République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur Et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF
Faculté des Mathématiques et Informatique

polycopié cours :
GRILLES INFORMATIQUES

SENHADJI Sarra

Avant propos

ce cours s'adresse particulièrement aux étudiants du cycle MASTER ayant déjà acquis quelques connaissances dans les systèmes distribués.
Il peut aussi être utilisé par les enseignants comme support à leur méthode et démarche d'enseignement selon un contexte pédagogique bien précis.

but du cours :

Ce cours permettra aux étudiants d'élargir les connaissances sur les systèmes distribués classiques et il leur donnera les fondements nécessaires pour pouvoir découvrir d'autres types de systèmes distribués dans la suite de leur cursus universitaires.

contenu du cours :

avant de d'introduire les grilles informatiques, un rappel sur les concepts et paradigmes des systèmes distribués sera présenté afin de comprendre les avantages et les invonvénients de ces derniers.
les systèmes pairs à pairs seront définis et comparés aux grilles informatiques.
les grilles seront détaillées par la suite, en définissant leur concepts, leurs architectures et leur fonctionnement.
Les problèmes majeurs liés à ce type de système seront présentés comme la réplication et la cohérence des données.
un exemple de middleware « GLOBUS » sera présenté à la fin de ce document.

organisation du cours :

Le cours est structuré suivant six principaux axes :

1. **systèmes répartis**
2. **systèmes pairs à pairs**
3. **grilles**
4. **Gestion des ressources dans les grilles**
5. **réplication**
6. **cohérence**
7. **GLOBUS**

remarques :

ce cours peut bien évidemment être complété par d'autres cours plus introductifs aux systèmes distribués et les différents concepts liés à ce type de systèmes.

Table des matières

Table des figures

Chapitre 1

INTRODUCTION AUX SYSTEMES DISTRIBUES

1.1 Introduction

Les systèmes distribués sont apparus afin de répondre aux limites de la gestion centralisée des intitutions multi sites. Ces systèmes permettent aussi de mettre en place des architectures informatiques permettant d'améliorer les performances, la transparence ainsi que la disponibilité des systèmes informatiques. Dans ce chapitre, nous décrivons l'architecture et le fonctionnement des systèmes distribués, tout en présentant leurs caractéristiques, avantages ainsi que leur mode de communication.

1.2 Définition 1

Un ensemble d'ordinateurs indépendants (autonomes) qui apparait à un utilisateur comme un système unique et cohérent.

1.3 Définition 2

Un système distribué est composé de :

- Un ensemble d'éléments reliés par un système de communication ; ces éléments ont des fonctions de traitement (processeurs), de stockage et de connexion avec le monde extérieur.
- Collaboration entre les différents éléments.

1.4 Avantages et inconvénients des systèmes distribués par rapport aux systèmes centralisés :

1.4.1 Avantages :

- **Economique** : excellent rapport performance/ prix des microprocesseurs.
- **Puissance de calcul :** c'est un système pluri processeurs qui offre une puissance de calcul supérieure à celle d'un seul processeur.

— **Distribution naturelle de certaines applications :** comme par exemple les systèmes de contrôle d'une chaine de production, distribution géographique d'agences bancaires.
— **Haute disponibilité :** la défaillance d'une machine n'affecte pas les autres.
— **Evolution progressive**

1.4.2 Inconvénients :

— **Réseaux de communication** : Saturation, perte des données, latence (temps perdu dans l'envoi des messages), pannes.
— **Sécurité :** Nombre d'entités impliquées (plus d'effort pour gérer le système).
— **Dynamisme** : Administration du système : est ce qu'il y a un administrateur ou plusieurs ?
— Pas d'horloge globale
— Pas d'état global.

l'une des caractéristiques principales d'un système réparti est la transparence .

1.5 Transparence

Le but de la transparence est de cacher l'architecture, le fonctionnement des applications ou la complexité du système distribué pour apparaitre à l'utilisateur comme une machine unique et cohérente.
La transparence se traduit de différentes manières :

— transparence d'accès : accès aux ressources distantes aussi facilement que locales.
— transparence de la localisation : accès aux ressources quelque soit leur localisation.
— transparence de la concurrence : exécution de plusieurs processus en parallèle avec l'utilisation de ressources partagées.
— transparence à la réplication : possibilité de dupliquer certains éléments par rapport aux ressources afin d'augmenter la fiabilité.
— transparence de la mobilité (migration) : possibilité de déplacer les éléments par rapport aux ressources.
— transparence des pannes : si une machine tombe en panne les processus continuent à s'exécuter grâce à une gestion de la tolérance aux pannes.
— transparence de la performance : possibilité de reconfigurer le système pour augmenter les performances.
— transparence d'échelle : doit supporter l'augmentation de la taille du système (nombre d'éléments, nombre de ressources,...).

ces systèmes répartis sont gérés par des protocoles de communication.

1.6 Communication

un protocole de communication est un ensemble de règles et de contraintes gérant une communication entre plusieurs entités.

1.6.1 Buts :

— Se mettre d'accord sur la façon de communiquer pour bien se comprendre.
— S'assurer que les données envoyées sont bien reçues.

1.6.2 Types d'informations :

- Les données à échanger.
- Les données de contrôle et de gestion des protocoles.

1.6.3 Modèles d'interaction

1.6.3.1 Client/ serveur

client : application qui cherche à exploiter des services proposés par le serveur.
ne dispose pas d'information, mais est capable d'aller en chercher.
serveur : contient et traite l'information.
avantage : simplicité de fonctionnement et de mise à jour.
i**nconvénent :** les performances (temps de réponse du serveur) risquent de se dégrader en fonction des demandes.

1.6.3.2 pairs à pairs (P2P)

les P2P sont caractérisés par :

- Chaque machine est serveur et client.
- Pas besoin de grosses machines.
- Détenteurs d'informations sont reliés entre eux.
- Une fois que le demandeur sait ou est l'information qu'il cherche, les deux ordiunateurs sont alors en mesure de s'échanger directement des informations.

Avantage : pas de machines couteuses.
Inconvénient : complique la notion de recherche.

1.7 Conclusion

après cette breve présentation des systèmes distribués nous passerons aux systèmes P2P dans le prochain chapitre.

Chapitre 2

SYSTEMES PAIRS A PAIRS « P2P »

2.1 Introduction

les systèmes P2P sont apparus afin de palier aux problèmes liés aux modèles client/serveur.

2.2 Architectures des systèmes P2P

Dans les systèmes P2P, on trouve 3 principaux types d'architecture : centralisée, décentralisée et hybride.

2.2.1 Architecture centralisée

Dans ce type d'architecture, on trouve 2 types :

2.2.1.1 Architecture centralisée à serveur unique

Cette architecture est basée sur 2 principales entités (client, serveur). chaque entité joue un rôle spécifique.
Serveur :

- Serveur centralisé ne contient pas les fichiers.
- Il dispose de deux types d'informations sur les fichiers (nom, taille,...) et sur les utilisateurs (nom utilisé, IP, nbr de fichier...).

Client :
Une fois connecté grâce au logiciel spécifique :

- Il peut faire des recherches comme avec un moteur de recherche classique.
- Il obtient une liste d'utilisateurs disposant de la ressource désirée.
- Il suffit alors de cliquer sur le bon lien pour commencer le téléchargement. Cette étape est indépendante du serveur et représente la seule partieP2P du logiciel.

La recherche est facilitée puisque un seul serveur contient les informations.
inconvénient :

- Tout repose sur le serveur et le problème se pose lorsque le réseau est inactif ou surcharghé.
- Aucun anonymat n'est possible car chaque utilisateur est identifié sur le serveur.

Exemple de réseau centralisé : NAPSTER est l'un des premiers logiciels **P2P** à architecture centralisée.

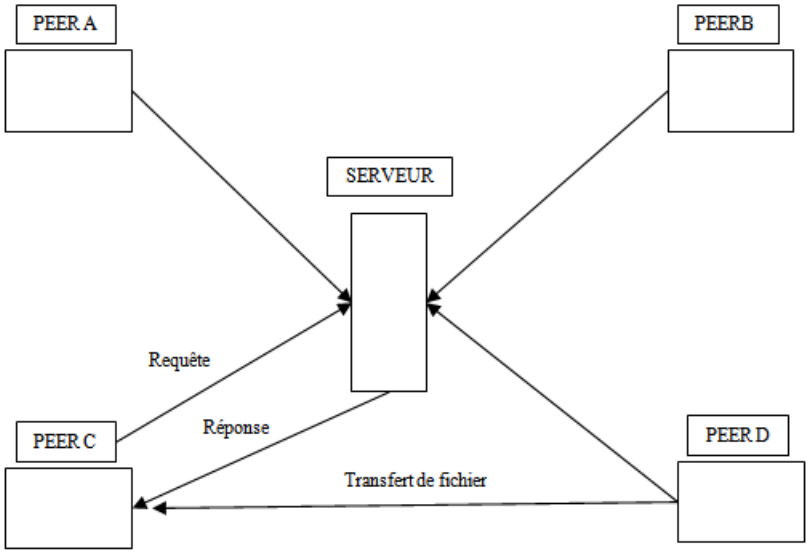

FIGURE 2.1 – architecture P2P centralisée

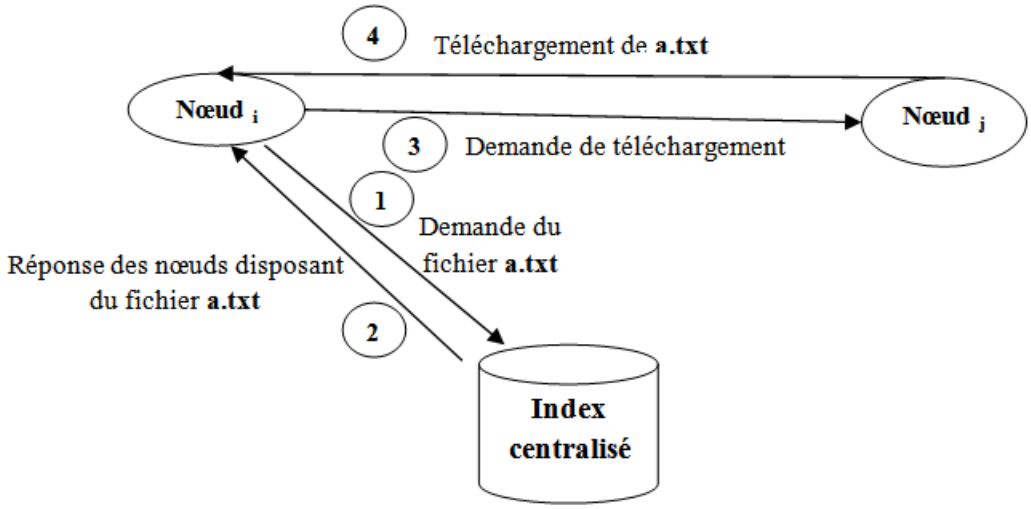

FIGURE 2.2 – Réseau NAPSTER

2.2.1.2 Architecture centralisée à serveurs multiples

Afin de remédier aux limites de l'architecture centralisée à un seul serveur, un réseau de serveurs est utilisé.

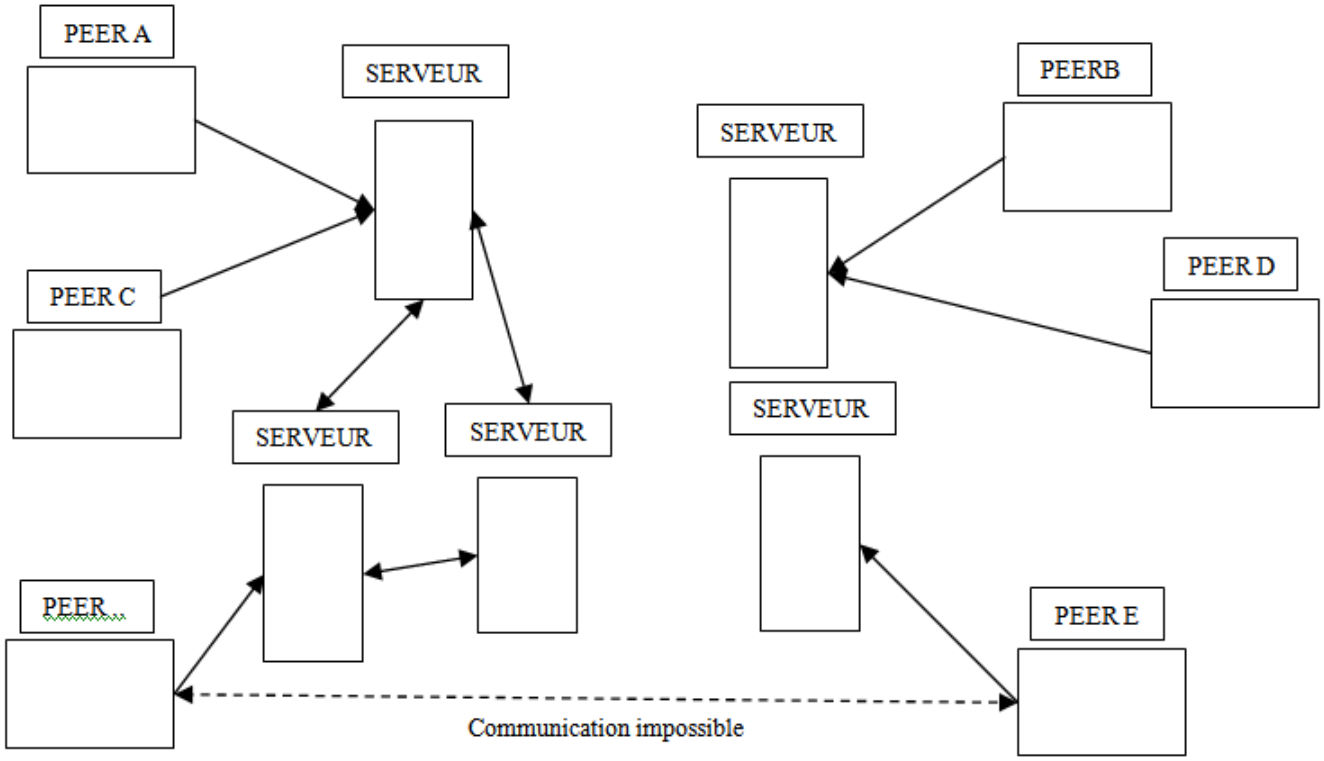

FIGURE 2.3 – Architecture centralisée à serveurs multiples

Lorsque le client lance le logiciel, il est obligé de choisir un serveur auquel se connecter.
Il doit connaître au moins un serveur.
Une liste de serveurs est disponible sur le net.
Le problème est de choisir le bon serveur.
Exemple de ce type d'architecture est le réseau eDonkey.

