Intérrogation des bases de données dans le cluster sans sgbd

DOUZI salah eddine

2012-2013

Résumé

On s'est intéressé dans le cadre de ce PFE à étudier quelques méthodes d'interrogation de base de données afin de montrer l'intérêt de MPI pour supporter le parallélisme dans un contexte distribué. L'outil MPI nous a permis d'atteindre nos objectifs, à savoir l'implémentation des requêtes : Selection, Jointure et Mise à jour. Nous avons proposé également un petit système de gestion de base de données basé sur la notion de clustering.

This thesis focuses on few methods for querring databases to show the benifit of MPI and to support the parallelism in a distributed context. The MPI allowed us to acheive our goals, execute the implementation of querries : Select, Join and update. We also proposed a database system management based on clustering.

Remerciements

En premier lieu je remercie Dieu, le tout puissant pour ses faveurs et ses gratitudes, de m'avoir donner le courage et la patience pour avoir mener ce travail durant toute cette année.

De plus, mes remerciements s'adressent à Monsieur **Smahi Mohammed**, Enseignant à l'Université Abou Bakr Belkaid-Tlemcen, pour m'avoir fait l'honneur de m'encadrer et guider durant cette année.

En effet, je lui est infiniment reconnaissant de m'avoir encouragé et soutenu durant ce travail j'ai ainsi pu apprécier sa rigueur scientifique, son recueil, ses grandes qualités humaines et son oeil critique qui m'as été trés précieux pour structurer mon travail et améliorer sa qualité. Qu'il soit persuadé de mon plus profonde considération et plus grand respect.

Egalement, je tien à remercie Monsieur **Ladjel Bellatreche** professeur, directeur du LISI / ENSMA dans l'université de poitiers pour son aide et son soutien.

Enfin, un grand remerciement destiné à l'ensemble de nos enseignants et enseignantes qui ont contribué à ma formation, depuis le cycle primaire jusqu'au cursus universitaire.

Dédicace

A mes parents, aux être qui sont les plus chères au monde et auxquels je ne saurais jamais exprimer ma gratitude et ma reconnaissance en quelques lignes, je les dédié ce modeste travail, que dieu le tout puissant les protège.

Pour votre amour, votre affection et votre soutien, pour votre courage et votre sacrifice, je vous dédié, pour la deuxième et mille fois, mes très chers parents, un résultat modeste de la bienveillance et vos longes années de patience.

A mes chers frères et soeurs, à ma grand mère, à ma femme, à mes chers amis et à tous qui j'aime et qu'ils m'aiment... ou qu'ils soient.

Table des matières

Table des figures

Introduction générale

Dans le cadre de bases de données, l'interrogation joue un rôle majeur, elle permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour, tout en s'assurant des droits accordés à ces derniers. Cela est d'autant plus utile que les données informatiques sont de plus en plus nombreuses.

Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau. L'avantage majeur de l'utilisation de bases de données est la possibilité de pouvoir être accédées par plusieurs utilisateurs simultanément.

Afin de pouvoir contrôler les données ainsi que les utilisateurs, le besoin d'un système de gestion dédié à la gestion des données s'est vite fait ressentir. Les Systèmes de Gestion de Bases de Données (SGBD) remplacent les anciennes organisations où les données, regroupées en fichiers, restaient liées à une application particulière. Mais la majorité des système de gestion qui existe utilisent une seule machine pour être installé, la nécéssité d'un environnement parallèle d'une façon coopérative dans le but d'accroitre la puissance disponible pour réaliser des requêtes sous forme d'un calcul distribué, ce qui a permis au monde de la recherche de s'intéresser aux réseaux de clusters, au peer to peer et à n'importe quel moyen d'utiliser toute puissance de calcul pourra être accessible.

Dans le contexte d'exécution, différents problèmes sont apparues tel que ceux qui peuvent engendrer l'hétérogénéité, la panne des machines et la non-fiabilité du réseau reliant ces machines. De nombreuses API permettant le déploiement et la résolution de ces problèmes tels que MPI, PM2, JACE ..., afin de répondre aux nouvelles contraintes iduites par ces contextes d'exécutions.

L'objectif de ce travail consiste en premier temps de montrer l'intérêt de clustering dans le domaine d'interrogation des bases de données, le premier chapitre de mon rapport aborde une brève présentation de l'interrogation de bases de données via les SGBD, le deuxième chapitre met le point sur l'importance des grilles de calculs. Le dernier chapitre présente les requêtes implémentées, nous présentons également l'environnement MPI qui permet d'exécuter ce type d'applications parallèles.

Chapitre 1

Interrogation de base de données via SGBD

1.1 Introduction

Depuis la nuit des temps, les bons commerçants ont soigneusement enregistré leurs opérations commerciales dans carnets et tablettes. Leurs fichiers clients faisaient leur richesse. Ils ne pouvaient gérer leurs informations que sur des supports « écrits ». Aujourd'hui, depuis l'avènement des outils de l'informatique, ce stockage des données devient de plus en plus complet. Il est donc nécessaire de bien les organiser afin d'en ressortir les informations les plus pertinentes : la richesse de leur relation client. L'information est stockée dans des fichiers (tables). Ces derniers sont organisés de façon à répondre efficacement aux attentes commerciales et de gestion.

1.2 Les bases de données

Définition 1 : Une base de données est un ensemble structuré de données enregistrées dans un ordinateur et accessibles de façon sélective par plusieurs utilisateurs[1].

Définition 2 : Une Base de données est un gros ensemble d'informations structurées mémorisées sur un support permanent[2].

Définition 3 : Une base de données est un recueil d'informations liées à un sujet donné[3].

La figure 1.1 montre un exemple d'une base de données simple, et la figure 1.2 un exemple d'une autre un peu complexe.

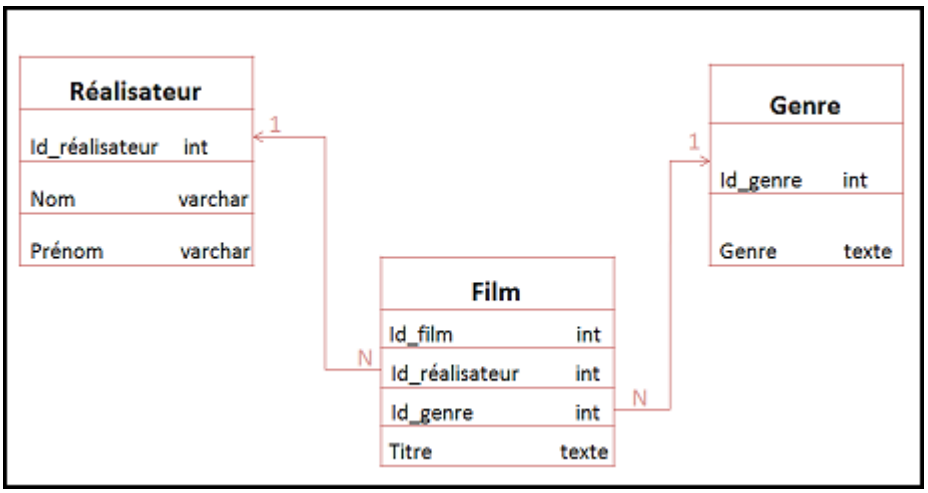

FIGURE 1.1 – Exemple 1 base de données.

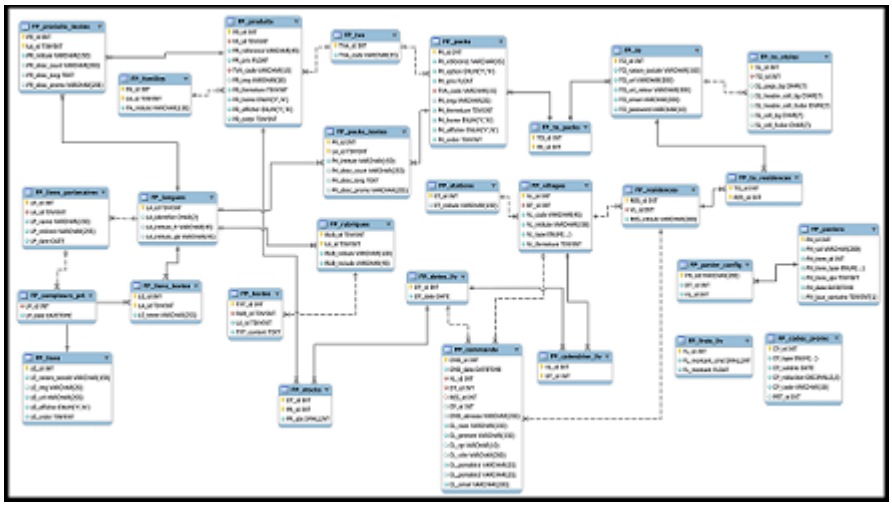

FIGURE 1.2 – Exemple 2 base de données.

1.3 L'intérrogation des bases de données

Aprés avoir concevoir une base de données, il est nécéssaire de trouver un moyen pour interroger ces données d'une façon rapide et optimale, le processus d'interrogation fait donc appel à la notion de requête, cette dernière est la résultante de **LMD**. *LMD = langage de programmation + langage d'interrogation.*

Langage de programmation : il se caractérise par :

- enchaînement d'instructions (itérations, conditionnelles, appel à des procédures ou des fonctions).
- affectation, saisie, impression.
- calcul d'expressions.
- manipulation de structures de données élaborées.

Langage d'interrogation : on destingue deux langage :

- langage de désignation (sélection, projection, jointure).
- langage de mise à jour (insertion, modification, suppression).

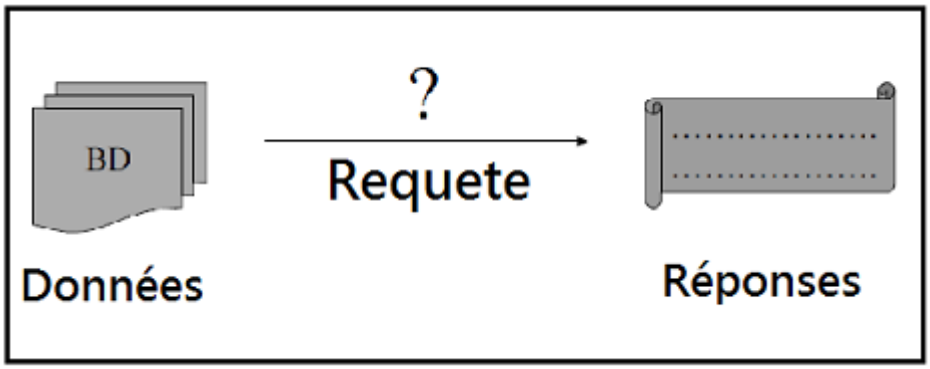

FIGURE 1.3 – Interrogation.

