# Les algorithmes itératifs asynchrones dans les systèmes distribués volatiles

YAHOUNI zakaria et BENDAOUD fayssal

2011-2012

**Résumé**

On s'est intéressé dans le cadre de ce PFE à étudier quelques méthodes de résolution des systèmes linéaires à grande échelle afin de montrer l'intérêt des algorithmes itératifs asynchrones (IACA) pour supporter le parallélisme dans un contexte distribué. L'outil JACE nous a permis d'atteindre nos objectifs, à savoir la simulation des trois méthodes analysées : Jacobi, Gradient optimal et Gradient conjugué. Nous avons proposé également une approche de compression basée sur CSR et Ellpack-Itpack qui a montrée son intérêt dans le cas des matrices creuses.

This thesis focuses on few methods for solving linear systems on a large scale to show the benefit of asynchronous iterative algorithms (IACA) and to support the parallelism in a distributed context. The API JACE allowed us to achieve our goals, execute the simulation of the three analyzed methods : Jacobi, Optimal Gradient and conjugate gradient. We also proposed an approach based on compression methods (CSR and Ellpack Itpack) which has shown its interest in the case of sparse matrices.

# Remerciements

En premier lieu nous remercions Dieu, le tout puissant pour ses faveurs et ses gratitudes, de nous avoir donnée le courage et la patience pour avoir mené ce travail durant toute cette année.

De plus, nos remerciements s'adressent à Monsieur **Badr Benmammar**, Enseignant à l'Université Abou Bakr Belkaid-Tlemcen, pour nous avoir fait l'honneur de nous encadrer et guider durant cette année.

En effet, nous lui sommes infiniment reconnaissants de nous avoir encouragés et soutenu durant ce travail nous avons ainsi pu apprécier sa rigueur scientifique, son recul, ses grandes qualités humaines et son œil critique qui nous a été très précieux pour structurer notre travail et améliorer sa qualité. Qu'il soit persuadé de notre plus profonde considération et plus grand respect.

Egalement, nous tenons à remercie Monsieur **Kamel Mazouzi** Ingénieur de recherche dans l'université de Franche-Comté pour son aide et son soutien ainsi Monsieur **Mourad Hakem** Enseignant à l'Université de Franche-Comté qui nous a apporté son aide, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Enfin, un grand remerciement destiné à l'ensemble de nos enseignants et enseignantes qui ont contribué à notre 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 ton amour, ton affection et ton soutien, pour ton courage et ton sacrifie, je te dédié, pour la deuxième et mille fois, ma très chère mère, un résultat modeste de la bienveillance et tes longes années de patience.*

*A mes chers sœurs, à mes chers amis Et à tous ceux qui j'aime et qu'ils m'aiment... où qu'ils soient.*

*ZAKARIA*

# Dédicace

*Je dédie ce modeste travaille :*

*A celui qui m'a indique la bonne voie en me rappelant que la volonté fait toujours les grands hommes....*

*A celle qui a attendu avec patience les fruits de sa bonne éducation ....*

*Tous simplement A mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.*

*A mes chers sœurs, à mes amis et à toute ma famille...*

*Et à tous ceux qui j'aime et qu'ils m'aiment... où qu'ils soient.*

*FAYSSAL*

# Table des matières

# Table des figures



# Liste des tableaux

# Introduction générale

Dans le cadre du calcul numérique scientifique, la résolution des systèmes linéaires de très grande taille a toujours jouée un rôle majeur, notamment dans le domaine de biologie pour le décodage des portions d'ADN ainsi que dans le domaine de météorologie pour la prévision des variations climatiques à venir, ce type d'application conduit souvent à des systèmes linéaires creux qui peuvent atteindre des tailles massives, voir des centaines de millions d'inconnues.

Ce type d'application est tellement gourmand en mémoire alors qu'ils mènent à des coûts prohibitifs en temps CPU et en espace mémoire qu'il est indispensable de les exécuter dans un environnement parallèle d'une façon coopérative dans le but d'accroitre la puissance disponible pour réaliser ce 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 ce contexte d'exécution, différents problèmes sont apparus 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 induites par ces contextes d'exécutions.

L'objectif de ce travail consiste en premier temps de montrer l'intérêt des algorithmes itératifs asynchrone dans ce type d'environnement et après d'envisager les méthodes de résolution de ces systèmes linéaires afin de sortir avec les bonnes approches qui mènent à des convergences plus rapide.

Le premier chapitre de notre rapport aborde une brève présentation des types et d'architectures des réseaux existants et une visualisation des différents problèmes affecteront les résultats appliqués dans ce type de réseaux, le deuxième chapitre met le point sur l'importance des grilles de calculs et l'indispensabilité des méthodes itératives asynchrone pour le partage des tâches et le gain de temps de traitement pour la résolution des systèmes linéaires dans un environnement distribué. Le dernier cha-

pitre présente les trois approches de résolution d'un système linéaire à savoir Jacobi, Gradient optimal et Gradient conjugué, nous présentons également l'environnement JACE qui permet d'exécuter ce type d'applications parallèles. Enfin, nous évaluons et analysons les performances de ces méthodes de résolution des systèmes linéaires avec l'environnement JACE, cependant, l'approche de Jacobi a bien montré son efficacité par rapport aux deux autres méthodes (convergence plus rapide), c'est pour cette raison que nous avons opté pour cette méthode afin de traiter le cas des matrices creuses, pour arriver à une amélioration du temps de calculs suivants les deux méthodes de compression des matrices creuses, CSR pour représenter les données de traitement et Ellpack-Itpack pour répartir les données de la matrice entre les différentes machines du calculateur.

# Chapitre 1

# Etat de l'art

## 1 Introduction

Aujourd'hui, la majorité des ordinateurs et la quasi-totalité des appareils « mobiles »(tels que les téléphones portables, agendas électroniques,...) disposent de moyens de connexion à un ou plusieurs types de réseaux sans fil, ainsi, il est très facile de créer en quelques minutes un réseau « sans fil »permettant à tous ces appareils de communiquer entre eux grâce au développement rapide des technologies sans fil ces dernières années, notamment la naissance de plusieurs types de réseau sans fil.

Dans ce chapitre nous allons détaillés les types des réseaux sans fil, leurs principe de fonctionnement et les protocoles de routage utilisés.

## 2 Réseaux sans fil

Un réseau sans fil est un ensemble d'appareils connectés entre eux et qui peuvent s'envoyer et recevoir des données sans qu'aucune connexion « filaire »physique reliant ces différents composants entre eux ne soit nécessaire.
Les réseaux sans fil sont utilisé pour :

- Faciliter la connexion des utilisateurs itinérants, en particulier dans les espaces collectifs.
- Pour connecter des locaux impossibles ou trop coûteux à câbler (amiante, monument historique).
- Pour mettre en place une connexion provisoire.

On distingue principalement 2 types de Réseaux sans fil :

- Les réseaux sans fil utilisant les ondes infrarouges
- Les réseaux sans fil utilisant les ondes radios (Bluetooth, Wifi, réseaux cellulaires,

## 2.1 Réseaux utilisant les ondes infrarouges :

Les ondes infrarouges sont couramment utilisées dans la vie courante (pour les télécommandes de télévisions par exemple). Grâce à elles, on peut créer des petits réseaux, notamment entre des téléphones portables et des ordinateurs.

Le principal inconvénient des réseaux créés avec les ondes infrarouges est qu'ils nécessitent que les appareils soient en face l'un de l'autre, séparés au maximum de quelques dizaines de mètres et qu'aucun obstacle ne sépare l'émetteur du récepteur puisque la liaison entre les appareils est directionnelle. Cependant, contrairement aux technologies utilisant les ondes radio, les ondes infrarouges sont peu perturbées par l'environnement extérieur (microondes, émetteurs radio, ...).

Les ondes infrarouges sont utilisées pour :
- La majorité des appareils avec une télécommande sans fil : télévision.
- Les télécommandes du verrouillage automatique des anciennes voitures.
- Les télécommandes des jouets : les petits hélicoptères télécommandés.

Bien entendu, la majorité des réseaux utilisables par cette technologie sont les WPAN.

## 2.2 Réseaux utilisant les ondes radios :

Quant aux ondes radios, elles sont utilisées par un grand nombre de réseaux sans fil. A la différence des réseaux utilisant les ondes infrarouges, il faut prendre garde aux perturbations extérieures qui peuvent affecter la qualité des communications dans le réseau, à cause, par exemple, de l'utilisation de mêmes fréquences par d'autres réseaux ou la présence de certains matériaux qui altère la qualité des transferts. Cependant, les ondes radios ont l'avantage de ne pas être arrêtées par les obstacles et sont en général émises de manière omnidirectionnelle.

Les ondes radios (RF pour Radio Frequency) se propagent toujours en ligne droite et sont émises de manière directionnelle (par les satellites par exemple) ou omnidirectionnelle (par les antennes Wifi par exemple).

En pratique, le signal peut être perturbé si une onde radio rencontre un obstacle (un mur par exemple) et la puissance du signal atténuée. L'atténuation augmente avec l'augmentation de la fréquence du signal et/ou de la distance. De plus, lors d'une collision, la valeur de l'atténuation dépend fortement du matériau composant l'obstacle. Par exemple, les obstacles métalliques provoquent généralement une forte réflexion, tandis que l'eau absorbe le signal.

Par définition une onde radio est susceptible (si elle est émise de manière omnidirectionnelle) de se propager dans plusieurs directions. Un signal source peut être amené à atteindre une station ou un point d'accès en empruntant des chemins multiples (on parle de multi-path ou, cheminements multiples).

Le délai de propagation entre deux signaux ayant emprunté des chemins différents peut provoquer des interférences au niveau du récepteur car les données reçues se chevauchent. Ces interférences deviennent de plus en plus importantes lorsque la vitesse de transmission augmente car les intervalles de temps entre la réception des données sont de plus en plus courts. Les chemins de propagation multiples limitent ainsi la vitesse de transmission dans les réseaux sans fil.

La plut part des technologies sans fil utilisent les ondes radios comme support de communication (Bluetooth, Wifi, réseaux cellulaires, Wimax, GSM, ...).

### 2.2.1 Types des réseaux sans fil

**WPAN (IEEE 802.15) :** Le réseau personnel sans fil (WPAN - Wireless Personal Area Network) est constitué de connexions entre des appareils distants de seulement quelques mètres (PC, assistants, périphériques divers...) comme dans un bureau ou une maison [3].

**WLAN (IEEE 802.11) :** Le réseau local sans fil (WLAN - Wireless Local Area Network) correspond au périmètre d'un réseau local installé dans une entreprise, dans un foyer ou encore dans un espace public [4].
Le groupe 802.11 a été initié en 1990 et la norme IEEE 802.11 définissant les réseaux locaux sans fil a vu le jour en 1997. La norme d'origine a défini trois couches physiques pour une même couche MAC, correspondant à trois types de produits 802.11 :

- IEEE 802.11 FHSS (Frequency Hopping Spread Spectrum), qui utilise la technique d'étalement de spectre basé sur le saut de fréquence.
- IEEE 802.11 DSSS (Direct Sequence Spread Spectrum), qui utilise aussi la technique d'étalement de spectre mais sur une séquence directe.
- IEEE 802.11 IR (InfraRed), de type infrarouge.

Les réseaux IEEE 802.11 FHSS et IEEE 802.11 DSSS sont des réseaux radio sans fil émettant dans la bande ISM. La norme IEEE 802.11 n'est pas restée figé et de nombreuses améliorations ont été apportées à la norme d'origine. Ces

améliorations continuent actuellement. Trois nouvelles couches physiques ont été ajoutées avec les normes IEEE 802.11b, IEEE 802.11a et IEEE 802.11g.
La couche physique est divisée en deux sous-couches :

- La sous-couche PMD (Physical Medium Dependent) qui gère l'encodage des données et effectue la modulation.
- La sous-couche PLCP (Physical Layer Convergence Protocol) qui s'occupe de l'écoute du médium et fournit un CCA (Clear Channel Assessment) à la couche MAC pour lui signaler que le canal est libre.

De point de vue de la standardisation, trois standards pour les réseaux locaux sans fil ont été établis à ce jour : HiperLan (Hight Perfermance Radio LAN),HomeRF (Home Radio Frequency) et Wi-Fi (IEEE 802.11b).

**HiperLan :** est un standard initié par l'ETSI (European Telecomunication Standards Institute) qui fonctionne dans la bande des 5GHz avec vitesse de transmission de 54 Mbps.HiperLan n'a jamais été commercialisé et n'est resté qu'à l'état de prototype.

**HomeRF :** est un standard initié par le HomeRF Working Group regroupant les grands noms de l'industrie informatique. Ce standards fonctionne dans la bande des 2.4GHz appelé bande ISM (Industrie Science et Médecine) et propose des vitesses jusqu'à 10Mbps, HomeRf a connu un certain succès commercial outre-Atlantique mais n'a pas réellement percé sur les autres marchés notamment européen et asiatique. C'est l'une des raisons qui poussé le consortium à stopper la standardisation d'HomeRf en janvier 2003.

Enfin IEEE802.11, standardisé en 1997, est le seul standard qui a réussi à s'imposer à ce jour comme standard de référence pour les réseaux locaux sans fil [5].

**WMAN :** Définit des réseaux métropolitains sans fil. Il est homologué comme standard IEEE 802 depuis octobre 2002.

Elle comporte la famille Wimax IEEE802.16 utilisant des fréquences au de la de 10GHZ et le standard MBWA (Mobile BroadBand Wireless Access) IEEE802.20 utilisant des fréquences moins de 3.5GHZ.

**Le standard Wimax IEEE802.16 :** Le Wifi se développe énormément et devient de plus en plus accessible financièrement. Mais le Wifi est limité par sa porté et ne peut donc être utilisé que pour des réseaux locaux.

Le Wimax est donc là pour combler cette "défaillance" et permettre aux utilisateurs de profiter d'un internet très haut débit sans être raccordé a la boucle locale .Wimax est une technologie complémentaire du Wifi, faisant partie des WMAN .

Le Wimax, qui signifie « **World Interoperability for Microwave Access** » est une norme technique basée sur le standard de transmission radio 802.16, validé en 2011 par l'organisme international de normalisation IEEE. Elle se situe dans la gamme de fréquence entre 10GHZ et 66 GHZ permettant un débit théorique jusqu'à 70Mb/s sur un rayon de 50km. Elle est activement développée par Intel, Fujitsu, Nokia et bien d'autres.

Pour les mécanismes de sécurité, Wifi utilise les clés WEP et WPA (64 ou 128 bits) alors que Wimax utilise Triple-DES (128 bits) et RSA (1024 bits). Il ya donc une sécurité accrue sur le Wimax

De plus les fréquences utilisées par Wimax et Wifi ne sont pas les mêmes. On peut donc dire que Wifi et Wimax sont complémentaires et peuvent coexister sans problème [6].

La norme 802.16e est très attendue car elle rajoute ce qu'il manque au Wimax par rapport au Wifi : **la mobilité**, ou autrement dit « **le roaming** », terme qui définit le fait de ne pas perdre le flux du réseau si l'on est mobile. C'est-à-dire que l'on pourra profiter d'un accès internet sans fil même en voyageant à plus de 100km/h à condition de rester dans la zone de couverture.

**Le standard MBWA IEEE802.20 :** **MBWA** est un standard en cours de développement IEEE802.20. Il devrait permettre la mise en place de réseaux métropolitains mobiles avec des vitesses allant jusqu'à 250 km/h. Le MBWA utilise des bandes de fréquences avec licence au dessous de 3.5GHZ.

Il permet des débits maximums par utilisateurs de 1Mb/s en descente et 300 Kb/s en montée (contrairement aux autres technologies où l'ensemble de la bande passante est partagé) avec des cellules d'un rayon de 2,5 km maximum.

Des versions utilisant un canal plus large de 5 MHz pourraient permettre des débits de 4 Mb/s en descente et 1,2 Mb/s en montée pour chaque utilisateur.

Le MBWA est bien adapté à la mobilité voix et données avec des terminaux centrés sur les données (par rapport aux réseaux mobiles de 3e génération qui sont adaptés à la mobilité voix et données avec des terminaux aujourd'hui centrés sur la voix).

L'ambition du projet IEEE 802.20 est de combler le fossé entre les réseaux sans fil haut débits à faible mobilité et les réseaux mobiles ayant un débit plus restreint [7].

