MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

Placement des répliques basé sur les propriétés de localité dans les grilles de données.

Etudiants : « **GHELMACI Noureddine** »

« **BEDANI Mohammed** »

Encadrant : « M. BOUHARAOUA Farouk »

Co-encadrante : « Mme. KAID SLIMANE Bouchra»

Année Universitaire 2018-2019

Dédicaces

A la mémoire de mon père

Bedani mohammed

Remerciements

Nous tenons à exprimer nos sincères remerciements et toute gratitude à notre encadreur Monsieur BOUHARAOUA Farouk, enseignant à l'Université Abd Elhamid Ibn Badis de Mostaganem, pour la confiance qu'il nous a manifestée tout au long de cette année. Malgré ses lourdes charges, il a toujours été là pour nous orienter et nous soutenir. Nous espérons avoir étiez à la hauteur de ses espérances.

Nous remercions aussi madame KAID SLIMANE Bouchra enseignante à l'Université Abd Elhamid Ibn Badis de Mostaganem, pour ses orientations, ses conseils précieux et ses remarques constructives.

ملخص

في ألنظمة الموزعة على نطاق واسع مثل شبكات البيانات ، يؤثر موقع النسخ)ألماكن التي يتم فيها وضع النسخ(على فعالية استراتيجيات النسخ. بالإضافة إلى ذلك، يلعب الموقع دو ًرا مه ًما في تحديد أماكن النسخ ، مما يساعد على تحسين الوقت الكلي للوصول لهذه النسخ وتوفرها في بيئة شبكة البيانات .عندما يوجد طلب الوصول لهذه البيانات، قد تكون هذه الطلبات والوصول إليها تخضع لخصائص الحواقع المختلفة)زمنية، جغرافية، مكانية(.

في هذا السياق، نقترح من خالل هذا العمل طريقة لوضع النسخ في شبكات البيانات التي تأخذ في االعتبار هذه الخصائص.

الكلمات المفتاحية: شبكة البيانات ، نسخ البيانات ، موضع البيانات ، عدد النسخ المتماثلة ، خصائص الحواقع.

Résumé

Dans les systèmes distribués à grande échelle tel que les grilles de données, l'emplacement des répliques (les endroits où les répliques sont placées) affectent l'efficacité des stratégies de réplication.

En plus, la localisation joue un rôle essentiel lors du placement des répliques, ce qui contribue à améliorer le temps d'accès global et la disponibilité des données dans l'environnement de grille de données. Quand les clients demandent l'accès aux données, les modèles de ces demandes, ou les modèles d'accès, peuvent présenter des propriétés de localité diverses (temporelle, géographique, spatiale).

Dans ce contexte, nous proposons à travers ce travail une approche de placement des répliques dans les grilles de données qui prend en considération ces propriétés de localité.

Mots-clés : Grille de données, réplication de données, placement de données, nombre de répliques, propriétés de localité.

Liste des tableaux

Liste des figures

Table des matières

Introduction Générale

Les demandes en puissance de calcul et en capacité de stockage sont énormes et dépassent largement les capacités informatiques d'un particulier, d'une institution et même d'un pays. Pour faire face à ces exigences, les systèmes à large échelle sont apparus vers la fin des années 90.

Parmi ces systèmes, la grille, telle que nous la considérons dans ce manuscrit est généralement définie comme une interconnexion des grappes d'ordinateurs géographiquement éloignés [1].

Ces architectures de grande taille remettent en cause un certain nombre de concepts, de méthodes, de techniques et d'outils informatiques [2].

Dans ce travail, nous nous intéressons particulièrement aux grilles de données qui permettent de connecter des milliers voire des millions de machines et de ressources de stockage distribuées à travers le monde. Ce type de grille est utilisé par les applications intensives qui génèrent des quantités importantes de données. Ces applications peuvent avoir besoin de données produites par d'autres applications géographiquement éloignées. Comme les données représentent la ressource la plus importante dans ce type de grilles, une gestion et un accès efficace à ce volume de données présentent des défis majeurs.

Une des techniques utilisée, pour faire face à ces défis est la réplication de données, qui permet en général de régler certains problèmes tels que les problèmes de la disponibilité des données à chaque instant. Elle consiste à répliquer les données dans les sites selon une disponibilité exigée par le système ou par l'utilisateur dans le but d'améliorer la disponibilité et réduire le temps de réponse selon certains critères.

Généralement, les gestionnaires des répliques sont interrogés pour déterminer l'emplacement de meilleures répliques en termes de coût d'accès. Ces répliques peuvent être distribuées selon certaines propriétés de localité.

Le travail que nous présentons dans ce mémoire rentre dans cette problématique. Il vise à proposer un algorithme pour déterminer les emplacements optimaux pour les nouvelles répliques et qui prend en considération les propriétés de localité.

Chapitre 1

Le simulateur OptorSim

1.1 Introduction

Les systèmes distribués sont naturellement largement étudiés en utilisant la modélisation mathématique, suivie généralement de simulations. Il existe des nombreux simulateurs permettent de simuler une grille de données. Mais ces simulateurs sont en général dédiés à la simulation d'un type d'application précis. Cependant une recherche étendue a été conduite dans le domaine de la simulation pour modéliser de tels systèmes et comprendre leur comportement.

Nous allons présenter brièvement quelques simulateurs suivis d'une présentation détaillée d'OptorSim

1.2 Présentation des simulateurs

1.2.1 3.2.1 Le simulateur GridSim

GridSim [14] est un système de modélisation et de simulation des ressources et des applications d'ordonnancement dans des environnements distribués et parallèles à grande échelle tels que les grilles et les réseaux pair à pair (P2P) .C'est un ensemble d'outils pour la simulation des grilles. Ces outils permettent la modélisation et la simulation des ressources hétérogènes d'une grille, des utilisateurs et des modèles d'applications.

Une architecture et une conception multicouche pour le développement de la plateforme de GridSim et de ses applications est montrée dans la figure 3.1 :

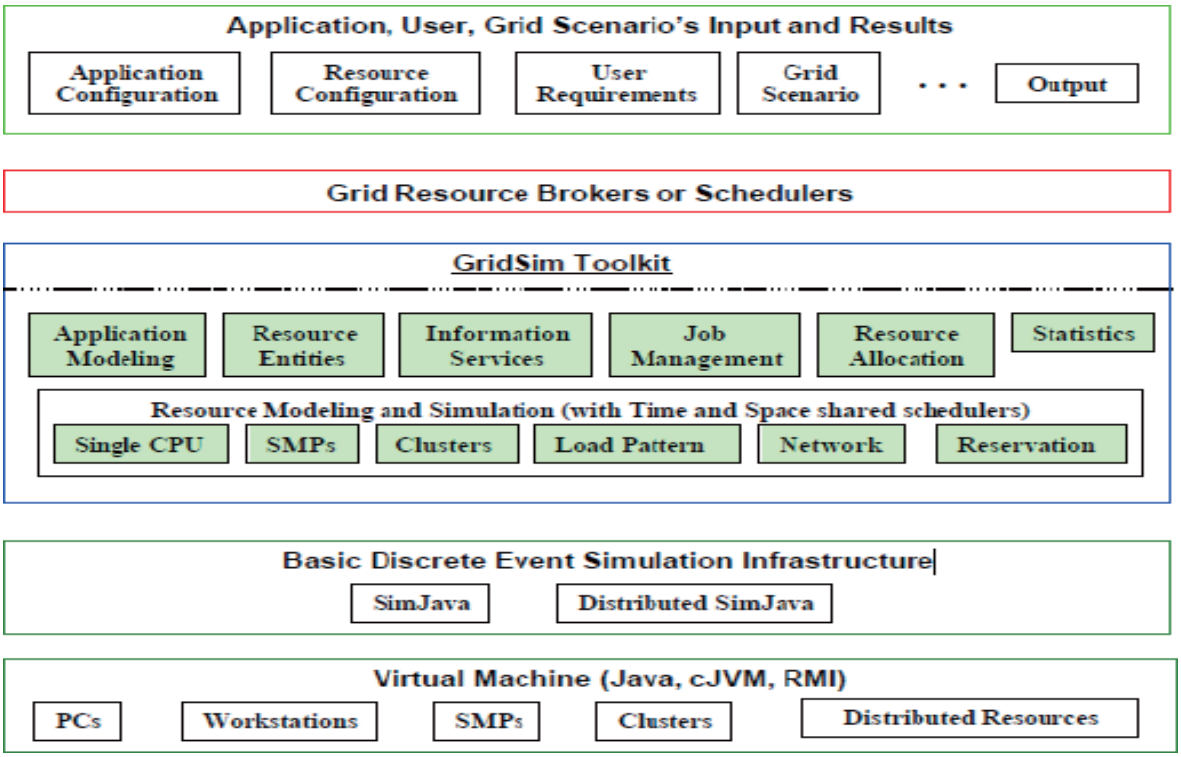

Figure 0.1- Architecture du simulateur GridSim

- La première couche est concernée par l'interface extensible et la machine d'exécution de Java, appelée JVM (Java Virtual Machine).
- La deuxième couche est concernée par une infrastructure de base d'évènement discret, tel que SimJava, établie en utilisant les interfaces fournies par la première couche. Une implémentation distribuée de SimJava est aussi disponible.
- La troisième couche est concernée par la modélisation et la simulation des entités de la grille telles que les ressources et les services d'informations.
- La quatrième couche est concernée par la simulation des ressources appelées ressources Broker (intermédiaires) et ressources d'ordonnancements de la grille.
- La dernière couche est consacrée à l'application et la ressource, modélisée avec différents scénarios en utilisant des services fournis par les deux couches inférieurs, pour évaluer l'ordonnancement et la politique de gestion de la ressource.

L'implémentation de GridSim dans Java est une contribution importante puisque Java fournit un ensemble riche d'outils qui augmentent la productivité de programmation, la portabilité d'application, et un environnement d'exécution extensible.

1.2.2 Le simulateur SimGrid

SimGrid [15] est développé à l'université de Californie à San Diego (UCSD) en 2000 par Henri Casanova et Martin Quinson. C'est un système de simulation des applications distribuée dans des environnements informatiques distribués dans le but de développer et d'évaluer des algorithmes d'ordonnancement. Logiciel développé en C, SimGrid V2 (deuxième version) [16] permet d'étudier des modèles et des topologies plus réalistes que la première version. Cette version intègre les modules suivants :

- Agent
- Endroit
- Tâche
- Chemin
- Conduite

Avec ces concepts, des algorithmes d'ordonnancement avec SimGrid devraient toujours être décrits en termes d'agents qui s'exécutent à des endroits et réagissent en envoyant, recevant et traitant des tâches de l'application simulée.