1.4 Les différents types de SGBD

Définition 1 : Un SGBD est un Système de Gestion de Base de Données. Le système gère des informations structurées, ce qui facilite la mise à jour (insertion, modification, suppression, consultation) de l'ensemble des informations. Cet ensemble est appelé 'base'[1].

Définition 2 : Un Système de Gestion de Bases de Données (SGBD) est un logiciel de haut niveau qui permet de manipuler les informations stockées dans une base de données. La complexité d'un SGBD est essentiellement issue de la diversité des techniques mises en oeuvre, de la multiplicité des composants intervenant dans son architecture, et des différents types d'utilisateurs (administrateurs, programmeurs, non informaticiens, ...) qui sont confrontés, à différents niveaux, au système[2].

Les systèmes de gestion de base de données sont des logiciels universels, indépendants de l'usage qui est fait des bases de données. Ils sont utilisés pour de nombreuses applications informatiques, notamment les guichets automatique bancaires, les logiciels de réservation, les bibliothèques numériques les logiciels d'inventaire, les progiciels de gestion intégrés ou la plupart des blogs et sites web. Il existe de nombreux systèmes de gestion de base de données. En 2008, Oracle détenait près de la moitié du marché des SGBD avec MySQL et Oracle Database. Vient ensuite IBM avec près de 20%, laissant peu de place pour les autres acteurs. Les SGBD sont souvent utilisés par d'autres logiciels ainsi que les administrateurs ou les développeurs. Ils peuvent être sous forme de composant logiciel, de serveur, de logiciel applicatif ou d'environnement de programmation. En 2011 la majorité des SGBD du marché manipulent des bases de données relationnelles.

Les SGBD sont les logiciels intermédiaires entre les utilisateurs et les bases de données. Une base de données est un magasin de données composé de plusieurs fichiers manipulés exclusivement par le SGBD. Ce dernier cache la complexité de manipulation des structures de la base de données en mettant à disposition une vue synthétique du contenu.
L'ensemble SGBD et base de données est destiné à permettre le stockage de données d'une manière offrant de nombreux avantages par rapport à un enregistrement conventionnel dans des fichiers. Il permet d'obtenir et de modifier rapidement des données, de les partager entre plusieurs usagers. Il garantit l'absence de redondance, l'intégrité, la confidentialité et la pérennité des données tout en donnant des moyens d'éviter les éventuels conflits de modification et en cachant les détails du format de fichier des bases de données.
Les données sont enregistrées sous forme de suites de bits représentant des lettres, des nombres, des couleurs, des formes,... Le SGBD comporte différents mécanismes destinés à retrouver rapidement les données et de les convertir en vue d'obtenir des informations qui aient un sens.

1.4.1 Fonctionalités

Un SGBD permet d'enregistrer des données, puis de les rechercher, de les modifier et de créer automatiquement des compte-rendus (anglais report) du contenu de la base de données. Il permet de spécifier les types de données, la structure des données contenues dans la base de données, ainsi que des règles de cohérence telles que l'absence de redondance.
Les caractéristiques des données enregistrées dans la base de données, ainsi que les relations, les règles de cohérence et les listes de contrôle d'accès sont enregistrées dans un catalogue qui se trouve à l'intérieur de la base de données et manipulé par le SGBD. Les opérations de recherche et de manipulation des données, ainsi que la définition de leurs caractéristiques, des règles de cohérence et des autorisations d'accès peuvent être exprimées sous forme de requêtes *(anglais query)* dans un langage informatique reconnu par le SGBD. SQL est le langage informatique le plus populaire, c'est un langage normalisé de manipulation des bases de données. Il existe de nombreux autres langages comme le Databasic de Charles Bachman, Dataflex, dBase ou xBaseScript (etc.). Les bases de données peuvent être d'une taille de plusieurs téraoctets ; une taille supérieure à la place disponible dans la mémoire centrale de l'ordinateur. Les bases de données sont enregistrées sur disque dur, ces derniers ont une capacité supérieure, mais sont moins rapides, et le SGBD est équipé de mécanismes visant à accélérer les opérations. Les SGBD contemporains enregistrent non seulement les données, mais également leur description, des formulaires, la définition des compte-rendus, les règles de cohérence, des procédures ; ils permettent le stockage de vidéos et d'images. Le SGBD manipule les structures complexes nécessaire à la conservation de ces informations.
Les SGBD sont équipés de mécanismes qui effectuent des vérifications à l'insu de l'utilisateur, en vue d'assurer la réussite des transactions, éviter des problèmes dus aux accès concurrents et assurer la sécurité des données :

Transactions : une transaction est une opération unitaire qui transforme le contenu de la base de données d'un état A vers un état B. La transformation peut nécessiter plusieurs modifications du contenu de la base de données. Le SGBD évite qu'il existe des états intermédiaires entre A et B en garantissant que les modifications sont effectuées complètement ou pas du tout. En cas de panne survenue durant des opérations de modification de la base de données, le SGBD remet la base de données dans l'état ou elle était au début de la transaction (état A).

Concurrence : la base de données peut être manipulée simultanément par plusieurs personnes, et le contrôle de la concurrence vérifie que ces manipulations n'aboutissent pas à des incohérences. Par exemple dans un logiciel de réservation, le SGBD vérifie que chaque place est réservée au maximum par une personne, même si des réservation sont effectuées simultanément.

Sécurité de données : le choix de permettre ou d'interdire l'accès à des données est donné par des listes de contrôle d'accès, et des mécanismes du SGBD empêchent des personnes non autorisées de lire ou de modifier des données pour lesquelles l'accès ne leur a pas été accordé.

1.4.2 Typologie

Selon leur construction et les possibilités qu'ils offrent les SGBD peuvent être dit hiérarchique, relationnels, orienté objet, objet-relationnel, XML/RDF ou mixte. Ils peuvent être distribués, centralisés ou embarqués et peuvent être spatials. Ils se différencient également par la taille des bases de données qu'ils peuvent manipuler. En 2010 la majorité des SGBD sont de type relationnel : ils manipulent des bases de données conformément au modèle de données relationnel.

Relationnel : Selon ce modèle, les données sont placées dans des tables avec lignes et colonnes et n'importe quelle donnée contenue dans la base de données peut être retrouvée à l'aide du nom de la table, du nom de la colonne et de la clé primaire. Le modèle relationnel est destiné à assurer l'indépendance des données et à offrir les moyens de contrôler la cohérence et d'éviter la redondance. Il permet de manipuler les données comme des ensembles en effectuant des opérations de la théorie des ensembles. Les règles de cohérence qui s'appliquent aux bases de données relationnelles sont l'absence de redondance ou de nul des clés primaires, et l'intégrité référentielle.

Hiérarchique : Une base de données hiérarchique est une base de données dont le système de gestion lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur. Elle a été utilisée dans les premiers systèmes de gestion de base de données de type mainframe et a été inventé par la NASA.

Orienté objet et objet-relationnel : Les SGBD orientés objet sont un sujet de recherche depuis 1980, lorsque sont apparus les premiers langages de programmation orientée objet. Ils sont destinés à offrir les fonctionnalités des SGBD à des langages orientés objet et permettre le stockage persistant des objets. Les objets sont manipulés en utilisant les possibilités natives des langages orientés objet et une interface de programmation permet d'exploiter les fonctionnalités du SGBD. Celui-ci est équipé des mécanismes nécessaires pour permettre

l'utilisation des possibilités d'encapsulation, d'héritage et de polymorphisme des langages de programmation orientée objet. Les SGBD objet-relationnel offrent à la fois les possibilités des SGBD orientés objet et ceux des SGBD relationnels.

A base de XML ou RDF : Une base de données XML Native *(NXD en anglais)* est une base de données qui s'appuie sur le modèle de données fourni par XML. Elle utilise typiquement des langages de requête XML comme XPath ou XQuery. Une extension possible est une base RDF, avec le langage d'interrogation SPARQL.

Mixte : de tels SGBD utilisent les différents paradigmes évoqués avant.

Centralisé ou distribué : Un SGBD est dit centralisé lorsque le logiciel contrôle l'accès à une base de données placée sur un ordinateur unique. Il est dit distribué lorsqu'il contrôle l'accès à des données qui sont dispersées entre plusieurs ordinateurs. Dans cette construction, un logiciel est placé sur chacun des ordinateurs, et les différents ordinateurs utilisent des moyens de communication pour coordonner les opérations. Le fait que les informations sont dispersées est caché à l'utilisateur, et celles-ci sont présentées comme si elles se trouvaient à une seule place.

Embarqué : Une base de données embarquée *(anglais embedded)* est un SGBD sous forme de composant logiciel qui peut être incorporé dans un logiciel applicatif. Contrairement à un SGBD client-serveur dans lequel un processus traite les requêtes, un modèle embarqué se compose de bibliothèques logicielles liées par liaison dynamique avec le logiciel qui utilise le SGBD. Dans ce type de SGBD, la base de données est souvent composée d'un fichier unique dont le format est identique quelles que soient les caractéristiques de l'ordinateur utilisé. Bien que le SGBD offre de nombreux avantages par rapport à un enregistrement sur fichier, ces derniers sont souvent préférés aux SGBD, qui ont la réputation d'être des logiciels lourds, encombrants et compliqués à installer.

Spatial : Les applications informatiques telles que les système d'information géographiques et les outils de conception assistée par ordinateur utilisent des SGBD spatial. Ce type de logiciel permet le stockage d'informations géométriques telles que des points, des lignes, des surfaces et des volumes. Ils comportent des fonctions permettant de retrouver une information sur la base de caractéristiques géométriques telles que les coordonnées ou la dimension. Le langage de requête du SGBD permet la manipulation d'informations de géométrie tels que lignes, point ou polygones, le SGBD met en oeuvre les algorithmes et les structures de fichiers nécessaire.

