الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Dr. Tahar Moulay SAIDA
Faculté : Technologie
التكنولوجيا
Département : Informatique

جامعة د الطاهر مولاي سعيدة
كلية:
قسم : الإعلام الآلي

MEMOIRE DE MASTER

Option : Sécurité Informatique et Cryptographie

THEME

Stratégie de réplication de données dans les grilles

Présenté par :

- **Roiussat Abd lhafidh**
- **Malti Mohamed**

Encadré par :

- ✓ **Mme Kouidri siham**

2015-2016

REMERCIMENTS

Nous avons abouti un travail, qui a été le résultat d'un cheminement de tout un parcours pédagogique, qui a duré tout le long de notre parcours éducatif dans l'enseignement supérieur.

Un remerciement particulier à notre encadreur Mme Kouidri pour sa présence, son aide et surtout pour ses précieux conseils qui nous ont assistés pour l'accomplissement de notre projet.

Nous tenons à exprimer nos sincères remerciements à tout le personnel de l'institut des sciences et de la technologie surtout les enseignants qui nous ont formé durant toutes nos années d'étude.

Nous remercions à l'avance tous les jurés pour leurs aides, conseils, et surtout pour leurs attentions et suivi.

Un remerciement particulier à nos très chers parents, frères, surs, collègues et amies respectives qui nous ont encouragés, soutenu durant tout notre parcours.

Merci à tous

DÉDICACE

A chaque fois qu'on achève une étape importante dans notre vie, on fait une pose pour regarder en arrière et se rappeler toutes ces personnes qui ont partagé avec nous tous les bons moments de notre existence, mais surtout les mauvais.

Ces personnes qui nous ont aidés sans le leur dire, soutenus sans réserve, aimé sans compter, ces personnes à qui notre bonheur devient directement le leur, à qui un malheur en nous, en eux se transforme en pleur.

Je dédie ce modeste travail en signe de reconnaissance et de respect.

- A mon père.
- A ma très chère mère qui m'a fait protéger pendant toute ma vie.
- A mes frères.
- A ma soeur.
- A toute ma famille.
- A mon binôme Malti Mohamed.
- A mes amis.
- A tous mes enseignent sans exception.

Rouissat Abdelhafid

DÉDICACE

A chaque fois qu'on achève une étape importante dans notre vie, on fait une pose pour regarder en arrière et se rappeler toutes ces personnes qui ont partagé avec nous tous les bons moments de notre existence, mais surtout les mauvais.

Ces personnes qui nous ont aidés sans le leur dire, soutenus sans réserve, aimé sans compter, ces personnes à qui notre bonheur devient directement le leur, à qui un malheur en nous, en eux se transforme en pleur.

Je dédie ce modeste travail en signe de reconnaissance et de respect.

- A mon père.
- A ma très chère mère qui m'a fait protéger pendant toute ma vie.
- A mes frères.
- A mes soeurs .
- A toute ma famille surtout Rouida et Djihan et Lojin .
- A ma femme que j'aime
- A mon binôme Rouissat Abdelhafid.
- A mes amis.
- A tous mes enseignent sans exception.

Malti Mohamed

Résumé

La réplication des données est employée dans les grilles pour augmenter la disponibilité de données et la tolérance aux fautes. Cependant une stratégie de réplication efficace doit s'appuyer sur un placement judicieux des répliques pour rapprocher les données des nœuds qui les demandent. De plus, la topologie logique de la grille affecte énormément la technique de réplication implémentée.

Dans ce document, nous proposons une stratégie dynamique de réplication basée sur la méthode FastSpread. L'approche implémentée tient compte de l'importance des répliques. Les résultats de simulation obtenus sous le simulateur OptorSim ont donné des résultats prometteurs en termes de temps de réponse aux requêtes, la consommation desressources réseaux ainsi que le nombre de répliques créées.

Mots clés : Réplication de données, Réplication dynamique, Fastspread, moins récemment utilisée, placement des répliques, Simulation, OptorSim.

Table des matières

Table des figures

Liste des tableaux

INTRODUCTION GÉNÉRALE

Les grilles informatiques sont actuellement des solutions proposées pour répondre aux besoins des systèmes à large échelle. Elles fournissent un ensemble de ressources variées, géographiquement distribuées dont le but est d'offrir une importante capacité de calcul parallèle, d'assurer un accès rapide et efficace aux données, d'améliorer la disponibilité et de tolérer les pannes.

Cependant, la distribution à la large échelle des données d'une grille et le dynamisme de ses sites posent respectivement le problème d'accès distant et de disponibilité des données. Ces paramètres sont extrêmement importants dans le contexte de grilles de données où l'accès des utilisateurs est fréquent. La maniere la plus eficace pour répondre à cela est la technique de réplication qui consiste à créer des copies identiques d'une même donnée sur plusieurs sites. Pour bénéficier au maximum du gain que peuvent apporter les répliques de données, leur placement stratégique dans le système est crucial. Plusieurs stratégies ont été proposées dans la littérature, elles visent principalement à créer et placer de manière efficace les répliques dans la grille afin qu'un nœud puisse trouver les données dont il a besoin en un temps minimal. Les modèles sur lesquels s'appuient ces stratégies différents pour atteindre au final les mêmes objectifs : améliorer le temps d'accès aux données, augmenter la disponibilité, tolérer les pannes et économiser l'utilisation de la bande passante. Dans ce travail, nous nous intéressons au problème de réplication dans les grilles de données. Nous proposons une stratégie de réplication basée sur la méthode FastSpread[19]. Cette solution utilise une topologie en graphe pour représenter la grille en entrée et implémente un algorithme de placement des répliques en prenant en compte divers facteur. Notre travail a été simulé sous l'environnement OptorSim. Qui inclut des outils de base pour le test et le développement de nouvelles solutions.

L'ensemble des concepts étudiés et résultats obtenus, sont synthétisés dans ce mémoire composé de 3 chapitres outre une introduction générale et une conclusion.

- Le premier chapitre est consacré à l'état de l'art sur les grilles informatiques.
- Le deuxième chapitre présente différentes techniques de réplication de données et décrit quelques travaux liés au travail présenté dans ce mémoire.
- Le troisième chapitre décrit la stratégie de réplication implémentée. Nous présentons d'abord le modèle logique de la grille, l'architecture fonctionnelle du modèle de réplication ainsi que ses différents services implémentés de façon détaillée. Nous terminons par présenter et discuter les résultats obtenus.

Finalement, nous concluons en rappelant la problématique abordée ainsi que les principaux résultats obtenus. Par la suite, nous présentons quelques perspectives de recherche qui nous semblent pertinentes.

Le lecteur trouvera à la fin du manuscrit dans l'annexe une description du simulateur utilisé.

Chapitre 1

Les grilles informatiques

1.1 Introduction

En tant que technologie, les grilles informatiques tentent de répondre aux besoins des applications scientifiques caractérisées par un calcul intensif et des volumes de données de l'ordre du Pétaoctet. L'idée principale de cette technologie est la distribution des calculs et/ou des données sur des ressources dispersées géographiquement à une large échelle et sous utilisées.

Les grilles informatiques trouvent leurs applications dans divers domaines de la science telles que la biologie, la physique, etc. Elles ont aussi envahi le monde industriel, le travail collaboratif et le e-business. Une telle émergence nécessite des standards offrant une interopérabilité entre systèmes. Pour cela, un modèle en couche protocolaire a été décrit et des middlewares ont été développés.

1.2 Les grilles informatiques

1.2.1 Origine

Le mot grille en français est un terme d'origine anglophone " Grid " ; il est apparu à la fin des années 1990 et provient de l'analogie avec le fonctionnement des réseaux d'alimentation en électricité, " Electrical Power Grid " en anglais. Les réseaux électriques produisent sans cesse de l'énergie électrique qui est fournie à la demande, à l'usager qui en a besoin. Ce système est composé de deux grandes parties : d'un côté se trouvent les unités de fabrication d'énergie électrique et de l'autre, il y a les consommateurs. Ces deux ensembles sont reliés entre eux par un réseau de transport assurant l'acheminement et la régulation de l'énergie. Ainsi, lorsque l'on branche un appareil électrique dans une prise de courant, on a accès à la ressource électricité, sans avoir besoin de savoir ni où ni comment elle a été produite, ni même quel chemin elle a parcouru pour arriver jusqu'à nous. On branche l'appareil dans la prise qui est au mur, et l'appareil électrique peut se mettre en marche.

L'idée des grilles informatiques est semblable à cela, à la différence que les ressources fournies aux consommateurs ne sont plus de l'électricité mais de la puissance de calcul, de l'espace de stockage informatique, différents instruments et capteurs ou encore l'accès à certaines données. Les unités de production sont alors des ressources informatiques placées dans des sites distants et reliés aux consommateurs par des réseaux de transport de données.

Ainsi, au lieu de se brancher dans une prise électrique, l'usager se connecterait à un réseau informatique sur lequel il pourrait alors consommer des calculs et/ou des données pour faire fonctionner ses programmes. L'idée est donc de lier des ressources de calcul et de stockage hétérogènes et distribuées afin que, pour l'utilisateur, elles apparaissent comme une unique entité. Une sorte de puissante machine virtuelle serait alors ainsi formée par le partage de nombreuses machines provenant d'organisations administratives différentes. Le niveau actuel des technologies et le fort intérêt qu'elles suscitent font que les grilles semblent offrir une solution intéressante aux problèmes du calcul et du stockage intensif [1].

D'une façon générale, on parle de grilles informatiques. Lorsqu'il s'agit de supercalculateurs, ou de grilles qui s'occupent plus de calculs on dit alors " grilles de calculs ", et on parle de " grilles de données " pour celles qui traitent plus des données.

1.2.2 Définition

En 1998, Ian Foster et Carl Kasselman ont défini les grilles de calcul comme suit : "Une grille de calcul est une infrastructure matérielle et logicielle qui fournit un accès fiable, uniforme, répandu, à coût réduit à des capacités de calcul importantes".

Les auteurs ont ensuite affiné cette définition en mettant l'accent sur l'aspect " partage de ressources et organisation virtuelle ". Ils considèrent notamment que le partage n'est pas un simple échange de fichiers, mais plutôt un accès direct aux ordinateurs, applications, données et autres ressources. Le partage est hautement contrôlé par les fournisseurs de ressources et les consommateurs, en définissant clairement ce qui est partageable, qui a accès aux ressources partagées et dans quelles conditions se fait ce partage. Les individus et/ou les institutions définis par la politique de partage constituent ce que l'on appelle une Organisation Virtuelle(VO) [2,8].

Depuis leur apparition, les grilles de calcul sont de plus en plus utilisées. Elles sont passées du cadre académique au cadre industriel. Cependant, des confusions apparaissent et on a tendance à parler de grille à chaque fois qu'on dispose d'un réseau dans lequel on partage des ressources. Pour éviter ce type de confusions, Ian Foster, dans son article "What is the Grid ?" [2,44] manifeste la nécessité d'une définition claire. Il présente trois critères de base pour distinguer une grille d'un autre système distribué. Il définit une grille comme un système qui :

1. Coordonne des ressources sans contrôle centralisé (ressources autonomes).
2. Se base sur des standards.
3. Délivre une Qualité de Service non négligeable (temps de réponse, disponibilité, sécurité, etc.).

Les systèmes qui ne respectent pas les trois points précédents ne sont pas considérés comme grilles. Ian Foster cite les exemples suivants :

- Sun Grid Engine, un système de ressources distribuées de Sun Microsystems, ne constitue pas réellement une grille du fait de son contrôle centralisé.
- Le Web ne constitue pas une grille car il ne coordonne pas les ressources pour offrir une bonne QoS.

Comme autres caractéristiques importantes des grilles de calcul, nous pouvons mentionner l'hétérogénéité et l'aspect dynamique. L'hétérogénéité peut être une hétérogénéité matérielle (processeurs, mémoires, réseaux de communication) ou logicielle (systèmes d'exploitation, systèmes de fichiers, environnements de développement, etc.). Les ressources sont dynamiques du fait qu'elles se connectent et se déconnectent de manière imprévisible. La définition suivante résume les points clés cités précédemment : " Une grille de calcul est une infrastructure constituée de ressources matérielles/ logicielles hétérogènes, distribuées, autonomes, partagées offrant un accès fiable, dynamique, hautement sécurisé à des capacités de calcul et de stockage importantes "[2].

1.2.3 Motivation

Pourquoi les grilles de calcul ? Quels bénéfices apportent-elles ? Comment expliquer l'émergence de cette technologie ? Nous apportons des éléments de réponse dans ce qui suit. Parmi les raisons pouvant amené à déployer une grille de calcul nous citerons le besoin de [4,9] :

- Partage de ressources à l'échelle mondiale : Plusieurs ordinateurs alimentent un réseau global pour fournir une ressource globale, c'est-à-dire une grande puissance de calcul et une grande capacité de stockage. Chaque utilisateur peut alors utiliser via les réseaux à très haut débit ces ressources considérables, au moyen d'une interface simplifiée : son propre ordinateur.
- Exploitation des ressources sous utilisées : Dans la plupart des organisations, les machines sont utilisées pour des tâches de bureautique. Le processeur est occupé au maximum 5
- Organisation virtuelle : L'une des motivations clés de l'émergence des grilles est l'organisation virtuelle. L'utilisateur accède à des ressources hétérogènes, dispersées géographiquement de manière transparente. L'organisation virtuelle cache la complexité de la structure de la grille et le partage des ressources [23, 24].

– Exécution parallèle : Il existe des applications qui peuvent être subdivisées en tâches pouvant s'exécuter indépendamment les unes des autres. Dans ce cas, l'exécution simultanée de chacune des tâches sur une machine à part de la grille conduit à la réduction du temps d'exécution totale de l'application. Cependant, ce ne sont pas toutes les applications qui peuvent être transformées pour être exécutées de manière parallèle sur une grille. Aujourd'hui, les applications aptes à être parallélisées le sont dès l'étape de la conception. Ces dernières sont facilement déployées sur les grilles et en bénéficient pleinement [23, 24].
– Fiabilité et disponibilité des services : Du fait que les ressources fédérées par une grille de calcul soient géographiquement dispersées et disponibles en importantes quantités. Une grille permet d'assurer la continuité du service si certaines ressources deviennent inaccessibles. Les logiciels de contrôle et de gestion de la grille seront en mesure de soumettre la demande de calcul à d'autres ressources.

1.2.4 Les différentes grilles informatiques

Selon les besoins et les objectifs visés par les applications, les grilles supportantces dernières peuvent être classées en différentes catégories :

– **Les grilles de stockage :** Cette catégorie de grille permet le partage de données externalisées entre plusieurs nœuds. Les cas les plus représentatifs de cette catégorie sont les réseaux d'échange Peer-to-Peer. Les grilles appartenant à cette catégorie permettent l'accès à des données (fichiers, flux, etc...) via un réseau de sites (ou serveurs) qui contiennent et partagent un index. Les données sont référencées pour optimiser les recherches à travers un moteur de recherche. Les fichiers peuvent se trouver sur des nœuds différents du réseau et en différents points du globe. Comme exemple de cette catégorie de grilles, nous pouvons citer notamment le réseau GNUTELLA ou encore KaZaA[4].
– **Les grilles de calcul :** Permettent de distribuer des calculs sur des ressources réparties pour bénéficier d'une plus grande puissance de calcul. Ces grilles sont en général formées de clusters et de serveurs, mais parfois des PC de bureau sont utilisés. Ces grilles deviennent de plus en plus nécessaires pour faire face à l'augmentation constante des besoins en puissance de calcul. Ainsi, la complexité des systèmes étudiés en recherche scientifique et dans l'industrie (thermique, fluides, biologie, etc.) induit des besoins de puissance de calcul pouvant atteindre plusieurs téraflops. Les projets nécessitant de telles puissances de calcul sont par exemple les modèles météo et les études sur le changement climatique global, les simulations de matériaux, les simulations et outils de conceptions en aéronautique, automobile, chimie ou nucléaire, certains calculs de risques dans le domaine de la finance[5].
– **Grille de service ou grille d'information :** Elles sont utilisées pour les systèmes qui offrent des services pouvant être disponibles sur une simple machine. On distingue deux catégories :
  – Grille à la demande : elle agrège dynamiquement déférentes ressources pour fournir de nouveaux services. On peut citer comme exemple, le problème de simulation qui nécessite des ressources (en nombre et en type), qui dépendent des paramètres d'exécution.
  – Grille de collaboration : elle rassemble les utilisateurs et les applications en groupe de travail collaboratif. Ce type de grille permet une interaction entre utilisateurs et applications en temps réel, au moyen d'un espace de travail virtuel [3].

1.3 L'évolution des grilles informatiques

1.3.1 première génération

Les premiers projets de grilles de calcul sontconnus sous le nom du " MetaComputing ". Ce terme regroupe l'ensemble des techniques permettant d'utiliser plusieurs supercalculateurs au sein d'une même application pour répondre aux besoins de calcul de haute performance, on peut citer deux autres projets qui ont, chacun à leur façon, influencés l'évolution de certains des projets technologiques clés en cours, afférents à la grille [23] :

– Le projet FAFNER (factorisation par récursivitédynamisée par réseau) avait pour objectif la factorisation de très grands nombres, qui est très couteux en temps de calcul.
– Le projet I-WAY (Information Wide Area Year) conçu en 1995, était un projet d'un an, visant à relier des supercalculateurs en utilisant uniquement des réseaux existants, sans en construire de nouveaux. Les outils développés pour monterce projet ont été intégrés au projet Globus.

1.3.2 Seconde génération : utilisation de middleware

Les années 2000 ont connu une véritable émergence des grilles de calcul grâce à leur maturité. Une grille ne se réduit plus à l'interconnexion des supercalculateurs, mais relie également de simples machines géographiquement dispersées pour supporter diverses applications nécessitant des capacités de calcul et/ou d'espace de stockage qui ne peuvent pas être supportées par des machines classiques. Cette génération fut par ailleurs, particulièrement marquée par l'apparition d'un outil essentiel d'automatisation dans l'usage de grilles, le gestionnaire ou le courtier de ressources (ressources broker). Beaucoup de middlewares ont été proposés et le projet Globus [13] est actuellement la norme de base pour les middlewares des grilles de calcul.

Cette deuxième génération est caractérisée par [25] :

- **Hétérogénéité des ressources :** les ressources dans une grille sont de nature hétérogène en termes de matériels et de logiciels.
- **Passage à l'échelle (scalability) :** une grille pourra consister de quelques dizaines de ressources à des millions voire des dizaines de millions.
- **Nature dynamique des ressources :** La grille permet d'ajouter des ressources au fur et à mesure de leur disponibilité et/ou des besoins ou d'en retirer pour effectuer des opérations de maintenance par exemple. Cependant, cela pose des contraintes sur les applications telles que l'adaptation au changement dynamique du nombre de ressources, la tolérance aux pannes.