2.2.2 Architecture décentralisée

L'exemple le plus connu de ce type d'architecture est GNUTELLA.
lorsqu'un client souhaite se connecter il envoie un message broadcast afin de savoir quels autres pairs sont actifs.
seuls les pairs répandant aux messages broadcast sont alors connecté au réseau.
un problème se pose, c'est qu'on ne peut pas connaitre la topologie de la grille.

Principe général :

plusieurs principes sont imposés dans ce type d'architecture pour les noeuds qui veulent se connecter à ce réseau.

Principe d'égalité :

- Même capacité (puissance, bande passante...)
- Même comportement (à la fois client et serveur)

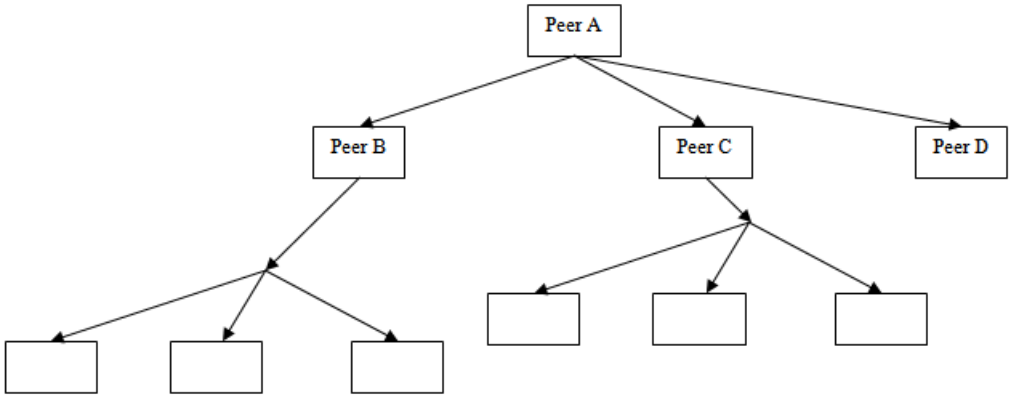

FIGURE 2.4 – GNUTELLA

— Bon comportement (pas de « mensonge »)

Principe de requêtes « populaires » :

Les ressources demandées sont fortement répliquées.
La plupart des requêtes ne concernent que peu de ressources.

Principe de topologie de réseau :

Graphe minimisant le nombre de chemins entre deux nœuds.
La longueur du chemin minimum entre 2 nœuds est faible (5 à 8).

Avantages :

— La taille d'un tel réseau est théoriquement infinie contrairement aux autres architectures dont le nombre de clients dépend du nombre et de la puissance des serveurs.
— L'utilisation d'un tel réseau est anonyme, il est impossible d'y repérer quelqu'un volontairement
— Réseau très tolérant aux fautes.
— S'adapte bien à la dynamique du réseau (apparition et disparition des pairs).

Inconvénients :

— Forte consommation de la bande passante.
— Pas de garantie de succès, ni d'estimation de la durée des requêtes.
— plusieurs messages sont utilisés dans le réseau Gnutella (voir le tableau2.1)

Type de message	description	Information
Ping	Annonce la disponibilité et lance une recherche de pair	Vide
Pong	Réponse à un ping	IP et N° de port Nombre et taille des fichiers partagés
Query	requête	Bande passante minimale demandée Critères de recherche
Query hit	Réponse à Query si on possède la ressource	IP + N° port+ bande passante+ nombre de réponse+ descripteur
Push	Demande de téléchargement pour les pairs derrière un firewall	IP du pair, index du fichier demandé @ IP + N° port où envoyer le fichier

TABLE 2.1 – GNUTELLA

Ajout d'un nœud :

Le nœud A souhaite se connecter au réseau GNUTELLA, il émet un message ping (en vert) ...
Le nœud B va :

- Répondre au nœud A par un pong (en rouge).
- Transférer le message de A vers les 2 clients qu'il connait C et D.
- Ceux-ci répètent alors la même chose (sauf si le nombre de propagation a déjà été atteint).

Remarque :
Le nœud E connait l'existence du nœud A grâce à C et D mais il ne répond qu'une fois (via le chemin le plus court ou le premier qu'il l'a contacté).

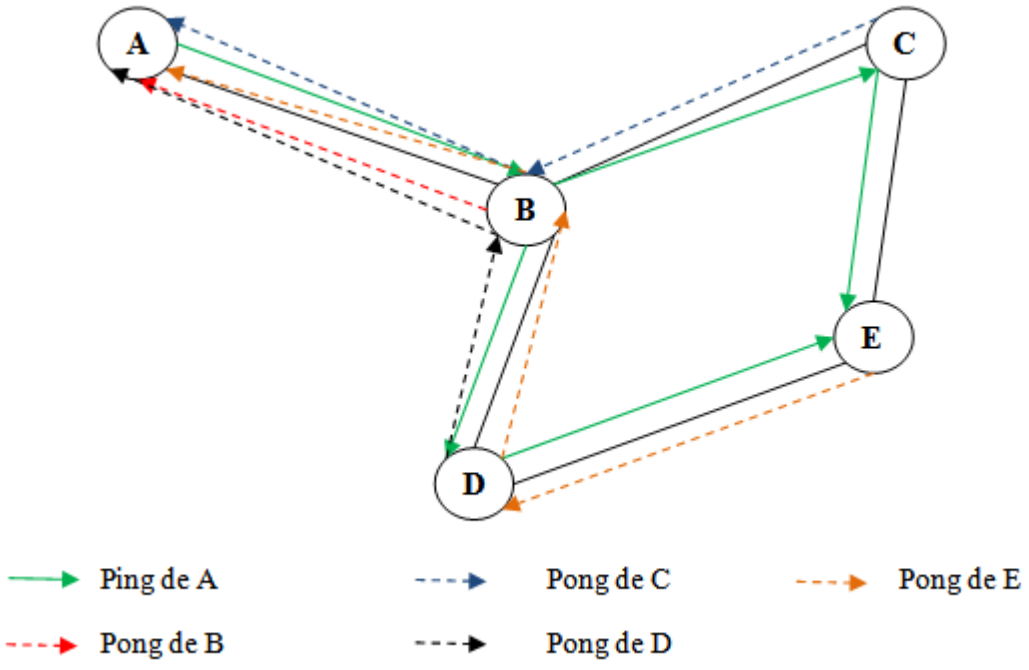

FIGURE 2.5 – Ajout d'un nœud

Recherche :

Le nœud A cherche le fichier a.txt chez C et E (ce que A ne sait pas bien évidemment)

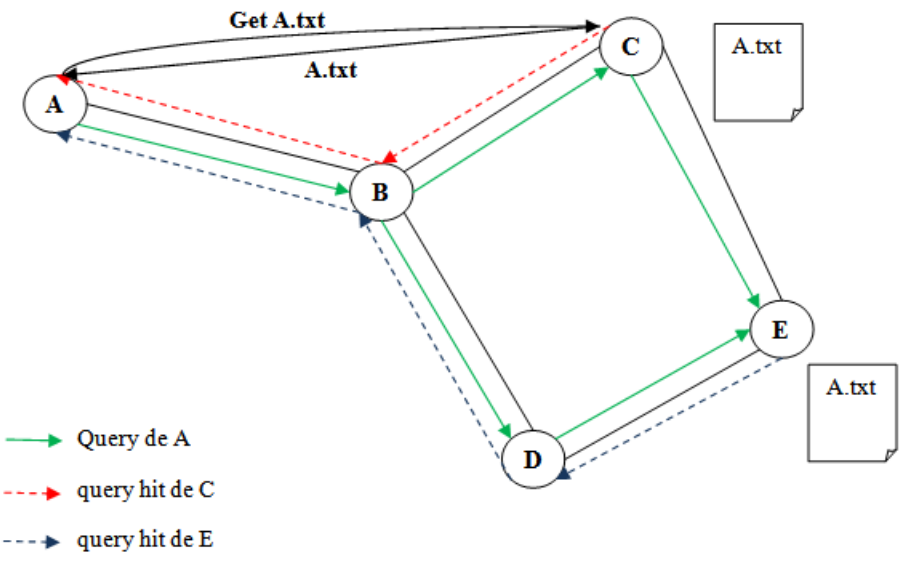

FIGURE 2.6 – Recherche

Le nœud A va envoyer un message query aux premiers nœuds qu'il connait, c.à.d. le nœud B.

Le nœud B se charge de :
- Répondre s'il dispose de la ressource (ce qui n'est pas le cas)
- De propager la requête.

Le nœud C dispose de la ressource donc :
- Il répond par un query hit
- Propage la requête au nœud E.

Le nœud C possède la ressource et répond par un query hit.
Le nœud A peut alors envoyer à C le message get A.txt et le transfert du fichier peut alors commencer.

Remarques :

Un nœud connait seulement une petite partie du réseau, ses voisins peuvent être alors imposés ou choisis.
La recherche se fait par brodcast (inondation)
Inondation bornée :
le nombre de transmission est limité dans la plupart des systèmes décentralisés.
TTL (Time To Live) est le nombre de retransmissions maximales. Dans GNUTELLA le TTL=7
Autre exemple FREENET : Est un P2P décentralisé dont le but est la diffusion de fichiers de façon totalement anonyme.
Les données ne sont pas téléchargées directement d'un pair à un autre mais peuvent transiter par un ou plusieurs pairs.
Impossible de déterminer l'origine exacte d'un message.

2.2.3 Architecture hybride : les supers peers

C'est un modèle hybride entre le mode client/serveur et le P2P.
Tous les nœuds ne sont pas égaux :
- Les nœuds disposant d'une bonne bande passante sont organisés en P2P, ceux sont les supers peers.
- Les nœuds avec une faible bande passante sont reliés en mode client/serveur à un super peer.
- Les supers peers disposent d'un index des ressources de leurs clusters.

Le logiciel P2Ple plus connu utilisant cette technologie est KAZAA.

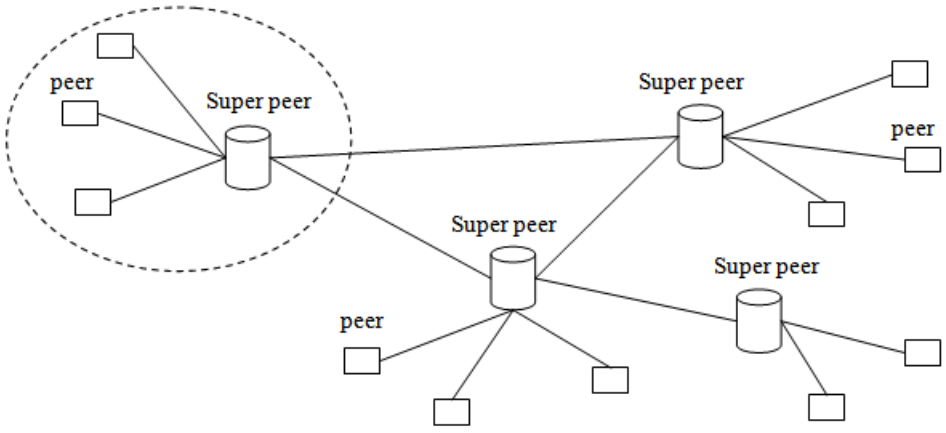

FIGURE 2.7 – KAZAA

- Un nouvel utilisateur doit ouvrir un compte et déclarer des répertoires de fichiers partagés.

— En fonction de sa bande passante, le logiciel décidera si le client est ou non un super peer.
— Lors d'une recherche, un nœud envoie sa requête à un super peer.
— Le super peer dispose d'un index de ses propres ressources, ainsi que celles des nœuds de son cluster.
— Si un super peer n'est plus joignable, tous ses clients sont coupés du réseau.
— Avoir K super peer au lieu d'un seul.
— Chaque super peer est relié à chaque client de son groupe et possède un index de toutes les ressources.

2.3 conclusion

après avoir définis les systèmes P2P, nous détaillerons les grilles informatiques.

Chapitre 3

LES GRILLES

3.1 Introduction

GRID « Globalisation des Ressources Informatiques et des Données » ou « **grilles informatique** » a été introduit pour la première fois aux U.S.A en 1990 pour décrire une infrastructure de calcul répartie utilisée dans des projets de recherche scientifiques et industrielles.

3.2 Définition

une grille informatique est une infrastructure virtuelle, constituée d'un ensemble coordonné de ressources informatiques potentiellement partagées, distribuées, hétérogènes, externalisées et sans administration centralisée.

3.3 Différence avec un cluster

- Un cluster (**grappe**) regroupe un ensemble de machines homogènes dédiées à faire une application donnée.
- Un cluster peut faire partie d'une grille.

le schéma suivant classe les systèmes distribués à grande échelle :

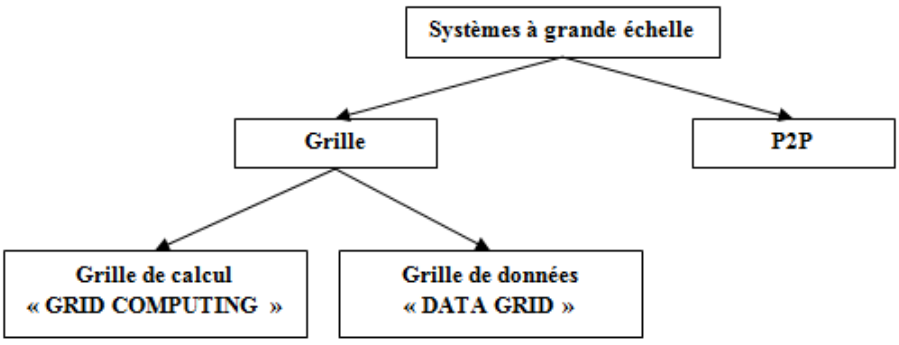

FIGURE 3.1 – Systèmes à large échelle

le tableau suivant compare les grilles et les systèmes P2P

	GRILLES	P2P
Données	**modifiables**	**non modifiables**
Architecture	**hiérarchique**	**plate**
Dynamicité	$\prec$	**forte**
Nombre de noeuds	10^3	10^6
Sécurité	**prise en charge de la sécurité**	**pas de sécurité**

TABLE 3.1 – comparaison entre grilles et P2P

Six questions fondamentales sont établies que toute personne impliquée dans la conception, le développement ou l'utilisation des grilles est susceptible de se poser :

1. Pourquoi avons-nous besoin de grilles ?
2. Quels types d'applications bénéficieront des grilles ?
3. Qui utilisera les grilles ?
4. Comment les grilles seront utilisées ?
5. Quelles sont les composantes impliquées dans la construction des grilles ?
6. Quels sont les obstacles à franchir afin que les grilles deviennent omniprésentes ?