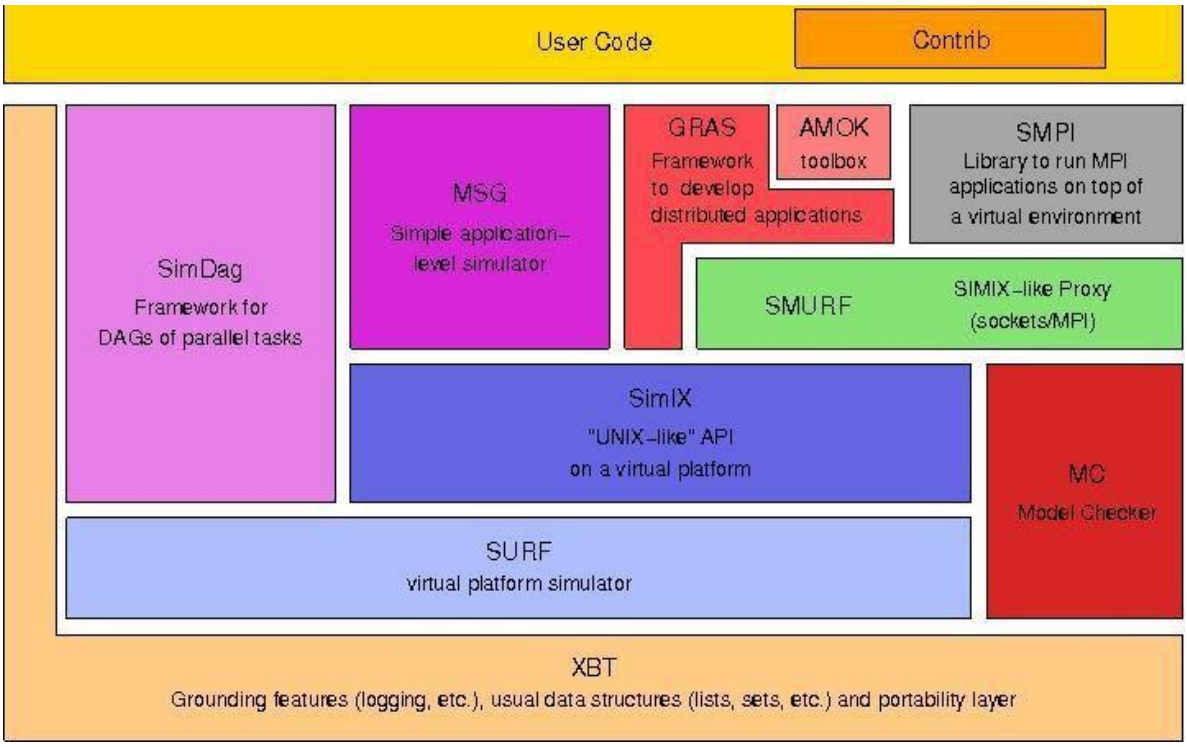

Figure 0.2- Architecture du simulateur SimGrid

Le SimGrid regorge en lui quelques modules importants (voir la figure 3.2) dont nous pouvons citer:

- SURF: noyau de simulation de SimGrid (produit les fonctionnalités nécessaire pour la plate-forme virtuelle)
- MSG: est l'API (interface) utilisée pour la simulation des applications distribuées SMPI: consiste à utiliser les applications MPI en faisant une recompilation de celle-ci
- Simdag: exprimer la simulation sous forme d'un graphe de tâche parallèle
- XBT: utilisé par les autres modules, implémentation des tâches de la gestion des mémoires et des structures très utiles comme des tableaux dynamiques ou des graphes

Le programme de SimGrid suit toujours les étapes suivantes :

- Définition du code de chaque agent
- Création des ressources

- Création et attribution des agents aux endroits
- La simulation peut être commencée par la fonction MSG_main.

SimGrid est limité à des systèmes d'ordonnancement et de temps partagé simples, il est difficile de simuler des applications et des programmes d'ordonnancements multiples, surtout dans des environnements tels que les grilles où beaucoup de ressources à grande échelle sont des machines à espace partagé et elles doivent être simulées.

1.2.3 Le simulateur Alea

Alea [17] est basée sur les caractéristiques des algorithmes de règles de priorité, un ensemble d'algorithmes ont été conçus pour les mettre en œuvre dans l'environnement de la grille. Architecture générale de ces algorithmes est illustrée à la figure 3.3, qui détermine la sélection des ressources et la répartition du tâches.

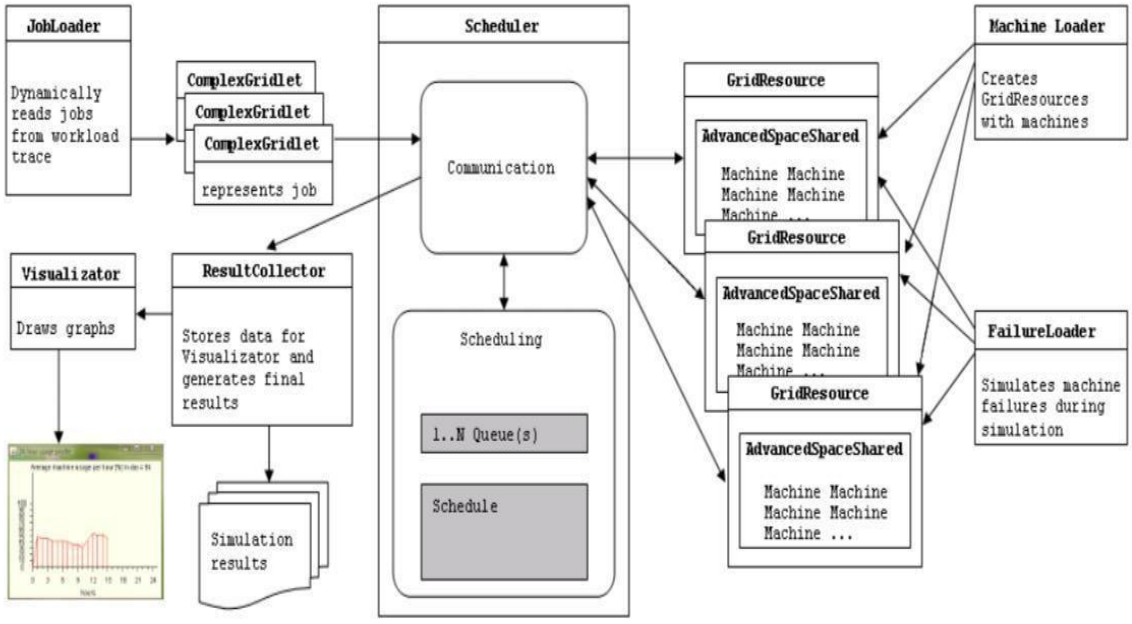

Figure 0.3- Architecture du simulateur Alea

Alea, version étendue de GridSim, est utilisée pour simuler le processus de planification dans un environnement informatique en grille. Ce dernier est constitué de plusieurs machines avec différents CPU avec différentes vitesses d'exécution et un nombre important de tâches à ordonnancer. Chaque tâche contient des fichiers différents avec un temps d'arrivée.

Le simulateur Alea est modulaire, composé d'entités indépendantes qui correspondent à la réalité. Il se compose de l'ordonnanceur centralisé, le travail sous-système de la mission, et les ressources de la grille. Ces entités communiquent entre eux par passage de messages. Actuellement, les utilisateurs de la grille ne sont pas directement simulés, mais un générateur de tâches attaché au système de soumission de tâches est utilisé pour simuler les arrivées de tâches.

Le système de soumission stocke les tâches avant et après leur exécution et communique avec l'ordonnanceur pour obtenir une stratégie d'ordonnancement, dont il se sert en outre de sélectionner une ressource pour exécuter une tâche.

Alea est un outil pour concevoir et tester les algorithmes d'ordonnancement pour certains scénarios typiques de la grille.

1.2.4 Le simulateur OptorSim

OptorSim [18] est un simulateur de grilles de données qui est développé dans le cadre du projet Européen DataGrid (EDG) [19] afin de tester différents algorithmes de réplication des données selon plusieurs scénarios. OptorSim a était développer suit aux manque d'environnement de simulation pour les applications nécessitant la manipulation de grands ensembles de données dans une grille.

Après cette brève description des simulateurs les plus célèbres en grilles de données, notre choix s'est porté sur le simulateur de grille de données OptorSim.

OptorSim a été développé à la base pour tester des stratégies de réplication dynamiques utilisées pour optimiser l'efficacité des grilles. Son langage de programmation « java » permet d'adopter l'approche orienté objet, efficace lorsqu'il s'agit de modéliser un réseau avec ses différents composants, et d'exécuter de nombreux threads simultanément. Son architecture est

simple et permet de modifier et d'intégrer de nouveaux codes (modules). Toutes ces raisons ont motivé le choix pour ce simulateur.

Dans ce qui suit nous allons présenter en détail le simulateur OptorSim.

1.3 OptorSim

C'est un simulateur a pour but d'étudier la stabilité et le comportement transitoire des méthodes d'optimisation des répliques. Son principe est de modélise les interactions des composants individuels d'une grille de données (DataGrid).

La simulation a été construite en supposant que la grille se compose de plusieurs sites (voir figure 3.4), dont chacun peut fournir les ressources informatiques, de données et de stockages pour les jobs soumis. Les éléments de calcul (CEs) exécutent les jobs, qui emploient les données stockées sur des éléments de stockage (SEs) et un courtier de ressource (RB) qui commande l'ordonnancement des jobs aux CEs.

- Resource Broker : (RB) Il est responsable de programmer un job sur un ou plusieurs CE. Les RBs peuvent agir les uns sur les autres avec le système de gestion de réplication pour optimiser l'ordonnancement des jobs et leurs données.
- Computing Element : (CE) Il fournit à des utilisateurs de grille des cycles d'unité centrale de traitement pour l'exécution des jobs. Chaque élément de calcul est situé dans un "site" particulier sur la grille. Les éléments de calcul ont des interfaces bien définies pour programmer et surveiller les jobs.
- Storage Element : (SE) Il fournit à des utilisateurs de grille la capacité de stockage. La quantité de capacité de stockage disponible pour les jobs de la grille. Les éléments de stockage ont des interfaces bien définies pour stocker et rechercher des dossiers.

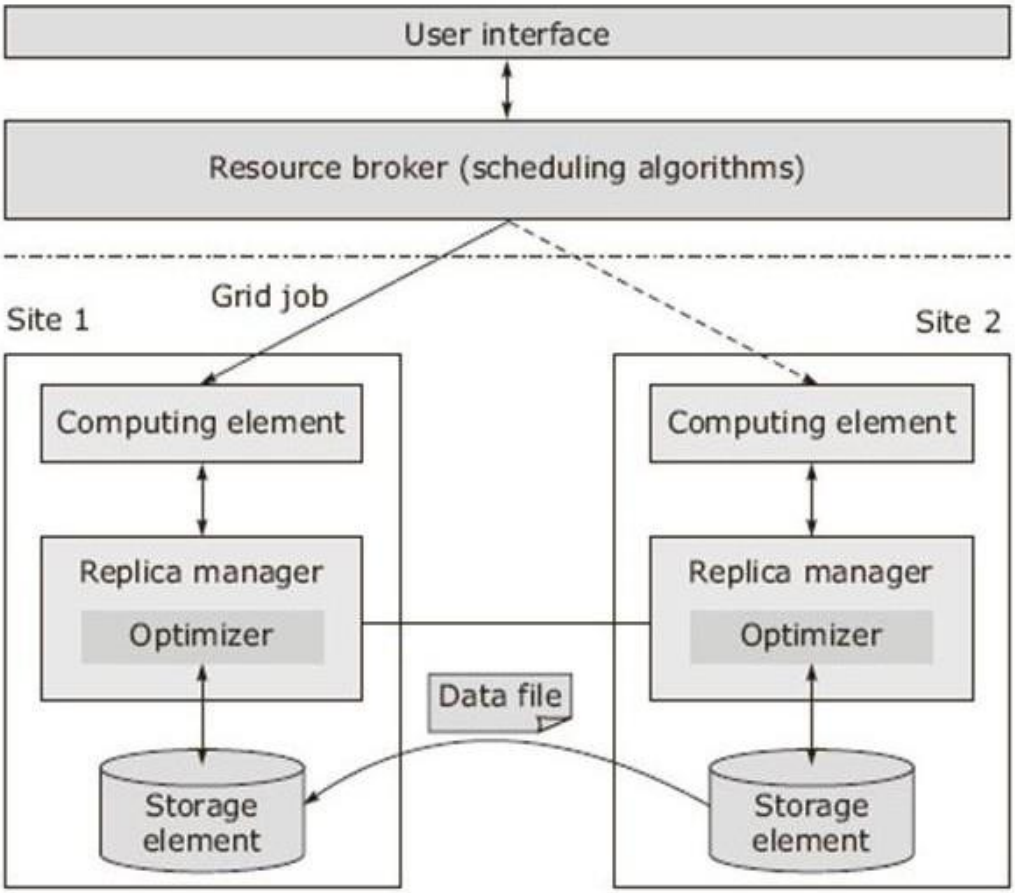

Figure 0.4- Architecture du simulateur OptorSim

Les tâches s'exécutent au niveau des éléments de calcul et utilisent les données stockées dans les éléments de stockage.il permet la description de la topologie du réseau en énumérant les liens et la largeur de bande reliant les sites. Avant le début de la simulation l'utilisateur peut configurer la topologie de la grille et définir les jobs soumis avec la liste des fichiers que nécessite chaque job. Les algorithmes d'ordonnancement et d'optimisation des répliques se trouvent dans un document de paramètres. D'autres paramètres peuvent être assignés, comme le mode de transfert des fichiers, la distribution initiale des données.

1.3.1 Les algorithmes d'optimisation

Les algorithmes d'optimisation de réplication sont le noyau de l'optimiseur de réplication.

La stratégie employée pour répliquer et quel fichier devrait être effacé, différencie les algorithmes d'optimisation. Nous présentons brièvement les algorithmes simples qui ont été mis en application dans OptorSim.

a) **Simple Optimiser (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.

b) **Least Recently Used (LRU)** :

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 récemment utilisée.

c) **Least Frequently Used (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.

d) **Le modèle économique** :

Lorsqu'un job a besoin d'une donnée qui n'existe pas sur le site sur lequel il s'exécute, on estime deux coûts :

- Le cout d'une exécution à distance du job. L'exécution se fait sur le site contenant la donnée puis transfert 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. Sur OptorSim, il en existe deux implémentations de ce modèle : Zipf-based economic model (utilisant une fonction de prédiction Zipf) et EconomicBinomial (basé sur le modèle binomial).