**WWAN :** Wireless Wide Area Network, Technologie utilisée par les téléphones portable actuellement, il existe plusieurs variantes :

**GSM (Global System for Mobile Communication) :** La norme GSM est maintenant adoptée dans nombreux pays. C'est l'unique norme numérique de téléphonie cellulaire 2G acceptée en Europe. Dans la plupart des autres régions du monde, elle est en concurrence avec d'autres normes de radiotéléphonie numérique, en général originaire des USA (Interim Standard IS-95) ou du Japon **(Personal Digital Cellular PDC)**.

En février 2004, le nombre d'abonnés est estimé à un milliard, réparti dans plus de 200 pays du monde. Néanmoins, les abonnés se situent principalement en Europe (environ 400 millions d'abonnés) et en Asie (environ 400 million d'abonnés). Le marché européen étant presque arrivé en saturation (plus de 70% de taux de pénétration), il y aura très prochainement nettement plus d'abonnés en Asie qu'on Europe. A titre de comparaison, il y a environ 65 millions d'abonnés PDC et moins d'un million d'abonnés IS-95.desservir.

Le réseau GSM est séparé en 3 ensembles distincts :

- Sous système radio **BSS** (Base Station Subsystem) : constitué des stations de base **BTS** (Base Transceiver Station)
- Le sous système réseau **NSS** (Network sub-System) regroupe les fonctions de commutation et de routage

- Le sous system d'exploitation et maintenance **OSS** (Operation and maintenance Sub System) utilisé par l'operateur pour administrer son réseau.

**GPRS (General Packet Radio Service) :** L'idée du GPRS est venue de l'utilisation de plus en plus courante de l'internet : lors de la consultation d'une page web, une session peut durer plusieurs dizaines de minutes alors que les données ne sont réellement transmises que pendant quelques secondes, lors du téléchargement des pages. Le trafic de données engendré est dons très sporadique, contrairement à un trafic de voix par exemple. Deux problèmes majeurs se posaient alors dans un réseau à communication de circuit comme GSM :

- Le monopole des ressources, dans tout le réseau, pour un seul utilisateur, pendant toute la durée de sa session, alors que ces ressources n'étaient réellement utilisées qu'épisodiquement.
- La taxation, basée sur le temps de connexion de l'utilisateur, aurait eu une tendance à s'alourdir très sensiblement, du fait du téléchargement des données à relativement faible débit

Le GPRS résout ces deux problèmes en définissant une architecture de réseau à communication de paquets qui permet de n'allouer des ressources à un utilisateur qu'au « coup par coup », lorsqu'il a réellement des données à émettre ou à recevoir et non durant toute la durée de sa connexion, ainsi que le taxer sur le volume (en kbits) des données échangées et non sur la durée totale de connexion.

Le réseau GPRS et le réseau GSM fonctionnent en parallèle : le premier est utilisé pour le transport des données, le second pour les services classiques de voix. Les deux réseaux utilisent les mêmes équipements BSS du GSM, mais ils se séparent ensuite, le GPRS pouvant être relié à divers réseaux de données fixes reposant sur plusieurs protocoles (IP, X25...) ou à un autre réseau GPRS (exploité par un autre opérateur) , tandis que le GSM sera relié à un réseau téléphonique commuté public (réseau fixe national ou international ) ou un autre réseau GSM exploité par un autre opérateur [5].

**UMTS (Universal Mobile Telecommunication System) :** IMT-2000 (International Mobile telecommunication) le terme utilisé par l'ITU pour la nor-

malisation, appelé aussi 3GSM, il utilise des nouvelles bandes de fréquences 1920-1980 Mhz et 2110-2170Mhz.

Basé sur la technologie W-CDMA, UMTS permet d'atteindre un débit jusqu'à 2Mb/s pour les abonnés immobiles, un débit de 384Kb/s pour les piétons et mobiles en zones urbaines et un débit de 144 Kb/s pour les usagers se déplaçant à vitesse élevée ($\sim 500km/h$).

Les vitesses de transmissions offertes par les réseaux UMTS sont nettement plus élevées, que celles des réseaux GSM. Techniquement UMTS utilise l'envoi par paquet contrairement a GSM l'envoi par circuit, l'architecture du réseau est composé de trois parties :

- **Domaines utilisateur**, il se compose d'un terminal capable de gérer l'interface radio et d'une carte à puce.
- **Domaines d'accès radio**, UTRAN (Universal Terrestrial Radio Access Network), il est constitué d'un ou plusieurs RNC (Radio Network Controllers), dont dépendent des Nodes B (BTS en GSM). Les RNC jouent un rôle proche de celui des BSC du GSM.
- **Domaine du réseau cœur, Core Network** est hérité de l'architecture NSS du GSM est constitué d'une partie commutation de circuits (MSC-GSM) et d'une partie commutation de paquets (SGSN-GPRS). Les nœuds de signalisation, la gestion de la mobilité, les services réseaux intelligents et les HLR, VLR, EIR, AUC..., devront évoluer pour intégrer les spécificités UMTS [5].

**WRAN IEEE802.22 :** La prolifération des services et des dispositifs sans fil pour des utilisations telles que les communications mobiles, la sécurité publique, le Wi-Fi et la télévision servent d'exemple le plus incontestable de combien la société moderne est devenue dépendante de spectre radioélectrique qui est devenue la ressource la plus précieuse de l'ère moderne, par l'utilisation des bandes ISM et UNII.

Le terme de **radio cognitive** a été fréquemment utilisé pour parler d'un système capable de prendre conscience de son environnement et de tirer profit de cette information. Parfois, il est considéré de façon plus restrictive comme un système disposant d'une grande agilité en fréquence pour explorer les opportunités qui peuvent exister dans le spectre fréquentiel.

Les réseaux **radios cognitifs** (Cognitive Radio Networks ou CRN) émergent comme un nouveau concept d'accès et de partage de canal dans les réseaux sans fil. Le but est d'exploiter les bandes passantes résiduelles sur les fréquences sous utilisées du spectre fréquentiel. En effet, plusieurs études récentes ont révélé l'utilisation sous-optimale des bandes radios en insistant sur le fait que les fréquences gratuites (comme celle utilisée par le WiFi par exemple) deviennent surchargées tandis que d'autres bandes restent largement sous-utilisées. Les CRNs forment une nouvelle famille d'utilisateurs radios qui tentent d'exploiter les bandes de fréquences sous-utilisées durant l'absence d'utilisateurs prioritaires et libèrent le canal d'es lors que les utilisateurs licenciés de ces fréquences tentent d'y accéder.

Plusieurs études se sont penchées sur les mécanismes des couches PHY et MAC nécessaires pour assurer la cohabitation entre les réseaux cognitifs et les réseaux déjà existants appelés réseaux primaires. Deux contraintes principales sont à respecter par les nœuds cognitifs [8] :

- La transmission des nœuds cognitifs ne doit pas perturber la transmission des nœuds primaires s'effectuant sur le même canal, ce qui nécessite un contrôle strict de la puissance de transmission des nœuds cognitifs.
- La transmission d'un nœud cognitif doit être immédiatement interrompue lorsqu'un nœud primaire (plus prioritaire sur le canal en question) commence une transmission concurrente. Cette contrainte nécessite une forte coordination entre les deux réseaux. Les études ont conduit à plusieurs propositions au niveau des couches PHY et MAC pour la détection des canaux disponibles et l'estimation de l'interférence et de la puissance de transmission d'un nœud cognitif.

### 2.2.2 Architecture des réseaux sans fil

**Mode infrastructure (centralisé) :** A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes (les stations de base) sont interconnectés entre eux à travers un réseau de communication filaire (Distributed System DS).

**Mode Ad hoc (distribué) :** L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calculs portables (les laptops par exemple), poussent aujourd'hui les chercheurs à faire des efforts afin

de réaliser le but des réseaux : « L'accès à l'information n'importe où et n'importe quand. »

Les réseaux ad hoc sont idéals pour les applications caractérisées par une absence (ou la non-fiabilité) d'une infrastructure préexistante *(Figure (1,1))*, tel que :

- Les applications militaires.
- Les autres applications de tactique comme les opérations de secours (incendies, tremblement de terre...) et les missions d'exploration.

**Définition** : Un réseau mobile ad hoc [9], appelé généralement **MANET** (Mobile Ad hoc NETwork ), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée.

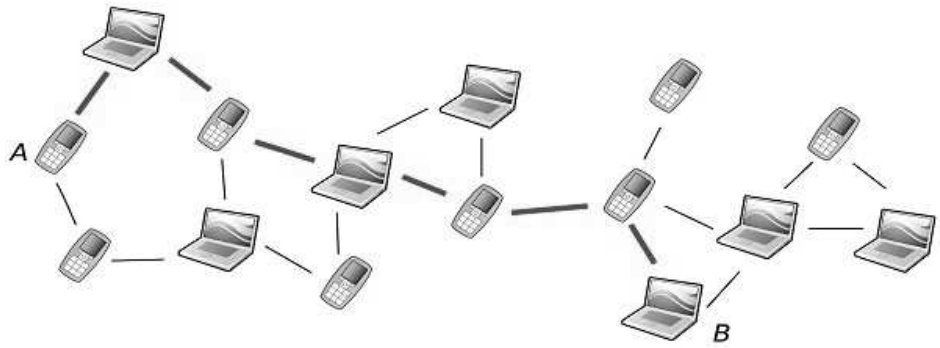


FIGURE 1.1 – Exemple réseau mobile sans fil (**MANET**)

Un réseau **MANET** [9] est un réseau sans fil capable de s'organiser sans infrastructure définie préalablement. Un tel réseau est composé de stations mobiles ou nœuds qui peuvent communiquer directement entre eux s'ils sont situés à portée radio. La portée des stations étant relativement limitée, le déploiement d'un réseau à grande échelle nécessite que le réseau MANET soit multi-saut, c'est-à-dire que des stations intermédiaires fassent office de point de relais. Les réseaux MANETs, grâce à leur auto-organisation et à l'absence d'infrastructure, peuvent facilement être déployés dans de nombreux domaines comme l'embarqué (intégré récemment dans le secteur automobile pour accroître la sécurité des usagers en les informant d'éventuels obstacles sur leur itinéraire), lors d'opérations de secours (sauvetage en mer, en zones sinistrées...) ou lors d'opérations militaires. Les réseaux MANETs se caractérisent également par leurs faibles ressources sur

la totalité de la ligne de communication. Cela se traduit par une autonomie limitée, car les stations sont généralement alimentées à l'aide de batteries et par une puissance relativement faible du fait de la compacité des équipements emportés. De plus, la capacité des liens sans fil s'avère relativement limitée offrant par conséquent un débit modeste comparé aux réseaux filaires.

Un réseau **MANET** n'est pas lié à une technologie de communication sans fil particulière. De nombreuses technologies sans fil permettent le déploiement d'un réseau MANET : les réseaux sans fil personnels (WPAN) avec les réseaux de type Bluetooth et Zibgee, les réseaux sans fil locaux (WLAN) avec IEEE 802.11 (ou WiFi) et HyperLan de de ETSI.

Les réseaux mobiles ad hoc sont caractérisés par ce qui suit :
- Une topologie dynamique.
- Une bande passante limitée.
- Des contraintes d'énergies.
- Une sécurité physique limitée.
- L'absence d'infrastructure.

**Modélisation des réseaux Ad hoc :** Un réseau ad hoc peut être modélisé par un graphe G = (V, E). Où : V représente l'ensemble des nœuds (i.e. les unités ou les hôtes mobiles) du réseau et E modélise l'ensemble les connexions qui existent entre ces nœuds.
Si $e = (u, v) \in E$, cela veut dire que les nœuds u et v sont en mesure de communiquer directement à l'instant t.

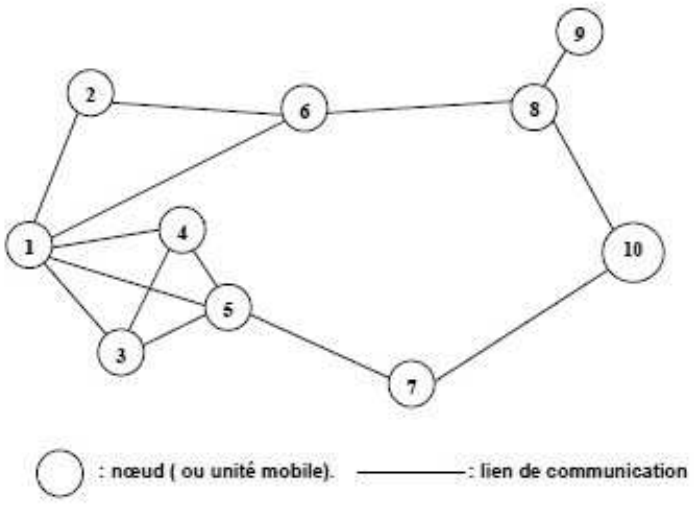


FIGURE 1.2 – Modélisation des réseaux Ad hoc

- Limites dues au support de transmission :

  **Partage du support de transmission :** les stations opèrent sur la même bande de fréquence ce qui peut engendrer des collisions.

  **Taux d'erreurs élevé :** les réseaux sans fil utilisent les ondes radios pour communiquer. Ces ondes ne peuvent pour autant s'affranchir des contraintes liées à leur medium de transmission, l'air. Les perturbations électromagnétiques, solaires ou les obstacles affectent les signaux transmis et sont de fait source de taux d'erreur en bit particulièrement élevés

  **Faible débit :** La modestie des débits des réseaux sans fil est un élément souvent mis en avant. Comparés à certains réseaux filaires, les débits peuvent paraitre faibles. Et dans le cadre de transferts multimédia nécessitant des échanges de données soutenus, ces débits peuvent ainsi poser problème.

  **Sécurité :** Les signaux étant diffusés, ils peuvent être écoutés par toute station mobile se trouvant dans la même zone de couverture. La confidentialité de certaines informations nécessite l'utilisation de mécanismes de sécurité adéquats.

- Limites dues aux stations mobiles :

  **Partage du support de transmission :** les stations opèrent sur la même bande de fréquence ce qui peut engendrer des collisions.

  **Faible puissance :** Les stations mobiles sont la plupart du temps conçues pour une utilisation mobile. De fait, elles se doivent d'être légères, de petite taille et surtout doivent être capables de fonctionner de manière autonome (sur batterie). La prise en compte de tous ces éléments participe à la faible puissance de l'électronique embarquée.

  **Duré d'utilisation restreinte :** Les batteries ont une durée de vie limitée. De fait, le temps d'utilisation nomade d'une station est contraint par la capacité de sa batterie mais aussi par la puissance demandée (ressources processeur ou transmissions sans fil). Il est nécessaire de trouver un juste milieu entre ces composantes.

  **Rayon d'action :** La zone de couverture est fonction de la puissance d'émission que peut fournir une station. Le standard IEEE 802.11 définit la puissance maximale à 100mW. Réduire la puissance d'émission, pour notamment économiser de l'énergie, peut engendrer des liens unidirectionnels.

**Le routage dans les réseaux Ad hoc :** Le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste à déterminer un acheminement optimal

des paquets à travers le réseau au sens d'un certain critère de performance. Le problème consiste à trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa réception en cas de n'importe quelle panne d'arc ou de nœud [5].

Les réseaux ad hoc étant de nature multi-sauts, le protocole de routage détermine une route entre un nœud source et un nœud destination. De par la faible bande passante offerte par les réseaux ad hoc et du fait de la diffusion des données, les protocoles de routage actuellement utilisés dans les réseaux filaires ne peuvent être utilisés, sans modifications, dans les réseaux MANETs. De fait, de nouveaux protocoles de routage ont dû être développés.