1.3.3 Troisième génération : approche orientée service

Le développement de nouvelles applications pour les grilles a suscité le besoin de définir des composants de base qui pourraient ensuite être utilisés comme briques de base dans un processus de développement. C'est ainsi qu'il a été proposé d'organiser les composants selon leurs fonctionnalités pour constituer un service. On parle d'architecture orientée service ou OGSA (OpenGrid Service Architecture) [38].

Les points majeurs de cette génération sont [4] :

1. Meilleure sécurité d'accès aux grilles et aux ressources par l'utilisateur.
2. Une gestion des fichiers adaptée à la prise en charge de données massives sur grille.
3. La généralisation de la notion d'organisation virtuelle [15] permettant de définir des sous-ensembles de ressources et leurs modalités d'usage par des communautés d'utilisateurs.
4. L'introduction de modèles économiques [13] pour l'allocation de ressources.
5. Le passage d'un mode d'accès à des ressources d'une grille vers la fourniture de services de grilles.
6. L'interopérabilité via une normalisation des services web appliqués aux grilles, tel qu'OGSA (Open Grid Services Architecture) et WSRF (Web Services Resource Framework) [5,38].

1.4 Domaine d'application des grilles

Les principales applications des grilles se font dans les domaines suivants [11,37] :

- Supercalculateur réparti (Distributed Supercomputing) : Une grille de calculs pourra réquisitionner une importante quantité de ressources (PCs inexploités) afin de fournir la puissance de calcul nécessaire pour alléger la charge des supercalculateurs.
- Calcul haut-débit (High-Throughput Computing) : Une grille de calculs sera utilisée pour ordonnancer en parallèle une importante quantité de tâches (jobs) indépendantes les unes des autres.
- Calcul à la demande (On-Demand Computing) : Une grille de calculs pourra fournir les ressources nécessaires pour satisfaire les demandes à court terme d'une application.
- CalculCollaboratif(CollaborativeComputing) :cetteclassed'applicationsinclutles applicationsd'interactionentrehumainsdansdesenvironnementsdesimulationen temps-réel par exemple la conception et l'interaction avec un nouveau moteur d'avion [25].

– Génération, traitement et stockage d'énormes quantités de données (Data- intensive Computing) : Dans de telles applications, une grille de calcul pourra absorber et stocker l'importante quantité d'informations générées. Comme exemple d'applications nous pouvons mentionner la production d'une carte de l'univers, la prévision météorologique à long terme, les simulations quantiques [25].

1.5 Principe des grilles

Les technologies de grille reposent sur cinq grands principes[5] :

– Le partage des ressources à l'échelle mondiale : plusieurs ordinateurs, situés partout dans le monde, mettent leur puissance de calcul et de stockage en commun. C'est le principe de base de la grille.
– Un accès sécurisé : les fournisseurs et les utilisateurs de cette puissance de calcul, qui ignorent leurs identités réciproques, doivent pouvoir échanger des données en toute sécurité.
– L'utilisation équilibrée des ressources : le mécanisme d'affectation répartit les travaux de manière efficace et équilibrée entre les ressources disponibles.
– L'abolition de la distance : le développement des connexions à très haute vitesse permet d'échanger rapidement des données avec un ordinateur situé à l'autre bout du monde.
– L'utilisation de normes compatibles : des applications faites pour être exécutées sur une grille doivent pouvoir l'être sur toutes les autres. Cela suppose que les normes qui régissent les grilles soient compatibles entre elles.

1.6 Architecture des grilles de données

L'architecture d'une grille est souvent décrite en termes de " couches " ayant chacune une fonction spécifique. Généralement, les couches supérieures sont orientées vers l'utilisateur, tandis que les couches inférieures sont orientées vers le matériel (axées sur les ordinateurs et les réseaux)[36].

L'architecture générale d'une grille de données est illustrée par la figure 1 [11,40] :

– Le niveau " Fabric " contient l'ensemble des ressources matérielles (calculateurs, serveurs de stockage, instruments de mesures etc.) connectées à travers un réseau rapide. Ces ressources exécutent des logiciels tels que des systèmes d'exploitation, systèmes de gestion de bases de données, systèmes de soumission de tâches, etc.
– Le niveau " Communication " contient les protocoles de communication qui sont construits au-dessus de protocoles tels qu'IP. Ces protocoles implantent la sécurité via des mots de passes ou des couches de types SSL (Secure Socket Layer). Les protocoles de transferts fournissent des mécanismes de transferts de données entre deux ressources de la grille de données.
– Le niveau " Service de grille de données " inclut des services de gestion, de transfert et de calcul de données sur la grille à travers des mécanismes transparents.
– Le niveau " Application " fournit des outils permettant l'accès aux applications distantes. Ces interfaces peuvent être des API, des portails par exemples définis dans le cadre des applications cibles.

De très nombreux outils existent implémentant tout ou partie des différentes couches d'une grille de données. Il est relativement difficile de faire une classification précise de chacun des projets en particulier parce que le nombre de recherches dans le domaine est très grand. Cependant, une donnée doit pouvoir être stockée, accédée, déplacée, répliquée de manière transparente.

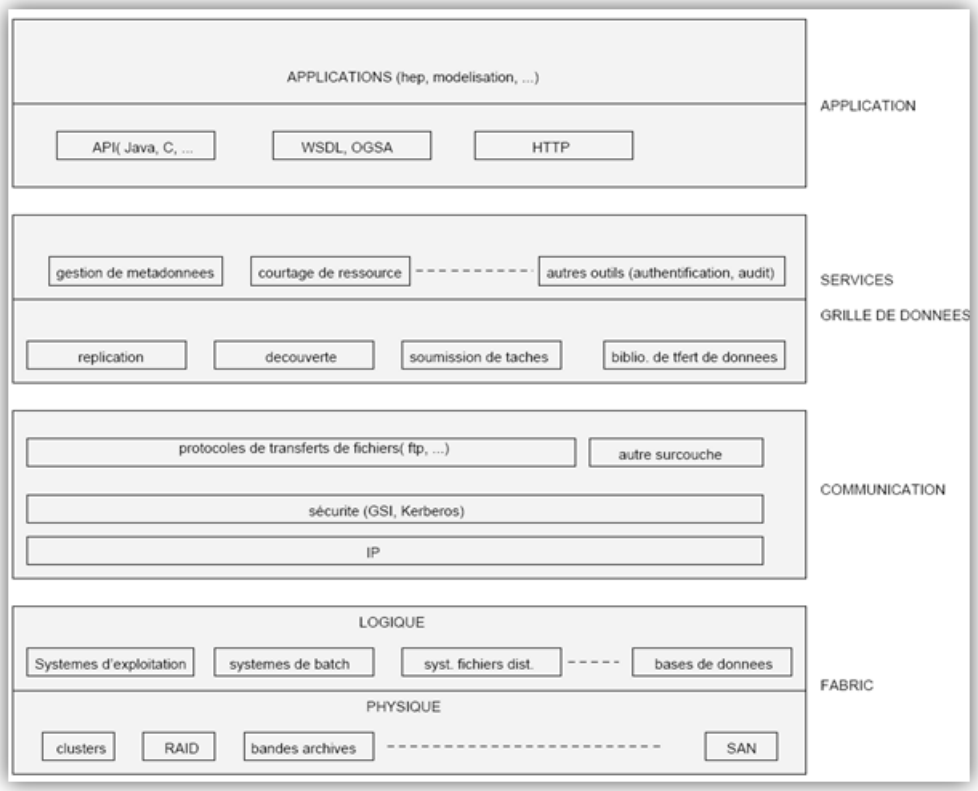

FIG. 1.1: Architecture d'une grille de données

1.7 Les intergiciels des grilles

Le Middleware est la brique de base regroupant l'ensemble des éléments logiciels pour la mise en œuvre d'une grille, il est informé en permanence de leur état : unités de traitement et de stockage constitutives des nœuds, branches de réseau, librairies de programmes toute entité qui peut être définie et administrée par le gestionnaire de ressources. Il assure aux utilisateurs et aux applications les services de :

- **Réservation** et **allocation** des ressources nécessaires
- **Ordonnancement** et **lancement** des travaux
- Suivi de l'activité
- Administration du système.

Dans l'architecture d'une grille, la couche la plus basse est la couche Fabrique composée des ressources proprement dites, la couche la plus haute est composée des applications utilisateurs. Le middleware se trouve entre ces deux couches. Dans cette architecture, le middleware est divisé en deux niveaux (Figure2) : le corps du middleware et le middleware niveau utilisateur. Le corps du middleware offre les services élémentaires tels que l'exécution des tâches, l'allocation des ressources, la sécurité, le service d'information et le transfert de fichier. Le niveau utilisateur comprend les environnements de développement, les outils de programmation, les applications de planification de tâche, etc [2].

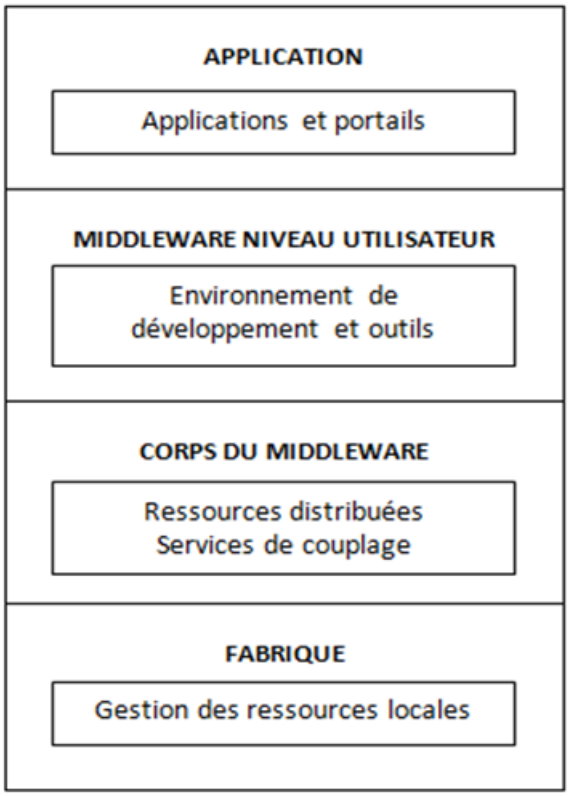

FIG. 1.2: Le middleware dans l'architecture de la grille

1.7.1 Globus

Parmi les intergiciels les plus utilisés dans le domaine des grilles, nous citons Globus, qui est une boîte à outils destinée à la construction d'une grille[39] qui a été développé aux Etats-Unis dans l'Argonne National Laboratory par l'équipe de Ian Foster. Le projet Globus a débuté en 1997 et a délivré sa première version en novembre 1998.

Les quatre constituants principaux de Globus sont classés en quatre grandes classes [16] :

Service	Nom	Description
Gestion de ressources	GRAM	Allocation des ressources et gestion des processus.
Communications	Nexus	Services de communication unicast et multicast.
Sécurité	GSI	Authentification et autorisation.
Information	MDS	Information sur la structure et l'état de la grille.

TAB. 1.1: Les constituants principaux de Globus

Le tableau ci-dessus est décrit en détails dans la figure suivante :

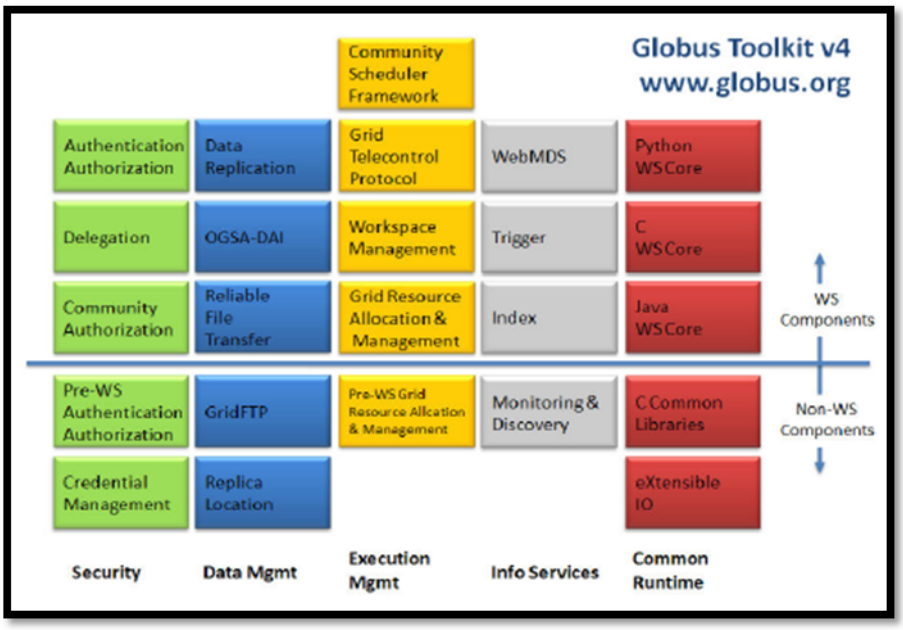

FIG. 1.3: Les briques de base définies par le Globus Toolkit.[41]

1.8 Les services de grilles

Nous présentons dans cette section les services les plus importants d'une grille. Il existe toutefois d'autres services que nous évoquerons seulement [5].

1.8.1 Service d'information grille

La grille est un environnement dynamique où la disponibilité et le type des ressources varient constamment. Il est nécessaire de connaître avec précision la liste ainsi que l'état des machines connectées à la grille. Le service d'information grille (GIS) est l'entité qui gère ces informations en temps réel. Il collecte des informations telles que :

- la liste des ressources de calcul et de stockage disponibles.
- la liste des utilisateurs de la grille.
- les services disponibles.
- l'état de chaque ressource (taux d'occupation processeur, mémoire et disque).
- l'état du réseau d'interconnexion au moyen d'outils comme NWS (Network Weather Service).

Cette liste n'est probablement pas exhaustive. Néanmoins, la précision des informations fournies par le GIS est altérée par la forte latence réseau présente entre les sites : le rafraîchissement n'est pas immédiat. Ces informations sont toutefois primordiales pour les autres services de la grille. Par exemple, l'ordonnanceur de tâche interroge régulièrement le GIS pour connaître la liste des ressources de calcul disponibles, le taux d'occupation des processeurs et de la mémoire, avant d'effectuer un choix définitif pour l'exécution de la tâche. Ce service doit donc être très disponible, et l'accès à l'information rapide. Pour cela, le GIS est distribué sur plusieurs nœuds de la grille. Au moins un GIS est présent dans chaque domaine administratif, et gère les ressources de son site, car la latence réseau au sein d'un site est relativement faible par rapport à celle qui relie les domaines. De plus, des agents disposés sur chaque nud sont chargés de récolter les diverses informations concernant l'état de la machine. Ces agents envoient périodiquement (ou bien sur une demande particulière), un rapport au GIS le plus proche. Les GIS ainsi répartis sur les différents sites, doivent alors régulièrement synchroniser leurs informations pour que celles-ci soient accessibles partout sur la grille. Cette notion est aussi appelée le monitoring des ressources.

1.8.2 Service d'ordonnancement de tâches

Les applications pour grille sont découpées en tâches indépendantes, de façon à paralléliser les calculs pour exploiter au mieux toutes les ressources disponibles. Le service d'ordonnancement des tâches est chargé de sélectionner les ressources (nœuds de calcul) de la grille, susceptibles de les exécuter. L'objectif principal de cet ordonnanceur est de trouver la meilleure configuration pour que le temps d'exécution d'une tâche soit minimal.

Les principaux critères de sélection des ressources sont : le type et la vitesse des processeurs, la mémoire disponible, la taille de la mémoire virtuelle, l'emplacement géographique ainsi que la charge courante de la machine.

Les outils de monitoring du GIS sont alors très sollicités pour connaître l'état des nœuds de la grille au moment de l'ordonnancement. L'ordonnanceur tient également compte des droits d'accès dont dispose l'utilisateur qui s'adresse à lui.

Pour pouvoir travailler, l'ordonnanceur a besoin de connaître certains détails relatifs aux tâches qu'il doit affecter à ces ressources. Pour cela, il existe des méthodes qui permettent de décrire les tâches ainsi que les ressources de la grille. Les détails pouvant ainsi caractérisé une tâche sont nombreux :

– Le système d'exploitation ou bien l'architecture de la machine sur laquelle la tâche doit s'exécuter.
– La quantité de mémoire et l'espace disque nécessaire.
– La puissance de calcul requise.
– La priorité de la tâche.

Avec toutes ces informations, l'ordonnanceur peut choisir une ou plusieurs ressources. Certains systèmes pratiquent la réservation de ressource. Parmi les ordonnanceurs les plus utilisés actuellement, on retrouve : Portable Batch System (PBS) qui est une solution pour les grappes fonctionnant sous Linux et les systèmes de calcul haute performance.

Bien souvent, l'ordonnancement tel que nous venons de le présenter est insuffisant, car il n'est pas assez dynamique. De plus dans certaines configurations grille, il est très difficile de faire de la réservation de ressources. Par exemple les machines de bureau sont des ressources non dédiées à la grille, qui peuvent exécuter d'autres tâches (e.g bureautique).L'ordonnancement se doit d'être encore plus dynamique par ce qui s'appelle l'adaptabilité d'une tâche en cours d'exécution. Lorsque le monitoring se rend compte qu'il ya un changement important dans l'état d'une ressource, il en informe un module, chargé de faire migrer une ou plusieurs tâches sur des ressources de calculs différentes. Le module doit alors sauvegarder le contexte d'exécution de chacune des tâches, pour pouvoir ensuite reprendre l'exécution là où elle s'était arrêtée. Les futurs ordonnanceurs intégreront cette dynamique pour permettre aux applications d'exploiter encore mieux l'environnement grille.

1.8.3 Service de gestion des données

Le service de gestion des données se divise en deux catégories : l'accès aux données et la gestion des métadonnées. Cette distinction permet de différencier le stockage des données utiles de celui des métadonnées.

Le service d'accès aux données fournit des mécanismes pour : gérer la localisation et le transfert des données. Il assure également la sécurité au niveau du dispositif de stockage, ainsi que lors des transferts. Il est capable de détecter et parfois de corriger les erreurs grâce à la mise en place de codes correcteurs.

Le service de métadonnées assure la gestion des informations qui caractérisent les données utiles de la grille. Très souvent, la gestion des métadonnées existe à plusieurs niveaux dans les intergiciels. Les métadonnées les plus connues sont celles concernant les fichiers (droits utilisateurs, nom du propriétaire, date de dernière modification, taille, nombre de blocs utilisés, type de fichier, etc.). Ceci dit, il en existe d'autres comme par exemple les métadonnées liées à la réplication, qui assurent la correspondance entre toutes les instances d'un fichier répliqué et leurs emplacements physiques respectifs sur la grille (i.e. la ressource de stockage utilisée).

