N° D'ORDRE:

N° DE SERIE:

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ ECHAHID HAMMA LAKHDAR EL OUED
FACULTÉ DES SCIENCES ET TECHNOLOGIES
Mémoire de fin d'étude
Présenté pour l'obtention du diplôme de

MASTER ACADEMIQUE

Domaine: Mathématiques et Informatique
Filière: Informatique
Spécialité: Systèmes distribués et intelligence artificielle

Présenté par:

- KEBSA Said
- BOURAHALA Rida

Thème

L'équilibrage de charge pour les requêtes de lecture dans un système de réplication de données

Session de juin
Devant le jury composé de**:**

Mr. KHELAIFA Abdennacer	MAA	Univ. Hamma Lakhdar	Président
Mr. NAWI Mohammad Anwar	MAA	Univ. Hamma Lakhdar	Examinateur
Mr. MUFTAH Mohammed Sharaf al-Din	MAA	Univ. Hamma Lakhdar	Examinateur

Année universitaire 2015 – 2016

N° D'ORDRE:

N° DE SERIE:

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ ECHAHID HAMMA LAKHDAR EL OUED
FACULTÉ DES SCIENCES ET TECHNOLOGIES

Mémoire de fin d'étude
Présenté pour l'obtention du diplôme de

MASTER ACADEMIQUE

Domaine: Mathématiques et Informatique
Filière: Informatique
Spécialité: Systèmes distribués et intelligence artificielle

Présenté par:

- KEBSA Said
- BOURAHALA Rida

Thème

L'équilibrage de charge pour les requêtes de lecture dans un système de réplication de données

Session de juin

Devant le jury composé de:

Mr. KHELAIFA Abdennacer	MAA	Univ. Hamma Lakhdar	Président
			Examinateur
			Examinateur

Année universitaire 2015 – 2016

Remerciements

Le travail présenté dans ce mémoire a été réalisé au département d'informatique de 'université Echahid Hamma Lakhdar à El Oued (Algérie). A travers ces quelques lignes nous voudrons évoquer tous ceux qui, par leurs conseils, par leur compétence ou tout simplement par leur chaleur humaine, ont contribué au bon déroulement de ce mémoire.

Nous tenons, tout d'abord, à remercier notre **DIEU** qui a bien voulu nous donner la force pour effectuer ce présent travail.

Nous remercions notre encadreur Mr. Abdennacer KHELAIFA d'avoir accepté de diriger ce travail et d'avoir mis à ma disposition tous les moyens qui nous ont permis de mener à terme cette étude et aussi pour ses précieux conseils et ses encouragements.

Nous remercions tous les membres du jury qui ont accepté d'évaluer notre travail.

Nous remercions nos très chères familles pour leurs soutiens et leurs patiences.

Nos remerciements vont également à tous mes enseignants qui ont contribués à notre formation, et à tous ceux qui nous ont aidé de près ou de loin.

Dédicace

Je dédier ce modeste travail :

Ma source éternelle d'inspiration, de motivation et de bénédiction … Ma mère
Mon maître de passion, de patience et de persévérance … Mon père
Mes sœurs et mes frères
Mes amis
Ma promotion

BOURAHALA Rida

Dédicace

Je dédier ce modeste travail :

A

Ma source éternelle d'inspiration, de motivation et de bénédiction … Ma mère
Mon maître de passion, de patience et de persévérance … Mon père
Mes sœurs et mes frères
Mes amis
Ma promotion

KEBSA Said

Résumé

La réplication des données permet d'offrir en même temps, une haute disponibilité de données, une répartition de charge entre les nœuds du système et une accélération du temps de traitement des requêtes (de consultation) en les exécutants en parallèle. Cependant, la charge sur les nœuds du système reste déséquilibre en dispose des nœuds plus chargée que les autres. L'objectif de ce travail est de proposer une solution efficace d'équilibrage de charge pour les requêtes de lecture en distribuant la charge de traitement sur plusieurs serveurs ce qui permettre de diminuer la charge sur un seul nœud et de gagner du temps d'exécution des requêtes.

Mots clés: système de réplication, algorithme de répartition, équilibrage de charge, requête de lecture.

Abstract

Data replication allows, at the same time, a high availability of data, a load distribution between the nodes of the system and an acceleration of the processing time of queries (read queries) by executing them in parallel. However, the load on the system nodes remains destabilized there are a nodes more charged than others. The goal of this work is to propose an effective solution for load balancing the read queries by distributing the processing load across several servers, which will reduce the load on one node and minimize the queries running time.

Keywords: replication system, algorithm of distribution, load balancing, read query.

ملخص

نظام النسخ المتماثل للبيانات هو نظام يسمح لنا في نفس الوقت بضمان وفرة عالية للمعطيات وبتجزئة وتقسيم الحمولة بين عقد النظام وكذا بزيادة سرعة معالجة الاستعلامات (استعلام القراءة) وذلك بتنفيذها في وقت واحد، إلا أن الحمولة على مستوى عقد النظام تبقى غير متوازنة حيث نجد هذه الحمولة مختلفة من عقدة إلى أخرى في نفس النظام. لذا فالهدف من العمل المقدم في هذه المذكرة هو اقتراح حلول فعالة لموازنة حمولة استعلامات القراءة وذلك بإعادة تقسيم هذه الحمولة على عدة خوادم، الشيء الذي يسمح لنا من جهة بتخفيض الحمولة على مستوى كل عقدة من عقد النظام وبتحفيظ زمن تنفيذ الاستعلام من جهة اخرى.

الكلمات المفتاحية :نظام النسخ المتماثل ,خوارزمية للتوزيع ,موازنة الحمولة ,استعلام القراءة .

Sommaire

Chapitre 2: La réplication dans les bases de données

Chapitre 3: Proposition d'une approche pour l'équilibrage de charge

Chapitre 4: Implémentation du système d'équilibrage de charge

Table des figures

Introduction générale

Introduction général

L'évolution des techniques informatiques depuis les vingt dernières années nous oblige à développer de plus en plus la théorie de la distribution des données. L'objectif de la distribution des données est de partager les données de travail entre un ensemble de ressources. Cet objectif peut être accompli à différents niveaux de finesse. Cependant, le problème de la distribution de charge est une partie de la distribution de données, ce problème souvent décrite dans la littérature comme un problème d'équilibrage de charge ou de partage de charge. Nous nous intéressons dans le cadre de ce travail au problème d'équilibrage de charge.

Généralement, l'importance d'équilibrage de charge se retrouve lors de la mise en place de services amenés à croître. Il faut s'assurer que la capacité à monter en charge soit la plus optimale possible afin d'éviter toute dégradation que ce soit en terme de performances ou de fiabilité lors d'affluences importantes.

Particulièrement, l'équilibrage de charge consiste à effectuer une distribution des tâches à des machines de façon intelligente, son but alors est d'optimiser l'utilisation des processeurs en spécifiant la localité des tâches. Autrement dit, dans ce travail, l'équilibrage de charge doit minimiser le temps d'exécution d'un ensemble de requêtes données, ce qui revient souvent à maintenir une charge équivalente sur l'ensemble des serveurs au avers de différents algorithmes tels que (Round Robin, Random, Least Resources...) développés dans ce sens.

Dans le cas des requêtes de lecture, il n'y a pas de modification des données mais les traitements peuvent être longs et porter sur une grande masse de données. On comprend donc dans le cas d'un nombre important de requêtes peut emboliser partiellement (ou complètement) le serveur. Il existe plusieurs solutions pour palier à ce genre de problèmes, la réplication des données est l'une de ces solutions, qui est un processus de partage d'informations pour assurer la cohérence des données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, la tolérance aux pannes, ou la disponibilité. On parle de réplication de données si les mêmes données sont dupliquées sur plusieurs serveurs.

L'objectif de ce travail est de développer un algorithme permet d'équilibrer les charges sous forme requêtes de lectures dans un système de réplication de données.

Ce mémoire résume le travail que nous avons réalisé, il est organisé en quatre chapitres, que nous décrivons dans ce qui suit.

Dans le premier chapitre, nous allons donner un bref aperçu sur les systèmes d'équilibrage de charge, à savoir, classification des systèmes d'équilibrage de charge, catégories des approches d'équilibrage de charge, propriétés et performances d'un d'équilibrage de charge et à la fin de ce chapitre, nous allons rappeler les déférentes algorithmes utilisés par un système d'équilibrage de charge.

Nous allons présenter dans le deuxième chapitre, de manière succincte les notions de base sur les bases de données distribués, la redondance des données, et particulièrement les systèmes de réplication de données.

Dans le troisième chapitre, nous allons donner l'architecture globale d'équilibrage de charge pour les requêtes de lecture dans le système de réplication, également, nous allons donner un schéma et une explication détaillée de cette architecture afin de l'exploiter lors de la mise en œuvre de notre application.

Dans le chapitre 4, nous allons réaliser notre travail à travers une application, dans un premier temps, nous allons présenter les outils d'implémentation utilisés dans cette application, tels que MySql, Java et Wampserver. Ensuite, nous allons configurer le système de réplication de donnée sous MS-DOS. Finalement, nous allons exécuter l'application sous l'environnement JAVA dans un réseau local.

Enfin, nous nous allons terminer ce mémoire par une conclusion qui résume notre travail et par une proposition de quelques perspectives de recherche.

Chapitre 1

Les principaux concepts d'équilibrage de charge

1.1.Introduction

Ce premier chapitre est consacré à une présentation générale de la technique de l'équilibrage de charge, notamment dans les bases de données. Nous allons présenter dans un premier temps la définition, la classification ainsi que les mécanismes des systèmes de l'équilibrage de charge. Ensuite, nous allons présenter les principaux problèmes de l'équilibrage de charge. Nous allons aborder également quelques critères de performances. En fin, nous allons présenter les différents algorithmes de l'équilibrage de charge.

1.2.Qu'est-ce qu'un équilibrage de charge?

L'équilibrage de charge (Load Balancing en anglais) par définition est l'ensemble de techniques permettant d'allouer des données ou des tâches au départ et de les distribuer entre différents machines d'un groupe de façon intelligente afin de minimiser leur temps de traitement. Cette procédure nécessite l'intégration d'un dispositif entre les serveurs et les utilisateurs de la ressource. La tâche principale de ce dispositif est de rediriger les utilisateurs de services en fonction de l'état d'occupation des serveurs.

En revanche, la technique de équilibrage de charge permette à la fois de répartir charge trop importante d'un service sur plusieurs serveurs, et de réduire l'indisponibilité potentielle de ce service qui pourrait provoquer une panne (logiciel ou matériel) d'un unique serveur [11]. Cette technique est représentée par la figure 1, où la charge est repartitionnée entre deux serveurs

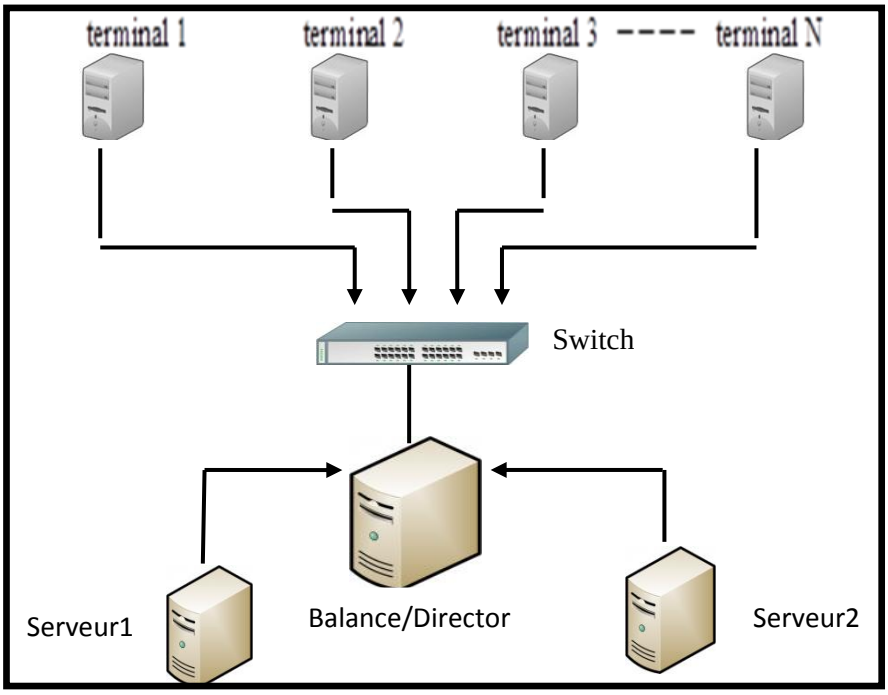

Figure 1.1: Répartition de charge entre deux serveurs

1.3. Classification des systèmes d'équilibrage de charge

Les systèmes d'équilibrage de charge peuvent être classifiées en de deux éléments essentiels, il s'agit de politiques et mécanismes: Les politiques considèrent l'ensemble des choix à effectuer pour déterminer les objectifs du système de distribution de la charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et également fournissent les informations exigées par les politiques. Dans la suite de cette section, nous allons détailler ces deux classes de systèmes d'équilibrage de charges.

1.3.1. Les Politiques: On distingue plusieurs types de politiques à savoir:

1.3.1.1. Politique de participation: Le but de cette politique consiste à déterminer si un site ou une ressource participe ou non au transfert de tâches (où en général à l'équilibrage de charge), comme source (site surchargé) ou comme receveur (site sous-chargé). Notons qu'il existe plusieurs critères de participation, nous présentons dans cette partie les deux critères les plus utilisés dans la littérature, à savoir: la politique de seuil et celle de possession [4].

