDAP

C'est la version allégée du protocole X.500 DAP. Il supporte des lectures/écritures distribuées, la réplication. C'est donc un protocole réseau pour l'accès au contenu de l'annuaire. C'est un standard de facto. Il définit un nommage définissant comment l'info est référencée et organisée. On distingue deux fonctions principales qui sont :

- Pages blanches :

  Recherche de l'adresse IP, de la quantité de mémoire, etc., associées à une machine particulière

- Pages jaunes :

  Trouver toutes les machines d'une classe particulière (machines SMP par exemple) ou ayant une propriété particulière (ex : plus de 256 Mo de mémoire)

Nous pouvons dire au point de vu structure que l'annuaire LDAP contient des classes d'objets dont leurs instance constituent son entrée. Ces entrées sont caractérisées par des attributs. Parmi les classes d'objets on peut citer :

- les ressources de calcul (systèmes d'exploitation, hiérarchie mémoire etc.)
- les interfaces réseaux (adresse IP, type d'interface, etc.)
- les performances (charge CPU, trafic réseau)
- les gestionnaires de ressources (contact, nœuds libres
- les logiciels (configuration, contrôle de version)
- les organisations, personnels

L'annuaire a une structure arborescente (Directory Information Tree). Les sous-arbres peuvent être distribués ou répliqués. La position d'un objet dans l'arbre est unique

(Distinguished Name : DN). On a donc une liste unique des noms et attributs depuis la racine jusqu'à l'objet

- <hn=popc.ens-lyon.fr, dc=ens-lyon, dc=fr, o=Grid>

Le nom d'un serveur LDAP est composé du nom d'hôte et du numéro de port avec une combinaison URL/LDAP sous la forme : ldap://<host>:<port>/DN

III-2.3-2 Recherche de serveurs :

Pour indexer le contenu d'un GRIS, un GIIS doit d'abord le trouver différentes options:

- Le GIIS peut être configuré à partir des noms des GRIS
- les GRIS s'enregistrent auprès du GIIS au démarrage. On dispose de plusieurs protocoles d'enregistrement supportés. (protocole « à la carte », le protocole de référence de LDAPv3 qui permet un ajout dynamique à l'arbre des références). Avec la possibilité de fédérations de serveurs d'infos, un GRIS peut être enregistré auprès de deux GIIS de sites différents.
- Le GIIS peut parcourir un arbre de références pour trouver les GRIS

Le MDS supporte ces 3 approches et la « bonne » dépend des besoins. Des combinaisons sont possibles.

Il existe plusieurs choix pour un utilisateur ou une application pour trouver un GIIS. On peut citer quelques uns :

- Connaissance du couple hôte/port
- Utilisation du serveur DNS
  - Renvoi du couple hôte/port pour un domaine donné
- Partie de l'arbre de références
  - Récursif : comment trouver l'arbre de références
- Indexation par un autre GIIS

III-2-4 Insuffisances du MDS :

L'augmentation du volume de l'information et du volume des transactions peut poser des problèmes de mises à jour régulières. De même une panne du serveur central entraine une indisponibilité de l'information pour les clients qui sollicitent le GIIS.

Ainsi, face à ces problèmes, le besoin de systèmes qui résistent à la montée en charge et qui fournissent un bon temps de réponse aux requêtes et transactions sur des gros volumes de données devient une nécessité.

C'est ainsi que certains équipes de recherche ont essayé de mettre en place des solutions distribuées de ce système d'information. Dans le paragraphe qui suit nous essayons de présenter quelques solutions

III.3 PRINCIPALES ARCHITECTURES DISTRIBUEES

Nous distinguons deux types d'architectures :

- Les architectures structurées qui implémentent en général une Table de Hachage Distribuée (DHT). Elles associent la localisation des informations a la topologie du réseau et sont particulièrement adaptées pour retrouver des informations peu répliquées. Cependant nous pouvons noter quelques inconvénients qui sont :
    - le coût élevé pour répondre à des requêtes approchées ou portant sur un intervalle
    - la connaissance complète à priori de la clé associée à une recherche.
- Les architectures non structurées n'imposent aucune contrainte entre la localisation des données et la topologie du réseau. Elles sont particulièrement adaptées pour retrouver de l'information ayant un grand nombre de copies. Avec ces architectures on a de faibles contraintes imposées sur la topologie virtuelle. Ces systèmes sont particulièrement adaptés aux environnements très dynamiques

III.3.1 Table de hachage distribué Chord :

Introduites en 2001, les tables de hac hage distribuées (DHT) [23] sont rapidement devenues incontournables tant au ni veau scientifique que dans les principales applications pair-à-pair (peer-to-peer). Ainsi, le très fort engouement académique pour ces systèmes est symbolisé par le nombre de citations scientifiques vers la publication séminale : l'article décrivant le système Chord [24] a ainsi été cité par plus de 4. 000 publications selon http://scholar.google.com. Cela en f ait un des articles les plus marquants de la dernière décennie.

Il s'agit d'un mécanisme distribué et décentralisé permettant d'associer des valeurs de clés à un c ontenu par hachage de l a clé où chaque participant gère une par tie de l a table de hachage.

III.3.1.1 Algorithme de base

Les clés et les pairs (adresse IP) sont hachés sur le même anneau (cercle). La fonction de hachage est sur m bits. Les identifiants sont obtenus de la maière suivante :

Identificateur nœud = hachage de son adresse IP

Identificateur clé = hachage de la clé

Les clés et pairs sont affectés à un identifiant dans [0..2m-1] (au plus 2m pairs).

Succ(kid) : le plus petit identifiant de pair qui soit supérieur ou égal à k mod 2m

Pred(kid) : le plus grand identifiant de pair inférieur à k mod 2m

Chaque clé d'identifiant k est gérée par le pair suivant ou égal , i.e., d'identifiant supérieur ou égal

Pred(Succ(kid)) < kid ≤ Succ(kid)

Les identificateurs sont ordonnés modulo 2m (ordre circulaire)

Correspondance entre nœud et clé : la clé k est associée au premier nœud dont l'identificateur est égal ou supérieur à celui de la clé k

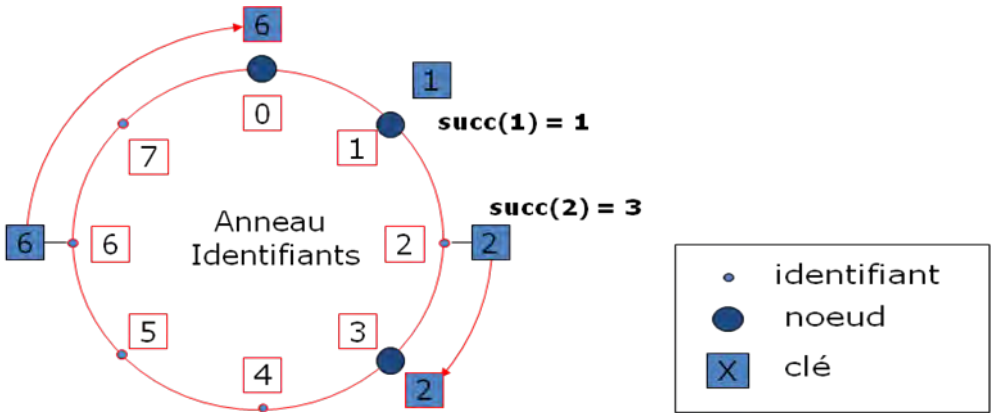


Figure 4.10

Avec cette architecture la recherche est en O(N) d'où une amélioration qui consiste à utiliser un table de routage appelée « Finger » au niveau de chaque pair contenant les adresses IP d'au plus m voisins.

III.3.1.2 Amélioration de l'Algorithme de base

Chaque nœud a une table des pointeurs (finger) d'au plus m voisins

Pour chaque noeud i :

- Succ[k]=finger[k] = premier nœud sur l'anneau qui vérifie (i + 2k-1) mod 2m, 1<= k <= m
- Successeur=succ[1]
- m entrées dans la table

Chaque nœud connait ses successeurs (N° et adresse IP) en puissances de 2 successives sur l'anneau

Il envoie ainsi la recherche au moins à moitié chemin de la distance restant à parcourir sur l'anneau pour trouver la clé

La recherche s'effectue ainsi par dichotomie. Le temps de recherche est en O(logN).

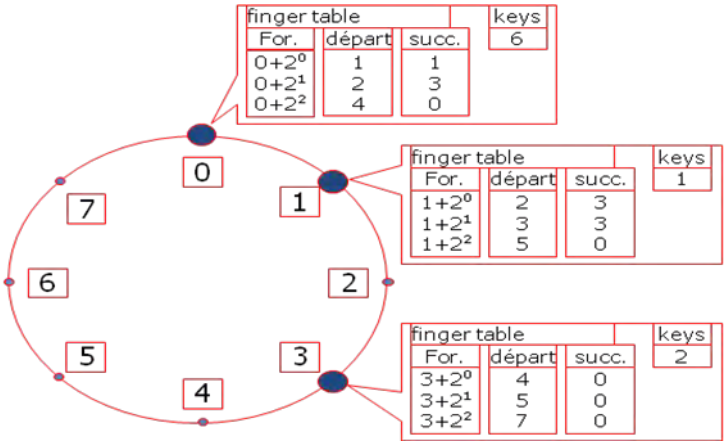


Figure 4.11

Dans l'exemple ci-dessous où M=6 (N0 à N63) le chemin suivi pour lookup(54) issue de N8 est montré par la figure 12

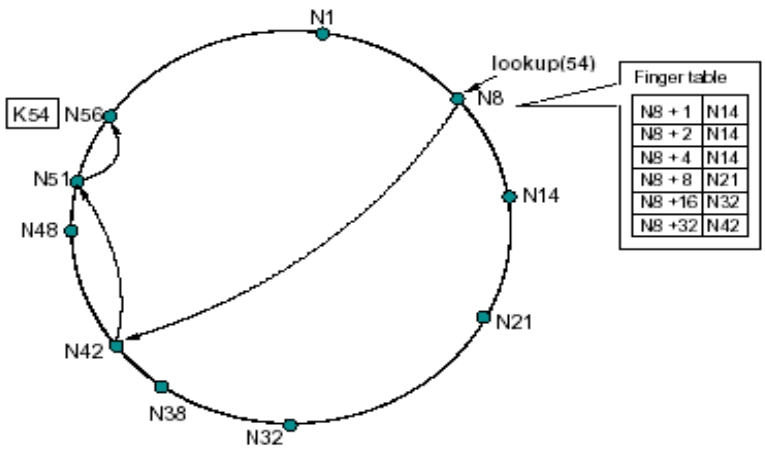


Figure 4.12

III.3.1.3 Exemple de MDS s'inspirant de Chord

III.3.1.3.1 SAMGrid : recherche d'informations (fichiers)

Repose sur la topologie Chord. En plus de l'anneau dispose d'un Serveur Central. Les IS (service d'information) qui reçoivent les requêtes des nœuds pour les envoyer au serveur central forment l'anneau. Chaque nœud a son IS

Les informations comme les fichiers sont stockées dans l'anneau. Quand un nœud a besoin d'un fichier, l'IS consulte d'abord l'anneau Chord. Si le fichier n'existe pas dans l'anneau la requête se poursuit au niveau du serveur central. Les mises à jour sont envoyées à l'anneau au serveur central

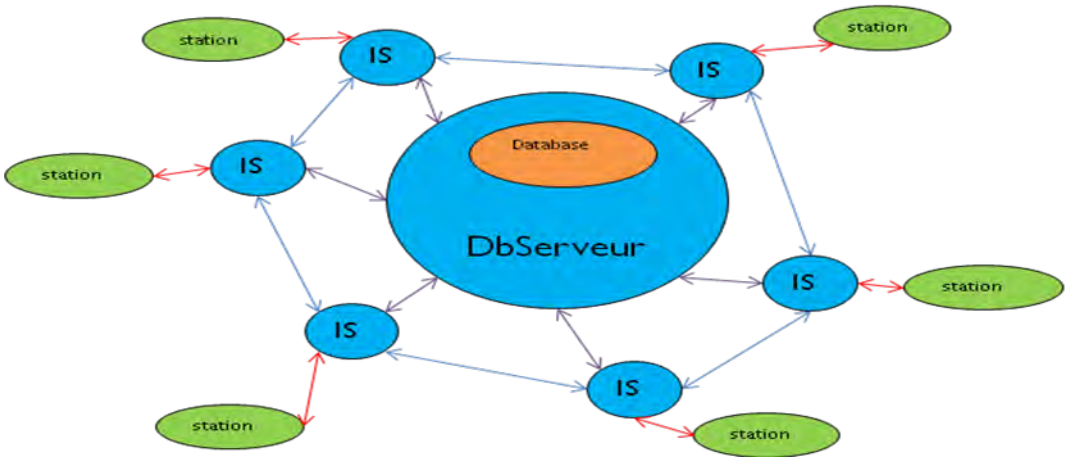
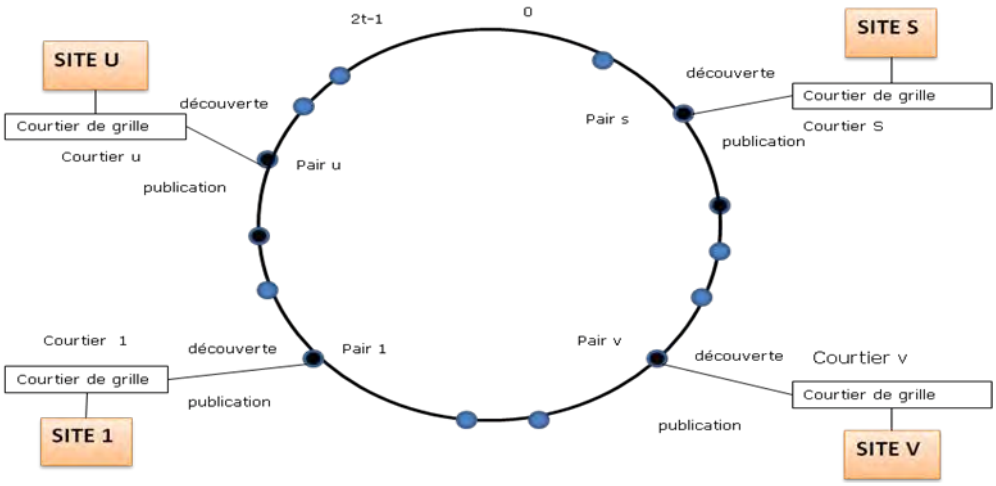


Figure 4.17

III.3.1.3.2 SERVICE DE DECOUVERTE DE RESSOURCES DISTRIBUE

Service mis en place par Rajiv Ranjan, Lipo Chan, Aaron Harwood, Rajkumar Buyya, Shanika Karunasekera du laboratoire de réseaux P2P et de grille du département d'informatique de l'université de Melbourne, Victoria, Australie.