1.8.4 Service de réplication des données

La réplication est aujourd'hui largement utilisée dans les grilles. Elle consiste à créer plusieurs copies d'un même fichier sur des ressources de stockage différentes de la grille. Cette technique permet d'augmenter la disponibilité des données, la charge étant alors répartie sur les différents nœuds possédant une réplique. Le service de réplication met en œuvre une politique de cohérence particulière. Il doit également maintenir un catalogue des répliques, décidé quand créer et supprimer une copie. Il s'appuie en général sur le système de monitoring des ressources du GIS, pour prendre ces décisions. Par exemple, lorsqu'une donnée est partagée par plusieurs tâches ou applications, la ressource de stockage ainsi que le réseau, peuvent très vite saturer. Le gestionnaire des répliques peut alors décider de créer une nouvelle copie sur un autre nud, afin de répartir la charge et permettre aux applications de s'exécuter plus vite.

1.8.5 Service de gestion de la concurrence

Dans un environnement distribué tel que la grille, la concurrence d'accès aux données est très forte. Pour une application utilisant MPI (Message Passing Interface), ce service n'est pas nécessaire car c'est l'application elle-même qui gère la concurrence d'accès à ses données. Par contre, si l'application respecte le standard POSI (Portable Operating System Interface), c'est le système de fichiers qui se doit de garantir la cohérence des données. Dans un environnement grille, celle-ci est gérée par une entité : le service de gestion de la concurrence. Il distribue des verrous sur des objets (métadonnées, données, entrée de cache, etc..).

1.9 La gestion des données dans les grilles

Les applications de grille sont amenées à accumuler des masses de données de plus en plus importantes,pouvant atteindre la centaine de TéraOctets [34]. Ces données doivent faire l'objet de traitementscomplexes, souvent couteux en terme de puissance de calcul.Actuellement, l'approche la plus utilisée pour la gestion des données, nécessaires aux calculs répartissur différentes machines d'une grille de calcul, repose sur des transferts explicites des données. A titred'exemple, la plateforme Globus [35] fournit des mécanismes d'accès distants aux données (GASS :Globus Access to Secondary Storage) basés sur le protocole GridFTP [35]. Ce protocole, qui est uneextension du protocole standard FTP pour les grilles, comprend un certain nombre de fonctionnalités, tels que la gestion de l'authentification, la gestion de l'intégrité des données, les transferts parallèles, la gestion des reprises en cas d'échec, etc. Bien que plus évolué que le protocole FTP de base, GridFTP exige toujours une gestion explicite de la localisation des fichiers.

Le concept de grille de données [32] est apparu suite à la demande des scientifiques de disposer d'une très grande capacité de stockage de l'ordre du Pétaoctets. Ce type de grilles permet la gestion et le contrôle de données distribuées. Elles peuvent être considérées comme un rassemblement de plusieurs GFS (Grid File Systems), et fournissent des fonctionnalités avancées en matière de gestion des données intra et inter-organisations. Un des exemples de ces grilles est la grille de données européenne Datagrid [7], qui a rassemblé 21 partenaires scientifiques et industriels européens, regroupant jusqu'à 15000 ordinateurs et plus de 15 Teraoctets de stockage répartis sur 25 sites en Europe [6]. De nombreux travaux ont été menés pour offrir un accès aisé aux données collectées, à des utilisateurs de cette grille de données. Un ensemble de mécanismes et de modules ont été mis en œuvre dans le cadre de ce projet. Ces modules sont représentés par la Figure 4 et peuvent se définir comme suit :

- **Module de gestion de la réplication des données :** Ce module est chargé des copies de fichiers et de métadonnées dans les zones de cache permettant un accès plus rapide aux données ;
- **Module d'optimisation des requêtes et gestion des accès :** Il est chargé de définir des plans d'exécution pour optimiser les temps d'accès en tenant compte de plusieurs critères, parmi lesquels le type de données à traiter, leurs emplacements ainsi que les performances réseaux ;
- **Module de déplacement de données :** Ce module est chargé d'exécuter les tâches de placement de données incluant la redistribution, la suppression et la création de nouvelles copies de données.
- **Module d'accès aux données :** Ce module gère les accès aux données tout en respectant les spécificités des méthodes d'accès de chaque système de fichiers.

Dans une grille, le rôle du système de gestion et de stockage de données devient crucial pour assurer des accès aisés aux données. En effet, développer et évaluer des modules de gestion de données, pouvant être intégrés à des intergiciels de grille, devient un vrai défi technique.

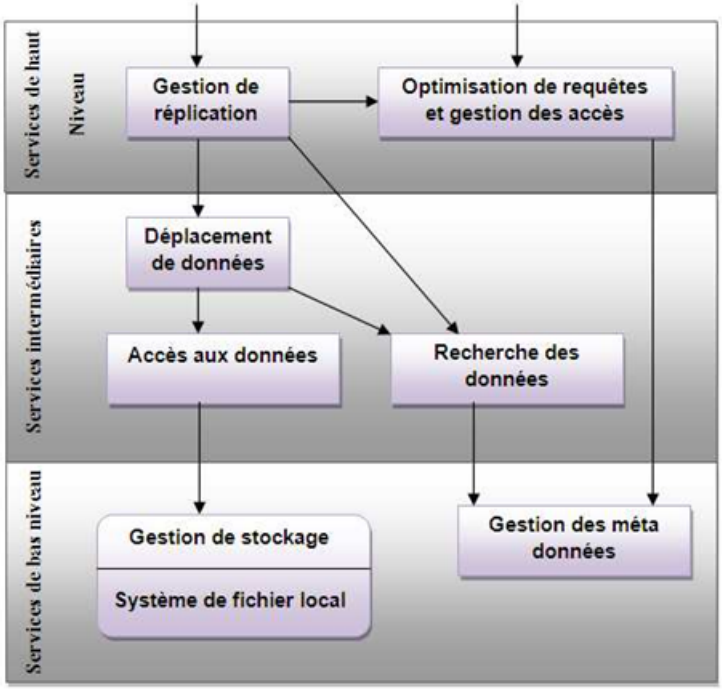

FIG. 1.4: Modules de gestion de données dans la grille européenne Datagrid.

1.9.1 Problèmes liés à la gestion de données dans les systèmes à large échelle

L'utilisation des architectures de type grille de calcul soulève des questions liées à l'administration, l'allocation, la volatilité de ressources distribuées et hétérogènes, la sécurité des données, et bien d'autres sujets qui donnent actuellement lieu à de nombreux travaux de recherche à travers le monde [33, 31, 29,26].Les problématiques liées à la gestion de données posées par les grilles peuvent se résumer ainsi :

- **Accès aux données :** Une des problématiques importantes de la gestion des données dans une grille est l'accès à de très larges volumes de données, qui sont souvent réparties ou dupliquées à travers la grille. L'accès aux données dans une grille de calcul est un problème difficile à appréhender du fait de leur répartition géographique et de leur hétérogénéité. Ainsi, malgré les efforts qui ont été faits dans le domaine de l'interconnexion locale de grappes avec des technologies comme Myrinet, Infiniband ou 10G Ethernet [30, 27], les grilles doivent traverser des WAN, avec les problèmes de bande passante et surtout de latence qui leurs sont associés.
- **Ordonnancement conjoint des calculs et des données :** Une autre problématique posée par les grilles est l'ordonnancement conjoint des calculs et des données. Cette problématique est soulevée par de nouvelles contraintes imposées par la grille : l'importante masse de données pouvant être traitée et l'hétérogénéité des latences et des débits réseaux. Il devient indispensable de proposer des techniques effectuant un ordonnancement multicritères combinant à la fois les calculs et la gestion des données. L'efficacité de telles techniques passe par une étude du lien entre le placement de données et l'ordonnancement des tâches.
- **Réplication et cohérence :** L'hétérogénéité des sources de données et le facteur de la grande échelle ont rendus le problème de la réplication très complexe dans les grilles. Les stratégies de réplication doivent être adaptées aux configurations de grande taille. Ceci nécessite des mécanismes de synchronisation globale qui peuvent être très pénalisants. Aussi, il faut définir des modèles et des protocoles de cohérence adaptés aux hypothèses de la grande échelle.
- **Placement de données :** Le placement des données sur différents sites, de façon à minimiser le coût d'accès aux données, reste un des problèmes majeurs de la gestion de données dans les grilles. Le placement de données sur une grille est rendu difficile par l'hétérogénéité des plates-formes, le caractère dynamique des ressources et surtout par le passage à l'échelle [28].

1.10 Conclusion

Ce chapitre a été l'occasion de fournir une vue générale sur les grilles, leurs architecture, leurs objectifs, ainsi que leurs domaines d'application. Nous avons tenté de répondre aux questions suivantes : Pourquoi recourir aux grilles ? Comment les grilles sont utilisées et quels sont les composants qui sont impliqués dans l'architecture d'une grille ?

Dans le chapitre suivant nous décrirons les concepts de base de la réplication qui est la technique clé pour la gestion des ressources dans les environnements à grandes échelle.

Chapitre 2

La réplication dans les grilles de données

2.1 Introduction

Les techniques de partage utilisées dans les grilles de données, sont souvent basées sur le principe de réplication qui permet de faire une copie d'un objet ou d'un fragment d'objet sur plusieurs nœuds.

La réplication des données est une solution efficace pour obtenir de bonnes performances, le traitement en parallèle des demandes d'accès améliore les performances en termes de temps de réponse et de charge acceptable par le système. De plus, une donnée ne devient indisponible que si tous les nœuds qui en possèdent une copie tombent en panne simultanément, d'où une meilleure disponibilité des données. En revanche, plus on réplique, plus on crée des points de divergence entre les copies d'une même donnée. Malheureusement, pour garantir la fiabilité d'un ensemble de répliques, il est nécessaire d'avoir une cohérence forte, ce qui peut dégrader les performances. A l'inverse, pour obtenir de bonnes performances, il nécessaire de relâcher la cohérence [9], ce qui qui pénalise la fiabilité.

2.2 Technique de la réplication

2.2.1 Définition

La réplication de données est un mécanisme fondamental pour la plupart des systèmes repartis afin de garantir un certain nombre de propriétés fortement souhaitables, c'est une méthode de base pour assurer le fonctionnement et la continuité des services[10], elle consiste à créer des copies identiques d'ensembles de données (fichiers, bases de données, etc.) sur des sites géographiquement distribués [13]. L'intérêt premier de cette réplication est que, si une donnée n'est plus disponible, le système peut continuer à assurer ses fonctionnalités en utilisant une donnée répliquée, ce qui permet d'augmenter la disponibilité des données et la tolérance aux pannes.

D'autre part, l'utilisation de cette technique va générer un coût supplémentaire à cause de l'augmentation du travail à fournir, la difficulté principale de la réplication est comment maintenir la cohérence des copies si l'une d'entre elle est modifiée[4].

2.2.2 Création de répliques

Les problèmes de coûts sont au centre des stratégies de réplication. La réduction de la latence d'accès ainsi que la consommation de bande passante est un enjeu majeur de la réplication. Le processus de création de copies dépend de la structure et de l'état de l'entité à répliquer. La structure de l'entité peut être indivisible ou composée, alors que l'état peut être constitué de données, de code et éventuellement d'un état d'exécution.

Ranganathan et Foster définissent les trois questions auxquelles une stratégie de création de répliques doit répondre : Quand créer les répliques ? Quels fichiers doivent être répliqués ? Où les répliques doivent-elles être placées ? [5,40,43].

1. **Moment de la réplication :** deux solutions sont possibles ;
   – La création statique : la réplique est créée suite à la demande d'un client et persiste jusqu'à ce qu'elle soit effacée par ce dernier ou que sa durée de vie expire. L'inconvénient de cette méthode est sa non-adaptabilité aux changements de comportementdes participants.
   – La création dynamique : contrairement à la réplication statique la réplication dynamique est indépendante des requêtes des clients. Son but est de permettre la gestion automatique de répliques avec des stratégies adaptées aux comportements des clients.
2. **Le choix de l'entité à répliquer :** les données répliquées sont généralement de deux types : des fichiers ou des objets. Les objets peuvent être composés d'un ensemble de fichiers distribués, et le choix de cette entité dépend de la stratégie de réplication et les besoins des requêtes.
3. **Le placement des répliques :** il affecte considérablement les performances de la grille en termes de disponibilité des données et de temps de réponse aux requêtes. Plusieurs stratégies de placementont été proposées dans la littérature,ces stratégies tiennent compte du fait que les sites potentiels :
   – ne possèdent pas déjà de réplique de la donnée.
   – possèdent l'espace de stockage suffisant.
   – sont à une distance raisonnable en termes de temps de transfert.

2.3 Les stratégies de réplication

Selon quand, qui et comment effectuer les mises à jour des répliques, on distingue entre réplications synchrone/asynchrone, symétrique/asymétrique et totale/partielle [18,36], Les stratégies de réplication peuvent être classées selon différents critères.

2.3.1 placement des répliques

C'est un critère de classification qui focalise sur le contenu de copies de la base. Si toutes les répliques possèdent une copie de chaque objet de la base, alors c'est une réplication totale. Sinon la réplication est partielle [14].

1. **Aucune réplication :** Les données ne sont pas dupliquées entre les sources du système. L'accès aux données distribuées est fait par un serveur centralisant l'information sur lequel les différents nœuds se connectent [15]. Donc si ce serveur tombe en panne, alors toutes ces données sont perdues.
2. **Réplication totale :** La réplication totale consiste à stocker une copie de chaque objet partagé dans tous les sites. Cette approche fournie un équilibrage de charge simple puisque tous les sites ont la même capacité, et une disponibilité maximale tel que chaque site peut remplacer un autre en cas de Panne. La figure ci-dessous représente un exemple de réplication totale de deux objets R et S sur trois sites.

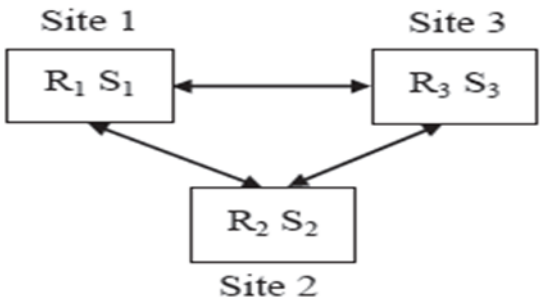

FIG. 2.1: Réplication totale

3. **Réplication partielle :** Dans la réplication partielle chaque site stocke un ensemble de copies d'objets partagés. Les Objets répliqués dans un site peuvent être différents dans un autre site tel qu'il est illustré dans La figure ci-dessous. Cette approche exige moins d'espace de stockage et réduit le nombre de Messages nécessaires pour mettre à jour les répliques puisque les mises à jour n'affectent que Les sites portant la copie de l'objet désiré. Ainsi, elles produisent moins de charge sur le Réseau et sur les sites.

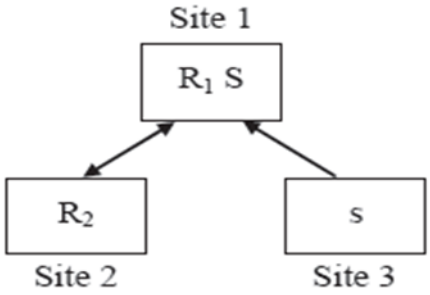

FIG. 2.2: Réplication partielle

2.3.2 Traitement des requêtes

La façon dont les répliques sont traitées par les serveurs qui les stockent est différente selon le mode de réplication. Nous pouvons distinguer trois principaux protocoles qui sont mis en œuvre [3,9,17] :

1. **Protocole de réplication passive :** Une seule copie (la copie primaire) reçoit et traite les requêtes du client et fournit les réponses en sortie, s'il n'y a pas de fautes, les autres copies (les copies secondaires) ne traitent pas les requêtes, et leur état est mis à jour régulièrement par la copie primaire. Si la copie primaire tombe en panne, une des copies secondaires la remplace et s'exécute à partir du dernier point de reprise transmis par la copie primaire (voir la Figure 3). Donc Le principe de la réplication passive est défini comme suit :
   – Réception des requêtes : la copie primaire est la seule à recevoir les requêtes.
   – Traitement des requêtes : la copie primaire est la seule à traiter les requêtes.
   – Emission des réponses : la copie primaire est la seule à émettre les réponses.

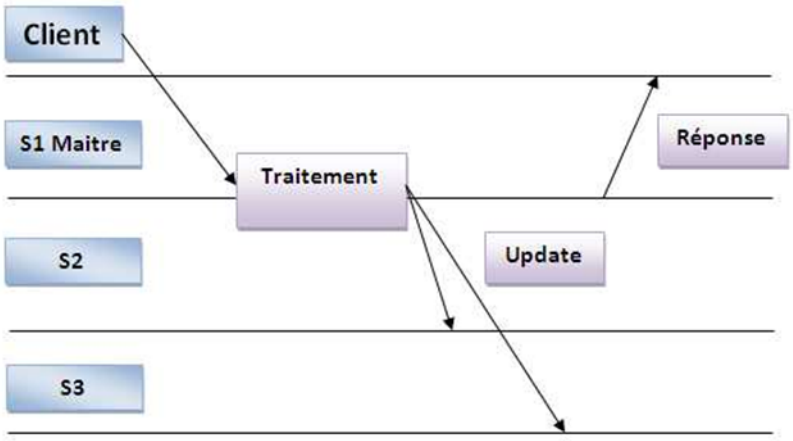

FIG. 2.3: Protocole de réplication passive

La technique de la copie primaire est une solution simple mais le nœud qui possède la copie primaire peut devenir un goulot d'étranglement, de plus le client doit être au courant à chaque sélection d'une nouvelle copie primaire pour pouvoir envoyer ses requêtes.

2. **Protocole de réplication active :** Dans un protocole de réplication active, toutes les copies reçoivent la même séquence totalement ordonnée des requêtes des clients, les exécutent puis renvoient la même séquence totalement ordonnée des réponses vers les clients (voir la Figure 4).

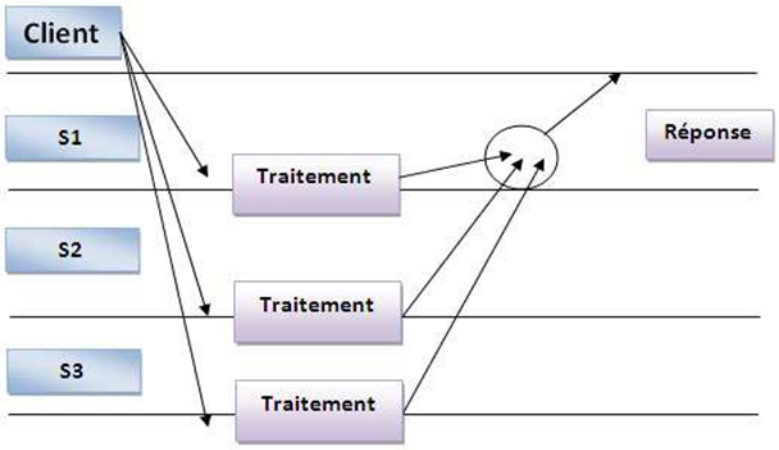

FIG. 2.4: Protocole de réplication active

3. **Protocole de réplication semi-active :** C'est un protocole hybride entre les deux protocoles précédents, où toutes les copies exécutent en même temps la requête du client, mais une seule copie (leader) d'entre elles émet la réponse, les autres copies (suivers) mettent à jour leur état interne et sont donc étroitement synchronisées avec le leader (voir la Figure 5).

