MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

BENZINA SOUELEF ET BENBADRA NACERA

THEME :

GESTION DE LA COHERENCE DES DONNEES DANS LES SYSTEMES DISTRIBUEES A GRANDE ECHELLE

Soutenu le :

Devant le jury composé de :

Nom et Prénom	Grade	Université de Mostaganem	Président
Nom et Prénom	Grade	Université de Mostaganem	Examinateur
Nom et Prénom	Grade	Université de Mostaganem	Encadreur

Année Universitaire 2020-2021

Dédicaces

A nos chers parents, pour tous leurs sacrifices, leurs encouragements, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

A toute nos familles pour leur soutien tout au long de mon parcours universitaire, que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour nous.

Remerciements

Tout d'abords, nous remercions particulièrement notre encadreur Mr BOUHAROUA Farouk enseignant à l'Université AbdElhamid Ibn Badis de Mostaganem, pour son aide précieuse et pour ses encouragements, et nos familles qui nous ont soutenue moralement. Nous ne pouvons pas terminer sans exprimer un remerciement venant du plus profond du cœur à toutes nos amies.

Résumé

La réplication des données est un mécanisme clé pour construire un système de gestion des données fiable et efficace. En effet, en conservant plusieurs copies de chaque donnée, la durabilité peut être améliorée. De plus, placer des copies appropriées peut réduire le temps d'accès aux données. Cependant, lors de la mise à jour des données, le fait de disposer de plusieurs copies pour un seul élément de données peut créer des problèmes de cohérence.
Dans les systèmes informatiques distribués, le partage d'informations est assuré par la réplication. Il existe plusieurs problèmes liés à la maintenance de la cohérence entre les différentes répliques, et pour cela nous proposons une stratégie de placement des répliques basée sur le choix de l'élément de stockage et le meilleur fichier répliqué et supprimé. Son évaluation par simulation sur OptorSim montre une amélioration considérable des temps d'exécution des jobs et une diminution importante du nombre de répliques entrainant moins d'espace de stockage et donc moins de problèmes de mises à jour.
Dans ce contexte, dans ce travail, nous allonsprésenter les concepts de la réplication des données et la cohérence dans les systèmes distribués.

Mots-clés :

La réplication, système distribués, cohérence, fiabilité, quorum, OptorSim.

Abstract:

Data replication is a key mechanism to build a reliable and efficient data management system. Indeed, by keeping several copies of each data, the durability can be improved. In addition, placing appropriate copies can reduce data access time. However, when updating data, having multiple copies of a single data item can create consistency issues.

In distributed computing systems, information sharing is provided by replication. There are several issues related to maintaining consistency between different replicas, and for this we provide a replica placement strategy based on the choice of storage item and the best replicated and deleted file. Its simulation evaluation on OptorSim shows a considerable improvement in job execution times and a significant reduction in the number of replicas resulting in less storage space and therefore fewer update problems.

In this context, in this work, we will present the concepts of data replication and consistency in distributed systems.

Keywords:

Replication, distributed system, consistency, reliabity, quorum,OptorSim.

ملخص

يعد تكرار البيانات آلية رئيسية لبناء نظام إدارة بيانات موثوق وفعال. في الواقع ، من خلال الاحتفاظ بعدة نسخ من كل بيانات ، يمكن تحسين المتانة. بالإضافة إلى ذلك ، يمكن أن يؤدي وضع نسخ مناسبة إلى تقليل وقت الوصول إلى البيانات. .ومع ذلك ، عند تحديث البيانات ، يمكن أن يؤدي وجود نسخ متعددة من عنصر بيانات واحد إلى حدوث مشكلات تناسق في أنظمة الحوسبة الموزعة ، يتم توفير مشاركة المعلومات عن طريق النسخ المتماثل. هناك العديد من المشكلات المتعلقة بالحفاظ على الاتساق بين النسخ المتماثلة المختلفة ، ولهذا نقدم إستراتيجية وضع نسخة طبق الأصل بناءً على اختيار تحسنًا كبيرًا في أوقات OptorSim عنصر التخزين وأفضل ملف تم نسخه وحذفه. يُظهر تقييم المحاكاة الخاص به على .تنفيذ المهام وانخفاضًا كبيرًا في عدد النسخ المتماثلة مما يؤدي إلى تقليل مساحة التخزين وبالتالي تقليل مشاكل التحديث في هذا السياق ، في هذا العمل ، سوف نقدم مفاهيم النسخ المتماثل واتساق البيانات في الأنظمة الموزعة.الكلماتالدالة

،النسخ المتماثل ، النظام الموزع ، الاتساق ، الموثوقية ، النصاب القانونيOptorSim.

.

Liste des figures

Liste des tableaux

Table de matière

Introduction Générale

Les systèmes distribués sont un ensemble d'ordinateurs indépendants qui constituent un système cohérent aux yeux des utilisateurs. La principale motivation d'un système distribué est d'améliorer le partage des ressources, c'est-à-dire que les ressources disponibles dans un ordinateur peuvent être utilisées par un autre ordinateur. Les systèmes de données distribués doivent fournir un stockage fiable et performant pour garantir un certain degré de cohérence. Pour cette raison, la réplication est une technique clé. Autrement dit les systèmes distribués peuvent avoir un objectif commun, tel que la résolution d'un gros problème informatique. Ensuite, l'utilisateur traite tous les processeurs autonomes comme une seule unité. Ou bien, chaque ordinateur peut avoir ses propres utilisateurs et avoir ses propres besoins, et le but d'un système distribué est de coordonner l'utilisation des ressources partagées ou de fournir aux utilisateurs des services de communication.

Parmi les solutions proposées pour gérer ces problèmes est la réplication des données. Ce mécanisme est considéré comme un outil très puissant dans les systèmes distribués à grande échelle pour donner plus de performance aux systèmes et une haute disponibilité des données.

La difficulté de la mise en œuvre la réplication réside dans la cohérence mutuelle des copies de chaque donnée. Autrement dit, elle réside dans la mise en œuvre des opérations de lecture et d'écriture (éventuellement des opérations plus complexes, mais toujours divisées entre consultation et modification.

Le concept central des systèmes distribués est le modèle de cohérence utilisé. Le modèle de cohérence est présenté comme un contrat entre le système et le programmeur. Il définit la condition pour déterminer la valeur renvoyée par la lecture en fonction de l'écriture.

Dans ce contexte, notre travail consiste à comparer entre les stratégies (protocoles) de gestion de la cohérence des données dans les systèmes distribués à grande échelle. Pour cela nous utiliserons le simulateur OptorSim, sur lequel nous introduisons les algorithmes de certains protocoles de cohérence, ce quinous permettradéterminerle meilleur de ces protocoles.

Notre document est organisé en quatre chapitres :

Nous présentons au premier chapitre les principes fonctionnement des systèmes distribués. Ensuite nous définissons.

En deuxième chapitre les différents concepts de réplication et de gestion de la cohérence des répliques.

Dans le troisième chapitre nous allonsprésenter le simulateur OptorSim.

Enfin, dans le quatrième chapitre nousdéfinissons notre approche et déduire la meilleur approche.

Chapitre1

Les systèmes distribués

1.1 Introduction

Dans les années 40, le monde a connu l'apparition des premiers ordinateurs (des gros calculateurs centralisés et très coûteux). La naissance des micros ordinateurs peu chers et facile à interconnecter en début des années 80, a fait naître de nouveaux concepts tels celui du système distribué. De nos jours, les machines multiprocesseurs se vulgarisent, les réseaux locaux sont utilisés dans presque tous les laboratoires d'informatique, quant aux réseauxà grande distance, ils arrivent à s'infiltrer dans la plupart des foyers permettant un accès aisé à de gigantesques et beaucoup d'autres services.

Pour comprendre les systèmes distribués, nous allons présenter dans les sections suivantes de manière générale.

1.2 Systèmes distribués

Un système distribué est un système disposant d'un ensemble d'entités communicantes, installées sur une architecture d'ordinateurs indépendants reliés par un réseau de communication, dans le but de résoudre en coopération une fonctionnalité applicative commune.

Autrement dit, un système distribué est défini comme étant un ensemble de ressources physiques et logiques géographiquement dispersées et reliées par un réseau de communication dans le but de réaliser une tâche commune. Cet ensemble donne aux utilisateurs une vue unique des données du point de vue logique [1].

1.2.1 Avantages d'un système distribué

- **Coût :** Un rapport performance, prix des micro-processeurs meilleur que celui des supercalculateurs.
- **Partage et mise à disposition :** Partager des ressources et des services disponibles.
- **Répartition géographique :** Mettre à disposition des usages les moyens informatiques locaux en même temps que ceux distants de leurs collègues.

- **Puissance de calcul :** Paralléliser les algorithmes de calcul avec des environnements d'exécution spécifique comme PVM (parallèle Virtual Machine).
- **Flexibilité :** permet d'exécuter un travail sur la machine la plus disponible.
- **Adaptabilité** à une forte croissance des besoins informatiques d'une entreprise [1].

1.2.2 Critères d'un système distribué

Un bon système distribué est un système qui semble très centralisé. Afin de simplifier la programmation dans les applications distribuées, il est préférable de masquer autant que possible les fonctionnalités subtiles de la distribution. Mais il est difficile d'être transparent dans tous les aspects.

- **La transparence :**

Fait pour une fonctionnalité un élément d'être invisible ou caché à l'utilisateur ou un autre élément formant le système distribué.

Le but est de cacher l'architecture, le fonctionnement de l'application ou de système distribué pour apparaître à l'utilisateur comme une application unique cohérente.

- **La souplesse :**

La structure du système doit faciliter son évolution. Pour cela le noyau doit être le plus petit possible et assurer quatre services minimums :

- Les mécanismes de communication interprocessus.
- La gestion de la mémoire.
- La gestion et l'ordonnancement des processus.
- La gestion des entrées/sorties de bas niveau.

- **La fiabilité :**

Le système doit rester opérationnel en cas de panne, voire de plusieurs serveurs.

Les fichiers et autres ressources doivent être protégés contre une utilisation abusive.

Il faut assurer une tolérance en panne au niveau de l'utilisation des fichiers en introduisant par exemple des serveurs de fichiers sans état.

- **Les performances :**

L'exécution des programmes devrait être plus rapide que sur un système monoprocesseur.

Les performances peuvent être mesurées par : le temps de réponse, le débit, le taux d'utilisation du système.

- **Scalability :**

Le passage à l'échelle réelle ou scalability concernent au moins trois aspects :

- Nombre d'utilisation et/ou de processus.
- Distance maximale entre les nœuds.
- Nombre de domaines administratifs.

1.2.3 Inconvénient des systèmes distribués

Dans un système où la communication s'effectue via un réseau et ente entités différentes de nombreux problèmes peuvent naître et créer d'énormes dégâts parmi lesquels [2] :

- Le partage et distribution de données impose mécanisme complexes
- Problèmes inhérents aux communications.
- Logiciels de gestion difficiles à concevoir.