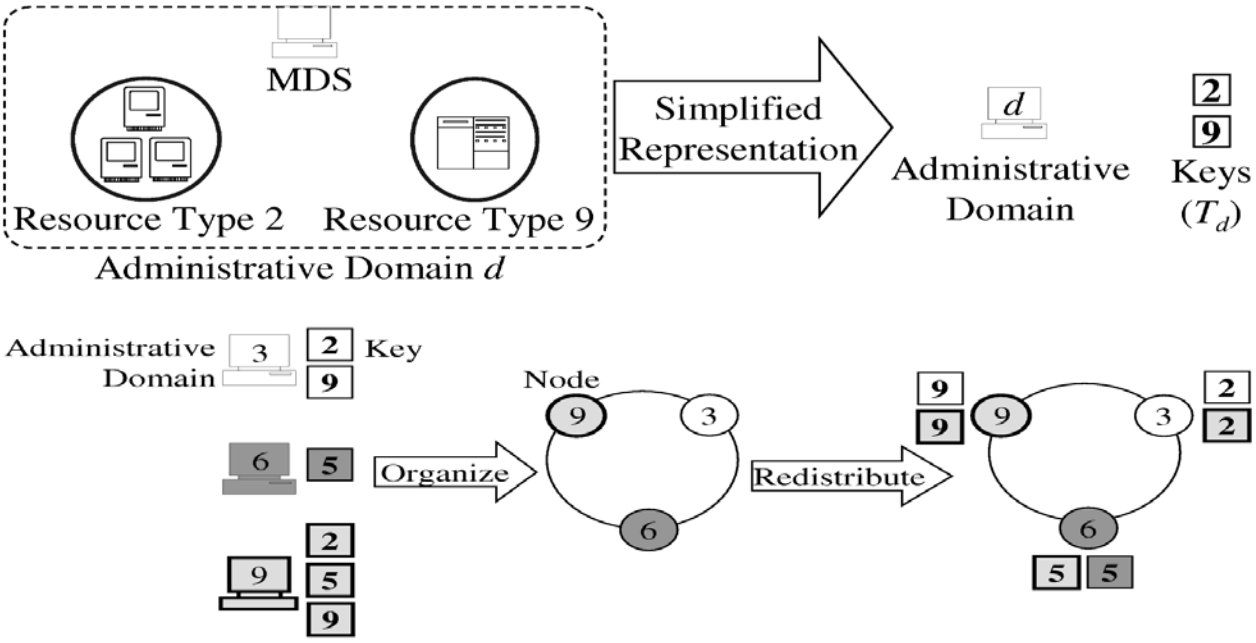
L'anneau Chord est formé des pairs (points noir) et des informations envoyées par les sites (points bleus).Chaque site est lié à un pair par le « courtier de grille »

Utilisation d'une table de hachage distribué fonction dans l'espace. Le service organise les données en maintenant un index de dimension d (dans l'espace). On dispose de deux types de requêtes (Localisation et Mise à jour).

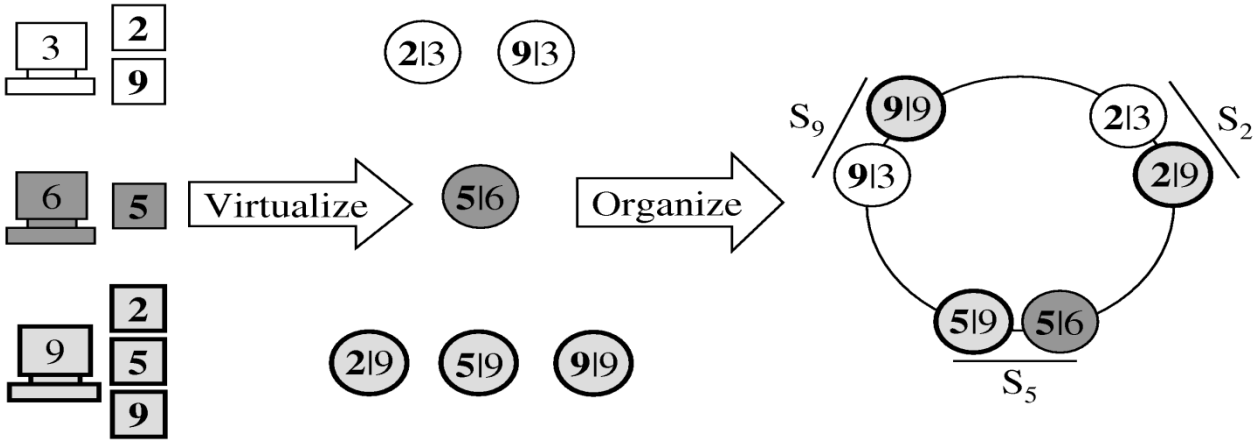
III.3.1.3.3 DGRID

Il s'agit d'une autre approche qui utilise une DHT. Chaque domaine administratif d stocke ses propres données

- Principe :
  - Grille=ensemble de N domaines d et K types de ressources. Un type de ressources étant un tuple t d'un ou de plusieurs attributs décrivant une ressource tel que t= {cpu="P4", mémoire="1 GB"}
  - Chaque domaine d à un ensemble T_d de types de ressources où chaque t $\in T_d$ est une instance de plusieurs ressources
  - Chaque type de ressource appartenant à un domaine a son propre identifiant qui est stocké sur un nœud index (qui a aussi son identifiant) qui contient l'ensemble des clés des ressources du domaine Td



Les différents domaines (identifiants) sont placés sur l'anneau et la distribution des clés suit le même principe que Chord.



Chaque domaine d est représenté au niveau de l'anneau logique par |td| nœud. L'identifiant d'un nœud $n_{t,d}$ est t|d.

Performance :

- Recherche : latence faible
- Message : beaucoup de bande passante. L'utilisation du concept de marches aléatoires permet une amélioration, mais surtout l'ensemble du réseau doit être étudiée
- Coût de stockage: faible
- Mise à jour et coûts d'entretien élevés

- Bonne résistance aux pannes:

Inconvénients :

- Recherche: basée sur des valeurs,
- Connaissance globale requise
- Grande autonomie des pairs

II.3 ETUDE COMPARATIVE

	TYPE DE RECHERCHE	TYPE DE RESEAU	GESTION DES RESSOURCES	NOMBRE DE MESSAGES	INCONVENIENT
SamGrid	Entité dont l'ID est connu	• Structuré • Anneau Chord	Stockage sur l'anneau en fonction de la clé et au niveau du dbserveur sous forme de liste	O(Log n)	• Complexité de m ise en œ uvre grande et recherche basée sur des valeurs • Maintenance requiert une gr ande bande passante • Pas adaptée si le nombre réplications augmente
SDRD	Entité dont l'ID est connu	• Structuré • Anneau Chord	Stockage sur l'anneau en fonction de la clé sous forme de liste	O(log N)	• Complexité de M ise en œ uvre grande et recherche basée sur des valeurs • Maintenance requiert une gr ande bande passante • Pas adaptée si le nombre réplications augmente
DGRID	Entité dont l'ID est connu	• tructuré • Anneau Chord	Stockage sur l'anneau en fonction de la clé sous forme de liste	O(log N)	• Complexité de M ise en œ uvre grande et recherche basée sur des valeurs • Maintenance requiert une gr ande bande passante • Pas adaptée si le nombre réplications augmente

Le tableau ci-dessus montre que **Chord** n'utilise pas beaucoup de bande passante, les recherche sont binaire basée sur des valeurs. Cependant il s'agit d'un réseau structuré. Gnutella par contre est un réseau non structuré de même que Freenet mais le nombre de messages générés par Gnutella est très élevé.

Dans le cas de notre étude il s'agit de réseau non structuré. Notre solution va s'inspiré de Gnutella (type de recherche identique) mais en essayant de réduire l'utilisation de la bande passant .

Chapitre 4

NOTRE METHODE DISTRIBUE ET DE RECHERCHE DE RESSOURCES BASE SUR LES ARBRES AVL

IV.1 INTRODUCTION

Si les réseaux structurés sont tout à fait adaptés à la recherche d'entités dont le nom est connu, ils le sont beaucoup moins lorsque le nom n'est pas connu et que ces entités doivent être recherchées en fonction de certains critères.

Certains projets (**Samgrid**,..) tentent d'utiliser un réseau structuré pour des recherches fondées sur un ensemble de critères mais la complexité de mise en œuvre est grande et la maintenance requiert une bande passante importante.

En revanche, les réseaux non structurés sont naturellement adaptés aux recherches d'entités fondées sur un ensemble de critères, dont le nombre évolue dans le temps et dont la valeur peut être définie sur un intervalle.

Notre approche du concept de service MDS distribuée utilise au niveau de chaque site une structure d'arbre AVL pour l'enregistrement des ressources locales.

Les arbres AVL sont intéressants parce que ce sont des arbres de recherches équilibrés qui permettent d'optimiser le temps d'accès à une information.

Pour le MDS, Nous proposons une architecture largement distribuée appelée DGIS (Distributed Grid Information Service).

Chaque site a son propre GIIS. Une telle structure ne nécessite pas de serveur central, un utilisateur arrivant sur le système peut se connecter à n'importe quel nœud du réseau pour en faire partie à son tour.

IV.2 RAPPEL SUR LES ARBRE

IV.2.1 ARBRE BINAIRES

Définition 1 : Un **Arbre Binaire** est un arbre où chaque nœud admet au plus 2 fils.

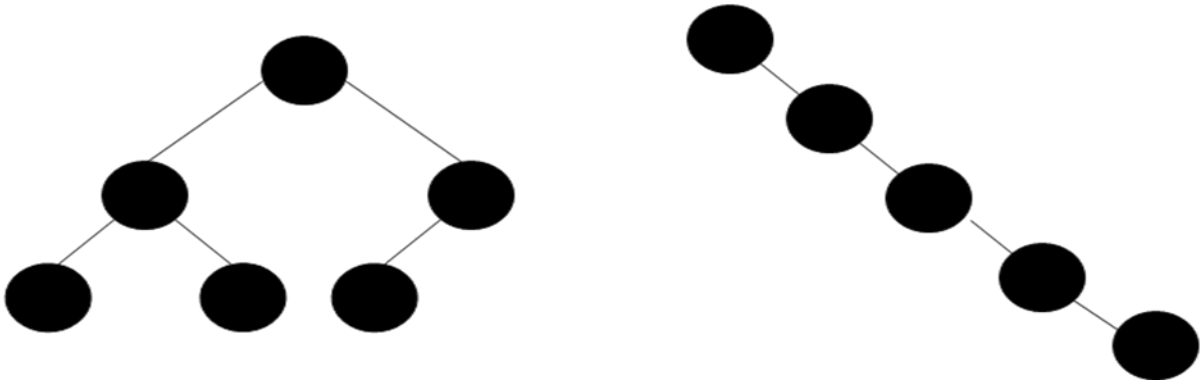


Figure 3.1 a et 3.1b : arbres binaires

Définition 2 : (Arbre binaire de recherche). Un arbre binaire est de recherche lorsque, si x est un nœud de l'arbre, et y un nœud du sous-arbre gauche (resp. droit) de x, on a y < x (resp. x < y).

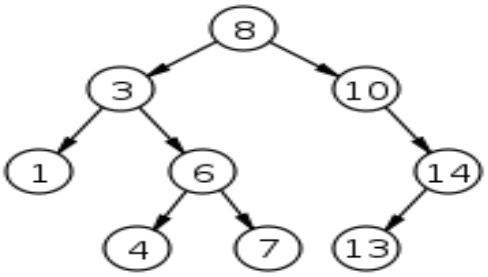


Figure 3.2 : arbre binaire de recherche

IV -2.2 ARBRES AVL

IV.2.2.1 DEFINTIONS ET PROPRIETES

Introduits pour la première fois par Adel'son-Vel'skiˇı et Landis en 1962.

Définition. : Un arbre binaire de recherche est un arbre AVL si, pour n'importe lequel de ses nœuds, la différence de hauteur entre ses deux fils diffère d'au plus un.

- ils "mémorisent" une dichotomie de rangement dans la structure de l'arbre.

=> Recherche en log n (n nombre de nœuds)

- **Ils possèdent les Propriétés**
  - Le **facteur d'équilibrage** d'un nœud est la différence entre la hauteur de son sous-arbre droit et celle de son sous-arbre gauche.

- Un nœud dont le facteur d'équilibrage est 1, 0, ou -1 est considéré comme équilibré.
- Un nœud avec tout autre facteur est considéré comme déséquilibré et requiert un rééquilibrage.
- Le facteur d'équilibrage est soit déduit des hauteurs des sous arbres, soit stocké dans chaque nœud de l'arbre (ce qui permet un gain de place, ce facteur pouvant être stocké sur deux bits, mais complexifie les opérations d'insertion et de suppression).