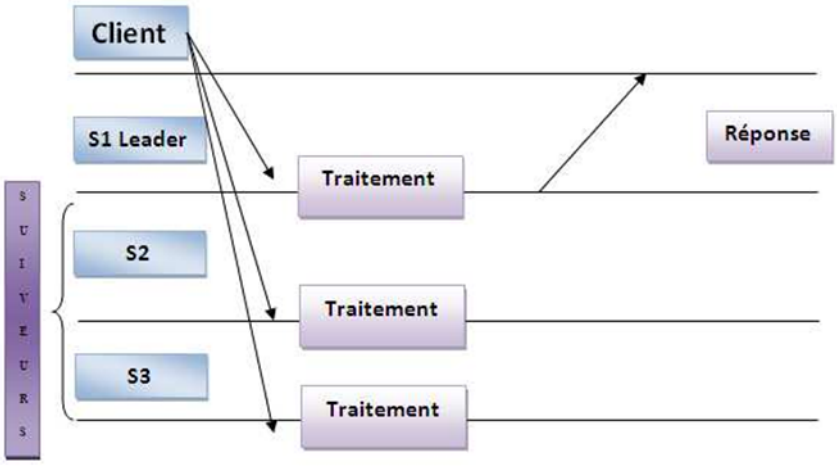

FIG. 2.5: Protocole de réplication semi-active

4. **Comparaison des protocoles de réplication**

Le tableau suivant présente une comparaison entre les protocoles de réplication passive et active [9] :

Caractéristiques	Protocole Passif	Protocole Actif
Ressources utilisées	Moins	Beaucoup
Transparence de Panne/Client	Non	Oui
Traitement de Panne	si (maitre = panne)alors Election	Non
Envoi de requêtes	La requête est envoyée au maitre	Diffusion
Nombre de réponses	Une seule	Plusieurs

TAB. 2.1: Comparaison des protocoles passif et actif

2.3.3 Propagation des mises à jour

1. **La réplication asynchrone :** Cette stratégie est utilisée quand les mises à jour se font sur les bases de données cliente et que la réplication de données se fait sur la base de données primaire.

   Elle permet aux sites de travailler en totale autonomie. La réplication des données est déclenchée par une règle de gestion périodique (exemple : tous les soirs). La mise à jour n'étant pas immédiate, les données transmises ne peuvent servir que pour la consultation

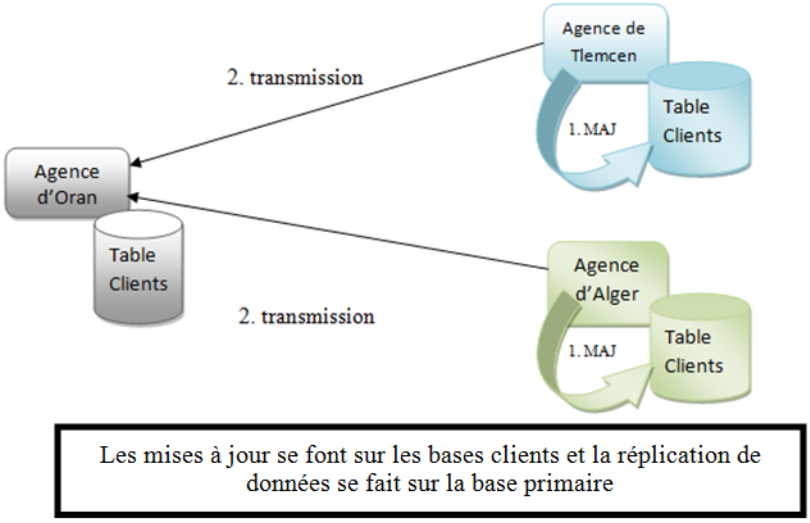

FIG. 2.6: Principe de la réplication asynchrone

2. **La réplication synchrone :**

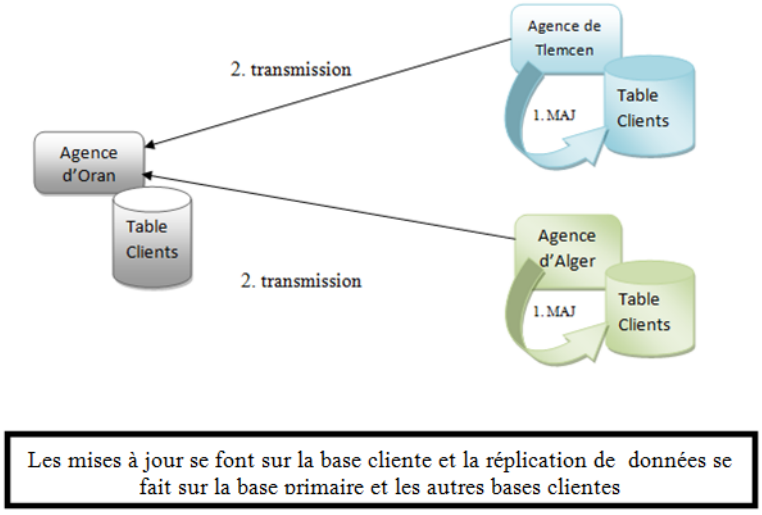

FIG. 2.7: Principe de la réplication synchrone

Cette stratégie est utilisée quand les mises à jour se font sur les bases de données cliente et que la réplication de données se fait immédiatement sur la base de données primaire et sur les autres bases de données clientes :

– Les sites doivent être connectés en permanence.

– La mise à jour n'étant pas immédiate, les données transmises peuvent être mise à jour par plusieurs sites différents.

3. **La réplication périodique :** Comme son nom l'indique, dans ce type de réplication les données sont propagées sur le réseau à des moments bien précis. Les transactions se font en temps réel sur la machine primaire mais leurs réplication sur le site secondaire se fait en mode Batch. La réplication concernera toutes les transactions enregistrées depuis la dernière propagation en date. L'ordre des écritures sur la machine secondaire n'est pas respecté et les données ne sont pas mises à jour continuellement. Le seul avantage de cette méthode est le contrôle du moment de la réplication et de l'impact qu'elle pourrait avoir sur le réseau[45,46].
4. **La réplication Symétrique/Asymétrique :** La réplication symétrique ne privilégie aucune copie. Elle permet les mises à jour simultanées de toutes les copies par des transactions différentes. Cette technique permet de gérer la propagation des mises à jour des copies à tout instant.

   Cette technique pose le problème de concurrence d'accès risquant de faire diverger les copies. Une technique globale de résolution de conflits doit donc être mise en œuvre.

   Par contre, la réplication asymétrique distingue un site maître chargé de centraliser les mises à jour. Appelé aussi site primaire, il est le seul autorisé à mettre à jour les données et est chargé de diffuser les mises à jour aux autres copies dites secondaires (ou esclave).

   Dans ce mode de réplication, le site primaire effectue les contrôles et garantit l'ordonnancement correct des mises à jour.

   Un problème de la gestion de copie asymétrique est la panne du site primaire. Dans ce cas, il faut choisir un remplaçant si l'on veut continuer les mises à jour. On aboutit alors à une technique asymétrique mobile dans laquelle le site primaire change dynamiquement. Il devient nécessaire de gérer à la fois les problèmes de pannes qui provoquent des échecs de transactions et l'évolution des travaux.

2.4 La réplication dans les grilles

La réplication dans les grilles doit s'adapter aux caractéristiques spécifiques de l'environnementhétérogène, dynamique et à large échelle. Un service dédié existe dansl'architecture d'une grille pour la gestion des répliques (Réplica Management Servicede Globus)[16].

2.4.1 Architecture du service de réplication

Dans l'architecture d'une grille, la couche Collective offre unservice de réplication. La figure 8 illustre l'architecture modulaire de ce service [12].

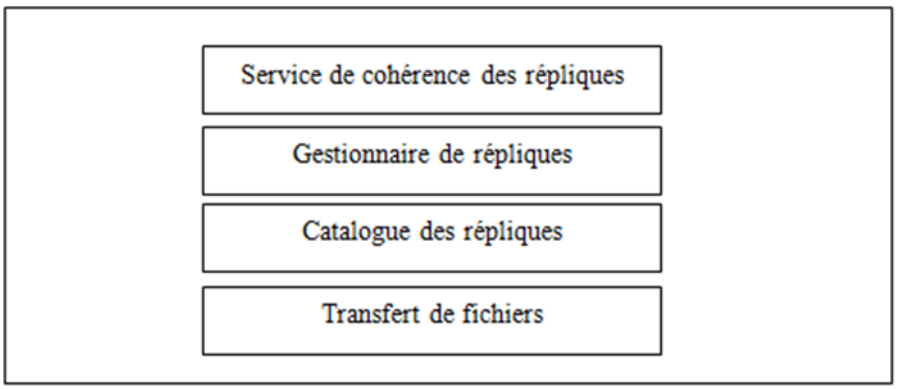

FIG. 2.8: Architecture du service de réplication

Cette architecture est composée de 4 couches qui sont défini comme suit :

- **Transfert de fichiers :** Le transfert de fichier entre les nœuds de la grille se fait au moyen du protocole GridFTP qui est une version de FTP adaptée à un environnement de type grille.
- **Catalogue des répliques :** La fonction du catalogue des répliques est d'assurer la correspondance entre les noms de fichiers logiques et leur localisation physique dans la grille.
- **Gestionnaire de répliques :** Cette couche est responsable de la création, de la suppression des répliques et de la mise à jour du catalogue. C'est la couche qui gère en particulier les problèmes de placement des répliques.
- **Service de cohérence des répliques :** Le rôle de ce service consiste à propager les mises à jour sur toutes les répliques et à synchroniser les écritures. Selon le type de réplication, différents niveaux de cohérence sont assurés.

2.5 Avantages et inconvénients de la réplication

La réplication est l'un des mécanismes les plus importants qui sont utilisés dans les grilles informatiques. Malgré ses avantages, la réplication présente quelques inconvénients que nous allons aborder dans ce qui suit.

2.5.1 Avantages

Parmi les avantages de la réplication on peut citer :

1. Une amélioration de la fiabilité ou bien une sûreté de fonctionnement :
   - Si une copie tombe en panne, il est toujours possible d'obtenir les données à partir d'une autre copie.
   - La redondance permet une meilleure protection contre la corruption de fichiers.
2. Amélioration des performances :
   - Diviser la charge de travail entre plusieurs serveurs.
   - Extensibilité géographique en rapprochant les serveurs des clients.
3. Une disponibilité de données : Les données sont disponibles localement et non plus par le biais de connexions des réseaux, elles sont accessibles localement, même en l'absence de toute connexion à un serveur central, de sorte que l'utilisateur n'est pas coupé de ses données en cas de défaillance d'une connexion réseau longue distance.
4. Temps de réponse : La réplication améliore les temps de réponse des requêtes d'interrogation pour deux raisons :
   - Les requêtes sont traitées sur un serveur local sans accès à un réseau étendu, ce qui accélère le débit.
   - Par ailleurs, le traitement local allège la charge du serveur de bases de données central, ce qui permet de moins solliciter le processeur.

2.5.2 Inconvénients

La réplication peut présenter quelques problèmes tels que :

1. Gestion de la cohérence des répliques : toutes les répliques doivent avoir le même état pour que les services fournis par la grille soient toujours fiables, ce qui implique un travail de gestion énorme (nous avons présenté dans la section III quelques techniques).
2. Coût des mises à jour : la propagation des mises à jour nécessite généralement un certain temps ce qui entraine une période de repos (pas de services). Donc une requête qui arrive pendant cette période sera retardée.

2.6 Travaux connexes

Une technique de placement des répliques représente un facteur important vis-à-vis des performances d'un système de gestion de répliques. Elle doit répondre aux questions : Quand faut-il créer ou supprimer une réplique ?et où la placer ?. La stratégie doit permettre à chaque nud de la grille de trouver les données dont il a besoin dans son

voisinage proche. Plusieurs stratégies de placement de répliques existent dans la littérature, les plus importantes sont présentées dans ce qui suit :

Foster et Ranganathan ont été les premiers à proposer six stratégies de réplication distinctes pour les grilles de données multi-niveaux qui sont [43,48,50] :

1. **No Replica ou caching :** dans cette stratégie seul le nud racine contient les répliques.
2. **Best Client :** une réplique est créée pour le nud qui a le plus d'accès au fichier.
3. **Cascading :** les fichiers originaux sont stockés à la racine, quand le nombre d'accès est supérieur au seuil de la racine, une réplique est créée au niveau suivant qui est sur le chemin du nœud demandeur.
4. **Plain Caching :** le client qui demande le fichier garde une copie locale.
5. **Caching plus Cascading :** cette stratégie fusionne tout simplement la stratégie plain caching et la stratégie cascading.
6. **Fast Spread :** des répliques du fichier sont stockées à chaque nœud sur le chemin au nud demandeur.

Ils ont également introduit trois types de localités :

– **localité temporelle :** les fichiers accédés récemment ont plus de chance d'être demandé à nouveau prochainement.
– **localité géographique :** les fichiers accédés récemment par un client peuvent être probablement demandée par un adjacent.
– **localité spatiale :** les fichiers liés à un fichier récemment accédé sont susceptibles d'être demandés dans un avenir proche.

Ces stratégies ont été évaluées avec différents modèles de données : d'abord, le modèle d'accès sans localité. Deuxièmement, l'accès aux données avec un petit degré de localité temporelle et enfin, l'accès aux données avec un petit degré de localité temporelle et géographique. Les résultats des simulations indiquent que le modèle d'accès différent aux différents besoins des stratégies de réplication. Cascading Replication et Fast Spread ont effectué les meilleurs résultats dans les simulations [47].

Dans [49] une stratégie de réplication basée sur la hiérarchie de la bande passante (BHR) a été présentée, cette stratégie réduit le temps d'accès aux données en maximisant la localité au niveau du réseau et évite sa congestion. Ils ont divisé les sites sur plusieurs régions, où la bande passante inter régions est inférieure à la bande intra régions. Donc, si le fichier requis se trouve dans la même région que le site demandeur, sa récupération sera plus rapide ; dans cette stratégie les répliques sont placé à chaque site demandeur. La stratégie BHR modifié[42] est une extension de la stratégie BHR qui réplique un fichier là où il a été accédé le plus et peut également être utilisé dans un avenir proche. Les résultats ont prouvé que l'algorithme proposé a réduit au minimum le temps de réponse et ont évité les répliques inutiles[50].

Enhance Fast Spread (EFS) [19] est une version améliorée de la stratégie Fast Spread, elle a été proposée pour améliorer le temps de réponse et la consommation de la bande passante, la stratégie EFS garde seulement les répliques importantes tandis que les moins importantes sont remplacées par des répliques plus importantes, l'importance des répliques est déterminée en prenant en compte quatre critère (le nombre et la fréquence des demandes, la taille de la réplique ainsi que la dernière foi où elle a été accédé) et en employant un seuil dynamique qui détermine si la réplique demandée sera stockée à chaque nœud le long de son chemin au demandeur .

La plus part des travaux qui existent se limitent aux topologies hiérarchiques au départ, ce qui exclut une large gamme de grilles de topologies différentes. Aussi les solutions sont statiques, du fait qu'elles fixent certains paramètres (chemin de retour) et ne s'adaptent donc pas à la nature dynamique d'une grille.

Parmi les grilles de topologie hiérarchique qui existent, on peut citer le projet GriPhyN qui utilise un modèle d'arbre de cinq niveaux. Dans cette architecture de grille de données multi-niveaux, le site racine contient tous les fichiers qui sont produits initialement dans la grille de données. Les niveaux suivants constituent le centre national, le centre régional, les groupes de travail et les feuilles qui représentent les bureaux [20].Ce projet a pour but de permettre l'accès aux ressources de calcul pour les grandes expériences scientifiques aux États-Unis [55].

La vraie représentation d'une grille est un graphe général dans lequel il n'existe pas de nœud central désigné comme nœud racine, et chaque nœud peut être connecté à n'importe quel nombre de nœuds, malgré ça les stratégies de réplication sur les topologies non hiérarchiques ne sont pas nombreuses.

Dans [22] les auteurs ont considérés une topologie de grille basée sur un graphe. Lors de la première étape de l'algorithme RPGBA (Replica Placement on Graph-Based data grid Architecture) proposé, cette structure est convertie en structure hiérarchique pour une meilleure gestion des serveurs et de leurs nœuds connexes. Dans un premier temps, le Round Trip Time (RTT) et les coûts de communication entre le site demandeur et le site cible sont calculés. Chaque demande dispose d'un délai spécifique, de sorte que l'algorithme proposé estime la capacité de répondre à la requête avant son échéance. Si (date limite de demande> RTT), l'utilisateur peut accéder au fichier à distance. Sinon, la réplication sera effectuée. Les auteurs ont décidé de répliqué si le nombre de demande est supérieur au seuil qui est défini par le rapport entre le nombre d'accès à une réplique et le nombre de site de chaque niveau, si le site n'a pas d'espace de stockage suffisant alors ils ont utilisé la méthode LRU (Least Recently Used) qui supprime la moins utilisé récemment pour libérer un espace de stockage suffisant pour la réplication.

2.7 Conclusion

Nous avons présenté dans ce chapitre la technique de réplication, son principe et ces avantages par rapport aux systèmes à grande échelle. Nous avons relevé qu'en dépit de la diversité des modèles et approches adoptées, le but est d'améliorer les performances du système. Nous avonségalement mis en évidence les problèmes relatifs à l'utilisation de la réplication dansle contexte des grilles. Nous avons terminé le chapitre par une étude des différents travaux dans le domaine de la réplication dans les grilles de données.

Chapitre 3

Conception et implémentation

3.1 Introduction

L'objectif de notre projet est de contribuer à la gestion des répliques dans les grilles de données en proposant une stratégie efficace de réplication qui permettra d'améliorer le temps de réponse aux requêtes, l'utilisation efficace des ressources réseaux et principalement la capacité de stockage des nœuds ainsi que la gestion des répliques.