- **Politique de seuil:** Leur fonction est de déterminer si une ressource est en état sous-chargée ou surchargée, selon que sa charge actuelle est en dessous ou en dessus du seuil. Sa forme la plus commune est de considérer les deux seuils.
- **Politique de possession:** Notons que la possession d'une ressource est considérée, dans certains systèmes d'équilibrage de charge. Dans ce type de politique, la ressource accepte uniquement les tâches des autres ressources (si elle est inactive). de surcroit, elle est toujours préparée le transfert des tâches aux ressources inactives, et à révoquer sa participation et rejeter également les tâches étrangères dans le cas où la tâche considérée est activée.

1.3.1.2. Politique de sélection de la localisation: La fonction de cette politique est de trouver, pour un site donné, une paire de ressources participantes, (source ou receveur), entre lesquelles la tâche sera transférée. Cette décision peut être prise de manière centralisée ou indépendante dans chaque ressource participante [4].

1.3.1.3. Politique de sélection des tâches à transférer: Après avoir que les politiques de participation et de localisation décident qu'un site Si est source et qu'un autre site Sj est receveur, alors que la politique de sélection des tâches à transférer détermine les tâches qui doivent être transférées de site Si vers le site Sj.

1.3.2. Les mécanismes: On distingue:

1.3.2.1. Mécanisme de transfert: Ce type de mécanisme représente le protocole employé pour transférer les tâches entre les nœuds [4]. Ce transfert peut être effectué en deux phases: placement initial ou processus de migration.

1.3.2.2. Mécanisme de mesure de la charge: Dans tous les type d'équilibrage de charge, une des difficultés majeures est celle qui consister à évaluer la mesure de la charge d'un site. Dans la plupart des travaux existants dans ce domaine, la longueur de la file d'attente celui qui détermine la charge d'un site. Cependant, certains auteurs préconisent comme indicateur de charge, la combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire.

1.3.2.3. Le mécanisme de communication de la charge: Le rôle de ce mécanisme est de déterminer la méthode par laquelle l'information de charge est communiquée entre la ressource et les agents responsables de la prise de décision. Cependant, ce mécanisme possède des problèmes de sécurité, de performance et du coût impliqué par la collecte et la distribution de l'information de la charge. Parmi les solutions proposées de ces problèmes on site, L'interrogation, la diffusion et la multicast [4]:

Le schéma suivant illustre les éléments essentiels des systèmes d'équilibrage de charge

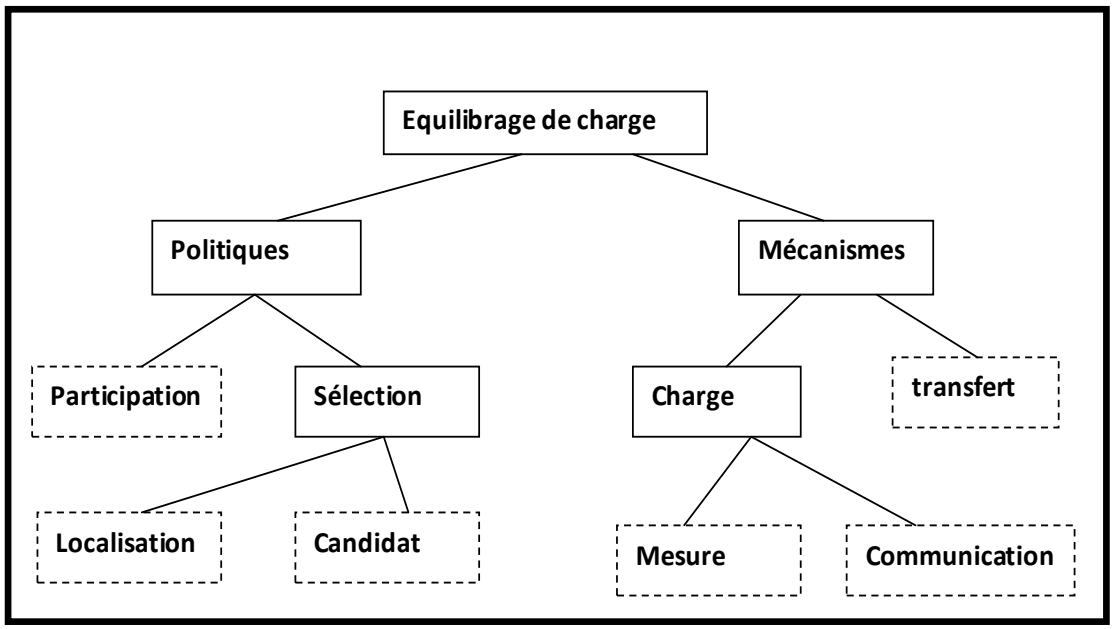

Figure 1.2: Eléments d'un système d'équilibrage de charge [4]

1.4. Catégories des approches d'équilibrage de charge

Beaucoup de catégories d'approches ont été proposées par les chercheures dans le domaine d'équilibrage de charge, nous citons dans cette partie que les plus utilisées à savoir:

1.4.1. Approche statique Vs Approche dynamique:

Dans l'approche d'équilibrage de charge statique, les tâches sont assignées aux machines avant l'exécution de l'application qui les-contient. Ainsi, les informations sur le temps d'exécution des tâches et celles des caractéristiques dynamiques des machines sont a priori supposées connues [3].

Dans l'approche d'équilibrage de charge dynamique, l'assignation des tâches aux machines se décide pendant la phase d'exécution, en fonction des informations collectées sur l'état de charge du système. Cette stratégie permet d'améliorer les performances d'exécution des tâches mais engendre une complexité dans la mise en œuvre de cette stratégie, particulièrement dans la définition de l'état de charge du système, qui doit se faire de manière continue [3].

1.4.2. Approche centralisée Vs Approche distribuée:

Dans l'approche centralisée, la collecte et la gestion de l'information de charge sont effectuées par une seule ressource désignée comme coordinateur. Il reçoit les informations courantes de charge de tous les autres sites qu'il assemble pour obtenir l'état de charge global du système ainsi que l'assemble dans un vecteur de charge. L'avantage de cette approche est qu'elle réduit les surcoûts du système grâce à la centralisation de l'information de charge.

Dans le cas de l'approche distribuée, chaque site du système est responsable de collecter les informations de charge sur les autres sites et de les rassembler afin d'obtenir l'état global du système. Les décisions de placement de tâches sont prises localement, étant donné que tous les sites ont la même perception de la charge globale du système [5].

1.4.3. Approche source-initiative Vs. Receveur-initiative:

L'approche source-initiative est appliquée lorsqu'un site, appelé source, détecte qu'il a une surcharge de travail et qu'il cherche à transférer le surplus vers un site faiblement chargé.

L'approche receveur initiative s'applique lorsqu'un site faiblement chargé, appelé receveur, demande à recevoir tout ou partie du surplus des sites surchargés [5].

1.5. Propriétés et performance d'un d'équilibrage de charge

Pour rendre meilleur la manière qu'un équilibrage de charge améliore l'exécution des tâches, ou autrement pour garantir un bon équilibrage de la charge dans un système, des algorithmes d'équilibrage de charge doivent remplir quelques propriétés et performance, on donne dans cette partie que les trois principales propriétés qui permettent de distinguer la qualité de différents équilibrages de charge à savoir [2]:

1.5.1. Efficacité:

La qualité principale d'un système d'équilibrage de charge doit être de pouvoir exploiter la puissance de la ressource (machine) à son maximum en minimisant son temps oisif. En général, Il est possible de vérifier l'efficacité d'un équilibrage de charge en observant directement la proportion de temps passé oisif pour chaque ressource.

1.5.2. Equité:

Cette propriété consiste à attribuer, à un moment donné pour des tâches ayant les mêmes priorités, le même temps d'exécution. Sur une ressource donnée, elle est assurée par l'ordonnanceur qui se charge d'assurer l'équité entre les tâches. Cependant, il n'existe pas de moyens pour ajuster le temps octroyé à chaque tâche en fonction de ce qui se passe sur les autres ressources. A cet effet, le système d'équilibrage de charge assure l'équité inter-ressources.

1.5.3. Localité:

Les taches qui ont une affinité particulière pour une ou plusieurs ressources doivent y être affectées en priorité. Cette propriété implique, également, une réduction du nombre de migration, notamment, pour une tâche donnée, un ratio de temps d'exécution supérieur sur certaines ressources que sur d'autres.

1.6. Stratégie d'équilibrage de charge

La structure arborescente du modèle proposé par cette approche, permet de développer une stratégie hiérarchique à trois niveaux d'équilibrage: Intra-Grappe, Extra-Grappe et Inter-Grappes, ces niveaux nous permette de distribuer les requêtes vers l'un des serveurs de la grappe [10]: d'abord, nous allons donner une définition de grappe de serveurs

✓ **Qu'est-ce que la grappe de serveurs?**

On parle de grappe de serveurs ou de ferme de calcul (cluster en anglais) pour désigner des techniques consistant à regrouper plusieurs ordinateurs indépendants appelés Nœuds, afin de permettre une gestion globale et de dépasser les limitations d'un ordinateur pour [12]:

- Augmenter la disponibilité.
- Faciliter la montée en charge.
- Permettre une répartition de la charge.
- Faciliter la gestion des ressources (processeur, mémoire vive, disques dur, bande passante réseau).

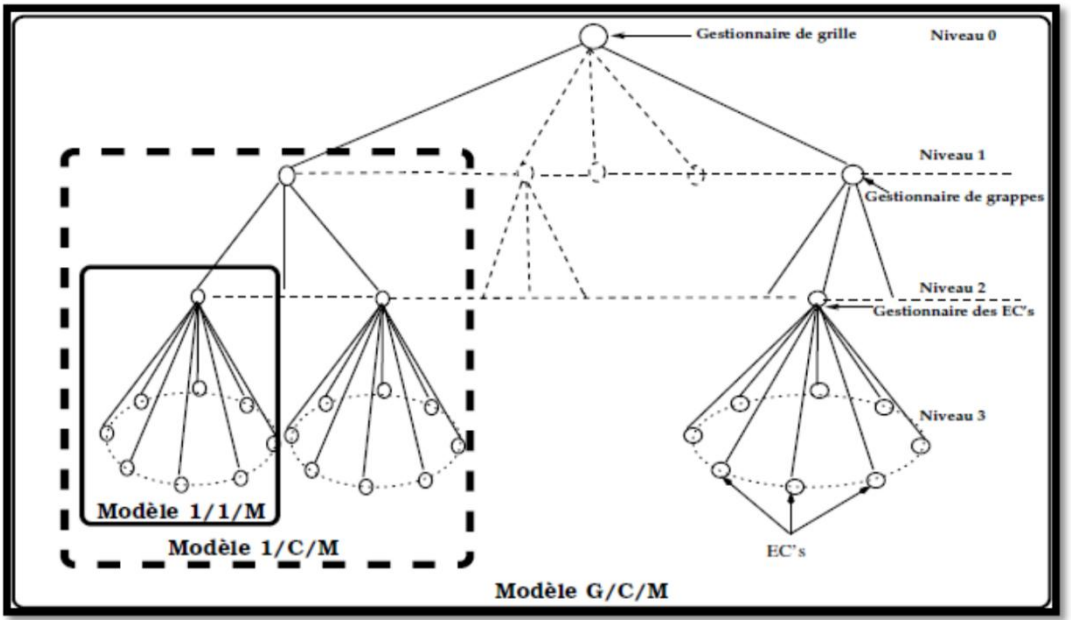

Figure 1.3: Equilibrage de charge basé sur le grappe[6]

1.6.1. Equilibrage Intra-Grappe: (en analogie avec Intranet)

La plus simple des grappes est l'intra-grappe, composée d'un ensemble relativement limitée de ressources et de services et appartenant à une organisation unique.

Dans le premier niveau, chaque gestionnaire d'éléments de calcul décide de déclencher une opération d'équilibrage en fonction de la charge courante de la grappe qu'il gère. Cette charge est estimée à partir des différentes informations de charge envoyées périodiquement par l'élément de calcul qui composent la grappe. Le gestionnaire tente, en priorité, d'équilibrer la charge de la grappe localement en la répartissant entre l'élément de calcul qui lui appartient. Cette approche de localité a pour objectif de réduire les coûts de communication, en évitant les communications extra/inter-Grappes [5].

1.6.2. Equilibrage Extra-grappe:(en analogie avec Extranet)

Une extra-grappe étend le modèle en regroupant plusieurs intra-grappes. Dans ce deuxième niveau, l'équilibrage se fait à l'échelle des extra-grappes. Il intervient dans le cas où certains gestionnaires des EC's n'ont pas réussi à équilibrer localement leurs charges. Il y aura ainsi transfert de tâches entre grappes surchargées et grappes sous-chargées de la même extra-grappe. Dans un souci de réduire au maximum les coûts de communication, les grappes réceptrices seront sélectionnées en fonction des débits des réseaux [5].

1.6.3. Equilibrage Inter-grappes:

Dans ce troisième niveau, l'équilibrage de charge n'est déclenché que si un ou plusieurs gestionnaires de grappes n'arrivent pas équilibrer leurs charges localement entre les grappes qu'ils gèrent. Dans ce cas extrême, il sera alors nécessaire au gestionnaire de grille de transférer un certain nombre de tâches à partir d'extra-grappes surchargées vers d'autres qui sont sous-chargée [5].

