Chapitre IV: Les simulateurs de grilles.

Chapitre V:Amélioration proposée et simulation

RÉSUMÉ :

La grille est l'une des technologies la plus rapide émergentes au sein de l'environnement informatique de haute performance, elle fournitdes ressources distribuéespour traiter les demandesà grande échellequegénère ungrand volume d'ensemble de données. Dans un environnement de grille de données, la réplication des données est un moyen efficace pour améliorer l'accessibilité des données. Toutefois, en raison dustockage limité, une stratégie de remplacement de réplique est nécessaire pour faire de la stratégie de gestion dynamique de réplique efficace. Le placement de la répliqueest un facteur important pour améliorer la performance dans les systèmes de grille de données et un bon algorithme de placement de réplique peut entraîner des gains de bonne performance.

Dans cette mémoire nous proposons une amélioration de la stratégie PRA (PredictionReplica replacement Algorithme) qui se base sur la popularité de réplique pour prédire les besoins du site dans le futur. Il est implémenté par un simulateur de grille de données Optorsim, qui est développé par des projets européens de grille de données.

Mots-clés:grillede données, réplication, stratégie de remplacement,simulation, optorsim.

ABSTRACT:

The gridis one oftheemergingtechnologiesfaster within thehigh performancecomputing environment,itprovides resourcesfordistributedprocessing applicationstolarge scalegenerate a largevolumedata sets. In adata gridenvironment, data replicationis an effective wayto improveaccessibility of data.However, due tolimited storage;areplicareplacement strategyis needed tomake the strategyofdynamic management ofeffectivereplica.The placement of thereplicais an important factorto improve performanceindata gridsystems and a goodreplicaplacement algorithmcan result ingoodearningsperformance.

Inthis paperwe proposean improvementof the PRA (Replica ReplacementAlgorithmPrediction) strategy based onthe popularity ofreplicapredictedfortheneeds of the sitein thefuture. It is implementedby a simulatordata gridOptorsim, whichdevelopedby Europeandata gridprojects.

Keywords: data grid, replication,replacement strategy,simulation,optorsim.

TABLE DES FIGURES :

LISTE DES TABLES :

Introduction générale :

La gestion et le stockage Dans les applications de données intensives telles que la physique des hautes énergies (HEP), la génétique et l'observation de la Terre où la taille des fichiers de données a atteint le téraoctets ou parfois le *Péta-Bytes* sont très difficiles. Ces énormes quantités de données sont partagées entre les chercheurs du monde entier. Par conséquent la gestion de données et de ressources distribuées dans cette grande échelle est difficile. La grille de données est une solution pour ce problème.

Ces dernières décennies, les progrès réalisés dans le domaine des télécommunications ont rendus possible le regroupement d'une multitude d'ordinateurs, connectés entre eux par un réseau large-échelle. La naissance des grilles informatiques a permis la collaboration des ressources géographiquement distribuées, autorisant ainsi l'exécution d'applications qui nécessitent une grande puissance de calcul et un large espace de stockage.

La grille informatique est un outil dont le but est de regrouper des ressources partagées, distribuées et hétérogènes afin de réaliser des actions globales difficiles voire impossible à réaliser sur une machine unique. Assurer un accès efficace aux énormes quantités de données largement distribuées est un défi pour les réseaux de créateurs, les stratégies de réplications sont considérées comme le moyen le plus efficace pour relever ce défi, la réplication des données augmente la disponibilité et la fiabilité des bases de données. Elle offre aux utilisateurs de meilleures performances et une plus grande disponibilité des données.

Dans cette mémoire nous avons essayé d'améliorer la stratégie de réplication PRA et évaluer cette dernière comparativement à d'autres algorithmes par simulation sur OptorSim, qui est un simulateur des grilles informatiques destiné particulièrement à l'implémentation des algorithmes de réplication et l'évaluation des performances de la grille.

Ce mémoire est organisé comme suit. Nous commençons par présenter le contexte des grilles informatiques. Ensuite, nous étudions des généralités sur le principe de réplication de données. Le troisième chapitre est consacré à un état de l'art sur les stratégies de réplications. Dans Le quatrième chapitre nous citons quelques simulateurs de grilles et nous finissons par une discussion brièvement du simulateur OptorSim. Nous détaillons dans le cinquième chapitre notre amélioration proposée ainsi l'ensemble des tests effectués. Enfin, nous présentons les résultats de l'implémentation obtenus. Nous terminons le mémoire par une conclusion synthétique.

I.1Introduction :

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 Péta-octet. 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 tels 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. Un modèle en couche protocolaire a été décrit et des middlewares ont été développés.

Dans ce chapitre nous décrivons tous les notions générales des grilles informatiques.

I.2Définition de la Grille: [BBL00]

Le concept de grille informatique a été introduit par **I. Foster et Kesselman** comme une infrastructure massivement distribuée pour le calcul scientifique.

Le principe des clusters a donc été repris mais élargi. Les différents composants de la grille sont variés, on retrouve des clusters, des supers calculateurs ou même des stations de travail.

- Une grille permet :

Un partage coordonné de ressources dans un environnement flexible et sécurisé par une collection dynamique d'individus et d'institutions.

Plusieurs ordinateurs alimentent un réseau et fournissent ainsi une grande puissance de calcul et une grande capacité de stockage. Tout utilisateur peut alors utiliser les ressources de la grille via une interface unique sur son propre ordinateur. On retrouve ce terme sous des dénominations diverses : GridComputing, DataGrid, informatique « on demande », Utility Computing. Le Web est par exemple une application particulièrement réussie de ce concept de grille.

On peut à l'heure actuelle définir trois types de grilles informatiques :

-**Le virtual Supercomputing** : consistant à associer plusieurs supercalculateurs entre eux répartis géographiquement, offrant la vision d'un supercalculateur virtuel.

-**L' Internet Computing :** combinant un grand nombre de machines (plusieurs dizaines de milliers), regroupées entre elles par le réseau internet.

Ces machines sont mises à disposition de la grille lorsque leur taux d'utilisation devient faible. Cette architecture permet d'obtenir une puissance de calcul très importante mais également très variable.

Le Meta Computing consiste à associer plusieurs machines disposant d'applications différentes. L'ensemble est alors vu comme un ordinateur virtuel unique proposant un vaste choix d'applications et de services.

Ce système permet d'obtenir des services à la demande puisque tout est fournit par la grille : systèmes, applications et gestion des données.

Figure I.1 : Exemple d'une grille informatique

I.3 Propriétés d'une grille : [FK03]

Ces ressources sont potentiellement qualifiées de :

- **Partagées** : elles sont mises à la disposition des différents consommateurs de la grille et éventuellement pour différents usages applicatifs.
- **Distribuées** : elles sont situées dans des lieux géographiques différents.
- **Hétérogènes :** elles sont toutes de nature différente par exemple par le système d'exploitation ou le système de gestion des fichiers.
- **Coordonnées** : les ressources sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements). Ces dispositions sont souvent assurées par un ou plusieurs agents, qu'ils soient centralisés ou répartis.
- **Non-contrôlées** (ou **autonomes**) : les ressources ne sont pas contrôlées par une unité commune. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle.

- **Délocalisées** : les ressources peuvent appartenir à plusieurs sites, organisations, réseaux et se situer à différents endroits géographiques.

I.4 Les services grille : [FK03]

I.4.1 Service d'information grille :

La grille est un environnement dynamique où la disponibilité et le type des ressources varient constamment. Il est donc 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).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 nœud sont chargés de récolter les diverses informations.

I.4.2 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.

I.4.3 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. Dans certains systèmes il est plus avantageux de regrouper les deux types de données (par exemple dans les bases de données distribuées).

Au contraire, dans d'autres systèmes, il peut être plus performant de séparer ces deux types de données, quitte à les stocker sur des ressources différentes.

I.4.4 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.

I.4.5 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 : c'est une norme définissant une bibliothèque de fonctions de communication par passage de messages, permettant d'exploiter des machines parallèles à mémoire partagée, ainsi que des grappes d'ordinateurs hétérogènes à mémoire distribuée.), 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 Posix (Portable Operating System Interface : Le X désigne l'héritage Unix de l'API. Ce standard défini par exemple les services d'entrées/sorties de base tel que fichiers, terminaux, réseau. 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...).

I.5 Taxonomie :

Selon les besoins et les objectifs visés par les applications, les grilles supportant ces dernières peuvent être classées en différentes catégories. Dans, on distingue principalement trois types de grilles : Grille de calcul, grille de données et grille de service.

I.5.1 Grille de calcul

Ce sont des grilles caractérisées par un calcul intensif où les ressources les plus importantes sont les unités de calcul. Les applications sont exécutées sur ces différentes unités de manière

parallèle quand cela est possible. Ceci maximise l'exploitation des ressources CPU et réduit le temps d'exécution total des applications. Les applications temps réel sont un bon exemple d'application qui tire profit d'une telle architecture. [Cha10]

I.5.2Grille de données

Ce sont des grilles caractérisées par des données volumineuses. Les ressources les plus importantes sont les unités de stockage. Les applications de DATAMINING et de DATA WAREHOUSE, par exemple, sont implémentées sur ce type de grille. Des services de gestion et d'accès aux données sont offerts. [Cha10]

I.5.3 Grille de services

Certaines applications offrent des services qui ne peuvent pas s'exécuter sur une seule machine et nécessitent une collaboration à large échelle telle que les applications de travail collaboratif à distance. Pour ce genre d'application, les grilles de services offrent une infrastructure virtuelle pour communiquer et interagir en temps réel et peuvent offrir des services de manière dynamique à la demande de l'utilisateur. [Cha10]

I.6 Architecture

L'architecture détermine les principaux composants d'une grille, les fonctionnalités de chaque composant et les interactions entre eux.

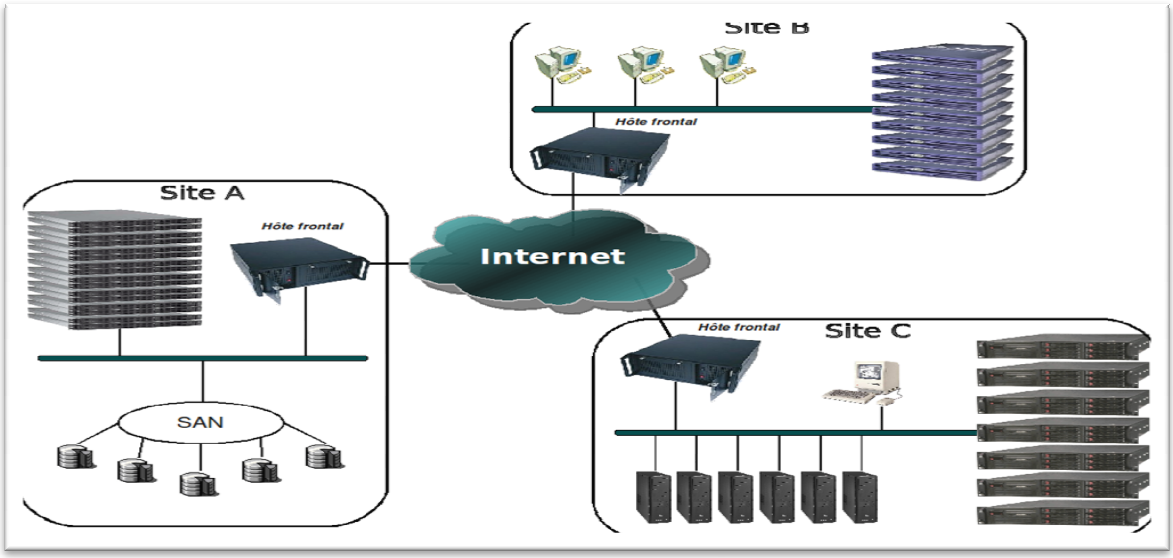


Figure I-2 : Architecture générale d'une grille informatique

La figure I.2 montre l'architecture typique d'une grille informatique. Les trois sites sont reliés entre eux, par un réseau étendu de type WAN. Nous retrouvons la machine frontale qui filtre toutes les communications. Cette dernière est dotée d'un pare-feu, non représenté sur le schéma. A l'intérieur de chaque site, l'hétérogénéité est bien présente : des baies de stockage reliées par un SAN (Storage Area Network), des grappes, ainsi que de simples machines de bureau.

Storage Area Network : Un SAN est un système de stockage relié par un réseau performant (ex. fibre), directement accessible en mode bloc par le système de fichiers des serveurs. Chaque serveur voit l'espace offert par une baie SAN comme son propre disque dur.[Ort09]

I.6.1 Modèle en couche :

Ian Foster et al. Définissent une grille selon le modèle en couches suivant :

L'architecture en couches protocolaires permet d'assurer l'interopérabilité entre systèmes [FK03].

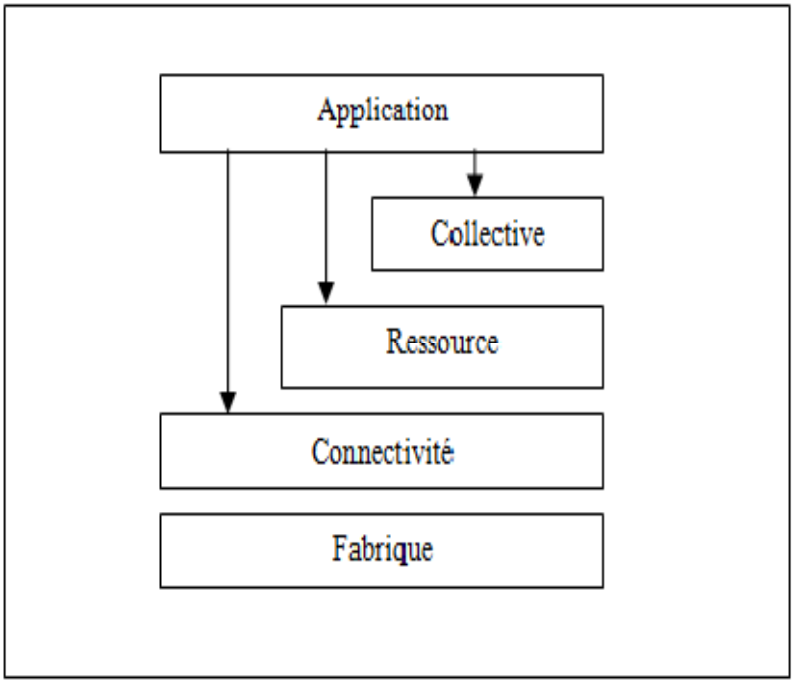


Figure I.3 : Architecture en couches d'une grille [FK03]

Couche Fabrique

Elle fournit des mécanismes pour la gestion des ressources de la grille tels que les mécanismes de démarrage, de réglage et de contrôle d'exécution des programmes, des mécanismes de lecture, écriture, transfert et sélection de données et enfin des mécanismes de contrôle du réseau.