3.4 Raisons des grilles

1. **Exploiter les ressources sous-utilisées :**
   — Les études montrent que les ordinateurs personnels et les stations web sont la plupart du temps inactifs le taux d'utilisation varie entre 30% (secteur académique et industriel) et 5% (machines du grand public).
   — Les grilles permettent d'utiliser les cycles processeurs durant lesquels les machines sont inactives. On peut aussi utiliser leurs capacités de stockage.
2. **Fournir une importante capacité de calcul parallèle.**
3. **Meilleure utilisation de certaines ressources :**
   — Une grille pourra fournir l'accès à des ressources spéciales, comme des équipements spécifiques (microscope électronique, bras robotique…) ou des logiciels dont le prix de la licence est élevé. Par conséquent, le matériel est mieux utilisé et partagé, ce qui nous évitera d'installer de nouveaux matériels et d'acheter de nouvelles licences.
4. **Fiabilité et disponibilité des ressources :**
   — En cas ou une ressources devient indisponible (tombe en panne, déconnexion du réseau…) on peut utiliser une autre.

3.5 Application des grilles

Les principales applications des grilles se feront dans les domaines suivants :

— **Supercalculateurs répartis (distributed supercomputing) :**
Une grille pourra agréger une importante quantité de ressources afin de fournir la puissance de calcul nécessaire pour de nombreuses applications que même les supercalculateurs les plus modernes ne peuvent fournir. Exemple : simulation distribuée dans la météorologie, la cosmologie et l'aéronautique.
— **Calcul haut débit (high throughout computing) :**
Une grille de calcul sera utilisée pour ordonnancer en parallèle une importante quantité de taches indépendantes les unes des autres Exemple : clés cryptographiques, simulation de modèles, analyse du génome.
— **Calcul à la demande (on demand computing) :**
Une grille de calcul pourra fournir des ressources pour satisfaire les demandes à court terme d'une application et que les ressources locales ne sont pas en mesure d'assurer (cycle processeur, stockage…)
— **Calcul collaboratif (collaborative computing) :**
cette classe d'applications inclut les applications d'interactions entre humains dans des environnements de simulation en temps réel (conception et interaction) .
exemple : conception d'un nouveau moteur d'avion.
— **Génération, traitement et stockage d'énormes quantités de données (data-intensive computing) :**
Exemple : génération d'une carte de l'univers, les prévisions météorologiques à long terme, les simulations quantiques...

3.6 caractéristiques des grilles

Les grilles possèdent quatre principales caractéristiques :

1. **Existence de plusieurs domaines administratifs :** Les ressources sont géographiquement distribuées et appartiennent à différentes organisations chacune ayant ses propres politiques de gestion et de sécurité. Il est indispensable de respecter les politiques de chacune de ces organisations.
2. **Hétérogénéité des ressources :** Les ressources sont hétérogènes en termes de matériels et de logiciels.
3. **Passage à l'échelle (scalability) :** Une grille est constituée de quelques dizaines de ressources à des millions ou des dizaines de millions, ce qui pose de nouvelles contraintes sur les applications et les algorithmes de gestion des ressources.
4. **Nature dynamique des ressources :** La dynamicité est l'une des caractéristiques principales des grilles. Ceci nécessite l'adaptation au changement dynamique du nombre de ressources, la tolérance aux pannes et aux délais.

Les ressources

les ressources considérées dans une grille sont :
Cycle processeur : plusieurs façons d'exploiter des processeurs non utilisés des machines de la grille
— Faire tourner une application sur une machine distante au lieu du processeur local.
— Utiliser une application conçue pour tourner sur différentes parties en parallèle sur différents processeurs.

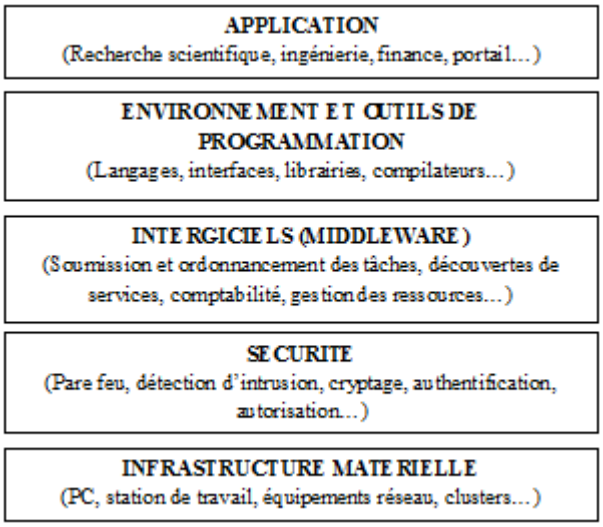

FIGURE 3.2 – Architecture générale d'une grille

— Faire tourner plusieurs occurrences d'une même application sur différentes machines afin de traiter différentes données plus rapidement.

Capacité de stockage : les machines de la grille peuvent fournir une partie de la capacité de stockage nécessaire au fonctionnement de la grille.

Equipements spéciaux et logiciels à licence élevée : certains logiciels dont le prix de la licence est élevé seront présents en quelques exemplaires seulement dans la grille et pourront ainsi être utilisés par beaucoup d'utilisateurs de la grille. De plus certains équipements spécifiques comme les microscopes électroniques et d'autres appareils médicaux peuvent aussi être utilisés dans la grille.

3.7 Architecture générale d'une grille

Chaque projet a sa propre architecture. Une architecture générale est importante pour expliquer certains concepts fondamentaux des grilles.

Intergiciel (middleware)

nécessaire à la gestion des ressources, la coordination de l'accès, l'ordonnancement des tâches...
Les principales fonctions sont :

— **Ordonnancement (scheduling) :** l'ordonnanceur devra trouver la machine la plus appropriée pour faire tourner les tâches qui lui sont soumises. Il tient compte de la charge de la grille. Il détermine les taux de charge de chaque ressources afin de bien ordonnancer les prochaines tâches.

— **Réservations :** Il est possible dans les grilles de réserver les ressources à l'avance et ceci afin de garantir certaines échéances. L'intergiciel devra fournir un système de réservation permettant aux utilisateurs d'exprimer leurs besoins en ressources en d'effectuer la réservation.

— **Service d'informations et d'annuaire (information and directory services) :** la grille est un environnement où le nombre et la charge des ressources sont dynamiques. L'objectif de la conception d'une grille est de rendre les ressources facilement accessibles à tous les processeurs. Il

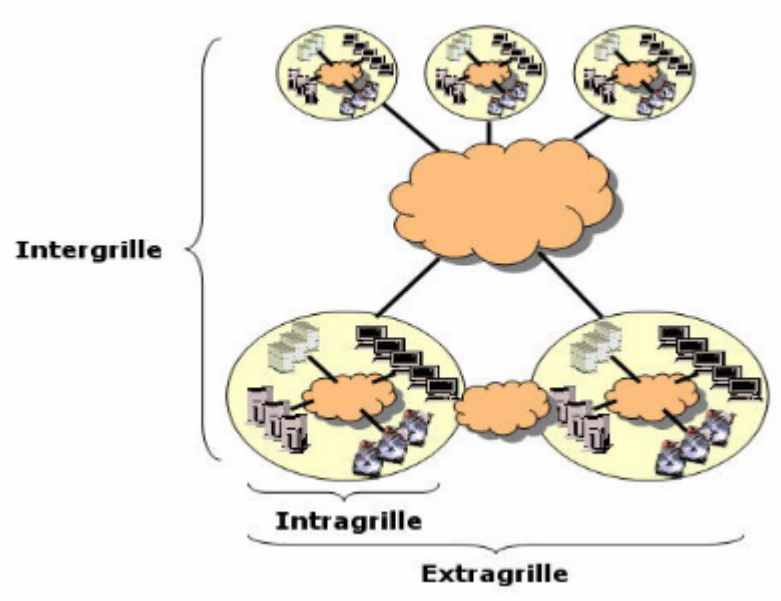

FIGURE 3.3 – Topologies des grilles

est nécessaire de fournir des mécanismes permettant d'enregistrer et de découvrir ces ressources et d'identifier leur état.

— **Service de nom (naming service) :** une grille devra permettre de référencer ses ressources d'une façon standard et uniforme.

3.8 Différentes topologies des grilles

On peut répartir les grilles de la façon suivante : intragrille, extragrille,intergrilles.

3.8.1 Intragrille (intragrids)

composée d'un ensemble relativement simple de ressources et de services appartenant à une organisation unique. elle est caractérisée par :

— un réseau d'interconnexion performant et haut débit.
— un domaine de sécurité unique et maîtrisé par les administrateurs.
— un ensemble statique et homogène des ressources.
— **Exemple :** l'entreprise AMD pour la phase de conception de ses processeurs K6 et K7.

3.8.2 Extragrille (extragrids)

agrège plusieurs intragrilles, elle est caractérisée par :

— un réseau d'interconnexion hétérogène haut et bas débit.
— Plusieurs domaines de sécurité distincts.
— un ensemble plus ou moins dynamique de ressources.
— **Exemple :** alliances et les échanges entre entreprises partenaires B2B (business to business)

3.8.3 Intergrille (intergrids)

agrège plusieurs grilles de multiples organisations en une seule grille

- réseaux d'interconnexion très hétérogène haut et bas débit.
- Plusieurs domaines de sécurité distincts et ayant parfois des politiques de sécurité différentes et même contradictoire.
- Ensemble très dynamique de ressources.
- **Exemple :** grands projets industriels.

3.8.4 Exemples de grilles

plusieurs grilles ont vu le jour. on cite les plus connues :
USA : La plupart des organisations gouvernementales et scientifiques disposent de leurs grilles telles que GRIPHYN pour la recherche nucléaire au niveau de la NASA.
EUROPE :

- **DATAGRID** : Ce projet vise à mettre à disposition des chercheurs un outil informatique en leur offrant la puissance nécessaire au traitement des quelques milliers de téraoctets des données qui seront collectées chaque année.
- EUROGRID
- CROSSGRID
- GRID START
- DAMIEN

FRANCE :
eToile
GRANDE BRETAGNE :

- GRIDpp : est une communauté de physiciens et d'informaticiens basée au Royaume-Uni et au Cern (Conseil européen pour la recherche nucléaire).
- Datatag : construite entre l'europe et l'amérique du nord.

Différents intergiciels pour grilles :

plusieurs intergiciels pour grilles ont été developpés. on cite dans ce qui suit quelques uns :

- GLOBUS (open source)
- UNICORE (allemand)
- NAREGI (japan)
- LEGION (université de virginie)
- GEDEON, CONDOR, GLITE, SRB
- Legion Global grid forum : composé de laboratoires de recherche informatique publics, d'universités privées, d'agences gouvernementales et d'entreprises privées. Le but étant de recenser les besoins des utilisateurs des technologies des grilles et sur la manière de les implanter. Les recherches ont abouti aux GMA (Grid Monitoring Architecture) qui est un système de surveillance pour les grilles et OGSA (Open Grid Service Information) qui représente une architecture et non pas une implémentation ou des outils particuliers de construction de grilles.

3.9 conclusion

dans le chapitre suivant les différentes arcitectures qui existent pour les grilles seront abordées.

Chapitre 4

GESTION DES RESSOURCES DANS LES GRILLES

4.1 introduction

afin de comprendre le fonctionnement des grilles il est important de savoir sur quelle architecture repose ce type d'infrastructure pour gérer tous les échanges entre les ressources.

4.2 Modèles architecturaux

On peut identifier plusieurs architectures susceptibles d'offrir un bon modèle architectural pour la gestion des ressources on peut citer les plus connus :

- Le modèle hiérarchique.
- Le modèle du propriétaire abstrait (abstract owner model).
- Le modèle de l'économie de marché (computational market economy model) : Ces modèles encouragent les fournisseurs à contribuer à la grille, assurer un partage juste de ces ressources entre utilisateurs et réguler efficacement l'offre et la demande.

4.3 Le modèle hiérarchique

On trouve ce modèle dans la plupart des intergiciels pour grille tels que : Globus et Legion.
Fonctionnellement, cette architecture est divisée en un ensemble de composants passifs et actifs :

- Composants passifs : des ressources, des tâches qui doivent s'exécuter et les ordonnanceurs de ces tâches sur les ressources.
- Composants actifs : les ordonnanceurs (schedulers), les services d'information (information service), les contrôleurs de domaines (domain control agents), les agents de déploiement (deployment agents), les agents de contrôle d'admission (admission control agents), les superviseurs (monitors), les agents de contrôle des taches (job control agents).

Les modèles hiérarchiques facilitent l'interaction entre l'utilisateur et le système pour la réservation des ressources. L'utilisateur soumet ses tâches à un agent de contrôle des tâches, ce dernier interroge un agent de contrôle d'admission puis consulte les services d'information pour connaitre l'état des ressources et pouvoir prendre la décision d'autoriser l'admission dans le système.

Dans le cas où les tâches sont admises, elles sont soumises aux ordonnanceurs de haut niveau qui se chargent de contacter les services d'information et les ordonnanceurs de bas niveau afin de localiser les ressources adéquates. Les agents de déploiement s'occupent alors de contacter les contrôleurs de domaine afin d'autoriser le déploiement des tâches sur les ressources.

4.4 Le modèle du propriétaire abstrait (abstract owner model)

Analogie avec le système téléphonique : l'entité avec laquelle l'utilisateur négocie l'établissement de l'appel n'est parfois pas le propriétaire des ressources (ex : appels internationaux). L'utilisateur traite cette entité abstraitement comme étant le propriétaire de ces ressources.
Le plus souvent l'entité que l'utilisateur contrôle n'est qu'un courtier représentant les vrais fournisseurs. Cette entité est appelée Abstract Owner AO. Un AO est une boite noire munie d'une entrée et d'une sortie. L'entrée permet à l'utilisateur d'exprimer ses besoins en ressources et la sortie de récupérer les ressources qui lui seront allouées.