1.7. Les Algorithmes d'équilibrage de charge

Dans ce qui suit, nous présentons les principaux algorithmes de l'équilibrage de charge à savoir: Algorithmes d'équilibrage Intra-grappe, Extra-grappe et Inter-grappes et Algorithmes d'équilibrage Aléatoire / Round-Robin:

1.7.1. Algorithmes d'équilibrage Intra-grappe, Extra-grappe et Inter-grappes:

1.7.1.1. Algorithme d'équilibrage Intra-grappe:

Cet algorithme constitue le noyau de notre stratégie. L'approche de localité adoptée fait que c'est le niveau d'équilibrage qui sera le plus fréquemment sollicité. Il est déclenché lorsqu'un gestionnaire d'EC's constate qu'il y a déséquilibre entre ses EC's. Pour faire ce constat, le gestionnaire reçoit, de manière périodique, les informations de charge à partir de chaque élément de calcul. Sur la base de ces informations et du seuil d'équilibrage estimé, il analyse de manière régulière la charge du groupe. En fonction du résultat de cette analyse, soit il décide de déclencher un équilibrage local en cas de déséquilibre, soit il décide d'informer son gestionnaire du niveau supérieur sur sa charge actuelle [1].

1.7.1.2. Algorithme d'équilibrage Extra-grappe:

Cet algorithme, qui utilise une approche source-initiative, est exécuté uniquement lorsque certains gestionnaires d'EC's n'ont pas réussi à équilibrer localement leur charge pour cause de saturation ou d'offre insuffisante. Dans ce cas, le gestionnaire de grappes tente

d'équilibrer la charge globale de l'extra-grappe à travers les éléments qu'il gère. Contrairement à l'algorithme intra-grappe, l'équilibrage extra-grappe devra tenir compte des coûts de communication. Durant le transfert de tâches, la grappe réceptrice celle qui nécessite le plus petit coût de transfert est choisit. Ainsi, le critère de sélection sera pondéré par le coût de communication (coût de transfert de tâches) pour assurer qu'une tâche ne peut être transférée que lorsque son temps de réponse estimé dans la grappe réceptrice, auquel le temps de transfert est rajouté à partir de la grappe source, est meilleur que son temps de réponse dans la grappe source [1].

1.7.1.3. Algorithme d'équilibrage Inter-grappes:

Ce troisième niveau d'algorithme procède à un équilibrage global à travers toutes les extra-grappes de la grille. Il est exécuté dans le cas extrême où la majorité des gestionnaires de grappes n'arrivent pas à équilibrer localement leur surplus de charge. Le recours à ce type d'équilibrage ne sera effectif que si les coûts de communication, associés aux différentes tâches à transférer, seront réellement intéressants. En d'autres termes, il serait plus judicieux de laisser un ensemble de tâches attendent la libération de ressources que de procéder à leur transfert vers d'autres sites sans un gain de temps substantiel [1].

1.7.2. Algorithme d'équilibrage Aléatoire / Round-Robin:

Dans cette partie nous allons présenter les algorithmes d'équilibrage de type aléatoire ainsi que les algorithmes d'équilibrage de type Round-Robin

1.7.2.1. Algorithme d'équilibrage Aléatoire:

Dans la répartition aléatoire, chaque requête entrante est envoyée à un serveur sélectionné aléatoirement. Cet algorithme est très rapide et il assure un bon équilibrage de charge sur une période de temps donnée. Par contre, si le nombre de requêtes est faible, la probabilité d'un déséquilibre de charge est forte. La figure suivante représente la distribution aléatoire des requêtes sur les serveurs Web [13].

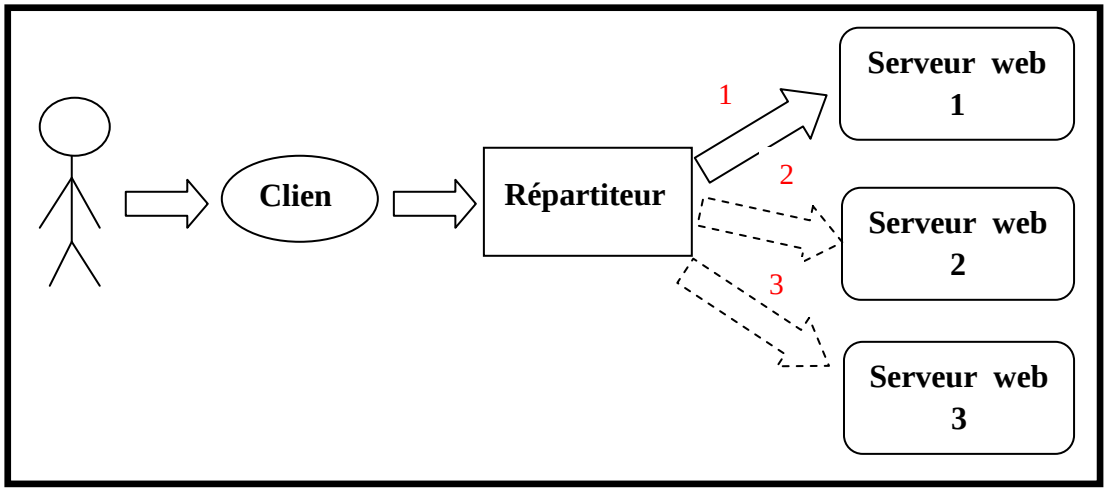

Figure 1.4: Distribution aléatoire des requêtes sur les serveurs Web

1.7.2.2. Algorithme d'équilibrage Round-Robin:

Ce type d'algorithme transfère chaque requête entrante au prochain serveur de la liste. Tous les serveurs sont traités de manière égalitaire sans prendre en compte leur nombre de connexions entrantes ou leur temps de réponse Contrairement au DNS Round-robin, la granularité est plus fin puisqu'elle se situe au niveau de la connexion réseau plutôt qu'au niveau de l'hôte. Cela évite les inconvénients liés au cache du DNS qui peut conduire à un déséquilibre de charge. Par contre, le mécanisme du Round-Robin peut engendrer un déséquilibre de charge si le temps des requêtes ou la capacité des serveurs sont différents. Ceci est dû au fait qu'ils ne prennent pas en compte la charge de travail des serveurs [13].

1.7.2.3:Algorithme d'équilibrage Round-robin à poids: L'algorithme de round-robin à poids est mis en place pour mieux gérer les serveurs ayant des capacités de traitement différents. Chaque serveur se voit assigner un poids qui indique sa capacité de traitement. L'ordonnancement par le répartiteur tiendra alors compte de ce poids pour transférer les requêtes aux serveurs.

Dépondant, la répartition par round-robin à poids est plus efficace que par round-robin simple lorsque la capacité des serveurs est différente. Cependant, cela peut conduire à un déséquilibre si la charge des requêtes varie beaucoup. Il est possible que la majorité des requêtes lourdes soit toujours dirigée vers le même serveur.

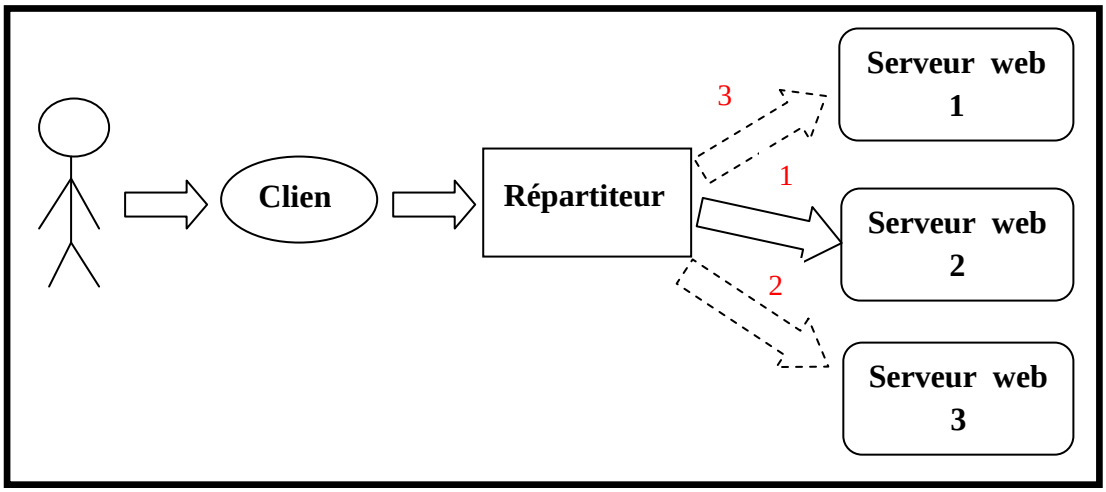

Figure 1.5: Distribution round-robin des requêtes sur les serveurs Web

1.8. Conclusion

Dans ce chapitre, nous avons présenté succinctement les notions de base concernant l'équilibrage de charge dans la base de données. Nous avons présenté au premier temps la classification des systèmes d'équilibrage de charge, puis nous en bref, avons présenté ces principales catégories. En fin, nous avons présenté les principaux algorithmes de l'équilibrage de charge. Dans le chapitre suivant, nous allons étudier les systèmes de réplication de données.

Chapitre 2

La réplication dans les bases de données

2.1. Introduction

Dans le deuxième chapitre, nous allons étudier le système de réplication de données. Grâce à laquelle nous essayons d'identifier ce mécanisme et à travers laquelle nous essayons de répondre à un certain nombre d'obstacles et de problèmes qui se produisent lors de la connexion de nombreux clients. Pour accéder à un site particulier soit pour lire ou écrire des données qui se chevauchent et peuvent s'effondrer le site à cause de cela le site perd beaucoup de données importantes et le client perd le contact avec ce site.

Lors de l'utilisation d'une application de base de données, les problèmes majeurs rencontrés sont: la disponibilité des données, le temps de réponse et la résistance de l'application à d'éventuel panne. Et l'une des solutions à ses problématiques est la réplication de données.

2.2. Problème de performances

Le problème de perte de données dans les bases de données conduit à un problème de performance à cause du manque d'équilibrage entre les serveurs de base de données.

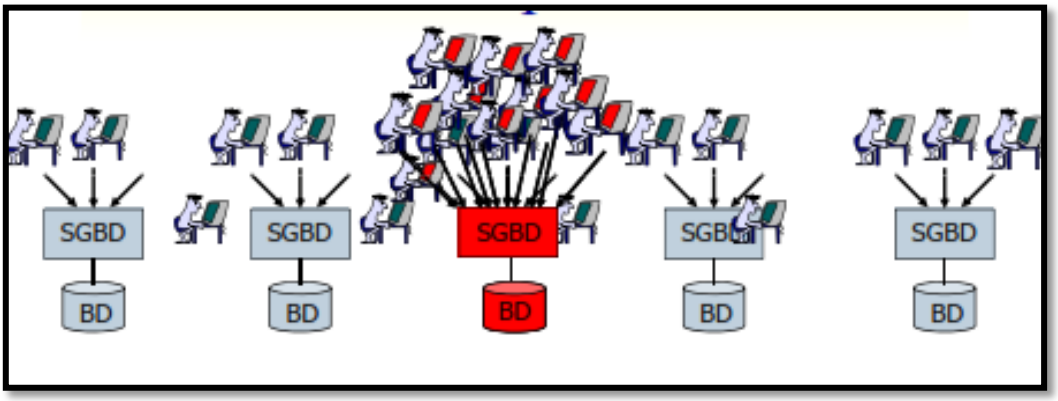

Figure 2.1: Problème de performances

2.3. Définition de réplication

La réplication est le processus qui permet de copier et de distribuer les données à partir d'une base de données à d'autres, puis synchroniser ces données afin de maintenir la cohérence [17]. Elles peuvent être des bases de données locales ou distantes et connectées à internet ou à d'autres modes de communication.

Remarque: Il existe une différence entre répliquer et copier dans ce sens que répliquer occasionne la cohérence des données et des répliques. De ce fait la réplication est plus vaste qu'une simple copie [14].

2.4. Principe de la réplication

Le principe de répétition, ce qui implique au moins deux systèmes de gestion de base de données (SGDB) est très simple et se déroule en trois étapes [7]:

- ✓ Base de données maître reçoit des instructions de mise à jour (INSERT, UPDATE ou DELETE).
- ✓ Les modifications faites sur les données sont détectées et stockées (dans une table, un fichier, une queue) en vue de leur propagation.
- ✓ Le processus de réplication publie les modifications à l'esclave, bien sûr, il peut y avoir plus qu'un esclave.

Il est tout à fait possible de faire la réplication dans les deux directions (maître vers esclave et vice versa). Parler dans ce cas une réplication bidirectionnelle ou symétrique. Si la réplication est unidirectionnelle (uniquement à partir de la principale forme de fin) et se réfère à la réplication de lecture seule ou non-identique. Elle peut être reproduit plus synchrone ou asynchrone. Dans le premier cas impliqué résoudre les conflits potentiels entre les deux sites avant de poster des transactions, dans le second cas, l'adoption de la résolution dans des transactions distinctes. Par conséquent, il est possible d'avoir quatre modèles de réplication [8]:

- ✓ Réplication asymétrique (maître/esclave) avec propagation asynchrone.
- ✓ Réplication asymétrique (maître/esclave) avec propagation synchrone.
- ✓ Réplication symétrique ou Peer-to-Peer (update everywhere) avec propagation asynchrone.
- ✓ Réplication symétrique avec propagation synchrone.