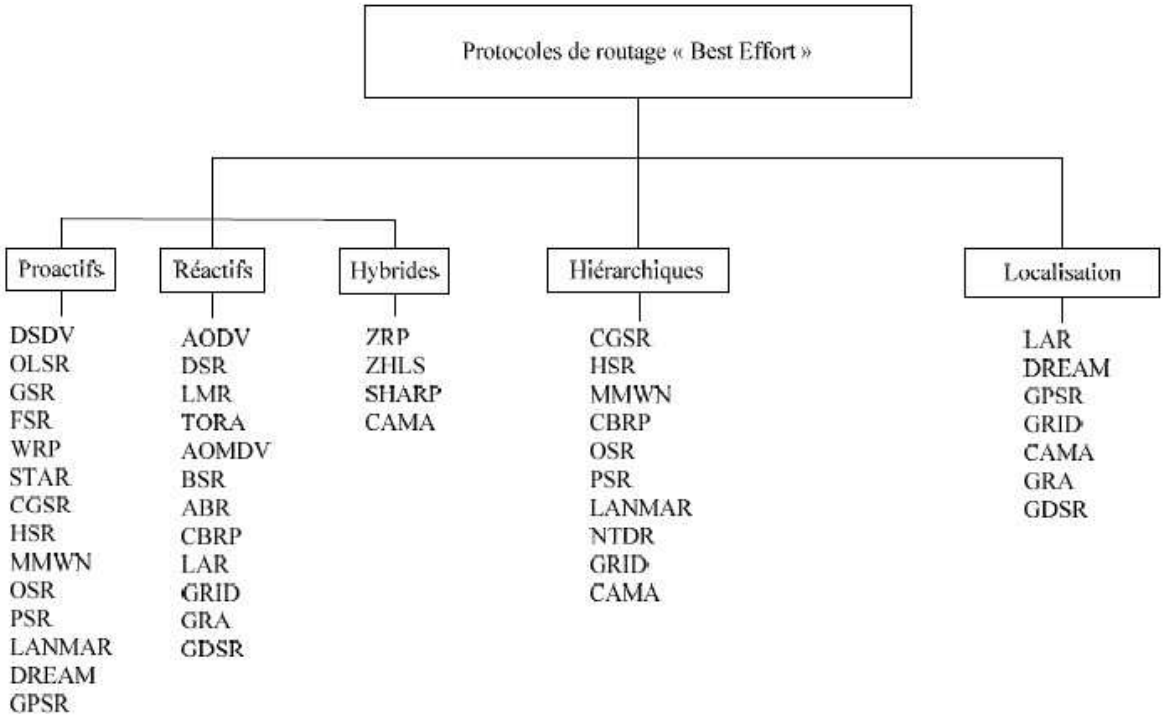


FIGURE 1.3 – Modélisation des réseaux Ad hoc

### Protocoles de routage

**Protocole proactifs :** Chaque nœud, employant un protocole de routage proactif, conserve la route nécessaire pour atteindre n'importe quel autre nœud du réseau. Chaque nœud maintient une table de routage contenant les informations nécessaires (par exemple le prochain nœud sur le chemin...) pour atteindre un autre nœud du réseau. En consultant sa table de routage, un nœud peut à tout instant transmettre un paquet de données vers un autre nœud du réseau. Des mises à

jour périodiques de l'état de la topologie gardent effectives les routes présentes dans la table de routage. Les performances de ce type de protocoles souffrent du trafic additionnel nécessaire au maintient de l'état des routes. Pour conserver des routes valides, le rafraîchissement des informations sur la topologie dépend de la mobilité des nœuds du réseau MANET. Si le rafraîchissement est trop élevé, comparé à l'évolution de la topologie, le nombre d'informations de routage émises sur le réseau est trop important, consommant inutilement de la bande passante. A contrario s'il est trop faible, les tables de routage ne sont pas suffisamment mises à jour, rendant les informations qu'elles contiennent obsolètes. Pour un fonctionnement optimal de ce type de protocoles, un compromis entre l'échange des informations de routage et la prise en compte de l'évolution de la topologie doit être trouvé [9].

Dans un premier temps, nous présentons les protocoles DSDV et OLSR. Ces protocoles sont les protocoles proactifs les plus répandus dans la littérature. Le protocole DSDV est une référence de part son ancienneté, alors que le protocole OLSR est le seul représentant standardisé des protocoles proactifs.

**Protocole DSLV :** L'algorithme de routage proactif étudié dans cette partie est l'algorithme de **routage Destination- Sequenced Distance-Vector** (DSDV). Il est basé sur l'algorithme du **vecteur de distance** utilisé dans RIP [RFC 2453]. Le protocole à vecteur de distance permet de limiter l'échange des messages de contrôle de la topologie uniquement aux voisins d'un nœud. Ce point est extrêmement important pour préserver la bande passante disponible sur le réseau.

Le protocole **DSDV** utilise les propriétés de la diffusion pour transmettre les informations de routage. En effet, le grand avantage de la diffusion est qu'une trame émise par une station est entendue par l'ensemble de ses voisins [5].

**Protocole OLSR :** Le protocole **Optimized Link State Routing** (OLSR) a été standardisé en 2003. Son fonctionnement est basé sur l'algorithme à état de liens. Un nœud du protocole à état de liens diffuse sa connaissance des voisins à l'ensemble de la topologie. De nombreux changements ont dû y être apportés pour être exploitable dans un réseau ad hoc. La bande passante étant limitée la diffusion de ses voisins à l'ensemble des nœuds du réseau est bien trop coûteuse. Le protocole OLSR prend en compte les spécificités de la diffusion (un paquet émis est reçu par l'ensemble des nœuds dans

son voisinage immédiat) pour réduire le nombre de paquets nécessaires à l'échange de la topologie.

Chaque nœud doit déterminer l'ensemble de ses voisins. Pour cela périodiquement, ils transmettent des paquets, dits Hello, pour se faire connaître. Ce type de paquet comprend la totalité de la base de liens connue par l'émetteur du paquet. La base de liens d'un nœud regroupe l'ensemble des nœuds lui ayant transmis un paquet Hello. A la réception des paquets Hello, chaque nœud dans le réseau connaît les nœuds situés dans son voisinage immédiat

**Protocoles réactifs :** Les protocoles de routage réactifs (ou sur demande) ne maintiennent une route que si elle est utilisée. Lorsqu'un nœud source a besoin de transmettre des données vers un nœud destination, il doit au préalable déterminer une route. Pour cela, des informations de contrôle sont transmises sur le réseau. Comparés aux protocoles proactifs qui conservent les routes vers l'ensemble des stations du réseau dans leur table de routage, les protocoles réactifs ne conservent que les routes qui ont une utilité. Par conséquent, la taille des tables de routage contenues en mémoire est moins importante que pour les protocoles proactifs.

Dans un premier temps, nous présentons les protocoles réactifs **AODV** et **DSR**. Ces protocoles sont les protocoles réactifs ayant le plus de chances d'être utilisés dans les réseaux MANET du fait qu'ils soient standardisés [5].

**Protocole AODV :** Le protocole de routage **Ad hoc On-Demand distance Vector** (AODV) permet le maintien des routes utilisées. En fait, si le changement de statut d'un lien n'affecte pas une communication, aucun échange entre les nœuds n'est donc nécessaire. Les effets des changements de topologie sont ainsi localisés seulement aux routes rencontrant ces modifications et non à la globalité du réseau. Ce protocole est opérationnel seulement dans un environnement où les liens sont symétriques. Ce protocole met en œuvre différentes opérations pour réaliser et maintenir le routage : **gestion de la connectivité locale, phase de découverte des routes, maintenance des routes** [5].

**Protocole DSR :** Le protocole **Dynamic Source Routing** (DSR) a été standardisé en 2007. Son fonctionnement est très proche du protocole AODV à la grande différence qu'il fournit dans les paquets de données l'ensemble des nœuds permettant d'atteindre une destination (routage par la source). Cet ajout dans les paquets de données accroît le surcoût et consomme un

peu plus de bande passante. A contrario, ces informations lui permettent de gérer l'asymétrie des liens présents dans le réseau. En effet, un paquet de données peut prendre une route différente de son acquittement. Le fonctionnement basique de DSR s'avère assez simple à mettre en œuvre. Il met en place uniquement deux phases : la phase de découverte des routes et la phase de maintenance de ces mêmes chemins [5].

**Protocole hybride :** Dans un souci de préserver la bande passante, les protocoles de routage hybrides combinent les avantages des protocoles proactifs et réactifs. Lorsqu'il faut traverser un grand nombre de nœuds, les protocoles réactifs deviennent plus intéressants au niveau de la consommation en bande passante. Excepté la latence qui augmente, ce type de protocoles fournit de nombreux avantages pour les topologies avec un nombre élevé de nœuds. En effet, l'entretien des routes est beaucoup plus facile, car seulement les routes utilisées ont besoin d'être mises à jour lors d'une modification de la topologie.

Les protocoles proactifs sont plus performants dans des réseaux ayant un faible nombre de nœuds. En effet, ils connaissent à tout moment au moins une topologie partielle du réseau et donc peuvent déterminer immédiatement le prochain nœud en direction de la destination. Aucune latence au niveau de l'émetteur ne se fait donc ressentir. La consommation de bande passante est dans ce cas relativement minime car peu de stations sont présentes dans le réseau.

Les protocoles hybrides vont donc tirer avantage de ces deux protocoles. Un nœud va utiliser, dans son proche entourage, un algorithme de routage proactif. Ainsi, chaque nœud a une connaissance globale de son voisinage. Puis à l'extérieur de son entourage immédiat, il va utiliser un algorithme de routage réactif. Ce type d'algorithme s'inspire du comportement humain, c'est-à-dire que nous avons une bonne connaissance du quartier où l'on habite, mais plus on s'en éloigne, plus on ne connaît que les axes pour atteindre notre lieu de destination et pas ce qui l'entoure.

**Protocole ZRP :** Le protocole de routage hybride le plus répandu est le protocole **Zone Routing Protocol** (ZRP). Ce protocole découpe la topologie du réseau en deux zones. La première zone est celle dans le voisinage de chaque nœud, elle est appelée Intrazone. En fait, c'est l'ensemble des nœuds qui se

trouvent à un nombre de sauts inférieur ou égal à Hmax. La seconde zone est la zone extérieure à un nœud, appelée Interzone, c'est-à-dire l'ensemble des nœuds qui se trouvent à un nombre de sauts supérieur à Hmax.

Pour déterminer le chemin pour joindre une destination, deux protocoles de routage vont être employés suivant la zone dans laquelle se trouve la destination. Ainsi, si la destination se situe dans l'Intrazone, le protocole de routage proactif Intrazone Routing Protocol (**IARP**) est utilisé. Si la destination est extérieure à cette zone, le protocole de routage réactif Interzone Routing Protocol (**IERP**) est employé.

Le protocole de routage **IARP** est basé sur un protocole à **état de liens**. Chaque nœud diffuse, périodiquement, sa connaissance de ses voisins. A l'aide des informations diffusées, les nœuds construisent la topologie et déterminent les routes vers les nœuds situés à proximité. Pour éviter que la diffusion des paquets de contrôle se propage sur la totalité du réseau, la source met le champ TTL à la valeur de Hmax, le nombre de saut maximum auquel se limite l'Intrazone. Chaque fois qu'un nœud reçoit un tel paquet, il met à jour sa table de routage puis décrémente de 1 le champ TTL du paquet. Si ce champ est égal à 0 le paquet est supprimé sinon il est propagé.

Lorsque le nœud source ne connaît pas de chemin vers la destination, c'est qu'elle ne se trouve pas dans l'Intrazone. Il utilise le protocole IERP pour déterminer un chemin jusqu'à elle. Le protocole IERP est responsable uniquement des communications entre les différentes zones. La source détermine un ensemble de nœuds frontières à son Intrazone. Elle utilise ces nœuds pour déterminer un chemin jusqu'à la destination, tout en réduisant le délai et le surcoût pris par la recherche. Lors de la réception de la requête de demande de création de route, les nœuds frontières ajoutent leur identifiant dans l'entête de la requête. Ensuite, deux procédures sont appliquées selon que ces nœuds connaissent une route vers la destination ou pas :

- La destination est dans l'Intrazone d'un nœud frontière : une réponse est envoyée à la destination en prenant le chemin inverse contenu dans l'entête de la requête.
- La destination ne se situe pas dans l'Intrazone d'un nœud frontière : la requête est propagée à l'ensemble de ses nœuds frontières et l'opération recommence jusqu'à déterminer un chemin.

Il ya d'autres protocoles hybrides comme : ZHLS et SHARP, Ces protocoles réduisent le trafic de contrôle pour déterminer une route avec un nœud situé dans l'Interzone.

# 3 Conclusion

Ce chapitre nous a permis de comprendre le principe de fonctionnement des réseaux sans fil, malgré leurs avantages, leurs facilités d'utilisation, ils présentent quelque inconvénients, on peut dire que le problème majeur de cette technologie (le sans fil) après le problème de sécurité est, la volatilité des liaisons et des noeuds représentés par les machines avec des contraintes d'énergies limitées.

# Chapitre 2

# Analyse numérique parallèle et distribuée

## 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.

## 2 Systèmes Distribués

### 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'exécution de différentes classes d'applications, ayant des exigences pouvant évoluer au cours du temps. Ces exigences concernent les aspects temporels (temps processeur, bande passante réseau...), mais aussi la sécurité, la protection, la tolérance aux fautes, la mobilité.

**Définition (Leslie Lamport) :** A distributed system is one on 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 principes de base régissant la communication, la gestion de l'information, le partage des ressources, la tolérance aux fautes [10].
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).

Le tableau *(2,1)* montre bien la différence et les caractéristiques des systèmes centralisés et distribués :

## 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.

**Besoins propres des applications :**

- 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.

| | |
|---|---|
| **Systèmes centralisés** :<br>– Une seule machine pour plusieurs tâches.<br>– Mémoire commune.<br>– Communication et synchronisation par partage de données en mémoire.<br>– Simple à mettre en œuvre.<br>– Pose le problème de la protection des données par rapport à des accès multiples (problème de cohérence) lorsque les opérations ne sont pas atomiques.<br>– Mécanismes de synchronisation : Sémaphores (Dijkstra 1965), Moniteurs. | **Systèmes distribués** :<br>– Persistence des données.<br>– Échange de données entre applications hétérogènes.<br>– Répartition des données sur des sites distants.<br>– Gestion de la cohérence permanente des données.<br>– Interopérabilité des plates-formes.<br>– Portabilité des applications.<br>– Gestion des accès concurrents.<br>– Ouverture et Sécurité.<br>– L'indépendance des activités des processus.<br>– La topologie du réseau est inconnue.<br>– La délivrance des messages est incertaine.<br>– Un mécanisme éventuel de tolérance aux pannes.<br>– Il n'existe pas de composante maître qui gère les autres et chacune est donc responsable de son propre fonctionnement.<br>– Plus évolutif : simple à étendre+plus disponible. |

TABLE 2.1 – Caractéristiques des systèmes centralisés et distribués

## 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, calcule 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).
- Télévision interactive.
- Agents intelligents.

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

# 3 Grilles informatiques

## 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 [11].

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.

## 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 *(Figure(2,1))*. Ainsi, chaque site d'une grille peut utiliser ces propres machines, avec son propre système d'exploitation. 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 milliers. 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'exécuter 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,

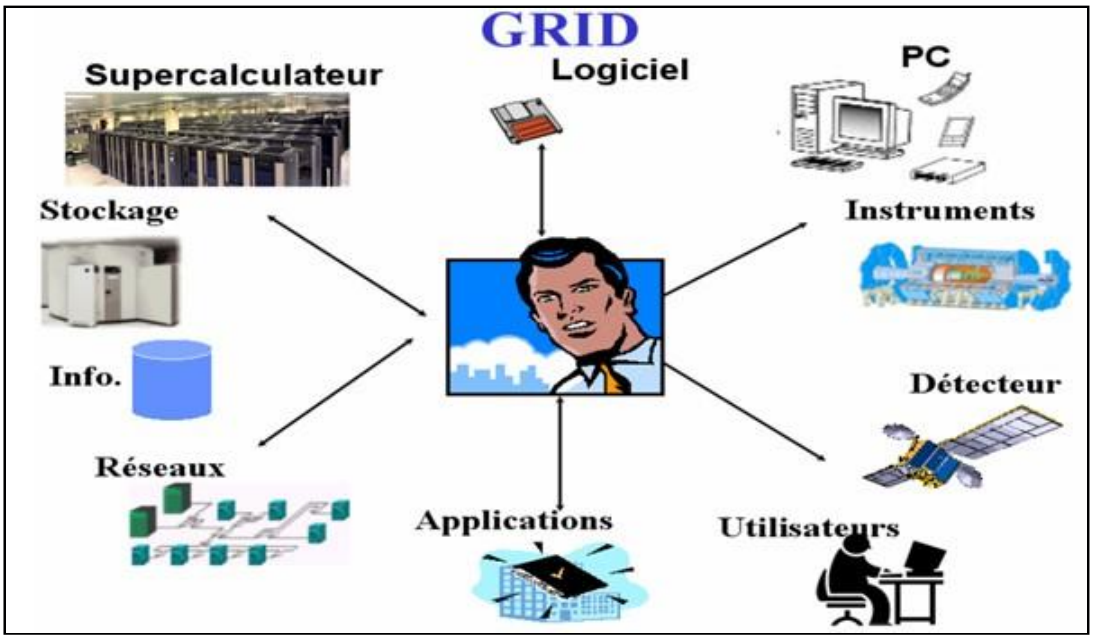


FIGURE 2.1 – Grille informatique

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 [2].
*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 70jours.