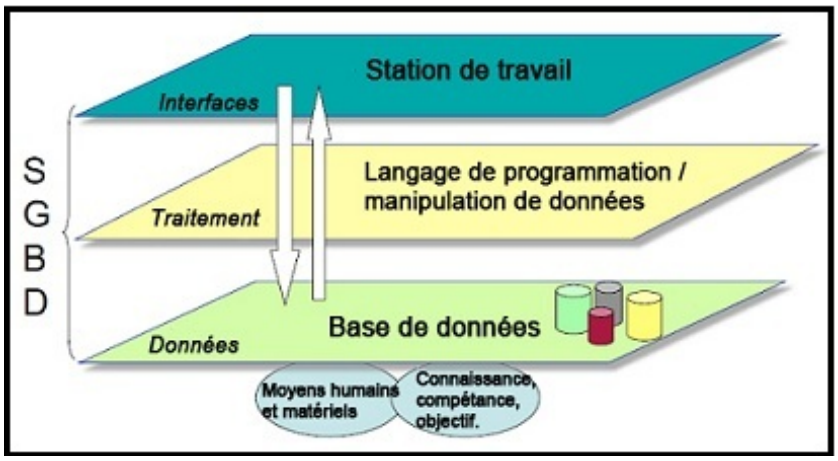

FIGURE 1.4 – SGBD.

1.5 L'intérrogation avec SGBD

1.5.1 Les langages d'interrogation

Un langage d'interrogation permet d'extraire les données d'un système de gestion de bases de données à l'aide de requêtes formulées. Parmi les langages d'interrogation qui existe, il y'a le langage standard pour les SGBD relationnels **SQL** :

SQL : SQL est un langage textuel aux règles syntaxiques précises. SQL a été développé pour le système R d'IBM. Il a été ensuite adopté comme norme de langage de manipulation de bases de données et a été repris dans la quasi-totalité des SGBD[4].
La structure de base des requêtes écrites en SQL :

```
SELECT <liste de champs>
FROM <liste de tables>
WHERE <condition>
```

Ce langage relationnel est non-procéduraux, c.-à-d. l'utilisateur indique uniquement le résultat qui l'intéresse sans préciser comment procéder pour y parvenir.SQL peut cependant posséder quelques particularités selon le SGBD utilisé[5].

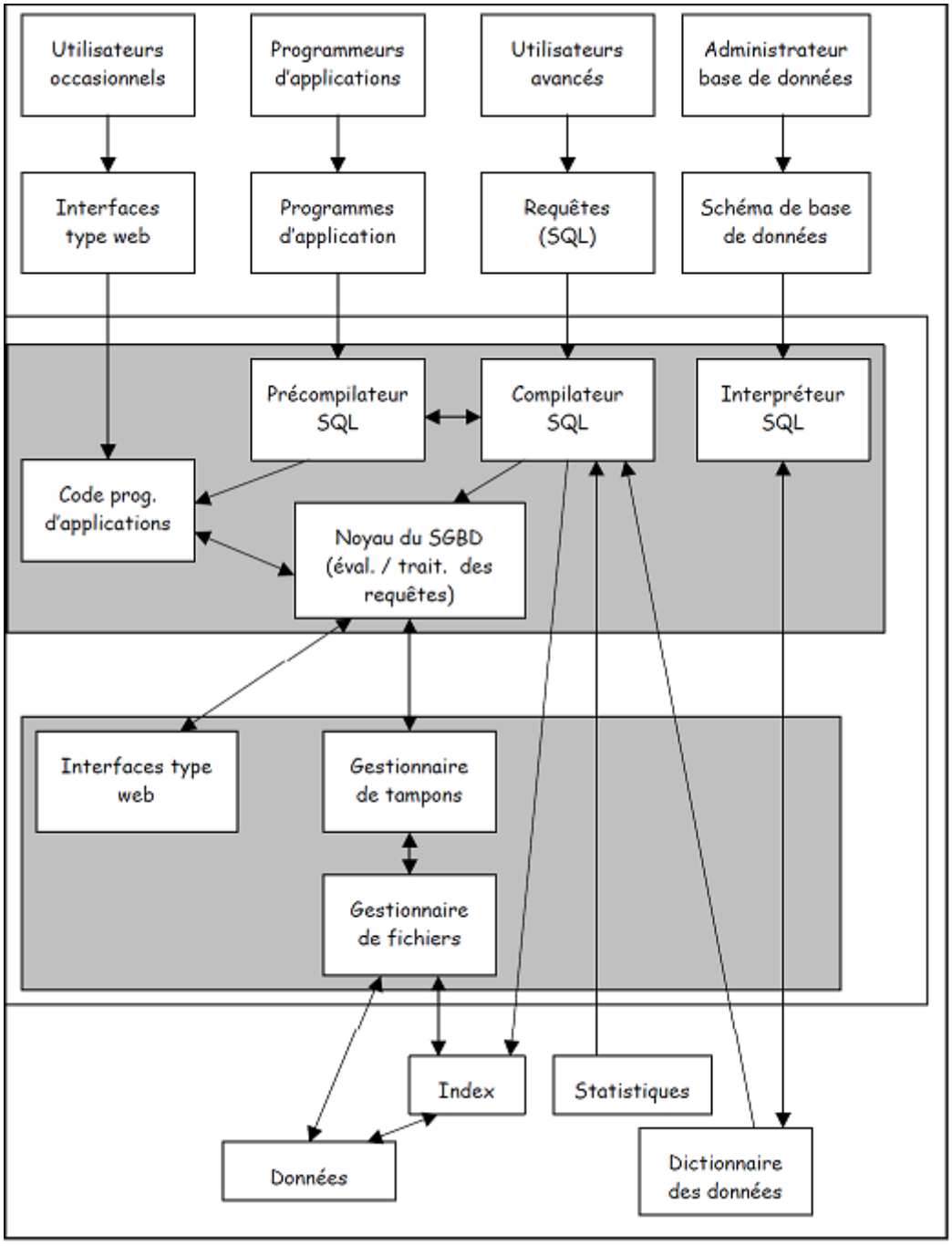

FIGURE 1.5 – Fonctionalités et organisation générale.

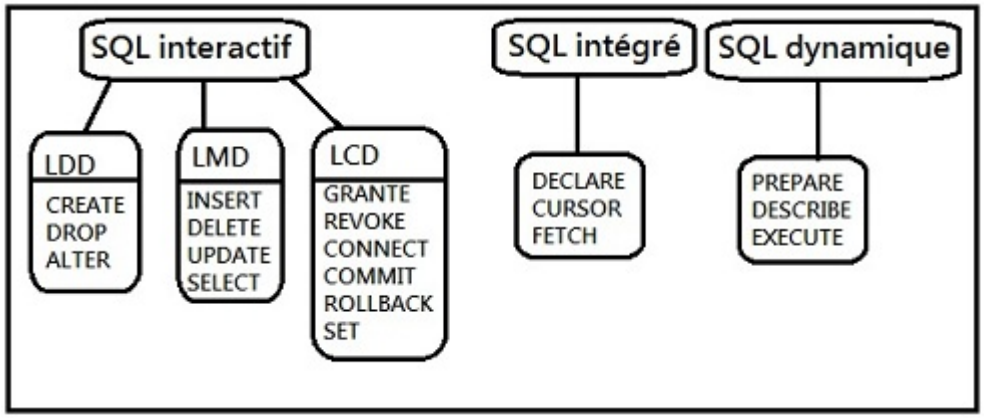

FIGURE 1.6 – Syntaxe SQL.

1.5.2 Syntaxe SQL

SQL est un langage déclaratif qui permet d'interroger une base de données sans se soucier de la représentation interne (physique) des données, de leur localisation, des chemins d'accès ou des algorithmes nécessaires. A ce titre il s'adresse à une large communauté d'utilisateurs potentiels (pas seulement des informaticiens) et constitue un des atouts les plus spectaculaires (et le plus connu) des SGBDR. On peut l'utiliser de manière interactive, mais également en association avec des interfaces graphiques, des outils de reporting ou, très généralement, des langages de programmation. Ce dernier aspect est très important en pratique car SQL ne permet pas de faire de la programmation au sens courant du terme et doit donc être associé avec un langage comme le C, le COBOL ou JAVA pour réaliser des traitements complexes accédant à une base de données. On peut exprimer en SQL des requêtes comme :

La projection : La projection permet de ne conserver que les champs (colonnes) intéressants.

La selection : La sélection permet de ne conserver que les enregistrements (lignes) respectant une condition vérifiée.

La jointure : La jointure est une des opérations les plus utiles (et donc une des plus courantes) puisqu'elle permet d'exprimer des requêtes portant sur des données réparties dans plusieurs tables.

Mises à jour : Les commandes de mise-à-jour (insertion, destruction, modification) sont considérablement plus simples que les requêtes.

1.6 Conclusion

Dans ce chapitre nous avons présenté le langage SQL d'interrogation et de manipulation de données (insertion, mise à jour, destruction). La syntaxe est celle de la norme SQL3, implantée dans la plupart des principaux SGBDR. La majorité des SGBD utilisent un précompilateur LDD pour produire des codes objets des programmes, un processeur de requêtes, un compilateur LDD, un gestionnaire de BD, ce dernier communique avec un gestionnaire de fichiers en cas de fichiers de données, ou directement avec un dictionnaire de données qui se situe dans le stockage disque.

Chapitre 2

Paraléllisme et distribution

2.1 Introduction

Depuis peu, avec le développement d'internet, on assiste à la globalisation des ressources et des données informatiques. Le réseau mondial offre aux utilisateurs des ressources « virtuellement illimitées »en les répartissant de manière dynamique sur un ensemble non figé d'équipements et en offrant un accès transparent à ces ressources.

Les applications scientifiques actuelles sont tellement gourmandes en ressources de calcul qu'il est indispensable de les exécuter de manière coopérative dans un environnement parallèle et sur des architectures spécifiques. La programmation de telles applications doit respecter des contraintes de performance et de fiabilité[6].

2.2 Systèmes distribués

2.2.1 Définition

Un système distribué est un ensemble composé d'éléments reliés par un système de communication ; les éléments ont des fonctions de traitement (processeurs), de stockage (mémoire), de relation avec le monde extérieur (capteurs, actionneurs).

Les différents éléments du système ne fonctionnent pas indépendamment mais collaborent à une ou plusieurs tâches communes. Conséquence : une partie au moins de l'état global du système est partagée entre plusieurs éléments (sinon, on aurait un fonctionnement indépendant).

Les systèmes répartis doivent supporter l'éxécution de différentes classes d'applications, ayant des exigences pouvant évouleur au cours du temps. Ces exigences

concernent les aspects temporels (temps processuer, bande passante réseau ...), mais aussi la sécurité, la protection, la tolérance aux fautes, la mobilité.