1.2.4 Sécurité des systèmes distribués

La nature d'un système distribué fait qu'il est beaucoup plus sujet à des attaques.

- La communication à travers le réseau peut être interceptée
- On ne connaît pas toujours bien un élément distant avec qui on communique [3].

Solutions

Face à ces problèmes de sécurité des solutions tel celui :

- De la connexion sécurisée par authentification avec les éléments.
- Du cryptage des messages circulant sur le réseau peuvent toujours être un plus pour ces systèmes dit distribués ou repartis.

1.3Exemples des systèmes distribués

1.3.1 Grilles informatiques

L'objectif des Grilles est de lier les ressources collectives de plusieurs parties indépendantes pour créer un supercalculateur virtuel haute performance capable d'exécuter des tâches de calcul intensif. La grille est généralement liée aux Sciences : la science nécessite un calcul hautement distribué [4].

1.3.2 Cloud computing

Le cloud est conceptuellement similaire au concept de la grille informatique et fait face aux mêmes défis. Le terme Cloud (nuage en anglais), particulièrement dans l'air du temps, recouvre l'ensemble des solutions de stockage distant. En clair, vos données, au lieu d'être

stockées sur vos disques durs ou mémoires, sont disponibles sur des serveurs distants et accessibles par internet [4].

1.4Conclusion

Dans ce chapitre, nous nous sommes focalisées sur les concepts de base des systèmes distribués. Nous avons présenté les caractéristiques des systèmes distribués, leur sécurité ainsi que deux exemples de système distribué.Le chapitre suivant sera consacré à la réplication et la cohérence.

Chapitre 2

La réplication et la cohérence

2.1 Introduction

Jusqu'à maintenant les développeurs ont essayésde donner plus de performance à leurs systèmes, pour diminuer les coûts de communications et agrandir la puissance de calcul. Avec l'apparition dans les systèmes distribués à grande échelle, les problèmes de gestion des données deviennent très compliqués, surtout en ce qui concerne l'équilibrage de la charge des nœuds, la disponibilité des données partagées, le dynamicien et la volatilité des nœuds du réseau.

Parmi les solutions proposées pour gérer ces problèmes est la réplication des données. Ce mécanisme est considéré comme un outil très puissant dans les dans les systèmes distribués à grande échelle pour donner plus de performance aux systèmes et une haute disponibilité des données.

La réplication peut dépendre du concept de propagation des mises à jour, donc dans un environnement largement distribué comme les grilles, la mise à jour effectuée sur une des copies d'une donnée doit être propagée vers toutes les autres copies, cela peut générer une dégradation des performances ou une incohérence entre la donnée modifiée et ses copies. Pour cela un modèle de gestion de la cohérence doit être mis en place, et doit être bien adapté à la stratégie de réplication utilisée.

Jusqu'ici le compromis entre cohérence et réplication pose toujours des problèmes surtout dans les systèmes à grande échelle (grille).

Dans ce chapitre nous allons discuter le concept de base concernant la réplication, et la cohérence des données répliquées.

2.2 La réplication des données

La réplication des données est un outil très efficace utilisé dans les systèmes distribués, pour donner plus de disponibilité aux données et en conséquence plus de performance d'accès

aux données. La réplication consiste à créer des copies d'une donnée et les stocker dans des endroits différents situés dans le réseau. [5]

2.2.1 Types de Réplication

Il existe deux types de réplications : la réplication synchrone et la réplication asynchrone.

2.2.1.1 La réplication synchrone

La réplication synchrone utilise un site maître qui pousse les mises à jour en temps réel vers un ou plusieurs sites esclaves.

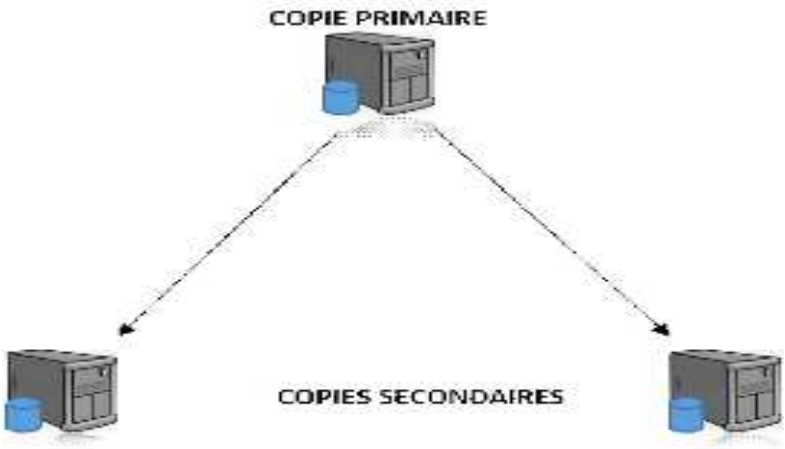

Figure 1. Réplication synchrone

2.2.2.2 La réplication asynchrone

Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, une horloge par exemple :

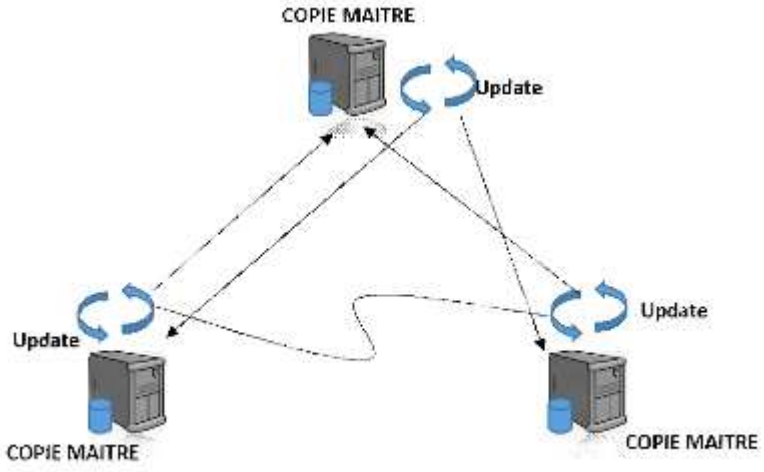

Figure2. Réplication asynchrone

2.3 Protocole de réplication

Nous avons cité trois principaux de protocole qui sont utilisées pour la gestion de réplication dans les systèmes distribués :

2.3.1 Réplication passive

Dans ce protocole, la copie primaire reçoit une requête d'un client et l'exécute. Cette copie faite tous les traitements. En cas cette copie tombe en panne, une copie secondaire devient la nouvelle copie primaire pour assurer la cohérence.

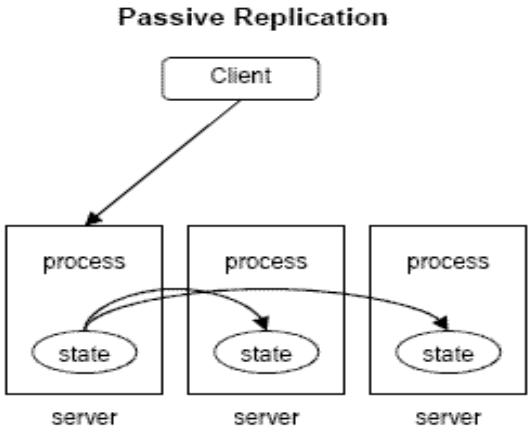

Figure3 : Réplication passive

2.3.2 Réplication active

Dans ce protocole, toutes les copies jouent le même rôle. Ils reçoivent les mêmes séquences entièrement structurées.

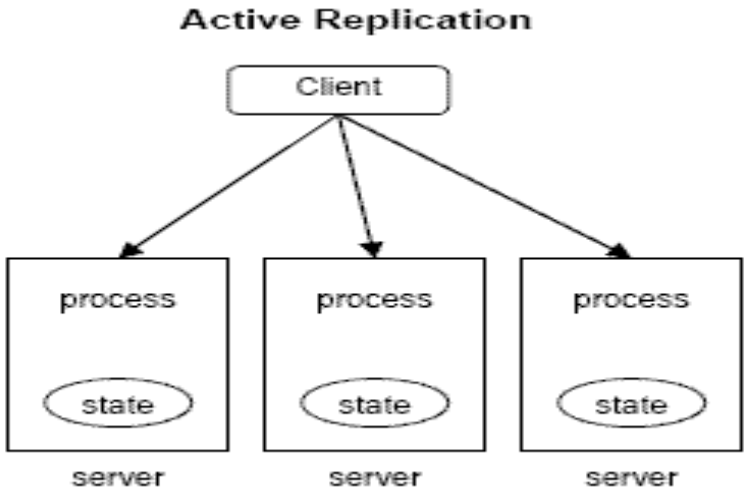

Figure 4 : réplication active

2.3.3 Réplication semi-active

C'est un protocole qui situé entre les deux protocoles précédents. Toutes les copies exécutées en même temps, mais juste une seule copie envoie la réponse, et les autres copies sont internes.

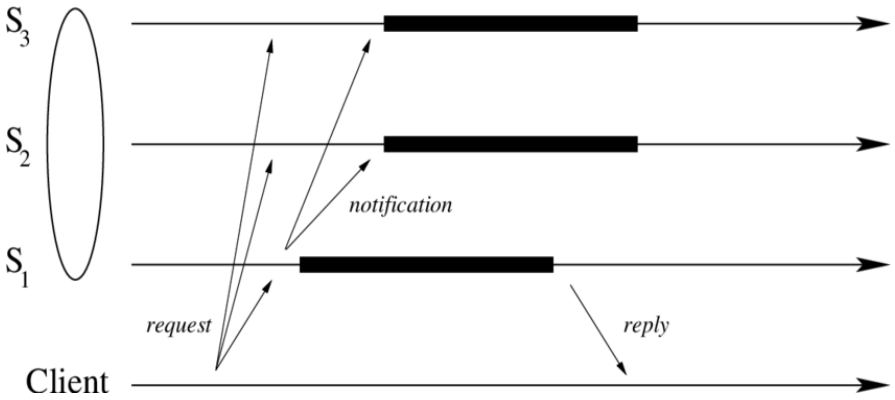

Figure 5 : Réplication semi-active

2.4 La cohérence

La cohérence est une relation qui définit le degré de similitude entre les copies d'entités répliquées. Idéalement, cette relation peut caractériser des copies qui ont le même comportement. Dans les situations réelles, le développement des copies est différent et la cohérence définit la limite des différences admissibles entre les copies. Traditionnellement, on dit que si la valeur retournée par une opération de lecture de données correspond toujours à la dernière valeur écrite des mêmes données, la mémoire est cohérente (cohérence stricte).