**Fournir un cadre distribué :** Une grille de calcul peut agréger une importante 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.

**Fiabilités 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 grand nombre 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.

# 4 Communications dans les systèmes distribués

## 4.1 Rappels des protocoles OSI

Les constructeurs informatiques ont proposé des architectures réseaux propres à leurs équipements. Par exemple, IBM a proposé SNA, DEC a proposé DNA... Ces architectures ont toutes le même défaut : du fait de leur caractère propriétaire, il n'est pas facile de les interconnecter, à moins d'un accord entre constructeurs. Aussi, pour éviter la multiplication des solutions d'interconnexion d'architectures hétérogènes, l'ISO (International Standards Organisation), organisme dépendant de l'ONU est composé de 140 organismes nationaux de normalisation, a développé un modèle de référence appelé modèle OSI [12] (Open Systems Interconnection). Ce modèle décrit les concepts utilisés et la démarche suivie pour normaliser l'interconnexion de systèmes ouverts (un réseau est composé de systèmes ouverts lorsque la modification, l'adjonction ou la suppression d'un de ces systèmes ne modifie pas le comportement global du réseau).

Au moment de la conception de ce modèle, la prise en compte de l'hétérogénéité des équipements était fondamentale. En effet, ce modèle devait permettre l'interconnexion avec des systèmes hétérogènes pour des raisons historiques et économiques. Il ne devait en outre pas favoriser un fournisseur particulier. Enfin, il devait permettre de s'adapter à l'évolution des flux d'informations à traiter sans remettre en cause les investissements antérieurs. Cette prise en compte de l'hétérogénéité nécessite donc l'adoption de règles communes de communication et de coopération entre les équipements, c'est à dire que ce modèle devait logiquement mener à une normalisation internationale des protocoles.

#### Les différentes couches du modèle OSI

**Couche physique :** assurer la transmission ,connecteurs physiques, codages physiques adaptation au mode de transmission,...

**Couche de liaison :** assurer la transmission sans erreur entre deux extrémités d'une liaison directe.

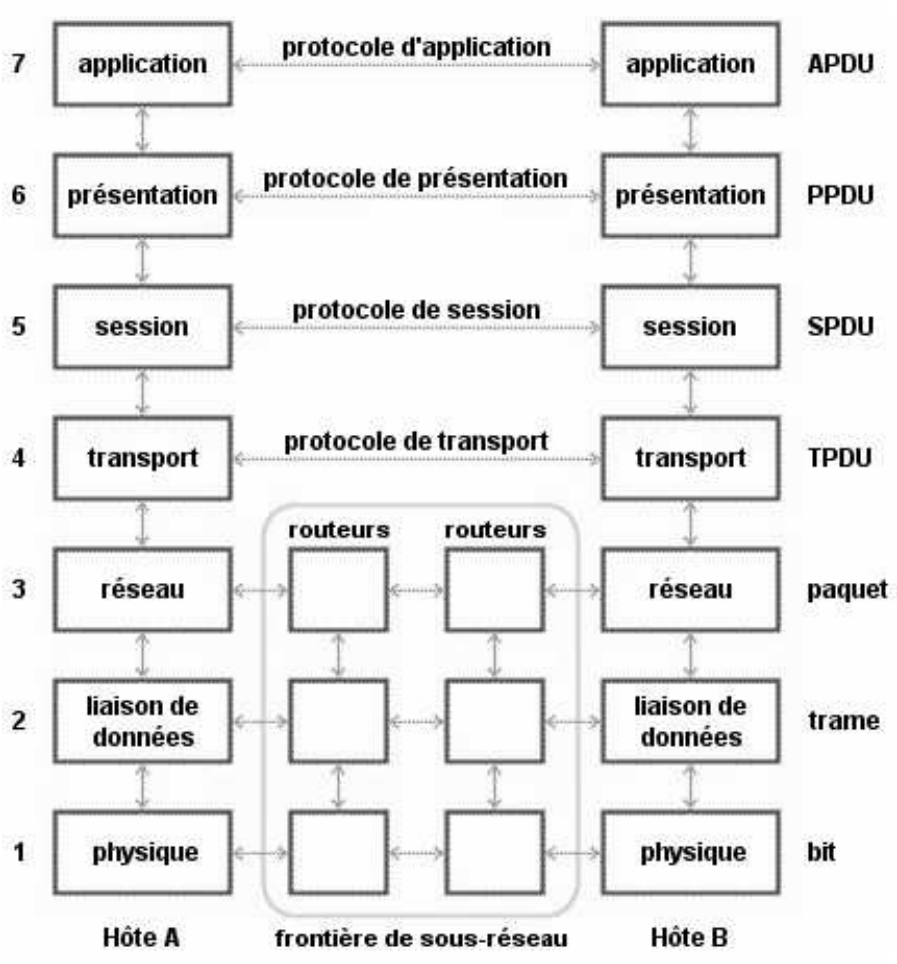


FIGURE 2.2 – Différentes couches du modèle OSI

**Couche réseau :** assurer le routage des messages sur les réseaux d'interconnexion (gestion de congestion, choix de chemins,...)

**Couche transport :** assurer une transmission fiable entre l'expéditeur et le destinataire.

**Couche de session :** extension de la couche transport : mécanismes de synchronisation.

**Couche de présentation :** format de données, adaptation au terminal.

**Couche d'application :** protocoles applicatifs : courrier électronique (X400), Ftp, Telnet, ...

### 4.1.1 Transmission de données à travers du modèle OSI

Le processus émetteur remet les données à envoyer au processus récepteur à la couche application qui leur ajoute un en-tête application AH (éventuellement nul). Le résultat est alors transmis à la couche présentation.

La couche présentation transforme alors ce message et lui ajoute (ou non) un nouvel en-tête (éventuellement nul). La couche présentation ne connaît et ne doit pas connaître l'existence éventuelle de AH ; pour la couche présentation, AH fait en fait partie des données utilisateur. Une fois le traitement terminé, la couche présentation envoie le nouveau "message" à la couche session et le même processus recommence.

Les données atteignent alors la couche physique qui va effectivement transmettre les données au destinataire. A la réception, le message va remonter les couches et les en-têtes sont epteur, la figure *(2,3)* décrit l

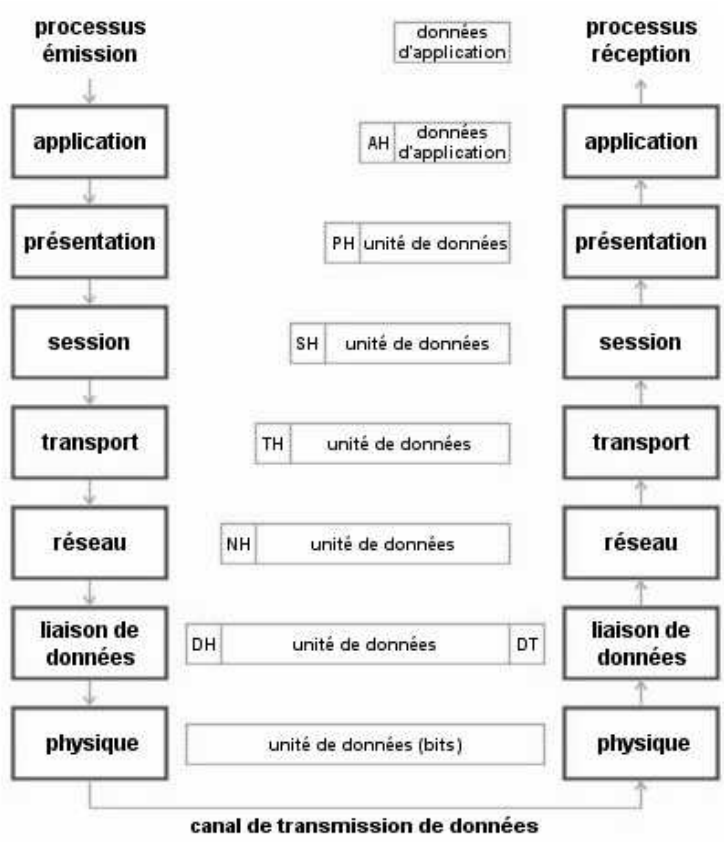


FIGURE 2.3 – Communication entre les couches dans OSI

## 4.2 Modèle TCP/IP

TCP/IP [1] désigne communément une architecture réseau, mais cet acronyme désigne en fait 2 protocoles étroitement liés : un protocole de transport, TCP (Transmission Control Protocol) qu'on utilise "par-dessus" un protocole réseau, IP (Internet Protocol). Ce qu'on entend par "modèle TCP/IP", c'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant, car ils en constituent l'implémentation la plus courante. Par abus de langage, TCP/IP peut donc désigner deux choses : le modèle TCP/IP et la suite de deux protocoles TCP et IP.

Le protocole TCP *(Transmission Control Protocol : Protocole de Contrôle de Transmission)* est défini dans le but de fournir un service de transfert de données de haute fiabilité (à la différence de UDP) entre deux ordinateurs "maîtres" raccordés sur un réseau de type « paquets commutés »et sur tout système résultant de l'interconnexion de ce type de réseaux.

### 4.2.1 Caractéristiques principales du protocole TCP :

- TCP permet de remettre en ordre les datagrammes en provenance du protocole IP.
- TCP permet de vérifier le flot de données afin d'éviter une saturation du réseau.
- TCP permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP.
- TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne.
- TCP permet l'initialisation et la fin d'une communication de manière normalisée.

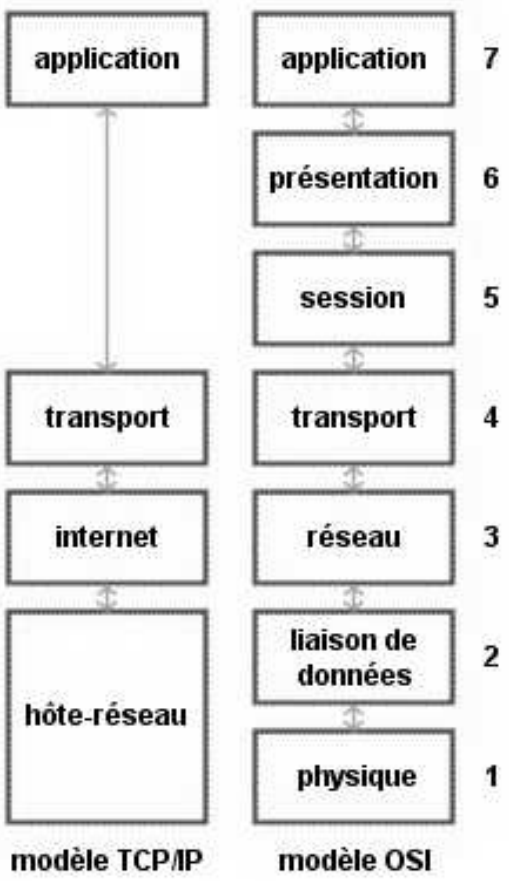


FIGURE 2.4 – Modèle TCP/IP [1]

### 4.2.2 Comparaison avec le modèle OSI

Tout d'abord, les points communs. Les modèles OSI et TCP/IP sont tous les deux fondés sur le concept de pile de protocoles indépendants. Ensuite, les fonctionnalités des couches sont globalement les mêmes.

Au niveau des différences, on peut remarquer la chose suivante : le modèle OSI faisait clairement la différence entre 3 concepts principaux, alors que ce n'est plus tout à fait le cas pour le modèle TCP/IP. Ces 3 concepts sont les concepts de services, interfaces et protocoles.

Une autre différence est liée au mode de connexion. Certes, les modes orienté connexion et sans connexion sont disponibles dans les deux modèles mais pas à la même couche : pour le modèle OSI, ils ne sont disponibles qu'au niveau de la couche réseau (au niveau de la couche transport, seul le mode orienté connexion n'est disponible), alors qu'ils ne sont disponibles qu'au niveau de la couche transport pour le modèle TCP/IP (la couche internet n'offre que le mode sans connexion). Le modèle TCP/IP a donc cet avantage par rapport au modèle OSI : les applications (qui utilisent directement la couche transport) ont véritablement le choix entre les deux modes de connexion [1].

## 4.3 Modèles de déploiement

### 4.3.1 Modèle Client/Serveur

L'architecture client-serveur est un modèle de fonctionnement logiciel qui peut se réaliser sur tout type d'architecture matérielle (petites ou grosses machines), à partir du moment ou ces architectures peuvent être interconnectées et des fois même en mode non connecté, sans aucune liaison à un réseau.

On parle de fonctionnement logiciel dans la mesure où cette architecture est basée sur l'utilisation de deux types de logiciels, à savoir un logiciel serveur et un logiciel client s'exécutant normalement sur 2 machines différentes. L'élément important dans cette architecture est l'utilisation de mécanismes de communication entre les 2 applications.

**Le dialogue entre les applications peut se résumer par :**

- Le client demande un service au serveur.
- Le serveur réalise ce service et renvoie le résultat au client.

(Un des principes fondamental est que le serveur réalise un traitement pour le client.)

### 4.3.2 Différents modèles de client-serveur

En fait, les différences sont essentiellement liées aux services qui sont assurés par le serveur, On distingue couramment :

***Le client-serveur de donnée :*** Dans ce cas, le serveur assure des tâches de gestion, stockage et de traitement de données. C'est le cas le plus connu de client-serveur est qui est utilisé par tous les grands SGBD.

La base de données avec tous ses outils (maintenance, sauvegarde...) est installée sur un poste serveur.

Sur les clients, un logiciel d'accès est installé permettant d'accéder à la base de données du serveur.

Tous les traitements sur les données sont effectués sur le serveur qui renvoie les informations demandées (souvent à travers une requête SQL) par le client.

***Client-serveur de présentation :*** Dans ce cas la présentation des pages affichées par le client est intégralement prise en charge par le serveur. Cette organisation présente l'inconvénient de générer un fort trafic réseau.

- Déporter l'affichage sur un réseau.
- Telnet.
- Xwindows.
- NTTeerminal Serveur.
- Le développement est " presque " centralise.
- Architecture dénommée " client léger ".

***Le client-serveur de traitement :*** Dans ce cas, le serveur effectue des traitements à la demande du client. Il peut s'agir de traitement particulier sur des données, de vérification de formulaires de saisie, de traitements d'alarmes...

Ces traitements peuvent être réalisés par des programmes installé sur des serveurs mais également intégrés dans des bases de données (triggers, procédures stockées), dans ce cas, la partie donnée et traitement sont intégrés.

- Le serveur est un serveur de base de données.
- Architecture dénommée « client obèse ».

Ce modèle est la première infrastructure informatique pour un travail coopératif qui se caractérise par la centralisation des traitements au niveau du serveur et ne fait pas de duplication pour une gestion plus simple de la cohérence et de l'intégrité des données.

La relation directe entre le client et le serveur, l'augmentation de l'hétérogénéité, l'absence de portabilité et l'absence de transparence de localisation pourra être vue comme des inconvénients dans ce modèle.

### 4.3.3 Pair-à-Pair (P2P)

P2P [13] est « peer to peer »en anglais. Communication de pair à pair, illustré dans ci-dessous : les deux machines qui communiquent sont sur un pied d'égalité, elles peuvent être toutes les deux clients ou serveur, voire client et serveur à la fois.

C'est le mode de fonctionnement systématique sur l'internet, mais la distinction client-serveur a provoqué un abus de langage : le sens commun a retenu que les postes de travail individuels sont uniquement des clients. Le P2P désigne alors fait de transformer un tel poste en serveur de fichier plus ou moins évolué [14].

Exemples types de programmes implantant (ou ayant mis en œuvre) ce type de communication : Gnutella, Napster, eMule.