2.5. Classification des solutions de réplication

2.5.1. La mise à jour synchrone:

Aussi appelée « Réplication en temps réel ». La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant la validation (commit) de la requête sur le serveur où la requête est exécutée. Ce type de réplication assure un haut degré d'intégrité des données mais requiert une disponibilité permanente des serveurs et de la bande passante. Ce type de réplication, fortement dépendant des pannes des systèmes, nécessite de gérer des transactions multi sites coûteuses en ressources [9].

2.5.2. La réplication synchrone asymétrique:

Elle utilise un site primaire qui propage les mises à jour en temps réel vers un ou plusieurs sites secondaires. La table répliquée est immédiatement mise à jour pour chaque modification par utilisation de trigger sur la table maître [7].

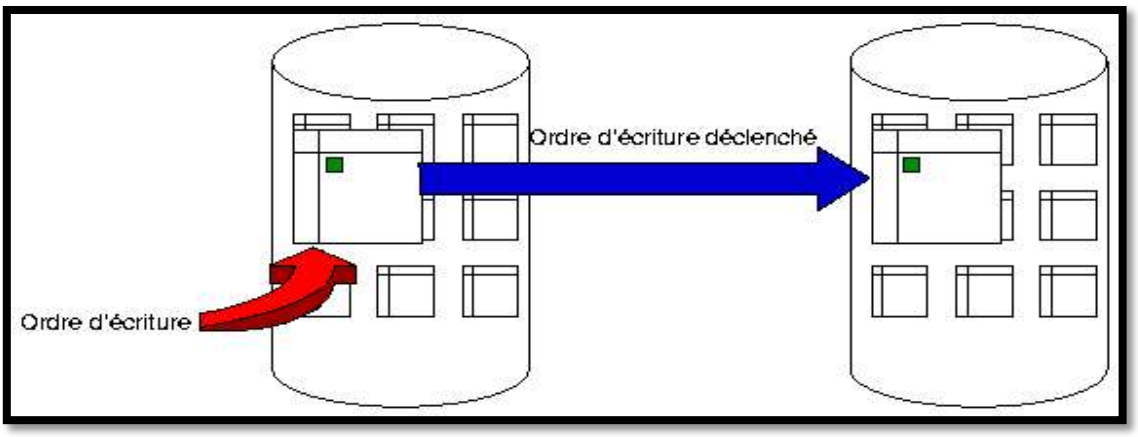

Figure 2.2: Réplication synchrone asymétrique

2.5.2.1. La réplication synchrone symétrique:

Lors de la réplication synchrone symétrique, il n'y a pas de table maître. L'utilisation de trigger sur chaque table doit différencier une mise à jour client à répercuter d'une mise à jour par réplication. Cette technique nécessite l'utilisation de jeton [7].

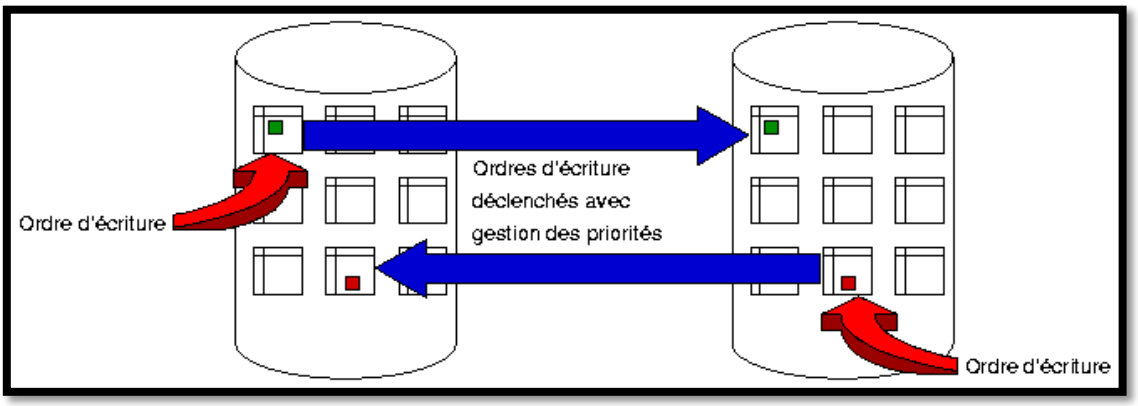

Figure 2.3: Réplication synchrone symétrique

2.5.2.2. La mise à jour asynchrone:

La réplication asynchrone stocke les opérations intervenues sur une base de données dans une queue locale pour les propager plus tard à l'aide d'un processus de synchronisation. Ce type de réplication est plus flexible que la réplication synchrone. Il permet en effet de définir les intervalles de synchronisation, ce qui permet à un site de fonctionner même si un site de réplication n'est pas accessible. Si le site distant est victime d'une panne, l'absence de synchronisation n'empêche pas la consistance de la base maître [15].

2.5.3. La réplication asynchrone asymétrique

Elle propage les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple [7].

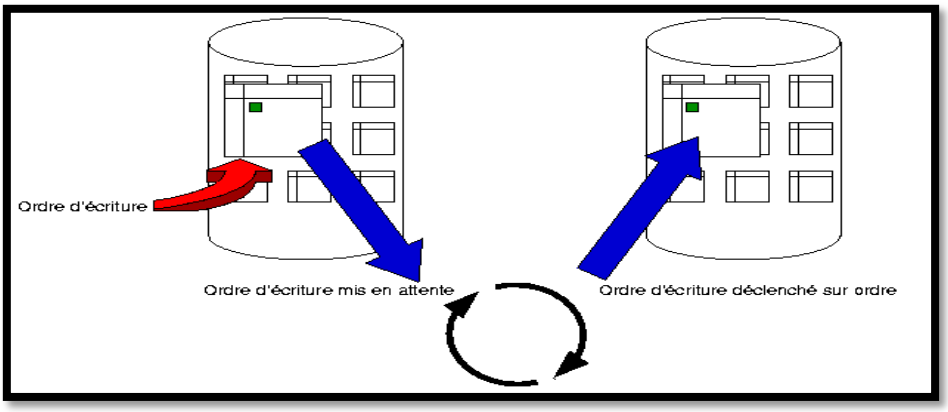

Figure 2.4: Réplication asynchrone asymétrique

2.5.4. La réplication asynchrone symétrique

Dans ce cas, la mise à jour des tables répliquées est différée. Cette technique risque de provoquer des incohérences de données [7].

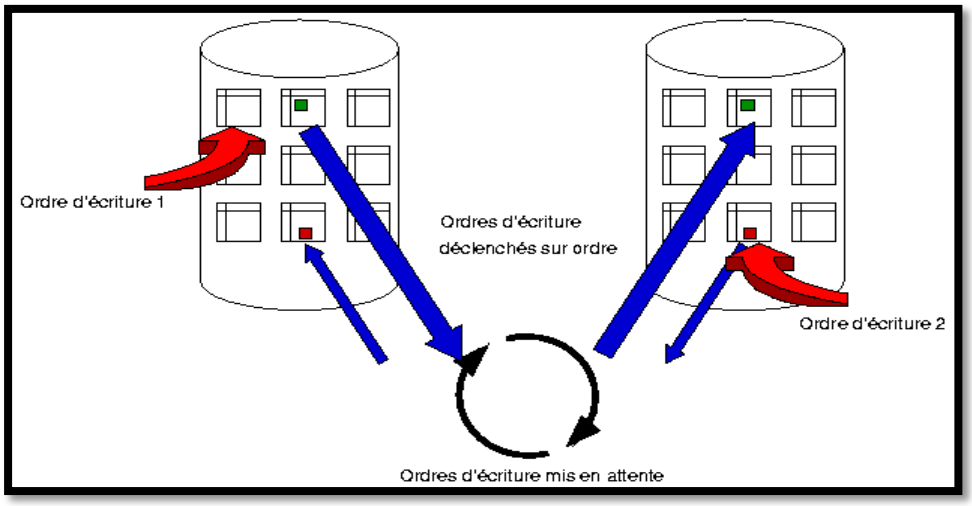

Figure 2.5: Réplication asynchrone symétrique

2.6. Les avantages et les inconvénients de la réplication de base de données

2.6.1. Les avantages:

✓ faciliter l'accès aux données en augmentant la disponibilité Soit [10]:

- parce que les données sont copiées sur différents sites permettant de répartir les requêtes.
- parce qu'un site peut prendre la relève lorsque le serveur principal s'écroule.

✓ Performances amélioré.

- ✓ Probabilité de panne plus faible → P(panne de N serveurs) = P (panne d'un serveur)^N [9].
- ✓ Accès à plusieurs copies dans un environnement Accès à plusieurs copies dans un environnement réparti [9]:
  - Parallélisme
  - Equilibrage de charge
  - Meilleur temps de réponse
  - Réduit les transferts de données (meilleure localité)

2.6.2. Les inconvénients:

- ✓ Gestion des mises à jour
  - Propager les mises à jours. à toutes les répliques concernées
  - Surcoût : échange de messages inter-sites
  - Cohérence mutuelle des répliques
  - Cohérence forte ou à terme
  - Synchroniser → messages supplémentaires

2.7.Tolérance aux fautes

2.7.1. Réplication symétrique (active):

La réplication active où chaque copie est répliquée et exécutée simultanément sur n machines distinctes. Les répliques doivent synchroniser leurs exécutions à chaque fois qu'un message est reçu ou envoyé afin d'assurer que toute réplique réalisant un même calcul reçoive les messages provenant des autres processus dans le même ordre. Plusieurs modes de défaillance d'une réplique existent, allant de la défaillance par arrêt (défaillance la plus simple) à la défaillance byzantine (défaillance la plus complexe). Dans le cas de défaillance par arrêt, une réplique défaillante ne produit plus aucun résultat.

Dans le cas de défaillances byzantines, une réplique défaillante continue de produire les résultats mais ceux-ci sont erronés. La tolérance aux fautes est assurée par masquage d'erreur. La défaillance d'une copie est masquée par le comportement des copies non défaillantes. Comme chaque copie joue un rôle identique. La défaillance de l'une d'entre elle ne perturbe pas le service fourni par le composant [16].

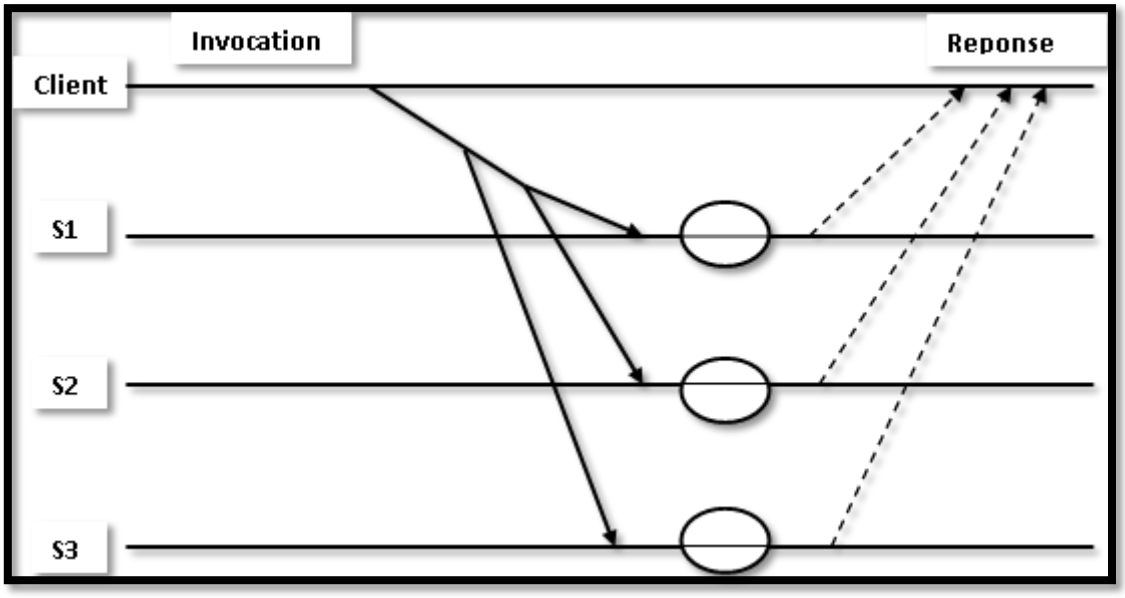

Figure 2.6: Principe de la réplication active

2.7.2. Réplication symétrique (passive):

La réplication passive où un composant logiciel est répliqué en n exemplaires, mais une seule des n répliqués effectue le calcul. Les n-1 autres répliques sont passives et ne prennent la relève que si la réplique active est défaillante. Pour que cette stratégie fonctionne, il est nécessaire que la relique active transmette aux répliques passives, à intervalles réguliers son état d'exécution. Cet état d'exécution composé d'une pile, de données et de registres est stocké par chacune des répliques passives et constitue un point de reprise. Si la réplique active est défaillante, une des répliques passive est activée et reprend l'exécution du calcul à partir du dernier point de reprise enregistré. On dit que le processus effectue un retour arrière. Recréer un état d'exécution simule une remontée dans le temps en ramenant le calcul dans l'état qu'il occupait avant la manifestation de la défaillance. La tolérance aux fautes est réalisée par détection et compensation de l'erreur. La défaillance d'une copie secondaire ne nécessite aucun traitement particulier. Par contre, la défaillance de la copie S1 implique que les copies secondaires désignent l'une d'entre elle (par exemple S2) comme la nouvelle copie primaire. Les cas suivants peuvent se présenter [16].