Dans les systèmes distribués la cohérence des répliques doit assurer que n'importe quelle opération de lecture sur une donnée (ou une copie de donnée) doit retourner la dernière valeur écrite [6].

Une transaction est une séquence d'instructions, modifiant des données (incluant lectures, écritures) afin de garantir la cohérence.

2.4.1 Les modèles de cohérence

- **Cohérence forte :**Les modèles de cohérence forte sont caractérisés pas des contraintes fortes entre la dernière écriture et la prochaine lecture.
- **Cohérence faible :** Ce modèle garantit que la mémoire est cohérente à chacun des points de synchronisation pendant lequel toutes les informations sont mises à jour.

Et nous pouvons dire que la cohérence est en relâchement dans les deux cas suivants :

- Lorsque les o
- pirations de lecture et d'écriture sont terminées sur tous les sites l'opération de relâchement peut se terminer.
- Seulement dans tous les cas peuvent lire et écrire des opérations L'opération de collecte précédente a été réalisée sur tous les sites.

Les acquisitions précédentes ont été réalisées dans tous les emplacements [6].

2.4.2 Les approches de gestion de la cohérence

2.4.2.1 Les approches pessimistes

Les approches pessimistes garantissent la cohérence en prévenant les risques de conflit.

Les algorithmes pessimistes empêchent l'accès à une réplique à moins qu'elle ne soit mise à jour.

L'avantage de ces méthodes est d'éviter les problèmes liés au rapprochement.

2.4.2.2 Les approches optimistes

Cela un algorithme de réconciliation est mis en place. Cette approche ne garantit pas que toutes les copies sont à jour [6].

Dans les approches optimistes les utilisateurs peuvent accéder à n'importe quelle réplique à tout moment. Supposant que les conflits de mise à jour sont rares, l'idée ici est de laisser l'utilisateur lire ou écrire n'importe quelle réplique, la mise à jour se propage et les conflits sont résolus après qu'ils se produisent. Ces approches offrent plus d'avantages par rapport aux approches pessimistes en termes de disponibilité de flexibilité et de sociabilité.

2.4.3 Les protocoles de gestion de cohérence

Plusieurs protocoles de gestion de la cohérence des données répliquées ont été proposés. Nous citons :

- **ROWA** (Read Once Write All) : la disponibilité des lectures est améliorée avec ce protocole. Les lectures verrouillent et accèdent à une seule copie, tandis que les écritures continuent à verrouiller et modifier toutes les copies. Toutes les copies sont mises à jour de manière synchrone ce qui implique que toutes les copies sont à jour. Mais si un site tombe en panne, toutes les écritures sont bloquées jusqu'à ce que la panne soit réparée [6].
- **ROWAA** (Read Once Write All Available) : ce protocole est adapté aux cas de pannes en ne verrouillant que les copies disponibles. Quand une copie reprend sa disponibilité, elle doit d'abord se resynchroniser pour effectuer les mises à jour manquantes. Toutes les copies disponibles sont mises à jour de manière synchrone et les copies indisponibles sont mises à jour de manière asynchrone ce qui implique que toutes les copies disponibles sont à jour et il existe une possibilité de copies non à jour. ROWAA tolère (n-1) pannes et ne supporte pas les partitions réseaux et les pannes de communication [6].
- **Primary Copy ROWA** : Une copie est définie comme copie de base, les autres comme copies secondaires. Si la copie de base tombe en panne, une autre est désignée comme nouvelle primaire. On trouve deux variantes :

Primaire/Secondaire : Les mises à jour sont faites sur la copie de base, les copies secondaires sont mises à jour de manière asynchrone ce qui implique que la copie de base est toujours à jour.

Primaire/Sauvegarde : Les mises à jour sont faites sur la copie de base, et une des copies secondaires est désignée comme étant la sauvegarde, celle-ci est responsable de la reprise sur défaillance de la copie de base. Toutes les autres copies secondaires sont mises à jour de

manière asynchrone ce qui implique que, lorsqu'elle est disponible, la copie de base est toujours à jour [6].

- **RAWA** (Read Any, Write All) : c'est un protocole basique, il consiste à obtenir un verrou exclusif sur toutes les copies avant d'effectuer une écriture sur une des copies [6].
- **Indépendant** : Dans cette approche, les mises à jour sont faites sur des copies quelconques. Mais ceci peut engendrer la possibilité d'avoir des copies incompatibles ou incohérentes, pour
- **Quorums :** ce protocole est basé sur le suffrage (votes). Avant chaque opération de lecture ou d'écriture un certain nombre de votes doit être requis.
- **IceCube :** cette approche représente un système de réconciliation, dont le but est d'exécuter une combinaison optimale de mises à jour concurrentes. Icecube déduit un journal optimal à partir des journaux des différents sites, contenant un nombre maximum de mises à jour qui ne présentent pas de conflits

Dans cette approche les mises à jour sont modélisées par des actions, une action est composée des paramètres suivants :

- Cibles : identifie les objets modifiés par l'action
- Pré-condition : permet de détecter les conflits.
- Opération : permet d'accéder aux objets partagés.
- Données privées : données propres à l'action, comme les paramètres et le type de l'opération.
- La réconciliation est effectuée en trois phases :
- Phase de sélection : parmi les ordonnancements retenus, on doit choisir un seul, le critère de choix est défini par l'administrateur.
- Phase d'ordonnancement : avec la combinaison des différentes actions issues des journaux des différents sites, on construit toutes les exécutions possibles.
- Phase de simulation : par l'exécution des différents ordonnancements trouvés dans la phase précédente on vérifie et on ne retient que ceux qui répondent aux contraintes de simulation.

Cette approche garantit la convergence des copies par l'imposition d'un ordre total sur l'exécution des opérations [7].

Bayou : dans cette approche une copie de l'objet partagé est située sur chaque serveur dans le système. Les applications interagissent avec cette copie via une interface de programmation Bayou. Les utilisateurs peuvent donc effectuer des opérations (Lectures/Écritures) sur cette donnée. Une fois qu'un serveur reçoit une opération, il tente de l'exécuter, pour assurer la

convergence des copies. Les serveurs doivent exécuter les opérations dans le même ordre. Cet ordre est décidé par un serveur principal désigné au lancement du système.
Un journal des opérations exécutées est maintenu par chaque serveur. Ce journal se compose de deux parties : un préfixe qui contient les opérations validées par le serveur principal, qui sont ordonnées définitivement selon un ordre total introduit lors de la validation, la deuxième partie du journal (appelé provisoire) contient le reste des opérations, qui sont ordonnées selon un ordre total définit au fur et à mesure de la réception de nouvelles opérations.
L'inconvénient de cette approche est le passage à l'échelle, car la mise en place d'un préfixe commun nécessite un ordre total, ce qui limite le passage à l'échelle [6].

2.4.4 Les quorums à consensus

La décision sur le nombre de copie à impliquer dans les opérations est appelée sélection de quorum [7].
Un énoncé moins formel de ces règles est comme suit :

- Un quorum d'écriture est requis pour la réussite d'une écriture.
- Un quorum de lecture est requis pour la réussite d'une lecture.
- Les quorums de lecture et d'écriture doivent toujours interagir.
- Les numéros de version doivent être utilisés, ou les écritures doivent couvrir la majorité du quorum de lecture.

2.4.4.1 Le principe des quorums à consensus pour la réplication

Soient :
Ne= quorum nombres d'écriture.
Nl= quorum nombres de lecture.
N=nombres de copies.

Pourgarantir la lecture de la plus récente écriture (la cohérence des données) il faut :

- avoir l'accord de ' *ne* 'copies pour effectuer une écriture.
- avoir l'accord de' *nl* ' copies pour effectuer une lecture.
- que*ne* +*nl*>*N*.

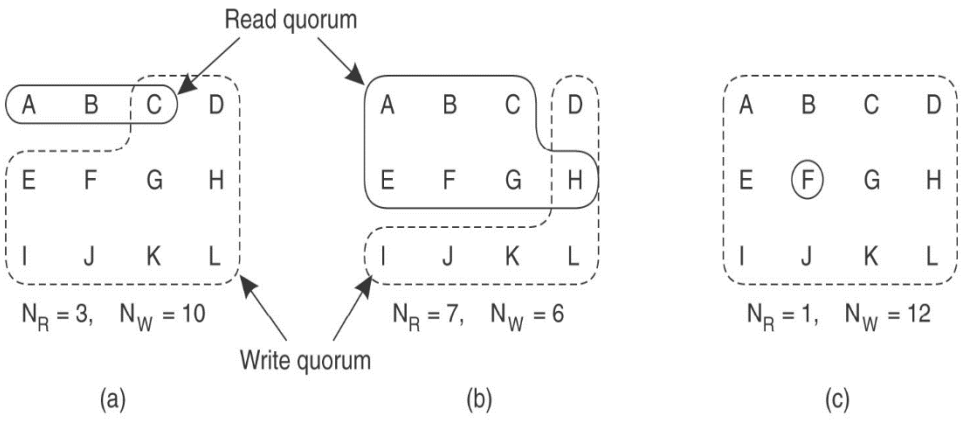

Figure6 : Exemple pour les Quorums [8].

2.5 Conclusion

Dans ce chapitre nous avons entamé les concepts de la réplication des données ainsi que ceux de la cohérence des données répliquées. Nous avons présenté les notions et les caractéristiques des protocoles de la réplication, et de la cohérence.

Chapitre 3

Le simulateur OptorSim

3.1.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.

En conséquence, beaucoup d'outils de simulation standards et spécifiques aux applications ont été établis. Ce chapitre permettra de présenter un simulateur destiné aux applications de grilles. Et puisque notre choix est porté sur le simulateur OptorSim pour simuler les protocoles, nous allonsprésenteren détailce simulateur.

3.2.Les simulateurs

Le simulateur est un programme capable d'interpréter des modèles dynamiques utilisé pour générer l'interface requise sur le modèle après l'exécution et collecter la sortie. [9]

3.2.1. Le simulateur OptorSim

OptorSim est un simulateur de grille de données en open source, écrit en Java. Il a été développé dans le cadre du projet Européen Data Grid (EDG). [9]

Ce simulateur sera présenté en détail un peu plus bas.

3.2.2. Le simulateur Alea

Basé 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 VI-3, qui détermine la sélection des ressources et la répartition de la tâche.[9]

3.2.3. Le simulateur GridSim

Est développé par Australien Melbourne UniversityRaj Kumar the Buyya Leadership Development. Comme SimGrid, GridSim est un simulateur consacré à l'ordonnancement des

tâches dans les grilles. GridSim fournit un service complet pour la simulation de différentes classes de ressources hétérogènes, des utilisateurs et d'applications. GridSim permet la

flexibilité et l'extensibilité pour incorporer de nouveaux composants dans son infrastructure.[10]

3.2.4. Le simulateur SimGrid

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) 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.[9]

3.3. Le simulateur OptorSim

OptorSim est un logiciel Open Source développé pour simuler et évaluer le comportement des algorithmes de réplication.OptorSim a était développer suit aux manques d'environnement de simulation pour les applications nécessitant la manipulation de grands ensembles de données dans une grille.