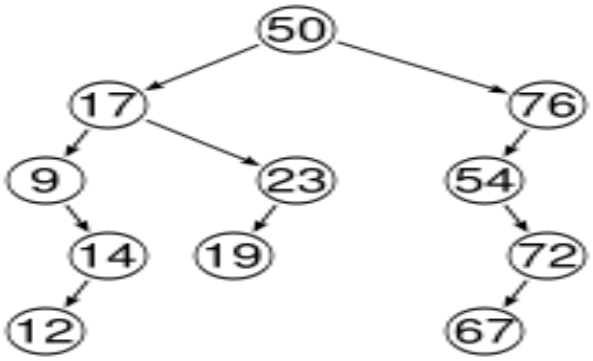


Figure 3.3 : Un exemple d'arbre **non-AVL**.

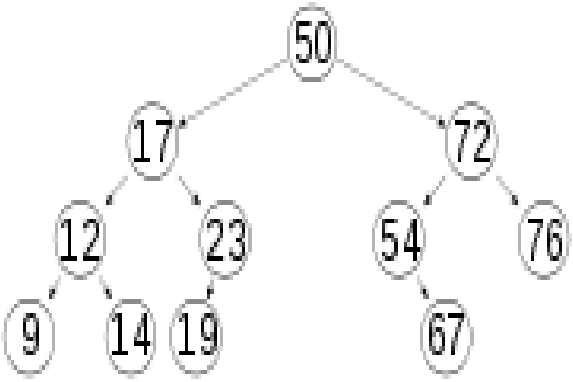


Figure 3.4 : Le même arbre après un rééquilibrage.

IV.2.2.2 INSERTION

L'arbre devient déséquilibré si l'élément ajouté est le descendant gauche (droit) d'un nœud avec un léger déséquilibre gauche (droit). Alors la hauteur de ce sous-arbre augmente. Le déséquilibrage peut donc être provoqué par l'ajout ou la *suppression* d'un *nœud.*

Noter que l'insertion d'un nœud peut provoquer des déséquilibres sur plusieurs nœuds.

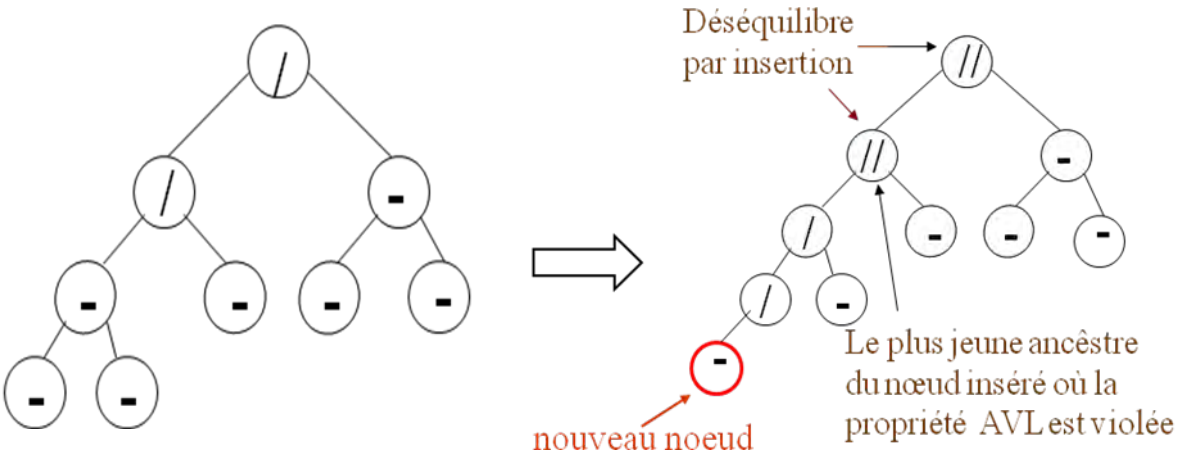


Figure 3.4 : Insertion nœud.

Supposons que le sous-arbre le plus haut est celui de **gauche** et qu'un nœud est inséré pour augmenter la hauteur de ce sous-arbre. L'arbre obtenu est déséquilibré

On peut rétablir un arbre AVL en utilisant des **rotations**

=> Soit **A** le plus jeune ancêtre où apparaît le déséquilibre

Dans l'arbre AVL, avant l'insertion, T_1, T_2 et T_3 ont une hauteur h.

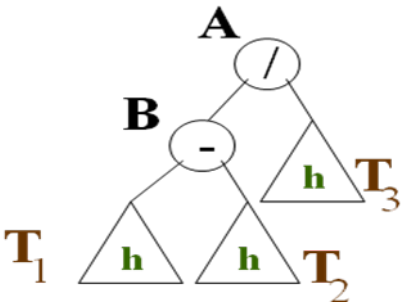


Figure 3.5 : arbre équilibré.

Le même raisonnement peut être utilisé si l'arbre le plus haut est celui de **droite**

Cas I : un nouveau nœud est inséré dans T_1

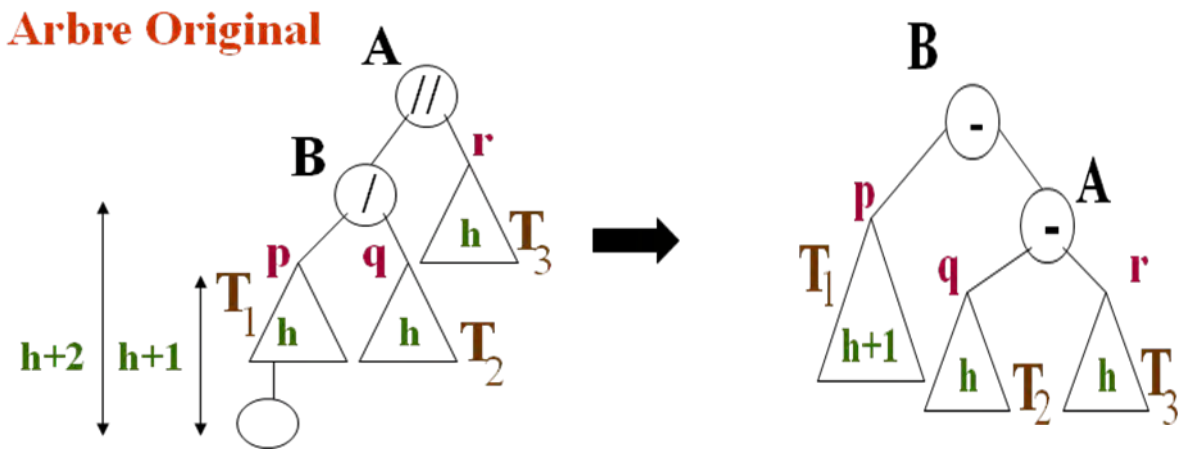


Figure 3.6 : rééquilibrage.

Rééquilibrer par **rotation droite:** P < B < q < A < r => propriété ABR maintenue!

On peut utiliser les algorithmes rotation Droite ou rotation Gauche suivants :

void RD(Arbre *a){

*Arbre aux= (*a)->fg;*

*(*a)->fg = aux->fd;*

*aux->fd= *a;*

**a= aux;*

}

void RG(Arbre *a){

*Arbre aux= (*a)->fd;*

*(*a)->fd = aux->fg;*

*aux->fg= *a;*

**a= aux;*

}

Cas II : nouveau nœud inséré dans T_2

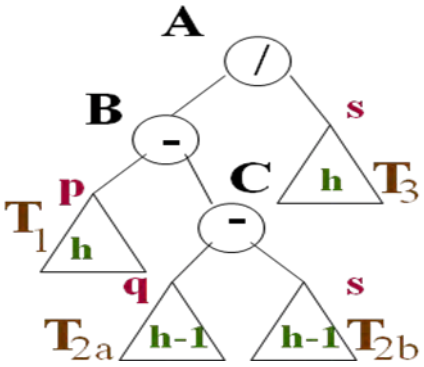


Figure 3.7 : Arbre binaire.

On a 3 cas à considérer à savoir un nouveau nœud en C, en T2a ou T2b. Les 3 cas sont similaires. On considérera le cas 2.

Cas II - T2a : Rééquilibrage de l'arbre AVL avec une double rotation (gauche sur B et ensuite droite sur A)

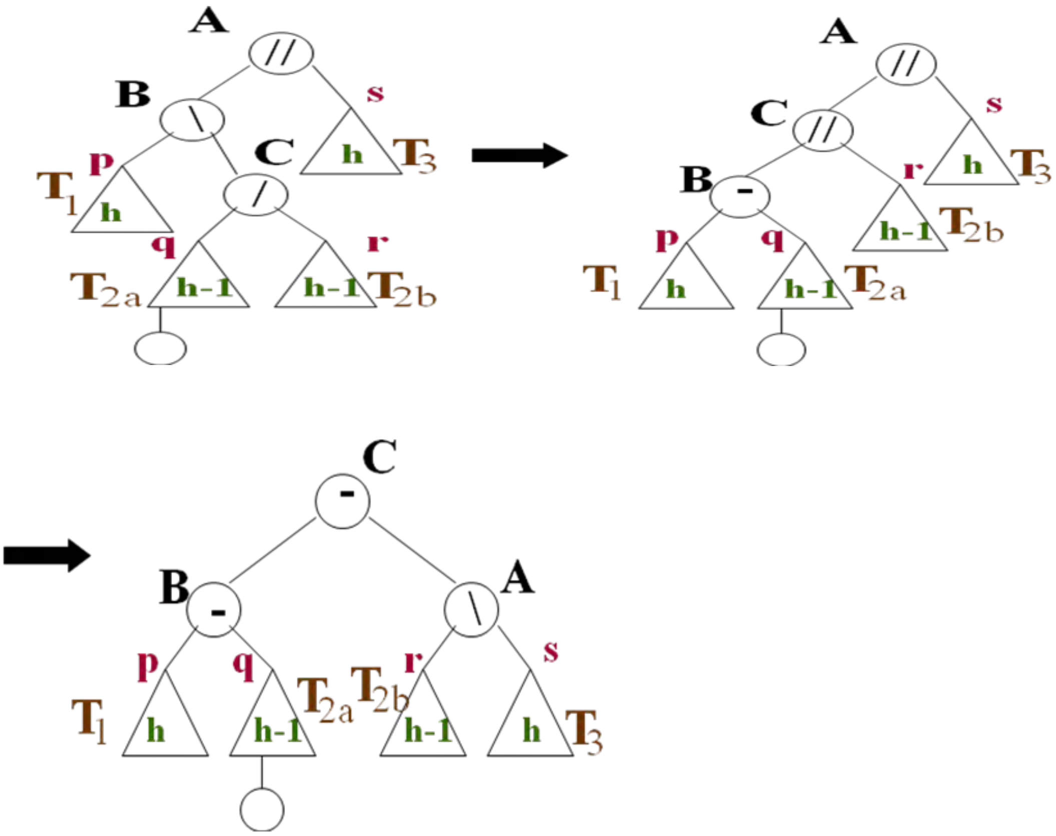


Figure 3.8 : ajout et rééquilibrage.

Voici l'algorithme correspondant :

```
void RGD(Arbre *a){
  RG( &((*a)->fg) );
  RD(a);
}
```

Cas II - T2b : Insertion en T2b => rotation droite sur B et ensuite gauche sur A. L'algorithme est le suivant :

```
void RDG(Arbre *a){
  RD( &((*a)->fd) );
  RG(a);
```

}

Nous avons défini un arbre **"équilibré"** et nous avons aussi montré comment insérer dans l'arbre en ut ilisant les algorithmes ABR de manière à maintenir l'équilibre (propriétés AVL) et la propriété ABR.

On appliquer le même principe que celui qui vient d 'être appliqué dans le cas d'une suppression entrainant un déséquilibre.

IV.2.2.3 Performances pour les arbres AVL

Rebalancer un nœud prend un temps O(1), si on implémente l'arbre AVL à l'aide d'une structure chaînée.

L'algorithme de recherche prend un temps O(log n)

- La hauteur de l'arbre est O(log n) et aucune restructuration est nécessaire

L'algorithme d'insertion prend un temps O(log n)

- Chercher l'endroit où insérer prend un temps O(log n)
- Trouver un nœud non balancé (si il y en a un) prend un temps O(log n)
- Restructurer l'arbre prend un temps O(1)

L'algorithme de suppression prend un temps O(log n)

- Chercher le nœud à enlever prend un temps O(log n)
- On devra faire, au plus, O(log n) restructuration
- Chaque restructuration prend un temps O(1)

IV.3 STRUCTURE DE LA GRILLE

Notre proposition contient trois composants essentiellement qui sont :

- Largeur de Bande et Latences
- CE : élément de calcul
    - caractérisé par sa marque, vitesse, os, multi-cœur, état (occupé, libre, … etc.)
- SE : élément de stockage
    - caractérisé par sa marque, capacité, état (taux d'occupation, …)

Chaque site peut être identifié par un GIIS (son adresse IP par exemple)

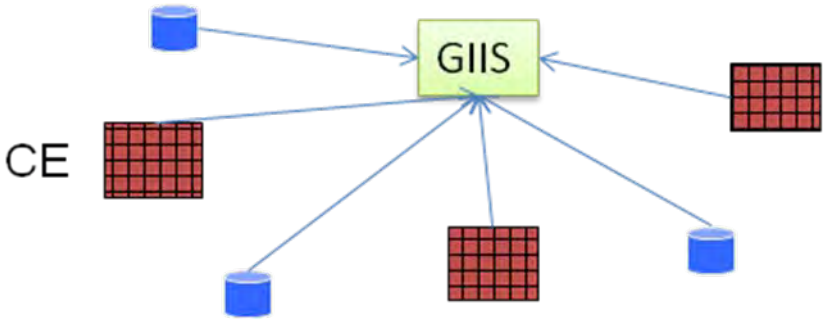


Figure 3.9 : composantes d'un site.