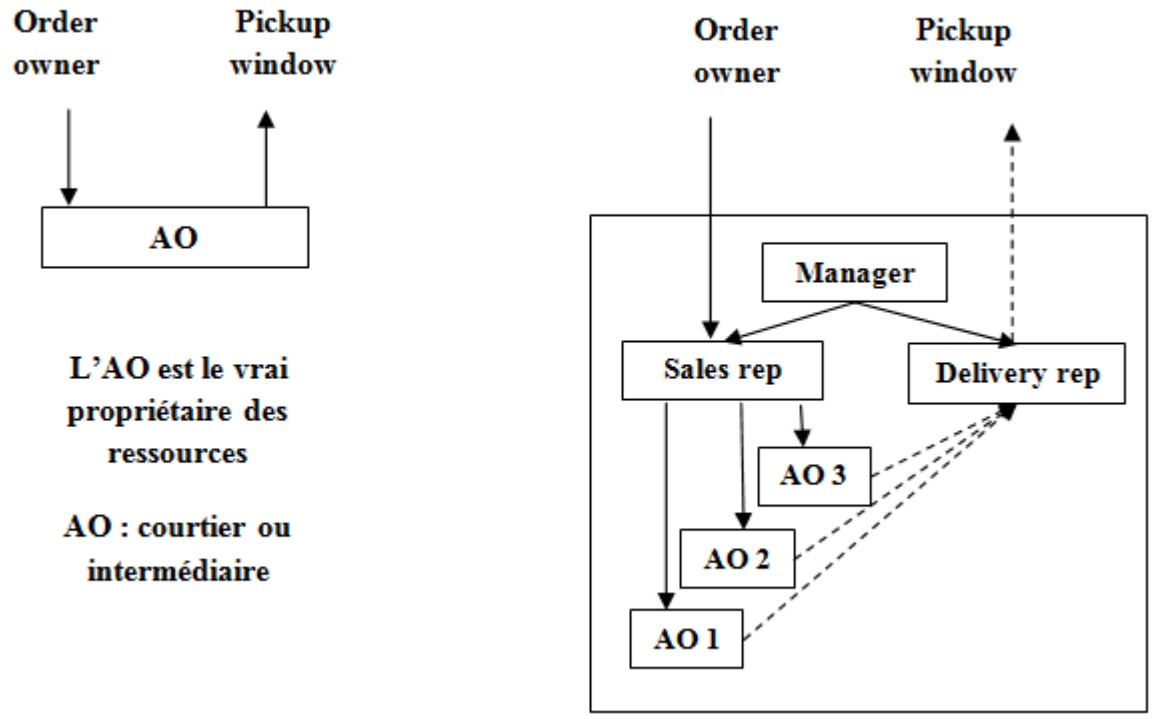

FIGURE 4.1 – modèle du Abstract Owner

4.5 Le modèle de l'économie de marché (computational market economy model)

Les arguments en faveur d'un modèle d'économie de marché pour la gestion des ressources d'une grille sont nombreux. Ce modèle incite les fournisseurs à contribuer leurs ressources car il leur offre la possibilité de tirer des profits (souvent financier) de l'utilisateur qui en sera faite. L'économie est le meilleur

régulateur de l'offre et de la demande. Les utilisateurs souhaitent utiliser les ressources en minimisant leur coût tandis que les fournisseurs souhaitent maximiser leur retour sur investissement.

4.5.1 Modèle économique pour la gestion des ressources d'une grille de calcul

Une grille utilisant le modèle économique pour la gestion des ressources doit comporter les éléments suivants :

- Un annuaire d'informations répertoriant les différentes entités de la grille (différentes ressources).
- Des modèles permettant d'établir la valeur de la ressource.
- Des protocoles de négociation entre producteurs et consommateurs.
- Des agents intermédiaires, agissant en tant qu'agents de régulation, chargés d'établir la valeur des ressources à partir de modèles de prix, de gérer les échanges monétaires, et de gérer au mieux les situations.
- Des mécanismes de facturation et de paiements.

Des systèmes d'ordonnancement garantissant le respect de la QoS demandée. Pour répondre à ces spécifications, plusieurs entités doivent être mises en place :

- Un courtier « Grid Ressource Broker » GRB dont le rôle est de faire :
  - La découverte et la sélection des ressources.
  - Initier les exécutions sur les ressources distantes.
  - Retourner les résultats aux utilisateurs.
  - Présenter la grille à l'utilisateur comme étant une seule ressource unifiée.
- Un annuaire « Grid Market Directory » GMD Qui contient les informations sur les différentes entités de la grille.
- Un serveur d'échange « Grid Trade Server » GTS
- Un système de comptabilité.

Les différents modèles économiques existants :

- Modèle de marché.
- Modèle de négociation.
- Modèle d'appel d'offre.
- Modèle aux enchères.
- Modèle du partage proportionnel des ressources.

4.5.1.1 Modèle de marché :

Les fournisseurs de ressources fixent un prix et les utilisateurs sont facturés à ce prix, proportionnellement à la quantité de ressources consommées.

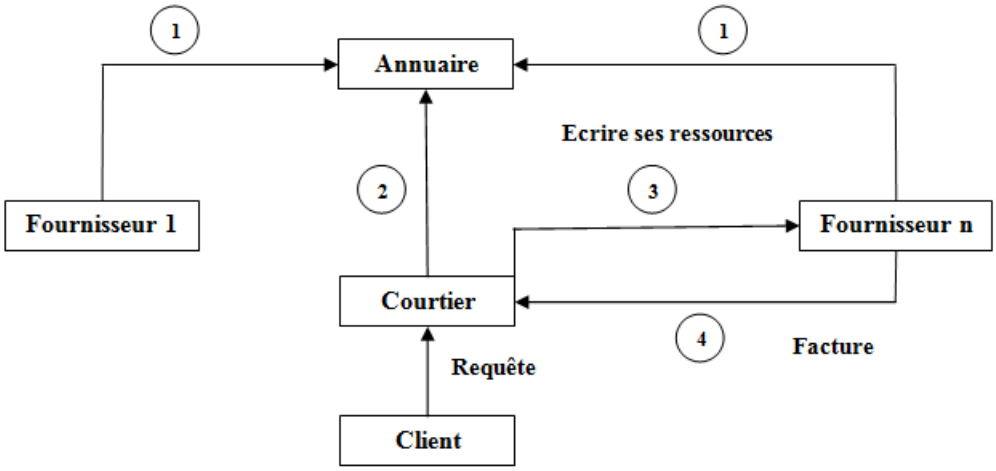

FIGURE 4.2 – Modèle de Marché

1. Chaque fournisseur fixe un prix pour ses ressources et le publie auprès de l'annuaire.
2. Le courtier consulte l'annuaire pour avoir la liste des fournisseurs qui répondent à la demande avec les prix.
3. Le courtier identifie le fournisseur qui satisfait au mieux les objectifs du client et lui confie l'application à exécuter.
4. Le fournisseur facture au client selon la quantité de ressources consommées.

4.5.1.2 Modèle de négociation

Ce modèle permet à l'utilisateur, contrairement au précédent, d'agir sur le prix des ressources qu'il désire consommer.

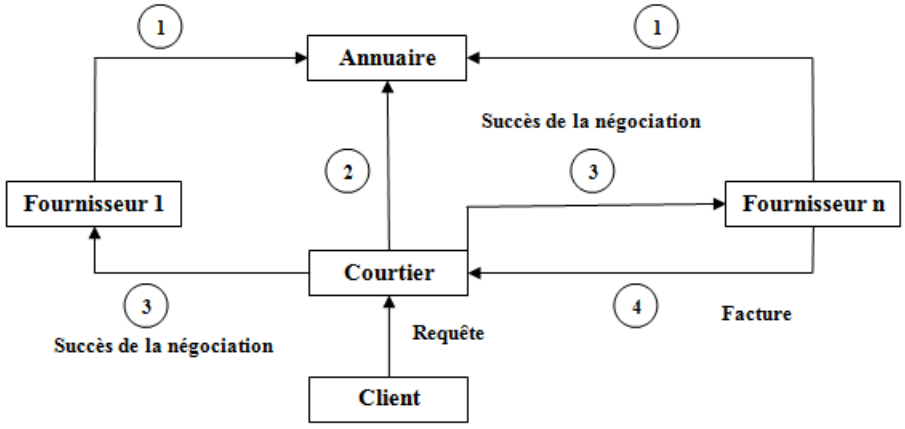

FIGURE 4.3 – Modèle de négociation

1. Chaque fournisseur s'enregistre auprès de l'annuaire.
2. Le courtier contacte l'annuaire pour obtenir la liste des fournisseurs.
3. Le courtier négocie avec le fournisseur pour obtenir un prix acceptable, le courtier commençant avec un prix bas et le fournisseur avec un prix haut.

4. Si les deux parties se mettent d'accord sur le prix, l'application est confiée au fournisseur qui pourra ensuite facturer le client. Si aucun terrain d'entente ne peut être trouvé, le courtier s'adresse à un autre fournisseur et entame une nouvelle négociation.

4.5.1.3 Modèle d'appel d'offre

Ce modèle permet au client de trouver le fournisseur de service approprié pour travailler sur une tâche donnée. Le client (ou le courtier qui le présente) est appelé manager tandis que chaque fournisseur est appelé un contractant potentiel.

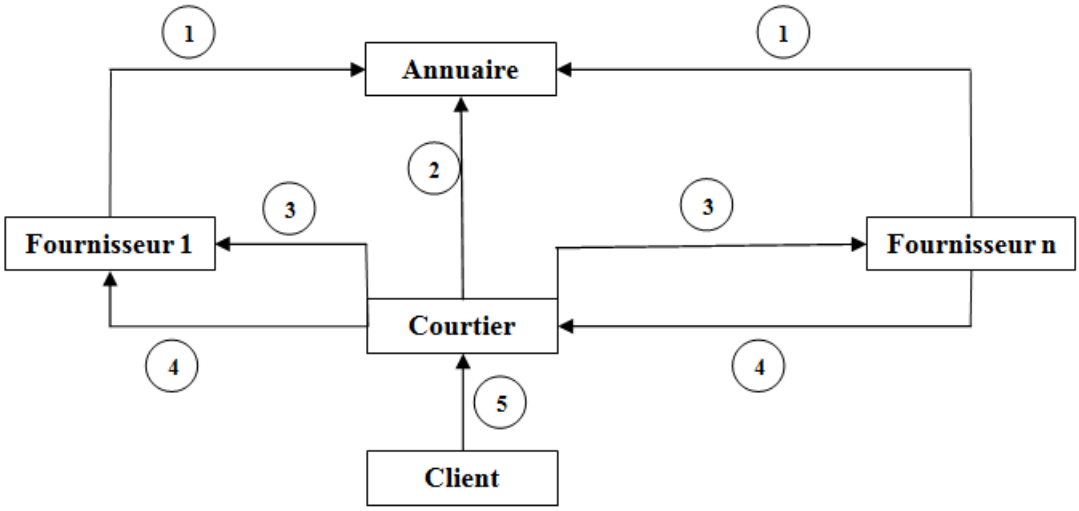

FIGURE 4.4 – Modèle de l'appel d'offre

1. Chaque fournisseur s'enregistre auprès de l'annuaire.
2. Le courtier contacte l'annuaire pour obtenir la liste des fournisseurs (contractants potentiels).
3. Le courtier lance un appel d'offre auprès des contractants potentiels, en leur fournissant les besoins.
4. Chaque contractant potentiel l'évalue et lui soumet une offre s'il est intéressé.
5. Le courtier choisi la meilleure offre et lui attribue un contrat.

L'appel d'offre se produit devant le public et en présence d'une personne qui représente la loi, son rôle est de décacheter les enveloppes qui contiennent les offres préalablement scellées.

4.5.1.4 Modèle de vente aux enchères

Le rôle du commissaire priseur est de veiller sur le bon déroulement des enchères.

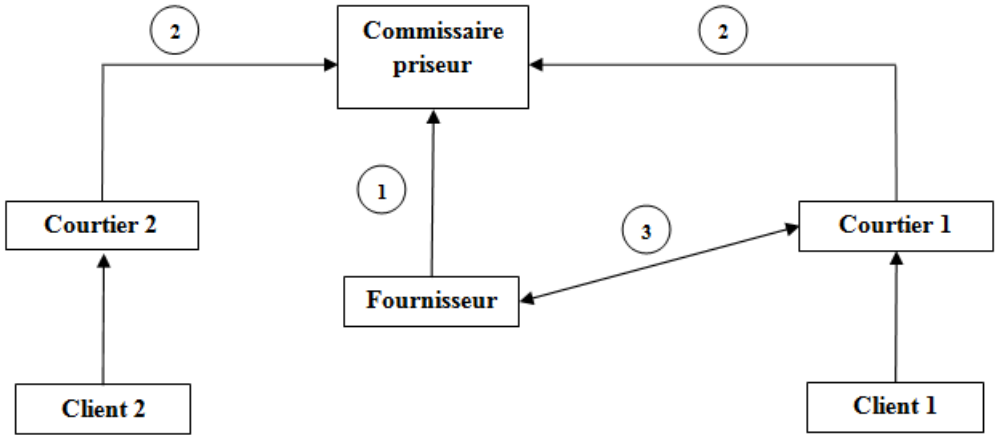

FIGURE 4.5 – Modèle de vente aux enchères

1. Le fournisseur publie ses services auprès de commissaire priseur et invite les utilisateurs à faire des offres.
2. Les différents courtiers font des offres et peuvent les modifier.
3. L'enchère s'arrête lorsque plus aucun courtier ne modifie l'offre.

4.5.1.5 Modèle de partage proportionnel des ressources

Une ressource peut être partagée par plusieurs utilisateurs. La part de chaque ressource allouée à chaque utilisateur par rapport aux autres utilisateurs est alors proportionnelle à son offre par rapport à celles des autres utilisateurs. Ceci permet à tous les utilisateurs d'avoir accès aux ressources, même s'ils n'ont pas un budget important par rapport aux autres utilisateurs.
Etablissement des prix et mécanismes de paiement : Parmi les ressources à facturer, on peut citer :

- Le temps processus consommé.
- La quantité de mémoire consommée.
- La quantité de stockage utilisée.
- La bande passante réseau consommée.
- Les logiciels et les librairies utilisés et les changements de contexte, par exemple : simulateur windows $\neq$ simulateur linux.

Le prix d'une ressource dépend des paramètres suivants

- La durée d'utilisation.
- Le moment d'utilisation.
- La fidélité de l'utilisateur.
- L'offre et la demande.
- La puissance de la ressource.
- Son coût physique.
- La surcharge du service.
- L'existence d'un précontrat entre l'utilisateur et le fournisseur.
- Les préférences locales du propriétaire.
- La valeur perçue par l'utilisateur.

Le paiement pour l'utilisation des ressources peut s'effectuer à plusieurs moments :

— Le paiement à l'avance : l'utilisateur achète au préalable des crédits qu'il peut ensuite dépenser pour utiliser les ressources.
— Le paiement après utilisation : le client est facturé conformément à sa consommation de ressources.
— Le paiement au fur et à mesure de la consommation.