Couche Connectivité

Cette couche définit les protocoles de communication (IP, ICMP, TCP, UDP, etc.) et d'authentification (X.509, SSL).

Couche Ressource

Elle s'occupe du contrôle de chaque ressource partagée séparément avec une éventuelle facturation correspondant à son utilisation.

Couche Collective

Cette couche s'occupe de la coordination entre ressources partagées. Elle implémente les services suivants : service d'annuaire, service d'allocation, service de maintenance et le service de réplication de données.

Couche Application

Elle comprend les applications implémentées sur la grille comme la bio informatique, l'astronomie et dernièrement le e-commerce, le travail collaboratif, etc.

I.7 Domaine Application des grilles :

Il existe des centaines de grilles informatiques dans le monde. Beaucoup d'entre elles sont utilisées pour la science en ligne, rendant ainsi possibles des projets nécessitant une puissance de calcul très importante**.**

- Les **biologistes** utilisent les grilles pour simuler depuis leur ordinateur des milliers de molécules afin de trouver de nouveaux traitements capables de bloquer les protéines responsables de certaines maladies.

- Les **géologues** utilisent les grilles informatiques pour traquer les niveaux d'ozone grâce aux données fournies par les satellites. Ils téléchargent ainsi plusieurs centaines de Giga octets de données chaque jour (l'équivalent d'environ 150 CD par jour !).

- Les **physiciens des hautes énergies** utilisent les grilles pour mieux comprendre l'univers. Ces grilles sont constituées de dizaines de milliers d'ordinateurs de bureau et permettent de stocker et d'analyser les 10 Pétaoctets de données (l'équivalent de 20 millions de CD) produites chaque année par le Grand collisionneur d'hadrons (LHC). Des milliers de physiciens de nombreuses universités du monde entier veulent analyser ces données.

- Les **ingénieurs** utilisent les grilles pour étudier des carburants alternatifs comme l'énergie de fusion.

- Les **artistes** utilisent les grilles pour créer des films d'animation toujours plus complexes (Kung Fu Panda par exemple).

- Les **chercheurs en sciences sociales** utilisent les grilles pour étudier le comportement des abeilles, les caractéristiques de notre société, les secrets de l'histoire. [Gri]

I.8 Conclusion :

Les grilles informatiques sont une technologie en plein essor, ils sont utilisés dans des domaines scientifiques tels que la physique. La stabilité d'une telle grille permet aux scientifiques, d'exécuter leurs applications dans un contexte bien défini, et sur lequel ils gardent un certain contrôle. Dans Ce chapitre on a présenté les notions des grilles, l'architecture générale et les Domaines d'application.

II.1Introduction:

Le terme « **réplication** » a été emprunté à la biologie. On aurait pu dire simplement « duplication ». Dans le jargon informatique en général, et dans celui des bases de données en particulier, ce terme désigne un mécanisme de copie automatique d'une base de données vers une autre, permettant de rapprocher des données de l'utilisateur, dans un système distribué.

La réplication utilise la technologie de base de données distribuée pour partager les données entre multiples sites mais une base de données répliquée et une base de données distribuée sont différentes. Pour les bases de données distribuées, les données sont disponibles à plusieurs endroits mais une table particulière réside à un seul site. La réplication signifie que les mêmes données sont disponibles à multiples endroits.

La réplication des données est reconnueaujourd'hui comme moyens efficaces pour augmenter la disponibilité et la fiabilitédes bases de données. De plus la réplication peut contribuer favorablement àl'amélioration des performances en utilisant les copies locales voire les copiesplus proches.

II.2Définition :

En informatique, la réplication est un processus de partage d'informations pour assurer la cohérence des données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, la tolérance aux pannes ou l'accessibilité.

On parle de réplication de données si les mêmes données sont dupliquées sur plusieurs périphériques. Une application de base de données repose sur un modèle client-serveur. Dans ce modèle, le client se connecte au SGBD pour passer des ordres. Ces ordres sont de deux natures : **lecture** (on parle alors de requêtes) ou **mise à jour** (on parle alors de transactions). Pour les transactions, il y a une modification des données sur le serveur, mais cela reste des ordres de courte durée.

A l'inverse, dans le cas d'une lecture, il n'y a pas de modification des données mais les traitements peuvent être longs et porter sur une grande masse de données.

II.3Principe de la réplication :

L'objectifprincipal de la réplication est de faciliter l'accès aux données en augmentant la disponibilité. Soit parce que les données sont copiées sur différents sites permettant de répartir les requêtes, soit parce qu'un site peut prendre la relève lorsque le serveur principal s'écroule. Une autre application tout aussi importante est la synchronisation des systèmes embarqués non connectés en permanence.

Le principe de la réplication, qui met en jeu au minimum deux SGBD, est assez simple et se déroule en trois temps :

· La base « maître » reçoit un ordre de mise à jour (INSERT, UPDATE ou DELETE).

· Les modifications faites sur les données sont détectées et stockées (dans une table, un fichier, une queue) en vue de leur propagation.

· Un processus de réplication prend en charge la propagation des modifications à faire sur une seconde base dite esclave. Il peut bien entendu y avoir plus d'une base esclave.

II.4Avantages et inconvénients de la réplication : [Bil09]

La réplication présente des avantages différents selon le type de réplication et les options choisies, mais l'intérêt général de la réplication est la disponibilité des données à tout moment et en tout lieu.

Les avantages sont les suivants :

- ✓ Possibilité pour plusieurs sites de conserver des copies des mêmes données.
- ✓ Autonomie accrue : Les utilisateurs peuvent manipuler des copies de données hors connexion, puis propager leurs modifications aux autres bases de données lorsqu'ils sont connectés.
- ✓ Davantage de méthodes pour accéder aux données, comme l'exploration de données à l'aide d'applications Web par exemple.
- ✓ Amélioration des performances de lecture des agrégats.
- ✓ Rapprochement des données par rapport aux utilisateurs individuels ou aux groupes. Cela permet de réduire les conflits liés aux modifications de données et requêtes impliquant plusieurs utilisateurs.

- ✓ Utilisation de la réplication dans le cadre d'une stratégie de serveur en attente personnalisée. La réplication est l'une des options de la stratégie de serveur en attente.

Plusieurs aléas sont à quand même prévoir avec un système de réplication :

- ✓ Consommation élevée des ressources du serveur suivant la position du distributeur.
- ✓ Les modifications de schéma peuvent ne pas être prises en compte suivant le mode de réplication.
- ✓ Les problèmes de réplication suivants affectent la performance de vos réseaux :
  - · Volume et taille usuelle des données circulant sur le réseau.
  - · Nombre d'abonnés à un éditeur particulier.
  - · Vitesse de la connexion.
  - · Capacité de traitement de l'éditeur, du distributeur et des abonnés.

II.5Types de réplication :

Selon quand, qui et comment effectuer les calculs et les mises à jour des répliques, On peut distinguer plusieurs types de réplications qu'on peut en citer quelques-unes :

a- Synchrone/Asynchrone:[DHJSSSS01]

- **Synchrone :** Dans la réplication synchrone, dite aussi « réplication en temps réel», lorsqu'une mise à jour est soumise à un réplica donné, elle est propagée à tous les autres au sein d'une transaction atomique avant de valider la requête initiale et de retourner la réponse à l'utilisateur. Cette approche assure la cohérence mutuelle et son implémentation est réalisée par une phase de verrouillage des données et une autre de validation.
- **Asynchrone :** est aussi appelée « stocker et propager », privilégie l'efficience sur la cohérence et peut ne pas garantir une forte cohérence mais par contre est facilement déployée puisqu'elle évite ou affaiblit les contraintes de la réplication synchrone. Lorsqu'une mise à jour est soumise à un réplica donné, elle est validée localement puis elle est propagée aux autres dans une transaction séparée .Cette approche peut passer à l'échelle et elle est beaucoup plus performante en terme de réduction de messages échangés. Elle ne bloque pas en cas de pannes et offre en plus une flexibilité au système qui peut choisir le moment convenable pour propager les mises à jour.

b- Symétrique/Asymétrique:[PWSKA00]

- **Symétrique** : Dans cette technique Laréplication tous les sites sont autorisés à effectuer des modifications et à les diffuser à toutes les répliques. Ellepermet d'organiserlapropagationdesmisesàjourdescopiesàtoutinstant.

  La réplication symétrique poseleproblèmedeconcurrenced'accèsrisquantdefairedivergerles copies.Unetechniqueglobalede résolutiondeconflitsdoitdoncêtremiseenœuvre.
- **Asymétrique :** les mises à jour sont centralisées. Ainsi, un seul site (appelé site primaire) a le droit d'effectuer des modifications et d'envoyer ensuite le résultat aux autresrépliques. En cas de panne du site primaire, il doit être remplacé par un autre site avec application des configurations nécessaires, ce qui parfois ne passe pas de manière inaperçue par rapport à la propagation des mises à jour (retard).

c- Totale/Partielle [PWSKA00]

- **La réplication totale** : consiste à répliquer tous les objets partagés et à les stocker chez tous les sites de la grille. L'équilibrage de charge est simple puisque tous les sites ont les mêmes capacités et la disponibilité est maximale puisque tout site peut remplacer un autre en cas de panne.
- **La réplication partielle** : consiste à répliquer pareillement les objets partagés : les sites peuvent ne pas stocker les mêmes réplicas. Son utilisation dans le cas de trafic volumineux d'information peut optimiser les échanges et réduire la consommation bande passant et d'espace de stockage. Les mises à jour seront propagées uniquement aux sites affectés (stockant une copie primaire ou secondaire de l'objet).Le réseau ainsi que les sites seront moins chargés. Cependant, certaines données peuvent être liées ou se référence ce qui peut créer des complications au niveau.

d- Réplication active, passive, semi-active :

- **Active :**La réplication active se définit par la symétrie des comportements des copies d'un composant répliqué. Chaque copie joue un rôle identique à celui des autres.[PWS99]

  Principe :

– réception des requêtes : toutes les copies reçoivent la même séquence de requêtes qu'est un ensemble totalement ordonné.(Les copies reçoivent les mêmes requêtes dans le même ordre).

– traitement des requêtes : toutes les copies traitent les requêtes de manière déterministe.

– émission des réponses : toutes les copies émettent la même séquence de réponses.

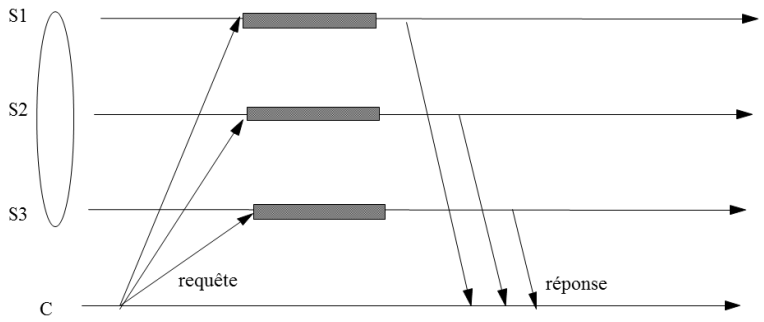


Figure II.1: Principe de la réplication active[Sen03]

Ce principe est illustré par la figure II.1. Les flèches horizontales représentent l'exécution de quatre composants : Client(c), S1, S2, S3. Les Si sont les copies du composant répliqué S. Elles appartiennent à un même groupe. Lorsque le client invoque S, il envoie une requête à tous les Si à l'aide d'un ABCAST assurant l'ordre total et la propriété d'atomicité. Chaque Si traite la requête et renvoie une réponse au client.

- **Semi-active :**

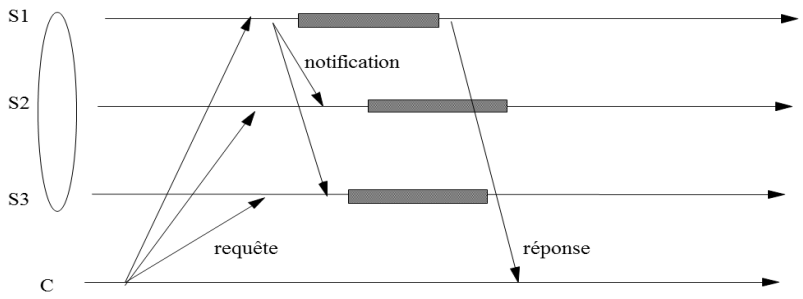
  La réplication semi-active (semi-active réplication) se situe à mi-cheminentre la réplication active et la réplication passive. La copie primaireest appelée leader et les copies secondaires sont appelées suiveurs.[PWS99]

  Principe :

  – réception des requêtes : toutes les copies reçoivent le même ensemblede requêtes (Les requêtes ne sont pas totalement ordonnées).

  – traitement des requêtes : toutes les copies traitent toutes les requêtes. La copieprimaire traite une requête dès qu'elle la reçoit. Par contre, une copiesecondaire doit attendre une notification de la copie primaire pour pouvoir traiter une requête.

  – émission des réponses : la copie primaire est la seule à émettre les réponses.



FigureII.2 : Principe de la réplication semi-active[Sen03]

La Figure II.2illustre ce principe. Le client c envoie une requête à tous les Si. La copie primaire S1 envoie une notification notifiée aux copies secondaires et commencent le traitement de la requête. Les copies secondaires S2 et S3 ne commencent à traiter la requête qu'après avoir reçu la notification de la copie primaire. Sitôt le traitement terminé, S1 envoie la réponse au client.

- **Passive :**

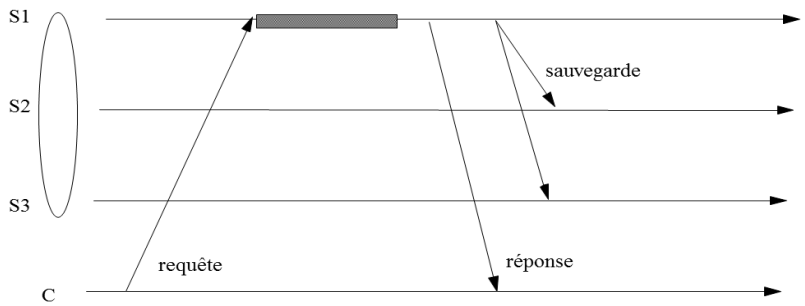
  La réplication passive distingue deux comportements d'un composant répliqué: la copie primaire(Primary Copy) et les copies secondaires(backups). La copie primaire est la seule à effectuer tous les traitements. Les copies secondaires, passivent, surveillent la copie primaire. En cas de défaillance de la copie primaire, une copie secondaire devient la nouvelle copie primaire.[PWS99]

  Principe :

  – 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.