Ces stratégies ont été associées à quatre modes d'ordonnancement des jobs :

- **Random** : Les jobs sont distribués aléatoirement aux CEs.
- **Queue Length** : les jobs sont ordonnancés aux CEs avec le moins de tâches en file d'attente. Si deux CEs ont le même nombre de tâches en file d'attente, l'un d'entre eux est choisi aléatoirement.
- **File Access Cost** : les jobs ordonnancés aux CEs dont le coût d'accès aux fichiers requis par le job (en terme de latence du réseau) est moindre. Si deux CEs ont le même coût d'accès l'un d'entre eux est choisi aléatoirement.
- **File Access Cost + Job Queue Access Cost** : L'ordonnancement est établi en combinant les stratégies Queue Lenght et File Access Cost.

1.3.2 Les fichiers de configuration

Il existe plusieurs fichiers de configuration utilisés pour contrôler les différents paramètres d'entrée d'OptorSim [20].

- Fichier de configuration de grille "Grid Configuration File": décrit la topologie de la grille ainsi que les ressources que contient chaque site.
- Fichier de configuration des taches "Job Configuration File": permet de définir les jobs, et les fichiers que nécessite l'exécution de chaque job.
- Fichier de configuration de la bande passante "Bandwidth Configuration File" : est le fichier de configuration du trafic sur le réseau.
- Fichier de configuration de paramètre "Parameters file": permet d'initialiser les paramètres d'entrée avant la simulation

1.3.2.1 Fichier de configuration de paramètre

Nom de ce fichier par défaut est ‘ parameters.conf’ (voir Figure 3.5)

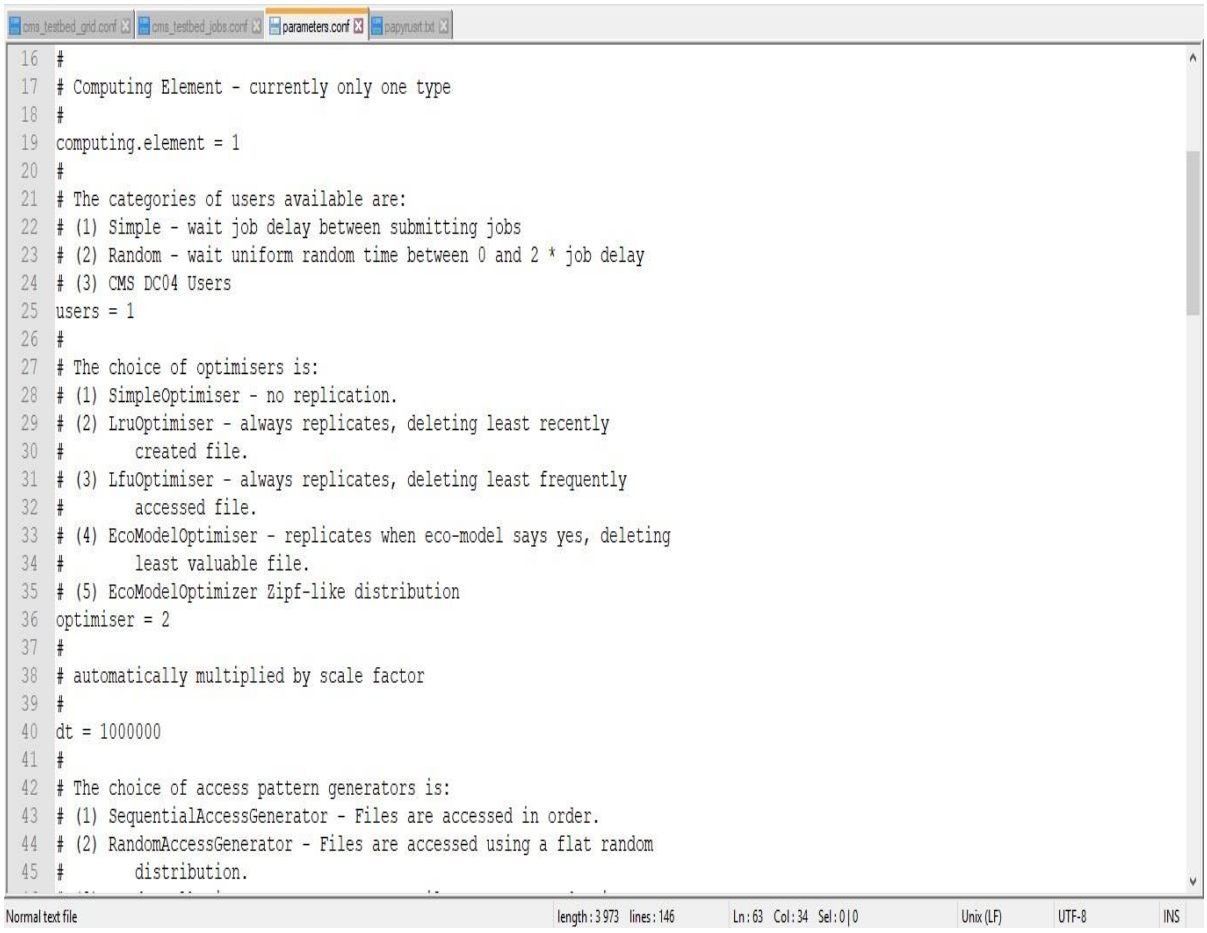

Figure 0.5- Contenu du fichier parameters.conf

Dans ce fichier, on spécifie :

- Les noms des fichiers de configurations et leur chemin d’accès (autres fichiers que parameters.conf).
- Le nombre de jobs (requêtes) à exécuter sur la grille.
- L’algorithme d’ordonnancement des jobs (utilisé par le Resource Broker).

- L'algorithme de réplication des données (utilisé par le gestionnaire de répliques).
- La distribution initiale des fichiers originaux (données originales non répliquées).

1.3.2.2 Fichier de configuration de grille

Nom de ce fichier par défaut est 'cms_testbed_grid.conf' (voir Figure 3.6)

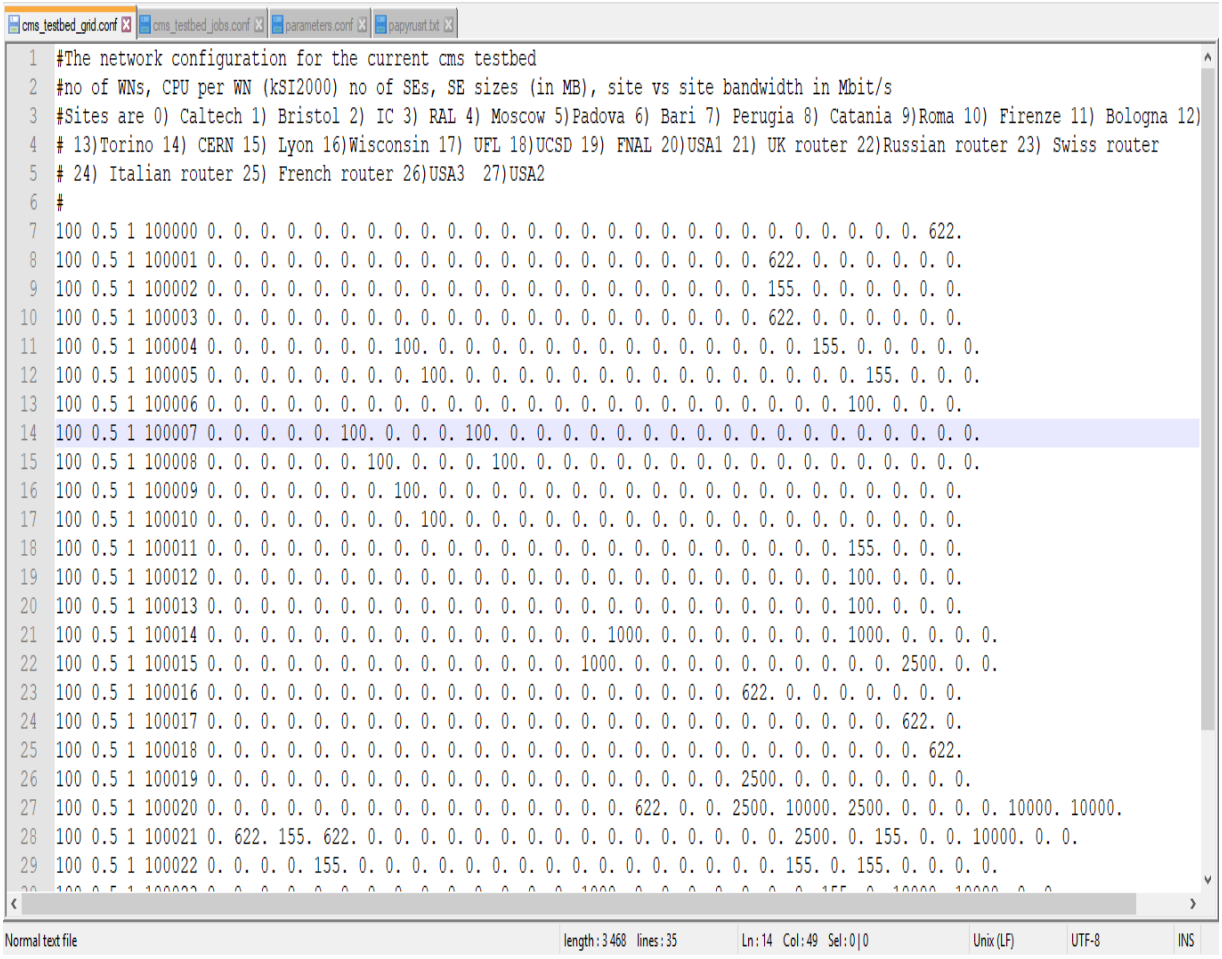

Figure 0.6- Contenu du fichier cms_testbed_grid.conf

Dans ce fichier:

- La première colonne désigne le nombre de worker nodes dans le site correspondant.

- La deuxième colonne désigne le nombre de ressources de stockage (SE) dans le site correspondant.

- La troisième colonne désigne la taille des ressources de stockage en Mo.

Le reste des colonnes de la matrice est une matrice carrée représentant la bande passante en Mb/s entre deux sites

1.3.2.3 Fichier de configuration des taches

Nom de ce fichier par défaut est 'cms_testbed_jobs.conf' (voir Figure 3.7)

cms_testbed_grid.conf | cms_testbed_jobs.conf | parameters.conf | papyrusrt.txt

```
# Job Table
# A job name and a list of files needed.
#
\begin{jobtable}
jpsijob jpsi0 jpsi1 jpsi2 jpsi3 jpsi4 jpsi5 jpsi6 jpsi7 jpsi8 jpsi9 jpsi110 jpsi001 jpsi301 jpsi401 jpsi501 jpsi601 jpsi701 jpsi
\end
#
# CE Schedule Table
# CE site id, jobs it will run
#
\begin{cescheduletable}
3 jpsijob
2 jpsijob
9 jpsijob
10 jpsijob
\end
#
# The probability each job runs
#
\begin{jobselectionprobability}
jpsijob 1
\end
#
#
\begin{jobexecutiontime}
jpsijob 0 5
\end
#
#
```

Normal text file | length : 1 084 lines : 66 | Ln : 48 Col : 11 Sel : 0 | 0 | Unix (LF) | UTF-8 | INS

Figure 0.7- Contenu du fichier cms_testbed_jobs.conf

Ce fichier décrit :

- L'ensemble des fichiers distribués dans la grille. Chaque fichier est défini par son nom, son identifiant et sa taille en Mo.
- L'ensemble des requêtes ou jobs lancés dans la grille. Pour chaque job, on définit l'ensemble des fichiers dont il a besoin pour son exécution, son temps d'exécution, sa probabilité d'exécution.

1.3.3 Principe de fonctionnement

Pour exécuter les jobs spécifiés dans le fichier de configuration, le Resource Broker utilise l'algorithme d'ordonnancement spécifié dans le fichier des paramètres. Une fois un job affecté à un site, il a besoin de retrouver les données nécessaire à son exécution. Trois cas se présentent :

- Données présentent sur le site d'exécution : le job est alors exécuté immédiatement.
- Réplication des données sur le site d'exécution : Si les données nécessaires ou une partie d'elles n'existent pas sur le site du job, alors le gestionnaire de répliques utilise l'algorithme de réplication spécifié dans le fichier de paramètres pour créer et placer les répliques nécessaires.
- Exécution à distance : dans certain algorithmes de réplication, le gestionnaire des répliques peut décider de faire une exécution à distance au lieu de répliquer les données (ex: modèle économique).