3.1.1 Proposition

Il est à noter que la topologie de la grille affecte énormément la technique de réplication implémentée. Cette dernière sera développée selon l'architecture sous-jacente de la grille. Après avoir étudié les stratégies proposées par différents chercheurs pour résoudre ce problème, nous présentons dans ce chapitre la conception et l'implémentation de la stratégie adoptée et ceci en définissant le modèle sur lequel se base notre proposition ainsi que la méthode implémentée. Une grille de donnée peut être représentée par une topologie sous forme de graphe, où chaque nœud peut être connecté a plusieurs nœuds sans restriction. Cette représentation est dépourvue de nœud central et les nœuds sont organisés indépendamment les uns des autres ce qui permet de pallier aux inconvénients de l'architecture hiérarchiques multi-tiers (arbre) et offre une flexibilité dans la communication entre nœuds. En étudiant les différents architectures d'une grille, la plus appropriée est un graphe générale car elle reflète la vraie représentation des grilles actuelles, elle permet d'utiliser les avantages de plusieurs représentation comme les arbres ou les anneaux.

La plus part des chercheurs ont travaillé sur les structures hiérarchisées et ont mentionné l'extension de leurs recherches aux topologies sous forme de graphe.

Dans notre travail, nous considérons qu'une grille est représentée en entrée par un graphe général où il n'existe pas un nœud désigné comme racine, ceci reflète la vraie représentation d'une grille.

Comme mentionné précédemment, le placement des répliques est l'un des défis importants pour améliorer l'exécution des requêtes, et de bonnes stratégies de placement peuvent avoir comme conséquence des gains significatifs dans le temps de réponse aux requêtes. Ce but peut être atteint, quand la disponibilité des données augmente pour chaque serveur. Pour garantir cette disponibilité nous avons travaillé sur la stratégie de réplication Fast Spread qui est une des meilleures stratégies de réplication particulièrement pour les modèles d'accès aléatoires. Dans cette stratégie une réplique du fichier demandé est stockée à chaque nœud le long de son chemin au demandeur. Si l'espace de stockage d'un de ces nœuds est plein, un groupe de répliques existantes sur ces derniers (qui contient une ou plusieurs répliques) doit être remplacé par la nouvelle réplique, le problème qui se pose est que lors de la phase de suppression, des répliques plus importantes pour le nœud que la nouvelle réplique peuvent être supprimées, pour cela dans notre travail nous avons utilisé un seuil dynamique pour déterminer l'importance des répliques ce qui permet de garder les répliques importantes, le calcul de ce seuil dynamique se base sur quatre facteurs qui sont le nombre d'accès à la réplique, la fréquence d'accès à cette réplique, sa taille et la dernière fois où elle a été demandée.

3.2 Stratégie de réplication proposée

3.2.1 Topologie de la grille

Dans notre travail, nous considérons une grille avec une topologie basée sur un graphe. À la première étape de l'algorithme implémenté, cette structure est convertie en une structure hiérarchique pour une meilleure gestion des nœuds (le parcours lors de la recherche des répliques est plus aisé) tout en permettant le transfert de données entre les nœuds frères, contrairement à la structure d'arbre connue, où la racine doit comporter un nœud particulier qui contient toutes les répliques et de ce fait la racine doit avoir un espace de stockage théoriquement infini, la racine du pseudo arbre obtenu par l'algorithme n'est pas particulière et ne diffère pas des autres nœuds. Pour le choix de cette racine, nous nous sommes basés sur un seul critère qui nous semble le plus important : c'est la connectivité. En effet, la racine est le v qui est le plus relié aux autres nœuds ceci nous évite d'avoir trop de niveaux hiérarchique et donc un parcours plus simple.

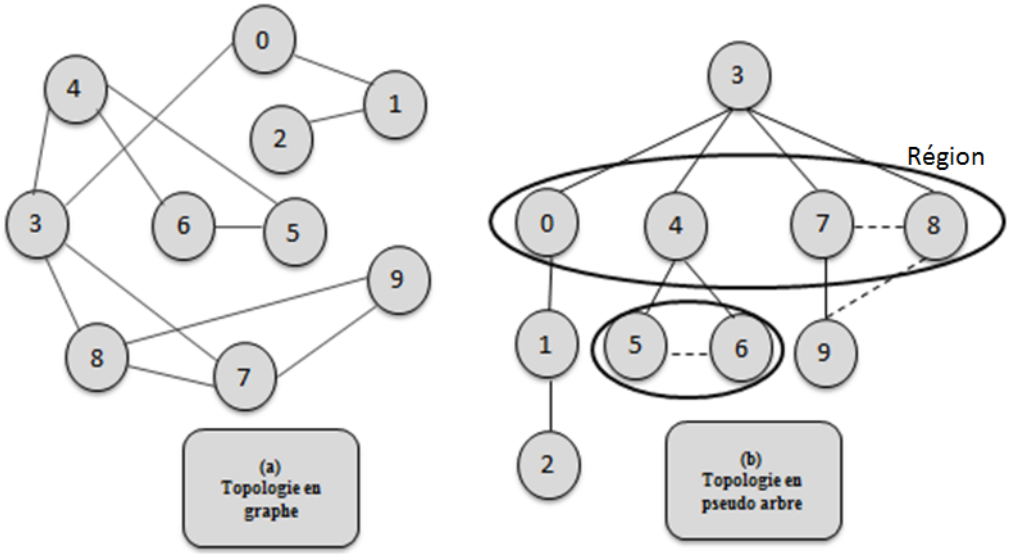

FIG. 3.1: La topologie de la grille

3.2.2 modèle de réplication

Hypothèses

- La grille est représentée sous forme de graphe.
- Chaque nœud a son propre identifiant, un élément de stockage (ES) et un élément de calcul (EC).
- Un nœud ne peut pas tomber en panne.
- Les requêtes sont en lecture.

Architecture Fonctionnelle du modèle de réplication

La figure 2 montre les différents services conçus. Une étape préliminaire précède l'étape de la réplication. Il s'agit d'un pré-traitement qui consiste en l'implémentation d'un algorithme de parcours pour passer de la topologie graphe (Figure 1 (a)) à la toplogie pseudo arbre (Figure 1 (b)). L'étape de traitement comprend la création des répliques et leur placement.

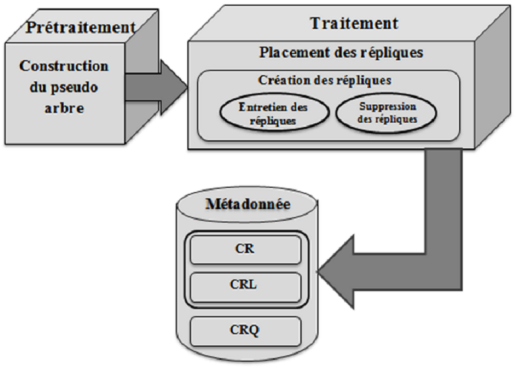

FIG. 3.2: L'architecture fonctionnelle du modèle de réplication

1. **Métadonnées :** Les métadonnées sont les données qui servent à décrire l'ensemble des informations contenues dans la grille. Ces métadonnées sont représentées sous forme de tables et de catalogues :
   – **CRL (Catalogue des répliques local)** : ce catalogue comporte les informations qui décrivent les répliques existantes dans un nœud.
   – **CR (Catalogue des répliques)** : ce catalogue comporte les informations qui décrivent les répliques existantes dans la sous arborescence du nœud contenant ce catalogue tel que leurs identificateurs et où elles se trouvent.
   – **CRQ (Catalogue des requêtes)** : les informations de ce catalogue représentent les requêtes provenant des autres nœuds qui requièrent une réplique. Ces informations sont la taille des requêtes, et la réplique concernée.
2. **Construction de l'arbre**
   Notations
   – **Sommet :** racine du pseudo arbre.
   – **Marqué :** liste qui permet de marquer les nœuds parcourus.
   – **Voisin(i) :** liste qui contient les voisins d'un nœud i.

Le but de cette phase de prétraitement est de créer une représentation logique en pseudo arbre du graphe ce qui permet comme nous l'avons précisé précédemment une meilleure gestion de la grille.Pour le choix de la racine on ne considère que le nombre de connectivité. Pour la construction de l'arbre nous parcourons les nœuds en profondeur en commençant par la racine et en marquant les nœuds qui n'ont pas encore été visités.

Algorithme 1 **Algorithme de construction de l'arbre**

```
1 : Begin
2 : Marqué.ajouter(sommet);
3   : For All noeud(i) ϵMarqué do
4:ForAll noeud(j) ϵ Voisin_i(noeud(i))do
5:IFMarqué.NeContientPas(noeud(j))do
6:Marquer.ajouté(noeud(i));
7   :          End if;
8   :       End For;
9 : End For;
10 : End;
```

FIG. 3.3: pseudo code de l'algorithme de construction de l'arbre

Exemple du choix de la racine :
Voisin0 :(1, 3).Voisin1 :(0, 2).Voisin2 :(1).Voisin3 :(0, 4, 7, 8).Voisin4 :(3, 5, 6). Voisin5 :(4, 6). Voisin6 :(4, 5).Voisin7 :(3, 9).Voisin8 :(3, 7, 9).Voisin9 :(7, 8).

Le nœud qui a le plus de connectivité est le nœud 3 car il a quatre voisins donc on le considère comme racine (voir Figure 1).

3. **Placement des répliques**
   - **Création des répliques :** Dans notre stratégie, à l'arrivée d'une requête si le nœud recevant cette requête contient la réplique demandée il l'utilise et la requête est exécutée, sinon la réplique est cherchée dans la sous arborescence du nœud si elle n'est pas trouvée, on passe au père et on refait le même processus jusqu'à ce qu'on trouve la réplique demandée tout en marquant les nœuds visités. A la fin nous parcourons les nœuds marqués et s'ils ont un espace de stockage libre supérieur à la taille de la réplique demandée on crée une copie de cette dernière sur ces nœuds.

     Notations
     - **ES(i) :** l'espace de stockage total d'un nœud i.
     - **EL(i) :** l'espace de stockage libre d'un nœud i.
     - **N :** le nombre de réplique dans le groupe.
     - **NAR(i) :** nombre d'accès à la réplique i dans le groupe.
     - **T(i) :** la taille de la réplique i.
     - **ITSF :** intervalle de temps spécifique pour le calcul de la fréquence.
     - **TC :** le temps courant.
     - **TDA(i) :** le temps du dernier accès à la réplique i dans le groupe.
     - **NARR(i) :** le nombre d'accès à la réplique i dans la région.
     - **NR(i) :** nombre de nœud dans la région i.
     - **LR :**liste des répliques.
     - **NARITSF(i) :** nombre d'accès à la réplique i dans le groupe dans ITSF.
     - **RD :** réplique demandée.
     - **Selectionné :** liste qui contient le groupe de répliques à supprimer ou à déplacer.
     - **Région :**ensemble des nœuds qui ont le même père.

Algorithme 2 **Algorithme de création des répliques.**

```
1.  : Begin
2.  : Arrivée d'une requête (i) ;
3.  : Réplique_T :=requête (i).réplique ;
4.  : If Réplique_T existe sur CRL du nœud demandeurThen l'utiliser ;
5.  : Else
6.  : While Réplique_T n'est pas trouvé sur CRL du nœud visité
7.  : Chemin.ajouter(nœud) ;
8.  : Chercher dans CR de ce nœud ;
9.  : passé au père ;
10. :End While
11. : End If;
12. : For all nœud_i∈ chemin do
13. : IfEL(nœud_i) > taille de Réplique_TThen
14. :noeud_i.CréerRéplique (réplique_T) ;
15. : End If;
16. : End for;
17. :End;
```

FIG. 3.4: pseudo code de l'algorithme de création des répliques.

   - **Entretien et suppression des répliques :** Lorsqu'une création de réplique est requise alors que l'espace libre de stockage est insuffisant, une suppression est nécessaire ; pour cela un groupe de réplique qui a une taille supérieure ou égale à T(i) doit être sélectionné. Comme mentionné précédemment, la stratégie Fast Spread supprime ce groupe sans conditions alors qu'il peut être plus important pour le nœud que la nouvelle réplique. Pour remédier à ce problème, la stratégie proposée calcule l'importance d'une réplique (VRi) pour un nœud par rapport aux répliques qu'il contient (VG).

     L'importance d'une réplique R est définie par quatre facteurs qui sont :

     - Le nombre d'accès à la réplique R.
     - La fréquence d'accès à la réplique R.
     - La taille de la réplique R.
     - La dernière fois où la réplique R a été accédée.

$$VG = \frac{\sum_{i=1}^{N} NAR_i}{\sum_{i=1}^{N} T_i} + \frac{\sum_{i=1}^{N} NAR\,ITSF_i}{ITSF} + \frac{1}{TC - \frac{\sum_{i=1}^{N} TDA_i}{N}} \quad (1)\ [19]$$

$$VRi = \frac{NARi}{Ti} + \frac{NARFITSi}{FITS} + \frac{1}{TC - TDAi} \quad (2)\ [19]$$

De plus, dans la stratégie proposée, certain types de répliques ne peuvent pas être supprimées. Il s'agit des répliques bloquées. Nous définissons une réplique bloquée comme une donnée à la fois importante et fréquemment accédée. Pour tester si une réplique est bloquée nous avons proposé un seuil dynamique pour déterminer si une réplique est importante au sein d'une même région, les répliques qui ont un nombre d'accès supérieur à ce seuil sont dites bloquées et ne peuvent pas être supprimées.

$$S = \frac{NARRi}{NR}$$

Afin de libérer un espace de stockage suffisant pour répliquer, on sélectionne un groupe de répliques qui doit avoir une taille supérieure ou égale à la taille de la nouvelle réplique en commençant par les répliques non bloquées.Si après cela, l'espace ne suffit toujours pas on sélectionne les répliques bloquées une par une jusqu'à arriver à l'espace requis. Si la nouvelle réplique est plus importante que le groupe de réplique sélectionné alors ce dernier sera remplacé par cette réplique en supprimant les répliques non bloquées et en déplaçant celles qui sont bloquées vers l'un des nœuds de la même région qui a suffisamment d'espace de stockage libre sinon le groupe de répliques sélectionnées est considéré comme plus important pour le nœud que la nouvelle réplique donc il faut le garder et on passe au nœud suivant sur le chemin de retour sans répliquer.

Algorithme 3 **Algorithme d'entretien et desuppression des répliques.**

```
1.  : Begin
2.  : E ← 0 ;
3.  :For all replique_i ∈LR do
4.  :     If replique_i non bloquée then
5.  : If E < RD.taillethen
6.  :E ← E + replique_i .taille;
7.  :         selectionné.ajouter(replique_i );
8.  : End if;
9.  :    End if;
10. : End for;
11. : If E <RD.Sizethen
12. : For all replique_i ∈LR do
13. :     If replique_i  bloquéethen
14. :        If E < RD.taille then
15. :        E ← E + replique_i .taille;
16. :        selectionné. ajouter (replique_i );
17. : End if;
18. : End if;
19. : End for;
20. : End if;
21. : IfE >RD.Sizethen
22. : calculer VR et VG ;
23. :    IfVR>VG then
24. : For all réplique_i∈selectionné
25. : Ifréplique bloquéethen
26. :    chercher dans la même région le nœud qui a EL > ∑_{i=1}^{n} replique_i . taille
27. :          déplacer le groupe de réplique vers ce nœud
28. :       else
29. :          supprimer replique_i
30. : End if;
31. : End;
```

FIG. 3.5: pseudo code de l'algorithme de suppression des répliques.

3.3 Conception et modélisation UML

Pour décrire les différentes étapes de conception de notre modèle de réplication dans une grille de données, nous avons opté pour le langage UML(Unified Modeling Language). UML est un formalisme graphique issu de notations employées dans différentes méthodes objets, il sert à décomposer le processus de développement, séparer l'analyse de la réalisation, prendre en compte l'évolution de l'analyse et du développement, mettre en relation les experts métiers et les analystes, et coordonner les équipes d'analyse et de conception. L'emploi de l'UML facilite grandement le développement d'applications dans des langages dirigés par les objets en permettant d'en établir un modèle formel et facile à comprendre.

3.3.1 Diagramme de classes du modèle proposé

Le diagramme de classes constitue un élément très important de la modélisation : il permet de définir quelles seront les composantes du système final. Un diagramme de classes proprement réalisé permet de structurer le travail de développement de manière très efficace, il permet aussi, dans le cas de travaux réalisés en groupe, de séparer les composantes de manière à pouvoir répartir le travail de développement entre les membres du groupe. Enfin, il permet de construire le système de manière correcte.

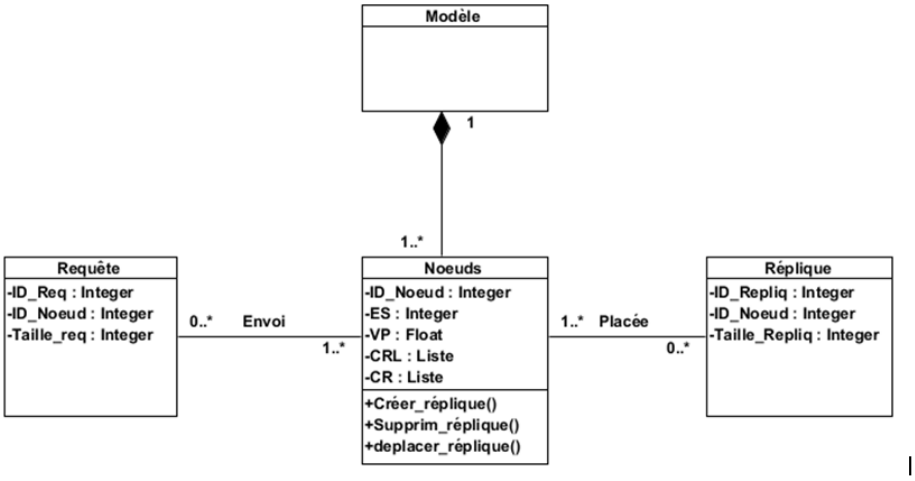

FIG. 3.6: Le diagramme de classes du modèle de réplication.

3.3.2 Diagramme d'activité du processus de réplication

Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Les diagrammes d'activités permettent de spécifier des traitements apriori séquentiels Ainsi, ils peuvent être utiles dans la phase de réalisation car ils permettent une description précise des opérations qu'autorise la génération automatique du code. Nous utilisons le diagramme d'activité pour représenter le processus de réplication qui est composé de deux sous-processus :

- **La création de la réplique :** ce processus représente les traitements effectués avant de prendre la décision de répliquer une donnée.
- **Le placement de la réplique :** ce processus consiste à décider du placement de la réplique en question, selon la stratégie.