  – émission des réponses : la copie primaire est la seule à émettre les réponses.



FigureII.3 : Principe de la réplication passive [Sen03]

Le client c envoie la requête uniquement à la copie primaire S1. Celle-ci traite la requête, construit un point de reprise et l'envoie à l'aide d'un multicast fiable assurant l'ordre FIFO, aux copies secondaires S2 et S3 en précisant le changement de son état (message update). Après la mise à jour de leurs états, les copies secondaires envoient un ACK à la copie primaire. La copie primaire envoie la réponse au client après réception des "ACK" de toutes les copies secondaires. Le point de reprise permet de synchroniser l'état des copies secondaires avec celui de la copie primaire puisque celle-ci est la seule qui communique avec le reste du système.

II.6La réplication dans les grilles :

La réplication est aujourd'hui largement utilisée dans les grilles. Elle consiste à créer plusieurs copies d'un même fichier sur des ressources de stockages différentes de la grille.

Cette technique permet d'augmenter la disponibilité des données, la charge étant alorsrépartie sur les différents nœuds possédants une réplique. Le service de réplication met en œuvre une politique de cohérence particulière. Il doit également maintenir un catalogue desrépliques, décidé quand créer et supprimer une copie. Il s'appuie en général sur le systèmede monitoring des ressources du GIS, pour prendre ces décisions.

Dans le cas contraire, il est moins avantageux de maintenir la cohérence entre les deuxcopies, plutôt que de n'avoir qu'une seule copie. Toutefois, la réplication sert également àéviter de perdre des données sensibles lorsque les nœuds tombent en panne. Il faut donctrouver le meilleur compromis en fonction du type de données et des applications.

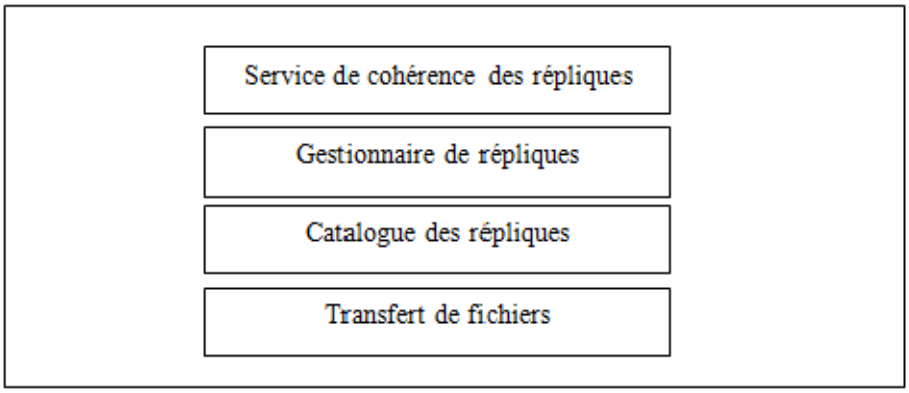
II.6.1Méthodes de réplication :

Il 'y a deux méthodes de réplication statiques et dynamiques.

i. **Réplication statique :**après qu'une réplique soit créée, elle existera dans le même lieu jusqu'à ce qu'elle soit effacée manuellement par des utilisateurs ou sa durée est expirée.
ii. **Réplication dynamique :** elle prend en compte les modifications des environnements de grille et crée automatiquement de nouvelles réplications pour les fichiers de données populaires ou déplacer les réplications vers d'autres sites si nécessaires pour améliorer la performance.

II.6.2Architecture du service de réplication :

Nous avons vu dans l'architecture d'une grille que la couche Collective offre un service de réplication. La figure**II.4** illustre l'architecture modulaire de ce service.Cette architecture est composée de 4 couches définies comme suit[Gri]:



FigureII.4: Architecture du service de réplication

- **Transfert de fichiers :**
  Le transfert de fichier entre les nœuds de la grille se fait au moyen du protocole GridFTP, 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.

II.6.3Les Topologiesdesrépliques :

Latopologie de réplicationdans les grilles, décrit les connexionslogiques quiutilisentla grille pour répliquer des fichiersentreles nœuds.Onpeutdistinguertrois topologieslesplus utilisées dans environnementsdegrillesdedonnées :anneau, hiérarchique et étoile.[PWCEKRT81]

- ✓ **La topologie en anneau :**Les fichiers se répliquent d'un nœud à un autre dans une configuration circulaire, avec chaque nœud relié aux nœuds de chaque côté de celui-ci dans le cycle. Ce modèle est utilisé dans une approche à la réplication paire, en présence de plusieurs répliques maîtresses.
- ✓ **La topologie hiérarchique :**(gros arbre) exploite lieu géographique et une haute disponibilité de la bande passante et offre une extension progressive du réseau de la topologie globale de distribution de réplique.Danscemodèle,lesrépliquessontplacéesàdifférentsniveaux,etcommuniquentlesunsavecles autres.
- ✓ **La topologie en étoile :**Une organisation des sites en étoile implique la centralisation de la communication autour d'un site unique par lequel doit passer toute mise à jour avant d'être propagée. Un site travaille en déconnecter sur les fichiers de son cache et se reconnecte à un groupe de serveurs disponibles pour propager ses modifications.

II.7Conclusion :

Nous avons présenté dans ce chapitre la technique de réplication, son principe, ses différents types, Architecture du son serviceet ses utilisations dans le contexte des grilles. La réplication est un mécanisme important parce qu'il permet à des organisations d'assurer à leurs utilisateurs un accès aux données courantes où et quand ils en ont besoin. Elle consiste à créer des copies identiques d'ensemble de données sur des sites géographiquement distants.Concerne les environnements de type grille, les modèles de réplication cherchent principalement à améliorer le temps d'accès aux ressources,assurer une continuité de fonctionnement du système en dépit des pannes et garantir la cohérence de leur contenu.Parmi les problèmes posés par l'utilisation de la réplication dans les grilles, nous mettrons l'accent sur le problème du placement des répliques, l'objet du chapitre suivant c'est l'une solution de ce problème qui est la bonne stratégie de réplique.

III.1 Introduction :

Quand les répliques devraient être créées ou supprimées? Où les placers ? Quels dossiers devraient être répliqués ? Les réponses à ces trois questions fondamentales nous portent à différentes stratégies de réplication.

La stratégie de placement des répliques peuvent amener un facteur important des performances et un maximum de gain d'un système de gestion de répliques. Plusieurs modèles de stratégies de placement de répliques ont été proposés dans la littérature, dans ce chapitre nous allons évoquercertains d'entre eux.

III.2 Différents axes de la recherche : [Cha10]

- **Placement des répliques**

  La stratégie 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 ? Où la placer? La stratégie doit permettre à chaque nœud de la grille de trouver les données dont il a besoin dans son voisinage proche.

- **Cohérence des répliques**

  La redondance de données par réplication nécessite la maintenance de la cohérence de leur contenu. Ainsi, un traitement supplémentaire est nécessaire lors des mises à jour. Chaque modification doit être propagée à toutes les répliques d'une même donnéepour assurer leur cohérence, ce qui aura pour effet d'engendrer un surcoût de traitement.

- **Nombre de répliques**

  La réplication nécessite une multiplication des ressources de stockage et de calcul,engendrant des coûts supplémentaires qu'il n'est pas toujours possible de prendre en charge. Pour cela, le nombre de répliques d'une donnée doit être limité et convenablement choisi.

III.3État de l'art des stratégies de placement de la réplication dans les grilles de données

Stratégie1:*Identifier lesstratégies de réplication dynamique pourunehaute performance*:[LSS02].

KavithaRanganathanetIan Foster ont implémenté et évalué six stratégies différentes :

Stratégie 1:Pas deréplicationou Caching: Le cas de basecontre laquellenous comparonsles différentes stratégiesestquandaucune réplicationn'a lieu. L'ensemble de donnéescomplet est disponibleà la racine dela hiérarchielors dela simulationcommence.

Stratégie2 : Meilleur client : Chaque nœud enregistre les demandes de chaque fichier et crée une répliquede ce fichierau meilleurclient qui a le taux de demande plus élevé.

Stratégie3 : Les données de cette stratégie dans flux d'une manière similaire. Une fois le seuil d'un fichier est dépassée à la racine, d'une réplique est créé à l'étape suivante, mais sur la voie du meilleurclient. Ainsi le nouveau site de la réplique est un ancêtre du meilleur client. Une fois que le seuil pour le fichier est dépassé au niveau 2, il est ensuite répliqué au niveau immédiatement inférieur et ainsi de suite. Un fichier populaire peut finalement être répliqué chez le client lui-même.

Stratégie4 : Mise en cache Plain: Leclient qui demandeun fichierstockeune copie locale. Comme cesfichierssont de grande taille(2 Go) etun clienta suffisamment d'espacepour stocker un seul fichierà la fois, les fichiersseremplacentrapidement.

Stratégie5 : La mise en cacheainsila réplication en cascadecombine ces deuxstratégies, permettant au client destockerlocalement etpériodiquementlefichier en identifiant des fichiersà usage fréquentpour se propagervers le bas. Propagationrapidestocke une copiede chaquefichier tel qu'ilse propagevers le basà travers lesnœuds.

Strategie6 : Diffusionrapide: danscette méthode, uneréplique dufichier est stockéà chaque nœudlong de sonchemin d'accès auclient C'est, quand un clientdemande un fichier, unecopie est stockéeà chaque niveausur le chemin. Cela conduit à unepropagationplus rapide des données.

Ils ont trouvé que la mise en cache ainsi que la réplication en cascade ont montré la meilleure performance. Cache ce que vous pouvez, déplacez ce que vous ne pouvez pas près des grands utilisateurs.

Stratégie 2 : ***Stratégie de réplication dynamique basée sur la hiérarchie de la bande passante*** **:** [MKKY03]

Cette stratégie de réplication dynamique appelée BHR est basée sur la hiérarchie de la bande passante d'Internet et le niveau de localité du réseau. Une région de réseau est un espace topologique de réseau où les sites sont situés les uns prés d'autres. Cette région de réseau peut être considérée comme une topologie de l'Internet dans un pays. Il est généralement connu que la bande passante inférieure peut être allouée pour la liaison réseau entre les sites à travers un pays. Le client peut accéder facilement à la réplique s'il estdans la même régioncar une bande passanteplus largepeutêtre fournie danscette région. En revanche, sila répliquedemandéese trouvesur un sited'une autre région, beaucoup de temps sera consomméàchercher cette répliqueparde nombreuxliens, une forme delocalité émergece que nous appelons le niveau de localité du réseau.

Le but principalde la stratégieBHR est de maximiser le niveau de localité du réseau dans la grillede données etelle tente derépliquer les fichiersqui sontsusceptibles d'être utilisésfréquemmentdans la région dans le futurproche.BHRoptimiseurfonctionne aussi bien surune régionetun sitequi peuvent se coopérer les uns avec les autres.

Dans la grille de données, il peut y avoir de popularité d'accès aux fichiers, c'est-à-dire certains fichiers seront demandés plus fréquemment que d'autres par l'emploi de la grille. Le BHR seconcentre sur la popularitéd'accès au niveau de la région et remplace les fichiers impopulaires du point de vue régional.

Les résultats de la simulation montrent que BHR prend moins de temps d'exécution que d'autres stratégies surtout lorsque les sites de réseaux ont la taille de stockage relativement petite et la hiérarchie des bandes passantes apparaît clairement.

Stratégie 3 : ***Stratégiede réplication de donnéespour augmenter ladisponibilité des données :***

Les auteurs dans ce travail[LV04] ont proposé une nouvelle stratégie qui minimise ledébit de donnéesmanquées,cette stratégieprend les décisionsderéplicationet de placementsur la base desavantages reçusà partirde laréplicationdufichier àlong terme. Sile fichier demandén'est passur le site, il sera répliqués'il yasuffisamment d'espacede stockage. S'il n'y a passuffisamment d'espace librepour stockerla réplique, un fichier existantdoit être remplacéetla stratégiedoit évaluer lesavantages destockerle fichier avecle coût de lasuppression d'un autre fichier.

La stratégie utilise la valeur Vi qui est calculée de la manière suivante : Ils ont supposé qu'il ya un fi fichier dans la mémoire à l'instant t, et associant un tel fichier avec une valeur Vi, qui sera mis au nombre de fois où ce fichier sera accessible à l'avenir. Pour certain modèle d'accès, basé sur l'histoire de l'accès aux fichiers, il est possible de faire une prédiction de la Vi.

Un poids est ensuite calculée sur la base de : 1) Vi, 2) l'existence d'un fichier, 3) la taille du fichier et 4) le nombre de copies du fichier. Nous notons que pour les données chaudes, Vi sera plus élevé que pour les données froides. Comme le calcul de poids considère Vi ainsi,le poids du fichier sera grandement affecté par la popularité du fichier. Les fichiers avec le plus petit poids sont pris en compte pour la suppression première. Les fichiers sont supprimés et remplacés par la nouvelle réplique, seulement si la valeur de la réplique est supérieure à la valeur de la perte résultante de la suppression des fichiers existants.

La stratégie passe en trois étapes pour effectuer la décision de la réplication. Tout d'abord, si le fichier demandé existe dans le SE, il ne sera pas reproduit de nouveau. Deuxièmement, les candidats potentiels seront choisis en fonction de leur poids. Cela est utile pour l'étude à long terme, car un fichier d'une valeur minimale sera remplacé par un autre plus précieux. En troisième lieu, la stratégie garantit le gain répliqué sera plus grand que la perte de remplacement.

Ils ont évaluéleur stratégieet ils démontrentque cette nouvelle stratégiesurpasseratoutes les autres dansla plupart des cas.