Nous gérons deux types de nœuds au niveau du site :

- les nœuds simples (feuille)
- les super-nœuds (GIIS) qui jouent le rôle de serveur d'annuaire au groupe et propage les requêtes aux autres têtes de groupe. Tous les nœuds simples (feuilles) d'un site s'enregistrer au niveau du GIIS local.

IV.4 NOTRE APPROCHE DE DISTRIBUTION DE L'ANNUAIRE

On part d'un MDS centralisé que nous distribuons en effectuant une répartition horizontale. Les ressources d'un site sont enregistrées au niveau d'un GIIS local. Chaque GIIS est responsable de la publication des informations de chaque nœud du site dans lequel il se trouve. Ces annonces peuvent être utilisées par tous les clients de la VO.

Pour maintenir l'indexe à jour il faut périodiquement actualiser les GIIS. Pour cela les annonces doivent avoir une durée de vie paramétrable.

Ressource	Type	Capacité ou vitesse	Site	AUTRES INFORMATIONS
PROC1	CE	0,5	S2	
PROC2	CE	1	S2	
C	SE	180	S1	ATA DISK drive
HDB	SE	80	S2	

PROC2	CE	2	S3	
D	SE	100	S1	
C	SE	160	S4	
E	SE	250	S1	
D	SE	50	S4	Maxtor 51024U2
PROC4	CE	1,5	S3	
hdb	SE	130	S2	ATA DISK drive
PROC5	CE	4	S2	
PROC6	CE	3	S2	

Tableau 1 : MD centralisé

Après répartition nous obtenons les MDS suivants

S1			
RESSOURCE	**TYPE**	**CAPACITE/ VITESSE**	**AUTRES INFORMATIONS**
C	SE	180	
D	SE	100	
E	SE	250	

S2			
RESSOURCE	**TYPE**	**CAPACITE/ VITESSE**	**AUTRES INFORMATIONS**
PROC1	CE	0,5	

PROC2	CE	1	
HDA	SE	80	
HDB	SE	130	ATA DISK drive
PROC5	CE	3	

S3			
RESSOURCE	**TYPE**	**CAPACITE/VITESSE**	**AUTRES INFORMATIONS**
PROC2	CE	2	
PROC4	CE	1,5	

S4			
RESSOURCE	**TYPE**	**CAPACITE/ VITESSE**	**AUTRES INFORMATIONS**
C	SE	160	Maxtor 51024U2
D	SE	50	

Comme déjà annoncé dans l'introduction nous utilisons les arbres AVL pour représenter les annuaires locaux. Ainsi nos différends MDS locaux sont représentés par les arbres ci-dessous :

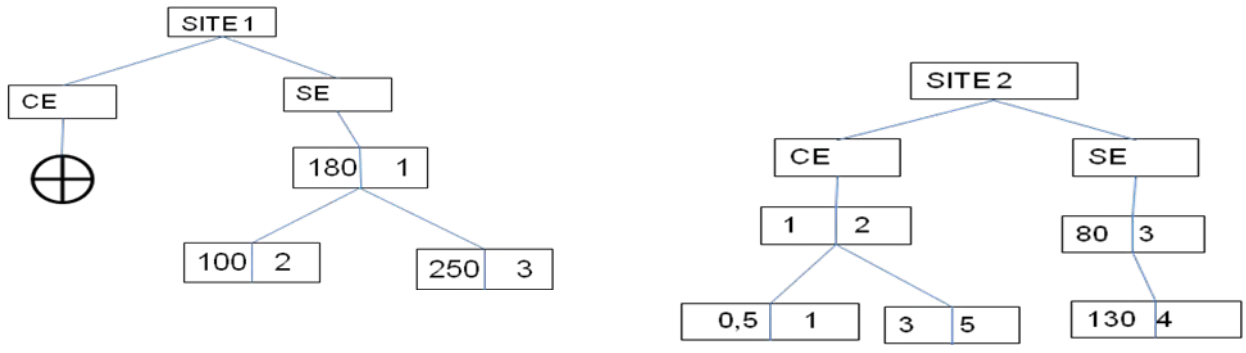
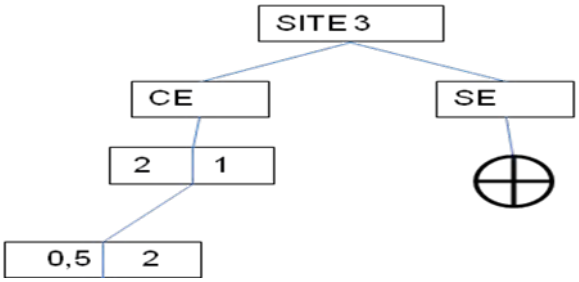


Figure 3.10a : arbre local du site 1

Figure 3.10b : arbre local du site 2



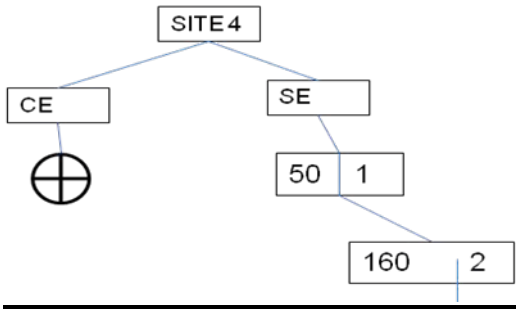


Figure 3.10c: arbre local du site 3

Figure 3.10d: arbre local du site 4

Notre grille a donc une topologie représentée dans la figure 3.11a. Chaque site gère son annuaire local.

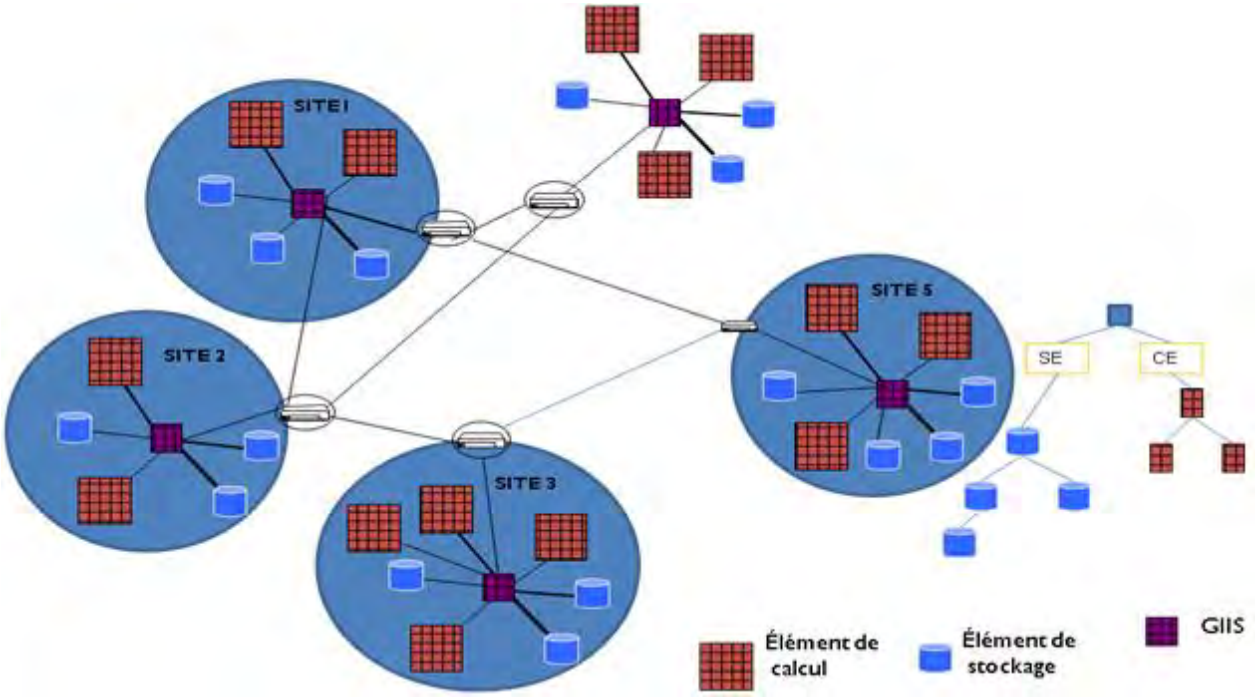


Figure 3.11 : architecture de la grille

La topologie de notre grille est une topologie hiérarchique à deux niveaux, comme illustré sur la figure 3.11. Les clients sur la première rangée sont connus en tant que GIIS (supernoeuds) et les clients sur la deuxième rangée (feuilles) sont connus comme étant des nœuds simples. Tous les nœuds d'un site sont reliés au GIIS.

Les GIIS ont une durée de connexion plus stable dans le temps. Du fait de leur rôle, ces GIIS ont aussi des responsabilités plus importantes dans la grille. Chaque nœud

simple a un parent GIIS, qu'il choisit quand il se connecte à la grille. Il envoie les métadonnées des ressources qu'il partage. Les GIIS possèdent donc toutes les informations d'indexation des nœuds dont ils sont responsables, et peuvent répondre aux requêtes qui concernent tous ces nœuds.

Pour que les requêtes puissent être envoyées dans tout le réseau, les GIIS maintiennent également des connexions avec d'autres GIIS.

Notre solution gère trois types d'opérations qui sont L'ajout (nœud ou site), la suppression (nœuds ou site) et la recherche de ressources.

IV.5 L'AJOUT (NŒUD OU SITE)

Pour insérer une ressource au niveau d'un site, on commence par exécuter l'algorithme d'insertion d'un arbre binaire de recherche.

On commence par chercher la ressource dans l'arbre. Si elle n'est pas dans l'arbre l'algorithme la recherche se terminera dans une feuille w. On insère la ressource dans w et on change w en un nœud interne.
Si la ressource est dans l'arbre l'algorithme la recherche se terminera dans un nœud interne w. On applique alors récursivement l'algorithme de recherche sur le fils Droit de w, jusqu'à ce qu'on trouve une feuille. Après une insertion, si l'arbre est un AVL alors on ne fait rien sinon il faut le rééquilibrer.

S1			
RESSOURCE	TYPE	CAPACITE/ VITESSE	AUTRES INFORMATIONS
C	SE	180	
D	SE	100	
E	SE	250	
C	SE	80	
F	SE	50	

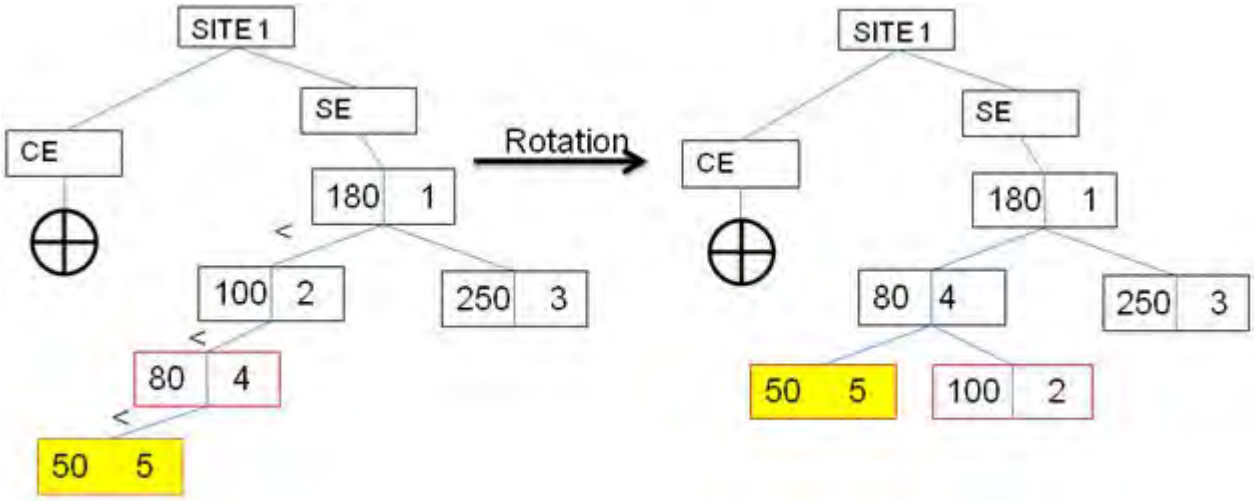


Figure 3.12 : Ajout de nœud

IV.7 LA SUPPRESSION (NŒUDS OU SITE)

La suppression d'un nœud feuille est identique à la suppression dans les arbre binaire. Toutefois il faut rééquilibrer l'arbre en cas de déséquilibre