Définition (Leslie Lamport) : A distributed system is one which I can't do my work some computer has failed that I never heard of. (Un système réparti est un système qui vous empêche de travailler quand une machine dont vous n'avez jamais entendu parler tombe en panne).

La conception et la réalisation des systèmes et applications répartis s'appuient sur un ensemble de principe de base régissant la communication, la gestion de l'information, le partage des ressources, la tolérance aux fautes. Donc un système réparti (ou distribué, « distributed system ») est :

- Composé de plusieurs systèmes calculatoires autonomes (sinon, non réparti).
- Sans mémoire physique commune (sinon c'est un système parallèle, cas dégénéré).
- Qui communiquent par l'intermédiaire d'un réseau (quelconque).

2.2.2 Utilité des systèmes répartis

La répartition est un état de fait pour un nombre important d'applications.

Accès distant : un même service peut être utilisé par plusieurs acteurs, situés à des endroits différents.

Redondance : des systèmes redondants permettent de pallier une faute matérielle, ou de choisir le service équivalent avec le temps de réponse le plus court.

Performance : la mise en commun de plusieurs unités de calcul permet d'effectuer des calculs parallélisables en des temps plus courts.

Confidentialité : les données brutes ne sont pas disponibles partout au même moment, seules certaines vues sont exportées.

Besoin propres des appliactions :

- Intégration d'applications existantes initialement séparées.
- Intégration massive de ressources (Grilles de calcul, gestion de données).
- Pénétration de l'informatique dans des domaines nouveaux d'application.

2.2.3 Exemples de systèmes distribués

- Le circuit électronique de la voiture moderne est composé d'un grand nombre de microcontrôleurs, chacun dédiés à une fonction différente (calcul de la vitesse, gestion d'ABS, calcul de la distance des obstacles).

- Système de commande des unités de traitement de gaz, de raffinage et de pétrochimie, gère un ensemble d'instruments (Compteurs, transmetteurs...) et de vannes permettant la régulation des paramètres (Température, pression...) de toute l'usine en temps réel.
- WWW, FTP, Mail.
- Guichet de banque, agence de voyage.
- Téléphones portables (et bornes).
- Télivision interactive.
- Agents intelligents.

Parmi les systèmes distribués on trouve les grilles informatiques (Grid computing) oú se révèle la puissance des calculs distribués.

2.3 Grilles informatiques

2.3.1 Définition d'une grille informatique

Le terme The grid ou grille de calcul a été introduit pour la première fois aux Etat Unis durant les années 1990 pour décrire une infrastructure de calcul distribué utilisée dans les projets de recherche scientifiques et industriels[5].

La notion de grille de calcul s'inspire fortement de la grille d'électricité (Power Grid). Donc, par analogie, une grille de calcul est définie comme étant un environnement matériel et logiciel dans lequel des machines hétérogènes, reliées par des réseaux hétérogènes, fédèrent leurs ressources à grande échelle (calcul, stockage, ...). L'accès à ces ressources (hétérogènes et distantes) et à la puissance de calcul doit être totalement transparent aux utilisateurs. Cette vitalisation d'accès aux ressources est l'un des fondements des grilles de calcul.

2.3.2 Caractéristiques des grilles de calcul

Les grilles de calcul ont des caractéristiques différentes de celles des architectures classiques. Tout d'abord, une grille de calcul est partagée par plusieurs utilisateurs indépendants. Ceci conduit à une grande variabilité dans l'utilisation des ressources mises à disposition par les grilles. De plus, la grille est répartie sur différents sites qui ne sont pas sous administration commune. Cela conduit à une grande hétérogénéité tant au niveau matériel que de l'environnement logiciel. Ainsi, chaque site d'une grille peut utiliser ces propres machines, avec son propre système d'éxploitation. Les politiques de sécurité peuvent aussi être différentes d'un site à l'autre. Les sites ne partagent pas non plus un même système de fichiers. La grille n'a pas une structure statique. que ce soit du fait de panne matérielle, de remplacement ou d'ajout, des

ressources peuvent apparaitre ou disparaitre à tout instant. Enfin, l'échelle des grilles de calcul est grande. Le nombre de processeurs d'une grille de calcul se compte par miliers. De ce fait, une grille de calcul est en mesure d'offrir des avantages que les infrastructures et les technologies actuelles ne sont pas capables d'assurer.

Par exemple :

Exploiter les ressources sous utilisées : Les études montrent que les ordinateurs personnels et les stations de travail restent inactifs la plupart du temps. Les grilles de calcul permettent ainsi d'utiliser les cycles processeurs durant lesquels les machines sont inactives afin d'éxécuter une application ou une partie d'application nécessitant une puissance de calcul importante. Les cycles processeur ne sont pas les seules ressources sous utilisées, les capacités de stockage le sont aussi. Ainsi, il est possible qu'une grille agrège des ressources de stockage afin de les partager par plusieurs utilisateurs, c'est le principe des grilles de données.

Exemple concret : Les interactions entre les différentes protéines responsable de dérèglement neuromusculaire, avec les Grid c'était un bénéficie d'une puissance de calcul de 500000 machines au lieu d'un seul ordinateur qui analyse la protéine pendant 70 jours.

Fournir un cadre distibué : Une grille de calcul peut agréger une importance quantité de ressources afin de fournir une puissance de calcul aussi performante que les gros calculateurs parallèles. Les ressources agrégées peuvent aller du simple PC à des calculateurs parallèles.

Gestion adéquate des ressources : En partageant les ressources, une grille peut fournir l'accès à des ressources spéciales comme des équipements (bras robotiques, caméras, ...) ou des bibliothèques (BLAS, LAPACK). Ainsi ces ressources seront utilisées et partagées par plusieurs utilisateurs.

Fiabilité et disponibilité des services : Les ressources fédérées par une grille de calcul sont géographiquement éloignées et disponibles en importantes quantités. Cela permet d'assurer la continuité du service si certaines ressources deviennent indisponibles. Dans ce cas, les logiciels de contrôle et de gestion de la grille sont en mesure d'allouer d'autres ressources et de leur transmettre les calculs à effectuer.

Programmer pour les grilles de calcul n'est pas une tâche aisée, encore moins pour obtenir du code efficace. Il existe principalement deux obstacles.
Premièrement l'architecture de la grille est caractérisée par une grande hétérogénéité des ressources et par l'absence d'une connaissance à priori des

ressources mises a disposition.
Deuxièmement, le grannombre de ressources et leur partage rend difficile leur gestion. Afin de faciliter la programmation de ces architectures, on trouve des outils qui cherchent à masquer cette hétérogénéité. Les services de base que l'on peut atteindre d'un intergiciel pour grille de calcul sont : la gestion des ressources, les communications, l'authentification, le démarrage d'une activité, l'accès et le stockage des données.

2.4 MPI Cluster

La réalisation d'un cluster avec MPI nécéssite au moins deux machines connéctés dans un réseau avec un système d'exploitaion ubuntu linux vesion 10.04.

2.4.1 Configuration du serveur

Avant tout, on doit avoir une adresse IP static ou résèrvée en DHCP. Ensuite quelques paquets sont aussi nécéssaires :

- libmpich1.0-dev.
- libmpich-mpd1.0-dev.
- libmpich-shmem1.0-dev.
- mpich2.
- mpich2-doc
- openssh-server.
- build-esentials.

Configuration du réseau : Un fichier nommé **hosts** dans le répertoire **etc** contient la définition de l'adresse IP de la machine et par défault est égale à **127.0.1.1**, on doit la remplacer par celle utilisée par la machine dans le réseau.

Configuration de l'utilisateur : on doit créer un utilisateur nommé **Cluster**, et ajouter le répertoire **bin** à son path.

Configuration de MPICH : cette configuration nécéssite la création d'un fichier de configuration **.mpd.conf** et un fichier **mpd.hosts** qui contient l'ensemble de machines autorisées dans notre cluster.

2.4.2 Configuration des noeuds

Cette étape se divise en deux partie, la première contient la configuration du réseau et de l'utilisateur, la deuxième contient la configuration de MPICH.
La première est la même que celle de la configuration du serveur, la deuxième doit être exécutée sur le serveur. On génère une clé RSA pour établir une connection sécurisée entre les machines.
La figure 2.1 montre le lancement du cluster.

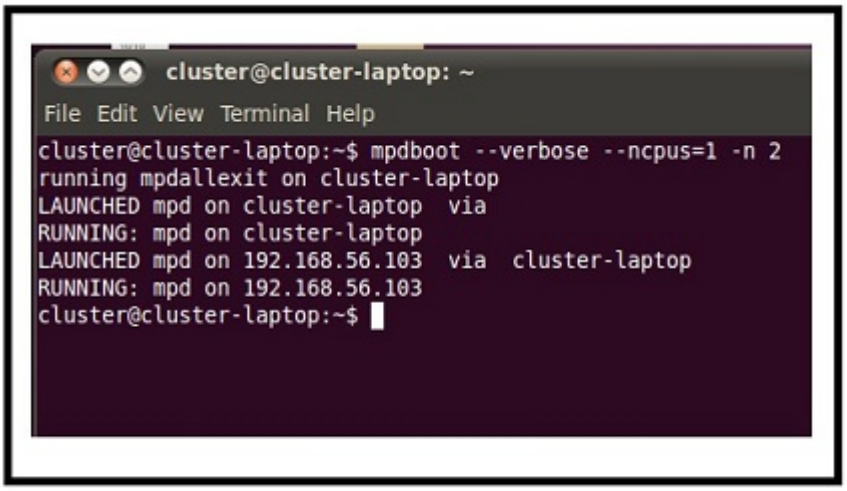

FIGURE 2.1 – Le Lancement du cluster.

2.5 Conclusion

Dans ce chapitre, nous avons vu les systèmes distribués et leurs intérêt en terme de temps et de mémoire, parmi les système distibué qui éxiste, il y'a la notion de cluster que nous avons vu un modèle de ce dernier qui est le MPI Cluster. Introduire le *clustering* permet de réduire le taux de perte du performances, améliorer le temps d'exécution et gagner l'espace mémoire. Dans le chapitre qui suit on va montrer l'intérêt de cluster pour l'intérrogation des bases de données.

Chapitre 3

Intérrogation de bases de données sans SGBD

3.1 Introduction

Dans ce chatpitre, on va voir la partie pratique de ce travail, notamment avec l'amélioration de temps d'éxecution des requettes dans le cluster à partir d'une intérrogation sans SGBD, pour ce cas, nous avons utilisé deux architectures, BDMT il s'agit d'une intérrogation d'une base de données distribué avec un même traitement, la deuxième BCTD il s'agit d'une intérrogation d'une base de données centralisée avec un traitement différent.

l'implémentation et le déploiement de ces architectures sont réalisés en utilisant l'environnement MPI qui est dédié au calcul parallèle.