✓ La défaillance de la copie primaire avant l'envoie du message "update" a pour conséquence que le client n'obtienne aucune réponse à la requête. Tout le traitement de la requête effectué par la copie primaire S1 est perdue. Le client doit alors réémettre sa requête en l'adressant à la nouvelle copie primaire S2. Cette dernière doit être en mesure de construire la réponse que le client attend.

✓ Deux situations peuvent se présenter selon que la défaillance de S1 ait été détectée pendant l'envoie du message "update" aux copies secondaires ou avant l'envoi de la réponse au client. Ceci est le cas le plus difficile à appréhender [16]:

1. L'atomicité doit être garantie le message " update" doit être reçu par tous ou par aucune des copies secondaires.

2. Le client doit réémettre sa requête au nouveau primaire.

- La défaillance du primaire S1 a lieu après avoir envoyé la réponse. Dans ce cas un nouveau primaire doit être désigné.

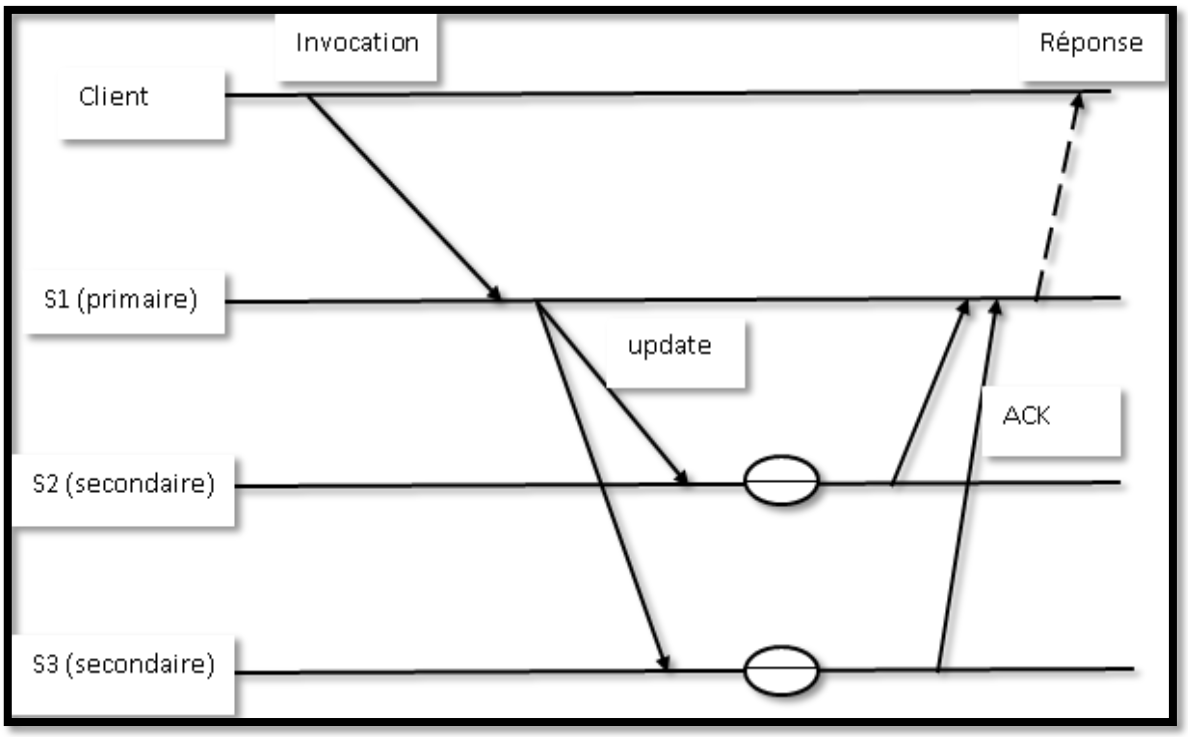

Figure 2.7: Principe de la réplication passive

2.8. Conclusion

Dans ce chapitre, nous avons rappelé de manière succincte les notions de base sur la réplication dans les bases de données, en tant que des moyens efficaces et fiables, et ceci sous la base d'une compréhension du fonctionnement du système de réplication de données, afin de fournir des solutions au problème de données et rendre disponible la mêmes données sur plusieurs serveurs. Dans le chapitre suivant nous allons proposer une solution pour l'équilibrage de charge entre les serveurs de réplication.

Chapitre 3

Proposition d'une approche pour l'équilibrage de charge

3.1. Introduction

Nous nous intéressons dans ce chapitre, à étudier le problème d'équilibre de charge pour les requêtes de lecture dans un système de réplication de données, nous présentons d'abord l'architecture itératif générale de l'équilibrage de charge pour les requêtes de lecture dans un système de réplication de données, dont nous mettons les différents éléments constrictifs de ce système, ensuite, nous expliquons le principe de fonctionnement de cette architecture en détaille, finalement, nous donnons l'algorithme utilisé dans cette application ainsi que les avantages de cet algorithme en ce qui concerne l'équilibrer la charge.

3.2. Architecture global

L'architecture globale du procédé de l'équilibrage de charge pour les requêtes de lecture dans un système de réplication est illustrée par la figure 3.1, elle présente, les différents états et transitions d'une requête de lecture lors du déclenchement du processus d'équilibrage de charge.

En générale, les équilibreurs de charge peuvent être ajoutés à un environnement de serveur pour améliorer les performances et la fiabilité de la distribution de la charge entre plusieurs serveurs. Si par exemple, l'un des serveurs a une charge plus grande par rapport à un autre serveur, alors l'équilibreur de charge affecte les nouvelles requêtes aux serveurs les moins chargés de façons à garantir un équilibrage entre les serveurs.

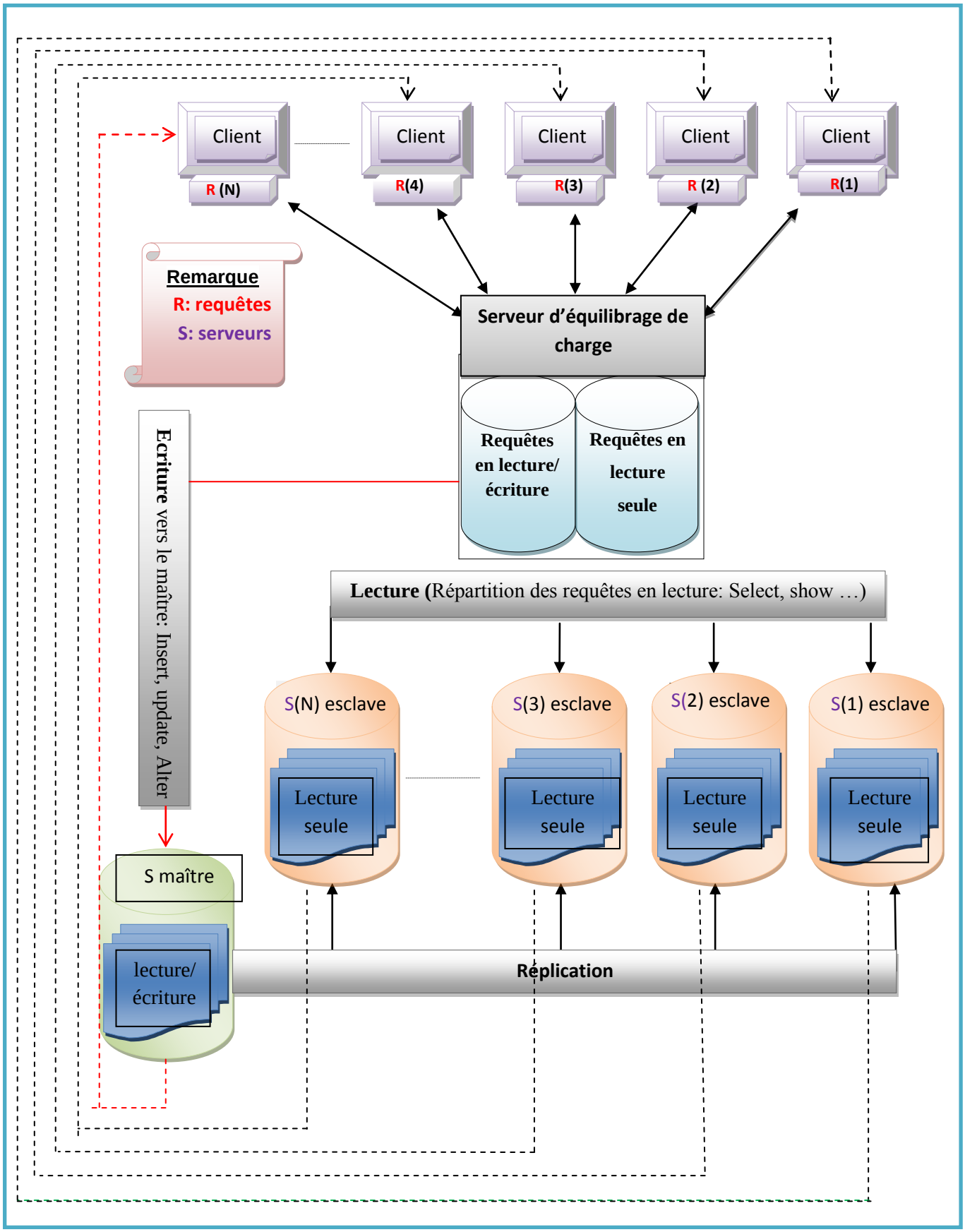

Figure 3.1: Architecture global d'équilibrage de charge pour les requêtes de lecture dans le système de réplication de données.

En effet, dans l'architecture générale présentée dans la figure 3.1, le processus d'équilibrage de charge passe par les étapes suivantes:

- Chaque client doit envoyer une requête vers le server d'équilibrage de charge. Il est important de noter qu'il existe deux types de requêtes: requête de lecture seule (Select, show…), et l'autre pour l'écriture (Insert, update, Alter…), on s'intéresse dans ce travail uniquement, aux requêtes de lecture.
- Le serveur d'équilibrage de charge reçoit les requêtes (lecture ou écriture), puis il les envoie vers l'un des serveurs de système de réplication, de façon à maintenir l'équilibrage entre les serveurs.
- lorsque serveur de lecture reçoit la requête, il l'exécute et renvoie la réponse au client à travers une liaison par un adresse IP et un port.

Nous pouvons également diviser la forme précédente en trois sections comme suit:

- **Section de clients:** Cette section est constituée d'un ensemble de clients indépendants les uns des autres, et qui sont situés dans des sites différents, dont chacun de ses clients désir avoir une réponse spécifique en fonction de la demande de chaque client pour la lecture, et cela en envoyant une demande à l'une des bases de données affectée pour cette utilisation.
- **Section d'équilibrage de charge:** Elle contient un serveur d'équilibrage qui implémente une application spécifique pour répondre aux demandes des clients d'une part et équilibrer la charge par la coordination avec le système de réplication d'autre part, cette équilibrage est faite par des algorithmes dont son objectif est de surveiller le processus de chargement de tous les serveurs de bases de données avec les demandes de chaque client.
- **Section de réplication:** Est un ensemble de serveurs de bases de données qui possèdent les mêmes données, ils sont configurés pour la lecture uniquement au niveau de chacun des serveurs de base de données.

3.3. Architecture Interne détaillée

Pour mieux comprendre plus profondément le principe de fonctionnement de l'architecture générale du processus d'équilibrage de charge, nous expliquerons les opérations qui se déroulent au niveau d'équilibreur de charge, ceci en raison de son importance, ce qui est de relier les clients avec le système de réplication, puis faire l'équilibrage entre eux selon la demande de client et la charge de chaque serveur. La figure ci-dessous représente l'architecture détaillée.