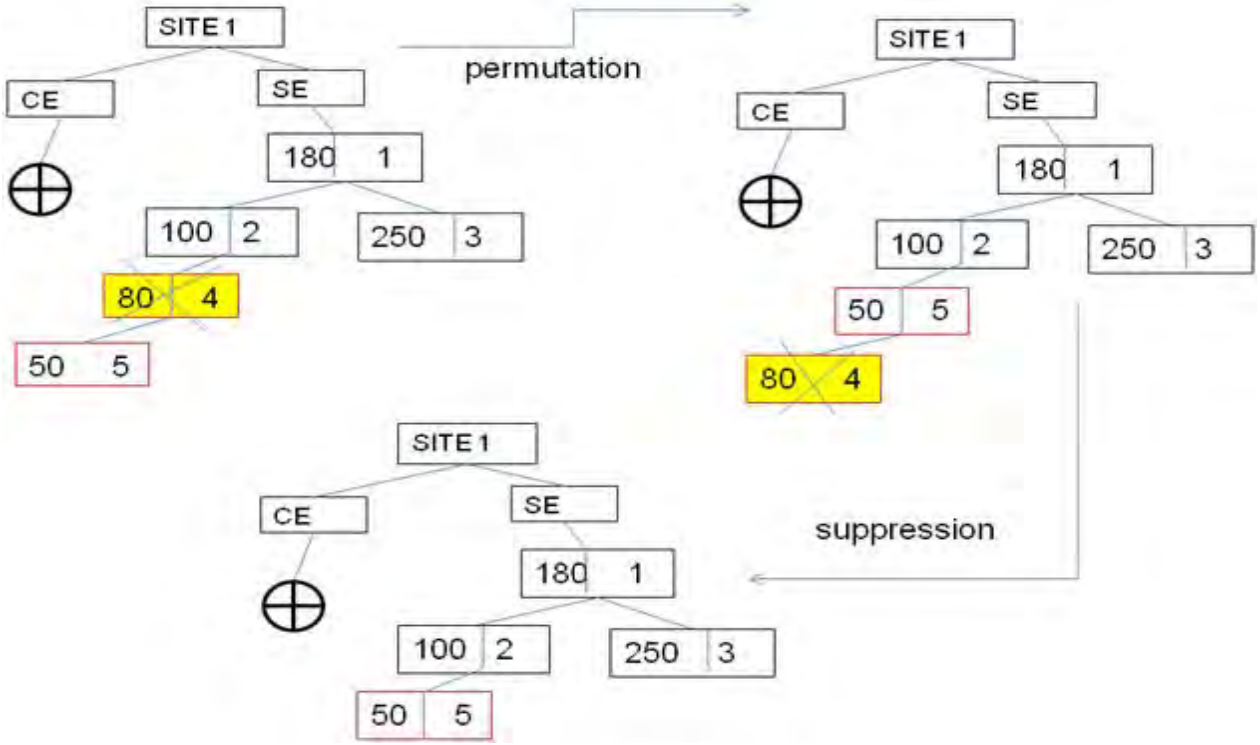


Figure 3.13: **Suppression d'un nœud avec un fils**

La suppression d'un site entraine la suppression de l'annuaire local et de tous les nœuds

IV.8 LA RECHERCHE DE RESSOURCES

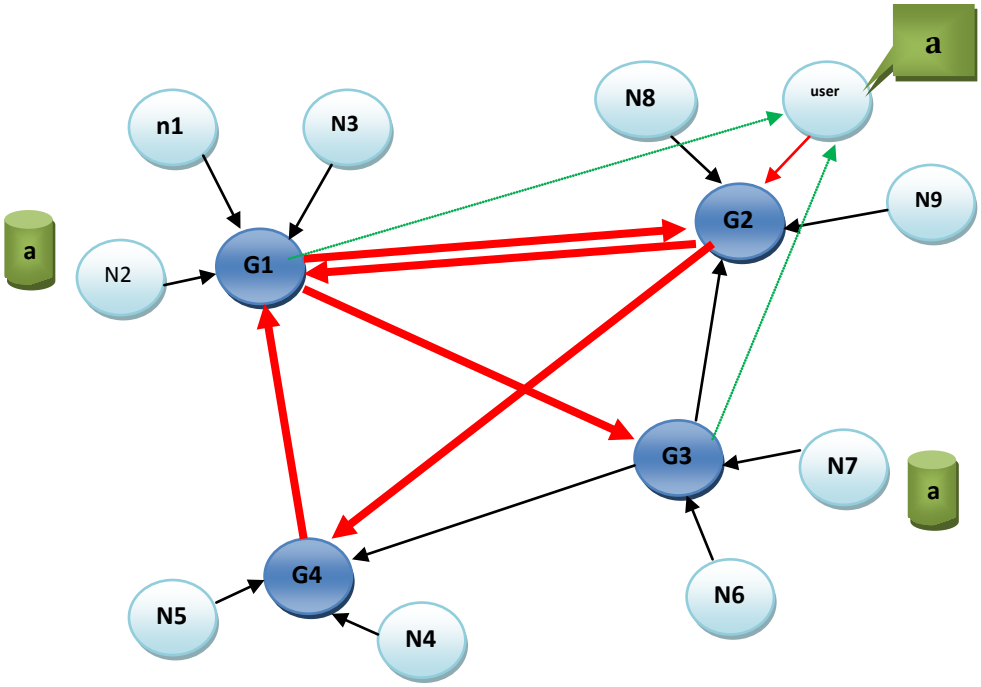
Nous pouvons distinguer deux types de requêtes :

- Demande de ressource de calcul CRQ
- Demande de ressource de stockage SRQ

Nous pouvons avoir une combinaison des deux. Toute recherche commence par une identification du type de la requête.

Pour la recherche nous nous distinguons les nœuds feuilles et les super-nœuds :

- Super-nœuds (GIIS)
  - Prennent en charge les recherches
  - Conserve la liste des ressources d'un ensemble de nœuds (annuaire local).
- Nœuds feuilles
  - Dépendent du super-nœud (ici GIIS) pour la recherche



Quand le nœud est trouvé la réponse est envoyée directement au client.

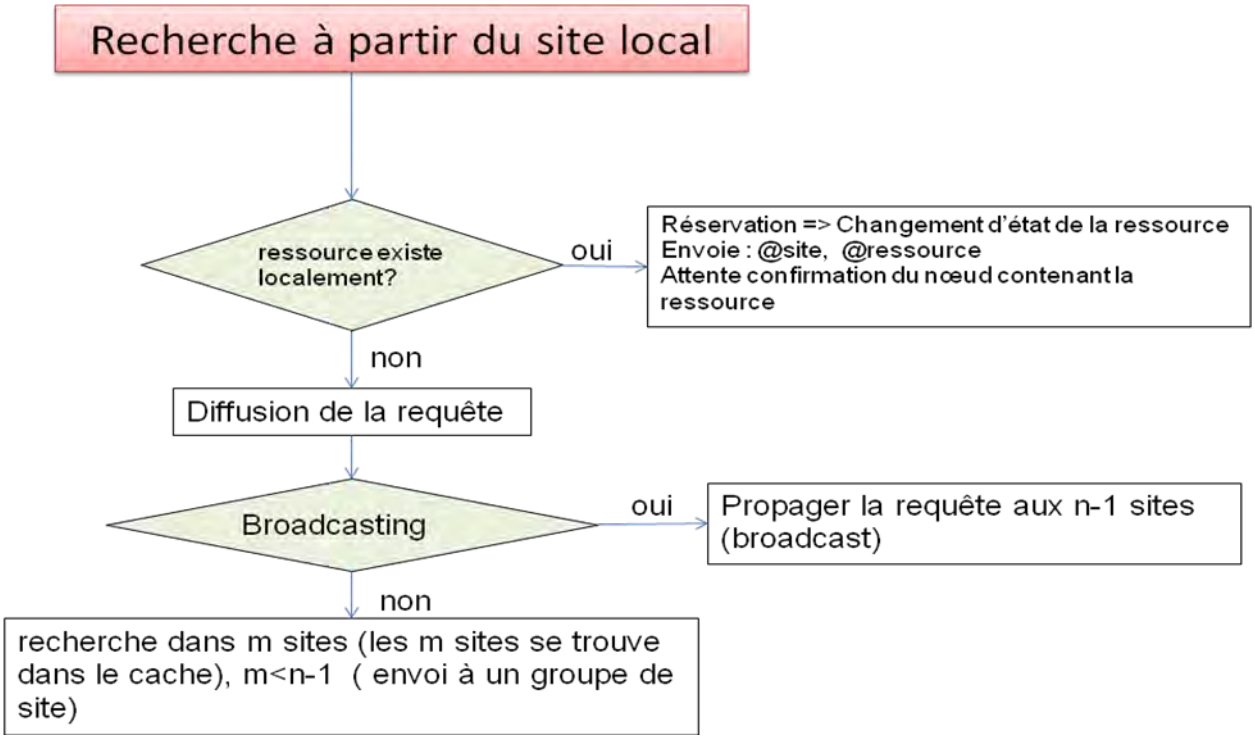


Figure 3.14 : algorithme de recherche

La recherches se fait à travers les GIIS et commence toujours par l'annuaire local ; si la solution n'existe pas localement (on n'a pas trouvé toutes les ressources) on a deux solutions :

- le GIIS local envoie la requête aux n-1 sites.
- Le GIIS local, connaissant un certains de sites m leurs envoie la requête.

IV.4.1 SOLUTION 1 : Inondation

IV.4.1.1 ARCHITECTURE

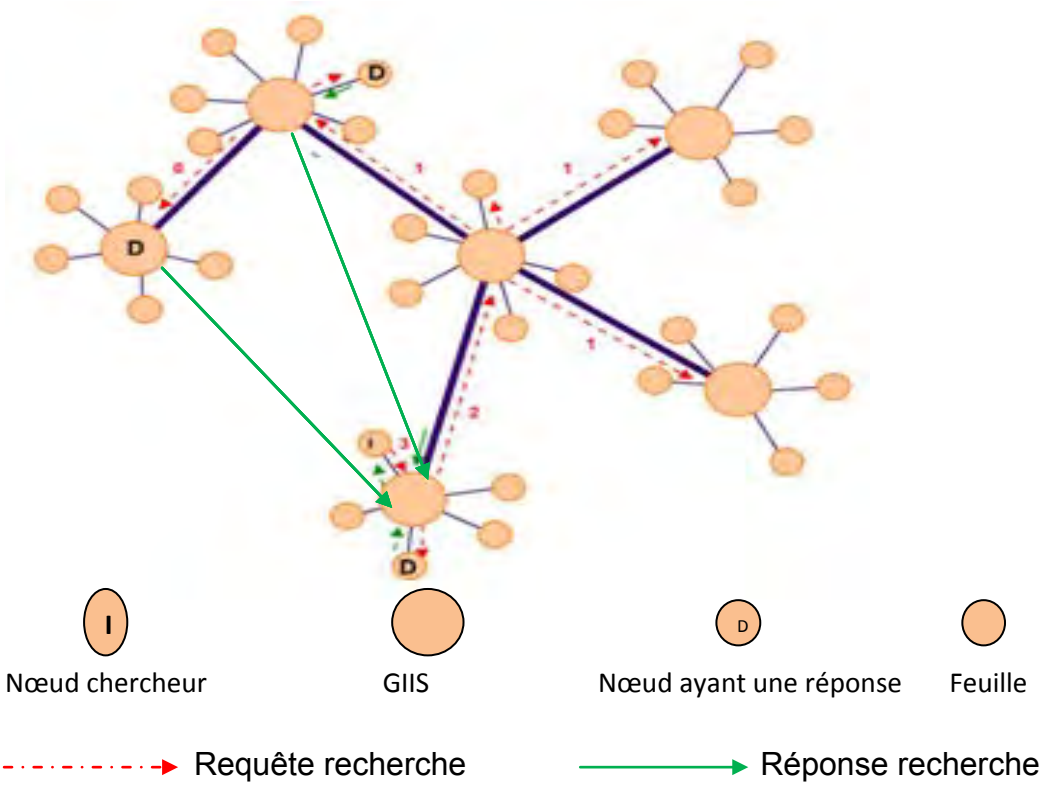


Figure 3.15 : Propagation d'une requête

Chaque site contient un GIIS lié à d'autre GIIS.

IV.4.1.2 FONCTIONNEMENT

La recherche se fait par inondation (voir figure 3.15) : un nœud (GIIS) envoie une requête de recherche à tous ses voisins qui la retransmettent à leur tour à tous leurs voisins et ainsi de suite. Pour limiter le nombre de retransmissions, un champ ttl (Time-To-Live) est associé à c haque message et est décrémenté de 1 à c haque retransmission. Quand celui-ci vaut 0, le message n'est plus retransmis.

Si un GIIS trouve dans ses index, une ressource qui convient à la requête, il envoie la description de la ressource et son adresse IP au GIIS initiateur de la requête. Ce GIIS choisit ensuite les ressources parmi les réponses reçues puis les transmet à l'utilisateur.

Recherche au niveau du GIIS locale

Ressources trouvées

Réservation

Envoi : @site, @ressource

Attente confirmation par le client (nœud feuille)

Ressources non trouvées.

Le GIIS local envoie un message Non Trouvée au client

Envoie la requête par inondation aux n-1 sites

L'adresse du site et de la ressource seront utilisées par le client qui va soumettre ses travaux.

La diffusion engendre un nombre important de messages donc une grande consommation de bande passante. Le trafic varie exponentiellement à la progression de la requête.

IV.4.2 SOLUTION 2

Cette solution consiste à interroger les nœuds qui ont le plus de chance de pouvoir répondre aux requêtes en utilisant les expériences passées. Cela permet entre autres d'interroger moins de voisins donc d'utiliser moins de bande passante, et de diminuer la latence tout en ayant autant voire plus de réponses.

Le système utilise deux types de liens entre les GIIS. D'une part, les GIIS sont connectés à un ensemble de voisins choisis aléatoirement, comme dans le cas de Gnutella.

Cet ensemble est supposé immuable et permet de garantir la connexité du réseau. D'autre part, les GIIS sont liés à un ensemble de « connaissances ». Chaque GIIS gère une liste ordonnée de connaissances (cache), et la met à jour à chaque réception de résultats d'une recherche. Les requêtes sont retransmises aux connaissances les mieux classés dans la liste.

Les informations mémorisées dans le cache pour un nœud donné sont : l'adresse logique du site (GIIS), l'adresse logique du nœud enregistré, les ressources dont il dispose (matérielles et logicielles), la date à laquelle le nœud a été enregistré dans le cache et la date de sa dernière allocation si elle est connue.

Chaque GIIS g associe à chaque connaissance r un score correspondant au ratio du nombre de requêtes que lui a envoyées u et pour lesquelles v a pu fournir des réponses, sur le nombre total de requêtes que lui a soumis g. Pour effectuer une recherche, les connaissances ayant les ratios les plus élevés sont interrogées. Des simulations montrent que cette technique, comparée à une approche à la Gnutella, permet de

diminuer d'un facteur 6 le nombre de requêtes de recherche reçues par nœud, et d'un facteur 3 le nombre de sauts nécessaire pour trouver une réponse.