4.6 conclusion

plusieurs traitements peuvent tirer profit des avantages fournis par les grilles tels que la réplication. la réplication est une technique trés utilisée dans les grille afin de garantir la disponibilité des données. cette solution est détaillée le chapitre suivant.

Chapitre 5

REPLICATION

5.1 Introduction

la réplication est une des techniques les plus utilisées dans les grilles. le but est de créer plusieurs copies sur différents sites afin d'augmenter la disponibilité des données. dans ce chapitre les différentes méthodes de réplications seront détaillées.

5.2 Définition

La réplication permet de faire une copie parfaite (ou conforme) d'un objet ou d'un fragment d'objet (fichiers, bdd) d'un site vers un autre.
Objectif principal de la réplication est la sureté de fonctionnement.

5.3 Avantages

- Une grande disponibilité des données.
- Une meilleure résistance aux pannes.
- Réduction des goulots d'étranglement.
- Equilibrage des charges des nœuds.
- Temps de réponse meilleur (bonnes performances).
- Permettant aux terminaux de travailler normalement même s'ils se déconnectent du système.

5.4 Inconvénients

- Choisir les objets à répliquer (faire des efforts pour sélectionner par des algorithmes ou faire un historique pour savoir quelles sont les données les plus utilisées).
- Degrés de la réplication (nombre de répliques).
- Placement des répliques (ou mettre les répliques ?)
- A quel moment répliquer (de manière statique ou de manière dynamique au fur et à mesure de l'exécution).
- Gestion de la cohérence (la modification d'une réplique doit être la même pour les autres répliques).

5.5 Caractéristiques des protocoles de réplication dans les systèmes distribués

certaines caractéristiques doivent être prises en considération lorsqu'on fait de la réplication :

5.5.1 nombre de noeuds concernés par une lecture et une écriture

Approche1 : (ROWA : Read One Write All)

écriture sur toutes les copies avant de valider une requête externe d'écriture.
consultation d'une seule copie pour répondre à une requête externe de lecture.

Approche2 :

le protocole valide une requête externe d'écriture lorsqu'on a (N/2+1) copies qui sont mises à jour (N : nombre de copies).
lors d'une requête externe de lecture, il est alors nécessaire de consulter N/2 copies pour répondre aux demandeurs.

5.5.2 droits d'accès aux copies

Approche1 : maitre /esclaves

chaque objet répliqué possède une copie dite maîtresse, les autres esclaves.
une requête de type MAJ ne peut être faite que sur le maître, celui-ci diffuse ensuite les modifications aux esclaves.

Approche2 :

copies identiques (droits de MAJ est le même aux différentes copies).

remarque :

l'approche 1 est centralisée donc facilement maitrisable, par contre la deuxième approche est moins maitrisable car elle est décentralisée.

5.5.3 moment de la synchronisation entre copies

- période
- condition sur le délai
- condition sur les versions (nombre de MAJ)
- condition numérique : valeur (modulo) ou comparaison (entre l'ancienne et la nouvelle MAJ)
- condition sur les évènements
- condition sur les objets

5.5.4 initiative de la MAJ

Approche 1 : c'est la copie source qui va propager les MAJ aux autres copies. appelée aussi rafraîchissement de type « PUSH ».
Approche 2 : les copies cibles demandent les MAJ à une (aux) copie(s) source(s). Appelée aussi rafraîchissement de type « PULL ».

5.5.5 nature de la MAJ

la nature des MAJ désigne le type d'information envoyée aux différentes copies lors de la synchronisation. deux types d'informations :

- l'état des copies nouvelles (copies MAJ)
- les opérations de MAJ

5.5.6 topographie de la synchronisation

la copie qui doit propager une MAJ, le fait par diffusion à toutes les copies.
d'autres protocoles propagent les MAJ de copie en copie vers un ensemble de copies (groupes de copies) où chacune d'elles les propage à un autre ensemble.

5.5.7 la capture de la MAJ

le mécanisme utilisé pour détecter et sélectionner les changements sur une copie afin de les propager aux autres copies est appelé « capture »
trois approches :
Approche1 : consulter la copie afin de connaître son dernier état.
Approche2 : consiste à enregistrer les modifications sur un support particulier (exemple : journal).
Approche3 : consiste à déclencher un mécanisme particulier (thread,..)

5.5.8 gestion des conflits

dans certains protocoles deux copies peuvent être modifiées de manière concurrente, dans ce cas le protocole se trouve face à un conflit :

1. détection d'un conflit,
2. réconciliation du conflit : une fois le conflit détecté, deux méthodes de réconciliation peuvent être envisagées :

- automatique (date récente, version des copies, copie la plus populaire...).
- manuelle.

5.5.9 gestion des fautes

trois types de fautes :

- fautes par arrêt : noeud déconnecté de manière volontaire ou non, en panne,..
- fautes valeur : une valeur hors intervalle de données/ une valeur temporelle anormale..
- fautes byzantines : (panne byzantine ou comportement byzantin) tout comportement d'un système ne respectant pas ces spécifications en donnant des résultats non conformes.

5.5.10 notion de copie

- combien de copies dans le système sont nécessaires ?
- ou les mettre ?
- à quel moment on crée de nouvelles copies (allocation statique ou dynamique ?)

5.5.11 transparence à la réplication

un objet est une abstraction représentant un ensemble de copies d'un même objet répliqué. cette abstraction permet de référencer l'ensemble des copies (objet physique).

5.6 La réplication dans les grilles

de manière générale, on trouve deux types de réplication : statique et dynamique.
la réplication statique consiste à créer en amont des répliques sur les noeuds bien défnis. contrairement à la réplication statique, dans la réplication dynamique les répliques sont placées sur des noeuds au fur et à mesure de leur évolution.
Dans une grille de données, où le nombre des utilisateurs et des données utilisées est volumineux, la réplique statique n'est pas très utile. On a besoin de stratégies de réplication dynamiques, où la création, la suppression et la gestion de répliques sont faites automatiquement et les stratégies ont la capacité de s'adapter aux changements du comportement des utilisateurs.
Trois questions fondamentales doivent être posées pour n'importe quelle stratégie de placement de répliques :

- Quand les répliques devraient elles être créées ?
- Quels fichiers devraient être répliqués ?
- Où doivent être placées les répliques ?

5.6.1 Evaluation des performances des stratégies de réplication :

La comparaison des différentes stratégies de réplication se fait en mesurant le temps de réponse moyen et de toute la bande passante consommée.
Temps de réponse : c'est le temps qui s'écoule entre le moment où un nœud envoie une demande d'une donnée et le moment où il la reçoit (si une copie locale de la donnée existe, le temps de réponse est considéré comme nul).
Bande passante : c'est la largeur de la bande consommée pour des transferts de données entre le nœud qui demande un dossier et le serveur qui contient une réplique de ce dossier.

5.6.2 quelques approches de réplication

On doit tenir compte de certaines notions :

- **Topologie de la grille :** comment sont arrangés les nœuds dans la grille.
- **Localisation de la réplique la plus proche** : en général la réplique la plus proche est localisée selon un nombre de sauts le plus bas possible, s'il en existe plusieurs une raplique est choisie aléatoirement.

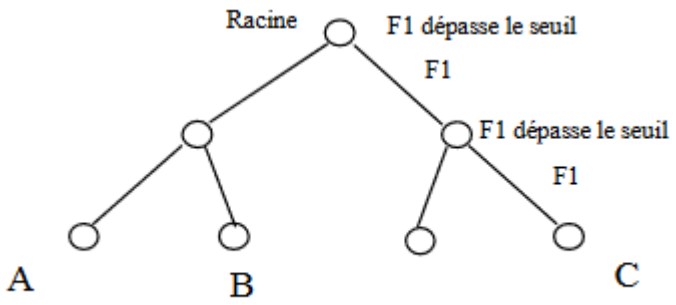

FIGURE 5.1 – réplication en cascade

Plusieurs travaux existent, par exemple dans un environnement de grille hiérarchique différentes stratégies ont été développées. le but de ces stratégies est d'optimiser le nombre de répliques, de diminuer le temps de réponse et de maintien de la cohérence.

1. **Meilleur client :** chaque nœud maintient un historique détaillé pour chaque dossier qu'il détient et il indique le nombre de requêtes qui concernent ce dossier et les nœuds d'où viennent ces requêtes. A un moment donné, chaque nœud vérifie si le nombre de requêtes sur l'un des dossiers qu'il contient a dépassé un seuil. Si oui le meilleur client pour ce dossier(le plus grand pourcentage de requêtes) est identifié et le nœud crée alors une réplique auprès de ce nœud.
2. **Diffusion rapide :** une réplique d'un dossier est stockée à chaque nœud tout au long du chemin jusqu'au client.
3. **Répliques en cascade :** chaque noeud stocke localement et périodiquement les informations de chaque fichier afin d'identifier ceux qui sont à usage fréquent pour propager une réplique vers le noeud déscendant. autrement dit, une fois le seuil pour un fichier est dépassé à la racine, une réplique est créée au prochain niveau, sur le chemin du meilleur client. Une fois le seuil pour le fichier est dépassé au niveau 2 alors il sera répliqué à la prochaine rangée et ainsi de suite.
4. **Approche économique :**

   Dans cette approche le modèle économique (selon l'offre et la demande) est adopté.
   - Un agent est situé sur chaque nœud.
   - Lorsqu'une donnée est demandée sur un site, l'agent concerné va interroger les serveurs de stockage.
   - Le serveur qui remporte l'enchère est celui qui a proposé le prix le plus bas.

   Le prix est calculé de la façon suivante pour chaque serveur interrogé :
   - Si la donnée est présente dans le serveur, alors le prix fixé est proportionnel au temps estimé pour le transfert du fichier entre le serveur de stockage considéré et le site demandeur.
   - Sinon si la donnée n'est pas présente, le serveur de stockage a la possibilité de déclencher lui aussi une demande d'enchères pour acquérir la donnée s'il estime que les revenus qu'elle va lui apporter seront plus grands que le cout de son achat.
5. **Stratégie de placement basée sur les données populaires :**

   Cette approche est basée sur un algorithme de placement de répliques dynamique dans des grilles de données hiérarchiques basé sur des fichiers populaires. L'objectif principal de la stratégie est de réduire le temps d'accès aux données par la création dynamique des répliques pour les fichiers populaires afin de les maintenir le plus longtemps prés des nœuds demandeurs.

   Le processus de réplication est fait en deux phases :

— **Agrégation d'accès de bas en haut** : un historique des demandes pour chaque fichier est enregistré au niveau des clients. A chaque niveau, tout nœud parent additionne le nombre de demandes d'accès enregistrés par ses nœuds fils pour chaque fichier. Le résultat de l'agrégation est stocké dans le nœud parent.
— **Placement des répliques de haut en bas** : On utilise les informations agrégées. Un seuil est défini.
On place les répliques de la façon suivante : En parcourant l'arbre de la racine vers les feuilles : lorsqu'un nœud a un nombre d'accès agrégé supérieur aou égal au seuil alors une réplique est placée sur ce nœud si le nombre d'accès agrégé d'un ou de plusieurs de ses enfants est inférieur au seuil.

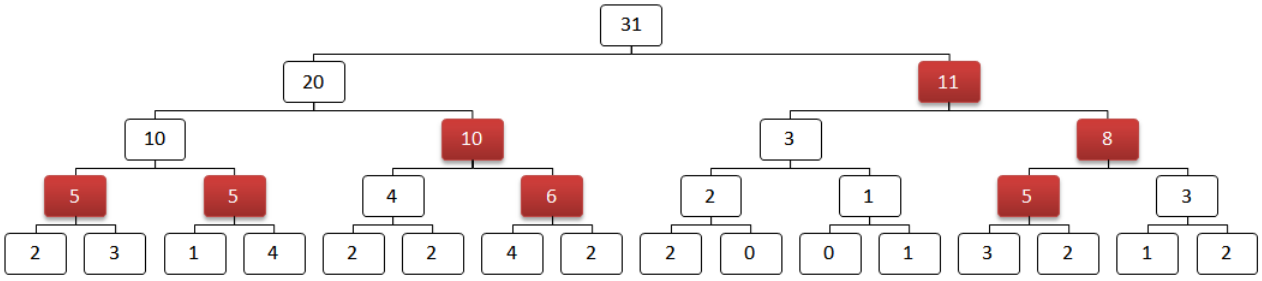

FIGURE 5.2 – **Stratégie de placement basée sur les données populaires**

5.7 conclusion

aprés avoir décrit les algorithmes de réplication qui existent, nous passerons dans le prochain chapitre aux problème engendrés par cette technique, càd la cohérence.

Chapitre 6

COHERENCE

6.1 Introduction

Bien que très avantageuse, la réplication présente tout de même un inconvénient majeur de cohérence des données où toutes les copies d'une même donnée doivent être identiques. Dans un système à large échelle, tel que les grille, il arrive que des copies d'une même donnée ne soient pas identiques. Pour cela des mécanismes doivent être mis en place pour assurer que les copies qui divergent à un moment donné, finissent toutes par converger vers le même état. De plus, la propagation des mises à jour d'une copie vers toutes les autres copies peut avoir parfois un impact négatif sur les performances du système, celle-ci ne doit pas altérer de façon excessive le temps de réponse aux requêtes des utilisateurs

6.2 Définition

La cohérence est une relation qui définit le degré de similitude entre les copies.

6.3 Modèles de cohérence

Les modèles de cohérence sont caractérisés par un ensemble de protocoles de gestion de la cohérence entre les répliques des données. Nous distinguons deux grandes familles : les modèles à cohérence forte et les modèles à cohérence relâchée (ou faible).
La cohérence forte garantie que toutes les copies soient identiques mais cette solution reste très coûteuse à mettre en œuvre dans les systèmes à grande échelle. Contrairement à la cohérence forte, les modèles de la cohérence relâchée tolèrent une certaine incohérence entre les répliques pour un certain moment afin d'alléger le système en terme de performances d'accès.

La mise en place de protocoles de la cohérence stricte dans un système réparti pose deux problèmes. Définition de « l'événement le plus récent » et la Réalisation « instantanée » des opérations. La définition de l'événement le plus récent nécessite des horloges parfaitement synchronisées (écart nul entre deux sites quelconques). Une opération qui nécessite un accès distant ne peut être instantanée (plus précisément, une opération locale lancée après une opération distante peut se terminer avant) La cohérence stricte est un modèle idéal (non réalisable), que l'on essaie d'approcher au moyen de modèles moins contraignants, à savoir des modèles de cohérence relâchée.