Stratégie 4 : ***Algorithme de placement a priori des répliques basée sur les distances.*****[Cha10]**

Dans ce travail, Les auteurs (Ko et al) proposent un protocole de réplication pour les réseaux à large échelle. Le réseau est modélisé par un graphe avec des nœuds colorés. Les nœuds du graphe désignent les sites de la grille, les arrêtes désignent les interconnexions entre sites, et la couleur d'un nœud désigne la donnée quecontient un site. A la création des répliques, les nœuds contenants des répliques identiques auront la même couleur.

Le principe du protocole de Ko et al est que chaque nœud du graphe choisit une couleur de manière à minimiser la distance entre nœuds de couleurs différentes et à maximiser la distance entre nœuds de même couleur.

Lesobjectives de cette approche sont :

- vise à rapprocher les ressources de n'importe quel nœud du réseau, afin d'améliorer les temps d'accès.

- vise à éloigner les copies identiques. Ceci évite, d'unepart, les conflits lors des transmissions simultanées, et minimise le nombre de répliques, d'autre part.

Parmi les avantages que présentent le protocole de Ko et al on a :

– complètement distribué : chaque nœud exécute des tâches pour décider du placementdes répliques.
– auto configurable : Aucune partie tierce ne coordonne les nœuds.
– asynchrone : aucune horloge n'est nécessaire pour coordonner les opérations, ce qui est très important dans un système à large échelle où les horloges des sites ne peuvent pas être synchronisées.

Strategies5: *Constrained Fast Spread (CFS)* [HKIL12]

Le CFS vise à atténuer lesprincipaux problèmes rencontrésdans les stratégiesde la réplicationactuellescommela négligence dela capacité de stockagedesnœuds. La nouvelle stratégieCFSa renforcé lastratégie depropagationrapideen se concentrant surla faisabilitéde reproduirela répliquedemandéesur chaque nœudau sein du réseau.

La principaleconcentrationde CFS estla prise en comptedesdeuxcritères, la taille de larépliquedemandée (RR) etle nombre de demandesde laRR. Unratio est calculépour produire lenombre de demandespartiel(PNOR). PNORest le rapport entrele nombre de demandes(NOR) à la pénuriede l'espacerequispour accueillir leRR.

EnCFS une répliqueest vérifiée pourêtre stockéà chaque nœudqui est duclient. Cela conduit à unepropagationplus rapidedes données.

L'algorithme de CFS fonctionne comme suit :

Si leprocessus de vérification del'existence de laRRàla RNconstate que leRRexiste dans ce casla répliquesera utiliséesans vérification, Sinon, le processus vérifiesi l'espacede stockage libredans le nœudest suffisantpour recevoir laRR.

Si les cas précédents ne sont réalisablesle processus fait les sommationsdestailles dequelques-unes desrépliqueset les compare àla taille de larépliquedemandée.Dans le cas oùl'espace allouéest suffisant pourêtre attribué parla répliquedemandée;toutes les répliquesseront supprimées

et les remplace par La répliquedemandéesile nombre de demande de ces répliques est inférieur à le PNOR, Sinon on ne les remplace pas.

Dans la simulation les auteurs ont comparé entre le CFS et la stratégie de **:** La mise en cacheet la réplication en cascade et la diffusion rapide. Ils ont trouvé que dela stratégie de mise en cache et la réplication en cascadeont bien fonctionné.D'une part lesmotifsde demandesontaléatoires et la stratégie de diffusion rapidefonctionnebien quand on a une petite quantitéde la localité del'utilisationdu fichier.D'autre part le CFS fonctionne mieux quand on aun motif de localités géographiques.

Stratégie6:PDDRA **« *Une nouvelle pré-lecture algorithme de réplication de données dynamique basée dans les réseaux de données* »** **[SM12]** :

Les auteursproposent un nouvel algorithme de réplication de données dynamique nommé PDDRA qui optimise les algorithmes traditionnels.
Cet algorithme proposé est basé sur une hypothèse: les membres d'un VO (Virtual Organisation) ont des intérêts similaires pour les fichiers. Sur la base de cette hypothèse et l'historique d'accès aux fichiers PDDRA prévoit les besoins futurs des sites de réseau et pré-récupère une séquence de fichiers sur le site de la grille du demandeur, donc la prochaine fois quand ce site a besoin d'un fichier, il sera disponible sur place. Cela réduit considérablement la latence d'accès, temps de réponse et la consommation de la bande passante. PDDRA se compose de trois phases:

- mémoriser les motifs d'accès aux fichiers,
- la demande d'un fichier et de la réplication;
- la pré-extraction et le remplacement.

L'algorithme a été testé à l'aide d'un simulateur de réseau OptorSim développé par projets européen grille de données.

Les résultats des simulations montrent que l'algorithme proposé à de meilleures performances en comparaison avec d'autres algorithmes en termes de temps d'exécution, l'utilisation efficace du réseau, le nombre total des répliques et le pourcentage des fichiers stockés.

Stratégie7 : ***Schéma réplique du Groupe de mise en cache pour les réseaux de la grille optique*** **[HK13]** :

Ce travail proposeun schémade remplacement de cachenommé« Group RéplicaCaching Schème » pour les réseaux de la grille optiques. Dans ces derniersles fichiers de données pourl'exécution du travailsont répliquéssur plusieursserveurs afin derépartir les charges

Les clientstéléchargent cesfichierspar l'intermédiaire deliaisons optiqueset de les stocker selon les besoins. Le téléchargement du fichier est bloqué quand les chemins optiques correspondants ne peuvent pas être établis. Laprobabilité de blocagedetéléchargement de fichiersdépend de l'emplacementdeces derniers.

Le cas où le stockage d'un fichierpour un client est idéal car le blocage des établissements du chemin optique ne se produit pas. Cependant, la taille du stockage du client est limitée. Pour utiliser efficacement des ressources du stockage des clients, notre plan proposé se concentre sur le fait que les clients peuvent télécharger des fichiersstockés dans les serveurs environnants avec une basseprobabilité du blocage.

Le schéma proposéconcerneun groupe destockaged'un clientetses serveurscomme unstockage.En particulier, ils stockent préférentiellementdifférents fichiers.En agissant ainsi, la probabilité qu'unfichier est stockéau niveau du clientousesserveursaugmente.À travers les expériences de la simulation, les auteursmontrent que le plan proposé améliore efficacement le téléchargement d'un dossier en diminuant la probabilité du blocage.

Stratégie8 : ***PHFS une méthode de réplication dynamique, pour diminuer la latence d'accès dans la grille de données multi-niveaux*** **[KIS11].**

PHFS (prédictive hiérarchique propagation rapide) c'est une nouvelle méthode de réplication dynamique dans une grille de données multi-niveaux, Considérant la localité spatiale, cette méthode essaie de prévoir les besoins futurs et les prés-répliquent de manière hiérarchique pour augmenter localité accès et par conséquent améliorent les performances.

Les PHFS sont une méthode qui prédit les requêtes de fichier subséquentes sur la base des relations entre les fichiers. Il est également indiqué que la relation entre les fichiers peut être déduite des modèles d'accès précédents. Ainsi, dans les PHFS, nous avons besoin d'un mécanisme pour garder les accès précédents et de les analyser pour trouver les relations. Par conséquent, PHFS comporte trois phases: surveiller, analyser, et de réplication.

- ✓ surveillance: un agent logiciel dans le nœud racine recueille les informations d'accès de fichier de tous les clients dans le système et les met dans un fichier journal
- ✓ Analyse : le moteur de l'exploitation minière fait l'extraction de données sur les fichiers journaux et découvre des informations utiles comme la relation entre les fichiers.
- ✓ Réplication: la phase de réplication est responsable de la gestion de répliques et aux deux étapes suivantes.
    - o Le gestionnaire de réplicationde la situationdansle nœud racineutilise les informationsfournies parle moteur del'exploitation minièreetrendPWS (predictivework set) pourle fichier qui est demandé pour la première fois.

L'étaped'adaptationde laconfigurationde réplication: cette étapeest responsable del'adaptationde laconfiguration de la réplicationà la situationexistant à labase du mécanismede priorité.

Strategie9 :stratégie de remplacementdérépliqueappelé PRA(PredictionReplica Replacement ***basé sur taille de la réplique et le coût de la bande passante.***[MKR13]

Les auteurs de cette stratégie présentent un algorithme de placement de réplique dynamique basé sur la prédiction de l'accès futur de la réplique et le coût de transfère future.

Cette stratégie est proposée pour un domaine basé sur le réseau où le poids de chaque réplique est calculé pour prendre la décision de remplacement, Elle passe par 3 étapes, elle prédit la popularité du fichier et aussi examine la taille de la réplique avec son coût de la bande passante.

Étape 1 : la Fréquence d'accès de tous les fichiers f est calculée comme suit :

$$AF(f) = \sum_{t=1}^{N_T} (a_f^t \times 2^{-(N_T - t)}), \forall f \in F$$

Soit$N_T$soitle nombre d'intervalle de tempspassé,Fest l'ensemble defichiers qui ont étédemandées,eta_f^tindiquele nombre d'accèspourle fichierfàintervalle de temps t.

Étape2: Le coûtde la répliqueà remplacerest calculé en utilisant la formule suivante :

$$C(f) = \frac{S(f)}{B(f)}$$

OùS(f)soitla taille de laréplique, B (f)est la bande passantede toutes les répliques du fichier f et elle se calculecomme suite :

$$B(f) = \frac{\sum_{i=0}^{NR} Bi}{NR}$$

Où NR estle nombre de répliquesdefichierf. Comparez le coûtde toutes lesrépliques.

Une valeur V est donnée pour le fichier, V est calculé par la formule suivant

$$V(f) = \alpha AF(f) + \beta C(f) / \alpha + \beta = 1$$

Le résultat de simulation avec Optorsim montre que PRA fonctionne mieux Par rapport à LRU et LFU algorithmes et elle réduit le temps d'exécution moyen et le temps d'accès aux données même avec l'augmentation du nombre de jobs, il montre une meilleure performance.

III.4Conclusion :

La réplication est considérée comme une technique importante pour réduire le coût d'accès aux données de la grille, Le placement des répliques affecte considérablement les performances de la grille en termes de disponibilité des données et des temps de réponse aux requêtes. Plusieurs stratégies de placement ont été proposées dans la littérature. La plupart des propositions dépendent de l'ensemble des requêtes soumises à la grille et placent les répliques de manière dynamique.

On trouve que Les décisions de placement sont basées sur un modèle de coût, qui dépend de plusieurs facteurs tels que la largeur de bande, la taille de la donnée, la latence du réseau, et le nombre d'opérations de lecture et d'écriture.

IV.1 Introduction :

La programmation parallèle et distribuée concerne l'étude du fonctionnement d'unProgramme réparti sur plusieurs ordinateurs, reliés par un réseau.
La recherche sur la programmation distribuée est un sujet majeur dans l'informatique actuelle. Parmi les options qui sont possibles pour expérimenter des programmes distribués,Il y a la possibilitéde *simuler* l'application distribuée.
La plate-forme de test est alors virtuelle, et modelable à l'envi. Le programme testé n'est toutefois pas le programme réel, juste une modélisation de son fonctionnement adaptée à la simulation.

IV.2Définitions:

IV.2.1 La simulation :

La simulation est un outil pour étudier les résultats d'une action sur un élément sans réaliser l'expérience sur l'élément réel,c'est-à-dire d'exercer l'action souhaitée sur l'élément en cause pour pouvoir observer ou mesurer le résultat.

L'outil de simulation peut être utilisé comme un simulateur ou comme un émulateur ; un simulateur est un outil qui représente un système réel par contre l'émulateur est un outil qui agit comme un système réel.

IV.2.2 Le Simulateur : Un simulateur est un outil de mise en œuvre de la simulationdu système permettant de reproduire de façon virtuelle une situation. Il présente sous des conditions contrôlables et observables l'évolution du modèle du phénomène.

IV.3 Quelques outils de simulation de grille :

Dans la littérature beaucoup d'outils standards et spécifiques à l'application ont été établis parmi lesquels nous pouvons citer :

IV.3.1Le simulateur GridNet :[LSSD02]

GridNet est un simulateur qui analyse les stratégies de la réplication de données en grilles de données.

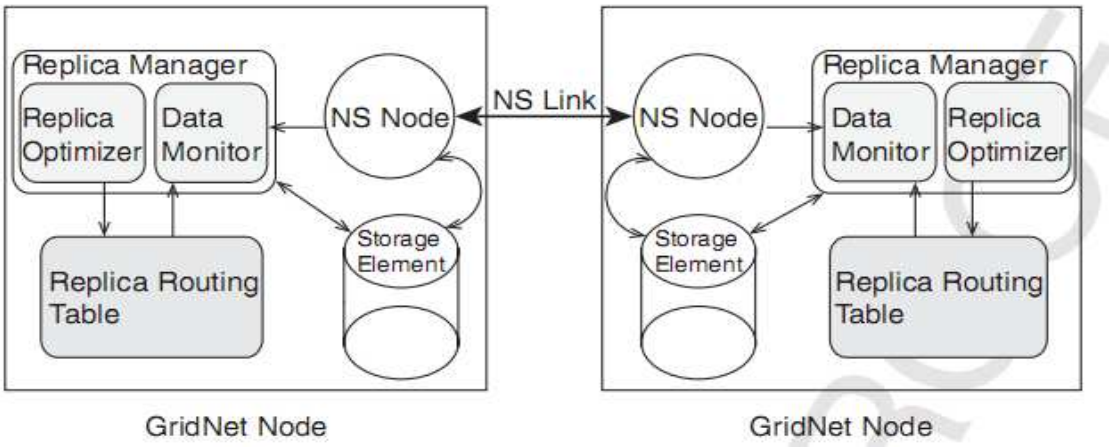
Il est écrit en C + + et construit au-dessus du simulateur de réseau NS2. Il fournit les capacités de simulation de réseau pour les nœuds, les liens et les messages.

GRIDNET pourrait être développé dans un réseau de communication de données très fiable et très susceptibles de survivre en charge le trafic entre un grand nombre de nœuds d'une manière efficace et efficiente.

Principe :

- Paquets représentant les demandes des utilisateurs de la grille ainsi que le début et la fin de la transmission des données de la grille en NS.
- transférer le contrôle de la simulation à GridNet.
- Ce dernier simule la décision de réplication à chaque nœud et génère nouveau trafic NS (demande de transfert vers d'autres nœuds ou l'envoi des données demandées au client).

Architecture:



FigureIV.1 Architecture de simulateur GridNet

Chaque nœud de GridNet consiste en trois éléments suivants:

1. un nœud NS de base.
2. un élément de stockage.
3. un gestionnaire de réplique ou d'un agent de surveillance.