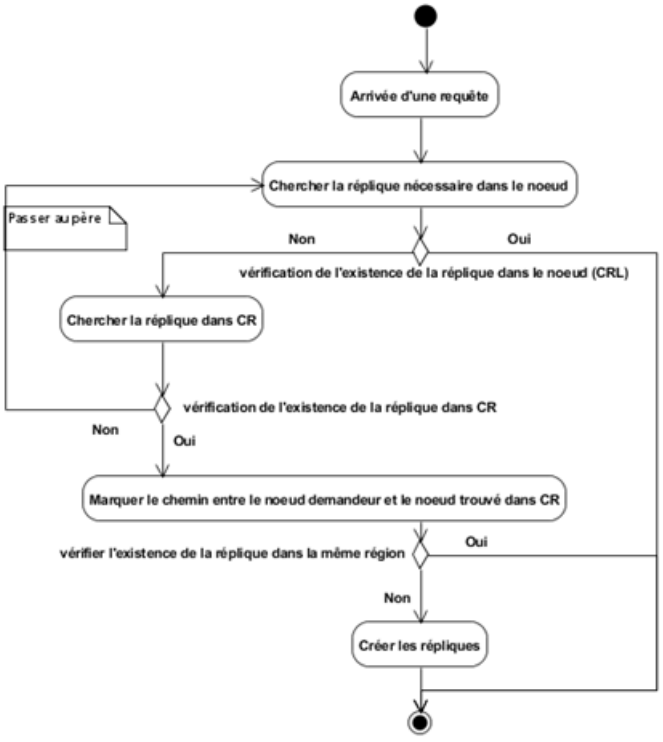

FIG. 3.7: Le diagramme d'activité de la création de répliques.

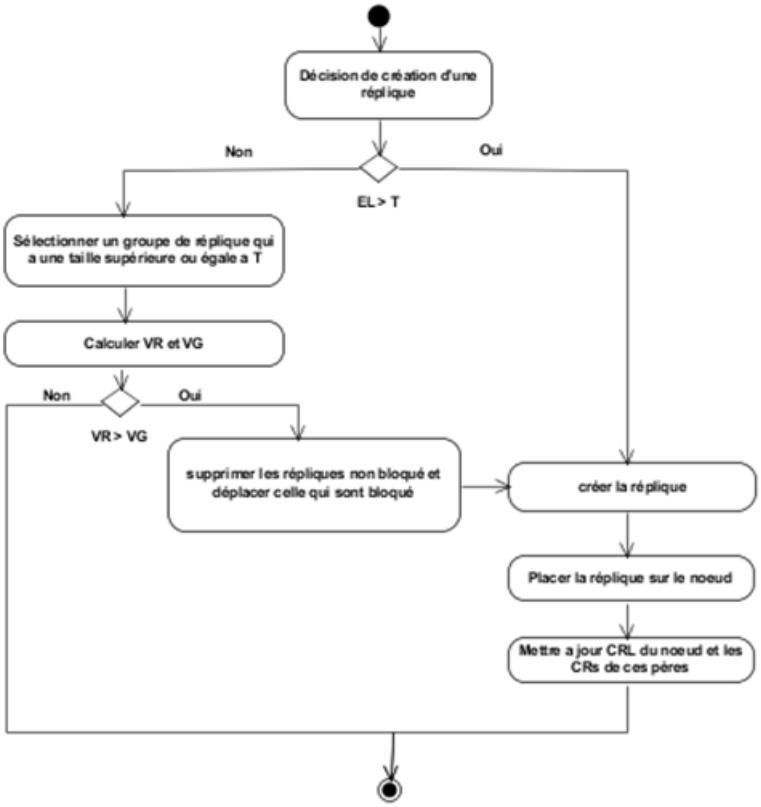

FIG. 3.8: Le diagramme d'activité du placement de répliques.

3.4 Environnement de développement et d'expérimentation

- **Environnement de développement :** Nous avons opté pour NetBeansIDE en sa version 7.0qui est un environnement de développement très puissant basé sur un compilateur JAVA. La particularité principale du langage java est que les logiciels écrits avec ce langage sont très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS . De plus Java dispose d'une riche bibliothèque de composants visuels. Ilpermet ainsi de bénéficier de tous les avantages de la programmation orientée-objet.
- **Simulateur utilisé :** Nous avons mis en œuvre nos algorithmes sur le Simulateur OptorSim [51] qui est réalisé avec Java et qui a été développé dans le cadre du projet DataGrid. Il reproduit d'ailleurs l'architecture de ce dernier qui est fondé sur une architecture modulaire. Chaque ressource de traitement est implémentée par un processus.
- **Environnement matériel et Logiciel :** Environnement matériel et logiciel : Nous avons développé notre application sur une machine fonctionnant sous le système Windows 7 avec les caractéristiques suivantes :
  - **Processeur :** Intel Core I3 M 2.4GHz.
  - **RAM :** 4 Go.

3.5 Métriques utilisées

Pour valider notre travail et évaluer le comportement de notre approche, nous avons utilisé les métriques suivantes :

(a) **Temps de réponse :** Cette métrique permet de mesurer le temps de réponse relatif à l'exécution d'une requête. Le temps de réponse d'une requête i est le temps passé entre l'envoi de la requête et la réception de la réponse.

Temps de réponse i

= Temps d'envoi du requête i + Temps d'attente de la requête i
+ Temps d'exécution de la requête + Temps d'envoi de la réponse i

(b) **Moyenne totale de temps de réponse :** Cette métrique représente la moyenne de temps de réponse pour toute la simulation.

$$\textbf{Temps de réponse moyen} = \frac{\sum_{j=1}^{Nb\_req.} \text{Temps de réponse j}}{Nb\_req.}$$

(c) **Espace de stockage libre :** Ce paramètre est défini par le pourcentage de l'espace de stockage libre par rapport à l'espace disponible. Il est affecté par le nombre de répliques créées.

$$\textbf{Espace\_libre\_Grille} = \sum_{j=1}^{NB} \text{Espace}_{\text{Libre}_{\text{Noeud(j)}}}$$

$$NB = \text{nombre des nœuds}$$

(d) **La consommation des ressources réseau :** Ce paramètre indique l'occupation et l'usage du réseau, il est défini par la formule suivante :

$$\textbf{Usage\_Réseau} = \frac{\textbf{Nombre\_accès\_distant} + \textbf{NombreRéplique}}{\textbf{Nombre\_accès\_locaux}}$$

(e) **Nombre de répliques transférées :** Cette métrique représente le nombre de répliques transférées d'un nœud vers un autre au lieu d'être supprimées. Elle mesure donc l'impact du service d'entretien proposé dans la stratégie.

(f) **Nombre de répliques créées :** Cette mesure compte le nombre de répliques créées par la stratégie proposée afin de le comparer avec d'autres stratégies.

3.6 Description du fonctionnement de notre logiciel

Dans ce qui suit, nous présenterons les différentes fonctionnalités de notre logiciel à travers quelques interfaces :

3.6.1 Fenêtre d'accueil

FIG. 3.9: Fenêtre d'accueil de l'application.

Cette fenêtre représente la première l'interface de notre simulateur.

3.6.2 Configuration d'une grille

La figure 10 illustre les deux fenêtres principaux qui permettent de configurer une grille en introduisant plusieurs paramètres tels que :

– Le nombre des nœuds, leurs capacités de stockage et vitesses de calcul.
– Le débit de bande passante.
– Le nombre de répliques initiales et leurs tailles.

FIG. 3.10: Configuration et creation d'une grille.

3.6.3 Affichage des configurations de la grille

Cette interface permet de visualiser les différentes configurations de la grille.

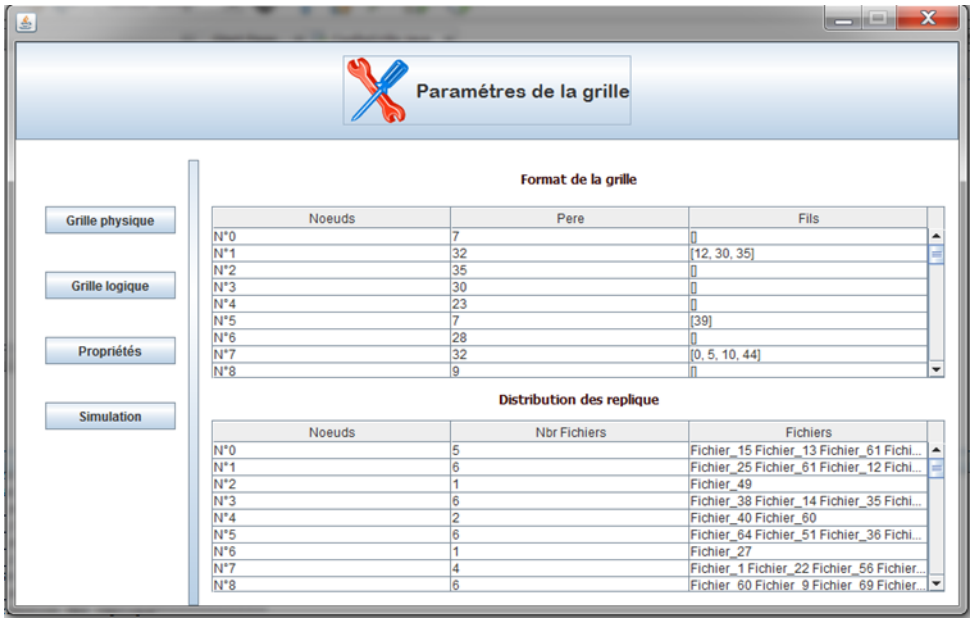

FIG. 3.11: Visualisation des configurations de la grille.

Notre application permet à l'utilisateur de visualiser la topologie physique donnée en entrée ainsi que la topologie logique obtenue par l'exécution de l'algorithme " construction du pseudo-arbre " en cliquant sur le boutant " Grille physique " ou le boutant "Grille logique".

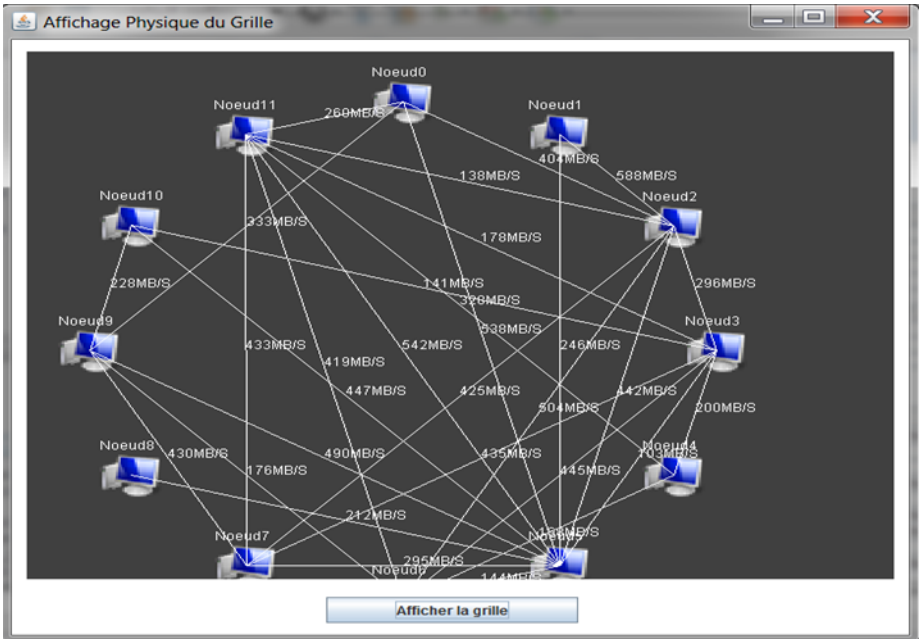

FIG. 3.12: Grille physique.

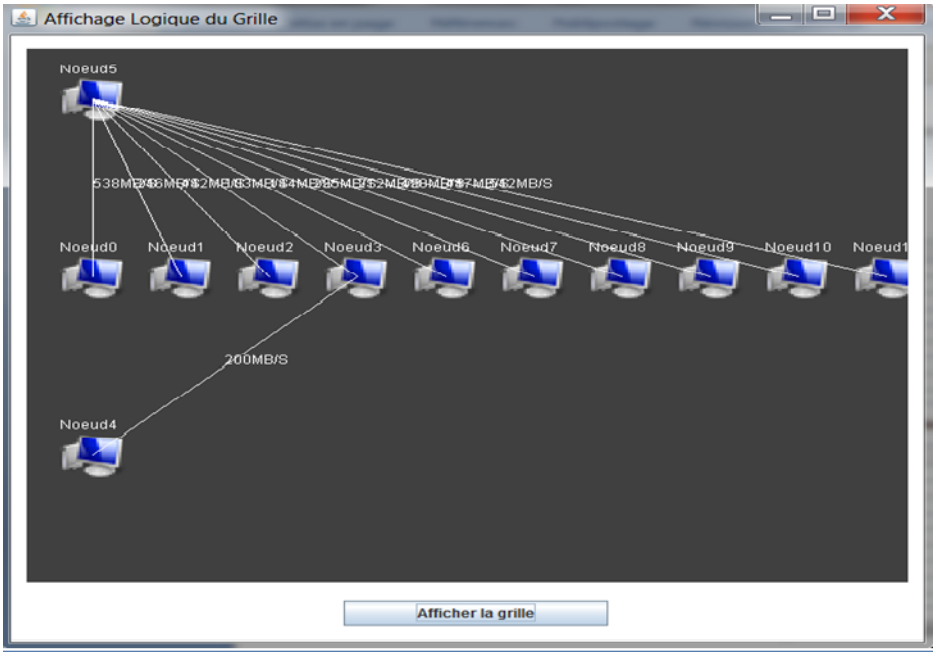

FIG. 3.13: Grille logique.

Pour afficher les propriétés d'un nœud de la grille telles que sa capacité de stockage, sa vitesse du processeur et les fichiers qu'il contient, on clique sur le bouton " Propriétés ". Une nouvelle fenêtre s'affichera où l'utilisateur peut choisir le nœud dont il veut voir les paramètres.

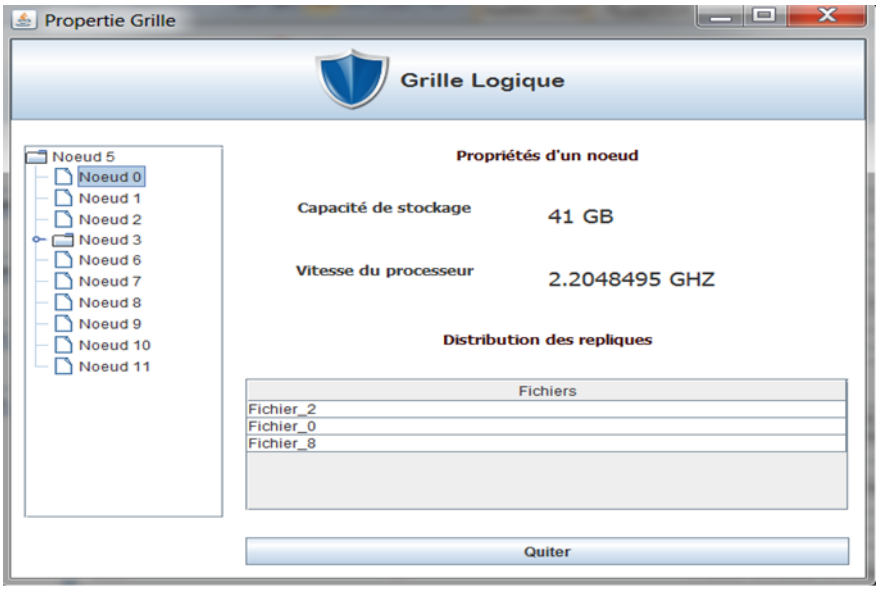

FIG. 3.14: Propriétés d'un nœud.

3.6.4 Lancement de la simulation

Le bouton " Simulation " permet de lancer la simulation. Une nouvelle fenêtre s'affiche où l'utilisateur introduit les informations nécessaires.

FIG. 3.15: Lancement de la simulation.

Pour afficher les différents résultats de la simulation, on doit choisir l'une des mesures présentes dans l'interface illustrée ci-dessous et cliquer sur le bouton " Résultats " qui s'affiche à la fin de la simulation.

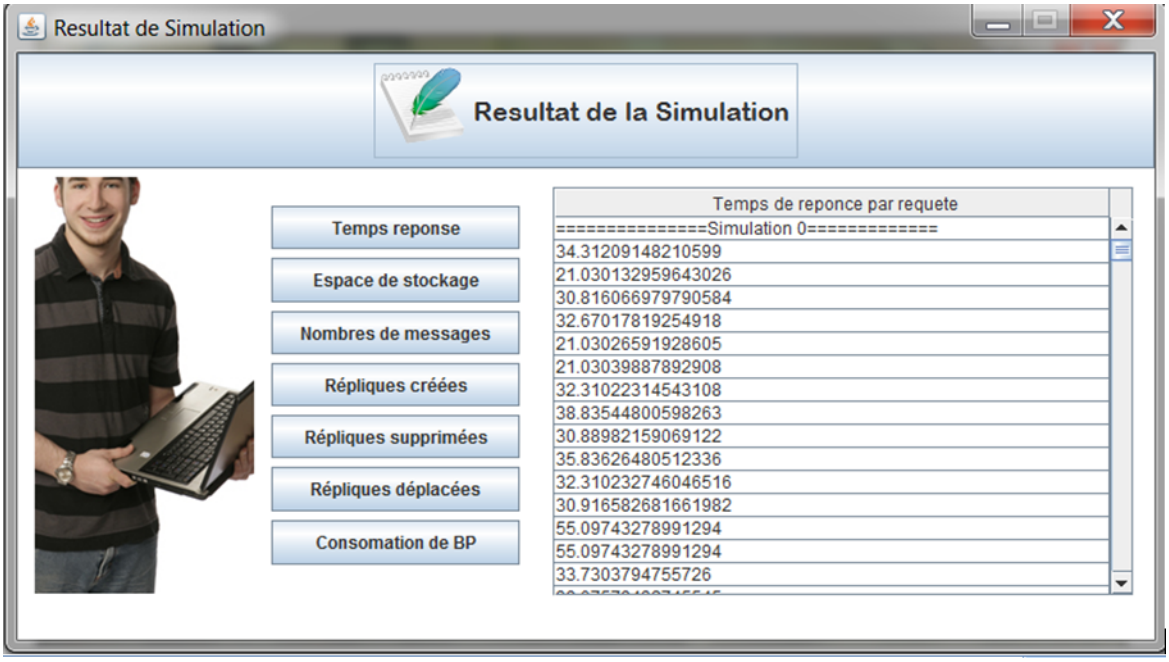

FIG. 3.16: Résultats de la simulation.

3.7 Etude expérimentale

Nous présentons dans cette partie les différents résultats obtenus à partir des expérimentations que nous avons effectuées. La comparaison des résultats est réalisée principalement entre cinq stratégies :

- **Sans réplication :** les répliques sont distribuées de façon statique et sont accédées à distance.
- **Fast Spread :** cette stratégie réplique tout au long du chemin de retour au demandeur en supprimant sans conditions s'il n'y a pas d'espace de stockage suffisant.
- **Fast Spread avec LRU :** cette stratégie réplique tout au long du chemin de retour au demandeur en supprimant les répliques les moins fréquemment utilisées.
- **Enhanced Fast Spread :** cette stratégie réplique tout au long du chemin de retour au demandeur en tenant compte de l'importance des répliques lors de la suppression.
- **Strategie proposée :** (FSGB : Fast Spread Based Graph).