En sortie, OptorSim donne l'évaluation des performances de la grille pour la configuration spécifiée. Les paramètres d'évaluation sont les suivants :

- Le temps d'exécution des jobs : C'est le temps moyen d'exécution d'un job, c'est à dire le temps total d'exécution de tous les jobs divisé par le nombre de jobs. Il inclut le temps propre à l'exécution plus le temps de communication dans le réseau (transfert de données) et le temps passé dans les files d'attentes. La minimisation de ce paramètre est importante si on vise à améliorer les performances de la grille.
- Le nombre de répliques : C'est le nombre de répliques créé durant l'exécution des jobs. Plus ce nombre est important plus il influe négativement sur le temps d'exécution des jobs

(+temps de réplication et transfert de données). Ce nombre peut être amélioré par un placement stratégique des répliques.

- Le nombre d'accès locaux : C'est le nombre de fois où les données nécessaires se trouvent sur le site d'exécution du job soit à l'origine soit après réplication des données vers ce site. Plus ce nombre est important plus le temps d'exécutions des jobs est meilleur (-temps de transfert). Le nombre d'accès distants : C'est le nombre de fois où le job fait un accès distant pour retrouver les données nécessaires.

La consommation en ressources de stockage : Ce paramètre est défini par le pourcentage de l'espace de stockage occupé par rapport à l'espace disponible. Ce paramètre est affecté par le nombre de répliques créées. Il est meilleur quand le nombre de réplications est minimal.

1.4 Conclusion

Dans ce chapitre nous avons présenté une brève description des simulateurs des grilles de données, ainsi qu'une présentation détaillée du simulateur OptorSim notre choix pour simuler notre approche s'est porté sur ce dernier car il est le plus approprié pour notre approche.

Chapitre 2

Conception et Implémentation

2.1 Introduction

Les grilles de données exigent la gestion d'un nombre massif de données afin de les rendre accessibles aux clients qui les réclament. Les stratégies de réplication des données représentent un facteur important vis-à-vis des performances des services proposés par une grille. Ces stratégies doivent répondre à des questions essentielles telle que : Où la réplique doit être placée ?

Placer les nouvelles répliques sur le site (emplacement) approprié peut favoriser la réduction de la consommation de bande passante du réseau et réduit donc le temps de réponse.

L'idée principale est de conserver les données à proximité de l'utilisateur afin de rendre l'accès efficace et rapide.

La stratégie de réplication est guidée par des facteurs tels que la demande des données, les conditions du réseau, le coût de transfert et le coût de stockage. Afin de maximiser le potentiel des gains de la réplication de fichiers, une stratégie de placement des répliques est très importante. Un service de placement des répliques est un élément de l'architecture des grilles de données qui décide où une réplique de fichier doit être placée dans le système.

Dans ce chapitre nous allons présenter nos stratégies de placement des répliques issues des stratégies de réplications de données prédéfinies du simulateur OptorSim en prenant en compte les propriétés de localité vus précédemment (temporelle, géographique et spatial). En effet après avoir effectué plusieurs simulations à l'aide d'OptorSim, nous avons pu définir nos

stratégies de placement, qui prennent en considération les propriétés de localité, en se basant sur celles prédéfinies dans OptorSim.

Dans ce qui suit nous commencerons par la présentation de nos stratégies suivie par leur implémentation et finir par la présentation des résultats obtenus des expérimentations qui nous permet de vérifier l'utilité de notre approche dans le choix de la stratégie la plus adéquate au modèle d'accès.

2.2 Conception

Un des problèmes lié à la réplication est le placement des différentes répliques. Dans ce contexte notre contribution consiste à proposer une nouvelle stratégie de placement basé sur les propriétés de localité en répliquant les donnés sur les sites ayant le plus grand nombre de voisins.

Nous proposons une nouvelle stratégie de placement qui se base sur les stratégies prédéfinies d'OptorSim, à savoir LRU, LFU, EconomicBinomial et Zipf-based economic model et qui en même temps prennent en compte les propriétés de localité temporelle, géographique et spatial. En effet, les pseudos codes présentés dans ce qui suit représentent les algorithmes de notre nouvelle stratégie.

2.2.1 Localité-LRU

```
SitePV= site qui a le plus grand nombre de voisin
Chercher fichier
Si trouvé localement alors exécuter
Sinon
    Si trouvé sur SitePV alors exécuté
    Sinon
        Si manque d'espace dans SitePV alors
            Supprimer la réplique la moins récemment utilisée
        Finsi
       Répliquer
       Exécuter.
    Finsi
Finsi
```

2.2.2 Localité-LFU

```
SitePV= site qui a le plus grand nombre de voisin
Chercher fichier
Si trouvé localement alors exécuter
Sinon
    Si trouvé sur SitePV alors exécuté
    Sinon
        Si manque d'espace dans SitePV alors
            Supprimer la réplique la moins fréquemment utilisée
        Finsi
      Répliquer
      Exécuter.
    Finsi
Finsi
```

2.2.3 Localité-EconomicBinomial

```
SitePV= site qui a le plus grand nombre de voisin
Chercher fichier
Si trouvé localement alors exécuter
Sinon
    Si trouvé sur SitePV alors exécuté
    Sinon
        Calcul Cout=min (cout en local SitePV, cout à distance)
        Utiliser le modèle binomial
    Finsi
      Exécuter.
    Finsi
```

2.2.4 Localité-Zipf-based economicmodel

```
SitePV= site qui a le plus grand nombre de voisin
Chercher fichier
Si trouvé localement alors exécuter
Sinon
    Si trouvé sur SitePV alors exécuté
    Sinon
        Calcul Cout=min (cout en local SitePV, cout à distance)
        Utiliser la fonction de prédiction Zipf
    Finsi
      Exécuter.
Finsi
```

2.3 Implémentation

Puisque OptorSim est écrit en JAVA, cela nous a permis de le modifier pour intégrer nos méthodes.

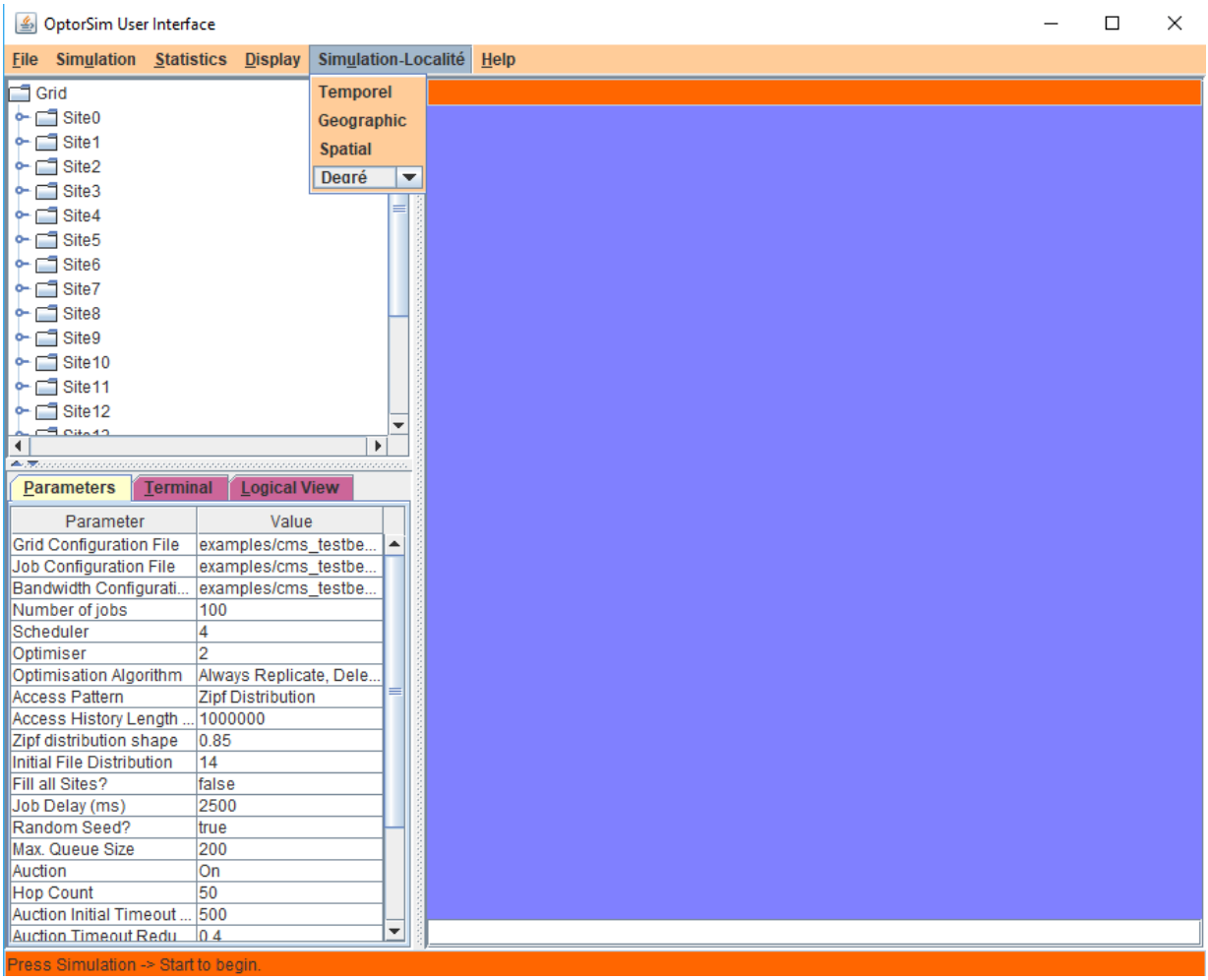

Figure 2.1- Fenêtre principale d'OptorSim

La figure 4.1 représente la fenêtre principale du simulateur OptorSim modifié. Dans le menu nous trouverons l'onglet « Simulation–Localité » composé des sous menus des trois propriétés de localité : temporel, géographique et spatiale avec leur degré de localité.

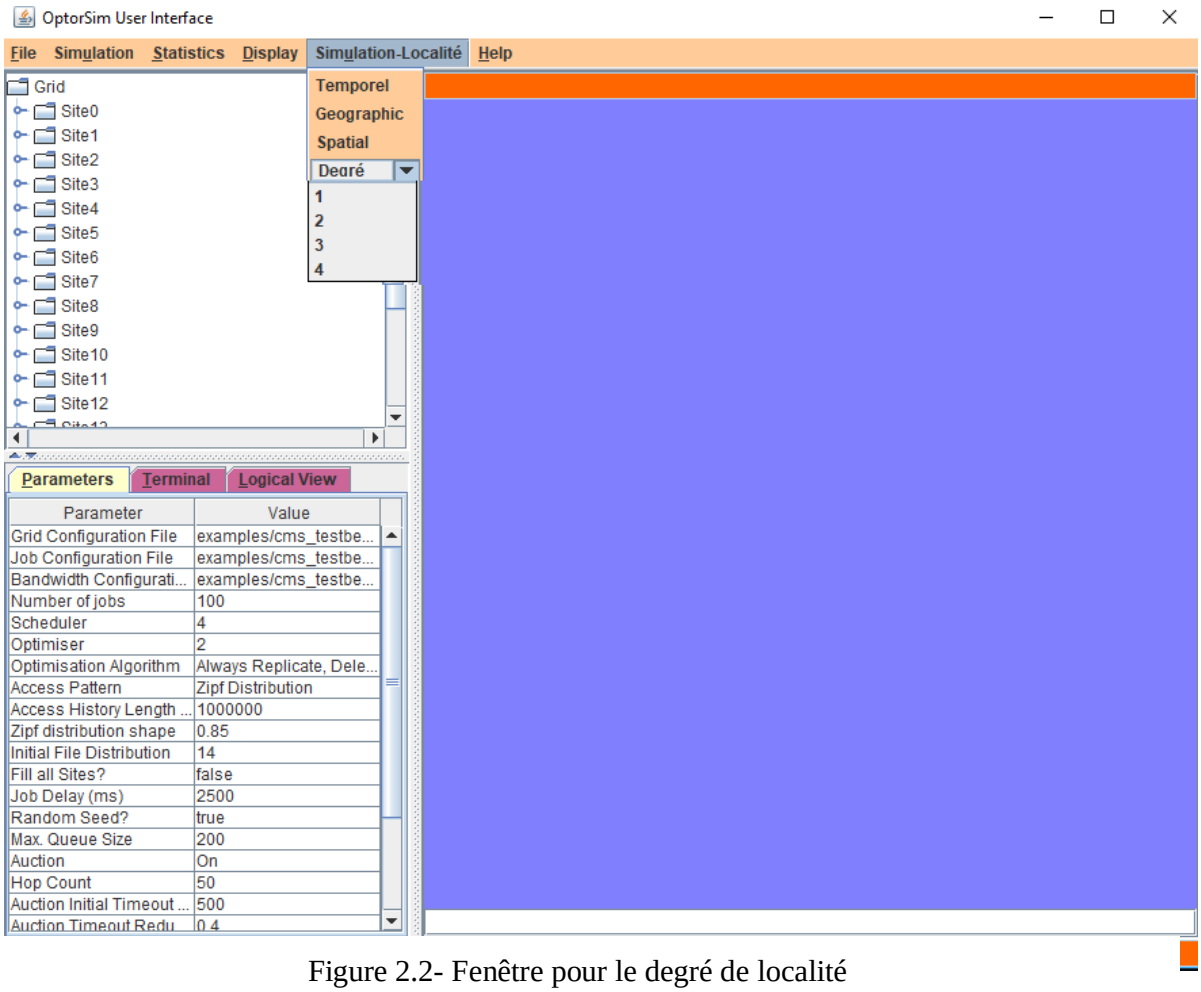

Figure 2.2- Fenêtre pour le degré de localité

La figure 4.2 nous permet de déterminer le degré de localité de chaque propriété de localité.