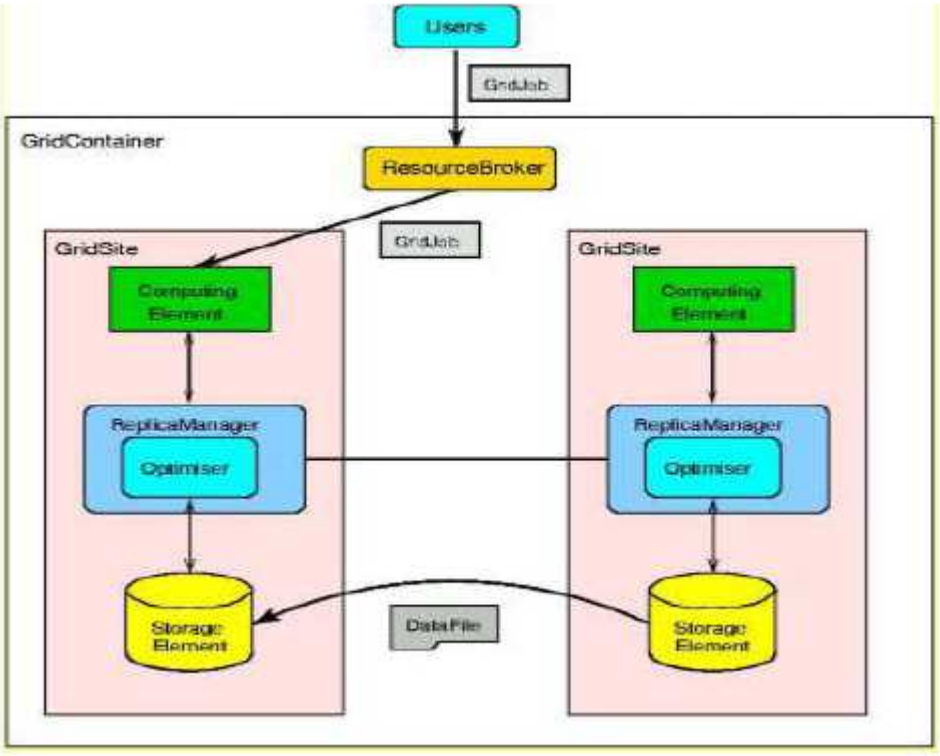

Figure 3.1 Structure du simulateur OptorSim

3.3.1.Principe

OptorSim modélise les interactions des composants individuels d'une grille de données (DataGrid).[10]

OptorSim reproduit les composants d'une grille réelle. Il est donc constitué de (figure 3-1) :

- 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 constitué de :
    - Une ou plusieurs ressources de calcul (CE : ComputingElement). Sur la version actuelle d'OptorSim, on accepte un seul CE seulement. Un CE peut contenir un ou plusieurs processeurs ou « workernodes ».

- Une ou plusieurs ressources de stockage (SE : Storage Element).
- Un gestionnaire de répliques, responsable de la création, suppression, placement, cohérence des répliques.

3.3.2.Motivation

OptorSim est un bon simulateur de grille qui vous permet de tester des stratégies de réplication mesurer la performance de la grille de données en fournissant un ensemble d'informations très intéressant en fin de simulation, ce qui permet aussi de faire cette dernière en une seule fois le temps de plus court.

3.4. Choix de simulateur

Après une description du simulateur de grille de données le plus célèbre, il y a un autre pour une recherche comparative entre ceux-ci, notre choix se porte sur le simulateur de grille Données OptorSim.

- OptorSim a été développé à l'origine pour tester des stratégies de réplication dynamique Utilisé pour optimiser l'efficacité de l'écran.
- Son langage de programmation "java" permet d'adopter une approche orientée objet, très efficace Lorsqu'il s'agit de modéliser et d'exploiter un réseau avec différents composants Plusieurs threads se déroulent en même temps.
- Son architecture simple et la modification et l'intégration faciles de nouvelles fonctionnalités de code font taches de programmeur plus simple.

Toutes ces raisons ont motivé le choix de ce simulateur.

3.5. Les fichiers de configuration

Il existe plusieurs fichiers de configuration utilisés pour contrôler les différents paramètres d'entrer d'OptorSim.

- 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 grille (Grid Configuration File) : décrit la topologie de la grille ainsi que les ressources que contient chaque site.

- Fichier de configuration de paramètre (Parameters file) : permet d'initialiser les paramètres d'entrée avant la simulation.
- Fichier de configuration de la bande passante (Bandwidth Configuration File) : est le fichier de configuration du trafic sur le réseau.

3.5.1. Fichier de configuration des taches

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.

cms_testbed_jobs.conf

```
# The jobs run on the cms testbed
#
# File Table
#
# Each file is 10 Gbyte.
# Secondary data set size based on cdf5858 reduced 100 times
\begin{filetable}
jpsi0 1000 0
jpsi1 1000 1
jpsi2 1000 2
jpsi3 1000 3
jpsi4 1000 4
jpsi5 1000 5
jpsi6 1000 6
jpsi7 1000 7
jpsi8 1000 8
jpsi9 1000 9
jpsi10 1000 10
jpsi11 1000 11
highptlep0 1000 100
highptlep1 1000 101
incelec0 1000 200
incelec1 1000 201
incelec2 1000 202
incelec3 1000 203
incelec4 1000 204
incmuon0 1000 300
incmuon1 1000 301
incmuon2 1000 302
incmuon3 1000 303
incmuon4 1000 304
incmuon5 1000 305
```

Figure 3.2Contenu du fichier cms_testbed_jobs.conf

3.5.2. Fichier de configuration de grille

Dans ce fichier :

- La première colonne désigne le nombre de workernodes 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.

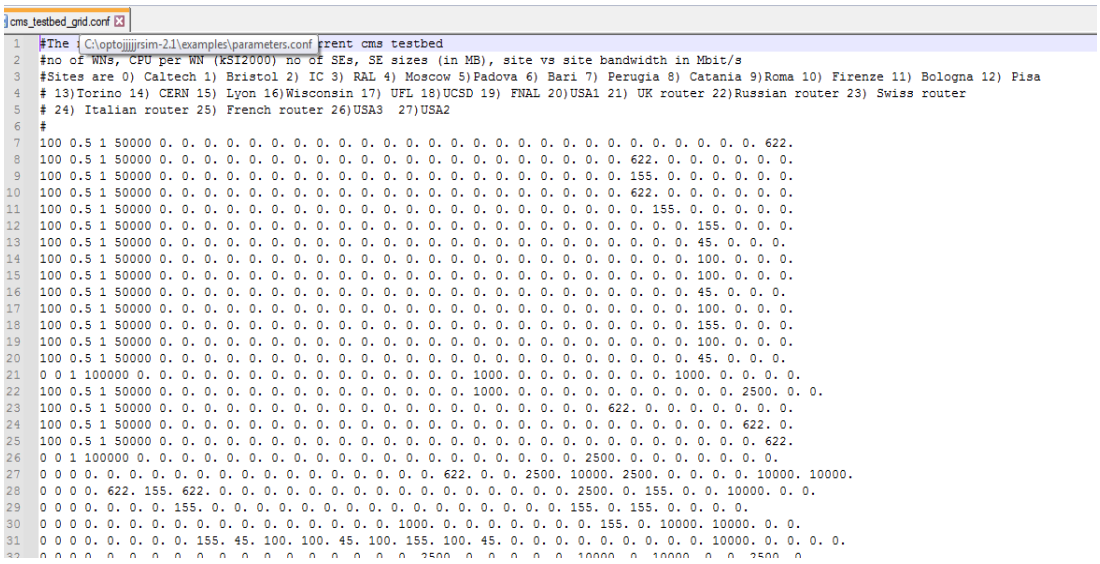

Figure 3.3Contenu du fichier cms_testbed_grid.conf

3.5.3. Fichier de configuration de paramètre

Dans ce fichier, nous avons spécifié :

- 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).

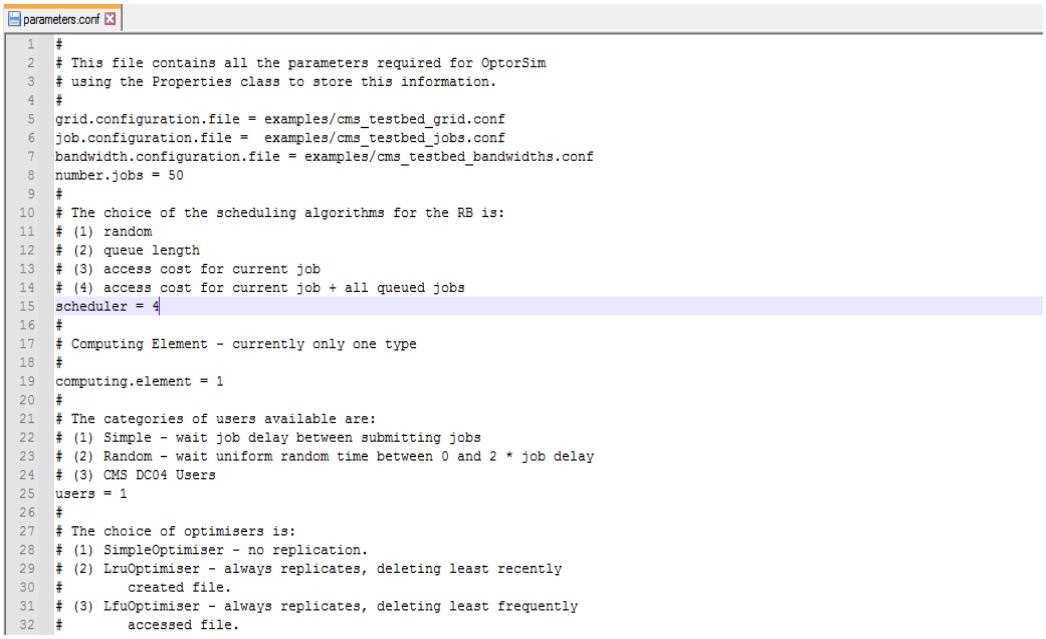

Figure 3.4Contenu du fichier parameters.conf

3.5.4. Fichier de configuration de la bande passante

Ce fichier de configuration est présenté sous forme d'une matrice Site-par-Site donnant, pour chaque paire de sites, le nom du fichier contenant les informations sur la bande passante qui les relie, ainsi que l'intervalle de temps qui sépare le site source du fuseau horaire référence.