Ainsi, lorsqu'un nœud effectue une requête de découverte de ressources, le cache est consulté et si des ressources présentes dans le cache peuvent répondre à la demande, elles sont contactées en priorité. Ensuite, si le nombre de réponses reçu est insuffisant, la requête est renvoyée aux voisins

Afin de réduire le nombre important de messages générés dans le système Gnutella notre choix porte sur cette deuxième solution.

IV.4.2.1 ARCHITECTURE

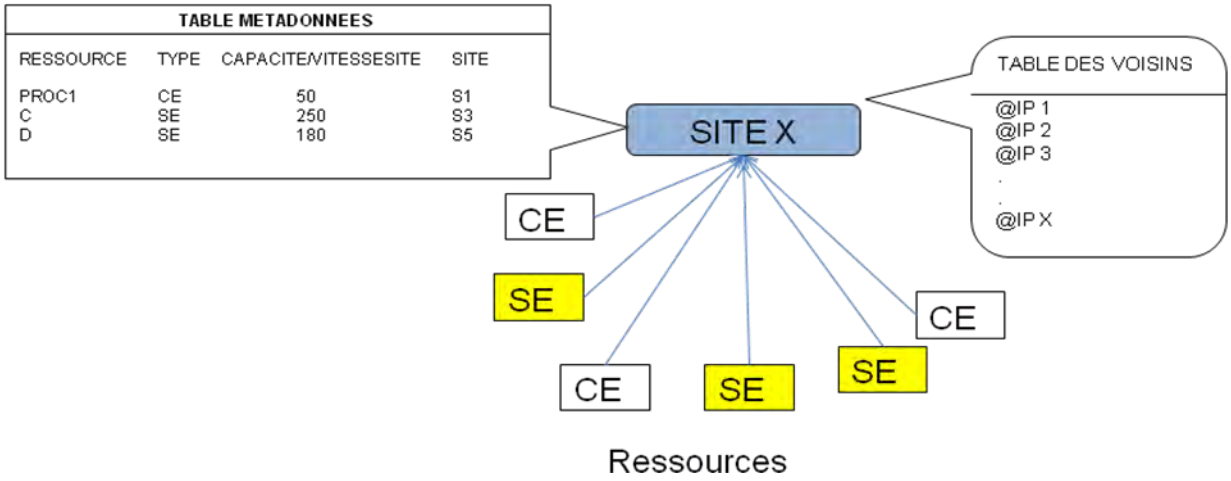
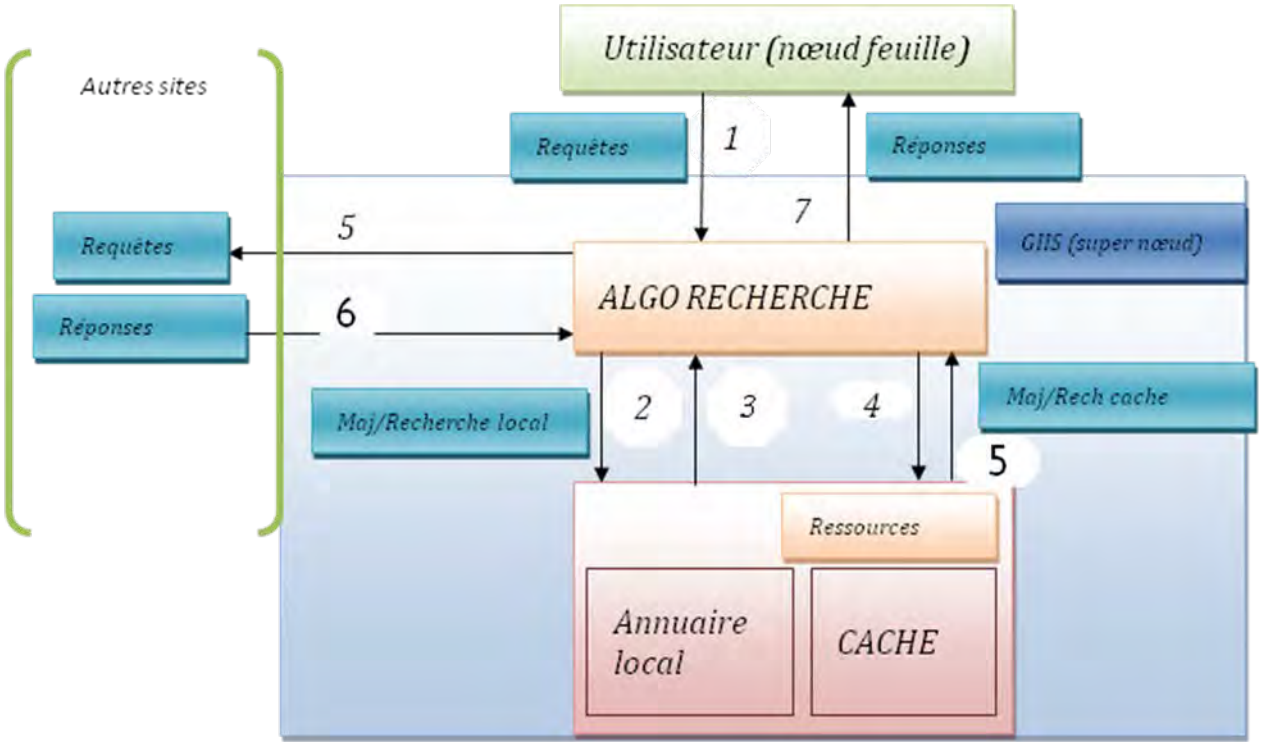


Figure 3.16 : les composants d'un site

Les super-nœuds (GIIS) gèrent une table de voisins et un cache (table de métadonnées. Il s'occupe de :

1. La localisation des sources
2. La décomposition des requêtes
3. L'envoi des requêtes aux sources
4. La recomposition des résultats

Cette solution suit une architecture hybride (décentralisée-centralisée) comme présentée précédemment.



3.17 : architecture de la solution 2

IV.4.2.2 FONCTIONNEMENT

L'utilisation d'un cache permet d'économiser de la bande passante mais pour que ce cache ait un intérêt dans notre contexte d'utilisation, il faut lui adjoindre une politique de gestion efficace.

Un cache n'est utile que si les données qu'il possède sont suffisamment à jour. Intuitivement, nous pouvons nous dire que les informations dans le cache ne sont utiles que si elles correspondent à des ressources libres. Par conséquent, notre politique de gestion de cache favorise les nœuds dont les ressources n'ont pas été récemment allouées. Avec l'utilisation d'un cache, les réponses réceptionnées sont ajoutées au cache si elles n'y sont pas déjà présentes. Par la suite, lorsque le service d'allocation de ressources choisit effectivement les ressources à utiliser parmi celles découvertes, il marque ces ressources comme allouées.

L'application de la politique de gestion du cache intervient pour deux opérations qui sont : la consultation du cache et l'éviction du cache.

Lorsque qu'un nœud effectue une requête de découverte de ressources et que le cache contient plus de nœuds pouvant répondre à la requête que ce qui a été demandé, la politique de gestion de cache retourne en priorité les nœuds n'ayant jamais été alloués ou ceux dont la date d'allocation est la plus ancienne.

Lorsque le cache est plein et qu'un nouveau nœud doit y être ajouté, le nœud dont la date d'allocation est la plus récente est évincé. Si la date d'allocation ne permet pas de faire la différence entre les nœuds présents dans le cache, c'est l'entrée la plus ancienne qui est évincée et dans le cas où la date d'insertion ne permet pas non plus de faire la différence, une entrée est supprimée au hasard.

Les découvertes de ressources peuvent être initiées depuis n'importe quel nœud de la grille et ceci, de façon aléatoire. Pour cette raison, la probabilité que deux requêtes de découverte de ressources soient initiées depuis le même nœud est relativement faible, surtout si le nombre de nœuds dans la grille est grand.

Étant donné que le cache d'un nœud est rempli à l'issue d'une découverte de ressources effectuée depuis ce nœud, il faut attendre un certain temps pour profiter des données insérées dans le cache, ce qui n'est pas souhaitable car les informations pourraient être obsolètes.

Afin d'améliorer la vitesse de remplissage des caches, nous pouvons ajouter un mécanisme de diffusion des informations découvertes. Ainsi, à l'issue d'une découverte de ressources, les réponses reçues, sauf celles correspondant aux ressources effectivement allouées, sont diffusées dans le voisinage du GIIS initiateur de la découverte dans le réseau non structuré.

Avec ce mécanisme, les caches se remplissent beaucoup plus vite, ce qui augmente considérablement leur intérêt. Cela pose toutefois un problème. Si un nœud ayant reçu une information diffusée de cette façon alloue la ressource concernée, le voisinage du nœud ayant diffusé l'information possède dans son cache une ressource récemment allouée sans le savoir, ce qui n'est pas très utile pour une allocation ultérieure. Pour cela, lorsqu'un nœud alloue une ressource dont les informations sont présentes dans son cache, il vérifie si ces informations proviennent d'une découverte de ressources ou d'une diffusion. Dans ce dernier cas, il prévient l'auteur de la diffusion de l'allocation de la ressource. Sur réception de ce type de message, le nœud qui avait effectué la diffusion d'information diffuse un message pour avertir ses voisins de l'allocation de la ressource concernée afin qu'ils puissent mettre à jour leur cache. Cela permet de garder les informations diffusées à jour dans les caches.

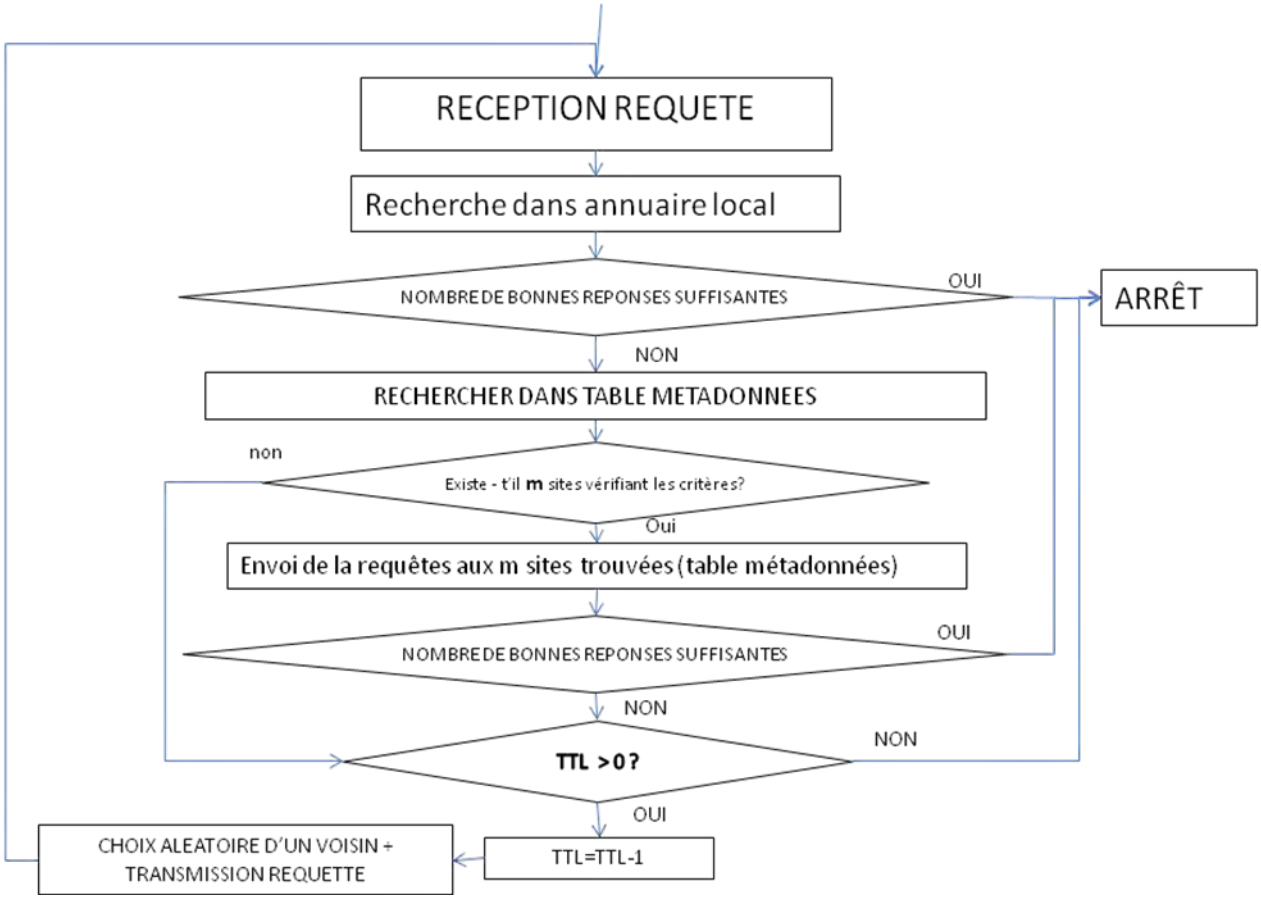
Lorsqu'un GIIS reçoit une requête il vérifie d'abord dans l'annuaire local et, à chaque fois s'il trouve une ressource, il le renvoie avec un tag indiquant qu'il est l'hébergeur. Sinon le GIIS transfert la requête (multicast) aux m sites (GIIS) trouvés dans le cache et dont les ressources ont des caractéristiques très proches de celles demandées. Si à

l'issu de ces recherches la requête n'aboutit toujours pas le GIIS transfert la requête à un autre GIIS (ratio le plus élevé) choisi dans la table des voisins.

Afin de contrôler la diffusion, l'utilisateur donne à chaque requête une durée de vie (Time To live) qui est décrémenté à chaque passage d'un GIIS. Si le TTL expire (=0), la requête échoue.

Pour éviter les boucles, chaque requête est associée à un ID global unique. Cet Identifiant permet aux nœuds de garder une trace des requêtes déjà traitées.