2.4 Expérimentations et Evaluation

Afin d'évaluer les performances de notre proposition nous avons mené plusieurs expériences.

2.4.1 Environnement de travail

Toutes nos expérimentations ont été réalisées sur une machine fonctionnant sous le système Windows 10 64 bits avec les caractéristiques suivantes :

- Processeur : Intel Core I3-3110 M CPU 2.40GHz 2.40GHz.
- RAM : 6 Go.

L'implémentation de nos algorithmes ont été développés en langage Java en utilisant l'environnement de programmation "Eclipse", afin de les intégrer dans le simulateur OptorSim.

Nous avons effectué 2 types d'expérimentions :

Le 1er type d'expérimentions est effectué pour pouvoir comparer notre approche à celles existantes dans OptorSim (LRU, LFU, EconomicBinomial, Zipf-based economic model). Et ensuite déterminer pour chaque propriété de localité la stratégie la plus appropriée.

Le 2eme type d'expérimentions est effectué pour étudier l'influence des degrés de localité sur les performances.

Ces expérimentations ont été effectuées en utilisant la version modifiée d'OptorSim qui nous a permis d'évaluer nos stratégies proposés (temporelle, géographique et spatiale).

2.4.2 Paramètres de simulation

Les expérimentions se sont effectuées sur une grille de 28 sites, comme illustrée dans la figure 4.3, selon les paramètres de simulation présentés dans le tableau 4.1.

Tableau 4.1 : paramètres de simulation

Les sites demandeurs	13
Distribution initiales des fichiers	14
Nombre de fichiers demandés	7
Nombre de jobs	100
Taille des fichiers	10 Go
La bande passante	100 Mo/s

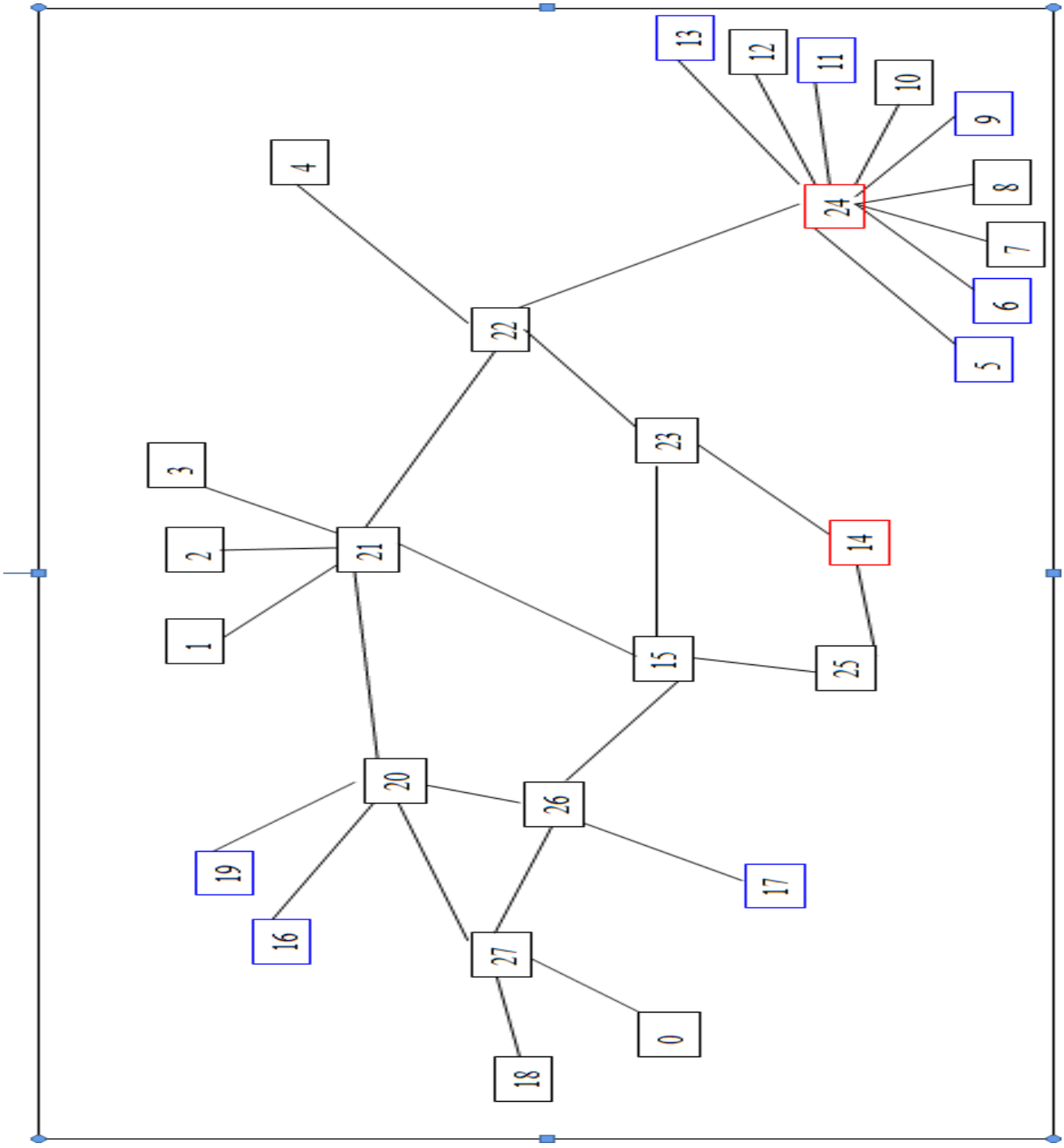

Figure 2.3- Exemple de la topologie de la grille utilisée

Nous désignons les sites en couleur rouge les emplacements des fichiers demandés, et les sites en couleur bleu les clients demandeurs.

2.4.3 La localité temporelle

Tableau 4.2 – Les temps de réponses moyens par stratégie de réplication (temporel)

	Moyenne des temps de réponses (seconde).		Amélioration (%)
	Sans localité	Avec localité	
LRU	531	323	39.18%
LFU	546	482	11.73%
EconomicBinomial	495	297	40%
Zipf-based economic model	1073	256	76.15%

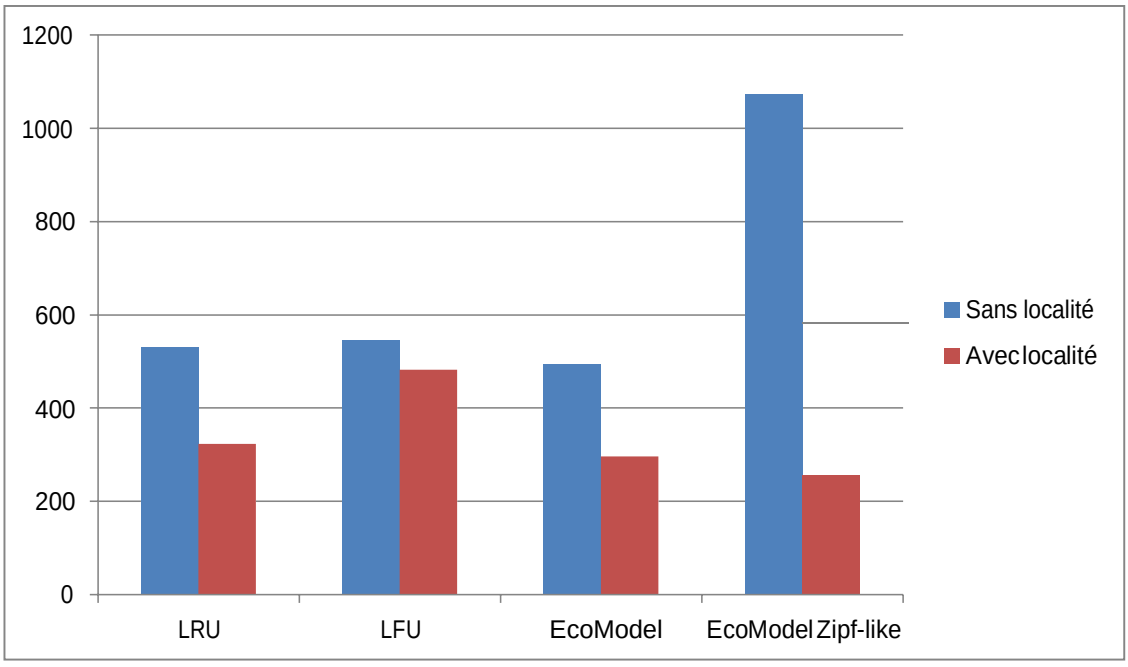

Figure 2.5- Les temps de réponses moyens par stratégie de réplication

Les résultats obtenus dans le tableau 4.2 et modélisés par le graphe de la figure 4.4 représentent les temps de réponses moyens par stratégie de réplication (sans et avec localité temporelle). Elle montre que notre approche améliore nettement les performances. Les résultats obtenus dans le Tableau 4.2 montrent que le temps de réponse moyen en prenant en compte la localité vaut 48.66% de moins que ceux sans prise en compte de la localité. Et nous remarquons aussi que la stratégie modifiée Zipf-based economic model est la plus appropriée pour la propriété de localité temporelle.

2.4.4 La localité géographique

Tableau 4.3 – Les temps de réponses moyens par stratégie de réplication (géographique)

	Sans localité	Avec localité
LRU	656	656
LFU	656	656
EconomicBinomial	732	732
Zipf-based economic model	732	732

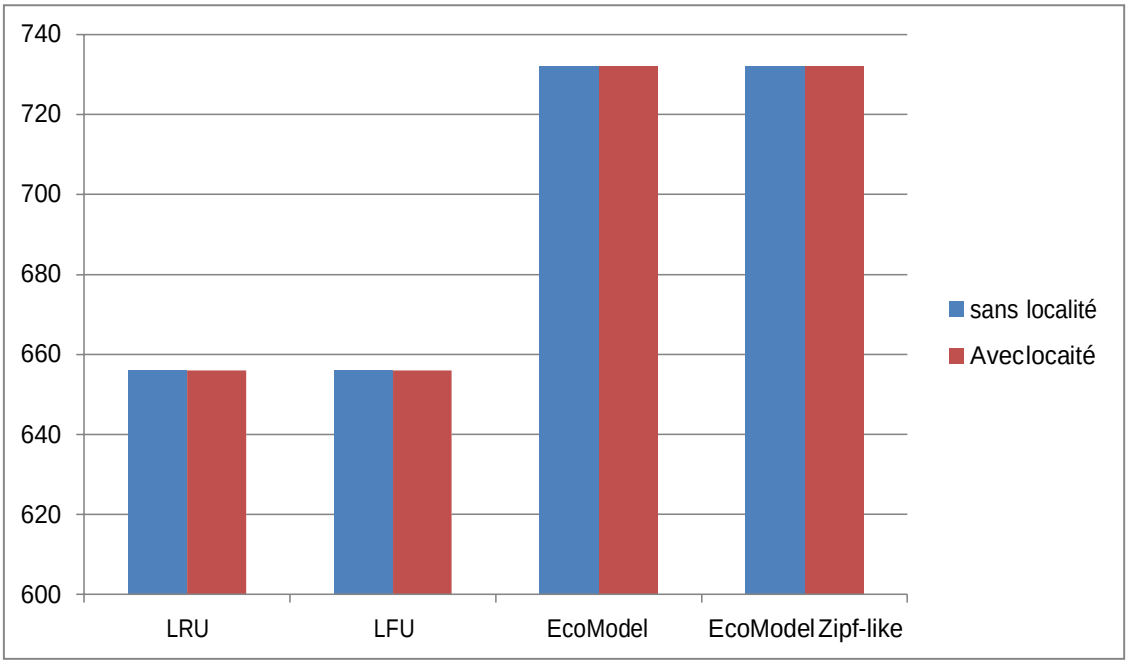

Figure 2.6 -Les temps de réponses moyens par stratégie de réplication (géographique)

Les résultats obtenus dans le tableau 4.3 et modélisés par le graphe de la figure 4.5 représentent les temps de réponses moyens par stratégie de réplication (sans et avec localité géographique).