### 4.3.4 Comparaison entre le modèle centralisé et P2P

C'est une liaison poste à poste par opposition au modèle client-serveur. Dans ce type de réseau les ordinateurs sont connectés les uns aux autres sans passer par un serveur central. Ce type de réseau est très utilisé pour les échanges de fichiers pirates (MP3, Films, Images, Livres...), en effet il est impossible de mettre un terme à ces échanges car il n'y a pas de serveur central *(Figure(2,5))*.

Dans tous les cas, un système Pair à Pair présume que toutes les machines coopèrent à une cible commune qui peut être un calcul ou le stockage/partage d'information.

Dans ce modèle, l'échange de donnée est plus directs et plus rapides, il permet l'optimisation de l'utilisation de la bande passante du réseau (équilibrage de la charge de réseau), ainsi d'autres avantages peuvent être localisés dans ce modèle comme la maintenance et coûts réduits, l'extensibilité (passage de 100 à 10000 nœuds sans problème) et l'utilisation des ressources inutilisées. Ce modèle possède aussi un nombre important d'inconvénients comme :

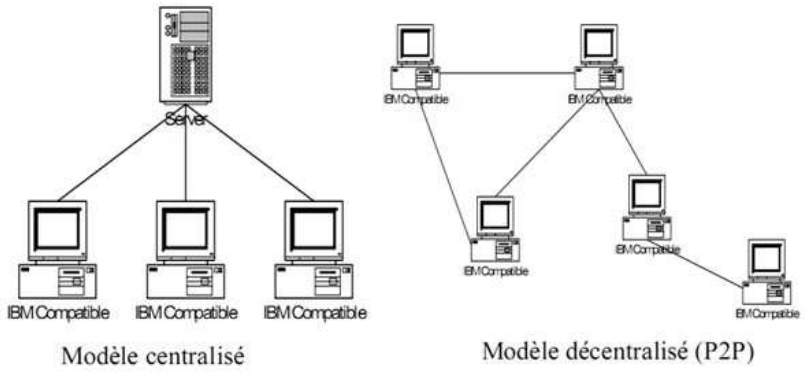


FIGURE 2.5 – Architecture P2P

- La sécurité :crackers,virus,DDoS...
- Contenu trompeur : consistance, contradiction.
- Loi (Wild Wild Web) : droit d'auteurs, contenu immoral.
- Régulation/Répression : application difficile des lois,...
- QoS : ligne peu fiable, débit peu élevé, utilisation des algorithmes plus complexe pour la communication entre les entités du réseau !

Les débits et les latences de communication dans les systèmes distribués étant variables, il est nécessaire d'élaborer des algorithmes prenant ces caractéristiques en compte.

# 5 Algorithmes itératifs synchrones et asynchrones

La résolution de problèmes complexes et de grande taille requiert l'utilisation simultanée de plusieurs ressources de calculs afin d'obtenir une capacité de calcul en théorie illimitée. On a vue que les grappes de calcules permettent de répondre à ce besoin de ressources.

Les grilles de calcul sont hétérogènes au niveau des machines et au niveau des réseaux. Des algorithmes nécessaires sont pris en compte pour répondre à la variation des débits et des latences dans ses communications.

Pour résoudre un problème numérique on distingue les méthodes directes des méthodes itératives. Les premières donnent la solution exacte du problème modulo les erreurs d'arrondi. Les secondes convergent vers la solution du problème par approximation successive de la solution. Les algorithmes itératifs sont utilisés pour résoudre de nombreuses classes de problèmes, citons par exemple :

- l'élimination de Gauss-Jordan pour la résolution d'un système d'équations linéaires.
- l'algorithme du simplexe en optimisation linéaire.

Les méthodes itératives sont généralement assez faciles à programmer et à paralléliser. La parallélisassion la plus simple consiste à distribuer le travail effectué dans une itération aux processeurs. A chaque nouvelle itération, les processeurs s'échangent les données dont ils ont besoin et déterminent si le critère de convergence est atteint. Ces algorithmes parallèles sont synchrones et ont le même comportement que les algorithmes séquentiels dont ils sont dérivés. Néanmoins, les synchronisations peuvent se révéler très couteuses dans un contexte hétérogène et distant. Une solution pour réduire les temps de synchronisations consiste à recouvrir une partie des communications par du calcul.

Les algorithmes parallèles itératifs asynchrones sont bien adaptés au contexte de grappes distantes puisqu'ils ne nécessitent aucune synchronisation. Ces algorithmes supportent l'hétérogénéité des processeurs et des réseaux. Les itérations sont différentes du mode synchrone et il faut étudier minutieusement la convergence des algorithmes asynchrones.

La conception d'un algorithme distribué pour un système asynchrone est bien plus difficile que pour un système synchrone. L'analyse de sa validité doit prendre en compte tous les scenarios possibles, alors qu'un synchrone il n'ya, le plus souvent, qu'un seul scenario possible.

D'un point de vue pratique, peu de personnes utilisent les algorithmes asynchrones. Ceci s'explique probablement par différentes raisons.

- Les algorithmes asynchrones sont beaucoup moins connus que les algorithmes synchrones.
- Les algorithmes asynchrones paraissent plus difficiles à programmer que les algorithmes synchrones.
- Les environnements de programmation parallèle usuels ne sont pas adaptés au développement des algorithmes asynchrones.

## 5.1 Algorithmes Itératifs :

Avant d'aborder le principe des algorithmes itératifs asynchrones, il est indispensable d'aborder le principe des algorithmes itératifs séquentiels.

### 5.1.1 Algorithmes itératifs séquentiels

On s'intéresse à un problème de type : X = f(x)

Le modèle d'algorithme itératif présente la structure suivante sous forme d'´equation ou d'algorithme.

$$x^{k+1} = f(x^k), \qquad \text{k=0,1,...} \qquad \text{(Avec } x^0 \text{ donné)}$$

**Algorithm 1** Modèle d'algorithme itératif séquentiel

1: $x^0$
2: **for** $k = 0$ to ... **do**
3: $\quad x^{k+1} = f(x^k)$
4: **end for**

Modèle d'algorithme itératif séquentiel

Les $X^k$ sont des vecteurs de dimmensiion n et $f$ est une fonction de $R^n$ dans $R^n$. Si la suite des itérés $X^k$ engendrée par l'algorithme ci-dessus converge et si $f$ est continue, alors la propriété $x^* = f(x^*)$ est vérifiée et $x^*$ est solution de cet algorithme.

Pour mettre en œuvre un algorithme itératif séquentiel, on définit un seuil d'arrêt afin de terminer l'algorithme dès lors que celui-ci est passé sous le seuil. La plupart des algorithmes utilisent un critère défini à partir d'une norme sur les 2 dernières itérations. Par exemple, on peut choisir d'appliquer la norme :

$$\max_{i \in 1..n} |x_i^{k+1} - x_i^k| < \epsilon$$

Où $\epsilon$ représente le critère d'arrêt et $x_i^k$ représente la composante i du vecteur x à l'itération k. En séquentiel, il existe d'autre normes qui sont applicables pour calculer le résidu entre 2 itérations. Néanmoins, pour la suite, nous utiliserons cette norme appelée norme du max.

Dans les algorithmes itératifs parallèles, les données et routines de calcul sont réparties sur plusieurs processeurs. Ces algorithmes peuvent être divisés en deux grandes catégories : les algorithmes synchrones et les algorithmes asynchrones.

### 5.1.2 Algorithmes itératifs parallèles synchrones

Le parallélisme le plus simple des algorithmes itératifs consiste à découper le vecteur $x^k$ en $m$ blocks de composantes et à partitionner la fonction en $m$ parties de manière compatible. Ainsi on obtient :

$$x^k = (x_1^k, x_2^k, ..., x_n^k) \equiv X^k = (X_1^k, X_2^k, ..., X_m^k)$$

L'équation précédente peut alors s'écrire sous la forme suivante :

$$(x_1^{k+1}, x_2^{k+1}, ..., x_m^{k+1}) = (F_1(X^k), F_2(X^k), ..., F_m(X^k)) \quad k = 1, 2, ...$$

Voici le modèle d'algorithme itératif synchrone correspondant :
A partir de cette formulation, un algorithme itératif peut être parallélisé en associant un bloc de composantes à chaque processeur.

**Algorithm 2** Modèle d'algorithme itératif synchone [2]

1: $(x_1^0, .., x_m^0)$ est fixé
2: **for** $k = 0$ to ... **do**
3: **for** $m = 0$ to ... **do**
4: $x_1^{k+1} = Fi(X_1^k, .., X_m^k)$
5: **end for**
6: **end for**

A chaque itération, le processeur chargé de calculer le bloc de composantes i connaît les valeurs de toutes les composantes impliquées dans le calcul de $F_i$, il calcule les nouvelles composantes $X_i^{k+1}$ et les envoie à tous les processeurs qui en ont besoin pour calculer l'itération suivante, ces étapes sont bien mentionnées dans la figure *(2,6)*.

Il est alors possible de proposer un algorithme générique qui peut s'adapter simplement en fonction du problème que l'on souhaite résoudre. Selon les modes de communications employés, on peut différencier deux variantes d'algorithmes.

La forme la plus simple consiste à utiliser des communications synchrones. Nous qualifions l'algorithme obtenu d'ISCS pour Itérations Synchrones avec Communication Synchrones. Avec cette variante, les communications sont effectuées à la fin d'une itération (donc après la phase de calcul). L'algorithme qui vient schématise le principe d'ISCS. Le tableau *NouvellesDonnéesLoc* (qui représente $X_i^{k+1}$) ne contient que les données locales d'un processeur.

Le déroulement d'un algorithme ISCS est représenté sur la figure ci-dessous. On voit qu'il ya des périodes d'inactivité dues aux échanges de données et aux calculs liés à la détections de convergence (les communications sont représentées par des flèches). Les algorithmes ISCA sont un peu plus rapides car les communications tendent à être masquées par les calculs, mais les périodes d'inactivité ne sont pas totalement supprimées pour autant. L'avantage de ces algorithmes synchrones parallèles est que la convergence est exactement la même que celle de leurs pendants séquentiels.

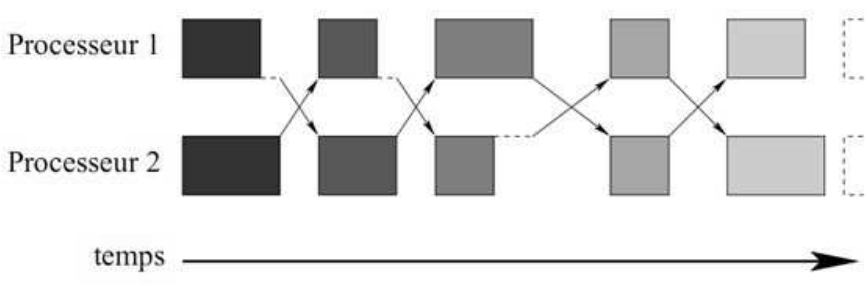


FIGURE 2.6 – Algorithmes ISCS [2]

**Algorithm 3** Algorithme itératif parallèle ISCS

*Initialisation de la bibliothèque de communication*
*AnciennesDonnées = Tableau des valeurs de l'itération précédente*
*NouvellesDonnéesLoc = Tableau des nouvelles valeurs locales*
*Initialisation des données*

**Repeat**
- Calcul des NouvellesDonnéesLoc en fonction des AnciennesDonnées
- Envois non bloquants des données de NouvellesDonnéesLoc aux processeurs qui ont besoin
- Réception bloquante des données des processeurs voisins dans AnciennesDonnées
- Recopie de NouvellesDonnéesLoc dans AnciennesDonnées
- Détection de convergence

**Until** *Convergence globale atteinte*

La seconde variante consiste à utiliser des communications asynchrones. L'avantage de cette variante est de pouvoir recouvrir une partie des communications par du calcul. Ainsi, l'exécution de l'algorithme est, à priori, plus rapide. Nous appelons cette variante ISCA pour Itérations Synchrones avec Communications Asynchrones. Au niveau algorithmique, les communications sont légèrement différentes afin de traiter le recouvrement. Selon le calcul considéré, il faut envoyer les données aux processeurs qui en ont besoin le plus rapidement possible afin de poursuivre le calcul et pour qu'une partie des communications soit effectuée en même temps que la fin du calcul de l'itération *(Figure(2,7))*.

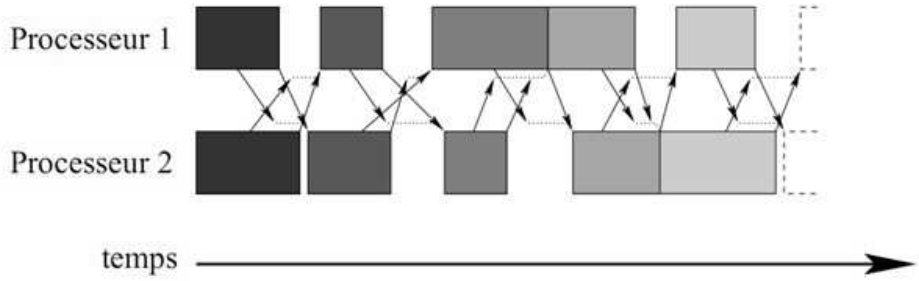


FIGURE 2.7 – Algorithmes ISCA

Ces deux variantes engendrent les mêmes suites d'itérés que les algorithmes séquentielles leur correspondant. L'intérêt des algorithmes ISCS et ISCA est qu'ils convergent dès lors que la version séquentielle converge. La détection de convergence au niveau pratique ne pose pas de difficulté. En effet, il suffit que chaque processeur applique le critère d'arrêt sur ces données locales. Ensuite les processeurs combinent leurs résultats afin d'obtenir la convergence globale. C'est cette phase de combinaison qui réalise une synchronisation.

**Algorithm 4** Algorithme itératif parallèle ISCA [2]

*Initialisation de la bibliothèque de communication*
*AnciennesDonnées = Tableau des valeurs de l'itération précédente*
*NouvellesDonnéesLoc = Tableau des nouvelles valeurs locales*
*Initialisation des données*

**Repeat**
- Calcul de NouvellesDonnéesLoc en fonction des AnciennesDonnées avec envois non bloquants des données de NouvellesDonnéesLoc aux processeurs qui ont besoin (tous les envois à l'exception du dernier sont susceptibles d'être recouverts par du calcul)
- Réception bloquante des données des processeurs voisins dans AnciennesDonnées
- Recopie de NouvellesDonnéesLoc dans AnciennesDonnées
- Recopie de NouvellesDonnéesLoc dans AnciennesDonnées
- Détection de convergence

**Until** *Convergence globale atteinte*

### 5.1.3 Algorithmes itératifs parallèles asynchrones

A partir du découpage du vecteur $X^k$ obtenu précédemment, on peut obtenir un algorithme parallèle itératif asynchrone en désynchronisant les itérations. Les algorithmes asynchrones sont plus généraux que les algorithmes synchrones. Autrement dit, un algorithme synchrone n'est qu'un cas particulier d'un algorithme asynchrone (dans lequel les itérations sont synchronisées). Ceci étant dit, les différences surviennent au niveau du modèle et de l'implantation. En désynchronisant les itérations, les communications sont forcément asynchrones. Afin de rester cohérent avec les dénominations précédentes, nous renommons les algorithmes asynchrones en algorithmes IACA. Cet acronyme a pour signification Itération Asynchrones avec Communications Asynchrones. Cette dénomination possède l'avantage de lever l'ambiguïté sur le mot "asynchrone" qui concerne les itérations et les communications [15].

Rappelons dans un premier temps le modèle d'un algorithme IACA ou algorithme parallèle itératif asynchrone. Partons de la décomposition par blocs de composantes $X^k$ obtenue dans la section précédente. Dans le modèle asynchrone, la dernière version d'un bloc de composantes $X_i$ n'est pas forcément disponible sur tous les processeurs, seule une version antérieure est disponible. Le nœud i responsable du calcul du bloc $X_i$ possède, bien entendu, la dernière version. Notons $X_i^{S_i^j}$ la dernière version disponible du bloc de composantes i sur le processeur j à l'itération k avec $0 \leqslant X_i^{S_i^j} \leqslant k$. Soit $j(k)$ l'ensemble des nœuds mis à jour à l'itération k, un algorithme IACA est modélisé par :

**Algorithm 5** Modèle d'algorithme IACA [2]

```
1: for k = 0 to ... do
2:    for m = 0 to ... do
3:       if  i ∈ J(k)  then
4:          x_1^{k+1} = Fi(X_1^k, .., X_m^{s_m^i(k)})
5:       else
6:          x_1^{k+1} = x_1^{k+1}
7:       end if
8:    end for
9: end for
```

A chaque itération, certains processeurs (ceux qui appartiennent à $J(k)$) calculent une nouvelle itération, les autres nœuds reprennent le résultat de l'itération précédente.Lorsqu'un processeur calcule une nouvelle itération, il prend les dernières données dont il dispose. Par conséquent, bien que ce modèle présente une allure synchrone, celui-ci est parfaitement adapté au cadre asynchrone, puisque tous les processeurs ne se mettent pas à jour simultanément.