Chaque nœud de GridNet maintient également une table réplique de routage

IV.3.2Le simulateur GSSIM :[KNOW07]

GSSIM(The GridScheduling Simulator) c'est un Outils de simulation de réseau, fournissent des cadres pour simuler l'ordonnancement d'applications dans les infrastructures de grille différentes.

Le cadre GSSIM est basé surGridSim et Sim-Java2 paquets. Toutefois, il fournit une couche supplémentaire au-dessus de la GridSim ajout de fonctionnalités pour permettre la modélisation simple et flexible des composants d'ordonnancement de la grille. GSSIM fournit également un module générateur avancé utilisant des charges de travail réelles et synthétiques.

- GSSIM partage des solutions entre le simulateur et l'environnement réel.
- GSSIM est un outil pour génération des taches chargées (Workload).
- GSSIM fournit une automatisation complète pour la tâche.
- Portail de l'échange des taches chargés,les algorithmes de simulation mises enœuvre et les résultats.

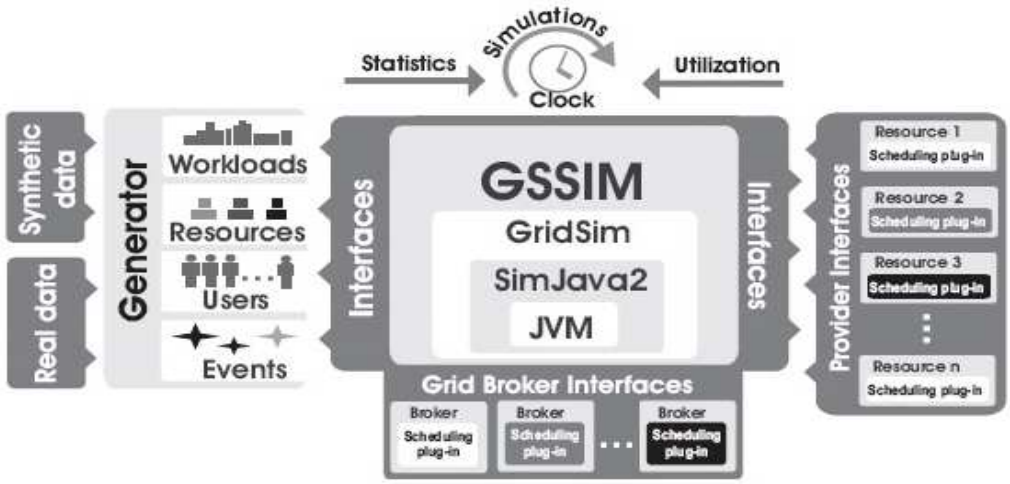
Architecture :

Le Framework est basé sur GSSIM GridSim et SimJava2 packages. Toutefois, il fournit une couche supplémentaire au-dessus de GridSim l'ajout de fonctionnalités pour permettre facile et flexible une modélisation des composants d'ordonnancements de la grille. GSSIM fournit également un module générateur avancé à l'aide réelle et synthétique de la charge de travail (workloads).

GSSIM distingue deux types d'ordonnancement :

- Les courtiers (Grid brokers) et les fournisseurs de ressources (ressources providers) de la grille.
- Les stratégies ordonnancement (SchedulingStrategies) multiples peuvent être prises dans les deux niveaux.

Les données d'entrée(input data) peuvent être lues à partir de sources réelles ou générées



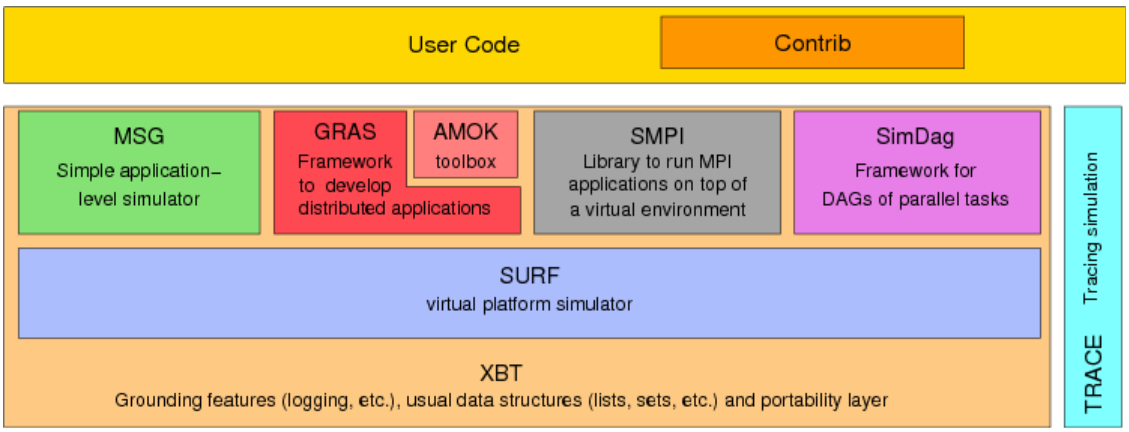
FigureIV.2 Architecture modulaire de GSSIM

IV.3.3Le simulateurSimGrid :[Cas01]

SimGrid est une boîte à outils pour la simulation d'applicationsdistribuées dans des environnements distribués hétérogènes. L'objectif central du projet est de faciliter la recherche dans les domaines de la programmation d'applications parallèles et distribuées, sur des plates-formes distribuées allant du simple réseau d'ordinateurs aux grilles de calcul.

Ce logiciel est développé en langage C, mais il est possible de l'utiliser dans d'autres langages tels que Java, Ruby ou Lau, grâce à des API (bibliothèque de fonctions destinées à être utilisées par un autre programme ou module.) spécifiques.

Il est consisté de 5 ´éléments : (Agents – Endroits – Tâches – Chemins – Canaux)



FigureIV.3 Schémas des différents modules de SimGrid

SimGrid est découpé en différents modules:

– **SURF**, noyau de simulation de simGrid (produit les fonctionnalités nécessaires pour la plate-forme virtuelle)

–**SimDag:** spécifier les heuristiques comme des tâches parallèles DAG (Directedacyclic graph).

–**MSG :** spécifier les heuristiques comme des processus concurrents Séquentiels.

–**GRAS:** développer des applications réelles, à étudier et à mettre au point dans le simulateur.

–**SMPI:** simuler MPI codes (Message Passing Interface).

– **XBT**, utilisé par tous les autres modules. Il implémente des fonctions de gestionde la mémoire, ainsi que des structures très utiles, comme les tableaux dynamiques, ou les graphes.

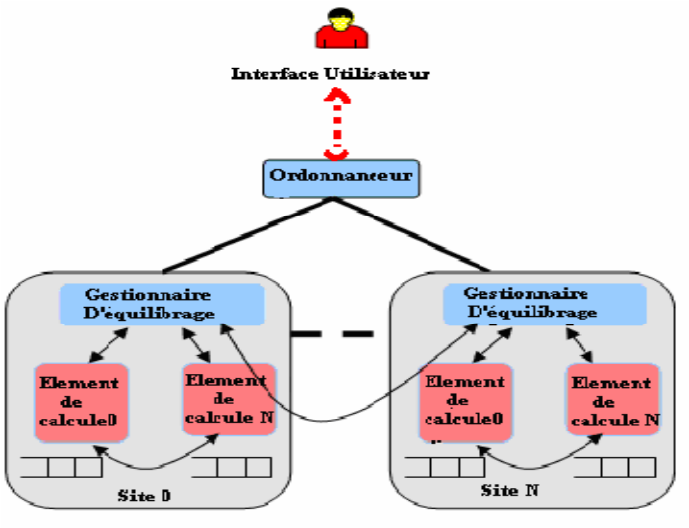
SimGrid étant destiné à simuler des applications d'ordonnancement, il fournit des fonctions avancées pour la gestion des tâches avec la prise en compte des dépendances.

IV.3.4Le simulateur OrientéSim:[BM08]

OrientéSim est un outil de simulation de grille, écrit en Java, il fournit des primitives pour la création et l'ordonnancement des tâches indépendantes et permet d'évaluer les paramètres de performance d'un modèle distribué pour résoudre le problème d'équilibrage de charge dans les grilles de calcul.

Dans OrientéSim il n'existe pas des protocoles réseaux ou des normes qui doivent être suivies. Tous les éléments sont reliés en utilisant des liens logiques.

La figure IV.4présente les différents composants du simulateur OrientéSim



FigureIV.4 Architecture d'OrientéSim.

D'un point de vue architectural OrientéSim est composé de:

_ *Interface utilisateur:* à travers laquelle on peut générer le fichier de configuration d'une grille (nombre de sites, nombre d'éléments de calcul, leurs caractéristiques, période d'envoi des informations de charge, largeur de bandes, etc...) ;

_ *Ordonnanceur:* effectue l'ordonnancement des tâches selon trois stratégies.

(i) *Aléatoire:* les tâches sont distribuées aléatoirement sur les éléments de calcul,

(ii) A priorité fixe: la tâche contenant le plus grands nombresd'instructions est assignée à l'élément de calcul le plus puissant,

(IV) *Round robin* : la première tâche est assignée au premier élément de calcul, la deuxième tâche au deuxième élément de calcul etc. …, d'une façon circulaire.

_ *Sites:* fournissent les ressources de calcul nécessaires à l'exécution des tâches soumises par l'ordonnanceur.

_ *Gestionnaire d'équilibrage:* chaque gestionnaire participe au maintien des informations de charge et à l'équilibrage de la charge globale des éléments de calcul de la grille. Les différents gestionnaires peuvent échanger leurs informations de charge.

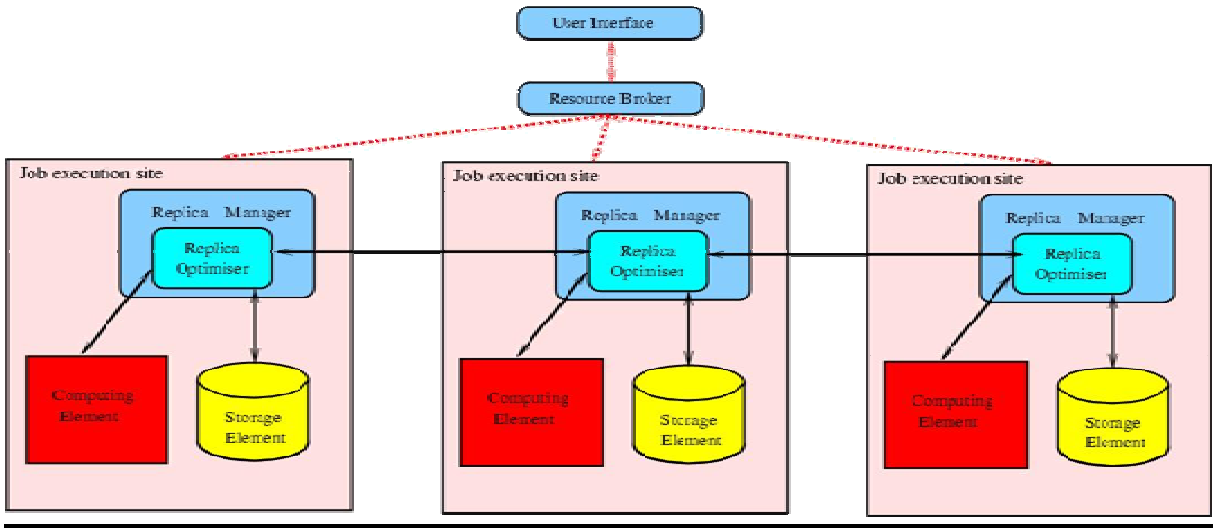
IV.3.5Le simulateur Optorsim :[BCCMZ03]

Définition :

OptorSim est un paquetage Java pour simuler des grilles de données. Il est basé sur l'architecture de DataGrid4 qui est une grille européenne construite dans l'objectif d'étudier le calcul intensif et l'interaction avec des bases de données à grande échelle (de 100 à 1000 TeraBytes). Lors de la simulation, chaque site fournit des ressources de calcul et de stockage.

Lorsqu'une tâche est exécutée, elle utilise des ressources locales. Un gestionnaire de réplication est chargé du transfert des données d'un site à un autre, en fonction des besoins. Pour cela, il fait appel à l'optimiseur de réplication appelé Optor. Il est ainsi possible de comparer différentes stratégies de réplication.

Architecture:



FigureIV.5 Architecture d'OptorSim

OptorSim est donc constitué de :

– Une interface utilisateur pour accéder à la grille.

– Un Resource Broker qui est un agent responsable de l'ordonnancement des requêtes soumises via l'interface.

– Les sites de la grille : Chaque site est en général un cluster de machines constituéesde :

1. Une ou plusieurs ressources de calcul (CE : ComputingElement). Sur la versiond'OptorSim, on accepte unCE seulement. Un CE peut contenir un ou plusieurs processeurs ou "workernodes".
2. Une ou plusieurs ressources de stockage (SE : Storage Element).
3. Un gestionnaire des répliques, responsable de la création, suppression, placement, cohérence des répliques.

Fonctionnement global :

La grille simulée est composée de sites. Un site contient une unité de calcul appelé ComputingElement (CE), un gestionnaire de réplicas, le Replica Manager (RM) ainsi qu'une unité de stockage de données, le Storage Element(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).

Détaillons un peu le fonctionnement de ces différents éléments.

ComputingElement (CE):consiste à simuler l'exécution des requêtes reçus. Lorsqu'un CE s'apprête à exécuter une requête il s'adresse au RM afin de récupérer les différentes données nécessaires à l'exécution.

La durée d'exécution des requêtes est identique pour toutes les requêtes quel que soit leur type, les données auxquelles elle accède ou le CE sur lequel elle est exécutée.

Rplica Manager (RM) : concerne la localisation et la gestion des mouvements des données. Il contient un Optimizer, il s'agit d'un objet dans lequel sont définies les stratégies de réplication et de suppression de données. Il s'agit donc de stratégies locales au niveau d'un site.

Storage Element (SE) : concerne le stockage des fichiers et est caractérisé par sa capacité de stockage (en Mégaotets). Un même fichier peut être stocké sur différents SE au même moment, et chaque SE a la charge de décider quel fichier doit être supprimé si de l'espace est requis pour un nouveau fichier alors que le SE est déjà plein.

Utilisateur : Il est la source des requêtes. C'est lui qui crée les requêtes selon un schéma bien précis. Là encore, divers schémas sont fournis et le choix de celui qui sera utilisé est fait dans la configuration.