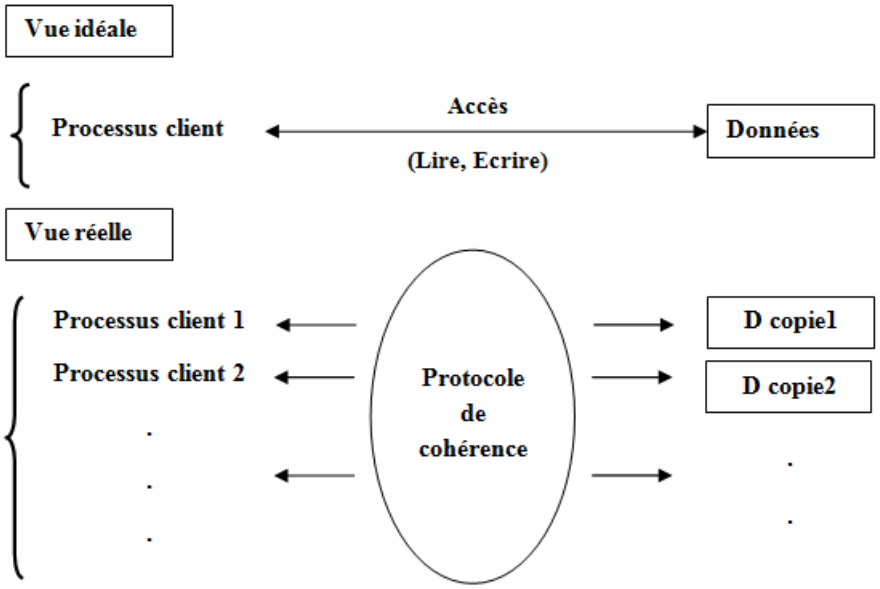

FIGURE 6.1 – protocole de cohérence

6.4 Modèles de cohérence forte (sans synchronisation)

Caractérisés par des contraintes fortes entre la dernière écriture et la prochaine lecture. Seules les primitives de lecture et d'écriture sont utilisées.

6.4.1 Modèle de cohérence stricte (cohérence atomique)

correspond à la vue idéale Toute lecture sur un élément X renvoie une valeur correspondante à l'écriture la plus récente sur X. Seulement comment est défini l'événement le plus récent ? Ce modèle est tellement idéal qu'il est irréalisable.

6.4.2 Modèle linéarisable

Le résultat de l'exécution d'un ensemble de processus clients est identique à celui d'une exécution dans laquelle :
• Toutes les opérations sur les données (vues comme centralisées) sont exécutées selon une certaine séquence S
• Si deux opérations op1 et op2 sont considérées telles que op1 précède op2, alors op1 et op2 figurent dans cet ordre dans S
• La cohérence interne des données est respectée dans S (après une écriture, et jusqu'à la suivante, une lecture délivre la valeur écrite)
Autre définition : Le résultat de l'exécution d'un ensemble de processus clients est identique à celui d'une exécution dans laquelle toutes les opérations sur les données (vues comme centralisées) sont exécutées selon une certaine séquence.

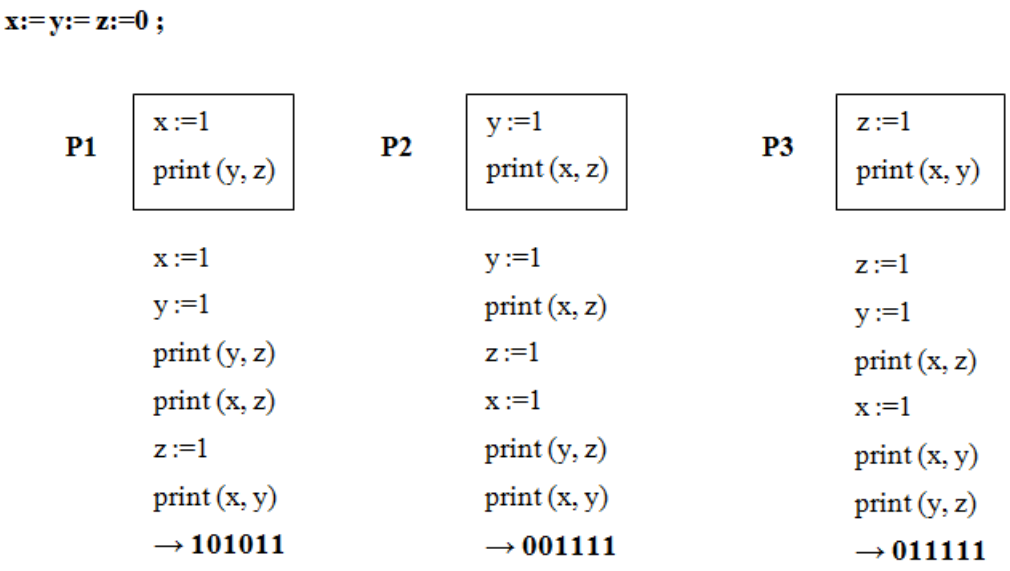

FIGURE 6.2 – Modèle de cohérence séquentiel

6.4.3 Modèle de cohérence séquentiel

le résultat de l'exécution d'un ensemble de processus clients est identique à celui d'une exécution dans laquelle :• Toutes les opérations sur les données (supposées centralisées) sont exécutées selon une certaine séquence S• Les opérations exécutées par tout processus p figurent dans S dans le même ordre que dans p• La cohérence interne des données est respectée dans S
Autre définition : Le résultat de l'exécution d'un ensemble de processus clients est identique à celui d'une exécution dans laquelle toutes les opérations sur les données sont exécutées selon une certaine séquence S d'un processus P.

Exemple :

6.4.4 Modèle de cohérence causal

C'est un modèle plus faible que le modèle de cohérence séquentielle, car on ne considère que des événements reliés par une relation de causalité.
Un modèle de cohérence est causal s'il garantit les deux conditions :

1. Les opérations d'écritures potentiellement causalement liées doivent être perçues par tous les processus dans le même ordre.
2. Les opérations d'écritures non causalement liées peuvent être perçues dans des ordres différents, sur des processus différents.

Relâchement de la cohérence séquentielle en fonction de la causalité entre les opérations.
Tous les processus perçoivent les opérations d'écriture causalement liées dans le même ordre.
Par contre, aucune contrainte sur les opérations d'écriture concurrentes.
il y a moins de synchronisation que dans le modèle de cohérence séquentielle et linéarisable.

6.4.5 Modèle de cohérence PRAM

Un modèle de cohérence est PRAM (ou bien FIFO) s'il garantit que les opérations d'écriture effectuées par un même processus sont perçues par tous les processus dans l'ordre où ces opérations ont été effectuées. Les opérations d'écriture effectuées par différents processus peuvent être perçues par chaque processus dans un ordre différent. Il permet un relâchement du modèle de cohérence causale : certaines transitivités dues à la relation de causalité ne sont pas considérées. Dans un processus unique, la causalité se réduit à l'ordre FIFO.

6.4.6 Modèle de cohérence à la longue

Un modèle de cohérence est à la longue s'il garantit que les opérations d'écriture seront propagées ultérieurement. C'est le modèle de cohérence le plus faible. Ce modèle n'impose aucun ordre sur les opérations.

6.5 Modèles de cohérence relachêe (avec synchronisation)

Faire rentrer les écritures en section critique (personne ne peut accéder aux variables) et d'envoyer les modifications à tout le monde lorsque les mises à jour sont effectuées. Dans les modèles de cohérence sans synchronisation les opérations pour synchroniser les processus se font par des mécanismes indépendants. En intégrant ces synchronisations avec les modèles de cohérence, il est possible de relâcher des contraintes et donc d'obtenir un protocole de cohérence moins coûteux et plus performant. Les modèles de cohérence avec synchronisation sont aussi appelés modèles de cohérence faibles. On distingue trois modèles : cohérence faible, cohérence relâchée, cohérence à l'entrée.

6.5.1 Cohérence faible

Un modèle de cohérence est faible s'il garantit les trois propriétés suivantes :

1. Les opérations sur les variables de synchronisation se font selon un modèle de cohérence séquentielle ;
2. L'accès aux variables de synchronisation ne peut se terminer que si toutes les opérations d'écriture et de lecture sont terminées sur tous les sites ;
3. Les opérations de lecture et d'écriture ne peuvent se faire que si toutes les opérations sur les variables de synchronisation sont terminées sur tous les sites.

Ce modèle garantie pour un processus : que toutes les modifications effectuées sont reçues par tous les processus et qu'il a reçu toutes les modifications faites par tous les processus, et il permet le regroupement des propagations.

6.5.2 Cohérence au relâchement

Un modèle de cohérence est au relâchement s'il garantit les trois propriétés suivantes :

1. Les opérations de synchronisation (acquire et release) sont utilisées selon le modèle de cohérence PRAM. Acquire (acquisition) : indique à la mémoire qu'une entrée en section critique est en cours. Release (libération) : indique à la mémoire une sortie de la section critique.
2. Les opérations de lecture et d'écriture ne peuvent se faire que si toutes les opérations d'acquisition précédentes sont terminées sur tous les sites

3. L'opération de relâchement ne peut se terminer que si toutes les opérations d'écriture et de lecture sont terminées sur tous les sites.

Quand un processus effectue une opération de relâchement, les modifications qu'il a effectuées seront observées par tous les processus faisant une acquisition ultérieure.

6.5.3 Cohérence à l'entrée

Ce modèle permet d'associer à chaque variable partagée un objet de synchronisation comme le verrou. Cette stratégie permet de ne transférer que des mises à jour en rapport avec la donnée. L'avantage est qu'une écriture interdit uniquement l'accès à la donnée et non pas l'accès à une zone mémoire où d'autres données peuvent être stockées. L'établissement de la relation entre donnée et verrou est cependant laissé à la charge du programmeur. La cohérence à l'entrée fait la distinction entre les accès en lecture et les accès en écriture grâce à l'utilisation de deux verrous.

6.6 Compromis de la cohérence

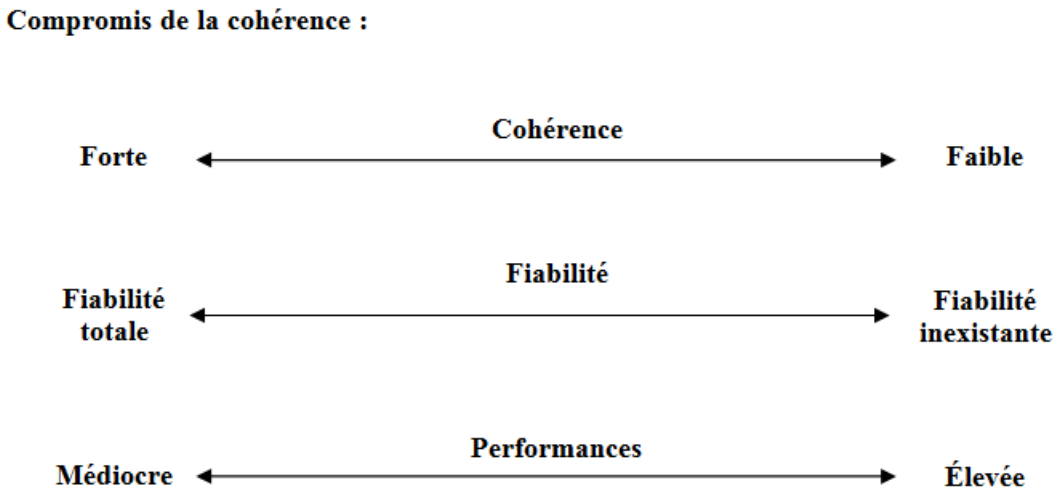

FIGURE 6.3 – Compromis de la cohérence

Pour obtenir une bonne fiabilité, il est nécessaire d'avoir une cohérence forte, ce qui pénalise sensiblement les performances du système.

6.7 RECONFIGURATION DYNAMIQUE DE COTERIE STRUCTUREE EN ARBRE

Les protocoles à quorum permettent de diminuer le nombre de messages à échanger afin de maintenir la cohérence entre les répliques. Dans les grilles, les nœuds peuvent tomber en panne et leur charge varie au cours du temps. → Les protocoles de maintien de la cohérence doivent s'adapter à cette dynamicité.

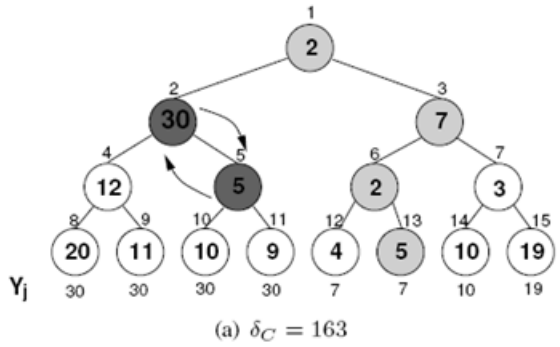

FIGURE 6.4 – Exemple de Coterie Structurée en arbre

6.8 Définitions

On considère : Pr= {P | P est un processeur}
A chaque processeur P, est associé une charge de travail notée Xp.
Chaque processeur possède une copie d'une donnée d. la copie de d sur le processeur P sera notée dp.
Coterie
Un ensemble C de parties de Pr est une coterie et est définie comme suit :

$$C = Q\,\varepsilon\,P(P_r) \mid \forall Q' : Q'\,\varepsilon\,P(P_r)\,et\,Q' \neq Q \rightarrow Q \cap Q' \neq \emptyset\,et\,Q \subsetneq Q'$$

La propriété Q⋂Q'=∅ est appelée propriété d'intersection et la propriété Q⊈Q' est appelée propriété de minimalité.
Chaque élément Q d'une coterie C est appelé Quorum
Les processeurs sont organisés logiquement en arbre. Les processeurs sont des nœuds ou des feuilles de l'arbre. Une opération de lecture (ou d'écriture) est effectuée sur un quorum de la coterie. Un quorum est obtenu en prenant tous les processeurs de n'importe quel chemin allant de la racine à une feuille de l'arbre.

6.8.1 Charge d'un quorum

La charge YQ d'un quorum Q est le maximum des charges xp des processeurs P qui composent le quorum.

$$Y_Q = Max(x_p : P\varepsilon Q)$$

La charge d'un quorum représente le maximum des charges car ce qui importe est le temps de réponse pour une opération de lecture ou d'écriture. Une telle opération doit attendre les réponses de tous les processeurs d'un quorum. C'est le temps du plus lent qui est déterminant.