Nous remarquons que généralement la localité géographique n'a pratiquement aucune influence sur les performances.

2.4.5 La localité spatiale

Tableau 4.4 – Les temps de réponses moyens par stratégie de réplication (spatiale)

	Moyenne des temps de réponses (seconde).		Amélioration (%)
	Sans localité	Avec localité	
LRU	1052	1064	-1.14%
LFU	1064	1066	-0.19%
EconomicBinomial	1325	1305	1.5%
Zipf-based economic model	1373	1333	2.91%

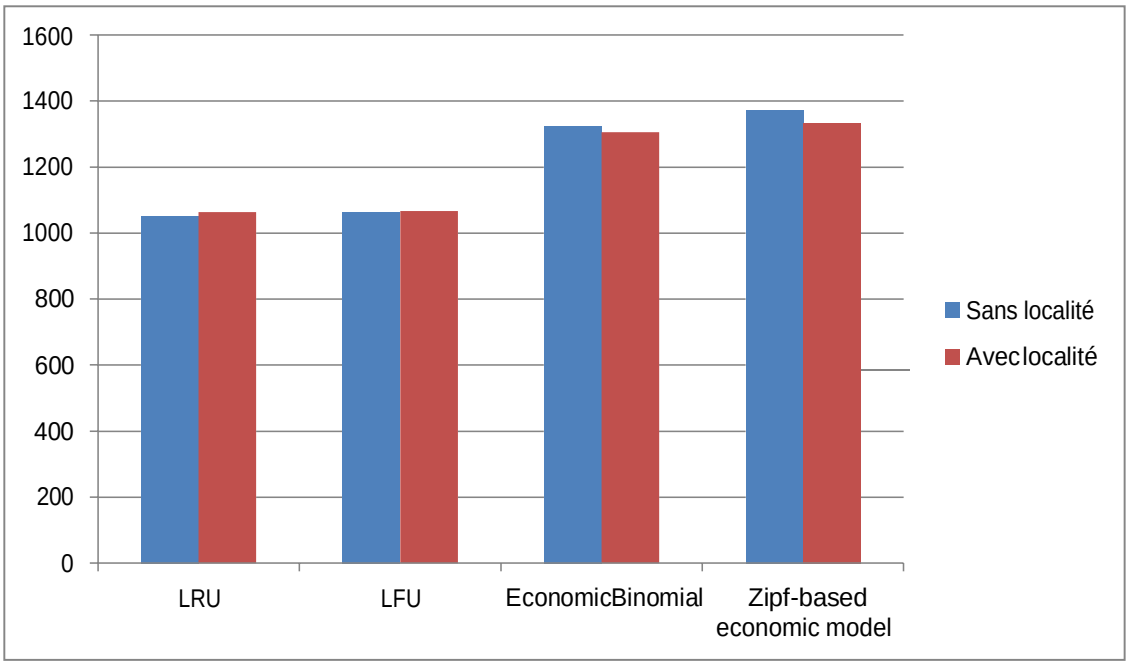

Figure 2.7- Les temps de réponses moyens par stratégie de réplication (spatiale)

Les résultats obtenus dans le tableau 4.4 et modélisés par le graphe de la figure 4.6 représentent les temps de réponses moyens par stratégie de réplication (sans et avec localité). Elle

montre que notre approche améliore les performances des deux stratégies EconomicBinomial (1.5%) et Zipf-based economic model (2.91%).

Mais notre approche n'a aucune amélioration pour les deux stratégies LRU et LFU. Et nous remarquons aussi que la stratégie LRU est la plus appropriée pour la propriété de localité spatiale.

2.5 Influence du degré de localité sur les performances

Apres avoir déterminer pour chaque propriété de localité la stratégie la plus appropriée. Nous avons étudié l'influence du degré de localité sur les performances.

Nous désignons par degré de localité le niveau de voisinage (le voisin le plus proche est de niveau 1).

Degré de localité 1 :

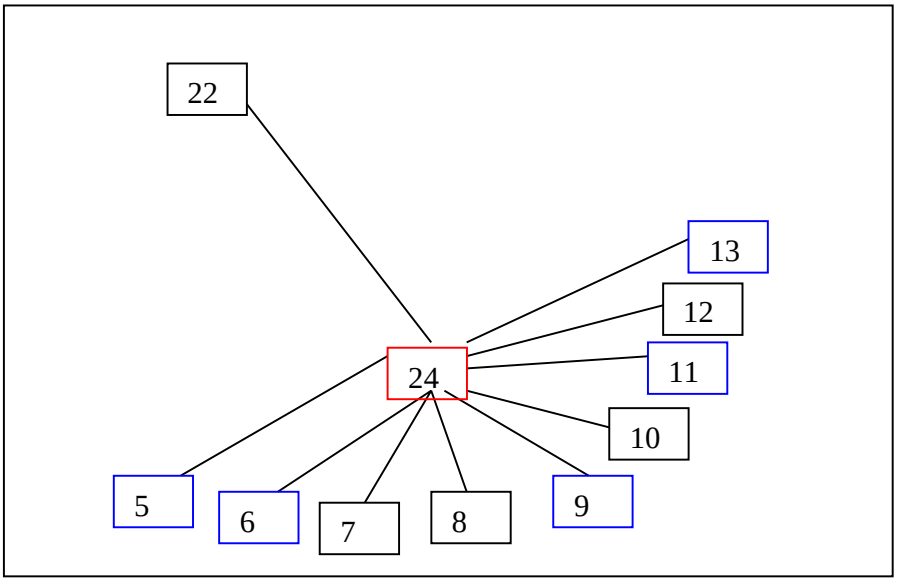

Figure 2.8- Exemple du degré 1 de localité

Degré de localité 2 :

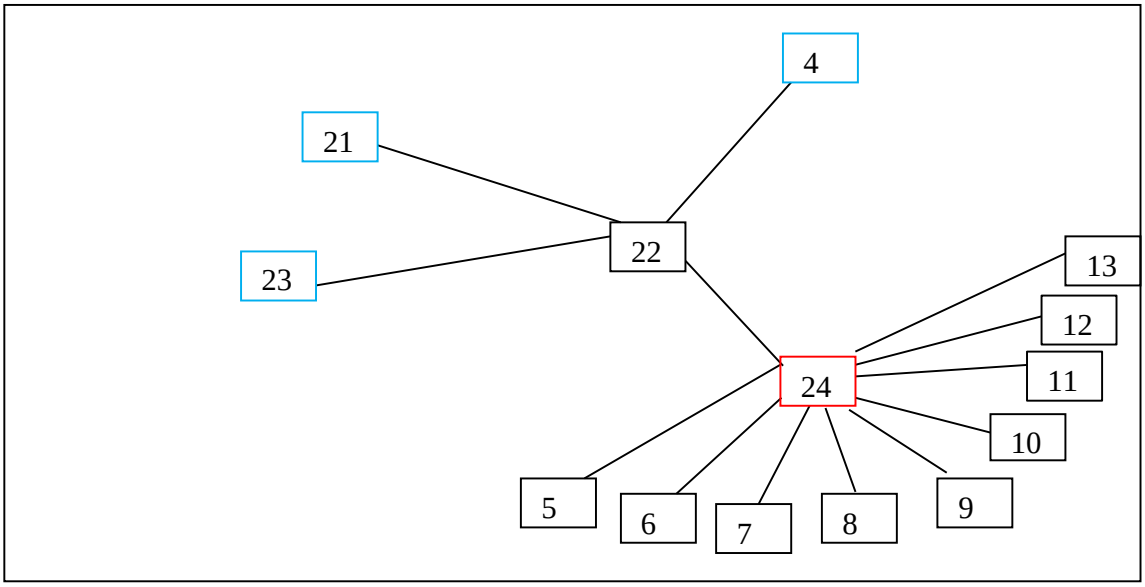

Figure 2.9- Exemple du degré 2 de localité

Degré de localité 3 :

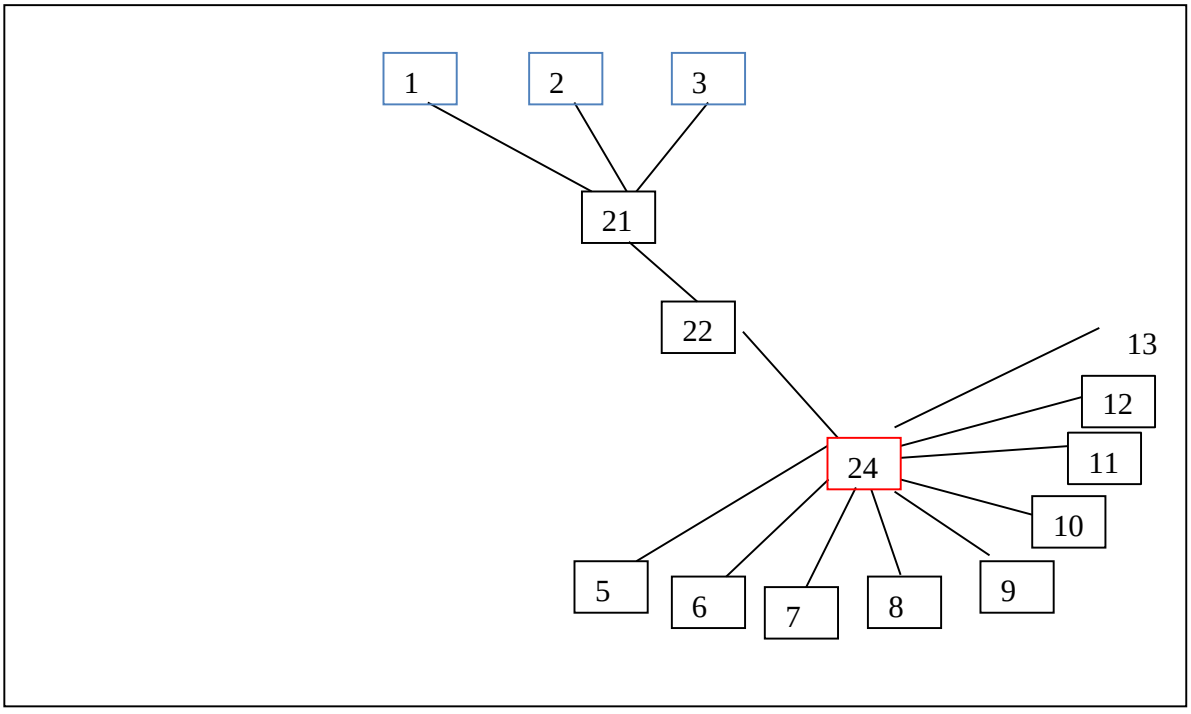

Figure 2.10- Exemple du degré 3 de localité

Dans la figure 4.7 les sites en bleu désignent les clients de degré de localité 1 par rapport au site 24 (fichier demandé).

Dans la figure 4.8 les sites en bleu désignent les clients de degré de localité 2 par rapport au site 24 (fichier demandé).

Dans la figure 4.9 les sites en bleu désignent les clients de degré de localité 3 par rapport au site 24 (fichier demandé).