Les lignes représentent les sites sources, alors que les sites destinataires sont alignés en colonnes et chaque entrée est présentée sous la forme, « filename », « time zone ».[10]

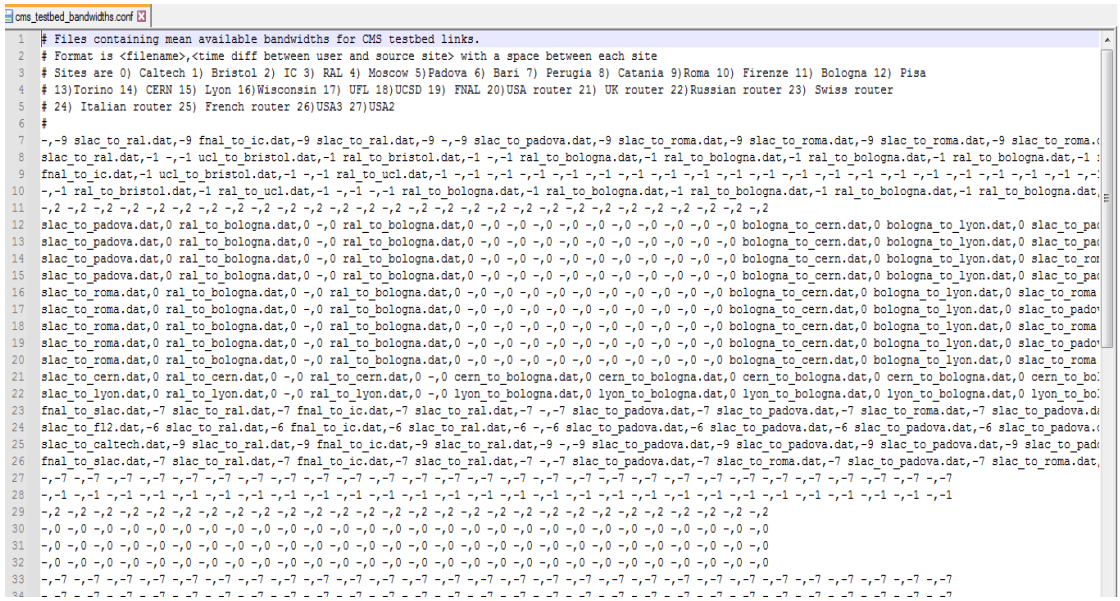

Figure 3.5Contenu du fichier Bandwidth Configuration File

3.6. Composants de l'interface principale(GUI)

L'interfaceprincipale est composée de trois parties : la partie hiérarchique, la partie statistique et la partie information. Ces parties peuvent être désactivées ou activées en utilisant le menu Display.

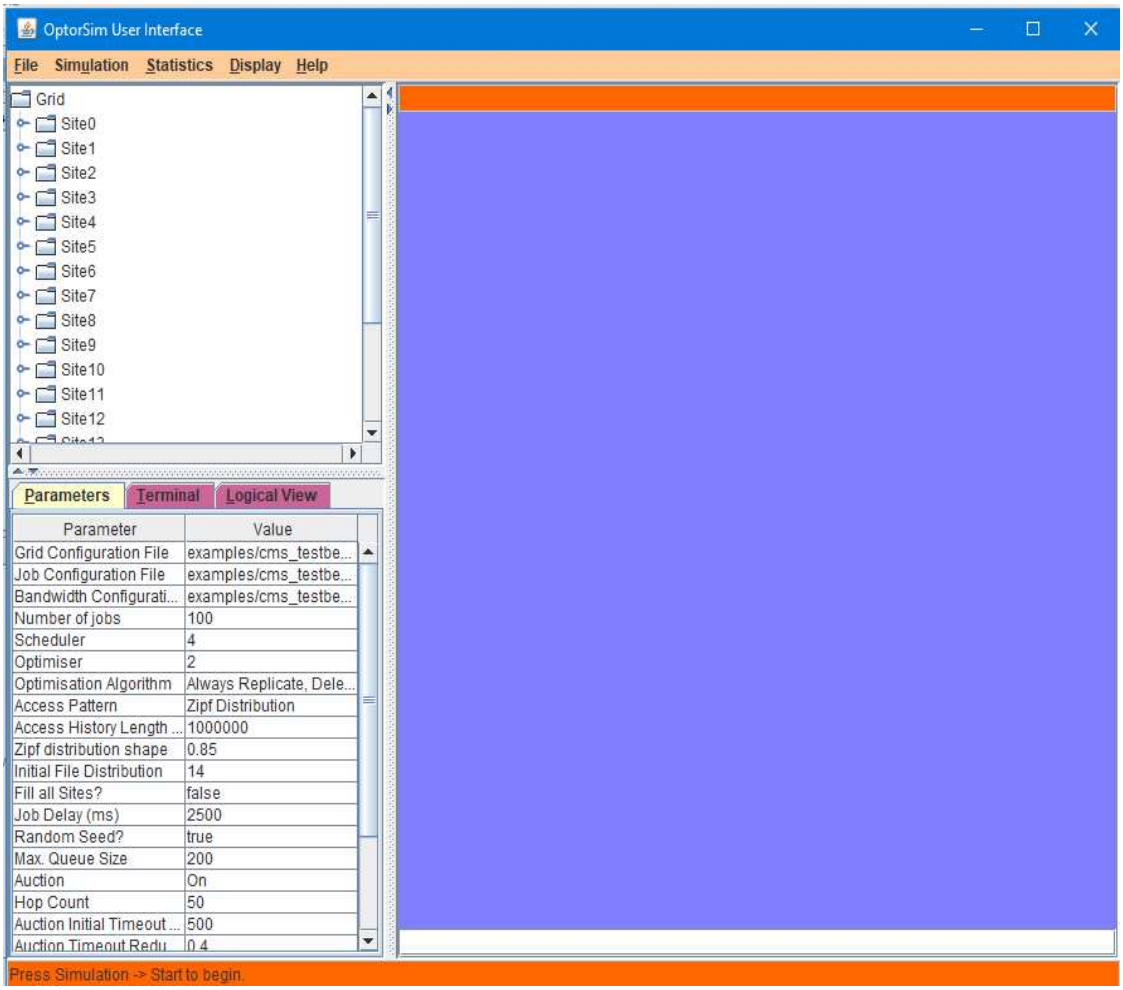

Figure 3.7Fenêtre principale GUI.

- **La Partie Statistique :** la partie droite de l'interface contient un nombre de tableaux de statistiques avec leurs graphes associés, concernant le nœud choisi. Le niveau de détail des résultats est réglable par l'utilisateur, ceci se fait grâce à la commande Change Sample Rate du menu Statistcs. Pour les histogrammes, le nombre de cases peut être changé en choisissant l'option appropriée dans le menu Statistics.
- **La Partie Hiérarchie** : la partie supérieure gauche de l'interface donne une vue hiérarchique sur les sites de la grille, et leurs ressources « CEs et SEs ».
- **La Partie Information :** cette partie contient trois panneaux : Terminal, LogicalView et Paramètres.

  Le panneau « Terminal » contient des informations sur le déroulement de la simulation.

Le panneau « LogicalView » schématise tous les transferts de fichiers entre les sites durant la simulation.

Le panneau « Paramètres » montre les paramètres de simulation.

3.8. Conclusion

Dans ce chapitre nous avonscité quelques simulateurs. Et nous avons présenté en détail le simulateur OptorSim, car ce dernier a été choisi pour simuler et tester les différentes stratégies de gestion de la cohérence des répliques.

Chapitre 4

Implémentation

4.1. Introduction

Dans ce chapitre sera consacré à l'implémentation des protocoles de gestion de la cohérence, où nous commencerons par la présentation les méthodes des protocoles du quorum à consensus, ROWA et ROWAA, suivie par la présentation des résultats obtenus des expérimentations qui nous permettra de vérifier le comportement de ces protocoles dans les systèmes distribués à grandeéchelle.

4.2. Environnement de travail

La simulation a été réalisée sous le système « Windows 7 » installé sur une machine dotée d'un processeur Intel Core i3 de 1.7 GHz.

L'implémentation de notre travaille ont été développés en langage Java en utilisant l'environnement de programmation « Eclipse ». Nous avons utilisé la version 2.1 d'OptorSim.

4.3. Notre approche

Nous avons appliqué les trois approches : Quorum à consensus, ROWA et ROWAA à l'aide de simulateur OptorSim. Puisque ce dernier est écrit en JAVA, cela nous a permis de le modifier pour intégrer nosméthodes.Ainsi nous pouvons comparerentre les différentes approches et en déduire la meilleure.

Dans ce qui suit, nous allons présenter les pseudos codes qui représentent les méthodes de lecture et d'écriture pour les protocoles du quorum à consensus, ROWA et ROWAA.

4.4. Les algorithmes

4.4.1. Lecture pour Quorum à consensus

```
m : élément /* copie */
R : quorum de lecture
T : entier /*toutes les copies*/
nmbr_rep, version: entier
début
envoyer une demande de lecture à l'ensemble des copies T
nmbr_rep← 0
Tant que nombre_rep< R fait
    Attendre (reponse) /*attendre quelque temps la réponse sinon échec*/
    Verrouiller (reponse.élément) /*verrouiller l'élément pour la lecture*/
nmbr_rep← nombre_rep+1
si version <reponse.version
version←reponse.version
m←reponse.version /*enregistrer l'élément contenant la dernière version*/
fin si
fin Tant que
accéder à la copie sauvegardée dans m
lire la copie
libérer les copies
fin
```

4.4.2. Ecriture pour Quorum à consensus

```
m : élément /* copie */
W : quorum d'écriture
T: entier /*toutes les copies*/
nmbr_rep , version :entier

debut
envoyer une demande d'écriture à l'ensemble des copies T
nombre_rep←0
Tant que nmbr_rep<W fait
    Attendre (réponse) /*attendre quelque temps la réponse sinon échec*/
    Verrouiller (response.element) /*verrouiller l'élément pour l'écriture */
nmbr_rep← nombre_rep+1
si version <reponse.version
version←reponse.version
m←reponse.version /*enregistrer l'élément contenant la dernière version*/
fin si
fin Tant que

accéder à la copie sauvegardée dans m
écrire sur la copie
propager les modifications sur toutes les copies de l'ensemble W
libérer les copies
fin
```

4.4.3. Lecture pour ROWA

```
T: entier /*toutes les copies*/
debut
envoyer une demande de lecture à l'ensemble des copies T
    Attendre (réponse) /*attendre quelque temps la réponse sinon échec*/
verrouille(response.element) /*verrouille l'élément pour la lecture*/

lire la copie
libérer la copie
fin
```