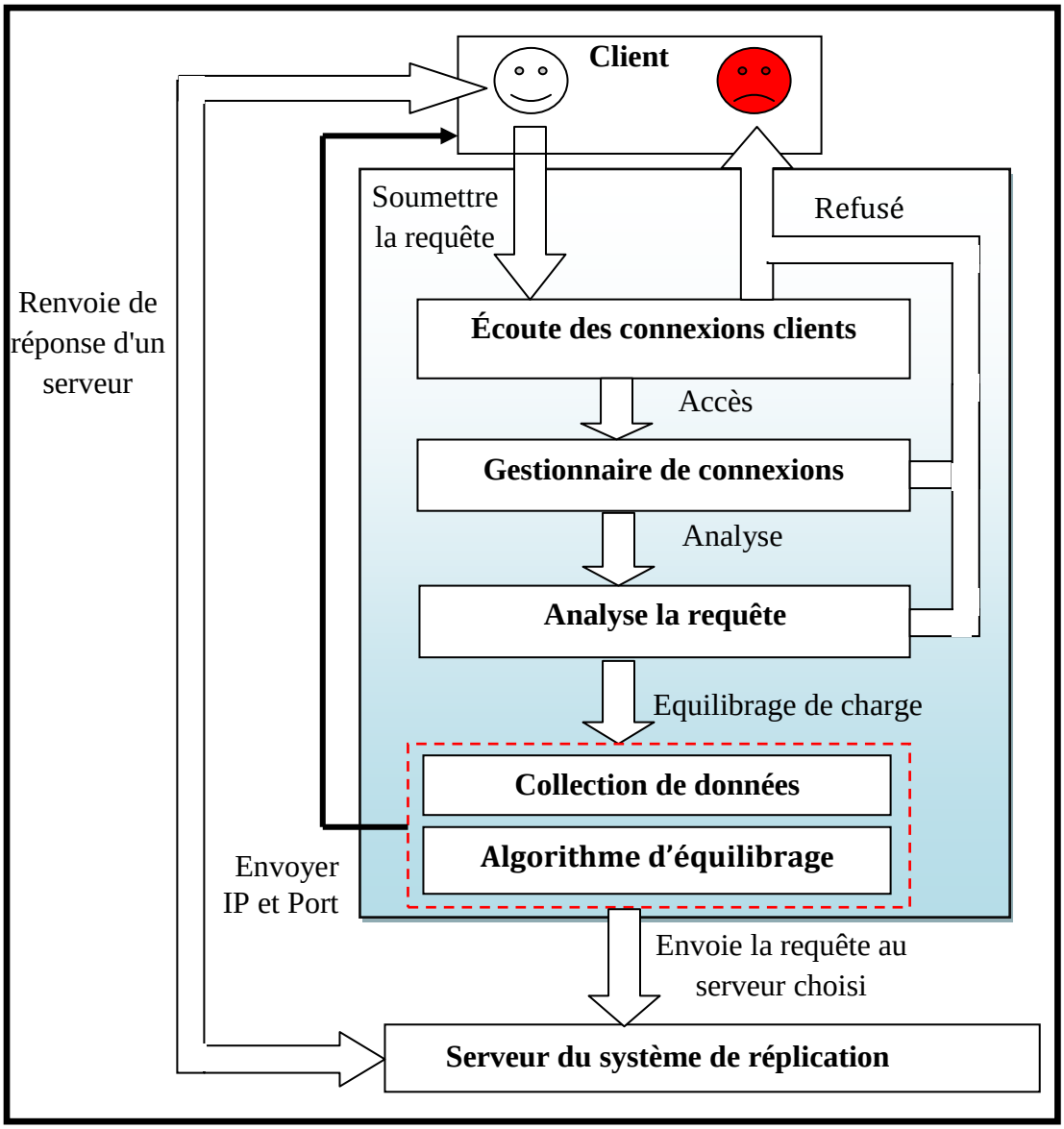

Figure 3.2: Structure détaillée du fonctionnement de serveur d'équilibrage de charge.

Dans la section suivante nous allons expliquer les opérations effectuées par le serveur d'équilibrage de charge.

3.3.1. Écoute des connexions de clients:

Afin d'effectuer cette opération, nous faisons un port de communication logique ouverte entre les clients et le serveur d'équilibrage de charge afin d'échanger les données entre eux, ainsi, pour permettre aux clients de se connecter à l'équilibreur lors de l'envoi des requêtes, ce port de communication devient alors le seul moyen d'accès au système pour les clients, en conséquence, le serveur reçoit toute requête des clients à travers ce port. La figure suivante montre ce concept.

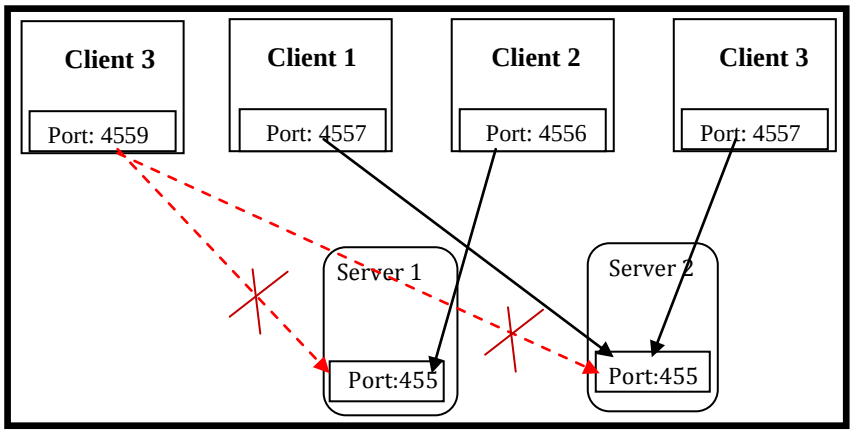

Figure 3.3: Exemple serveur Écoute la connexion client

3.3.2. Gestionnaire de connexions:

Dès que les clients se connectent au serveur, ce dernier décide d'accepter ou refuser le client, ainsi, il fait plusieurs processus afin de gérer les contacts avec les clients, d'où, il surveille une application particulière de chacun de ces clients quand il est connecté, tels que la détermination du nombre de clients connectés, l'interdiction la connexion d'un client particulier, ou le refus de la communication d'un tel client au plus d'une application.

La figure suivante montre cette gestion de serveur:

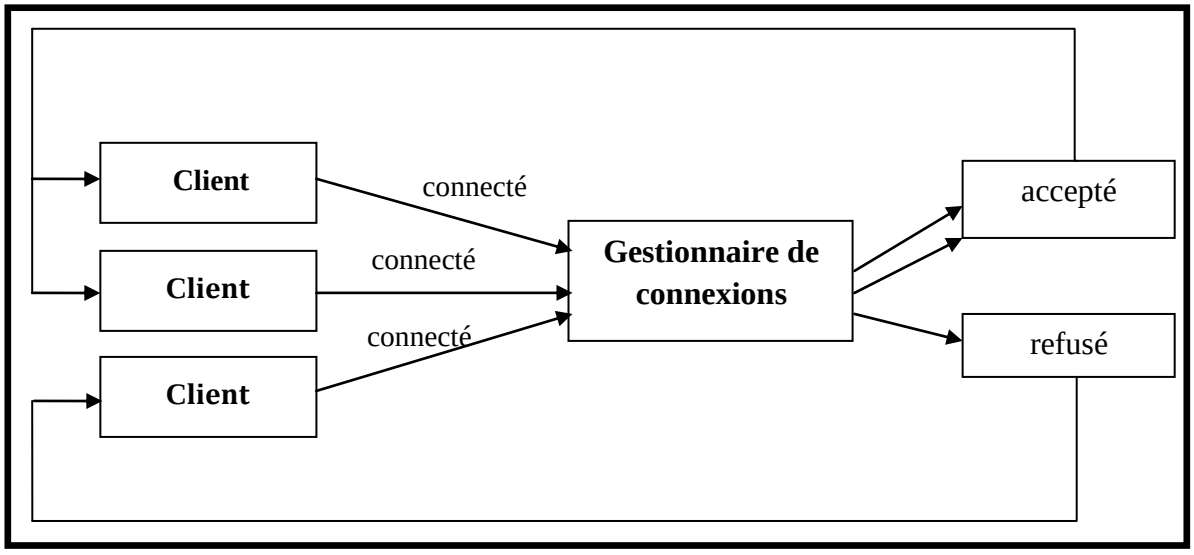

Figure 3.4: Gestionnaire de connexions

3.3.3. Analyse de requête:

Ce processus reçoit la requête envoyée par le client, puis il afin de décider si cette requête est exécutable ou non. Ensuit, il prend la première partie de la requête et la compare avec les requêtes de lecture existantes afin de pouvoir décider si cette requête de lecture ou non.

Dans le cas de lecture il affecte la requête selon la charge de chaque serveur esclave, sinon il envoie la requête au serveur master.

La figure suivante montre clairement cette opération:

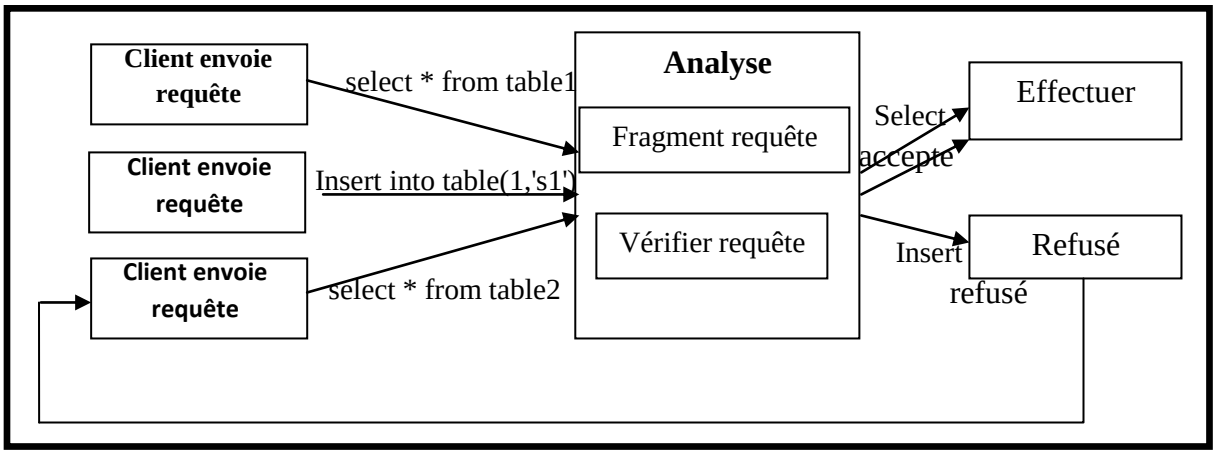

Figure 3.5: Analyse de la requête

Il est important de noter que le processus d'analyse est basé sur deux points essentiels, la partition et la vérification des requêtes.

- La partition des requêtes: dans cette étape, on effectue la division de la requête puis, on prend le premier élément pour la vérification.
- La vérification de la requête : dans cette étape, on doit vérifier l'existence de cette requête à l'intérieur des éléments de lecture.

3.3.4. Equilibrage de charge:

Dans cette étape, deux parties doivent être réalisées afin de d'équilibrer la charge, à savoir:

- ✓ **Collection de données:**

Ce sont les données sur les serveurs distribués, ces serveurs sont de lecture seule et contiennent la même base de données. Tous les serveurs sont identifiés et définis pour exécuter les requêtes selon la demande de client.

- ✓ **Algorithme d'équilibrage:**

Cet algorithme est le processus principal du serveur, dont il collecte toutes les informations sur le clients et les serveurs existants dans le système de réplication, après avoir réorganisé toutes ces informations collectées, il commence le processus d'équilibrer la charge selon une approche particulière et en fonction des clients connectés à chaque compte du serveur.

3.4. Les algorithmes proposés

Dans cette partie, nous allons donner les algorithmes utilisées dans notre application, à savoir:

3.4.1. Algorithme de collection d'informations sur les nœuds de travail:

La fonction principale de cet algorithme est d'apporter toutes les informations de chacun des serveurs du système de réplication de donnée, comme par exemple, le titre, le port, le type, etc... Elle est donnée comme suit:

```
Information _ NT()
début
        tab _ esclave  information noeuds de travail
        tab _ maître  information noeuds de travail
        /* NT nœuds de travail */
        /* C ensemble de nœuds */
        Pour (chaque NTi du cluster C)
            si (NTi  type esclave) alors
              ajouter NTi à tab _ esclave
               sinon
                 si (NTi  type maître) alors
                   ajouter NTi à tab _ maître
                 Finsi
            Finsi
        finpour
fin
```

3.4.2. Algorithme d'analyse de la requête:

L'objectif de cet algorithme est de déterminer le type de la requête, (soit une requête de lecture soit d'écriture), l'idée consiste à décomposer cette requête, puis la vérifié, cette algorithme est donné comme suit:

```
Analyse _ requête (R requête)
début
        Résulta est Résulta de la Analyse;
        S= {Select, show,…}
        /* R C'est l'avenir de la requête du client */
        /* S Est l'ensemble des mots pour la comparaison
        comme si de ce groupe la requête lui lire ou pas*/
        Pour (premier mot de la requête R)
            Si (mot dans S) alors
                    Résulta←Cette requête lecture
              sinon
                    Résulta←Cette requête ne pas  lecture
              Finsinon
            Finsi
        finpour
fin
```

3.4.3. Algorithme Equilibrage de charge:

Cet algorithme est consacré à l'équilibrage de la charge de requête envoyé par chacun des clients, et cela après avoir effectué les tâches suivantes:

- Sélectionner les informations des serveurs
- Vérifier la connexion de chaque client
- Vérifier la requête envoyée par le client

Cet algorithme est donné comme suit:

```
Equilibrage _ de _ charge (R requête)
début
        S←information _ NT()
        /* S C'est le résultat de la fonction information _ NT()*/
        RE ←Analyse _ requête(R)
       /* RE C'est le résultat de la fonction Analyse _ requête */
       A←Connexion _ client(C)
       /*A  C'est le résultat de la fonction Connexion _ client */
       Si (A=accepte et RE=lecture) alors
          Pour (chaque Si du Taille de S)
             si (Si La dernière opération sur lui)alors
               Si+1← Exécute une requête sur ce serveur
                     Envoyer le résultat au client;
              sinon
                   Ne répond ne pas;
              Finsinon
            Finsi
          Finpour
      Finsi
fin
```

3.4.4. Algorithme de connexion client:

Cet algorithme a pour objectif de vérifier la connectivité du client, pour pouvoir envoyer des requêtes et recevoir des réponses selon le serveur d'équilibrage de charge. On peut résumer cet algorithme comme suit:

```
Connexion _ client (C client)
début
        Réponse est Résultade la connexion des gestionnaires
        /* C Est le client en mode hors connexion*/
        Si (client C Rencontrer toutes les conditions) alors
                Réponse ← accepte
         sinon
                Réponse ← inacceptable
          Finsinon
       Finsi
 fin
```

3.5. Avantages de notre approche

L'approche que nous avons choisi, a des caractéristiques particulières, qui ont été adaptés afin de réaliser l'équilibrage de la charge de requêtes de lecture seule, à travers la décomposition de cette requête puis la vérification de son type. Parmi les principales caractéristiques, on distingue :

- Mise en œuvre simple de l'algorithme.
- Facilité de traitement des requêtes de clients.
- La vitesse de réponse aux clients.
- Algorithme simple, non-complexe et rapide.