2.5.1 La localité temporelle

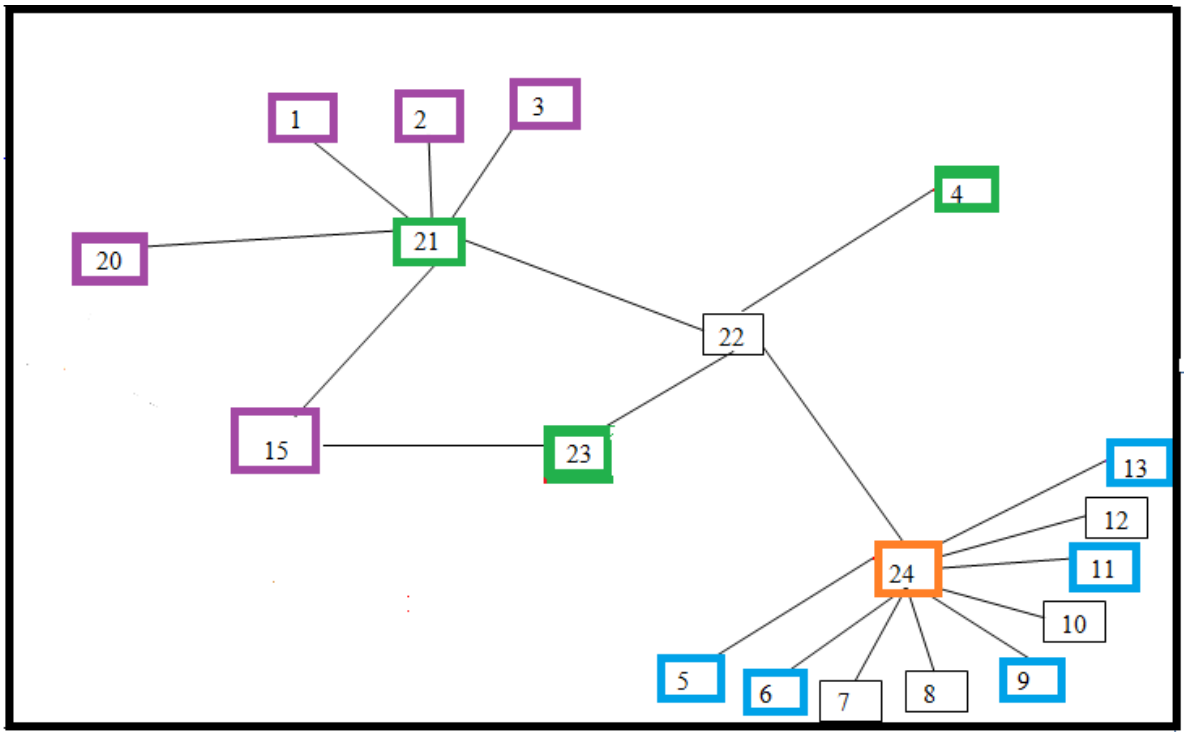

Figure 4.10 -Les différents degrés de localité (Temporel)

Dans la figure 4.10 les sites en couleur bleu désignent les clients de degré de localité 1 par rapport au site 24 (fichier demandé).Les sites en couleur vert désignent les clients de degré de

localité 2 par rapport au site 24 (fichier demandé).Les sites en mauve désignent les clients de degré de localité 3 par rapport au site 24 (fichier demandé).

Les différents résultats sont représentés dans le tableau suivant :

Tableau 4.5 – Les temps de réponses moyens (Temporel-Degrés)

	Moyenne des temps de réponses (seconde).		
Degrés	1	2	3
Localité-LRU	709	720	741
Localité-LFU	798	801	811
Localité-EconomicBinomial	837	840	852
Localité-Zipf-based economic model	840	847	857

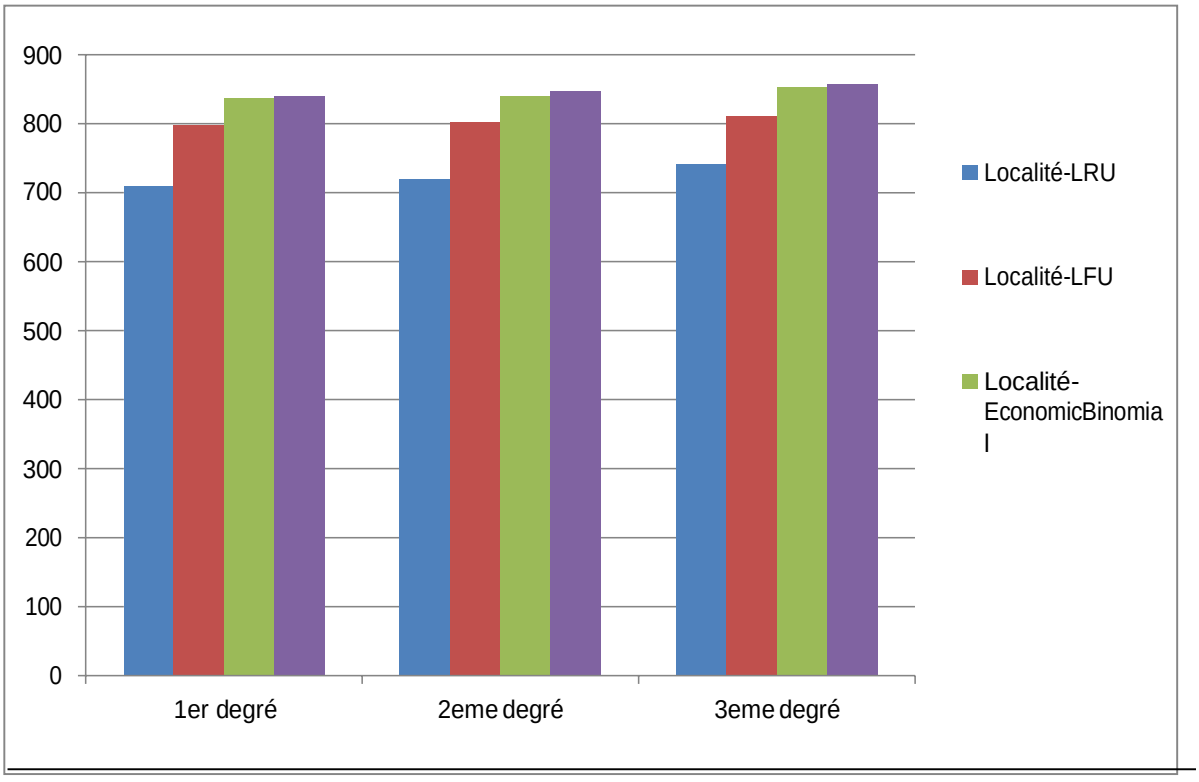

Figure 2.11- Temps moyen d'exécution (Temporel-Degrés)

Ces graphes montrent que dans le modèle de localité temporelle et pour les trois degrés la stratégie Localité-LRU est la plus performante en termes de temps d'exécution.

2.5.2 La localité géographique

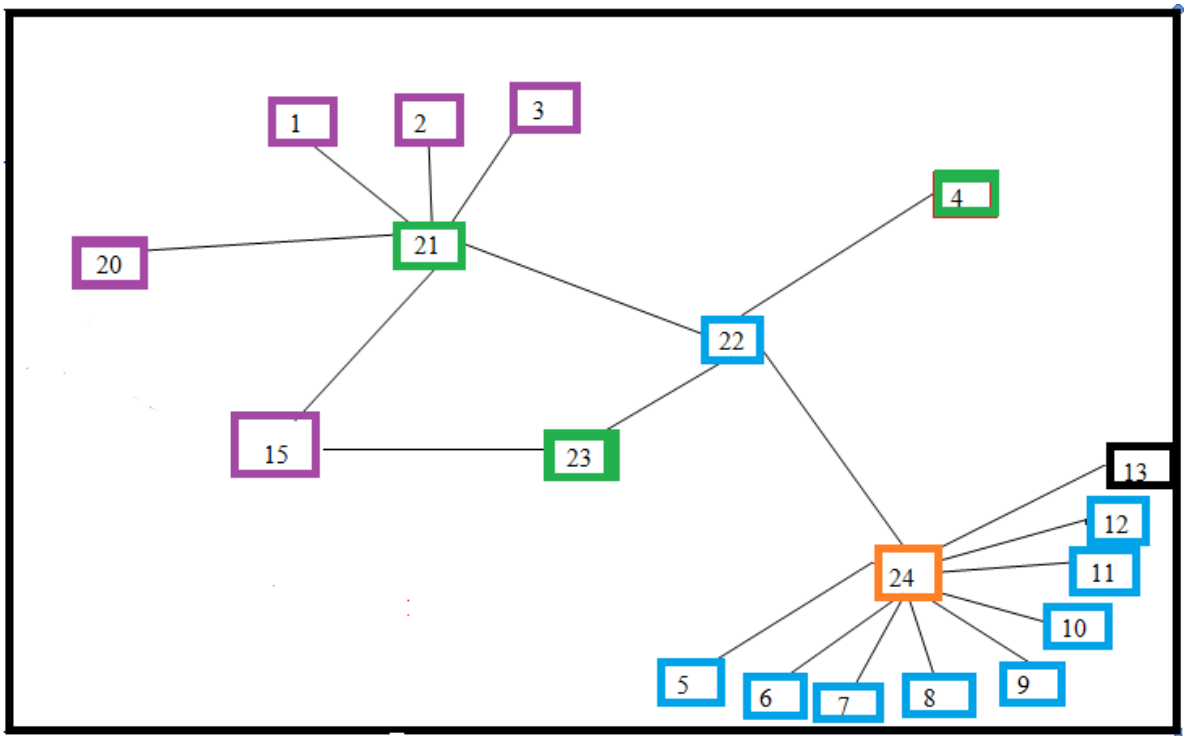

Figure 4.12 -Les différents degrés de localité (Géographique)

Dans la figure 4.12 les sites en couleur bleu désignent les clients de degré de localité 1 par rapport au site 24 (fichier demandé).Les sites en couleur vert désignent les clients de degré de localité 2 par rapport au site 22 (1er client demandeur).Les sites en mauve désignent les clients de degré de localité 3 par rapport au site 22 (1er client demandeur).

Les différents résultats sont représentés dans le tableau suivant :

Tableau 4.6 – Les temps de réponses moyens (Géographique-Degrés)

	Moyenne des temps de réponses (seconde).		
Degrés	1	2	3
Localité-LRU	527	1749	1751
Localité-LFU	527	1747	1748
Localité-EconomicBinomial	678	1219	1220
Localité-Zipf-based economic model	678	1218	1219

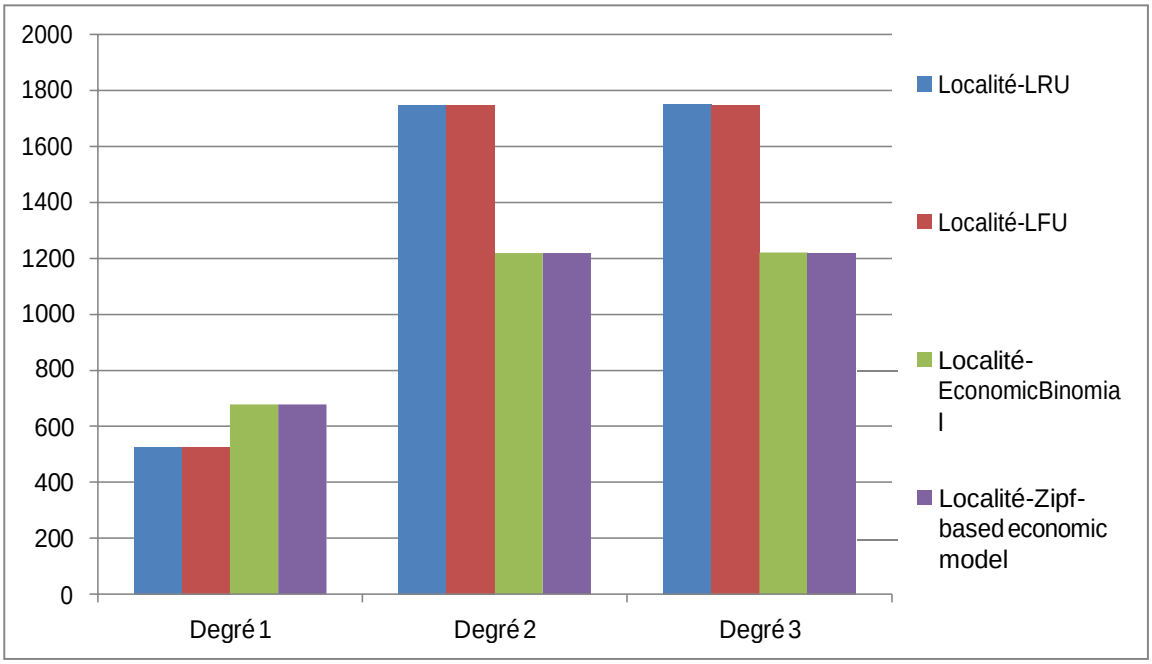

Figure 4.13 - Temps moyen d'exécution (Geographique-Degrés)

Ces graphes montrent que dans le modèle de localité géographique et pour le premier degré les stratégies Localité-LRU et Localité-LFU sont les plus performantes en termes de temps d'exécution.