4.4.4. Ecriture pour ROWA

```
m : élément /* copie */
T: entier /*toutes les copies*/
nmbr_rep , version :entier
debut
envoyer une demande d'écriture à l'ensemble des copies T
nombre_rep← 0
pour toutes les copies de l'ensemble T faire
    Attendez (réponse) /*attendez quelque temps la réponse*/
    Verrouille (reponse.element) /*verrouille l'élément pour l'écriture*/
nmbr_rep←nombre_rep+1
si version <reponse.version alors
version←reponse.version
m←reponse.version /*enregistrer l'élément contenant la dernière version*/
fin si
```

```
fin pour
accéder à la copie sauvegardée dans m
écrire sur la copie
propager les modifications sur nmbre_resp copies disponibles
libérer les copies.
fin
```

4.4.3. Lecture pour ROWAA

```
T: entier /*toutes les copies*/
debut
envoyer  une demande de lecture à l'ensemble des copies T
    Attendre (réponse) /*attendre quelque temps la réponse sinon échec*/
verrouille(response.element) /*verrouille l'élément pour la lecture*/
lire la copie
libérer la copie
fin
```

4.4.4. Ecriture pour ROWAA

```
m : élément /* copie */
T: entier /*toutes les copies*/
nmbr_rep , version :entier
debut
envoyer une demande d'écriture à l'ensemble des copies T
nombre_rep← 0
pour toutes les copies de l'ensemble T faire
```

```
    Attendez (réponse) /*attendez quelque temps la réponse*/
    Verrouille (reponse.element) /*verrouille l'élément pour l'écriture*/
nmbr_rep←nombre_rep+1
si version <reponse.version alors
version←reponse.version
m←reponse.version /*enregistrer l'élément contenant la dernière version*/
fin si
fin pour
accéder à la copie sauvegardée dans m
écrire sur la copie
propager les modifications sur nmbre_resp copies disponibles
libérer les copies.
fin
```

4.4.5. Mise à jour asynchrone des copies divergents pour ROWAA

```
T: entier /*toutes les copies*/
nmbr_rep , version :entier
debut
pour toutes les copies de l'ensemble T faire
   Attendez (réponse) /*attendez quelque temps la réponse*/
si version >reponse.version alors
verrouille(response.element) /*verrouille l'élément pour l'écriture*/
mettre à jour la copie
fin si
fin pour
fin
```

4.5. Implémentation

4.5.1. Fenêtreprincipale

Cette fenêtre del'interface principale d'OptorSim, où nous avons ajoutél'onglet « Calcul » composé de sous menu d'une seule propriété : Distribution.

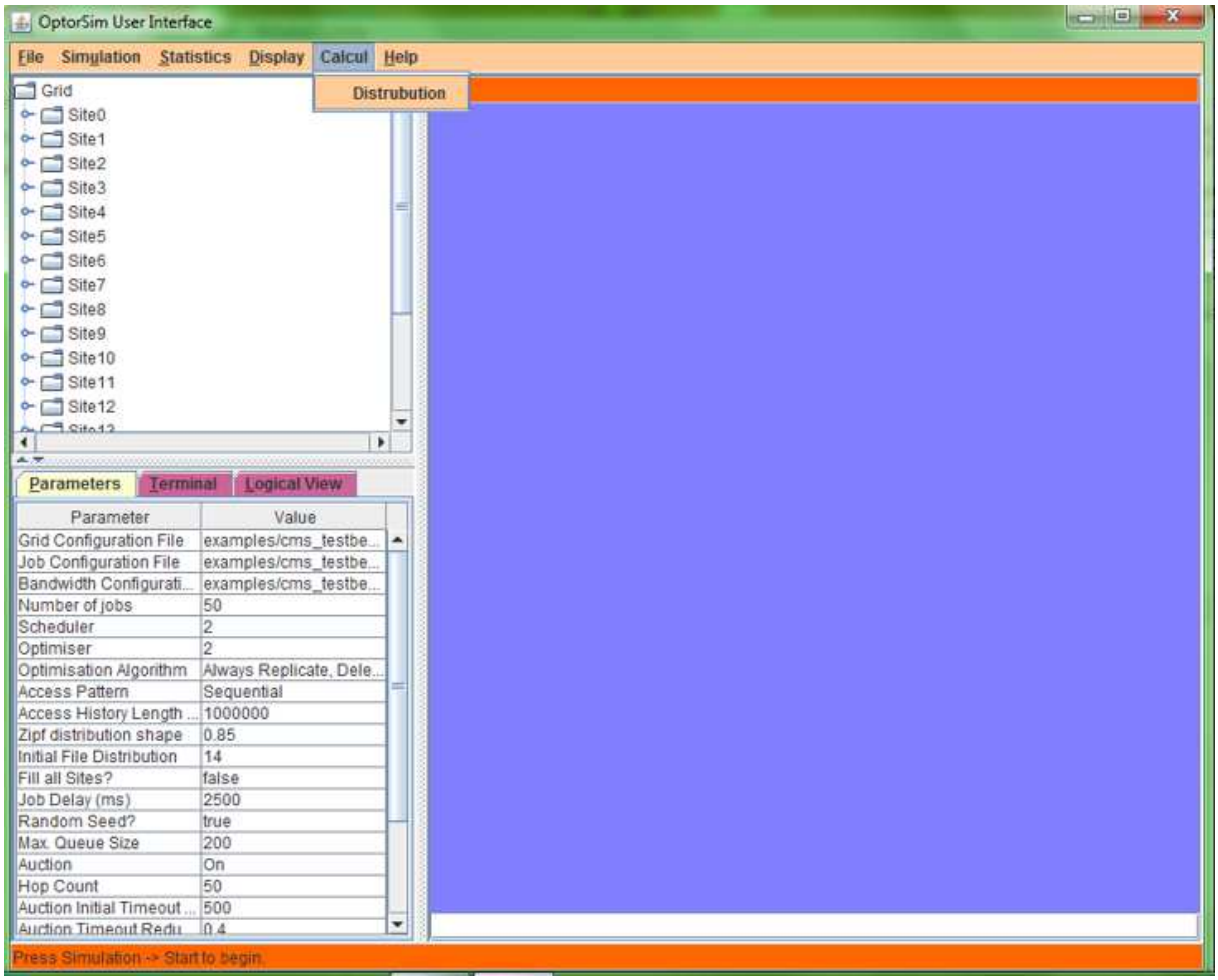

Figure 4.1 Fenêtre principale d'OptorSim.

4.5.2. Fenêtre distribution

La figure 4.2 nous permet de calculer le nombre de copies à partirdu tableau pour déterminer le nombre global des requêtes et nombre d'écriture et nombre de lecture.

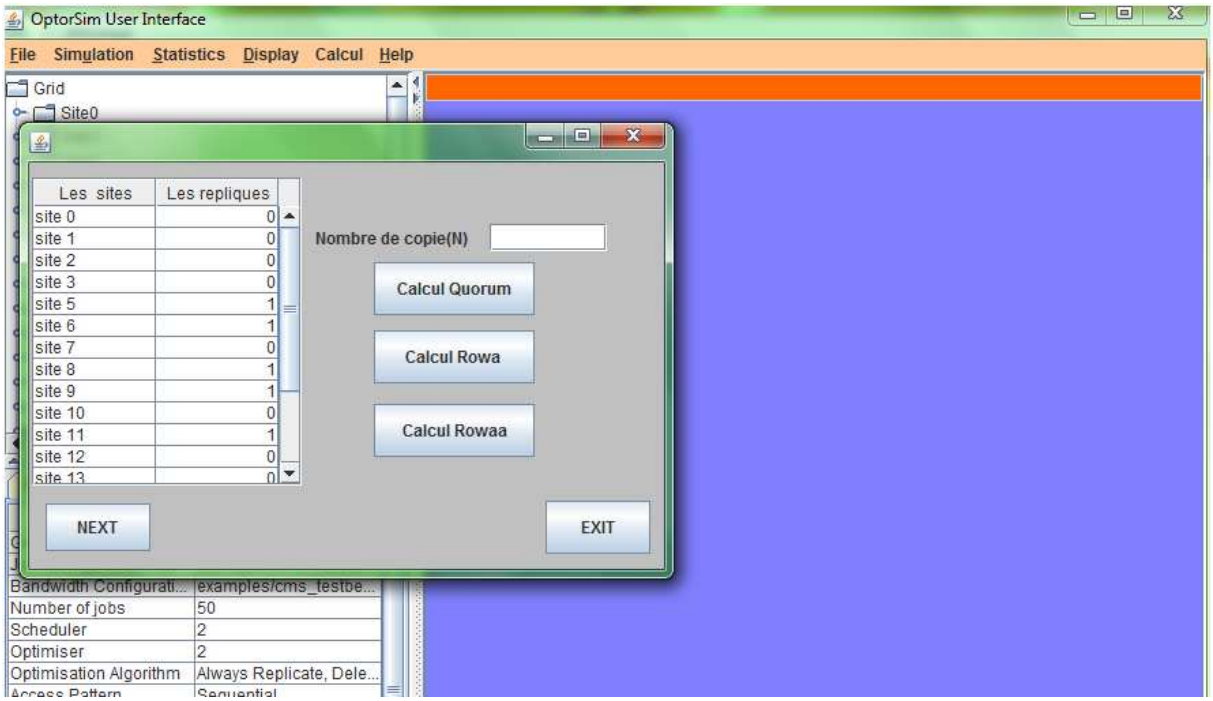

Figure 4.2 Fenêtre d'onglet Distribution.

4.5.3. Fenêtre Quorum

Cette fenêtre permet d'afficherle quorum à consensus de lecture et d'écriture à partir de nombre de copie de la fenêtre précédente.