3.2 Architectures

3.2.1 BDMT

La taille des bases de données qui existent augmente plus en plus parsuite, l'intérrogation de ces dernières nécessite un matériel performant en temps CPU et en espace mémoire, l'architecture BDMT répond a ce besoin en divisant la base de données sur l'ensemble des éléments du cluster, ces noeuds vont éxecuter le même traitement.

Cette architecture a pour avantage de diminuer le temps d'éxecution et une distribution personalisée de la base de données.La figure 3.1 montre le mécanisme de cette architecture.

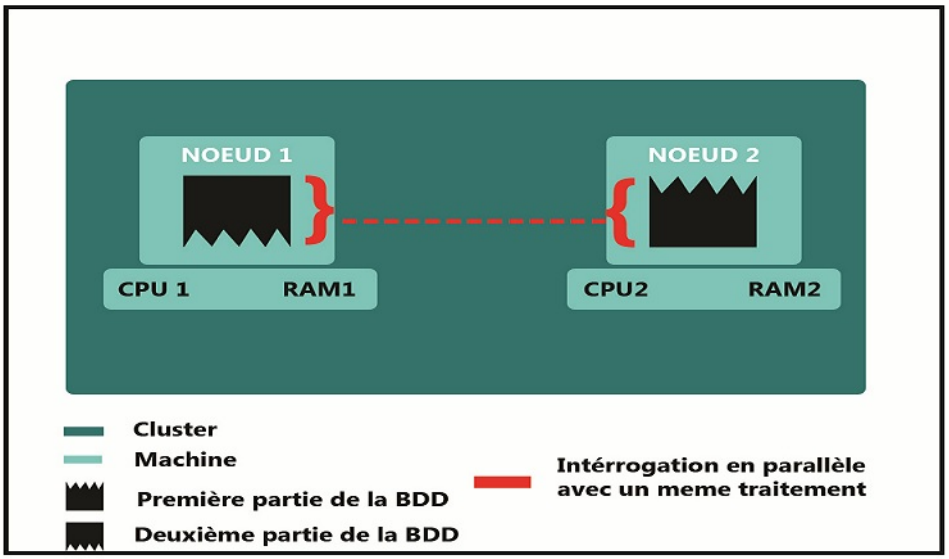

FIGURE 3.1 – Architecture BDMT

3.2.2 BCTD

L'intérrogation des bases de données diffère d'une requêtte à une autre, cette dernière a un niveau de complexité, et parsuite l'éxecution de ces requêttes devient en plus en plus longue en terme de temps et couteuse en terme de mémoire. L'architecture BCTD est utilisée pour améliorer ces deux facteurs en divisant le traîtement sur l'ensemble des élements du cluster qui va être éxecuté sur une base de données centralisée. La figure 3.2 montre le fonctionnement de cette architecture.

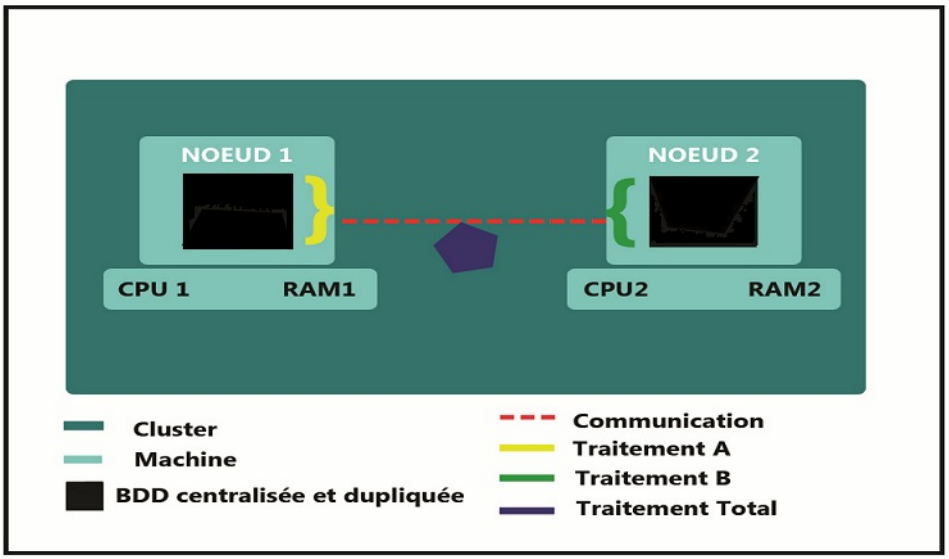

FIGURE 3.2 – Architecture BCTD

3.3 Environnement MPI

3.3.1 Introduction

Un code de calcul ou programme peut être écrit suivant deux modèles de programmation : séquentiel ou parallèle. Dans le modèle séquentiel, un programme est exécuté par un unique processus. Ce processus tourne sur un processeur d'une machine et a accès à la mémoire du processeur. Il arrive que pour certains codes de calcul, la mémoire d'un seul processeur ne suffise plus (manipulation de gros tableaux) et/ou le temps de calcul soit trop important...

Pour palier à ces problèmes, on peut avoir recours à différentes méthodes comme le raffinement de maillage adaptatif, le grid-meshing ou la programmation parallèle. On s'intéresse ici plus particulièrement à la programmation parallèle. Celle-ci permet de répartir les charges de calcul sur plusieurs processus. Il existe deux types de programmation parallèle : le MPI (Message Passing Interface) et le openMP (Multithreading). On se limitera au MPI.

Dans un modèle de programmation parallèle par échange de messages (MPI), le programme est dupliqué sur plusieurs processus. Chaque processus exécute un exemplaire du programme et a accès à sa mémoire propre. De ce fait, les variables du programme deviennent des variables locales au niveau de chaque processus. De plus un processus ne peut pas accéder à la mémoire des processus voisins. Il peut toutefois envoyer des informations à d'autres processus à condition que ces derniers (processus récepteurs) soient au courant qu'ils devaient recevoir ces informations du

processus émetteur.

La communication entre processus se fait uniquement par passage de messages entre processus (c'est à dire : envoi et reception de messages). Techniquement, cette communication se fait via des fonctions de la bibliothèque MPI appelées dans le programme. L'environnement MPI permet de gérer et interpréter ces messages.

3.3.2 Description

Pour utiliser la bibliothèque MPI, le programme source doit impérativement contenir :

1. l'appel au module MPI : include mpif.h en fortran77, use MPI en fortran90, include mpi.h en C/C + +.
2. l'initialisation de l'environnement via l'appel à la subroutine MPI_INIT(code). Cette fonction retourne une valeur dans la variable *code*. Si l'initialisation s'est bien passée, la valeur de *code* est égale à celle dans MPI_SUCCESS.
3. la désactivation de l'environnement via l'appel à la subroutine MPI_FINALIZE (code). L'oublie de cette subroutine provoque une erreur.

Une fois l'environnement MPI initialisé, on dispose d'un ensemble de processus actifs et d'un espace de communication au sein duquel on va pouvoir effectuer des opérations MPI. Ce couple (processus actifs,espace de communication) est appelé communicateur. Le communicateur par défaut est MPI_COMM_WORLD et comprend tous les processus actifs. Il est initialisé lors de l'appel à la fonction MPI_INIT() et désactivé par l'appel à la fonction MPI_FINALIZE(). On peut connaître le nombre de processus actifs gérés par un communicateur avec la fonction MPI_COMM_SIZE(*comm, nb_procs, code*) ainsi que le rang (ou numéro) d'un processus avec la fonction MPI_COMM_RANK(*comm, rang, code*).

3.3.3 Du programme source à l'éxecution

Trois étapes sont nécessaires : l'écriture du programme, sa compilation puis son exécution.

1. **L'écriture :** La figure 3.3 montre un exemple de programme en C.
2. **La compilation** du programme peut se faire par l'intermédiaire d'un Makefile. Les options de compilations dépendent du compilateur.
3. **L'exécution** du programme en interactif sur 2 processus se fait via la commande : **mpiexec -n 2 pairimpair.**
   Réponse du programme :
   Coucou, je suis le processus pair rang 0.
   Coucou, je suis le processus impair rang 1.

```c
#include "mpi.h"
#include <stdio.h>

int main(int argc, char *argv[]) {
  int rang, nb_processus;

  MPI_Init( &argc, &argv);

  MPI_Comm_rank(MPI_COMM_WORLD, &rang);
  MPI_Comm_size(MPI_COMM_WORLD, &nb_processus);

  if ( (rang % 2) == 0)
    printf("Coucou, je suis le processus pair   %d\n", rang);
  else
    printf("Coucou, je suis le processus impair %d\n", rang);

  MPI_Finalize();
  return 0;
}
```

FIGURE 3.3 – Programme (en C) processus pair et impair.

3.3.4 Communications

On distingue deux type de communications sous MPI :

Communications point à point : La communication point à point est une communication entre deux processus. L'un d'eux envoie un message (c'est l'émetteur , l'autre le reçoit (c'est le récepteur).

Communications collectives : Elles permettent de communiquer en un seul appel avec tous les processus d'un communicateur. Ce sont des fonctions bloquantes, c'est à dire que le système ne rend la main à un processus qu'une fois qu'il a terminé la tache collective.

Pour ce type de communication, les étiquettes sont automatiquement gérées par le système. On détaille quelques fonctions de communication collective.

3.3.5 Optimisation d'un programme parallèle

L'optimisation d'un code séquentiel concerne la minimisation du temps de calcul. Lorsqu'on parallélise un code, un autre temps s'ajoute au temps de calcul, c'est le temps de communication entre les processus **(temps total = temps calcul + temps communication)**. L'optimisation d'un code parallèle consiste donc à minimiser le temps de communication entre les processus. Celui-ci peut être mesuré via la fonction MPI_WTIME(). Avant de se lancer dans l'optimisation d'un programme parallèle, il faut d'abord comparer le temps de calcul et le temps de communication au temps total de simulation. Si le temps de communication est prépondérant

devant le temps de calcul, alors on peut passer à la phase d'optimisation. Cette étape consiste à réduire le temps de communication. Celui-ci contient un temps de préparation du message et un temps de transfert. Le temps de préparation contient un temps de latence pendant lequel les paramètres réseaux sont initialisés. Le reste du temps de préparation des messages (appelé aussi temps de surcoût) est lié à l'implémentation MPI et au mode de transfert utilisé (voir figure 3.4 pour plus de détails). Il existe plusieurs possibilités pour optimiser le temps de communication, parmi elles :

- recouvrir les communications par des calculs.
- limiter les modes de transfert qui utilisent la recopie du message dans un espace mémoire temporaire (buffering).
- limiter les appels répétitifs aux fonctions de communication MPI (qui coûtent cher en temps).