Ces algorithmes sont extrêmement performants dans les cas où les données à traiter sont de taille assez conséquente et que les machines sont très hétérogènes avec des liaisons réseaux tout aussi hétérogènes. Ceci permet de palier les problèmes dûs à l'environnement matériel. Les temps de communication sont recouverts par des temps de calculs. Ces algorithmes ne sont donc à utiliser que dans des conditions bien déterminées.

Il faut, par ailleurs, ajouter que dans le cadre le plus général les délais de mise à jour des données ne sont pas bornés, c'est-à-dire qu'il est tout à fait possible que ceux-ci soient très longs. Pour assurer la convergence de tels modèles, deux conditions doivent être vérifiées, la première condition énonce que les composantes du système doivent être infiniment souvent mises à jour ( $\forall i \in 1, ...m$ l'ensemble $k, i \in J(k)$ est infini). La seconde condition consiste à vérifier que les mises à jour des données non locales suivent l'évolution des itérations $lim_{k\to\infty} s_j^i(k) = \infty, \forall i$ et $j \in 1, ...m$ bien que les délais ne soient pas bornés [2].

A partir du modèle des algorithmes IACA, il est possible d'implanter ce type d'algorithme de plusieurs manières différentes.

| **ISCS** :<br>– Les communications sont effectuées à la fin de la phase de calcul.<br>– Présence des périodes d'inactivités dues aux communications et aux calculs liés à la détection de convergence.<br>– Communication et synchronisation par partage de données en mémoire.<br>– Unification de la convergence que celle de leurs pendants séquentiels.<br>– Ces algorithmes sont un peu plus rapides. | **ISCA** :<br>– L'exécution de l'algorithme est, à priori, plus rapide.<br>– Recouvrement des parties de communication par des calculs.<br>– Convergent dès lors que la version séquentielle converge.<br>– les itérations sont toujours synchronisées. | **IACA** :<br>– La désynchronisation des itérations.<br>– Les processeurs ne calculent pas la même itération à un instant t.<br>– Le temps de mise à jour des données est indéterministe.<br>– Il n'y a plus de temps mort entre les itérations.<br>– Adaptable dans un environnement hétérogène et volatile.<br>– Difficulté de détection de convergence (Plus de conditions et de traitements). |
|---|---|---|

TABLE 2.2 – Comparaison entre les trois algorithmes itératifs

# 6 Conclusion

Les algorithmes itératifs sont généralement simple à programmer et à paralléliser ainsi elles sont moins gourmandes en mémoire ce qui est pas le cas pour les algorithmes directes. La synchronisation dans ces algorithmes peut se révéler très coûteuse dans un contexte hétérogène et distant où ces algorithmes sont souvent pénalisés par les communications inter-processeurs, notamment sur grille.

Introduire l'asynchronisme permet de réduire le taux de perte de ces performances dû au fait que les communications sont recouvertes par le calcul et l'absence de synchronisation pénalisante, ce qui provoque une difficulté de détection de convergence dans ces algorithmes asynchrones itératifs, dans le chapitre qui suit on va montrer l'intérêt des algorithmes IACA avec l'outil JACE.

# Chapitre 3

# Résolution de systèmes linéaires de très grande taille

## 1 Introduction

Dans le chapitre précédent, nous avons présenté le principe des algorithmes itératifs asynchrones, nommés IACA, le gain apporté par ce genre d'algorithme en terme de temps de calcul par rapport aux algorithmes synchrones est important.

Dans celui-là, on va passer à la pratique, notamment avec la résolution d'un système linéaire dense et creux, pour le premier cas, nous avons utilisé trois méthodes à savoir la méthode de Jacobi, gradient avec pas optimal et gradient conjugué, pour aller plus loin nous avons essayé d'améliorer la résolution des systèmes linéaires représentés par une matrice creuse par la méthode de Jacobi en utilisant les formats de compression des matrices creuses CSR et Ellpack-Itpack.

L'implémentation et le déploiement de ces méthodes sont réalisés en utilisant l'environnement JACE qui est dédié au calcul parallèle asynchrone.

# 2 Systèmes linéaires

De nombreuses applications en calcul scientifique, telles que la simulation en mécanique des structures ou en dynamique des fluides, se ramènent à la résolution de problèmes matriciels linéaires de grande taille (généralement dits creux), un tel massif système nécessite un temps de calcul laborieux. La résolution de ces systèmes a toujours joué un grand rôle dans la simulation de nombreux problèmes scientifique et industrielle, Un tel calcul présente l'inconvénient de mener à des coûts prohibitifs en temps CPU et en espace mémoire, donc l'optimisation de ces systèmes est capitale.

Ces systèmes linéaires sont généralement représentés sous la forme des matrices $N*N$, généralement creuses, la spécificité de ces matrices nécessite de mettre en œuvre des méthodes et des mécanismes d'optimisation très contraignante.

# 3 Résolution des systèmes linéaires :

## 3.1 Méthodes de relaxation

Soit le système $A*X = b$ avec A matrice définie positive symétrique, les méthodes dites de relaxation sont fondées sur le principe général suivant : décomposé la matrice A sous la forme : $A = M - N$, avec $M \in R^n$, on génère la suite des vecteurs $x^k, k \geq 1$ en résolvant le système :

$$Mx^{k+1} = Nx^k + b \tag{3.1}$$

Si la suite $x^k$ converge, alors sa limite est la solution du système (3,1).

Pour implémenter la méthode de relaxation d'une manière efficace, il faut choisir la matrice $M$ d'une manière à ce qu'elle soit simple à inverser.

Pour la convergence de la suite $x^k$ , on pose $T = M^{-1}N, T^k$ est la k-ième puissance de la matrice T.

**Définition1 :** On dit qu'une matrice $T \in R^{n,n}$ est convergente ssi :

$$\lim_{k\to\infty} T^k = 0$$

**Définition2 :** si la matrice T est convergente alors, quelque soit le vecteur $x^0$ la suite $x^k$ converge.

Dans le choix de la matrice M, il faut s'assurer que la matrice T soit convergente.

**Définition3 :** on appelle rayon spectrale de T (noté spect(T)), la plus grande des valeurs propres en valeur absolue, le système (3,1) converge vers $x^*$ , si et seulement si spect(T) <1.

Quelques exemples classiques des méthodes de relaxation : Jacobi, Gauss-Seidel et SOR...

La méthode de Gauss-Seidel et la méthode SOR ne peuvent être implémenté en parallèle, car elles ne respectent pas le schéma général des algorithmes IACA qui consiste à calculer $x_i^{k+1}$ en fonction de $x_i^k$ seulement.En effet, elles calculent la composante $x_i^{k+1}$ en fonction de $x_i^k$ et les éléments $x_i^{k+1}$ calculés précédemment.

**Remarque :** On note par $(.,.)_N$ est le produit scalaire euclidien sur $R^N$ et $||.||_N$ la norme induite.

### 3.1.1 Méthode de Jacobi

Basé sur la décomposition $A = D - U - L$, avec $D \in R^{n,n}$ matrice diagonale, L triangulaire inférieure stricte et U triangulaire supérieure stricte.

La méthode de Jacobi consiste à prendre : $M = D$ et $N = -(U + L)$ ce qui convient à $T = D^{-1}(U + L)$.

Donc la méthode de Jacobi consiste à résoudre la formule récurrente suivante :

$$x_i^{k+1} = \frac{1}{A_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}) \tag{3.2}$$

Avec $x^0$ donnée.

On considère que la convergence est atteinte, lorsque pour un ($\epsilon$>0 ) donnée, fixé au début du problème on a : $max|x_i^k - x_i^{k+1}| < \epsilon$ .

*Théorème :* Si la matrice A est diagonale strictement dominante i.e. :
$\forall i |A_{ii}| > \sum_{j \neq i} |A_{ij}|$ , alors la méthode de Jacobi converge.

Pour la résolution en parallèle, il faut découper la matrice T en bandes horizontales $T_i$ et les répartir sur l'ensemble des tâches participantes aux calculs.

**Algorithm 6** Les algorithmes IACA

Initialisation de T et $x^0$
K=1
P=nombre de tâches
**Do**
$x_i^{k+1} = T_i(x_{i\to 0}^k, x_{i\to 1}^k, ..., x_i^k, , x_{i\to p-1}^k)$
Envoie $x_{j\to i}^{k+1}$ pour j=0,...,p-1 avec $i \neq j$
Reception $x_{i\to j}^{k+1}$ pour j=0,...,p-1 avec $i \neq j$
$x_i^{k+1} = x_i^k$
Test de convegence global (réception du message stop/continue)
k++
**While** *Stop* message n'est pas reçu

$x_{i\to j}^k$ : représente l'ensemble des composantes qui sont calculées à l'itération k sur la tâche j et dont dépend la tâche i et chaque tâche applique la formule (3,2) en fonction des informations qu'elles possèdent.

## 3.2 Méthodes de Gradient :

L'idée de ce type d'algorithme, est de passer de la résolution d'un système linéaire

$$Ax = b \tag{3.3}$$

En un problème d'optimisation de la forme :

$$min_{x\in V} J(x) \tag{3.4}$$

Avec J une fonction à décrire, définie sur un espace vectorielle V.

### 3.2.1 Méthode de Gradient à pas optimal :

Si on pose

$$J(x) = \frac{1}{2}(x, Ax) - (b, x) \tag{3.5}$$

Pour assurer la condition (3,4) il faut que le gradient $\nabla J(x) = 0$. Or pour la fonction J qu'on a définie $\nabla J(x) = Ax - b$ .
*Règle :* le vecteur $x^*$ est la solution du système linéaire A x=b, si et seulement si $\nabla J(x^*) = 0$ c'est-à-dire $x^*$ réalise le minimum de la fonction J.

Cette règle est à la base de méthodes itératives de type gradient, en effet il faut chercher le vecteur qui minimise la fonction J.

La démarche à suivre pour minimiser $J(x)$, est donc de trouver le $x^*$ la solution du (3,5) est :

- Choisir un vecteur initial $x^0$ . puis pour tous $k \geq 1$ faire :
- Calculer le gradient de $J : \nabla J(x^k) = Ax^k - b = -R$ .
- Trouver une direction de descente $D^k = -\nabla J(x^k)$.
- Calculer $x^{k+1} = x^k + \alpha D^k$. Avec $\alpha \in R$.

Si $\alpha$ est fixé au début du calcul, alors on parle de la méthode de gradient à pas fixe.

*Théorème :* la méthode du gradient à pas fixe converge si et seulement si, $0 < \alpha < \dfrac{2}{\lambda_n}$ , où $\lambda_n$ est la plus grande valeur propre de la matrice A.

Cette méthode (gradient à pas fixe) présente un inconvénient majeur qui réduit beaucoup son utilisation, c'est la connaissance exacte ou approchée du $\alpha$ ce qui est difficile généralement. La méthode de gradient à pas optimal permet de remédier cela, car elle permet de déterminer la valeur optimal du paramètre $\alpha$.

$$\alpha^k = \frac{(R^k, R^k)}{(AR^k, R^k)}$$

Ce rapport est définie si $R^k \neq 0$ et pour $R^k = 0$ on arrête les itérations et $x^k$ est la solution cherchée.

### 3.2.2 Méthode de Gradient Conjugué

C'est une généralisation de la méthode de gradient à pas optimal, le principe de la méthode de gradient conjugué consiste à construire trois suites de vecteurs $U^k, R^k, P^k$ telles que : $U^0 \in R^n, P^0 = R^0 = b - Ax^0$ et pour $k \geq 1$ :

- $R^k$ Le résidu de $x^k$.
- $K_k$ =vect $(R^0, ..., R^{k-1})$=vect $(P^0, ..., P^{k-1}$ ).
- La famille $R^0, ..., R^{k-1}$ forme une base orthogonal de $K_k$ .
- La famille $P^0, ..., R^{k-1}$ est A-conjugué.

**Définition 1 :** soit un entier $k \geq 0$, une famille de vecteur $P^0, ..., P^{k-1}$ est dite ***A-conjugué*** si pour $m, n \in 0, ..., k$ avec $m \neq n$ , on a $(AP^m, P^n)_N = 0$ .

**Définition 2 :** soit un entier $k \geq 0$, une famille de vecteur $R^0, ..., R^{k-1}$ est **orthogonale** si pour $m, n \in 0, ..., k$ avec $m \neq n$ , on a $(R^m, R^n)$=0 c.à.d $R^m * R^n = 0$.

La construction de ces suites est basée sur le procédé d'orthogonalisation du Gram-Shmidt en se basant sur les récurrences suivantes

**Algorithm 7** Algorithme du gradient conjugué

- Choisir $x^0 \in R^N$ , poser $R^0 = b - Ax$ et $P^0 = R^0$
- Choisir une tolérance T.
- Poser k=0.

**While**$(||R^k|| > T)$ **do**

- Calculer le vecteur $Z^k = AP^k$ .
- $s^k = (R^k, R^k)/(Z^k, P^k)$.
- $x^{k+1} = x^k + s^k P^k$.
- $R^{k+1} = R^k - s^k Z^k$.
- $t^k = (R^{k+1}, R^{k+1})/(R^k, R^k)$.
- $P^{k+1} = R^{k+1} + t^k P^k$.
- $k = k + 1$.

**End while**

–

$$x^{k+1} = x^k + s^k P^k. \tag{3.6}$$

–

$$R^{k+1} = R^k - s^k A P^k \tag{3.7}$$

–

$$P^{k+1} = R^{k+1} - t^k P^k \tag{3.8}$$

Où $s^k$ et $t^k$ des réels donnés par les formules suivantes :

$$s^k = \frac{(R^k, R^k)}{(AP^k, P^k)} \text{ et } t^k = -\frac{(R^{k+1}, AP^k)}{(AP^k, P^k)}$$

Le paramètre $s^k$ permet d'orthogonaliser $R^k$ et $R^{k+1}$ ; et $t^k$ assure la A-conjugaison entre $P^k$ et $P^{k+1}$.

# 4 Environnement JACE (Java Asynchronous Computation Environment)

JACE [2] est un environnement de programmation distribuée, permettant le calcul itératif synchrone et asynchrone, il permet de relier un ensemble de machine pour construire une machine virtuelle parallèle, il est écrit entièrement en java donc il bénéficie des caractéristiques de ce langage notamment : il est multithreadé et portable sur toutes les plates-formes de plus :

- Permet la création d'une machine virtuelle parallèle.
- Utilisation des threads comme unité de calcul.
- utilisation des machines avec adresses Ip privées.
- La communication via RMI (Remote Method Invocation).

- Gestion automatique de la convergence.
- Tolérance aux pertes des messages (c.à.d. la volatilité du système en terme de **communication** ) est géré automatiquement par le noyau JACE via le mécanisme des files pour garder les messages en plus les Algorithmes IACA traitent ce genre de volatilité)
- Gestion transparente des mécanismes liés aux itérations asynchrones.

## 4.1 Spécificité de JACE

Les environnements **MPI** et **PVM** sont les plus utilisés pour programmer des applications distribuées parallèles, mais ces environnements ne sont pas appropriés pour implémenter des algorithmes IACA efficaces.

Les environnements comme **MPI** et **PVM** sont monothreadés c.à.d ne support pas l'utilisation des processus léger donc il est impossible de concevoir un algorithme IACA efficace.Pour les environnements multithreadés comme **PM2**, le calcul évolue en parallèle avec la communication dans des threads différents. Le problème est que c'est au programmeur de gérer lui même les threads de communication et de calcul, la synchronisation l'accès au ressources partagées ..., ce qui n'est pas évident car il faut faire très attention aux situations d'interblocages le problème classique de la programmation multithreadée, en plus les threads ne sont pas gérer de la même façon sur les différents systèmes d'exploitation. Par contre JACE : il est portable car écrit entièrement en Java, gère de façon transparente les mécanismes liés à l'asynchronisme : écrasement des messages, détection de la convergence ..., c'est pour ça qu'il est le plus adéquats pour l'implémentation des algorithmes IACA.