Chaque ressource a un ensemble d'état. Après chaque découverte de ressource le site le site auquel elle appartient la verrouille temporairement en positionnant son état comme NON_DISPONIBLE. Cet état repassera à DISPONIBLE s'il est alloué ou si le délai de garde est passé. L'envoi de la réponse est effectué de façon directe en utilisant une communication point-à-point



IV.4.2.2.1 RECHERCHE LOCALE

Au niveau d'un site la recherche se déroule exactement de la même manière que pour un arbre AVL (complexité : O(ln n)). Cependant la recherche n'est plus binaire dans le cas d'une recherche d'espace disque non contigu.

Algorithme 3.1 : Recherche locale de ressources

```
chercherContigu (Arbre racine, Type clé): élément
    si racine = NIL
        retourner NIL
    c = racine
    si c = clé
        retourner racine.contenu
    si c < clé
        retourner chercherContigu(racine.fils droit, clé)
    si c > clé
        retourner chercherContigu(racine.fils gauche, clé)
```

Les grilles permettent de stocker une donnée (fichier par exemple) sur plusieurs nœuds. Dans ce cas la recherche dans ce cas n'est pas binaire. L'algorithme 3.2 est exécuté. Les paramètres suivants sont utilisés par cet algoritme..

- R : ressource
- D : espace disponible de R
- filsGauche : fils gauche de R
- filsDroit : fils droit de R

Algorithme 3.2 : Recherche d'espace de stockage disponible

```
ChercherNonContigu(Arbre racine, TypeCle clé)
    v=clé
    Si v=0
        ArrêtRecherche
    Si racine=NIL
        ArrêtRecherche
    Si racine.contenu.D=0
        ChercherNonContigu(racine.filsGauche,v)
```

```
            ChercherNonContigu(racine.filsDroit,v)
     Sinon
            Si racine.contenu.D<v
                reserver(racine.contenu.D)
                v=v-racine.contenu.D
               ChercherNonContigu(racine.filsGauche,v)
               ChercherNonContigu(racine.filsDroit,v)
    Sinon
            reserver (v)
            v=0
            Retourner( racine)
            ArrêtRecherche
           FinSi
FinSi
```

Si l'espace recherché n'est pas atteinte on transmet la requête pour avoir le complément

Deux cas sont possibles pour possible pour la récupération des réponses :

- Attendre d'abord toutes les réponses
  - Trier et choisir les meilleures
- Réserver après chaque réponse positive

Chapitre 5

VALIDATION ET EXPERIMENTATION

Pour tester ou valider les algorithmes utilisés dans des systèmes de recherche ou dans des grilles, trois approches sont couramment employées : la preuve formelle, la simulation et le déploiement.

Si quelques preuves formelles ont été effectuées sur des systèmes de grille, nous pensons que le nombre de paramètres utilisés dans notre système et la problématique étudiée (pertinence et personnalisation des résultats) rend la première approche difficile dans notre cas.

La simulation permet d'observer des propriétés du système dans le cas où ses acteurs, ici les utilisateurs ou leurs nœuds, se comportent d'une manière prédéfinie. La simulation est, par définition, une imitation ou une représentation d'un système par un autre. En tant que telle, plusieurs aspects de la simulation sont des simplifications du système réel ou proviennent de déductions faites à partir d'études sur des systèmes similaires. Bien entendu, ces abstractions peuvent biaiser les résultats observés par simulation.

Le déploiement consiste en l'implémentation du système proprement dit, et en sa diffusion. Cette approche, bien qu'elle permette de tester en situation réelle le logiciel, demande un investissement important pour la conception et l'implémentation du logiciel sans assurance qu'il soit réellement utilisé.

V.1 VALIDATION THEORIQUE

Dans cette partie, malgré le nombre important de paramètres nous présentons un ensemble d'évaluations théoriques des propositions que nous avons présentées dans ce mémoire.

V.1.1 COMPLEXITE

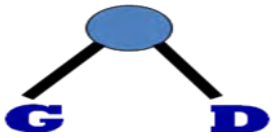
- ***Recherche local*** : au niveau de chaque site nous avons une structure d'arbre AVL pour l'enregistrement des ressources locales. Comme toutes les opérations

AVL sont en $O(\log_2 N)$, nous pouvons conclure qu'au niveau d'un site toutes les opération sont en $O(\log_2 N)$ si on suppose que le délai de transmission est nul..

- ***Recherche global*** : en supposant le temps de propagation de la requête entre deux sites négligeable par rapport au temps de recherche on peut conclure que la recherche a une complexité moyenne de $O(\log_2 N)$ ($O(m \log_2 N)$) et une complexité maximale de $O(N \log_2 N)$.
  La recherche est en $O(\log_2 N)$) dans le meilleur des cas (ressources trouvées localement) si on suppose que le délai de transmission est nul.

Preuve :

Soit N_h = minimum des différents nœuds d'un arbre local avec une hauteur h.



- $N_0 = 0$.
- $N_1 = 1$.
- N_h, $h > 1$
- G et D sont des arbres AVL.
- La hauteur de l'un est h-1
- La hauteur de l'autre est h-2.
- Le sous arbre ayant une hauteur h-1 à N_{h-1} noeuds.
- Le sous arbre ayant h-2 à N_{h-2} noeuds.
- alors, $N_h = N_{h-1} + N_{h-2} + 1$.

Séquence de nombre de Fibonacci

- $F_0 = 0$, $F_1 = 1$.
- $F_i = F_{i-1} + F_{i-2}$, $i > 1$.
- $N_0 = 0$, $N_1 = 1$.
- $N_h = N_{h-1} + N_{h-2} + 1$, $h > 1$
- $N_h = F_{h+2} - 1$.
- $F_i \sim \frac{\phi^i}{\frac{1}{\sqrt{5}}}$
- $F_i \approx \frac{1}{\sqrt{5}} \left[\left[\frac{1+\sqrt{5}}{2}\right]\right]^i$.
- ϕ = (1 + sqrt(5))/2

$$N_h \approx \frac{1}{\sqrt{5}} \left[\left[\frac{1+\sqrt{5}}{2}\right]\right]^{h+3} \quad \Rightarrow \quad h \approx 1.44 log_2 N_h \quad \Rightarrow \quad h \approx 1.44 log_2 N$$

$\Rightarrow$ Toutes les opérations sont en O($\log_2$ N)

V.1.2 OVERHEAD (charge)

Notre algorithme permet de limiter de façon significative le coût, du point de vue de la bande passante réseau consommée, d'une découverte de ressources par rapport à l'inondation de requêtes puisque la recherche peut être arrêtée dès que suffisamment de réponses sont reçues.

V.1.3 GESTION RESSOURCE

L'objet principal de notre travail était de présenter une solution distribué du MDS. Dans notre solution nous avons distribué l'annuaire en utilisant une topologie d'arbre AVL pour la représentation des ressources. Ce choix a été motivé par les propriétés de ces structure qui permettent d'obtenir une complexité en O($\log_2$ N) pour toutes les opérations (ajout, suppression et recherche)

IV.1.3.1 REPARTITION

Chaque site gère ses propres ressources. L'ajout et la suppression se font localement. Tout d'abord, les services distribués que nous proposons peuvent être utilisés dans un contexte où les nœuds sont volatils. Ainsi, nous pouvons envisager une utilisation de notre système dans une grille très dynamique mais aussi dans des des grilles moins dynamiques telles que les fédérations de grappes.

	Type de réseau
DGRID	structuré
Notre modèle DGIIS	Non structuré

IV.1.3.2 RECHERCHE

Les utilisateurs de notre système pourront rechercher des ressources disponibles sur les différents sites de la grille. Nous nous sommes inspirés des mécanismes de

recherche des systèmes pair-à-pair présentés précédemment pour fournir des résultats à jour.

	DGRID	NOTRE SOLUTION DGIIS
TYPE DE REAU	STRUCTURE	NON STRUCTURE
RESSOURCE	ENREGISTREMENT SUR L'ANNEAU	AU NIVEAU DU GIIS LOCAL
REPRESENTATION DES RESSOURCES	LISTE	ARBRE AVL => recherche dichotomique
RECHERCHE	BASEE SUR DES VALEURS	VALEUR ET INTERVALLE
GRANDE REPLICATION	PAS ADAPTER	ADAPTER

V.1.4 Conclusion

Dans la plupart des modèles déjà proposé, les requêtes de recherche dans les systèmes d'échanges d'informations correspondent à une recherche de fichiers selon leurs identifiants (nom du fichier). Ceci est différent d'un système de recherche dans les grilles où l'utilisateur ne connaît pas, à priori, la ressource qu'il recherche et pour laquelle il ne possède comme seul indice que quelques caractéristiques. En plus la plupart des solutions existant propose des grilles structurées.

Comparé aux solutions existantes comme Dgrid et Samgrid, notre modèle de grille non-structurée est plus adapté aux recherches multicritères, et aux réplications.

V.2 VALIDATION PRATIQUE : EXPERIMENTATION

V.2.1 Choix d'un outil de simulation

V.2.1.1 Critères utilisés

Dans cette section nous essayons de trouver, d'après une série de critères pertinents vis-à-vis de notre problème, les simulateurs qui s'y conforment.

Niveaux de détail Comment sont modélisés le réseau physique, l'annuaire local au niveau de chaque site ?

Le passage à l'échelle Quel est le maximum de nœuds que grille peut gérer ? Le nombre de nœuds des grilles étant relativement élevé, il est nécessaire, pour des raisons de validité, qu'on puisse vérifier le fonctionnement de la grille sur un nombre conséquent de nœuds

La latence et la bande passante

V.2.1.2 Les Simulateurs existants

Le tableau 4.1 présente un résumé de l a revue bibliographique de différents simulateurs. Une remarque que l'on pourrait soulever ici, est l'absence d'un simulateur « standard ». Malgré des efforts dans ce sens et à l'inverse de la simulation des réseaux dits « bas niveau », la simulation de grilles ne semble pas avoir atteint un s tade de maturité satisfaisant pour pouvoir bénéficier d'un référent reconnu par tous. La section suivante présente le simulateur qui semble le plus correspondre aux critères énoncés ci-dessus. Il s'agit du simulateur Oversim qui a déjà implémentés des protocoles très proches du notre.

SIMULATEUR	AVANTAGES	INCONVENIENTS
P2PSim		Statistiques limitées, passage à l'échelle : 3000 nœuds
PeerSim	passage à l'échelle : 106 nœuds en mode cyclique	Réseau physique non modélisé, Peu de documentation sur le mode événement discret
Gridsim	Simulateur de grille	N'a pas implémenter beaucoup de protocole

PlanetSim	passage à l'échelle : 105 noeuds	pas de statistiques, réseau physique non modélisé
OverSim	passage à l'échelle 105 noeuds, réseau physique modélisé, statistiques	protocoles implantés : chord et GIA

Tableau. 4.1 – Caractéristiques techniques

IV.2.1.3 OVERSIM

OverSim est un simulateur de réseau, open-source fonctionnant sous l'environnement de simulation OMNeT + + / OMNEST. Il contient plusieurs modèles de protocoles de réseaux structurés (par exemple, Chord, Kademlia, Pastry) et non structurées (par exemple GIA).

IV.2.2 Expérimentation

Nous présentons, dans cette partie, les résultats de l'expérimentation sur l'algorithme de recherche de ressources dans notre architecture distribuée du MDS. Pour expérimenter l'algorithme de rechercher de ressources proposé, nous avons utilisé un PC Pentium M de 2GHz, doté d'une mémoire de 2Go et fonctionnant sous Windows-Vista

.

IV.2.2 .1 Protocole expérimental

Cette partie décrit les paramètres qui peuvent influer sur le système (voir la liste ci-après) et un scénario référent. Ce scénario est relativement simple, il comprend deux périodes principales : la première période est une phase dans laquelle un certain nombre de recherches est lancé dans la grille. La deuxième concerne l'augmentation du nombre de sites. La section suivante présente les diverses expérimentations effectuées.

- Le nombre de GIIS (sites) présents dans le système N ;
- Le nombre de ressource présents dans le système S ;
- Le nombre de voisins *v* pour chaque GIIS.
- La taille maximale d'un cache
- Le nombre de messages m présents dans la grille

IV.2.2 .2 expérimentations

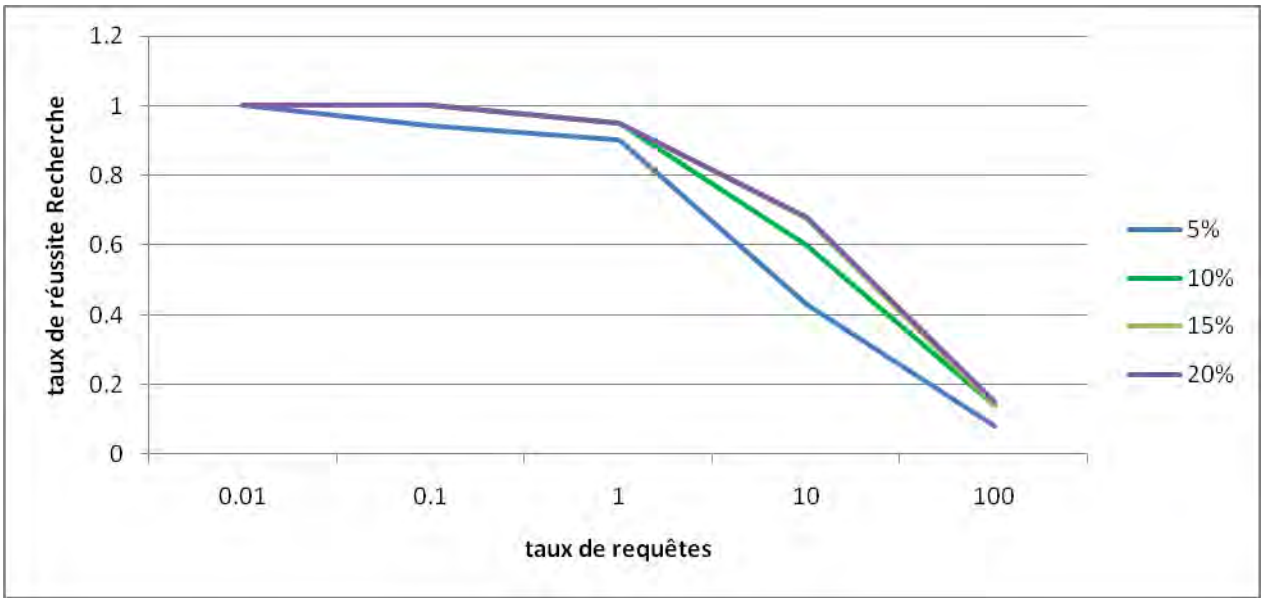
Dans la première série d'expérimentations, on se propose de vérifier la surcharge de la grille en augmentant le nombre de requêtes (donc le nombre de message). On se pose la question suivante : est-cette augmentation de m perturbe pas le comportement du système? Pour y répondre, on ef fectue plusieurs séries d'expérimentations. Dans chaque série on ne fera varier qu'un seul paramètre (m) et on confrontera les résultats avec les résultats attendus.