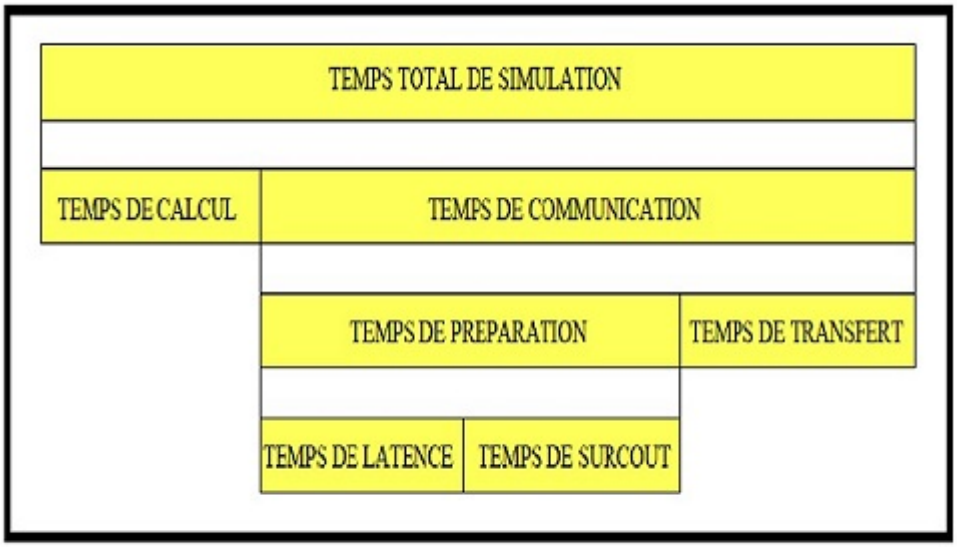

FIGURE 3.4 – Composition du total de simulation d'un programme parallèle.

3.4 Mise en oeuvre avec MPI Cluster

Notre but ultime était de réaliser une exécution distribuée des requêttes : Selection, Jointure, Réstriction et mise à jour afin de sortir avec un petit système de gestion de base de données qui introduit la notion de *clustering*, pour cela nous avons crée un petit cluster composé de deux machines d'un 4GO d'espace de la ram et 4 coeurs pour la première et 2GO et 2 coeurs pour la deuxième, elles sont reliées par un modem de quatre ports (ZTE) avec un débit de 10/100 Mbps.

Un fichier de configuration est défini dans la machine centrale afin d'éxecuter des commandes à distance, **SSH** (*Secure Shell*) et **MPICH2** ont été installés dans toutes les machines afin d'assurer le bon fonctionnement de l'exécution des programmes C avec une communication sécurisée entre les différentes entités.

3.4.1 Base de données utilisée

L'etape qui vient aprés la configuration de *cluster* est le choix de la base de données, nous avons utilisé une base de données texte qui contient trois fichiers, chacun entre eux contient une table, la table Etudiant(Id_Etud, Nom, Prénom), Enseignant(Id_Enseign, Nom, Prénom) et la table Module(Id_Mod, Id_Etud, Id_Enseign, Intitulé).

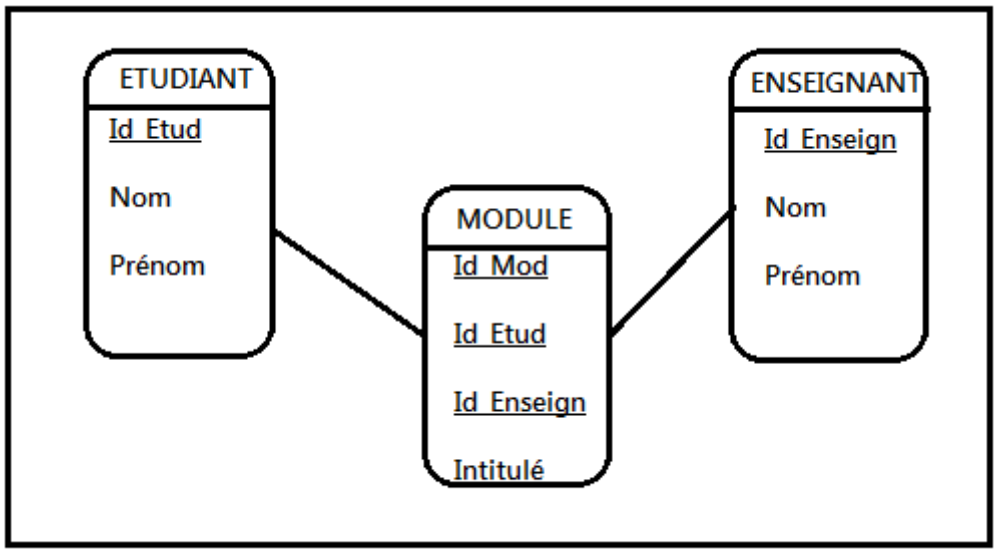

FIGURE 3.5 – Base de données test.

nous avons divisé cette base en deux partie, la première dans la machine 1 et la deuxième dans la machine 2, afin de pouvoir exploiter la notion de calcul paralèlle. Nous avons utilisé nos propres algorithmes pour réaliser les requêtes qui vont venir aprés. La figure 3.5 montre un schéma de notre base.

3.4.2 Implémentation de la selection

La requête de selection consiste a extraire un ensemble de données selon un critère donné, nous avons implémenté un exemple de cette requête :

```
SELECT *
FROM Table
```

et :

```
SELECT *
FROM Table
WHERE Id=?
```

Les figures 3.6 et 3.7 montre le resultat d'éxecution de ces deux requêtes.

3.4.3 L'implémentation de la jointure

nous avons implémenté une requête de jointure entre la table Etudiant et la table Enseignant, cette requête consiste a donner l'ensemble des etudiants enseignés par un enseignant donné.

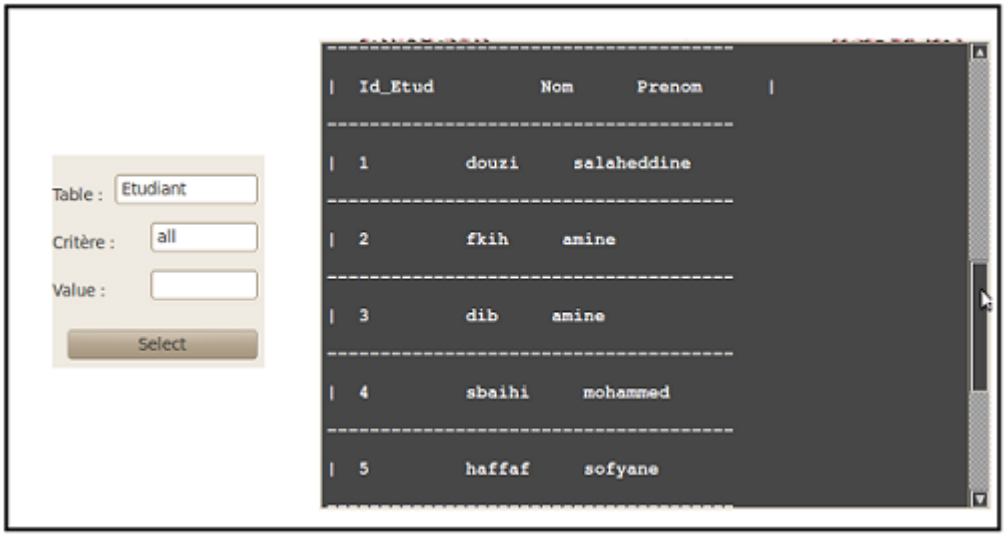

FIGURE 3.6 – Selection de tout les enregistrements.

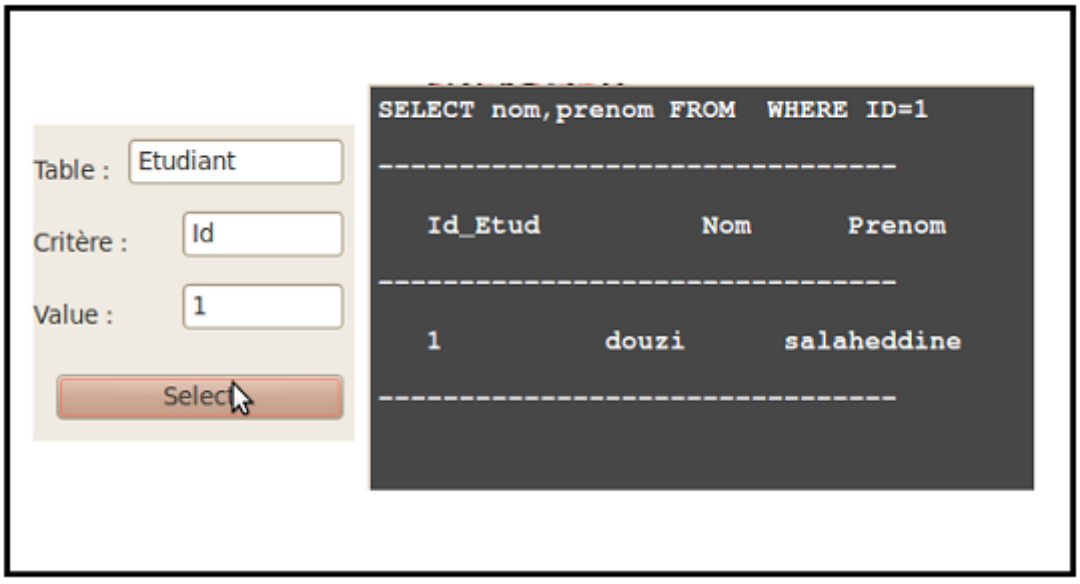

FIGURE 3.7 – Selection d'un seul enregistrement.