3.7.1 Paramètres de simulation

Afin d'effectuer les expérimentations, nous avons utilisé un certain nombre de paramètres dont les valeurs sont définies dans le tableau ci-dessous :

Paramètre	Valeur
Nombre des nœuds	[10..1100]
Débit de bande passante intra-cluster	[40..1000] Mb/S
Capacités de stockage des nœuds	[10..180] Gb
Vitesse des processeurs	[2.0..4.0] GHz
Nombre de données	[10..3000]
Nombre de répliques	[4..20]
Tailles des répliques	[1..4] Gb
Nombre de requêtes	[50..1200]
Taille des requêtes	[100..1024] Ko

TAB. 3.1: Paramètres de simulation

3.7.2 Paramètres des expériences :

Les résultats présentés ci-dessous ont été obtenus avec les paramètres suivant :

Paramètre	Valeur
Nombre des nœuds	200
Débit de bande passante intra-cluster	[50..600] Mb/S
Capacités de stockage des nœuds	[30..45] Gb
Vitesse des processeurs	[2.0..4.0] GHz
Nombre de données	100
Nombre de répliques	5
Tailles des répliques	[1..4] Gb
Nombre de requêtes	600
Taille des requêtes	200 Ko

TAB. 3.2: Paramètres des expériences

3.7.3 Expérience 1 : les temps de réponse

Dans cette première expérience, nous mesurons le temps de réponse moyen obtenu avec les stratégies implémentées. La figure 17 illustre le temps de réponse moyen obtenu pendant toute la simulation. Ces résultats montrent que : Le temps de réponse pour la stratégie Sans réplication est effectivement élevé car les nœuds accèdent toujours aux répliques à distance. Par contre Fast Spread augmente la disponibilité des répliques ce qui améliore le temps de réponse mais cette stratégie ne tient compte d'aucun critère lors de la suppression ce qui peut amener à perdre des répliques nécessaires. Notre stratégie est largement meilleure que les autres car elle réduit le taux de réplication du même fichier dans une même région et transfère les fichiers importants se qui a comme impact de réduire le temps de réponse.

Stratégie	Temps de réponse (ms)
Sans Réplication	166.327
Fast Spread	151.206
Fast Spread avec LRU	116.977
Enhenced Fast Spread	93.316
Stratégie Proposée	71.987

TAB. 3.3: résultats de l'expérience 1

D'après le tableau ci-dessus, le gain apporté par notre proposition concernant le temps de réponse moyen aux requêtes est de 56.73

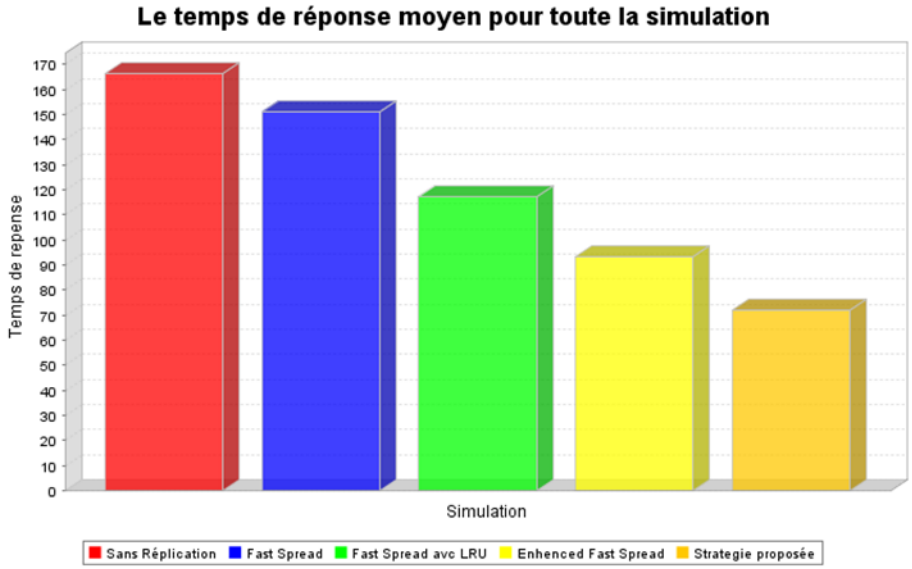

FIG. 3.17: Le temps de réponse de toute la simulation.

3.7.4 Expérience 2 : le nombre de messages échangés

Dans cette expérience, nous étudions l'apport de notre stratégie concernant le nombre de messages échangés entre sites afin d'exécuter les requêtes. La courbe de la figure 18 montre que la réplication permet de réduire le nombre de messages.

Malgré que la stratégie proposée donne de meilleurs résultats par rapport aux autres stratégies (sans réplication, Fast Spread, Fast Spread avec LRU), nous constatons que le schéma s'inverse par rapport à Enhanced Fast Spread et ceci s'explique très simplement par le fait que dans notre stratégie,on accède à distance dans la même région ce qui augmente le nombre de messages mais permet également :

– D'éviter de trop répliquer et ceci a l'avantage de diminuer le temps de réponse d'exécution des requêtes comme nous l'avons constaté dans l'expérience précédente.
– De réduire le nombre de requêtes créées. (voir expérience 3)

Cette expérience confirme que les objectifs qu'on voudrait atteindre en utilisant la réplication sont conflictuels.

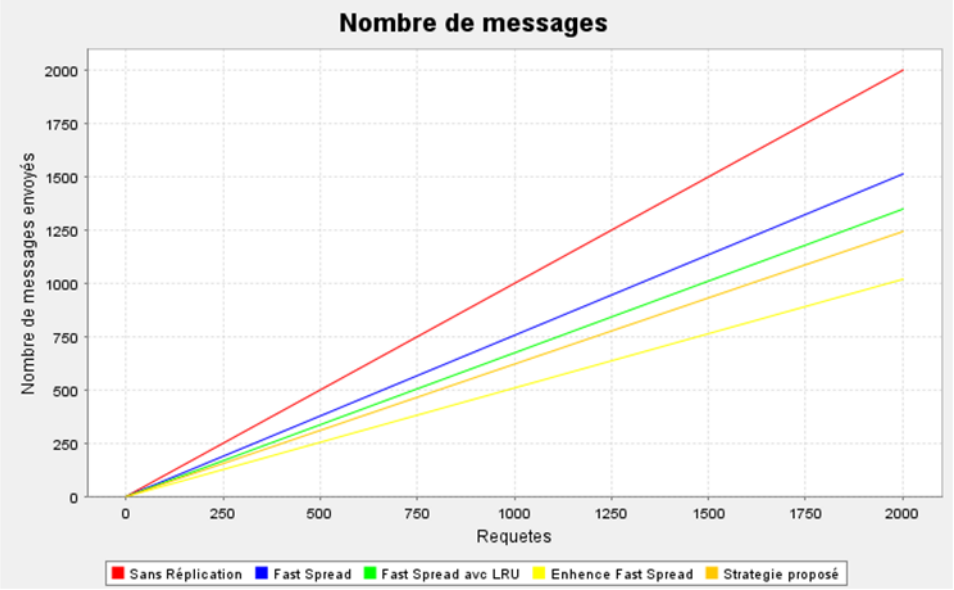

FIG. 3.18: Le nombre de messages échangés.

3.7.5 Expérience 3 : Le nombre de répliques créées

Dans cette expérimentation, nous mesurons le nombre de répliques créées. La figure 19 montre que la stratégie proposée crée moins de répliques par rapport aux autres stratégies car elle prend en considération l'importance des répliques et ne réplique pas dans la même région.

Le tableau ci-dessous donne le nombre de répliques créés pour chaque stratégie, Notons que notre stratégie réduit ce nombre de 37,58

Stratégie	**Répliques créées**
Fast Spread	556
Fast Spread avec LRU	525
Enhenced Fast Spread	517
Stratégie Proposée	347

TAB. 3.4: Résultats de l'expérience 3

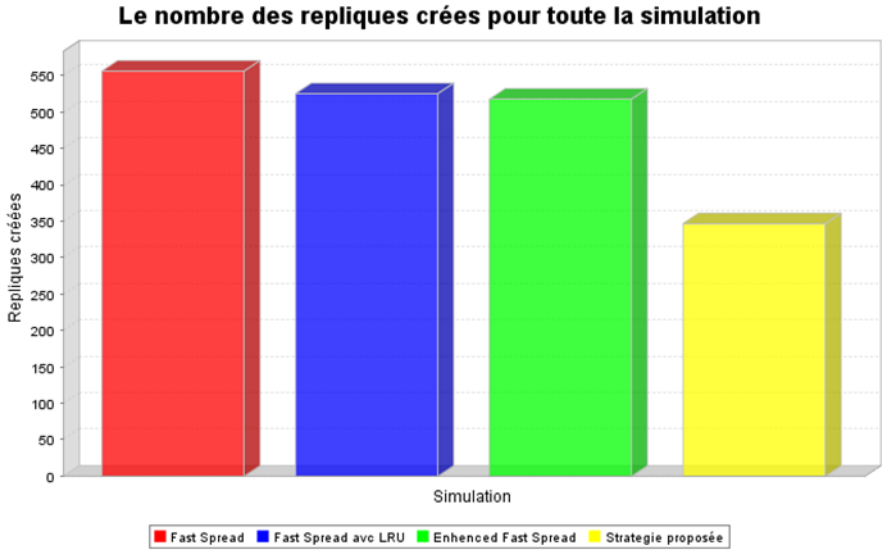

FIG. 3.19: Nombre de répliques créées.

3.7.6 Expérience 4 : Le nombre de répliques supprimées

Dans cette expérience, nous mesurons le nombre de répliques supprimées. La figure 20 montre que le taux de suppression de la stratégie proposée (FSBG) est inferieur par rapport aux autres stratégies et ceci car c'est la seule stratégie qui déplace les répliques en intégrant la notion de " répliques bloquées " au lieu de les supprimer. Le nombre de répliques supprimées est diminué de la moitié par rapport à la stratégie Fast Spread, et de 39.66

Stratégie	**Répliques supprimées**
Fast Spread	522
Fast Spread avec LRU	425
Enhenced Fast Spread	421
Stratégie Proposée	254

TAB. 3.5: Résultats de l'expérience 4

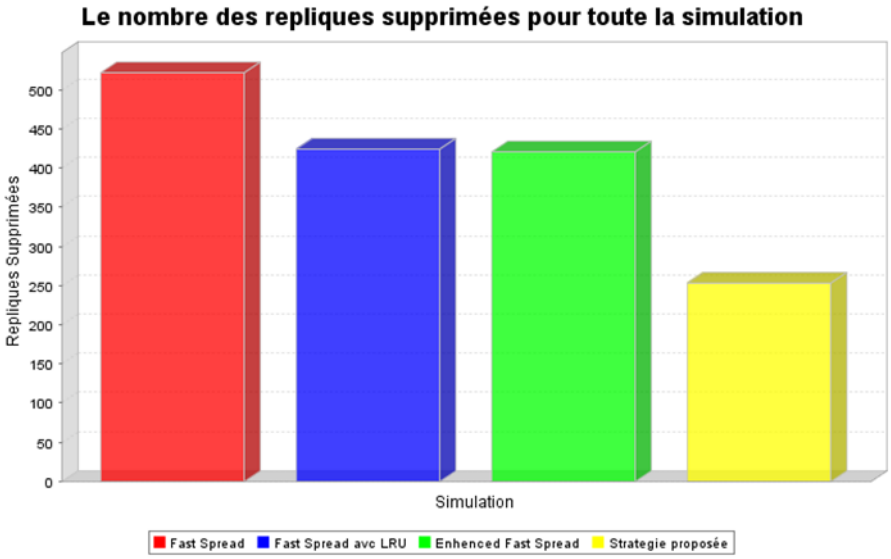

FIG. 3.20: Nombre de répliques supprimées.

3.7.7 Expérience 5 : Le nombre de répliques déplacées

Dans cette expérience nous mesurons le nombre de répliques déplacées dans chaque stratégie. Nous remarquons que seule la stratégie proposée (FSBG) transfert les répliques faute d'espace de stockage. Il s'agit-là d'une gestion des répliques déjà existantes dans la grille pour conserver les répliques les plus importantes d'une part et pour économiser le temps de création des répliques d'autre part. Ce temps influe directement sur le temps de réponse des requêtes.

Stratégie	**Répliques déplacées**
Fast Spread	0
Fast Spread avec LRU	0
Enhenced Fast Spread	0
Stratégie Proposée	22

TAB. 3.6: Résultats de l'expérience 5

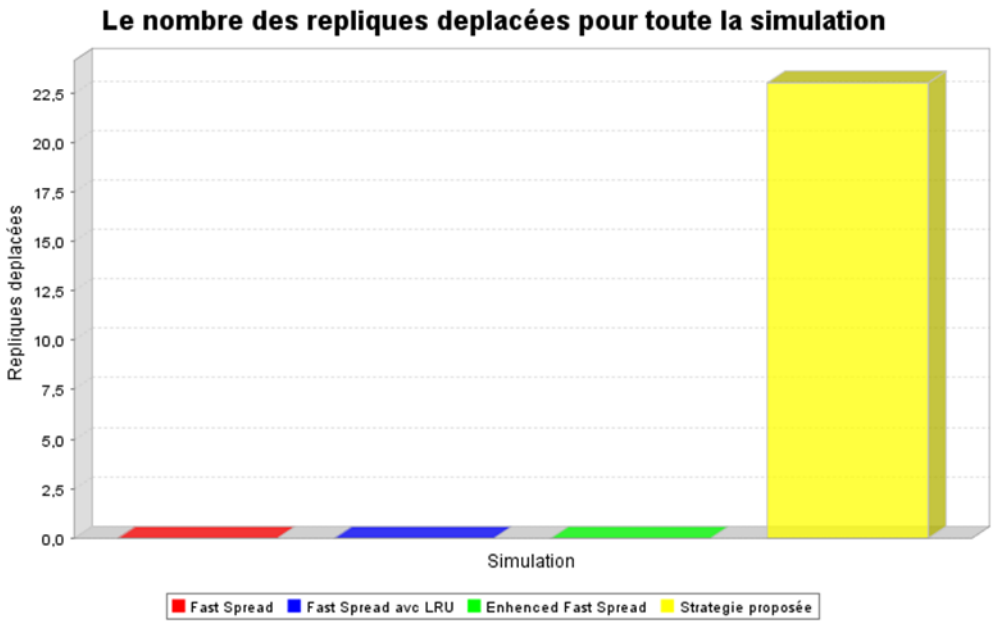

FIG. 3.21: Le nombre de répliques déplacées.

3.7.8 Expérience 6 : Les espaces de stockage

Stratégie	Percentage Espace libre	Percentage Espace utilisé
Fast Spread	25.656	74.344
Fast Spread avec LRU	25.31	74.69
Enhenced Fast Spread	25.499	74.501
Stratégie Proposée	29.145	70.885

TAB. 3.7: Résultats de l'expérience 6

Le but de cette expérience, est de voir la variation de l'espace de stockage libre de la grille dans chaque stratégie. Ce résultat est directement lié avec le nombre de répliques créées ainsi que leurs tailles.

Nous pouvons voir dans la figure 22 que l'espace de stockage libre de la grille est le plus important dans la stratégie (FSBG) car cette stratégie ne réplique pas les données qui existent dans la même région. Malgré que la stratégie proposée transfert quelques répliques au lieu de les supprimées l'espace de stockage libre reste plus élevé que pour les autres stratégies.

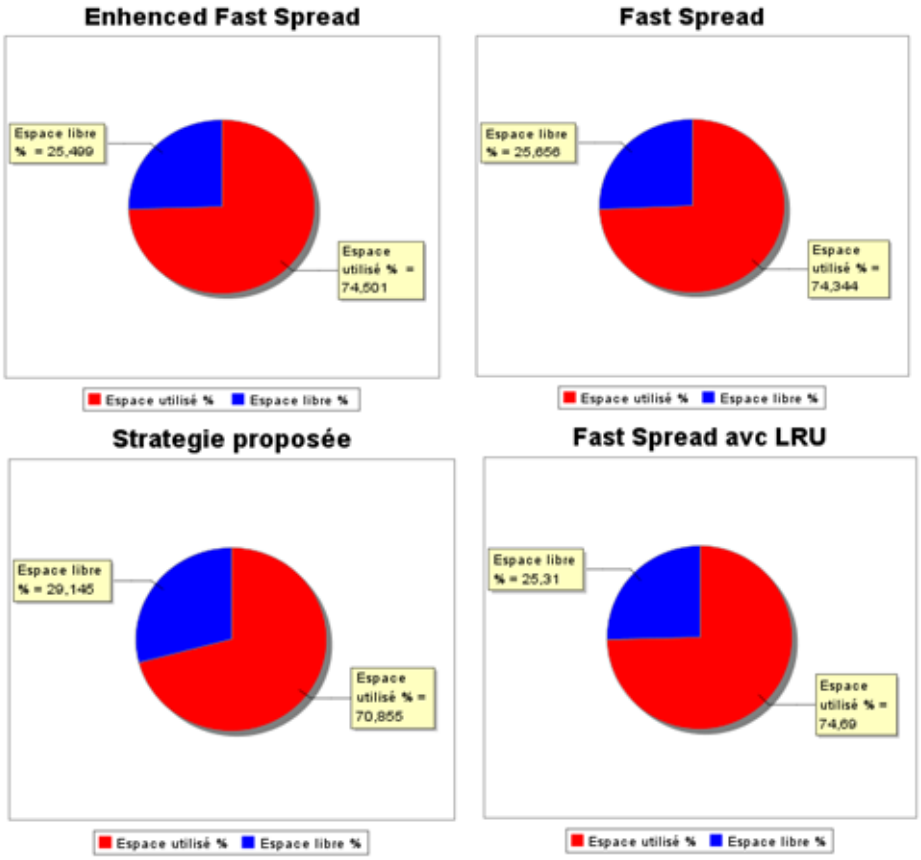

FIG. 3.22: Espaces de stockage.