3.6. Conclusion

Durant ce chapitre, nous avons abordé le problème d'équilibrage de charge dans les requêtes de lecture dans un système de réplication de donnée, nous avons présenté au départ, l'architecture itératif générale de ce problème, suivi par une explication du principe de fonctionnement à chaque section de cette architecture, ensuit, nous avons élaborer une structure générale détaillée du fonctionnement de serveur d'équilibrage de charge, ainsi, les opérations effectuées par le serveur d'équilibrage de charge sont également montrées, finalement, nous avons donné l'algorithme générale utilisé dans notre application ainsi que leur avantages à propos du processus d'équilibrage de charge. Dans le dernier chapitre, nous allons implémenter notre proposition et effectuer quelque testes pour valider cet approche.

Chapitre 4

Implémentation du système d'équilibrage de charge

4.1. Introduction

Ce chapitre est la dernière étape conduisant à la réalisation de notre travail. Nous allons présenter d'abord les outils d'implémentation utilisés dans notre travail, tels que MySQL, Java et Wampserver. Ensuite, nous allons configurer le système de réplication de donnée sous MS-DOS. Finalement, nous allons exécuter l'application sous l'environnement JAVA dans un réseau local, nous allons donner quelques images illustratives sur cette exécution.

4.2. Les outils de développement utilisés

Afin de développer notre projet, nous avons eu recours à un poste configuré en tant que serveur est utilisé en même temps par le client.

Les outils utilisés sont:

1. Un micro-ordinateur portable de marque HP (Intel), Core (TM) i3-3110M CPU @ 2.40 GHz
2. Un système de gestion de données MySQL
3. L'outil de programmation Java
4. L'outil WAMPS version 2.0 (PHP 5.3.0, phpMyAdmin 3.2.0.1, Apache 2.2.11, MySQL5.1.36).

4.2.1. Système de gestion de base de données MySQL:

MySQL dérive directement de SQL (Structured Query Language): le langage standard pour les traitements de bases de données, MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server.

4.2.1.1. Caractéristiques de MySQL: MySQL bénéficie d'un large public, parmi ses caractéristiques principales nous citons:

5. Sa syntaxe simple en fait un langage facile à comprendre pour les programmeurs et les débutants
6. MySQL fonctionne sur de nombreuses plates –formes différentes
7. MySQL Dispose d'une vaste bibliothèque de fonctions

8. MySQL a une grande capacité de stockage

9. MySQL dispose d'une assistance technique importante

4.2.2. Langage Java:

4.2.2.1. Qu'est-ce que la technologie Java?: Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, par la société Sun Microsystems en 1995.Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour.

Importance de Java: Java est un langage rapide, sécurisé et fiable. Sa technologie est présentée sur tous les fronts, sur des ordinateurs portables aux centres de données, sur des consoles de jeux aux superordinateurs scientifiques, et sur des téléphones portables à Internet.

4.3. Configuration du système de réplication

L'architecture minimale pour implémenter la réplication est composée de deux machines. Il est tout à fait possible d'implémenter la réplication entre deux instances de bases de données placées sur la même machine mais l'intérêt est limité.

Dans le cas présent, nous aurons la machine "MASTER" servant de maître) et "SLAVE" pour l'esclave. C'est deux machines sont installées sous windows7/xp. Pour le moment nous allons considérer que ni la base de maître, ni la base de l'esclave ne sont installées sur les machines.Nous allons utiliser dans ce travail la version 5.6.12 de MySQL.

4.3.1. Réplication avec MySQL: La première chose consiste à installer la base, nous devons créer un utilisateur spécial qui va contrôler la réplication. Il doit pouvoir accéder à la base maître depuis la machine où se trouvera l'esclave.

4.3.2. Configuration sur le maître: Cette configuration passe par les étapes suivantes:

A- Nous attribuons à cet utilisateur "repl" les droits FILE et REPLICATION SLAVE.

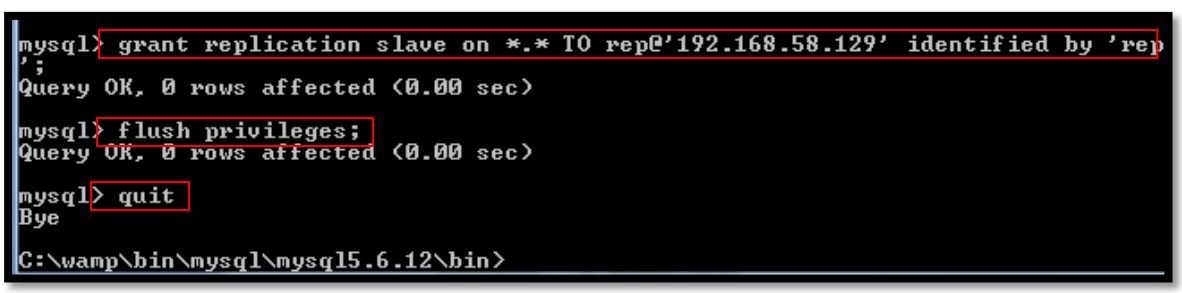

Figure 4.1: Les commandes pour le travail maître

B- Il faut maintenant sauvegarder les données de la base afin de pouvoir les transférer, plus tard, à l'esclave. Pour savoir où se trouve le répertoire de données, regarder l'entrée data dir dans le fichier de configuration de la base (C:\wamp\bin\mysql\mysql5.6.12\ my.cnf).

Avant de pouvoir relancer la base, on doit modifier le fichier de configuration en ouvrant ce fichier avec l'éditeur Notepad et on ajoute les deux lignes suivantes à la fin de la section [mysqld] :

server-id = 1

log-bin = mysql-bin

C- Redémarrez wampmysqld.

Menu Démarrer > Panneau de configuration > Outils d'administration >services, puis en sélectionnant Serveur wampmysqld

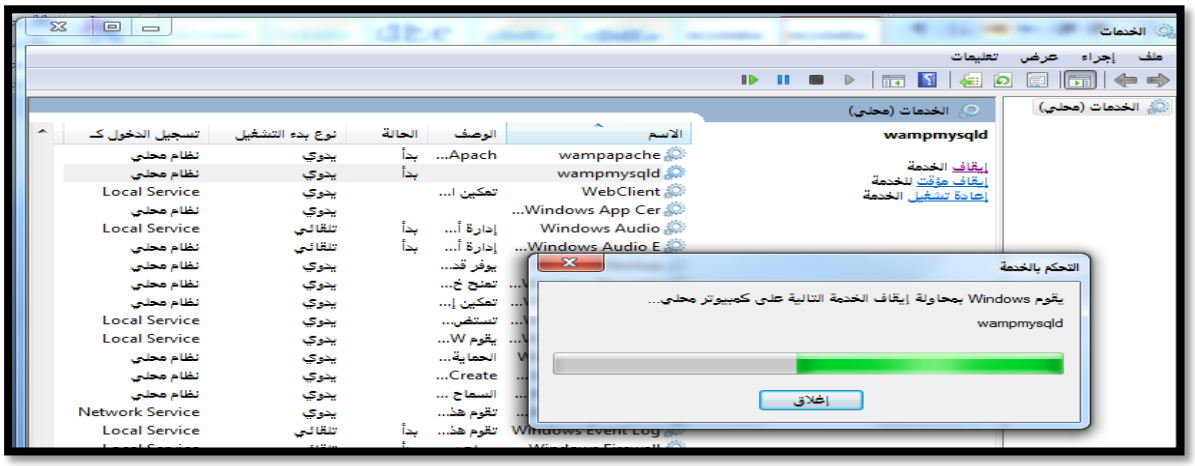

Figure 4.2: Interface redémarrage wampmysqld

D- Nous Utilisons la commande suivante pour assurer que personne ne peut écrire dans la base de données maître lors d'un vidage de la base de données. A noter également le nom du fichier et la position du log binaire, car vous aurez besoin de ces valeurs pour terminer la configuration de la réplication sur SLAVE.

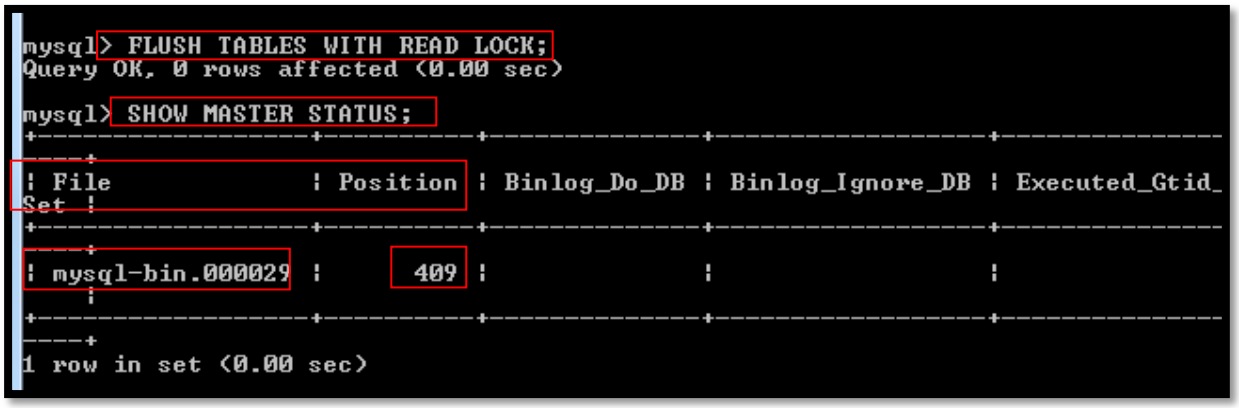

Figure 4.3: Commandes pour extraire le nom du fichier et la position du log binaire

E- On esclave le verrou en lecture depuis le maître

UNLOCK TABLES déverrouillera automatiquement tous les verrous posés par le thread courant.

Figure 4.4: Commande UNLOCK TABLES

4.3.3. Configuration sur esclave:

1. on modifie le fichier my.cnf en ajoutant les entrées suivantes:

[MySQL]:

server-id = 2

log-bin = MySQL-bin

2. on Redémarre wampmysqld.

Menu Démarrer > Panneau de configuration > Outils d'administration > services, puis en sélectionnant Serveur wampmysqld

3. la suite des étapes de réplication esclave:

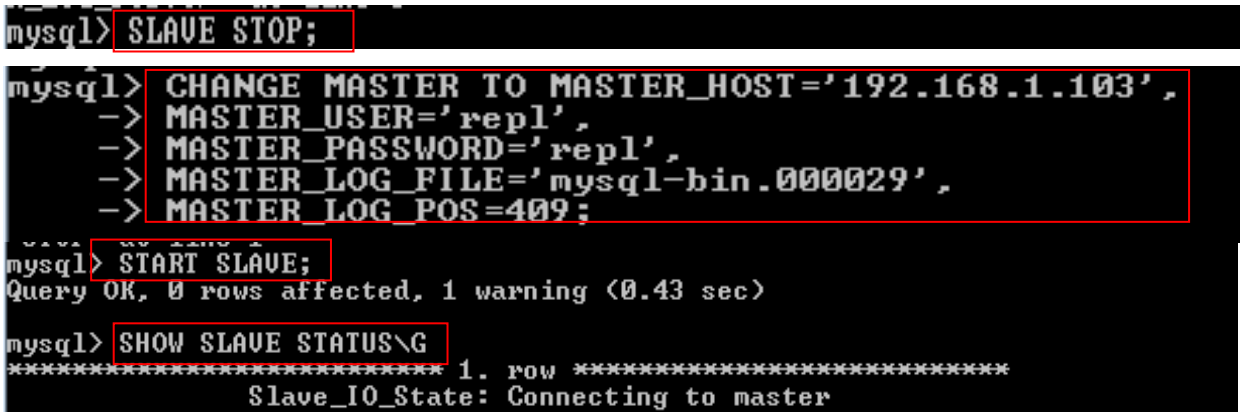

Figure 4.5: Les commandes pour le travail esclave

4.4. Le serveur d'équilibrage de charge

- Interface de contrôle serveur d'équilibrage de charge

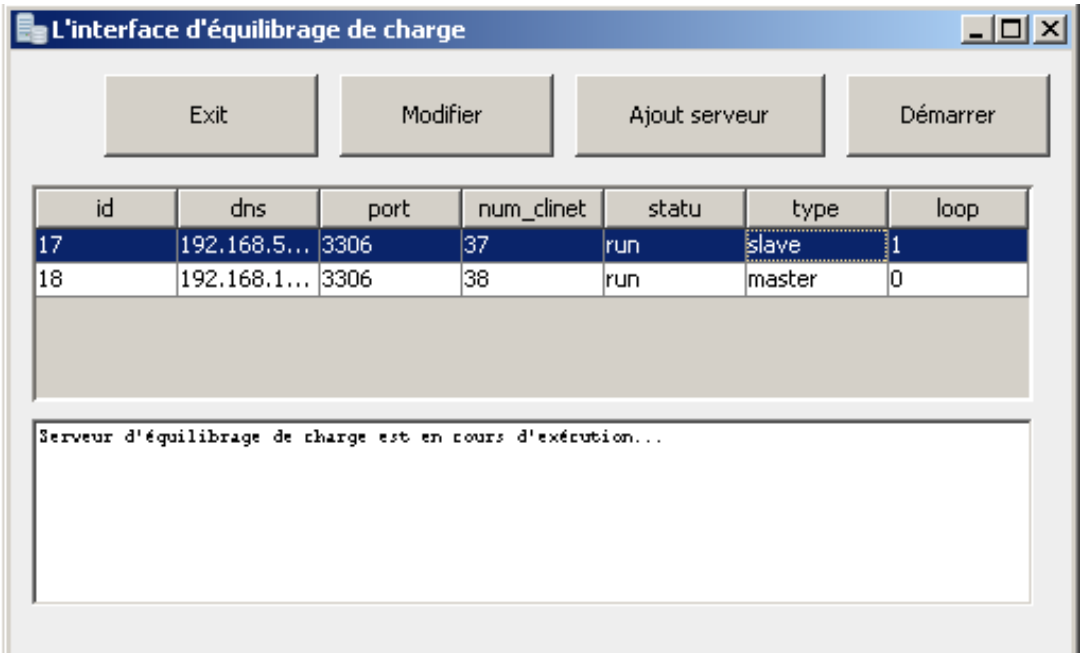

Figure 4.6: Interface serveur d'équilibrage de charge

1. Ajout d'un serveur esclave ou maître:

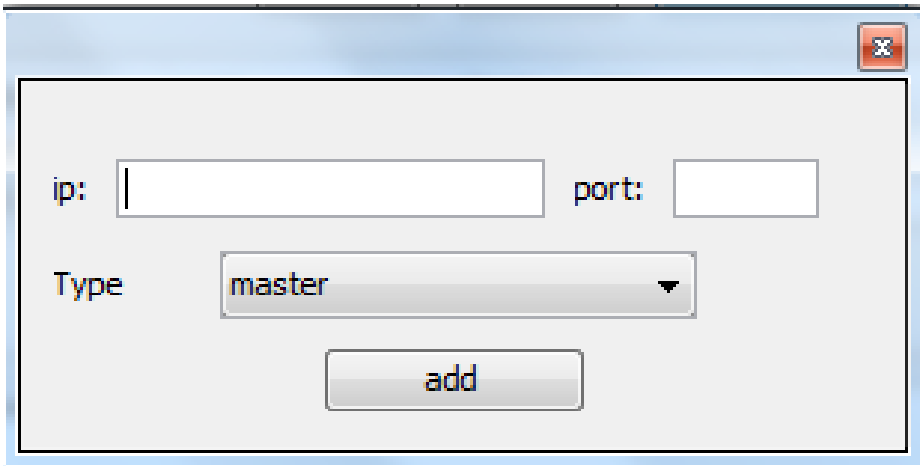

Figure 4.7: L'interface Ajout d'un serveur

4.5. Test d'application

Dans cette étape nous allons tester la validité de notre application comme suit:

Client 1:

Ce client permet d'envoyer les requêtes show databases puis il attendre des réponses comme suit:

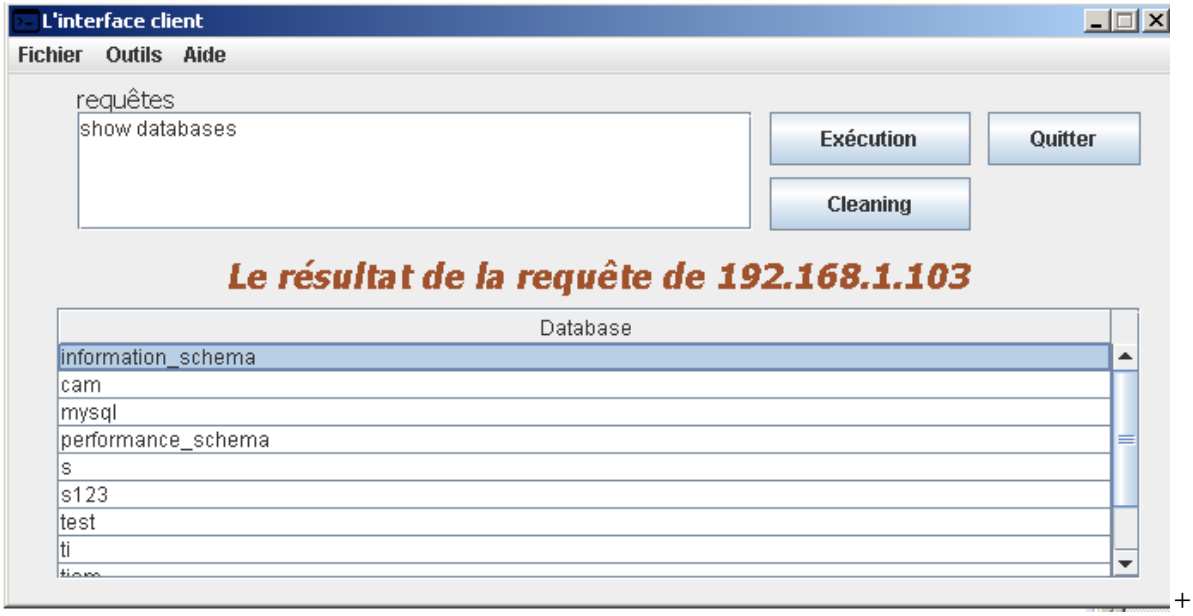

Figure 4.8: L'interface test de client 1

Client 2:

Ce client peut envoyer les requêtes SELECT * FROM test.table puis il attend les réponses comme suit:

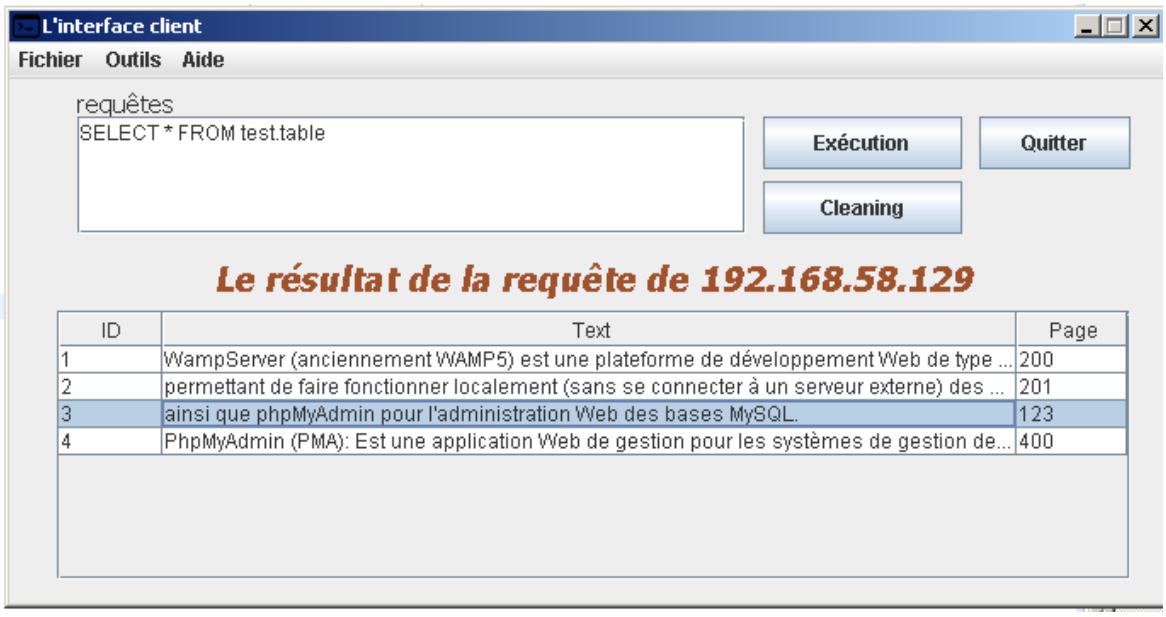

Figure 4.9: L'interface test de client 2

4.6. Conclusion

Ce chapitre a été consacré à l'implémentation de la solution proposé dans le chapitre précédant, nous avons présenté au premier lieu les outils de développement ainsi que la configuration de serveurs ensuite, nous avons testé notre application sur un système simple de réplication de donnée sous l'environnement Java.

Conclusion générale

Conclusion Générale

En informatique, le problème de l'équilibrage de charge (Load Balancing) consiste à allouer des données au départ, puis éventuellement à les redistribuer sur un ensemble d'ordinateur d'un groupe afin de minimiser leur temps de traitement. Ce problème peut être classifié en deux sous-catégories : l'équilibrage de charge dynamique et l'équilibrage de charge statique. Bien que nous avons abordé principalement l'équilibrage de charge dynamique dans ce travail.

En effet, la réplication des données permet d'offrir en même temps, une haute disponibilité de données, une répartition de charge entre les nœuds du système et une accélération du temps de traitement des requêtes (de consultation) en les exécutant en parallèle. Cependant, la charge sur les nœuds du système reste déséquilibre en dispose des nœuds plus chargée que les autres. Notre objectif était double, d'une part, proposer une solution efficace d'équilibrage de charge pour les requêtes de lecture en distribuant la charge de traitement sur plusieurs serveurs, et d'autre part, de diminuer la charge sur un seul nœud et de gagner du temps d'exécution des requêtes.

Afin de parvenir à cet objectif, notre travail s'est effectué en plusieurs étapes que nous avons présentées tout au long de ce document et que nous résumons dans ce qui suit:

Nous nous sommes intéressés au départ, à la présentation des notions de base d'un sur les systèmes d'équilibrage de charge, ainsi qu'un bref aperçu sur les algorithmes utilisés a également présenté. Afin d'éclaircir ce document, nous avons présenté en seconde partie, la base de donnée distribuée, la redondance des données ainsi que les systèmes de réplication de données. Nous nous pencherons en troisième partie vers l'étude de notre problème, d'où, nous avons commencé par la présentation de l'architecture générale d'un processus d'équilibrage de charge pour les requêtes de lecture dans un système de réplication des données, nous avons expliqué en détail les déférents algorithmes développés dans ce sens. Cet algorithme est simple et rapide et permet de facilité le traitement des requêtes des clients.

Nous avons, dans une dernière étape, réalisé notre travail à travers une application développée sous l'environnement Jave, afin de la-exécuter au premier lieu dans un ordinateur, puis sur un réseau local au seconde lieu.

Comme perspectives, nous envisageons d'implémenter cet algorithme dans les grappes pour améliorer sa performance.

Bibliographie

Bibliographie

[1]: Hennane.H, Belbachir.I, 'Equilibrage de charge dans les systèmes distribués', mémoire de Master en informatique, Université Abou Bakr Belkaid-Tlemcen, 2015.

[2]: Jean-Luc Dekeyser, Philippe Marquet, 'Équilibrage de charge pour systèmes temps-réel asymétriques sur multi-processeurs', mémoire de DtyEA d'informatique, Université des sciences et technologies de Lille- France, 2004.

[3]: Renard.H, 'Équilibrage de charge et redistribution de données sur plates-formes hétérogènes', mémoire de doctorat en informatique, université de Lyon, 2005

[4]: Rahmouni.K, 'Modèle Distribué d'Equilibrage de Charge pour les Grilles de Calcul', mémoire de magister en informatique, Université d'Oran, 2008.

[5]: Yagoubi.B, 'Modèle d'Équilibrage de Charge pour les Grilles de Calcul', mémoire de Master en informatique, Université d'Oran (Es-sénia) ,2008.

[6]: Rachida AOUDJIT, 'Répartition et Equilibrage de Charges dans les hôtes mobiles', mémoire de magister en informatique, Université moumoud mammeri de tizi ourou.

[7]: Madi.H, 'Conception et réalisation d'une base de données repartie sous oracle: cas de l'hébergement des résidences universitaires', mémoire de Master, Université A/Mira - Bejaia, 2009.

[8]: Biley.R, 'Mise en place d'un système de réplication de base de données entre sites distants', mémoire de Licence en Professionnelle ès Génie Logiciel, Université de Dschang, 2009.

[9]: Iskounen.S, Messaoudi.B, 'Réplication des bases de données', site slideshare, 2014.

[10]: Nkongolo.D, 'Etude d'une réplication symétrique asynchrone dans une base de données répartie. Application à l'enrôlement des électeurs', mémoire de Licence, Université de Kinshasa, 2011.

[11]: https://fr.wikipedia.org/wiki/Répartition_de_charge, 30 juillet 2015

[12]: https://fr.wikipedia.org/wiki/Grappe_de_serveurs, 30 décembre 2015

[13]: https://fr.wikipedia.org/wiki/%C3%89quilibrage_de_charge_des_serveurs_Web#.C3.89volution_des_usages, 5 février 2016

[14]: https://www.academia.edu/7594906/Memoire_Logo_Algorithmes_de_Replication_des_D onn%C3%A9es, 2014.

[15]: http://greg.rubyfr.net/pub/?page_id=20, 17 février 2006

[16]: http://docplayer.fr/6796098-Replication-et-durabilite-dans-les-systemes-repartis.html, 19 Février 2001

[17]: https://fr.wikipedia.org/wiki/R%C3%A9plication_(informatique), 11 novembre 2015