La problématique du passage à l'échelle, la capacité de celui-ci d'intégrer un nombre de sites de plus en plus important vis-à-vis des performances, est relativement sensible. Ici il s'agira de vérifier la variation du temps d'exécution des simulations.

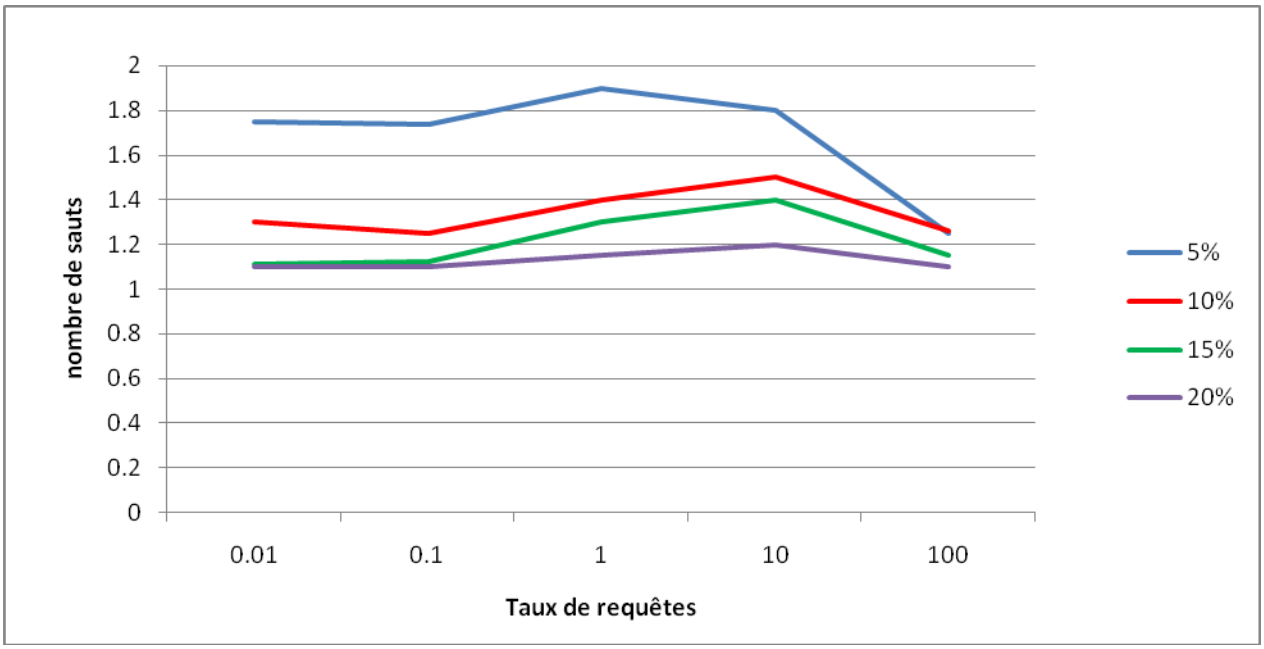
En raison des contraintes de temps, et les simulations étant toujours en cours, ce point sera évoqué dans les perspectives

IV.2.2 .3 Premiers résultats

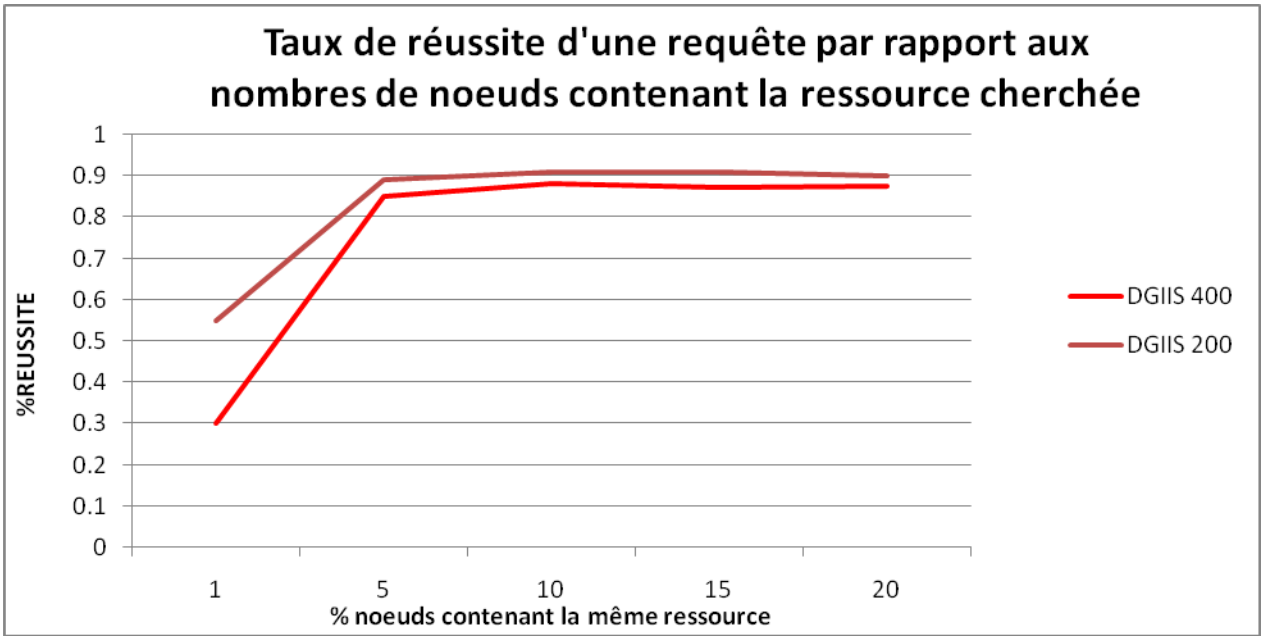
IV.2.2 .3.1 taux bonnes réponses /taux de requêtes



IV.2.2 .3.2 nombre sauts /taux de requêtes



- IV.2.2 .3.3 pourcentage de réponses justes par rapport aux pourcentage de nombre de noeuds contenant la ressource



La croissance de la grille n'a pas une grande influence sur le taux de réponses justes quand le nombre de réplications augmente.

Chapitre 5

CONCLUSION ET PERSPECTIVES

V.1 CONCLUSIONS

Nous avons proposé un système d'information distribué adapté à des grilles de très grande échelle. Il permet de découvrir des ressources dans la grille à partir d'une description composée d'un ensemble de critères qui représentent les pré-requis matériels et logiciels pour l'exécution d'une application.

Ce service repose sur l'utilisation de grille non structurées et sur le concept de marches aléatoires. Cette approche permet de s'affranchir d'une solution de type annuaire centralisé, fréquemment utilisée dans l'état de l'art, qui pose une limite au passage à l'échelle.

L'utilisation d'une grille non structurée rend le système d'information auto-organisant. En effet, l'ajout de ressources dans la grille se résume à une opération locale d'insertion dans le système et la suppression d'une ressource se fait en local.

L'évolution des caractéristiques d'une ressource ne nécessite aucune intervention de l'administrateur.

Comme le reste de l'architecture du système, le système d'information est tolérant à la défaillance de n'importe quel élément de la grille grâce aux propriétés d'auto-réparation d'une grille non structurée.

Nous avons proposé une optimisation du protocole de marches aléatoires afin d'en augmenter l'efficacité dans le cadre d'utilisation des grilles. Le concept repose sur l'utilisation de caches dotés d'une politique favorisant la découverte de ressources libres et sur la dissémination d'une partie des informations découvertes dans le réseau non-structuré.

Nous avons également proposé un mécanisme permettant d'augmenter les chances de localiser les ressources rares d'une grille.

Le système d'information que nous avons proposé est un service dont les propriétés sont nécessaires à la conciliation des éléments suivants : la grande échelle d'une grille, la volatilité des ressources et la simplicité d'administration. Ce service contribue aussi à la simplicité d'exécution d'applications sur la grille puisqu'il affranchit les utilisateurs de

la recherche de ressources disponibles et du besoin de connaître globalement les ressources de la grille.

V.2 PERSPECTIVES

Au terme de cette thèse, nous envisageons un ensemble de perspectives de recherche à plus ou moins long terme.

- Court terme :
  - Évaluations étendues. Nous proposerons une évaluation+ expérimental
  - Un premier axe de travail à court terme est lié à l'évaluation de nos concepts. Il serait intéressant d'évaluer plus finement le protocole de découverte de ressources proposé en faisant varier d'autres paramètres comme la profondeur d'une marche aléatoire ou encore la taille des caches.

Bibliographie

[1] I. Foster, C. Kesselman, S. Tuecke, « The Anatomy of the Grid : Enabling Scalable Virtual Organizations", IJSCAHPC, 15(3), 2001: http:/www.globus.org/alliance/publications/papers/anatomy.pdf

[2] IBM Corporation, 2006, «Pour une prise en main rapide, économique et à faible risque des technologies d'informatique distribuée par réseau : Grid and Grow Express IBM »

[3] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", I Fran Berman, Anthony J.G. Hey Geoffrey Fox, editor, Grid Computing : Making The Global Infrastructure a Reality, John WILEY,2003.ISBN:0-47085319-0: http: ://media.wiley.com/product_data/excerpt/90/04708531//0470853190.pdf

[4] Douglas Thain, Todd Tannenbaum, and Miron Livny,"Distributed Computing in Pratice : The Condor Expérience", Concurrency and Computation : Practice and Experience, Vol., 17, No. 2-4, pages 323-356, February-April, 2005. http:/www.cs.wisc.edu/condor/doc/condor-practice.pdf

[5] Avizienis, A, Laprie, J.C. and Randell, B, " Fundamental Concept of Dependability", in Procedings of the 3rd IEEE information Survivability Workshop (ISW-2000), Boston Massachusetts, USA, October 24-26, 200 pp. 7-12: http/www.cs.ncl.ac.uk/research/pubs/trs/papers/739.pdf

[6] IBM Corporation, 2006, «Pour une prise en main rapide, économique et à faible risque des technologies d'informatique distribuée par réseau : Grid and Grow Express IBM »

[7] Michel Cosnard et Thierry Priol INRIA Sophia Antipolis, « Globalisation des ressources informatiques et des données »

[8] The Globus Alliance. URL : http://www.Globus.org/.

[9] Ian T. FOSTER, Carl KESSELMAN, Gene TSUDIK et Steven TUECKE. A Security Architecture for Computational Grids. In 5th ACM Conference on Computer and

Communications Security, pages 83–92. ACM Press, New York, NY, San Francisco, CA, 1998.

[10] DOE Science Grid. URL : http://DOEScienceGrid.org/.

[11]GriPhyN: Grid Physics Network. URL : http://www.GriPhyN.org/.

[12] http://www.renater.fr/

[13] 49 DataTAG. URL : http://DataTAG.web.cern.ch/datatag/.

[14] Karl CZAJKOWSKI, Steven FITZGERALD, Ian FOSTER et Carl KESSELMAN. Grid Information Services for Distributed Resource Sharing. In 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC), pages 181–194. San Francisco, CA, août 2001

[15] Globus MDS. URL : http://www.Globus.org/toolkit/mds/.

[16] RSL Specification 1.0. URL : http://wwwfp.Globus.org/gram/rsl_spec1.html.

[17] THE GLOBUS PROJECT. GridFTP: Universal Data Transfer for the Grid. Whitepaper, University of Chicago, septembre 2000.

[18] THE GLOBUS PROJECT. GridFTP update. Rapport technique, Argonne National Laboratory, janvier 2002.

[19] RFT: Reliable File Transfer Service. URL : http://www.Globus.org/toolkit/docs/3.2/rft/index.html.

[20] RLS: Replica Location Service. URL : http://www.Globus.org/toolkit/docs/3.2/rls/key/index.html.

[21] Joseph BESTER, Ian FOSTER, Carl KESSELMAN, Jean TEDESCO et Steven TUECKE. GASS: A Data Movement and Access Service for Wide Area Computing Systems. In 6th Workshop on Input/Output in Parallel and Distributed Systems (IOPADS), pages 78–88. ACM Press, Atlanta, GA, mai 1999

[22] Matei RIPEANU et Ian FOSTER. A Decentralized, Adaptive Replica Location Mechanism. In 11th IEEE International Symposium on High Performance Distributed Computing (HPDC), pages 24–32. Edimbourg, Écosse, juillet 2002.

.[23] Nour-Eddine Oussous, ´Eric Wegrzynowski : Les tables de hachage; http://www2.lifl.fr/~wegrzyno/portail/API2/Doc/Cours/tables.4on1.pdf

.[24] Ion Stoica□, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan_ MIT Laboratory for Computer Science chord@lcs.mit.edu http://pdos.lcs.mit.edu/chord/