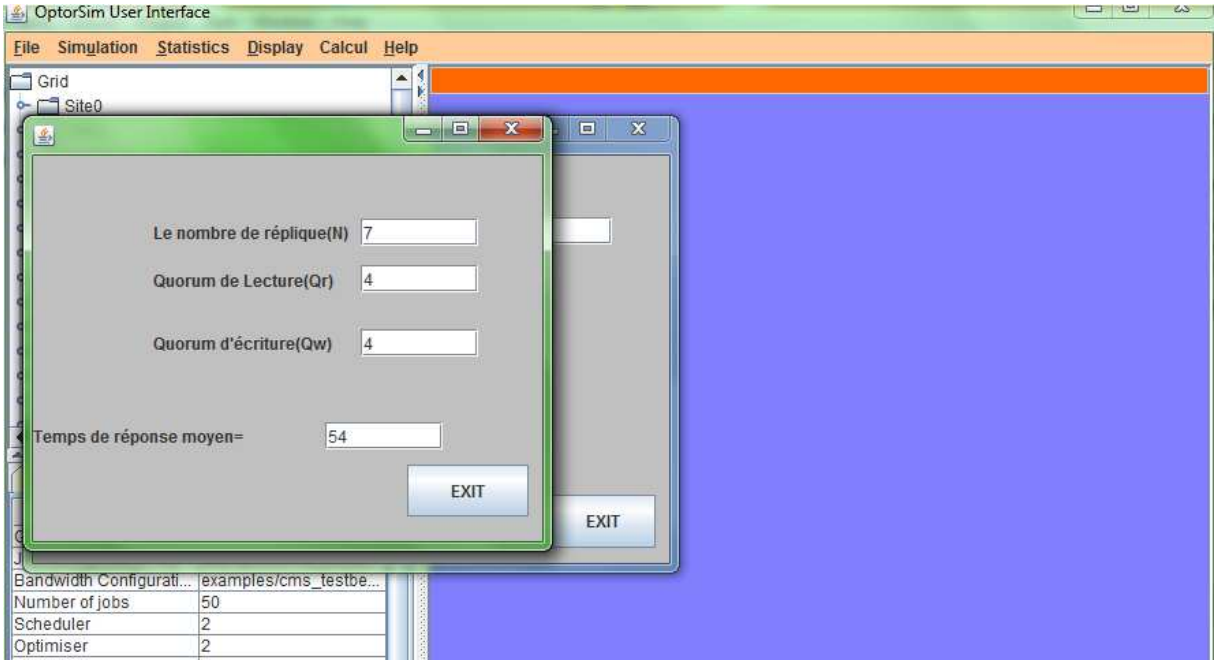

Figure 4.3 Fenêtre de Quorum

4.5.4. Fenêtre ROWA

Cette fenêtre permet d'afficherROWA à partir de nombre de copie de la fenêtre précédente.

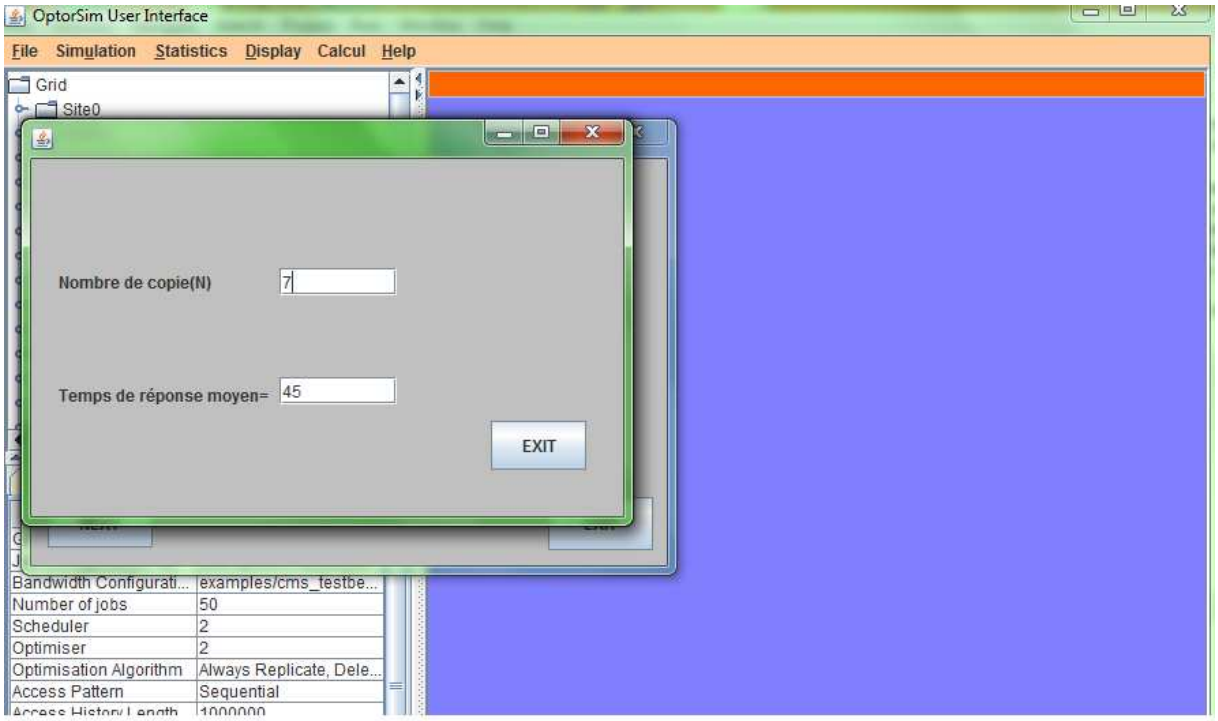

Figure 4.4 Fenêtre ROWA.

4.5.5. Fenêtre ROWAA

Cette fenêtre permet d'afficherROWAA à partir de nombre de copie de la fenêtre précédente.

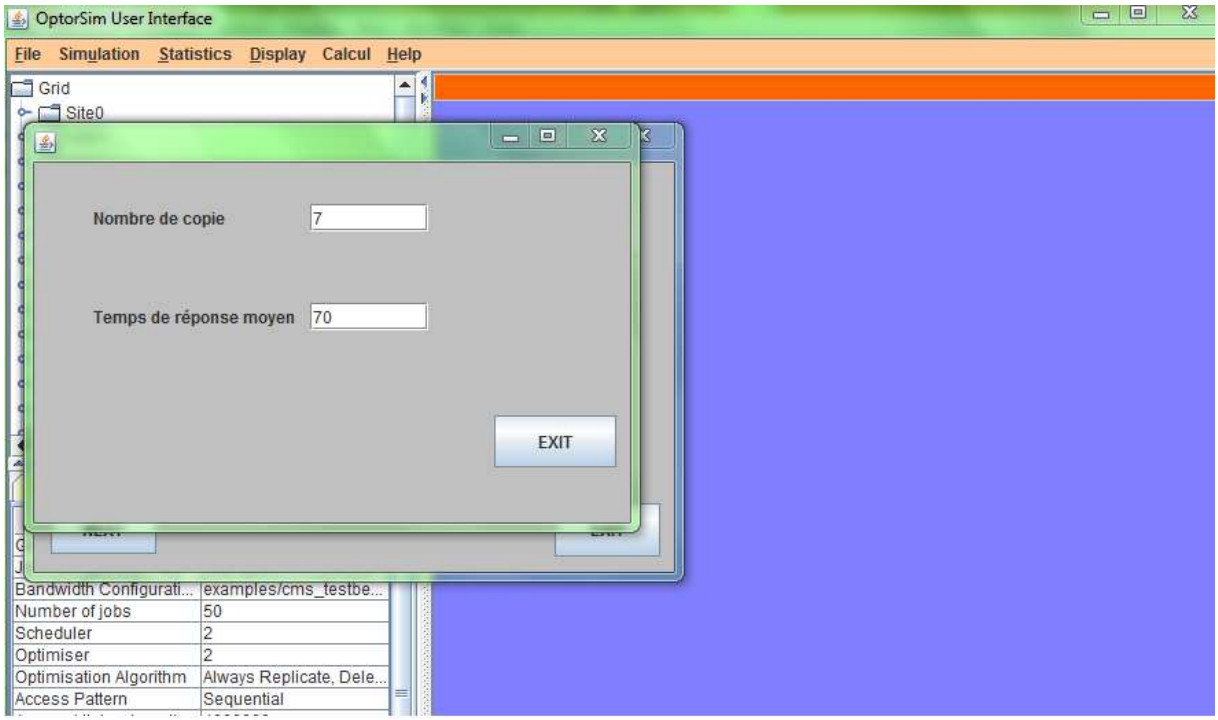

Figure 4.5Fenêtre ROWAA.

4.5.6. Fenêtre Résultats

La fenêtre suivante affiche les informations et les résultats obtenus après l'application de l'une des trois approches (quorum à consensus, ROWA et ROWAA)

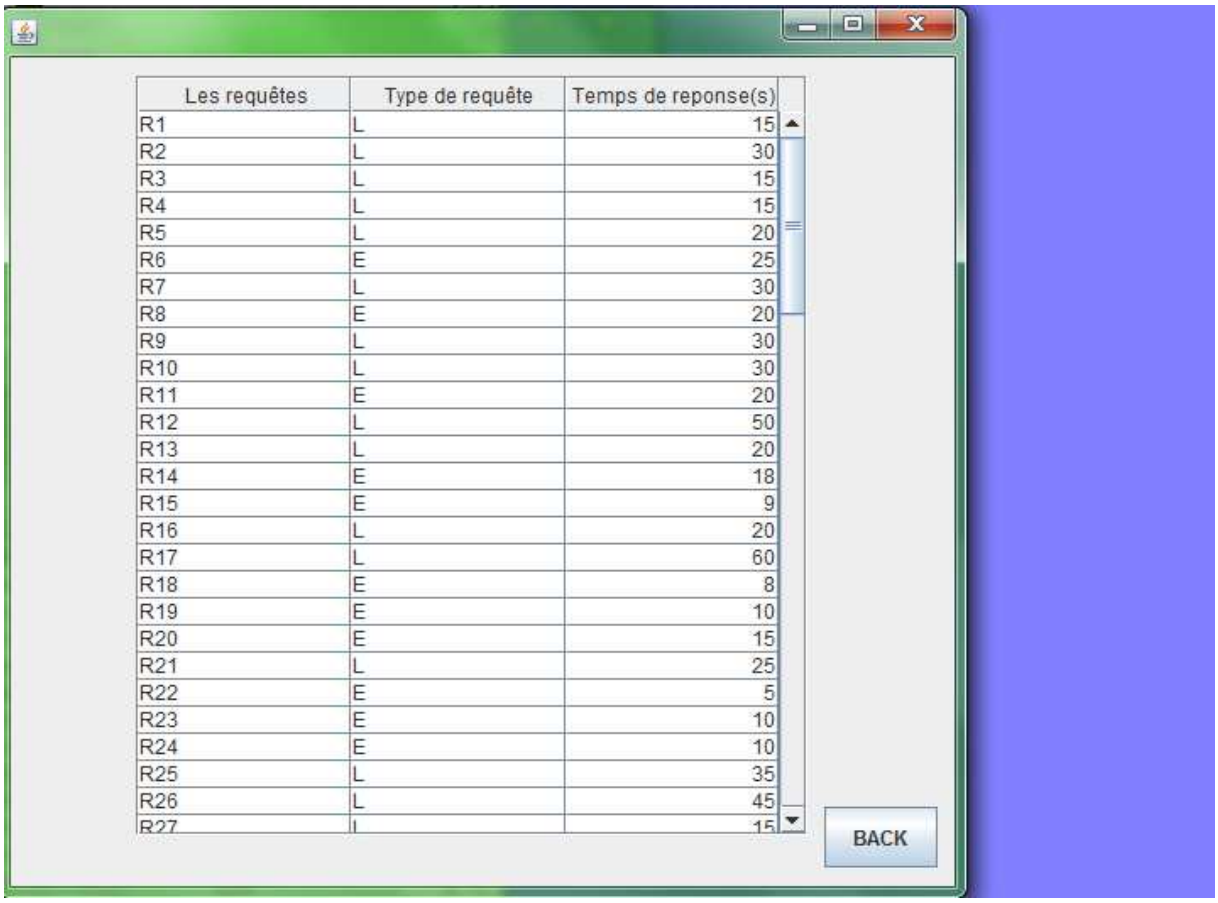

Figure 4.6 Fenêtrerésultats.