Resource Broker (RB) :C'est l'organe central de décision. Il est capable de communiquer avec les différents CEs et RMs pour récupérer toutes les informations nécessaires à l'ordonnancement. Plusieurs algorithmes sont disponibles pour effectuer l'ordonnancement, le choix de celui qui doit être employé est défini dans les fichiers de configuration du système.

Les Avantages et Les Inconvénients d'OptorSim :

Les Avantages :

- ✓ Simulation en un temps minime
- ✓ Il permet également à un utilisateur de visualiser les performances de l'algorithme
- ✓ Il permet de tester et comparer des algorithmes d'optimisation dans des scénarios différents dans la grille

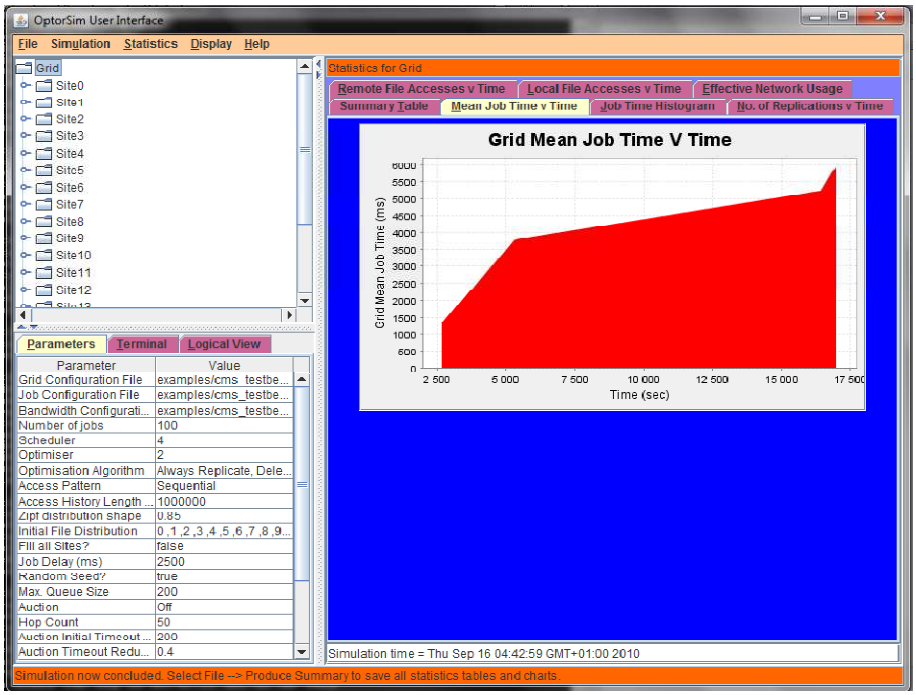
Inconvénients :

- ✓ Ne marche qu'avec Netbeans

- ✓ A besoin d'une configuration détaillée

L'interface graphique :

OptorSim offre une interface graphique qui facilite la lecture des paramètres et résultats.
L'interface est divisée en deux parties :la partie gauche pour les paramètres de configuration et la partie droite pour les résultats.



FigureIV.6 Interface graphique d'OptorSim

IV.4Conclusion :

Dans ce chapitre nous avons cité quelques simulateurs de grille. Mais en raison de notre projet, nous sommes intéressés à une étude plus détaillée des simulateurs de grilles informatiques OptorSim, qui sont destinées particulièrement à l'implémentation d'algorithmes de réplication et l'évaluation des performances de la grille.

V.1 Introduction :

Dans ce dernier chapitre, nous commencerons par présenter la topologie de la grille et le banc d'essaiutilisé. Ensuite nous présentons notre amélioration proposéedela stratégie de réplication PRA (PredictionReplicareplacementAlgorithme), en détaillant les modifications et les formules que nous proposons pour résoudre le problème rencontré dans la stratégie PRA. Après nous détaillons nos expérimentations et les résultats obtenus parsimulation sous OptorSim. Une comparaison entre les résultats de notre approche et plusieurs stratégies de réplication qui sont déjà implémentées.

Enfin, nous conclurons par une synthèse des résultats.

V.2Topologie de la grille:

La topologie de grilleest celle dubanc d'essaiCMSqui se compose de8routeurs et20nœuds, ces dernierssontdispersés entre l'USA et l'Europe.Nous avons expérimenté notre stratégie sur le banc d'essai qui est détaillédanslaTable V.1 et sa topologie du réseau affichée sur laFigure V.1.

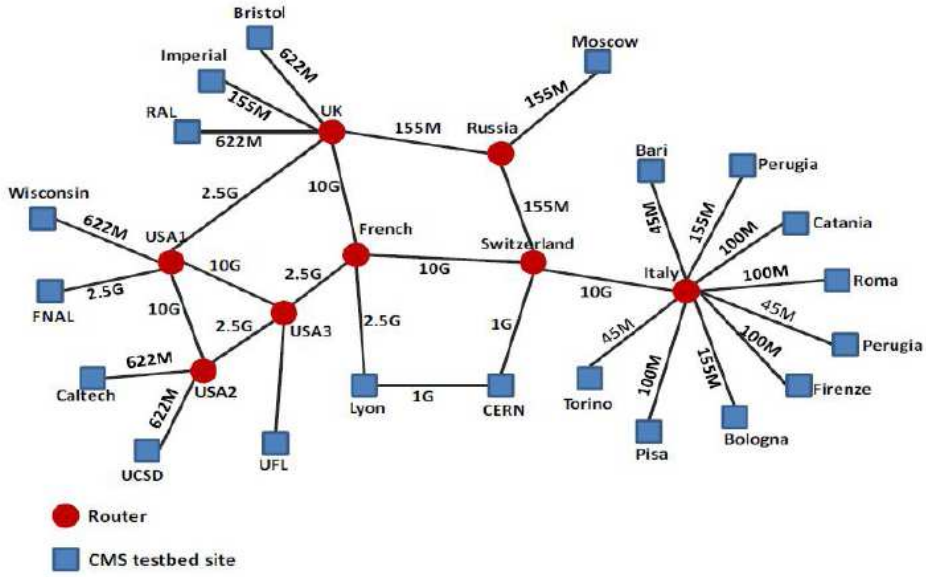


Figure V.1 LaTopologie simulée.

Nombre de fichiers	97
Taille de chaque fichier	10 GB
Nombre de jobs	1000
Temps d'exécution d'un job	05
Délai entre jobs	2500
Bande passante	45 – 10000 MB
Nombre de sites	28
Nombre de SE par site	1
Capacité d'un SE	50 – 100 GB

Table V.1 le banc d'essai CMS.

V.3Approche proposée :

Après une étude comparative entre un ensemble des stratégies de réplication qui existe, on a proposé une amélioration de la stratégie PRA.

V.3.1 Principe de la Stratégie PRA:

La stratégie de placement des répliques PRAest appliquée lorsque le site choisi pour placer la nouvelle réplique à une capacité de stockage insuffisant. PRA est une stratégie qui sélectionne le fichier victime à partir des fichiers qui sont stockés dans l'élément de stockage ciblé afin de libérer un espace de stockage suffisant pour la nouvelle réplique.

Le placement des répliques pour approche PRA passe par deux étapes principales:

- **Phase d'évaluation du fichier :** dans cette phase les auteursdésignent une valeur de prédiction pour chaque fichier selon des informations historiques et le coût de transfert, cette étape est divisée en trois parties :
  - **Évaluation de popularité de la réplique**: la modèle demi-vie est appliquée pour déterminer la réplique popularité.
  - **Évaluation des coûts futurs de transfert :** le facteur de coût souvent affecté par certains facteurs tels que la taille de la réplique et la bande passante.

- **Priorisation des répliques :** une valeur est déterminée pour la réplique basée sur la popularité et le futur coût de transfert.

❖ **La phase d'élimination:**Dans cette phase, les fichiers victime sont éliminés selon la taille de la nouvelle réplique crée et la capacité de stockage nécessaire.

❖ **L'évaluation de la popularité de réplique :** faite par la formule AF (fréquence d'accès) .AF montre l'importance de l'accès historique dans déférent intervalle de temps.

NT est le nombre d'intervalles de temps passé, F est l'ensemble des fichiers qui y sont dans un élément de stockage cible. **aij** indique le nombre d'accès pour le fichier i à intervalles de temps j. La fréquence d'accès pour le fichier F est représentée comme:

$$AF(f) = \sum_{t=1}^{N_T} (a_f^t \times 2^{-(N_T - t)}), \forall f \in F$$

❖ **Évaluation des futurs coûts de transfert :**

Le facteur de coût est un facteur important au cours du processus de remplacement de réplique, qui est souvent affectée par des facteurs tels que la taille de la réplique et de la bande passante.

Trop de consommation de bande passante peut bloquer le réseau et améliorer la possibilité d'apparition de défauts au cours du processus de transfert.

La bande passante B (f) est définie comme la moyenne de la bande passante de toutes les répliques de la grille de données et NR est le nombre de répliques du fichier f et Bi est la bande passante de la ième réplique.

$$B(f) = \frac{\sum_{i=1}^{N_R} B_i}{N_R}$$

Donc le coûtde la réplique, C (f), est défini comme:

$$C(f) = \frac{S(f)}{B(f)}$$

S(f) : la tailledu fichier

❖ **La priorisation des répliques :**

La priorisation des répliques est déterminée sur la base de la valeur de réplique dans chaque élément de stockage. Nous définissons la valeur de réplique V (F) Qui se compose de futur coût de transfert de réplique et de la réplique populaire.

$$\boldsymbol{V(f) = \alpha AF(f) + \beta C(f)}$$

Lespondérations des facteurs α et β définis dansV(F) doivent être choisisde telle sorte que α+ β=1.

Cependant, cette approche dans la phase d'évaluation de la popularité du fichier suppose que le tauxde décroissance estconstant etest égale à½ cela signifie quetous les fichiersdécroissent dans le mêmetauxquel que soitle tauxde chacund'accès. En conséquence, le taux depoidsdedéclinaisonsera plus lent.

Par la suitel'élément de stockageprendra plus de temps poursupprimer lesfichiers indésirables (lesfichiersmoins importants).

V.3.2 L'amélioration proposée :

Pour résoudre ce problème on a calculé la popularité du fichier par une autre méthode dont le taux de décroissance /croissance est varié en fonction dutauxde l'accès fichier.

Cela signifie que letauxdecroissance/ décroissancede chaque fichiern'est pas le même.

Le taux de croissance/décroissance qu'on a proposé est calculé comme suit :

On suppose que n est le nombre intervalle passé*I1, I2,……In.*

Et le *Nbre_a*(n) est le nombre d'accès pour le fichier f dans l'intervalle n :

Nbre_a(1), Nbre_a(2),……,. Nbre_a(n)

Chaque intervalle du temps a un taux de croissance/ décroissance du fichier et pour le récupéreron calcule la variation de nombre d'accès entre chaque deux intervalles par la formule suivante :

$$\boldsymbol{Taux(i) = Nbre\_a(i+1) - Nbre\_a(i)}$$

Donc on a n-1 taux à calculer :

$Taux(1) = Nbre\_a(2) - Nbre\_a(1), Taux(2) = Nbre\_a(3) - Nbre\_a(2) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, Taux(n-1) = Nbre\_a(n) - Nbre\_a(n-1)$

Enfin la moyenne de taux de tous les intervalles est :

$$\boldsymbol{Taux\_m = \sum_{1}^{n-1} Taux/(n-1)}$$

Le *taux_m*c'est la moyenne detaux de croissance/ décroissance dans le passé

$$Future_a = Nbre_{a(plus\ récent)} * Taux\_m$$

C-a-d:

$$Future_a = Nbre\_a(n-1) * Taux\_m$$

V.3.3 Algorithme :

Le code algorithmique de la stratégie de remplacement de la réplique modifie PRA est représenté comme ci-dessous :

1. ***Si*** *le fichier* $(\boldsymbol{f}_i)$ *éxist dans le site*

   Ne rien faire

2. ***Si*** *le* ***SE.*** *Size ()>* $\boldsymbol{f}_i$*.Size ()*

   Avancer

3. ***Si*** *l'espace libre dans* ***SE>*** $(\boldsymbol{f}_i)$*.Size ()*

   Stocker la réplique dans le site

4. ***Sinon*** *//*

{

Sélectionner le fichier en utilisons la méthode de PRA de suppression

Supprimer les fichierssélectionnés ;

}

5. ***Si*** *(espace libre est suffisant dans le SE)*

Stocker la nouvelle réplique

Prediction Replacement Method :

1. *Int **ALocal** []= (L1,……..Ln);*

***RS**=l'espace Libre ;*

2. *Calculer la valeur de chaque fichier utilisant la popularité et le cout de transfère de fichier.*
3. *Stocker les fichiers dans la liste **Alocal**en utilisant les valeurs par ordre décroissant*
4. ***Tantque**(**Alocal** !=Null&&**RS**<(f_i).Size ())*

Faire

{

***MinR**=PremieFichier(**Alocal**) ;*

***RS**=**RS**+ PremieFichier.Size() ;*

*Supprimer (**Alocal**,**MinR**) ;*

***RS**=**RS**+**MinR**.Size() ;*

}

5. ***Si** (**RS**>=(f_i).Size ()*

Stocker la nouvelle réplique;

V.3.4 Exemple:

Afin de comprendre comment fonctionne la stratégie PRA modified qu'on a proposé on prend l'exemple suivant :

On suppose que l'élément de stockage contient 10 fichiers et un espace libre de 300 MB

(EL=300 MB), Supposons aussiquenous avons un fichierNfavec une taillede 1000MBil doit être placé.

Tant que l'espace de stockage est insuffisant pour stocker la nouvelle réplique (EL<Nf.size), on passe donc à l'algorithme de modified PRA de suppression et on applique les étapes suivantes :

A. la phase d'évaluation de fichier :

Supposons que les informations de l'accès historique pour chaque fichier sont disponibles.

Par exemple, un fichierXa été consulté12 fois, 15 fois ,20 fois et 10 fois dans lepremier intervalle de temps, lesecond,letroisièmeet le quatrième intervalle respectivement.