On remarque aussi que pour le deuxième et le troisième degré la stratégie Localité-Zipf-based economic model est la plus performante avec un même temps d'exécution.

2.5.3 La Localité spatiale

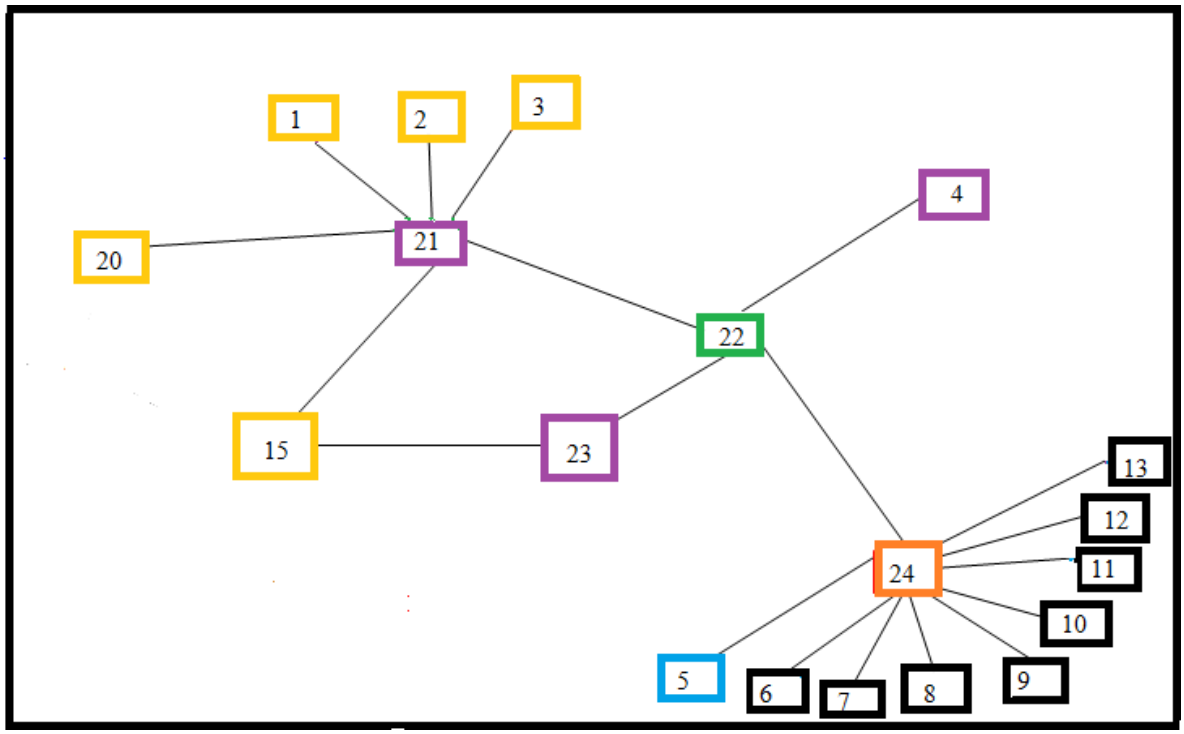

Figure 4.14 -Les différents degrés de localité (Spatiale)

Dans la figure 4.13 le site en couleur bleu désigne le client demandeur de fichier, par rapport au site 24 (fichier demandé en premier temps).Les sites en couleur vert désignent les fichiers de degré de localité 1 par rapport au site 24 .Les sites en mauve désignent les fichiers de degré de localité 2 par rapport au site 24 .Les sites en jaune désignent les fichiers de degré de localité 3 par rapport au site 24.

Les différents résultats sont représentés dans le tableau suivant :

Tableau 4.7 – Les temps de réponses moyens (Spatiale-Degrés)

	Moyenne des temps de réponses (seconde).		
Degrés	1	2	3
Localité-LRU	1066	1070	1075
Localité-LFU	1078	1081	1088
Localité-EconomicBinomial	1239	1263	1286
Localité-Zipf-based economic model	1379	1383	1384

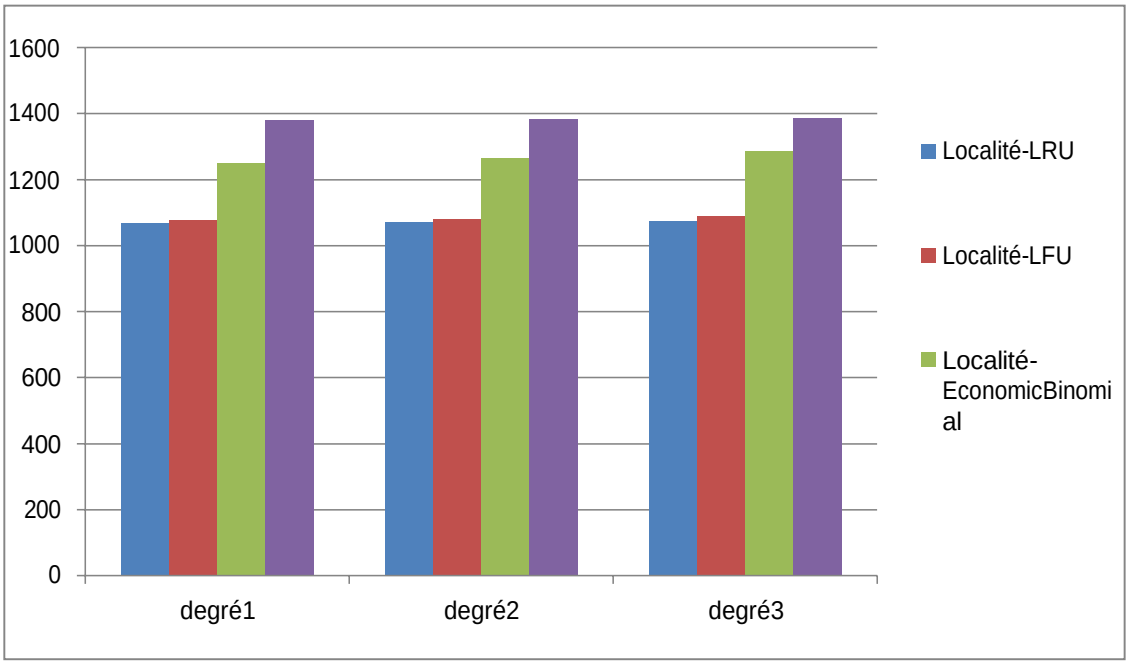

Figure 2.15- Temps moyen d'exécution (Spatiale-Degrés)

Ces graphes montrent que dans le modèle de localité spatiale et pour les trois degrés la stratégie Localité-LRU est la plus performante en termes de temps d'exécution.

2.6 Conclusion

Dans ce chapitre nous avons effectuée plusieurs tests avec notre approche sur le simulateur OptorSim. Les résultats obtenus des différentes expérimentations, ont démontré que le temps de réponse dans la localité temporelle a été réduit.

Et en prennent compte les degrés de localité pour chaque modèle d'accès, les résultats ont démontré les stratégies les plus appropriés.

Conclusion Générale

Les stratégies de réplication dans les grilles de données tentent d'optimiser certaines mesures de performance tels que : le temps de réponse, le coût d'accès et la consommation de la bande passante.

Dans ce sens, dans ce mémoire nous avons abordé le problème de placement des répliques dans les grilles de données prenant en compte les différents critères de localité : temporelle géographique et spatiale.

En premier temps, nous avons introduit les notions générales des grilles informatiques, puis nous avons étudié le processus de réplication dans sa globalité dans les systèmes distribués.

Ensuite nous avons présenté des brèves descriptions sur quelques simulateurs suivies d'une présentation détaillée du simulateur OptorSim qui a été choisie pour implémenter et simuler notre approche.

Le travail que nous avons réalisé dans ce mémoire est de proposer une nouvelle stratégie de placement des répliques en prenant en compte les propriétés de localité. Son objectif principal est d'améliorer les performances de la réplication en réduisant le temps de réponse d'exécutions des requêtes et ou le temps de transfert des répliques.

Plusieurs expérimentations ont été effectuées pour évaluer l'approche proposée. Les résultats obtenus ont été comparés à ceux des stratégies "*Lru, Lfu,* Economic Binomial *et* Zipf-*based economic model* " prédéfinies dans OptorSim. Nous constatons que les résultats obtenus sont satisfaisants puisque notre approche réduit considérablement le temps d'exécution par rapport aux stratégies prédéfinies dans OptorSim.

Dans le futur, nous projetons de déployer notre approche sur une grille réelle, et de prendre en considération d'autres critères tel que l'espace de stockage.

Bibliographie

[1] I. Foster. “What is the grid? a three point checklist.” *GRIDtoday*, Vol.1, N°.6,July2002.

[2] J. Nabrzyski et Al. “Grid Resource Management”, Kluwer Publishing, edition, 2003.

[3] Globus alliance. http : //www.globus.org/.

[4] I. Foster et Al. “The anatomy of the grid: enabling scalable virtual organizations”, *International Journal of High Performance Computing* vol.15, No.3, 2001.

[5] D.G. Cameron et Al “OptorSim: a simulation tool for scheduling and replica optimization in data grids”, *Proceedings Computing in High Energy and Nuclear Physics* ,Paysinterlaken,Switzerland , 2004.

[6] I. Foster, C.Kesselman “The Grid 2: Blueprint for a new computing infrastructure”, *second edition, Elsevier* 2004

[7] M. Tang et Al. “Dynamic replication algorithms for the multi-tier data grid”, *Future Generation Computer Systems journal*, Vol. 21, No.5, 2005, pp 775- 790.

[8] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Design of a Replica Optimisation Framework. Technical Report DataGrid02-TED-021215,CERN, Geneva, Switzerland, December 2002. EU DataGrid Project

[9] I. Foster et Al. “The anatomy of the grid: enabling scalable virtual organizations”, *International Journal of High Performance Computing* vol.15, No.3, 2001.

[10] I.Foster K.Ranganathan, “Design and evaluation of replication strategies for a high performance data grid”, Proceedings International Conference on Computing and High Energy and Nuclear Physics, Beijing, China, 2001.

[11] I. Foster K.Ranganathan, " Identifying dynamic replication strategies for high performances data grids", Proceedings. 3 rd IEEE/ACM International Workshop of grid computing, London, UK, 2001.

[12] Y. Nemati et Al "A novel data replication policy in data grid", Australian Journal of Basics and Applied Sciences, Vol.6, N°.7, 2012.

[13] G. Belalem, "Contribution à la gestion de la cohérence de répliques de fichiers dans les systèmes à large échelle", *Thèse de Doctorat*, département d'informatique, université d'Oran, 2007.

[14] R. Buyya and M. Murshed. Gridsim : "A toolkit for the modeling andsimulation of distributed resource management and scheduling for gridcomputing". The Journal of Concurrency and Computation: Practice and Experience (CCPE), 14(13-15), 2002.

[15] H. Casanova. Simgrid: a toolkit for the simulation of application scheduling. In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01), Brisbane, Australia, pages 430-437, may 2001.

[16] H. Casanova, A. Legrand, and L. Marchal. Scheduling distributed applications: the simgrid simulation framework. In Proceedings of the third Bibliographie IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03), may 2003.

[17] Dalibor Kluscek, Hana Rudov. Alea 2 – Job Scheduling Simulator. SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain. Copyright 2010 ICST, ISBN 78-963- 9799-87-5.

[18] D.Yang et Al. "A Comparative study of Replicas Placement Strategies in Data Grids", P*roceedings of Advances in Web and Network Technologies, and Information Management*, Computer Sciences vol 4537,2007, pp 135-143.

[19] The European DataGrid Project, http://www.eu-datagrid.org

[20] DG. Cameron, RC. Schiaffino, J. Ferguson, P. Millar, C. Nicholson, K. Stockinger, F.Zini, "optorsim v2.1 installation and user guid", 2006.

.