## 4.2 Architecture de JACE

JACE est composé de quatre composantes*(Figure(3,1))* : le daemon, les tâches, le spawner ou le distributeur de tâches et la console.

### 4.2.1 Le daemon

Lancer sur toutes les machines choisies, le daemon est le cœur de JACE il tourne en tâche de fond, il contient les objets nécessaires pour la configuration de la machine, en plus il est responsable de routage des messages et gestion des tâches, la synchronisation et la communication entre les nœuds de calcul.

Le démarrage des daemons nécessite une synchronisation entre tous les nœuds, car chaque nœud doit obtenir une référence des objets serveurs de tous les autres nœuds, la localisation des objets distant se fait grâce à RMI. Pour chaque nœud de

la machine virtuel, le daemon est composé d'un ensemble d'objets *JaceRuntime* et d'objets *JaceServer*.

***JaceRuntime*** **:** dont les méthodes sont accessibles depuis tout objet local du programme, c'est la composante responsable de la communication et la localisation des nœuds.

***JaceServer*** **:** dont les méthodes sont accessibles depuis tout nœud distant, en s'exécutant dans un thread séparé, ces méthodes seront appelées par les autres daemons et le distributeur de tâche.

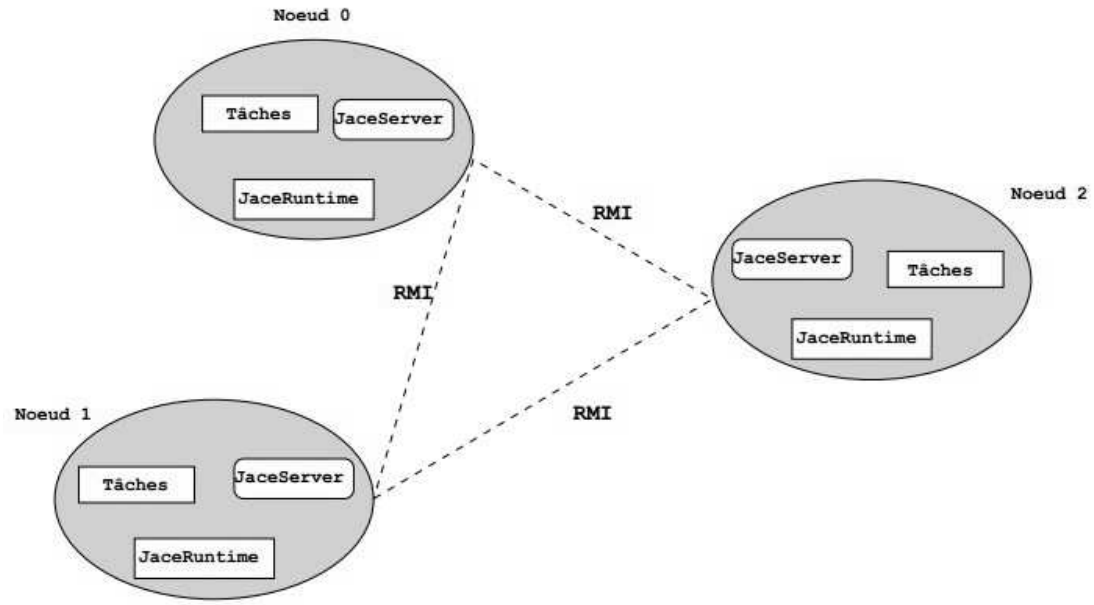


FIGURE 3.1 – Architecture des démons Jace [2]

### 4.2.2 Les tâches

Les tâches dans JACE sont des threads exécutés par le daemon, plusieurs tâches peuvent êtres exécutés par un seul daemon et donc partagent les même ressources système.

Une tâche JACE est un Byte-code java qui encapsule un calcul. Une tâche hérite de la classe **Task**, pour créer une tâche l'utilisateur doit redéfinir deux méthodes (**jaceInitTask()** et **run()**), la première méthodes contient toutes les initialisations nécessaires avant commencer le traitement genre constructeur et dans la deuxième c'est la où le programmeur doit définir son traitement .

Les tâches JACE sont identifiés de 0 à n-1, avec n le nombre total de tâches.

### 4.2.3 Distribution de tâche (spawner)

Elle permet de créer un ensemble de tâches identiques et de les distribuer sur les nœuds disponibles, indépendamment de leurs nombre. Les mécanismes internes du spawner sont masqués, le programmeur doit fournir que le nombre total de tâches .en interne, le spawner envoie des requêtes à l'ensemble de daemon en utilisant RMI, en recevant les requêtes les daemons chargent la tâche à partir d'une URL valide et lorsque la tâche sera chargé l'exécution du programme commence. Les tâches communiquent entres elles via l'identifiant de tâche sans connaitre la position réelle, la localisation est assuré par les daemons.

### 4.2.4 La console JACE

La console est un shell qui permet de configurer la machine virtuel, lancé des tâches, recevoir les informations concernant les tâches, elle peut être lancée à tout moment sur n'importe quelle machine où tourne déjà un daemon.

## 4.3 Communication sous JACE

Pour la communication, JACE utilise le passage de message et RMI*(Figure(3,2))*. Les daemons communiquent directement en utilisant RMI, alors que les tâches communiquent entre elles par envoi et réception des messages. Le choix de RMI est naturel car il permet l'invocation des méthodes sur des machines distantes en plus l'architecture cible est de grille de calcul où les machines sont hétérogènes, le passage de message est implémenté via RMI, mais ceci est transparent à l'utilisateur qui manipule des primitives JACE spéciales pour la communication.

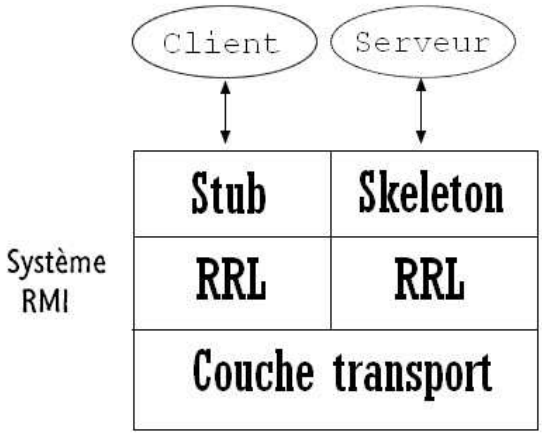


FIGURE 3.2 – Communication sous JACE

## 4.4 JaceLite

JaceLite est une version légère et stable de la bibliothèque JACE permet d'exécuter des applications (SPMD, MIMD) sur des architectures parallèles et distribuées. JaceLite est un environnement très proche du MPI, avec l'utilisation des threads java comme unité de calcul, les classes doivent héritées de la classe *MPITask* pour une application MPI et MasterSlave dans le cas d'une application *Client/Serveur*.

Exemple d'une classe qui hérite du MPITask :

```
import net.mazouzi.core.MPITask;
public class UnivTlemcen extends MPITask {

public void run() {
final int MonNum = MPI.COMM_WORLD.Rank();
final int Taille = MPI.COMM_WORLD.Size();
final String host = MPI.Get_processor_name();

System.out.println("Je suis le processus numéro " +MonNum + "/ " + size
+ ", je m'exécute sur la machine " + host);

MPI.Finalize();
 }
     }
```

### 4.4.1 Principe de fonctionnement

L'exécution du programme précédent se fera par la commande suivante :
**jacerun -np *val* [-pe smp | -m | -machinefile *machines*] *UnivTlemcen***
Tel que val représente le nombre de processus lancé pour l'exécution du programme, -pe smp est indiquée dans le cas d'un environnement local parallèle (multithread) sinon un fichier texte de configuration (machines) est défini pour une exécution distribuée dans un mode cluster. Chaque ligne de ce fichier contient le nom ou l'adresse IP de la machine suivi par le nombre de cœurs attribués : Si aucun nombre n'est spécifié, un seul cœur pour cette machine .
*exemple*

```
zaki-laptop
fayssal-laptop : 4
```

```
192.168.1.101   : 2
poste10         : 2
```

# 5 Mise en œuvre avec JaceLite

Notre but ultime était de réaliser une exécution distribuée des trois algorithmes de résolution des systèmes linéaires : Jacobi, Gradient optimal et Gradient conjugué, afin de sortir avec une comparaison entre ces trois méthodes selon différents critères (le temps, nombre de cœurs ...), pour cela nous avons créé un petit cluster composé de huit machines d'un GO d'espace de la Ram et deux cœurs pour chacune, elles sont reliées par un switch de huit ports (TENDA TEH8005) et un routeur de quatre ports (HUAWEI HG520b) avec un débit de 10/100 Mbps.

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

La matrice de résolution d'un système linéaire pourra être donnée selon différents formats (dense, creuse,...)

Une matrice creuse correspond à une matrice dont la plupart des éléments sont nuls, par opposition des matrices denses dont la plupart des éléments sont différent de zéro.

## 5.1 Résolution d'équations représentées par une matrice dense

Après avoir exposé les trois algorithmes (Jacobi, Gradient optimal et Gradient conjugué) aux mêmes conditions (même matrice, même machines...) nous avons obtenu les résultats mentionnés dans *(Table(3,1))* et *(Figure(3,3))*.

On peut bien remarquer que l'algorithme de Jacobi est le plus performent par rapport aux autres algorithmes spécialement si le nombre de machines est assez important cela est dû à la complexité élevée des algorithmes de type Gradient par contre dans une exécution séquentielle où le nombre de machines est très limité et la taille de la matrice est grande l'algorithme de Gradient optimal est légèrement plus rapide.

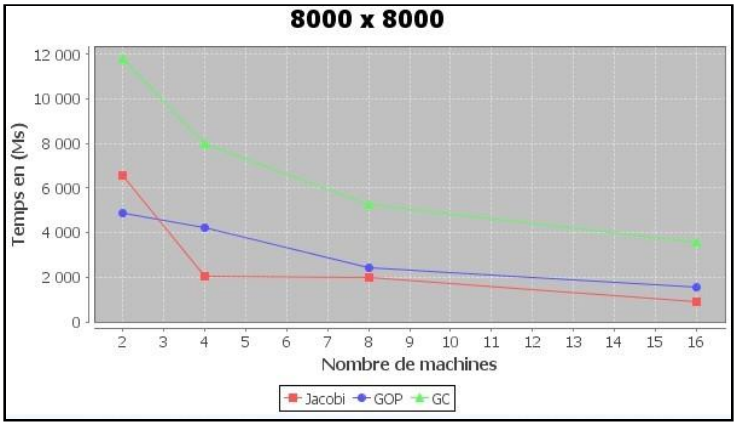


FIGURE 3.3 – Comparaison entre les trois algorithmes

| | *Taille (N)* / *Nb machines* | **80** | **800** | **2000** | **4000** | **8000** |
|---|---|---|---|---|---|---|
| **Jacobi** | **2** | 4(ms) | 158 | 1300 | 1500 | 6588 |
| **G_Opt** | **2** | 246 | 1200 | 2400 | 2400 | 4870 |
| **G_Conj** | **2** | 608 | 2007 | 3360 | 5540 | 11786 |
| **Jacobi** | **4** | 1 | 58 | 362 | 596 | 2026 |
| **G_Opt** | **4** | 87 | 394 | 998 | 1958 | 4221 |
| **G_Conj** | **4** | 188 | 915 | 2068 | 4169 | 8014 |
| **Jacobi** | **8** | 3 | 52 | 252 | 512 | 2000 |
| **G_Opt** | **8** | 63 | 263 | 606 | 1231 | 2411 |
| **G_Conj** | **8** | 119 | 635 | 1397 | 2676 | 5246 |
| **Jacobi** | **16** | 2 | 51 | 227 | 495 | 907 |
| **G_Opt** | **16** | 37 | 203 | 489 | 914 | 1534 |
| **G_Conj** | **16** | 81 | 498 | 1103 | 2047 | 3569 |

TABLE 3.1 – Résultats obtenus sur une matrice N*N

| | *Taille (N)* / *Nb machines* | **800** | **2000** | **4000** | **8000** |
|---|---|---|---|---|---|
| **T de comm** | **2** | 860 | 4852 | 6835 | 10421 |
| **T de calcul** | **2** | 1200 | 2300 | 1958 | 870 |
| **T de comm** | **4** | 1170 | 3226 | 5780 | 11927 |
| **T de calcul** | **4** | 394 | 998 | 1958 | 4221 |
| **T de comm** | **8** | 1357 | 3096 | 4534 | 12847 |
| **T de calcul** | **8** | 263 | 606 | 1231 | 2411 |
| **T de comm** | **16** | 2230 | 5917 | 9490 | 31898 |
| **T de calcul** | **16** | 203 | 489 | 914 | 1534 |

TABLE 3.2 – Temps de communication VS temps de calcul (Gradient optimal)

On constate d'après la table (3,2) que le temps de calcul diminue avec plus de processeurs participants en raison de moins de données par calcul. Évidemment, davantage de processeurs impliqués causeront des communications plus intensives qui entraîneront plus de temps de communication *Figure(3,4))*, ce qui est logique et compatible avec les formules de complexité suivantes :

Si on pose N le nombre de machines et M la taille de la matrice, pour le calcul la complexité dépend de M/N, car la matrice est divisée en bande horizontale suivant le nombre de machines. Pour la communication on envoie ce qu'on a calculé à l'ensemble de machine (une machine envoie au (N-1) autres machines) donc c'est $N * (N - 1) => O(N^2)$.

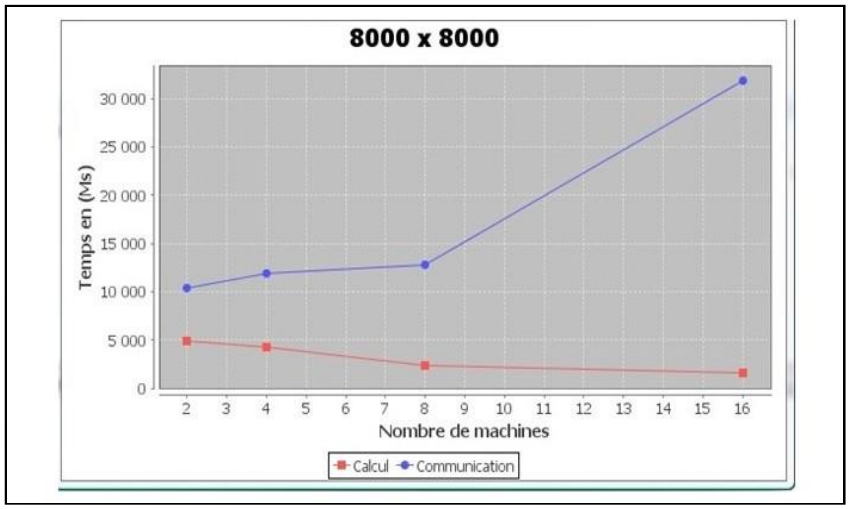


FIGURE 3.4 – Comparaison de temps de calcul et de temps de communication (GOP)

Donc il faut faire des compromis, c.à.d, si on veut gagner en temps de calcul on doit perdre en temps de communication pour des matrices de taille normal, ce qui permet de dire que le temps de communication domine le temps de calcul. Mais dés que la taille de la matrice devient trop important, voire (1 000 000*1 000 000), le temps de calcul devient le dominant, mais faire des calculs sur ce type de matrice et stocker cette quantité d'éléments est très couteux en terme de mémoire.