1 : on calcule le taux de croissance/ décroissance moyenne

$$\boldsymbol{Taux(i) = Nbre\_a(i+1) - Nbre\_a(i)}$$

- ✓ *Le taux du 1 Ier intervalle :*

$$Taux(1) = 15 - 12 = 3$$

- ✓ *Le taux du 2 Ime intervalle :*

$$Taux(2) = 20 - 15 = 5$$

- ✓ *Le taux du 3 Ime intervalle :*

$$Taux(3) = 10 - 20 = -10$$

Et par conséquence le taux moyenne :

$$\boldsymbol{Taux\_m = \sum_{1}^{n-1} Taux/(n-1)}$$

$$\boldsymbol{Taux_m = (3 + 5 - 10)/3}$$

$$\boldsymbol{Taux_m = -0.66}$$

2. on calcule le coût de transfère de ce fichier :

On a: $\boldsymbol{C(f) = \frac{S(f)}{B(f)} = \frac{1000}{b(f)}}$

$B(f) = \frac{\sum_{i=0}^{NR} Bi}{NR} = \frac{100+311+155+77}{4} = 160.75$

$Donc C(f) = 6.22$

3. et enfin la valeur de fichier :

$$v = \alpha * Taux_m + \beta * cost$$

V=0.7*(-0.66) +0.3*6.22=1.4

Ensuite on calcule les valeurs de tous les fichiers qui ont été demandés. Un fichier avec la plus grande valeur est choisi comme le fichier populaire.

4-***Trier les fichiers dans l'ordre croissant*** : en fonction de ces valeurs.

Ensuite, identifier le fichier victime. Qui a la plus petite Valeur. Le Tableau 1 montre l'exemple des fichiers stockés dans un élément,le stockage avec la valeur du fichier correspondante et la taille du fichier

Le nom du fichier	La valeur du fichier	La taille du fichier
Fichier 5	**13**	**300**
Fichier 8	**31**	**789**
Fichier 2	**42**	**123**
Fichier 9	**45**	**1588**
Fichier 4	**55**	**154**
Fichier 1	**66**	**100**
Fichier 7	**71**	**406**
Fichier 3	**78**	**1111**
Fichier 6	**80**	**488**
Fichier 10	**92**	**8852**

Table V.2: exemple des fichiers de données stocké dans un élément de stockage avec leur valeur correspondante et leur taille

B- la phase d'élimination :

Dans cette étape, nous éliminons les fichiers victime selon la taille de la nouvelle réplique créée et la capacité de stockage nécessaire.

Comme montre le tableau, le fichier victime est le fichier5, car c'est le fichier qui a lavaleurminimale.

EL=El+fVictime.size=600;

Dans ce casil faut supprimer un autre fichier parce qu'on a besoin de 800 Mo pour stocker la nouvelle réplique (EL**<Newf.size)**, donc ladeuxieme victime est le fichier 8.

EL=300+789=1089 ;

Nous devons doncsupprimerdeux fichiersafin destockerunautre.

V.4 Les Expérimentations et Résultats :

Dans lapremièreexpérimentation, nous exécutons notre stratégie de réplique (Modified PRA) en combinaison avec la stratégie originale PRA et les algorithmes qui sont déjà implémentés dans OptorSim : (Simple – LRU – LFU– EconomicBinomial – Zipf- basedeconomic model).

- **Les algorithmes d'optimisation :**

1- Aucune réplique : Cet algorithme ne réplique jamais un fichier. La distribution des réplications initiales de fichier est décidée au début de la simulation et ne change pas pendant son exécution. Cet algorithme renvoie le nom du fichier avec le temps d'accès le plus rapide.

2- Le Modèle Économique (ME) : Lorsqu'un job a besoin d'une donnée qui n'existe pas sur le site et sur lequel il s'exécute, on estime deux coûts :

– le coût d'une exécution à distance du job: L'exécution se fait sur le site contenantla donnée puis transfère des résultats au site lançant le job.

– le coût d'une exécution locale après réplication de la donnée sur le site lançant le job:le modèle économique choisit la solution qui donne le meilleur coût.

3- Least RecentlyUsed (LRU) : Lorsqu'un job a besoin d'une donnée qui n'existe pas sur le site et sur lequel il s'exécute, on la réplique automatiquement depuis le site où elle se

trouve. En cas de manque d'espace de stockage pour l'héberger, il supprime la réplique la moins récemment utilisée.

4- Least FrequentlyUsed (LFU) : Lorsqu'un job a besoin d'une donnée qui n'existe pas sur le site sur lequel il s'exécute, on la réplique automatiquement depuis le site où elle se trouve. En cas de manque d'espace de stockage pour l'héberger, il supprime la réplique la moins fréquemment utilisée.

➢ **politiques d'ordonnancement :**

Pour simulé notre approche nous choisissons les configurations d'accèscités ci-dessus, et les différentes politiques d'ordonnancementqu'ils sont :

- Randomscheduler :(L'ordonnanceur aléatoire) choisit aléatoirement un nœud de traitement pour exécuter un travail spécifique,
- Shortest Queue scheduler :(L'ordonnanceur de la file d'attente la plus courte) calcule la longueur de toutes les files d'attente des nœuds de traitement et choisit un qui a le moindre nombre de travaux qui attendent dans la file d'attente.
- Access Costscheduler : (L'ordonnanceur de coût d'accès) assigne le travail à l'élément de calcul où le coût d'accès au fichier est le plus bas.
- Queue Access Costscheduler : (L'ordonnanceur de coût d'accès de file d'attente) choisit l'élément de calcul qui a la plus petite somme du coût d'accès pour le travail et les coûts d'accès de tous les travaux attendant dans la file d'attente.

➢ **Les configurations d'accès :**

On a considéréQuatre configurations importantes d'accès :

- Sequentiel(Séquentielle) : l'ensemble des fichiers est ordonné, formant une liste de demandes successives.
- Random(Aléatoire) : des fichiers sont choisis aléatoirement à partir d'un positionnement avec une distribution de probabilité.
- UnitaryRandomWalk(Marche aléatoire unitaire) : l'ensemble est ordonné et les requêtes successives de fichiers sont exactement un élément à partir de la demande précédente de fichier, la direction est aléatoire.
- GaussianRandomWalk(Marche aléatoire gaussienne) : comme avec la marche aléatoire unitaire, mais les fichiers sont choisis par une distribution gaussienne portée sur la demande précédente de fichier.

➢ **Les critères d'évaluation :**

Dans notre simulation, nous analysons trois paramètres d'évaluation qui sont les suivants :

- ✓ **Le temps d'exécution des jobs :** C'est le temps total d'exécution de tous les jobs divisé par le nombre de jobs.
- ✓ **Le nombre de répliques :** C'est le nombre de répliques crée durant l'exécutiondes jobs, Ce nombre peut être amélioré par un placement stratégique des répliques.
- ✓ **La consommation des ressources réseau :**Ce paramètre indique l'occupation etl'usage du réseau.

- La configuration d'accès choisie dans notre simulation est : GaussianRandomWalk(Marche aléatoire gaussienne) et la politique d'ordonnancement maintenue est : Queue Access Costscheduler :(L'ordonnanceur de coût d'accès de file d'attente).

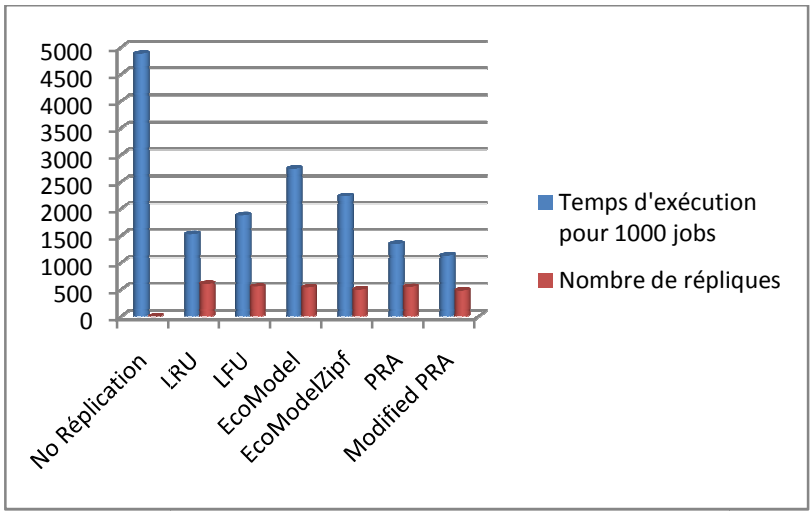


Figure V.2Récapitulation des scénarios de simulation de 1000 jobs

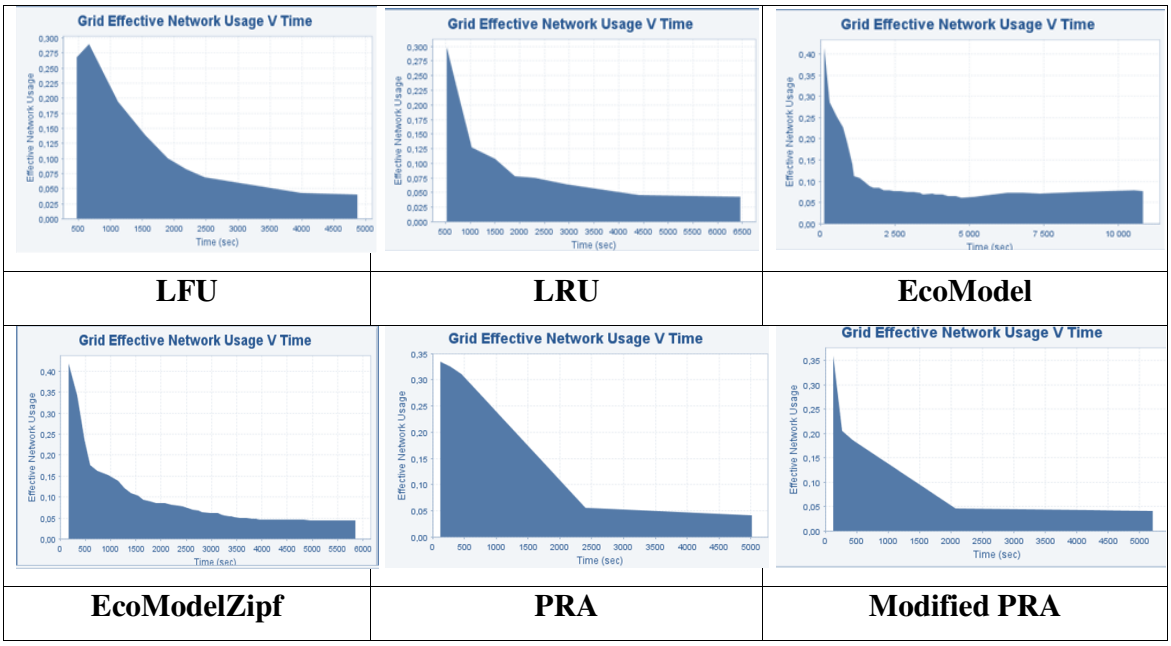


Figure V.3Évaluation de la consommation des ressources réseaux.

- Les résultats obtenus dans lesFigure**s (V.2, V.3)**montrent que notre stratégie (Modified PRA) améliore le temps d'exécution des jobs sans dégradation dans la consommation des ressources de réseau par rapport à toutes les autres stratégies.

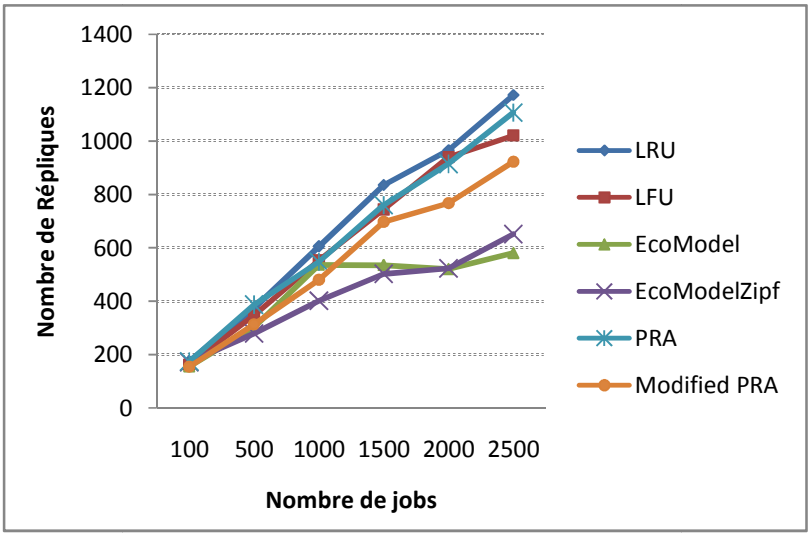


Figure V.4 Nombre de répliques basé sur la variation du nombre de travaux (jobs)

Les Figures (**V.4**, **V.5**, **V.6**) montrentconsidérablementque Modified PRAréduit le nombre de répliquespar rapport à PRA dans toutes les politiques d'ordonnancement, ainsi que le temps d'exécution des travaux (jobs) et tous les cas de configurations d'accès et utilisent le séquentiel accès.