3.7.9 Expérience 7 : les temps de réponse par requête

Les résultats présentés ci-dessous ont été obtenus avec les paramètres suivant :

Paramètre	Valeur
Nombre des nœuds	400
Débit de bande passante intra-cluster	[50..600] Mb/S
Capacités de stockage des nœuds	[30..45] Gb
Vitesse des processeurs	[2.0..4.0] GHz
Nombre de données	100
Nombre de répliques	5
Tailles des répliques	[1..4] Gb
Nombre de requêtes	1000
Taille des requêtes	200 Ko

TAB. 3.8: Paramètres des expériences

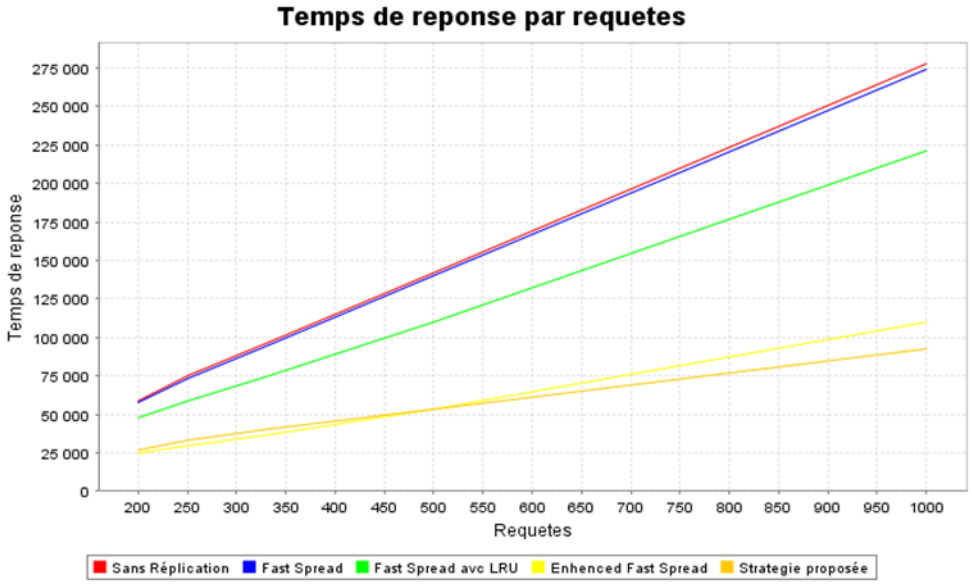

FIG. 3.23: Temps de réponse moyen avec une variation de nombre de requêtes.

Cette expérience étudie l'impact de la variation du nombre de requêtes sur le temps de réponse moyen. Nous avons fait varier le nombre de requêtes de 200 à 1000 par pas de 50. Nous remarquons d'après la figure 23 que la courbe de l'approche proposée a une tendance linéaire tout comme les autres courbes. De plus notre stratégie, réduit le temps de réponse par rapport à toutes les autres stratégies. Le graphique de la figure 31 montre un résultat intéressant : la stratégie Enhence Fast Spread donne de meilleurs résultats au début de la simulation mais la tendance s'inverse au profit de la stratégie proposée quand le nombre de requêtes augmente (500 requetes) ceci est du aux contributions apportées à la méthode.

3.8 Conclusion

Dans ce dernier chapitre nous avons d'abord présenté notre environnement de travail pour évaluer la stratégie proposée de création et de placement dynamique des répliques, par la suite nous avons détaillé l'implémentation de la stratégie et à la fin nous avons clôturé ce chapitre par la présentation des résultats des expérimentations réalisées.

L'analyse des résultats expérimentaux ont révélés une amélioration considérable dans les temps d'exécution des requêtes grâce au rapprochement des données nécessaires à leur exécution. De plus la stratégie proposée a donné des résultats satisfaisants pour les autres métriques analysées.

CONCLUSION GÉNÉRALE

Les grilles de calcul et de données constituent à l'heure actuelle des plates-formes pour répondre aux besoins des applications à grande échelle. Elles fournissent un ensemble de ressources variées, géographiquement distribuées dont le but est de réduire les coûts d'exécution des applications, d'assurer un accès rapide et efficace aux données, d'améliorer la disponibilité des ressources et la tolérance aux pannes. Dans de tels systèmes, ces objectifs ne peuvent être atteints que par l'utilisation de mécanismes de réplication. Dans ce contexte, un même objet peut être répliqué en autant de copies que nécessaire.

Dans ce mémoire, nous avons proposé une approche de création et de placement des répliques dans une grille de données de topologie basée sur un graphe qui a pour but de minimiser le coût d'accès aux données en les rapprochant des nœuds qu'ils les demandent au fur et à mesure des requêtes formulées tout en prenant en compte l'importance de ces répliques.

L'évaluation de cette stratégie sur le simulateur de grilles " OptorSim " a donné des résultats d'expérimentation très prometteurs au niveau :

- Des temps de réponse.
- La consommation des ressources réseaux.
- Le nombre de répliques créées
- L'espace de stockage occupé.
- La gestion des répliques au sein de la grille (entretien).

Ce travail permet d'ouvrir quelques perspectives intéressantes que nous récapitulons dans les points suivants :

- Considérer d'autres facteurs pour le calcul de l'importance des répliques.
- Prévoir un système de prédiction et de gestion de la défaillance car la grille est un environnement mobile et dynamique.
- Enrichir l'approche par un module de répartition de la charge entre nœuds " loadbalancing " pour diriger les requêtes vers les nœuds appropriés.
- Nous pensons à étendre l'approche aux requêtes d'écritures parprendre en charge la gestion de la cohérence des répliques.

Enfin, dans un autre travail, nous espérons toucher aux environnements de Cloud Computing.

ANNEXE

OptorSim est un simulateur de grille open source, développé en Java. Il a été développé dans le cadre du projet DataGrid. Il reproduit d'ailleurs l'architecture de ce dernier qui est présentée dans la Figure1.

La grille est composée de sites. Un site contient une unité de calcul appelé Computing Élément (CE), un gestionnaire de réplique Réplica Manager (RM) ainsi qu'une unité de stockage de données Storage Élément (SE). En dehors de ces sites se trouvent deux autres entités : l'Utilisateur, qui génère les requêtes et les soumet à l'ordonnanceur de la grille, le Resource Broker(RB).

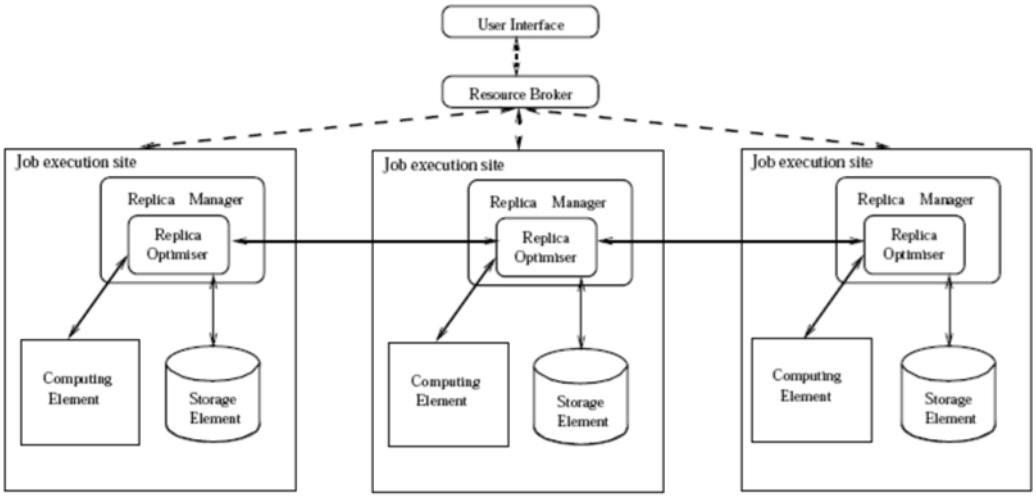

FIG. 3.24: Architecture d'OptorSim.

(a) **Computing Élément (CE) :** L'élément de calcul a pour rôle de simuler l'exécution des requêtes qui lui sont soumises. Il est défini par le nombre des nœuds et La durée d'exécution des requêtes qui est identique pour toutes les requêtes quelques soit leur type et qui est fixée dans les fichiers de configuration du simulateur.

(b) **Réplica Manager (RM) :** Il existe un RM par site et il a en charge la localisation et la gestion des mouvements des données. Le RM contient un Optimiser, il s'agit d'un objet dans lequel sont définies les stratégies de réplication et de suppression de données. Ce que les auteurs d'OptorSim appellent stratégies de réplication sont en fait la façon dont sont calculés les coûts de transfert d'une donnée, la façon dont est choisie la source d'un transfert ainsi que le choix des données à supprimer sur le SE lorsque cela est nécessaire.

(c) **Storage Element (SE) :** Un SE a la charge du stockage caractérisé par sa capacité de stockage. Il peut exister sans être associé à une CE, il peut être vu alors comme un dépôt de données distant. Le temps d'accès aux données situées sur le SE local par le CE est considéré comme négligeable.

(d) **Utilisateur :** Il est la source des requêtes. C'est lui qui crée les requêtes selon un schéma bien précis. Divers schémas sont fournis et le choix de celui qui sera utilisé est fait dans la configuration.

(e) **Resource Broker (RB) :** C'est l'organe central de décision. Il responsable de l'ordonnancement des requêtes soumises via l'interface.

BIBLIOGRAPHIE

[1] : A. Vernois, " Ordonnancement et réplication de données bioinformatiques dans un contexte de grille de calcul ", Thèse PHD, Octobre 2006.école normale supérieure de lyon France.

[2] : Zakia CHALLAL, " La réplication de données dans les grilles biologiques ",Mémoire MAGISTER,2010, Ecole Nationale Supérieure en Informatique Oued-Smar Alger.

[3] : Belayachi Naima, Behidji Rajaa, " Service de gestion de cohérence et d'équilibrage de charge dans les grilles de données ", mémoire d'ingénieur,2009, Université d'Oran.

[4] :kouidri Sihem," Gestion de la cohérence des répliques tolérante aux fautes dans une grille de données ",mémoire de magister,2010, Université d'Oran.

[5] : Haimoune Mohammed Elamin, SellaouiMohammed, Evaluation d'une stratégie de réplication adaptative pour les grilles basée sur un modèle de cout en utilisant OptorSim,Mémoire de master, Juin 2012, Université d'Oran.

[6] : G.Belalem, Y.Slimani, " A hybrid approach for consistency management in large scale system, Networking and services", International Journal of Applied Mathematics and Computer Sciences Volume 2 Number 3, 2006, Université d'Oran.

[7] : A. Imine, " Conception Formelle d'Algorithmes de Réplication Optimiste Vers l'Edition Collaborative dans les Réseaux Pair-à-Pair ", Thèse PHD,Novembre 2006, université Henri Poincaré Nancy 1, France.

[8] : I. Foster, C. Kesselman. " The Grid 2 : Blueprint for a New Computing InfraStructure ".Morgan Kaufmann Publishers Inc.,décembre 2003.

[9] :] G. Belalem. " Contribution à la gestion de la cohérence de répliques de fichiers dans les systèmes à large échelle ". Thèse PHD, Novembre 2007, Université d'Oran.

[10] : S. Bouamama, A. Benzerga, " Modèle économique pour la gestion de cohérence des répliques dans un environnement OptorSim ", Mémoire d'ingénieur d'état, Juin 2008, Université d'Oran.

[11] : Kella Abdelaziz, Ketrouci Djamal, " Une stratégie de réplication avec contrainte de stockage pour améliorer la disponibilité des données dans une grille de données ", Mémoire Master, 2011, Université d'Oran.

[12] : D. Dullmann, W. Hoschek, J. Jaen-Martinez, B.Segal,A Samar ,H. Stockinger and K. Stockinger. "Models for replica synchronisation and consistency in a data grid", 10th IEEE Symposium on High Performance and Distributed Computing(HPDC-10),2001.

[13] : The Globus Alliance. Globus toolkit : an open source software toolkit used for building grids ; http ://www.globus.org/to [Dernière consultation : 10 Mars 2016]

[14] : Abdennacer Khelaifa, "Une architecture hybride pour la réplication de données dans le SGBD PostgreSQL ", Mémoire de magistère,08/11/2010, Ecole nationale Supérieure en Informatique Oued-Smar Alger.

[15] :] Christelle Pierkot, " Gestion de la Mise à Jour de Données Géographiques Répliquées ", Thèse de doctorat, 02 Juillet 2008, Université de Toulouse III- Paul Sabatier.

[16] : A.Chien, "Grids and High Performance Distributed Computing", Note de cours, Mars 2004, University of Chicago, USA.

[17] : G. Oster. " Replication Optimiste et Coherence des Donnees dans les Environnements Collaboratifs Repartis ".Thèse PHD, Novembre 2005, Nancy-Universite France.

[18] :] F. Pedone, M. Wiesmann, A. Schiper, B. Kemme et G. Alonso," Understanding replication in databases and distributed systems", In Proceedings of 20th International Conference on Distributed Computing Systems, 2000.

[19] : Bsoul M., Al-Khasawneh A., Eddien Abdallah E., Kilani Y, " Enhanced Fast Spread Replication strategy for Data Grid", Journal of Network Computer Application 34(2), 2011,The Hashemite University Jordan.

[20] : Leila Azari, Mashalla Abbasi Dezfouli , "A New Architecture for Group Replication in Data Grid", nternational Journal of Computer Applications Technology and Research, 2013,Islamic Azad UniversityKhouzestan Iran.

[21] : Seong-il Hahm, Seongho Cho, Han Choi, Chong-kwon Kim et Pillwoo Lee, "The Design and Implementation of an Available Bandwidth Measurement Scheme in the Grid System", International Journal of Information Processing Systems Vol.2 No.2, June 2006.

[22] : Zeinab Fadaie , Amir Masoud ahmani,"A new Replica Placement Algorithm in Data Grid", International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, Mars 2012, Islamic Azad University,Tehran, Iran.

[23] : F. Magoulès, J. P. Kiat, et T. A. Kumar. " Introduction to grid computing". Taylor Francis Group, 2009.

[24] : B. Jacob, M. Brown, K. Fukui, et N. Trivedi. " Introduction to grid computing". IBM RedBooks, December 2005.

[25] : C. Fiévet. Legion " un système entièrement peer-to-peer ".
http ://www.01net.com/article/177530.html ?rub=3547, 2002.[Dernière consultation : 04 Janvier 2016].

[26] : H. Stockinger, Omer F. Rana, R. Moore et A. Merzky. " Data management for grid environments In High-Performance Computing and Networking",9th International Conference, HPCN Europe,June 2001, Amsterdam pays-bas.

[27] : A. J. Van Der Steen. "An evaluation of some beowulf clusters". Cluster Computing 6(4)springer publishers, 2003.

[28] : Y. Huang et N. Venkatasubramanian. "Data placement in intermittently available environments". In 9th International Conference of High Performance Computing (HiPC 2002), December 2002, Bangalore, India.

[29] : W. Hoschek, F. J. Janez, A. Samar, H. Stockinger et K. Stockinger." Data management in an international data grid project". In 1st IEEE/ACM International Workshop on Grid Computing (GRID 2000),December 2000, Bangalore, India.

[30] : E. S. Gunawan et W. Cai. " Performance analysis of a myrinet-based cluster". Cluster Computing, 6(4)Springer publishers, 2003.

[31] : K. Czajkowski, S. Fitzgerald, I. Foster et C. Kesselman. " Grid information services for distributed resource sharing". In 10th IEEE International Symposium on High Performance Distributed Computing,August 2001, San Francisco, California, CA USA.

[32] : A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. " The data grid : Towards an architecture for the distributed management and analysis of large scienti ?c datasets". Journal of Network and Computer Applications, 2001.

[33] : M. Bennani et D. A. Menasce." Resource allocation for autonomic data centers using analytic performance models". In 2nd IEEE International Conference on Autonomic Computing (ICAC'05), June 2005, Seattle, Washington, USA.

[34] : W. Allcock, A. Chervenak, I. Foster, C. Kesselman, and M. Livny. "Data grid tools : Enabling science on big distributed data". In SciDAC Conference,June 2005, San Francisco, CA, USA.

[35] : W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, et I. Foster. "The globus striped gridftp framework and server". In Proceedings of the ACM/IEEE SC 2005 Conference of SuperComputing (SC05), November 2005, Washington, USA.

[36] : http ://www.e-sciencecity.org/FR/gridcafe/architecture.html [Dernière consultation : 17 Avril 2016]

[37] : L. Cantone, "Le GridComputing et son utilisation dans les entreprises et les industries", mémoire de master, 2005, Université Claude Bernard Lyon 1, France.

[38] : I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, et J. V. Reich. "The Open Grid Services Architecture, Version 1.5",http ://www.gridforum.org/documents/GFD.80.pdf, 2006.[Dernière consultation :11 Janvier 2016].

[39] : Eric Robette ,"Etude et implémentations d'un logiciel Grid light " ; Mémoire de Licence informatique ,2007,Université de BRUXELLES, France.

[40] : Bruno DEL-FABBRO, " Contribution à la gestion des données dans les grilles de calcul à la demande : de la conception à la normalisation ",Thèse de doctorat,24 novembre 2004,l'UFR des sciences et techniques de l'université de franche-comté, France.

[41] : I.Foster and S.Tuecke "Globus : Bridging the gap",http ://www.mcs.anl.gov/ itf/talks.htm,2005,[Dernière consultation : 09 Janvier 2016].

[42] : Sashi, K. et Dr. Antony Selvadoss Thanamani, "Dynamic Replica Management for Data Grid",IACSIT International Journal of Engineering and Technology 2(4), 2010.

[43] : I.Foster, K. Ranganathan, "Identifying dynamic replication strategies for high performance data grids", in : Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing, in : Lecture Notes on Computer Science, 2002. Denver, USA.

[44] : I.Foster, "what is the grid ? A three point checklist", GridToday, 2002.

[45] : Anissa Abbad, Nabil Sid. " Une application peer-to-peer pour l'accroissement de la disponibilité des données". Mémoire d'ingénieur d'état, Ecole national Supérieur d'Informatique (ESI) Oued-Smar Alger.2009.

[46] : Gharbi Chahinez, Filali Fatima zohra, "Réplication des données dans un environnement Peer-to-Peer non structuré ",Mémoire Master ,2011,Université d'Oran.

[47] : Najme Mansouri , "An Effective Weighted Data Replication Strategy for Data Grid ", Australian Journal of Basic and Applied Sciences, 6(10), 2012.

[48] : Foster, I., K. Ranganathan,"Design and evaluation of dynamic replication strategies for high performance data grids", in : Proceedings of International Conference on Computing in High Energy and Nuclear-Physics, September 2001, Beijing, China.

[49] : Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko, Won-Sik Yoon, "Dynamic Data Replication Strategy

Based on Internet Hierarchy BHR", in : Lecture notes in Computer Science Publisher, vol. 3033, 2004, Heidelberg.

[50] : Yaser Nemati, Faramarz Samsami, Mehdi Nikhkhah"A Novel Data Replication Policy in Data Grid", Australian Journal of Basic and Applied Sciences, 6(7), 2012, Islamic Azad University, Fars, Iran.

[51] :] D.G. Cameron, R. Schiaffino, J. Ferguson, A.P. Millar, C. Nicholson, K. Stockinger, F. Zini, "OptorSim v2.1 Installation and User Guide", October 2006.