Exemple :

Q1 (1, 2, 4, 5) la charge de ce quorum est YQ1=30
Q2 (1, 2, 4, 9) la charge de ce quorum est YQ2=30

6.8.2 Charge d'une coterie

On note δC la charge de la coterie C.
La charge de cette coterie est équivalente à la somme des charges des quorums de C.

$$\delta_c = \sum_{Q \varepsilon C} Y_Q$$

La charge d'une coterie représente la somme des charges des quorums, car les opérations de lecture ou d'écriture initiées par un processeur sont réparties uniformément sur l'ensemble des quorums d'une coterie.
Dans l'exemple la charge de la coterie est $\delta_c = 163$

6.9 Permutation élémentaire d'une coterie structurée en arbre

On applique une permutation élémentaire sur une coterie afin d'obtenir une nouvelle coterie moins chargée, qui consiste à :

1. Trouver deux nœuds dans l'arbre qui sont parents, un père et son fils tel que la charge du fils soit inférieure à la charge du père.
2. Une permutation élémentaire consiste à permuter le nœud père avec le nœud fils s'ils satisfont la contrainte de charge.

Cela a pour effet de positionner le processeur le moins chargé au dessus du processeur le plus chargé → coterie moins chargée
Dans l'exemple la permutation entre nœuds a permis de réduire la charge de la coterie à 135.

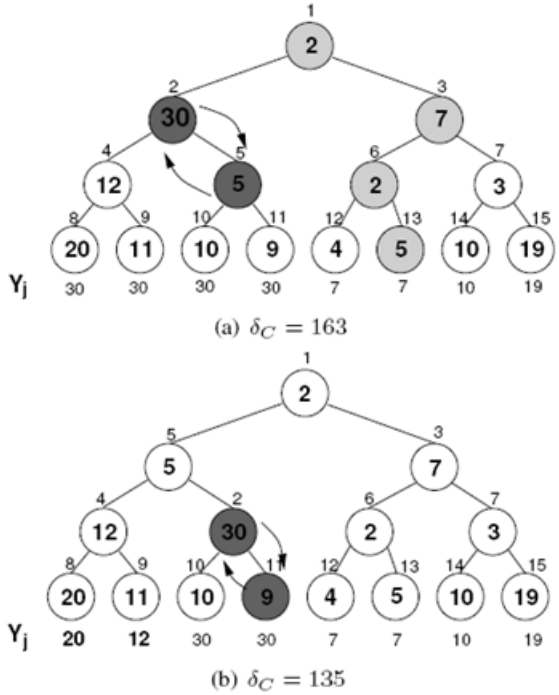

FIGURE 6.5 – Permutation élémentaire d'une coterie structurée en arbre

6.10 Lecture / écriture sans reconfiguration

Les opérations de lecture ou d'écriture d'une donnée sont décomposées en deux phases : une phase de requête et une phase de propagation.

6.10.1 Phase de requête :

- Un quorum en lecture est contacté
- Chacun des nœuds renvoie la valeur et la version de la donnée.
- La version la plus récente de la donnée est utilisée.

6.10.2 Phase de propagation :

- Un quorum en écriture est contacté afin de propager la nouvelle valeur et la nouvelle version de la donnée.

Les deux phases sont effectuées systématiquement pour une lecture ou une écriture de la donnée : cela permet de mettre à jour les copies obsolètes même lors d'opérations de lecture.

6.11 Lecture / écriture avec reconfiguration

La reconfiguration de la coterie peut être appliquée en même temps que les opérations de lecture ou d'écriture.
Le protocole de reconfiguration est découpé en trois phases :

- **Phase d'installation :** Le reconfigureur contacte un ensemble de processeurs(union de deux quorums, l'un en lecture, l'autre en écriture) en envoyant la nouvelle configuration. Les processeurs renvoient au reconfigureur la valeur et la version de la donnée qu'ils possèdent.
- **Phase de propagation :** (identique à celle du protocole de lecture/ écriture)
- **Phase d'acquittement :** Permet de confirmer la mise en place de la nouvelle configuration.

Le reconfigureur est soit un nœud élu, soit un nœud dédié.

6.12 conclusion

dans le prochain chapitre un des middlware le plus connu « GLOBUS » sera détaillé.

Chapitre 7

GLOBUS

7.1 introduction

Nous décrivons dans ce chapitre le Globus Toolkit 2.2, en définissant les composantes suivantes :

— Single sign-on, authorization, and authentication
— Job submission
— Resource monitoring, searching, and allocation
— Data movement

De plus, Globus Toolkit permet la programmation des API (Application Programming Interface) et SDK (System Development Kits)

Globus Toolkit possède trois pyramides construites en dessus d'une couche de sécurité (Security Infrastructure GSI)

— Gestion des ressources (Resource management)
— Gestion des données (Data management)
— Services d'information (Information services)

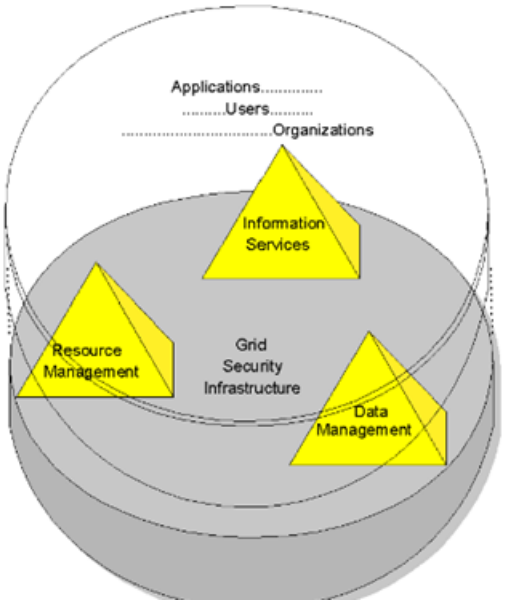

FIGURE 7.1 – Security Infrastructure GSI

Ceci fournit des fonctions de sécurité comprenant l'authentification unique / mutuelle, la confidentialité,

l'autorisation et la délégation de droits.

1. Gestion des ressources (Resource management)
   Cette pyramide offre les fonctionnalités suivantes :
   — Allocation des ressources
   — Soumission de taches (jobs)
   — Gestion de l'évolution des taches (jobs) ainsi que leur statut
2. Gestion des données (Data management)
   Permet le transfert et l'échange de fichiers entre machines.
3. Services d'information (Information services)
   Permet la collecte d'information pour pouvoir traiter les requêtes en se basant sur Lightweight Directory Access Protocol (LDAP).

7.2 Composantes du Globus Toolkit

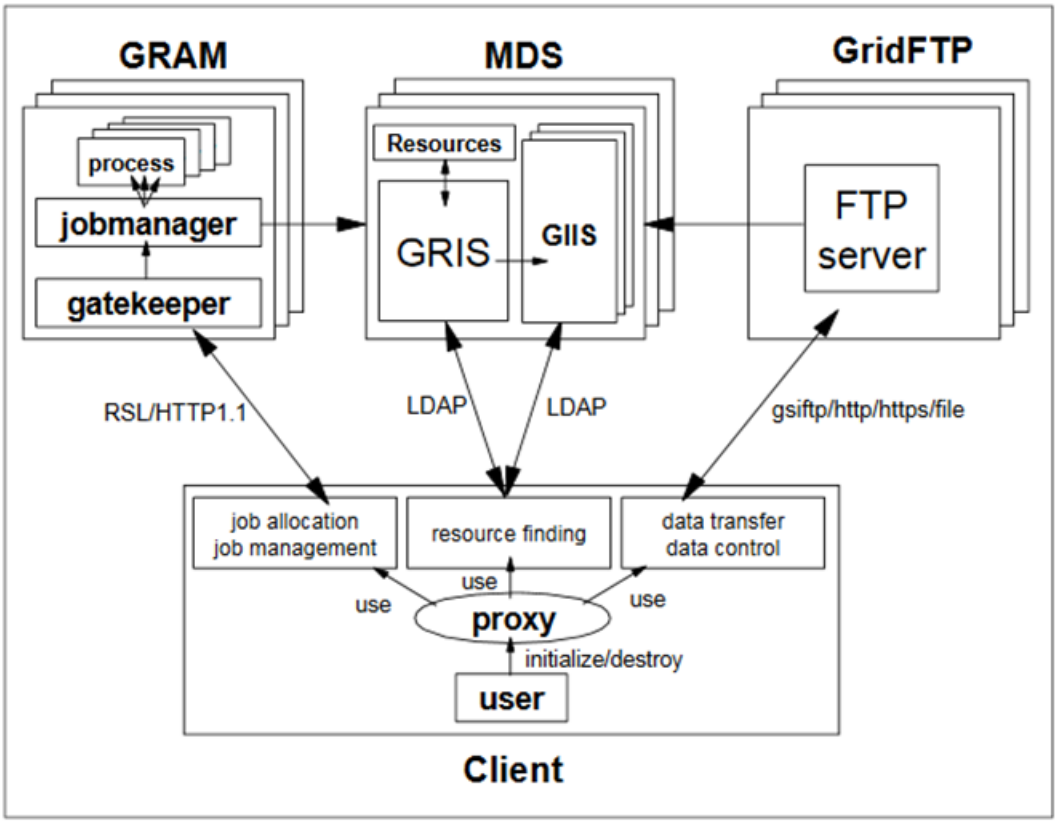

FIGURE 7.2 – Globus Toolkit

7.2.1 GRAM/GASS

Grid Resource Allocation Manager (GRAM) et Global Access to Secondary Storage (GASS) sont les deux principaux composants de la pyramide de gestion des ressources.

7.2.2 MDS (GRIS/GIIS)

Basé sur les trois principaux composants : le Lightweight Directory Access Protocol (LDAP), le Grid Resource Information Service (GRIS) et le Grid Index Information Service (GIIS) qui sont configurés de manière hiérarchique pour la collecte et la distribution des données. Les données collectées peuvent être statiques ou dynamiques (utilisation de CPU, taille de stockage des disques...). Un rapport d'informations retourné par GRIS est renvoyé aux serveurs GIIS de la grille.

7.2.3 GridFTP

GridFTP est un composant clé pour un transfert sécurisé et performant des données. Globus Replica Catalog and Management est utilisé pour la gestion partielle ou complète des données répliquées.

7.2.4 GSI

Tous les composants ci-dessus sont construits sur l'infrastructure GSI (Grid Security Infrastructure). Ceci fournit des fonctions de sécurité comprenant l'authentification unique / mutuelle, la confidentialité, l'autorisation et la délégation de droits.

7.2.5 Grid Resource Allocation Manager (GRAM)

GRAM est un module qui gère l'exécution des taches à distance avec leur statut. Lorsqu'un client soumet une tache, une demande est envoyée à une machine distante pour être traitée au niveau du gatekeeper daemon. Ce dernier va créer un job manager pour lancer et contrôler l'exécution de la tache. Une fois la tache terminée, le job manager envoie le statut au client.
GRAM contient les éléments suivants :

- globusrun command
- Resource Specification Language (RSL)
- The gatekeeper daemon
- The job manager
- The forked process
- Global Access to Secondary Storage (GASS)
- Dynamically-Updated Request Online Coallocator (DUROC)

7.2.5.1 The globusrun command

La commande globusrun permet la soumission et la gestion des taches (jobs) distantes de la manière suivante :

- soumission des taches à une machine distante
- transfert du résultat d'exécution des taches

7.2.5.2 Resource Specification Language (RSL)

RSL est un langage utilisé par un client qui souhaite soumettre une tache. Toutes les requêtes de soumission des taches sont décrites dans RSL, en définissant le fichier exécutable et les paramètres à considérer. Par exemple, spécifier la capacité mémoire nécessaire pour l'exécution d'une tache dans une machine distante.

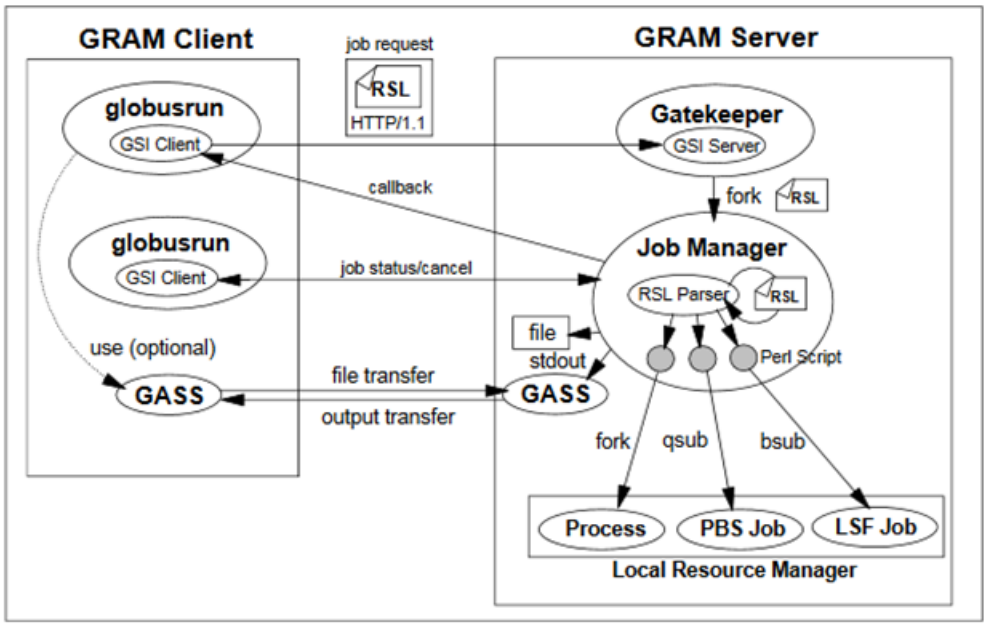

FIGURE 7.3 – Grid Resource Allocation Manager (GRAM)

7.2.5.3 Gatekeeper

Le gatekeeper daemon assure la sécurité des communications entre le client et le serveur. Il communique avec le GRAM client (globusrun) afin d'authentifier les droits de soumission des taches. Après authentification, le gatekeeper crée et autorise le job manager à communiquer avec les clients.

7.2.5.4 Job manager

le job manager offre une interface qui permet le contrôle des allocations de chaque ressource locale, tel que le job scheduler. Le rôle du job manager est :

- Analyser le langage de la ressource Decompose le script RSL.
- Allocation des requêtes des taches aux gestionnaires de ressources locaux
- Envoyer des rappels aux clients, si nécessaire.
- Recevoir le statut et annuler les demandes des clients.
- Envoyez les résultats aux clients en utilisant GASS, si nécessaire.