4.6. Expérimentation et résultats

Nous allons présenter les résultats de certaines expérimentations selon le nombre de copies et le types de requêtes (lecture / écriture). Ces expérimentations ont été effectuées selon les paramètres suivant :

Nombre de copies	14
Taille de données	500MB
Nombre de requêtes	100

Tableau 4.1 paramètres de simulation.

4.6.1. Expérimentation par type de requêtes

Les résultats de cette expérimentation représentent les temps de réponses moyennes par nombre de requêtes de lecture émises (nombre de requêtes de lecture émises parmi l'ensemble des requêtes).

Selon le tableau ci-dessous, nous constatons que le temps de réponse moyen de ROWA est le meilleur.

Les approches	Quorum à consensus	ROWA	ROWAA
Temps de réponse moyen(s)	30	28.125	30.625

Table 4.2 temps de réponse moyen par type de requêtes.

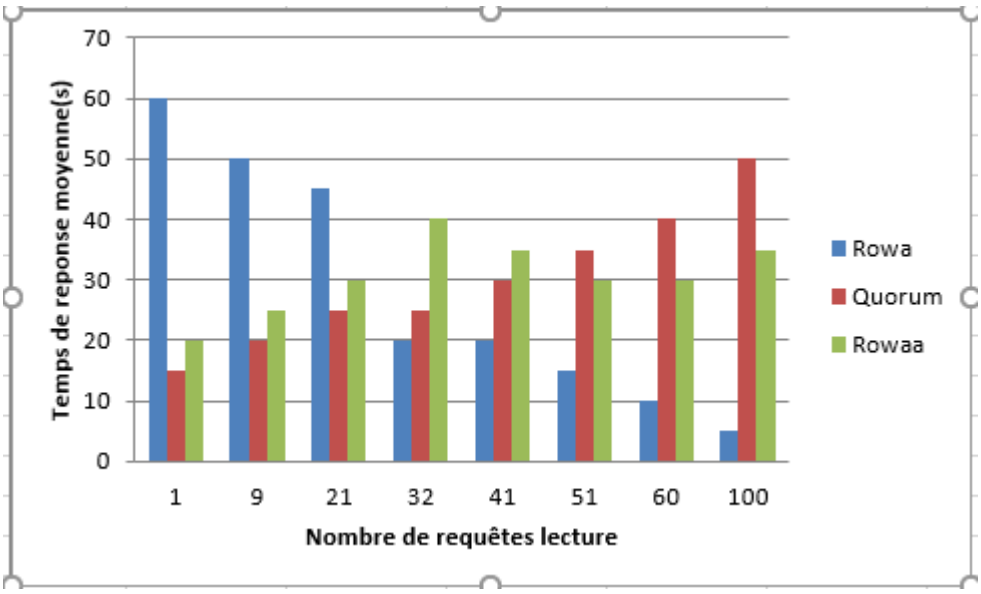

Figure 4.5 Temps de réponse partype de requêtes.

La figure 4.5 représente le temps de réponse de ces approches (Quorum à consensus,ROWA et ROWAA) par le nombre de requête de lecture.

Nous remarquons que le temps de réponse diminue avec l'augmentation de nombre des requêtes de lecture de l'approche ROWA, et le contraire pour l'approche Quorum à consensus, et pour ROWAA le temps de réponsevarie.

Nous déduisons d'après ces résultatsque ROWA est meilleure si le nombrede requêtes de lecture est élevé, et que Quorum à consensus est intéressant si le nombrede requêtes d'écriture est grand.

4.6.2. Expérimentation par nombre de copie

Les résultats de cette expérimentation représentent les temps de réponse moyen par nombre de copie (temps de réponse moyen des différentes copies).

Taille de données	500MB
Nombre de requêtes	100

Tableau 4.3 paramètres de simulation.

Les résultats obtenus sont modélisés par l'histogramme de la figure 4.6

Selon le tableau ci-dessous, nous constatons que le temps de réponse moyen de Quorum à consensus est le meilleur.

Les approches	Quorum à consensus	ROWA	ROWAA
Temps de réponse moyen	14.4	22	24

Table 4.4 temps de réponse moyen par nombre de copie.

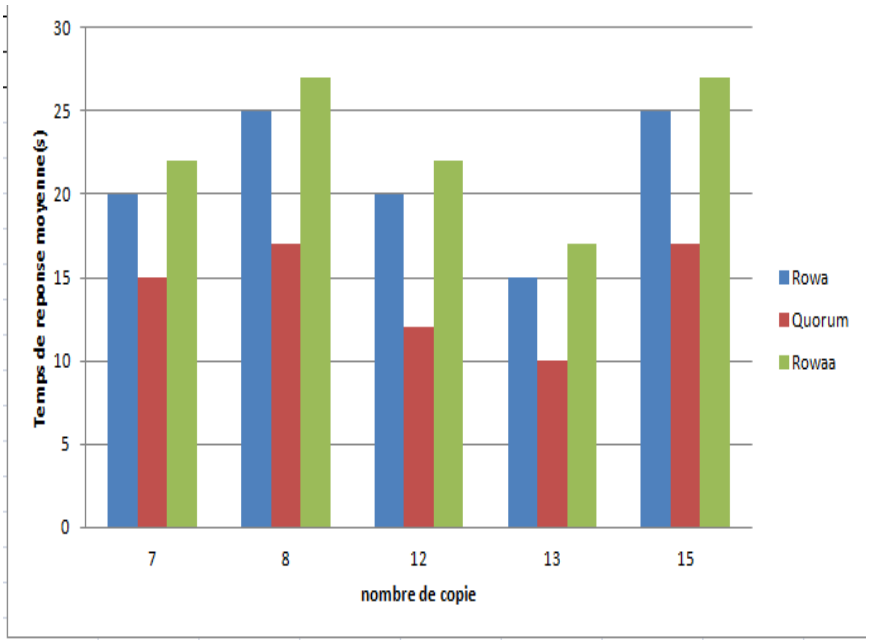

Figure 4.6 Le temps de réponse moyen par nombre de copie.

La figure 4.6 représente le temps de réponse moyen de ces approches (Quorum à consensus, ROWA, ROWAA) par le nombre de copie.

Nous remarquonsque la meilleure approche est le Quorum à consensus, car ilprend le moins de temps.

4.7. Discussion des résultats

Dans les approches pessimistes (Quorum à consensus,ROWA et ROWAA) le temps de réponse moyen des requêtes tend à augmenter avec l'augmentation du nombre de requêtes.

De plus, cette augmentation est très importante dans l'approche ROWA par rapport à l'approche Quorum à consensus, ce qui réduit considérablement ses performances. Sur la base des résultats de ces simulations numériques, nous avons pu évaluer que le gain des approches ROWA et Quorum à consensusest très important.

Nous pouvons conclure que ces approches peuvent être plus au moins performantes pour les systèmes distribuésà grande échelle.

4.8. Conclusion

Dans ce chapitre nous avons décrit notre simulateur où nous avons intégré les approches du Quorum à consensus, ROWA et ROWAA.

Après plusieurs tests sur ces approches à l'aidedu simulateur OptorSim.Nous pouvons déduire que ces approches peuvent être, selon les cas, performantes pour les systèmes à grande échelle.

Conclusion Générale

Un système distribué est un système qui s'exécute sur un groupe de machines sans mémoire partagée, mais les utilisateurs le voient comme une machine, le principe des systèmes distribués sont d'améliorer le partage des ressources. Les systèmes distribués sont des systèmes très instables, dans lesquels les nœuds sont susceptibles d'être connectés ou déconnecté, ses données peuvent alors disparaître, entraînant sa perte.

Parmi les solutions proposées pour résoudre le problème de la perte de données, nous avons notamment découvert une solution de réplication qui réplique et place les données dans différents sites. Cette technique de réplication présente à la fois des avantages et des inconvénients. Ensuite, la cohérence est obtenue en définissant un protocole de cohérence, dans lequel lors des opérations de lecture et d'écriture de données, un sous-ensemble de nœuds appelé (Quorum à consensus) hébergent la même copie de données sera traité pour améliorer la cohérence et la disponibilité de la copie de données.

Pour cela, nous avons présenté les notions des systèmes distribués à grand échelle. Puis nous avons détaillé les principes de la réplication et de la cohérence. En effet, dans notre travail nous nous sommes intéressés aux protocoles de gestion des répliques dansles systèmes distribués à grand échelle.

Pour pouvoir étudier le comportementdes trois stratégies de gestion de lacohérence(Quorum à consensus, ROWA et ROWAA) dans un système distribué à grande échelle, nous avons utilisé le simulateur OptorSim, sur lequel nous avons intégré les méthodes de ces trois protocoles.

Cette intégration nous a permis de comparer entre ces trois protocoles et déterminer dans quels cas ils peuvent être performantes.

Bibliographie

[1] SALEH YOUSSEF : Systèmes Distribués, Université Cameron, 2016.

[2] Tshiamua, J. :Mise en oeuvre d'un système distribué pour l'identification et le suivi du casier judiciaire. Université pédagogique nationale, 2016.

[3] Babahenini Mohamed Chaouki, Introduction aux Systèmes répartis, Université de Biskra.

[4] DJEBBARA Mohamed Rédha, GESTION DE REPLIQUES DANS LES CLOUDS COMPUTING, 2018-2019.

[5] SENHADJ sarra, Gestion de la cohérence des données répliquées sur un environnement de grille, 2014, PP 17-18-22.

[6] KOUIDRI SIHAM, Gestion de la cohérence des répliques aux fautes dans une grille de données, PP 19 - 24.

[7] François Verndat, Introduction à l'Algorithme Distribuée/Répartie, 2015.

[8] BOUHARAOUA FAROUK, « Stratégies de réplication dynamique dans les grilles de données ». Faculté des sciences et sciences de l'ingénieur, Département d'informatique, Université de AbdElhamid Ibn, Mostaganem, juin 2007.

[9] BELATRECHE MANSOUR, « Placement des répliques dans les grilles de données hiérarchiques » Faculté des Sciences Exactes et de l'Informatique, Département d'Informatique,2013.

[10]MAGHNI SANDID ZOULIKHA,«La réplication dans les grilles de données »
Université des sciences et de la technologie d'Oran-Mohamed Boudiaf, UstoMB, département d'Informatique, 2011.

Documents web

[9] SlidePlayer, https://slideplayer.fr/slide/4205514 .