## 5.2 Résolution d'équations représentées par une matrice creuse

Une matrice creuse contient en général 90% d'éléments nuls, ce qui implique la présence des calculs inutiles genre 0∗0, prenant l'algorithme de Jacobi pour une matrice creuse nous avons obtenu les résultats suivants :

## 5.3 Amélioration et évaluation de performance des résultats

Si on veut stocker et manipuler efficacement des matrices creuses, il faut prendre en compte le creux dans l'algorithme. En fait, il n'est pas nécessaire de stocker des

| *Taille (N)* / *Nb machines* | **80** | **800** | **2000** | **4000** | **8000** |
|---|---|---|---|---|---|
| **2** | 3(ms) | 147 | 748 | 3222 | 4850 |
| **4** | 2 | 68 | 492 | 1066 | 2529 |
| **8** | 3 | 37 | 268 | 734 | 1479 |
| **16** | 1 | 22 | 174 | 636 | 1270 |

TABLE 3.3 – Résultats de l'algorithme de Jacobi pour une matrice creuse

éléments nuls ou d'exécuter des calcules les concernant, c'est pour ça qu'il est nécessaire d'utiliser des algorithmes et des formats qui prennent en compte la structure peu dense de la matrice, ces algorithmes permettent de gagner en espace mémoire et en temps de calcules en permettant de :

- Ne pas enregistrer les éléments nuls.
- Retrouver aisément un élément en connaissant ses cordonnés.
- Exécuter facilement les opérations courantes.

### 5.3.1 Formats de représentation des matrices creuses

#### Format CSR

CSR [16][14] est l'acronyme de *Compress Sparse Row*, c'est le format le plus économe en terme d'espaces mémoire, en fait une matrice creuse est remplacée par 3 tableaux unidimensionnels, si on note n le nombre d'éléments non nuls de la matrice et m le nombre de lignes de M :

- Un premier tableau A de taille n, contenant les éléments non nuls de la matrice rangée de gauche à droite.
- Un deuxième tableau JA, de taille identique à A. il contient le numéro de colonne de chaque élément de A.
- Le troisième tableau IA, de taille m+1, l'élément i de IA contient le rang dans le tableau A du 1$^{er}$ élément de la ligne i de la matrice M, le dernier élément IA (m+1) contient n+1 où n est la taille de A.

*Exemple :* soit la matrice M

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

La compression au format CSR, donnent les tableaux suivants :
A={1,2,3,4,5,6,7,8,9,10,11,12}

JA={1,5,1,2,4,1,3,4,5,3,4,5}
IA= {1, 3, 6, 10, 12, 13}
L'accès à l'élément M (l, c) de la matrice M se fait par :

- Recherche des indices de début et de fin de la ligne $n^0 l$ :

$$\begin{cases} Debut &= IA(l) \\ Fin &= IA(l+1)-1 \end{cases}$$

- Recherche de la valeur c dans **JA** entre **JA** (début) et **JA** (fin) (**recherche obtenue en complexité logarithmique**).
- Si c est trouvé à la position **JA(i)**, alors l'élément **M (l, c)=A(i)**, sinon **M (l, c)=0**.

**Format Ellpack-Itpack**

Le format CSR présente quelques défauts (l'accès est un peu lent) au contraire pour Ellpack-Itpack [16][14], ce format est basé sur l'utilisation de deux tableaux à deux dimensions (1 : n, 1 : m) ou n est nombre de lignes de la matrice d'origine et m est nombre maximum d'éléments non nuls de cette matrice ligne par ligne.

- Le vecteur A, contient les éléments non nuls de la ligne correspondante dans la matrice.
- Le vecteur JA, contient le numéro de la colonne des éléments non nuls de A .zéro pour les éléments nuls.

*Exemple :* soit la matrice M définit précédemment

Le format Ellpack-Itpack donne les deux matrices suivantes :

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 6 & 7 & 8 & 9 \\ 10 & 11 & 0 & 0 \\ 12 & 0 & 0 & 0 \end{pmatrix} \text{ et } JA = \begin{pmatrix} 1 & 5 & 0 & 0 \\ 1 & 2 & 4 & 0 \\ 1 & 3 & 4 & 5 \\ 3 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 \end{pmatrix}$$

L'accès à l'élément M (l, c) de la matrice M devient plus rapide que dans le format précédent, il se fera comme suit :

- Recherche de la valeur de c dans JA(l) (**recherche obtenue en complexité logarithmique**).
- Si c est trouvé à la position JA (l, i), M (l, c)=A (l, i), sinon M (l, c)=0.

### 5.3.2 Parallélisation dans Ellpack-Itpack et CSR

L'hybridation entre ces deux méthodes pourra mener à une compression bien optimisée, on pourra profiter de la facilité de découpage d'une matrice et la rapidité d'accès aux données de cette dernière d'une part et d'autre part de l'optimisation de stockage d'informations, ce qui est une spécificité pour la méthode CSR pour avoir un gain substantiel en espace mémoire.

La matrice est générée au format Ellpack-Itpack et remplie par un processus d'initialisation, ce dernier est chargé de la distribution des données sur les différents P processeurs, cela pourra se faire en découpant les tableaux A et JA du format Ellpack-Itpack en bandes horizontales, où chaque ligne sera allouée à un processeur, alors que chaque colonne sera distribuée sur l'ensemble des processeurs.

Après, chaque bande sera convertie au format CSR à l'instant de l'envoi de données aux processus de calculs où le volume de données à transmettre étant limité.

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix} => A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 6 & 7 & 8 & 9 \\ 10 & 11 & 0 & 0 \\ 12 & 0 & 0 & 0 \end{pmatrix} \text{ et } JA = \begin{pmatrix} 1 & 5 & 0 & 0 \\ 1 & 2 & 4 & 0 \\ 1 & 3 & 4 & 5 \\ 3 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 \end{pmatrix}$$

### 5.3.3 Comparaison des résultats finales

Prenant la même matrice creuse, le tableau ci dessous montre bien la comparaison entre le temps de calcul par la méthode de Jacobi pour une matrice creuse standard et une matrice creuse compréssée (8000X8000) :

| *Nb machines* / *Type de matrice* | **2** | **4** | **8** | **16** |
|---|---|---|---|---|
| **Jacobi** | 4850 | 2529 | 1479 | 1270 |
| **Jacobi_compr** | 15(ms) | 11 | 10 | 10 |

TABLE 3.4 – Résultat de résolution d'une matrice creuse(8000X8000)

La table (3.4) et la figure (3.5) montrent bien qu'il y a une très grande différence en temps de calcul, c'est logique, car l'algorithme utilisant la compression CSR élimine les éléments inutiles (les zéros) ce qui implique un gain d'espace mémoire et de temps de calcul, ce qui nous permet de dire que même si l'algorithme standard de Jacobi est performent par rapport aux autres algorithmes, lorsqu'il s'agit d'une matrice creuse la compression de cette dernière devient une nécessité.

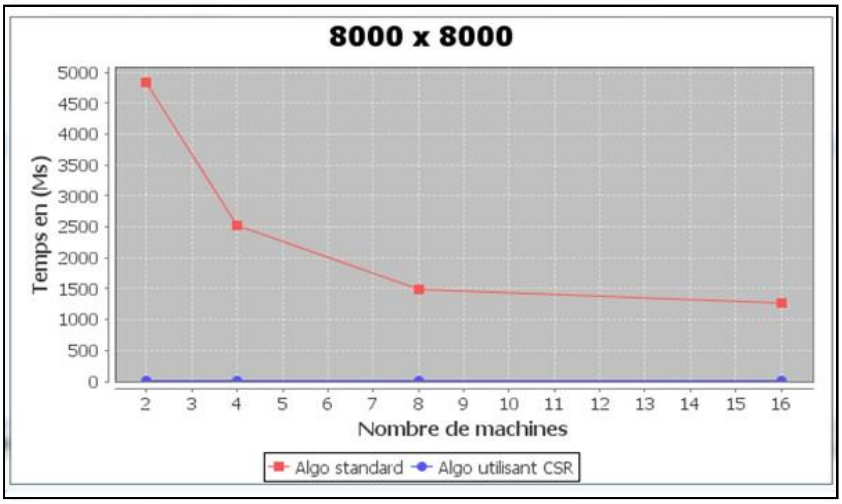


FIGURE 3.5 – Performance de la compréssion CSR

# 6 Conclusion

Dans ce chapitre, on a fait un survol sur quelques méthodes de résolution des systèmes linéaires, nous avons détaillé l'environnement JACE utilisé pour concevoir des applications parallèles et distribuées. Nous avons simulé trois méthodes de résolution avec JACE, le résultat obtenu montre clairement l'intérêt des IACAs pour ce type de méthodes notamment pour la méthode Jacobi qui était la plus satisfaisante, nous avons proposé également une approche de compression basée sur CSR et Ellpack-Itpack qui a montrée son intérêt dans le cas des matrices creuses.

# Conclusion générale

L'objectif de ce travail est de montrer l'intérêt des algorithmes IACA dans les environnements distribués parallèles, pour cette raison nous avons analysé trois algorithmes : Jacobi, Gradient optimal et Gradient conjugué en utilisant l'environnement JACE pour lancer des exécutions parallèles.

Nous nous sommes intéressés également à la compression et à la forme de représentation d'un système linéaire creux avant de le résoudre, cette compression présente l'avantage d'avoir un gain massif non seulement en terme de temps, mais aussi en espace mémoire.

Concernant la volatilité de la communication, elle a été traitée automatiquement par ce type d'algorithme (IACA), pour ce qui concerne la volatilité des nœuds, elle sera laissée comme perspective en utilisant un réseau mobile de type MANET.

Suite au travail réalisé dans le cadre de ce PFE, nous espérons que différentes études pourraient dans l'avenir compléter les résultats obtenus, car si l'application doit être exécutée sur des architectures hétérogènes, il faut prendre en compte le cas de la volatilité des nœuds pour éviter la situation de blocage.

La réalisation de ce travail au sein de l'université Abou Bekr Belkaid nous a permis de connaître de près le fonctionnement des systèmes distribués et l'implémentation des algorithmes itératifs asynchrones 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 JACE et l'établissement d'un super calculateur(cluster de 16 cores), ce cluster nous a permis de réaliser l'exécution des différents algorithmes utilisés.

Ce travail nous a permis aussi d'améliorer nos compétences dans différents domaines comme : La programmation réseau sou Linux (SSH,Shell,..), Java, RMI. Nous

avons également appris plusieurs techniques comme : l'utilisation de Latex, l'utilisation de l'outil graphique jFreeChart,... ce qui nous a permis en final de concrétiser nos connaissances en réseaux et systèmes distribués que nous avons acquis durant nos études académiques au sein de notre université.

Les compétances aquises 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 réseaux et systèmes distribués.

# Bibliographie


[1] Sylvain. Le modèle tcp/ip, mai 2003. http://www.frameip.com/tcpip/.

[2] Kamel MAZOUZI. JACE : un environnement d'exécution distribué pour le calcul itératif asynchrone. PhD thesis, Université de Franche-Comté, 2005.

[3] Greg Hackmann. 802.15 personal area networks. 802.15 Personal Area Networks, page 19 pages, sept 2006.

[4] Vu Duc Trung. La compatibilité entre ieee 802.11b et ieee 802.11g dans les réseaux sans fil, mémoire de maste, 2009.

[5] Khaldoun al agha. Réseaux sans fil et mobiles. Khaldoun al agha, 2004.

[6] Defez al. Le wimax. Université Montpellier II, page 26 pages, Décembre 2004.

[7] Claude Duvallet. les Réseaux sans fil, Architectures et protocoles des réseaux. Claude Duvallet, 2008.

[8] Insa Ngom et Louis Diouf. La radio cognitive, 2008.

[9] David Esprés. protocoles de routage réactifs pour l'optimiosation de bande passante et la garantie de délai dans les réseaux ad hoc mobiles. PhD thesis, Université de Toulouse, 2008.

[10] Sacha Krakowiak. Algorithmique et techniques de base des systèmes répartis, avril 2005. http://krakowiak.developpez.com/cours/systeme-reparti/.

[11] jamal Eddine GHAFFOUR. Sécurité dans les grilles de calcul. PhD thesis, Université de RENNES1, 2004.

[12] Raymond. Introduction aux réseaux tcp ip, août 2008. http://ram-0000.developpez.com/tutoriels/reseau/tcp-ip/.

[13] Fabien HANTZ. Plate-forme pair-à-pair pour l'exécution d'applications exprimables sous la forme de graphes de dép endances de tâches. PhD thesis, UNIVERSITÉ DE FRANCHE-COMTÉ, 2006.

[14] Haiwu HE. ANALYSES AVANCÉES DE LA MÉTHODE HYBRIDE GMRES/LS-ARNOLDI ASYNCHRONE PARALLÈLE ET DISTRIBUÉE POUR LES GRILLES DE CALCUL ET LES SUPERCALCULATEURS. PhD thesis, Université de LILLE, 2005.

[15] Frédéric Robin. Etude d'architectures VLSI numériques parallèles et asynchrones pour la mise en oeuvre de vouveaux algorithmes d'analyse et rendu d'images. PhD thesis, Ecole supérieure des télécommunications, PARIS, 2010.

[16] Olfa Hamdi-Larbi et Zaher Mahjoub Nahid Emad. Vers la détection du meilleur format de compression pour matrice creuse dans un environnement parallèle. Université Montpellier II, page 25 pages, Octobre 2006.

***Glossaire***

**WPAN** : Wireless Personal Area Network.
**WLAN** : Wireless Local Area network.
**WMAN** : Wireless Metropolitan Area Network.
**WWAN** : Wireless Wide Area Network.
**RF** : Radio Frequency.
**Wi-Fi** : Wireless Fidelity.
**Wimax** : World Interoperability for Microwave Access.
**GSM** : Global System for Mobile communication.
**GPRS** : General Packet Radio Service.
**UMTS** : Universal Mobile telecommunication System.
**FHSS** : Frequency Hopping Spread Spectrum.
**DSSS** : Direct Sequence Spread Spectrum.
**IR** : Infra Red.
**ISM** : Industry Scientific Medical.
**PMD** : Physical Medium dependent.
**PLCP** : Physical Layer Convergence Protocol.
**CCA** : Clear Channel Assessment.
**ETSi** : European Telecommunication Standard Institute.
**MBWA** : Mobile Broadband Wireless Access.
**BSS** : Base Station Subsystem.
**BTS** : Base Transceiver Station.
**NSS** : Network Sub-System.
**OSS** : Operation and Maintain Sub-System.
**IMT** : International Mobile telecommunication.
**CDMA** : Code Division Multiple Access.
**UTRAN** : Universal Terrestrial Radio Access Network.
**RNC** : Radio Network Controllers.
**CRN** : Cognitive radio Network.
**DS** : Distributed System.
**MANET** : Mobile Ad hoc Network.
**DSDV** : Destination- Sequenced Distance-Vector.
**OLSR** : Optimized Link State Routing.
**MPRs** : Multi-point relais.
**GSR** : Global State Routing.
**FSR** :Fisheye State Routing.
**WRP** : Wireless Routing Protocol.
**STAR** : Source-Tree Adaptative routing.

**AODV** : Ad hoc On-Demand distance Vector.
**RREQ** : Response Requete .
**TTL** : Time To Leave.
**DSR** : Dynamic Source Routing.
**LMR** :Light-Weight Mobile Routing.
**TORA** :Temporally Ordred Routing Algorithm.
**BSR** :Backup source routing.
**ABR** : Associativity -based Routing.
**ZRP** : Zone Routing Protocol.
**IARP** : Intrazone Routing Protocol.
**IERP** : Interzone Routing Protocol.
**ZHLS** :Zone-Based Hierarchical Link State protocol.
**SHARP** :Hybrid Adaptative Routing Protocol.
**FTP** : File Transfer Protocol.
**BLAS** : Basic Linear Algebra Subprograms.
**LAPACK** : Linear Algebra Package.
**SNA** : Systems Network Architecture.
**DEC** : Digital Equipment Corporation.
**DNA** : Digital Network Architecture.
**ISO** : International Standards Organization.
**OSI** : Open Systems Interconnection.
**TCP** : Transmission Control Protocol.
**IP** : Internet Protocol.
**P2P** : Peer to Peer.
**ISCS** : Itérations Synchrones avec Communication Synchrones.
**ISCA** : Itérations Synchrones avec Communications Asynchrones.
**IACA** : Itération Asynchrones avec Communications Asynchrones.
**JACE** : Java Asynchronous Computation Environment.
**CSR** : Compress Sparse Row.
**SOR** : Successive Over Relaxation.
**RMI** : Remote Method Invocation.
**MPI** : Message Passing Interface.
**PVM** :Parallèle Virtual Machines.
**DDoS** : Distributed Denial-of-Service.
**SPMD** : Single Process Multiple Data.
**MIMD** : Multiple Instruction Multiple data.
**SSH** : Secure Shell.
**JVM** : Java Virtuel Machine.
**GOP** :Gradient Optimal.