7.2.5.5 Global Access to Secondary Storage (GASS)

GRAM utilise GASS afin d'assurer le transfert des fichiers résultats (output file) des serveurs aux clients. Certaines API sont fournis sous le protocole GSI afin d'assurer des transferts sécurisés. Ce mécanisme est utilisé par le globusrun command, le gatekeeper et le job manager.

7.2.5.6 Dynamically-Updated Request Online Coallocator (DUROC)

En utilisant le mécanisme DUROC, les utilisateurs peuvent soumettre des taches à différents job manager et à différentes machines.

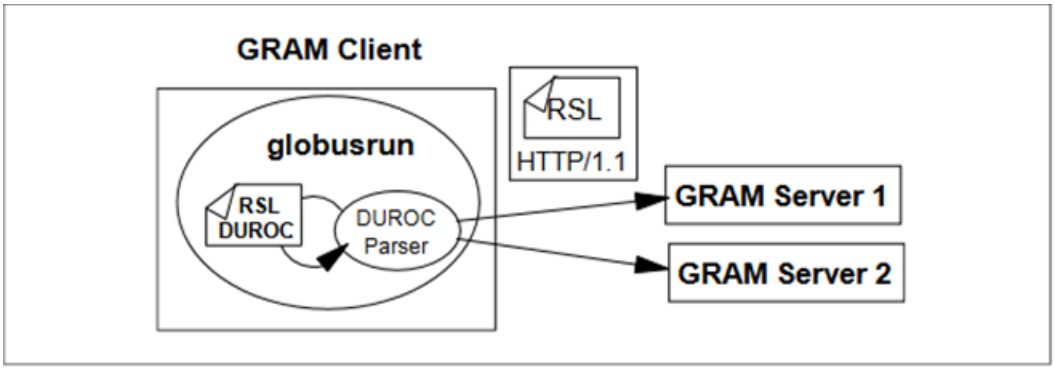

FIGURE 7.4 – DUROC

7.2.5.7 Monitoring and Discovery Service (MDS)

MDS offre un accès aux informations (statiques/ dynamiques) des ressources. Il est composé principalement des éléments suivants :

- Grid Resource Information Service (GRIS)
- Grid Index Information Service (GIIS)
- Information Provider
- MDS client

Dans la figure suivante, une vue conceptuelle des interconnexions entre les composants du MDS est représentée. Comme illustré, les informations d'une ressource sont obtenues par le « information provider » et renvoyées au GRIS. GRIS enregistre ses informations locales dans le GIIS, qui seront enregistrées dans un autre GIIS et ainsi de suite. Un client MDS peut obtenir les informations des ressources directement du GRIS (pour les ressources locales) et du GIIS (pour les ressources distantes). MDS utilise LDAP, qui permet une maintenance décentralisée des informations des ressources.

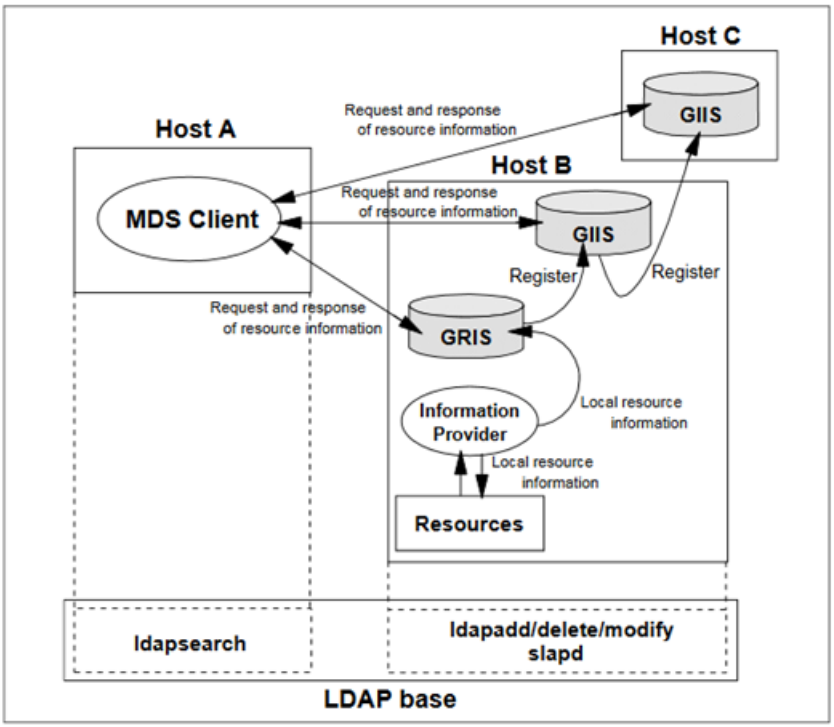

FIGURE 7.5 – MDS (Monitoring and Discovery Service)

Resource information

« Resource information » contient les objets qui sont gérés par le MDS, spécifiant les composantes de ces ressources (statiques ou dynamiques) comme suit :

— Infrastructure : par exemple, nom du job manager ou le nom du job en cours d'exécution
— ressources : Par exemple, l'interface réseau, l'adresse IP, la taille de la mémoire

Grid Resource Information Service (GRIS)

GRIS est un référentiel qui détient les informations des ressources locales prises du « information provider ». GRIS doit enregistrer ses informations avec GIIS. Les informations locales détenues par le GRIS sont mises à jour et mises en cache pour un certain temps nommé TTL (Time To Live). Si aucune requête d'information n'est reçue par le GRIS, l'information sera alors supprimée. Toutefois, si jamais une requête d'information est reçue, GRIS appellera le fournisseur d'informations « information provider » le plus pertinent pour récupérer l'information la plus récente.

Grid Index Information Service (GIIS)

GIIS est un repository qui détient les indexes des informations des ressources enregistrées par le GRIS et les autres GRIISs. Cela peut être vu comme un grand serveur de grilles d'information. GIIS est base sur un mécanisme hiérarchique, comme le DNS, et chaque GIIS possède son propre nom. Càd, un utilisateur client peut spécifier le nom d'un GIIS lors de la recherche d'information.

Information providers

Les fournisseurs d'informations « information providers » traduisent les propriétés et le statut des ressources locales au format défini dans les fichiers de schéma et de configuration. Afin d'ajouter votre propre ressource à utiliser par MDS, vous devez créer des fournisseurs d'informations spécifiques pour traduire les propriétés et le statut en GRIS.

MDS client

Le client MDS est basé sur la commande client LDAP, ldapsearch. Une recherche des informations de ressources que vous souhaitez dans votre environnement de grille est initialement effectuée par le client MDS.

Hierarchical MDS

Le mécanisme de hiérarchie MDS est similaire à celui utilisé dans DNS. GRIS et GIIS, au niveau des couches inférieures de la hiérarchie, s'inscrivent avec le système GIIS aux niveaux supérieurs. Les clients peuvent interroger le système GIIS pour obtenir des informations sur les ressources permettant de créer un environnement de grille.

GridFTP

GridFTP fournit un transfert de données sécurisé et fiable entre les nœuds de la grille. Le mot GridFTP peut désigner un protocole, un serveur ou un ensemble d'outils.

GridFTP protocol

GridFTP est un protocole destiné à être utilisé dans tous les transferts de données sur le réseau. Il est basé sur FTP, mais étend le protocole standard avec des fonctionnalités telles que le transfert en streaming, le réglage automatique et la sécurité basée sur Globus. Ce protocole est toujours en version préliminaire. Pour plus d'informations, reportez-vous au site Web suivant :
http ://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf le protocole GridFTP n'étant pas encore complètement défini, Globus Toolkit ne prend pas en charge l'ensemble des fonctionnalités de protocole actuellement présentées. Un ensemble d'outils GridFTP est distribué par Globus sous forme de packages supplémentaires.

GridFTP server and client

Globus Toolkit fournit le serveur GridFTP et le client GridFTP, qui sont implémentés respectivement par le démon in.ftpd et par la commande globus-url-copy. Ils supportent la plupart des fonctionnalités définies sur le protocole GridFTP. Le serveur GridFTP et le client prennent en charge deux types de transfert de fichier : standard et tiers. Le transfert de fichier standard est l'endroit où un client envoie le fichier local à la machine distante, qui exécute le serveur FTP. Le transfert de fichier tiers est un fichier volumineux stocké dans un stockage distant et que le client souhaite copier sur un autre serveur distant.

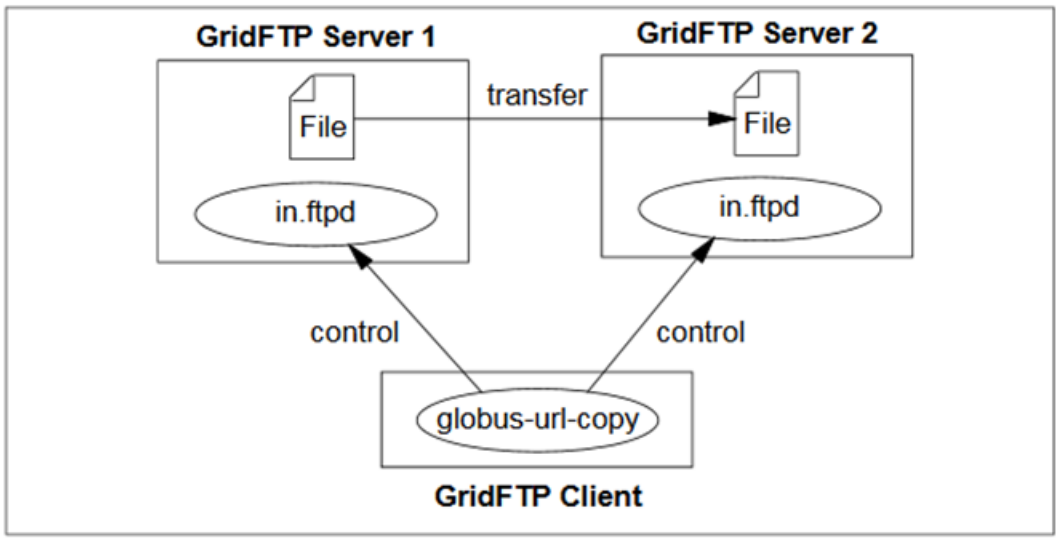

FIGURE 7.7 – transfert de fichier à tiers

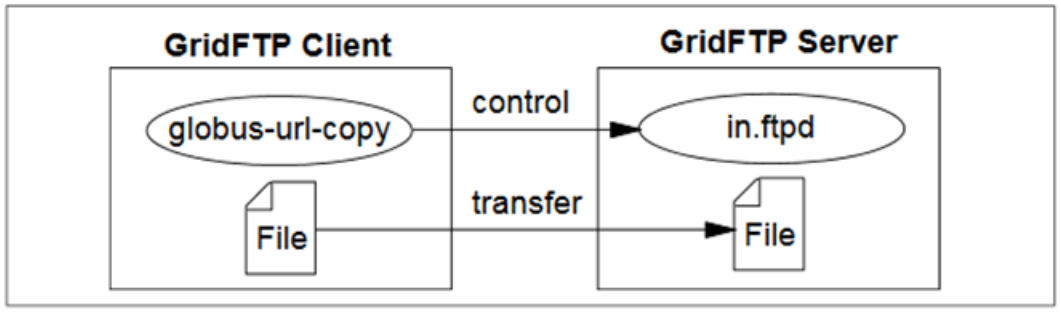

FIGURE 7.6 – transfert de fichier standard

7.3 conclusion

dans ce chapitre le middlware « GLOBUS » a été définis avec toutes ses fonctionnalités.

1. Sébastien Monnet. Gestion des données dans les grilles de calcul : support pour la tolérance aux fautes et la cohérence des données. Réseaux et télécommunications. Université Rennes 1,2006. Français. tel-00411447
2. Luis Ferreira,Viktors Berstis et al. « Introduction to Grid Computing with Globus, International Technical Support Organization, septembre 2003 http ://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf
3. Ghalem Belalem, Naima Belayachi, Radjaa Behidji etBelabbes Yagoubi ; Load Balancing to Increase the Consistency of Replicas in Data Grids ; International Journal of Distributed Systems and Technologies (IJDST), Vol. 1, No. 4, pp. 42-57, 2010.
4. Ghalem Belalem etYahya Slimani ;A hybrid approach to replica management in data grids ;International Journal of Web and Grid Services (IJWGS), Vol. 3, No.1, pp. 2-18,2007
5. [Djebbara et Belbachir 2010]Réda M. Djebbara etHafida Belbachir : Dynamic threshold for replicas placement strategy ;International Conference on Machine and Web Intelligence (ICMWI), pp. 394 –401, Alger 2010.
6. [Fekete et Ramamritham 2010] Alan D. Fekete etKrithi Ramamritham ;Consistency Models for Replicated Data ;Replication Lecture Notes in Computer Science,Vol.5959, pp.1-17, 2010.
7. [Foster 2002] Ian Foster ;What is the Grid ? A Three Point Checklist ;Grid Today, Vol.1, No.6, juillet 2002.
8. [Frain et al. 2005] Ivan Frain, Jean-Paul Bahsoun etAbdelaziz 'zoughi : dynamic reconfiguration of a coterie tree-structured ;Journées Francophones sur la Cohérence des Données en Univers Réparti (CDUR'2005),pp. 34-40,2005.
9. [Hieu et al. 2010] Quang Hieu Vu, Mihai LupuetBeng Chin Ooi ;Load Balancing and Replication ;Peer-to-Peer Computing, chapitre 5, pp.127-156, 2010
10. [Lamehamedi et al. 2002] HoudaLamehamedi, Boleslaw Szymanski, ZujunShentu etEwaDeelman ;Data replication strategies in Grid environments ;5th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02),pp. 378–383,Chine, 2002
11. [Rahman et al. 2007] Rashedur M. Rahman, Ken Barker etReda Alhajj ;Replica Placement Strategies in Data Grid ;Journal of Grid Computing, Vol.6, No. 1, pp.103-123, Mars 2008
12. [Storm 2012] Christian Storm ;Specification of Quorum Systems ;Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes, pp.81-153, 2012.
13. Extensive information about Globus Toolkit and Globus Project can be found at : http ://www.globus.org
14. Open standards : Globus Toolkit is an open standard software developed and blueprinted by the Globus Project.
15. Information about the collaborators of the project can be found at the following Web site : http ://www.globus.o
16. In addition to the Globus Project, the Global Grid Forum has contributed a collection of references and standards. The GGF, as the Global Grid Forum is known, is a community initiative that includes participants of over 200 organizations in more than 30 countries.
17. The GGF Web site is : http ://www.ggf.org/