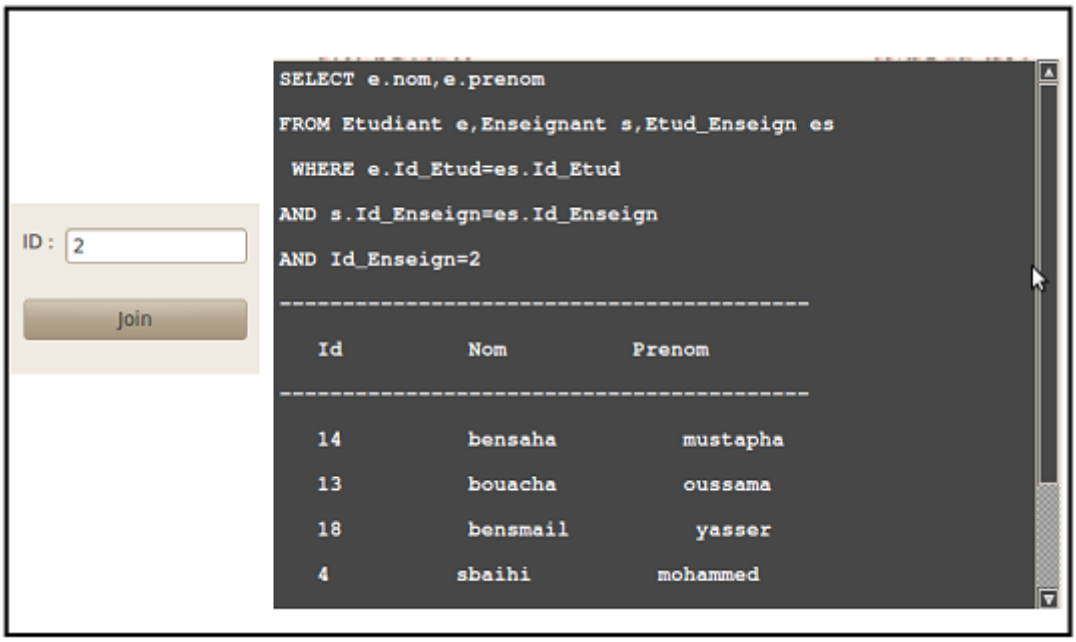

FIGURE 3.8 – jointure entre deux table.

```
   SELECT *
FROM Etudiant e, Enseignant s, Module m
WHERE e.Id_Etud=m.Id_Etud
AND s.Id_Enseign=m.Id_Enseign
AND s.Id_Enseign=?
```

La figure 3.8 montre le resultat d'éxecution de cette requête.

3.4.4 Implémentation de mise à jour

Insertion Comme nous avons dit avant, que la base de données est répartie donc la requête d'insertion nécéssite un prétraitement pour pouvoir insérer un enregistrement. Nous avons implémenté un algorithme qui vérifie le nombre d'enregistrement dans la table de destination sur l'architecture distribué afin de laisser notre base de données équilibrée.

La requête de l'insertion est :

```
INSERT INTO table(champ1,champ2,champ3)
VALUES('value1','value2','value3')
```

La figure 3.10 montre le resultat d'éxecution de cette requête.

Suppression Cette requête a pour but de supprimer un enregistrement souhaité.

La requête est :

```
DELETE FROM table WHERE Id=?
```

La figure 3.11 montre le resultat d'éxecution de cette requête.

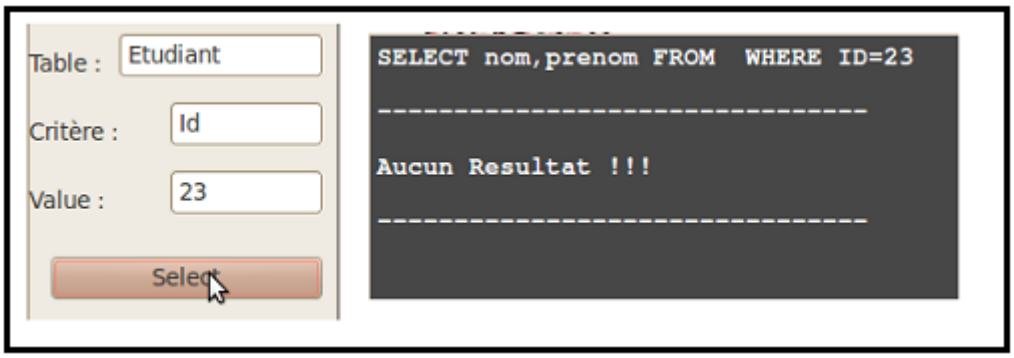

FIGURE 3.9 – Avant insertion.

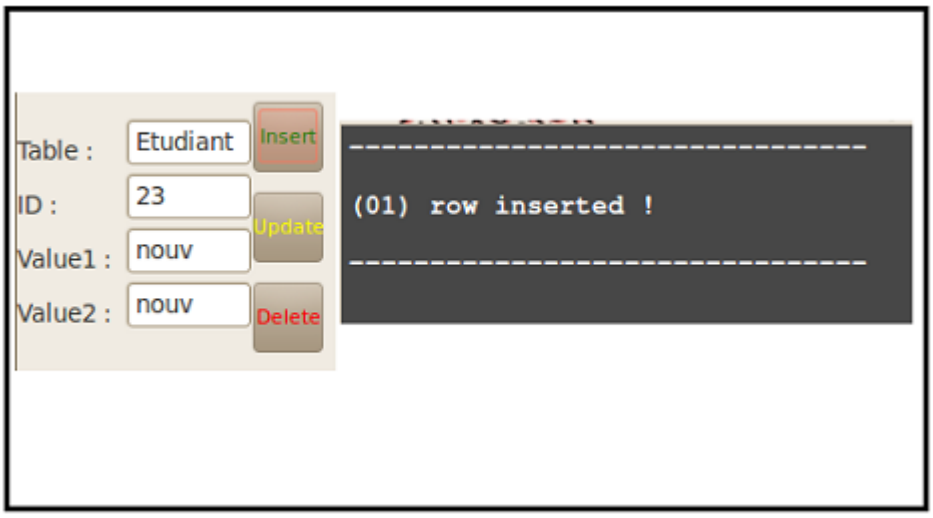

FIGURE 3.10 – Aprés insertion.

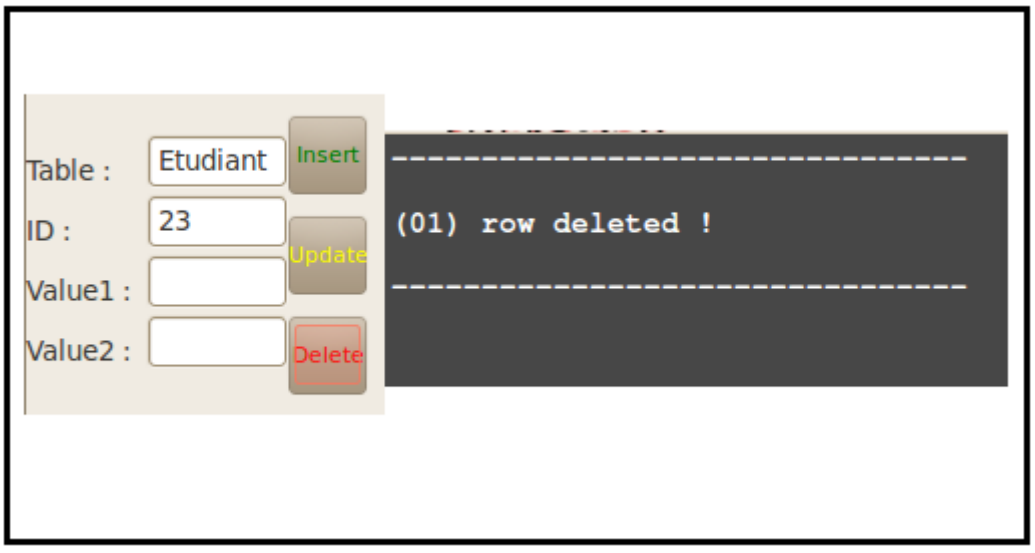

FIGURE 3.11 – Suppression.

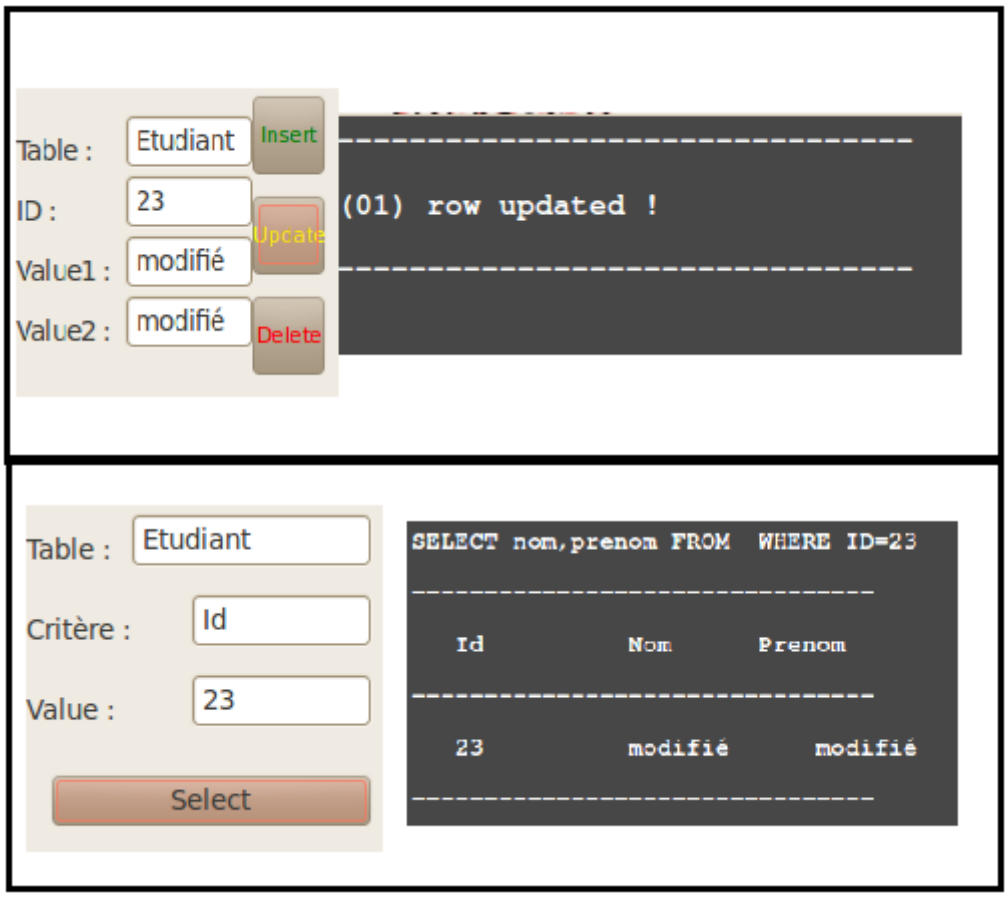

FIGURE 3.12 – Modification.