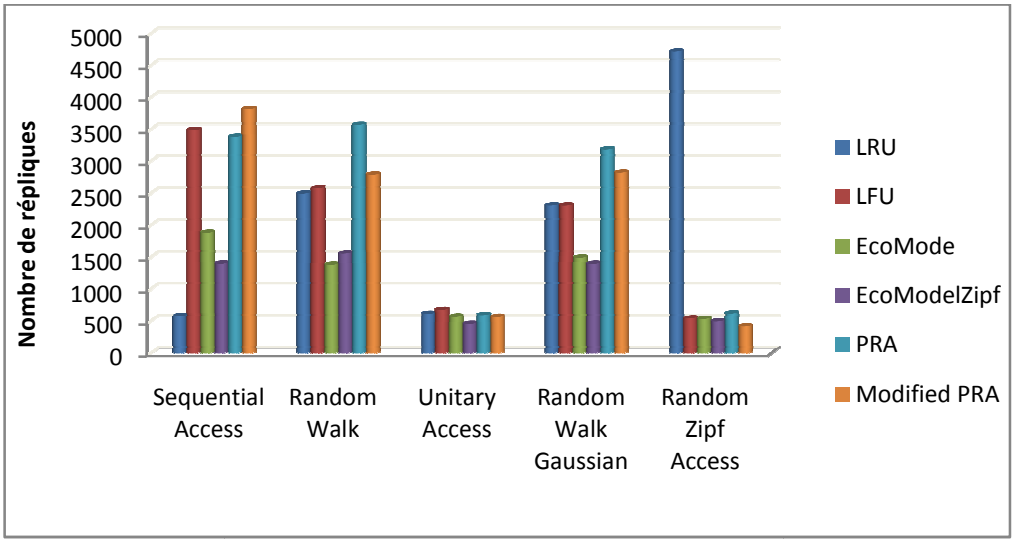


Figure V.5 Nombre de répliques basé sur les différentes configurations d'accès

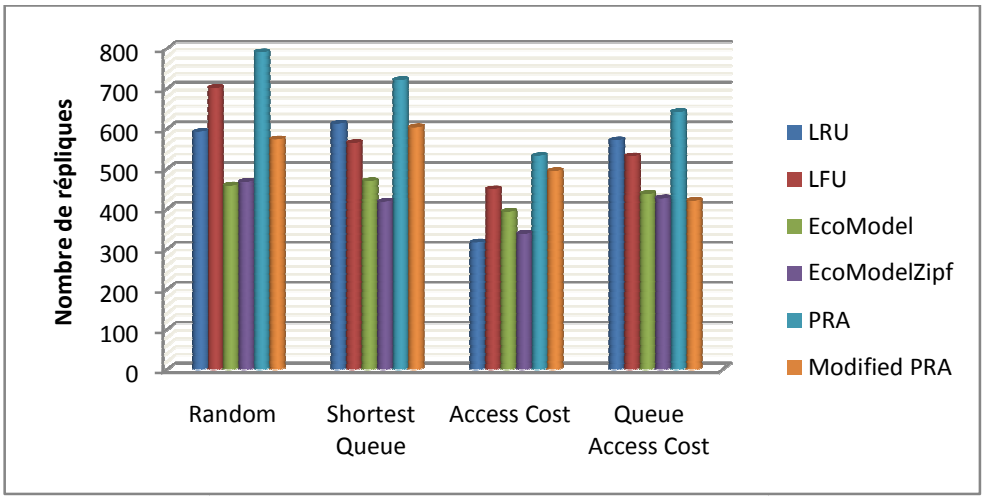


Figure V.6 Nombre de répliques basé sur les différentes politiques d'ordonnancement

Dans le **Figure V.7**et**Figure V.8** nous avons varié le paramètre fréquence des jobs "délai entre jobs" et nous testons notre stratégie en combinaison avec l'algorithme original PRA.

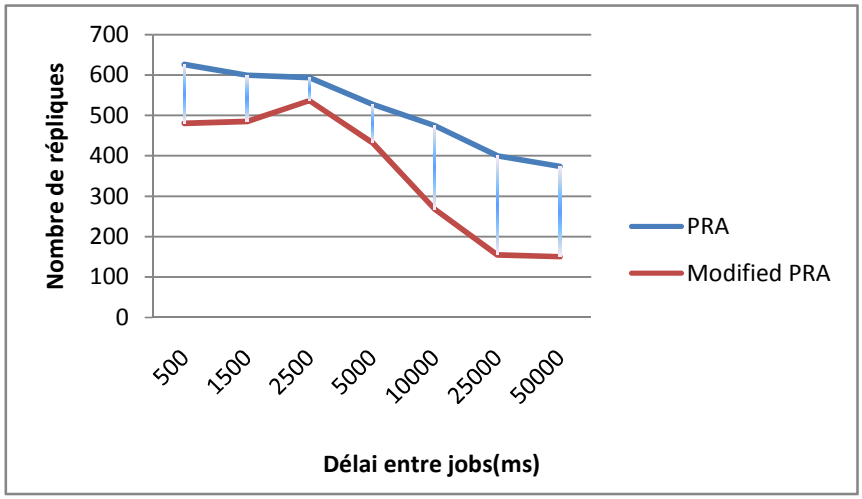


Figure V.7 Variation des temps d'exécution en fonction de la fréquence des requêtes.

- Pour le paramètre fréquence des requêtes,on remarque qu'il y a un gain de 10% à 32% sur le nombre de répliques crées durant l'exécution dans l'algorithme modified PRApar rapport à PRA, et le temps d'exécution des jobs est amélioré de 25% (voir **Figure V.8**)

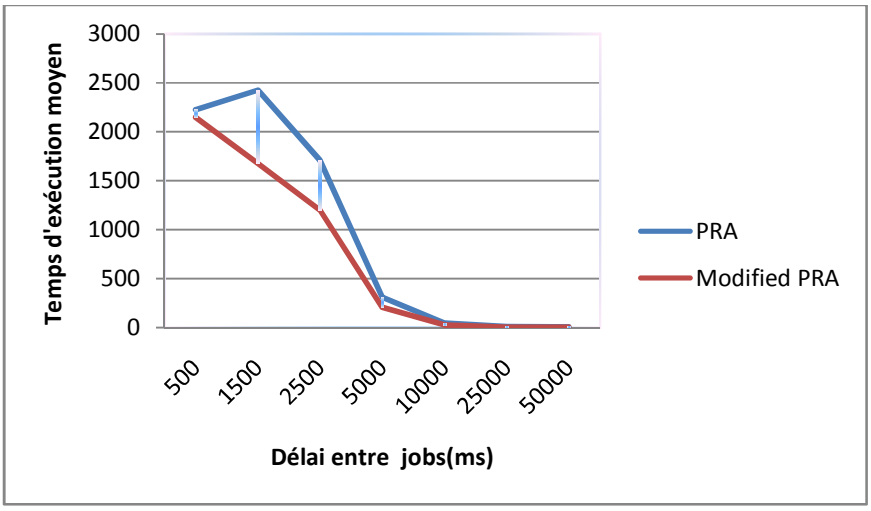


Figure V.8 Variation du nombre de réplicationen fonction de la fréquence des requêtes.

V.5 Conclusion :

A partir des amendements et des Ajouts qui sont ajoutés dans l'algorithme PRA et pour résoudre le problème du taux posé, nous avons atteint une nouvelle stratégie de réplication, cette stratégie basée sur la prédiction de nombre d'accès du fichier dans le futur avec un taux de poids de déclinaison plus rapide. Grâce aux simulations, nous avons pu estimer que notre algorithme donne de bons résultats par rapport à l'algorithme original PRA, Une amélioration considérable est observée dans le nombre de jobs et leurs temps d'exécution sans dégradation dans la consommation des ressources. Nous avons présenté dans ce chapitre les détails de notre amélioration proposant leréseau.

Conclusion générale

La grille informatique est un outil dont le but est de regrouper des ressources partagées, distribuées et hétérogènes afin de réaliser des actions globales difficiles vues impossibles à réaliser sur une machine unique. L'objectif de la grille est donc d'apporter à l'utilisateur la possibilité d'utiliser des ressources distantes ou de lancer une application qui demande beaucoup de ressources non disponibles localement.

Dans ce mémoire, nous sommes particulièrement intéressés à l'un des composants critiques de l'environnement de la grille à usage intensif de données qui est la réplication de données, le besoin de ce composant se fait sentir dans diverses zones d'analyse de données.

La réplication en informatique est un processus de partage d'informations pour assurer la cohérence de données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, latolérance aux pannes, ou la disponibilitédes données ainsi que les performances d'accès.Parmi les problèmes posés par l'utilisation de la réplication dans les grilles, nous mettrons l'accent sur le problème du placement des répliques, qui reste un vaste thème de recherche.

Nous avons dressé dans ce travail, un état de l'art sur lesstratégiesde placement des répliques,que nous avons illustrées par des travaux menés dans ce domaine. Pendant cet état, serrons notre attention un problème sur un algorithme qui s'appelle PRA (PrédictionRéplicareplacementAlgorithme), et nous avons vu que nous pouvons le résoudre.

PRA est une stratégie de placement des répliques basées sur la popularité de la réplique, elle est appliquée lorsque la nouvelle réplique a besoin d'un espace de stockage suffisant.Pour déterminer la popularité de la réplique, PRA calcule lafréquence d'accès de chaque fichier avec un taux de décroissance fixé, ce qui augmente le temps de suppression des fichiers moins importants par l'élément de stockage. Pour résoudre ce problème, nous avons alors proposé une solution qui consiste à recalculer la popularité du fichier par une autre méthode et prédit le nombre d'accès avec des formules qu'on aproposées, au moment quand nous avons élaboré la nouvelle l'algorithme obtenu, nous remarquons que notre proposition s'avère être une solution très intéressante pour réduire le nombre de jobs, leurs temps d'exécution et améliorer la consommation des ressources de réseau. Nous prouvons ceci par expérimentation sur le simulateur OptorSim.

Les perspectives qui nous pouvons ouvrir sont au niveau de l'amélioration de notre approche et au niveau de la prédiction d'accès du fichier. Les perspectives consistent à prévoir

lesbesoins futursde sitesde réseauavec plus de précisionen considérantplusieurs facteurs autres que seules leursséquences d'accès historique, nous prévoyons aussi d'examiner plusieurs facteurs pour déterminer la popularité de la réplique .En plus, Nous souhaitons à implémenter à l'avenir notre approche proposée dans un environnement plus réaliste.

[BBL00] - M.Baker, R.Buyya, and D.Laforenza. « The grid : International e_orts in global computing ». In Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science,andEducation on the Internet (SSGRR 2000).

[BCCMZ03] -W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K.Stockinger, and F. Zini, «*OptorSim – A Grid Simulator for StudyingDynamic Data ReplicationStrategies*», Int. J. of HighPerformance Computing Applications, 17(4), 2003, 403–416.

[Bil09] - R.BILEY, « Mise en place d'un système de réplication de base de données entre sites distants ». Université de Dschang - Licence Professionnelle ès Génie logiciel 2009.

[BM08] - YagoubiBelabbas et Meriem Meddeber. Conception d'un Simulateur de Grilles Orienté Gestion d'Équilibrage. Université d'Oran, Faculté des Sciences, Département d'Informatique, 31000 Oran, Algérie 2008.

[Cas01] - H. Casanova, *«Simgrid: A Toolkit for the Simulation of Application Scheduling»*. Proc. Of the First IEEE/ACM Int. Symposium on Cluster Computing and the Grid, Brisbane, Australia, 2001, 430–437.

[Cha10] - Z.CHALLAL, « la réplication de données dans les grilles biologiques ». Mémoire de magister, Ecole nationale Supérieure en Informatique -Alger ,2010.

[DHJSSSS01] - D. Düllmann, W. Hoschek, F. J. Jaén-Martínez, B. Segal, H. Stockinger, K. Stockinger, and A. Samar. « Models for replica synchronisation and consistency in a data grid ». In HPDC, pages 67–75, 2001.

[FK03] - I. Foster and C. Kesselman. « The Grid2: Blueprint for a New Computing Infrastructure ». Morgan Kaufmann Publishers Inc., 2003.

[Gri] - Gridcafe: http://www.gridcafe.org/FR/des-grilles-pour-des-besoins-differents.html.

[HK13] - K.Hirata, M.Kawahara. Group replica caching scheme for optical grid networks. « Optical Switching and Networking » (Impact Factor: 0.72). 07/2013.

[HKIL12] - F.Hanandeh, M.Khazaaleh, H.Ibrahim, and R.Latip ,Prince Al- Hussein bin Abdullah II Faculty of Information Technology, HashemiteUniversity, Jordan Irbid College, Al-BalqaAppliedUniversity, Jordan, Faculty of Computer

Science and Information Technology, UniversityPutra Malaysia, Malaysia ,2012.

[KIS11] -Leyli Mohammad Khanli ,AyazIsazadeh, Tahmuras N. Shishavan. PHFS: A dynamicreplicationmethod, to decreaseaccesslatency in the multi-tier data grid. À Cs Department, University of Tabriz, Tabriz, Iran .b University of Tabriz, Tabriz, Iran. (2011).

[KNOW07] - Krzysztof Kurowski , JarekNabrzyski ,Ariel Oleksiak , Jan Weglarz , GSSIM – Grid Scheduling Simulator. 1Poznan Supercomputing and Networking Center, 2Institute of Computing and Management Sciences PoznanUniversity of Technology. November 26, 2007).

[LSS02] H.Lamehamedi ,B.Szymanski, , and Z. Shentu, « Data ReplicationStrategies in GridEnvironments ». In Proceedings of the 5th InternationConference on Algorithms and Architectures for ParallelProcessing (ICA3PP '02) 2002.

[LSSD02] -H.Lamehamedi ,B.Szymanski, , Z. Shentu, E.Deelman,« Data ReplicationStrategies in GridEnvironments ». In Proceedings of ICAP '035TO APPEAR),Beijing,ChinaOctober 2002.

[LV04] - M.Lei and V. Vrbsky, A Data ReplicationStrategy to Increase Data Availability in Data Grids. Department of Computer Science University of Alabama.2004.

[MKKY03] - S.Min Park, J.Kim, Y.Ko, and W.Yoon, « Dynamic Data GridReplicationStrategybased on Internet Hierarchy ». AjouUniversity, South Korea, 2 College of Information TechnologyAjouUniversity, South Korea.2003.

[MKR13] - MaryamMehraban, Ahmad Khademzadeh, M.RezaSalehnamadi.A Prediction-BasedReplica Replacement Strategy in Data Grid. *Department of Computer Engineering, South TehranBranch, Islamic AZAD University, Tehran, Iran,J. Basic. Appl. Sci. Res*, 3(4)928-939, 2013.

[Ort09] - A.Ortiz. « Contrôle de la concurrence dans les grilles informatiques ». Application au projet ViSaGe. Thèse de doctorat, l'Université Toulouse, décembre 2009.

[PWCEKRT81] -G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G.Thiel, *"Locus: A Network Transparent High Reliability Distributed System,"* Proceedings of the Eighth Symposium on Operating Systems Principles, pp 169-177 ACM, December 1981.

[PWS99] - F. Pedonne, M. Wiesman and A. Shipper, « A Systematic Classification of Replicated Database Protocols based on Atomic Broadcast », in proceedings of the 3th European Research Seminar on Advances in distributed systems (ERSADS99). Madeira Island, Portugal, 1999.

[PWSKA00] - F. Pedone, M. Wiesmann, A. Schiper, B. Kemme, and G. Alonso. « Understandingreplication in databases and distributedsystems ». In ICDCS, pages 464–474, 2000.

[Sen03] - P.Sens. « Tolérance aux fautes et passage à l'échelle ». In Réunion DataGraal 30-31 Janvier 2003.Grenoble.

[SM12] - N.Saadat, A.MasoudRahmani. « PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids ».Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran (Future Generation Computer Systems 28 (2012) 666–681).