Modification Cette requête a pour but de modifier un enregistrement souhaité. La requête est :

```
UPDATE table SET champ1=? champ2=? champ3=?
```

La figure 3.12 montre le resultat d'éxecution de cette requête.

3.4.5 Exemples de BDMT et BCTD

BDMT Un premier exemple qui implémente une requête sous l'architecture BDMT qui calcul la moyenne des notes enregistrées dans deux fichiers, le premier processus calcul la moyenne et envoi la valeur au deuxième processus,ce dernier a déja calculé la moyenne en paralèlle avec le premier, et en plus il calcul la moyenne générale. La figure 3.13 montre le traitement de calcul de la moyenne.

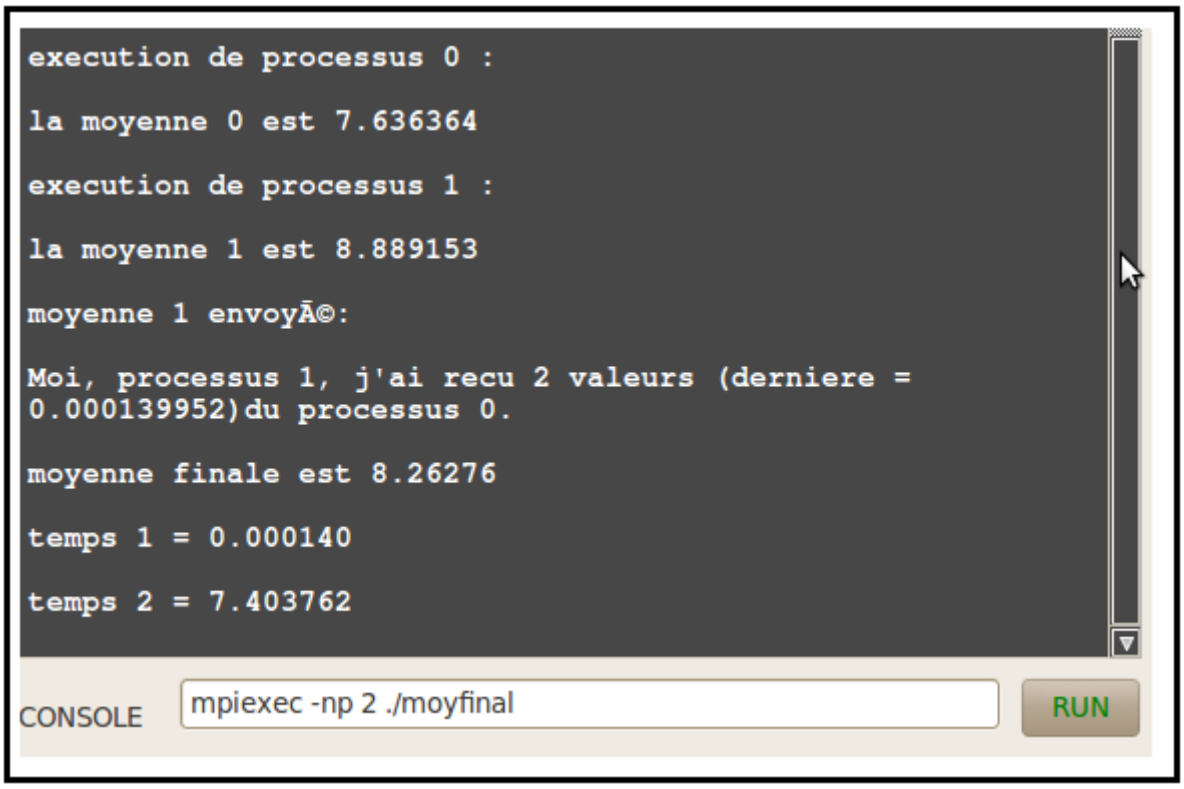

FIGURE 3.13 – calcul de la moyenne en paralèlle.

BCTD Un deuxième exemple qui implémente une requête sous l'architecture BCTD qui calcul le max et min d'un ensemble de notes enregistrées dans un fichier dupliqué. Le premier processus fait le traitement de max et deuxième fait le traitement de min.
La figure 3.14 montre le traitement de max et min.

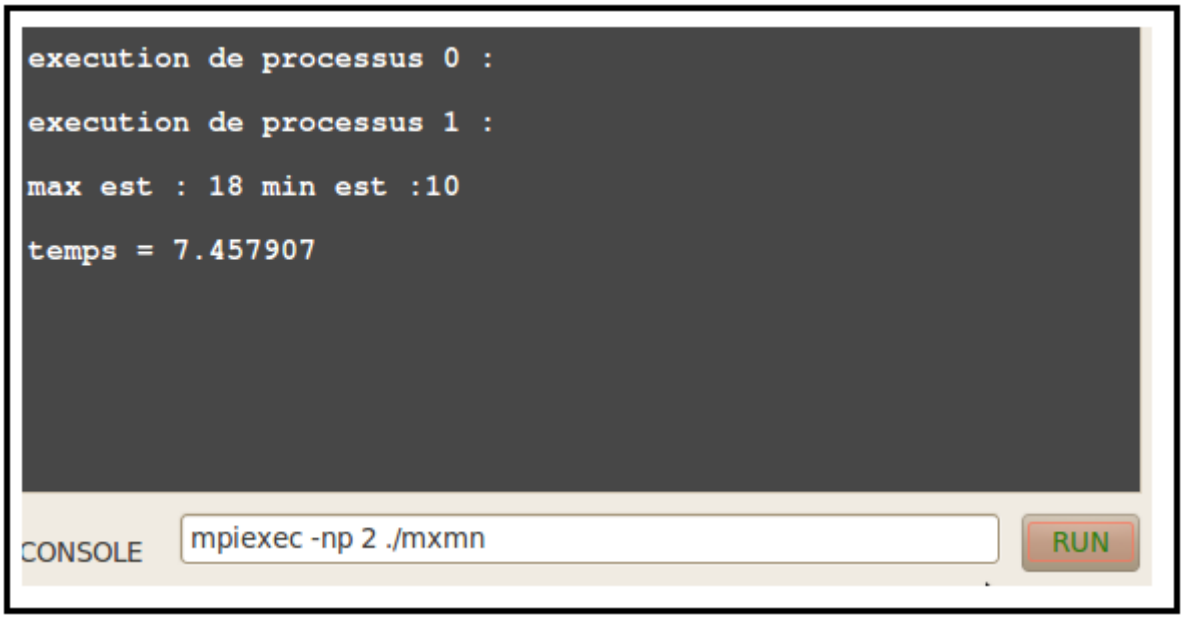

FIGURE 3.14 – calcul de max et min.

3.5 Conclusion

Dans ce chapitre, nous avons détaillé l'environnement MPI qui permet de faire du calcul parallèle en se basant sur le principe d'échanges de messages (communications)entre processus, chacun ayant sa propre mémoire. Ces opérations de communications peuvent être collectives ou point à point. Dans un code de calcul parallèle, si le temps de communication devient prépondérant devant celui de calcul, on peut optimiser le temps de communication grâce aux différents modes de transfert de messages proposés par MPI ainsi qu'en recouvrant les communications par des calculs. Nous avons simulé trois requêttes avec MPI, le résultat obtenu montre clairement l'intérêt de MPI pour ce type de requêttes notamment pour la requête jointure qui était la plus satisfaisante.

Conclusion générale

L'objectif de ce travail est de montrer l'intérêt de l'environnement distribué parallèle dans l'interrogation de bases de données de trés grandes tailles, pour cette raison nous avons implémenté trois type de requêtes : selection, jointure et mise à jour en utilisant l'invironnement MPI pour lancer des exécutions parallèles.

Concernant la portabilité de notre système, elle a été traité dans des conditions bien précises, pour ce qui concerne la généralisation, elle sera laissée comme perspectives en utilisant un super calculateur de trés grand nombre de coeurs.

Suite au travail réalisé dans le cadre de ce PFE, nous espérons que différentes études pourraient dans l'avenir compléter les résultats obtenus. La réalisation de ce travail au sein de l'université Abou Bakr Belkaid nous a permis de connaître de prés le fonctionnement des systèmes distibués et l'implémentation des différente type de requêtes dans ce type d'environnement.

En effet, tout au long de cette période, nous étions en face à de nombreux problèmes à savoir des difficultés majeures étant la compréhension du fonctionnement de MPI et l'établissement d'un calculateur(cluster de 6 coeurs), ce cluster nous a permis de réaliser l'éxecution des différentes requêtes.

Ce travail nous a permis aussi d'améliorer nos compétances dans différents domaines comme : La programmation réseau sous Linux (SSH, Shell, ...), C, C++. Nous avons également appris plusieurs techniques comme : l'utilisation de Latex, l'utilisation de QT creator, ... ce qui nous a permis en final de concrétiser nos connaissances en système d'information et de connaissances que nous avons acquis durant nos études académiques au sein de notre université.

Les compétances acquises tout au long de ce PFE et l'ensemble de techniques apprises durant la réalisation de ce projet nous ont permis de consolider nos connaissances dans le domaine de système d'information et de connaissances.

Bibliographie

[1] R.Grin. Introduction aux bases de données. *Université de Nice Sophia-Antipolis Version 2.1*, decembre 2000.

[2] P.Rigaux. Cours de bases de données. *cours*, page 9, juin 2001.

[3] JP.Antoni et Y.Flety. Introduction aux bases de données. *cours*.

[4] C.Marée et G.Ledant. Sql2. *Edition Armand Colin ISBN : 2-200-21411-1*.

[5] G.jamal eddine. Sécurité dans les grilles de calcul. *phD thesis*, 2004.

[6] Y.Zakaria et B.Fayssal. Les algorithmes itératifs asynhrones dans les systèmes distribués volatiles. *Thèse de master, Université des Sciences et Technologies de tlemcen*, juillet 2011.

Glossaire

LMD : Langage de manipulation de données.

SGBD : Système de gestion de base de données.

SQL : Structured query language.

SGBDR : Système de gestion de base de données relationnelles.

LDD : Langage de définition de données.

FTP : File transfer protocol.

MPI : Message passing interface.

IP : Internet protocol.

BDMT : Base de données distribué même traitement.

BCTD : Base de données centralisée traitement différent.

SHH : Secure shell.